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

1. 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 $s'_i = \max\left(\sum\limits_{j=1}^{m} w_{ij} \cdot s_j, 0\right)$.
- These outputs serve as inputs to the next layer.
- We do this until we reach the last layer, where we use *softmax*.
- Namely, based on c neural outputs z_j, we compute the probability p_i of being in a i-th class as

$$p_i = \frac{\exp(\beta \cdot z_i)}{\sum\limits_{j=1}^{c} \exp(\beta \cdot z_j)} \text{ for some } \beta > 0.$$

2. Selecting Initial Weights Is Important

- 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 the initial weights are too far from the actual ones, training takes much longer.

3. 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 $w_{ij} \sim U([-Z, Z])$.
- This means that w_{ij} are 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 $y_i^{(1)}, \ldots, y_i^{(K)}$.
- We then multiply all the the weights w_{ij} by some constant C: $w_{ij} \to C \cdot w_{ij}$.
- The constant is selected so that so that the standard deviation of the K values $y^{(k)}$ become equal to $\sigma_0 = 1$.

4. How Initial Weights Are Selected Now

- 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.

5. 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 $\sigma_0 = 1$.
- Moreover, the smaller σ_0, the better results.
- In this talk, we provide a theoretical explanation for this unexpected result.

6. Analysis of the Problem

- If we use $\sigma_0 < 1$, then on the first layer, instead of the original weights w_{ij}, we get new weights $w'_{ij} = \sigma_0 \cdot w_{ij}$.
- Then, with the same weights on other layers, we get standard deviation σ_0 on each of them.
- After L layers, we get new signals $z'_i = \sigma_0 \cdot z_i$.

7. Our Explanation

- Until we get to the last layer, we do not use the actual output.
- So we do now know the actual probabilities $q_1, \ldots, q_c$ of different classes.
- It is therefore reasonable to select the initial weights so that:
  - the resulting probabilities p_i
  - are, on average, as close to the actual (unknown) probabilities q_i as possible.

8. Our Explanation (cont-d)

- The closeness can be described:
  - either by the Euclidean distance

$$\|p-q\|^2=\sum_i (p_i-q_i)^2 \to \min,$$

  - or by any other strictly convex function $C(p,q)$ of p, e.g., by relative entropy.
- So, we minimize the expected value $\int C(p,q)\cdot\rho(q)\,dq$.
- At this stage, we do not have any information about the probabilities of different classes.
- So, it is reasonable to assume that this criterion does not change if we simply re-order the classes.
- Since the function $C(p,q)$ is convex, there is only one p for which this minimum is attained.

9. Our Explanation (cont-d)

- Thus, the optimal tuple p should also be invariant under such re-ordering, i.e., $p_i = p_j$ for all i and j.
- Hence, in the optimal case, we get $p_i = 1/c$ for all i.
- The use of $\sigma_0 < 1$ places all z_i closer to 0.
- Thus, the corresponding softmax values p_i are closer to the optimal values $1/c$.
- This explains why the results of using $\sigma_0 < 1$ are better.
- There is a minor difference between z_i and 0 – and thus, between p_i and the optimal values $1/c$.
- The smaller σ_0, the smaller this difference.
- This explains why the smaller σ_0, the better the results.