School of Computing and Information Systems
The University of Melbourne
COMP30027 Machine Learning (Semester 1, 2021)

Sample Solutions: Week 5

1. For the following dataset:

ID	*Outl*	*Temp*	*Humi*	*Wind*	PLAY
TRAINING INSTANCES					
A	s	h	h	F	N
B	s	h	h	T	N
C	o	h	h	F	Y
D	r	m	h	F	Y
E	r	c	n	F	Y
F	r	c	n	T	N
TEST INSTANCES					
G	o	c	n	T	?
H	s	m	h	F	?

a) Classify the test instances using the method of **0-R.**

0-R is the quintessentially baseline classifier: we throw away all of the attributes, other than the class labels, and just predict each test instance according to whichever label is most common in the training data. (Hence, it is also common called the "majority class classifier".)

In this case, the two labels Y and N are equally common in the training data — so we are required to apply a tiebreaker. Remember that we're simply choosing one label to be representative of the entire collection, and both labels seem equally good for that here: so, let's say N.

Consequently, both test instances are classified as N.

b) Classify the test instances using the method of **1-R**.

1-R is a slightly better baseline, which requires us to choose a single attribute to represent the entire decision–making process. For example, if `Outl` is our preferred attribute, then we base the classification of each test instance solely based on its value of `Outl`. (This is sometimes called a "Decision Stump".)

Given our preferred attribute, we will label a test instance according to whichever label in the training data was most common, for training instances with the corresponding attribute value.

How do we decide which attribute to choose? Well, the most common method is simply by counting the errors made on the training instances.

- Let's say we chose `Outl`: first, we need to observe the predictions for the 3 values (`s`, `o`, and `r`), and then we will count up the errors that those predictions will make on the training data (it turns out that we can do this simultaneously):
  - When `Outl` is `s`, there are two such instances in the training data. Both of these instances are labelled as `N`: our label for this attribute value will be `N`. We will therefore predict the label of both of these training instances correctly (we will predict `N`, and both of these instances are actually `N`).
  - When `Outl` is `o`, there is just a single instance in the training data, labelled as `Y`: our label for this attribute value will be `Y`. We will therefore predict the label of this training instance correctly.
  - When `Outl` is `r`, there are three such instances in the training data. Two of these instances are labelled as `Y`, the other is `N`: our label for this attribute value will

be Y (as there are more Y instances than N instances — you can see that we're applying the method of 0-R here). We will therefore make one error: the instance which was actually n.

- In total, that is 1 error for Outl. Hopefully, you can see that this is a very wordy explanation of a very simple idea. (We will go through the other attributes more quickly.)
- For Temp:
  - When Temp is h, there are two N instances and one Y: we will make one mistake.
  - When Temp is m, there is one Y instance: we won't make any mistakes.
  - When Temp is c, there is one N instance and one Y instance: we will make one mistake.
- This is 2 errors in total for Temp, which means that it's less good that Outl.

We'll leave the other attributes as an exercise, and assume that Outl was the best attribute (it's actually tied with Wind): how do the test instances get classified?

- For test instance G, Outl is o — the 0-R classifier over the o instances from the training data gives Y, so we predict Y for this instance.
- For test instance H, Outl is s — the 0-R classifier over the s instances from the training data gives N, so we predict N for this instance.

c) Classify the test instances using the **ID3 Decision Tree** method:

i) Using the **Information Gain** as a splitting criterion

For Information Gain, at each level of the decision tree, we're going to choose the attribute that has the largest difference between the entropy of the class distribution at the parent node, and the average entropy across its daughter nodes (weighted by the fraction of instances at each node);

$$IG(A|R) = H(R) - \sum_{i \in A} P(A = i)H(A = i)$$

In this dataset, we have 6 instances total — 3 Y and 3 N. The entropy at the top level of our tree is $H(R) = -\left[\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right]$

This is a very even distribution. We're going to hope that by branching the tree according to an attribute, that will cause the daughters to have an uneven distribution - which means that we will be able to select a class with more confidence - which means that the entropy will go down.

For example, for the attribute Outl, we have three attribute values: s, o, r.

- When Outl=s, there are 2 instances, which are both N. The entropy of this distribution is $H(O = s) = -\left[0\log_2 0 + \frac{2}{2}\log_2\frac{2}{2}\right] = 0$. Obviously, at this branch, we will choose N with a high degree of confidence.
- When Outl=o, there is a single instance, of class Y. The entropy here is going to be 0 as well.
- When Outl=r, there are 2 Y instances and 1 N instance. The entropy here is $H(o = r) = -\left[\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right] \approx 0.9183$

To find the average entropy (the "mean information"), we sum the calculated entropy at each daughter multiplied by the fraction of instances at that daughter: $MI(O) = \frac{2}{6}(0) + \frac{1}{6}(0) + \frac{3}{6}(0.9183) \approx 0.4592$

The overall Information Gain here is $IG(O) = H(R) - MI(O) = 1 - 0.4592 = 0.5408$.

	R	Outl			Temp			H		Wind		ID					
		s	o	r	h	m	c	h	n	T	F	A	B	C	D	E	F
Y	3	0	1	2	1	1	1	2	1	0	3	0	0	1	1	1	0
N	3	2	0	1	2	0	1	2	1	2	1	1	1	0	0	0	1
Total	6	2	1	3	3	1	2	4	2	2	4	1	1	1	1	1	1
P(Y)	1/2	0	1	2/3	1/3	1	1/2	1/2	1/2	0	3/4	0	0	1	1	1	0
P(N)	1/2	1	0	1/3	2/3	0	1/2	1/2	1/2	1	1/4	1	1	0	0	0	1
H	1	0	0	0.9183	0.9183	0	1	1	1	0	0.8112	0	0	0	0	0	0
MI		0.4592			0.7924			1		0.5408		0					
IG		0.5408			0.2076			0		0.4592		1					
SI		1.459			1.459			0.9183		0.9183		2.585					
GR		0.3707			0.1423			0		0.5001		0.3868					

The table above lists the Mean Information and Information Gain, for each of the 5 attributes.

At this point, `ID` has the best information gain, so hypothetically we would use that to split the root node. At that point, we would be done, because each daughter is purely of a single class — however, we would be left with a completely useless classifier! (Because the IDs of the test instances won't have been observed in the training data.)

Instead, let's take the second-best attribute:`Outl`.

There are now three branches from our root node: for s, for o, and for r. The first two are pure, so we can't improve them anymore. Let's examine the third branch (`Outl=r`):

- Three instances (D, E, and F) have the attribute value r; we've already calculated the entropy here to be 0.9183.
- If we split now according to Temp, we observe that there is a single instance for the value m (of class N, the entropy is clearly 0); there are two instances for the value c, one of class Y and one of class N (so the entropy here is 1). The mean information is $\frac{1}{3}(0) + \frac{2}{3}(1) \approx 0.6667$, and the information gain at this point is 0.9183 − 0.6667 ≈0.2516.
- For `Humi`, we again have a single instance (with value `h`, class Y, H = 0), and two instances (of `n`) split between the two classes (H = 1). The mean information here will also be 0.6667, and the information gain 0.2516.
- For `Wind`, there are two `F` instances, both of class Y (H = 0), and one `T` instance of class N (H = 0). Here, the mean information is 0 and the information gain is 0.9183.
- `ID` would still look like a good attribute to choose, but we'll continue to ignore it.
- All in all, we will choose to branch based on Wind for this daughter.

All of the daughters of `r` are pure now, so our decision tree is complete:

- `Outl=o` ∪ (`Outl=r` ∩ `Wind=F`) → `Y` (so we classify `G` as Y)
- `Outl=s` ∪ (`Outl=r` ∩ `Wind=T`) → `N` (so we classify `H` as N)

ii) Using the **Gain Ratio** as a splitting criterion

Gain ratio is similar, except that we're going to weight down (or up!) by the "split information" — the entropy of the distribution of instances across the daughters of a given attribute.

For example, we found that, for the root node, `Outl` has an information gain of 0.5408. There are 2 (out of 6) instances at the s daughter, 1 at the o daughter, and 3 at the r daughter.

The split information for Outl is $SI(o) = -\left[\frac{2}{6}\log_2\frac{2}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{3}{6}\log_2\frac{3}{6}\right] \approx 1.459.$

The Gain ratio is consequently $GR(o) = \frac{IG(o)}{SI(o)} \approx \frac{0.5408}{1.459} \approx 0.3707$

The values for split information and gain ratio for each attribute at the root node are shown in the table above. The best attribute (with the greatest gain ratio) at the top level this time is Wind.

Wind has two branches: T is pure, so we focus on improving F (which has 3 Y in- stances (C, D, E), and 1 N instance (A)). The entropy of this daughter is 0.8112.

- For Outl, we have a single instance at s (class N, H = 0), a single instance at o (class Y, H = 0), and 2 Y instances at r (H = 0). The mean information here is clearly 0; the information gain is 0.8112. The split information is $SI(o|(W = F)) = -\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{2}\log_2\frac{1}{2}\right] = 1.5$, so the gain ratio is $GR\left(o\middle|(W = F)\right) = \frac{0.8112}{1.5} \approx 0.5408$
- For Temp, we have two h instances (one Y and one N, so H = 1), a single m instance (Y, H = 0), and a single c instance (Y, H = 0). The mean information is $\frac{2}{4}(1) + \frac{1}{4}(0) + \frac{1}{4}(0) = 0.5$, so the information gain is 0.8112-0.5 = 0.3112. The distribution of instances here is the same as Outl, so the split information is also 1.5, and the gain ratio is $GR\left(T\middle|(W = F)\right) = \frac{0.3112}{1.5} \approx 0.2075$
- For Humi, we have 3 h instances (2 Y and 1 N, H = 0.9183), and 1 n instance (Y, H = 0): the mean information is $\frac{3}{4}$ (0.9183) + $\frac{1}{4}$(0) = 0.6887 and the information gain is 0.8112 − 0.6887 = 0.1225. The split information is $SI(H|(W = F)) = -\left[\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right] \approx 0.8112$, so the gain ratio is $GR\left(H\middle|(W = F)\right) = \frac{0.1225}{0.8112} \approx 0.1387.$
- For ID, the mean information is obviously still 0, so the information gain is 0.8112. The split information at this point is $-\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4}\right] = 2$, so the gain ratio is approximately 0.4056.

Of our four choices at this point, Outl has the best gain ratio. The resulting daughters are all pure, so the decision tree is finished:

- Wind=F ∩(Outl=o ∪ Outl=r) → Y
- Wind=T ∪(Wind=F ∩ Outl=s) → N (so we classify G and H as N)

Note that this decision tree is superficially similar to the tree above but gives different classifications because of the order in which the attributes are considered.

Note also that we didn't need to explicitly ignore the ID attribute for Gain Ratio (as we needed to do for Information Gain) — the split information pushed down its "goodness" to the point where we didn't want to use it anyway!

2. 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	?

a) 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`.

b) 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* ($\varepsilon = 1$), *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, epsilon (ε) is given as 1:

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

  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 $\frac{1}{d+\varepsilon} = \frac{1}{5+1} \approx 0.17$
  - The second neighbour (a `FRUIT`) gets a weight of $\frac{1}{d+\varepsilon} = \frac{1}{6+1} \approx 0.14$
  - The first neighbour (a `COMPUTER`) gets a weight of $\frac{1}{d+\varepsilon} = \frac{1}{7+1} = 0.12$

  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.

c) 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