Smaller Standard Deviation for Initial Weights Improves Neural Networks Performance: A Theoretical Explanation of Unexpected Simulation Results

Diego Aguirre, Philip Hassoun, Rafael Lopez
Crystal Serrano, Marcoantonio R. Soto, Andrea Torres,
and Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso, El Paso, TX 79968, USA
daguirre6@utep.edu, pchassoun@miners.utep.edu
relopez6@miners.utep.edu, cserrano5@miners.utep.edu
mrsoto3@miners.utep.edu, aftorres@miners.utep.edu
vladik@utep.edu

Selecting initial weights is important. In deep learning, a neural network that classifies into c classes starts with the inputs $x_1, \ldots, x_v$. On each layer, input signals $s_1, \ldots, s_m$ to this layer get transformed into outputs $s'_i = \max\left(\sum_{j=1}^{m} w_{ij} \cdot s_j, 0\right)$ which serve as inputs to the next layer. We do this until we reach the last layer, where we use *softmax*, i.e., where, based on c neural outputs z_j, we compute the probability p_i of being in a i-th class as $p_i = \dfrac{\exp(\beta \cdot z_i)}{\sum_{j=1}^{c} \exp(\beta \cdot z_j)}$ for some $\beta > 0$.

Training a neural network means selecting the weights w_{ij} for which, for the training set, the outputs are the closest to the desired ones. We start with some initial weights, and then iteratively update them until we get a good match. How fast the network learns depends on how well we selected the initial weights: if they are too far from the actual weights, training takes much longer.

How initial weights are selected now. Weights are selected layer-by-layer, starting with the input layer.

For each neuron i in the currently considered layer, we start with weights w_{ij} uniformly distributed on some interval $[-Z, Z]$. Then, we apply Gram-Schmidt orthonormalization to the vectors $w_i = (w_{i1}, \ldots, w_{in})$. Then, for each neuron i, we select a small sample of K patterns, get outputs $y_i^{(1)}, \ldots, y_i^{(K)}$, and re-scale the corresponding weights so that the standard deviation of these K values is $\sigma_0 = 1$.

Then we freeze these weights and go to the next layer.

To select the weights from the last (linear) layer, we match with the desired outputs.

Empirical observation that needs explaining. One of us (DA) tried to use $\sigma_0 < 1$ in the above algorithm. He got much better results than for $\sigma_0 = 1$ – and the smaller σ_0, the better results.

What we do. In this talk, we provide a theoretical explanation for this unexpected result.