The Role of **Proper Scoring Rules** in Training and Evaluating Probabilistic Speaker and Language Recognizers

Niko Brümmer

AGNITIO Research, South Africa

Odyssey 2012, Singapore
In speaker and language recognition, I have been working on
- calibration, since 2004
- fusion, since 2005
Haizhou Li suggested that my talk should cover these topics.

I’ll do so by discussing proper scoring rules, the theoretical principle that underlies all of this work.
- Proper scoring rules do work very well for fusion, but you can do fusion in other ways too.
- But in my opinion, proper scoring rules are essential when dealing with calibration.
Proper scoring rules

- have been around since 1950, when the Brier score was proposed for the evaluation of the goodness of probabilistic weather forecasting;
- are well represented in statistics literature—up to the present;
- are less known in pattern recognition / machine learning and speech engineering,
- but if you have ever done generative training (with maximum likelihood), or discriminative training (with cross-entropy) you have implicitly used the logarithmic proper scoring rule!
In the future, we may be seeing more use of proper scoring rules in machine learning:

- A new cleverly crafted scoring rule (Hyvärinen, 2005) provides a solution to training probabilistic models (generative or discriminative) that cannot be normalized, such as the currently popular restricted Boltzmann machines and other energy-based models.
- For recent work see e.g. (Swersky, 2011).
In this talk I will concentrate on my own application of proper scoring rules to speaker and language recognition. By discussing proper scoring rules, I hope to promote better understanding of

- Calibration itself, which I argue should be defined in terms of proper scoring rules.
- Calibration-sensitive training algorithms (e.g. logistic regression)
- Calibration-sensitive evaluation criteria (e.g. $C_{llr}$)
Outline

1. Introduction

2. The problem
   - Probabilistic pattern recognition
   - Likelihood vs posterior
   - Why do we need calibration?
   - Measurement and training

3. Proper scoring rules

4. Proper scoring rule design
Not all pattern recognition needs to be probabilistic,
- You can build an accurate SVM classifier, which outputs hard decisions, without even thinking about probabilities.

In this talk, we are interested in **probabilistic pattern recognizers** that output **probabilities / likelihoods / likelihood ratios**.
- Provided such outputs are reasonably well-calibrated, they are more generally useful than hard decisions.
In machine learning (and also in speech recognition) it is customary to work with probabilistic recognizers that output posterior probabilities.

- But in speaker and language recognition, there are good reasons for the recognizers to output instead class likelihoods.
- If there are only two classes (e.g. speaker verification), then likelihood ratios are most convenient.
Likelihood vs posterior

If prior is given, likelihood and posterior are equivalent:
If one is well-calibrated, so is the other.
Likelihood vs posterior

data
speech / MFCC / i-vector / score/ ...

pattern recognizer

class likelihoods
P(data | class 1)
P(data | class 2)

Bayes' rule

prior distribution
P(class 1)
P(class 2)

posterior distribution
P(class 1 | data)
P(class 2 | data)

class
target / non-target
english / german / ...

This is the recognizer output.
But we measure calibration here.
Almost all tractable probabilistic models are imperfect models of the data.

- Even when such models extract information that could discriminate between classes with high accuracy,
- the probabilistic representation of this information is usually far from optimal.

This is a calibration problem: the discriminating information is available, but misrepresented.

- Calibration analysis can detect such problems and then help to improve the representation.
Calibration
Measurement vs transformation

Calibration has two meanings:

- **Measure of goodness** of probabilistic recognizer output
- **Transformation** of recognizer outputs to improve the above measure of goodness.
Calibration transformation

- data
- pattern recognizer
- raw model likelihoods (scores)
- calibration transform
- calibrated likelihoods
- user

probabilistic model: $P(\text{data}, \text{class})$

probabilistic model: $P(\text{scores}, \text{class})$
Calibration measurement

Bayes' rule

Pattern recognizer

Raw likelihoods (scores)

calibration
transform

calibrated
likelihoods

user

Bayes' rule

Calibration measurement

Calibration-sensitive measure of goodness

Proper scoring rule

True class labels
Bayes' rule

Pattern recognizer

Raw likelihoods (scores)

Calibration transform

Calibrated likelihoods

User

Bayes' rule

Prior

Posterior

Proper scoring rule

True class labels

Training algorithm (optimizer)

Calibration parameters

Measure of goodness (optimization objective)
Bayes' rule
pattern recognizer
Fusion training
calibration transform
prior
training algorithm (optimizer)
proper scoring rule
calibrated likelihoods
user
fusion weights
calibration parameters
measure of goodness (optimization objective)
true class labels
posterior
Bayes' rule
prior
More general calibration-sensitive discriminative training

Bayes' rule

pattern recognizer

raw likelihoods (scores)
calibration transform
calibrated likelihoods
user

recognizer parameters

calibration parameters
Bayes' rule

training algorithm (optimizer)

measure of goodness (optimization objective)
proper scoring rule
true class labels

recognizer parameters

prior
posterior
In summary

- Calibration transformation is easiest to apply to class likelihoods—because simple, affine transforms work well in log-likelihood domain.

- Calibration-sensitive measurement is done on class posteriors—because proper scoring rules work with probability distributions (not likelihoods).

So let’s introduce proper scoring rules.
Proper scoring rules

Proper scoring rules can be approached via

1. The classical definition
2. Decision theory
3. Information theory:

\[
\text{expectation}\{\text{scoring rule}\} = \text{cross-entropy}
\]

We’ll discuss (1) and (2). See (Brümmer, 2010) for (3).
We explain the classical definition via a weather forecasting example.

\[ P(\text{rain}) = ? \]
Proper scoring rules

Classical definition

Weather forecasting example:

- A weather forecaster predicts rain for tomorrow with probability $p$.
- The next day, once we have observed whether it rains or not, how do we judge the goodness of forecast $p$?

We can choose some cost function:

$$C(p, h), \quad h \in \{\text{rain, no rain}\}$$

What properties should this cost function have to encourage accurate, well-calibrated predictions $p$?
It is far from obvious how to design the cost function $C(p, h)$.

- If temperature had been predicted, that would be easier, e.g.
  \[
  \text{cost} = (t_{\text{predicted}} - t_{\text{observed}})^2
  \]

- But we can’t observe a true probability for rain—we can just observe whether it rains or not.

\[
\text{cost} = (p_{\text{predicted}} - p_{\text{observed}})^2
\]
There is a family of cost functions (proper scoring rules), of the form $C(p, h)$, which encourages the weather predictor to:

- make prediction $p$ as accurate as possible, and then to
- honestly report $p$ and not some other probability $q$.

Below, we show how to design the cost function for honesty and then show that the accuracy follows automatically.
Forecaster trusts model and observation: \( p \) is the best prediction he can make.

Does he report \( p \) as is?

Or could circumstances motivate him to instead report some other probability \( q \)?
We want our cost function to motivate the forecaster to make $p$ as accurate as possible and then report $p$, not something else.
The weatherman’s probability for rain tomorrow is $p$, but he issues a possibly different $q$ as his forecast. Tomorrow he will be evaluated with cost function $C(q, \text{observed weather})$. His expected cost is:

$$\langle C(q, h) \rangle_p = pC(q, \text{rain}) + (1 - p)C(q, \text{no rain})$$

The cost function $C$ is a proper scoring rule if it satisfies:

$$\langle C(p, h) \rangle_p \leq \langle C(q, h) \rangle_p$$

which should motivate him to issue: $q = p$. 
The forecaster’s current data and model give prediction $p$. But if the data and model could be improved he would be able to compute a more accurate prediction $p'$. Even if the forecaster does not know the value of $p'$, he realizes that if he did have a more accurate $p'$, he would calculate his expected cost w.r.t. $p'$, in which case:

$$\langle C(p', h) \rangle_{p'} \leq \langle C(p, h) \rangle_{p'}$$

which should motivate him to improve his resources so that he can issue a more accurate $p'$ instead of $p$. 
Expected cost vs $p$, for two different proper scoring rules.

Minimizing expected cost encourages `sharpness' (or precision), so that $p$ is close to 0 or 1.
What has this got to do with us?

Why am I going on about motivating humans to make better weather predictions? What has this got to do with automatic speaker and language recognition?

‘Motivating’ machines via a cost function is just discriminative training. When proper scoring rules are used as discriminative training objective, we can expect the same benefits as explained above.
Examples of proper scoring rules

There are many different proper scoring rules. The original one is the Brier score:

\[ C(p, \text{rain}) = (1 - p)^2, \quad C(p, \text{no rain}) = p^2 \]

Arguably, the most useful one is the logarithmic score:

\[ C(p, \text{rain}) = -\log(p), \quad C(p, \text{no rain}) = -\log(1 - p) \]

In both cases, elementary calculus (or Jensen’s inequality) show that they obey the expectation requirement for proper scoring rules.
Outline

1. Introduction
2. The problem
3. Proper scoring rules
   - Overview
   - Classical definition
   - Bayes decisions
   - Discrimination/calibration decomposition
4. Proper scoring rule design
Let us now turn from statistics to engineering.

- We are building probabilistic recognizers to serve some useful purpose.
- **Bayes decision theory** provides the mathematical model for how to use probabilistic information.
- If you ask: ‘What are the consequences of the Bayes decision that I can make with probability $q$?, you are constructing a proper scoring rule for evaluating $q$.
- All proper scoring rules can be interpreted in this way.

This construction is shown next:
true hypothesis: $h$

Detection Cost Function: $C(\text{decision}, h)$

<table>
<thead>
<tr>
<th>Decision</th>
<th>Target</th>
<th>Non-target</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>0</td>
<td>$C_{fa}$</td>
</tr>
<tr>
<td>reject</td>
<td>$C_{miss}$</td>
<td>0</td>
</tr>
</tbody>
</table>
probability distribution: 
$q, 1-q$

true hypothesis: 
$h$

Proper scoring rule

Bayes decision:
min expected cost

probability distribution:
$q, 1-q$

true hypothesis:
$h$

scoring rule induced by DCF: $C(q, h)$
This is exactly how NIST’s new 2012 Speaker Recognition Evaluation criterion is defined. This year SRE’12 will use a proper scoring rule!

- Until SRE’10, NIST required hard decisions, which were evaluated by DCF.
- For SRE’12, NIST requires soft, probabilistic decisions (likelihood-ratios), which will be evaluated by the above DCF-induced proper scoring rule.
Bayes decisions form proper scoring rules

This recipe can be generalized to more than two classes and to other cost functions:

- Choose a cost function which reflects your application.
- Given a probability distribution for the unknown class, make a minimum-expected-cost Bayes decision.
- The value of the proper scoring rule is the cost of the Bayes decision, given the true class.

(A trivial inequality shows that this recipe always satisfies the expectation requirement for proper scoring rules.)
The Bayes decision interpretation shows:

- If you need a proper scoring rule, you can derive it from your favourite cost function via a Bayes decision.
- This proper scoring rule will measure (and optimize) the cost-effectiveness of your probabilistic recognizer.
The Bayes decision interpretation makes it clear that proper scoring rules evaluate the full cost of using the probabilistic recognizer to make decisions.

Often it is useful to decompose this cost into two components:

1. The cost attributed to the underlying inability of the recognizer to perfectly discriminate between classes (even if calibration is perfect).

2. The extra cost attributed to poor calibration (poor probabilistic representation of the discriminating information extracted by the recognizer).
NIST’s actual DCF vs min DCF is a good example of this decomposition.

- In this DCF case, calibration refers only to decision threshold setting.

Can we do the same with proper scoring rules, to address more general probabilistic calibration?

- Yes indeed, this is a useful exercise to do also with proper scoring rules. The next slide gives a recipe.
Decomposition recipe

1. Isolate (or add) a few calibration parameters at the output end of your recognizer. (This choice can be the subject of much debate, especially for multiclass!!)

2. On your supervised evaluation database, evaluate the actual cost of your system as is, using your proper scoring rule.

3. Using the true class labels, the evaluator re-trains only the calibration parameters, by minimizing the same scoring rule as in 2.

Compare the minimum and actual costs. If they are close, calibration was good.
Outline

1. Introduction
2. The problem
3. Proper scoring rules
4. Proper scoring rule design
   - Combination
   - Low FA region
   - Multiclass scoring rules
   - Conclusion
New proper scoring rules can be constructed by combining existing ones (corresponding to mixtures of applications).

- In SRE’12, NIST will use a combination of two proper scoring rules (induced by two different DCF operating points).

Indeed all binary (two-class) proper scoring rules can be constructed by weighted DCF combinations.

- DCF is a fundamental building block for binary proper scoring rules.
probability distribution: $q, 1-q$

true hypothesis: $h$

Bayes decision

$q > t$

Accept

Reject

Proper scoring rule

DCF

<table>
<thead>
<tr>
<th></th>
<th>target</th>
<th>non-target</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>0</td>
<td>$Cfa = 1 / (1 - t)$</td>
</tr>
<tr>
<td>reject</td>
<td>$Cmiss = 1 / t$</td>
<td>0</td>
</tr>
</tbody>
</table>

normalized DCF scoring rule: $C^*_t (q, h)$

- threshold at $t$
Normalized DCF scoring rule: $t = 0.3$

- $C_{\text{miss}}$
- $C_{\text{fa}}$

$C^*_t(p, \text{target})$
$C^*_t(p, \text{nontar})$
\[ C_{\log}(p, \text{target}) \quad C_{\log}(p, \text{nontar}) \quad C^*(p, \text{target}) \quad C^*(p, \text{nontar}) \]
All binary proper scoring rules can be expressed as an expectation over operating points:

$$C_w^*(q, h) = \int_0^1 C_t^*(q, h)w(t) \, dt$$

$C_t^*(q, h)$ is the normalized DCF scoring rule
- operating point (threshold) $t$.

$w(t)$ is a weighting distribution:
- $w(t) \geq 0$ and $\int_0^1 w(t) \, dt = 1$.
- determines relative importance of operating points
- controls important properties of the rule.
There is a rich variety of binary scoring rules that can be obtained by various choices of the weighting distribution.

- If $w(t)$ is an impulse, we get a proper scoring rule that represents a single operating point.
- If $w(t)$ is a sum of impulses, we can cover multiple discrete operating points.
- If $w(t)$ is a smooth function, the rule averages over a continuous range of operating points.
Examples of discrete (impulse) weighting:

- SRE 2010 operating point:
  \[ w(t) = \delta(t - 0.999) \]

- SRE 2012 evaluation criterion:
  \[ w(t) = \frac{1}{2} \delta(t - 0.999) + \frac{1}{2} \delta(t - 0.99) \]
Examples of smooth weighting:

- Brier score:
  \[ w(t) = 6t(1 - t) \]

- logarithmic score (essentially $C_{llr}$):
  \[ w(t) = 1 \]

The weighting matters a lot. The Brier score forms a poor (non-convex) discriminative optimization objective. The logarithmic score forms a good (convex) objective.
This is how the weighted integral forms the logarithmic scoring rule, with \( w(t) = 1 \), \( C_{\text{miss}} = \frac{1}{t} \) and \( C_{\text{fa}} = \frac{1}{1-t} \):

\[
\begin{align*}
C_{\text{w}}^*(q, \text{target}) &= \int_q^1 C_{\text{miss}}(t) w(t) \, dt = \int_q^1 \frac{1}{t} \, dt = -\log(q) \\
C_{\text{w}}^*(q, \text{nontar}) &= \int_0^q C_{\text{fa}}(t) w(t) \, dt = \int_0^q \frac{1}{1-t} \, dt = -\log(1 - q)
\end{align*}
\]

where \( q \) is the posterior probability for the target, accepted when \( q > t \).
The range of thresholds, $0 < t < 1$, correspond to operating points on the DET-curve. The choice of $w(t)$ allows tailoring of proper scoring rules to represent e.g. the low false alarm region.

George Doddington recently proposed another way to achieve the same effect, called $C_{llr-M10}$:

- Use standard logarithmic scoring rule ($C_{llr}$), and just
- omit scores outside of the low false alarm region.

(See SRE’12 Evaluation Plan, and Interspeech 2012)
Doddington’s scoring rule

truncated $C_{llr}$

$C_{llr-M10}$ does not quite adhere to the proper scoring rule form, $C(p, h)$, because it involves a system-and-answer-key dependent threshold, at $P_{\text{miss}} = 10\%$.

Let’s define instead: truncated $C_{llr}$, with a fixed threshold, so that it does fit the definition. It is formed via choice of $w(t)$:

- original $C_{llr}$: $w(t) = 1$
- truncated $C_{llr}$: $w(t) = 10 \times \text{unit-step}(t - 0.9)$
The low FA region

$C_{llr}$ variations

$C_{llr}$ is the logarithmic rule, applied to log-likelihood-ratio’s via prior=0.5. In the next slide, we compare the original to 2 variations:

<table>
<thead>
<tr>
<th></th>
<th>$w(t)$</th>
<th>prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>original $C_{llr}$</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>shifted $C_{llr}$</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>truncated $C_{llr}$</td>
<td>$10 \text{ustep}(t - 0.9)$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

We plot weighting against LLR threshold, $x$:

- $x = \log \frac{t}{1-t} - \log \frac{\text{prior}}{1-\text{prior}}$ (inverse Bayes’ rule)

- weighting: $w(t) \, dt = w(t)t(1 - t) \, dx$
Weighting distributions for $C_{llr}$ variations

- **original $C_{llr}$ (prior=0.5)**
- **shifted $C_{llr}$ (prior=0.1)**
- **truncated $C_{llr}$ (prior=0.5)**
- **DCF 2008**
- **DCF 2010**

**LLR threshold (minimum-expected-cost)**

- **low miss region**
- **EER region**
- **low FA region**

**Weight**
Buja’s scoring rules
The beta family

The beta family (Buja, 2005) is a flexible family of proper scoring rules, weighted with the beta distribution:

\[ w(t) = \text{Beta}(t|a, b), \quad a, b > 0 \]

This family is convenient, because it is
- general: includes Brier, logarithmic and \( C_t^* \)
- flexible: 2 adjustable parameters
- integral can be solved in closed form (integer \( a, b \)).

Here we propose: \( w(t) = \text{Beta}(t|10, 1) \) for a smoother alternative to truncated \( C_{llr} \).
In case anyone wants to try it, the proposed Beta(10,1) scoring rule has the formula:

\[ C_{10,1}(p, \text{target}) = \frac{1 - p^9}{9} \]

\[ C_{10,1}(p, \text{nontar}) = - \log(1 - p) - \sum_{k=1}^{9} \frac{p^k}{k} \]

Let’s compare it graphically against the three \( C_{llr} \) variations:
Weighting distributions: $C_{llr}$ variations vs Beta

prior = 0.5 (except for shifted $C_{llr}$)

low miss region
EER region
low FA region

original $C_{llr}$
shifted $C_{llr}$ (prior=0.1)
truncated $C_{llr}$
Beta(10,1)
DCF 2008
DCF 2010
But what about multiclass?
Multiclass scoring rules have received less attention in the literature (they are much more difficult to analyze). But, it is useful to know:

- construction via Bayes decisions and combination.
- logarithmic scoring rule = expectation over weighted misclassification errors.
- logarithmic scoring rule forms a good evaluation criterion and discriminative training objective—will be exercised as such in Albayzín 2012 Language Recognition Evaluation.
Proper scoring rules

- are essential for recognizers with probabilistic output.
- work well for discriminative training.
- have a rich structure—can be tailored.
- may become useful also for generative training of energy-based models.
Selected References

see Brümmer, 2010 for more references


Any questions?