

CMPE 462 FINAL EXAM

Berkay GÜMÜŞ 2015401183

July 4, 2020

1 Question

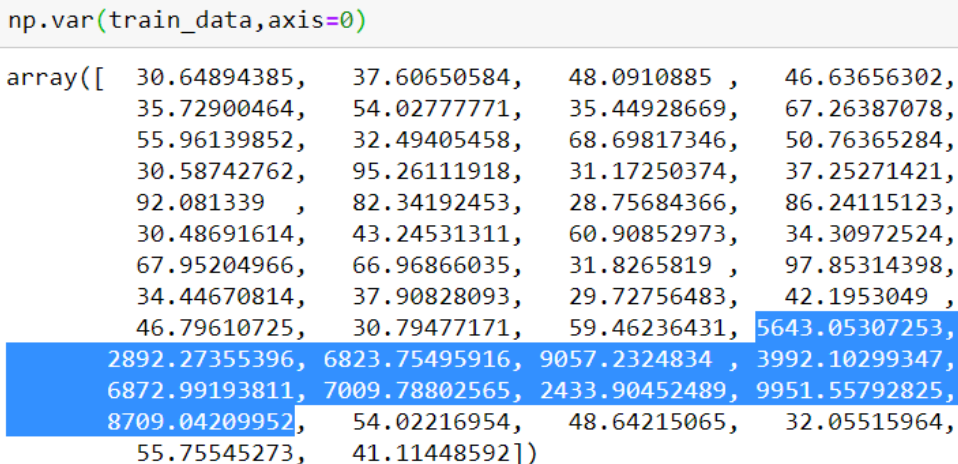
- Part A

The problem must be about the data. The data may not have enough discriminative features, their ranges may be too much different. So, I need to check ranges and discriminations of the features.

- Part B

Firstly, I check ranges. I use numpy variance functions and I observed that 10 features have too high variance whereas other features have too low. It's a big problem because the aim of SVM is to maximize margin (fatness) and it's more sensitive to features with high variance when the data has features with different variances. It tries to classify the data using especially the features with high variance. The other features affect the classification less even if they are more discriminative.

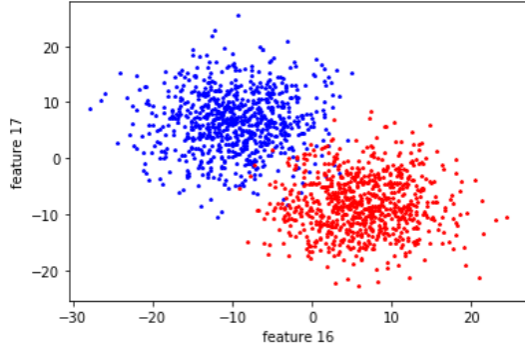
```
np.var(train_data,axis=0)
```



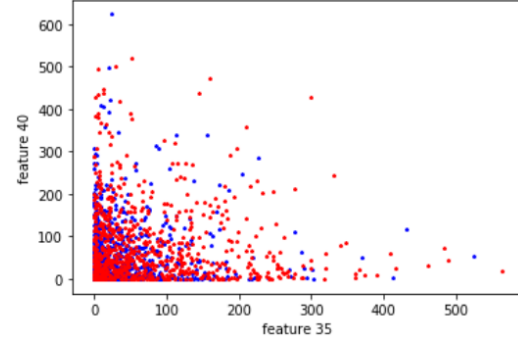
30.64894385,	37.60650584,	48.0910885 ,	46.63656302,
35.72900464,	54.02777771,	35.44928669,	67.26387078,
55.96139852,	32.49405458,	68.69817346,	50.76365284,
30.58742762,	95.26111918,	31.17250374,	37.25271421,
92.081339 ,	82.34192453,	28.75684366,	86.24115123,
30.48691614,	43.24531311,	60.90852973,	34.30972524,
67.95204966,	66.96866035,	31.8265819 ,	97.85314398,
34.44670814,	37.90828093,	29.72756483,	42.1953049 ,
46.79610725,	30.79477171,	59.46236431,	5643.05307253,
2892.27355396,	6823.75495916,	9057.2324834 ,	3992.10299347,
6872.99193811,	7009.78802565,	2433.90452489,	9951.55792825,
8709.04209952,	54.02216954,	48.64215065,	32.05515964,
55.75545273,	41.11448592,		

Figure 1: Variance

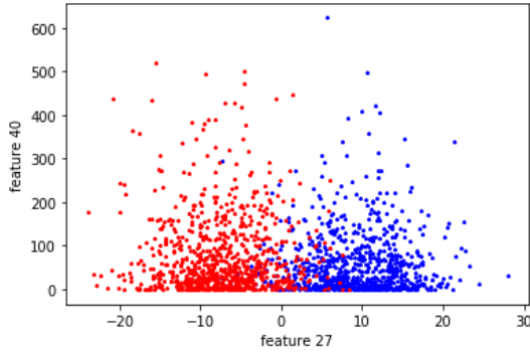
Secondly, I check discriminations of the features by plotting them at 2D plot. Many features are discriminative whereas about 10-15 features are not discriminative. I think these discriminative features (about 35-40 features) are enough to classify data.



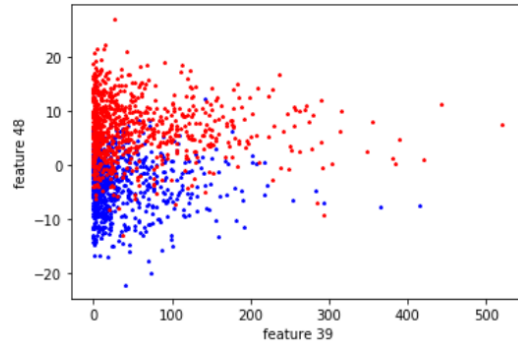
(a) feature 16 and 17 are discriminative



(b) feature 35 and 40 are not discriminative



(c) feature 27 is discriminative, but 40 is not



(d) feature 48 is discriminative, but 39 is not

Figure 2: Features

Thirdly, I check the relation between variances and discriminations. If the features with high variance are not discriminative, the classifier can not be successful. I observed that all features with high variance (feature 35, 36, ..., 44) are not discriminative. This situation is the reason of poor performance at this case.

The solution is that the data must be normalized. After normalization, all features can be used effectively to classify data by SVM. When I apply this solution, the training and test accuracies become 100%.

- Part C

I append training and test data and use sklearn normalization function [1] for the normalization. It makes the all features zero mean unit variance. After that, I split them as training and test data.

```

train_data = np.load('data/train_data1.npy')
train_label = np.load('data/train_label1.npy')
test_data = np.load('data/test_data1.npy')
test_label = np.load('data/test_label1.npy')

q1_data = np.append(train_data, test_data, axis=0)
#np.shape(q1_data)

from sklearn import preprocessing
q1_scaled_data = preprocessing.scale(q1_data)

scaled_train_data = q1_scaled_data[0:1600,:]
scaled_test_data = q1_scaled_data[1600:2000,:]

scaled_clf = svm.SVC(gamma=0.001, C=100.)
scaled_clf.fit(scaled_train_data, train_label)
scaled_train_y_pred = scaled_clf.predict(scaled_train_data)
scaled_train_correct_prediction = np.equal(scaled_train_y_pred, train_label)
scaled_train_accuracy = np.mean(scaled_train_correct_prediction.astype(np.float32))
scaled_train_accuracy

1.0

scaled_test_y_pred = scaled_clf.predict(scaled_test_data)
scaled_test_correct_prediction = np.equal(scaled_test_y_pred, test_label)
scaled_test_accuracy = np.mean(scaled_test_correct_prediction.astype(np.float32))
scaled_test_accuracy

1.0

```

Figure 3: Normalization and SVM Code

```

d1 = 16
d2 = 17
plt.scatter(train_data[train_label==0,d1],train_data[train_label==0,d2],color='blue', s = 5, marker = "**")
plt.scatter(train_data[train_label==1,d1],train_data[train_label==1,d2],color='red', s = 5, marker = "**")
plt.xlabel('feature ' + str(d1))
plt.ylabel('feature ' + str(d2))

```

Figure 4: 2D Plot Code

2 Question

- Part A

The data shape is 1000x2. There are 2 features. I plot the data and check it.

```

q2_data = np.load('data/data2.npy')

plt.scatter(q2_data[:,0],q2_data[:,1],color='blue', s = 13, marker = "**")
plt.title('data')

```

Figure 5: 2D Plot Code

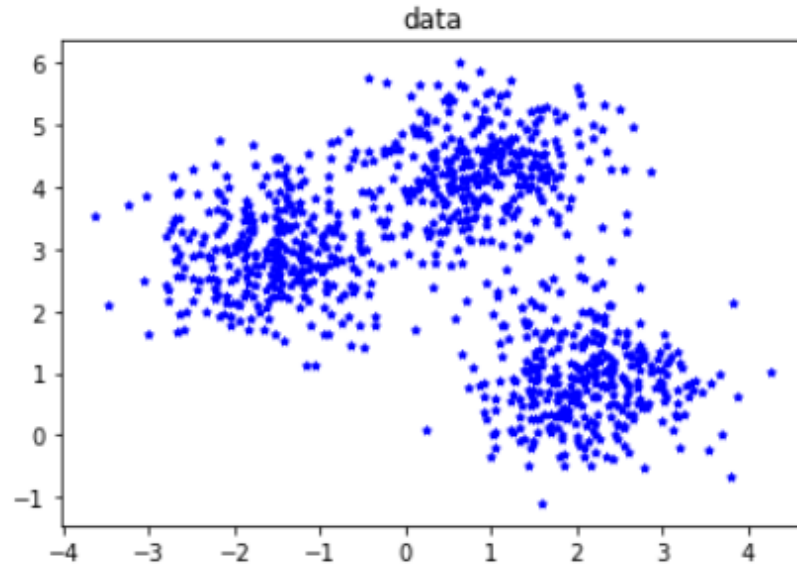


Figure 6: Unlabeled Data

There are 3 main groups at the plot and the points can be separated as 3 clusters easily using these 2 features. The labels can be obtained using K-Means clustering as unsupervised learning technique. I use the sklearn k-means clustering functions [2].

```
In [3]: from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

In [4]: q2_data = np.load('data/data2.npy')

In [5]: kmeans = KMeans(n_clusters=3, random_state=0).fit(q2_data)
q2_label = kmeans.labels_

In [6]: plt.scatter(q2_data[q2_label==0,0],q2_data[q2_label==0,1],color='blue', s = 13, marker = "*")
plt.scatter(q2_data[q2_label==1,0],q2_data[q2_label==1,1],color='red', s = 13, marker = "*")
plt.scatter(q2_data[q2_label==2,0],q2_data[q2_label==2,1],color='green', s = 13, marker = "*")
plt.title('Sklearn Cluster K-Means Assignments')
```

Figure 7: K-Means Clustering and 2D Plot Code

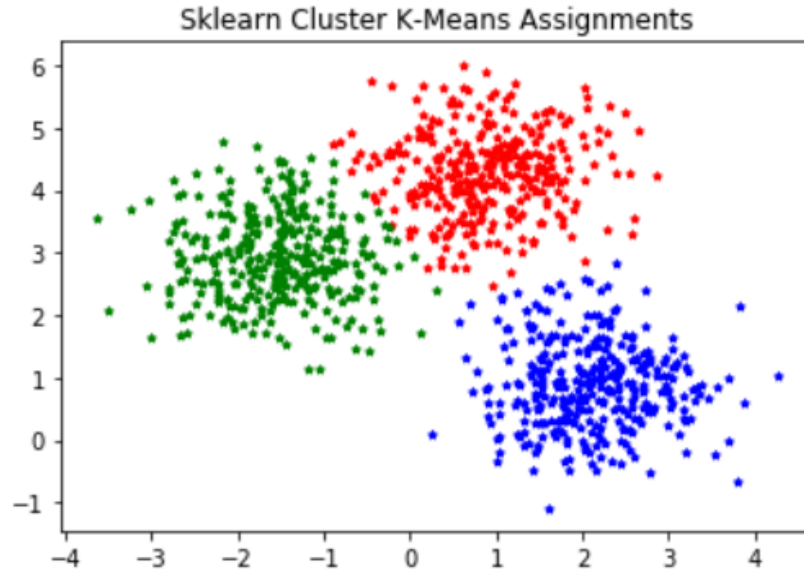


Figure 8: Clusters

- Part B

After labeling and shuffling data, I split the data as training data(80%) and test data(20%). To classify the data, I use SVM as classifier, similarly the 1. question. Gamma is set as 0.1 and C as 100.

```
In [37]: from sklearn.utils import shuffle
X_shuffled,y_shuffled = shuffle(q2_data, q2_label, random_state=0)

In [38]: train_data2 = X_shuffled[0:800,:]
train_label2 = y_shuffled[0:800]
test_data2 = X_shuffled[800:1000,:]
test_label2 = y_shuffled[800:1000]

In [39]: #svm model
q2_clf = svm.SVC(gamma=0.1, C=100.)
q2_clf.fit(train_data2, train_label2)
#training accuracy
y_train_pred2 = q2_clf.predict(train_data2)
correct_train_prediction2 = np.equal(y_train_pred2, train_label2)
train_accuracy2 = np.mean(correct_train_prediction2.astype(np.float32))
train_accuracy2

Out[39]: 0.99875

In [40]: #test accuracy
y_test_pred2 = q2_clf.predict(test_data2)
correct_test_prediction2 = np.equal(y_test_pred2, test_label2)
test_accuracy2 = np.mean(correct_test_prediction2.astype(np.float32))
test_accuracy2

Out[40]: 0.995
```

Figure 9: Classification Code

Training accuracy is 0.99875 and test accuracy is 0.995 (199/200). There is only one mislabeled point at test data.

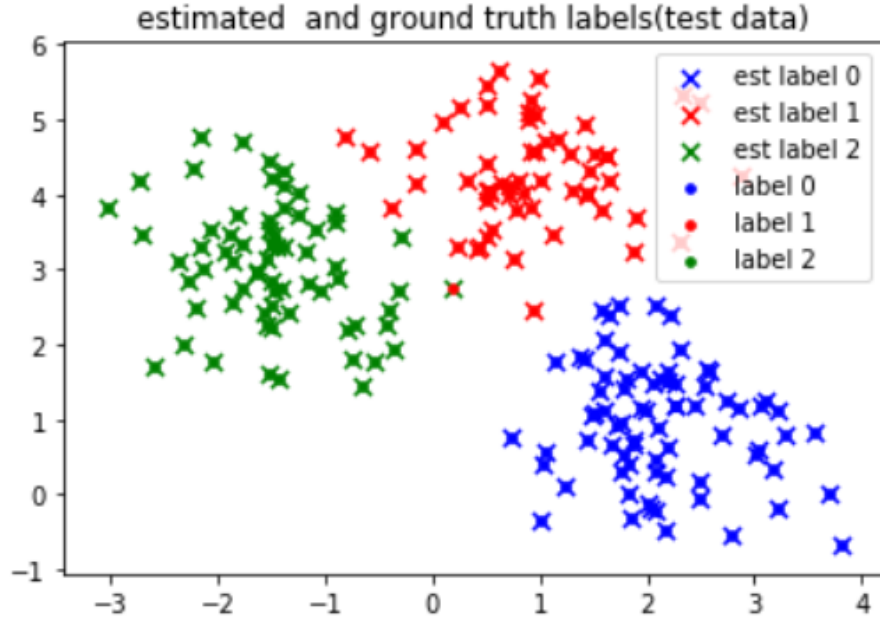


Figure 10: Test Data Classification

```
plt.scatter(test_data2[y_test_pred2==0,0],test_data2[y_test_pred2==0,1],color='blue', s = 50, marker = "x")
plt.scatter(test_data2[y_test_pred2==1,0],test_data2[y_test_pred2==1,1],color='red', s = 50, marker = "x")
plt.scatter(test_data2[y_test_pred2==2,0],test_data2[y_test_pred2==2,1],color='green', s = 50, marker = "x")

plt.scatter(test_data2[test_label2==0,0],test_data2[test_label2==0,1],color='blue', s = 13, marker = "o")
plt.scatter(test_data2[test_label2==1,0],test_data2[test_label2==1,1],color='red', s = 13, marker = "o")
plt.scatter(test_data2[test_label2==2,0],test_data2[test_label2==2,1],color='green', s = 13, marker = "o")

plt.legend(["est label 0", "est label 1", "est label 2", "label 0", "label 1", "label 2"])

plt.title('estimated and ground truth labels(test data)')
```

Figure 11: 2D Plot Code

3 Question

Patient id gives the lowest entropy (zero entropy) because any two patients can not have same ID. The number of node is equal to the number of patients when the root node is patient ID. There is only one patient at each child node.

The patient ID is completely meaningless to learn a generalizable tree because it is a random variable, there is not any correlation with diabetes. Even if there is any correlation, it can not be generalized because there is not another person with the same ID. I think the most meaningful attributes are glucose and insulin, because they are indicative factors to

determine diabetes or health. Also, gender and age can be meaningful to learn a generalizable tree.

4 Question

$$\text{Data} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_{800,1} & x_{800,2} & \dots & x_{800,p} \\ x_{801,1} & x_{801,2} & \dots & x_{801,p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1000,1} & x_{1000,2} & \dots & x_{1000,p} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_{800,1} & x_{800,2} & \dots & x_{800,p} \\ ? & x_{801,2} & \dots & x_{801,p} \\ \vdots & \vdots & \vdots & \vdots \\ ? & x_{1000,2} & \dots & x_{1000,p} \end{bmatrix}$$

I assume that the last 200 customers' first attribute is missing. I represent the attribute i for the first 800 customers (there is not any missing value for any attribute as $x_i \in R^{800}$ and the attribute i for the last 200 customers (the values at the 1. attribute are missing, other values are not missing) as $\tilde{x}_i \in R^{200}$. \tilde{x}_1 is unknown and other vectors are known. The data is represented as:

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \dots & x_p \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_p \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

To implement PCA, the covariance matrix must be calculated. I assume that this data is standardized (zero mean).

$$\begin{aligned} \text{Covariance Matrix: } S \in R^{p \times p} &= \frac{1}{n} X^T X = \frac{1}{n} \begin{bmatrix} \dots & x_1 & \dots & \dots & \tilde{x}_1 & \dots \\ \dots & x_2 & \dots & \dots & \tilde{x}_2 & \dots \\ & \vdots & & & \vdots & \\ \dots & x_p & \dots & \dots & \tilde{x}_p & \dots \end{bmatrix}_{p,1000} \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \dots & x_p \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_p \\ \vdots & \vdots & & \vdots \end{bmatrix}_{1000,p} \\ &= \frac{1}{n} \begin{bmatrix} x_1^T x_1 + \tilde{x}_1^T \tilde{x}_1 & x_1^T x_2 + \tilde{x}_1^T \tilde{x}_2 & \dots & x_1^T x_p + \tilde{x}_1^T \tilde{x}_p \\ x_2^T x_1 + \tilde{x}_2^T \tilde{x}_1 & x_2^T x_2 + \tilde{x}_2^T \tilde{x}_2 & \dots & x_2^T x_p + \tilde{x}_2^T \tilde{x}_p \\ \vdots & \vdots & \dots & \vdots \\ x_p^T x_1 + \tilde{x}_p^T \tilde{x}_1 & x_p^T x_2 + \tilde{x}_p^T \tilde{x}_2 & \dots & x_p^T x_p + \tilde{x}_p^T \tilde{x}_p \end{bmatrix} \end{aligned}$$

$$= \frac{1}{n} \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \dots & x_1^T x_p \\ x_2^T x_1 & x_2^T x_2 & \dots & x_2^T x_p \\ \vdots & \vdots & \dots & \vdots \\ x_p^T x_1 & x_p^T x_2 & \dots & x_p^T x_p \end{bmatrix} + \frac{1}{n} \begin{bmatrix} \tilde{x}_1^T \tilde{x}_1 & \tilde{x}_1^T \tilde{x}_2 & \dots & \tilde{x}_1^T \tilde{x}_p \\ \tilde{x}_2^T \tilde{x}_1 & \tilde{x}_2^T \tilde{x}_2 & \dots & \tilde{x}_2^T \tilde{x}_p \\ \vdots & \vdots & \dots & \vdots \\ \tilde{x}_p^T \tilde{x}_1 & \tilde{x}_p^T \tilde{x}_2 & \dots & \tilde{x}_p^T \tilde{x}_p \end{bmatrix}$$

The eigenvectors with higher eigenvalues for the covariance matrix are the principal components for PCA. At this step, there are 2 options:

- Discarding the customers with missing values

It means all \tilde{x}_i vectors become zero vector and n becomes 800.

$$S_1 = \frac{1}{800} \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \dots & x_1^T x_p \\ x_2^T x_1 & x_2^T x_2 & \dots & x_2^T x_p \\ \vdots & \vdots & \dots & \vdots \\ x_p^T x_1 & x_p^T x_2 & \dots & x_p^T x_p \end{bmatrix}$$

- Imputing the missing attribute with the average value of 800 customers.

Average is zero for all attribute because of standardized. \tilde{x}_0 vector become zero vector and the inner products of \tilde{x}_0 with other vectors become zero.

$$S_2 = \frac{1}{1000} \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \dots & x_1^T x_p \\ x_2^T x_1 & x_2^T x_2 & \dots & x_2^T x_p \\ \vdots & \vdots & \dots & \vdots \\ x_p^T x_1 & x_p^T x_2 & \dots & x_p^T x_p \end{bmatrix} + \frac{1}{1000} \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \tilde{x}_2^T \tilde{x}_2 & \dots & \tilde{x}_2^T \tilde{x}_p \\ \vdots & \vdots & \dots & \vdots \\ 0 & \tilde{x}_p^T \tilde{x}_2 & \dots & \tilde{x}_p^T \tilde{x}_p \end{bmatrix}$$

The S_1 and the first part of S_2 are same ($1/n$ is ignored) and their eigenvectors are same. However, the second part of S_2 changes the eigenvalues and eigenvectors and the principal components (the eigenvectors with higher eigenvalues) will be different for these two approaches.

5 Question

$$x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, y_1 = -1; x_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, y_2 = 1$$

- Part A

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \text{ and bias is } b.$$

$$\begin{aligned} & \underset{w, b}{\text{minimize}} && w_1^2 + w_2^2 \\ & \text{subject to} && y_i(w^T x_i + b) \geq 1, \quad i = 1, 2 \end{aligned} \tag{1}$$

$$\begin{aligned}
& \underset{w, b}{\text{minimize}} && w_1^2 + w_2^2 \\
& \text{subject to} && -(w_1 + b) \geq 1, \\
& && (3w_1 + b) \geq 1
\end{aligned} \tag{2}$$

constraints:

$$-(w_1 + b) \geq 1 \tag{3}$$

$$3w_1 + b \geq 1 \tag{4}$$

The summation of the equation 3 and the equation 4:

$$2w_1 \geq 2 \tag{5}$$

$$w_1 \geq 1 \tag{6}$$

In order to minimize the objection function, w_1 must be 1. w_2 must be 0 because there is not any constraint for w_2 .

$w_1 = 1$ and $w_2 = 0$ The constraints at 3 and 4 become

$$-1 - b \geq 1 \tag{7}$$

$$3 + b \geq 1 \tag{8}$$

Then;

$$b \leq -2 \tag{9}$$

$$b \geq -2 \tag{10}$$

b^* must be -2.

$$w^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, b^* = -2 \text{ and } g(x) = \text{sign}(x_1 - 2)$$

- Part B

$$\begin{aligned}
& \underset{\alpha \in R^2}{\text{maximize}} && \alpha_1 + \alpha_2 - 1/2(y_1^2 \alpha_1^2 x_1^T x_1 + 2y_1 y_2 \alpha_1 \alpha_2 x_1^T x_2 + y_2^2 \alpha_2^2 x_2^T x_2) \\
& \text{subject to} && y_1 \alpha_1 + y_2 \alpha_2 = 0
\end{aligned} \tag{11}$$

$$\begin{aligned}
& \underset{\alpha \in R^2}{\text{maximize}} && \alpha_1 + \alpha_2 - 1/2(\alpha_1^2 - 6\alpha_1 \alpha_2 + 9\alpha_2^2) \\
& \text{subject to} && -\alpha_1 + \alpha_2 = 0
\end{aligned} \tag{12}$$

The constraint becomes $\alpha_1 = \alpha_2$ and the objective function becomes

$$\underset{\alpha \in R^2}{\text{maximize}} \quad 2\alpha_1 - 2\alpha_1^2 \tag{13}$$

When I take the derivative with respect to α_1 ,

$$2 - 4\alpha_1 = 0 \tag{14}$$

α_1 must be 1/2 and α_2 must be 1/2.

- Part C

I used the formulas at the SVM slide [3].

$$w^* = y_1\alpha_1x_1 + y_2\alpha_2x_2 = -1/2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 1/2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$b^* = y_s - y_1\alpha_1x_1^Tx_s - y_2\alpha_2x_2^Tx_s$$

Both y_1 and y_2 are support vectors. I select y_1 as the support vector for the formula above.

$$b^* = y_1 - y_1\alpha_1x_1^Tx_1 - y_2\alpha_2x_2^Tx_1$$

$$b^* = -1 + 1/2 - 1/2 * 3 = -2$$

$$g(x) = \text{sign}(x_1 - 2)$$

The results of the primal and dual formulation are same.

References

- [1] Scikit-learn.org. 2020. 6.3. Preprocessing Data — Scikit-Learn 0.23.1 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/preprocessing.html> [Accessed 4 July 2020].
- [2] Scikit-learn.org. 2020. Sklearn.Cluster.Kmeans — Scikit-Learn 0.23.1 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> [Accessed 3 July 2020].
- [3] Baytaş, İ., 2020. Support Vector Machines Slide.CMPE 462 Machine Learning Bogazici University, p.32.