Progress Report
1-17 August 2012

Amin Karbasi
E-mail: amin,karbasi@epfl.ch
Amir Hesam Salavati
E-mail: hesam.salavati@epfl.ch

Supervisor: Prof. Amin Shokrollahi
E-mail: amin.shokrollahi@epfl.ch

Algorithmics Laboratory (ALGO)
Ecole Polytechnique Federale de Lausanne (EPFL)

1 Summary

In the past two weeks or so, our main focus was to address the comments provided by the reviewers of NIPS 2012. To this end, we continued simulating various approaches to memorize features from a dataset of spoken English words. Additionally, we studied a couple of papers mentioned by the reviewers, which turned out to be very helpful. A brief summary for two of these papers is provided below.

2 Summary of the paper ”extracting and composing robust features with denoising autoencoders”

This turned out to be a fantastic paper in terms of relevance to our work, i.e. memorizing a set of patterns and dealing with noise in the recall phase.

In [1] the author propose a novel approach to design a learning algorithm which extracts features from patterns in a database and is robust to partial corruption among patterns at the same time. In order to ensure this robustness, the authors *manually* introduce some noise during the learning. Then, an objective function is minimized which is related to the reconstruction error for reconstructing the *original* pattern from the set of found features. In this way, after repeating the learning several times (possibly with different corruptions), the network eventually learns the correct mapping. This become more interesting when considered from the viewpoint of learning a **manifold**, i.e. the case when the input patterns lie on a manifold. Instead of designing an algorithm which learns the specific rules that form the manifold (like our paper [2]), the method proposed here learns the manifold via trial and error. The only downside is probably elongated learning phase. But could be beneficial when one does not know which type of rule is present among input patterns.

More specifically, let $\mathbf{x} \in [0,1]^d$ be an input pattern. An *autoencoder* extracts the features of this input and maps it to another vector $\mathbf{y} \in [0,1]^{d'}$. The extraction is done through a deterministic mapping $y = f_\theta(x) = s(W\mathbf{x}+\mathbf{b})$, where $s(.)$ is a nonlinear function, W is the weighted connectivity matrix and $\mathbf{b}$ is the bias. $\theta = \{W, \mathbf{b}\}$ specifies the system parameters. The quality of extraction is determined by trying to reconstruct the pattern $\mathbf{z}$ from the features $\mathbf{y}$ to estimate $\mathbf{x}$, i.e. $\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(W'\mathbf{y} + \mathbf{b'})$, where $\theta' = \{W', \mathbf{b'}\}$ is the set of parameters for reconstruction system.

The major goal is usually to minimize the reconstruction error between $\mathbf{z}$ and $\mathbf{x}$. However, the additional goal in this paper is to make the whole process robust to noise as well, i.e. if we feed the network some corrupted version of $\mathbf{x}$, denoted by $\tilde{\mathbf{x}}$, the network should be able to find the correct $\mathbf{z}$ that estimates $\mathbf{x}$.

The beautiful idea behind the solution is to **manually** introduce noise during the learning phase. More specifically, during the learning phase, when we are given a pattern $\mathbf{x}$ to learn, the algorithm randomly introduce some corruptions in $\mathbf{x}$ to generate another pattern $\tilde{\mathbf{x}}$. This pattern is then used to learn $\theta = \{W, \mathbf{b}\}$ and $\theta' = \{W', \mathbf{b'}\}$. However, learning is done in such a way that the reconstruction error between $\mathbf{z}$ and $\mathbf{x}$ is minimized (instead of $\tilde{\mathbf{x}}$). This way, after seeing several instances of a corrupted pattern and *knowing* that these corrupted patterns should be mapped to a specified correct version, the network learns the correct mapping.

As mentioned above, this approach is closely related to our objective, i.e. to retrieve neural patterns from partial information. Furthermore, the proposed method is very similar to our line of work to increase the pattern retrieval capacity in the sense that since it is designed for classi-

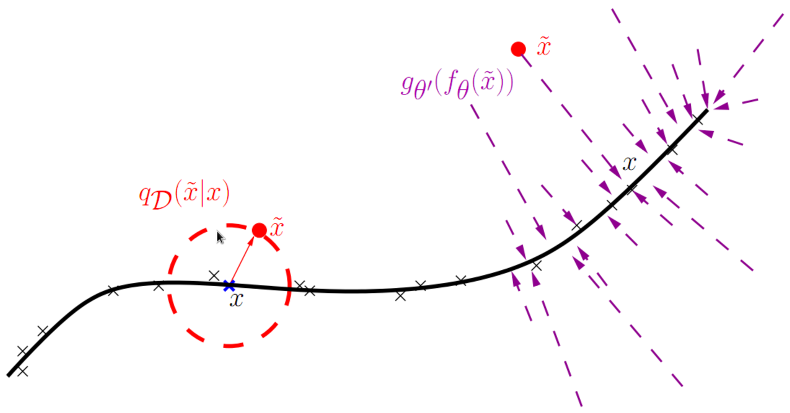
$q_{\mathcal{D}}(\tilde{x}|x)$
$\tilde{x}$
x
$g_{\theta'}(f_\theta(\tilde{x}))$
$\tilde{x}$
x

Figure 2. **Manifold learning perspective.** Suppose training data ($\times$) concentrate near a low-dimensional manifold. Corrupted examples ($\bullet$) obtained by applying corruption process $q_{\mathcal{D}}(\widetilde{X}|X)$ will lie farther from the manifold. The model learns with $p(X|\widetilde{X})$ to "project them back" onto the manifold. Intermediate representation Y can be interpreted as a coordinate system for points on the manifold.

Figure 1

fication, it tends to extract structures from the patterns in a class which can then be used in the classification/retrieval phase. In contrast to other classification methods, the approach mentioned in this paper is robust to noise, thanks to its special design. Finally, since there is no particular assumption about a pre-specified structure among patterns, the method is quite general and could potentially learn ***any*** structure. This is in sharp contrast to our work where we assume patterns are coming from subspaces. The main disadvantage is probably lengthy learning phase. Because the approach should be exposed to many corrupted instances of a pattern to be able to learn the correct mapping.

2.1 Manifold learning perspective

When the input patterns form a manifold, the proposed approach can be defined as a way to learn this manifold (see Figure 1). The intermediate representation $Y = f(X)$ can be then thought of as a **coordinate** for the points on the manifold.

More detailed comments about this paper is available in our notes in [3].

3 Summary fo the paper "the importance of encoding versus training with sparse coding and vector quantization"

In [4] the authors investigate the reason for success of sparse coding methods in feature extraction and classification methods. To this end, the authors decouple the training and encoding (feature extraction from new patterns) phases. For each phase, the authors use different algorithms, including sparse coding. Then, the authors compare the performance of the various combination of training-encoding algorithms to see if the particular choice of sparse coding for both training and encoding is the main reason behind its success. Surprisingly, the authors find out that it is almost not important what algorithm is used for the training phase. But in the encoding phase, one usually gets better results with the sparse coding technique. Therefore, the main reason behind the success of sparse coding approaches is in the encoding phase. As a result, it is much more efficient to use a fast training approach, such as *vector quantization techniques* (or even more interestingly, *random patches of data*), and spend the resources on a good encoder which uses sparse coding.

More specifically, let $x^{(i)} \in \mathbb{R}^n$ denote the set of input vectors (corresponding to either the batches or SIFT descriptors). Then, the problem of interest is to learn a matrix of d features $D \in \mathbb{R}^{n\times d}$, in which each column $D^{(j)}$ correspond to a feature. For each pattern $x^{(i)}$ in the training set, one also learns the set of features $s^{(i)}$ such that $Ds^{(i)} \simeq x^{(i)}$. In the retrieval phase, D is fixed an we are looking for the best representation for the new input. Various algorithms are considered for each phase and different combination of training-encoding approaches are compared in practice to decide which combination is better.

3.1 Algorithms for feature learning phase

Some of the considered feature learning approaches are:

- **Sparce Coding** (SC): the Sparse Coding <u>learning</u> problem can be mathematically formulated

$$\min_{D,s^{(i)}} \sum_i \|Ds^{(i)} - x^{(i)}\|_2^2 + \lambda \|s^{(i)}\|_1 \tag{1a}$$

  subject to

$$\|D^{(j)}\|_2 = 1, \forall j \tag{1b}$$

  - The authors use the ***coordinate descent algorithm*** to solve the above problem [5].

- **Orthogonal Matching Pursuit** (OMP): OMP <u>learning</u> problem is

$$\min_{D,s^{(i)}} \sum_i \|Ds^{(i)} - x^{(i)}\|_2^2 \tag{2a}$$

  subject to

$$\|D^{(j)}\|_2 = 1, \forall j \tag{2b}$$

  and

$$\|s^{(i)}\|_0 \leq k, \ \forall i \tag{2c}$$

  - The above problem is solved using the OMP algorithm [6].

- **Sparse RBMs**: details can be found in [7].
- **Sparse Auto-encoders**: details of this algorithm can be found in [8].
- **Randomly Sampled Patches** (RP): this is a heuristic approach in which the column of D are filled with randomly chosen $x^{(i)}$'s (after normalization).
- **Random Weights** in this approach, the weights are completely random!!! It has been shown in the past that this approach can perform surprisingly well.

3.2 Algorithms for retrieval (encoding) phase

Some of the considered feature learning approaches are:

- **Sparce Coding** (SC): to obtain the set of features, we should solve the problem (1) but with D fixed. Note than the choice of λ for this phase can be different from the learning phase.
- after finding s using the above method, the features vector f is determined by

$$f_j = \max\{0, s_j\} \tag{3a}$$

$$f_{j+d} = \max\{0, -s_j\} \tag{3b}$$

  - So instead of having a feature vector with d entries, we will get one with $2d$ entries.
  - One disadvantage of this approach with respect to our approach is its complexity since one has to solve an optimization problem for each new input x.

- **Orthogonal Matching Pursuit** (OMP-k): similar to the above approach, one solves the problem (2) with D fixed. Given the solution s, the features vector f is determined using equation (3).
  - One disadvantage of this approach with respect to our approach is its complexity since one has to solve an optimization problem for each new input x.

- **Soft threshold**: a simple feed forward network is used in which

$$f_j = \max\{0, D^{(j)T}x - \alpha\} \tag{4a}$$

$$f_{j+d} = \max\{0, -D^{(j)T}x - \alpha\} \tag{4b}$$

  where α is a fixed threshold.
  - This approach is very similar to our approach in [2] in terms of complexity, albeit this one is simpler since only one iteration is done.

- **Natural Encoding**: this refers to the setup in which the same algorithm is used both for learning and encoding phases.

3.3 Results and Conculsions

The authors show that the choice of learning algorithm is not important when the size of training set is large. But when the number of training patterns is small, learning with Sparse Coding yield better results. Therefore, when we have a large number of training patterns, we can use almost any fast training algorithm and there is no need to spend resources with sparse coding in the training phase. More interestingly, the authors have found out that a simple learning algorithm which selects *random* patches from the training set as the components of the feature extraction matrix can be combined with sparse coding in the encoding phase to achieve a performance really really close to the state of the art benchmarks.

Since many of the mentioned approaches work with *sparse signals*, they could be useful in our future works. For instance, many approaches try to extract features from the signal for classification purposes later. But, they require these feature sets to be sparse, i.e. representing an input with as few features as possible. This can be helpful for our case to learn the relation between these features for denoinsing purposes, as in [2]. The sparsity of the signals may help us reduce the learning time.

More detailed comments are available in our notes for this paper [9].

References

[1] P. Vincent, H. Larochelle, Y. Bengio, P. A. Manzagol, *Extracting and composing robust features with denoising autoencoders*, Proc. International Conference on Machine Learning, 2008, pp. 1096-1103

[2] K.R. Kumar, A.H. Salavati and A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary associative memory*, Proc. IEEE Information Theory Workshop, 2011.

[3] A. H. Salavati, *Notes on the paper "extracting and composing robust features with denoising autoencoders"*, August 2012, Available at `https://docs.google.com/open?id=0B2poqrkA0LCyZFNEWm5vLTVHajQ`.

[4] A. Coates, A. Y. Ng,*The importance of encoding versus training with sparse coding and vector quantization*, Proc. Int. Conf. Machine Learning, 2011.

[5] T. T., Wu, K. Lange, *Coordinate descent algorithms for lasso penalized regression.* Annals of Applied Statistics, Vol. 2, No. 1, 2008, pp. 224-244.

[6] Y. C. Pati, R. Rezaifar, P. S. Krishnaprasad, *Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition.* Proc. Asilomar Conference on Signals, Systems and Computers, Vol. 1, 1993, pp. 40-44.

[7] G. E. Hinton, S. Osindero, Y. W. Teh, *A fast learning algorithm for deep belief nets.* Neural Computation, Vol. 18, No. 7, 2006, pp. 15271554.

[8] M. Ranzato, Y. Boureau, Y. LeCun, *Sparse feature learning for deep belief networks.* Proc. Advances in Neural Information Processing Systems (NIPS), 2007.

[9] A. H. Salavati,*Notes on the paper "the importance of encoding versus training with sparse coding and vector quantization"*, August 2012, Available at `https://docs.google.com/open?id=0B2poqrkA0LCyblVwcFZ3LVYxVzQ`.