

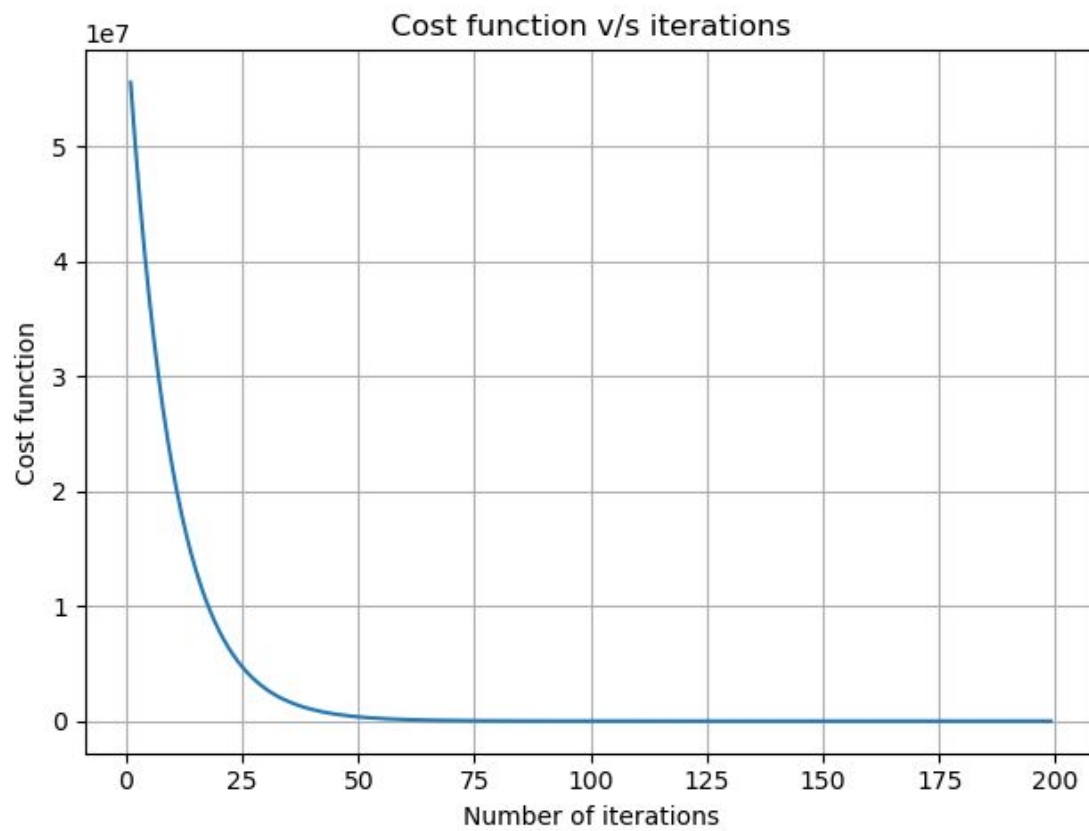
NEURAL NETWORK AND FUZZY LOGIC

ASSIGNMENT-1

GRANDHI SNIGDHA
2015A8PS0495H

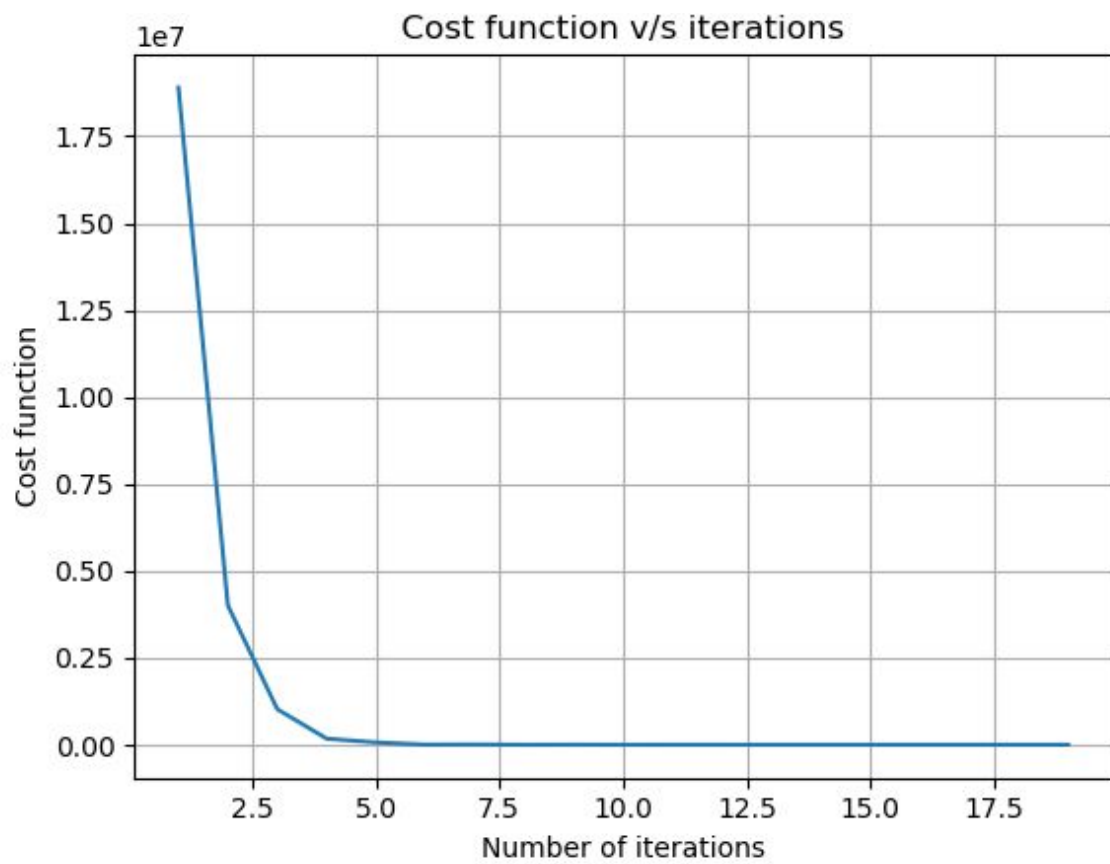
PROBLEM 1

```
1 # linear regression algorithm
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import xlrd
6 from pandas import DataFrame
7
8 data = pd.read_excel('data.xlsx')
9 # print(data)
10 var1 = data['row1']
11 var2 = data['row2']
12 y = data['row3']
13 x = []
14 for i in data.index:
15     temp = []
16     temp.append(1);
17     temp.append(var1[i])
18     temp.append(var2[i])
19     x.append(temp)
20 # print(x)
21 alpha=0.000000001; #learning rate
22 cost = []
23 w = [1,1,1]
24 w = np.array(w)
25 y = np.array(y)
26 for epoch in range(100):
27     for j in range(len(x[0])):
28         # print("me");
29         sumx = 0
30         tempcost = 0
31         for i in range(len(x)):
32             hx = w[0] + w[1]*x[i][1] + w[2]*x[i][2]
33             hx = hx - y[i]
34             tempcost += hx**2
35             hx = hx*x[i][j]
36             sumx += hx
37             # print(hx)
38         # print(hx)
39         w[j] -= alpha*sumx
```



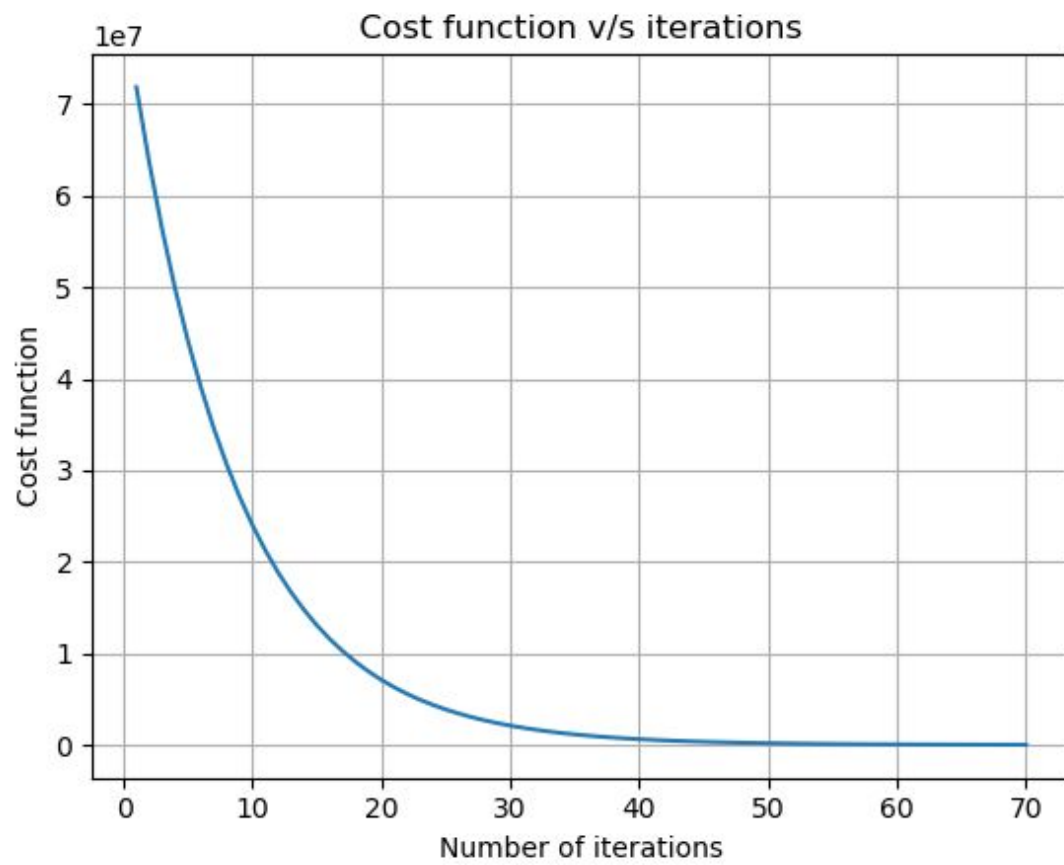
PROBLEM 2

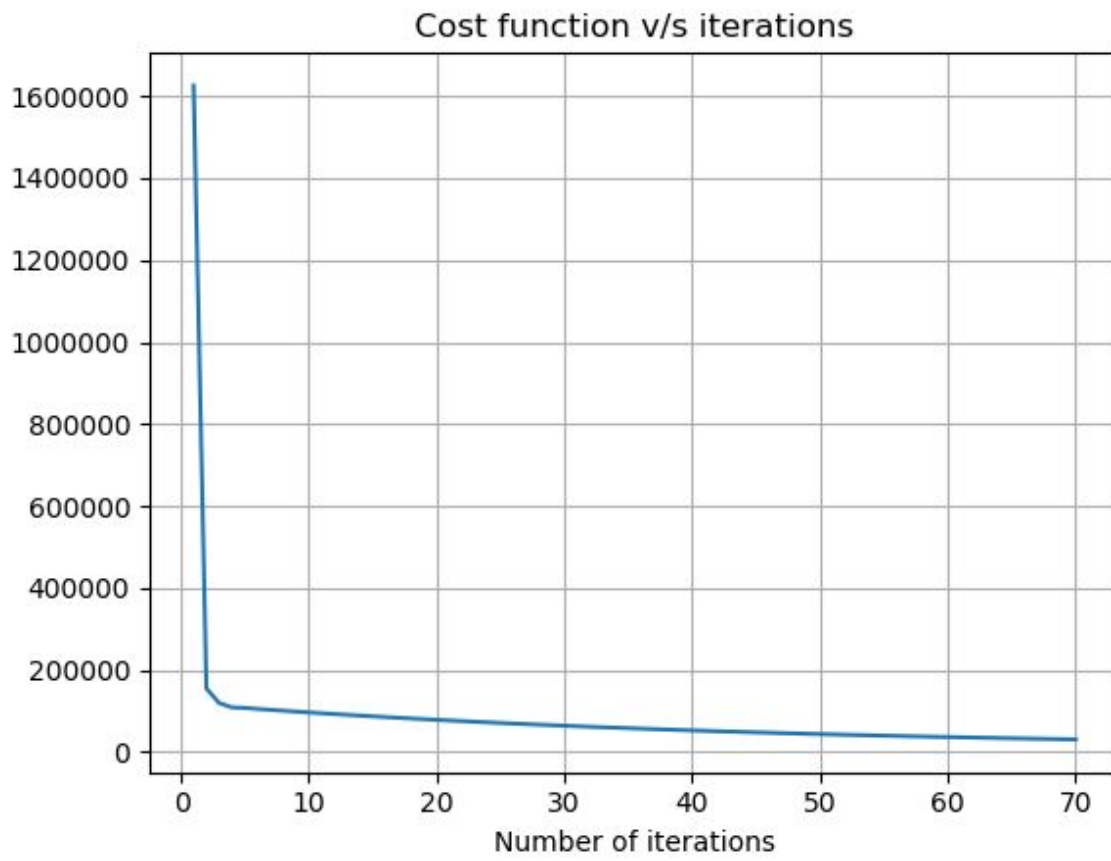
```
1 # linear regression algorithm with stochastic gradient descent
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import xlrd
6 from pandas import DataFrame
7
8 data = pd.read_excel('data.xlsx')
9 # print(data)
10 var1 = data['row1']
11 var2 = data['row2']
12 y = data['row3']
13 x = []
14 for i in data.index:
15     temp = []
16     temp.append(1);
17     temp.append(var1[i])
18     temp.append(var2[i])
19     x.append(temp)
20 # print(x)
21 alpha=0.000000001; #learning rate
22 w = [1,1,1]
23 for epoch in range(100):
24     for j in range(len(x[0])):
25         # print("me");
26         sumx = 0
27         for i in range(len(x)):
28             hx = w[0] + w[1]*x[i][1] + w[2]*x[i][2]
29             hx = hx - y[i]
30             hx = hx*x[i][j]
31             sumx += hx
32             # print(hx)
33         # print(hx)
34         w[j] -= alpha*sumx
35     print(w)
36     print("\n")
```



PROBLEM 3

```
1 # linear regression algorithm
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import xlrd
6 from pandas import DataFrame
7
8 data = pd.read_excel('data.xlsx')
9 # print(data)
10 var1 = data['row1']
11 var2 = data['row2']
12 y = data['row3']
13 x = []
14 for i in data.index:
15     temp = []
16     temp.append(1);
17     temp.append(var1[i])
18     temp.append(var2[i])
19     x.append(temp)
20 # print(x)
21 alpha=0.000000001; #learning rate
22 lmd = 1
23 w = [1,1,1]
24 for epoch in range(10):
25     for j in range(len(x[0])):
26         # print("me");
27         sumx = 0
28         for i in range(len(x)):
29             hx = w[0] + w[1]*x[i][1] + w[2]*x[i][2]
30             hx = hx - y[i]
31             hx = hx*x[i][j]
32             sumx += hx
33             # print(hx)
34         # print(hx)
35         w[j] = (1-alpha*lmd)*w[j] - alpha*sumx
36     print(w)
37     print("\n")
```





PROBLEM 4

```

1 # vectorised linear regression
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import xlrd
6 from pandas import DataFrame
7 from numpy.linalg import inv
8
9 data = pd.read_excel('data.xlsx')
10 # print(data)
11 var1 = data['row1']
12 var2 = data['row2']
13 y = data['row3']
14 x = []
15 for i in data.index:
16     temp = []
17     temp.append(1);
18     temp.append(var1[i])
19     temp.append(var2[i])
20     x.append(temp)
21 matx = np.array(x)
22 maty = np.array(y)
23 matxt = matx.transpose()
24 wght = np.matmul(matxt, matx)
25 wght = inv(wght)
26 wght = np.matmul(wght, matxt)
27 wght = np.matmul(wght, y)
28 print(wght)

```

PROBLEM 5


```

1 # vectorisation based least angle regression
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import xlrd
6 from pandas import DataFrame
7 from numpy.linalg import inv
8
9 data = pd.read_excel('data.xlsx')
10 # print(data)
11 var1 = data['row1']
12 var2 = data['row2']
13 y = data['row3']
14 x = []
15 for i in data.index:
16     temp = []
17     temp.append(1);
18     temp.append(var1[i])
19     temp.append(var2[i])
20     x.append(temp)
21
22 lamda = 2
23 matx = np.array(x)
24 maty = np.array(y)
25 matxt = matx.transpose()
26
27 wght = np.matmul(matxt, matx)
28 wght = inv(wght)
29 wght = np.matmul(wght, (np.matmul(matxt, maty) - (lamda/2)))
30
31 print(wght)

```

PROBLEM 6

```

1  # clustering with k=2
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import xlrd
6  from pandas import DataFrame
7  from numpy.linalg import inv
8  def dist(a, b):
9      sm=0
10     for i in range(len(a)):
11         sm += (a[i]-b[i])**2
12     return sm
13
14  data = pd.read_excel('data2.xlsx')
15  # print(data)
16  var1 = data['row1']
17  var2 = data['row2']
18  var3 = data['row3']
19  var4 = data['row4']
20  v1 = [1, 1, 1, 1]
21  v2 = [2, 10, 5, 2]
22  x = []
23  for i in data.index:
24      temp = []
25      temp.append(var1[i])
26      temp.append(var2[i])
27      temp.append(var3[i])
28      temp.append(var4[i])
29      x.append(temp)
30  for epoch in range(10):
31      cls1 = []
32      cls2 = []
33      for i in range(len(x)):
34          if dist(x[i], v1)<dist(x[i], v2):
35              cls1.append(x[i])
36          else:
37              cls2.append(x[i])
38      # find new centre
39      v1 = [0, 0, 0, 0]
40      for i in range(len(cls1)):
41          for j in range(len(cls1[0])):
42              v1[j] += cls1[i][j]
43      v1 = np.array(v1)
44      if len(cls1)!= 0:
45          v1 = v1/len(cls1)
46
47      v2 = [0, 0, 0, 0]
48      for i in range(len(cls2)):
49          for j in range(len(cls2[0])):
50              v2[j] += cls2[i][j]
51      v2 = np.array(v2)
52      if len(cls2)!= 0:
53          v2 = v2/len(cls2)

```

PROBLEM 7

```

1  # logistic regression
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import xlrd
6  from pandas import DataFrame
7  from numpy.linalg import inv
8  import math
9  from sklearn.cross_validation import train_test_split
10
11 class LogisticRegression:
12     def __init__(self, lr=0.01, num_iter=10, fit_intercept=True, verbose=False):
13         self.lr = lr
14         self.num_iter = num_iter
15         self.fit_intercept = fit_intercept
16
17     def __add_intercept(self, X):
18         intercept = np.ones((X.shape[0], 1))
19         return np.concatenate((intercept, X), axis=1)
20
21     def __sigmoid(self, z):
22         return 1 / (1 + np.exp(-z))
23
24     def __loss(self, h, y):
25         return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
26
27     def fit(self, X, y):
28         if self.fit_intercept:
29             X = self.__add_intercept(X)
30
31         # weights initialization
32         self.theta = np.zeros(X.shape[1])
33
34         for i in range(self.num_iter):
35             z = np.dot(X, self.theta)
36             h = self.__sigmoid(z)
37             gradient = np.dot(X.T, (h - y)) #/ y.size
38             self.theta -= self.lr * gradient
39
40             # if(i % 10000 == 0):
41             #     z = np.dot(X, self.theta)
42             #     h = self.__sigmoid(z)
43             #     print('loss: {self.__loss(h, y)} \t')

```

```

42         #     h = self.__sigmoid(z)
43         #     print('loss: {self.__loss(h, y)} \t')
44         print(self.theta)
45
46     def predict_prob(self, X):
47         if self.fit_intercept:
48             X = self.__add_intercept(X)
49
50         return self.__sigmoid(np.dot(X, self.theta))
51
52     def predict(self, X, threshold):
53         return self.predict_prob(X) >= threshold
54
55 def main():
56     data = pd.read_excel('data3.xlsx')
57     var1 = data['row1']
58     var2 = data['row2']
59     var3 = data['row3']
60     var4 = data['row4']
61     var5 = data['row5']
62     x = []
63     y = var5
64     for i in data.index:
65         temp = []
66         # temp.append(1)
67         temp.append(var1[i])
68         temp.append(var2[i])
69         temp.append(var3[i])
70         temp.append(var4[i])
71         x.append(temp)
72     x = np.array(x)
73     y = np.array(y)
74     xtrain, xtest, ytrain, ytest = train_test_split(x, y, train_size=0.6)
75     model = LogisticRegression()
76     model.fit(xtrain, ytrain)
77     preds = model.predict(xtest, 0.5)
78     # print(preds)
79     acc = (preds == ytest).mean()
80     print(acc)
81     # alpha=0.001; #learning rate
82     # w = [0, 0, 0, 0, 0]
83     # xtrain = np.array(xtrain)
84     # ytrain = np.array(ytrain)

```

PROBLEM 8

```
1  # logistic regression
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import xlrd
6  from pandas import DataFrame
7  from numpy.linalg import inv
8  import math
9  from sklearn.cross_validation import train_test_split
10
11  class LogisticRegression:
12      def __init__(self, lr=0.01, num_iter=1000, fit_intercept=False, verbose=False):
13          self.lr = lr
14          self.num_iter = num_iter
15          self.fit_intercept = fit_intercept
16
17      def __add_intercept(self, X):
18          intercept = np.ones((X.shape[0], 1))
19          return np.concatenate((intercept, X), axis=1)
20
21      def __sigmoid(self, z):
22          return 1 / (1 + np.exp(-z))
23
24      def __loss(self, h, y):
25          return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
26
27      def fit(self, X, y):
28          if self.fit_intercept:
29              X = self.__add_intercept(X)
30
31          # weights initialization
32          self.theta = np.zeros(X.shape[1])
33
34          for i in range(self.num_iter):
35              z = np.dot(X, self.theta)
36              h = self.__sigmoid(z)
37              gradient = np.dot(X.T, (h - y)) #/ y.size
38              self.theta -= self.lr * gradient
39
40              # if(i % 10000 == 0):
41              #     z = np.dot(X, self.theta)
42              #     h = self.__sigmoid(z)
43              #     print('loss: {self. loss(h, y)} {i}')
```

```

43         # print('loss: {self.__loss(h, y)} \t')
44     # print(self.theta)
45     return self.theta
46
47     def prob(self, X, w):
48         if self.fit_intercept:
49             X = self.__add_intercept(X)
50
51         return self.__sigmoid(np.dot(X, w))
52
53 def main():
54     data = pd.read_excel('data4.xlsx')
55     var1 = data['row1']
56     var2 = data['row2']
57     var3 = data['row3']
58     var4 = data['row4']
59     var5 = data['row5']
60     x1 = []
61     x2 = []
62     x3 = []
63     xmain = []
64     for i in data.index:
65         temp = []
66         temp.append(var1[i])
67         temp.append(var2[i])
68         temp.append(var3[i])
69         temp.append(var4[i])
70         temp.append(var5[i])
71         xmain.append(temp)
72     xmain = np.array(xmain)
73     xtra, xte, ytr, yte = train_test_split(xmain[:, :4], xmain[:, 4], train_size=0.6)
74     # print(xtr)
75     # print(ytr.T)
76     xtr = []
77     for i in range(len(ytr)):
78         # temp = []
79         # temp.append(xtra[i].tolist())
80         temp = xtra[i].tolist()
81         temp = np.append(temp, ytr[i].tolist()).tolist()
82         xtr.append(temp)
83     # print(xtr)
84     for xx in xtr:
85         if xx[4]==1 :

```



```

97     x3t=x3
98     x = np.concatenate((x1, x2), axis=0)
99     x = [np.append(xx.tolist(),0) for xx in x]
100    x3t = [np.append(xx.tolist(),1) for xx in x3t]
101    x = np.concatenate((x, x3t), axis=0)
102    x = np.array(x)
103    # training
104    model = LogisticRegression()
105    wa = model.fit(x[:, :4], x[:, 4])
106    # x.empty()
107
108    # 1 and 3 vs 2
109    x2t = x2
110    xb = np.concatenate((x1, x3), axis=0)
111    xb = [np.append(xx.tolist(),0) for xx in xb]
112    x2t = [np.append(xx.tolist(),1) for xx in x2t]
113    xb = np.concatenate((xb, x2t), axis=0)
114    xb = np.array(xb)
115    # training
116    wb = model.fit(xb[:, :4], xb[:, 4])
117
118    # 3 and 2 vs 1
119    xc = np.concatenate((x3, x2), axis=0)
120    xc = [np.append(xx.tolist(),0) for xx in xc]
121    x1 = [np.append(xx.tolist(),1) for xx in x1]
122    xc = np.concatenate((xc, x1), axis=0)
123    xc = np.array(xc)
124    # training
125    wc = model.fit(xc[:, :4], xc[:, 4])
126    # computing the accuracy
127    y = []
128    for xx in xte:
129        a = model.prob(xx, wa)
130        b = model.prob(xx, wb)
131        c = model.prob(xx, wc)
132        if a>b and a>c:
133            y.append(3)
134        elif b>a and b>c:
135            y.append(2)
136        else:
137            y.append(1)
138
139    y = np.array(y)

```

PROBLEM 9

```
52
53     return self.__sigmoid(np.dot(X, w))
54
55 def main():
56     data = pd.read_excel('data4.xlsx')
57     var1 = data['row1']
58     var2 = data['row2']
59     var3 = data['row3']
60     var4 = data['row4']
61     var5 = data['row5']
62     xmain = []
63     for i in data.index:
64         temp = []
65         temp.append(var1[i])
66         temp.append(var2[i])
67         temp.append(var3[i])
68         temp.append(var4[i])
69         temp.append(var5[i])
70         xmain.append(temp)
71     xmain = np.array(xmain)
72     # print(xmain)
73     kfold = KFold(5, True, 1)
74     for train, test in kfold.split(xmain):
75         # xtra, xte, ytr, yte = train_test_split(xmain[:, :4], xmain[:, 4], train_size=0.6)
76         # print(xmain[test])
77         # print(ytr.T)
78         xtra = xmain[train][:, :4]
79         ytr = xmain[train][:, 4]
80         xte = xmain[test][:, :4]
81         yte = xmain[test][:, 4]
82         xtr = []
83         for i in range(len(ytr)):
84             # temp = []
85             # temp.append(xtra[i].tolist())
86             temp = xtra[i].tolist()
87             temp = np.append(temp, ytr[i].tolist()).tolist()
88             xtr.append(temp)
89         # print(xtr)
90         x1 = []
91         x2 = []
92         x3 = []
93         for xx in xtr:
94             if xx[4]==1 ;
```



```

100
101 x1 = np.array(x1)
102 x2 = np.array(x2)
103 x3 = np.array(x3)
104
105 # 1 and 2 vs 3
106 x3t=x3
107 x = np.concatenate((x1, x2), axis=0)
108 x = [np.append(xx.tolist(),0) for xx in x]
109 x3t = [np.append(xx.tolist(),1) for xx in x3t]
110 x = np.concatenate((x, x3t), axis=0)
111 x = np.array(x)
112 # training
113 model = LogisticRegression()
114 wa = model.fit(x[:, :4], x[:, 4])
115 # x.empty()
116
117 # 1 and 3 vs 2
118 x2t = x2
119 xb = np.concatenate((x1, x3), axis=0)
120 xb = [np.append(xx.tolist(),0) for xx in xb]
121 x2t = [np.append(xx.tolist(),1) for xx in x2t]
122 xb = np.concatenate((xb, x2t), axis=0)
123 xb = np.array(xb)
124 # training
125 wb = model.fit(xb[:, :4], xb[:, 4])
126
127 # 3 and 2 vs 1
128 xc = np.concatenate((x3, x2), axis=0)
129 xc = [np.append(xx.tolist(),0) for xx in xc]
130 x1 = [np.append(xx.tolist(),1) for xx in x1]
131 xc = np.concatenate((xc, x1), axis=0)
132 xc = np.array(xc)
133 # training
134 wc = model.fit(xc[:, :4], xc[:, 4])
135 # computing the accuracy
136 y = []
137 for xx in xte:
138     a = model.proba(xx, wa)
139     b = model.proba(xx, wb)
140     c = model.proba(xx, wc)
141     if a>b and a>c:
142         y.append(3)

```

PROBLEM 10

```
1  # likelihood ratio test (LRT)
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import xlrd
6  from pandas import DataFrame
7  from numpy.linalg import inv
8  import math
9  from sklearn.cross_validation import train_test_split
10 from numpy.linalg import inv
11 from numpy import cov
12
13 def main():
14     data = pd.read_excel('data3.xlsx')
15     var1 = data['row1']
16     var2 = data['row2']
17     var3 = data['row3']
18     var4 = data['row4']
19     var5 = data['row5']
20     x = []
21     for i in data.index:
22         temp = []
23         # temp.append(1)
24         temp.append(var1[i])
25         temp.append(var2[i])
26         temp.append(var3[i])
27         temp.append(var4[i])
28         temp.append(var5[i])
29         x.append(temp)
30     x = np.array(x)
31     x1 = []
32     x2 = []
33     # print(x)
34     # print('printed x\n')
35     xtrain, xtest, ytrain, ytest = train_test_split(x[:, :4], x[:, 4], train_size=0.6)
36     for i in range(len(ytrain)):
37         if ytrain[i] == 0:
38             temp = xtrain[i].tolist()
39             # temp.append(1)
40             x1.append(temp)
41         else:
42             temp = xtrain[i].tolist()
43             x2.append(temp)
```

```

46     u1 = [0,0,0,0]
47     u1 = np.array(u1)
48     x1 = np.array(x1)
49     for xx in x1:
50         u1 = np.add(u1, xx)
51     u1 = u1/len(x1)
52
53     u2 = [0,0,0,0]
54     u2 = np.array(u2)
55     x2 = np.array(x2)
56     for xx in x2:
57         u2 = np.add(u2, xx)
58     u2 = u2/len(x2)
59     cov1 = cov(x[:, :4]).T)
60     cov1 = inv(cov1)
61     print(cov1)
62     # [[ 11.74272179  -7.24443147  -7.60918297   5.13458222]
63     #    [-7.24443147  11.4468994   7.29671775  -6.39827837]
64     #    [-7.60918297   7.29671775  17.29107951 -33.05113455]
65     #    [ 5.13458222  -6.39827837 -33.05113455  78.26853974]]
66
67     w = np.dot(cov1, (u2 - u1))
68     print(w)
69     # [-0.09231329  0.91722705 -1.33392136  0.56030023]
70
71     b = (1/2)*(np.dot(u2.T, np.dot(cov1, u2)) - np.dot(u1.T, np.dot(cov1, u1)))
72     print(b)
73     # -1.06923186932
74
75     # computing the accuracy
76     y = []
77     for xx in xtest:
78         temp = np.dot(w.T, xx) + b
79         if temp<0:
80             y.append(0)
81         else:
82             y.append(1)
83     y = np.array(y)
84     acc = (y == ytest).mean()
85     print(acc)
86     # 0.65
87
88 if __name__ == "__main__":

```

PROBLEM 11

```
1 # Logistic Regression
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import xlrd
6 from pandas import DataFrame
7 from numpy.linalg import inv
8 import math
9 from sklearn.cross_validation import train_test_split
10
11 class MaximumLikelihood:
12     def __init__(self, lr=0.01, num_iter=1000, fit_intercept=False, verbose=False):
13         self.lr = lr
14         self.num_iter = num_iter
15         self.fit_intercept = fit_intercept
16
17     def __add_intercept(self, X):
18         intercept = np.ones((X.shape[0], 1))
19         return np.concatenate((intercept, X), axis=1)
20
21     def __sigmoid(self, z):
22         return 1 / (1 + np.exp(-z))
23
24     def __loss(self, h, y):
25         return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
26
27     def fit(self, X, y):
28         if self.fit_intercept:
29             X = self.__add_intercept(X)
30
31         # weights initialization
32         self.theta = np.zeros(X.shape[1])
33
34         for i in range(self.num_iter):
35             z = np.dot(X, self.theta)
36             h = self.__sigmoid(z)
37             gradient = np.dot(X.T, (h - y)) #/ y.size
38             self.theta -= self.lr * gradient
39
40             # if(i % 10000 == 0):
41             #     z = np.dot(X, self.theta)
42             #     h = self.__sigmoid(z)
43             #     print('loss: {self.__loss(h, y)} \t')
44         # print(self.theta)
45         return self.theta
46
```

```

53 def main():
54     data = pd.read_excel('data4.xlsx')
55     var1 = data['row1']
56     var2 = data['row2']
57     var3 = data['row3']
58     var4 = data['row4']
59     var5 = data['row5']
60     x1 = []
61     x2 = []
62     x3 = []
63     xmain = []
64     for i in data.index:
65         temp = []
66         temp.append(var1[i])
67         temp.append(var2[i])
68         temp.append(var3[i])
69         temp.append(var4[i])
70         temp.append(var5[i])
71         xmain.append(temp)
72     xmain = np.array(xmain)
73     xtra, xte, ytra, yte = train_test_split(xmain[:, :4], xmain[:, 4], train_size=0.6)
74     # print(xtr)
75     # print(ytr.T)
76     xtr = []
77     for i in range(len(ytr)):
78         temp = xtra[i].tolist()
79         temp = np.append(temp, ytr[i].tolist()).tolist()
80         xtr.append(temp)
81     # print(xtr)
82     for xx in xtr:
83         if xx[4]==1 :
84             x1.append(xx[:4])
85         elif xx[4]==2:
86             x2.append(xx[:4])
87         else :
88             x3.append(xx[:4])
89
90     x1 = np.array(x1)
91     x2 = np.array(x2)
92     x3 = np.array(x3)
93     model = MaximumLikelihood()
94     # training
95     w = model.fit(xc[:, :4], xc[:, 4])
96     # computing the accuracy
97     y = []

```

PROBLEM 12

```
1  # logistic regression
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import xlrd
6  from pandas import DataFrame
7  from numpy.linalg import inv
8  import math
9  from sklearn.cross_validation import train_test_split
10
11 class MaximumLikelihood:
12     def __init__(self, lr=0.01, num_iter=1000, fit_intercept=False, verbose=False):
13         self.lr = lr
14         self.num_iter = num_iter
15         self.fit_intercept = fit_intercept
16
17     def __add_intercept(self, X):
18         intercept = np.ones((X.shape[0], 1))
19         return np.concatenate((intercept, X), axis=1)
20
21     def __sigmoid(self, z):
22         return 1 / (1 + np.exp(-z))
23
24     def __loss(self, h, y):
25         return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
26
27     def fit(self, X, y):
28         if self.fit_intercept:
29             X = self.__add_intercept(X)
30
31         # weights initialization
32         self.theta = np.zeros(X.shape[1])
33
34         for i in range(self.num_iter):
35             z = np.dot(X, self.theta)
36             h = self.__sigmoid(z)
37             gradient = np.dot(X.T, (h - y)) #/ y.size
38             self.theta -= self.lr * gradient
39
40             # if(i % 10000 == 0):
41             #     z = np.dot(X, self.theta)
42             #     h = self.__sigmoid(z)
43             #     print('loss: {self.__loss(h, y)} {t}')
```



```

43         # print(loss: {self.__loss(n, y) / n})
44         # print(self.theta)
45         return self.theta
46
47     def prob(self, X, w):
48         if self.fit_intercept:
49             X = self.__add_intercept(X)
50
51         return self.__sigmoid(np.dot(X, w))
52
53 def main():
54     data = pd.read_excel('data4.xlsx')
55     var1 = data['row1']
56     var2 = data['row2']
57     var3 = data['row3']
58     var4 = data['row4']
59     var5 = data['row5']
60     x1 = []
61     x2 = []
62     x3 = []
63     xmain = []
64     for i in data.index:
65         temp = []
66         temp.append(var1[i])
67         temp.append(var2[i])
68         temp.append(var3[i])
69         temp.append(var4[i])
70         temp.append(var5[i])
71         xmain.append(temp)
72     xmain = np.array(xmain)
73     xtra, xte, ytra, yte = train_test_split(xmain[:, :4], xmain[:, 4], train_size=0.6)
74     # print(xtr)
75     # print(ytr.T)
76     xtr = []
77     for i in range(len(ytr)):
78         temp = xtra[i].tolist()
79         temp = np.append(temp, ytr[i].tolist()).tolist()
80         xtr.append(temp)
81     # print(xtr)
82     for xx in xtr:
83         if xx[4]==1 :
84             x1.append(xx[:4])
85         elif xx[4]==2:

```

```

73 xtra, xte, ytra, yte = train_test_split(xmain[:, :4], xmain[:, 4], train_size=0.6)
74 # print(xtr)
75 # print(ytr.T)
76 xtr = []
77 for i in range(len(ytr)):
78     temp = xtra[i].tolist()
79     temp = np.append(temp, ytr[i].tolist()).tolist()
80     xtr.append(temp)
81 # print(xtr)
82 for xx in xtr:
83     if xx[4]==1 :
84         x1.append(xx[:4])
85     elif xx[4]==2:
86         x2.append(xx[:4])
87     else :
88         x3.append(xx[:4])
89
90 x1 = np.array(x1)
91 x2 = np.array(x2)
92 x3 = np.array(x3)
93 model = MaximumLikelihood()
94 # training
95 w = model.fit(xc[:, :4], xc[:, 4])
96 # computing the accuracy
97 y = []
98 for xx in xte:
99     a = model.prob(xx, wa)
100     b = model.prob(xx, wb)
101     c = model.prob(xx, wc)
102     if a>b and a>c:
103         y.append(3)
104     elif b>a and b>c:
105         y.append(2)
106     else:
107         y.append(1)
108
109 y = np.array(y)
110
111 acc = (yte == y).mean()
112 print(acc)
113
114 if __name__ == "__main__":
115     main()

```