CSE546 HW2

Kaiser Sun

May 2020

1 A.0. Conceptual Questions

(a)(check) Not necessarily. When we remove the feature "number of bathrooms," it is possible that there still exists related features, such as "number of showers" in the features we kept. Therefore, removing "number of bathrooms" will likely lead to a increase of weight of "number of showers", which helped preserved the performance.

(b) L1 norm will likely to result in a larger number of 0s.
By the theorem, for any $\lambda \geq 0$ for which $\hat{w}_r$ achieves the minimum, $\exists v \geq 0$ s.t. $\hat{w}_r = \arg\min_w \sum(y_i - x_i^T w)^2$ subject to $r(w) \leq v$.
Observe the penalty function $r(w)$ of L2 and L1 norm, we can see that when the region of constraint($||w||_1 \leq v$) for $\hat{w}$ hit the contour, a feature will always vanish for L1 norm. However, when the region of constraint produced by L2 norm($||w||_2^2 \leq v$) touches contour, it is possible for it to locate at any place of the coordinate system of w.

(c) Upside: It can help zero out some features(feature selection).
Downside: It will not give a closed-form solution if using this as regularizer.

(d) True. If the step size is too large, it will goes away from our optimal value. Thus it might not converge and even lead to worse results.

(e) Because the expectation of the stochastic gradient(produced by uniformly drew a data point from 1 to n) is equal to the gradient of loss over all data points.

(f) Advantage of SGD over GD: The time complexity of computing SGD is much less than that of GD.
Disadvantage of SGD over GD: SGD takes more steps converge.

2 A.1. Convexity and Norms

2.1 (a)

Proof: (i) Non-negativity: By the property of absolute value, $|x_i| \geq 0$. Thus, $f(x) = \sum_{i=1}^n x_i \geq 0$.

(ii) Absolute scalability: $f(ax) = \sum_{i=1}^n |ax_i| = a\sum_{i=1}^n |x_i| = af(x)$.

(iii) Triangle inequality: We want to prove that $f(x+y) \leq f(x) + f(y)$, that is, $\sum_{i=1}^n |x_i + y_i| \leq \sum_{i=1}^n |x_i| + \sum_{i=1}^n |y_i| = \sum_{i=1}^n |x_i| + |y_i|$.
We begin by proving the triangle inequality for real numbers.
By the properties of absolute value, we have $-|x_i| \leq x_i \leq |x_i|$ and $-|y_i| \leq y_i \leq |y_i|$.
$\therefore -(|x_i| + |y_i|) \leq x_i + y_i \leq |x_i| + |y_i|$, implying that $-(|x_i| + |y_i|) \leq |x_i + y_i| \leq |x_i| + |y_i|$.
$\forall i$, we have this property. Therefore, $f(x+y) = \sum_{i=1}^n |x_i + y_i| \leq \sum_{i=1}^n |x_i| + |y_i| = f(x) + f(y)$.

In conclusion, $f(x)$, which satisfies three properties of norm definition, is a norm. ■

2.2 (b)

Proof: Suppose $g(x)$ is a norm. Then it must satisfies triangle inequality, which is $g(x+y) = g(x) + g(y)$.
Pick $n = 2, x = (9, 16), y = (16, 9)$, we have $g(x+y) = (\sum_{i=1}^{n} |x_i + y_i|^{\frac{1}{2}})^2 = 100$.
However, $g(x) + g(y) = (\sum_{i=1}^{n} |x_i|^{\frac{1}{2}})^2 + (\sum_{i=1}^{n} |y_i|^{\frac{1}{2}})^2 = 49 + 49 = 98 \leq 100 = g(x+y)$, which contradicts to our assumption.
Therefore, $g(x)$ is not a norm. ■

3 B.1. Norms

We begin by induction on n.
Base Case: When $n = 1$, we have $||x||_1 = |x_1|$, $||x||_2 = |x_1|$, and $||x||_\infty = \max_{i=1} |x_i| = x_1$. This satisfy the inequality.

Induction Step: Suppose $||x||_\infty \leq ||x||_2 \leq ||x||_1$ holds for n, then we want to prove that it holds for $n+1$.
Let $||x||_1 = a_1, ||x||_2 = a_2, ||x||_\infty = a_\infty$ for dimension $= n$. By induction hypothesis we have $a_\infty \leq a_2 \leq a_1$
Then, $||x||_1 = a_1 + |x_{n+1}|$, $||x||_2 = \sqrt{a_2^2 + |x_{n+1}|^2}$, and $||x||_\infty = max\{a_\infty, |x_{n+1}|\}$.
$\because a_2 \geq a_\infty$, $\therefore \sqrt{a_2^2 + |x_{n+1}|^2} \geq a_2, |x_{n+1}| \geq max\{a_\infty, |x_{n+1}|\}$.
$\therefore ||x||_\infty \leq ||x||_2$.
For $||x||_1$ and $||x||_2$, we square them, resulting $||x||_1^2 = a_1^2 + x_{n+1}^2 + 2a_1|x_{n+1}| \geq a_2^2 + |x_{n+1}^2| = ||x||_2^2$.
Thus $||x||_1 \geq ||x||_2$.
Thus, in conclusion, $||x||_\infty \leq ||x||_2 \leq ||x||_1$. ■

4 A.2.

I. It is not convex. Connect point b and c, the line goes out of the shaded area.
II. It is convex. We cannot find any line segments created by connecting two points in the shaded set goes out of shaded area.
III. It is not convex. Connect a and d, the line goes out of the shaded area.

5 A.3.

(a) Function in panel I on $[a, c]$ is convex. From the shape we can observe that it looks like a bowl.
(b) Function in panel II on $[a, c]$ is not convex. Try connecting any point between a, b or between b, c, the line segment is below the function line.
(c) Function in panel III on $[a, d]$ is not convex. Try connecting any line between a, c, the line segment will below the function line.
(d) Function in panel III on $[c, d]$ is convex. Because we can observe that it looks like a bowl.

6 B.2.

6.1 (a)

To prove that $f(x)$ is convex, we want to prove that $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \forall x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$.
By the properties of norm, we know that $f((1-\lambda)x + \lambda y) \leq f((1-\lambda)x) + f(\lambda y) = |1-\lambda|f(x) + |\lambda|f(y)$.
Recall that $\lambda \in [0, 1]$, we have $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$.
Thus, $f(x)$ is convex. ■

6.2 (b)

Let set $S = \{x \in \mathbb{R}^n : ||x|| \leq 1\}$. Consider $x, y \in S$, and suppose $||x|| \leq ||y||$.
Take arbitrary $\lambda \in [0, 1]$, we have $(1-\lambda)||x|| + \lambda||y|| \leq (1-\lambda)||y|| + \lambda||y|| = ||y|| \leq 1$.

Therefore, we have $z \in \mathbb{R}^n$, s.t. $||z|| = (1-\lambda)||x|| + \lambda||y||$, and then $z \in S$.
Thus set S is a convex set. ■

6.3 (c)

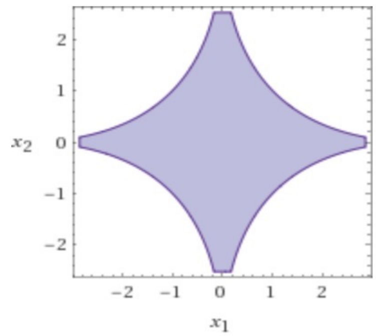

Figure 1: B2c Plot

As we can observe, the set is not convex. Take a point where $x_1 = 0$ and $x_2 = 2$, connect it with $x_1 = -3$ and $x_2 = 0$, the line segment will goes out of the blue-shaded area. Thus it is not convex.

7 B.3.

7.1 (a)

We first want to prove the **Claim**: If f, g are convex, then $f + g$ are convex.
Suppose f, g are convex and share domain of $\mathbb{R}^d$.
By convexity of f, g, we have $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$, $g((1-\lambda)x + \lambda y) \leq (1-\lambda)g(x) + \lambda g(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$.
Therefore $(f+g)((1-\lambda)x+\lambda y) = f((1-\lambda)x+\lambda y)+g((1-\lambda)x+\lambda y) \leq (1-\lambda)(f(x)+g(x))+\lambda(f(y)+g(y))$, which implies that $f + g$ is convex.

Then, since we know that $l_i(w)$ is convex $\forall i$.
$\because || \cdot ||$ is a norm and $\lambda > 0$, thus $\lambda||w|| = |\lambda|||w|| = ||\lambda w||$, which is also convex, as we proved in B.2.
Thus, by the claim, we know that $\sum_{i=1}^{n} l_i(w) + \lambda||w||$ is convex.

7.2 (b)

Because if we use the loss functions and regularized loss functions convex, we can confirm that the local minimum we found is actually the global minimum, which means that we minimize the loss.

8 A.4. LASSO

8.1 (a)

The plot is

8.2 (b)

8.3 (c)

9 A.5. LASSO(cont.)

9.1 (a)

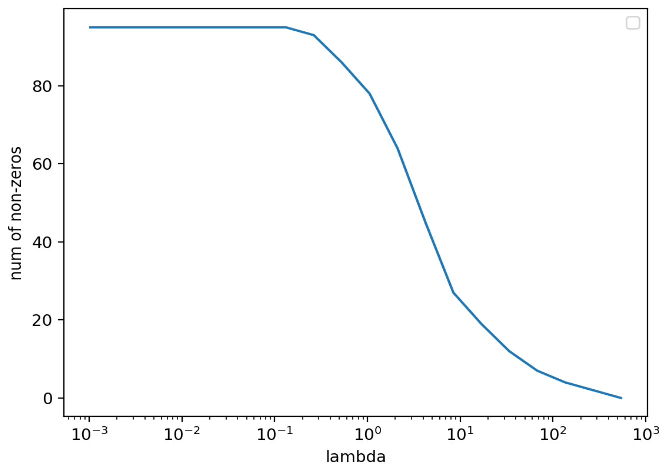

Figure 2: A5(a): Number of Nonzeros w.r.t.λ

9.2 (b)

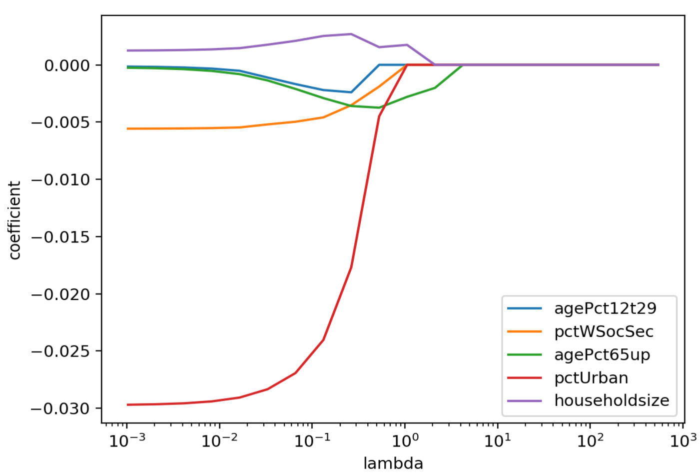

Figure 3: A5(b): Regularization Path

9.3 (c)

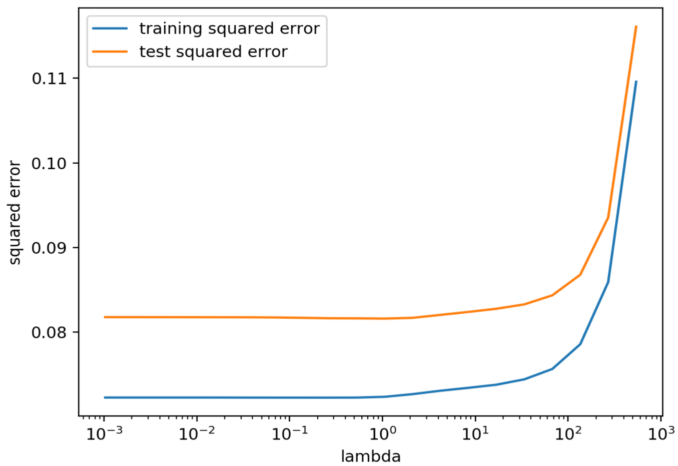

Figure 4: A5(c): Squared Error

```
# A.5.
# May 11, 2020 Kaiser Sun

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

class LASSO:

    def __init__(self, reg_lambda=1E-8, delta=0.005):
        """
        Constructor
        """
        self.reg_lambda = reg_lambda
        self.w = None
        self.b = 0.0
        # The stopping time of convergence
        self.delta = delta

    def fit(self, X, y):
        """
        Run coordinate descent
        Arguments:
            X is a n-by-d array
            y is a n-by-1 array
        Returns:
            No return value
        """
        # Initialize w of dim [d, 1]
        d = X.shape[1]
        n = X.shape[0]
        w_curr = np.zeros((d, 1))

        A = 2 * np.sum(np.power(X, 2), axis=0)  # Power and then sum up the rows
        # C = np.zeros((d, 1))
        # Iterate until convergence
        converged = False
        loss_now = np.sum(np.power((self.b * np.ones((n, 1)) + X.dot(w_curr) - y), 2)) +
    self.reg_lambda * np.sum(
            abs(w_curr))
```

```
        while not converged:
            self.b = np.average(y - X.dot(w_curr))
            # converged = True
            loss_prev = loss_now
            # Compute C, w_new
            prev_w = np.copy(w_curr)  # Diff conv
            for k in range(d):
                sliced_x = X[:, k]
                prev_wk = np.copy(w_curr[k])
                w_curr[k] = 0
                c_k = np.dot((y - (self.b * np.ones((n, 1)) + X.dot(w_curr))).T, sliced_x)
                if 2 * c_k + self.reg_lambda < 0:
                    w_curr[k] = (c_k * 2 + self.reg_lambda) / A[k]
                elif 2 * c_k - self.reg_lambda > 0:
                    w_curr[k] = (c_k * 2 - self.reg_lambda) / A[k]
                else:
                    w_curr[k] = 0
                # Check if converged
                # if abs(w_curr[k] - prev_wk) > self.delta:
                #     converged = False
            if sum(abs(w_curr - prev_w)) <= sum(abs(self.delta * prev_w)):
                converged = True
            # Sanity Check
            loss_now = np.sum(
                np.power((self.b * np.ones((n, 1)) + X.dot(w_curr) - y), 2)) + self.
    reg_lambda * np.sum(abs(w_curr))
            if (loss_now - loss_prev > 0):
                print("Loss increasing.", loss_now - loss_prev)
        self.w = w_curr

    def predict(self, X):
        """
        Use the trained model to predict the values
        Arguments:
            X is a n-by-d array
        Returns:
            an n-by-1 array of the predictions
        """
        return X.dot(self.w)

    def num_of_nonzeros(self):
        """
        Returns:
            the number of nonzeros a in w
        """
        return np.count_nonzero(self.w)

def create_lambdas(init_lambda, ratio, num):
    return init_lambda * (1 / ratio) ** np.arange(0, num)

if __name__ == '__main__':
    # Importing Data
    df_train = pd.read_table("crime-train.txt")
    df_test = pd.read_table("crime-test.txt")
    y_train = df_train['ViolentCrimesPerPop'].values.reshape(df_train.shape[0], 1)
    x_train = df_train.drop('ViolentCrimesPerPop', axis=1).values
    x_test = df_test.drop('ViolentCrimesPerPop', axis=1).values
    y_test = df_test['ViolentCrimesPerPop'].values.reshape(df_test.shape[0], 1)
    print("data import done")

    # Computing
    lambda_ratio = 2
    lambda_num = 20

    lambda_max = np.max(2 * np.abs(x_train.T.dot(y_train - np.average(y_train))))
    lambda_max_model = LASSO(lambda_max)
    lambda_max_model.fit(x_train, y_train)
    lambda_max_w = lambda_max_model.w
```

```
    print("lambda_max is ", lambda_max)
    lambdas = create_lambdas(lambda_max, lambda_ratio, lambda_num)
    print("lambdas are ", lambdas)
    models = [LASSO(reg_lambda) for reg_lambda in lambdas]
    for model in models:
        model.fit(x_train, y_train)
    # a
    num_of_nonzeros = [model.num_of_nonzeros() for model in models]
    trained_w = [model.w for model in models]
    print("lambdas non-zeros are", num_of_nonzeros)

#   Plot nonzeros
    plt.xscale('log')
    plt.xlabel("lambda")
    plt.ylabel("num of non-zeros")
    plt.plot(lambdas, num_of_nonzeros)
    plt.legend()
    plt.show()

    # b Plot reg path
    agePct12t29_index = df_train.columns.get_loc('agePct12t29')
    pctWSocSec_index = df_train.columns.get_loc('pctWSocSec')
    pctUrban_index = df_train.columns.get_loc('pctUrban')
    agePct65up_index = df_train.columns.get_loc('agePct65up')
    householdsize_index = df_train.columns.get_loc('householdsize')
    agePct12t29_coeffs = [item[agePct12t29_index] for item in trained_w]
    print(agePct12t29_coeffs)
    pctWSocSec_coeffs = [item[pctWSocSec_index] for item in trained_w]
    pctUrban_coeffs = [item[pctUrban_index] for item in trained_w]
    agePct65up_coeffs = [item[agePct65up_index] for item in trained_w]
    householdsize_coeffs = [item[householdsize_index] for item in trained_w]

    plt.plot(lambdas, agePct12t29_coeffs, label="agePct12t29")
    plt.plot(lambdas, pctWSocSec_coeffs, label="pctWSocSec")
    plt.plot(lambdas, agePct65up_coeffs, label="agePct65up")
    plt.plot(lambdas, pctUrban_coeffs, label="pctUrban")
    plt.plot(lambdas, householdsize_coeffs, label="householdsize")
    plt.xlabel("lambda")
    plt.xscale('log')
    plt.ylabel("coefficient")
    plt.legend()

    # c Plot Errors
    y_train_predicted = np.asarray([model.predict(x_train) for model in models])
    y_test_predicted = np.asarray([model.predict(x_test) for model in models])
    train_squared_error = np.average(np.power(y_train_predicted - y_train, 2), axis=1).squeeze(axis=1)
    test_squared_error = np.average(np.power(y_test_predicted - y_test, 2), axis=1).squeeze(axis=1)
    plt.plot(lambdas, train_squared_error, label="training squared error")
    plt.plot(lambdas, test_squared_error, label="test squared error")
    plt.xlabel("lambda")
    plt.xscale('log')
    plt.ylabel("squared error")
    plt.legend()

    # d
    test_lambda = 30
    test_model = LASSO(test_lambda)
    test_model.fit(x_train, y_train)
    print("training complete")
    test_w = test_model.w.squeeze()
    top_positive_indices = test_w.argsort()[-1:][::-1]
    top_negative_indices = test_w.argsort()[:1][::-1]
    print("top_positive_indices is", top_positive_indices, " with name ", df_train.columns[top_positive_indices+1], "value is ", test_w[top_positive_indices])
    print("top_negative_indices is", top_negative_indices, " with name ", df_train.columns[top_negative_indices+1], "value is ", test_w[top_negative_indices])
```

10 A.6. Logistic Regression

10.1 (a)

The gradients are:

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

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

10.2 (b)

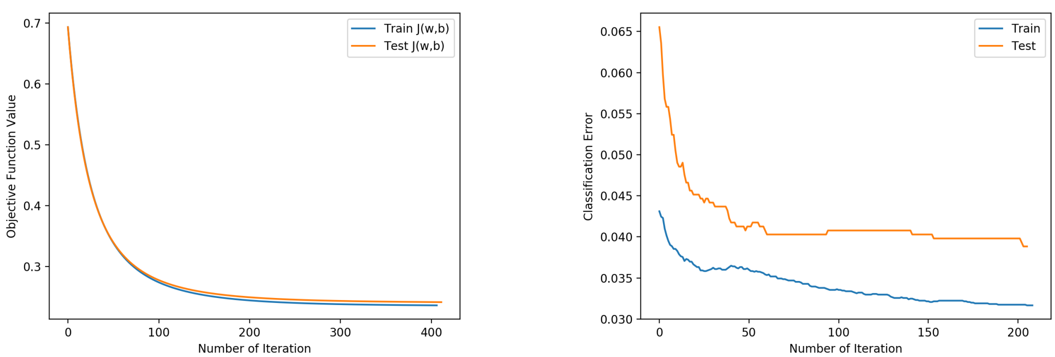

Figure 5: Plots with $\delta = 10^{-5}$, step size $= 0.01$, $\lambda = 0.1$

Notice that here I removed the first value of error, to make the plot more recognizable.

10.3 (c)

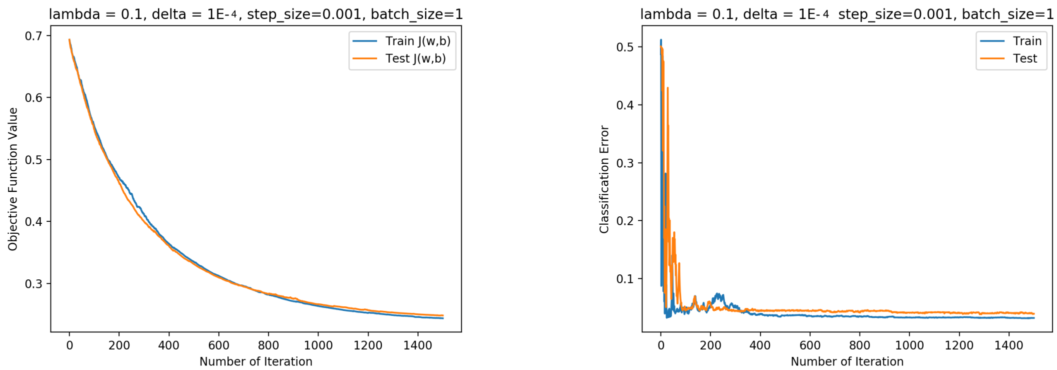

Figure 6: Plots with $\delta = 10^{-5}$, step size $= 0.01$, $\lambda = 0.1$

10.4 (d)

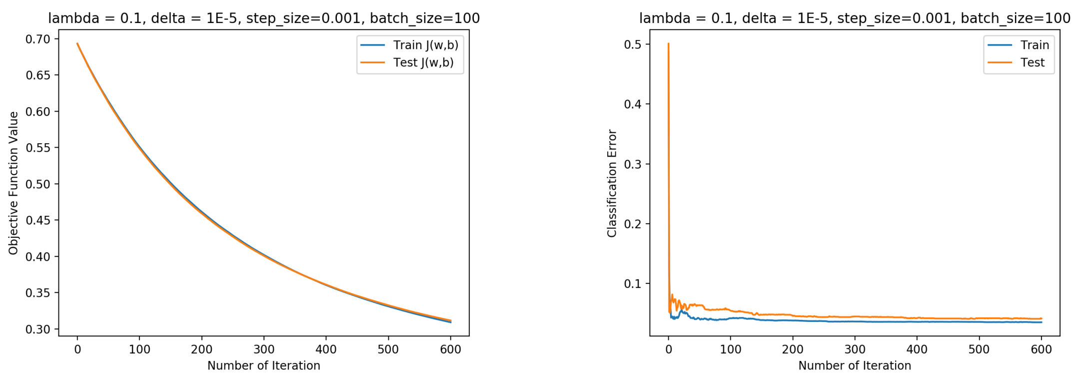

Figure 7: Plots with $\delta = 10^{-5}$, step size $= 0.01$, $\lambda = 0.1$