Back propagation algoirthm

Zhe Feng

December 21, 2018

1 Problem Description

Back propagation lays in the central part of the neuralnet. Essentially, consider general definition of a neural network:

$$\hat{y}_k := z_k^L \text{ for } k \in \{1, \cdots, n_y\} \tag{1}$$
$$z_k^\ell := \sigma^\ell(m_k^\ell) \text{ for } \ell \in \{2, \cdots, L\} \tag{2}$$
$$m_k^\ell := \sum_i^{n_{\ell-1}} w_{ik}^\ell z_i^{\ell-1} \text{ for } k \in \{1, \cdots, n_\ell\} \tag{3}$$
$$z_k^1 := x_k \text{ for } k \in \{1, \cdots, n_x\} \tag{4}$$

where

1. $z^\ell \in \mathbb{R}^{n_\ell}$ is the net output of the ℓ^{th} layer (so the L is the output layer of the net.
2. $m^\ell \in \mathbb{R}^{n_\ell}$ is the net input of the ℓ^{th} layer.
3. $w^\ell := [w_{11}^\ell, w_{12}^\ell, \cdots, w_{n_{\ell-1}1}^\ell, \cdots, w_{n_{\ell-1}n_\ell}^\ell]^\top \in \mathbb{R}^{n_{\ell-1}n_\ell}$ is the weight parameters at layer ℓ, where w_{ij}^ℓ is the weight at layer ℓ connect node i at layer $\ell - 1$ to node j in layer ℓ.
4. $W^\ell := \begin{bmatrix} w_{11}^\ell & w_{12}^\ell & \cdots & w_{1n_\ell}^\ell \\ w_{21}^\ell & \ddots & & \\ \vdots, & & & \\ w_{n_{\ell-1}1}^\ell & & & w_{n_{\ell-1}n_\ell}^\ell \end{bmatrix} \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ is the weight matrix at layer ℓ, this is for the convenience of notation later.
5. $\sigma^\ell(x)$ is the activation function at the ℓ^{th} layer (in most cases are sigmoid function, hence the symbol σ).

So the ultimate goal is to tune the weights and make sure that the error is minimised. So our cost function would be:

$$\min_w Q(y; w) := \frac{1}{N} \sum_{i}^{N} \|\hat{y}^{(i)} - y^{(i)}\| \tag{5}$$

$$\text{where } w := \begin{bmatrix} w^1 \\ \vdots \\ w^L \end{bmatrix} \in \mathbb{R}^{\Pi_\ell^L n_\ell} \tag{6}$$

2 Back propagation algorithm

In general, for most optimisation algoirthm, the bottomline comes to find a stationary point where the derivative is zero, that is:

$$w^* \text{ such that } \nabla_w Q = 0 \tag{7}$$

2.1 Output layer delta

Consider the weights W^L in the output layer, the partial derivative to those weights are:

$$\boxed{\frac{\partial Q}{\partial w_{ij}^L} = \frac{\partial Q}{\partial z_j^L} \frac{\partial z_j^L}{\partial m_j^L} \frac{\partial m_j^L}{\partial w_{ij}^L}} \tag{8}$$

This is done by using chain rule, and we know that w_{ij}^L will only contribute to the j^{th} output $y_j = z_j^L$.

2.1.1 Find the term $\frac{\partial m_j^L}{\partial w_{ij}^L}$

Now for $\frac{\partial z_j^L}{\partial m_j^L}$, we can see:

$$\boxed{\frac{\partial z_j^L}{\partial m_j^L} = \frac{\partial}{\partial m_j^L} \sigma^L(m_j^L) = \sigma^{L\prime}(m_j^L)} \tag{9}$$

where

$$\sigma^{L\prime}(m_j^L) := \frac{d}{dm_j^L} \sigma^L(m_j^L) \tag{10}$$

One special case, is when the activation function is the logistic sigmoid function:

$$\sigma^L(m) := \frac{1}{1+e^{-m}} \tag{11}$$

$$\therefore \sigma^{L'}(m) = \sigma^L(m)(1-\sigma^L(m)) \tag{12}$$

detailed derivation can be done easily using chain rule.

2.1.2 Find the term $\frac{\partial z_j^L}{\partial m_j^L}$

Recall the definition of m^L, (3), then the next term is easy:

$$\boxed{\frac{\partial m_j^L}{\partial w_{ij}^L} = z_i^{L-1}} \tag{13}$$

Put things together and substitute (9) and (13) back to (8), we have

$$\frac{\partial Q}{\partial w_{ij}^L} = \frac{\partial Q}{\partial z_j^L}\sigma^{L'}(m_j^L)z_i^{L-1} \tag{14}$$

wich in a more compact vector form, we can write it as

$$\nabla_{w^L} Q = (\nabla_{z^L} Q \odot \sigma^{L'}(m^L)) \otimes z^{L-1} \tag{15}$$

where

$$\nabla_{w^L} Q = \begin{bmatrix} \frac{\partial Q}{\partial w_{11}^L} \\ \frac{\partial Q}{\partial w_{12}^L} \\ \vdots \\ \frac{\partial Q}{\partial w_{n_{\ell-1}n_\ell}^L} \end{bmatrix} \tag{16}$$

For implementation and notation convenience, we can also re-write the above equation as

$$\boxed{\Delta W^L = (\nabla_{z^L} Q \odot \sigma^{L'}(m^L))^\top z^{L-1}} \tag{17}$$

where

$$\Delta W^L = \begin{bmatrix} \frac{\partial Q}{\partial w_{11}^L} & \frac{\partial Q}{\partial w_{12}^L} & \cdots \\ \frac{\partial Q}{\partial w_{21}^L} & \ddots & \\ \vdots, & & \end{bmatrix} \in \mathbb{R}^{n_{\ell-1}\times n_\ell} \tag{18}$$

which can then be easily plug in to the update equation $\boxed{W^L -= \eta\Delta W^L}$ and η is the learning rate.

2.1.3 δ notation

To make sense of the backpropagation, we use δ notation to denote the error delta for a particular node. Define:

$$\delta^L := \nabla_{z^L} Q \odot {\sigma^L}'(m^L) \tag{19}$$

where

$$\delta^L = \begin{bmatrix} \delta_1^L \\ \vdots \\ \delta_{n_L}^L \end{bmatrix} \tag{20}$$

$$\text{and } \delta_j^L = \frac{\partial Q}{\partial z_j^L} {\sigma^L}'(m_j^L) \tag{21}$$

Then equation (17) can be rewritten as:

$$\Delta W^L = {\delta^L}^\top z^{L-1} \tag{22}$$

2.2 Hidden layers deltas

Now for hiddenlayers, similar principle applies:

$$\boxed{\frac{\partial Q}{\partial w_{ij}^\ell} = \sum_k^{n_{\ell+1}} \frac{\partial Q}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial m_k^{\ell+1}} \frac{\partial m_k^{\ell+1}}{\partial z_j^\ell} \frac{\partial z_j^\ell}{\partial m_j^\ell} \frac{\partial m_j^\ell}{\partial w_{ij}^\ell} \text{ for } \ell \in \{2, \cdots, L-1\}} \tag{23}$$

The key difference here is for hidden layers, the partial derivative of weight w_{ij}^ℓ will affect all the nodes in $\ell + 1$ layer (unlike in output layer L, where w_{ij}^L will only affect j^{th} node, which is z_j^L), hence the sum over all the partial derivatives of $\ell + 1$ layer nodes: $\frac{\partial Q}{\partial w_{ij}^\ell} = \sum_k^{n_{\ell+1}} \frac{\partial Q}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial w_{ij}^\ell}$.

2.2.1 Recursively define δ^ℓ through back propagation

It is easy to see that:

$$\frac{\partial Q}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial m_k^{\ell+1}} = \frac{\partial Q}{\partial z_k^{\ell+1}} {\sigma^{\ell+1}}'(m_k^{\ell+1}) \tag{24}$$

and when $\ell = L - 1$, from equation (21) we have δ_j^L. It is also easy to see that:

$$\frac{\partial m_k^{\ell+1}}{\partial z_j^\ell} = w_{jk}^{\ell+1} \tag{25}$$

from the definition in equation (3). Through similar logic in deducing output layer delta, with equations (9) and (13), we can write equation (23) as:

$$\frac{\partial Q}{\partial w_{ij}^{\ell}} = \sum_{k}^{n_{\ell+1}} \delta_k^{\ell+1} w_{jk}^{\ell+1} \sigma^{\ell\prime}(m_j^{\ell}) z_i^{\ell-1} \text{ for } \ell \in \{2, \cdots, L-1\} \tag{26}$$

and

$$\boxed{\delta_j^{\ell} := \sum_{k}^{n_{\ell+1}} \delta_k^{\ell+1} w_{jk}^{\ell+1} \sigma^{\ell\prime}(m_j^{\ell})} \tag{27}$$

$$\boxed{\therefore \text{ in vector form } \delta^{\ell} := W^{\ell+1} \delta^{\ell+1} \odot \sigma^{\ell\prime}(m^{\ell})} \tag{28}$$

Note that when $\ell = L - 1$, the $\delta^{\ell+1} = \delta^{L}$ which is the output layer nodes' deltas defiend in equation (21).

So now for hidden layer, we can have the same form of definition as for output layer:

$$\boxed{\Delta W^{\ell} = {\delta^{\ell}}^{\top} z^{\ell-1}} \tag{29}$$

2.3 The back propagation algorithm

Now we have all the ingradient to summarise our back propagation algorithm:

1. run feedforward through the neuralnet work, calculate all m^{ℓ}, z^{ℓ} and consequently $\hat{y}$.
2. back propagate the error, calculate ΔW^{ℓ} for all $\ell \in \{2, \cdots, L\}$.
3. update all W^{ℓ} with

$$\boxed{W^{\ell} = W^{\ell} - \eta \Delta W^{\ell} \text{ for } \ell \in \{2, \cdots, L\}} \tag{30}$$

   where

$$\Delta W^{\ell} = {\delta^{\ell}}^{\top} z^{\ell-1} \tag{31}$$

$$\delta^{\ell} = W^{\ell+1} \delta^{\ell+1} \odot \sigma^{\ell\prime}(m^{\ell}) \text{ for } \ell \in \{2, \cdots, L-1\} \tag{32}$$

$$\delta^{L} = \nabla_{z^L} Q \odot \sigma^{L\prime}(m^{L}) \tag{33}$$

4. Repeat till converge ($\nabla_w Q \leq \epsilon$).

Note that we still haven't discussed bias at each layer b^{ℓ} yet. It is actually very simple, it can be shown that

$$\boxed{\Delta b^{\ell} = \delta^{\ell} \text{ for } \ell \in \{2, \cdots, L\}} \tag{34}$$

therefore bias b^ℓ can be updated by

$$\boxed{b^\ell = b^\ell - \eta \Delta b^\ell \text{ for } \ell \in \{2, \cdots, L\}} \tag{35}$$