CMSI 485 – Classwork 5

SOLUTION

Instructions:
This worksheet will not only provide you with practice problems for your upcoming exam, but will add to your toolset as initiate data scientists taking various machine learning problems from start to finish. Specific notes:

- Provide answers to each of the following questions and write your responses in the blanks. If you are expected to show your work in arriving at a particular solution, space will be provided for you.
- Place the names of your group members below:

Group Members:

1. ________________________________
2. ________________________________
3. ________________________________

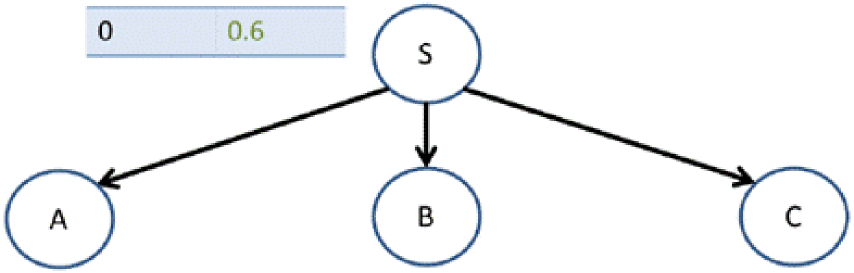
Problem 1 – Naïve Bayes Classifiers

Forney Industries' FornGene group has continued work developing a classifier to identify cases of Schistoforneymiosis [S] based on indicators of three genetic markers [A, B, C]. In this section, we will attempt to craft a Naïve Bayes Classifier that explains the data collected by the FornGene group, and use that to answer classification queries related to the disease. Note how this problem is different than the "Bag of Words" example from class in that each feature's CPT has its own parameters.

S	A	B	C	# of datum
0	0	0	0	48
0	0	0	1	144
0	0	1	0	12
0	0	1	1	36
0	1	0	0	72
0	1	0	1	216
0	1	1	0	18
0	1	1	1	54
1	0	0	0	21
1	0	0	1	189
1	0	1	0	7
1	0	1	1	63
1	1	0	0	9
1	1	0	1	81
1	1	1	0	3
1	1	1	1	27

1.1: In the table above, we track each combination of A, B, C, and S witnessed from the group's study. From this data, construct the *Maximum Likelihood* CPTs over each feature below. *Place your final answers directly in the tables below.*

S	P(S)
0	0.6



S	A	P(A | S)
0	0	0.40
1	0	0.70

S	B	P(B | S)
0	0	0.80
1	0	0.75

S	C	P(C | S)
0	0	0.25
1	0	0.10

1.2. Using the data from the previous section, suppose we instead *smooth* our estimates for the feature likelihoods using Laplacian Smoothing with $k = 10$. Compute $P_{LAP,10}(C = 0|S = 1)$

$$
\begin{aligned}
P_{LAP,10}(C = 0|S = 1) &= \frac{c(C = 0, S = 1) + k}{c(S = 1) + k|X|} \\
&= \frac{40 + 10}{400 + 10 * 2} \\
&\approx 0.12
\end{aligned}
$$

[!] Note: |X| = 2 here because the variable to which we are artificially adding samples is C, which is binary. Since we're adding $k = 10$ samples to each of $C = 0,1$, that's why we add 20 extra to the total in the denominator!

1.3: Using the NBC you learned in Part 1.1, determine (showing your work) the most likely class for each of the following data points.

1. $\{A = 0, B = 1, C = 0\}$

$$
\begin{aligned}
P(S = 0|A = 0, B = 1, C = 0) &\propto P(S = 0)P(A = 0|S = 0)P(B = 1|S = 0)P(C = 0|S = 0) \\
&= 0.6 * 0.4 * 0.2 * 0.25 \\
&= 0.012
\end{aligned}
$$

$$
\begin{aligned}
P(S = 1|A = 0, B = 1, C = 0) &\propto P(S = 1)P(A = 0|S = 1)P(B = 1|S = 1)P(C = 0|S = 1) \\
&= 0.4 * 0.7 * 0.25 * 0.1 \\
&= 0.007 \text{ (in Die Another Bayes)}
\end{aligned}
$$

$$
P(S = 0|A = 0, B = 1, C = 0) > P(S = 1|A = 0, B = 1, C = 0)
$$
$$
\therefore \mathit{Choose}\ S = 0
$$

2. $\{A = 1, C = 1\}$ [Hint: Remember that NBCs are still Bayesian Networks, so B as a missing feature shouldn't be a problem – make sure your answer derives why]

$$
P(S|A, C) \propto \sum_b P(S)P(A|S)P(b|S)P(C|S) = P(S)P(A|S)P(C|S)\sum_b P(b|S) = P(S)P(A|S)P(C|S)
$$

$$
\begin{aligned}
P(S = 0|A = 1, C = 1) &\propto P(S = 0)P(A = 1|S = 0)P(C = 1|S = 0) \\
&= 0.6 * 0.6 * 0.75 \\
&= 0.270
\end{aligned}
$$

$$
\begin{aligned}
P(S = 1|A = 1, C = 1) &\propto P(S = 1)P(A = 1|S = 1)P(C = 1|S = 1) \\
&= 0.4 * 0.3 * 0.9 \\
&= 0.108
\end{aligned}
$$

$$
P(S = 0|A = 1, C = 1) > P(S = 1|A = 1, C = 1)
$$
$$
\therefore \mathit{Choose}\ S = 0
$$

Problem 2 – Linear Perceptrons

Returning to our email classification supervised learning task, consider that we have a trinary class variable $Y \in \{0,1,2\} = \{Spam, Ham, Important\}$. Moreover, we've decided on a simple feature extractor f that takes an email x as input and returns a vector of features indexed as:

1. $f_0(x) =$ *# of capitalized words*
2. $f_1(x) =$ *# of occurrences of word "free"*
3. $f_2(x) =$ *whether or not email is in known contacts (0 = not in contacts, 1 = in contacts)*

2.1. What feature vector would the above feature extractor return for the following email?

$Email\ (x)$	$f_0(x)$	$f_1(x)$	$f_2(x)$
From: Ray.Toal@lmu.edu *Message:* Hi all, There is free pizza in the Keck Lab, COME GET IT!	3 (Also OK: 7, since prompt was vague)	1	1

Consider that the above is in a training set with label $y = 2$ (this is a very important email). If, during learning, our class weight vectors are as follows…

$w_0 = \langle 2, 2, -3 \rangle$ $w_1 = \langle -1, 1, 2 \rangle$ $w_2 = \langle 2, -3, 1 \rangle$

2.2. First, determine which class y our perceptron would currently assign this email.

$$w_0 \cdot f(x) = 2 * 3 + 2 * 1 + (-3) * 1 = 5$$
$$w_1 \cdot f(x) = -1 * 3 + 1 * 1 + 2 * 1 = 0$$
$$w_2 \cdot f(x) = 2 * 3 + (-3) * 1 + 1 * 1 = 4$$

$$y = argmax_y\, w_y \cdot f(x) = 0$$

2.3. Did our Perceptron make a mistake? If so, calculate the updated weights that would amount from its misclassification. If not, draw BlindBot on vacation (or draw him in either case if you want, I'm a classwork, not a cop).

Nope, bad Perceptron! It said $y = 0$ when the correct was $y = 2$. As such, we diminish the $y = 0$ weights and enhance the $y = 2$ weights proportionate to the feature vector that caused the error:

$$w_0 = w_0 - f(x) = \langle 2,2,-3 \rangle - \langle 3,1,1 \rangle = \langle -1,1,-4 \rangle$$
$$w_2 = w_2 + f(x) = \langle 2,-3,1 \rangle + \langle 3,1,1 \rangle = \langle 5,-2,2 \rangle$$

Problem 3 – Logistic Regression

Starting over with the class-weights our perceptron had in the previous section (before any updating in 2.3):

$$w_0 = \langle 2, 2, -3 \rangle \qquad w_1 = \langle -1, 1, 2 \rangle \qquad w_2 = \langle 2, -3, 1 \rangle$$

3.1. Compute the likelihoods of each class $P(y|x; w)$ that a Logistic Regression classifier would give for the sample features $f(x) = \langle 1,2,3 \rangle$.

$$z_0 = w_0 \cdot f(x) = \langle 2,2,-3 \rangle \cdot \langle 1,2,3 \rangle = 2 + 4 - 9 = -3$$
$$z_1 = w_1 \cdot f(x) = \langle -1,1,2 \rangle \cdot \langle 1,2,3 \rangle = -1 + 2 + 6 = 7$$
$$z_2 = w_2 \cdot f(x) = \langle 2,-3,1 \rangle \cdot \langle 1,2,3 \rangle = 2 - 6 + 3 = -1$$

$$e^{z_0} = e^{-3} \approx 0.05$$
$$e^{z_1} = e^{7} \approx 1096$$
$$e^{z_2} = e^{-1} \approx 0.37$$

…OK so maybe I didn’t quite due the math before making up the numbers here, pretty clear winner for class $y = 1$…

$$P(Y = 0|x; w) = \frac{e^{z_0}}{e^{z_0} + e^{z_1} + e^{z_2}} = \frac{0.05}{1096.42} \approx 0$$
$$P(Y = 1|x; w) = \frac{e^{z_1}}{e^{z_0} + e^{z_1} + e^{z_2}} = \frac{1096}{1096.42} \approx 1$$
$$P(Y = 2|x; w) = \frac{e^{z_2}}{e^{z_0} + e^{z_1} + e^{z_2}} = \frac{0.37}{1096.42} \approx 0$$

…usually it’s not this dramatic a difference, sorry about that!

$P(Y = 0\|x; w) =$ 0	$P(Y = 1\|x; w) =$ 1	$P(Y = 2\|x; w) =$ 0

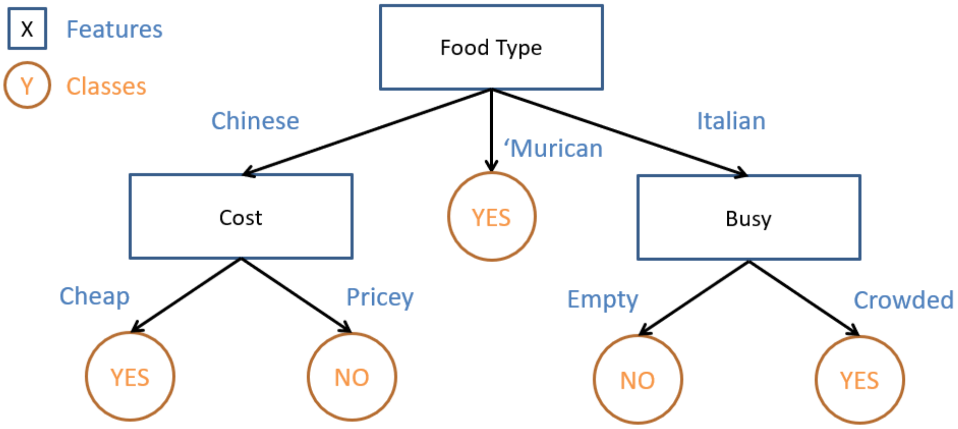
Problem 4 – Decision Trees

Although not a technique that we saw in lecture, one of the most successful supervised learning models for smaller datasets is what is known as a *decision tree* (or d-tree for short). Not only are these tools powerful at combatting overfitting, but are one of the most *interpretable* models for humans to understand, and are often used in medical application for doctors.

In decision trees:

- Nodes are features on which to partition the dataset / a sample.
- Edges are values of those features.
- Leaves are classifications / assigned labels.
- Any sample x can be classified by starting at the root and then following the path for each variable's value in the sample until a leaf is hit.

Consider the following example d-tree that might be used to classify whether or not an app's user will want to dine at a given restaurant (binary $Y \in \{No, Yes\}$).



4.1. How would the above d-tree classify a restaurant for which:

$$Cost = Pricey \quad Type = Chinese \quad Busy = Empty$$

No, since we take the left branch at the root for $Food\ Type = Chinese$ and then a right branch at the $Cost = Pricey$ feature split, leading to the class No leaf.

4.2. Sketch a strategy that you think could be used to learn a d-tree from some supervised dataset (i.e., paired samples of input features X with expected output classes Y).

At each feature node, we partition the dataset recursively such that, as soon as we've reached some level of confidence that a split leads to a decent classification (being vague here on what "decent" qualifies as), or we run out of additional features to split on, we place a class leaf that maximizes the likelihood of being correct in the partition of the dataset remaining at that part of the tree.