Unit 3 - lecture 9: Feedforward Neural Networks (Learning)

Objectives:

- Write down Recursive relations with back propagation algorithm to compute the gradient of the loss function with respect to the weight parameters.
- Use the Stochastic descent Algorithm to train a feedforward neural Network.
- Understand that it is not guaranteed to reach global (only local) optimum with SGD to minimize the training loss.
- Recognize when a network has overcapacity.

Back-propagation Algorithm: (computing the loss)

$$X \xrightarrow[w]{\text{complicated maping}} f = f(x; w)$$

we have y from the training set.

(x, y) we have to nudge them in the reverse direction of the gradient.

$$w_{ij}^{l} \leftarrow w_{ij}^{l} - \eta \left[\frac{\partial \, Loss(y, f(x; w))}{\partial w_{ij}^{l}} \right]$$

$$x \; O \xrightarrow{w_1} \overset{z_1}{O} \xrightarrow{f_1 \;\; w_2} \overset{z_2}{O} \xrightarrow{f_2} O \qquad \xrightarrow{w_L} \overset{z_L}{O} \xrightarrow[\in \mathbb{R}]{f_L} y$$

$\in \mathbb{R}$

$z_1 = x w_1$

$f_1 = \tanh(x w_1)$

$\vdots$

$f_L = \tanh(f_{L-1} w_L)$

$$Loss(y, f_L) = \frac{1}{2}(y - f_L)^2$$

how much our prediction differ from our target value

$$f_{l-1} \; \boxed{w_l} \; f_l \qquad \frac{\partial f_l}{\partial f_{l-1}}$$

$$\frac{\partial \, loss}{\partial w_1} ? = \underbrace{\frac{\partial f_1}{\partial w_1}}_{(1-f_1^2)x} \; \underbrace{\frac{\partial \, loss}{\partial f_1}}_{?}$$

$$\frac{\partial \, loss}{\partial f_1} = \underbrace{\frac{\partial f_2}{\partial f_1}}_{(1-f_2^2) w_2} \; \underbrace{\frac{\partial \, loss}{\partial f_2}}$$

backward

Once we set up the architecture of our (feed-forward) neural Network, our goal will be to find Weight parameters that minimize our loss function. We will use the Stochastic gradient descent Algorithm.

$$\theta \leftarrow \theta - \eta \nabla_\theta Loss$$

$L(y, f_L)$ denote the loss function as a function of the prediction f_L and the true label y.

$$z_1 = x w_1$$

$$\text{for } i = 2 \ldots L. \quad z_i = f_{i-1} w_i \quad \text{where } f_{i-1} = f(z_{i-1})$$

$$\text{let } \delta_i = \frac{\partial L}{\partial z_i}$$

The 1^{st} step to updating any weight w is to calculate $\frac{\partial L}{\partial w}$

$$x \; \bigcirc \xrightarrow{w_1} \overset{z_1}{\bigcirc} \, f_1 \xrightarrow{w_2} \overset{z_2}{\bigcirc} \, f_2 \longrightarrow \bigcirc$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial z_1}{\partial w_1} \frac{\partial L}{\partial z_1} \qquad \text{since } z_1 = w_1 x :$$

$$\frac{\partial z_1}{\partial w_1} = \frac{\partial (w_1 x)}{\partial w_1} = x \implies \frac{\partial L}{\partial w_1} = x \cdot \frac{\partial L}{\partial z_1} = x \cdot \delta_1$$

<u>Recursive Expression:</u>

In this problem, we derive a recurrence relation between δ_i and δ_{i+1}

$$f(x) = \tanh(x)$$
$$f'(x) = (1 - \tanh^2(x))$$

Expression of δ_1 in terms of δ_2?

- The chain Rule gives: $\delta_1 = \frac{\partial L}{\partial z_1} = \frac{\partial f_1}{\partial z_1} \cdot \frac{\partial z_2}{\partial f_1} \cdot \frac{\partial L}{\partial z_2}$
- Since $f_1 = \tanh(z_1) \implies \frac{\partial f_1}{\partial z_1} = (1 - f_1^2)$
- Since $z_2 = w_2 \cdot f_1 \implies \frac{\partial z_2}{\partial f_1} = w_2$
- Substituting the values of $\frac{\partial f_1}{\partial z_1}$, $\frac{\partial z_2}{\partial f_1}$ into the main expression for δ_1 we get:

$$\delta_1 = (1 - f_1^2) \cdot w_2 \cdot \frac{\partial L}{\partial z_2} = (1 - f_1^2) \cdot w_2 \cdot \delta_2$$

Final Expression of the Gradient:

Let $L(y, f_L)$ denote the loss function as a function of the predictions f_L and the true label y.

$$\delta_i = \frac{\partial L}{\partial z_i}$$

$$L(y, f_L) = (y - f_L)^2$$

Compute $\frac{\partial L}{\partial w_1}$:

From the previous problem : $\frac{\partial L}{\partial w_1} = x\, \delta_1$

$$\delta_1 = (1 - f_1^2) \cdot w_2 \cdot \delta_2$$

Similarly, $\delta_2\ \delta_3 \ldots \delta_L$ can be given as follows:

$$\delta_2 = (1 - f_2^2)\, w_3 \cdot \delta_3$$

$$\delta_3 = (1 - f_3^2)\, w_4 \cdot \delta_4$$

$$\vdots$$

$$\delta_{L-1} = (1 - f_{L-1}^2) \cdot w_L \cdot \delta_L$$

$$\delta_L = \frac{\partial L}{\partial z_L}$$

$$\delta_L = \frac{\partial L}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \rightarrow \delta_L = \frac{\partial (f_L - y)^2}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} = 2(f_L - y)\frac{\partial f_L}{\partial z_L}$$

$$\delta_L = 2(f_L - y)(1 - f_L^2)$$

Plugging the above equations into the expression for $\frac{\partial L}{\partial w_1}$ we get:

$$\frac{\partial L}{\partial w_1} = x \cdot \delta_1$$

$$\frac{\partial L}{\partial w_1} = x \cdot (1 - f_1^2) \cdot w_2 \cdot \delta_2$$

$$\frac{\partial L}{\partial w_1} = x \cdot (1 - f_1^2) \cdot w_2 \cdot (1 - f_2^2) \cdot w_3 \cdot \delta_3$$

$$\vdots$$

$$\frac{\partial L}{\partial w_1} = x(1 - f_1^2)(1 - f_2^2)\cdots(1 - f_L^2)\, w_2 w_3 \cdots w_L (2(f_L - y))$$

- For multi-layer neural networks the loss function is no longer convex and any stochastic gradient descent (SGD) method is not guaranteed to reach global optimum.
- Larger models tend to be easier to learn because their units need to be adjusted so that they are collectively sufficient to solve the task.