Established in collaboration with MIT

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

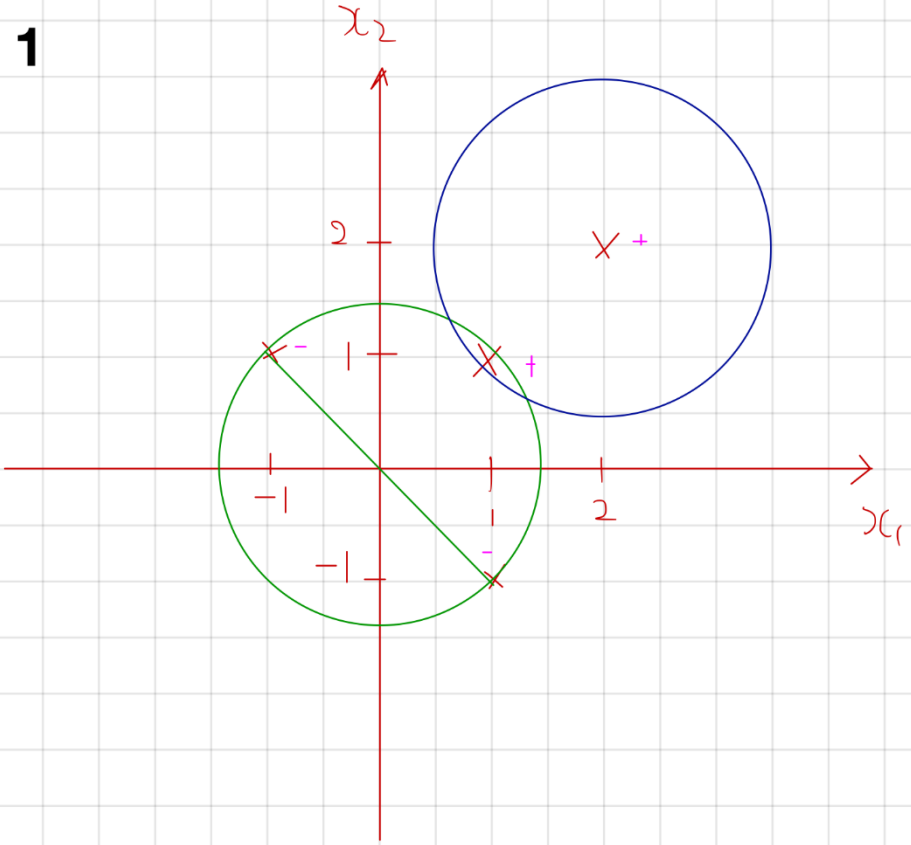
01.112 Machine Learning

HW 1

Barry Tee Wei Cong

1001549

Qn 1



Qn 1a

I. No classifier exists. With origin as the centre of the circle drawn in green coupled with radius r = (1^2 + 1^2)^0.5, it will encompass not only the two negative examples but also encompass the positive example (1,1). As such, even by adjusting the radius r, there is no classifier to separate the positive example (1,1) from the two negative examples (-1,1) and (1,-1).

II. Classifier exists. An example can be origin of the circle be (2,2) and the radius r can be more than 2^0.5 < r < 10^0.5. This will then only encompass the positive examples (1,1) and (2,2) within the circle and allow the negative examples (-1,1) and (1,-1) to be outside the circle if the radius range is met.

III. No classifier exists. FOr the line to pass through the origin, the line will have the negative examples (-1,1) and (1,-1) on the line that passes through origin with gradient of -1. Hence, the negative examples cannot be separated from the positive examples by use of a line.

Qn 1b

Only Qn 1a III is a linear classifier if it could separate the positive and negative examples and not mix both categories of examples. Qn 1a I and II are definitely not as they are in circular and not linear form, thus not linear classifiers.

Qn 2

Instructions on How to Run the Code for Qn 2a and 2b

1) Download Anaconda. Highly recommended to download Anaconda's latest Python 3 version (currently Python 3.6).
2) Install the version of Anaconda which you downloaded, following the instructions on the download page. Please ensure that Anaconda installer is installed via ***Run By Administrator***. This is to prevent an error in installing Jupyter Notebook and access the local directory.
3) Unzip the Folder named HW1_1001549_BarryTeeWeiCong
4) Open Jupyter Notebook and access the folder to open `Homework1.2final.ipynb`.
5) Press Play Button once to obtain the answer for Qn2a.
6) Press Play Button again to obtain the answer for Qn2b.
7) Feel free to call Barry Tee at +65 81393748 if there are any errors running the code.
8) Below is my code with the answers to each question

Qn 2a

```
from csv import reader
from random import randrange

def load_csv(filename):
    dataset = list()
    numrow=0
    with open(filename,'r') as file:
        csv_reader = reader(file)
        for row in csv_reader:
            if not row:
                continue
            dataset.append(row)
            numrow += 1
        print("number of rows in csv file:", numrow)
    return dataset

#Convert string column 0 and 1 to float
def str_column_to_float(dataset, column):
    for row in dataset:
        row[column] = float(row[column].strip())
        #remove white space and convert to float
    print ('str_column_to_float is complete')

#Convert string column 2 to integer
def str_column_to_int(dataset,column):
    class_values = [row[column] for row in dataset]
#    print(class_values)
#    Print all the values
    unique = set(class_values)
#    print (unique)
#    Finding only unique values in set of class_values
    lookup = dict()
    for i , value in enumerate(unique):
        print (i , value)
        lookup[value] = i
    for row in dataset:
        row[column] = lookup[row[column]]
    print ('str_column_to_int is complete')
    return lookup

# Make a prediction with weights
def predict(row, weights):
    activation = weights[0]
    for i in range(len(row)-1):
        #print ('i =' , i)
        #print ('activation before =' , activation)
```

```
        activation += weights[i + 1] * row[i]
        #print ('weights[i+1] is ' , weights[i+1] ,'and row[i] is' , row[i])
        #if activation >= 0.0:
            #print ('activation after =' , activation)
    return 1.0 if activation >= 0.0 else 0.0

# Estimate Perceptron weights using stochastic gradient descent
def train_weights(train, l_rate, n_epoch):
    weights = [0.0 for i in range(len(train[0]))] #create a list with 3 items
    #print(len(train[0]))
    #print(weights)
    for epoch in range(n_epoch): #reiterate from 0 to 4
        sum_error = 0.0
        for row in train: #row from 1st row to 983rd row
            prediction = predict(row, weights)
            error = row[-1] - prediction
            #print ('row[-1] is' , row[-1] , ', prediction is' , prediction , 'and error is' , error)
            sum_error += error**2
            weights[0] = weights[0] + l_rate * error
            #print('')
            #print ('weights[0] is ' , weights[0]) #error for true/false
            for i in range(len(row)-1):
                weights[i + 1] = weights[i + 1] + l_rate * error * row[i] #error for column 0 and 1
                #print('weights[i+1] is ' , weights[i+1])
            #print ('')
        print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))
    return weights

# Train the Perceptron Algorithm on the Train Data Set
filename = 'train_1_5.csv'
dataset = load_csv(filename)
# convert string class to float
for i in range(len(dataset[0])-1):
#    print (dataset[i])
#    print (len(dataset[i]))
#    print (i)
    str_column_to_float(dataset, i)
# convert string class to integers
str_column_to_int(dataset, len(dataset[0])-1)

# Calculate weights
l_rate = 0.1
n_epoch = 5
weights = train_weights(dataset, l_rate, n_epoch)
print('weights = ' , weights)
print('Training Done')
```

Output for 1st 92 lines of Python Code for Qn2a

```
number of rows in csv file: 983
str_column_to_float is complete
str_column_to_float is complete
0 -1.0000000000000000e+00
1 1.0000000000000000e+00
str_column_to_int is complete
>epoch=0, lrate=0.100, error=78.000
>epoch=1, lrate=0.100, error=42.000
>epoch=2, lrate=0.100, error=52.000
>epoch=3, lrate=0.100, error=54.000
>epoch=4, lrate=0.100, error=45.000
weights =  [0.1, -0.20571911389981537, -0.9157699463953628]
Training Done
```

Qn 2b

```
93  # Test the Perceptron algorithm on the Test Data Set
94  filename = 'test_1_5.csv'
95  dataset = load_csv(filename)
96  for i in range(len(dataset[0])-1):
97          str_column_to_float(dataset, i)
98  # convert string class to integers
99  str_column_to_int(dataset, len(dataset[0])-1)
100 # Split a dataset into k folds
101 def cross_validation_split(dataset, n_folds):
102         dataset_split = list()
103         dataset_copy = list(dataset)
104         fold_size = int(len(dataset) / n_folds)
105         for i in range(n_folds):
106                 fold = list()
107                 while len(fold) < fold_size:
108                         index = randrange(len(dataset_copy))
109                         fold.append(dataset_copy.pop(index))
110                 dataset_split.append(fold)
111         return dataset_split
112
113 # Calculate accuracy percentage
114 def accuracy_metric(actual, predicted):
115         correct = 0
116         for i in range(len(actual)):
117                 if actual[i] == predicted[i]:
118                         correct += 1
119         return correct / float(len(actual)) * 100.0
120
121 # Evaluate an algorithm using a cross validation split
122 def evaluate_algorithm(dataset, algorithm, n_folds, *args):
123         folds = cross_validation_split(dataset, n_folds)
124         scores = list()
125         for fold in folds:
126                 train_set = list(folds)
127                 train_set.remove(fold)
128                 train_set = sum(train_set, [])
129                 test_set = list()
130                 for row in fold:
131                         row_copy = list(row)
132                         test_set.append(row_copy)
133                         row_copy[-1] = None
134                 predicted = algorithm(train_set, test_set, *args)
135                 actual = [row[-1] for row in fold]
136                 accuracy = accuracy_metric(actual, predicted)
137                 scores.append(accuracy)
138         return scores
```

```
139
140 # Perceptron Algorithm With Stochastic Gradient Descent
141 def perceptron(train, test, l_rate, n_epoch):
142         predictions = list()
143         weights = train_weights(train, l_rate, n_epoch)
144         for row in test:
145                 prediction = predict(row, weights)
146                 predictions.append(prediction)
147         return(predictions)
148
149 # evaluate algorithm
150 n_folds = 3
151 l_rate = 0.01
152 n_epoch = 10
153 scores = evaluate_algorithm(dataset, perceptron, n_folds, l_rate, n_epoch)
154 print('Scores: %s' % scores)
155 print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
156 print('Testing Done')
```

Output for Next[t] 64 lines of Python Code for Qn2b

```
number of rows in csv file: 1002
str_column_to_float is complete
str_column_to_float is complete
0 -1.0000000000000000e+00
1 1.0000000000000000e+00
str_column_to_int is complete
>epoch=0, lrate=0.010, error=50.000
>epoch=1, lrate=0.010, error=46.000
>epoch=2, lrate=0.010, error=40.000
>epoch=3, lrate=0.010, error=39.000
>epoch=4, lrate=0.010, error=40.000
>epoch=5, lrate=0.010, error=36.000
>epoch=6, lrate=0.010, error=36.000
>epoch=7, lrate=0.010, error=34.000
>epoch=8, lrate=0.010, error=36.000
>epoch=9, lrate=0.010, error=36.000
>epoch=0, lrate=0.010, error=68.000
>epoch=1, lrate=0.010, error=48.000
>epoch=2, lrate=0.010, error=48.000
>epoch=3, lrate=0.010, error=39.000
>epoch=4, lrate=0.010, error=34.000
>epoch=5, lrate=0.010, error=42.000
>epoch=6, lrate=0.010, error=42.000
>epoch=7, lrate=0.010, error=38.000
>epoch=8, lrate=0.010, error=38.000
>epoch=9, lrate=0.010, error=42.000
>epoch=0, lrate=0.010, error=68.000
>epoch=1, lrate=0.010, error=43.000
>epoch=2, lrate=0.010, error=40.000
>epoch=3, lrate=0.010, error=38.000
>epoch=4, lrate=0.010, error=37.000
>epoch=5, lrate=0.010, error=36.000
>epoch=6, lrate=0.010, error=30.000
>epoch=7, lrate=0.010, error=36.000
>epoch=8, lrate=0.010, error=36.000
>epoch=9, lrate=0.010, error=32.000
Scores: [96.40718562874252, 96.7065868263473, 95.80838323353294]
Mean Accuracy: 96.307%
Testing Done
```