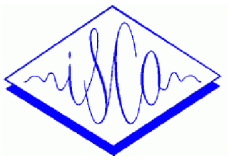

INTERSPEECH 2012
ISCA's 13th Annual Conference
Portland, OR, USA
September 9-13, 2012

Are Sparse Representations Rich Enough for Acoustic Modeling?

Oriol Vinyals[1], *Li Deng*[2]

[1]University of California at Berkeley (Berkeley, CA, US)
[2]Microsoft Research (Redmond, WA, US)
vinyals@eecs.berkeley.edu, deng@microsoft.com

Abstract

We propose a novel approach to acoustic modeling based on recent advances in sparse representations. The key idea in sparse coding is to compute a compressed local representation of a signal via an over-complete basis or dictionary that is learned in an unsupervised way. In this study, we compute the local representation on speech spectrogram as the raw "signal" and use it as the local sparse code to perform a standard phone classification task. A linear classifier is used that directly receives the coding space for making the classification decision. The simplicity of the linear classifier allows us to assess whether the sparse representations are sufficiently rich to serve as effective acoustic features for discriminating speech classes. Our experiments demonstrate competitive error rates when compared to other shallow approaches. An examination of the dictionary learned in sparse feature extraction demonstrates meaningful acoustic-phonetic properties that are captured by a collection of the dictionary entries.

Index Terms: sparse coding, acoustic modeling, phone recognition, acoustic-phonetic properties

10.21437/Interspeech.2012-8

1. Introduction

Sparse coding has achieved state-of-the-art performance on many applications in computer vision and has attracted much research in recent years. The main idea behind sparse coding is to map the original feature space (e.g. pixels, spectrograms, etc.) to a (typically larger) sparse representation. Some evidence has been shown that this strategy may be utilized by the first stages of vision [7]. In particular, it seems that we have specialized neurons firing only when particular patterns are present (e.g. an edge in an image at a certain orientation).

Recently, the machine learning community has been exploring why sparse representations followed by a linear classifier performs as well as carefully designed features (such as SIFT or MFCC) followed by non-linear classifiers (e.g. kernel SVMs and Deep Neural Networks). In [12] a theoretical formulation shows that, if we want to learn a general non-linear mapping function (with some constrains such as Lipschitz smoothness) from the original features to a label, one can approximate such a function with a local, sparse coding step followed by a linear classifier. This is because one can approximate any smooth, non-linear function locally by a piecewise linear function to an arbitrarily accurate degree.

The paradigm of sparse coding followed by a linear classifier is appealing as learning linear classifiers is a well understood problem, both in terms of optimization (as it usually involves solving a convex objective function), and of overfitting (linear models, due to their simplicity, tend to exhibit less overfitting). In addition, the learning is very simple and time efficient, and the models can be interpreted thanks to the simple classification function.

Nonetheless, sparse coding has not been popular in speech recognition research until recently. The work in [10] is the closest to ours, and uses sparse coding as an input of a neural network (NN) for phone recognition. The work in [9] explores inducing sparsity in hidden layers of a NN. Another approach is to do exemplar based speech recognition based on sparse representations, which was recently proposed in [8] with promising results. Also, the well established neural network approaches to acoustic modeling can be seen as sparse coding (as the unsupervised pre-training is akin to dictionary learning), and the sigmoid non-linearity can be interpreted as a sparsifying factor. Another important reason why sparse coding has had little attention is the fact that, even though specialized software and optimization techniques exist for efficient coding, to convert the original feature vector into a sparse code, one has to solve an optimization problem for each sample during train and test time. This makes the approach not as computationally attractive as a neural network or a GMM, which typically can be computed with a few matrix/vector multiplications.

However, recent advances in understanding sparse coding observed that the set of basis (or dictionary) was not crucial, and that the encoding could be carried out with a simple matrix vector multiplication [2]. Empirical results suggest that the data preprocessing step is more important than learning the dictionary in achieving good performance when using fast approximations to sparse coding. In particular, zero-phase whitening filters (ZCA) and contrast normalization are typically applied as a preprocessing step to the data.

The main contributions of this paper is to apply sparse representations based on a very simple and fast approximation to sparse coding on a widely studied acoustic modeling benchmark; giving an interpretation of the unsupervised learned basis that are most relevant for certain classes (phones); and providing an alternative to deep learning that can be implemented in a few lines of code[1].

2. Related Work

In this section we give an overview of sparse coding, as well as previous related work in both speech and vision. It is out of the scope of this paper to describe all previous work on other techniques based on sparsity, such as compressed sensing, and the authors encourage interested readers to refer to [5] for further information.

2.1. Sparse Coding

Sparse representations were initially motivated by neuroscience in [7], where it was shown that the receptive fields in the early

[1]http://www.eecs.berkeley.edu/~vinyals/code

stages of the virtual cortex could be learned by applying sparse coding to a set of natural images. In sparse coding, we are given a set of d dimensional observations $\mathbf{x}_i$ (e.g. images), and the objective is to jointly learn a code $\mathbf{c}_i$ (of dimension k) and a dictionary $\mathbf{D}$ such that the reconstruction of the original signal is as close as possible in L2 norm, and the codes are sparse through the following objective function [7]:

$$\min_{\mathbf{c}_i,\mathbf{D}} \sum_i ||\mathbf{x}_i - \mathbf{D}\mathbf{c}_i||_2^2 + \alpha||\mathbf{c}_i||_1 \quad (1)$$

Note that $\mathbf{D}$ is a $d \times k$ matrix whose columns are the basis (or dictionary elements or codebooks), and α a parameter to control the sparsity of our solution (note that if $\alpha = 0$, then a solution with 0 error exists if we choose $k \geq d$). Typical approaches to solve Eq. 1 involve coordinate descent on two convex sub-problems, alternating between minimizing $\mathbf{D}$ (simple least squares), and $\mathbf{c}_i$ (well studied problem in the compressed sensing community).

Understanding sparse coding has been of interest for the machine learning community, as excellent results were obtained in computer vision by adopting a sparse coding step followed by a linear classifier (typically a linear Support Vector Machine (SVM)), in contrast to learning complicated kernel functions and features to solve object recognition [6]. In recent work, the authors of [12] gave the explanation that an overcomplete sparse coding scheme (one where the codebook size, k, is larger than the dimensionality of the input feature space d) could serve as a way to locally encode a smooth non-linear function, which can then be approximated as a globally linear function, and thus learned through, for example, linear SVM. Several state-of-the-art results have been published on computer vision benchmarks such as object recognition, face detection, or action recognition [6, 1].

2.1.1. Application to Acoustic Modeling

One of the main drawbacks for applying sparse coding in speech recognition is that, for a fix $\mathbf{D}$, for each sample $\mathbf{x}_i$ (a window containing several spectrotemporal frames of speech) one has to find $\mathbf{c}_i$ through minimizing Eq. 1. This problem has been extensively studied in the optimization community, and efficient solutions exist that are only a few times slower than a straightforward matrix-vector multiplication (the typical cost of other approaches such as Deep Neural Networks (DNN) or Gaussian Mixture Models (GMM)). In fact, vector quantization, a technique that has been used to model emission probabilities for Hidden Markov Models (HMM), can be seen as a crude approximation to sparse coding, where a signal $\mathbf{x}_i$ is coded with just the closest element in the dictionary in the following way:

$$\mathbf{c}_i = \arg\min_{\mathbf{c}_i} ||\mathbf{x}_i - \mathbf{D}\mathbf{c}_i||_2^2$$
$$s.t. ||\mathbf{c}_i||_0 = 1$$

where $||.||_0$ counts the number of non-zero elements.

Another possible alternative to this is to impose a sparse penalty to either the weights learned, or the activations of the hidden neurons in a DNN setting [9, 11]. At decoding time, the cost does not change (and in fact one can obtain speed ups if the weights become sparse), but the model still poses a non-convex objective function to find the parameters of a DNN.

In [10], the authors proposed to use sparse coding on the spectrotemporal representation of the acoustic signal, but their codebook size was fairly small, and the dictionary learned did not seem to capture the basics of human speech generation. Furthermore, a more complicated non-linear classifier was used to map the code to phone posterior estimates.

In light of recent developments and simplifications found in [2], we develop a very simple and efficient scheme to perform phone recognition based on an overcomplete sparse representation of the signal followed by a simple linear SVM classifier.

3. Proposed Method

In [2], two key observations motivated our simplified sparse coding scheme for acoustic modeling. First, the fact that the dictionary (or set of basis vectors) learned is not as important as the coding step, and that even random samples can help expressing a sample locally. This is in accordance to the result found in [10], where the difference between a randomly initialized basis and the one found by sparse coding yielded similar results. Secondly, one can replace solving the optimization problem in Eq. 1, by an inner product followed by a sparsity inducing non-linearity, which was found to give similar results in various vision benchmarks.

Furthermore, two data normalization steps help in terms of capturing invariances and aiding the (simplified) dictionary learning step:

- The first is to normalize each $\mathbf{x}_i$ by subtracting its mean and dividing by the standard deviation, commonly known as contrast normalization (Line 2 in Algorithm 1). Thus, for example, having a shift in all the value for a given sample gives the same feature, which helps making the feature invariant to different illumination conditions. In the spectrotemporal domain, this could, in principle, remove some valuable information, and we empirically verify this in Section 4.
- The second is to perform ZCA whitening of the data (Line 3 in Algorithm 1), as means to "spread" the data $\mathbf{X}$ uniformly around the sphere in order to help the simplified dictionary learning step (which is a modification of standard k-means). Even though the effect for this in images is quite important, the benefit of performing this step for speech data seems less obvious.

After normalizing the data, we form a dictionary by performing Orthogonal Matching Pursuit 1 (OMP1) [2] (Line 4 in Algorithm 1), which can be seen as a modification to the k-means algorithm. Thus, this algorithm can be applied to large amounts of data and is easy to parallelize (we ran it on the 1.1 million samples in TIMIT on a single machine in less than 20 minutes). Examples of dictionary elements learned are shown in Figure 1. The basis seem to capture relevant information about formants and energy distribution that will be used to code a given spectrogram.

The encoding step is trivial, involving just a matrix multiplication to code the data, followed by a element-wise max operation to induce sparsity (Line 5 in Algorithm 1):

$$\mathbf{c}_i = \max(0, \mathbf{D}^T\mathbf{x}_i - \alpha)$$

where α is a parameter that controls the amount of sparsity introduced, and was set to 0.25, the same value that was found via cross validation on vision benchmarks. The last step involves the discriminative training of a linear function on the code space, and we use a linear SVM where the data is $\mathbf{C}$ and the corresponding labels are phone states (Line 6 in Algorithm 1).

Note that a single machine implementation will require to store the matrix $\mathbf{C}$, which can be larger than the data itself as typically $k > d$. For large datasets as in speech, where N is also large, both k-means and SVM are trivial to parallelize via map-reduce, which is a big advantage when comparing to training a DNN, which is typically done with stochastic gradient descent, much harder to parallelize.

The complete process for performing acoustic modeling from the spectrotemporal features is described in Algorithm 1.

Algorithm 1 Acoustic Modeling with Sparse Representations

1: // $\mathbf{x}_i$ is the i-th spectrotemporal frame, $\mathbf{X}$ is the $d \times N$ matrix whose columns are $\mathbf{x}_i$, and $\mathbf{Y}$ is the N dimensional vector with phone labels for each sample
2: Normalize each spectrotemporal frame $\mathbf{x}_i$
3: Apply ZCA whitening on the whole dataset $\mathbf{X}$
4: Run OMP1 on $\mathbf{X}$ to construct a $d \times k$ dictionary matrix $\mathbf{D}$
5: Form $\mathbf{C}$, a $k \times N$ matrix of codes by computing $\mathbf{C} = \max(0, \mathbf{D}^T\mathbf{X} - \alpha)$
6: Train a linear SVM on the pair $(\mathbf{C}, \mathbf{Y})$ (in one-vs-all fashion)

4. Experimental Results

In this experiment, we carry out a TIMIT phone classification task using the proposed paradigm. We use all data from the 462 speakers in the training set, which amounts to approximately 3 hours and about 1.12 million samples at 100 frames/second. We extract log-mel-spectrogram features with 24 filter banks and a window size of 25ms. We use a context of 5 frames on each side, with a total of 11 frames of context with 24 real numbers each. We also add delta and delta-delta as with many other standard pipelines, and we found that this did not significantly change the reported results.

We report the results on the standard TIMIT Core Test Set consisting of 192 utterances, using the standard development set to tune the only parameter in our Algorithm — the regularization coefficient in the SVM.

In our setup, we use a single machine. Given the size of the training samples, and that we did not do any splitting of the training set for simplicity, when using large values of k, we downsampled the training set by 3. This is obviously not optimal, and better results could probably be obtained by using multiple machines or sequentially loading the data instead of processing it all at once. However, since we release our code, we wanted to keep it as simple and conceptual as possible. As a consequence, when reporting numbers with 1600 or 2000 codes, all the training set was used, but when using 6000, only 1/3 was used in the SVM training phase. The whole training process took a little more than an hour to complete, with 95% of time spent in the SVM training. Since this step is trivial to parallelize, we expect our method to be very attractive when scaling up the training size by orders of magnitude.

The SVM is trained to predict one of the 61 phone states (making the number of total classes equal to 183). Besides reporting per frame accuracies, we take the naive approach of converting each frame to its predicted value from the multi-class SVM, and then using dynamic programming to take into account the dynamics at the utterance level. An even better approach would be to convert the SVM decisions to probabilities (typically done through logistic regression), and then use an HMM with a phonetic language model to decode.

Code. size	Tr. Size	Tr. Frame Acc.	Tst. Frame Acc.
1600	Full	56.1%	50.1%
2000	Full	61.3%	50.6%
6000	1/3	68.2%	51.1%
SVM	Full	40.4%	39.7%
1-DCN[4]	Full	72.8%	50.2%
7-DCN[4]	Full	98%	55.3%

Table 1: Results on the TIMIT dataset comparing different dictionary sizes of sparse coding and a few baseline methods.

Method	Tr. Frame Acc.	Tst. Frame Acc.
OMP1	58.7%	50.1%
+ ZCA	71.3%	49.6%
+ contrast	59.5%	50.2%
+ ZCA + contrast	68.2%	51.1%
Random	69.4%	49.1%

Table 2: Results on the TIMIT dataset with codebook size of 6000, comparing different normalizations and coding strategies.

Note that our approach falls in the category of "shallow" learning, and the most time consuming step involves solving a convex, parallelizable problem. At test time, two simple matrix-vector multiplications need to be carried in order to convert from the spectrogram to the likelihoods that would be used for HMM decoding, making our approach compatible with LVCSR tasks.

Table 1 shows the effect of the codebook size (value of k) on the per frame phone state accuracy, and compares the results with applying SVM directly to the raw features (SVM), and the Deep Convex Network (DCN) proposed in [2] in its shallow (1-DCN) and deep versions (7-DCN). We note that, in contrast to [2] work, sparse coding does not outperform deep learning on TIMIT, which may indicate that speech acoustic modeling benefits more from deep architectures.

4.1. Learning discussion

As described in Algorithm 1, there are certain normalizations that can be done (Lines 2and 3), and the dictionary learning step which can be switched by a more trivial approach (Line 4). In Table 2 we can see the effect of each of these optional steps. Indeed, having ZCA whitening plus contrast normalization helps in terms of classification performance by 1% absolute, but it is surprising that ZCA alone seems to not help. Also, using a dictionary based on randomly selected spectrograms performs a little worse than the dictionary found by OMP1 (but not by a large amount).

Lastly, if we perform phone recognition with dynamic programming (without phonetic language model) using the standard 39 phones used in TIMIT, we obtain 75.1% phone classification accuracy.

4.2. Dictionary Analysis

We have examined the learned dictionary elements associated with some typical phonetic categories. Since the training samples can be expressed as a linear sum of the basis functions in the dictionary, we expect the important dictionary elements contain useful acoustic patterns that code important phonetic distinctions consistent with phonetic literature.

In Figure 1 we show 16 examples of dictionary elements corresponding to each of three states of /aa/ (top), /t/ (middle),

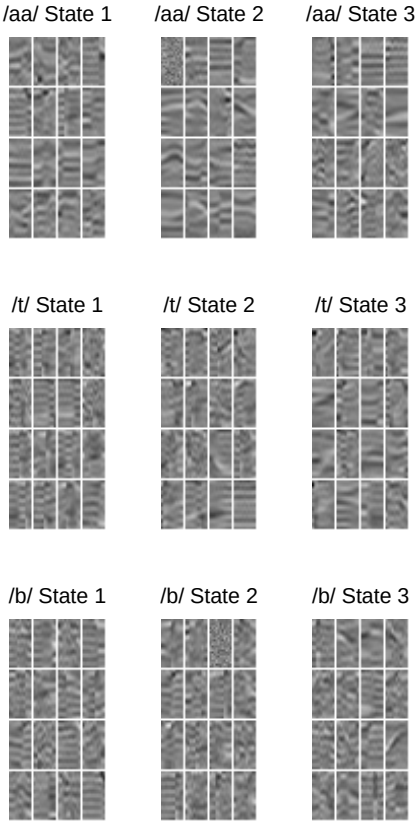

Figure 1: Most discriminant basis for some sub-phone states in TIMIT.

and /b/ (bottom). These are rows of the matrix $\mathbf{D}$ in Eq. 1, as elements of the dictionary. Each row of $\mathbf{D}$ has the dimensionality of 11*24 (the number of frames by the number of filterbanks). To aid visual inspection, each row is re-arranged to form an 11 by 24 two-dimensional image as shown in Figure 1. The 16 selected dictionary elements associated with each of state correspond to 16 largest weights determined by the SVM.

From the top panel of Figure 1, we see clear formant patterns in each of the three states. Both up and down formant transitions are seen in State 2, while in States 1 and 3, mainly one directional formant movements are represented. This is consistent with the intuition and acoustic-phonetic knowledge about formant movements [3].

In contrast, for unvoiced stop consonant /t/, we see much noisier basis functions, corresponding to burst and aspiration portions of that sound. The amount of noise in voiced stop consonant /b/ appears to be smaller, reflecting the absence of large aspiration noise. Interestingly, in State 3 of /b/, we observe clear formant transitions, which is consistent with the following property of /b/: it is relatively short; And hence in its final state and with 5 frames appended to its right the /b/ segment tends to cover the transitional sounds into its right vowel. Such transitions in the vowel are clearly visible in the basis functions associated with the final state of /b/.

5. Summary and Conclusion

In this paper we have shown how a simple approach based on dictionary learning on log mel spectrogram integrated with a straightforward coding stage performs very well on the TIMIT phone classification task. The coding stage we developed involves a projection onto the dictionary matrix followed by a non-linearity to make the features sparse. The sparse coded features are then used by a linear classifier, a linear SVM in this work, to perform classification. In particular amongst "shallow" models (those not using very deep architectures), our proposed method performs as well as state of the art, while being highly parallelizable and simple to implement. However, deep architectures still seem to perform better. In fact, making sparse coding compatible with deep learning is an active research topic in the machine learning community (see, for example, [11]). We are currently extending the method presented in this paper to generalize sparse coding with deep architectures. The preliminary results are highly promising in reducing classification errors while keeping the simplicity, parallelization capabilities, and convexity on the whole training process.

6. Acknowledgements

We would like to thank Suman Ravuri, Nelson Morgan, Kai Yu and Yuanqing Lin for useful discussion about this work.

7. References

[1] A. Castrodad and G. Sapiro. Sparse modeling of human actions from motion imagery. Preprint, 2011.

[2] A. Coates and A.Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, volume 8, page 10, 2011.

[3] L. Deng and D. O'Shaughnessy. *Speech processing: a dynamic and optimization-oriented approach*, volume 17. CRC, 2003.

[4] L. Deng and D. Yu. Deep convex network: A scalable architecture for deep learning. In *Interspeech*, 2011.

[5] D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[6] J. Mutch and D.G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, volume 1, pages 11–18. IEEE, 2006.

[7] B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[8] T. N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky. Exemplar-based sparse representation features: from timit to lvcsr. *Audio, Speech, and Language Processing, IEEE Transactions on*, (99):1–1, 2011.

[9] G. Sivaram and H. Hermansky. Multilayer perceptron with sparse hidden outputs for phoneme recognition. In *ICASSP*, pages 5336–5339. IEEE, 2011.

[10] G. Sivaram, S.K. Nemala, M. Elhilali, T.D. Tran, and H. Hermansky. Sparse coding for speech recognition. In *ICASSP*, pages 4346–4349. IEEE, 2010.

[11] D. Yu, F. Seide, G. Li, and L. Deng. Exploiting sparseness in deep neural networks for large vocabulary speech recognition. In *ICASSP*, pages 5336–5339. IEEE, 2012.

[12] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. *NIPS*, 22:2223–2231, 2009.