Reading Minds is Easy, MRI-te?

Shilpa Kumar
Department of Computer Science
University of Washington
Seattle, WA 98105
shilpak@cs.uw.edu

Siddhartha S. Gorti
Department of Computer Science
University of Washington
Seattle, WA 98105
sgorti3@cs.uw.edu

Abstract

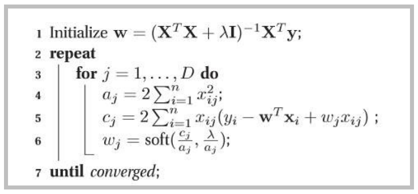
This project involves discovering how fMRI brain scans can be used to predict what word a person is reading based off activation patterns in the brain. The goal is to first learn 218 sparse linear models, each predicting a semantic feature of the word based on an fMRI input. Using Coordinate Descent and Stochastic Coordinate Descent for Lasso, as well Proximal Gradient Descent with L1 penalty, we figured out the coefficients to predict the value of a semantic feature. Given a new brain scan input, we are able to build a 218-dimensional vector representing the values of semantic features of the word read. Given two candidate words, we use binary classification to choose the word whose semantic feature vector is closer to the predicted one.

1 Overview of Methods

We used multiple methods to find the sparse linear model for each semantic feature, to minimize overfitting. Lasso is a regression analysis method that uses λ||w||1 to penalize the magnitude of the coefficients. Lasso allows coefficients to be 'pushed' to 0; this allows for us to achieve sparse linear models without compromising some important features of the model. Each different lasso method used was tested with multiple L1 penalty tuning parameters, and across several semantic features.

1.1 Coordinate Descent for LASSO

Coordinate descent is used in cases where the gradient cannot be easily computed. Lasso uses λ||w||1 as a penalty on the loss function, with λ being the tuning parameter. The absolute value function is not differentiable at x = 0, and a way to work around the complicated gradient that results is to use coordinate descent instead of gradient descent.



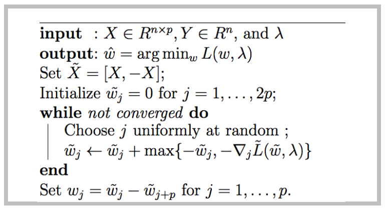
```
Initialize $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$;
repeat
    for $j = 1, \ldots, D$ do
        $a_j = 2\sum_{i=1}^{n} x_{ij}^2$;
        $c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - \mathbf{w}^T\mathbf{x}_i + w_j x_{ij})$ ;
        $w_j = \text{soft}(\frac{c_j}{a_j}, \frac{\lambda}{a_j})$;
until converged;
```

Figure 1: General Algorithm for Coordinate Descent for LASSO[1]

[1]Murphy

1.2 Stochastic Coordinate Descent (SCD)

The coordinate descent algorithm involves going through all data points multiple times per iteration, which can be time-consuming for large datasets. To combat this problem, we utilized stochastic coordinate descent, which uses one random index associated with the voxel (one column of input matrix X) per iteration to minimize the loss function L(w, λ).

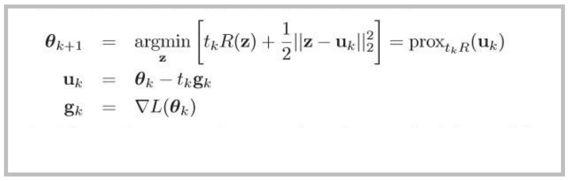


```
input  : $X \in R^{n \times p}, Y \in R^n$, and $\lambda$
output: $\hat{w} = \arg\min_w L(w, \lambda)$
Set $\tilde{X} = [X, -X]$;
Initialize $\tilde{w}_j = 0$ for $j = 1, \ldots, 2p$;
while not converged do
    Choose $j$ uniformly at random ;
    $\tilde{w}_j \leftarrow \tilde{w}_j + \max\{-\tilde{w}_j, -\nabla_j \tilde{L}(\tilde{w}, \lambda)\}$
end
Set $w_j = \tilde{w}_j - \tilde{w}_{j+p}$ for $j = 1, \ldots, p$.
```

Figure 2: General Algorithm for Stochastic Coordinate Descent[2]

1.3 Proximal Gradient Descent (PGD)

To compare our previous algorithms with a method that does not use a version of coordinate descent, we used proximal gradient descent with L1 penalty. Proximal methods are useful for large-scale problems where other methods might be too slow.[3]



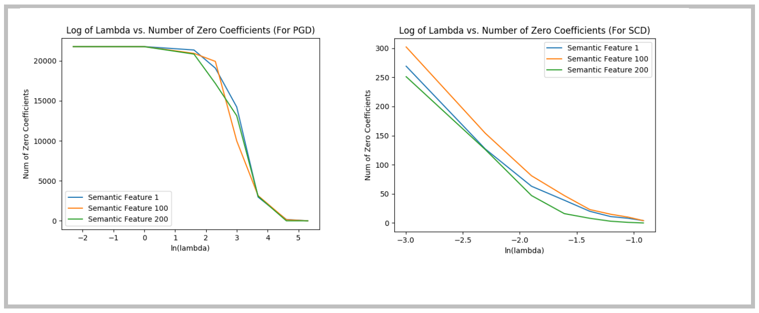
$$
\begin{aligned}
\boldsymbol{\theta}_{k+1} &= \underset{\mathbf{z}}{\operatorname{argmin}} \left[t_k R(\mathbf{z}) + \frac{1}{2} ||\mathbf{z} - \mathbf{u}_k||_2^2 \right] = \operatorname{prox}_{t_k R}(\mathbf{u}_k) \\
\mathbf{u}_k &= \boldsymbol{\theta}_k - t_k \mathbf{g}_k \\
\mathbf{g}_k &= \nabla L(\boldsymbol{\theta}_k)
\end{aligned}
$$

Figure 2: General Algorithm for Proximal Gradient Descent

2 Choosing Tuning Parameters

To help us solve overfitting, we introduce the concept of regularization which helps prevent the generation of overfit models. In building these models, we used two different methods to select our tuning parameters, k-fold cross validation and test set validation. While k-fold is more comprehensive, using the test set to find the optimal lambda value was faster and because this computation had to be done for each model, we settled on using the test set to choose our tuning parameter.

3 Results



[2] Homework 3, CSE 599
[3] Murphy

Figure 4: Log of tuning parameter vs number of nonzero coefficients

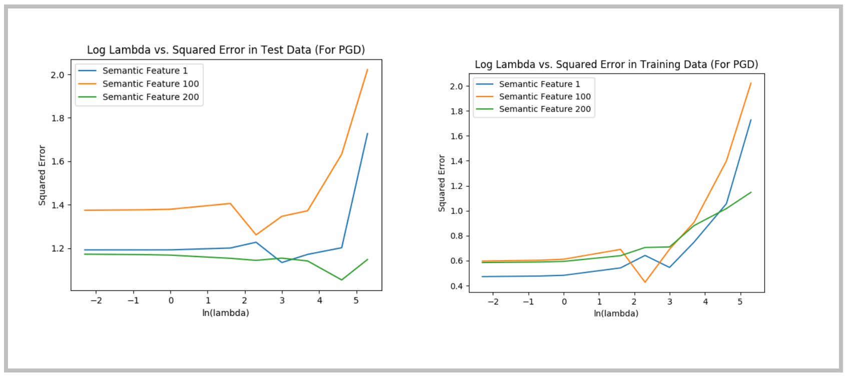


Figure 5: Log of tuning parameter vs test and training error for PGD

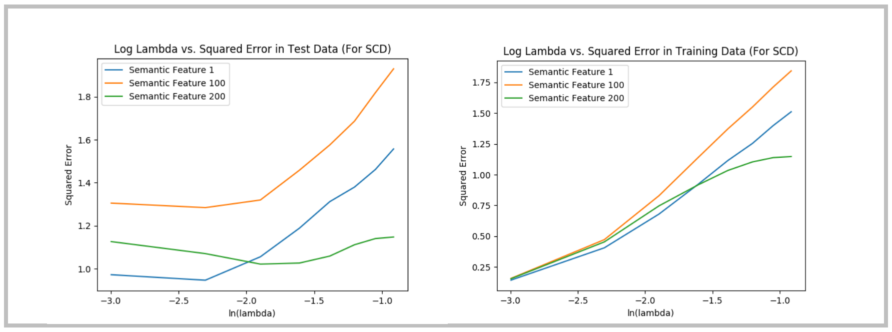


Figure 6: Log of tuning parameter vs test and training error for SCD

4 Comparison

Coordinate descent for Lasso, on our dataset, took an unreasonably long amount of time. For predicting the model of just one semantic feature, the algorithm took upwards of 2.5 hours. We used coordinate descent simply as a baseline comparison, but did not calculate training error or test error for various lambdas because of the length of time necessary for 218 models and various lambdas.

Both our stochastic coordinate descent (SCD) and proximal gradient descent (PGD) followed the correct trends in training error and test error. Training error exponentially increased for various semantic features when lambda was increased, and test error decreased up to a certain lambda, and increased again. However, the graphs for PGD seemed less smooth than those of SCD, and the training error graph for PGD had a small dip in error near $\ln(\lambda) = 2$. In addition, PGD took more time to complete than SCD for the same number of iterations. For the number of nonzero coefficients λ, convergence took much longer for PGD.

5 Future Work

For the next phase of our project we will be working towards feeding in brain scan input into all these linear models to generate a 218 dimensional vector. Then given a choice between two words, we will be comparing the generated vector (representing the semantic features) between the semantic features vector for both words, picking the word whose semantic features vector most closely resembles the generated one, using 1-NN binary classification.

References

[1] Bradley, Joseph K., Aapo Kyrola, Danny Bickson, and Carlos Geustrin. *Parallel Coordinate Descent for L1-Regularized Loss Minimization*. 2011. MS. Carnegie Mellon University, Pittsburgh.

[2] Caramanis, and Sanghavi. "Lecture 4 — September 11." EE 381V: Large Scale Optimization. UTexas. Lecture.

[3] "Homework 3 -- Midterm." CSE 599 / Stat 592: Machine Learning (Statistics) for Big Data. University of Washington Computer Science. Homework Assignment.

[4] Murphy, Kevin P. Machine Learning: A Probabilistic Perspective. Cambridge: The MIT Press, 2012. Print.

[5] Shalev-Schwartz, Shai, and Ambuj Tewari. *Stochastic Methods for L1 Regularized Loss Minimization*. 2011. MS. Toyota Technological Institute at Chicago, Chicago.