Paper Template for INTERSPEECH

Arvid Fahlström Myrman and Giampiero Salvi

KTH Royal Institute of Technology,
School of Computer Science and Communication,
Dept. of Speech, Music and Hearning, Stockholm, Sweden

{arvidfm, giampi}@kth.se

Abstract

This work seeks to find speaker-invariant speech features which can be used to discriminate between different linguistic units in a robust manner. We focus in particular on simpler, interpretable models that improve on features inferred from unlabelled data, thus also making use of the full set of unlabelled data, in addition to speech fragment information. The goal is for the model to noticably improve on the input features, as well as to achieve competitive results in the context of the first track of the Zero Resource Speech Challenge.

Index Terms: zero-resource speech challenge, unsupervised learning, speech recognition

1. Introduction

The area of unsupervised learning in speech technology has been active for many decades in the attempt to reduce the amount of handcrafted information needed to build speech recognition and synthesis systems. Many aspects of language acquisition have been covered, from the discovery of alternative sub-word units compared to pre-defined phonemes [1]–[5] to the discovery of recurrent patterns that may constitute word candidates [6]–[12]. Recently, Versteegh *et al.* introduced the Zero Resource Speech Challenge [13], which was developed with the goal of standardising these endeavours. Specifically, the first track, most relevant to this paper, involves finding speaker independent linguistic units from speech with no or weak supervision. The discovery of linguistic units follows two main approaches in the literature that we will here refer to as bottom-up and top-down. In the first case, acoustic models are inferred directly from the acoustic features [14]–[21]. The second approach is to first segment the speech into syllable- or word-like units, and afterwards break these units into smaller subword units [13], [19], [22]–[30]

1.1. Related Work: Bottom-Up Approaches

Varadarajan *et al.* [14] first define a one-state HMM, and then iteratively split and merge states as needed to account for the data according to a heuristic. It should be noted, however, that in order map each state (allophones) into phonemes, they train a separate model using labelled speech, making the method not fully unsupervised. Lee and Glass [15] use an infinite mixture model of tri-state HMM-GMMs that performs segmentation and acoustic modelling joinly. Inference of the model is done using Gibbs sampling. A similar model but, without constraints on the topology of the HMMs was studied in [12]. Siu *et al.* [16] first use a segmental GMM (SGMM) to generate a transcription of the data and then iteratively train a standard HMM and improve the transcriptions by, in turns, maximising the likelihood of the model parameters given the transcription, and the transcription given the model parameters. Note that the number of allowed states are here defined in advance.

Diverging from previous approaches using temporal models, Chen *et al.* [17] perform standard clustering of speech frames using an infinite Gaussian mixture model. After training, the speech frames are represented as posteriorgrams, which have been shown to be more speaker-invariant than other features such as MFCCs [18]. Despite the simple approach, this turned out to be the overall best-performing model in the first track of the 2015 Zero Resource Speech Challenge [19]. Heck *et al.* [20] further improved on the model by performing clustering in two stages, with an intermediate supervised dimensionality reduction step using the clusters derived from the first clustering step as target classes.

Synnaeve and Dupoux [21] use a siamese network [31] to create an embedding where speech frames close to each other are considered to belong to the same subword unit, while distant speech frames are said to differ.

1.2. Related Work: Top-Down Approaches

Top-down approaches start by first finding pairs of longer word-like segments using unsupervised term discovery (UTD). This information provides constraints that can be use to find speech frame representations that are more stable within a given subword unit. The rationale is that while at the frame level the same speech sound can seem quite different between different speakers or even different realisations of the sound by the same speaker, patterns over a longer duration of time are easier to identify; this idea is illustrated in Jansen *et al.* [22].

The UTD systems used in this context are generally based on the segmental dynamic time warping (S-DTW) developed by Park and Glass [23]. S-DTW works by repeatedly performing DTW on two audio streams while constraining the maximum amount of warping allowed, each time changing the starting point of the DTW in both streams. This yields a set of alignments, from which the stretches of lowest average dissimilarity in each alignment can be extracted. Unfortunately, this approach is inherently $O(n^2)$ in time. To remedy this, Jansen and Van Durme [24] introduced an approximate version that uses binary approximations of the feature vectors to perform the calculations in $O(n \log n)$ time using sparse similarity matrices; this system also serves as the baseline for the second track of the Zero Resource Speech Challenge [13].

Jansen and Church [25] describe a method for finding subword units, assuming that clusters corresponding to words, each cluster containing multiple examples of that word in the form of audio, are given. For each word, an HMM is trained on all the corresponding examples, the number of states in the model being set to a number proportional to the average duration of the word. The states from each HMM are then collected and clustered based on the similarity of their distributions, forming clusters that hopefully correspond to subword units.

Jansen *et al.* [22] take somewhat of an inverse approach, starting by clustering the whole data on a frame level, with the assumption that each cluster will tend to correspond to some speaker- or context-dependent subword unit. They then look at pairs of word-like segments known to be of the same type and calculate how often clusters tend to co-occurr. The clusters are then partitioned so that clusters that co-occurr often are placed in the same partition.

Synnaeve *et al.* [26] introduce a neural network known referred to as the ABnet, based on siamese networks [31]. The network takes a pair of speech frames as input, and adjusts its parameters so that the outputs are collinear if the inputs are known to correspond to the same subword unit, and orthogonal otherwise, using a cosine-based loss function. Thiolliere *et al.* [27] made use of this approach in the Zero Resource Speech Challenge, also incorporating unsupervised term discovery so as to make the whole process unsupervised, yielding competitive results [19]. Zeghidour *et al.* [28] experiment with supplying the ABnet with scattering spectrum features instead of filter bank features, showing that with the right features, a shallow architecture may outperform a deep architecture, especially when the amount of available data is low.

Kamper *et al.* [29] use an autoencoder-like structure, where a neural network is trained to "reconstruct" a frame given another frame known to be of the same type. Renshaw *et al.* [30] used this architecture in the Zero Resource Speech Challenge, albeit with a deeper decoder.

1.3. This Work

make description more application agnostic?

Two of the most successful approaches so far are the clustering approach of Chen *et al.* [17] and the siamese network approach of Thiolliere *et al.* [27]. We pose the question of whether it is possible to combine the two approaches by first clustering the data in an unsupervised manner using a probabilistic model, and then improving the resulting posteriorgrams using speech fragment information. This way we are able to take advantage of both the whole unlabelled data set, and the smaller set of discovered fragments.

Many probabilistic models, such as Gaussian mixture models and hidden Markov models, have a concept of latent states or classes. We pose the problem of improving posteriorgrams from such a model as one of merging, or partitioning, these classes. By first training the model in a fully unsupervised manner, it learns classes that can generally be assumed to be highly speaker-specific. We can then use weak supervision to merge these classes, yielding representations that are more speaker invariant.

A partitioning of classes can be viewed as a surjection from the original set of classes to a class set of lower cardinality, but finding this surjection is a discrete problem which is difficult to optimise for. However, a benefit of posteriorgrams is that the probability of an output class can be described as a simple sum of the probabilities of the classes that map to the class in question. This means that the surjection can be approximated using a continuous linear model which can be optimised through standard gradient descent. A linear model also has the added benefit of being more interpretable than deep networks such as that of Thiolliere *et al.* [27]. While the approach of partitioning posteriorgrams is very reminiscent of Jansen *et al.* [22], the major difference is that in place of direct clustering of classes, we are instead trying to maximise the similarity/dissimilarity between pairs of speech fragments, which only indirectly results in a partitioning of the classes.

2. Method

The goal of our method is to merge acoustic clusters obtained by bottom-up unsupervised classification such that the resulting classes correspond more closely to phonemic units in the language.

We take as input a set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ of N pairs of M-dimensional posteriorgrams, i.e. (row) vectors of probabilities such that the probabilities sum to one. Additionally, we have a set of indicators $\{c_i\}_{i=1}^N$ such that c_i is 1 if $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ belong to the same class, and 0 otherwise. The posteriorgrams are taken to represent a distribution over M discrete "pseudo"-classes (e.g. allophones), where several pseudo-classes together describe a single "true" class (e.g. phonemes). Our goal is then to find a surjection that maps the M pseudo-classes to a smaller set of D classes, where we take the probability of a single output class to be the sum of the probabilities of the pseudo-classes that map to the class in question.

To simplify optimisation we relax the problem to one of instead finding a continuous linear mapping $f : [0,1]^M \rightarrow [0,1]^D$ from the original space to a lower-dimensional space, such that $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{y}_i)$ are close if $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ belong to the same true class, and distant otherwise. We consider each output probability to be a weighted combination of input probabilities: $f(\boldsymbol{x})_j = \sum_{i=1}^M x_i w_{ij}$, or in matrix notation:

$$f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W} \tag{1}$$

where $\boldsymbol{W} = (w_{ij}) \in \mathbb{R}^{M \times D}$. If the elements of $\boldsymbol{W}$ are constrained to only take on values in $\{0, 1\}$, and each row of $\boldsymbol{W}$ contains exactly one element with the value 1, the problem is reduced to finding an exact surjection.

Even in the relaxed version of the problem, we need to put certain constraints on $\boldsymbol{W}$ in order to ensure that the output $f(\boldsymbol{x})$ is a posteriorgram. First, we need to ensure that all outputs are positive. As the input $\boldsymbol{x}$ is a posteriorgram, meaning that all elements in $\boldsymbol{x}$ are positive, it clearly suffices to ensure that all elements in $\boldsymbol{W}$ are positive. Second, the output probabilities must sum to 1. This can be achieved by ensuring that the elements of each row of $\boldsymbol{W}$ sum to 1, as can be seen by:

$$f(\boldsymbol{x})\mathbf{1}_D = \boldsymbol{x}\boldsymbol{W}\mathbf{1}_D = \boldsymbol{x}\mathbf{1}_D = 1 \tag{2}$$

where $\mathbf{1}_D$ is a column vector of D ones.

In order to ensure that these constraints hold, we construct our model as follows:

$$\boldsymbol{V} \in \mathbb{R}^{M \times D} \tag{3}$$

$$\widetilde{\boldsymbol{W}} = |\boldsymbol{V}| \tag{4}$$

$$\boldsymbol{W} = \widetilde{\boldsymbol{W}} \oslash \left(\widetilde{\boldsymbol{W}}\mathbf{1}_D\mathbf{1}_D^T\right) \tag{5}$$

$$f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W} \tag{6}$$

where $|\cdot|$ denotes the element-wise absolute value, and $\oslash$ denotes element-wise division. This formulation makes it possible to optimise the model while ensuring that the constraints on $\boldsymbol{W}$ hold, by performing gradient descent with respect to $\boldsymbol{V}$. Note that the absolute value is almost everywhere differentiable, and the non-differentiability at 0 does not matter in practice.

To encourage the model to place points belonging to the same class close together in the output space, we consider the model as a siamese network. Conceptually this involves duplicating the model, creating two identical copies of the same network, with the parameters shared. We then feed one input each to both copies, and calculate the loss function using the corresponding outputs:

$$L(\boldsymbol{V};\boldsymbol{x},\boldsymbol{y},c) = \begin{cases} D_{\text{same}}(f(\boldsymbol{x};\boldsymbol{V}), f(\boldsymbol{y};\boldsymbol{V})) & \text{if } c = 1 \\ D_{\text{diff}}(f(\boldsymbol{x};\boldsymbol{V}), f(\boldsymbol{y};\boldsymbol{V})) & \text{if } c = 0 \end{cases} \tag{7}$$

where D_{same} and D_{diff} are the dissimilarity/similarity measures for pairs belonging to the same class, and pairs belonging to different classes, respectively. The loss function over a minibatch B is given by the average

$$\frac{1}{|B|}\sum_{i\in B} L(\boldsymbol{V};\boldsymbol{x}_i,\boldsymbol{y}_i,c_i) \tag{8}$$

which is minimised with respect to $\boldsymbol{V}$.

2.1. Loss function

As the output of the model is a probability distribution, it makes intuitive sense to use a statistical divergence as a measure of similarity. Perhaps the most well-known divergence is the Kullback-Leibler (KL) divergence, defined as:

$$\text{KL}(\boldsymbol{x}||\boldsymbol{y}) = \sum_i x_i \log_2 \frac{x_i}{y_i}, \tag{9}$$

where we take $0 \log_2 0$ to be 0. The KL divergence is always positive, and is 0 only if $\boldsymbol{x} = \boldsymbol{y}$. However, it is unbounded, and undefined if there is an i such that $y_i = 0$ but $x_i \neq 0$. As such, trying to maximise the dissimilarity between two distributions with respect to the KL divergence is an ill-posed problem, as this will force the divergence to tend towards infinity.

A better choice is the Jensen-Shannon (JS) divergence, defined as

$$\text{JS}(\boldsymbol{x}||\boldsymbol{y}) = \frac{1}{2}\text{KL}(\boldsymbol{x}||\boldsymbol{m}) + \frac{1}{2}\text{KL}(\boldsymbol{y}||\boldsymbol{m}) \tag{10}$$

where $\boldsymbol{m} = (\boldsymbol{x}+\boldsymbol{y})/2$. The JS divergence is always defined, and is bounded between 0 (for identical distributions) and 1 (for distributions with disjoint support), assuming that the base 2 logarithm is used. Additionally, the square root of the JS divergence is a metric satisfying the triangle inequality [32]; here we make use of this fact, in the hope that the metric properties will result in a more well-behaved loss function.

Thus, we define the loss function as

$$L_{\text{JS}}(\boldsymbol{V};\boldsymbol{x},\boldsymbol{y},c) = \begin{cases} \sqrt{\text{JS}(f(\boldsymbol{x};\boldsymbol{V})||f(\boldsymbol{y};\boldsymbol{V}))} & \text{if } c = 1 \\ 1 - \sqrt{\text{JS}(f(\boldsymbol{x};\boldsymbol{V})||f(\boldsymbol{y};\boldsymbol{V}))} & \text{if } c = 0, \end{cases} \tag{11}$$

thereby minimising the root JS divergence between pairs belonging to the same class, and maximising the divergence between pairs belonging to different classes[1].

2.2. Entropy penalty

To make the output of the model interpretable, it is desirable to ensure that for a given input, only one output unit is active. This can be done by introducing an entropy penalty, which attempts to minimise the spread of the probability mass. The entropy of a probability vector $\boldsymbol{x} = (x_1, \ldots, x_D)$ is defined as

$$H(\boldsymbol{x}) = -\sum_{i=1}^{D} x_i \log_2 x_i. \tag{12}$$

However, this definition is sensitive to the value of D; for instance, the entropy of a uniform distribution vector is $\log_2 D$.

As we may wish to vary the number of outputs of the model, it is of interest for the entropy penalty to be invariant to the number of outputs. We therefore introduce the normalised entropy, defined as

$$\hat{H}(\boldsymbol{x}) = \frac{1}{\log_2 D} H(\boldsymbol{x}). \tag{13}$$

The normalised entropy is always between 0 (for degenerate distributions) and 1 (for uniform distributions).

The entropy penalty implicitly encourages sparsity in $\boldsymbol{W}$, as the only way to avoid spreading the probability mass across several outputs is for each row of $\boldsymbol{W}$ to only contain a single element close to 1. It is thus through this penalty that we enforce the model to find an approximate surjection. In summary, our final loss function over a minibatch B is as follows:

$$\begin{aligned} L(\boldsymbol{V};B) = \frac{1}{|B|}\sum_{i\in B} L_{\text{JS}}(\boldsymbol{V};\boldsymbol{x}_i,\boldsymbol{y}_i,c_i) + \\ + \frac{\lambda}{2|B|}\sum_{i\in B}\left(\hat{H}\left(f(\boldsymbol{x}_i;\boldsymbol{V})\right) + \hat{H}\left(f(\boldsymbol{y}_i;\boldsymbol{V})\right)\right) \end{aligned} \tag{14}$$

where λ is a hyperparameter.

3. Experiments

3.1. Data

To test our method we use the data from the 2015 Zero Resource Speech Challenge. The Challenge makes use of two corpora: The Buckeye corpus of conersational English [33] and the NCHLT speech corpus of read Xitsonga [34]. For the challenge only a subset of the data is used, consisting of 12 speakers for a total of 5 hours of data for the Buckeye corpus, and 24 speakers for a total of 2.5 hours of data for the NCHLT Xitsonga corpus. Additionally provided is voice activity information indicating segments containing clean speech, as well as labels indicating the identity of the speaker.

MFCCs features were extracted from the data using a frame window length 25 ms which was shifted 10 ms for each frame, an FFT resolution of 512 frequency steps, and 40 mel-spaced triangular filter banks. 13 coefficients with both delta and delta-delta features were used. The MFCCs corresponding to segments with voice activity were clustered using an implementation of a Gaussian mixture model (GMM) provided by scikit-learn [35]. The GMM was trained using the expectation maximisation algorithm, using $M = 1024$ Gaussians with diagonal covariance matrices, for a maximum of 200 iterations. After training the posteriorgram for the nth frame is constructed as $\boldsymbol{p}_n = (p_n^1, p_n^2, \ldots, p_n^{1024})$ where $p_n^i = p(z_i \mid \boldsymbol{x}_n)$ is the posterior probability of the ith class given the nth frame.

z_i^n or z_i be
ter?

3.2. Unsupervised term discovery

Pairs of similar speech fragments were discovered using the system developed by Jansen and Van Durme [24], which serves as a baseline for the second track of the Zero Resource Speech Challenge. The system works by calculating the approximate cosine

[1]For identical or near-identical $\boldsymbol{x}$ and $\boldsymbol{y}$, the JS divergence may become negative due to rounding errors caused by limited floating point precision; this can be counteracted by adding a small constant value before taking the square root.

similarity between pairs of frames of two input audio segments, based on discretised random projections of PLP features. For efficiency only frames found using an approximate nearest neighbour search are compared, yielding a sparse similarity matrix. Stretches of similar frames are then found by searching for diagonals in the similarity matrix, which which are then aligned using dynamic time warping (DTW). Pairs of segments with a DTW score above a certain threshold are kept and clustered based on pairwise DTW similarity, resulting in a set of clusters of speech segments, or fragments, thought to be of the same class (e.g. word).

This process yielded 6512 fragments and 3149 clusters for the Buckeye corpus, and 3582 fragments and 1782 clusters for the NCHLT Xitsonga corpus[2]. For each cluster every possible pair of fragments was extracted from the collection of posteriorgrams retrieved from the GMM and aligned using DTW, yielding pairs of speech frames belonging to the same class. Let K be the total number of pairs of fragments aligned. To generate a set of pairs of frames belonging to different classes, K fragments were sampled uniformly from the full collection of fragments. For each such fragments, another fragment was sampled uniformly from the fragments belonging to a different cluster. When sampling fragments belonging to a different cluster, the sampling was performed using only either fragments spoken by the same speaker, or fragments spoken by a different speaker, with a probability corresponding to the ratio of same-speaker to different-speaker pairs among the same-class fragment pairs. The different-class fragment pairs were aligned by simply truncating the longer fragment.

70% of the same-class and different-class fragment pairs were used for training, with the remaining pairs used for validation to determine when to interrupt the training of the models.

3.3. Model implementation

mention server specification? cpu/ram/gpu

We used $D = 64$ outputs for all models. The models were trained using AdaMax [36] with the recommended default parameters $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All frames used for training were shuffled once at the start of training, and a minibatch size of 1000 frames was used. The models were trained until no improvement had been observed on a held-out validation set for 15 epochs, where one epoch is defined as one complete scan over the training data.

All network models were implemented in Python 3.5 using Theano [37] for automatic differentiation and GPU acceleration, librosa [38] for feature extraction, scikit-learn [35] for various utilities, and numba [39] for accelerating various code, in particular dynamic time warping.

3.4. Tuning the entropy penalty

The entropy penalty λ is a free parameter, which is data dependent and must be manually specified. Ideally, λ should be such that the entropy is reduced to a satisfactory degree, without sacrificing the Jensen-Shannon loss. As both the normalised entropy loss and the Jensen-Shannon loss are bounded between 0 and 1, one might expect the optimal value of λ to be in the vicinity of 1. Section 4 reports on the optimisation of λ with respect to the validation error both for the Buckeye and NCHLT Xitsonga corpora.

[2]The cluster files used for this work were generously provided by Roland Thiollière and Aren Jansen.

3.5. Balancing same-class and different-class losses

When enforcing low entropy in the output distribution, the resulting weight matrix becomes sparse. For instance, after training the model with $\lambda = 0.1$, and inspecting the row-normalised matrix $\boldsymbol{W}$, we find that the largest element on each row is close to 1: on average across the 1024 rows 0.98 for English and 0.92 for Xitsonga. We can thus inspect $\boldsymbol{W}$ to see how many of the 64 outputs are actually being used by the model. We take the sum over each column of $\boldsymbol{W}$. This sum describes roughly how many inputs are mapped to each output. We find that for both English and Xitsonga, this sum is above 0.5 for only a minority of outputs: 11 outputs for English, and 10 outputs for Xitsonga. For English, where $\boldsymbol{W}$ is particularly sparse, none of the other 53 sums even reach 0.05.

Thus, it seems that the entropy penalty naturally encourages the model to make use of only a subset of the outputs. However, the actual number of outputs used is not realistic in terms of how many phonemes one would expect to find in a language; it seems that the same-class loss is forcing too many input classes to merge. To solve this, we restate the Jensen-Shannon loss function, allowing us to specify how much relative weight to give to the same-class and different-class losses. Let $B_1 = \{i \in B : c_i = 1\}$ be the subset of same-class frame pairs in the current minibatch, and $B_0 = \{i \in B : c_i = 0\}$ the subset of different-class frame pairs. We then restate the loss as

$$\frac{1}{(\alpha+1)|B_1|}\sum_{i\in B_1} L_{\text{same}}(\boldsymbol{V};\boldsymbol{x}_i,\boldsymbol{y}_i)+\frac{\alpha}{(\alpha+1)|B_0|}\sum_{i\in B_0} L_{\text{diff}}(\boldsymbol{V};\boldsymbol{x}_i,\boldsymbol{y}_i), \tag{15}$$

where L_{same} and L_{diff} are the same-class and different-class losses defined in equation (11). α is a hyperparameter specifying how much more to weight the different-class loss over the same-class loss.

α needs to be carefully tuned: A too small α will cause too many input classes to merge, including classes that correspond to completely different phonemes, while a too large α will cause input classes that do correspond to the same phoneme to fail to merge. In order to find a good value for α, without making use of the gold transcription or prior knowledge of the number of phonemes present in the languages in question, we make use of the fragment clusters discovered by the unsupervised term discovery system. The intuition is that the goal of our model is to push apart different clusters, while keeping fragments within a cluster as similar as possible. To measure the success of our model, then, we can make use of a cluster separation measure.

Here we use the silhouette [40], which makes use of the average similarity between a sample and every other sample in the same cluster, and between a sample and every sample in the most similar other cluster. The silhouette ranges from -1 to 1, with a value close to 1 indicating that the clusters are well separated. Models were trained for $\alpha \in \{1, 1.5, 2, 2.5, 3, 3.5, 4\}$, with an entropy penalty of $\lambda = 0.1$. The silhouette was then calculated on a subset of 1000 of the fragment clusters, using the output of the trained models to represent the frames of the fragments. The similarity between to fragments was calculated as the DTW score using the symmetrise Kullback-Leibler divergence as a similarity measure between individual frames.

is the below paragraph uninteresting? a bit too model-specific?

To easily get an estimate of the number of outputs used by the model, we also define the “spread” of the model as follows.

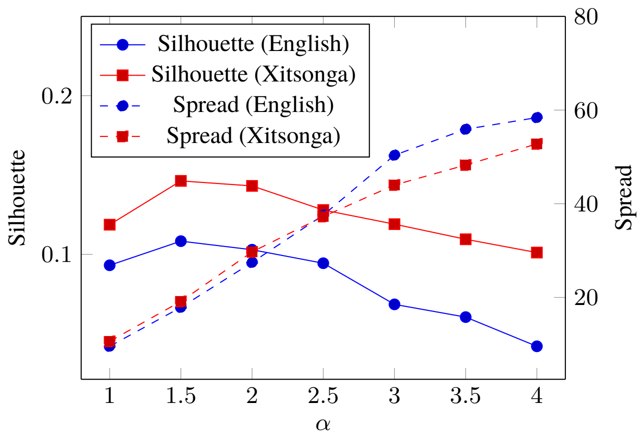

Figure 1: *Silhouette and spread for different weightings of the same-class and different-class losses.*

We take the average of the jth column:

$$q_j = \frac{1}{M} \sum_{i=1}^{M} w_{ij}. \tag{16}$$

This represents the average mapping to the ith output. As $\boldsymbol{W}$ is row-normalised, the elements of $Q = (q_1, q_2, \ldots, q_D)$ sum to 1, and we can thus treat Q as describing a probability distribution. A uniform distribution means that each output has the same number of inputs mapped to it. Now consider the case where there are K outputs such that the same number of inputs maps to each output, while no inputs map to any other outputs. The normalised entropy of Q is then given by

$$\hat{H}(Q) = -\frac{1}{\log_2 D} \sum_{i=1}^{K} \frac{1}{K} \log_2 \frac{1}{K} = \frac{\log_2 K}{\log_2 D}. \tag{17}$$

Solving for K we have

$$K = D^{\hat{H}(Q)}, \tag{18}$$

which is an approximation of the number of outputs used by the model, which we define as the spread. A value of K close to D is an indicator that all the outputs are being used equally, suggesting that it may be a good idea to increase the number of outputs.

Figure 1 shows the silhouette and spread for different values of α. As one might expect, more emphasis on the different-class loss results in a higher spread, i.e. a larger number of output classes. The optimal value of α seems to be around 1.5 for both data sets, we use this value of α going forward.

3.6. Discretising the model

As the resulting model is sparse, we can retrieve an exact surjection by discretising the model. We do this by for each row in $\boldsymbol{W}$ setting the largest element to 1 and the remaining elements to 0. Using the discretised model as a base, we additionally experiment with discretising the output distribution by setting the largest output to 1 and the rest to 0; this can be thought of as taking the argmax of the output distribution.

3.7. Comparison with deep models

To get an idea of how the JS loss performs in general, we build a deep network with two hidden layers of 500 sigmoid units each, with 64 softmax outputs. The network is trained using the non-rebalanced JS loss. As softmax outputs are naturally sparse, we do not enforce any entropy penalty. For comparison we train the same architecture, albeit with sigmoid outputs instead, using the coscos^2 loss of Synnaeve *et al.* [26]. This is the architecture used by Thiolliere *et al.* [27] in the 2015 Zero Resource Speech Challenge.

As input to both networks we use the log-scale outputs of 40 mel-scaled filter banks. All other relevant parameters are the same as for the MFCCs calculated in **??**. The filter bank outputs are normalised over the whole data set to have zero mean and unit variance for all dimensions. Each frame is fed to the network with a context of 3 frames on both sides, for a total of 280 values used as input to the network. All fragments are DTW aligned and sampled as in section 3.2.

4. Results

4.1. Tuning the entropy penalty

We train models using $\lambda \in \{0, 0.05, 0.1, \ldots, 0.95\}$ for both the Buckeye and NCHLT Xitsonga corpora. The final validation errors for each model are reported in figure 2. For both corpora, the entropy drops quickly even for small λ, suggesting that the entropy is relatively easy to optimise for. As the entropy penalty is increased, the entropy itself does not decrease; however, the different-class JS loss decreases at the expense of the same-class JS loss. For future experiments, a penalty of $\lambda = 0.1$ is used.

why?

4.2. ABX evaluation

We evaluate the models discussed on the minimal-pair ABX task [41]. In the task we are presented with three speech fragments A, B and X, where A and B form minimal pairs, i.e. they only differ by a single phoneme. The task is to decide which of either A or B belongs to the same category as X. This is done by DTW-aligning A and B with X with respect to some underlying frame-based metric. The fragment closest to X according to the DTW score is chosen. The task takes two forms: within-speaker discriminability, where all fragments belong to the same speaker, and across-speaker discriminability, where A and B belong to one speaker while X belongs to another.

The models are evaluated using a evaluation toolkit provided for the Zero Resource Speech Challenge. The results are shown in table 1, along with the silhouette for each model. The frame-based metric is chosen as the symmetrised Kullback-Leibler divergence (with the model output normalised as necessary), with the exception of the model with discretised output, which uses the cosine distance, which for one-hot vectors amounts to a distance of 0 for identical and 1 for non-identical vectors.

We can see that in general, the silhouette seems to be indicative of the relative performance on the ABX task. Our suspicion that the number of outputs used were too few when using the Jensen-Shannon loss as originally stated is validated, with the rebalanced loss performing better for both English and Xitsonga. The performance of the model with discretised weights further suggests that the basic premise of improving posteriorgrams by partitioning is a sound one.

The deep model performs poorly when trained with the Jensen-Shannon loss, despite the same architecture performing well when trained with the coscos^2 loss. Inspecting the average output of the deep model over the English data set, we found that only 6 outputs are actually used by the model. This suggests that the JS loss is more sensitive than the coscos^2 loss when it comes

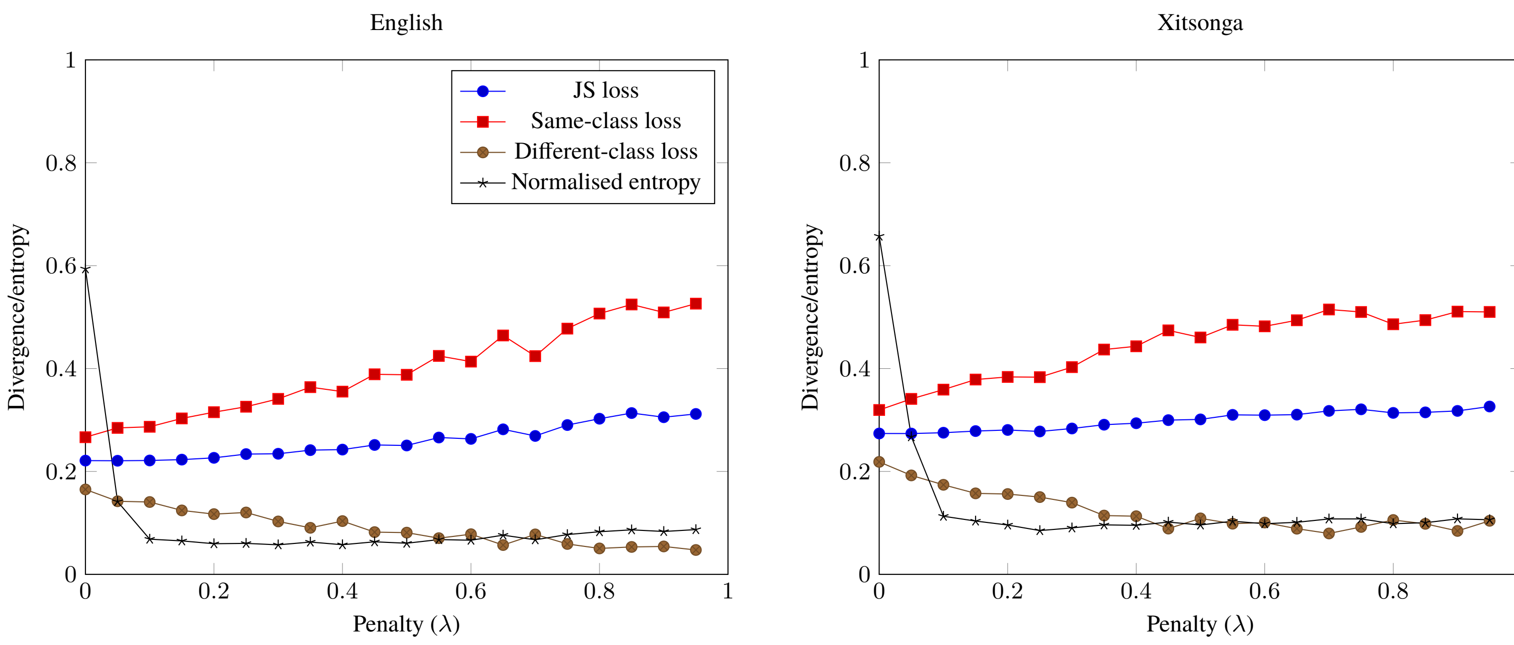

Figure 2: *Effect of varying the entropy penalty for the English (left) and Xitsonga (right) corpora. The average entropy of the output distribution over the validation samples is shown along with the (root) Jensen-Shannon loss: Both the combined JS loss that is optimised for, and separately for same-class and different-class frame pairs.*

	English			Xitsonga		
Model	Silhouette	Within	Across	Silhouette	Within	Across
GMM posteriors	0.008	12.313	23.841	0.066	11.434	23.181
Non-rebalanced	0.089	14.195	21.369	0.111	16.477	25.551
Rebalanced	0.108	12.770	19.831	0.146	13.990	23.202
Discretised $\boldsymbol{W}$	0.124	12.013	19.261	0.170	12.702	21.888
Discretised output	0.010	16.513	24.565	0.014	19.404	29.150
Deep JS	-0.370	22.376	28.233	-0.320	18.190	24.759
Deep coscos^2	0.187	12.294	19.561	0.174	11.934	19.052

Table 1: *Within-speaker and across-speaker ABX scores as well as the silhouette for the different models for both the English and Xitsonga data sets. GMM posteriors is the posteriorgrams extracted from the 1024-component Gaussian mixture model; non-rebalanced is the original loss presented in equation* (14)*; rebalanced is the alternative loss presented in equation* (15) *with* $\alpha = 1.5$*; discretised* $\boldsymbol{W}$ *and discretised output are the models presented in section 3.6; and the deep models are those presented in section 3.7. The silhouette is calculated on a subset of 1000 clusters for each language. All shallow models are trained with an entropy penalty of* $\lambda = 0.1$.

to balancing the same-class and different-class losses.

5. Discussion

We have seen that the model is indeed able to improve on the input posteriors. In particular, the model improves the across-speaker performance, with little to no degradation of the within-speaker performance. However, the Jensen-Shannon loss function used is shown to perform worse in general than coscos^2, possibly as a result of being more sensitive to the balancing of the same-class and different-class losses. This can be explained by the fact that the Jensen-Shannon divergence is not directly interpretable—for instance, it is not clear that a same-class loss of 0.1 is as good as a different-class loss of 0.9. On the other hand, the cosine difference is more readily (geometrically) interpretable.

However, the model itself does come with a number of advantages over deep models. The linear nature of the model means that the number of parameters is small, making the model fast and easy to train, and robust against overfitting. This is especially the case when imposing the entropy penalty, which can be seen as restricting the capacity of the model. The sparsity of the model additionally makes it more interpretable, providing insight into how exactly the input classes are mapped to the output. The model is also readily convertible into an exact surjection, resulting in a proper partition of the input classes.

Another feature of the model is that it can take any kind of probability distribution as input, with the only requirement being that the underlying true classes are disentangled in the input. This makes it possible to use any kind of probabilistic model that admits a discrete posterior distribution over classes or states, including e.g. Gaussian mixture models or hidden Markov models. The resulting posteriorgrams can then be improved further by using the model to find a mapping to a smaller number of classes.

One important question is how sensitive the model is to the dimensionality of the input. As the model requires evidence in terms of same-class or different-class pairs to know where to map each input class, a lack of evidence can result in classes being incorrectly merged (or unmerged, conversely). As the input size grows, the amount of evidence required grows as well. As such, it is advisable to choose an input size that reflects the amount of evidence available. This may explain the poor performance of the model on the Xitsonga data set, as far fewer speech fragments were found for Xitsonga than for English.

6. Conclusions

A linear model for approximate partitioning of posteriorgrams was introduced. Using posteriorgrams from a Gaussian mixture model trained on MFCCs as a proof of concept, the model was shown to improve the across-speaker performance, with competitive results for the English data set. While the better-performing versions of the model depends on two hyperparameters, the hyperparameter search is alleviated somewhat by ease of training the linear model. Additionally, the entropy penalty was shown to be easy to optimise for, allowing a small value for the corresponding hyperparameter. The silhouette cluster separation measure was shown to be indicative of ABX performance, enabling hyperparameter search without making use of the gold transcription.

The resulting model is sparse and easily interpretable. However, the Jensen-Shannon loss function used is sensitive to the balancing of the same-class and different-class losses, making it particularly unsuitable for deep architectures.

6.1. Future work

A natural extension of this work is to use different probabilistic models to generate the posteriorgrams, and see how this affects the performance of the model. For instance, would the model be able to improve on the posteriorgrams generated by the model of Chen *et al.* [17]? Of interest are also models that directly model time dependencies, such as hidden Markov models.

The model as presented here can be seen as a kind of radial basis function (RBF) network, where the RBF units (i.e. the Gaussian mixture model) are trained on the complete data set, while the output weights are trained using gradient descent on the fragment pair data. As such it might be interesting to see whether joint training of both the input clusters and the linear mapping by treating the model as a single RBF network would lead to any improvements.

Finally, as we have seen the Jensen-Shannon loss needs to reweighted in order to properly balance the same-class and different-class losses. It is thus desirable to find an alternative loss function suitable for probability distributions, for which the losses are naturally more balanced.

7. Acknowledgements

The GeForce GTX TITAN used for this research were donated by the NVIDIA Corporation. This research was partly supported by the CHIST-ERA project IGLU.

8. References

[1] C.-H. Lee, F. Soong, and B.-H. Juang, “A segment model based approach to speech recognition,” in *Proc. of IEEE ICASSP*, vol. 1, 1988, pp. 501–504.

[2] T. Svendsen, K. Paliwal, E. Harborg, and P. Husoy, “An improved sub-word based speech recognizer,” in *Proc. of IEEE ICASSP*, vol. 1, 1989, pp. 108–111.

[3] M. Bacchiani, M. Ostendorf, Y. Sagisaka, and K. Paliwal, “Design of a speech recognition system based on acoustically derived segmental units,” in *Proc. of IEEE ICASSP*, vol. 1, 1996, pp. 443–446.

[4] M. Huijbregts, M. McLaren, and D. van Leeuwen, “Unsupervised acoustic sub-word unit detection for query-by-example spoken term detection,” in *Proc. of Interspeech*, 2011.

[5] P. O’Grady, “Discovering speech phones using convolutive non-negative matrix factorisation with a sparseness constraint,” *Neurocomputing*, vol. 72, no. 1-3, pp. 88–101, 2008.

[6] O. Räsänen, “A computational model of word segmentation from continuous speech using transitional probabilities of atomic acoustic events,” *Cognition*, vol. 120, no. 2, pp. 149–176, 2011.

[7] A. S. Park and J. R. Glass, “Unsupervised pattern discovery in speech,” *IEEE Trans. Audio, Speech and Lang. Proc.*, vol. 16, no. 1, 2008.

[8] G. Aimetti, R. K. Moore, and L. ten Bosch, “Discovering an optimal set of minimally contrasting acoustic speech units: A point of focus for whole-word pattern matching,” in *Proc. of Interspeech*, 2010, pp. 310–313.

[9] V. Stouten, K. Demuynck, and H. van Hamme, “Discovering phone patterns in spoken utterances by non-negative matrix factorization,” *IEEE Signal Processing Lett.*, vol. 15, pp. 131–134, 2008.

[10] J. Driesen, L. ten Bosch, and H. van Hamme, “Adaptive non-negative matrix factorization in a computational model of language acquisition,” in *Proc. of Interspeech*, 2009.

[11] N. Vanhainen and G. Salvi, "Word discovery with beta process factor analysis," in *Proc. of Interspeech*, Portland, OR, USA, Sep. 2012.

[12] ——, "Pattern discovery in continuous speech using block diagonal infinite hmm," in *Proc. of IEEE ICASSP*, 2014.

[13] M. Versteegh, R. Thiolliere, T. Schatz, *et al.*, "The zero resource speech challenge 2015," in *Proc. of INTERSPEECH*, 2015.

[14] B. Varadarajan, S. Khudanpur, and E. Dupoux, "Unsupervised learning of acoustic sub-word units," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, Association for Computational Linguistics, 2008, pp. 165–168.

[15] C.-y. Lee and J. Glass, "A nonparametric bayesian approach to acoustic model discovery," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, Association for Computational Linguistics, 2012, pp. 40–49.

[16] M.-h. Siu, H. Gish, A. Chan, *et al.*, "Unsupervised training of an hmm-based self-organizing unit recognizer with applications to topic classification and keyword discovery," *Computer Speech & Language*, vol. 28, no. 1, pp. 210–223, 2014.

[17] H. Chen, C.-C. Leung, L. Xie, *et al.*, "Parallel inference of dirichlet process gaussian mixture models for unsupervised acoustic modeling: A feasibility study," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[18] Y. Zhang and J. R. Glass, "Towards multi-speaker unsupervised speech pattern discovery," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, IEEE, 2010, pp. 4366–4369.

[19] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015: Proposed approaches and results," *Procedia Computer Science*, vol. 81, pp. 67–72, 2016.

[20] M. Heck, S. Sakti, and S. Nakamura, "Unsupervised linear discriminant analysis for supporting dpgmm clustering in the zero resource scenario," *Procedia Computer Science*, vol. 81, pp. 73–79, 2016.

[21] G. Synnaeve and E. Dupoux, "A temporal coherence loss function for learning unsupervised acoustic embeddings," *Procedia Computer Science*, vol. 81, pp. 95–100, 2016.

[22] A. Jansen, S. Thomas, and H. Hermansky, "Weak top-down constraints for unsupervised acoustic model training.," in *ICASSP*, 2013, pp. 8091–8095.

[23] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 186–197, 2008.

[24] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, IEEE, 2011, pp. 401–406.

[25] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models.," in *INTERSPEECH*, 2011, pp. 1693–1692.

[26] G. Synnaeve, T. Schatz, and E. Dupoux, "Phonetics embedding learning with side information," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*, IEEE, 2014, pp. 106–111.

[27] R. Thiolliere, E. Dunbar, G. Synnaeve, *et al.*, "A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling," in *Proc. Interspeech*, 2015.

[28] N. Zeghidour, G. Synnaeve, M. Versteegh, and E. Dupoux, "A deep scattering spectrum — deep siamese network pipeline for unsupervised acoustic modeling," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 4965–4969.

[29] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 5818–5822.

[30] D. Renshaw, H. Kamper, A. Jansen, and S. Goldwater, "A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge," in *Proc. Interspeech*, 2015.

[31] J. Bromley, I. Guyon, Y. LeCun, *et al.*, "Signature verification using a "siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, 1994, pp. 737–744.

[32] D. M. Endres and J. E. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information theory*, vol. 49, no. 7, pp. 1858–1860, 2003.

[33] M. Pitt, L. Dilley, K. Johnson, *et al.*, *Buckeye corpus of conversational speech (2nd release)*, Online, Columbus, OH: Department of Psychology, Ohio State University (Distributor), 2007. [Online]. Available: `www.buckeyecorpus.osu.edu`.

[34] E. Barnard, M. H. Davel, C. J. van Heerden, *et al.*, "The nchlt speech corpus of the south african languages.," *SLTU*, vol. 2014, pp. 194–200, 2014.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: `http://scikit-learn.org/`.

[36] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv preprint arXiv:1412.6980*, 2014.

[37] R. Al-Rfou, G. Alain, A. Almahairi, *et al.*, "Theano: a Python framework for fast computation of mathematical expressions," *ArXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: `http://arxiv.org/abs/1605.02688`.

[38] B. McFee, M. McVicar, O. Nieto, *et al.*, *Librosa 0.5.0*, Feb. 2017. [Online]. Available: `https://github.com/librosa/librosa`.

[39] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ACM, 2015, p. 7.

[40] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[41] T. Schatz, V. Peddinti, F. Bach, *et al.*, "Evaluating speech features with the minimal-pair abx task: Analysis of the classical mfc/plp pipeline," in *INTERSPEECH 2013: 14th Annual Conference of the International Speech Communication Association*, 2013, pp. 1–5.