EE 364B Convex Optimization

An ADMM Solution to the Sparse Coding Problem

Sonia Bhaskar, Will Zou

Final Project
Spring 2011

I. Introduction

For our project, we apply the method of the alternating direction of multipliers and sequential convex optimization to sparse coding of images. The motivation behind sparse coding of images is to model how the brain is able to efficiently utilize the human visual system for a variety of tasks, such as separating a car from a background, as well as general classification tasks. Sparse coding aims to determine a generalized set of overcomplete bases Φ to represent any natural image, where we desire that each image can be represented by a linear combination of a few of these bases, i.e. represented by a sparse vector of coefficients α. When learned on natural images, the sparse coding bases are localized edge detectors in space, orientation and frequency.

Feature learning on natural images with sparse coding can be challenging because there can arbitrarily large input dimensions, dependent on how big the visual field for the algorithm is. Further, in many situations, an over-complete set of bases is desired [1], which means that the number of features to learn exceeds the number of input dimensions. Both aspects demands faster algorithms to make the sparse coding problem feasible, either on a single computer or on multiple in a parallel computing setting.

Among models for natural images, sparse coding is not a convex objective but is convex when fixing either Φ or α. This formulation makes it slow to train especially when the input dimensions are high. We wish to consider a distributed solution using ADMM, which we believe will speed up the algorithm. Additionally we wish to try out a heuristic in order to be able to use the desired optimization methods, which we will soon discuss.

II. Problem Description

First, we present the problem formulation for sparse coding below. The optimization problem, for $x^{(i)} \in \mathcal{R}^n$, $\Phi \in \mathcal{R}^{n\times k}$, and $\alpha^{(i)} \in \mathcal{R}^k$, where i refers to the i^{th} training example, is

$$\underset{\Phi,\ \alpha^{(i)}\forall i}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|x^{(i)} - \Phi\alpha^{(i)}\|_2^2 + \lambda\|\alpha^{(i)}\|_1\right) \quad \text{subject to} \quad \|\Phi_j\|_2^2 \le 1,\ j = 1,\dots,k. \tag{1}$$

where Φ_j refers to the columns of Φ, $\alpha_j^{(i)}$ refers to the j^{th} entry of $\alpha^{(i)}$, $\|\Phi_j\|_2^2 \le 1\forall j$ is called the non-degeneracy constraint, and where we define the objective function f as given in Eqn. (2):

$$\frac{1}{m}\sum_{i=1}^{m} f(x^{(i)},\alpha^{(i)},\Phi) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|x^{(i)} - \Phi\alpha^{(i)}\|_2^2 + \lambda\|\alpha^{(i)}\|_1\right) \tag{2}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|x^{(i)} - \sum_{j=1}^{n}\alpha_j^{(i)}\Phi_j\|_2^2 + \lambda\|\alpha^{(i)}\|_1\right) \tag{3}$$

Thus this problem answers the question “Given an input $x^{(i)}$, what is the optimal vector of coefficients $\alpha^{(i)}$ that will give us the best linear reconstruction of the input using the columns of a matrix of the optimal bases, Φ?”

Looking at both of these cases, we see that the objective function is not convex if we are solving for both Φ and $\alpha^{(i)}\forall i$. However, if we only solve for either the bases Φ or the coefficients $\alpha^{(i)}$ while holding the other constant, we have a convex problem with convex constraints. Sequential convex programming can be used here, but it is often slow. So we now explore some solutions to this issue.

III. Sequential Convex Programming

We see that for Eqn. (2), this is not a convex objective, although the constraint on the bases is convex. Often sequential convex programming is used to solve this problem, that is, the problem is separated into first solving for the $\alpha^{(i)}$'s, which is then a least-squares problem regularized by an L1-norm (sparsity), commonly known as the lasso problem, and then solving for Φ, which is a constrained least-squares problem.

Thus the convex problem would need to be solved sequentially, since when we fix Φ and solve for the $\alpha^{(i)}$'s it is convex, and when we fix the $\alpha^{(i)}$'s and then solve for Φ it is convex. The sequential program would go as follows. First, we would optimize for the $\alpha^{(i)}$'s:

$$\underset{\alpha^{(i)}\forall i}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|x^{(i)}-\Phi\alpha^{(i)}\|_2^2+\lambda\|\alpha^{(i)}\|_1\right)$$

Then we would solve for the bases:

$$\begin{aligned}&\underset{\Phi}{\text{minimize}} && \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\|x^{(i)}-\Phi\alpha^{(i)}\|_2^2\\ &\text{subject to} && \|\Phi_j\|_2^2\leq 1,\ j=1,\ldots,n.\end{aligned}$$

As mentioned in Section I, this iterative optimization is slow – we now turn to the alternating direction method of multipliers (ADMM) in order to solve it. Previous methods have used dual decomposition, but ADMM is a more robust method with faster convergence which we hope to exploit for speed.

IV. Alternating Direction Method of Multipliers

We can formulate the convex optimizaton problems into their ADMM form, resulting in what we believe will be a much faster solution.

A. *Solving for the coefficients*

Solving for the $\alpha^{(i)}$'s, we have

$$\begin{aligned}&\underset{\alpha^{(i)}}{\text{minimize}} && \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|x^{(i)}-\Phi\alpha^{(i)}\|_2^2+\lambda\|\hat{\alpha}^{(i)}\|_1\right)\\ &\text{subject to} && \alpha^{(i)}=\hat{\alpha}^{(i)},\ i=1,\ldots,m.\end{aligned}$$

The ADMM algorithm updates would run as follows - note that we have a set of updates for each of the m training examples. We can do them in parallel we can do them simultaneously in matrix format (for our implementation we chose the latter):

$$\begin{aligned}\alpha^{(i),\ k+1} &:= (\Phi^T\Phi+\rho I)^{-1}(\Phi^T x^{(i)}+\rho\hat{\alpha}^k-y^k)\\ \hat{\alpha}^{(i),\ k+1} &:= \max\left(\alpha^{(i),\ k+1}+\frac{1}{\rho}y^k-\frac{\lambda}{\rho},0\right)-\max\left(-\alpha^{(i),\ k+1}-\frac{1}{\rho}y^k-\frac{\lambda}{\rho},0\right)\\ y^{(i),\ k+1} &:= y^{(i),\ k}+\rho(\alpha^{(i),\ k+1}-\hat{\alpha}^{(i),\ k+1})\end{aligned}$$

B. *Solving for the bases*

Optimizing for the bases Φ, where each basis is a column Φ_j, the problem is cast as in Section III. Since the problem is convex, we can solve it directly using CVX, however, the found empirically that solving the problem directly is time-consuming. This is because when we solve for the bases $\Phi\in\mathbf{R}^{n\times k}$, the number of variables is nk. An alternative solution is given in [2]. Instead for solving nk variables, we can form the dual problem and solve with only n variables. We implemented this method and report results in the results section.

Further, we can use the ADMM form for the constrained convex problem. The updates are, where $U^k \in \mathcal{R}^{n\times k}$:

$$\begin{aligned}
\Phi^{k+1} &:= \arg\min_{\Phi} \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\|x^{(i)} - \Phi\alpha^{(i)}\|_2^2 + \frac{\rho}{2}\|\Phi - \hat{\Phi}^k + U^k\|_2^2 \\
\hat{\Phi}^{k+1} &:= \Pi_{\mathcal{C}}(\Phi^{k+1} + U^k) \\
U^{k+1} &:= U^k + \Phi^{k+1} - \hat{\Phi}^{k+1}
\end{aligned}$$

where the projection $\Pi_{\mathcal{C}}$ is applied to each column of the matrix individually as

$$\Pi_{\mathcal{C}}(X) = \Pi_{\mathcal{C}}(X_j)\forall j, \quad \Pi_{\mathcal{C}}(X_j) = \begin{cases} \frac{X_j}{\|X_j\|^2} & \text{for } \|X_j\|^2 > 1 \\ X_j & \text{for } \|X_j\|^2 \le 1 \end{cases} \tag{4}$$

and X_j is the jth column of X. Unfortunately this is very slow to solve for the first step in the iteration, and CVX cannot handle problems where we have chosen the number of bases to be very large and thus we did not experience much success because of the amount of time we had to wait for each iteration. So next we present a heuristic solution which is much faster.

V. Alternative Solution to Non-degeneracy Constraint

In order to employ an ADMM solution to solving for the bases where we are considering the columns as bases, we recast the minimization problem using a heuristic. We recast the problem given in Eqn. (1). We replace the non-degeneracy constraint and regularize the least-squares problem with a Frobenius norm:

$$\underset{\Phi}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\|x^{(i)} - \Phi\alpha^{(i)}\|_2^2 + \beta\|\Phi\|_F^2$$

This objective function is a convex function with a nice solution. First taking the gradient with respect to Φ we find that

$$\begin{aligned}
&\nabla_\Phi\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\|x^{(i)} - \Phi\alpha^{(i)}\|_2^2 + \beta\|\Phi\|_F^2\right) \\
&\quad = \nabla_\Phi\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(x^{(i)T}x^{(i)} - 2x^{(i)T}\Phi\alpha^{(i)} + \alpha^{(i)T}\Phi^T\Phi\alpha^{(i)}) + \beta\|\Phi\|_F^2\right) \\
&\quad = \frac{1}{m}\sum_{i=1}^{m}\left(-x^{(i)}\alpha^{(i)T} + \Phi(\alpha^{(i)}\alpha^{(i)T})\right) + 2\beta\Phi \\
&\quad = \Phi\left(\frac{1}{m}\sum_{i=1}^{m}\alpha^{(i)}\alpha^{(i)T} + 2\beta I\right) - \frac{1}{m}\sum_{i=1}^{m}x^{(i)}\alpha^{(i)T}
\end{aligned}$$

Solving for the global minimum, we find

$$\Phi^* = \left(\sum_{i=1}^{m}x^{(i)}\alpha^{(i)T}\right)\left(\sum_{i=1}^{m}\alpha^{(i)}\alpha^{(i)T} + 2m\beta I\right)^{-1} \tag{5}$$

This heuristic is approximately equivalent to the original problem and gives “relevant” solutions. Using this heuristic is advantageous because this convex optimization problem is a convex quadratic that has an analytical solution and the inverse is over a $k \times k$ matrix, which is not terribly large.

VI. Results

To illustrate the practicality of our methods, experiments are performed by learning a sparse coding dictionary from natural image patches. Compared to other feature learning algorithms for images, such as Independent Component Analysis and its variants [3], sparse coding on natural images is much more computationally expensive. At the same time, multiple papers have shown that sparse coding yields better results in tasks such as object recognition and detection. We show that our propsed methods are able to significantly speed up the sparse coding algorithm and our implementation is a useful tool for computer vision researchers.

A. CPU implementation

In our experiments, 14x14 image patches were extracted from the same image dataset as provided in the original work on sparse coding [1]. We run sequential convex optimization for 100 iterations to learn the coefficients and 196 bases. To improve the speed of convergence, the data was split into minibatches of 1000 and the algorithm iteratively optimizes with respect to each minibatch. For ADMM we select $\rho = 10$ and run for 5-10 iterations in each lasso solve; we found $\rho = 10$ to give us the fastest convergence. Figure 1 shows the bases learned by ADMM lasso solve with (left) Frobenius norm regularization on bases and (right) dual solve for bases. The two methods yield very nice features (we desire edge-like features) – note that there is not a 1:1 correspondence between each block in Figure 1.

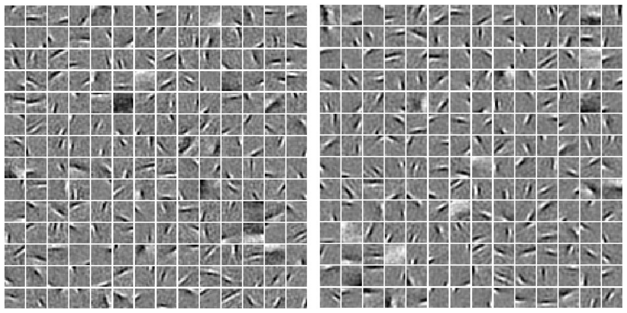


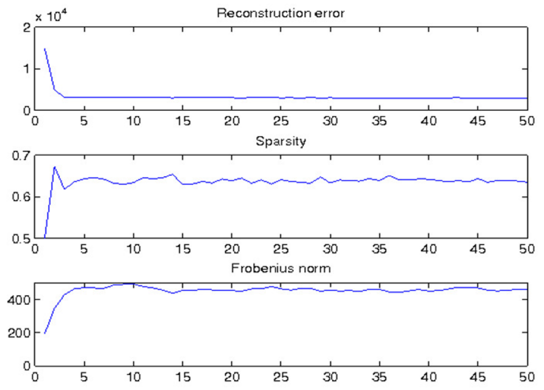
Fig. 1. Bases learned from (left) Frobenius norm regularization on the bases and (right) dual solve for bases combined with ADMM lasso solve in sequential convex optimization. The regularization parameter is selected as $\beta = 5e - 3$

For the proposed method of using Frobenius norm regularization to avoid degeneracy of the dictionary, we plot convergence, sparsity, Frobenius norm, as well as histogram of coefficients in Figure 2. The objective of the optimization converges smoothly with sequential convex optimization, a significant extent of sparsity is maintained in the coefficients, and the Frobenius norm is well controlled. We note that even though the plot shows sparsity at 65%, we look at the histogram and see that we have the desired distribution with the vast majority of coefficients being zero, and very few of them being greater than zero. The resulting coefficients form a sparse empirical distribution.

In Table I, timing results are presented for learning the dictionary (Φ) with 5 bases. Table II shows timing results for learning the dictionary with the same number of bases as the number of input dimensions (196). In both tables, evaluation is performed against popular sparse coding algorithms: the feature-sign algorithm [4] and SPAMS toolbox [5]. In Table II, 'NF' indicates not feasible in computation time or in memory.

TABLE I
Learning 5 bases: time comparison with SPAMS and feature-sign

# Algorithm	time (sec)	speedup
Feature-sign authors' code [4]	0.07	4.6e5×
SPAMS authors' code [5] (mex)	0.14	2.2e5×
CVX lasso + CVX Primal solve for bases	32105.1	base
ADMM lasso + CVX Primal solve for bases	14695.0	2.2×
ADMM lasso + Dual solve for bases	8.5	3.8e3×
ADMM lasso + Frobenius norm regularization	2.4	1.3e4×



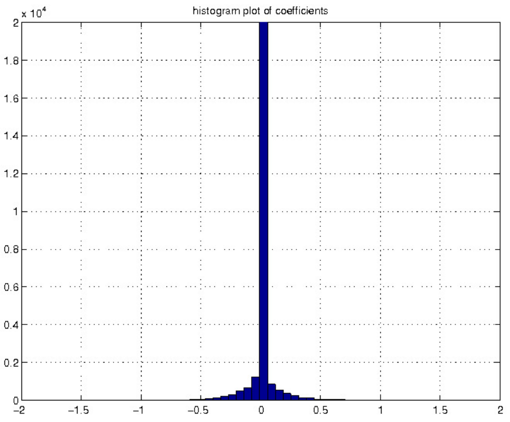


Fig. 2. Plots for the Frobenius norm regularization method combined with ADMM lasso solve. On the left we show the reconstruction cost convergence, sparsity (percentage of zeros) in coefficients, and the Frobenius norm of the dictionary matrix. On the right, we show the histogram of resulting coefficients.

TABLE II
LEARNING 196 BASES: TIME COMPARISON WITH SPAMS AND FEATURE-SIGN

# Algorithm	time (sec)	speedup
Feature-sign authors' code [4]	195.1	base
SPAMS authors' code [5] (mex)	4.9	39×
CVX lasso + CVX primal solve for bases	NF	–
ADMM lasso + CVX primal solve for bases	NF	–
ADMM lasso + Dual solve for bases	64.3	3×
ADMM lasso + Frobenius norm regularization	20.4	10×

In the lasso solve, using ADMM to quickly solve L1/L2 sub-problems significantly improves the speed of inferring coefficients. The ADMM lasso solver only needs to run a small number of iterations (5-10) for efficient learning of the dictionary. As can be seen in rows 4 and 5 in Table I and Table II, the method of solving for bases using the dual problem with many fewer number of variables is faster than solving for the primal problem by a large margin. Comparing rows 5 and 6 in both tables, our method of using Frobenius norm regularization for solving for the bases further further improves speed by a considerable factor.

From the above results, the speed-up factor of our method compared to the naive implementation is **1.3e4 times**.

The Feature-sign algorithm performs well when the number of bases is small. However, it becomes a lot slower as the number of bases increase (Table II), and in fact we use it as the baseline and our method perform 10× faster. Of all methods, the SPAMS sparse coding tool box is still fastest. We note that compared to our simple implementation, SPAMS uses a highly optimized block-coordinate descent algorithm and with mex implementation. Further, since the algorithm heavily exploits sparsity with block-coordinate descent, its performance drops when we reduce the value of λ. We compare to our algorithm with different λ settings in Figure 3.

To summarize, our method performs reasonably well with a simple implementation. Importantly, it is not affected by the extent of sparsity or number of feature dimensions. This is advantageous when we hope to learn an over-complete set of bases to represent natural images.

Further, we can think of two technical ways to speed up our algorithm: GPU and parallel computing using ADMM. We employed a GPU implementation and obtained a significant speedup.

B. GPU implementation

We implemented our algorithm with ADMM lasso solve and Frobenius norm regularied solver for the bases with Jacket (http://www.accelereyes.com/), a GPU implementation for MATLAB. The result is shown in Table III. Since our proposed method iterates between two fast steps with multiple matrix-vector multiplications, the GPU is able to significantly speed up our algorithm. More concretely, it speeds

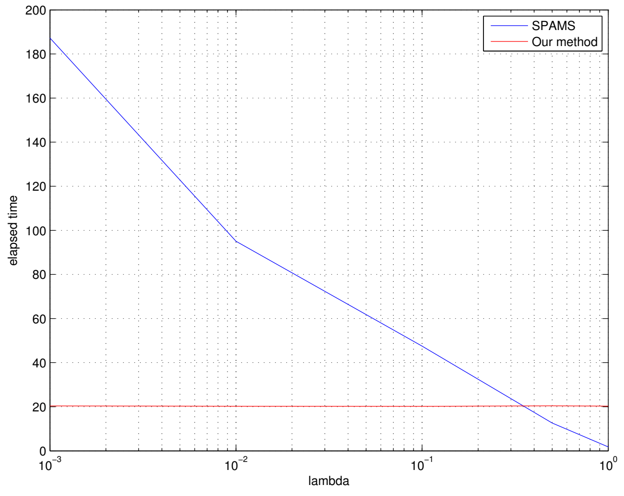


Fig. 3. Effect of sparsity setting on speed of SPAMS: since SPAMS heavily exploits sparsity, we experiment with a large range of λ values. Our algorithm is unaffected by the sparsity setting

up our algorithm by 8 times. With GPU, our method is $81\times$ faster than the Feature-sign algorithm and even $2\times$ faster than the state-of-the-art SPAMS toolbox.

TABLE III
LEARNING 196 BASES: TIME COMPARISON WITH SPAMS AND FEATURE-SIGN

# Algorithm	time (sec)	speedup
Feature-sign authors' code [4]	195.1	base
SPAMS authors' code [5] (mex)	4.9	$39\times$
GPU ADMM lasso + Frobenius norm regularization	2.4	$81.3\times$

VII. CONCLUSION

In the report, we analyzed the method of sequential convex optimization to solve the sparse coding problem applied on natural images. The ADMM lasso solver significantly speeds up the practical problems of learning the sparse coding dictionary and inferring the sparse representation of images, when the dictionary is known. Further, we experimented with fast ways of solving the second step, solving for the bases using the dual as proposed in [4], and developed a method of using the Frobenius norm regularization on the dictionary. With the regularization method combined with ADMM lasso solve, we presented a fast implmentation of sparse coding on natural images. When computation is ported on GPUs, the proposed method achieves state-of-the-art speed-up for sparse coding on natural images.

VIII. FUTURE DIRECTIONS

The methods we experimented with in this report is run on a single computer. The proposed advantage of the ADMM method extends to the capability of running multiple sub-problems on multiple computers. This is yet to be explored and we would expect a slightly less than linear-speed up with the number of computers used, when we use ADMM to parallelize data and solve for the lasso step.

Another interesting future direction is local receptive fields with sparse coding on large images. There has been recent work in artificial intelligence literature showing encouraging results using this method [6]. The idea of applying ADMM to speed-up and parallelize local sub-problems is rather simple. However, from our experiments, implementation of this idea is technically challenging.

REFERENCES

[1] B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 1996.

[2] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007.

[3] A. Hyvarinen, J. Hurri, and P. Hoyer. *Natural Image Statistics*. Springer, 2009.
[4] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007.
[5] J. Mairal, F. Bachand J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 2010.
[6] K. Gregor and Y. LeCun. Emergence of complex-like cells in a temporal product network with local receptive fields. *arXiv:1006.0448*, 2009.