Measuring, refining and calibrating speaker and language information extracted from speech

by

Niko Brümmer

Dissertation presented in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering at Stellenbosch University

Promoter: Prof. Johan Adam du Preez
Faculty of Engineering
Department of Electrical & Electronic Engineering

December 2010

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 2010

Abstract

We propose a new methodology, based on *proper scoring rules*, for the evaluation of the goodness of pattern recognizers with probabilistic outputs. The recognizers of interest take an input, known to belong to one of a discrete set of classes, and output a calibrated *likelihood* for each class. This is a generalization of the traditional use of proper scoring rules to evaluate the goodness of probability distributions. A recognizer with outputs in well-calibrated probability distribution form can be applied to make cost-effective Bayes decisions over a range of applications, having different cost functions. A recognizer with likelihood output can additionally be employed for a wide range of prior distributions for the to-be-recognized classes.

We use automatic speaker recognition and automatic spoken language recognition as prototypes of this type of pattern recognizer. The traditional evaluation methods in these fields, as represented by the series of NIST Speaker and Language Recognition Evaluations, evaluate hard decisions made by the recognizers. This makes these recognizers cost-and-prior-dependent. The proposed methodology generalizes that of the NIST evaluations, allowing for the evaluation of recognizers which are intended to be usefully applied over a wide range of applications, having variable priors and costs.

The proposal includes a family of evaluation criteria, where each member of the family is formed by a proper scoring rule. We emphasize two members of this family: (i) A non-strict scoring rule, directly representing error-rate at a given prior. (ii) The strict logarithmic scoring rule which represents information content, or which equivalently represents summarized error-rate, or expected cost, over a wide range of applications.

We further show how to form a family of secondary evaluation criteria, which by contrasting with the primary criteria, form an analysis of the goodness of *calibration* of the recognizer's likelihoods.

Finally, we show how to use the logarithmic scoring rule as an objective function for the discriminative training of fusion and calibration of speaker and language recognizers.

Opsomming

Ons wys hoe om die onsekerheid in die uittree van outomatiese sprekerherkenning- en taalherkenningstelsels voor te stel, te meet, te kalibreer en te optimeer. Dit maak die bestaande tegnologie akkurater, doeltreffender en meer algemeen toepasbaar.

Acknowledgements

I would like to thank all of the following colleagues:

- My supervisor, Johan du Preez, who convinced me to start this Ph.D. in 2004 (which was relatively easy); who bravely accepted the challenge of organizing Odyssey 2008 with me; and who finally convinced me in 2010 to finish (which definitely was not easy).
- My previous employer, Spescom DataVoice, and in particular Wynand Coetzer, who started my speaker recognition career, Viv Crone who sent me to my first NIST evaluation, and Sakkie Nel, for all manner of support and encouragement during this period.
- In equal measure, my present employer, AGNITIO, and in particular Javier Castaño, Emilio Martínez and Marta García Gomar, for sending me to more NIST evaluations and for continued support and encouragement during this period.
- David van Leeuwen, Daniel Ramos and Joaquin Gonzalez-Rodriguez who took a special interest in this work. Your enthusiasm, discussion, collaboration, support and encouragement were vital to the conclusion of this work.
- Patrick Kenny, for showing all of us how to really do speaker recognition; and with whom I had the pleasure to discuss more speaker recognition and probability theory than with anybody else.
- The organizers of the NIST SREs and LREs, and in particular Alvin Martin and George Doddington, not only for defining the original forms of those evaluations, which served as an excellent starting point for this work, but also for always welcoming my discussion of alternative ways to evaluate.
- George Doddington again, whose crazy idea to shift the operating point by two orders of magnitude in SRE 2010 showed us all that our recognizers could do more than we believed possible; and which motivated me to formulate the normalized Bayes error-rates curves described in chapter 7, which cover a wider range of operating points than my APE-curves do.

- All those who have collaborated with me in various NIST evaluations, in particular David van Leeuwen, Lukáš Burget and all the many other BUT researchers, Patrick Kenny, Albert Strasheim, Edward de Villiers and Luis Buera.
- Honza Černocký and Lukáš Burget for their enthusiastic interest in my work and for twice hosting me at BUT in Brno.
- The Johns Hopkins Center for Language and Speech Processing, for hosting me at the 2008 JHU Summer Workshop in Baltimore.
- Edward de Villiers who translated the preface, who helped me tame BibTeX and some of the finer points of technical writing in English.
- Hermann Künzel, who warned me against calibrating speaker recognition scores by fitting them to Procrustes's bed.
- Doug Reynolds, who suggested the idea of normalizing away the evaluation data proportions and instead weighting them with a given synthetic prior; and who provided all manner of support, advice and data through the years.
- Several participants to various NIST Evaluations, who have provided their speaker and language recognition scores to me for experimental purposes.
- Jason Pelecanos, who helped me build my first state-of-the-art speaker recognition system in 2002.
- Richard Brady, who was there when I explored how to normalize proper scoring rules.
- Many others, in the speaker and language recognition research communities, who have shown interest in my work and with whom I had the pleasure to interact during workshops, conferences and evaluations.

Finally, I would like to thank my wife Madrie, whose support defies description.

Contents

List of Figures

List of Tables

Preface

The following is a children's story, in Afrikaans, which illustrates much of the essence of this work. An English translation is provided afterwards. In summary, the protagonist, an intelligent monkey, tries to solve a problem by asking for information from a few less than perfect information sources. He solves the problem by first measuring and then calibrating these sources.

Die aap en die gevaarlike rivier

—Niko Brümmer, 2007—

****

Die aap het deur die bos geswaai, toe kom hy by 'n rivier. Aan die oorkantste oewer sien hy 'n paar bome met die aanloklikste vrugte. Hy is baie honger, maar hy is te versigtig om sommer die rivier oor te steek. Hy kyk eers rond en ontdek 'n padda vir wie hy vra: "Goeiedag padda, gee jy om as ek jou 'n paar vrae vra?"

"Nee", sê die padda.

"Dankie, padda. Is hier enige krokodille in die rivier?"

"Nee", sê die padda.

"Ek is bly om dit te hoor! Is hier ander gevaarlike diere of miskien selfs groot honger visse?"

"Nee", sê die padda.

"A! So dan kan ek veilig oorswem na daardie vrugtebome."

"Nee", sê die padda.

"Hoekom nie? Is daar ander gevare?"

"Nee", sê die padda.

"Nou hoekom kan ek nie oorswem nie?"

"Nee", sê die padda. Die aap kry toe hond se gedagte en vra:

"Antwoord jy altyd *nee* op elke vraag?"

"Nee", sê die padda. Toe vra die aap:

"Is jy 'n padda?"

"Nee", sê die padda.

"Nouja dankie vir jou hulp padda en totsiens", groet die aap beleefd.

"Nee", groet die padda.

****

*D*ie aap soek toe maar verder en kry 'n vis in die vlak water, vir wie hy vra: "Goeiedag vis, kan ek jou maar 'n paar vrae vra?"

"Ja", antwoord die vis vriendelik.

"Is dit veilig om oor die rivier te swem?"

"Ja", sê die vis.

"Maar is hier krokodille in die rivier?", vra die aap net om seker te maak.

"Ja", sê die vis.

"Nou hoe kan dit dan veilig wees?"

"Ja", sê die vis.

"Is jy 'n padda?" vra die uitgeslape aap.

"Ja", sê die vis.

"Jy antwoord altyd *ja* op elke vraag!"

"Ja", erken die vis skamerig.

"Nouja dankie vir jou hulp en totsiens."

"Ja", groet die vis effens teleurgesteld en swem weg.

****

*D*ie aap ondervra volgende 'n waterskilpad: "Goeiedag skilpad, is dit veilig om oor die rivier te swem?"

"Ek weet nie", sê die waterskilpad.

"Maar is hier krokodille?"

"Ek weet nie", sê die waterskilpad.

"Is hier ander gevaarlike goed?"

"Ek weet nie", sê die waterskilpad.

"Is jy 'n waterskilpad?"

"Ek weet nie", sê die waterskilpad.

"Dankie skilpad. Totsiens."

"Ek weet nie", groet die waterskilpad.

****

*D*ie aap is nou moedeloos gesukkel, en hy raak al hoe hongerder. Toe ontdek hy met 'n groot skrik dat daar juis 'n krokodil besig is om hom vanuit die vlak water te bekruip! Hy spring net betyds na veiligheid op 'n boomtak, maar besluit toe om tog 'n geselsie met die krokodil aan te knoop: "Hallo krokodil, hoe gaan dit?"

"Baie goed dankie aap," antwoord die krokodil, "ek is so bly om jou te ontmoet!"

"Hoekom?" vra die aap, "jy wil my seker eet!"

"Nog nooit! Ek eet net plante en so af en toe 'n klein vissie."

"Jok jy, krokodil?"

"Nee aap, ek jok nooit!", sê die krokodil terwyl hy ewe lomp aan 'n graspol kou.

"Is jy 'n krokodil?"

"Nee", sê die krokodil, "ek is eintlik 'n vegetariese likkewaan."

"Is hier enige ander veilige plek om oor die rivier te kom?"

"Nee, nêrens nie", sê die krokodil.

"Is die oorgangplek ver?", vra die aap.

"Ja", sê die krokodil, "al loop jy vir drie dae en drie nagte aaneen, sal jy dit nog nie bereik nie!"

"Is dit stroomop of stroomaf?"

"Ja, stroomop," sê die krokodil, "maar onthou dis baie ver. Swem liewer oor, dis baie makliker en heeltemaal veilig. En ek sal jou help!"

"Nee dankie, krokodil, maar dankie vir jou hulp. Totsiens." Die krokodil grynslag net.

Die aap swaai vinnig stroomaf deur die bome langs die oewer en kry sommer gou 'n tak wat regoor die rivier hang. Hy klouter veilig oor en bereik die vrugtebome, waar hy hom trommeldik vreet en lekker lag vir sy eie slimmigheid.

****

The monkey and the treacherous river

—translated by Edward de Villiers, 2009—

****

The monkey was swinging through the trees when he came to a river. On the opposite bank, he saw a clump of trees bearing the most delicious fruit. He was very hungry, but he was cautious about crossing the river. He looked around and saw a frog. He addressed the frog: "Good day Mr Frog, would you mind if I asked you a few questions?"

"No", answered the frog.

"Thank you, Mr Frog. Are there any crocodiles in this river?"

"No", answered the frog.

"I am very pleased to hear that! Are there any other dangerous animals or maybe even large hungry fish?"

"No", answered the frog.

"Ah! So then I can safely swim over to those fruit trees."

"No", answered the frog.

"Why not? Are there other dangers?"

"No", answered the frog.

"Then why should I not swim across?"

"No", answered the frog. The monkey became suspicious and asked:
"Do you answer *no* to every question?"
"No", answered the frog. Then the monkey asked:
"Are you a frog?"
"No", answered the frog.
The monkey politely took his leave of the frog: "Well, thank you for your help Mr Frog and goodbye".
"No", was the frog's farewell.

****

The monkey searched further and found a fish in the shallow water. He asked the fish, "Good day Mr Fish, may I ask you a few questions?"
"Yes", the fish responded pleasantly.
"Is it safe to swim across the river?"
"Yes", said the fish.
"But are there crocodiles in the river?", asked the monkey, just to be certain.
"Yes", said the fish.
"Then how can it be safe?"
"Yes", said the fish.
"Are you a frog?" asked the shrewd monkey.
"Yes", said the fish.
"You always answer *yes* to every question!"
"Yes", admitted the embarrassed fish.
"Well, thank you for your help and goodbye."
"Yes", responded the fish sadly and swam away.

****

Next, the monkey questioned a turtle: "Good day Mr Turtle, is it safe to swim across the river?"
"I don't know", said the turtle.
"But are there crocodiles in the river?"
"I don't know", said the turtle.
"Are there other dangerous creatures in the river?"
"I don't know", said the turtle.
"Are you a turtle?"
"I don't know", said the turtle.
"Thank you, Mr Turtle. Goodbye."
"I don't know", responded the turtle.

****

The monkey was now fed up and was getting hungrier and hungrier. Just then he saw a crocodile in the shallow water stalking him! He jumped to the safety of a tree branch just in time, but then decided to use the opportunity to pick the crocodile's brain: "Hello Mr Crocodile, how are you?"

"Very well, Mr Monkey," answered the crocodile, "I am so pleased to meet you!"

"Why?" asked the monkey, "I suppose you want to eat me!"

"Certainly not! I only eat plants and occasionally a small fish."

"Surely you're telling a fib Mr Crocodile?"

"No, Mr Monkey, I never lie!", said the crocodile while clumsily chewing on a tuft of grass.

"Are you a crocodile?"

"No", answered the crocodile, "I'm actually a vegetarian monitor lizard."

"Is there any other safe place to cross the river?"

"No, nowhere", said the crocodile.

"Is the crossing far?", asked the monkey.

"Yes", said the crocodile, "even if you walk for three days and three nights, you won't reach it!"

"Is it upstream or downstream?"

"It's upstream," said the crocodile, "but remember it is very far away. Rather swim across here—it will be much easier and it is completely safe. I'll even help you!"

"No thank you, Mr Crocodile, but thank you for your help. Goodbye.".

The crocodile just grinned.

The monkey quickly swung downstream through the trees beside the bank and soon found a branch that grew right over the river. He crossed safely to the other side and reached the fruit trees where he stuffed himself, grinning all the while at his own ingenuity.

Chapter 1

Introduction

The subject of this dissertation is how to evaluate the goodness of a certain class of automatic pattern recognizers. The problem with automatic pattern recognizers is that they are not infallible. They make errors—and some more than others. If one wants to design, improve, sell, buy, or use a pattern recognizer, then it is important to have some understanding of these errors and their implications. In short, the need exists to evaluate the goodness of pattern recognizers.

In this chapter we introduce the evaluation problem, summarize the solutions proposed in this work and mention the response of other researchers to this work.

1.1 Outline of the problem

The experience of the author is in speech processing and specifically in *automatic speaker recognition* and *automatic spoken language recognition.* Standard problems from speaker and language recognition will serve throughout this work as prototypes of the class of pattern recognition of interest. Moreover, this work is grounded in the literature and terminology of the traditional evaluation methods in speaker and language recognition. However, the new evaluation methods proposed here are applicable in a more general context of pattern recognition.

In this section, we outline the class of pattern recognizers of interest and introduce the problem of evaluating them.

1.1.1 The recognizer and its output format

We are interested in pattern recognition problems where an input is given that is known to belong to one of a set of *discrete classes.* The recognizer is expected, broadly speaking, to *recognize* the class of the input. Our running examples throughout this work are:

Speaker detection. A pair of speech segments is given as input. The two classes to be recognized are: (i) the proposition that both segments were spoken by the same speaker, or (ii) the proposition that the segments were spoken by two different speakers.

Language recognition where a single speech segment is given, known to belong to a pre-defined set of language classes.

The recognition result may take different forms. We use speaker detection as an example to discuss output forms. Given a pair of speech segments, the speaker detector may respond in any of the following ways:

1. A categorical statement: The segments *are/are not* of the same speaker.
2. A decision: *Accept/reject* the same-speaker hypothesis as the best decision under the circumstances.
3. A *score*, with the *sense* that larger scores favour the same-speaker hypothesis and smaller scores favour the different-speaker hypothesis. The user may, for example, apply a pre-determined threshold to the score to make an accept/reject decision.
4. A *probability* for the same-speaker hypothesis. This is a specialized form of *calibrated* score.
5. A *likelihood-ratio* for the same-speaker against the different-speaker hypothesis. This is another specialized form of calibrated score.

The main subject of this work is how to evaluate recognizer outputs of the last form. Below we briefly introduce each form. Except for the first, the others will be discussed in more detail in later chapters.

Categorical statements

Form 1, the categorical statement, is a bad idea. If the recognizer gets it wrong it is viewed by the user as a bug. It should only be considered for a very accurate recognizer that has a very low probability of being wrong. We exclude this form from further consideration in this work.

Hard decisions

Form 2, the hard decision, is subtly different from the categorical statement. We assume these decisions are made with the objective of minimizing the risk of the consequence of each decision, where the circumstances define the risk parameters. This will be analysed in detail in chapter 2, where we use Bayes decision theory to quantify the risk via *prior* and *cost function.* Although such hard decisions can be made and understood in a principled way, they limit the recognizer to very restricted use, because the prior and cost are hard-coded into the recognizer. Not only are they hard-coded, they are also hidden from view, so that the user may not understand why the recognizer seems biased towards one or the other decision.

Scores

Form 3, the score, is in practice a very good format. It shifts the responsibility for making decisions from the recognizer to the user and with it gives the user the ability to effectively adjust not only the risk parameters, but also to correct for miss-calibration in the recognizer. It is also very useful in applications where hard decisions are not required but where inputs just need to be sorted to prioritize them for further action.

The speaker detection score represents an important part of the state-of-the-art in speaker recognition and is well supported with evaluation tools such as ROC/DET curves, which we analyse in chapter 7. In general, scores play an important part in pattern recognition and machine learning, where the SVM (support vector machine) classifier is a prototypical example of a recognizer with non-probabilistic scores [1].

For multi-class problems, the raw recognizer outputs are also typically referred to as scores. But in this case it is less clear how to define the *sense* for an uncalibrated score format or how to evaluate the goodness of such scores. Multi-class score evaluation is discussed in chapter 8. The main subject of this work is however the evaluation of *calibrated* scores.

Probabilities

Form 4, the probability, can be used in the same way as a score, but if well-calibrated, can do more. For multi-class pattern recognition, the probability generalizes in a straight-forward and unproblematic way to a *probability distribution.*

To make hard decisions with uncalibrated scores, the user needs to exercise the recognizer to gain some experience of the behaviour of the score and to thus learn where to put the thresholds. In contrast, given a well-calibrated probability distribution, the user can calculate risk expectation in a straight-forward way and thus make minimum risk decisions with no previous experience of the recognizer. The probabilistic output is in general more widely applicable than the hard decision or the score. This idea is not new in pattern recognition, see for example [2], where a method was proposed to transform uncalibrated SVM scores into calibrated probabilities.

Although this form of output is not our final goal in this work, it is very closely related to that goal and we will use the probability distribution format extensively as an analysis tool. This format and its evaluation are analysed in detail in chapters 2 and 3.

Likelihoods

Form 5, the *likelihood-ratio*, can be more appropriate than the probability form for certain types of two-class problems. Likewise, for some multi-class problems, an output in the form of a set of class likelihoods is more appropriate than a probability distribution over the classes. This form of recognizer output is introduced in chapter 4.

Consider a spoken language recognizer, built to distinguish between the eleven official languages in South Africa. The prior probability distribution over these languages should be strongly dependent on where in the country the speech is collected. A recognizer that gives its output as a set of likelihoods, one for each of the eleven languages, would be more generally useful than one which gives a posterior distribution over these languages. The likelihoods supplied by the recognizer can be combined in a straight-forward way with a prior supplied by the user[1] via Bayes' rule to produce the posterior.

The situation for speaker detection is similar. In an access control application, the prior for the same-speaker hypothesis is high, while in a database search, or monitoring application, the prior may be very low. The speaker detector would be most generally useful if its output is in prior-independent, likelihood-ratio form, so that the user may specify the applicable prior as needed.

The proviso for such use, as in the case of the probability form, is that the likelihoods should be well-calibrated. This naturally raises some questions: What is the definition of well-calibrated? How does one tell by looking at a likelihood or a probability that it is well-calibrated? Is calibration all that matters?

Calibration of pattern recognizer outputs is an important part of the subject matter of this dissertation and will be discussed throughout. Below, we give a brief intuitive introduction.

1.1.2 Accuracy, precision and calibration

Consider the example of two weather forecast sources: A predicts a maximum of 40 degrees for tomorrow and B predicts 10 degrees. The next day, the temperature barely reaches 5 degrees Celsius. Which forecaster was better? The second appears to be closer to the actual temperature. However if we observe both forecast sources for a few days, comparing their predictions to the actual temperature, it becomes clear that A uses Fahrenheit and B uses Celsius, so that A (at about 4.5°C) was in fact more accurate on the day in question. If Celsius is the agreed standard, then source A is not well-calibrated, even though (after re-calibration to the standard scale) it may be more accurate than source B.

[1]or installer, if the user prefers not to have to understand what a prior is

An important lesson to remember from this example is that the calibration problem of A could not be deduced by looking at a single forecast. (The discrepancy between 40 and 5 was so big that the misunderstanding may have been obvious. But if the temperature is around -40 where the two scales intersect, then one would indeed need to observe the source for several days to see whether Fahrenheit or Celsius gives a better fit to the actual temperature.)

Calibration and accuracy are both important. The same applies to probabilistic forecasts or to probabilistic pattern recognition outputs. Below we revisit the weather forecasting example to introduce *precision*, which makes those forecasts probabilistic.

Precision

The temperature forecasts above were *point estimates*. If the forecaster wants to acknowledge the uncertainty in the forecast, it may be stated, for example: 'The maximum temperature tomorrow is estimated to be $t \pm \delta$'. This is roughly equivalently to an explicit probabilistic forecast of the form: 'The probability distribution for the maximum temperature tomorrow is normal with mean t and variance σ^2.' The precision of the forecast increases as δ, or σ, is made smaller.

Here t can still be judged as before by the criteria of calibration and accuracy, but δ or σ is purely a matter of calibration. An *inaccurate, precise* forecast source, or conversely, an *accurate, imprecise* source can both be considered to be badly calibrated.

Precision is built into categorical probability distributions

We used the temperature forecast example to introduce precision, but in the type of recognizer output that is discussed in this work, the precision is not specified as a separate parameter. We are interested in recognizing discrete classes, not a continuous value like temperature.

Consider therefore the example of a weather forecast for the discrete events of *rain*, or *no rain*, tomorrow. A probabilistic forecast is now just a *single* number, the probability, p, for rain. This number indicates (i) which of the two events is favoured ($p > 0.5$ favours rain) and (ii) the degree of precision of the forecast. The maximum precision is when $p = 0$ or $p = 1$ and the minimum precision is at $p = 0.5$, where neither event is favoured. In pattern recognition, when there are more than two classes, a categorical probability distribution over the classes includes the degree of precision in a similar way.

In the case of the temperature prediction, evaluating the quality of the forecasts is done in the obvious way, because the true temperature is revealed every day and one can do a straight-forward comparison between predicted and actual temperature. In the case of the probability for rain, no 'true' probability is revealed the next day: it either rains or it doesn't. How does one judge p? This is the basic question that is addressed in this work.

1.2 Summary of solutions

In this work, we propose and analyse the use of *proper scoring rules* for the evaluation of probabilistic pattern recognition outputs.

Background

Proper scoring rules were known at least since 1950, see [3] where the Brier score was proposed for evaluating probabilistic weather forecasts. They are now well-known in statistics [4, 5, 6, 7, 8]. Their use in pattern recognition and machine learning [9, 10, 11, 1] and speech recognition [12] is mostly limited to implicit application of the logarithmic scoring rule in discriminative training. In [13] a family of proper scoring rules is explicitly associated with boosting and other discriminative training techniques. In [14], the logarithmic scoring rule is employed as an evaluation criterion for Gaussian process recognizers, although it is not identified as a proper scoring rule. The logarithmic scoring rule has been used in the guise of normalized cross-entropy (NCE), to evaluate confidence in speech recognition [15].

Introduction of proper scoring rules to speaker and language recognition

Proper scoring rules as an evaluation tool were introduced to the speaker recognition field by the author in 2004 [16], followed by a journal paper in 2006 [17] and a book chapter [18], co-authored with David van Leeuwen. This proposal was generalized to the multi-class case by the author in 2006 [19] for language recognition and again, in 2010 for more general multi-class speaker recognition problems [20].

Taking the proposed evaluation criterion a step further, the author employed it as a discriminative training criterion for speaker recognition, as described in the journal paper [21], and also for language recognition as described in [22, 23, 24].

This dissertation is a compendium of the material in the above publications, interleaved with detailed analysis of traditional speaker and language recognition evaluation methods. There are two, as yet unpublished, proofs in appendices C and D.

The remainder of this section is a summary of the contents and structure of this dissertation.

1.2.1 The NIST Speaker and Language Recognition Evaluations

We use as a starting point for our proposed evaluation methodology, the series of NIST Speaker Recognition Evaluations[2] (SREs) and the NIST Language Recognition Evaluations[3] (LREs), which have served as a significant driving force for the state-of-the-art in these fields in the last decade. For recent publications describing them, see [25, 26]. The author has participated in 7 SREs and 3 LREs between the years 2000 and 2010.

In both SRE and LRE series, the evaluations are formed by exercising the to-be-evaluated recognizers on a supervised evaluation database. The recognizers submitted to evaluation are of form 3, which compute *scores*. The scores themselves are analysed with secondary evaluation criteria, such as DET-curves, but *hard decisions* of form 2 are also required to be explicitly submitted. These hard decisions are evaluated with the *primary* evaluation criterion, known as the *detection cost function*, which evaluates the goodness of the decisions as minimum-expected-cost Bayes decisions. We analyse Bayes decisions in detail in chapter 2 and discuss the specific forms used in the SREs and LREs, respectively in chapters 7 and 8.

1.2.2 Evaluation by Bayes decision

NIST's primary criteria in the SREs and LREs ask the recognizer for *hard decisions*. We show this generalizes in a straight-forward way to a recipe for evaluating *probability distributions* output by the recognizer. This is done by simply shifting the responsibility of making hard decisions from the evaluee to the evaluator and proceeding with evaluation as before.

For a given trial, the evaluee outputs a (posterior) probability distribution for the to-be-recognized classes and the evaluator makes the Bayes decision that minimizes the expected value of the agreed-upon decision cost function, where the expectation is taken w.r.t. the submitted probability distribution. These Bayes decisions are then evaluated just like the hard decisions were evaluated before, with the same decision cost function.

However, as mentioned above, we really would like the recognizer to output *likelihoods*, rather than posteriors. This is solved by also shifting the responsibility of applying Bayes' rule from the evaluee to the evaluator. For a given trial, the recognizer outputs a likelihood for each class. Then the evaluator: (i) applies Bayes' rule, using an agreed-upon prior, to get a posterior probability distribution over the classes, (ii) makes the Bayes decision, and (iii) evaluates the decision, just as before.

If the evaluee makes hard decisions for submission to the traditional SRE or LRE by applying steps (i) and (ii) to go from likelihood via posterior to

[2]See `http://www.itl.nist.gov/iad/mig//tests/sre/`.

[3]See `http://www.itl.nist.gov/iad/mig//tests/lre/`.

decision, then the evaluation result of the new proposal and the traditional recipe would be identical, so we seem not to have gained very much.

However, the evaluator can now go a step further. The evaluator can apply more than one prior and perhaps also more than one cost function to the submitted class likelihoods for each trial. In this way, the evaluator can sweep the operating point and exercise the decision-making ability of the submitted likelihoods over a range of *applications*, where each application is represented by a prior and a cost function.

The result of such an operating point sweep may be represented graphically to show the evaluation criterion at every evaluated operating point. Alternatively, the evaluation criterion may be summed or integrated over the sweep to provide an application-spanning, scalar, summary evaluation criterion.

This evaluation methodology is treated in depth in the rest of this work. We analyse Bayes decisions in chapter 2, discuss the recognizer's format for submitting likelihoods in chapter 4 and motivate the recipe for empirical evaluation over a given supervised evaluation database in chapters 5 and 6.

One of our main results is that the integration over operating points, to produce a scalar summary, can be achieved analytically by using a variety of *proper scoring rules*. In chapter 7 we discuss the choice of proper scoring rules for two-class problems and show that the logarithmic rule is a sensible default choice. In chapter 8 we show that the logarithmic rule has this same interpretation for the multi-class case.

1.2.3 Evaluation by proper scoring rule

Evaluation by Bayes decision as described above and *evaluation by proper scoring rule* are two sides of the same coin. For any cost function, the cost of the associated Bayes decision forms a proper scoring rule. Conversely, every proper scoring rule can be interpreted as the cost of a Bayes decision made with some cost function. We discuss proper scoring rules in detail in chapter 3.

Proper scoring rules come in two flavours, *strict* and *non-strict*. The proper scoring rules induced by the above-described Bayes decision evaluation recipe are of the non-strict variety. In other words, non-strict proper scoring rules are closely associated with pattern recognition applications that require hard discrete decisions. The strict proper scoring rules, which form better evaluators for probabilistic pattern recognition outputs (or probabilistic weather forecasts) are not as obviously associated with applications. However, we show in this work how integration over a continuous range of applications forms families of strict proper scoring rules. This interpretation is introduced in chapter 3 and revisited in chapters 6, 7 and 8.

In particular, in this work and in our above-mentioned publications, we single out and propose the logarithmic scoring rule as a general purpose evaluation tool with desirable properties. See section 1.3 for the response of other researchers to this proposal.

1.2.4 Generalized information theory

Evaluating the goodness of a probability distribution by proper scoring rule is equivalent to measuring the information content in the probability distribution.

In the pattern recognition context, the prior probability distribution can be interpreted as the uncertain state of knowledge about the to-be-recognized class of the input, *before* processing the input. The recognizer extracts from the input some information about the class of the input and delivers that information to its user in the form of class likelihoods. The user can then combine the prior information with the information in the likelihoods, using Bayes' rule, to produce the posterior distribution. The recognizer is useful in this role, provided its output is *well-calibrated*, in which case the posterior will on average contain more information (be less uncertain) about the identity of the to-be-recognized class.

In the case of the logarithmic scoring rule, the above concept of uncertainty becomes Shannon's entropy [27, 28], which then serves to quantify the change in information content between prior and posterior. The decrease in entropy (uncertainty) between prior and posterior equals the information that was gained by processing the input through the recognizer.

However, as we explain in chapter 2, any proper scoring rule induces a generalized entropy, which obeys the same key inequalities[4] as Shannon's entropy. This interpretation shows that information can be viewed in a more general sense than that defined by Shannon's entropy. Generalized information content can be interpreted as the cost-effective decision-making ability of the data. If the likelihoods or posteriors are well-calibrated, then on average they make better (lower cost) decisions than those made with the prior alone.

This generalized information-theoretic interpretation helps to give a deeper understanding of the proposed evaluation methodology and in particular helps to define the somewhat slippery concept of calibration.

1.2.5 Calibration analysis

As noted in the temperature forecasting example above, it is not obvious how to define calibration of a probability distribution, if no 'true' or reference probability distribution is given for comparison. We put 'true' between quotes, because there is no such thing. The posterior probability distribution, for the class given the input data, is dependent on some generative probability model which jointly models the classes and the data. The posterior changes as different models are assumed. In the context of evaluation, no such model is given.

It is not the role of the evaluator to model the data, or indeed to build a reference recognizer against which the evaluated recognizer can be compared.

[4](i) Cross-entropy is lower bounded by entropy. (ii) Average posterior entropy is upper bounded by prior entropy. (iii) The data processing inequality.

This would make evaluation extremely difficult and would in any case give a non-objective and ill-defined evaluation criterion.

Calibration analysis is achieved by a subtle shift in thinking. Given some evaluation criterion, the evaluator can ask by how much this criterion can be improved by re-calibrating the submitted likelihoods. The re-calibration transformations are strongly constrained, so that the re-calibration improvement is not overly optimistic and represents a result that the evaluee could have reasonably obtained in the original submission.

The NIST SRE series uses exactly this principle to effect a calibration analysis. The primary evaluation criterion, which evaluates submitted hard decisions, is known as DCF. A secondary evaluation criterion, minDCF, is computed from the submitted scores, by finding the score decision threshold that minimizes DCF. The discrepancy between DCF and minDCF indicates the quality of calibration of the original submission. We generalize this recipe to work with submitted likelihoods, rather than with hard decisions. Calibration analysis is discussed in chapters 2, 5, 7 and 8.

1.2.6 Discriminative training

There is a close connection between evaluation and discriminative training. If one adjusts parameters of the recognizer to optimize an evaluation criterion, that is discriminative training. Bishop [1] defines a *discriminative model* as a pattern recognizer that outputs posterior probability distributions, which agrees with the class of recognizer subject to our evaluation methods.

To do discriminative training, one needs the following ingredients: (i) an objective function, (ii) a set of parameters to adjust, (iii) partial derivatives of the objective function w.r.t. the adjustable parameters and (iv) an optimization algorithm. In chapter 8, we show how to apply this recipe for discriminative training of the fusion and calibration of both speaker and language recognition systems. We choose the logarithmic scoring rule, which when applied to an affine fusion and calibration transformation, reduces the training problem to the well-known logistic regression, for which efficient optimization algorithms are known in the literature.

Our logistic regression algorithm differs from the standard approach in that we train the pattern recognizer to output likelihoods, rather than posterior probabilities. This is explained in chapter 8. This makes our implementation especially suitable for use in speaker and language recognition and we have used it in our own submissions to NIST SREs and LREs, from 2005 to the present. This implementation was made publicly available as a toolkit, see the response in section 1.3 below.

1.2.7 Comment on experiments

In this dissertation, we do not exhaustively reproduce the experimental results of our above-mentioned publications. We prefer to concentrate on theoretical analysis of the proposed evaluation methodology. We do present selected experimental results as examples throughout this work. For further experimental results, the reader is referred to our publications[5] and also to the experiments others performed by using our discriminative training toolkit, as listed below in section 1.3.

1.3 Response

The author's proposals and work in this field have led to various responses, mainly from the speaker and language recognition research community, of which the most important are listed here.

Forensic speaker recognition. In response to the author's publications [16, 17, 18], which proposed the logarithmic scoring rule as an evaluation criterion, it has been advocated by Daniel Ramos for use in forensic speaker recognition in his Ph.D. dissertation [29] and related publications [30, 31, 32, 33, 34]. Citations by other authors in forensic speaker recognition include [35, 36, 37]. The logarithmic cost function is also proposed in [38] for use in forensic glass fragment analysis.

NIST SRE and LRE. The logarithmic evaluation criterion proposed in [17] was included as an alternative evaluation criterion in both of NIST's Speaker and Language Recognition Evaluations. See the evaluation plans[6] for SREs 2006, 2008 and 2010 and LREs 2007 and 2009.

Discriminative training. The publicly available **FoCal Toolkit**,[7] our implementation for discriminative training for fusion and calibration, using the logarithmic cost function, has been used by many research laboratories in their submissions for NIST's SRE and LRE as well as for other purposes. See, for speaker recognition [39, 40, 41, 42, 43, 44, 45, 46], for language recognition [47, 48, 49, 50, 51], and also for automatic emotion recognition [52, 53].

Research workshops. The author was invited to participate in two extended research workshops to work on discriminative training in speaker recognition. These were the JHU 2008 CLSP Summer Workshop in Baltimore,

[5]available online at `http://niko.brummer.googlepages.com`

[6]The evaluation plans for SRE are available at `http://www.itl.nist.gov/iad/mig//tests/sre/` and for LRE at `http://www.itl.nist.gov/iad/mig//tests/lre/`.

[7]The FoCal Toolkit, coded in MATLAB, is freely available at: `http://focaltoolkit.googlepages.com`.

MD. USA [54] and the 2010 BOSARIS Workshop hosted by the Speech Group at the Brno University of Technology.

Odyssey 2008. The author and his supervisor, Johan du Preez, jointly organized Odyssey 2008: The Speaker and Language Recognition Workshop in Stellenbosh in January 2008.

Evalita 2009. The author was involved in the organizing of the Evalita 2009 speaker recognition evaluation track, and in particular, implemented all of the evaluation procedures. See [55].

Research stay. Daniel Ramos spent three months in 2006 in Stellenbosch, collaborating with the author as part of his Ph.D. study.

Seminar. The author was invited by the ATVS Group at the Universidad Autónoma de Madrid to give a seminar talk on this subject in February 2008.[8]

1.4 Electronic version

For readers who prefer an electronic version of this document, with hot-linked cross-references, see:
`http://dl.dropbox.com/u/7377794/phd/msli.pdf`

[8]The slides are available at `http://arantxa.ii.uam.es/~jms/seminarios_doctorado/abstracts2007-2008/20070226NBrummer.html`.

Chapter 2

From Decision Theory to Information Theory

In this chapter we show in detail how the theory of optimal Bayes decisions, as applied to the pattern recognition problems of interest, leads to a generalized information theory. We work from the assumption that pattern recognizers are useful only in as far as they can be used to make cost-effective decisions in the face of uncertainty. We show that the expected cost, or risk, associated with such decisions forms a natural generalization of Shannon's well-known entropy [27]. This generalized entropy provides a useful and intuitively satisfying way to quantify the amount of relevant information that a recognizer can extract from its input and deliver to the user.

This decision-and-information-theoretic analysis forms the basis for deriving, motivating and understanding the evaluation methodology for assessing the goodness of pattern recognizers that is proposed in this work. It also serves to establish the vocabulary and notation in which everything that follows will be expressed.

This chapter is largely a re-interpretation, in terms of pattern recognition, of material in M.H. DeGroot's book [56] on Bayes decisions, and of a technical report [57] by A.P. Dawid, which explored the generalized information-theoretic interpretation of proper scoring rules. We also rely on the book [28] by Cover and Thomas as a standard reference on information theory.

Original ideas in this chapter include: (i) The device, introduced in section 2.2.2, which we call the *observer*. It makes explicit the fact that all the probability distributions that we use in this work are conditional on some probability model. We do not work with the abstraction that there is an underlying 'true' probability distribution that produced the data. Our probability distributions quantify what some 'observer' knows or assumes about the data. (ii) The conclusion in section 2.3.3 that the optimal output, the observer's posterior distribution, is the *most processed* form of the relevant information in the data, even though the data processing inequality shows that all processing can only reduce and never increase the amount of relevant information.

This chapter is organized as follows: In section 2.1 we show how the expected cost of Bayes decisions forms a generalized entropy measure. In section 2.2 we apply generalized entropy to define our primary criterion for judging the usefulness of pattern recognizers. In section 2.3 we examine how *functions*, which are the basic building blocks of recognizers, affect information content and which functions form optimal recognizers. Finally, in section 2.4, we introduce our secondary evaluation criterion, called calibration.

2.1 Bayes decisions, proper scoring rules and entropy

This work is about *extracting useful information from speech.* The setting is the following:

- An input, x, is given to a pattern recognizer. In our context of interest, x represents one or more utterances of speech, or processed versions of these utterances such as sequences of feature vectors.

- There is a set of N contrastive propositions, each of which makes a statement about the input x, but exactly one of which is true. In the canonical speaker recognition problem $N = 2$ and in the canonical language recognition problem $N \geq 2$.

- Before processing x we don't know which proposition is true, but this uncertainty is quantified by a given *prior probability distribution.* The information in the prior is independent of the information in the speech. In this work, we are interested only in extracting information from the speech and not from other sources. We will therefore consider the prior (i.e. information from other sources) as given. We cannot ignore the prior, because the amount of information that can be extracted from the speech is dependent on the prior: If there is no prior uncertainty, then no further relevant information can be extracted from the speech. More generally, the amount of relevant information than can be extracted from the speech cannot exceed the prior uncertainty. We therefore regard the prior as a given parameter and analyse how the amount of information that is extracted from speech varies as a function of the prior.

- The given input speech, x, is processed by a speaker or language recognizer in order to increase the information (or reduce the uncertainty) about which proposition is true. The recognizer's output is in the form of a *posterior* probability distribution for the unknown proposition.

- By using the recognizer, we can potentially gain some benefit, or equivalently avoid some cost, provided the recognizer is accurate enough.

In this section we show that the expected benefit (or cost) incurred when using the posterior to make recognition decisions is a general way to quantify information and uncertainty. Shannon's information measure is a special case, induced by choosing a logarithmic cost function. Below we discuss these points in more detail.

The reader may want to look ahead at figure 2.3, which summarizes the agenda of this section. The notation in that figure will be defined in what follows.

2.1.1 The set of propositions

Let the set of N contrastive propositions be denoted $\mathbf{\Theta}_N = \{\theta_1, \theta_2, \dots, \theta_N\}$. Examples are:

- In the canonical speaker recognition problem a pair of speech utterances is given. There are two contrastive propositions, θ_1: *The utterances were spoken by the same speaker*; and θ_2: *The utterances were spoken by two different speakers.*
- In the canonical language recognition problem, one speech utterance is given, which is known to belong to one of $N \geq 2$ languages. Hence, there are N contrastive propositions, θ_1: *The utterance is in language* 1; θ_2: *The utterance is in language* 2; etc.

As is customary in pattern recognition, we shall also use the terminology that input x is in *class* i, if proposition θ_i is true of x.

2.1.2 Probability distributions over the propositions

Let $\mathbb{P}_N$ denote the set (simplex) in which categorical probability distributions for $\theta \in \{\theta_1, \theta_2, \dots, \theta_N\}$ live. If $\mathbf{q} = (q_1, q_2, \dots, q_N) \in \mathbb{P}_N$, then $q_i \geq 0$, $\sum_{i=1}^N q_i = 1$, and $q_i = P(\theta_i|\mathbf{q})$ denotes *the probability for*[1] *proposition* θ_i. When recognizing the propositions from speech input, the following probability distributions are of primary interest:

- The *prior* probability distribution for θ is denoted: $\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_N) \in \mathbb{P}_N$, the components of which are the prior probabilities:

$$\pi_i = P(\theta_i|\boldsymbol{\pi})\,. \tag{2.1}$$

[1]We follow the recommendation of Jaynes [7] to use the preposition *for*, rather than *of*. The probability is not a property *of* the event. The probability is given by the distribution $\mathbf{q}$ and different distributions give different probabilities for the same event.

- The speaker or language recognizer is represented as a function, $\mathcal{R}$, which maps the input, x, and the prior to a *posterior.* This mapping is denoted $\mathbf{r} = \mathcal{R}(x, \boldsymbol{\pi})$, where $\mathbf{r} = (r_1, r_2, \ldots, r_N) \in \mathbb{P}_N$ is the *posterior probability distribution*, the components of which are the posterior probabilities:

$$r_i = P(\theta_i|\mathbf{r}) = P(\theta_i|x, \boldsymbol{\pi}, \mathcal{R})\,. \tag{2.2}$$

2.1.3 Cost functions

Next, we employ the posterior, $\mathbf{r}$, to do some useful work, namely to make decisions. We start by introducing cost functions, which quantify the consequences of decisions. Then we introduce Bayes decisions, which optimize expected consequences. Finally, we analyse the properties of the expected consequences of Bayes decisions to show that such expected consequences form a generalized measure of uncertainty (or entropy), which forms a calculus of information very similar in form to Shannon's information theory.

We consider decisions where an *action*, a, from some set of actions $\mathcal{A}$ is chosen. The *outcome* of a decision, a, is the pair, (a, θ), where θ is the proposition that is really true. The *consequence* of the decision is $C(a|\theta)$, where C is a *cost function* which maps outcomes to the real line. The cost function provides a preferential ordering of outcomes—outcomes with low cost are deemed better than outcomes with high cost.

Some examples may help to facilitate understanding.

Cost function examples

Perhaps the most natural form of cost function identifies the action set with the proposition set: $\mathcal{A} = \boldsymbol{\Theta}_N$. The function $C(\theta_j|\theta_i)$ is interpreted as the cost of recognizing proposition θ_j, when θ_i is really true. The most familiar example of this form is the *zero-one cost*, C_{err}, expressed in terms of Kronecker delta, as $C_{\text{err}}(\theta_j|\theta_i) = 1 - \delta_{ij}$. Here the outcome (θ_j, θ_i) when $i \neq j$ is deemed a recognition *error* and is penalized with a cost of 1, while the outcome (θ_i, θ_i) is deemed a correct decision, which has zero cost.

However, the Bayes decision framework accommodates more general cost functions, where the number of elements of the action set, $\mathcal{A}$, can be different from the number of propositions. Consider for example an automatic language recognizer that pre-processes speech inputs in order to refer them to suitable human agents for further processing. Assume the inputs may be in any of N different languages (i.e. N propositions) and there are M different agents, each with different language skills. In a small facility, with M multilingual agents, it is possible that $M < N$. In a big facility, with multiple agents per language, one could have $M > N$. Here the decision, a, takes the form of assigning one of the M agents to the speech input. The cost function could penalize choosing an agent that is not fluent in the true language of the input. Moreover, it could give a larger penalty for choosing an agent that cannot even recognize which language it is. Our point here is that applications of a recognizer can require a rich variety of cost functions that are significantly different from the simple zero-one cost.

So far we have considered only *hard* decisions, where $\mathcal{A}$ has a finite number of elements, but we can also accommodate continuous-valued decisions. In fact, in this work we are particularly interested in *soft decisions*, in the form of probability distributions, where $\mathcal{A} = \mathbb{P}_N$. The canonical example is the *logarithmic cost*, $C_{\log} : \mathbb{P}_N \times \mathbf{\Theta}_N \mapsto [0, +\infty]$, where $C_{\log}(\mathbf{q}|\theta_i) = -\log q_i$.

Cost function properties

We could alternatively and equivalently order outcomes with a *utility* function, with the opposite sense, where higher values are better. In this work, we choose to use cost rather than utility for two reasons: Expected cost generalizes error-rates, which are familiar in pattern recognition. Moreover, expected cost of Bayes decisions generalizes information-theoretic cross-entropy.

It will be shown later that we can assume without loss of generality, that the cost function value is non-negative and that, for every $\theta_i \in \mathbf{\Theta}_N$, there is at least one 'best' decision, which has zero cost:

$$\min_{a \in \mathcal{A}} C(a|\theta_i) = 0\,. \tag{2.3}$$

In the rest of this work (except in chapter 3 where we analyse this assumption), we assume that this condition holds.

In what follows, we shall use the short-hand notation $C(a|\mathbf{q})$ to denote *expected* cost:

$$C(a|\mathbf{q}) = \Big\langle C(a|\theta) \Big\rangle_{\theta|\mathbf{q}} = \sum_{i=1}^{N} q_i C(a|\theta_i) \tag{2.4}$$

where $q_i = P(\theta_i|\mathbf{q})$. (In places, we shall alternatively use the above triangular bracket notation to indicate expectation w.r.t. the probability distribution in the bracket subscript.)

Now, let $\mathcal{A}^*_{\mathbf{q}} \subseteq \mathcal{A}$ denote the subset that minimizes expected cost:

$$\mathcal{A}^*_{\mathbf{q}} = \arg\min_{a' \in \mathcal{A}} C(a'|\mathbf{q}) = \left\{ a \in \mathcal{A} \middle| C(a|\mathbf{q}) = \min_{a' \in \mathcal{A}} C(a'|\mathbf{q}) \right\}. \tag{2.5}$$

We require for every well-defined cost function that $\mathcal{A}^*_{\mathbf{q}}$ be non-empty for every $\mathbf{q} \in \mathbb{P}_N$.

A cost function for which $\mathcal{A}^*_{\mathbf{q}}$ has a single element for every $\mathbf{q}$ is denoted *strict*. If it has multiple elements for some $\mathbf{q}$, then the cost function is *non-strict*. For example, logarithmic cost is strict and zero-one cost is non-strict.

For notational convenience in what follows, we shall require that for every cost function there is an auxiliary *tie-breaking* function, b, that selects a unique element from every $\mathcal{A}^*_{\mathbf{q}}$. We require that b is defined for every $\mathcal{A}^*_{\mathbf{q}}$ and that $b(\mathcal{A}^*_{\mathbf{q}}) \in \mathcal{A}^*_{\mathbf{q}}$. Of course, for strict cost functions the tie-breaker is trivial.

The details of the tie-breaking function are unimportant in what follows, it is just a mechanism to simplify our notation for Bayes decisions. We use the tie-breaking function to define the special argmin* operator, which returns a unique minimizing argument:

$$\operatorname*{argmin^*}_{a\in\mathcal{A}} C(a|\mathbf{q}) = b(\mathcal{A}^*_{\mathbf{q}})\,. \tag{2.6}$$

Of course, we have:

$$C\bigl(\operatorname*{argmin^*}_{a\in\mathcal{A}} C(a|\mathbf{q})\big|\mathbf{q}\bigr) = \min_{a\in\mathcal{A}} C(a|\mathbf{q})\,. \tag{2.7}$$

2.1.4 Bayes decisions

The Bayes decision mechanism works as follows: It cannot always choose a zero cost outcome, because the true proposition θ is unknown. Instead it uses a given probability distribution $\mathbf{q} = (q_1, q_2, \dots, q_N)$ for θ, which enables the computation of $C(a|\mathbf{q}) = \sum_{i=1}^N q_i C(a|\theta_i)$, the *expected cost* of choosing a, for every $a \in \mathcal{A}$. A *Bayes decision* is any action that minimizes the expected cost, i.e. any $a \in \mathcal{A}^*_{\mathbf{q}}$.

For general cost functions, a Bayes decision is a *minimum-expected-cost* decision. For zero-one cost, it is also a *minimum-probability-of-error* decision, and if $\mathbf{q}$ is a posterior distribution, then it may also be called a *maximum-posterior* (MAP) decision.

To see that the Bayes decision is not necessarily unique, consider the example of the non-strict zero-one cost and $q_1 = q_2 = \cdots = q_N$, in which case every $a \in \mathcal{A}$ is a minimizer. In contrast, the logarithmic cost, $C_{\log}(\mathbf{p}|\theta)$, is strict, because $\mathbf{q} = \arg\min_{\mathbf{p}} \sum -q_i \log p_i$.

Next, we ask the natural question:

> How good are the Bayes decisions made with $\mathbf{q}$ and therefore how good is the probability distribution $\mathbf{q}$ for the purpose of making decisions?

In the rest of this section we analyse in general the cost of Bayes decisions. Then in the next section we specialize to the case where posterior probability distributions are used to make Bayes decisions.

2.1.5 Bayes cost is a proper scoring rule

In what follows, the Bayes decision itself, $a \in \mathcal{A}^*_{\mathbf{q}}$, is not important, but the *cost of the outcome* of this decision is of primary interest. In order to express this cost as a function of $\mathbf{q}$, we need to choose a unique Bayes decision, denoted $a^*_{\mathbf{q}}$, for every $\mathbf{q}$ and we do this with argmin*:

$$a^*_{\mathbf{q}} = b(\mathcal{A}^*_{\mathbf{q}}) = \operatorname*{argmin^*}_{a} \sum_{j=1}^{N} q_j C(a|\theta_j)\,. \tag{2.8}$$

The cost of the outcome, $(a^*_{\mathbf{q}}, \theta)$, of this decision is denoted with the special notation C^*:

$$C^*(\mathbf{q}|\theta) = C(a^*_{\mathbf{q}}|\theta)\,. \tag{2.9}$$

Notice that (2.9) transforms any cost function C, defined on $\mathcal{A} \times \mathbf{\Theta}_N$, to a new cost function C^*, defined on $\mathbb{P}_N \times \mathbf{\Theta}_N$. If $\mathcal{A}$ has a finite number of elements, then $a \in \mathcal{A}$ is termed a 'hard decision', while C^* assigns costs to 'soft decisions' in the form of probability distributions, $\mathbf{q} \in \mathbb{P}_N$. This special cost function is known as a *proper scoring rule.* An explanation of why it is called 'proper' is deferred to chapter 3, where proper scoring rules are discussed in more detail.

For non-strict cost functions (such as zero-one cost), the tie-breaking function b influences the value of C^*, for some values of $\mathbf{q}$. To see this, consider two-class zero-one cost, for $\mathbf{\Theta}_N = \{\theta_1, \theta_2\}$, where $C_{\text{err}}(\theta_j|\theta_i) = 1 - \delta_{ij}$. Now $C^*_{\text{err}}(\mathbf{q}|\theta_1) = 1 - u(q_1 - 0.5)$ and $C^*_{\text{err}}(\mathbf{q}|\theta_2) = u(q_1 - 0.5)$, where u denotes the *unit step* function. However, the value of $u(0)$ and of $C^*_{\text{err}}(q_1 = 0.5|\theta)$ is undefined. If we now let $b(\{\theta_1, \theta_2\}) = \theta_1$, then $u(0) = 1$ and the step functions are right-continuous. Choosing $b(\{\theta_1, \theta_2\}) = \theta_2$, gives left-continuous step functions.[2]

For strict cost functions (such as logarithmic cost), the value of C^* is determined without consulting the tie-breaker, since $\mathcal{A}^*_{\mathbf{q}}$ has a unique element for every $\mathbf{q}$.

Condition (2.3) still holds for C^*, because $C^*(q_i = 1|\theta_i) = \min_a C(a|\theta_i) = 0$.

Figure 2.1 shows two examples of proper scoring rules, for $N = 2$. Notice that $C^*(q_1|\theta_1)$ is decreasing, because when θ_1 is true, then a larger q could make lower cost decisions. Similarly, when θ_2 is true, then the scoring rule is increasing, because then a smaller q could make lower cost decisions. A probability distribution $\mathbf{q}$ is judged by a proper scoring rule according to the cost of the decisions that could be made with the distribution.

Next, we analyse further properties of proper scoring rules in terms of their *expected values.*

[2]In fact, we can construct a similar proper scoring rule defined in terms of a unit step with an arbitrary value at $u(0)$. Let the decision be compound: $(\theta_j, \theta_k) \in \mathcal{A} = \mathbf{\Theta}_N \times \mathbf{\Theta}_N$. With $0 \leq \alpha \leq 1$, define $C_\alpha\big((\theta_j, \theta_k)|\theta_i\big) = \alpha C_{\text{err}}(\theta_j|\theta_i) + (1-\alpha) C_{\text{err}}(\theta_k|\theta_i)$ and choose the tie-breaker as $b(\mathcal{A}) = (\theta_1, \theta_2)$. Then $C^*_\alpha(\mathbf{q}|\theta_1) = 1 - u(q_1 - 0.5)$ and $C^*_\alpha(\mathbf{q}|\theta_2) = u(q_1 - 0.5)$, where $u(0) = \alpha$.

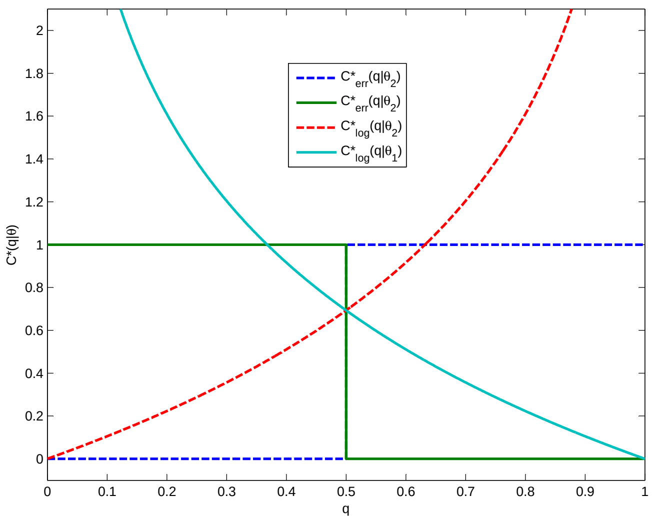

Figure 2.1: The proper scoring rules C^*_{log} and C^*_{err}, where $\mathbf{q} = (q, 1-q)$.

2.1.6 Bayes risk is generalized entropy

The *expected cost of the outcome of a Bayes decision* is known as the *Bayes risk*, denoted as:

$$C^*(\mathbf{q}|\mathbf{p}) = \left\langle C^*(\mathbf{q}|\theta) \right\rangle_{\theta|\mathbf{p}} = \sum_{i=1}^{N} p_i C^*(\mathbf{q}|\theta_i) \tag{2.10}$$

where $C^*(\mathbf{q}|\theta)$ is the above-defined cost of the Bayes decision made with $\mathbf{q}$ and where in general the expectation over θ is w.r.t. some other probability distribution, $\mathbf{p} \in \mathbb{P}_N$.

Below, we consider in more detail two variants of Bayes risk:

- The special case, where $\mathbf{p} = \mathbf{q}$, forms a generalized *entropy* function.
- The general case, where $\mathbf{p}$ may be different from $\mathbf{q}$, forms a generalized *cross-entropy*.

Generalized entropy

The expected cost of a Bayes decision, made with probability distribution $\mathbf{q}$, where the expectation is that of the maker of the decision is the *generalized entropy*, denoted[3] with the short-hand $C^*(\mathbf{q}) = C^*(\mathbf{q}|\mathbf{q})$:

$$C^*(\mathbf{q}) = C^*(\mathbf{q}|\mathbf{q}) = C(a^*_{\mathbf{q}}|\mathbf{q}) = \min_{a \in \mathcal{A}} C(a|\mathbf{q}) \,. \tag{2.11}$$

[3]The reader may feel burdened at this stage with notation overload. However compare it to the traditional Cover and Thomas notation for information theory, which has separate notations for entropy, cross-entropy, KL-divergence and mutual information. We avoid in-

Notice from the form of rightmost expression that $C^*(\mathbf{q})$ is *not* dependent on the tie-breaker, b. This is because if there are multiple Bayes decisions for a given $\mathbf{q}$, then all of them have the same expected cost under $\mathbf{q}$.

Generalized entropy has the following properties which follow directly from definition (2.11), and which help to make it an intuitively satisfying measure of the uncertainty about θ when $\mathbf{q}$ is given:

- It is non-negative, $C^*(\mathbf{q}) \geq 0$.
- It is zero at every vertex of $\mathbb{P}_N$: $C^*(q_i = 1) = 0$, for every $1 \leq i \leq N$. When $q_i = 1$, then $\mathbf{q}$ leaves no uncertainty about θ and zero-expected-cost decisions can be made.
- It is a concave function.[4] It is therefore continuous and has a maximum somewhere on the simplex. We shall use the concavity below to derive further properties.
- Generalized entropy is the link that establishes equivalence between Bayes decisions and proper scoring rules. We elaborate in chapter 3, where we discuss proper scoring rules.

Figure 2.2 shows two examples of (generalized) entropy functions: For zero-one cost, $C^*_{\text{err}}(\mathbf{q})$ is the probability of error of the minimum-probability-of-error Bayes decision: $C^*_{\text{err}}(\mathbf{q}) = \min_i(1 - q_i)$. For logarithmic cost, $C^*_{\log}(\mathbf{q}) = \sum_{i=1}^N -q_i \log q_i$ is the canonical entropy function of Shannon.

Generalized cross-entropy

Although our Bayes decision is made with $\mathbf{q}$, the goodness of this decision may be judged by an independent observer who has a *different* state of knowledge about which proposition may be true. If the observer knows the true proposition θ with certainty, then the cost of the Bayes decision made with $\mathbf{q}$ is $C^*(\mathbf{q}|\theta)$. If however, the observer also has uncertainty about θ, in the form of a different probability distribution $\mathbf{p} \in \mathbb{P}_N$, then the observer may judge $\mathbf{q}$ via the *generalized cross-entropy* of $\mathbf{q}$, given $\mathbf{p}$, denoted $C^*(\mathbf{q}|\mathbf{p})$:

$$C^*(\mathbf{q}|\mathbf{p}) = C(a^*_{\mathbf{q}}|\mathbf{p})\,. \tag{2.12}$$

The salient property of generalized cross-entropy follows by comparing right-hand-sides of (2.11) and (2.12):

$$C^*(\mathbf{q}|\mathbf{p}) \geq C^*(\mathbf{p}|\mathbf{p}) = C^*(\mathbf{p}) \tag{2.13}$$

from which we note:

troducing notations for generalized divergence and mutual information, preferring to express everything in terms of entropy or cross-entropy. In the next subsection, we will summarize the notation introduced in this section with graphical aids.

[4]For any $\mathbf{p}, \mathbf{q} \in \mathbb{P}_N$ and any $0 \leq t \leq 1$, $C^*(t\mathbf{p} + (1-t)\mathbf{q}) \geq tC^*(\mathbf{p}) + (1-t)C^*(\mathbf{q})$.

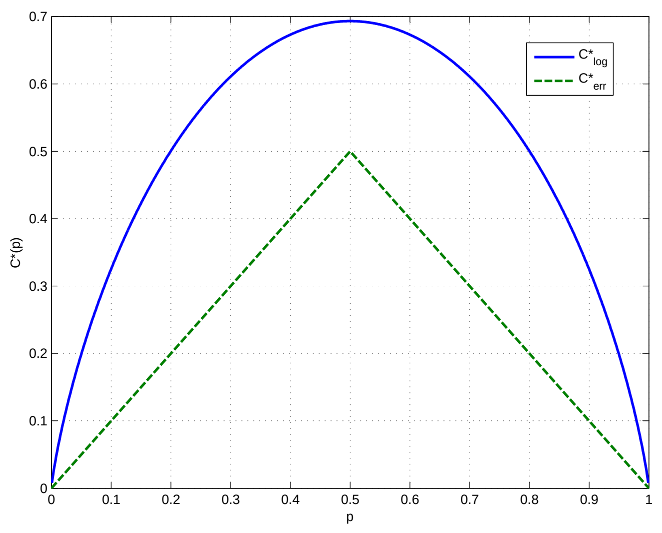

Figure 2.2: Generalized entropy functions $C^*_{\log}$ and C^*_{err}, where $\mathbf{p} = (p, 1-p)$.

- If your state of knowledge about θ is the distribution $\mathbf{p}$, then the best Bayes decisions you could make are those made with $\mathbf{p}$—and not with some other distribution.

- While generalized entropy $C^*(\mathbf{q})$ is concave and therefore bounded from above, the generalized cross-entropy $C^*(\mathbf{q}|\mathbf{p})$ can be unbounded above for certain cost functions. For example, $C^*_{\log}(q_i = 0|p_i = 1) = -\log(q_i) = \infty$. A decision made in good faith, given some $\mathbf{q}$, can nevertheless have consequences that are arbitrarily bad.

Interpretation as a measure of information

In Shannon's information theory, a decrease in entropy is an increase in the amount of information. Here our categorical probability distributions for the unknown proposition carry information about the proposition. The generalized entropy quantifies that amount of information. If we change a probability distribution, for example by going from prior to posterior via Bayes' rule, there is an associated change in the information about the proposition, which is quantified by the change in the entropies between the two distributions.

Although we use the term information everywhere, we do not introduce a notation for quantifying information. Because of the relationship Δinformation $= -\Delta$entropy, such a notation would be redundant and we prefer to express our equations in terms of entropy.

2.1.7 Bayes decision summary

This section is summarized in figure 2.3. The cost of a Bayes decision is a proper scoring rule. The expected value of a proper scoring rule, or Bayes risk, forms a generalized entropy measure. Generalized entropy measures the uncertainty of a given probability distribution as the risk of making decisions with this probability distribution.

Generalized entropy is defined in terms of some cost function. When this cost is logarithmic cost, then the resultant entropy is the canonical Shannon's entropy.

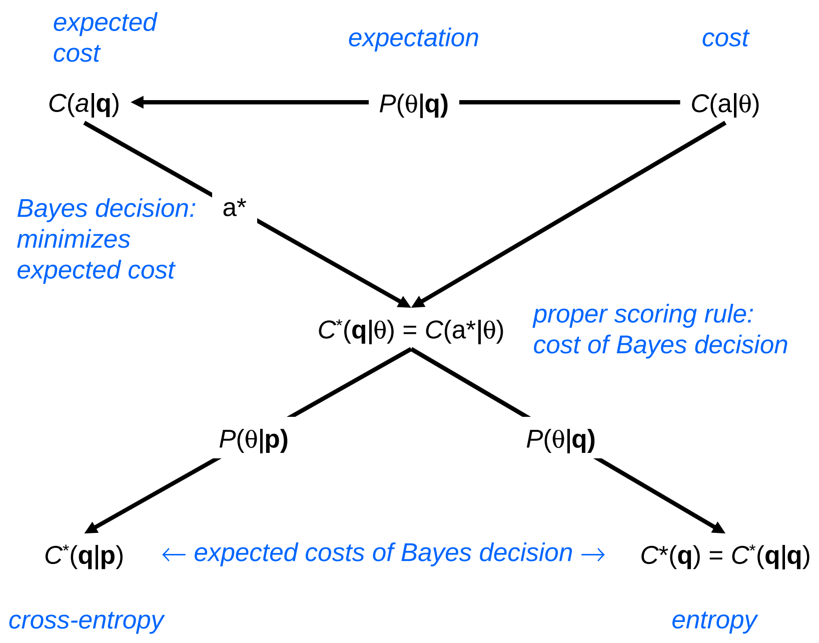

Figure 2.3: Summary of section 2.1: From cost function, via Bayes decision, to proper scoring rule, to generalized entropy. $C(a|\theta)$ is the cost of having made decision a, when it turns out that θ is the true class. Expectation and minimization of cost define the quantities of interest. The main result is $C^*(\mathbf{q}|\mathbf{p}) \geq C^*(\mathbf{p})$.

2.2 Posterior cross-entropy

In this section, we return to our agenda of analysing the usefulness of the information extracted from speech by a recognizer.

Our tool is still the expected cost of making Bayes decisions, but now these decisions are made with posterior distributions. Since the posterior-distributions are data-dependent, we now have to extend our analysis to include expectations over the data. The resultant expectation is a generalized *posterior cross-entropy.*

Below we discuss the recognizer in more detail; introduce a new role player called the *observer*; derive posterior cross-entropy from posterior Bayes risk and finally analyse its basic properties.

Again, the reader may want to look ahead at figure 2.4, which summarizes this section. The notation in that figure will be defined in what follows.

2.2.1 The recognizer

Let $x \in \mathcal{X}$ denote the *input* to the recognizer. Here x represents one or more utterances of speech—or processed versions thereof, such as sequences of feature vectors. The goal of the pattern recognizer is to recognize which one of the N propositions $\{\theta_1, \theta_2, \dots, \theta_N\}$ is true for the given input data $x \in \mathcal{X}$.

To be generally applicable in the face of uncertainty, our analysis will show that the output of the recognizer should be in the form of a *posterior* probability distribution over the propositions.

In addition to x, a further input, $\boldsymbol{\pi}$, namely a *prior* probability distribution over the propositions is given. We regard the prior as a variable input and not as a fixed part of the pattern recognizer. The pattern recognizer is therefore a function $\mathcal{R} : \mathcal{X} \times \mathbb{P}_N \mapsto \mathbb{P}_N$, where $\mathbb{P}_N$ is the simplex where categorical probability distributions over the N propositions live. We denote the function output as $\mathbf{r} = (r_1, r_2, \dots, r_N) = \mathcal{R}(x, \boldsymbol{\pi})$, so that:

$$r_i = P(\theta_i|\mathbf{r}) = P(\theta_i|x, \boldsymbol{\pi}, \mathcal{R})\,. \tag{2.14}$$

It is worth emphasizing that the recognizer $\mathcal{R}$ gives *only* the posterior for θ. It does not specify any probability distributions for x, so that in the general case, distributions of the form $P(x|\boldsymbol{\pi}, \mathcal{R})$ or $P(x|\theta, \mathcal{R})$ are not defined. In the special case where the recognizer uses a fully specified generative model, such distributions for x would be defined, but we want to be able to analyse a more general class of recognizers.

2.2.2 The observer

Thus far in our analysis we have worked only with expectations over θ. In what follows, we shall also need to express expectations over x, for which we shall need probability distributions for x.

For this purpose, we introduce the concept of an independent *observer*, denoted $\mathcal{O}$. The only purpose of this new role player is to have a probability distribution for x of the form $P(x|\theta_i, \mathcal{O})$, for every $i = 1, \dots, N$.

In what follows, we shall always analyse recognizer performance from the point of view of an observer. In this chapter, we treat the case of a general, unspecified observer. In chapter 3, we let the recognizer observe *itself* as a mechanism for analysing the key properties of proper scoring rules. In contrast, in chapter 5, we construct a special independent observer called the *evaluator*, which observes and thereby evaluates the recognizer.

The observer's posterior

Notice that the observer's distribution $P(x|\theta,\mathcal{O})$ also determines the *observer's posterior*, $P(\theta|x,\boldsymbol{\pi},\mathcal{O})$. In analogy to the recognizer's posterior, $\mathcal{R}(x,\boldsymbol{\pi})$, we define[5] the function $\mathcal{O}_X(x,\boldsymbol{\pi})$ which maps x to the observer's posterior distribution:

$$\begin{aligned}\mathbf{p} &= (p_1,p_2,\dots,p_N) = \mathcal{O}_X(x,\boldsymbol{\pi}),\\ p_i &= P(\theta_i|\mathbf{p}) = P(\theta_i|x,\boldsymbol{\pi},\mathcal{O}) = \frac{\pi_i P(x|\theta_i,\mathcal{O})}{\sum_{j=1}^N \pi_j P(x|\theta_j,\mathcal{O})}\,.\end{aligned} \tag{2.15}$$

2.2.3 Posterior Bayes risk

We now have the tools in place to express the expected cost of using the recognizer's posterior to make Bayes decisions. As shown in section 2.1.5, when proposition θ is true for x, the cost of the Bayes decision that was made (before knowing the true proposition) by using the recognizer's posterior, $\mathbf{r}=\mathcal{R}(x,\boldsymbol{\pi})$, is given by the *proper scoring rule* $C^*(\mathbf{r}|\theta)$. The expected value of this cost as seen by an observer $\mathcal{O}$, is the *posterior Bayes risk*, denoted $\bar{C}(\mathcal{R}|\mathcal{O})$. It can be expressed as:

$$\bar{C}(\mathcal{R}|\mathcal{O}) = \Big\langle C^*(\mathcal{R}(x,\boldsymbol{\pi})|\theta)\Big\rangle_{x,\theta|\boldsymbol{\pi},\mathcal{O}} \tag{2.16}$$

$$= \sum_{i=1}^N \pi_i \Big\langle C^*(\mathcal{R}(x,\boldsymbol{\pi})|\theta_i)\Big\rangle_{x|\theta_i,\mathcal{O}} \tag{2.17}$$

where the expectation is w.r.t. the joint distribution for θ and x, factored as: $P(\theta_i,x|\boldsymbol{\pi},\mathcal{O}) = \pi_i P(x|\theta_i,\mathcal{O})$.

Although not explicit in the notation $\bar{C}(\mathcal{R}|\mathcal{O})$, it should be understood that it is dependent on the prior $\boldsymbol{\pi}$, as well as on the details of the cost function C. In later chapters, when we consider varying the prior and cost function, we shall use a more explicit notation.

The posterior Bayes risk $\bar{C}(\mathcal{R}|\mathcal{O})$ will play a central part in what follows, because it forms the basis of our evaluation criterion for the goodness of pattern recognizers. Below, we examine some of its properties.

[5]The subscript in $\mathcal{O}_X$ serves two purposes. First, it distinguishes $\mathcal{O}_X$, and $\mathcal{O}$. The former is the function that outputs the posterior. The latter is the conditioning for $P(x|\theta,\mathcal{O})$. Secondly, it distinguishes $\mathcal{O}_X$ from other similar observer's posteriors that we shall define later.

2.2.4 Posterior Bayes risk is posterior cross-entropy

Now we express $\bar{C}(\mathcal{R}|\mathcal{O})$ in a different way by factoring the joint distribution the other way round: $P(\theta_i, x|\boldsymbol{\pi}, \mathcal{O}) = P(\theta_i|x, \boldsymbol{\pi}, \mathcal{O})P(x|\boldsymbol{\pi}, \mathcal{O})$, where $P(\theta_i|x, \boldsymbol{\pi}, \mathcal{O})$ is the *observer's posterior* and where the marginal for x is $P(x|\boldsymbol{\pi}, \mathcal{O}) = \sum_{i=1}^{N} \pi_i P(x|\theta_i, \mathcal{O})$.

By using $\mathcal{O}_X(x, \boldsymbol{\pi})$, defined in (2.15), for the function that computes the observer's posterior, we can now express the posterior Bayes risk in terms of expected cross-entropy between observer's and recognizer's posteriors:

$$\bar{C}(\mathcal{R}|\mathcal{O}) = \Big\langle C^*\big(\mathcal{R}(x, \boldsymbol{\pi})|\mathcal{O}_X(x, \boldsymbol{\pi})\big)\Big\rangle_{x|\boldsymbol{\pi},\mathcal{O}} . \tag{2.18}$$

In analogy to the notation in Cover and Thomas [28], we denote this expectation as the generalized *posterior cross-entropy.*

In summary, posterior Bayes risk, $\bar{C}(\mathcal{R}|\mathcal{O})$, is also posterior cross-entropy. This identity turns out to be very useful below in determining some of the properties of Bayes risk.

2.2.5 Minimum posterior cross-entropy is posterior entropy

By applying (2.13) for every x in (2.18), we find that if we minimize $\bar{C}(\mathcal{R}|\mathcal{O})$ by considering all possible functions of the form $\mathcal{R} : \mathcal{X} \times \mathbb{P}_N \mapsto \mathbb{P}_N$, then a global minimum (not necessarily unique) is given by the observer's posterior function $\mathcal{O}_X$. The minimum posterior cross-entropy is therefore:

$$\bar{C}(\mathcal{O}_X|\mathcal{O}) = \Big\langle C^*\big(\mathcal{O}_X(x, \boldsymbol{\pi})\big)\Big\rangle_{x|\boldsymbol{\pi},\mathcal{O}} . \tag{2.19}$$

Again, in analogy to Cover and Thomas, we denote the expected value of the entropy of the posterior simply as the generalized *posterior entropy*, for which we adopt the short-hand $\bar{C}(\mathcal{O}_X)$. The far-reaching result of this subsection is now summarized as:

$$\bar{C}(\mathcal{O}_X) = \bar{C}(\mathcal{O}_X|\mathcal{O}) \leq \bar{C}(\mathcal{R}|\mathcal{O}) . \tag{2.20}$$

2.2.6 Summary of generalized entropy notation

At this point, a summary of our notation defined so far may help the reader:

$C^*(\mathbf{q}|\mathbf{p})$ is cross-entropy and $C^*(\mathbf{p}) = C^*(\mathbf{p}|\mathbf{p})$ is entropy, when categorical distributions $\mathbf{p}$ and $\mathbf{q}$ are given. In particular, we shall refer to $C^*(\boldsymbol{\pi})$ as the prior entropy.

$\bar{C}(\mathcal{R}|\mathcal{O})$ is posterior cross-entropy and $\bar{C}(\mathcal{O}_X) = \bar{C}(\mathcal{O}_X|\mathcal{O})$ is posterior entropy. These are the *expected* entropies of the outputs of the functions $\mathcal{R}$ and $\mathcal{O}_X$, where the expectation is conditioned on $\mathcal{O}$. Figure 2.4 gives a graphical summary of posterior entropy.

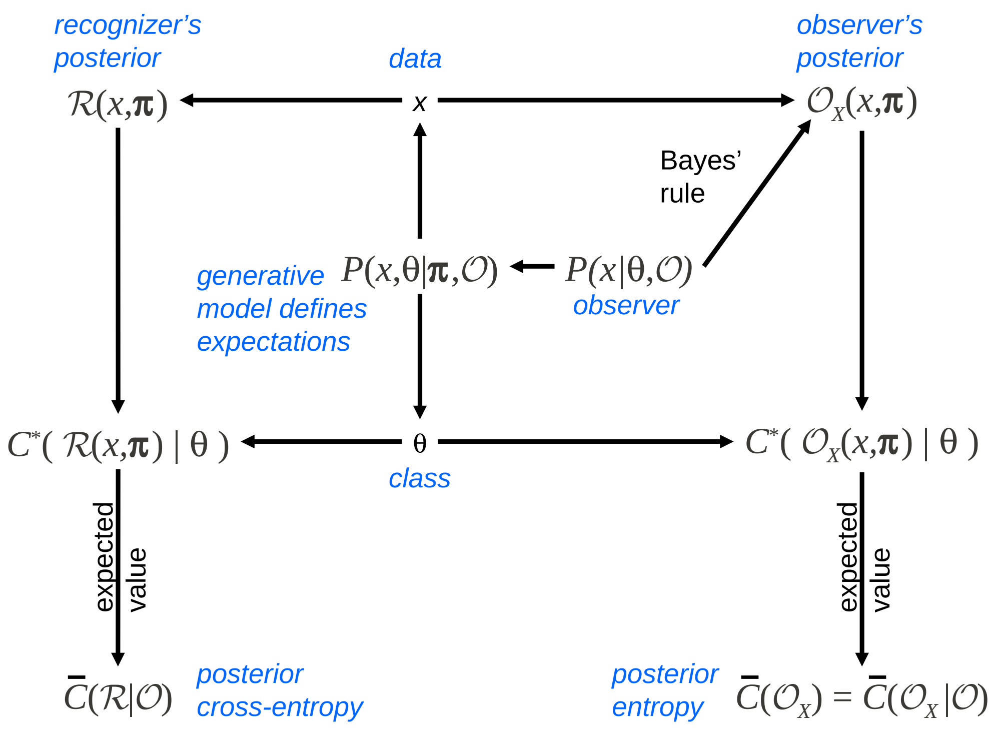
recognizer's posterior
data
observer's posterior
$\mathcal{R}(x,\boldsymbol{\pi})$
x
$\mathcal{O}_X(x,\boldsymbol{\pi})$
Bayes' rule
generative model defines expectations
$P(x,\theta|\boldsymbol{\pi},\mathcal{O})$
$P(x|\theta,\mathcal{O})$
observer
$C^*(\mathcal{R}(x,\boldsymbol{\pi})\,|\,\theta)$
θ
class
$C^*(\mathcal{O}_X(x,\boldsymbol{\pi})\,|\,\theta)$
expected value
expected value
$\bar{C}(\mathcal{R}|\mathcal{O})$
posterior cross-entropy
posterior entropy
$\bar{C}(\mathcal{O}_X) = \bar{C}(\mathcal{O}_X|\mathcal{O})$

Figure 2.4: Summary of section 2.2: The class prior, $\boldsymbol{\pi}$, is given. An *observer*, $\mathcal{O}$, additionally defines class-conditional distributions for the data. This forms a complete generative model which defines not only (i) the expected value of any function of the data, but also (ii) the observer's posterior. The expectations of interest are of the costs, C^*, of making Bayes decisions with the recognizer's or observer's posterior. The main result is $\bar{C}(\mathcal{R}|\mathcal{O}) \geq \bar{C}(\mathcal{O}_X|\mathcal{O}) \leq C^*(\boldsymbol{\pi})$, where $C^*(\boldsymbol{\pi})$ is the cost of making Bayes decisions with the prior alone.

2.2.7 Maximum posterior entropy is prior entropy

If we use the observer's posterior to make decisions, these decisions are on average at least as good as those made with the prior:

$$\bar{C}(\mathcal{O}_X) \leq C^*(\boldsymbol{\pi}) \tag{2.21}$$

or posterior entropy cannot exceed the prior entropy.

Proof. This proof is based on [57]. Note that the expected value of the posterior, $\mathcal{O}_X(x,\boldsymbol{\pi})$, is just the prior:

$$\Big\langle \mathcal{O}_X(x,\boldsymbol{\pi}) \Big\rangle_{x|\mathcal{O},\boldsymbol{\pi}} = \boldsymbol{\pi} \tag{2.22}$$

and by using Jensen's inequality and the concavity of $C^*(\mathbf{q})$:

$$\bar{C}(\mathcal{O}_X) = \Big\langle C^*\big(\mathcal{O}_X(x,\boldsymbol{\pi})\big)\Big\rangle_{x|\boldsymbol{\pi},\mathcal{O}} \leq C^*\left(\Big\langle \mathcal{O}_X(x,\boldsymbol{\pi})\Big\rangle_{x|\boldsymbol{\pi},\mathcal{O}}\right) = C^*(\boldsymbol{\pi}) \tag{2.23}$$

which proves (2.21). ■

2.2.8 Interpretation via mutual information

The entropy difference $C^*(\boldsymbol{\pi}) - \bar{C}(\mathcal{O}_X)$ forms a useful definition of the *generalized mutual information* between x and θ. Mutual information quantifies the relevant information about θ that the observer sees in x. In the case of logarithmic cost, this definition agrees with the classical definition, as given for example by Cover and Thomas [28].

Equation (2.21) and the non-negativity of our cost functions show that the mutual information is bounded between 0 and the prior entropy, $C^*(\boldsymbol{\pi})$.

For reasons noted above, we do not provide a notation for mutual information. Moreover, we will find that we will not be able to do meaningful practical measurement of mutual information. This is explained in chapter 5. In contrast, we will be making practical measurement of quantities that can be interpreted as generalized cross-entropy.

2.2.9 Posterior cross-entropy summary

In this section, we defined the *posterior Bayes risk*, $\bar{C}(\mathcal{R}|\mathcal{O})$, which we will use in the rest of this work as the basis for our evaluation criterion for pattern recognizers.

Posterior Bayes risk may be expressed as (2.17), which as we will show in chapter 5, is useful for deriving a practical evaluation recipe. However, it also has the alternative formulation of (2.18), as generalized *posterior cross-entropy*. This formulation proved useful in this section for deriving some important properties of this evaluation criterion, which we summarize below.

$\bar{C}(\mathcal{R}|\mathcal{O})$ is the observer's expectation of the cost of making Bayes decisions with a recognizer $\mathcal{R}$. $C^*(\boldsymbol{\pi})$ is the expected cost of making Bayes decisions using the prior alone. A recognizer may therefore be deemed *useful* if $\bar{C}(\mathcal{R}|\mathcal{O}) \leq C^*(\boldsymbol{\pi})$. By combining (2.20) and (2.21), we find:

$$\min_{\mathcal{R}} \bar{C}(\mathcal{R}|\mathcal{O}) = \bar{C}(\mathcal{O}_X|\mathcal{O}) \leq C^*(\boldsymbol{\pi}) \tag{2.24}$$

which shows that a useful recognizer exists for every cost function, C, and every prior, $\boldsymbol{\pi}$.

In the next section, we make further use of the posterior cross-entropy interpretation to demonstrate that there exist also other useful recognizers, which are not as theoretically optimal as $\mathcal{O}_X$, but which can be much easier to realize in practice.

2.3 Data processing

In the previous section we showed that the best recognizer, from the point of view of the observer, is the one that computes the observer's posterior $\mathcal{O}_X$. We also showed that $\mathcal{O}_X$ is *useful* in the sense that the expected cost of its decisions is no greater than the expected cost of using the prior to make decisions.

Here we show that *other* useful recognizers also exist. The search for other useful recognizers is important, because the ideal recognizer $\mathcal{O}_X$ is in general not available, for two reasons:

- Usually $P(x|\theta,\mathcal{O})$ (which defines $\mathcal{O}_X$) is not available.
- Even if $P(x|\theta,\mathcal{O})$ were available, the computational complexity to implement $\mathcal{O}_X$ may be prohibitive.

We therefore examine here the consequences of *processing* the input data, x, not via $\mathcal{O}_X(x,\boldsymbol{\pi})$, but via some other function of x.

The results of this section will further provide us with the tools to analyse a secondary evaluation criterion, namely *calibration*, that will be introduced in the next and final section of this chapter.

2.3.1 Statistics of the data

A function $S(x)$ that processes x in order to infer the value of θ is called a *statistic* of x, for θ. The key to analyse the effect of data processing with $S(x)$ is the observer's posterior, given the output of $S(x)$.

Posteriors given statistics

We have already defined the observer's posterior $\mathcal{O}_X(x,\boldsymbol{\pi})$, which can be viewed as a function of the *identity statistic*, $x = X(x)$.

We now generalize this to other statistics. Let $s = S(x)$, where S is some statistic of x. Then the observer's conditional distributions for s may be written in terms of the Dirac delta, δ, as:

$$P(x,s|\theta_i,S,\mathcal{O}) = P(s|S,x)P(x|\theta_i,\mathcal{O}) = \delta\big(s-S(x)\big)P(x|\theta_i,\mathcal{O}), \tag{2.25}$$

$$P(s|\theta_i,S,\mathcal{O}) = \int_{\mathcal{X}} P(x,s|\theta_i,S,\mathcal{O})\,dx \tag{2.26}$$

which in turn defines the observer's posterior, given s as:

$$P(\theta_i|s,\boldsymbol{\pi},S,\mathcal{O}) = \frac{\pi_i P(s|\theta_i,S,\mathcal{O})}{\sum_{j=1}^{N}\pi_j P(s|\theta_j,S,\mathcal{O})}\,. \tag{2.27}$$

In analogy to $\mathcal{O}_X$, we now define the function $\mathcal{O}_S(s,\boldsymbol{\pi})$ as one that outputs the posterior distribution with components as defined in (2.27).

Note on expectation

Let $f(s)$ be some function of s, then (see e.g. [56]) for $s = S(x)$:

$$\Big\langle f(s) \Big\rangle_s = \Big\langle f\big(S(x)\big) \Big\rangle_x . \tag{2.28}$$

(To see this, define the LHS in terms of (2.26) and integrate out s, which is trivial because of the Dirac delta.) We shall make use of this fact several times in this work—the first two applications follow in the next two equations below.

Posterior entropy

In analogy to the posterior entropy $\bar{C}(\mathcal{O}_X)$, we now define the posterior entropy of $\mathcal{O}_S$, for which we use short-hand $\bar{C}(\mathcal{O}_S)$:

$$\begin{aligned} \bar{C}(\mathcal{O}_S) = \bar{C}(\mathcal{O}_S|\mathcal{O}) &= \Big\langle C^*\big(\mathcal{O}_S(s,\boldsymbol{\pi})\big) \Big\rangle_{s|\boldsymbol{\pi},\mathcal{O}} \\ &= \Big\langle C^*\big(\mathcal{O}_S(S(x),\boldsymbol{\pi})\big) \Big\rangle_{x|\boldsymbol{\pi},\mathcal{O}} . \end{aligned} \tag{2.29}$$

As we show below, $\bar{C}(\mathcal{O}_S)$ has properties analogous to $\bar{C}(\mathcal{O}_X)$.

Minimum posterior cross-entropy is posterior entropy

Consider a recognizer, $\mathcal{R}$, which processes the data only via some statistic $s = S(x)$, so that $\mathcal{R}(x,\boldsymbol{\pi}) = \mathcal{R}_S\big(S(x),\boldsymbol{\pi}\big)$. The posterior Bayes risk, or posterior cross-entropy, for this recognizer is:

$$\begin{aligned} \bar{C}(\mathcal{R}_S|\mathcal{O}) &= \Big\langle C^*\Big(\mathcal{R}_S\big(S(x),\boldsymbol{\pi}\big)\Big|\theta\Big) \Big\rangle_{x,\theta|\boldsymbol{\pi},\mathcal{O}} \\ &= \Big\langle C^*\big(\mathcal{R}_S(s,\boldsymbol{\pi})\big|\theta\big) \Big\rangle_{s,\theta|\boldsymbol{\pi},\mathcal{O}} \\ &= \Big\langle C^*\big(\mathcal{R}_S(s,\boldsymbol{\pi})\big|\mathcal{O}_S(s,\boldsymbol{\pi})\big) \Big\rangle_{s|\boldsymbol{\pi},\mathcal{O}} . \end{aligned} \tag{2.30}$$

As before, applying (2.13) for every s, we find:

$$\bar{C}(\mathcal{R}_S|\mathcal{O}) \geq \bar{C}(\mathcal{O}_S|\mathcal{O}) = \bar{C}(\mathcal{O}_S) . \tag{2.31}$$

Since $\mathcal{R}_S$ is an arbitrary function of s, this shows that the best recognizer of the form $\mathcal{R}_S\big(S(x),\boldsymbol{\pi}\big)$, is the observer's own posterior, $\mathcal{O}_S\big(S(x),\boldsymbol{\pi}\big)$.

Maximum posterior entropy is prior entropy

Here we show that $\mathcal{O}_S$ is also a useful recognizer:

$$\bar{C}(\mathcal{O}_S) \leq C^*(\boldsymbol{\pi}) . \tag{2.32}$$

Proof. By making use of (2.28), we can re-use the proof of (2.21). ■

The benefit of a statistic

In section 2.2 we demonstrated the existence of at least one useful recognizer, $\mathcal{O}_X(x,\boldsymbol{\pi})$. Here we have just shown that 'pre-processing' the input x with some statistic, $s = S(x)$, can give *another* useful recognizer, $\mathcal{O}_S(s,\boldsymbol{\pi})$. This can be exploited by designing the statistic, S, to give its output in a form that is more easily modelled than the original x. Then it may be easier for the recognizer, $\mathcal{R}_S\big(S(x),\boldsymbol{\pi}\big)$, to approximate $\mathcal{O}_S\big(S(x),\boldsymbol{\pi}\big)$. In speech processing, the feature extractor may be understood in this way.

However, there may be a downside to using $S(x)$, rather than the original x. By computing $S(x)$, some relevant information that was present in x may be irretrievably lost. This is made explicit in the form of *the data processing inequality.*

2.3.2 The generalized data processing inequality

The data processing inequality is traditionally stated in terms of *mutual information*, see e.g. Cover and Thomas [28]. As mentioned above, mutual information is just prior entropy minus posterior entropy and we prefer to give an equivalent statement in terms of posterior entropy.

The *generalized data processing inequality* states, when $s = S(x)$ and $\mathcal{O}_X(x,\boldsymbol{\pi})$ and $\mathcal{O}_S(s,\boldsymbol{\pi})$ are the observer's posteriors given respectively x and s, then the posterior entropies are related as:

$$\bar{C}(\mathcal{O}_X) \leq \bar{C}(\mathcal{O}_S)\,. \tag{2.33}$$

This means that the best decisions the observer could make using s as input cannot have better expected cost than the best decisions the observer could make by using x as input. Equivalently, the relevant information in s cannot exceed that in x.

Proof. This proof of (2.33) is based on [57]. Since s is a function of x, θ is conditionally independent of s, given x, or $P(\theta_i|x,s,\boldsymbol{\pi},\mathcal{O}) = P(\theta_i|x,\boldsymbol{\pi},\mathcal{O})$. We can use this to show the following relation between the two posteriors:

$$\begin{aligned}
\Big\langle P(\theta|x,\boldsymbol{\pi},\mathcal{O})\Big\rangle_{x|s,\boldsymbol{\pi},\mathcal{O}} &= \int_{\mathcal{X}} P(\theta|x,\boldsymbol{\pi},\mathcal{O})P(x|s,\boldsymbol{\pi},\mathcal{O})\,dx \\
&= \int_{\mathcal{X}} P(\theta|x,s,\boldsymbol{\pi},\mathcal{O})P(x|s,\boldsymbol{\pi},\mathcal{O})\,dx \\
&= \int_{\mathcal{X}} P(\theta,x|s,\boldsymbol{\pi},\mathcal{O})\,dx \\
&= P(\theta|s,\boldsymbol{\pi},\mathcal{O})
\end{aligned} \tag{2.34}$$

or in short-hand:

$$\Big\langle \mathcal{O}_X(x,\boldsymbol{\pi})\Big\rangle_{x|s,\boldsymbol{\pi},\mathcal{O}} = \mathcal{O}_S(s,\boldsymbol{\pi})\,. \tag{2.35}$$

Again, we use Jensen's inequality and the concavity of $C^*(\mathbf{q})$:

$$\Big\langle C^*(\mathcal{O}_X(x,\boldsymbol{\pi}))\Big\rangle_{x|s,\boldsymbol{\pi},\mathcal{O}} \le C^*\left(\Big\langle \mathcal{O}_X(x,\boldsymbol{\pi})\Big\rangle_{x|s,\boldsymbol{\pi},\mathcal{O}}\right) = C^*(\mathcal{O}_S(s,\boldsymbol{\pi})) \tag{2.36}$$

and then take the expectation w.r.t. s on both sides of the inequality:

$$\Big\langle C^*(\mathcal{O}_X(x,\boldsymbol{\pi}))\Big\rangle_{x|\boldsymbol{\pi},\mathcal{O}} \le \Big\langle C^*(\mathcal{O}_S(s,\boldsymbol{\pi}))\Big\rangle_{s|\boldsymbol{\pi},\mathcal{O}} \tag{2.37}$$

which by definitions (2.20) and (2.29) proves (2.33). ■

Finally, we ask the devil's advocate question: If processing the data cannot create information, and has the potential to destroy information, why do we process the data? The answer is that we need to compute the posterior $\mathcal{O}_X(x,\boldsymbol{\pi})$, or at least some approximation $\mathcal{R}(x,\boldsymbol{\pi})$, in order to be able to utilize the information in the data. In fact, as we show below, $\mathcal{O}_X(x,\boldsymbol{\pi})$ is the *most processed* form of the relevant information in x.

2.3.3 The posterior is minimal sufficient

Any function $S(x)$ can be denoted a *statistic* of x. Here we give the conditions for a statistic to be a *sufficient statistic* and a *minimal sufficient statistic* [58, 59].

As before, let $s = S(x)$, and let respectively $\mathcal{O}_X(x,\boldsymbol{\pi})$ and $\mathcal{O}_S(s,\boldsymbol{\pi})$ denote the observer's posteriors for θ, given x and given s. The statistic S is a *sufficient statistic*[56] of x for θ, if for every x and $\boldsymbol{\pi}$:

$$\mathcal{O}_X(x,\boldsymbol{\pi}) = \mathcal{O}_S(S(x),\boldsymbol{\pi})\,. \tag{2.38}$$

A sufficient statistic can therefore be used to make decisions which are as good as those made via the original data. Trivially, $\mathcal{O}_X(x,\boldsymbol{\pi})$ is a sufficient statistic of x, for θ, because it *is* the posterior that defines sufficiency.

A sufficient statistic is *minimal sufficient* if it is a function of every sufficient statistic. That is, a sufficient statistic $M(x)$ is minimal sufficient, if for every sufficient statistic $s = S(x)$ there exists some function $g(s)$, so that $M(x) = g(S(x))$.

We now show that $\mathcal{O}_X(x,\boldsymbol{\pi})$ is a minimal sufficient statistic of x for θ, by choosing the function g, such that $g(s) = \mathcal{O}_S(s,\boldsymbol{\pi})$. Definition (2.38) shows that if $S(x)$ is sufficient, then $\mathcal{O}_X(x,\boldsymbol{\pi}) = g(S(x))$. Since $S(x)$ is arbitrary, the posterior $\mathcal{O}_X(x,\boldsymbol{\pi})$ is a function of *every* sufficient statistic $S(x)$ and is therefore minimal sufficient.

Our point is that since $\mathcal{O}_X(x,\boldsymbol{\pi})$ is a function of every sufficient statistic, it is at least as much processed as any other sufficient statistic and is therefore the *most processed* form of the relevant information in x.

2.3.4 Invertible functions preserve information

We conclude this section by pointing out that statistics can be partitioned into equivalence classes, where all statistics in an equivalence class extract the same amount of relevant information. Two statistics are equivalent if one can be derived from the other via an invertible transformation.

Let $s = S(x)$ and $z = Z(x)$ be two statistics of x, with associated posteriors $\mathcal{O}_S(s, \boldsymbol{\pi})$ and $\mathcal{O}_Z(z, \boldsymbol{\pi})$. Further, let there be a bijection[6] f from the range of S to the range of Z, such that $f(S(x)) = Z(x)$ and $f^{-1}(Z(x)) = S(x)$, for every $x \in \mathcal{X}$. Then the posteriors are equal and so is the relevant information about θ extracted by each statistic:

$$\mathcal{O}_S\big(S(x), \boldsymbol{\pi}\big) = \mathcal{O}_Z\big(Z(x), \boldsymbol{\pi}\big) \tag{2.39}$$

and

$$\bar{C}(\mathcal{O}_S) = \bar{C}(\mathcal{O}_Z)\,. \tag{2.40}$$

2.3.5 Data processing summary

In summary of this section, suppose that $x = X(x)$, $s = S(x) = f^{-1}(Z(x))$ and $z = Z(x) = f(S(x))$, then the best possible recognizers (according to the observer), that take respectively x, s and z as their inputs are $\mathcal{O}_X$, $\mathcal{O}_S$ and $\mathcal{O}_Z$. The expected costs of using these recognizers are related as:

$$\bar{C}(\mathcal{O}_X) \leq \bar{C}(\mathcal{O}_S) = \bar{C}(\mathcal{O}_Z) \leq C^*(\boldsymbol{\pi}) \tag{2.41}$$

where $C^*(\boldsymbol{\pi})$ is the expected cost of decisions made with the prior alone. Thus all of the observer's posteriors, $\mathcal{O}_X$, $\mathcal{O}_S$ and $\mathcal{O}_Z$ are useful recognizers in the sense that they make decisions with lower expected cost than decisions that can be made with the prior.

Processing with non-invertible functions can reduce the relevant information content (equivalently the decision making ability). Processing with invertible functions preserves the information content.

Even though processing can destroy relevant information, processing is unavoidable. In order to optimally use the information in data, the data *has* to be processed to compute the posterior. In fact we showed that the posterior is the *most* processed form of the relevant information.

2.4 Calibration

The bulk of this chapter was dedicated to deriving and analysing posterior Bayes risk as basis for our primary evaluation criterion. In this section, we introduce *calibration* as a secondary evaluation criterion.

[6]See appendix A, which explains the difference between invertible and bijective.

In section 2.3, we analysed how processing the data through a statistic affects the information content, or decision-making ability, of the data. In this section, we regard the given recognizer, $\mathcal{R}$, as a statistic and then ask what may be gained by *further* processing of the recognizer's output.

The recognizer's output is already formatted to be application-ready, in the form of a posterior probability distribution. We define a *calibration transformation* as the further processing of the recognizer's posterior, to produce another, hopefully better, version of the posterior. The calibration transformation essentially treats the original posterior merely as a statistic (or a feature) and then computes the new posterior, given this statistic.

The calibration transformation mechanism allows us to define calibration as a *criterion of the goodness* of the original recognizer: If the original recognizer cannot be improved by such transformation, then we say the calibration of the original recognizer is good.

The term calibration can be used in a second, related sense, namely as the *act of trying to improve* the calibration criterion.

In order to use calibration as a practical criterion of recognizer goodness, we need a more precise definition. A candidate for this definition, which we analyse below, is what is known in the literature as the *calibration-refinement decomposition.* Unfortunately, this definition, via its immediate intuitive appeal, led this work into an extended wild-goose chase. Ironically, it was the inspiration for the title of this dissertation, but it was also the most significant impediment to its completion. The problem lay in trying to interpret the practical mechanism for computing a calibration criterion which is proposed later in this dissertation, as an instance of this decomposition.

The key to finally resolving this difficulty was the idea of viewing our probability distributions as conditioned on the *observer.* The calibration-refinement decomposition makes theoretical sense for the case of some general hypothetical observer, but it reduces to a trivial and useless decomposition in the case of the special observer that we use for practical evaluation.

Below, we explain the calibration-refinement decomposition. In chapter 5 we explain how it breaks down and we propose how to modify the definition of calibration, so that it leads to a practical criterion. It is important for the reader to understand the calibration-refinement decomposition, since our final solution is very closely related and will be explained in terms of concepts introduced here.

2.4.1 The calibration-refinement decomposition

The calibration-refinement decomposition was shown in [5] to be applicable when a probabilistic forecaster is evaluated via any proper scoring rule. This decomposition can be conveniently explained in terms of the tools we developed in this chapter.

In particular, the decomposition can be based on inequality (2.31), which shows that posterior entropy is minimum posterior cross-entropy, also for the case where the posterior is conditioned on a statistic. As explained above, the statistic of interest here is the recognizer's posterior, $\mathbf{r} = \mathcal{R}(x, \boldsymbol{\pi})$.

The *observer's* posterior for θ given $\mathbf{r}$ is defined by (2.27) and we denote it as $\mathcal{O}_{\mathcal{R}}(\mathbf{r}, \boldsymbol{\pi})$. Notice that $\mathcal{O}_{\mathcal{R}}$ is a *calibration transformation* as defined above, because it takes the recognizer's posterior as input and outputs another posterior.

We already know by the data processing inequality that by computing $\mathbf{r}$, we may have lost some information, so that $\bar{C}(\mathcal{O}_{\mathcal{R}}) \geq \bar{C}(\mathcal{O}_X)$. But $\mathcal{O}_{\mathcal{R}}(\mathbf{r}, \boldsymbol{\pi})$ is the best we can do with input $\mathbf{r}$, so that using $\mathbf{r}$ itself may be a worse recognizer. These two facts can be summarized as:

$$\bar{C}(\mathcal{R}|\mathcal{O}) \geq \bar{C}(\mathcal{O}_{\mathcal{R}}) \geq \bar{C}(\mathcal{O}_X) \geq 0\,. \tag{2.42}$$

If $\mathcal{O}_{\mathcal{R}}(\mathbf{r}, \boldsymbol{\pi}) = \mathbf{r}$, for every $\mathbf{r}$ and $\boldsymbol{\pi}$ then, according to the observer, $\mathcal{R}$ is *perfectly calibrated.* In this case, $\bar{C}(\mathcal{R}|\mathcal{O}) = \bar{C}(\mathcal{O}_{\mathcal{R}})$, for every prior and cost function.

More generally, we want a *degree of goodness* of calibration. We do this by defining the following *loss* quantities:

$$L_{\text{tot}} = L_{\text{cal}} + L_{\text{ref}} = \bar{C}(\mathcal{R}|\mathcal{O})\,, \tag{2.43}$$
$$L_{\text{cal}} = \bar{C}(\mathcal{R}|\mathcal{O}) - \bar{C}(\mathcal{O}_{\mathcal{R}})\,, \tag{2.44}$$
$$L_{\text{ref}} = \bar{C}(\mathcal{O}_{\mathcal{R}}) \leq L_{\text{def}}\,, \tag{2.45}$$
$$L_{\text{def}} = \bar{C}(\mathcal{R}_0|\mathcal{O}) = C^*(\boldsymbol{\pi})\,. \tag{2.46}$$

Here

L_{tot}, the *total loss*, is our original, undecomposed, posterior Bayes risk (cross-entropy) evaluation criterion.

L_{ref}, the *refinement loss* (posterior entropy), is the loss incurred by having the information in $\mathbf{r}$, rather than the complete information we would have had if θ itself were available.

L_{cal}, the *calibration loss* (the object of the whole exercise), is the additional loss due to imperfect calibration, incurred when using $\mathbf{r}$ to make Bayes decisions. Note that L_{cal} can be interpreted as a *divergence* between the observer's and recognizer's posterior distributions. In fact, for the logarithmic cost function, this is the Kullback-Liebler, or KL-divergence [28].

L_{def} (prior entropy) is the reference value obtained by evaluating the *default recognizer*, $\mathcal{R}_0$, which always outputs the prior independently of the input: $\mathcal{R}_0(x, \boldsymbol{\pi}) = \boldsymbol{\pi}$. Notice that although the default recognizer discards all relevant information in the data (by not processing it), it is nevertheless perfectly calibrated.

All of these losses are non-negative. By (2.32), L_{def} serves as upper bound for L_{ref}. In general, there is no upper bound for L_{cal} or for L_{tot}, unless the cost function C is bounded above.

We can interpret the generalized mutual information, $L_{\text{def}} - L_{\text{ref}}$, as the amount of relevant information that $\mathbf{r}$ carries about the unknown θ. L_{cal} is the amount of information that is lost due to bad calibration. Note the calibration loss can be larger than $L_{\text{def}} - L_{\text{ref}}$, so that the total effective information gain, $L_{\text{def}} - L_{\text{tot}}$, is negative. This happens when the calibration is so bad that the decisions give larger cost than the default recognizer.

In summary: If $L_{\text{cal}} = 0$, calibration is *perfect.* If L_{cal} is small, calibration is *good.* If $L_{\text{cal}} \leq L_{\text{def}} - L_{\text{ref}}$, calibration is *useful.* If $L_{\text{cal}} > L_{\text{def}} - L_{\text{ref}}$ then calibration is *bad.*

Example of bad calibration

The above equations show that perfect calibration exists. Here we show by example that at the other end of the scale, calibration can also be arbitrarily bad. Take the example of a potentially good recognizer with $L_{\text{ref}} \ll L_{\text{def}}$, but where the components of the posterior $\mathbf{r}$ have been permuted. Since the permutation is invertible, L_{ref} remains unaffected, but Bayes decisions will be adversely affected, so that for any cost function with high relative penalties we will find: $L_{\text{tot}} \gg L_{\text{def}} \gg L_{\text{ref}}$.

To see this, consider a good two-class recognizer that outputs $\mathbf{r} = (1-\alpha, \alpha)$ whenever θ_1 is true and $(\alpha, 1-\alpha)$ whenever θ_2 is true, for some small $0 < \alpha \ll 1$. If we evaluate with $C_{\log}$ (which we show later to be a combination of cost functions, including some with very high cost ratios), then $L_{\text{tot}} = -\log(1-\alpha) \approx \alpha$, is small. But if we permute the components of $\mathbf{r}$, then $L_{\text{tot}} = -\log(\alpha)$ is *large* for small α. Since $L_{\text{cal}} \geq L_{\text{tot}} - L_{\text{def}}$ this means L_{cal} is also large.

The beauty and the difficulty

In practice, for the special observer that plays the role of evaluator, the probability distributions are going to be conditioned on available evaluation data. If one chooses to let $P(x|\theta, \mathcal{O})$ represent some unknown distribution from which the evaluation data was drawn, then since L_{tot} is the expected value (2.17) of a function of the evaluation data, it can be approximated as an *average* over a sufficiently large database. Since the function of the data, $C^*(\mathcal{R}(x, \boldsymbol{\pi})|\theta)$, is independent of $\mathcal{O}$, this plan works *regardless* of the details of the probability model, as long as we assume the law of large numbers holds for the model. This is the beauty of evaluation by proper scoring rule.

This blind averaging plan unfortunately does not also work for computing L_{ref}, because it is the expected value (2.29) of the function $C^*\big(\mathcal{O}_{\mathcal{R}}(\mathbf{r}, \boldsymbol{\pi})\big)$, which is defined by the observer's probability model. In other words, L_{tot} can be computed in a purely data-driven manner, but L_{ref} is also model-dependent.

2.4.2 Bad calibration is obvious, regardless of the definition of calibration

Finally, note that bad calibration can be spotted given only L_{tot}, without having to evaluate L_{ref}, because:

$$L_{\text{cal}} \geq L_{\text{tot}} - L_{\text{def}}\,. \tag{2.47}$$

If L_{tot} is large it can be attributed to bad calibration, independently of the evaluator's probability model.

2.5 Summary

In this chapter we showed that the *risk* of using a probability distribution to make a Bayes decision obeys similar inequalities to Shannon's *entropy*. The Bayes risk can therefore be interpreted as a *generalized* entropy.

The risk of making decisions with the posteriors computed by a recognizer, $\mathcal{R}$, is quantified by a given observer, $\mathcal{O}$, as the *posterior cross-entropy*, or equivalently *posterior Bayes risk*, $\bar{C}(\mathcal{R}|\mathcal{O})$. We shall base the rest of this work on $\bar{C}(\mathcal{R}|\mathcal{O})$, using it as our *primary evaluation criterion.* We also introduced the concept of calibration as an *auxiliary evaluation criterion.*

In this chapter, these criteria were based on the hypothetical observer, $\mathcal{O}$, whose role it was to define the necessary probability distributions. In chapter 5 we propose a special observer, called the *evaluator*, which leads to a practical evaluation recipe.

However, before we do that, we need to take care of some technical details of cost functions in chapter 3 and of the recognizer in chapter 4.

Chapter 3

Proper Scoring Rules

Proper scoring rules were briefly introduced in the previous chapter. In the following chapters, they will play a central role, because we shall use them to evaluate the goodness of the probabilistic outputs of pattern recognizers. The purpose of this chapter is to properly introduce them and to discuss some of their properties that we shall need later.

The idea of proper scoring rules dates back at least to Brier [3], who proposed the Brier score as a means of evaluating the goodness of probabilistic weather forecasts in 1950, and Good [4], who proposed the logarithmic score in 1952. The term *proper* is attributed to Winkler and Murphy [60]. For a wide-ranging review see [8].

Bayes decisions and proper scoring rules are two sides of the same coin. In section 2.1.5 we showed that all Bayes decisions form proper scoring rules. Below we mention a result from the literature that shows the converse is also true, that all proper scoring rules can be interpreted as Bayes decisions. This equivalence is one of the ground principles on which this work is based. Our study of Bayes decisions helped to keep the proposed evaluation methodology relevant to real applications. Our study of proper scoring rules helped (i) to understand calibration and (ii) to introduce a much richer variety of cost functions. Pure application-centred reasoning would probably not have led to considering strict cost functions, like logarithmic cost. In this chapter we will close the loop by showing that logarithmic cost has a very close connection to simple applications that require hard decisions.

In the rest of this chapter, we define proper scoring rules, discuss key properties, establish equivalence with Bayes decisions, explore equivalence between proper scoring rules and show that combinations of proper scoring rules form new and useful proper scoring rules.

3.1 Definition

In the literature, a *proper scoring rule* is defined as a special cost function, $f(\mathbf{q}|\theta)$, that satisfies

$$\mathbf{p} \in \arg\min_{\mathbf{q}} \Big\langle f(\mathbf{q}|\theta) \Big\rangle_{\theta|\mathbf{p}} \quad \text{equivalently } \Big\langle f(\mathbf{p}|\theta) \Big\rangle_{\theta|\mathbf{p}} \leq \Big\langle f(\mathbf{q}|\theta) \Big\rangle_{\theta|\mathbf{p}} \tag{3.1}$$

for any $\mathbf{p}, \mathbf{q} \in \mathbb{P}_N$. Moreover, if $\mathbf{p}$ is the unique minimizer, then f is called a *strictly proper scoring rule.* Expressed in the argmin* notation, a strictly proper scoring rule satisfies

$$\mathbf{p} = \operatorname*{argmin^*}_{\mathbf{q}} \Big\langle (\mathbf{q}|\theta) \Big\rangle_{\theta|\mathbf{p}} . \tag{3.2}$$

A proper scoring rule 'scores' the goodness of $\mathbf{q}$, given the truth reference θ and it is called 'proper' by virtue of satisfying condition (3.1). We illustrate this condition with some examples and then discuss some important properties.

3.1.1 Examples

The definition is clarified with three examples of proper scoring rules. We let $N = 2$ and $\mathbf{q} = (q, 1-q)$ and $\mathbf{p} = (p, 1-p)$.

Logarithmic score, for two classes, can be expressed as: $C_{\log}(q|\theta_1) = -\log(q)$ and $C_{\log}(q|\theta_2) = -\log(1-q)$. The expectation w.r.t. $\mathbf{p}$ is $\langle C_{\log}(\mathbf{q}|\theta)\rangle_{\theta|\mathbf{p}} = -p\log(q) - (1-p)\log(1-q)$. This can be minimized w.r.t. q, by differentiating w.r.t. q and setting the derivative to zero. The unique minimum is found at $q = p$, which shows that $C_{\log}$ is a strictly proper scoring rule.

Brier score, defined as $C_{\text{Brier}}(q|\theta_1) = (1-q)^2$ and $C_{\text{Brier}}(q|\theta_2) = q^2$ can be analysed in the same way to show that it is also strictly proper.

Zero-one score, C^*_{err}, derived by (2.9) from zero-one cost, is a proper scoring rule, but it is not strict, because minimization of its expected value behaves[1] as follows:

$$\arg\min_q C^*_{\text{err}}(q|p) = \begin{cases} \big[0, \frac{1}{2}\big) & \text{if } p \in \big[0, \frac{1}{2}\big)\,, \\ \big[0, 1\big] & \text{if } p = \frac{1}{2}\,, \\ \big[\frac{1}{2}, 1\big] & \text{if } p \in \big(\frac{1}{2}, 1\big]\,. \end{cases} \tag{3.3}$$

Note that in each case $p \in \arg\min_q C^*_{\text{err}}(q|p)$, as required by (3.1).

[1]with tie-breaker $b\big(\{\theta_1, \theta_2\}\big) = \theta_1$

3.2 Key Properties

Here we show that proper scoring rules have two desirable properties as evaluation criteria. These properties are discussed in many sources, see for example [7], chapter 13, the section entitled 'The honest weatherman'.

3.2.1 Encouragement of honesty: minimization of calibration loss

Consider a weather forecaster who has computed, to the best of his ability, a probability, p, for rain tomorrow. It is required of the forecaster to report *some* probability, q, for rain tomorrow, but it may not be clear to the forecaster that the best output would indeed be $q = p$, or whether the forecaster himself, or the consumers of his forecast, could not perhaps derive greater benefit if q were modified (warped, or re-calibrated) in some way. If however, the forecaster knows that his reported forecast, q, will be evaluated as $f(q|\theta)$, tomorrow when it will be known whether it rains (θ_1) or not (θ_2), then he will compute his own expected evaluation result as $\langle f(q|\theta)\rangle_{\theta|p}$. If f is a proper scoring rule, he will optimize his own expected evaluation result by choosing to report $q = p$. The proper scoring rule therefore encourages him to honestly report what he calculated and not to try to game the system by reporting something else.

The same result would be achieved if a conscientious weather forecaster (who is not necessarily subject to such explicit evaluation) believes that her consumers would effectively use her report, q, to make Bayes decisions to determine their plans for tomorrow. Then by reporting $q = p$, she would be maximizing her expectation of the benefit to her consumers.

Actually, the above motivations of the male and female forecasters are equivalent, because as we showed in the previous chapter, if q is evaluated as the cost of the Bayes decision it makes, then that forms a proper scoring rule.

Recalling the calibration-refinement decomposition of section 2.4.1, this *honesty inducing* effect of proper scoring rules corresponds to optimization of calibration—the forecaster minimizes the calibration loss by reporting $q = p$. But the forecaster is further motivated by the scoring rule to also make p as good a forecast as possible, where the goodness of p corresponds to the posterior entropy (refinement loss). This is explained below.

3.2.2 Encouragement of diligence: minimization of refinement loss

We continue the example of the weather forecaster. The forecaster has available a prior for rain, π, deduced from the climatological average for that time of the year. The prior could serve as a default, lazy forecast. A forecaster who has no data to process (or who neglects to do so) could satisfy the calibration-minimizing property of the proper scoring rule by just reporting $q = \pi$.

In contrast, a diligent forecaster would process some data, x, gathered from a variety of meteorological sensors and process it through his forecasting model, $\mathcal{R}$, to calculate the posterior, $p = P(\text{rain}|x, \pi, \mathcal{R})$. Being diligent, the forecaster would also ask: Which forecast is better, the prior or the posterior? If goodness is measured by proper scoring rule and if the forecaster evaluates himself by equating $\mathcal{R} = \mathcal{O}$, then the answer is given by the result of section 2.2.7: the posterior has better expected cost than the prior. Thus diligence is encouraged by the proper scoring rule.

3.3 Equivalence between proper scoring rules and Bayes decisions

For any cost function, the cost of the associated Bayes decision forms a proper scoring rule. Conversely, every proper scoring rule can be interpreted as the cost of a Bayes decision made with some cost function.

3.3.1 Construction via Bayes decisions

As shown in the previous chapter, a proper scoring rule can be derived form any cost function by making minimum-expected-cost Bayes decisions. For convenience, we repeat (2.9) here.

Start with some cost function $C(a|\theta)$. Recalling the short-hand notation, $C(a|\mathbf{q}) = \sum_{i=1}^{N} q_i C(a|\theta_i)$, the derived cost function, C^*, is defined as:

$$C^*(\mathbf{q}|\theta_i) = C\bigl(\operatorname*{argmin^*}_a C(a|\mathbf{q})\big|\theta_i\bigr)\,. \tag{3.4}$$

This transformation takes a given cost function, C, on $\mathcal{A} \times \boldsymbol{\Theta}_N$ and creates a new cost function C^* on $\mathbb{P}_N \times \boldsymbol{\Theta}_N$. By inequality (2.13), we see C^* satisfies (3.1) and is therefore a proper scoring rule.

Idempotence of transformation

Let the cost function, $C(\mathbf{q}|\theta)$, be a strictly proper scoring rule, so that by definition it satisfies $\mathbf{q} = \operatorname{argmin^*}_{\mathbf{q}'} C(\mathbf{q}'|\mathbf{q})$. If we apply (3.4) to derive C^*, we find $C^*(\mathbf{q}|\theta) = C(\mathbf{q}|\theta)$. A strictly proper scoring rule is therefore left unchanged by transformation (3.4). An example is: $C^*_{\text{log}} = C_{\text{log}}$.

This also means (trivially) that any strictly proper scoring rule can be interpreted as having been constructed via (3.4).

3.3.2 All proper scoring rules are Bayes decisions

The above remark establishes that all strictly proper scoring rules can be expressed as the cost of some Bayes decision. This is true in general also of non-strict proper scoring rules.

In [8], theorem 2, it is shown that every generalized entropy function (recall our section 2.1.6) defines at least one proper scoring rule and conversely that any proper scoring rule can be defined by a generalized entropy function. The link between entropy functions and proper scoring rules involves generalized differentiation, involving sub-gradients, the details of which need not concern us here. Since generalized entropy is just minimum Bayes risk, this establishes that any proper scoring rule can be interpreted as the cost of some Bayes decision.

3.4 Equivalence between cost functions

Here we analyse a different equivalence that exists between proper scoring rules and indeed between cost functions in general. Apparently different cost functions can give the same Bayes decision for every probability distribution. If this is the case, we consider them to be *equivalent* for our purposes, because when used to construct evaluation criteria, the equivalent cost functions and the associated equivalent proper scoring rules will lead to equivalent evaluation results. If one is faced with the problem of choosing which cost function to use for evaluation, it is important to understand this equivalence. This concept of equivalence is based on DeGroot's book [56].

Cost function equivalence is what allows us, without loss of generality, to impose on all our cost functions the condition (2.3). To show this, let C_{eq} be a cost function for which (2.3) is not necessarily true. We do however require that C_{eq} is bounded from below. Now define the new cost function:

$$C(a|\theta_i) = k_0 C_{\text{eq}}(a|\theta_i) + k_i \tag{3.5}$$

where $0 < k_0 < \infty$ and $-\infty < k_i < \infty$ for $i = 1, 2, \ldots, N$, and notice that this transformation does not alter the Bayes decision:

$$\operatorname*{argmin^*}_a \sum_{j=1}^{N} q_j C(a|\theta_j) = \operatorname*{argmin^*}_a \sum_{j=1}^{N} q_j C_{\text{eq}}(a|\theta_j) \tag{3.6}$$

Now we examine the proper scoring rule constructed from C via (3.4):

$$\begin{aligned} C^*(\mathbf{q}|\theta_i) &= C\bigl(\operatorname*{argmin}_a{}^* \sum_{j=1}^N q_j C(a|\theta_j)\big|\theta_i\bigr) \\ &= k_0 C_{\text{eq}}\bigl(\operatorname*{argmin}_a{}^* \sum_{j=1}^N q_j C_{\text{eq}}(a|\theta_j)\big|\theta_i\bigr) + k_i \\ &= k_0 C^*_{\text{eq}}(\mathbf{q}|\theta_i) + k_i \end{aligned} \tag{3.7}$$

from which we see that the proper scoring rules are related in the same way as the cost functions. Finally, we plug C^* into our Bayes risk criterion (2.16) for the evaluation of a recognizer $\mathcal{R}$:

$$\begin{aligned} \bar{C}(\mathcal{R}|\mathcal{O}) &= \Bigl\langle C^*(\mathcal{R}(x,\boldsymbol{\pi})|\theta)\Bigr\rangle_{x,\theta|\boldsymbol{\pi},\mathcal{O}} \\ &= k_0 \Bigl\langle C^*_{\text{eq}}(\mathcal{R}(x,\boldsymbol{\pi})|\theta)\Bigr\rangle_{x,\theta|\boldsymbol{\pi},\mathcal{O}} + \sum_{i=1}^N \pi_i k_i \\ &= k_0 \bar{C}_{\text{eq}}(\mathcal{R}|\mathcal{O}) + \sum_{i=1}^N \pi_i k_i \end{aligned} \tag{3.8}$$

so that the criteria $\bar{C}$ and $\bar{C}_{\text{eq}}$ are related via scaling and shift (just like Fahrenheit and Celsius).

The main result of this section is now that, if C and C_{eq} are related as in (3.5), then $\bar{C}(\mathcal{R}|\mathcal{O})$ and $\bar{C}_{\text{eq}}(\mathcal{R}|\mathcal{O})$ are *equivalent* criteria for the evaluation of $\mathcal{R}$. By equivalent we mean that if we compare two recognizers, $\mathcal{R}$ and $\mathcal{R}'$, then

$$\bar{C}(\mathcal{R}|\mathcal{O}) \le \bar{C}(\mathcal{R}'|\mathcal{O}) \quad \text{if and only if} \quad \bar{C}_{\text{eq}}(\mathcal{R}|\mathcal{O}) \le \bar{C}_{\text{eq}}(\mathcal{R}'|\mathcal{O})\,. \tag{3.9}$$

3.4.1 Canonical form for cost functions

Since the choice of k_i is irrelevant for evaluation, we can set each k_i to a convenient value. We therefore use (3.5) to transform an arbitrary cost function C_{eq}, which is bounded from below, but for which (2.3) may not be true, to a cost function C for which (2.3) is true, by choosing for every $i = 1, 2, \dots, N$:

$$k_i = -\min_{a\in\mathcal{A}} C_{\text{eq}}(a|\theta_i)\,. \tag{3.10}$$

If we require all our cost functions to satisfy (2.3), then we have already eliminated N degrees of freedom from the cost function definition. Then, there is one more degree of freedom as represented by the scale factor, k_0. If we additionally assume some condition that standardizes the scale of the cost function, then we can also eliminate this degree of freedom.

Example: Normalized error-weighted cost

Here we generalize zero-one cost for the case $N = 2$, by assigning different weights to each of the *four* possible outcomes. Then we show that the canonical form gives a *one*-parameter family of cost functions.

The particular normalization we introduce here to obtain a canonical form is original and as we show in chapter 7, leads to a new characterization of proper scoring rules for two classes.

If we have two propositions, $\mathbf{\Theta}_N = \{\theta_1, \theta_2\}$, and we identify the action set $\mathcal{A}$ with $\mathbf{\Theta}_N$, then there are four possible outcomes, of which two are correct decisions and two are errors. If we weight each outcome with a different parameter, then we get a four-parameter family of cost functions. We assume for each i, that if θ_i is true, then choosing θ_i has a smaller cost than choosing the other proposition. If we now use (3.10) to enforce (2.3), then the weights for correct decisions become zero, leaving us with just two parameters, namely a positive weight for each error. Here we follow the standard speaker recognition terminology and denote these two costs as C_{miss} and C_{fa}.

Now, we can also impose some convenient constraint on the scale. For our purposes, the most convenient constraint is to let the expected cost (of any decision) at the Bayes decision threshold be unity: $\eta C_{\text{miss}} = (1-\eta)C_{\text{fa}} = 1$, for some $0 < \eta < 1$. This gives $C_{\text{miss}} = \frac{1}{\eta}$ and $C_{\text{fa}} = \frac{1}{1-\eta}$, where η is the Bayes decision threshold. Notice that when η approaches either 0 or 1, then one of the costs becomes arbitrarily large, while the other cost is close to unity.

We have now derived a parametric cost function family, C_η, with parameter $0 < \eta < 1$, where for $i, j \in \{1, 2\}$:

$$C_\eta(\theta_j|\theta_i) = \begin{cases} 0 & \text{if } i = j\,, \\ \frac{1}{\eta} & \text{if } i = 1 \text{ and } j = 2\,, \\ \frac{1}{1-\eta} & \text{if } i = 2 \text{ and } j = 1\,. \end{cases} \tag{3.11}$$

The associated proper scoring rule, C^*_η, expressed in terms of the unit step, u, is:

$$C^*_\eta(q|\theta_1) = \frac{1 - u(q-\eta)}{\eta}\,, \qquad C^*_\eta(q|\theta_2) = \frac{u(q-\eta)}{1-\eta} \tag{3.12}$$

where $q = P(\theta_1|\mathbf{q})$ and where η is the Bayes decision threshold: θ_1 is recognized if $q \geq \eta$, and θ_2 otherwise.[2]

Finally, notice that C_η and zero-one cost, C_{err} are related as

$$C_{\text{err}}(q|\theta) = \frac{1}{2} C_{\frac{1}{2}}(q|\theta)\,. \tag{3.13}$$

In what follows, we shall make extensive use of C_η as a representative cost function for two-class recognition problems.

[2] with tie-breaker $b\big(\{\theta_1, \theta_2\}\big) = \theta_1$

3.5 Combinations of proper scoring rules

Here we discuss *combinations of proper scoring rules*, which we show are also proper scoring rules. Such combinations are naturally constructed via *compound decisions*. These combinations will play an important role in the rest of this work.

3.5.1 Compound decisions

Consider the minimization of the expected value of the weighted sum of K different cost functions, w.r.t. a *compound decision*, $\mathbf{a} = (a_1, a_2, \dots, a_K) \in \mathcal{A}^K$ and notice that:

$$\begin{aligned}\min_{\mathbf{a}\in\mathcal{A}^K} \Big\langle \sum_{k=1}^{K} \rho_k C_k(a_k|\theta) \Big\rangle_{\theta|\mathbf{q}} &= \min_{\mathbf{a}\in\mathcal{A}^K} \sum_{k=1}^{K} \rho_k C_k(a_k|\mathbf{q}) \\ &= \sum_{k=1}^{K} \rho_k \min_{a_k\in\mathcal{A}} C_k(a_k|\mathbf{q})\end{aligned} \tag{3.14}$$

provided the weights are positive: $\rho_k > 0$.

If we make compound Bayes decisions, we need to re-examine the behaviour of argmin*, which was introduced in section 2.1.3. We now adopt the convention that argmin* for the compound decision behaves as follows:

$$\begin{aligned}&\operatorname*{argmin^*}_{\mathbf{a}\in\mathcal{A}^K} \Big\langle \sum_{k=1}^{K} \rho_k C_k(a_k|\theta) \Big\rangle_{\theta|\mathbf{q}} \\ &= \big(\operatorname*{argmin^*}_{a\in\mathcal{A}} C_1(a|\mathbf{q}), \dots, \operatorname*{argmin^*}_{a\in\mathcal{A}} C_K(a|\mathbf{q})\big) .\end{aligned} \tag{3.15}$$

This is achieved by defining the tie-breaker for the compound decision as:

$$b(\mathcal{A}^*_{\mathbf{q},1} \times \cdots \times \mathcal{A}^*_{\mathbf{q},K}) = \big(b_1(\mathcal{A}^*_{\mathbf{q},1}), \dots, b_K(\mathcal{A}^*_{\mathbf{q},K})\big) \tag{3.16}$$

where $b_k(\mathcal{A}^*_{\mathbf{q},k})$ is the tie-breaker associated with C_k.

Decision functions

We can generalize the above by letting the compound decisions be *functions*. Let $\mathcal{K}$ be some subset of R^N. In this chapter we shall consider an example where $\mathcal{K}$ is the real interval, $(0, 1)$, and in a later chapter we shall generalize this to the interior of the simplex $\mathbb{P}_N$. Let $\mathcal{A}^{\mathcal{K}}$ denote the set of functions from $\mathcal{K}$ to $\mathcal{A}$. Let some family of cost functions, $C_{\boldsymbol{\tau}}$, be parametrized by $\boldsymbol{\tau} \in \mathcal{K}$ and let $\rho(\boldsymbol{\tau}) > 0$ be some weighting function over the parameters. We minimize the expected cost of the *continuous combination* over $\boldsymbol{\tau}$, by pointwise minimization at every η:

$$\begin{aligned}\min_{\mathsf{a}\in\mathcal{A}^{\mathcal{K}}}\left\langle\int_{\mathcal{K}}\rho(\boldsymbol{\tau})C_{\boldsymbol{\tau}}\big(\mathsf{a}(\boldsymbol{\tau})\big|\theta\big)\,\mathbf{d}\boldsymbol{\tau}\right\rangle_{\theta|\mathbf{q}} &= \min_{\mathsf{a}\in\mathcal{A}^{\mathcal{K}}}\int_{\mathcal{K}}\rho(\boldsymbol{\tau})C_{\boldsymbol{\tau}}\big(\mathsf{a}(\boldsymbol{\tau})\big|\mathbf{q}\big)\,\mathbf{d}\boldsymbol{\tau} \\ &= \int_{\mathcal{K}}\rho(\boldsymbol{\tau})\min_{a\in\mathcal{A}}C_{\boldsymbol{\tau}}(a|\mathbf{q})\,\mathbf{d}\boldsymbol{\tau}\end{aligned} \tag{3.17}$$

and we define argmin* for continuous combination, such that:

$$\mathsf{a}^* = \operatorname*{argmin}^*_{\mathsf{a}\in\mathcal{A}^{\mathcal{K}}}\int_{\mathcal{K}}\rho(\boldsymbol{\tau})C_{\boldsymbol{\tau}}\big(\mathsf{a}(\boldsymbol{\tau})\big|\mathbf{q}\big)\,\mathbf{d}\boldsymbol{\tau} \tag{3.18}$$

is given by

$$\mathsf{a}^*(\boldsymbol{\tau}) = \operatorname*{argmin}^*_{a\in\mathcal{A}} C_{\boldsymbol{\tau}}(a|\mathbf{q})\,. \tag{3.19}$$

We shall refer to a as a *decision function.* This can be understood in two ways: (i) the whole function a is the result of a compound decision, or (ii) $\mathsf{a}(\boldsymbol{\tau})$ gives a simple decision for every $\boldsymbol{\tau}$.

3.5.2 Closure under combination

Let the compound decision $\mathbf{a} = (a_1,\dots,a_K) \in \mathcal{A}^K$ be evaluated with a weighted combination of K different cost functions C_k:

$$C_\rho(\mathbf{a}|\theta) = \sum_{k=1}^{K}\rho_k C_k(a_k|\theta) \tag{3.20}$$

where $\rho_k > 0$. Now we apply (3.4) to form C_ρ^* and then use (3.15) to find:

$$\begin{aligned}C_\rho^*(\mathbf{q}|\theta) &= C_\rho\big(\operatorname*{argmin}^*_{\mathbf{a}\in\mathcal{A}^K} C_\rho(\mathbf{a}|\mathbf{q})\big|\theta\big) \\ &= \sum_{k=1}^{K}\rho_k C_k\big(\operatorname*{argmin}^*_{a\in\mathcal{A}} C_k(a|\mathbf{q})\big|\theta\big) \\ &= \sum_{k=1}^{K}\rho_k C_k^*(\mathbf{q}|\theta)\,.\end{aligned} \tag{3.21}$$

This shows that proper scoring rules are closed under conical[3] combination: A positively weighted combination of proper scoring rules is also a proper scoring rule. Moreover, compound decisions are a natural way to construct such combinations of existing scoring rules.

[3] *Conical* combination is linear combination with weights that are constrained to be positive. Compare this to convex combination, where the weights are positive and sum to one.

Proper scoring rule combination tends to reduce the size of the subsets that minimize the expectation. Let the subset that minimizes $C_k^*(\mathbf{q}|\mathbf{p}) = \left\langle C_k^*(\mathbf{q}|\theta)\right\rangle_{\theta|\mathbf{p}}$ be denoted as:

$$\mathbb{P}_{\mathbf{p},k} = \arg\min_{\mathbf{p}\in\mathbb{P}_N} C_k^*(\mathbf{q}|\mathbf{p}) \tag{3.22}$$

then the subset that minimizes the expectation of the combination C_ρ^* is the intersection:

$$\mathbb{P}_{\mathbf{p},\rho} = \arg\min_{\mathbf{p}\in\mathbb{P}_N} C_\rho^*(\mathbf{q}|\mathbf{p}) = \mathbb{P}_{\mathbf{p},1} \cap \mathbb{P}_{\mathbf{p},2} \cap \cdots \cap \mathbb{P}_{\mathbf{p},K}\,. \tag{3.23}$$

We shall explore this effect in the examples below, where we show that a *continuous combination* over a family of simple, non-strict scoring rules gives a strict rule.

Continuous combination

A continuous combination results when constructing a proper scoring rule from function-valued compound decisions. Let some decision function, $\mathsf{a} \in \mathcal{A}^{\mathcal{K}}$, be evaluated by the cost function:

$$C_\rho(\mathsf{a}|\theta) = \int_{\mathcal{K}} \rho(\boldsymbol{\tau}) C_{\boldsymbol{\tau}}(\mathsf{a}(\boldsymbol{\tau})|\theta)\, \mathbf{d}\boldsymbol{\tau} \tag{3.24}$$

where $\rho(\boldsymbol{\tau}) > 0$. Forming the proper scoring rule, C_ρ^*, by using (3.19) we find:

$$C_\rho^*(\mathbf{q}|\theta) = \int_{\mathcal{K}} \rho(\boldsymbol{\tau}) C_{\boldsymbol{\tau}}^*(\mathbf{q}|\theta)\, \mathbf{d}\boldsymbol{\tau}\,. \tag{3.25}$$

3.5.3 Examples

Here we make use of the parametric family of two-class, non-strict proper scoring rules, C_η^*, as defined by (3.12). The minimization of the expected value, $C_\eta^*(q|p)$, behaves as follows:

$$\arg\min_q C_\eta^*(q|p) = \begin{cases} [0,\eta) & \text{if } p \in [0,\eta)\,, \\ [0,1] & \text{if } p = \eta\,, \\ [\eta,1] & \text{if } p \in (\eta,1] \end{cases} \tag{3.26}$$

where $p = P(\theta_1|\mathbf{p})$ and $q = P(\theta_1|\mathbf{q})$. Now consider the combination of two such scoring rules, C_η^* and $C_{\eta'}^*$, where $0 < \eta < \eta' < 1$. The minimization of the expected value of this combination behaves as follows:

$$\arg\min_q \alpha C_\eta^*(q|p) + \beta C_{\eta'}^*(q|p) = \begin{cases} [0,\eta) & \text{if } p \in [0,\eta)\,, \\ [0,\eta') & \text{if } p = \eta\,, \\ [\eta,\eta') & \text{if } p \in (\eta,\eta')\,, \\ [\eta,1] & \text{if } p = \eta'\,, \\ [\eta',1] & \text{if } p \in (\eta',1] \end{cases} \tag{3.27}$$

for any $\alpha, \beta > 0$.

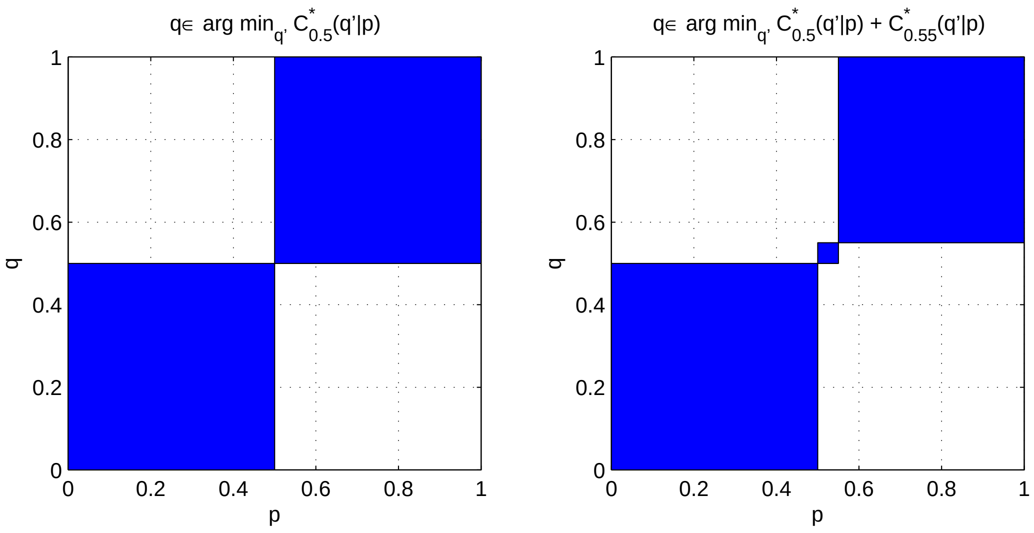

Figure 3.1: Scoring rule combination gives a stricter rule. This is most dramatic for $p \in (0.5, 0.55)$, where the range of q decreases from $[0.5, 1]$ on the left to $[0.5, 0.55)$ on the right.

Figure 3.1 graphically compares the simple scoring rule C^*_η with the combination $C^*_\eta + C^*_{\eta'}$. The shaded area on the left shows combinations of p, q that satisfy (3.26) and on the right (p, q) that satisfy (3.27). Notice in particular that for the combination, when $p \in (\eta, \eta')$, then also $q \in [\eta, \eta')$ and that if we choose the interval (η, η') to be arbitrarily small, then locally in this interval, the scoring rule becomes arbitrarily close to a strict scoring rule. Now intuitively, by combining infinitely many of the C^*_η, with η ranging from 0 to 1, we can make the resultant combination strict over the whole range of p.

In fact, for the two-class case, it is easy to prove with a few inequalities that this is indeed the case: Any continuous combination of the form $\int_0^1 C^*_\eta(q|\theta)\rho(\eta)\, d\eta$, where $\int_0^1 \rho(\eta)\, d\eta < \infty$ and $\rho(\eta) > 0$, forms a *strictly* proper scoring rule.

We illustrate this for the simplest case of $\rho(\eta) = 1$, which gives the logarithmic rule:

$$\begin{aligned}\int_0^1 C^*_\eta(q|\theta_1)\,d\eta &= \int_0^1 \frac{1-u(q-\eta)}{\eta}\,d\eta = \int_q^1 \frac{1}{\eta}\,d\eta = -\log(q)\\ \int_0^1 C^*_\eta(q|\theta_2)\,d\eta &= \int_0^1 \frac{u(q-\eta)}{1-\eta}\,d\eta = \int_0^q \frac{1}{1-\eta}\,d\eta = -\log(1-q)\,.\end{aligned} \tag{3.28}$$

In other words, recalling $\mathbf{q} = (q, 1-q)$ and $C_{\log}(\mathbf{q}|\theta_i) = -\log(q_i)$, we find:

$$\int_0^1 C^*_\eta(q|\theta)\,d\eta = C_{\log}(q|\theta)\,. \tag{3.29}$$

This shows that the logarithmic scoring rule has an interpretation as a continuous combination over simple binary decisions problems. In chapter 8, we show that this generalizes also to multi-class, where $N > 2$.

Finally, it may be helpful to examine this example as a minimization w.r.t. the decision function, $\mathsf{a} : (0,1) \mapsto \{\theta_1, \theta_2\}$. The Bayes decision,

$$\mathsf{a}^*_q = \operatorname*{argmin^*}_{\mathsf{a}\in\mathcal{A}^{\mathcal{K}}} \int_0^1 C_\eta\big(\mathsf{a}(\eta)\big|q\big)\,d\eta \tag{3.30}$$

is given[4] by:

$$\mathsf{a}^*_q(\eta) = \begin{cases}\theta_1, & \text{if } q \geq \eta\,,\\ \theta_2, & \text{if } q < \eta\end{cases} \tag{3.31}$$

and the proper scoring rule formed by this decision is:

$$\int_0^1 C_\eta\big(\mathsf{a}^*_q(\eta)\big|\theta\big)\,d\eta = \int_0^1 C^*_\eta(q|\theta)\,d\eta = C_{\log}(q|\theta)\,. \tag{3.32}$$

3.6 Summary

We showed that our derived cost function, $C^*(\mathbf{q}|\theta)$, which is the cost of using $\mathbf{q}$ to make a Bayes decision, satisfies the classical requirements for it to be a proper scoring rule. Evaluation of a recognizer via C^* encourages the recognizer to output good probability distributions that can be used to make cost-effective Bayes decisions. This evaluation recipe is dependent on which proper scoring rule is used. This chapter examined some of the aspects of how different proper scoring rules are related and how new ones may be constructed by combining existing ones. In particular, we showed how the strictly proper logarithmic scoring rule for two classes can be interpreted as a combination over a range of non-strict scoring rules associated with binary decisions.

[4]with tie-breaker $b(\{\theta_1, \theta_2\}) = \theta_1$

Chapter 4

The Recognizer

In this chapter, we revisit the recognizer in order to add the additional requirement that the recognizer's posterior be calculated according to Bayes' rule.

References for this chapter include any text on probability theory that states Bayes' rule, e.g. [7, 1, 56] and any text that gives the axioms for a vector space, e.g. [61]. The representation of log-likelihoods in an abstract vector space is original.

4.1 Bayes' rule

Thus far, we have required the recognizer to implement the function $\mathcal{R}(x, \boldsymbol{\pi})$, of which we specified the output to have the form of a posterior probability distribution, denoted thus:

$$\mathbf{r} = (r_1, r_2, \dots, r_N) = \mathcal{R}(x, \boldsymbol{\pi}) \,, \tag{4.1}$$

$$r_i = P(\theta_i|\mathbf{r}) = P(\theta_i|x, \boldsymbol{\pi}, \mathcal{R}) \,. \tag{4.2}$$

This requirement was sufficient for our analysis up to this point. Now, we place an additional requirement on how $\mathcal{R}$ is to be implemented, namely that $\mathcal{R}$ should process the prior in accordance with Bayes' rule. Specifically, we require $\mathcal{R}$ to be implemented as the function composition:

$$\mathcal{R}(x, \boldsymbol{\pi}) = \mathcal{B}(\mathcal{W}(x), \boldsymbol{\pi}) \tag{4.3}$$

where $\mathcal{W} : \mathcal{X} \mapsto \mathbb{R}^N$ is a new recognizer function, which maps the input to a vector of log-likelihoods. Bayes' rule is represented by the function $\mathcal{B} : \mathbb{R}^N \times \mathbb{P}_N \mapsto \mathbb{P}_N$, where the components of $\mathbf{r} = \mathcal{B}(\mathbf{w}, \boldsymbol{\pi})$ are:

$$r_i = P(\theta_i|\mathbf{r}) = P(\theta_i|\mathbf{w}, \boldsymbol{\pi}) = \frac{\pi_i \exp(w_i)}{\sum_{j=1}^N \pi_j \exp(w_j)} \,. \tag{4.4}$$

The *log-likelihood vector*, $\mathbf{w} = (w_1, w_2, \dots, w_N) \in \mathbb{R}^N$ is the output of $\mathcal{W}$. The new function $\mathcal{W}$ now defines the recognizer, because $\mathcal{R}$ is completely determined by $\mathcal{W}$. Henceforth we shall use either $\mathcal{R}$, or $\mathcal{W}$, to refer to the recognizer, depending on which is more convenient.

In the next section, we discuss the nature of the recognizer's log-likelihood vector $\mathbf{w}$ in more detail.

4.2 The recognizer's log-likelihoods

The interpretation of $\mathbf{w} = (w_1, w_2, \dots, w_N) = \mathcal{W}(x)$ is based on its purpose, namely to serve as input to Bayes' rule $\mathcal{B}(\mathbf{w}, \boldsymbol{\pi})$ as defined in (4.4).

Component w_i of $\mathbf{w}$ is the recognizer's log-likelihood for the proposition θ_i, given the data x.

The log-likelihood vector is closely related to the posterior distribution, except that it is independent of the prior and thus represents *only the relevant information extracted from the speech by the recognizer.* In contrast, the posterior distribution is the *combination* of the independent prior information, $\boldsymbol{\pi}$, with the speech information, $\mathbf{w}$.

There is another superficial difference between the posterior and $\mathbf{w}$. The posterior is *normalized* in the sense that its components sum to one. In contrast, the log-likelihood vector as defined by (4.4) is not normalized, because it has one degree of freedom in the sense that:

$$\mathcal{B}\big((w_1, w_2, \dots, w_N), \boldsymbol{\pi}\big) = \mathcal{B}\big((w_1 + k, w_2 + k, \dots, w_N + k), \boldsymbol{\pi}\big) \tag{4.5}$$

where k is any real number. Any such log-likelihood offset, k, or equivalently likelihood scale factor, $\exp(k)$, cancels when computing the posterior, or when computing any other ratio of likelihoods. The value of any w_i is meaningless on its own, because of the arbitrary additive constant. It always has to be interpreted relative to one or more of the other components.

Finally, recall that thus far we have required of the recognizer to compute only the posterior and not also to have to assign probability distributions of the form $P(x|\theta_i, \mathcal{R})$. This is still the case. The only purpose of $\mathbf{w}$ is to compute the posterior. Distributions of the form $P(\mathbf{w}|\theta_i, \mathcal{R})$ do not have to be defined by the recognizer.

4.2.1 The vector space of log-likelihood lines

In what follows, especially when analysing calibration functions, it will be convenient to have a somewhat more technical (but also more concise) description of the log-likelihood vectors, which will help us to encapsulate the degree of freedom described by (4.5).

When there are N classes, the log-likelihood vectors live in the N dimensional Euclidean vector space $\mathbb{R}^N$, but they have the above-mentioned one dimensional degree of freedom. If we remove this degree of freedom, the resultant vectors live in $N-1$ dimensions. In speaker detection for example, where

$N = 2$, it is customary to work with the (one dimensional) difference between the two log-likelihoods, which is called the *log-likelihood-ratio.*

We now define a general scheme, for $N \geq 2$, for representing the log-likelihood vectors in an $N-1$ dimensional vector space which we denote[1] $\mathbb{L}_N$. The elements of $\mathbb{L}_N$ are *lines* in $\mathbb{R}^N$ and all of the lines are parallel. For any $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, we define the line, $\acute{\mathbf{w}} \in \mathbb{L}_N$, as the set of points:

$$\acute{\mathbf{w}} = \left\{ (w_1 + k, \ldots, w_N + k) \in \mathbb{R}^N | k \in \mathbb{R} \right\} . \tag{4.6}$$

The notation $\acute{\mathbf{w}}$ can be read as *the log-likelihood line* through $\mathbf{w}$, the direction of all log-likelihood lines being *diagonal*, as determined by (4.6). A given line, $\acute{\mathbf{w}}$, which is represented here by the point $\mathbf{w}$ on the line, can also be represented by any other point, $\mathbf{v}$ on the line. That is to say, $\acute{\mathbf{w}} = \acute{\mathbf{v}}$ if and only if $\mathbf{w} - \mathbf{v} \in \acute{\mathbf{0}}$, the log-likelihood line through the origin of $\mathbb{R}^N$.

The *addition operator* is defined for every $\acute{\mathbf{w}}, \acute{\mathbf{v}} \in \mathbb{L}_N$ as

$$\acute{\mathbf{w}} + \acute{\mathbf{v}} = \acute{\mathbf{s}} , \qquad \text{where} \qquad \mathbf{s} = \mathbf{w} + \mathbf{v} . \tag{4.7}$$

Equivalently, if $\mathbf{x} \in \acute{\mathbf{w}}$ and $\mathbf{y} \in \acute{\mathbf{v}}$, then $\mathbf{x} + \mathbf{y} \in \acute{\mathbf{w}} + \acute{\mathbf{v}}$. The *origin*, or *additive identity* of $\mathbb{L}_N$ is $\acute{\mathbf{0}}$.

Scalar multiplication, with a scalar, $\alpha \in \mathbb{R}$, is defined as:

$$\alpha \acute{\mathbf{w}} = \acute{\mathbf{s}} , \qquad \text{where} \qquad \mathbf{s} = \alpha \mathbf{w} . \tag{4.8}$$

The *dimensionality* of $\mathbb{L}_N$ is $N-1$, because any $\acute{\mathbf{w}} \in \mathbb{L}_N$ can be represented in terms of an $N-1$ dimensional basis, $\{\acute{\mathbf{v}}_1, \ldots, \acute{\mathbf{v}}_{N-1}\}$, as:

$$\acute{\mathbf{w}} = \sum_{i=1}^{N-1} \alpha_i \acute{\mathbf{v}}_i \tag{4.9}$$

provided $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N-1}, (1, 1, \ldots, 1)\}$ is a basis for $\mathbb{R}^N$.

Example: log-likelihood-ratio

We can choose as basis for $\mathbb{L}_2$, the set $\{\acute{\mathbf{v}}_1\}$, where $\mathbf{v}_1 = (1, 0)$. Any given $\acute{\mathbf{w}}$, where $\mathbf{w} = (w_1, w_2)$, is represented as $\acute{\mathbf{w}} = \alpha \acute{\mathbf{v}}_1$, where $\alpha = (w_1 - w_2)$ is the above-mentioned *log-likelihood-ratio.*[2]

Recognizer redefined

For notational convenience in the rest of this work, we now redefine the recognizer function as $\mathcal{W} : \mathcal{X} \mapsto \mathbb{L}_N$, so that the recognizer outputs are, in theory, *lines* in $\mathbb{R}^N$.

[1]Note, $\mathbb{L}_N$ has dimension $N-1$. This is analogous to our notation for the simplex $\mathbb{P}_N$, which is also $N-1$ dimensional.

[2]To see this, solve $\alpha(1 + k, k) = (w_1 + k', w_2 + k')$. The solution, $\alpha = w_1 - w_2$, is independent of k and k'.

In practical implementations the author uses the one dimensional log-likelihood-ratio representation for speaker recognition where $N = 2$, but a redundant, N dimensional, symmetrical log-likelihood vector representation for language recognition where $N > 2$.

Bayes' rule

We now adopt the convention that by $\mathcal{B}(\acute{\mathbf{w}}, \boldsymbol{\pi})$ we mean $\mathcal{B}(\mathbf{v}, \boldsymbol{\pi})$, for any $\mathbf{v} \in \acute{\mathbf{w}}$, all of these values being equal.

This allows us to attach some meaning to the origin and the two operations of the vector space $\mathbb{L}_N$. The origin $\acute{\mathbf{0}}$ has the meaning that is does not modify the prior: $\mathcal{B}(\acute{\mathbf{0}}, \boldsymbol{\pi}) = \boldsymbol{\pi}$. For scalar multiplication, consider any $\acute{\mathbf{w}} \neq \acute{\mathbf{0}}$ and observe that $\mathcal{B}(\alpha\acute{\mathbf{w}}, \boldsymbol{\pi})$ moves closer to $\boldsymbol{\pi}$ as $|\alpha|$ becomes smaller, eventually reaching $\boldsymbol{\pi}$ at $\alpha = 0$. Conversely, increasing $|\alpha|$ moves $\mathcal{B}(\alpha\acute{\mathbf{w}}, \boldsymbol{\pi})$ away from $\boldsymbol{\pi}$, ending up at the boundary of $\mathbb{P}_N$, as $|\alpha| \to \infty$.

For vector addition, let $\bar{\boldsymbol{\pi}} = (\frac{1}{N}, \dots, \frac{1}{N})$ denote the uniform prior and let $\log\acute{\boldsymbol{\pi}}$ denote the log-likelihood line through $(\log \pi_1, \dots, \log \pi_N)$, then $\mathcal{B}(\acute{\mathbf{w}}, \boldsymbol{\pi}) = \mathcal{B}(\acute{\mathbf{w}} + \log\acute{\boldsymbol{\pi}}, \bar{\boldsymbol{\pi}})$. Translation in $\mathbb{L}_N$ has the same effect as modifying the prior.

4.3 Log-likelihood calibration

We conclude this chapter by briefly revisiting the topic of calibration as introduced in section 2.4. We give an equivalent definition of calibration in terms of the log-likelihoods, rather than the posterior. The definitions are equivalent, because when the prior is given, Bayes' rule forms a bijection[3] between log-likelihood line and posterior distribution and we showed in chapter 2 that bijections do not alter information content.

We define a *log-likelihood calibration transformation* to be any function, $f : \mathbb{L}_N \mapsto \mathbb{L}_N$, so that a posterior $\mathcal{B}(\acute{\mathbf{w}}, \boldsymbol{\pi})$ is recalibrated as $\mathcal{B}\big(f(\acute{\mathbf{w}}), \boldsymbol{\pi}\big)$.

In a similar derivation to section 2.4.1, we now regard $\acute{\mathbf{w}} = \mathcal{W}(x)$ as a statistic, for which the hypothetical observer, $\mathcal{O}$, has the likelihoods $\ell_i = P(\acute{\mathbf{w}}|\theta_i, \mathcal{O})$, for $i = 1, \dots, N$. If we let $\boldsymbol{\lambda} = (\log \ell_1, \dots, \log \ell_N)$, then the observer's optimal calibrator, which gives lowest posterior Bayes risk for any prior and cost function, is $f_{\mathcal{O}}(\acute{\mathbf{w}}) = \acute{\boldsymbol{\lambda}}$. A recognizer has *perfect calibration* if its output, $\acute{\mathbf{w}}$, always satisfies $\acute{\mathbf{w}} = f_{\mathcal{O}}(\acute{\mathbf{w}})$. The *default recognizer*, $\mathcal{W}_0$, defined as $\mathcal{W}_0(x) = \acute{\mathbf{0}}$, extracts no information from the data, but nevertheless has perfect calibration.

Because of the above-mentioned information-content equivalence, all of the losses defined in section 2.4.1 could alternatively be defined in terms of the log-likelihood calibration of this section.

[3]See appendix A, which explains the difference between invertible and bijective.

4.4 Summary

In this chapter we revisited the recognizer to add the requirement that the recognizer implements its mapping from prior to posterior in accordance with Bayes' rule. This redefines the recognizer as a calculator of log-likelihoods, rather than as a calculator of posteriors.

Chapter 5

The Evaluator

In this chapter, we use the framework of chapter 2 to derive a practical recipe for the evaluation of the goodness of the class of pattern recognizers of interest.

Thus far, the role of the hypothetical observer, $\mathcal{O}$, has been to have probability distributions for the data, so that we could analyse *qualitatively* the expected consequences of processing data though a recognizer. Here, we consider a special observer called the *evaluator*, denoted $\mathcal{E}$. The role of the evaluator is to judge the goodness of a recognizer *quantitatively*.

5.1 Evaluation by Bayes risk

As explained in chapter 2, we choose as evaluation criterion, the *posterior Bayes risk*. We let the evaluator, $\mathcal{E}$, play the role of the observer, $\mathcal{O}$, in (2.17) in order to judge the decision-making ability of the recognizer as $\bar{C}(\mathcal{R}|\mathcal{E})$. We now switch to a modified notation, which will be more convenient for our further purposes. We represent $\bar{C}(\mathcal{R}|\mathcal{E})$ as:

$$\mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \pi_i \Big\langle C^*_{\boldsymbol{\tau}}\Big(\mathcal{B}\big(\mathcal{W}(x), \boldsymbol{\pi}\big)\Big|\theta_i\Big)\Big\rangle_{x|\theta_i,\mathcal{E}}. \tag{5.1}$$

As before, $C^*_{\boldsymbol{\tau}}$ is the *proper scoring rule* derived via (3.4) from the cost function $C_{\boldsymbol{\tau}}$. We now qualify the cost function with the subscript $\boldsymbol{\tau}$, because in what follows we shall explore different cost functions. For the same reason we also make explicit the dependence on the prior $\boldsymbol{\pi}$. Finally, as discussed in chapter 4, we now refer to the recognizer as $\mathcal{W}$, rather than $\mathcal{R}$, because $\mathcal{R}(x, \boldsymbol{\pi}) = \mathcal{B}\big(\mathcal{W}(x), \boldsymbol{\pi}\big)$. Recall that $\mathcal{B}$ maps the log-likelihoods produced by $\mathcal{W}$ to the recognizer's posterior distribution.

5.2 The evaluator's expectation

We present two alternative motivations for the *same* implementation of our proposed evaluation recipe. The first is motivated as an approximation, the second as an exact calculation. Both motivations are useful as alternative interpretations of the evaluation recipe.

5.2.1 Approximating expected cost by average cost

As mentioned in section 2.4.1, since the evaluation criterion (5.1) is an expected cost, it may be approximated via averaging cost over a given evaluation database, if we assume that the evaluation data was sampled from some unknown[1] probability model $P(x|\theta,\mathcal{E})P(\theta|\boldsymbol{\pi})$. In short, such averaging is exactly what our proposed evaluation recipe (5.4) below does. But first, we will give an alternative motivation for it.

5.2.2 Exact calculation by assigning probability distributions

Here we consider explicitly *assigning*[2] the evaluator's probability distributions $P(\cdot|\theta_i,\mathcal{E})$ and then using them for an exact calculation of the expectation (5.1). To do this, the evaluator apparently has some choices, because of property (2.28) of expectations: It could compute the expectations by assigning class-conditional probability distributions for any of x, $\acute{\mathbf{w}}$, $\mathbf{r}$, or c, where $\acute{\mathbf{w}} = \mathcal{W}(x)$, $\mathbf{r} = \mathcal{B}(\acute{\mathbf{w}},\boldsymbol{\pi})$, and $c = C^*_{\boldsymbol{\tau}}(\mathbf{r}|\theta_i)$. However we rule out all but $\acute{\mathbf{w}}$:

- The input x has the highest dimensionality and most complex form. In the worst case it can be the original speech waveform. To build an evaluator that does probabilistic modelling of x would be even more difficult than building the to-be-evaluated recognizer. Then also, the evaluator typically is not given the evaluator's function, $\mathcal{W}$, it is only given some input-output pairs. This alone effectively rules out using distributions of the form $P(x|\theta_i,\mathcal{E})$ to compute the expectation (5.1).

- The posterior $\mathbf{r}$ depends on the parameter $\boldsymbol{\pi}$. The cost, c, further depends on the details of the cost function, $C_{\boldsymbol{\tau}}$. In this work, we want to be able to freely vary both $\boldsymbol{\pi}$ and $C_{\boldsymbol{\tau}}$ and this should not have complicating implications for the evaluation procedure.

[1]The prior, $\boldsymbol{\pi}$, is considered given, so that only the factor $P(x|\theta,\mathcal{E})$ is unknown.

[2]We follow the recommendation of Jaynes [7] and use the terminology of *assigning*, rather than *estimating* probability distributions. Estimating a probability distribution makes sense only if the probability distribution is interpreted as some unknown random process generating the data. If a probability distribution is interpreted more generally as representing a state of uncertain knowledge, then the probability distribution *itself* is the estimate of the unknown variable (here the recognizer output). This estimate is assigned (or just made) by the evaluator. In particular, the distribution (5.3) that we end up assigning below cannot be interpreted as an estimate.

This leaves $\acute{\mathbf{w}}$, in terms of which we now rewrite (5.1):

$$\mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \pi_i \Big\langle C^*_{\boldsymbol{\tau}}\Big(\mathcal{B}\big(\acute{\mathbf{w}}, \boldsymbol{\pi}\big)\Big|\theta_i\Big)\Big\rangle_{\acute{\mathbf{w}}|\theta_i,\mathcal{E}}. \tag{5.2}$$

The role of the evaluator is now to have a conditional probability distribution of the form $P(\acute{\mathbf{w}}|\theta_i, \mathcal{E})$, for every $i = 1, 2, \dots, N$. For this purpose it has available the resource of the *supervised evaluation database*, on which these distributions should be conditioned.

5.2.3 Evaluation database

The evaluation database has a set of T *trials*, each with an associated speech input, $x_1, x_2, \dots, x_T$. These inputs are made available to $\mathcal{W}$, the recognizer under evaluation, which computes $\acute{\mathbf{w}}_t = \mathcal{W}(x_t)$, for every t. The recognizer outputs are given, in turn, to the evaluator.

For ease of exposition, we make the mild assumption that the speech inputs are all different: $x_1 \neq x_2 \neq \cdots \neq x_T$ and that the processing precision is high enough so that the recognizer's outputs, $\acute{\mathbf{w}}_t$, are also all different.

We assume that for every x_t, *exactly one* of the propositions $\{\theta_1, \theta_2, \dots, \theta_N\}$ is true and also that for every proposition, θ_i, there is at least one (preferably many) x_t for which θ_i is true.

The supervision is given by a partitioning of the set of input indices, $\{1, 2, \dots, T\}$, into N non-empty subsets $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_N$, such that if index $t \in \mathcal{T}_i$, then proposition θ_i is true for input x_t.

Now the problem is, how do we use this database?

5.2.4 The evaluator should not do predictive modelling

There are very many ways to assign probability distributions, given data. It is difficult to choose among them and some of them are difficult to implement. Most of them also require prior assumptions to be made about the data that is being modelled. In fact, methods such as maximum likelihood that do not require explicit prior assumptions, still have implicit assumptions, corresponding to flat priors. Here we are modelling the outputs of the recognizers under evaluation. It may be that the assumptions on which the modelling is based apply to some of the recognizers under evaluation, but not to others. This may lead to unfair comparisons between evaluators.

In summary, we do not want the evaluator to be yet another pattern recognizer which has to do complex modelling of data. Rather, we want the evaluation procedure to be objective, easy to define, easy to understand and easy to apply.

The resolution to this problem is to realize that when we write $P(\acute{\mathbf{w}}|\theta, \mathcal{E})$, which can be rephrased as $P(\acute{\mathbf{w}}|\theta, \text{evaluation database}, \text{some assumptions})$, this is really a *prediction* of what the values of $\acute{\mathbf{w}}$ may be in some other, as yet unseen database. It is this perceived predictive role that is making the evaluation difficult. We now make the following choice as to the purpose of the evaluator:

> The primary purpose of the evaluation is to document the performance of the recognizer on the given, supervised evaluation database, rather than to estimate what the performance could be on some other as yet unseen data.

We argue that measuring performance on a given database is a necessary first step, without which estimates of performance on unseen databases would be undefined. This is in the same spirit as the traditional error-rates obtained by counting errors in a supervised evaluation database—we need to define the error-rate on given data, before we can consider estimates of the error-rate on unseen data. The focus of this work is on defining evaluation criteria as measured on given data, while estimates of the values of these criteria on unseen data are out of scope.

If we are not interested in prediction, but just in the data at hand in the evaluation database, then probability distributions for the data are in fact inappropriate. But we also do not want to just throw away our laboriously constructed probabilistic evaluation criterion! The solution is to use *empirical* distributions, which can be understood as $P(\acute{\mathbf{w}}|\theta, \text{evaluation database}, \acute{\mathbf{w}} \text{ is in the database})$.

5.2.5 Empirical evaluation

The evaluator's empirical probability distribution for $\acute{\mathbf{w}}$ is concentrated as impulses[3] at the values of the given supervised evaluation data:

$$P(\acute{\mathbf{w}}|\theta_i, \mathcal{E}) = \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} \delta(\acute{\mathbf{w}} - \acute{\mathbf{w}}_t) \tag{5.3}$$

where $\mathcal{T}_i$ is the subset of input indices for which θ_i is true and where $|\mathcal{T}_i|$ denotes the subset size. Using this in (5.2), we get:

$$\begin{aligned} \mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}|\boldsymbol{\pi}) &= \left\langle C^*_{\boldsymbol{\tau}}\big(\mathcal{B}(\acute{\mathbf{w}}, \boldsymbol{\pi})\big|\theta\big) \right\rangle_{\acute{\mathbf{w}},\theta|\boldsymbol{\pi},\mathcal{E}} \\ &= \sum_{i=1}^{N} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} C^*_{\boldsymbol{\tau}}\big(\mathcal{B}(\acute{\mathbf{w}}_t, \boldsymbol{\pi})\big|\theta_i\big) . \end{aligned} \tag{5.4}$$

This is finally the practical form of our evaluation objective. It is still general in the sense that the prior and cost function remain unspecified. Later we shall specialize it by making specific choices for $\boldsymbol{\pi}$ and $C_{\boldsymbol{\tau}}$. Some comments are in order:

[3]i.e. Dirac deltas

- The prior $\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_N)$ appears twice in each version of the RHS of (5.4) and indeed it plays two roles: It is the prior for both the evaluator and the recognizer.

- By concentrating the distributions at the values of the evaluation data, the expectation becomes a *weighted average* over the costs of the recognizer's Bayes decisions on the trials of the evaluation database.

- The choice we made above as to what variable to model now becomes irrelevant in any case. We could equivalently have modelled any of the other variables with empirical distributions, with the same numerical result.

- If we use zero-one cost and let $\pi_i = \frac{|\mathcal{T}_i|}{T}$, then (5.4) is just the well-known empirical error-rate as obtained by counting the errors in the decisions made with the recognizer's posterior.

- If one does want to interpret (5.4) as an estimate of the cost of using the recognizer on unseen data, then if one assumes the unseen data is 'similar' to the evaluation data, then the average cost over a sufficiently large evaluation database is not an unreasonable estimate for the average cost on unseen data. This is in fact the interpretation mentioned in section 5.2.1.

The evaluator's posterior

Thus far in this chapter, there was no need to explicitly consider the evaluator's posterior, $P(\theta|\acute{\mathbf{w}}_t, \boldsymbol{\pi}, \mathcal{E})$, because it is not needed to practically compute the evaluation criterion (5.4). However, we will need it when discussing calibration in the next section.

Under the empirical distribution (5.3), and because we assumed every $\acute{\mathbf{w}}_t$ to be unique, this posterior would have *zero entropy*, i.e. it would assign posterior probability of 1 to the true proposition for every t.

5.3 Evaluation of calibration

Recall the calibration-refinement decomposition, $L_{\text{tot}} = L_{\text{ref}} + L_{\text{cal}}$, of section 2.4.1. Here L_{tot} refers to our main evaluation criterion which is computed via (5.4). To judge calibration, L_{cal}, as a secondary evaluation criterion, we additionally need to compute L_{ref}. Unfortunately, both of our interpretations given so far of L_{tot} are unhelpful for defining a meaningful L_{ref}:

- If we use the average-as-approximate-expectation interpretation of section 5.2.1, the probability model remains unknown and undefined and so does L_{ref} and L_{cal}.

- If we use the empirical distribution of section 5.2.5, the evaluator's zero posterior entropy trivializes the calibration-refinement decomposition because now $L_{\text{ref}} = 0$ and $L_{\text{cal}} = L_{\text{tot}}$. The total loss is attributed to calibration. Since L_{tot} is our primary evaluation criterion, this leaves us without a secondary criterion.

To make the calibration-refinement decomposition work, the evaluator needs to explicitly assign probability distributions to the recognizer output, but this cannot be the empirical distribution. This is indeed what was done in [5], where the decomposition was proposed. Their probabilistic model for the posterior probability of a two-class recognizer (or forecaster) was a histogram approach. They quantized the (one dimensional) posterior probability into a small number of discrete bins and counted occurrences under each of the two classes. The problem with this approach is that L_{ref} is entirely dependent on the number of bins in the histogram: For sufficiently fine binning, this reduces to the empirical solution, with $L_{\text{ref}} = 0$, while for the coarsest binning, we reach the maximum entropy at $L_{\text{ref}} = L_{\text{def}}$. How many bins should one use? This effectively still leaves L_{ref} undefined.[4]

So how do we clean up this mess? We have one answer that says $L_{\text{ref}} = 0$ and another that says L_{ref} is undefined. But this is as it should be:

- In the first case, our model is: 'The only values of $(\theta, \acute{\mathbf{w}})$ you will see are those that you have already seen in the database.' In this case, if the values of $\acute{\mathbf{w}}$ in the database are all distinct, then when given one such value of $\acute{\mathbf{w}}$, the evaluator has no further uncertainty about θ, so that $L_{\text{ref}} = 0$.

- In the other case, in the interpretation of section 5.2.1, or indeed in the histogram solution of [35], the evaluator is not committing to any model and L_{ref} remains undefined.

The evaluator's posterior entropy, L_{ref}, or equivalently the amount of relevant information, $L_{\text{def}} - L_{\text{ref}}$, that the evaluator sees in the recognizer output, is not an intrinsic property of the recognizer output. The information gained by revealing the value of a previously uncertain variable depends on what was known beforehand about the variable and what probability model was used for the remaining uncertainty. For example, if a clear proof of P=NP would be revealed to a theoretical computer scientist, she would probably regard that as a very large amount of information.[5] In contrast, for someone[6] who has never heard of P=NP, such a proof would contain only about 1 bit of information.

[4]For a contrast to this conclusion, see [35], where the histogram method of [5] was indeed proposed for use in speaker recognition.

[5]In bits of Shannon entropy this information would be $-\log_2 p$, where p, the probability that such a proof exists and will be found, is very small for most computer scientists.

[6]Speech engineers are seldom concerned with P vs NP. For us the boundary between easy and hard is between linear and polynomial.

Different observers see different amounts of information. These hypothetical observers are not directly relevant to our problem of evaluating the recognizer. We do not want to compare the recognizer under evaluation to some other recognizers somehow constructed by the evaluator. We just want to evaluate how well the recognizer works on the evaluation data.

5.3.1 Calibration redefined

We hereby relinquish the calibration-refinement decomposition, but not calibration. The key to salvaging calibration is to define it slightly differently. Here we propose to use as reference for the decomposition, the best recognizer that could reasonably have been submitted to the evaluation. We let this alternative recognizer be related to the submitted one via a *calibration transformation*, such as defined in sections 2.4 and 4.3.

The difference between the old and new plans is subtle. The idea of an optimal reference recognizer is common to both plans as is the requirement that the submitted and reference recognizers be related via calibration transformation. But we are no longer interpreting the reference recognizer as defined by a probability model somehow assigned by the evaluator. Instead, we constrain the allowed calibration transformations and then optimize L_{tot} subject to this constraint.

An example of an unreasonable calibration transformation that should be disallowed is the one given by the evaluator's empirical distribution. This transformation could not reasonably have been submitted by the evaluee, because it is defined by the full detail of the supervised evaluation database.

For now, let us defer to later chapters detailed specification of reasonable calibration transformations and just denote by Φ the set of allowed transformations. If we work with log-likelihood transformations, then Φ is a set of functions from $\mathbb{L}_N$ to $\mathbb{L}_N$, so that if $f \in \Phi$, then $f(\acute{\mathbf{w}})$ is the recalibrated version of $\acute{\mathbf{w}}$. If the original recognizer is $\mathcal{W}$, then the recalibrated recognizer is denoted $\mathcal{W}_f$, where $\mathcal{W}_f(x) = f\big(\mathcal{W}(x)\big)$. We can now define:

$$L_{\text{tot}} = \mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}|\boldsymbol{\pi}) , \qquad \text{and} \qquad L_{\min} = \min_{f\in\Phi} \mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}_f|\boldsymbol{\pi}) \tag{5.5}$$

where as before, L_{tot} is the primary evaluation criterion for the actual, unmodified recognizer under evaluation, while $L_{\min}$ is the criterion for the recognizer with the best calibration that could reasonably have been submitted. The calibration loss can now be redefined as the non-negative difference, $L_{\text{cal}} = L_{\text{tot}} - L_{\min}$.

We require that $f_0, f_1 \in \Phi$, where $f_0(\acute{\mathbf{w}}) = \acute{\mathbf{0}}$ is the *null calibrator* and $f_1(\acute{\mathbf{w}}) = \acute{\mathbf{w}}$ is the *identity calibrator*. Including the null calibrator guarantees that $L_{\min} \leq L_{\text{def}}$. Including the identity calibrator allows the possibility that the original recognizer is already optimally calibrated.

Since $L_{\min} \leq L_{\text{def}}$, we shall denote a recognizer that has $L_{\text{tot}} > L_{\text{def}}$ as *badly calibrated* at $(C_{\boldsymbol{\tau}}, \boldsymbol{\pi})$. This definition of bad calibration holds independently of the definition of the set Φ, as long as f_0 is in that set. This agrees with our previous definition of bad calibration of section 2.4.1, where that definition held independently of the probability model.

5.4 Summary

In this chapter, we formed our evaluation criterion from the posterior Bayes risk $\bar{C}(\mathcal{R}|\mathcal{E})$, by constructing a special observer, $\mathcal{E}$, known as the *evaluator*. The evaluator measures the performance of the recognizer on a given, supervised evaluation database. This measurement is not intended, in the first place, as an estimate of what recognizer performance might be on other unseen data, rather it is just a measurement of the performance on the given evaluation data.

We argued against the calibration-refinement decomposition of the cross-entropy interpretation of the evaluation criterion, where we would have needed explicit probabilistic modelling of the recognizer outputs by the evaluator. Instead, we proposed a solution for quantifying calibration loss as a minimization over suitably constrained calibration transformations.

Chapter 6

The Application

In this work we are interested in how to evaluate the goodness of pattern recognizers that are designed to be applied to a range of different *applications*. We model real-world applications mathematically via two parameters: *prior* $\boldsymbol{\pi}$ and cost $C_{\boldsymbol{\tau}}$. We refer to $(C_{\boldsymbol{\tau}}, \boldsymbol{\pi})$ or just $(\boldsymbol{\tau}, \boldsymbol{\pi})$ as *the application.*

In this chapter, we are interested in how the application interacts with the evaluation recipe. How does the evaluation criterion behave as a function of the application and how can we vary the application? We have already laid most of the groundwork in chapter 3, where we analysed cost function variation in terms of equivalence and cost function combination. Here we generalize our analysis to equivalence and combination of applications. We start our analysis by stating the general application-dependent evaluation criterion.

6.1 The general evaluation criterion

For convenience, we repeat the general evaluation criterion of (5.4):

$$\mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C^*_{\boldsymbol{\tau}}\big(\mathcal{B}(\mathcal{W}(x_t), \boldsymbol{\pi})\big|\theta_i\big) \,. \tag{6.1}$$

The recognizer under evaluation is $\mathcal{W}$, which is a function that maps an input x_t to a log-likelihood vector $\acute{\mathbf{w}}_t = \mathcal{W}(x_t)$. The recognizer's posterior is $\mathcal{B}(\acute{\mathbf{w}}_t, \boldsymbol{\pi})$, where $\mathcal{B}$ is Bayes' rule and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_N)$ is the prior. The evaluation criterion is the expected cost of using the recognizer's posterior to make Bayes decisions, when the cost of the outcomes of those decisions is computed via some cost function $C_{\boldsymbol{\tau}}$. The subscript $\boldsymbol{\tau}$ identifies and/or parametrizes the cost function. $C^*_{\boldsymbol{\tau}}$ is the proper scoring rule derived from the cost function, $C_{\boldsymbol{\tau}}$. The supervised evaluation database consists of the inputs, x_t, where trial t is in class i if $t \in \mathcal{T}_i$. Note that the prior, π_i, for class i can be chosen independently of the number of trials, $|\mathcal{T}_i|$, in that class.

The important point for this chapter is that the general evaluation criterion is parametrized by the application, $(C_{\boldsymbol{\tau}}, \boldsymbol{\pi})$.

6.2 Application combination

In chapter 3 we examined cost function combinations. Here we generalize to application combinations, where the prior is also varied over the combination. Let there be K different applications of the form $(C_k, \boldsymbol{\pi}_k)$, indexed by $k = 1, 2, \dots, K$. Here we evaluate every trial t, by using the recognizer output, $\acute{\mathbf{w}}_t = \mathcal{W}(x_t)$, to make a *different* Bayes decision for *each* of the K applications. The combined evaluation criterion is a weighted sum over the application-dependent criteria, $\mathcal{E}_k(\mathcal{W}|\boldsymbol{\pi}_k)$, with weights $\rho_k \geq 0$:

$$\begin{aligned}
&\sum_{k=1}^{K} \rho_k \mathcal{E}_k(\mathcal{W}|\boldsymbol{\pi}_k) \\
&= \sum_{k=1}^{K} \rho_k \sum_{i=1}^{N} \frac{\pi_{ik}}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C_k^*\big(\mathcal{B}(\acute{\mathbf{w}}_t, \boldsymbol{\pi}_k)\big|\theta_i\big) \qquad (6.2)\\
&= \sum_{k=1}^{K} \rho_k \sum_{i=1}^{N} \frac{\pi_{ik}}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C_k\big(\operatorname*{argmin}_a{}^* \sum_{j=1}^{N} \exp(w_{jt})\pi_{jk} C_k(a|\theta_j)\big|\theta_i\big)
\end{aligned}$$

where $(w_{1t}, \dots, w_{Nt}) \in \acute{\mathbf{w}}$ is any point on the log-likelihood line $\acute{\mathbf{w}}$. Now define a new application-dependent cost function, $\tilde{C}_k$, such that

$$\pi_{ik} C_k(a|\theta_i) = \tilde{\pi}_i \tilde{C}_k(a|\theta_i) \qquad (6.3)$$

for some convenient and constant $\tilde{\boldsymbol{\pi}} = (\tilde{\pi}_1, \dots, \tilde{\pi}_N)$, with $\tilde{\pi}_i > 0$. This gives:

$$\begin{aligned}
&\sum_{k=1}^{K} \rho_k \mathcal{E}_k(\mathcal{W}|\boldsymbol{\pi}_k) \\
&= \sum_{i=1}^{N} \frac{\tilde{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \sum_{k=1}^{K} \rho_k \tilde{C}_k\big(\operatorname*{argmin}_a{}^* \sum_{j=1}^{N} \exp(w_{jt}) \tilde{\pi}_j \tilde{C}_k(a|\theta_j)\big|\theta_i\big) \qquad (6.4)\\
&= \sum_{i=1}^{N} \frac{\tilde{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \sum_{k=1}^{K} \rho_k \tilde{C}_k^*\big(\mathcal{B}(\acute{\mathbf{w}}_t, \tilde{\boldsymbol{\pi}})\big|\theta_i\big).
\end{aligned}$$

Finally, by defining

$$C_\rho^*(\mathbf{q}|\theta) = \sum_{k=1}^{K} \rho_k \tilde{C}_k^*(\mathbf{q}|\theta), \qquad (6.5)$$

or equivalently, recalling section 3.5.2

$$C_\rho\big((a_1, \dots, a_K)\big|\theta\big) = \sum_{k=1}^{K} \rho_k \tilde{C}_k(a_k|\theta) \qquad (6.6)$$

we find:

$$\sum_{k=1}^{K} \rho_k \mathcal{E}_k(\mathcal{W}|\boldsymbol{\pi}_k) = \mathcal{E}_\rho(\mathcal{W}|\tilde{\boldsymbol{\pi}})\,. \tag{6.7}$$

This shows that the generalized evaluation recipe of (6.2), which exercises the recognizer on each of multiple applications, with variable priors and costs, is in fact *not* a generalization. It is numerically equivalent to an evaluation parametrized by a suitably defined *single* application, $(C_\rho, \tilde{\boldsymbol{\pi}})$.

The same holds for continuous combination of applications. A similar derivation shows that an integral of the form $\int_{\mathcal{K}} \rho(\boldsymbol{\tau})\mathcal{E}_{\boldsymbol{\tau}}(\mathcal{W}|\boldsymbol{\pi}_{\boldsymbol{\tau}})\,\mathbf{d}\boldsymbol{\tau}$ can also be written as an evaluation parametrized by a single application.

Finally note that if ρ_k or $\rho(\boldsymbol{\tau})$ is a normalized probability distribution, then the combination can also be interpreted as an expectation over a mixture of applications, distributed according to ρ.

6.3 Application equivalence

The above derivation further shows that (6.3) defines an equivalence relation between applications. The applications $(C, \boldsymbol{\pi})$ and $(\tilde{C}, \tilde{\boldsymbol{\pi}})$ are equivalent for evaluation purposes, if:

$$\pi_i C(a|\theta_i) = \tilde{\pi}_i \tilde{C}(a|\theta_i) \tag{6.8}$$

and therefore also

$$\pi_i C^*\big(\mathcal{B}(\acute{\mathbf{w}}, \boldsymbol{\pi})\big|\theta_i\big) = \tilde{\pi}_i \tilde{C}^*\big(\mathcal{B}(\acute{\mathbf{w}}, \tilde{\boldsymbol{\pi}})\big|\theta_i\big)\,. \tag{6.9}$$

6.3.1 Canonical form for applications

This equivalence allows us to standardize the prior, without loss of generality. The prior does not contribute any more degrees of freedom to the variability of application-dependent evaluation. As far as evaluation is concerned, the full scope of application variation can be represented with a family of applications that has variable cost, but a fixed prior $\tilde{\boldsymbol{\pi}}$. A convenient choice is $\tilde{\boldsymbol{\pi}} = \bar{\boldsymbol{\pi}}$, where $\bar{\boldsymbol{\pi}}$ is the *uniform prior* with $\bar{\pi}_1 = \cdots = \bar{\pi}_N = \frac{1}{N}$. We can now define an application to be in canonical form, if the cost function is in canonical form and the prior is uniform.

6.4 Examples: Interpretations of logarithmic cost

Here we show two examples of continuous application combinations. In the first, we fix the prior and integrate over a parametrized cost function. In the second, we fix the cost function and integrate over a parametrized prior. Both simplify to the evaluation criterion parametrized by $(C_{\text{log}}, \bar{\boldsymbol{\pi}})$.

6.4.1 Example 1: Fixed prior, variable cost

Recall the non-strict, proper scoring rule, $C^*_\eta(q|\theta)$, as defined in (3.12), which is derived from the two-class, normalized error-weighted cost. Let the prior be constant and uniform: $\boldsymbol{\pi} = \bar{\boldsymbol{\pi}} = (\frac{1}{2}, \frac{1}{2})$. The result of integrating over the cost function parameter with constant weight, $\rho(\eta) = 1$, is given by (3.29), so that the evaluation criterion for this application combination is:

$$\begin{aligned}
&\int_0^1 \mathcal{E}_\eta(\mathcal{W}|\bar{\boldsymbol{\pi}})\, d\eta \\
&= \sum_{i=1}^{2} \frac{\bar{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \int_0^1 C^*_\eta\big(\mathcal{B}(\acute{\mathbf{w}}_t, \bar{\boldsymbol{\pi}})\big|\theta_i\big)\, d\eta \\
&= \sum_{i=1}^{2} \frac{\bar{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C_{\text{log}}\big(\mathcal{B}(\acute{\mathbf{w}}_t, \bar{\boldsymbol{\pi}})\big|\theta_i\big) \\
&= \mathcal{E}_{\text{log}}(\mathcal{W}|\bar{\boldsymbol{\pi}})
\end{aligned} \tag{6.10}$$

where $C_{\text{log}}(\mathbf{q}|\theta_i) = -\log(q_i)$. Since $\int_0^1 \rho(\eta)\, d\eta = 1$, we can interpret $\rho(\eta)$ as a probability distribution over cost functions, or over applications. Now $\mathcal{E}_{\text{log}}(\mathcal{W}|\bar{\boldsymbol{\pi}})$ can be interpreted as the expected cost of using a recognizer, $\mathcal{W}$, to make Bayes decisions in a *mixture of applications* distributed as $\rho(\eta)$. Recall that C_η weights the two types of errors as $C_{\text{miss}} = \frac{1}{\eta}$ and $C_{\text{fa}} = \frac{1}{1-\eta}$, so that this mixture includes cost functions that range from a weight ratio of $\lim_{\eta\to 0} \frac{C_{\text{miss}}}{C_{\text{fa}}} \to \infty$ down to $\lim_{\eta\to 1} \frac{C_{\text{miss}}}{C_{\text{fa}}} \to 0$.

6.4.2 Example 2: Fixed cost, variable prior

Here we use the fixed cost function, $C_{\text{err}} = \frac{1}{2}C_{0.5}$, where by $C_{0.5}$ we mean C_η at $\eta = 0.5$. We parametrize the prior as $\boldsymbol{\pi}_\tau = \big(\pi(\tau), \pi(-\tau)\big)$, where

$$\pi(\tau) = 1 - \pi(-\tau) = \text{logit}^{-1}(\tau) = \frac{1}{1 + \exp(-\tau)} \tag{6.11}$$

for $-\infty < \tau < \infty$. Notice that $\tau = \text{logit}(\pi) = \log \frac{\pi}{1-\pi}$ is the *prior log odds.* We choose constant weight, $\rho(\tau) = 1$ for the integration. Letting $\bar{\boldsymbol{\pi}} = (\frac{1}{2}, \frac{1}{2})$, the evaluation criterion for this combination is:

$$\begin{aligned}
&\int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\boldsymbol{\pi}_\tau)\, d\tau \\
&= \sum_{i=1}^{2} \frac{\bar{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \int_{-\infty}^{\infty} \frac{\pi_i(\tau)}{\bar{\pi}_i} C^*_{\text{err}}\big(\mathcal{B}(\acute{\mathbf{w}}_t, \boldsymbol{\pi}_\tau)\big|\theta_i\big)\, d\tau \\
&= \sum_{i=1}^{2} \frac{\bar{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \int_{-\infty}^{\infty} \pi_i(\tau) C^*_{0.5}\big(\mathcal{B}(\acute{\mathbf{w}}_t, \boldsymbol{\pi}_\tau)\big|\theta_i\big)\, d\tau\,.
\end{aligned} \tag{6.12}$$

Now do a change of variables, with $\eta = 1 - \pi(\tau) = \text{logit}^{-1}(-\tau)$ and $d\eta = -\eta(1-\eta)d\tau$ and use the fact that

$$C^*_{0.5}\Big(\mathcal{B}\big(\acute{\mathbf{w}}_t, (1-\eta, \eta)\big)\Big|\theta_i\Big) = C^*_\eta\big(\mathcal{B}(\acute{\mathbf{w}}_t, \bar{\boldsymbol{\pi}})\big|\theta_i\big) \tag{6.13}$$

to find

$$\begin{aligned}
&\int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\boldsymbol{\pi}_\tau)\, d\tau \\
&= \sum_{i=1}^{2} \frac{\bar{\pi}_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \int_{0}^{1} C^*_\eta\big(\mathcal{B}(\acute{\mathbf{w}}_t, \bar{\boldsymbol{\pi}})\big|\theta_i\big)\, d\tau \\
&= \int_0^1 \mathcal{E}_\eta(\mathcal{W}|\bar{\boldsymbol{\pi}})\, d\eta \\
&= \mathcal{E}_{\log}(\mathcal{W}|\bar{\boldsymbol{\pi}})
\end{aligned} \tag{6.14}$$

where we recalled (6.10) for the final equality. Now we have $\int_{-\infty}^{\infty} \rho(\tau)\, d\tau \to \infty$, so that $\rho(\tau)$ cannot be normalized to a proper probability distribution. Therefore, $\int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\boldsymbol{\pi}_\tau)\, d\tau$ as an interpretation of evaluation by logarithmic cost is *not* an expectation. We can interpret it as *total expected error-rate* of the combination over the specified range of priors, because $\mathcal{E}_{\text{err}}(\mathcal{W}|\boldsymbol{\pi}_\tau)$ is the expected error-rate at $\boldsymbol{\pi}_\tau$.

6.5 Summary

The general evaluation criterion is parametrized by some specified *application*. Evaluating over combinations or mixtures of applications does not generalize this recipe. However, these mechanisms do provide insightful interpretations of strictly proper scoring rules like logarithmic cost in terms of non-strict proper scoring rules, which are more directly related to simple decision problems.

Our tools for evaluation are now in place. In the next chapter, we apply them to find specific evaluation solutions for two-class problems such as speaker recognition. Thereafter we turn to multi-class problems such as language recognition.

Chapter 7

Solutions for two-class recognition

In this chapter we assemble concrete evaluation solutions for two-class pattern recognition, for which we choose the canonical *speaker detection* problem as representative.[1]

We are interested in two-class pattern recognition problems where there may be variable cost functions and priors for different applications and where it makes sense for the pattern recognizer to give its output in cost-and-prior-independent log-likelihood form, rather than in prior-dependent posterior probability form, or prior-and-cost-dependent hard decision form.

We consider two-class evaluation separately from multi-class, because we present analysis and solutions that work only for two-classes. The multi-class case will be treated in chapter 8.

In this rather large chapter, we will make use of almost all of the material discussed up to this point, first to analyse traditional speaker detection evaluation methods and then to generalize them to new methods with new and complementary capabilities.

7.1 Speaker detection

In this chapter, the recognizer of interest is a speaker detector. The input, x, to the speaker detector consists of a pair of speech segments. We assume that both segments do contain speech and that each segment has speech of only one speaker. For each such input x, exactly one of the following two propositions may be true: θ_1: the segments in x are of the same speaker; or θ_2: the segments are of two different speakers. It is traditional to denote θ_1 as the *target* class and θ_2 as the *non-target* class.

[1]For a unified view of a large class of very general speaker recognition problems, see our paper [20], but here we consider only the speaker detection problem.

As explained in section 4.2.1, the output of the speaker detector is in the form of the one dimensional log-likelihood-ratio, where the numerator of this ratio is understood to be the likelihood for θ_1 and the denominator the likelihood for θ_2. For a well-calibrated log-likelihood-ratio, positive values favour the target class, while negative values favour the non-target class. The magnitude is an indication of the confidence, or the degree of support for the favoured class. We use the symbol w for log-likelihood-ratio. It is related to our more general log-likelihood notation, $\acute{\mathbf{w}}$, as: $(w, 0) \in \acute{\mathbf{w}}$.

By *well-calibrated* we do not mean perfectly calibrated as we have defined it, we just mean that calibration is reasonably good, so that w makes good Bayes decisions. Calibration is a joint property of *all* of the scores of a given detector on a given evaluation database. Calibration is *not* a property of a score for a single trial: for a single target trial the best score is $+\infty$ and for a single non-target trial the best score is $-\infty$ and there is nothing more to say. Calibration asks whether we can transform *all* the scores of the database with a single transformation to get better Bayes decisions. Calibration of a given detector can vary from database to database—it could be good for one and bad for another.

If the output of a speaker detector cannot be assumed to be well-calibrated, it is called the *score*, rather than the log-likelihood-ratio. The score can be interpreted in the sense that larger (more positive) scores favour the target class and smaller (more negative) scores favour the non-target class. However, the zero score has no special reference value. A score may be subjected to a calibration transformation, the output of which is intended to act as a well-calibrated log-likelihood-ratio. The log-likelihood-ratio also qualifies as a score, but not vice-versa.

7.2 Idealized scores

Here we analyse idealized detection scores, the properties of which will inspire some of the choices we have to make below when choosing practical evaluation strategies.

7.2.1 Assumptions

Let $f(s)$ and $g(s)$ be probability densities, with support $\mathbb{R}$, so that they are non-zero for every $s \in \mathbb{R}$, but $f(\infty) = f(-\infty) = g(\infty) = g(-\infty) = 0$. We also assume both are differentiable for any $s \in \mathbb{R}$. We interpret these functions as the *score likelihoods*, $f(s) = P(s|\text{target}, \mathcal{O})$ and $g(s) = P(s|\text{non-target}, \mathcal{O})$. As before, $\mathcal{O}$ plays the role of conditioning these probability distributions. We further assume that f and g are related such that the *likelihood-ratio*,

$$\ell(s) = \frac{f(s)}{g(s)} \tag{7.1}$$

is a *strictly monotonic rising bijection*[2] from $[-\infty, \infty]$ to $[0, \infty]$, with $\ell(-\infty) = 0$ and $\ell(\infty) = \infty$, so that the inverse function $\ell^{-1}(y)$ is defined for every $y \in [0, \infty]$ and the derivative, $\ell'(s) > 0$, for every $s \in \mathbb{R}$.

In the notation of section 4.3, $w = \log \ell(s)$ is a *calibration transformation* between score and log-likelihood-ratio. The above assumptions are equivalent to requiring that there exists a strictly monotonic rising, continuous, differentiable bijection between scores and log-likelihood-ratios.

We give three examples that satisfy these criteria:

Example 1: Gaussian scores

For normal distributions, $f(s) = \mathcal{N}(s|\mu_1, \sigma^2)$ and $g(s) = \mathcal{N}(s|\mu_2, \sigma^2)$, where $\mu_1 > \mu_2$, we find: $\ell(s) = \exp(\frac{\mu_1-\mu_2}{\sigma^2}s + \frac{\mu_2^2-\mu_1^2}{2\sigma^2})$. Note this doesn't work when the variances are different, because then $\ell(s)$ is not monotonic.

Example 2: Calibrated scores

If the score s is a perfectly calibrated log-likelihood-ratio, so that $s = \log \frac{f(s)}{g(s)} = \log \ell(s)$, then $\ell(s) = \exp(s)$.

Example 3: Calibratable scores

If $s = \alpha + \beta \log \ell(s)$, for $\beta > 0$, then $\ell(s) = \exp \frac{s-\alpha}{\beta}$. This also works for more general bijections between s and $\log \ell(s)$.

7.2.2 Properties

Under the above assumptions, the following properties hold:

Scores can make optimal decisions

We already know thresholding the likelihood-ratio, $\ell(s)$, can be used to make optimal detection decisions. However, thresholding the score can do so too, because for any likelihood-ratio threshold, $y \in [0, \infty]$, the score threshold, $\gamma = \ell^{-1}(y)$, makes equivalent decisions, since: $s \geq \gamma$ if and only if $\ell(s) \geq y$.

For some score threshold, $\gamma \in [-\infty, \infty]$, define the *miss rate*, $F(\gamma)$, and *false-alarm rate*, $G(\gamma)$, as:

$$F(\gamma) = P(s < \gamma|\text{target}) = \int_{-\infty}^{\gamma} f(s)\, ds \tag{7.2}$$

$$G(\gamma) = P(s >= \gamma|\text{non-target}) = \int_{\gamma}^{\infty} g(s)\, ds \tag{7.3}$$

[2]See appendix A, which explains the difference between invertible and bijective.

with derivatives $F'(\gamma) = f(\gamma)$ and $G'(\gamma) = -g(\gamma)$. This shows F and G are continuous, strictly monotonically increasing and decreasing respectively, and both invertible.

For a given prior, $(\pi, 1-\pi)$, let the *expected error-rate* be:

$$E(\pi,\gamma) = \pi F(\gamma) + (1-\pi)G(\gamma)\,. \tag{7.4}$$

For any $0 < \pi < 1$, we can use first and second derivatives w.r.t. γ to verify that E has a unique minimum at

$$\gamma^* = \ell^{-1}\big(\frac{1-\pi}{\pi}\big) \tag{7.5}$$

where the first derivative is zero and the second derivative is $\pi\ell'(\gamma^*)g(\gamma^*) > 0$. At the boundaries, $\pi = 0$ or $\pi = 1$, we have the minima respectively at $E(0,\infty) = E(1,-\infty) = 0$. Here $\frac{1-\pi}{\pi}$ is the likelihood-ratio threshold and γ^* is the score threshold and both give the same minimum-expected-error-rate Bayes decisions.

We use the notation E^*, for the minimum:

$$E^*(\pi) = E(\pi,\gamma^*) = \min_\gamma E(\pi,\gamma) \tag{7.6}$$

As mentioned above, $E^*(0) = E^*(1) = 0$. It is easy to show[3] that $E^*(\pi)$ is *concave* and therefore has a unique maximum. $E^*(\pi)$ can be interpreted as the error-rate of the detector at the prior, π, provided the optimal score threshold is used. For well-calibrated detectors, the error-rate vanishes when there is no prior uncertainty, but is non-zero in between. (Our experiments on real speaker detection scores show that the maximum invariably occurs somewhere near $\pi = 0.5$.)

ROC properties

If we sweep the score threshold, γ, from $-\infty$ to ∞ (with F increasing and G decreasing) and plot G against F, this gives[4] what is known as the *receiver operating characteric*, or ROC [62]. It gives a trade-off between false-alarm-rate and miss-rate as the threshold is varied. This curve is a strictly decreasing and *strictly convex* function, which maps false-alarm-rate, in $[0,1]$, to miss-rate, in $[0,1]$:

$$P_{\text{miss}} = F\left(G^{-1}(P_{\text{fa}})\right)\,. \tag{7.7}$$

The curve end-points are at $(P_{\text{fa}}, P_{\text{miss}}) = (0,1)$ and $(1,0)$. The slope of this function, at a given threshold γ is $-\ell(\gamma)$. The convexity follows from the strictly increasing property of ℓ.

[3]This can be done as in section 2.1.6, or by noting that together G and F satisfy the contract of a cost function, with $\mathcal{A} \in [-\infty,\infty]$, so that E^* has the form of a generalized entropy function.

[4]ROC is often plotted as G vs $1-F$, but for convenience we use G vs F here to agree with the DET-curve in speaker recognition.

EER interpretation

The *equal-error-rate*, or EER, is a special point on the ROC, which acts as a scalar summary of the whole curve. It can be defined as the point on the ROC for which $(P_{\text{fa}}, P_{\text{miss}}) = (\text{EER}, \text{EER})$. Below we show how it acts as a summary for the whole curve [6].

Under the assumptions above, the expected error-rate, $E(\pi, \gamma)$, satisfies the conditions for Sion's *minimax theorem* [63], which allows interchanging of nested minimum and maximum over the two variables. In particular, E is quasi-convex in γ, since it has a unique minimum. E is linear in π and therefore quasi-concave in π. We can now interpret the EER as the maximum optimal error-rate:

$$\begin{aligned}
\max_\pi E^*(\pi) &= \max_\pi \min_\gamma E(\pi, \gamma) \\
&= \min_\gamma \max_\pi E(\pi, \gamma) \\
&= \min_\gamma \max\bigl(E(0, \gamma), E(1, \gamma)\bigr)\,, \text{ by linearity in } \pi \\
&= \min_\gamma \max\bigl(G(\gamma), F(\gamma)\bigr) \\
&= G(\gamma_{\text{eer}}) = F(\gamma_{\text{eer}}) \\
&= \text{EER}
\end{aligned} \tag{7.8}$$

where the last step can be understood by noting that F increases from 0 to 1, while G decreases from 1 to 0, forming a rough X, of which the top v is the inner discrete maximum, which in turn has its minimum w.r.t. γ at the cusp—where the two functions are equal.

The *max min* form of the first line of (7.8) shows that the EER summarizes the curve as a tight *upper bound on error-rate*: Provided you use the optimal threshold (found by the inner minimization), the error-rate at any prior (scanned by the outer maximization) will not exceed the EER.[5]

If a detector is optimized by using EER as an objective to be minimized, error-rates at all operating points will be driven down. E^* is concave, so that by 'pushing down' on its maximum at EER, you cannot make a local dent that violates concavity, instead (if anything budges) the whole curve has to go down.

[5]Although the *min max* form seems to show that maximum is always found at $\pi = 0$, or $\pi = 1$, E actually becomes locally independent of π at the EER saddle-point.

Generalization of EER

In fact, we can interpret *any* point on the ROC via a generalization of (7.8). Letting $\alpha, \beta > 0$, a similar derivation shows:

$$\begin{aligned} C_{\alpha,\beta} &= \max_{\pi} \min_{\gamma} \pi\alpha F(\gamma) + (1-\pi)\beta G(\gamma) \\ &= \alpha F(\gamma_{\alpha,\beta}) = \beta G(\gamma_{\alpha,\beta}) \end{aligned} \tag{7.9}$$

so that $\left(\frac{C_{\alpha,\beta}}{\beta}, \frac{C_{\alpha,\beta}}{\alpha}\right)$ is a point on the ROC. By varying the error-rate ratio, $\frac{\alpha}{\beta}$, from 0 to ∞, we can plot out the whole ROC. In this sense, any point on the ROC can be seen as an upper bound on the weighted cost of errors, provided the optimal threshold is used.

AUC

The *area under* the ROC, often called AUC, is the probability that a randomly picked non-target score will exceed a randomly picked target score [62]. This can be seen by integrating the joint probability, $f(\tilde{s})g(s)$ of a target score $\tilde{s}$ and a non-target score s over the region where $s > \tilde{s}$:

$$\begin{aligned} P(s > \tilde{s}) &= \int_{-\infty}^{\infty} g(s) \int_{-\infty}^{s} f(\tilde{s})\, d\tilde{s}\, ds \\ &= \int_{-\infty}^{\infty} g(s) F(s)\, ds \\ &= \int_{0}^{1} F\left(G^{-1}(p)\right)\, dp \\ &= \text{AUC} \end{aligned} \tag{7.10}$$

where we used the change of variables $p = G(s)$, $dp = -g(s)\, ds$. Although this is a useful figure of merit for applications where scores are sorted to prioritize targets, rather than comparing each score to a fixed threshold, AUC has apparently not had much exposure in the speaker recognition literature.

For practical computation of AUC, we propose using the ROCCH variant of the empirical ROC, which we shall discuss in section 7.3.6 below. But so far, we have not used this metric in practice.

7.3 Analysis and generalization of DCF and ROC

The speaker recognition literature is large and spans several decades. The general pattern recognition literature is even larger. Many evaluation solutions have been used and we will not attempt a review here. Instead, in this section we analyse the evaluation solutions that have been the main tools in the speaker recognition community that has been participating in the regular NIST Speaker Recognition Evaluations over the last decade. We consider these tools to be representative of the state-of-the-art.

We discuss three tools: DCF, DET-curves and EER. The first two have been used by the evaluator, NIST, to evaluate the performance of speaker detectors. EER, although not used by NIST, is reported in almost every publication on speaker recognition. The solutions proposed in this work were inspired by DET, DCF and EER, and they can be viewed as a generalization and a synthesis of some of their properties, and as complementary to them.

We discuss these three tools in turn, comparing them to our proposed solution.

7.3.1 DCF vs Bayes error-rate

The DCF, or *detection cost function*,[6] has been used by NIST as their primary speaker detection evaluation criterion for more than a decade, from 1997 to the present.[7] In this section, we define and analyse DCF and compare it with our proposed evaluation recipe.

DCF can be specified concisely (as NIST does):

$$\text{DCF} = \pi C_{\text{miss}} P_{\text{miss}} + (1-\pi) C_{\text{fa}} P_{\text{fa}} \tag{7.11}$$

or laboriously as we do below to relate to our analysis. (For the uninitiated, the symbols in (7.11) will be defined in due course.)

In our terminology, DCF is an evaluation criterion parametrized by a specified *application*. Evaluation by DCF requires the detector to make hard binary decisions. The decision set is $\mathcal{A} = \{\text{accept}, \text{reject}\}$, where *accept* means the target hypothesis has been recognized and *reject* means the non-target hypothesis has been recognized. A *miss* is defined as the erroneous outcome $(\text{reject}, \text{target})$ and a *false-accept* as the erroneous outcome $(\text{accept}, \text{non-target})$. As explained in section 3.4.1, these respective errors are weighted with the specified parameters C_{miss} and C_{fa}. The remaining two correct outcomes have zero cost. Let us denote this cost function as C_{dcf}. Additionally, there is specified a prior parameter, in our notation: $\boldsymbol{\pi} = (\pi, 1-\pi)$, where $\pi = P(\text{target}|\boldsymbol{\pi})$.

The triplet $(C_{\text{miss}}, C_{\text{fa}}, \pi)$, parametrizes a family of applications of the form (C_{dcf}, π). As explained in section 3.4 on the equivalence of cost functions and section 6.3 on the equivalence of applications, a three-parameter specification is redundant—for evaluation we need only a *single* parameter. We give two

[6] This terminology may be confusing. Our cost function, $C_{\boldsymbol{\tau}}$, evaluates a single detection trial. DCF is the expected cost over all the trials in an evaluation.

[7] See `www.itl.nist.gov/iad/mig//tests/sre/`, where the "Speaker Recognition Evaluation Plans" from 1997 to 2010 are available.

such single-parameter application families that are equivalent to the three-parameter family, $(C_{\text{dcf}}, \boldsymbol{\pi})$:

$$(C_{\text{dcf}}, \pi) \equiv (C_{\text{err}}, \tilde{\pi}) \equiv (C_\eta, 0.5) \tag{7.12}$$

where

$$\tilde{\pi} = \text{logit}^{-1}\left(\text{logit}(\pi) + \log\frac{C_{\text{miss}}}{C_{\text{fa}}}\right) \tag{7.13}$$

$$\eta = 1 - \tilde{\pi} \tag{7.14}$$

where[8] C_{err} is zero-one cost as defined in section 2.1.3 and C_η is the normalized cost defined in section 3.4.1. A good name for $\tilde{\pi}$ is the *effective prior*. The Bayes decision threshold for all three families is η.

As previously explained, these three families are *equivalent* for evaluation purposes because they would rank recognizers in the same order. All three share the same Bayes decision threshold, all three would give the same Bayes decision for a given posterior and all three would give the same miss and false-alarm rates for a given recognizer. This equivalence can be expressed in terms of our evaluation criterion as $\mathcal{E}_{\text{dcf}}(\mathcal{W}|\pi) = k\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi}) = k'\mathcal{E}_\eta(\mathcal{W}|0.5)$, where k and k' are unimportant positive scale factors.

Our evaluation criterion, $\mathcal{E}_{\text{dcf}}$ is a *generalization* of DCF in the following sense. For evaluation by DCF, the detector is required to submit a hard accept/reject decision for every trial, while for $\mathcal{E}_{\text{dcf}}$, the detector is required to submit a log-likelihood-ratio for every trial. DCF evaluates the actual decisions made by a detector, while $\mathcal{E}_{\text{dcf}}$ measures the ability of the detector to make Bayes decisions. In the DCF recipe, the evaluee (the recognizer) makes the decisions. In our recipe, the evaluator makes the decisions. DCF evaluates by cost function, while $\mathcal{E}_{\text{dcf}}$ evaluates by proper scoring rule. In the DCF recipe, for a given submitted detector, there are fixed miss and false-alarm rates:

$$\text{DCF} = \sum_{i=1}^{2} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C_{\text{dcf}}(a_t|\theta_i) = \pi C_{\text{miss}} P_{\text{miss}} + (1-\pi) C_{\text{fa}} P_{\text{fa}} \tag{7.15}$$

where

$$P_{\text{miss}} = \frac{1}{|\mathcal{T}_1|} \sum_{t\in\mathcal{T}_1} C_{\text{err}}(a_t|\text{target}) \tag{7.16}$$

$$P_{\text{fa}} = \frac{1}{|\mathcal{T}_2|} \sum_{t\in\mathcal{T}_2} C_{\text{err}}(a_t|\text{non-target}) \tag{7.17}$$

[8]Recall $\text{logit}(p) = \log\frac{p}{1-p}$ and $\text{logit}^{-1}(x) = \frac{1}{1+\exp(-x)}$.

and where $a_t \in \{\text{accept}, \text{reject}\}$ is the detector output for trial t. In our recipe, for a given detector, the error-rates vary as a function of the application parameter, as shown by the equivalent parametrization $(C_{\text{err}}, \tilde{\pi})$:

$$\begin{aligned} \mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi}) &= \sum_{i=1}^{2} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C^*_{\text{err}}\big(\mathcal{B}(w_t,\tilde{\pi})\big|\theta_i\big) \\ &= \tilde{\pi} P_{\text{miss}}(\tilde{\pi}) + (1-\tilde{\pi}) P_{\text{fa}}(\tilde{\pi}) \end{aligned} \tag{7.18}$$

where

$$\begin{aligned} P_{\text{miss}}(\tilde{\pi}) &= \sum_{t\in\mathcal{T}_1} C^*_{\text{err}}\big(\mathcal{B}(w_t,\tilde{\pi})\big|\text{target}\big) \\ P_{\text{fa}}(\tilde{\pi}) &= \sum_{t\in\mathcal{T}_2} C^*_{\text{err}}\big(\mathcal{B}(w_t,\tilde{\pi})\big|\text{non-target}\big) \end{aligned} \tag{7.19}$$

and where w_t is the detector's submitted log-likelihood-ratio for trial t.

DCF does not allow the application to be varied. The prior and cost function are effectively hard-coded into the detector. Up to 2008, NIST had always parametrized the DCF at $(C_{\text{miss}}, C_{\text{fa}}, \pi) = (10, 1, 0.01)$, or equivalently at an effective prior of $\tilde{\pi} \approx 0.091$. In 2010 they specified $C_{\text{miss}} = C_{\text{fa}} = 1$ and $\tilde{\pi} = 0.001$. Now $\tilde{\pi} = 0.091$ is referred to as the *old operating point* and $\tilde{\pi} = 0.001$ is referred to as the *new operating point.* The limitation of DCF evaluation is that only a single operating point is evaluated in a given evaluation.

Our innovation with $\mathcal{E}_{\text{dcf}}$ is that the application can be varied for a given detector, in order to evaluate it over a *range* of different operating points. By using the equivalence $\mathcal{E}_{\text{dcf}}(\mathcal{W}|\pi) = k\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi}) = k'\mathcal{E}_{\eta}(\mathcal{W}|0.5)$, we can conveniently parametrize the range of applications in terms of $\tilde{\pi}$, or in terms of η. We demonstrate with an example below, where we parametrize with $\tilde{\pi}$. (The η-parametrization will come in handy later.)

In what follows, we shall refer to $\mathcal{E}_{\text{err}}$ as *Bayes error-rate*, since it is the error-rate obtained when making Bayes decisions.

Example: Normalized Bayes error-rate plot

In this example we do an evaluation of a single speaker detector, $\mathcal{W}$. We use a fixed evaluation database of about a hundred thousand trials, but the *Bayes error-rate* evaluation criterion, $\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})$, is varied by adjusting $\tilde{\pi}$. That is, we are evaluating the recognizer over a range of different applications, where the effective prior is varied, but the cost function, C_{err}, remains fixed.

To illustrate our point, we use a typical uncalibrated speaker detector. Its output scores are evaluated as is, as if they were calibrated log-likelihood ratios.

The result of the evaluation is given in figure 7.1, where the horizontal axis represents the prior on a *log odds* scale and the vertical axis is a normalized version of $\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})$.

The horizontal axis of the plot represents the prior and at the same time the Bayes decision threshold. The horizontal axis is the prior log odds: $h = \text{logit}(\tilde{\pi}) = \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$. The logit is a monotonic rising invertible transformation that sends the interval $[0, 1]$ to the extended real line, $[-\infty, \infty]$. The h-axis is infinite, so we have to be content with plotting a limited interval, which we centre at $\text{logit}(0.5) = 0$.

The Bayes decision threshold is just $\lambda = -h$, so that scores $w_t \geq -h$ give accept decisions, while those below give reject decisions.

The vertical axis, v, of the plot is our evaluation criterion *normalized* thus:

$$v = \frac{\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})}{\mathcal{E}_{\text{err}}(\mathcal{W}_0|\tilde{\pi})} = \frac{\tilde{\pi} P_{\text{miss}}(\tilde{\pi}) + (1-\tilde{\pi}) P_{\text{fa}}(\tilde{\pi})}{\min(\tilde{\pi}, 1-\tilde{\pi})} \tag{7.20}$$

where $\mathcal{W}_0$ is the default recognizer that always outputs $\mathcal{W}_0(x_t) = 0$. The denominator changes abruptly at $\tilde{\pi} = 0.5$, hence the cusp at $h = 0$. The numerator is the *Bayes error-rate*, where everything varies as functions of h: $\tilde{\pi} = \text{logit}^{-1}(h)$, $P_{\text{miss}}(\tilde{\pi})$ is the proportion of target trials with scores below the threshold and $P_{\text{fa}}(\tilde{\pi})$ is the proportion of non-target trials with scores above the threshold, as given by (7.19).

The normalization places the *reference* value at 1 for default performance. This is indicated by the dashed red line. This plot shows the uncalibrated detector, $\mathcal{W}$, is useful for applications with prior log odds greater than about -2, while for applications with smaller prior, this recognizer is badly calibrated and would not be useful, since performance is worse than the default detector.

The important message here is that if we had evaluated the detector at just one specific application $(C_{\text{err}}, \tilde{\pi})$, as represented by just one point on the horizontal axis, then the measurement at that point would not necessarily give a good indication of usefulness at some other point (application) along the axis.

The above graphical evaluation procedure forms the essence of the author's current favourite tool for judging the calibration of speaker recognition scores. We made extensive use of this tool during the 2010 NIST Speaker Recognition Evaluation. Examples will be given in section 7.6 below.

This graphical solution is applicable only to two-class problems, because for multi-class problems, applications cannot be parametrized by a single parameter. This makes graphical representation difficult, especially when there are several classes. We defer discussion of multi-class evaluation solutions to chapter 8.

Moreover, the graphical solution does not provide an application-spanning, scalar summary criterion. In practice, such scalar summary criteria are invaluable tools for discriminative training of fusion and calibration parameters of pattern recognizers. We discuss summary criteria in section 7.4.

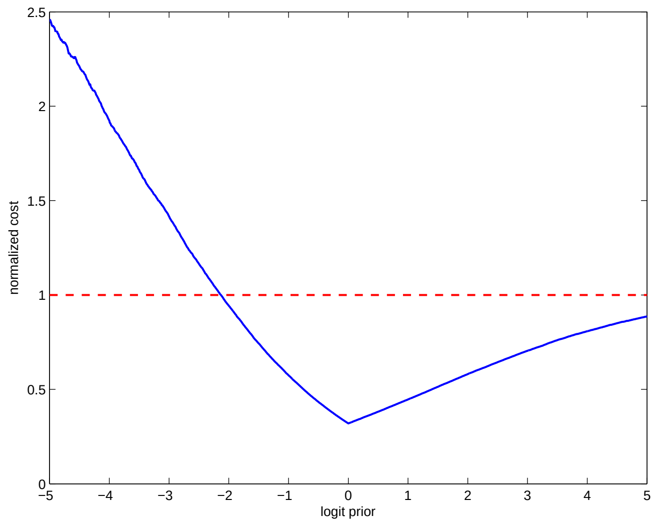

Figure 7.1: Normalized Bayes error-rate for a speaker detector, as a function of the prior.

7.3.2 Limitations of empirical evaluation

Before continuing, a word of caution is appropriate. The empirical evaluation methodology discussed in this work is data-driven and therefore subject to the limitations of the available evaluation data. For a speaker detector to be strictly useful when the effective target prior, $\tilde{\pi}$, is arbitrarily close to 0 or to 1—when the log-likelihood-ratio threshold is very large positive or negative—the recognizer will also have to be able to output log-likelihood-ratio scores of arbitrarily large magnitude. If there were no scores of large magnitude, the detector assisted decisions would never differ from the Bayes decision made with the prior alone. But for a well-calibrated recognizer, such large scores will be rare and will most likely not be observed at all if the evaluation database is too small. In this case, the decision-making ability of the scores at the large threshold will not be exercised by the evaluator and a meaningful evaluation at this operating point will not be achieved. In short, if the threshold magnitude is very large, the evaluation will just give the uninteresting result of either $(P_{\text{miss}}, P_{\text{fa}}) = (0, 1)$, or $(P_{\text{miss}}, P_{\text{fa}}) = (1, 0)$.

For a quantitative statement of this effect, based on a binomial error distribution model, see *Doddington's Rule of 30* [64], or see our interpretation in appendix B. In summary, the rule says you need at least 30 false-alarms and at least 30 misses for meaningful evaluation. In our development for SRE 2010, where the *new operating point* made false alarms rare, we found this rule to be a good practical rule of thumb for choosing the sizes of our development databases.

Apart from the size of the evaluation database, the accuracy of its supervision also becomes increasingly important as the threshold magnitude gets larger. In short, if error-rates are below the supervision label noise floor, we cannot measure those error-rates accurately.

Indeed, with the *new operating point* in the 2010 NIST Speaker Recognition Evaluation [65], both of these problems, data scarcity and label accuracy, presented significant challenges to both evaluator and participants.

In view of these limitations, our solutions for evaluation will either eliminate, or minimize, the effect of extreme applications that have Bayes decision thresholds very close to the boundaries of the posterior simplex. The first example is our graphical evaluation of figure 7.1, where we limited the range of evaluation to $|\operatorname{logit}\tilde{\pi}| \leq 5$.

7.3.3 minDCF

NIST uses an auxiliary evaluation criterion, called minDCF, that effectively does a calibration analysis, similar to our proposal discussed in section 5.3.1. In contrast with DCF, which evaluates hard decisions made by the detector, minDCF evaluates *scores* produced by the detector. Given scores, the evaluator—rather than the detector—now makes the decisions.

For evaluation by minDCF, the (implicit) contract between the evaluator and the evaluee is that the scores submitted by the evaluee conform to the idealized score assumptions of section 7.2.1. In the notation of that section: $f(s) = P(s|\text{target}, \mathcal{R})$ and $g(s) = P(s|\text{non-target}, \mathcal{R})$, where s is the score and $\mathcal{R}$ represents the probability model of the *evaluee*. As discussed in chapter 5, the evaluator uses a *different*, empirical probability model. Under this contract, the evaluator can now use score thresholds to make detection decisions that are equivalent to the decisions that could have been made by calibrated log-likelihood-ratios submitted by the evaluee.

Since the log-likelihood-ratios are not given and the calibration transformation is also not given, the evaluator optimizes over all possible strictly monotonic rising calibrations to find out what the DCF of the detector could have been if the best possible such transformation had been used. Since DCF is defined for a single operating point, the evaluator does not have to find the whole function, but only the single optimal score threshold. This proceeds as follows:

Let γ be a score threshold, so that $P_{\text{miss}}(\gamma)$ is the fraction of target trials with scores below the threshold and $P_{\text{fa}}(\gamma)$ the fraction of non-target trials with scores above the threshold. Then

$$\text{minDCF} = \min_{-\infty\leq\gamma\leq\infty} \pi C_{\text{miss}} P_{\text{miss}}(\gamma) + (1-\pi) C_{\text{fa}} P_{\text{fa}}(\gamma) \,. \tag{7.21}$$

Note that the evaluator does the minimization, while making use of the true class labels (without which $P_{\text{miss}}(\gamma)$ and $P_{\text{fa}}(\gamma)$ could not be computed).

Consider hard decisions evaluated by DCF and scores evaluated by minDCF: if the decisions are made by thresholding the scores, with a single fixed threshold, then indeed: minDCF $\leq$ DCF.

Under the evaluator's empirical probability model, the minDCF optimization is a discrete one. The evaluator does not need to consider all possible thresholds. If the scores are sorted, then one threshold at $-\infty$, one at ∞ and one in every interval between adjacent scores needs to be considered. In contrast to the result of section 7.2.2, which was based on idealized continuous probability densities, the discrete minimum is often *not* unique.

In practice, minDCF has two uses:

- It can be used in conjunction with DCF to judge calibration loss, as the difference, $L_{\text{cal}} = \text{DCF} - \text{minDCF}$. Since DCF is confined to a fixed operating point, so is this rendering of L_{cal}. It measures *point calibration* of hard decisions. However, by generalizing DCF to $\mathcal{E}_{\text{err}}$, we can overcome this limitation and sweep the prior parameter over a range of applications, measuring calibration everywhere in this range. This will be demonstrated in section 7.6.

- It can be used on its own, without DCF, as a calibration-insensitive evaluation criterion. It is commonplace for publications in speaker detection to quote just minDCF and EER, which is also calibration-insensitive.

In section 7.7, we will present a generalization of minDCF, where the evaluator indeed optimizes the whole calibration transformation function, so that it is applicable to a range of applications, not just a single operating point.

7.3.4 Empirical ROC

We have already introduced the idealized, continuous ROC in section 7.2.2. An analogous, empirical ROC can be obtained by score thresholding, exactly as done to compute minDCF above. Let $P_{\text{miss}}(\gamma)$ and $P_{\text{fa}}(\gamma)$ be defined as above, where γ is the score threshold. As a function of γ, $P_{\text{miss}}(\gamma)$ increases in discrete steps of size $\frac{1}{|\mathcal{T}_1|}$ and $P_{\text{fa}}(\gamma)$ decreases in steps of $\frac{1}{|\mathcal{T}_2|}$. Making the database large will make these functions smoother in the sense of making the steps small.

The result on the ROC plane is not a continuous curve, but a set of discrete $(P_{\text{fa}}, P_{\text{miss}})$ points. The points at $(1,0)$ and $(0,1)$ correspond to the two thresholds at infinity and each of the other points corresponds to a threshold in the interval between two adjacent scores. The points are separated by horizontal or vertical steps, where each horizontal step corresponds to a non-target score and each vertical step to a target score.

The ROC is seldom used as is in speaker recognition, but is usually transformed to the DET-plot as explained below.

7.3.5 DET-plot

The *detection error trade-off*, or DET-plot, is used by NIST and is ubiquitous in the speaker recognition literature [66]. The DET-plot evaluates speaker detection scores, rather than hard decisions. In this respect it agrees with minDCF. It also agrees with minDCF in the sense that the evaluator effectively performs a threshold optimization as part of the evaluation procedure, so that both are calibration-insensitive. It is however different from minDCF in the sense that it does not just evaluate at a single operating point. It represents detection performance over a range of operating points.

The DET-plot is derived from the empirical ROC plot by transforming both axes with the probit transformation, which can be expressed as $\text{probit}(p) = \sqrt{2}\,\text{erf}^{-1}(2p-1)$, where erf^{-1} is the *inverse error function* [67]. It sends the interval $[0,1]$ to $[-\infty,+\infty]$, in a very similar way to the logit transform.

A plot of $\text{probit}(P_{\text{fa}})$ against $\text{probit}(P_{\text{miss}})$ is known as a DET-plot. Since the probit axes are infinite, the DET is confined to a finite rectangle, defined by choosing P_{fa} and P_{miss} intervals of interest. The DET is useful to distinguish small differences between detectors, even in regions of the plot with low error-rates. As in the case of ROC, it is also just a finite set of discrete points, which if connected with horizontal and vertical line-segments, forms a steppy curve, which appears smooth if the database is large enough.

To see that the ROC/DET is calibration-insensitive, notice that any strictly monotonic rising calibration transformation of the scores does not change the plot. (A non-strict rising transformation, with flat regions, can lump scores together, thus removing some of the points.) In fact, the ROC/DET depends only on the order of the scores, not on their absolute values.

7.3.6 ROCCH

There is an attractive and principled solution for defining a continuous curve to summarize the points on the discrete ROC plot, known as the *ROC convex hull*, or ROCCH. The ROCCH is not new, having been introduced[9] in [68], but it is new to speaker recognition, where we believe it deserves more attention.

The ROCCH is obtained by *interpolating* between the discrete points of the ROC, effectively forming the convex hull of these points. We shall specifically refer to the *bottom, left boundary of the convex hull* as the ROCCH. See figure 7.2 for examples.

The ROCCH is a piecewise linear, convex, monotonically decreasing curve, with endpoints at $(P_{\text{fa}}, P_{\text{miss}}) = (0,1)$ and $(1,0)$. It has some of the same properties as the idealized continuous ROC discussed in section 7.2.2. Specifically, it is convex, and the negative of the slope gives a meaningful value for the likelihood-ratio which can be used in calibration analysis. Compare this to the

[9]See `http://home.comcast.net/~tom.fawcett/public_html/ROCCH` for more publications on the ROCCH.

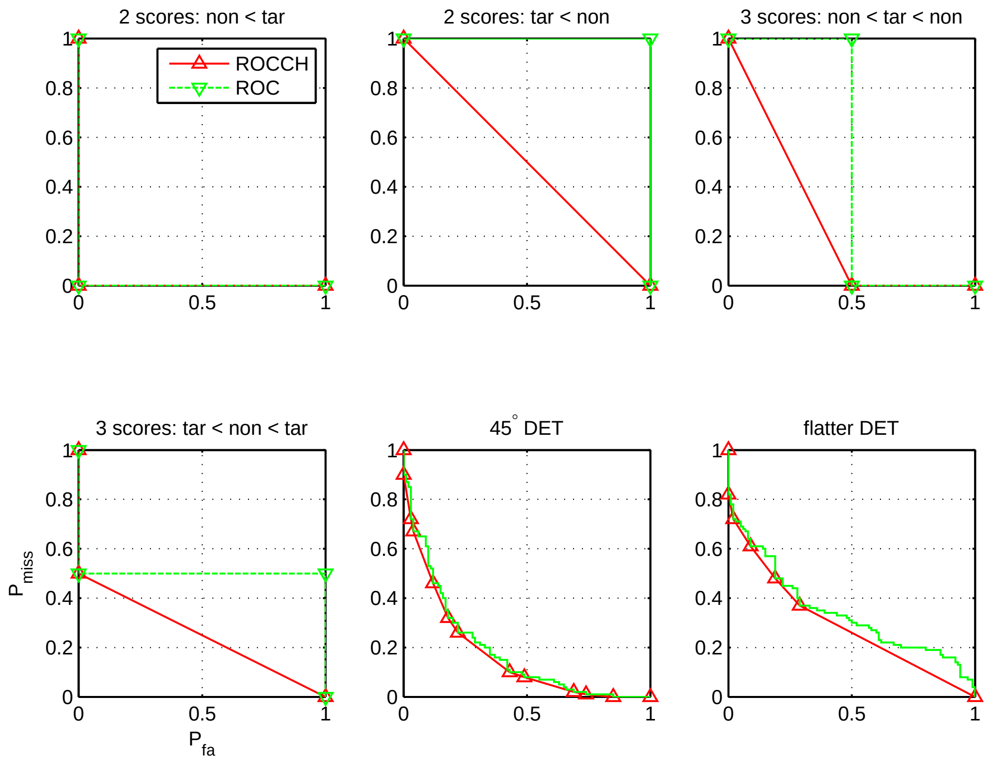

Figure 7.2: Examples of ROC (green) vs ROCCH (red). The ROC curves are steppy, the ROCCH curves are convex. The Δ marks ROCCH vertices and the ∇ marks ROC vertices. Where ROC and ROCCH coincide, the ROC obscures the ROCCH (the red is behind the green). The top left plot is for the case of two scores: a target (tar) and a non-target (non), where the non-target score is smaller than the target score. Similarly, the middle top has two scores; right top has three, with a non-target sandwiched between two targets; and bottom left has three, sandwiched the other way. The remaining two have many scores.

discrete ROC, which has only horizontal or vertical line-segments and thus no useful slope. (We elaborate on ROCCH calibration analysis in section 7.7.)

What follows is mostly contained in the results in [68], but the definition and proof of equivalence below are original.

Definition

Let the K points on the discrete ROC plot[10] be denoted $\mathbf{e}_i = \big(P_{\text{miss}}(\gamma_i), P_{\text{fa}}(\gamma_i)\big)$, for $i = 1, 2, \dots, K$. Let $\boldsymbol{\pi} = (\pi, 1-\pi)$ and $C_{\text{miss}} = C_{\text{fa}} = 1$, then the DCF at operating point π and threshold γ_i can be expressed in inner product notation as:

$$\text{DCF}_i(\pi) = \boldsymbol{\pi} \cdot \mathbf{e}_i = \pi P_{\text{miss}}(\gamma_i) + (1-\pi)P_{\text{fa}}(\gamma_i)\,. \tag{7.22}$$

Now consider the *convex hull* of the points $\mathbf{e}_i$, which is the set of all *interpolated points* of the form:

$$\mathbf{e}(\boldsymbol{\alpha}) = \sum_{i=1}^{K} \alpha_i \mathbf{e}_i \tag{7.23}$$

where the interpolating weights are positive and sum to one: $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_K) \in \mathbb{P}_K$. Now define E^*_{ch}, which minimizes the DCF within the convex hull:

$$E^*_{ch}(\pi) = \min_{\boldsymbol{\alpha}\in\mathbb{P}_K} \boldsymbol{\pi} \cdot \mathbf{e}(\boldsymbol{\alpha})\,. \tag{7.24}$$

The minimization ensures that for any minimizing $\boldsymbol{\alpha}$, the interpolated point $\mathbf{e}(\boldsymbol{\alpha})$ is on the bottom, left boundary of the convex hull—thus being a point on the ROCCH curve.

Equivalence between ROC and ROCCH

The points on the continuous ROCCH curve are equivalent to the discrete points of the ROC in the sense that they have the same minDCF, for any prior and cost weighting. To see this, we now show that for any DCF parametrization, π:

$$E^*_{ch}(\pi) = \text{minDCF}(\pi) \tag{7.25}$$

where $\text{minDCF}(\pi) = \min_i \text{DCF}_i(\pi)$.

Proof. Let m be a minimizing index, so that

$$\text{minDCF}(\pi) = \text{DCF}_m(\pi) \le \text{DCF}_i(\pi)\,. \tag{7.26}$$

[10]Here we work with the ordering, $(P_{\text{miss}}, P_{\text{fa}})$, to facilitate the concise inner-product notation. The reader is reminded that when we plot it, P_{fa} is on the horizontal axis and P_{miss} on the vertical axis.

By representing the summation constraint as $\alpha_m = 1 - \sum_{i \neq m} \alpha_i$, we can express the DCF of any point in the convex hull as:

$$\begin{aligned} \boldsymbol{\pi} \cdot \mathbf{e}(\boldsymbol{\alpha}) &= \boldsymbol{\pi} \cdot \sum_{i=1}^{K} \alpha_i \mathbf{e}_i \\ &= \boldsymbol{\pi} \cdot \mathbf{e}_m + \sum_{i \neq m} \alpha_i \left(\boldsymbol{\pi} \cdot \mathbf{e}_i - \boldsymbol{\pi} \cdot \mathbf{e}_m \right) \\ &= \text{minDCF}(\pi) + \sum_{i \neq m} \alpha_i \big(\text{DCF}_i(\pi) - \text{DCF}_m(\pi) \big) \,. \end{aligned} \tag{7.27}$$

By (7.26) and since $\alpha_i \geq 0$, no term in the summation can be negative, so that (7.27) is minimized at $\alpha_m = 1$, proving (7.25). ■

This minimum at $\alpha_m = 1$ is of course not very interesting, because it is not an interpolation, it is just one of the existing points on the discrete ROC. But it is easy to see that other, interpolated points on the ROCCH will also satisfy (7.24). Notice that the slope of the vector $\boldsymbol{\pi}$ is non-negative and that any line-segment on the ROCCH (the bottom, left boundary of the convex hull) has non-positive slope, so that we can always find some $\boldsymbol{\pi}$ orthogonal to the line-segment. For that value of $\boldsymbol{\pi}$, all points on the line-segment satisfy (7.24). By varying $\boldsymbol{\pi}$, and minimizing DCF, we can plot all points on the ROCCH.

It is straight-forward to generalize these results to DCF parametrization with any non-negative values for C_{miss} and C_{fa}.

In summary: Minimizing DCF over any interpolation of the discrete ROC points does not give any over-optimistic result. The possible values for minDCF are the same, whether one does a discrete minimization over the discrete ROC points, or over the continuous interpolation weights. This holds for any DCF parametrization, at any prior and any cost weighting.

Computation

The ROCCH can be conveniently and efficiently computed via the PAV algorithm, see [69]. We elaborate on the ROCCH-PAV relationship in section 7.7.

The ROCCH can be plotted[11] as a DET-curve, by using the probit transformation, but we doubt if it will replace the traditional steppy DET. See figure 7.3 for examples and note that the straight line-segments of the ROCCH become curves under the probit transform. More examples of ROCCH-DET-curves, plotted by the author as part of the Evalita 2009 Speaker Recognition Evaluation are available in [55].

The ROCCH is also a convenient tool for practical computation of minDCF in very large score sets. The ROCCH is efficiently computed via the PAV algorithm and the vertices of the ROCCH then give a very economical summary of

[11]The author's MATLAB tools for ROCCH-DET are available here: `http://focaltoolkit.googlepages.com`.

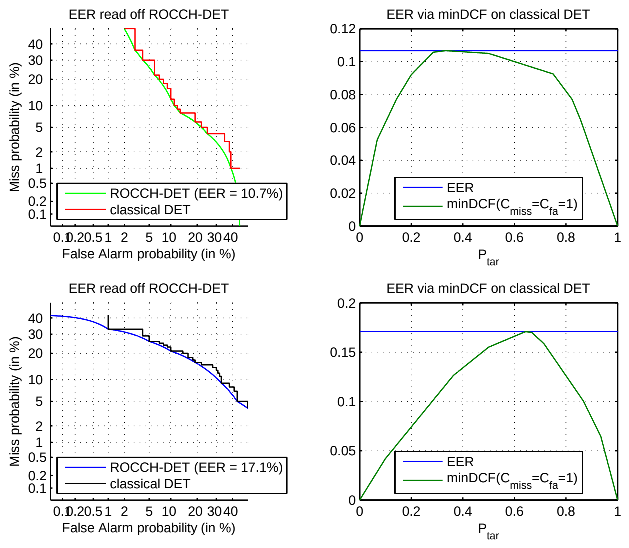

Figure 7.3: Two examples of ROCCH-DET vs classical steppy DET. The equality of ROCCH-EER and max minDCF is demonstrated.

the whole ROC, from which minDCF can be computed for any parametrization. We make use of this property in the calibration-analysis tools in section 7.6.

Finally, the ROCCH provides a principled way of computing EER, as discussed in the next subsection.

7.3.7 ROCCH-EER

We define the ROCCH-EER, which is newly proposed here, as the point on the ROCCH where $P_{\text{fa}} = P_{\text{miss}}$.

The traditional EER, as used in speaker recognition, is actually not clearly defined for the case of the discrete ROC. Usually none of the points in it satisfies $P_{\text{fa}} = P_{\text{miss}}$ exactly. One has to do some interpolation to make it so. The ROCCH-EER is a principled way of defining this interpolation, which conserves the value of minDCF. Since the ROCCH is readily computed, the ROCCH-EER is also readily computed.

Let us analyse the ROCCH-EER is some more detail. It turns out we can use an analogy to (7.8) to interpret the ROCCH-EER. We could try any of four ways to interpret EER:

$$\text{EER}_1 = \max_{\pi} \min_{\boldsymbol{\alpha}\in\mathbb{P}_K} \boldsymbol{\pi}\cdot\mathbf{e}(\boldsymbol{\alpha}) \tag{7.28}$$

$$\text{EER}_2 = \max_{\pi} \min_{i} \boldsymbol{\pi}\cdot\mathbf{e}_i \tag{7.29}$$

$$\text{EER}_3 = \min_{\boldsymbol{\alpha}\in\mathbb{P}_K} \max_{\pi} \boldsymbol{\pi}\cdot\mathbf{e}(\boldsymbol{\alpha}) \tag{7.30}$$

$$\text{EER}_4 = \min_{i} \max_{\pi} \boldsymbol{\pi}\cdot\mathbf{e}_i\,. \tag{7.31}$$

By (7.25) we already know that $\text{EER}_1 = \text{EER}_2$. Since $\boldsymbol{\pi}\cdot\mathbf{e}(\boldsymbol{\alpha})$ is bilinear,[12] this satisfies not only the conditions for Sion's minimax theorem [63], but also of Von Neumann's original minimax theorem [70], so that $\text{EER}_1 = \text{EER}_3$. However, it is easy to construct counter-examples that show that EER_4 is the odd one out. In summary: $\text{EER}_1 = \text{EER}_2 = \text{EER}_3 \leq \text{EER}_4$. Finally, we can use the formula for EER_3 in the same way as we did in (7.8) to show that $(\text{EER}_3, \text{EER}_3)$ is a point on the ROCCH. We conclude:

$$\begin{aligned}\text{ROCCH-EER} &= \text{EER}_1 = \text{EER}_2 = \text{EER}_3 \\ &= \max_{\pi} E^*_{ch}(\pi) \\ &= \max_{\pi}\ \text{minDCF}(\pi)\,.\end{aligned} \tag{7.32}$$

As before, the last form interprets ROCCH-EER as an upper-bound on error-rate at any prior, provided optimal thresholds are used. This is graphically demonstrated in figure 7.3.

Maximization over minDCF may also be a convenient way to compute ROCCH-EER if no implementation for ROCCH is available. It requires an iterative maximization search over π, but this is a nicely-behaved, one dimensional search for the maximum of a concave function. Our preferred implementation however, which is fast even for very large score sets, uses the above-mentioned PAV-algorithm to explicitly compute the vertices of the ROCCH.

Finally, it is interesting to note that in our experiments, when we generalize the formula for EER_4 in the way of (7.9), the resulting curve is just the traditional steppy curve connecting the points of the discrete ROC. Note however that this defines the whole *continuous curve*, not just the discrete points at the vertices of the steps. Treating the formula for EER_3 in the same way, plots out the ROCCH, as expected.

7.3.8 PRBEP

The EER generalization (7.9) may be employed to compute another standard point on the ROCCH curve. The *precision-recall-break-even-point*, or PRBEP [71], is the point at which the *absolute* numbers of misses and false-alarms are equal. (The EER equates normalized error-rates.) This is achieved

[12]linear in each of $\boldsymbol{\pi}$ and $\boldsymbol{\alpha}$

by choosing $\alpha = |\mathcal{T}_1|$ and $\beta = |\mathcal{T}_2|$ in equation (7.9) and applying it to the ROCCH. This criterion squeezes the last bit of use out of a database by choosing the operating point that maximizes the minimum of the two absolute error counts.

We find it good practice to report the PRBEP as the absolute number of errors (i.e. not normalized), to (i) emphasize that it can be used to compare different detectors on the same database, but that comparing PRBEP across databases is not meaningful, and (ii) to alert the evaluator if the errors are running out. If PRBEP is read off the ROCCH, the number of errors is typically not an integer, because of the interpolating nature of the ROCCH.

7.3.9 Summary of DCF and ROC and Bayes error-rate

We summarize this section in table 7.1, where all of the evaluation criteria discussed in this section are compared. Some notes are in order:

The table includes three criteria not discussed in this section: $\mathcal{E}_{\text{log}}$, which is the subject of the next section and also $\mathcal{E}_{\text{err}}^{\text{min}}$, $\mathcal{E}_{\text{log}}^{\text{min}}$, which will be discussed in sections 7.6 and 7.7.

We are not aware of publications in speaker recognition where the parametrization of minDCF has been varied over applications. Usually minDCF is just presented as a single number at the NIST-specified operating point. However, there is nothing in principle that prevents it from being used to produce an application-spanning plot similar to our figure 7.1. This is indeed what we shall do in section 7.6 below.

The main innovation presented in this section was the generalization of DCF to $\mathcal{E}_{\text{err}}$. The former evaluates hard decisions, while the latter evaluates *soft decisions* in log-likelihood-ratio (llr) form. $\mathcal{E}_{\text{err}}$ forms an evaluation criterion that is at the same time calibration-sensitive and variable over applications.

In the next section, we discuss how $\mathcal{E}_{\text{log}}$ forms a calibration-sensitive summary over applications.

7.4 Calibration-sensitive summary criteria

In the previous section, we showed how to complement the existing detection evaluation tools, by introducing $\mathcal{E}_{\text{err}}$, which is calibration-sensitive in the manner of DCF (which does not span applications), and which *also* spans applications in the manner of the DET-curve (which does not evaluate goodness of calibration).

$\mathcal{E}_{\text{err}}$ spans applications by essentially doing a separate evaluation at each of very many operating points and graphically representing all of them. However, for some purposes one needs a scalar criterion. Most notably, for discriminative training, a scalar objective function is imperative. If the discriminatively

Table 7.1: Evaluation criterion comparison

criterion	evaluates	calibration-sensitive	spans applications	scalar summary
DCF	decision	yes	no	no
minDCF/$\mathcal{E}_{\text{err}}^{\min}$	score	no	as plot	no
EER	score	no	as upper bound	yes
AUC	score	no	as integral	yes
ROC/DET	score	no	as plot	no
$\mathcal{E}_{\log}^{\min}$	score	no	as integral	yes
$\mathcal{E}_{\text{err}}$	llr	yes	as plot	no
$\mathcal{E}_{\log}$	llr	yes	as integral	yes

trained detector is intended for use over a range of different applications, then the discriminative training objective should also somehow span this range of applications.

The tools for constructing such application-spanning summaries were presented in chapters 3 and 6, where we showed how to form expectations/combinations over families of cost functions/proper scoring rules/applications. In particular, we showed in section 6.4 that evaluation by logarithmic cost can be interpreted as such:

$$\mathcal{E}_{\log}(\mathcal{W}|0.5) = \int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}\big(\mathcal{W}|\,\text{logit}^{-1}\,\tau\big)\,d\tau \quad = \int_{0}^{1} \mathcal{E}_{\eta}\big(\mathcal{W}|0.5\big)\,d\eta\,. \tag{7.33}$$

The experience of this author and others has shown that this criterion, $\mathcal{E}_{\log}$, works very well in practice as a discriminative training objective. It was introduced to the speaker recognition community by the author in 2004, see [16], followed by a journal paper in 2006, see [17]. We have used it for discriminative training purposes in all of the NIST SREs from 2005 to 2010. It works equally well for multi-class problems, where we have used it also for the NIST Language Recognition Evaluations between 2005 and 2009. It is also the principle behind the author's *FoCal Toolkit*,[13] which has been used by many other laboratories, from 2005 to the present, for speaker and language recognition. We will discuss discriminative training using $\mathcal{E}_{\log}$ in chapter 8.

Moreover, the information-theoretic interpretation of $\mathcal{E}_{\log}$ has made it attractive to researchers in the forensic speaker recognition community, as a tool to evaluate the goodness of speaker recognition evidence delivered to court in likelihood-ratio form [29, 32, 31].

In the rest of this section we are interested in how $\mathcal{E}_{\log}$ forms a *calibration-sensitive summary*:

[13]The FoCal Toolkit, coded in MATLAB, is freely available here: `http://focaltoolkit.googlepages.com`.

- The integrals of (7.33) already show that $\mathcal{E}_{\log}$ is a summary, but they do not yet make it clear whether it summarizes all possible two-class applications. In the next subsection, we show that indeed it does.
- $\mathcal{E}_{\log}$ is by no means the only such summary. We discuss a family of summary measures and then motivate why we prefer $\mathcal{E}_{\log}$ out of this family.
- We conclude this section with a discussion of calibration sensitivity.

7.4.1 Regular binary proper scoring rules

Here we show that $C_{\log}$ is a member of a family of proper scoring rules, each of which acts as a linear (additive) summary over *all* applications that make Bayes decisions with categorical probability distributions over two classes. That is, we are interested in all applications where the recognizer provides a posterior probability, q, for the target class, which has to be used to make some decision, $a \in \mathcal{A}$, that optimizes the expected value, $C_{\boldsymbol{\tau}}(a|q)$, of some cost function $C_{\boldsymbol{\tau}}$.

Chapter 2 showed that every such application can be evaluated via proper scoring rule, $C^*_{\tau}(q|\theta)$. When there are two classes ($N = 2$), we use the terminology *binary* proper scoring rule. What remains now is (i) to show how to represent all binary proper scoring rules as an appropriately defined family, and (ii) that some members of this family, including $C_{\log}$, act as summaries for the whole family.

It is possible to construct pathological proper scoring rules that are always infinite. Excluding such pathologies, we are left with what are known in the literature as *regular* proper scoring rules, see [8] and references there-in. There are different ways to characterize all regular binary proper scoring rules. Here we propose a variant of previously published characterizations and then show how it is equivalent to those characterizations. Our characterization is:

$$C^*_\rho(q|\theta) = \int_0^1 C^*_\eta(q|\theta)\rho(\eta)\,d\eta \tag{7.34}$$

where $\rho(\eta)$ is a normalized probability distribution, with $\int_0^1 \rho(\eta)\,d\eta = 1$. By choosing the distribution $\rho(\eta)$, we can form any regular binary proper scoring rule in canonical form. As already shown, $\rho(\eta) = 1$ gives $C_{\log}$. We give other examples later. If we insert definition (3.12) of C^*_η, we get:

$$C^*_\rho(q|\theta_1) = \int_q^1 \frac{\rho(\eta)}{\eta}\,d\eta\,, \qquad\qquad C^*_\rho(q|\theta_2) = \int_0^q \frac{\rho(\eta)}{1-\eta}\,d\eta\,. \tag{7.35}$$

To see that C^*_ρ is in canonical form, notice that $C^*_\rho(1|\theta_1) = 0$ and $C^*_\rho(0|\theta_2) = 0$ and that the scale is normalized by virtue of the normalization of $\rho(\eta)$. By equivalence, as explained in section 3.4, we do not have to consider any proper scoring rules that are not in canonical form.

What happens when $\rho(\eta) \geq 0$, but it cannot be normalized, so that $\int_0^1 \rho(\eta)\, d\eta \to \infty$? Since $\frac{1}{\eta}\rho(\eta) \geq \rho(\eta) \leq \frac{1}{1-\eta}\rho(\eta)$ in the interval $\eta \in [0,1]$, we have $\int_0^q \frac{1}{1-\eta}\rho(\eta)\, d\eta + \int_q^1 \frac{1}{\eta}\rho(\eta)\, d\eta \geq \int_0^1 \rho(\eta)\, d\eta$. This means that if $\rho(\eta)$ cannot be normalized, then one or both of $C^*_\rho(\theta_1, q)$ and $C^*_\rho(\theta_2, q)$ must also be infinite, for every q. This pathology is clearly not a useful proper scoring rule. We can therefore, without loss of generality, restrict $\rho(\eta)$ to be a normalized distribution.

In the literature, the class of regular binary proper scoring rules has been expressed by integrals that are equivalent (but not identical in form) to our (7.35). See for example [35], where the form

$$\int_q^1 \rho'(\eta)\, d\eta, \qquad \int_0^q \frac{\eta}{1-\eta}\rho'(\eta)\, d\eta, \qquad \rho'(\eta) \geq 0, \text{ for } 0 \leq \eta \leq 1$$

is used; or [13] and [8] where

$$\int_q^1 (1-\eta)\rho''(\eta)\, d\eta, \qquad \int_0^q \eta\rho''(\eta)\, d\eta, \qquad \rho''(\eta) \geq 0, \text{ for } 0 \leq \eta \leq 1$$

is used. Equivalence to (7.35) is established by letting $\rho'(\eta) = \frac{\rho(\eta)}{\eta}$ and $\rho''(\eta) = \frac{\rho(\eta)}{\eta(1-\eta)}$. The advantage of the form (7.35) which we adopt here, is that the weighting function $\rho(\eta)$ is always in the form of a *normalized* probability density, which gives the natural interpretation of *expectation* to these integrals. In contrast, if for example we wanted to construct the logarithmic scoring rule via ρ' or ρ'', we would need to choose $\rho'(\eta) = \frac{1}{\eta}$, or $\rho''(\eta) = \frac{1}{\eta(1-\eta)}$, neither of which can be normalized.

Next we consider some concrete examples of what (7.34) gives with different distributions, $\rho(\eta)$:

Binary decisions

If we concentrate the distribution to a point mass, or Dirac-delta, as $\rho(\eta) = \delta(\eta - \eta_1)$, we recover the integrand at η_1, so that $C^*_\rho = C^*_{\eta_1}$. This is just the normalized proper scoring rule obtained when making binary accept/reject decisions, where the cost ratio gives a Bayes threshold of η_1. See equation (3.12).

Ternary decisions

If we use the convex combination of two point masses: $\rho(\eta) = \alpha\delta(\eta - \eta_1) + (1-\alpha)\delta(\eta - \eta_2)$, we get a proper scoring rule associated with applications that require *ternary* accept/reject/undecided decisions. If $\eta_1 < \eta_2$, then the decision is *reject* if $q < \eta_1$, *undecided* if $\eta_1 \leq q < \eta_2$, or *accept* if $\eta_2 \leq q$. There are $2 \times 3 = 6$ outcomes, two of which, (accept, target) and (reject, non-target) have zero cost. The remaining four have non-zero weights, determined by the three parameters η_1, η_2, α. There are only 3 parameters for 4 weights, because of the normalization constraint (just as η_1 determines 2 weights in the previous example). The reader may want to refer back to equation (3.27) and figure 3.1 where we analysed (a scaled version of) this two-threshold proper scoring rule.

M-ary decisions

We can generalize the above with $\rho(\eta) = \sum_{i=1}^{M-1} \alpha_i \delta(\eta - \eta_i)$. This represents an application requiring M-ary decisions, with a set, $\mathcal{A} = a_1, a_2, \dots, a_M$, of M discrete decisions. These decisions could, for example, represent a discrete scale, ranging from strong support for the target class, through neutrality, to strong support for the non-target class.

Strictly proper scoring rules as summary criteria

Any applications requiring discrete decisions can be represented with one or more point masses in $\rho(\eta)$. This is also true for applications that require compound decisions and it is also true for mixtures of applications. We can construct a summary evaluation criterion, summarizing all discrete-decision applications (still constrained to $N = 2$ classes), by weighting all operating points with non-zero weight. In fact, if we let $\rho(\eta) > 0$ everywhere in the interval $\eta \in (0, 1)$, then C^*_ρ is a *strictly* proper scoring rule, see section C.3 in the appendix, or [8] theorem 3.

We now have the desired result: The family C^*_ρ, the regular binary proper scoring rules, represents all two-class Bayes decision applications. If $\rho(\eta)$ is strictly positive, we get the family of regular binary strictly proper scoring rules, each of which summarizes all these applications as a weighted combination. What remains is which weighting to choose. Examples are:

- $\rho(\eta) = 1$ gives: $C^*_\rho(q|\theta_i) = C_{\log}(q|\theta_i)$.
- $\rho(\eta) = 6\eta(1-\eta)$ gives: $C^*_\rho(q|\theta_i) = 3C_{\text{Brier}}(q|\theta_i)$, where
$$C_{\text{Brier}}(q|\theta_1) = (1-q)^2\,, \qquad C_{\text{Brier}}(q|\theta_2) = q^2\,. \tag{7.36}$$
  The Brier scoring rule is possibly the oldest proper scoring rule. It was proposed in 1950, for evaluating probabilistic weather forecasts [3].
- $\rho(\eta) = \frac{1}{\pi\sqrt{\eta(1-\eta)}}$ gives: $C^*_\rho(\theta, q) = \frac{2}{\pi} C_{\text{boost}}(\theta, q)$, where
$$C_{\text{boost}}(q|\theta_1) = \sqrt{\frac{1-q}{q}}\,, \qquad C_{\text{boost}}(q|\theta_2) = \sqrt{\frac{q}{1-q}} \tag{7.37}$$
  is a proper scoring rule which has recently been shown to be the objective that is effectively optimized by the discriminative machine-learning technique of *boosting*, see [13].

7.4.2 Choice of strictly proper scoring rule

We have now established that a strictly proper scoring rule forms the kind of summary criterion we seek. What remains is to analyse some different candidates and to show in which respects the logarithmic rule is preferable. To do this, we follow [13], and embed the Brier, logarithmic and boosting rules into a parametric family.

(In [13] they use both parameters of the beta independently to create asymmetric scoring rules, but our solution to introducing asymmetry in the application is to vary the prior instead. This will be discussed under discriminative training in chapter 8.)

Symmetric beta family of proper scoring rules

The Brier, logarithmic and boosting rules have in common that each is induced by a distribution of the form $\rho(\eta) = k\eta^\gamma(1-\eta)^\gamma$, for some exponent γ. In fact, this is just a special case of the beta or Dirichlet[14] distributions [1]. The beta distribution has two parameters and if they are equal, the distribution is symmetric. We now limit our attention to the subset of strictly proper scoring rules that are induced when ρ is a symmetric beta distribution and we analyse their nature for different values of the exponent γ. Let

$$\rho(\eta) = \text{Beta}(\eta|\gamma+1,\gamma+1) = \frac{\Gamma(2\gamma+2)}{\Gamma^2(\gamma+1)}\eta^\gamma(1-\eta)^\gamma \tag{7.38}$$

where Beta is the *beta distribution* [1] and Γ is the *gamma function* [67, 72]. Since $\text{Beta}(\eta|a,b)$ can be normalized only when $a,b>0$, this also limits $\gamma > -1$.

The proper scoring rules induced by the integrals of (7.35), with this choice of ρ can be analytically solved for the cases $\gamma = -\frac{1}{2}, 0, \frac{1}{2}, 1, 1\frac{1}{2}, 2, 2\frac{1}{2}, \cdots$, or they can be numerically evaluated for $-1 < \gamma < \infty$ via implementations of the *beta*,[15] *incomplete beta* and/or *gamma* functions [13, 72, 67].

We already know $\gamma = -\frac{1}{2}$ gives C_{boost}, $\gamma = 0$ gives $C_{\log}$ and $\gamma = 1$ gives C_{Brier}. Increasing γ further takes the proper scoring rule closer to a step function, until at $\gamma \to \infty$ it degenerates into the non-strict C^*_η, with $\eta = 0.5$. This is shown in figure 7.4.

Choice of γ

Here we motivate our preferred value of $\gamma = 0$, which finally clinches the deal for logarithmic cost.

[14]The beta distribution is the special case of the N-state Dirichlet distribution, when $N = 2$.

[15]The beta *function* is not to be confused with the beta *distribution*. The former is the normalization factor of the latter.

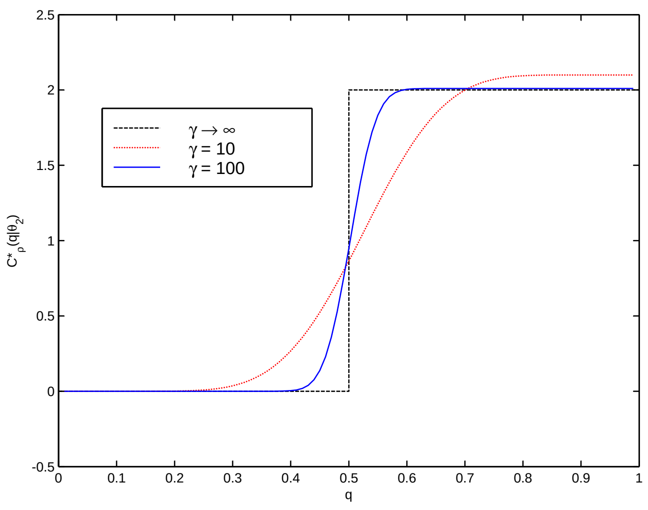

Figure 7.4: $\lim_{\gamma\to\infty} C^*_\rho(q|\theta_2) \to \dfrac{\mathrm{u}(q-0.5)}{0.5}$.

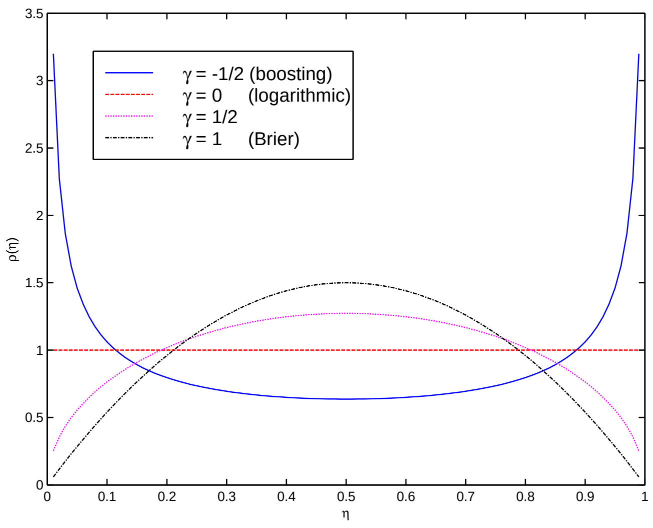

Figure 7.5: Weighting distribution, $\rho(\eta)$, of (7.38), for a few values of γ.

The salient effect of varying γ is the behaviour of $\rho(\eta)$ at $\eta = 0$ and $\eta = 1$. Recall that η is the Bayes decision threshold as applied to the posterior distribution. For $-1 < \gamma \leq 0$, these two values of η get *non-zero* weight, while for $\gamma > 0$, they get *zero* weight. For example, for $\gamma = 0$, $\rho(0) = \rho(1) = 1$; but for $\gamma = 1$, $\rho(0) = \rho(1) = 0$. This is demonstrated in figure 7.5.

The effect of γ on the induced proper scoring rule is as follows:

- When $-1 < \gamma \leq 0$, then $C^*_\rho(0|\theta_1) = C^*_\rho(1|\theta_2) = \infty$ is *unbounded.*
- When $\gamma > 0$, then $C^*_\rho(0|\theta_1) = C^*_\rho(1|\theta_2) < \infty$ is *bounded.*

This is demonstrated in figure 7.6.

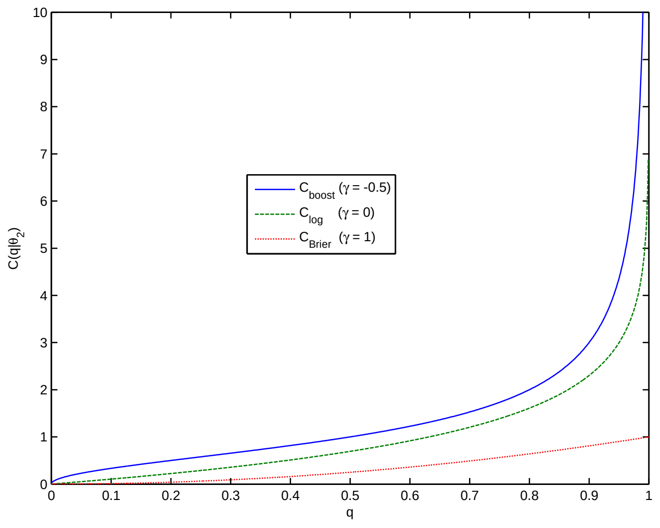

Figure 7.6: Comparison of three strictly proper scoring rules, showing that when $\gamma > 0$, the rule is bounded. $C(q|\theta_2)$ is shown, $C(q|\theta_1)$ is symmetrical about 0.5.

To understand why this is important, recall that η parametrizes the application, where $(C_{\text{miss}}, C_{\text{fa}}) = (\frac{1}{\eta}, \frac{1}{1-\eta})$. As η approaches 0 or 1, it represents applications for which the risk of bad decisions may be arbitrarily high. The weighting function $\rho(\eta)$ forms an expectation over these applications. If the weighting function goes to zero at $\eta = 0$ and $\eta = 1$, then we are effectively excluding these applications from consideration. It is obvious from equation (7.38) that this happens for any $\gamma > 0$.

In real-world applications, the consequences of bad decisions can be *arbitrarily* bad. If a recognizer outputs a log-likelihood-ratio of $\pm\infty$, which translates to a posterior probability of exactly 1 for one of the propositions, then a Bayes decision would be made in favour of that proposition, *regardless* of the cost of error. But if the recognizer is wrong, then that potentially high cost would be incurred. More generally, the larger the log-likelihood-ratio magnitude, the larger the potential costs of bad decisions. The proper scoring rule that evaluates the recognizer output should reflect this.

We therefore consider it imperative when evaluating recognizers that are to be used to make real decisions, to also include such consequences into consideration. For this reason, we choose to work only with strictly proper scoring rules that effectively include these extremes and that therefore satisfy $\rho(\eta) > 0$ over the *closed* interval $\eta \in [0, 1]$. For the γ-family, this translates to $\gamma \in (-1, 0]$, which gives $C^*_\rho(0|\theta_1) = C^*_\rho(1|\theta_2) = \infty$. Note however, that $0 < C^*_\rho(q|\theta) < \infty$ when $0 < q < 1$ and that the zeros and infinities happen only at $q = 0$ or $q = 1$.

But this is not the only reason for wanting unbounded cost functions. There is also the further practical consideration of obtaining an objective that is amenable to numerical optimization. As has been shown with the recent successes of machine-learning techniques such as support-vector-machines and boosting, *convex* optimization objectives are a great advantage. It turns out that unbounded proper scoring rules tend to lead to convex optimization problems such as boosting and logistic regression, while bounded proper scoring rules such as the quadratic Brier score lead to non-convex optimizations such as are common in neural networks [73, 13].

We have now established that if we work within the γ-family, then our choice of parameter must be limited to $-1 < \gamma \leq 0$. Let us now finally make a choice within this interval. The more negative we make γ, the more weight is placed at the extremities, see figure 7.5. We argue here that placing too much weight at the extremities is bad for the purpose of constructing a general evaluation criterion. This is because of the limitations imposed by the size and accuracy of the evaluation database, as discussed in section 7.3.2. Placing too much weight near $\eta = 0$ or $\eta = 1$ amplifies the inaccuracies in error-rate estimates that result there. By this argument it is desirable to place as little weight as possible on the extremities.

Minimizing weight at the extremities, subject the the unboundedness constraint gives $\gamma = 0$ and therefore the logarithmic scoring rule.

This argument for preferring the logarithmic scoring rule does not exclude the use of other scoring rules for all purposes. There may be situations in which using others gives useful results, as has been demonstrated by the success of boosting for certain machine learning problems [13]. However, in the absence of good reasons to choose something else, the logarithmic scoring rule is a sensible default choice.

7.4.3 The logarithmic evaluation criterion

Here we assemble the evaluation criterion, $\mathcal{E}_{\text{log}}$, which results when using the logarithmic scoring rule for two classes.

The logarithmic scoring rule is usually defined as $C_{\text{log}}(\mathbf{q}|\theta_i) = -\log q_i$, where $\mathbf{q}$ is the posterior probability distribution output by the recognizer. In our context however, for the case of two classes, we are using it to evaluate the detector output in log-likelihood-ratio form. For a log-likelihood-ratio w and a target prior π we find:

$$\begin{aligned} C_{\log}(\mathcal{B}(w,\pi)|\text{target}) &= \log\bigl(1+\exp(-w-\operatorname{logit}\pi)\bigr) \\ C_{\log}(\mathcal{B}(w,\pi)|\text{non-target}) &= \log\bigl(1+\exp(w+\operatorname{logit}\pi)\bigr) \end{aligned} \tag{7.39}$$

where $\mathcal{B}$ is Bayes' rule. To see how this rule works as a function of w, notice first that $\operatorname{logit}\pi$ just causes a translation along the input axis. Let us arbitrarily zero this shift at $\operatorname{logit}\frac{1}{2}=0$ and then examine the behaviour of the cost given that the target proposition is true: $\log\bigl(1+\exp(-w)\bigr)$:

- For a good recognizer output, with $w \gg 1$, the cost vanishes: $\log\bigl(1+\exp(-w)\bigr) \approx \exp(-w) \ll 1$.
- For a bad recognizer output, with $w \ll -1$, the cost grows linearly (and thus unboundedly) with the magnitude of w: $\log\bigl(1+\exp(-w)\bigr) \approx -w \gg 1$.
- For a non-committal recognizer output, $w \approx 0$: $\log\bigl(1+\exp(-w)\bigr) \approx \log 2$.

The case of non-targets is symmetric on the w axis about $\operatorname{logit}\pi$. See figure 7.7.

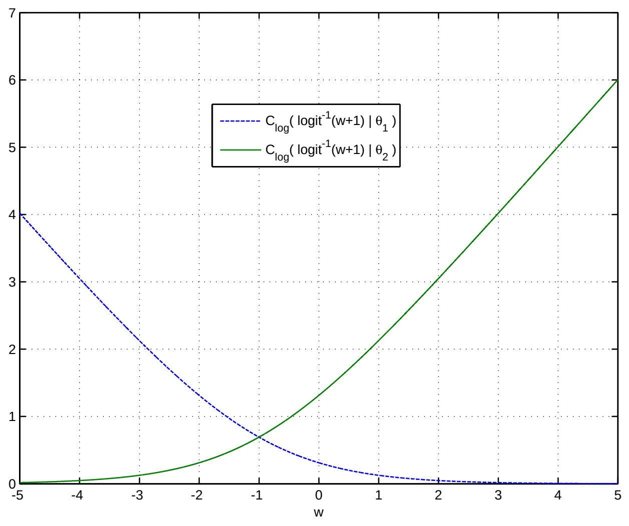

Figure 7.7: Logarithmic cost vs log-likelihood-ratio, w, at $\operatorname{logit}\tilde{\pi}=1$.

Forming our empirical evaluation criterion, with the supervised evaluation database, by plugging (7.39) into (5.4), we get:

$$\begin{aligned} \mathcal{E}_{\log}(\mathcal{W}|\pi) = &\frac{\pi}{|\mathcal{T}_1|}\sum_{t\in\mathcal{T}_1} C_{\log}(\mathcal{B}(w,\pi)|\text{target}) \\ &+ \frac{1-\pi}{|\mathcal{T}_2|}\sum_{t\in\mathcal{T}_2} C_{\log}(\mathcal{B}(w,\pi)|\text{non-target})\,. \end{aligned} \tag{7.40}$$

Nomenclature

In our paper [17], we denoted the criterion formed at $\tilde{\pi} = 0.5$, as:

$$C_{\text{llr}}(\mathcal{W}) = \frac{1}{\log 2}\mathcal{E}_{\text{log}}(\mathcal{W}|0.5) \tag{7.41}$$

where the subscript *llr* indicated that the criterion evaluates log-likelihood-ratios.

In the Ph.D. dissertation [29] of Daniel Ramos, $\mathcal{E}_{\text{log}}(\mathcal{W}|\pi)$ is referred to as *empirical cross-entropy*, or ECE.

Computation

Practical computation of $\mathcal{E}_{\text{log}}$ via (7.40) is straight-forward and very similar in implementation to traditional evaluation performed by counting of errors. The only detail that needs some care is for example when doing discriminative training, when large scores can result, to avoid numerical overflow of the exponent. If $\exp(w)$ would overflow, just use the approximation $\log(1 + \exp(w)) \approx w$.

Calibration sensitivity

To see that $\mathcal{E}_{\text{log}}$ is calibration-sensitive, consider that the value of C_{log} and therefore $\mathcal{E}_{\text{log}}$ can be arbitrarily large. If there is but one target trial for which the recognizer gives posterior 0, or one non-target for which the recognizer gives posterior 1, then the whole evaluation criterion becomes infinite. Recall that we concluded that large values, greater than the reference, $\mathcal{E}_{\text{log}}(\mathcal{W}_0|\pi)$, given by the default recognizer, must be due to bad calibration, whatever the definition of calibration.

Applications differ appreciably in their sensitivity to calibration. An extreme case is $\mathcal{E}_{\text{err}}(\mathcal{W}|0.5)$, which will show bad calibration that exceeds $\mathcal{E}_{\text{err}}(\mathcal{W}_0|0.5) = 0.5$ only if the signs of all the scores are reversed, in which case the value may reach 1. However, $\mathcal{E}_\eta$, if η is close to 0 or to 1, can attain large values. Evaluation by logarithmic cost can be understood to reach large values because as a summary over applications, it also includes such high-cost applications.

The calibration-sensitivity of $\mathcal{E}_{\text{log}}$ (as well as evaluation by other strictly proper scoring rules) is in contrast to the traditional summary measures such as EER and AUC, which evaluate uncalibrated scores.

7.4.4 Summary of summary criteria

In this section, we showed that binary strictly proper scoring rules form linear additive summaries over all applications that make two-class Bayes decisions. This forms a class of application-spanning, calibration-sensitive, summary criteria. We argued that $\mathcal{E}_{\text{log}}$ is a good default choice within this class.

7.5 Evaluating calibration

So far in this chapter, we showed how to generalize the traditional evaluation criteria to create new, calibration-sensitive, application-spanning criteria. These criteria are useful when developing and testing detectors that are designed to make hard Bayes decisions over a range of applications.

Now we investigate how to complement our new calibration-sensitive criteria with calibration-insensitive versions, so that the contrast effectively evaluate goodness of calibration. In short, in the same way that DCF − minDCF traditionally evaluates calibration, we now define $\mathcal{E}_{\text{err}}^{\min}$ and $\mathcal{E}_{\log}^{\min}$, so that $\mathcal{E}_{\text{err}} - \mathcal{E}_{\text{err}}^{\min}$, or $\mathcal{E}_{\log} - \mathcal{E}_{\log}^{\min}$, can be used in the same way.

The traditional way of measuring calibration as $L_{\text{cal}} = \text{DCF} - \text{minDCF}$ in section 7.3.3 agrees with our conclusion in section 5.3.1 of how to measure calibration. However it has the limitation that it is fixed at the specific DCF parametrization and therefore represents only the single application that requires binary decisions at the given prior and cost. In the next two sections, we show how to overcome this limitation in two ways:

- We replace DCF by $\mathcal{E}_{\text{err}}$, thereby allowing a sweep over applications and thus giving a plot of calibration quality against the application parameter. This is presented in the next section, directly below.

- The above is already a very useful tool, but it is limited to evaluation by the proper scoring rule C_{err}^*. We go further and show how to generalize this recipe to arbitrary proper scoring rules. This effectively defines scalar summary criteria for goodness of calibration. This is presented in section 7.7.

7.6 Normalized Bayes error-rate plots

Here we generalize the traditional DCF/minDCF calibration decomposition, which evaluates decisions and scores at a fixed application, to the $\mathcal{E}_{\text{err}}/\mathcal{E}_{\text{err}}^{\min}$ decomposition, which evaluates log-likelihood-ratios over a range of applications. Given our analysis in this chapter, the recipe is straight-forward:

The recognizer, $\mathcal{W}$, provides output, w_t, in *calibrated* log-likelihood-ratio format for every trial t of the supervised evaluation database. This is evaluated by (7.18) as $\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})$. As already demonstrated in figure 7.1, this criterion can be normalized and plotted as a function of the application parameter (the effective prior), $\tilde{\pi}$.

Now treat the same submitted log-likelihood-ratios, w_t, as *uncalibrated* scores and let the evaluator do the calibration. This takes the form of computing the traditional minDCF (which evaluates scores), but for multiple closely

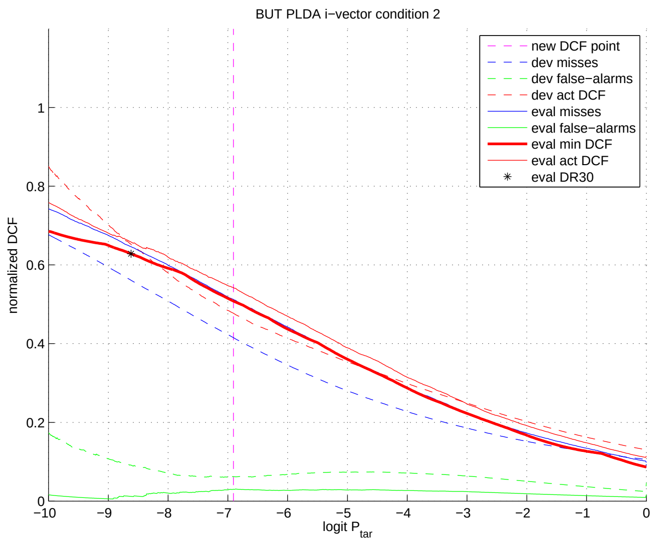

Figure 7.8: Normalized Bayes error-rate plot for an SRE 2010 speaker detector with *good* calibration. Here *eval* denotes the evaluation database and *dev* the development database. DCF and minDCF refer to $\mathcal{E}_{\text{err}}$ and $\mathcal{E}_{\text{err}}^{\text{min}}$. P_{miss} and normalized P_{fa} are also shown separately. DR30 refers to the point to the left of which there are fewer than 30 false-alarms. The vertical magenta dashed line represents the *new operating point* at $\tilde{\pi} = 0.001$.

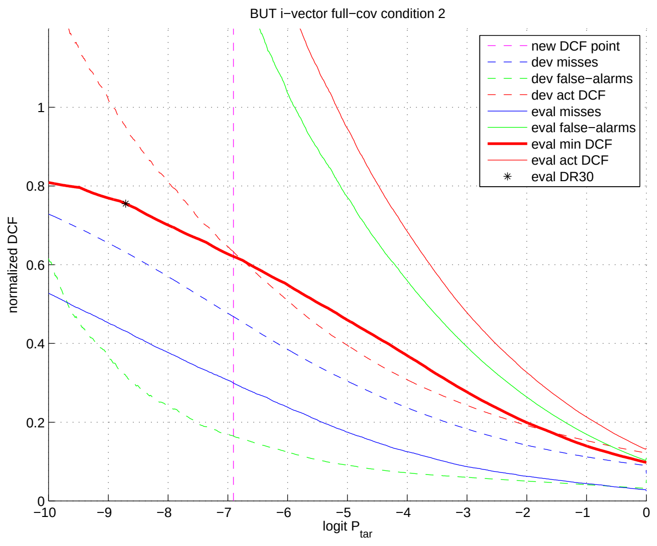

Figure 7.9: Normalized Bayes error-rate plot for an SRE 2010 speaker detector with *bad* calibration. See caption of figure 7.8 for details.

spaced values of $\tilde{\pi}$ over the plotting range range of interest. We define:

$$\mathcal{E}_{\text{err}}^{\text{min}}(\mathcal{W}|\tilde{\pi}) = \text{minDCF}(\tilde{\pi}) = \min_{i} \tilde{\pi} P_{\text{miss}}(\gamma_i) + (1-\tilde{\pi}) P_{\text{fa}}(\gamma_i) \tag{7.42}$$

where as explained above, the γ_i are all of the score threshold values that give different $(P_{\text{fa}}, P_{\text{miss}})$ points in the empirical ROC. Now $\mathcal{E}_{\text{err}}^{\min}$ can be normalized and plotted alongside $\mathcal{E}_{\text{err}}$.

The minimization ensures: $\mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\tilde{\pi}) \leq \mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})$. If these two are close, the calibration is good, if they are very different, the calibration is bad. For a good detector, it is desirable to have good calibration everywhere.

As before, we assume all the scores are finite, so that there is always a threshold that makes either one of the error-rates zero, so that $\mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\tilde{\pi}) \leq \mathcal{E}_{\text{err}}(\mathcal{W}_0|\tilde{\pi}) = \min(\tilde{\pi}, 1-\tilde{\pi})$. If we normalize by $\mathcal{E}_{\text{err}}(\mathcal{W}_0|\tilde{\pi})$, then:

$$\frac{\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})}{\mathcal{E}_{\text{err}}(\mathcal{W}_0|\tilde{\pi})} \geq \frac{\mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\tilde{\pi})}{\mathcal{E}_{\text{err}}(\mathcal{W}_0|\tilde{\pi})} \leq 1\,. \tag{7.43}$$

This agrees with our previous conclusion: any performance worse than the default is due to bad calibration.

This plot of normalized $\mathcal{E}_{\text{err}}$ and $\mathcal{E}_{\text{err}}^{\min}$ vs $h = \operatorname{logit}\tilde{\pi}$ is the author's current favourite tool for judging the calibration of a given speaker detector and it is the tool we used both in preparation for the NIST 2010 Speaker Recognition Evaluation (SRE2010) and for our analysis of the results afterwards. In figures 7.8 and 7.9, we show examples of two subsystems submitted to SRE2010, with respectively good and bad calibration.

Notice that when the horizontal axis is $h = \operatorname{logit}\tilde{\pi}$, then for the region $h < 0$, which we plot in these figures, the vertical axis (normalized error-rate) is:

$$\begin{aligned}
v &= \frac{\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})}{\min(\tilde{\pi}, 1-\tilde{\pi})} \\
&= \frac{\tilde{\pi}P_{\text{miss}}(\tilde{\pi}) + (1-\tilde{\pi})P_{\text{fa}}(\tilde{\pi})}{\tilde{\pi}} \\
&= P_{\text{miss}}(\tilde{\pi}) + \exp(-\operatorname{logit}\tilde{\pi})P_{\text{fa}}(\tilde{\pi}) \\
&= P_{\text{miss}}(\operatorname{logit}^{-1} h) + \exp(-h)P_{\text{fa}}(\operatorname{logit}^{-1} h)
\end{aligned} \tag{7.44}$$

where we used (7.18). The exponential amplification of false-alarms induced by this normalization explains the shape of the $\mathcal{E}_{\text{err}}$ curves for regions of bad calibration. Some form of amplifying normalization is needed to make the effects of calibration visible in regions of low error-rate. This normalization is the main difference between these curves and APE-curves (see section 7.7.4 below). The normalized Bayes error-rate plot is able to display a wider range of operating points than the APE-curve.

Finally, recall the discussion of section 7.3.2, which effectively means one cannot extend the horizontal axis indefinitely in either direction, because the errors will run out somewhere along the way. To make this effect explicit, we plot what we call the *DR30 point*, to the left of which the absolute number false-alarms drops below 30. This point is on the $\mathcal{E}_{\text{err}}^{\min}$ curve, because we use the false-alarm count which results from the evaluator's optimized threshold. DR30 refers to Doddington's Rule of 30, see appendix B.

7.6.1 Computation

In SRE2010, large score sets—up to a few million trials—were needed to support the new operating point. With inefficient algorithms, evaluation of a single detector by $\mathcal{E}_{\text{err}}$ and $\mathcal{E}_{\text{err}}^{\text{min}}$ may take several minutes. We propose the following efficient algorithms, which in our implementation takes a few seconds to execute:[16]

To efficiently compute $\mathcal{E}_{\text{err}}$, pool all the scores, w_t, with all the different thresholds, $-\operatorname{logit}\tilde{\pi}_i$, at which $\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi}_i)$ is to be evaluated. Sort them all together, keeping track of where the thresholds end up. A simple calculation involving the index of each threshold gives the desired miss and false-alarm rate at each threshold.

To efficiently compute $\mathcal{E}_{\text{err}}^{\text{min}}$, compute the vertices of the ROCCH, using the PAV algorithm (to be discussed below). There are typically very few of these vertices and as shown in section 7.3.6, the original large ROC can be replaced with these vertices, without changing the value of minDCF.

7.7 PAV calibration analysis

In the previous subsection, we solved the problem of calibration analysis by explicitly finding an optimal threshold at every operating point of interest. This works for applications that require binary accept/reject decisions and that therefore have a single score threshold per operating point.

A similar strategy would probably still work for applications that require ternary accept/reject/undecided decisions, involving optimization of two independent thresholds per operating point. But it certainly does not generalize to the case of continuous-valued (soft) decisions in any obvious way.

To make progress with such problems, we need to forget about discrete thresholds and think again in terms of *calibration transformations*, as discussed in section 5.3.1. If the evaluator can optimize the whole calibration function from score to log-likelihood-ratio, inside of a suitably constrained function space, then the evaluator can apply the optimized function to the scores and then evaluate the output as log-likelihood-ratios, with any desired proper scoring rule (including the strict ones which are not amenable to threshold optimization).

Here we need to: (i) define a suitably constrained function space, (ii) find an optimization algorithm that works in this space and (iii) interpret the result of the optimization. For two classes there are good solutions to all three problems: (i) The function space is naturally defined and has useful properties. (ii) An efficient optimization algorithm, called *PAV*, exists in the literature and (iii)

[16]The algorithms, coded in MATLAB, are freely available here: `http://focaltoolkit.googlepages.com`.

we show it gives a unique solution that is optimal for all regular binary proper scoring rules.

The idea to use PAV for a calibration transformation is due to [74]. The original PAV algorithm was published in 1955 [75].

7.7.1 Calibration function space

Our analysis of idealized scores in section 7.2 and the related discussion of the interpretation of minDCF in section 7.3.3 already suggest the definition of our function space in which calibration transformations should live, namely that they should be monotonic rising bijections. These criteria (i) preserve the *sense* of the scores (more positive favours target, more negative favours non-target) on both sides of the transformation and (ii) ensure that there is a one-to-one correspondence of thresholds on either side of the transformation.

Consider a transformation, $w_t = f(s_t)$, applied by the evaluee (the detector) to transform the raw scores, s_t, to calibrated log-likelihood-ratios, w_t, which are submitted to the evaluator. To judge the calibration of the w_t, the evaluator in turn applies a calibration transformation: $w'_t = g(w_t)$. The evaluator optimizes g to find out how good the w_t could have been, if the evaluee had submitted $w'_t = g(f(s_t))$, instead of $w_t = f(s_t)$.

Now if the family of calibration transformations is *closed under function composition*, then h, where $h(s_t) = g(f(s_t))$, would *also* be a calibration transformation. Then, provided f is *invertible*, by optimizing g, the evaluator is at the same time answering the question of how good the w_t could have been if the evaluee had instead submitted $w'_t = h(s_t)$. This is a simpler and perhaps more natural question. The proviso of the invertibility of f is so that there can be no information loss through f. If there could be information loss[17] through f, then optimizing g would in general not reach as good an optimum as would be possible by directly optimizing h.

We now have two requirements: closure and invertibility. A third requirement is inclusion of the *identity* transformation, to allow for the possibility that the submitted w_t are already optimally calibrated. Since function composition is *associative*, it is now clear that the problem is telling us that the set of calibration transformations really wants to be a *group*,[18] with function composition as the group operator [76].

Two technical details need to be taken care of. First, the condition of invertibility is not strong enough. We need all of the functions of the group to be *bijections*, so that inversions work in both directions. Second, in section 5.3.1, we required the family to include the *null* transformation which maps its input to 0. This is in conflict with the invertibility requirement, so

[17]Recall section 2.3.4.

[18]A *group* is a set that is *closed* under an *associative* binary operation, where every group element has an *inverse*. When the operation is applied to an element and its inverse, it gives the unique *identity* element in the group.

we have to exclude the null transformation from the group. But fortunately it ends up on the boundary, where it effectively still plays the required role. In fact, the set of invertible transformations is open and *all* the optima will be on the boundary, just outside the set.

Example: affine score calibration

Working with score transformations, we can define the group elements to be all affine transformations of the form $f(s) = \alpha s + \beta$, provided $\alpha > 0$. Every such function has an inverse and composition of two such functions is still an affine transform. The null transformation is on the boundary of this group, if we agree that these functions live in a metric space where by making $\alpha > 0$ very small, we can bring $f(s) = \alpha s$ and $g(s) = 0$ arbitrarily close to each other.

In the multi-class case, in chapter 8, we shall indeed limit ourselves to affine calibration transformations. For two-class, however, we consider a more general set of functions.

7.7.2 Non-parametric calibration transformation

For what follows, it will be convenient to work with calibration transformations between posterior probabilities. Once we have formulated the solution in posterior space, we will translate it back to log-likelihood-ratio space.

Let the detector be denoted by $\mathcal{R}$ and let $q_t = \mathcal{R}(x_t, \pi) = P(\theta_1|x_t, \pi, \mathcal{R})$ be the detector's target posterior for trial t. Let $f : [0,1] \mapsto [0,1]$ be the calibration transform and let $\mathcal{R}_f$ denote the recalibrated version of the original detector, so that $p_t = \mathcal{R}_f(x_t, \pi) = f\big(\mathcal{R}(x_t, \pi)\big) = f(q_t)$.

Our evaluation criterion, defined by plugging any regular binary proper scoring rule, C^*_ρ, into (5.4), is denoted as $\mathcal{E}_\rho(\mathcal{R}_f|\pi)$.

The problem at hand, namely to choose a group of calibration transformations and to optimize our evaluation criterion inside this group, is simplified by realizing we need only a finite number of points on the function. If we have T input scores, $q_1, \dots, q_T$, and the transformation is $p_t = f(q_t)$, then all we really need, in order to compute $\mathcal{E}_\rho(\mathcal{R}_f|\pi)$, are the T values $p_1, \dots, p_T$.

For the reasons given above, we now constrain f to be strictly monotonic rising. The input scores q_t can be sorted by the evaluator, without changing the problem (provided we remember to keep track of the labels). We can therefore assume, without loss of generality, that $q_1 < q_2 < \cdots < q_T$. Now the strict monotonicity requires also $p_1 < p_2 < \cdots < p_T$. As explained above, this restriction forms an open set and for a well-defined optimization, we need a closed set, so we optimize instead within the closure given by the requirement:

$$0 \leq p_1 \leq p_2 \leq \cdots \leq p_T \leq 1 \tag{7.45}$$

where we have added the boundaries 0 and 1 to make it clear the p_t are probabilities. For strict (i.e. smooth) proper scoring rules, the criterion, $\mathcal{E}_\rho(\mathcal{R}_f|\pi)$, on the boundary will be arbitrarily close to that of some nearby solution in the interior of the set.

This monotonicity constraint will now serve as the definition of our family of allowed calibration transformations: any sequence, $p_1, p_2, \dots, p_T$, that satisfies (7.45) is allowed as the output of the calibration transformation.

7.7.3 The PAV solution

The evaluator's problem in optimizing the calibration transformation can now be stated as follows. There is given:

- a regular binary proper scoring rule, $C^*_\rho(q|\theta)$,
- a prior, $(\pi_1, \pi_2) = (\pi, 1-\pi)$, where $0 < \pi < 1$,
- and (after sorting) a sequence of trial indices, $t = 1, 2, \dots, T$, which is partitioned into the target subset, $\mathcal{T}_1$, and the non-target subset, $\mathcal{T}_2$.

The evaluator has to minimize the evaluation criterion, $\mathcal{E}_\rho(\mathcal{R}_f|\pi)$, to find:

$$\mathcal{E}^{\min}_\rho(\mathcal{R}|\pi) = \min_{p_1,\dots,p_T} \sum_{i=1}^{2} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C^*_\rho(p_t|\theta_i) \tag{7.46}$$

subject to (7.45). This optimization problem is discussed in detail in appendix C, where we give proofs and references to the relevant literature. Remarkably, the problem has a unique solution, *independent* of the proper scoring rule, which can be expressed in closed form as:

$$p^*_t = \max_{1\le i\le t} \min_{t\le j\le T} r(i,j) = \min_{t\le j\le T} \max_{1\le i\le t} r(i,j) \tag{7.47}$$

$$r(i,j) = \frac{\pi b_1(i,j)}{\pi b_1(i,j) + (1-\pi) b_2(i,j)} \tag{7.48}$$

$$b_k(i,j) = \frac{n_k(i,j)}{n_k(1,T)} \tag{7.49}$$

where $n_1(i,j)$ is the number of target trials, and $n_2(i,j)$ is the number of non-target trials in the subsequence indexed by t', such that $i \le t' \le j$.

As shown[19] in appendix C, if we translate this optimization problem to the log-likelihood-ratio domain, then the log-likelihood-ratio solution is also independent of the prior:

$$w^*_t = \operatorname{logit} p^*_t - \operatorname{logit} \pi = \log \ell^*_t \tag{7.50}$$

[19] Another way to see this is simply that (7.48) processes the prior in accordance with Bayes' rule.

where

$$\ell_t^* = \log \frac{b_1(i_t, j_t)}{b_2(i_t, j_t)} \tag{7.51}$$

and where i_t, j_t are the indices as found in the minimax problem (7.47). For $\mathcal{R}(x, \boldsymbol{\pi}) = \mathcal{B}(\mathcal{W}(x), \boldsymbol{\pi})$, we find:

$$\mathcal{E}_\rho^{\min}(\mathcal{W}|\boldsymbol{\pi}) = \mathcal{E}_\rho^{\min}(\mathcal{R}|\boldsymbol{\pi})\,. \tag{7.52}$$

This works, because of the correspondence (7.50) and because the monotonicity constraint (7.45) can be equivalently stated in terms of sorting the input log-likelihood-ratios, rather than sorting the input posteriors as in our explanation above. Some comments about this solution are in order:

- The optimum log-likelihood-ratio solution, w_t^*, is independent of the prior and of the proper scoring rule. The posterior solution, p_t^*, is dependent on the prior, but independent of the scoring rule. The value of the minimized evaluation criterion, $\mathcal{E}_\rho^{\min}(\mathcal{R}|\boldsymbol{\pi})$, is dependent on the prior and the proper scoring rule C_ρ^*.

- The independence of w_t^* of the proper scoring rule and the prior is a rather special property. If, for example, we use a more constrained set of calibration functions, such as the set of affine transforms, $w_t^* = \alpha s_t + \beta$, of the example above, then the optimizing solution is dependent on *both* the proper scoring rule and the prior.

- The solution partitions the sequence into a number of blocks, each of which has a constant solution, typically spanning several trials. This corresponds to flat regions in the calibration function, making it non-invertible, so that it is on the boundary of the set of invertible calibration functions.

- Ironically, this solution, which by (7.49) effectively counts labels in score bins, is very close to the histogram solution of [35], which we mentioned during our discussion of the calibration/refinement decomposition in section 5.3. The difference is that here, the number of bins and the size of each bin is adaptively determined by optimizing a well-defined criterion, rather than just arbitrarily choosing some number of fixed-width bins.

- The solution is dependent only on the *order* of the input scores. After sorting the input scores, it is only the sequence of target and non-target labels that matters, not the values of the scores. (This is also true of ROC, EER and minDCF.) This means it does not matter whether the input scores are uncalibrated scores, likelihood-ratios, log-likelihood-ratios or probabilities. This solution works for all sortable scores, with the sense that larger scores favour the target and smaller scores favour the non-target.

- It is shown in [69] that the slopes of the line-segments of the ROCCH are given by $-\ell_t^*$, where $i_t, \dots, j_t$ are the indices of the scores spanned by that line-segment. Compare this to section 7.2.2, where the slope of the ideal continuous ROC also gives the likelihood-ratio.
- We shall refer to this solution as the *PAV solution*, because it is efficiently computed with the *PAV algorithm*.

Computation: the PAV algorithm

Notice that the computational complexity of a straight-forward solution to (7.47) increases as T^2. Fortunately the *pool adjacent violators* (PAV) algorithm solves this problem with linear complexity. See appendix C for details and references. Note, however that the PAV algorithm must be preceded by sorting, which typically has complexity of order $T\log(T)$. In our implementation, sorting and applying PAV takes a few seconds for a few million scores.

7.7.4 Applications of PAV calibration analysis

We can use (7.46) to compute $\mathcal{E}_\rho^{\min}(\mathcal{W}|\boldsymbol{\pi})$, which is a calibration-insensitive version of any evaluation criterion, $\mathcal{E}_\rho(\mathcal{W}|\boldsymbol{\pi})$. The contrast between $\mathcal{E}_\rho$ and $\mathcal{E}_\rho^{\min}$ can be used to evaluate quality of calibration, or $\mathcal{E}_\rho^{\min}$ can be used on its own, when calibration is not of immediate interest. In the author's experience thus far, the main interest has been in $\mathcal{E}_{\text{err}}^{\min}$ and $\mathcal{E}_{\log}^{\min}$:

- $\mathcal{E}_{\text{err}}$ and $\mathcal{E}_{\text{err}}^{\min}$ form the normalized Bayes error-rate plot, which as described above, proved useful for preparation and post-analysis in NIST SRE 2010. They also appear in APE-curves, described below.
- $L_{\text{cal}} = \mathcal{E}_{\log} - \mathcal{E}_{\log}^{\min}$ forms a scalar, application-spanning, summary criterion of the goodness of calibration. We used this criterion extensively for SREs 2005, 2006 and 2008.

APE curves

In our journal paper [17], we used all four of the above criteria together in what we called *applied probability of error*, or APE, plots. An APE plot includes the following:

- Unnormalized curves of $\mathcal{E}_{\text{err}}(\mathcal{W}|\tilde{\pi})$ and $\mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\tilde{\boldsymbol{\pi}})$ against the horizontal axis, $h = \operatorname{logit}\tilde{\pi}$.
- The default reference curve $\mathcal{E}_{\log}(\mathcal{W}_0|\tilde{\boldsymbol{\pi}})$.
- $L_{\text{dis}} = \mathcal{E}_{\log}^{\min}(\mathcal{W}|0.5)$, called *discrimination loss*, and $L_{\text{cal}} = \mathcal{E}_{\log}(\mathcal{W}|0.5) - \mathcal{E}_{\log}^{\min}(\mathcal{W}|0.5)$, called *calibration loss* are plotted as a single, stacked bar

graph, so that the total height is the *total loss*, $L_{\text{tot}} = L_{\text{dis}} + L_{\text{cal}} = \mathcal{E}_{\log}(\mathcal{W}|0.5)$. The logarithmic criterion is not varied as a function of the prior, since it is already an integral over the prior.

The APE-plot has the following, graphically interpretable properties:

- $L_{\text{tot}} = \int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\operatorname{logit}^{-1} h)\, dh$.
- $L_{\text{dis}} = \int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\operatorname{logit}^{-1} h)\, dh$.
- $L_{\text{cal}} = \int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\operatorname{logit}^{-1} h) - \mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\operatorname{logit}^{-1} h)\, dh$.
- $\text{ROCCH-EER} = \max_h \mathcal{E}_{\text{err}}^{\min}(\mathcal{W}|\operatorname{logit}^{-1} h)$.

See our book chapter [18] for more on APE-plots. We have used APE plots from 2005 to the present as a standard tool for evaluating calibration of many speaker recognition systems, including those we submitted to NIST SREs in 2005, 2006 and 2008.

ECE curves

Daniel Ramos introduced *empirical-cross-entropy* (ECE) curves in his Ph.D. dissertation [29], as an alternative to APE curves, more suitable[20] for the forensic interpretation of speaker recognition evidence. In these curves, $\mathcal{E}_{\log}(\mathcal{W}|\tilde{\boldsymbol{\pi}})$ and $\mathcal{E}_{\log}^{\min}(\mathcal{W}|\tilde{\boldsymbol{\pi}})$ are plotted against $\operatorname{logit}\tilde{\pi}$. The canonical Shannon entropy, induced by the logarithmic cost function, lends a standard information-theoretic interpretation of the quantities on this graph, as the effective information delivered by the detector to the decision-maker.

(As an irrelevant[21] aside: The quantity $\max_{\tilde{\pi}} \mathcal{E}_{\log}(\mathcal{W}_0|\tilde{\pi}) - \mathcal{E}_{\log}^{\min}(\mathcal{W}|\tilde{\pi})$ has an interesting communications engineering interpretation. $\mathcal{E}_{\log}(\mathcal{W}_0|\tilde{\pi}) - \mathcal{E}_{\log}^{\min}(\mathcal{W}|\tilde{\pi})$ behaves similarly to mutual information between the detector output and the proposition. Maximizing mutual information over the prior is the *channel capacity* [27].)

7.8 Summary

In this chapter we applied the theoretical first part of this work for analysing traditional speaker detection evaluation criteria and to generalize them to new criteria with complementary properties. Table 7.1 gives a comparison of properties.

[20]When the legal professions are involved, one cannot use the word *error* as freely as one does in a pure engineering environment. This disqualifies $\mathcal{E}_{\text{err}}$ in favour of $\mathcal{E}_{\log}$, where the errors are hidden from view by integrating over them.

[21]The footnote above may be irreverent, but not irrelevant.

Traditional evaluation criteria are either calibration-sensitive or application-spanning. The new criteria presented here do both, while still providing auxiliary application-insensitive versions for comparative calibration analysis.

The new criteria span applications in one of two ways, either by plotting the criterion over a range of applications, or by creating a linear additive summary via analytical integration over applications. Both are useful for evaluation, while the latter is useful for discriminative training.

This chapter treated two-class recognition in great detail. The next chapter treats multi-class in lesser detail, because for multi-class most of the two-class methods don't work, are not useful, are too complicated, or remain to be discovered. Fortunately, as we shall show, the logarithmic cost function provides most of what we need for practical development and testing of application-spanning multi-class recognizers.

Chapter 8

Solutions for multi-class recognition

In this chapter, we use language recognition as a prototype for multi-class recognizers that provide their output in cost-and-prior-independent log-likelihood form.

We are interested in the pattern recognition problem where an input, x, is given, for which it is known that *exactly one* of a set of $N \geq 2$ propositions, $\boldsymbol{\Theta}_N = \{\theta_1, \dots, \theta_N\}$, is true and where the recognizer, $\mathcal{W}$, outputs a vector of log-likelihoods $\acute{\mathbf{w}} = \mathcal{W}(x)$, as discussed in section 4.2.1.

In this chapter we consider both evaluation solutions and discriminative training solutions. Except for logarithmic cost, our evaluation solutions for multi-class are *not* generalizations of our two-class solutions. The discriminative training solutions are however of the same form for $N = 2$ and for $N > 2$ and for that reason are convenient to treat as one topic in this chapter.

As in chapter 7, the evaluation methods of the NIST Language Recognition Evaluations provide a valuable starting point for our discussion.

8.1 LRE's CAVG as discrete summary criterion

Here we analyse the evaluation criterion of the primary task in the NIST Language Recognition Evaluations (LREs) of 2005, 2007 and 2009[1], in terms of our notation.

The primary evaluation task involves $N > 2$ language classes. The input, x_t, for each trial, t, is a single speech segment, known to contain speech in exactly one of these classes. NIST's primary evaluation criterion, called[2] CAVG, is formed by a *mixture* of N different applications, as explained in section 6.2.

[1]See the *evaluation plans* here: `www.itl.nist.gov/iad/mig/tests/lre`.

[2]NIST renders CAVG as C_{avg}, but this is confusable with our notation for cost functions, which evaluate trials. CAVG evaluates *all* the trials in an evaluation database, just like DCF.

The mixture components are detection applications, where accept/reject decisions are required.

The language recognizer is required to output a *compound*, N-component decision, $\mathbf{a}_t = (a_{1t}, \dots, a_{Nt}) \in \{\text{accept}, \text{reject}\}^N$, for each trial, t. The compound decision is evaluated with an N-fold combination of the criteria formed by N different applications. The prior and the cost function are *both* varied across applications. For application k, class θ_k is denoted the *target* class, and the prior, $\boldsymbol{\pi}_k = (\pi_{1k}, \pi_{2k}, \dots, \pi_{Nk})$, is

$$\pi_{ik} = \frac{\delta_{ik}}{2} + \frac{1 - \delta_{ik}}{2(N-1)} \tag{8.1}$$

where δ_{ik} denotes Kronecker delta. The target class gets half of the probability mass and the rest, the non-targets, share the other half. The cost function for application k, which evaluates each decision, a_{kt}, is

$$C_k(a|\theta_i) = \begin{cases} 1 - \delta_{ik} & \text{if } a = \text{accept}\,, \\ \delta_{ik} & \text{if } a = \text{reject} \end{cases} \tag{8.2}$$

so that detection errors, when θ_k is the target, are penalized with zero-one cost. NIST's evaluation criterion, CAVG, is now assembled as the *average* over the N applications:

$$\text{CAVG} = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{N} \frac{\pi_{ik}}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} C_k(a_{kt}|\theta_i)\,. \tag{8.3}$$

As we showed in section 6.2, such a mixture of applications can be rewritten as a single application, by absorbing the priors into the cost functions and then forming a new combined cost function:

$$\begin{aligned} \text{CAVG} &= \sum_{i=1}^{N} \frac{\tilde{\pi}_i}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} \sum_{k=1}^{N} \tilde{C}_k(a_{kt}|\theta_i) && (8.4) \\ &= \sum_{i=1}^{N} \frac{\tilde{\pi}_i}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} C_{\text{lre}}(\mathbf{a}_t|\theta_i) && (8.5) \end{aligned}$$

where $\tilde{\pi}_i = \bar{\pi} = \frac{1}{N}$ is now constant and

$$C_{\text{lre}}(\mathbf{a}_t|\theta_i) = \sum_{k=1}^{N} \tilde{C}_k(a_{kt}|\theta_i) \tag{8.6}$$

where

$$\begin{aligned}\tilde{C}_k(a|\theta_i) &= \pi_{ik} C_k(a|\theta_i) \\ &= \left(\frac{\delta_{ik}}{2} + \frac{1-\delta_{ik}}{2(N-1)}\right)\begin{cases}1-\delta_{ik} & \text{if } a = \text{accept}\,, \\ \delta_{ik} & \text{if } a = \text{reject}\end{cases} \\ &= \frac{1}{2}\begin{cases}\frac{1-\delta_{ik}}{N-1} & \text{if } a = \text{accept}\,, \\ \delta_{ik} & \text{if } a = \text{reject}\,.\end{cases}\end{aligned} \tag{8.7}$$

We have thus shown that the criterion CAVG can be equivalently parametrized with the single application $(C_{\text{lre}}, \bar{\boldsymbol{\pi}})$. The cost function $C_{\text{lre}}(\mathbf{a}|\theta)$ evaluates a *vector*, $\mathbf{a}$, of accept/reject decisions. Since the decision vector has binary values, $\mathbf{a}$ can also be interpreted as a subset of $\boldsymbol{\Theta}_N$. The decision space is therefore just the power set: $\mathcal{A} = 2^{\boldsymbol{\Theta}_N}$.

In summary, unlike NIST's speaker detection criterion, DCF, which represents a single operating point, the language recognition criterion, CAVG, represents a discrete mixture of operating points, which exercises the decision-making ability of the recognizer's output more thoroughly.

8.1.1 CAVG as proper scoring rule

We now follow the same process as for speaker detection, where we generalized DCF to $\mathcal{E}_{\text{err}}$, by requiring log-likelihoods from the recognizer, rather than hard decisions. We convert CAVG to $\mathcal{E}_{\text{lre}}$ by replacing each $\tilde{C}_k$ by the proper scoring rule $\tilde{C}_k^*$.

The expected-cost, $\tilde{C}_k(a|\mathbf{q})$, is minimized by the Bayes decision, to accept if and only if $q_k \geq \bar{\pi}$. Notice that for a given $\mathbf{q} = (q_1, \ldots, q_N)$, more than one component may exceed the threshold, so that the compound decision $\mathbf{a} = (a_1, \ldots, a_N)$ may contain *more than one* accept decision, even though it is known that exactly one proposition must be true for trial t. This is not an anomaly, because the cost function for each application k is different.

The individual proper scoring rules are now:

$$\tilde{C}_k^*(\mathbf{q}|\theta_i) = \frac{1}{2}\left(u(q_k - \bar{\pi})\frac{1-\delta_{ik}}{N-1} + (1 - u(q_k - \bar{\pi}))\delta_{ik}\right) \tag{8.8}$$

where u denotes the unit step function. The combination is the proper scoring rule:

$$C_{\text{lre}}^* = \sum_{k=1}^{N} \tilde{C}_k^*(\mathbf{q}|\theta_i) \tag{8.9}$$

$$= \frac{1 - u(q_i - \bar{\pi})}{2} + \frac{\sum_{k\neq i} u(q_k - \bar{\pi})}{2(N-1)}\,. \tag{8.10}$$

Notice that the first term gives a penalty of $\frac{1}{2}$ if the true class θ_i is missed, when $q_i < \bar{\pi}$. The second term gives an additional, smaller penalty of $\frac{1}{2(N-1)}$ for each false-accept, when $k \neq i$ and $q_k \geq \bar{\pi}$. Since the decisions are *sets* of classes, multiple penalties can be incurred at once.

The minimum score is 0, for a region (polytope) of $\mathbf{q}$ including the vertex at $q_i = 1$. The maximum score is 1, for the opposed region including the point where $q_i = 0$ and all the other $q_k = \frac{1}{N-1} > \frac{1}{N}$. In this region all possible detection errors happen, the target θ_i is missed and every one of the non-targets is falsely accepted. The decision regions for the case $N = 3$ are illustrated in figure 8.1.

The evaluation recipe is completed by requiring the recognizer to output a log-likelihood vector, $\acute{\mathbf{w}}_t = \mathcal{W}(x_t)$, for every trial t, to form the evaluation criterion:

$$\mathcal{E}_{\text{lre}}(\mathcal{W}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C^*_{\text{lre}}\left(\mathcal{B}(\acute{\mathbf{w}}_t|\boldsymbol{\pi})\right) . \tag{8.11}$$

Some comments are in order:

- If the hard decisions submitted to CAVG are made with Bayes decisions using the $\acute{\mathbf{w}}_t$, then CAVG $= \mathcal{E}_{\text{lre}}(\mathcal{W}|\bar{\boldsymbol{\pi}})$, where $\bar{\boldsymbol{\pi}}$ is the uniform prior.

- CAVG and $\mathcal{E}_{\text{lre}}$ can be interpreted as the average error-rate over the mixture of detection applications, and are bounded between 0 and 1.

- In principle we could do the same as for speaker detection, to perform a more exhaustive evaluation by sweeping the prior, $\boldsymbol{\pi}$, over a range of values and recording all the values of $\mathcal{E}_{\text{lre}}(\mathcal{W}|\boldsymbol{\pi})$. However, for a continuous variation of the prior, that would be difficult to plot, since $\boldsymbol{\pi}$ is multidimensional. Instead, in section 8.3.3 we show an example of a plot representing thousands of discrete points in prior space, each evaluated by $\mathcal{E}_{\text{lre}}(\mathcal{W}|\boldsymbol{\pi})$.

8.2 Logarithmic cost as continuous summary criterion

Since C^*_{lre} is defined via hard decisions, it is a non-strict scoring rule. Here we examine a strict proper scoring rule for multi-class, namely logarithmic cost, $C_{\log}$.

We will show that, like in the two-class case, multi-class $C_{\log}$ can also be interpreted as a continuous combination over a parametrized family of simple hard-decision applications. However, this simple 'building block' will not be the above-discussed LRE application $(C_{\text{lre}}, \bar{\boldsymbol{\pi}})$. Instead, we generalize our two-class building block, C_η, to a multi-class version, $C_{\boldsymbol{\eta}}$.

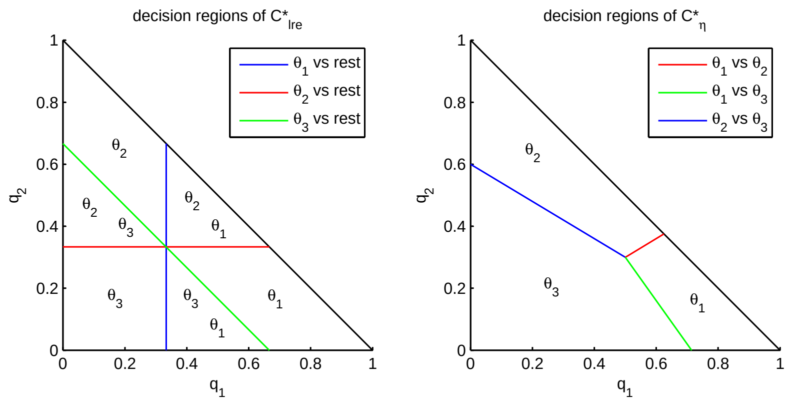

Figure 8.1: Comparison of the decision regions in the simplex $\mathbf{q} = (q_1, q_2, q_3)$, as respectively induced by the proper scoring rules C^*_{lre}, on the left, and $C^*_{\boldsymbol{\eta}}$, with $\boldsymbol{\eta} = (0.5, 0.3, 0.2)$, on the right.

In the two-class version, the Bayes decision threshold is just the *point* η on the one dimensional simplex, or line-segment $[0, 1]$. In multi-class, we have a multidimensional simplex and thresholding becomes more complex—compare the two-class threshold to the tessellations of the simplex in figure 8.1.

We now define $C_{\boldsymbol{\eta}} : \boldsymbol{\Theta}_N^2 \mapsto \mathbb{R}$ as:

$$[C_{\boldsymbol{\eta}}(\theta_j|\theta_i)] = \frac{1}{N-1} \begin{bmatrix} 0 & \frac{1}{\eta_1} & \cdots & \frac{1}{\eta_1} \\ \frac{1}{\eta_2} & 0 & \cdots & \frac{1}{\eta_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\eta_N} & \frac{1}{\eta_N} & \cdots & 0 \end{bmatrix}, \text{ where } \boldsymbol{\eta} = (\eta_1, \ldots, \eta_N) \in \mathbb{P}_N^o \quad (8.12)$$

for row i and column j of the matrix.[3] The decision, θ_j, when the true proposition is θ_i, is evaluated as $C_{\boldsymbol{\eta}}(\theta_j|\theta_i)$. The cost of a correct decision is 0 and the cost of *missing* class θ_i is $\frac{1}{\eta_i}$, regardless of which class, θ_j, was falsely recognized. As before, the costs are parametrized by the vector $\boldsymbol{\eta}$, which is confined to the *interior*, $\mathbb{P}_N^o$, of the probability simplex $\mathbb{P}_N$. In other words, $\sum_{i=1}^N \eta_i = 1$ and $0 < \eta_i < 1$.

As in the case of two classes, the penalties can grow arbitrarily large, but this cost function is normalized in the sense that the expected value of $C_{\boldsymbol{\eta}}$ w.r.t. $\boldsymbol{\eta}$ is $C_{\boldsymbol{\eta}}(\theta_j|\boldsymbol{\eta}) = \frac{1}{N-1}$, for every decision θ_j.

The decision regions for the associated proper scoring rule $C^*_{\boldsymbol{\eta}}$ are compared to those of C^*_{lre} in figure 8.1, for the case $N = 3$. Notice that the simple

[3]We could alternatively have derived CAVG by letting the k-th cost function also be of the form $\boldsymbol{\Theta}_N^2 \to \mathbb{R}$, by letting $C_k(\theta_j|\theta_i) = (\delta_{ik} - \delta_{jk})^2$. This would still have required compound decisions.

decisions required by $C_{\boldsymbol{\eta}}$ lead to a simpler tessellation than the compound decisions required by C_{lre}. Comparing the proper scoring rules directly, $C^*_{\text{lre}}(\mathbf{q}|\theta)$ exercises the decision-making ability of $\mathbf{q}$ more thoroughly than $C^*_{\boldsymbol{\eta}}(\mathbf{q}|\theta)$. But the latter is just a building block for an exhaustive exercising of $\mathbf{q}$, effected by varying the parameter $\boldsymbol{\eta}$.

To see this, note that $\boldsymbol{\eta}$ is the point in the simplex where the decision boundaries meet. This is shown in the right-hand part of figure 8.1, where the boundaries are line-segments. In the general case, the boundaries will be hyperplane-segments. If we vary $\boldsymbol{\eta}$ over the whole simplex where $\mathbf{q}$ lives, we will be involving all possible values of $\mathbf{q}$ in making decisions. This generalizes the sweeping of the scalar threshold, which gave us the two-class result, $\int_0^1 C^*_\eta(q|\theta)\,d\eta = C_{\log}(q|\theta)$. This also works for multi-class:

$$\int_{\mathbb{P}_N} \rho(\boldsymbol{\eta}) C^*_{\boldsymbol{\eta}}(\mathbf{q}|\theta_i)\,\mathbf{d}\boldsymbol{\eta} = -\log(q_i) = C_{\log}(\mathbf{q}|\theta_i) \tag{8.13}$$

if $\rho(\boldsymbol{\eta}) = \Gamma(N)$ is the uniform distribution[4] over the simplex.

Proof. The proof is in appendix D.

Several comments follow:

- For the two-class case, proper scoring rules formed by integrals similar to (7.34) have been studied at least since 1966, see [77]. However, we are unaware of any such formulations for the multi-class case.[5] In particular, our result (8.13) appears to be a new interpretation of the logarithmic cost function.

- As in the two-class case, generalizing (8.13) by choosing different distributions for $\rho(\boldsymbol{\eta})$ will form a large family of multi-class proper scoring rules, because of closure under combination. Different parametrizations of the Dirichlet distribution may give similar results to those of section 7.4.2. We leave this for future work.

- It is not clear whether or not (8.13), with general $\rho(\boldsymbol{\eta})$, could characterize any multi-class proper scoring rule, as (7.34) does for two-class. For example, we are not aware of a choice for $\rho(\boldsymbol{\eta})$ that would induce C^*_{lre}. We therefore cannot repeat the claim we made for two-class: we cannot claim that the multi-class logarithmic cost summarizes all multi-class Bayes decision applications. For now we have to be content with

[4]Here $\Gamma(N) = 2 \times 3 \times \cdots \times (N-1)$ is the gamma function. To see that this is the correct normalization factor for the uniform distribution over the simplex, note that the uniform distribution is a special case of the Dirichlet distribution [1].

[5]There is however a general categorization of multi-class proper scoring rules formed by generalized differentiation (subgradients) of convex (negative generalized entropy) functions, see [8], theorem 2. This implies a corresponding integral characterization, which however seems not to have been explicitly developed in the literature.

our interpretation above that shows that the logarithmic cost exercises the whole probability simplex by sweeping the boundary intersection everywhere.

- As in the two-class case, the multi-class logarithmic cost function is unbounded,[6] by virtue of including applications where the cost of errors are arbitrarily high, when any $\eta_i \approx 0$. For the same reasons as before, we prefer an unbounded proper scoring rule, both as a representative of real-world applications and as a discriminative training criterion with optimizer-friendly qualities.

- For $N > 2$, the logarithmic scoring rule has been shown [78] to be the *only* strictly proper scoring rule with the property that when θ_i is true, then its value depends only on q_i and not on any of the other probabilities. We argue this is appropriate in applications where there is no concept of distance between the classes. In such cases, the probability distribution over the other classes is unimportant. As a counter-example, consider a multi-class age recognition application, where age is binned into a number of age classes. Here adjacent bins can be regarded as 'close' and a recognizer that outputs a low probability for the true bin, but a high probability for an adjacent bin should be less harshly penalized than one that assigns low probabilities also for the adjacent bins. See for example [79].

- For the two-class case, we found an alternative representation of the logarithmic evaluation criterion, by integrating error-rate as a function of the prior log odds: $\int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\operatorname{logit}^{-1} h)\, dh = \mathcal{E}_{\log}(\mathcal{W}|\bar{\pi})$. To generalize this to the multi-class case, one needs a non-uniform weighting over a multidimensional integral. This is briefly demonstrated (with a nice graphic) in appendix D.1.

8.2.1 Logarithmic evaluation criterion

The evaluation recipe is completed by requiring the recognizer to output a log-likelihood vector, $\acute{\mathbf{w}}_t = \mathcal{W}(x_t)$, for every trial t, to form the evaluation criterion:

$$\mathcal{E}_{\log}(\mathcal{W}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C_{\log}\big(\mathcal{B}(\acute{\mathbf{w}}_t|\boldsymbol{\pi})\big|\theta_i\big)\,. \tag{8.14}$$

When $N = 2$, this is equal to the two-class version (7.40) and it has similar properties. In our work in language recognition, we used $\mathcal{E}_{\log}(\mathcal{W}|\bar{\boldsymbol{\pi}})$, with the uniform prior, as an optimization-friendly surrogate for $\mathcal{E}_{\text{lre}}(\mathcal{W}|\bar{\boldsymbol{\pi}})$.

[6]The well-known multi-class version of the Brier score, $C_{\text{Brier}}(\mathbf{q}|\theta_i) = 1 - 2q_i + \sum_{j=1}^{N} q_j^2$, is an example of a bounded proper scoring rule.

8.3 Discriminative training

Here we describe how we use the logarithmic proper scoring rule to form an objective function for discriminative training of two-class and multi-class pattern recognizers.

In [1] chapter 1, Bishop distinguishes between two flavours of discriminative pattern recognition, namely *discriminative models*, which output posterior distributions and *discriminant functions*, which directly output decisions. Logistic regression is an example of discriminative modelling, while support vector (SVM) classifiers are discriminant functions. Since the recognizers discussed in our work output posteriors (or equivalently likelihoods), they qualify as discriminative models.

To discriminatively train a model, one needs an objective function that maps the model parameter set to a real scalar. Depending on the sense of the objective, it must be either minimized or maximized to discriminatively train the parameters. Our empirical evaluation criterion, $\mathcal{E}_{\boldsymbol{\mathcal{T}}}(\mathcal{W}|\boldsymbol{\pi})$, when computed on a *supervised training database*, can fulfil the role of discriminative training objective, for certain proper scoring rules, $C^*_{\boldsymbol{\mathcal{T}}}$.

For the two-class case, the PAV solution of section 7.7.3 can be viewed as discriminative training of the non-parametric calibration transformation. In this special case, it does not matter which proper scoring rule is used, since PAV optimizes all of them. In the parametric case which we consider here, it does matter.

Since we defined proper scoring rules with the sense of cost rather than utility, the objective, $\mathcal{E}_{\boldsymbol{\mathcal{T}}}(\mathcal{W}|\boldsymbol{\pi})$, must be minimized. Non-strict proper scoring rules give non-differentiable objective functions and are therefore not optimization-friendly. Even some strict proper scoring rules, such as the quadratic Brier score, which give differentiable objectives, tend to give non-convex objective functions, which are still difficult to optimize. The unbounded cost functions, at least for models with log-likelihoods that are linear functions of the parameters, give convex, optimization-friendly [80] objective functions [13].

We start our discussion by letting the discriminatively trained recognizer output be in posterior form. We switch to log-likelihood form later. Let $\mathcal{R}_{\boldsymbol{\lambda}}$ represent a recognizer with adjustable parameter set, $\boldsymbol{\lambda}$. The recognizer output for every trial, t, in a supervised training database is $\mathbf{q}_t = \mathcal{R}_{\boldsymbol{\lambda}}(x_t, \boldsymbol{\pi})$. The evaluation criterion is now the discriminative training objective function and is defined/parametrized by $(C^*_{\boldsymbol{\mathcal{T}}}, \boldsymbol{\pi})$, where $C^*_{\boldsymbol{\mathcal{T}}}$ is some proper scoring rule. The objective function is:

$$\mathcal{E}_{\boldsymbol{\mathcal{T}}}(\mathcal{R}_{\boldsymbol{\lambda}}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C^*_{\boldsymbol{\mathcal{T}}}\bigl(\mathcal{R}_{\boldsymbol{\lambda}}(x_t, \boldsymbol{\pi})\big|\theta_i\bigr) \,. \tag{8.15}$$

Discriminative training is the minimization of $\mathcal{E}_{\boldsymbol{\mathcal{T}}}(\mathcal{R}_{\boldsymbol{\lambda}}|\boldsymbol{\pi})$ w.r.t. $\boldsymbol{\lambda}$. Notice that

$\boldsymbol{\pi}$ plays two roles: the same prior is used by both recognizer and evaluator. In this respect, our formulation of discriminative training is different from the usual formulation (see e.g. [1] chapter 4), where the recognizer's prior is included in $\boldsymbol{\lambda}$. The latter may be more appropriate for some tasks such as speech recognition, where learning the prior (over phones for example) is part of the training task. In our context, however, we regard the prior as variable and not part of the recognizer.

Our proper scoring rule of choice for discriminative training is $C_{\log}$, which works for both two-class and for multi-class. The objective, $\mathcal{E}_{\log}(\mathcal{R}_{\boldsymbol{\lambda}}|\boldsymbol{\pi})$, is very well known in several fields as a discriminative training objective:

- If the model log-likelihoods are linear functions of the parameters, then the result is logistic regression, see e.g. [1] chapter 4. We elaborate on this below.
- Since the logarithmic cost function induces the canonical Shannon entropy, discriminative training based on it is often referred to as maximum-mutual-information (MMI), see e.g. [12].
- The sum of logarithms of posteriors is also the logarithm of the product of posteriors. In the special case where the prior, $\boldsymbol{\pi}$, is chosen to reflect the proportions of classes in the training database, and under appropriate conditional independence assumptions, minimizing $\mathcal{E}_{\log}$ is equivalent to maximizing the posterior probability of all the class labels, given all of the input data [9, 10].

8.3.1 Logistic regression for fusion and calibration

Thus far, our efforts at large-scale discriminative training for both speaker recognition [54] and language recognition [23] have not been able to convincingly outperform state-of-the-art generative models. However, in both of these fields, discriminative training of a small number of parameters involved in fusing and recalibrating the outputs of existing recognizers has been very successful.

Affine fusion and calibration

The following recipe works for two-class and for multi-class. Let there be K recognizers, denoted $\mathcal{W}_k$, for $k = 1, \ldots, K$. They are all designed to recognize the same N classes and have outputs in the same log-likelihood format. In what follows it is unimportant whether these outputs are in the log-likelihood line format, $\acute{\mathbf{w}}$, of section 4.2.1, or whether they are Euclidean vectors, $\mathbf{w} \in \acute{\mathbf{w}}$. In the two-class case they may also be scalar log-likelihood-ratios, w, where $(w, 0) \in \acute{\mathbf{w}}$. For all of these formats, addition and scalar multiplication have the appropriate meaning. For generality, we formulate our equations in log-likelihood-line format. We form a new recognizer, $\mathcal{W}_{\boldsymbol{\lambda}}$, with parameter set $\boldsymbol{\lambda} = (\alpha_1, \alpha_2, \dots, \alpha_N, \acute{\mathbf{b}})$, where the α_k are real, scalar weights and $\mathbf{b} \in \mathbb{L}_N$ is a log-likelihood line of the same dimension as the output of the recognizer. For a given input x_t, simultaneously given to each existing recognizer, the output of the new recognizer is formed as:

$$\mathcal{W}_{\boldsymbol{\lambda}}(x_t) = \acute{\mathbf{b}} + \sum_{k=1}^{K} \alpha_k \mathcal{W}_k(x_t)\,. \tag{8.16}$$

The weighted sum over recognizers forms what is known as a *fusion* or *combination* of recognizers. By adding the offset, $\acute{\mathbf{b}}$, we are also simultaneously performing an affine calibration transformation on the fusion output. (There is no additional calibration scaling factor required for the fusion output, because this scaling is implicit in the fusion weights, α_k. In our work, we do not constrain the fusion weights to be positive and indeed we have found good solutions which included negative weights [21]. Note that (8.16) is still useful for the case of one system, when $K = 1$. In this case it is not a fusion, but just an affine log-likelihood calibration.

Linear model and logarithmic cost gives logistic regression

Discriminative training of (8.16) is performed by minimizing the objective,

$$\mathcal{E}_{\log}(\mathcal{W}_{\boldsymbol{\lambda}}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \frac{\pi_i}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} C_{\log}\Big(\mathcal{B}(\mathcal{W}_{\boldsymbol{\lambda}}(x_t), \boldsymbol{\pi})\Big|\theta_i\Big) \tag{8.17}$$

w.r.t. $\boldsymbol{\lambda}$. Note that for two-class, Bayes' rule, $\mathcal{B}$, is the logistic sigmoid of $w + \operatorname{logit}(\pi)$, while for multi-class, Bayes' rule is the softmax function of $\mathbf{w} + (\log \pi_1, \dots, \log \pi_N)$. This shows the close relationship between our objective (8.17) and the traditional formulations of two-class or multi-class logistic regression in [1] chapter 4. The only difference is in how the prior is handled. In our case the given, fixed log priors are explicitly added to the model log-likelihoods, while in [1], the model is expected to implicitly include the prior in its log-likelihoods.

Our objective $\mathcal{E}_{\log}(\mathcal{W}_{\boldsymbol{\lambda}}|\boldsymbol{\pi})$ changes as a function of the prior. For different priors, slightly different solutions are obtained, but in general we have found that models trained at one prior, and then evaluated over a range of priors, tend to perform well over a large range of priors. Examples are given below.

Optimization algorithms

Logistic regression optimization does not have a known closed-form solution and is done via iterative numerical optimization. Efficient optimization methods for this problem typically make use of first and second-order derivatives of the objective function w.r.t. the parameters. See e.g. [1] for formulas of the derivatives. These derivatives can be used in various optimization algorithms, such as non-linear conjugate gradient, or quasi-Newton methods [80]. Our implementations are based on [73] and are available here.[7]

8.3.2 Examples: Speaker recognition

When fusing and calibrating speaker recognition systems for submission to a NIST Speaker Recognition Evaluation where the evaluation criterion is DCF, we parametrize our training objective, $\mathcal{E}_{\text{log}}(\mathcal{W}_{\boldsymbol{\lambda}}|\pi)$, at NIST's operating point: $\pi = 0.091$ for the old and $\pi = 0.001$ for the new, as explained in section 7.3.1.

In figure 8.2 we show an example of the DET-curve of the fusion of multiple speaker recognizers for SRE 2006. See our journal paper [21] for further details.

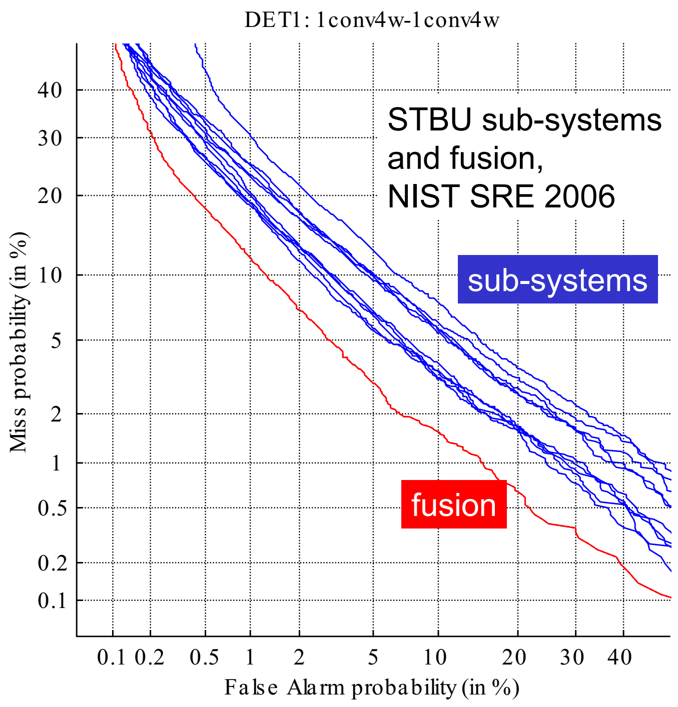

Figure 8.2: DET-curve of logistic regression fusion (red) of multiple speaker recognition sub-systems (blue).

Also recall figure 7.8, where a speaker recognition system was calibrated at the new operating point on a development database and then performed well over the whole displayed range of operating points on the independent evaluation data of SRE 2010.

[7]FoCal, a MATLAB toolkit for fusion and calibration, based on logistic regression, is freely available at `http://focaltoolkit.googlepages.com`.

Calibration is not normalization

Now also recall figure 7.9, where calibration did not work. The fact that the term *calibration* is used to refer to the *calibration criterion* as well as the *act of calibrating* a recognizer may be confusing here. The calibration criterion for this system was bad, but in this case was not due to a bad calibration strategy. Logistic regression effectively computes the log-likelihood-ratio between smooth (exponential family) probability distributions [1]. If these distributions shift between training and evaluation, then there is nothing calibration can do to correct the problem. The calibration transform takes only the score as input and cannot normalize or compensate the score in any data-dependent way. The recognizer of figure 7.8 has good calibration, because its (pre-calibrated) scores did not shift between training and evaluation as did the scores of the system of figure 7.9.

8.3.3 Examples: Language recognition

When fusing and calibrating language recognition systems for submission to a NIST Language Recognition Evaluation where the evaluation criterion is CAVG, we parametrize the training criterion, $\mathcal{E}_{\text{log}}(\mathcal{W}_{\boldsymbol{\lambda}}|\boldsymbol{\pi})$, at the uniform prior, $\boldsymbol{\pi} = \bar{\boldsymbol{\pi}} = (\frac{1}{N}, \dots, \frac{1}{N})$.

Figure 8.3 shows an example from NIST LRE 2007, where there were $N = 14$ languages. We trained a fusion, $\mathcal{W}_{\boldsymbol{\lambda}}$, of multiple subsystems at the uniform prior $\pi_i = \frac{1}{14}$ on an independent training database. Then we performed evaluations on LRE 2007 data, at 16369 different priors. By zeroing some of the prior components, and renormalizing the remaining components, we effectively formed a different LRE over each of the 16369 non-empty subsets of the 14 languages. The LRE for each subset was evaluated with $\mathcal{E}_{\text{lre}}(\mathcal{W}_{\boldsymbol{\lambda}}|\boldsymbol{\pi})$, and plotted on the vertical axis of figure 8.3. Every dot represents the performance of one subset. This experiment shows that calibration trained at one prior gives a recognizer that works over a large range of different applications. We argue that this is due to the application summarizing character of the logarithmic cost function.

For further examples of fusion of multiple language recognition systems in LRE 2009, see our paper [24].

Multi-class vs one-against-the-rest calibration

The formulation of CAVG as a mixture of discrete applications may have served as inspiration for several LRE 2007 participants to build a separate, one-against-the-rest, two-class detector for each target language. Each of the 14 language detectors was calibrated independently of the others. In figure 8.4 we show the CAVG performance of four anonymous participants in this evaluation. The blue bars represent the performance of the original systems as they

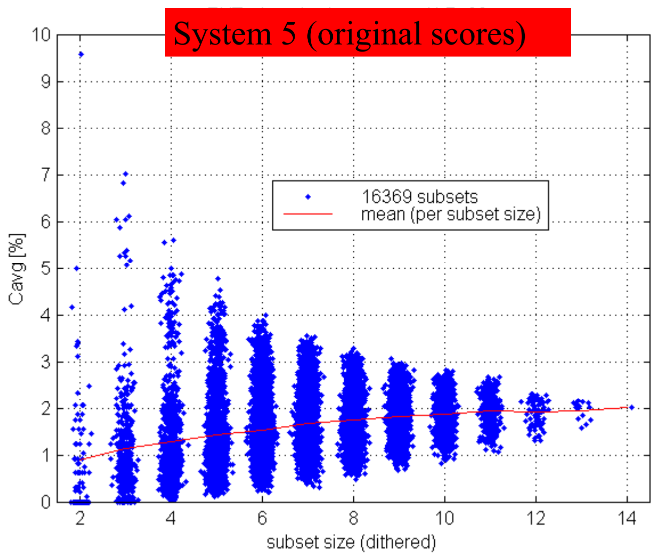

Figure 8.3: Evaluation of a single well-calibrated recognizer on LRE 2007 data reorganized into all the non-empty subsets of 14 languages. Each dot represents CAVG $= \mathcal{E}_{\text{lre}}(\mathcal{W}_{\boldsymbol{\lambda}}|\boldsymbol{\pi})$, where the non-zero components of $\boldsymbol{\pi}$ define the subsets.

were submitted. The brown bars represent our experiments at re-calibrating these recognizers, each done with a single multi-class transform as defined by (8.16), with $K = 1$. The calibration parameters were trained with logistic regression on independent training data. In each case, the multi-class calibration outperformed the original one-against-the-rest calibration.

8.4 Calibration analysis

In previous chapters, we have already done most of the groundwork for explaining calibration analysis. For the two-class case, we showed that the PAV solution has some attractive properties in the sense that (i) the calibration analysis is independent of the prior and of the proper scoring rule and (ii) the *monotonic rising* restriction on calibration transformations agrees with the score thresholding principle that defines ROC, ROCCH, DET, minDCF and EER analysis.

We were unable to find any such neat solution for the multi-class case. One of the big problems as we see it, is that the considerably more complex behaviour of decision boundaries in the multi-class probability (or log-likelihood) space does not suggest an analogue to the above monotonicity principle. Here we present our solution that was originally proposed in [19].

In the absence of a suitable restriction on non-parametric calibration transformations, we instead choose a family of parametric transformations. As shown in section 7.7.1, a principled restriction to place on calibration transformations is that they form a group under function composition. This restric-

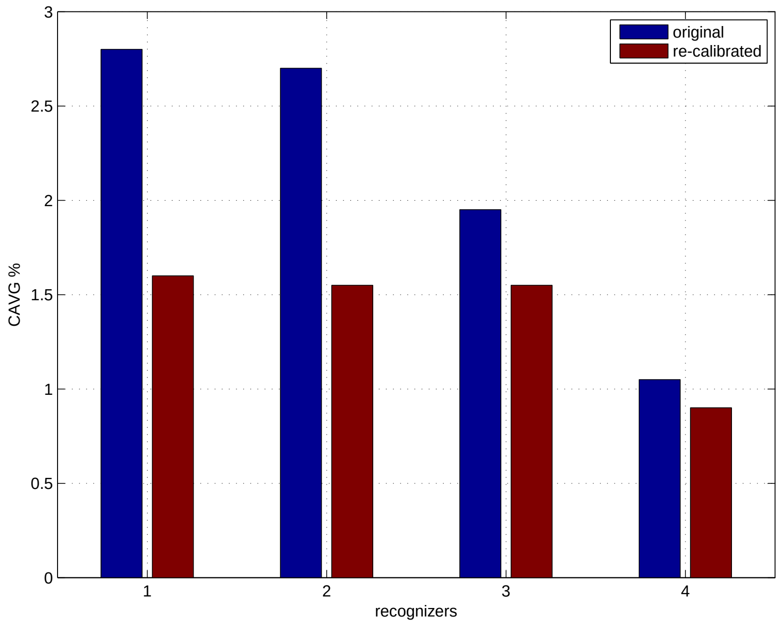

Figure 8.4: Calibration experiments on LRE 2007. The blue bars represent CAVG of four language recognizers as originally submitted to LRE'07 and which used one-against-the-rest calibration. The brown bars represent the same recognizers, re-calibrated with multi-class logistic regression. The re-calibration was done using independent training data.

tion leads to the interpretation that if the evaluator optimizes the calibration transformation it can be interpreted as the optimal calibration that the evaluee could have used.

For the multi-class case, we work with calibration transformations which are bijections from $\mathbb{L}_N$ to $\mathbb{L}_N$. (The reader may need to review section 4.2.1 to recall how $\mathbb{L}_N$, the space where log-likelihood vectors live, is defined.) The input to the transformation, even though it is in $\mathbb{L}_N$, can be regarded as an uncalibrated *score vector*, while the output fulfils the role of a calibrated log-likelihood vector. If it is assumed that the evaluee used some calibration transformation in this group to convert score vector to log-likelihood vector, then the evaluator's subsequent transformation of the submitted log-likelihood vector can be interpreted as undoing the original calibration transformation to recover the original score vector and then re-calibrating it.

8.4.1 Affine calibration transformations

Here we show how to define a practical implementation of a calibration transformation $F : \mathbb{L}_N \mapsto \mathbb{L}_N$. Let $\acute{\mathbf{s}} \in \mathbb{L}_N$ be the input to the calibration transformation and let $\acute{\mathbf{w}} = F(\acute{\mathbf{s}}) \in \mathbb{L}_N$ be the output. We can interpret $\acute{\mathbf{s}}$ as the uncalibrated score vector and $\acute{\mathbf{w}}$ as the calibrated log-likelihood vector. Recall from section 4.2.1 that $\acute{\mathbf{s}}$ and $\acute{\mathbf{w}}$ are diagonal *lines* in $\mathbb{R}^N$, such that any point on the line is equivalent when used in Bayes' rule.

Now we implement $\acute{\mathbf{w}} = F(\acute{\mathbf{s}})$, by using a function $f : \mathbb{R}^N \mapsto \mathbb{R}^N$ that maps any point $\mathbf{u} \in \acute{\mathbf{s}}$ to some point in $\acute{\mathbf{w}}$:

$$f(\mathbf{u}) \in F(\acute{\mathbf{s}}) \text{ for every } \mathbf{u} \in \acute{\mathbf{s}}\,. \tag{8.18}$$

This places a restriction on f, because *different* points $\mathbf{u} \in \acute{\mathbf{s}}$, must all be mapped to the *same* output line $\acute{\mathbf{w}}$. Another way to state this condition is as follows:

> Let $\mathbf{1} = (1, 1, \dots, 1) \in \mathbb{R}^N$. For any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$ and $k \in \mathbb{R}$, such that $\mathbf{u} - \mathbf{v} = k\mathbf{1}$, there must be some $k' \in \mathbb{R}$, such that $f(\mathbf{u}) - f(\mathbf{v}) = k'\mathbf{1}$.

This condition is satisfied by letting f be an affine transform:

$$f(\mathbf{u}) = \mathbf{A}\mathbf{u} + \mathbf{b} \tag{8.19}$$

where $\mathbf{b} \in \mathbb{R}^N$ and where $\mathbf{A}$ is an N-by-N matrix for which $\mathbf{1}$ is an eigenvector.

Pure calibration excludes rotation

We now present arguments for further restriction on $\mathbf{A}$. Recall that we want to limit the evaluator's calibration transformations to ones that could reasonably have been used by the evaluee. A full N-by-N matrix allows too many degrees of freedom, which allows the evaluator, who has access to the true class labels, to over-optimize.[8]

Moreover a full matrix does rotation of the log-likelihood vector which we argue violates the sense of the submitted scores. Consider the case of $N = 3$, where the to-be-recognized classes are Spanish, German and Czech. The submitted log-likelihood vector, represented in $\mathbb{R}^N$, is $\mathbf{w} = (w_1, w_2, w_3)$, where w_1 is the log-likelihood for Spanish, w_2 for German and w_3 for Czech. Now suppose a prior, $\boldsymbol{\pi} = (\pi, 1-\pi, 0)$, is given which excludes the possibility of Czech. The posterior, $\mathcal{B}(\mathbf{w}, \boldsymbol{\pi})$, is independent of the value of w_3, so that Spanish and German are distinguished without using the log-likelihood for Czech. If however, we use the re-calibrated $\mathbf{w}' = \mathbf{A}\mathbf{w} + \mathbf{b}$, then the posterior $\mathcal{B}(\mathbf{w}', \boldsymbol{\pi})$ becomes dependent on w_3. Such calibration transformations could therefore use the Czech log-likelihood to distinguish between Spanish and German.

[8]Compare this to the non-parametric PAV solution for two classes, where if there are millions of scores, we also seem to have millions of degrees of freedom. But the monotonicity constraint also allows very little leeway for each individual score. The more scores there are, the more they 'crowd' each other, since they are not allowed to move past each other. The effective number of degrees of freedom is the number of distinct blocks in the PAV solution, which is typically very small.

It turns out that in many state-of-the-art language recognition systems, such rotating score transformations can indeed improve accuracy. Both generatively and discriminatively trained, full-matrix, transformations can be used. See for example [48] and also our work [19, 24].

However, we argue that such transformation are fusions, which combine information sources, rather than calibrations which just re-format the available information. If we want to limit our calibration analysis to use transformations that have a pure calibration sense, we need to restrict $\mathbf{A}$ to be diagonal.

We now have two constraints on $\mathbf{A}$: It should have $\mathbf{1}$ as an eigenvector and it should be diagonal. This is satisfied if the diagonal elements are all equal. The calibration transformation now reduces to:

$$f(\mathbf{u}) = \alpha\mathbf{u} + \mathbf{b} \tag{8.20}$$

for some scalar $\alpha > 0$. Notice this family of transformations forms a group under function composition as required. Since this transformation involves only the vector-space operations of scaling and translation, we can equivalently specify it as:

$$F(\acute{\mathbf{s}}) = \alpha\acute{\mathbf{s}} + \acute{\mathbf{b}}\,. \tag{8.21}$$

8.4.2 Optimization of calibration transform

We are now ready to specify the full recipe for multi-class calibration analysis. Let $\mathcal{W}$ be the original submitted recognizer. Given a supervised evaluation database, $\mathcal{W}$ can be evaluated as $\mathcal{E}_{\text{log}}(\mathcal{W}|\boldsymbol{\pi})$, or $\mathcal{E}_{\text{lre}}(\mathcal{W}|\boldsymbol{\pi})$. Now let $\mathcal{W}_F$ be the recalibrated recognizer, such that $\mathcal{W}_F(x) = \alpha\mathcal{W}(x) + \acute{\mathbf{b}}$, where we used (8.21). Now we define:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}_F} \mathcal{E}_{\text{log}}(\mathcal{W}_F|\boldsymbol{\pi}) \tag{8.22}$$

$$\mathcal{E}_{\text{log}}^{\text{opt}}(\mathcal{W}|\boldsymbol{\pi}) = \mathcal{E}_{\text{log}}(\mathcal{W}^*|\boldsymbol{\pi}) \tag{8.23}$$

$$\mathcal{E}_{\text{lre}}^{\text{opt}}(\mathcal{W}|\boldsymbol{\pi}) = \mathcal{E}_{\text{lre}}(\mathcal{W}^*|\boldsymbol{\pi}) \tag{8.24}$$

where the minimization is w.r.t. α and $\mathbf{b}$. Notice that we optimize only the logarithmic objective and then use the optimum calibration parameters to define both $\mathcal{E}_{\text{log}}^{\text{opt}}$ and $\mathcal{E}_{\text{lre}}^{\text{opt}}$. As before, when optimizing for calibration analysis, we relax the restriction of positive α to $\alpha \geq 0$.

Letting $\mathcal{W}_0$ be the default recognizer (which corresponds to $\mathcal{W}_F$ at $\alpha = 0, \acute{\mathbf{b}} = \acute{\mathbf{0}}$), we find:

$$\mathcal{E}_{\text{log}}(\mathcal{W}|\boldsymbol{\pi}) \geq \mathcal{E}_{\text{log}}^{\text{opt}}(\mathcal{W}|\boldsymbol{\pi}) \leq \mathcal{E}_{\text{log}}(\mathcal{W}_0|\boldsymbol{\pi}) = C_{\text{log}}^*(\boldsymbol{\pi}) \tag{8.25}$$

which shows for example that $L_{\text{cal}} = \mathcal{E}_{\text{log}}(\mathcal{W}|\boldsymbol{\pi}) - \mathcal{E}_{\text{log}}^{\text{opt}}(\mathcal{W}|\boldsymbol{\pi})$ forms a well-defined, non-negative, calibration loss criterion. We used this in preparation and post-analysis for LREs 2007 and 2009.

The above inequalities do not apply to $\mathcal{E}_{\text{lre}}(\mathcal{W}|\boldsymbol{\pi})$ and $\mathcal{E}_{\text{lre}}^{\text{opt}}(\mathcal{W}|\boldsymbol{\pi})$, because the optimization here was not done for the (optimization-unfriendly) $\mathcal{E}_{\text{lre}}$ criterion itself. Nevertheless, if performance in a NIST LRE is of interest, this contrast is a useful indicator of calibration problems in the recognizer. Also $\mathcal{E}_{\text{lre}}^{\text{opt}}$ on its own forms a useful criterion when calibration is not of immediate interest. We used this as evaluation criterion in our language-recognition paper [23].

Finally notice that for the case $N = 2$: $\mathcal{E}_{\text{log}}^{\text{min}}(\mathcal{W}|\boldsymbol{\pi}) \leq \mathcal{E}_{\text{log}}^{\text{opt}}(\mathcal{W}|\boldsymbol{\pi})$, because the affine calibration restriction is stricter than the monotonicity constraint of the PAV solution.

Computation

The optimization (8.22) is just multi-class logistic regression and can be implemented with the methods discussed in section 8.3.1.

8.4.3 Example: Three-class calibration

We conclude this section with an example of the application of calibration analysis in a three-class speaker recognition problem, as originally published in our paper [20].

test	train	$\mathcal{E}_{\text{log}}$	$\mathcal{E}_{\text{log}}^{\text{opt}}$	L_{cal}	% err
2006	-	0.92	0.24	0.68	6.67
2006	2006	0.24	0.24	0.00	5.44
2008	-	0.78	0.21	0.57	8.20
2008	2006	0.23	0.21	0.01	6.54
2008	2008	0.21	0.21	0.00	6.05

Table 8.1: Logarithmic cost and error-rate, when recognizing the number of speakers present in three speech segments. The test databases were assembled from the male speakers in SRE 2006 or 2008. An optional calibration transformation was trained on either 2006 or 2008.

The problem of interest was, given three input speech segments, to recognize how many speakers, 1, 2 or 3, are present. This is a three-class problem. A recognizer for this problem was built (using independent development data) and then tested on each of two supervised databases, respectively assembled from NIST SRE 2006 and 2008. The results are shown in table 8.1.

The recognizer output was in log-likelihood form and was therefore suitable for evaluation by $\mathcal{E}_{\text{log}}$. We evaluated the raw recognizer outputs, as is (rows with '-' in the second column) as well as a re-calibrated version, re-calibrated via a discriminatively trained affine calibration transform of the form (8.21).

The training was done with logistic regression on either 2006 or 2008, as indicated in the second column.

To do calibration analysis, we reported $\mathcal{E}_{\text{log}}^{\text{opt}}$ and $L_{\text{cal}} = \mathcal{E}_{\text{log}} - \mathcal{E}_{\text{log}}^{\text{opt}}$. In the rows that report train and test on the same database: $\mathcal{E}_{\text{log}} = \mathcal{E}_{\text{log}}^{\text{opt}}$, per definition. In the last column we report the total percentage of misclassification errors. For details, see [20].

The conclusion is that discriminatively trained re-calibration gives a dramatic improvement when evaluating by $\mathcal{E}_{\text{log}}$, even when train and test databases are independent. The effect of re-calibration on the misclassification error-rate is less dramatic, because for that criterion, which has fixed prior and cost, the log-likelihood scaling is unimportant. If one sweeps the prior or cost function parameters, as $\mathcal{E}_{\text{log}}$ effectively does, then calibration becomes important.

The most important point however, is to appreciate the fact that by measuring the calibration loss of the original solution, we saw that there was indeed a calibration problem and that results could be dramatically improved with re-calibration, which involves a minor effort compared to building the original recognizer.

8.5 What happened to DET, minCAVG and EER?

Readers that are familiar with the language recognition literature and the NIST language recognition evaluations in particular, will have seen the use of DET-curves, minCAVG and EER applied to language recognition. See for example [26, 48]. We analysed DET-curves, minDCF and EER for the two-class problem at great length in chapter 7. Why are they being ignored here?

The reason is that those tools, in the way they have been applied in the language recognition literature are not multi-class evaluation criteria, but are still the two-class versions. The multi-class language recognition scores are subjected to a non-invertible transformation to form what appear to be two-class scores, which are then plugged into the existing two-class machinery for computing DET-curves, minDCF and EER. The problem is that by so doing, these criteria lose most of their meaning. In particular, the threshold sweep that is implicit in all these criteria no longer plays the role of calibration optimization as explained in chapter 7.

The problem starts with the format of the scores. The NIST LRE evaluation plans to date[9] have asked for N *detection scores* for every speech segment, such that score k can be thresholded to accept or reject the proposition that the speech segment is in target language k. Given a log-likelihood vector, $\mathbf{w} = (w_1, \ldots, w_N)$, a detection score can be computed for each target, k, for

[9]See `www.itl.nist.gov/iad/mig/tests/lre`.

example, as:

$$s_k = w_k - \log \sum_{j=1}^{K} \exp(w_j)\,. \tag{8.26}$$

If $\mathbf{w}$ is regarded as well-calibrated, then so is s_k and it can be thresholded at $-\log N$ to make the optimal Bayes decision, which minimizes CAVG. (Any other form for s_k, which differs from (8.26) by a continuous, strictly monotonic rising transformation could do the same job.) So far there is no problem.

The first problem arises if the input to (8.26) is not well-calibrated. If we still assume that $\mathbf{w}$ is well-calibrated, but that the input to (8.26) is scaled and shifted, then the resultant detection score is

$$s'_k = \alpha w_k + \beta_k - \log \sum_{j=1}^{K} \exp(\alpha w_j + \beta_j)\,. \tag{8.27}$$

The problem is now that in general, there is *no* function f, such that $s_k = f(s'_k)$. Re-calibrating the s'_k individually can improve accuracy, but not as much as re-calibrating the input $\mathbf{w}$, as was demonstrated in figure 8.4.

The next problem occurs because we still have N score streams and we want to evaluate them with two-class tools that expect a single score stream. Now since all the scores have the same sense (larger, more positive, scores favour the target, smaller ,more negative, scores favour the non-target), it sounds like a good idea just to *pool* the score streams into a single score stream. Moreover, as we pointed out above, each of the pooled score streams has the same optimal threshold.

The problem is that the miss-calibrations in the different score streams are of *opposing* sense. If the recognizer is biased towards one class, the scores for that class would be shifted towards the positive side and all the other scores towards the negative. Now sweeping a single threshold cannot find a threshold that would be optimal for both senses. For these reasons, late calibration and score pooling, any two-class threshold-sweeping score analysis tool, like DET, minDCF and EER, cannot give a calibration-insensitive criterion like they do for a true two-class problem.

The argument has been voiced that the intention for using these two-class tools is not calibration insensitivity, but just to sweep a range of operating points for scores that are intended to be well-calibrated. Yes, the DET-curve does sweep operating points, but its (now broken) calibration compensation mechanism is still conflated with the operating point sweep. (We propose below how a pure operating point sweep can be achieved.) Moreover, if the intention is for the evaluator to not optimize calibration, then what is the meaning of minCAVG, which goes through the motions of optimizing a threshold for a fixed operating point? Likewise, the meaning and utility of EER remains unclear. It certainly does not give a calibration-insensitive, application-spanning summary as it does for speaker detection.

Finally, the argument has been voiced that in speaker detection, the scores of many speakers are pooled for analysis by DET, minDCF and EER. Why should scores not be pooled across languages? But speaker detection is a *two-class* problem—and nobody has ever suggested pooling the scores for the target and the non-target propositions. Specifically a speaker detection evaluation is structured in such a way that an equation of the form (8.26) does not apply. The speaker detection score for every trial is not computed via the likelihoods of a *fixed* set of speakers.

8.5.1 Proposal for sweeping the LRE operating point

Here we propose a way to sweep a parametrized CAVG-like evaluation criterion over a range of operating points, defined in such a way that all targets are rejected in the one extreme and all are accepted in the other extreme. This is done by introducing a variable target prior, say α, into (8.1), which then becomes:

$$\pi_{ik} = \alpha\delta_{ik} + \frac{(1-\alpha)(1-\delta_{ik})}{(N-1)}\,. \tag{8.28}$$

where α is varied from 0 to 1. The same derivation can then be followed as before, to find a version of C^*_{lre} that is parametrized by α. A curve of $\mathcal{E}_{\text{lre}}$ (maybe normalized) against α, or maybe logit α, can then be plotted to show error-rate as a function of the target prior. In addition, $\mathcal{E}^{\text{opt}}_{\text{lre}}$ may be added as a contrast to analyse calibration.

8.6 Summary

In this chapter we analysed LRE's CAVG by viewing it as a mixture of applications that exercises the recognizer's output at a number of discrete operating points in the probability simplex. We showed that the multi-class logarithmic cost function forms a continuous mixture of applications that exercises the recognizer's output *everywhere* in the probability simplex. In this way logarithmic cost forms a representative evaluation criterion for a wide range of applications. Moreover, logarithmic cost forms a convenient discriminative training criterion, which reduces to logistic regression for linear models.

For multi-class, a direct analogue to the continuous application-sweeping plots of chapter 7 are problematic because of high dimensionality. Instead we showed how to form a scatter-plot of very many discrete operating points.

For multi-class, a generalization of the non-parametric PAV calibration analysis is not available. Instead we derived a simple, affine, parametric transform that can be optimized via logistic regression. We showed how it can be applied to both measure and improve calibration of a multi-class recognizer.

Chapter 9

Conclusion

In this work we examined data-driven methods to evaluate the goodness of probabilistic pattern recognizer outputs in a very direct way, by effectively using the outputs to make decisions and then recording how good those decisions are. There is a close connection between this form of evaluation and *discriminative modelling*, where the model parameters are discriminatively trained by optimizing the evaluation criterion.

This forms an interesting contrast with *generative modelling* for pattern recognition [1, 10]. Like discriminative modelling, generative modelling is also data-driven, but the concept of training differs. Ideally, a generative model should not be trained—all hidden model parameters should be integrated out in a fully Bayesian way. However, in practice one usually has to resort to making point estimates of some of the parameters in the model and this constitutes *training* of those parameters. This training is accomplished by optimization of an objective function, which makes it similar to discriminative training. However, the optimization objective is different.

In generative training, one optimizes the model to maximize the *evidence*, $P(\text{data}, \text{classes}|\text{model})$, while in discriminative training one maximizes[1], or more generally optimizes, $P(\text{classes}|\text{data}, \text{model})$.

In the last decade in text-independent speaker recognition research, there has been an interesting tug-of-war between generative and discriminative training techniques. Some landmark papers (by no means exhaustive), in chronological order, are: [81], which represented the state-of-the-art in generative GMM-based speaker recognition; [82, 83], which established SVM as a discriminatively trained alternative to GMM (although the associated NAP-transformation was more generative than discriminative in nature); [21], an example of the power of discriminative system fusion; [84], which introduced the large-scale generatively trained JFA as a new monolithic state-of-the-art, which often outperformed the best heterogeneous fusions; [54], the 2008 JHU summer workshop, originally conceived to explore new discriminative training

[1]for the logarithmic cost function

techniques, but which ended up instead improving on generative JFA, as well as seeding the new, generative, i-vector modelling [85, 86], which is currently an important part of the state-of-the-art.

This leaves the question whether there may be good generative alternatives to solving the problems addressed in this work. Thus far, good generative models have been a partial answer to discriminative fusion, but as far as we know, no convincing generative alternative for implementing *calibration transformation* has been demonstrated.

Ideally, if generative modelling works well enough, then both fusion and calibration transformation should become unnecessary. The raw scores of an ideal generative model should be accurate enough not to have to fuse with any other recognizer and moreover, should be naturally well-calibrated.

The problem that remains however, is how do we know a recognizer is well-calibrated? Can one ever get away with not explicitly testing calibration as we proposed here, or with some equivalent method? The problem with the methodology in our proposal is, as we pointed out, that it breaks down at extreme operating points, where the errors become too scarce to count. But is there any other way of judging the goodness of calibration that does not rely on large amounts of evaluation data?

A contrast exists between the evaluation methods proposed in this work and the field of forensic DNA, as represented by Balding in [87], where generative models are used to compute likelihood-ratios, which are very similar in interpretation and function to the likelihood-ratios in speaker detection. The big difference is that the DNA is typically more discriminative than speaker recognition by orders of magnitude, so that DNA can be employed to literally find a single individual out of millions of candidates. The problem is that performing data-driven evaluation to test calibration at such extreme operating points would be intractable. The result is that evaluation of the goodness of the calibration of DNA likelihood-ratios can only be performed by expert perusal of the generative modelling strategies that are used to compute them.

Is there a good way to bridge this gap between the blind data-driven evaluation paradigm proposed in this work and the pure expert evaluation paradigm of DNA?

Appendix A

Functions: Surjective, injective, bijective

This note explains the difference between an invertible function and a bijection. Bijections are invertible, but there are invertible functions which are not bijections. See for example [88].

A.1 Domain, codomain, range

For a function that is defined as $f : \mathcal{X} \mapsto \mathcal{Y}$, we have the following definitions:

domain is the set $\mathcal{X}$ of input values which are allowed.

codomain is the set $\mathcal{Y}$, in which this function may take values.

range which we write: $\text{range}(f) = \{y \in \mathcal{Y} : y = f(x) \text{ and } x \in \mathcal{X}\}$ is the set of output values that can actually be reached with inputs from the domain. Note that in general[1], $\text{range}(f) \subseteq \mathcal{Y}$.

Consider for example the function that is defined as:

$$f : \mathbb{R}^2 \mapsto \mathbb{R}, \text{ such that } f(x, y) = x^2 + y^2.$$

Here $\mathbb{R}^2$ is the *domain*, or the set of inputs (x, y) for which the function is defined. The *codomain* of this example is $\mathbb{R}$, and the *range* is the subset of non-negative real numbers.

A.2 Surjective, injective, bijective

A succinct definition of these terms is:

[1]If there is an $x \in \mathcal{X}$, such that $f(x) \notin \mathcal{Y}$, then the function definition is malformed.

> A function $f : \mathcal{X} \mapsto \mathcal{Y}$ is *surjective/injective/bijective*, if for every $y \in \mathcal{Y}$, there is *at least/at most/exactly* one $x \in \mathcal{X}$, such that $f(x) = y$.

A bijective function is both surjective and injective. An injective function is invertible, but there are invertible functions which are not surjective. An example is $f : \mathbb{R} \mapsto \mathbb{R}$, where $p = f(x) = \frac{1}{1+\exp(-x)}$, which has inverse $x = f^{-1}(p) = \log \frac{p}{1-p}$, but range$(f) = (0, 1) \subset \mathbb{R}$. The inverse, f^{-1}, is not defined everywhere in the codomain, $\mathbb{R}$.

Appendix B

Doddington's Rule of 30

Given a few assumptions to be listed below, *Doddington's Rule of 30* applies when estimating an unknown error-rate p, as $\hat{p} = \frac{t}{T}$, when t errors are observed out of a total of T independent trials [64]:

> If you want to be 90% confident that $\hat{p}$ is within 30% of p, you need $t \geq 30$.

This rule may be derived given the following assumptions:

1. The number of errors, t, is distributed according to a binomial distribution with parameters p (the true error-rate) and T (the number of trials).
2. p is small
3. The expected number of errors, Tp is sufficiently large for the normal approximation to the binomial distribution to hold. (Sufficiently large is often taken as $Tp \geq 5$. This is well below the 30 errors prescribed by Doddington's rule, so that this assumption will most probably[1] hold.)

Using the normal approximation, we may write:

$$t \sim \mathcal{N}\big(Tp, Tp(1-p)\big), \qquad \hat{p} \sim \mathcal{N}\left(p, \frac{p(1-p)}{T}\right) \tag{B.1}$$

where $\sim$ denotes *is distributed as* and $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. Now let the normalized[2] error of the estimate be

$$e = \frac{\hat{p} - p}{\hat{p}} \tag{B.2}$$

[1]Given only experimental data, we cannot guarantee it is satisfied, because p is unknown. In fact we cannot guarantee that any of the assumptions hold.

[2]We could have defined the normalized error as $e' = \frac{\hat{p}-p}{p}$, but $e = \frac{\hat{p}-p}{\hat{p}}$ gives a more convenient result. Note that when $\hat{p} \approx p$, then also $e' \approx e$.

so that

$$e \sim \mathcal{N}\left(0, \sigma_e^2\right), \qquad \sigma_e^2 = \frac{(1-p)}{t} \approx \frac{1}{t}. \tag{B.3}$$

This final approximation $\sigma_e^2 \approx \frac{1}{t}$ is very convenient because the error-variance is now expressed as a function of the error-count t alone.

The derivation is completed by requiring a probability of $\geq 90\%$ that e is in the interval $[-0.3, 0.3]$. Using the *inverse error-function* [67, 72] gives: $\sigma_e^2 \leq 0.033$, which by the above approximation gives $t \geq 30$. ■

Appendix C

PAV optimizes regular binary proper scoring rules

In this appendix we prove via a series of lemmas and theorems that the PAV algorithm assigns optimal probabilities w.r.t. any regular binary proper scoring rule, subject to a monotonicity constraint. The problem of interest may be stated as follows:

- A *regular binary proper scoring rule* (RBPSR), as defined in section 7.4.1, is a function $C^*_\rho : \{\theta_1, \theta_2\} \times [0,1] \mapsto [0,\infty]$, such that

$$C^*_\rho(q|\theta_1) = \int_q^1 \frac{1}{\eta}\rho(\eta)\,d\eta, \qquad C^*_\rho(q|\theta_2) = \int_0^q \frac{1}{1-\eta}\rho(\eta)\,d\eta \tag{C.1}$$

  where $\rho(\eta)$ is a probability distribution over $[0,1]$. If $\rho(\eta) > 0$ almost everywhere, then the RBPSR is denoted *strict*, otherwise it is *non-strict*[1].

- We are given as input:
  - A sequence of T indices, denoted $(1,T) = 1, 2, \ldots, T$ with a corresponding sequence of labels $\ell_1, \ell_2, \ldots, \ell_T \in \{\theta_1, \theta_2\}$.
  - A pair of positive weights, $v_1, v_2 > 0$. We shall use the notation $\mathsf{v}(\ell_t)$ to associate one of these weights with every label, by letting $\mathsf{v}(\theta_1) = v_1$ and $\mathsf{v}(\theta_2) = v_2$.

- The problem is now to find the sequence of T probabilities, denoted $\mathbf{p}_{1,T} = p_1, p_2, \ldots, p_T$, that minimizes the following *objective*:

$$\mathcal{O}_{1,T}(\mathbf{p}_{1,T}) = \sum_{t=1}^{T} \mathsf{v}(\ell_t) C^*_\rho(p_t|\ell_t) \tag{C.2}$$

[1]We define below in the proof of lemma 4 what we mean by *almost everywhere* and how this affects the behaviour (strictness) of the RBPSR.

subject to the *monotonicity constraint*:

$$0 \leq p_1 \leq p_2 \leq \cdots \leq p_T \leq 1\,. \tag{C.3}$$

We require the solution to hold (be a feasible minimum) *simultaneously* for every RBPSR C^*_ρ. We already know that if such a solution exists, it must be unique, because the original PAV algorithm, as published in [75] in 1955, was shown to give a unique optimal solution for the special case of $\bigl(C^*_\rho(p|\theta_1), C^*_\rho(p|\theta_2)\bigr) = \bigl(-\log(p), -\log(1-p)\bigr)$. See theorem 1 and lemma 1 below for details.

We construct a proof that the PAV algorithm solves the above problem, by roughly following the pattern of the unpublished document [89], where the optimality of PAV was proved for the case of *strictly convex* cost functions. That proof is not applicable as is for our purposes, because while some RBPSRs like the logarithmic and Brier scores are strictly convex, some are not (recall figure 7.4, which shows three examples of non-convex RBPSRs.).

We will show however in lemma 4 below, that all RBPSRs and their expectations are *quasiconvex* and that the proof can be based on this quasiconvexity, rather than on convexity. Note that when working with convex cost functions, one can use the fact that positively weighted combinations of convex functions are also convex, but this is not true in general for quasiconvex cost functions. For our case it was therefore necessary to prove explicitly that expectations of RBPSRs are also quasiconvex.

A further complication that we needed to address was that non-strict RBPSRs lead to unidirectional implications in places where the strictly convex cost functions of the proof in [89] give *if and only if* relationships. We note that although the more general case of PAV for non-strict convex cost functions was treated in [90], we could not base our proof on theirs, because they used properties of convex functions, such as subgradients, which are not applicable to our quasiconvex RBPSRs.

Finally, we then show that the PAV algorithm can also be applied to find optimal log-likelihood-ratios, subject to a similar monotonicity constraint and that this solution is not only independent of the RBPSR, but also of the prior. We shall call this application of the PAV algorithm the PAV-LLR algorithm.

See figure C.1 for a roadmap of the proof: Theorem 1 and lemma 1 give the closed-form solution for the logarithmic RBPSR. For the PAV-algorithm, we use lemma 1 just to show there is a unique solution, but we also use it later to prove the prior-independence of the PAV-LLR algorithm. Inside the dashed box, theorem 2 shows how optimal subproblem solutions can constitute the optimal solution to the whole PAV problem. Theorems 3 and 4 show how to find and combine optimal subproblem solutions, so that the PAV algorithm can use them to meet the requirements of theorem 2.

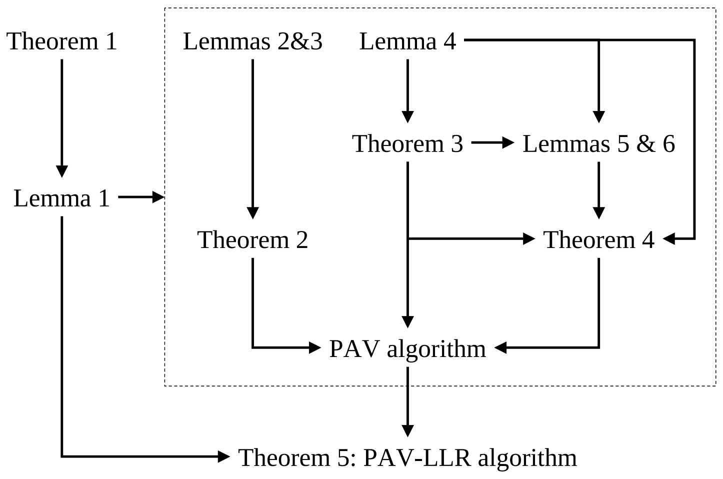

Figure C.1: Proof structure: PAV is optimal for all RBPSRs and PAV-LLR is optimal for all RBPR's and priors.

C.1 Unique solution (lemma 1)

In this section, we use the work of Ayer et al, reproduced here as theorem 1, to show via lemma 1 that if our problem does have a solution for every RBPSR, then it must be unique, because the special case of the logarithmic scoring rule (when $\rho(\eta) = 1$) does have a unique solution.

Theorem 1 (Ayer et al., 1955)**.** *Given non-negative real numbers a_t, b_t, such that $a_t + b_t > 0$ for every $t = 1, 2, \ldots, T$, the maximization of the objective $\mathcal{O}'_{1,T}(\mathbf{p}_{1,T}) = \prod_{t=1}^{T}(p_t)^{a_t}(1-p_t)^{b_t}$, subject to the monotonicity constraint* (C.3)*, has the unique solution, $\mathbf{p}_{1,T} = p_1, p_2, \ldots, p_T$, where:*

$$p_t = \max_{1\le i\le t}\ \min_{t\le j\le T} r'_{i,j} = \min_{t\le j\le T}\ \max_{1\le i\le t} r'_{i,j}\,, \tag{C.4}$$

$$r'_{i,j} = \frac{\sum_{k=i}^{j} a_k}{\sum_{k=i}^{j} a_k + b_k}\,. \tag{C.5}$$

Proof. See[2] [75], theorem 2.2 and its corollary 2.1. In that work, the monotonicity constraint was non-increasing, rather than the non-decreasing constraint (C.3) that we use here. The solution that they give therefore has to be transformed by letting the index t go in reverse order, which means exchang-

[2]Available online (with open access) at `http://projecteuclid.org/euclid.aoms/1177728423`.

ing the roles of the subsequence endpoints i, j, which then has the result of exchanging the roles of max and min in the solution. ■

We now show that this theorem supplies the solution for the special case of the logarithmic RBPSR:

Lemma 1. *If* $\big(C^*_\rho(p|\theta_1), C^*_\rho(p|\theta_2)\big) = \big(-\log(p), -\log(1-p)\big)$*, then the problem of minimizing objective* (C.2)*, subject to constraint* (C.3)*, has the unique solution,* $\mathbf{p}_{1,T} = p_1, p_2, \dots, p_T$*, where:*

$$p_t = \max_{1\le i\le t}\ \min_{t\le j\le T} r_{i,j} = \min_{t\le j\le T}\ \max_{1\le i\le t} r_{i,j}\,, \tag{C.6}$$

$$r_{i,j} = \frac{m_{i,j}v_1}{m_{i,j}v_1 + n_{i,j}v_2} \tag{C.7}$$

where $m_{i,j}$ *is the number of* θ_1*-labels and* $n_{i,j}$ *the number of* θ_2*-labels in subsequence* $\ell_i, \ell_{i+1}, \dots, \ell_j$*.*

Proof. Observe that if we let

$$(a_t, b_t) = \begin{cases} (v_1, 0), & \text{if } \ell_t = \theta_1, \\ (0, v_2), & \text{if } \ell_t = \theta_2, \end{cases}$$

then $r'_{i,j} = r_{i,j}$ and $\mathcal{O}'_{1,T}(\mathbf{p}'_{1,T}) = \exp\big(-\mathcal{O}_{1,T}(\mathbf{p}_{1,T})\big)$ so that the constrained maximization of theorem 1 and the constrained minimization of this lemma have the same solution. ■

By this lemma, we now have a closed-form solution to the problem, and from [75] we also know that this is the solution that is calculated by the iterative PAV algorithm.[3] As noted in the introduction, it has so far [75, 89, 90] only been shown that this solution is valid for logarithmic and other RBPSRs which have *convex* expectations. In the rest of this appendix therefore, we prove that this same solution also holds for all other RBPSRs.

C.2 Decomposition into subproblems (theorem 2)

We need to consider *subsequences* of $(1, T)$: For any $1 \le i \le j \le T$, we denote as (i, j) the subsequence of $(1, T)$ which starts at index i and ends at index j. We may compute a partial objective function over a subsequence (i, j) as:

$$\mathcal{O}_{i,j}(\mathbf{p}_{i,j}) = \sum_{t=i}^{j} \mathsf{v}(\ell_t) C^*_\rho(p_t|\ell_t)\,. \tag{C.8}$$

[3]The PAV algorithm, if efficiently implemented, is shown in the references cited here to have *linear* computational load (of order T), which is superior to a straight-forward implementation of the explicit form (C.6).

where $\mathbf{p}_{i,j} = p_i, p_{i+1}, \dots, p_j$. We can now define the *subproblem* (i, j) as the problem of minimizing $\mathcal{O}_{i,j}(\mathbf{p}_{i,j})$, simultaneously for every RBPSR, and subject to the monotonicity constraint $0 \leq p_i \leq p_{i+1} \leq \cdots \leq p_j \leq 1$. In what follows, we shall use the following notational conventions:

- The subproblem $(1, T)$ is equivalent to the original problem.
- We shall denote a subproblem solution, $\mathbf{p}_{i,j}$, as *feasible* when the monotonicity constraint is met and *non-feasible* otherwise.
- By *subproblem solution* we mean just a sequence $\mathbf{p}_{i,j}$, feasible or not, such that $p_i, p_{i+1}, \dots, p_j \in [0, 1]$.
- Since any subproblem is isomorphic to the original problem, lemma 1 also shows that if[4] it has a feasible minimizing solution for every RBPSR, then that solution must be unique. Hence, by *the optimal subproblem solution*, we mean the unique feasible solution that minimizes $\mathcal{O}_{i,j}(\cdot)$, for every RBPSR.
- By a *partitioning* of the problem $(1, T)$ into a set, $\mathcal{S}$, of adjacent, non-overlapping subproblems, we mean that every index occurs exactly once in all of the subproblems, so that:

$$\mathcal{O}_{1,T}(\mathbf{p}_{1,T}) = \sum_{(i,j)\in\mathcal{S}} \mathcal{O}_{i,j}(\mathbf{p}_{i,j})\,. \tag{C.9}$$

Our first important step is to show with theorem 2, proved via lemmas 2 and 3, how the optimal total solution may be constituted from optimal subproblem solutions:

Lemma 2. *For a given RBPSR and for a given partitioning, $\mathcal{S}$, of $(1, T)$ into subproblems, let:*

(i) $\mathbf{p}^*_{1,T} = p^*_1, p^*_2, \dots, p^*_T$ *be a feasible solution to the whole problem, with minimum total objective* $\mathcal{O}_{1,T}(\mathbf{p}^*_{1,T})$*; and*

(ii) for every subproblem $(i, j) \in \mathcal{S}$*, let* $\mathbf{q}^*_{i,j} = q^*_i, q^*_{i+1}, \dots, q^*_j$ *denote a feasible subproblem solution with minimum partial objective* $\mathcal{O}_{i,j}(\mathbf{q}^*_{i,j})$*; and*

(iii) $\mathbf{q}^*_{1,T} = q^*_1, q^*_2, \dots, q^*_T$ *denote the concatenation of the all subproblem solutions* $\mathbf{q}^*_{i,j}$*, in the correct order, to form a (not necessarily feasible) solution to the whole problem* $(1, T)$*,*

[4]The object of this whole exercise is to prove that the optimal solution exists for every subproblem and is given by the PAV algorithm, but until we have proved this, we cannot assume that the optimal solution exists for every subproblem.

then

$$\mathcal{O}_{1,T}(\mathbf{q}^*_{1,T}) = \sum_{(i,j)\in\mathcal{S}} \mathcal{O}_{i,j}(\mathbf{q}^*_{i,j}) \leq \sum_{(i,j)\in\mathcal{S}} \mathcal{O}_{i,j}(\mathbf{p}^*_{i,j}) = \mathcal{O}_{1,T}(\mathbf{p}^*_{1,T})\,. \tag{C.10}$$

Proof. Follows by recalling equation (C.9) and by noting that for every (i,j), $\mathcal{O}_{i,j}(\mathbf{q}^*_{i,j}) \leq \mathcal{O}_{i,j}(\mathbf{p}^*_{i,j})$, because (except at $i = 1$ and $j = T$) minimization of the RHS is subject to the extra constraints $p^*_{i-1} \leq p^*_i$ and $p^*_j \leq p^*_{j+1}$. ■

Lemma 3. *For a given RBPSR and for a given partitioning, $\mathcal{S}$, of $(1,T)$ into subproblems, let $\mathbf{p}^*_{1,T} = p^*_1, p^*_2, \ldots, p^*_T$ be a feasible solution to the whole problem, with minimum total objective $\mathcal{O}_{1,T}(\mathbf{p}^*_{1,T})$; and let $\mathbf{q}_{1,T} = q_1, q_2, \ldots, q_T$ be any feasible solution to the whole problem, with total objective $\mathcal{O}_{1,T}(\mathbf{q}_{1,T})$. Then*

$$\mathcal{O}_{1,T}(\mathbf{q}_{1,T}) = \sum_{(i,j)\in\mathcal{S}} \mathcal{O}_{i,j}(\mathbf{q}_{i,j}) \geq \mathcal{O}_{1,T}(\mathbf{p}^*_{1,T})\,. \tag{C.11}$$

Proof. Follows directly from equation (C.9) and the premise. ■

Theorem 2. *Let $\mathbf{q}^*_{1,T} = q^*_1, q^*_2, \ldots, q^*_T$ be a* feasible *solution for $(1,T)$ and let $\mathcal{S}$ be a partitioning of $(1,T)$ into subproblems, such that for every $(i,j) \in \mathcal{S}$, the subsequence $\mathbf{q}^*_{i,j} = q^*_i, q^*_{i+1}, \ldots, q^*_j$ is the optimal solution to subproblem (i,j), then $\mathbf{q}^*_{1,T}$ is the optimal solution to the whole problem $(1,T)$.*

Proof. The premises make lemmas 2 and 3 applicable, for every RBPSR. Since both inequalities (C.10) and (C.11) are satisfied, $\mathcal{O}_{1,T}(\mathbf{q}^*_{1,T}) = \mathcal{O}_{1,T}(\mathbf{p}^*_{1,T})$, where $\mathbf{p}^*_{1,T}$ is an optimal solution for each RBPSR. Hence $\mathbf{q}^*_{1,T}$ is optimal for every RBPSR and is by lemma 1 the unique optimal solution. ■

C.3 Constant subproblem solutions (theorem 3)

In what follows, constant subproblem solutions will be of central importance. A solution $\mathbf{p}_{i,j}$ is constant if $p_i = p_{i+1} = \cdots = p_j = q$, for some $0 \leq q \leq 1$. In this case, we use the short-hand notation $\mathcal{O}_{i,j}(q) = \mathcal{O}_{i,j}(\mathbf{p}_{i,j})$ to denote the subproblem objective, and this may be expressed as:

$$\begin{aligned}\mathcal{O}_{i,j}(q) = \mathcal{O}_{i,j}(\mathbf{p}_{i,j}) &= \sum_{t=i}^{j} \mathsf{v}(\ell_t) C^*_\rho(q|\ell_t) \\ &= m v_1 C^*_\rho(q|\theta_1) + n v_2 C^*_\rho(q|\theta_2)\end{aligned} \tag{C.12}$$

where m is the number of θ_1-labels and n the number of θ_2-labels. Note:

- A constant subproblem solution is always *feasible.*

- If it exists, the optimal solution to an arbitrary subproblem may or may not be constant.

Whether optimal or not, it is important to examine the behaviour of subproblem solutions that are constrained to be constant. This behaviour is governed by the quasiconvex[5] properties of $\mathcal{O}_{i,j}(q)$ as summarized in the following lemma:

Lemma 4. *Let $r_{i,j} = \frac{v_1 m}{v_1 m + v_2 n}$, where m is the number of θ_1-labels and n the number of θ_2-labels in the subsequence (i,j), and let $\mathcal{O}_{i,j}(q) = mv_1 C^*_\rho(q|\theta_1) + nv_2 C^*_\rho(q|\theta_2)$ be the objective for the constant subproblem solution, $p_i = p_{i+1} = \cdots = p_j = q$, then the following properties hold, where C^*_ρ is* any *RBPSR, and where we also note the specialization for strict RBPSRs:*

1. *If $q \le q' \le r_{i,j}$, then $\mathcal{O}_{i,j}(q) \ge \mathcal{O}_{i,j}(q') \ge \mathcal{O}_{i,j}(r_{i,j})$.*

   strict case: *If $q < q' \le r_{i,j}$, then $\mathcal{O}_{i,j}(q) > \mathcal{O}_{i,j}(q')$.*

2. *If $q' \ge q \ge r_{i,j}$, then $\mathcal{O}_{i,j}(q') \ge \mathcal{O}_{i,j}(q) \ge \mathcal{O}_{i,j}(r_{i,j})$.*

   strict case: *If $q' > q \ge r_{i,j}$, then $\mathcal{O}_{i,j}(q') > \mathcal{O}_{i,j}(q)$.*

3. $\min_q \mathcal{O}_{i,j}(q) = \mathcal{O}_{i,j}(r_{i,j})$,

   strict case: $q = r_{i,j}$ *is the unique minimum.*

   This is just the defining property of a (strict) binary proper scoring rule.

Proof. For convenience in this proof, we drop the subscripts i,j, letting $r = r_{i,j} = \frac{mv_1}{mv_1 + nv_2}$. The expected value of $C^*_\rho(q|\theta)$ w.r.t. probability r is:

$$\begin{aligned} e(q) &= \mathbb{E}_{\theta|r}\{C^*_\rho(q|\theta)\} = \tfrac{1}{mv_1+nv_2}\mathcal{O}_{i,j}(q) \\ &= rC^*_\rho(q|\theta_1) + (1-r)C^*_\rho(q|\theta_2)\,. \end{aligned} \tag{C.13}$$

Clearly, if the above properties hold for $e(q)$, then they will also hold for $\mathcal{O}_{i,j}(q)$. We prove these properties for $e(q)$ by letting $q \le q'$ and by examining the sign of $\Delta_e = e(q') - e(q)$, which by equation (C.1) can be expressed, for $q < q'$, as:

$$\Delta_e = \int_q^{q'} (\eta - r)\frac{\rho(\eta)}{\eta(1-\eta)}\,d\eta\,. \tag{C.14}$$

Properties 1,2 and 3 now follow from the following observations:

- If $q' = q$, then $\Delta_e = 0$.

[5]A real-valued function $f(p)$, defined on a real interval, is *quasiconvex*, if every sublevel set of the form $\{p|f(p) < a\}$ is convex, in this case, a real interval [91]. Lemma 4 shows that $\mathcal{O}_{i,j}(q)$ is *quasiconvex*.

- Since $\rho(\eta) \geq 0$, for $0 \leq \eta \leq 1$, when $q < q'$, the sign of the integrand and therefore of Δ_e depends solely on the sign of $(\eta - r)$, giving:

$$\begin{aligned}\Delta_e &\geq 0, \text{ if } r \leq q < q' \\ \Delta_e &\leq 0, \text{ if } q < q' \leq r\,.\end{aligned}$$

- We say $\rho(\eta) > 0$ *almost everywhere*, if, for any $0 \leq q < q' \leq 1$, we have $|\Delta_e| > 0$. In this case, the RBPSR is denoted *strict* and we have:

$$\begin{aligned}\Delta_e &> 0, \text{ if } r \leq q < q' \\ \Delta_e &< 0, \text{ if } q < q' \leq r\,.\end{aligned}$$

■

For now, we need only the proper scoring rule definition (property 3) to proceed. We use the other properties later. The optimal constant subproblem solution is characterized in the following theorem:

Theorem 3. *If the optimal solution to subproblem (i, j) is constant, then:*

1. *The constant is $r_{i,j}$. This follows directly from property 3 of lemma 4.*
2. *For any index k, such that $i \leq k \leq j$, the following are both true:*

   a) $r_{i,k} \geq r_{i,j}$

   b) $r_{k,j} \leq r_{i,j}$

   where $r_{i,k}$ and $r_{k,j}$ are defined in a similar way to $r_{i,j}$, but for the subproblems (i, k) and (k, j).

   Proof. We use contradiction: If the negation, $r_{i,k} < r_{i,j}$, of property 2a were true, then the non-constant solution $p_i = \cdots = p_k = r_{i,k} < p_{k+1} = \cdots = p_j = r_{i,j}$ would be feasible and (by property 3 of lemma 4) would have lower objective, $\mathcal{O}_{i,k}(r_{i,k}) + \mathcal{O}_{k+1,j}(r_{i,j})$, for any strict RBPSR, than that of the constant solution, $\mathcal{O}_{i,k}(r_{i,j}) + \mathcal{O}_{k+1,j}(r_{i,j})$. This contradicts the premise that the optimal solution is constant, so that 2a must be true. Property 2b is proved similarly. ■

C.4 Pooling adjacent constant solutions (theorem 4)

This section shows (using lemmas 5 and 6 to prove theorem 4) when and how optimal constant subproblem solutions may be assembled by pooling smaller adjacent constant solutions:

Lemma 5. *Given a subproblem (i, j), for which the optimal solution is constant (at $r_{i,j}$), we can form the* augmented subproblem, *with the additional constraint that the solution at j must satisfy $p_j \leq \alpha$, for some $0 \leq \alpha < r_{i,j}$. That is, the solution to the augmented subproblem must satisfy $0 \leq p_i \leq p_{i+1} \leq \cdots \leq p_j \leq \alpha < r_{i,j}$. Then the augmented subproblem solution is optimized, for every RBPSR, by the constant solution $p_i = p_{i+1} = \cdots = p_j = \alpha$.*

Proof. Feasible solutions to the augmented subproblem must satisfy either (i) $p_i = \cdots = p_j = \alpha$, or (ii) $p_i < \alpha$. We need to show that there is no feasible solution of type (ii), which has a lower objective value, for any RBPSR, than solution (i).

For a given solution, let k be an index such that $i \leq k \leq j$ and $p_i = p_{i+1} = \cdots = p_k$. By combining the premises of this lemma with property 2a of theorem 3, we find: $p_i = \cdots = p_k \leq \alpha < r_{i,j} \leq r_{i,k}$, or more succinctly: $p_i = \cdots = p_k \leq \alpha < r_{i,k}$. Now the monotonicity property 1 of lemma 4 shows that the value of $p_i = \cdots = p_k$, which is optimal for all RBPSRs must be as large as allowed by the constraints. This means if we start at $k = i$, then p_i is optimized at the constraint $p_i = p_{i+1}$. Next we set $k = i + 1$ to see that $p_i = p_{i+1}$ is optimized at the next constraint $p_i = p_{i+1} = p_{i+2}$. We keep incrementing k, until we find the optimum for the augmented subproblem at the constant solution $p_i = \cdots = p_j = \alpha$. ■

Lemma 6. *Given a subproblem (i, j), for which the optimal solution is constant (at $r_{i,j}$), we can form the* augmented subproblem, *with the additional constraint that the solution at i must satisfy $\alpha \leq p_i$, for some $r_{i,j} \leq \alpha \leq 1$. That is, the solution to the augmented subproblem must satisfy $r_{i,j} < \alpha \leq p_i \leq p_{i+1} \leq \cdots \leq p_j \leq 1$. Then the augmented subproblem solution is optimized, for every RBPSR, by the constant solution $p_i = p_{i+1} = \cdots = p_j = \alpha$.*

Proof. The proof is similar to that of lemma 5, but here we invoke property 2b of theorem 3, to find: $r_{k,j} < \alpha \leq p_k = \cdots = p_j$ and we use the monotonicity property 2 of lemma 4 to show that the value of $p_k = \cdots = p_j$, which is optimal for all RBPSRs, must be as small as allowed by the constraints. ■

Theorem 4. *Given indices $i \leq k \leq j$ such that the optimal subproblem solutions for the two* adjacent *subproblems, (i, k) and $(k + 1, j)$, are constant and therefore (by theorem 3) have the respective values $r_{i,k}$ and $r_{k+1,j}$, then, whenever $r_{i,k} \geq r_{k+1,j}$, the optimal solution for the pooled subproblem (i, j) is also constant, and has the value $r_{i,j}$.*

Proof. First consider the case $r_{i,k} = r_{k+1,j}$. Since this forms a constant solution to subproblem (i, j), by theorem 3, the optimal solution is $r_{i,j}$.

Next consider $r_{i,k} > r_{k+1,j}$. The solution $p_i = \cdots = p_k = r_{i,k} > p_{k+1} = \cdots = p_j = r_{k+1,j}$ is not feasible. A feasible solution must obey $p_k \leq \alpha \leq p_{k+1}$, for some α. There are three possibilities for the value of α: (i) $\alpha \leq r_{k+1,j}$; (ii) $r_{k+1,j} < \alpha < r_{i,k}$; or (iii) $r_{i,k} \leq \alpha$. We examine each in turn:

(i) If $\alpha \leq r_{k+1,j} < r_{i,k}$, then the left subproblem (i,k) is augmented by the constraint $\alpha < r_{i,k}$, so that lemma 5 applies and it is optimized at the constant solution α, while the right subproblem $(k+1,j)$ is not further constrained and is still optimized at $r_{k+1,j}$. We can now optimize the total solution for (i,j) by adjusting α: By the monotonicity property 1 of lemma 4, the left subproblem objective and therefore also the total objective for (i,j) is optimized at the upper boundary $\alpha = r_{k+1,j}$. In other words, in this case, the optimum for subproblem (i,j) is a constant solution.

(ii) If $r_{k+1,j} < \alpha < r_{i,k}$, then lemma 5 applies to the left subproblem and lemma 6 applies to the right subproblem, so that both subproblems and therefore also the total objective for (i,j) are all optimized at α. In this case also we have a constant solution for (i,j).

(iii) If $r_{k+1,j} < r_{i,k} \leq \alpha$, then the right subproblem is augmented while the left subproblem is not further constrained. We can now use lemma 6 and property 2 of lemma 4, in a similar way to case (i) to show that in this case also, the optimum solution is constant.

Since the three cases exhaust the possibilities for choosing α, the optimal solution is indeed constant and by theorem 3 the optimum is at $r_{i,j}$. ∎

C.5 Total solution: PAV algorithm

We can now use theorems 2, 3 and 4 to construct a version of the *pool-adjacent-violators* (PAV) algorithm for solving problem $(1,T)$. The strategy is to satisfy the conditions for theorem 2, by starting with optimal constant subproblem solutions of length 1 and then to iteratively combine them via theorem 4, into longer optimal constant solutions until the total solution is feasible:

input:

labels, $\ell_1, \ell_2, \dots, \ell_T \in \{\theta_1, \theta_2\}$.

weights, $v_1, v_2 > 0$.

variables:

$\mathcal{S}$, a partitioning of problem $(1,T)$ into adjacent, non-overlapping subproblems.

$\mathbf{q}^*_{1,T} = q^*_1, q^*_2, \dots, q^*_T$, a tentative (not necessarily feasible) solution for problem $(1,T)$.

loop invariant:

For every subproblem $(i,j) \in \mathcal{S}$:

(i) The optimal subproblem solution is constant.

(ii) The partial solution $\mathbf{q}^*_{i,j} = q^*_i, q^*_{i+1}, \dots, q^*_j$ is equal to the optimal subproblem solution, i.e. constant, with value $r_{i,j}$ (by theorem 3).

initialization:

Let $\mathcal{S}$ be the finest partitioning into subproblems, so that there are T subproblems, each spanning a single index. Clearly every subproblem (i,i) has a constant solution, optimized at $q^*_i = r_{i,i}$, which is 1, if $\ell_t = \theta_1$, or 0, if $\ell_t = \theta_2$. This initial solution $\mathbf{q}^*_{1,T}$ respects the loop invariant, but is most probably not feasible.

iteration:

While $\mathbf{q}^*_{1,T}$ is not feasible:

1. Find any pair of adjacent subproblems, $(i,k), (k+1,j) \in \mathcal{S}$, for which the solutions are equal or violate monotonicity: $r_{i,k} \geq r_{k+1,j}$.
2. Pool (i,k) and $(k+1,j)$ into one subproblem (i,j), by adjusting $\mathcal{S}$ and by assigning the constant solution $r_{i,j}$ to $\mathbf{q}^*_{i,j}$, which by theorem 4 is optimal for (i,j), thus maintaining the loop invariant.

termination:

Clearly the iteration must terminate after at most $T-1$ pooling steps, at which time $\mathbf{q}^*_{1,T}$ is now feasible and is still optimal for every subproblem. By theorem 2, $\mathbf{q}^*_{1,T}$ is then the unique optimal solution to problem $(1,T)$. ∎

C.6 The PAV-LLR algorithm (theorem 5)

The PAV algorithm as presented above finds solutions in the form of *probabilities.* However, in this work, when working with binary problems, we are more interested in assigning *log-likelihood-ratios.* In particular, we are interested in solving the following problem:

There is given:

- An RBPSR C^*_ρ
- Prior log-odds π, where $-\infty < \pi < \infty$.
- Labels, $\ell_1, \ell_2, \dots, \ell_T \in \{\theta_1, \theta_2\}$.

There is required a solution, $\mathbf{w}_{1,T} = w_1, w_2, \ldots, w_T$, that minimizes the following objective:

$$
\begin{aligned}
\mathcal{O}_{1,T}(\mathbf{w}_{1,T}) &= \sum_{t=1}^{T} \mathsf{v}(\ell_t) C^*_\rho(p_t|\ell_t)\,, && \text{(C.15)}\\
p_t &= \sigma(w_t + \pi)\,, && \text{(C.16)}\\
v_1 &= \mathsf{v}(\theta_1) = \frac{\sigma(\pi)}{T_1}\,, && \text{(C.17)}\\
v_2 &= \mathsf{v}(\theta_2) = \frac{\sigma(-\pi)}{T_2} && \text{(C.18)}
\end{aligned}
$$

where T_1 and T_2 are the respective numbers of θ_1-labels and θ_2-labels in $\ell_1, \ell_2, \ldots, \ell_T$; and where $\sigma()$ is the inverse of the logit function. The minimization is subject to the monotonicity constraint:

$$
-\infty \leq w_1 \leq w_2 \leq \cdots \leq w_T \leq \infty\,. \tag{C.19}
$$

This problem is solved by first finding the probabilities $p_1, p_2, \ldots, p_T$ via the PAV algorithm and then inverting (C.16) to find $w_t = \operatorname{logit}(p_t) - \pi$. We already know that the solution is independent of the RBPSR, but remarkably, it is *also* independent of the prior π. This is shown in the following theorem:

Theorem 5. *Let* $\mathbf{p}_{1,T} = \operatorname{PAV}\big((\ell_1, \ell_2, \ldots, \ell_T), (v_1, v_2)\big)$ *denote an application of the PAV algorithm, then the problem of minimizing objective* (C.15)*, subject to monotonicity constraint* (C.19) *has the unique solution:*

$$
\mathbf{w}_{1,T} = \operatorname{logit} \operatorname{PAV}\big((\ell_1, \ell_2, \ldots, \ell_T), (1,1)\big) - \operatorname{logit} \frac{T_1}{T}\,. \tag{C.20}
$$

This solution is simultaneously optimal for every RBPSR, C^*_ρ, and any prior log-odds, $-\infty < \pi < \infty$.

Proof. By the properties of the PAV as proved in section C.5 and since logit is a strictly monotonic rising bijection, it is clear that for all RBPSRs and for a given π, this minimization is solved as

$$
\mathbf{w}_{1,T} = \operatorname{logit} \operatorname{PAV}\big((\ell_1, \ell_2, \ldots, \ell_T), (v_1, v_2)\big) - \pi \tag{C.21}
$$

where π determines v_1 and v_2 via (C.17) and (C.18). By lemma 1, we can write component t of this solution, in closed form:

$$
\begin{aligned}
w_t &= \operatorname{logit}\left(\max_{1\leq i\leq t}\ \min_{t\leq j\leq T} r_{i,j}\right) - \pi \\
&= \max_{1\leq i\leq t}\ \min_{t\leq j\leq T} \operatorname{logit} r_{i,j} - \pi\,.
\end{aligned} \tag{C.22}
$$

Now observe that:

$$\begin{aligned}\operatorname{logit} r_{i,j} &= \operatorname{logit} \frac{v_1 m_{i,j}}{v_1 m_{i,j} + v_2 n_{i,j}} \\ &= \operatorname{logit} \frac{m_{i,j}}{m_{i,j} + n_{i,j}} - \operatorname{logit} \frac{T_1}{T} + \pi\end{aligned} \tag{C.23}$$

which shows that w_t is independent of π. Now the prior may be chosen conveniently equal to the label proportion, $\pi = \operatorname{logit} \frac{T_1}{T}$, to give an un-weighted PAV, with $v_1 = v_2 = 1$. ■

Appendix D

Proof of multi-class logarithmic cost integral

Here we prove (8.13), which we repeat for convenience:

$$C_{\log}(\mathbf{p}|\theta_i) = -\log(p_i) = \int_{\mathbb{P}_N} \Gamma(N) C^*_{\boldsymbol{\eta}}(\mathbf{p}|\theta_i)\,\mathbf{d}\boldsymbol{\eta} \tag{D.1}$$

where $\Gamma(N) = 2 \times 3 \times \cdots \times (N-1)$ and $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_N)$. We start by noting that when θ_i is true, the cost of making a Bayes decision,[1] given cost function $C_{\boldsymbol{\eta}}$ and a probability distribution $\mathbf{p} = (p_1, \ldots, p_N)$ is:

$$C^*_{\boldsymbol{\eta}}(\mathbf{p}|\theta_i) = \tfrac{1}{N-1} \begin{cases} 0, & \text{if } \frac{p_i}{\eta_i} \geq \max_{j=1}^N \frac{p_j}{\eta_j}, \\ \frac{1}{\eta_i}, & \text{otherwise.} \end{cases} \tag{D.2}$$

The integral (D.1) can now be formed with integrand $\frac{\Gamma(N-1)}{\eta_i}$ and with boundaries that need to respect not only the simplex $\mathbb{P}_N$, but also the *negation* of the condition $\frac{p_i}{\eta_i} \geq \max_{j=1}^N \frac{p_j}{\eta_j}$. The joint boundaries are uncomfortable to work with, so we transform the simplex to $\mathbb{R}^{N-1}$, conveniently mapping its boundaries to infinity. What follows is notationally easier if we let, without loss of generality, $i = N$. We perform a change of variables by letting:[2]

$$\mathbf{x} = (x_1, \ldots, x_{N-1}) = \Big(\log\frac{\eta_1}{\eta_N}, \ldots, \log\frac{\eta_{N-1}}{\eta_N}\Big) \tag{D.3}$$

$$\mathbf{y} = (y_1, \ldots, y_{N-1}) = \Big(\log\frac{p_1}{p_N}, \ldots, \log\frac{p_{N-1}}{p_N}\Big)\,. \tag{D.4}$$

The simplex boundaries are now trivial to handle, because the simplex is mapped to the whole of $\mathbb{R}^{N-1}$; while the other condition, $\frac{p_i}{\eta_i} \geq \max_{j=1}^N \frac{p_j}{\eta_j}$,

[1] with tie-breaker $b\big(\{\theta_1, \theta_2\}\big) = \theta_1$

[2] In the general case, we choose η_i and p_i as denominators and omit the components $x_i = 0$ and $y_i = 0$ from the vectors.

is mapped to the *negation of the conjunction* of the $N-1$ conditions: $x_j \geq y_j, j \neq i$. The inverse transform gives:

$$\eta_N = \frac{1}{1+\sum_{k=1}^{N-1}\exp(x_k)} \qquad \text{and} \qquad \eta_j = \exp(x_j)\eta_N \tag{D.5}$$

where $j < N$ and the Jacobian determinant gives:

$$\mathbf{d}\boldsymbol{\eta} = \prod_{k=1}^{N} \eta_k \, \mathbf{dx}\,. \tag{D.6}$$

We can now rewrite the integral (for the case $i = N$) as:

$$\begin{aligned}
&\int_{\mathbb{P}_N} \Gamma(N) C^*_{\boldsymbol{\eta}}(\mathbf{p}|\theta_N)\,\mathbf{d}\boldsymbol{\eta} \\
&= \int_{-\infty}^{y_1}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \mathcal{I}(\mathbf{x})\,dx_{N-1}\,dx_{N-2}\cdots dx_1 \\
&+ \int_{y_1}^{\infty}\int_{-\infty}^{y_2}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \mathcal{I}(\mathbf{x})\,dx_{N-1}\,dx_{N-2}\cdots dx_1 \\
&+ \int_{y_1}^{\infty}\int_{y_2}^{\infty}\int_{-\infty}^{y_3}\cdots\int_{-\infty}^{\infty} \mathcal{I}(\mathbf{x})\,dx_{N-1}\,dx_{N-2}\cdots dx_1 \\
&+\cdots \\
&+ \int_{y_1}^{\infty}\int_{y_2}^{\infty}\int_{y_3}^{\infty}\cdots\int_{-\infty}^{y_{N-1}} \mathcal{I}(\mathbf{x})\,dx_{N-1}\,dx_{N-2}\cdots dx_1
\end{aligned} \tag{D.7}$$

where

$$\mathcal{I}(\mathbf{x}) = \frac{\Gamma(N-1)}{\eta_N}\prod_{k=1}^{N}\eta_k = \Gamma(N-1)\frac{e^{\sum_{k=1}^{N-1}x_k}}{\left(1+\sum_{k=1}^{N-1}e^{x_k}\right)^{N-1}} \tag{D.8}$$

and where the multiple terms of (D.7) are the result of applying the boundaries, one dimension at a time, to exclude from the integral a semi-infinite rectangular region. Each term of nested integrals can be solved from the inside out, by using the antiderivatives $\int \frac{e^x}{(k+e^x)^n}\,dx = \frac{-1}{n-1}\frac{1}{(k+e^x)^{n-1}}$, for $n = 2, 3, 4, \ldots$; or $\int \frac{e^x}{1+e^x}\,dx = \log(1+e^x)$. After some simplification, the terms of (D.7) can be written as:

$$
\begin{aligned}
&\int_{\mathbb{P}_N} \Gamma(N) C^*_{\boldsymbol{\eta}}(\mathbf{p}|\theta_N)\,\mathbf{d}\boldsymbol{\eta} \\
&= \log(1+e^{y_1}) \\
&+ \log(1+e^{y_1}+e^{y_2}) - \log(1+e^{y_1}) \\
&+ \log(1+e^{y_1}+e^{y_2}+e^{y_3}) - \log(1+e^{y_1}+e^{y_2}) \\
&+ \cdots \\
&+ \log(1+\textstyle\sum_{j=1}^{N-1} e^{y_j}) - \log(1+\textstyle\sum_{j=1}^{N-2} e^{y_j}) \\
&= \log(1+\textstyle\sum_{j=1}^{N-1} e^{y_j}) \\
&= -\log(p_N)
\end{aligned}
\tag{D.9}
$$

where the inverse transform (D.5) was applied in the last line. This completes the proof for $i = N$. The cases for $i < N$ follow by symmetry.■

D.1 Alternative integral representation

For the two-class case, we found an alternative representation of the logarithmic evaluation criterion, by integrating error-rate as function of the prior log odds: $\int_{-\infty}^{\infty} \mathcal{E}_{\text{err}}(\mathcal{W}|\operatorname{logit}^{-1} h)\,dh = \mathcal{E}_{\log}(\mathcal{W}|\bar{\pi})$. To generalize this to the multi-class case, one needs a non-uniform weighting over an $N-1$ dimensional logarithmic parametrization of the prior. An example of this weighting for $N = 3$ is shown in figure D.1.

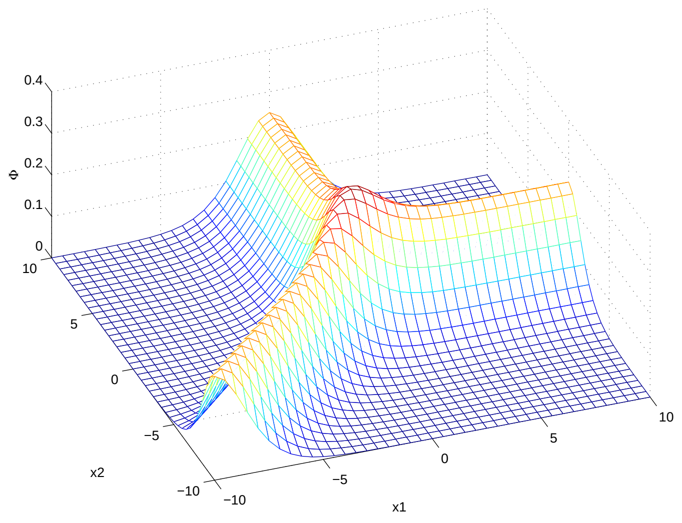

Figure D.1: Weighting function for error-rate as a function of the prior to integrate to the logarithmic evaluation criterion. The axes are $x_1 = \log(\pi_1) - \log(\pi_3)$ and $x_2 = \log(\pi_2) - \log(\pi_3)$, where $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3)$, is the prior.

Bibliography

[1] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 2007.

[2] John C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large-Margin Classifiers*, Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, Eds., pp. 61–74. MIT Press, 1999.

[3] Glenn W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, Jan. 1950.

[4] Irving J Good, "Rational decisions," *Journal of the Royal Statistical Society, Series B*, vol. 14, no. 1, pp. 107–114, 1952.

[5] Morris H. DeGroot and Stephen E. Fienberg, "The comparison and evaluation of forecasters," *The Statistician*, vol. 32, pp. 14–22, 1983.

[6] José M. Bernardo and Adrian F. M. Smith, *Bayesian Theory*, John Wiley & Sons, 1994.

[7] Edwin T. Jaynes, *Probability Theory: The Logic of Science*, Cambridge University Press, 2003.

[8] Tilmann Gneiting and Adrian E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, Mar. 2007.

[9] David J. C. MacKay, *Bayesian Methods for Adaptive Models*, Ph.D. thesis, California Institute of Technology, Dec. 1991.

[10] David J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 1st edition, 2003.

[11] Ji Zhu and Trevor Hastie, "Kernel logistic regression and the import vector machine," *Journal of Computational and Graphical Statistics*, vol. 14, no. 1, pp. 185–205, 2005.

[12] Phil C. Woodland and Daniel Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 25–47, Jan. 2002.

[13] Andreas Buja, Werner Stuetzle, and Yi Shen, "Loss functions for binary class probability estimation and classification: Structure and applications," Technical report, Statistics Department, The Wharton School, University of Pennsylvania, Nov. 2005, Online: `www.wharton.upenn.edu/buja`.

[14] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[15] Georg Stemmer, Stefan Steidl, Elmar Nöth, Heinrich Niemann, and Anton Batliner, "Comparison and combination of confidence measures," in *Text, Speech and Dialogue, 5th International Conference, Brno, Czech Republic, 2002*, Petr Sojka, Ivan Kopeček, and Karel Pala, Eds. Sept. 2006, Lecture Notes in Computer Science, pp. 561–582, Springer.

[16] Niko Brümmer, "Application-independent evaluation of speaker detection," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Toledo, Spain, June 2004, pp. 33–40.

[17] Niko Brümmer and Johan A. du Preez, "Application-independent evaluation of speaker detection," *Computer Speech and Language*, vol. 20, no. 2–3, pp. 230–275, 2006.

[18] David A. van Leeuwen and Niko Brümmer, "An introduction to application-independent evaluation of speaker recognition systems," in *Speaker Classification I: Fundamentals, Features, and Methods (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence)*, Christian Müller, Ed., pp. 330–353. Springer, 1st edition, 2007.

[19] Niko Brümmer and David A. van Leeuwen, "On calibration of language recognition scores," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, June 2006.

[20] Niko Brümmer and Edward de Villiers, "The speaker partitioning problem," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[21] Niko Brümmer, Lukáš Burget, Jan "Honza" Černocký, Ondřej Glembek, František Grézl, Martin Karafiát, David A. van Leeuwen, Pavel Matějka, Petr Schwarz, and Albert Strasheim, "Fusion of heterogenous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2072–2084, Sept. 2007.

[22] David A. van Leeuwen and Niko Brümmer, "Channel-dependent GMM and multi-class logistic regression," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, June 2006.

[23] Niko Brümmer, Albert Strasheim, Valiantsina Hubeika, Pavel Matějka, Lukáš Burget, and Ondřej Glembek, "Discriminative acoustic language recognition via channel-compensated GMM statistics," in *Proceedings of Interspeech*, Brighton, UK, Sept. 2009.

[24] Zdeněk Jančík, Oldřich Plchot, Niko Brümmer, Lukáš Burget, Ondřej Glembek, Valiantsina Hubeika, Martin Karafiát, Pavel Matějka, Tomáš Mikolov, Albert Strasheim, and Jan "Honza" Černocký, "Data selection and calibration issues in automatic language recognition — investigation with BUT-AGNITIO NIST LRE 2009 system," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[25] Alvin F. Martin and Craig S. Greenberg, "NIST 2008 speaker recognition evaluation: Performance across telephone and room microphone channels," in *Proceedings of Interspeech*, Brighton, UK, Sept. 2009, pp. 2579–2582.

[26] Alvin Martin and Craig Greenberg, "The 2009 NIST language recognition evaluation," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[27] Claude E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–625, 1948.

[28] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, Wiley Interscience, 1991.

[29] Daniel Ramos-Castro, *Forensic Evaluation of the Evidence Using Automatic Speaker Recognition Systems*, Ph.D. thesis, Universidad Autónoma de Madrid, Madrid, Spain, Nov. 2007.

[30] Daniel Ramos-Castro, Joaquin Gonzalez-Rodriguez, and Javier Ortega-Garcia, "Likelihood ratio calibration in a transparent and testable forensic speaker recognition framework," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, June 2006.

[31] Joaquin Gonzalez-Rodriguez, Andrzej Drygajlo, Daniel Ramos-Castro, Marta Garcia-Gomar, and Javier Ortega-Garcia, "Robust estimation, interpretation and assessment of likelihood ratios in forensic speaker recognition," *Computer Speech and Language*, vol. 20, no. 2–3, pp. 331–355, 2006.

[32] Joaquin Gonzalez-Rodriguez, Phil Rose, Daniel Ramos-Castro, Doroteo Torre-Toledano, and Javier Ortega-Garcia, "Emulating DNA: Rigorous quantification of evidential weight in transparent and testable forensic speaker recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 7, pp. 2104–2115, Sept. 2007.

[33] Daniel Ramos-Castro and Joaquin Gonzalez-Rodriguez, "Evaluation of likelihood ratios based on information theory," *Problems in Forensic Science*, vol. 69, pp. 62–70, 2007.

[34] Daniel Ramos-Castro, Joaquin Gonzalez-Rodriguez, and Jose-Juan Lucena-Molina, "Accuracy assessment methods for likelihood-ratio-based evidence evaluation," in *Proceedings of the 5th European Academy of Forensic Science Conference*, Sept. 2009, Invited contribution.

[35] William M. Campbell, Kevin J. Brady, Joseph P. Campbell, R. Granville, and Douglas A. Reynolds, "Understanding scores in forensic speaker recognition," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, June 2006.

[36] Tharmarajah Thiruvaran, Eliathamby Ambikairajah, and Julien Epps, "FM features for automatic forensic speaker recognition," in *Proceedings of Interspeech*, Brisbane, Australia, Sept. 2008.

[37] Timo Becker, Michael Jessen, Sebastian Alsbach, Franz Broß, and Torsten Meier, "SPES: The BKA forensic automatic voice comparison system," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[38] Grzegorz Zadora and Daniel Ramos, "Evaluation of glass samples for forensic purposes—an application of likelihood ratios and an information-theoretical approach," *Chemometrics and Intelligent Laboratory Systems*, vol. 102, no. 2, pp. 63–83, July 2010.

[39] Mitchell McLaren, Robbie Vogt, Brendan Baker, and Sridha Sridharan, "A comparison of session variability compensation techniques for SVM-based speaker recognition," in *Proceedings of Interspeech*, Antwerp, Belgium, Aug. 2007.

[40] Boštjan Vesnicer and France Mihelič, "The likelihood ratio decision criterion for nuisance attribute projection in GMM speaker verification," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 158, Jan. 2008.

[41] Tomi Kinnunen, Juhani Saastamoinen, Ville Hautamäki, Mikko Vinni, and Pasi Fränti, "Comparative evaluation of maximum a posteriori vector quantization and Gaussian mixture models in speaker verification," *Pattern Recognition Letters*, vol. 30, no. 4, pp. 341–347, Mar. 2009.

[42] Marijn Huijbregts and David van Leeuwen, “The RU submission to the EVALITA’09 “application track” speaker recognition evaluation,” in *Proceedings of EVALITA*, Reggio Emilia, Italy, Dec. 2009, Online: `http://evalita.fbk.eu/proceedings.html`.

[43] Jan Silovský, Petr Červa, and Jindřich Žd’ánský, “Comparison of generative and discriminative approaches for speaker recognition with limited data,” *Radioengineering*, vol. 18, no. 3, pp. 307–316, Sept. 2009.

[44] Emanuele Dalmasso, Fabio Castaldo, Pietro Laface, Daniele Colibro, and Claudio Vair, “Loquendo-Politecnico di Torino’s 2008 NIST Speaker Recognition Evaluation system,” in *Proceedings of ICASSP*, Taipei, Taiwan, Apr. 2009, pp. 4213–4216.

[45] Anthony Larcher, Christophe Lévy, Driss Matrouf, and Jean-Francois Bonastre, “LIA NIST-SRE’10 systems,” in *Proceedings of the NIST-SRE Workshop*, Brno, Czech Republic, June 2010, Online: `http://lia.univ-avignon.fr/fileadmin/documents/.../LIA_system_NIST-SRE.pdf`.

[46] Maider Zamalloa, Luis Javier Rodríguez-Fuentes, Germán Bordel, Mikel Penagarikano, and Juan Pedro Uribe, “Low-latency online speaker tracking on the AMI corpus of meeting conversations,” in *Proceedings of ICASSP*, Dallas, USA, Mar. 2010, pp. 4962–4965.

[47] Donglai Zhu, Haizhou Li, Bin Ma, and Chin-Hui Lee, “Optimizing the performance of spoken language recognition with discriminative training,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 8, Nov. 2008.

[48] Alan McCree, Fred Richardson, Elliot Singer, and Douglas A. Reynolds, “Beyond frame independence: Parametric modelling of time duration in speaker and language recognition,” in *Proceedings of Interspeech*, Brisbane, Australia, Sept. 2008.

[49] Mohamed Faouzi BenZeghiba, Jean-Luc Gauvain, and Lori Lamel, “Gaussian backend design for open-set language detection,” in *Proceedings of ICASSP*, Taipei, Taiwan, Apr. 2009, pp. 4349–4352.

[50] Andreas Stolcke, Murat Akbacak, Luciana Ferrer, Sachin Kajarekar, Colleen Richey, Nicolas Scheffer, and Elizabeth Shriberg, “Improving language recognition with multilingual phone recognition and speaker adaptation transforms,” in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[51] Fabio Castaldo, Daniele Colibro, Sandro Cumani, Emanuele Dalmasso, Pietro Laface, and Claudio Vair, “Loquendo-Politecnico di Torino system

for the 2009 NIST Language Recognition Evaluation," in *Proceedings of ICASSP*, Dallas, Texas, USA, Mar. 2010, pp. 5002–5005.

[52] Marcel Kockmann, Lukáš Burget, and Jan "Honza" Černocký, "Brno University of Technology system for Interspeech 2009 emotion challenge," in *Proceedings of Interspeech*, Brighton, UK, Sept. 2009.

[53] Pierre Dumouchel, Najim Dehak, Yazid Attabi, Réda Dehak, and Narjès Boufaden, "Cepstral and long-term features for emotion recognition," in *Proceedings of Interspeech*, Brighton, UK, Sept. 2009.

[54] Lukáš Burget et al., "Robust speaker recognition over varying channels," in *Johns Hopkins University CLSP Summer Workshop Report*, 2008, Online: `http://www.clsp.jhu.edu/workshops/ws08/documents/jhu_report_main.pdf`.

[55] Guido Aversano, Niko Brümmer, and Mauro Falcone, "EVALITA 2009 speaker identity verification "application" track organizer's report," in *Proceedings of EVALITA*, Reggio Emilia, Italy, Dec. 2009, Online: `http://evalita.fbk.eu/proceedings.html`.

[56] Morris H. DeGroot, *Optimal Statistical Decisions*, McGraw-Hill, 1970.

[57] A. Philip Dawid, "Coherent measures of discrepancy, uncertainty and dependence, with applications to Bayesian predictive experimental design," Technical Report 139, Department of Statistical Science, University College London, Aug. 1998, Online: `http://www.ucl.ac.uk/Stats/research/reports/abs94.html#139`.

[58] George Casella and Roger L. Berger, *Statistical Inference*, chapter 6, Duxbury Press, 2nd edition, 2001.

[59] Evgenii Borisovich Dynkin, "Necessary and sufficient statistics for a family of probability distributions," *Selected Translations in Mathematical Statistics and Probability*, vol. 1, pp. 23–41, 1961.

[60] Robert L. Winkler and Allan H. Murphy, "Good probability assessors," *Journal of Applied Meteorology*, vol. 7, no. 5, pp. 751–758, Oct. 1968.

[61] Steven Roman, *Advanced Linear Algebra*, Graduate Texts in Mathematics. Springer, 2nd edition, 2005.

[62] Tom Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.

[63] Maurice Sion, "On general minimax theorems," *Pacific Journal of Mathematics*, vol. 8, no. 1, pp. 171–176, 1958, Online: `http://projecteuclid.org/euclid.pjm/1103040253`.

[64] George R. Doddington, "Speaker recognition evaluation methodology: a review and perspective," in *Proceedings of RLA2C Workshop: Speaker Recognition and its Commercial and Forensic Applications*, Avignon, France, Apr. 1998, pp. 60–66.

[65] The National Institute of Standards and Technology, "The NIST year 2010 speaker recognition evaluation plan," `http://www.itl.nist.gov/iad/mig//tests/sre/2010/NIST_SRE10_evalplan.r6.pdf`, Apr. 2010.

[66] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki, "The DET curve in assessment of detection task performance," in *Proceedings of the 5th European Conference on Speech Communication and Technology, EUROSPEECH*, Rhodes, Greece, Sept. 1997, pp. 1895–1898.

[67] Milton Abramowitz and Irene Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover Publications, 1972.

[68] Foster J. Provost and Tom Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, no. 3, pp. 203–231, Mar. 2001.

[69] Tom Fawcett and Alexandru Niculescu-Mizil, "PAV and the ROC convex hull," *Machine Learning*, vol. 68, no. 1, pp. 97–106, July 2007.

[70] Ding zhu Du and Panos M. Pardalos, Eds., *Minimax And Applications*, Kluwer, 1st edition, 1995.

[71] Thorsten Joachims, "A support vector method for multivariate performance measures," in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005, pp. 377–384, ACM Press.

[72] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2nd edition, 1992.

[73] Thomas P. Minka, "A comparison of numerical optimizers for logistic regression," Technical report, Department of Statistics, Carnegie Mellon University, Oct. 2003, Online, with (MATLAB code): `http://www.stat.cmu.edu/~minka/papers/logreg`.

[74] Bianca Zadrozny and Charles Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 2002, pp. 694–699.

[75] Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman, “An empirical distribution function for sampling with incomplete information,” *Annals of Mathematical Statistics*, vol. 26, no. 4, pp. 641–647, 1955, Online: `http://projecteuclid.org/euclid.aoms/1177728423`.

[76] Israel Nathan Herstein, *Topics in Algebra*, John Wiley & Sons, 2nd edition, 1975.

[77] Emir Shuford, Arthur Albert, and H. Edward Massengill, “Admissible probability measurement procedures,” *Psychometrika*, vol. 31, no. 2, pp. 125–145, June 1966.

[78] Norman C. Dalkey, “Inductive inference and the maximum entropy principle,” in *Maximum Entropy and Bayesian Methods in Inverse Problems*, C. Ray Smith and Walter T. Grandy, Eds., pp. 351–364. Reidel, Dordrecht, 1st edition, 1985.

[79] Carl-Axel S. Staël von Holstein, “A family of strictly proper scoring rules which are sensitive to distance,” *Journal of Applied Meteorology*, vol. 9, no. 3, pp. 360–364, 1970.

[80] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, Springer, 2nd edition, 2006.

[81] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000.

[82] Alex Solomonoff, Carl Quillen, and William M. Campbell, “Channel compensation for SVM speaker recognition,” in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Toledo, Spain, June 2004.

[83] Andreas Stolcke, Luciana Ferrer, Sachin Kajarekar, Elizabeth Shriberg, and Anand Venkataraman, “MLLR transforms as features in speaker recognition,” in *Proceedings of the 9th European Conference on Speech Communication and Technology, EUROSPEECH*, Lisbon, Portugal, Sept. 2005, pp. 2425–2428.

[84] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel, “Joint factor analysis versus eigenchannels in speaker recognition,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 4, pp. 1435–1447, May 2007.

[85] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, “Front-end factor analysis for speaker verification,” to appear in IEEE Transactions on Audio, Speech and Language Processing, 2010.

[86] Patrick Kenny, “Bayesian speaker verification with heavy tailed priors,” in *Proceedings of the Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[87] David J. Balding, *Weight-of-evidence for Forensic DNA Profiles*, John Wiley & Sons, 2005.

[88] Robert G. Bartle, *The Elements of Real Analysis*, John Wiley & Sons, 2nd edition, 1976.

[89] Ravindra K. Ahuja and James B. Orlin, “Solving the convex ordered set problem with applications to isotone regression,” Working papers SWP#3988, Massachusetts Institute of Technology, Sloan School of Management, Feb. 1998, Online: `http://www.mit.edu/bitstream`.

[90] Michael J. Best, Nilotpal Chakravarti, and Vasant A. Ubhaya, “Minimizing separable convex functions subject to simple chain constraints,” *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 658–672, 1999.

[91] Mordecai Avriel, Walter E. Diewert, Siegfried Schaible, and Israel Zang, *Generalized Concavity*, Plenum Press, 1988.