Problem Set 1

CS7641 Machine Learning, 2019 Fall

Hui Xia (hxia40)
903459648
Georgia Institute of Technology

2. Design a two-input perceptron that implements the boolean function $A \wedge \neg B$. Design a two-layer network of perceptrons that implements $A \oplus B$ ($\oplus$ is XOR).

As described by the textbook (Mitchel) **Section 4.4.1,** a single perceptron can represent many of the primitive boolean functions, with a few exceptions such as XOR. One way of describing the boolean function $A \wedge \neg B$ can be shown here in **Figure 1.** Note that this is just one viable input/weight/threshold combination, and there are many such possible combinations that can make a single perceptron can represent the boolean function $A \wedge \neg B$.

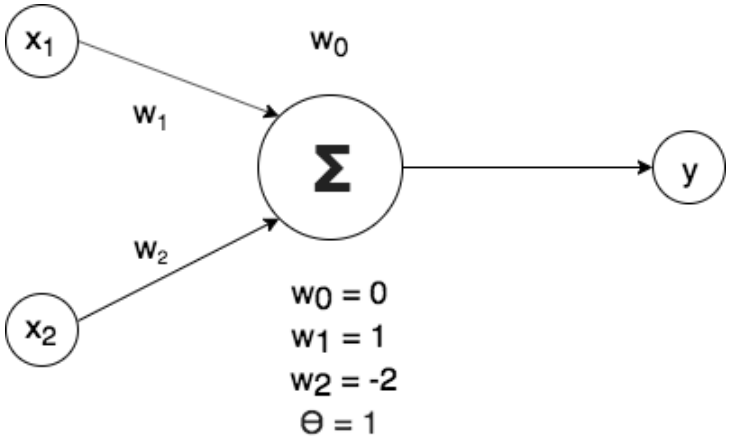

Figure 1: A single perceptron with two inputs that represent the boolean function $A \wedge \neg B$

While a single perceptron cannot represent XOR, a two-layer network of perceptrons can do so. Again, there are many viable input/weight/threshold combinations. One example is shown in **Figure 2**, which demonstrate a two-layer network, with the first (hidden) layer of two nodes – h_1 and h_2, and the second layer as y. As there are many weights to show, they are just shown as either bias on each hidden layer and on y (represent w_0 on a single perceptron), or as numbers on the side of arrows (represent other weights).

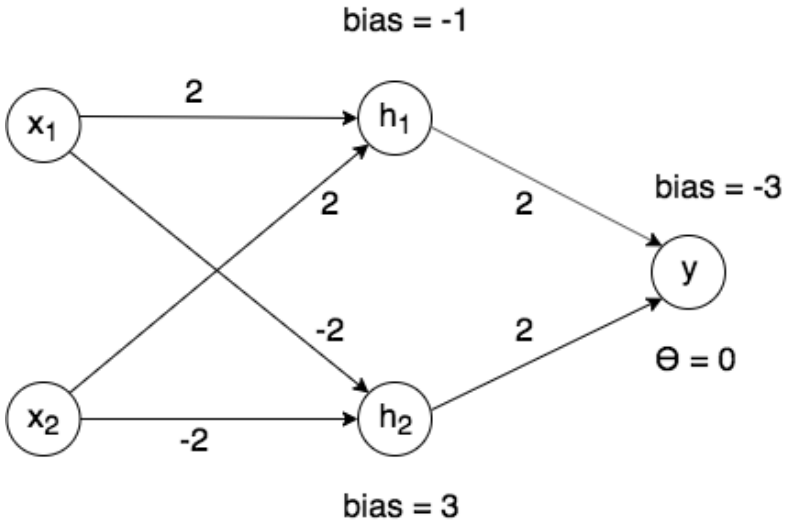

Figure 2: A two-layer network of perceptrons that represent XOR.

5. Suggest a lazy version of the eager decision tree learning algorithm ID3. What are the advantages and disadvantages of your lazy algorithm compared to the original eager algorithm?

The eager decision tree learning algorithm ID3 is 'eager', is because this algorithm focuses on information gain. In other words, ID3 choose 'best' (the one that results largest information entropy gain) attribute to separate data, then use the average/median of the selected features to make the separation. By being so eager and try to grab any single bit of information, ID3 will make many unmeaningful branches on the tree doing so will not only overfit the data (as there are too much information to split on – the algorithm kind of lost focus!), but also not cost-effective in words of running speed (as a large proportion of the calculation force is wasted on meaningless splits).

Thus, a 'lazy' version of the decision tree learning algorithm will be, to not being so 'eager', and try to gain information on any possible chance. Instead, the lazy algorithm will simply choose random attributes (instead of choosing the 'best' attribute) and make random splits (instead of making split on the average/median).

Adopting such 'lazy' decision tree will have several advantages:

1. Less tendency to overfit data – again, the lazy tree tends to not focus on specific cases.
2. Cost-effective on calculation power- this is not only because of we saved calculation power on calculation of information entropy, but also due to that as we skipped many meaningless splits, the lazy tree will tend to be shorter compared with the eager tree.
3. The lazy tree will be more tolerable to missing data points.

However, adopting the lazy tree algorithm will also have disadvantages.

1. The lazy tree cannot prune itself. The information on which section of the tree can provide more information gain is lost.
2. A single lazy tree is too weak and cannot utilize the information effectively (as opposite to eager trees). Thus, it needs to be used with bagging/boosting algorithms to use data effectively.

6. Imagine you had a learning problem with an instance space of points on the plane and a target function that you knew took the form of a line on the plane where all points on one side of the line are positive and all those on the other are negative. If you were constrained to only use decision tree or nearest-neighbor learning, which would you use? Why?

I will choose nearest neighbors. As per described by the question, I have to use one single line to separate points on a plane. If such line is defined by decision tree, to make such separation acceptably accurate, the decision tree will need to be very complex, looping around the interface of the two groups of points. Each time when this line makes a turn, there will be another feature (likely to be the coordinates on the plane) choose and separated. As the decision tree keep learning the coordinates of new points, the line will get even more and more complex, and the decision tree will get bigger and bigger.

On the other hand, however, using nearest neighbor, a line on the plane can be simply defined as the interface-line of the two groups of points that was generated by the nearest neighbor algorithm. This is a more concise method.