6.824 Artificial Intelligence, Fall 1994
Problem Set 3

Due: Beginning of class Wednesday, Oct. 5

Problems 1 and 2 are about fitting a function to data points (regression). More specifically, we consider setting weights so as to minimize error. Consider the following functions.

1. $f_1(x_1,\ x_2,\ x_3) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$
2. $f_2(x_1,\ x_2,\ x_3) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3$
3. $f_3(x_1,\ x_2,\ x_3) = s(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$ where $s(y) = \frac{1}{1+e^{-y}}$. The function s is a "sigmoid" function, i.e., a continuous and differentiable function which is approximately zero for large negative values and approximately one for large positive values, and equal to $\frac{1}{2}$ at $y = 0$.
4. $f_4(x_1,\ x_2,\ x_3) = s(w_1 s(w_2x_1 + w_3x_2) + w_4 s(w_5x_2 + w_6x_3))$ where s is the sigmoid function used in f_3.
5. $f_5(x_1,\ x_2,\ x_3) = w_1 \sin(w_2x_1) + w_3 \cos(w_2x_1) + w_4 \sin(w_5x_2) + w_6 \cos(w_5x_2) + w_7 \sin(w_8x_3) + w_9 \cos(w_8x_3)$
6. $f_6(x_1,\ x_2,\ x_3) = w_1 \sin(x_1) + w_2 \cos(x_1) + w_3 \sin(x_2) + w_4 \cos(x_2) + w_5 \sin(x_3) + w_6 \cos(x_3)$

Suppose you are given a large set of desired input/output pairs, i.e., data points where each pair is of the form $\langle\langle x_{i,1},\ x_{i,2},\ x_{i,3}\rangle,\ y_i\rangle$. For each function f_k given above we want to find values of the weights w_i such that the error

$$E_k(w_1,\ w_2, \ldots,\ w_{n_k}) = \sum_i \frac{1}{2}(y_i - f_k(x_{i,i},\ x_{i,2},\ x_{i,3}))^2$$

is minimized. Note that the value $f_k(x_{i,1},\ x_{i,2},\ x_{i,3})$ implicitly depends on the parameters $w_1,\ w_2,\ \ldots,\ w_{n_k}$.

We will be concerned with two techniques for setting weights so as to minimize error. The first technique is computing weights which exactly minimize the error by solving an appropriate system of linear equations. The second technique is minimizing $E_k(w_1,\ \ldots,\ w_{n_k})$ by gradiant descent.

Problem 1.

Part a. The technique for finding a global optimum based on solving a system of linear equations can be used for function f_k above provided that $f_k(x_1,\ x_2,\ x_3)$ is linear in each implicit parameter w_i. For each of the functions f_1 through f_6 state whether or not that function satisfies this condition.

Parts b and c: Gradiant descent starts with some initial value for the weights and then updates all the weights simultaneously according to the formula.

$$w_i := w_i - \eta \frac{\partial E(w_1,\ \ldots,\ w_{n_k})}{\partial w_i}.$$

This updating is repeated until it no longer changes the weights. The constant η can be empirically adjusted to make gradiant descent work well (find good weights quickly).

Part b. Give a brief description of the geometric interpretation of the above formula. It should involve a hilly terrain and a principle of walking down hill.

I will not give any further clarifications of part b. If the problem seems ill-posed, note that real life is mostly about solving ill-posed problems. You should give what *you* think is the best response.

Part c. Apply gradiant descent to f_5 above. Actually, you need only consider the weight w_2 in the function f_5. Use the gradiant descent formula to give the update rule for w_2. The rule should be of the form $w_2 := w_2 - \eta\delta$ where δ is some expression involving the data points and weights.

Problem 2. For the function f_4 defined above we have that $f_4(x_{i,1},\ x_{i,2},\ x_{i,3}) = s(n_{i,1})$ where

$$n_{i,1} = w_1 s(n_{i,2}) + w_4 s(n_{i,3})$$

$$n_{i,2} = w_2 x_{i,1} + w_3 x_{i,2}$$

$$n_{i,3} = w_5 x_{i,2} + w_6 x_{i,3}.$$

We also define E_i by

$$E_i = \frac{1}{2}(y_i - s(n_{i,1}))^2.$$

Part a. Draw the function $f_4(x_{i,1},\ x_{i,2},\ x_{i,3})$ as a three neuron neural network. Label the lines in your drawing corresponding to the "net neuron inputs" $n_{i,1}$, $n_{i,2}$, and $n_{i,3}$.

Part b. Define

$$\delta_{i,1} = \frac{\partial E_i}{\partial n_{i,1}}$$

$$\delta_{i,3} = \frac{\partial E_i}{\partial n_{i,3}}$$

$$\delta_{i,5} = \frac{\partial E_i}{\partial w_5}$$

Give expressions for $\delta_{i,1}$, $\delta_{i,3}$ and $\delta_{i,5}$. The expression for $\delta_{i,1}$ should be in terms of the "target value" y_i and $n_{i,1}$ (which is a known function of the weights and the input vector $\langle x_{i,1},\ x_{i,2},\ x_{i,3}\rangle$). The expression for $\delta_{i,3}$ should involve $\delta_{i,1}$, w_4, and $n_{i,3}$. The expression for $\delta_{i,5}$ should involve $\delta_{i,3}$ and $x_{i,2}$ δ_5 should involve δ_3 as well as weights and data points.

Part c. Give the update rules for the weights w_4 and w_5 corresponding to the gradiant descent equation given in problem 1. Your update equations should involve a sum over all data points and should be fairly simple expressions involving the quantities defined in parts a and b.

Part d. Decribe how the equations from part c can be viewed as "backpropagating" "blame" from the output of the network back toward the inputs. This is again a vague question. Make your description brief.

Problem 3. This problem is about a Baysian inference approach to concept learning. We consider an instance space where each instance assigns Boolean values to n attributes x_1, ..., x_n. We consider the space of all concepts definable by Boolean expressions over these features. For example, $x_1 \wedge \neg x_2$ is the concept that is true of all instances that assign x_1 the value T and x_2 the value F.

In this problem we will define a probability distribution over concepts using a *stochastic context free grammar*. Consider the (normal) context free grammar given below.

$$F \rightarrow (F \wedge F) \mid (F \vee F) \mid L$$

$$L \rightarrow (\neg A) \mid A$$

$$A \rightarrow x_1 | x_2 | \ldots | x_n$$

The notation $X \rightarrow E_1 | E_2 | \ldots | E_n$ is an abbreviation for the n productions $X \rightarrow E_1$, ..., $X \rightarrow E_n$. Under this grammar the symbol F generates all the Boolean concepts over the attributes x_1, ..., x_n. A stochastic context free grammar is a context free grammar in which each production is associated with a real number in the interval $[0,1]$ and such that for each nonterminal symbol X the sum of the weights of the productions from X is 1. The above grammar can be converted to a stochastic context free grammar by adding weights to the productions as follows.

$$F \xrightarrow{\frac{1}{3}} (F \wedge F)$$

$$F \xrightarrow{\frac{1}{3}} (F \vee F)$$

$$F \xrightarrow{\frac{1}{3}} L$$

$$L \xrightarrow{\frac{1}{3}} (\neg A)$$

$$L \xrightarrow{\frac{2}{3}} A$$

$$A \xrightarrow{\frac{1}{n}} x_1$$

$$\vdots$$

$$A \xrightarrow{\frac{1}{n}} x_n$$

We can now define a random process for rewriting nonterminals in which every time a nonterminal symbol is rewritten we select a production for that nonterminal in such a way that the probability of selecting a given production equals the given weight of that production. In any grammar where each nonterminal generates a nonempty langauge this process determines a probability distribution over the strings in the language generated by each nonterminal symbol (the sum over all strings in the language of the probability of that string equals 1). The above grammar determines a proability distribution over Boolean formulas.

Part a. Suppose that we are trying to learn a target concept where the target concept has been generated at random using the above grammar. After the target concept has been selected, instances for *on which the concept is true* are selected at random, giving each instance on which the concept is true equal probability. Let Ψ_1 be the formula $x_1 \wedge x_2$ and let Ψ_2 be the formula $x_1 \wedge (x_2 \wedge x_3)$. Suppose that we are given a data sequence D consisting of k instances such that both Ψ_1 and Ψ_2 are true of every instance in D. For what values of k and n is $P_r(\Psi_1|D)$ larger than $P_r(\Psi_2|D)$? Justify your answer.

Problem 4. This problem relates probabilities on concepts to PAC learning. In PAC learning we generally do not use prior probabilites on concepts. However, the probabilities defined in problem 3 can be reinterpreted as "concept weights" and used in a dynamic sampling PAC algorithm. Recall that in dynamic sampling PAC algorithms the number of examples needed depends on the concept learned. In lecture we used the length of the concept to determine the number of examples needed to learn that concept. Here we use the weight of the concept instead. Note that shorter concepts have more weight under the distribution defined above. The dynamic sampling PAC algorithm will require more examples to learn low weight (low probability) concepts.

We make the standard assumptions needed for PAC learning — that the examples are drawn independently from a fixed but unknown probability distribution. We also assume a dynamic sampling PAC algorithm — the algorithm will never output a concept (formula) Ψ unless it has seen at least $m(\Psi)$ examples and all the examples it has seen are consistent with Ψ. Here m is some function from formulas to positive integers. It was shown in lecture that the the probability that the algorithm can output a concept with error rate greater than ϵ is no larger than

$$\sum_i (1-\epsilon)^{m(\Psi_i)}$$

which is no larger than

$$\text{I.} \quad \sum_i e^{-\epsilon m(\Psi_i)}$$

where Ψ_0, Ψ_1, Ψ_3, ... is an enumeration of all the concepts. Suppose that we impose the following condition on the function m

$$\text{II.} \quad e^{-\epsilon m(\Psi)} \leq \delta P_r(\Psi)$$

where $P_r(\Psi)$ is the probability (or weight) of Ψ as defined in the problem 3.

Part a. Assuming that condition I is an upper bound on the probability of outputing a concept with error greater than ϵ, explain why condition II ensures the standard PAC condition.

Part b. Convert condition II into a lower bound on $m(\Psi)$ stated in terms of ϵ, δ, and $P_r(\Psi)$.

Part c. In lecture we showed that the condition

$$m(\Psi) \geq \frac{1}{\epsilon}((2|\Psi| + 1)\ln 2 + \ln\frac{1}{\delta})$$

guaranteed the PAC condition in a dynamic sampling algorithm where $|\Psi|$ is the number of bits in the bit string name for the concept Ψ. Compare this concept length sample size requirement with the sample size requirement given in part b — which requires fewer examples? Give an intuitive explanation for your answer.