# CIS 419/519: Homework 1

Rutuja Moharil

February 8, 2019

Although the solutions are entirely my own, I consulted with the following people while working on this homework: {G.Sai,Gauri Pradhan, Sagar}

## 1 Decision Tree Learning

1. play = yes = $p_+ = \frac{9}{14}$
   play = no = $p_- = \frac{5}{14}$

   For outlook at the root node , we need to see entropy associated with outlook = sunny,rain,overcast first
   For outlook = sunny , there are play =yes =2 and play = no = 3

$$\text{play} = \text{yes} = p_+ = \frac{2}{5}$$
$$\text{play} = \text{no} = p_- = \frac{3}{5}$$
$$Entropy = -p_+ log(p_+) - p_- log(p_-)$$

   For (outlook = overcast) , there are play =yes =4 and play = no = 0

$$E(outlook = overcast) = -\frac{1}{1}log(1) - 0 = 0$$

For outlook = rainy , there are

play =yes =3 and play = no = 2

$$E(outlook = rain) =$$

$$-\frac{3}{5}log\frac{3}{5} - \frac{2}{5}log\frac{2}{5} = 0.971$$

$$\text{play} = \text{yes} = p_+ = \frac{2}{5}$$

$$\text{play} = \text{no} = p_- = \frac{3}{5}$$

$$I(S, Outlook) = \sum_i \frac{|S_i|}{|S|}.E(S_i)$$

For the case of outlook

Average entropy of outlook = $I(Outlook) =$

$$= \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.693$$

$$E(S) = -\frac{9}{14}log\frac{9}{14} - \frac{5}{14}log\frac{5}{14} = 0.940$$

$$Gain(S, Outlook) = E(S) - I(S, Outlook)$$

$$InfoGain(Outlook) = 0.940 - 0.693 = 0.247$$

Similarly for humidity let's calculate the entropy

For Humidity $> 75$ we have 9 samples

Out of the 9 samples let's see the entropy

$$\text{play} = \text{yes} = p_+ = \frac{5}{9}$$

$$\text{play} = \text{no} = p_- = \frac{4}{9}$$

$$Entropy = -p_+log(p_+) - p_-log(p_-)$$

$$E(Humidity > 75) = -\frac{5}{9}log\frac{5}{9} - \frac{4}{9}log\frac{4}{9}$$

$$= 0.991$$

For Humidity $\leq 75$ we have 5 samples

$$\text{play} = \text{yes} = p_+ = \frac{4}{5}$$

$$\text{play} = \text{no} = p_- = \frac{1}{5}$$

$$E(Humidity \leq 75) = -\frac{4}{5}log\frac{4}{5} - \frac{1}{5}log\frac{1}{5}$$

$$= 0.721$$

We already calculated E(S)

$$E(S) = 0.940$$

$$Gain(S, Humidity) = 0.940 - I(Humidity) = 0.940 - \frac{9}{14} * 0.991 + \frac{5}{14} * 0.721 = 0.045$$

$$InfoGain(Humidity) = 0.0455$$

2.

Gain Ratio is defined as

$$Ratio(S, A) = \frac{\text{Gain(S,A)}}{\text{Split in Information(S,A)}}$$

$$SplitInfo(S, A) = -\sum_i \frac{|S_i|}{|S|}.log\frac{|S_i|}{|S|}$$

$$SplitInfo(Outlook) = \frac{5}{14} * E(Sunny) + \frac{4}{14} * E(overcast) + \frac{5}{14} * E(Rainy)$$

Taking values of the entropy from part (a) we get the following

$$SplitInfo(S, Outlook) = \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.693$$

$$GainRatio(Outlook) = \frac{0.247}{0.693} = 0.355$$

Similarly for Humidity $SplitInfo(S, Humidity) = \frac{9}{14} * 0.991 + \frac{5}{14} * 0.721 = 0.895$

$$GainRatio(Humidity) = \frac{0.045}{0.895} = 0.0502$$

3. To show the class predictions and draw the complete tree we would need to decide on some parameter. Information gain is basically a parameter that helps us decide which attribute should be chosen to cause the split.
The attribute with the maximum Information gain would be selected to make the split.
Let's check for the cases of temperature and wind.

Considering the wind parameter we notice either windy is true or false . So this is an easier case for splitting.
For windy as true, there are play =yes =3 and play = no = 3

$$\text{(play = yes)} = p_+ = \frac{3}{6}$$

$$\text{(play = no)} = p_- = \frac{3}{6}$$

$$E(windy = true) = -\frac{3}{6}log\frac{3}{6} - \frac{3}{6}log\frac{3}{6} = 1$$

For windy as false, there are play =yes =6 and play = no = 2

$$\text{play = yes} = p_+ = \frac{6}{8}$$

$$\text{play = no} = p_- = \frac{2}{8}$$

$$E(windy = false) = -\frac{6}{8}log\frac{6}{8} - \frac{2}{8}log\frac{2}{8} = 0.830$$

$$InfoGain(S, Windy) = E(S) - I(Windy) = 0.940 - \frac{6}{14} * E(windy = True) + \frac{8}{14} * E(Windy = False)$$

$$InfoGain(S, Windy) = 0.940 - 0.89 = 0.048$$

Considering the temperature which is a continuous valued parameter
What threshold based attribute should be defined

On sorting we get change of value of play tennis at

$$temp < 70, 70 \leq temp \leq 75 and > 75$$

For $< 70$ play =yes =3 and play = no = 1

$$E(temp < 70) = -\frac{3}{4}log\frac{3}{4} - \frac{1}{4}log\frac{1}{4}$$

For 70≤temp≤75, there are play =yes =4 and play = no = 2

$$E(70 \leq temp \leq 75) = -\frac{4}{6}log\frac{4}{6} - \frac{2}{6}log\frac{2}{6}$$

For $> 75$ play =yes =2 and play = no = 2

$$E(temp > 75) = -\frac{2}{4}log\frac{2}{4} - \frac{2}{4}log\frac{2}{4}$$

$$InfoGain(S, Temperature) = E(S) - I(temp)$$

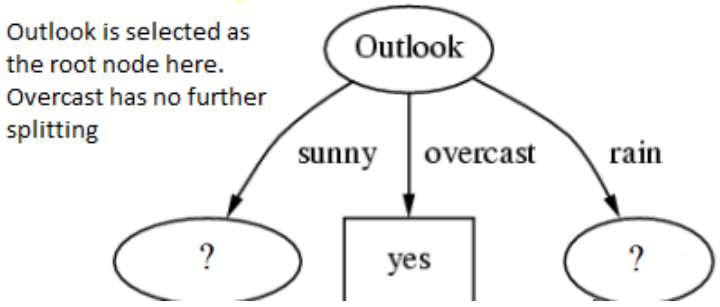
$$= 0.940 - \frac{4}{14} * E(< 70) + \frac{6}{14} * E(70 \leq temp \leq 75) + \frac{4}{14} * E(> 70)$$

$$InfoGain(S, Temperature) = 0.940 - 0.91 = 0.03$$

So as we can see we have the following values

| Information gain | Values |
|---|---|
| Outlook | 0.267 |
| Humidity | 0.0455 |
| Temperature | 0.03 |
| Windy | 0.0048 |

So we can see that Outlook has the highest Information gain.
This just shows that Outlook attribute can be selected as the root node.So basically our parent node is Outlook.
So our next step is splitting the parent node into Humidity,Temperature and Windy .
One thing to notice is that we have overcast which has yes to play outside. So we can definitely conclude that overcast does not require any splitting.




(34).png
(Image courtersy : http: //www.ke.tu-darmstadt.de/lehre/archiv/ws0809/mldm/dt.pdf)

Now next step is to think about the further split.Let's think of placing an attribute below sunny .

For Humidity below sunny
We observe that for humidity $> 75$ there is only don't play
Wheareas for humidity $< 75$ there is only play. So there are no impurities

In terms of impurities in entropy there are no such impurities for Humidity attribute under Sunny
We know that for a entropy of a group where all examples belong to the same class is $-1log(1) = 0$e

$$(\text{play} = \text{yes}) = p_+ = \frac{2}{5}$$
$$(\text{play} = \text{no}) = p_- = \frac{2}{5}$$
$$InfoGain(Sunny, Humidity) = E(Sunny) - I(Humidity)$$
$$= 0.971 - \frac{2}{5} * 0 + \frac{3}{5} * 0 = 0.971$$

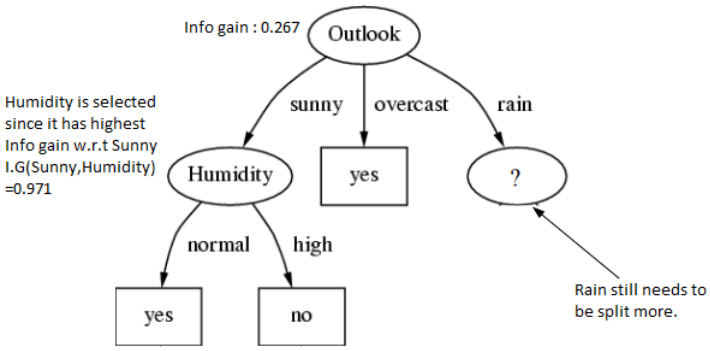
Similarly carrying out the process we get the Information gain for temperature below sunny as

$$E(temp < 70) = -1log(1)$$
$$E(70 < temp > 75) = -0.5log(0.5) - 0.5log(0.5)$$
$$E(temp > 75) = -1log(1)$$
$$InfoGain(Sunny, Temperature) = E(Sunny) - I(Temperature)$$
$$= 0.971 - \frac{1}{5} * 0 + \frac{2}{5} * 1 + \frac{2}{5} * 0 = 0.571$$

Similarly carrying out the process we get the Information gain for windy below sunny as
In case of windy we notice that for true condition there is play and don't play .
As we saw above for equal impurity the entropy is 1.

$$E(windy = false) = -\frac{1}{3} * log\frac{1}{3} + \frac{2}{3} * log\frac{2}{3}$$
$$E(windy = true) = -0.5log(0.5) - 0.5log(0.5) = 1$$
$$InfoGain(Sunny, Windy) = E(Sunny) - I(Windy)$$
$$= 0.971 - \frac{2}{5} * E(true) + \frac{3}{5} * E(false) = 0.01$$

Comparing the values of Information gain from above we realise that humidity is the best fit below sunny.

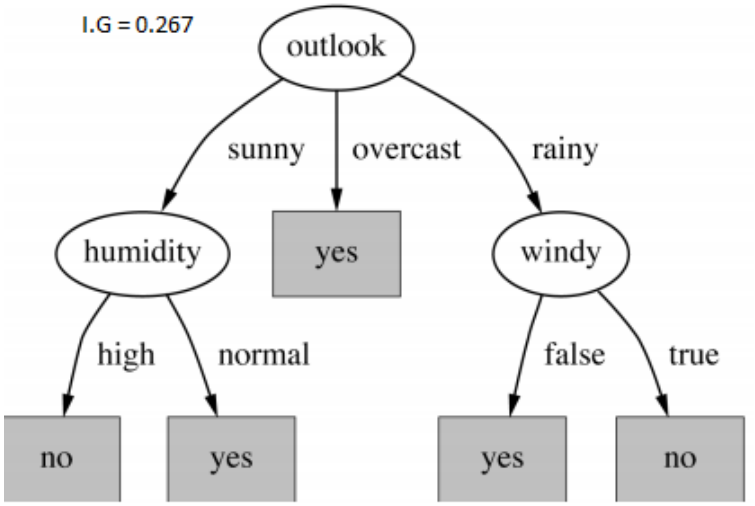


(35).png

We are only left with rain attribute. Let's check temperature and windy for it.
For windy under rainy we get entropies without any impurities for windy as false and windy as true repsectively.

$$InfoGain(Rainy, Windy) = 0.971 - \frac{2}{5} * 0 - \frac{3}{5} * 0 = 0.971$$
$$InfoGain(Rainy, Temperature) = 0.971 - somevalue < 0.971$$

So for temperature we could easily check that because there are impurities the info gain is definitely lesser than Info gain(Rain,Windy).
So we found windy below the rainy leaf.



(38).png

4. No .The ID3 algorithm does not guarantee a globally optimum decision tree. It follows a greedy approach and hence can converge at local optima.
   It gives locally the best solution because it selects the best attributes locally . Due to this local selection of attributes during each iteration we call this a greedy approach
   Backtracking can improve the optimality , however it comes at the cost of time complexity issues .
   All in all because of the greedy heuristic approach followed by ID3 algorithm usually produces small trees, but it does not always produce the smallest possible decision tree.

## 2 Decision Trees & Linear Discriminants [CIS 519 ONLY]

A decision tree can include oblique splits by... So oblique trees comes up when the decision boundary in a feature space is not aligned to the feature axis.
This would lead to complicated boundary structures and hence normal decision tree algorithms When the true decision boundaries are not aligned with the feature axes, this approach can produce a complicated boundary structure. So we use other methods since our normal decision tree algorithm becomes computationally expensive

As the name suggests Oblique decision trees use oblique decision boundaries to simplify this predicament of complicated (oblique) boundaries.

One way of implementing such a method is that we accomodate minimum cases of impurity by finding a optimal separating hyperplane can be called as a decision surfaces.The major limitation of this approach is that it is computationally expensive.

So one way is to keep sloping the decision surface to accomodate for minimum impurities. Another method

is that consider we have an oblique decision surface in 2D plane ,now consider the oblique decision surface as the feature axis and then form a new decision surface w.r.t to the new axes. So basically converting an oblique surface to an axis which is a axis parallel split in itself.

This approach was present in a HHCART algorithm which can be found : https://arxiv.org/abs/1504.03415
In this approach they mention about using a series of householder matrices to reflect the training data at each node. Basically forming the new axes (which is an axis parallel split in itself)at these nodes and the direction of reflection comes from the eigenvectors.

So a reflected axis parallel split in training data makes it easier to findn oblique spliton test data .(Note :In a system the eigenvectors gives the direction and eigenvalues give the respective scaling)
Problem of constructing an oblique surface is Np hard problem .

# 3 Programming Exercises

**Features**: Following is the list of all classifiers used and the features given to them :

1. Decision Tree: So the features that are given to the decisiontreeclassifier are the features computed using the computefeatures function
   So each instance or name is denoted by X and we calculate the features corresponding to each instance.
   As mentioned in the question we will have boolean features one for each letter in the alphabet .
   So essentially we have $5 \times 3 \times 26 = 390$ features for each name . Thus every instance has 390 features and these 390 features and we get a a feature matrix of dimensions mx390 because m represents the folds used for cross validation . In our case it was $m = 5$ and hence the feature matrix was $5 \times 390$ dimension passed to the Decision tree classifier.
   We also used the same feature matrix for a decision tree of depth 4 and 8 .

2. SGD : So the features that are given to the SGDclassifier are the same set of features given to the decision tree classifier
   Each instance is X and is every X has 390 features as mentioned above. The same feature matrix dimension holds for the SGD as $5 \times 390$ dimension where 5 denotes the 5 fold cross validation

3. SGD with Stumps : So this case is a bit different . Unlike the above classifiers where we computed features and gave them to classifiers ,using the feature set that we obtained (like in above cases) we train 200 different decision trees of maximum depth 8 on the entire training set.
   Then, these decision trees are used to generate 200 features for the data set. Hence creating a new feature matrix X where X[i][j] represents i as the instance number (basically ith name) and X[i][j] is the prediction of decision tree j on instance i ,using the original feature set (both 0-indexed).
   This new feature matrix with dimensions $200 \times 560$ 560 from 4 training set folds and 200 for the 200 number of stumps, is used to train an SGD classifier, which will be the final model.

**Parameters**: description of the parameters you used for each classifier, such as the learning rate and error threshold.

1. SGD classfier : Here the parameters given were loss=log and learning rate = 'optimal'. The other additional parameters that were used/experimented with were the alpha and tol values .
   Depending on the alpha and tol values the prediction accuracies displayed some variations .
   Varying alpha while keeping tol at the default value 1e-3

| $p_A$ | alpha | tol |
|---|---|---|
| 0.68 | 0.005 | 1e-3 |
| 0.6599 | 0.001 | 1e-3 |
| 0.625 | 0.0001 | 1e-3 |
| 0.59 | 0.00001 | 1e-3 |

Varying tol while keeping alpha at the default value 0.0001

| SGD $p_A$ values | alpha | tol |
|---|---|---|
| 0.65 | 0.0001 | 1e-2 |
| 0.6285 | 0.0001 | 1e-3 |
| 0.62 | 0.0001 | 0.00008 |
| 0.615 | 0.0001 | 1e-4 |

So with increasing alpha and tol values $p_A$ increases. However the increase is observed only upto a certain alpha values. This is due to the fact that higher learning rate might cause the algorithm to diverge or overshoot.
(Included observations on alpha in comment sections)
The reason for keeping loss='log is because this setting gives linear regression which is a probability classifier.

2. Decision tree : For the decision tree classifier we have criterion = entropy and depth = 0 ,4,8. So there were 3 types of trees in this exercise . The depth 4 tree, depth 8 tree and unpruned tree. For all the three type the criterion = entropy was same .
   I experimented with the additional parameter which was the random state . So I kept the random state = 0 and after every run I got the same result i.e the value of tets acccuracy.
   SGD with stumps : For this particular classifier the parameters used were those of SGD classifier and decisison tree . To be specific loss= log,criterion=entropy ,learning rate = optimal and depth = 8

I have combined the ranking and confidence interval data for better visualization
The ranking of all of the classifiers according to their $p_A$ values:

| Algorithm | $tr_A$ | $p_A$ | $p_A$ Conf. Interval [519 ONLY] |
|---|---|---|---|
| SGD + Decision Stump Features | 0.7899 | 0.6691 | (0.5587,0.77729) |
| Decision Tree - Height 8 | 0.7310 | 0.6671 | (0.600,0.73316) |
| SGD | 0.8021 | 0.6599 | (0.56477,0,75503) |
| Decision tree -Height 4 | 0.6928 | 0.65285 | 0.5839-0.73206 |
| Decision tree -unpruned | 0.8382 | 0.6428 | (0.5482,0.7374) |

1. The $p_A = 0.6691$ value of SGD-DT stumps lies in Decision tree -8 confidence interval . Thus this pair of SGD-stumps and Decision tree-8 is not statistically significant.

2. The $p_A = 0.6671$ value of Decision tree-8 stumps lies in SGD confidence interval . Thus this pair of SGD and Decision tree-8 is not statistically significant.

3. The $p_A = 0.6599$ value of SGD lies in Decision tree-4 confidence interval . Thus this pair of SGD and Decision tree-4 is not statistically significant.

4. The $p_A = 0..65285$ value of Decision tree-4 lies in Decision tree-unpruned confidence interval . Thus this pair of Decision tree-4 and Decision tree-unpruned is not statistically significant.

For performing the t-test , I implemented it on the spyder IDE by import the ttest function. After ranking the classifiers according to $p_A$ values we check the t-test for every pair. The threshold value for comparison of the obtained p from the ttest is 0.01.
So I compared SGD+Decision stumps Vs DT -height 8 ,Dt-8 Vs SGD, SGD Vs DT-4,DT-4 Vs Unpruned DT.
And in all the cases the output values from the function were higher than 0.01. This shows that it is statistically significant.
A p-value is the probability that the results from your sample data occurred by chance. Low p-values are good,they indicate your data did not occur by chance. I refered Towards Data science blog (Medium blogs) for this concept.

Thus the accuracy results are correct.

**Conclusion**:
SGD + stumps should be the best classifier . This is because we are computing 200 stumps and training it on randomized data which made me thought that this way it would be a better trained model.
Also even after feature extraction we generate new features for the stumps with random training data from 5 fold cross validation and thus it appealed to me that probably this would prove to be a better model .
Comparing the $p_A$ and $tr_A$ values of all other SGD with stumps has better $p_A$ values .
On $tr_A$ Vs $p_A$ we can see that training accuracy is higher. We actually change the hyperparameters to tune the testing accuracy. So the tuning of alpha and tol is done to improve the testing accuracy.

**Comments**:

1. After some playing around with the parameters I noticed that after increase in a certain value of alpha (which determines the learning rate for learning rate ='optimal' condition) the $p_A$ decreases .

2. Well this made sense since I could relate the fact that if alpha becomes very large it may diverge or overshoot the minimum (in terms of gradient descent).
   I expected that SGD+stumps would demonstrate the highest $p_A$ and I got the result with $p_A = 0.669$ (As stated in algorithm ranking table) however for me Decision tree - 8 also gave comparable values but slightly lesser than SGD with stumps and hence proving SGD stumps to be the best in terms of $p_A$ .

3. The decision tree (Id3) algorithm prefers a smaller depth tree due to its greedy heuristic approach. So we prefer smaller trees to give us better results.
   I observed that training accuracy for the decision tree unpruned was the highest and lowest for the depth 4 tree . This is obvious because the unpruned tree has several nodes and thus this means it splits considering more features.