Ensemble Feature Selection for Multi-Stream Automatic Speech Recognition

by

David Gelbart

B.S. (University of British Columbia) 1999

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nelson Morgan, Chair
Professor Jerome Feldman
Professor Keith Johnson

Fall 2008

The dissertation of David Gelbart is approved.

Chair	Date
	Date
	Date

University of California, Berkeley

Fall 2008

Abstract

Ensemble Feature Selection for Multi-Stream Automatic Speech Recognition

by

David Gelbart

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Nelson Morgan, Chair

Multi-stream automatic speech recognition (ASR) systems consisting of an ensemble of classifiers working together, each with its own feature vector, are popular in the research literature. Published work on feature selection for such systems has dealt with indivisible blocks of features. I break from this tradition by investigating feature selection at the level of individual features. I use the OGI ISOLET and Numbers speech corpora, including noisy versions I created using a variety of noises and signal-to-noise ratios. I have made these noisy versions available for use by other researchers, along with my ASR and feature selection scripts.

I start with the random subspace method of ensemble feature selection, in which each feature vector is simply chosen randomly from the feature pool. Using ISOLET, I obtain performance improvements over baseline in almost every case where there is a statistically significant performance difference, but there are many cases with no such difference.

I then try hill-climbing, a wrapper approach that changes a single feature at a time when the change improves a performance score. With ISOLET, hill-climbing

gives performance improvements in most cases for noisy data, but no improvement for clean data. I then move to Numbers, for which much more data is available to guide hill-climbing. When using either the clean or noisy Numbers data, hill-climbing gives performance improvements over multi-stream baselines in almost all cases, although it does not improve over the best single-stream baseline. For noisy data, these performance improvements are present even for noise types that were not seen during the hill-climbing process. In mismatched condition tests involving mismatch between clean and noisy data, hill-climbing outperforms all baselines when Opitz's scoring formula is used. I find that this scoring formula, which blends single-classifier accuracy and ensemble diversity, works better for me than ensemble accuracy as a performance score for guiding hill-climbing.

Professor Nelson Morgan
Dissertation Committee Chair

Contents

Acknowledgements

This work would not have been possible without the direct and indirect aid I received from so many others. I am very grateful for the constant personal support I received from my family and friends. The graduation party I threw in Vancouver was one way for me to say thank you. Intellectually my work owes a deep debt to my advisor, Nelson Morgan, as well as Alexey Tsymbal, David Opitz, and Aldebaro Klautau. The International Computer Science Institute has been a great environment for research and I have been helped with technical problems more times than I can count. I would like to particularly thank Barry Chen, Chuck Wooters, Andreas Stolcke, Adam Janin, Arlo Faria, Luke Gottlieb, Stephane Dupont, Lara Docio, Guenter Hirsch, Werner Hemmert, Marcus Holmberg, and Huan Wang. Finally, I want to express my thanks for the financial support I received from the SmartKom and SmartWeb projects as well as from Infineon Technologies.

Chapter 1

Introduction

Contents

1.1 Motivation

1.1.1 Multi-stream automatic speech recognition

Automatic speech recognition (ASR), the automatic mapping of speech to the corresponding text, has progressed from isolated-word digit recognition in the 1950's to today's speaker-independent, large-vocabulary continuous speech recognizers. However, there is still a painful gap between ASR accuracy and human speech recognition

performance [1], especially for more difficult speech types such as speech in noisy environments [2], accented speech, and conversational speech. The performance gap exists for speech without any semantic content, such as nonsense syllables [1]. This shows that progress in ASR is possible without solving the natural language understanding problem.

The standard approach to ASR uses hidden Markov models (HMMs) [3][4][5][6] which model speech as a sequence of observations which are statistically related to a sequence of hidden states. The hidden states usually represent linguistic phones (in general or in particular contexts) or parts of phones. The probability of a particular observation for a particular hidden state is determined by an acoustic modeling stage, which is usually implemented with Gaussian mixture models (GMMs) or sometimes with another technique such as a multi-layer perceptron (MLP). Each observation represents a short frame of time.

The HMM does not directly model the speech waveform. Instead, the observations it models are the output of a feature extraction process meant to reduce dimensionality and discard irrelevant variation. Feature extraction usually results in some kind of smoothed representation of the short-term (i.e., frame level) speech spectrum. The probabilities estimated by the HMM are used by a decoding (hypothesis search) stage which searches for the most likely sentence, taking into account a dictionary of word pronunciations and tables (referred to as a language model or a grammar) of word transition probabilities. This architecture is illustrated in Figure 1.1.

There has been a wealth of productive research on multi-stream ASR systems. The usual architecture of a multi-stream ASR system is a set of classifiers acting in parallel on the same classification problem, producing parallel streams of classifier output which are then combined. Figure 1.2 shows an example of a multi-stream ASR system. Such an architecture is referred to as an ensemble of classifiers in the

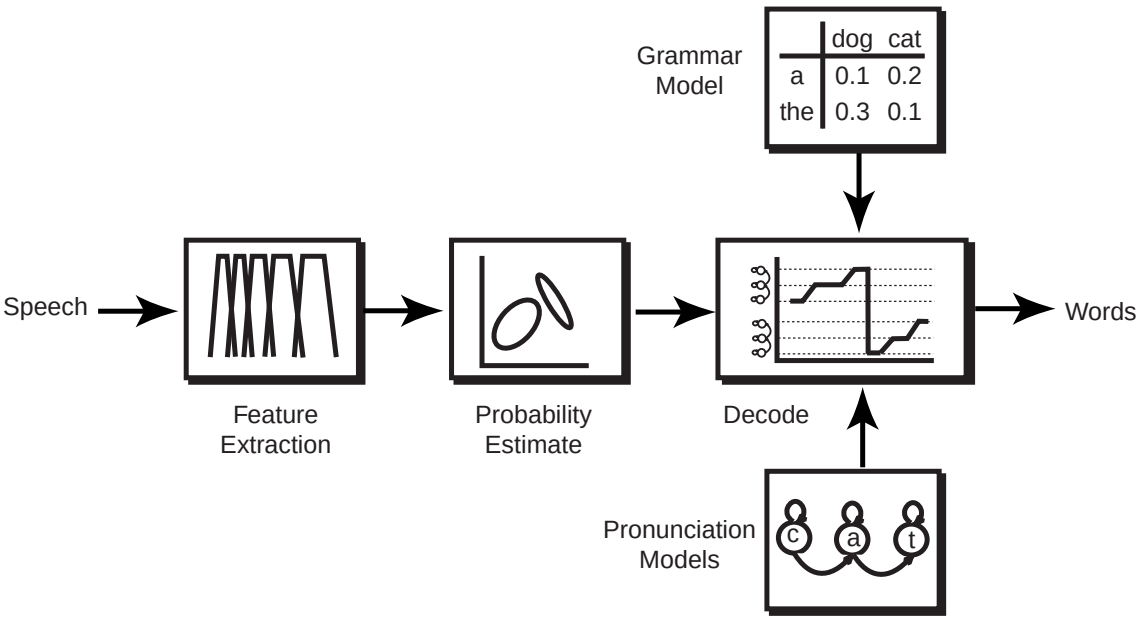

Figure 1.1. A typical ASR system. (This figure is from [7].)

pattern recognition literature [8][9][10]. In this thesis, the terms multi-stream and ensemble will be used interchangeably. A popular approach to differentiating the classifiers from each other in this type of ASR system is to provide a different feature vector to each classifier. This approach will be our sole focus in this thesis.

1.1.2 Feature selection in multi-stream automatic speech recognition

When there are S classifiers in an ensemble and a total of F features available, defining the feature vectors for each classifier involves choosing S subsets of the full set of F features. We will refer to the full set of F features as the feature pool. In the pattern recognition literature, choosing the S subsets of the feature pool is known as ensemble feature selection (EFS) [11][12].

Despite the popularity of multi-stream ASR systems in the research literature, we are aware of no previously published work that tries to do EFS for ASR at the level

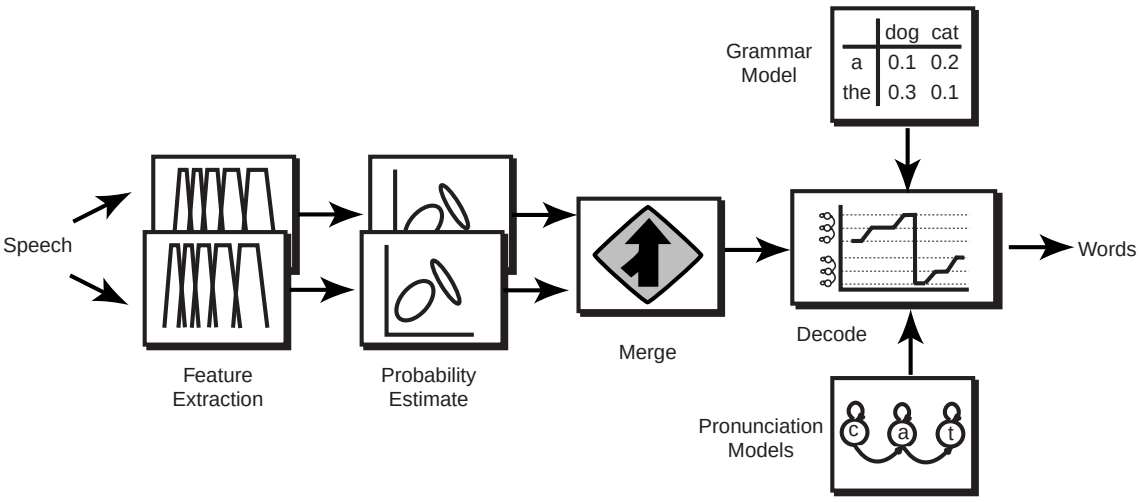

Figure 1.2. A multi-stream ASR system with parallel feature extraction and acoustic modeling stages. (This figure is from [7].)

of individual features. Instead, published work has dealt with indivisible blocks of features rather than individual features [13][14][15][16][17][18][19], so that a feature vector must contain either all the features in a block or none of them. Most often, each block is the entire output of a feature extraction algorithm (for example, MFCC features). In [14], delta features are treated independently of static features, but both delta features and static features are dealt with as blocks. In multi-band approaches [15][20], each block corresponds to a frequency band.

In this thesis, we pursue EFS for ASR at the level of individual features rather than blocks. Doing EFS for individual features means that, for example, if our feature pool contains MFCC and PLP features, then the feature vector of each classifier in an ensemble may contain both some MFCC features and some PLP features, while leaving out other features, and a feature may appear in the feature vectors of more than one classifier. The search space of possibilities is far larger when doing EFS for individual features instead of blocks, and thus it is no longer practical to explore the search space using manual exploration or exhaustive comparisons as in past work. Instead, we use algorithmic EFS strategies from the pattern recognition literature.

The first strategy we use is the random subspace method [21], in which EFS is performed by simply choosing the features in each of the feature vectors at random from the feature pool. RSM is a popular approach which was originally used with decision trees and has also been used with nearest-neighbor classifiers [22], linear discriminant analysis [23], and HMMs using GMM emission probabilities [24].

The second strategy we use is hill-climbing [25]. This is a guided, wrapper [26] method in which initially chosen feature vectors are iteratively improved by adding or removing a single feature at a time to or from a feature vector if the change improves performance. The process is stopped when no more performance improvement can be achieved by the algorithm.

1.2 Goals and Accomplishments

The primary goal of this thesis is to demonstrate that the performance of multi-stream ASR systems can be improved using ensemble feature selection at the level of individual features. Using the random subspace method for the OGI ISOLET task with nine different feature pools, we obtain performance improvements over baseline in almost every case where there is a statistically significant performance difference, but there are many cases with no such difference.

We then try hill-climbing, with which we use only a single feature pool since it is slower than the random subspace method. With ISOLET, for the noisy data hill-climbing gives performance improvements over the starting system in most cases, and the best hill-climbing system for the noisy data is better than the best baseline system, but hill-climbing gives no improvement for the clean data. The results suggest that performance may have been held back by the small amount of data used to guide the hill-climbing process. Therefore, we next evaluate hill-climbing on the OGI Numbers

task, for which much more data is available to guide hill-climbing. When using either the clean or noisy Numbers data, hill-climbing gives performance improvements over multi-stream baselines in almost all cases, although it does not improve over the best single-stream baseline. For noisy data, these performance improvements are present even for those noise types that were not seen during the hill-climbing process. In mismatched condition tests involving mismatch between clean and noisy data, hill-climbing outperforms all baselines when Opitz's scoring formula is used. We find that this scoring formula, which blends single-classifier accuracy and ensemble diversity, works better for us than ensemble accuracy as a performance score for guiding hill-climbing.

The secondary goal of this thesis is to turn the data and code we create along the way into online resources that can productively be reused by other researchers. We created versions of the ISOLET and Numbers corpora that are degraded by background noise, using various noises and signal-to-noise ratios, and we have made it possible for other researchers to exactly reproduce these noisy corpora given copies of the original corpora from OGI. We have also set up ASR systems for ISOLET and Numbers which are built using open source components and are available to other researchers. Our ensemble feature selection scripts are also available online.

1.3 Outline

In Chapter 2, we review ASR, multi-stream ASR, and feature selection in more detail. In Chapter 3, we explain the corpora and tools we use for benchmarking feature selection approaches: the ISOLET and Numbers tasks, our noisy versions of those tasks, and our ASR systems. In Chapter 4, we present our experimental results using RSM for ensemble feature selection for ISOLET. Then, we present our results using hill-climbing feature selection for ISOLET in Chapter 5 and for Numbers in

Chapter 6. Finally, in Chapter 7 we give our overall summary and conclusions and our thoughts on future work.

Chapter 2

Background

Contents

2.1 Introduction

In this chapter, we give an overview of the ideas in automatic speech recognition and feature selection that provide the conceptual background to our work.

2.2 Automatic speech recognition (ASR)

2.2.1 A brief introduction to ASR

This section reviews some key concepts in automatic speech recognition (ASR), drawing heavily on the presentation in [5]. For a deeper introduction, we recommend the excellent tutorials [27][28][29][30][31][6] and textbooks [3][4][5] that are available. We have placed a more complete list of tutorials and textbooks online at `http://www.dev.voxforge.org/projects/Main/wiki/TheoryAndAlgorithms`.

Classification in the simplest case

Consider a classification problem where we have an observation $\vec{x}$ and we want to know what underlying class c_k ($k = 1, 2, ..., K$) produced it. We can relate this to speech recognition by imagining an isolated word speech recognition system in which each class is one of the words to be recognized. The vector $\vec{x}$ represents a sound and is usually referred to as a feature vector. The optimum error-minimizing classification rule [5] is to choose the class k that maximizes $p(c_k|\vec{x})$:

$$k = \underset{m}{\operatorname{argmax}}\, p(c_m|\vec{x}) \tag{2.1}$$

Using Bayes' rule (also known as Bayes' theorem: $p(a|b) = p(b|a)p(a)/p(b)$) we can rewrite the classification rule as:

$$k = \underset{m}{\operatorname{argmax}}\, p(\vec{x}|c_m)p(c_m)/p(\vec{x}) \tag{2.2}$$

$p(c_m|\vec{x})$ and $p(\vec{x}|c_m)$ are commonly referred to as the posterior and the likelihood, respectively. Since $p(\vec{x})$ is the same across classes it can be omitted from the equation. Also, in a practical statistical pattern recognition setting, statistical model parameters

Θ are usually learned from training data. Omitting $p(\vec{x})$ and including the dependence on Θ gives:

$$k = \underset{m}{\operatorname{argmax}}\, p(\vec{x}|c_m, \Theta)p(c_m|\Theta) \tag{2.3}$$

$p(\vec{x}|c_m, \Theta)$ explicitly depends on the acoustic signal, while $p(c_m|\Theta)$ does not. This leads us to split Θ into two models, the acoustic model Θ_A and the language model Θ_L:

$$k = \underset{m}{\operatorname{argmax}}\, p(\vec{x}|c_m, \Theta_A)p(c_m|\Theta_L) \tag{2.4}$$

The acoustic model Θ_A represents the statistical relationship between the acoustic vectors and the classes. In an isolated-word recognizer, the language model Θ_L represents probabilities of particular words, and in a continuous speech recognizer it also represents the probabilities of sequences of words. (The language model is sometimes called a grammar.)

This separation of statistical modeling into acoustic modeling and language modeling is standard in speech recognition technology. This thesis focuses solely on acoustic modeling.

In practice, speech recognition systems are usually based on some kind of subword units, such as linguistic phones or parts of phones, rather than directly classifying entire words. This allows for more efficient use of available training data, since the same subword units can appear in more than one word, and also makes it easier to model variation in word durations.

Acoustic modeling

As discussed above, the purpose of the acoustic model is to model the relationship between feature vectors and the underlying classes through modeling the distribution of the likelihood $p(\vec{x}|c_m, \Theta_A)$. This is challenging as there is an enormous amount of variability in how a given word may sound, due to variation in speaker physiology, speaking style, mood, accent, and other factors [32].

The most popular acoustic modeling approach is the Gaussian mixture model (GMM). In the GMM approach, the distribution is modeled by a weighted mixture of multivariate Gaussian distributions. Usually, the Gaussian distributions are restricted to have a diagonal covariance matrix, in order to reduce the number of model parameters. A separate GMM can be used for each subword unit being modeled, but often state tying approaches are used in which GMM parameters are shared between related subword units. This further reduces the total number of parameters and thus reduces the amount of training data needed to obtain reliable parameter estimates.

An alternative acoustic modeling approach uses the multi-layer perceptron (MLP) [33][34][35], which is a feedforward neural network with an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.1. The parameters of the MLP are the node-to-node connection weights ω_j and the node biases β_k. The current feature vector is fed in at the input layer, often with context provided as well by also feeding in preceding and following feature vectors. Usually a single MLP is used to model all classes, providing class posterior probabilities at the output layer. Given an appropriate training procedure [33][34][35], an MLP that has an output node for each class and is trained with binary class labels as targets (equal to 1 for the labeled class of the current feature vector and 0 for the other classes) can be used to estimate class posterior probabilities $p(c_m|\vec{x})$.

Since a hidden Markov model relates classes to feature vectors through likelihoods,

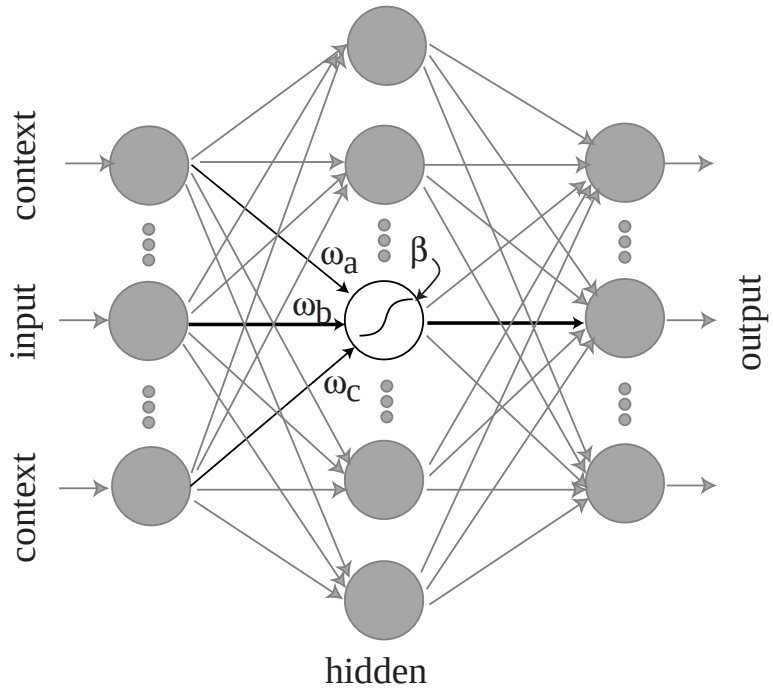

Figure 2.1. A multi-layer perceptron with a single hidden layer. (This figure is from [7].)

the posterior probabilities calculated by the MLP are divided by class prior probabilities $p(c_m)$ before decoding. To obtain likelihoods, Bayes' rule actually also requires multiplying by $p(\vec{x})$, but since $p(\vec{x})$ is the same across classes it can be omitted without affecting the outcome of classification.

Sequence classification

To move from isolated-word recognition to recognition of continuous speech, or to perform isolated-word recognition using models built from subword units, we need a way to recognize sequences of classes. The standard means of doing so is the hidden Markov model (HMM). An HMM models speech as a sequence of observations (feature vectors) which are statistically related to a sequence of discrete hidden states (classes). The term hidden is used because the states are not observed directly, but instead have a probabilistic relationship with the observations which is represented by the acoustic

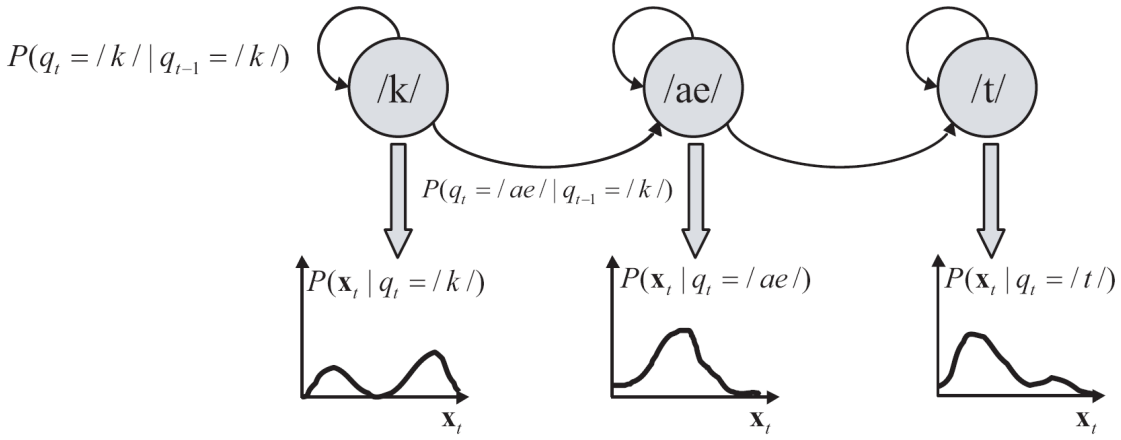

Figure 2.2. A hidden Markov model of the word "cat" for speech recognition. The subscripts t and $t-1$ refer to the current time step and the previous time step. (This figure is from [36].)

model. The HMM also represents state-to-state transition probabilities, which are based on the language model, a word pronunciation dictionary, and a state duration model.

Figure 2.2 shows an HMM for the word "cat" in which the states q_t are phonemes. As shown in the figure, it is possible for a state to transition to itself. By looping back to the same state, state duration can be extended to more than one time step, which allows the HMM to model variability in state durations.

To recognize speech, a decoder is used to search for the most probable sequence of spoken words given the trained HMM. The term decoder comes from the use of related techniques in communications technology to process error correcting codes. The overall architecture is illustrated in Figure 1.1.

Feature extraction

Speech waveforms are very variable and high dimensional and so are impractical to model directly with current acoustic modeling techniques. The main goal of ASR feature extraction is to reduce the amount of non-lexical variation, thus making acoustic

modeling less demanding, while at the same time preserving key lexical information. The standard approach is to divide the waveform into overlapping, fixed-length frames which are then converted into some kind of smoothed spectral representation. Frames are commonly around 25 ms long with about 10 ms between the start of adjacent frames.

The most popular ASR feature extraction methods are Mel-Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP) [37]. Both start with the power spectrum or magnitude spectrum of the current frame, and then reduce the dimensionality by convolving the spectrum with a set of overlapping critical band filters. These filters are wider at high frequencies, as with critical bands in human hearing. In MFCC the log of the filter bank outputs is then taken. In PLP, the filter bank outputs are weighted by an equal-loudness-perception curve based on human hearing (in MFCC a related preemphasis step may be applied at the very beginning), and the cube root is then taken. For MFCC, the next step is to apply a DCT to produce cepstral coefficients, discarding the higher-order cepstral coefficients (which correspond to finer details in the spectrum) in order to obtain a representation corresponding to a smoothed spectrum. For PLP, the next step is to perform linear prediction (based on autocorrelation coefficients computed from the filter bank outputs) and then calculate cepstral coefficients based on the linear prediction coefficients. The resulting representation corresponds to a spectrum with a number of peaks depending on the order of the linear prediction. This linear prediction approach to smoothing tracks spectral peaks in the original spectrum more closely than spectral valleys.

The dynamics of speech carry a great deal of information, and it is popular to augment MFCC and PLP feature vectors with time derivatives. Thus, a thirteen-dimensional feature vector might be augmented with a thirteen-dimensional vector containing the velocity of those features and another thirteen-dimensional vector con-

taining the acceleration. These three blocks of features are commonly referred to as the static, delta, and delta-delta features respectively.

To add diversity to our feature pool, in this thesis we also make use of Modulation SpectroGram (MSG) features [38]. Slower temporal modulations in speech are less influenced by reverberation, but have been found to carry substantial speech information in experiments with human listeners. The key idea behind MSG is thus to calculate features based on temporal modulations below 16 Hz in the outputs of critical band filters. The modulation-filtered features are then processed by an auditory-inspired automatic gain control mechanism. MSG features were designed for reverberant speech but have been found useful for speech in background noise as well.

2.2.2 Multi-stream ASR

Multi-stream systems (also known as ensembles) that have been successfully used in ASR research include systems that combine word hypotheses [39][40][41][42][13][43], systems that combine probabilities at the hidden Markov model (HMM) state level [44][45], systems that combine HMM state probabilities at the phonetic segment level [46][47], and systems that combine HMM state probabilities at the frame level [14][48][15][49][50][20]. In this thesis, we will always combine HMM state probabilities at the frame level, as illustrated in Figure 1.2. However, the feature selection techniques we work with can be used with other combination styles.

Both the accuracy of individual classifiers in the ensemble and the diversity of the overall ensemble contribute to ensemble performance. Diversity, or in other words disagreement between members of the ensemble, is necessary for an ensemble to perform better than its best individual member. A popular way of creating diversity in ASR is to use a different feature vector with each classifier. Some such

systems [51][52][53][15][20] have split the spectrum into frequency bands and used different classifiers for different bands. Other systems have combined different feature extraction methods, each of which could conceivably have been used on its own [48][15][49][50][54], and this approach will be our focus in this thesis.

2.2.3 Why we use multi-layer perceptrons (MLPs)

In this thesis, we use a "hybrid connectionist" speech recognition approach [33][34], in which MLPs are used for acoustic modeling within an HMM. In fact, Gaussian mixture model based systems are more popular than MLP based systems. The reason we chose MLPs is that they sometimes handle novel feature types better than GMMs do. In work outside of this thesis, we explored this point using novel auditory-based features [55]. In this thesis, our feature pool contains MFCC, PLP and MSG features. MFCC and PLP actually have a solid track record in GMM systems, but in [56][57][58] MSG features were found to perform better with MLPs than with GMMs, even when decorrelation and Gaussianization techniques were tried in an effort to improve GMM performance. Furthermore, it seemed possible to us that even the MFCC and PLP features would become harder for the GMM to model once they were mixed together into new feature vectors by the feature selection process.

Our results using MLPs are made more relevant to researchers using GMM-based systems by past work on the "tandem" [59][60][49] approach, which can be used as a bridge between these two paradigms. In the tandem approach, an MLP or an ensemble of MLPs is used as a non-linear discriminant analysis stage prior to GMM acoustic modeling. A number of successes with novel feature vectors have been reported using the tandem approach [56][49][61][62].

2.2.4 How we perform stream combination

In this thesis, we combine the outputs of the MLPs in an ensemble by combining posterior probabilities at the frame level. We do this by taking the geometric mean of the posterior probabilities for each phone across MLPs. For numerical reasons, we calculate this by taking an arithmetic mean of logarithmic probabilities. This is a simple and effective method for stream combination [54][14]. There are other MLP combination approaches, such as inverse entropy weighted combination [63][64] and combination using Dempster-Shafer theory [50]. We did not try those since comparing different combination techniques is not a focus of this thesis.

2.3 Feature selection

We use the term feature selection to refer to selecting a subset or subsets from a set of features. The features are then used for classification by a machine learning algorithm. Most commonly, feature selection is performed for a single classifier, but this thesis focuses on ensemble feature selection in which a subset must be chosen for each classifier in an ensemble.

It can be useful to view feature selection algorithms as being "wrappers" or "filters" [26]. A wrapper approach evaluates a feature subset under consideration by simply measuring the classification performance of the learning algorithm using those features. The name wrapper reflects the fact that the learning algorithm is used as a black box component of the feature selection algorithm, meaning that the feature selection algorithm does not incorporate knowledge of the model built by the learning algorithm, other than its classification performance. A filter approach selects features based on the data, without using the learning algorithm that will eventually be used for classification. The name filter reflects how the feature selection algorithm is used

as a preprocessing step to filter out unwanted features prior to the use of the learning algorithm.

From our perspective as ASR researchers, the advantage of a wrapper approach is its natural connection to the classification task, but the disadvantage is the need to repeatedly train and test acoustic models, which can be very time consuming. Filter approaches appeal because of their potential for greater speed, but cleverness is required to make them relevant to the ASR task that the features will eventually be used for. Feature selection approaches for ASR can combine aspects of both wrappers and filters by defining objective functions which incorporate knowledge of the acoustic model but can be optimized without repeatedly training and testing new acoustic models (see the discussion of [65][66] below).

Some of the literature on non-ensemble feature selection methods is surveyed in [26][67][68]. We discuss ensemble feature selection literature further below.

2.3.1 Non-ensemble feature selection for ASR

The vast majority of past work on feature selection for ASR has dealt with non-ensemble ASR systems. In order to put our work into context, in this section we survey some of the published literature on non-ensemble feature selection for ASR.

The most commonly used approaches perform some kind of transform of the features that makes feature selection straightforward in the transformed feature space. Principal components analysis [69], also known as the Karhunen-Loève transform (KLT), is one such approach. The PCA approach is a matrix transform which projects the features into a new feature space determined by the eigenvectors of the feature covariance matrix. When appropriate normalization is used, the eigenvalues represent the amount of data variance accounted for by each dimension in the new feature space. This leads to a simple feature selection approach: remove dimensions which represent

small amounts of variance. This approach has been used in ASR research for decades [70].

The discrete-cosine transform (DCT) is a related technique. Under certain statistical assumptions, it is an approximation to the KLT [71]. An implicit, DCT-based feature selection operation (i.e., leaving higher-order cepstral coefficients out of the final feature vector) is a standard step in the computation of MFCC features.

Linear discriminant analysis (LDA) [69][72][73], which is designed to preserve information that is useful for class discrimination, is another popular transform approach. The LDA transform is chosen to maximize the ratio of a measure of between-class variance to a measure of within-class variance. While LDA is often viewed as a single, dimensionality-reducing transform operation, it can also be viewed as a dimensionality-preserving transform followed by a feature selection step which, as for PCA, is based on choosing the eigenvectors corresponding to the largest members of a set of eigenvalues [72]. LDA has been criticized for embodying an assumption that different classes have identical covariance statistics [72], and this has been addressed by generalizing LDA to heteroscedastic discriminant analysis (HDA) [72][74].

We are not aware of any other ASR feature selection techniques that approach the popularity of PCA, DCT, LDA and HDA. However, a number of interesting ideas have been proposed.

In the Feature-Finding Neural Network (FFNN) approach [62][75][76][77], a simple linear perceptron classifier was used to evaluate the performance of many different feature vectors in order to select a feature vector from a very large, parametrized pool of possible feature vectors. The selected feature vector was then shown to perform well when used with an MLP. Compared to trying out all the different feature vectors with an MLP, the speedup from using a linear perceptron was immense since training and testing the linear perceptron only requires a few matrix operations. The search

process started off with an initial feature vector of length L and then repeatedly (1) identified the least useful feature in the current feature vector and (2) replaced the least useful feature with a randomly chosen new feature from the pool. Identifying the least useful feature was done by measuring classification accuracy for all the possible feature vectors resulting from removing a single feature from the current feature vector. The feature whose removal reduced accuracy the least was judged the least useful. Source code for FFNN is available at [78]. We did not use FFNN in our work, but it may have potential for use in ensemble feature selection if it can be adapted for use with ensembles. There is published work on ensembles of linear perceptrons (outside the ASR field) that might be relevant.

In Klautau's approach [79], phonetic classification was divided into a large number of binary classification problems, each of which was handled by its own linear support vector machine. A very large feature pool was constructed using seven feature extraction algorithms (all of which can be carried out with tools that are freely available online) and feature selection was performed separately for each binary problem. Two different feature selection algorithms were tried, one based on mutual information between labels and features, and the other inspired by the use of AdaBoost for feature selection in computer vision [80].

Kirchhoff et al. [65][66] started with 65 features and removed one feature at a time until they reached the desired number of features, 39. At each step, they chose the feature to be removed by considering all the possible feature vectors resulting from removing a single feature, and picking the one that maximized a discriminative objective function. The objective function measured between-class discrimination using a formula based on summing (over frames) the distance between acoustic model probabilities for the HMM state assumed to be correct (which was chosen using a labeling created by forced alignment) and other HMM states. The acoustic models for the reduced-dimensionality feature vectors were created by simply deleting the

appropriate dimensions from the model means and covariance matrices. This is an interesting way to avoid spending time retraining models during feature selection, and perhaps it could be extended to ensemble feature selection by using post-stream-combination probabilities. They used Gaussian mixture models and it is not clear to us whether there is any comparable way to avoid retraining in MLP-based ASR.

Su and Lee [81] used feature selection as part of their strategy to add a second, discriminant classification stage to a conventional ASR system. They used a conventional HMM to create a 45-dimensional feature vector where each element was an averaged HMM state log-likelihood score. This vector was then used by a second stage of classification, a discriminant word classifier. To improve accuracy, they reduced the size of this feature vector by performing feature selection using a divergence measure designed to measure the usefulness of each feature for class discrimination. The test data recognition accuracy, as a function of the number of features kept, peaked at 13 features. This is much smaller than the feature vectors usually used in ASR. Perhaps this was due to the unconventional nature of the features, but it might also be because of the use of a small training set (900 words), which may have made the curse of dimensionality a bigger factor than usual. A divergence-based approach was also used in [82].

Valente and Wellekens [83] implemented feature selection by model selection in the context of Gaussian mixture models. Perhaps this could be made into an ensemble feature selection method by using post-stream-combination probabilities, as in [44][45]. Valente and Wellekens assigned a relevancy to each feature, and their results show that there can be considerable variation in relevancy between individual features calculated by the same feature extraction method (for example, between different delta-MFCC features).

Kommer and Hirsbrunner [84] used a genetic algorithm wrapper approach for

feature selection from a pool of possible wavelet front ends. As with our work in Chapters 5 and 6, this paper shows that wrapper approaches in speech recognition are slow but not too slow to be feasible.

Missing feature approaches to ASR try to identify what features have been rendered unreliable by noise (or other degradations) at a given point in time. Because the selection is time dependent, this is not feature selection in the sense we use the term in this thesis. These approaches are nonetheless interesting and they are surveyed in [85].

Much of the published work on non-ensemble feature selection for ASR reports little or no recognition accuracy gains from feature selection. In fact, many of these papers state their goal as reducing storage and computation overhead by reducing the number of features, rather than improving recognition accuracy. We speculate that accuracy gains from feature selection are easier to achieve with an ensemble system than with a non-ensemble system because in the ensemble case there are more factors that can be exploited to improve the performance of the overall system: in addition to the factors that apply in both cases, such as the curse of dimensionality [69], EFS is also optimizing for the best combination of single-classifier accuracies and ensemble diversity. This is only speculation, as we have not done a controlled experimental comparison or a rigorous theoretical analysis.

Regarding accuracy gains, another relevant point is that few papers on single-classifier feature selection for ASR have used more than 78 features. It would be interesting to see how well those approaches perform with many more features in the pool, as in [79][17][13]. Perhaps accuracy gains would be easier to achieve due to a larger effect from the curse of dimensionality.

2.3.2 Ensemble feature selection in ASR

In this section, we review published work on ensemble feature selection for ASR. As we discussed in Section 1.1.2, the published work we are aware of has performed selection at the level of blocks of features rather than individual features.

Using three feature types in a multi-band system with four bands, Christensen et al. [15] exhaustively tested all $3^4 = 81$ possible placements of feature types in bands to find out which performed the best. They then tested all possible combinations of the three full-band feature streams.

Ellis and Bilmes [16] took an information theory approach, investigating the usefulness of conditional mutual information (CMI) as a guide to which streams will be useful together in an ensemble. CMI between classifier outputs was used as a measure of diversity (with a lower CMI meaning higher diversity), and was useful for predicting which pairs of streams would combine well. They also investigated using CMI between feature vectors as a guide to what combination method to use, but found this to be less effective, perhaps because of the simplifications they used when computing CMI.

Burget [17][13] defined dependent word error rate (DWER) as the number of identical word recognition errors that are made by two systems, and used it as a measure of diversity. He found that low DWER was an indicator of what systems contributed well to an ensemble, and high DWER was an indicator of what systems were better left out of the ensemble. Related work on "Identical-Incorrect" errors was presented in Chapter 6 of [86]. In [13] Burget also investigated a number of other, related diversity measures.

Li and Stern [44][45] designed a feature transformation (rather than feature selection) approach for ensemble ASR. Starting from 120 features, they generated two

complementary feature streams with 39 features each by using matrix transformations that maximized (for training data) the post-stream-combination normalized acoustic likelihood of the most likely state sequence. They used a two-classifier ensemble, but their approach could be generalized to larger ensembles.

2.3.3 Ensemble feature selection with individual features

Since the published work we know of on EFS for ASR has worked with blocks of features rather than individual features, we turned to the general literature on pattern recognition to find methods for EFS at the level of individual features. We discuss some examples here. Our discussion is heavily influenced by [25], which gives a more detailed survey.

In the random subspace method [21], EFS is performed by simply choosing the features in each of the feature vectors at random from the feature pool. As originally defined by Ho [21], the feature vectors all have the same length, but some other authors [12][87][25] have chosen different lengths randomly for each feature vector. We picked the random subspace method for initial experiments in this thesis because it is simple, popular and has often been found effective. For details on our implementation, see Section 4.1.

Opitz [12] proposed the GEFS (Genetic Ensemble Feature Selection) genetic algorithm for ensemble feature selection. This is a wrapper method which he found was usually able to improve performance over the random subspace method, which he used for the initial assignment of features to classifiers. GEFS maintains a population of candidate feature vectors represented as variable-length strings of feature indices. New members of the population are created from the existing strings by crossover (breeding) and mutations. Classifiers are trained on the features specified

by each string in the population. Each string is then scored on its usefulness and the least useful strings are pruned out of the population. The process then repeats.

An important difference between GEFS and other early work on genetic algorithms for ensemble feature selection [88][89][90] is that GEFS scores a candidate feature vector using a weighted combination of the accuracy of a single classifier using those features and the contribution of that classifier to the diversity of the ensemble of classifiers. This is Equation 5.1, which we discuss further in Section 5.5. In contrast, candidate feature vectors were scored by single-classifier accuracy in [89][90] and by ensemble accuracy in [88]. The authors of [88] mentioned that overfitting was sometimes a problem and suggested the inclusion of a diversity term in the scoring formula as a response.

Equation 5.1 is presumably still prone to overfitting of individual classifiers, but it may have an advantage in that an ensemble of overfitted classifiers is not necessarily an overfitted ensemble. In fact, some work [91][92][93][94] suggests that overfitting of individual classifiers may actually help ensemble performance in some situations. Tsymbal et al. [25] made several modifications to GEFS, but kept Equation 5.1 as the scoring function.

Cunningham and Carney [95] proposed a hill-climbing wrapper algorithm for ensemble feature selection. Their hill-climbing algorithm started with randomly chosen feature vectors and then iteratively improved them by adding or removing a single feature at a time to or from a feature vector if the change improved the performance score. The process was stopped when no more performance score improvement could be achieved by the algorithm. The performance score for a candidate feature vector was the single-classifier accuracy for that feature vector. However, in their conclusions Cunningham and Carney criticized this scoring function, since it does not take into account the diversity of the ensemble, and they suggested that "any work with

classification ensembles should explicitly measure diversity in the ensemble and use this measure to guide decisions on the constitution of the ensemble." This theme was explored further in [96]. Tsymbal et al. [87][25] modified the hill-climbing algorithm from [95] to use Equation 5.1 to measure performance, and found that this indeed improved the accuracy of the final ensemble.

In [25], Tsymbal et al. compared the performance of their genetic algorithm, their hill-climbing algorithm, the random subspace method, and two other wrapper EFS approaches (Ensemble Forward Sequential Selection and Ensemble Backward Sequential Selection). This is the most comprehensive comparison of EFS methods we are aware of. The best performance was obtained with the genetic algorithm and the second best performance was obtained with the hill-climbing algorithm. Since ASR corpora are much more time-consuming to work with than the small corpora used in [25], we decided to investigate just one wrapper method in this thesis. We picked hill-climbing, since it is simple and its performance in [25] was almost as good as the more complex genetic algorithm approach. We try both ensemble accuracy and Equation 5.1 as performance scores. For details on our implementation of hill-climbing, see Section 5.5.

Chapter 3

Our benchmarks

Contents

3.1 Introduction

In this chapter we explain the ASR benchmarks we created for our research. We initially used the OGI ISOLET [97] corpus and then switched to the OGI Numbers [98] corpus. We created noisy versions of these corpora using various noise types and signal-to-noise ratios. These noisy versions are available to other researchers. We also created ASR systems for these corpora. These ASR systems are not tied to ensemble feature selection. They are based on open source components and are available to other researchers. There are two ISOLET ASR systems, one based on multi-layer perceptrons using the Quicknet toolkit [99] and the other based on Gaussian mixture models using the HTK [100] toolkit. For Numbers, there is one ASR system based on multi-layer perceptrons.

3.2 How we make our noisy versions of copyrighted corpora available to other researchers

Due to OGI's copyright on the ISOLET and Numbers corpora, we could not simply place our noisy versions online. Instead, we placed scripts[1] and noise recordings online that others can use to exactly reproduce the noisy speech corpora given copies of the original speech corpora which they have obtained from OGI. We were granted

[1]The scripts use the FaNT [101] noise-adding tool. FaNT's author, Prof. Hans-Guenter Hirsch, graciously added new features that we required.

permission to share the noise recordings online, but this approach would still have worked if the noise recordings needed to be purchased separately.

Authors of ASR research papers dealing with additive noise or reverberation often start with a publicly available, copyrighted corpus and then artificially add noise or reverberation, resulting in a corpus that is no longer publicly available. Thus the model of corpora sharing we followed for our noisy ISOLET and Numbers corpora could be useful for other corpora as well.

3.3 The OGI ISOLET corpus

The ISOLET [97] corpus consists of recordings of the letters of the English alphabet, spoken as isolated words in a quiet environment, using a sampling rate of 16000 Hz with a Sennheiser HMD 224 microphone. (The HMD 224 is a close-talking headset microphone, according to descriptions in various papers that use this corpus.) The recordings contain about 1.25 hours of audio. There are a total of 7800 words in the corpus, produced by 150 speakers each speaking each letter of the alphabet twice.

The ISOLET corpus comes divided into five parts each containing 1560 words. Our ASR systems for ISOLET train on four of the five parts and test on the remaining one part. There are five possible ways to do this, according to which of the five parts is chosen for testing. Using four parts to train and one part to test, about 60 minutes of audio is used for training and about 15 minutes of audio is used for testing. Some of the audio consists of before-word and after-word pauses, leaving about 45 minutes of speech for training and about 11 minutes of speech for testing.

3.4 Our noisy version of the ISOLET corpus

Since speech recognition in noisy environments is an area of current research interest, we created a noisy version of the ISOLET corpus in which one of eight different noise types was artificially added to each utterance at one of six different A-weighted signal-to-noise ratios: no noise added, 20 dB, 15 dB, 10 dB, 5 dB, and 0 dB. We used A-weighting for the reasons discussed in [102]. We will refer to this version of ISOLET as noisy and to the original ISOLET data as clean.

The noise types, taken from the RSG-10 [103] collection, were pink noise, speech babble, two types of car factory noise, car interior noise, Leopard military vehicle noise, fighter jet cockpit noise, and naval vessel operations room noise. (RSG-10 is the same collection that provided the noises for the NOISEX-92 corpus. Some of the ASR literature apparently refers to these noises under the name NOISEX-92 rather than RSG-10.) Three of the noise types were used for all five parts of ISOLET, and the remaining five noise types were used for one part each. This means that for each of the five different divisions of ISOLET into training and test data that we use, there are three matched noise types that are found in both the training and test data, one noise type found only in the test data, and four noise types found only in the training data.

The noises were added to ISOLET utterances starting at random starting points within the noise recordings. A different random starting point was picked for each ISOLET utterance (the noise recordings are much longer than the individual ISOLET utterances). The scripts we provide to allow other researchers to recreate our noisy ISOLET corpus replicate the same random starting points.

The way we designed the noisy version of the ISOLET corpus (using various noises at various SNRs, with both matched and mismatched noise types) was heavily influenced by the popular Aurora 2 ASR benchmark [104]. Unlike Aurora 2, with

ISOLET it is easy to make a distinction between development data used to tweak system parameters and evaluation data used to report final results, since there are five possible ways to divide the corpus into training and test data and one of the five can be used as development data. Making this distinction was important to us because in our work with the hill-climbing ensemble feature selection algorithm we had to separate the development data we used to guide hill-climbing from the evaluation data that we reported our final results on, since otherwise our experimental results might have been considerably biased [105].

Scripts and noise recordings that others can use to make their own copy of the noisy version of ISOLET, as well as some additional details about the noisy version, can be found at `http://www.icsi.berkeley.edu/Speech/papers/eurospeech05-onset/isolet`. Our noisy version of the ISOLET corpus has been used in work outside of this thesis [55][106].

3.5 Our ISOLET ASR system based on multi-layer perceptrons

We created scripts and configuration files for performing speech recognition on the ISOLET corpus (both the original and our noisy version) with multi-layer perceptrons using the Quicknet toolkit [99] and the noway decoder [107].

There are five parts to the ISOLET corpus, and thus five different ways to divide the corpus into four training parts and one testing part. When we report experimental results on ISOLET, we will refer to each of these possible divisions of the data as a fold of the data, with fold 1 referring to the division in which the first of the five parts is used for testing, fold 2 referring to the division in which the second of the

five parts is used for testing, and so on. Thus training and testing can be run five times, increasing the effective amount of test data.

We reserved fold 1 as development data for tuning decoder parameters (and for guiding hill-climbing in our ensemble feature selection experiments) and used folds 2-5 as evaluation data for reporting performance results. Thus our development data results are for 1560 test words and our evaluation data results are for 6240 test words. Fold 1 was used to tune three parameters of the decoder (the acoustic model/language model match factor, duration scale factor, and phone exit transition probability scale factor) using a grid search.

An MLP with I input units, H hidden units, and O output units has $(I+O)*H$ weights and $H+O$ biases, for a total of $(I+O)*H+H+O$ acoustic model parameters. In this thesis, we always presented the MLPs with five frames of feature context (two previous frames, one current frame, and two future frames) so the number of MLP input units was always five times the length of the input feature vector. For example, for a 39-dimensional MFCC feature vector, there were $5 * 39 = 195$ input units. The number of MLP output units was set at 28, corresponding to 28 phones in the ISOLET corpus (including a silence "phone").

Our scripts reserve about 10% of the training data as a cross-validation set for early stopping to guard against overfitting during training.

The scripts support using an ensemble of MLPs, using a different feature vector for each MLP. The ensemble is combined by taking the geometric mean of posterior probabilities across streams (see Section 2.2.4).

Our scripts perform decoding using the noway decoder [107], which is open source and available online [99]. Each phone P is represented by N_P left-to-right HMM states (with no skips), where N_P is set to determine the minimum phone duration. The emission probability estimates for all states of a phone were tied to the same

monophone (and these probabilities were taken to be the scaled likelihoods obtained by dividing the MLP-estimated phone posterior probabilities by phone prior probabilities). After we completed the experiments in this thesis, we noticed we could improve the recognition accuracy of our scripts by making a change to how we handled duration modeling. For more information on this, see the README_DURATION file we have included in the documentation for our scripts.

For the results in this thesis, we ran this ASR system using Quicknet version 3.11 and noway version 2.9. So that our MLP trainings would be as consistent as possible, we always used the same version (Pentium 4 SSE2) of the Quicknet training tool qnstrn for our trainings.

We also created scripts that can be used to test whether ASR performance differences are statistically significant, using McNemar's test. We chose McNemar's test because of the suggestion in [108], a classic paper on significance testing in ASR research, to use McNemar's test for ASR experiments on corpora containing isolated words.

The scripts, configuration files, and documentation for our MLP-based ISOLET ASR system can be downloaded at `http://www.icsi.berkeley.edu/speech/papers/eurospeech05-onset/isolet`. A README file is included which gives some additional details about the ASR system and provides a tutorial with examples of how to use the scripts. This ASR system has been used in work outside of this thesis [55].

3.6 Our ISOLET ASR system based on Gaussian mixture models

We have also made an ASR system available for our clean and noisy ISOLET corpora which use Gaussian mixture models (GMMs) for acoustic modeling. This was built using the popular HTK [100] toolkit. It was originally created by Montri Karnjanadecha and enhancements were made by Chuck Wooters, Prof. Hans-Guenter Hirsch, and ourselves.

The GMM-based system can be downloaded at `http://www.icsi.berkeley.edu/speech/papers/eurospeech05-onset/isolet`. It is not used in this thesis. In [55], we compared the performance of the MLP-based system and the GMM-based system for several feature types, using our noisy ISOLET corpus. The GMM-based system is also being used in [106].

3.7 The OGI Numbers corpus

The Numbers corpus [98] consists of strings of spoken numbers collected over telephone connections. The sampling rate is 8000 Hz. Some researchers use the name Numbers95 rather than Numbers; this is more common when referring to an earlier version of the corpus.

We began using the Numbers corpus because we wanted a corpus that was larger than ISOLET. We used version 1.3 of the corpus, which was released in 2002 and was also used in [109] and [110]. Our research group previously made heavy use [7][86][38][36] of earlier versions of the corpus. We switched to version 1.3 since it is much larger.

The authors of [109] selected the utterances containing only the 30 most frequent

words, excluded utterances containing truncated words, and divided the remaining utterances into $\frac{3}{5}$ training set, $\frac{1}{5}$ "validation" set, and $\frac{1}{5}$ "test" set. We did the same thing, using the same utterance lists that were used in [109], which the authors shared with us. These three sets respectively contain about 6 hours, 2 hours and 2 hours of audio. We used [109]'s "validation" set as development data and their "test" set as evaluation data (see Section 3.4). For the rest of this chapter and in following chapters, we will refer to those two sets as the development set and evaluation set.

Past users of the Numbers corpus in our research group often added 100ms of padding to the start and end of the audio files (for example, to provide more time for filter warmup in feature extraction). For simplicity, we did not do this.

3.8 Our noisy version of the Numbers corpus

As with ISOLET, we created a noisy version of the Numbers corpus using noises from the RSG-10 [103] collection. Since the Numbers data was collected over telephone channels and the noises were not, we used FaNT to apply the MIRS telephony filter from the ITU Software Tools Library [111][112] to the noises before adding them to Numbers data. Our intent was to make the noises more like noises that had been collected over telephone channels. We then added one of each of ten different noise types to each utterance at one of six different non-frequency-weighted signal-to-noise ratios: no noise added, 20 dB, 15 dB, 10 dB, 5 dB, and 0 dB. (We did not use frequency weighting in the SNR calculation because the audio was already filtered to telephony bandwidth.) We will refer to this version of Numbers as noisy and to the original Numbers data as clean.

The noises used with the training set were speech babble, factory floor noise type 1, car interior noise, F-16 cockpit noise, factory floor noise type 2, and Buccaneer

cockpit noise at 190 knots. The noises used with the development set were speech babble, factory floor noise type 1, car interior noise, F-16 cockpit noise, M109 tank noise, and Buccaneer cockpit noise at 450 knots. The noises used with the evaluation set were speech babble, factory floor noise type 1, car interior noise, F-16 cockpit noise, destroyer operations room noise, and Leopard military vehicle noise. So the first four noise types were used for all three sets (training, development, evaluation) and the other noise types were only used in one set. That means that there is a mix of matched noise types (found in both the training set and the test sets) and mismatched noise types (only found in one set).

As with our noisy version of ISOLET, the design of our noisy version of Numbers is heavily influenced by the Aurora 2 benchmark [104], but unlike Aurora 2 there is a distinction between development and evaluation data.

For scripts and noise recordings that others can use to make their own copy of the noisy version of Numbers, as well as some additional details about the noisy version, see `http://www.icsi.berkeley.edu/Speech/papers/gelbart-ms/numbers`. An earlier version of our noisy Numbers corpus, based on version 1.0 of the Numbers corpus rather than version 1.3, was used in [113].

3.9 Our Numbers ASR system based on multi-layer perceptrons

As we did for ISOLET, we created scripts and configuration files for performing speech recognition on the Numbers corpus (both the original and our noisy version) using MLPs, based on the Quicknet and noway tools. The scripts are very similar to the scripts we created for ISOLET, so in the following discussion we will focus only on the differences. Thus we recommend you read Section 3.5 before this section.

When creating our noisy Numbers corpus, we used the utterance lists from [109], so that the noisy corpus can be easily used by other people who are using those same utterance lists. There were 10441 training set utterances, 3582 development set utterances, and 3620 evaluation set utterances. But when creating our ASR system for Numbers, we removed additional utterances due to transcription errors or other problems noticed by ourselves or by the authors of [110], who supplied us with the utterance lists they used. Thus our ASR system uses 10231 training set utterances, 3515 development set utterances, and 3548 evaluation set utterances.

Our ASR scripts use the cross-validation (CV) portion of the training set (about 10% of the training data) both as CV data for early stopping during the MLP training process and as a test set for tuning decoder options using a grid search. We chose to tune decoder options on the CV data since that was done by users of earlier versions of the Numbers corpus in our research group [7][38]. Since exploring points on the decoder parameter tuning grid runs much more slowly for Numbers than for ISOLET, for Numbers we use a much coarser grid, which might put novel feature vectors at some disadvantage.

The number of MLP output units was 28, corresponding to 28 phones (including a silence "phone"). Coincidentally, our ISOLET MLP-based ASR system also had 28 output units. For decoding, we used a trigram language model.

For the results in this thesis, we ran our Numbers ASR system using Quicknet version 3.20 and noway version 2.9. For consistency, we always used the same version (Pentium 4 SSE2) of the Quicknet training tool.

We also created scripts for statistical significance testing of ASR performance differences using a matched pairs sign test. We have placed an explanation of the matched pairs sign test in the README file that accompanies the scripts.

Our Numbers ASR system can be downloaded at `http://www.icsi.berkeley.`

edu/speech/papers/gelbart-ms/numbers. A README file is included which gives additional details about the ASR system and provides a tutorial for using the scripts.

3.10 How our ASR systems compare to published results

3.10.1 ISOLET

Using a GMM-based ASR system, Karnjanadecha and Zahorian [114] reported 4.1% WER (word error rate) on the clean ISOLET corpus using MFCC features and 2.3% WER using their novel DCSC features. WER was averaged over folds 1-5. The other published ISOLET results that we are aware of fall within this range.

Our best baseline result using our own MLP-based ASR system was 2.9% WER (averaged over folds 2-5) using a feature vector of MFCC, PLP and MSG features concatenated together in Table 4.4. Thus our baseline system performance is comparable to previously published results.

3.10.2 Numbers

The best result we know of on the clean Numbers corpus is 2.0% WER. This is an unpublished result achieved recently in our research group by Arlo Faria using version 1.0 of the corpus. (Table 4 in [109] compares ASR performance between version 1.0 and version 1.3.)

Faria used the SRI DECIPHER recognizer with cross-word triphones. Each state was represented with 16 full covariance Gaussians, and states were tied using decision-tree clustering. The front end used PLP and MFCC features in a tandem (see Sec-

tion 2.2.3) approach: PLP features were used as input to an MLP classifier, and the outputs of the MLP were then reduced to 21 dimensions using PCA and concatenated with MFCC features to form a 60-dimensional feature vector for the main ASR system. Three speaker-level normalization techniques (cepstral mean normalization, cepstral variance normalization and vocal tract length normalization) were used, and recognition was performed in a single pass without maximum likelihood or maximum a posteriori speaker adaptation.

Our best baseline result using our own ASR system is 4.5% WER on the Numbers 1.3 evaluation set using MFCC, PLP, and MSG features concatenated together into one feature vector (see Table 6.3 and the note on performance in Appendix A). This is much worse than Faria's WER, but comparable to the best published results we are aware of for the Numbers corpus. For example, [109] reports 5.1% WER on the Numbers 1.3 evaluation set using MFCC features with a GMM-based ASR system and [115] reports 4.5% WER for Numbers 1.0 with a GMM-based ASR system using cepstral mean normalization, cepstral variance normalization and vocal tract length normalization.

3.11 Summary

To pursue our research agenda, we created noisy versions of the ISOLET and Numbers corpora, which other researchers can recreate by using our scripts to repeat the noise-adding process. We also set up ISOLET and Numbers ASR systems which are available for use by other researchers. The word error rates for these systems are comparable to typical published results and thus these systems provide a credible framework for ASR research.

Chapter 4

Random subspace feature selection for the ISOLET task

Contents

4.1 Introduction

In this chapter, we use the random subspace method (RSM) [21] for EFS, which we discussed briefly in Section 2.3.3. In RSM, EFS is performed by simply choosing the features in each of the feature vectors at random from the feature pool. Multi-stream ASR systems typically use a small number of classifiers, but RSM tends to be performed using a large number of classifiers. For example, in [21] RSM was used to create ensembles of 50-100 decision trees, and in [24] it was used to create ensembles of 25 HMMs. We chose to use 25 MLPs in our RSM ensembles in this chapter. Our implementation of RSM allows the same feature to occur in more than one feature vector, but does not allow the same feature to occur more than once in the same feature vector. We fixed the length of all 25 feature vectors at 39, which is a common length for ASR feature vectors. (MFCC and PLP, the two most common feature extraction methods in ASR research, are usually used to generate 36 or 39 features.)

Throughout this chapter, we use a two-tailed McNemar's test to judge statistical significance (see Section 3.5), and we use a probability of the null hypothesis of 0.05 as the threshold for statistical significance. In order to simplify the presentation of results, we only state whether or not differences were significant, without providing the P values.

4.2 Automatic speech recognition system

To measure RSM performance, we used the MLP-based ISOLET ASR system that we described in Section 3.5. Our ISOLET ASR scripts reserve fold 1 as a tuning set for tuning decoder parameters. We repeated this tuning process whenever we made a change to feature vectors (e.g., changing between PLP and MFCC), MLP topology (i.e., changing the number of input or hidden units), or the training condition (i.e.,

changing between clean and noisy training). In our experiments for the clean train and noisy test condition, we used decoder parameters determined by tuning on clean ISOLET. Because we used fold 1 for decoder parameter tuning, we used folds 2-5 for evaluation of ASR system performance and we report word error rates (WERs) averaged over folds 2-5.

We always used 25 streams and 39 features per stream in our RSM systems, with the MLP for each stream having 800 hidden units. Thus, based on the formula given in Section 3.5, our RSM systems had 4,480,700 acoustic model parameters (MLP weights and biases) each. As an experimental control, for all the non-RSM systems examined in this chapter, we chose the number of MLP hidden units so that the total number of acoustic model parameters would also be approximately 4,480,700. In the ensemble non-RSM systems, we chose the number of hidden units for each MLP in the ensemble so that the number of acoustic model parameters per MLP was approximately equal and the total number of acoustic model parameters (added up across all MLPs) was approximately 4,480,700. We will refer to the non-RSM systems as "baseline" systems.

While our RSM systems had a fairly small number of parameters per MLP due to the 25 MLPs in the ensemble, for the baseline systems 4,480,700 is a large number of parameters for the amount of training data that we used. While we employed early stopping in the neural network training to guard against overfitting, that technique is imperfect so the large number of parameters may have resulted in suboptimal baseline performance in some cases.

4.3 Feature extraction

Our MFCC feature type was 39 Mel-Frequency Cepstral Coefficient features (13 static features, 13 delta features, and 13 double-delta features). We calculated MFCC features using the HCopy tool from Cambridge's HTK package [100]. The features were calculated using a 25 ms analysis window length. Deltas and double-deltas were calculated over five frames, which is the HTK default. (After the ISOLET experiments in this thesis were completed, we noticed that we could improve MFCC performance on the ISOLET task by using the HTK ZMEANSOURCE option. For more information on this, see the README_MFCC file in our recognition testbed: `http://www.icsi.berkeley.edu/Speech/papers/gelbart-ms/hybrid-testbed`.)

Our PLP feature type was 39 Perceptual Linear Prediction [37] features (13 static features, 13 deltas, and 13 double-deltas). We calculated PLP features using the feacalc tool from the ICSI SPRACHcore [99] package. The features were calculated using a 25 ms analysis window length. Deltas and double-deltas were calculated over nine frames, which is the feacalc default.

Our MSG feature type was 36 Modulation SpectroGram [38] features. To calculate these features we always used the "'msg3" calculation mode offered by the msgcalc tool from the SPRACHcore package, with a 25 ms analysis window length.

We also used PLP and MFCC features that were calculated using a 15 ms or 35 ms analysis window length instead of 25 ms. The use of multiple window lengths was motivated by the observation that using a single window length forces a particular trade-off between spectral resolution and temporal resolution [46][47][116][117].

We used a window step at 10 ms for all features, so that all our features were extracted 100 times per second. We did not use any additional robustness techniques

such as cepstral mean normalization or noise removal [61], although MSG does include a type of automatic gain control.

The scripts we used to invoke MFCC, PLP, and MSG feature extraction are included in the downloadable ISOLET script archive described in Section 3.5.

When we identify feature pools in our tables of results, analysis window lengths given at the end of a line listing feature extraction methods apply to all of the methods. For example, "MFCC and PLP (15 ms)" means a pool containing MFCC and PLP features, both calculated using a 15 ms analysis window length, for a total of 78 features. "MFCC and PLP (15, 25, and 35 ms)" means a pool containing MFCC and PLP features, both calculated using 15, 25, and 35 ms analysis window lengths, for a total of 234 features. In the baseline systems using geometric mean posterior combination, each feature type in the pool is given its own MLP. The 39 MFCC features, the 39 PLP features, and the 36 MSG features are each considered a feature type. Features calculated with a particular analysis window length are considered a distinct feature type from features calculated with other lengths.

Because RSM selects features in a single step, it is very quick and thus in this chapter we were able to examine RSM performance for nine feature pools.

4.4 Experimental results and discussion

4.4.1 RSM vs. non-RSM performance compared for each feature pool

Table 4.4 in the next section gives complete word error rate (WER) results for our experiments. Since that table is lengthy, we have used Tables 4.1, 4.2, and 4.3 to summarize how RSM and non-RSM performance compare to each other for each of

our nine feature pools. When we use RSM, there are always 39 features per stream. Thus we cannot meaningfully use RSM if only 39 or fewer features are available. Therefore, the single feature types in the first seven rows of Table 4.4 do not count as part of our nine feature pools and they are not included in Tables 4.1, 4.2, and 4.3.

For each feature pool and train/test condition, Table 4.1 compares the RSM system to the geometric mean posterior combination baseline system (in which a separate MLP is used for each feature type), Table 4.2 compares the RSM system to the feature concatenation baseline system (in which all features in the pool are concatenated into a single feature vector used by a single MLP), and Table 4.3 compares the RSM system to the best performing of the two types of baseline system.

In these three tables, we see that in both the clean train and clean test condition and the noisy train and noisy test condition RSM performed better than baseline in all cases for which there was a statistically significant difference in performance between the RSM system and the baseline system. In the clean train and clean test condition, it was easier for RSM to improve on the posterior combination baseline (which it improved on for five of nine pools) than the feature concatenation baseline (which it improved on for two of nine pools).

RSM obtained the greatest number of improvements in the clean train and noisy test condition. In that condition, RSM performed better than the feature concatenation baseline for all seven pools for which there was a statistically significant difference in performance, and better than the posterior combination baseline for eight pools. RSM performed worse than the posterior combination baseline for the PLP and MSG pool.

We received this anonymous review comment after submitting a conference paper based on some of the work in this chapter: "People in my group have tried using randomized features on large state-of-the-art Switchboard type systems, with no suc-

Feature pool	Clean train and test	Noisy train and test	Clean train, noisy test
MFCC and PLP	C	A (5.2%)	A (2.9%)
MFCC and PLP (15, 25, and 35 ms)	A (12.9%)	C	A (9.7%)
MFCC, PLP and MSG	C	C	A (2.8%)
MFCC and MSG	A (12.0%)	C	A (2.2%)
PLP and MSG	C	C	B (5.3%)
MFCC (15, 25, and 35 ms)	A (14.4%)	A (8.6%)	A (8.2%)
PLP (15, 25, and 35 ms)	A (17.8%)	A (5.3%)	A (9.3%)
MFCC and PLP (15 ms)	C	C	A (3.0%)
MFCC and PLP (35 ms)	A (9.6%)	A (3.2%)	A (4.6%)

Table 4.1. RSM compared to geometric mean posterior combination baselines for particular feature pools. Analysis window lengths given in parentheses as part of the feature pool name apply to all features in the pool. For example, for the "MFCC and PLP (15, 25, and 35 ms)" feature pool there are six feature types (MFCC and PLP each calculated for three analysis window lengths). The symbol A indicates that the RSM system was better than the baseline system and this was statistically significant; relative WER improvement follows in parentheses. The symbol B indicates the baseline system was better than the RSM system and this was statistically significant; relative WER improvement follows in parentheses. The symbol C indicates the difference between the RSM system and the baseline system was not statistically significant.

Feature pool	Clean train and test	Noisy train and test	Clean train, noisy test
MFCC and PLP	C	C	C
MFCC and PLP (15, 25, and 35 ms)	C	C	A (4.1%)
MFCC, PLP and MSG	C	A (5.2%)	A (7.0%)
MFCC and MSG	C	A (15.7%)	A (12.9%)
PLP and MSG	C	A (5.6%)	A (4.7%)
MFCC (15, 25, and 35 ms)	A (12.9%)	A (5.7%)	A (3.5%)
PLP (15, 25, and 35 ms)	A (27.6%)	A (7.4%)	A (8.6%)
MFCC and PLP (15 ms)	C	C	C
MFCC and PLP (35 ms)	C	C	A (4.1%)

Table 4.2. RSM compared to feature concatenation baselines for particular feature pools.

cess... selecting features randomly from the MFCC/PLP feature sets, and combining the recognition results either through ROVER [see [39]] or confusion network combi-

Feature pool	Clean train and test	Noisy train and test	Clean train, noisy test
MFCC and PLP	*C*	*C*	*C*
MFCC and PLP (15, 25, and 35 ms)	*C*	*C*	*A* (4.1%)
MFCC, PLP and MSG	*C*	*C*	*A* (2.8%)
MFCC and MSG	*C*	*C*	*A* (2.2%)
PLP and MSG	*C*	*C*	*B* (5.3%)
MFCC (15, 25, and 35 ms)	*A* (12.9%)	*A* (5.7%)	*A* (3.5%)
PLP (15, 25, and 35 ms)	*A* (17.8%)	*A* (5.3%)	*A* (8.6%)
MFCC and PLP (15 ms)	*C*	*C*	*C*
MFCC and PLP (35 ms)	*C*	*C*	*A* (4.1%)

Table 4.3. RSM compared to the best baselines for particular feature pools. Baseline systems used either feature concatenation or geometric mean posterior combination; the baseline system considered in the table is the best of these two (i.e., the one with the lowest WER) for the given feature pool and train-test condition. (If the two combination methods were tied for the best performance, one was arbitrarily chosen for comparison with RSM.)

nation [see [40]] simply [did] not work." Like the tables in this chapter, this comment suggests that the effectiveness of RSM can depend greatly on the feature pool. However, another plausible interpretation of this comment is simply that it was harder to improve on a large, state-of-the-art baseline system than it was to improve on the simpler baselines used in this chapter.

4.4.2 RSM vs. non-RSM performance compared across all feature pools

Table 4.4 below gives complete WER results for this chapter. Double horizontal lines separate results for different feature pools.

System type	Clean train and test	Noisy train and test	Clean train, noisy test
Table 4.4 continues on next page			

Table 4.4 continued from previous page

MFCC	4.5	17.6	57.8
PLP	4.0	17.2	55.0
MSG	5.5	16.5	38.0
MFCC (15 ms window)	4.7	18.3	55.5
MFCC (35 ms window)	3.8	17.4	56.6
PLP (15 ms window)	4.9	17.6	56.5
PLP (35 ms window)	3.6	17.1	54.7
Pool: MFCC and PLP			
RSM	3.2	15.9	52.3
All features concatenated	3.3	16.5	51.8
One MLP per feature type	3.2	16.8	53.8
Pool: MFCC and PLP (15, 25, and 35 ms windows)			
RSM	2.9	15.7	48.8
All features concatenated	3.0	15.8	50.9
One MLP per feature type	3.3	16.2	54.0
Pool: MFCC, PLP, and MSG			
RSM	3.1	13.8	45.0
All features concatenated	2.9	14.6	48.4
One MLP per feature type	3.0	14.1	46.3
Pool: MFCC and MSG			

Table 4.4 continues on next page

Table 4.4 continued from previous page

RSM	3.0	13.2	43.5
All features concatenated	3.2	15.7	49.9
One MLP per feature type	3.5	13.6	44.5
Pool: PLP and MSG			
RSM	3.2	13.5	45.0
All features concatenated	3.4	14.3	47.2
One MLP per feature type	3.2	13.8	42.6
Pool: MFCC (15, 25, and 35 ms windows)			
RSM	3.2	16.0	50.4
All features concatenated	3.7	17.0	52.2
One MLP per feature type	3.8	17.5	54.8
Pool: PLP (15, 25, and 35 ms windows)			
RSM	3.1	16.2	49.0
All features concatenated	4.3	17.5	53.6
One MLP per feature type	3.8	17.1	54.1
Pool: MFCC and PLP (15 ms windows)			
RSM	3.5	16.5	52.6
All features concatenated	3.6	16.7	52.1
One MLP per feature type	3.7	16.8	54.2
Pool: MFCC and PLP (35 ms windows)			

Table 4.4 continues on next page

Table 4.4 continued from previous page

RSM	3.2	15.9	51.7
All features concatenated	3.4	15.5	54.0
One MLP per feature type	3.5	16.4	54.2

Table 4.4: This table gives the complete word error rate (WER) results for our experiments, averaged over folds 2-5. The first seven rows give results for baseline systems using just one feature type each. Later rows give RSM and baseline results for feature pools containing more than one feature type. Double horizontal lines are used to group together systems which use the same feature pool. Feature calculation window lengths are 25 ms unless otherwise stated. "All features concatenated" means a baseline system for which a single MLP was used with every feature in the pool. "One MLP per feature type" means a baseline system for which an ensemble of MLPs was used with a separate MLP for each feature type in the pool. For example, for the pool containing MFCC and PLP feature calculated using 15, 25 and 35 ms windows, the ensemble contains six MLPs, each one using MFCC or PLP features calculated using a particular window length.

We now refer to Table 4.4 to consider what RSM and non-RSM approaches gave the lowest WERs of all, without restricting comparisons to particular feature pools. In the clean train and clean test condition, the best RSM system used the pool of MFCC and PLP features calculated at 15, 25, and 35 ms analysis window lengths, and the best baseline system used feature concatenation with MFCC, PLP, and MSG features. Both systems had a 2.9% WER.

In the noisy train and noisy test condition, the best RSM system used the pool of MFCC and MSG features (13.2% WER), and the best baseline system used geometric mean posterior combination with the same features (13.6% WER). The performance difference between these two systems was not statistically significant.

In the clean train and noisy test condition, the best RSM system used the pool of MFCC and MSG features (43.5% WER) and the best baseline system used only MSG features (38.0% WER). The performance difference between these two systems was statistically significant. (Since the 38.0% WER system uses only one feature type, it is not included in Table 4.3. The lowest baseline WER using more than one feature type was 42.6% using geometric mean posterior combination with the pool of PLP and MSG features, and the difference between that and the 43.5% WER RSM system was also statistically significant.)

4.5 The feature vectors chosen by the random subspace method

To download lists of the feature vectors chosen by RSM, or the script we used to create those lists, see `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/rsm`.

For eight of the nine feature pools, every feature in the pool was part of at least one of the 25 streams. For the remaining pool (the pool of MFCC and PLP features calculated at 15, 25, and 35 ms analysis window lengths), RSM utilized 231 out of 234 features.

4.6 Alternatives to our RSM implementation choices

In this section we discuss two alternative ways to implement RSM which may be worth exploring in future work.

Every feature vector in our RSM ensembles in this chapter contained 39 features. We chose this size because it is common in ASR, but we did not compare performance to other sizes. In Ho's work on RSM performance with ensembles of decision trees [21] she found that setting feature vector length to half the size of the feature pool worked well.

Furthermore, in both [21] and our own work the feature vector length was the same for different classifiers in the same RSM ensemble. However, some other authors [12][87][25] have chosen the length of each feature vector in the ensemble randomly instead. By fixing the feature vector length, we avoided the possibility of classifiers that had very low accuracies due to being assigned very few features. There are comments in the literature that support this philosophy. The authors of [43] wrote that "previous [ASR] work has shown that the combination of independent systems with very different error rates often yields no gain, and so it is desirable to combine independent systems with comparable error rates", and the author of [7] noted that a very poorly performing member of an ASR ensemble may drag down ensemble performance. On the other hand, the ASR systems referred to in [43][7] were different from ours. For example, perhaps a wider error rate spread is workable with 25 classifiers in the ensemble, which is many more than are typically used in multi-stream ASR. And the authors of [25] felt that varying feature vector lengths can provide useful additional diversity. So whether our approach was optimal is an experimental question which we have left open.

4.7 Summary

Tables 4.1, 4.2, and 4.3 showed that when we compared RSM and baseline performance for particular feature pools and train/test conditions, RSM was better than the baseline in all but one of the cases where there was a statistically significant performance difference. However, for many other cases, there was no statistically significant difference between RSM and the baseline. In Section 4.4.2, we saw that when comparing the best RSM system to the best baseline systems without restricting ourselves to particular feature pools, RSM did not give a statistically significant performance improvement over baseline in any of the three train and test data conditions.

So we have demonstrated that RSM can improve ASR ensemble performance, but also that in many cases it does not. In Section 4.6, we discussed alternative ways to implement RSM which could be explored in further work. However, after reviewing our RSM results, we decided instead to follow the RSM experiments in this chapter by moving from random selection to a guided selection algorithm, hill-climbing, which we use with the ISOLET corpus in Chapter 5 and (in our most successful experiments) for the Numbers corpus in Chapter 6.

Chapter 5

Hill-climbing feature selection for the ISOLET task

Contents

5.1 Introduction

In this chapter we try a guided, wrapper (see Section 2.3) approach to feature selection. We use the hill-climbing algorithm, which we discussed earlier in Section 2.3.3. As we did for the random subspace method, we evaluate the hill-climbing algorithm using our publicly available benchmark based on the OGI ISOLET corpus, which we described in detail in Chapter 3. We have some successes, but our best results with hill-climbing come after this chapter when we switch to the Numbers corpus, a larger corpus for which more data is available to guide hill-climbing.

Since multi-stream ASR systems typically use a small number of classifiers, in our hill-climbing experiments we use three MLPs per ensemble.

As in Chapter 4, in this chapter we use a two-tailed McNemar's test to judge statistical significance (see Section 3.5), and we use a probability of the null hypothesis of 0.05 as the threshold for statistical significance. When we judge a difference to be statistically significant, we provide the P value since this allows the reader to use a different threshold for significance if they choose.

5.2 Automatic speech recognition system

As in our experiments with the random subspace method in Chapter 4, in this chapter we used the MLP-based ISOLET ASR system that we described in Section 3.5. We used fold 1 for decoder parameter tuning and folds 2-5 for evaluation of ASR system performance. We will refer to fold 1 as the "tuning fold" and folds 2-5 as the "evaluation folds". As in Chapter 4, we repeated the decoder parameter tuning process whenever we made changes to feature vectors, MLP topology, or the training condition.

5.3 Feature extraction

Since it is iterative, hill-climbing is much slower than the random subspace method, and so in our hill-climbing experiments we used only a single feature pool, rather than the nine pools we used in Chapter 4.

Our feature pool consisted of MFCC, PLP and MSG features calculated using a 25 ms analysis window length. Thus the size of our feature pool was $39 + 39 + 36 = 114$ features. See Section 4.3 for details, including a note about how MFCC performance could have been improved using the ZMEANSOURCE option.

5.4 Acoustic model size

Based on Table 5.1, which shows word error rates (WERs) on the tuning fold for various MLP hidden layer sizes, we decided that 1600 hidden units was a reasonable choice for a single-MLP MFCC or PLP system[1]. Such a system has about 358,400 acoustic model parameters (MLP weights and biases). To control acoustic model size, for each experiment in this chapter (outside of Table 5.1) we chose the number of MLP hidden units so that the total number of acoustic model parameters in the ASR system was roughly 358,400. For a system with more than one MLP, this total was the sum of the number of parameters for all the MLPs, and the number of hidden units for each MLP was chosen so that the number of parameters for each MLP was approximately equal. When feature vector sizes were changed during hill-climbing, we adjusted hidden layer sizes to satisfy these requirements.

The number of acoustic model parameters for an MLP is $(I + O) * H + H + O$, as explained in Section 3.5. For our ISOLET testbed, O was 28, our MFCC and PLP

[1] In fact, that Table suggests that 1400 hidden units might have been a slightly better choice, but there is not much difference. We chose 1600 for comparability with some past results that are not included here.

feature vectors had 39 features and our MSG feature vector had 36 features. Thus, our single-MLP MFCC or PLP system had 1600 hidden units, and our single-MLP MSG system had 1715 hidden units. When we concatenated MFCC, PLP and MSG features into a single feature vector, we used a single-MLP system with 598 hidden units. And our three-MLP MFCC, PLP, and MSG systems used 533 hidden units for the MFCC and PLP MLPs and 572 hidden units for the MSG MLP.

Number of hidden units	Clean train and test			Noisy train and test		
	MFCC	PLP	Mean	MFCC	PLP	Mean
600	3.8	3.6	3.7	18.1	17.8	18.0
700	3.6	2.8	3.2	18.5	17.5	18.0
800	3.3	2.8	3.1	18.3	17.9	18.1
900	3.3	3.2	3.3	17.9	17.5	17.7
1000	3.3	2.9	3.1	18.3	17.6	18.0
1100	3.0	2.9	3.0	18.3	17.7	18.0
1200	2.9	2.9	2.9	18.1	17.6	17.9
1300	3.0	3.0	3.0	18.7	17.0	17.9
1400	2.9	2.9	2.9	18.1	16.6	17.4
1500	3.0	3.4	3.2	18.3	17.3	17.8
1600	3.0	2.7	2.9	18.2	17.1	17.7
1700	3.0	2.7	2.9	18.3	17.9	18.1
1800	2.9	3.0	3.0	18.5	17.1	17.8

Table 5.1. The table shows word error rates (WERs) calculated on the tuning fold (fold 1) for single-MLP systems using either MFCC or PLP features, as a function of hidden layer size. Since these WERs are reported on the tuning fold, they are low compared to results in other tables.

5.5 Hill-climbing procedure

The hill-climbing EFS algorithm was defined under the name EFS_SBC in [87], and later used in [25]. Below, we provide a pseudocode listing for our HILL_CLIMBING procedure and the helper procedure CALCULATE_SCORE. The

pseudocode listing is inspired by the listing in [87], but has been modified to match our implementation. Pseudocode lines starting with a triangle symbol are comments.

There are differences between EFS_SBC as defined in [87] and our implementation of hill-climbing. First, EFS_SBC assumes the use of simple (naive) Bayesian classification, while we use a hidden Markov model approach based on MLPs. Second, in half of our experiments we scored candidate feature vectors by ensemble word recognition accuracy (or ensemble WER, which is equivalent except that decreases, rather than increases, are improvements), rather than using the $fitness_s$ function defined below in Equation 5.1, which was used in [87]. Third, for EFS_SBC the initial assignment of features to classifiers is always performed by the random subspace method (RSM) [21]. We used RSM for this in half of our experiments, but in the other half we started each classifier off using a particular feature extraction algorithm (one MLP using MFCC, another using PLP, and a third using MSG). And when we used RSM, we determined the number of features per classifier ahead of time, so that the initial MLP topology (the number of inputs, hidden units, and output units) would be the same whether or not we initialized by RSM. When our initial three feature vectors were respectively MFCC, PLP and MSG, there were 39 features for each of the first two MLPs and 36 features for the third MLP. So when our initial feature vectors were chosen by RSM we also used 39 features for the first two MLPs and 36 features for the third MLP. In the implementation of RSM in [87][25], on the other hand, the length of feature vectors was chosen randomly. (See Section 4.6 for further discussion.)

```
HILL_CLIMBING(FS, S, N, α)
 1  ▷ FS: the set of feature vectors (already initialized)
 2  ▷ S: the number of feature vectors in FS
 3  ▷ N: the number of features in the feature pool
 4
 5  ▷ Perform hill-climbing for each stream in turn
 6  for s ← 1 to S
 7        do
 8           ▷ Initialize this stream's score.
 9           score ← CALCULATE_SCORE(FS, s, α)
10           repeat
11                    improvement ← false
12                    for i ← 1 to N
13                          do
14                             ▷ If feature i is in stream s, remove it.
15                             ▷ If feature i is not in stream s, add it.
16                             SWITCH(i, FS[s])
17                             newScore ← CALCULATE_SCORE(FS, s, α)
18                             ▷ If the change improved the score
19                             if newScore > score
20                                then
21                                      score ← newScore
22                                      improvement ← true
23                                else
24                                      ▷ Undo the change.
25                                      SWITCH(i, FS[s])
26              until improvement = false
```

```
CALCULATE_SCORE(FS, s, α)
1  if using α
2      then
3          score ← STREAM_ACCURACY(FS[s]) + α * DIVERSITY(s, FS)
4      else
5          score ← ENSEMBLE_ACCURACY(FS)
6
7  return score
```

In half our hill-climbing experiments, we use the following formula to score candidate feature vectors:

$$fitness_s = acc_s + \alpha * div_s \tag{5.1}$$

This formula, which we discussed in Section 2.3.3, combines an individual classifier's accuracy and its contribution to the diversity of the ensemble. When we used this formula, we set acc_s to the single-stream word recognition accuracy for stream s, and set div_s to stream s's contribution to word level ensemble diversity. We calculated both accuracy and diversity as percentages between 0% and 100%. We calculated div_s by calculating the pairwise diversity between stream s and each other stream, and then averaging the pairwise diversities. We defined pairwise diversity as the number of word hypotheses that differ divided by the total number of words. We calculated pairwise diversity by calculating the WER of one system using the other system's output as if it were the ground truth transcript. (If the WER changed depending on which of the two outputs we used as a transcript, we took the minimum of the two WERs. This never happened with ISOLET due to the simplicity of WER scoring for isolated words, but it happened sometimes with the Numbers connected speech corpus used in the next chapter.)

The α parameter provides an adjustable trade-off between accuracy and diversity. We always used the same value for α ($\alpha = 1$), and we based this on published results [87] that did not involve speech recognition, so it might be possible to improve performance further by changing the value of α.

Our diversity measure div_s is a word-level rather than frame-level diversity measure, even though we combine MLPs at the frame level when we do ensemble recognition. We think this is reasonable, though not perhaps not ideal, because we expect that when different MLPs make different errors at the word level there tends to be useful diversity at the frame level as well.

The scripts we used for hill-climbing can be downloaded at `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/isolet-hillclimb`.

5.6 The time taken by hill-climbing

The hill-climbing procedure, being a wrapper method (see Section 2.3), requires the calculation of word error rates at every step, and this makes it quite time consuming. To help with this, we parallelized the process.

The key idea behind the parallelization, which was suggested to us by Alexey Tsymbal, is to look ahead beyond the current feature when making a pass over the feature pool, using the prediction that the current feature will not be switched. "'Switched", here, means changed from being included in the feature vector to not being included, or vice versa. For instance, we could compute the score for switching feature n in parallel with computing the score for switching feature $n + 1$, if we compute the latter score using the prediction that feature n will not be switched. If the prediction turned out to be wrong, we would have to recompute the score for switching feature $n+1$. We could additionally compute the score for switching feature

$n+2$ on the assumption that both feature n and feature $n+1$ will not be switched, and so on. This parallelization strategy is an example of the speculative execution [118] technique used in computer architecture.

We chose to use just a single step of speculative execution, computing the score for switching feature n in parallel with computing the score for switching feature $n+1$. Specifically, we ran the MLP training for feature $n+1$ in parallel with decoder parameter tuning for feature n. So the time taken by a single inner loop iteration of the hill-climbing algorithm was roughly the maximum of MLP training time and decoder tuning time (plus some overhead), except for the times when we recomputed a score because the prediction used for speculative execution was wrong. The number of wrong guesses corresponds to the number of changes made to feature vectors, which is shown in Table 5.2. We further sped up the decoder parameter tuning by parallelizing the grid search across 12 cores. This reduced the time taken by a tuning so it was similar to the time taken by an MLP training. For MLP training we used only a single core, since MLP training time was not a major bottleneck.

The time taken to run hill-climbing varied between experiments, ranging roughly from four days to eight days. The processors used were mainly AMD Opteron 875s at 2.2 GHz and Intel Xeons at 2.8 GHz or 3.0 GHz. The experiments that used Equation 5.1 to score performance took the longest, because using that scoring function led to more changes being made to feature vectors.

Decoder parameter tuning was very time consuming and we could have sped up hill-climbing by doing less of it. The hill climbing procedure has three nested loops, as shown in the HILL_CLIMBING pseudocode listing above. The inner loop examines whether switching a particular feature would improve performance for the current stream. The middle loop repeats the inner loop over and over for the current stream until the inner loop stops making changes. And the outermost loop invokes the middle

loop once for each stream. We did decoder parameter tuning in the inner loop but we could have moved it to the middle or outer loop, or done it only at the start and end of the hill-climbing process. It would be interesting to know how much effect this would have had on the performance of new feature vectors that were tried during hill-climbing.

5.7 Results and discussion

5.7.1 Hill-climbing results

Table 5.2 shows our hill-climbing results. Results are averaged over the evaluation folds (folds 2-5). We exclude the tuning fold (fold 1) from the table since it was used as development data to guide the hill-climbing process (see Section 3.4).

For the noisy data, hill-climbing improved WER compared to the initial ensemble that hill-climbing started with in three out of four cases, and the improvement was statistically significant each time ($P = 0.004$ in row (b) of the table, $P = 0.02$ in row (c), and $P = 0.0006$ in row (d)). In row (a), hill-climbing worsened WER from 14.3% to 14.5% but this change was not statistically significant. For the clean data, in three out of four experiments hill-climbing worsened WER, and in one experiment hill-climbing improved WER, but the change in WER due to hill-climbing was not statistically significant in any of the four experiments. Thus, the only statistically significant effect of hill-climbing was to improve performance, never to worsen it.

There was no statistically significant difference in final WER between the two scoring approaches used to guide hill-climbing (ensemble WER and Equation 5.1) in the clean case where RSM was not used, the clean case where RSM was used, or the noisy case where RSM was used. This was despite the fact that guiding hill-

climbing using Equation 5.1 resulted in many more feature vector changes during the hill climbing process. There was, however, a statistically significant difference ($P = 0.001$) in the noisy case where RSM was not used, with Equation 5.1 resulting in a 13.6% final ensemble WER compared to 14.5% using the other scoring approach.

Whether or not we chose the initial feature vectors for hill-climbing randomly made a statistically significant difference to final WER in the clean case when we used ensemble WER to guide hill-climbing (random initialization raised final WER from 3.0% to 3.4%, $P = 0.02$) and the noisy case when we used Equation 5.1 to guide hill-climbing (random initialization raised final WER from 13.6% to 14.4%, $P = 0.01$). (There was no statistically significant difference in the other two cases.) Thus performance was better when we did not choose the initial feature vectors randomly.

Experiment	Clean train and test			Noisy train and test		
	Changes	Initial WER	Final WER	Changes	Initial WER	Final WER
(a) HC	4	3.1	3.0	7	14.3	14.5
(b) HC, RSM initialization	5	3.2	3.4	11	15.3	14.5
(c) HC using $\alpha = 1$	14	3.1	3.3	22	14.3	13.6
(d) HC using $\alpha = 1$, RSM initialization	20	3.2	3.4	18	15.3	14.4

Table 5.2. This table shows hill-climbing results on the ISOLET corpus. The "Changes" columns give the number of features changed (added to or deleted from a feature vector) during hill-climbing. The "Initial WER" columns give the initial ensemble (i.e., three MLPs) word error rate on the evaluation folds before the hill-climbing algorithm has made any changes to the feature vectors. The "Final WER" columns give the ensemble word error rate on the evaluation folds once hill climbing has finished. If a value is given for α it means that the score used to guide hill-climbing was the accuracy and diversity formula defined in Equation 5.1, calculated on the tuning fold. Otherwise, the score used to guide hill-climbing was the ensemble WER on the tuning fold. "RSM" means that initial feature vectors prior to the start of hill-climbing were chosen randomly. Otherwise, the initial feature vectors were respectively MFCC, PLP and MSG.

5.7.2 Additional baseline results

Table 5.3 shows additional baseline results for the ISOLET corpus. In both the clean and noisy cases, the lowest WER is in row (e), which is the three-MLP system using MFCC, PLP and MSG features. In the clean case, there is no statistically significant difference between the system in row (e) and the system with the second lowest WER in row (f) or the system with the third lowest WER in row (d). However, there is a statistically significant difference between the system with the fourth lowest WER in row (b) and the system in row (e) ($P = 0.01$). In the noisy case, the system with the second lowest WER is the system in row (d) which uses MFCC, PLP and MSG feature concatenated into a single feature vector. There is a statistically significant difference between the system in row (e) and the system in row (d) ($P = 0.004$).

For noisy data, the best performance of a system chosen by hill-climbing was 13.6% WER in row (c) of Table 5.2, which is a statistically significant improvement over the system in row (e) of Table 5.3 ($P = 0.02$). For both clean and noisy data, there were no other statistically significant differences in performance between the system in row (e) of Table 5.3 and the systems chosen by hill-climbing.

In the previous chapter, we found a 25-MLP RSM ensemble using the MFCC, PLP and MSG feature pool had a 3.1% WER on clean data and a 13.8% WER on noisy data. In Table 5.3, the three-MLP RSM ensemble has a 3.2% WER on clean data and a 15.3% WER on noisy data. In the noisy case, the performance difference is statistically significant ($P < 10^{-8}$). This is consistent with the experiments with different RSM ensemble sizes using decision trees in [21], in which 25-classifier ensembles generally outperformed three-classifier ensembles. In our case, the comparison is complicated by the different number of acoustic model parameters in the two systems as well as the role of random chance in the feature assignment. In [21] the effect of

random chance was reduced by providing results for many data sets and ensemble sizes.

Experiment	Clean train and test	Noisy train and test
(a) MFCC	4.1	17.6
(b) PLP	3.6	17.4
(c) MSG	5.6	16.4
(d) All features concatenated	3.4	15.2
(e) MFCC, PLP, MSG (three MLPs)	3.1	14.3
(f) RSM (three MLPs)	3.2	15.3

Table 5.3. This table shows baseline results on the ISOLET corpus. The results shown are word error rates (WERs) on the evaluation folds. The number of hidden units (HUs) for each MLP was chosen as explained in Section 5.4. In row (d), all features in the feature pool are concatenated into a single feature vector. In row (e), there are three MLPs respectively using 39 MFCC features (533 HUs), 39 PLP features (533 HUs), and 36 MSG features (572 HUs). Row (e) corresponds to the "Initial WER" column in Table 5.2 for the experiments without RSM initialization. In row (f), there are three MLPs, using randomly chosen feature vectors with the same number of features and hidden units as in row (e). Row (f) corresponds to the "Initial WER" column in Table 5.2 for the experiments with RSM initialization.

5.7.3 Tuning fold results compared to evaluation fold results

Table 5.4 shows hill-climbing results on the tuning fold. Compared to the initial system that hill-climbing started with, hill-climbing lowered tuning fold WER in seven out of eight experiments. This is much better than the results on the evaluation folds in Table 5.2. Would we have been able to predict from Table 5.4 for what cases hill-climbing would improve performance on the evaluation folds? It doesn't seem so. In fact, we might have mistakenly predicted that hill-climbing would improve performance more for the clean data than for the noisy data, because the relative reductions in tuning fold WER in Table 5.4 range from 14%-28% for the clean data, but only from 0%-10% for the noisy data.

Experiment	Clean train and test		Noisy train and test	
	Initial WER	Final WER	Initial WER	Final WER
(a) HC	2.6	2.1	14.6	13.5
(b) HC, RSM initialization	2.8	2.0	14.9	13.5
(c) HC using $\alpha = 1$	2.6	2.2	14.6	13.2
(d) HC using $\alpha = 1$, RSM initialization	2.8	2.4	14.9	14.9

Table 5.4. Ensemble WERs for the tuning fold. The table rows are the same as in Table 5.2.

5.7.4 The feature vectors chosen by hill-climbing

To view diagrams showing the initial and final feature vectors in each hill-climbing experiment, visit `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/isolet-hillclimb`. You can also download the lists of initial and final features in plain text from that location.

5.8 Summary

Comparing the systems chosen by hill-climbing to the initial systems that hill-climbing started from, the only statistically significant effect of hill-climbing on the evaluation folds was to improve performance, never to worsen it (Table 5.2). These improvements were present for three out of the four hill-climbing experiments with noisy data, which was encouraging. Furthermore, for noisy data the best system chosen by hill-climbing outperformed all of the baseline systems in Table 5.3. However, for clean data hill-climbing gave no statistically significant improvements on the evaluation folds, despite the fact that on the tuning fold hill-climbing resulted in relative reductions in WER between 14%-28% for clean data.

Overall, non-random choice of the initial feature vectors for hill-climbing per-

formed better (in terms of final ensemble WER on the evaluation folds) than random initialization, and guiding hill-climbing with Equation 5.1 performed better than guiding it using ensemble WER.

Because we achieved substantial gains on the tuning fold that did not generalize to the evaluation folds, we suspect that hill-climbing performance was limited by the fact that the tuning fold used to guide hill-climbing contained a quite small amount of speech. Thus, in the next chapter, we switch from ISOLET to the Numbers corpus, which allows us to use much more speech to guide the hill-climbing process. With the Numbers corpus, hill-climbing improves over the initial system it starts from in almost all cases.

Since we are moving on to try hill-climbing with the Numbers task in the next chapter, we kept this chapter brief. We will explore the Numbers task results in more detail, because the Numbers task is more realistic (i.e., more similar to typical commercially deployed ASR applications and large-scale research ASR systems), due to its larger training set size. For example, for the Numbers task we will provide plots of hill-climbing progress over time and we will examine hill-climbing performance in mismatched conditions.

Chapter 6

Hill-climbing feature selection for the Numbers task

Contents

6.1 Introduction

The hypothesis driving our work is that ensemble performance in ASR can be improved through ensemble feature selection. In our experiments with hill-climbing using the ISOLET corpus, hill-climbing usually improved ensemble performance for noisy data, but for clean data it had no statistically significant effect. One possible reason is that we used very little data to score candidate feature vectors during hill-climbing. The ISOLET tuning fold contained about 15 minutes of audio. Using so little data to guide hill-climbing may have reduced the hill-climbing algorithm's ability to find performance gains that generalize well to evaluation data that was not used during the hill-climbing process.

Therefore, in this chapter we evaluate hill-climbing using our publicly available benchmark based on the much larger OGI Numbers corpus. We described this benchmark in detail in Chapter 3. In the rest of this chapter we will refer to the Numbers "validation" set as the "development" set, since we use it as test data for guiding the hill-climbing process, and the Numbers "test" set as the "evaluation" set, since we use it as held-out data for final performance evaluations. The Numbers development set, which we used to score candidate feature vectors during hill-climbing, contains about two hours of audio.

As with ISOLET in Chapter 5, our feature pool contained MFCC, PLP, and MSG

features calculated using a 25 ms analysis window length. We calculated MFCC features using the HTK ZMEANSOURCE option, which removes the mean of each frame of the input waveform. We used 28 MSG features, compared to 36 MSG features for ISOLET. The number of features was reduced because the sampling rate of the Numbers sampling rate data is 8000 Hz, rather than 16000 Hz as for ISOLET. This reduction is an automatic feature of the MSG calculation tools. Regardless of the feature type, we always presented MLPs with five frames of feature context.

After our experiments were complete, we noticed that we had used a suboptimal bunch size for MLP training. Details about this are given in Appendix A.

Throughout this chapter, we use a two-tailed matched pairs sign test to judge whether differences between two systems are statistically significant (see Section 3.9). We use a probability of the null hypothesis P of 0.05 as the threshold for significance.

6.2 Acoustic model size

We chose the number of acoustic model parameters we used in our Numbers experiments by measuring MFCC and PLP performance on the development set. We decided that 3600 hidden units was a reasonable choice for a single-stream MFCC or PLP system, based on Table 6.1. (That table actually suggests that 2400, 2800, 3200, or 4000 hidden units might have been a better choice. The decision to use 3600 hidden units was based on an earlier version of the table that was unaffected by the bunch size issue described in Appendix A.)

A single-stream system using 39 MFCC or PLP features with 3600 hidden units has about 806,400 acoustic model parameters (i.e., MLP weights and biases). Therefore, for the experiments in this chapter we chose the number of MLP hidden units so that the total number of acoustic model parameters in each system was about

806,400. For systems with more than one MLP, the number of hidden units for each MLP was chosen so that the number of acoustic model parameters in each MLP was approximately equal.

For the Numbers corpus, our MFCC and PLP feature vectors had 39 features and our MSG feature vector had 28 features. Thus, following the formulas given in Section 3.5, our single-MLP MFCC and PLP systems had 3600 hidden units each and our single-MLP MSG systems had 4772 hidden units. Concatenating the MFCC, PLP and MSG features into a single feature vector results in a single-MLP system with 1442 hidden units. And our three-MLP MFCC, PLP, and MSG systems used 1200 hidden units for the MFCC and PLP MLPs and 1590 hidden units for the MSG MLP.

Number of hidden units	Clean train and test			Noisy train and test		
	MFCC	PLP	Mean	MFCC	PLP	Mean
800	6.6	5.5	6.1	21.1	16.8	19.0
1200	6.5	5.3	5.9	20.3	16.5	18.4
1600	6.1	5.3	5.7	20.7	16.4	18.6
2000	6.2	5.2	5.7	19.8	17.2	18.5
2400	6.1	5.3	5.7	19.7	16.3	18.0
2800	6.0	5.2	5.6	20.0	16.2	18.1
3200	6.3	5.0	5.7	20.2	16.4	18.3
3600	6.3	5.2	5.8	20.3	16.4	18.4
4000	6.2	5.2	5.7	19.7	16.2	18.0
4400	6.0	5.3	5.7	19.7	17.1	18.4
4800	6.2	5.2	5.7	20.1	16.3	18.2
5200	6.4	5.1	5.8	19.8	16.2	18.0

Table 6.1. The table shows development set WER as a function of hidden layer size for single-MLP systems using MFCC or PLP.

6.3 Hill-climbing procedure

We used the same hill-climbing procedure that we used for ISOLET (as described in Section 5.5), apart from a few changes which are explained below.

When we did not initialize the feature vectors randomly, hill-climbing started with MFCC for the first MLP, PLP for the second MLP and MSG for the third MLP. Thus there were 39 features for each of the first two MLPs and 28 features for the third MLP. When we initialized randomly (i.e., using RSM), we also used 39 features for the first two MLPs and 28 features for the third MLP. Due to the reduction in the length of the MSG feature vector, we chose fresh random feature vectors for all three MLPs, rather than using the same random feature vectors as we did for ISOLET.

For ISOLET, we used the tuning fold both for decoder parameter tuning and for scoring candidate feature vectors during hill-climbing. For Numbers, we used the cross-validation part of the training set for decoder parameter tuning, and used the development set for scoring candidate feature vectors. We repeated decoder parameter tuning whenever we made a change to feature vectors (e.g., changing between PLP and MFCC), MLP topology (i.e., changing the number of input or hidden units), or the training condition (i.e., changing between clean and noisy training). In our experiments in the clean train and noisy test condition, we used decoder parameters determined by tuning on clean data.

The scripts we used for Numbers hill-climbing can be downloaded at `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/numbers-hillclimb`.

6.4 The time taken by hill-climbing

We parallelized our Numbers hill-climbing experiments using speculative execution, as we did for ISOLET. While running decoder parameter tuning and development set testing for the current feature being considered, we ran MLP training for the next feature to be considered in parallel. The time taken by a single iteration of the hill-climbing algorithm's inner loop was thus roughly the maximum of MLP training time on one hand and decoder tuning and development set decoding time on the other hand, plus some overhead. Sometimes trainings had to be repeated because the feature vector for the next iteration was guessed wrong by the speculative execution heuristic. As with ISOLET, the number of wrong guesses corresponds to the number of changes made to feature vectors, which is shown in Table 6.2.

Since our experiments were lengthy, we ran them using the Amazon EC2 grid computing service, which allows machines to be rented on an hourly basis. We ran each experiment (i.e., we determined each of the eight EFS "Final WER" values in Table 6.2) using two Amazon EC2 virtual machines with four cores each. (This was before EC2 offered eight-core virtual machines.) The speed of each virtual machine core was about 2 "EC2 Compute Units", where according to Amazon "One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor".

We used one virtual machine for MLP training and forward pass, and the other one for decoding. Communication between the two virtual machines was handled by using standard Unix networking tools (ssh and scp). Parallelization of MLP training over four cores was handled using the multi-threaded training capabilities built into the Quicknet library. Parallelization of decoding over four cores was handled by dividing up decoding work into pieces. For decoder parameter tuning, each piece

corresponding to different values of those parameters. For validation set testing, each piece corresponded to a different set of utterances within the validation set.

The time taken to run hill-climbing varied between experiments, ranging from 14 days to 40 days. The experiments on clean data took 14-23 days and the experiments on noisy data took 31-40 days. Of the experiments on clean data, those using ensemble WER to guide hill-climbing took 14-17 days and those using Equation 5.1 to guide hill-climbing took 22-23 days. For noisy data, the experiments using ensemble WER took 31-34 days and those using Equation 5.1 took 38-40 days. Decoder parameter tuning took much more time (roughly 80% more, on average) for noisy data than for clean data. As we discussed in Section 5.6, decoder parameter tuning was very time consuming, and reducing the amount of decoder tuning would be an interesting trade-off.

6.5 Results and discussion

6.5.1 Hill-climbing results

Table 6.2 shows our hill-climbing results. In all eight hill-climbing experiments, evaluation set WER was lower for the final ensemble chosen by hill-climbing than for the initial ensemble that hill-climbing started with. The average relative WER reduction due to hill-climbing was 6.7% for the clean data and 9.7% for the noisy data. In seven out of eight cases, the improvement was statistically significant: $P < 10^{-5}$ for the four noisy data experiments and $P < 0.006$ for the clean data experiments in row (a), (c) and (d). Thus, hill-climbing was indeed more successful for Numbers than it was for ISOLET. For the clean data experiment in row (b), the improvement was not statistically significant.

Experiment	Clean train and test			Noisy train and test		
	Changes	Initial WER	Final WER	Changes	Initial WER	Final WER
(a) Hill-climbing (HC)	4	4.9	4.6	5	15.7	14.8
(b) HC, RSM initialization	2	4.8	4.6	17	17.2	14.8
(c) HC using $\alpha = 1$	14	4.9	4.5	20	15.7	14.8
(d) HC using $\alpha = 1$, RSM initialization	17	4.8	4.4	15	17.2	14.9

Table 6.2. This table shows hill-climbing results on the Numbers corpus. The "Changes" columns give the number of features changed (added to or deleted from a feature vector) during hill-climbing. The "Initial WER" columns give the initial ensemble (i.e., three MLPs) word error rate on the evaluation set before the hill-climbing algorithm has made any changes to the feature vectors. The "Final WER" columns give the ensemble word error rate on the evaluation set once hill climbing has finished. If a value is given for α it means that the score used to guide hill-climbing was the formula defined in Equation 5.1, calculated on the development set. Otherwise, the score used to guide hill-climbing was the ensemble WER on the development set. "RSM" means that initial feature vectors prior to the start of hill-climbing were chosen randomly. Otherwise, the initial feature vectors were respectively MFCC, PLP and MSG.

There was no statistically significant difference in final evaluation set WER in Table 6.2 between the two scoring formulas used to guide hill-climbing (ensemble WER and Equation 5.1) for the clean case where RSM was not used, the noisy case where RSM was not used, or the noisy case where RSM was used. There was, however, a statistically significant difference ($P = 0.01$) in the clean case where RSM was used, with Equation 5.1 resulting in a 4.4% final ensemble WER compared to 4.6% for the other approach. It is interesting that there was no difference in the other cases, considering that there tended to be many more feature vector changes when Equation 5.1 was used (see Table 6.2).

For a given scoring formula used to guide hill-climbing (ensemble WER or Equation 5.1), it made essentially no difference to the recognition accuracy of the final ensemble in Table 6.2 whether or not we chose the initial feature vectors for hill-

climbing randomly. When we used ensemble WER to guide hill-climbing, the final evaluation set WERs were the same for the two initialization approaches (row (a) or row (b) in Table 6.2). When we used Equation 5.1 to guide hill-climbing, there were slight difference in final evaluation set WER depending on which initialization approach we used (row (c) or row (d) in Table 6.2), but these differences were not statistically significant.

6.5.2 Additional baseline results

Table 6.3 shows additional baseline results for the Numbers corpus. In both the clean and noisy cases, the lowest WER is in row (d) and comes from using all features in the feature pool concatenated together into a single feature vector. In the clean case, the system with the second lowest WER is the system in row (f) which uses three randomly chosen streams. In the noisy case, the system with the second lowest WER is the system in row (e) which uses MFCC, PLP and MSG streams. In both the clean and noisy cases, there is a statistically significant difference between the system with the lowest WER in row (d) and the system with the second lowest WER ($P = 0.01$ in the clean case and $P < 10^{-6}$ in the noisy case).

6.5.3 Hill-climbing results compared to additional baselines

Comparing Tables 6.3 and 6.2, we see that system that used all features concatenated into one feature vector has word recognition accuracy similar to the final ensembles chosen by hill-climbing. In fact, for the results in Tables 6.3 and 6.2 there is no statistically significant accuracy difference between that baseline and any of the ensembles chosen by hill-climbing. However, we will see below in Section 6.5.7 that this baseline performs worse than the ensembles chosen by hill-climbing when we mix clean training data with noisy test data.

Experiment	Clean train and test	Noisy train and test
(a) MFCC	6.5	21.4
(b) PLP	5.0	17.5
(c) MSG	7.3	16.3
(d) All features concatenated	4.5	14.7
(e) MFCC, PLP, MSG (three MLPs)	4.9	15.7
(f) RSM (three MLPs)	4.8	17.2

Table 6.3. This table shows baseline results on the Numbers corpus. The results shown are word error rates (WERs) on the evaluation set. The number of hidden units (HUs) for each MLP was chosen as explained in Section 6.2. In row (d), all features in the feature pool are concatenated into a single feature vector. In row (e), there are three MLPs respectively using 39 MFCC features (1200 HUs), 39 PLP features (1200 HUs), and 28 MSG features (1590 HUs). Row (e) corresponds to the "Initial WER" column in Table 6.2 for the experiments without RSM initialization. In row (f), there are three MLPs, using randomly chosen feature vectors with the same number of features and hidden units as in row (e). Row (f) corresponds to the "Initial WER" column in Table 6.2 for the experiments with RSM initialization.

All the other baselines in Table 6.3 had worse WERs than all the ensembles chosen by hill-climbing, for both clean and noisy data. For the noisy data, these differences were statistically significant in every case ($P < 10^{-4}$). For the clean data, the differences were statistically significant in every case when comparing the hill-climbing ensembles to the single-stream baselines that used one of MFCC, PLP or MSG ($P < 0.003$) or to the three-stream baseline using MFCC, PLP and MSG streams ($P < 0.006$ for the hill-climbing ensembles in rows (a), (c) and (d) of Table 6.2; $P = 0.04$ for the hill-climbing ensemble in row (b)). When comparing to the three-stream baseline using randomly chosen features, the difference was statistically significant for clean data in three out of four cases ($P = 0.03$ for the hill-climbing ensemble in row (a), $P = 0.04$ for the hill-climbing ensemble in row (c), and $P = 0.0001$ for the hill-climbing ensemble in row (d)).

The fact that the baseline that used all features concatenated together often performed equivalently to the much more slow and complicated hill-climbing approach

should be looked at in perspective. In Table 6.3, that baseline outperformed our ensemble baselines. Thus, in this case a single-classifier feature concatenation approach without hill-climbing outperforms an ensemble approach without hill-climbing, and this puts hill-climbing at a disadvantage from the start. There are situations in ASR where an ensemble approach outperforms a single-classifier feature concatenation approach (as in [14]), and presumably hill-climbing would be more useful there.

Also, we did not try feature selection for the single-classifier case. However, as we discussed in section 2.3.1, we speculate that improving performance by feature selection in the single-classifier case would be harder than in the ensemble case.

6.5.4 The feature vectors chosen by hill-climbing

To view diagrams showing the initial and final feature vectors in each hill-climbing experiment, visit `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/numbers-hillclimb`. You can also download the lists of initial and final features in plain text from that location.

6.5.5 Hill-climbing progress over time

Figures 6.1-6.8 show the progress of the hill-climbing algorithm over time for each experiment. Points labeled as "Initial" in the figures correspond to the initial feature vectors before any changes were made. The other points correspond to changes made to feature vectors and are labeled according to what stream was changed.

For experiments where we used ensemble WER to guide the hill-climbing process, the figures show ensemble WER on the development set and ensemble WER on the evaluation set. For experiments where we used Equation 5.1 to guide the hill-climbing process, the figures show Equation 5.1 for the development set, development

set ensemble word recognition accuracy, and evaluation set ensemble word recognition accuracy. We plot ensemble word recognition accuracy instead of ensemble WER in this case because the former is easier to plot on the same graph as Equation 5.1. Word recognition accuracy is simply 100% minus the WER.

The values of Equation 5.1 labeled "Initial" in the figures were calculated for stream 1 using the initial feature vectors for each stream. Since the value of Equation 5.1 depends on the current stream (since the equation is based on that stream's accuracy and its contribution to ensemble diversity), it's possible for it to get worse when the hill-climbing algorithm moves from one stream to the next.

Based on these plots, we conclude that hill-climbing performance on the Numbers evaluation set was not significantly lowered by overfitting [105][119][120] [121] to the development set during the hill-climbing process. In the experiments where we used development set WER to guide EFS, this can be seen by comparing development set WER to evaluation set WER in Figures 6.1-6.4. In Figures 6.1, 6.3 and 6.4, the final evaluation set WER once EFS is finished is equal to the lowest evaluation set WER in the figure. In Figure 6.2, the final evaluation set WER is greater than the minimum, but the difference is very small and is not statistically significant. In the experiments where we used Equation 5.1 to guide EFS, the plots are more difficult to interpret because the trajectory of word accuracy is much less monotonic. But because of the roughly similar ways that evaluation set word accuracy and development set word accuracy change over time, we do not think overfitting to the development set was a significant issue.

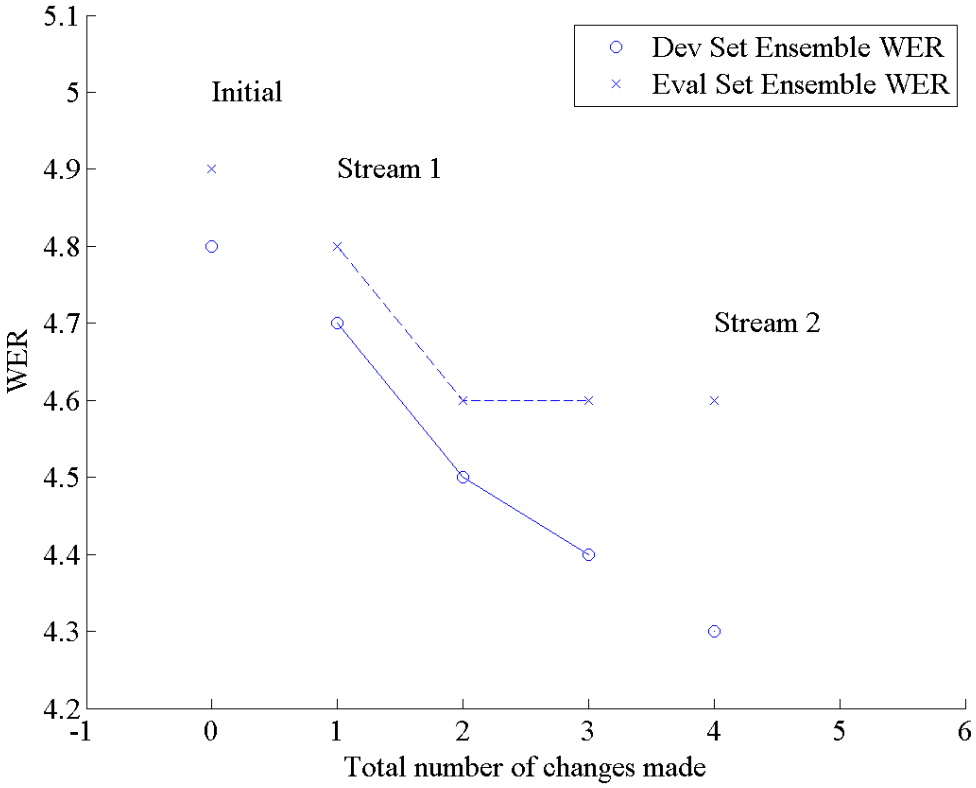

Figure 6.1. Progress of the hill-climbing algorithm over time for the clean Numbers data. The initial three streams were respectively MFCC, PLP and MSG. Candidate feature vectors were scored by ensemble WER on the development set. No changes were made to stream 3.

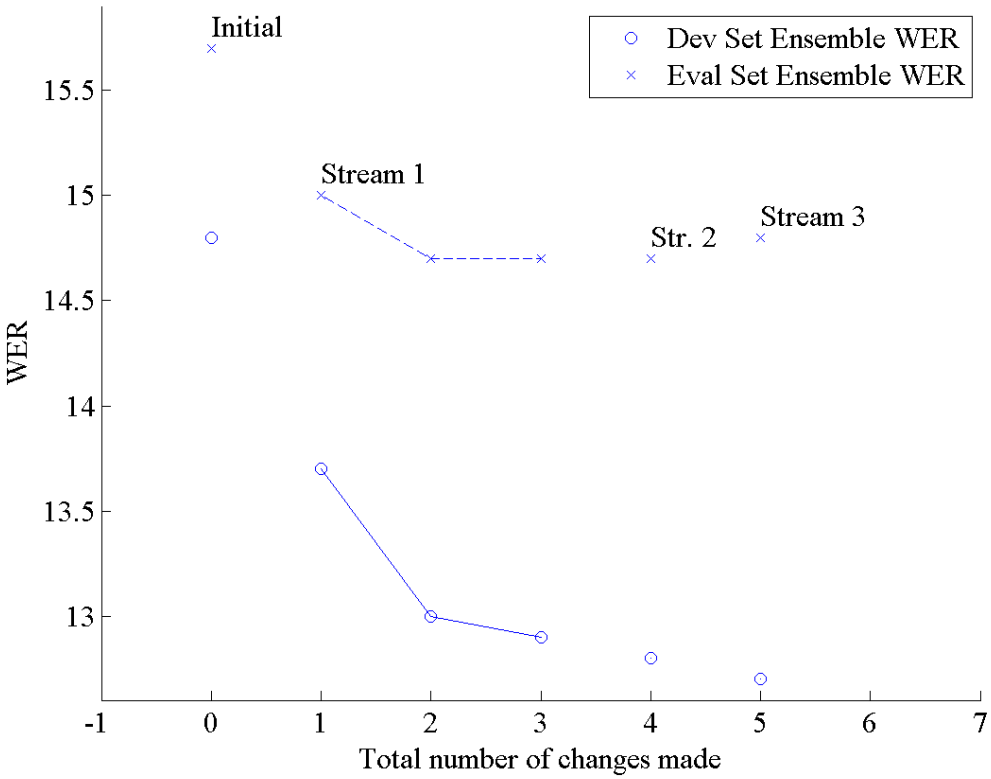

Figure 6.2. Progress of the hill-climbing algorithm over time for the noisy Numbers data. The initial three streams were respectively MFCC, PLP and MSG. Candidate feature vectors were scored by ensemble WER on the development set.

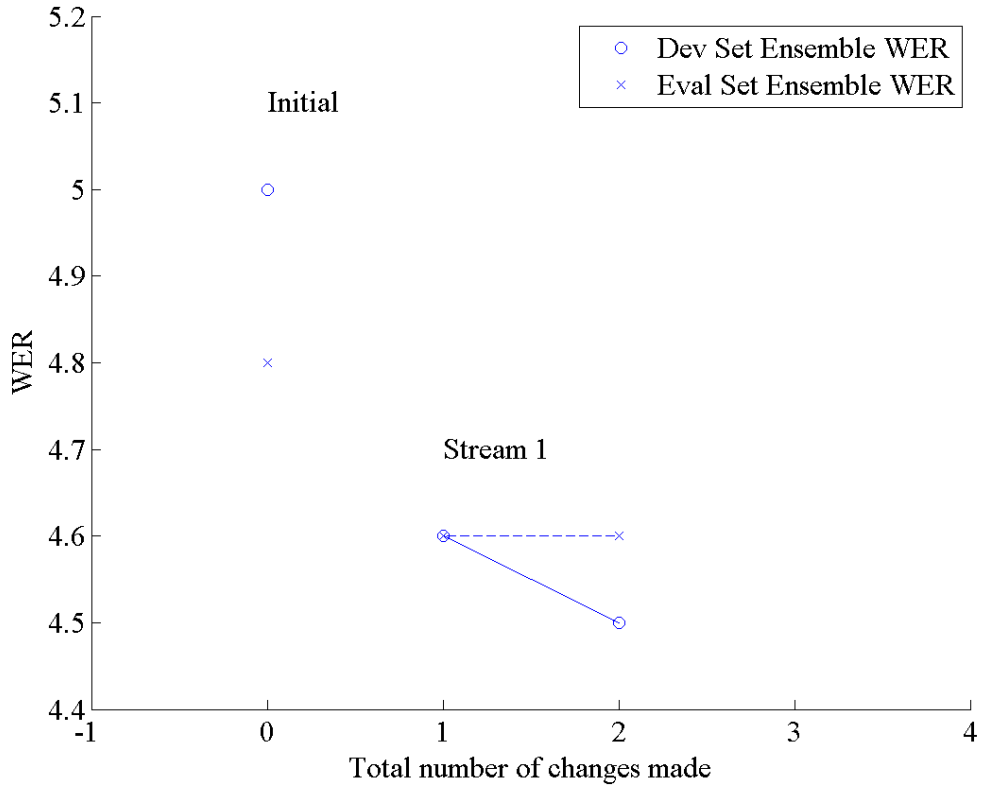

Figure 6.3. Progress of the hill-climbing algorithm over time for the clean Numbers data. The initial three streams were randomly chosen from the feature pool. Candidate feature vectors were scored by ensemble WER on the development set. No changes were made to streams 2 or 3.

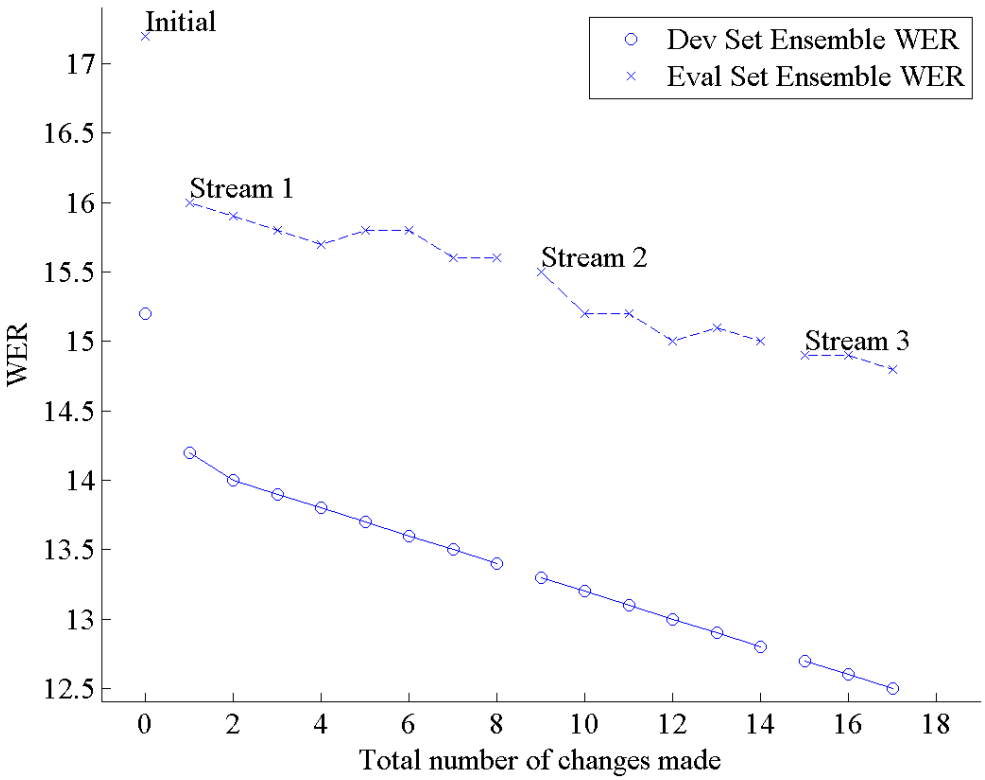

Figure 6.4. Progress of the hill-climbing algorithm over time for the noisy Numbers data. The initial three streams were randomly chosen from the feature pool. Candidate feature vectors were scored by ensemble WER on the development set.

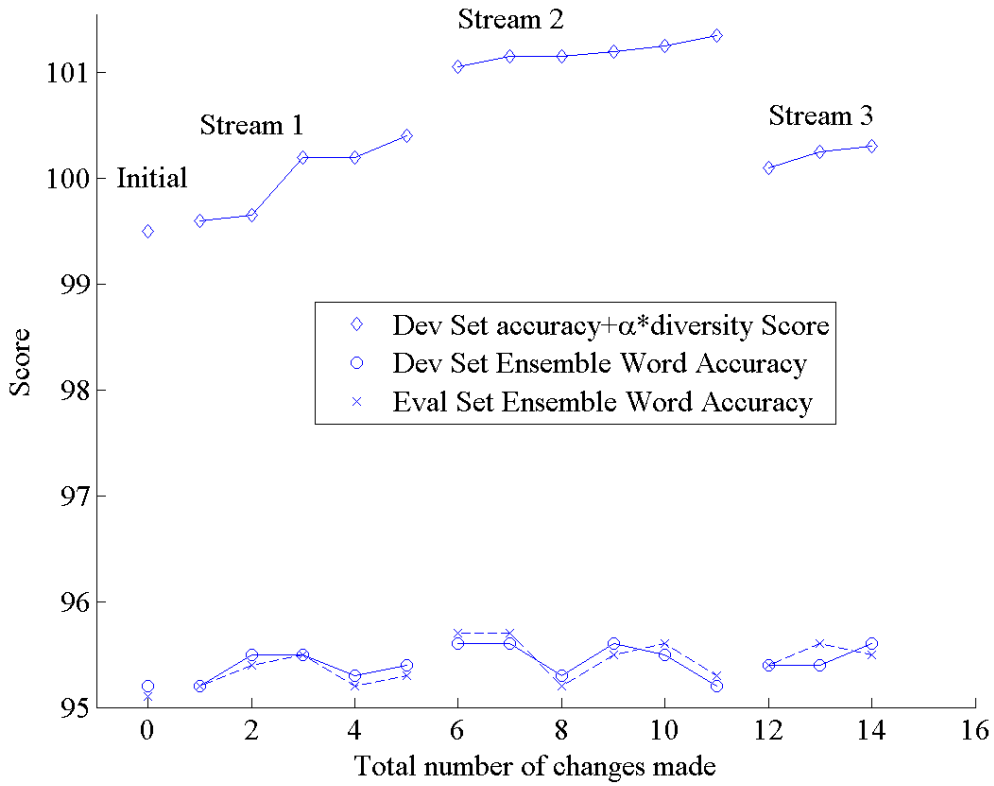

Figure 6.5. Progress of the hill-climbing algorithm over time for the clean Numbers data. The initial three streams were respectively MFCC, PLP and MSG. Candidate feature vectors were scored using Equation 5.1. The figure shows Equation 5.1 as well as ensemble word accuracy scores.

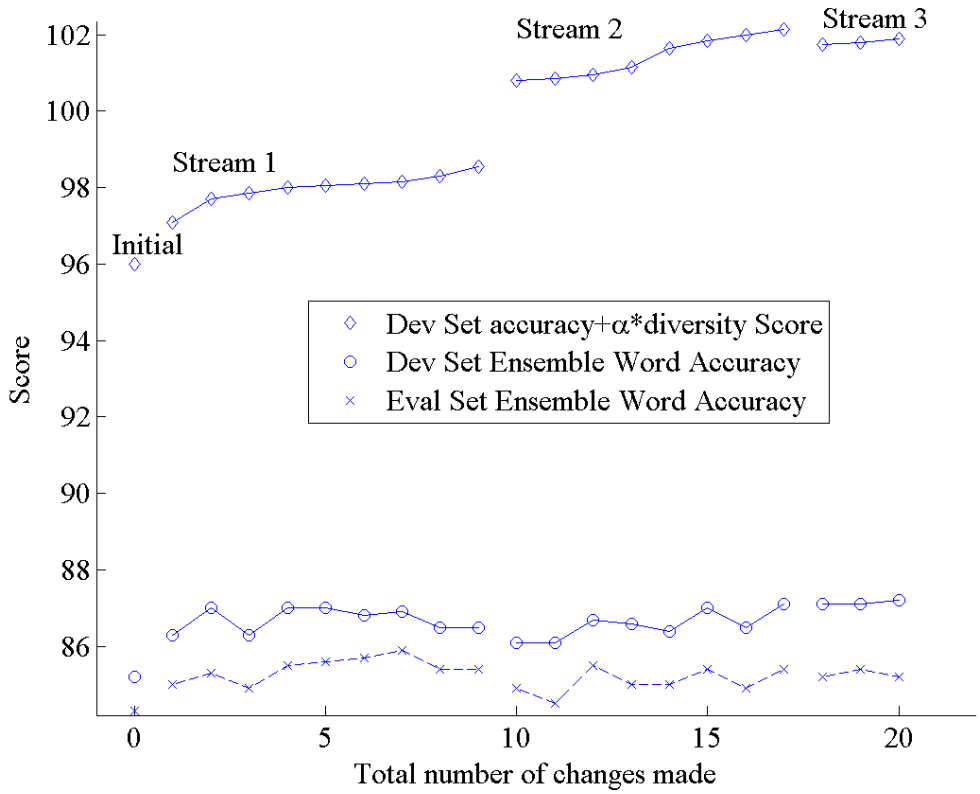

Figure 6.6. Progress of the hill-climbing algorithm over time for the noisy Numbers data. The initial three streams were respectively MFCC, PLP and MSG. Candidate feature vectors were scored using Equation 5.1.

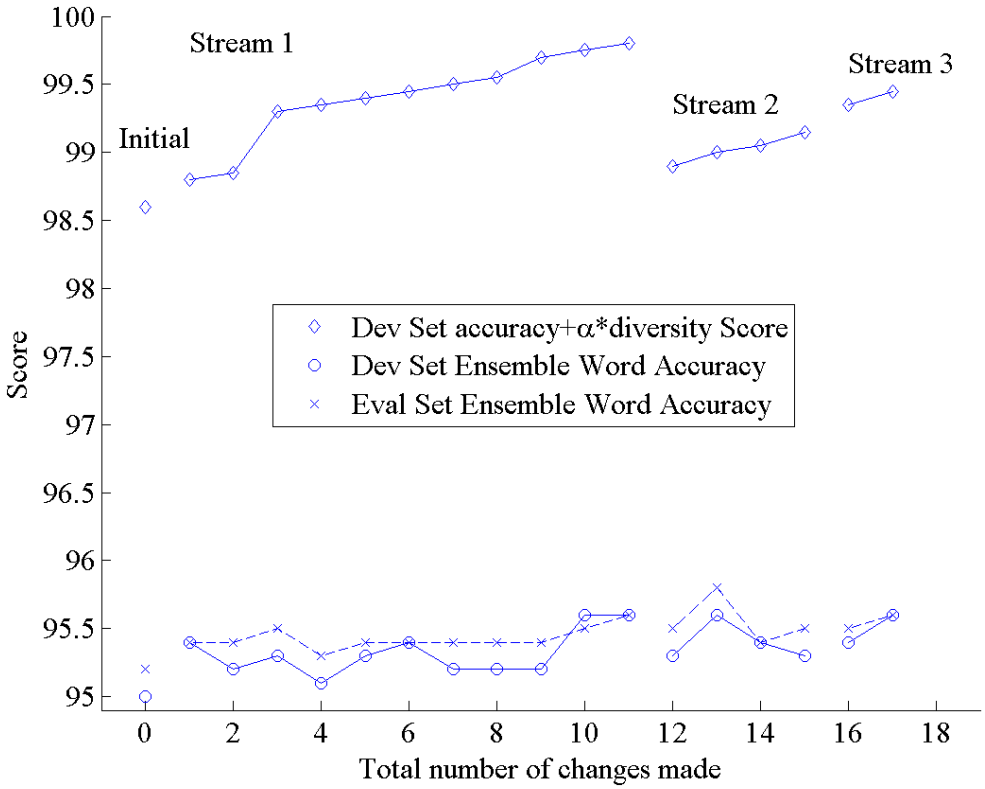

Figure 6.7. Progress of the hill-climbing algorithm over time for the clean Numbers data. The initial three streams were randomly chosen from the feature pool. Candidate feature vectors were scored using Equation 5.1.

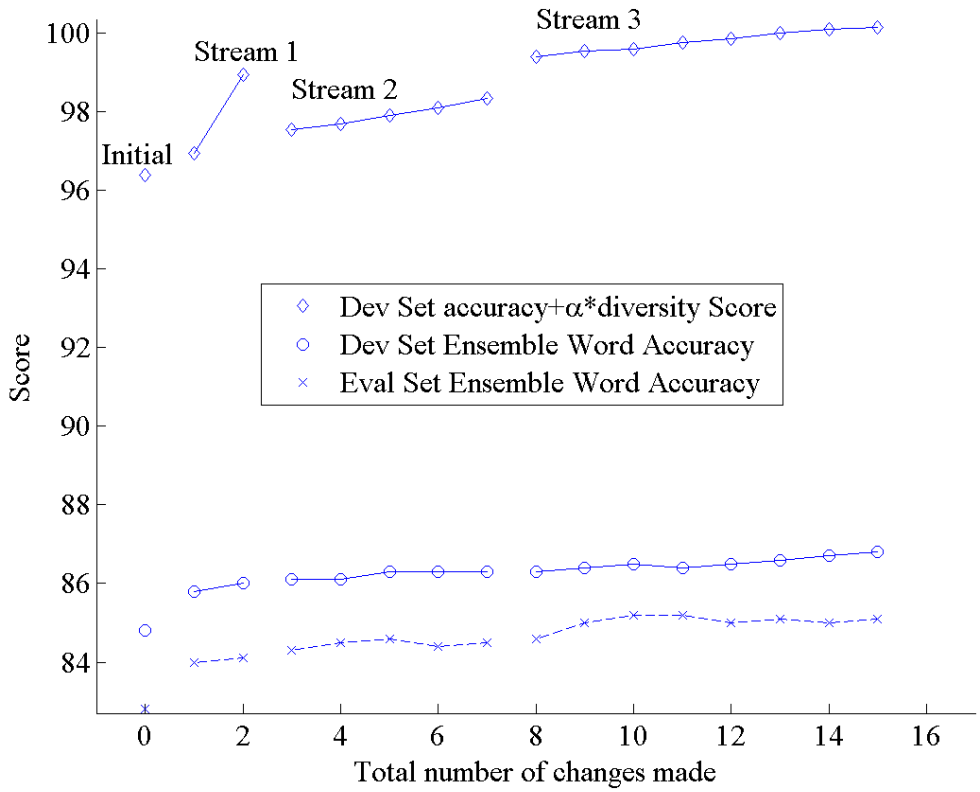

Figure 6.8. Progress of the hill-climbing algorithm over time for the noisy Numbers data. The initial three streams were randomly chosen from the feature pool. Candidate feature vectors were scored using Equation 5.1.

6.5.6 Hill-climbing improved ensemble performance even for unseen noises

As described in Section 3.8, in our noisy Numbers corpus there are four noise types that are used in the training set, the development set, and the evaluation set. There are also two noises that are only in the training set, two that are only in the development set, and two that are only in the evaluation set.

This raises the question of whether hill-climbing improved evaluation set performance for all six noise types in the evaluation set, or only for the four shared noise types. Table 6.4 compares evaluation set WERs for the four shared noises and the two evaluation-only noises. The table shows that hill-climbing improved on the systems that it started from even for the evaluation-only noises, which were not seen during the hill-climbing process. In each case, the improvement was statistically significant ($P < 0.002$) for both the shared noises and the evaluation-only noises. The differences in WER between the four systems chosen by hill-climbing were not statistically significant.

Experiment	Shared noises	Evaluation-only noises
(a) Initial three streams, non-RSM	14.1	25.5
(b) Initial three streams, chosen by RSM	15.0	28.8
(c) Hill-climbing (HC)	13.3	23.4
(d) HC, with RSM initial streams	13.3	23.6
(e) HC using $\alpha = 1$	13.1	24.2
(f) HC using $\alpha = 1$, with RSM initial streams	13.1	24.5

Table 6.4. Word error rates on the evaluation set for the noisy Numbers corpus, divided into shared noises and evaluation-only noises. The WERs for shared noises were calculated over 1970 utterances (9821 words). The WERs for evaluation-only noises were calculated over 986 utterances (4787 words). There were also 592 evaluation set utterances without any noise added.

6.5.7 Testing the features chosen by hill climbing in heavily mismatched conditions

We now investigate the question of whether the features chosen by hill-climbing are simply better features in general, or only better if used for the condition (clean or noisy) that hill-climbing was performed for.

First, we present results for which we used the clean Numbers corpus for feature selection, MLP training and decoder parameter tuning, and then tested on the evaluation set from the noisy Numbers corpus. Second, we present results for which we used the noisy Numbers corpus for feature selection, but then calculated final evaluation set results by doing MLP training and decoder parameter tuning using the clean corpus, and then testing on the evaluation set from the noisy corpus.

Using the noisy evaluation set with the clean training and development sets

Table 6.5 shows the results where we used the clean Numbers training and development sets, but the noisy Numbers evaluation set. The systems chosen by hill-climbing all performed better in this highly mismatched situation than the initial three-stream systems hill-climbing started from, and this was statistically significant in each case ($P < 0.0003$). This is further evidence that improvements in performance gained by hill-climbing can carry over to situations that were unseen during the hill-climbing process. Using Equation 5.1 to guide EFS gave much better performance in row (c) of Table 6.5 than using ensemble WER to guide EFS (rows (a) and (b)), and this was statistically significant in both cases ($P < 10^{-42}$).

Table 6.6 shows the corresponding baseline results. For these MLP training and decoder parameter tuning was done using the clean Numbers training set, and testing

was done using the noisy evaluation set. The single-stream MSG baseline has the lowest WER (30.4% WER in row (c)). The three-stream baselines perform nearly identically (30.5% in rows (e) and (f)). In the baseline results presented in Table 6.3 in section 6.5.2, the lowest WER came from using all features in the feature pool concatenated together into a single feature vector, but in Table 6.6 that approach is outperformed both by the three-MLP baselines (this was statistically significant, $P < 10^{-8}$) and by the single-MLP MSG baseline (also statistically significant, $P = 0.01$). This reversal of the previous relationship between the feature concatenation system and the three-MLP baseline systems is consistent with the theory that multi-classifier systems can have an advantage over feature concatenation in mismatched conditions [14].

Comparing Table 6.5 to Table 6.6, we see that in this highly mismatched condition the worst hill-climbing result (29.0% WER in row (a) in Table 6.5) is better than the best baseline result (30.4% WER in row (c) in Table 6.6). This is statistically significant ($P < 10^{-9}$). This again supports the idea that hill-climbing can be useful even in mismatched conditions.

Experiment	Initial WER	Final WER
(a) Hill-climbing (HC)	30.5	29.0
(b) HC, RSM initialization	30.5	28.2
(c) HC using $\alpha = 1$	30.5	23.4
(d) HC using $\alpha = 1$, RSM init.	30.5	27.9

Table 6.5. Hill-climbing WER results on the evaluation set in a highly mismatched condition: the clean training and development sets were used with the noisy evaluation set. "Initial WER" refers to the WER for the initial features that the hill-climbing process started with. "Final WER" refers to the WER of the final features chosen by hill-climbing.

Experiment	WER
(a) MFCC	33.2
(b) PLP	38.0
(c) MSG	30.4
(d) All features concatenated	32.1
(e) MFCC, PLP, MSG (three streams)	30.5
(f) RSM (three streams)	30.5

Table 6.6. Baseline WERs on the evaluation set in a highly mismatched condition: the clean training set was used with the noisy evaluation set. The feature vectors and MLP hidden layer sizes are the same as in Table 6.3.

Using features from hill-climbing on noisy data with the clean training set and the noisy evaluation set

Table 6.7 shows the results of using features selected by hill-climbing for the noisy data, but then calculating evaluation set WER by performing MLP training and decoder tuning on the clean training set and then testing on the noisy evaluation set. Overall, the results are worse than in Table 6.5. The final WER in each row of Table 6.7 is worse than the final WER in the corresponding row of Table 6.5, and this is statistically significant ($P < 10^{-6}$ in rows (a), (b) and (c); $P = 0.03$ in row (d)).

In Table 6.7, the ensembles chosen by hill-climbing performed better than the initial ensembles hill-climbing started from only in the cases where Equation 5.1 was used to score candidate feature vectors (rows (c) and (d) in Table 6.7). In those two cases, the improvements were statistically significant ($P < 10^{-11}$). The hill-climbing results in row (c) and (d) are better than the best baseline result in Table 6.6 (30.4% WER in row (c)), and this is statistically significant ($P < 10^{-13}$). There is no statistically significant difference in WER between row (c) and row (d). That is quite different from Table 6.5, in which row (c) was much better than row (d).

When ensemble WER was used instead of Equation 5.1, Table 6.7 shows that hill-climbing actually worsened performance from 30.5% WER to 31.9% WER when

the initial three streams were chosen randomly (this was statistically significant, $P < 10^{-8}$) and hill-climbing made no difference to WER when the initial streams were not chosen randomly.

Experiment	Initial WER	Final WER
(a) Hill-climbing (HC)	30.5	30.5
(b) HC, RSM initialization	30.5	31.9
(c) HC using $\alpha = 1$	30.5	28.2
(d) HC using $\alpha = 1$, RSM init.	30.5	28.5

Table 6.7. Hill-climbing WER results on the evaluation set in a highly mismatched condition: hill-climbing was performed using the noisy data, but evaluation set WER was measured by performing MLP training and decoder tuning on the clean training set and then testing on the noisy evaluation set. "Initial WER" refers to the WER for the initial features that the hill-climbing process started with. "Final WER" refers to the WER of the final features chosen by hill-climbing.

6.5.8 How different are the different systems chosen by hill-climbing?

Word hypothesis disagreement rate

How much do the ensembles chosen by hill-climbing differ between hill-climbing experiments? One way of approaching this question is to measure how different the word hypotheses produced by these ensembles are. We can do this by calculating the WER of one system using another system's output as if it were the ground truth transcript. Since this is not a true WER, we will call it a "word disagreement rate" (WDR). The pairwise diversities that we calculate when computing Equation 5.1 are calculated the same way; see Section 5.5 for details. WDR can be calculated very easily using any WER scoring tool. Tables 6.8 and 6.9 summarize WDRs calculated on the evaluation set. Each table is in the form of a matrix showing the WDR between the system identified by the row and the system identified by the column.

Tables 6.8 is for the clean Numbers corpus and Table 6.9 is for the noisy Numbers corpus. (We did not calculate WDR results for the mismatched clean training and noisy test conditions in Section 6.5.7.)

The WDR tables shows that the disagreements are considerable. Comparing the WDRs in Tables 6.8 and 6.9 to the WERs in Table 6.2, we see that the four systems selected by hill-climbing for the clean data have WERs of 4.4-4.6% and WDRs (with each other) of 1.6-2.5%, and the four systems selected for the noisy data have WERs of 14.8-14.9% and WDRs (again, with each other) of 5.7-8.4%. So for both the clean and noisy data, the WDRs are all more than a third of the relevant WERs, and in some cases they are more than half.

Considering that the WDRs are fairly large, it seems likely that further performance gains could be obtained by combining different systems chosen by hill-climbing, or combining one or more systems chosen by hill-climbing with one or more of the initial systems that EFS started from. Such combining could be done at the frame level as with the system combination done in this thesis, or at the word hypothesis level using techniques such as ROVER or confusion network combination [39][40] [41]. For more on the relationship between WDR and the performance of system combination, see the discussion of the related concept of "dependent" or "Identical-Incorrect" errors in [17][13] and Chapter 6 of [86].

Experiment	(a)	(b)	(c)	(d)	(e)	(f)
(a) Initial three streams	0.0	2.4	2.5	3.2	2.7	3.3
(b) Initial three streams, RSM		0.0	3.0	2.6	3.2	2.8
(c) Hill-climbing (HC)			0.0	2.3	1.9	2.3
(d) HC, RSM initialization				0.0	2.4	1.6
(e) HC using $\alpha = 1$					0.0	2.5
(f) HC using $\alpha = 1$, RSM init.						0.0

Table 6.8. Word disagreement rate (WDR) matrix for clean Numbers corpus.

Experiment	(a)	(b)	(c)	(d)	(e)	(f)
(a) Initial three streams	0.0	10.7	7.9	9.9	8.3	9.9
(b) Initial three streams, RSM		0.0	9.2	7.4	9.7	6.7
(c) Hill-climbing (HC)			0.0	8.0	6.2	7.9
(d) HC, RSM initialization				0.0	8.4	5.7
(e) HC using $\alpha = 1$					0.0	8.2
(f) HC using $\alpha = 1$, RSM init.						0.0

Table 6.9. Word disagreement rate (WDR) matrix for noisy Numbers corpus.

Feature difference counts

Another way to measure the differences between systems chosen by hill-climbing is to compare feature vectors. Tables 6.10-6.13 count how many features are in common between pairs of systems and how many are not in common. We have used four tables due to the amount of information presented. Each table is in the form of a matrix showing the counts of features in common and not in common between the system identified by the row and the system identified by the column. Each table cell gives feature difference counts for all three streams. Stream 1 of the first system is compared to stream 1 of the second system, then stream 2 to stream 2, and finally stream 3 to stream 3. Each comparison is broken down into three numbers which are separated by slashes: the number of features in both the system named to the left of the cell and the system named above the cell, the number of features in only the system named to the left, and the number of features in only the system named above. The comparisons for different streams are separated by commas.

Comparing Tables 6.10-6.13 to Tables 6.8-6.9, we observe that just a few differences in features can result in a considerable WDR between systems.

Experiment	(a)	(b)	(c)
(a) Initial three streams	39/0/0, 39/0/0, 28/0/0	36/3/0, 38/1/0, 28/0/0	35/4/1, 37/2/4, 28/0/3
(b) Hill-climbing (HC)		36/0/0, 38/0/0, 28/0/0	33/3/3, 36/2/5, 28/0/3
(c) HC using $\alpha = 1$			36/0/0, 41/0/0, 31/0/0

Table 6.10. Counts of feature differences between feature vectors selected by hill-climbing on the clean Numbers corpus, without random initialization of feature vectors.

Experiment	(a)	(b)	(c)
(a) Initial three streams, RSM	39/0/0, 39/0/0, 28/0/0	39/0/2, 39/0/0, 28/0/0	37/2/7, 37/2/2, 28/0/2
(b) Hill-climbing (HC), RSM initialization		41/0/0, 39/0/0, 28/0/0	38/3/6, 37/2/2, 28/0/2
(c) HC using $\alpha = 1$, RSM initialization			44/0/0, 39/0/0, 30/0/0

Table 6.11. Counts of feature differences between feature vectors selected by hill-climbing on the clean Numbers corpus, with random initialization of feature vectors.

6.6 Summary

When we ran experiments using either the clean or noisy Numbers corpus, hill-climbing almost always improved performance compared to the initial ensembles that hill-climbing started with (Table 6.2), even for noises types that were unseen during the hill-climbing process (Table 6.4). This confirms our hypothesis that appropriate use of ensemble feature selection will improve ensemble performance in ASR. However, the ensembles obtained by hill-climbing performed the same as a baseline system that used the entire feature pool with a single MLP (Table 6.3). Later, when we used the clean Numbers training and development sets but measured final performance on the noisy Numbers evaluation set, we found that the ensembles chosen by hill-climbing outperformed all baseline systems (Table 6.5 and Table 6.6). When we used features selected by hill-climbing for the noisy data, but then measured final performance

Experiment	(a)	(b)	(c)
(a) Initial three streams	39/0/0, 39/0/0, 28/0/0	36/3/0, 39/0/1, 28/0/1	31/8/1, 38/1/7, 27/1/2
(b) Hill-climbing (HC)		36/0/0, 40/0/0, 29/0/0	30/6/2, 38/2/7, 27/2/2
(c) HC using $\alpha = 1$			32/0/0, 45/0/0, 29/0/0

Table 6.12. Counts of feature differences between feature vectors selected by hill-climbing on the noisy Numbers corpus, without random initialization of feature vectors.

Experiment	(a)	(b)	(c)
(a) Initial three streams, RSM .	39/0/0, 39/0/0, 28/0/0	34/5/3, 37/2/4, 28/0/3	38/1/1, 38/1/4, 25/3/5
(b) Hill-climbing (HC), RSM initialization		37/0/0, 41/0/0, 31/0/0	35/2/4, 36/5/6, 25/6/5
(c) HC using $\alpha = 1$, RSM initialization			39/0/0, 42/0/0, 30/0/0

Table 6.13. Counts of feature differences between feature vectors selected by hill-climbing on the noisy Numbers corpus, with random initialization of feature vectors.

by performing MLP training and decoder tuning on the clean training set and then testing on the noisy evaluation set, we found that the ensembles chosen by hill-climbing outperformed all baseline systems when Equation 5.1 was used to guide hill-climbing, but worsened or did not change performance when ensemble WER was used to guide hill-climbing (Table 6.7).

Overall, hill-climbing performed better when guided with Equation 5.1 rather than ensemble WER, and when feature vectors were initialized non-randomly rather than randomly.

Chapter 7

Conclusions

Contents

7.1 Summary and contributions

Multi-stream ASR systems have become popular among researchers, but only a small number of papers have been published on the topic of ensemble feature selection

(EFS) for such systems, and those have worked with blocks of features rather than individual features. In this thesis, we have demonstrated that the performance of multi-stream ASR systems can be improved by using ensemble feature selection at the level of individual features.

Initially, we used the OGI ISOLET task (both the original, clean version and our own noise-added version) to evaluate the random subspace method (RSM). RSM, well known in the pattern recognition community, simply chooses each feature vector randomly from the feature pool. We used 25 classifiers in our RSM ensembles, and we fixed the number of features per classifier at 39, a common feature vector length for ASR front ends. We experimented with nine feature pools and three different train/test conditions (clean train with clean test, noisy train with noisy test, and clean train with noisy test). For a given feature pool and train/test condition, we found RSM improved performance over baseline systems in all but one of the cases where there was a statistically significant performance difference. However, there were many cases with no such difference. Statistically significant differences were more common for the clean train and noisy test condition than for the other two conditions. When we compared the best RSM system to the best baseline systems without restricting ourselves to a particular feature pool, RSM did not give a statistically significant performance improvement over baseline in any of the three train and test data conditions.

We then moved on to evaluate hill-climbing, a guided approach to EFS in which initially chosen feature vectors are iteratively improved by adding or removing one feature at a time if the change improves performance. Since hill-climbing is much slower than the random subspace method, in our hill-climbing experiments we used only one feature pool, which contained MFCC, PLP and MSG features. Also, we used three MLPs per ensemble instead of 25. We started by evaluating hill-climbing on the ISOLET task. We tried four different variations of hill-climbing with the original

ISOLET task and our noisy version of it, for a total of eight experiments. The variations were whether Equation 5.1 or ensemble WER was used as the performance score to guide hill-climbing, and whether the initial three feature vectors were chosen by RSM or by giving each feature type (MFCC, PLP, or MSG) its own feature vector. We found that for our noisy version of ISOLET, hill-climbing improved performance on held-out data (compared to the three-stream system that hill-climbing started from) in three out of four experiments, with relative WER reductions of 5-6%. There was no statistically significant difference in the remaining experiment. However, for the original, clean version of ISOLET hill-climbing made no statistically significant difference to performance on held-out data. This is despite the fact that relative reductions in WER between 14%-28% were achieved for the data used to guide hill-climbing. Because these gains did not generalize to the held out data, we suspected that hill-climbing performance was limited by the very small amount of data used to guide the hill-climbing process. Thus, we switched from the ISOLET task to the OGI Numbers task, which allowed us to use much more speech to guide hill-climbing process.

For the Numbers task, hill-climbing improved performance (compared to the three-stream system hill-climbing started from) in seven out of eight cases when we ran experiments using either the clean or noisy Numbers corpus, giving relative WER reductions of 6-8% compared to non-RSM initial feature vectors and 8-14% compared to RSM initial feature vectors. (In the eighth case, which used RSM initial feature vectors with the clean corpus, there was no statistically significant performance difference.) With the noisy corpus, hill-climbing improved performance even for noise types that were unseen during the hill-climbing process. However, for both the clean and noisy data the systems obtained by hill-climbing performed the same as the best non-ensemble baseline system, which used the entire feature pool in one feature vector.

In later Numbers experiments we investigated performance in mismatched conditions by using clean data for training MLPs, tuning decoder parameters and guiding hill-climbing but using the noisy evaluation set to measure final performance, or alternatively using the features selected by hill-climbing for the noisy data, but then measuring final performance by performing MLP training and decoder parameter tuning on the clean training set and then testing on the noisy evaluation set. In these experiments, we found that the ensembles chosen by hill-climbing outperformed all ensemble and non-ensemble baseline systems when Equation 5.1 was used to guide hill-climbing.

The hill-climbing algorithm, as defined in [25], scores candidate feature vectors with Equation 5.1, which combines single-classifier accuracy with a measure of that classifier's contribution to ensemble diversity. Since the eventual goal is to improve ensemble accuracy, we compared the effectiveness of using Equation 5.1 to using ensemble accuracy (or ensemble WER, which is equivalent). Overall, we found that guiding hill-climbing using Equation 5.1 indeed performed better than guiding it using ensemble WER in our ISOLET and Numbers experiments.

As a byproduct of our work on ensemble feature selection, we have placed several resources online that can be reused by other researchers. We created noisy versions of the OGI ISOLET and Numbers corpora using various noises and signal-to-noise ratios, and put scripts and noise files online [122][123] so that others can reproduce these noisy corpora. This is useful since past publications on noisy ASR using these corpora have used various different and unpublished noisy versions, making results more difficult to compare across papers. It also is a time saver for other researchers. We have also made scripts and configuration files available [122][123] for ISOLET and Numbers ASR systems based on open source software, which are useful for the same reasons. Finally, we have made our ensemble feature selection scripts and lists of selected features available online, to make it easier for future work on EFS for

ASR to compare to or build on our work. Those are online at `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/numbers-hillclimb` and `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/isolet-hillclimb`, and we may use those same locations in the future if we need to publish updates or corrections regarding the work described in this thesis.

7.2 Possible future directions

Our work on feature selection for multi-stream ASR at the level of individual features is a first step which hopefully could be improved on by others. This section points out some possible directions for future work.

7.2.1 Different ensemble sizes

If there were more classifiers in the ensemble, the hill-climbing algorithm would have more options to choose from when distributing features, which might result in greater accuracy gains. We are currently running experiments on the Numbers corpus in which we use five classifiers with the same MFCC, PLP and MSG feature pool we previously used with three classifiers. The first classifier uses the 13 static MFCC features, the second uses the 26 dynamic (delta and delta-delta) MFCC features, the third uses the 13 static PLP features, the fourth uses the 26 dynamic PLP features, and the fifth uses the 28 MSG features. We plan to publish the final results in a conference paper or technical report or at `http://www.icsi.berkeley.edu/speech/papers/gelbart-ms/numbers-hillclimb`.

7.2.2 A larger feature pool

A larger feature pool also might make hill-climbing more useful. The authors of [79] and [17][13] explored feature selection using feature pools containing hundreds of features, all of which can be calculated using tools that are freely available online [124]. And there are many other possible features. For example, the ETSI Aurora standards process produced two carefully engineered feature extraction front ends for noise-robust ASR [61][125] for which the source code is available online [126] for research use.

7.2.3 Doing less decoder parameter tuning

During hill-climbing, we re-ran decoder parameter tuning every time we made any change to a feature vector, which was very time consuming. As we discussed in Section 5.6, there are several possible ways to reduce the amount of time spent on decoder parameter tuning, and it would be interesting to know what effect, if any, this would have on the usefulness of the final feature vectors chosen by hill-climbing.

7.2.4 Different versions of the random subspace method

We used the random subspace method on its own in Chapter 4 and as a starting point for hill-climbing in later chapters. We always used 39 features per classifier in our RSM ensembles. As discussed in Section 4.6, there are other possibilities which could be worth investigating.

7.2.5 Generalizing the gains from hill-climbing

Our Numbers ASR systems perform comparably to the best published results we are aware of, but we know of an unpublished result that is much better (see Section 3.10.2). That raises the question of whether performance gains from hill-climbing on the Numbers corpus would still be achieved if we started from a truly state-of-the-art Numbers system.

Investigating how easily hill-climbing can be scaled up to larger-scale ASR systems is a related topic. As a wrapper approach, hill-climbing might be impractically slow for a large-vocabulary ASR system trained on thousands of hours of data. This raises the question of whether it would be useful to run hill-climbing on a smaller, faster task and then use the selected features for the larger task.

7.2.6 Tuning α

As discussed in Section 5.5, it might be possible to improve hill-climbing performance by adjusting the value of α in Equation 5.1.

7.2.7 Alternative ways to score candidate ensembles

When scoring candidate ensembles with Equation 5.1 during hill-climbing, we calculated diversity by averaging word-level pairwise diversities (see Section 5.5). Diversity measures for ASR in particular are discussed in [13][17][16]. and in Chapter 6 of [86]. Various possible diversity measures for classifier ensembles in general are discussed in [25][127].

The use of classification margin [127] instead of accuracy might also be worth investigating. Perhaps margin would be less vulnerable to overfitting [128]. There has been successful recent work using margin in ASR model training [129][130][131].

7.2.8 Genetic algorithms

It might be possible to improve performance by replacing hill-climbing with another guided, wrapper-style search approach. A genetic algorithm would be a reasonable choice. Tsymbal et al. showed in [25] that a genetic algorithm approach to EFS can outperform hill-climbing, and in [132] they proposed a newer genetic algorithm approach which often improved performance over the one in [25].

Bibliography

[1] B. Meyer, T. Wesker, T. Brand, A. Mertins, and B. Kollmeier, "A human-machine comparison in speech recognition based on a logatome corpus," in *Speech Recognition and Intrinsic Variation (SRIV2006)*, Toulouse, France, 2006.

[2] B. Meyer, M. Wächter, T. Brand, and B. Kollmeier, "Phoneme confusions in human and automatic speech recognition," in *INTERSPEECH*, Antwerp, Belgium, 2007.

[3] X. Huang, A. Acero, and H. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*, Prentice Hall, 2001.

[4] D. Jurafsky and J. Martin, *Speech and Language Processing*, Prentice Hall, second edition, 2008.

[5] B. Gold and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, Wiley, 1999.

[6] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, 1989.

[7] M. Shire, *Discriminant training of front-end and acoustic modeling stages to heterogeneous acoustic environments for multi-stream automatic speech recognition*, Ph.D. thesis, University of California Berkeley, 2000, online at `http://www.icsi.berkeley.edu`.

[8] A. Sharkey, "Types of multinet system," in *Multiple Classifier Systems*, Cagliari, Italy, 2002.

[9] T. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, Cagliari, Italy, 2000.

[10] T. Dietterich, *The Handbook of Brain Theory and Neural Networks*, chapter Ensemble Learning, MIT Press, second edition, 2002.

[11] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in ensemble feature selection," Tech. Rep. TCD-CS-2003-44, Trinity College Dublin, 2003, available at `http://www.cs.tcd.ie`.

[12] David Opitz, "Feature selection for ensembles," in *Sixteenth National Conference on Artificial Intelligence (AAAI)*, Orlando, Florida, 1999.

[13] L. Burget, *Complementarity of speech recognition systems and system combination*, Ph.D. thesis, Brno University of Technology, Czech Republic, 2004, available at `http://speech.fit.vutbr.cz`.

[14] D. Ellis, "Stream combination before and/or after the acoustic model," Tech. Rep. 00-007, International Computer Science Institute, 2000, available at `http://www.icsi.berkeley.edu`.

[15] H. Christensen, B. Lindberg, and O. Andersen, "Employing heterogeneous information in a multi-stream framework," in *ICASSP*, Istanbul, Turkey, 2000.

[16] D. Ellis and J. Bilmes, "Using mutual information to design feature combinations," in *ICSLP*, Beijing, China, 2000.

[17] L. Burget, "Measurement of complementarity of recognition systems," in *Seventh International Conference on Text, Speech and Dialogue (TSD)*, Brno, Czech Republic, 2004.

[18] H. Misra and H. Bourlard, "Spectral entropy feature in full-combination multi-stream for robust ASR," in *INTERSPEECH*, Lisbon, Portugal, 2005.

[19] W. Abdulla and N. Kasabov, "Reduced feature-set based parallel CHMM speech recognition systems," *Information Sciences*, November 2003.

[20] A. Hagen and A. Morris, "Recent advances in the multi-stream HMM/ANN hybrid approach to noise robust ASR," *Computer Speech and Language*, vol. 19, no. 1, 2005.

[21] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, 1998.

[22] T. K. Ho, "Nearest neighbors in random subspaces," in *Second International Workshop on Statistical Techniques in Pattern Recognition*, Sydney, Australia, 1998.

[23] M. Skurichina and R. Duin, "Bagging, boosting and the random subspace method for linear classifiers," *Pattern Analysis and Applications*, vol. 5, no. 2, 2002.

[24] S. Günter and H. Bunke, "Ensembles of classifiers derived from multiple prototypes and their application to handwriting recognition," in *Multiple Classifier Systems*, Cagliari, Italy, 2004.

[25] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, no. 1, 2005.

[26] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, December 1997.

[27] D. O'Shaughnessy, "Automatic speech recognition: History, methods and challenges," *Pattern Recognition*, vol. 41, no. 10, 2008.

[28] D. Jurafsky, "Slides from ASR classes taught at Stanford," `http://www.stanford.edu/class/cs224s` and `http://www.stanford.edu/class/linguist236`.

[29] S. Chen, M. Picheny, E. Eide, and G. Potamianos, "Slides from an ASR class taught at Columbia by IBM researchers," `http://www.ee.columbia.edu/~stanchen/e6884/outline.html`, 2005.

[30] B. Pellom, "Slides from an ASR class at the Helsinki University of Technology," `http://www.cis.hut.fi/Opinnot/T-61.6040/pellom-2004`.

[31] J. Glass and V. Zue, "Slides from an ASR class at the Massachusetts Institute of Technology," `http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-345Automatic-Speech-RecognitionSpring2003/CourseHome`.

[32] M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, and C. Wellekens, "Automatic speech recognition and speech variability: A review," *Speech Communication*, vol. 49, no. 10-11, 2007.

[33] N. Morgan and H. Bourlard, "Continuous speech recognition: an introduction to the hybrid HMM/connectionist approach," *IEEE Signal Processing Magazine*, May 1995, but note that the IEEE Xplore index lists the title as just the three words continuous speech recognition.

[34] H. Bourlard and N. Morgan, *Connectionist Speech Recognition - A Hybrid Approach*, Kluwer Academic Press, 1994.

[35] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1996.

[36] B. Chen, *Learning discriminant narrow-band temporal patterns for automatic recognition of conversational telephone speech*, Ph.D. thesis, University of California Berkeley, 2005, online at `http://www.icsi.berkeley.edu`.

[37] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, April 1990.

[38] B. Kingsbury, *Perceptually-inspired signal processing strategies for robust speech recognition in reverberant environments*, Ph.D. thesis, University of California Berkeley, 1998, available at `http://www.icsi.berkeley.edu`.

[39] J. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Santa Barbara, California, 1997.

[40] G. Evermann and P. Woodland, "Posterior probability decoding, confidence estimation and system combination," in *NIST Speech Transcription Workshop*, College Park, Maryland, 2000.

[41] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. Gadde, M. Plauché, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng, "The SRI March 2000 Hub-5 conversational speech transcription system," in *NIST Speech Transcription Workshop*, College Park, Maryland, 2000.

[42] R. Singh, M. Seltzer, B. Raj, and R Stern, "Speech in noisy environments: Robust automatic segmentation, feature extraction and hypothesis combination," in *ICASSP*, Salt Lake City, Utah, 2001.

[43] C. Breslin and M. Gales, "Complementary system generation using directed decision trees," in *ICASSP*, Honolulu, Hawaii, 2007.

[44] X. Li and R. Stern, "Parallel feature generation based on maximizing normalized acoustic likelihood," in *ICSLP*, Jeju Island, Korea, 2004.

[45] X. Li, *Combination and generation of parallel feature streams for improved speech recognition*, Ph.D. thesis, Carnegie Mellon University, 2005, available at `http://www.cs.cmu.edu/~robust`.

[46] A. Halberstadt and J. Glass, "Heterogeneous measurements and multiple classifiers for speech recognition," in *ICSLP*, Sydney, Australia, 1998.

[47] A. Halbertstadt, *Heterogeneous acoustic measurements and multiple classifiers for speech recognition*, Ph.D. thesis, Massachusetts Institute of Technology, 1998, available at `http://groups.csail.mit.edu/sls`.

[48] D. Ellis, "Improved recognition by combining different features and different systems," in *Applied Voice Input/Output Society (AVIOS)*, San Jose, California, 2000, available at `http://www.icsi.berkeley.edu`.

[49] Q. Zhu, A. Stolcke, B. Chen, and N. Morgan, "Using MLP features in SRI's conversational speech recognition system," in *INTERSPEECH*, Lisbon, Portugal, 2005.

[50] F. Valente, J. Vepa, and H. Hermansky, "Multi-stream features combination based on Dempster-Shafer rule for LVCSR System," in *INTERSPEECH*, Antwerp, Belgium, 2007.

[51] Hervé Bourlard and Stéphane Dupont, "Sub-band-based speech recognition," in *ICASSP*, Munich, Germany, 1997.

[52] S. Dupont, *Etude et développement d'architectures multi-bandes et multi-modales pour la reconnaissance robuste de la parole*, Ph.D. thesis, Faculté Polytechnique de Mons, June 2000.

[53] N. Mirghafori, *A multi-band approach to automatic speech recognition*, Ph.D. thesis, University of California Berkeley, 1998.

[54] A. Janin, D. Ellis, and N. Morgan, "Multi-stream speech recognition: Ready for prime time?," in *EUROSPEECH*, Budapest, Hungary, 1999.

[55] H. Wang, D. Gelbart, H. Hirsch, and W. Hemmert, "The value of auditory offset adaptation and appropriate acoustic modeling," in *INTERSPEECH*, Brisbane, Australia, 2008.

[56] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky, "Feature extraction using non-linear transformation for robust speech recognition on the Aurora database," in *ICASSP*, Istanbul, Turkey, 2000.

[57] D. Ellis, "Aurora experiments," `http://www.icsi.berkeley.edu/~dpwe/respite/multistream/`.

[58] D. Ellis, "Comparing features statistics: MFCCs, MSGs, etc.," `http://www.icsi.berkeley.edu/~dpwe/respite/multistream/msgmfcc.html`.

[59] H. Hermansky, D. Ellis, and S. Sharma, "Tandem connectionist feature stream extraction for conventional HMM systems," in *ICASSP*, Istanbul, Turkey, 2000.

[60] Q. Zhu, B. Chen, N. Morgan, and A. Stolcke, "On Using MLP Features in LVCSR," in *ICSLP*, Jeju Island, Korea, 2004.

[61] A. Adami, L. Burget, S. Dupont, H. Garudadri, F. Grezl, H. Hermansky, P. Jain, S. Kajarekar, N. Morgan, and S. Sivadas, "Qualcomm-ICSI-OGI features for ASR," in *ICSLP*, Denver, Colorado, 2002.

[62] M. Kleinschmidt and D. Gelbart, "Improving word accuracy with Gabor feature extraction," in *ICSLP*, Denver, Colorado, 2002.

[63] H. Misra, H. Bourlard, and V. Tyagi, "New entropy based combination rules in HMM/ANN multi-stream ASR," in *ICASSP*, Hong Kong, 2003.

[64] H. Misra, *Multi-stream processing for noise robust speech recognition*, Ph.D. thesis, Ecole Polytechnique Fédérale de Lausanne, 2006, available at `http://www.idiap.ch`.

[65] K. Kirchhoff, GA Fink, and G. Sagerer, "Conversational speech recognition using acoustic and articulatory input," in *ICASSP*, Istanbul, Turkey, 2000.

[66] K. Kirchhoff, G.A. Fink, and G. Sagerer, "Combining acoustic and articulatory feature information for robust speech recognition," *Speech Communication*, July 2002.

[67] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 2000.

[68] I. Oh, J. Lee, and B. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 2004.

[69] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, Wiley-Interscience, second edition, 2000.

[70] L. Pols, "Real-time recognition of spoken words," *IEEE Transactions on Computers*, September 1971.

[71] H. Malvar and D. Staelin, "The LOT: transform coding without blocking effects," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, April 1989.

[72] N. Kumar and A.G. Andreou, "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition," *Speech Communication*, December 1998.

[73] S. Balakrishnama, A. Ganapathiraju, and J. Picone, "Linear discriminant analysis for signal processing problems," in *IEEE Southeastcon*, 1999.

[74] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *ICASSP*, Istanbul, Turkey, 2000.

[75] M. Kleinschmidt, "Localized spectro-temporal features for automatic speech recognition," in *EUROSPEECH*, Geneva, Switzerland, 2003.

[76] B. Meyer and B. Kollmeier, "Optimization and evaluation of Gabor feature sets for ASR," in *INTERSPEECH*, Brisbane, Australia, 2008.

[77] Michael Kleinschmidt, *Robust speech recognition based on spectro-temporal processing*, Ph.D. thesis, Carl von Ossietzky-Universitt Oldenburg, 2002, available at `http://www.uni-oldenburg.de/medi`.

[78] M. Kleinschmidt, B. Meyer, and D. Gelbart, "Gabor feature extraction for automatic speech recognition," `http://www.icsi.berkeley.edu/speech/papers/gabor`.

[79] Aldebaro Klautau, "Mining speech: automatic selection of heterogeneous features using boosting," in *ICASSP*, Hong Kong, 2003.

[80] K. Tieu and P. Viola, "Boosting image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina, 2000.

[81] K.Y. Su and C.H. Lee, "Speech recognition using weighted HMM and subspace projection approaches," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, 1994.

[82] J. Novovicová, P. Pudil, and J. Kittler, "Divergence based feature selection for multimodal class densities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 1996.

[83] F. Valente and C. Wellekens, "Variational Bayesian feature selection for Gaussian mixture models," in *ICASSP*, 2004.

[84] R. Kommer and B. Hirsbrunner, "Distributed genetic algorithm to discover a wavelet packet best basis for speech recognition," in *EUROSPEECH*, Geneva, Switzerland, 2003.

[85] B. Raj and RM Stern, "Missing-feature approaches in speech recognition," *IEEE Signal Processing Magazine*, vol. 22, no. 5, 2005.

[86] S. Wu, *Incorporating information from syllable-length time scales into automatic speech recognition*, Ph.D. thesis, University of California Berkeley, 1998, online at `http://www.icsi.berkeley.edu`.

[87] A. Tsymbal, S. Puuronen, and D.W. Patterson, "Ensemble feature selection with the simple Bayesian classification," *Information Fusion*, vol. 4, no. 2, 2003.

[88] L. Kuncheva and L. Jain, "Designing classifier fusion systems by genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, 2000.

[89] C. Guerra-Salcedo and D. Whitley, "Feature selection mechanisms for ensemble creation: A genetic search perspective," in *AAAI-99 and GECCO-99 Workshop on Data Mining with Evolutionary Algorithms*, Orlando, Florida, 1999.

[90] C. Guerra-Salcedo and D. Whitley, "Genetic approach to feature selection for ensemble creation," in *Genetic and Evolutionary Computation Conference (GECCO-99)*, Orlando, Florida, 1999.

[91] P. Sollich and A. Krogh, "Learning with ensembles: How overfitting can be useful," in *Neural Information Processing Systems (NIPS)*, 1996.

[92] W.N. Street and Y.S. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Knowledge Discovery and Data Mining (KDD)*, San Francisco, California, 2001.

[93] J.V. Hansen and A. Krogh, "A general method for combining predictors tested on protein secondary structure prediction," in *Artificial Neural Networks in Medicine and Biology (ANNIMAB-1)*, Goteborg, Sweden, 2000.

[94] P. Granitto, P. Verdes, H. Navone, and H. Ceccatto, "Aggregation algorithms for neural network ensemble construction," in *Proceedings of the VII Brazilian Symposium on Neural Networks (SBRN'02)*, 2002.

[95] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection," in *European Conference on Machine Learning*, Barcelona, Spain, 2000.

[96] G. Zenobi and P. Cunningham, "Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error," in *European Conference on Machine Learning*, Freiburg, Germany, 2001.

[97] Corpora group at Center for Spoken Language Understanding, "ISOLET version 1.3," `http://www.cslu.ogi.edu/corpora`.

[98] Corpora group at Center for Spoken Language Understanding, "Numbers version 1.3," `http://www.cslu.ogi.edu/corpora`.

[99] "SPRACHcore speech recognition tools," `http://www.icsi.berkeley.edu/~dpwe/projects/sprach/sprachcore.html`.

[100] "HTK speech recognition toolkit," `http://htk.eng.cam.ac.uk/`.

[101] H. Hirsch, "FaNT: Filtering and Noise Adding Tool," `http://dnt.kr.hsnr.de/download.html`.

[102] "FaNT and the calculation of the signal-to-noise-ratio (SNR)," `http://dnt.kr.hsnr.de/download/snr_comments.html`.

[103] H. Steeneken and F. Geurtsen, "Description of the RSG-10 noise database," Tech. Rep., TNO Institute for Perception, Holland, 1988.

[104] H. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Automatic Speech Recognition: Challenges for the New Millenium (ASR2000)*, Paris, France, 2000, also published in a shorter version in ICSLP 2000.

[105] J. Reunanen, "Overfitting in making comparisons between variable selection methods," *Journal of Machine Learning Research*, vol. 3, 2003.

[106] K. Schutte, *PhD thesis in progress*, Ph.D. thesis, Massachusetts Institute of Technology, not yet published.

[107] S. Renals and M. Hochberg, "Efficient evaluation of the LVCSR search space using the NOWAY decoder," in *ICASSP*, 1996.

[108] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *ICASSP*, 1989.

[109] J. Mariéthoz and S. Bengio, "A new speech recognition baseline system for Numbers 95 Version 1.3 based on Torch," Tech. Rep. IDIAP-RR 04-16, IDIAP, 2004, available at `http://www.idiap.ch`.

[110] J. Frankel, M. Wester, and S. King, "Articulatory feature recognition using dynamic Bayesian networks," *Computer Speech & Language*, vol. 21, no. 4, 2007.

[111] S. Neto, "The ITU-T Software Tool Library," *International Journal of Speech Technology*, May 1999.

[112] International Telecommunication Union, "ITU Software Tools Library (ITU-T Recommendation G.191)," `http://www.itu.int`.

[113] S. Zhao and N. Morgan, "Multi-stream spectro-temporal features for robust speech recognition," in *INTERSPEECH*, Brisbane, Australia, 2008.

[114] M. Karnjanadecha and S. Zahorian, "Signal modeling for high-performance robust isolated word recognition," *IEEE Transactions on Speech and Audio Processing*, September 2001.

[115] A. Faria and D. Gelbart, "Efficient pitch-based estimation of VTLN warp factors," in *INTERSPEECH*, Lisbon, Portugal, 2005.

[116] V. Tyagi, C. Wellekens, and H. Bourlard, "On variable-scale piecewise stationary spectral analysis of speech signals for ASR," in *INTERSPEECH*, Lisbon, Portugal, 2005.

[117] A. Hagen and H. Bourlard, "Using multiple time scales in the framework of multi-stream speech recognition," in *ICSLP*, Beijing, China, 2000.

[118] Wikipedia, "Speculative execution — Wikipedia, the free encyclopedia," 2008, [Online; accessed September 10, 2008].

[119] J. Loughrey and P. Cunningham, "Using early-stopping to avoid overfitting in wrapper-based feature selection employing stochastic search," Tech. Rep. 2005-37, Trinity College Dublin, 2005, available at `http://www.cs.tcd.ie`.

[120] J. Loughrey and P. Cunningham, "Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets," Tech. Rep. 2005-17, Trinity College Dublin, 2005, available at `http://www.cs.tcd.ie`.

[121] P. Cunningham, "Overfitting and diversity in classification ensembles based on feature selection," Tech. Rep. 2000-07, Trinity College Dublin, 2000, available at `http://www.cs.tcd.ie`.

[122] "Noisy Numbers and Numbers ASR scripts," `http://www.icsi.berkeley.edu/Speech/papers/gelbart-ms/numbers`.

[123] "Noisy ISOLET and ISOLET ASR scripts," http://www.icsi.berkeley.edu/Speech/papers/eurospeech05-onset/isolet.

[124] ISCA Student Advisory Commitee, "List of free software for speech processing," http://isca-students.org/freeware.

[125] D. Macho, L. Mauuary, B. Noé, Y. Cheng, D. Ealey, D. Jouvet, H. Kelleher, D. Pearce, and F. Saadoun, "Evaluation of a noise-robust DSR front-end on AURORA databases," in *ICSLP*, Denver, Colorado, 2002.

[126] "Aurora noise robust front ends," http://www.icsi.berkeley.edu/speech/papers/qio.

[127] M.N. Kapp, R. Sabourin, and P. Maupin, "An empirical study on diversity measures and margin theory for ensembles of classifiers," in *International Conference on Information Fusion (Fusion 2007)*, Quebec City, Canada, 2007.

[128] A. Tsymbal, "Personal communication," 2007.

[129] F. Sha and L. Saul, "Large margin hidden markov models for automatic speech recognition," in *Neural Information Processing Systems (NIPS)*, 2007.

[130] D. Yu, L. Deng, X. He, and A. Acero, "Large-margin minimum classification error training for large-scale speech recognition tasks," in *ICASSP*, Honolulu, Hawaii, 2007.

[131] H. Jiang and X. Li, "Incorporating training errors for large margin HMMs under semi-definite programming framework," in *ICASSP*, Honolulu, Hawaii, 2007.

[132] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Sequential genetic search for ensemble feature selection," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, Edinburgh, Scotland, 2005.

[133] J. Bilmes, K. Asanovic, C. Chin, and J. Demmel, "Using PHIPAC to speed error back-propagation learning," in *ICASSP*, Munich, Germany, 1997.

Appendix A

Bunch size and MLP training

One of the Quicknet MLP training options is bunch size, as explained in [133]: "two modes of back-propagation learning [are traditionally defined], on-line mode where only one training pattern is used at a time to update the weight matrices, and batch mode where all training patterns are used simultaneously to update the weight matrices. An alternate strategy, which we call bunch-mode..., uses more than one training pattern simultaneously to update the weight matrices." The bunch size is the number of training patterns used to update the weight matrices in bunch mode.

A bunch size of 1 is equivalent to traditional online training. Higher bunch sizes can result in faster Quicknet training speeds, and the default in Quicknet 3.20 is 16. However, to obtain the maximum training speed for multi-threaded trainings, it is necessary to increase bunch size beyond 16.

When working with the Numbers corpus, we performed trainings using four threads and a bunch size of 2048. This bunch size had been used previously for another project which used eight-threaded trainings with a much larger speech corpus. A bunch size of 2048 was judged to be an appropriate speed/quality tradeoff for that project, but it turned out to be a suboptimal choice for our Numbers corpus. It often resulted in considerable degradation in quality of the trained nets. Table A.1 shows the effect of the bunch size choice. We could have achieved a comparable training speed using a smaller bunch size, but we did not realize that using 2048 would have this effect.

We did not notice this effect until we already had completed thousands of CPU-hours worth of Numbers experiments. Thus, we chose not to re-run our experiments with a different bunch size. Instead, Numbers results presented in this thesis (with the sole exception of the default bunch size results in Table A.1) were consistently calculated using four-threaded training with a bunch size of 2048. In other words, while our training choices were not optimal, they were consistent.

The bunch size issue affected only our results for the Numbers corpus. In our experiments on the ISOLET corpus, we did not use multi-threaded training, and so we left the bunch size at the default.

Experiment	Clean train and test		Noisy train and test	
	Default bunch size	Bunch size 2048	Default bunch size	Bunch size 2048
(a) MFCC (3600 hidden units)	5.7	6.5	20.4	21.4
(b) PLP (3600 HUs)	4.8	5.0	17.2	17.5
(c) MSG (4772 HUs)	6.3	7.3	14.8	16.3
(d) MFCC, PLP, MSG	4.4	4.9	15.5	15.7
(e) MFCC (1200 HUs)	6.3	6.4	21.5	21.2
(f) PLP (1200 HUs)	5.0	5.1	17.8	17.1
(g) MSG (1590 HUs)	5.8	7.0	14.9	15.9

Table A.1. This table shows the effect of bunch size for the Numbers corpus. The results are word error rates (WERs) on the evaluation set. Rows (a), (b) and (c) are for single-stream MFCC, PLP and MSG respectively, with the number of hidden units (HUs) chosen as explained in Section 6.2. Row (d) is for a three-MLP MFCC, PLP and MSG system with the same total number of acoustic model parameters. It uses 1200 HUs for the MFCC and PLP MLPs and 1590 for the MSG MLP. Rows (e), (f) and (g) show the single-stream performance of the MLPs that are used in the three-stream system in row (d).