CS 471-Spring 2021
Name: Noah Cornett
Purdue ID: 0029684020
Date: 29 April, 2021

CS47100 Homework 5

Due date: Friday April 30, 11:59pm

This homework will involve both written exercises (`hw5_written.pdf`) and a programming component (`hw5_coding.pdf`). Submission instructions are detailed below.

Written Assignment (16 pts)

1. (3 pts) Consider the problem faced by an infant learning to speak and understand language. Explain how this process fits into the general learning model. Describe the percepts and actions of the infant, and the types of learning the infant must do. Describe the subfunctions the infant is trying to learn, including types of inputs and outputs, and available example data.

   In this situation, there are several percepts that the baby can perceive: (1) Any other person talking to them (2) the noises that the baby is making and (3) the reaction of other when the baby says something.

   The actions of the infant in relation to leaning to speak and understand is mainly just trying to speak. So, the main action that a baby would take is combining noises in attempt to communicate with the others around them.

   The infant is likely to perform a combination of reinforcement learning and unsupervised learning.

   The reinforcement portion of learning comes from the fact that the infant is making meaningless noises over and over again which correlates to the exploration phase of learning. The reward function of making these noises, the actions of the infant, depend on the reaction of the parents or others around the infant.

   The unsupervised learning portion comes from the fact that the infant can hear other around them speak, but the infant does know what it means. In machine learning terms, the infant has a lot of data readily available to analyze, but the infant does not have any of that data classified, so they have to try and find the patterns themselves.

   So, the infant is applying a combination of reinforcement and unsupervised learning to learn how to put together noises to formulate words to communicate with the other around them.

2. (3 pts) Suppose we generate a training set from a decision tree and then apply decision tree learning to that training set.

   (a) Is it the case that the learning algorithm will eventually return the correct tree as the training set goes to infinity? Why or why not?

   Note: Since the training set given comes from a decision tree, each example in the training set has at least one path from the root to a leaf node that explains that part of the data.

   This decision tree learned would be correct only for the training data. If the learning algorithm continually learns on the training set and tunes the decision tree accordingly, eventually the decision would have one path to a leaf node for each example in the data. Thus, the decision tree will have perfect accuracy on the training data.

   However, this decision tree only has good accuracy on the training data. This model would be extremely overfit to the training data and would likely perform very poorly in practice.

   (b) What if we generated the training set from a Bayesian network, but learned a decision tree?

   When we generate the training set from a Bayesian network, the same property does not hold since the training data was generated in a non-deterministic way (probabilistic nature due to Bayes net).

   Consider an example that has the set of attributes denoted by X. Then, according to the bayes net, there is a p chance of X given the class label. So, several points of data that are generated based off of X has a chance of having different classifications.

   In this case, decision tree learning would never be able to output a 100% correct model. In a decision tree, the same set of attributes will always lead you to the same result, but in this case, the same attributes do not necessarily have to have the same class label.

3. (4 pts) Suppose you are running a learning experiment on a new algorithm for Boolean classification. You have a data set consisting of 100 positive and 100 negative examples. You compare your algorithm to a baseline function, a simple majority classifier. (A majority classifier is given a set of training data and then always outputs the class that is in the majority in the training set, regardless of input.)

   (a) You plan to use leave-one-out cross validation (see Section 18.4 in the book) for evaluation and you expect the majority classifier to score about 50% accuracy on leave-one-out cross validation, but to your surprise, it scores zero every time. Explain why.

   Leave-one-out cross validation is a special case of k-fold cross validation where you take one element out of the data for validation and train your data on the remainder of the data.

   In this case, if we take one positive classifying element out of the training set, then the majority classifier will classify everything as negative since there are now 99 positive and 100 negative. Consequently, when testing the trained classifier on the left out data point, it will always give the wrong classification since that data point is classified as positive.

   The same argument is made when the left out data point is classified as negative.

   Therefore, this will always get an accuracy of 0%.

   (b) Describe how the performance of the majority classifier will change as the k in cross validation is reduced from n to 10. Do you ever expect to see accuracy greater than 50%?

   If we take out 10 elements from the training data and, of those 10 elements, there is a majority of positive classifying elements, then the remainder of the training has a majority of negative classifying elements. So, the majority classifier will classify everything as negative. Then, when you test this classifier on the held-out validation data, it will score an accuracy of less than 50% since the majority of the elements in the held out data are positive and the classifier outputs that they are negative.

   A similar argument is made when the majority of the held-out data is negative.

   Now consider when the 10 elements of the held out data consist of 5 positive and 5 negative elements. In this case, the training data has the same amount of positive and negative elements. So, there is not a majority element in the remainder of the training data. In this case, let us assume that the majority classifier will classify an object as positive 50% of the time and negative the other 50% of the time. In this case, we do expect to see the classifier getting close to 50% accuracy, and potentially even being higher than 50%. Actually, we would expect the classifier to score an accuracy of exactly 50%, but due to the proabilistic nature of the classifier, there is a chance that it scores higher or lower than that.

4. (6 pts) Consider an arbitrary Bayesian network, a complete set of data for that network, and the likelihood for the data set according to the network.

   (a) Give a simple proof that the likelihood of the data cannot decrease if we add a new arc to the network and recompute the maximum likelihood parameter values.

   When you add an arc to the network, you are adding a new parent to some other node. So, you are adding another variable to include in the prior information when calculating the probability of the child node.

   When we are given more information about a child node via the parent nodes, the estimated probability of that child node is going to be greater than or equal to the estimate without the parent node. In the worst case, the new arc added does not have any causality, in which case $P(c|p)P(c)$ where c is the child node and p is the parent node. In the best case, the added node has some level of causality, so the probability will be more accurate.

   While this extra prior knowledge may not give us any more evidence, it is not going to decrease the probability. Hence, the likelihood of the data given this new model will not decrease.

   (b) What will result if we use maximum likelihood as the evaluation function in greedy hill-climbing search for the best Bayes net structure?

   When searching for the best structure using greedy-hill climbing and maximum likelihood as the score function, the algorithm will eventually reach a state where the structure it has generated is a pretty good one. However, as shown in the previous problem, adding an arc to the Bayes net cannot decrease the likelihood of that model being correct given the data. So, the greedy-hill climbing algorithm would just continue to add arcs since the score is not decreasing.

   This would lead to an incredibly complex Bayes net. This is why penalized likelihood is used as a score function when using greedy-hill climbing to find the structure of a Bayes net.

Submission Instructions:

Upload your **typed** answers to the written questions as a **pdf** in gradescope:
`https://www.gradescope.com/courses/231780`

- For your pdf file, use the naming convention `username_hw#.pdf`. For example, your TA with username *roy98* would name his pdf file for HW4 as `roy98_hw5.pdf`.
- To make grading easier, please start a new page in your pdf file for each subquestion. Hint: use a `\newpage` command in LaTeX after every question ends.
- After uploading to gradescope, mark each page to identify which question is answered on the page. (Gradescope will facilitate this.)
- Follow the above convention and instruction for future homeworks as well.