University of Washington
Department of Computer Science and Engineering / Department of Statistics

CSE 547 / Stat 548 Machine Learning (Statistics) for Big Data

Homework 3– Midterm

Spring 2015

Must be done individually, without any communication with other students. If you have questions, please contact the instructors for assistance.

Issued: Thursday, April 30, 2015 **Due:** Thursday, May 7, 2015

Suggested Reading: Assigned Readings in Case Study III (see website).

Problem 3.1

Short Answer [15 points]

i. **Nearest Neighbor Search**
Suppose we have a dataset $X = \{x^1, ..., x^N\}$, $N = 1$ million, and want to perform nearest neighbor search. Further, suppose that the dimensionality of the data is large ($d = 10,000$), but for any single data point x^i the number of non-zero entries k_{x^i} is small, say 10 on average. In the following, recall that *support* refers to the number of non-zero entries of x^i.

(a) [**1 point**] True or False: KD-trees modify the sparsity structure of the data. What is the relationship between the support of the KD-tree representation of x^i and that of the original x^i (i.e., greater than, equal to, less than)? Explain.

(b) [**1 point**] True or False: Locality Sensitive Hashing as described in lecture modifies the sparsity structure of the data. What is the relationship between the support of the LSH representation of x^i and that of the original x^i (i.e., greater than, equal to, less than)? Explain.

(c) [**1 point**] True or False: Hashing kernels modify the sparsity structure of the data. What is the relationship between the support of the hashing kernel representation of x^i and that of the original x^i (i.e., greater than, equal to, less than)? Explain.

(d) [**1 point**] Briefly discuss the advantages and disadvantages of the above methods in terms of:

- Scaling with N and d
- Their use for exact or approximate nearest neighbor search
- An ability to control an accuracy versus computation cost tradeoff.

(e) [**1 point**] True or False: If we use TF-IDF and Euclidean distance for nearest neighbor search, we no longer need to worry about normalizing vectors.

ii. **LASSO, Elastic Net, Ridge Regression, Fused LASSO**
Recall that the LASSO objective is to minimize $\text{RSS}(\beta) + \lambda \sum_j |\beta_j|$ whereas the ridge regression objective is to minimize $\text{RSS}(\beta) + \lambda \|\beta\|_2^2$.

(a) [**1 point**] True or False: It is possible to achieve the ridge regression solution using a LASSO objective. Explain.

(b) [**1 point**] True or False: Given any two LASSO solutions corresponding to λ_1 and λ_2 with $\lambda_2 > \lambda_1$ and the same support for these two solutions, it is possible to write out a closed-form expression for *all* solutions corresponding to λ with $\lambda_1 < \lambda < \lambda_2$. Explain.

(c) [**1 point**] Assume we have a value for λ for which there is a sparse solution to the LASSO objective. True or False: stochastic coordinate descent result in a sparse solution.

(d) [**1 point**] Assume we have a value for λ for which there is a sparse solution to the LASSO objective. True or False: the divide-and-average algorithm will result in a sparse solution.

(e) [**1 point**] True or False: The generalized Lasso reduces to the traditional Lasso when $D = I$.

iii. **SCD and SGD**

(a) [**1 point**] True or False: Using Shotgun with a data set of dimension d, if we use $P < d$ processors, we can get interference on every dimension of the regression coefficient vector. Explain.

(b) [**1 point**] True or False: Using parallel SGD, with a data set of dimension d, if we use $P < d$ processors, we can get interference on every dimension of the regression coefficient vector. Explain.

iv. **Map-Reduce**

(a) [**1 point**] True or False: For each input into a mapper, the mapper will only output one key/value pair.

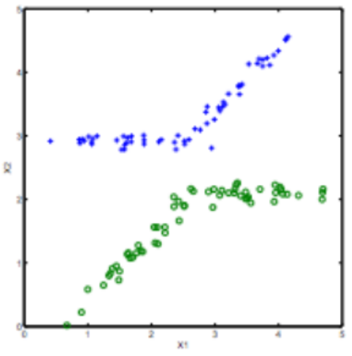
(b) [**1 point**] True or False: A single reducer is responsible to all values associated with the same key.

(c) [**1 point**] True or False: Some reducers can be started before all mappers finish.

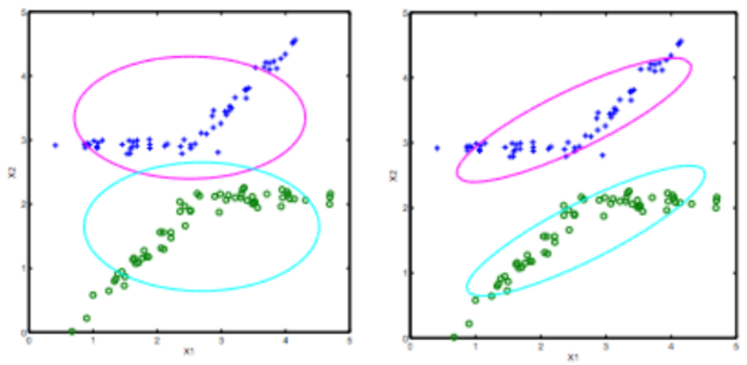
Problem 3.2

Gaussian Mixture Models [10 points]

Consider the labeled training points depicted in the dataset below, where blue "+" and green "o" denote positive and negative labels, respectively. We are going to fit Gaussian Mixture Models on this dataset.



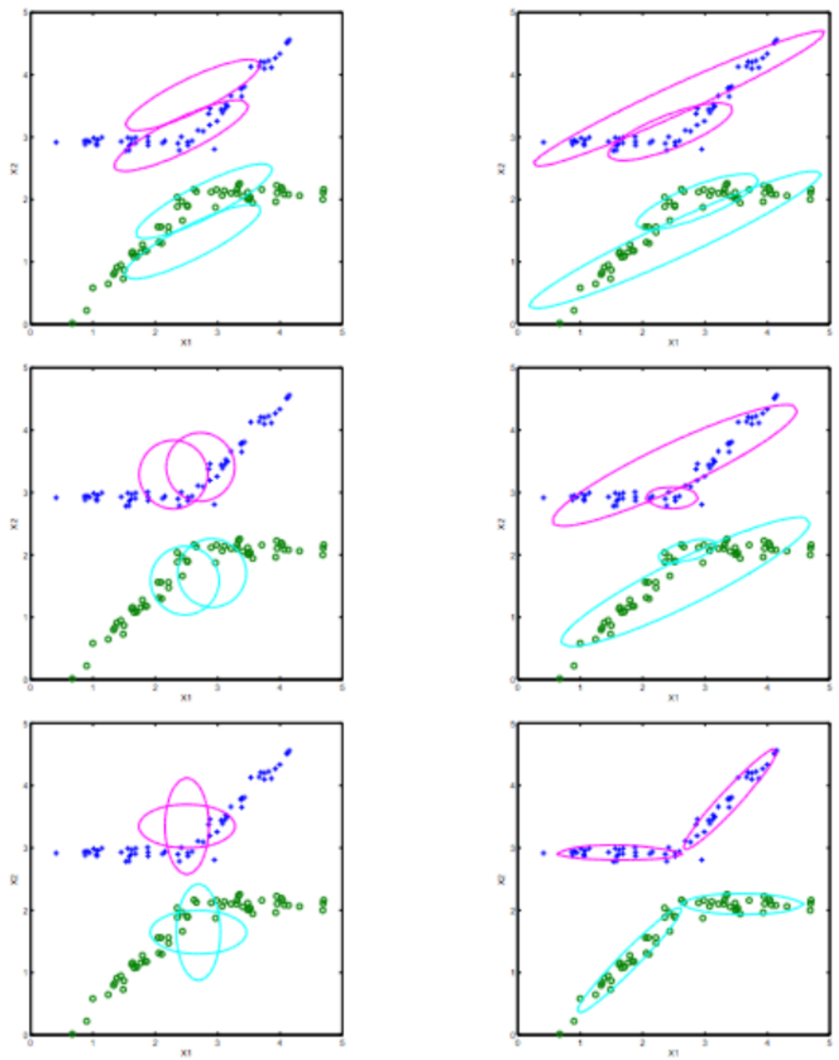
(a) [**4 points**] Let's say Marco and Alden decide to use one Gaussian distribution for positive examples and one for negative examples. The darker elipse indicates the positive Gaussian distribution contour, and the lighter ellipse indicates the negative Gaussian distribution contour.



(a) Marco's model **(b)** Alden's model

Whose model would you prefer for this dataset? What causes the difference between these two models?

(b) [**6 points**] Now Emily decides do use two Gaussian distributions for positive examples and two Gaussian distributions for negative examples. She uses the EM algorithm to iteratively update parameters, and also tries different initializations (she knows very well that EM is susceptible to local minima). The left column of the figure below shows 3 different initializations and the right column shows 3 possible models after the first iteration. For each initialization on the left, draw an arrow to the model on the right that will result after the first EM iteration. Your answer should consisst of 3 arrows, one for each initialization.



(a) Initialization (b) After first iteration

Problem 3.3

Coordinate Descent for the Graphical Lasso [45 points]

The graphical lasso is a method for structure learning in an undirected Gaussian graphical model, using ℓ_1 regularization to encourage sparsity in the precision matrix Ω (the inverse of covariance matrix Σ^{-1}). If the ij-th component of $\Omega = \Sigma^{-1}$ is zero, then variables i and j are conditionally independent given the other variables. Thus, zeros in the precision matrix encode conditional independence statements.

Let $Y = [y^1, ..., y^N]' \in R^{N \times d}$, where $y^i \in R^d$, N is the number of samples, and d is the number of dimensions. Let S denote the sample covariance matrix. As shown in lecture, the graphical lasso objective is to minimize

$$\mathcal{G}(\Omega) = -\log|\Omega| + \mathrm{tr}(S\Omega) + \lambda\|\Omega\|_1$$

over positive definite matrices Ω. The first two terms correspond to the negative log likelihood and $||\Omega||_1 = \sum_{i,j} |\omega_{ij}|$, where ω_{ij} are the elements of Ω.

One approach to solving the graphical lasso is based on a block coordinate descent algorithm, similar to the shooting algorithm for lasso. This approach operates by iteratively updating an estimate of the covariance matrix $\hat{\Sigma}$, from which an estimate of the precision $\hat{\Omega}$ can be readily formed. Letting σ_{ij} be the ij^{th} element of Σ and letting $\sigma_{i\cdot}$ (respectively $s_{i\cdot}$) be the i^{th} column of Σ (respectively S) with the i^{th} element removed, the algorithm is given in Algorithm 1.

Algorithm 1: Graphical Lasso

```
1. Initialize $\Sigma = S + \lambda I$.
2. Repeat for $i = 1, 2, \dots, d, 1, 2, \dots, d, \dots$ until convergence:
    (a) Partition the matrix $\Sigma$ as in Eq. (1) below.
    (b) Using any lasso solver, solve for $\hat{\beta}$ using the lasso objective $\mathcal{L}(\beta)$ with
        $X = (\Sigma_{-i})^{1/2}$ and $y = (\Sigma_{-i})^{-1/2} s_{i\cdot}$.
    (c) Update $\sigma_{i\cdot}$ in terms of $\hat{\beta}$ as $\sigma_{i\cdot} = \Sigma_{-i}\hat{\beta}$
3. Recover $\Omega$ using $\Omega = \Sigma^{-1}$, where $\sigma_{ii} = s_{ii} + \lambda$ and the values of $\sigma_{i\cdot}$ are found from the
    final iteration of step 2.
```

Below, we are going to derive the update steps in the algorithm. Note that the *subgradient* of a function is just a generalization of the gradient to non-differentiable functions. The subgradient at any point is the same as the gradient at that point when a function is differentiable. When the function is not differentiable at a point, the subgradient is any tangent plane to the function at that point. For example, the gradient of $f(x) = |x|$ is $\mathrm{sign}(x)$ when $x \neq 0$ but undefined when $x = 0$. The subgradient, however, is $\mathrm{sign}(x)$ when $x \neq 0$, and when $x = 0$, the subgradient is any point in $[-1, 1]$. We will define $\mathrm{Sign}(x)$ so that $\mathrm{Sign}(x) = \mathrm{sign}(x)$ when $x \neq 0$ and $\mathrm{Sign}(x) \in [-1, 1]$ when $x = 0$.

Using this, we have that $\gamma_{ij} = \partial_{\omega_{ij}} \|\Omega\|_1$, the subgradient of the matrix ℓ_1 norm, is simply

$$\gamma_{ij} = \partial_{\omega_{ij}} \|\Omega\|_1 = \partial_{\omega_{ij}} \sum_k \sum_l |\omega_{kl}| = \mathrm{Sign}(\omega_{ij})$$

(a) [**5 points**] Let Γ be the matrix with elements γ_{ij}. Write the graphical lasso subgradient $\frac{\partial}{\partial \Omega}\mathcal{G}(\Omega)$ in terms of Σ, S, Γ, and λ.
Note: using the properties of the adjugate matrix in linear algebra, we have

$$\frac{\partial}{\partial \Omega}\log|\Omega| = \frac{1}{|\Omega|}\frac{\partial}{\partial \Omega}|\Omega| = \frac{1}{|\Omega|} adj(\Omega)^T = \Omega^{-1}$$

(b) [**5 points**] Find the optimal σ_{ii} by setting the subgradient from the previous part equal to 0 and solving. (*Hint: use the fact that $\omega_{ii} > 0$ for all $i = 1, \ldots, d$.*)

Now we know how to find the optimal diagonal terms of Σ. The remaining parts will guide you through the derivation of a coordinate descent algorithm for the terms σ_{ij}, $i \neq j$. We can actually think about updating an entire column of Σ all at once. Recalling that Σ is symmetric, we are really updating a row *and* column.

Assume we have a current estimate of our covariance matrix, which for simplicity of notation we also denote by Σ with corresponding inverse Ω. When considering the update to the ith row and column, we examine the partitioned matrix

$$\begin{bmatrix} \Sigma_{-i} & \sigma_{i\cdot} \\ \sigma_{i\cdot}^T & \sigma_{ii} \end{bmatrix}, \tag{1}$$

where Σ_{-i} denotes the $(d-1) \times (d-1)$ matrix of rows and columns of Σ except for the ith, and $\sigma_{i\cdot}$ denotes the ith column of Σ except for the element σ_{ii} itself. We also rearrange and partition S and Ω accordingly:

$$\begin{bmatrix} S_{-i} & s_{i\cdot} \\ s_{i\cdot}^T & s_{ii} \end{bmatrix}, \quad \begin{bmatrix} \Omega_{-i} & \omega_{i\cdot} \\ \omega_{i\cdot}^T & \omega_{ii} \end{bmatrix}.$$

Since we have already solved for σ_{ii}, we focus exclusively on the vector $\sigma_{i\cdot}$ of terms σ_{ij}, $j \neq i$.

(c) Recall that we showed in part (b) that the solution to the graphical lasso problem will satisfy

$$-\sigma_{i\cdot} + s_{i\cdot} + \lambda\gamma_{i\cdot} = 0. \tag{2}$$

In this part we will show that for some definition of $\beta \in \mathbb{R}^{d-1}$, we can rewrite this equation as

$$-\Sigma_{-i}\beta + s_{i\cdot} - \lambda\nu = 0, \tag{3}$$

where $\nu_j = \mathrm{Sign}(\beta_j)$.

(i) [**3 points**] Determine $\sigma_{i\cdot}$ in terms of Σ_{-i} and elements of Ω.
(*Hint: Use the identity $\Sigma\Omega = I$ for the partitioned matrices, expanding the terms for the top righthand block.*)

(ii) [**2 points**] Find β such that the first term of Eq. (2) matches that of Eq. (3).

(iii) [**2 points**] Show that with this definition of β, Eq. (2) and Eq. (3) are equivalent.

(d) [**3 points**] In our standard lasso setting, we aimed to minimize the following objective with respect to β:

$$\mathcal{L}(\beta) = \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1$$

The subgradient of this objective is

$$\frac{\partial}{\partial\beta}\mathcal{L}(\beta) = X^T X\beta - X^T y + \lambda\nu$$

where $\nu = \text{Sign}(\beta)$.
Letting $X = (\Sigma_{-i})^{1/2}$ and $y = (\Sigma_{-i})^{-1/2} s_{i\cdot}$, show that this lasso subgradient corresponds to the negative of the left-hand side of Eq. (3).

This shows that we can frame the solution for β in terms of a lasso solution. Thus, we can use any number of lasso algorithms to iteratively solve for $\hat{\beta}$. Using the optimal $\hat{\beta}$, we can then solve for $\sigma_{i\cdot}$.

(e) [**2 points**] Recover $\sigma_{i\cdot}$ from $\hat{\beta}$.

(f) [**20 points**] We have provided a data set of 1000 samples from a multivariate normal distribution in $\mathbb{R}^5$ in `glasso_data.csv`. Implement Algorithm 1 using ADMM as the internal LASSO solver and run it on the provided data set.
Use $\lambda = 0.1$ and $\rho = 1.0$. For stopping criteria on each ADMM run, use $\|\beta^{(t)} - \beta^{(t-1)}\|_2 < 10^{-10}$ or a maximum of 100 iterations. For the outer algorithm convergence, use $\|\Sigma^{(t)} - \Sigma^{(t-1)}\|_2 < 10^{-10}$ or a maximum of 100 iterations (going through all 5 dimensions is one iteration).
Report the final $\hat{\Omega}$ matrix you obtain, rounding all values to 3 decimal places.

(g) [**3 points**] Note that the original precision matrix from which the data was generated is

$$\Omega = \begin{bmatrix} 3.0 & 0.8 & 0 & 0 & 0 \\ 0.8 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0.3 & 0 \\ 0 & 0 & 0.3 & 1.0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 1.0 \end{bmatrix}$$

Comment on your answer to part (f). Do the zero and nonzero entries make sense? What has happened to the nonzero entries?

Problem 3.4

READ THE MIND [30 points]

In this problem, you will implement stochastic coordinate descent (SCD) for ℓ_1 regularized regression. The goal is to learn a set of sparse linear models to predict word semantic features from fMRI signals. Using the predicted features, you will construct a binary classifier (using 1-NN) that takes 2 candidate words and predicts which word the subject was thinking.

Matrix notation: We will use subscript (superscript) to denote the column (row) of a matrix, e.g. For any matrix X, X_j is the jth column, X^i is the ith row, and $X_{i,j}$ is the entry (i, j).

Let $X \in R^{N\times p}$ and $Y \in R^N$. The LASSO solves the following optimization problem:

$$\min_{\beta_0,\beta} \frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - (X^i)^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \tag{4}$$

If Y is centered (the mean is 0), we can usually ignore β_0 in the above equation. We will keep this assumption for the rest of this problem. In matrix form, we can rewrite Eq. (4) as

$$\min_{\beta} L(\beta,\lambda) = \frac{1}{2N}\|Y - X\beta\|^2 + \lambda\|\beta\|_1 \tag{5}$$

3.4.1 Deriving the Shooting Algorithm in Practice [15 points]

The general Stochastic Coordinate Descent algorithm is shown in Algorithm 2.

```
Algorithm 2: General Stochastic Coordinate Descent
input  : Objective function $L(\beta)$
output: Optimal $\beta$
$\beta^{(0)} = 0$;
$t = 0$;
while not converged do
    Choose a coordinate $j$ uniformly at random ;
    $\beta_j^{new} \leftarrow \min_{\beta_j} L(\beta_0^{(t)}, \ldots, \beta_{j-1}^{(t)}, \beta_j, \beta_{j+1}^{(t)}, \ldots, \beta_p^{(t)})$;
    $\beta^{(t+1)} \leftarrow \beta^{(t)}$;
    $\beta_j^{(t+1)} \leftarrow \beta_j^{new}$;
    $t \leftarrow t + 1$;
end
```

In order to find the minimum at coordinate j, the Shooting algorithm applies the following procedure: First, let's split each β_i into its positive part β_i^+ and negative part β_i^-, such that

$\beta_i = \beta_i^+ - \beta_i^-$. Note that both β_i^+ and β_i^- are non-negative. When β_i is positive, $\beta_i^- = 0$, and vice versa. Then, we can rewrite Eq. (5) as follows:

$$\min_{\tilde{\beta}} \tilde{L}(\tilde{\beta}, \lambda) = \frac{1}{2N} \|Y - \tilde{X}\tilde{\beta}\|_2^2 + \lambda \sum_{j=1}^{2p} \tilde{\beta}_j \tag{6}$$

where $\tilde{X} = [X, -X] \in R^{N \times 2p}$, and $\tilde{\beta} = [\beta^+; \beta^-] \in R^{2p}$. Taking $\beta_i = \tilde{\beta}_i - \tilde{\beta}_{i+p}$ will recover the original β.

After getting rid of the ℓ_1 norm, it becomes straight-forward to take the derivative $\nabla_j \tilde{L}(\tilde{\beta})$ at each coordinate $j = [1, 2, .., 2p]$. However, in the update step, we need to take extra care to enforce the non-negativity constraint. In particular, the minimization step becomes simply:

$$\tilde{\beta}_j^{(t+1)} \leftarrow \tilde{\beta}_j^{(t)} + \max\{-\tilde{\beta}_j^{(t)}, -\nabla_j \tilde{L}(\tilde{\beta}^{(t)})\} \tag{7}$$

Algorithm 3 shows the pesudo-code for the Shooting algorithm.

Algorithm 3: Shooting: Sequential Stochastic Coordinate Descent for LASSO

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

(a) **[3 points]** Derive $\nabla_j \tilde{L}(\tilde{\beta}, \lambda)$ for Eq. (6) in terms of $\tilde{X}, \tilde{\beta}$, Y, and λ.

(b) **[5 points]** Show how to compute $\nabla_j \tilde{L}(\tilde{\beta}, \lambda)$ without explicitly replicating X. In other words, rewrite your solution in terms of X, β^+, β^-, Y and λ. (In practice, this step is important to avoid using extra memory.)

(c) **[2 points]** What is the complexity, in $\mathcal{O}$ notation, of computing the derivative?

(d) **[2 points]** Alternatively, if we cache $\tilde{X}\tilde{\beta}$, we can significantly decrease the complexity. If we knew the result of this product, what is the cost of the update now?

(e) **[3 points]** Suppose you updated $\tilde{\beta}_j \leftarrow \tilde{\beta}_j + \delta$. Show how to update $\tilde{X}\tilde{\beta}$ in terms of δ efficiently.

3.4.2 Shooting at the Brain [15 points]

Our fMRI data records the brain activity of one subject while viewing a set of different words. The original data for this question is from:
`http://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html`.

The vocabulary set contains 60 nouns covering different categories. For each word, 218 semantic features are extracted. Each feature corresponds to a question about the semantic meaning of this word. The value is a score ranging from 1 to 5 provided by a human labeler in response to the question. For example, feature 1 corresponds to the question: "IS IT AN ANIMAL?". The word "ant" has value 4 for feature 1, whereas the word "airplane" has value 1.

At trial i, $word^i$ is shown to the subject. The fMRI recordings provide the activity in each of a set of 21764 voxels (cubes in the 3-d coordinates of the brain), which we represent by a 21764 dimension vector X^i. There are 360 trials in total, for which the results are put into $X \in R^{360\times 21764}$ and $Y \in R^{360\times 218}$. $X_{i,j}$ is the signal at jth voxel at trial i, and $Y_{i,\ell}$ is the ℓth semantic feature of the word displayed at trial i.

We further standardize X to have mean 0 and unit norm for each column and center Y to have mean 0 for each column. Finally, the data is split into a training set (300 trials) and test set (60 trials).

The goal is to learn 218 sparse linear models, each predicting a semantic feature of the output space: $\beta^i = LASSO(X, Y_i) \quad i = 1 \dots 218$. Given a new input X^{n+1} the entire model will output a 218 dimension vector Y^{n+1}. Given two candidate words, we create a binary classifier by choosing the word whose semantic feature vector is closer to the predicted one in ℓ_2 distance.

- Download `fmri.zip` from the course website.
- After unzipping the folder, there should be seven files:
  - `subject1_fmri_std.train.mtx`: The standardized fMRI signal for subject1. This will be the X of the training data where $X_{i,j}$ is the signal at jth voxel at trial i.
  - `subject1_wordid_std.train.mtx`: Each line corresponds to a word id. Line i is the id of the word shown to subject1 at trial i.
  - `subject1_fmri_std.test.mtx`: Same format as "subject1_fmri_std.train.mtx", used for testing.
  - `subject1_wordid_std.test.mtx`: Contains two columns. The first column is the ids of the ground truth words for test data. The second column contains ids chosen at random.
  - `word_feature_centered.mtx`: A 60×218 matrix, where row i is the semantic feature vector for word with $id = i$. The matrix is written in the dense Matrix Market format.

- `meta/dictionary.txt`: Mapping from id to the word. Line i is the word whose $id = i$.
- `meta/semantic_feature_name.txt`: Contains the meta information of the 218 semantic features.

For this question, we will focus on only the first semantic feature.

Implement the sequential Shooting algorithm for semantic feature 1. Using $\lambda = [0.00, \dots, 0.15]$ spaced by 0.01, please submit three plots for this question:

- Plot the ℓ_2 loss on training data against λ.
- Plot the ℓ_2 loss on test data against λ.
- Plot the ℓ_0 norm of the weight vector $\hat{\beta}$ against λ

(Hint: Complete the two core functions: `shoot()` and `scd()` in `Shooting.[py|java]` in the starter code. In the Java code, you may find pre-implemented linear matrix and vector operations in `MatUtil.java`. In the Python code, we rely on `numpy` and `scipy` for linear algebra routines. After implementing `shoot()` and `scd()`,you should be able to run `FMRIWordPrediction.[py|java]`, which will load the data, run `scd()`, and output the result.)