Exersice Sheet 1 (Solution)

Muhammad Fahad Bin Ashraf (412924), Shabi Turabi, Waleed Basit

May 2

1. Identifying learning problems
   Part (a) and (b)

   - Predictive maintenance:
     Data: large amount of irregular factory data
     Goal: to minimizes the risk of unexpected failures and reduces the amount of unnecessary preventive maintenance activities. humans can't do this task more accuratetely as Machines because its hard to manage billions of data without any automated or smart system.
   - Customer support
     Data: large volume of customer data (quantitative data, historical data)
     Goal: to improve chatbots and conversational interfaces for customer service.
     Human can do this task more better than any smart device because they can think of quick answers which machine cant do.
   - Product recommendation
     Data: purchase history for a customer and a large inventory of products
     Goal: to identify those products in which that customer will be interested and likely to purchase.
     human cant do this task better than intelligent devices because its depend on the historical purchases which need to maintain all records manually.

   AI/ML companies in Kaiserslautern:

   - Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
     The German Research Center for Artificial Intelligence (DFKI) was founded in 1988 as a non-profit public-private partnership. It has research facilities in Kaiserslautern, Saarbrücken and Bremen, a project office in Berlin, a Laboratory in Niedersachsen and a branch office in St. Wendel. In the field of innovative commercial software technology using Artificial Intelligence, DFKI is the leading research center in Germany.
   - Insiders Technologies
     Insiders Technologies supports companies worldwide on their way to the digital transformation with its intelligent software solutions based on Artificial Intelligence. More than 1,500 customers rely on Insiders' innovative solutions for intelligent input management

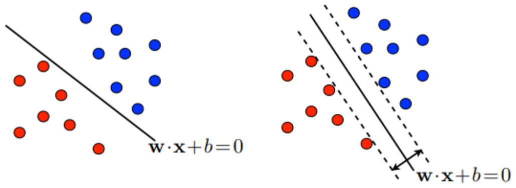
and modern customer communication. Insiders Technologies is your ideal partner for becoming a digital company.

- Digital Devotion
  The DDG – Digital Devotion Group® is dedicated to provide founders, scientists,investors and entrepreneurs with the best ecosystem for developing digital business models with a focus on Cross Reality (AR/VR/MR), Blockchain (BC) and Artificial Intelligence (AI). The ecosystem enables all partners and startups not only to present and test their innovations in e.g. virtual labs, b ut also to access clients, experts and know-how, the network and capital.

Visit to DFKI:

DFKI mission is to propel the utilization of human languages by machines and to make and improve IT-solutions that benefit by language use. They direct propelled research in language innovation and give novel computational methods to processing text, speech and knowledge. They take a stab at a more profound comprehension of human language and thought, concentrating the genuine needs of the end user and the requests of the market. They create novel and upgraded solutions identified with data and knowledge management, content generation, and natural communication.

2. General equation is $w > x + b = 0$



$w \epsilon \mathbb{R}$ d is a non-zero normal vector

b is a scalar intercept

Divides the space in half, i.e. $w > x + b > 0$ and $w > x + b < 0$

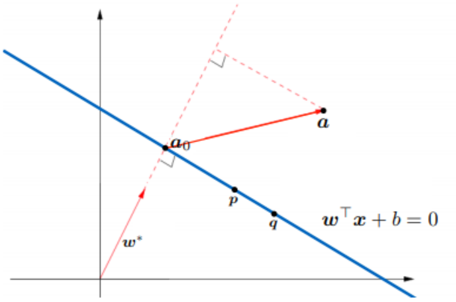


Figure 1: If two points, p and q are both on the line, then w > (p − q) = 0

A hyperplane is a line in 2D and a plane in 3D

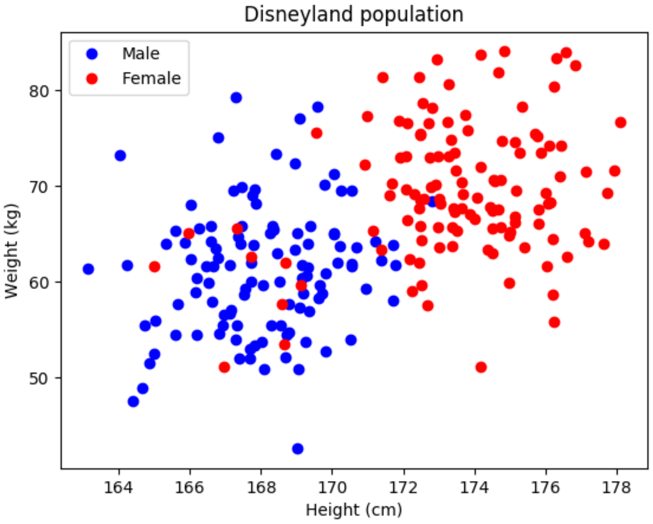
p − q is an arbitrary vector parallel to the line, thus w is orthogonal

$w_* = w \parallel w \parallel$ is the unit normal vector
We want to find distance between a and line in direction of w_*
If we define point a_0 on the line, then this distance corresponds to length of $a - a_0$ in direction of w_* , which equals $w_* > (a - a_0)$
Since $w > a_0 = -b$, the distance equals $\frac{1}{\|w\|}(w^\top a + b)$
Similarly the signed distance from a point x^* to decision boundary (hyperplane) H is

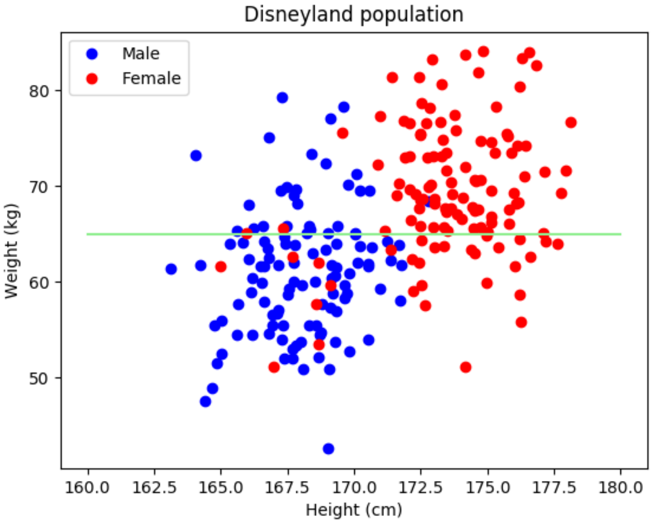
$d(x^*, H) = \frac{w^\top x^* + b}{\|w\|}$

3. Distribution of weight and height among the citizens genders

   (a) Scatter plot

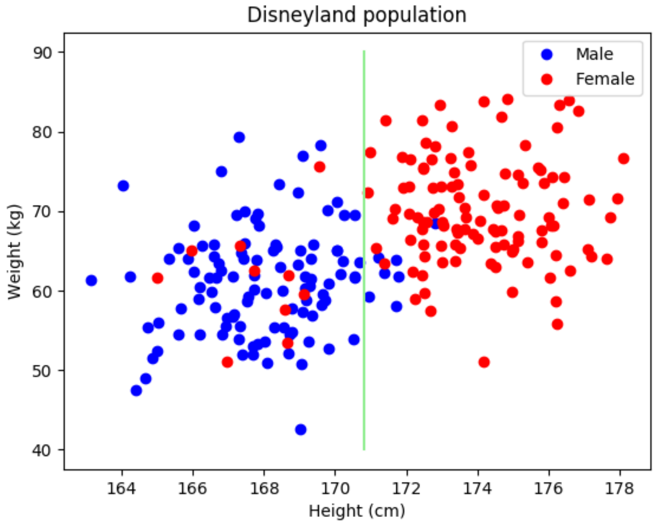


   (b) Scatter plot with horizontal line to best seperate male and female citizens



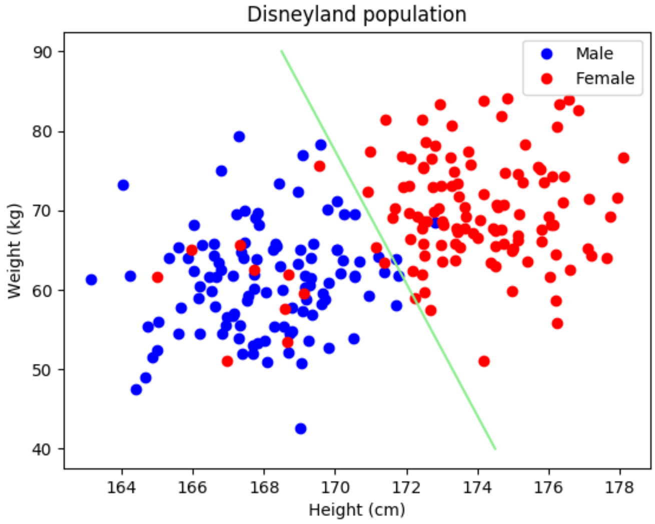
   (c) We would say that he/she is his father, but we cannot guarantee that because we do not know the hight value.

(d) Scatter plot with vertical line to best seperate male and female citizens



(e) We would say that he/she is his sister, because according to the plot, almost all of the citizens above the hight of 173 are females.

(f) Scatter plot with line to best seperate male and female citizens



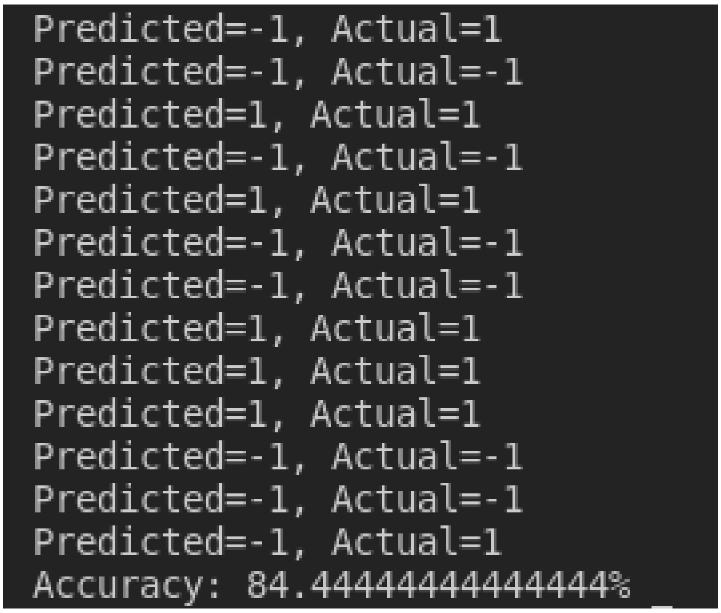
(g) We would classify he/she as a ”female” and no we would not classify differently if we use (b) or (d) lines because all of the citizens in that area are females.

5. k-nearest-neighbor learning algorithm

(a) Algo implementation leaving 'k' as parameter

```python
import csv
import math
import operator
import numpy
import itertools

# function to import data from csv
def import_data(file_name, dataset):
  with open(file_name,'r') as csvfile:
    lines = csv.reader(csvfile, delimiter=',')
    for row in lines:
      dataset.append([float(row[1]), float(row[2]), int(row[3])])

# as all the citizens hight and weight are numeric and have same units we can use euclidean distance
def euclidean_distance(instance1, instance2, length):
  distance = 0
  for x in range(length):
    distance += pow((instance1[x] - instance2[x]), 2)
  return math.sqrt(distance)

def get_neighbors(training_set, test_instance, k):
  distances = []
  length = len(test_instance)-1
  for x in range(len(training_set)):
    dist = euclidean_distance(test_instance, training_set[x], length)
    distances.append((training_set[x], dist))
  # sorting w.r.t distance
  distances.sort(key=operator.itemgetter(1))
  neighbors = []
  for x in range(k):
    neighbors.append(distances[x][0])
  return neighbors

# returns the maximum voted gender in the neighbors
def get_response(neighbors):
  gender_vote = {}
  for x in range(len(neighbors)):
    gender = neighbors[x][-1]
    if gender in gender_vote:
      gender_vote[gender] += 1
    else:
      gender_vote[gender] = 1
  # sorting gender votes on maximum votes
  sorted_votes = sorted(gender_vote.items(), key=operator.itemgetter(1), reverse=True)
  return sorted_votes[0][0]

# calculates the accuracy of predictions with the test set
def get_accuracy(test_set, predictions):
  correct = 0
  for x in range(len(test_set)):
    if test_set[x][-1] == predictions[x]:
      correct += 1
  return (correct/float(len(test_set))) * 100.0

# the main function
def main():

  training_set = []
  test_set = []

  # importing training data set
  import_data('data/DWH_Training.csv', training_set)

  # importing test data set
  import_data('data/DWH_test.csv', test_set)

  predictions=[]

  k = 3

  for x in range(len(test_set)):
    neighbors = get_neighbors(training_set, test_set[x], k)
    result = get_response(neighbors)
    predictions.append(result)
    print('Predicted=' + repr(result) + ', Actual=' + repr(test_set[x][-1]))

  accuracy = get_accuracy(test_set, predictions)

  print('Accuracy: ' + repr(accuracy) + '%')

main()
```

Output:



```
Predicted=-1, Actual=1
Predicted=-1, Actual=-1
Predicted=1, Actual=1
Predicted=-1, Actual=-1
Predicted=1, Actual=1
Predicted=-1, Actual=-1
Predicted=-1, Actual=-1
Predicted=1, Actual=1
Predicted=1, Actual=1
Predicted=1, Actual=1
Predicted=-1, Actual=-1
Predicted=-1, Actual=-1
Predicted=-1, Actual=1
Accuracy: 84.44444444444444%
```

(b) k-fold cross validation algo show best avarage accuracy for k=5

```
import csv
import math
import operator
import numpy
import itertools

# function to import data from csv
def import_data(file_name, dataset):
  with open(file_name,'r') as csvfile:
    lines = csv.reader(csvfile, delimiter=',')
    for row in lines:
      dataset.append([float(row[1]), float(row[2]), int(row[3])])

# as all the citizens hight and weight are numeric and have same units we can use euclidean distance
def euclidean_distance(instance1, instance2, length):
  distance = 0
  for x in range(length):
    distance += pow((instance1[x] - instance2[x]), 2)
  return math.sqrt(distance)

def get_neighbors(training_set, test_instance, k):
  distances = []
  length = len(test_instance)-1
```

```python
  for x in range(len(training_set)):
    dist = euclidean_distance(test_instance, training_set[x], length)
    distances.append((training_set[x], dist))
  # sorting w.r.t distance
  distances.sort(key=operator.itemgetter(1))
  neighbors = []
  for x in range(k):
    neighbors.append(distances[x][0])
  return neighbors

# returns the maximum voted gender in the neighbors
def get_response(neighbors):
  gender_vote = {}
  for x in range(len(neighbors)):
    gender = neighbors[x][-1]
    if gender in gender_vote:
      gender_vote[gender] += 1
    else:
      gender_vote[gender] = 1
  # sorting gender votes on maximum votes
  sorted_votes = sorted(gender_vote.items(), key=operator.itemgetter(1), reverse=True)
  return sorted_votes[0][0]

# calculates the accuracy of predictions with the test set
def get_accuracy(test_set, predictions):
  correct = 0
  for x in range(len(test_set)):
    if test_set[x][-1] == predictions[x]:
      correct += 1
  return (correct/float(len(test_set))) * 100.0

def get_predictions(training_set, test_set, k):
  predictions = []
  for x in range(len(test_set)):
    neighbors = get_neighbors(training_set, test_set[x], k)
    result = get_response(neighbors)
    predictions.append(result)
  return predictions

def chunks(l, n):
    """Yield successive n-sized chunks from l."""
    for i in range(0, len(l), n):
        yield l[i:i + n]

# the main function
```

```
def main():

  training_set = []
  test_set = []

  # importing training data set
  import_data('data/DWH_Training.csv', training_set)

  # importing test data set
  import_data('data/DWH_test.csv', test_set)

  k = 10
  # spliting the sample into k sets of same size folds
  n = int(len(training_set) / k)
  data = [training_set[i:i + n] for i in range(0, len(
    training_set), n)]
  # remove 11th fold (if any)
  if len(data) > k:
    data.remove(data[-1])
  accuracy = 0
  for j in range(k):
    new_test_set = data[j]
    new_training_set = list(itertools.chain.
   from_iterable(data[0:j])) + list(itertools.chain.
   from_iterable(data[j+1:k]))
    predictions = get_predictions(new_training_set,
   new_test_set, 5)
    accuracy += get_accuracy(new_test_set, predictions)

  print(accuracy / k)

main()
```

Output: 91.73913043478262