COMP.4220 - Machine Learning
Homework 7

April 3, 2025

student name: Christian Blake

---

1. Q1? (1 Mark)

Ans:

(a) We want to minimize: $f(x) = 8x_1^2 - 2x_2$

according to the constraint: $x_1^2 + x_2^2 = 1$ First we should form the Lagrangian function:

$$L(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda(\text{constraint} - \text{constant}) \tag{1}$$
$$L(x_1, x_2, \lambda) = 8x_1^2 - 2x_2 - \lambda(x_1^2 + x_2^2 - 1) \tag{2}$$

And now find partial derivatives, set them equal to zero: For x_1:

$$\frac{\partial L}{\partial x_1} = \frac{\partial}{\partial x_1}[8x_1^2 - 2x_2 - \lambda(x_1^2 + x_2^2 - 1)] \tag{3}$$
$$= \frac{\partial}{\partial x_1}[8x_1^2] - \frac{\partial}{\partial x_1}[\lambda x_1^2] - \frac{\partial}{\partial x_1}[\lambda x_2^2] + \frac{\partial}{\partial x_1}[\lambda] \tag{4}$$
$$= 16x_1 - 2\lambda x_1 - 0 + 0 \tag{5}$$
$$= x_1(16 - 2\lambda) = 0 \tag{6}$$

We set this equal to zero, and get two possible outcomes:

$$\text{Either } x_1 = 0 \text{ or } \lambda = 8 \tag{7}$$

For x_2:

$$\frac{\partial L}{\partial x_2} = \frac{\partial}{\partial x_2}[8x_1^2 - 2x_2 - \lambda(x_1^2 + x_2^2 - 1)] \tag{8}$$
$$= \frac{\partial}{\partial x_2}[8x_1^2] - \frac{\partial}{\partial x_2}[2x_2] - \frac{\partial}{\partial x_2}[\lambda x_1^2] - \frac{\partial}{\partial x_2}[\lambda x_2^2] + \frac{\partial}{\partial x_2}[\lambda] \tag{9}$$
$$= 0 - 2 - 0 - 2\lambda x_2 + 0 \tag{10}$$
$$= -2 - 2\lambda x_2 = 0 \tag{11}$$

Solving for x_2:

$$x_2 = -\frac{1}{\lambda} \tag{12}$$

For λ:

$$\frac{\partial L}{\partial \lambda} = -(x_1^2 + x_2^2 - 1) = 0 \tag{13}$$

Manipulating the equation a bit gives our original constraint: $x_1^2 + x_2^2 = 1$
Now solve for critical points: There is two cases to solve:

Case 1: $x_1 = 0$

If $x_1 = 0$, from the constraint:

$$x_1^2 + x_2^2 = 1 \tag{14}$$
$$0^2 + x_2^2 = 1 \tag{15}$$
$$x_2^2 = 1 \tag{16}$$
$$x_2 = \pm 1 \tag{17}$$

From the equation $x_2 = -\frac{1}{\lambda}$:

$$\text{If } x_2 = 1, \text{ then } \lambda = -1 \tag{18}$$
$$\text{If } x_2 = -1, \text{ then } \lambda = 1 \tag{19}$$

This gives us two critical points:

$$(0, 1) \text{ with } \lambda = -1 \tag{20}$$
$$(0, -1) \text{ with } \lambda = 1 \tag{21}$$

Case 2: $\lambda = 8$

If $\lambda = 8$, then $x_2 = -\frac{1}{8}$ (because $x_2 = -\frac{1}{\lambda}$)
From the constraint:

$$x_1^2 + x_2^2 = 1 \tag{22}$$
$$x_1^2 + \left(-\frac{1}{8}\right)^2 = 1 \tag{23}$$
$$x_1^2 + \frac{1}{64} = 1 \tag{24}$$
$$x_1^2 = \frac{63}{64} \tag{25}$$
$$x_1 = \pm\sqrt{\frac{63}{64}} = \pm\frac{\sqrt{63}}{8} \tag{26}$$

Now we have two more critical points:

$$\left(\frac{\sqrt{63}}{8}, -\frac{1}{8}\right) \text{ with } \lambda = 8 \tag{27}$$

$$\left(-\frac{\sqrt{63}}{8}, -\frac{1}{8}\right) \text{ with } \lambda = 8 \tag{28}$$

The last step is to calculate the function value for each critical point: For $(0, 1)$:

$$\begin{aligned} f(0,1) &= 8(0)^2 - 2(1) && (29)\\ &= 0 - 2 && (30)\\ &= -2 && (31) \end{aligned}$$

For $(0, -1)$:

$$\begin{aligned} f(0,-1) &= 8(0)^2 - 2(-1) && (32)\\ &= 0 + 2 && (33)\\ &= 2 && (34) \end{aligned}$$

For $\left(\frac{\sqrt{63}}{8}, -\frac{1}{8}\right)$:

$$\begin{aligned} f\left(\frac{\sqrt{63}}{8}, -\frac{1}{8}\right) &= 8\left(\frac{\sqrt{63}}{8}\right)^2 - 2\left(-\frac{1}{8}\right) && (35)\\ &= 8 \cdot \frac{63}{64} + \frac{2}{8} && (36)\\ &= \frac{504}{64} + \frac{16}{64} && (37)\\ &= \frac{520}{64} && (38)\\ &= 8.125 && (39) \end{aligned}$$

For $\left(-\frac{\sqrt{63}}{8}, -\frac{1}{8}\right)$:

$$f\left(-\frac{\sqrt{63}}{8}, -\frac{1}{8}\right) = 8.125 \text{ (same answer as positive x1 due to square)} \tag{40}$$

Minimum value is -2, which occurs at point $(0, 1)$

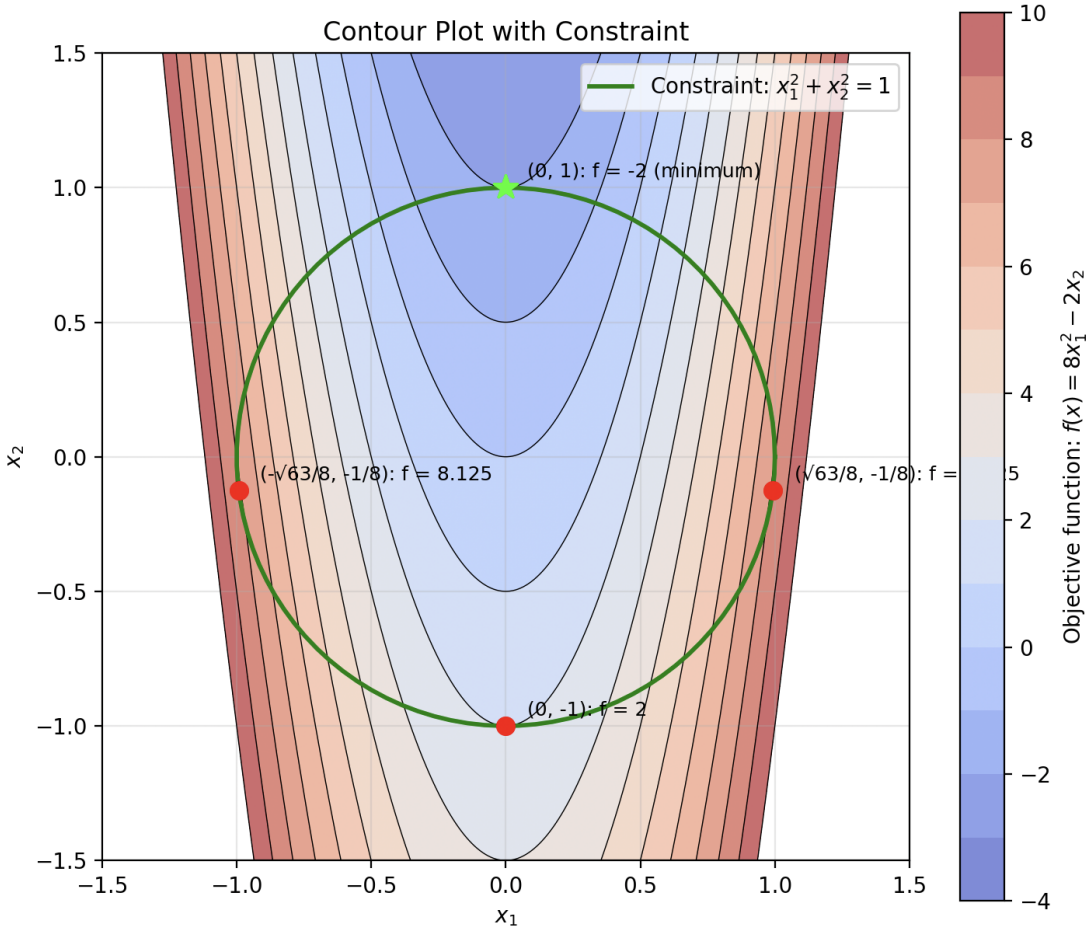

Figure 1: Question 1 Part B

(b)

2. Q2?

Ans:

(a) Question 2 Part B: Dataset description: ========== Labeled Faces in the Wild (LFW) Pairs Dataset: ---------- Training samples shape: (2200, 5828) Test samples shape: (1000, 5828) Target shape: (2200,) Target values: [0 1] Target distribution in training set: [1100 1100] Target distribution in test set: [500 500]

Dataset Information: LFW pairs dataset contains pairs of face images: - Class 0: different people with similar faces - Class 1: same person with different photos

The pairs of images are flattened into a feature vector. Need to determine if the two face images are the same person or not. Features are pixel values in images, count is: 5828

(b) Question 2 Part D:

	Precision	Recall	F1-Score	Support
Class 0	0.63	0.57	0.60	500
Class 1	0.61	0.67	0.64	500
Accuracy			0.62	1000
Macro Avg	0.62	0.62	0.62	1000
Weighted Avg	0.62	0.62	0.62	1000

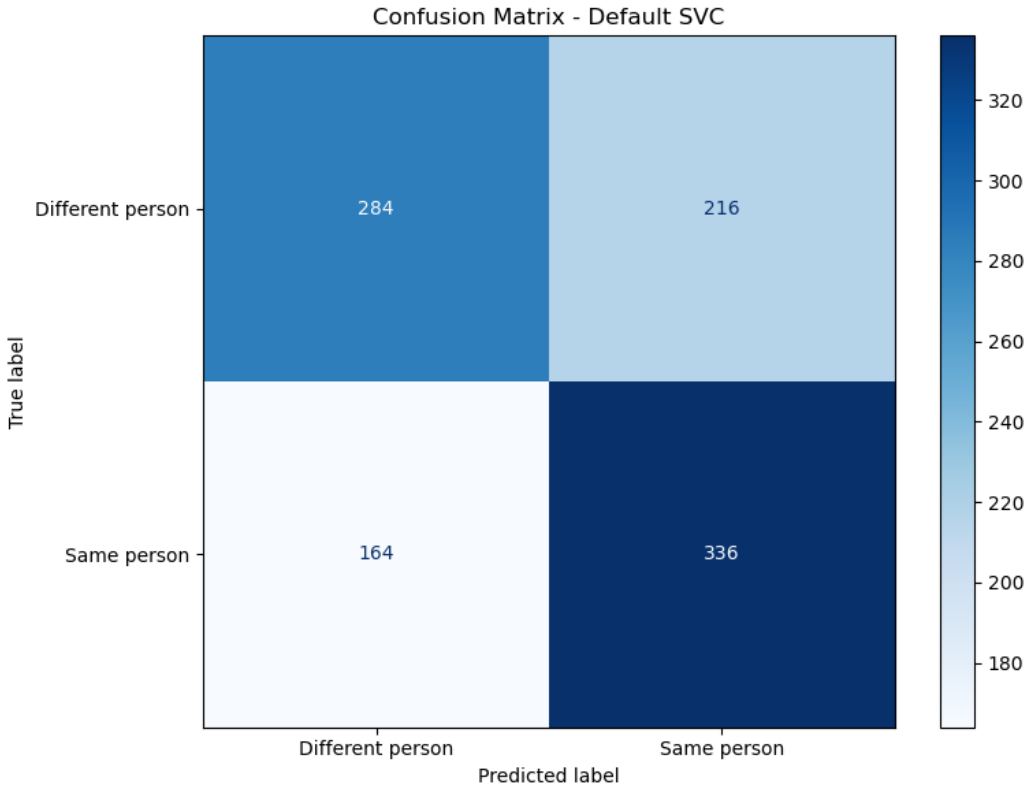

Figure 2: Default SVC Confusion Matrix

(c) Question 2 Part F:

	Precision	Recall	F1-Score	Support
Class 0	0.63	0.59	0.61	500
Class 1	0.61	0.66	0.64	500
Accuracy			0.62	1000
Macro Avg	0.62	0.62	0.62	1000
Weighted Avg	0.62	0.62	0.62	1000

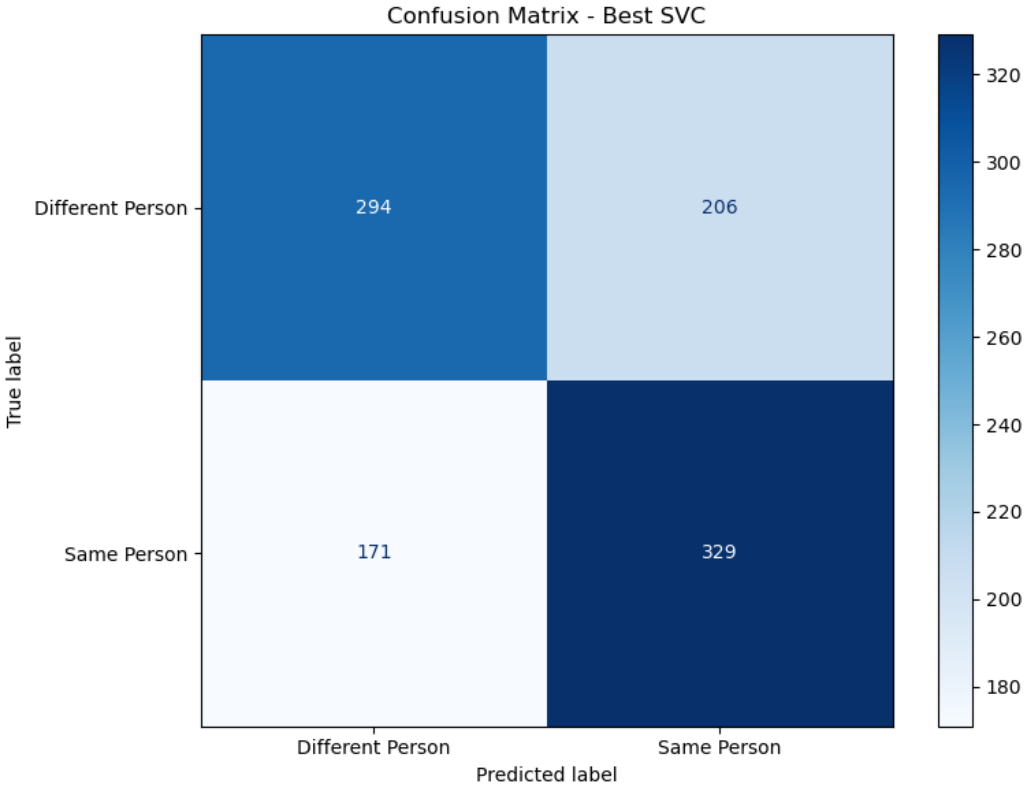

Figure 3: Custom Classifier Confusion Matrix

(d) Question 2 Part G: Initially my classifier worked much worse than the default SVC (Confusion matrix showed one vertical blue column). I believe my classifier was overfitting heavily because I was using the wrong kernel (sigmoid) with a high gamma value, and I also wasn't doing any prepossessing on the pixel values. I then tried to use StandardScaler to scale the data before running the RSCV, and I also messed around with various parameters (I lowered the range for C, narrowed down to rbf and linear for kernel). This gave about the same results as the default, giving the same average of 0.62 precision.

(e) Question 2 Part H: To improve the results I have, I am trying to normalize the data in preprocessing by dividing the X train and X test pixel data by 255. Additionally, I thought I would also try to use PCA to reduce the dimensionality of the dataset, running it through a pipeline with StandardScaler and SVC. The problem I am facing though is the training time is taking very long, with my current run now at 80 minutes. A better understanding of the underlying parameters being tweaked would definitely do me some good. From my understanding, normalizing the data right before using a pipeline is pointless since it is being scaled anyway. I found that the linear kernel took much longer and hung the program on several occasions.