Abstract

In this lecture, we will see the details of the back-propagation algorithm. We will also look at questions about the power of depth in neural networks.

1 SGD on NN

Recall that \( \sigma(t) = (1 + \exp(-t))^{-1} \) and then \( \sigma'(t) = \sigma(t)(1 - \sigma(t)) \). Note that the network structure is chosen “beforehand”. Therefore, the challenge is to figure out the edge weight. In principle, we can compute \( \frac{\partial f}{\partial w} \) for every weight \( w \). We use SGD to compute the weight. The algorithm works as the following.

Given the training data \((x_1, y_1), \ldots, (x_n, y_n)\) and objective function \( \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 \).

For iteration \( t = 1, \ldots, T \),
  - Pick a random example \( x_i \).
  - Change weight using derivative of loss \( \nabla f(x_i) \).

2 Example of 3-layer NN

Denote the following notation.

- \( f \) is the output node
- \( z_i \) is node in first layer
- \( y_i \) is node in second layer
- \( x_i \) is node in third layer
- \( u_i \) is edge between \( f \) and \( z_i \)
- \( V_{ij} \) is edge between \( z_i \) and \( y_j \)
- \( W_{ij} \) is edge between \( y_i \) and \( x_j \)

In order to update in SGD, our goal is to find \( \frac{\partial f}{\partial u_i}, \frac{\partial f}{\partial V_{ij}} \) and \( \frac{\partial f}{\partial W_{ij}} \).
For any two nodes in the network structure $n_1, n_2$, denote $n_1 \leftrightarrow n_2$ that there is an edge between $n_1$ and $n_2$. In the first layer, we have $f = \sigma(\sum_{f \leftrightarrow z_i} u_i z_i)$. For any $u_i$,
\[
\frac{\partial f}{\partial u_i} = \sigma'(\sum_{f \leftrightarrow z_i} u_i z_i) \cdot z_i = f(1-f) z_i
\]

In the second layer, for any $z_i$, we have $z_i = \sigma(\sum_{z_i \leftrightarrow y_k} V_{ik} y_k)$ and then, for any $V_{ij}$, $\frac{\partial z_i}{\partial V_{ij}} = z_i(1 - z_i) y_j$. Therefore,
\[
\frac{\partial f}{\partial V_{ij}} = \frac{\partial f}{\partial z_i} \cdot \frac{\partial z_i}{\partial V_{ij}} = [f(1-f) u_i] \cdot [z_i(1 - z_i) y_j]
\]

The key difference between the first layer and the second layer is we need to deal with both node gradient and edge gradient in the second layer while we only need to compute edge gradient in first layer.

node gradient: $\frac{\partial f}{\partial z_i}, \frac{\partial f}{\partial y_i}, \frac{\partial f}{\partial x_i}$

edge gradient: $\frac{\partial f}{\partial u_i}, \frac{\partial f}{\partial V_{ij}}, \frac{\partial f}{\partial W_{ij}}$

The edge gradient affects the node gradient of previous layer and the node gradient affects the edge gradient in the same layer.

In the third layer, by the same procedure, for any $W_{ij}$,
\[
\frac{\partial f}{\partial W_{ij}} = \frac{\partial f}{\partial y_i} \cdot \frac{\partial y_i}{\partial W_{ij}} = \frac{\partial f}{\partial y_i} \cdot y_i(1-y_i)x_j
\]

To compute $\frac{\partial f}{\partial y_i}$,
\[
\frac{\partial f}{\partial y_i} = \sum_{z_k \leftrightarrow y_i} \frac{\partial f}{\partial z_k} \cdot \frac{\partial z_k}{\partial y_i}
\]
\[
= \sum_{z_k \leftrightarrow y_i} [f(1-f) u_k] \cdot [z_k(1-z_k) V_{ki}]
\]

There are some properties about SGD-backprop:

- Initialization matters (random is usually OK)
- No guarantees (local minimum)
- Matrix-vector product (highly parallelizable)
- "Momentum" term

3. VC dimension and NN

We have already known that the class of $m$-edge, $n$-node network has VC dimension $(m+n) \log(m+n)$. However, consider the following two networks. The first one has five layers and each layer has $n$ nodes. The second one has $\sqrt{n}$ layers and each layer has $n^{3/4}$ nodes. Both of them are complete network. That is, every pair of nodes in the consecutive layer have an edge. Both have
VC dimension $n^2 \log n$. However, the class of function they could compute is quite different since, intuitively, the more layer the network has the more complicated function it can compute.

There are some general properties of NN.

- depth can capture “oscillation” or “spikiness”
- function computed by low depth network cannot oscillate too much (unless it is very wide)
- depth $k$ network can have $\exp(k)$ oscillations

For one variable function, the number of oscillation is at most $O(m^k)$ where $k$ is the number of layer and $m$ is the number of node in each layer. Also, there is a $k'$ layers network that has at least $2^{k'/3}$ oscillations.