Problem 10

Consider the simple experts setting: we have n experts $E_1, \dots, E_n$, 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. Give an example in which this strategy can be really bad – specifically, the number of mistakes made by the algorithm is roughly a factor n worse than that of the best expert in hindsight.

We will show that given n experts and this algorithm, we can construct an adversarial scenario where the algorithm makes a factor of n more mistakes than the best expert. Consider the scenario where the true label is 0 for all T days. Each experts prediction is based on the following function of the current day t

$$prediction_i(t) = \begin{cases} 1 & t \bmod n = i - 1 \\ 0 & otherwise \end{cases}$$

In this scenario the algorithm will make T mistakes (always predicts wrong) and the best expert will be wrong T/n number of times.

To give a concrete example, consider the scenario where we have 3 experts and we run the algorithm for 9 days.

expert / t	0	1	2	3	4	5	6	7	8
E_1	1	0	0	1	0	0	1	0	0
E_2	0	1	0	0	1	0	0	1	0
E_3	0	0	1	0	0	1	0	0	1
true prediction	0	0	0	0	0	0	0	0	0
selected expert	E_1	E_2	E_3	E_1	E_2	E_3	E_1	E_2	E_3

We can see that in this example the algorithm always makes a mistake. Each expert is tied for being the best expert as they are each wrong T/n or 3 times. We have shown that this algorithm can make a factor of n more mistakes than the best expert.

Problem 11

We saw in class a proof that the VC dimension of the class of n-node, m-edge *threshold* neural networks is $O((m+n)\log n)$. 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.

$\square$

(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?]

$\square$

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

$\square$

Problem 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.

$\square$

Problem 13

Let us consider networks in which each node computes a rectified linear (ReLU) function (described in the next problem), and show how they can compute very 'spiky' functions of the input variables. For this exercise, we restrict to one-variable.

(a) Consider a single (real valued) input x. Show how to compute 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 and coefficients for the ReLU's appropriately.] [HINT2: play with a few ReLU networks, and try to plot the output as a function of the input.]

$\square$

(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.]

$\square$

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

$\square$

Problem 14

In this exercise, we make a simple observation that width isn't as "necessary" as depth. Consider a network in which each node computes a rectified linear (ReLU) unit – specifically the function at each node is of the form $max\{0, a_1y_1 + a_2y_2 + ... + a_my_m + b\}$, for a node that has inputs $y1, ... , ym$. Note that different nodes could have different coefficients and offsets (b above is called the offset).

Consider a network with one (real valued) input x, connected 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 $n + O(1)$ 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.]

$\square$

Collaboration

I collaborated with Sierra Allred, Maks Cegielski-Johnson, and Dietrich Geisler on problem 11, problem 12, and problem 13.