Stony Brook University
CSE512 – Machine Learning – Spring 18
Homework 1, Due: Feb 14 at midnight 11:59pm

This homework contains 4 questions. The last question requires programming. The maximum number of points is 100 plus 10 bonus points. Bonus points will not be used to fit the grade curve.

For Questions 3 and 4 (and it is a good idea in general), we will use the following notations. Bold uppercase letters denote matrices (e.g. $\mathbf{D}$), bold lowercase letters denote column vectors (e.g. $\mathbf{d}$). $\mathbf{d}_i$ represents the i^{th} column of the matrix $\mathbf{D}$. d_{ij} denotes the scalar in the row j^{th} and column i^{th} of the matrix $\mathbf{D}$; it is also the j^{th} entry of column vector $\mathbf{d}_i$. Non-bold letters represent scalar variables. $\mathbf{1}_k \in \Re^{k\times 1}$ is a column vector of ones. $\mathbf{0}_k \in \Re^{k\times 1}$ is a column vector of zeros. $\mathbf{I}_k \in \Re^{k\times k}$ is the identity matrix. We use **comma** to denote horizontal concatenation of two vectors (or matrices) (e.g., $[\mathbf{a}, \mathbf{b}]$). We use **semicolon** to denote vertical concatenation of two vectors (e.g., $[\mathbf{a}; \mathbf{b}]$).

1 Question 1 – Probability (15 points)

Let X_1 and X_2 be independent, continuous random variables uniformly distributed on $[0, 1]$. Let $X = \max(X_1, 2X_2)$. Compute

1. *(5 points)* $E(X)$.
2. *(5 points)* $Var(X)$.
3. *(5 points)* $Cov(X, X_1)$.

2 Question 2 – Parameter estimation (30 points)

This question uses a discrete probability distribution known as the Poisson distribution. A discrete random variable X follows a Poisson distribution with parameter λ if

$$p(X = k|\lambda) = \frac{\lambda^k}{k!}e^{-\lambda} \qquad k \in \{0, 1, 2, \dots\}$$

The Poisson distribution is a useful discrete distribution which can be used to model the number of occurrences of something per unit time. For example, it can be used to model the number of customers arriving at a bank per hour, or the number of calls handled by a telephone operator per minute.

2.1 Question 2.1 – MLE (10 points)

Consider catching the LIRR train from Stony Brook to Penn Station. Of course, the train is always late, and assume the amount of delay in minutes is Poisson distributed (i.i.d) with parameter λ. Since you like Times Square so much, you have visited NYC several times and record the amount of train delays in each trip.

Trip	1	2	3	4	5	6	7
Delay in minutes	4	5	3	5	6	9	3

Let $\mathbf{X} = (X_1, \dots, X_n)$ be a random vector where X_i is the number of delay minutes for your i^{th} trip:

1. *(4 points)* Give the log-likelihood function of $\mathbf{X}$ given λ.
2. *(4 points)* Compute the MLE for λ in the general case.
3. *(2 point)* Compute the MLE for λ using the observed $\mathbf{X}$.

2.2 Question 2.2 – MAP (10 points)

Now let's be Bayesian and put a prior over the parameter λ. You talk to your party-goer friends, who tell you about the expected delays that they experience. You plot the distribution of the expected delays and your extensive experience in statistics tells you that the plot resembles a Gamma distribution pdf. So you believe a good prior distribution for λ may be a Gamma distribution. The Gamma distribution has pdf:

$$p(\lambda|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \lambda > 0. \tag{1}$$

$\Gamma(\alpha)$ is the Gamma function, which is the normalization factor that is introduced to have a proper probability density function (i.e., sum to 1). Do not worry if you don't know the explicit format of $\Gamma(\alpha)$; it is not important for this question. Also, if $\lambda \sim \Gamma(\alpha,\beta)$, then it has mean α/β and the mode is $(\alpha-1)/\beta$ for $\alpha > 1$. Assume the prior distribution of λ is $\Gamma(\lambda|\alpha,\beta)$.

1. *(5 points)* Compute the posterior distribution over λ. Hint:

   $$\lambda^{\sum_{i=1}^n X_i+\alpha-1}e^{-\lambda(n+\beta)}$$

   looks like a Gamma distribution! Is the rest of the expression constant with respect to λ? Working out a messy integral can lead to the answer but it is unnecessary.

2. *(5 points)* Derive an analytic expression for the maximum a posterior (MAP) estimate of λ.

2.3 Question 2.3 – Estimator Bias (10 points)

In class, we saw that the maximum likelihood estimator for the variance of a Gaussian distribution:

$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^N (x_i - \hat{\mu})^2$$

is biased, and that an unbiased estimator of variance is:

$$\hat{\sigma}^2_{unbiased} = \frac{1}{N-1}\sum_{i=1}^N (x_i - \hat{\mu})^2$$

For the Gaussian distribution, these estimators give similar results for large enough N, and it is unclear whether one estimator is preferable to the other. In this problem, we will explore an example in which the maximum likelihood estimate is dramatically superior to any unbiased estimator.

Suppose we are not quite interested in estimating λ. We are more interested in estimating a quantity that is a *nonlinear* function of λ, namely $\eta = e^{-2\lambda}$. Suppose we want to estimate η from a single observation $X \sim Poisson(\lambda)$.

1. [4pts] Let $\hat{\eta} = e^{-2X}$. Show that $\hat{\eta}$ is the maximum likelihood estimate of η.

2. [3pts] Show that the bias of $\hat{\eta}$ is $e^{-2\lambda} - e^{\lambda(1/e^2-1)}$. The following Taylor expansion may be useful:

   $$e^x = \sum_{k=0}^\infty \frac{x^k}{k!}$$

3. [3pts] It turns out that $(-1)^X$ is the *only* unbiased estimate of η. Prove that it is indeed unbiased and briefly explain why this is a bad estimator to use.

3 Question 3 – Regression and MLE [15 points]

Suppose the data $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \Re^k, y_i \in \Re\}_{i=1}^n$ are generated by a linear regression model:

$$y_i = \mathbf{w}^T\mathbf{x}_i + \epsilon_i. \tag{2}$$

3.1 Question 3.1 (6 points)

Assume that $\epsilon_1, \cdots, \epsilon_n$ are independent Gaussians with mean 0, but the variances are different, i.e., $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$. Show that the MLE of $\mathbf{w}$ can be found by solving the weighted least squares problem, i.e.,

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{i=1}^{n} s_i(y_i - \mathbf{w}^T\mathbf{x}_i)^2. \tag{3}$$

Give the weights s_i. Derive the closed-form solution for the MLE of $\mathbf{w}$.

3.2 Question 3.2 (6 points)

If $\epsilon_1, \cdots, \epsilon_n$ are independent Laplace with mean 0 and the same scale parameter b, i.e., the pdf of ϵ_i is $p(\epsilon_i) = \frac{1}{2b}exp(-\frac{|\epsilon_i|}{b})$, give the formulation for calculating MLE for $\mathbf{w}$ (closed form solution is not required).

3.3 Question 3.3 (3 points)

Sometimes the model in the second question is preferred because its solution tends to be more robust to noise. Explain why this is true.

4 Question 4 – Programming [40 points]

The Lasso is the problem of solving

$$\arg\min_{\mathbf{w},b} \sum_i (\mathbf{w}^T\mathbf{x}_i + b - y_i)^2 + \lambda \sum_j |w_j| \tag{4}$$

Here $\mathbf{X}$ is an $d \times n$ matrix of data, and $\mathbf{x}_i$ is the i^{th} column of the matrix. $\mathbf{y}$ is an $n \times 1$ vector of response variables, $\mathbf{w}$ is a d dimensional weight vector, b is a scalar offset term, and λ is a regularization tuning parameter. For the programming part of this homework, you are required to implement the coordinate descent method that can solve the Lasso problem.

This question contains three parts, 1) analyze and optimize the time complexity 2) test your code using toy examples 3) try your code on a real world dataset. The first part involves no programming, but is crucial for writing an efficient algorithm.

4.1 Time complexity and making your code fast [12 points]

In class, you derived the update rules for coordinate descent, which is shown in Algorithm 1 (including the update term for b). In this part of question, we will analyze the algorithm and discuss how to make it fast. There are two key points: utilizing sparsity and caching your predictions. Assume we are using a sparse matrix representation, so that a dot product takes time proportional to the number of non-zero entries. In the following questions, your answers should take advantage of the sparsity of $\mathbf{X}$ when possible.

1. *(2 points)* Define the residual $r_i = y_i - (\mathbf{w}^T\mathbf{x}_i + b)$. Write the rule to compute $\mathbf{r}$ using matrix notation (no for loop). Let $\|\mathbf{X}\|_0$ be the number of nonzero entries in $\mathbf{X}$. What is the time complexity to compute $\mathbf{r}$?

2. *(2 points)* Simplify the update rule for b using $\mathbf{r}$. Your solution should use matrix notation. There should no longer be a for loop to sum over i. What is the time complexity when $\mathbf{r}$ is already computed.

Algorithm 1 Coordinate Descent Algorithm for Lasso

while not converged **do**

$b \leftarrow \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)$

for $k \in \{1, 2, \cdots, d\}$ **do**

$a_k \leftarrow 2\sum_{i=1}^{n} x_{ik}^2$

$c_k \leftarrow 2\sum_{i=1}^{n} x_{ik}\left(y_i - (b + \sum_{j\neq k} w_j x_{ij})\right)$

$$w_k \leftarrow \begin{cases} (c_k + \lambda)/a_k & c_k < -\lambda \\ 0 & c_k \in [-\lambda, \lambda] \\ (c_k - \lambda)/a_k & c > \lambda \end{cases}$$

end for

end while

Algorithm 2 Efficient Coordinate Descent Algorithm

$a_k \leftarrow 2\sum_{i=1}^{n} x_{ik}^2$ *% You can compute* $\mathbf{a}$ *without for loop*

while not converged **do**

1. Compute the residual vector $\mathbf{r}$ (Q1 of 4.1, recalculate $\mathbf{r}$ each iteration to avoid numerical drift)
2. Update b (Q2 of 4.1)
3. Update $\mathbf{r}$ (Q3 of 4.1)

for $k \in \{1, 2, \cdots, d\}$ **do**

4. Calculate c_k (Q4 of 4.1)
5. Update w_k (See Algorithm 1)
6. Update the residual $\mathbf{r}$ (Q5 of 4.1)

end for

end while

3. *(2 points)* Once b is updated, find the update rule for the residual vector $\mathbf{r}$. Again, use no for loop. What is the time complexity for the update?

4. *(2 points)* Simplify the computation for c_k in Algorithm 1 using the residual vector $\mathbf{r}$. Hint: there should no longer a sum over j. Suppose z_k is the number of non-zeros in the k^{th} row of $\mathbf{X}$, what is the time complexity to compute c_k when $\mathbf{r}$ is already computed?

5. *(2 points)* Suppose we update w_k from w_k^{old} to w_k^{new}. Derive an update rule for $\mathbf{r}$. Express the time complexity in terms of z_k.

6. *(2 points)* Putting this all together, you get the efficient coordinate descent algorithm in Algorithm 2. What is per iteration complexity of this algorithm (cost of each round of the while loop)?

4.2 Implement coordinate descent to solve the Lasso

Now we are ready to implement the coordinate descent algorithm in Algorithm 2. Your code must

- Include an offset term b that is not regularized
- Take optional initial conditions for $\mathbf{w}$.
- Be able to handle sparse $\mathbf{X}$. It is ok for your code not being able to handle dense $\mathbf{X}$. In Python, this means you can always assume input is a sparse matrix.
- Avoid unnecessary computation (i.e take advantage of what you learned from Problem 4.1)

You may use any language for your implementation, but we recommend Matlab. Matlab is a very useful language, and you should find that Matlab achieves reasonable performance for this problem. Our implementation takes $< 1s$ for an input matrix X of size 2500×10000. Matlab has many toolboxes, and you cannot use any existing Lasso solver.

Before you get started, here are some hints that you may find helpful:

- The only matrix operations required are: multiplying a matrix and a vector, adding vectors, computing dot product between two vectors, point-wise matrix operations (e.g., power of 2), calculate the sum of columns or rows or a matrix. Try to use as much vector/matrix computation as possible.
- The most important check is to ensure the objective value is non-increasing with each step.
- To ensure that a solution $(\hat{\mathbf{w}}, \hat{b})$ is correct, you also can compute the value

$$2\mathbf{X}(\mathbf{X}^T\hat{\mathbf{w}} + \hat{b} - \mathbf{y})$$

  This is a d-dimensional vector that should take the value $-\lambda \text{sign}(w_j)$ at j for each w_j that is nonzero. For the zero indices of $\hat{\mathbf{w}}$, this vector should take values lesser in magnitude than λ. (This is similar to setting the gradient to zero, though more complicated because the objective function is not differentiable.)
- It is up to you to decide on a suitable stopping condition. A common criteria is to stop when the objective value does not decrease by more than some small δ during an iteration. If you need your algorithm to run faster, an easy place to start is to loosen this condition.
- For several problems, you will need to solve the Lasso on the same dataset for many values of λ. This is called a regularization path. One way to do this efficiently is to start at a large λ, and then for each consecutive solution, initialize the algorithm with the previous solution, decreasing λ by a constant ratio until finished.
- The smallest value of λ for which the solution $\hat{\mathbf{w}}$ is entirely zero is given by

$$\lambda_{max} = 2\left\|\mathbf{X}\left(\mathbf{y} - \frac{1}{n}\sum_i y_i\right)\right\|_\infty$$

  This is helpful for choosing the first λ in a regularization path.

Finally here are some pointers toward useful parts of Matlab

- `[I,J,S] = find(X)`. Finding the row indices, column indices, and non-zero values of a sparse matrix $\mathbf{X}$.
- Use `sum(A, dim)` to compute the sum of row or column of a matrix $\mathbf{A}$.

4.3 Try out your work on synthetic data [12 points]

We will now try out your solver with some synthetic data. A benefit of the Lasso is that if we believe many features are irrelevant for predicting $\mathbf{y}$, the Lasso can be used to enforce a sparse solution, effectively differentiating between the relevant and irrelevant features.

Let's see if it actually works. Suppose that $\mathbf{x} \in \Re^d, y \in \Re, k < d$, and pairs of data $(\mathbf{x}_i, y_i)$ are generated independently according to the model

$$y_i = (\mathbf{w}^*)^T\mathbf{x}_i + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is some Gaussian noise. Note that since $k < d$, the features $k+1$ through d are unnecessary (and potentially even harmful) for predicting y.

With this model in mind, you are now tasked with the following:

1. *(8 points)* Let $N = 250, d = 80, k = 10$, and $\sigma = 1$. Generate some data by drawing each element of $\mathbf{X} \in \Re^{d \times n}$ from a standard normal distribution $\mathcal{N}(0, 1)$. Let $b^* = 0$ and create a $\mathbf{w}^*$ by setting the first k elements to ± 10 (choose any sign pattern) and the remaining elements to 0. Finally, generate a Gaussian noise vector ϵ with variance σ^2 and form $\mathbf{y} = \mathbf{X}^T \mathbf{w}^* + b^* + \epsilon$.

   With your synthetic data, solve multiple Lasso problems on a regularization path, starting at λ_{max} and decreasing λ by a constant ratio of 2 for 10 steps. Compare the sparsity pattern of your Lasso solution $\hat{\mathbf{w}}$ to that of the true model parameters $\mathbf{w}^*$. Record values for precision (number of correct nonzeros in $\hat{\mathbf{w}}$/total number of nonzeros in $\hat{\mathbf{w}}$) and recall (number of correct nonzeros in $\hat{\mathbf{w}}$/k).

   How well are you able to discover the true nonzeros? Comment on how λ affects these results and include plots of precision and recall versus λ.

2. *(4 points)* Change σ to 10, regenerate the data, and solve the Lasso problem using a value of λ that worked well when $\sigma = 1$. How are precision and recall affected? How might you change λ in order to achieve better precision or recall?

4.4 Predicting goodness points of a wine given its review [16 points + 10 bonus]

We'll now put the Lasso to work on some real data. For this homework, the task is to predict the points the wine should get based on its review text alone.

For many Kaggle competitions (and machine learning methods in general), one of the most important requirements for doing well is the ability to discover great features. We can use our Lasso solver for this as follows. First, generate a large amount of features from the data, even if many of them are likely unnecessary. Afterward, use the Lasso to reduce the number of features to a more reasonable amount.

The data was scraped from WineEnthusiast during the week of June 15th, 2017 and can be found on the analytics website Kaggle (`https://www.kaggle.com/zynicide/wine-reviews`). As a side note, browsing other competitions on the site may also give you some ideas for class projects.) This dataset provides a variety of features, the *points*, *description (review)*, *variety*, *country*, *province* etc. For this homework we will be using the points and description features. Points are ratings within range 1-100. However, the dataset contains description for wines rated in range 80-100. We already provide you the feature vectors for each review. You can download the data for this homework from- `https://www.kaggle.com/c/hw1-wine-goodness-cse512-spr18/data`.

`trainData.txt`	Training data matrix for predicting number of stars
`trainLabels.txt`	Training labels, list of the number of stars for each review
`valData.txt`	Validation data
`valLabels.txt`	Validation labels
`testData.txt`	Test data
`featureTypes.txt`	List of features used

The data matrices are stored in Matrix Market Format, a format for sparse matrices. Each line is is a tuple of three elements for instance ID, feature ID, and feature value. Meta information for each feature is provided in `featureTypes.txt`. The features are strings of one, two, or three words (n-grams). The feature values correspond roughly to how often each word appears in the review. All rows have also been normalized.

In Matlab, you can use the following code to load train data and labels:

```
D = load('trainData.txt');
trX = sparse(D(:,2), D(:,1), D(:,3));
trLb = load('trainLabels.txt');
```

For this part of the problem, you have the following tasks:

1. *(8 points)* Solve Lasso to predict the number of points a Wine will receive. Starting at λ_{max}, run Lasso on the training set, decreasing λ by a factor of 2, using previous solutions as initial conditions to each problem. Stop when you have considered enough λ's that, based on validation error, you can choose a good solution with confidence (for instance, when validation error begins increasing or stops decreasing significant). At each solution, record the root-mean-squared-error (RMSE) on training and validation data. In addition, record the number of nonzeros in each solution.

   Plot the train and validation RMSE values together on a plot against λ. Separately plot the number of nonzeros as well.

2. *(5 points)* Find the λ that achieves best validation performance. Compute a model using this λ and list the top 10 features with largest weights and the top 10 features with lowest weights. Comment if the weights make sense intuitively.

3. *(3 points)* Use your model to predict the points for the reviews in test data. Save the predicted values on a CSV file `predTestLabels.csv` (see 5.2 for the format). The order of predicted values should correspond to the order of the reviews in `testData.txt` (counting starts from 1). Submit this `predTestLabels.txt` file on Kaggle and report the RMSE.

4. *(10 bonus points)* Submit `predTestLabels.csv` and enter our Kaggle competition for fame. You must use your Stony Brook email to register on Kaggle. We will maintain a leader board, and the top three entries at the end of the competition (due date) will receive 10 bonus points. The ranking is based on RMSE.

   To prevent exploiting test data, you are allowed to make a maximum of 2 submissions per 24 hours. Your submission will be evaluated immediately and the leader board will be updated.

   You are allowed to create new features from the existing set of features. You are allowed to perform feature normalization on existing and new features. You can use both training and validation data to train your classifier. You can also use Ridge regression instead of Lasso. If you use Ridge regression, you must implement it yourself. You cannot use any other classifier types such as SVM or boosting.

5 What to submit?

5.1 Blackboard submission

You will need to submit both your code and your answers to questions on Blackboard. Do not submit the provided data. Put the answer file and your code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_lionel_messi*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SUBID_FirstName_LastName.zip. The answer file should be named: answers.pdf, and it should contain:

1. Answers to Question 1, 2, 3, and 4.1
2. Answers to Question 4.3, including the requested plots.
3. Answers to Question 4.4.1, and 4.4.2, including the requested plots.

5.2 Kaggle submission

For Questions 4.4.3 and 4.4.4, you must submit a CSV file to get the RMSE from the competition site `https://www.kaggle.com/c/hw1-wine-goodness-cse512-spr18`. A submission file should

contain two columns: instanceId and Prediction. The file should contain a header and have the following format.

$$\begin{array}{cc} instanceId, & Prediction \\ 1, & 82 \\ 2, & 91 \\ \ldots & \ldots \end{array} \tag{5}$$

A sample submission file is available from the competition site and our handout.

Cheating warnings: Don't cheat. You must do the homework yourself, otherwise you won't learn. You cannot ask and discuss with students from previous year. You cannot look up the solution online. You must use your real name to register on Kaggle. Do not create multiple accounts to bypass the submission limitation per 24 hours. Doing so will be considered cheating.