Homework #2

CSE 446/546: Machine Learning
Professors Matt Golub & Pang Wei Koh
Due: **Monday** November 04, 2024 11:59pm
A: 108 points, **B**: 20 points

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

- All code must be written in Python and all written work must be typeset (e.g. LaTeX).
- Make sure to read the "What to Submit" section following each question and include all items.
- Please provide succinct answers and supporting reasoning for each question. Similarly, when discussing experimental results, concisely create tables and/or figures when appropriate to organize the experimental results. All 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 deductions of up to 10% of the value of each question not properly linked. For instructions, see `https://www.gradescope.com/get_started#student-submission`.

Not adhering to these reminders may result in point deductions.

Important: By turning in this assignment (and all that follow), you acknowledge that you have read and understood the collaboration policy with humans and AI assistants alike: `https://courses.cs.washington.edu/courses/cse446/24au/assignments/`. Any questions about the policy should be raised at least 24 hours before the assignment is due. There are no warnings or second chances. If we suspect you have violated the collaboration policy, we will report it to the college of engineering who will complete an investigation.
Not adhering to these reminders may result in point deductions.

Conceptual Questions

A1. 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 the feature `number of bathrooms`. If we remove this feature and refit the model, will the new model have a strictly higher error than before? Why?

> Solution:
> If we remove this feature and refit the model, the new model will not necessarily have a strictly higher error than before because the magnitude of a feature's weighting does not indicate its importance to the model.

b. *[2 points]* Compared to L2 norm penalty, explain why a L1 norm penalty is more likely to result in sparsity (a larger number of 0s) in the weight vector.

> Solution:
> A L1 norm penalty is more likely to result in sparsity compared to a L2 norm penalty in the weight vector due to the geometry of the L1 norm that consists of absolute value functions, giving it a spiky shape. These sharp edges on the constraint region encourage solutions that "land" directly on the axes, making some of the weights exactly zero.

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

> Solution:
> Upside: The regularizer encourages sparse solutions, more so than a L1 regularizer.
> Downside: The regularizer is nonconvex - optimization is computationally intensive.

d. *[1 point]* True or False: If the step-size for gradient descent is too large, it may not converge.

> Solution:
> True

e. *[2 points]* In your own words, describe why stochastic gradient descent (SGD) works, even though only a small portion of the data is considered at each update.

> Solution:
> Stochastic Gradient Descent works even though only a small portion of the data is considered at each update because the cumulative effect of these small updates over many iterations can still lead to convergence to a good solution. Also, the nature of SGD, uniformly picking points at random, allows for it to avoid local minima and navigate the parameter space more effectively.

f. *[2 points]* In at most one sentence each, state one possible advantage of SGD over GD (gradient descent), and one possible disadvantage of SGD relative to GD.

> Solution:
> Advantage: SGD is less computationally intensive than GD as you are only calculating the gradient at a point rather than across every point
> Disadvantage: The convergence path of SGD is noisier than that of original gradient descent due to its stochastic nature.

What to Submit:

- **Part d:** True or False.
- **Parts a-f:** Brief (2-3 sentence) explanation.

Convexity and Norms

A2. A *norm* $\|\cdot\|$ over $\mathbb{R}^n$ is defined by the properties: (i) non-negativity: $\|x\| \geq 0$ 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 $f(x) = (\sum_{i=1}^n |x_i|)$ is a norm. (Hint: for (iii), begin by showing that $|a+b| \leq |a|+|b|$ for all $a, b \in \mathbb{R}$.)

> Solution: We will prove f(x) is a norm by looking at each of the properties of a norm $\|\cdot\|$ over $\mathbb{R}^n$.
>
> i Since $|x_i| \geq 0$ for all $x_i \in R$, $f(x) = (\sum_{i=1}^n |x_i|) \geq 0$. Thus, $f(x)$ fulfills the non-negativity property of a norm.
>
> ii $f(ax) = \sum_{i=1}^n |ax_i| = \sum_{i=1}^n |a||x_i| = |a|f(x)$. Thus, $f(x)$ fulfills the absolute scalability property of a norm.
>
> iii Since $x + y \leq |x| + |y| \rightarrow |x + y| \leq |x| + |y|$ for all x, y $\in R$, $f(x + y) = \sum_{i=1}^n |x_i + y_i| \leq \sum_{i=1}^n |x_i| + \sum_{i=1}^n |y_i| = f(x) + f(y)$. Thus, $f(x)$ fulfills the triangle inequality property of a norm.
>
> Altogether, since f(x) meets all three properties of a norm, we can conclude f(x) is a norm.

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

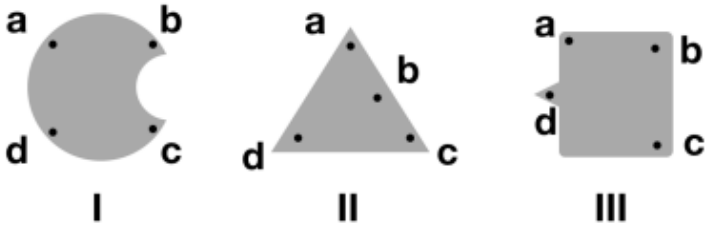
> Solution: We will prove g(x) is not a norm by looking at each of the properties of a norm $\|\cdot\|$ over $\mathbb{R}^n$.
>
> i Since a^2 for all $a \in R$, $g(x) = (\sum_{i=1}^n |x_i|^{1/2})^2 \geq 0$. Thus, $g(x)$ fulfills the non-negativity property of a norm.
>
> ii $g(ax) = (\sum_{i=1}^n |ax_i|^{1/2})^2 = \sum_{i=1}^n (|a|^1/2|x_i|^{1/2})^2 = \sum_{i=1}^n |a|(|x_i|^{1/2})^2 = |a|g(x)$. Thus, $g(x)$ fulfills the absolute scalability property of a norm.
>
> iii Suppose $x = (1, 0)$ and $y = (0, 1)$. Then, we have $g(x + y) = (\sum_{i=1}^n |x_i + y_i|^{1/2})^2 = (2)^2 = 4$, and $g(x) = (\sum_{i=1}^n |x_i|^{1/2})^2 = (1)^2 = 1$ and $g(y) = (\sum_{i=1}^n |y_i|^{1/2})^2 = (1)^2 = 1$. Using simple arithmetic, we can conclude $g(x + y) = 4 \nleq g(x) + g(y) = 1 + 1 = 2$. Thus, $g(x)$ does not fulfill the triangle inequality property of a norm.
>
> Altogether, since g(x) does not meets all three properties of a norm, we can conclude g(x) is not a norm.

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.

What to Submit:

- **Parts a, b:** Proof.

A3. *[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 below (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.



Solution:

I The grey-shaded set is not convex as $\lambda x + (1-\lambda)y$ is not in I for points b and c and $\lambda \in [0,1]$.

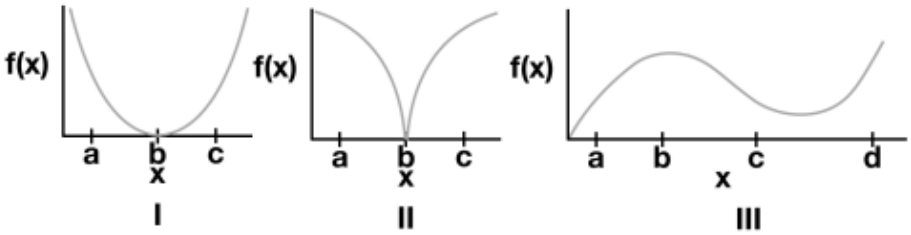
II The grey-shaded set is convex as $\lambda x + (1-\lambda)y \in II$ for all $x, y \in II$ and $\lambda \in [0,1]$.

III The grey-shaded set is not convex as $\lambda x + (1-\lambda)y$ is not in III for points a and d and $\lambda \in [0,1]$.

What to Submit:

- **Parts I-III:** 1-2 sentence explanation of why the diagram is convex or not.

A4. *[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 functions shown below (I-III), state whether each is convex on the specified 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 $[a, c]$

Solution:
The function is convex on the specified interval.

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

Solution:
The function is not convex on the specified interval as $f(\lambda x + (1-\lambda)y) \not\leq \lambda f(x) + (1-\lambda)f(y)$ for points a and b and $\lambda \in [0,1]$.

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

> Solution:
> The function is not convex on the specified interval as $f(\lambda x + (1-\lambda)y) \not\leq \lambda f(x) + (1-\lambda)f(y)$ for points a and c and $\lambda \in [0, 1]$.

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

> Solution:
> The function is convex on the specified interval.

What to Submit:

- **Parts a-d:** 1-2 sentence explanation of why the function is convex or not.

B1. Use just the definitions above and let $\|\cdot\|$ be a norm.

a. *[3 points]* Show that $f(x) = \|x\|$ is a convex function.

b. *[3 points]* Show that $\{x \in \mathbb{R}^n : \|x\| \leq 1\}$ is a convex set.

c. *[2 points]* Draw a picture of the set $\{(x_1, x_2) \ : \ g(x_1, x_2) \leq 4\}$ where $g(x_1, x_2) = \left(|x_1|^{1/2} + |x_2|^{1/2}\right)^2$. (This is the function considered in 1b above specialized to $n = 2$.) We know g is not a norm. Is the defined set convex? Why not?

Context: It is a fact that a function f defined over a set $A \subseteq \mathbb{R}^n$ is convex if and only if the set $\{(x, z) \in \mathbb{R}^{n+1} : z \geq f(x), x \in A\}$ is convex. Draw a picture of this for yourself to be sure you understand it.

What to Submit:

- **Parts a, b:** Proof.
- **Part c:** A picture of the set, and 1-2 sentence explanation.

Lasso on a Real Dataset

Given $\lambda > 0$ and data $\left(x_i, y_i\right)_{i=1}^n$, the Lasso is the problem of solving

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

where λ is a regularization parameter. For the programming part of this homework, you will implement the coordinate descent method shown in Algorithm 1 to solve the Lasso problem. This is a variant of the subgradient descent method and a more detailed discussion can be found in these slides.
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, the following hints may be useful:

- Wherever possible, use matrix libraries for matrix operations (not `for` loops).
- There are opportunities to considerably speed up parts of the algorithm by precomputing quantities like a_k before the for loop; you are permitted to add these improvements (and it may save you some time).

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

- 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).
- The smallest value of λ for which the solution $\widehat{w}$ is entirely zero is given by

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

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

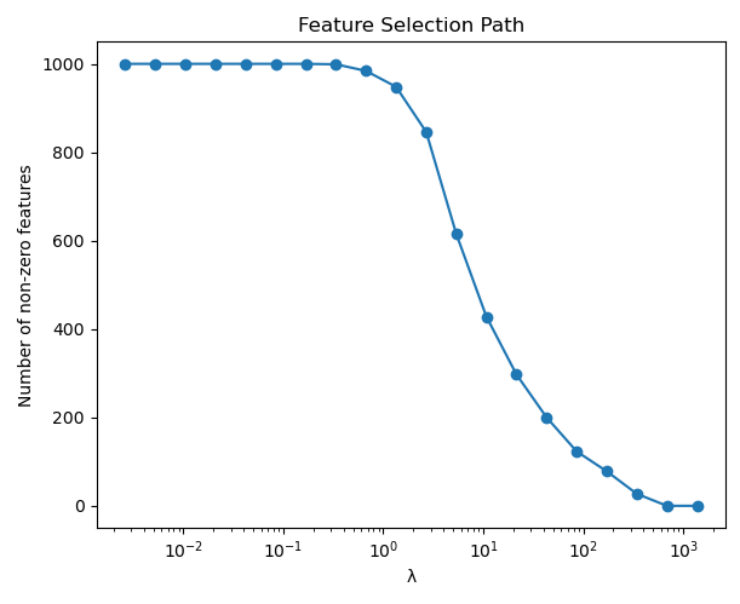
A5. We will first try out your solver with some synthetic data. A benefit of the Lasso is that if we believe many features are irrelevant for predicting y, the Lasso can be used to enforce a sparse solution, effectively differentiating between the relevant and irrelevant features. Suppose that $x \in \mathbb{R}^d, y \in \mathbb{R}, k < d$, and data are generated independently according to the model $y_i = w^T x_i + \epsilon_i$ where

$$w_j = \begin{cases} j/k & \text{if } j \in \{1, \ldots, k\} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is noise (note that in the model above $b = 0$). We can see from Equation (2) that since $k < d$ and $w_j = 0$ for $j > k$, the features $k + 1$ through d are irrelevant for predicting y.
Generate a dataset using this model with $n = 500, d = 1000, k = 100$, and $\sigma = 1$. You should generate the dataset such that each $\epsilon_i \sim \mathcal{N}(0, 1)$, and y_i is generated as specified above. You are free to choose a distribution from which the x's are drawn, but make sure standardize the x's before running your experiments.

a. *[10 points]* With your synthetic data, solve multiple Lasso problems on a regularization path, starting at λ_{max} where no features are selected (see Equation (1)) and decreasing λ by a constant ratio (e.g., 2) 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')`).

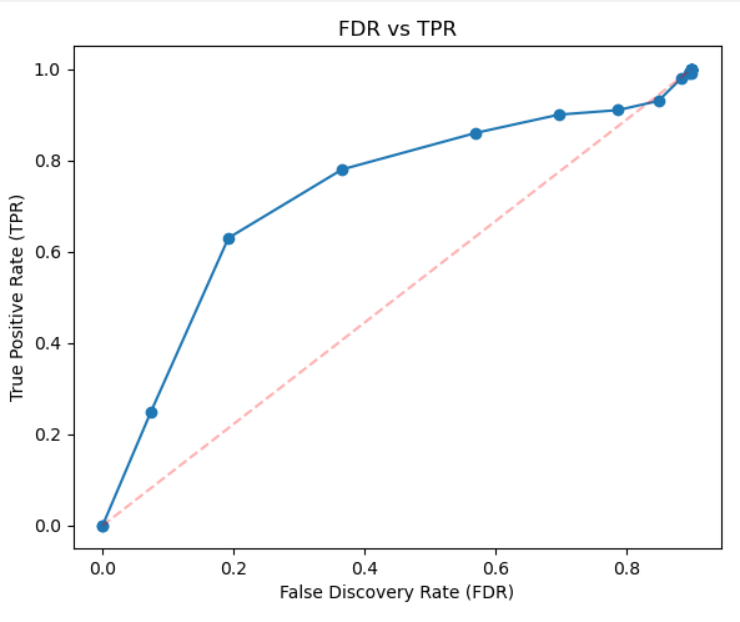
Solution:



b. *[10 points]* For each value of λ tried, record values for false discovery rate (FDR) (number of incorrect nonzeros in $\widehat{w}$/total number of nonzeros in $\widehat{w}$) and true positive rate (TPR) (number of correct nonzeros in $\widehat{w}$/k). Note: for each j, $\widehat{w}_j$ is an incorrect nonzero if and only if $\widehat{w}_j \neq 0$ while $w_j = 0$. In plot 2, plot these values with the x-axis as FDR, and the y-axis as TPR.

Note that in an ideal situation we would have an (FDR,TPR) pair in the upper left corner. We can always trivially achieve $(0, 0)$ and $(\frac{d-k}{d}, 1)$.

Solution:



c. *[5 points]* Comment on the effect of λ in these two plots in 1-2 sentences.

Solution:
As λ increases, the number of non-zero features decreases. With FDR and TPR, it appears as λ decreases, performance improves; however, after a certain point, smaller and smaller $\lambda's$ deteriorate performance.

What to Submit:

- **Part a:** Plot 1.
- **Part b:** Plot 2.
- **Part c:** 1-2 sentence explanation.
- **Code** on Gradescope through coding submission

A6. We'll now put the Lasso to work on some real data in `crime_data_lasso.py`. We have read in the data for you with the following:

```
df_train, df_test = load_dataset("crime")
```

This stores the data as Pandas `DataFrame` objects. `DataFrame`s are similar to Numpy `array`s but more flexible; unlike `array`s, `DataFrame`s store row and column indices along with the values of the data. Each column of a `DataFrame` can also store data of a different type (here, 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 and other systemic and historical factors. 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 threefold: (i) to encourage you to think about how data collection processes affect the resulting model trained from that data; (ii) to encourage you to think deeply about models you might train and how they might be misused; and (iii) to see how Lasso encourages sparsity of linear models in settings where d is large relative to n. **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.**

The dataset is split into a training and test set with 1,595 and 399 entries, respectively[1]. We will use this training set to fit a model to predict the crime rate in new communities and evaluate model performance on the

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

test set. As there are a considerable number of input variables and fairly few training observations, overfitting is a serious issue. In order to avoid this, use the coordinate descent Lasso algorithm implemented in the previous problem.

a. *[4 points]* Read 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 decision makers.

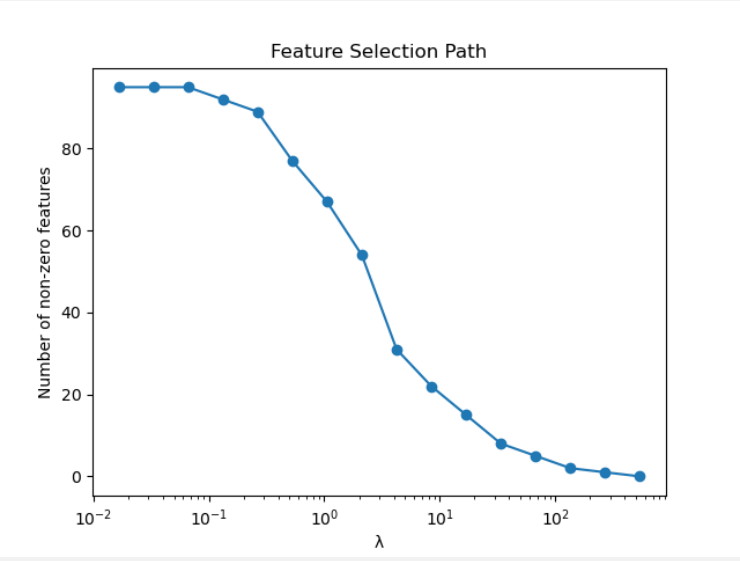
> Solution:
> Three features included in this dataset for which historical *policy* choices in the US would lead to variability in these features are median gross rent (i.e. rent caps), number of different kinds of drugs seized (legislation-based), and percent of people using public transit for commuting (funding-based).

b. *[4 points]* Before you train a model, 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.

> Solution:
> Three features in the dataset that might actually be a *result* rather than the cause of higher levels of violence are percentage of families (with kids) that are headed by two parents, percent of vacant housing that is boarded up, and number of homeless people counted in the street. In all honesty, some of these features could be interpreted as both reasons and results.

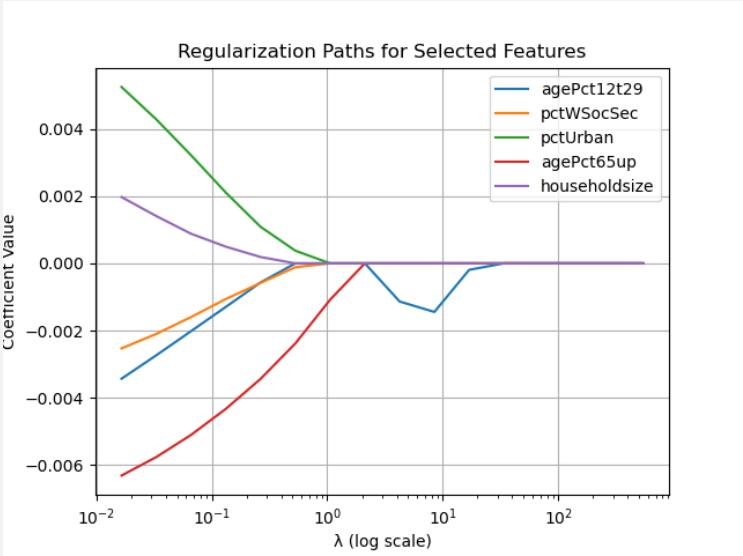
Now, we will run the Lasso solver. Begin with $\lambda = \lambda_{\max}$ defined in Equation (1). Initialize all weights to 0. Then, reduce λ by a factor of 2 and run again, but this time initialize $\hat{w}$ from your $\lambda = \lambda_{\max}$ solution as your initial weights, as described above. Continue the process of reducing λ by a factor of 2 until $\lambda < 0.01$. For all plots use a log-scale for the λ dimension (Tip: use `plt.xscale('log')`).

c. *[4 points]* Plot the number of nonzero weights of each solution as a function of λ.

> Solution:
>
> 
>

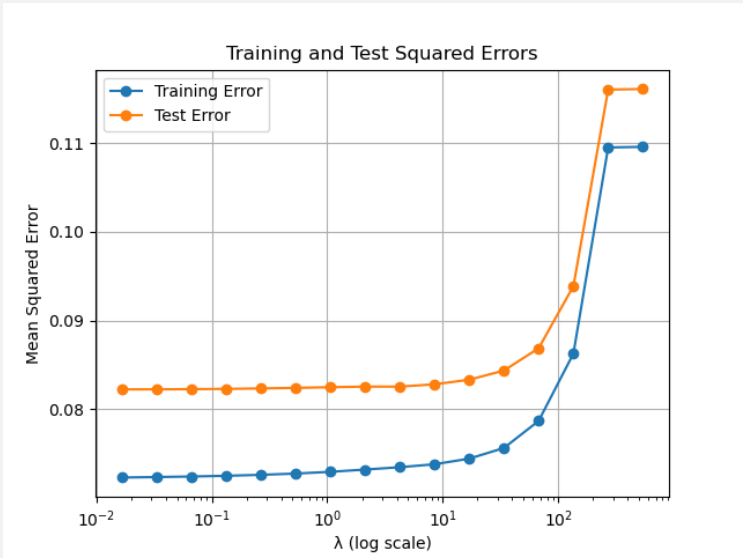
d. *[4 points]* Plot the regularization paths (in one plot) for the coefficients for input variables `agePct12t29`, `pctWSocSec`, `pctUrban`, `agePct65up`, and `householdsize`.

Solution:



e. *[4 points]* On one plot, plot the squared error on the training and test data as a function of λ.

Solution:



f. *[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. Retrain and inspect the weights $\hat{w}$ for $\lambda = 30$ and for *all* input variables. Which feature had the largest (most positive) Lasso coefficient? What about the most negative? Discuss briefly.

Solution:
Feature with the most positive coefficient: PctIlleg (0.06505220003188966)
Feature with the most negative coefficient: PctKids2Par (-0.06982508109436404)

As the percentage of children born to unmarried parents (PctIlleg) in a community increases, the model predicts an increase in the rate of violent crimes per population. On the other hand, as the percentage of children living in two-parent households (PctKids2Par) increases, the model predicts a decrease in ViolentCrimesPerPop.

g. *[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?)

Solution:
The fundamental flaw in the politician's line of reasoning is actually two-fold not only do we have the fact that correlation does not imply causation, but we also have reverse causality fallacy. In other words, just as fire trucks are not causes of fires just because every time there is fire, we see fire trucks, the number of people 65 and up are not causes of low crime areas just because we see them in low crime areas. Not to mention, there could also be omitted variables that affect both.

What to Submit:

- **Parts a, b:** 1-3 sentence explanation.
- **Part c:** Plot 1.
- **Part d:** Plot 2.
- **Part e:** Plot 3.
- **Parts f, g:** Answers and 1-2 sentence explanation.
- **Code** on Gradescope through coding submission.

Logistic Regression

Binary Logistic Regression

A7. Here we consider the MNIST dataset, but for binary classification. Specifically, the task is to determine whether a digit is a 2 or 7. Here, let $Y = 1$ for all the "7" 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).

Solution:

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

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

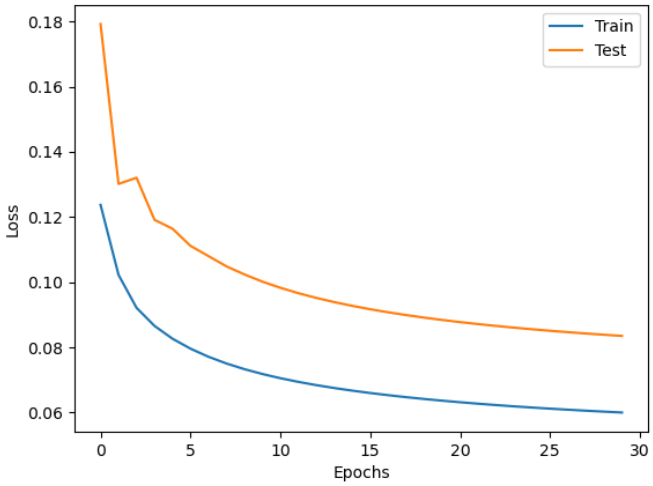
b. *[8 points]* Implement gradient descent with an initial iterate of all zeros. Try several values of step sizes to find one that appears to make convergence on the training set as fast as possible. Run until you feel you are near to convergence.

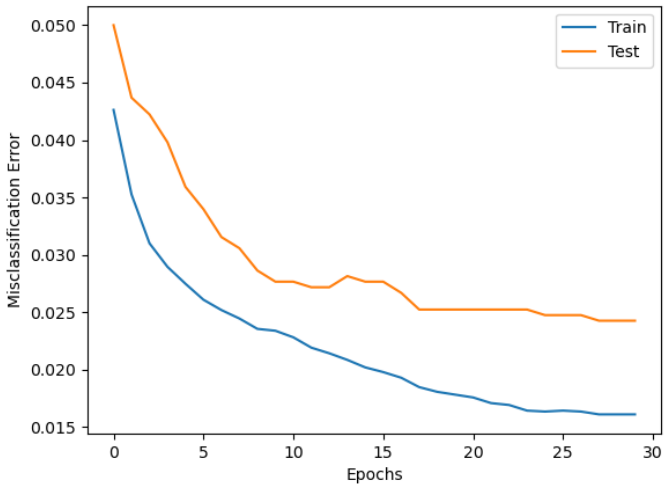
(i) For both the training set and the test, plot $J(w,b)$ as a function of the iteration number (and show both curves on the same plot).

(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).

Reminder: Make sure you are only using the test set for evaluation (not for training).

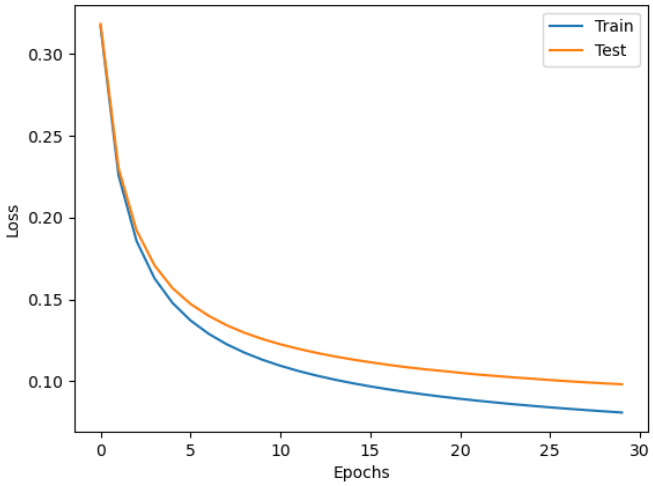
Solution:
Best Found Learning Rate: 2

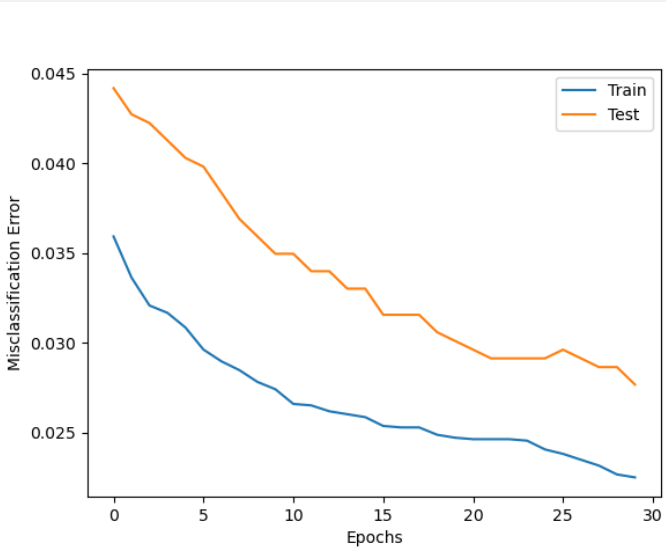




c. *[7 points]* Repeat (b) using stochastic gradient descent with a batch size of 1. Note, the expected gradient with respect to the random selection should be equal to the gradient found in part (a). Show both plots described in (b) when using batch size 1. Take careful note of how to scale the learning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

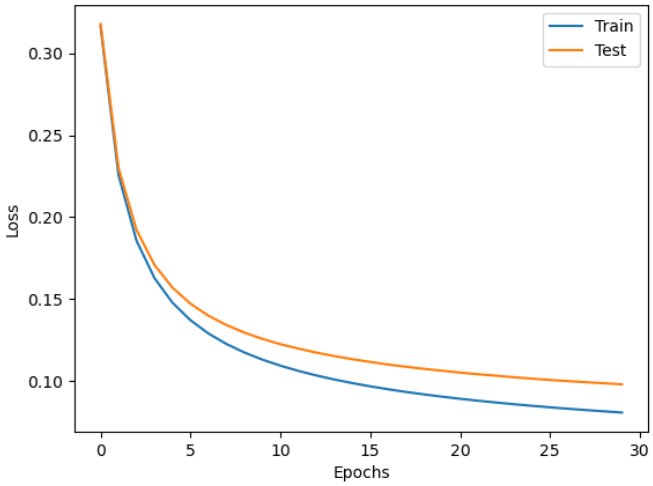
Solution:
Best Found Learning Rate: 4e-5

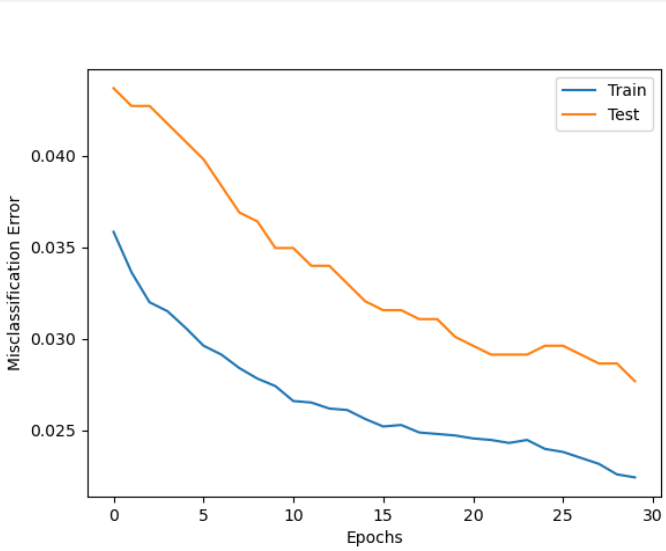




d. *[7 points]* Repeat (b) using stochastic gradient descent with batch size of 100. That is, instead of approximating the gradient with a single example, use 100. Note, the expected gradient with respect to the random selection should be equal to the gradient found in part (a).

Solution:
Best Found Learning Rate: 4e-3





What to Submit

- **Part a:** Proof
- **Part b:** Separate plots for b(i) and b(ii).
- **Part c:** Separate plots for c which reproduce those from b(i) and b(ii) for this case.
- **Part d:** Separate plots for c which reproduce those from b(i) and b(ii) for this case.
- **Code** on Gradescope through coding submission.

Confidence Interval of Least Squares Estimation

Bounding the Estimate

B2. Let us consider the setting, where we have n inputs, $X_1, ..., X_n \in \mathbb{R}^d$, and n observations $Y_i = \langle X_i, \beta^* \rangle + \epsilon_i$, for $i = 1, ..., n$. Here, β^* is a ground truth vector in $\mathbb{R}^d$ that we are trying to estimate, the noise $\epsilon_i \sim \mathcal{N}(0, 1)$, and the n examples piled up — $X \in R^{n \times d}$. To estimate, we use the least squares estimator $\widehat{\beta} = \min_\beta \|X\beta - Y\|_2^2$. Moreover, we will use $n = 20000$ and $d = 10000$ in this problem.

a. *[3 points]* Show that $\widehat{\beta}_j \sim \mathcal{N}(\beta_j^*, (X^T X)^{-1}_{j,j})$ for each $j = 1, ..., d$. *(Hint: see notes on confidence intervals.)*

b. *[4 points]* Fix $\delta \in (0, 1)$ suppose $\beta^* = 0$. Applying the proposition from the notes, conclude that for each $j \in [d]$, with probability at least $1 - \delta$, $|\widehat{\beta}_j| \leq \sqrt{2(X^T X)^{-1}_{j,j} \log(2/\delta)}$. Can we conclude that with probability at least $1 - \delta$, $|\widehat{\beta}_j| \leq \sqrt{2(X^T X)^{-1}_{j,j} \log(2/\delta)}$ for all $j \in [d]$ simultaneously? Why or why not?

c. *[5 points]* Let's explore this question empirically. Assume data is generated as $x_i = \sqrt{(i \mod d) + 1} \cdot e_{(i \mod d)+1}$ where e_i is the ith canonical vector and $i \mod d$ is the remainder of i when divided by d. Generate each y_i according to the model above. Compute $\widehat{\beta}$ and plot each $\widehat{\beta}_j$ as a scatter plot with the x-axis as $j \in \{1, \ldots, d\}$. Plot $\pm\sqrt{2(X^T X)^{-1}_{j,j} \log(2/\delta)}$ as the upper and lower confidence intervals with $1 - \delta = 0.95$. How many $\widehat{\beta}_j$'s are outside the confidence interval? *Hint: Due to the special structure of how we generated x_i, we can compute $(X^T X)^{-1}$ analytically without computing an inverse explicitly.*

What to Submit:

- **Parts a, b:** Proof.
- **Part b:** Answer.
- **Part c:** Plots of $\hat{\beta}$ and its confidence interval **on the same plot**.

Administrative

A8.

a. *[2 points]* About how many hours did you spend on this homework? There is no right or wrong answer :)

Solution:
10-15 hours