# COMPSCI 671D: Homework 4

Samuel Eure

Due: February 28, 2019

## 1 Convexity I

---

**1. Prove the sum of two convex functions is also convex.**

Let $x, y \in \mathbb{R}^n$ with $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}$, and $p(x) := f(x)+g(x)$. Suppose $f$ and $g$ are convex functions over the set $[x, y]$ and that $t \in [0, 1]$. By definition, $tg(x)+(1-t)g(y) \geq g(tx+(1-t)y)$ and $tf(x)+(1-t)f(y) \geq f(tx+(1-t)y)$. Thus,

$tg(x) + (1-t)g(y) + tf(x) + (1-t)f(y)$

$= t\big(g(x) + f(x)\big) + (1-t)\big(g(y) + f(y)\big)$

$= tp(x) + (1-t)p(y)$

$\geq f(tx + (1-t)y) + g(tx + (1-t)y) = p(tx + (1-t)y)$

$\Rightarrow tp(x) + (1-t)p(y) \geq p(tx + (1-t)y)$, and thus, $p(x)$ is convex by definition. Thus, the sum of two convex functions is convex.

**2. Let $f(x) = h(g_1(x), g_2(x), .., g_n(x))$. Then for each of the following cases, prove that $f$ is convex:**

---

**2 a) h is convex, $\forall i \in \{1, .., n\}$, $g_i$ is a convex function and $h$ is increasing in its i-th component.**

A function is convex over the set $X$ if it's second derivative is non-negative for all $x \in X$. Thus, referring to the chain rule:

$$\frac{df}{dx} = \frac{df}{dh} \sum_{i=1}^{n} \frac{\partial h}{\partial g_i} \frac{dg_i}{dx}$$

Since $\frac{df}{dh} = 1$,

$$\Rightarrow \frac{df}{dx} = \sum_{i=1}^{n} \frac{\partial h}{\partial g_i} \frac{dg_i}{dx}$$

$$\Rightarrow \frac{d^2 f}{dx^2} = \sum_{i=1}^{n} \frac{d}{dx}\left(\frac{\partial h}{\partial g_i} \frac{dg_i}{dx}\right) = \sum_{i=1}^{n} \frac{\partial h}{\partial g_i} \frac{d^2 g_i}{dx^2} + \left(\frac{d}{dx} \frac{\partial h}{\partial g_i}\right) \frac{dg_i}{dx}$$

Since $\frac{\partial h}{\partial g_i}$ does not depend on $x$ $\forall i$, $\left(\frac{d}{dx} \frac{\partial h}{\partial g_i}\right) = 0$, and since $h$ is increasing $\forall$ index $i$, $\frac{\partial h}{\partial g_i} > 0$. Thus,

$$\Rightarrow \frac{d^2 f}{dx^2} = \sum_{i=1}^{n} \frac{\partial h}{\partial g_i} \frac{d^2 g_i}{dx^2}$$

Since each $g_i$ is convex, $\frac{d^2 g_i}{dx^2} \geq 0$, $\frac{d^2 f}{dx^2} \geq 0$ since it is the sum of non-negative terms. Thus, $\frac{d^2 f}{dx^2}$ is convex.

---

**2 b) $h$ is convex, $\forall i \in \{1, .., n\}, g_i$ are affine functions..**

Since $\frac{\partial h}{\partial g_i}$ does not depend on $x$ $\forall i, \frac{d}{dx} \frac{\partial h}{\partial g_i} = 0$. Since $g_i$ is affine $\forall i$, $\frac{dg_i}{dx}$ is equal to a constant and thus $\frac{d^2 g_i}{dx^2} = 0$. Thus, as was the case before in part $2a$ of this problem,

$$\frac{d^2 f}{dx^2} = \sum_{i=1}^{n} \frac{\partial h}{\partial g_i} \frac{d^2 g_i}{dx^2} + \left(\frac{d}{dx} \frac{\partial h}{\partial g_i}\right) \frac{dg_i}{dx} = \sum_{i}^{n} 0 + 0 = 0 \geq 0$$

and thus $f$ is convex by definition.

---

**2 c) $h$ is convex, $foralli \in \{1, .., n\} g_i$ is a concave function and h is decreasing in its i-th component.**

From part $2a$, it was show that

$$\frac{d^2 f}{dx^2} = \sum_{i=1}^{n} \frac{\partial h}{\partial g_i} \frac{d^2 g_i}{dx^2}$$

Since $\forall i$, $g_i$ is concave, $\frac{d^2 g_i}{dx^2} \leq 0$. Since $h$ is decreasing in it's $i^{th}$ component for each $i$, $\frac{\partial h}{\partial g_i} < 0$ for all $i$. Thus, $\forall i$, $\frac{\partial h}{\partial g_i} \frac{d^2 g_i}{dx^2} \geq 0$ since a negative number is multiplied by a non-positive number. Thus, $\frac{d^2 f}{dx^2}$ is the sum of many non-negative numbers, which is non-negative. Thus, $\frac{d^2 f}{dx^2} \geq 0$, and is thus convex.

**3. The maximum of a convex function $f$ over the polyhedron $P$ is achieved at one of its vertices.**

Let $P$ be a polyhedron with verticies $\{v_1, \ldots, v_n\}$. By definition[1] of a polyhedron, $P = \{x | x = a_1 v_1 + \ldots + a_n v_n\}$ for some $(\sum_{i=1}^{n} a_i = 1)$ with $1 \geq a_i \geq 0 \ \forall i = 1, \ldots, n$. Let $f : P \to \mathbb{R}$ be a convex function over this set. Since $f$ is convex, $\forall z \in P$, where $z = c_1 v_1 + \ldots + c_n v_1$ for some valid constants $c_1, \ldots, c_n$, $f(z) = f(c_1 v_1 + \ldots + c_n v_1)$. Since $f$ is convex,

$$f(z) = f(c_1 v_1 + \ldots + c_n v_1) \leq \sum_{i=1}^{n} c_i f(v_i)$$

and since $\sum_{i=1}^{n} c_i = 1$,

$$f(z) = \sum_{i=1}^{n} c_i f(v_i) \leq max_{i=1:n}\big(f(v_i)\big)$$

Thus, $f$ is bounded above at one of the vertices of $P$, and thus its maximum occurs at one of the verticies of $P$ $\square$.

[1] reference: https://en.wikipedia.org/wiki/Polyhedron

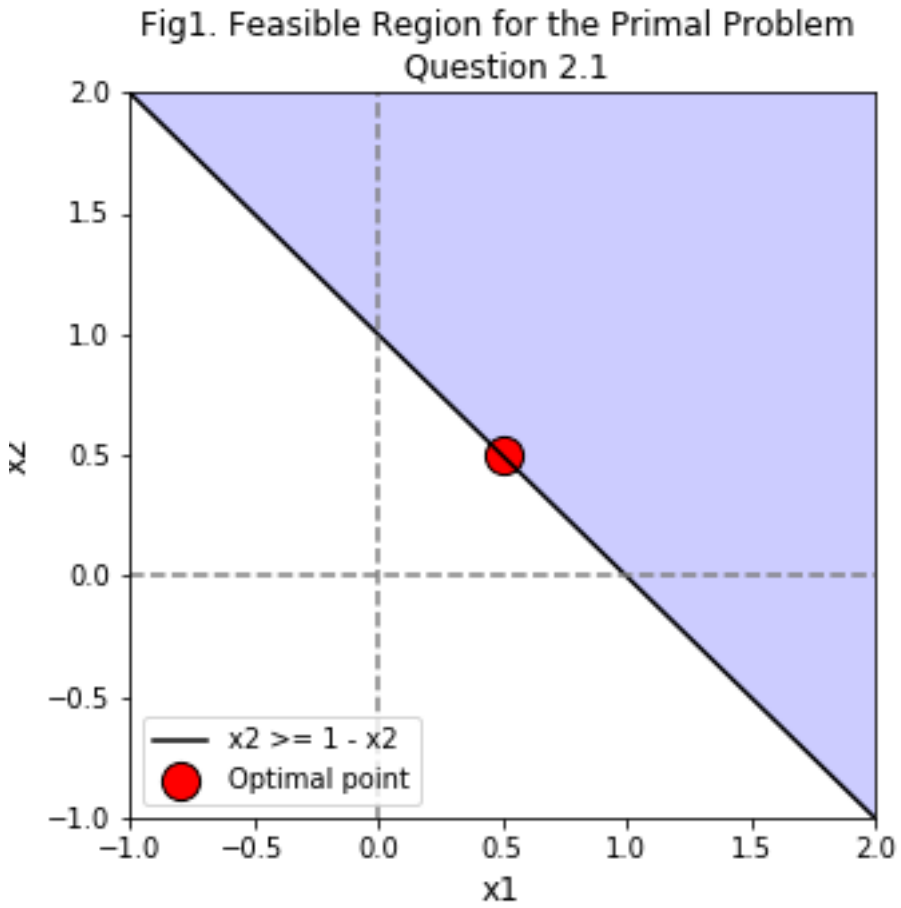
## 2 Convexity II

**Part 1**

Since $g(x_1, x_2) := 1 - x_1 - x_2$ and the constraints to the constraint of the problem is $g(x_1, x_2) \leq 0$, the feasible region is comprised of all $x_1, x_2$ such that $x_2 \geq 1 - x_1$. Thus, the feasible region is shown the region shaded in blue below (and of course the black line). As for $(x_1^*, x_2^*)$, since $f(x_1, x_2) = x_1^2 + x_2^2$, $\bigtriangledown_x f(x) = [2x_2, 2x_1]$. Thus, the minimum occurs when $x_1 = x_2 = 0$. Since this point does not satisfy the constraint $x_2 \geq 1 - x_1$, the most optimal point will fall on the line $x_2 \geq 1 - x_1$ above it. Modifying $f$ such that $f(x) \to \hat{f}(x_1) = x_1^2 + (1 - x_1)^2$ only takes values along the constraint line, the optimal value of $x_1$ can be found by setting the derivative to zero:

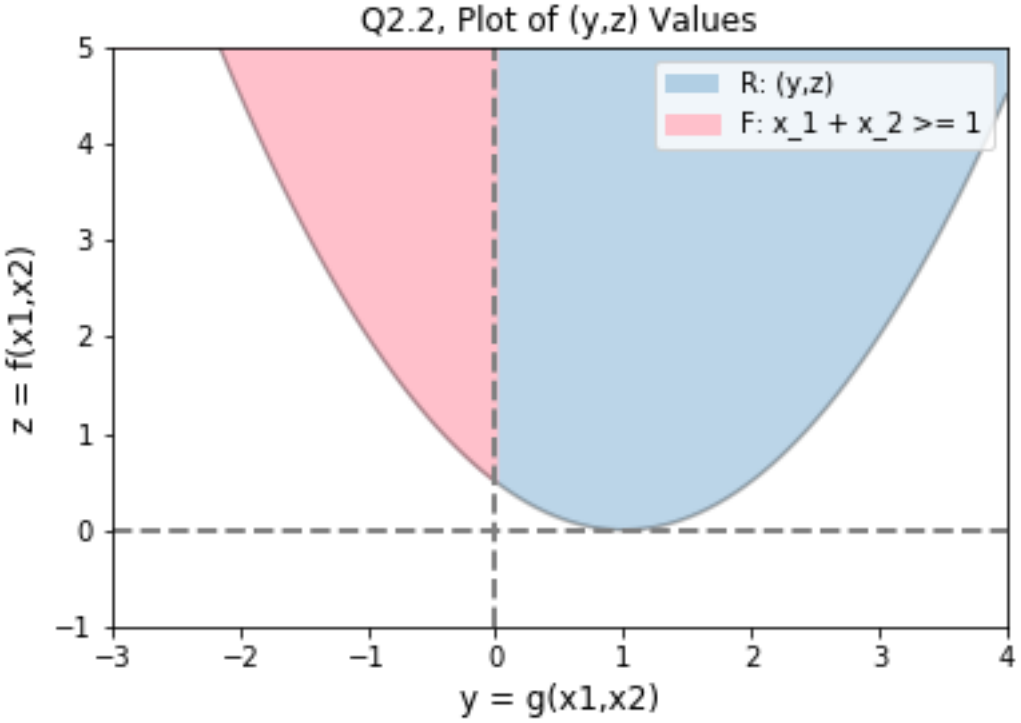
$$\left(\frac{d\hat{f}}{dx_1} = 2x_1 - 2(1 - x_1) = 0\right) \Rightarrow \left(x_1 = 1 - x_1 \Rightarrow x_1 = 0.5\right)$$

Due to the symmetry of the problem, the optimal value of $x_2 = 0.5$. Since $(0.5, 0.5) \in \{(x_1, x_2) | g(x_1, x_2) \leq 0\}$, the point which minimized $f$ and is in the feasible region is $(0.5, 0.5)$.



**Part 2**

Since a point $\vec{w} = (x_1, x_2) \in \mathbb{R}^2$ is feasible if $g(x_1, x_2) = 1 - x_1 - x_2 = 1 - (x_1 + x_2) = 1 - ||\vec{w}||_1 \leq 0$, $F = \{(y, z)|y = g(\vec{w}), z = f(\vec{w}), ||\vec{w}||_1 \geq 1\}$. $R$ can be solved for analytically. First, note that $(x_1 - x_2)^2 \geq 0$. Thus, $x_1^2 + x_2^2 - 2x_1x_2 \geq 0$ and thus $2x_1^2 + 2x_2^2 \geq x_1^2 + x_2^2 + 2x_1x_2 = (x_1 + x_2)^2 = (1 - y)^2$ and thus $z \geq \frac{1}{2}(1 - y)^2$.



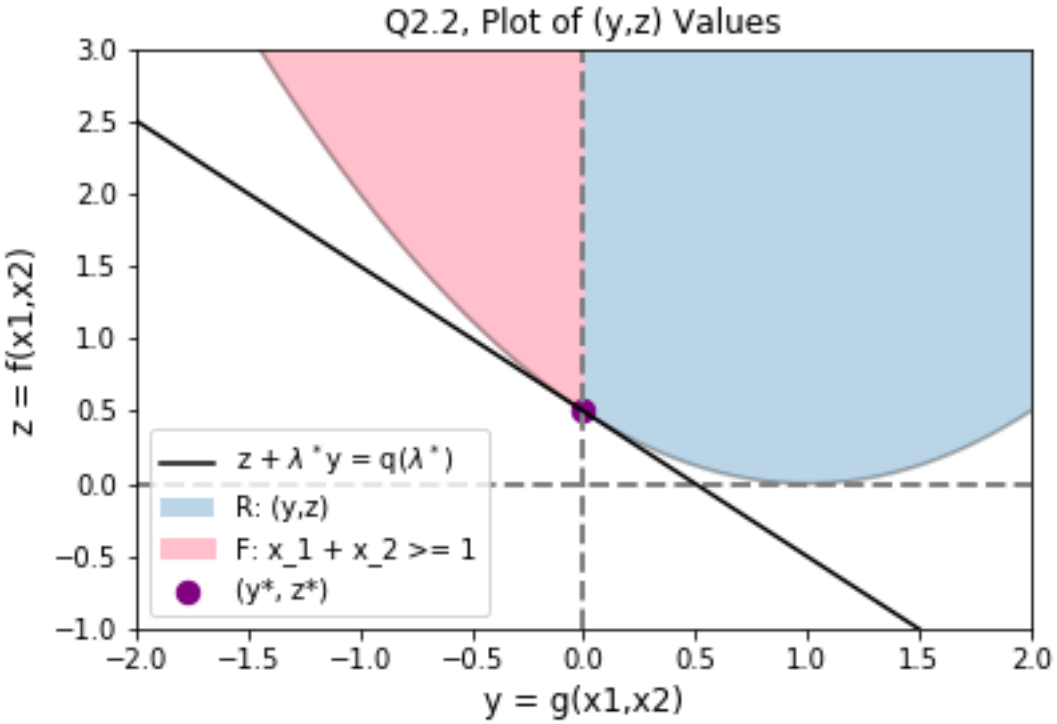
**Part 3**

Here, I will solve the dual problem by calculating $max_{\lambda \in \mathbb{R}} \Big( min_{x \in \mathbb{R}^2} \big( f(x) + \lambda g(x) \big) \Big)$. First I'll determine $q(\lambda)$:

$$\bigtriangledown_x \Big( f(x) + \lambda g(x) \Big) = \bigtriangledown_x \Big( x_1^2 + x_2^2 + \lambda - \lambda x_1 - \lambda x_2 \Big) = [2x_1 - \lambda, 2x_2 - \lambda]$$

$$\Big( [2x_1 - \lambda, 2x_2 - \lambda] = 0 \Big) \Longrightarrow \Big( x_1 = x_2 = \frac{\lambda}{2} \Big)$$

Thus, a critical point occurs at $x_1 = x_2 = \lambda/2$. Since the the second derivative for each of these coordinates is $[2, 2]$ which are both positive, this point is a minimum. Thus, $q(\lambda) = \frac{\lambda^2}{2^2} + \frac{\lambda^2}{2^2} + \lambda(1 - \frac{\lambda}{2} - \frac{\lambda}{2}) = \frac{\lambda^2}{2} + \lambda - \lambda^2 = -\frac{\lambda^2}{2} + \lambda$. To maximize this function I set it's derivative equal to zero: $\frac{d}{d\lambda} q(\lambda) = -\lambda + 1 =$. Thus, the critical point occurs at $\lambda = 1$. Since $\frac{d^2 q(\lambda)}{d\lambda^2} = -1$, this point must be a maximum. Thus, $\lambda^* = 1$, and thus $x^* = [x_1^*, x_2^*] = \lambda^*/2 = [0.5, 0.5]$. Thus, $y^* = g(x^*) = 1 - 0.5 - 0.5 = 0$ and $z^* = f(x^*) = 2(0.5^2) = 2/4 = 1/2$. As for the line $z + \lambda^* y = q(\lambda^*)$, since $\lambda^* = 1 \Rightarrow q(\lambda^*) = 0.5$. Thus, the equation I must plot is $z + y = 0.5$ or $z = 0.5 - y$.



**Part 4**

Show that $q(\lambda)$ is a concave function on $S_q$ where $S_q = \{\lambda | q(\lambda) \in \mathbb{R}\}$

Since $q(\lambda) = min_x(f(x) + \lambda g(x)) = f([\lambda/2, \lambda/2]) + \lambda(1 - \lambda/2 - \lambda/2) = \frac{\lambda^2}{2} + \lambda - \lambda^2 = -\frac{\lambda^2}{2} + \lambda$, we can prove this directly by observing the second derivative of $q(\lambda)$:

$$\frac{d^2}{d\lambda^2} q(\lambda) = \frac{d}{d\lambda}\Big( - \lambda + 1\Big) = -1 \leq 0$$

Thus, $\forall \lambda \in \mathbb{R}$, $\frac{d^2}{d\lambda^2} q(\lambda) \leq 0$, and thus $q(\lambda)$ is a concave function over $S_q$.

# 3 Support Vector Machine

### 3.1

The constraints for this problem are, $\forall i$, $\epsilon_i \geq 0$ and $y_i(w^T x_i + b) \geq (1 - \epsilon_i)$. This can equivalently be written as $0 \geq -\epsilon_i$ and $0 \geq (1 - \epsilon_i) - y_i(w^T x_i + b)$. Thus, the Lagrangian Dual Problem is presented as

$$L(w, b, \epsilon_{i:n}, \alpha_{1:n}, r_{i:n}) = \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{n}\epsilon_i + \sum_{i=1}^{n}\alpha_i\Big((1-\epsilon_i) - y_i(w^T x_i + b)\Big) + \sum_{i=1}^{n} r_i(-\epsilon_i) \tag{1}$$

I will now check the KKT conditions for Lagrangian stationary where $x_{1:n}$ represents a vector:

$$\Big(\bigtriangledown_w L(w, b, \epsilon_{i:n}, \alpha_{1:n}, r_{i:n}) = w^T - \sum_{i=1}^{n}\alpha_i y_i x_i^T = 0_{1:n}\Big) \Longrightarrow \Big(w = \sum_{i=1}^{n}\alpha_i y_i x_i\Big) \tag{2}$$

$$\frac{d}{db}L(w, b, \epsilon_{i:n}, \alpha_{1:n}, r_{i:n}) = \Big(\sum_{i=1}^{n}\alpha_i y_i = 0\Big) \Longrightarrow \Big(\sum_{i=1}^{n}\alpha_i y_i = 0\Big) \tag{3}$$

$$\Big(\bigtriangledown_{\epsilon_{1:n}} L(w, b, \epsilon_{i:n}, \alpha_{1:n}, r_{i:n}) = C1_{1:n}^T - \alpha_{1:n}^T - r_{1:n}^T = 0_{1:n}\Big) \Longrightarrow \Big(r_{1:n} = C1_{1:n} - \alpha_{1:n}\Big) \tag{4}$$

where $1_{1:n}$ is a vector of ones.

Thus, conditions (2), (3), and (4) must be met. Next, I will rewrite the Lagrangian above:

$$L(w, b, \epsilon_{i:n}, \alpha_{1:n}, r_{i:n}) = \frac{1}{2}\sum_{j=1}^{n} w_j^2 + C\sum_{i=1}^{n}\epsilon_i + \sum_{i=1}^{n}\alpha_i\Big((1-\epsilon_i) - y_i\big(\big(\sum_{j=1}^{n} w_j x_{i,j}\big) + b\big)\Big) + \sum_{i=1}^{n}(C - \alpha_i)(-\epsilon_i)$$

$$= \frac{1}{2}\sum_{j=1}^{n}\Big(\sum_{i=1}^{n}\alpha_i y_i x_{i,j}\Big)\Big)^2 + C\sum_{i=1}^{n}\epsilon_i - C\sum_{i=1}^{n}\epsilon_i + \sum_{i=1}^{n}\alpha_i\epsilon_i + \sum_{i=1}^{n}\alpha_i(1-\epsilon_i) - \sum_{i=1}^{n}\alpha_i y_i\Big(\big(\sum_{j=1}\sum_{k=1}\alpha_k y_k x_{k,j} x_{i,j}\big) + b\Big)$$

$$= \frac{1}{2}\sum_{i=1}\sum_{k=1}\alpha_i\alpha_k y_i y_k x_i^T x_k + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}\sum_{k=1}\alpha_i\alpha_k y_i y_k x_i^T x_k + b\sum_{i=1}^{n}\alpha_i y_i$$

$$= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}\sum_{k=1}\alpha_i\alpha_k y_i y_k x_i^T x_k$$

$$\Longrightarrow$$

$$L(\alpha_{1:n}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}\sum_{k=1}\alpha_i\alpha_k y_i y_k x_i^T x_k \tag{5}$$

Thus, the Lagrangian Dual Problem of the SVM primal problem can be given by a quadratic program with Lagrange multipliers $\alpha_i$, $\forall i \in \{1, ..., n\}$

**3.2) Show that for the primal and dual optimal solutions, the Lagrange multiplier $\alpha_i$ is non-zeros for samples where $y_i(W^T x_i + b) \leq 1$.**

First, I must note that as shown in equation (5) above, the optimal values of $\alpha$ are very data dependent, and thus the equality of this problem is not necessarily true. For example, suppose that included in the data set are two identical pairs of x and y values (i.e. $\exists\{x_a, y_a\}, \{x_b, y_b\}$ where $a \neq b$ however $x_a = x_b$ and $y_a = y_b$). In this situation, if $\alpha_a + \alpha_b$ need to equal come constant $c$, then one could merely make $\alpha_a = c$ and $\alpha_b = 0$. However, I will still prove the proposition for $\{x_i, y_i\}$ where $y_i(W^T x_i + b) < 1$.

First, suppose $y_i(W^T x_i + b) < 1$ and let $\epsilon_i = 1 - y_i(W^T x_i + b) > 0$. Next, note that the value of $C > 0$ in our primal problem. Since we would like to accrue a penalty whenever $y_i(W^T x_i + b) < 1$, as shown above, an $r_i(-\epsilon)$ term included in the Lagrangian Dual Problem, as shown in equation (1) above. As shown in equation (4) above, the optimal value of $r_i = C - \alpha_i$. At the primal and dual optimal solutions, complementary slackness is achieved, thus, at the optimal solution, $r_i\epsilon_i = (C - \alpha_i)\epsilon_i = 0$. Since $\epsilon_i > 0$ at this point, it must be that $C = \alpha_i > 0$, $\square$.

# Question3

February 27, 2019

## 1 Question 3 Code

```
In [1]: import numpy as np
        import pandas as pd
        import sklearn as sk
        from tqdm import tqdm
        import matplotlib.pyplot as plt
        from sklearn import model_selection
        from sklearn.svm import SVC
```

### 1.1 Q3.3

```
In [68]: #Importing the data
         originalData  = pd.read_csv("messidor.csv")
         featureNames = originalData.columns.tolist()
         data = np.array(originalData)
         #Splitting the data
         a,b,c,d = sk.model_selection.train_test_split(
             data[:,:-1],data[:,-1],test_size = 0.4)
         x_train, x_test, y_train, y_test = a,b,c,d
         x_train  = sk.preprocessing.normalize(x_train)
         x_test  = sk.preprocessing.normalize(x_test)
         print("Units of training data:",len(x_train)/(len(x_test)/2))
         print("Units of testing  data:",len(x_test)/(len(x_test)/2))
```

```
Units of training data: 2.9934924078091107
Units of testing  data: 2.0
```

### 1.2 Q3.4

```
In [77]: def getAcc(pred, y):
             correct = np.mean((np.array(pred) == np.array(y)))
             return(correct)

         #Fitting the model
         svm = sk.svm.SVC(C = 10e-10, kernel = 'linear')
         linearModel = svm.fit(x_train, y_train)
```

```
        #Predicting
        trainPred = linearModel.predict(x_train)
        testPred  = linearModel.predict(x_test)
        testA  = getAcc(testPred, y_test)
        trainA = getAcc(trainPred, y_train)
        coefs = pd.DataFrame()
        coefs['Features'] = featureNames[:-1]
        coefs["Weights"] = linearModel.coef_[0]

        #Results
        print("training results:", "(Acc =",trainA,")")
        print("testing  results:", "(Acc = ",testA,")\n")
        coefs
```

```
training results: (Acc = 0.5463768115942029 )
testing  results: (Acc =  0.5075921908893709 )
```

Out[77]:

| | Features | Weights |
|---|---|---|
| 0 | quality_assessment | -6.362927e-10 |
| 1 | pre-screening_results | -7.602453e-10 |
| 2 | MA_1 | 1.199317e-08 |
| 3 | MA_2 | 7.724687e-09 |
| 4 | MA_3 | 4.226712e-09 |
| 5 | MA_4 | 4.449798e-10 |
| 6 | MA_5 | -1.824233e-09 |
| 7 | MA_6 | -3.225786e-09 |
| 8 | exudate_1 | -4.036268e-09 |
| 9 | exudate_2 | -5.016140e-09 |
| 10 | exudate_3 | -8.428159e-11 |
| 11 | exudate_4 | 1.559994e-09 |
| 12 | exudate_5 | 1.526612e-09 |
| 13 | exudate_6 | 7.433253e-10 |
| 14 | exudate_7 | 3.311588e-10 |
| 15 | exudate_8 | 1.311546e-10 |
| 16 | distance_macula_opticdisk | -3.719758e-10 |
| 17 | diameter_opticdisk | -8.661792e-11 |
| 18 | am_fm | -4.301586e-10 |

### 1.3 Q3.5

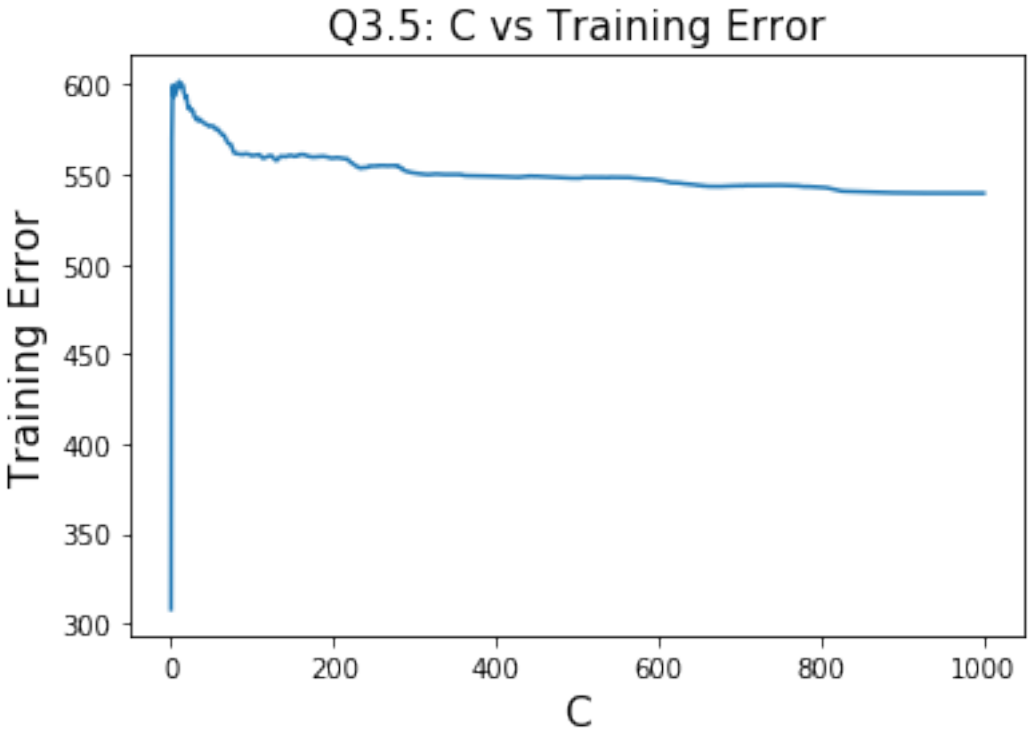
In [58]:
```
def getError(pred, y):
    margins = y*pred
    error = np.ones(len(margins))- margins
    error = error*(error > 0)
    return(np.sum(error))
```

```
CValues = np.linspace(10e-10,1000,1000)

errorSum = []

def fitLinear(c, x_train, y_train):
    svm = sk.svm.SVC(C = c, kernel = 'linear')
    linearModel = svm.fit(x_train, y_train)
    return(linearModel)

for c in tqdm(CValues):
    modelC = fitLinear(c, x_train, y_train)
    trainPred = modelC.decision_function(x_train)
    errorSum.append(getError(trainPred, y_train))
```

```
100%|| 1000/1000 [00:32<00:00, 30.74it/s]
```

In [59]:
```
plt.plot(CValues, errorSum)
plt.title("Q3.5: C vs Training Error", fontsize = 15)
plt.xlabel("C", fontsize = 15)
plt.ylabel("Training Error", fontsize = 15)
```

Out[59]:
```
Text(0,0.5,'Training Error')
```



```
In [64]: print("Min Error of", errorSum[np.argmin(errorSum)],
                "occured at C=", CValues[np.argmin(errorSum)])
```

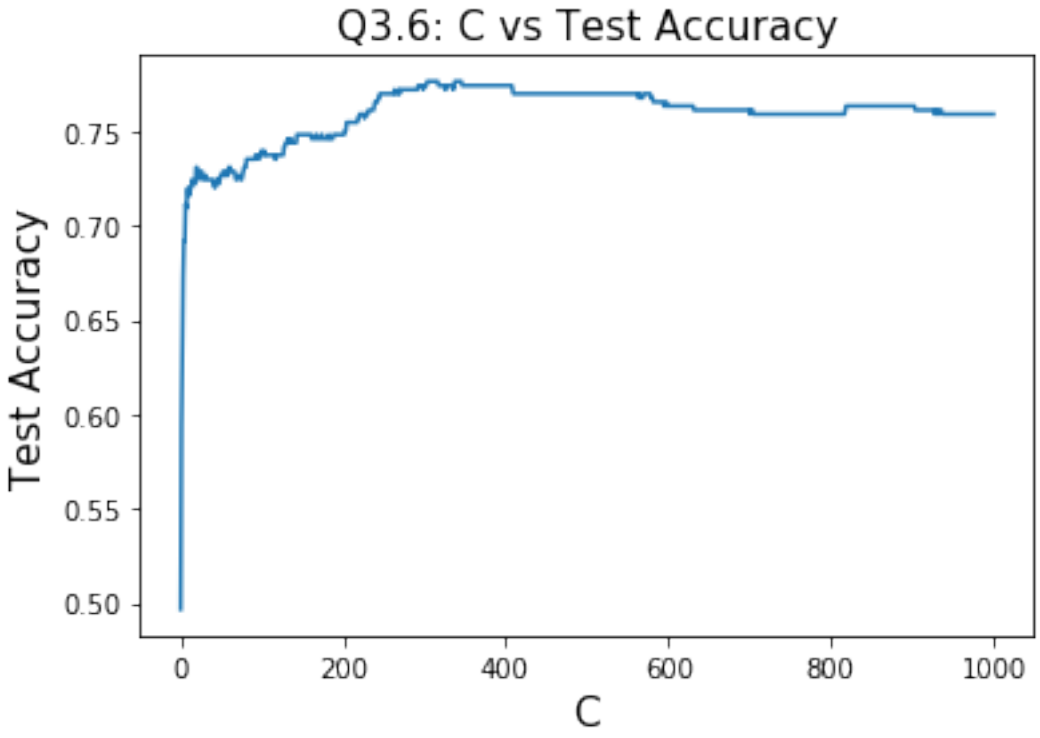
```
Min Error of 308.0000008249135 occured at C= 1e-09
```

**I observe that the training error seems to to start out quite small (around 308) for C values near zero, then it seems to shoot up, quite quickly, and then it gradually decreases, apparently asymptotically approaching around 540. This plot suggests that the best value of C is near zero.**

### 1.4 Q3.6

```
In [63]: accs = []
         for c in tqdm(CValues):
             modelC = fitLinear(c, x_train, y_train)
             testPred = modelC.predict(x_test)
             accs.append(getAcc(testPred, y_test))

         plt.plot(CValues, accs)
         plt.title("Q3.6: C vs Test Accuracy", fontsize = 15)
         plt.xlabel("C", fontsize = 15)
         plt.ylabel("Test Accuracy", fontsize = 15)
```

```
Out[63]: Text(0,0.5,'Test Accuracy')
```



```
In [65]: print("Max Accuracy of",accs[np.argmax(accs)],
                "occured at C=", CValues[np.argmax(accs)])
```

```
Max Accuracy of 0.7765726681127982 occured at C= 303.303303304
```

**It appears that the accuracy is near zero for a C value near zero, rapidlly increases until about C = 5, then steadily increases until C = 303, and then gradually decreases but esentially levels off just above 0.75 or greater. This would suggest that the optimal value of C is close to 303. Since we are more concerned with the test accuracy than the actual training error, this plot is more informative for our optimal value of C, and should thus "overpower" the information that the C vs Training Error plot provides. Thus, I would suggest the optimal value of C is around 303.**

## 1.5 Q3.7

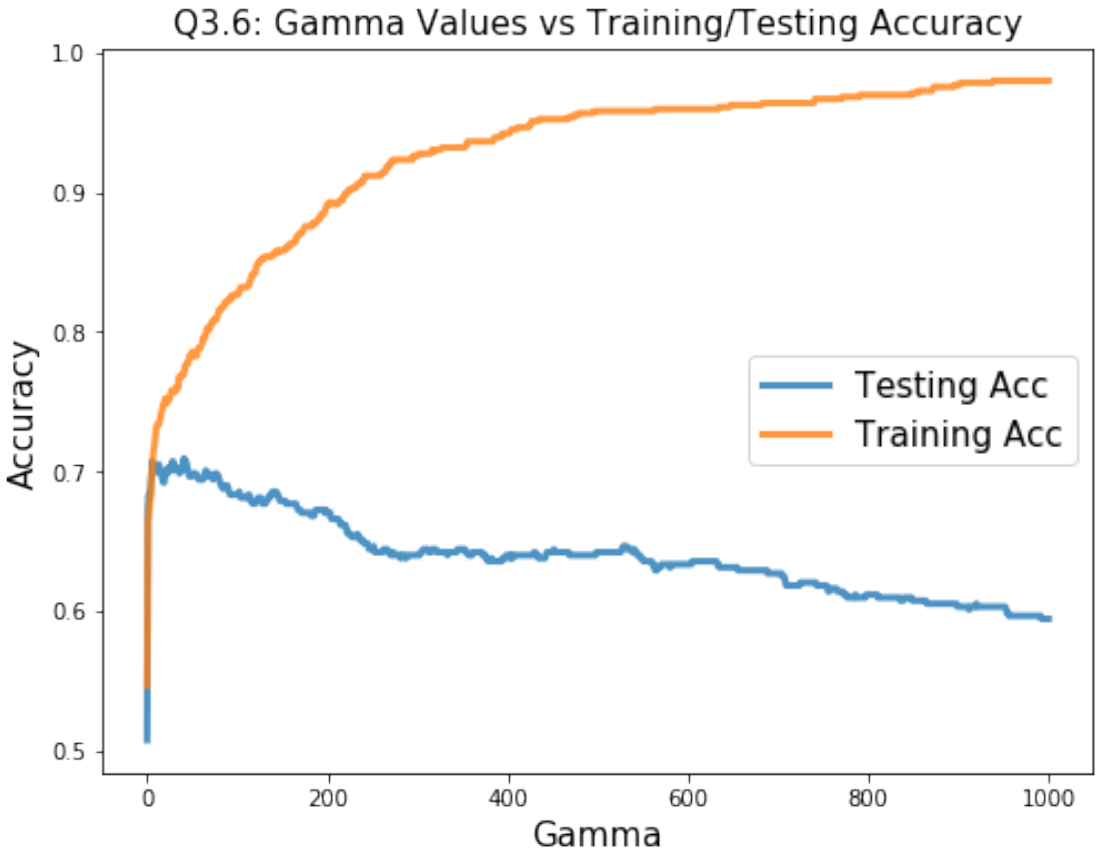
### 1.5.1 I will choose to due Q3.7b, where I hold C constant at 303 and change the gamma value for 'rbf'

```
In [106]: def fitRBF(g, x_train, y_train):
              svm = sk.svm.SVC(gamma = g, kernel = 'rbf')
              model = svm.fit(x_train, y_train)
              return(model)

          GAMMAS = np.linspace(10e-10,1000,1000)
          trainAcc = []
          testAcc  = []
          for g in tqdm(GAMMAS):
              modelG = fitRBF(g, x_train, y_train)
              testPred = modelG.predict(x_test)
              testAcc.append(getAcc(testPred, y_test))
              trainPred = modelG.predict(x_train)
              trainAcc.append(getAcc(trainPred, y_train))
```

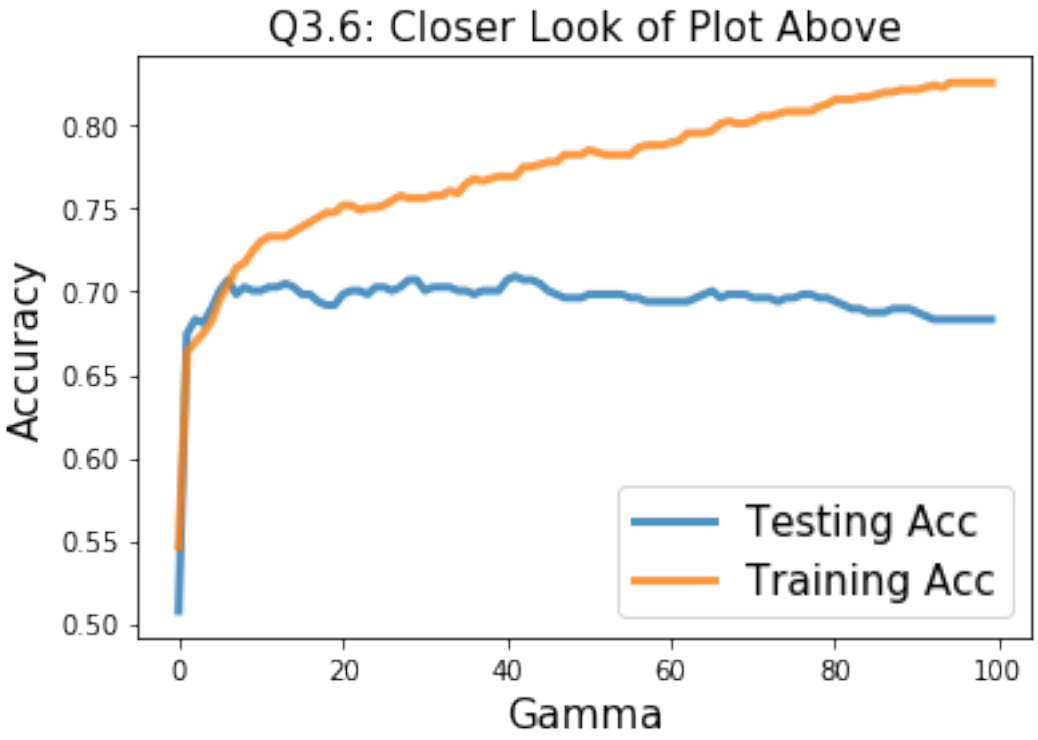
```
100%|| 1000/1000 [00:58<00:00, 17.20it/s]
```

```
In [110]: fig = plt.figure(figsize = (8,6))
          plt.plot(GAMMAS, testAcc, label = 'Testing Acc',
                   linewidth = 3, alpha = 0.8)
          plt.plot(GAMMAS, trainAcc, label = 'Training Acc',
                   linewidth = 3, alpha = 0.8)
          plt.title("Q3.6: Gamma Values vs Training/Testing Accuracy",
                    fontsize = 15)
          plt.xlabel("Gamma", fontsize = 15)
          plt.ylabel("Accuracy", fontsize = 15)
          plt.legend(fontsize = 15, loc = "center right")
```

```
Out[110]: <matplotlib.legend.Legend at 0x1a1a58a6a0>
```




```
In [111]: plt.plot(GAMMAS[:100], testAcc[:100],
                    label = 'Testing Acc',linewidth = 3, alpha = 0.8)
          plt.plot(GAMMAS[:100], trainAcc[:100],
                    label = 'Training Acc',linewidth = 3, alpha = 0.8)
          plt.title("Q3.6: Closer Look of Plot Above", fontsize = 15)
          plt.xlabel("Gamma", fontsize = 15)
          plt.ylabel("Accuracy", fontsize = 15)
          plt.legend(fontsize = 15, loc = "lower right")

Out[111]: <matplotlib.legend.Legend at 0x1a1a605f60>
```




```
In [112]: print("Max Testing Accuracy of", testAcc[np.argmax(testAcc)],
                "occured at gamma =", GAMMAS[np.argmax(testAcc)])
```

```
Max Testing Accuracy of 0.7093275488069414 occured at gamma = 41.041041042
```

**My experiment yeilded a gamma value of around 41 to give a maximum training accuracy, however all gamma values between 10 and about 60 give very similar testing accuracies. The first plot above shows that training accuracy steadily increases to one as gamma increases, however my testing accuracy reaches it's peak at around 10 to 60 and it there could no real difference between any of the gamma values within this interval in terms of testing accuracy.**

# 4 Kernels

**Question 1**

First, let $x \in \chi$, and define the feature map $\Phi : \chi \to \mathbb{R}^{\chi}$ such that $\Phi(x) := f_x(\cdot)$ where for $z \in \chi$, $f_x(z) := x^T z$. Define $H := span\{\Phi(x) | x \in \chi\}$. Thus, $\forall f(\cdot) \in H$, $f(\cdot) = \sum_{x \in \chi} \alpha_x \Phi(x)$. Next, for $f, g \in H$ where $f(\cdot) = \sum_{x \in \chi} \alpha_x \Phi(x)$ and $g(\cdot) = \sum_{y \in \chi} \beta_y \Phi(y)$ , define the inner product $\langle f, g \rangle_{\mathbb{R}^{\chi}} = \sum_{y \in \chi} \sum_{x \in \chi} \alpha_x \beta_y x^T y$. Since a matrix dot product is an inner product, this inner product is a valid inner product. Lastly, for $x, y \in \chi$, define the kernel $k(x, y) := x^T y$. By construction of $H$, $\Phi(x)$ and $\Phi(y)$ are part of the definitive basis of $H$, and thus $\langle \Phi(x), \Phi(y) \rangle_{\mathbb{R}^{\chi}} = x^T y = k(x, y)$. Tellingly, this implies that $\forall f \in H$ can be re-written as $f(\cdot) = \sum_{x \in \chi} \alpha_x k(\cdot, x)$ and the inner product can be re-written as $\langle f, g \rangle_{\mathbb{R}^{\chi}} = \sum_{y \in \chi} \sum_{x \in \chi} \alpha_x \beta_y k(x, y)$. Thus $\forall f \in H$ and $z \in \chi$, $\langle f, k(\cdot, x) \rangle_{\mathbb{R}^{\chi}} = \sum_{x \in \chi} \alpha_x k(x, z) = f(z)$. Thus, $H$ is a RKHS.

Let $\chi_s$ represent the subset of $\chi$ which comprises our dataset. From the Representer Theorem, I know that for any loss function used in this problem (and thus, for this loss function), the optimal solution of $\theta$ will be within the span of $\{x | x \in \chi_s\}$, and thus, $\theta = \sum_{x \in \chi_s} \nu_x x$ where $\nu_x \in \mathbb{R}$ $\forall x \in \chi_s$ since the optimal classifier $f^* = \sum_{x \in \chi_s} \nu_x k(\cdot, x)$

**Question 2**

Let $k_i$ have feature map $\Phi_i$ and inner product $\langle \cdot, \cdot \rangle_{H_i}$

---

**a)** $k(x,z) = \alpha k_1(x,z) + \beta k_2(x,z)$ **for** $\alpha, \beta \geq 0$

Proof: From Mercer's Theorem, we know that each of the kernels can be represented by inner products. Thus,

$$\alpha k_1(x,z) + \beta k_2(x,z) = \langle \sqrt{\alpha}\Phi_1(x), \sqrt{\alpha}\Phi_1(z) \rangle_{H_1} + \langle \sqrt{\beta}\Phi_2(x), \sqrt{\beta}\Phi_2(z) \rangle_{H_2}$$

Let $\Phi(\cdot) := [\sqrt{\alpha}\Phi_1(\cdot), \sqrt{\beta}\Phi_2(\cdot)]$. Thus,

$$\alpha k_1(x,z) + \beta k_2(x,z) = \langle \Phi(x), \Phi(z) \rangle_H$$

for some Hilbert space $H$. This function must be symmetric since $k_1$ and $k_2$ are symmetric and the Gram matrix is positive semi-definite since the entries are formed from a valid inner product. Thus, $k(x,z) = \alpha k_1(x,z) + \beta k_2(x,z)$ is a valid kernel!

---

**b)** $k(x,z) = k_1(x,z)k_2(x,z)$

Proof: From Mercer's Theorem, we know that each of the kernels can be represented by inner products. Thus,

$$k_1(x,z)k_2(x,z) = \left( \sum_i^n \Phi_1(x)_i \Phi_1(z)_i \right) \left( \sum_j^m \Phi_2(x)_j \Phi_2(z)_j \right) = \sum_{i,j} \left( \Phi_1(x)_i \Phi_2(x)_j \right) \left( \Phi_1(z)_i \Phi_2(z)_j \right)$$

Define $\Phi(\cdot) := [\Phi_1(\cdot)_1 \Phi_2(\cdot)_1, \Phi_1(\cdot)_1 \Phi_2(\cdot)_2, \ldots, \Phi_1(\cdot)_n \Phi_2(\cdot)_m]$. Thus,

$$k(x,z) = k_1(x,z)k_2(x,z) = \Phi(x)^T \Phi(z) = \langle \Phi(x), \Phi(z) \rangle_{H_{new}}$$

This function must be symmetric since $k_1$ and $k_2$ are symmetric and the Gram matrix is positive semi-definite since the entries are formed from a valid inner product. Thus, it is a valid kernel.

---

**c)** $k(x,z) = f(x)f(z)$**,** $f : \chi \to \mathbb{R}$

Proof: Since $f : \chi \to \mathbb{R}$, and scalar multiplication is commutative, clearly $k(x,z) = k(z,x)$, and thus it is symmetric. Moreover, $\forall x \in \chi$, $f(x)f(x) = \big(f(x)\big)^2 \geq 0$ and $\big(f(x)\big)^2 = 0 \iff f(x) = 0$. Lastly, let $\Phi(x) = f(x)$. Then, $k(x,z) = f(x)f(z) = \Phi(x)\Phi(z) = \langle (\Phi(x), \Phi(z) \rangle_{\chi^\Phi}$ and thus, $k$ can be represented as an inner product and it's Gram matrix is thus positive semi-definite.

**d)** $k(x,z) = f(k_1(x,z))$ **for** $f$ **a polynomial with positive coefficients**

Proof: Let $n$ be the degree of the polynomial $f$. Then, $f(k_1(x,z)) = \sum_{i=1}^{n} \alpha_i k_1(x,z)^i$ for positive constants $\alpha_i \in \mathbb{R}$, $\forall i$. From part question 4.2.$b$, we know that $k_1(\cdot,\cdot)k_1(\cdot,\cdot)$ is a valid kernel, and thus $k_1(\cdot,\cdot)^i$ is a valid kernel for all positive integers $i$. From question 4.2.$a$ we know that $\alpha_i k_1(\cdot,\cdot)^i$ is a kernel if $k_1(\cdot,\cdot)^i$ is a kernel if $\alpha_i > 0$. thus, $k(\cdot,\cdot)$ is the sum of various kernels, and is thus a kernel itself.

---

**e)** $k(x,z) = e^{-\alpha||x-z||_2^2}$ **is a kernel with** $\alpha \in \mathbb{R}$**.**

Proof: Let $K(x,z), x, z \in \chi$ be a valid kernel. Since polynomials functions with positive coefficients of kernels are themselves kernels, assuming $\chi$ is compact, this implies $\sum_{i=0}^{\infty} \frac{K(x,z)^i}{i!} = e^{K(x,z)}$ is a valid kernel as well. Since $-\alpha||x-z||_2^2 = -\alpha||x||_2^2 - \alpha||z||_2^2 + 2\alpha x^T z$, this implies

$$e^{-\alpha||x-z||_2^2} = e^{-\alpha||x||_2^2} e^{-\alpha||z||_2^2} e^{2\alpha x^T z}$$

Define function $f : \chi \to \mathbb{R}$ s.t. $f(\cdot) = e^{-\alpha||\cdot||_2^2}$ and kernel $g : \chi x \chi \to \mathbb{R}$ such that $g(a,b) = 2\alpha a^T b$. Thus, from question 4.2.$b, c$, the following two are kernels:

$$k_1(x,z) := e^{-\alpha||x||_2^2} e^{-\alpha||z||_2^2}$$

$$k_2(x,z) := e^{g(x,z)}$$

and thus $k(x,z) = k_1(x,z)k_2(x,z) = e^{-\alpha||x-z||_2^2}$ is a kernel.