Cory Sweet
Math 251
Homework 5

Cory Sweet
Math 251
HW 5

1a) $\min_{x,y} (y - x^2)$ subject to $4 - x^2 - y^2 \geq 0$

$g(x,y) \geq b$

$f(x,y) = y - x^2$ $\qquad$ $g(x,y) = 4 - x^2 - y^2$

$f_x = -2x$ $\qquad$ $g_x = -2x$

$f_y = 1$ $\qquad$ $g_y = -2y$

$\nabla f = \lambda \nabla g$, $\lambda(4 - x^2 - y^2) = 0$, $\lambda \geq 0$, $g \geq 0$

$-2x = \lambda(-2x)$, $1 = -2\lambda y$

$\boxed{1 = \lambda}$, $\frac{-1}{2\lambda} = y$, $\qquad$ $4 - x^2 - (\frac{-1}{2})^2 = 0$

$\boxed{-\frac{1}{2} = y}$ $\qquad$ $4 - \frac{1}{4} = x^2$

$x = \pm\sqrt{\frac{15}{4}}$

min @ $(-\sqrt{\frac{15}{4}}, \frac{-1}{2})$ and $(\sqrt{\frac{15}{4}}, \frac{-1}{2})$.

min is $-\frac{1}{2} - \frac{15}{4} = -4.25$

1b) $\min_{x} \frac{1}{2}x^2$ subject to $2x-1 \geq 0$

single: $f = \frac{1}{2}x^2$ $g = 2x-1$

$\nabla f = \lambda \nabla g$ $2x-1 \geq 0$, $\lambda \geq 0$

$2x = \lambda \cdot 2$ $x \geq 1/2$

$x = \lambda$

min @ $x = 1/2$ $f(1/2) = \boxed{1/8}$

dual: $L = \frac{1}{2}x^2 - \lambda(2x-1) = \frac{1}{2}x^2 - 2x\lambda + \lambda$

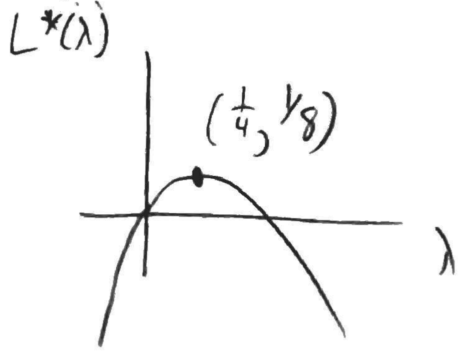
$$\frac{\partial L}{\partial x} = x - 2\lambda = 0$$

$x = 2\lambda$

$$L^*(\lambda) = \min_{x} L = \frac{1}{2}(2\lambda)^2 - \lambda(\overset{4\lambda - 1}{2(2\lambda)-1}) = 2\lambda^2 - 4\lambda^2 + \lambda = -2\lambda^2 + \lambda$$

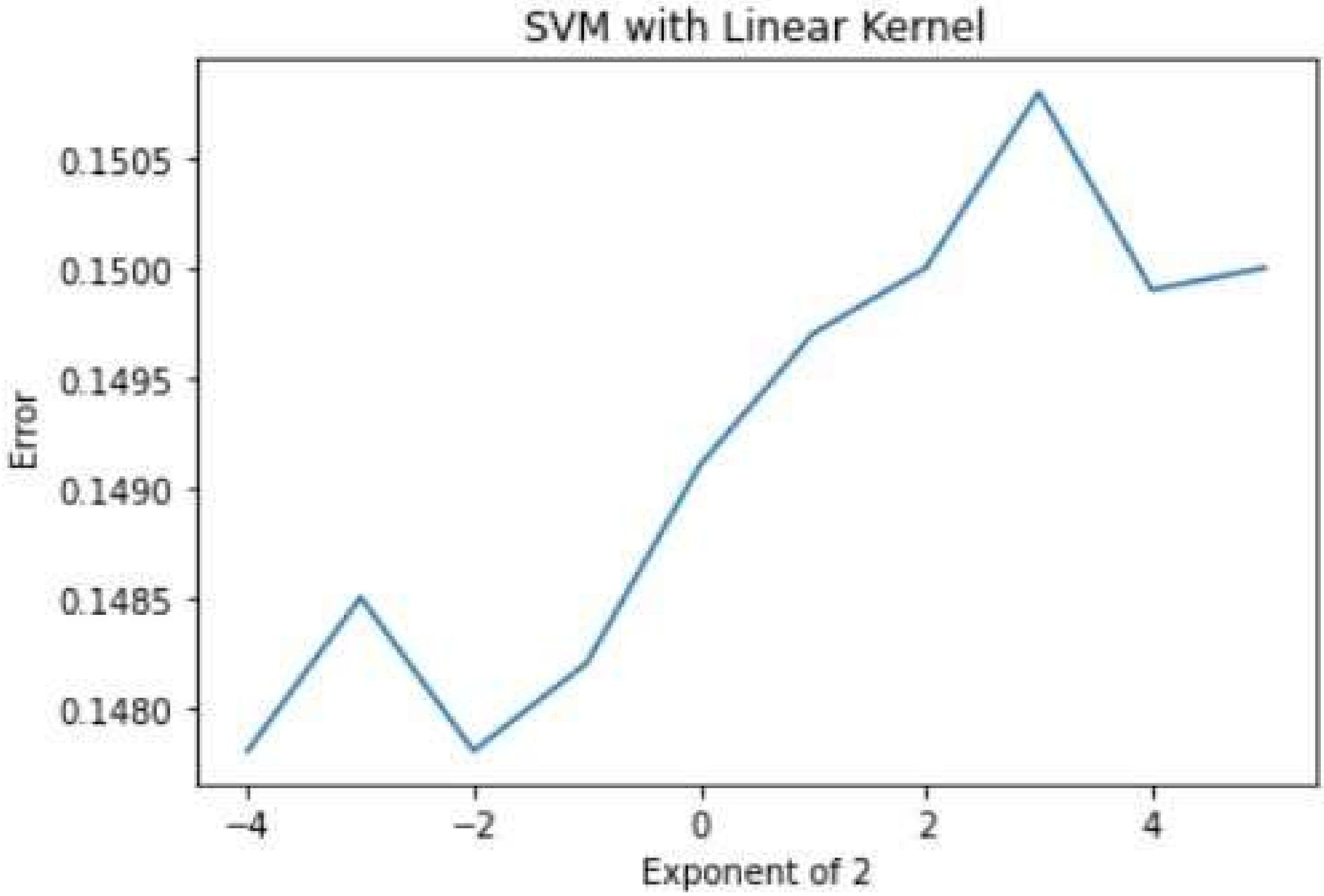
$L^*(\lambda) = -2\lambda^2 + \lambda$

$\max_{\lambda} L^*\lambda = \boxed{1/8}$

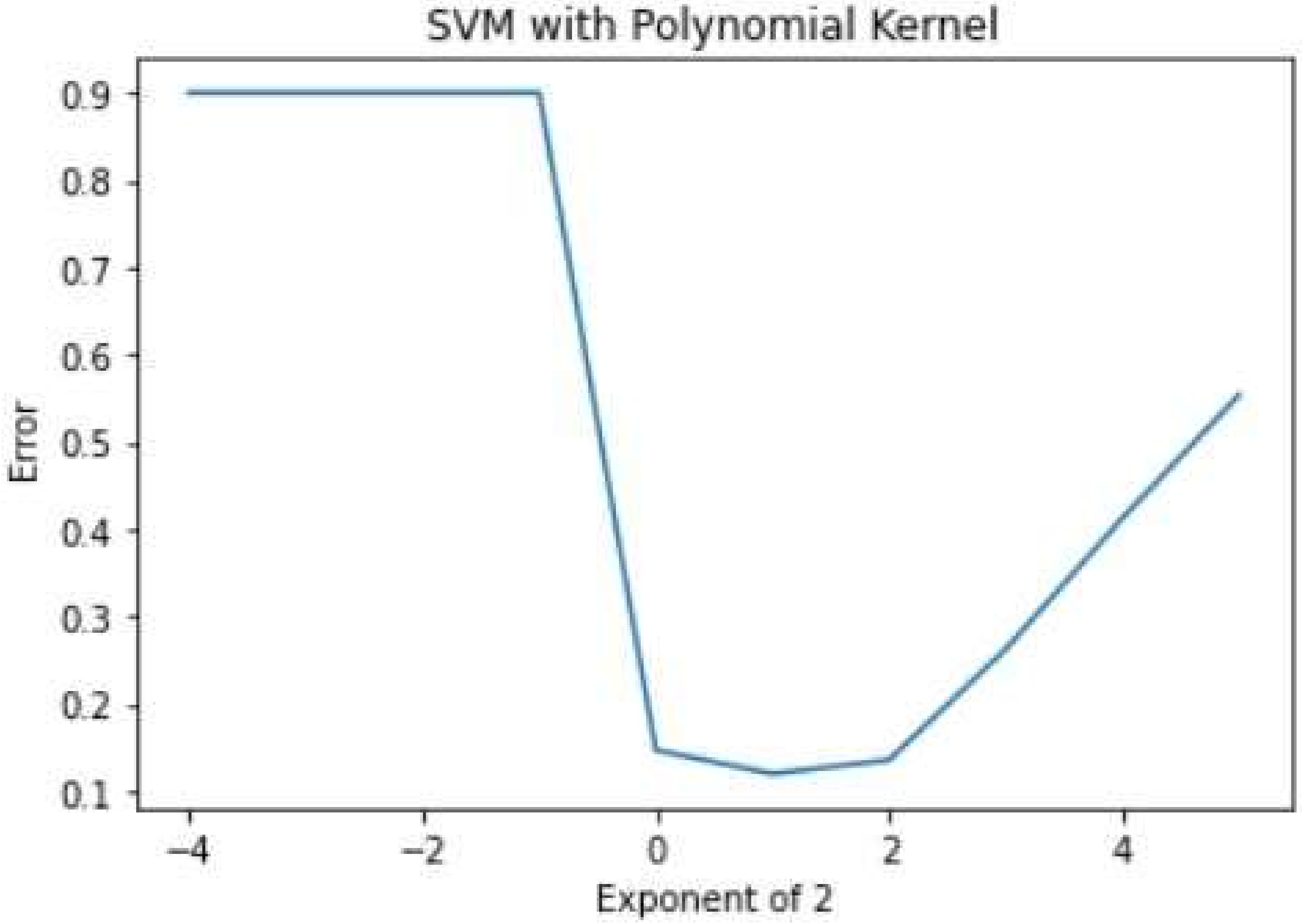


Same min for primal
& dual problem

Q2 Linear SM

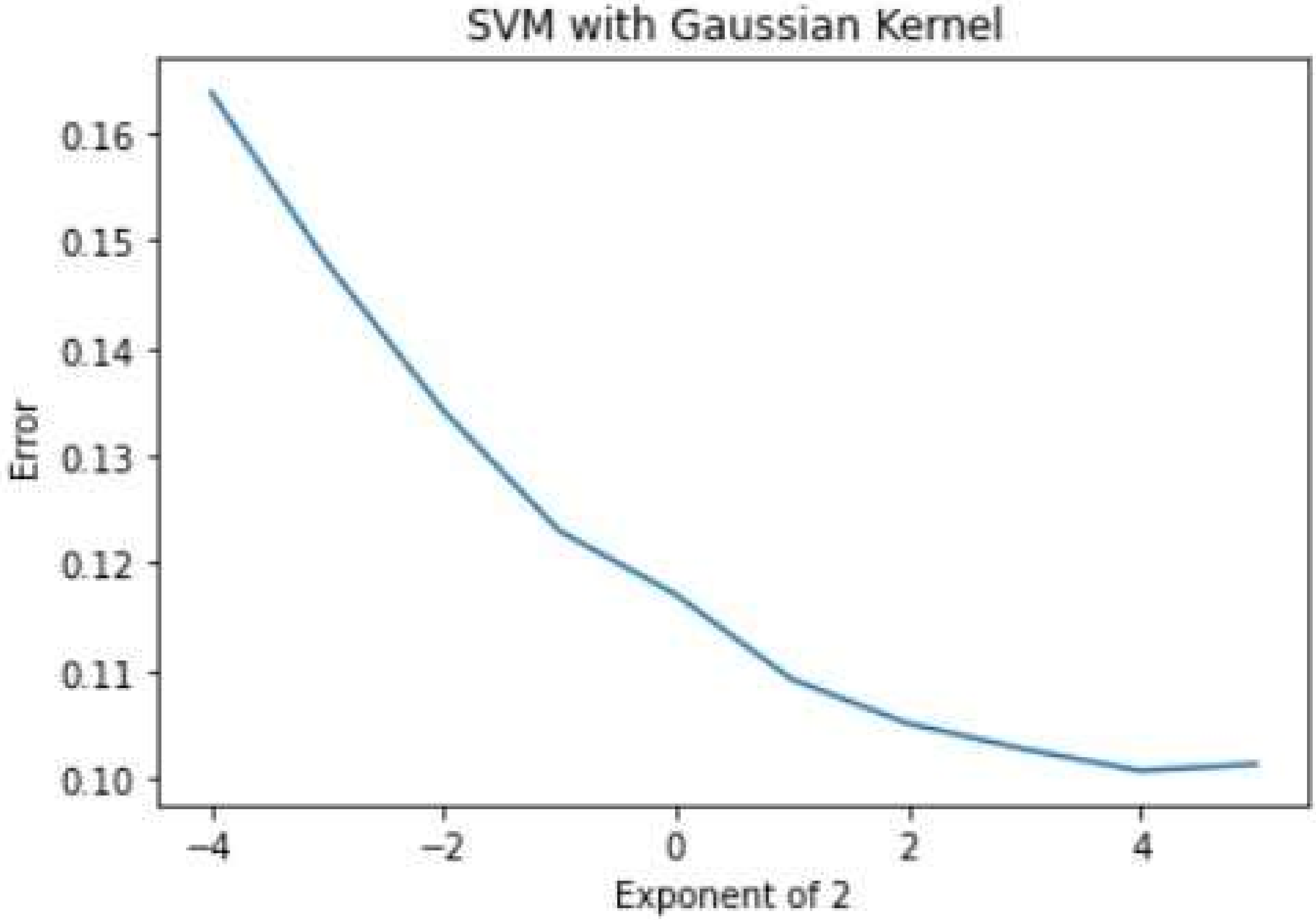

SVM with Linear Kernel
Error
0.1480
0.1485
0.1490
0.1495
0.1500
0.1505
−4
−2
0
2
4
Exponent of 2

Q3



I’m not sure why the error was so bad for C < 1

Q4


SVM with Gaussian Kernel
Error
0.16
0.15
0.14
0.13
0.12
0.11
0.10
−4
−2
0
2
4
Exponent of 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from sklearn.metrics import accuracy_score
import sklearn.decomposition
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from time import time
import math as math

fashion_mnist = keras.datasets.fashion_mnist;
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data();
"""
0       T-shirt/top
1       Trouser
2       Pullover
3       Dress
4       Coat
5       Sandal
6       Shirt
7       Sneaker
8       Bag
9       Ankle boot
"""

label_names=['T-shirt/top','Trouser','Pullover','Dress','Coat','Sandal',
       'Shirt','Sneaker','Bag','Ankle boot']

X_train = np.zeros([60000,784])
for i in range(60000):
  img=train_images[i,:,:]
  X_train[i,:] = img.reshape([784])

X_test = np.zeros([10000,784])
for i in range(10000):
  img=test_images[i,:,:]
  X_test[i,:] = img.reshape([784])

#Standardization
train_mean = np.mean(X_train,axis=0)
test_mean = np.mean(X_test,axis=0)
train_s = np.std(X_train,axis=0)
test_s = np.std(X_test,axis=0)

X_train = (X_train - train_mean)/train_s
```

```
X_test = (X_test - test_mean)/test_s

col_means = np.mean(X_train, axis = 0)
X_tilda = X_train - col_means
X_test_centered = X_test - col_means

k=190
PCA = sklearn.decomposition.PCA(n_components = k)
PCA.fit(X_tilda)
Y_train = PCA.transform(X_tilda)
Y_test = PCA.transform(X_test_centered)

#2

#SVM
Cs =  [-4,-3,-2,-1,0,1,2,3,4,5]
scores = []

for e in Cs:
    C = 2**e
    SVM = sklearn.svm.SVC(kernel = 'linear',C = C)
    SVM.fit(Y_train, train_labels)
    preds = SVM.predict(Y_test)
    score = accuracy_score(y_true = test_labels, y_pred = preds)
    scores.append(score)


error = 1-np.array(scores)

plt.plot(Cs, error)
plt.xlabel('Exponent of 2')
plt.ylabel('Accuracy')
plt.title('SVM with Linear Kernel')

scores =[0.8522,0.8515,0.8522,0.8518,0.8509,
        0.8503,0.85  ,0.8492,0.8501,0.85  ]
error = 1-np.array(scores)
Cs =  [-4,-3,-2,-1,0,1,2,3,4,5]


plt.plot(Cs, error)
plt.xlabel('Exponent of 2')
plt.ylabel('Error')
plt.title('SVM with Linear Kernel')

#3

Cs = [-4,-3,-2,-1,0,1,2,3,4,5]
```

```
scores = []

for x,e in enumerate(Cs):
    C = 2**e
    SVM = sklearn.svm.SVC(kernel = 'poly',degree=3, C=C)
    SVM.fit(Y_train, train_labels)
    preds = SVM.predict(Y_test)
    score = accuracy_score(y_true = test_labels, y_pred = preds)
    print(score)
    scores.append(score)

error = 1-np.array(scores)

plt.plot(Cs, error)
plt.xlabel('Exponent of 2')
plt.ylabel('Error')
plt.title('SVM with Polynomial Kernel')

Cs =  [-4,-3,-2,-1,0,1,2,3,4,5]
scores = []
for e in Cs:
    C = 2**e
    SVM = sklearn.svm.SVC(kernel = 'rbf', gamma = 'scale',C=C) #gamma = 1/(2*sigma^2)
    SVM.fit(Y_train, train_labels)
    preds = SVM.predict(Y_test)
    score = accuracy_score(y_true = test_labels, y_pred = preds)
    scores.append(score)


error = 1-np.array(scores)

plt.plot(Cs, error)
plt.xlabel('Exponent of 2')
plt.ylabel('Error')
plt.title('SVM with Gaussian Kernel')
```