HOMEWORK 3

EXERCISES ON DATA, DATA EXPLORATION AND CLASSIFICATION

Submitted by

Arun Rajagopalan

A20360689

For CS422 Data Mining

Illinois Institute of Technology

EXERCISES ON DATA, DATA EXPLORATION AND CLASSIFICATION

Exercise 1 (Chapter 2 #18):

(i) Compute the Hamming distance and the Jaccard similarity between the following two binary vectors: x = 0101010001; y = 0100011000

Hamming distance is a measure of the count of positions in two binary strings/vectors of equal length where the corresponding bits are dissimilar and distinct. It can also be viewed as the smallest number of positions that need to be changed to transform one binary vector to another. Also it serves as a measure of number of errors that might have undesirably changed one vector to another.

Thus, the **Hamming distance between the given x and y is 3** since we observe that the corresponding bits are changing in 3 positions as highlighted below.
x = 010**1**01**0**00**1**; y = 010**0**01**1**00**0**

On the other hand, Jaccard index or similarity coefficient is a measure mainly used to compare the similarity between two vectors. It is derived by dividing the number of positions among the given vectors that have corresponding bits as 1 divided by the complement of number of positions that have corresponding bits as 0 which can be expressed mathematically as follows.

Jaccard similarity = number of 1-1 matches /(number of bits – number 0-0 matches)

In our case, for the given x and y,
number of 1-1 matches: 2 (x = 0**1**010**1**0001; y = 0**1**000**1**1000)
number of 0-0 matches: 5 (x = **0**1**0**10**00**1; y = **0**1**0**0**0**11**00**0)

Hence **Jaccard similarity = 2/(10-5) = 2/5 = 0.4**

(ii) Suppose that you are comparing how similar two organisms of different species are in terms of the number of genes they share. Describe which measure, Hamming or Jaccard, you think would be more appropriate for comparing the genetic make up of two organisms. Explain. (Assume that each animal is represented as a binary vector, where each attribute is 1 if a particular gene is present in the organism and 0 otherwise.)

The goal is to compare the genetic structure of organisms using the shared genes or bits. It would make better sense to compare the **similarities** in this task since the organisms belong to diverse species. Thus, the **Jaccard similarity measure will be more appropriate**.

(iii) If you wanted to compare the genetic make up of two organisms of the same species, e.g., two human beings, would you use the Hamming distance, the Jaccard coefficient, or a different measure of similarity or distance? Explain. (Note that two human beings share > 99.9% of the same genes.)

When we are studying the genetic structure of same species, we are more interested on differences because the differences shape up the character of every single organism and bring about individuality. Hence, **Hamming distance would suit this case of comparison as we are honing on the differences**.

Exercise 2 (Chapter 3 #8):

(i) Describe how a box plot can give information about whether the value of an attribute is symmetrically distributed. What can you say about the symmetry of the distributions of the attributes shown in Figure 3.11?

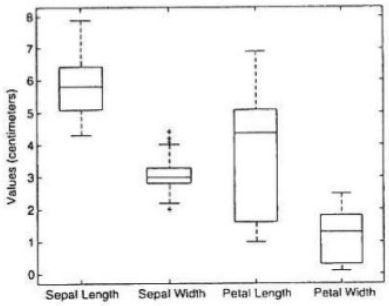

Figure 3.11. Box plot for Iris attributes.

Box plot is a comparatively compressed form of visualization which is used to represent the distribution of instances of a singular numerical attribute through the quartiles. In this type of plot, 25th and the 75th percentiles of values are denoted by the lower and the upper endings of the box. 50th percentile is marked by the dash contained within the box and thus the space between different regions of box shows the spread of values. Values lying outside the box are represented by dotted lines called whiskers or tails while the upper and lower dashes of the tails represent 10th and the 90th percentiles. Outliers, if any, are marked by the + symbols.

From the given box plot, comparing all four attributes of Iris dataset, we can say that the sepal length and sepal width have nearly uniform class distributions with atleast three -fourths of the values lying inside the box, since the 50th percentile line or the median is near the middle of the box. The tails and the outliers also represent instances but are not given much credibility as they represent only a few records which may sometimes have noise values. Petal length and petal width have a relatively skewed class spread since the median is pushed towards one end of the box.

Exercise 3 (Chapter 4 #5):

Consider the following data set for a binary class problem

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

For getting an optimal decision tree, we choose the attribute to split upon carefully, at every node. To do so, we commonly use impurity measure of child nodes with that of the parent node. If the measure of impurity is lesser, the class spread will be more skewed. Similarly, if the impurity measure is

maximum, then the class spread will be uniform. This degree/measure of impurity can be derived in quite a few ways, some of them being entropy, gini index and classification error.

Entropy and gini index at a node t can be calculated by using the below formulae respectively.

Entropy (t) = $- {}^{c-1}\Sigma_{i=0}\, p(i|t) \log_2 p(i|t)$
Gini (t) = $1 - {}^{c-1}\Sigma_{i=0}\, [p(i|t)]^2$
where c indicates the number of classes and p(i|t) indicates the instances from class i at any given node t.

Using any of the impurity measure, we compute the gain for the all attribute at the given node and the decision tree algorithm picks the attribute that gives maximum gain.

The gain (Δ) for the chosen attribute at the given node is derived by

$\Delta = I(\text{parent}) - {}^{k}\Sigma_{j=1}\, (N(v_j)/N)\, I(v_j)$
where I is the impurity measure, N is the number of number of instances at parent node, $N(v_j)$ is the number of instances at the child node v_j and k is the count of attributes.

(i) Calculate the information gain when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

Now, for the given dataset, the original entropy prior to splitting on any attribute, E_{PS} is given by

$E_{PS} = -(4/10) \log_2 (4/10) - (6/10) \log_2 (6/10) = 0.5288 + 0.4422$
$E_{PS} = 0.971$
since among the 10 overall instances, there are 4 instances with + class and 6 instances with - class.

Splitting the dataset using the attribute A, we get the following contingency table or confusion matrix.

	A = T	A = F
Class value = +	4	0
Class value = -	3	3

Calculating the entropy and in turn, the information gain achieved by splitting on A,

Entropy for A = T, $E_{AT} = - (4/7) \log_2 (4/7) - (3/7) \log_2 (3/7) = 0.4613 + 0.5239$
$E_{AT} = 0.9852$
since there are 4 instances with + class and 3 instances with - class for A = T.

Entropy for A = F, $E_{AF} = - (0/3) \log_2 (0/3) - (3/3) \log_2 (3/3)$
$E_{AF} = 0$
since there are no instances with + class and 3 instances with - class for A = F.

Gain using A for split, $\Delta_A = E_{PS} - (7/10)\, E_{AT} - (3/10)\, E_{AF} = 0.971 - 0.7(0.9852) - 0.3(0)$

$\Delta_A = 0.28136$
since there are 7 total instances for A = T and 3 total instances for A = F.

Similarly when splitting the dataset using the attribute B, the contingency table is as below.

	B = T	B = F
Class value = +	3	1
Class value = -	1	5

Entropy for B = T, $E_{BT} = - (3/4) \log_2 (3/4) - (1/4) \log_2 (1/4) = 0.3113 + 0.5$
$E_{BT} = 0.8113$
since there are 3 instances with + class and 1 instance with - class for B = T.

Entropy for B = F, $E_{BF} = - (1/6) \log_2 (1/6) - (5/6) \log_2 (5/6) = 0.4308 + 0.2192$
$E_{BF} = 0.65$
since there is 1 instance with + class and 5 instances with - class for B = F.

Gain using B for split, $\Delta_B = E_{PS} - (4/10) E_{BT} - (6/10) E_{BF} = 0.971 - 0.4(0.8113) - 0.6(0.65)$
$\Delta_B = 0.25648$
since there are 4 total instances for B = T and 6 total instances for B = F.

Since the information gain by using attribute A is higher, **decision tree algorithm would choose A** to split the root node.

(ii) Calculate the gain in the Gini index when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

The original gini index prior to splitting on any attribute, G_{PS} is given by

$G_{PS} = 1 - (4/10)^2 - (6/10)^2 = 0.48$
since among the 10 overall instances, there are 4 instances with + class and 6 instances with – class.

Gain calculated using Gini index by partitioning root using A, Δ_A is as follows

Gini for A = T, $G_{AT} = 1 - (4/7)^2 - (3/7)^2 = 0.4898$
since there are 4 instances with + class and 3 instances with - class for A = T.

Gini for A = F, $G_{AF} = 1 - (0/3)^2 - (3/3)^2 = 0$
since there are no instances with + class and 3 instances with - class for A = F.

Gain, $\Delta_A = G_{PS} - (7/10) G_{AT} - (3/10) G_{AF} = 0.48 - 0.7(0.4898) - 0.3(0)$
$\Delta_A = 0.13714$
since there are 7 total instances for A = T and 3 total instances for A = F.

Gain calculated using Gini index by partitioning using B, Δ_B is as follows

Gini for B = T, $G_{BT} = 1 - (3/4)^2 - (1/4)^2 = 0.375$
since there are 3 instances with + class and 1 instance with - class for B = T.

Gini for B = F, $G_{BF} = 1 - (1/6)^2 - (5/6)^2 = 0.2778$
since there is 1 instance with + class and 5 instances with - class for B = F.

Gain, $\Delta_B = G_{PS} - (4/10)\, G_{BT} - (6/10)\, G_{BF} = 0.48 - 0.4(0.375) - 0.6(0.2778)$
$\Delta_B = 0.16332$
since there are 4 total instances for B = T and 6 total instances for B = F.

Since the information gain by using attribute B is higher for this case, **decision tree algorithm would choose B** to split the root node.

(iii) Figure 4.13 shows that entropy and the Gini index are both monotonously increasing on the range [0, 0.5] and they are both monotonously decreasing on the range [0.5, 1]. Is it possible that information gain and the gain in the Gini index favor different attributes? Explain.

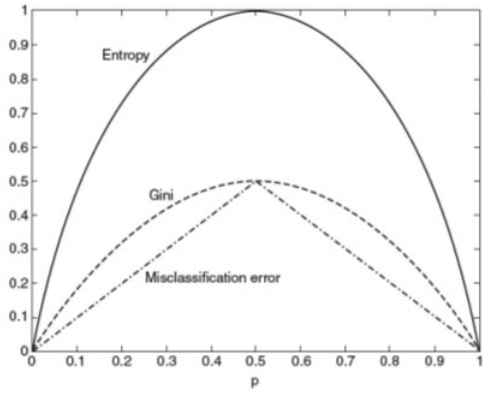

Figure 4.13. Comparison among the impurity measures for binary classification problems.

Yes. In spite of our observation from the above comparison that both entropy and gini behave similarly, like they both attain maximum value when class distribution is uniform and they increase and decrease symmetrically when the class distribution changes, it is possible that gain calculated using entropy and gini might favour different attributes. Since gain is calculated by scaling the impurity measures and finding difference between them, obviously different entropy and gini values expectedly end up with different gains that may favour different attributes like it does for the given dataset in this exercise.

Exercise 4 (Chapter 4 #7):

The following table summarizes a data set with three attributes A, B, C and two class labels +, −. Build a two-level decision tree.

(i) According to the classification error rate, which attribute would be chosen as the first splitting attribute? For each attribute, show the contingency table and the gains in classification error rate.

A	B	C	Number of Instances	
			+	−
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

Classification error, which is also a measure of impurity levels, can be achieved by using the below formula

Classification error = $1 - \max_i [p(i|t)]$
where, as before, p(i|t) indicates the instances from class i at any given node t.

Now, error rate on the original dataset before partitioning can be written as

CE_{PS} (Error rate prior to splitting) = 1 – Max (50/100, 50/100) = 1 – 0.5 = 0.5
since there are 50 instances each with + class and – class among the total 100 instances.

Partitioning on A, we get the following contingency table, using which we calculate error rates and finally the gain.

	A = T	A = F
Class value = +	25	25
Class value = -	0	50

Error rate for A = T, $CE_{AT} = 1 - Max (25/25, 0/25) = 1 - 1$
$CE_{AT} = 0$

Error rate for A = F, $CE_{AF} = 1 - Max (25/75, 50/75) = 1 - (50/75)$
$CE_{AF} = 0.3334$

Gain using A for split, $\Delta_A = CE_{PS} - (25/100)\, CE_{AT} - (75/100)\, CE_{AF} = 0.5 - 0.25(0) - 0.75(0.3334)$
$\Delta_A = 0.24995$ or 0.25

Similarly for B and C, we calculate the error rates to find gain obtained by splitting on them.

Splitting on B:

Contingency table:

	B = T	B = F
Class value = +	30	20

Class value = -	20	30

Error rate for B = T, $CE_{BT} = 1 – Max\ (30/50, 20/50) = 1 – 0.6$
$CE_{BT} = 0.4$

Error rate for B = F, $CE_{BF} = 1 – Max\ (20/50, 30/50) = 1 – 0.6$
$CE_{BF} = 0.4$

Gain using B for split, $\Delta_B = CE_{PS} – (50/100)\ CE_{BT} – (50/100)\ CE_{BF} = 0.5 – 0.5(0.4) – 0.5(0.4)$
$\Delta_B = 0.1$

Splitting on C:

Contingency table:

	C = T	C = F
Class value = +	25	25
Class value = -	25	25

Error rate for C = T, $CE_{CT} = 1 – Max\ (25/50, 25/50) = 1 – (25/50)$
$CE_{CT} = 0.5$

Error rate for C = F, $CE_{CF} = 1 – Max\ (25/50, 25/50) = 1 – (25/50)$
$CE_{CF} = 0.5$

Gain using C for split, $\Delta_C = CE_{PS} – (50/100)\ CE_{CT} – (50/100)\ CE_{CF} = 0.5 – 0.5(0.5) – 0.5(0.5)$
$\Delta_C = 0$

Since **splitting by A gives higher gain**, it will be chosen as the first splitting attribute by the algorithm.

(ii) Repeat for the two children of the root node.

Splitting by A, we can rearrange the provided dataset as below for easier analysis.

A	B	C	Number of Instances	
			+	-
T	T	T	5	0
T	F	T	20	0
T	T	F	0	0
T	F	F	0	0
F	**T**	**T**	**0**	**20**
F	**F**	**T**	**0**	**5**

F	**T**	**F**	**25**	**0**
F	**F**	**F**	**0**	**25**

When A = T, we get 25 instances with class + and no instances of class -. Hence it is a pure node and does not need further splitting.
At the case of A = F, the child node is impure, and the spread of records across B and C in this node is shown by the emboldened part in the above table.

The error rate at the A = F child node, before partitioning is
CE_{AFPS} (Error rate at A = F node prior to splitting) = 1 – Max (25/75, 50/75) = 1 – 0.6667
CE_{AFPS} = 0.3333
since there are 25 instances with + class and 50 instances with – class among the total 75 instances.

Here, we split on B and C, come up with the contingency table, calculate the error rates and at last the gain with which we decide which attribute will be chosen next.

Splitting A = F child on B:

Contingency table:

	B = T	B = F
Class value = +	25	0
Class value = -	20	30

Error rate at A = F child for B = T, CE_{AFBT} = 1 – Max (25/45, 20/45) = 1 – 0.5556
CE_{AFBT} = 0.4444

Error rate for B = F, CE_{AFBF} = 1 – Max (0/30, 30/30) = 1 – 1
CE_{AFBF} = 0

Gain using B for split, Δ_{AFB} = CE_{AFPS} – (45/75) CE_{AFBT} – (30/75) CE_{AFBF} = 0.3333 – 0.6(0.4444) – 0.4(0)
Δ_{AFB} = 0.0667

Splitting A = F child on C:

Contingency table:

	C = T	C = F
Class value = +	0	25
Class value = -	25	25

Error rate at A = F child for C = T, CE_{AFCT} = 1 – Max (0/25, 25/25) = 1 – 1
CE_{AFCT} = 0

Error rate for C = F, CE_{AFCF} = 1 – Max (25/50, 25/50) = 1 – 0.5

$CE_{AFCF} = 0.5$

Gain using C for split, $\Delta_{AFC} = CE_{AFPS} – (25/75)\ CE_{AFCT} – (50/75)\ CE_{AFCF} = 0.3333 – 0.3333(0) – 0.6666(0.5)$
$\Delta_{AFC} = 0$

As the gain using B at A = F child, Δ_{AFB} is greater, **B will be used to split that node**.

(iii) How many instances are misclassified by the resulting decision tree?

A = T is a pure leaf node. At A = F child, since B is used to split, we can identify the misclassified instances by observing the contingency table of using B at this node.

	B = T	B = F
Class value = +	25	0
Class value = -	20	30

When B = F, all 30 records are classified correctly while B = T classifies only 25 of the 45 instances as class +.
Hence, **20 records are misclassified** here leaving the tree with an error rate of 20/100.

(iv) Repeat parts (a), (b), and (c) using C as the splitting attribute.

As seen earlier, splitting the original dataset on C, yields the below contingency table.

	C = T	C = F
Class value = +	25	25
Class value = -	25	25

Since no node is pure here, we have to split both C = T and C = F child nodes.

C	A	B	Number of Instances	
			+	-
T	T	T	5	0
T	F	T	0	20
T	T	F	20	0
T	F	F	0	5
F	T	T	0	0
F	F	T	25	0
F	T	F	0	0
F	F	F	0	25

C = T node:

The original error rate at the C = T child node is
CE_{CTPS} (Error rate at C = T node prior to splitting) = 1 – Max (25/50, 25/50) = 1 – 0.5
CE_{CTPS} = 0.5
since there are 25 instances with + class and 25 instances with – class among the total 50 instances.

Splitting C = T child on A:

Contingency table:

	A = T	A = F
Class value = +	25	0
Class value = -	0	25

Error rate at C = T child for A = T, CE_{CTAT} = 1 – Max (25/25, 0/25) = 1 – 1
CE_{CTAT} = 0

Error rate for A = F, CE_{CTAF} = 1 – Max (0/25, 25/25) = 1 – 1
CE_{CTAF} = 0

Gain using A for split, Δ_{CTA} = CE_{CTPS} – (25/50) CE_{CTAT} – (25/50) CE_{CTAF} = 0.5 – 0.5(0) – 0.5(0)
Δ_{CTA} = 0.5

Splitting C = T child on B:

Contingency table:

	B = T	B = F
Class value = +	5	20
Class value = -	20	5

Error rate at C = T child for B = T, CE_{CTBT} = 1 – Max (5/25, 20/25) = 1 – 0.8
CE_{CTBT} = 0.2

Error rate for B = F, CE_{CTBF} = 1 – Max (20/25, 5/25) = 1 – 0.8
CE_{CTBF} = 0.2

Gain using B for split, Δ_{CTB} = CE_{CTPS} – (25/50) CE_{CTBT} – (25/50) CE_{CTBF} = 0.5 – 0.5(0.2) – 0.5(0.2)
Δ_{CTB} = 0.3

As the gain Δ_{CTA} is greater than $\Delta_{CTB,}$ **A will be used to split the C = T node**. There are no misclassified instances when we split this node with A.

C = F node:

The overall error rate at the C = F child node is
CE_{CFPS} (Error rate at C = F node prior to splitting) = 1 – Max (25/50, 25/50) = 1 – 0.5
CE_{CFPS} = 0.5
since there are 25 instances with + class and 25 instances with – class among the total 50 instances.

Splitting C = F child on A:

Contingency table:

	A = T	A = F
Class value = +	0	25
Class value = -	0	25

Error rate at C = F child for A = T, CE_{CFAT} = 1 – Max (0/0, 0/0)
CE_{CFAT} = indeterminate

Error rate for A = F, CE_{CFAF} = 1 – Max (25/50, 25/50) = 1 – 0.5
CE_{CFAF} = 0.5

Gain using A for split, Δ_{CFA} = CE_{CFPS} – (0/50) CE_{CFAT} – (50/50) CE_{CFAF} = 0.5 – 0(indeterminate) – 1(0.5)
Δ_{CFA} = 0

Splitting C = F child on B:

Contingency table:

	B = T	B = F
Class value = +	25	0
Class value = -	0	25

Error rate at C = F child for B = T, CE_{CFBT} = 1 – Max (25/25, 0/25) = 1 – 1
CE_{CFBT} = 0

Error rate for B = F, CE_{CFBF} = 1 – Max (0/25, 25/25) = 1 – 1
CE_{CFBF} = 0

Gain using B for split, Δ_{CFB} = CE_{CFPS} – (25/50) CE_{CFBT} – (25/50) CE_{CFBF} = 0.5 – 0.5(0) – 0.5(0)
Δ_{CFB} = 0.5

At C = F, gain achieved by splitting using B is higher. **Hence B is used for partitioning**. In this case too, there are no misclassified instances, obviating the need for further splits.

(v) Use the results in parts (c) and (d) to conclude about the greedy nature of the decision tree induction algorithm.

Generally, decision tree algorithm grows the classifiers with a greedy approach by making locally optimal decisions. With the given dataset, the algorithm would favour splitting the root by attribute A, since it gives the best gain at that level. But as we have seen from the results of (iii) and (iv), splitting root by A has a high error rate while splitting the same by C gives a better error-less decision tree. Hence we can conclude that the **greedy nature of decision tree induction algorithm does not always result in the best decisions or trees**.