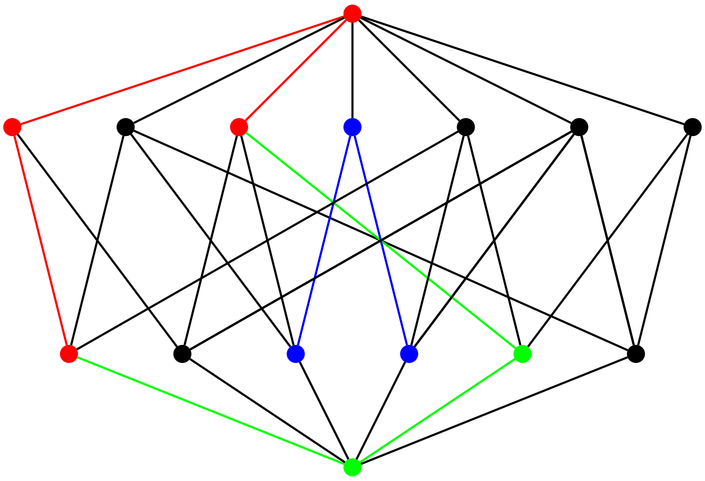
Meta-embeddings: a probabilistic generalization of embeddings in machine learning

Niko Brümmer,[1] Lukáš Burget, Paola Garcia, Oldřich Plchot,
Johan Rohdin, Daniel Romero, David Snyder,
Themos Stafylakis, Albert Swart, Jesús Villalba

JHU HLTCOE 2017 SCALE Workshop



[1] Corresponding author: niko.brummer@gmail.com. The other authors are in alphabetical order.

Contents

List of Figures

Chapter 1

Introduction

Embeddings are familiar in modern machine learning. Neural nets to extract word embeddings[1] were already proposed in 2000 by Bengio [1]. Now embeddings are used more generally, for example in state-of-the-art face recognition, e.g. Facenet [2].

Embeddings are becoming popular also in speech and speaker recognition. At Interspeech 2017, eighteen papers had the word 'embedding' in the title.[2] In speaker recognition and spoken language recognition, we have been using i-vectors, a precursor to embeddings, for almost a decade [3, 4, 5]. More general embeddings are now appearing in speaker recognition, see for example the Voxceleb paper [6] and David Snyder's work [7, 8], while a new embedding method for language recognition is proposed in [9].

Input patterns (sequences of acoustic feature vectors, images, text, ...) live in large, complex spaces, where probability distributions and geometrical concepts such as distance are difficult to formulate. The idea with embeddings is that they are representations of complex input patterns that live in simpler spaces, e.g. $\mathbb{R}^d$ (multidimensional Euclidean space), where distance is naturally defined and can be put to work to compare patterns.

At the Johns Hopkins HLTCOE SCALE 2017 Workshop[3] one of the topics of interest was embeddings for speaker recognition. During the workshop, the idea to generalize to *meta-embeddings* was conceived. At the time of the writing of this document, we are still developing the theory and very little coding and experiments have been done.[4] This document describes the theory.

[1] en.wikipedia.org/wiki/Word_embedding

[2] www.interspeech2017.org/program/technical-program/.

[3] http://hltcoe.jhu.edu/research/scale/scale-2017

[4] This document and some code to produce some of the examples in it can be found at https://github.com/bsxfan/meta-embeddings.

We envisage that meta-embeddings could be widely applicable to a variety of machine learning or pattern recognition problems, but we shall make our discussion concrete by using the language of automatic speaker verification/recognition. To interpret everything (say) as face recognition, just do *recording* $\mapsto$ *image* and *speaker* $\mapsto$ *face*, and so on ....

1.1 Meta-embeddings

An important concept used throughout this document is that of the *hidden identity variable*—a hypothetical, multidimensional variable that ideally contains all of the information that distinguishes one speaker/face/individual from others. In speaker recognition, the hidden *speaker identity variable* is well known from the work of Patrick Kenny in JFA [10] and PLDA [11]. Similar identity variables appeared in earlier applications of PLDA to face recognition [12, 13, 14].

We argue that the way embeddings are currently treated, essentially makes them *point estimates of hidden variables.* We propose instead to let the identity variables—and therefore the embeddings—remain hidden and to instead extract *meta-embeddings*, which are probabilistic representations of what the values of the hidden variables might be. For example, if the embedding lives in $\mathbb{R}^d$, then the meta-embedding could be in the form of a multivariate normal distribution, where the mean could act as point estimate, but where there is also a covariance matrix that quantifies the uncertainty around the mean.

Quantifying the uncertainty is very important if the recognizer is to be applicable to variable and sometimes challenging conditions. In speaker recognition, a short, noisy, narrow-band recording should leave much more uncertainty about the speaker than a long, clean, wideband recording. In face recognition, compare a well-lit, high resolution, full-frontal face image to a grainy, low resolution, partly occluded face. In fingerprint recognition, compare a clean, high-resolution ten-print, to a single, distorted, smudged fingermark retrieved from a crime scene.

The idea to represent the uncertainty is not novel. Indeed, it is *obvious* that to do things properly, we must take the uncertainty into account. The problem is that it turns out to be difficult to do in practice.

- Part of the problem is computational. For example, a point estimate for an identity variable in $\mathbb{R}^d$ can be represented as a d-dimensional vector. But for a multivariate normal meta-embedding, the covariance is a d-by-d matrix, which is more expensive to store and manipulate.

- Then there is the added complication that covariance matrices have to be positive definite. If we use a neural net to compute our uncertainty for us, we have to specially structure the neural net to respect this requirement.

- But the problem has another facet—if a neural net manufactures meta-embeddings for us, how do we know that the uncertainty thus represented is reasonable? Given meta-embeddings extracted from some supervised database, what criteria can we apply to measure the goodness of the information and the uncertainty they contain? Such criteria are important—we need them to measure performance, but they are also essential for any discriminative training of the neural nets that extract our meta-embeddings.

This document will provide some suggestions for tackling the above problems.

The main purpose of the document is however to show that the meta-embeddings are themselves embeddings in the sense that they live in a vector space, equipped with geometric and algebraic structure that can be employed to compute probabilistic recognition scores (likelihood ratios). The hope is that the theoretical part of this document will help to provide the structure to guide further research into finding better solutions for the practical problems of working with this kind of uncertainty.

1.2 Application mechanisms

In this section, we briefly sketch how meta-embeddings might be applied. Obvious applications are speaker recognition, face recognition, fingerprint comparison and so on—we do not attempt to list all possible applications here. Rather, we are interested in the mechanisms whereby meta-embeddings are extracted and then used.

The first part of the document, chapters 2 to 6, describes the nature of meta-embeddings and how to use them to recognize speakers, faces and so on.

But before we can use them, they have to be extracted from the input data (voice recordings, images, etc.). Since meta-embeddings are probabilistic, they will be extracted using probabilistic models, which can be either generative or discriminative.

PLDA is a good example of a generative model that naturally extracts meta-embeddings. The PLDA model can be trained either with maximum likelihood (the classical generative training criterion) [15, 16], or with discriminative methods [17, 18]. Once the PLDA model has been trained, extraction of meta-embeddings can be done in closed form, as will be explained in chapter 7. More sophisticated deep generative models could also be used to extract meta-embeddings, but in those cases, we envisage the extraction would be done with a trainable neural network component, similar to that in variational autoencoders [19].

Meta-embeddings can also be extracted purely discriminatively, without also training a generative model. For classical embeddings, this is the usual strategy, see for example [2, 7, 8, 6, 9]. In chapter 8 we generalize existing discriminative training strategies to be applicable to meta-embeddings.

1.3 Prior work

In [20], Gaussian embeddings are proposed to represent uncertainty. To be expanded, . . .

Mention uncertainty propagation in i-vectors[21, 22, 23, 24], . . .

Chapter 2

Distillation of meta-embeddings from independence assumptions

In this chapter we use probability theory and some basic independence assumptions to motivate and derive the concept of meta-embeddings.

We use speaker recognition terminology, but everything is applicable more widely (images, etc. ...). Our inputs are *voice recordings*, each assumed to contain a single speaker. Recordings can have various representations, i-vectors, sequences of feature vectors, the raw speech samples, etc. .... At the writing of this document, recordings represented as feature vector sequences are of primary interest.

In general, we will be interested in *sets* of recordings, ranging from the set of all the recordings in a training database, down to just a pair of recordings in a verification trial. We represent a set of n recordings as:

$$\mathcal{R} = \{r_1, r_2, \dots, r_n\}$$

We shall work with hypotheses about how such sets of recordings are partitioned w.r.t. speaker and we shall make use of independence assumptions conditioned on these hypotheses.

2.1 Independence assumptions

We base everything that follows on a pair of simple independence assumptions:

- Multiple recordings of the same speaker are *exchangeable.*
- Recordings of different speakers are *independent.*

Almost all current probabilistic solutions in speaker recognition—also PLDA—make use of these assumptions. While these assumptions are mathematically convenient, in the real world they are not exactly true:

- The speech of any speaker will change over a time span of years [25] and recordings made over such a period will not be exchangeable.

- The independence assumption between different speakers can only apply if our recognizer is sufficiently well trained: The recognizer has to have been exposed to so much speech, that only the speech of a new speaker itself can provide information about that speaker [26].

As long as we understand the limitations imposed by our assumptions, we can proceed.

In what follows, we use the notation, H_1, for the hypothesis that the set of recordings of interest, $\mathcal{R} = \{r_1, \dots, r_n\}$, were all spoken by the same (one) speaker. More generally, for the hypothesis that they were spoken by m different speakers, we use H_m. Exchangeability means:

$$P(\mathcal{R} \mid H_1) = P(r_1, r_2 \dots, r_n \mid H_1) = P(r_2, r_1, \dots, r_n \mid H_1) = \cdots$$

where the joint probability does not depend on the order of the recordings. According to De Finetti's exchangeability theorem,[1] the only way to obtain exchangeability is to introduce some hidden variable, say $\mathbf{z} \in \mathcal{Z}$, such that the recordings are *conditionally independent* given $\mathbf{z}$. Marginalizing over $\mathbf{z}$ w.r.t. some prior distribution $\pi(\mathbf{z})$, gives the exchangeable joint distribution [27]:

$$P(\mathcal{R} \mid H_1) = \bigl\langle \prod_{i=1}^{n} P(r_i \mid \mathbf{z}) \bigr\rangle_{\mathbf{z}\sim\pi} \tag{2.1}$$

where the angle brackets denote expectation—for continuous $\mathbf{z}$, this could be written as an integral and for discrete $\mathbf{z}$ as a summation. This marginalization is the model's mechanism for inducing dependence between recordings of the same speaker, while retaining exchangeability. *We do need this dependence, otherwise speaker recognition would be impossible!*

Let $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_m$ be a number of non-overlapping sets of recordings, spoken respectively by m different speakers. The between-speaker independence

[1] en.wikipedia.org/wiki/Exchangeable_random_variables

assumption gives:

$$P(\mathcal{R}_1,\dots,\mathcal{R}_m \mid m \text{ speakers}) = \prod_{i=1}^m P(\mathcal{R}_i \mid H_1) = \prod_{i=1}^m \bigl\langle \prod_{r\in\mathcal{R}_i} P(r \mid \mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi} \tag{2.2}$$

where we have expanded $P(\mathcal{R}_i \mid H_1)$ using (2.1).

According to this model, the hidden variable, $\mathbf{z}\in\mathcal{Z}$, contains everything that is to be known about recordings of a speaker. Patrick Kenny calls $\mathbf{z}$ the *speaker identity variable*. At this stage, we are not committing ourselves to the nature of the hidden variables and therefore we leave $\mathcal{Z}$ undefined—after all, $\mathbf{z}$ is hidden so there is much freedom in choosing its nature. (If you need something concrete to shape your thoughts, $\mathcal{Z}=\mathbb{R}^d$ is a good choice.)

Think of the identity variable, $\mathbf{z}$, as the ideal embedding that an oracle could extract. If we were given $\mathbf{z}$, the problem would be instantly solved. Unfortunately, there is noise, distortion, occlusion and all kinds of other mechanisms that induce uncertainty, so we can never exactly extract $\mathbf{z}$. Let's not pretend that we can do this. Let us not extract some point-estimate for $\mathbf{z}$ and call that our embedding. Let's instead extract a *meta-embedding*, which is *information about* the ideal embedding, $\mathbf{z}$, where the information has a probabilistic form. We derive this form below.

2.2 Speaker identity likelihood functions

Given a pair of recordings, $\mathcal{R}=\{r,r'\}$, we can answer the question of whether they belong to the same speaker or not in terms of the *likelihood-ratio (LR)* [28]:

$$\begin{aligned}\frac{P(r,r' \mid H_1)}{P(r,r' \mid H_2)} &= \frac{\bigl\langle P(r\mid\mathbf{z})\,P(r'\mid\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}}{\bigl\langle P(r\mid\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}\,\bigl\langle P(r'\mid\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}}\\ &= \frac{\bigl\langle kP(r\mid\mathbf{z})\,k'P(r'\mid\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}}{\bigl\langle kP(r\mid\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}\,\bigl\langle k'P(r'\mid\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}}\\ &= \frac{\bigl\langle f_r(\mathbf{z})\,f_{r'}(\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}}{\bigl\langle f_r(\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}\,\bigl\langle f_{r'}(\mathbf{z})\bigr\rangle_{\mathbf{z}\sim\pi}}\end{aligned} \tag{2.3}$$

where we have defined the *speaker identity likelihood functions*:

$$f_r(\mathbf{z}) = kP(r\mid\mathbf{z}) \qquad \text{and} \qquad f_{r'}(\mathbf{z}) = k'P(r'\mid\mathbf{z}), \tag{2.4}$$

where $k, k' > 0$ are arbitrary constants that may depend on r, r', but not on $\mathbf{z}$. Notice that the LR depends on the data (r, r') only via f_r and $f_{r'}$. The speaker information in r is represented by the *whole function*

$$f_r : \mathcal{Z} \to \mathbb{R}$$

rather than by the function value $f_r(\mathbf{z})$ at some particular value of $\mathbf{z}$ (remember $\mathbf{z}$ is hidden and we are never given a fixed value for it).

The full speaker information cannot be represented by some estimate of $\mathbf{z}$. We could for example use $\hat{\mathbf{z}} = \operatorname{argmax} f_r(\mathbf{z})$ as a maximum-likelihood point-estimate for $\mathbf{z}$, but that would be throwing information away. The full information about the speaker is contained in the whole function, f_r.

The representation of the speaker information can be further understood by noticing that f_r and $f_{r'}$ represent r, r' in any posteriors for the speaker identity variable, for example:

$$P(\mathbf{z} \mid r, \pi) = \frac{\pi(\mathbf{z}) f_r(\mathbf{z})}{\left\langle f_r(\mathbf{z}') \right\rangle_{\mathbf{z}' \sim \pi}} \tag{2.5}$$

and if r and r' are known to be of the same speaker, then:

$$P(\mathbf{z} \mid r, r', H_1, \pi) = \frac{\pi(\mathbf{z}) f_r(\mathbf{z}) f_{r'}(\mathbf{z})}{\left\langle f_r(\mathbf{z}') f_{r'}(\mathbf{z}') \right\rangle_{\mathbf{z}' \sim \pi}} \tag{2.6}$$

2.3 The general case

Our speaker identity likelihood functions are applicable more generally than just to pairs of recordings. Under the independence assumptions of section 2.1, speaker identity likelihood functions can be used to answer *any* question that can be formulated in terms of partitioning a set of recordings according to speaker [16].

Let $\mathcal{R} = \{r_1, \dots, r_n\}$ be a set of recordings and let $A : \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m \subseteq \{1, 2, \dots, n\}$ be a hypothesized partition of $\mathcal{R}$ into subsets belonging to $m \geq 1$ different hypothesized speakers.[2] Similarly, let $B : \mathcal{S}'_1, \dots, \mathcal{S}'_{m'}$ be a different hypothesized partition, having m' hypothesized speakers. A and B are labels that refer to the two alternate partitioning hypotheses. We shall use the notation $A, B \in \mathcal{P}_n$, where $\mathcal{P}_n$ refers to the set of all partitions of $\{1, \dots, n\}$. The cardinality of $\mathcal{P}_n$ is the n-th *Bell number*. Do not confuse partition with subset: a partition is an element of $\mathcal{P}_n$, while each partition is composed of one or more index subsets (aka *blocks*).

[2] The subsets are non-empty, non-overlapping and their union equals $\{1, \dots, n\}$. We allow $m = 1$, in which case $\mathcal{S}_1 = \{1, \dots, n\}$.

Using within-speaker exchangeability and between-speaker independence, the LR comparing A to B is:

$$\begin{aligned}\frac{P(\mathcal{R} \mid A)}{P(\mathcal{R} \mid B)} &= \frac{\prod_{i=1}^{m} \left\langle \prod_{j\in\mathcal{S}_i} P(r_j \mid \mathbf{z}) \right\rangle_{\mathbf{z}\sim\pi}}{\prod_{i=1}^{m'} \left\langle \prod_{j\in\mathcal{S}'_i} P(r_j \mid \mathbf{z}) \right\rangle_{\mathbf{z}\sim\pi}} \\ &= \frac{\prod_{i=1}^{m} \left\langle \prod_{j\in\mathcal{S}_i} f_j(\mathbf{z}) \right\rangle_{\mathbf{z}\sim\pi}}{\prod_{i=1}^{m'} \left\langle \prod_{j\in\mathcal{S}'_i} f_j(\mathbf{z}) \right\rangle_{\mathbf{z}\sim\pi}}\end{aligned} \tag{2.7}$$

where, as above, the *speaker identity likelihood function* extracted from r_j is:

$$f_j(\mathbf{z}) = k_j P(r_j \mid \mathbf{z}) \tag{2.8}$$

The last equality in (2.7) follows because the arbitrary scaling constants, $\{k_j\}_{j=1}^n$, are the same in the numerator and denominator and cancel. This LR form is convenient because of this cancellation.

Any kind of speaker recognition problem that can be expressed in terms of partitioning recordings according to speaker, can be scored at runtime via likelihood-ratios of the form (2.7). We shall show some examples below. Moreover, any discriminative training criteria that are expressed as functions of the LR scores can then also be expressed in terms of (2.7). In summary:

> *For discriminative speaker recognition, all scoring and training calculations can be expressed in terms of the speaker identity likelihood functions.*

To better appreciate this generality, we shall present some examples in the next section.

Again, for readers that are comfortable with the idea that probability distributions convey information [29], the speaker identity likelihood functions are closely associated with posteriors for $\mathbf{z}$. Any such posterior, conditioned on a number of recordings of a speaker, where these recordings indexed by $\mathcal{S}$, can be computed as:

$$P(\mathbf{z} \mid \mathcal{R}, \mathcal{S}, H_1, \pi) = \frac{\pi(\mathbf{z}) \prod_{j\in\mathcal{S}} f_j(\mathbf{z})}{\left\langle \prod_{j\in\mathcal{S}} f_j(\mathbf{z}') \right\rangle_{\mathbf{z}'\sim\pi}} \tag{2.9}$$

The likelihood function, f_j, or indeed a product of any number of them, can be regarded as a *pre-posterior* for $\mathbf{z}$. We use the term pre-posterior, because the likelihood function is independent of the choice of prior, π, and is also un-normalized. Nevertheless, f_j contains all the speaker information that a given probabilistic model could extract from r_j.

2.4 Examples

The examples below should aid understanding of the meaning and applications of (2.7). We will frequently refer back to these examples in the rest of this document.

Binary verification. Let $\mathcal{R} = \{r_1, r_2\}$, $A : \mathcal{S}_1 = \{1, 2\}$ and $B : \mathcal{S}'_1 = \{1\}, \mathcal{S}'_2 = \{2\}$. Then $\frac{P(\mathcal{R}|A)}{P(\mathcal{R}|B)}$ is the canonical (single-enroll) speaker verification LR.

Multi-enroll verification. Let $\mathcal{R} = \{r_1, r_2, r_3\}$, $A : \mathcal{S}_1 = \{1, 2, 3\}$ and $B : \mathcal{S}'_1 = \{1, 2\}, \mathcal{S}'_2 = \{3\}$. Then $\frac{P(\mathcal{R}|A)}{P(\mathcal{R}|B)}$ is the speaker verification LR, with enrollment recordings r_1, r_2 and test recording r_3. This generalizes in the obvious way to more than two enrollments.

Open-set classification. Let a set of enrollment recordings, $\mathcal{R} = \{r_1, \dots, r_n\}$, be partitioned into m speakers, indexed by $\mathcal{S}_1, \dots, \mathcal{S}_m \subseteq \{1, \dots, n\}$. Let $r' \notin \mathcal{R}$ be an additional (test) recording. For $1 \leq i \leq m$, let A_i be the hypothesis that r' is spoken by speaker i, as indexed by $\mathcal{S}_i$. Let A_{m+1} be the hypothesis that there are $m+1$ speakers: m of them indexed by $\mathcal{S}_1, \dots, \mathcal{S}_m$ and r' spoken by a new speaker. Then we can score the open-set speaker classification problem as follows: For $1 \leq i \leq m+1$, the likelihood[3] for A_i is

$$L_i = \frac{P(\mathcal{R}, r' \mid A_i)}{P(\mathcal{R}, r' \mid A_{m+1})} \tag{2.10}$$

Notice that likelihood that r' was spoken by a new speaker is just $L_{m+1} = 1$. Assuming some hypothesis prior, $P(A_i)$, the hypothesis posterior is:

$$P(A_i \mid \mathcal{R}, r') = \frac{P(A_i) L_i}{\sum_{j=1}^{m+1} P(A_j) L_j} \tag{2.11}$$

Agglomerative hierarchical clustering. Let $\mathcal{R} = \{r_1, \dots, r_n\}$ be partitioned into m hypothesized speakers, indexed by $A : \mathcal{S}_1, \dots, \mathcal{S}_m \subseteq \{1, \dots, n\}$. For $1 \leq i, j \leq m$ and $i \neq j$, let B_{ij} be the hypothesis that the speakers are correctly indexed as given by A, except that the recordings indexed by $\mathcal{S}_i$ and $\mathcal{S}_j$ are from the same speaker. That is, if we accept hypothesis B_{ij}, then

[3] The numerator of L_i by itself, $P(\mathcal{R}, r' \mid A_i)$, could also serve as likelihood for A_i. Since the denominator is independent of i, the normalized ratio, L_i, is still a valid likelihood for A_i, with the advantage that the troublesome data-dependent constants cancel.

we should reduce the number of speakers to $m-1$ and merge $\mathcal{S}_i$ and $\mathcal{S}_j$. We let

$$L_{ij} = \frac{P(\mathcal{R} \mid B_{ij})}{P(\mathcal{R} \mid A)} \tag{2.12}$$

be the (conveniently normalized) likelihood for hypothesis B_{ij}, while the likelihood for hypothesis A is just 1. These likelihood scores can be used as part of an iterative, greedy optimization[4] procedure (agglomerative hierarchical clustering) to find a partition of $\mathcal{R}$ according to speaker, with high posterior probability.

Summary. All of these examples are scored using likelihood ratios of the form (2.7) and are therefore dependent on the data only via the speaker identity likelihood functions.

We have so far demonstrated that f_j qualifies as an embedding in the sense that it contains all of the relevant information in r_j. In what follows we shall refer to $\mathbf{z}$ as the (hidden, ideal) *embedding*, while the f_j are our *meta-embeddings*.

It remains to be shown in the next chapter that these meta-embeddings live in a space with useful algebraic and geometric structure.

[4] Finding the global optimum is apparently hopelessly intractable. For a set of $n = 75$ recordings, we already have that the number of possible ways to partition this set exceeds the number of atoms (about 10^{80}) in the observable universe.

Chapter 3

The structure of meta-embedding space

Classical embeddings in machine learning live in finite-dimensional, Euclidean space, $\mathbb{R}^d$. When equipped with the standard scalar product, vector addition, dot-product and Euclidean distance, $\mathbb{R}^d$ is a vector space, inner-product space, metric space, Hilbert space, etc. Our meta-embeddings, the functions f_j, live in a slightly generalized space, which is still a vector space, inner product space, metric space and (subject to regularity conditions) a Hilbert space.

Depending on the nature of the hidden variable, $\mathbf{z} \in \mathcal{Z}$, our meta-embeddings might live in a space with very high, or even infinite dimensionality. In the theory that follows, everything can be more elegantly described in this high/infinite-dimensional space, but in practice—to be discussed in the next chapter—we will consider various finite, tractable representations of our meta-embeddings.

In this chapter we explore first the geometric (vector space) properties of meta embeddings that establish concepts such as length, distance and angle. The vector space operations include addition, scalar multiplication and inner product.

Second, we enrich the vector space by introducing another (elementwise) multiplicative operator, which then forms an algebra, the properties of which we explore. This paves the way for the next chapter where we assemble these algebraic manipulations to compute likelihood-ratios of the form (2.7).

3.1 Geometric properties

Let $\mathbb{R}^{\mathcal{Z}}$ denote the space of all possible functions from $\mathcal{Z}$ to $\mathbb{R}$ and let our meta-embeddings live in $\mathcal{F}$, where $\mathcal{F} \subset \mathbb{R}^{\mathcal{Z}}$. We need to confine meta-embeddings to some subset of $\mathbb{R}^{\mathcal{Z}}$ in order to exclude all kinds of pathological functions and to ensure that the operations defined on meta-embeddings are well behaved. Since our meta-embeddings are likelihoods, the function values are always non-negative, so we might consider restricting ourselves to $\mathbb{R}_+^{\mathcal{Z}}$, where $\mathbb{R}_+ = [0, \infty)$. But this does not give a vector space, because it will not be closed w.r.t. scalar multiplication. It will therefore be more convenient to work in a more general space, $\mathcal{F}$, that includes functions with negative values.

3.1.1 Functions as vectors

Consider the Euclidean vector, $\mathbf{x} \in \mathbb{R}^d$ and notice that $\mathbf{x}$ can be viewed as a *function* that maps the component index to the real line: $\{1, \dots, d\} \to \mathbb{R}$.

Now let the hidden speaker identity variable be a d-dimensional Bernoulli vector, $\mathbf{z} \in \mathcal{Z} = \{0, 1\}^d$. That is, $\mathbf{z}$ is a vector with d components, each of which can be either 0 or 1. A function, $f : \{0, 1\}^d \to \mathbb{R}$, can be represented as a vector having 2^d real components. More generally, whenever $\mathcal{Z}$ has finite cardinality, $f : \mathcal{Z} \to \mathbb{R}$ can be represented as a vector in Euclidean space.

However, for the continuous case, when $\mathbf{z} \in \mathbb{R}^d$, the function $f(\mathbf{z})$ is like a vector with an infinite number of components. For the vector-space operations of *scalar multiplication* and *addition*, the infinite dimensionality poses no difficulty. These operations are defined as follows. Let $f, g, h \in \mathcal{F}$ and $\alpha, \beta \in \mathbb{R}$, then:

$$h = \alpha f + \beta g \qquad \Leftrightarrow \qquad h(\mathbf{z}) = \alpha f(\mathbf{z}) + \beta g(\mathbf{z}) \tag{3.1}$$

As long as we choose $\mathcal{F}$ such that it is closed under these operations and to include the constant function with value 0, then $\mathcal{F}$ is a vector space, even though the vectors are infinite-dimensional. To quality as a vector space, the properties required of the addition and scalar multiplication operators — commutativity, associativity and distributivity—follow directly from (3.1).

3.1.2 Inner product

We already have that $\mathcal{F}$ is a vector space. To further make it an *inner product space*, we need to define the inner product, which is a function, $\mathcal{F} \times \mathcal{F} \to \mathbb{R}$. For $f, g, h \in \mathcal{F}$, and $\alpha, \beta \in \mathbb{R}$, we define our inner product as:

$$\langle f, g \rangle = \langle f(\mathbf{z}) g(\mathbf{z}) \rangle_{\mathbf{z} \sim \pi} \tag{3.2}$$

which is the expectation of the product of f and g, w.r.t π, a probability distribution over $\mathcal{Z}$. If we choose π to be our hidden variable prior, then the inner product coincides with the numerator in the likelihood-ratio of (2.3). Notice that we use the same triangular bracket notation for both inner product and expectation—this ambiguous notation will be convenient later.

For (3.2) to be a valid inner product, it needs to satisfy symmetry, linearity, and positive definiteness. The symmetry, $\langle f, g \rangle = \langle g, f \rangle$ and linearity, $\langle \alpha f + \beta g, h \rangle = \alpha \langle f, h \rangle + \beta \langle g, h \rangle$, follow directly from the definition. It also follows that $\langle f, f \rangle \geq 0$, which is necessary for positive definiteness. However, we also need that $\langle f, f \rangle = 0$ if and only if $f(\mathbf{z}) = 0$ everywhere. For this we need $\pi(\mathbf{z})$ to be non-zero everywhere on $\mathcal{Z}$ and we need $\mathcal{F}$ to impose certain smoothness constraints—e.g. we need to exclude from $\mathcal{F}$ functions that differ from each other only on subsets of $\mathcal{Z}$ of measure zero. In what follows, we shall assume that $\mathcal{F}$ is chosen such that it forms a valid inner-product space together with the operations defined here.[1]

Notice that (3.2) is a generalization of the usual dot product in Euclidean space. It generalizes the simple summation in the dot product by introducing the weighting by $\pi(\mathbf{z})$. For continuous $\mathcal{Z}$, summation is further generalized to integration.

Is $\mathcal{F}$ a Hilbert space? A Hilbert space is an inner product space with the additional requirement of *completeness*. If $\mathcal{F}$ is chosen to have this property, then yes, we have a Hilbert space. In what follows, we will not make use of completeness and we will therefore refer to our space with (slightly) greater generality as an inner product space.

3.1.3 Norm, distance, angle

The inner product naturally supplies the notions of length, distance and angle, making our space a metric space and a normed space. The *norm, or length* is defined as:

$$\|f\| = \sqrt{\langle f, f \rangle} \tag{3.3}$$

Once we have length, *distance* is defined as the length of the difference between two vectors, so that squared distance is given by:

$$\|f - g\|^2 = \langle f - g, f - g \rangle = \|f\|^2 + \|g\|^2 - 2\langle f, g \rangle \tag{3.4}$$

[1] An alternative construction of the inner product space could be to use the quotient space, populated by equivalence classes of functions, where two functions, $f(\mathbf{z})$ and $g(\mathbf{z})$ are equivalent if $P(f(\mathbf{z}) = g(\mathbf{z}) \mid \mathbf{z} \sim \pi) = 1$. See [30].

Conversely, it may be useful in some situations to express inner product in terms of norms. We can solve for $\langle f, g\rangle$ in (3.4), but notice that we can alternatively use:

$$\|f+g\|^2 = \langle f+g, f+g\rangle = \|f\|^2 + \|g\|^2 + 2\langle f, g\rangle \tag{3.5}$$

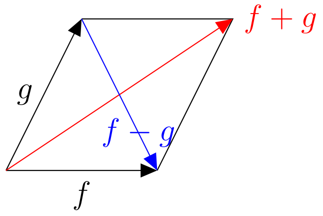
Addition of (3.4) and (3.5) gives the *parallelogram law*:

$$\|f+g\|^2 + \|f-g\|^2 = 2\,\|f\|^2 + 2\,\|g\|^2 \tag{3.6}$$

Combining everything, we can express the *inner product in terms of norms* in at least three different ways:

$$\begin{aligned}\langle f, g\rangle &= \frac{1}{2}\big(\|f\|^2 + \|g\|^2 - \|f-g\|^2\big)\\ &= \frac{1}{2}\big(\|f+g\|^2 - \|f\|^2 - \|g\|^2\big)\\ &= \frac{1}{4}\big(\|f+g\|^2 - \|f-g\|^2\big)\end{aligned} \tag{3.7}$$

To better understand this geometry, view the parallelogram, with sides f, g, and diagonals $f+g$ and $f-g$:



Finally, we can also define angle. For any inner product space we have the *Cauchy-Schwartz inequality*:

$$\big|\langle f, g\rangle\big| \le \|f\|\,\|g\| \tag{3.8}$$

so that the *angle* between f and g, say θ, can be defined as:

$$\cos\theta = \frac{\langle f, g\rangle}{\|f\|\,\|g\|} \tag{3.9}$$

because then $|\cos\theta| \le 1$, as it should.

Having defined angle, we can also talk about pairs of orthogonal vectors, for which the inner product is zero and which have an angle of $\pi/2$ between them. For orthogonal vectors, with $\langle f, g\rangle = 0$, we can use (3.4) to derive the *theorem of Pythagoras*:

$$\|f-g\|^2 = \|f\|^2 + \|g\|^2 \tag{3.10}$$

3.1.4 Note on meta-embeddings as random variables

This note is of theoretical interest and is not essential to understanding the rest of the document.

Let $(\mathcal{Z}, \mathcal{A}, \pi)$ be a probability space, where $z \in \mathcal{Z}$ is the identity variable as defined above. $\mathcal{A}$ is a suitable sigma-algebra, consisting of subsets of $\mathcal{Z}$. The probability measure of this space, $\pi : \mathcal{A} \to [0, 1]$ is the prior on z. Under the regularity condition that our meta-embeddings are measurable functions from $\mathcal{Z}$ to $\mathbb{R}$, the meta-embeddings can be viewed as *random variables* [31]. Informally, when f_j is a meta-embedding and if we sample $z \sim \pi$, then the likelihood $f_j(z)$ is a sample from the random variable f_j.

The interested reader is encouraged to read both [30] and [32] where it is explained how the expectation of the product of two random variables, i.e. our (3.2), forms a well-defined inner product. Further reading may include appendix B of [33] and references therein.

3.2 Algebraic properties

Having described the inner-product space, we now enrich our meta-embedding space with the elementwise multiplicative operation. This operation combines in useful ways with the other operations to enable us to do the calculations we need for our ultimate goal of computing the likelihood ratios of the form (2.7).

Alert readers may notice that our addition operator does not seem to play any useful, practical role in all of this. First, the fact that addition between meta-embeddings is possible helps to establish that we have a vector space. Once this is established, all of the well-known properties of vector spaces become available to us. (For example, to derive the basic property of distance, we need subtraction, which in turn is defined by scalar multiplication and addition.) Moreover, in later chapters, we do find a practical application for addition, in the form of mixture distributions.

3.2.1 Elementwise product

We now equip our space with the *elementwise product* (Hadamard product). Since this product is associative, we can write it using juxtaposition, defining it as:

$$h = fg \qquad \Leftrightarrow \qquad h(\mathbf{z}) = f(\mathbf{z})g(\mathbf{z}) \tag{3.11}$$

The vector space, $\mathcal{F}$, equipped additionally with the elementwise product forms an *algebra*, because the elementwise product is bilinear w.r.t. addition and scalar multiplication.[2]

We introduce the elementwise product to represent *pooling* of the relevant information extracted from multiple recordings. Let $\mathcal{R} = \{r_1, \ldots, r_n\}$ be a set of recordings of the same speaker. If the corresponding meta-embeddings are $f_j(\mathbf{z}) = k_j P(r_j \mid \mathbf{z})$, then

$$f_{\mathcal{R}}(\mathbf{z}) = \prod_{j=1}^{n} f_j(\mathbf{z}) = \bigl(\prod_j k_j\bigr) P(\mathcal{R} \mid \mathbf{z}) \tag{3.12}$$

gives the *pooled meta-embedding*, $f_{\mathcal{R}} = \prod_j f_j$, which represents all of the speaker information in $\mathcal{R}$, in the sense that:

$$P(\mathbf{z} \mid \mathcal{R}, H_1, \pi) = \frac{\pi(\mathbf{z}) f_{\mathcal{R}}(\mathbf{z})}{\bigl\langle f_{\mathcal{R}}(\mathbf{z}') \bigr\rangle_{\mathbf{z}' \sim \pi}} \tag{3.13}$$

3.2.2 Multiplicative identity

We define the *multiplicative identity* as the constant function with value 1 everywhere: $\mathbf{1}(\mathbf{z}) = 1$. This turns out to be a (conveniently normalized) meta-embedding extracted from any *empty recording*. An empty recording is a recording that contains no speech—it could have zero duration, or otherwise contain silence, noise, or any other non-speech sounds. Since empty recordings have no speaker information, speaker identity variable posteriors conditioned on them are just equal to the prior. Letting r_ϕ be an empty recording, we have:

$$P(\mathbf{z} \mid r_\phi, \pi) = \frac{\mathbf{1}(\mathbf{z}) \pi(\mathbf{z})}{\bigl\langle \mathbf{1}(\mathbf{z}) \bigr\rangle_{\mathbf{z} \sim \pi}} = \pi(\mathbf{z}) \tag{3.14}$$

Notice that:

$$\|\mathbf{1}\| = \sqrt{\bigl\langle 1^2 \bigr\rangle_{\mathbf{z} \sim \pi}} = 1 \tag{3.15}$$

so that $\mathbf{1}$ represents the *diagonal axis* of length 1, with all 'components' equal to 1. The inner product, $\bigl\langle f, \mathbf{1} \bigr\rangle$, will be of special interest below. It is the length of the orthogonal projection of f onto this diagonal axis—see section 4.4 for a graphical example.

[2] Compare this to the complex numbers, which under addition and (real) scalar multiplication, form a 2-dimensional vector space. If complex multiplication is added, it is also an algebra. See: https://en.wikipedia.org/wiki/Algebra_over_a_field.

3.2.3 L1 and L2 norms

We have already introduced the norm, $\|f\| = \sqrt{\langle f, f\rangle}$. Observe that this is an L2 norm:

$$\|f\| = \sqrt{\langle f, f\rangle} = \sqrt{\langle f^2(\mathbf{z})\rangle_{\mathbf{z}\sim\pi}} = \|f\|_2 \tag{3.16}$$

We shall also find the L1 norm useful. Notice that when f is non-negative, we can interpret $\langle f, \mathbf{1}\rangle$ as an L1 norm:

$$\|f\|_1 = \langle |f(\mathbf{z})|\rangle_{\mathbf{z}\sim\pi} \tag{3.17}$$

It is easy to show using (3.15) and Cauchy-Schwartz (3.8) that:

$$\|f\|_1 \leq \|f\|_2 \tag{3.18}$$

If f is non-negative, then $\langle f, 1\rangle = \|f\|_1 \leq \|f\|_2$. More generally:

$$\big|\langle f, 1\rangle\big| \leq \|f\|_2 \tag{3.19}$$

Below we restrict attention to meta-embeddings for which $\big|\langle f, \mathbf{1}\rangle\big| < \infty$ and it is useful to know that this condition is automatically satisfied if $\|f\| = \|f\|_2 < \infty$.

3.2.4 Normalized meta-embedding

Let $f \in \mathcal{F}$ be a meta-embedding. We define the corresponding *normalized meta-embedding* as:

$$\overline{f} = \frac{1}{\langle f, \mathbf{1}\rangle} f \qquad\qquad \Leftrightarrow \qquad\qquad \overline{f}(\mathbf{z}) = \frac{f(\mathbf{z})}{\langle f(\mathbf{z}')\rangle_{\mathbf{z}'\sim\pi}} \tag{3.20}$$

If f_r is the meta-embedding for recording r and $f_r(\mathbf{z}) = kP(r \mid \mathbf{z})$, then:

$$P(\mathbf{z} \mid r, \pi) = \pi(\mathbf{z})\overline{f_r}(\mathbf{z}) \tag{3.21}$$

Similarly, for a set of recordings, $\mathcal{R}$, all of the same speaker, let $f_\mathcal{R}$ be the pooled meta-embedding, such that $f_\mathcal{R}(\mathbf{z}) = kP(\mathcal{R} \mid \mathbf{z}, H_1)$, then we can rewrite (3.13) more compactly as:

$$P(\mathbf{z} \mid \mathcal{R}, H_1, \pi) = \pi(\mathbf{z})\overline{f_\mathcal{R}}(\mathbf{z}) \tag{3.22}$$

We shall require that all meta-embeddings are *normalizable* in the sense that $\big|\langle f, 1\rangle\big| < \infty$. By (3.19), $\|f\| < \infty$ is a sufficient condition. Alternatively, any bounded function, $f(\mathbf{z})$ also satisfies this requirement.

Properties

The following properties of normalized meta-embeddings are easily shown. By definition, we have that the length of the projection of any normalized meta-embedding onto $\mathbf{1}$ is unity:

$$\langle \overline{f}, \mathbf{1} \rangle = 1 \tag{3.23}$$

This gives that *differences between normalized meta-embeddings* live in a hyperplane perpendicular to $\mathbf{1}$:

$$\langle \overline{f} - \overline{g}, \mathbf{1} \rangle = 0 \tag{3.24}$$

By (3.18) we have:

$$\|\overline{f}\| \geq 1 \tag{3.25}$$

with equality at $\overline{f} = \mathbf{1}$. Using Pythagoras (3.10), we find the interesting formula:

$$\|\overline{f}\|^2 = \|\mathbf{1}\|^2 + \|\overline{f} - \mathbf{1}\|^2 = 1 + \|\overline{f} - \mathbf{1}\|^2 \tag{3.26}$$

The sum of two normalized meta-embeddings is easy to renormalize:

$$\overline{\overline{f} + \overline{g}} = \frac{1}{2}(\overline{f} + \overline{g}) \tag{3.27}$$

which, when used with (3.26), gives:

$$\left\|\frac{1}{2}(\overline{f} + \overline{g})\right\|^2 = \left\|\overline{\overline{f} + \overline{g}}\right\|^2 = 1 + \left\|\frac{1}{2}\overline{f} + \frac{1}{2}\overline{g} - \mathbf{1}\right\|^2 \tag{3.28}$$

or

$$\|\overline{f} + \overline{g}\|^2 = 4 + \|(\overline{f} - \mathbf{1}) + (\overline{g} - \mathbf{1})\|^2 \tag{3.29}$$

This compares to:

$$\|\overline{f} - \overline{g}\|^2 = \|(\overline{f} - \mathbf{1}) - (\overline{g} - \mathbf{1})\|^2 \tag{3.30}$$

which inspires the definition of *centered* meta-embeddings:

3.2.5 Centered meta-embeddings

For a meta-embedding f, the normalized meta-embedding is $\overline{f}$ and the *centered* meta embedding is:

$$\tilde{f} = \overline{f} - \mathbf{1} \tag{3.31}$$

The inner product is:

$$\begin{aligned} \langle \tilde{f}, \tilde{g} \rangle &= \langle \overline{f} - \mathbf{1}, \overline{g} - \mathbf{1} \rangle \\ &= \langle \overline{f}, \overline{g} \rangle - \langle \overline{f}, \mathbf{1} \rangle - \langle \mathbf{1}, \overline{g} \rangle + \langle \mathbf{1}, \mathbf{1} \rangle \\ &= \langle \overline{f}, \overline{g} \rangle - 1 \end{aligned} \tag{3.32}$$

By (3.7), we therefore have:

$$\begin{aligned} \langle \overline{f}, \overline{g} \rangle &= 1 + \langle \tilde{f}, \tilde{g} \rangle \\ &= 1 + \frac{1}{2}(\left\|\tilde{f}\right\|^2 + \|\tilde{g}\|^2 - \left\|\tilde{f} - \tilde{g}\right\|^2) \\ &= 1 + \frac{1}{2}(\left\|\tilde{f} + \tilde{g}\right\|^2 - \left\|\tilde{f}\right\|^2 - \|\tilde{g}\|^2) \\ &= 1 + \frac{1}{4}(\left\|\tilde{f} + \tilde{g}\right\|^2 - \left\|\tilde{f} - \tilde{g}\right\|^2) \end{aligned} \tag{3.33}$$

3.2.6 Notations for inner product and expectation

Using the definitions of inner product and elementwise multiplication, notice that $\langle f, g \rangle = \langle fg, \mathbf{1} \rangle$ and indeed that $\langle fgh, \mathbf{1} \rangle = \langle fg, h \rangle = \langle f, gh \rangle = \langle \mathbf{1}, fgh \rangle$, and so on. The comma in the inner product notation has no real function and it can be omitted. We shall write:

$$\langle f, g \rangle = \langle fg \rangle = \langle f(\mathbf{z})g(\mathbf{z}) \rangle_{\mathbf{z}\sim\pi} \tag{3.34}$$

and more generally,

$$\langle \prod_j f_j \rangle = \langle \prod_j f_j(\mathbf{z}) \rangle_{\mathbf{z}\sim\pi} \tag{3.35}$$

In what follows, we shall interchangeably use the notations with or without comma, to respectively emphasize the dot-product or expectation interpretations.

Chapter 4

Likelihood-ratios

We are now ready to demonstrate the utility of our meta-embedding operators. We show how they can be used to express the LR formulas of chapter 2.

We start with *simple LRs*, which can be used to compare any two hypotheses for the partitioning of a set of recordings according to speaker, where the partitions differ by splitting or combining a single subset. Then we treat the general case, where we compute LRs that compare arbitrary hypotheses. The chapter concludes with a graphical toy example to aid understanding.

4.1 Simple LRs

Here we show how to write the LR of each of the examples of section 2.4 as an inner product between normalized meta-embeddings.

The *binary verification* likelihood ratio (2.3), rewritten in a variety of notations, is:

$$\frac{P(r, r' \mid H_1)}{P(r, r' \mid H_2)} = \frac{\langle f, f' \rangle}{\langle f, \mathbf{1} \rangle \langle f', \mathbf{1} \rangle} = \frac{\langle f f' \rangle}{\langle f \rangle \langle f' \rangle} = \langle \overline{f}, \overline{f'} \rangle \tag{4.1}$$

where f and f' are the meta-embeddings extracted from r and r'.

The *multi-enroll verification* LR, with enrollments: $r_1 \mapsto f_1$ and $r_2 \mapsto f_2$ and test: $r_3 \mapsto f_3$, is:

$$\frac{P(r_1, r_2, r_3 \mid H_1)}{P(r_1, r_2, r_3 \mid H_2)} = \frac{\langle f_1 f_2, f_3 \rangle}{\langle f_1 f_2, \mathbf{1} \rangle \langle f_3, \mathbf{1} \rangle} = \frac{\langle f_1 f_2 f_3 \rangle}{\langle f_1 f_2 \rangle \langle f_3 \rangle} = \langle \overline{f_1 f_2}, \overline{f_3} \rangle \tag{4.2}$$

For *open-set classification*, with enrollment recordings, $\mathcal{R} = \{r_1, \dots, r_n\}$ for m speakers, indexed by $\mathcal{S}_1, \dots, \mathcal{S}_m \subseteq \{1, \dots, n\}$ and a new test recording, r', we use the meta-embeddings: $\{r_\ell \mapsto f_\ell\}_{\ell=1}^n$ and $r' \mapsto f'$ and we calculate the LR that r' is of speaker i, vs that r' is a new speaker, as:

$$\begin{aligned} L_i &= \frac{\langle f' \prod_{\ell\in\mathcal{S}_i} f_\ell\rangle \prod_{j\neq i}\langle \prod_{\ell\in\mathcal{S}_j} f_\ell\rangle}{\langle f'\rangle \prod_{j=1}^m \langle \prod_{\ell\in\mathcal{S}_j} f_\ell\rangle} \\ &= \frac{\langle f' \prod_{\ell\in\mathcal{S}_i} f_\ell\rangle}{\langle f'\rangle\langle \prod_{\ell\in\mathcal{S}_i} f_\ell\rangle} \\ &= \langle \overline{f'}, \overline{\prod_{\ell\in\mathcal{S}_i} f_\ell}\rangle \end{aligned} \tag{4.3}$$

For *agglomerative clustering* we can similarly write the LR (2.12) in terms of normalized meta-embeddings as:

$$L_{ij} = \frac{\langle \prod_{\ell\in\mathcal{S}_i\cup\mathcal{S}_j} f_\ell\rangle}{\langle \prod_{\ell\in\mathcal{S}_i} f_\ell\rangle\langle \prod_{\ell\in\mathcal{S}_i} f_\ell\rangle} = \langle \overline{\prod_{\ell\in\mathcal{S}_i} f_\ell}, \overline{\prod_{\ell\in\mathcal{S}_j} f_\ell}\rangle \tag{4.4}$$

In summary: In each of these simple cases, the likelihood-ratio is most compactly represented as an *inner product between normalized meta-embeddings.* In what follows, we refer to these inner products more concisely as *normalized inner products.*

4.2 The general case

If we consider the most general case of LRs of the form (2.7), we find that there are cases that cannot be written as a single normalized inner product. But we show that they *can* however be written as products and ratios of such inner products. We start with an example.

Example 1. Consider a set of four recordings, $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ with associated meta-embeddings, $\{f_1, f_2, f_3, f_4\}$ and consider the two hypotheses:

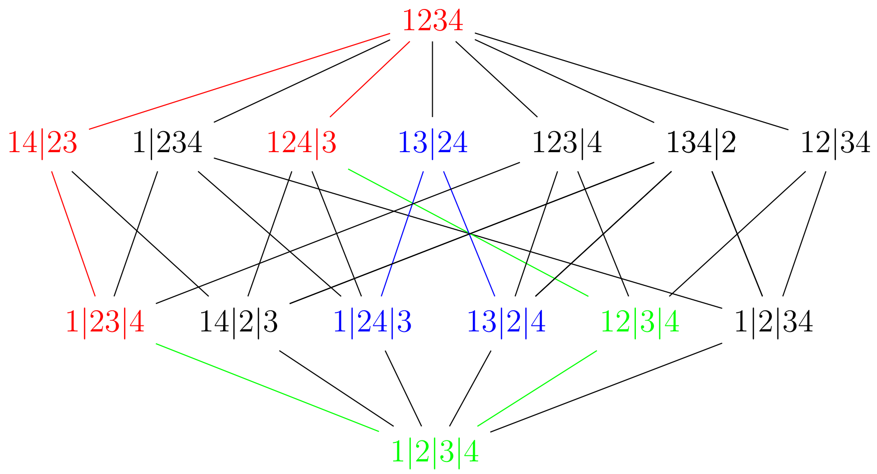


Figure 4.1: Hasse diagram of $\mathcal{P}_4$, the lattice of partitions of a set of 4 recordings, ordered by 'refines'. Every downward arc is an an atomic refinement, which splits a single subset. Example 1 is highlighted in blue, example 2a in red and 2b in green.

$A : 1|24|3$ and $B : 13|2|4$, written in a convenient short-hand. The LR is:

$$\begin{aligned}
\frac{P(\mathcal{R} \mid A)}{P(\mathcal{R} \mid B)} &= \frac{\langle f_1\rangle\langle f_2 f_4\rangle\langle f_3\rangle}{\langle f_1 f_3\rangle\langle f_2\rangle\langle f_4\rangle} \\
&= \frac{\langle f_1\rangle\langle f_2 f_4\rangle\langle f_3\rangle}{\langle f_1 f_3\rangle\langle f_2 f_4\rangle} \times \frac{\langle f_1 f_3\rangle\langle f_2 f_4\rangle}{\langle f_1 f_3\rangle\langle f_2\rangle\langle f_4\rangle} \\
&= \frac{\langle f_1\rangle\langle f_3\rangle}{\langle f_1 f_3\rangle} \times \frac{\langle f_2 f_4\rangle}{\langle f_2\rangle\langle f_4\rangle} \\
&= \frac{\langle \overline{f_2}, \overline{f_4}\rangle}{\langle \overline{f_1}, \overline{f_3}\rangle}
\end{aligned} \tag{4.5}$$

We have achieved this re-arrangement by observing that we can get from A to B via the auxiliary hypothesis $C : 13|24$, where C is reached from A by joining $\{1\}\{3\}$ and B is reached from C by splitting $\{2, 4\}$. The join contributes a normalized inner product factor below the line and the split contributes another above the line.

In general, the partitions of a set form a *partial order*, where $A < C$ is defined as: A *is finer than* C. In our example, we have both $A < C$ and $B < C$, but A and B are *not directly comparable*. If two hypotheses *are* directly comparable (if they differ by joining or splitting a single subset), then their LR can be expressed as a single normalized inner product, as we have seen in section 4.1.

More can be said about this partial order. It is in fact a *lattice*, where every pair of partitions has a unique supremum (least upper bound) and a unique infimum (greatest lower bound). Figure 4.1 shows the Hasse diagram representing the lattice of four recordings. Every arc in the Hasse diagram corresponds to splitting or joining a single subset. As we shall see, there is a simple LR associated with every arc.

To form the LR between any pair of hypotheses, we can traverse the lattice from one hypothesis to the other via any path connected by arcs, but the shortest paths will go via the supremum or the infimum. Since the infimum and supremum always exist, there will always be a path. Every step (split or join)[1] contributes a normalized inner product factor to the final LR.

In example 1 we traversed the blue path via $13|24 = \sup(A, B)$. If we instead traverse via $1|2|3|4 = \inf(A, B)$, we get the *same* decomposition. This is however not true in general—different paths can give different decompositions of the LR. This is shown in the next example.

Example 2a. Let's try a more complex path, highlighted in red in figure 4.1. Let $A : 1|23|4$ and $B : 124|3$. We have $\sup(A, B) = 1234$ and $\inf(A, B) = 1|2|3|4$. Whether we go up or down, we need at least three steps to traverse between A and B. There are three such paths via the supremum and three more via the infimum. Following the red highlighted path, the LR can be re-arranged as:

$$\begin{aligned}\frac{P(\mathcal{R} \mid A)}{P(\mathcal{R} \mid B)} &= \frac{\langle f_1\rangle\langle f_2 f_3\rangle\langle f_4\rangle}{\langle f_1 f_2 f_4\rangle\langle f_3\rangle} \\ &= \frac{\langle f_1\rangle\langle f_2 f_3\rangle\langle f_4\rangle}{\langle f_1 f_4\rangle\langle f_2 f_3\rangle} \times \frac{\langle f_1 f_4\rangle\langle f_2 f_3\rangle}{\langle f_1 f_2 f_3 f_4\rangle} \times \frac{\langle f_1 f_2 f_3 f_4\rangle}{\langle f_1 f_2 f_4\rangle\langle f_3\rangle} \\ &= \frac{1}{\langle \overline{f_1}, \overline{f_4}\rangle} \times \frac{1}{\langle \overline{f_1 f_4}, \overline{f_2 f_3}\rangle} \times \langle \overline{f_1 f_2 f_4}, \overline{f_3}\rangle\end{aligned} \tag{4.6}$$

Each upward step (join) contributes a normalized inner product below the line, while the downward step (split) contributes another above.

Example 2b. Let's also try a path via the infimum, highlighted in green, to traverse between the same two nodes as before. Now the LR can be

[1] In lattice theory, infimum and supremum are alternatively termed *meet* and *join*. In our usage here, we mean by *join* simply to form the union of two subsets of recordings.

re-arranged as:

$$\begin{aligned}\frac{P(\mathcal{R}\mid A)}{P(\mathcal{R}\mid B)} &= \frac{\langle f_1\rangle\langle f_2 f_3\rangle\langle f_4\rangle}{\langle f_1 f_2 f_4\rangle\langle f_3\rangle} \\ &= \frac{\langle f_1\rangle\langle f_2 f_3\rangle\langle f_4\rangle}{\langle f_1\rangle\langle f_2\rangle\langle f_3\rangle\langle f_4\rangle} \times \frac{\langle f_1\rangle\langle f_2\rangle\langle f_3\rangle\langle f_4\rangle}{\langle f_1 f_2\rangle\langle f_3\rangle\langle f_4\rangle} \times \frac{\langle f_1 f_2\rangle\langle f_3\rangle\langle f_4\rangle}{\langle f_1 f_2 f_4\rangle\langle f_3\rangle} \\ &= \langle \overline{f_2}, \overline{f_3}\rangle \times \frac{1}{\langle \overline{f_1}, \overline{f_2}\rangle} \times \frac{1}{\langle \overline{f_1 f_2}, \overline{f_4}\rangle}\end{aligned} \tag{4.7}$$

Again, each upward step (join) contributes a normalized inner product below the line, while the downward step (split) contributes another above. This rule applies in general. If we follow a path from the numerator (hypothesis A), to the denominator (hypothesis B), joins contribute below the line, while splits contribute above. (If we go from denominator to numerator, this rule is reversed.)

Equivalence of paths

As the two paths that we examined in example 2 demonstrate, there may be many different paths and therefore many different decompositions of a given LR in terms of normalized inner products. Nevertheless, as long as our calculations are exact, all such decompositions must give the same numerical result. This may serve as a useful regularization constraint if we are training neural nets to approximate inner products between meta-embeddings. (As we shall see in the next chapter, there may be various practical reasons to prefer approximate calculations.)

The equivalence of paths is not limited to shortest paths. The products and ratios of the normalized inner products of any path between two nodes must give the same numerical result, irrespective of the path. This is also true in particular of *cycles*. If we start and end at the same node, the LR we calculate thus is *unity*. The interesting thing about this is that if we are given a set of unlabelled recordings, we can use this fact to manufacture a (very) large set of constraints on the values of normalized inner products. (If we work with logarithms, this becomes a homogeneous system of linear equations.) These constraints may perhaps help to regularize neural nets in both labelled and unlabelled training regimes.

Primitive operations

Which primitive building blocks do we need to compute general LRs of the form (2.7)? Let's assume we can always extract raw (unnormalized) meta-embeddings from any recording. Then, as (2.7) shows, we can compute any LR if we can:

- arbitrarily pool meta-embeddings (do elementwise multiplication) and
- compute arbitrary expectations of single or pooled meta-embeddings.

Alternatively, as we have shown in this section, we can compute arbitrary LRs if we can

- pool and
- compute arbitrary normalized inner products.[2]

We can however somewhat restrict the requirement on normalized inner products. We can get away with inner products where *one of the arguments is always a single (unpooled) meta-embedding.* To do normalization, we need $\langle f, \mathbf{1} \rangle$, which respects this restriction. To further enforce this restriction, look again at the lattice in figure 4.1 and convince yourself that you can traverse between any two nodes by always adding or removing but a single element of some subset in the partition. For example, the red path does not respect this restriction, but the blue and green ones do. For such paths, all the inner product factors will have at least one unpooled argument.

When we discuss practical representations of meta-embeddings in the next chapter, we will see that for some of the representations, pooling may change the form of the representation. The form may have implications on how easy it is to do inner products. If one of the arguments is always unpooled, this may facilitate calculations.

Finally, we mention another interesting building block. It is not difficult to show that we can form arbitrary LRs from factors of the form:

$$\frac{\langle \prod_{i=1}^m f_i \rangle}{\prod_{i=1}^m \langle f_i \rangle} = \left\langle \prod_{i=1}^m \overline{f_i} \right\rangle \tag{4.8}$$

for $m = 1, 2, \ldots$. This factor is just the LR comparing the coarsest to the finest partition of a set of m recordings. It can be argued that this building block is not primitive, because it can be assembled via pooling and expectation (or pooling and inner products).

[2] The combination of pooling and inner products happens every day in our own vision. A light source has a spectrum, which is a function from $\mathbb{R}$ to $\mathbb{R}$. Every time the light reflects off a surface, the modified spectrum of the relected light is the product of the incoming spectrum and a similar function that represents the properties of the reflecting surface. Finally, each colour receptor in the eye has a similar function. Inner products between the light that reaches the retina and the receptors determine how much of each colour we see.

4.3 Note on cosine similarity

It is of interest to note that our normalized inner product (LR):

$$\langle \overline{f_1}, \overline{f_2} \rangle = \frac{\langle f_1, f_2 \rangle}{\|f_1\|_1 \, \|f_2\|_1}$$

is very similar to the popular cosine similarity between traditional embeddings, say $\mathbf{x}_1, \mathbf{x}_2$, given by:

$$\frac{\langle \mathbf{x}_1, \mathbf{x}_2 \rangle}{\|\mathbf{x}_1\|_2 \, \|\mathbf{x}_2\|_2}$$

The difference is the use of L1 vs L2 normalization.

In the literature, there are many examples where embeddings are L2-normalized before computing dot products or distances. In speaker recognition, the first i-vector scoring recipe used cosine-similarity [3], and indeed in current PLDA scoring recipes, it is still standard practice to use L2-normalized i-vectors [34]. In face recognition, Facenet uses cosine similarity to compare embeddings [2], although subsequently [35] advises against such normalization. In [9], the embeddings for language recognition are compared with cosine similarity.

Notice that by (3.4), if embeddings have unit L2 norm, then there is no essential difference (other than the sign) between cosine similarity (inner product) and squared distance. By the Cauchy-Schwartz inequality (3.8), inner products between L2-normalized embeddings are limited to the range $[-1, 1]$, where -1 corresponds to embeddings on opposite sides of the unit hypersphere (maximum distance) and where 1 corresponds to embeddings that coincide (zero distance).

In speaker recognition, cosine similarity between embeddings in Euclidean space plays the role of uncalibrated *log* likelihood-ratio. (The cosine similarity is signed, just like the log likelihood-ratio.) In contrast, our inner products between L1-normalized meta-embeddings give non-negative *likelihood-ratios*. Notice also that by (3.18), our likelihood-ratio magnitudes are *not* limited to be at most unity.

4.4 Toy example, with binary hidden variable

Let us now construct a graphical example to reinforce our geometric understanding of our meta-embeddings. In order to be able to plot the meta-embeddings, we choose perhaps the simplest possible meta-embeddings, which live in $\mathbb{R}^2$.

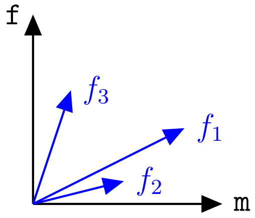
We do this by imagining a very weak speaker recognizer (meta-embedding extractor) that can only distinguish voices by whether they sound male or female. That is, we choose the simplest possible hidden speaker identity variable by making it discrete and binary: $\mathbf{z} \in \{\mathtt{m}, \mathtt{f}\}$. We choose the prior to have males and females equally likely:

$$\pi(\mathtt{m}) = \pi(\mathtt{f}) = \frac{1}{2}$$

Seen as a function, the meta-embedding for recording r_i is $f_i(\mathbf{z}) = k_i P(r_i \mid \mathbf{z})$. Since $\mathbf{z}$ has only two possible values, we can represent the meta-embedding as a *two-dimensional vector*: $f_i = \bigl[k_i P(r_i \mid \mathtt{m}), k_i P(r_i \mid \mathtt{f})\bigr]$. Because of the prior weighting, $\pi(\mathtt{m}) = \pi(\mathtt{f}) = \frac{1}{2}$, the inner product and dot product differ by a factor 2: if $f_i = [f_{i1}, f_{i2}]$, then:

$$\bigl\langle f_i, f_j \bigr\rangle = \frac{1}{2}(f_{i1}f_{j1} + f_{i2}f_{j2}) \tag{4.9}$$

For three hypothetical recordings, let the meta-embeddings be $f_1 = [2, 1]$, $f_2 = [1.2, 0.3]$, $f_3 = [0.5, 1.5]$ and we can plot these meta-embeddings as:



The first thing to notice is the difference between embedding and meta-embedding: The ideal embedding here is binary, $\mathbf{z} \in \{\mathtt{m}, \mathtt{f}\}$, while the meta-embedding is continuous: $f_i \in \mathbb{R}^2$. Here, and in general, the meta-embedding lives in a more complex space than $\mathbf{z}$.

Since likelihoods are positive, the meta-embeddings are confined to the positive quadrant. We should never be working outside of this quadrant, but to see the space as a vector space, we need to be aware of the existence of the other quadrants. Indeed, when we design our neural nets to extract meta-embeddings, we will have to make sure they end up being in the positive quadrant.

Because of the arbitrary scaling constants, k_i, the lengths of our meta-embedding vectors do not carry information—but from the directions we see that f_1 and f_2 are probably male, while f_3 is probably female. Our end-goal is not to infer the genders of the speakers, but to infer whether the speakers of different recordings are the same or not. A little thought should convince the reader that:

- The smallest possible likelihood-ratio, $\frac{P(r_i,r_j|H_1)}{P(r_i,r_j|H_2)} = 0$, would be obtained if we were certain that one speaker is male and the other female. This would happen when we have:

$$P(r_i \mid \mathtt{m}) \gg P(r_i \mid \mathtt{f}) \qquad \text{and} \qquad P(r_j \mid \mathtt{m}) \ll P(r_j \mid \mathtt{f})$$

  so that f_i is on the horizontal (male) axis, while f_j is on the vertical (female) axis. Then indeed, the inner product (and dot product) between these two orthogonal meta-embeddings would be zero and so would the LR.

- The largest possible LR with the weak, binary hidden variable would be just 2, which would be obtained when we are certain that both recordings are of the same gender. We need to consider details of the normalization to see how this works out.

Figure 4.2 shows the same three meta-embeddings, together with their normalized versions, with normalization as defined by (3.20). Remember the prior-weighting: when $f_i = [f_{i1}, f_{i2}]$, then the normalization constant is

$$\langle f_i, \mathbf{1} \rangle = \frac{1}{2}(f_{i1} + f_{i2})$$

Inner products between the normalized meta-embeddings give the likelihood-ratios,

$$\frac{P(r_i, r_j \mid H_1)}{P(r_i, r_j \mid H_2)} = \langle \overline{f_i}, \overline{f_j} \rangle$$

Since the discrimination given by the binary hidden variable is weak, the likelihood-ratios are not too far from the neutral value of 1. The LR, $\langle \overline{f_1}, \overline{f_2} \rangle = 1.2$ is greater than 1, slightly favouring the hypothesis that the speakers of r_1 and r_2 are the same. The other two LRs are less than one, favouring the H_2 hypothesis in each case. The LR, $\langle \overline{f_2}, \overline{f_3} \rangle = 0.64$ is *stronger* (further from 1) than $\langle \overline{f_1}, \overline{f_3} \rangle = 0.8$, because we are more certain of the maleness of f_2 than we are of f_1. The largest possible LR of 2 would be obtained when both normalized meta-embeddings coincide on the horizontal or vertical axis, at $[2,0]$, or $[0,2]$.[3]

Figure 4.3 shows what happens when we know that r_1 and r_2 are of the same speaker and we pool their meta-embeddings, giving us more certainty that this is a male speaker and more certainty that this speaker is different from the (probably female) speaker of r_3.

[3] Remember the prior weighting of $\frac{1}{2}$.

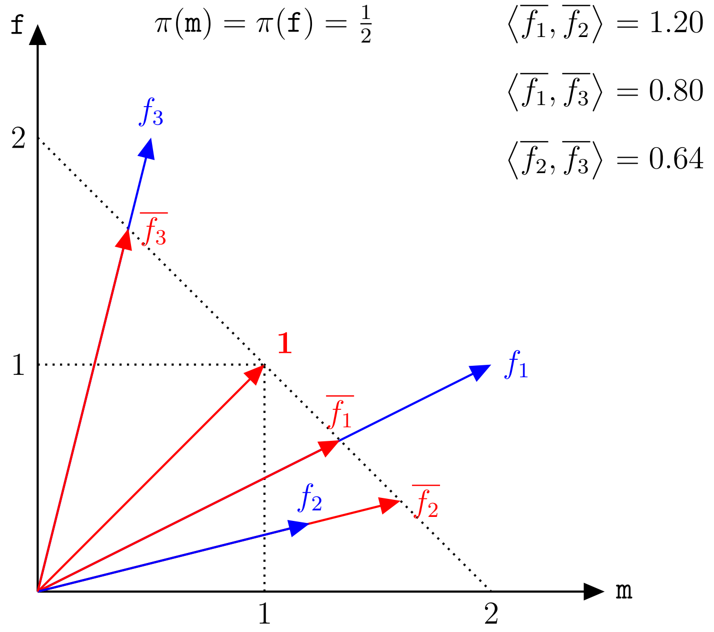

$\pi(\mathtt{m}) = \pi(\mathtt{f}) = \frac{1}{2}$
$\langle \overline{f_1}, \overline{f_2} \rangle = 1.20$
$\langle \overline{f_1}, \overline{f_3} \rangle = 0.80$
$\langle \overline{f_2}, \overline{f_3} \rangle = 0.64$
f
2
1
f_3
$\overline{f_3}$
1
f_1
$\overline{f_1}$
f_2
$\overline{f_2}$
1
2
m

Figure 4.2: Normalized meta-embeddings (red) and their inner products. Raw meta-embeddings (blue) are normalized by scaling them so that their projections on **1** are unity.

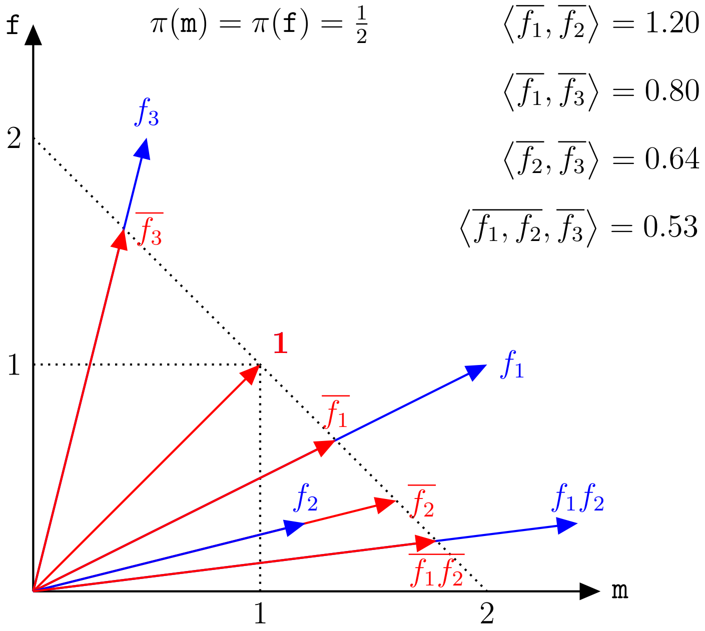

f
$\pi(\mathtt{m}) = \pi(\mathtt{f}) = \frac{1}{2}$
$\langle \overline{f_1}, \overline{f_2} \rangle = 1.20$
$\langle \overline{f_1}, \overline{f_3} \rangle = 0.80$
$\langle \overline{f_2}, \overline{f_3} \rangle = 0.64$
$\langle \overline{f_1, f_2}, \overline{f_3} \rangle = 0.53$
f_3
2
$\overline{f_3}$
1
1
f_1
$\overline{f_1}$
f_2
$\overline{f_2}$
$f_1 f_2$
$\overline{f_1 f_2}$
m
1
2

Figure 4.3: Pooling: If we know that f_1 and f_2 are from the same speaker, we can pool them, using the elementwise product $f_1 f_2$. This increases the certainty that this is a male speaker. Correspondingly, the strength (difference from 1) of $\langle \overline{f_1 f_2}, \overline{f_3} \rangle$ is more than either of $\langle \overline{f_1}, \overline{f_3} \rangle$ and $\langle \overline{f_2}, \overline{f_3} \rangle$. Pooling has increased the certainty that the speaker represented by $f_1 f_2$ is not the same speaker as the probably female f_3.

Chapter 5

Practical meta-embedding representations

We have thus far developed a theoretical idea of the nature of meta-embeddings. We are now ready to explore a few proposals of how to practically represent meta-embeddings.

As mentioned in the introduction, meta-embeddings can be naturally extracted from generative models, or can be extracted by purely discriminative means. In chapters 7 and 8 below we shall take a closer look at generative and discriminative meta-embedding extraction. Here we are interested in the form that practical meta-embedding representations might take and we shall explore several possibilities.

Before proceeding to these possibilities, let us consider in general, some desirable properties of our representations. Ideally, we need all of the following properties for our meta-embeddings:

Non-negativity: $f(\mathbf{z}) \geq 0$ everywhere.

Normalizability: $\langle f, \mathbf{1} \rangle < \infty$.

Pooling should be tractable, and the pooled result, $f_i f_j$, should have the same representation as f_i and f_j.

Expectation: $\langle f(\mathbf{z}) \rangle_{\mathbf{z}\sim\pi}$ should be tractable for any meta-embedding f, whether it is raw or pooled.

Backpropagation of derivatives through pooling and expectation is necessary for training.

Recall that if we can pool and do expectations, then we can also do any inner products, since $\langle f, g \rangle = \langle fg \rangle$.

The first two proposals below, based on exponential family distributions, meet all of these requirements, although the expectations are somewhat computationally expensive. In some of the other proposals, various approximations and compromises have to be made.

5.1 GME: Gaussian meta-embedding

An obvious candidate for a practical meta-embedding is the multivariate Gaussian. A big advantage of this representation is that everything can be computed in closed form, which makes it an important building block in everything that follows. We shall refer to *Gaussian meta-embeddings* with the acronym GME.

This representation supposes a continuous speaker identity variable, $\mathbf{z} \in \mathbb{R}^d$. For each recording, r_j, the corresponding meta-embedding, f_j, takes the form of a $(d+D)$-dimensional representation:

$$r_j \mapsto \mathbf{e}_j = (\mathbf{a}_j, \mathbf{b}_j)$$

where $\mathbf{a}_j \in \mathbb{R}^d$ and $\mathbf{b}_j \in \mathbb{R}_+^D$, where $D \geq 1$ and the elements of $\mathbf{b}_j$ are non-negative. While the representation, $\mathbf{e}_j$, is finite-dimensional, it parametrizes the *infinite-dimensional* meta-embedding, f_j, defined as:

$$f_j(\mathbf{z}) = \exp\bigl[\mathbf{a}_j'\mathbf{z} - \frac{1}{2}\mathbf{z}'\mathbf{B}_j\mathbf{z}\bigr] \tag{5.1}$$

where $\mathbf{B}_j$ is a d-by-d, positive semi-definite precision matrix, composed as a conical combination:

$$\mathbf{B}_j = \sum_{i=1}^{D} b_{ij}\mathbf{E}_i \tag{5.2}$$

where b_{ij} is a component of $\mathbf{b}_j$ and where the $\{\mathbf{E}_i\}_{i=1}^D$ are fixed, d-by-d, positive semi-definite matrices—they are fixed in the sense of being independent of the input data (recordings), but the elements of these matrices are still trainable, together with any parameters of the model that extracts the $\mathbf{e}_j$. These matrices can be full, low rank, diagonal, etc.

Since $\mathbf{z} \in \mathbb{R}^d$ is hidden, we are free to choose the dimensionality, d. Experience with PLDA suggests $100 \leq d \leq 200$ is a good choice. This can also be compared to the Facenet [2] embeddings, which are 120-dimensional.

The size, D, of the precision parametrization can be used to trade off computational complexity vs capacity. To keep the complexity significantly less than that which would be needed for fully specified precision matrices, we probably want to constrain $D \ll \frac{d(d+1)}{2}$.

Scalar multiplication could be easily done in this framework but in practice, we probably won't need it. Addition of the meta-embeddings would give mixtures of Gaussians, with more complex representations, but for this representation we don't need addition.

5.1.1 Elementwise product

The important elementwise product is done by simply adding the representation vectors. For $\mathcal{R} = \{r_1, \dots, r_n\}$, we can extract the individual representations, $\{\mathbf{e}_j\}_{j=1}^n$, which respectively represent the $\{f_j\}_{j=1}^n$. The representation for $f_\mathcal{R} = \prod_{j=1}^n f_j$ is then $\mathbf{e}_\mathcal{R} = \sum_{j=1}^n \mathbf{e}_j$. The representation $\mathbf{e}_j$ is essentially logarithmic, which gives us the benefit of automatic positivity, as well as easy elementwise multiplication. The disadvantage is somewhat complex expectation computation.

Note on group structure

For future reference, it might be useful to note that since Gaussians are closed under (associative) multiplication, our space of Gaussian meta-embeddings is a *monoid* (a semigroup, with identity). Since we exclude precision matrices with negative eigenvalues, our monoid lacks inverse elements, which would have made it a full group. The identity element is a Gaussian with zero mean and zero precision, which is just our previously defined identity, $\mathbf{1}(\mathbf{z}) = 1$.

5.1.2 Prior

To do expectations we need to define the prior. The simplest choice is the standard Gaussian:[1]

$$\pi(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I}) = \frac{e^{-\frac{1}{2}\mathbf{z}'\mathbf{z}}}{\sqrt{(2\pi)^d}} \tag{5.3}$$

5.1.3 Expectation (L1 norm)

Since our logarithmic representation is closed under elementwise multiplication, our expectations can always be performed on representations of the

[1] Note here and elsewhere, we use two different fonts to differentiate $\pi(\mathbf{z})$, from the trigonometric constant π.

form (5.1). Dropping the subscript j to avoid clutter, we derive an expression for $E_G(\mathbf{a}, \mathbf{B}) = \langle f \rangle$, where $f(\mathbf{z}) = \exp[\mathbf{a}'\mathbf{z} - \frac{1}{2}\mathbf{z}'\mathbf{B}\mathbf{z}]$:

$$\begin{aligned}
E_G(\mathbf{a}, \mathbf{B}) &= \langle f \rangle \\
&= \int_{\mathbb{R}^d} f(\mathbf{z})\pi(\mathbf{z})\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} f(\mathbf{z})\mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} \frac{\exp\bigl[\mathbf{a}'\mathbf{z} - \frac{1}{2}\mathbf{z}'(\mathbf{I} + \mathbf{B})\mathbf{z}\bigr]}{\sqrt{(2\pi)^d}}\, d\mathbf{z}
\end{aligned} \tag{5.4}$$

If we define $\boldsymbol{\mu} = (\mathbf{I} + \mathbf{B})^{-1}\mathbf{a}$, we can rewrite this as:

$$\begin{aligned}
E_G(\mathbf{a}, \mathbf{B}) &= \langle f \rangle \\
&= \int_{\mathbb{R}^d} \frac{\exp\bigl[\boldsymbol{\mu}'(\mathbf{I} + \mathbf{B})\mathbf{z} - \frac{1}{2}\mathbf{z}'(\mathbf{I} + \mathbf{B})\mathbf{z}\bigr]}{\sqrt{(2\pi)^d}}\, d\mathbf{z} \\
&= \frac{\exp\bigl[\frac{1}{2}\boldsymbol{\mu}'(\mathbf{I} + \mathbf{B})\boldsymbol{\mu}\bigr]}{|\mathbf{I} + \mathbf{B}|^{\frac{1}{2}}} \int_{\mathbb{R}^d} \frac{|\mathbf{I} + \mathbf{B}|^{\frac{1}{2}}}{\sqrt{(2\pi)^d}} \exp\bigl[-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu})'(\mathbf{I} + \mathbf{B})(\mathbf{z} - \boldsymbol{\mu})\bigr]\, d\mathbf{z} \\
&= \frac{\exp\bigl[\frac{1}{2}\boldsymbol{\mu}'(\mathbf{I} + \mathbf{B})\boldsymbol{\mu}\bigr]}{|\mathbf{I} + \mathbf{B}|^{\frac{1}{2}}} \int_{\mathbb{R}^d} \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}, (\mathbf{I} + \mathbf{B})^{-1})\, d\mathbf{z} \\
&= \frac{\exp\bigl[\frac{1}{2}\boldsymbol{\mu}'(\mathbf{I} + \mathbf{B})\boldsymbol{\mu}\bigr]}{|\mathbf{I} + \mathbf{B}|^{\frac{1}{2}}} \\
&= \frac{\exp\bigl[\frac{1}{2}\mathbf{a}'\boldsymbol{\mu}\bigr]}{|\mathbf{I} + \mathbf{B}|^{\frac{1}{2}}} = \frac{\exp\bigl[\frac{1}{2}\mathbf{a}'(\mathbf{I} + \mathbf{B})^{-1}\mathbf{a}\bigr]}{|\mathbf{I} + \mathbf{B}|^{\frac{1}{2}}}
\end{aligned} \tag{5.5}$$

For convenience, let us repeat this formula in logarithmic form:

$$\begin{aligned}
\log E_G(\mathbf{a}, \mathbf{B}) = \log\langle f \rangle &= \frac{1}{2}\mathbf{a}'(\mathbf{I} + \mathbf{B})^{-1}\mathbf{a} - \frac{1}{2}\log|\mathbf{I} + \mathbf{B}| \\
&= \frac{1}{2}\mathbf{a}'\boldsymbol{\mu} - \frac{1}{2}\log|\mathbf{I} + \mathbf{B}|
\end{aligned} \tag{5.6}$$

Both $\boldsymbol{\mu}$ and the determinant $|\mathbf{I} + \mathbf{B}|$ can be found for example by Cholesky decomposition of the positive definite matrix $\mathbf{I} + \mathbf{B}$. As we shall see later, other matrix factorizations, such as eigenvalue decomposition can also be used.

Our pooling and expectation operations can now be combined for LR calculation. For example, when f_1, f_2 are represented by $(\mathbf{a}_1, \mathbf{B}_1)$ and $(\mathbf{a}_2, \mathbf{B}_2)$, the normalized inner product (binary verification LR) between them is given by:

$$\langle \overline{f_1}, \overline{f_2} \rangle = \frac{\langle f_1 f_2 \rangle}{\langle f_1 \rangle \langle f_2 \rangle} = \frac{E_G(\mathbf{a}_1 + \mathbf{a}_2, \mathbf{B}_1 + \mathbf{B}_2)}{E_G(\mathbf{a}_1, \mathbf{B}_1) E_G(\mathbf{a}_2, \mathbf{B}_2)} \tag{5.7}$$

5.1.4 L2 norm

As we have seen, we can represent all LR calculations via pooling and expectation. For non-negative Gaussians, expectation is also L1 norm. However, as (3.7) shows, we can alternatively assemble our LR calculations via the L2 norm. For Gaussian meta-embeddings, the L2 norm is readily calculated via the same expectation mechanism derived above. For an embedding f represented by $(\mathbf{a}, \mathbf{B})$, we have:

$$\|f\|^2 = \langle f, f \rangle = \langle f^2 \rangle = \int_{\mathbb{R}^d} f(\mathbf{z})^2 \pi(\mathbf{z}) \, d\mathbf{z} = E_G(2\mathbf{a}, 2\mathbf{B}) \tag{5.8}$$

Notice that

$$\|\overline{f}\|^2 = \langle \overline{f}, \overline{f} \rangle = \frac{\langle f^2 \rangle}{\langle f \rangle^2} = \frac{E_G(2\mathbf{a}, 2\mathbf{B})}{E_G(\mathbf{a}, \mathbf{B})^2} \tag{5.9}$$

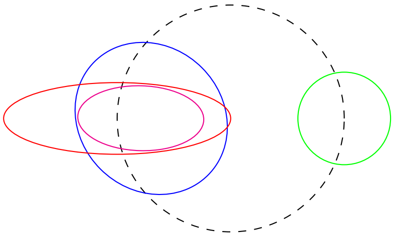
If L2 norm is computed using L1 norm, is there any advantage to the L2 norm? If we use the closed-form expressions, maybe not. But as we shall see in section 6.3.2 below, stochastic approximations to the L2 norm may have advantages relative to stochastic L1 norm approximations.

5.1.5 Graphical multivariate Gaussian examples

Let us again turn to some simple examples to better understand our multivariate Gaussian meta-embeddings. We choose $\mathbf{z} \in \mathcal{Z} = R^2$, so that we can plot the examples. Recall that in the example of section 4.4 we were able to directly plot points in meta-embedding space. Here, that space is infinite-dimensional, so we cannot plot our meta-embeddings as points. But we *can* plot representations of the Gaussian *distributions* in $\mathcal{Z}$-space.

In our plots below, every meta-embedding is shown as an ellipse, centered at the mean of the Gaussian. The ellipse represents the $\sigma = 1$ contour so that it is elongated in directions where the meta-embedding has less certainty about the location of $\mathbf{z}$, while the ellipse is more compressed in directions of greater certainty.[2]

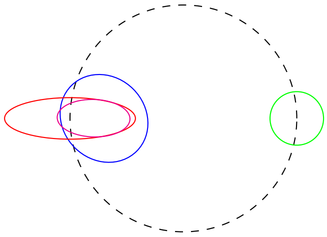
The figure below shows some hypothetical meta-embeddings and the LRs between them, computed using (5.5) in (4.1), or (4.2). The standard normal prior is shown in dashed black. Blue, red and green represent raw embeddings, while magenta represents blue and red pooled, where pooling is done by adding natural parameters ($\mathbf{a}_{\text{mag}} = \mathbf{a}_{\text{blue}} + \mathbf{a}_{\text{red}}$, and $\mathbf{B}_{\text{mag}} = \mathbf{B}_{\text{blue}} + \mathbf{B}_{\text{red}}$):



$\langle \overline{f_i}, \overline{f_j} \rangle$	blue	red	magenta
green	0.37	0.80	0.23
blue		2.34	

We can imagine that green was extracted from a long, clean recording; blue from a long noisy recording; and red from a short, clean recording. The short red recording gives reasonable certainty along the vertical axis, but crucial information is missing, resulting in greater uncertainty on the horizontal axis. Notice that $\text{LR}(\text{blue}, \text{green}) < \text{LR}(\text{red}, \text{green})$, even though the blue mean is closer to green than the red mean. This is due to the greater red uncertainty in the horizontal direction. Pooling (magenta) dramatically reduces the uncertainty about the location of $\mathbf{z}$ for the speaker represented by red and blue, and consequently increases the certainty that this speaker is *not* the same as the green speaker: $\text{LR}(\text{magenta}, \text{green})$ is smaller than all the other LRs. There is enough overlap between red and blue to give some support for the hypothesis that red and blue are of the same speaker: $\text{LR}(\text{red}, \text{blue}) > 1$.

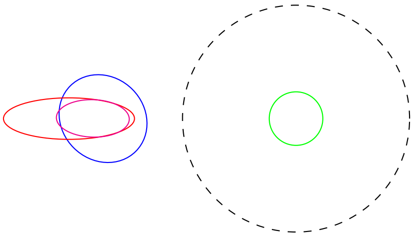
Next, we repeat the example, keeping everything the same, except that all the embeddings have smaller uncertainty. The effect is that the LRs also become more certain, moving further away from the neutral value of 1:



$\langle \overline{f_i}, \overline{f_j} \rangle$	blue	red	magenta
green	0.008	0.069	7.4e-4
blue		5.264	

[2] The ellipse axes are aligned with the eigenvectors of the covariance, and the radii are the square roots of the eigenvalues. MATLAB code that computes LLRs and outputs Ti*k*Z code for plotting the ellipses in LaTeX is available here: `http://github.com/bsxfan/meta-embeddings/tree/master/code/Niko/matlab`.

Finally, we change the example by shifting the meta-embeddings left, relative to the prior. This affects all the LRs:

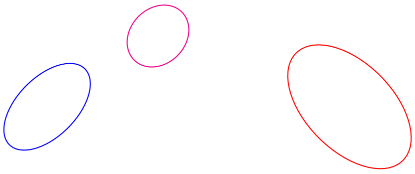


$\langle \overline{f_i}, \overline{f_j} \rangle$	blue	red	magenta
green	0.015	0.162	0.0013
blue		14.260	

The most dramatic effect of the above shift is on LR(red, blue), which has increased threefold due to the increased rarity of the red and blue embeddings, relative to the prior.

Food for thought

Let us examine pooling in more detail. In the above examples, when we pooled the suboptimal information extracted by the (short) red and (noisy) blue recordings, the result (magenta) gave less uncertainty about the value of $\mathbf{z}$ for the speaker common to red and blue. Intuition suggests that things are working properly. But now consider below the pooling of red and blue embeddings that are far apart:



$$\text{LR}(\text{blue}, \text{red}) = 0.0047$$

This result (magenta) is perhaps counter-intuitive:

- Why is the magenta mean so far away from the line connecting the red and blue means?
- If the same speaker is observed in two relatively distant locations, should this not cause the resulting uncertainty to *increase*, rather than the decrease we see here?

Nevertheless, we have followed the theory and the magenta result is correct. One reason why our intuition has a hard time to accept this result is because pooling is conditioned on the assumption that the speakers are the same, yet $\text{LR}(\text{blue}, \text{red}) \ll 1$ gives strong support to the hypothesis that this is not so. If we were to encounter this situation in practice, it would suggest that the same-speaker assumption is wrong, or that the uncertainties represented in the meta-embeddings have been underestimated.

The apparently wayward magenta mean is the location where the blue and red distributions 'overlap most'. It is located at the maximum of the product of the two distributions. It can be interpreted as a soft intersection between the two distributions.

5.2 SGME: Simple Gaussian meta-embedding

A useful specialization of the GME, which we call *simple Gaussian meta-embedding* (SGME), is obtained when $D = 1$, so that precisions of different meta-embeddings differ by a single scale factor. As in the general case, we have $\mathbf{a}_j = \mathbb{R}^d$, but there is only a single precision weight, which we denote with b_j, so that the precision is:

$$\mathbf{B}_j = b_j\mathbf{E} \tag{5.10}$$

where $\mathbf{E}$ is a constant, d-by-d, positive semi-definite matrix. As we shall see in chapter 7, generative PLDA models produce SGMEs. An advantage to SGMEs is that we can find faster recipes for computing expectations.

5.2.1 Expectation (L1 norm)

To compute expectations using (5.6), we need factorizations of

$$\mathbf{B} = \mathbf{I} + b\mathbf{E} \tag{5.11}$$

where b varies but $\mathbf{E}$ remains fixed. It turns out we do not have to redo the matrix factorization for every b. Since $\mathbf{E}$ is symmetric positive semi-definite, we can factorize it as:

$$\mathbf{E} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}' \tag{5.12}$$

where $\mathbf{V}$ is an orthonormal matrix having the eigenvectors along its columns, while $\boldsymbol{\Lambda}$ is a diagonal matrix of real, non-negative eigenvalues. By orthonormality we have $\mathbf{V}^{-1} = \mathbf{V}'$, so that:

$$\mathbf{B} = \mathbf{I} + b\mathbf{E} = \mathbf{V}(\mathbf{I} + b\boldsymbol{\Lambda})\mathbf{V}' \tag{5.13}$$

This means $\mathbf{B}$ has a factorization of the same form as $\mathbf{E}$, so that the eigenvectors remain fixed, for any $b \geq 0$. Denote the eigenvalues of $\mathbf{E}$ as $\{\lambda_i\}_{i=1}^d$, then the corresponding eigenvalues of $\mathbf{I} + b\mathbf{E}$ are $\{1 + b\lambda_i\}_{i=1}^d$. The log-determinant—the second term of (5.6)—is just:

$$\log|\mathbf{I} + b\mathbf{E}| = \sum_{i=1}^{d} \log(1 + b\lambda_i) \tag{5.14}$$

Inversion is also simple:

$$(\mathbf{I} + b\mathbf{E})^{-1} = \mathbf{V}(\mathbf{I} + b\boldsymbol{\Lambda})^{-1}\mathbf{V}' \tag{5.15}$$

which gives the other term of (5.6) as:

$$\mathbf{a}'(\mathbf{I} + b\mathbf{E})^{-1}\mathbf{a} = \mathbf{a}'\mathbf{V}(\mathbf{I} + b\boldsymbol{\Lambda})^{-1}\mathbf{V}'\mathbf{a} = \sum_{i=1}^{d} \frac{\tilde{a}_i^2}{1 + b\lambda_i} \tag{5.16}$$

where the $\tilde{a}_i$ are the components of $\tilde{\mathbf{a}} = \mathbf{V}'\mathbf{a}$. Assembling everything gives:

$$\log E_G(\mathbf{a}, b) = \log\langle f \rangle = \frac{1}{2} \sum_{i=1}^{d} \frac{\tilde{a}_i^2}{1 + b\lambda_i} - \log(1 + b\lambda_i) \tag{5.17}$$

Diagonalized SGME

Another way to derive the same calculation is to transform the hidden variable as $\tilde{\mathbf{z}} = \mathbf{V}'\mathbf{z}$. The Jacobian determinant magnitude of this transform is unity, so that it leaves the standard normal prior unchanged and it diagonalizes all precision matrices. Then the meta-embedding in the transformed space is:

$$f(\tilde{\mathbf{z}}) = f(\mathbf{z}) = \exp[\tilde{\mathbf{z}}'\tilde{\mathbf{a}} - \frac{1}{2}\tilde{\mathbf{z}}'\tilde{\mathbf{B}}\tilde{\mathbf{z}}] \tag{5.18}$$

with the natural parameters:

$$\tilde{\mathbf{a}} = \mathbf{V}'\mathbf{a} \qquad \text{and} \qquad \tilde{\mathbf{B}} = \mathbf{V}'\mathbf{B}\mathbf{V} = b\boldsymbol{\Lambda} \tag{5.19}$$

5.3 DGME: Diagonal GME

A mild generalization of the diagonalized SGME—with similar computational requirements—is the *diagonal Gaussian meta-embedding* (DGME), where we restrict the precisions, $\mathbf{B}_j$, to be diagonal. In the canonical GME representation, we could express the diagonal precision as $\mathbf{B}_j = \sum_{i=1}^{D} b_{ij}\mathbf{E}_i$, where all the $\mathbf{E}_i$ are diagonal and $D \leq d$. For the full dimensionality, $D = d$, which we expect would be used by default, we don't need the $\mathbf{E}_i$ and we can represent $\mathbf{B}_j$ as the diagonal matrix with diagonal $\mathbf{b}_j \in \mathbb{R}^d$.

Expectation (L1 norm)

The expected value of a DGME, $f(\mathbf{z}) = \exp[\mathbf{z}'\mathbf{a} - \frac{1}{2}\mathbf{z}'\mathbf{B}\mathbf{z}]$, where $\mathbf{B}$ is diagonal, with diagonal elements b_i, is given by:

$$\log E_G(\mathbf{a}, \mathbf{b}) = \log\langle f \rangle = \frac{1}{2}\sum_{i=1}^{d} \frac{a_i^2}{1 + b_i} - \log(1 + b_i) \tag{5.20}$$

5.4 Autoregressive Gaussians

The autoregressive Gaussian GME (AR-GME) is a multivariate Gaussian represented in a way that retains the additive pooling of the GME representation, but it facilitates sampling for approximate expectation computation. In the GME representation, $f(\mathbf{z}) = \exp[\mathbf{a}'\mathbf{z} - \frac{1}{2}\mathbf{z}'\mathbf{B}\mathbf{z}]$, Cholesky decomposition of $\mathbf{B}$ is required for sampling. We define the AR-GME representation as:

$$\begin{aligned} f(\mathbf{z}) &= \prod_{i=1}^{d} \exp\left[z_i a_i - z_i \sum_{k=1}^{i} \rho_{ik} z_k\right] && (5.21) \\ &= \exp\left[\sum_{i=1}^{d} z_i a_i - z_i \sum_{k=1}^{i} \rho_{ik} z_k\right] && (5.22) \\ &= \exp\left[\mathbf{z}'\mathbf{a} - \mathbf{z}'\mathbf{R}\mathbf{z}\right] && (5.23) \\ &= \exp\left[\mathbf{z}'\mathbf{a} - \frac{1}{2}\mathbf{z}'(\mathbf{R} + \mathbf{R}')\mathbf{z}\right] && (5.24) \end{aligned}$$

where $\mathbf{R}$ is a lower triangular matrix with strictly positive entries on the diagonal. The last equality is obtained via $\mathbf{z}'\mathbf{R}\mathbf{z} = (\mathbf{z}'\mathbf{R}\mathbf{z})' = \mathbf{z}'\mathbf{R}'\mathbf{z}$.

5.5 Complex Gaussians

This section is incomplete. First, we need to establish whether the Fourier transform (or equivalently characteristic function) of a complex Gaussian[3] is again a complex Gaussian function. I don't know if this is true or not. For ordinary (real) multivariate Gaussians, it is not true: the transform of a zero-mean Gaussian is another Gaussian, but for a non-zero mean the transform has a more complex form.

If it turns out that complex Gaussians are indeed closed under Fourier transform, we can do business. We can let our speaker identity variables be

[3] http://en.wikipedia.org/wiki/Complex_normal_distribution

complex-valued vectors and our meta-embeddings can be complex Gaussians. The convolution theorem shows we can approximate pooling stochastically in the transform domain, by simple addition of samples drawn from the complex Gaussians in the frequency domain. By Parseval's theorem, for expectations, we don't have to transform back after pooling—we can compute the stochastic averages directly in the frequency domain. See appendix A for some notes on the multivariate Fourier transform.

5.6 MoGME: Mixture of GME

A mixture of Gaussians gives a more powerful representation that retains some of the advantages of the GME. For a moderate number of mixture components, we retain tractability of LR calculations, but pooling is not so straight-forward.

5.6.1 Definition and normalization

Let $\overline{\text{GME}}_{jk}$ denote a *normalized* Gaussian meta-embedding,[4] with parameters indexed by jk. We define a *mixture of Gaussian meta-embeddings* (MoGME) of order K as:

$$f_j(\mathbf{z}) = \sum_{k=1}^{K} w_{jk} \overline{\text{GME}}_{jk}(\mathbf{z}) \tag{5.25}$$

where every $w_{jk} > 0$. As with any meta-embedding, the scaling is unimportant, so we do not insist that $\sum_{k=1}^{K} w_{jk} = 1$. However, the *relative* scaling of the components *is* important, so we do insist on having normalized components. Normalization of the whole MoGME is done by weight normalization:

$$\overline{f_j} = \frac{f_j}{\sum_{k=1}^{K} w_{jk}} \tag{5.26}$$

[4] Remember, a normalized meta-embedding is *not* a normalized probability distribution. In fact, it need not even be normalizable. The meta-embedding is normalized when its product with the prior is a normalized distribution.

5.6.2 Likelihood ratio

The LR, or normalized inner product, can be computed using the GME inner product (5.7):

$$\begin{aligned}\langle \overline{f}_i, \overline{f}_j \rangle &= \frac{\bigl\langle (\sum_{k=1}^K w_{ik}\overline{\text{GME}}_{ik})(\sum_{\ell=1}^K w_{j\ell}\overline{\text{GME}}_{j\ell}) \bigr\rangle}{(\sum_{k=1}^K w_{ik})(\sum_{\ell=1}^K w_{j\ell})} \\ &= \frac{\sum_{k=1}^K \sum_{\ell=1}^K w_{ik}w_{j\ell}\langle \overline{\text{GME}}_{ik}, \overline{\text{GME}}_{j\ell} \rangle}{(\sum_{k=1}^K w_{ik})(\sum_{\ell=1}^K w_{j\ell})}\end{aligned} \tag{5.27}$$

The complexity scales as K^2.

5.6.3 Pooling

Pooling (product) of two MoGMEs of order K gives a MoGME of order K^2, where the resultant weights have an interesting form. To derive the weight formula, notice that for any normalized meta-embeddings, $\overline{f}, \overline{g}$, renormalizing their product gives:

$$\overline{\overline{f}\,\overline{g}} = \frac{\overline{f}\,\overline{g}}{\langle \overline{f}\,\overline{g}, \mathbf{1} \rangle} = \frac{\overline{f}\,\overline{g}}{\langle \overline{f}, \overline{g} \rangle} \tag{5.28}$$

We use this to write the MogME product in the required form with normalized components:

$$\begin{aligned} f_i f_j &= \sum_{k=1}^K \sum_{\ell=1}^K w_{ik}w_{j\ell}\overline{\text{GME}}_{ik}\overline{\text{GME}}_{j\ell} \\ &= \sum_{k=1}^K \sum_{\ell=1}^K \Bigl(w_{ik}w_{j\ell}\langle \overline{\text{GME}}_{ik}, \overline{\text{GME}}_{j\ell} \rangle \Bigr) \overline{\overline{\text{GME}}_{ik}\overline{\text{GME}}_{j\ell}} \end{aligned} \tag{5.29}$$

where the parenthesis shows the new weight for component $k\ell$. Pairs of matching components get large weights, while mismatched pairs get smaller weights.

To avoid the polynomial explosion of complexity associated with pooling, the possibility exists to form approximate products that retain only K terms. The weights could be used to select the best K. More sophisticated methods could numerically optimize the normalized inner product between full and approximate representations, but this would have K^3 complexity per iteration.

5.7 Exponential family Gaussian mixture

The multivariate Gaussian of the previous section is of course an exponential family distribution. We can derive very similar recipes—also with logarithmic representations—from other exponential families. We develop one such example here.

Although a Gaussian mixture is not an exponential family, the *joint* distribution for the continuous and discrete (state) variable *is* exponential family. We do this with a D-component mixture of d-dimensional Gaussians, by choosing our hidden speaker identity variable as: $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, \ldots, x_D)$ is a one-hot vector of size D, while $\mathbf{y} \in \mathbb{R}^d$. Our meta-embedding for recording j is:

$$f_j(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{D} \Bigl(w_{ij} \exp\bigl[\mathbf{a}'_{ij}\mathbf{y} - \frac{1}{2}\mathbf{y}'\mathbf{B}_{ij}\mathbf{y}\bigr] \Bigr)^{x_i} \tag{5.30}$$

where $x_i \in \{0, 1\}$, $w_{ij} > 0$, $\mathbf{a}_{ij} \in \mathbb{R}^d$ and the $\mathbf{B}_{ij}$ are d-by-d positive semi-definite. To see that this is exponential family, take the logarithm and rearrange:

$$\log f_j(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{D} x_i \log w_{ij} + (x_i\mathbf{y}')\mathbf{a}_{ij} - \frac{1}{2} \operatorname{tr}\bigl[(x_i\mathbf{y}\mathbf{y}')\mathbf{B}_{ij}\bigr] \tag{5.31}$$

where we see the sufficient statistics are: $\{x_i, x_i\mathbf{y}, x_i\mathbf{y}\mathbf{y}'\}_{i=1}^{D}$ and the natural parameters are $\{\log w_{ij}, \mathbf{a}_{ij}, \mathbf{B}_{ij}\}_{i=1}^{D}$. To form more compact representations, we can let the natural parameters be linear functions of smaller vectors (of which some components have to be constrained to be non-negative). As above, elementwise multiplication (pooling) is accomplished by simple vector addition of these representations.

5.7.1 Prior

We choose a parameterless prior, of the same form as the meta-embeddings:

$$\pi(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{D} \Bigl(\frac{\mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{I})}{D} \Bigr)^{x_i} \tag{5.32}$$

and

$$\begin{aligned} \log \pi(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{D} x_i \bigl(-\frac{1}{2}\mathbf{y}'\mathbf{y} - \frac{d}{2}\log(2\pi) - \log(D) \bigr) \\ &= -\frac{d}{2}\log(2\pi) - \log(D) + \sum_{i=1}^{D} -\frac{1}{2} \operatorname{tr}\bigl[(x_i\mathbf{y}\mathbf{y}')\mathbf{I}\bigr] \end{aligned} \tag{5.33}$$

As in the multivariate Gaussian case, the practical function of the prior is to add $\mathbf{I}$ to the possibly semi-definite matrices, $\mathbf{B}_{ij}$, of the meta-embeddings, to make them properly positive definite.

5.7.2 Expectation

Dropping the subscript j, let $f(\mathbf{x},\mathbf{y}) = \prod_{i=1}^D \Bigl(w_i \exp\bigl[\mathbf{a}_i'\mathbf{y} - \frac12\mathbf{y}'\mathbf{B}_i\mathbf{y}\bigr]\Bigr)^{x_i}$, then:[5]

$$
\begin{aligned}
E\bigl(\{w_i,\mathbf{a}_i,\mathbf{B}_i\}_{i=1}^D\bigr) &= \langle f\rangle \\
&= \sum_{i=1}^D \frac{w_i}{D}\int_{\mathbb{R}^d} f(\mathbf{x}=i,\mathbf{y})\mathcal{N}(\mathbf{y}\mid\mathbf{0},\mathbf{I})\,d\mathbf{y} \\
&= \sum_{i=1}^D \frac{w_i}{D}\int_{\mathbb{R}^d} \exp\bigl[\mathbf{a}_i'\mathbf{y} - \frac{1}{2}\mathbf{y}'\mathbf{B}_i\mathbf{y}\bigr]\mathcal{N}(\mathbf{y}\mid\mathbf{0},\mathbf{I})\,d\mathbf{y} \\
&= \sum_{i=1}^D \frac{w_i}{D}\frac{\exp\bigl[\frac12\mathbf{a}_i'\boldsymbol{\mu}_i\bigr]}{|\mathbf{I}+\mathbf{B}_i|^{\frac12}}
\end{aligned}
\tag{5.34}
$$

where $\boldsymbol{\mu}_i = (\mathbf{I}+\mathbf{B}_i)^{-1}\mathbf{a}_i$.

This computation is much the same as for the multivariate Gaussian case, except that we have to do D such calculations every time. Obviously, if we want to use a large value for D, then the individual calculations need to be cheap. Since Cholesky decomposition of full matrices $\mathbf{I}+\mathbf{B}_i$ requires $\mathcal{O}(d^3)$ computation, we need to simplify these calculations. There are various way to do that. We can let the $\mathbf{B}_{ij}$ differ from each other by scaling, by low-rank modifications, or by forming them via Kronecker products, etc.

5.8 Mixtures

Let us compare the previous solution with one where the state is not considered part of the speaker identity variable. The meta-embedding for recording r_j is:

$$
f_j(\mathbf{z}) = \sum_{i=1}^D w_{ij} \tag{5.35}
$$

[5] We abuse notation by writing $\mathbf{x} = i$ to indicate that component i is the hot element (value 1) in the otherwise zero one-hot vector, $\mathbf{x}$.

5.9 Free form, inspired by exponential family distribution

5.10 Discrete Factorial

5.11 Mixture with fixed components

5.12 Mixture with shifted components

5.13 Kernel approximation

Let us think in terms of inner products, rather than expectations. If we want fast LR computation, we need fast inner product computation.

5.14 Mean embedding

Chapter 6

Approximate computation

Meta-embeddings are more powerful than existing methods and therefore potentially more accurate, but the required calculation of inner products or expectations may be too slow to be useful in practice, unavailable in closed form, and maybe even completely intractable. Even the multivariate Gaussian meta-embeddings, for which we have tractable closed-form calculations are a lot slower than traditional Gaussian PLDA (GPLDA) scoring. In this chapter we examine some proposals for fast approximations. For reference below, we briefly derive the GPLDA scoring function.

6.1 The GPLDA score

When recordings are represented by i-vectors, which are in turn modelled by Gaussian PLDA, then a Gaussian meta-embedding (GME), with *constant precision*, can be extracted in closed form from every i-vector. In this case, no neural net needs to be trained to extract the meta-embeddings. Each meta-embedding is extracted with a simple function of the i-vector and the GPLDA parameters. (This formula is given by (7.7) below, by letting $\nu \to \infty$.) The GPLDA parameters can be generatively, or discriminatively trained.

Adapting the notation of section 5.1, for the case $D = 1$, the GPLDA GMEs have $\mathbf{B}_j = b_j\mathbf{E}$. At extraction time (before any pooling) all raw GMEs given by the PLDA model will have the same precision, so that we can conveniently set all $b_j = 1$. This leads to a fast, quadratic scoring recipe. Let the raw embeddings f_i, f_j be respectively represented by $\mathbf{a}_i$ and $\mathbf{a}_j$, then by (5.7) the log-LR between them is:

$$\begin{aligned}\log\langle \overline{f_i}, \overline{f_j}\rangle &= \log E_G(\mathbf{a}_i + \mathbf{a}_j, 2\mathbf{E}) - \log E_G(\mathbf{a}_i, \mathbf{E}) - \log E_G(\mathbf{a}_j, \mathbf{E}) \\ &= (\mathbf{a}_i + \mathbf{a}_j)'\mathbf{K}(\mathbf{a}_i + \mathbf{a}_j) - \mathbf{a}_i'\mathbf{L}\mathbf{a}_i - \mathbf{a}_j'\mathbf{L}\mathbf{a}_j + C\end{aligned} \tag{6.1}$$

where C is a constant representing the log-determinant terms and

$$\mathbf{K} = (\mathbf{I} + 2\mathbf{E})^{-1} \qquad \text{and} \qquad \mathbf{L} = (\mathbf{I} + \mathbf{E})^{-1} \tag{6.2}$$

6.2 Taylor series approximations

In this section, we assume that we have available a slow, but closed-form scoring recipe (e.g. GME) to compute the log-LR, $\log\langle\overline{f},\overline{g}\rangle$. Second-order Taylor series approximations of $\log\langle\overline{f},\overline{g}\rangle$ will give us scoring recipes with the same functional form—and therefore the same speed—as the quadratic GPLDA score above.

Let f, g be represented respectively by $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$ and let the scoring function be denoted:

$$F(\mathbf{x},\mathbf{y}) = \log\langle\overline{f},\overline{g}\rangle \tag{6.3}$$

We assume that F can be evaluated and differentiated at least twice. There are at least two ways to form second-order Taylor series approximations. We consider symmetric and asymmetric variants.

6.2.1 Symmetric Taylor series approximation

Define $\mathbf{t}$ as the stacked vector containing both sides: $\mathbf{t} = \begin{bmatrix}\mathbf{x}' & \mathbf{y}'\end{bmatrix}'$. We do the second-order expansion about some convenient, fixed value $\mathbf{t}_0 = \begin{bmatrix}\mathbf{x}_0' & \mathbf{y}_0'\end{bmatrix}'$. The approximation is:

$$\begin{aligned} F(\mathbf{x},\mathbf{y}) &\approx F(\mathbf{x}_0,\mathbf{y}_0) + (\mathbf{t}-\mathbf{t}_0)'\nabla_{\mathbf{t}}(\mathbf{x}_0,\mathbf{y}_0) \\ &\quad + \frac{1}{2}(\mathbf{t}-\mathbf{t}_0)'\mathbf{H}_{\mathbf{t}}(\mathbf{x}_0,\mathbf{y}_0)(\mathbf{t}-\mathbf{t}_0) \end{aligned} \tag{6.4}$$

where $\nabla_{\mathbf{t}}$ and $\mathbf{H}_{\mathbf{t}}$ are gradient and Hessian of F, obtained by differentiating w.r.t. $\mathbf{t}$. Since $\mathbf{t}_0$ is fixed, all of $F(\mathbf{x}_0,\mathbf{y}_0)$, $\nabla_{\mathbf{t}}(\mathbf{x}_0,\mathbf{y}_0)$ and $\mathbf{H}_{\mathbf{t}}(\mathbf{x}_0,\mathbf{y}_0)$ can be precomputed before runtime. The storage requirement is quadratic in the size of $\mathbf{t}$.

It is easy to verify that when this approximation is applied to the PLDA score (6.1), at for example $\mathbf{t}_0 = \mathbf{0}$, we recover the GPLDA score exactly.

6.2.2 Asymmetric Taylor series approximation

A potentially more accurate, but somewhat slower approximation can be obtained by evaluating $F(\mathbf{x},\mathbf{y})$ at the exact, required value of say $\mathbf{x}$ and doing the expansion only w.r.t. $\mathbf{y}$. The approximation is:

$$\begin{aligned}F(\mathbf{x},\mathbf{y}) \approx F(\mathbf{x},\mathbf{y}_0) &+ (\mathbf{y}-\mathbf{y}_0)'\nabla_{\mathbf{y}}(\mathbf{x},\mathbf{y}_0) \\ &+ \frac{1}{2}(\mathbf{y}-\mathbf{y}_0)'\mathbf{H}_{\mathbf{y}}(\mathbf{x},\mathbf{y}_0)(\mathbf{y}-\mathbf{y}_0)\end{aligned} \tag{6.5}$$

This recipe is slower, because for every $\mathbf{x}$ we need a new computation of $F(\mathbf{x},\mathbf{y}_0)$, $\nabla_{\mathbf{y}}(\mathbf{x},\mathbf{y}_0)$ and $\mathbf{H}_{\mathbf{y}}(\mathbf{x},\mathbf{y}_0)$. Of course, this recipe can be symmetrized by doing it also from the other side and averaging.

6.2.3 Taylor series for log expectation

Yet more approximation recipes can be derived by approximating the log expectations, rather than the log LR scores. Consider for example the SGME log expectation of (5.17):

$$\log E_G(\tilde{\mathbf{a}}, b) = \frac{1}{2}\sum_{i=1}^{d} \frac{\tilde{a}_i^2}{1+b\lambda_i} - \log(1+b\lambda_i) \tag{6.6}$$

This expression is quadratic in $\tilde{\mathbf{a}}$, just like for GPLDA, but more complex as a function of b. We can reduce the complexity w.r.t. b by doing a second-order Taylor series expansion w.r.t. b. Otherwise, we can do it w.r.t. a reparametrization, for example in terms of $\gamma = \log(b+c)$. In preliminary experiments, we found that choosing c as the reciprocal of the mean of the eigenvalues of $\mathbf{E}$ gives an almost linear relationship between γ and $\log|\mathbf{I}+b\mathbf{E}|$.

6.2.4 Implementation

In an environment where both complex arithmetic and first-order derivatives of F are available, *complex-step automatic differentiation* can be used (with some caveats) to compute the Hessian to high accuracy, without having to resort to laborious hand-derived and hand-coded second-order differentiation.

6.3 Stochastic approximations

Since our meta-embeddings are essentially probability distributions, we can sample from them in many cases. This is explored below.

6.3.1 Stochastic expectation for GMEs

For the GMEs of section 5.1, pooling is fast (addition), but expectation is slower (Cholesky decomposition). Unfortunately, when precisions vary, this representation requires a new Cholesky decomposition for *every* LR calculation.

In applications where many verification trials are processed on limited hardware, this will have a significant impact on speed. Even if we have enough CPU power available at runtime for some applications, discriminative training of meta-embedding extractors remains a problem because discriminative training criteria typically require evaluating very many trials. Let us therefore consider some plans for speeding up scoring of trials by making stochastic approximations.

We envisage that such stochastic approximation could work out to be especially cheap at training time, in the same way that stochastic evaluation of the ELBO in *variational autoencoder* (VAE) training typically requires but a single stochastic sample per training example [19]. Given sufficiently many examples, the stochastic approximation errors tend to cancel, provided the errors are independent.

We will not be too concerned about expectations of a single factor, of the form $\langle f \rangle$. These expectations are required to normalize trial sides (typically meta-embeddings extracted from single recordings) and there are far fewer trial sides than there are trials. The calculations for which we need speed, are expectations of the form $\langle \overline{f_i}\,\overline{f_j} \rangle$. We consider two proposals below.

Prior sampling

An obvious choice, perhaps naive, would be to form the expectation by sampling from the prior, π. The expectation could be approximated via N such samples as:

$$\langle \overline{f_i}(\mathbf{z})\overline{f_j}(\mathbf{z}) \rangle_{\mathbf{z}\sim\pi} \approx \frac{1}{N} \sum_{\tilde{\mathbf{z}}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \overline{f_i}(\tilde{\mathbf{z}})\overline{f_j}(\tilde{\mathbf{z}}) \tag{6.7}$$

We fear this might be inaccurate, because in high-dimensional space, samples from π will be very unlikely to hit the peak[1] of $\overline{f_i}\,\overline{f_j}$ and therefore an affordable number of samples might not be able to sufficiently explore the volume under the peak. Experiments might be needed to check whether this is indeed a problem. Let us nevertheless explore in more detail the calculation required for each sample:

$$\begin{aligned} \overline{f_i}(\tilde{\mathbf{z}})\overline{f_j}(\tilde{\mathbf{z}}) &= c_i c_j \exp\left[\mathbf{a}_i'\tilde{\mathbf{z}} + \mathbf{a}_j'\tilde{\mathbf{z}} - \frac{1}{2}\tilde{\mathbf{z}}'\mathbf{B}_i\tilde{\mathbf{z}} - \frac{1}{2}\tilde{\mathbf{z}}'\mathbf{B}_j\tilde{\mathbf{z}}\right] \\ &= c_i c_j \exp\left[\mathbf{a}_i'\tilde{\mathbf{z}} + \mathbf{a}_j'\tilde{\mathbf{z}} - \frac{1}{2}\mathbf{b}_i'\tilde{\boldsymbol{\gamma}} - \frac{1}{2}\mathbf{b}_j'\tilde{\boldsymbol{\gamma}}\right] \end{aligned} \tag{6.8}$$

[1] The product is still Gaussian and therefore has single peak.

where using (5.2), we have defined $\tilde{\boldsymbol{\gamma}} = [\tilde{\gamma}_1, \dots, \tilde{\gamma}_D] \in \mathbb{R}^D$, with $\tilde{\gamma}_\ell = \tilde{\mathbf{z}}'\mathbf{E}_\ell\tilde{\mathbf{z}}$ and where c_i, c_j reflect the normalizations. The meta-embedding representations, $\mathbf{a}_i, \mathbf{b}_i$ and $\mathbf{a}_j, \mathbf{b}_j$, and the trainable constants, $\{\mathbf{E}_\ell\}_{\ell=1}^D$, are as defined in the beginning of section 5.1.

Notice that $\tilde{\boldsymbol{\gamma}}$ is independent of the meta-embeddings and therefore independent of the data and can be precomputed. If the $\mathbf{E}_\ell$ are constrained to be either diagonal or rank-one, then the calculations $\tilde{\mathbf{z}}'\mathbf{E}_\ell\mathbf{z}$ will be cheap. Keep in mind that we need N different samples, $\tilde{\mathbf{z}}$, and therefore also N different versions of $\tilde{\boldsymbol{\gamma}}$. In stochastic minibatch training, if we update the $\{\mathbf{E}_\ell\}$ after every minibatch, we will have to update all of the $\tilde{\boldsymbol{\gamma}}$ also, in which case it might make sense to also resample N new values for $\tilde{\mathbf{z}}$. This type of strategy, which is well-known also in VAE [19], is termed *doubly stochastic* by Michalis Titsias [36].

An advantage of (6.8) is that it generalizes to expectations of more (or fewer) than two factors—the argument in the exponent will have independent terms for each of the factors in the expectation.

Experiments would have to be conducted to verify if an affordably small N could give reasonable accuracy. Keep in mind that in such experiments, we can compare accuracy against the exact calculation (5.5).

Posterior sampling

If Gaussian meta-embedding f_i represents recording r_i, then

$$\pi(\mathbf{z})\overline{f_i}(\mathbf{z}) = P(\mathbf{z} \mid r_i, \pi)$$

is a properly normalized Gaussian, from which we can sample.[2] We can now rewrite the expectation as:

$$\left\langle \overline{f_i}(\mathbf{z})\overline{f_j}(\mathbf{z}) \right\rangle_{\mathbf{z}\sim\pi} = \left\langle \overline{f_j}(\tilde{\mathbf{z}}) \right\rangle_{\tilde{\mathbf{z}}\sim\pi\overline{f_i}} \approx \frac{1}{N} \sum_{\tilde{\mathbf{z}}\sim\pi\overline{f_i}} c_j \exp\left[\mathbf{a}_j'\tilde{\mathbf{z}} - \frac{1}{2}\mathbf{b}_j'\tilde{\boldsymbol{\gamma}}\right] \tag{6.9}$$

This might be more accurate than prior sampling, because when f_i and f_j represent the same speaker, their peaks should overlap. If they are of different speakers, their peaks might be far apart, but then we do want a value close to zero for the expectation. With this variant, we may be able to use smaller values for N.

This method is asymmetric. Which chirality is best—should we sample from f_i, or from f_j? If f_i is much sharper than f_j, so that f_j is almost constant

[2] The same is not always true of f_i or $\overline{f_i}$, which might have a singular precision, in which case sampling could not be done.

compared to f_j, then one sample would suffice, while if we sampled the other way round, we would need many samples to properly explore the volume of f_i. Pooling generally sharpens peaks, so it would be more accurate to sample from the side that consists of the most pooled factors. Unfortunately, fast pooling and sampling are conflicting requirements—after adding natural parameters to compute the pooling, we need a Cholesky factorization before we can sample.

Whether we pool or not, a disadvantage of posterior sampling vs prior sampling is that we have reintroduced the need for Cholesky factorization. Fortunately, we need this only for one side of the trial and we are still avoiding per-trial Cholesky decompositions. Also note that for the sampled side of the trial we do not need to separately compute the normalization constant, because the samples already come from the normalized distribution.

Another option would be to change the representation. We could let:

$$\mathbf{B}_i = \Bigl(\sum_{\ell=1}^{D} b_{\ell i}\mathbf{T}_\ell\Bigr)\Bigl(\sum_{\ell=1}^{D} b_{\ell i}\mathbf{T}_\ell\Bigr)' \tag{6.10}$$

where the $\mathbf{T}_\ell$ are triangular and the $b_{\ell i}$ are no longer constrained to be non-negative. This gives a free Cholesky decomposition, and fast, approximate evaluation of LRs of the form $\langle \overline{f_i}, \overline{f_j} \rangle$, but it complicates pooling for more complex LR calculations.

Cholesky sampling

The standard way to sample from a multivariate Gaussian of which the natural parameters, $\mathbf{a}, \mathbf{B}$, are given, is via Cholesky decomposition of the precision matrix, $\mathbf{B}$. Let $\mathbf{C}^{-1} = \mathbf{B} = \mathbf{R}'\mathbf{R}$, where $\mathbf{C}$ is covariance and $\mathbf{R}$ is the (upper-triangular) Cholesky decomposition of $\mathbf{B}$. This gives $\mathbf{C} = \mathbf{R}^{-1}(\mathbf{R}')^{-1}$. Given natural parameter, $\mathbf{a} = \mathbf{B}\boldsymbol{\mu}$, we can recover the mean as:

$$\boldsymbol{\mu} = \mathbf{C}\mathbf{a} = \mathbf{R}^{-1}(\mathbf{R}')^{-1}\mathbf{a} \tag{6.11}$$

If $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a sample from the standard normal, then

$$\boldsymbol{\mu} + \mathbf{R}^{-1}\mathbf{x} = \mathbf{R}^{-1}[(\mathbf{R}')^{-1}\mathbf{a} + \mathbf{x}] \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C}) \tag{6.12}$$

is a sample from the desired distribution. This is implemented via the standard backsubstitution algorithm for the solution of linear equations involving triangular matrices.

Autoregressive sampling

We can do autoregressive sampling by using a cheaper *additive* decomposition of the precision. Let $\mathbf{L}$ be *lower triangular*, such that:

$$\mathbf{B} = \mathbf{L} + \mathbf{L}' \tag{6.13}$$

We obtain $\mathbf{L}$ by copying the lower triangle (including the diagonal) from $\mathbf{B}$ and then dividing the diagonal by 2, while the superdiagonal triangle remains zero. Observing that:

$$\frac{1}{2}\mathbf{z}'\mathbf{B}\mathbf{z} = \frac{1}{2}\mathbf{z}'\mathbf{L}\mathbf{z} + \frac{1}{2}\mathbf{z}'\mathbf{L}'\mathbf{z} = \mathbf{z}'\mathbf{L}\mathbf{z} \tag{6.14}$$

we can rewrite the Gaussian with natural parameters, $\mathbf{a}, \mathbf{B}$ as:

$$\begin{aligned}
\mathcal{N}(\mathbf{z} \mid \mathbf{B}^{-1}\mathbf{a}, \mathbf{B}^{-1}) &= \frac{|\mathbf{B}|^{\frac{1}{2}}}{e^{\frac{1}{2}\mathbf{a}'\mathbf{B}^{-1}\mathbf{a}}} \exp[\mathbf{z}'\mathbf{a} - \frac{1}{2}\mathbf{z}'\mathbf{B}\mathbf{z}] \\
&= \exp[\mathbf{z}'\mathbf{a} - \mathbf{z}'\mathbf{L}\mathbf{z}] \\
&= \exp\Big[\sum_{i=1}^{d} z_i(a_i - \sum_{j=1}^{i} \ell_{ij} z_j)\Big] \\
&= \exp\Big[\sum_{i=1}^{d} z_i(a_i - \sum_{j=1}^{i-1} \ell_{ij} z_j) - \frac{1}{2}(2\ell_{ii}) z_i^2\Big] \\
&= \exp\Big[\sum_{i=1}^{d} z_i(a_i - \sum_{j=1}^{i-1} b_{ij} z_j) - \frac{1}{2} b_{ii} z_i^2\Big] \\
&= \prod_{i=1}^{d} \exp\Big[z_i(a_i - \sum_{j=1}^{i-1} b_{ij} z_j) - \frac{1}{2} b_{ii} z_i^2\Big] \\
&\propto \prod_{i=1}^{d} \mathcal{N}(z_i \mid b_{ii}^{-1}(a_i - \sum_{j=1}^{i-1} b_{ij} z_j), b_{ii}^{-1})
\end{aligned} \tag{6.15}$$

where z_i, a_i, ℓ_{ij} and b_{ij} are elements of $\mathbf{z}, \mathbf{a}, \mathbf{L}$ and $\mathbf{B}$. The RHS is in autoregressive form. If $\{x_i\}_{i=1}^{d}$ are independent standard normal samples, then sampling can be done as:

$$z_i = \frac{a_i - \sum_{j=1}^{i-1} b_{ij} z_j}{b_{ii}} + \frac{x_i}{\sqrt{b_{ii}}} \tag{6.16}$$

6.3.2 Stochastic L2 norm

The above stochastic expectation recipes are fragile because the distributions from which we sample can be mismatched to the arguments of the expectations. If we instead do stochastic L2 norm, we can make use of the symmetry to combat this problem. The analysis in this section is applicable more generally, not just to multivariate Gaussian meta-embeddings.

Using (3.7), we want to do the following calculation:

$$\langle \overline{f}, \overline{g} \rangle = \frac{1}{2}\big(\|\overline{f}+\overline{g}\|^2 - \|\overline{f}\|^2 - \|\overline{g}\|^2\big) \tag{6.17}$$

Again, we assume that we can afford to compute each $\|\overline{f}\|^2$ and $\|\overline{g}\|^2$ in closed form, using (5.9). However, we need a faster way to compute $\|\overline{f}+\overline{g}\|^2$. Keep in mind that $\pi(\mathbf{z})\overline{f}(\mathbf{z})$ and $\pi(\mathbf{z})\overline{g}(\mathbf{z})$ are normalized probability densities from which we can sample. We can also sample from the *mixture*:

$$\mathcal{M}(\mathbf{z}) = \frac{1}{2}\pi(\mathbf{z})\big(\overline{f}(\mathbf{z}) + \overline{g}(\mathbf{z})\big) \tag{6.18}$$

Expanding the term of interest, we find:

$$\begin{aligned}
&\frac{1}{2}\|\overline{f}+\overline{g}\|^2 \\
&= \int_{\mathbb{R}^d} \frac{1}{2}\pi(\mathbf{z})\big(\overline{f}(\mathbf{z}) + \overline{g}(\mathbf{z})\big)^2 \, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} \mathcal{M}(\mathbf{z})\big(\overline{f}(\mathbf{z}) + \overline{g}(\mathbf{z})\big) \, d\mathbf{z} \\
&= \frac{1}{2}\int_{\mathbb{R}^d} \pi(\mathbf{z})\overline{f}(\mathbf{z})\big(\overline{f}(\mathbf{z}) + \overline{g}(\mathbf{z})\big) \, d\mathbf{z} + \frac{1}{2}\int_{\mathbb{R}^d} \pi(\mathbf{z})\overline{g}(\mathbf{z})\big(\overline{f}(\mathbf{z}) + \overline{g}(\mathbf{z})\big) \, d\mathbf{z} \\
&\approx \frac{1}{N}\sum_{\tilde{\mathbf{z}}\sim\pi\overline{f}} \frac{\overline{f}(\tilde{\mathbf{z}}) + \overline{g}(\tilde{\mathbf{z}})}{2} + \frac{1}{N}\sum_{\tilde{\mathbf{z}}\sim\pi\overline{g}} \frac{\overline{f}(\tilde{\mathbf{z}}) + \overline{g}(\tilde{\mathbf{z}})}{2}
\end{aligned} \tag{6.19}$$

where N samples are drawn from each distribution. If f and g are both sharp relative to the prior, then the sampling distribution and argument of the expectation are very well matched. Even if f or g are not so sharp, then the peaks of $\mathcal{M}(\mathbf{z})$ will be sharper, which is the good way round.

In a variant of this recipe, we could also compute $\|\overline{f}\|^2$ and $\|\overline{g}\|^2$ stochastically. If we use the same samples everywhere, cancellation of terms gives:

$$\begin{aligned}
\langle \overline{f}, \overline{g} \rangle &= \frac{1}{2}\|\overline{f}+\overline{g}\|^2 - \frac{1}{2}\|\overline{f}\|^2 - \frac{1}{2}\|\overline{g}\|^2 \\
&\approx \frac{1}{N}\sum_{\tilde{\mathbf{z}}\sim\pi\overline{f}} \frac{\overline{g}(\tilde{\mathbf{z}})}{2} + \frac{1}{N}\sum_{\tilde{\mathbf{z}}\sim\pi\overline{g}} \frac{\overline{f}(\tilde{\mathbf{z}})}{2}
\end{aligned} \tag{6.20}$$

which is just a symmetrized application of (6.9). If the matched sampling recipe (6.19) is beneficial, then the above symmetrized cross-sampling should presumably enjoy the same benefits.

6.4 Laplace approximation for pooling

Here we show how the *Laplace approximation* [37, 38] can be applied to continuous unimodal meta-embeddings, where closed-form pooling is unavailable, but where we can afford an iterative calculation to find an approximate pooled meta-embedding. An example would be the t-distribution meta-embeddings which will be discussed in section 7.1 below.

For meta-embeddings, $f_1, \dots, f_n$, that we want to pool, define the function:

$$F(\mathbf{z}) = -\sum_{i=1}^{n} \log f_i(\mathbf{z}) \tag{6.21}$$

Use numerical optimization to find $\hat{\mathbf{z}} = \operatorname{argmin} F(\mathbf{z})$. Find the Hessian, $\mathbf{H}$, at $\hat{\mathbf{z}}$. The gradient should be zero, so that the second-order Taylor-series approximation about $\hat{\mathbf{z}}$ is:

$$F(\mathbf{z}) \approx F(\hat{\mathbf{z}}) + \frac{1}{2}(\mathbf{z}-\hat{\mathbf{z}})'\mathbf{H}(\mathbf{z}-\hat{\mathbf{z}}) \tag{6.22}$$

Exponentiating this gives a Gaussian, which is the Laplace approximation to the pooled meta-embedding:

$$e^{-F(\mathbf{z})} \approx \exp[-\frac{1}{2}(\mathbf{z}-\hat{\mathbf{z}})'\mathbf{H}(\mathbf{z}-\hat{\mathbf{z}}) + \text{const}] \tag{6.23}$$

6.5 Multiclass classification

Multiclass classification can be scored with (4.3). The likelihood that a test recording, r_j, represented by meta-embedding f_j belongs to class i, for which we have as training examples a set of meta-embeddings, $\mathcal{F}_i$, is:

$$L_{ij} = \langle \overline{g_j}, \overline{f_i} \rangle, \qquad \text{where} \qquad f_i = \prod_{f\in\mathcal{F}_i} f \tag{6.24}$$

Here we are interested in the case where for every class i, we have available a large set of recordings, so that the uncertainty in f_i becomes very small. For the continuous case, we assume that $\overline{f_i}(\mathbf{z})\pi(\mathbf{z}) \approx \delta(\mathbf{z} - \hat{\mathbf{z}}_i)$, where δ denotes the Dirac delta impulse. In this case we have:

$$L_{ij} \approx \int_{\mathcal{Z}} \overline{g_j}(\mathbf{z})\delta(\mathbf{z} - \hat{\mathbf{z}}_i)\, d\mathbf{z} = \overline{g_j}(\hat{\mathbf{z}}_i) \tag{6.25}$$

For discrete $\mathcal{Z}$ we get the same result, if f_i concentrates at a single element of $\mathcal{Z}$.

6.5.1 Open-set vs closed-set

For N classes, each of which has a large training set, we can use the above L_{ij} as-is for closed-set classification scores. For open-set classification, all we need is another class with no training data. For that class, say $i = N+1$, we would have $f_{N+1} = \mathbf{1}$ and when scored against test g_j, we get $L_{(N+1)j} = \langle \overline{g_j}, \mathbf{1} \rangle = 1$. (Note that here we do not use the approximation, because in this case, f_{N+1} does not concentrate—in fact, it gives equal likelihood to all elements.)

For the open-set case, since we are comparing all the other likelihoods to $L_{(N+1)j} = 1$, the normalization is important, so we need: $L_{ij} \approx \overline{g_j}(\hat{\mathbf{z}}_i)$. On the other hand, in the closed-set case, all likelihoods can be scaled by an arbitrary factor dependent on j, but independent of i. In the closed-set case, we can use $L_{ij} \approx \overline{g_j}(\hat{\mathbf{z}}_i) \propto g_j(\hat{\mathbf{z}}_i)$ if that turns out to be more convenient.

Caveat

A caveat is in order. For the open-set score to work properly in practice, we need a good prior, $\pi(\mathbf{z})$. In a PLDA speaker recognizer for D-dimensional i-vectors, and $\mathcal{Z} = \mathbb{R}^d$, where $d < D$, the prior effectively spans a d-dimensional subspace which best represents all of the speakers seen in training, where the number of training speakers is always greater than d. In spoken language recognition, there are typically very few training languages [39], not enough to span a subspace that would give a good prior for new, unseen languages.

Chapter 7

Generative meta-embeddings

In this chapter we analyze a few simple generative models that give closed-form meta-embeddings. Although simple, these models could have practical benefit, for example as i-vector scoring backends. Moreover, the material in this chapter forms a tutorial to help us better understand meta-embeddings before venturing into the novel domain of discriminative meta-embeddings in the next chapter.

7.1 PLDA with heavy-tailed noise

In this section, we analyze a variant of heavy-tailed PLDA [15] that is simple enough to allow closed-form derivation of the meta-embeddings, yet complex enough to demonstrate data-dependent uncertainty. Although these meta-embeddings have a closed form, the LRs computed from them do not. However, we show how to construct approximations that could serve as practical alternatives to the popular combination [34] of length-normalized i-vectors with Gaussian PLDA.

Our hidden speaker identity variables live in $\mathcal{Z} = \mathbb{R}^d$. The observed data (representations of the recordings) are i-vectors or similar and live in $\mathbb{R}^D$. In general, we could allow any $D, d \geq 1$, but we get desirable properties for $D - d \gg 1$. The model described here is somewhat simpler than Kenny's heavy-tailed PLDA of [15], which has hidden speaker and channel factors, both with heavy-tailed priors.

7.1.1 Prior

The identity variable for speaker i is denoted $\mathbf{z}_i \in \mathbb{R}^d$ and they are drawn independently from $N(\mathbf{0}, \mathbf{I})$.

7.1.2 Noise model and likelihood function

The likelihood, $P(\mathbf{r}_j \mid \mathbf{z}_i)$ is parametrized by a rectangular, D-by-d *factor loading matrix*, $\mathbf{F}$, and the *degrees of freedom*, an integer, $\nu \geq 1$. For speaker i, multiple observations, $\mathbf{r}_j \in \mathbb{R}^D$, are produced by projection into the D-dimensional space and the addition of independently drawn noise, $\boldsymbol{\eta}_j$:

$$\mathbf{r}_j = \mathbf{F}\mathbf{z}_i + \boldsymbol{\eta}_j \tag{7.1}$$

The noise is heavy-tailed in the sense that it is Gaussian with variable precision, λ_j, sampled from the chi-squared distribution, χ^2_ν, parametrized by ν, the *degrees of freedom*. The noise is generated as:

$$\boldsymbol{\eta}_j \sim \mathcal{N}\bigl(\mathbf{0}, (\frac{\lambda_j}{\nu}\mathbf{W})^{-1}\bigr) \qquad\qquad \text{and} \qquad\qquad \lambda_j \sim \chi^2_\nu \tag{7.2}$$

where $\mathbf{W}$ is D-by-D positive definite. The expected value of the precision modulation factor is $\langle \frac{\lambda_j}{\nu} \rangle = 1$. Marginalizing out the hidden λ_j gives the likelihood, which is a *t-distribution*:

$$\begin{aligned} P(\mathbf{r} \mid \mathbf{z}_i) &= \mathcal{T}(\mathbf{r} \mid \mathbf{F}\mathbf{z}_i, \mathbf{W}, \nu) \\ &\propto \Bigl[1 + \frac{(\mathbf{r} - \mathbf{F}\mathbf{z}_i)'\mathbf{W}(\mathbf{r} - \mathbf{F}\mathbf{z}_i)}{\nu}\Bigr]^{-\frac{\nu+D}{2}} \end{aligned} \tag{7.3}$$

where we have ignored the somewhat complex normalization constant, which is irrelevant to our present purpose of inference for $\mathbf{z}$. We can use (7.3) as-is for our meta-embedding:

$$f_j(\mathbf{z}) \propto \Bigl[1 + \frac{(\mathbf{r}_j - \mathbf{F}\mathbf{z})'\mathbf{W}(\mathbf{r}_j - \mathbf{F}\mathbf{z})}{\nu}\Bigr]^{-\frac{\nu+D}{2}} \tag{7.4}$$

But if $D \geq d$ and $\mathbf{F}'\mathbf{W}\mathbf{F}$ is invertible, we can conveniently rearrange this into a t-distribution for $\mathbf{z}$.

7.1.3 TME: T-distribution meta-embedding

To see that $f_j(\mathbf{z})$ is a t-distribution, we define (dropping subscripts for now):

$$\mathbf{E} = \mathbf{F}'\mathbf{W}\mathbf{F}, \qquad \mathbf{m} = \mathbf{E}^{-1}\mathbf{F}'\mathbf{W}\mathbf{r} \qquad \text{and} \qquad \mathbf{G} = \mathbf{W} - \mathbf{W}\mathbf{F}\mathbf{E}^{-1}\mathbf{F}'\mathbf{W} \tag{7.5}$$

and we rearrange the quadratic form as:

$$\begin{aligned} Q_{\mathbf{r}|\mathbf{z}} &= (\mathbf{r} - \mathbf{F}\mathbf{z})'\mathbf{W}(\mathbf{r} - \mathbf{F}\mathbf{z}) \\ &= \mathbf{r}'\mathbf{W}\mathbf{r} - 2\mathbf{z}'\mathbf{F}'\mathbf{W}\mathbf{r} + \mathbf{z}'\mathbf{F}'\mathbf{W}\mathbf{F}\mathbf{z} \\ &= \mathbf{r}'\mathbf{W}\mathbf{r} - 2\mathbf{z}'\mathbf{E}\mathbf{m} + \mathbf{z}'\mathbf{E}\mathbf{z} \\ &= (\mathbf{r}'\mathbf{W}\mathbf{r} - \mathbf{m}'\mathbf{E}\mathbf{m}) + (\mathbf{z}'\mathbf{E}\mathbf{z} - 2\mathbf{z}\mathbf{E}\mathbf{m} + \mathbf{m}'\mathbf{E}\mathbf{m}) \\ &= \mathbf{r}'\mathbf{G}\mathbf{r} + (\mathbf{z} - \mathbf{m})'\mathbf{E}(\mathbf{z} - \mathbf{m}) \\ &= q + Q_{\mathbf{z}|\mathbf{r}} \end{aligned} \tag{7.6}$$

where we have defined $q = \mathbf{r}'\mathbf{G}\mathbf{r}$ and $Q_{\mathbf{z}|\mathbf{r}} = (\mathbf{z}-\mathbf{m})'\mathbf{E}(\mathbf{z}-\mathbf{m})$. Reintroducing the subscript j, and defining $\nu' = \nu+D-d$, let's reassemble our t-distribution likelihood:

$$\begin{aligned}
f_j(\mathbf{z}) &\propto \left[1+\frac{Q_{\mathbf{r}|\mathbf{z}}}{\nu}\right]^{-\frac{\nu+D}{2}} \\
&\propto \left[\frac{\nu}{\nu'}+\frac{Q_{\mathbf{r}|\mathbf{z}}}{\nu'}\right]^{-\frac{\nu'+d}{2}} \\
&= \left[\frac{\nu}{\nu'}+\frac{q_j+Q_{\mathbf{z}|\mathbf{r}}}{\nu'}\right]^{-\frac{\nu'+d}{2}} \\
&= \left[\frac{\nu+q_j}{\nu'}+\frac{Q_{\mathbf{z}|\mathbf{r}}}{\nu'}\right]^{-\frac{\nu'+d}{2}} \\
&\propto \left[1+\frac{(\frac{\nu'}{\nu+q_j})Q_{\mathbf{z}|\mathbf{r}}}{\nu'}\right]^{-\frac{\nu'+d}{2}} \\
&= \left[1+\frac{(\frac{\nu'}{\nu+q_j})(\mathbf{z}-\mathbf{m}_j)'\mathbf{E}(\mathbf{z}-\mathbf{m}_j)}{\nu'}\right]^{-\frac{\nu'+d}{2}} \\
&\propto \mathcal{T}(\mathbf{z}\mid \mathbf{m}_j, b_j\mathbf{E}, \nu')
\end{aligned} \tag{7.7}$$

where

$$\mathbf{m}_j = \mathbf{E}^{-1}\mathbf{F}'\mathbf{W}\mathbf{r}_j\,, \qquad b_j = \frac{\nu'}{\nu+q_j} \qquad \text{and} \qquad q_j = \mathbf{r}_j'\mathbf{G}\mathbf{r}_j \tag{7.8}$$

We have shown that the meta-embedding, $f_j(\mathbf{z})$, is proportional to a t-distribution, with *increased* degrees of freedom $\nu' = \nu + (D-d)$, location parameter $\mathbf{m}_j$ and precision[1] $b_j\mathbf{E}$. The mean exists only for $\nu' > 1$ and is then $\mathbf{m}_j$. The mode is $\mathbf{m}_j$ and is therefore also the maximum-likelihood estimate for $\mathbf{z}$. We shall refer to the *t-distribution meta-embedding* as TME.

In a typical i-vector PLDA speaker recognizer, we have $400 \leq D \leq 600$, while $100 \leq d \leq 200$. Since $\nu > 0$, this means $\nu' > 200$, which for all practical purposes means that $f_j(\mathbf{z})$ is Gaussian:

$$f_j(\mathbf{z}) \propto \mathcal{T}(\mathbf{z}\mid \mathbf{m}_j, b_j\mathbf{E}, \nu') \approx \mathcal{N}\bigl(\mathbf{z}\mid \mathbf{m}_j, (b_j\mathbf{E})^{-1}\bigr) \tag{7.9}$$

Specifically, the RHS is a *simple* Gaussian meta-embedding (SGME), as explained in section 5.2. That is:

$$f_j(\mathbf{z}) \approx \exp\bigl[\mathbf{z}'\mathbf{a} - \frac{1}{2}\mathbf{z}'\mathbf{B}_j\mathbf{z}\bigr] \tag{7.10}$$

where the natural parameters are:

$$\mathbf{B}_j = b_j\mathbf{E} \qquad \text{and} \qquad \mathbf{a}_j = \mathbf{B}_j\mathbf{m}_j = b_j\mathbf{F}'\mathbf{W}\mathbf{r}_j \tag{7.11}$$

[1] *Precision* for the t-distribution is not quite the same as inverse variance. The variance of a t-distribution is defined only for $\nu' > 2$ and is given in this case by $\frac{\nu+q_j}{\nu'-2}\,\mathbf{E}^{-1}$.

Ancillary statistic

To better understand b_j, let us examine $q_j = \mathbf{r}_j'\mathbf{G}\mathbf{r}$ in more detail, which is very interesting indeed. It turns out that q_j is an *ancillary statistic*—it is *independent* of $\mathbf{z}$, because $\mathbf{G}\mathbf{F} = \mathbf{0}$ and so:

$$\mathbf{G}\mathbf{r} = \mathbf{G}(\mathbf{F}\mathbf{z} + \boldsymbol{\eta}) = \mathbf{G}\boldsymbol{\eta} \tag{7.12}$$

Even though q_j is independent of $\mathbf{z}$, which we are trying to infer, it is nevertheless relevant to that inference, because it modulates the uncertainty. Notice that this statistic only works for $D > d$. If we have $d = D$, and $\mathbf{F}$ is invertible, then $\mathbf{G} = \mathbf{0}$ and $q_j = 0$.

To better understand what $\mathbf{r}'\mathbf{G}\mathbf{r}$ does, consider applying an invertible linear transform to the data, such that the model for the transformed data then has $\mathbf{W} = \mathbf{I}$. Since this transform does not change anything essential, the role played by $\mathbf{G}$ is much the same as in the general case. For $\mathbf{W} = \mathbf{I}$ we have:

$$\mathbf{G} = \mathbf{I} - \mathbf{F}(\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}$$

which is an *idempotent projection matrix*, for which:

$$q_j = \mathbf{r}_j'\mathbf{G}\mathbf{r}_j = (\mathbf{F}\mathbf{z} + \boldsymbol{\eta}_j)'\mathbf{G}(\mathbf{F}\mathbf{z} + \boldsymbol{\eta}_j) = \boldsymbol{\eta}_j'\mathbf{G}\boldsymbol{\eta}_j = (\mathbf{G}\boldsymbol{\eta}_j)'(\mathbf{G}\boldsymbol{\eta}_j)$$

That is, $\mathbf{G}\mathbf{r}_j$ is the projection of $\mathbf{r}_j$ onto the complement of the subspace spanned by the columns of $\mathbf{F}$. Since $\mathbf{G}\mathbf{r}_j$ is $(D-d)$-dimensional, $(\mathbf{G}\mathbf{r}_j)'(\mathbf{G}\mathbf{r}_j)$ is effectively composed of the sum of $D - d$ squares. Intuitively,

$$b_j = \frac{\nu'}{\nu + q_j} = \frac{\nu + (D-d)}{\nu + q_j}$$

forms a regularized estimate of the hidden noise precision modulation factor $\frac{\lambda_j}{\nu}$. The scale of q_j is proportional to $D - d$ and will (as indeed intuition suggests) have a greater effect for larger dimensionality differences. The scale of q_j is however independent of ν, so that for Gaussian noise, where $\nu \to \infty$ and $\frac{\nu}{\lambda_j} \to 1$, we also have $\frac{\nu'}{\nu + q_j} \to 1$.

7.1.4 Generalizations

Our model can be generalized in the following ways:

- Our derivation of the t-distribution, with precision scaling factor drawn from χ^2_ν is from Wikipedia.[2] Noting that $\chi^2_\nu(\lambda) = \mathcal{G}(\lambda \mid \frac{\nu}{2}, \frac{1}{2})$, where $\mathcal{G}$ is

[2] http://en.wikipedia.org/wiki/Multivariate_t-distribution.

the gamma distribution, we can indeed generalize the model slightly to non-integer degrees of freedom, by drawing λ from a Gamma distribution—see Bishop's equation (2.161) in chapter 2 of [38].

- Consider making the *whole noise precision matrix* hidden, rather than just a scalar precision modulation factor. That would require a Wishart prior, rather a Gamma prior. Surprisingly, this does *not* lead to a generalization. When this matrix is marginalized out, we get exactly the *same* t-distribution likelihood. Compare for example Bishop's equations (2.161) and (B.68).

7.1.5 LR computation

LRs between TMEs are not available in closed form. However, as we found above, when $(D-d) \gg 1$, the TME is very close to an SGME, even though the noise may be very heavy tailed. Let us proceed with the SGME approximation, with natural parameters, $\mathbf{a}_j, b_j\mathbf{E}$ given by (7.11). We already know that SGME expectations can be done more efficiently than for the general GME case, as explained in section 5.2. Given the precomputed eigendecomposition, $\mathbf{E} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'$, the SGME log-expectation (5.17) is:

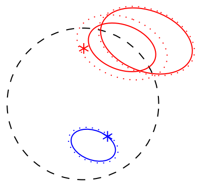
$$\begin{aligned}\log E_G(\mathbf{a}, b) &= \mathbf{a}'(\mathbf{I} + b\mathbf{E})^{-1}\mathbf{a} - \log|\mathbf{I} + b\mathbf{E}| \\ &= \operatorname{tr}\Bigl[(\tilde{\mathbf{a}}\tilde{\mathbf{a}}')(\mathbf{I} + b\boldsymbol{\Lambda})^{-1}\Bigr] - \log|\mathbf{I} + b\boldsymbol{\Lambda}|\end{aligned} \tag{7.13}$$

where $\tilde{\mathbf{a}} = \mathbf{V}'\mathbf{a}$.

Further speed-up can be achieved for example by the Taylor series approximation of section 6.2.3.

7.1.6 Example

Below is an example of three SGME approximations extracted from three data points (recordings) randomly generated from a heavy-tailed PLDA model, with $d = 2$, $D = 20$ and $\nu = 2$. There is one recording from a blue speaker and two recordings from a red one:



As before, the dashed black circle represents $\pi(\mathbf{z})$. The true values of the identity variables are indicated with the blue and red asterisks. The solid ellipses reflect the precisions as calculated by $b_j\mathbf{E}$, while the dotted ones reflect the true (oracle) values of the precision modulation factors.

7.1.7 Multiclass classification

For multiclass classification, with large amounts of training data for each class, we can apply the scoring approximation of section 6.5 to this model. We assume that the normalized distribution (product of t-distributions) that represents the training data for every class i, concentrates to a Dirac delta at $\hat{\mathbf{z}}_i$. As explained above, for runtime scoring, we can simply plug these values into the meta-embeddings that represent the test recordings.

Training

Let class i have a set of training data, $\mathcal{R}_i = \{\mathbf{r}_{ij}\}_{j=1}^{N_i}$, where each $\mathbf{r}_{ij}$ is represented by (7.9) as the approximate Gaussian meta-embedding:

$$f_{ij}(\mathbf{z}) = \mathcal{N}\bigl(\mathbf{z} \mid \mathbf{m}_{ij}, (b_{ij}\mathbf{E})^{-1}\bigr) \tag{7.14}$$

where by (7.8):

$$\mathbf{m}_{ij} = \mathbf{E}^{-1}\mathbf{F}'\mathbf{W}\mathbf{r}_{ij}\,, \qquad b_{ij} = \frac{\nu + D - d}{\nu + q_{ij}} \qquad \text{and} \qquad q_{ij} = \mathbf{r}'_{ij}\mathbf{G}\mathbf{r}_{ij} \tag{7.15}$$

This gives:

$$\pi(\mathbf{z})f_i(\mathbf{z}) = \pi(\mathbf{z})\prod_j f_{ij}(\mathbf{z}) \propto P(\mathbf{z} \mid \mathcal{R}_i, \pi) = \mathcal{N}(\mathbf{z} \mid \hat{\mathbf{z}}_i, \bar{\mathbf{E}}_i^{-1}) \tag{7.16}$$

where

$$\bar{\mathbf{E}}_i = \mathbf{I} + \sum_j b_{ij}\mathbf{E} \tag{7.17}$$

and

$$\hat{\mathbf{z}}_i = \bar{\mathbf{E}}_i^{-1}\mathbf{E}\sum_j b_{ij}\mathbf{m}_{ij} = \bar{\mathbf{E}}_i^{-1}\mathbf{E}\sum_j b_j\mathbf{m}_{ij} = \bar{\mathbf{E}}^{-1}\mathbf{E}\mathbf{F}'\mathbf{W}\sum_j b_j\mathbf{r}_{ij} \tag{7.18}$$

Notice that the class representative, $\hat{\mathbf{z}}_i$, is formed via a weighted combination of the data for that class. The weights are inversely proportional to the estimated noise variances. If we have enough data, so that $\bar{\mathbf{E}}_i \gg \mathbf{I}$, then we can approximate:

$$\pi(\mathbf{z})\overline{f_i} \approx \delta(\mathbf{z} - \hat{\mathbf{z}}_i)$$

Scoring

At runtime, given a test recording, r_j, represented by the approximate Gaussian meta-embedding:

$$g_j(\mathbf{z}) = \exp[\mathbf{a}_j'\mathbf{z} - \frac{b_j}{2}\mathbf{z}'\mathbf{E}\mathbf{z}] \tag{7.19}$$

we can use (6.25) to approximate the log-likelihood for each class i as:[3]

$$\begin{aligned}\log L_{ij} &\approx \log g_j(\hat{\mathbf{z}}_i) - \log\langle g_j\rangle \\ &= \mathbf{a}_j'\hat{\mathbf{z}}_i - \frac{b_j}{2}\hat{\mathbf{z}}_i'\mathbf{E}\hat{\mathbf{z}}_i - \frac{1}{2}\mathbf{a}_j(\mathbf{I}+b_j\mathbf{E})^{-1}\mathbf{a}_j + \frac{1}{2}\log|\mathbf{I}+b_j\mathbf{E}|\end{aligned} \tag{7.20}$$

This can be used for open-set case, while for the *closed-set* case, we can omit all terms independent of i, to give:

$$\log L_{ij} \approx \mathbf{a}_j'\hat{\mathbf{z}}_i - \frac{b_j}{2}\hat{\mathbf{z}}_i'\mathbf{E}\hat{\mathbf{z}}_i + \text{const} \tag{7.21}$$

This can be arranged into a $(d+1)$-dimensional dot-product for computational convenience.

Intuitively it would be satisfying for the the score to be modulated by b_j, so that score magnitudes become smaller for noisier data. This is true for the second term in (7.21), but what about the first term? We can make the the dependence on b_j clear by considering $\boldsymbol{\mu}_j = (\mathbf{I}+b_j\mathbf{E})^{-1}\mathbf{a}_j$, which is the mean of the Gaussian approximation to $P(\mathbf{z} \mid \mathbf{r}_j, \pi)$. Recall that similarly, $\hat{\mathbf{z}}_i$ is the mean of the Gaussian approximation to $P(\mathbf{z} \mid \mathcal{R}_i, \pi)$. The first term can now be rewritten as:

$$\mathbf{a}_j'\hat{\mathbf{z}}_i = \boldsymbol{\mu}_i'(\mathbf{I}+b_j\mathbf{E})\hat{\mathbf{z}}_i$$

where the role of b_j becomes clear.

Doing it without Gaussians

As an alternative (perhaps a good idea when $D-d$ is small), both training and scoring can be done without Gaussian approximations. For training, the class representative, $\hat{\mathbf{z}}_i$, can be found by numerical maximization of the product of t-distributions. If in doubt about the sharpness of the peaks at the maxima, we could verify this by using the Laplace approximation as explained in section 6.4. If all peaks are indeed sharp, we can score closed-set multiclass likelihoods by plugging the $\hat{\mathbf{z}}_i$ into the t-distribution meta-embeddings of the test segments, as in (6.25).

[3] As a sanity check, we can derive the same result, without makes use of the Dirac delta, by directly taking the limit of (5.7) when taking the precision of one of the Gaussians to infinity.

7.2 Mixture of PLDAs

Our PLDA model with heavy-tailed noise makes use of a continuous mixture of Gaussians. The resultant meta-embeddings are t-distributions, which we then resorted to approximate by discrete mixtures of Gaussians. In this section we investigate instead what happens if we start out with a discrete mixture of PLDA models instead.

In this generative model, the prior remains standard normal, but the likelihood is now a mixture of the form:

$$f_j(\mathbf{z}) \propto P(\mathbf{r}_j \mid \mathbf{z}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{r}_j \mid \mathbf{F}_k\mathbf{z}, \mathbf{W}_k^{-1}) \tag{7.22}$$

We shall express the likelihood as a mixture of normalized GMEs of the form:

$$\overline{g_{jk}}(\mathbf{z}) = \frac{|\mathbf{I}+\mathbf{B}_k|^{\frac{1}{2}}}{\exp[\frac{1}{2}\mathbf{a}'_{jk}(\mathbf{I}+\mathbf{B}_k)\mathbf{a}_{jk}]} \exp[\mathbf{z}'\mathbf{a}_{jk} - \frac{1}{2}\mathbf{z}'\mathbf{B}_k\mathbf{z}] \tag{7.23}$$

with parameters of the form:

$$\mathbf{B}_k = \mathbf{F}'_k\mathbf{W}_k\mathbf{F}_k, \qquad \text{and} \qquad \mathbf{a}_{jk} = \mathbf{F}'_k\mathbf{W}_k\mathbf{r}_j \tag{7.24}$$

The likelihood can be rewritten without assuming invertibility of the $\mathbf{B}_k$ as:

$$\begin{aligned}
f_j(\mathbf{z}) &\propto \sum_{k=1}^{K} w_k \sqrt{|\mathbf{W}_k|} \exp\bigl[-\frac{1}{2}(\mathbf{r}_j - \mathbf{F}_k\mathbf{z})'\mathbf{W}_k(\mathbf{r}_j - \mathbf{F}_k\mathbf{z})\bigr] \\
&= \sum_{k=1}^{K} w_k \sqrt{|\mathbf{W}_k|} \exp\bigl[-\frac{1}{2}\mathbf{r}'_j\mathbf{W}_k\mathbf{r}_j - \frac{1}{2}\mathbf{z}'\mathbf{B}_k\mathbf{z} + \mathbf{z}'\mathbf{a}_{jk})\bigr] \\
&= \sum_{k=1}^{K} w_k \sqrt{\frac{|\mathbf{W}_k|}{|\mathbf{I}+\mathbf{B}_k|}} \exp\bigl[\frac{1}{2}\mathbf{a}'_{jk}(\mathbf{I}+\mathbf{B}_k)^{-1}\mathbf{a}_{jk} - \frac{1}{2}\mathbf{r}'_j\mathbf{W}_k\mathbf{r}_j\bigr]\, \overline{g_{jk}}(\mathbf{z}) \\
&= \sum_{k=1}^{K} w_k \sqrt{\frac{|\mathbf{W}_k|}{|\mathbf{I}+\mathbf{B}_k|}} \exp\bigl[\frac{1}{2}\mathbf{r}'_j(\mathbf{W}_k\mathbf{F}_k(\mathbf{I}+\mathbf{B}_k)^{-1}\mathbf{F}'_k\mathbf{W}_k - \mathbf{W}_k)\mathbf{r}_j\bigr]\, \overline{g_{jk}}(\mathbf{z}) \\
&= \sum_{k=1}^{K} w_k \sqrt{\frac{|\mathbf{W}_k|}{|\mathbf{I}+\mathbf{B}_k|}} \exp\bigl[-\frac{1}{2}\mathbf{r}'_j\tilde{\mathbf{G}}_k\mathbf{r}_j\bigr]\, \overline{g_{jk}}(\mathbf{z})
\end{aligned} \tag{7.25}$$

where $\tilde{\mathbf{G}}_k$ is defined below and simplified with the Woodbury identity:[4]

$$\tilde{\mathbf{G}}_k = \mathbf{W}_k - \mathbf{W}_k\mathbf{F}_k(\mathbf{I}+\mathbf{B}_k)^{-1}\mathbf{F}'_k\mathbf{W}_k = (\mathbf{W}^{-1} + \mathbf{F}\mathbf{F}')^{-1} \tag{7.26}$$

[4] http://en.wikipedia.org/wiki/Woodbury_matrix_identity

The RHS shows that $\tilde{\mathbf{G}}_k^{-1}$ is the variance of $\mathbf{r}$, when sampled from:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad\qquad \text{and} \qquad\qquad \mathbf{r} \sim \mathcal{N}(\mathbf{F}_k\mathbf{z}, \mathbf{W}_k^{-1})$$

The matrix determinant lemma[5] further reveals that:

$$|\tilde{\mathbf{G}}_k| = \frac{|\mathbf{W}_k|}{|\mathbf{I} + \mathbf{B}_k|} \tag{7.27}$$

so that we can finally simplify the meta-embedding as:

$$\begin{aligned} f_j(\mathbf{z}) &\propto \sum_{k=1}^{K} w_k \, \mathcal{N}(\mathbf{r}_j \mid \mathbf{0}, \tilde{\mathbf{G}}_k^{-1}) \, \overline{g_{jk}}(\mathbf{z}) \\ &\propto \sum_{k=1}^{K} P(k \mid \mathbf{r}_j) \, \overline{g_{jk}}(\mathbf{z}) \end{aligned} \tag{7.28}$$

[5] http://en.wikipedia.org/wiki/Matrix_determinant_lemma

Chapter 8

Objective functions for discriminative training

In this chapter, we explore ways to discriminatively train meta-embedding extractors. To design a discriminative recipe, we need to make choices for the following main ingredients:

- The meta-embedding representation
- The training objective function
- The optimizer

The choices for representation and objective function are mostly orthogonal. As long as we can backpropagate derivatives through the calculations required by a given representation, we can in principle match such a representation to any of the objective functions to be discussed below. Since representations have already been treated in chapter 5, we concentrate in this chapter on objective functions.

The optimizer could be a full batch optimizer such as L-BFGS [40], or some variant of stochastic gradient descent, with mini-batches. The choice of optimizer is also to some extent independent of the other choices, although large optimization problems will tend to favour stochastic mini-batch methods. Moreover, as mentioned in chapter 6, approximate stochastic expectations can play well with stochastic optimization.

8.1 Proper scoring rules

In the introduction we have already mentioned that, since meta-embeddings quantify uncertainty, we need objective functions that properly evaluate the goodness of the uncertainty. Since our uncertainty is expressed in a probabilistic framework, its goodness can be evaluated via *proper scoring rules* [41]. If a proper scoring rule is used as training criterion, this is known as *minimum scoring rule inference* [42]. Well known-training criteria, such as maximum likelihood, maximum conditional likelihood and minimum cross-entropy are all special cases of minimum scoring rule inference.

There are two equivalent ways to define proper scoring rules. The first (more well-known) works like this. Let P and Q be probability distributions to predict an event $e \in \mathcal{E}$. A proper scoring rule is a real-valued function $S(Q, e)$, with the property:

$$\big\langle S(P, e) \big\rangle_{e \sim P} \leq \big\langle S(Q, e) \big\rangle_{e \sim P} \tag{8.1}$$

In other words, the expected value w.r.t. P is minimized at $Q = P$. The rule is termed *strictly* proper if the minimum is unique. The canonical proper scoring rule is the *logarithmic scoring rule*, defined as:

$$S(Q, e) = -\log Q(e) \tag{8.2}$$

With the logarithmic rule, training based on scoring rule minimization is just (depending on the context) maximum likelihood, maximum conditional likelihood, or minimum cross-entropy. Here we are interested in discriminative training and therefore in maximum conditional likelihood (MCL), which we will discuss in the next section. Unfortunately, for our training problem, MCL is intractable, so we need to find alternatives, which we will construct using scoring rules other than the logarithmic one.

The other definition of proper scoring rules is given by decision theory [43]. Let Q be a probability distribution to predict an event $e \in \mathcal{E}$. Let $d \in \mathcal{D}$, be a *decision* that is made when Q is given, while e is still unknown. Let $C(d, e)$ be a real-valued function, known as a *cost function*, that quantifies the goodness of decision d when afterwards it becomes known that the value of the event is e. Let $d_Q^* \in \mathcal{D}$ be a minimum-expected-cost *Bayes decision*, with the property:

$$\big\langle C(d_Q^*, e) \big\rangle_{e \sim Q} \leq \big\langle C(d, e) \big\rangle_{e \sim Q} \quad \forall d \in \mathcal{D} \tag{8.3}$$

Then

$$S(Q, e) = C(d_Q^*, e) \tag{8.4}$$

is a proper scoring rule. It is easy to check that (8.4) implies (8.1), but the converse is also true [44]. If we let $\mathcal{D} = \mathcal{E}$, then the decision, d, can be interpreted as a prediction for e. But the decision space, $\mathcal{D}$, and the cost function, $C(d,e)$ can be much more general. This gives much flexibility in designing scoring rules—see for example [45, 44, 46]. As we shall see below, maximum conditional likelihood is intractable for our problem, but we shall make use of the flexibility of proper scoring rules to find other tractable solutions, similar to the ones in [47, 48].

8.1.1 Why are proper scoring rules good for training?

8.2 The CRP partition prior

To do discriminative training with proper scoring rules as objective functions, those proper scoring rules need to be applied to posterior probability distributions over all possible ways to partition a set of n recordings. Since our meta-embeddings provide likelihoods rather than posteriors, we additionally need some *prior* distribution over partitions in order to form posteriors.

For our purposes, a convenient prior over partitions can be constructed using the two-parameter (Pitman-Yor) *Chinese restaurant process* (CRP) [49, 50]. The CRP gives a recursive construction for a probability distribution over partitions of a set of recordings, as follows.

Let $\mathcal{L} \in \mathcal{P}_n$, where $\mathcal{L}: \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ be a k-speaker partition of the set of n recordings, expressed as before in terms of index subsets, $\mathcal{S}_j \subseteq \{1, \dots, n\}$. The CRP probability distribution can be expressed as:

$$P(\mathcal{L}) = P(\ell_i \mid \mathcal{L}_{\setminus i}) P(\mathcal{L}_{\setminus i}), \qquad \forall i \in \{1, \dots, n\} \tag{8.5}$$

which is invariant to the choice of i. The conditional factor is given by:

$$P(\ell_i = j \mid \mathcal{L}_{\setminus i}) = \begin{cases} \frac{|\mathcal{S}'_j| - \beta}{n-1+\alpha}, & \text{for } 1 \leq j \leq k' \\ \frac{k'\beta + \alpha}{n-1+\alpha}, & \text{for } j = k'+1 \end{cases} \tag{8.6}$$

where $|\cdot|$ denotes set size and where $\mathcal{L}_{\setminus i}: \mathcal{S}'_1, \mathcal{S}'_2, \dots, \mathcal{S}'_{k'}$ is the partition that results by removing recording index i from $\mathcal{L}$. That is, $\mathcal{L}_{\setminus i} \in \mathcal{P}_{n-1}$ is the partition, obtained from $\mathcal{L} \in \mathcal{P}_n$ by removing i from the subset where it occurs. If the modified subset remains non-empty, then $k' = k$, else that subset is removed, so that $k' = k - 1$. In the trivial case when $n = 1$ and $\mathcal{L}: \mathcal{S}_1 = \{1\}$, we allow $\mathcal{L}_{\setminus i}$ to be empty, with $k' = 0$.

We refer to the categorical variable, $\ell_i \in \{1, 2, \dots, k', k'+1\}$ as the *speaker label*. Its value identifies the subset to which recording i is hypothesized to belong. If $\ell_i = k' + 1$, then a new subset (speaker) that is not in $\mathcal{L}_{\setminus i}$ is implied.

The definition (8.5) is recursive, so that $P(\mathcal{L}_{\setminus i})$ is similarly defined, where the recursion ends when $\mathcal{L}_{\setminus i}$ is empty. The CRP is parametrized by the parameters α (concentration) and β (discount), where $\alpha \geq 0$ and $0 \leq \beta < 1$.[1]

The CRP distribution is *exchangeable*, meaning we can arbitrarily shuffle the labels without affecting the value of $P(\mathcal{L})$. To compute the probability for a given $\mathcal{L}$, the recursion can be done for example from large to small, or from small to large. The small-to-large direction can also be used as a generative method to sample from this prior. In that case, we can rewrite the distribution in *autoregressive* form as follows:

$$P(\mathcal{L}) = \prod_{i=1}^{n} P(\ell_i \mid \mathcal{L}_{<i}) \tag{8.7}$$

where $\mathcal{L}_{<i} \in \mathcal{P}_{i-1}$ is obtained from $\mathcal{L} \in \mathcal{P}_n$ by retaining all indices less than i and discarding the rest. (For $i = 1$, we have $\ell_1 = 1$.) The conditional factors, $P(\ell_i \mid \mathcal{L}_{<i})$ are given by using $\mathcal{L}_{<i} = \mathcal{L}_{\setminus i}$ and $n = i$ in (8.6). This form illustrates the Chinese restaurant analogy, where customers (recordings) arrive one by one and are randomly seated at either occupied or new tables (speakers).

8.2.1 Expected number of speakers

For a CRP parametrized with α, β, the expected number of speakers in n recordings is:

$$\langle k \rangle = \begin{cases} 1 & \text{for } \alpha = \beta = 0, \\ \alpha(\psi(n+\alpha) - \psi(\alpha)) & \text{for } \alpha > 0, \beta = 0, \\ \frac{\Gamma(\alpha+\beta+n)\Gamma(\alpha+1)}{\beta\Gamma(\alpha+n)\Gamma(\alpha+\beta)} - \frac{\alpha}{\beta} & \text{for } \alpha \geq 0, \beta > 0 \end{cases} \tag{8.8}$$

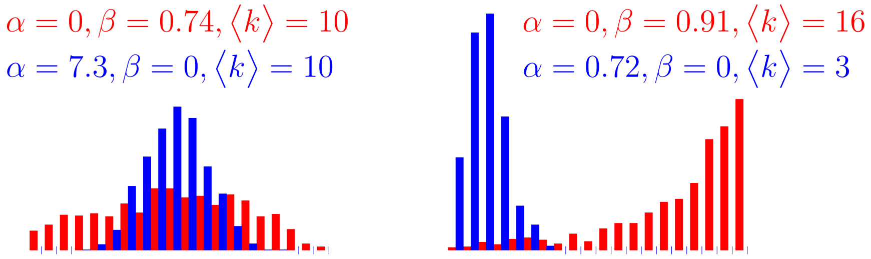
where Γ and ψ denote the gamma and digamma functions. Notice that $1 \leq \langle k \rangle < n$, where the upper limit is reached at $\alpha \to \infty$, or $\beta \to 1$.

8.2.2 Example

We let the number of recordings be $n = 20$. For each of four different choices of the CRP parameters, α, β, we generate 1000 random partitions using (8.7). The histograms below are for the distribution of k (number of speakers) in

[1] The model can be extended to allow some negative values for the parameters. See http://en.wikipedia.org/wiki/Chinese_restaurant_process, or [51].

these random partitions. The four different CRP parameter choices and their histograms are shown below for each of $\{\text{left}, \text{right}\} \times \{\text{red}, \text{blue}\}$:[2]



As can be seen, the distribution for k can be controlled by appropriate parameter choices. We can control the expected number of speakers, $\langle k \rangle$ using (8.8) and iterative root-finding.[3] We can also control the width of the distribution—in particular, setting $\alpha = 0$ gives wide distributions.

8.3 The intractable partition posterior

Let $\mathcal{L}_0 \in \mathcal{P}_n$ represent some convenient, fixed partition hypothesis—for example the finest, or the coarsest partition.[4] The partition posterior can be expressed in terms of prior and likelihoods, or likelihood-ratios, as follows:

$$\begin{aligned} P(\mathcal{L} \mid \mathcal{R}) &= \frac{P(\mathcal{L})P(\mathcal{R} \mid \mathcal{L})}{\sum_{\mathcal{L}'} P(\mathcal{L}')P(\mathcal{R} \mid \mathcal{L}')} \\ &= \frac{P(\mathcal{L})\frac{P(\mathcal{R}|\mathcal{L})}{P(\mathcal{R}|\mathcal{L}_0)}}{\sum_{\mathcal{L}'} P(\mathcal{L}')\frac{P(\mathcal{R}|\mathcal{L}')}{P(\mathcal{R}|\mathcal{L}_0)}} \end{aligned} \tag{8.9}$$

where the summation is over all possible partitions. If we are given a meta-embedding for every recording, and if arbitrary pooling and expectations (or inner products) of meta-embeddings are tractable, then we also have that the likelihood-ratio, $\frac{P(\mathcal{R}|\mathcal{L})}{P(\mathcal{R}|\mathcal{L}_0)}$, is tractable via the methods of section 4.2. The problem is that the number of terms in the denominator (the Bell number) is hopelessly intractable, except for very small n. A speaker recognizer is typically trained with n in the thousands, but for $n = 75$, we already have

[2] Each histogram has 20 bins, for the 20 possible values that k can have. For visibility, the blue bins are staggered towards the right relative to the red bins.

[3] See our code example, `create_PYCRP.m` at http://github.com/bsxfan/meta-embeddings.

[4] The finest partition is the the hypothesis that $\mathcal{R} = \{r_1, \dots, r_n\}$ was spoken n different speakers, while the coarsest partition hypothesizes a single speaker.

that the Bell number exceeds 10^{80}, the estimated number of atoms in the observable universe!

Although the posterior is intractable to evaluate, we *can* sample from it, for example with Gibbs sampling, which we discuss below in section 8.5.

For small n, see for example [52], for an algorithm to iterate through all the partitions.

8.4 The intractable MCL objective

In supervised training of generative models, *maximum likelihood* (ML) is the natural training criterion:

$$\underset{\theta}{\operatorname{argmax}}\, P(\text{data} \mid \text{labels}, \theta)$$

where θ are the model parameters. For discriminative training, the natural criterion is *maximum conditional likelihood*[5] (MCL):

$$\underset{\theta}{\operatorname{argmax}}\, P(\text{labels} \mid \text{data}, \theta)$$

In the case of supervised training of a (multiclass) classifier, with independent training examples, MCL is just the well-known (multiclass) cross-entropy objective:

$$\underset{\theta}{\operatorname{argmax}}\, P(\text{labels} \mid \text{data}, \theta) = \underset{\theta}{\operatorname{argmin}} \sum_i -\log P(\text{label}_i \mid \text{datum}_i, \theta)$$

In our case, MCL is equivalent to applying the logarithmic proper scoring rule to the partition posterior:

$$\underset{\theta}{\operatorname{argmax}}\, P(\mathcal{L}^* \mid \mathcal{R}, \theta) = \underset{\theta}{\operatorname{argmin}} -\log P(\mathcal{L}^* \mid \mathcal{R}, \theta) \tag{8.10}$$

where we now make the to-be-trained parameters, θ explicit; where $\mathcal{R}$ is the set of recordings in the training data; and where $\mathcal{L}^* \in \mathcal{P}_n$ is a single 'label', namely the true partitioning of $\mathcal{R}$ w.r.t. speaker. As discussed above, the normalization constant of this posterior is intractable. Since the normalization constant is a function of θ, MCL cannot be applied directly and we shall need to consider alternatives. Such alternatives include solutions based on sampling from the posterior (e.g. contrastive divergence) and also alternative scoring rules, which can be evaluated without needing the normalization constant (e.g. pseudolikelihood and composite likelihood).

[5] Why is $P(\text{labels} \mid \text{data})$ termed *conditional*, while $P(\text{data} \mid \text{labels})$ is not? In the former case, $P(\text{data})$ is not used: everything is conditional on the data, which is given. In the latter case, $P(\text{labels})$ could be used, but often $P(\text{labels})$ is so trivial that it is ignored.

8.4.1 Is one label enough?

The MCL objective of (8.10) is very different from the familiar cross-entropy objective: cross-entropy is usually applied to a training database with very many independent, labelled examples, while (8.10) has but a single 'label'. Is this one label enough?

One answer is that this single label does contain many bits of information. We use an approximate expression for the log Bell number,[6] to show that the number of bits needed to uniquely specify a partition of n recordings is roughly:

n	10	100	1000	10000	100000
bits	17	380	6400	92000	1200000

Also notice that the number of bits is a superlinear function of n. This shows that if we were to chop a training database into subsets, the sum of the subset labels would contain less information than the original label.

We show in the example below, via a brute-force calculation for a small data set, with $n = 8$ and a single label worth about 12 bits, that the MCL objective already *works*.

8.4.2 Example: MCL calibration of SGMEs

Figure 8.1 shows an experiment on synthetic data that demonstrates the calibration-sensitive nature of the MCL objective.

We randomly sampled a data set, $\mathcal{R}$, consisting of eight i-vector-like recording representations, of dimensionality $D = 20$, from the heavy-tailed PLDA model of section 7.1. We used $\mathbf{z} \in \mathbb{R}^2$ to make plotting possible. The model was parametrized with $\nu = 3$, $\mathbf{W} = \mathbf{I}$ and $\mathbf{F}$ randomly generated using a scale that gives reasonable separation between speakers.

We extracted the eight simple Gaussian meta-embeddings (SGMEs), which as explained in section 7.1 are a good approximation (for $D - d \gg 1$) to the true meta-embeddings for this model. We then applied a deliberate miscalibration to these meta-embeddings, by scaling the natural parameters, with scale varying logarithmically through about 4 orders of magnitude. Scale factors larger than 1 cause overoptimistic precisions (they would show smaller ellipses if plotted), while those smaller than 1 do the opposite.

The MCL objective, $-P(\mathcal{L}^* \mid \mathcal{R}, \text{scale})$, where $\mathcal{L}^*$ is the true partition w.r.t speaker, was evaluated for every scale factor and plotted. As the plot shows, the optimal MCL value is close to unity (no miscalibration)

[6] See: http://en.wikipedia.org/wiki/Bell_number.

as it should be. Recall that MCL requires a prior, for which we used CRP parametrized to have an expected number of speakers equal to 5.

For tractability, we chose the size of $\mathcal{R}$ at just $n = 8$, for which there are already 4140 partition possibilities. Our MCL implementation requires (for every value of the scale factor) the following:

- Pooling different combinations of meta-embeddings for each of the possible $2^n - 1$ non-empty subsets. Since the SGME natural parameters are additive, this is implemented with (sparse) matrix multiplication.
- Computing log-expectations (5.17) for every subset, $2^n - 1$ of them.
- Adding log-expectations (again with sparse matrix multiplication) to compute the log-likelihood for every possible partition, n-th Bell number of them.

By doing some precomputation of some large, sparse index matrices,[7] we got every MCL computation (per scale factor) to be reasonably fast. Even for $n = 10$, it takes about 10ms per MCL evaluation on a high-end laptop. However, because of the explosive Bell number, we can't go much higher than $n = 10$, so for real datasets, as already mentioned, we cannot rely on brute-force MCL. We can however allow ourselves to assemble other recipes that apply MCL to subsets of data, as long as the subset sizes are no larger than say $n = 10$.

8.5 Gibbs sampling

As we shall see below in section 8.6, the intractable MCL objective can be replaced by recipes based on Monte-Carlo sampling from the intractable partition posterior. In this section we present some recipes based on Gibbs sampling.

If the prior, $P(\mathcal{L})$, is the CRP of section 8.2, then a Gibbs sampler for the posterior $P(\mathcal{L} \mid \mathcal{R})$ can be constructed as follows. Let n be the number of recordings. Given the previously sampled partition, $\mathcal{L}$, the next sample is obtained as follows:

1. Select $i \in \{1, \dots, n\}$, randomly, or in round-robin fashion.
2. Obtain $\mathcal{L}_{\setminus i}$ by removing index i from $\mathcal{L}$, as explained in section 8.2.
3. Sample a new speaker label, ℓ_i, from $P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R})$.

[7] Code is available at `http://github.com/bsxfan/meta-embeddings`.

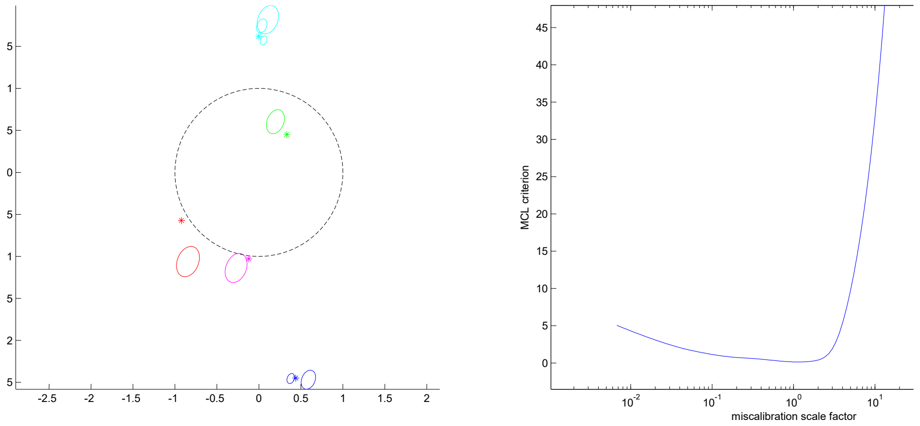


Figure 8.1: MCL calibration of SGMEs. Left: 8 SGMEs for 5 speakers, using the same representation as in section 5.1.5. Dashed black shows $\pi(\mathbf{z})$. Asterisks show the true values of the speaker identity variables. Right: The MCL objective, $-\log P(\mathcal{L}^* \mid \mathcal{R}, \text{scale})$, as a function of a deliberate miscalibration scale factor. The optimal value is close to unity as it should be.

4. Reassemble $\mathcal{L}$ from ℓ_i and $\mathcal{L}_{\setminus i}$.

In contrast to the intractable full posterior, $P(\mathcal{L} \mid \mathcal{R})$, the posterior factor $P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R})$ is tractable, because there are at most $k'+1$ possibilities for ℓ_i, where k' is defined in section 8.2. Notice that $k'+1 \leq n$ and indeed, for a typical training database $k' \ll n$. We have:

$$\begin{aligned} P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}) &\propto P(\ell_i \mid \mathcal{L}_{\setminus i}) P(\mathcal{R} \mid \ell_i, \mathcal{L}_{\setminus i}) \\ &\propto P(\ell_i \mid \mathcal{L}_{\setminus i}) \frac{P(\mathcal{R} \mid \ell_i, \mathcal{L}_{\setminus i})}{P(\mathcal{R} \mid \bar{\mathcal{L}}_i)} \end{aligned} \tag{8.11}$$

where as above, $\bar{\mathcal{L}}_i$ is a conveniently chosen constant partition. To be clear, $\bar{\mathcal{L}}_i$ must be independent of the value of ℓ_i, but may be dependent on the index i. The first RHS factor (prior) is given by (8.6), while the second RHS factor (likelihood-ratio) can be computed with the methods of section 4.1. The most convenient choice of $\bar{\mathcal{L}}_i$ could vary depending on the context. We highlight $\bar{\mathcal{L}}_i = (\ell_i = k'+1, \mathcal{L}_{\setminus i})$, which casts (8.11) as an open-set classifier, for which the LR factor is given by (4.3).

Gibbs sampling and the Hasse diagram

To better understand this Gibbs sampling procedure and possible generalizations, consider again the Hasse diagram in figure 4.1. The Gibbs sampler moves between nodes of the lattice, but the moves it can do are not the same as the arcs in the Hasse diagram. For example, the Gibbs sampler can jump from 124|3 to 12|24, a move for which there is no arc in the diagram. Conversely, in the Hasse diagram, 14|23 and 1234 are connected, but the Gibbs sampler cannot do this in one step, because more than one speaker label has to change for this move.

Other ways to construct a Gibbs sampler are possible, where moves might be allowed that change multiple speaker labels in a single step. As long as the number of possible moves that can be done in a single step does not become too large, such Gibbs samplers are feasible. The set of possible moves must also be such that starting from any node, every other node can eventually be reached, so that sampling Markov chain is ergodic.

8.5.1 Generalized Gibbs sampler

Let us construct a Gibbs sampler that samples multiple labels, (m of them, where $m \leq n$) in the same step. Given the previously sampled partition $\mathcal{L}$, the next sample is obtained as follows:

1. Select an index subset, $\mathcal{I} \subseteq \{1, \ldots, n\}$, randomly, or in round-robin fashion, such that $\mathcal{I}$ has m elements.
2. Obtain $\mathcal{L}_{\setminus\mathcal{I}}$ by removing all elements of $\mathcal{I}$ from $\mathcal{L}$.
3. Sample a set of m new speaker labels, denoted $\mathcal{L}_{\mathcal{I}}$, from $P(\mathcal{L}_{\mathcal{I}} \mid \mathcal{L}_{\setminus\mathcal{I}}, \mathcal{R})$.
4. Reassemble $\mathcal{L}$ from $\mathcal{L}_{\mathcal{I}}$ and $\mathcal{L}_{\setminus\mathcal{I}}$.

Let us examine the case $m = 2$. Let the number of speakers hypothesized by $\mathcal{L}_{\setminus\mathcal{I}}$ be k'. The number of possibilities for $\mathcal{L}_{\mathcal{I}}$ is $(k'+1)^2 + 1$. (Each of the two recordings can belong independently to one of k' speakers, or be a new speaker, giving $(k'+1)^2$ possibilities, but there is another possibility that both recordings belong to the same new speaker.) In the general case, all partition possibilities having up to m new speakers, must be included. We cannot make m too large in this Gibbs sampler.

8.5.2 Approximate Gibbs sampler

For computational efficiency, we could construct an approximate Gibbs sampler and use that as the proposal distribution in a Metropolis-Hastings sampler, where the samples drawn from the proposal distribution are subjected to an acceptance test. Given the previously sampled partition $\mathcal{L}$, the next sample is obtained as follows:

1. Select $\mathcal{I} \subseteq \{1, \ldots, n\}$, randomly, or in round-robin fashion, such that $\mathcal{I}$ has m elements.

2. Obtain $\mathcal{L}_{\setminus\mathcal{I}}$ by removing all elements of $\mathcal{I}$ from $\mathcal{L}$. Denote the number of speakers in $\mathcal{L}_{\setminus i}$ by k'.

3. Sample the m new speaker labels independently (and therefore efficiently), as if this were an open-set speaker recognition problem, with k' enrolled speakers. This gives anywhere from 0 to m new speakers, say k'' of them.

4. If $k'' > 0$, sample a partition for this set of k'' recordings. (If this is done by some flavour of Markov chain Monte Carlo, some thought will have to be given to which initial state to use.)

5. Reassemble $\mathcal{L}$ from $\mathcal{L}_{\mathcal{I}}$ and $\mathcal{L}_{\setminus\mathcal{I}}$.

6. Apply the Metropolis-Hastings acceptance criterion to the new partition $\mathcal{L}$. For details, see for example [38], chapter 11, and references therein.

8.5.3 Deterministic annealing

A problem with the Gibbs sampler, especially if only one label is changed per iteration, is that it can mix[8] very slowly, getting stuck in some local mode of the posterior. For exponential family meta-embeddings, e.g. GMEs, the posterior can be flattened by multiplying the natural parameters with some factor smaller than one. This factor can then be gradually increased back to unity.

8.6 Contrastive divergence

One way to handle the intractable MCL objective is via *contrastive divergence* (CD) [53, 54]. Let $\tilde{P}$ denote an unnormalized distribution and express the relationship between unnormalized (LHS) and normalized (RHS) partition posterior be:

$$\tilde{P}(\mathcal{L} \mid \mathcal{R}, \theta) = Z(\mathcal{R}, \theta) P(\mathcal{L} \mid \mathcal{R}, \theta) \tag{8.12}$$

[8] MCMC slang for reaching equilibrium, where samples come from the true target distribution.

where $Z(\mathcal{R},\theta)$ is the troublesome normalization constant. In CD we do approximate gradient ascent on the MCL objective, $\log P(\mathcal{L}^* \mid \mathcal{R},\theta)$, where $\mathcal{L}^*$ is the true partition of $\mathcal{R}$ w.r.t. speaker. We make use of the fact that the required gradient can be expressed, without needing $Z(\mathcal{R},\theta)$, as an expectation w.r.t. the posterior:

$$\begin{aligned}
&\nabla_\theta \log P(\mathcal{L}^* \mid \mathcal{R},\theta) \\
&= \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \nabla_\theta \log Z(\mathcal{R},\theta) \\
&= \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \nabla_\theta \log \sum_{\mathcal{L}} \tilde{P}(\mathcal{L} \mid \mathcal{R},\theta) \\
&= \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \sum_{\mathcal{L}} \frac{\nabla_\theta \tilde{P}(\mathcal{L} \mid \mathcal{R},\theta)}{Z(\mathcal{R},\theta)} \\
&= \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \sum_{\mathcal{L}} \frac{\tilde{P}(\mathcal{L} \mid \mathcal{R},\theta)\nabla_\theta \log \tilde{P}(\mathcal{L} \mid \mathcal{R},\theta)}{Z(\mathcal{R},\theta)} \qquad (8.13) \\
&= \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \sum_{\mathcal{L}} P(\mathcal{L} \mid \mathcal{R},\theta)\nabla_\theta \log \tilde{P}(\mathcal{L} \mid \mathcal{R},\theta) \\
&= \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \left\langle \nabla_\theta \log \tilde{P}(\mathcal{L} \mid \mathcal{R},\theta) \right\rangle_{P(\mathcal{L}|\mathcal{R},\theta)} \\
&\approx \nabla_\theta \log \tilde{P}(\mathcal{L}^* \mid \mathcal{R},\theta) - \frac{1}{K} \sum_{\mathcal{L}} \nabla_\theta \log \tilde{P}(\mathcal{L} \mid \mathcal{R},\theta)
\end{aligned}$$

where K samples (partitions) are drawn from $P(\mathcal{L} \mid \mathcal{R},\theta)$. The samples can be drawn by a Monte Carlo procedure, for example Gibbs sampling. The CD recipe involves (approximate) gradient ascent updates to θ, interleaved with sampling at a fixed θ. For details, see for example [54].

8.7 Pseudolikelihood

Pseudolikelihood as an alternative to intractable likelihood has been around for decades [55], but it seems it has only very recently been pointed out that pseudolikelihood is also a proper scoring rule [47, 56].

Pseudolikelihood plays well with our CRP prior and indeed its calculation has much in common with the above Gibbs sampling procedures. For our problem, given a set, $\mathcal{R}$, of n recordings, the *pseudolikelihood* can be expressed as:

$$\Psi(\theta,\mathcal{L}) = \prod_{i=1}^{n} P(\ell_i \mid \mathcal{L}_{\setminus i},\mathcal{R},\theta) \qquad (8.14)$$

where θ represents the trainable parameters and where ℓ_i and $\mathcal{L}_{\setminus i}$ are as defined above. The RHS factors are the same as those required for the Gibbs sampler and can be computed using (8.11). We define the *pseudoscore* as:

$$S_\Psi(\theta, \mathcal{L}) = -\log \Psi(\theta, \mathcal{L}) = -\sum_{i=1}^{n} \log P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}, \theta) \tag{8.15}$$

To see that the pseudoscore forms a proper scoring rule, consider:

$$\begin{aligned}
&\min_\theta \big\langle S_\Psi(\theta, \mathcal{L}) \big\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)} \\
=&\min_\theta \Big\langle \sum_i -\log P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}, \theta) \Big\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)} \\
=&\min_\theta \sum_i \big\langle -\log P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}, \theta) \big\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)} \\
=&\min_\theta \sum_i \big\langle -\log P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}, \theta) \big\rangle_{P(\ell_i|\mathcal{L}_{\setminus i},\mathcal{R},\theta^*)P(\mathcal{L}_{\setminus i}|\mathcal{R},\theta^*)} \\
=&\min_\theta \sum_i \Big\langle \big\langle -\log P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}, \theta) \big\rangle_{P(\ell_i|\mathcal{L}_{\setminus i},\mathcal{R},\theta^*)} \Big\rangle_{P(\mathcal{L}_{\setminus i}|\mathcal{R},\theta^*)} \\
=&\sum_i \Big\langle \big\langle -\log P(\ell_i \mid \mathcal{L}_{\setminus i}, \mathcal{R}, \theta^*) \big\rangle_{P(\ell_i|\mathcal{L}_{\setminus i},\mathcal{R},\theta^*)} \Big\rangle_{P(\mathcal{L}_{\setminus i}|\mathcal{R},\theta^*)} \\
=&\big\langle S_\Psi(\theta^*) \big\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)}
\end{aligned} \tag{8.16}$$

where we have used the fact that the argument of the inner expectation is the logarithmic proper scoring rule, so that the inner expectation is minimized at θ^*, for every i and $\mathcal{L}_{\setminus i}$.

It can be shown that the pseudoscore is indeed also *strictly* proper. The pseudoscore is a special case of a *composite score* (to be discussed below). According to [47], a composite score is strictly proper provided that (i) every term in the composition is strictly proper; and that (ii) the individual terms completely determine the full joint distribution. The first condition is met because the pseudoscore is composed of logarithmic scores. The second condition is met thanks to *Brook's lemma* [57, 58].

8.7.1 Note on autoregressive full conditional likelihood

To better understand the relationship between the pseudolikelihood of (8.14) and the full (intractable) conditional likelihood, let us rewrite the full conditional likelihood in *autoregressive* form:

$$P(\mathcal{L} \mid \mathcal{R}) = \prod_{i=1}^{n} P(\ell_i \mid \mathcal{L}_{<i}, \mathcal{R}) \tag{8.17}$$

where $\mathcal{L}_{<i}$ is obtained from $\mathcal{L}$ by retaining all indices less than i and removing the rest. As before, ℓ_i is the speaker label for recording r_i and takes values in $\{1, \ldots, k'+1\}$, when there are k' speakers in $\mathcal{L}_{<i}$. The pseudolikelihood factors, $P(\ell_i \mid \mathcal{L}_{\backslash i}, \mathcal{R})$, are tractable because there are no hidden labels, but the autoregressive factors, $P(\ell_i \mid \mathcal{L}_{<i}, \mathcal{R})$, are intractable (except for small $n-i$), because all the missing labels ($n-i$ of them) have to be summed out.[9]

Finally, recall that the CRP prior *is* tractable in autoregressive form (8.7), but that is because the CRP distribution is exchangeable.

8.7.2 Example: MCL vs Pseudolikelihood

In an experiment similar to that of section 8.4.2, also with 8 recordings, we compare the MCL criterion with the pseudolikelihood. The result is shown in figure 8.2. Further calibration experiments were done on much larger synthetic datasets, for which pseudolikelihood remains tractable, but MCL not. The pseudolikelihood curves show the same behaviour, with minima at or close to a well-calibrated scale value of unity.

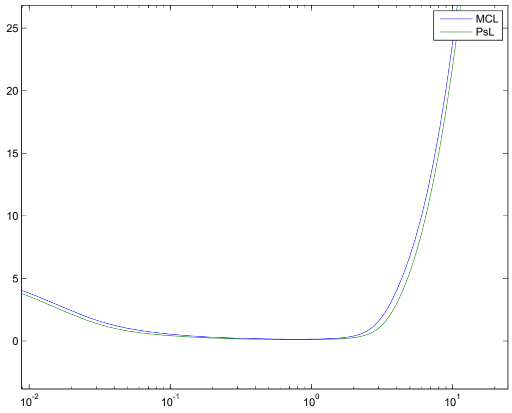


Figure 8.2: MCL vs Pseudolikelihood. The experiment is similar to that of section 8.4.2, although with different random data. The miscalibration scale is on the horizontal axis and the criteria on the vertical. This and other similar experiments show that MCL and Pseudolikelihood (PsL) behave in much the same way.

[9] $P(\ell_i \mid \mathcal{L}_{<i}, \mathcal{R}_{\leq i})$ *is* tractable, where $\mathcal{R}_{\leq i} = \{r_1, \ldots, r_i\}$.

8.7.3 Computation

The pseudolikelihood computational load scales as kn, where k is the number of speakers and n the number of recordings. In this respect it is similar to generative training of PLDA models and similar to discriminative cross-entropy training of multiclass classifiers. It must however be noted that pseudolikelihood may not play well with stochastic minibatch optimizers. The most important condition for stochastic optimization is that the objective function must decompose into a (large) number of terms. This is true of pseudolikelihood—it has n terms, but unfortunately every term involves statistics that are accumulated over the whole dataset.

8.8 Composite score

A family of proper scoring rules known as *composite scores* [47] can be constructed for our application as follows:

$$S_C(\theta, \mathcal{L}) = \sum_{i=1}^{N} S_i(\theta, \mathcal{L}) \tag{8.18}$$

where each S_i is a proper scoring rule operating on a conditional (or marginal) distribution of the form $P(\mathcal{L}^{(i)} \mid \mathcal{L}^{[i]}, \mathcal{R}, \theta)$, where $\mathcal{L}^{(i)}$ and $\mathcal{L}^{[i]}$ are subsets of the full label set, $\mathcal{L}$. For example, letting $N = n$, $\mathcal{L}^{(i)} = \ell_i$ and $\mathcal{L}^{[i]} = \mathcal{L}_{\setminus i}$, together with the logarithmic scoring rule, gives the pseudoscore. For later, it is important to note that while each of the S_i operates on some posterior conditioned on a subset of labels, the posterior is also conditioned on the *full* recording set $\mathcal{R}$.

To see that S_C is proper, consider:

$$\begin{aligned}
&\min_\theta \bigl\langle S_C(\theta, \mathcal{L}) \bigr\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)} \\
=&\min_\theta \sum_{i=1}^{N} \bigl\langle S_i(\theta, \mathcal{L}) \bigr\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)} \\
=&\min_\theta \sum_{i=1}^{N} \Bigl\langle \bigl\langle S_i(\theta, \mathcal{L}) \bigr\rangle_{P(\mathcal{L}^{(i)}|\mathcal{L}^{[i]},\mathcal{R},\theta^*)} \Bigr\rangle_{P(\mathcal{L}^{[i]}|\mathcal{R},\theta^*)} \\
=&\sum_{i=1}^{N} \Bigl\langle \bigl\langle S_i(\theta^*, \mathcal{L}) \bigr\rangle_{P(\mathcal{L}^{(i)}|\mathcal{L}^{[i]},\mathcal{R},\theta^*)} \Bigr\rangle_{P(\mathcal{L}^{[i]}|\mathcal{R},\theta^*)} \\
=&\bigl\langle S_C(\theta^*, \mathcal{L}) \bigr\rangle_{P(\mathcal{L}|\mathcal{R},\theta^*)}
\end{aligned} \tag{8.19}$$

where the third line follows because $S_i(\theta, \mathcal{L})$ is independent of any remaining labels in $\mathcal{L}$ that are not in $\mathcal{L}^{(i)}$ or $\mathcal{L}^{[i]}$. As before, the argument of the inner expectation is (by construction) a proper scoring rule, so that each inner expectation—and therefore the whole composition—is minimized at θ^*. As mentioned above, S_C will be *strictly* proper if all the S_i are strictly proper and all the $P(\mathcal{L}^{(i)} \mid \mathcal{L}^{[i]}, \mathcal{R}, \theta)$ together uniquely determine $P(\mathcal{L} \mid \mathcal{R}, \theta)$.

8.9 Composite likelihood

If the composite score is composed specifically of log scores, then the composite score is the negative logarithm of the more well-known *composite likelihood* [47, 59]. That is, we form the composite score as:

$$S_C(\theta, \mathcal{L}) = \sum_{i=1}^{N} -\log P(\mathcal{L}^{(i)} \mid \mathcal{L}^{[i]}, \mathcal{R}, \theta) \tag{8.20}$$

Below we explore a few different composite scores that differ in how $\mathcal{L}^{(i)}$ and $\mathcal{L}^{[i]}$ are defined. For the pseudolikelihood construction above, we split the information in $\mathcal{L}$ according to recording index. Below we explore more general ways of splitting the information.

8.9.1 Coarse conditioning

$\mathcal{L} \in \mathcal{P}_n$ is a partition w.r.t. speaker of the set of n recordings. For every i, also let $\mathcal{L}^{[i]} \in \mathcal{P}_n$, but subject to $\mathcal{L} < \mathcal{L}^{[i]}$, meaning that $\mathcal{L}^{[i]}$ is a strictly coarser partition than $\mathcal{L}$. Denote $\mathcal{L}^{[i]} : \mathcal{I}_{i1}, \mathcal{I}_{i2}, \ldots, \mathcal{I}_{im_i}$. The condition $\mathcal{L} < \mathcal{L}^{[i]}$ means that some or all of the index subsets are the unions of more than one speaker set. The important property of this construction is that for a given $\mathcal{L}^{[i]}$, the subsets do not overlap w.r.t. speaker. Now let $\mathcal{L}^{(i)}$ denote the *complement* of the information in $\mathcal{L}^{[i]}$, such that combining $\mathcal{L}^{[i]}$ and $\mathcal{L}^{(i)}$ recovers $\mathcal{L}$. We can now further decompose (8.20) as follows:

$$\begin{aligned} S_C(\theta, \mathcal{L}) &= \sum_{i=1}^{N} -\log P(\mathcal{L}^{(i)} \mid \mathcal{L}^{[i]}, \mathcal{R}, \theta) \\ &= \sum_{i=1}^{N} -\log \prod_{j=1}^{m_i} P(\mathcal{L}^{(ij)} \mid \mathcal{R}_{\mathcal{I}_{ij}}, \theta) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{m_i} -\log P(\mathcal{L}^{(ij)} \mid \mathcal{R}_{\mathcal{I}_{ij}}, \theta) \end{aligned} \tag{8.21}$$

where $\mathcal{L}^{(ij)}$ is a partition w.r.t. speaker of $\mathcal{R}_{\mathcal{I}_{ij}}$, the set of recordings indexed by $\mathcal{I}_{ij}$. The second line follows from the independence induced by the fact that the index subsets do not overlap w.r.t. speaker.

Although at $N=1$, we do have a proper scoring rule, the to-be-trained system never gets to demonstrate that it can differentiate between speakers in different subsets. It would be better to use a smallish $N>1$, with a few different groupings. Otherwise, in a stochastic training setting, we could repeatedly generate $\mathcal{L}^{[i]}$ at random.

8.9.2 Fine conditioning

In a dual recipe, we still have $\mathcal{L}, \mathcal{L}^{[i]} \in \mathcal{P}_n$, but now $\mathcal{L}^{[i]} < \mathcal{L}$, meaning every $\mathcal{L}^{[i]}$ is strictly finer than $\mathcal{L}$. Denote $\mathcal{L}^{[i]} : \mathcal{I}_{i1}, \mathcal{I}_{i2}, \dots, \mathcal{I}_{im_i}$. Every subset of $\mathcal{L}^{[i]}$ is pure (has exactly one speaker), but some subsets overlap w.r.t speaker. Again, $\mathcal{L}^{(i)}$ is defined as the complement, so that for every i, combining $\mathcal{L}^{[i]}$ and $\mathcal{L}^{(i)}$ recovers $\mathcal{L}$. In this case we have $\mathcal{L}^{(i)} \in \mathcal{P}_{m_i}$, meaning it is a partition of a set of m_i elements. In this case, term i of (8.21) is computed by first pooling the meta-embeddings according to the subsets $\mathcal{I}_{ij}$, resulting in m_i pooled meta-embeddings. If m_i is small enough, then all possible partitions of the set of pooled meta-embeddings can be visited, so that $P(\mathcal{L}^{(i)} \mid \mathcal{L}^{[i]}, \mathcal{R}, \theta)$ can be computed, complete with normalization constant.

Again, at $N=1$ we do have a proper scoring rule, but the system never gets to demonstrate that it can match the meta-embeddings within subsets. $N>1$ seems better.

8.9.3 Hybrid recipe

The coarse conditioning recipe will still be computationally demanding if there are too many recordings per speaker. Conversely, the fine conditioning recipe remains challenging when there are too may speakers. The two strategies can however be combined. Below we sketch an example of how to do this to get a stochastic gradient descent training recipe:

- Require:
  - Integers s, t, such that $2 \leq s \leq t$ and where the t-th Bell number, B_t is small.
  - Training data: a set of recordings and their true speaker labels.
  - A partition prior, $P(\mathcal{L})$, e.g. the CRP prior. The prior parameters can be fixed, or they can be optimized too.

- For each term, i, of the stochastic objective function, $-\log(\text{composite likelihood})$, do:
  - (Randomly) select a subset of s speakers.
  - Find all recordings of those speakers and extract their meta-embeddings, where the extractor is parametrized by θ. If there are more than t of these meta-embeddings, pool some of them, respecting speaker purity, until the number of (pooled and raw) meta-embeddings is k, such that $s \leq k \leq t$. Denote this set of k meta-embeddings as $\mathcal{F}_\theta^{[i]}$ and their true partition w.r.t. speaker as $\mathcal{L}^{(i)} \in \mathcal{P}_k$.
  - Compute the gradient for term i: $\nabla_\theta - \log P(\mathcal{L}^{(i)} \mid \mathcal{F}_\theta^{[i]})$. This requires computation of B_k different likelihoods.
- Accumulate gradients in mini-batches of say 10 or 100 terms.
- Do a gradient descent step, using your current favourite minibatch stochastic optimizer.

8.10 Binary cross-entropy

Binary cross-entropy, applied to pairs of recordings, is a popular discriminative training criterion in speaker recognition [60, 17, 61, 62, 7]. Binary cross-entropy can be interpreted as an approximation to a composite score. We choose the composite score as:

$$S_C(\theta, \mathcal{L}) = \sum_{i=1}^{N} -\log P(h_i \mid t_i, e_i, \mathcal{R}) \tag{8.22}$$

where $t_i, e_i \in \{1, \dots, n\}$ are two indices (enroll and test) that select a pair of recordings and $h_i \in \{H_1, H_2\}$ is the (true) hypothesis that states whether this pair was spoken by one or two speakers. We can use all pairs that can be formed from $\mathcal{R}$ (where $e_i \neq t_i$), or we can use some subset of pairs chosen via some heuristic [18]. (Given indices t_i, e_i, the true h_i can be found from $\mathcal{L}$.) Unfortunately, $P(h_i \mid t_i, e_i, \mathcal{R})$ is intractable, because it is conditioned on *all* the recordings in $\mathcal{R}$. For a given pair, the rest of $\mathcal{R}$ may still contain speech from the same speaker(s) present in the pair, so that we effectively have a large unsupervised dataset, requiring an intractable sum over all the possibilities. If however, the prior $P(\mathcal{L})$ is such that the large majority of the recordings in $\mathcal{R}$ do not have a speaker in common with the pair under test, then we could approximate:

$$\begin{aligned} S_C(\theta, \mathcal{L}) &= \sum_{i=1}^{N} -\log P(h_i \mid t_i, e_i, \mathcal{R}) \\ &\approx \sum_{i=1}^{N} -\log P(h_i \mid r_{t_i}, r_{e_i}) \end{aligned} \tag{8.23}$$

where $r_{t_i}, r_{e_i} \in \mathcal{R}$ are the test and enroll recordings, so that the RHS is the binary cross-entropy criterion.

8.10.1 Example

See figure 8.3 for a comparison of binary cross-entropy (all pairs) vs pseudolikelihood. The calibration behaviour of the two criteria are very similar. Since the binary cross-entropy uses all pairs, it is slower. Heuristics to choose a subset of the trials could improve this.

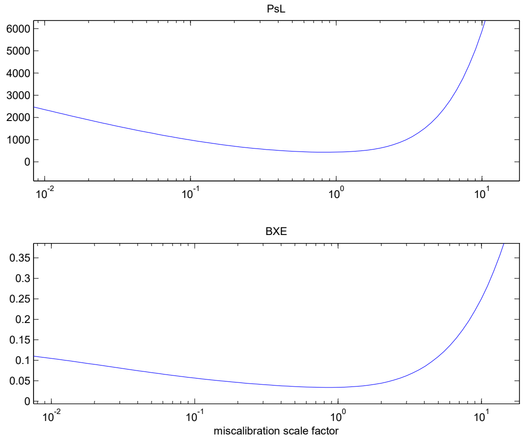


Figure 8.3: Pseudolikelihood vs binary cross-entropy. The experiment is similar to that of section 8.4.2, although with 1000 recordings. This and other similar experiments show that binary cross-entropy (BXE) and pseudolikelihood (PsL) behave in much the same way.

8.10.2 Note on triplet loss

First published in [2].

Good paper: [35]: explains original triplet loss, variants for hinge vs softplus loss and gives nice recipe for mining moderately hard triplets,

8.11 Multiclass cross-entropy

[8, 9]

8.12 Example: Full SGME training

With the following experiment, we close the loop, demonstrating that we can use pseudolikelihood to discriminatively train the full parameter set of a meta-embedding extractor (rather than just a calibration constant).

As in previous examples, synthetic i-vector-like data of dimension $D = 20$ was generated from the heavy-tailed PLDA model, with speaker identity variable of dimension $d = 2$, and $\nu = 3$. We used independently sampled data sets for train and test, each having 1000 recordings. The labels for train and test were also independently sampled from the CRP model, with $\beta = 0$ and α chosen to give an expected number of 100 speakers.

The discriminative SGME extractor was parameterized as follows:

- A $(D - d)$-by-D matrix, $\mathbf{H}$, used for $q_j = \mathbf{r}_j'\mathbf{H}'\mathbf{H}\mathbf{r}_j$, where $\mathbf{r}_j$ is an input vector. The meta-embedding natural parameter b_j is extracted as $b_j = \frac{\nu+D-d}{\nu+q_j}$. The true value of ν was given.
- A d-by-D matrix, $\mathbf{P}$ that does the same work as $\mathbf{V}'\mathbf{F}'\mathbf{W}$ in the generatively derived SGME of section 7.1. The meta-embedding natural parameter $\mathbf{a}_j$ is extracted $\mathbf{a}_j = b_j\mathbf{P}\mathbf{r}_j$.
- A d-dimensional vector, with non-negative components to represent the diagonal matrix $\boldsymbol{\Lambda}$ in (7.13).

The values of $\mathbf{H}, \mathbf{P}$ and $\boldsymbol{\Lambda}$ were found with a full batch L-BFGS optimizer. In this experiment (and in the larger one below) L-BFGS, running on a laptop, converged in a few minutes.

The accuracy of the discriminatively trained SGME extractor is compared against the generative SGME approximation of section 7.1, where we used the *true parameters* that generated the synthetic data. The results are given in the table below:

		S_Ψ	BXE	EER
discriminative	train	0.11	0.042	0.026
generative	train	0.13	0.042	0.026
discriminative	test	0.12	0.075	0.037
generative	test	0.11	0.067	0.034

Here S_Ψ represents negative log pseudolikelihood, conveniently normalized by $n\log(m)$, where n is the number of recordings and m the true number of speakers. BXE is binary-cross entropy (in bits), computed for the upper triangle of the symmetric matrix of LLR-scores of all possible pairs. EER is equal-error-rate (the ROCCH version [44]), computed from the same LLR-scores as the BXE criterion.[10]

Notice that there is some evidence of mild overtraining—on the train data, discriminative does better than generative, while on test, it is the other way round. Do keep in mind that this is an unfair comparison, because the generative method had access to the true parameters.

8.12.1 Using more data

A similar experiment, now with 5000 synthetic recordings (for each of train and test) and a prior expectation of 500 speakers, shows that the overtraining effect is less for bigger data sets:

		S_Ψ	BXE	EER
discriminative	train	0.27	0.032	0.054
generative	train	0.28	0.031	0.054
discriminative	test	0.28	0.030	0.054
generative	test	0.28	0.029	0.052

The accuracy[11] of likelihood-ratios computed for all pairs of recordings for the generative and discriminative systems are further analyzed in figure 8.4, using the well-known speaker recognition tools of DET-plots [63] and Bayes-error-rate plots [44, 64].

The DET-plot is a trade-off of miss rate against false-alarm rate (on a probit scale) as the score decision threshold is varied over the real line. It is a calibration-insensitive analysis of the decision-making ability of binary classifier scores. This plot shows that the two systems are essentially equivalent at all operating points and that the overtraining problem has virtually disappeared for the larger training set.[12]

The Bayes error-rate (aka DCF) plot is designed to be calibration sensitive. It shows the (normalized) prior-weighted average of the miss and

[10] See, for example the BOSARIS Toolkit for a tool to compute ROCCH EER: http://sites.google.com/site/bosaristoolkit/.

[11] The plots and the above table were for different experimental runs, with different model parameters that generated the synthetic data. The plot and table therefore show different accuracies.

[12] We ascribe the unfamiliar, steep angle of the DET plot to the use of low-dimensional synthetic data.

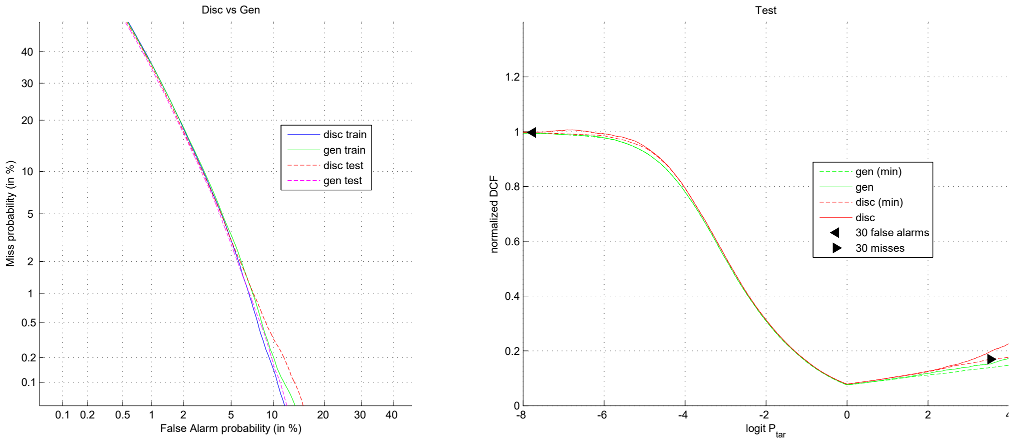


Figure 8.4: Pseudolikelihood SGME training. Left: DET plot showing accuracy of both systems on both train and test data. Right: Bayes error-rate plot for both systems on the test data. (The training data plot is similar).

false-alarm rates as the class prior, $P(\text{target}) = 1 - P(\text{non-target})$ is varied along the horizontal axis (on a logit scale). The decision threshold (applied to the log-likelihood-ratios) is the Bayes threshold, namely $-\log \frac{P(\text{target})}{P(\text{non-target})}$. See the references for further explanation. This plot shows that the calibration of both systems is good (solid plots lie on dashed plots) and that the two systems are also almost equivalent.

8.12.2 Using binary cross-entropy

The previous experiment (training size 5000) is repeated, but now using as training objective binary cross-entropy (BXE), rather than pseudolikelihood (PsL). The main difference is speed—the BXE complexity scales as n^2, while PsL scales as mn, where there are n recordings and m speakers. Apart from that, accuracy is similar, although there is still some evidence of overtraining:

		S_Ψ	BXE	EER
discriminative	train	0.26	0.024	0.046
generative	train	0.26	0.025	0.048
discriminative	test	0.29	0.025	0.051
generative	test	0.28	0.024	0.049

Bayes-error-rate plots are given in figure 8.5 and the DET plot in figure 8.6.

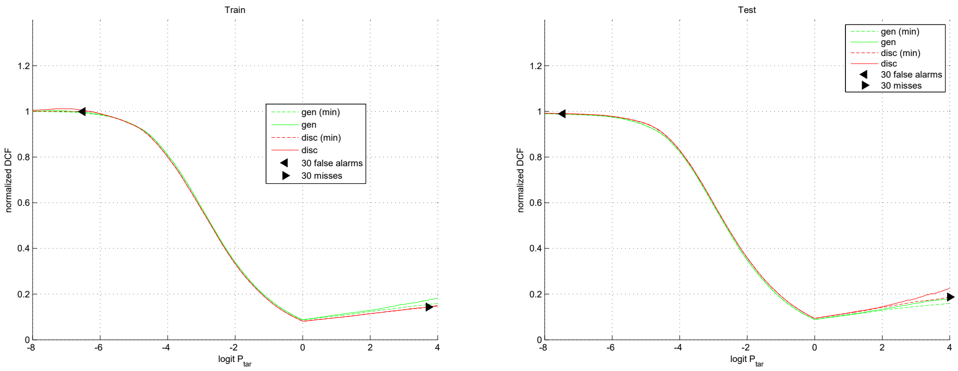


Figure 8.5: Binary cross-entropy SGME training. Bayes error-rate plot for both systems on the training data (left) and test data (right).

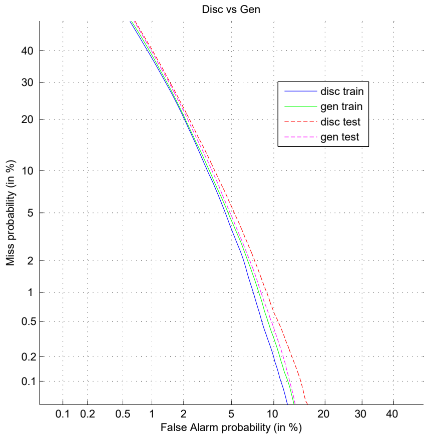


Figure 8.6: Binary cross-entropy SGME training. DET plot for both systems, evaluated on both training and test data.

Chapter 9

Unsupervised training

With some trouble, unsupervised training recipes can be found for a generative model of the observed data, $\mathcal{R}$. But can we also do unsupervised training of a discriminative the meta-embedding extractor, without having to train a full generative model? Below we consider first the generative case, before examining the discriminative case.

9.1 Unsupervised generative model training

For unsupervised training, we have a database of n recordings, denoted $\mathcal{R}$, but we do not have available, $\mathcal{L}^* \in \mathcal{P}_n$, the true partition w.r.t. speaker. We do need to assign some prior over partitions, say $P(\mathcal{L} \mid \gamma)$, where we have parametrized the prior by γ. In some situations we might assign a fixed γ, while in others it might make sense to learn γ as part of the training. In both cases, choice of the

The generative model provides the *likelihood*, $P(\mathcal{R} \mid \mathcal{L}, \phi)$, which is parametrized by ϕ, For convenience, we define $\theta = (\phi, \gamma)$, where it is understood γ might be fixed, or variable. The task of unsupervised generative training is then to solve:

$$\underset{\theta}{\operatorname{argmax}}\, P(\mathcal{R} \mid \theta) = \underset{\theta}{\operatorname{argmax}} \sum_{\mathcal{L} \in \mathcal{P}_n} P(\mathcal{L} \mid \gamma) P(\mathcal{R} \mid \mathcal{L}, \phi) \tag{9.1}$$

We already know that the summation is hopelessly intractable. We also know that even in the simpler case of fixed γ, stochastic approximation of the optimand, done by sampling from the prior, $P(\mathcal{L} \mid \gamma)$ is a bad idea, because the likelihood, $P(\mathcal{R} \mid \mathcal{L}, \phi)$ might have a very sharp peak, which will never be discovered by an affordable number of samples from the poor, dumb, clueless prior.

It is a much better plan to sample from the posterior, $P(\mathcal{L} \mid \mathcal{R}\,\theta)$, to approximate the expectation of the much flatter *log-likelihood*, rather than the sharply peaked likelihood. We discuss three closely related strategies that all give the above-mentioned benefits.

9.1.1 Stochastic variational Bayes

For stochastic variational Bayes (SVB) [65, 19, 36], we would be optimizing a stochastic approximation to the RHS of:

$$\log P(\mathcal{R} \mid \theta) \geq \left\langle \log \frac{P(\mathcal{R}, \mathcal{L} \mid \theta)}{Q(\mathcal{L} \mid \tilde{\theta})} \right\rangle_{Q(\mathcal{L}|\tilde{\theta})} \tag{9.2}$$

w.r.t. both the model parameters, θ and the variational parameters, $\tilde{\theta}$. If the constraints on the approximate posterior, $Q(\mathcal{L} \mid \tilde{\theta})$ are not too restrictive, then optimization w.r.t. $\tilde{\theta}$ would give a good approximation to the true posterior. However, this recipe is only do-able if the approximate posterior can be sampled from and can be *evaluated* (including its normalization constant) at every sample. At present, we are not aware of a suitable form for the approximate posterior that satisfies these requirements and therefore do not have a concrete plan to do SVB.

9.1.2 Monte Carlo EM

The SVB recipe can be simplified to Monte Carlo EM (MCEM) [66, 67, 68], *provided we can sample from the true posterior*, $P(\mathcal{L} \mid \mathcal{R}, \theta)$.

Starting from the more general VB solution, we let:

$$Q(\mathcal{L} \mid \tilde{\theta}) = P(\mathcal{L} \mid \mathcal{R}, \tilde{\theta})$$

which gives:

$$\log P(\mathcal{R} \mid \theta) \geq \left\langle \log \frac{P(\mathcal{R}, \mathcal{L} \mid \theta)}{P(\mathcal{L} \mid \mathcal{R}, \tilde{\theta})} \right\rangle_{P(\mathcal{L}|\mathcal{R},\tilde{\theta})} \tag{9.3}$$

with equality at $\theta = \tilde{\theta}$. Seen as VB, this means maximization w.r.t. $\tilde{\theta}$ is trivial and is done by choosing $\tilde{\theta} = \theta$. Given some initial value $\tilde{\theta}$, the basic MCEM algorithm proceeds by alternating the E and M steps below for a number of iterations:

E-step. Draw m samples, $\{\mathcal{L}_i\}_{i=1}^m$ from $P(\mathcal{L} \mid \mathcal{R}, \tilde{\theta})$.

M-step. Update: $\tilde{\theta} = \text{argmax}_\theta \sum_{i=1}^m \log P(\mathcal{R}, \mathcal{L}_i \mid \theta)$

For more detail, see the above references.

9.1.3 Contrastive divergence

The quirky *candidate's formula* [69] to evaluate an inconvenient marginal likelihood, is obtained by re-arranging the product rule as:

$$P(\mathcal{R} \mid \theta) = \frac{P(\mathcal{R}, \tilde{\mathcal{L}} \mid \theta)}{P(\tilde{\mathcal{L}} \mid \mathcal{R}, \theta)} \tag{9.4}$$

Since the LHS is independent of $\mathcal{L}$, the RHS can be evaluated at any $\mathcal{L} = \tilde{\mathcal{L}}$, provided the denominator is non-zero. For contrastive divergence (CD), we want to do gradient ascent on the log marginal likelihood, so we take logs and differentiate:

$$\nabla_\theta \log P(\mathcal{R} \mid \theta) = \nabla_\theta \log P(\mathcal{R}, \tilde{\mathcal{L}} \mid \theta) - \nabla_\theta \log P(\tilde{\mathcal{L}} \mid \mathcal{R}, \theta) \tag{9.5}$$

Since the joint is also the unnormalized posterior, we can define the unnormalized posterior as:

$$\tilde{P}(\mathcal{L} \mid \mathcal{R}, \theta)) = P(\mathcal{R}, \mathcal{L} \mid \theta) \propto P(\mathcal{L} \mid \mathcal{R}, \theta)$$

and plug that into (8.13) to expand the second term of the RHS of (9.5). The terms dependent on $\tilde{\mathcal{L}}$ cancel, to give:[1]

$$\nabla_\theta \log P(\mathcal{R} \mid \theta) = \left\langle \nabla_\theta \log P(\mathcal{R}, \mathcal{L} \mid \theta) \right\rangle_{P(\mathcal{L} \mid \mathcal{R}, \theta)} \tag{9.6}$$

As before, the RHS expectation can be approximated by sampling and the CD recipe does gradient ascent using these approximate gradients. This recipe has much in common with MCEM. The essential difference is that in MCEM, a sophisticated iterative numerical method might be used to find good optima in the M-step, while with CD, much simpler stochastic gradient ascent is used. With CD, many cheap M-steps are used compared to fewer, expensive M-steps in MCEM.

[1] The same result can be obtained by using the well known fact that at $\tilde{\theta} = \theta$, the gradients of the EM auxiliary and the log marginal likelihood coincide.

Appendix A

Multidimensional Fourier transform

In a mathematically rigorous treatment of Fourier transforms, one has to carefully define the function spaces in which the transforms can be applied [70]. In this document, we shall mostly just tacitly assume that wherever we employ definite integrals, that those integrals exist and are finite.

The *Fourier transform* (FT) is an *operator*—it maps one function to another function. We are interested in the n-dimensional FT, which maps some function, $f : \mathbb{R}^d \to \mathbb{C}$ to its transform, $\tilde{f} : \mathbb{R}^d \to \mathbb{C}$, where $\mathbb{R}^d$ is n-dimensional real Euclidean space and $\mathbb{C}$ is the field of complex numbers. Denoting the transform by $\tilde{f} = \mathcal{F}\{f\}$ and letting $\mathbf{z}, \boldsymbol{\zeta} \in \mathbb{R}^d$, we have:

$$\mathcal{F}\{f\}(\boldsymbol{\zeta}) = \tilde{f}(\boldsymbol{\zeta}) = \int_{\mathbb{R}^d} f(\mathbf{z}) e^{-2\pi i \mathbf{z}'\boldsymbol{\zeta}} \, d\mathbf{z} \tag{A.1}$$

where $\mathbf{z}'\boldsymbol{\zeta}$ denotes dot product. The inverse transform (IFT) is:

$$\mathcal{F}^{-1}\{\tilde{f}\}(\mathbf{z}) = f(\mathbf{z}) = \int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta}) e^{2\pi i \mathbf{z}'\boldsymbol{\zeta}} \, d\boldsymbol{\zeta} \tag{A.2}$$

Below we shall sometimes use the notation $\mathcal{F}_{\mathbf{z}}\{f(\mathbf{z}, \mathbf{y})\}$ to denote that the transform is done w.r.t. $\mathbf{z}$.

A.1 Properties

A.1.1 Symmetry

A real-valued function, $k : \mathbb{R}^d \to R$, which is also symmetric, $k(\mathbf{z}) = k(-\mathbf{z})$, has a real-valued FT: $\tilde{k}(\boldsymbol{\zeta}) = \overline{\tilde{k}(\boldsymbol{\zeta})}$, where the overline indicates complex conjugation. This follows readily from the definition (A.1).

A.1.2 The convolution theorem

Given functions f and g, their *convolution* is defined as:

$$(f * g)(\mathbf{z}) = \int_{\mathbb{R}^d} f(\mathbf{z}-\mathbf{y})g(\mathbf{y})\, d\mathbf{y} = \int_{\mathbb{R}^d} f(\mathbf{y})g(\mathbf{z}-\mathbf{y})\, d\mathbf{y} \tag{A.3}$$

which simplifies to multiplication in the transform domain:

$$\mathcal{F}\{f * g\}(\boldsymbol{\zeta}) = \tilde{f}(\boldsymbol{\zeta})\tilde{g}(\boldsymbol{\zeta}) \tag{A.4}$$

where $\tilde{f} = \mathcal{F}\{f\}$ and $\tilde{g} = \mathcal{F}\{g\}$.

A.1.3 Dirac delta

The *dirac Delta*, δ, has the property that:

$$\int_{\mathbb{R}^d} \delta(\mathbf{z}-\mathbf{y})f(\mathbf{z})\, d\mathbf{z} = f(\mathbf{y}) \tag{A.5}$$

which can be used to compute its FT as:

$$\mathcal{F}_{\mathbf{z}}\{\delta(\mathbf{z}-\mathbf{y})\}(\boldsymbol{\zeta}) = \int_{\mathbb{R}^d} \delta(\mathbf{z}-\mathbf{y})e^{-2\pi\mathbf{z}'\boldsymbol{\zeta}}\, d\mathbf{z} = e^{-2\pi i\mathbf{y}'\boldsymbol{\zeta}} \tag{A.6}$$

where we have introduced a subscript for the operator, $\mathcal{F}_{\mathbf{z}}$, to make clear that it operates on $\delta(\mathbf{z}-\mathbf{y})$ as a function of $\mathbf{z}$. Applying the IFT, $\mathcal{F}_{\boldsymbol{\zeta}}^{-1}$, on both sides of (A.6) gives:

$$\delta(\mathbf{z}-\mathbf{y}) = \int_{\mathbb{R}^d} e^{-2\pi i\mathbf{y}'\boldsymbol{\zeta}}e^{2\pi i\mathbf{z}'\boldsymbol{\zeta}}\, d\boldsymbol{\zeta} = \int_{\mathbb{R}^d} e^{2\pi i(\mathbf{z}-\mathbf{y})'\boldsymbol{\zeta}}\, d\boldsymbol{\zeta} \tag{A.7}$$

A.1.4 Shifting property

If $\mathcal{F}_{\mathbf{z}}\{f(\mathbf{z})\}(\boldsymbol{\zeta}) = \tilde{f}(\boldsymbol{\zeta})$, we want to express the FT of a shifted version of f, namely $f(\mathbf{z}-\mathbf{y})$ in terms of $\tilde{f}$. Letting $\mathbf{r} = \mathbf{z}-\mathbf{y}$, and noting that $d\mathbf{r} = d\mathbf{z}$, we have:

$$\begin{aligned}
\mathcal{F}_{\mathbf{z}}\{f(\mathbf{z}-\mathbf{y})\}(\boldsymbol{\zeta}) &= \int_{\mathbb{R}^d} f(\mathbf{z}-\mathbf{y})e^{-2\pi i\mathbf{z}'\boldsymbol{\zeta}}\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} f(\mathbf{r})e^{-2\pi i(\mathbf{r}+\mathbf{y})'\boldsymbol{\zeta}}\, d\mathbf{r} \\
&= e^{-2\pi i\mathbf{y}'\boldsymbol{\zeta}} \int_{\mathbb{R}^d} f(\mathbf{r})e^{-2\pi i\mathbf{r}'\boldsymbol{\zeta}}\, d\mathbf{r} \\
&= e^{-2\pi i\mathbf{y}'\boldsymbol{\zeta}}\tilde{f}(\boldsymbol{\zeta})
\end{aligned} \tag{A.8}$$

Shifting in the primary domain causes amplitude modulation in the transformed domain.

A.1.5 Linear transformation property

Let $\mathbf{y} = \mathbf{A}\mathbf{z}$, where $\mathbf{A}$ is an invertible square matrix. Then we have that $\mathbf{z} = \mathbf{A}^{-1}\mathbf{y}$ and $d\mathbf{y} = |\det(\mathbf{A})|\, d\mathbf{z}$.

$$\begin{aligned}
\mathcal{F}\{f(\mathbf{A}\mathbf{z})\}(\boldsymbol{\zeta}) &= \int_{\mathbb{R}^d} f(\mathbf{A}\mathbf{z})e^{-2\pi i \mathbf{z}'\boldsymbol{\zeta}}\, d\mathbf{z} \\
&= \frac{1}{|\det(\mathbf{A})|}\int_{\mathbb{R}^d} f(\mathbf{y})e^{-2\pi i (\mathbf{A}^{-1}\mathbf{y})'\boldsymbol{\zeta}}\, d\mathbf{y} \\
&= \frac{1}{|\det(\mathbf{A})|}\int_{\mathbb{R}^d} f(\mathbf{y})e^{-2\pi i \mathbf{y}'(\mathbf{A}^{-1})'\boldsymbol{\zeta}}\, d\mathbf{y} \\
&= \frac{1}{|\det(\mathbf{A})|}\mathcal{F}\{f\}\big((\mathbf{A}^{-1})'\boldsymbol{\zeta}\big)
\end{aligned} \tag{A.9}$$

A.1.6 Definite integral

It follows directly from the definition (A.1) that:

$$\int_{\mathbb{R}^d} f(\mathbf{z})\, d\mathbf{z} = \int_{\mathbb{R}^d} f(\mathbf{z})e^0\, d\mathbf{z} = \mathcal{F}\{f\}(0) \tag{A.10}$$

A.1.7 Parseval / Plancherel

Here we need to assume that f, g and their transforms, $\tilde{f}, \tilde{g}$ live in a Hilbert function space, with inner product defined as:

$$\langle f, g\rangle = \int_{\mathbb{R}^d} f(\mathbf{z})\overline{g(\mathbf{z})}\, d\mathbf{z} = \overline{\int_{\mathbb{R}^d} g(\mathbf{z})\overline{f(\mathbf{z})}\, d\mathbf{z}} = \overline{\langle g, f\rangle} \tag{A.11}$$

where the overline denotes complex conjugate. This Hilbert space is also the Lp function space, $L^2(\mathbb{R}^d)$, for which $\|f\|_2, \|g\|_2 \leq \infty$, where we use the Hilbert space norm $\|f\|_2 = \sqrt{\langle f, f\rangle}$.

Note the distinction between norm, $\|f\|_2$, which is a scalar and absolute value, $|f(\mathbf{z})|$ which is a function of $\mathbf{z}$. For a complex-valued function, the absolute value can be defined as as:

$$|f(\mathbf{z})| = \sqrt{f(\mathbf{z})\overline{f(\mathbf{z})}} \tag{A.12}$$

Of course, we have: $\|f\|_2^2 = \int_{\mathbb{R}^d} |f(\mathbf{z})|^2\, d\mathbf{z}$.

The Parseval / Plancherel[1] theorem equates the original inner-product to one between their Fourier transforms: $\langle f, g\rangle = \langle \tilde{f}, \tilde{g}\rangle$, where the RHS may

[1] One theorem says $\langle f, f\rangle = \langle \tilde{f}, \tilde{f}\rangle$ and the other, more generally, $\langle f, g\rangle = \langle \tilde{f}, \tilde{g}\rangle$, but I cannot figure out from the literature which one is Plancherel and which Parseval.

sometimes be more convenient to evaluate. More specifically, we also have $\|f\|_2 = \|\tilde{f}\|_2$. To show this, we expand the inner-product in terms of the inverse transforms of $\tilde{f}$ and $\tilde{g}$, rearrange, employ (A.7) and then (A.5):

$$\begin{aligned}
\langle f, g\rangle &= \int_{\mathbb{R}^d} f(\mathbf{z})\overline{g(\mathbf{z})}\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} \mathcal{F}^{-1}\{\tilde{f}\}(\mathbf{z})\overline{\mathcal{F}^{-1}\{\tilde{g}\}(\mathbf{z})}\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta})e^{2\pi i\boldsymbol{\zeta}'\mathbf{z}}\, d\boldsymbol{\zeta}\right)\overline{\left(\int_{\mathbb{R}^d} \tilde{g}(\mathbf{y})e^{2\pi i\mathbf{y}'\mathbf{z}}\, d\mathbf{y}\right)}\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta})e^{2\pi i\boldsymbol{\zeta}'\mathbf{z}}\, d\boldsymbol{\zeta}\right)\left(\int_{\mathbb{R}^d} \overline{\tilde{g}(\mathbf{y})}e^{-2\pi i\mathbf{y}'\mathbf{z}}\, d\mathbf{y}\right)\, d\mathbf{z} \\
&= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta})\overline{\tilde{g}(\mathbf{y})}\left(\int_{\mathbb{R}^d} e^{2\pi i(\boldsymbol{\zeta}-\mathbf{y})'\mathbf{z}}\, d\mathbf{z}\right)\, d\boldsymbol{\zeta}\, d\mathbf{y} \\
&= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta})\overline{\tilde{g}(\mathbf{y})}\delta(\boldsymbol{\zeta}-\mathbf{y})\, d\boldsymbol{\zeta}\, d\mathbf{y} \\
&= \int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta})\overline{\tilde{g}(\boldsymbol{\zeta})}\, d\boldsymbol{\zeta} \\
&= \langle\tilde{f}, \tilde{g}\rangle
\end{aligned} \tag{A.13}$$

This means we can form the dot product either in the original domain, or in the transformed domain.

A.2 Characteristic function

The characteristic function of a probability distribution, P, over $\mathbf{z} \in \mathbb{R}^d$ is usually defined as, $\phi_P : \mathbb{R}^d \to \mathbb{C}$, where:

$$\phi_P(\mathbf{t}) = \left\langle e^{i\mathbf{t}'\mathbf{z}}\right\rangle_{\mathbf{z}\sim P} \tag{A.14}$$

Notice that since $|e^{i\mathbf{t}'\mathbf{z}}| \le 1$, we also have $|\phi_P(\mathbf{t})| \le 1$. Indeed, the characteristic function always exists. If P has a density function, $p(\mathbf{z})$, then density and characteristic function essentially form a Fourier transform pair:

$$\phi_p(\mathbf{t}) = \mathcal{C}\{p\}(\mathbf{t}) = \int_{\mathbb{R}^d} e^{i\mathbf{t}'\mathbf{z}}p(\mathbf{z})\, d\mathbf{z} \tag{A.15}$$

where we denote this reparametrized version of the Fourier transform by the operator $\mathcal{C}$. Observe that if we let $\mathbf{t} = -2\pi\boldsymbol{\zeta}$, then:

$$\phi_p(-2\pi\boldsymbol{\zeta}) = \int_{\mathbb{R}^d} e^{-2\pi i\boldsymbol{\zeta}'\mathbf{z}}p(\mathbf{z})\, d\mathbf{z} = \tilde{p}(\boldsymbol{\zeta}) \tag{A.16}$$

or

$$\mathcal{C}\{p\}(-2\pi\boldsymbol{\zeta}) = \mathcal{F}\{p\}(\boldsymbol{\zeta}) \tag{A.17}$$

The characteristics functions for most standard probability distributions are known [71].

A.2.1 Parseval

How does Parseval's theorem translate in terms of characteristic functions? Let f, g be probability density functions with characteristic functions, ϕ_f and ϕ_g. Then, letting $\mathbf{t} = -2\pi\boldsymbol{\zeta}$ and $d\mathbf{t} = (2\pi)^d d\boldsymbol{\zeta}$

$$\begin{aligned}\int_{\mathbb{R}^d} f(\mathbf{z})g(\mathbf{z})\,d\mathbf{z} &= \int_{\mathbb{R}^d} \tilde{f}(\boldsymbol{\zeta})\overline{\tilde{g}(\boldsymbol{\zeta})}\,d\boldsymbol{\zeta}\\ &= \int_{\mathbb{R}^d} \phi_f(-2\pi\boldsymbol{\zeta})\overline{\phi_g(-2\pi\boldsymbol{\zeta})}\,d\boldsymbol{\zeta}\\ &= \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \phi_f(\mathbf{t})\overline{\phi_g(\mathbf{t})}\,d\mathbf{t}\end{aligned} \tag{A.18}$$

A.2.2 Convolution

The convolution theorem also works for the characteristic function [71]:

$$\begin{aligned}\mathcal{C}\{f * g\}(\mathbf{t}) &= \mathcal{F}\{f * g\}\big(\frac{-1}{2\pi}\mathbf{t}\big)\\ &= \mathcal{F}\{f\}\big(\frac{-1}{2\pi}\mathbf{t}\big)\,\mathcal{F}\{g\}\big(\frac{-1}{2\pi}\mathbf{t}\big)\\ &= \mathcal{C}\{f\}(\mathbf{t})\,\mathcal{C}\{g\}(\mathbf{t})\end{aligned} \tag{A.19}$$

A.2.3 Gaussian characteristic function

The characteristic function for a multivariate Gaussian is [71]:

$$\begin{aligned}\mathcal{C}\{\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\}(\mathbf{t}) &= \int_{\mathbb{R}^d} \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \exp(i\mathbf{t}'\mathbf{z})\,d\mathbf{z}\\ &= \exp(i\boldsymbol{\mu}'\mathbf{t} - \frac{1}{2}\mathbf{t}'\boldsymbol{\Sigma}\mathbf{t})\\ &= e^{i\boldsymbol{\mu}'\mathbf{t}}\sqrt{|2\pi\boldsymbol{\Sigma}^{-1}|}\,\mathcal{N}(\mathbf{t} \mid \mathbf{0}, \boldsymbol{\Sigma}^{-1})\end{aligned} \tag{A.20}$$

Notice that for zero mean, $\boldsymbol{\mu} = \mathbf{0}$, the characteristic function is also an unnormalized zero mean Gaussian, with the roles of covariance and precision interchanged. As the PDF gets more concentrated, its transform gets more spread out—this inverse scaling is of course due to (A.9). If $\boldsymbol{\mu}$ is non-zero, then the characteristic function is not even real-valued and is therefore not a probability density.

A.2.4 Empirical characteristic function

The empirical distribution of m observed data points, $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m \in \mathbb{R}^d$, does not have a density function in the strict sense, but the following mixture of Dirac delta's can be used for most purposes as a drop-in replacement for the density:

$$p(\mathbf{z}) = \frac{1}{m}\sum_{\ell=1}^{m} \delta(\mathbf{z} - \mathbf{z}_\ell) \tag{A.21}$$

The characteristic function, using (A.6), is:

$$\tilde{p}(\boldsymbol{\zeta}) = \frac{1}{m}\sum_{\ell=1}^{m} e^{-2\pi i \boldsymbol{\zeta}' \mathbf{z}_\ell} \tag{A.22}$$

A.2.5 Positive definite function

We term a function $k : \mathbb{R}^d \to \mathbb{C}$ *positive definite*, if for every $m \geq 1$ and every $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m \in \mathbb{R}^d$ and every $\alpha_1, \alpha_2, \dots, \alpha_m \in \mathbb{C}$, we have:

$$\sum_{\ell=1}^{m}\sum_{j=1}^{m} \alpha_\ell \overline{\alpha_j} k(\mathbf{z}_\ell - \mathbf{z}_j) \geq 0 \tag{A.23}$$

This is the same as requiring that any m-by-m Gram matrix formed with elements $k(\mathbf{z}_i - \mathbf{z}_j)$ must be Hermitian and positive (semi) definite.

Notice that for positive definite k, we can define a shift-invariant *kernel function*, $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$, as $K(\mathbf{z}, \mathbf{y}) = k(\mathbf{z} - \mathbf{y})$, so that K is positive definite in the sense required for the theory of reproducing kernel Hilbert spaces (RKHS) and kernel methods in machine learning (SVM, Gaussian processes, etc.).

A.2.6 Bochner's theorem

Let $r(\mathbf{z})$, be a *probability density function* on $\mathbf{z} \in \mathbb{R}^d$. We show that its characteristic function, $\tilde{r}$, is positive definite:

$$\begin{aligned}
&\sum_{\ell=1}^{m}\sum_{j=1}^{m} \alpha_\ell \overline{\alpha_j} \tilde{r}(\boldsymbol{\zeta}_\ell - \boldsymbol{\zeta}_j) \\
&= \sum_{\ell=1}^{m}\sum_{j=1}^{m} \alpha_\ell \overline{\alpha_j} \int_{\mathbb{R}^d} r(\mathbf{z}) e^{-2\pi i (\boldsymbol{\zeta}_\ell - \boldsymbol{\zeta}_j)'\mathbf{z}} \, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} r(\mathbf{z}) \sum_{\ell=1}^{m}\sum_{j=1}^{m} \alpha_\ell e^{-2\pi i \boldsymbol{\zeta}_\ell' \mathbf{z}} \overline{\alpha_j e^{-2\pi i \boldsymbol{\zeta}_j' \mathbf{z}}} \, d\mathbf{z} \\
&= \int_{\mathbb{R}^d} r(\mathbf{z}) \left| \sum_{\ell=1}^{m} \alpha_\ell e^{-2\pi i \boldsymbol{\zeta}_\ell' \mathbf{z}} \right|^2 \, d\mathbf{z} \geq 0
\end{aligned} \tag{A.24}$$

Since $r(\mathbf{z})$ is normalized, we also have by (A.10), that $\tilde{r}(\mathbf{0}) = 1$. Bochner's theorem says that the converse (although harder to prove) is also true—the FT (or IFT) of a positive definite function (which evaluates to 1 at $\mathbf{0}$), is a probability density [70].

Appendix B

Some properties of multivariate Gaussians

B.1 Two ways to express conditionals

Consider $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and define the following partitions:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}$$
$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{bmatrix} \qquad \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix}$$

Then, Bishop [38] gives the following formula for the conditional, expressed in terms of the precision blocks:

$$P(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}), \quad \boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b) \tag{B.1}$$

On the other hand, [72] gives an equivalent formula in terms of the covariance blocks:

$$P(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{a|b}, \boldsymbol{\Sigma}_{a|b}), \quad \boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba}(\mathbf{x}_b - \boldsymbol{\mu}_b) \tag{B.2}$$

where

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Lambda}_{aa}^{-1} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba} \tag{B.3}$$

B.2 Predictive properties of the precision matrix

Let $(x_1, \dots, x_n) \sim \mathcal{N}(\boldsymbol{\mu} \mid \boldsymbol{\Sigma})$. The covariance matrix, $\boldsymbol{\Sigma}$ encodes unconditional dependency: for $i \neq j$, if covariance element i, j is zero (non-zero), then x_i and x_j are independent (dependent). The precision matrix, $\mathbf{\Lambda} = \mathbf{\Sigma}^{-1}$ encodes *conditional* dependency. Let $\mathbf{x}_{-ij}$ denote all of the elements of $\mathbf{x}$, except x_i and x_j. For $i \neq j$, if precision element i, j is zero (non-zero), then x_i and x_j are conditionally independent (dependent), given $\mathbf{x}_{-ij}$. For further reading on the close association between the precision matrix and the undirected graphical model for multivariate Gaussians, see the Gaussian Markov random field (GMRF) literature, for example [72].

B.2.1 Complete conditional

The above conditional dependency properties of the precision matrix can be more specifically expressed in terms of the following predictive distribution (see [72], theorem 2.3). Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu} \mid \mathbf{\Lambda}^{-1})$, where x_i, μ_i and λ_{ij} are elements of $\mathbf{x}, \boldsymbol{\mu}$ and $\mathbf{\Lambda}$. Let $\mathbf{x}_{-i}$ denote all of the elements of $\mathbf{x}$, except x_i. Then for any component x_i, we have the univariate Gaussian predictive distribution:[1]

$$x_i \mid \mathbf{x}_{-i} \sim \mathcal{N}(\hat{x}_{-i}, \lambda_{ii}^{-1}) \quad \text{where} \quad \hat{x}_{-i} = \mu_i - \lambda_{ii}^{-1} \sum_{j:j\neq i} \lambda_{ij}(x_j - \mu_j) \tag{B.4}$$

Notice that the product of these univariate density factors would give the pseudolikelihood, while sampling from them would be equivalent to Gibbs sampling.

B.2.2 Autoregressive conditional

Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu} \mid \mathbf{\Lambda}^{-1})$ and let $\mathbf{L}$ be the (lower triangular) Cholesky decomposition of $\mathbf{\Lambda}$, so that $\mathbf{L}\mathbf{L}' = \mathbf{\Lambda}$. Let ℓ_{ij} denote a component of $\mathbf{L}$ and let $\mathbf{x}_{>i}$ denote all the components of $\mathbf{x}$ that have indices greater than i. Then for any component x_i, we have the univariate Gaussian predictive distribution ([72], theorem 2.7):

$$x_i \mid \mathbf{x}_{>i} \sim \mathcal{N}(\hat{x}_{>i}, \ell_{ii}^{-2}) \quad \text{where} \quad \hat{x}_{>i} = \mu_i - \ell_{ii}^{-1} \sum_{j:j>i} \ell_{ij}(x_j - \mu_j) \tag{B.5}$$

In contrast to (B.4), we see that (B.5) gives an autoregressive predictor, because the product of all the univariate density factors gives the full multivariate density:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \mathbf{\Lambda}^{-1}) = \prod_{i=1}^{n} \mathcal{N}(x_i \mid \hat{x}_{>i}, \ell_{ii}^{-2}) \tag{B.6}$$

[1] In fact, this is just a special case of (B.1).

With this decomposition, sampling can also be done more efficiently.[2] Specifically, we can use ancestral sampling, starting at $i = n$ and working back to $i = 1$. (It is left as an exercise to the reader to check whether the upper triangular Cholesky variant would give a recipe that samples from 1 to n.)

[2] The disadvantages of Gibbs sampling includes burn-in time and samples which are not independent.

Bibliography

[1] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *NIPS*, 2000.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015. [Online]. Available: http://arxiv.org/abs/1503.03832

[3] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Interspeech*, Brighton, UK, September 2009.

[4] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

[5] D. Martínez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," in *Interspeech*, 2011.

[6] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Interspeech*, 2017. [Online]. Available: https://arxiv.org/abs/1706.08612

[7] D. Snyder, P. Ghahremani, and D. Povey, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *IEEE Workshop on Spoken Language Technology*, 2016.

[8] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Interspeech*, Stockholm, 2017.

[9] G. Gelly and J. L. Gauvain, "Spoken language identification using lstm-based angular proximity," in *Interspeech*, Stockholm, 2017.

[10] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," CRIM, Montreal, Tech. Rep. CRIM-06/08-13, 2005.

[11] ——, "Bayesian speaker verification with heavy-tailed priors," in *Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010, keynote presentation.

[12] S. Ioffe, "Probabilistic linear discriminant analysis," in *9th European Conference on Computer Vision*, Graz, Austria, 2006.

[13] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE 11th International Conference on Computer Vision*, 2007.

[14] P. Li, Y. Fu, U. Mohammed, J. H. Elder, and S. J. Prince, "Probabilistic linear discriminant analysis for inferences about identity," *IEEE Trans. PAMI*, vol. 34, no. 1, January 2012.

[15] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Odyssey 2010, The Speaker and Language Recognition Workshop, Brno*, 2010.

[16] N. Brümmer and E. de Villiers, "The speaker partitioning problem," in *Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[17] S. Cumani, N. Brümmer, L. Burget, P. Laface, O. Plchot, and V. Vasilakakis, "Pairwise discriminative speaker verification in the i-vector space," *IEEE Transactions on audio, speech and language processing*, vol. 21, no. 6, pp. 1217–1227, 2013.

[18] S. Cumani and P. Laface, "Large scale training of pairwise support vector machines for speaker recognition," *IEEE Transactions on audio, speech and language processing*, vol. 22, no. 11, pp. 1590–1600, 2014.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[20] L. Vilnis and A. McCallum, "Word representations via Gaussian embedding," in *ICLR*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6623

[21] S. Cumani, O. Plchot, and P. Laface, "On the use of i-vector posterior distributions in PLDA," *IEEE Trans. ASLP*, vol. 22, no. 4, 2014.

[22] P. Kenny, T. Stafylakis, P. Ouellet, M. J. Alam, and P. Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *IEEE ICASSP*, 2013.

[23] T. Stafylakis, P. Kenny, P. Ouellet, J. Perez, M. Kockmann, and P. Dumouchel, "Text-dependent speaker recognition using PLDA with uncertainty propagation," in *Interspeech*, 2013.

[24] P. Kenny, T. Stafylakis, J. Alam, V. Gupta, and M. Kockmann, "Uncertainty modeling without subspace methods for text-dependent speaker recognition," in *Speaker Odyssey: The Speaker and Language Recognition Workshop*, Bilbao, 2016.

[25] F. Kelly and N. Harte, "Effects of long-term ageing on speaker verification," *Biometrics and ID Management*, 2011.

[26] J. Villalba and N. Brümmer, "Towards fully Bayesian speaker recognition: Integrating out the between-speaker covariance," in *Interspeech*, Florence, 2011.

[27] Y. S. Chow and H. Teicher, *Probability theory: Independence, interchangeability, martingales*, 3rd ed., ser. Springer Texts in Statistics. New York: Springer, 1997.

[28] N. Brümmer and J. du Preez, "Application independent evaluation of speaker detection," *Computer Speech and Language*, vol. 20, no. 2-3, pp. 230–275, 2006.

[29] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[30] K. Siegrist, "Random: Probability, mathematical statistics, stochastic processes," 1997, see section 3.11 Vector Spaces of Random Variables. [Online]. Available: http://www.math.uah.edu/stat/expect/Spaces.html

[31] P. Billingsley, *Probability and Measure*, 3rd ed. John Wiley & Sons, 1995.

[32] P. Ouwehand, "Spaces of random variables," AIMS, lecture, November 2010. [Online]. Available: http://users.aims.ac.za/~pouw/Lectures/Lecture_Spaces_Random_Variables.pdf

[33] D. Bigoni, "Uncertainty quantification with applications to engineering problems," Ph.D. dissertation, Thechnical University of Denmark, 2015, see Appendix B: Probability theory and functional spaces. [Online]. Available: http://orbit.dtu.dk/files/106969507/phd359_Bigoni_D.pdf

[34] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Interspeech*, Florence, Italy, 2011.

[35] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017. [Online]. Available: http://arxiv.org/abs/1703.07737

[36] M. Titsias and M. Lázaro-Gredilla, "Doubly stochastic variational Bayes for non-conjugate inference," in *ICML*, 2014. [Online]. Available: www.jmlr.org/proceedings/papers/v32/titsias14.pdf

[37] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[38] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[39] A. F. Martin, C. S. Greenberg, J. M. Howard, G. R. Doddington, and J. J. Godfrey, "NIST language recognition evaluation: Past and future," in *Speaker Odyssey: The Speaker and Language Recognition Workshop*, Joensuu, Finland, 2014.

[40] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[41] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 447, pp. 359–378, 2007.

[42] A. P. Dawid, M. Musio, and L. Ventura, "Minimum scoring rule inference," *Scandinavian Journal of Statistics*, vol. 43, pp. 123–138, 2016.

[43] M. H. DeGroot, *Optimal Statistical Decisions.* McGraw-Hill, 1970.

[44] N. Brümmer, "Measuring, refining and calibrating speaker and language information extracted from speech," Ph.D. dissertation, Stellenbosch University, 2010. [Online]. Available: https://scholar.sun.ac.za/handle/10019.1/5139

[45] A. Buja, W. Stuetzle, and Y. Shen, "Loss functions for binary class probability estimation and classification: Structure and applications," Statistics Department, The Wharton School, University of Pennsylvania, Tech. Rep., November 2005.

[46] N. Brümmer and G. Doddington, "Likelihood-ratio calibration using prior-weighted proper scoring rules," in *Interspeech*, Lyon, 2013.

[47] A. Dawid and M. Musio, "Theory and applications of proper scoring rules," *Metron*, vol. 72, 2014. [Online]. Available: http://arxiv.org/abs/1401.0398

[48] ——, "Bayesian model selection based on proper scoring rules," *Bayesian Analysis*, vol. 10, no. 2, 2015. [Online]. Available: http://arxiv.org/abs/1409.5291

[49] S. Goldwater, T. L. Griffiths, and M. Johnson, "Producing power-law distributions and damping word frequencies with two-stage language models," *Journal of machine Learning Research*, 2011. [Online]. Available: www.jmlr.org/papers/volume12/goldwater11a/goldwater11a.pdf

[50] J. Pitman, "Exchangeable and partially exchangeable random partitions," *Probability Theory and Related Fields*, vol. 102, pp. 145–158, 1995. [Online]. Available: www.stat.berkeley.edu/~aldous/206-Exch/Papers/pitman95a.pdf

[51] ——, *Combinatorial Stochastic Processes: Ecole d'Eté de Probabilités de Saint-Flour XXXII 2002*, J. Picard, Ed. Springer-Verlag, 2006.

[52] M. Orlov, "Efficient generation of set partitions," Computer Science Department of Ben-Gurion University in Israel, Tech. Rep., March 2002. [Online]. Available: http://www.cs.bgu.ac.il/~orlovm/papers/partitions.pdf

[53] G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[54] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *International Conference on Machine Learning*, Helsinki, Finland, 2008.

[55] J. Besag, "Statistical analysis of non-lattice data," *Journal of the Royal Statistical Society, series D (The Statistician)*, vol. 24, pp. 179–195, 1975.

[56] A. P. Dawid, S. Lauritzen, and M. Parry, "Proper local scoring rules on discrete sample spaces," *The Annals of Statistics*, 2012. [Online]. Available: http://arxiv.org/abs/1104.2224

[57] S. Lyu, "Interpretation and generalization of score matching," in *Conference on Uncertainty in Artificial Intelligence*, Montreal, December 2009.

[58] D. Brook, "On the distinction between the conditional probability and the joint probability approaches in the specification of nearest-neighbour systems," *Biometrika*, vol. 3,4, no. 51, pp. 481–483, December 1964.

[59] *Statistica Sinica*, vol. 21, no. 1, 2011, Special issue on composite likelihood. [Online]. Available: http://www3.stat.sinica.edu.tw/statistica/j21n1/21-1.html

[60] N. Brümmer, L. Burget, J. Černocky, O. Glembek, F. Grézl, M. Karafíat, D. A. van Leeuwen, P. Matějka, P. Schwarz, and A. Strasheim, "Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST Speaker Recognition Evaluation 2006," *IEEE TASLP*, vol. 15, no. 7, September 2007.

[61] N. Brümmer, A. Swart, and D. van Leeuwen, "A comparison of linear and non-linear calibrations for speaker recognition," in *Odyssey 2014: The Speaker and Language Recognition Workshop*, Joensuu, 2014. [Online]. Available: http://arxiv.org/abs/1402.2447

[62] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*, 2016. [Online]. Available: http://arxiv.org/abs/1509.08062

[63] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," in *EUROSPEECH*, Rhodes, Greece, Sept 1997, pp. 1895–1898.

[64] N. Brümmer and E. de Villiers, "The BOSARIS Toolkit: Theory, algorithms and code for surviving the New DCF," in *The NIST SRE'11 Analysis Workshop*, Atlanta, December 2011. [Online]. Available: http://arxiv.org/abs/1304.2865

[65] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1303–1347, May 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2502581.2502622

[66] G. Wei and M. Tanner, “A Monte Carlo implementation of the EM algorithm and the poor man’s data augmentation algorithms,” *Journal of the American Statistical Association*, vol. 85, pp. 699–704, 1990.

[67] R. Levine and G. Casella, “Implementations of the Monte Carlo EM algorithm,” *Journal of Computational and Graphical Statistics*, vol. 10, pp. 422–439, 2001.

[68] R. C. Neath, *On Convergence Properties of the Monte Carlo EM Algorithm*, ser. Collections. Beachwood, Ohio, USA: Institute of Mathematical Statistics, 2013, vol. Volume 10, pp. 43–62. [Online]. Available: https://doi.org/10.1214/12-IMSCOLL1003

[69] J. Besag, “A candidate’s formula: A curious result in Bayesian prediction,” *Biometrika*, vol. 76, 1989. [Online]. Available: biomet.oxfordjournals.org/content/76/1/183.abstract

[70] E. M. Stein and G. Weiss, *Introduction to Fourier Analysis on Euclidean Spaces*. Princeton University Press, 1971.

[71] F. Oberhettinger, *Fourier Transforms of Distributions and their Inverses: A Collection of Tables*. Academic Press, 1973.

[72] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*. Chapman & Hall, 2005.