UNIVERSITY OF CAMBRIDGE
Department of Engineering

Generative Kernels and Score-Spaces for Classification of Speech: Progress Report

R. C. van Dalen rcv25@cam.ac.uk
J. Yang jy308@cam.ac.uk
M. J. F. Gales mjfg@eng.cam.ac.uk
A. Ragni ar527@cam.ac.uk
S. X. Zhang sxz20@cam.ac.uk

Technical Report CUED/F-INFENG/TR.676

January 2012

This is the first progress report for EPSRC Project EP/I006583/1 (Generative Kernels and Score Spaces for Classification of Speech) within the Global Uncertainties Programme. This project combines the current generative models developed in the speech community with discriminative classifiers. An important aspect of the approach is that the generative models are used to define a score-space that can be used as features by the discriminative classifiers. This work reports progress on HMM compensation for noise, efficient computation of generative scores, and various forms of classifiers.

Contents

1 Introduction

This is the first progress report for EPSRC Project EP/I006583/1 (Generative Kernels and Score Spaces for Classification of Speech) within the Global Uncertainties Programme.

The aim of this project is to significantly improve the performance of automatic speech recognition systems across a wide-range of environments, speakers and speaking styles. The performance of state-of-the-art speech recognition systems is often acceptable under fairly controlled conditions and where the levels of background noise are low. However for many realistic situations there can be high levels of background noise, for example in-car navigation, or widely ranging channel conditions and speaking styles, such as observed on YouTube-style data. This fragility of speech recognition systems is one of the primary reasons that speech recognition systems are not more widely deployed and used. It limits the possible domains in which speech can be reliably used, and increases the cost of developing applications as systems must be tuned to limit the impact of this fragility. This includes collecting domain-specific data and significant tuning of the application itself.

The vast majority of research for speech recognition has concentrated on improving the performance of systems based on hidden Markov models (HMMs). HMMs are an example of a generative model and are currently used in state-of-the-art speech recognition systems. A wide number of approaches have been developed to improve the performance of these systems under changes of speaker and noise. Despite these approaches, systems are not sufficiently robust to allow speech recognition systems to achieve the level of impact that the naturalness of the interface should allow.

One of the major problems with applying traditional classifiers, such as support vector machines, to speech recognition is that data sequences of variable length must be classified. This project combines the current generative models developed in the speech recognition community with discriminative classifiers used both in speech processing and in machine learning. Figure 1 gives a schematic overview of the approach that this project takes. The shaded part of the diagram indicates the generative model of a state-of-the-art speech recogniser. In this project, the generative models are used to define a score-space. These scores then form features for the discriminative classifiers. This

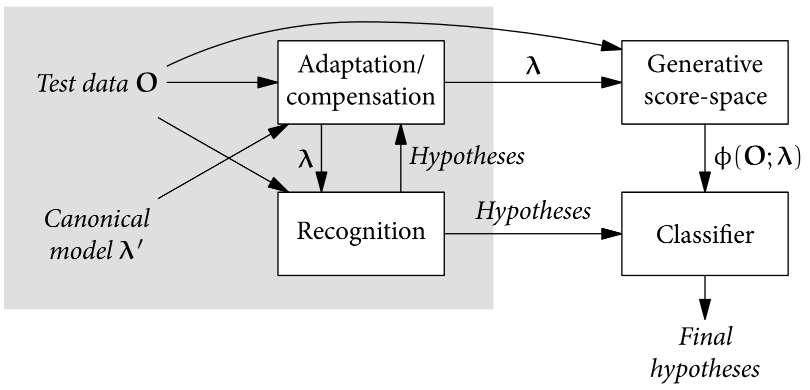


Figure 1 *Flow diagram of the project plan. The shaded region encompasses the components of a state-of-the-art speech recogniser.*

approach has a number of advantages. It is possible to use current state-of-the-art adaptation and robustness approaches to compensate the acoustic models for particular speakers and noise conditions. As well as enabling any advances in these approaches to be incorporated into the scheme, it is not necessary to develop approaches that adapt the discriminative classifiers to speakers, style and noise. Using generative models also allows the dynamic aspects of speech data to be handled without having to alter the discriminative classifier. The final advantage is the nature of the score-space obtained from the generative model. Generative models such as HMMs have underlying conditional independence assumptions that, whilst enabling them to efficiently represent data sequences, do not accurately represent the dependencies in data sequences such as speech. The score-space associated with a generative model does not have the same conditional independence assumptions as the original generative model. This allows more accurate modelling of the dependencies in the speech data.

This report will report on improvements in three of the components in figure 1: compensation, score-space computation, and classifiers. It will also report on some work related to the project. Section 2 will discuss compensation. It presents a novel variational method for compensating HMM speech recognisers for noise, which does not assume that the resulting distribution is Gaussian (published as van Dalen and Gales 2011*a*). Section 3 will discuss score-space computation. It presents a method to compute scores from generative models for all segmentations in amortised constant time (published as van Dalen *et al.* 2012). Section 4 will discuss a recognition with various forms of classifiers. It will report on log-linear models, both for acoustic modelling (work related to the project and published as Ragni and Gales 2011*b*;*a*) and language modelling (unpublished). A structured SVM is another type of classifier whose use will be discussed (work related to the project and published as Zhang and Gales 2011*b*;*a*). An initial discussion will be given of a Bayesian non-parametric approach to classification using infinite Gaussian mixture models (unpublished). Section 5 will report results of initial experiments on data from YouTube supplied by Google (unpublished). Finally, section 6 will summarise the findings so far and discuss research directions for the next two years.

2 Model compensation

This section describes a variational approach to compensating speech recognisers based on hidden Markov model for noise. It is a summary of work published as van Dalen and Gales (2011*a*).

Model compensation techniques modify the HMM's state-conditional distributions so they model the speech in the target environment. Because the interaction between speech and noise is non-linear, the corrupted-speech distribution has no closed form. In particular, even if the speech and noise distributions are Gaussian, the corrupted speech is not. However, the majority of schemes assume the corrupted speech Gaussian-distributed. Additionally, despite correlation changes, the covariance matrices are normally diagonalised. Thus, improvements from different ways of computing the same form of distribution are limited (see e.g. Li *et al.* (2010)).

Two approaches that remove the Gaussian assumption have been proposed. The "Algonquin" algorithm (Kristjansson 2002) still uses a Gaussian approximation, but tuned for each clean speech component and observation vector. Thus, the effective distribution over observation vectors is non-Gaussian. Another approach (van Dalen and

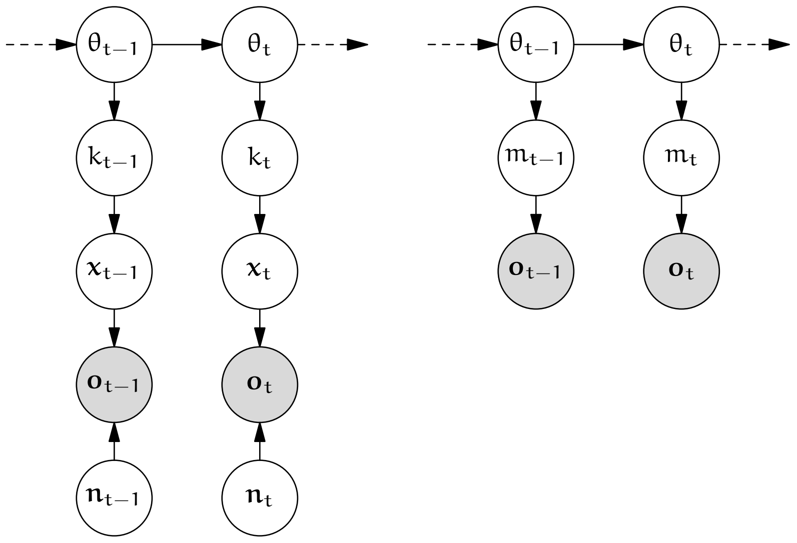


Figure 2 *The noise-corrupted speech* p *(left), with components* k, *speech* $\mathbf{x}$, *noise* $\mathbf{n}$, *corrupted speech* $\mathbf{o}$; *its approximation* q *(right) with components* m.

Gales 2010) uses a non-parametric distribution, by approximating the integral in the corrupted-speech likelihood expression with a sampling scheme. However, both these approaches require extensive computation for each incoming feature vector. Thus, for most situations they are impractical.

The aim of this section is to find forms of fast likelihood computation that approximate these non-Gaussian distributions. Here, the framework of *predictive methods* is applied to this problem. Predictive methods approximate a complicated model with a simpler form by minimising the KL divergence between them (van Dalen *et al.* 2009). How this applies to noise-robust speech recognition can be seen in Fig. 2. The left graphical model represents noise-corrupted speech. The speech is governed by a hidden Markov model, with states θ_t. Associated with each state is a mixture of Gaussians with component indicator k_t and Gaussian component distributions generating speech vectors $\mathbf{x}_t$. The independently and identically distributed noise $\mathbf{n}_t$ corrupts the speech, resulting in noise-corrupted speech $\mathbf{o}_t$.

The right graphical model in Fig. 2 approximates the corrupted speech distribution. The Markov model $p(\theta_{t-1}|\theta_t)$, which governs the clean speech, is assumed unchanged. Many compensation methods map each component k of the predicted distribution onto one component m of the approximation and marginalise out only $\mathbf{x}$ and $\mathbf{n}$. This is not a problem if the marginalisation is exact; however, usually each component is approximated with a Gaussian (Acero *et al.* 2000; Li *et al.* 2010; Seltzer *et al.* 2010; van Dalen and Gales 2011*c*). This *matched-pair approximation* to the KL divergence neglects the potential for a mixture of Gaussians to represent a mixture of non-Gaussian distributions. To approximate a whole state-conditional distribution at once, this section severs the hard link between components in the left- and right-hand models. Instead, a variational approach can assign part of the probability mass of component k in the left-hand model to any component m in the right-hand model. This tightens an upper bound on the KL divergence. Within this variational framework, other forms of mixture models can also be estimated. A new form of predictive CMLLR, which transforms clean speech Gaussians with shared linear transformations, can be derived. This yields optimal linear transformations for modelling the non-linear effects of the noise.

2.1 Experiments

To illustrate the limitations of the matched-pair approximation, the variational predictive methods described in this paper are tested on the 1000-word-vocabulary Resource Management corpus (Price *et al.* 1988). This task contains 109 training speakers reading 3990 sentences, a total of 3.8 hours of data. All results are averaged over three of the four available test sets, Feb 89, Oct 89, and Feb 91 (Sep 92 is not used), a total of 30 test speakers and 900 utterances. Operations Room noise from the NOISEX-92 database (Varga and Steeneken 1993) is artificially added at 20 and 14 dB. For a fair comparison, the influence of the noise estimation algorithm should be eliminated. Therefore, a full-covariance Gaussian noise model is extracted directly from the noise audio. (A method that could estimate a noise model for Monte Carlo PCMLLR is proposed in van Dalen and Gales (2011*b*).) Because a noise model also implicitly compensates for speaker differences, word error rates will be slightly higher than in van Dalen and Gales (2011*c*).

State-clustered triphone models with six components per mixture are built using the HTK RM recipe (Young *et al.* 2006). The number of components is about 9500. The extended speech statistics are striped as in van Dalen and Gales (2011*c*). The language model is a word-pair grammar.

Table 1 on the next page contains word error rates for model compensation. "VTS" is a standard scheme. It linearises the mismatch function and applies the *continuous-time approximation*, which makes off-diagonal elements of the covariance matrix unreliable (van Dalen and Gales 2011*c*), so the covariance matrices are (as is usual) diagonalised.

Extended DPMC (eDPMC) estimates means and covariances for each component separately using Monte Carlo. As the number of samples goes to infinity (at 100 000 samples, performance has converged) it yields the optimal Gaussian-for-Gaussian compensation. Compared to VTS, eDPMC removes the linearisation of the mismatch function and the continuous-time approximation (see van Dalen and Gales (2011*c*)), and thus yields improved performance. Removing the diagonalisation for eDPMC is especially useful at lower signal-to-noise ratios, when the noise affects feature correlations most. At 14 dB, it yields another 15 % relative increase in performance.

The above results still assume that one clean speech Gaussian generates Gaussian-distributed corrupted speech. The bottom row of Table 1 shows the effect of removing this constraint. Variational eDPMC uses the variational approach from section 2, which allows corrupted-speech samples to be re-assigned to different components. Because probability mass can be moved to different components, the state-conditional mixture distribution is able to model the non-linear effects of the interaction of the speech and noise. This yields a 10 % relative reduction in word error rate compared to the optimal Gaussian-for-Gaussian compensation.

Table 2 on the following page shows the same contrasts, but estimating only the parameters of linear transformations, by applying predictive CMLLR. Non-variational PCMLLR can be trained from different forms of statistics. The word error rates in the first row use statistics from a VTS-compensated model. The combination of the approximations in VTS and PCMLLR leads to reduced performance. The numbers in the second row are from CMLLR essentially trained from full-covariance eDPMC statistics,

Table 1 *Word error rates for model compensation.*

Method	Optimisation	Shape	20 dB	14 dB
VTS	per component	diag	8.6	17.4
eDPMC	per component	diag	7.5	14.9
		full	7.4	13.3
	variational	full	6.9	12.0

Table 2 *Word error rates for Predictive* CMLLR.

	Base classes			
	16	1024	16	1024
Statistics	20 dB		14 dB	
VTS	10.9	10.0	23.8	20.9
eDPMC	9.2	8.1	19.3	16.4
Variational	8.7	7.9	19.6	15.1

with 10 000 samples per base class. Though the clean speech samples can be drawn off-line, collecting the statistics is still costly. However, for the 16 base class system, the word error rate is the same with 5000 samples per base class, which makes the total number of samples 80 000. The complexity of this is invariant to the number of components, so for larger systems this operating point could yield a reasonable trade-off. With 1024 base classes, PCMLLR with eDPMC does perform better than VTS. Though the full transformation matrix of PCMLLR implicitly performs some compensation for correlation changes, it cannot model the non-Gaussian shape of the distributions.

The bottom row in Table 2 shows results for variational PCMLLR, which applies a variational approach to estimating linear transformations per base class. It allows the transformations to move components in space to model the non-Gaussian distribution of the corrupted speech. At 20 dB, around 26 % of the probability mass of the samples goes to different components; at 14 dB, 37 %. At 16 base classes this does not consistently yield benefits, but with 128 or more, and especially at lower signal-to-noise ratios, word error rates improve compared to non-variational PCMLLR. This ability to model the non-Gaussian aspect of the corrupted-speech distribution with just linear transformations of clean speech Gaussians shows the power of the variational approach.

This section has viewed model compensation for noise-robustness and predictive linear transformations from a variational perspective. These methods can be seen as minimising an upper bound on the divergence between the corrupted speech and the model for decoding. It is possible to find a tighter bound by considering the divergence between states rather than between components. When applied to eDPMC and predictive CMLLR, this yields reductions in the word error rate. It is possible to model the non-linear impact of the noise with just linear transformations of Gaussians. The variational predictive framework should allow a wide range of schemes to be developed. These could, for example, address the computational cost of the schemes proposed here.

3 Score-space generation

This section describes an efficient approach to computing generative score-spaces. It is a summary of work published as van Dalen *et al.* (2012).

Generative score-spaces, which generalise Fisher score-spaces (Jaakkola and Haussler 1998), consist of log-likelihoods of generative models, and their derivatives. There are two reasons for using these scores as features for a classifier. With a zeroth-order generative score-space, which contains only the log-likelihood itself, based on word HMMs, classification is closely related to that of a HMM speech recogniser. This allows state-of-the-art techniques for HMM speech recognisers, such as methods for noise-robustness, to be leveraged. Secondly, derivatives even of frame-level log-likelihoods are functions of all frames in the segment. Thus, the conditional independence assumptions of the HMM are relaxed.

In this work a joint feature space is used. The feature vector is divided into separate sets of dimensions for each vocabulary entry v, and the other dimensions are zero:

$$\boldsymbol{\phi}(\mathbf{O}, w) = \begin{bmatrix} \delta(w = 1)\, \boldsymbol{\phi}_1(\mathbf{O}) \\ \vdots \\ \delta(w = V)\, \boldsymbol{\phi}_V(\mathbf{O}) \end{bmatrix}, \tag{1}$$

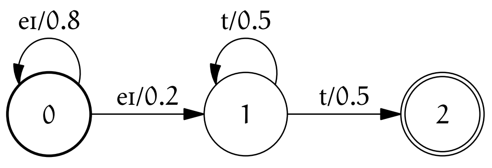
where $\delta(\cdot = \cdot)$ equals 1 if its argument is true, and 0 otherwise. In this expression, it selects the feature vector $\boldsymbol{\phi}_v(\mathbf{O})$ for word v.

Since the derivatives in the generative score-space depend on the frames in a whole segment, it seems obvious that they need to be re-computed completely for every hypothesised segment. This is, indeed, what Layton (2006) did, with an algorithm with nested passes of forward–backward that was essentially run separately for each hypothesised segmentation. However, this section will introduce a method that incrementally, with only a forward pass, computes scores for all segmentations that share a start time. It views the generative model as a weighted finite state transducer. In this formalism it is possible to generalise the weights (which would canonically represent HMM output and transition probabilities) to another semiring. This section will use *expectation semirings*, which allow for more extensive book-keeping. As long as the HMMs have only few states, which for word HMMs is the case, this algorithm requires a modest amount of extra storage. Its advantage is that in combination with a decoding method that finds the optimal segmentation, it computes the scores in amortised constant time.

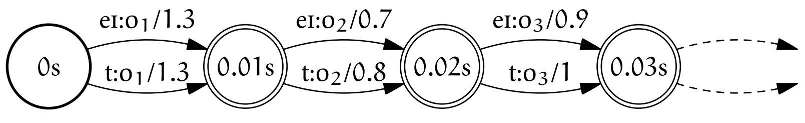
3.1 Log-likelihoods as weighted finite state transducers

Figure 3 illustrates the two component state machines required for computing the likelihood of a segment of speech. The automaton $\mathcal{S}$, in figure 3a, produces possible sequences of phones. It produces a sequence of one or more symbols eɪ and one or more t. A sequence of output symbols corresponds to a path, a sequence of arcs, starting at the start state ("0", with a bold circle), and finishing at the final state ("2", with a double circle). The total weight of the sequence is found by multiplying the weights of the arcs on the path. This represents the probability of the phone sequence. In the transducer drawn here, the probability is normalised, but that is not required of WFSTs.

For segments starting at each time, a transducer is set up representing phone likelihoods for the observations. A weighted finite state transducer $\mathcal{O}$ that does that for $\mathbf{O}_{1:t}$



(a) Weighted automaton $\mathcal{S}$ produces a sequenceof phones.



(b) WFST $\mathcal{O}$ with phone likelihoods for each observation $\mathbf{o}_t$.

Figure 3 *Component WFSTs for computing the likelihood of a segment.*

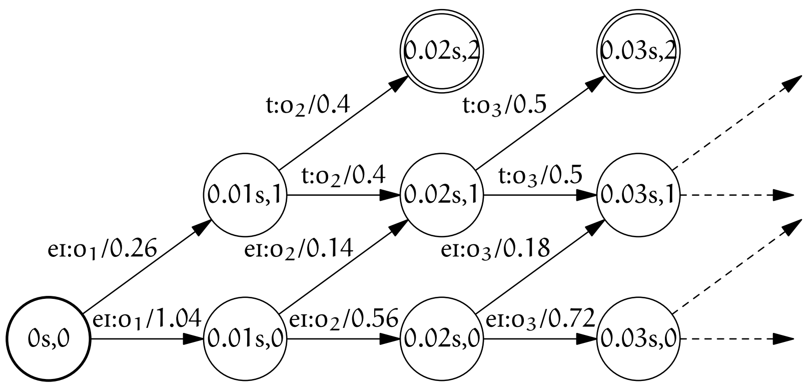


Figure 4 *WFST $\mathcal{T}$ for computing the likelihood of a segment: the two WFSTs from figure 3 composed.*

for consecutive t is depicted in figure 3b. Each state here represents a time (indicated in fractions of seconds from 0s). The audio in between two consecutive time steps is represented by feature vector $\mathbf{o}_t$. For each observation there is an arc for each of the phones. The weight on each arc is set to the likelihood that the output distribution of that phone (say, a mixture of Gaussians) gives for that observation $\mathbf{o}_t$. The product of the weights on the arcs on a path through this transducer therefore equals the observation likelihood for the phone sequence corresponding to the path.

In theory, automaton $\mathcal{S}$ produces an infinite number of sequences of phones with a weight assigned to each of the sequences. Transducer $\mathcal{O}$ represents an exponential number of pairs of an input and an output sequence, where each input sequence consists of phonemes, and each output sequence of observations $\mathbf{o}_1, \mathbf{o}_2, \ldots$. The interest is in only those combinations of sequences from $\mathcal{S}$ and $\mathcal{O}$ with the same phone sequence. Because both automata are finite-state machines, it is possible to represent the sequences of interest, and their weights, with a third transducer $\mathcal{T}$, which is obtained through composition of $\mathcal{S}$ and $\mathcal{O}$:

$$\mathcal{T} = \mathcal{S} \circ \mathcal{O}. \qquad (2)$$

This yields the transducer in figure 4. The states of this transducer are tuples of states from the two original transducers. For clarity, the states corresponding to one state from the transducer $\mathcal{O}$ representing the observations are in the same horizontal position, and those corresponding to phone sequences in $\mathcal{S}$ in the same vertical position. This way, the graph corresponds to the trellis diagrams sometimes drawn to explain Viterbi or the forward algorithm in HMMs.

The product of the weights that $\mathcal{S}$ and $\mathcal{O}$ assign represents the joint distribution of phone sequence and observation sequence. This is equivalent to the product of the weights on the corresponding path $\pi = e_1 \ldots e_t$ from a start to an end state in the WFST $\mathcal{T}$. The interest here is in the total weight over all paths from a specific start state and a specific end state. The forward algorithm is a well-known algorithm that computes this quantity. It uses dynamic programming, incrementally computing the weights up to a frontier of states. The frontier is normally moved forward time-synchronously. For this reason, it is possible to compute the likelihood of a word from one start time to a range of end times with one forward pass. Assuming the number of states in the HMM constant, each additional end state requires only constant time to compute. This means that if the likelihood is required for all possible segments, with the forward algorithm they can be computed in amortised constant time (Ragni and Gales 2012).

3.2 Computing scores with higher-order expectation semirings

The *expectation semiring* was introduced in Eisner (2002). Its initial purpose was to allow expectation–maximisation on weighted finite state transducers with a probabilistic interpretation. The statistics, which in speech recognition training would be computed after applying the forward–backward algorithm, are appended to the weights. By defining the weights to be in the expectation semiring, which defines operations $\oplus$ and $\otimes$ in a specific way, only a forward pass is required to gather all required statistics. For normal speech recognition training, the cost of carrying statistics for a complete HMM in each state would be prohibitive. However, in this paper the HMMs are small, and different lengths for the observation segments need to be considered. The following will therefore define weights in a semiring so that the algorithm in section 3.1 can be applied, and higher-order generative scores computed in amortised constant time.

A simple way of viewing the required semiring is as appending derivatives to the weights (Li and Eisner 2009). The weight for arcs e then becomes

$$w[e;\boldsymbol{\lambda}] \triangleq \langle l[e;\boldsymbol{\lambda}]\,, \nabla_{\boldsymbol{\lambda}} l[e;\boldsymbol{\lambda}]\rangle\,. \tag{3}$$

The semiring operations defined on these new weights can be derived in various ways. The simplest for the purpose of this paper is to describe the derivatives of the sum or product of two weights l_1 and l_2:

$$\nabla_{\boldsymbol{\lambda}}(l_1 + l_2) = \nabla_{\boldsymbol{\lambda}} l_1 + \nabla_{\boldsymbol{\lambda}} l_2; \tag{4a}$$
$$\nabla_{\boldsymbol{\lambda}}(l_1 \cdot l_2) = l_1 \cdot \nabla_{\boldsymbol{\lambda}} l_2 + l_2 \cdot \nabla_{\boldsymbol{\lambda}} l_1\,. \tag{4b}$$

Denoting the weights with $\langle l, l' \rangle$, the semiring operations should be defined as

$$\langle l_1, l_1' \rangle \oplus \langle l_2, l_2' \rangle \triangleq \langle l_1 + l_2, l_1' + l_2' \rangle\,; \tag{5a}$$

$$\langle l_1, l_1' \rangle \otimes \langle l_2, l_2' \rangle \triangleq \langle l_1 \cdot l_2, l_1 \cdot l_2' + l_2 \cdot l_1' \rangle\,; \tag{5b}$$

$$\overline{0} \triangleq \langle 0, 0 \rangle\,; \tag{5c}$$

$$\overline{1} \triangleq \langle 1, 0 \rangle\,. \tag{5d}$$

This definition ensures that both operations $\oplus$ and $\otimes$ always produce the derivative of the first element in the second element. Applying the forward algorithm on a weighted finite state transducer in the expectation semiring therefore produces en each final state a tuple with as first element the likelihood, and as second element its derivative. If the likelihood for a segment $\mathbf{O}_{\tau:t}$ is denoted with $l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}]$, the resulting tuple is

$$\langle l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}]\,, \nabla_{\boldsymbol{\lambda}} l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}] \rangle\,. \tag{6}$$

Finding the first-order score with the derivative of the log-likelihood (as opposed to the likelihood) is straightforward:

$$\nabla_{\boldsymbol{\lambda}} \log l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}] = \frac{\nabla_{\boldsymbol{\lambda}} l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}]}{l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}]}, \tag{7}$$

which can be found with the values in (6).

It is also possible to find second-order derivatives in the same way. The *second-order expectation semiring* (Li and Eisner 2009) is found through a "lifting trick": since first-order weights are in a semiring, their derivatives can be appended:

$$\begin{aligned} w[e; \boldsymbol{\lambda}] &\triangleq \Big\langle \langle l[e;\boldsymbol{\lambda}]\,, \nabla_{\boldsymbol{\lambda}} l[e;\boldsymbol{\lambda}] \rangle, \nabla_{\boldsymbol{\lambda}}^{\mathsf{T}} \langle l[e;\boldsymbol{\lambda}]\,, \nabla_{\boldsymbol{\lambda}} l[e;\boldsymbol{\lambda}] \rangle \Big\rangle \\ &= \Big\langle \langle l[e;\boldsymbol{\lambda}]\,, \nabla_{\boldsymbol{\lambda}} l[e;\boldsymbol{\lambda}] \rangle, \langle \nabla_{\boldsymbol{\lambda}}^{\mathsf{T}} l[e;\boldsymbol{\lambda}]\,, \nabla_{\boldsymbol{\lambda}}^{\mathsf{T}} \nabla_{\boldsymbol{\lambda}} l[e;\boldsymbol{\lambda}] \rangle \Big\rangle. \end{aligned} \tag{8}$$

Assuming the number of generative parameters constant, the forward algorithm can produce $l[\mathbf{O}_{\tau:t}; \boldsymbol{\lambda}]$ and its derivatives for $t = \tau \ldots T$ in $\Theta(T - \tau)$ time, i.e. amortised constant time. Computing generative scores for each of these segments also takes constant time. First- or second-order generative scores for optimal decoding, which requires scores for all segmentations, can be found in amortised constant time.

3.3 Experiments

Optimal decoding with generative score-spaces was tested in a log-linear model (see section 4.1) on a small, noisy corpus: AURORA 2. This makes it possible to test the interaction with noise compensation methods. The task uses a small vocabulary and no language model, which makes experiments without such optimisations as pruning possible. AURORA 2 (Hirsch and Pearce 2000) is a standard digit string recognition task. The generative model has whole-word HMMs with 16 states and 3 components per state. The number of HMM parameters is 46 732. The HMMs are compensated with unsupervised vector Taylor series (VTS) compensation as in Gales and Flego (2010).

Three different sets of HMM parameters are used to derive features for the discriminative model: trained on clean data, trained on corrupted data with VTS adaptive training (VAT), and on corrupted data with discriminative VTS adaptive training (DVAT).

With zeroth-order score-spaces, the discriminative model has 13 parameters, corresponding to the log-likelihoods of the 13 words (11 digits plus "sil" and "sp"), found as in section 3.1. In first-order score-spaces the derivatives of the log-likelihood are computed as in section 3.2 and appended. Only derivatives of the compensated means are used, since including variances led to rapid over-fitting. The number of parameters was 21 554. Second-order score-spaces resulted in generalisation problems because of the small training set, and initial experiments did not yield improvements over first-order score-spaces.

The discriminative models were initialised to use the likelihoods from the generative models unchanged. They were then trained with a minimum Bayes risk criterion as in Ragni and Gales (2011*b*) and see section 4.1. This used a large lattice with many, but not all, segmentations to represent the numerator and denominator. One of the three test sets, test set A, was used as the validation set to stop training.

Generative model	Score-space order	Test set A	B	C	Average
	—	9.8	9.1	9.5	9.5
VTS	zeroth	7.8	7.3	8.0	7.6
	first	6.8	6.4	7.3	6.7
	—	8.9	8.3	8.8	8.6
VAT	zeroth	7.1	6.8	7.5	7.1
	first	6.2	6.1	6.8	6.3
	—	6.7	6.6	7.0	6.7
DVAT	zeroth	6.6	6.5	6.9	6.6
	first	6.1	6.1	6.6	6.2

Table 3 *Word error rates for decoding with generative score-spaces.*

Table 3 contains word error rates for the experiments. Comparing the first two rows of each block will give an insight in the properties of the log-linear model. The differences between second and third rows of each block indicate the effects of using derivatives as features.

Results obtained by the generative model with Viterbi decoding are in the first row of each block. For the second row, log-likelihoods are extracted from this model as features for the log-linear model, and the optimal segmentation is found. For generative models with ("VTS') and without ("VAT") adaptive training this gives an improvement close to 20 % relative. However, discriminatively trained HMMs ("DVAT") are similar to the log-linear model derived from just log-likelihoods (Heigold *et al.* 2011). The most significant differences here are that the log-linear model chooses the optimal word sequence and marginalises out over state sequences within the word. This gives only a tiny improvement.

The bottom row of each block contains word error rates using first-order derivatives of log-likelihoods as features. These break the Markov assumption of HMMs. Interestingly, the effect of this seems only partly dependent of how good the underlying HMM is. The improvement compared to score-spaces with just log-likelihoods is 11–12 % relative for VTS and VAT. The discriminatively trained HMM (DVAT) has been optimised for decoding with it, rather than for use within a log-linear model. It is therefore encouraging that the relative gain with first-order derivatives is as high as 6 %.

4 Classifiers

Most current speech recognition systems are based on hidden Markov models (HMMs). These are generative models, which can be used for classifying data using Bayes' rule. The restriction on the form of classifier this yields can be lifted by extracting features from the audio with a generative model, and applying any of a number of known classifiers.

Section 3 has shown how to generate generative score-spaces from HMM-like models to model the acoustics. This section will discuss how to use these in various classifiers. The main challenge for this is that speech is structured, as sequences of words. Without exploiting this structure, the number of possible classes is infinite (exponential in the number of words). Section 4.1 will detail log-linear models that train parameters for acoustic features and language model features. Section 4.2 will report on speech recognition with structured SVMs, which use large-margin training. Section 4.3 will discuss a Bayesian non-parametric classifier: the infinite Gaussian mixture model.

4.1 Log-linear models

One type of classifier is a log-linear model. It uses a feature function $\boldsymbol{\phi}(\cdot)$ and a corresponding parameter vector $\boldsymbol{\alpha}$. Unlike a generative model, a conditional model yields the probability of hidden variables, in this case the word sequence $\boldsymbol{w}$, given the observation sequence $\mathbf{O}$. An issue is that the input sequence must be segmented into, say, words. Each of the elements w_i of $\boldsymbol{w}$ is equal to one element v from the vocabulary $\boldsymbol{v}$. The log-linear form of the model can be written:

$$P(\boldsymbol{w}|\mathbf{O}, \mathbf{s}; \boldsymbol{\alpha}) \triangleq \frac{1}{Z(\mathbf{O})} \exp\bigl(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \boldsymbol{w}, \mathbf{s})\bigr). \tag{9}$$

Here, $\mathbf{s} = \{s_i\}_{i=1}^{|\boldsymbol{w}|}$ is a segmentation of the observation sequence into segments s_i. $Z(\mathbf{O})$ is the normalisation constant. $\boldsymbol{\phi}(\mathbf{O}, \boldsymbol{w}, \mathbf{s})$ is the feature function that returns a feature vector characterising the whole observation sequence. $\boldsymbol{\alpha}$ is the parameter vector.

The feature vector is divided into two parts, containing scores both related to the acoustics and to the language model:

$$\boldsymbol{\phi}(\mathbf{O}, \boldsymbol{w}, \mathbf{s}) = \begin{bmatrix} \boldsymbol{\phi}_{a}(\mathbf{O}, \boldsymbol{w}, \mathbf{s}) \\ \boldsymbol{\phi}_{l}(\boldsymbol{w}) \end{bmatrix}. \tag{10}$$

Section 4.1.1 will discuss acoustic modelling with $\boldsymbol{\phi}_{a}$, and section 4.1.2 language modelling with $\boldsymbol{\phi}_{l}$.

4.1.1 Acoustic modelling

This section discusses work related to the project, published as Ragni and Gales (2011*a*;*b*).

To investigate acoustic modelling, the language model will be assumed to be its generative form, and the corresponding parameter set to 1. The acoustics are modelled with a semi-Markov model, so that the distribution factorises over the segments, i.e. the feature function is a sum of features for each segment:

$$\boldsymbol{\phi}_{\mathsf{a}}(\mathbf{O}, \boldsymbol{w}, \mathbf{s}) \triangleq \sum_{i} \boldsymbol{\phi}(\mathbf{O}_{s_i}, w_i), \tag{11}$$

where $\mathbf{O}_{s_i}$ indicates the observations in segment s_i.

In section 3 it was assumed that the segmentation consists of words for simplicity of notation. This works well if the training data contain enough instances of each word. However, for large-vocabulary speech recognition this is not normally the case. The usual way around this problem is to share parameters between different phones or sub-phones. In this work, the generative model is sub-phone-tied using a decision tree. For the discriminative model, however, it is best to be tied on the phone level. Another decision tree can be built, but care must be taken to ensure that the effective models, combining generative and discriminative parameters, can be trained robustly. Here, only those discriminative parameters are clustered where the generative model for the correct class appears at the leaf nodes of the decision trees created for the generative models.

An additional problem is that the assignment of mixture components is not fixed. Applying the same discriminative parameters for, say, component 4 in one mixture as component 4 in another mixture is meaningless. Therefore, the discriminative parameters are additionally tied over all components within a mixture. Expressing the tying structure as a matrix $\mathbf{A}$ which contains only one 1 per column and otherwise only zeros, (11) becomes

$$\boldsymbol{\phi}_{\mathsf{a}}(\mathbf{O}, \boldsymbol{w}, \mathbf{s}) \triangleq \sum_{i} \mathbf{A}\boldsymbol{\phi}(\mathbf{O}_{s_i}, w_i). \tag{12}$$

4.1.1.1 Parameter estimation The standard criterion for training log-linear models is the conditional likelihood. This type of training is called conditional maximum likelihood, or CML for short.

A criterion that is used to train the acoustic models of HMMs is an approximation to the Bayes' risk. This is called minimum Bayes' risk (MBR) training and it assumes a deterministic decoder. The objective is to minimise the expected loss in the average sentence accuracy over utterances $r = 1 \ldots R$

$$c_{\mathsf{mbr}}(\boldsymbol{\alpha}) \triangleq \frac{1}{R} \sum_{r} \sum_{\boldsymbol{w}} \sum_{\mathbf{s}} P\big(\boldsymbol{w} \big| \mathbf{O}^{(r)}, \mathbf{s}; \boldsymbol{\alpha}\big) k\big(\boldsymbol{w}, \boldsymbol{w}^{(r)}\big), \tag{13}$$

where $k\big(\boldsymbol{w}, \boldsymbol{w}^{(r)}\big)$ is the loss for word sequence $\boldsymbol{w}$ if the correct sequence is $\boldsymbol{w}^{(r)}$. Minimising (13) is expensive. An approach often used for training HMM speech recognisers is to align the reference transcription with the audio (Povey 2003). The accuracy of a hypothesised segment can then be approximated by examining the overlap with the reference transcription. The minimisation of the expected Bayesian risk can be performed numerically. It is usual to use phone segments, as will be done here. This is called minimum phone error (MPE).

System	Order of features	Test set A	B	C	Average
VTS		9.8	9.1	9.5	9.5
Log-linear	zeroth	8.1	7.4	8.2	7.6
	first	7.0	6.6	7.6	7.0

Table 4 *Results for log-linear modelling of acoustics on* AURORA *2.*

4.1.1.2 Experiments To test acoustic modelling with a log-linear model, the AURORA 2 and 4 corpora were used.

AURORA 2 is a small vocabulary digit string recognition task. It is a standard corpus for testing noise-robustness. Utterances are one to seven digits long and based on the TIDIGITS database with noise artificially added. The clean speech training data comprises 8440 utterances from 55 male and 55 female speakers. The test data is split into three sections. Test set A comprises 4 noise conditions: subway, babble, car and exhibition hall. Matched training data is available for these test conditions, but not used in this work. Test set B comprises 4 different noise conditions. For both test set A and B the noise is scaled and added to the waveforms. For the two noise conditions in test set C convolutional noise is also added. Each of the conditions has a test set of 1001 sentences with 52 male and 52 female speakers. The HMM set is trained on clean data. The "simple" back-end is used. VTS compensation is estimated separately for each utterance. The results are averaged over the different signal-to-noise ratios from 0–20 dB.

AURORA 4 is a noise-corrupted medium- to large-vocabulary task based on the Wall Street Journal data. The HMM model is trained on 14 hours of clean or multi-style data. The HMMs are state-clustered triphones (with 3140 states) with 16 components per mixture. Four iterations of VTS compensation are performed for the test data.

RProp is used as the optimisation algorithm. Both training and testing is done within a lattice rescoring framework: the lattices are found with the HMM recogniser. The alignments in the lattices are used.

Table 4 summarises results for the AURORA 2 task. The first row contains results for a system trained on clean data, and compensated for the inferred noise with VTS. This is the state of the art for HMMs with noise compensation. Using just the log-likelihoods that this system yields as features for the log-linear model yields the results in the second row. The substantial average gain of 20 % relative results from the freedom in the parametrisation that the discriminative model gives. The bottom row additionally uses first-order derivatives of the word log-likelihoods. This additional information, which, breaks the frame-level Markov assumption, produces an additional 10 % relative gain.

Results on a noise-corrupted medium-vocabulary task, AURORA 4, are in table 5 on the facing page. The first row contains the standard clean-trained, VTS-compensated system. For the second row, the generative model was trained on multi-style data with maximum-likelihood estimation, but using the VTS compensation while training and optimising for that. This is called "VTS adaptive training" (VAT). The next row uses "discriminative VTS adaptive training" (DVAT) (Flego and Gales 2011), which trains the generative model discriminatively. This retains the nominal property of the HMM model, so that the distributions are locally normalised, but the training objective is minimum Bayes' risk.

System	Order of features	Test set A	B	C	D	Average
VTS		7.1	15.3	12.1	23.1	17.9
VAT		8.6	13.8	12.0	20.1	16.0
DVAT		7.2	12.8	11.5	19.7	15.3
Log-linear	zeroth	7.7	13.1	11.0	19.5	15.3
	first	7.4	12.6	10.7	19.0	14.8

Table 5 *Results for log-linear modelling of acoustics on* AURORA *4.*

The log-linear system instead trains discriminative parameters on the log-likelihoods given by the VAT system. With only 4020 parameters, it yields similar performance on average to discriminatively training of the complete HMM set, but it is more robust to noisy data. Adding first-order derivatives of the likelihoods, the bottom row, yields a consistent gain of 4 % on top of that by breaking the frame-level Markov assumption.

4.1.2 Language modelling

To investigate discriminative language modelling, a standard HMM acoustic model is used. This means that the acoustic feature vector $\boldsymbol{\phi}_{\mathsf{a}}(\mathbf{O}, \boldsymbol{w}, \mathsf{s})$ in (10) is set to a single scalar. It approximates the likelihood according to an HMM speech recogniser with the Viterbi approximation.

The language model features $\boldsymbol{\phi}_{\mathsf{l}}(\boldsymbol{w})$, is the interest. Its first entry is the log-probability returned by the generative language model. Appended to this are features for each transition in the finite-state representation of the language model, including back-off transitions. These count the number of times each transition is taken. The corresponding parameter in the parameter vector $\boldsymbol{\alpha}$ therefore adds a bias to the probability given by the generative model. This is a very similar set-up to that of Roark *et al.* (2007). To reconstruct the generative language model, the parameter vector for $\boldsymbol{\phi}_{\mathsf{l}}$ can be set to $\begin{bmatrix} 1 & 0 & 0 & \ldots \end{bmatrix}^{\mathsf{T}}$.

The standard criterion for training log-linear models is the conditional likelihood over the observations:

$$\mathsf{c}_{\mathsf{cml}}(\boldsymbol{\alpha}) \triangleq \frac{1}{\mathsf{R}} \sum_{\mathsf{r}} \sum_{\mathsf{s}} \mathsf{P}\big(\boldsymbol{w}^{(\mathsf{r})} \big| \mathbf{O}^{(\mathsf{r})}, \mathsf{s}; \boldsymbol{\alpha}\big). \tag{14}$$

Note that unlike (13), this does not explicitly take into account how word sequences compete with the reference sequence, just the reference sequence.

4.1.2.1 Experiments

To test discriminative language modelling, it is required to have sufficient coverage of the transitions in the language, i.e. a large training corpus. The Cambridge English conversational telephone speech system for the DARPA/NIST evaluation from 2004 (Evermann *et al.* 2005) was used. It uses 2200 hours of training data of conversational telephone speech.

The SCARF toolkit (Zweig and Nguyen 2010) is used. Its input is lattices generated by a HMM system trained with minimum phone error. The acoustic scores are given by the lattice. The language model is a trigram model. All transitions in this language

		ML	MPE	
Discriminative training	L_2 std dev		subset	all data
Fixed scaling factor		30.5	24.4	
Scaling factor		53.5		
Full language model	2.0×10^{-06}			24.8
	8.5×10^{-06}		24.4	
	1.2×10^{-05}		24.4	
	2.7×10^{-05}		24.5	
	4.4×10^{-05}			30.8

Table 6 *Training a language model with SCARF.*

model are used for the feature vector. To train the corresponding parameters, RProp is used. This is a gradient-based approach that adaptively changes the step size. SCARF by default sets the initial step size to 0.1. This yielded erratic behaviour of the objective function. The initial step size was therefore changed to 10^{-5}, which yielded smooth behaviour and convergence in around 50 iteratons.

Table 6 contains the results of training a language model with SCARF. The numbers quoted are on the eval03 test set. The first row contains the baseline numbers, which are similar to those in Evermann *et al.* (2005). This uses a fixed language model scaling factor. A first experiment (second row) estimated a language model scaling factor. This impaired performance severely. It is suspected that this is caused by how the training lattices are generated. To increase acoustic confusability, the language model used for generating the lattices is a pruned bigram model. Though for training the language model scaling factor the lattice is rescored with the correct language model, this cannot introduce new hypotheses.

However, the lattices can still contain enough information to train the language model discriminatively. The sparseness of the training data for this purpose means that regularisation is required. L_2 regularisation is used, since sparsification is not necessary for initial experiments. Since each iteration over the full training set takes 12 hour on a modern 16-core machine, only a limited number of experiments is run, and initial experiments use 1/20th of the training data. SCARF multiplies the L_2 norm by the number of utterances. The regularisation constants therefore are not consistent between test sets. Table 6 reports the regularisation constant as the standard deviation.

The results are disappointing: it is possible to match the original performance, but not to improve on it. A number of factors play a role. First, unlike in Roark *et al.* (2007), which uses ML-trained acoustic models, the results here are based on discriminatively trained acoustic models. This may already have taken out many of the errors, so that discriminative training of the language model does not add anything. On a more practical note, the training data is re-used after MPE training, which implies that the acoustic models may match the training data too well. An alternative scheme would hold out part of the training data for language model training. It might also be possible to train the acoustic and language models at the same time. Another cause may lie in the mismatch between the ideal lattices for acoustic model training and language model training. Discriminative training of the acoustic model in practice requires an uninformative language model to artificially increase the acoustic confusability. Yet another factor could be that there are too many parameters to train, and some tying scheme should be found. Investigation of these factors remains future work.

4.2 Structured support vector machines

The theory behind binary support vector machines (Vapnik 2000) and multi-class SVMs have been well established in the machine learning literature. More recently structured support vector machines (SSVMs) (Tsochantaridis *et al.* 2005) have been proposed to handle situations where there is structured data to classify. This section describes the application of SSVMs to speech recognition. This work is related to the project, and published as Zhang and Gales (2011*b*;*a*).

When SVMs are generalised to deal with speech recognition, the decision rule in its primal form is

$$\boldsymbol{w}^* \triangleq \arg\max_{\boldsymbol{w}} \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \boldsymbol{w}, \mathbf{s}), \tag{15}$$

where the segmentation $\mathbf{s}$ is assumed known. This decision rule is equivalent to the decision rule that drops out for decoding with a log-linear as in (9) with the same parameter vector and feature function.

However, training is different. Just like in normal SVMs, it optimises the margin between the correct classification and the closest competitors. The SSVM criterion can be expressed as minimising

$$\begin{aligned} &\tfrac{1}{2}\|\boldsymbol{\alpha}\|_2^2 + \frac{\mathrm{C}}{\mathrm{R}}\Bigg[-\max_{\mathbf{s}}\left\{\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \boldsymbol{w}^{(r)}, \mathbf{s})\right\} \\ &\qquad + \max_{\boldsymbol{w}\neq\boldsymbol{w}^{(r)},\mathbf{s}}\left\{\mathrm{k}\Big(\boldsymbol{w}, \boldsymbol{w}^{(r)}, +\Big)\,\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \boldsymbol{w}, \mathbf{s})\right\}\Bigg]_+, \end{aligned} \tag{16}$$

where the first term inside the square brackets relates to the correct hypothesis and the second term to the competing hypotheses. Both max functions are convex with respect to $\boldsymbol{\alpha}$, but the first one is negated, so the complete optimisation is non-convex. To solve this non-convex optimisation problem, an approach similar to the concave-convex procedure can be applied.

Optimising the segmentations is an additional problem compared to standard structured SVMs. This is slow, so that it becomes important to run the optimisation algorithm in parallel. Two possibilities are considered: using the "1-slack" algorithm instead of the "n-slack" algorithm, or to approximate the latter. For more details, see Zhang and Gales (2011*b*;*a*).

4.2.1 Experiments

The AURORA 2 and 4 corpora are used to evaluate speech recognition with a structured SVM. Both tasks have been described in section 4.1.1.2. Only zeroth-order scores, log-likelihoods, are used as features. The generative models are compensated with VTS compensation.

Table 7 on the following page compares the baseline HMM system and structured SVMs on the smaller corpus, AURORA 2. The 1-slack algorithm is used, with a prior on the parameter vector. (The n-slack algorithm performed slightly worse due to the approximations it requires in parallel computation.) Because the maximum-margin criterion provides better generalisation, performance is 20 % relative better than for HMMs.

Model	A	B	C	Average
HMM	9.8	9.1	9.5	9.5
SSVM	7.5	7.1	7.9	7.4

Table 7 *Word error rates for structured support vector machines on* AURORA *2.*

Model	A	B	C	D	Average
HMM	7.1	15.3	12.2	23.1	17.8
SSVM	7.4	14.2	11.3	21.9	16.8

Table 8 *Word error rates for structured support vector machines on* AURORA *4.*

Table 7 compares the baseline HMM system and structured SVMs on AURORA 4. In this case, the n-slack algorithm cannot be run because due to the larger vocabulary it generates too many constraints. The structured SVM produces a 6 % relative gain compared to the HMM system.

4.3 Infinite Gaussian mixture models

An alternative to the standard forms of discriminative models is to use a non-parametric Bayesian approach (Rasmussen 2000; Beal *et al.* 2002; Blei *et al.* 2004; MacKay and Peto 1994; Teh 2006; Fox *et al.* 2008). This section describes some initial work in this direction based on a simple form of generative Bayesian model, the infinite Gaussian mixture model (IGMM). Later work will investigate more complex forms of model for example the infinite HMM and the infinite SVM for classifying speech.

4.4 Model Definition

The infinite Gaussian mixture model is a Gaussian mixture model with an infinite number of components. The key insight to making inference in such a model possible is that since the training data $\mathcal{D} = \{x_n\}_{n=1}^{N}$ has N points, the maximum number of components that can have generated these points is N. First consider a standard Gaussian mixture model (GMM) with M components:

$$p(\mathbf{x}|\theta) = \sum_{m=1}^{M} c^{(m)} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{(m)}, \boldsymbol{\Sigma}^{(m)}) \tag{17}$$

Thus the set of the model parameters θ has the form

$$\theta = \left\{ M, \{c^{(1)}, \dots, c^{(M)}\}, \{\boldsymbol{\mu}^{(1)}, \dots, \boldsymbol{\mu}^{(M)}\}, \{\boldsymbol{\Sigma}^{(1)}, \dots, \boldsymbol{\Sigma}^{(M)}\}, \right\} \tag{18}$$

with a sum-to-one constraint on the component priors $\{c^{(1)}, \dots, c^{(M)}\}$. This GMM is a parametric model where a single set of parameters is used. Typically the number of components M is fixed a-priori (though this is not a requirement) and the priors, means and covariances estimated using ML or MAP. The MAP setting of the parameters is

$$\hat{\theta}|M, \mathcal{D} = \arg\max_{\theta} \{p(\mathcal{D}|\theta)p(\theta|M)\} \tag{19}$$

where $\mathcal{D}$ are the training data. Through the prior in $p(\theta|M)$, the variances are forced to be non-zero.

Rather than considering a single set of model parameters, with a fixed number of components, a Bayesian approach can be adopted. Here the likelihood of the next observation is defined as

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\theta)p(\theta|\mathcal{D})d\theta \tag{20}$$

Though simple to express in this form, marginalising over the distribution is impractical for GMMs (for individual Gaussians, it is possible to use a prior that yields a closed-form solution) and the form of the posterior of the parameters $p(\theta|\mathcal{D})$ is highly complicated. To handle this problem a non-parametric Bayesian approach can be adopted. Here samples of the parameters are drawn from $p(\theta|\mathcal{D})$. The posterior distribution does not have to be explicitly specified. The samples are used to approximate the marginalisation over the parameters. Thus, if N samples are drawn,

$$p(\mathbf{x}|\mathcal{D}) \approx \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{x}|\theta_i) \tag{21}$$

where

$$\theta_i \sim p(\theta|\mathcal{D}). \tag{22}$$

This model and procedure can be extended to the case where the number of components is infinite. In this case, instead of drawing component priors from a Dirichlet distribution, components are drawn from a Dirichlet process with an appropriate base measure. At each stage of the sampling procedure, only the parameters of the components that are assigned data points need to be represented explicitly. The other, infinitely many, components' posterior distribution is equal to their prior. It is therefore possible to integrate out over them. The process of drawing these samples is described in detail in Rasmussen (2000).

In this project the task is classification of speech. This contrasts with most published approaches where these non-parametric Bayesian approaches are used for clustering (Fox *et al.* 2011). Classification with IGMMs can be achieved by splitting the training data into appropriate classes $\mathcal{D}_\omega$, and training a separate IGMM for each class:

$$\hat{\omega}|\mathbf{x} = \arg\max_{\omega}\{P(\omega)p(\mathbf{x}|\mathcal{D}_\omega)\} \tag{23}$$

where $\mathcal{D}_\omega$ is the training data associated with class ω.

4.5 Adaptation and Compensation

One of the advantages of using simple forms of generative Bayesian model is that it is possible to use compensation and adaptation approaches. These approaches are not yet well-known in the machine learning community, but they are standard in speech recognition. Consider the case of global CMLLR[1] (Gales 1998). Given a transform $\{\mathbf{A}, \mathbf{b}\}$

[1]It is not possible to use standard regression classes as the component number and parameters are not fixed. However, it is possible to use class-specific transforms.

classification is based on

$$\begin{aligned}\hat{\omega}|\mathbf{x},\mathbf{A},\mathbf{b} &= \arg\max_{\omega}\{P(\omega)p(\mathbf{x}|\mathcal{D}_{\omega},\mathbf{A},\mathbf{b})\}\\ &= \arg\max_{\omega}\{P(\omega)p(\theta|\mathcal{D}_{\omega},\mathbf{A},\mathbf{b})p(\mathbf{x}|\theta,\mathbf{A},\mathbf{b})\}\\ &\approx \arg\max_{\omega}\left\{P(\omega)\frac{1}{N}\sum_{i=1}^{N}p(\mathbf{x}|\theta_{\omega i},\mathbf{A},\mathbf{b})\right\}\end{aligned} \tag{24}$$

where given a class ω samples i of the set of parameters $\theta_{\omega i}$ are drawn as

$$\theta_{\omega i} \sim p(\theta|\mathcal{D}_{\omega}) \tag{25}$$

Since each set of parameters $\theta_{\omega i}$ defines a GMM the likelihood can simply be written as

$$p(\mathbf{x}|\theta_{\omega i},\mathbf{A},\mathbf{b}) = |\mathbf{A}|\sum_{m=1}^{M_{\omega i}} c_{\omega i}^{(m)}\mathcal{N}(\mathbf{A}\mathbf{x}+\mathbf{b};\boldsymbol{\mu}_{\omega i}^{(m)},\boldsymbol{\Sigma}_{\omega i}^{(m)}) \tag{26}$$

In common with standard adaptation the parameters of the transform need to be estimated from some adaptation data $\mathcal{D}_a = \{\mathbf{x}_1,\ldots,\mathbf{x}_T\}$. Labels for each of these samples $y_1,\ldots,y_T$ are either known, or derived using an existing system. The transform parameters are then found from[2]

$$\begin{aligned}\hat{\mathbf{A}},\hat{\mathbf{b}}|\mathcal{D}_a,\mathcal{D} &= \arg\max_{\mathbf{A},\mathbf{b}}\left\{\sum_{\omega,\tau}\delta(w,y_\tau)\log\left(\int p(\mathbf{x}_\tau|\theta,\mathbf{A},\mathbf{b})p(\theta|\mathcal{D}_{\omega})\right)\right\}\\ &\approx \arg\max_{\mathbf{A},\mathbf{b}}\left\{\sum_{\omega,\tau}\delta(w,y_\tau)\log\left(\frac{1}{N}\sum_{i=1}^{N}p(\mathbf{x}_\tau|\theta_{\omega i},\mathbf{A},\mathbf{b})\right)\right\}.\end{aligned} \tag{27}$$

Here, $\delta(\cdot,\cdot)$ is the Kronecker delta. This now has the form of standard parameter estimation. Using EM the following auxiliary function can be obtained (ignoring constant terms)

$$\mathcal{Q}(\hat{\mathbf{A}},\hat{\mathbf{b}};\mathbf{A},\mathbf{b}) = \sum_{\omega,\tau}\delta(\omega,y_\tau)\sum_{i=1}^{N}\sum_{m=1}^{M_{\omega i}}\gamma_{\omega i}^{(m)}(\tau)\log\left(p(\mathbf{x}_\tau|\theta_{\omega i},m,\hat{\mathbf{A}},\hat{\mathbf{b}})\right) \tag{28}$$

where

$$\gamma_{\omega i}^{(m)}(\tau) = P(m|\mathbf{x}_\tau,\theta_{\omega i},\mathbf{A},\mathbf{b}) \tag{29}$$

This allows the parameters of the transform to be estimated in the same fashion as standard CMLLR. The same approach can be adopted for other compensation and adaptation approaches.

If the IGMM is to be trained on heterogeneous data, then it is also possible to apply the equivalent of adaptive training. Consider the training data is split into S blocks for class ω, $\mathcal{D}_{\omega} = \{\mathcal{D}_{\omega}^{(1)},\ldots,\mathcal{D}_{\omega}^{(S)}\}$ and there is a transform $\{\mathbf{A}^{(s)},\mathbf{b}^{(s)}\}$ for each of these blocks. As a global CMLLR transform can be applied to the features, the data from

[2] Here only standard ML approaches are adopted. It would be interesting to apply Bayesian approaches to the transforms as well.

each of the blocks can be transformed using the appropriate linear transform. Thus the draws for the IGMM are then based on

$$\theta_{\omega i} \sim p\big(\theta \big| \{\mathbf{A}^{(1)}, \mathbf{b}^{(1)}\mathcal{D}_{\omega}^{(1)}\}, \ldots, \{\mathbf{A}^{(S)}, \mathbf{b}^{(S)}\mathcal{D}_{\omega}^{(S)}\}\big) \tag{30}$$

and

$$\{\mathbf{A}^{(s)}, \mathbf{b}^{(s)}, \mathcal{D}_{\omega}^{(s)}\} = \left\{\mathbf{A}^{(s)}\mathbf{x}_1 + \mathbf{b}^{(s)}, \ldots, \mathbf{A}^{(s)}\mathbf{x}_{\tau_{\omega}^{(s)}} + \mathbf{b}^{(s)}\right\} \tag{31}$$

Since the transforms depend on the draws from the models, and the draws of the model depend on the transforms, the standard iterative approach of alternating the draws from the model and the transform estimation must be performed.

4.6 Preliminary Experiments

In theory IGMMs can be directly applied to sequence data as samples can be considered independent of each other. However in this work the scores from generative models are used to handle the dynamic aspects of the data. This allows the standard forms of noise robustness to be applied as an initial level of compensation. Additionally the IGMM can be adapted to the test conditions (here unsupervised adaptation is performed), or adaptive training performed. For all experiments a diagonal covariance matrix IGMM was used.

These experiments were conducted using the AURORA 2 corpus, using the same HMM training and VTS compensation configuration as Gales and Flego (2010) along with the use of the compensated models to segment the data. The set-up was a form of acoustic code-breaking, where the words in the one-best hypothesis are reclassified one by one.

The score-space in this work was based on the log-likelihood score space. However class log-posteriors, rather than log-likelihoods, were used.[3] Though many other forms of transformation of the log-likelihood score-space are possible these were not investigated.

System	Train Adapt	Test Adapt	Test set WER (%) A	B	C	Avg.
VTS	—	—	9.84	9.11	9.53	9.48
IGMM	—	—	11.47	10.95	11.63	11.30
	—	CMLLR	9.49	9.38	9.54	9.46
	CMLLR	CMLLR	9.62	9.67	9.72	9.62

Table 9 *AURORA 2 results comparing VTS and IGMM systems.*

Table 9 shows the performance of the baseline VTS system (used to derive the segmentation and score-space) and various configurations of the IGMM system. The first row gives results for IGMM reclassification without adaptation. For the second row, the CMLLR transform is estimated on the 1-best for all test data for each noise type and signal-to-noise ratio separately. The third row uses cluster-adaptive training, where the multi-condition training subsets from Gales and Flego (2010) are used for estimating

[3] An “acoustic deweighting” factor of 0.1 was used in deriving the class posteriors.

Task	Set	OOV rate (%)	Perplexity
YTElect	—	0.66	122.51
YTGeneral	dev	1.40	217.8
	eval	1.33	195.6

Table 10 *Perplexity and OOV rates on the YouTube data.*

Task	Set	Segmentation	
		Auto	Manual
YTElect	—	27.5	
YTGeneral	dev	54.0	47.5
	eval	59.0	

Table 11 *Word error rates on the YouTube data.*

the transforms. The performance of the IGMM in all configurations is disappointing. At this stage it is unclear what the issues are. These issues will be investigated in future work.

5 Experiments on YouTube data from Google

As part of the project, Google has provided audio data from YouTube videos that displays differences in speaking and adverse environments of various kinds. This is a hard task for a speech recogniser. Most of the videos that the data is from is still on YouTube, so that metadata can be found. This is not currently used.

The YTElect data (see also Alberti *et al.* 2009) contains audio from videos about the 2008 American presidential election, such as candidates' speeches. It is 9 hours of audio. The YTGeneral data is a random sample from YouTube data, and contains a "dev" set of 9 hours of audio and an "eval" set of 8½ hours.

Since especially the YTGeneral data contains a variety of audio that should not be recognised (such as music, background noise, and speech in other languages), automatic segmentation was performed. The YTGeneral corpus also has been segmented manually. The total fraction of audio present in the manual segmentation that is missing in the automatic segmentation is 5.3 %.

To have a baseline for the project, the CU-2004 Broadcast News system was used, up to the P1+P2 stage (Kim *et al.* 2005). Table 10 shows the perplexity and out-of-vocabulary rate on both corpora. It is clear that the language model fits the election data much better than the more general data.

Table 11 shows the corresponding word error rates. Performance on the YTElect task is not as good as on broadcast news (Kim *et al.* 2005). The general YouTube data, however, turns out to be much harder. With automatic segmentation, word error rates are higher than 50 %. With the manual segmentation, performance improves slightly, to 47.5 %. This is comparable to the performance of a Google baseline system.

6 Conclusion

This report has documented progress in three key areas within the EPSRC Project EP/I006583/1, Generative Kernels and Score Spaces for Classification of Speech. The areas are model compensation, score-space generation, and the classifiers.

A new approach for model compensation for noise-robustness was presented in section 2. It removes the Gaussian assumption that underlies standard model compensation, but finds a parametric form of compensation that it is fast to decode with.

Section 3 proposed a method to computing generative scores for many segmentations at once. This is vital for making speech recognition with generative score-spaces efficient (the standard Viterbi algorithm cannot be used). The method uses the forward algorithm for an generative model for one word. However, it replaces the forward probabilities with values in the *expectation semiring*, which allows derivatives of the probabilities to be propagated.

Section 4 discussed various classifiers to be used with generative score-spaces. Log-linear models were discussed for both acoustic modelling and for language modelling. Structured SVMs, which use a max-margin criterion for training, were then discussed. Finally, the use of infinite Gaussian mixture models was investigated.

6.1 Future work

The same three areas provide opportunities for further work.

Adaptation and noise-robustness specifically for within the classifiers is interesting. Currently the generative models are adapted with no reference to the generative scores or the classifier they are used in. Optimising the actual likelihood used for decoding may further increase performance.

The second area is the score-space generation. Currently the features that are extracted are log-likelihoods and their derivatives. Other features may be valuable. Also, regularisation techniques may allow higher-order derivatives to be included without harming generalisation and speed.

The third area is the classifiers, where three directions are identified. Firstly, approaches to bring the sophistication of log-linear models with generative models to large-vocabulary speech recognition. This will require a number of optimisations and approximations, both for training and decoding. Second is the the use of different kernels in structured SVMs. In the primal form, kernels are essentially restricted to a dot product between score-spaces. Structured SVMs can use the dual form, which allows, for example, radial basis kernels to be used. Thirdly, other non-parametric Bayesian approaches than infinite GMMs can be investigated. Combining the two types of classifiers mentioned, either infinite log-linear models (extending log-linear models in the way infinite HMMs extend HMMs) or infinite structured SVMs can be used.

Bibliography

Alex Acero, Li Deng, Trausti Kristjansson, and Jerry Zhang (2000). "HMM adaptation using vector Taylor series for noisy speech recognition." In *Proceedings of ICSLP*. vol. 3, pp. 229–232.

Christopher Alberti, Michiel Bacchiani, Ari Bezman, Ciprian Chelba, Anastassia Drofa, Hank Liao, Pedro Moreno, Ted Power, Arnaud Sahuguet, Maria Shugrina, and Olivier Siohan (2009). "An audio indexing system for election video material." In *Proceedings of ICASSP*. pp. 4873–4876.

Matthew J. Beal, Zoubin Ghahramani, and Carl E. Rasmussen (2002). "The Infinite Hidden Markov Model." In *Advances in Neural Information Processing Systems*. MIT Press, pp. 29–245.

D. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum (2004). "Hierarchical topic models and the nested Chinese restaurant process." *Advances in Neural Information Processing Systems 16.* .

Jason Eisner (2002). "Parameter Estimation for Probabilistic Finite-State Transducers." In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. pp. 1–8.

G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, D. Mrva, P. C. Woodland, and K. Yu (2005). "Training LVCSR Systems on Thousands of Hours of Data." In *Proceedings of ICASSP*.

F. Flego and M. J. F. Gales (2011). "Factor Analysis Based VTS and JUD Noise Estimation and Compensation." Tech. Rep. CUED/F-INFENG/TR.653, Cambridge University.

E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky (2011). "A Sticky HDP-HMM with Application to Speaker Diarization." *Annals of Applied Statistics* 5 (2A), pp. 1020–1056.

Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky (2008). "An HDP-HMM for systems with state persistence." In *Proceedings of the 25th international conference on Machine learning*. ICML, pp. 312–319.

M. J. F. Gales (1998). "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition." *Computer Speech and Language* 12 (2), pp. 75–98.

M. J. F. Gales and F. Flego (2010). "Discriminative classifiers with adaptive kernels for noise robust speech recognition." *Computer Speech and Language* 24 (4), pp. 648–662.

Georg Heigold, Hermann Ney, Patrick Lehnen, Tobias Gass, and Ralf Schlüter (2011). "Equivalence of Generative and Log-Linear Models." *IEEE Transactions on Audio, Speech, and Language Processing* 19 (5), pp. 1138–1148.

Hans-Günter Hirsch and David Pearce (2000). "The AURORA experimental framework for the performance evaluation of speech recognition systems under noise conditions." In *Proceedings of ASR*. pp. 181–188.

Tommi Jaakkola and David Haussler (1998). "Exploiting Generative Models in Discriminative Classifiers." In *Proceedings of NIPS*.

D. Y. Kim, H. Y. Chan, G. Evermann, M. J. F. Gales, D. Mrva, K. C. Sim, and P. C. Woodland (2005). "Development of the CU-HTK 2004 Broadcast News Transcription Systems." In *Proceedings of ICASSP*. vol. 1, pp. 861 – 864.

Trausti Thor Kristjansson (2002). *Speech Recognition in Adverse Environments: a Probabilistic Approach*. Ph.D. thesis, University of Waterloo.

Martin Layton (2006). *Augmented Statistical Models for Classifying Sequence Data*. Ph.D. thesis, Cambridge University.

Jinyu Li, Dong Yu, Yifan Gong, and L. Deng (2010). "Unscented Transform with Online Distortion Estimation for HMM Adaptation." In *Proceedings of Interspeech*. pp. 1660–1663.

Zhifei Li and Jason Eisner (2009). "First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

David J. C. MacKay and Linda C. Bauman Peto (1994). "A Hierarchical Dirichlet Language Model." *Natural Language Engineering* 1, pp. 1–19.

Daniel Povey (2003). *Discriminative Training for Large Vocabulary Speech Recognition*. Ph.D. thesis, Cambridge University.

P. Price, W. M. Fisher, J. Bernstein, and D. S. Pallett (1988). "The DARPA 1000-word Resource Management database for continuous speech recognition." In *Proceedings of ICASSP*. vol. 1, pp. 651–654.

A. Ragni and M. J. F. Gales (2012). "Inference Algorithms for Generative Score-Spaces." In *Proceedings of ICASSP*. p. accepted.

A. Ragni and M.J.F. Gales (2011*a*). "Derivative Kernels for Noise Robust ASR." In *Proceedings of ASRU*.

A. Ragni and M.J.F. Gales (2011*b*). "Structured Discriminative Models for Noise Robust Continuous Speech Recognition." In *Proceedings of ICASSP*.

Carl Edward Rasmussen (2000). "The infinite Gaussian mixture model." In *Advances in Neural Information Processing Systems*. MIT Press, pp. 554–560.

Brian Roark, Murat Saraclar, and Michael Collins (2007). "Discriminative n-gram language modeling." *Computer Speech and Language* 21 (2), pp. 373–392.

Mike Seltzer, Alex Acero, and Kaustubh Kalgaonkar (2010). "Acoustic Model Adaptation via Linear Spline Interpolation for Robust Speech Recognition." In *Proceedings of ICASSP*. pp. 4550–4553.

Yee Whye Teh (2006). "A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes." In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pp. 985–992.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun (2005). "Large margin methods for structured and interdependent output variables." *Journal of Machine Learning Research* 6, pp. 1453–1484.

R. C. van Dalen, F. Flego, and M. J. F. Gales (2009). "Transforming Features to Compensate Speech Recogniser Models for Noise." In *Proceedings of Interspeech*. pp. 2499–2502.

R. C. van Dalen and M. J. F. Gales (2010). "Asymptotically Exact Noise-Corrupted Speech Likelihoods." In *Proceedings of Interspeech*. pp. 709–712.

R. C. van Dalen and M. J. F. Gales (2011*a*). "A Variational Perspective on Noise-Robust Speech Recognition." In *Proceedings of ASRU*.

R. C. van Dalen and M. J. F. Gales (2011*b*). "A variational perspective on noise-robust speech recognition." Tech. Rep. CUED/F-INFENG/TR.670, Cambridge University Engineering Department.

R. C. van Dalen, A. Ragni, and M. J. F. Gales (2012). "Efficient decoding with continuous rational kernels using the expectation semiring." Tech. Rep. CUED/F-INFENG/TR.674, Cambridge University Engineering Department.

Rogier C. van Dalen and Mark J. F. Gales (2011*c*). "Extended VTS for Noise-Robust Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 19 (4), pp. 733–743.

Vladimir N. Vapnik (2000). *The nature of statistical learning theory*. Springer-Verlag, New York, NY, USA.

A. Varga and H. J. M. Steeneken (1993). "Assessment for automatic speech recognition II: NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems." *Speech Communication* 12 (3), pp. 247–251.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland (2006). "The HTK book (for HTK Version 3.4)." URL: ⟨http://htk.eng.cam.ac.uk/docs/docs.shtml⟩.

S.-X. Zhang and M.J.F. Gales (2011*a*). "Extending Noise Robust Structured Support Vector Machines to Larger Vocabulary Tasks." In *Proceedings of ASRU*.

S.-X. Zhang and M.J.F. Gales (2011*b*). "Structured Support Vector Machines for Noise Robust Continuous Speech Recognition." In *Proceedings of Interspeech*.

Geoffrey Zweig and Patrick Nguyen (2010). "SCARF: A Segmental Conditional Random Field Toolkit for Speech Recognition." In *Proceedings of Interspeech*.