Analyzing Task Driven Dictionary Learning Algorithms

AMSC 663/4 Final Report

Mike Pekala (mike.pekala@gmail.com)

Advisor: Prof. Doron Levy (dlevy@math.umd.edu)
Department of Mathematics, CSCAMM

May 10, 2012

Abstract

The underlying notion of *sparse coding* is that, in many domains, data vectors can be concisely represented as a sparse linear combination of basis elements or dictionary atoms. Recent results suggest that for many tasks (e.g. classification, denoising) performance can be improved by explicitly *learning* the sparsifying dictionary directly from the data. This is in contrast with classical approaches, such as wavelets, that use predefined dictionaries known to work well on a broad class of signals. Furthermore, results suggest that additional performance gains are possible by *jointly optimizing* the dictionary for both the data and the task of interest. In this project, we implemented the task-driven dictionary learning algorithm of [13] within the context of binary classification. We verified the correctness of our implementation through a combination of unit testing and comparison with published results on open data sets. Finally, we applied this algorithm to hyperspectral imaging data sets not considered in the original task-driven dictionary learning paper. A significant result of this work is that, for certain categories of hyperspectral classification problems, the task-driven dictionary learning approach significantly outperforms baseline algorithms such as k-nearest neighbors and logistic regression.

1 Introduction

Classical techniques for sparse coding leverage the notion that natural signals can often be represented compactly using relatively few elements from some basis (or, more generally, an *overcomplete dictionary* $\mathbf{D} \in \mathbb{R}^{m \times p}$ where $p > m$). A data vector $\mathbf{x} \in \mathbb{R}^m$ admits a sparse representation over the dictionary $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_p] \in \mathbb{R}^{m \times p}$ when there exists a good approximation for $\mathbf{x}$ consisting of a linear combination of only a few columns of $\mathbf{D}$. The sparse approximation problem of finding this representation can be written,

$$\min_{\boldsymbol{\alpha}} ||\boldsymbol{\alpha}||_0 \text{ s.t. } ||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2 < \epsilon \tag{1}$$

Due to the intractability introduced by the zero semi-norm, typically one instead optimizes the convex relaxation,

$$\min_{\boldsymbol{\alpha}} ||\boldsymbol{\alpha}||_1 \text{ s.t. } ||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2 < \epsilon \tag{2}$$

Recently, state-of-the-art results in many signal processing tasks have been achieved using dictionaries learned from test data in place of a predefined basis, such as wavelets [15, 16, 3]. Intuitively, these performance gains are possible since the dictionary elements $\mathbf{d}_i$ (also called "atoms") can be tailored specifically to the data of interest. Unlike other classical data-driven techniques, such as principal component analysis, dictionary learning approaches do not require the resulting atoms be orthogonal or form a basis. This relative freedom provides greater flexibility to adapt the representation to the data.

1.1 Dictionary Learning

Dictionary learning can be framed as the process of taking a set of input signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ and producing a dictionary $\mathbf{D} \in \mathbb{R}^{m \times p}$ by minimizing the empirical cost function,

$$g_n(\mathbf{D}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_u(\mathbf{x}_i, \mathbf{D}) \tag{3}$$

where ℓ_u, the *unsupervised* loss function, is small when $\mathbf{D}$ is "good" at representing $\mathbf{x}_i$ sparsely [13]. One possible measure of the quality of a sparse approximation is the elastic net criterion [20],

$$L(\lambda_1, \lambda_2, \boldsymbol{\alpha}, \mathbf{x}, \mathbf{D}) = \frac{1}{2}||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \lambda_1||\boldsymbol{\alpha}||_1 + \frac{\lambda_2}{2}||\boldsymbol{\alpha}||_2^2 \tag{4}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}$ with $\lambda_1, \lambda_2 \geq 0$ are regularization parameters tuned for the problem at hand. In this case the unsupervised loss becomes,

$$\ell_u(\mathbf{x}, \mathbf{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} L(\lambda_1, \lambda_2, \boldsymbol{\alpha}, \mathbf{x}, \mathbf{D}) \tag{5}$$

When $\lambda_2 = 0$, minimizing the elastic net criterion corresponds to an unconstrained version of the lasso [17].

To prevent artificially improving ℓ_u by simply scaling the columns of $\mathbf{D}$, one typically constrains the set of permissible dictionaries by restricting the two norm of each column to be less than or equal to one. This produces a convex set[1] $\mathcal{D}$ of admissible matrices,

[1]Recall that a set C is convex if, for any $x, y \in C$ and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$, it is the case that $\theta x + (1 - \theta)y \in C$. If $A, B \in \mathcal{D}$ then consider the ith column of any convex combination of A and B, i.e.

$$\mathcal{D} \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \text{ s.t. } \forall j \in \{1, \ldots, p\}, ||\mathbf{d}_j||_2 \leq 1\}$$

Note that minimizing the empirical cost g_n directly can be very expensive when the training set is large (as is often the case in dictionary learning problems). However, in practice, one is typically concerned about performance on as of yet unseen data. Thus, a more relevant metric of is the expected loss,

$$g(\mathbf{D}) \triangleq \mathbb{E}_{\mathbf{x}}\left[\ell_u(\mathbf{x}, \mathbf{D})\right] = \lim_{n \to \infty} g_n(\mathbf{D}) \quad \text{a.s.}$$

(where expectation is taken with respect to the unknown distribution of data objects $p(\mathbf{x})$). Stochastic online techniques provide effective ways to obtain stationary points of this optimization problem [14].

1.2 Task-Driven Dictionary Learning

If the sole objective is to find a sparse representation for data, then dictionary learning is sufficient. However, quite often sparse approximation is a preprocessing step for some higher-level task. In classification problems, for example, overall performance is often highly dependent upon finding good representations of the observations. In this case, sparse approximation acts as a data preprocessing step which maps data from the input space into a new feature space where the classification problem can be solved more effectively.

For classification, the signals $\mathbf{x}_i \in \mathbb{R}^m$ are observations and the goal is to identify the associated labels $y_i \in \mathcal{Y}$ taken from some discrete set. For example, in binary classification the space of labels is $\mathcal{Y} = \{-1, +1\}$. Given a fixed dictionary $\mathbf{D}$ and the sparsity criterion (4), the sparse approximation,

$$\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}) = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} L(\lambda_1, \lambda_2, \boldsymbol{\alpha}, \mathbf{x}, \mathbf{D}) \tag{6}$$

provides the features for predicting y. The (linear) classification problem is then to learn model parameters $\mathbf{W}$ by solving,

$$\min_{\mathbf{W} \in \mathcal{W}} f(\mathbf{W}) + \frac{\nu}{2}||\mathbf{W}||_F^2$$

where $\nu \in \mathbb{R}$, $\nu \geq 0$ is a regularization parameter and f measures the prediction quality,

$$f(\mathbf{W}) \triangleq \mathbb{E}_{y,\mathbf{x}}\left[\ell_s(y, \mathbf{W}, \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}))\right]$$

$(\theta A + (1-\theta)B)_i = \theta a_i + (1-\theta)b_i$. The triangle inequality and positive homogeneity properties of norms along with $\gamma \triangleq \max(||a_i||_2, ||b_i||_2)$ gives

$$||\theta a_i + (1-\theta)b_i||_2 \leq \theta||a_i||_2 + (1-\theta)||b_i||_2 \leq \theta\gamma + (1-\theta)\gamma = \gamma$$

Therefore, the convex combination is also in $\mathcal{D}$. Other classes of matrices can also be shown to form convex sets, e.g. symmetric positive definite matrices [10].

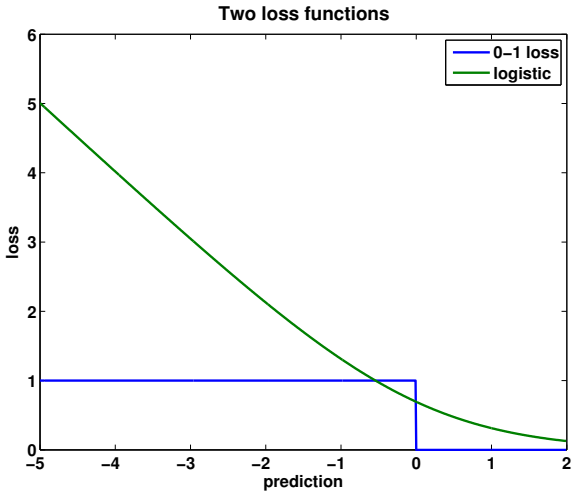


Figure 1: Comparing the 0-1 loss with the logistic loss. The 0-1 loss is not differentiable everywhere and is neither concave nor convex. The logistic loss provides a smooth, convex approximation.

Here, ℓ_s denotes a supervised loss, which is the classification analog of the unsupervised loss (5). There are many options for supervised loss functions; typically they are chosen to have properties amenable to optimization. The authors of [13] utilize the logistic loss function. Assuming the linear classification model $\mathbf{W} = \mathbf{w} \in \mathcal{W} = \mathbb{R}^p$, the prediction and logistic loss functions are,

$$m = \mathbf{w}^T \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}) \qquad \text{Prediction} \tag{7}$$

$$\ell_s = \log\left(1 + e^{-ym}\right) \qquad \text{Logistic loss} \tag{8}$$

Figure 1 depicts the logistic loss and compares it to the 0-1 loss, a function which maps correct classifications to a fixed loss of zero and incorrect classifications to a fixed loss of 1.

Task-driven dictionary learning combines the notion of a supervised task, such as classification, with the unsupervised dictionary learning problem. The result is an optimization problem where the goal is to jointly learn $\mathbf{D}$ and $\mathbf{W}$,

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W}} f(\mathbf{D}, \mathbf{W}) + \frac{\nu}{2}||\mathbf{W}||_F^2 \tag{9}$$

where, for the classification task,

$$f(\mathbf{D}, \mathbf{W}) \triangleq \mathbb{E}_{y,\mathbf{x}}\left[\ell_s(y, \mathbf{W}, \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}))\right]$$

In the case of the logistic loss, the overall optimization problem for classification task driven dictionary learning becomes,

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{w}\in\mathbb{R}^p} \mathbb{E}_{y,\mathbf{x}}\left[\log\left(1 + e^{-y\mathbf{w}^T\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})}\right)\right] + \frac{\nu}{2}||\mathbf{w}||_2^2 \tag{10}$$

The remainder of this paper describes how we implemented a task-driven dictionary learning algorithm and applied it to real-world data sets. Section 2 details the approach proposed by [13] for solving the task-driven dictionary learning optimization problem. We then describe our implementation of relevant algorithms in Section 3. Section 4 describes the databases used in this project. We then describe how the algorithms were validated and the results obtained on a new dataset in Sections 5 and 6. Finally, we conclude with a brief recap of the programatic aspects of the project (schedules, milestones and deliverables).

2 Approach

This project developed an implementation of the task-driven dictionary learning algorithm proposed by [13] for solving optimization problem (9). In particular, we focused on an implementation for the binary classification problem (10). The authors provide additional extensions and formulations beyond classification; however, implementing these extensions was outside the scope of this project.

The task driven dictionary learning algorithm is outlined in Algorithm 1. At a high level, it consists of an outer stochastic gradient descent loop that incrementally samples the training data, uses this sample and a sparse coding solver to approximate gradients with respect to the classifier model $\mathbf{w}$ and the dictionary $\mathbf{D}$ and then uses these gradients to update $\mathbf{D}$ and $\mathbf{w}$.

2.1 Stochastic Gradient Descent

The stochastic gradient descent (SGD) algorithm is an iterative, "on-line" approach for optimizing an objective function based on a sequence of approximate gradients obtained by randomly sampling from the training data set. In many respects, it is similar to the classical Robbins-Monroe procedure for finding roots of a real-valued function based on noise-corrupted measurements/observations [11]. In the simplest case, SGD estimates the objective function gradient on the basis of a single randomly selected example x_t,

$$w_{t+1} = w_t - \rho_t \nabla_w Q(x_t, w_t)$$

where Q is a loss function, w is a weight being optimized and ρ_t a step size known as the learning rate. The stochastic process $\{w_t, t = 1, \ldots\}$ depends upon the sequence of randomly selected examples x_t and thus optimizes the empirical cost (which is hoped to be a good proxy for the expected cost).

The authors of [13] show that, under suitable conditions, equation (9) is differentiable on $\mathcal{D} \times \mathcal{W}$, and that

$$\begin{aligned}
\nabla_{\mathbf{W}} f(\mathbf{D}, \mathbf{W}) &= \mathbb{E}_{y,\mathbf{x}} \left[\nabla_{\mathbf{W}} \ell_s(y_t, \mathbf{w}, \boldsymbol{\alpha}^\star) \right] \\
\nabla_{\mathbf{D}} f(\mathbf{D}, \mathbf{W}) &= \mathbb{E}_{y,\mathbf{x}} \left[-\mathbf{D}\boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star T} + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star T} \right]
\end{aligned}$$

where $\boldsymbol{\beta}^\star \in \mathbb{R}^p$ is defined by the properties:

Algorithm 1 Task-driven dictionary learning [13]

Require: Inputs: $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (number of iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

1: **for** $t = 1$ to T **do**

2: **Sample:** draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$

3: **Sparse coding:** compute $\boldsymbol{\alpha}^\star$ by sparse coding

4: **Compute:** the active set Λ and vector $\boldsymbol{\beta}^\star$

$$\begin{aligned} \Lambda &\leftarrow \{j \in \{1, \ldots, p\} : \boldsymbol{\alpha}^\star[j] \neq 0\} \\ \boldsymbol{\beta}^\star_\Lambda &\leftarrow (D_\Lambda^T D_\Lambda + \lambda_2 \mathbf{I})^{-1} \nabla_{\alpha_\Lambda} \ell_s(y_t, \mathbf{W}, \boldsymbol{\alpha}^\star) \\ \boldsymbol{\beta}^\star_{\Lambda^C} &\leftarrow 0 \end{aligned}$$

5: **Update learning rate:** $\rho_t \leftarrow \min(\rho, \rho t_0 / t)$

6: **Projected gradient step:**

$$\begin{aligned} \mathbf{W} &\leftarrow \Pi_{\mathcal{W}} \left[\mathbf{W} - \rho_t \left(\nabla_{\mathbf{W}} \ell_s(y_t, \mathbf{w}, \boldsymbol{\alpha}^\star) + \nu \mathbf{W} \right) \right] \\ \mathbf{D} &\leftarrow \Pi_{\mathcal{D}} \left[\mathbf{D} - \rho_t \left(-\mathbf{D} \boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star T} + (\mathbf{x} - \mathbf{D} \boldsymbol{\alpha}^\star) \boldsymbol{\beta}^{\star T} \right) \right] \end{aligned}$$

(where $\Pi_{\mathcal{S}}$ denotes orthogonal projection onto the set $\mathcal{S}$)

7: **end for**

8: **return** $\mathbf{D}, \mathbf{w}$

$$\boldsymbol{\beta}^\star_{\Lambda^C} = 0 \text{ and } \boldsymbol{\beta}^\star_\Lambda = (D_\Lambda^T D_\Lambda + \lambda_2 \mathbf{I})^{-1} \nabla_{\alpha_\Lambda} \ell_s(y_t, \mathbf{W}, \boldsymbol{\alpha}^\star)$$

and Λ are the indices of the nonzero coefficients of $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$. These gradients take the form of expectations, and thus SGD is applicable.

In theory, draws from the distribution of training data $p(\mathbf{x}, y)$ should be made i.i.d. (Algorithm 1, step 2). However, this may be difficult since the distribution itself is typically unknown. As an approximation, samples are instead drawn by iterating over random permutations of the training set [5].

2.2 Sparse Reconstruction

The most computationally intensive step of Algorithm 1 consists of solving the sparse reconstruction problem (Algorithm 1, line 3). In this project, we implemented two different sparse coding algorithms: Least Angle Regression [8] and Feature-Sign [12]. Both are iterative approaches for obtaining the exact solution to the sparse coding problem. For problems of modest size, they represent effective solution techniques; for larger problems, greedy iterative approaches such as orthogonal matching pursuit may be better suited.

2.2.1 Least Angle Regression

Least Angle Regression (LARS) is a model selection technique that can be viewed as a less greedy version of Forward Selection (also known as "forward stepwise regression" or "orthogonal matching pursuit") or as a more efficient implementation of the Forward Stagewise algorithm [8]. The LARS solution procedure always begins with all coefficients equal to zero and incrementally adds or removes predictors one at a time based on correlations with the current residual. Unlike Forward Selection, which incrementally evolves the solution along the selected covariate directions d_j, the LARS solution evolves along a path which is equiangular to all covariates currently in the active set. This particular method for evolving the solution (depicted graphically in Figure 2) has a number of desirable properties, including:

- (**Theorem 1**, [8]) With a small modification to the LARS step size calculation, and assuming covariates are added/removed one at a time from the active set, the complete LARS solution path yields *all* Lasso solutions (i.e. solutions for all choices of regularization parameter λ).
- (**Equation (2.22)**, [8]) Successive LARS estimates $\hat{\mu}_k$ always approach but never reach the OLS estimate $\bar{y}_k$ (except maybe on the final iteration, which means that the $\ell 1$ constraint was non-binding).
- (**Section 3.1**, [8]) With a change to the covariate selection rule, LARS can be modified to solve the *Positive Lasso* problem:

$$\min_\beta \ ||y - X\beta||_2^2 \qquad s.t. \ ||\beta||_1 \leq t$$
$$0 \leq \beta_j \ \ \forall j$$

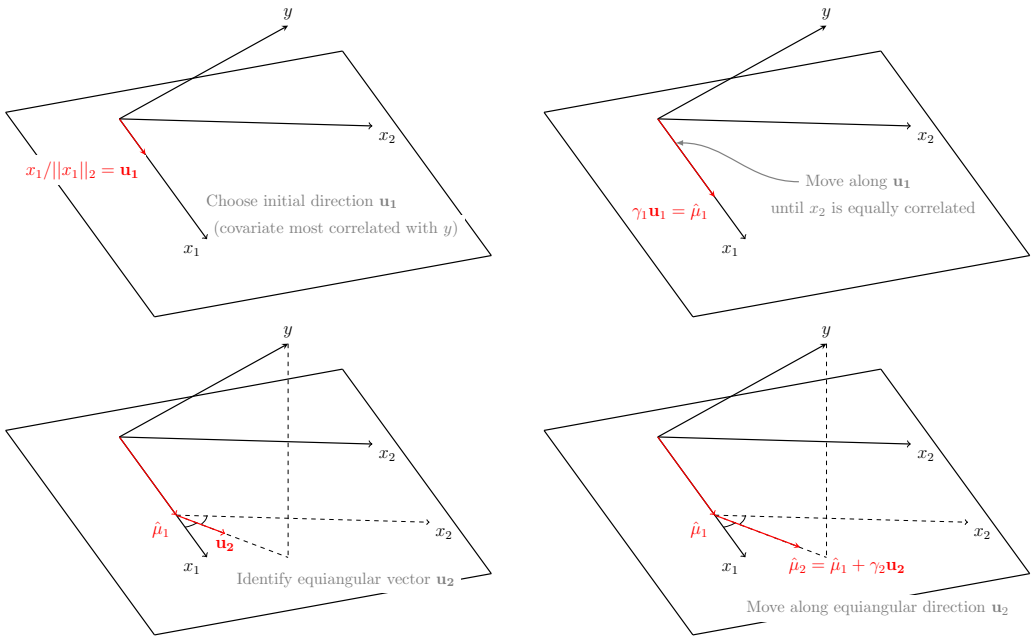


Figure 2: The LARS solution $\hat{\mu}_k$ evolving along equiangular directions in the case where the number of covariates $p = 2$.

This version of the Lasso is useful in domains where the underlying signal being approximated is a positive linear combination of constituent signals and it would be desirable for the sparse approximation to have the same form (e.g. hyperspectral image [6]).

- (**Section 7**, [8]) The cost of LARS is comprable to that of a least squares fit on m variables. The LARS sequence incrementally generates a Cholesky factorization of $X^T X$ in a very specific order.

2.2.2 Feature-Sign

The Feature-Sign algorithm of [12] is another incremental approach for solving sparse coding problems. It works by leveraging the observation that, if the signs of the solution vector elements were known, the lasso optimization problem reduces to an unconstrained quadratic program (QP) with a closed form solution. The Feature-Sign algorithm progresses by making a sequences of guesses at the solution vector signs and uses the resulting residual to improve these guesses. Unlike LARS, it does not have the property that the progression of the algorithm simultaneously sweeps out the full space of possible lasso solutions. However, it has the benefit that the solution procedure can be started from an arbitrary point. When an approximate solution in the vicinity of the true solution is known, this property can improve performance.

3 Implementation

All software for this project was developed in Matlab. The code is broken down into two main software packages. The "solvers" package includes validated implementations of the LARS, Feature-Sign and task-driven dictionary learning algorithms. These implementations are general-purpose, and can be used to solve problems beyond those considered in this project. The solvers package consists of approximately 2000 lines of Matlab, and includes separate scripts for validating the implementation (contained in directories named "Unittests"). The code also includes some limited doxygen tags which can be used to generate browseable HTML documentation.

The "experiments" package consists of code used to preprocess data and generate the results presented in section 6. This code is specific to this project, and not intended for general use. In includes Matlab scripts for running the experiments as well as a number of bash shell scripts for use on machines equipped with Platform's Load Sharing Facility (LSF). Note these scripts contain hard-coded paths and will require modifications in order to run on other platforms.

3.1 LARS

The LARS implementation includes support for the non-negative lasso variant (described in Section 2), the ability to choose various termination criteria (e.g. by specifying a hard constraint on $||\beta||_1$, by specifying the parameter λ_1 in the regularized least squares lasso formulation, setting an explicit number of iterations or a fixed tolerance on the residual) and the ability to enable/disable an incremental Cholesky decomposition for large problems (not used in this project).

3.2 Feature-Sign

The Feature-Sign implementation includes support for the non-negative lasso, the ability to specify an arbitrary starting location and various termination criteria. Note that in the case of the non-negative lasso, the unconstrained QP is replaced by a constrained QP, which is solved using Matlab's quadprog() function. This is substantially less efficient than the unconstrained version. In tests on hyperspectral data sets the runtime of the algorithm increased by a factor of approximately four when using this feature.

3.3 Task-driven Dictionary Learning

The task-driven dictionary learning algorithm implementation includes various options, including the ability to specify a constant bias to the learned weight vector, the ability to restrict dictionary atoms to positive values and the ability to specify a subset of dictionary atoms which should remain immutable throughout the solution process. This latter feature is useful in cases where some atoms are known to be "correct" a-priori and should remain

unchanged while the algorithm adapts the remaining algorithms to the data. Note this feature is not described in the original paper and is included here to enable experiments beyond the scope of this project.

4 Databases

4.1 Synthetic Sparse Approximation

For validation purposes, we generated a number of simulated signals and dictionaries for which the optimal solution to the elastic net sparse approximation problem is known. This is true when the matrix $\mathbf{D}$ is orthogonal. In this case, the elastic net criterion (4) becomes:

$$L = \frac{1}{2}\left(\mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{D}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{D}^T\mathbf{x} + \boldsymbol{\alpha}^T\underbrace{\mathbf{D}^T\mathbf{D}}_{\mathbf{I}}\boldsymbol{\alpha}\right) + \lambda_1\sum_i|\boldsymbol{\alpha}_i| + \frac{\lambda_2}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}$$

The elastic net criterion is convex, and thus we can seek an optimal value by taking the subgradient with respect to $\boldsymbol{\alpha}$ and setting it equal to zero. Consider the ith element of this subgradient,

$$\begin{aligned} 0 &= -\mathbf{D}^T\mathbf{x}_i + \boldsymbol{\alpha}_i + \lambda_1\text{sign}(\boldsymbol{\alpha}_i) + \lambda_2\boldsymbol{\alpha}_i \\ (1+\lambda_2)\boldsymbol{\alpha}_i &= \mathbf{D}^T\mathbf{x}_i - \lambda_1\text{sign}(\boldsymbol{\alpha}_i) \end{aligned}$$

Recall that $\lambda_1, \lambda_2 \geq 0$. If $\mathbf{D}^T\mathbf{x}_i = 0$ it must be that $\boldsymbol{\alpha}_i = 0$ (as this gives $\boldsymbol{\alpha}_i = c\,\text{sign}(\boldsymbol{\alpha}_i)$ with $c < 0$). Consider then the case where $\mathbf{D}^T\mathbf{x}_i \neq 0$. Use $\mathbf{D}^T\mathbf{x}_i = \text{sign}(\mathbf{D}^T\mathbf{x}_i)|\mathbf{D}^T\mathbf{x}_i|$ to write,

$$\boldsymbol{\alpha}_i = \left(|\mathbf{D}^T\mathbf{x}_i| - \lambda_1\frac{\text{sign}(\boldsymbol{\alpha}_i)}{\text{sign}(\mathbf{D}^T\mathbf{x}_i)}\right)\text{sign}(\mathbf{D}^T\mathbf{x}_i)(1+\lambda_2)^{-1}$$

From this equation we see that $\text{sign}(\mathbf{D}^T\mathbf{x}_i)$ and $\text{sign}(\boldsymbol{\alpha}_i)$ cannot differ, as this would lead to a positive (negative) quantity on the LHS equaling a negative (positive) quantity on the RHS. If $\text{sign}(\boldsymbol{\alpha}_i) = \text{sign}(\mathbf{D}^T\mathbf{x}_i)$, it must be the case that $\left(|\mathbf{D}^T\mathbf{x}_i| - \lambda_1\right) \geq 0$ by a similar sign consideration. Therefore the optimal solution for the elastic net can be written,

$$\boldsymbol{\alpha} = \frac{\left(|\mathbf{D}^T\mathbf{x}| - \lambda_1\right)_+\text{sign}(\mathbf{D}^T\mathbf{x})}{1+\lambda_2}$$

where $(z)_+$ denotes the positive part of z. This agrees, with the result provided (without proof) in [20] (after taking into account that the elastic net formulation in [20] does not include the factor of 1/2 in equation (4)).

Patient	Age	Sex	BMI	BP	x_5	x_6	x_7	x_8	x_9	x_{10}	Response
1	59	2	32.1	101	157	93.2	38	4	4.8598	87	151
2	48	1	21.6	87	183	103.2	70	3	3.8918	69	75
3	72	2	30.5	93	156	93.6	41	4	4.6728	85	141
4	24	1	25.3	84	198	131.4	40	5	4.8903	89	206
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
442	36	1	19.6	71	250	133.2	97	3	4.5951	92	57

Table 1: Subset of the diabetes data set provided by the authors of [8]. The measurements $x_5, \ldots x_{10}$ represent blood serum measurements.

4.2 Diabetes

The authors of the original LARS paper provide a data set which contains 10 baseline variables for 442 diabetes patients along with a quantitative measure of disease progression after one year [8]. The baseline measurements include age, sex, body mass index, average blood pressure and six blood serum measurements. This data set is provided in plain text files with the data in both unnormalized and normalized forms. In the latter case, the ten baseline variables are normalized to have mean 0 and Euclidean norm one while the response variable has been centered (i.e. has mean 0). Table 1 shows a subset of the unnormalized data set.

4.3 USPS

The USPS data set is an open collection of handwriting images, corresponding to digits 0-9. The images are 16×16 greyscale images with pixel values scaled to either $[-1, 1]$ or $[0, 2]$ depending upon the data set provider [1, 2]. The training data set contains 7291 images and the test set contains 2007 images. Note there is a known bias between the training and test data sets, making the test data set more difficult than the training set ([19], Appendix B).

4.4 Hyperspectral

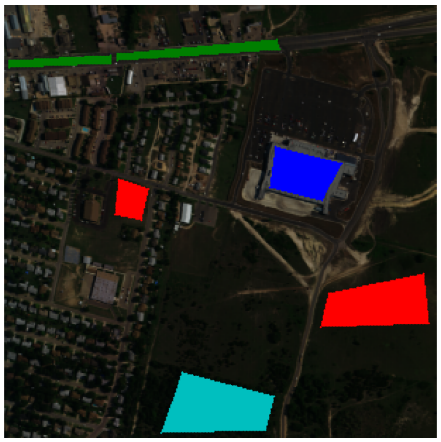
Since dictionary learning algorithms are particularly amenable to large data sets of natural signals, we applied our implementation to hyperspectral imaging data sets. Several open data sets exist in the literature; for this project we used the "Urban" hyperspectral image provided by the US Army Corps of Engineers and the splib06a spectral library provided by the US Geological Survey [18, 7].

The Urban image was captured by the HYperspectral Digital Imagery Collection Experiment (HYDICE) sensor which was flown aboard the Environmental Institute of Michigan's (ERIM) CV-580 aircraft during 1994 and 1995. The image consists of 307×307 pixels with 210 spectral bands containing 16 bit band interleaved line (BIL) data collected from an urban area of Texas. The Urban data set does not include any kind of ground truth, so as a pre-processing step we labeled a subset of pixels as one of rooftops, asphalt, trees and grass

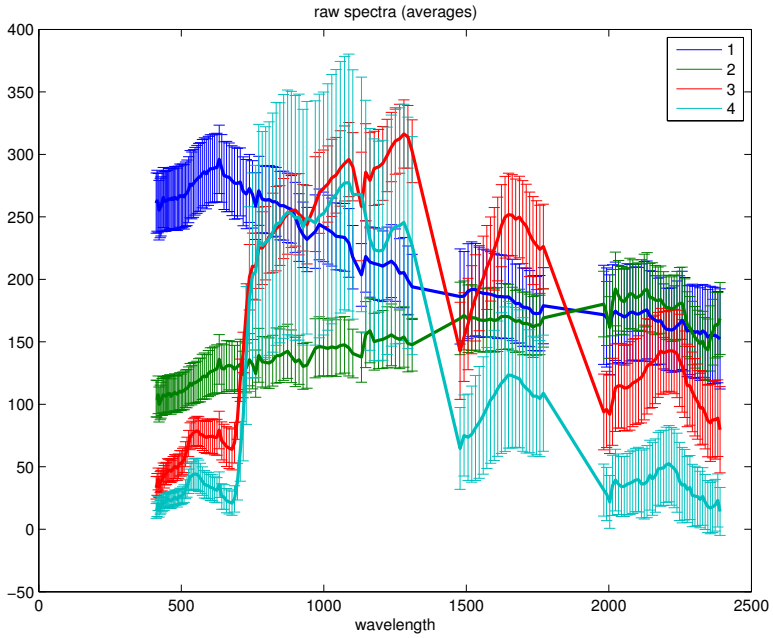
(Figure 3). Note that this data set as provided is in units of radiance; we did not transform this data to reflectance (i.e. perform atmospheric correction) in this effort.

The splib06a spectral library is not a single image but a reference collection of reflectance data for over 1300 different materials, encompassing minerals, vegetation and man-made materials. For our experiments, we used a subset of 44 spectra from the vegetation category with approximately 1200 spectral bins ranging from 0.3-2.5 μm. Figure 4 shows a subset of the USGS spectra as viewed by a pre-processing tool developed as part of this project to identify relatively incoherent subsets of spectral signatures from the USGS library.

(a) Urban Image (approximate RGB)



(b) Classified pixels



(c) Spectra: means and one standard deviation

Figure 3: Urban data set and manually identified ground truth for rooftops, asphalt, trees and grass.

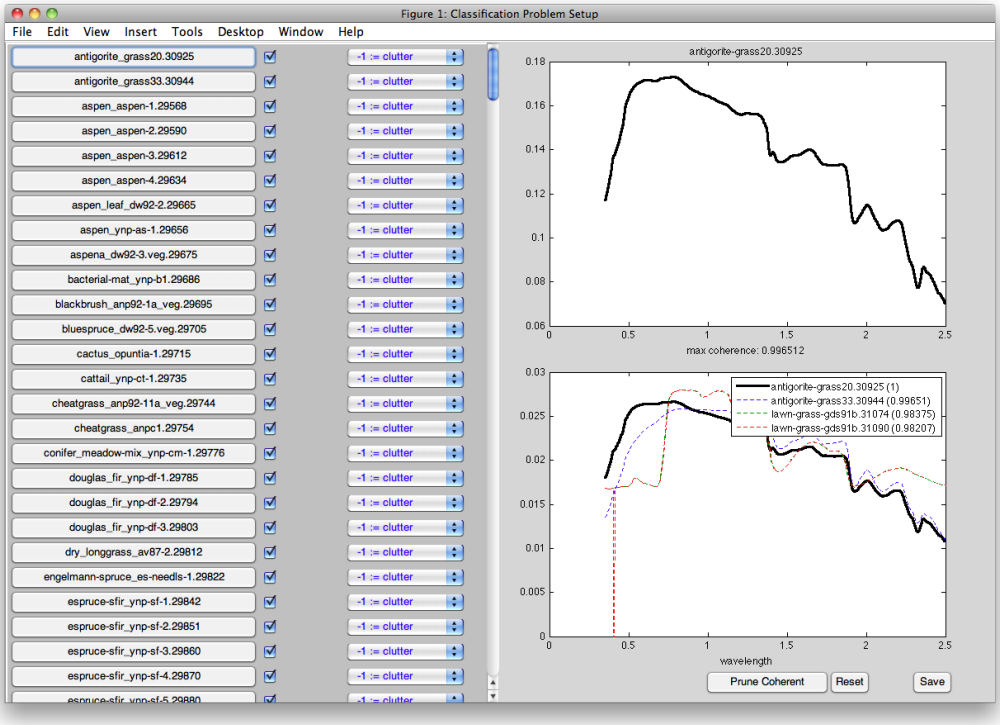


Figure 4: A subset of the USGS vegetation library shown in the spectral viewer developed as part of the pre-processing tools for this project. The upper right panel shows the selected spectral signature while the lower right panel shows the selected signature and the three most correlated signatures.

5 Validation

5.1 LARS

To validate our implementation of the LARS algorithm, we used two approaches. First, we qualitatively compared results obtained using our implementation on the diabetes data set (see Section 4) with those presented in [8]. Figure 5 shows the solution obtained using our implementation of LARS. Qualitatively the two figures are in good agreement, including the relative trajectories of each coefficient and the qualitative behaviors of each trajectory (e.g. the "kink" in the trajectory of coefficient 8 when coefficient 7 leaves and then reenters the active set near $||\beta||_1 = 2800$). In addition to a visual comparison, the code includes a unit test that checks each covariate is added/removed in the expected order and that their relative magnitudes are in agreement with those reported by [8].

We further validated our implementation using the synthetic sparse approximation data set detailed in Section 4. In this case, our unit test randomly generates 200 sparse approximation problems using orthogonal design matrices (half use an identity design matrix and the other half use the Q from a QR decomposition of a Gaussian matrix). We verified that the theoretically expected optimal solution was obtained for both the constrained and the penalized least squares formulations of the lasso.

5.2 Feature-Sign

To validate our implementation of the Feature-Sign algorithm, we checked to make sure that it generated the same results as LARS (within tolerance) on the same diabetes data set used to validate the LARS algorithm.

5.3 Task-driven dictionary learning

To validate our implementation of Algorithm 1, we replicated the handwriting classification experimental setup of [13] and compared results obtained using our implementation on the USPS dataset with those reported in the original paper. The experimental setup consists of the following steps:

1. Pre-process the USPS handwriting data (zero mean, normalize, split into train/validate/test subsets, upsample/shift training data).
2. Learn an initial dictionary using the training data and an unsupervised variant of dictionary learning.
3. For each digit 0-9, create corresponding 1-vs-all binary classification problem.
4. Use the training and validation sets to identify good values for ρ and λ (parameter selection).
5. Use task-driven dictionary learning to find $\mathbf{\Phi}, \mathbf{w}$.

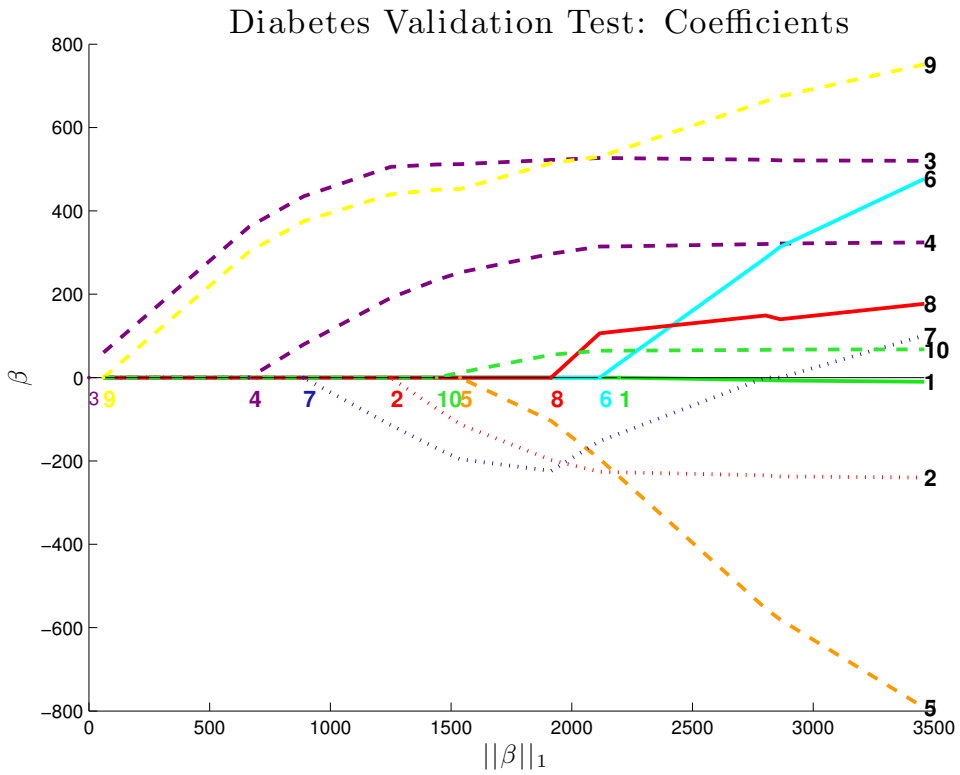


Figure 5: Estimates of regression coefficients $\hat{\beta}_j, j = 1, \ldots, 10$ for the diabetes data set (see Section 4). For each possible choice of constraint on $||\beta||_1$ (x axis), the figure shows the corresponding lasso solution (y axis values for each individual curve). For example, given the constraint $||\beta||_1 \leq 500$, the only nonzero coefficients in the lasso solution are β_3 and β_9. This result matches the corresponding Figure 1 in [8].

6. Evaluate performance on a held-out test set.

Table 2 shows the results obtained using our implementation on the USPS data set. The table shows that the aggregate results match quite well with the reported value.

Digit	ρ	λ	# in D_0	Runtime (h)	Accuracy
0	10	.150	5	8.2	.926
1	10	.225	7	7.1	.990
2	10	.225	7	6.8	.972
3	10	.225	7	7.4	.968
4	10	.225	4	7.6	.971
5	10	.225	4	7.2	.972
6	10	.225	2	7.5	.969
7	10	.175	5	7.9	.983
8	10	.200	3	8.5	.951
9	10	.200	3	8.1	.969
mean					.967
reported					.964

Table 2: Validating the task-driven dictionary learning algorithm by matching performance on the USPS data set. Columns show the target digit, the learning rate parameter, the ℓ_1 regularization parameter, the (approximate) number of times the target digit appeared in the initial dictionary, the runtime in hours and the resulting classification accuracy. The authors of [13] do not provide individual results for each 1-vs-all classification experiment, but the aggregate results are in good agreement (final two rows). Runtimes reported are for a quad core i7 processor.

6 Testing

Sparse coding and related techniques, such as nonnegative matrix factorization, have proven effective in the hyperspectral imaging domain for solving problems in classification (e.g. [6]) and spectral unmixing (e.g. [4, 9, 21]). In this section, we consider sparse coding in the context of the hyperspectral *subpixel detection problem*, where the goal is to identify single pixels containing some known target signature which will only be present in a fractional amount. This problem is related to spectral unmixing in that we assume each pixel consists of a linear combination of pure spectra, but is a relaxation in that we are not interested in identifying *all* constituent spectra and their precise proportions. Instead, we frame the problem as one of classifying mixtures as either containing the target signature (a target mixture) or not containing the target signature (a clutter mixture).

While a perfect algorithm for solving spectral unmixing would also solve the subpixel detection problem, such an algorithm is doing more work than strictly necessary. In cases where the unmixing problem is challenging, it is reasonable to hypothesize that better performance could be obtained by solving the discrimination problem alone. To the best of our knowledge, there have been no prior attempts to apply the task-driven dictionary learning framework to this problem; existing approaches utilize spectral unmixing or tend to preprocess the data (e.g. by performing dimension reduction or sparse coding) and then solve a subsequent (separate) optimization problem.

To test our implementation of the task-driven dictionary learning algorithm, we generated

synthetic pixels using real-world hyperspectral data from two different sources, the Urban data set and a subset of the USGS spectral library (see Section 4 for more details on these data sets). To generate a pixel x we construct a linear combination of n spectral "ingredients",

$$\mathbf{x} = \sum_{k=1}^{n} \phi_k \alpha_k + \epsilon$$

where the mixture abundances α_k are non-negative and satisfy a sum-to-one constraint (i.e. $\sum_i \alpha_i = 1$). Here, ϵ represents Gaussian white noise. The abundance non-negativity constraint (ANC) and abundance sum-to-one constraint (ASC) are frequently used in the hyperspectral literature, and are often included explicitly as constraints in related optimization problems. The ANC motivates our use of the non-negative variants of LARS/Feature-Sign within our task-driven learning implementation. However, we do not explicitly enforce the ASC constraint in our solvers, which is also consistent with certain approaches from the hyperspectral community [9].

For each experiment we construct 1000 mixtures, 500 of which contain the target signal and 500 which are clutter mixtures. These data sets are then split 50/50 into train and test subsets. When the nth mixture is the target signature, we denote the resulting 1-vs-all classification problem as Mn. In the case of target mixtures, we constrain the target signal abundance to lie in the interval [.05 .25]. Since there are no published results corresponding to our specific combination of data sets and problem setup, we also implemented a number of baseline algorithms as a means of comparison. We evaluate the performance of logistic regression, k-nearest neighbors with $k = 1$ and k-nearest neighbors with $k = 3$ for both data in the input space (the raw spectral mixture data) and also the feature space defined via sparse coding (i.e. the sparse coding coefficents become feature vectors for classification). The motivation for considering performance in both the input and feature space is to establish whether any performance gains observed using the task-driven approach can be attributed entirely to the sparse coding preprocessing step or whether the joint optimization is indeed adding value.

From a processing standpoint, these experiments are computationally intensive, and we ran them on a 20 node compute cluster equipped with Platform's Load Sharing Facility (LSF). Each CPU was an Intel Xeon X5650 with 6 cores / 12 threads. The provided software will include scripts to run on this system (named "Maul"); however, these scripts contain hardcoded paths that will have to be changed in order to run on other systems.

In our first experiment, we consider mixtures derived from the Urban data set. In this case, there are only four classes of spectral signature, and each mixture is generated by selecting three of the four signatures. We do not add any Gaussian noise to these mixtures, as they are already derived from a real-world sensor and should have suitably representative noise included. Table 3 shows classification accuracies achieved on this data set using our baseline algorithms and a task-driven dictionary learning algorithm using a dictionary of ten atoms. These results suggest that there is not any significant advantage to be gained with the task-driven approach in this particular scenario. However, a cursory examination of the spectra used from the Urban data set suggests that the different material categories

	Classification Accuracy						
	LR	kNN1	kNN3	LR-SC	kNN1-SC	kNN3-SC	TD-10
M1	84.0	83.2	82.0	**86.8**	77.0	78.4	82.4
M2	82.6	79.2	76.6	75.8	72.8	77.4	81.2
M3	76.8	74.6	75.0	74.0	75.2	69.0	**81.4**
M4	86.4	87.8	82.6	84.0	77.6	78.6	**88.2**

Table 3: 1-vs-all classification accuracy for mixtures generated using spectra from the Urban data set. Best-performing results are shown in bold. In this set of experiments, it is not clear that any algorithm clearly outperforms the others.

are fairly distinct and that sophisticated processing may not be needed in order to achieve reasonable classification performance (Figure 3).

Motivated by the relatively distinct spectral signatures obtained from the Urban data set, we then sought to identify a population of signatures likely to pose a greater challenge. To this end we identified 44 spectra from the vegetation category of the USGS data set that were similar but not so coherent as to lead to unreasonably difficult classification problems. Since the USGS spectra are "pure" measurements obtained in a controlled laboratory setting, we added Gaussian noise to each mixture (with a variance of 0.001). With a larger population of spectra to draw from (forty-four instead of only four), we also increased the number of ingredients in each mixture from three to five. The results for this new classification problem are shown in Table 4. Here we begin to see the promise of the task-driven framework. With more complex mixtures, logistic regression on the raw input space exhibits poor performance (column one), but this performance improves on the sparse coded data (column four). However, for many test cases, an even greater boost in classification performance is achieved when moving to the task-driven framework with 300 atom dictionaries (column eight). In fact, this approach is consistently among the best, and outperforms all baselines in at least five of the eight cases considered. Also worth observing is that the k-nearest neighbors algorithm does a reasonable job, especially for $k = 3$. Given the relative simplicity of this algorithm (both computationally and in terms of the amount of parameter selection required for tuning) these results suggest that k-nearest neighbors makes a reasonable starting point for this classification problem, and more complex procedures can be brought to bear if the initial accuracies are insufficient. Furthermore, given the data-driven nature of k-nearest neighbors, one might expect its performance to improve as more training data is made available.

	Classification Accuracy							
	LR	kNN1	kNN3	LR-SC	kNN1-SC	kNN3-SC	TD-50	TD-300
M1	60.2	63.4	65.8	67.7	57.4	61.6	**70.2**	**70.2**
M2	49.2	61.6	**63.2**	56.4	52.8	60.0	59.4	60.8
M3	52.8	56.2	54.0	56.4	52.2	50.4	54.6	55.0
M4	57.8	62.4	61.8	60.6	54.0	56.6	63.6	**70.8**
M5	55.0	67.6	68.6	66.6	59.6	63.2	64.2	**73.34**
M6	52.2	59.0	62.6	61.0	54.2	57.6	62.8	**65.2***
M7	44.8	56.4	59.6	60.2	56.8	58.4	**63.4**	**64.2***
M8	64.8	80.8	81.4	66.6	64.0	65.2	82.2	81.2

Table 4: 1-vs-all classification results for mixtures generated using the USGS vegetation data set. Best-performing results are shown in bold. Values marked with an '*' indicate test results obtained using a 200 atom dictionary. In this set of experiments, the task-driven algorithm performs as well as or better than all the other baseline algorithms.

7 Schedule, Milestones and Deliverables

7.1 Schedule

The following schedule was proposed at the project inception, and followed throughout the academic year. No significant deviations from this schedule were required. Note that the Feature-Sign algorithm was not initially a part of the original plan, and was added into Phase II as time permitted.

- Phase I: Algorithm development (Sept 23 - Jan 15)
  - Phase Ia: Implement LARS (Sept 23 $\sim$ Oct 24)
  - Phase Ib: Validate LARS (Oct 24 $\sim$ Nov 14)
  - Phase Ic: Implement SGD framework (Nov 14 $\sim$ Dec 15)
  - Phase Id: Validate SGD framework (Dec 15 $\sim$ Jan 15)
- Phase II: Analysis on new data sets (Jan 15 - May 1)

7.2 Milestones

All originally identified project milestones have been met and the results presented. The list below summarizes these milestones and when they were presented.

- Phase I: Initial proposal (**presented September 22, 2011**)
- Phase Ia: Initial Matlab implementation of LARS algorithm available (**presented December 6, 2011**)

- Phase Ib: LARS implementation validated using synthetic and diabetes data (**presented December 6, 2011**)
- Phase Ib: Code review (**conducted December 2, 2011**)
- Phase Ib: Mid-year status presentation (**presented December 6, 2011**)
- Phase Ic: Initial Matlab implementation of SGD code available (**presented March 15, 2012**)
- Phase Id: SGD algorithm validated using MNIST data set (**presented March 15, 2012**)
- Phase II: Presentation on preliminary results for selected dataset (**presented March 15, 2012)**
- **Phase II: Final presentation and report (presented May 1 2012, delivered May 11, 2012)**

7.3 Deliverables

Deliverables for this project include all source code, synthetically generated test data sets and this written report. URLs to hyperspectral data sets used in this project have been provided in the references. Synthetic data sets are provided in the form of Matlab test scripts included with the source. Presentation slides and status reports have been delivered periodically throughout the academic year.

References

[1] Datasets for "The Elements of Statistical Learning". `http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html`.

[2] USPS Dataset. `http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html`.

[3] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: Design of dictionaries for sparse representation. In *Proceedings of SPARS05*, pages 9–12, 2005.

[4] José M. Bioucas-Dias and Antonio Plaza. An overview on hyperspectral unmixing: Geometrical, statistical, and sparse regression based approaches. In *IGARSS*, pages 1135–1138, 2011.

[5] L. Bottou. Online learning and stochastic approximations, 1998.

[6] Adam S. Charles, Bruno A. Olshausen, and Christopher J. Rozell. Learning sparse codes for hyperspectral imagery. *J. Sel. Topics Signal Processing*, 5(5):963–978, 2011.

[7] R.N. Clark, G.A. Swayze, R. Wise, E. Livo, T. Hoefen, R. Kokaly, and S.J. S.J. Sutley. USGS digital spectral library splib06a: U.s. geological survey, digital data series 231, 2007. `http://speclab.cr.usgs.gov/spectral.lib06/`.

[8] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[9] M.-D. Iordache, J.M. Bioucas-Dias, and A. Plaza. Sparse unmixing of hyperspectral data. *Geoscience and Remote Sensing, IEEE Transactions on*, 49(6):2014 –2039, june 2011.

[10] Zico Kolter. Convex optimization overview, 2008.

[11] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms with Applications*. second edition, 2003.

[12] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *In NIPS*, pages 801–808. NIPS, 2007.

[13] Julien Mairal, Francis Bach, and Jean Ponce. Task-Driven Dictionary Learning. Rapport de recherche RR-7400, INRIA, 2010.

[14] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM.

[15] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, 2007.

[16] Ignacio Ramrez, Pablo Sprechmann, and Guillermo Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, pages 3501–3508. IEEE, 2010.

[17] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[18] Army Geospatial Center US Army Corps of Engineers, 2012. `http://www.agc.army.mil/hypercube/`.

[19] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

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

[21] A Zymnis, S J Kim, J Skaf, M Parente, and S Boyd. Hyperspectral image unmixing via alternating projected subgradients. *Machine Learning*, (i):1164–1168, 2007.

A Solver APIs

A.1 LARS

```
> @file lars.m
> @brief An implementation of the Least Angle Regression algorithm [1].

       lars(y,D,params)

   Given measurement vector y \in R^m, dictionary D \in R^{mxn} and
   scalar t \in R, solves the problem:

      min_{betaHat}  ||y - D * betaHat ||_2
       s.t. ||betaHat||_1 <= t

   See sections 2,7 in [1] for algorithm details.

   PARAMETERS:
     y         := the measurement (mx1)
     D         := the dictionary  (mxn)
     lambda1   := the ell1 regularization coefficient (scalar)
     params    := various parameters/flags that control the algorithm.  Values include:
                  [EARLY TERMINATION PARAMETERS]
                    nSteps     := the max number of iterations to run
                    tol        := desired accuracy of 2-norm of residual
                    t          := ||betaHat||_1 <= t
                    lambda1    := penalized least squares coefficient
                                  ||y-A*betaHat||_2^2 + lambda1 ||betaHat||_1
                    condThresh := early termination threshold for condition number of active set.
                                  Only applies to the LASSO case.
                  [SOLVER PARAMETERS]
                    doQR       := true => incremental Cholesky via the QR decomp
                    doLasso    := set to true to implement LASSO (vs LARS)
                    doLassoNN  := true => use Non-Negative LASSO.  Implies doLasso.
                  [MISC PARAMETERS]
                    errCheckLevel :=
                                  0 := no extra checks (you probably don't want to do this though...)
                                  1 := do least expensive checks only
                                  2 := do all extra checks
                    skipNormD  := set to true to skip normalizing D within this function.
                                  *ONLY* do this if D is already normalized...

   REFERENCES:
     [1] Efron et. al. "Least Angle Regression," 2003
     [2] Golub, Van Loan "Matrix Computations," 1996

   VERSIONS:
     10/29/2011 (PM) Reproduced plot from [1]
     11/30/2011 (PM) Ditched the covariate centering.  Now matches
                     l1_ls() fairly well.
     12/01/2011 (PM) LASSO-NN seems more stable finally.
```

A.2 Feature-Sign

```
> @file fs_l1.m
> @brief Implements the sparse coding algorithm 1 from [1]

 Parameters:
   y     : The m dimensional vector to approximate
   A     : The mxn dictionary
   gamma : l1 regularization coefficient
   opts  : Options with default arguments.  See below for details.

 References:
   [1] Lee,Battle,Raina,Ng "Efficient sparse coding algorithms", 2006.
March 2012, mjp
```

A.3 Task-driven Dictionary Learning

```
> @file tddl.m
> @brief  Task driven dictionary learning solver

  An implementation of the Task-driven Dictionary Learning (TDDL)
  Algorithm 1 of [1] (see section 4.2).

  Currently assumes a binary classification problem solved using a
  linear model with a logistic loss and lambda2=0. (i.e. unconstrained
  LASSO vs. elastic net)

  PARAMETERS
   X:    A matrix of objects/features, encoded as columns.
   y:    The class labels associated with X (a column vector).
   D0:   The initial dictionary (a matrix).
   w0:   The initial classifier (a column vector).
   opts: A dictionary of parameters with default values
           [SGD/TDDL OPTIONS]
             tMax:         The number of SGD iterations to run
             nu:           regularization coeff. for W
             nu2:          regularization coeff. for Z
             Z0:           initial linear transform (if any)
             t0:           initial iteration count
             rho:          learning rate.
             nMiniBatch:   SGD mini batch iteration size
             tol:          early termination criteria
           [SPARSE CODING OPTIONS]
             see below
           [MISC OPTIONS]
             quiet:        suppresses staus messages if = 1

  REFERENCES
   [1] Mairal, Bach, Ponce "Task-Driven Dictionary Learning,", 2010
 January 2012, mjp
```