IEOR 4004: Optimization Models and methods

Lecture : Introduction to prediction using neural networks

Suppose we have data for $N = 10000$ people. The data contains the height (H), age (A), weight (W) and resting heart rate (R). We would like to use this data to predict R as a function of (H, A, W). The plan is to use the N data points so as to construct (or "train") a function $f(H, A, W)$ that will be used as the predictor for R that we will apply when we get data for people outside of this training set of N people.

To do so we will choose a function f to be chosen from a certain family $\mathcal{F}$ that we will describe below. Postponing this point, we will compute f by (approximately) solving an **optimization problem**. To describe this optimization problem, suppose we indicate the training data as (H_i, A_i, W_i, R_i), $i = 1, \ldots, N$. The optimization problem is of the form

$$\min \sum_{i=1}^{N} (R_i \; - \; f(H_i, A_i, W_i))^2 \tag{1a}$$

$$\text{s.t. } f \in \mathcal{F}. \tag{1b}$$

What this definition states is that we want to select a function from the family $\mathcal{F}$ so that the "score" or (more properly stated) *loss function* given by (1a) is minimized. As an example, suppose that we wanted to apply multilinear regression. In that setting we would want to predict R as an affine function of (H, A, W). In that case we would be searching for scalars α, β, γ and δ so as to predict R through the formula

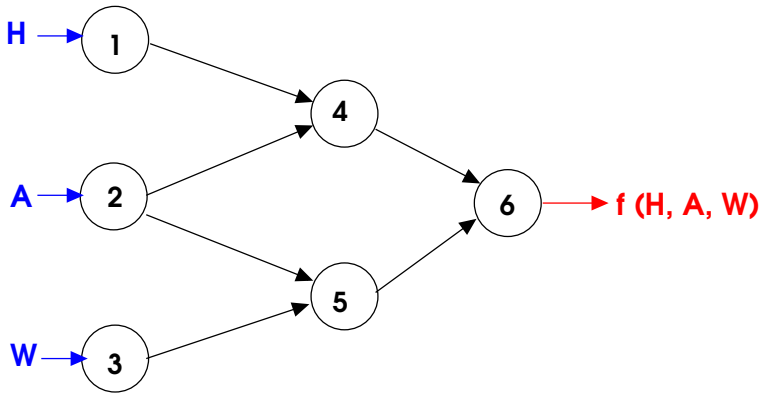
$$\alpha H + \beta A + \gamma W + \delta$$

In this example, problem (1) becomes

$$\min \sum_{i=1}^{N} (R_i \; - \; (\alpha H_i + \beta A_i + \gamma W_i + \delta))^2 \tag{2a}$$

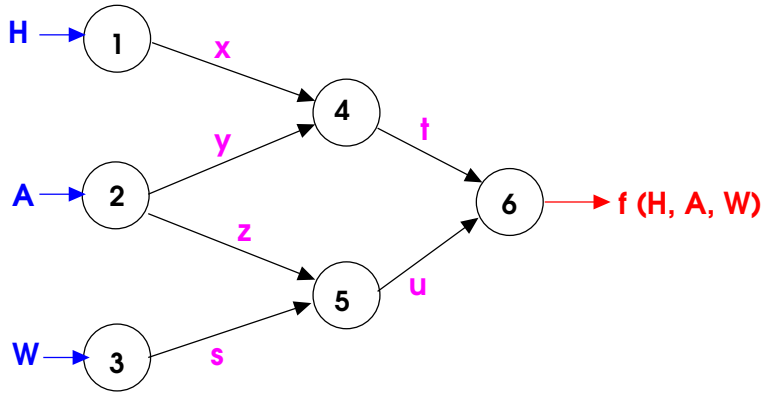
$$\text{s.t. } \alpha, \beta, \gamma, \delta \text{ scalars} \tag{2b}$$

When considering neural networks we use a different (and more complex) approach. Consider a network as given in the following figure. The network in this figure operates as follows. First we input H, A, W into nodes 1, 2 and 3.



The output, $f(H, A, W)$ will come out of node 6. In between the inputs and outputs we have the network, which will *flow* the values that we input and modify them, from node to node. The modifications take two forms.

First, at each node j there is an "activation" function σ_j. For example, the activation function at every node j could be $\sigma_j(t) = \ln(|t| + 1)$. For a given training exercise we assume that these functions are given – they are **not** part of the selection that we make in solving optimization problem (1). Second, and most important, on each arc of the network we have a parameter, or *weight*, as shown in the following figure: This figure shows six parameters



(x, y, z, s, t, u), one per each arc. **These are the parameters that we will select in optimization problem (1), i.e. these are the variables in the optimization.** The network operates by *layers*. First we have the (input) layer made up by nodes $1, 2, 3$. Then we have the (inner) layer made up by nodes $4, 5$. And finally we have the output layer made up by node 6. When we process a layer, each node in that layer takes input from the arcs pointing into this node, and then uses the weight of those arcs plus its own activating function so as to compute the values on the arcs coming out of this node.

Example. Suppose again that we use $\sigma_j(t) = \ln(|t| + 1)$ as indicated above. Moreover suppose

$$(x, y, z, s, t, u) = (0.1, 0.2, 0.3, 0.4, 10, 60).$$

With this choice of values we have specified the function f as we will describe next. Suppose we want to compute $f(6, 40, 150)$. Then we proceed as follows.

Layer 1. The input to node 1 is 6. So the output of node 1 is ≈ 1.95. Likewise the output of nodes 2 and 3 equals, approximately, 3.71 and 5.02.

Layer 2. Consider node 4. Its input from node 1 equals 1.95, and from node 2 equals 3.71. Then the output from node 4 equals

$$\sigma_4(1.95x + 3.71y) = \sigma_4(1.95 * 0.1 + 3.71 * 0.2) \approx \sigma_4(0.94) \approx 0.66$$

And at node 5, the input from node 2 equals 3.71 while the input from node 3 equals 5.02. So the output from node 5 equals

$$\sigma_5(3.71z + 5.02s) = \sigma_5(3.71 * 0.3 + 5.02 * 0.4) \approx \sigma_5(3.118) \approx 1.137$$

Node 6. The input from node 4 is approximately 0.66 and the input from node 5 is approximately 1.137. So the output from node 6, which is our prediction, is

$$\sigma_6(10 * 0.66 + 60 * 1.137) \approx 4.33$$

We can see that this is a poor prediction for heart rate! That is to say, the choice $(x, y, z, s, t, u) = (0.1, 0.2, 0.3, 0.4, 10, 60)$ is **not** good. However ... what if we actually want to optimally choose (x, y, z, s, t, u)?

Gradient descent and the chain rule

In order to compute (or approximate) the "best" choice for (x, y, z, s, t, u) we will apply the gradient descent method to problem (1), where the family $\mathcal{F}$ of functions is all the functions f as in the above examples (one function per 6-tuple (x, y, z, s, t, u)).

To proceed in a formal way, instead of simply writing $f(H_i, A_i, W_i)$ in (1), we will use a different notation. Note that for a given index i the computation of $f(H_i, A_i, W_i)$ uses our current choice of the six-tuple (x, y, z, s, t, u), i.e. it is a function of (x, y, z, s, t, u). So we will write

$$g_i(x, y, z, s, t, u) \doteq f(H_i, A_i, W_i)$$

Using this notation, the loss function can be written

$$L(x, y, z, s, t, u) \doteq \sum_{i=1}^{N} (R_i - g_i(x, y, z, s, t, u))^2 \tag{3a}$$

We want to choose (x, y, z, s, t, u) to as to minimize $L(x, y, z, s, t, u)$. It is simple enough to compute the value $L(x, y, z, s, t, u)$: we use formula (3). For each i we compute $f(H_i, A_i, W_i)$ by operating the network as we described above. It is important to formally describe how we do this. Thus, for each node $j = 1, 2, \ldots, 6$ let us define

$\theta_{i,j}$ = the output of node j, when the input to the network is (H_i, A_i, W_i) and we use weights (x, y, z, s, t, u)

[We should really say $\theta_{i,j}(x, y, z, s, t, u)$ but we want to save space.] Using this notation, we have

1. $f(H_i, A_i, W_i) = g_i(x, y, z, s, t, u) = \theta_{i,6}(x, y, z, s, t, u)$ i.e. the function f is the output of node 6. Moreover

$$\theta_{i,6} = \sigma_6(t \times \theta_{i,4} + u \times \theta_{i,5}) \tag{4}$$

2. For the second layer we have

$$\theta_{i,4} = \sigma_4(x \times \theta_{i,1} + y \times \theta_{i,2}) \tag{5}$$
$$\theta_{i,5} = \sigma_5(z \times \theta_{i,2} + s \times \theta_{i,3}) \tag{6}$$

3. Finally:

$$\theta_{i,1} = \sigma_1(H_i),\ \theta_{i,2} = \sigma_2(A_i),\ \theta_{i,3} = \sigma_3(W_i) \tag{7}$$

How about $\nabla L(x, y, z, s, t, u)$? We need this quantity in order to run a gradient descent method.

- We will need to compute all the partial derivatives $\frac{\partial L(x,y,z,s,t,u)}{\partial v}$ where "v" is each of x, y, z, s, t, u. Using **the chain rule**, we will always get a formula of the form

$$-2\sum_{i=1}^{N} (R_i - g_i(x, y, z, s, t, u)) \frac{\partial g_i(x, y, z, s, t, u)}{\partial v}$$

- Let us first apply this principle to compute $\frac{\partial L(x,y,z,s,t,u)}{\partial u}$; that is to say, let us compute $\frac{\partial g_i(x,y,z,s,t,u)}{\partial u}$. Using (4), **and the chain rule**,

$$\frac{\partial g_i(x, y, z, s, t, u)}{\partial u} = \frac{\partial \theta_{i,6}}{\partial u} = \sigma_6'(t \times \theta_{i,4} + u \times \theta_{i,5})\theta_{i,5}$$

- Similarly,

$$\frac{\partial g_i(x, y, z, s, t, u)}{\partial t} = \frac{\partial \theta_{i,6}}{\partial t} = \sigma_6'(t \times \theta_{i,4} + u \times \theta_{i,5})\theta_{i,4}$$

- How about $\frac{\partial g_i(x,y,z,s,t,u)}{\partial s} = \frac{\partial \theta_{i,6}}{\partial s}$? Again the chain rule:

$$\frac{\partial \theta_{i,6}}{\partial s} = \sigma_6'(t \times \theta_{i,4} + u \times \theta_{i,5}) \frac{\partial (t \times \theta_{i,4} + u \times \theta_{i,5})}{\partial s}$$

According to the structure of the network, $\theta_{i,4}$ is **independent** of s. So we get

$$\frac{\partial \theta_{i,6}}{\partial s} \;=\; \sigma_6'(t \times \theta_{i,4} \;+\; u \times \theta_{i,5}) \frac{\partial (u \times \theta_{i,5})}{\partial s}$$

and using the chain rule we get

$$\frac{\partial \theta_{i,6}}{\partial s} \;=\; \sigma_6'(t \times \theta_{i,4} \;+\; u \times \theta_{i,5})\, u\, \frac{\partial \theta_{i,5}}{\partial s}$$

Applying formula (6) we obtain

$$\frac{\partial \theta_{i,6}}{\partial s} \;=\; \sigma_6'(t \times \theta_{i,4} \;+\; u \times \theta_{i,5})\, u\, \sigma_5'(z \times \theta_{i,2} + s \times \theta_{i,3})\, \theta_{i,3}$$

Exercise: compute the remaining partials.

Which are useful activation functions? (Examples from http://www.deeplearningbook.org/)

- "logistic sigmoid": $\sigma(x) \;=\; \frac{1}{1+e^{-x}}$
- "softplus": $\sigma(x) \;=\; \ln(1+e^{x})$.

STue.Mar.27.172849.2018@blacknwhite