Homework 3

Christopher Mertin

CS6966: Theory of Machine Learning

April 4, 2017

10. Consider the simple experts setting: we have n experts $E_1, \ldots, E_n$, and each one makes a 0/1 prediction each morning. Using these predictions, we need to make a prediction each morning, and at the end of the day we get a loss of 0 if we predicted right, and 1 if we made a mistake. This goes on for T days.

    Consider an algorithm that at every step, goes with the prediction of the "best" (*i.e.* the one with the least mistakes so far) expert so far. Suppose that ties are broken by picking the expert with a smaller index. Given an example in whcih this strategy can be really bad – specifically, the number of mistakes made by the algorithm is roughly a factor of n worse than that of the best expert in hindsight.

    Solution:

11. We saw in class a proof that the VC dimension of the class of n-node, m-edge *threshold* neural networks is $\mathcal{O}\left((m+n)\log(n)\right)$. Let us give a "counting" proof, assuming the weights are binary (0/1). (This is often the power given by VC dimension based proofs – they can "handle" continuous parameters that cause problems for counting arguments).

    (a) Specifically, how many "network layouts" can there be with n nodes and m edges? Show that $\binom{n(n-1)/2}{m}$ is an upper bound.
    Solution:

    (b) Given a network layout, argue that the number of "possible networks" is at most $2^m(n+1)^n$.
    [*Hint:* What can you say about the potential values for the thresholds?]
    Solution:

    (c) Use these to show that the VC dimension of the class of binary-weight, threshold neural networks is $\mathcal{O}\left((m+n)\log(n)\right)$.
    Solution:

12. (Importance of random initialization) Consider a neural network consisting of (resp.) the input layer x, hidden layer y, hidden layer z, followed by the output node f. Suppose that all the nodes in all the layers compute a "standard" sigmoid. Also

suppose that every node in a layer is connected to every node in the next layer (*i.e.*, each layer is fully connected).

Now suppose that all the weights are initialized to 0, and suppose we start performing SGD using backprop, with a fixed learning rate. Show that at every time step, all the edge weights in a layer are equal.

Solution:

13. Let us consider networks in whcih each node computes a rectified linear (ReLU) function, and show how they can comptue very "spiky" functions of the inptue variables. For this exercise, we restrict to one-variable.

    Solution:

    (a) Consider a single input x. Show how to comptue a "triangle wave" using one hidden layer (constant number of nodes) connected to the input, followed by one output f. Formally, we should have $f(x) = 0$ for $x \leq 0$, $f(x) = 2x$ for $0 \leq x \leq 1/2$, $f(x) = 2(1-x)$ for $1/2 \leq x \leq 1$, and $f(x) = 0$ for $x \geq 1$.

    [*Hint:* Choose the thresholds for the ReLU's appropriately]

    Solution:

    (b) What happens if you stack the network on top of itself? (Describe the function obtained).

    [Formally, this means the output of the network you constructed above is fed as the input to an identical network, and we are interested in the final output function.]

    Solution:

    (c) Prove that there is a ReLU network with one input variable x, $2k + \mathcal{O}(1)$ layers, all coefficients and thresholds being constants, that computes a function that has 2^k "peaks" in the interval $[0, 1]$.

    [The function above can be shown to be impossible to approximate using a small depth ReLU network, without an exponential blow-up in the width.]

    Solution:

14. In this exercise, we make a simple observatino that width isn't as "necessary" as depth. Consider a network in which each node comptues a rectified linar (ReLU) unit – specifically the function at each node is of the form $\max\{0, a_1x_1 + a_2x_2 + \cdots + a_mx_m + b\}$, for a node that has inputs $\{x_1, \ldots, x_m\}$. Note that different nodes could have different coefficients and offsets (b above is called the offset).

    Consider a network with one input x, connect to n nodes in a hidden layer, which are in turn connected to the output node, denoted f. Show that one can construct a depth $n + \mathcal{O}(1)$ network, with just 3 nodes in each layer, to compute the same f.

    [*Hint:* Three nodes allow you to "carry over" the input; ReLU's are important for this]

    Solution: