Neural Networks for Superforecasting

Devanshu Agrawal

April 2, 2016

A neural network (modified for superforecasting) is a directed acyclic graph with exactly one root node that can therefore be organized into layers. Every arc is weighted by conditional probabilities. Every node receives as "input" a weighted average of the outputs of its parent nodes. A node that receives an input is said to be "activated". Every node "fires" an "output" that is a function of its input. The function that maps an input of a node to its output is a sigmoid "activation function".

Suppose that we provide inputs for the leaf nodes of a neural network. The leaf nodes are activated and fire. This activates the nodes in the next layer and causes them to fire. Activation and firing cascade layer-after-layer until the root node fires a final output. This is how a neural network runs.

A neural network is therefore a function based on a set of parameters. Suppose we have a "loss function" that assigns a value to every set of parameters. We often want to adjust the parameters of the hneural network such that the loss function attains a minimum value. We do this by differentiating the loss function with respect to the parameters of the neural network and then adjusting the parameters in the appropriate direction.

1 The Sigmoid Activation Function

All inputs and outputs of nodes are probabilities. We therefore desire an activation function $\sigma : [0, 1] \mapsto [0, 1]$ that satisfies the following properties:

1. σ is a continuous and strictly increasing surjection.
2. σ is odd about the point $\left(\frac{1}{2}, \frac{1}{2}\right)$; i.e.,

$$\sigma(1 - x) = 1 - \sigma(x), \tag{1}$$

for all $x \in [0,1]$.

3. σ is differentiable with vanishing half-derivatives at $x = 0$ and $x = 1$.

Given any $s \geq 1$, we choose an activation function $\sigma(\cdot; s) : [0,1] \mapsto [0,1]$ defined by

$$\sigma(x) = \begin{cases} \frac{1}{2} + \frac{1}{2}\tanh[s \operatorname{arctanh}(2x-1)], & \text{if } x \in (0,1) \\ 0, & \text{if } x = 0 \\ 1, & \text{if } x = 1. \end{cases} \tag{2}$$

It is clear that σ is continuous and strictly increasing on $(0,1)$. We leave it to the reader to verify the limits

$$\lim_{x \to 0^+} \sigma(x) = 0$$
$$\lim_{x \to 1^-} \sigma(x) = 1.$$

We also leave it to the reader to verify that

$$\sigma(1-x) = 1 - \sigma(x),$$

for all $x \in [0,1]$. Therefore, σ is a continuous and strictly increasing sigmoid surjection on $[0,1]$.

The derivative of σ is given by

$$\begin{aligned} \sigma'(x) &= \frac{1}{2}\operatorname{sech}^2[s \operatorname{arctanh}(2x-1)] \frac{2s}{1-(2x-1)^2} \\ &= \left(1 - \tanh^2[s \operatorname{arctanh}(2x-1)]\right) \frac{s}{1-(2x-1)^2} \\ &= [1-(2\sigma(x)-1)^2] \frac{s}{1-(2x-1)^2} \\ &= s\left[\frac{1-(2\sigma(x)-1)^2}{1-(2x-1)^2}\right]. \end{aligned} \tag{3}$$

Since σ is a sigmoid function, then $\sigma\left(\frac{1}{2}\right) = \frac{1}{2}$. It follows that $\sigma'(\frac{1}{2}) = s$. That is, s is the slope of $y = \sigma(x)$ at the inflection point $\left(\frac{1}{2}, \frac{1}{2}\right)$. We call s the *steepness parameter*. Indeed, as s increases, the curve $y = \sigma(x)$ appears more "sigmoid". We leave it to the reader to show that $\sigma(x; s = 1) = x$; i.e., if $s = 1$, then $y = \sigma(x)$ is linear.

The numerator and denominator of Equation 3 both vanish as $x \to 0$ from the right. Applying $L'Hopital's Rule$, we have

$$\begin{aligned}\lim_{x\to 0^+} \sigma'(x) &= s \lim_{x\to 0^+} \frac{-2(2\sigma(x)-1)2\sigma'(x)}{-2(2x-1)2} \\ &= s \lim_{x\to 0^+} \frac{2\sigma(x)-1}{2x-1} \lim_{x\to 0^+} \sigma'(x) \\ &= s \lim_{x\to 0^+} \sigma'(x).\end{aligned}$$

Hence, we have

$$(s-1) \lim_{x\to 0^+} \sigma'(x) = 0.$$

If $s = 1$, then σ is linear so that $\sigma'(x) = 1$ for all $x \in [0, 1]$. If $s > 1$, then

$$\lim_{x\to 0^+} \sigma'(x) = 0.$$

Differentiating Equation 1 gives us

$$\sigma'(1-x) = \sigma'(x),$$

which immediately implies that

$$\lim_{x\to 1^-} \sigma'(x) = 0.$$

Therefore, σ has vanishing half-derivatives at $x = 0$ and $x = 1$.

We leave it to the reader to show that the inverse $\sigma^{-1} : [0, 1] \mapsto [0, 1]$ is given by

$$\sigma^{-1}(y; s) = \sigma\left(y; \frac{1}{s}\right). \tag{4}$$

We define the function $\tau : [0, 1] \mapsto [0, s]$ by

$$\tau(y) = \sigma'(\sigma^{-1}(y)) \tag{5}$$

This will be useful later.

We will use the sigmoid activation function to "extremize" probabilities; i.e., the function σ takes as input a conservative probability estimate that may be close to 0.5 and transforms it to a more decisive estimate that is closer to either 0 or 1. We say that the input of σ is an "unextremized" probability and that its output is an "extremized" probability.

2 The Network Architecture

Let $G = (V, E)$ be a directed acyclic graph with exactly one root node. Let $\text{In}(A)$ and $\text{Out}(A)$ be the sets of in-neighbors and out-neighbors of a node $A \in V$ respectively. Every node $A \in V$ is a binary random variable where $A = 1$ means event A happens and $A = 0$ means event A does not happen. Let $x_A = \Pr(A = 1)$ be the "unextermized" probability that A happens, and let

$$y_A = \sigma(x_A) \tag{6}$$

be the "extremized" probability that A happens. Define the vector

$$\mathbf{y}_A = \begin{bmatrix} 1 - y_A \\ y_A \end{bmatrix} = \begin{bmatrix} 1 - \sigma(x_A) \\ \sigma(x_A) \end{bmatrix} = \begin{bmatrix} \sigma(1 - x_A) \\ \sigma(x_A) \end{bmatrix} = \begin{bmatrix} \sigma(\Pr(A = 0)) \\ \sigma(\Pr(A = 1)) \end{bmatrix}. \tag{7}$$

We think of x_A as the "input" of node A and y_A as the "output" of node A.

Every arc $(B, A) \in E$ is weighted by the vector

$$\mathbf{w}_{AB} = \begin{bmatrix} w^0_{AB} & w^1_{AB} \end{bmatrix} = \begin{bmatrix} \Pr(A = 1 \mid B = 0) & \Pr(A = 1 \mid B = 1) \end{bmatrix}. \tag{8}$$

(Note that $\mathbf{w}_{AB}$ is the weight on the arc from node B to node A). Observe that

$$\begin{aligned} \Pr(A = 1) &= \Pr(A = 1, B = 0) + \Pr(A = 1, B = 1) \\ &= \Pr(A = 1 \mid B = 0) \Pr(B = 0) + \Pr(A = 1 \mid B = 1) \Pr(B = 1) \\ &= \begin{bmatrix} \Pr(A = 1 \mid B = 0) & \Pr(A = 1 \mid B = 1) \end{bmatrix} \begin{bmatrix} \Pr(B = 0) \\ \Pr(B = 1) \end{bmatrix}. \end{aligned}$$

If the entries of the column vector in the last line are extremized probabilities, then we have that

$$P(A = 1) = \mathbf{w}_{AB} \mathbf{y}_B.$$

We can compute $\Pr(A = 1)$ in this way for every $B \in \text{In}(A)$. We then define x_A to be the average

$$x_A = \frac{1}{|\text{In}(A)|} \sum_{B \in \text{In}(A)} \mathbf{w}_{AB} \mathbf{y}_B. \tag{9}$$

More explicitly,

$$\begin{aligned} x_A &= \frac{1}{|\operatorname{In}(A)|} \sum_{B \in \operatorname{In}(A)} \begin{bmatrix} w_{AB}^0 & w_{AB}^1 \end{bmatrix} \begin{bmatrix} 1 - y_B \\ y_B \end{bmatrix} \\ &= \frac{1}{|\operatorname{In}(A)|} \sum_{B \in \operatorname{In}(A)} [w_{AB}^0 (1 - y_B) + w_{AB}^1 y_B] \\ &= \frac{1}{|\operatorname{In}(A)|} \sum_{B \in \operatorname{In}(A)} [w_{AB}^0 + (w_{AB}^1 - w_{AB}^0) y_b]. \end{aligned} \tag{10}$$

3 Running the Network

Let $R \in V$ be the unique root node of G. Suppose that the input probabilities x_A are given for all leaf nodes $A \in V$. Suppose also that the weights $\mathbf{w}_{AB}$ are given for all arcs $(B, A) \in E$. It is then possible to compute the extremized probability $y_R = \Pr(R = 1)$. This requires that we first organize the graph G to have a K-partite architecture. Since G is a directed acyclic graph, then this is always possible. We partition the set of nodes V into subsets or *layers* $V^1, V^2, \ldots, V^K$ where V^1 is the set of leaf nodes in G, $V^K = \{R\}$, and for every arc $(B, A) \in E$ with $B \in V^\ell$ and $A \in V^k$, we have $\ell < k$. In other words, the layers are defined such that every arc points from one node to another node that is closer to the root node. This implies that all in-neighbors of a non-leaf node must be contained in preceding layers; for all nodes $A \in V^k$ with $k > 1$, we have

$$\operatorname{In}(A) \subseteq \bigcup_{i=1}^{k-1} V^i.$$

We organize G into layers using the following recursion: Let V^1 be the set of all leaf nodes in G. For $k \geq 2$, let G_k be the subgraph of G generated by the set of nodes

$$V \setminus \left(\bigcup_{i=1}^{k-1} V^i \right).$$

Let V^k be the set of all leaf nodes in G_k.

We can now compute y_R by the following recursion: We are given x_A for all leaf nodes $A \in V^1$. Given x_A for all $A \in V^k$, we can compute y_A for all

$A \in V^k$ using Equation 6. Given y_B for all $B \in \bigcup_{i=1}^{K} V^i$, we can compute x_A for all $A \in V^{k+1}$ using Equation 9. Proceeding in this way ultimately yields the output y_R.

4 Back Propagation

Let $M \in V$ be a node. Let $G_M = (V_m, E_m)$ be the subgraph of G generated by M and all of its ancestor nodes. Note that M is the unique root of G_M. Let $L(y_M)$ be a real-valued loss function of y_M. By Section 3, L also implicitly depends on y_A for all nodes $A \in V_M$ and on $\mathbf{w}_{AB}$ for all arcs $(B, A) \in E_M$. Our goal is to adjust the parameters of the network such that the loss function L is minimized. This requires that we first compute the gradient of L.

We would like to compute how the loss function L changes with respect to the parameters of the subgraph G_M. Let $B \in V_M$. By the chain rule, we have

$$\begin{aligned}\frac{\partial L}{\partial y_B} &= \sum_{A \in \mathrm{Out}(B)} \frac{\partial x_A}{\partial y_B} \frac{\partial L}{\partial x_A} \\ &= \sum_{A \in \mathrm{Out}(B)} \frac{\partial x_A}{\partial y_B} \frac{\partial y_A}{\partial x_A} \frac{\partial L}{\partial y_A}.\end{aligned}$$

By Equation 10, we have

$$\frac{\partial x_A}{\partial y_B} = \frac{1}{|\operatorname{In}(A)|}(w^1_{AB} - w^0_{AB}).$$

By Equation 6, we have

$$\frac{\partial y_A}{\partial x_A} = \sigma'(x_A) = \tau(y_A).$$

We therefore have

$$\frac{\partial L}{\partial y_B} = \sum_{A \in \mathrm{Out}(B)} \frac{1}{|\operatorname{In}(A)|}(w^1_{AB} - w^0_{AB})\tau(y_A)\frac{\partial L}{\partial y_A}. \tag{11}$$

This results in the following recursion: Suppose we run the network so that y_A is known for all nodes $A \in V_M$. If we know $\frac{\partial L}{\partial y_A}$ for all nodes A in a

given layer of G_M, then we can use Equation 11 to compute $\frac{\partial L}{\partial y_B}$ for all nodes B contained in the layer of G_M preceding the one containing A. Moreover, suppose we first compute $\frac{\partial L}{\partial y_M}$ (the change of L with respect to the root node M of G_M). Then, we can compute $\frac{\partial L}{\partial y_A}$ for all nodes $A \in G_M$ by propagating back from layer-to-layer in G_M. This implementation of the chain rule on a K-partite graph is called *back propagation*.

Let $(B, A) \in E_M$. By the chain rule, we have

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}_{AB}} &= \frac{\partial y_A}{\partial \mathbf{w}_{AB}} \frac{\partial L}{\partial y_A} \\ &= \frac{\partial x_A}{\partial \mathbf{w}_{AB}} \frac{\partial y_A}{\partial x_A} \frac{\partial L}{\partial y_A}.\end{aligned}$$

By Equation 9, we have

$$\frac{\partial x_A}{\partial \mathbf{w}_{AB}} = \frac{1}{|\operatorname{In}(A)|}\mathbf{y}_B^\top = \frac{1}{|\operatorname{In}(A)|}\begin{bmatrix}1 - y_B & y_B\end{bmatrix}.$$

By Equation 6, we have

$$\frac{\partial y_A}{\partial x_A} = \sigma'(x_A) = \tau(y_A).$$

We therefore have

$$\frac{\partial L}{\partial \mathbf{w}_{AB}} = \frac{1}{|\operatorname{In}(A)|}\mathbf{y}_B^\top \tau(y_A) \frac{\partial L}{\partial y_A}. \tag{12}$$

It is therefore possible to compute $\frac{\partial L}{\partial \mathbf{w}_{AB}}$ once we have computed $\frac{\partial L}{\partial y_A}$.

We want to adjust the parameters of the network such that the loss function L decreases. We perform the adjustment in small steps. In each step, we adjust the parameters in the direction in which L decreases the fastest. This direction is the negative gradient of L. Let $\alpha > 0$ be a small number called the *learning rate*. For all arcs $(B, A) \in E_M$, we adjust the weight $\mathbf{w}_{AB}$ by the change

$$\Delta \mathbf{w}_{AB} = -\alpha \frac{\partial L}{\partial \mathbf{w}_{AB}}. \tag{13}$$

This method of adjusting parameters to minimize L is called *gradient descent*.

5 Decreasing Error

Let R be the unique root node of G. Let $y_R = \Pr(R = 1)$ be the extremized probability estimate for the occurrence of R is computed by G. Suppose we observe an instance of R either happening or not happening in the real world. Let $y = 1$ if we observed R happening and $y = 0$ if not. We would like to adjust the parameters of the network G such that the estimate y_R is closer to the true value y. Define the mean-squared-error loss function

$$L_E(y_R) = \frac{1}{2}(y_R - y)^2. \tag{14}$$

We would like to decrease $L_E(y_R)$. Since R is the root node of the entire graph G, then we perform back propagation and gradient descent on the entire graph G to decrease $L_E(y_R)$. These methods require that we compute $\frac{\partial L_E}{\partial y_R}$, which is simply

$$\frac{\partial L_E}{\partial y_R} = y_R - y.$$

Back propagation and gradient descent proceed as usual.

6 Decreasing Variance

Recall that x_A is the average value of $\Pr(A = 1)$ computed along all incoming arcs of node A. The rules of probability demand that

$$\Pr(A = 1) = \mathbf{w}_{AB}\mathbf{y}_B,$$

for all $B \in \mathrm{In}(A)$. We would therefore like to minimize the variance in the computation of the average x_A. For every non-leaf node $A \in V$, we define a loss function

$$L_{V_A} = \frac{1}{2|\,\mathrm{In}(A)|} \sum_{B \in \mathrm{In}(A)} (\mathbf{w}_{AB}\mathbf{y}_B - x_A)^2, \tag{15}$$

which is one-half the variance associated to x_A. Note that L_{V_A} depends only on the parameters attached to the subgraph G_A generated by node A and its ancestors. We want to minimize the total variance loss function

$$L_V = \sum_{\{A \in V : \mathrm{In}(A) \neq \emptyset\}} L_{V_A}. \tag{16}$$

We accomplish this by back propagation and gradient descent. For every non-leaf node $A \in V$, we calculate $\frac{\partial L_{V_A}}{\partial \mathbf{w}_{BC}}$ for every arc (C, B) contained in the subgraph $G_A = (V_A, E_A)$ generated by A and its ancestors. We then have

$$\frac{\partial L_V}{\partial \mathbf{w}_{BC}} = \sum_{\{A \in V : (B,C) \in E_A\}} \frac{\partial L_{V_A}}{\partial \mathbf{w}_{BC}}.$$

We can use these derivatives to update the weights $\mathbf{w}_{BC}$. We can update the input probabilities x_A for leaf nodes $A \in V$ similarly.

Back propagation requires that we compute the initial derivative $\frac{\partial L_{V_A}}{\partial y_A}$. But L_{V_A} cannot be expressed as a function strictly in the variable y_A. We must instead compute $\frac{\partial L_{V_A}}{\partial \mathbf{w}_{AB}}$ and $\frac{\partial L_{V_A}}{\partial y_B}$ for all $B \in \mathrm{In}(A)$ as the initial derivatives for back propagation. Letting δ_{BC} denote the Kronecker delta, we have

$$\begin{aligned}
\frac{\partial L_{V_A}}{\partial \mathbf{w}_{AB}} &= \frac{1}{2|\operatorname{In}(A)|} \sum_{C \in \mathrm{In}(A)} \frac{\partial}{\partial \mathbf{w}_{AB}} (\mathbf{w}_{AC}\mathbf{y}_C - x_A)^2 \\
&= \frac{1}{|\operatorname{In}(A)|} \sum_{C \in \mathrm{In}(A)} (\mathbf{w}_{AC}\mathbf{y}_C - x_A) \left(\delta_{BC}\mathbf{y}_C^\top - \frac{\partial x_A}{\partial \mathbf{w}_{AB}} \right) \\
&= \frac{1}{|\operatorname{In}(A)|} \sum_{C \in \mathrm{In}(A)} (\mathbf{w}_{AC}\mathbf{y}_C - x_A)(\delta_{BC}\mathbf{y}_C^\top - \mathbf{y}_B^\top) \\
&= \frac{1}{|\operatorname{In}(A)} \sum_{C \in \mathrm{In}(A)} (\mathbf{w}_{AC}\mathbf{y}_C - x_A)\delta_{BC}\mathbf{y}_B^\top - \frac{1}{|\operatorname{In}(A)|} \sum_{C \in \mathrm{In}(A)} (\mathbf{w}_{AC}\mathbf{y}_C - x_A) \\
&= \frac{1}{|\operatorname{In}(A)|}(\mathbf{w}_{AB}\mathbf{y}_B - x_A)\mathbf{y}_B^\top - \left[\frac{1}{|\operatorname{In}(A)|} \sum_{C \in \mathrm{In}(A)} \mathbf{w}_{AC}\mathbf{y}_C - \frac{1}{|\operatorname{In}(A)|} \sum_{C \in \mathrm{In}(A)} x_A \right] \\
&= \frac{1}{|\operatorname{In}(A)|}(\mathbf{w}_{AB}\mathbf{y}_B - x_A)\mathbf{y}_B^\top - (x_A - x_A) \\
&= \frac{1}{|\operatorname{In}(A)|}(\mathbf{w}_{AB}\mathbf{y}_B - x_A)\mathbf{y}_B^\top.
\end{aligned}$$

It can similarly be shown that

$$\frac{\partial L_{V_A}}{\partial y_B} = \frac{1}{|\operatorname{In}(A)|}(\mathbf{w}_{AB}\mathbf{y}_B - x_A)(w_{AB}^1 - w_{AB}^0).$$

Given these initial derivatives, back propagation and gradient descent proceed as usual.