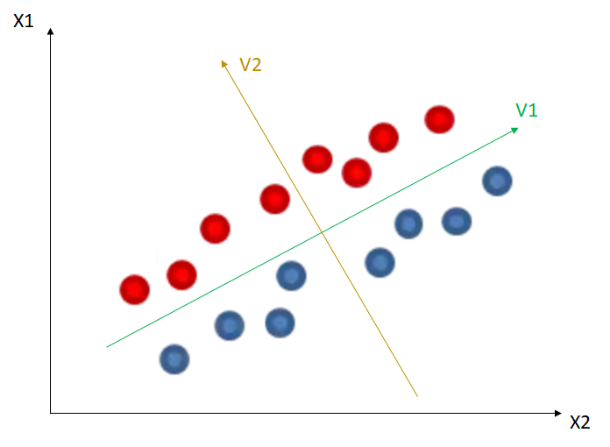
**INFO6105 Data Sci Eng Methods - Final Exam**

**Full Name: Xijing Zhang**

1. Red and blue data points represent two classes in the training data. Which of V1 or V2 is chosen by **LDA** and which one is chosen by **PCA** as the best new feature (component)?

Give reasons for your answer.



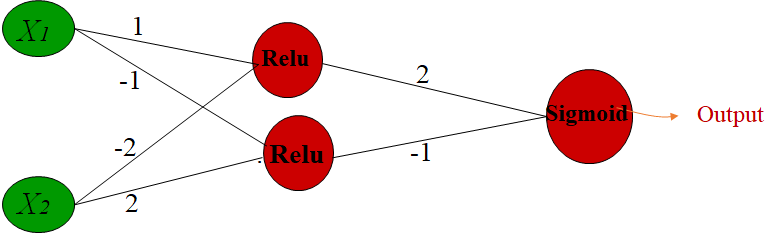
(x1, x2 are current features in the training data)

V1 is chosen by PCA. PCA seeks to reduce the dimensionality of the dataset by identifying the direction (or directions) that capture the maximum variance in the data.

V2 is chosen by LDA. It seeks to maximize the ratio of the between-class variance to the within-class variance in any particular data set thereby ensuring maximum class separability.

1. Consider a **Super Learner** with two base classifiers and a meta learner as follows:
2. A feedforward **Neural Network** (NN) as a base classifier.

Suppose that Relu(x)=Max(x,0) , Sigmoid(x)=0.5 + 0.1x



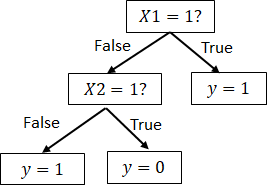
(If Output > 0.5 then y = 1 else y = 0 )

X1 = 1, x2 = 1, relu1 = 0, relu2 = 1, sigmoid = 0.5 – 0.1 = 0.4, y = 0

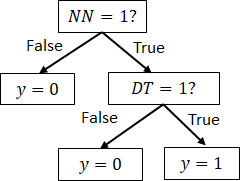
X1 = 1, x2 = 0, relu1 = 1, relu2 = 0, sigmoid = 0.5 + 0.2 + 0 = 0.7, y = 1

X1 = 0, x2 = 1, relu1 = 0, relu2 = 2, sigmoid = 0.5 – 0.2 = 0.3, y = 0

1. A **Decision Tree** (DT) as a base classifier.



1. A decision tree as the **Meta Learner**



Calculate **accuracy** of the super learner model on the following test data:

( Accuracy = Number of correct predicted labels / Total number of predictions )

|  |  |  |
| --- | --- | --- |
| X1 | X2 | y |
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

(X1, X2: features, y: label)

Y = 0, true,

Y = 1, false

Y = 0, false

Accuracy = 1/3

1. For **feature selection** in the wrapper method, use a **Forward** algorithm to select a minimum subset of features providing best accuracy.

Stop adding features if the improvement in accuracy is less than 2%.

|  |  |
| --- | --- |
| Feature(s) | Accuracy of model using feature(s) |
| X1 | 44% |
| X2 | 45% |
| X3 | 42% |
| X1,X2 | 55% |
| X2,X3 | 61% |
| X1,X3 | 61% |
| X1,X2,X3 | 62% |

(X1, X2, X3 are features in the training data)

Step 1:

X1: 44%

X2: 45%

X3: 42%

Among X1, X2, and X3, X2 has the highest initial accuracy at 45%. So, we start with the feature set {X2}.

Step 2:

With X2 selected, the next step is to see which additional feature improves the model's accuracy the most:

X1, X2: 55%

X2, X3: 61%

Adding X3 to X2 improves the accuracy more significantly (from 45% to 61%) compared to adding X1 to X2 (from 45% to 55%). Therefore, we update our feature set to {X2, X3}.

Step 3:

The current feature set {X2, X3} yields an accuracy of 61%. We now consider adding the remaining feature:

X1, X2, X3: 62%

Adding X1 to {X2, X3} increases accuracy from 61% to 62%, which is an improvement of 1%. Since this improvement is less than the 2% threshold, we do not add X1 to the model.

Result: {X2, X3}

1. Calculate the following **probabilities** in a **Naive Bayes** model using the training data.

( P() = count of in training data / total number of training data

P(in ) = count of in / count of in training data )

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Diabetes | No Diabetes |  |  | Diabetes | No Diabetes |
| P() | 2/5 |  | P(High BMI) | 1/2 | 1/3 |  | P(High Glucose) | 2/2 | 1/3 |
| P() | 3/5 |  | P(Low BMI) | 1/2 | 2/3 |  | P(Low Glucose) | 0/2 | 2/3 |
| Total | 1.00 |  | Total | 1.00 | 1.00 |  | Total | 1.00 | 1.00 |

Training Data:

|  |  |  |
| --- | --- | --- |
| BMI | Glucose | Class |
| High | High | Diabetes |
| Low | Low | No Diabetes |
| Low | High | Diabetes |
| High | Low | No Diabetes |
| Low | High | No Diabetes |

(Low BMI, High BMI, Low Glucose, and High Glucose are feature values in the training data )

Using the calculated probabilities in the last step, **find most probable class** (predict class) of a test data with **Low** **BMI** and **High Glucose**

( Most Probable Class = Class with Maximum of [p() \* p(in ) \* p( in )] )

P(Diabetes) \* P(BMI=Low ∣Diabetes) \* P(Glucose=High∣ Diabetes) = 2/5 \* 1/2 \* 2/2 = 0.2

P(NoDiabetes) \* P(BMI=Low ∣NoDiabetes) \* P(Glucose=High∣ NoDiabetes) = 3/5 \* 2/3 \* 1/3 = 0.1334

Therefore, the predicted class for this test instance is Diabetes.

1. Employ **K-means Clustering** for creating **two clusters** on the following training data using Euclidean distance. Suppose that Patient3 is selected as the initial center of cluster one () and Patient4 is selected as the initial center of cluster two ().

( Euclidean distance between and =

New center of = Sum of BMI values in / Number of patients in )

|  |  |
| --- | --- |
|  | BMI |
| Patient 1 | 22 |
| Patient 2 | 21 |
| Patient 3 | 28 |
| Patient 4 | 30 |
| Patient 5 | 17 |

Step 1: Initialize the clusters

Given initial centers:

Cluster 1 (Center\_1) = BMI of Patient3 = 28

Cluster 2 (Center\_2) = BMI of Patient4 = 30

Step 2: Assign each point to the nearest cluster center

Calculate the Euclidean distance from each point (BMI value) to both cluster centers (28 and 30).

Assign each point to the cluster whose center it is closest to.

Distance calculations:

For Cluster 1 (Center\_1 = 28):

Distance from Patient 1 (BMI = 22) to Center\_1 (28) = |22 - 28| = 6

Distance from Patient 2 (BMI = 21) to Center\_1 (28) = |21 - 28| = 7

Distance from Patient 3 (BMI = 28) to Center\_1 (28) = |28 - 28| = 0

Distance from Patient 4 (BMI = 30) to Center\_1 (28) = |30 - 28| = 2

Distance from Patient 5 (BMI = 17) to Center\_1 (28) = |17 - 28| = 11

For Cluster 2 (Center\_2 = 30):

Distance from Patient 1 (BMI = 22) to Center\_2 (30) = |22 - 30| = 8

Distance from Patient 2 (BMI = 21) to Center\_2 (30) = |21 - 30| = 9

Distance from Patient 3 (BMI = 28) to Center\_2 (30) = |28 - 30| = 2

Distance from Patient 4 (BMI = 30) to Center\_2 (30) = |30 - 30| = 0

Distance from Patient 5 (BMI = 17) to Center\_2 (30) = |17 - 30| = 13

Step 3: Assign clusters based on distances

Assign patients to the cluster with the closest center:

Cluster 1 (Center\_1 = 28): Patients 3, 4

Cluster 2 (Center\_2 = 30): Patients 1, 2, 5

Step 4: Update cluster centers

Calculate new cluster centers (mean BMI) for each cluster:

Cluster 1 (Center\_1 = 28): New center = (28 + 30) / 2 = 29

Cluster 2 (Center\_2 = 30): New center = (22 + 21 + 17) / 3 = 20

Step 5: Repeat until convergence

Repeat steps 2-4 until cluster assignments stabilize:

Calculate distances to updated centers.

Reassign points based on the closest center.

Recalculate centers based on the assigned points.

Final Clusters:

Cluster 1 (Center\_1 = 29): Patients 3, 4

Cluster 2 (Center\_2 = 20): Patients 1, 2, 5

Code:

```

import numpy as np

# Given BMI values for each patient

bmi\_values = np.array([22, 21, 28, 30, 17])

# Initialize cluster centers (given initial centers)

center\_1 = 28

center\_2 = 30

# Define a function to perform K-means clustering

def k\_means\_clustering(data, center\_1, center\_2, max\_iterations=100):

# Initialize cluster assignments

cluster\_assignments = np.zeros(len(data))

for \_ in range(max\_iterations):

# Assign each data point to the nearest cluster center

for i in range(len(data)):

# Calculate Euclidean distances to each center

distance\_to\_center\_1 = abs(data[i] - center\_1)

distance\_to\_center\_2 = abs(data[i] - center\_2)

# Determine the closest cluster center

if distance\_to\_center\_1 <= distance\_to\_center\_2:

cluster\_assignments[i] = 1 # Assign to Cluster 1

else:

cluster\_assignments[i] = 2 # Assign to Cluster 2

# Update cluster centers based on the mean of assigned data points

cluster\_1\_indices = np.where(cluster\_assignments == 1)[0]

cluster\_2\_indices = np.where(cluster\_assignments == 2)[0]

# Calculate new cluster centers

new\_center\_1 = np.mean(data[cluster\_1\_indices])

new\_center\_2 = np.mean(data[cluster\_2\_indices])

# Check for convergence

if new\_center\_1 == center\_1 and new\_center\_2 == center\_2:

break

# Update centers for the next iteration

center\_1 = new\_center\_1

center\_2 = new\_center\_2

return cluster\_assignments, center\_1, center\_2

# Perform K-means clustering

cluster\_assignments, final\_center\_1, final\_center\_2 = k\_means\_clustering(bmi\_values, center\_1, center\_2)

# Print final cluster assignments and centers

print("Final Cluster Assignments:", cluster\_assignments)

print("Final Cluster Center 1:", final\_center\_1)

print("Final Cluster Center 2:", final\_center\_2)

```

Result:

Final Cluster Assignments: [1. 1. 2. 2. 1.]

Final Cluster Center 1: 20.0

Final Cluster Center 2: 29.0