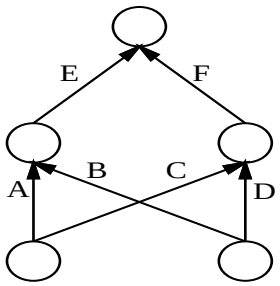
Homework 4
Machine Learning 15-681
Avrim Blum and Tom Mitchell, Fall 1994

Due October 25, 1994

1. Consider the following neural network:



Suppose we begin with all weights having exactly the same value and run backprop on some set of data. Which statements below will be always true? For each, explain your reasoning in at most a few sentences.

(a) Weights A and B never differ.

(b) Weights A and C never differ.

(c) Weights E and F never differ.

Give a short explanation of one good reason NOT to initialize all weights to the same value when running backprop.

2. Consider learning a union of two subintervals of [0,1]. In other words, the target concept is something like:

"positive if in the range [0.2, 0.4] or in the range [0.6, 0.75] and negative otherwise"

(a) Give an algorithm for learning this class in the PAC model, and an upper bound on how many examples you need to see to have error at most ϵ with probability at least 1-δ. You may assume if you want that "for free" you are given one positive example in each subinterval at the start.

(b) What is the VC-dimension of this class?

3. The "winnow" algorithm discussed in class was the following:

   (a) Initialize weights $w_1, ..., w_n$ to 1.

   (b) Given an example x, predict positive if $w_1x_1 + w_2x_2 + ... + w_nx_n \geq n$ and predict negative otherwise.

   (c) If a mistake was made by predicting negative on an example x that was actually positive, then update weights according to: $w_i = 2w_i$, for each w_i for which $x_i = 1$.

   (d) If a mistake was made by predicting positive on an example x that was actually negative, then update weights according to: $w_i = 0.5w_i$, for each w_i for which $x_i = 1$.

   (e) Go to (b).

   We proved that this algorithm makes at most $O(r \log n)$ mistakes when exposed to examples consistent with some disjunction of r out of n features.

   Below are some modifications to the algorithm that we might attempt in order to improve performance. For each one, is the modification "safe" in the sense that it does not cause the algorithm to exceed the above mistake bound? Explain your reasoning in at most a few sentences.

   1. Perform the update in step (d) on all negative examples seen, not just those on which the algorithm has made a mistake.

   2. Perform the update in step (c) on all positive examples seen, not just those on which the algorithm has made a mistake.

   3. When performing the update in step (c), keep doubling the weights until the weights are high enough so that the algorithm would now predict positive on that example. (i.e., keep doubling until $w_1x_1 + ... + w_nx_n \geq n$)