Homework #2A

CSE 546: Machine Learning
Cassia Cai
October 31, 2022

Collaborators: Apoorva Kalaskar, Madeleine Grunde-McLaughlin, and Aaleyah Lewis

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]* 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.

Compared to L2 norm penalty, which adds a ball-shaped region constraint and finds dense solutions, the L1 norm adds a diamond-shaped region constraint. This diamond-shaped region constraint is more likely to produce a sparse model (where at the intersection, one component of the solution is 0). This is due to the geometric properties of diamonds, which have corners where one component is 0. Reference is slide 19 of Lecture 7.

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

This regularizer is non-convex (downside). This regularizer produces an even sparser solution than the L1 norm penalty (upside).

c. *[2 points]* True or False: If the step-size for gradient descent is too large, it may not converge.

True. If the step-size for gradient descent is too large, it may not converge. If our step-size jump is too large, we might jump to higher values that are further from the minimum.

d. *[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.

SGD is less computationally demanding, while GD is more computationally demanding. SGD might take longer to converge to some minimum value than GD. However, fewer steps are needed to converge to the minimum with GD compared to SGD.

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 $a, b \in \mathbb{R}$.)

We first must check that $f(x)$ abides by the three properties: non-negativity, absolute scalability, and triangle inequality.

From the hint, we can show that $|a+b| \leq |a| + |b|$.

$$|a+b|^2 = a^2 + 2ab + b^2 \leq |a|^2 + 2|a||b| + |b|^2 = (|a| + |b|)^2 \tag{1}$$

This leads to the following:

$$|a+b| \leq |a| + |b| \tag{2}$$

First, let's check non-negativity: $x_i \in \mathbb{R}$ and $|x_i| \geq 0$ for all i:

$$f(x) = \|x\| = \sum_{i=1}^n |x_i| \geq 0 \tag{3}$$

This is true for all $x \in \mathbb{R}^n$ and $f(x) = 0$ if and only if $x = 0$

Next, let's check absolute scalability:

$$f(ax) = \|ax\| = \sum_{i=1}^n |ax_i| = \sum_{i=1}^n |a||x_i| = |a| \sum_{i=1}^n |x_i| = |a|\|x\| = |a|f(x) \tag{4}$$

Finally, let's check the triangle inequality:

$$f(x+y) = \|x+y\| = \sum_{i=1}^n |x_i + y_i| \leq \sum_{i=1}^n (|x_i| + |y_i|) = \sum_{i=1}^n |x_i| + \sum_{i=1}^n |y_i| = \|x\| + \|y\| = f(x) + f(y) \tag{5}$$

This leads to the following:

$$f(x+y) \leq f(x) + f(y) \tag{6}$$

We have thus shown that $f(x) = \sum_{i=1}^n |x_i|$ 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.)

Let's start from the hint and consider 2 points $x = [0, c]^T$ and $y = [c, 0]^T$ where c is some positive constant.

$$g(x) + g(y) = 2c \tag{7}$$

$$g(x+y) = 4c \tag{8}$$

We thus have $g(x+y) \geq g(x) + g(y)$

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.

A3. *[2 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-II), 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.
- I is not convex. We exit the image when we try to connect b and c.
- II is not convex because when $(x, y) = (a, d)$, $\lambda x + (1-\lambda)y$ does not hold.

A4. *[2 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-II), 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 $[a, c]$

   The function is convex as there are no points that exit the graph on $[a, c]$.

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

   The function $[a, d]$ is not convex. If we try connecting $f(a)$ and $f(d)$, we exit the graph.

Lasso on a Real Dataset

Given $\lambda > 0$ and data $\Big(x_i, y_i\Big)_{i=1}^{n}$, 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. Implement the coordinate descent method shown in Algorithm 1 to solve the Lasso problem in `coordinate_descent_algo.py`.

Algorithm 1: Coordinate Descent Algorithm for Lasso

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

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

$$w_j = \begin{cases} j/k & \text{if } j \in \{1, \dots, k\} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is noise (note that in the model above $b = 0$). We can see from Equation (9) 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 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.

   See plot on the next page.

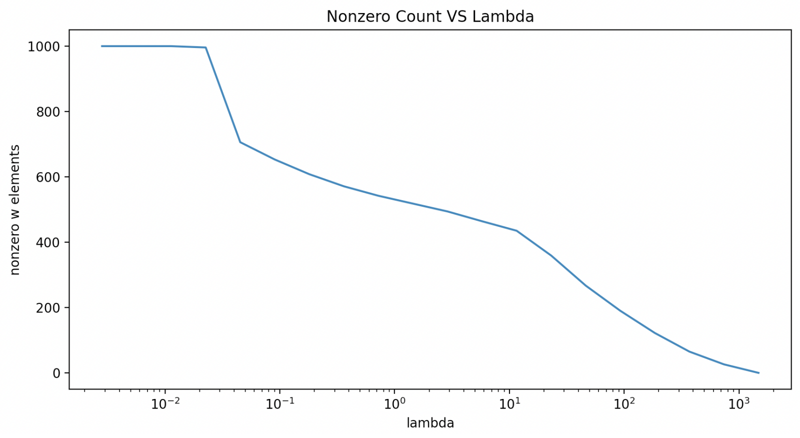
b. *[10 points]* For each value of λ tried, record values for false discovery rate (FDR) (number of incorrect nonzeros in $\widehat{w}$/total number of nonzeros in $\widehat{w}$) and true positive rate (TPR) (number of correct nonzeros in $\widehat{w}$/k). Note: for each j, $\widehat{w}_j$ is an incorrect 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.

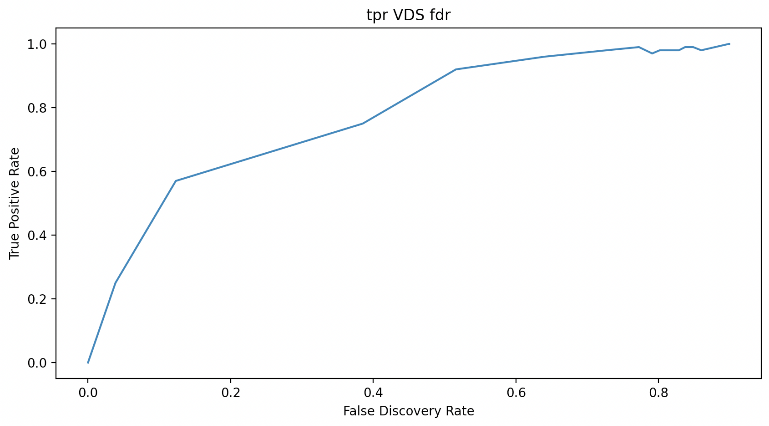
   Note that in an ideal situation we would have an (FDR,TPR) pair in the upper left corner. We can always trivially achieve $(0,0)$ and $(\frac{d-k}{d}, 1)$.

   See plot on the next page.

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

   A larger λ means that we have more zeroes in our weight vector. A smaller λ means that more weights in our weight vector will be non-zero, so that the model will use more features. From our second plot, we





see that a too large λ leads to 0 solution (poor performance). If λ is too small, then we have a high FDR. As the hint says, we want to choose λ in the upper left corner. Our choice of λ should preserve a sparse model that is still accurate.

```
def precalculate_a(X: np.ndarray) -> np.ndarray:
    return 2*np.sum(X*X, axis = 0)

def step(X, y, weight, a, _lambda):
    n,d = X.shape
    w = weight
    b = np.mean(y-np.dot(X,w))
    for k in range(d):
        xk = X[:,k]
        w[k] = 0.0
        ck = 2.0 * np.dot(xk, y - (b + np.dot(X,w)))
        if ck < -_lambda:
            w[k] = (ck + _lambda)/a[k]
        elif ck > _lambda:
            w[k] = (ck - _lambda)/a[k]
    return (w,b)

def loss(X, y, weight, bias, _lambda):
    return np.linalg.norm(X@weight+bias-y)**2+_lambda * np.linalg.norm(weight,1)

def train(X,y,_lambda,convergence_delta,start_weight,):
    n, d = X.shape
    if start_weight is None:
        start_weight = np.zeros(d)
    old_w = start_weight + np.inf
    a = precalculate_a(X)
    while convergence_criterion(start_weight, old_w, convergence_delta) is False:
        old_w = np.copy(start_weight)
        w = start_weight
        train_weights_and_bias = step(X, y, w, a, _lambda)
    return train_weights_and_bias

def convergence_criterion(weight, old_w, convergence_delta):
    max_abs_change = np.linalg.norm(weight - old_w, ord=np.inf)
    if max_abs_change > convergence_delta:
        return False

def main():
    sigma, n, d, k, no_iter = 1, 500, 1000, 100, 5
    rgn = np.random.normal(0, sigma**2.0, size=n) # random gaussian noise
    x = np.random.normal(size = (n,d))
    w = np.zeros((d, ))
    for j in range(1, k+1):
        w[j-1] = j/k
    y = x @ w + rgn
    lambda_max = np.max(2 * np.sum(x.T * (y - np.mean(y))  , axis=0))
    lambdas = [lambda_max/(2**i) for i in range(no_iter)]
    convergence_delta = 1e-4
    noNumZeros, fdr, tpr = [], [], []
    for lambda_reg in lambdas:
        w = train(x, y, lambda_reg, 1e-4, None)[0]
        notZeros = np.count_nonzero(w); noNumZeros.append(notZeros)
        correctNonZeros = np.count_nonzero(w[:k])
        incorrectNonZeros = np.count_nonzero(w[k+1:])
        try:
            fdr.append(incorrectNonZeros/notZeros)
        except ZeroDivisionError:
            fdr.append(0)

        tpr.append(correctNonZeros/k)

    # part a
    plt.figure(figsize = (10,5))
    plt.plot(lambdas, noNumZeros)
    plt.xscale('log')
    plt.title('Nonzero␣Count␣VS␣Lambda')
    plt.xlabel('lambda')
    plt.ylabel('nonzero␣w␣elements')
    plt.show()

    # part b
    plt.figure(figsize = (10,5))
    plt.plot(fdr, tpr)
    plt.title('tpr␣VDS␣fdr')
    plt.xlabel('False␣Discovery␣Rate')
    plt.ylabel('True␣Positive␣Rate')
    plt.show()
```

```
if __name__ == "__main__":
    main()
```

A6. We'll now put the Lasso to work on some real data in `crime_data_lasso.py`. . Download the training data set "crime-train.txt" and the test data set "crime-test.txt" from the website. Store your data in your working directory, ensure you have the `pandas` library for Python installed, 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. `DataFrame`s are similar to Numpy `arrays` but more flexible; unlike `arrays`, `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 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 originalcversion 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.

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

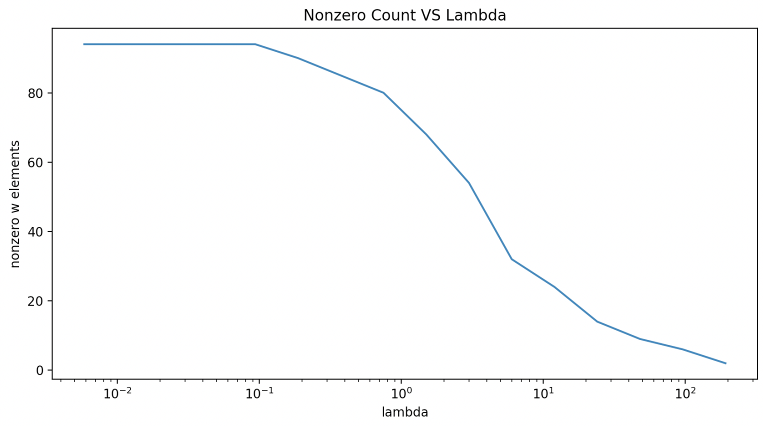
The police operating budget (per population) is often a consequence of policy choices. The number of kinds of drugs seized is also probably affected by policies. The number of officers assigned to special drug units is also probably affected by policies. I think that numbers related to the police force or drugs are likely to have been influenced by policy decisions.

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.

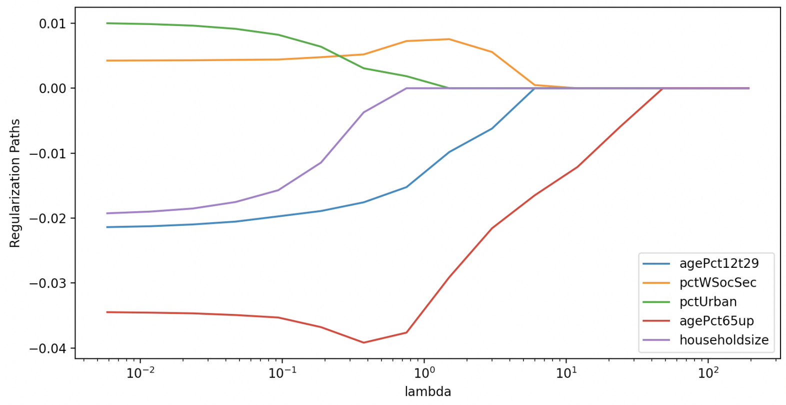
Features related to requests might actually be a result rather than the cause of violent crime. This is because whenever there is a request for police/help, it is likely because of some form of crime. The request itself is very unlikely to be the cause of the crime. So the features are: LemasTotReqPerPop (total requests for police per 100K population), LemasTotalReq (total requests for police), and PolicReqPerOfficg (total requests for police per police officer).

Now, we will run the Lasso solver. Begin with $\lambda = \lambda_{\max}$. 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')`).

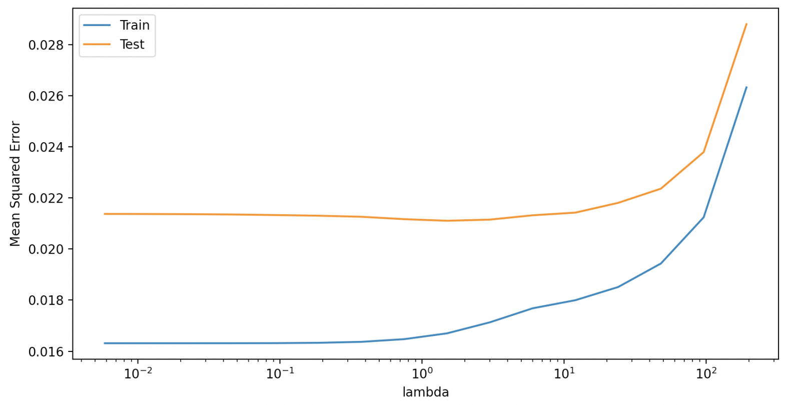
c. *[4 points]* Plot the number of nonzero weights of each solution as a function of λ.



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



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



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. Inspect the weights $\hat{w}$ for $\lambda = 30$. Which feature had the largest (most positive) Lasso coefficient? What about the most negative? Discuss briefly.

The largest is PctIlleg (percentage of kids born to never married), which is 0.06873. The smallest is PctKids2Par (percentage of kids in family housing with two parents), which is -0.06922. PctIlleg has the highest weight, and indicates that kids born to never married parents are predicted to be associated with high violent crimes per population. This may be perhaps that kids born to never married parents might not have the same amount of stability as kids in family housing with two parents (which has a negative weight).

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?)

Correlation does not necessarily mean causation. So a large negative weight on agePct65up indicates that there is a high correlation between agePct65up and ViolentCrimesPerPop. agePct65up (percentage of population that is 65 and over) might not the reason for a lower crime rate. Let us imagine a region with a high crime rate already. Perhaps the case is that people over 65 move out of these regions and into regions with lower crime rates.

```
def main():
    df_train , df_test = load_dataset("crime")
    x_df = df_train .drop('ViolentCrimesPerPop' , axis=1)
    y_df = df_train['ViolentCrimesPerPop']
    x = np.asarray(df_train .drop('ViolentCrimesPerPop' , axis=1))
    y = np.asarray(df_train['ViolentCrimesPerPop'])
    n, d = x.shape
    x_test = np.asarray(df_test .drop('ViolentCrimesPerPop' , axis=1))
    y_test = np.asarray(df_test['ViolentCrimesPerPop'])

    lambda_max = np.max(2 * np.sum(x.T * (y - np.mean(y)) , axis=0))
    w = np.zeros((d, ))

    no_iter = 16
    lambdas = [lambda_max/(2**i) for i in range(no_iter)]
    convergence_delta = 1e-4

    noNumZeros, train_mse, test_mse = [], [], []
    feat_names = ['agePct12t29', 'pctWSocSec', 'pctUrban', 'agePct65up', 'householdsize']
```

```
feat_indices = [x_df.columns.get_loc(i) for i in feat_names]
agePct12t29_ls, pctWSocSec_ls, pctUrban_ls = [], [], []
agePct65up_ls, householdsize_ls = [], []
for lambda_reg in lambdas:
    w = train(x, y, lambda_reg, 1e-4, None)[0]
    b = train(x, y, lambda_reg, 1e-4, None)[1]
    notZeros = np.count_nonzero(w); noNumZeros.append(notZeros)
    y_preds = x.dot(w) + b
    y_test_preds = x_test.dot(w) + b
    train_mse_val = np.mean((y-y_preds)**2)
    test_mse_val = np.mean((y_test-y_test_preds)**2)
    train_mse.append(train_mse_val)
    test_mse.append(test_mse_val)
    agePct12t29_ls.append(w[0])
    pctWSocSec_ls.append(w[1])
    pctUrban_ls.append(w[2])
    agePct65up_ls.append(w[3])
    householdsize_ls.append(w[4])

# part a
plt.figure(figsize = (10,5))
plt.plot(lambdas, noNumZeros)
plt.xscale('log')
plt.title('Nonzero Count VS Lambda')
plt.xlabel('lambda')
plt.ylabel('nonzero w elements')
plt.show()

# part b
plt.figure(figsize = (10,5))
plt.plot(lambdas, agePct12t29_ls, label='agePct12t29')
plt.plot(lambdas, pctWSocSec_ls, label='pctWSocSec')
plt.plot(lambdas, pctUrban_ls, label='pctUrban')
plt.plot(lambdas, agePct65up_ls, label='agePct65up')
plt.plot(lambdas, householdsize_ls, label='householdsize')
plt.xscale('log')
plt.xlabel('lambda')
plt.ylabel('Regularization Paths')
plt.legend ()
plt.show()

# part c
plt.figure(figsize=(10, 5))
plt.plot(lambdas, train_mse , label='Train')
plt.plot(lambdas, test_mse , label='Test')
plt.xscale ('log')
plt.xlabel('lambda')
plt.ylabel('Mean Squared Error')
plt.legend ()
plt.show()

# part d
lambda_val = 30
w = train(x, y, lambda_val, 1e-4, None)[0]
# print(np.argmax(w))
```

```
    # print(np.argmin(w))
    print(x_df.columns[np.argmax(w)])
    print(np.max(w))
    print(x_df.columns[np.argmin(w)])
    print(np.min(w))

if __name__ == "__main__":
    main()
```

Gradient Descent

A7. Consider an experiment in which you have n observations and corresponding labels $\{x_i, y_i\}_{i=1}^n$ such that $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$. Let $X \in \mathbb{R}^{n \times d}$ be the data matrix containing x_i^T as rows. For this question, assume X has full rank, such that $X^T X$ is invertible.

As we saw before, in linear regression we would like to come up with a model $w \in \mathbb{R}^d$ that minimizes the loss

$$L(w) = ||y - Xw||_2^2 = (y - Xw)^T(y - Xw)$$

We know from lecture that the optimal value for w is $w^* = (X^T X)^{-1} X^T y$. However, we can also use gradient descent instead of calculating this directly. In this question, we will start to prove that for a good choice of the learning rate, the gradient descent algorithm arrives at the same optimal result.

a. *[5 points]* Show (step-by-step) that the gradient $\nabla_w L(w) \in \mathbb{R}^d$ is equal to

$$\nabla_w L(w) = 2(X^T X)w - 2X^T y$$

From lecture, we had $\nabla_w (Aw + b)^T(Aw_b) = 2A^T(Aw + b)$. In this case, we have $A = -X$ and $b = y$. We can thus find:

$$\nabla_w L(w) = (-Xw_y)^T(-Xw + y) = -2X^T(-Xw + y) = 2(X^T X)w - 2X^T y \tag{10}$$

b. *[3 points]* In gradient descent, we start with a vector w_0 and iteratively update it with the rule

$$w_{k+1} = w_k - \alpha \nabla_{w_k} L(w_k)$$

Using this rule, show that

$$||w_{k+1} - w^*|| = ||(I - 2\alpha X^T X)(w_k - w^*)||$$

(Hint: We have that $\nabla_{w^*} L(w^*) = 0$, thus you can replace $\nabla_{w_k} L(w_k)$ with $\nabla_{w_k} L(w_k) - \nabla_{w^*} L(w^*)$ in your solution, and then expand the definitions.)

$$w_{k+1} = w_k - \alpha \nabla_{w_k} L(w_k) \tag{11}$$

$$w_{k+1} = w_k - \alpha(\nabla_{w_k} L(w_k) - \nabla_{w^*} L(w^*)) \tag{12}$$

$$w_{k+1} = w_k - \alpha \nabla_{w_k} L(w_k) + \alpha \nabla_{w^*} L(w^*) = w_k - \alpha(2(X^T X)w_k - 2X^T y) + \alpha(2(X^T X)w^* - 2X^T y) \tag{13}$$

$$w_{k+1} = w_k - \alpha 2 w_k (X^T X) - \alpha 2 (X^T X) w^* \tag{14}$$

$$w_{k+1} - w^* = w_k - \alpha 2 w_k (X^T X) - \alpha 2 (X^T X) w* = w_k - \alpha 2 (X^T X)(w_k - w^*) - w^* \tag{15}$$

$$w_{k+1} - w^* = (1 - 2\alpha X^T X)(w_k - w^*) \tag{16}$$

Hence, we have shown that:

$$||w_{k+1} - w^*|| = ||(I - 2\alpha X^T X)(w_k - w^*)|| \tag{17}$$

c. *[4 points]* Since $X^T X$ is positive semi-definite, all its eigenvalues are nonnegative. Let h be the smallest, and H be the largest eigenvalue of $X^T X$. Using the previous part, show that

$$||w_{k+1} - w^*|| \leq \rho ||w_k - w^*||$$

where $\rho = \max(|1 - 2\alpha h|, |1 - 2\alpha H|)$.

Hints:

- Let λ be the eigenvalue of a matrix $A \in \mathbb{R}^{d \times d}$ that has the largest magnitude. Then for any vector $x \in \mathbb{R}^d$ we have $||Ax|| \leq |\lambda| \cdot ||x||$.)
- If $\gamma_1, \ldots, \gamma_d$ are the eigenvalues of a matrix A, then $\gamma_1 + 1, \ldots, \gamma_d + 1$ are the eigenvalues of $A + I$).

Let us start with:

$$||w_{k+1} - w^*|| = ||(I - 2\alpha X^T X)(w_k - w^*)|| \tag{18}$$

where $(I - 2\alpha X^T X) = A$ and $(w_k - w^*) = x$. From our hint, we have:

$$||Ax|| \leq |\lambda| \cdot ||x|| = |\lambda| \cdot ||(w_k - w^*)|| \tag{19}$$

We can then tell that $|\lambda| = \rho = \max(|1 - 2\alpha h|, |1 - 2\alpha H|)$. This problem is about recognizing that from Equation 17, we can say the items in the first bracket are A and the items in the second bracket are x and then work from the hint.

Administrative

A8.

a. *[2 points]* About how many hours did you spend on this homework? About 10.