*CSci 5525: Machine Learning*

# Homework 2

*Name:* Anuradha Uduwage (Anu) *Student ID:* ***0427

**Question 1**

---

(i) The approach behind decision stump is that you identify the best feature to split and carry out the split and stop. This can be done using a binary split or categorical split. Here I have used categorical split. Using categorical split you end up with a breath wise tree respect to depth wise tree.

**Steps:**

we use ten fold to split the data in to 10 fold. Each fold will have its training data and test data. *GetTenFold* function provide 10 folds to any give data set.

In each fold we identify feature to split by calculating the information gain. Information gain can be calculate by using the entropy function.

$$H(X) = \sum_{i=1}^{n} -p(x_i) log_2 p(x_i)$$

Once we have the entropy, we then calculate the conditional entropy which is basically the entropy of data given the overall class entropy. We can calculate that by using following function.

$$H(X|Y) = \sum_{j=1}^{m} p(y_j) H(X|y_j)$$

At this point we have both entropy and the conditional entropy for a given feature, then we need to find the information gain to identify if this is a good split or not. What it means by good split is that using the as a root will the data will be assign to pure class or close to pure as possible. For information gain

$$IG(X|Y) = H(X) - H(X|Y)$$

**Categorical Split**

In this approach, I have used categorical split to do the *Decision Stump*

Feature $5$ has been select as the best split with unique values, $1, 2, 3, 4, 5, 6, 7, 8, 9$. Base on this split we have Feature 5 as the root and its unique values as the branches. They have they labels as, $1, -1, -1, -1, 1, 1, -1, -1$. Based on the image, You can see clearly why the feature 5 was taken as the best split. Because among 21 features feature 5 has the highest information gain.

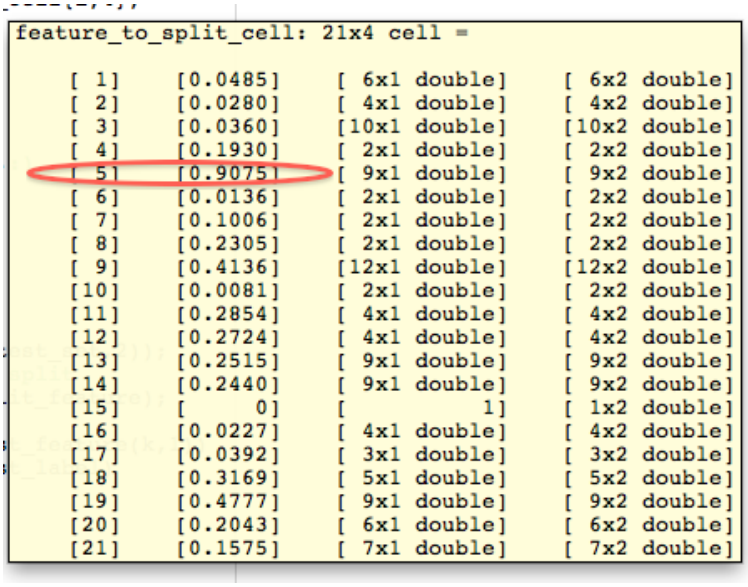
Single and complete hierarchical clustering, and dendrogram.

**Notes** We were told that feature 11 has missing values, there are two ways to handle this. I have removed feature 11 prior 10 fold creation due to this you will see 21 features not 22 features.

I have use tabulate function in my entropy function. This is a easy way to build the contingency table. Given a vector this function will give use percentage of existence of data in that vector.

**Error Rates for Decision Stump**

(ii) To try a different approach, I used binary split for this, since we are have to print out the tree structure clearly it made more sense to do the binary split for this. Mat lab has a great function call *treeplot*. I have use both treeplot function and sprintf to clearly print the tree in a human readable fashion. When



feature_to_split_cell: 21x4 cell

| | | | |
|---|---|---|---|
| [ 1] | [0.0485] | [ 6x1 double] | [ 6x2 double] |
| [ 2] | [0.0280] | [ 4x1 double] | [ 4x2 double] |
| [ 3] | [0.0360] | [10x1 double] | [10x2 double] |
| [ 4] | [0.1930] | [ 2x1 double] | [ 2x2 double] |
| [ 5] | [0.9075] | [ 9x1 double] | [ 9x2 double] |
| [ 6] | [0.0136] | [ 2x1 double] | [ 2x2 double] |
| [ 7] | [0.1006] | [ 2x1 double] | [ 2x2 double] |
| [ 8] | [0.2305] | [ 2x1 double] | [ 2x2 double] |
| [ 9] | [0.4136] | [12x1 double] | [12x2 double] |
| [10] | [0.0081] | [ 2x1 double] | [ 2x2 double] |
| [11] | [0.2854] | [ 4x1 double] | [ 4x2 double] |
| [12] | [0.2724] | [ 4x1 double] | [ 4x2 double] |
| [13] | [0.2515] | [ 9x1 double] | [ 9x2 double] |
| [14] | [0.2440] | [ 9x1 double] | [ 9x2 double] |
| [15] | [ 0] | [ 1] | [ 1x2 double] |
| [16] | [0.0227] | [ 4x1 double] | [ 4x2 double] |
| [17] | [0.0392] | [ 3x1 double] | [ 3x2 double] |
| [18] | [0.3169] | [ 5x1 double] | [ 5x2 double] |
| [19] | [0.4777] | [ 9x1 double] | [ 9x2 double] |
| [20] | [0.2043] | [ 6x1 double] | [ 6x2 double] |
| [21] | [0.1575] | [ 7x1 double] | [ 7x2 double] |

Figure 1: Best Split for Decision Stump

we convert from category to binary tree our best split feature will change. I have use following function to find the best split of the tree and each branch is clearly label the splits.
The function build tree, Builds a decision tree to predict Y from X. The tree is grown by recursively splitting each node using the feature which gives the best information gain until the leaf is consistent or all inputs have the same feature values. X is an nxm matrix, where n is the number of points and m is the number of features. Y is an nx1 vector of classes cols is a cell-vector of labels for each feature. Since our labels are actually number of the feature it would be values of features from 1-21.
**Steps to Run**
Switch to folder binary_split prior to running the code and execute the myDtree('Mushroom',2,10), the second variable will indicate the level of the tree, based on this recursion will stop doing the split once it reaches the depth given to the function.

(iii) Ass shown in the figure, ** indicate that we can further split these branches, but we stop due the the recursion. In the binary split 9 as the best feature to split. then 19, and 8 as next best splits. Unlike Categorical split we have to convert our data into binary format to get the actual split. This can be done multiple ways, but my approach was following. We identify the unique values in a given feature {*These would be our branches*}, then using following formula I defined the split, *splits = 0.5*(vals(1:end-1) + vals(2:end))* here vals are the unique values of a feature. This will give me the split point. *R*efer to the image:

** indicator inform that the classes are not yet pure, at level two from the binary split we are not getting pure classes at all leaf. But at level 5 we can see pure classes at all leaf nodes. **Please refer to the image.**
What I have notice is that doing binary split has a performance issue, initially I started with the categorical split, during the debug process I found out that categorical splits are much faster process that binary split. This might be due to the fact that we need to identify the split points and convert that to make binary. I am using repmat. This might have performance issues.
Each iteration level, the decision tree will generate a new image.

```
>> myDstump('Mushroom', 10)
10 fold: Feature to split and error rate
ans =

    5.0000    0.0160
    5.0000    0.0160
    5.0000    0.0074
    5.0000    0.0184
    5.0000    0.0258
    5.0000    0.0160
    5.0000    0.0123
    5.0000    0.0135
    5.0000    0.0135
    5.0000    0.0098

>>
```

Figure 2: Best Split for Decision Stump

**Question 2**

---

(a) Please see the written answer at the end of the report

**Question 3**

---

(a) Derive the corresponding additive model using the logistic loss function $L(y, f(x)) = log(1+exp(-yf(x)))$

(b) **logit boost**

(c) **Note**
Initially when I started this project, I planned on doing all implementation using categorical split. But half way through I realize categorical split actually is not the best approach for boosting. To be boost you need to start with the week learner but using categorical split I get a 99% accuracy. In the slides the graph shows that week learner has started with 50% error. I initially implemented using categorical and due to time constrain didn't have time to switch it back to the binary split. If it was the binary split looking at my implementation for the part 2 and part 3 of the first question, it is possible to the stump of binary split would start with the weak learner. Nevertheless, I have implemented the adaboost algorithm as follows.

**Steps**
Initialize $w_1(i) = 1/N$ (N , total number of points.)
for $t = 1...T$ {T total number of stumps
Train weak learner using distribution $W_t$
Get week hypotheisis $G_t$ with error $\epsilon_t = Pr_x\ w_t[G_t(x) \neq y]$
choose $\alpha_t = 1/2ln(1-\epsilon_t)/\epsilon_t$
update
and use the sign of the classifier to get the classification.

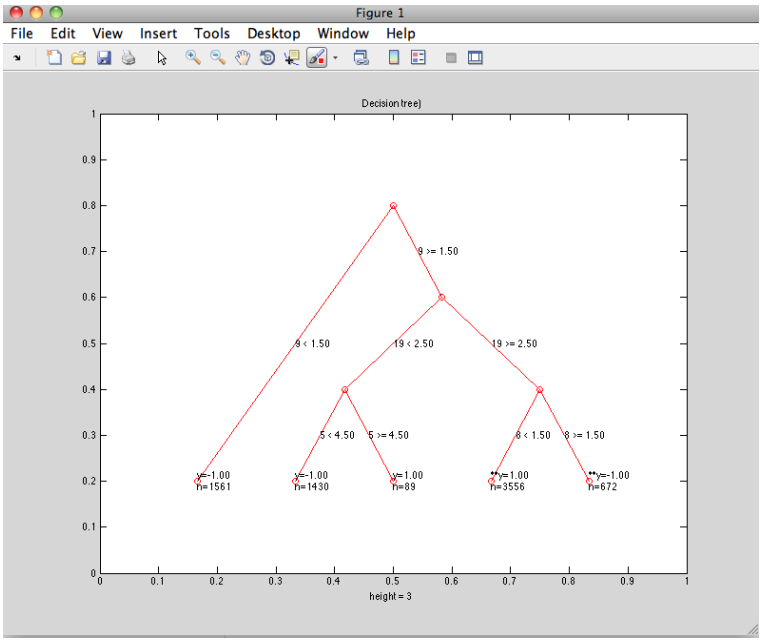



Figure 3: Level 2 Decision Tree

AdaBoost code should execute by the following command
*myAdaBoost('Mushroom',10,15)* second variable is for fold and 3rd variable is for number of stumps. What we do in boosting is that we use a update function to update all the points of the training data, and use the sign function as a classifier. If we predict correctly then weights get lower and if we predict wrong weights increse.

(d) As shown in the proof, Doing minimization on regression is same as minimizing log loss function (Please refer to the hand written proof). This way we get the following update function.

$$W_{t+1}(x_i) = W_t(x_i) * 1/1 + exp(y_i, F_t(x_i))$$

The class value that we get, then send through a sigmoid function because we need to get a logistic loss, which mean this is no longer a classification problem but a regression problem. The Boosting Approach to Machine Learning paper actually mention this clearly.

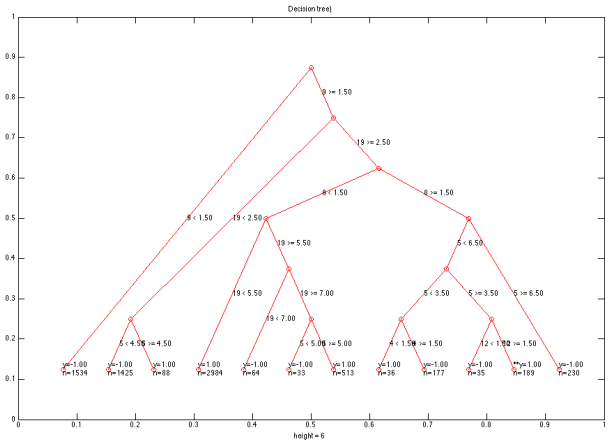



Figure 4: Level 5 Decision Tree

| Training Error | Test Error | Training Error | Test Error | Training Erro | Test Error | Training Error |
|---|---|---|---|---|---|---|
| 0.0146 | 0.016 | 0.0145 | 0.0172 | 0.0149 | 0.0135 | 0.015 |
| 0.015 | 0.0123 | 0.0149 | 0.0135 | 0.0155 | 0.0086 | 0.0145 |
| 0.0148 | 0.0148 | 0.0146 | 0.016 | 0.0144 | 0.0185 | 0.0141 |
| 0.0142 | 0.0197 | 0.0153 | 0.0098 | 0.0145 | 0.0172 | 0.0157 |
| 0.0148 | 0.0147 | 0.0149 | 0.0135 | 0.0156 | 0.0074 | 0.0148 |
| 0.015 | 0.0123 | 0.0145 | 0.0172 | 0.0146 | 0.016 | 0.0142 |
| 0.0142 | 0.0196 | 0.015 | 0.0123 | 0.0148 | 0.0147 | 0.0149 |
| 0.0148 | 0.0147 | 0.0146 | 0.016 | 0.0142 | 0.0196 | 0.0144 |
| 0.0156 | 0.0074 | 0.0144 | 0.0184 | 0.0148 | 0.0147 | 0.0153 |
| 0.0145 | 0.0172 | 0.0149 | 0.0135 | 0.0145 | 0.0172 | 0.0148 |
| 5Stump | 5Stump | 10Stump | 10Stump | 15 Stumps | 15 Stumps | 20 Stumps |
| 0.01475 | 0.01487 | | 0.01474 | 0.01478 | 0.01474 | 0.01477 |

Test Error

0.0123
0.0172
0.0209
0.0061
0.0147
0.0197
0.0135
0.0184
0.0098
0.0147

20 Stumps

0.01473

(a) $\hat{y}(w) = \frac{1}{1+\exp(-w^T x)}$ & $E(w) = (y - \hat{y}(w))^2$

What we need to prove is that there exist A, such that $v^T A v \geq 0$ — (*)

$E(w) = (y - w^T x)^2$ dimensionality of these matrices are $d \times 1$

$d \times 1$ $d \times 1$

If we can prove such that $\nabla_w^2 E(w)$ is positive semi definite, which is actually a property of (*), then we have prove that error function is convex in w.

$$E(w) = (y - w^T x)^2$$
$$= (y - w^T x)^T \cdot (y - w^T x)$$
$$= (y - w^T x)(y - w^T x)^T$$
$$= (y - w^T x)(y^T - x^T w)$$ we do expansion

take double derivative

$$\nabla_w^2 = \{ y y^T - y x^T w - w^T x y^T + w^T x x^T w \}$$

$= 0$

$$\nabla w^2 = (w^T x \cdot x^T w)$$

Lets assume that $X X^T = B$

then we have

$$\nabla_w \left( (B + B^T) w \right)$$

↳ respect to Matrix Cook Book we can write this as.

↳ $(w^T x x^T w) = (B + B^T) w$

$B^T = (X X^T)^T$

$= X X^T$

$\therefore \nabla_w = B + B^T$

$= X X^T + X^T X$

$= 2 X X^T$

This is in the form of

$V^T A V \geq 0$

$\therefore$ $V^T X X^T V$ we can rewrite this as

$(X^T V)^T (X^T V)$

$y^T y \geq 0$

$\therefore$ this is convex and it positive Semi definite.

$\therefore$ it is Convex.

(2(b)) $E(w) = y \log \frac{y}{y(w)} + (1-y) \log \frac{1-y}{(1-\hat{y}(w))}$

$$y \log \frac{y}{\left(\frac{1}{1+e^{-w^T x}}\right)} + (1-y) \log \frac{(1-y)}{\left(\frac{1-y}{1+e^{-w^T x}}\right)} \quad \text{--- (*)}$$

↑ we get this by substituting

$\hat{y}(w) = 1/1+\exp(-w^T x)$

(*) by log tranformation.

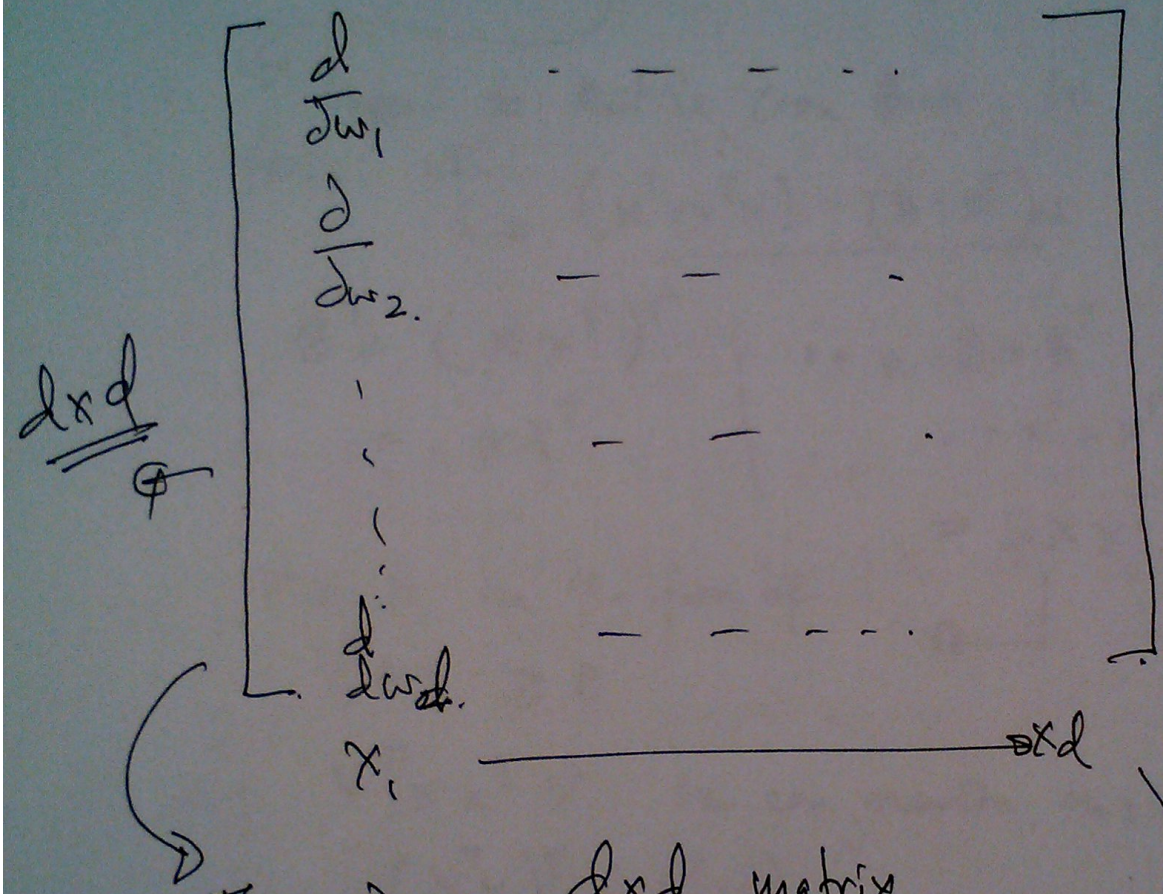
$$= y \log y - y \log\left(\frac{1}{1+e^{-w^T x}}\right) + (1-y)\log(1-y) + (1-y)\log\left(\frac{e^{-w^T x}}{1+e^{-w^T x}}\right)$$

+ y lets keep $y \log y$ and $(1-y)\log(1-y)$ as it is and do log tranformation for the rest of equation.

$$= y \log y - y \log\left(\frac{1}{1+e^{-w^T x}}\right) + \frac{\log e^{-w^T x}}{(1+e^{-w^T x})} - y \log\left(\frac{e^{-w^T x}}{1+e^{-w^T x}}\right)$$

$$y \log(1+e^{-w^T x}) + \frac{\log e^{-w^T x}}{1+e^{-w^T x}} - y\left[\log(e^{-w^T x}) - \log(1+e^{-w^T x})\right]$$

Now we take the derivation of this respect to $w$:




This is a dxd matrix

what is happening above is that we take the hessian and do chain Rule and update rule.

As a result of this we get this in the form of

$$A = (B^T V)^T (B^T V)$$

$A = BB^T$ ← dx1 matrix

this acts like $V^T BB^T V$.
It is positive semi definite

∴ it is convex.

Linear functions are much easier to do the overfitting.

Logistic always gives you nicer bounds and linear doesn't.

Sigmoid function is always between 0 and 1.

(3.)

(a) $L(y, f(x)) = \exp(-yf(x))$ ← Adaboost.

Logistic Regression (Bounds)

$P(Y=1|X) = \frac{1}{1+\exp(f(x))}$

$L(y, f(x)) = \log(1+\exp(-yfx))$

Derive.

and in logistic we try to maximize data likelyhood

$P(D|H) = \prod_{i=1}^{m} \frac{1}{1+\exp(-y_i f(x_i))}$ ← (*)

In Logistic Regresion we maximize this function which is

This is equivallent to minimizing log loss → $\sum_{i=1}^{m} \ln(1+\exp(-y_i f(x_i)))$ ↓

Minimizing the this function.

Lets take $f(x) = w_i + \sum_i w_i x_i$

To Prove for a point we take this functionand set class labels to y

$\log \prod_i \frac{1}{y_i} = -\sum_i \log y$

from (*) if $y_i = +1$ then $\frac{1}{1+e^{-f(x)i}}$ and if $y = -1$ then $\frac{1}{1+e^{f(xi)}}$

$= \frac{e^{-f(xi)}}{1+e^{-f(xi)}}$

I have used Logistic Regression from the Book and The boosting Approach Paper to Prove this theory.

∴ we get the update function :- $W_t(x_i) \times \frac{1}{1+\exp(\pm y_i F_t(x_i))}$