Recklessly Approximate Sparse Coding

Misha Denil
Department of Computer Science
University of British Columbia
mdenil@cs.ubc.ca

Nando de Freitas
Department of Computer Science
University of British Columbia
nando@cs.ubc.ca

Abstract

Introduction of the so called K-means features caused significant discussion in the deep learning community. Despite their simplicity, these features have achieved state of the art performance on several benchmark image classification tasks, beating out many more sophisticated learning methods. In this paper we demonstrate that a variant of these features arises as a one-step approximation to non-negative sparse coding with a fixed dictionary. This result connects these features to a broader theoretical framework and provides an explanation for their success.

1 Introduction

The publication of the so called "K-means" or "triangle" features in [1] caused significant discussion in the deep learning field. These simple features are able to achieve state of the art performance on standard image classification benchmarks, outperforming much more sophisticated methods including deep belief networks, convolutional nets, factored RBMs, mcRBMs, convolutional RBMs, sparse autoencoders and several others. Moreover, the triangle features are extremely simple and easy to compute.

Several intuitive arguments have been put forward to describe this remarkable performance, yet no mathematical justification has been offered. In [2], the authors improve on the triangle features with "soft threshold" features, adding a hyperparameter to tune performance, and compare these features to sparse coding. Both soft thresholding and sparse coding are found to often yield similar classification results, though soft threshold features are much faster to compute.

In this paper we show that the soft threshold features are realized as a single step of proximal gradient descent on a non-negative sparse coding objective. This result is important because it provides a unifying theoretical theme by connecting the soft threshold features to a well understood field. This result also accentuates the disconnection between generative encoding objectives and classification performance.

The second point arises due to the degree of approximation. Our result shows that only a single step of optimization is sufficient to give a good encodings for classification, while achieving low reconstruction error typically requires much more work. The soft threshold approach produces encodings which achieve high classification performance despite the fact that these encodings give rise to very inaccurate reconstructions. We demonstrate this by studying the behaviour of features produced by a few iterations of several different optimization methods.

2 Background

2.1 Soft threshold features

In [1] and [2], Coates and Ng found that two very simple feature encodings were able to achieve state of the art results on several image classification tasks. In [1] they consider the so called "K-means" or “triangle” features, which they define as

$$z_k^{\text{tri}}(x) = \max\{0, \mu(x) - ||x - w_k||_2\}$$

where $\{w_k\}_{k=1}^K$ is a the dictionary of elements obtained by clustering data samples with K-means, and $\mu(x)$ is the average of $||x - w_k||_2$ over k. In [2] they consider the closely related “soft threshold” features, given by

$$z_k^{\text{st}}(x) = \max\{0, w_k^{\mathrm{T}} x - \lambda\} \tag{1}$$

with λ as a parameter to be set by cross validation. These feature encodings have proved to be surprisingly effective, achieving state of the art results on popular image classification benchmarks. However, what makes these features especially appealing is their simplicity. Given a dictionary, producing an encoding requires only a single matrix multiply and threshold operation.

Similar approaches to feature encoding are well known in the computer vision literature under the name of vector quantization. In this approach, data are encoded by hard assignment to the nearest dictionary element. Van Gemert et al. [3] consider a softer version of this idea and find that using a kernel function for quantization rather than hard assignment leads to better performance. Bourdeau et al. [4] consider a similar soft quantization scheme but find that sparse coding performs better still.

Following the success of [1], the triangle and soft threshold features (or slight variants thereof) have been applied in several settings. Blum et al. [5] use triangle features for encoding in their work on applying unsupervised feature learning to RGB-D data using a dictionary designed by their own convolutional K-means approach. Knoll et al. [6] apply triangle features to image compression using PAQ. An unthresholded version of triangle features was used in [7] as the low-level image features for a system which extracts and redesigns chart images in documents.

In [8] and [9] soft threshold features are used for the detection and recognition of digits and text (respectively) in natural images. The same features have also been employed in [10] as part of the base learning model in a system for selecting the receptive fields for higher layers in a deep network.

Jia et al. [11] consider both triangle and soft threshold features for encoding image patches and investigate the effects of optimizing the spatial pooling process in order to achieve better classification accuracy.

We should emphasize that explanations for the success of triangle and soft threshold features, where offered, are always intuitive. Although these feature encodings have been observed to work well, and have become increasingly popular, no connection to a broader theoretical framework has been offered in any of the cited works.

We also note that triangle and soft threshold features are merely slight variations on the same idea. If we modify the triangle features to be defined in terms of squared distances we can write

$$z_k^{\text{tri}}(x) = 2\max\{0, w_k^{\mathrm{T}} x - \lambda(x)\}$$

which we can see is just a scaled version of soft threshold features, where the threshold is chosen as a function of x rather than by cross validation. Due to this similarity we consider only the soft threshold variant in the remainder of this paper.

2.2 Sparse coding

Sparse coding [12] considers encoding a vector x as a sparse linear combination of a dictionary $\{w_k\}_{k=1}^K$. In general solving this problem involves two stages:

1. A *learning* phase, where the dictionary $\{w_k\}_{k=1}^K$ is constructed, and
2. an *encoding* phase, where we seek a representation of a new vector x in terms of elements of the dictionary.

In this paper we focus on the encoding phase. If we collect the dictionary elements into a matrix $W = [w_1 \mid \cdots \mid w_K]$ we can formulate the encoding problem as an optimization

$$\hat{z} = \arg\min_z \frac{1}{2}||Wz - x||_2^2 + \lambda||z||_1 \tag{SC}$$

where $\lambda \geq 0$ is a regularization parameter that represents our willingness to trade reconstruction error for sparsity. The ℓ_1 norm in (SC) is used as a sparsity encouraging proxy for the ℓ_0 norm, since this relaxation makes the problem tractable.

Often it is useful to consider a non-negative version of sparse coding which leads to the same optimization as (SC) with the additional constraint that the elements of z must be non-negative. Typically non-negative sparse coding is treated as a non-negative matrix factorization problem thus also requires that the elements of W be non negative [13]. In our setting, W is a fixed parameter and is not subject to these constraints.

3 A reckless approximation

The use of λ as both the soft threshold parameter in Equation 1 and as the regularization parameter in (SC) is intentionally suggestive. The key insight in our work is to notice that Equation 1 is computing an approximate solution to the sparse coding problem (SC), by taking one step of proximal gradient descent. Proximal minimization is an efficient optimization technique, which operates by making the cost function strictly convex through the iterative addition of quadratic terms [14]. We summarize this result in the following proposition:

Proposition 1. *The soft threshold features*

$$z_k(x) = \max\{0, w_k^{\mathrm{T}} x - \lambda\}$$

are given by a single step (of size 1) of proximal gradient descent on the non-negative sparse coding objective with regularization parameter λ and known dictionary W, starting from the fully sparse solution.

Proof. Let

$$f(z) = \frac{1}{2}||Wz - x||_2^2$$

then we can write the non-negative sparse coding objective as

$$\hat{z} = \arg\min_z f(z) + \lambda||z||_1 + \Pi(z)$$

where

$$\Pi(z) = \begin{cases} 0 & \text{if } z_i \geq 0 \quad \forall i \\ \infty & \text{otherwise} \end{cases}$$

is an indicator function on the non-negative orthant. Taylor expansion of $f(z)$ about an arbitrary point z^0 gives

$$\hat{z} = \arg\min_z \Pi(z) + \lambda||z||_1 + f(z^0) + \nabla f(z^0)(z - z^0) + \frac{1}{2}(z - z^0)\nabla^2 f(z^0)(z - z^0)$$

In order to derive a proximal gradient update for this objective we make a diagonal approximation to the Hessian, $\nabla^2 f(z^0) \approx \alpha I$, where α is the step-size parameter referred to in the statement of the proposition, which we set equal to 1. This allows us to express $\hat{z}$ as the limit of the following sequence

$$z^{t+1} = \arg\min_z \Pi(z) + \lambda||z||_1 + \frac{1}{2}||z - (z^t - \nabla f(z^t))||_2^2 \tag{2}$$

where we have dropped terms which do not depend on z, since they do not affect the minimization. Since we have approximated the Hessian to derive Equation 2, it does not give a solution to (SC) directly; however, iterating the fixed point equation it defines will produce a sequence of iterates z^t which converge to $\hat{z}$ as $t \to \infty$ [15]. Equation 2 is known as the proximal gradient algorithm, since it involves evaluating the proximal operator of the regularizer $\Pi(z) + \lambda||z||_1$ applied to the gradient descent vector $u = z^t - \nabla f(z^t)$. Since at each step u is constant Equation 2 is separable, and we can write the solution for each element of z^{t+1} independently

$$z_k^{t+1} = \arg\min_{z_k \geq 0} \lambda z_k + \frac{1}{2}(z_k - u_k)^2$$

This minimization is quadratic in z_k, and therefore the optimum of the constrained problem is given by $z_k^{t+1} = \max\{0, z_k^*\}$ where $z_k^* = u_k - \lambda$ is the unconstrained optimum.

Unpacking the definition of u_k, and setting $z^0 = 0$ to start from the fully sparse solution, we have $z_k^1 = \max\{0, w_k^{\mathrm{T}} x - \lambda\}$, which is the desired result. □

Variants of the sparse coding features that appear in the literature can be obtained by slight modifications of this argument. For example, the split encoding used in [2] can be obtained by setting $W = \{w_k, -w_k\}$.

Once stated the proof of Proposition 1 is nearly immediate; however, this immediacy only appears in hindsight. In [2], soft threshold features and sparse coding features are treated as two separate and competing entities (see, for example, Figure 1 in [2]). In [16], triangle features and sparse coding are treated as two separate sparsity inducing objects.

From this proposition we can draw two important insights:

1. Proposition 1 provides a nice explanation for the success of soft threshold features for classification. Sparse coding is a well studied problem and it is widely known that the features from sparse coding models are effective for classification tasks.
2. On the other hand, Proposition 1 tells us that even very approximate solutions to the sparse coding problem are sufficient to build effective classifiers.

Even optimizers specially designed for the sparse coding problem typically take many iterations to converge to a solution with low reconstruction error, yet here we see that a single iteration of proximal gradient descent is sufficient to give features which have been shown to be highly discriminative.

These insights open up three questions:

1. Is it possible to decrease classification error by doing a few more iterations of proximal descent?
2. Does making a better approximation of the Hessian lead to more discriminative features?
3. Do different optimization methods for (SC) lead to different tradeoffs between classification accuracy and computation time?

In the remainder of this paper we will briefly introduce some popular methods for solving (SC) and answer the above questions through experimentation.

4 Some algorithms for sparse coding

4.1 FISTA

The Iterative Soft Thresholding Algorithm (ISTA) is a proximal gradient method for solving (SC), which works by iterating the following fixed point equation

$$z^{t+1} = \mathrm{soft}_{\lambda/L}(z^t - \frac{1}{L} W^{\mathrm{T}}(Wz^t - x))$$

Here λ is the regularization parameter from (SC), L is a step size parameter that must be larger than the largest eigenvalue of $W^{\mathrm{T}}W$ in order to guarantee convergence, and the soft operator,

$$\mathrm{soft}_{\lambda}(v) = \mathrm{sign}(v) \max\{0, |v| - \lambda\}$$

is applied elementwise to its argument.

Fast ISTA (FISTA) [17] is an accelerated variant of ISTA which selects a descent direction based on the previous two iterates, as in Nesterov's method, and has greatly improved convergence in practice.

A single iteration of FISTA starting at the fully sparse solution is equivalent to the soft threshold features scaled by a factor of $1/L$.

4.2 Sparse Reconstruction by Separable Approximation

Sparse Reconstruction by Separable Approximation (SpaRSA) [18] is an optimization framework that specializes to give solutions to (SC). This scheme subsumes ISTA style algorithms, but the variant we consider here includes refinements that make it different from FISTA as described above.

The first refinement is that unlike FISTA, which uses a fixed step size, the step size in SpaRSA is chosen adaptively using a Barzilai-Borwein scheme. This scheme selects the step size α_t at time t so that the diagonal matrix $\alpha_t I$ approximates the Hessian of the quadratic term in (SC).

This choice of step size leads to a sequence of iterates that do not give a monotonic decrease of the objective function. This is characteristic of Barzilai-Borwein schemes, and allowing the objective value to occasionally increase appears to be essential to achieving good performance with these methods [18]. Unlike FISTA, SpaRSA allows for occasional increases in the objective value by requiring that the objective at each step is slightly smaller than the largest objective value that has been observed in the past M iterations, where M is a fixed window size.

The Barzilai-Borwein step size scheme requires two iterates in order to select a step size, and thus does not affect the first step. One iteration of SpaRSA is equivalent to the soft threshold features.

4.3 Alternating Direction Method of Multipliers

In light of Proposition 1, it is natural to consider taking Newton steps instead of first order steps on the differentiable term in (SC). This leads to an iteration of the form

$$z^{t+1} = \mathrm{soft}_\lambda(z^t - (W^\mathrm{T} W)^+ W^T(Wz^t - x)) \tag{3}$$

where $(W^\mathrm{T}W)^+$ is the pseudo-inverse of the Hessian, and is obtained by inverting the operator $W^\mathrm{T}W$ on its range. Unfortunately, this scheme is very unstable in the presence of an overcomplete dictionary and we do not consider it further; however, next we introduce an alternative optimization method whose one-step update can be interpreted as a single smoothed Newton step.

The Method of Multipliers is an augmented Lagrangian method that is equivalent to proximal minimization, but operates in the dual space [14]. Again, a quadratic term is added to the primal cost function to make it strictly convex, which ensures differentiability of the dual. The Alternating Direction Method of Multipliers (ADMM) [14, 19] is a variant this approach suitable for separable objective functions and hence parallel computation.

For problems of the form of (SC), ADMM reduces to a sequence of soft thresholded ridge regressions. We refer the reader to Section 6.4 of [19] for a full treatment of this algorithm for sparse coding.

We can write the first step ADMM for sparse coding (starting from the fully sparse solution) as

$$z^1 = \mathrm{soft}_{\lambda/\rho}((W^\mathrm{T}W + \rho I)^{-1}W^\mathrm{T}x) \tag{4}$$

where $\rho > 0$ is a parameter of the optimizer. As long as we are only interested in taking a single step of this optimization, the matrix $(W^\mathrm{T}W + \rho I)^{-1}W^\mathrm{T}$ can be precomputed and the encoding cost for ADMM is the same as for soft threshold features. Unfortunately, if we want to perform more iterations of ADMM then we are forced to solve a new linear system in $W^\mathrm{T}W + \rho I$ at each iteration, although we can cache an appropriate factorization in order to avoid the full inversion at each step.

Comparing Equations 3 and 4 (and setting $z^t = 0$ to start with the fully sparse solution) we see the only difference is that the pseudo-inverse in Equation 3 has been replaced by a ridge term in Equation 4 in order to increase the rank of $W^\mathrm{T}W$. In ADMM, the ridge term appears from evaluating the proximity operator of the least squares term in (SC); however, we can also interpret Equation 4 as a single proximal Newton step on the Elastic Net objective [20],

$$\arg\min_z \frac{1}{2}||Wz - x||_2^2 + \rho||z||_2^2 + \frac{\lambda}{\rho}||z||_1$$

Here the ADMM parameter ρ trades off the magnitude of the ℓ_2 term, which smooths the inverse, and the ℓ_1 term, which encourages sparsity.

4.4 Boosted Lasso

Boosted Lasso (BLasso) [21] is a very different approach to solving (SC) than those considered above. Rather than solving (SC) for a specific λ, BLasso works by varying the value of λ and maintaining a corresponding solution to (SC) at each step.

As the name suggests BLasso draws on the theory of Boosting, which can be cast as a problem of functional gradient descent on the mixture parameters of an additive model composed of weak learners. In this setting the weak learners are elements of the dictionary and their mixing parameters are found in z. Similarly to the algorithms considered above, BLasso starts from the fully sparse solution but instead of applying proximal iterations, it proceeds by taking two types of steps: *forward* steps, which decrease the quadratic term in (SC) and *backward* steps which decrease the regularizer. In truth, BLasso also only gives exact solutions to in the limit as $\epsilon \to 0$; however, setting ϵ small enough can force the BLasso solutions to be arbitrarily close to exact solutions to (SC).

BLasso can be used to optimize an arbitrary convex loss function with a convex regularizer; however, in the case of sparse coding the forward and backward steps are especially simple. This simplicity means that each iteration of BLasso is much cheaper than a single iteration of the other methods we consider, although this advantage is reduced by the fact that several iterations of BLasso are required to produce reasonable encodings. Another disadvantage of BLasso is that it cannot be easily cast in a way that allows multiple encodings to be computed simultaneously.

5 Experiments

We evaluate classification performance using features obtained by approximately solving the sparse coding problem using the different optimization methods discussed in Section 4. We report the results of classifying the CIFAR-10[1] and STL-10[2] data sets using an experimental framework similar to [1]. The images in STL-10 are 96×96 pixels, but we scale them to 32×32 to match the size of CIFAR-10 in all of our experiments. For each different optimization algorithm we produce features by running different numbers of iterations for each and examine the effect on classification accuracy.

We can divide our experimental procedure into a training and testing phase, each of which we performed separately for the two data sets we consider. During the training phase we produce candidate dictionaries W for use with sparse coding. We found the following procedure to give the best performance:

1. Extract a large library of 6×6 patches from the training set. Normalize each patch by subtracting the mean and dividing by the standard deviation, and whiten the entire library using ZCA.
2. Run K-means with 1600 centroids on this library to produce a dictionary for sparse coding.

We experimented with other methods for constructing dictionaries as well, including using a dictionary built by omitting the K-means step above and using whitened patches directly. We also considered a dictionary of normalized random noise, as well as a smoothed version of random noise obtained by convolving noise features with a Guassian filter. However, we found that the dictionary created using whitened patches and K-means together gave uniformly better performance, so we report results only for this choice of dictionary. An extensive comparison of different dictionary choices appears in [2].

In the testing phase we build a representation for each image in the CIFAR-10 and STL-10 data sets using the dictionary obtained during training. We build representations for each image using patches as follows:

1. Extract 6×6 patches densely from each image and whiten using the ZCA parameters found during training.
2. Encode each whitened patch using the dictionary found during training by running one or more iterations of each of the algorithms from Section 4.

[1]`http://www.cs.toronto.edu/~kriz/cifar.html`
[2]`http://www.stanford.edu/~acoates/stl10/`

3. For each image, pool the encoded patches in a 2×2 grid, giving a representation with 6400 features for each image.
4. Train a linear classifier to predict the class label from these representations.

This procedure involves approximately solving (SC) for each patch of each image of each data set, requiring the solution to just over 4×10^7 separate sparse coding problems to encode CIFAR-10 alone, and is repeated for each number of iterations for each algorithm we consider. Since iterations of the different algorithms have different computational complexity, we compare classification accuracy against the time required to produce the encoded features rather than against number of iterations.

We perform the above procedure using features obtained with non-negative sparse coding as well as with regular sparse coding, but find that projecting the features into the positive orthant always gives better performance, so all of our reported results use features obtained in this way.

Parameter selection is performed separately for each algorithm and data set. For each algorithm we select both the algorithm specific parameters (ρ for ADMM and ϵ for BLasso) as well as λ in (SC) in order to maximize classification accuracy using features obtained from a single step of optimization.[3]

The results of these experiments are summarized in Figure 1. Several things are evident from these plots. First, running few iterations of BLasso leads to features which give relatively poor performance. Although BLasso is able to find exact solutions to (SC), running this algorithm for a limited number of iterations means that the regularization is very strong. We can also see that for small numbers of iterations the performance of features obtained with FISTA, ADMM and SpaRSA is nearly indistinguishable. The right table in Figure 2 shows the best test accuracy achieved by each algorithm during our experiments.

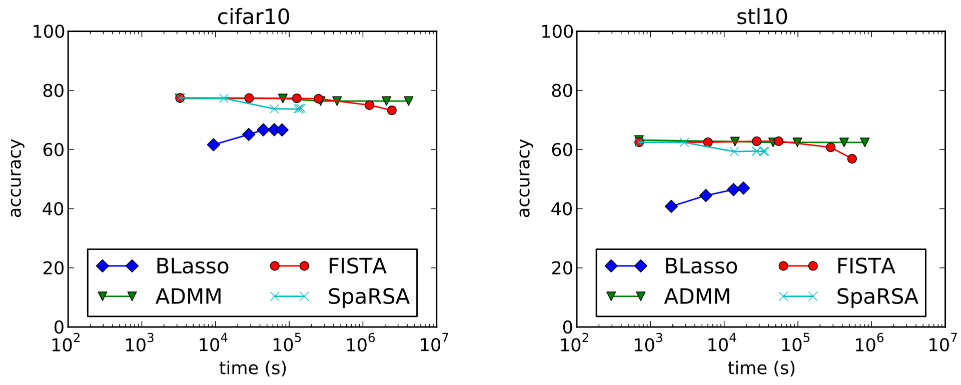


Figure 1: Classification performance versus computation time on CIFAR-10 and STL-10 using different sparse coding algorithms. For FISTA, ADMM and SpaRSA the markers show performance measured with a budget of 1, 5, 10, 50, 100 iterations. The left-most marker on each of these lines shows performance using an implementation designed to perform exactly one step of optimization. The line for BLasso shows performance measured with a budget of 10, 50, 200, 500 iterations. In all cases early stopping is allowed if a termination criterion has been met (which causes some markers to overlap).

The most notable feature of Figure 1 is that for large numbers of iterations the performance of FISTA and SpaRSA actually drops below what we see with a single iteration, which at first blush seems obviously wrong. It is important to interpret the implications of these plots carefully. In these figures all parameters were chosen to optimize classification performance for one-step features. There is no particular reason one should expect the same parameters to also lead to optimal performance after many iterations. We have found that the parameters one obtains when optimizing for one-step

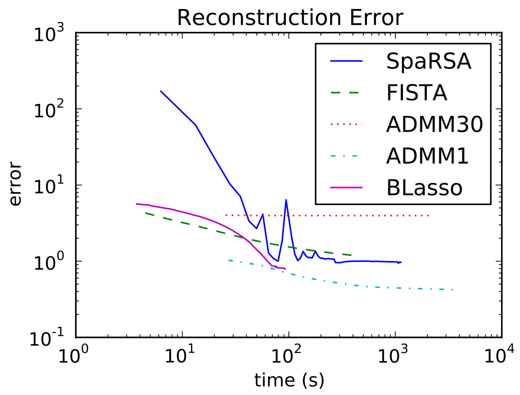
[3]For BLasso we optimized classification after 10 steps instead of 1 step, since 1 step of BLasso produced features too sparse to have any signal.

classification performance are generally not the same as the parameters one gets by optimizing for performance after the optimization has converged.

This observation is consistent with the experiments in [2] which found different optimal values for the sparse coding regularizer and the threshold parameter in the soft threshold features in their comparisons. It should also be stated that as reported in [2], the performance of soft threshold features and sparse coding is often not significantly different. We refer the reader to the above cited work for a discussion of the factors governing these differences.

To emphasize the different characteristics of reconstruction error and classification accuracy, we conduct another experiment where we record mean reconstruction error after each iteration while encoding a small sample of patches from CIFAR-10. The results of this experiment are shown in Figure 2. Comparing these results to Figure 1, we see that there is surprisingly little correlation between the reconstruction error and classification performance. In Figure 1 we saw one-step features give the best classification performance, here we see that these features also lead to the worst reconstruction.

Another interesting feature of this experiment is that the parameters we found to give the best features for ADMM actually lead to an optimizer which makes no progress in reconstruction beyond the first iteration. To confirm that this is not merely an artifact of our implementation we have also included the reconstruction error from ADMM run with an alternative setting for ρ which gives the lowest reconstruction error of any of our tested methods, while producing inferior performance in classification.



CIFAR10	Accuracy
BLasso	66.7
FISTA	**77.5**
ADMM	77.3
SpaRSA	**77.5**

STL10	Accuracy
BLasso	47.0
FISTA	62.8
ADMM	**63.2**
SpaRSA	62.5

Figure 2: **Left:** Mean reconstruction error versus computation time on a small sample of patches from CIFAR-10. FISTA, ADMM and SpaRSA were run for 100 iterations each, while BLasso was run for 500 iterations. ADMM30 corresponds to ADMM run with parameters which gave the best one-step classification performance. ADMM1 was run with a different ρ parameter which leads to better reconstruction but worse classification. **Right:** The best test accuracy achieved using features obtained by the different algorithms on each data set.

6 Conclusion

In this paper we have shown that soft threshold features, which have enjoyed much success recently, arise as single step of proximal gradient descent on a non-negative sparse coding objective. This result serves to situate these surprisingly successful features in a broader theoretical framework.

This result is also surprising, since one would not expect a single step of optimization to produce features which allow good reconstruction of the data. Our experiments have confirmed this intuition and demonstrated that, when using features obtained by sparse coding, there is in fact surprisingly little correlation between reconstruction accuracy and classifier performance.

We believe that future work in this area should focus on understanding the effects of different regularizers on features produced in this way. The addition of an indicator function to the regularizer in (SC) appears essential to good performance and proximal methods, which which appear very effective in this setting, work by adding a quadratic smoothing terms to the objective function at each step. The connection between one-step ADMM and the Elastic Net is also notable in this regard.

References

[1] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[2] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, 2011.

[3] J. van Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. Kernel codebooks for scene categorization. *Computer Vision–ECCV 2008*, pages 696–709, 2008.

[4] Y-L. Boureau, F. Bach, LeCun Y., and J. Ponce. Learning mid-level features for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[5] M. Blum, J. Springenberg, Wülfing J., and M. Riedmiller. On the applicability of unsupervised feature learning for object recognition in RGB-D data. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[6] B. Knoll and N. de Freitas. A machine learning perspective on predictive coding with PAQ8. In *Data Compression Conference*, pages 377–386, 2012.

[7] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. ReVision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th anual Symposium on User Interface Software and Technology*, 2011.

[8] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[9] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, and A. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Proceedings of the 11th International Conference on Document Analysis and Recognition*, pages 440–445. IEEE, 2011.

[10] A. Coates and A. Ng. Selecting receptive fields in deep networks. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2528–2536. 2011.

[11] T. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition*, 2012.

[12] B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[13] P. Hoyer. Non-negative sparse coding. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[14] D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[15] D. Bertsekas. *Nonlinear Programming: Second Edition*. Athena Scientific, 1995.

[16] J. Ngiam, P. Koh, Z. Chen, S. Bhaskar, and A. Ng. Sparse filtering. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1125–1133. 2011.

[17] A. Beck and M. Teboulle. A fast iterative shrinkage-threshold algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, pages 183–202, 2009.

[18] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[20] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320, 2005.

[21] P. Zhao and B. Yu. Boosted lasso. *Journal of Machine Learing Research*, 8(Dec):2701–2726, 2007.

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.