CSE446 Machine Learning, Winter 2017: Homework 1

Due: Monday, January 23^{rd}, beginning of class

Instructions There are 4 written questions on this assignment, plus a fifth coding question. Submit both your written answers (as a pdf) and your implementation to the Dropbox at `https://catalyst.uw.edu/collectit/dropbox/summary/qaz2wsx3/39488`. Please show your work for all questions.

Your written answers may be typed, or you may scan in hand-written answers. You're welcome to type your solutions in LaTex if you know how. If you don't know LaTex but want to type, there a number of markdown editors with real-time preview and equation editing. Here are two: `https://stackedit.io/`, `http://marxi.co/`. Writing your solutions by hand is also fine as long as they're neat.

You are welcome to use any Python libraries for data munging, visualization, and numerical linear algebra. Examples includes Numpy, Pandas, and Matplotlib. You may NOT, however, use any Python machine learning libraries such as Scikit-Learn or TensorFlow. If in doubt, email the instructors.

Please note also that all references to Murphy refer to the fourth printing of the textbook. Page and section numbers may differ between additions.

1 Probability and Bayes' Rule [10 points]

(Source: Murphy exercise 2.4.) After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a serious disease, and that the test is 99% accurate (i.e., the probability of testing positive given that you have the disease is 0.99, as is the probability of testing negative given that you dont have the disease). The good news is that this is a rare disease, striking only one in 10,000 people. What are the chances that you actually have the disease? (Show your calculations as well as giving the final result.)

2 MLE [20 points]

2.1 The Poisson distribution [12 points]

You're a Seahawks fan, and the team is six weeks into its season. The number touchdowns scored in each game so far are given below:

$$[1, 3, 3, 0, 1, 5].$$

Let's call these scores $x_1, \ldots, x_6$. Based on your data, you'd like to build a model to understand how many touchdowns the Seahaws are likely to score in their next game. You decide to model the number of touchdowns scored per game using a *Poisson distribution*. The Poisson distribution with parameter λ assigns every non-negative integer $x = 0, 1, 2, \ldots$ a probability given by

$$\text{Poi}(x|\lambda) = e^{-\lambda}\frac{\lambda^x}{x!}.$$

So, for example, if $\lambda = 1.5$, then the probability that the Seahawks score 2 touchdowns in their next game is $e^{-1.5} \times \frac{1.5^2}{2!} \approx 0.25$. To check your understanding of the Poisson, make sure you have a sense of whether raising λ will mean more touchdowns in general, or fewer.

1. *(8 points)* Derive an expression for the maximum-likelihood estimate of the parameter λ governing the Poisson distribution, in terms of your touchdown counts $x_1, \ldots, x_6$. (Hint: remember that the log of the likelihood has the same maximum as the likelihood function itself.)
2. *(4 points)* Given the touchdown counts, what is your numerical estimate of λ?

2.2 The Multinomial Distribution [8 points]

It is the first day of spring, and a gloomy 110-day Seattle winter has finally ended. Over those 110 days, $n_{clear} = 11$ of them were clear, $n_{cloudy} = 24$ of them were cloudy (but not rainy), and the remaining $n_{rain} = 75$ were rainy. Denote the probability that a given winter day next year will be clear, cloudy, and rainy by p_{clear}, p_{cloudy}, and p_{rainy}, respectively, and assume that next winter's weather will follow the same pattern as this year's. Please write down expressions for the maximum-likelihood estimates of p_{clear}, p_{cloudy}, and p_{rainy}, in terms of this winter's weather data.

3 Online Linear Regression [20 points]

(Source: Murphy exercise 7.7.) Consider fitting the model $\hat{y} = w_0 + w_1 x$ using least squares. Unfortunately we did not keep the original data, x_i, y_i, but we do have the following functions (statistics) of the data:

$$\bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$C_{xx}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}^{(n)})^2, \quad C_{xy}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}^{(n)})(y_i - \bar{y}^{(n)}), \quad C_{yy}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}^{(n)})^2.$$

Also, it may be useful for you to know that, when fitting the model $\hat{y} = w_0 + w_1 x$ in a batch (offline) fashion, the model weights w_1 and w_0 may be computed according to:

$$w_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \quad w_0 = \bar{y} - w_1 \bar{x}.$$

1. *(3 points)* What are the minimal set of statistics that we need to estimate w_1?
2. *(3 points)* What are the minimal set of statistics that we need to estimate w_o?
3. *(6 points)* Suppose a new data point, x_{n+1}, y_{n+1} arrives, and we want to update our sufficient statistics without looking at the old data, which we have not stored. (This is useful for online learning). Show that we can do this for $\bar{x}$ as follows.

   $$\bar{x}^{(n+1)} \triangleq \bar{x}^{(n)} + \frac{1}{n+1}\left(x_{n+1} - \bar{x}^{(n)}\right)$$

   This has the form: new estimate is old estimate plus correction. We see that the size of the correction diminishes over time (i.e., as we get more samples). Derive a similar expression to update $\bar{y}$.
4. *(6 points)* Show that one can update $C_{xy}^{(n+1)}$ recursively using

   $$C_{xy}^{(n+1)} = \frac{1}{n+1}\left[x_{n+1}y_{n+1} + nC_{xy}^{(n)} + n\bar{x}^{(n)}\bar{y}^{(n)} - (n+1)\bar{x}^{(n+1)}\bar{y}^{(n+1)}\right]$$

   Derive a similar expression to update C_{xx}.

5. *(2 points)* Give two examples of settings where it would be advantageous to perform regression online (rather than batch) using the updates you've derived.

4 Regularization Constants [15 points]

We have discussed the importance of regularization as a technique to avoid overfitting our models. For linear regression, we have mentioned both LASSO (which uses the L_1 norm as a penalty), and ridge regression (which uses the squared L_2 norm as a penalty). In practice, the scaling factor of these penalties has a significant impact on the behavior of these methods, and must often be chosen empirically for a particular dataset. In this problem, we look at what happens when we choose our regularization factor poorly.

4.1 LASSO regression [8 points]

Recall that the loss function to be optimized under LASSO regression is:

$$E_L = \sum_{i=1}^{n} (Y_i - (\hat{w}_0 + X_i\hat{w}))^2 + \lambda\|\hat{w}\|_1$$

where

$$\lambda\|\hat{w}\|_1 = \lambda \sum_{i=1}^{d} |\hat{w}_i| \tag{1}$$

and λ is our regularization constant.

1. Suppose our λ is much too small; that is,

$$\sum_{i=1}^{n} (Y_i - X\hat{w})^2 + \lambda\|w\|_1 \approx \sum_{i=1}^{n} (Y_i - X\hat{w})^2$$

   How will this affect the magnitude of:

   (a) *(1 point)* The error on the training set?

   (b) *(1 point)* The error on the testing set?

   (c) *(1 point)* $\hat{w}$?

   (d) *(1 point)* The number of nonzero elements of $\hat{w}$?

2. Suppose instead that we overestimated on our selection of λ. What do we expect to be the magnitude of:

   (a) *(1 point)* The error on the training set?

   (b) *(1 point)* The error on the testing set?

   (c) *(1 point)* $\hat{w}$?

   (d) *(1 point)* The number of nonzero elements of $\hat{w}$?

4.2 Ridge Regression [7 points]

Recall that the loss function to be optimized under ridge regression is now:

$$E_R = \sum_{i=1}^{n} (Y_i - (\hat{w}_0 + X_i \hat{w}))^2 + \lambda \|\hat{w}\|_2^2$$

where

$$\lambda \|\hat{w}\|_2^2 = \lambda \sum_{i=1}^{d} (\hat{w}_i)^2 \tag{2}$$

If λ is too small, then the misbehavior of ridge regression is similar to that of LASSO in the previous section. Let's suppose λ has been set too high, and compare the behavior of ridge regression to LASSO.

To make this comparison, let's look at what happens to a particular feature weight $\hat{w}_i$. Since $\hat{w}$ is the solution that minimizes our error function E, it must be the case that $\frac{\partial E}{\partial \hat{w}_i} = 0$.

1. *(2 points)* Suppose we use the LASSO loss function E_L as defined in the previous section. Let's ignore the first term (which corresponds to the Sum Squared Error of our prediction), and calculate the partial derivative of the penalty term in Equation 1 with respect to $\hat{w}_i$. This can be thought of as the direction that LASSO "pushes" us away from the SSE solution. *Note that the absolute value is not differentiable at $\hat{w}_i = 0$, so you only need to answer for the case that $\hat{w}_i \neq 0$.*
2. *(2 points)* Instead, suppose we use the ridge regression loss function E_R from above. Again, ignore the SSE term and calculate the partial derivative with respect to $\hat{w}_i$ of Equation 2 to find how ridge regression "pushes" us away from the SSE solution.
3. *(3 points)* When $|\hat{w}_i| < 1/2$, which method "pushes away" more strongly from the SSE solution: ridge or lasso? What about when $|\hat{w}_i| > 1/2$? Use these answers to compare and contrast the behaviors of ridge and lasso when λ has been set too high.

5 Programming Question [35 Points]

Download the training data set "crime-train.txt" and the test data set "crime-test.txt" from the website under Homework 1. Store your data in your working directory and read in the files with:

```
import pandas as pd
df_train = pd.read_table("crime-train.txt")
df_test = pd.read_table("crime-test.txt")
```

This stores the data as Pandas DataFrame objects. DataFrames are similar to Numpy arrays but more flexible; unlike Numpy arrays, they store row and column indices along with the values of the data. Each column of a DataFrame can also, in principle, store data of a different type. For this assignment, however, all data are floats. Here are a few commands that will get you working with Pandas for this assignment:

```
df.head()                    # Print the first few lines of DataFrame df.
df.index                     # Get the row indices for df.
df.columns                   # Get the column indices.
df[''foo'']                  # Return the column named ''foo''.
df.drop(''foo'', axis = 1)   # Return all columns except ''foo''.
df.values                    # Return the values as a Numpy array.
df[''foo''].values           # Grab column foo and convert to Numpy array.
df.iloc[:3,:3]               # Use numerical indices (like Numpy) to get 3 rows and cols.
```

The data consist of local crime statistics for 1,994 US communities. The response y is the crime rate. The name of the response variable is `ViolentCrimesPerPop`, and it is held in the first column of `df_train` and `df_test`. There are 95 features x_i. These features include possibly relevant variables such as the size of the police force or the percentage of children that graduate high school. The data have been split for you into a training and test set with 1,595 and 399 entries, respectively[1].

We'd like to use the training set to fit a model which can predict the crime rate in new communities, and evaluate model performance on the test set. As there are a considerable number of input variables, overfitting is a serious issue. In order to avoid this, implement the coordinate descent LASSO algorithm. This is algorithm 13.1 of Murphy, found on p.443. [2] .

Your function should accept a scalar value of λ, a vector-valued response variable ($\mathbf{y}$), a matrix of input variables ($\mathbf{X}$), and an initial vector of weights ($\mathbf{w}_0$). It should output a vector of coefficient values ($\hat{\mathbf{w}}$).

Define "converging" as the change in any coefficient between one iteration and the next is no larger than 10^{-6}. (That is, $||\hat{\mathbf{w}}_{old} - \hat{\mathbf{w}}_{new}||_\infty < 10^{-6}$)

Once you have implemented a LASSO solver function, begin by running the solver with $\lambda = 600$. For the initial weights, just use $\mathcal{N}(0, 1)$ random variables. Then, cut λ down by a factor of 2 and run again, but this time pass in the values of $\hat{\mathbf{w}}$ from your $\lambda = 600$ solution as your initial weights. This is faster than initializing with random weights. Continue the process of cutting λ by a factor of 2 until you have models for 10 values of λ in total.

In your analysis, include:

1. *(10 points)* All of your code
2. *(10 points)* The regularization paths (in one plot) for the coefficients for input variables agePct12t29, pctWSocSec, PctKids2Par, PctIlleg, and HousVacant — use $\log(\lambda)$ instead of λ.
3. *(4 points)* A plot of $\log(\lambda)$ against the squared error in the training data.
4. *(4 points)* A plot of $\log(\lambda)$ against the squared error in the test data.
5. *(3 points)* A plot of λ against the number of nonzero coefficients.
6. *(2 points)* A brief commentary on the task of selecting λ
7. *(2 points)* For the λ that gave the best test set performance, which variable had the largest (most positive) Lasso coefficient? What about the most negative? Discuss briefly. A description of the variables in the data set can be found here: `http://archive.ics.uci.edu/ml/machine-learning-databases/communities/communities.names`.

For your plots, you'll likely want to use Pyplot as covered in the introductory Python tutorial.

Submit your work: please compress your python scripts into a zip file, with file name `First_Last_hw1_p5.zip`. Please include the plots into your pdf file.

[1] The features have been standardized to have mean 0 and variance 1.

[2] In the text Murphy discusses both a hard and a soft threshold, use the soft threshold as described in the referenced algorithm