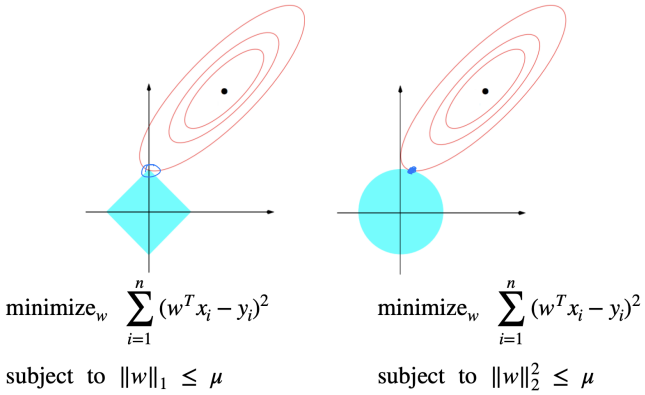
Homework #2

CSE 446/546: Machine Learning
Profs. Jamie Morgenstern and Ludwig Schmidt
Ray Chen
Collaborator: Kevin Shao

November 1, 2022

A1:

- **Part a:**
  L1 norm is more likely to have an intersection area due to its geometric shape like ellipses, disks, and diamonds.



$$\text{minimize}_w \sum_{i=1}^{n} (w^T x_i - y_i)^2 \quad \text{subject to } \|w\|_1 \leq \mu$$

$$\text{minimize}_w \sum_{i=1}^{n} (w^T x_i - y_i)^2 \quad \text{subject to } \|w\|_2^2 \leq \mu$$

- **Part b:**
  The Regularizer may yield an much sparser solution in up side, and it is also non-convex in down side.

- **Part c:**
  True. If the step-size for gradient descent is too large, it may not converge. And it will never converge to Min since it overshoot every time.

- **Part d:**
  SGD is faster than GD when doing the caculations. However, SGD takes more time to converge than GD, which means GD takes fewer steps to converge to Min than SGD.

A2:

- **Part a:**
  First, we have

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

  To show $f(x) = (\sum_{i=1}^{n} |x_i|)$ is a norm, we need to check non-negativity, absolute scalability and triangle inequality:

  Non-Negativity: $x_i \geq 0$ for all the i, the function $f(x) = (\sum_{i=1}^{n} |x_i|)$ should also satisfied $f(x) \geq 0$ for all the i. Besides, only when $x = 0$ then $f(x) = 0$.

  Absolute Scalability: $f(ax) = (\sum_{i=1}^{n} |ax_i|) = |a| (\sum_{i=1}^{n} |x_i|) = |a| f(x)$ for all $a \in \mathbb{R} and x \in \mathbb{R}^n$

  Triangle Inequality: $f(x) + f(y) = (\sum_{i=1}^{n} |x_i|) + (\sum_{i=1}^{n} |y_i|) = (\sum_{i=1}^{n} |x_i + y_i|) \geq (\sum_{i=1}^{n} |x_i| + |y_i|)$ for all $x, y \in \mathbb{R}^n$ (According to Equation(1))

  So, $f(x) = (\sum_{i=1}^{n} |x_i|)$ is a norm.

- **Part b:**
  To show $g(x) = \left(\sum_{i=1}^{n} |x_i|^{1/2}\right)^2$ is a norm, we need to check non-negativity, absolute scalability and triangle inequality:

  Triangle Inequality: We can use $x = [0, a] and y = [a, 0]$. $g(x) + g(y) = 2a \leq g(x + y) = 4a$. And it is not satisfied the Equation(1).

  So, $g(x) = \left(\sum_{i=1}^{n} |x_i|^{1/2}\right)^2$ is a norm.

A3:

- **I :**

  If the set I is convex, all the line segments between any two points must be in the sets $\lambda \in [0, 1]$

  when(x,y)=(b,c)

  So, I is not convex.

- **III :**

  If the set II is convex, all the line segments between any two points must be in the sets $\lambda \in [0, 1]$

  when(x,y)=(a,d), $\lambda x + (1 - \lambda)y \in A$ for all $x, y \in A$ is not applicable.

  So, II is not convex.

A4:

- **I :**

  Because $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all $x, y \in A$ and $\lambda \in [0,1]$

  So, I is convex.

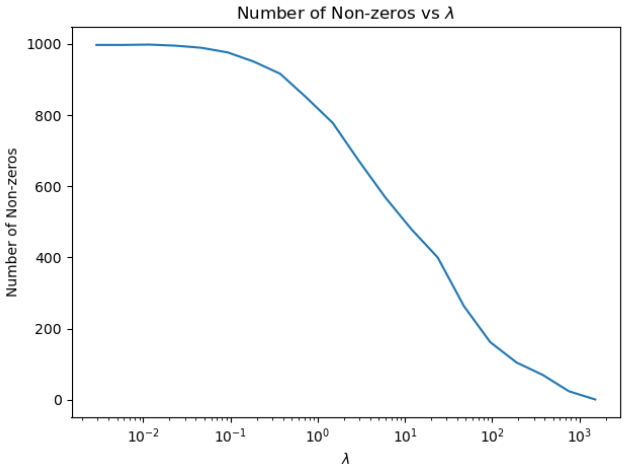
- **III :**

  Because $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all $x, y \in A$ and $\lambda \in [0,1]$

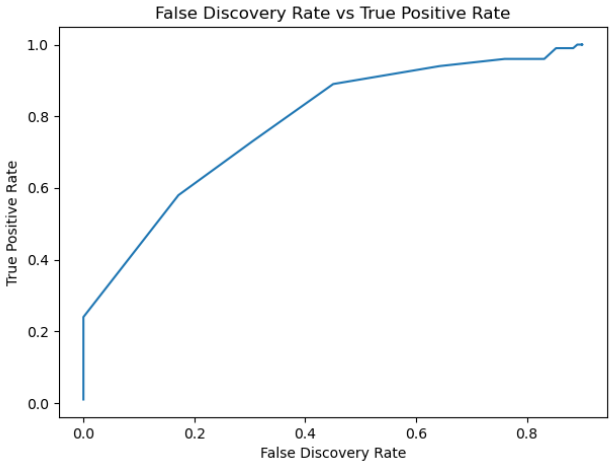
  So, II is convex.

A5:

- **Part a:**



- **Part b:**



- **Part c:**

  Effect of λ on plot 1:

  As λ increases, just as shown in the plot the number of non-zero decreases. Small λ will make the model use all the features, while big λ use less features.

  Effect of λ on plot 2:

  As increase of λ, just as shown in the plot the number of non-zero features decrease. So, FDR will decrease. As increase of λ, number of non-zero decrease. This is good for lasso since it makes easier for lasso to get correct number of non-zero.

  As a result, the value of λ can easily influence the lasso model accuracy.

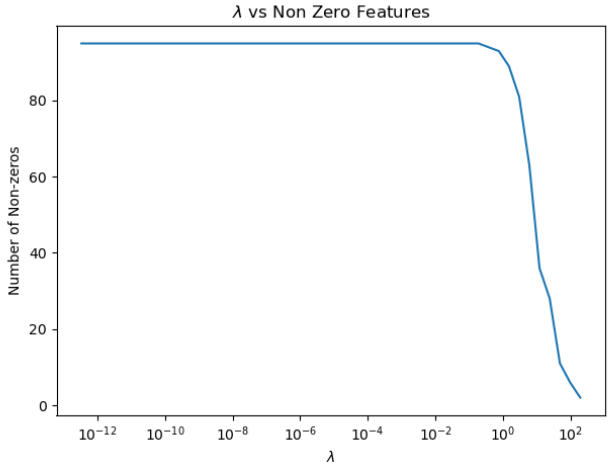
- **Code:**

```
from os import times
from typing import Optional, Tuple
import matplotlib.pyplot as plt
import numpy as np
from utils import problem


@problem.tag("hw2-A")
def precalculate_a(X: np.ndarray) -> np.ndarray:
    return 2 * np.sum(X ** 2, axis = 0)

@problem.tag("hw2-A")
def loss(
    X: np.ndarray, y: np.ndarray, weight: np.ndarray, bias: float, _lambda: float
) -> float:
    return  ((np.linalg.norm(X.dot(weight) + bias - y))**2
            + _lambda * (np.linalg.norm(weight, ord=1)))

@problem.tag("hw2-A")
def step(
    X: np.ndarray, y: np.ndarray, weight: np.ndarray, a: np.ndarray, _lambda: float
) -> Tuple[np.ndarray, float]:
    bias = np.average(y - X @ weight)
    for k in range(X.shape[1]):
        ck = 2 * X[:, k] @ (y - (bias + X @ weight - X[:, k] * weight[k]))
        if ck < -_lambda:
            weight[k] = (ck + _lambda) / a[k]
        elif ck > _lambda:
            weight[k] = (ck - _lambda) / a[k]
        else: weight[k] = 0.0
    return weight, bias

@problem.tag("hw2-A", start_line=4)
def train( X: np.ndarray, y: np.ndarray, _lambda: float = 0.01,
convergence_delta: float = 1e-4, start_weight: np.ndarray = None,
) -> Tuple[np.ndarray, float]:
    if start_weight is None:
        start_weight = np.zeros(X.shape[1])
    a = precalculate_a(X)
    old_w: Optional[np.ndarray] = None
    # initializing
    while not convergence_criterion(start_weight, old_w, convergence_delta):
        old_w = np.copy(start_weight) # renew the array
        weight, bias = step(X, y, start_weight, a, _lambda)

    return weight, bias

@problem.tag("hw2-A")
def convergence_criterion(weight: np.ndarray, old_w: np.ndarray,
convergence_delta: float) -> bool:
    if old_w is None: return False
    else: return ((abs(max(old_w) - max(weight))) < convergence_delta)
```

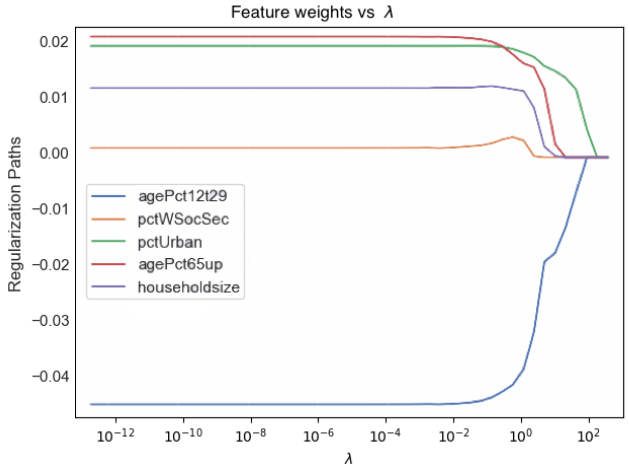
```
@problem.tag("hw2-A")
def main():
    # setting based on question
    n, d, k, sigma = 500, 1000, 100, 1
    points = 50
    weight = np.zeros((d, ))
    for j in range(1, k+1):
        weight[j-1] = j/k
    X = np.random.normal(size=(n, d))
    y = X.dot(weight) + np.random.normal(size=(n,))
    reg_lambda = max(2*np.sum(X * (y-np.mean(y)) [:, None], axis=0))
    # initilazing arrays for plotting
    lambdas, nonzeros, fdrs, tprs = [], [], [], []
    # compute those arrays
    for _ in range(points):
        f_weight, f_bias = train(X, y, reg_lambda, 1e-3, weight)
        lambdas.append(reg_lambda)
        nonzero_cnt = np.sum(abs(f_weight) > 0)
        fdr = np.sum(abs(f_weight[k:]) > 0) / nonzero_cnt
        tpr = np.sum(abs(f_weight[:k]) > 0) / k
        nonzeros.append(nonzero)
        fdrs.append(fdr)
        tprs.append(tpr)
        reg_lambda /= 2
    # plot A5a
    plt.figure(figsize=(7, 5))
    plt.plot(lambdas, nonzeros)
    plt.xscale('log')
    plt.title("Number of Non-zeros vs $\lambda$")
    plt.xlabel('$\lambda$')
    plt.ylabel('Number of Non-zeros')
    plt.savefig('A5a.png')
    # plot A5b
    plt.figure(figsize=(7, 5))
    plt.plot(fdrs, tprs)
    plt.title("False Discovery Rate vs True Positive Rate")
    plt.xlabel('False Discovery Rate')
    plt.ylabel('True Positive Rate')
    plt.savefig('A5b.png')

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

A6:

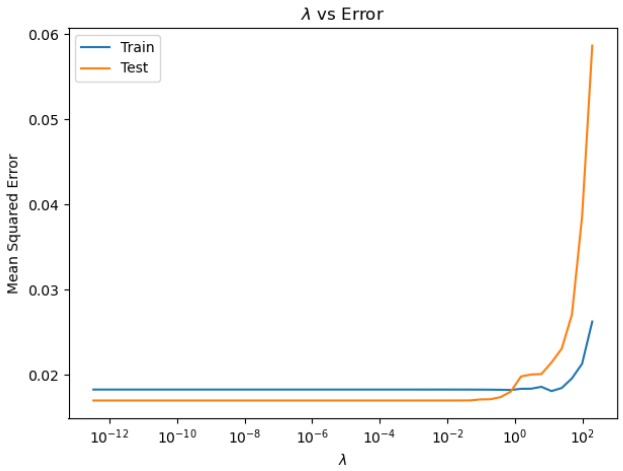
- **Part a:**
  - **NumUnderPov:** number of people under the poverty level (numeric - decimal)
  - **blackPerCap:** per capita income for african americans (numeric - decimal)
  - **LemasPctOfficDrugUn:** percent of officers assigned to drug units (numeric - decimal)
- **Part b:**
  We think some additional house features are not well related to the voliance. So, below three features should be nonzero weight in our model.
  - **PctHousNoPhone:** percent of occupied housing units without phone (numeric - decimal)
  - **PctWOFullPlumb:** percent of housing without complete plumbing facilities (numeric - decimal)
  - **MedNumBR:** median number of bedrooms (numeric - decimal)
- **Part c:**



- **Part d:**



- **Part e:**



- **Part f:**

```
(cse446) zc@Rays-MacBook-Pro lasso % python crime_data_lasso.p
Max lambda:  191.98052915360506
Largest feacture  :  PctIlleg 0.06802480660005247
Smallest feacture  :  PctKids2Par -0.07017507675275239
```

So, in this area percentage of kids born to never married has most positive Lasso coefficient; the percentage of kids in family housing with two parents has most negative Lasso coefficient. It is reasonable that kids' family situation relate to the local crime rate.

- **Part g:**

In statistics, the high negative weight only means it doesn't have many correlation with the result instead of it can reduce the chance of that result. Also, negative weight on the agePct65up feature may be related to other feature. So, higher percentage of people who over 65 is not the cause of low crime rate.

- **Code:**

```
if __name__ == "__main__":
    from coordinate_descent_algo import train  # type: ignore
else:
    from .coordinate_descent_algo import train

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from utils import load_dataset, problem

@problem.tag("hw2-A", start_line=3)
def main():
    df_train, df_test = load_dataset("crime")
    df_train, df_test= pd.read_table("crime-train.txt"), pd.read_table("crime-test.txt")
    X, y= df_train.drop('ViolentCrimesPerPop', axis=1), df_train['ViolentCrimesPerPop']
    X_test, y_test = df_test.drop('ViolentCrimesPerPop', axis=1), df_test['ViolentCrimesPerPop']
    reg_lambda = max(2 * np.sum(X.T * (y - np.mean(y)), axis=0))
    print(reg_lambda)

    # use to test Lasso coefficien, while set lamda = 30
    f_weight, f_bias = train(X_test.values, y_test.values, 30)
    print('Largest feacture: ', X.columns[np.argmax(f_weight)], max(f_weight))
    print('Smallest feacture: ', X.columns[np.argmin(f_weight)], min(f_weight))

    points = 50
    # initilazing arrays for plotting
    lambdas, nonzeros, mse_train, mse_test, reg_path = [], [], [], [], []
    for i in range(points):
        lambdas.append(reg_lambda)
        f_weight_train, f_bias_train = train(X.values, y.values, reg_lambda, 1e-3)
        f_weight_test, f_bias_test = train(X_test.values, y_test.values, reg_lambda, 1e-3)

        y_train_pred = X.values.dot(f_weight_train) + f_bias_train
        y_test_pred = X_test.values.dot(f_weight_test) + f_bias_test

        nonzeros.append(np.sum(abs(f_weight_train) > 0))
        mse_train.append(np.mean((y.values-y_train_pred)**2))
        mse_test.append(np.mean((y_test.values-y_test_pred)**2))

        reg_lambda /= 2
    # plot for Number of Non-zeros
    plt.figure(figsize=(7, 5))
    plt.plot(lambdas, nonzeros)
    plt.xscale('log')
    plt.title("$\lambda$ vs Non Zero Features")
    plt.xlabel('$\lambda$')
    plt.ylabel('Number of Non-zeros')
    plt.savefig('A6c.png')
    # plot for $\lambda$ vs Error
    plt.figure(figsize=(7, 5))
    plt.plot(lambdas, mse_train, label='Train')
    plt.plot(lambdas, mse_test, label='Test')
    plt.title("$\lambda$ vs Error")
```

```
    plt.xscale('log')
    plt.xlabel('$\lambda$')
    plt.ylabel('Mean Squared Error')
    plt.legend()
    plt.savefig('A6e.png')
    # plot for Regularization path, Feature weights vs $\lambda$
    plt.figure(figsize=(7, 5))
    vis_feat_name = ['agePct12t29', 'pctWSocSec', 'pctUrban', 'agePct65up', 'householdsize']
    for name in vis_feat_name:
        idx = X.columns.get_loc(name)
        weight_path = []
        for i, v in reg_path.items():
            weight_path.append(v[num])
        plt.plot(lambdas, weight_path, label=name)
    plt.title('Feature weights vs $\lambda$")
    plt.xscale('log')
    plt.xlabel('$\lambda$')
    plt.ylabel('Regularization Paths')
    plt.legend()
    plt.savefig('A6d.png')

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

A7:

- **Part a:**

$$\begin{aligned}
L(w) &= (y - Xw)^T(y - Xw) \\
L(w) &= (y^T - X^T w^T)(y - Xw) \\
L(w) &= y^T y - y^T Xw - X^T w^T y + X^T w^T Xw \\
\nabla_w L(w) &= \nabla_w (y^T y - y^T Xw - X^T w^T y + X^T w^T Xw) \\
\nabla_w L(w) &= 0 - y^T X - X^T y + 2X^T Xw \\
\nabla_w L(w) &= 2(X^T X)w - 2X^T y
\end{aligned}$$

- **Part b:**

$$\begin{aligned}
LHS &= ||w_{k+1} - w^*|| \\
&= ||w_k - \alpha \nabla_{w_k} L(w_k) - w^*|| \\
&= ||w_k - \alpha \nabla_{w_k} L(w_k) + \alpha \nabla_{w^*} L(w_k) - w^*|| \\
&= ||w_k - 2\alpha(X^T X)w_k + 2\alpha X^T y + 2\alpha(X^T X)w^* - 2\alpha X^T y - w^*|| \\
&= ||w_k - 2\alpha X^T X w_k + 2\alpha X^T X w^* - w^*|| \\
RHS &= ||(I - 2\alpha X^T X)(w_k - w^*)|| \\
&= ||Iw_k - Iw^* - 2\alpha X^T X w_k + 2\alpha X^T X w^*|| \\
&= ||w_k - 2\alpha X^T X w_k + 2\alpha X^T X w^* - w^*||
\end{aligned}$$

$$\begin{aligned}
\text{So, } LHS = RHS &= ||w_k - 2\alpha X^T X w_k + 2\alpha X^T X w^* - w^*|| \\
\text{Prove } ||w_{k+1} - w^*|| &= ||(I - 2\alpha X^T X)(w_k - w^*)||
\end{aligned}$$

- **Part c:**

$$\begin{aligned}
||w_{k+1} - w^*|| &\leq \rho ||w_k - w^*|| \\
||(I - 2\alpha X^T X)(w_k - w^*)|| &\leq \rho ||w_k - w^*|| \\
||(I - 2\alpha X^T X)|| ||(w_k - w^*)|| &\leq \rho ||w_k - w^*|| \\
||(I - 2\alpha X^T X)|| &\leq \rho \\
||I|| - ||2\alpha X^T X|| &\leq \rho \\
1 - ||2\alpha X^T X|| &\leq \rho
\end{aligned}$$

Because we have $||Ax|| \leq |\lambda| \cdot ||x||$, then

$$\begin{aligned}
||I - 2\alpha X^T X|| &\leq 1 - 2||H||||\alpha|| \\
1 - ||2\alpha X^T X|| &\leq 1 - 2||H||||\alpha|| \\
&and: \\
1 - ||2\alpha X^T X|| &\leq 1 - 2||h||||\alpha||
\end{aligned}$$

We also have $\rho = \max(|1 - 2\alpha h|, |1 - 2\alpha H|)$. As a result, we obtain:

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

A8:

- **Answer:**

  It takes around 18 hours.