BAYESIAN COMPRESSIVE SENSING FOR PHONETIC CLASSIFICATION

Tara N. Sainath[1]*, Avishy Carmi*[2]*, Dimitri Kanevsky*[1] *and Bhuvana Ramabhadran*[1]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
[2] The Signal Processing Group, Department of Engineering, University of Cambridge, UK
[1]tnsainat@us.ibm.com, [2]ac599@cam.ac.uk, [1]{kanevsky, bhuvana}@us.ibm.com

ABSTRACT

In this paper, we introduce a novel bayesian compressive sensing (CS) technique for phonetic classification. CS is often used to characterize a signal from a few support training examples, similar to k-nearest neighbor (kNN) and Support Vector Machines (SVMs). However, unlike SVMs and kNNs, CS allows the number of supports to be adapted to the specific signal being characterized. On the TIMIT phonetic classification task, we find that our CS method outperforms the SVM, kNN and Gaussian Mixture Model (GMM) methods. Our CS method achieves an accuracy of **80.01%**, one of the best reported result in the literature to date.

Index Terms— Compressive sensing, Pattern classification

1. INTRODUCTION

Parametric modeling techniques, such as Gaussian Mixture Models (GMMs) continue to be extremely popular for pattern recognition tasks in speech recognition. While these techniques 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. Alternatively, non-parametric methods, including k-nearest neighbors (kNNs) [1] and Support Vector Machines (SVMs) [2] utilize the details of actual training examples. Since the number of training examples in speech tasks can be very large, both of these non-parametric techniques adopt a common theme of using a few number of training examples to characterize a test vector. However, both techniques do not adapt their 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. Similarly kNN methods characterize a 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. In theory, adapting the number of k points per test vector could help.

In recent years, compressive sensing (CS) [3] has become a popular technique to represent a test signal by a few small training examples. Mathematically speaking, in a typical CS formulation, a matrix H is constructed consisting of possible examples of the signal, that is $H = [h_1; h_2 \ldots; h_n]$. To reconstruct a signal y from H, CS solves the equation $y = H\beta$. A sparseness condition is enforced on β, such that it selects a small number of examples from H to describe y. CS has typically been used for signal compression and recover, though recently it has also shown success in face recognition [4] over linear SVM and 1-NN methods. One benefit of CS is that for a given test example, CS adaptively selects the relevant support vectors β from the training set H. It is this benefit of CS that motivates us to explore its benefit over SVMs, kNNs and GMMs in speech recognition. Specifically, we compare the performance of CS to the three classifiers on the benchmark TIMIT phonetic classification task [5].

In this paper, we introduce three novel contributions in applying CS for phonetic classification. First, we explore a novel method for solving the CS problem. State-of-the-art methods for sparse signal recovery typically utilize the Lasso technique to solve for the support vectors [6]. However, this method only provides a point estimate for β, and can thus be considered to be a sub-optimal solution. Recently, we have introduce an Approximate Bayesian Compressive Sensing (ABCS) formulation [7] which allows us to derive a closed-form recursion for estimating sparseness parameters. We have found that this ABCS method offered improvements over to the Lasso method for a small fMRI binary classification task [7]. In this paper, we explore the generality of ABCS for a large scale 39-phone TIMIT classification task. In addition, the benefit of this Bayesian approach is that it allows us to build CS on top of other Bayesian classifiers, for example a GMM. Therefore, we will show that one can think of CS as being a non-parametric classifier on top of a GMM.

A CS problem simply gives a solution of sparse vectors β given a set of training examples in H and a test vector y. Our second contribution introduces a framework of using the CS solution to generate an actual classification decision. Finally, many non-parametric methods such as SVM have been shown to offer better performance when the data is mapped to a nonlinear space. Thus far, CS techniques represent y as a combination of examples in H, where both y and H reside in the original input space. Our third contribution is the introduction of nonlinearity into CS but constructing H such that it contains both linear and non-linear features. We find that this non-linear CS approach offers improvements over linear CS.

Our experiments on TIMIT are conducted with two different feature sets. Using MFCC features, we find that our nonlinear CS approach outperforms both the GMM and kNN methods, and offers similar performance to the SVM. Next, using discriminative feature space Boosted Maximum Mutual Information (fBMMI) features, we find that CS technique offers an accuracy of 80.01% which outperforms both the GMM, kNN and SVM methods, and is close to the best reported result of 81.3% in the literature to date ([8]).

The rest of this paper is organized as follows. Section 2 reviews our novel ABCS formulation, while Section 3 discusses applying CS to phonetic classification. 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. ABCS DERIVATION

In this section, we formulate the ABCS solution. Ultimately, we would like to use CS to solve the following problem:

$$y = H\beta \ \texttt{s.t.} \quad \| \beta \|_1^2 < \epsilon \ \texttt{for} \ \beta \tag{1}$$

Here $\| \beta \|_1^2 < \epsilon$ denotes a sparseness-promoting semi-gaussian constraint, which we will describe in more detail below. In addition, y is a frame of data from the test set such that $y \in \Re^m$ where m is the dimension of the 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 distribution as:

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

Assuming β is a random variable with prior $p(\beta)$, we can obtain the maximum a posteriori (MAP) estimate for β 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 actual the product of two prior constraints, namely a Gaussian constraint $p_G(\beta)$ and a semi-gaussian constrain $p_{SG}(\beta)$ which enforces sparseness. Below, we present a two-step solution to solve the following problem in the ABCS framework.

$$\beta^* = \arg\max_\beta p(y|\beta)p_G(\beta)p_{SG}(\beta) \quad (3)$$

2.1. Step 1

In step 1, we solve for the β which maximizes the following expression. Equation 4 is equivalent to solving the equation $y = H\beta$ without enforcing a sparseness constraint on β [7].

$$\beta^* = \arg\max_\beta p(y|\beta)p_G(\beta) \quad (4)$$

We assume that $p_G(\beta)$ is a Gaussian, i.e., $p_G(\beta) = N(\beta|\beta_0, P_0)$. Here β_0 and P_0 are initialized statistical moments utilized in the algorithm. In [7], we show that the solution to Equation 4 has a closed form solution given by Equations 5a and 5b.

$$\beta^* = \beta_1 = \left(I - P_0H^T(HP_0H^T + R)^{-1}H\right)\beta_0 + P_0H^T(HP_0H^T + R)^{-1}y \quad (5a)$$

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 5b.

$$P_1 = (I - P_0H^T(HP_0H^T + R)^{-1}H)P_0 \quad (5b)$$

2.2. Step 2

Step 1 essentially solved for the pseudo-inverse of $y = H\beta$, of which there are many solutions. In this section, we impose an additional constraint that β will have a sparseness-promoting semi-Gaussian prior, as given by Equation 6. Here σ^2 is a constant parameter which controls the degree of sparseness of β.

$$p_{SG}(\beta) = exp\left(-\frac{||\beta||_1^2}{2\sigma^2}\right) \quad (6)$$

Given the solutions to Step 1 in Equations 5a and 5b, we can simply rewrite Equation 4 as another Gaussian as $p'(\beta|y) = p(y|\beta)p_G(\beta) = N(\beta|\beta_1, P_1)$. Therefore, let us assume now that we would like to solve for the MAP estimate of β given the constraint that it is semi-gaussian, in other words:

$$\beta^* = \arg\max_\beta p'(\beta|y)p_{SG}(\beta) \quad (7)$$

In order to represent $p_{SG}(\beta)$ as a Gaussian the same way that $p(y|\beta)$ in Equation 2 was represented, let us define β^i to be the i^{th} entry of the vector β. We introduce a matrix $\hat{H}$ of which the entries are set as $\hat{H}^i(\beta^i) = \text{sign}(\beta^i)$, for $i = 1, \ldots, n$. Here $\hat{H}^i(\beta^i) = +1$ for $\beta^i > 0$, $\hat{H}^i(\beta^i) = -1$ for $\beta^i < 0$, and $\hat{H}^i(\beta^i) = 0$ for $\beta^i = 0$. This matrix $\hat{H}$ is motivated from the fact that

$$\| \beta \|_1^2 = (\sum_i (|\beta^i|))^2 = (\sum_i (\hat{H}^i(\beta^i)\beta^i))^2 = (\hat{H}\beta)^2 \quad (8)$$

Substituting the expression for $\| \beta \|_1^2$ given in Equation 8 and assuming a that $y = 0$, we can rewrite Equation 6 as Equation 9. Notice that Equation 9 has the same form as Equation 2 with H and R now replaced by $\hat{H}$ and σ respectively.

$$p_{SG}(\beta) = p(y = 0|\beta) = exp\left(\frac{-(0 - \hat{H}\beta)^2}{2\sigma^2}\right) \quad (9)$$

The only problem with using Equation 7 to solve for β is the dependency of $\hat{H}$ on β in Equation 5a. Therefore, we make an assumption, by calculating $\hat{H}$ based on the sign of the previously estimated β. In other words $\hat{H}^i(\beta^i) \approx \hat{H}^i(\beta^i_{k-1})$. With this approximation we can use Equations 5a and 5b to solve Equation 9. However, because of this semi-gaussian approximation, we must estimate β and P iteratively. As [7] shows, this iteration also requires that we set σ^2 as $\sigma^2 \times d$, where d is the total number of iterations of Step 2. Equation 10 gives the recursive formula which solves Equation 7 at iteration k for $k > 1$ to d. Note that $p'(\beta|y) = N(\beta|\beta_{k-1}, P_{k-1})$.

$$\beta_k = \beta_{k-1} - \frac{P_{k-1}\hat{H}^T}{\hat{H}P_{k-1}\hat{H}^T + d \times \sigma^2}\hat{H}\beta_{k-1} \quad (10a)$$

$$P_k = \left[I - \frac{P_{k-1}\hat{H}^T}{\hat{H}P_{k-1}\hat{H}^T + d \times \sigma^2}\right]P_{k-1} \quad (10b)$$

In [7], we show that for large σ^2 and large k, the estimate of β and P using the approximate semi-gaussian given in Equation 9 is bounded from the estimate of these parameters for the true semi-gaussian given in Equation 6 by $O(1/\sigma^2)$.

3. ABCS FOR CLASSIFICATION

Now that we have described the ABCS formulation, in this section we discus how CS can be used for classification.

3.1. Classification Based on Sparse Representation

The goal of classification is to use training data from k different classes to determine the best class to assign to test vector y. 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, [4] shows that a test sample y from the same class can be represented as a linear combination of the entries in H_i weighted by β, that is:

$$y = \beta_{i,1}x_{i,1} + \beta_{i,2}x_{i,2} + \ldots + \beta_{i,n_i}x_{i,n_i} \quad (11)$$

However, since the class membership of y is unknown, we define a matrix H to include training examples from all k classes in the training set, in other words the columns of H are defined as $H = [H_1, H_2, \ldots, H_k] = [x_{1,1}, x_{1,2}, \ldots, x_{k,n_k}] \in \Re^{m \times N}$. Here m is the dimension of each feature vector x and N is the total number of all training examples from all classes. 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. This motivates us to solve for the sparse representation of β using the ABCS formulas presented in Section 2.

3.2. Classification Rule

Now that we have described our method to solve for β is via ABCS, we now discuss how to assign y as belonging 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 one training example from H, and we can assign y to the class which has the largest support in β. However, due to noise and modeling error, β belonging to other classes could potentially be non-zero. 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 $\parallel \delta_i(\beta) \parallel_2$. 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

$$i^* = \max_i \parallel \delta_i(\beta) \parallel_2 \tag{12}$$

3.3. Construction of H

Since parametric techniques such as GMMs continue to be popular for pattern recognition tasks in speech recognition, this implies the availability of training data required to sufficiently estimate model parameters. Pooling together all training data from all classes into H will make the columns of H large (i.e., can be greater than 100,000 for TIMIT), and will make solving for β using Equations 5 and 10 intractable. Furthermore, given $H \in \Re^{m \times N}$, [3] shows that condition given by Equation 13 must hold in order for the CS solution of β to be sparse. Here s is the number of non-zero support vectors in β. For sufficiently large N, Equation 13 will not hold.

$$m > 2s \log(N) \tag{13}$$

Therefore, to reduce the size of N and make ABCS problem more solvable, for each y, we find a neighborhood of closest points to y in the training set using a kd-tree [1]. These k neighbors become the entries of H. k is chosen to be in the large to ensure that β is sparse and all training examples are not chosen from the same class.

3.4. 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. 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 anyways. 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 initial P_0, the more weight is given to examples in H belonging to this class in ABCS. Therefore, the GMM posterior picks out the most likely supports, and ABCS provies an addition step by using the actual training data to refine these supports.

3.5. Non-Linear CS

The traditional CS implementation represents y as a linear combination of samples in H. Many pattern recognition algorithms, such as SVMs [2] have shown better performance can be achieved by a nonlinear mapping of the feature set to a higher dimensional space. After this mapping, a weight vector w is found which projects all dimensions within a particular feature vector to a single dimension where different classes are linearly separable. We can think of this weight vector w as selecting some linear combination of dimensions within a feature vector to make it linearly separable. The goal of CS is to find a linear combination of actual features, not dimensions within a feature vector. Therefore, we introduce nonlinearity into CS, by constructing H such that the entries of H themselves are nonlinear. For example, one such nonlinearity is to square all the elements within H. That is if we define $H_{lin} = [x_{1,1}, x_{1,2}, \ldots, x_{k,n_k}]$, then H^2 is defined as $H^2 = [x_{1,1}^2, x_{1,2}^2, \ldots, x_{k,n_k}^2]$ and similarly H^3 would take cubed products of each of the x entries. We could also take products between different x_i as $H_{inner} = [x_{1,1}x_{1,2}, x_{1,1}x_{1,3} \ldots, x_{k,8}x_{k,n_k}]$. We then take a specific nonlinear H_{nonlin} and combine it with the linear H_{lin} to form a new $H_{tot} = [H_{lin}, H_{nonlin}]$ and use ABCS to solve for β. In Section 5.1, we discuss the performance of the ABCS algorithm for different choices of nonlinear H.

4. EXPERIMENTS

Classification experiments are conducted on TIMIT [5] 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 [9], the 61 phonetic labels are collapsed into a set of 48 for acoustic model training, ignoring the glottal stop [q]. For testing purposes, the standard practice is to collapse the 48 trained labels into a smaller set of 39 labels [9].

We compare two sets of segmental features in our experiments. First, we represent each frame in our signal by a 13 dimensional MFCC. We split each phonetic segment into thirds, taking the average of these frame-level features around 3rds, and splice them together to form a 39 dimensional vector. This allows us to capture time dynamics into each segment. Then, at each segment, segmental feature vectors to the left and right of this segment are joined together and a Linear Discriminative Analysis (LDA) transform is applied to project 117 dimensional feature vector down to 40 dimensions. We also explore using discriminative features, namely Feature Space Boosted Maximum Mutual Information (fBMMI) features. Again, we first compute these features at a frame level, and then employ a similar averaging scheme as with the MFCC features to create a segmental feature vector. Then, at each segment, features from the neighboring frames are spliced together and an LDA is applied to reduce these features down to 40 dimensions.

First, we analyze the performance of the CS classifier for different choices of linear and nonlinear H as described in Section 3.4. Next, we compare the performance of CS with three other standard classifiers used on this task, namely a Gaussian Mixture Model (GMM), Support Vector Machine (SVM) [2] and k-nearest Neighbors (kNN) classifier [1]. The parameters of each classifier were optimized for each feature set on the development set. Specifically, we found that modeling each phone as a 16-component GMM was appropriate. The kernel type and parameters within this kernel were optimized for the SVM. In addition, the number of k closest neighbors for kNN was also learned. And finally, for CS the size of H_{lin}

was optimized to be 200 examples from the kd-tree. In addition to compute H_{nonlin}, 100 columns were randomly chosen from H_{lin} to compute each type of nonlinear H.

5. RESULTS

5.1. Performance for Different H

Table 1 shows the accuracy on the development set for different choices of H using MFCC features. Notice that the nonlinear CS-$H_{lin}H^2$ method offers improvements over the linear CS-H_{lin} method. Taking $H_{lin}H^2H^3$ offers addition improvements, though overtraining occurs when higher order features past H^3 are used. Furthermore, there is very little difference between squaring individual entries of H (i.e. $H_{lin}H^2$) or taking products between different entries of H (i.e., $H_{lin}H_{inner}$). While not shown here, similar trends were also observed for fBMMI features. Since the CS-$H_{lin}H^2H^3$ method offers the best performance of the CS methods, we will report the results for this classifier in subsequent sections.

Method	Dev-MFCC
CS-H_{lin}	76.64
CS-$H_{lin}H^2$	76.84
CS-$H_{lin}H^2H_{inner}$	76.53
CS-$H_{lin}H^2H^3$	**76.89**
CS-$H_{lin}H^2H^3H^4$	76.86

Table 1. Accuracy for Different H using MFCC features

5.2. Comparison of Different Classifiers

Table 2 compares the performance of the CS classifier with the GMM, kNN and SVM methods for both MFCC and fBMMI features. Classifiers which are not statistically significant from the CS classifier, as confirmed by McNemar's Test, are also indicated by '='. First, notice that when MFCC features are used, CS outpeforms both then kNN and GMM methods, and offers similar performance to the SVM. When discriminative features are used, the GMM technique is closely matched to the SVM though CS is able provide further gains over these two methods. This is one of the benefits of CS - a discriminative non-parametric classifier built on top of the GMM.

Method	MFCC	fBMMI
GMM	74.19	78.31
kNN	73.69	79.58 (=)
SVM	76.20 (=)	78.38
CS-$H_{lin}H^2H^3$	**76.44**	**80.01**

Table 2. Accuracy for Different Classifiers on TIMIT Testcore Set

5.3. Analysis of Results

To better understand the gains achieved by the CS classifier compared to the other three techniques, Figure 1 plots the relative difference in error rates within 6 broad phonetic classes (BPCs) for CS compared to the three other methods. First, notice that CS offers improvements over the GMM in all BPCs, again confirming its benefit of a non-parametric discriminative classifier on top of the GMM. Secondly, while the SVM technique offers improvements over the CS method in the vowel/semi-vowel class, the CS method significantly outperforms the SVM in the weak fricative, stop and closure classes. Finally, the CS method offers slight improvements over the kNN method in the nasal, strong fricative and stop classes, while kNN offers slight improvements in the vowel, weak fricative and closure classes. Thus, we can see that with the exception of the GMM, the gains from CS do not come from it outperforming the kNN and SVM techniques within all BPCs, but only within certain BPCs.

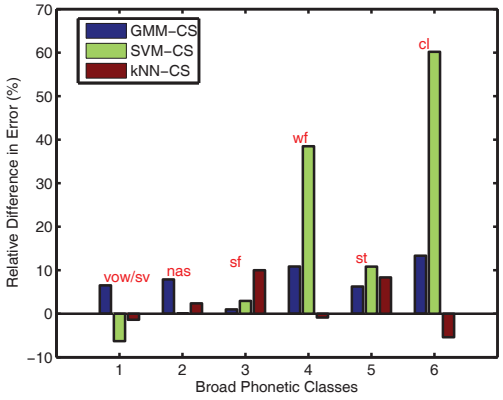

Fig. 1. Rel. Diff. in Error Rates Between CS and Other Methods

6. CONCLUSIONS AND FUTURE WORK

In this paper, we explored the use of a bayesian CS approach and applied this to phonetic classification. On the TIMIT phonetic classification task, we found that our nonlinear CS method outperformed the GMM, SVM and kNN methods and offered an accuracy of 80.01%, close to the best reported result in the literature. In the future, we would like to explore the use of CS for speech recognition. Specifically, since CS appears to offer extra discrimination on top of a GMM, we would like to explore using it to provide acoustic scores for Hidden Markov Model (HMM) states, the output of which follows a GMM distribution. In addition, we would like to explore methods to decrease the computational complexity of the CS method, making it more usable for speech recognition tasks.

7. REFERENCES

[1] D. Mount and S. Arya, *ANN: A Library for Approximate Nearest Neighbor Searching*, 2006, Software available at http://www.cs.umd.edu/ mount/ANN/.

[2] C. Chang and C. Lin, *LIBSVM: A Library for Support Vector Machines*, 2001, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, 2006.

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

[5] 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.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] A. Carmi, P. Gurfil, D. Kanevsky, and B. Ramabhadran, "ABCS: Approximate Bayesian Compressed Sensing," Tech. Rep., Human Language Technologies, IBM, 2009.

[8] H. Chang and J. Glass, "Hierarchical Large-Marging Gaussian Mixture Models for Phonetic Classification," in *Proc. ASRU*, 2007.

[9] K. F. Lee and H. W. Hon, "Speaker-independent Phone Recognition Using Hidden Markov Models," *IEEE Transacations on Acoustics, Speech and Signal Processing*, vol. 37, pp. 1641–1648, 1989.