15-381 Artificial Intelligence: Representation and Problem Solving Homework 5 Solutions

1 [20 pts] Naive Bayes

Consider the following set of training examples for the unknown target function $< X1, X2 > \rightarrow Y$. Each row indicates the values observed, and how many times that set of values was observed. For example, (+, T, T) was observed 3 times, while (-, T, T) was never observed.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

Construct a Naive Bayes classifier for this data using the method described in the class and state the values of the parameters learnt.

Solution

$P(X_1 = T|Y = +) = 7/12$ $P(X_1 = T|Y = -) = 1/9$ $P(X_2 = T|Y = +) = 7/12$ $P(X_2 = T|Y = -) = 3/9$ $P(Y = +) = 12/21$

2 [15 pts] Linear Regression

Consider a series of observations of the form $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$. Given these observations("training data"), our aim is to find a function f of the form $f(x) = wx$ such that $f(x)$ is a good estimate of y for this dataset. We saw this problem in class; this is just the problem of linear regression.

In class, we found the w that minimized the square error over the data($\sum_i^n (y_i - wx_i)^2$) and considered that to be a good estimate. We will now see why that might be considered a good estimate.

Suppose the likelihood of the observation y_i is a Gaussian distribution with mean wx_i and variance σ, i.e. $P(Y = y_i|w, x_i) = 1/(2\pi\sqrt{\sigma})e^{-(y_i - wx_i)^2/(2\sigma^2)}$

1. Derive the expression for the conditional likelihood of the data assuming that each observation was independently generated. ie. write the expression for $P(y_1, y_2, \dots, y_n|w, x_1, x_2, \dots x_n) = \prod_i^n P(y_i|w, x_i)$
2. Now, using the expression for the conditional likelihood of the data, write down the expression for the conditional *log*-likelihood, $\log(P(y_1, y_2, \dots, y_n|w, x_1, x_2, \dots x_n))$

3. Show that computing the Maximum Likelihood Estimate of w is equivalent to finding the w that minimizes the square error. Recall from earlier classes, that a Maximum Likelihood Estimate of a parameter(in this case w) is the value of the parameter that maximizes the likelihood of the data.

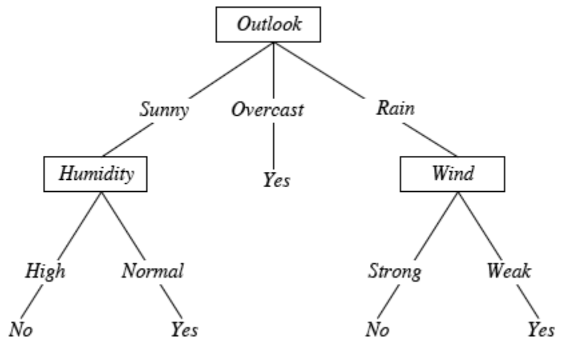
Solution

1. $\prod_i^n P(y_i|w, x_i) = \prod_i^n 1/(2\pi\sqrt{\sigma})e^{-(y_i - wx_i)^2/(2\sigma^2)} = (1/(2\pi\sqrt{\sigma}))^n e^{-(\sum_i^n (y_i - wx_i)^2)/2\sigma^2}$
2. $l = \log(\prod_i^n P(y_i|w, x_i)) = n\log((1/(2\pi\sqrt{\sigma}))) - (\sum_i^n (y_i - wx_i)^2)/2\sigma^2$
3. Need to maximize likelihood, or equivalently, the log-likelihood. $n\log((1/(2\pi\sqrt{\sigma})))$ is a constant independent of w, therefore need to maximize $-(\sum_i^n (y_i - wx_i)^2)/2\sigma^2$, or(dropping minus sign), minimize $(\sum_i^n (y_i - wx_i)^2)/2\sigma^2$. Since σ is a constant, this is equivalent to minimizing $(\sum_i^n (y_i - wx_i)^2)$, i.e. finding the w that minimizes the square error.

3 [15 pts] Learning Decision Trees

Consider the following training data and the following decision tree learned from this data using the algorithm described in class. The decision tree predicts the value of the boolean attribute PlayTennis using the values of the rest of the attributes.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



1. Show that the choice of the Wind attribute at the second level of the tree is correct, by showing that its information gain is superior to the alternative choices.
2. Add one new example to the above data set, so that the learned tree will contain additional nodes.

3. Is it possible to add new examples to the above training set, which are consistent with the above tree, to produce a larger training set such that the algorithm will now learn a tree whose root node is not Outlook?. (We say an example is consistent with the above tree if the tree classifies the example correctly). Justify your answer by explaining informally why this is impossible, or explaining the new data you would add.

Solution

1.

$$H(PlayTennis|Wind, Outlook = Rain)$$
$$= P(Wind = Strong|Outlook = Rain)H(PlayTennis|Wind = Strong, Outlook = Rain)$$
$$+P(Wind = Weak|Outlook = Rain)H(PlayTennis|Wind = Weak, Outlook = Rain)$$
$$= 2/5 * 0 + 2/5 * 0 = 0$$
$$H(PlayTennis|Humidity, Outlook = Rain) =$$
$$P(Humidity = High|Outlook = Rain)H(PlayTennis|Humidity = High, Outlook = Rain)$$
$$+P(Humidity = Normal|Outlook = Rain)H(PlayTennis|Humidity = Normal, Outlook = Rain)$$
$$= 2/5*(-1/2*log(1/2)-1/2*log(1/2))+3/5*(-1/3log(1/3)-2/3log(2/3)) = 2/5(1)+3/5(0.91) = 0.951$$
$$H(PlayTennis|Temperature, Outlook = Rain) =$$
$$P(Temperature = Hot|Outlook = Rain)H(PlayTennis|Temperature = Hot, Outlook = Rain)$$
$$+P(Temperature = Mild|Outlook = Rain)H(PlayTennis|Temperature = Mild, Outlook = Rain)$$
$$+P(Temperature = Cool|Outlook = Rain)H(PlayTennis|Temperature = Cool, Outlook = Rain)$$
$$= 0+3/5*(-2/3log(2/3)-1/3log(1/3))+2/5*(-1/2log(1/2)-1/2log(1/2)) = 3/5(0.91)+2/5(1) = 0.951$$

2. Add D15 (a variant of $D4$) : Outlook=Sunny AND Temperature=Hot AND Humidity=High AND Wind=Weak AND PlayTennis=No. The combination of Outlook=Sunny and Wind=Weak is no longer enough to uniquely determine the value of PlayTennis (since D4 and D15 agree on Outlook and Wind, but have different values for PlayTennis). There will have to be an additional decision incorporating Temperature.

3. Add a large number of entries(say 500 each) having Outlook= Sunny and Humidity = High and PlayTennis = No, and, Outlook = Sunny and Humidity = Normal and PlayTennis=Yes (and randomly selected values for the rest of the attributes). Notice that that for these 1000 entries, splitting on Humidity will give a one node Decision Tree. On the entire dataset(D1 through D14 plus the new data), the information gain caused by splitting on Humidity will now be much higher(since it is a near perfect indicator of PlayTennis on a huge number of data points) than splitting on Outlook.

4 [15 pts] Decision Trees to Rules

Any decision tree can be re-expressed as a set of rules, with one rule for each leaf. The preconditions of the rule correspond to the sequence of attribute tests along the path from the tree root to the leaf. For example, the leftmost leaf in the tree in Q. 3 corresponds to the rule IF (Outlook = sunny AND Humidity=High) THEN PlayTennis=No

1. Write the rules for the remaining leaves. Note this set of rules produces classifications that are identical to the above tree, over the training data and over any other possible instance.

2. It is possible to translate any tree into a set of rules that represents an equivalent classifier. Is it possible to translate any set of rules into an equivalent tree? Explain how or give a counterexample. You may assume that all your attributes are boolean.

Solution:

1. (a) IF (Outlook = sunny AND Humidity=Normal) THEN PlayTennis=Yes
   (b) IF (Outlook = sunny AND Overcast=Yes) THEN PlayTennis=Yes
   (c) IF (Outlook = sunny AND Wind=Strong) THEN PlayTennis=No
   (d) IF (Outlook = sunny AND Wind=Weak) THEN PlayTennis=Yes

2. Any classifier over boolean attributes can be thought of as a boolean function over the attributes giving the label(0/1) as an answer. We will show that a Decision Tree can encode any truth table, and therefore any boolean function. Therefore it can encode any set of (internally consistent) rules to produce an equivalent classifier.

   Consider a full Decision Tree as follows: The node at the root is "input 1", the nodes at the next level represent splits on "input 2", and so forth to the last level of internal nodes, which represent a split on "input n". It is easy to see that there are 2^n leaves in this tree, each corresponding to a full instantiation to the boolean attributes, ie, each leaf coresponds to one entry in the truth table. Now, for each leaf, just run the rule-classifier on the input corresponding to the path to the leaf and store the output of the rule-classifier as the label of the leaf. It is easy to see that this mimics the rule-classifier on all possible inputs(by construction) and therefore is equivalent.

5 [15 pts] Neural Networks

Suppose that you have two types of activation functions at hand:

- Identity function: $g_I(x) = x$
- Step function: $g_s(x) = 1$ if $x \geq 0$, 0 otherwise

So, for example, the output of a neural network with one input x, a single hidden layer with K units having step function activations, and a single output with identity activation can be written as $out(x) = g_I(w_0 + \sum_{i=1} w_i g_s(w_0^{(i)} + w_1^{(i)} x))$, and can be drawn as in Fig. 5

1. Consider the step function: $u(x) = y$ if $x < a$, 0 otherwise.

   Construct a neural network with one input x and one hidden layer whose response is $u(x)$. Draw the structure of the neural network, specify the activation function for each unit (either g_I or g_s), and specify the values for all weights (in terms of a and y).

   Solution:
   $g_I(y g_s(x - a))$

2. Now, Construct a neural network with one input x and one hidden layer whose response for given real values y, a, and b is y if $x \in [a, b)$, and 0 otherwise. Draw the structure of the neural network, specify the activation function for each unit (either g_I or g_s), and specify the values for all weights (in terms of a, b, and y).

   Solution:
   $g_I(y/2 g_s(x - a) + y/2 g_s(b - x))$

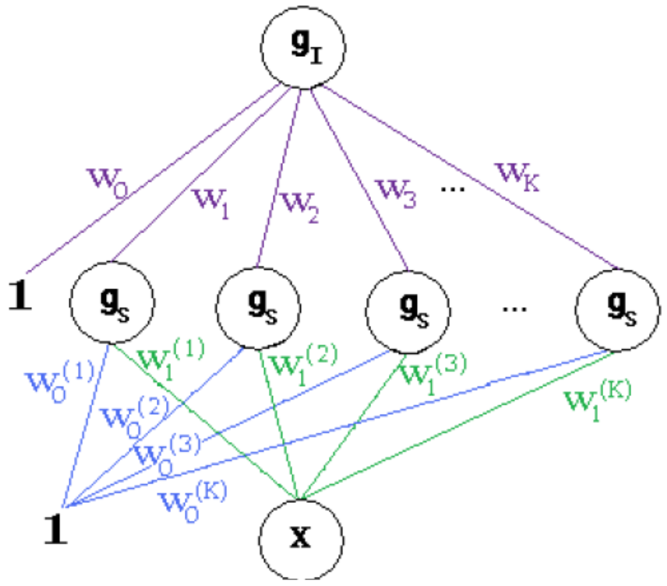


Figure 1: Example Neural Network for Q. 5

6 [20 pts] DTrees vs Neural Networks

Consider the problem of learning a function $Y = f(X_1, X_2, \ldots X_n)$ over boolean valued attributes $X_1, X_2, \ldots X_n$. For each of the following cases, state if it is possible for a Decision tree and a Neural Network(NN) to compute this function. If possible, describe/draw the resulting decision tree/neural network. Also, for the decision tree, state the number of levels in the tree; for the NN, state the number of hidden layers that your NN uses.

1. f = Majority function. I.e. $f(X_1, X_2, \ldots X_n) = 1$ if atleast half the X_is are 1; 0 otherwise.
2. f = Parity function. I.e. $f(X_1, X_2, \ldots X_n) = 1$ if the sum of all X_is is odd; 0 otherwise.

Solution

1. *Decision Trees* From solution to Q 3.2, any boolean function can be encoded as a Decision Tree. Therefore both Majority and Parity can be encoded as DTrees. To implement the majority function: Each leaf node in the complete DTree is reached by traversing a number of "true" splits and a number of "false" splits. If the number of "true" splits is greater than the number of "false" splits, the assignment to the leaf node should be "true", else the assignment to the leaf node should be false. To implement the parity function: Construct a full binary tree in the same manner as for the majority function. Assign the following values to the leaf nodes: If the tree walk required to get to the leaf traverses an odd number of "true" splits, the assignment to the leaf shall be "true", else its value shall be "false"

2. *Neural Networks* The majority function is easy. You can a simple two layer Neural Network in the following manner: $g_s(\sum_i^n g_I(x_i) - n/2)$

   The parity function is harder. You can show that you need atleast one hidden layer in order to encode this. The network is a feed-forward network with n input units, n binary threshold units in the middle layer, and a single binary threshold output unit. All weights from the inputs to the the middle layer are set equal to 1, whilst those from the middle layer to the output are given the values + 1, -1, + 1,-1, . . . , respectively, that is, the weight from middle unit j to the output unit is set equal to $(-1)^{j+i}$

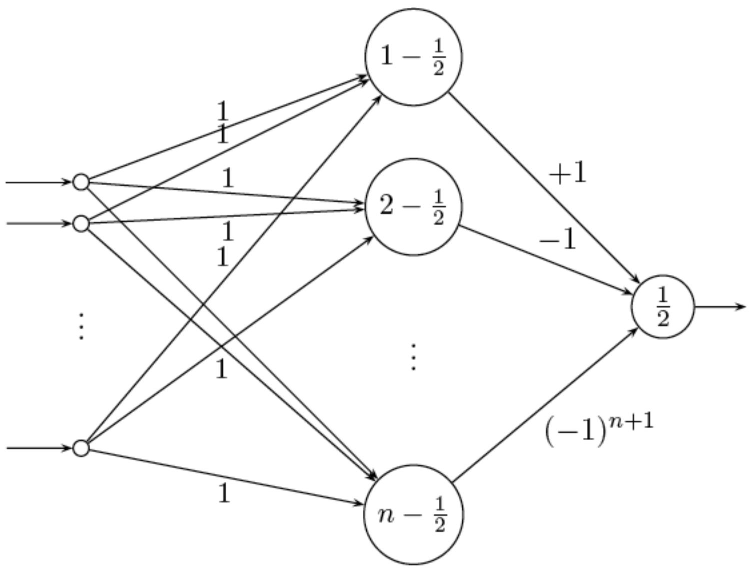


Figure 2: Neural network for n-bit parity

The threshold values of the middle units are 1-1/2, 2 - 1/2 , . . . , n -1/2 and that for the output unit is set to 1/2 2 . When k input units are "on", i.e., have value 1, then the total input (net activation) to the jth middle unit is equal to k. Thus the units 1, 2, . . . , k in the middle layer all fire (since k - (j - 1/2) $\geq$ 0 for all $1 \leq j \leq k$). The middle units k + 1, . . . , n do not fire. The net activation potential of the output unit is therefore equal to 1-1+1...(k terms) = 1 (if k is odd); 0 (if k is even)

See http://www.mth.kcl.ac.uk/ iwilde/notes/nn/nnnotespdf.pdf for a more detailed treatment.