CS 760 Homework 5:

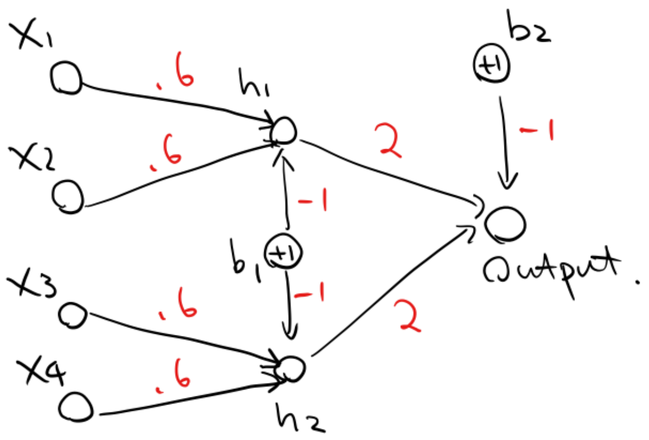
Qihong Lu

December 12, 2016

Question 1

Show a neural network that represents the logical function y = (x1 ∧ x2) ∨ (x3 ∧ x4). Specifically, show the network topology, weights and biases. You should assume that hidden and output units use sigmoid output functions, and an output-unit activation of 0.5 or greater represents a true prediction for y.

The network architecture is shown below. The weights are denoted in black. X_i are the input units. h_i are the hidden units. b_i are the bias units. Finally, there is a single output unit.



The unit h1 is representing an AND function over X1 and X2. Similarly, the unit h2 is representing an AND function over X3 and X4. Finally, the output unit represents an OR function over h1 and h2.

QUESTION2

Consider the concept class C in which each concept is an interval on the line of real numbers. Each training instance is represented by a single real-valued feature x, and a binary class label $y \in \{0,1\}$. A learned concept is represented by an interval [a, a + b] where a is real value and b is a positive real value, and the concept predicts y=1 for values of x in the interval, and y=0 otherwise. Show that C is PAC learnable.

The hypothesis space, H consists of all possible intervals in the form of [a, a + b]. The VC-dimension(H) is 4. Because when there are 4 instances: $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ with $x_1 < x_2 < x_3 < x_4$ and $y_1 = y_3 \neq y_2 = y_4$, there is no $h \in H$ that can achieve zero training error. Because we have a continuous hypothesis space, we can use the following bound for the sample complexity:

$$m \geq \frac{1}{\epsilon}(4\log_2\frac{2}{\delta} + 8\text{VC-dim}(H)\log_2\frac{13}{\epsilon}) = \frac{1}{\epsilon}(4\log_2\frac{2}{\delta} + 32\log_2\frac{13}{\epsilon})$$

Therefore, m grows polynomially in $\frac{1}{\epsilon}, \frac{1}{\delta}$.

I still need to show that we can find a consistent hypothesis in polynomial time.
Given a training set $D = \{x_i, y_i\}$, i = 1,2,..., N.
Let $c = y_1 \in 0,1$. Loop over y_i from 1 to m. When $y_i \neq c$, set a = x_i, c = y_i, t = i. Keep looping over value of y_i, start from i = t. When $y_i \neq c$, set b = x_i - a.

This algorithm will find a consistent hypothesis (assuming consistent hypothesis exists) in linear time w.r.t to the number of training example. Therefore, C is PAC learnable.

QUESTION3

Consider the concept class that consists of disjunctions of exactly two literals where each literal is a feature or its negation, and at most one literal can be negated. Suppose that the number of features n = 3. Show what the Halving algorithm would do with the following two training instances in an on-line setting. Specifically, show the initial version space, the prediction made by the Halving algorithm for each instance, and the resulting version space after receiving the label of each instance.

x1	x2	x3	y
T	F	F	pos
F	T	T	neg

At time t, given the training instance $x^{(t)}$, the Halving algorithm reduces the version space, VS_t, by remove all hypotheses, $h \in H$, that are inconsistance with $x^{(t)}$.

Initiailly, $VS_0 = \{x_1 \wedge x_2, x_1 \wedge x_3, x_2 \wedge x_3, x_1 \wedge \neg x_2, x_1 \wedge \neg x_3, x_2 \wedge \neg x_3, \neg x_1 \wedge x_2, \neg x_1 \wedge x_3, \neg x_2 \wedge x_3\}$, where element in the version space in the form of $x_i \wedge x_j$ represents a hypothesis that predicts positive when $x_i \wedge x_j$ is satisfied.

Step 1: Given the first instance $[x_1, x_2, x_3] = [T, F, F]$ with y = Positive. The majority vote of VS_0 is Negative, which is a mistake. The predictions of the inidividual hypothesis are listed here:

hypothesis	prediction	remove?
$x_1 \wedge x_2$	Negative	T
$x_1 \wedge x_3$	Negative	T
$x_2 \wedge x_3$	Negative	T
$x_1 \wedge \neg x_2$	Postive	F
$x_1 \wedge \neg x_3$	Postive	F
$x_2 \wedge \neg x_3$	Negative	T
$\neg x_1 \wedge x_2$	Negative	T
$\neg x_1 \wedge x_3$	Negative	T
$\neg x_2 \wedge x_3$	Negative	T

Update $VS_1 = \{x_1 \wedge \neg x_2, x_1 \wedge \neg x_3\}$

Step 2: Given the 2nd instance $[x_1, x_2, x_3] = [F, T, T]$ with y = Negative. The majority vote VS_1 is Negative, which is correct. The predictions of the inidividual hypothesis are listed here:

hypothesis	prediction	remove?
$x_1 \wedge \neg x_2$	Negative	F
$x_1 \wedge \neg x_3$	Negative	F

Nothing needs to be removed, so update $VS_2 = VS_1 = \{x_1 \wedge \neg x_2, x_1 \wedge \neg x_3\}$

QUESTION4

In this same setting, suppose the learner can pick the next training instance it will be given. That is, the learner can pick the feature vector part of the instance; the class label will be provided by the teacher. Which instance should it ask for next? Justify your answer.

After the 2nd step, the version space is $VS_2 = \{x_1 \wedge \neg x_2, x_1 \wedge \neg x_3\}$
I think asking the y value of the instance $[x_1, x_2, x_3] = [T, F, T]$ or $[x_1, x_2, x_3] = [T, T, F]$ is informative (there are other instances with the same effect). Because one hypothesis will predict Postive and the other one will predict Negative.

Take the instance $[x_1, x_2, x_3] = [T, F, T]$ for example, here are the predictions generated by VS_2:

hypothesis	prediction
$x_1 \wedge \neg x_2$	Positive
$x_1 \wedge \neg x_3$	Negative

In this case, no matter what the true label is, we can reduce the current size of the Version space by a half, since the two hypotheses disagree.

QUESTION5

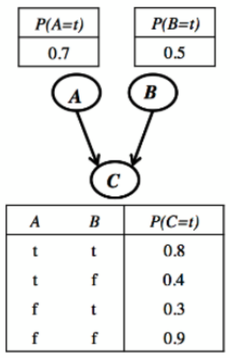
How many mistakes will the Halving algorithm make for this concept class in the worst case? Justify your answer.

By assumption, the concept class that consists of disjunctions of exactly two literals out of n features. And each literal is a feature or its negation, and at most one literal can be negated. Therefore, there are $\binom{n}{2} \times 3 = \frac{3}{2}(n-1)n$ possible hypotheses.
In the worsest case, assume the Halving algorithm always make mistakes. Namely, for each x_i, majority of the hypotheses are inconsistent with x_i. Then we know that at each iteration, the hypotheses space shrink by at least $\frac{1}{2}$. So the havling algorithm will make at most $\lfloor \log_2 |H| \rfloor$ mistakes. Plug in the size of |H|, I get $\lfloor \log_2 \frac{3}{2}(n-1)n \rfloor$.

For our case when n = 3. $\lfloor \log_2 \frac{3}{2}(n-1)n \rfloor = \lfloor \log_2 9 \rfloor = \lfloor 3.1699 \cdots \rfloor = 3$

QUESTION6

Given the initial Bayes net parameters and training set depicted below, show how the network parameters would be updated after one step of the EM procedure. The '?' symbol indicates that the value for the variable A is missing in a given training instance.



P(A=t)
0.7

P(B=t)
0.5

A	B	P(C=t)
t	t	0.8
t	f	0.4
f	t	0.3
f	f	0.9

A	B	C
?	t	t
f	t	f
t	t	f
t	t	t
?	f	t
t	f	f
t	f	t
f	f	f

E-step: fill in the missing values in the A column

For the 1st missing instance:

$$P(a|b,c) = \frac{P(a,b,c)}{P(a,b,c)+P(\neg a,b,c)} = \frac{P(a)P(b)P(c|a,b)}{P(a)P(b)P(c|a,b)+P(\neg a)P(b)P(c|\neg a,b)}$$

$$= \frac{.7\times.5\times.8}{.7\times.5\times.8+.3\times.5\times.3} \approx .86$$

$$P(\neg a|b,c) = 1 - P(a|b,c) \approx .14$$

For the 5th instance, which is also missing:

$$P(a|\neg b,c) = \frac{P(a,\neg b,c)}{P(a,\neg b,c)+P(\neg a,\neg b,c)} = \frac{P(a)P(\neg b)P(c|a,\neg b)}{P(a)P(\neg b)P(c|a,\neg b)+P(\neg a)P(\neg b)P(c|\neg a,\neg b)}$$

$$= \frac{.7\times.5\times.4}{.7\times.5\times.4+.3\times.5\times.9} \approx .51$$

$$P(\neg a|\neg b,c) = 1 - P(a|\neg b,c) \approx .49$$

M-step: re-estimate the conditional probability table using expected counts

$$P(a) = \frac{4+.86+.51}{8} = .67125$$

$$P(c|a,b) = \frac{E\#(c,a,b)}{E\#(a,b)} = \frac{.86+1}{2+.86} \approx .65$$

$$P(c|\neg a,b) = \frac{E\#(c,\neg a,b)}{E\#(\neg a,b)} = \frac{.14}{2+.14} \approx .065$$

$$P(c|a,\neg b) = \frac{E\#(c,a,\neg b)}{E\#(a,\neg b)} = \frac{.51+1}{2+.51} \approx .60$$

$$P(c|\neg a,\neg b) = \frac{E\#(c,\neg a,\neg b)}{E\#(\neg a,\neg b)} = \frac{.49}{1+.49} \approx 33$$

QUESTION7

Consider a learning task in which you are given n features and you want to use a feature selection method along with your learning algorithm. Specifically, suppose you are using forward selection along with k-fold cross validation to evaluate each feature set during the search process. Assume that there are r relevant features, and forward selection stops after selecting r features. Given a single training set, how many models are learned in the process of finding a feature subset of size r?

For forward selection, we begin with the null model (the model with no feature) and add one informative feature at each iteration, where the informativeness of a given feature is determined based on the cross-validated performance.

At the 1st iteration, there are n candidate features, I need to evaluate n models.
At the 2nd iteration, there are n-1 candidate features, I need to evaluate n-1 models.
......
At the r-th iteration, there are n-r candidate features, I need to evaluate n-r models. $(r \leq n)$

Therefore, to choose r features, we need to evaluate $n + (n-1) + \cdots + (n-r)$ models.

When evaluating each model, a k-folds cross validation is required. So in total, we need to fit $k(n + (n-1) + \cdots + (n-r))$ models.

QUESTION8

Now suppose instead you are using backward elimination for the same task. Again, assume that the search process stops after selecting >r features and does not consider feature subsets smaller than this. Given a single training set, how many models are learned in the process of finding a feature subset of size r?

For backward elimination, we begin with the full model (the model with all features) and remove the least informative feature at each iteration, where the informativeness of a given feature is determined based on the cross-validated performance.

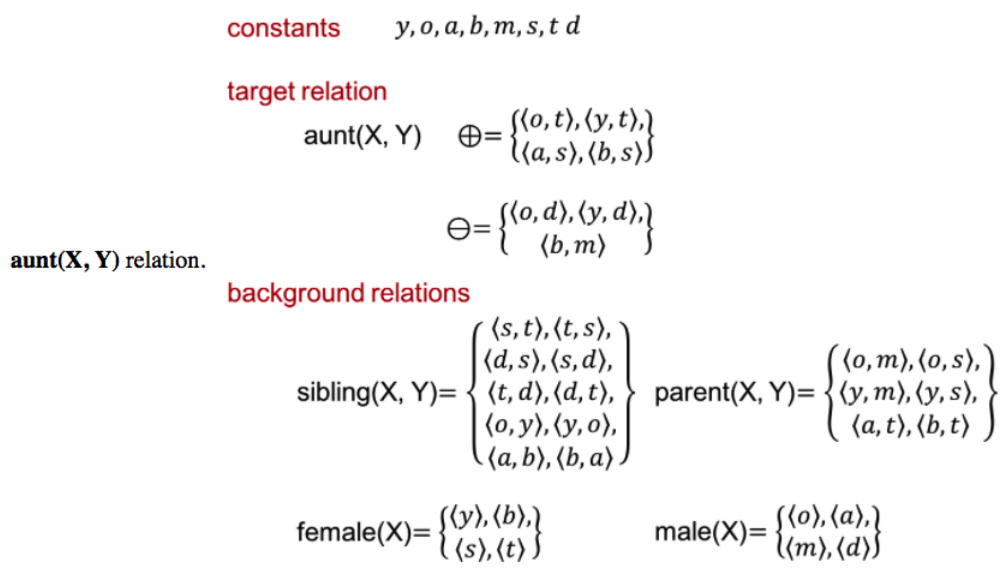
1st iteration: there are n possible features can be removed, need to evaluate n models.
2nd iteration: there are n-1 possible features can be removed, need to evaluate n-1 models.
......
(n-r)-th iteration: there are n-r possible features can be removed, need to evaluate n-(n-r) = r models. ($r \leq n$)

Therefore, to remove (n-r) features, we need to evaluate $n + (n-1) + \cdots + (n-(n-r))$ models.

When evaluating each model, a k-folds cross validation is required. So in total, we need to fit $k(n + (n-1) + \cdots + (n-(n-r)))$ models.

QUESTION9

Consider the relational learning task defined below. List all of the literals that would be considered by FOIL algorithm on the first step of leaning a rule for the aunt(X, Y) relation.



aunt(X, Y) relation.

constants $y, o, a, b, m, s, t\ d$

target relation

aunt(X, Y) $\oplus = \left\{ \begin{matrix} \langle o,t\rangle, \langle y,t\rangle, \\ \langle a,s\rangle, \langle b,s\rangle \end{matrix} \right\}$

$\ominus = \left\{ \begin{matrix} \langle o,d\rangle, \langle y,d\rangle, \\ \langle b,m\rangle \end{matrix} \right\}$

background relations

sibling(X, Y)= $\left\{ \begin{matrix} \langle s,t\rangle, \langle t,s\rangle, \\ \langle d,s\rangle, \langle s,d\rangle, \\ \langle t,d\rangle, \langle d,t\rangle, \\ \langle o,y\rangle, \langle y,o\rangle, \\ \langle a,b\rangle, \langle b,a\rangle \end{matrix} \right\}$ parent(X, Y)= $\left\{ \begin{matrix} \langle o,m\rangle, \langle o,s\rangle, \\ \langle y,m\rangle, \langle y,s\rangle, \\ \langle a,t\rangle, \langle b,t\rangle \end{matrix} \right\}$

female(X)= $\left\{ \begin{matrix} \langle y\rangle, \langle b\rangle, \\ \langle s\rangle, \langle t\rangle \end{matrix} \right\}$ male(X)= $\left\{ \begin{matrix} \langle o\rangle, \langle a\rangle, \\ \langle m\rangle, \langle d\rangle \end{matrix} \right\}$

Here're are the literals that would be considered, for the relation aunt(X, Y), on the 1st step:

$X == Y$
$X \neq Y$
sibling(X, Y)
¬sibling(X, Y)
parent(X, Y)
¬parent(X, Y)
female(X)
¬female(X)
female(Y)
¬female(Y)
male(X)
¬male(X)
male(Y)
¬male(Y)
$X == c,$where c is a constant
$X \neq c,$where c is a constant
$Y == c,$where c is a constant
$Y \neq c,$where c is a constant

QUESTION10

Show the FOIL_gain calculation when sibling(Y, Z) is considered as the first literal to be added to the first rule learning by FOIL. Also show the tuples that are involved in this calculation.

Consider aunt(X, Y) <- sibling(Y, Z) Now we can expand all the tuples:

Positive Tuples:

<o,t>	->	<o,t,s>, <o,t,d>
<y,t>	->	<y,t,s>, <y,t,o>
<a,s>	->	<a,s,t>, <a,s,d>
<b,s>	->	<a,s,t>, <a,s,d>

Negative Tuples:

<o,d>	->	<o,d,s>, <o,d,t>
<y,d>	->	<y,d,s>, <y,d,t>
<b,m>		NA

FOIL evaluates the addition of a literal L to a rule R by

$$\text{FOIL\_Gain}(L,R) = t(\text{Info}(R_0) - \text{Info}(R_1)) = t(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0})$$

where

p_0 = the number of positive tuples covered by R_0 = 4

n_0 = the number of negative tuples covered by R_0 = 3

p_1 = the number of positive tuples covered by $R_0 \wedge L$ = 8

n_1 = the number of negative tuples covered by $R_0 \wedge L$ = 4

t = the number of positive of tuples of R also covered by $R_0 \wedge L = p_1 - p_0 = 4$

Therefore,

$$\text{FOIL\_Gain}(L,R) = 4(\log_2 \frac{2}{3} - \log_2 \frac{4}{7}) \approx .88957$$