School of Computing and Information Systems
The University of Melbourne
COMP90049 Introduction to Machine Learning (Semester 1, 2021)
Week 3: Sample Solution

1. For the following dataset:

apple	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES				
4	0	1	1	FRUIT
5	0	5	2	FRUIT
2	5	0	0	COMPUTER
1	2	1	7	COMPUTER
TEST INSTANCES				
2	0	3	1	?
1	2	1	0	?

(i). Using the **Euclidean distance** measure, classify the test instances using the 1-NN method.

In this method we need to calculate the distance between a test instance and a prototype. To do so we need a to use a similarity/distance function. Here our distance function is the Euclidian Distance:

$$d_E(A,B) = \sqrt{\sum_k (a_k - b_k)^2}$$

Using this function, we will calculate the distance between the test instance and each training instance:

$$d_E(T_1,A) = \sqrt{(2-4)^2 + (0-0)^2 + (3-1)^2 + (1-1)^2} = \sqrt{8} \approx 2.828$$

$$d_E(T_1,B) = \sqrt{(2-5)^2 + (0-0)^2 + (3-5)^2 + (1-2)^2} = \sqrt{14} \approx 3.742$$

$$d_E(T_1,C) = \sqrt{(2-2)^2 + (0-5)^2 + (3-0)^2 + (1-0)^2} = \sqrt{35} \approx 5.916$$

$$d_E(T_1,D) = \sqrt{(2-1)^2 + (0-2)^2 + (3-1)^2 + (1-7)^2} = \sqrt{45} \approx 6.708$$

The nearest neighbour is the one with the smallest distance — here, this is instance `A`, which is a `FRUIT` instance. Therefore, we will classify this instance as `FRUIT`.

The second test instance is similar:

$$d_E(T_2,A) = \sqrt{(1-4)^2 + (2-0)^2 + (1-1)^2 + (0-1)^2} = \sqrt{14} \approx 3.742$$

$$d_E(T_2,B) = \sqrt{(1-5)^2 + (2-0)^2 + (1-5)^2 + (0-2)^2} = \sqrt{40} \approx 6.325$$

$$d_E(T_2,C) = \sqrt{(1-2)^2 + (2-5)^2 + (1-0)^2 + (0-0)^2} = \sqrt{11} \approx 3.317$$

$$d_E(T_2,D) = \sqrt{(1-1)^2 + (2-2)^2 + (1-1)^2 + (0-7)^2} = \sqrt{49} = 7$$

Here, the nearest neighbour is instance `C`, which is a `COMPUTER` instance. Therefore, we will classify this instance as `COMPUTER`.

(ii). Using the **Manhattan distance** measure, classify the test instances using the 3-NN method, for the three weightings we discussed in the lectures: *majority class*, *inverse distance*, *inverse linear distance*.

The first thing to do is to calculate the Manhattan distances, which is like the Euclidean distance, but without the squares/square root:

$$d_M(A, B) = \sum_k |a_k - b_k|$$

$$d_M(T_1, A) = |2-4| + |0-0| + |3-1| + |1-1| = 4$$

$$d_M(T_1, B) = |2-5| + |0-0| + |3-5| + |1-2| = 6$$

$$d_M(T_1, C) = |2-2| + |0-5| + |3-0| + |1-0| = 9$$

$$d_M(T_1, D) = |2-1| + |0-2| + |3-1| + |1-7| = 11$$

$$d_M(T_2, A) = |1-4| + |2-0| + |1-1| + |0-1| = 6$$

$$d_M(T_2, B) = |1-5| + |2-0| + |1-5| + |0-2| = 12$$

$$d_M(T_2, C) = |1-2| + |2-5| + |1-0| + |0-0| = 5$$

$$d_M(T_2, D) = |1-1| + |2-2| + |1-1| + |0-7| = 7$$

The nearest neighbours for the first test instance are `A`, `B`, and `C`. For the second test instance, they are `C`, `A`, and `D`.

The **majority class** weighting method:

In this method we effectively assign a weight of 1 to every instance in the set of nearest neighbours:

- For the first test instance, there are 2 `FRUIT` instances and 1 `COMPUTER` instance. There are more `FRUIT` than `COMPUTER`, so we predict `FRUIT`.
- For the second test instance, there are 2 `COMPUTER` instances and 1 `FRUIT` instance. There are more `COMPUTER` than `FRUIT`, so we predict `COMPUTER`.

The **inverse distance** weighting method:

In this method we first need to choose a value for ϵ, let's say 1:

- For the first test instance:
  - The first neighbour (a `FRUIT`) gets a weight of $\frac{1}{d+\epsilon} = \frac{1}{4+1} = 0.2$
  - The second neighbour (a `FRUIT`) gets a weight of $\frac{1}{d+\epsilon} = \frac{1}{6+1} \approx 0.14$
  - The first neighbour (a `COMPUTER`) gets a weight of $\frac{1}{d+\epsilon} = \frac{1}{9+1} = 0.1$

  Overall, `FRUIT` instances have a score of 0.2+0.14 = 0.34, and `COMPUTER` instances have a score of 0.1, so we would predict `FRUIT` for this instance.

- For the second test instance:
    - The first neighbour (a COMPUTER) gets a weight of $\frac{1}{d+\epsilon} = \frac{1}{5+1} \approx 0.17$
    - The second neighbour (a FRUIT) gets a weight of $\frac{1}{d+\epsilon} = \frac{1}{6+1} \approx 0.14$
    - The first neighbour (a COMPUTER) gets a weight of $\frac{1}{d+\epsilon} = \frac{1}{7+1} = 0.12$

  Overall, FRUIT instances have a score 0.14, and COMPUTER instances have a score of 0.17+0.12=0.29, so we would predict COMPUTER for this instance.

  Note: If we have used Euclidean distance (instead of Manhattan distance) would give a different result here.

The **inverse linear distance** weighting method:

In this method we are going to weight instances by re-scaling the distances according to the following formula, where d_j is the distance of the j^{th} nearest neighbour:

$$w_j = \frac{d_3 - d_j}{d_3 - d_1}$$

Note: Compared to the lecture version, we have substituted k = 3 here, because we are using the 3-Nearest Neighbour method.

- For the first test instance:
    - The first neighbour (a FRUIT) gets a weight of $\frac{d_3-d_1}{d_3-d_1} = \frac{9-4}{9-4} = 1$
    - The second neighbour (a FRUIT) gets a weight of $\frac{d_3-d_2}{d_3-d_1} = \frac{9-6}{9-4} = 0.6$
    - The first neighbour (a COMPUTER) gets a weight of $\frac{d_3-d_3}{d_3-d_1} = \frac{9-9}{9-4} = 0$

  Overall, FRUIT instances have a score of 1+0.6 = 1.6, and COMPUTER instances have a score of 0, so we would predict FRUIT for this instance.

- For the second test instance:
    - The first neighbour (a COMPUTER) gets a weight of $\frac{d_3-d_1}{d_3-d_1} = \frac{7-5}{7-5} = 1$
    - The second neighbour (a FRUIT) gets a weight of $\frac{d_3-d_2}{d_3-d_1} = \frac{7-6}{7-5} = 0.5$
    - The first neighbour (a COMPUTER) gets a weight of $\frac{d_3-d_3}{d_3-d_1} = \frac{7-7}{7-5} = 0$

  Overall, FRUIT instances have a score of 0.5, and COMPUTER instances have a score of 1+0=1, so we would predict COMPUTER for this instance.

(iii). Can we do weighted k-NN using **cosine similarity**?

Of course! If anything, this is easier than with a distance, because we can assign a weighting for each instance using the cosine similarity directly. An overall weighting for a class can be obtained by summing the cosine scores for the instances of the corresponding class, from among the set of nearest neighbours.

Let's summarise all of these predictions in a table (overleaf). We can see that there is some divergence for these methods, depending on whether B or D is the 3rd neighbour for T_2:

Inst	Measure	k	Weight	Prediction
T_1	d_E	1	-	FRUIT
		3	Maj	FRUIT
		3	ID	FRUIT
		3	ILD	FRUIT
	d_M	1	-	FRUIT
		3	Maj	FRUIT
		3	ID	FRUIT
		3	ILD	FRUIT
	cos	1	-	FRUIT
		3	Maj	FRUIT
		3	Sum	FRUIT
T_2	d_E	1	-	COMPUTER
		3	Maj	FRUIT
		3	ID	FRUIT
		3	ILD	COMPUTER
	d_M	1	-	COMPUTER
		3	Maj	COMPUTER
		3	ID	COMPUTER
		3	ILD	COMPUTER
	cos	1	-	COMPUTER
		3	Maj	FRUIT
		3	Sum	FRUIT

2. Approximately 1% of women aged between 40 and 50 have breast cancer. 80% of mammogram screening tests detect breast cancer when it is there. 90% of mammograms DO NOT show breast cancer when it is **NOT** there. Based on this information, complete the following table.

Cancer	**Probability**
No	99%
Yes	1%

Cancer	**Test**	**Probability**
Yes	Positive	80%
Yes	Negative	?
No	Positive	?
No	Negative	90%

Based on the probability rule of sum for mutually exclusive events (events that cannot both happen at the same time), we know that the sum of positive and negative test results should sum up to 1 (or 100%).

Therefore, when we have a patient with cancer (`Cancer` = '`Yes`'), and we know that there is 80% probability that the test detects it (`Test` returns '`Positive`'), it means that there is 20% chance (1 – 0.80 = 0.20) that the test does not detect the cancer (`Test` returns '`Negative`' results). We call this a **False Negative** (wrong negative); you will learn more about it later in lectures.

Similarly, when a patient does not have cancer (`Cancer` = '`No`'), and we have that there is 90% chance that the test proves that (`Test` returns '`Negative`'), it means that there is 9% chance (1 – 0.9 = 0.1) that the test detects cancer (returns '`positive`' results) when it is not there! We call this a **False Positive** (wrong positive), and again you will learn more about it later in lectures when we talk about evaluations.

So, the filled table would be as follow:

Cancer	**Test**	**Probability**
Yes	Positive	80%
Yes	Negative	20%
No	Positive	10%
No	Negative	90%

3. Based on the results in question 2, calculate the **marginal probability** of 'positive' results in a Mammogram Screening Test.

According to the law of total probability, we know that

$$P(A) = \sum_{n} P(A|B_n)\, P(B_n)$$

So, to calculate the probability of '`positive`' result for Test, we will have:

$$P(Test = 'positive') = P(Test = 'positive'|Cancer = 'no').P(Cancer = 'no') + P(Test = 'positive'|Cancer = 'yes').P(Cancer = 'yes')$$

Based on the question definition, we know that the chance of having breast cancer (for females aged between 40 and 50) is 1% . So `P(Cancer = 'yes') = 0.01` and `P(Cancer = 'no') = 0.99`.

From question 1, we know that the probability of a `positive` test result is 80% for a patient with cancer (`P(Test = 'positive'|Cancer= 'yes')=0.8`) and the probability of a `positive` test result is 10% for a patient with no cancer (`P(Test='positive'|Cancer='yes')=0.1`).

So, we have:

$$P(Test = 'positive') = 0.1 \times 0.99 + 0.8 \times 0.01 = 0.107$$

We can show all these in a **Joint Probability Distribution** table as follow.

		Test		Total
		Positive	Negative	
Cancer	Yes	0.01 x 0.8 = 0.008	0.01 x 0.2 = 0.002	0.01
	No	0.99 x 0.1 = 0.099	0.99 x 0.9 = 0.891	0.99
Total		0.107	0.893	1

We call the totals (row and column) the **Marginal Probability**, because they are in the margin!

4. Based on the results in question 2, calculate *P(Cancer = 'Yes' | Test = 'Positive')*, using the Bayes Rule.

According to the Bayesian Rule, we know that we can calculate the probability that a person actually has breast cancer given that her mammography test results return positive, using the following formula:

$$P(Cancer = 'yes'|Test = 'positive') = \frac{P(Test = 'positive'|Cancer = 'yes').P(Cancer = 'yes')}{P(Test = 'positive')}$$

Based on the given information in the question text, we know that "80% of mammogram screening tests detect breast cancer when it is there", so `P(Test = 'positive' | Cancer = 'yes')` is `0.8 (80%)`.

Also, there `1%` chance of having breast cancer (for females aged between 40 and 50). So `P(Cancer = 'yes') = 0.01`.

Also, from Question2, we have the `P(Test = 'positive') = 0.107` (the expectation of '`positive`' results for a mammogram test).

So we can easily calculate the `P(Cancer= 'yes' | Test = 'positive')`:

$$P(Cancer = 'yes' | Test = 'positive') = \frac{0.8 \times 0.01}{0.107} \cong 0.075 = 7.5\%$$

This result shows that even if a mammography test results return positive, there is only a 7.5% chance that the person actually has Cancer! ☺