AI534 — Written Homework Assignment 3 —

1. (Naive Bayes Classifier) (7 pts) Consider the following training set:

A	B	C	Y
0	1	1	0
1	1	1	0
0	0	0	0
1	1	0	1
0	1	0	1
1	0	1	1

(a) (3 pts) Learn a Naive Bayes classifier by estimating all necessary probabilities (there should be 7 independent probabilities to be estimated in total).

Below are the list of probabilities that are learned from the training data.
Class prior: $p(y = 1) = 1/2$
Class conditional probability for $y = 1$*:*
$p(A = 0|y = 1) = 1/3$; $p(B = 0|y = 1) = 1/3$; $p(C = 0|y = 1) = 2/3$
Class conditional probability for $y = 0$*:*
$p(A = 0|y = 0) = 2/3$; $p(B = 0|y = 0) = 1/3$; $p(C = 0|y = 0) = 1/3$

(b) (3 pts) Compute the probability $P(y = 1|A = 1, B = 0, C = 0)$. *Prediction for (1,0,0):*

$$p(y = 1|X) = \frac{p(y = 1) * P(A = 1|y = 1)P(B = 0|y = 1)P(C = 0|y = 1)}{P(A = 1, B = 0, C = 0)} = \frac{0.5 * \frac{2}{3} * \frac{1}{3} * \frac{2}{3}}{Z} = \frac{\frac{2}{27}}{Z}$$

To compute the normalizing factor, let's first work out the numerator for $P(y = 0|X)$*:*

$$p(y = 0|X) = \frac{p(y = 0) * P(A = 1|y = 0)P(B = 0|y = 0)P(C = 0|y = 0)}{P(A = 1, B = 0, C = 0)} = \frac{0.5 * \frac{1}{3} * \frac{1}{3} * \frac{1}{3}}{Z} = \frac{\frac{1}{54}}{Z}$$

Because $p(y = 1|X) + p(Y = 0|X) = 1$*, we must have:*

$$Z = \frac{2}{27} + \frac{1}{54} = \frac{5}{54}$$

Note that this is essentially calculating

$$P(A = 1, B = 0, C = 0) =$$

$$P(A = 1, B = 0, c = 0|y = 1) * P(y = 1) + P(A = 1, B = 0, C = 0|y = 0) * P(y = 0)$$

This gives us:

$$p(y = 1|X) = \frac{4}{5}; p(y = 0|X) = \frac{1}{5}$$

Note that if you are asked to make a MAP prediction, there is no need to calculate the normalization factor Z *since we can tell* $p(y = 1|X)$ *is greater than* $p(y = 0|X)$ *by simply comparing the numerators.*

(c) (1 pts) Suppose we know that the three features A, B and C are independent from one another, can we say that the Naive Bayes assumption is valid? (Note that the particular data set is irrelevant for this question). If your answer is yes, please explain why; if you answer is no please give an counter example.

No. $p(A, B, C) = p(A)p(B)p(C)$ *does not imply* $p(A, B, C|y) = p(A|y)p(B|y)p(C|y)$*. Consider the case where A, B and C are the outcome of three independent coin tosses, where the class label* Y *is 1 if there are even number of heads out of the three tosses and 0 otherwise.* A*,* B *and* C *clearly are independent from one another, but we can infer the value of* C *if we know the value of* A *and* B *given* y*.*

2. (Naive Bayes learns linear decision boundary.) (10 pts) Show that the following naive Bayes classifiers learn linear decision boundary $w_0 + w_1 x_1 + w_2 x_2 + ... + w_d x_d = 0$. Express the weights using the corresponding Naive Bayes parameters. Hint: start with the decision boundary defined by $\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = 0$.

   (a) Bernoulli Naive Bayes model, where features $x_1, x_2, ..., x_d$ are binary indicating the presence/absence of words in the vocabulary.

   Start from $\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = 0$ *and*
   plug in $P(y=1|\mathbf{x}) = P(\mathbf{x}|y=1)P(y=1)/P(\mathbf{x})$ *and* $P(y=0|\mathbf{x}) = P(\mathbf{x}|y=0)P(y=0)/P(\mathbf{x})$
   The decision boundary can be expressed as:

$$\log \frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x}|y=0)P(y=0)} = 0$$

$$\Rightarrow \log \frac{P(y=1)\prod_i \left(p_{i|1}^{x_i}(1-p_{i|1})^{(1-x_i)}\right)}{P(y=0)\prod_i \left(p_{i|0}^{x_i}(1-p_{i|0})^{(1-x_i)}\right)} = 0$$

   where $p_{i|1}$ *and* $p_{i|0}$ *represent* $P(x_i = 1|y=1)$ *and* $P(x_i = 1|y=0)$ *respectively. This is equivalent to:*

$$\log \frac{P(y=1)}{P(y=0)} + \log \prod_i (\frac{p_{i|1}}{p_{i|0}})^{x_i} + \log \prod_i (\frac{1-p_{i|1}}{1-p_{i|0}})^{1-x_i} = 0$$

$$\log \frac{P(y=1)}{P(y=0)} + \sum_i x_i \log(\frac{p_{i|1}}{p_{i|0}}) + \sum_i (1-x_i)\log(\frac{1-p_{i|1}}{1-p_{i|0}}) = 0$$

$$\log \frac{P(y=1)}{P(y=0)} + \sum_i x_i \left(\log(\frac{p_{i|1}}{p_{i|0}}) - \log(\frac{1-p_{i|1}}{1-p_{i|0}})\right) + \sum_i \log(\frac{1-p_{i|1}}{1-p_{i|0}}) = 0$$

   From this, we can see that Naive Bayes classifier learns a linear decision boundary, where

$$w_0 = \log \frac{P(y=1)}{P(y=0)} + \sum_i \log(\frac{1-p_{i|1}}{1-p_{i|0}})$$

$$w_i = \log(\frac{p_{i|1}}{p_{i|0}}) - \log(\frac{1-p_{i|1}}{1-p_{i|0}})$$

   Further rearrange the expression for w_i*, we have:*

$$w_i = \log(\frac{p_{i|1}}{1-p_{i|1}}) - \log(\frac{p_{i|0}}{1-p_{i|0}})$$

   where $\log(\frac{p_{i|\cdot}}{1-p_{i|\cdot}})$ *is the log odds of* $x_i = 1$ *for class* $\cdot$*. Hence the weight co-efficient is the difference between the log-odds of* $x_i = 1$ *for class 1 and class 0. Let's say for class 1 the probability of* $x_1 = 1$ *is 0.8, and for class 0, the probability of* $x_1 = 1$ *is 0.5. Then the weight for* x_1 *is* $\log(4) - \log(1) = \log(4)$.
   It should be noted that the base of the log does not matter here since changing the base simply introduces a positive scaling factor that is constant for all weights.

   (b) Multinomial Naive Bayes Model, where $x_1, ..., x_d$ representing counts of words $w_1, ..., w_d$ in the vocabulary. Express the weights using the Naive Bayes parameters: the class priors $P(y = 1), P(y = 0)$ and the class conditionals: $p(w_i|y=1) and p(w_i|y=0)$

   Start from:

$$\log \frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x}|y=0)P(y=0)} = 0 \Rightarrow$$

$$\log \frac{P(y=1)}{P(y=0)} + \log \frac{P(\mathbf{x}|y=1)}{P(\mathbf{x}|y=0)} = 0$$

Now replace $P(\mathbf{x}|y=\cdot)$ with $\prod_i P(w_i|y=\cdot)^{x_i}$, we have:

$$\log\frac{P(y=1)}{P(y=0)}+\sum_i \log\frac{P(w_i|y=1)^{x_i}}{P(w_i|y=0)^{x_i}}=0 \Rightarrow$$

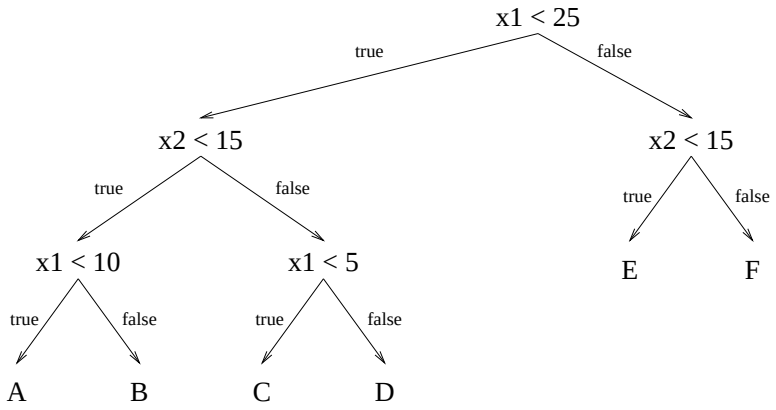
$$\log\frac{P(y=1)}{P(y=0)}+\sum_i x_i \log\frac{P(w_i|y=1)}{P(w_i|y=0)}=0$$

From this we can see:

$$w_0=\log\frac{P(y=1)}{P(y=0)}$$

$$w_i=\log\frac{P(w_i|y=1)}{P(w_i|y=0)}=\log P(w_i|y=1)-\log P(w_i|y=0)$$

3. (6 pts) Consider the following decision tree:



(a) (2 pts) Draw the decision boundaries defined by this tree. Each leaf of the tree is labeled with a letter. Write this letter in the corresponding region of input space.

See Figure 1 for the decision boundary. Note that we label A, B, C, D, E, F with binary labels, for example $\{A, B, C\}$ = positive, and $\{D, E, F\}$ =negative, then we can further determine that some of the line segments do not separate the two classes and thus omitted from this figure.

(b) (2 pts) Give another decision tree that is syntactically different but defines the same decision boundaries. This demonstrates that the space of decision trees is syntactically redundant.

See Figure 2 for the alternative tree. The redundancy is likely to be a computational advantage, because it makes it easier for an imperfect greedy heuristic to find a good solution. For depth-first search methods (such as our greedy algorithm), the ideal search space is one such that every path leads to a solution.

(c) (2pts) How does this redundancy influence learning (does it make it easier or harder to find an accurate tree)? *The redundancy introduced by the fact that different trees can achieve the same decision boundaries suggests that we don't have to find a single "best" tree, as there are multiple equally good trees. This will make learning easier in the sense that a random sequence of node expansions will have a higher chance to lead to a good tree.*

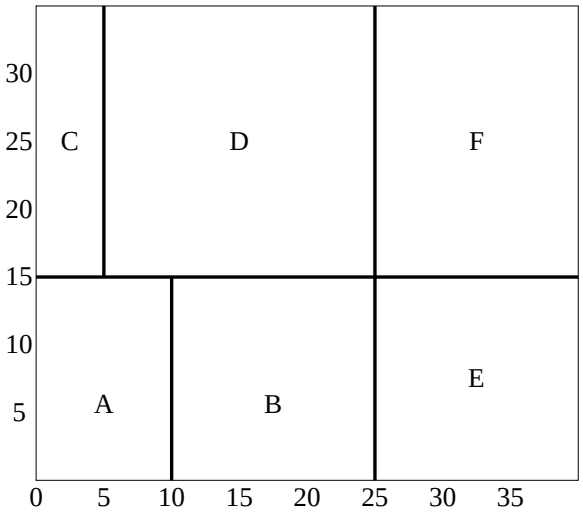


Figure 1: Decision boundary

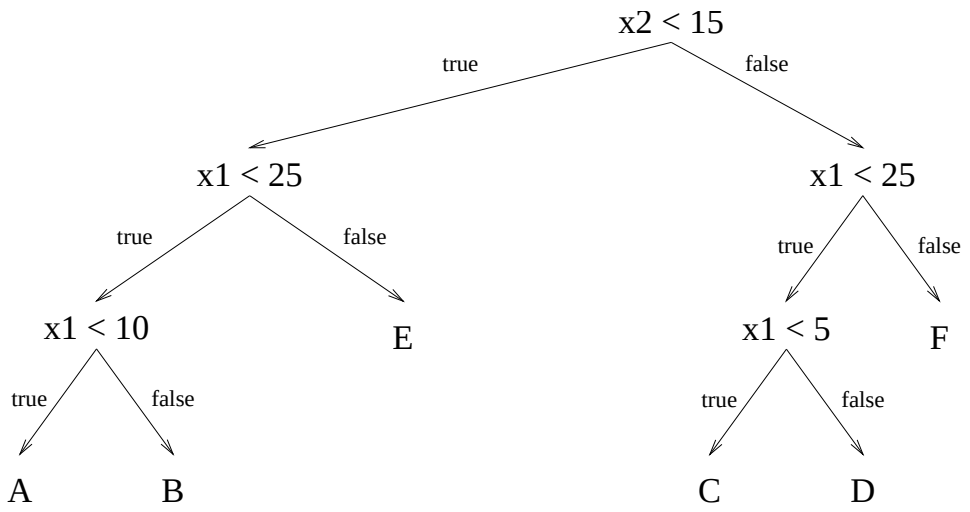


Figure 2: Alternative decision tree.

4. (6 pts) In the basic decision tree algorithm (assuming we always create binary splits), we choose the feature/value pair with the maximum information gain as the test to use at each internal node of the decision tree. Suppose we modified the algorithm to choose at random from among those feature/value combinations that had non-zero mutual information, and we kept all other parts of the algorithm unchanged.

   (a) (2 pts) What is the maximum number of leaf nodes that such a decision tree could contain if it were trained on m training examples?

   A learned tree can have no leaves with zero training examples, because otherwise it will lead to zero information gain. The maximum number would be obtained if each training example were alone in its own leaf: m leaves. (Actually, let m' be the number of distinct training examples. Then no tree can have more than m' leaves.)

   (b) (2 pts) What is the maximum number of leaf nodes that a decision tree could contain if it were trained on m training examples using the original maximum mutual information version of the algorithm? Is it bigger, smaller, or the same as your answer to (b)?

   Using maximum mutual information, in the worst case we could also get one example in each

leaf node. (or m' distinct examples and leaves). But on average, we will have smaller number of leaves.

(c) (2 pts)How do you think this change (using random splits vs. maximum information mutual information splits) would affect the testing accuracy of the decision trees produced on average? Why?

Although in the worst case, both decision trees will have the same size, we expect that in the average case, using randomized splits would give lower accuracy, particularly if there are irrelevant or noisy features in the data. A random split is more likely to split on an irrelevant or inappropriate feature. This will have the unfortunate result of subdividing the data randomly. This effectively creates two learning problems equivalent to the original learning problem, but each has only half as much data. This unnecessary fragmentation of data can be harmful as we know that the more data we have available, the more accurate the learned hypothesis will be on average. Conversely, the less data we have available, the less accurate the hypothesis will be on the average. Therefore, we expect the mutual information selection method to produce more accurate trees on average. This is also the reason why Random Forest, which relies on some randomness in selecting the splits, usually works the best when most of the features are at least somewhat relevant to the classification task.

5. (8 pts) Consider the following training set:

A	B	C	Y
0	1	1	0
1	1	1	0
0	0	0	0
1	1	0	1
0	1	0	1
1	0	1	1

Learn a decision tree from the training set shown above using the information gain criterion. Show your steps, including the calculation of information gain (you can skip $H(y)$ and just compute $H(y|\mathbf{x})$) of different candidate tests. You can randomly break ties (or better, choose the one that give you smaller tree if you do a bit look ahead for this problem).

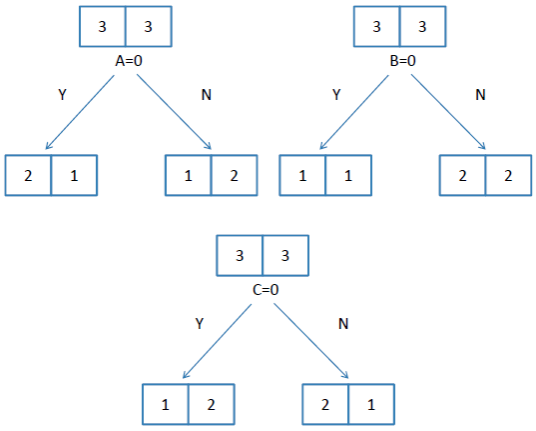


Figure 3: Decision tree step 1.

See Figure 3 for root node selection.

$$H(y|A=0) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$

$$H(y|A=1) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$
$$H(Y|A) = p(A=0) * H(y|A=0) + p(A=1) * H(y|A=1) = 0.9183$$

$$H(y|B=0) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$$
$$H(y|B=1) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$$
$$H(Y|B) = p(B=0) * H(y|B=0) + p(B=1) * H(y|B=1) = 1$$

$$H(y|C=0) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$
$$H(y|C=1) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$
$$H(Y|C) = p(C=0) * H(y|C=0) + p(C=1) * H(y|C=1) = 0.9183$$

Since A and C give the same amount of information gain, we randomly pick C as the root node test. For the next step, we focus on the left branch. See Figure 4. It is clear from the figure without computation that B is a better choice.

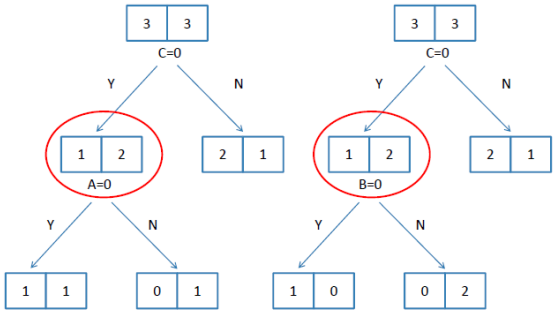


Figure 4: Decision tree step 2.

We can continue and finally reach the decision tree shown in Figure 5.

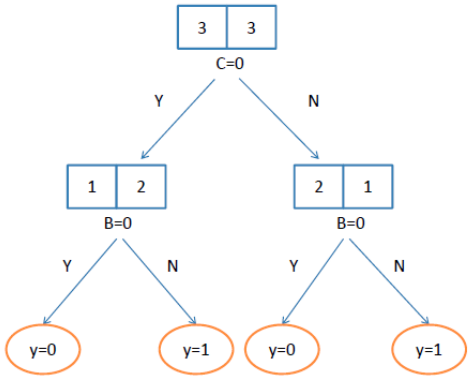


Figure 5: Final decision tree for problem 3.

Note that if you happen to choose A as the root node, the resulting decision tree will be bigger, but correct for this assignment none-the-less.

6. (8pts) Prove that

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

.

Hint: you should start with the definition $H(X,Y) = -\sum_{x,y} P(x,y)\log P(x,y)$. Here we use X, Y to denote the random variables and x, y denote the values X and Y take, $P(x,y)$ is a short hand notation denoting $P(X = x, Y = y)$.

start from the definition:

$$H(X,Y) = -\sum_{x,y} P(x,y)\log P(x,y)$$

first we factorize $P(x,y)$ as $P(x|y)P(y)$:

$$H(X,Y) = -\sum_{x,y} P(x|y)P(y)\log P(x|y)P(y)$$

$$= -\sum_{x}\sum_{y} [P(x|y)P(y)\log P(x|y) + P(x|y)p(y)\log P(y)]$$

$$= -\sum_{x}\sum_{y} P(x|y)p(y)\log P(x|y) - \sum_{x}\sum_{y} P(x|y)P(y)\log P(y)$$

Exchanging the two summations:

$$= -\sum_{y}\sum_{x} P(y)P(x|y)\log P(x|y) - \sum_{y}\sum_{x} P(x|y)P(y)\log P(y)$$

$$= -\sum_{y} P(y)\sum_{x} P(x|y)\log P(x|y) - \sum_{y} P(y)\log P(y)\sum_{x} P(x|y)$$

$$= H(X|Y) + H(Y)$$

If we factor $P(x,y)$ as $P(y|x)p(x)$, following the same steps we will arrive at

$$H(X,Y) = H(X) + H(Y|X)$$

.