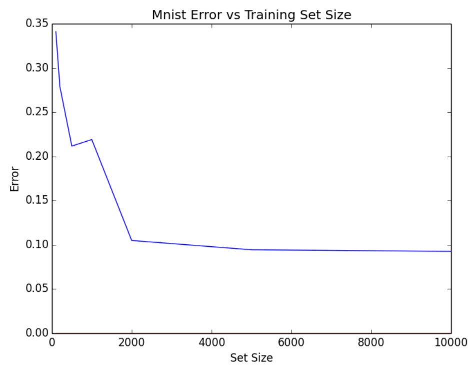
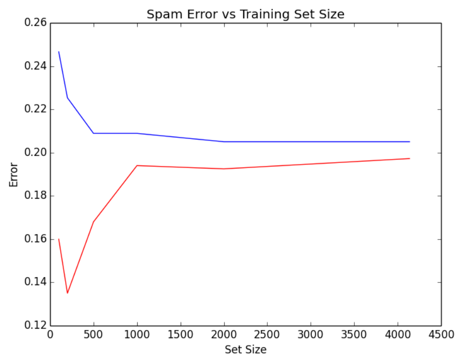
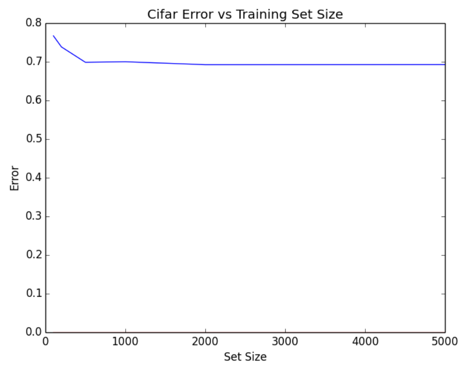
Paul Kim 24699265. Discussed with Caren Thomas, Karan Warrier and Timi Fasubaa

Homework 1 Write-up

Problem 2: Plots







The training set error is in red while the validation set error is in blue. For the cifar and mnist datasets, the training error is 0, which is due to a high C value. However, since hyperparameter tuning was not in the scope of this problem, the linear SVC was ran with no modifications, leading to zero training error.

Problem 3:

C values at powers of ten were tested. Because extreme C-values are slow to compute, the first two trials were performed on 1000 data points. First I tried from 10^-7 to 10^8, which gave scores of 0.8796 for 10^-7, 0.8909 for 10^-6, and 0.8827 for all other C-values. I subsequently tried C values from 10^-14 to 10^-6. This gave scores of 0.66806 for 10^-8, 0.8796 for 10^-7, 0.8909 for 10^-6, and 0.2057 for all other values.

I subsequently tried C-values of 10^-7, 10^-6 and 10^-5 on 10,000 data points. This gave scores of

10^-7	10^-6	10^-5
0.92319231923192324	0.93149314931493155	0.91629162916291629

Therefore, 10^-6 was chosen as the C-value for the mnist dataset.

Problem 4:

5-fold cross-validation was used for the spam dataset. Values tested were between 10^-9 and 10^2. For values 10^-4 and smaller, the score was 0.7099.

10^-3	10^-2	10^-1	1	10	100
0.7172	0.7505	0.7787	0.7971	0.8031	0.8037

C of 100 was chosen.

Problem 5:
My Kaggle name is PaulKim, and I got a score of 0.926 on the Mnist dataset and 0.841 on the Spam dataset.

Appendix: Code

From DataLoad.py:

```python
from scipy.io import loadmat
from numpy.random import shuffle
from numpy import concatenate
import numpy as np

# loading data from file
mnist_raw = np.array(loadmat("hw01_data/mnist/train.mat").get('trainX'))
mnist_test = np.array(loadmat("hw01_data/mnist/test.mat").get('testX'))

spam = loadmat("hw01_data/spam/spam_data.mat")
spam_train_data = spam.get('training_data')
spam_train_labels = spam.get('training_labels')
spam_test_data = spam.get('test_data')

cifar_train = loadmat("hw01_data/cifar/train.mat").get('trainX')
cifar_test = loadmat("hw01_data/cifar/test.mat").get('testX')


# Problem 1: Data Partitioning

# shuffling the data:

shuffle(mnist_raw)

spam_raw = np.array(concatenate((spam_train_data, spam_train_labels.T), axis=1))
shuffle(spam_raw)
shuffle(cifar_train)

# make training and validation sets
mnist_valid = mnist_raw[:9999]
mnist_train = mnist_raw[10000:]

length_spam = spam_raw.shape[0]
spam_valid = spam_raw[:length_spam*0.2]
spam_train = spam_raw[(length_spam*0.2) + 1:]

cifar_valid = cifar_train[:4999]
cifar_train = cifar_train[5000:]

mnist_valid = np.array(mnist_valid)
mnist_train = np.array(mnist_train)

spam_valid = np.array(spam_valid)
spam_train = np.array(spam_train)

cifar_valid = np.array(cifar_valid)
cifar_train = np.array(cifar_train)
```

From SVM.py:

```
from DataLoad import *
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np
import sklearn.model_selection


# mnist SVC
mnist_training_sizes = [100, 200, 500, 1000, 2000, 5000, 10000]
mnist_train_error = [0, 0, 0, 0, 0, 0, 0]
mnist_error = [0, 0, 0, 0, 0, 0, 0]


for i in range(0, 7):
    mnist_mac = SVC(kernel='linear')
    train = mnist_train[:mnist_training_sizes[i], 0:783]
    label = mnist_train[:mnist_training_sizes[i], 784]
    mnist_mac.fit(train, label)
    mnist_train_error[i] = 1 - mnist_mac.score(train, label)
    valid = mnist_valid[:, 0:783]
    valid_lab = mnist_valid[:, 784]
    mnist_error[i] = 1 - mnist_mac.score(valid, valid_lab)

plt.plot(mnist_training_sizes, mnist_error, 'b-')
plt.plot(mnist_training_sizes, mnist_train_error, 'r-')
plt.ylabel('Error')
plt.xlabel('Set Size')
plt.title('Mnist Error vs Training Set Size')
plt.show()


# spam SVC
spam_training_sizes = [100, 200, 500, 1000, 2000, len(spam_train)]
spam_train_error = [0, 0, 0, 0, 0, 0]
spam_error = [0, 0, 0, 0, 0, 0]
```

```python
for i in range(0, 6):
    spam_mac = SVC(kernel='linear')
    train = spam_train[:spam_training_sizes[i], 0:len(spam_train[0]) - 1]
    label = spam_train[:spam_training_sizes[i], len(spam_train[0]) - 1]
    spam_mac.fit(train, label)
    spam_train_error[i] = 1 - spam_mac.score(train, label)
    valid = spam_valid[:, 0:len(spam_train[0]) - 1]
    valid_lab = spam_valid[:, len(spam_train[0]) - 1]
    spam_error[i] = 1 - spam_mac.score(valid, valid_lab)

plt.plot(spam_training_sizes, spam_error, 'b-')
plt.plot(spam_training_sizes, spam_train_error, 'r-')
plt.ylabel('Error')
plt.xlabel('Set Size')
plt.title('Spam Error vs Training Set Size')
plt.show()

# cifar SVC
cifar_training_sizes = [100, 200, 500, 1000, 2000, 5000]
cifar_train_error = [0, 0, 0, 0, 0, 0]
cifar_error = [0, 0, 0, 0, 0, 0]

for i in range(0, 6):
    cifar_mac = SVC(kernel='linear')
    train = cifar_train[:cifar_training_sizes[i], 0:len(cifar_train[0]) - 1]
    label = cifar_train[:cifar_training_sizes[i], len(cifar_train[0]) - 1]
    cifar_mac.fit(train, label)
    cifar_train_error[i] = 1 - cifar_mac.score(train, label)
    valid = cifar_valid[:, 0:len(cifar_train[0]) - 1]
    valid_lab = cifar_valid[:, len(cifar_train[0]) - 1]
    cifar_error[i] = 1 - cifar_mac.score(valid, valid_lab)

plt.plot(cifar_training_sizes, cifar_error, 'b-')
plt.plot(cifar_training_sizes, cifar_train_error, 'r-')
plt.ylabel('Error')
plt.xlabel('Set Size')
plt.title('Cifar Error vs Training Set Size')
plt.show()


# Problem 3
# C-value

error_values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

# First pass:
for i in range(0, 15):
    mnist_mac = SVC(kernel='linear', C=(pow(10, (i-7))))
    train = mnist_train[:1000, 0:783]
    label = mnist_train[:1000, 784]
    mnist_mac.fit(train, label)
    valid = mnist_valid[:, 0:783]
    valid_lab = mnist_valid[:, 784]
    error_values[i] = mnist_mac.score(valid, valid_lab)

# Ok so interesting C-values happen at the low end of the scale

# Second pass:
```

```
error_val_2 = [0, 0, 0, 0, 0, 0, 0, 0, 0]
for i in range(0, 9):
    mnist_mac = SVC(kernel='linear', C=pow(10, (i-14)))
    train = mnist_train[:1000, 0:783]
    label = mnist_train[:1000, 784]
    mnist_mac.fit(train, label)
    valid = mnist_valid[:, 0:783]
    valid_lab = mnist_valid[:, 784]
    error_val_2[i] = mnist_mac.score(valid, valid_lab)

# 10^-5, 10^-6 and 10^-7 are the best. Now testing on 10000 data points

error_val_3 = [0, 0, 0]
for i in range(0, 3):
    mnist_mac = SVC(kernel='linear', C=pow(10, (i-7)))
    train = mnist_train[:10000, 0:783]
    label = mnist_train[:10000, 784]
    mnist_mac.fit(train, label)
    valid = mnist_valid[:, 0:783]
    valid_lab = mnist_valid[:, 784]
    error_val_3[i] = mnist_mac.score(valid, valid_lab)

# best is 10^-6.

# Problem 4
# Cross-Validation
cv_train = concatenate((spam_train, spam_valid), axis=0)  # we can use the entire set

error_values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

for i in range(0, 11):
    cv_mac = SVC(kernel='linear', C=(pow(10, (i-9))))
    train = cv_train[:, 0:len(cv_train[0]) - 1]
    label = cv_train[:, len(cv_train[0]) - 1]
    error_values[i] = np.mean(sklearn.model_selection.cross_val_score(cv_mac, train,
label, cv=5))


# Anything C larger than 100 takes a prohibitive length of time, so I will use C = 100
```

From Submit.py

```
# For Kaggle submissions, using best SVM's

# MNIST best SVM had C of 10^-6, spam had C of 100

from DataLoad import *
from sklearn.svm import SVC
import numpy as np
import pandas as pd

# mnist
mnist_mac = SVC(kernel='linear', C=pow(10, -6))
mnist_train = mnist_raw[np.random.choice(mnist_train.shape[0], 10000)]
mnist_X = mnist_train[:, 0:783]
mnist_c = mnist_train[:, 784]

mnist_mac.fit(mnist_X, mnist_c)
```

```
mnist_predict = mnist_mac.predict(mnist_test[:, 0:783])

d = {
    "Id": np.arange(0, len(mnist_predict)),
    "Category": mnist_predict
}
df = pd.DataFrame(data=d)
df.to_csv("mnist_predict.csv", index=False)

# spam
spam_mac = SVC(kernel='linear', C=100)
spam_X = spam_raw[:, 0:len(spam_raw[0]) - 1]
spam_c = spam_raw[:, len(spam_raw[0]) - 1]

spam_mac.fit(spam_X, spam_c)

spam_predict = spam_mac.predict(spam_test_data)

d = {
    "Id": np.arange(0, len(spam_predict)),
    "Category": spam_predict
}
df = pd.DataFrame(data=d)
df.to_csv("spam_predict.csv", index=False)
```