COMP7703 – Homework Task 8
Joel Thomas 44793203

1.

After plotting both sets of points from the two classes, it is evident the optimal separating line must cut through diagonally between the point $(1,1)$ belonging to class 1 and the points $(0,1)$ and $(1,0)$ belonging to class 2 (see figure 1 below). The upper margin (class 1) exactly passes $(1,1)$ whereas the lower margin (class 2) passes both $(0,1)$ and $(1,0)$. We can evaluate the gradient for the lower margin as follows:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$m = \frac{1 - 0}{0 - 1} = -1$$

We know the vertical-axis intercept already given by $c = 1$ from the point $(0,1)$ which lies on this line. So, the equation for the **lower margin** is given by:

$$\therefore x_2 = -x_1 + 1$$

Since both margins and the optimal separating line are parallel, they must share the same gradient. The equation for the **upper margin** is given by:

$$x_2 = -x_1 + c$$

Using $(1,1)$ which lies on this line:

$$1 = -1 + c$$

$$\rightarrow c = 2$$

$$\therefore x_2 = -x_2 + 2$$

The optimal separating line passes halfway between both the margins. By the equation for the lower margin, we know that $\left(\frac{1}{2}, \frac{1}{2}\right)$ also lies on this line. Finding the midpoint between the points $\left(\frac{1}{2}, \frac{1}{2}\right)$ and $(1,1)$:

$$m = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

$$m = \left(\frac{\frac{1}{2} + 1}{2}, \frac{\frac{1}{2} + 1}{2}\right) = \left(\frac{3}{4}, \frac{3}{4}\right)$$

Since both margins and the optimal separating line are parallel, they must share the same gradient. The equation for the **optimal separating line** is given by:

$$x_2 = -x_1 + c$$

Using the midpoint $\left(\frac{3}{4}, \frac{3}{4}\right)$ which lies on this line:

$$\frac{3}{4} = -\frac{3}{4} + c$$

$$c = 2 \cdot \frac{3}{4} = \frac{3}{2}$$

$$\therefore x_2 = -x_1 + \frac{3}{2}$$

The **margin** is given by the distance between the optimal separating line and either one of the upper or lower margin lines. We can calculate this by finding the distance between the midpoint $\left(\frac{3}{4}, \frac{3}{4}\right)$ and $(1,1)$:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d = \sqrt{\left(1 - \frac{3}{4}\right)^2 + \left(1 - \frac{3}{4}\right)^2}$$

$$d = \sqrt{2 \cdot \frac{1}{4}}$$

$$d = \sqrt{\frac{1}{2}} = \frac{1}{\sqrt{2}}$$

Finally, based on the setup of the problem, the **support vectors** are $(1,0)$, $(0,1)$ and $(1,1)$. The optimal separating line, margin and support vectors for this problem are summarised in figure 1 below:

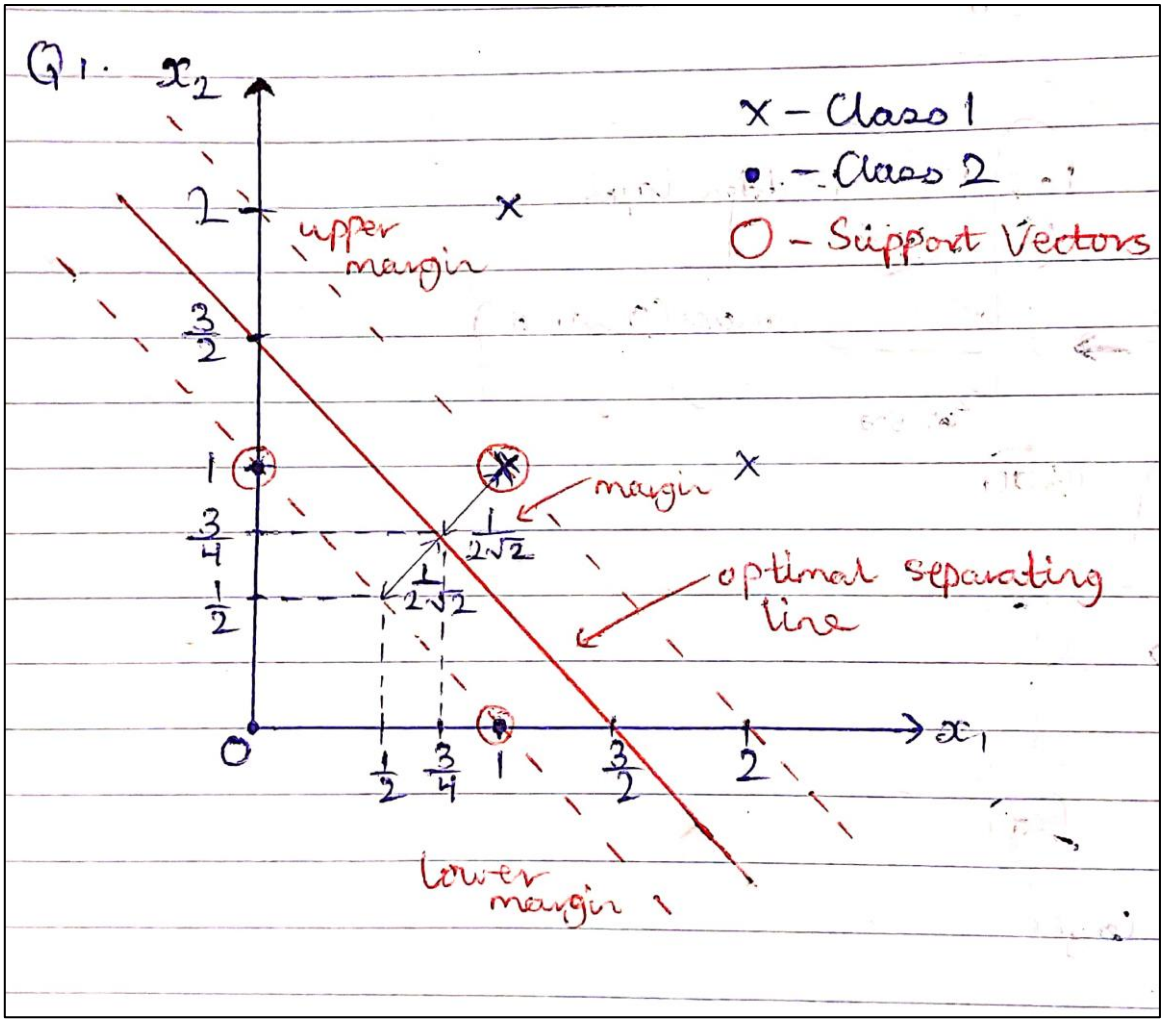

Figure 1: For a two-class problem where the instances of the classes are shown by ·/×, the solid red line is the optimal separating line whereas the dashed red lines define the margins on either side. Circled instances are the support vectors

2.

Use the formula from Alpaydin, Chapter 13.2 page 351:

$$d = \frac{|\boldsymbol{w}^T \boldsymbol{x}^t + w_0|}{||\boldsymbol{w}||} = \text{distance from } x^t \text{ to discriminant}$$

The equation for the optimal separating line can be rewritten as:

$$x_2 + x_1 - \frac{3}{2} = 0$$

$$[1 \quad 1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \frac{3}{2} = 0$$

$$\boldsymbol{w}^T \boldsymbol{x}^t + w_0 = 0$$

$$w_0 = -\frac{3}{2}, \qquad \boldsymbol{w}^T = [1 \quad 1], \qquad \boldsymbol{x}^t = \begin{bmatrix} 5.23 \\ 7.14 \end{bmatrix}$$

$$||\boldsymbol{w}|| = \sqrt{1^2 + 1^2} = \sqrt{2}$$

$$\therefore d = \frac{\left|[1 \quad 1]\begin{bmatrix}5.23\\7.14\end{bmatrix} - \frac{3}{2}\right|}{\sqrt{2}}$$

$$d = \frac{\left|5.23 + 7.14 - \frac{3}{2}\right|}{\sqrt{2}}$$

$$d = \frac{10.87}{\sqrt{2}} \approx 7.6863$$

3.

Based on string subsequence kernel (SSK) example between pages 422-423, it is observed that we obtain an 8-dimensional feature space based solely on the words “cat”, “car”, “bat” and “bar”. If we include the word “rat”, with $k = 2$, we obtain the following substrings:

c-a, c-t, a-t, b-a, b-t, c-r, a-r, b-r, r-a, r-t

The substring a-t in “rat” is not included since it would be included twice above. Hence we now obtain a 10-dimensional feature space based on the words “cat”, “car”, “bat”, “bar” and “rat”.

4.

The values of the Arc-cosine kernel k_{ac} and its equivalence k_{approx} as functions of θ (using $\boldsymbol{x}, \boldsymbol{x}'$ and calculating the angle θ between these two vectors) are plotted below in figure 2 for $n \in \{10,100,1000,10000\}$:

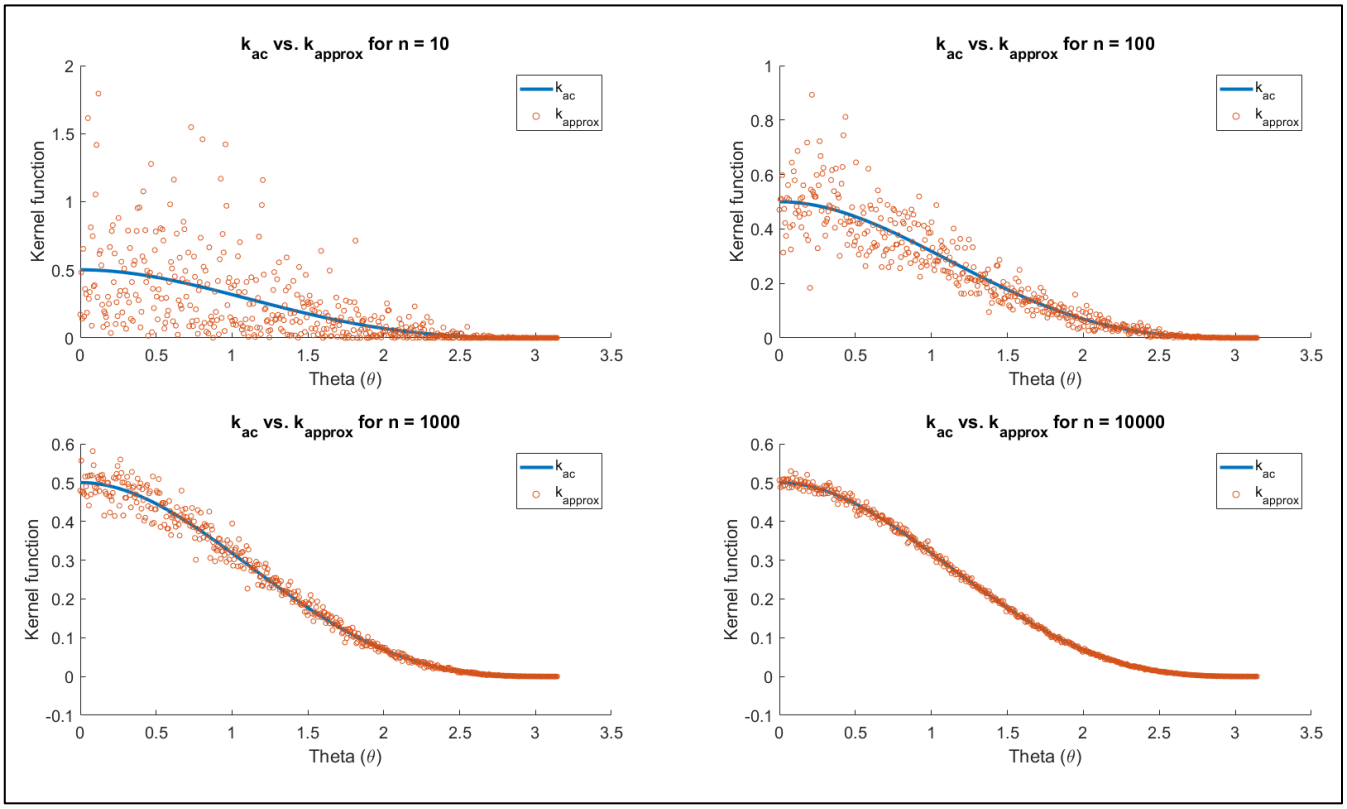

Figure 2: Plot of k_{ac} vs. k_{approx} for $n \in \{10, 100, 1000, 10000\}$

The code $Q4.m$ used to generate these plots is provided below:

```
clear all;


% Read in dataset
data = readtable("hw8.csv");
% Angle between x (first 500 columns) and x_dash (next 500 columns) stored
% in final column (1001th column)
thetas = data{1:end, end};


% Define function to calculate k_ac
k_ac = @(x, x_dash, theta) (norm(x)*norm(x_dash))/(2*pi) * ...
    (sin(theta) + (pi - theta)*cos(theta));


% Store 500 values in list (500 samples total in data)
k_acs = [];
% Calculate k_ac first
```

```
for j = 1:height(data)
    x = data{j, 1:500}';
    x_dash = data{j, 501:1000}';
    theta = data{j, end};

    k_acs = [k_acs; k_ac(x, x_dash, theta)];
end

% List of n to test for k_approx
n = [10, 100, 1000, 10000];
for i = 1:length(n)
    % Store 500 values in list (500 samples total in data)
    k_approxs = [];

    % Calculate k_approx for given n
    for j = 1:height(data)
        x = data{j, 1:500}';
        x_dash = data{j, 501:1000}';
        theta = data{j, end};

        % See k_approx.m function
        k_approxs = [k_approxs; k_approx(x, x_dash, n(i))];
    end

    % Create a subplot comparing k_ac vs. k_approx for given n
    subplot(2, 2, i)
    hold on
    plot(thetas, k_acs, "LineWidth", 3)
    scatter(thetas, k_approxs, 20)
    xlabel("Theta (\theta)")
    ylabel("Kernel function")
    title("k_{ac} vs. k_{approx} for n = " + n(i))
    legend("k_{ac}", "k_{approx}")
    set(gca, 'FontSize', 15)
    hold off
end
```

Furthermore, the code for a helper function $k\_approx.m$ used in $Q4.m$ above to calculate $k_{approx}(\boldsymbol{x}, \boldsymbol{x}')$ is also provided below:

```
function [result] = k_approx(x, x_dash, n)
    % k_approx function to use in Q4.m
    % Inputs:
        % x = first data sample (vector)
        % x_dash = second data sample (vector)
        % n = number of w_i draws from N(0, 1) distribution
    % Output
        % result = value of k_approx

    result = 0;
    % Following formula given to calculate k_approx (see Homework 8 Q4)
    for i = 1:n
        % size(w_i) = size(x)' = (1, 500), each element in w_i ~ N(0, 1)
        w_i = randn(size(x))';
        result = result + max(0, dot(w_i, x)) * max(0, dot(w_i, x_dash));
    end

    % Find average from sum (see formula in Homework 8 Q4)
    result = result/n;
end
```

5.

The equation for k_{approx} describes a neural network with 3 layers – an input layer with 500 neurons where each neuron takes a 2-dimensional input $(x_i, x_i'), i = 1, \dots, 500$ (similar to the input for a convolutional neural network). This is because we wish to feed in both inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ (single sample) simultaneously into the network. We also have 500 neurons because:

$$\dim(\boldsymbol{x}) = \dim(\boldsymbol{x}') = 500 \times 1$$

The next layer is a fully connected hidden layer (each neuron in the previous input layer is connected to every neuron in the current hidden layer) with n hidden units. The corresponding weights between both layers are given by:

$$w_{i,j} \sim N(0,1), \qquad i = 1, \dots, 500, \qquad j = 1, \dots, n$$

Note that there are **no bias weights** included in this network. The activation function used is the ReLU given by $\max(0, x)$ for an input x. We also multiply the corresponding weights only by the first input $\boldsymbol{x}$ at this stage. Combining all this information, for example, the first and last hidden units are calculated as (see figure 3 below):

$$\max\left(0, w_{1,1} \cdot x_1 + w_{2,1} \cdot x_2 + \cdots + w_{500,1} \cdot x_n\right) = \max(0, \boldsymbol{w}_1 \boldsymbol{x})$$

$$\max\left(0, w_{1,n} \cdot x_1 + w_{2,n} \cdot x_2 + \cdots + w_{500,n} \cdot x_n\right) = \max(0, \boldsymbol{w}_n \boldsymbol{x})$$

The final layer is a fully connected output layer with only 1 neuron. The trick here in this layer is to simply make the corresponding weights equal to:

$$\left\{w_{1,1}, \dots, w_{n,1}\right\} = \{\max(0, \boldsymbol{w}_1 \boldsymbol{x}'), \dots, \max(0, \boldsymbol{w}_n \boldsymbol{x}')\}$$

where the $\boldsymbol{w}_j, j = 1, \dots, n$ are the same vectors of weights linking the input layer with the middle-hidden layer and $\boldsymbol{x}'$ is the second input fed into the network.

$$\boldsymbol{w}_j = \begin{bmatrix} w_{1,j} \\ \vdots \\ w_{500,j} \end{bmatrix}, \qquad j = 1, \dots, n$$

Let W be the matrix that stores each $\boldsymbol{w}_j$ as **row** vectors within itself. Then:

$$\dim(W) = n \times 500, \qquad \dim\left(\boldsymbol{w}_j'\right) = 1 \times 500, \qquad \dim(\boldsymbol{x}) = \dim(\boldsymbol{x}') = 500 \times 1$$

The activation function for the output neuron is $\frac{1}{n} < \max(0, W\boldsymbol{x}), \max(0, W\boldsymbol{x}') >$ where $<\cdot>$ denotes the usual inner product (dot product). We can confirm this is the desired output because:

$$\frac{1}{n} < \max(0, W\boldsymbol{x}), \max(0, W\boldsymbol{x}') >$$

$$= \frac{1}{n}(\max(0, \boldsymbol{w}_1\boldsymbol{x}) \cdot \max(0, \boldsymbol{w}_1\boldsymbol{x}') + \cdots + \max(0, \boldsymbol{w}_n\boldsymbol{x}) \cdot \max(0, \boldsymbol{w}_n\boldsymbol{x}'))$$

$$= \frac{1}{n}\sum_{i=1}^{n} \max(0, \boldsymbol{w}_i\boldsymbol{x}) \max(0, \boldsymbol{w}_i\boldsymbol{x}')$$

$$= k_{approx}(\boldsymbol{x}, \boldsymbol{x}') \approx k_{ac}(\boldsymbol{x}, \boldsymbol{x}')$$

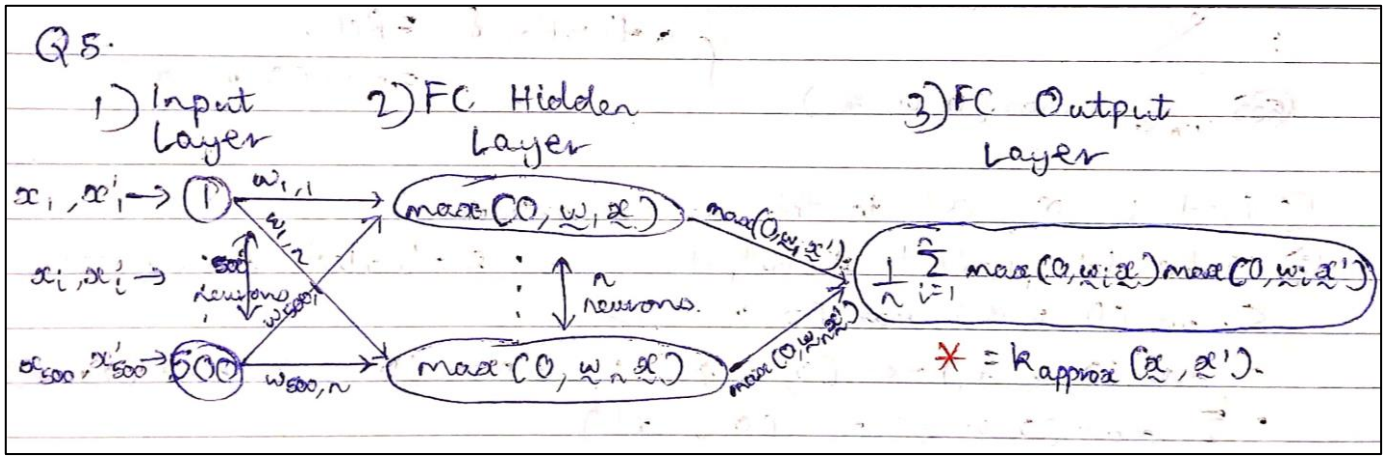

Figure 3: Three-layer neural network model described by $k_{approx}(x, x')$