

Aman Chauhan - achauha3
Khantil Choksi – khchoksi

Q1 D-Separation

(1) $d\text{-sep}(B,D / A)$

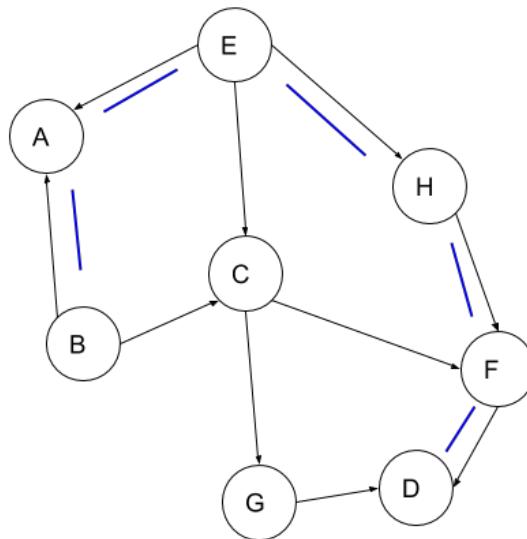
$$X_1 = B$$

$$X_3 = D$$

$$X_2 = A$$

$$B \rightarrow A \leftarrow E \rightarrow H \rightarrow F \rightarrow D$$

- A is converging connection and it is in X_2 ; therefore doesn't block information
- E is diverging connection and it is not in X_2 ; therefore doesn't block information
- H is serial connection and it is not in X_2 ; therefore doesn't block information
- F is serial connection and it is not in X_2 ; therefore doesn't block information
- Therefore, for this given path, B and D are not-separated given A.
(According to blue path shown in figure)



There are also other paths present which is also not blocking information.

$$B \rightarrow A \leftarrow E \rightarrow C \rightarrow F \rightarrow D$$

$$B \rightarrow A \leftarrow E \rightarrow C \leftarrow G \rightarrow D$$

$$B \rightarrow C \leftarrow G \rightarrow D$$

Conclusion: B and D are not d-separated given {A}.

(2) $d\text{-sep}(A,D / \{C,H\})$

$X_1 = A$

$X_3 = D$

$X_2 = \{C,H\}$

$A \leftarrow E \rightarrow H \rightarrow F \rightarrow D$

- E is diverging connection and it is not in X_2 ; therefore doesn't block information
- H is serial connection and in X_2 ; therefore does block information
- Therefore, A and D are separated by this path.

Consider this alternative path:

$A \leftarrow E \rightarrow C \rightarrow F \rightarrow D$

- E is diverging connection and it is not in X_2 ; therefore doesn't block information
- C is serial connection and in X_2 ; therefore does block information
- Therefore, A and D are separated by this path.

Consider this alternative path:

$A \leftarrow E \rightarrow C \rightarrow G \rightarrow D$

- E is diverging connection and it is not in X_2 ; therefore doesn't block information
- C is serial connection and in X_2 ; therefore does block information
- Therefore, A and D are separated by this path.

Consider the following alternative path:

$A \leftarrow B \rightarrow C \rightarrow G \rightarrow D$

$A \leftarrow E \rightarrow C \rightarrow F \rightarrow D$

In all above paths,

- C is serial connection and in X_2 ; therefore does block information
- Therefore, A and D are separated by this path.

Conclusion: Therefore, A and D are d-separated given $\{C,H\}$.

(3) $d\text{-sep}(A,B / \{F,E\})$

$X_1 = A$

$X_3 = B$

$X_2 = \{F,E\}$

$A \rightarrow E \rightarrow G \rightarrow F \rightarrow B$

- E is serial connection and it is in X_2 ; therefore does block information

- Therefore, A and B are d-separated given {F,E}.

Consider this path:

$$A \rightarrow E \leftarrow H \rightarrow F \rightarrow B$$

- E is converging connection and it is in X_2 ; therefore doesn't block information.
- H is diverging connection and not in X_2 ; therefore doesn't block information.
- F is serial connection and not in X_2 ; therefore doesn't block information.
- Therefore, A and B are not d-separated.

Conclusion: Therefore, we found this one path which proves A and B are not d-separated given {F,E}.

(4) d-sep(C,D / {B})

$$X_1 = C$$

$$X_3 = D$$

$$X_2 = \{B\}$$

$$C \leftarrow E \rightarrow G \rightarrow F \leftarrow D$$

- E is diverging connection and it is in X_2 ; therefore doesn't block information.
- G is serial connection and not in X_2 ; therefore doesn't block information.
- F is converging connection and not in X_2 ; therefore does block information.
- Therefore, C and D are d-separated given {B}.

Consider this path:

$$C \leftarrow E \rightarrow H \rightarrow F \leftarrow D$$

- E is diverging connection and it is in X_2 ; therefore doesn't block information.
- H is serial connection and not in X_2 ; therefore doesn't block information.
- F is converging connection and not in X_2 ; therefore does block information.
- Therefore, C and D are d-separated given {B}.

Conclusion: C and D are d-separated given {B}.

Q2

(a) Compute $P(E)$

$$P(E) = P(E | B) * P(B) + P(E | \sim B) * P(\sim B)$$

$$P(B) = P(B | A) * P(A) + P(B | \sim A) * P(\sim A)$$

$$\begin{aligned} P(E) &= P(E | B) * P(B | A) * P(A) + \\ &\quad P(E | B) * P(B | \sim A) * P(\sim A) + \\ &\quad P(E | \sim B) * P(\sim B | A) * P(A) + \\ &\quad P(E | \sim B) * P(\sim B | \sim A) * P(\sim A) \end{aligned}$$

$$\begin{aligned} &= 0.6 * 0.2 * 0.75 + \\ &\quad 0.6 * 0.5 * 0.25 + \\ &\quad 0.3 * 0.8 * 0.75 + \\ &\quad 0.3 * 0.5 * 0.25 \\ &= 0.09 + 0.075 + 0.18 + 0.0375 \\ &= 0.3825 \end{aligned}$$

(b) Compute $P(\sim B, C, D, E)$

$$\begin{aligned} P(\sim B, C, D, E) &= P(D | \sim B, C) * P(C) * P(E | \sim B) * P(\sim B) \\ &= P(D | \sim B, C) * P(C | A) * P(E | \sim B) * P(\sim B | A) * P(A) \\ &\quad + P(D | \sim B, C) * P(C | \sim A) * P(E | \sim B) * P(\sim B | \sim A) * P(\sim A) \\ &= 0.1 * 0.7 * 0.3 * 0.8 * 0.75 + (0.1 * 0.25 * 0.3 * 0.5 * 0.25) \\ &= 0.0126 + 0.0009375 \\ &= 0.0135375 \end{aligned}$$

(c) Compute $P(D | A)$

$$\begin{aligned} P(A, B, C, D) &= P(D | B, C) * P(B | A) * P(C | A) * P(A) \\ &= 0.3 * 0.2 * 0.7 * 0.75 \\ &= 0.0315 \end{aligned}$$

$$\begin{aligned} P(A, B, \sim C, D) &= P(D | B, \sim C) * P(B | A) * P(\sim C | A) * P(A) \\ &= 0.25 * 0.2 * 0.3 * 0.75 \\ &= 0.01125 \end{aligned}$$

$$\begin{aligned} P(A, \sim B, C, D) &= P(D | \sim B, C) * P(\sim B | A) * P(C | A) * P(A) \\ &= 0.1 * 0.8 * 0.7 * 0.75 \\ &= 0.042 \end{aligned}$$

$$P(A, \sim B, \sim C, D) = P(D | \sim B, \sim C) * P(\sim B | A) * P(\sim C | A) * P(A)$$

$$\begin{aligned} &= 0.35 * 0.8 * 0.3 * 0.75 \\ &= 0.063 \end{aligned}$$

$$\begin{aligned} P(D,A) &= P(A,B,C,D) + P(A,B,\sim C,D) + P(A,\sim B,C,D) + P(A,\sim B,\sim C,D) \\ &= 0.0315 + 0.01125 + 0.042 + 0.063 \\ &= 0.14775 \end{aligned}$$

$$\begin{aligned} P(D | A) &= P(D,A) / P(A) \\ &= 0.14775 / 0.75 \\ &= 0.197 \end{aligned}$$

Q3 - Linear Regression

Import Libraries

```
1 from sklearn.model_selection import LeaveOneOut, cross_val_predict
2 from sklearn.linear_model import LinearRegression
3 from sklearn.metrics import mean_squared_error
4 import numpy as np
5 import math
6 import sys
7 import os
```

(a) Given the following three data points of (x, y) : (1, 2), (2, 1), (0, -1), try to use a linear regression $y = \beta_0 + \beta_1 x$ to predict y. Determine the values of β_1 and β_0 and show each step of your work.

x	y	xy	x^2	y^2
1	2	2	1	4
2	1	2	4	1
0	-1	0	0	1
$\sum_i x_i = 3$	$\sum_i y_i = 2$	$\sum_i x_i y_i = 4$	$\sum_i x_i^2 = 5$	$\sum_i y_i^2 = 6$

We want to predict the dependent random variable y , and it is given by $y' = \beta_0 + \beta_1 x$

$$\beta_0 = \frac{\sum_i y_i \sum_i x_i^2 - \sum_i x_i \sum_i x_i y_i}{n(\sum_i x_i^2) - (\sum_i x_i)^2} = \frac{2*5 - 3*4}{3*5 - 3*3} = \frac{-1}{3}$$

$$\beta_1 = \frac{n(\sum_i x_i y_i) - \sum_i x_i \sum_i y_i}{n(\sum_i x_i^2) - (\sum_i x_i)^2} = \frac{3*4 - 3*2}{3*5 - 3*3} = 1$$

Thus the equation is $y' = \frac{-1}{3} + x$

(b) Linear Regression Programming Assignment

Apply the following three linear regressions: (1) $y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4$ (2) $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$ (3) $y = \gamma_0 + \gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3 + \gamma_4 x_4$ **to the provided data file "hw3q3(b).csv", which is from a combined cycle power plant dataset (<https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant>)**. In the given data file, x_i are features and y is the prediction target which indicates hourly electrical energy output.

(i) Load the data. Fit the whole dataset to the three linear regression models, respectively. Report the coefficients (alphas, betas, gammas) of the three models.

```

1 def data_and_headers(filename):
2     data = None
3     with open(filename) as fp:
4         data = [x.strip().split(',') for x in fp.readlines()]
5     headers = data[0]
6     headers = np.asarray(headers)
7     class_field = len(headers) - 1
8     data_x = [[float(x[i]) for i in range(class_field)] for x in data[1:]]
9     data_x = np.asarray(data_x)
10    data_y = [[float(x[i]) for i in range(class_field, class_field + 1)] for x in data[1:]]
11    for x in data[1:]:
12        data_y = np.asarray(data_y)
13    return headers, data_x, data_y

```

```

1 headers, features_x, labels_y = data_and_headers('Data' + os.sep +
'hw3q3(b).csv')

```

```

1 modela = LinearRegression().fit(features_x, labels_y.flatten())
2 modelb = LinearRegression().fit(features_x**2, labels_y.flatten())
3 modelc = LinearRegression().fit(features_x**3, labels_y.flatten())
4 print('Coefficients of Simple LR - \t{}, Intercept -\n\t{:.4f}'.format(modela.coef_, modela.intercept_))
5 print('Coefficients of Quadratic LR - \t{}, Intercept -\n\t{:.4f}'.format(modelb.coef_, modelb.intercept_))
6 print('Coefficients of Cubic LR - \t{}, Intercept -\n\t{:.4f}'.format(modelc.coef_, modelc.intercept_))

```

Output -

```

1 Coefficients of Simple LR -      [-12.38926535   2.80059786 -12.32760055
-64.67916351], Intercept - 500.2071
2 Coefficients of Quadratic LR - [ -6.05581029   7.28426322 -15.38358205
-54.34236701], Intercept - 477.0979
3 Coefficients of Cubic LR -   [  1.24877688  16.3800381 -24.20328807
-43.63328247], Intercept - 466.2837

```

The above output can be interpreted as follows, where $c_i \in \{\alpha_i, \beta_i, \gamma_i\}$ and $i \in \{0, 1, 2, 3, 4\}$ -

Equation	c_0	c_1	c_2	c_3	c_4
$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4$	500.2071	-12.3892	2.8005	-12.3276	-64.6791
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$	477.0979	-6.0558	7.2842	-15.3836	-54.3423
$y = \gamma_0 + \gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3 + \gamma_4 x_4$	466.2837	1.2487	16.3800	-24.2032	-43.6333

(ii) Use leave-one-out cross validation to determine the RMSE (root mean square error) for the three models. Specifically, in each fold, fit the training data to the model to determine the coefficients, then apply the coefficients to get predicted label for testing data (You don't need to report the coefficients in each fold). Report RMSE for the three models. Based on the RMSE, which model is the best for fitting the given data?

```

1 model1 = LinearRegression()
2 model2 = LinearRegression()
3 model3 = LinearRegression()
4 loocv = LeaveOneOut()
5 ypred1 = cross_val_predict(model1, features_x, labels_y.flatten(),
cv=loocv)
6 ypred2 = cross_val_predict(model2, features_x**2, labels_y.flatten(),
cv=loocv)
7 ypred3 = cross_val_predict(model3, features_x**3, labels_y.flatten(),
cv=loocv)
8 print('Normal LR RMSE -
{: .4f}'.format(math.sqrt(mean_squared_error(labels_y.flatten(), ypred1))))
9 print('Quadratic LR RMSE -
{: .4f}'.format(math.sqrt(mean_squared_error(labels_y.flatten(), ypred2))))
10 print('Cubic LR RMSE -
{: .4f}'.format(math.sqrt(mean_squared_error(labels_y.flatten(), ypred3))))

```

Output -

```
1 | Normal LR RMSE - 4.4927
2 | Quadartic LR RMSE - 6.4587
3 | Cubic LR RMSE - 8.0864
```

Based on the RMSE, simple linear regression is the best for fitting the given data.

Q4 - Artificial Neural Networks

Train, validate, and test a neural network model using the dataset in hw3q4.zip, which contains training data (75%), validation data (12.5%), and test data (12.5%). There are two output classes in this data set. You can either choose matlab or a python neural networks package, Keras for this problem.

Import Libraries

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import zipfile
4 import math
5 import sys
6 import os
7 np.random.seed(7)
8 from tensorflow.keras.layers import Activation
9 from tensorflow.keras.layers import Input
10 from tensorflow.keras.layers import Dense
11 from tensorflow.keras import Model
12 import tensorflow as tf
13 tf.set_random_seed(7)
14 %matplotlib inline
```

(a) Please briefly describe how to construct your working environments (e.g. language, package version, backend for neural networks, installation, etc.) in your report, and write how to execute your codes on 'readme' file.

Language

- python 3.7

Prerequisites

- tensorflow 1.8+ (CPU only)

- matplotlib
- numpy

Installation

Install the above packages using the following commands -

- tensorflow 1.8+ (CPU only) - `pip install tensorflow`
- matplotlib - `pip install matplotlib`
- numpy - `pip install numpy`

Data

hw3q4.zip should be in the same directory level as this notebook.

(b) Keras Model

Helper Functions

```

1 def get_data(zipname,filename):
2     zf = zipfile.ZipFile(zipname)
3     data = None
4     with zf.open(filename, 'r') as fp:
5         data = np.asarray([[float(v.strip()) for v in
x.decode().strip().split(',')]] for x in fp.readlines()])
6     return data

```

```

1 def ANN(n_neurons, output_dims):
2     input_layer = Input(batch_shape = (None, 61), name='input_layer')
3     layer = Dense(n_neurons, name='hidden_layer')(input_layer)
4     layer = Activation(activation='relu', name='relu')(layer)
5     layer = Dense(output_dims, name='output_layer')(layer)
6     output_layer = Activation(activation='sigmoid', name='sigmoid')(layer)
7     model = Model(inputs=input_layer, outputs=output_layer, name='ann')
8     model.compile('adam', 'mse', ['accuracy'])
9     return model

```

Read Data

```

1 train_x = get_data('Data' + os.sep + 'hw3q4.zip', 'hw3q4' + os.sep +
2 'X_train.csv')
3 train_y = get_data('Data' + os.sep + 'hw3q4.zip', 'hw3q4' + os.sep +
4 'Y_train.csv')
5 val_x = get_data('Data' + os.sep + 'hw3q4.zip', 'hw3q4' + os.sep +
6 'X_val.csv')
7 val_y = get_data('Data' + os.sep + 'hw3q4.zip', 'hw3q4' + os.sep +
8 'Y_val.csv')
9 test_x = get_data('Data' + os.sep + 'hw3q4.zip', 'hw3q4' + os.sep +
10 'X_test.csv')
11 test_y = get_data('Data' + os.sep + 'hw3q4.zip', 'hw3q4' + os.sep +
12 'Y_test.csv')

```

(1) Construct neural networks using the given training dataset (X train, Y train) using different number of hidden neurons. Set the parameters as follows: activation function for hidden layer='relu', activation for output layer ='sigmoid', loss function ='mse', metrics= 'accuracy', epochs=10, batch size=50. For each model, change the number of hidden neurons in the order of 2, 4, 6, 8, 10.

```

1 hidden_neurons = [2,4,6,8,10]
2 models = [ANN(hidden_neurons[i], train_y.shape[1]) for i in
range(len(hidden_neurons))]
3 histories = []
4 for i in range(len(models)):
5     print('Training Model with {} neurons in hidden
layer'.format(hidden_neurons[i]))
6     history = models[i].fit(x=train_x,
7                             y=train_y,
8                             batch_size=50,
9                             epochs=10,
10                            shuffle=True)
11    histories.append(history)
12    print()

```

```

1 Training Model with 2 neurons in hidden layer
2 Epoch 1/10
3 1500/1500 [=====] - 0s 273us/step - loss: 0.2538
- acc: 0.5333
4 Epoch 2/10

```

```
5 1500/1500 [=====] - 0s 25us/step - loss: 0.2481
- acc: 0.5533
6 Epoch 3/10
7 1500/1500 [=====] - 0s 24us/step - loss: 0.2432
- acc: 0.5727
8 Epoch 4/10
9 1500/1500 [=====] - 0s 25us/step - loss: 0.2387
- acc: 0.5907
10 Epoch 5/10
11 1500/1500 [=====] - 0s 24us/step - loss: 0.2338
- acc: 0.6087
12 Epoch 6/10
13 1500/1500 [=====] - 0s 25us/step - loss: 0.2285
- acc: 0.6193
14 Epoch 7/10
15 1500/1500 [=====] - 0s 22us/step - loss: 0.2229
- acc: 0.6307
16 Epoch 8/10
17 1500/1500 [=====] - 0s 22us/step - loss: 0.2164
- acc: 0.6347
18 Epoch 9/10
19 1500/1500 [=====] - 0s 22us/step - loss: 0.2095
- acc: 0.6480
20 Epoch 10/10
21 1500/1500 [=====] - 0s 25us/step - loss: 0.2022
- acc: 0.6733
22
23 Training Model with 4 neurons in hidden layer
24 Epoch 1/10
25 1500/1500 [=====] - 0s 195us/step - loss: 0.3337
- acc: 0.4927
26 Epoch 2/10
27 1500/1500 [=====] - 0s 25us/step - loss: 0.3077
- acc: 0.5280
28 Epoch 3/10
29 1500/1500 [=====] - 0s 24us/step - loss: 0.2827
- acc: 0.5680
30 Epoch 4/10
31 1500/1500 [=====] - 0s 25us/step - loss: 0.2596
- acc: 0.5967
32 Epoch 5/10
33 1500/1500 [=====] - 0s 37us/step - loss: 0.2387
- acc: 0.6307
34 Epoch 6/10
35 1500/1500 [=====] - 0s 27us/step - loss: 0.2213
- acc: 0.6547
```

```
36 Epoch 7/10
37 1500/1500 [=====] - 0s 24us/step - loss: 0.2066
- acc: 0.6827
38 Epoch 8/10
39 1500/1500 [=====] - 0s 23us/step - loss: 0.1944
- acc: 0.7133
40 Epoch 9/10
41 1500/1500 [=====] - 0s 26us/step - loss: 0.1840
- acc: 0.7360
42 Epoch 10/10
43 1500/1500 [=====] - 0s 24us/step - loss: 0.1753
- acc: 0.7580
44
45 Training Model with 6 neurons in hidden layer
46 Epoch 1/10
47 1500/1500 [=====] - 0s 209us/step - loss: 0.2943
- acc: 0.4933
48 Epoch 2/10
49 1500/1500 [=====] - 0s 25us/step - loss: 0.2667
- acc: 0.5333
50 Epoch 3/10
51 1500/1500 [=====] - 0s 26us/step - loss: 0.2441
- acc: 0.5673
52 Epoch 4/10
53 1500/1500 [=====] - 0s 25us/step - loss: 0.2260
- acc: 0.6000
54 Epoch 5/10
55 1500/1500 [=====] - 0s 25us/step - loss: 0.2110
- acc: 0.6453
56 Epoch 6/10
57 1500/1500 [=====] - 0s 27us/step - loss: 0.1988
- acc: 0.6727
58 Epoch 7/10
59 1500/1500 [=====] - 0s 26us/step - loss: 0.1883
- acc: 0.7087
60 Epoch 8/10
61 1500/1500 [=====] - 0s 27us/step - loss: 0.1791
- acc: 0.7387
62 Epoch 9/10
63 1500/1500 [=====] - 0s 31us/step - loss: 0.1711
- acc: 0.7573
64 Epoch 10/10
65 1500/1500 [=====] - 0s 23us/step - loss: 0.1643
- acc: 0.7760
66
67 Training Model with 8 neurons in hidden layer
```

```
68 Epoch 1/10
69 1500/1500 [=====] - 0s 214us/step - loss: 0.2895
- acc: 0.5273
70 Epoch 2/10
71 1500/1500 [=====] - 0s 25us/step - loss: 0.2614
- acc: 0.5653
72 Epoch 3/10
73 1500/1500 [=====] - 0s 26us/step - loss: 0.2389
- acc: 0.6087
74 Epoch 4/10
75 1500/1500 [=====] - 0s 27us/step - loss: 0.2202
- acc: 0.6460
76 Epoch 5/10
77 1500/1500 [=====] - 0s 27us/step - loss: 0.2051
- acc: 0.6740
78 Epoch 6/10
79 1500/1500 [=====] - 0s 29us/step - loss: 0.1924
- acc: 0.7073
80 Epoch 7/10
81 1500/1500 [=====] - 0s 28us/step - loss: 0.1815
- acc: 0.7347
82 Epoch 8/10
83 1500/1500 [=====] - 0s 25us/step - loss: 0.1725
- acc: 0.7533
84 Epoch 9/10
85 1500/1500 [=====] - 0s 25us/step - loss: 0.1647
- acc: 0.7680
86 Epoch 10/10
87 1500/1500 [=====] - 0s 23us/step - loss: 0.1580
- acc: 0.7747
88
89 Training Model with 10 neurons in hidden layer
90 Epoch 1/10
91 1500/1500 [=====] - 0s 268us/step - loss: 0.2728
- acc: 0.5547
92 Epoch 2/10
93 1500/1500 [=====] - 0s 28us/step - loss: 0.2387
- acc: 0.6080
94 Epoch 3/10
95 1500/1500 [=====] - 0s 26us/step - loss: 0.2110
- acc: 0.6700
96 Epoch 4/10
97 1500/1500 [=====] - 0s 25us/step - loss: 0.1890
- acc: 0.7140
98 Epoch 5/10
```

```
99 1500/1500 [=====] - 0s 29us/step - loss: 0.1722
  - acc: 0.7473
100 Epoch 6/10
101 1500/1500 [=====] - 0s 22us/step - loss: 0.1596
  - acc: 0.7687
102 Epoch 7/10
103 1500/1500 [=====] - 0s 29us/step - loss: 0.1499
  - acc: 0.7880
104 Epoch 8/10
105 1500/1500 [=====] - 0s 22us/step - loss: 0.1425
  - acc: 0.8053
106 Epoch 9/10
107 1500/1500 [=====] - 0s 28us/step - loss: 0.1364
  - acc: 0.8220
108 Epoch 10/10
109 1500/1500 [=====] - 0s 26us/step - loss: 0.1315
  - acc: 0.8240
```

(2) Validate each neural network using the given validation dataset (X val, Y val). The validation accuracy is used to determine how many number of hidden neurons are optimal for this problem.

```
1 eval_results = []
2 for i in range(len(models)):
3     print('Evaluating Model with {} neurons in hidden
layer'.format(hidden_neurons[i]))
4     eval_result = models[i].evaluate(x=val_x, y=val_y, batch_size=50)
5     print('Mean Squared Error - {:.4f}'.format(eval_result[0]))
6     print('Accuracy - {:.4f}'.format(eval_result[1]))
7     eval_results.append(eval_result)
8     print()
```

```
1 Evaluating Model with 2 neurons in hidden layer
2 250/250 [=====] - 0s 360us/step
3 Mean Squared Error - 0.1976
4 Accuracy - 0.7400
5
6 Evaluating Model with 4 neurons in hidden layer
7 250/250 [=====] - 0s 346us/step
8 Mean Squared Error - 0.1650
```

```

9 Accuracy - 0.7600
10
11 Evaluating Model with 6 neurons in hidden layer
12 250/250 [=====] - 0s 372us/step
13 Mean Squared Error - 0.1585
14 Accuracy - 0.7920
15
16 Evaluating Model with 8 neurons in hidden layer
17 250/250 [=====] - 0s 341us/step
18 Mean Squared Error - 0.1595
19 Accuracy - 0.7440
20
21 Evaluating Model with 10 neurons in hidden layer
22 250/250 [=====] - 0s 318us/step
23 Mean Squared Error - 0.1238
24 Accuracy - 0.8280

```

(c) Plot a figure, where the horizontal x-axis is the number of hidden neurons, and the vertical y-axis is the accuracy. Please plot both training and validation accuracy in your figure. (Note that the exact accuracy could be slightly different according to your working environments, however you can analyze the trend.

```

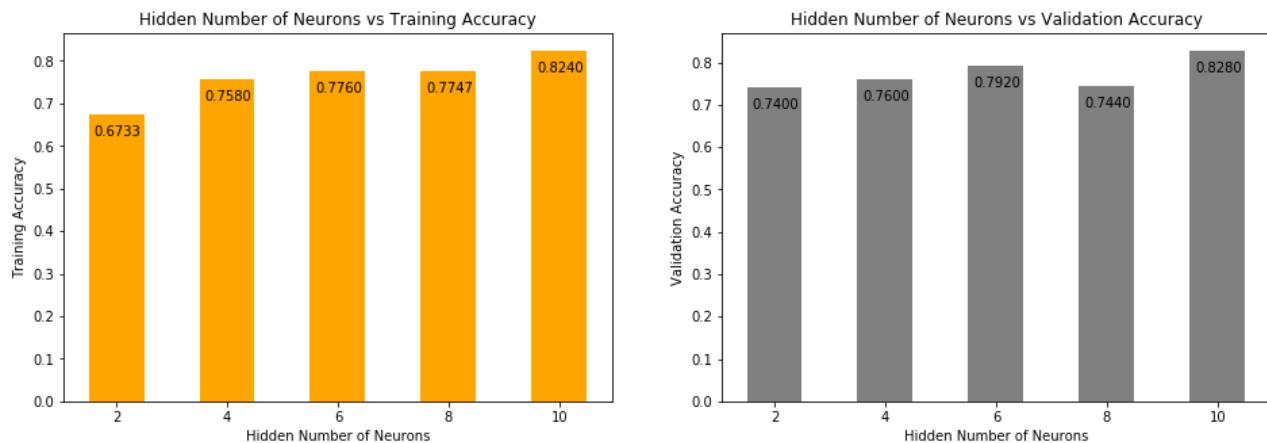
1 fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(16,5))
2 ax[0].set_title('Hidden Number of Neurons vs Training Accuracy')
3 ax[0].set_ylabel('Training Accuracy')
4 ax[0].set_xlabel('Hidden Number of Neurons')
5 ax[0].bar(hidden_neurons,
6             [histories[i].history['acc'][-1] for i in
7              range(len(hidden_neurons))],
8             width=1,
9             color='orange')
10 for i in range(len(hidden_neurons)):
11     ax[0].annotate('{:.4f}'.format(histories[i].history['acc'][-1]),
12                   (hidden_neurons[i]-0.4, histories[i].history['acc']
13 [-1]-0.05))
14
15 ax[1].set_title('Hidden Number of Neurons vs Validation Accuracy')

```

```

14 ax[1].set_ylabel('Validation Accuracy')
15 ax[1].set_xlabel('Hidden Number of Neurons')
16 ax[1].bar(hidden_neurons,
17             [eval_results[i][1] for i in range(len(hidden_neurons))],
18             width=1,
19             color='gray')
20 for i in range(len(hidden_neurons)):
21     ax[1].annotate('{:.4f}'.format(eval_results[i][1]),
22                   (hidden_neurons[i]-0.4, eval_results[i][1]-0.05))

```



(d) Provide a simple analysis about your results and choose the optimal number of hidden neuron from the analysis.

```
1 | model = models[4]
```

Based on the above plots, we can clearly see that we get the best validation accuracy when number of neurons in hidden layer is 10. Thus we select the model with 10 neurons in hidden layers.

(e) Report the test accuracy using the given test dataset (X test, Y test) on the neural network with the optimal number of hidden neurons.

```

1 print('Testing Model with 10 neurons in hidden layer')
2 eval_result = model.evaluate(x=test_x, y=test_y, batch_size=50)
3 print('Mean Squared Error - {:.4f}'.format(eval_result[0]))
4 print('Accuracy - {:.4f}'.format(eval_result[1]))
5 print()

```

```
1 Testing Model with 10 neurons in hidden layer
2 250/250 [=====] - 0s 19us/step
3 Mean Squared Error - 0.1594
4 Accuracy - 0.7880
```

Test Accuracy is 0.7880 with 10 neurons in Hidden Layer.

Q5 SVM Theory

(A)

Weight Vector

Given Datapoints:

X ₁	X ₂	Y (Class Label)
1	4	1
3	2	1
5	4	2
5	6	2

(I)

Class 1 (label 1)

Positive Hyperplane : $WX_1 + W_0 = 1$

Class 2 (label -1)

Negative Hyperplane : $WX_1 + W_0 = -1$

Let's say Weight vector $w = [w_1 \ w_2]$

As per given hint that, $w_1 = w_2$

Considering support vectors to be (3,2), (1,4) and (5,4)

For support vector (3,2) -> Class 1 (label 1),

$$[w_1 \ w_2] \begin{bmatrix} 3 \\ 2 \end{bmatrix} + W_0 = 1 \quad \rightarrow \quad 3w_1 + 2w_2 + W_0 = 1 \quad \dots \dots \dots \text{(i)}$$

For support vector (1,4) -> Class 1 (label 1),

$$[w_1 \ w_2] \begin{bmatrix} 1 \\ 4 \end{bmatrix} + W_0 = 1 \quad \rightarrow \quad w_1 + 4w_2 + W_0 = 1 \quad \dots \dots \dots \text{(ii)}$$

For support vector (5,4) -> Class 2 (label -1),

$$[w_1 \ w_2] \begin{bmatrix} 5 \\ 4 \end{bmatrix} + W_0 = -1 \quad \rightarrow \quad 5w_1 + 4w_2 + W_0 = -1 \quad \dots \dots \dots \text{(iii)}$$

$$\text{(iii)} - \text{(ii)}$$

$$4w_1 = -2$$

$w_1 = -0.5 \dots \text{(iv)}$

Substituting value of w_1 into (i) and (ii)

$-3.5 + 2w_2 + W_0 = 1 \dots \text{(v)}$

$-0.5 + 4w_2 + W_0 = 1 \dots \text{(vi)}$

$(\text{vi}) - (\text{v})$

$1 + 2w_2 = 0$

$w_2 = -0.5 \dots \text{(vii)}$

substituting values of w_1 and w_2 into (i)

$3(-0.5) + 2(-0.5) + w_0 = 1$

$w_0 = 1 + 1.5 + 1$

$w_0 = 3.5$

Therefore, weighted vector, $W = [-0.5 \ -0.5]$

Bias $W_0 = 3.5$

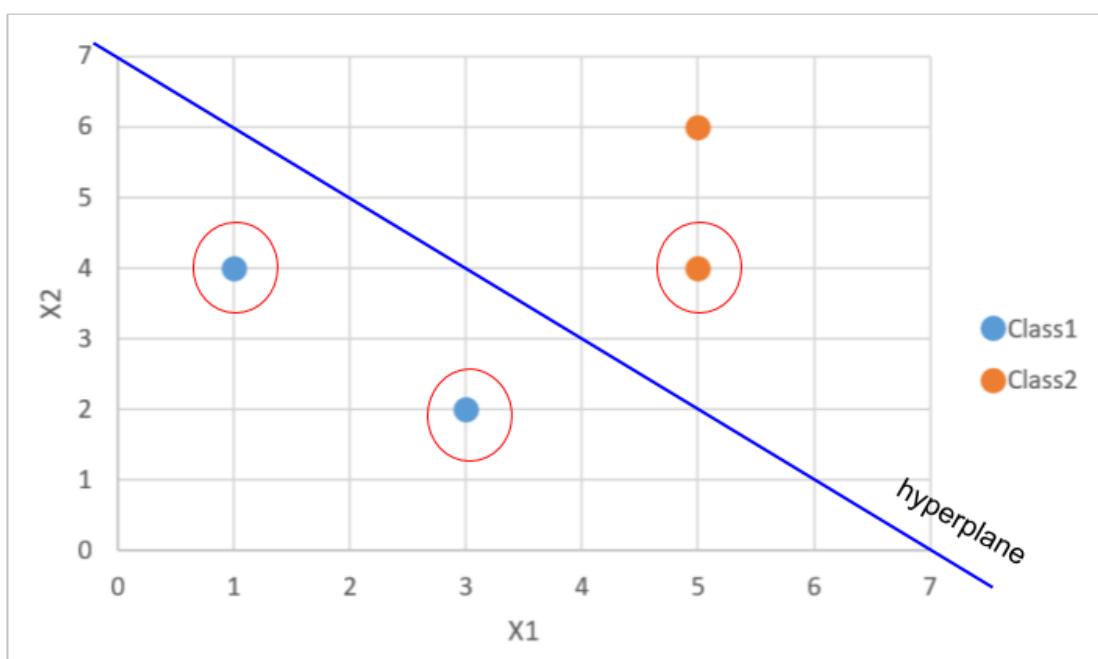
(ii) Support Vectors and Decision Boundary

Decision Boundary : $WX_1 + W_0 = 0$

$[-0.5 \ -0.5] \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + W_0 = 0$

$-0.5X_1 - 0.5X_2 + 3.5 = 0$

$X_2 = -X_1 + 7$



Q5

(B)

Data ID	x_1	x_2	y
X^1	0	2	-1
X^2	2	0	-1
X^3	0	0	1
X^4	2	2	1

(i) Kernel Function

$$K(X^i, X^j) = (1 + X^i \cdot X^j)^2 \dots (1)$$

$$\text{We know } X^i \cdot X^j = [X_1^i, X_2^i] \cdot [X_1^j, X_2^j] = X_1^i \cdot X_1^j + X_2^i \cdot X_2^j$$

$$(1 + X^i \cdot X^j)^2$$

$$= 1 + (X^i \cdot X^j)^2 + 2 \cdot (X^i \cdot X^j)$$

$$= 1 + (X_1^i \cdot X_1^j + X_2^i \cdot X_2^j)^2 + 2 \cdot (X_1^i \cdot X_1^j + X_2^i \cdot X_2^j)$$

$$= 1 + (X_1^i \cdot X_1^j)^2 + (X_2^i \cdot X_2^j)^2 + 2 \cdot (X_1^i \cdot X_1^j \cdot X_2^i \cdot X_2^j) + 2 \cdot (X_1^i \cdot X_1^j) + 2 \cdot (X_2^i \cdot X_2^j) \dots (2)$$

$$\text{We know } X^i \cdot X^j = \phi(X^i) \cdot \phi(X^j) \dots (3)$$

From (1), (2) and (3),

$$\phi(X^i) = 1 + (X_1^i)^2 + (X_2^i)^2 + \sqrt{2} \cdot (X_1^i \cdot X_2^i) + \sqrt{2} \cdot (X_1^i) + \sqrt{2} \cdot (X_2^i)$$

$$\phi(X^j) = 1 + (X_1^j)^2 + (X_2^j)^2 + \sqrt{2} \cdot (X_1^j \cdot X_2^j) + \sqrt{2} \cdot (X_1^j) + \sqrt{2} \cdot (X_2^j)$$

$$\phi(v) = < 1, v_1^2, v_2^2, \sqrt{2}v_1v_2, \sqrt{2}v_1, \sqrt{2}v_2 >$$

(ii) Transformed Space

Data ID	1	x_1^2	x_2^2	$\sqrt{2}x_1x_2$	$\sqrt{2}x_1$	$\sqrt{2}x_2$	y
X^1	1	0	4	0	0	$2\sqrt{2}$	-1
X^2	1	4	0	0	$2\sqrt{2}$	0	-1
X^3	1	0	0	0	0	0	1
X^4	1	4	4	$4\sqrt{2}$	$2\sqrt{2}$	$2\sqrt{2}$	1

Q-5 (b)

(iii)

Data ID	1	2	3	4
(x_1, x_2)	(0, 2)	(2, 0)	(0, 0)	(2, 2)
label y	-1	-1	1	1

→ Dot products of data-points:

Data ID	1	2	3	4
1	4	0	0	4
2	0	4	0	4
3	0	0	0	0
4	4	4	0	8

→ Kernel Values $k(x_i, x_j) = (1 + x_i \cdot x_j)^2$

	1	2	3	4
1	25	1	1	25
2	1	25	1	25
3	1	1	1	1
4	25	25	1	81

$$\begin{aligned}
 L &= \sum x_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j x_i x_j y_i y_j \\
 &= \sum x_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j \cdot K(x_i, x_j) y_i y_j \\
 &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\
 &\quad - \frac{1}{2} \left[\alpha_1^2 \cdot 25 \cdot 1 + \alpha_1 \cdot \alpha_2 - \alpha_1 \alpha_3 - 25 \alpha_1 \alpha_4 \right. \\
 &\quad + \alpha_1 \cdot \alpha_2 + 25 \alpha_2^2 - \alpha_2 \cdot \alpha_3 - 25 \alpha_2 \alpha_4 \\
 &\quad - \alpha_3 \alpha_1 - \alpha_2 \alpha_3 + \alpha_3^2 + \alpha_3 \alpha_4 \\
 &\quad \left. - 25 \alpha_1 \alpha_4 - 25 \alpha_2 \alpha_4 + \alpha_3 \alpha_4 + 81 \alpha_4^2 \right]
 \end{aligned}$$

$$\begin{aligned}
 L &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\
 &\quad - \frac{25}{2} \alpha_1^2 - \frac{25}{2} \alpha_2^2 - \frac{\alpha_3^2}{2} - \frac{81}{2} \alpha_4^2 \\
 &\quad - \alpha_1 \alpha_2 + \alpha_1 \alpha_3 + 25 \alpha_1 \alpha_4 + \alpha_2 \alpha_3 \\
 &\quad + 25 \alpha_2 \alpha_4 - \alpha_3 \alpha_4
 \end{aligned}$$

$$\therefore \frac{dL}{d\alpha_1} = 1 - 25\alpha_1 - \alpha_2 + \alpha_3 + 25\alpha_4 = 0 \quad - \textcircled{1} \quad \textcircled{3}$$

$$\therefore \frac{dL}{d\alpha_2} = 1 - 25\alpha_2 - \alpha_1 + \alpha_3 + 25\alpha_4 = 0 \quad - \textcircled{2}$$

$$\therefore \frac{dL}{d\alpha_3} = 1 - \alpha_3 + \alpha_1 + \alpha_2 - \alpha_4 = 0 \quad - \textcircled{3}$$

$$\therefore \frac{dL}{d\alpha_4} = 1 - 8\alpha_4 + 25\alpha_1 + 25\alpha_2 - \alpha_3 = 0 \quad - \textcircled{4}$$

$$\textcircled{1} - \textcircled{2}$$

$$-24\alpha_1 + 24\alpha_2 = 0 \\ \therefore \boxed{\alpha_1 = \alpha_2} \quad - \textcircled{5}$$

$$\textcircled{2} + \textcircled{4}$$

$$2 + 24\alpha_1 - 56\alpha_4 = 0$$

$$1 + 12\alpha_1 - 28\alpha_4 = 0$$

$$\therefore \boxed{12\alpha_1 = 28\alpha_4 - 1} \quad - \textcircled{6}$$

$$\textcircled{2} + \textcircled{3}$$

$$2 - 24\alpha_2 + 4\alpha_4 = 0$$

$$1 - 12\alpha_2 + 12\alpha_4 = 0 \\ \therefore \boxed{12\alpha_4 = 12\alpha_2 - 1} \quad - \textcircled{7}$$

$\textcircled{6}, \textcircled{7}$ and $\textcircled{5}$

$$12\alpha_1 = 28\alpha_4 - 1 \quad - \textcircled{6}$$

$$12\alpha_4 = 12\alpha_1 - 1 \quad - \textcircled{7}$$

Substituting value of $\textcircled{6}$ into $\textcircled{7}$

$$12\alpha_4 = (28\alpha_4 - 1) - 1$$

$$\therefore 16\alpha_4 = 2$$

$$\therefore \boxed{\alpha_4 = \frac{1}{8}} \quad - \textcircled{8}$$

Substituting value of α_4 in $\textcircled{7}$.

$$12\left(\frac{1}{8}\right) = 12\alpha_1 - 1$$

$$\therefore \frac{3}{2} = 12\alpha_1 - 1$$

$$\therefore 12\alpha_1 = \frac{3}{2} + 1 = \frac{5}{2}$$

$$\therefore \boxed{\alpha_1 = \frac{5}{24} = \alpha_2} \quad - \textcircled{9}$$

Substituting values $\alpha_1, \alpha_2, \alpha_4$ in $\textcircled{3}$.

$$1 + \frac{5}{24} + \frac{5}{24} - \frac{1}{8} = \alpha_3$$

$$\therefore \alpha_3 = \frac{24 + 10 - 3}{24} = \frac{31}{24}$$

$$\therefore \boxed{\alpha_3 = \frac{31}{24}}$$

(4)

For x^i datapoint, (support vector).

$$w \cdot x_1 + w_0 = y_1$$

$$(w \cdot \phi(x_1)) + w_0 = y_1$$

$$\therefore \sum_{i=1+0}^4 \alpha_i y_i \phi(x_i) \cdot \phi(x_1) + w_0 = y_1$$

$$\therefore \alpha_1 y_1 k(x_1, x_1) + \alpha_2 y_2 k(x_2, x_1) + \alpha_3 y_3 k(x_3, x_1) \\ + \alpha_4 y_4 k(x_4, x_1) + w_0 = y_1$$

$$\therefore \left(\frac{5}{24}\right)(-1)(25) + \frac{5}{24}(-1)(1) + \frac{31}{24}(1)(1) + \frac{1}{8}(1)(25) \\ + w_0 = -1$$

$$\therefore \frac{-125 - 5 + 31 + 75}{24} + w_0 = -1$$

$$\therefore \frac{-24}{24} + w_0 = -1$$

$$\therefore \boxed{w_0 = 0}$$

Decision Boundary

$$W \cdot X + W_0 = 0$$

$$\sum_{i=1}^4 \alpha_i \cdot y_i \cdot \phi(X^i) \cdot \phi(X) + W_0 = 0$$

$$\alpha_1 \cdot y_1 \cdot \phi(<1, 0, 4, 0, 0, 2\sqrt{2}>) \cdot \phi(<1, x_1^2, x_2^2, \sqrt{2} \cdot x_1 \cdot x_2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2>)$$

$$+ \alpha_2 \cdot y_2 \cdot \phi(<1, 4, 0, 0, 2\sqrt{2}, 0>) \cdot \phi(<1, x_1^2, x_2^2, \sqrt{2} \cdot x_1 \cdot x_2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2>)$$

$$+ \alpha_3 \cdot y_3 \cdot \phi(<1, 0, 0, 0, 0, 0>) \cdot \phi(<1, x_1^2, x_2^2, \sqrt{2} \cdot x_1 \cdot x_2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2>)$$

$$+ \alpha_4 \cdot y_4 \cdot \phi(<1, 4, 4, 4\sqrt{2}, 2\sqrt{2}, 2\sqrt{2}>) \cdot \phi(<1, x_1^2, x_2^2, \sqrt{2} \cdot x_1 \cdot x_2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2>) + W_0 = 0$$

$$1 - \frac{1}{3}x_1^2 - \frac{1}{3}x_2^2 + 8 \cdot x_1 \cdot x_2 + 8 \cdot x_1 + 8 \cdot x_2 = 0$$

$x_1^2 + x_2^2 + 24 \cdot x_1 \cdot x_2 + 24 \cdot x_1 + 24 \cdot x_2 - 3 = 0$ is the decision boundary.