RC24983 (W1004-111) April 26, 2010
Electrical Engineering

IBM Research Report

Sparse Representation Phone Identification Features for Speech Recognition

Tara N. Sainath, David Nahamoo, Bhuvana Ramabhadran, Dimitri Kanevsky
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Sparse Representation Phone Identification Features for Speech Recognition

Tara N. Sainath TNSAINAT@US.IBM.COM
1101 Kitachawan Road, Yorktown Heights, NY, 10598, USA

David Nahamoo NAHAMOO@US.IBM.COM
1101 Kitachawan Road, Yorktown Heights, NY, 10598, USA

Bhuvana Ramabhadran BHUVANA@US.IBM.COM
1101 Kitachawan Road, Yorktown Heights, NY, 10598, USA

Dimitri Kanevsky KANEVSKY@US.IBM.COM
1101 Kitachawan Road, Yorktown Heights, NY, 10598, USA

Abstract

Sparse representation techniques, such as Support Vector Machines (SVMs), k-nearest neighbor (kNN) and Bayesian Compressive Sensing (BCS), can be used to characterize a test sample from a few support training samples in a dictionary set. In this paper, we introduce a semi-gaussian constraint into the BCS formulation, which allows support parameters to be estimated using a closed-form iterative solution. We show that using this approach for phonetic classification allows for a higher accuracy than other non-parametric techniques. These phones are the basic units of speech to be recognized. Motivated by this result, we create a new dictionary which is a function of the phonetic labels of the original dictionary. The support vectors now select relevant samples from this new dictionary to create a new representation of the test sample, where the test sample is better linked to the actual units to be recognized. We present results using these new features in a Hidden Markov Model (HMM) framework for speech recognition. We find that these features allow for a Phonetic Error Rate (PER) of **23.9%** on the TIMIT phonetic recognition task, the best result on TIMIT to date when HMM parameters are trained using the maximum likelihood principle.

1. Introduction

In machine learning theory, problems can be cast in a multi-class regression or classification framework. While the former is the task of decomposing signals into a common basis, the latter is the task of discriminating between different classes. The regression problem reduces to identifying the common sparsity pattern of relevant variables selected from a relatively high-dimensional space. In statistical signal processing, whenever the optimal representation is sufficiently sparse, it can be efficiently computed by convex optimization (Donoho, 2006).

The original goal of these sparse representation research efforts was not for classification, but rather the efficient representation and compression of signals (Candes et al., 2006), and their performance was measured in terms of sparsity of the representation. The sparsest representation selects the subset which most compactly expresses the input signal from other less compact representations. In this paper, we exploit the sparse representation of training samples, which we will refer to as a dictionary, to perform multi-class classification of test samples through an optimal selection of sparse examples from this dictionary.

Non-parametric methods, including kNNs and SVMs, also utilize details of the training examples when solving classification-type problems. However, parametric modeling techniques, such as Gaussian Mixture Models (GMMs) continue to be extremely popular for recognition-type problems in speech recognition. 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. And yet, while non-parametric methods have been shown to offer improvements in classification accuracy over parametric methods, gains on classification tasks using these non-parametric methods do not necessarily translate into large gains for direct recognition applications (Deselaers et al., 2007). 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 non-parametric techniques. In recognition, class boundaries are not known beforehand, and thus must be determined via a dynamic programming approach (e.g., Hidden Markov Models (HMMs)). In this paper, we look at creating a new set of features using a non-parametric sparse representation, but still explore these features for HMMs.

Sparse representations (SRs), including methods such as compressive sensing (CS) (Candes et al., 2006), have become a popular technique in recent years for efficient representation and compression of signals. Recently, SRs have also been used as a non-parametric classifier for classification tasks (Sainath et al., 2010), (Wright et al., 2009). Mathematically speaking, in the SR formulation for classification, a matrix H is constructed consisting of possible examples of the signal, that is $H = [h_1; h_2 \ldots; h_n]$. For example, in speaker classification, each $h_i \in H$ could represent features from different speakers in the training set. Given a test speaker's feature vector y, the goal of SRs is to solve the following problem in Equation 1 for β, where $\parallel \beta \parallel_1 < \epsilon$ imposes an l_1 regularization on the vector β, thus selecting a small number of examples from H.

$$y = H\beta \;\; \texttt{s.t.} \quad \parallel \beta \parallel_1 < \epsilon \tag{1}$$

The above problem can be solved using a variety of techniques, include Lasso, Dantzig (Candes et al., 2006) or BCS (Ji et al., 2008) methods. In this paper, we introduce an Approximate Bayesian Compressive Sensing (ABCS) (Carmi et al., 2009) formulation which allows us to derive an iterative closed-form solution for estimating sparseness parameters β. While we do not assume that our dictionary H obeys a restricted isometric property (RIP), as in traditional compressive sensing, we term this algorithm ABCS as the initial use of it was for image classification (Carmi et al., 2009) where H did obey an RIP property. Each element of β in some sense characterizes how well the corresponding $h_i \in H$ represents speaker y, with larger elements in β representing better quality speakers h_i. We can develop a speaker classification decision for y, by choosing the speaker from H that has the maximum size of β elements (Wright et al., 2009).

Notice that using SR for classification utilizes details about the training sample dictionary to characterize a test sample using a few number of support training examples, similar to kNNs and SVMs. However, the kNN and SVM techniques do not adapt the number and type of supports to each test example. SVMs select a sparse subset of relevant training examples, known as support vectors, and use these supports to characterize "all" examples in the test set. kNN methods characterizes any test point by selecting a small "fixed" number of k points from the training set which are closest to the test vector, and voting on the class that has the highest occurrence from these k samples. The SR method presented in this paper is a superset of these popular classification schemes since the test sample is not only represented by training samples from its class but also across multiple classes, while changing the number of support training samples for each test sample. In fact, (Sainath et al., 2010) explored SR for phonetic classification on the TIMIT task (Lamel et al., 1986), and found that the SR technique outperformed the GMM, SVM and kNN methods, and offered one of the best reported results in the literature to date.

In this paper, we first solve Equation 1 to find the best β, where y is a feature vector from the test set, and H is a collection of feature vectors from training. Again, we show SRs allow for a higher classification accuracy compared to other techniques, implying that β is selecting a few appropriate samples from the overcomplete dictionary H to represent y. After β is computed, we change the dictionary to be a function of the actual phonetic labels in H. β now selects relevant examples from this new dictionary. We now represent y using this new dictionary which we refer to as Sparse Representation Phone Identification Features (S_{pif}). The new features are better linked to the actual phonetic units to be recognized. Since SR selects appropriate β values and provides a higher classification accuracy than a GMM, scoring the S_{pif} vectors, derived from the same β values and now linked to phonetic units, with a parametric model should produce a higher classification accuracy than the original y features. In turn, higher classification accuracy may correspond to higher recognition accuracy, when used in combination with parametric HMM models. Thus, we take advantage of the higher accuracy offered by the SR method to derive a new set of features, while still exploiting the use of the parametric HMM for speech recognition.

In summary, the key contributions of this paper are:

- A novel ABCS sensing formulation to solve problems in speech recognition
- A novel set of S_{pif} vectors that can be used in tandem with parametric GMM or HMM classifiers

2. ABCS

State-of-the-art methods for sparse signal recovery commonly utilize convex relaxation methods, including the Lasso and Dantzig selector techniques. However, these techniques require considerable effort to tune the sparseness constraint. Moreover, these methods only provide a point estimate for β, and can thus be considered to be a sub-optimal solution. Alternatively, an optimization method known as BCS has recently been introduced, which uses a probabilistic framework to estimate the spareness parameters. This technique limits the effort required to tune the sparseness constraint, and also provides complete statistics for the estimate of β. The approach in (Ji et al., 2008) uses a Laplacian prior, which is not conjugate to the Gaussian likelihood and is thus approximated with a relevance vector machine (RVM). In this paper, we address drawbacks of the Lasso and Dantzig methods. Following a probabilistic formulation similar to BCS, we introduce a semi-gaussian prior into this framework which allows for the derivation of a iterative closed-form solution for the sparseness parameters. We call this new technique Approximate Bayesian Compressive Sensing (ABCS).

2.1. ABCS Derivation

Before we present the ABCS derivation, we provide an intuitive explanation about the characteristics of $\parallel \beta \parallel_1^2 = (\sum_i |\beta_i|)^2$ and $\parallel \beta \parallel_1 = (\sum_i |\beta_i|)$. We can denote the semi-gaussian density function as proportional to $p_{semi-gauss} \propto exp(- \parallel \beta \parallel_1^2)$ and the laplacian density function proportional to $p_{laplace} \propto exp(- \parallel \beta \parallel_1)$. When $\parallel \beta \parallel_1 < 1$, it is straightforward to see that $p_{semi-gauss} > p_{laplace}$. When $\parallel \beta \parallel_1 = 1$, the density functions are the same, and when $\parallel \beta \parallel_1 > 1$ then $p_{semi-gauss} < p_{laplace}$. Therefore the semi-gaussian density is more concentrated than the laplacian density in the convex area inside $\parallel \beta \parallel_1 < 1$. As shown in (Chartrand, 2007), given the sparseness constraint $\parallel \beta \parallel_q$, as the fractional norm q goes to 0, the density becomes concentrated at the coordinate axes and the problem of solving for β becomes a non-convex optimization problem where the reconstructed signal has the least mean-squared-error (MSE). As stated above, the semi-gaussian density has more concentration inside the region $\parallel \beta \parallel_1 < 1$. Intuitively, we expect the solution using the semi-gaussian prior to behave closer to the non-convex solution.

Our CS formulation using the semi-gaussian constraint, similar to Equation 1, is given below:

$$y = H\beta \ \texttt{s.t.} \quad \parallel \beta \parallel_1^2 < \epsilon \tag{2}$$

In Equation 2, y is a sample of data from the test set such that $y \in \Re^m$ where m is the dimension of feature vector y. H is a matrix of training examples and $H \in \Re^{m \times n}$, where $m << n$.

We assume that y satisfies a linear model as $y = H\beta + \zeta$ where $\zeta \sim N(0, R)$. This allows us to represent $p(y|\beta)$ as a Gaussian:

$$p(y|\beta) \propto exp(-1/2(y - H\beta)^T R^{-1}(y - H\beta)) \tag{3}$$

Assuming β is a random parameter with some prior $p(\beta)$ we can obtain the maximum a posteriori (MAP) estimate for β given y as follows: $\beta^* = \arg\max_\beta p(\beta|y) = \max_\beta p(y|\beta)p(\beta)$.

In the ABCS formulation, we assume that $p(\beta)$ is actually the product of two density functions, namely a gaussian density function $p_G(\beta)$, representing prior belief on β, and a semi-gaussian density function $p_{SG}(\beta)$, which represents the sparseness constraint $\parallel \beta \parallel_1^2 < \epsilon$. Therefore, the total objective function J, which we would like to maximize to find β, is given as follows:

$$\beta^* = \arg\max_\beta J = \arg\max_\beta p(y|\beta)p_G(\beta)p_{SG}(\beta) \tag{4}$$

We assume that $p_G(\beta)$ is represented as $p_G(\beta) = N(\beta|\beta_0, P_0)$. Here β_0 and P_0 are initialized statistical moments. The semi-Gaussian prior, $p_{SG}(\beta)$, as given by Equation 5, imposes sparseness on β with σ^2 controlling the degree of sparseness.

$$p_{SG}(\beta) = exp\left(-\frac{||\beta||_1^2}{2\sigma^2}\right) \tag{5}$$

Let us define β^i to be the i^{th} entry of the vector $\beta = [\beta^0, \beta^1, \ldots, \beta^n]$. We define a vector V with entries set as $V^i(\beta^i) = \text{sign}(\beta^i)$, for $i = 1, \ldots, n$. Here $V^i(\beta^i) = +1$ for $\beta^i > 0$, $V^i(\beta^i) = -1$ for $\beta^i < 0$, and $V^i(\beta^i) = 0$ for $\beta^i = 0$. Using this definition for V, we obtain:

$$\parallel \beta \parallel_1^2 = (\sum_i (|\beta^i|))^2 = (\sum_i (V^i(\beta^i)\beta^i))^2 = (V\beta)^2 \tag{6}$$

Substituting this expression for $\parallel \beta \parallel_1^2$ given in Equation 6 and assuming $y = 0$, we can rewrite Equation 5 as Equation 7 given below, which provides a Gaussian-like representation.

$$p_{SG}(\beta) = p(y=0|\beta) = exp\left(\frac{-(0-V\beta)^2}{2\sigma^2}\right) \tag{7}$$

Given the dependency of V on β, in order to solve Equation 4, we introduce an iterative procedure that computes V based on the sign of the previously estimated β. Thus, to estimate β at iteration k, we define each entry $V^i(\beta^i) \in V$ as $V^i(\beta^i) \approx V^i(\beta^i_{k-1})$, where k refers to the iteration index. This iteration also requires replacing σ^2 in Equation 7 by $d \times \sigma^2$, where d is the total number of iterations. Below we will give a further explanation for using the term $d \times \sigma^2$ but for now we define the semi-gaussian at iteration k as:

$$p_{SGd}(\beta) = exp\left(\frac{-(0-V\beta)^2)}{2d\sigma^2}\right) \tag{8}$$

The objective function maximized at each iteration is outlined in Equation 9. First, only the gaussian constraint is enforced (i.e. Equation 9a), and then the semi-gaussian constraint is imposed starting with Equation 9b, for d iterations.

$$J_0(\beta) = p(y|\beta)p_G(\beta) \tag{9a}$$

$$J_1(\beta) = J_0(\beta)p_{SGd}(\beta) \tag{9b}$$

$$\vdots$$

$$J_d(\beta) = J_{d-1}(\beta)p_{SGd}(\beta) = J \tag{9c}$$

Using the fact that $exp(-a/b) = exp(-a/(b \times d))^d$, taking the product of d exponential products p_{SGd} is equivalent to the true semi-gaussian p_{SG} in Equation 5. This proves that $J_d(\beta)$ in Equation 9c is equivalent to the total objective function J in Equation 4. In addition, this also illustrates why changing the variance in Equation 8 to $d \times \sigma^2$ is equivalent to a true semi-gaussian in Equation 5 with variance σ^2.

Now that we have given an overview of the ABCS method, in the next section, we describe how to estimate β for each step given in Equation 9.

2.2. ABCS Solutions

In this section, we present a two-step approach to solve for β, which we call the ABCS Solution.

2.2.1. Step 1

In step 1, we solve for the β which maximizes Equation 9a. This equation is equivalent to solving $y = H\beta$ without enforcing a sparseness constraint on β. As described in Appendix A, a closed form solution can be obtained for β since Equation 9a is represented as the product of two Gaussians. This closed for solution is given more explicitly by Equation 10:

$$\beta^* = \beta_1 = \left(I - P_0H^T(HP_0H^T+R)^{-1}H\right)\beta_0 + P_0H^T(HP_0H^T+R)^{-1}y \tag{10a}$$

Similarly, we can express the variance of β_1 as $P_1 = E\left[(\beta-\beta^1)(\beta-\beta^1)^T\right]$, given more explicitly by Equation 10b.

$$P_1 = (I - P_0H^T(HP_0H^T+R)^{-1}H)P_0 \tag{10b}$$

2.2.2. Step 2

Step 1 essentially solved for the pseudo-inverse of $y = H\beta$, of which there are many solutions. Using the solutions to Step 1 in Equation 10, we can rewrite Equation 9a as another gaussian as $p'(\beta|y) = p(y|\beta)p_G(\beta) = N(\beta|\beta_1, P_1)$. Therefore, we would now like to solve for the MAP estimate of β given the sparseness semi-gaussian constraint, in other words:

$$\beta^* = \arg\max_\beta p'(\beta|y)p_{SG}(\beta) \tag{11}$$

Because of the semi-gaussian approximation given in Equation 7, an iterative procedure is used in Step 2 to solve for β and P, as indicated by Equations 9b and 9c. Since the objective function at each iteration can be written as the product of gaussians, as shown by Equation 9c, a closed form solution can be obtained for β and P for each iteration. Equation 12 gives the recursive formula which solves Equation 11 at iteration k, for $k > 1$ to d.

$$\beta_k = \beta_{k-1} - \frac{P_{k-1}V^T}{VP_{k-1}V^T + d\times\sigma^2}V\beta_{k-1} \tag{12a}$$

$$P_k = \left[I - \frac{P_{k-1}V^T}{VP_{k-1}V^T + d\times\sigma^2}\right]P_{k-1} \tag{12b}$$

Thus, the ABCS approach allows for closed-form solution and the complete statistics of β, addressing the issue of the Lasso, Dantzig and Bayesian CS methods.

Let us denote $\hat{p}(\beta|y)$ as the posterior pdf obtained by using the ABCS solutions where the approximate semi-gaussian is used. In addition, denote $p(\beta|y)$ as the pdf when the true semi-gaussian is used to estimate β. It can be shown in (Carmi et al., 2009) that for a large number of iterations k, the Kullback-Leibler divergence between the two pdfs is bounded by $O(1/\sigma^2)$.

3. Phone Identification Features

In this section, we review the use of SR for classification (Sainath et al., 2010). We then introduce a novel set of features that are derived from this classification method, and demonstrate its success when applied in speech recognition.

3.1. Classification Using Sparse Representations

3.1.1. Motivation

The goal of classification is to use training data from k different classes to determine the best class to assign to a test vector y. First, let us consider taking all training examples n_i from class i and concatenating them into a matrix H_i as columns, in other words $H_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}] \in \Re^{m \times n_i}$, where $x_{i,j} \in \Re^m$ represents a feature vector from the training set of class i with dimension m. Given sufficient training examples from class i, (Wright et al., 2009) shows that a test sample y from the same class i can approximately be represented as a linear combination of the entries in H_i weighted by β, that is:

$$y = \beta_{i,1}x_{i,1} + \beta_{i,2}x_{i,2} + \ldots + \beta_{i,n_i}x_{i,n_i} \tag{13}$$

However, since the class membership of a test sample y is unknown, we define a matrix H to include training examples from all w classes in the training set, in other words the columns of H are defined as $H = [H_1, H_2, \ldots, H_w] = [x_{1,1}, x_{1,2}, \ldots, x_{w,n_w}] \in \Re^{m \times N}$. Here m is the dimension of each feature vector x and N is the total number of all training examples from all classes. We can then represent 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 which belong to the same class as y. This motivates us to solve for the sparse representation of β using any SR method, including ABCS.

3.1.2. Classification Rule

After using a SR method to solve $y = H\beta$, we must then assign y to a specific class. Ideally, all nonzero entries of β should correspond to the entries in H with the same class as y. In this ideal case, y will assign itself to training samples from one class in H, and we can assign y to the class which has the largest support in β. However, due to noise and modeling errors, β might have a non-zero value for more than one class. Therefore, we compute the l_2 norm for all β entries within a specific class, and choose the class with the largest l_2 norm support.

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 l_2 norm for β for class i as $\| \delta_i(\beta) \|_2$. The best class for y will be the class in β with the largest l_2 norm. Mathematically, the best class i^* is defined as

$$i^* = \max_i \| \delta_i(\beta) \|_2 \tag{14}$$

This SR technique was first explored in (Sainath et al., 2010) for Phonetic Classification on TIMIT and outperformed the GMM, SVM and kNN classifiers. In addition, in (Sainath et al., 2010), many different classification decision rules were explored, with the rule in Equation 14 yielding the best performance.

3.2. Novel Phone Identification Features

In this section, we discuss in more detail how we can use β from solving the initial $y = H\beta$ to create a new set of S_{pif} vectors. First, let us define a matrix $H_{phnid} = [p_{1,1}, p_{1,2}, \ldots, p_{w,n_w}] \in \Re^{r \times n}$, which has the same number of columns n as the original H matrix (i.e. meaning the same number of training examples), but a different number of rows r. Each $p \in \Re^r$ represents an entry in this matrix H_{phnid}. Recall that each sub-matrix $H_i \in H$ contains examples from a specific class. We can think of associating examples from each H_i class with a class index, for example entries from class H_0 belong to index 0, class H_1 to index 1, etc. We will define each $p \in H_{phnid}$ corresponding to feature vector $x \in H$ to be a vector with zeros everywhere except at the index corresponding to class of x. Figure 1 shows the H_{phnid} corresponding to H, where each p_i becomes a phone identification vector with a value of 1 corresponding to the class of x_i. Here r, the dimension of each p, is equivalent to the total number of classes.

$$\text{H}= \begin{array}{cccc} x_{0,1} & x_{0,2} & x_{1,1} & x_{2,1} \\ \hline 0.2 & 0.3 & 0.7 & 0.1 \\ 0.5 & 0.6 & 0.1 & 0.1 \\ \hline c=0 & c=0 & c=1 & c=2 \end{array} \longrightarrow \text{H}_{\text{phnid}} = \begin{array}{cccc} p_{0,1} & p_{0,2} & p_{1,1} & p_{2,1} \\ \hline 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}$$

Figure 1. H_{phnid} corresponding to H

Once β is found using SR, we use this same β to select important classes within the new dictionary H_{phnid} which is derived from the original H dictionary. Specifically, let us define a new feature vector S_{pif}, as $S_{pif} = H_{phnid}\beta^2$, where each element of β is squared, i.e., $\beta^2 = \{\beta_i^2\}$. Notice that we are using β^2, as this is similar to the $\| \delta_j(\beta) \|_2$ classification rule given by Equation 14. We will refer to this S_{pif} vector as a sparse representation phone identification (S_{pif}) vector. Each row j of the S_{pif} vector roughly represents the l_2 norm of β entries for class j.

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) (Sainath, 2009). For every test sample $y^t \in Y$, we solve $y^t = H^t \beta^t$ to compute a β^t. Then given this β^t, a corresponding S^t_{pif} vector is formed. Thus a series of S_{pif} vectors is created as $\{S^1_{pif}, S^2_{pif} \dots S^n_{pif}\}$. It is these new vectors, which take advantage of the non-parametric benefits of SRs, that we use as input features for recognition.

3.3. Choice of Parameters

In this section, details of specific parameters for the ABCS method are discussed further.

3.3.1. Construction of H

Ideally, H represents a dictionary of all training examples. However, pooling together all training data from all classes into H will make the columns of H large (i.e., greater than 2,000,000 for TIMIT), and will make solving for β using Equations 10 and 12 intractable. Furthermore, given $H \in \Re^{m \times N}$, (Donoho, 2006) shows that the following condition given by Equation 15 must hold in order for the SR solution of β to be sparse. Here s is the number of non-zero support vectors in β. For large N, Equation 15 will not hold.

$$m > 2s \log(N) \tag{15}$$

Therefore, to reduce the size of N and make the ABCS problem more practical from an implementation point of view, for each y, we find a neighborhood of closest points to y in the training set using a kd-tree. These k neighbors become the entries of H. k is chosen to be large enough to ensure that β is sparse and all training examples are not chosen from the same class, but small enough to ensure that Equation 15 holds. Furthermore, experiments in (Sainath et al., 2010) revealed that using a neighborhood of points from a kd-tree was an appropriate choice for H.

3.3.2. Choice of P_0

As discussed in Section 2.1, constants P_0 and β_0 must be chosen to initialize the ABCS algorithm. Recall that β_0 and the diagonal elements of P_0 all correspond to a specific class, from H defined in Section 3.1.1. We choose β_0 to be 0 since we do not have a very confident estimate of β and we assume its sparse around 0. We choose to initialize a diagonal P_0 where the entries corresponding to a particular class are proportional to the GMM posterior for that class. The intuition behind this is that the larger the entry in the initial P_0, the more weight is given to examples in H belonging to this class. Therefore, the GMM posterior picks out the most likely supports, and ABCS refines it further.

3.3.3. Choice of Class Identification

The S_{pif} vectors are defined based on the class labels in H. We explore two choice of class labels in this paper. First, we explore using 49 class labels, which is the exact number of phonemes in TIMIT. Since each phoneme corresponds to one dimension of the S_{pif} vector, one disadvantage of this approach is that a classification error could lead to β values from incorrect classes being over-emphasized. Thus some phonemes in the S_{pif} vector might dominate others, leading to potential recognition errors.

To address this issue, we also explore labeling classes in H by a set of context-independent (CI) HMM states. Specifically, we build a 3-state HMM for each of the 49 CI phonemes, giving a total of $3 \times 49 = 147$ states. A first pass recognition is performed to align each sample in the training data to one of the 147 states. While 147 increases the dimension of the S_{pif} vector, the elements in the vector are less sharp now since β values for a specific phoneme are more likely to be distributed within each of the CI states of this phoneme.

3.3.4. Using S_{pif} Vectors for Recognition

It is important to ensure a reasonable dynamic range for the S_{pif} vectors. In this section, we present two approaches that ensure numerical stability for practical applications such as phone recognition. First, since H is constructed using examples from a kd-tree, not all classes are contained in H. This implies that some of the entries in the S_{pif} vectors will be zero. Thus, we smooth out each entry by perturbing it with a small value sampled randomly from a uniform distribution, thus ensuring that no entry will be 0.

Secondly, β^t at each sample represents a weighting of entries in H^t that best represent test vector y^t. This makes it difficult to compare β^t values and the S_{pif} vectors across samples, which is necessary for recognition (Sainath, 2009). Therefore, to ensure that the values can be compared across samples, the S_{pif} vectors are normalized at each sample. Thus, the new S^t_{pif} at sample t is computed as $\frac{S^t_{pif}}{\|S^t_{pif}\|_1}$.

4. Experiments

Recognition experiments are conducted on the TIMIT (Lamel et al., 1986) acoustic phonetic corpus. The corpus contains over 6,300 phonetically rich utterances divided into three sets. The standard NIST training set consists of 3,696 sentences, used to train various models used by the recognizer. The development set is composed of 400 utterances and is used to train various classifier tuning parameters. The full test set includes 944 utterances, while the core test set is a subset of the full test set containing 192 utterances. In accordance with standard experimentation on TIMIT, the 61 phonetic labels are collapsed into a set of 49 for acoustic model training. For testing purposes, the standard practice is to collapse the 49 trained labels into a smaller set of 39 labels, ignoring the glottal stop [q]. All results are reported on the core test set.

The system in this work uses MFCC features as raw features. This feature set is based on an initial spectral analysis that uses 20-ms frames smoothed with a Hamming window and a 5-ms frame step. The final recognition feature set for all systems in this work are generated by concatenating raw features from nine consecutive frames and projecting to a 40-dimensional feature space using a Linear Discriminative Analysis (LDA) transform. These LDA features are used for both y and H to solve $y = H\beta$ at each frame using ABCS. Once a β is computed for each frame, the phone labels from H are using to construct a new dictionary H_{phnid} and S_{pif} vector, as described in Section 3.2.

A number of experiments are conducted to analyze the performance of the S_{pif} vectors. First, we analyze the frame-level accuracy of the SR classifier compared to a GMM and kNN method. The parameters of each classifier were optimized on the development set. Specifically, the number of k closest neighbors for kNN was learned. Also, for SRs the size of H was optimized to be 200 examples from the kd-tree. This number was chosen to satisfy the SR sparsity relationship in Equation 15, and till have the true class contained in the 200 examples more than 99% of the time.

Next, we explore the performance of S_{pif} for recognition, using both CI and Context-Dependent (CD) HMMs. A set of CI HMMs are trained using information from the phonetic transcription. Maximum Likelihood (ML) estimation is used to train parameters of the HMM. The output distribution of each CI state is a 32-component GMM. The CI models are then used for bootstrapping the training of a set of triphone CD ML-trained HMMs. Totally the CD system has 2,400 states and 15,000 Gaussian components, also optimized on the development set. A trigram language model is used for all experiments. We first compare the performance of the CI system using LDA and S_{pif} features, followed by an investigation of the behavior of the CD system with both of these features.

5. Results

5.1. Frame Accuracy

The success of S_{pif} first relies on the fact that the β vectors give large support to correct classes when computing $y = H\beta$ at each frame. Thus, the classification accuracy per frame, computed using Equation 14, should ideally be high. Table 1 shows the classification accuracy for the GMM, kNN and ABCS methods.

Classifier	Frame-Level Accuracy
GMM	51.8
kNN	62.1
ABCS	**64.0**

Table 1. Frame Accuracy on TIMIT Testcore Set

Notice that the ABCS technique offers significant improvements over the GMM method, again showing the benefit of the non-parametric ABCS classifier. In addition, ABCS also offers improvements over the kNN, showing the advantages of dynamically adjusting the support vector β per frame. The strength of the SR classifier over other classification techniques motivates us to further explore its use for recognition.

5.2. Recognition Results

Table 2 shows the phonetic error rate (PER) at the CI level for different features. Notice that both S_{pif} features outperform LDA features, showing the benefit of using a non-parametric technique to derive features with better frame classification accuracy. Notice that decreasing the sharpness of S_{pif} features by using 147 phones results in a decrease in error rate.

Features	PER
Baseline LDA Features	25.9
S_{pif} - 49 phones	25.7
S_{pif} - 147 phones	**25.3**

Table 2. PER on TIMIT Core Test Set - CI Level

Next, Table 3 shows the PER at the CD level for LDA and S_{pif} features. Please note we have only included S_{pif} results for 147 phones since it offered lower error rates than using 49 phones at the CI level. Again the S_{pif} features outperform the LDA features, and a Matched Pairs Sentence Segment Word Error

(MPSSWE) (Sainath, 2009) significance test indicates that S_{pif} result is statistically significant.

Features	PER
Baseline LDA Features	24.9
S_{pif} - 147 phones	**23.9**

Table 3. PER on TIMIT Core Test Set - CD Level

Finally, Table 4 compares our results to other CD-ML trained systems reported in the literature on TIMIT. The S_{pif} features offers the best result of all methods at the CD level for ML trained systems. This demonstrates the advantage of using the SR method to create S_{pif} vectors, which can be used in tandem with the parametric HMM.

System	PER (%)
LDA Features, IBM CD HMM	24.9
Monophone HTMs (Deng & Yu, 2007)	24.8
Heterogeneous Measurements (Halberstat & Glass, 1998)	24.4
S_{pif} Features, IBM CD HMM	**23.9**

Table 4. Comparison of CD ML Trained Systems on TIMIT Core Test Set

6. Conclusions

In this paper, the following key contributions were made:

- We presented a new SR method for learning sparseness parameters. Specifically, we introduced a sparseness promoting semi-gaussian prior. This allowed for an efficient closed-form solution in a probabilistic framework for estimating the sparseness parameters and avoids any parameter tuning.
- We demonstrated that using SR for phone classification at the frame level resulted in higher accuracy compared to the GMM and kNN methods.
- We derived a novel set of S_{pif} features and demonstrated how these features can be successfully applied in an HMM framework for phone recognition. This yields the best performance reported in the literature to date when the parameters of the HMM are trained using the Maximum Likelihood principle.

Acknowledgements

The authors would like to thank Avishy Carmi for his help in developing the theory behind the ABCS algorithm utilized in this paper. Also, thank you to Abhinav Sethy for many useful discussions and guidance.

A. Appendix

This section shows how to obtain a closed-form solution for β when J_0 in Equation 9a is represented as the product of Gaussians. Specifically J_0 is written as:

$$J_0(\beta) = p(y|\beta)p_G(\beta) = N(y|H\beta, R) \times N(\beta|\beta_0, P_0) \tag{16}$$

Now, we can maximize J_0 with respect to β by solving:

$$\frac{\partial \log p(J_0(\beta))}{\partial \beta} = 0 \tag{17}$$

This in turn yields

$$\beta_1 = \left(P_0^{-1} + H^T R^{-1} H\right)^{-1} \left[P_0^{-1}\beta_0 + H^T R^{-1} y\right] \tag{18}$$

The term $\left(P_0^{-1} + H^T R^{-1} H\right)^{-1}$ requires taking the inverse of an $n \times n$ matrix. To reduce inverse computation, using the matrix inversion lemma (Carmi et al., 2009), this term can be rewritten as:

$$\left(I - P_0 H^T \left(H P_0 H^T + R\right)^{-1} H\right) P_0 \tag{19}$$

Multiplying Equation 19 by the term $\left[P_0^{-1}\beta_0 + H^T R^{-1} y\right]$ in Equation 18 gives the following expression for β_1 after some algebra.

$$\beta^* = \beta_1 = \left(I - P_0 H^T (H P_0 H^T + R)^{-1} H\right)\beta_0 + P_0 H^T (H P_0 H^T + R)^{-1} y \tag{20}$$

Thus, a closed form solution can be obtained for β assuming that J is the product of Gaussians. In addition, the inverse is computed for an $m \times m$ matrix. The computation for P has a similar derivation to β and can be found in (Carmi et al., 2009).

References

Candes, E. J., Romberg, J., and Tao, T. Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information. *IEEE Trans. on Information Theory*, 52:489–509, 2006.

Carmi, A., Gurfil, P., Kanevsky, D., and Ramabhadran, B. ABCS: Approximate Bayesian Compressed Sensing. Technical report, Human Language Technologies, IBM, 2009.

Chartrand, R. Exact Reconstruction of Sparse Signals via Non-Convex Optimization. *IEEE Signal Processing Letters*, 14:707–710, 2007.

Deng, L. and Yu, D. Use of Differential Cepstra as Acoustic Features in Hidden Trajectory Modeling for Phonetic Recognition. In *Proc. ICASSP*, 2007.

Deselaers, T., Heigold, G., and Ney, H. Speech Recognition With State-based Nearest Neighbour Classifiers. In *Proc. ICSLP*, 2007.

Donoho, D. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

Halberstat, A. and Glass, J. Heterogeneous Measurements and Multiple Classifiers for Speech Recognition. In *Proc. ICSLP*, 1998.

Ji, S., Xue, Y., and Carin, L. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56:2346–2356, 2008.

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

Sainath, T. N. *Applications of Broad Class Knowledge for Noise Robust Speech Recognition.* PhD thesis, Massachusetts Instutite of Technology, 2009.

Sainath, T. N., Carmi, A., Kanevsky, D., and Ramabhadran, B. Bayesian Compressive Sensing for Phonetic Classification. In *To Appear in ICASSP*, 2010.

Wright, J., Yang, A.Y., Ganesh, A., Sastry, S. Shankar, and Ma, Y. Robust Face Recognition via Sparse Representation. *IEEE Trans. on PAMI*, 31:210–227, 2009.