ECE-GY 6143 INTRO TO MACHINE LEARNING

HOMEWORK-3

Pratyush Shukla (ps4534)

Question 1:

19/10/21

HW3

1) A) a) Proof →

$$K_{ij} = K(x_i, x_j) = \phi(x_i)\,\phi(x_j)$$

$$c^T K_c = \sum_i \sum_j c_i c_j K_{ij} = \sum_i \sum_j c_i c_j \phi(x_i)\phi(x_j)$$

$$= \left(\sum_i c_i \phi(x_i)\right)\left(\sum_i c_i \phi(x_i)\right)$$

$$c^T K_c = \left|\left|\sum_i c_i \phi(x_i)\right|\right|^2 \geq 0$$

$$k(x, y) = \alpha k_1(x, y) + \beta k_2(x, y)$$

$$= \langle \sqrt{\alpha}\,\phi_1(x), \sqrt{x}\,\phi_1(y)\rangle +$$

$$\langle \sqrt{\beta}\,\phi_2(x), \sqrt{\beta}\,\phi_2(y)\rangle$$

$$= \langle [\sqrt{\alpha}\,\phi_1(x), \sqrt{\beta}\,\phi_2(x)], [\sqrt{\alpha}\,\phi_1(y), \sqrt{\beta}\,\phi_2(y)]\rangle$$

b) <u>Proof</u> $\rightarrow$ Let $f_i(x)$ be ith feature value under feature map ϕ, $g_i(x)$ be in feature value under feature map ϕ_2.

$$k(x,y) = k_1(x,y)\,k_2(x,y)$$

$$= \left(\phi_1(x) \cdot \phi_1(y)\right)\left(\phi_2(x) \cdot \phi_2(y)\right)$$

$$= \left[\sum_{i=0}^{\infty} f_i(x) f_i(y)\right]\left[\sum_{j=0}^{\infty} g_i(x)\, g_j(y)\right]$$

$$= \left[\sum_{i,j} f_i(x) f_i(y)\, g_j(x)\, g_j(y)\right]$$

$$= \sum_{i,j} \left(f_i(x)\, g_j(x)\right)\left(f_i(y)\, g_j(y)\right)$$

$$k(x,y) = \langle \phi_3(x), \phi_3(y) \rangle$$

where ϕ_3 has feature

$$h_{ij}(x) = f_i(x)\, g_j(x)$$

c) Since each polynomial term is a product coefficient, the proof as follows from part a) & b).

d) By Taylor expansion -

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Then the proof follows part c).

B) We wish to show that the kernel $k(x,y) = e^{-\frac{1}{2}\|x-y\|^2}$ can be written as an inner product between some mapping ϕ on x and y, in other words, $k(x,y) = \langle \phi(x), \phi(y) \rangle$. Assume that $x, y \in R^d$. Consider the formula for $\phi_2(x) = \left(\frac{\pi}{2}\right)^{-d/4} e^{(-\|x-z\|^2)}$ which is an infinite dimensional function over $z \in R^d$

(rather than a finite dimensional vector with z being a discrete index). Similarly we have $\phi_z(y) = \left(\frac{\pi}{2}\right)^{-d/4} e^{(-\|y-z\|^2)}$. Then, we define the kernel as

$$k(x,y) = \langle \phi_z(x), \phi_z(y) \rangle$$

$$= \int_z \phi_z(x) \times \phi_z(y)\, dz \,.$$

This integral evaluates to

$$k(x,y) = \int_z \left(\frac{\pi}{2}\right)^{-d/4} e^{(-\|x-z\|^2)} \left(\frac{\pi}{2}\right)^{-d/4} e^{(-\|y-z\|^2)} dz$$

$$= \left(\frac{\pi}{2}\right)^{-d/2} \int_z e^{(-x^Tx - z^Tz + 2x^Tz)} e^{(-y^Ty - z^Tz + 2y^Tz)} dz$$

$$= \left(\frac{\pi}{2}\right)^{-d/2} e^{(-x^Tx - y^Ty)} \int_z e^{(-2z^Tz + 2(y+x)^Tz)} dz$$

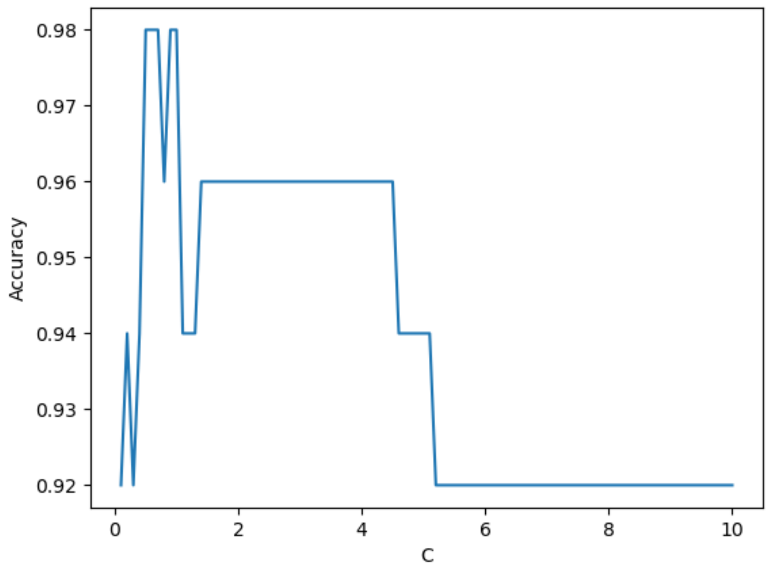
Question 2:

```
from sklearn.svm import SVC

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from scipy.io import loadmat

import numpy as np

import matplotlib.pyplot as plt


data = loadmat('dataset.mat')

X, y = data['X'], data['Y']

y = np.where(y==0, -1, 1)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)


score = []

C = [x*0.1 for x in range(1,101)]

print('LINEAR KERNEL ----> Best Estimator:')


for i in range(len(C)):

    clf = SVC(C=C[i], kernel='linear')

    clf.fit(X_train, y_train.ravel())

    y_predict = clf.predict(X_test)

    score.append(accuracy_score(y_test, y_predict))


idx = score.index(max(score))

print('C={:.2f}, Accuracy={}'.format(C[idx], score[idx]*100))


fig = plt.figure()

ax = plt.axes()

ax.plot(C, score)

ax.set_xlabel('C')

ax.set_ylabel('Accuracy')
```

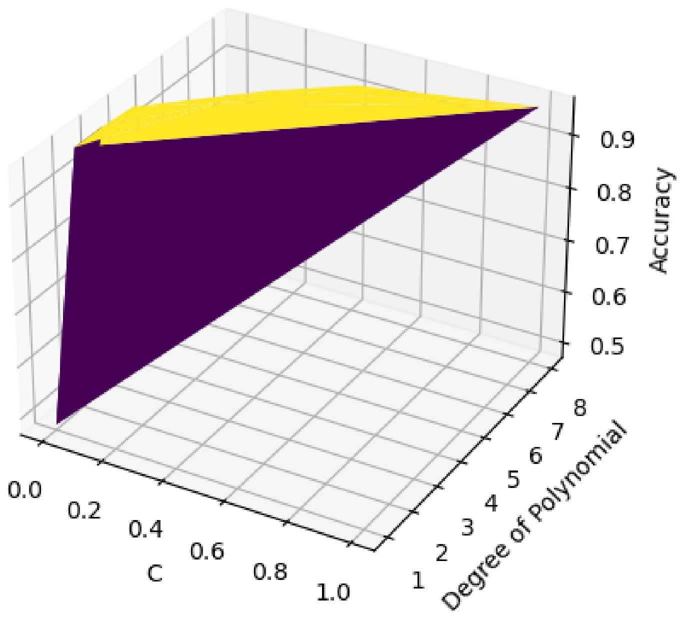
```
score = []

C = [0.0001, 0.0009, 0.001, 0.01, 0.21, 0.301, 0.49, 1]

degree = [x for x in range(1,9)]

print()

print('POLY KERNEL ----> Best Estimator:')

for i in range(len(C)):

    clf = SVC(C=C[i], degree=degree[i], gamma=0.9, kernel='poly')

    clf.fit(X_train, y_train.ravel())

    y_predict = clf.predict(X_test)

    score.append(accuracy_score(y_test, y_predict))


idx = score.index(max(score))

print('C={}, Degree of Polynomial={}, Gamma=0.9, Accuracy={}'.format(C[idx], degree[idx],
score[idx]*100))

fig = plt.figure()

ax = plt.axes(projection='3d')

ax.plot_trisurf(C, degree, score, cmap='viridis')

ax.set_xlabel('C')

ax.set_ylabel('Degree of Polynomial')

ax.set_zlabel('Accuracy')


fig = plt.figure()

ax = plt.axes(projection='3d')

dx = np.ones(len(C))

dy = np.ones(len(C))

dz = np.ones(len(C))

score, dz = dz, score

ax.bar3d(C, degree, score, dx, dy, dz, shade=True)

ax.set_xlabel('C')

ax.set_ylabel('Degree of Polynomial')

ax.set_zlabel('Accuracy')
```

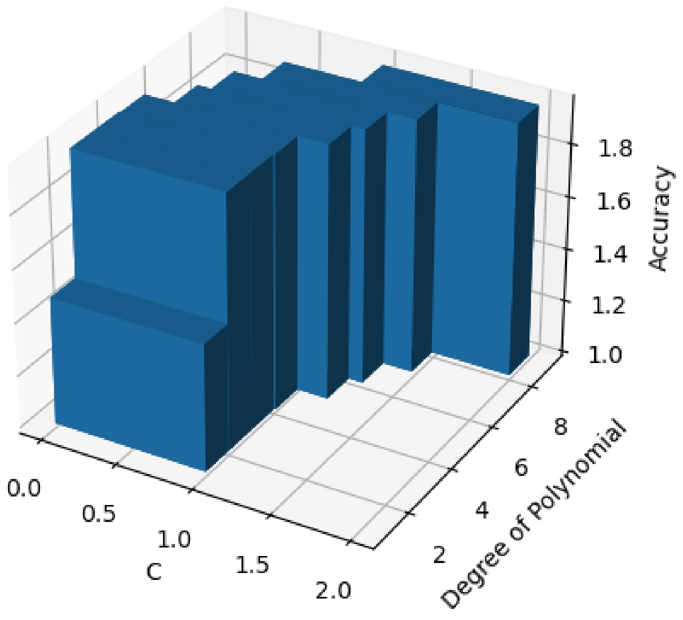
```
score = []
C = [0.01, 0.1, 0.5, 1, 10, 50, 100]
gamma = [0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9]
print()
print('RBF KERNEL ----> Best Estimator:')
for i in range(len(C)):
    clf = SVC(C=C[i], gamma=gamma[i], kernel='rbf')
    clf.fit(X_train, y_train.ravel())
    y_predict = clf.predict(X_test)
    score.append(accuracy_score(y_test, y_predict))

idx = score.index(max(score))
print('C={}, Gamma={}, Accuracy={}'.format(C[idx], gamma[idx], score[idx]*100))
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_trisurf(C, gamma, score, cmap='viridis')
ax.set_xlabel('C')
ax.set_ylabel('Degree of Polynomial')
ax.set_zlabel('Accuracy')

fig = plt.figure()
ax = plt.axes(projection='3d')
dx = np.ones(len(C))
dy = np.ones(len(C))
dz = np.ones(len(C))
score, dz = dz, score
ax.bar3d(C, gamma, score, dx, dy, dz, shade=True)
ax.set_xlabel('C')
ax.set_ylabel('Gamma')
ax.set_zlabel('Accuracy')
plt.show()
```

Plots ->

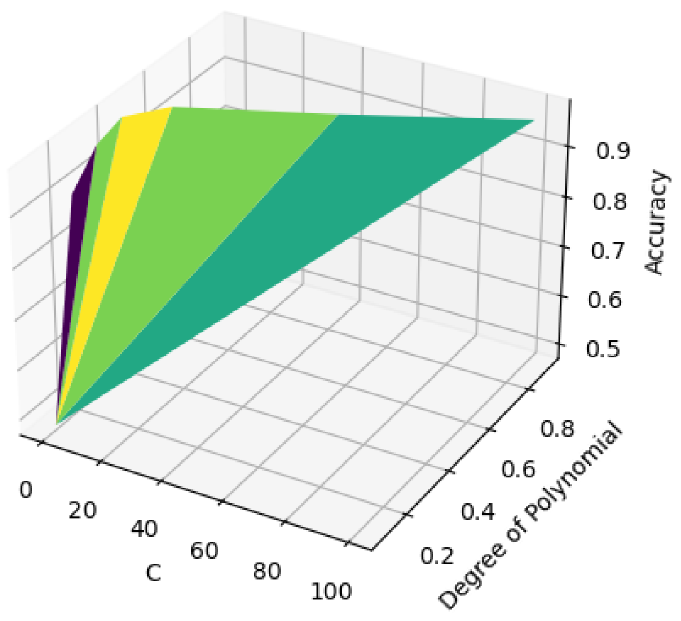


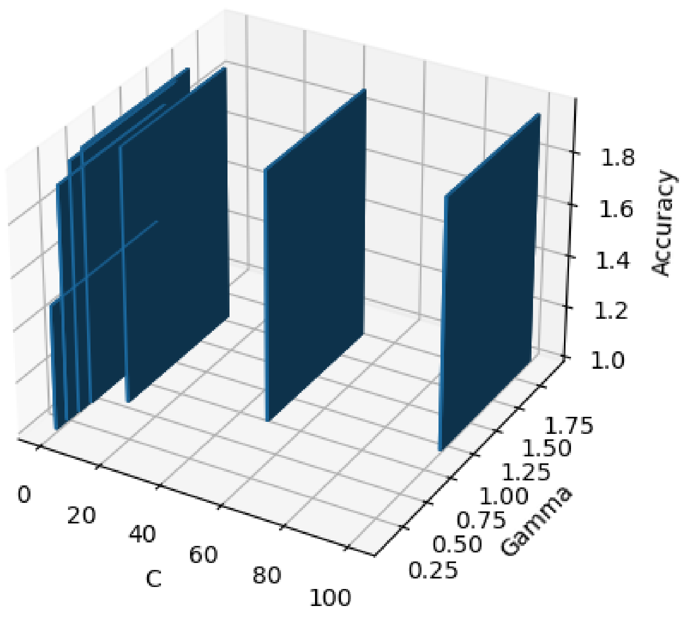
For Linear kernel, the best accuracy 98% is obtained for C = 0.5. It is evident that C hard minimizes the Linear kernel in Support Vector Machines.





For Polynomial kernel, the best accuracy 96% is obtained for C = 0.009, degree of polynomial = 2 and γ = 0.9. As illustrated from the plots, the value for degree of polynomial greatly affects the Polynomial kernel, with minor values of C tending to increase the accuracy and no effect of γ.





For RBF kernel, the best accuracy 98% is obtained for γ = 0.4 and C = 1. The RBF kernel is robust and depends greatly on γ. The values of C tend to be relatively large as compared to Linear and Polynomial kernels, for better accuracy.

Note that multiple solutions exist for each of the kernels but we seek to find the best estimator for lowest complexity. In case of Polynomial kernels, the lower the degree, the lower the complexity.

Question 3:

Let $r = \frac{(y+x)}{2}$, so -

$$k(x,y) = \left(\frac{\pi}{2}\right)^{-d/2} e^{(-x^Tx - y^Ty)} \int_z e^{(-2z^Tz + 4r^Tz)} dz$$

$$= \left(\frac{\pi}{2}\right)^{-d/2} e^{(-x^Tx - y^Ty)} e^{(2r^Tr)} \int_z e^{(-2z^Tz + 4r^Tz - 2r^Tr)} dz$$

$$= \left(\frac{\pi}{2}\right)^{-d/2} e^{(-x^Tx - y^Ty)} e^{(2r^Tr)} \int_z e^{(-2\|z-r\|^2)} dz$$

$$= \left(\frac{\pi}{2}\right)^{-d/2} e^{(-x^Tx - y^Ty)} e^{(2r^Tr)} \left(\frac{\pi}{2}\right)^{d/2}$$

$$= e^{(-x^Tx - y^Ty)} e^{(\frac{1}{2}x^Tx + \frac{1}{2}y^Ty + x^Ty)}$$

$$= e^{(-\frac{1}{2}x^Tx - \frac{1}{2}y^Ty + x^Ty)}$$

$$k(x,y) = e^{-\frac{1}{2}\|x-y\|^2}$$

3) $f(x, \alpha) = \alpha^{-2} x e^{-x/\alpha}$, $x > 0$, $\alpha > 0$

$x_1 = 0.25$, $x_2 = 0.75$, $x_3 = 1.5$, $x_4 = 2.5$, $x_5 = 20$

$$L(\alpha) = \prod_{i=1}^{n} f(x_i ; \alpha) = \prod_{i=1}^{n} \alpha^{-2} x_i e^{-x_i/\alpha}$$

$$L(\alpha) = \alpha^{-2n} \left(\prod_{i=1}^{n} x_i \right) e^{-\sum_{i=1}^{n} x_i / \alpha}$$

$$\log L(\alpha) = -2n \log \alpha + \sum_{i=1}^{n} \log x_i - \sum_{i=1}^{n} x_i / \alpha$$

$$\frac{d}{d\alpha} \log L(\alpha) = \frac{-2n}{\alpha} + \frac{\sum_{i=1}^{n} x_i}{\alpha^2} = 0$$

$$\boxed{\hat{\alpha} = \frac{\sum_{i=1}^{n} x_i}{2n} = \frac{\bar{x}}{2}}$$

Plugging the values in above -

$$\hat{\alpha} = \frac{0.25 + 0.75 + 1.5 + 2.5 + 2.0}{2.5} = 0.70$$

$$\boxed{\hat{\alpha} = 0.70}$$