Assignment 23

Leroy Souz

15th November 2024

1 Question 1

1.1 Part 1.1

Entropy of dataset:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$
$$Entropy(S) = -\left(\frac{5}{10}\log_2\frac{5}{10} + \frac{5}{10}\log_2\frac{5}{10}\right)$$
$$Entropy(S) = 1$$

Calculating the InformationGain for every feature:
Weather: Calculating the Entropy for the Weather feature:

- Cloudy: 0
- Rainy: 0.811
- Sunny: 0.981

Weighted Entropy for Weather: 0.6
InformationGain for Weather: 1-0.6 = 0.4

Temperature: Calculating the Entropy for the Temperature feature:

- Hot: 1
- Mild: 0.97
- Cool: 0.

Weighted Entropy for Temperature: 0.885
InformationGain for Temperature: 1-0.885 = 0.115

Humidity: Calculating the Entropy for the Humidity feature:

- High: 0.985
- Normal: 0.918

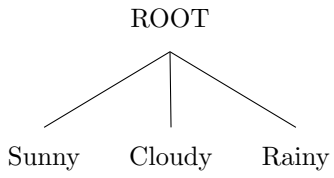
Weighted Entropy for Humidity: 0.965
InformationGain for Humidity: 1-0.965 = 0.035

Wind: Calculating the Entropy for the Wind feature:

- Weak: 0.811
- Strong: 0.918

Weighted Entropy for Wind: 0.875
InformationGain for Wind: 1-0.875 = 0.125

Based on the InformationGain, the Weather feature has the most InformationGain and is the best feature to split on.
The root node looks like:



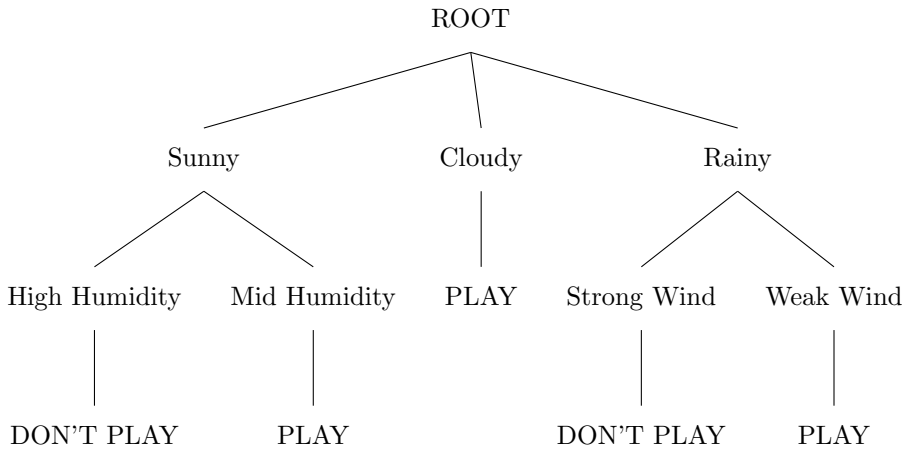
1.2 Part 1.2

The leaf node for Cloudy is pure and this doesn't need to be expanded.

Information Gain when expanding the Rainy node:
Temperature: 0.918
Humidity: 0.918
Wind: 0.252
Thus we expand using Humidity.
The leaf node for Rainy is pure and this doesn't need to be expanded.

Information Gain when expanding the Sunny node:
Temperature: 0.123
Humidity: 0.123
Wind: 0.811
Thus we expand using Wind.
The leaf node for Sunny is pure and this doesn't need to be expanded.

The final tree looks like:



1.3 Part 1.3

Let us first consider the initial C(T) for the entire tree:

$$C(T) = \sum_{\tau=1}^{T'} Q(\tau) + \lambda \cdot |\text{num of leaves in } T'|$$
$$C(T) = 0 + \lambda \cdot 5 = 5\lambda$$

When we prune the Humidity sub-tree, the cost function becomes:

$$C(T) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) + \lambda \cdot |\text{num of leaves in } T'|$$
$$C(T) = 0.918 + \lambda \cdot 4 = 4\lambda + 0.918$$

When we prune the Wind sub-tree, the cost function becomes:

$$C(T) = -\left(\frac{1}{4}\log_2\left(\frac{27}{256}\right)\right) + \lambda \cdot |\text{num of leaves in } T'|$$
$$C(T) = 0.811 + \lambda \cdot 4 = 4\lambda + 0.811$$

now we are only left with the root whose cost function is:

$$C(T) = -\left(\frac{5}{10}\log_2\left(\frac{5}{10}\right) + \frac{5}{10}\log_2\left(\frac{5}{10}\right)\right) + \lambda \cdot |\text{num of leaves in } T'|$$
$$C(T) = 1 + \lambda \cdot 1 = \lambda + 1$$

We can see that as lamba goes up, it starts to penalize the leaf count more.
At $\lambda = 0$, the best tree is the un pruned but when $\lambda = 0.25$ the best tree is the root node only.

2 Question 2

2.1 Part 2.1

When w_0 is initialized as (0,0) with a step size of 1, the algorithm will converge in 1 iteration. We can prove this by the following:
If the weights are initialized as 0, then $w_1x_1 + w_2x_2 \leq 0$ and so the algorithm classifies the sample as -1 which is incorrect. Thus it will update the weights using:

$$w_t + 1 = w_t + \eta y_t x_t$$
$$w_t + 1 = (0,0) + (x_1, x_2) = (x_1, x_2)$$

Now on the next update, $w_1x_1 + w_2x_2 = x_1^2 + x_2^2 > 0$ (assuming x_1, x_2 are not 0) and so the algorithm classifies the sample as +1 which is correct and no further updates are needed.

2.2 Part 2.2

When w_0 are randomly initialized, there can be 2 cases:
Case 1: The weights are initialized such that $w_1x_1 + w_2x_2 > 0$. In this case no further updates are needed and the algorithm converges without any iteration
Case 2: The weights are initialized such that $w_1x_1 + w_2x_2 \leq 0$. In this case the algorithm the algorithm will perform the following weight update:

$$w_t + 1 = w_t + \eta y_t x_t$$
$$w_t + 1 = (w_1, w_2) + (x_1, x_2) = (w_1 + x_1, w_2 + x_2)$$

Now on the 2nd iteration if $w_1x_1 + w_2x_2 <= 0$ then the algorithm will update the weights again

$$w_t + 1 = w_t + \eta y_t x_t$$
$$w_t + 1 = (w_1 + x_1, w_2 + x_2) + (x_1, x_2) = (w_1 + 2x_1, w_2 + 2x_2)$$

This will go on for say n iterations. The algorithm will keep updating the weights till $w_1x_1 + w_2x_2 > 0$ and so the weights will be $(w_1 + nx_1, w_2 + nx_2)$. We can write the convergence rule as:

$$\vec{w}_n^T x > 0$$

Substituting the weights we get:

$$(w_0 + nx)^T x > 0$$
$$w_0^T x + nx^T x > 0$$
$$n > -\frac{w_0^T x}{x^T x}$$

This gives us a lower bound on the number of iterations needed for convergence.

2.3 Part 2.3

I use only mmisclassified points to update the weights.

iteration	$\vec{w}$
0	$w_0 = (0, 0)$
1	$w_1 = (0, 0) + (0, 1) = (0, 1)$
2	$w_2 = (0, 1) - (1, 0.5) = (-1, 0.5)$
3	$w_3 = (-1, 0.5) + (1, 1) = (0, 1.5)$
4	$w_4 = (0, 1.5) - (1, 0.5) = (-1, 1)$
5	$w_5 = (-1, 1) + (1, 1) = (0, 2)$
6	$w_6 = (0, 2) - (1, 0.5) = (-1, 1.5)$

3 Question 3

3.1 Part 3.1

class	mean	var
pos(+)	-0.072	1.30
neg(-)	0.94	1.94

3.2 Part 3.2

The test accuracy is 61%

3.3 Part 3.3

The MLE estimate is not a good rule in this case. MLE only considers how likely one class is compared to the other and does not take into account if one class is way more freqeuent than the other. For our data the negative class is 9x more frequent and MLE is unoptimal. We instead incorporate the prior probabilities to maka a better classifier.
Through this we get an accuracy of 90%

3.4 Part 3.4

class	mean	COV
pos(+)	[0.013, 0.063]	[-0.023, -0.021]
neg(-)	$\begin{pmatrix} 0.98285498 & 0.00612046 \\ 0.00612046 & 1.05782804 \end{pmatrix}$	$\begin{pmatrix} 1.00329037 & -0.01142356 \\ -0.01142356 & 4.97693356 \end{pmatrix}$

3.5 Part 3.5

The test accuracy is 84%

3.6 Part 3.6

The testing accuracy is 85%. There was not a big difference in the increase in accuracy between the two models. We can thus conclude that as long as the distributiuon of data is somewhat gaussian, using GDA for classification will perform well on the data

4 Question 4

4.1 Part 4.1

IF-IDF tries to capture how many times a term appears in a document with respect to all documents. It combines TF, which calculate the importanct of a word in a document, and IDF, which calculates the how unique the word is across all documents. TF-IDF gives us a score for each word that is high if the word appears many times in a document but not in many documents thus showing us important words while filtering out common words.

4.2 Part 4.2

Find code in the attached file

4.3 Part 4.3

Find code in the attached file

4.4 Part 4.4

train accuracy is 96.5% test accuracy is 97%

4.5 Part 4.5

The model classifies 'mail.txt' as spam.