Sparse Representation Features for Speech Recognition

Tara N. Sainath, Bhuvana Ramabhadran, David Nahamoo, Dimitri Kanevsky, Abhinav Sethy

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A
{tsainath, bhuvana, nahamoo, kanevsky, asethy}@ibm.us.com

Abstract

In this paper, we explore the use of exemplar-based sparse representations (SRs) to map test features into the linear span of training examples. We show that the frame classification accuracy with these new features is 1.3% higher than a Gaussian Mixture Model (GMM), showing that not only do SRs move test features closer to training, but also move the features closer to the correct class. Given these new SR features, we train up a Hidden Markov Model (HMM) on these features and perform recognition. On the TIMIT corpus, we show that applying the SR features on top of our best discriminatively trained system allows for a **0.7**% absolute reduction in phonetic error rate (PER), from 19.9% to 19.2%. In fact, after applying model adaptation we reduce the PER to **19.0**%, the best results on TIMIT to date. Furthermore, on a large vocabulary 50 hour broadcast news task, we achieve a reduction in word error rate (WER) of **0.3**% absolute, demonstrating the benefit of this method for large vocabulary speech recognition.

1. Introduction

Gaussian Mixture Models (GMMs) continue to be extremely popular for recognition-type problems in speech. While GMMs allow for fast model training and scoring, training samples are pooled together for parameter estimation, resulting in a loss of information that exists within individual training samples. At the other extreme, exemplar based techniques, including k-nearest neighbors (kNNs), support vector machines (SVMs) and sparse representation (SR) methods, utilize information about actual training examples. While these exemplar-based methods have been shown to offer improvements in accuracy over GMMs for classification tasks [5], applying these techniques for recognition has sofar been met with limited success [1], with gains mainly coming in small-vocabulary tasks [2].

The limited success of exemplar-based methods for recognition tasks can be attributed to the following reasons. First, characterizing a test sample by searching over a large amount of training data (typically greater than 50 hours for large vocabulary) is more computationally expensive relative to evaluating a set of Gaussian mixtures. Second, in both classification and recognition, the goal is to determine classes that best represent the test samples. In classification, the segments associated with each class are known ahead of time, and thus decision scores can directly be calculated for each segment using exemplar-based techniques. In recognition, class boundaries are not known beforehand, and thus must be determined via a dynamic programming approach (e.g., HMMs). This requires estimating class probabilities that can be compared across frames, something which exemplar-based methods cannot easily do [1].

In this paper, we explore the use of a sparse representation exemplar-based technique [5] to create a new set of features while utilizing the benefits of HMMs to efficiently compare scores across frames. This is in contrast to previous exemplar-based methods which try to utilize the decision scores from the exemplar-based classifiers themselves to generate probabilities ([1], [2]). In our SR approach, given a test vector y and a set of exemplars h_i from the training set, which we put into a dictionary $H = [h_1; h_2 \ldots; h_n]$, we represent y as a linear combination of training examples by solving $y = H\beta$ subject to a spareness constraint on β. The feature $H\beta$ can be thought of as mapping test sample y back into the linear span of training examples in H. We will show that the frame classification accuracy is higher for the SR method[1] compared to a GMM, showing that not only does the $H\beta$ representation move test features closer to training, but also moves these features closer to the correct class. Given these new set of $H\beta$ features, we train up an HMM on these features and perform recognition.

We investigate these new SR features for both small and large vocabulary tasks. On the TIMIT corpus [4], we show that applying the SR features on top of our best discriminatively trained system allows for a 0.7% absolute reduction in phonetic error rate (PER), from 19.9% to 19.2%. In fact, after applying MLLR we are able to reduce the PER to 19.0%, the best results on TIMIT to date. Furthermore, on a large vocabulary 50 hour broadcast news task [3], we achieve a reduction in word error rate (WER) of 0.3% absolute with the SR features, demonstrating the benefit of this method for large-scale tasks.

The rest of this paper is organized as follows. Section 2 reviews the construction of spare representation $H\beta$ features, while Section 3 discusses different choices of the exemplar dictionary H. Section 4 presents the experiments performed, followed by a discussion of the results in Section 5. Finally, Section 6 concludes the paper and discusses future work.

2. Sparse Representation Features

In this section, we discuss the formulation of the SR features.

2.1. Sparse Representation Formulation

To create a set of SR features, first, let us consider taking all training examples n_i from class i and concatenate them into a matrix H_i as columns, i other words $H_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}] \in \Re^{m \times n_i}$, where $x \in \Re^m$ represents a feature vector from the training set of class i with dimension m. Given sufficient training examples from class i, [7] shows that a test sample $y \in \Re^m$ from the same class can be represented as a linear combination of the entries in H_i weighted by β, that is:

$$y = \beta_{i,1}x_{i,1} + \beta_{i,2}x_{i,2} + \ldots + \beta_{i,n_i}x_{i,n_i} \qquad (1)$$

However, since the class membership of y is unknown, we define a matrix H to include training examples from k different classes in the training set, in other words the columns of H are

[1]Using SRs to compute accuracy is described in [5].

defined as $H = [H_1, H_2, \dots, H_k] = [x_{1,1}, x_{1,2}, \dots, x_{k,n_k}] \in \Re^{m \times N}$. Here m is the dimension of each feature vector x and N is the total number of all training examples from all classes. H can be thought of as an over-complete dictionary where $m << N$. We can then write test vector y as a linear combination of all training examples, in other words $y = H\beta$. Ideally the optimal β should be sparse, and only be non-zero for the elements in H will belong to the same class as y. Thus ideally y will assign itself to lie in the linear span of examples from the training set of the true class it belongs to.

In this work, we solve the problem $y = H\beta$ subject to a sparseness constraint on β. As [7] discusses, the sparseness constraint on β acts as a regularization term to prevent over-fitting, and often allows for better classification performance than without sparseness. Various SR methods can be used to solve for β. In this paper, we solve for β using the Approximate Bayesian Compressive Sensing (ABCS) method [5], which imposes a combination of an l_1 and l_2 regularization on β.

A speech signal is defined by a series of feature vectors, $Y = \{y^1, y^2 \dots y^n\}$, for example Mel-Scale Frequency Cepstral Coefficients (MFCCs). For every test sample $y^t \in Y$, we choose an appropriate H^t and then solve $y^t = H^t\beta^t$ to compute a β^t via ABCS. Then given this β^t, a corresponding $H^t\beta^t$ vector is formed. Thus a series of $H\beta$ vectors is created at each frame as $\{H^1\beta^1, H^2\beta^2 \dots H^n\beta^n\}$. The sparse representation features are created for both training and test. An HMM is then trained given this new set of features and recognition is performed in this new feature space.

2.2. Measure of Quality

Now that we have described our method to solve for β given y and H, we now discuss how to measure the quality of how well the test vector y is mapped back into the linear span of training examples in H. We can measure how well y assigns itself to different classes in H by looking at the residual error between y and the $H\beta$ entries corresponding to a specific class [7]. Ideally, all nonzero entries of β should correspond to the entries in H with the same class as y and the residual error will be smallest within this class. More specifically, let us define a selector $\delta_i(\beta) \in \Re^N$ as a vector whose entries are non-zero except for entries in β corresponding to class i. We then compute the residual error for class i as $\parallel y - H\delta_i(\beta) \parallel_2$. The best class for y will be the class with the smallest residual error. Mathematically, the best class i^* is defined as

$$i^* = \min_i \parallel y - H\delta_i(\beta) \parallel_2 \qquad (2)$$

3. Choices of Dictionary H

Success on the sparse representation features depends heavily on a good choice of H. Pooling together all training data from all classes into H will make the columns of H large (typically millions of frames), and will make solving for β intractable. Therefore, in this section we discussion various methodologies to select H from a large sample set. Recall that H is selected for each frame y, and then β is found using ABCS, in order to create an $H\beta$ feature for each frame.

3.1. Seeding H from Nearest Neighbors

For each y, we find a neighborhood of closest points to y in the training set. These k neighbors become the entries of H. We refer the reader to [5] for a discussion on choosing the number of k neighbors for SRs. A set of $H\beta$ features is created for both training and test, but H is always seeded with data from training data. To avoid overtraining of $H\beta$ features on the training set, we require that only when creating $H\beta$ features on training, samples be selected from training that are of a different speaker than the speaker corresponding to frame y. While this kNN approach is computationally feasible on small-vocabulary tasks, using a kNN for large vocabulary tasks can be computationally expensive. To address this, we discuss other choices for seeding H below, tailored to large vocabulary applications.

3.2. Using a Trigram Language Model

Ideally only a small subset of Gaussians are typically evaluated at a given frame, and thus training data belonging to this small subset can be used to seed H. To determine these Gaussians at each frame, we decode the data using a trigram language model (LM), and find the best aligned Gaussian at each frame. For each Gaussian, we compute the 4 other closest Gaussians to this Gaussian. Here closeness is defined by finding Gaussian pairs which have the smallest Euclidean distance between their means. After we find the top 5 Gaussians at a specific frame, we seed H with the training data aligning to these top 5 Gaussians. Since this still typically amounts to thousands of training samples in H, we must sample this further. Our method for sampling is discussed in Section 3.7. We also compare seeding H using the top 10 Gaussians rather than top 5.

3.3. Using a Unigram Language Model

One problem with using a trigram LM is that this decode is actually the baseline system we are trying to improve upon. Therefore, seeding H with frames related to the top aligned Gaussian is essentially projecting y back down to the same Gaussian which initially identified it. Thus to increase variability between the Gaussians used to seed H and the best aligned Gaussian from the trigram LM decode, we explore using a unigram LM to find the best aligned Gaussian at each frame. Again, given the best aligned Gaussian, the 4 closest Gaussians to this are found and data from these 5 Gaussians is used to seed H.

3.4. Using no Language Model Information

To further weaken the effect of the LM, we explore seeding H using only acoustic information. Namely, at each frame we find the top 5 scoring Gaussians. H is seeded with training data aligning to these Gaussians.

3.5. Enforcing Unique Phonemes

Another problem with seeding H by finding the 5 closest Gaussians relative to the best aligned Gaussian is that all of these Gaussians could come from the same phoneme (i.e. phoneme "AA"). Therefore, we explore finding the 5 closest Gaussians relative to the best aligned such that the phoneme identities of these Gaussians are unique (i.e. "AA", "AE", "AW", etc.). H is then seeded by from frames aligning to these 5 Gaussians.

3.6. Using Gaussian Means

The above approaches of seeding H use actual examples from the training set, which is computationally expensive. To address this, we investigate seeding H from Gaussian means. Namely, at each frame we use a trigram LM to find the best aligned Gaussian. Then we find the 499 closest Gaussians to this top Gaussian, and use the means from these 500 Gaussians to seed H.

3.7. Choice of Sampling

In this section, we explore two different approaches to sampling a subset of this data to seed H.

3.7.1. Random Sampling

For each gausssian we want to select training data from, we explore randomly sampling N training examples from the total set of training frames that aligned to this Gaussian. This process is repeated for each of the closest 5 Gaussians. We reduce the size of N as the "closeness" decreases. For example, for the closest 5 Gaussians, the number of data points N chosen from each Gaussian is 200, 100, 100, 50 and 50 respectively.

3.7.2. Sampling Based on Cosine Similarity

While random sampling offers a relatively quick approach to select a subset of training examples, it does not guarantee that we select "good examples" from this Gaussian which actually are close to frame y. Alternatively, we explore splitting training points aligning to a Gaussian as being 1σ, 2σ, etc. away from the mean of the Gaussian. Here σ is chosen to be the total number of training points aligned to this Gaussian, divided by number of samples N we want to sample from this Gaussian. Then within each σ set, we find the training point which has the closest cosine similarity to the test point y. This is repeated for all 1σ, 2σ, etc. values. Again the number of samples taken from each Gaussian reduces as "closeness" decreases.

4. Experiments

The small vocabulary recognition experiments in this paper are conducted on the TIMIT phonetic corpus [4]. Similar to [6], acoustic models are trained on the training set, and results are reported on the core test set. The initial acoustic features are 13-dimensional MFCC features. The large vocabulary experiments are conducted on an English broadcast news transcription task [3]. The acoustic model is trained on 50 hours of data from the 1996 and 1997 English Broadcast News Speech Corpora. Results are reported on 3 hours of the EARS Dev-04f set. The initial acoustic features are 19-dimensional PLP features.

Both small and large vocabulary experiments utilize the following recipe for training acoustic models [6]. First, a set of CI HMMs are trained, either using information from the phonetic transcription (TIMIT) or from flat-start (broadcast news). The CI models are then used to bootstrap the training of a set of CD triphone models. In this step, at each frame, a series of consecutive frames surrounding this frame are joined together and a Linear Discriminative Analysis (LDA) transform is applied to project the feature vector down to 40 dimensions. Next, vocal tract length normalization (VTLN) and feature space Maximum Likelihood Linear Regression (fMLLR) are used to map the features into a canonical speaker space. Then, a set of discriminatively trained features and models are created using the boosted Maximum Mutual Information (BMMI) criterion. Finally, the set of models is adapted using MLLR.

We create a set of $H\beta$ features from a set of fBMMI features. We choose this level as these features offer the highest frame accuracy relative to LDA, VTLN, or fMLLR features, allowing us to further improve on the accuracy with with the $H\beta$ features. A set of $H\beta$ features are created at each frame from the fBMMI features for both training and test. A new ML HMM is trained up from these new features and used for both training and test. Since $H\beta$ features create a linear combination of the discriminatively trained fBMMI features, we argue that some discrimination can be lost. Therefore, we explore applying another fBMMI transformation to the $H\beta$ features before applying model space discriminative training and MLLR.

5. Results

In this section we present results using $H\beta$ features on both small and large vocabulary tasks.

5.1. Sparsity Analysis

We first analyze the β coefficients obtained by solving $y = H\beta$ using ABCS [5]. For two randomly selected frames y, Figure 1 shows the β coefficients corresponding to 200 entries in H for TIMIT and 500 entries for Broadcast News. Notice that for both datasets, the β entries are quite sparse, illustrating that only a few samples in H are used to characterize y. As [7] discusses, this sparsity can be thought of as a form of discrimination, as certain examples are selected as "good" in H while jointly assigning zero weights "bad" examples in H. We have seen advantages of the SR approach for classification, even on top of discriminatively trained y features, compared to a GMM [5]. We will also re-confirm this behavior in Section 5.2. The extra benefit of SRs on top of discriminatively trained fBMMI features, coupled with an exemplar-based nature of SRs, motivates us to further explore its behavior for recognition tasks.

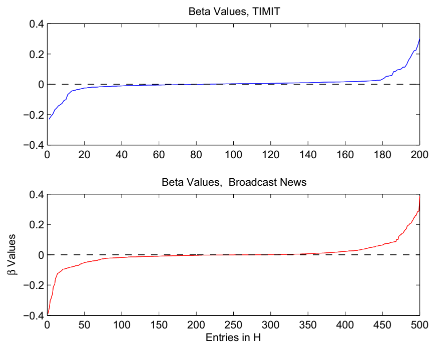

Figure 1: β Coefficients on TIMIT and Broadcast News

5.2. TIMIT Results

5.2.1. Frame Accuracy

The success of $H\beta$ first relies on the fact that the β vectors give large support to correct classes and small support to incorrect classes (as demonstrated by Figure 1) when computing $y = H\beta$ at each frame. Thus, the classification accuracy per frame, computed using Equation 2, should ideally be high. Table 1 shows the frame accuracy for the GMM and SR methods.

Classifier	Frame Accuracy
GMM	70.4
Sparse Representations	**71.7**

Table 1: Frame Accuracy on TIMIT Testcore Set

Notice that the SR technique offers significant improvements over the GMM method, again confirming the benefit of exemplar-based classifiers.

5.2.2. *Error Rate for $H\beta$ features*

Table 2 shows the recognition performance of $H\beta$ features on TIMIT. Due to the small vocabulary nature of TIMIT, we only explore seeding H from nearest neighbors. Notice that creating a set of $H\beta$ features in the fBMMI space offers a 0.7% absolute improvement in PER. Given the small vocabulary nature of TIMIT, no gain was found applying another fBMMI transform to the baseline or $H\beta$ features. After applying BMMI and MLLR to both feature sets, the $H\beta$ features offer a 0.5% improvement in PER over the baseline system. This shows that using exemplar-based SRs to produce $H\beta$ features not only moves test features closer to training, but also moves the feature vectors closer to the correct class, resulting in a decrease in PER.

Baseline System	PER	$H\beta$ System	PER
fBMMI	19.9	$H\beta$	**19.2**
+BMMI+MLLR	19.5	+BMMI+MLLR	**19.0**

Table 2: WER on TIMIT

5.3. Broadcast News Results

5.3.1. *Selection of H*

Table 3 shows the WER for the $H\beta$ features for different H choices discussed in Section 3. Note that the baseline fMMI system as a WER of 21.1%. The following can be observed:

- There is little difference in WER when sampling is done randomly or using cosine similarity. For speed efficiencies, we use random sampling for H selection methods.
- There is little difference between using 5 or 10 Gaussians.
- Seeding H using nearest neighbors is worse than using the trigram LM. On broadcast news, we find that a kNN has lower frame-accuracy than a GMM, a result similarly observed in the literature for large vocabulary corpora [1]. This lower frame accuracy translates into a higher WER when H is seeded with nearest neighbors.
- Seeding H from unique Gaussians provides too much variability of phoneme classes into the $H\beta$ feature, also leading to a higher WER.
- Using a unigram LM to reduce the link between the Gaussians used to seed H and the best aligned Gaussian from the trigram LM decode offers a slight improvement in WER over the trigram LM.
- Utilizing no LM information results in a very high WER.
- Using Gaussian means to seed H reduces the computation to create $H\beta$ without a large increase in WER.

H Selection Method	WER
Trigram LM, Random Sampling, Top 5 Gaussians	21.2
Trigram LM, Cosine Similarity Sampling, Top 5 Gaussians	21.3
Trigram LM, Top 10 Gaussians	21.3
Nearest Neighbor, 500	21.4
Trigram LM, 5 Unique Gaussians	21.6
Unigram LM, Top 5 Gaussians	**21.1**
No LM Information, Top 5 Gaussians	22.7
Gaussian Means, Top 500 Gaussians	21.4

Table 3: WER of $H\beta$ features for different H

For further recognition experiments on Broadcast news, we explore using $H\beta$ features created from a unigram LM decode.

5.3.2. *WER for $H\beta$ Features*

Table 4 shows the performance of $H\beta$ features on the Broadcast News task. Creating a set of $H\beta$ features at the fBMMI space offers a WER of 21.1% which is comparable to the baseline system. However, after applying an fBMMI transform to the $H\beta$ features we achieve a WER of 20.2%, a 0.2% absolute improvement when another fBMMI transform is applied to the original fBMMI features. Finally, after applying BMMI and MLLR to both feature sets, the $H\beta$ features offer a WER of 18.7%, a 0.3% absolute improvement in WER over the baseline system. This demonstrates again that using information about actual training examples to produce a set of features which are mapped closer to training and have a higher frame accuracy than GMMs improves accuracy for large vocabulary as well.

Baseline System	WER	$H\beta$ System	WER
fBMMI	21.1	$H\beta$	21.1
+fBMMI	20.4	+fBMMI	**20.2**
+BMMI+MLLR	19.0	+BMMI+MLLR	**18.7**

Table 4: WER on Broadcast News

6. Conclusions and Future Work

In this paper, we explored mapping feature vectors from test into the linear span of actual training examples through sparse representations. We demonstrated that not only do these sparse representations move test features closer to training, but also move the features closer to the correct class. On the TIMIT corpus, we show that applying the sparse representation features on top of our best discriminatively trained system achieves a PER of **19.0**%, the best results on TIMIT to date. Furthermore, on a large vocabulary 50 hour broadcast news task, we are able to achieve a reduction in word error rate (WER) of **0.3**% absolute.

7. Acknowledgements

The authors would like to thank Hagen Soltau, George Saon, Brian Kingsbury and Stanley Chen for their contributions towards the IBM toolkit and recognizer utilized in this paper.

8. References

[1] T. Deselaers, G. Heigold, and H. Ney, "Speech Recognition With State-based Nearest Neighbour Classifiers," in *Proc. Interspeech*, 2007.

[2] J. F. Gemmeke and T. Virtanen, "Noise Robust Exemplar-Based Connected Digit Recognition," in *Proc. ICASSP*, 2010.

[3] B. Kingsbury, "Lattice-Based Optimization of Sequence Classification Criteria for Neural-Network Acoustic Modeling," in *Proc. ICASSP*, 2009.

[4] L. Lamel, R. Kassel, and S. Seneff, "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," in *Proc. of the DARPA Speech Recognition Workshop*, 1986.

[5] T. N. Sainath, A. Carmi, D. Kanevsky, and B. Ramabhadran, "Bayesian Compressive Sensing for Phonetic Classification," in *Proc. ICASSP*, 2010.

[6] T. N. Sainath, B. Ramabhadran, and M. Picheny, "An Exploration of Large Vocabulary Tools for Small Vocabulary Phonetic Recognition," in *Proc. ASRU*, 2009.

[7] J. Wright, A. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 210–227, 2009.