Homework #2

CSE 446: Machine Learning
Prof. Byron Boots
Due: **Tuesday** February 16, 2021 11:59 PM

Please review all homework guidance posted on the website before submitting to Gradescope. Reminders:

- Please provide succinct answers along with succinct reasoning for all your answers. Points may be deducted if long answers demonstrate a lack of clarity. Similarly, when discussing the experimental results, concisely create tables and/or figures when appropriate to organize the experimental results. In other words, all your explanations, tables, and figures for any particular part of a question must be grouped together.
- For every problem involving generating plots, please include the plots as part of your PDF submission.
- When submitting to Gradescope, please link each question from the homework in Gradescope to the location of its answer in your homework PDF. Failure to do so may result in point deductions. For instructions, see `https://www.gradescope.com/get_started#student-submission`.
- If you collaborate on this homework with others, you must indicate who you worked with on your homework. Failure to do so may result in accusations of plagiarism.

Conceptual Questions

Problem 0. The answers to these questions should be answerable without referring to external materials. Briefly justify your answers with a few words.

a. *[2 points]* Suppose that your estimated model for predicting house prices has a large positive weight on number of bathrooms. Does this imply that if we remove the feature number of bathrooms and refit the model, the new predictions will be strictly worse than before? Why?

b. *[2 points]* Compared to L2 norm penalty, explain why L1 norm penalty is more likely to result in a larger number of zeros in the weight vector.

c. *[2 points]* In at most one sentence each, state one possible advantage and one possible disadvantage of using the following regularizer: $\left(\sum_i |w_i|^{0.5}\right)$

d. *[2 points]* Briefly explain why the estimated true error from k-fold cross validation is biased. Why might k=10 be a reasonable choice for k?

e. *[1 points]* True or False: If the step-size for gradient descent (GD) is too large, it may not converge.

f. *[2 points]* In your own words, describe why stochastic gradient descent (SGD) works.

g. *[2 points]* In at most one sentence each, state one possible advantage of SGD over GD and one possible disadvantage.

Convexity and Norms

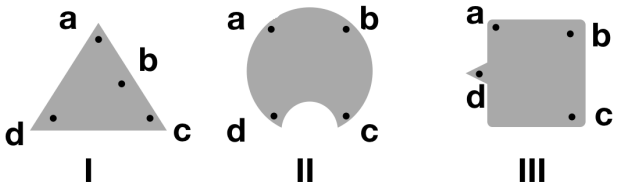
Problem 1. A *norm* $\|\cdot\|$ over $\mathbb{R}^n$ is defined by the properties: i) non-negative: $\|x\| \geq 0$ for all $x \in \mathbb{R}^n$ with equality if and only if $x = 0$, ii) absolute scalability: $\|a\,x\| = |a|\,\|x\|$ for all $a \in \mathbb{R}$ and $x \in \mathbb{R}^n$, iii) triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

a. *[3 points]* Show that $f(x) = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$ is a norm. (Hint: You are allowed to use the Cauchy-Schwarz inequality.)

b. *[2 points]* Show that $g(x) = \left(\sum_{i=1}^n |x_i|^{1/3}\right)^3$ is not a norm. (Hint: it suffices to find two points in $n = 2$ dimensions such that the triangle inequality does not hold.)

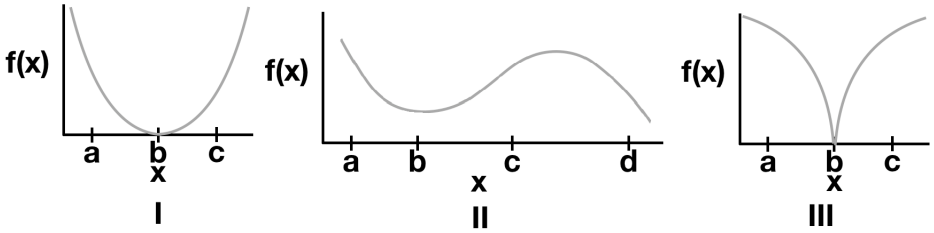
Context: norms are often used in regularization to encourage specific behaviors of solutions. If we define $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$ then one can show that $\|x\|_p$ is a norm for all $p \geq 1$. The important cases of $p = 2$ and $p = 1$ correspond to the penalty for ridge regression and the lasso, respectively.

Problem 2. *[3 points]* A set $A \subseteq \mathbb{R}^n$ is *convex* if $\lambda x + (1-\lambda)y \in A$ for all $x, y \in A$ and $\lambda \in [0, 1]$.



For each of the grey-shaded sets above (I-III), state whether each one is convex, or state why it is not convex using any of the points a, b, c, d in your answer.

Problem 3. *[4 points]* We say a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex on a set A if $f(\lambda x+(1-\lambda)y) \leq \lambda f(x)+(1-\lambda)f(y)$ for all $x, y \in A$ and $\lambda \in [0, 1]$.



For each of the grey-colored functions below (I-III), state whether each one is convex on the given interval or state why not with a counterexample using any of the points a, b, c, d in your answer.

a. Function in panel I on $[a, c]$

b. Function in panel II on $[a, d]$

c. Function in panel II on $[a, b]$

d. Function in panel III on $[a, c]$

Problem 4. *[5 points]* Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. Show that the function $f(x) = (x^T Ax)^{1/2}$ is convex. Hint: First show that $f(x)$ is a norm, then show that all norms are convex functions. To show that $f(x)$ is a norm, you may use the spectral theorem. The spectral theorem states that any symmetric real matrix A can be written in the form $A = V\Lambda V^T$, where $V, \Lambda \in \mathbb{R}^{n\times n}$, Λ is a diagonal matrix, and $V^T V = I$ (i.e. V is an orthonormal matrix). Show that $f(x) = \|\Lambda^{1/2} V^T x\|_2$ and invoke problem 1a above. What does the notation "$\Lambda^{1/2}$" mean and how do we know it exists?

Lasso on a real dataset

Given $\lambda > 0$ and data $(x_1, y_1), \ldots, (x_n, y_n)$, the Lasso is the problem of solving

$$\arg \min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \sum_{i=1}^n (x_i^T w + b - y_i)^2 + \lambda \sum_{j=1}^d |w_j|$$

λ is a regularization tuning parameter. For the programming part of this homework, you are required to implement the coordinate descent method of Algorithm 1 that can solve the Lasso problem.
You may use common computing packages (such as NumPy or SciPy), but do not use an existing Lasso solver (e.g., of scikit-learn).
Before you get started, here are some hints that you may find helpful:

- For-loops can be slow whereas vector/matrix computation in Numpy is very optimized; exploit this as much as possible.
- The pseudocode provided has many opportunities to speed up computation by precomputing quantities like a_k before the for loop. These small changes can speed things up considerably.
- As a sanity check, ensure the objective value is nonincreasing with each step.
- It is up to you to decide on a suitable stopping condition. A common criteria is to stop when no element of w changes 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.
- 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 (e.g., by a factor of 2) until finished.
- The smallest value of λ for which the solution $\widehat{w}$ is entirely zero is given by

$$\lambda_{max} = \max_{k=1,\ldots,d} 2\left|\sum_{i=1}^{n} x_{i,k}\left(y_i - \left(\frac{1}{n}\sum_{j=1}^{n} y_j\right)\right)\right|$$

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

Problem 5. We will first try out your solver with some synthetic data. A benefit of the Lasso is that if we believe many features are irrelevant for predicting y, the Lasso can be used to enforce a sparse solution, effectively differentiating between the relevant and irrelevant features. Suppose that $x \in \mathbb{R}^d, y \in \mathbb{R}, k < d$, and pairs of data (x_i, y_i) for $i = 1, \ldots, n$ are generated independently according to the model $y_i = w^T x_i + \epsilon_i$ where

$$w_j = \begin{cases} j/k & \text{if } j \in \{1, \ldots, k\} \\ 0 & \text{otherwise} \end{cases}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is some Gaussian noise (in the model above $b = 0$). Note that since $k < d$ and $w_j = 0$ for $j > k$, the features $k+1$ through d are unnecessary (and potentially even harmful) for predicting y.
With this model in mind, let $n = 500, d = 1000, k = 100$, and $\sigma = 1$. Generate some data by choosing $x_i \in \mathbb{R}^d$, where each component is drawn from a $\mathcal{N}(0, 1)$ distribution and y_i generated as specified above.

Algorithm 1: Coordinate Descent Algorithm for Lasso

while *not converged* **do**
  $b \leftarrow \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} w_j x_{i,j}\right)$
  for $k \in \{1, 2, \cdots d\}$ **do**
    $a_k \leftarrow 2\sum_{i=1}^{n} x_{i,k}^2$
    $c_k \leftarrow 2\sum_{i=1}^{n} x_{i,k}\left(y_i - (b + \sum_{j \neq k} w_j x_{i,j})\right)$
    $w_k \leftarrow \begin{cases} (c_k + \lambda)/a_k & c_k < -\lambda \\ 0 & c_k \in [-\lambda, \lambda] \\ (c_k - \lambda)/a_k & c_k > \lambda \end{cases}$
  end
end

a. *[10 points]* With your synthetic data, solve multiple Lasso problems on a regularization path, starting at λ_{max} where 0 features are selected and decreasing λ by a constant ratio (e.g., 1.5) until nearly all the features are chosen. In plot 1, plot the number of non-zeros as a function of λ on the x-axis (Tip: use `plt.xscale('log')`).

b. *[10 points]* For each value of λ tried, record values for false discovery rate (FDR) (number of incorrect nonzeros in $\widehat{w}$ [1] /total number of nonzeros in $\widehat{w}$) and true positive rate (TPR) (number of correct nonzeros in $\widehat{w}$/k). In plot 2, plot these values with the x-axis as FDR, and the y-axis as TPR and note that in an ideal situation we would have an (FDR,TPR) pair in the upper left corner, but that can always trivially achieve $(0, 0)$ and $(\frac{d-k}{d}, 1)$.

c. *[5 points]* Comment on the effect of λ in these two plots.

Problem 6. We'll now put the Lasso to work on some real data. Download the training data set "crime-train.txt" and the test data set "crime-test.txt" from the website under Homework 2. 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 rate of violent crimes reported per capita in a community. 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. These features include many variables. Some features are the consequence of complex political processes, such as the size of the police force. Others are demographic characteristics of the community, including self-reported statistics about race, age, education, and employment drawn from Census reports.
The goals of this problem are twofold: first, to encourage you to think deeply about models you might train and how they might be misused; and second, to see how Lasso encourages sparsity of linear models in settings where the feature set is very large relative to the number of training examples. **We emphasize that training a model on this dataset can suggest a degree of correlation between a community's demographics and the rate at which a community experiences and reports violent crime.** We strongly encourage students to consider why these correlations may or may not hold more generally, whether correlations might result from a common cause, and what issues can result in misinterpreting what a model can explain.

We have split the dataset into a training and test set with 1,595 and 399 entries, respectively[2]. We'd like to use this training set to fit a linear model to predict the crime rate in new communities and evaluate model performance on the test set. As there are a considerable number of input variables and fairly few training datapoints, overfitting is a serious issue. In order to avoid this, use the coordinate descent LASSO algorithm you just implemented in the previous problem.

[1] For each j, $\widehat{w}_j$ is an incorrect nonzero if and only if $\widehat{w}_j \neq 0$ while $w_j = 0$
[2] The features have been standardized to have mean 0 and variance 1.

a. *[4 points]* Begin by reading the documentation for the original version of this dataset: `http://archive.ics.uci.edu/ml/datasets/communities+and+crime`. Report 3 features included in this dataset for which historical *policy* choices in the US would lead to variability in these features. As an example, the *number of police* in a community is often the consequence of decisions made by governing bodies, elections, and amount of tax revenue available to decisionmakers.

b. *[4 points]* Before you train a model for this prediction task, describe 3 features in the dataset which might, if found to have nonzero weight in model, be interpreted as *reasons* for higher levels of violent crime, but which might actually be a *result* rather than (or in addition to being) the cause of this violence.

Now, we will run the LASSO solver with $\lambda = \lambda_{\max}$ defined above. For the initial weights, just use 0. Then, cut λ down by a factor of 2 and run again, but this time pass in the values of $\hat{w}$ from your $\lambda = \lambda_{\max}$ solution as your initial weights. This is faster than initializing with 0 weights each time. Continue the process of cutting λ by a factor of 2 until the smallest value of λ is less than 0.01. For all plots use a log-scale for the λ axis (Tip: use `plt.xscale('log')`).

a. *[4 points]* Plot the number of nonzeros of each solution versus λ.

b. *[4 points]* Plot the regularization paths (in one plot) for the coefficients for input variables agePct12t29, pctWSocSec, pctUrban, agePct65up, and householdsize.

c. *[4 points]* On one plot, plot the squared error on the training and test data versus λ.

d. *[4 points]* Sometimes a larger value of λ performs nearly as well as a smaller value, but a larger value will select fewer variables and perhaps be more interpretable. Inspect the weights (on features) for $\lambda = 30$. Which feature variable had the largest (most positive) Lasso coefficient? What about the most negative? Discuss briefly.

e. *[4 points]* Suppose there was a large negative weight on agePct65up and upon seeing this result, a politician suggests policies that encourage people over the age of 65 to move to high crime areas in an effort to reduce crime. What is the (statistical) flaw in this line of reasoning? (Hint: fire trucks are often seen around burning buildings, do fire trucks cause fire?)

Logistic Regression

Binary Logistic Regression

Problem 7. Let us again consider the MNIST dataset, but now just binary classification, specifically, recognizing if a digit is a 2 or 7. Here, let $Y = 1$ for all the 7's digits in the dataset, and use $Y = -1$ for 2. We will use regularized logistic regression. Given a binary classification dataset $\{(x_i, y_i)\}_{i=1}^n$ for $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ we showed in class that the regularized negative log likelihood objective function can be written as

$$J(w, b) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i(b + x_i^T w))) + \lambda ||w||_2^2$$

Note that the offset term b is not regularized. For all experiments, use $\lambda = 10^{-1}$. Let $\mu_i(w, b) = \frac{1}{1+\exp(-y_i(b+x_i^T w))}$.

a. *[8 points]* Derive the gradients $\nabla_w J(w, b)$, $\nabla_b J(w, b)$ and give your answers in terms of $\mu_i(w, b)$ (your answers should not contain exponentials).

b. *[8 points]* Implement gradient descent with an initial iterate of all zeros. Try several values of step sizes to find one that appears to make convergence on the training set as fast as possible. Run until you feel you are near to convergence.

   (i) For both the training set and the test, plot $J(w, b)$ as a function of the iteration number (and show both curves on the same plot).

(ii) For both the training set and the test, classify the points according to the rule $\text{sign}(b + x_i^T w)$ and plot the misclassification error as a function of the iteration number (and show both curves on the same plot).

Note that you are only optimizing on the training set. The $J(w, b)$ and misclassification error plots should be on separate plots.

c. *[7 points]* Repeat (b) using stochastic gradient descent with a batch size of 1. Note, the expected gradient with respect to the random selection should be equal to the gradient found in part (a). Take careful note of how to scale the regularizer.

d. *[7 points]* Repeat (b) using stochastic gradient descent with batch size of 100. That is, instead of approximating the gradient with a single example, use 100. Note, the expected gradient with respect to the random selection should be equal to the gradient found in part (a).