Back propagation algoirthm

Zhe Feng

May 17, 2020

1 Introduction

Deep learning is the hottest topic in AI these days. Convolution neural network (CNN) is widely used in image recognition and gained huge success, recurrent neural networks (RNN) used in sequential data prediction and achieved unprecedented performance in automatic speech recognition (ASR) and automatic translation. Behind all these success, is an algorithm that is developed 1960s, independently from researchers in different disciplines, and its name is back propagation. Back propagation lays in the heart of artificial neural networks, whether it's a simple network with one hidden layer or a complex CNNs, they all use back propagation to calculate the gradient (with different variations). Of course, these days to use complicated neural networks, you don't need to know detail about how back propagation work. However, as a personal interest, I decided to have a proper understanding of how it works and appreciate its beautiful mathematical structure.

2 Problem Description

Essentially, given a feature vector x of dimension n_x and a target vector y with dimension y_{n_y} neural network will try to predict and approximate it with $\hat{y}$ which is the output from our neural network. A general neural network can be described by the following equations:

$$\hat{y}_k := z_k^L \text{ for } k \in \{1, \cdots, n_y\} \tag{1}$$

$$z_k^\ell := \sigma^\ell(m_k^\ell) \text{ for } \ell \in \{2, \cdots, L\} \tag{2}$$

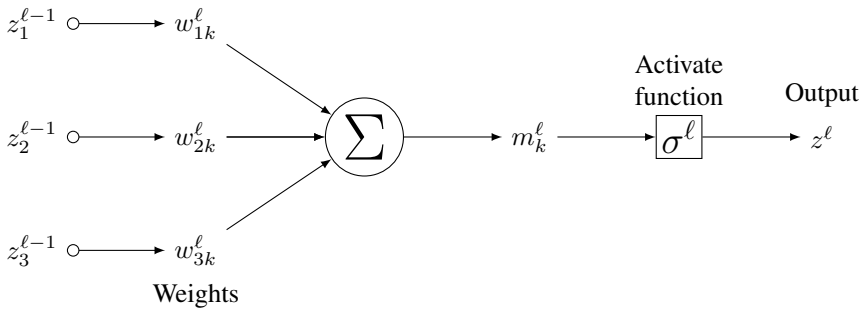
$$m_k^\ell := \sum_i^{n_{\ell-1}} w_{ik}^\ell z_i^{\ell-1} \text{ for } k \in \{2, \cdots, n_\ell\} \tag{3}$$

$$z_k^1 := x_k \text{ for } k \in \{1, \cdots, n_x\} \tag{4}$$

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. $w^\ell := [w^\ell_{11}, w^\ell_{12}, \cdots, w^\ell_{n_{\ell-1}1}, \cdots, w^\ell_{n_{\ell-1}n_\ell}]^\top \in \mathbb{R}^{n_{\ell-1}n_\ell}$ is the weight parameters at layer ℓ, where w^ℓ_{ij} is the weight at layer ℓ connect node i at layer $\ell-1$ to node j in layer ℓ. Note that we don't have w^1 since the 1st layer is the input layer.
4. $W^\ell := \begin{bmatrix} w^\ell_{11} & w^\ell_{12} & \cdots & w^\ell_{1n_{\ell-1}} \\ w^\ell_{21} & \ddots & & \\ \vdots, & & & \\ w^\ell_{n_\ell 1} & & & w^\ell_{n_\ell n_{\ell-1}} \end{bmatrix} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ 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).

The 4 equation description is very simple and elegant: equation (1) essentially just say the last layer's output z^L is our target estimation $\hat{y}$ and equation (4) state that the first layer's output z^1 is the input feature variable x. With the start and the end defined, equation (3) and (2) are recursive definition of the layers. This is illustrated in the following example plot:



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

$$\min_w Q(y; w) := \frac{1}{N} \sum_i^N \frac{1}{2} \|\hat{y}^{(i)} - y^{(i)}\|^2$$

$$\text{s.t. eqs. (1) to (4)}$$

$$\text{where } w := \{w^2 \cdots w^L\}$$

3 Back propagation algorithm

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

$$w^* \text{ such that } \nabla_w Q = 0 \tag{5}$$

And that is what makes optimisation of neural network hard: neural networks' definition function is a recursive definition and finding gradient of it is very challenging. Back propagation essentially leverage chain rule in calculus to recursively propagate the error gradient backwards from the last output layer to the first input layer.

3.1 Output layer delta

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

$$\boxed{\frac{\partial Q}{\partial w_{ij}^L} = \frac{\partial Q}{\partial z_j^L}\frac{\partial z_j^L}{\partial m_j^L}\frac{\partial m_j^L}{\partial w_{ij}^L}} \tag{6}$$

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

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

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

$$\boxed{\frac{\partial z_j^L}{\partial m_j^L} = \frac{\partial}{\partial m_j^L}\sigma^L(m_j^L) = \sigma^{L\prime}(m_j^L)} \tag{7}$$

where

$$\sigma^{L\prime}(m_j^L) := \frac{d}{dm_j^L}\sigma^L(m_j^L) \tag{8}$$

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

$$\sigma^L(m) := \frac{1}{1+e^{-m}} \tag{9}$$

$$\therefore \sigma^{L\prime}(m) = \sigma^L(m)(1-\sigma^L(m)) \tag{10}$$

detailed derivation can be done easily using chain rule.

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

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

$$\boxed{\frac{\partial m_j^L}{\partial w_{ij}^L} = z_i^{L-1}} \tag{11}$$

or in a more general form:

$$\frac{\partial m_j^\ell}{\partial w_{ij}^\ell} = z_i^{\ell-1} \tag{12}$$

Put things together and substitute (7) and (12) back to (6), we have

$$\frac{\partial Q}{\partial w_{ij}^L} = \frac{\partial Q}{\partial z_j^L} \sigma^{L\prime}(m_j^L) z_i^{L-1} \tag{13}$$

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

$$\nabla_{w^L} Q = (\nabla_{z^L} Q \odot \sigma^{L\prime}(m^L)) \otimes z^{L-1} \tag{14}$$

where

$$\nabla_{w^L} Q = \begin{bmatrix} \frac{\partial Q}{\partial w_{11}^L} \\ \frac{\partial Q}{\partial w_{12}^L} \\ \vdots \\ \frac{\partial Q}{\partial w_{n_{\ell-1} n_\ell}^L} \end{bmatrix} \tag{15}$$

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

$$\boxed{\Delta W^L = z^{L-1} (\nabla_{z^L} Q \odot \sigma^{L\prime}(m^L))^\top} \tag{16}$$

where

$$\Delta W^L = \begin{bmatrix} z_1^{L-1} \\ z_2^{L-1} \\ \vdots \\ z_{n_{L-1}}^{L-1} \end{bmatrix} \left[\frac{\partial Q}{\partial z_1^L} \sigma^{L\prime}(m_1^L), \frac{\partial Q}{\partial z_2^L} \sigma^{L\prime}(m_2^L), \cdots \frac{\partial Q}{\partial z_{n_L}^L} \sigma^{L\prime}(m_{n_L}^L) \right] \tag{17}$$

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

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

3.1.3 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

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

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

or more generally:

$$\delta_j^\ell := \frac{\partial Q}{\partial z_j^\ell} \frac{\partial z_j^\ell}{\partial m_j} \tag{22}$$

Then equation (16) can be rewritten as:

$$\Delta W^L = (z^{L-1}) \delta^{L\top} \tag{23}$$

3.2 Hidden layers deltas

Now for hiddenlayers, similar principle applies:

$$\boxed{\frac{\partial Q}{\partial w_{ij}^\ell} = \sum_k^{n_{\ell+1}} \frac{\partial Q}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial m_k^{\ell+1}} \frac{\partial m_k^{\ell+1}}{\partial z_j^\ell} \frac{\partial z_j^\ell}{\partial m_j^\ell} \frac{\partial m_j^\ell}{\partial w_{ij}^\ell} \text{ for } \ell \in \{2, \cdots, L-1\}} \tag{24}$$

This holds because it follows the chain rule for partial differentiation in the following form:

$$\frac{\partial f(g_1(x), g_2(x), \cdots g_n(x))}{\partial x} = \sum_i^n \frac{\partial f}{\partial g_i} \frac{dg_i(x)}{dx}$$

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: $\frac{\partial Q}{\partial w_{ij}^\ell} = \sum_k^{n_{\ell+1}} \frac{\partial Q}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial w_{ij}^\ell}$.

3.2.1 Recursively define δ^ℓ through back propagation

It is easy to see that:

$$\frac{\partial Q}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial m_k^{\ell+1}} = \frac{\partial Q}{\partial z_k^{\ell+1}} \sigma^{\ell+1\prime}(m_k^{\ell+1}) = \delta_k^{\ell+1} \tag{25}$$

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

$$\frac{\partial m_k^{\ell+1}}{\partial z_j^\ell} = w_{jk}^{\ell+1} \tag{26}$$

from the definition in equation (3). Through similar logic in deducing output layer delta, with equations (7) and (12), we can write equation (24) as:

$$\frac{\partial Q}{\partial w_{ij}^\ell} = \sum_k^{n_{\ell+1}} \delta_k^{\ell+1} w_{jk}^{\ell+1} \sigma^{\ell\prime}(m_j^\ell) z_i^{\ell-1} \text{ for } \ell \in \{2, \cdots, L-1\} \tag{27}$$

since we can also write the above as:

$$\frac{\partial Q}{\partial w_{ij}^\ell} = \frac{\partial Q}{\partial z_j^\ell} \frac{\partial z_j^\ell}{\partial m_j^\ell} \frac{\partial m_j^\ell}{\partial w_{ij}^\ell} \text{ for } \ell \in \{2, \cdots, L-1\} \tag{28}$$

and we can see $\frac{\partial m_j^\ell}{\partial w_{ij}^\ell} = z_i^{\ell-1}$:

$$\boxed{\delta_j^\ell := \frac{\partial Q}{\partial z_j^\ell} \frac{\partial z_j^\ell}{\partial m_j^\ell} = \sum_k^{n_{\ell+1}} \delta_k^{\ell+1} w_{jk}^{\ell+1} \sigma^{\ell\prime}(m_j^\ell)} \tag{29}$$

$$\boxed{\therefore \text{ in vector form } \delta^\ell := W^{\ell+1} \delta^{\ell+1} \odot \sigma^{\ell\prime}(m^\ell)} \tag{30}$$

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:

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

3.3 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

$$\boxed{W^\ell = W^\ell - \eta \Delta W^\ell \text{ for } \ell \in \{2, \cdots, L\}} \tag{32}$$

   where

$$\boxed{\begin{aligned} \Delta W^\ell &= z^{\ell-1} {\delta^\ell}^\top && (33)\\ \delta^\ell &= W^{\ell+1}\delta^{\ell+1} \odot {\sigma^\ell}'(m^\ell) \text{ for } \ell \in \{2, \cdots, L-1\} && (34)\\ \delta^L &= \nabla_{z^L} Q \odot {\sigma^L}'(m^L) && (35) \end{aligned}}$$

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

$$\boxed{\Delta b^\ell = \delta^\ell \text{ for } \ell \in \{2, \cdots, L\}} \tag{36}$$

therefore bias b^ℓ can be updated by

$$\boxed{b^\ell = b^\ell - \eta \Delta b^\ell \text{ for } \ell \in \{2, \cdots, L\}} \tag{37}$$