# Analyzing Task Driven Dictionary Learning Algorithms

AMSC 663 Mid-Year Report

Mike Pekala (mike.pekala@gmail.com)

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

December 16, 2011

**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 involving compressible data (e.g. classification, denoising), performance on that task can be improved by explicitly *learning* the sparsifying dictionary directly from the data [16, 17]. 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 also suggest that additional performance gains are possible by *jointly optimizing* the dictionary for both the data and the task of interest [13].

In this project, we propose to implement the task-driven dictionary learning algorithm of [13] with a focus on the binary classification task. We will verify the correctness of our implementation through a combination of unit testing and comparison with published results on open data sets. Finally, we shall apply this algorithm to data sets not considered by [13].

## 1 Introduction

Classical techniques for sparse coding leverage the notion that natural signals can frequently 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 $p > m$). For example, wavelet have a longstanding reputation as an

effective means of compactly representing natural images [15]. Formally, 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 posed:

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

Recently, state-of-the-art results in many signal processing tasks have been achieved using dictionaries learned from test data instead of a predefined one, such as wavelets [16, 17, 6]. 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 data-driven techniques, such as principal component analysis (which, at its core, uses a singular value decomposition), dictionary learning approaches do not require the resulting atoms be orthogonal. Removing this restriction allows even more flexibility to adapt the representation to the data.

Most dictionary learning problems take as input a data set of signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ and produce a dictionary $\mathbf{D} \in \mathbb{R}^{m \times p}$ by minimizing the empirical cost function:

$$g_n(\mathbf{D}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_u(\mathbf{x}_i, \mathbf{D}) \tag{2}$$

where $\ell_u$, 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]:

$$L(\lambda_1, \lambda_2, \boldsymbol{\alpha}, \mathbf{x}, \mathbf{D}) = \frac{1}{2}||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \lambda_1||\boldsymbol{\alpha}||_1 + \frac{\lambda_2}{2}||\boldsymbol{\alpha}||_2^2 \tag{3}$$

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:

$$\ell_u(\mathbf{x}, \mathbf{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} L(\lambda_1, \lambda_2, \boldsymbol{\alpha}, \mathbf{x}, \mathbf{D}) \tag{4}$$

When $\lambda_2 = 0$, minimizing the elastic net criterion corresponds to *basis pursuit*, a convex relaxation of the unconstrained version of the sparse approximation problem (1). Basis pursuit can also be viewed as an unconstrained version of the *Lasso* [18].

To prevent artificially improving $\ell_u$ by simply scaling $\mathbf{D}$, 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 $\mathcal{D}$ of admissible matrices [1]:

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

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 often 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 efficient ways to obtain stationary points of this optimization problem [14].

### 1.1 Task-Driven Dictionary Learning

If the sole objective is to find a sparse representation for data, then the preceding completes the story. 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 preconditioner or feature selector. Here, the signals $\mathbf{x}_i \in \mathbb{R}^m$ are observations and the goal is to identify the associated labels $y_i \in \mathcal{Y}$ (e.g. in binary classification the space of labels is $\mathcal{Y} = \{-1, +1\}$). Given a fixed dictionary $\mathbf{D}$ and the sparsity criterion (3), the sparse approximation:

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

[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 $i$th column of any convex combination of $A$ and $B$, i.e. $(\theta A + (1 - \theta)B)_i = \theta a_i + (1 - \theta)b_i$. 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 [12].

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

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

where $\nu \in \mathbb{R}$ with $\nu \geq 0$ is a regularization parameter and $f$ is a function measuring the prediction quality:

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

Here, $\ell_s$ denotes a supervised loss, which is the classification analog of the unsupervised loss (4). 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:

$$m = \mathbf{w}^T\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D}) \qquad \text{Prediction} \tag{6}$$

$$\ell_s = \log\left(1 + e^{-ym}\right) \qquad \text{Logistic loss} \tag{7}$$

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 described previously [13]. The result is an optimization problem where the goal is to jointly learn $\mathbf{D}$ *and* $\mathbf{W}$:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W}} f(\mathbf{D}, \mathbf{W}) + \frac{\nu}{2}||\mathbf{W}||_F^2 \tag{8}$$

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:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{w}\in\mathbb{R}^p} \mathbb{E}_{y,\mathbf{x}}\left[\log\left(1 + e^{-y\mathbf{w}^T\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})}\right)\right] + \frac{\nu}{2}||\mathbf{w}||_2^2 \tag{9}$$

Section 2 details an approach for solving this optimization problem.

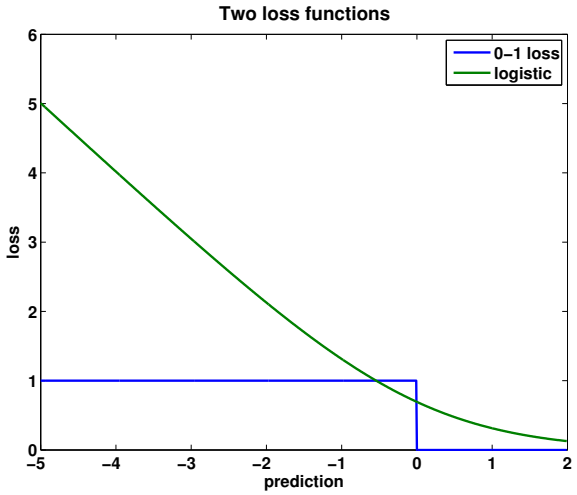


Figure 1: Comparing the 0-1 loss with the logistic loss. The 0-1 loss is not differentiable everywhere and is neither concave nor convex. The logistic loss provides a smooth, convex approximation.

# 2 Approach

This project will develop an implementation of the algorithm defined in [13] for solving the optimization problem (8). In particular, we will focus on an implementation for the optimization problem (9) arising from the binary classification problem using the logistic loss. The authors of [13] provide additional extensions and formulations beyond classification; however, implementing these extensions is not currently within 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 set and an inner loop that solves the sparse approximation problem and updates the dictionary $\mathbf{D}$ and classifier model $\mathbf{w}$.

## 2.1 Stochastic Gradient Descent

The stochastic gradient descent (SGD) algorithm is often used as a way to approximate the gradient of a function, such as the empirical cost $g_n$ [8]. Instead of a direct calculation, SGD incrementally estimates the gradient on the basis of a single randomly selected example $x_t$,

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

**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 using LARS to solve (5)

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

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

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** $\mathbf{D}, \mathbf{w}$

where $Q$ is a loss function, $w$ is a weight being optimized and $\rho_t$ 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 (8) is differentiable on $\mathcal{D} \times \mathcal{W}$, and that

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

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

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

and $\Lambda$ 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 [7].

### 2.2 Least Angle Regression (LARS)

The most significant computational step of the inner loop in Algorithm 1 involves solving the sparse coding problem (Algorithm 1, line 3). This problem is solved efficiently using Least Angle Regression (LARS), 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 [11]. At a high level, LARS begins with all coefficients equal to zero and incrementally adds predictors most correlated with the current residual. Unlike Forward Selection, which incrementally evolves the solution along the 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:

- ([11], Theorem 1) 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.

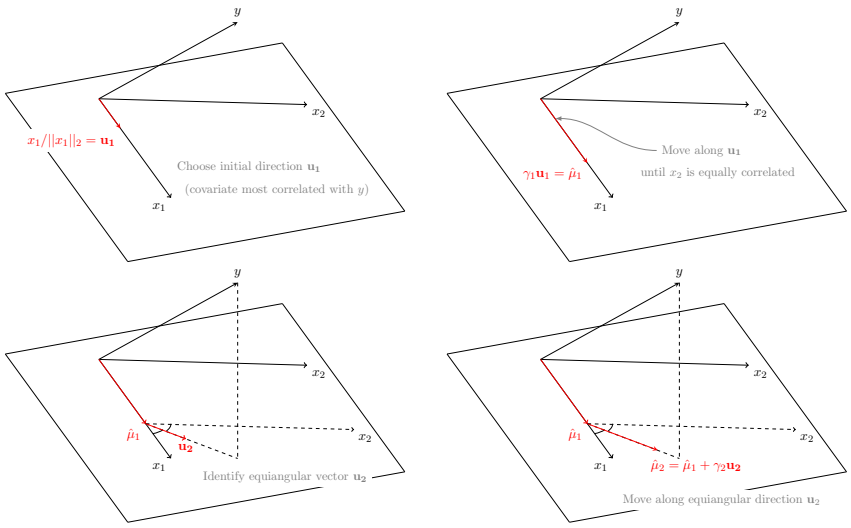


Figure 2: The LARS solution $\hat{\mu}_k$ evolving along equiangular directions in the case where the number of covariates $p = 2$.

- ([11], Equation (2.22)) Successive LARS estimates $\hat{\mu}_k$ always approach but never reach the OLS estimate $\bar{y}_k$ (except maybe on the final iteration).
- ([11], Section 3.1) With a change to the covariate selection rule, LARS can be modified to solve the *Positive Lasso* problem:

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

  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 images [9]).
- ([11], Section 7) 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.

# 3 Implementation

The software for this project will be developed in Matlab on a quad core Mac Pro running OSX. Should performance considerations dictate, computationally intensive parts of the code may be developed in C/C++ and integrated using Matlab's MEX interface [2]. Python and/or standard Matlab toolboxes may also be used for data pre/post processing.

In large scale learning problems, computational complexity is always a potential issue. While stochastic gradient descent has empirically been shown to be effective on large scale problems and much work has been invested analyzing its asymptotic performance [8], care will be required in developing the implementation and problem setup to ensure tractability in practice. An implicit goal of this project is to obtain an improved practical and theoretical understanding of these issues.

Another possible issue is the selection of parameters and initial conditions for new data sets. Identifying effective parameters is crucial for effective performance, and can be quite time consuming. In an attempt to mitigate this risk, the project plan allocates significant time in the second semester for precisely this issue.

## 3.1 LARS

We have a validated implementation of the LARS algorithm. The implementation consists of approximately 560 lines of Matlab (including comments), and includes the Lasso extension to LARS, 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 $\lambda_1$ 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 which may prove useful for large problems. The current API for the LARS algorithm is given in Appendix A.

# 4 Databases

## 4.1 Synthetic Sparse Approximation

For validation purposes, we shall generate 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 (3) 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 $i$th 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 (3)).

Leveraging the above, we will generate test databases for sparse approximation problems in Matlab.

| 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 [11]. 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 [11]. 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 MNIST

The MNIST dataset is a collection of images representing handwritten samples of the digits 0-9 generated by 500 different writers. The images themselves are $28 \times 28$ greyscale images, and the data set contains 60,000 training examples and 10,000 test examples. MNIST is a fairly standard data set in the machine learning community, and is openly available [4].

### 4.4 USPS

The USPS dataset is another open collection of handwriting images, again corresponding to digits 0-9. In this case, the images are $16 \times 16$ greyscale images with pixel values scaled to either $[-1, 1]$ or $[0, 2]$ depending upon the data set provider [1, 5]. 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).

### 4.5 Hyperspectral

Since dictionary learning algorithms are particularly amenable to large datasets of natural signals, we propose to apply our implementation to a hyperspectral imaging data set. Several open datasets exist in the literature; one promising set is included as part of the MultiSpec data analysis system developed by Purdue University [3]. A second option is to leverage the AVIRIS hyperspectral data set, which was also used in [10] (and has the advantage that baseline classification performance is available).

## 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 [11]. Figure 3 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 [11].

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 Task-driven dictionary learning

To validate our implementation of algorithm 1, we shall replicate the handwriting classification experimental setup of [13] and compare results from our implementation on the MNIST and USPS datasets with those in the paper.

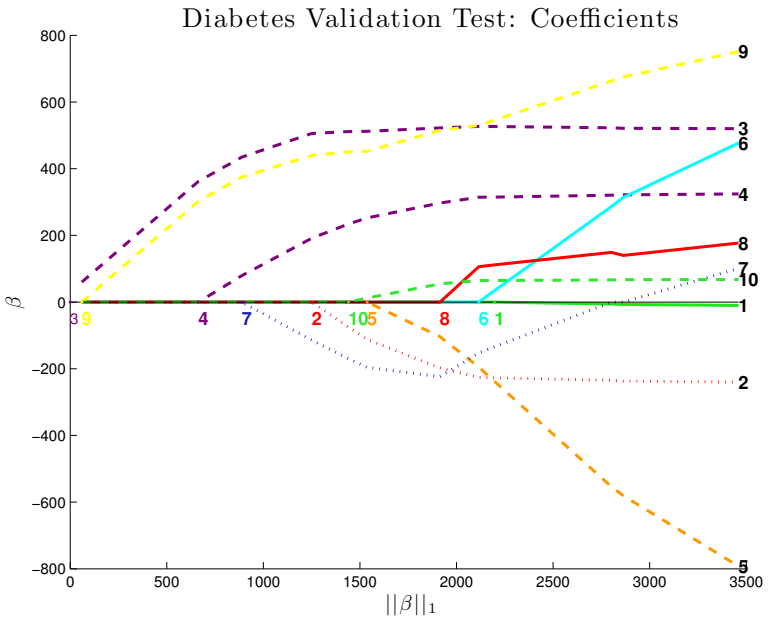


Figure 3: Estimates of regression coefficients $\hat{\beta}_j, j = 1, \ldots, 10$ for the diabetes data set (see Section 4). For each possible choice of constraint on $||\beta||_1$ (x axis), the figure shows the corresponding Lasso solution (y axis values for each individual curve). For example, given the constraint $||\beta||_1 \leq 500$, the only nonzero coefficients in the Lasso solution are $\beta_3$ and $\beta_9$. This result qualitatively matches the corresponding Figure 1 in [11].

# 6 Testing

For testing we propose to use our implementation of the task-driven dictionary learning algorithm to classify subsets of hyperspectral images described in Section 4. In this case, classification will be done on a per-pixel basis by decomposing the original image into a sequence of patches (as opposed to working with the entire image as a single data vector). This approach is often used in dictionary learning problems for images, e.g. [13].

Optionally, we will also consider using this algorithm on non-imaging datasets for comparison. For example, a vaccine data set similar to the diabetes data set described in Section 4 is available, and will be used provided public release issues can be resolved.

# 7 Schedule, Milestones and Deliverables

The following schedule and milestones will be updated as the project progresses.

## 7.1 Schedule

- 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

- Phase Ia: Initial Matlab implementation of LARS algorithm available (Complete)
- Phase Ib: LARS implementation validated using synthetic and diabetes data (Complete)
- Phase Ib: Code review (December 2, 2011) (Complete)
- Phase Ib: Mid-year status presentation (December 6, 2011) (Complete)
- Phase Ic: Initial Matlab implementation of SGD code available (In progress)

- Phase Id: SGD algorithm validated using MNIST and USPS data (In progress)
- Phase II: Preliminary results on selected dataset ($\sim$ Mar 1)
- Phase II: Final report and presentation ($\sim$ May 1)

### 7.3 Deliverables

Deliverables for this project include all source code, synthetically generated test data sets and a written report of results obtained on all data sets.

## References


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

[2] Matlab Product Support: MEX-files Guide. `http://www.mathworks.com/support/tech-notes/1600/1605.html`.

[3] MultiSpec: A Freeware Multispectral Image Data Analysis System. `https://engineering.purdue.edu/~biehl/MultiSpec/index.html`.

[4] The MNIST Database of Handwritten Digits. `http://http://yann.lecun.com/exdb/mnist`.

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

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

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

[8] L. Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent*, page 177187. Springer, 2010.

[9] 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.

[10] T. Doster. Nonlinear Dimensionality Reduction for Hyperspectral Image Classification. `http://www2.math.umd.edu/~rvbalan/TEACHING/AMSC663Fall2010/PROJECTS/P2/index.html`.

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

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

[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] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. 2009.

[16] 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.

[17] 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.

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

[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.

# A LARS Matlab API

```
LARS  An implementation of the Least Angle Regression algorithm [1].

  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)
  params    := termination criteria (structure)
               Relevant fields:
                 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
                 doLasso   := set to true to implement LASSO (vs LARS)
                 doLassoNN := true => use Non-Negative LASSO.  Implies doLasso.
                 doQR      := true => incremental Cholesky via the QR decomp
                 errCheck := set to false to disable assertions for speed


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.

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