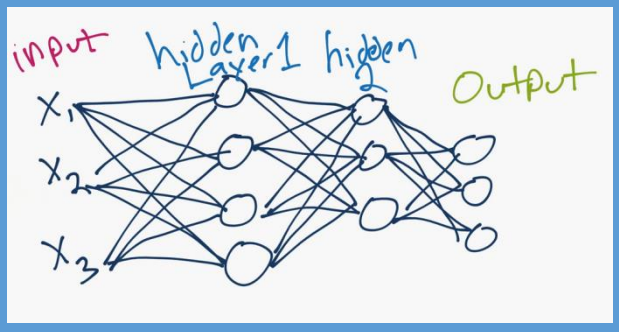
EMS702P Statistical Thinking and Applied Machine Learning

Week 9.1 – Neural Networks Training

Yunpeng Zhu



Neural Networks

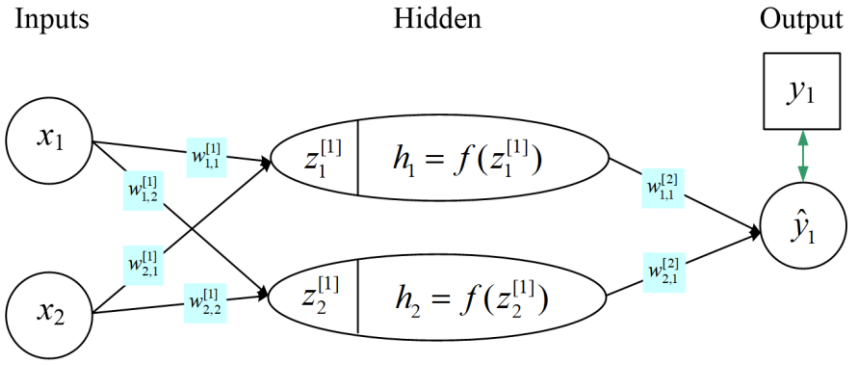
Edition: v1.1

Table of Contents

1 Back-Propagation (BP) algorithm for network training

1.1 A single training set

Considering a simple neural network with 2 input nodes, 2 hidden nodes, and 1 output node.



The neural network can be written as

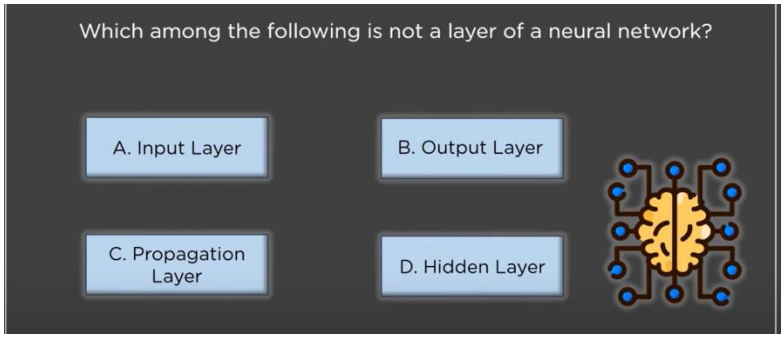
$$\begin{cases} \begin{cases} z_1^{[1]} = w_{1,1}^{[1]} x_1 + w_{2,1}^{[1]} x_2 \\ z_2^{[1]} = w_{1,2}^{[1]} x_1 + w_{2,2}^{[1]} x_2 \end{cases} \\ \begin{cases} h_1 = f(z_1^{[1]}) \\ h_2 = f(z_2^{[1]}) \end{cases} \\ \hat{y}_1 = w_{1,1}^{[2]} h_1 + w_{2,1}^{[2]} h_2 \end{cases}$$

where $f(z) = \dfrac{1}{1+\exp(-z)}$.

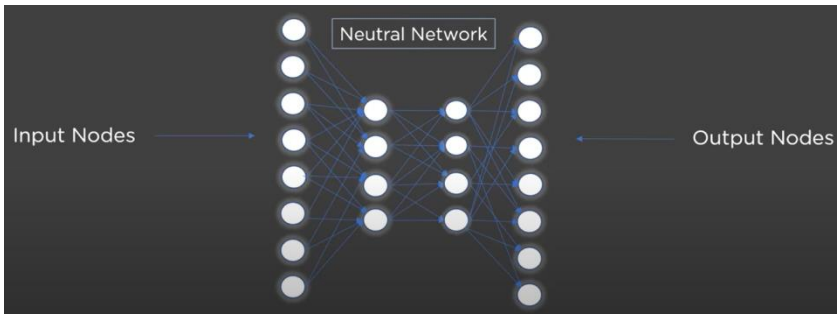
For a given input $[x_1, x_2]^{\mathrm{T}}$, the true output is y_1. The mean square error of the predicted output is defined as the cost function

$$J = \frac{1}{2}(\hat{y}_1 - y_1)^2$$

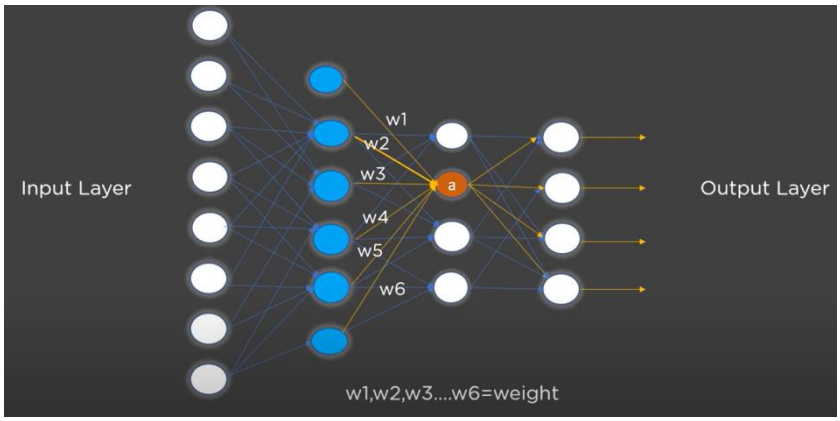
The aim of the BP algorithm is to find the weights that minimize the cost function. We use the terminology **Back-Propagation** as we shall see the prediction error propagated backward in the neural network while determine the derivatives.



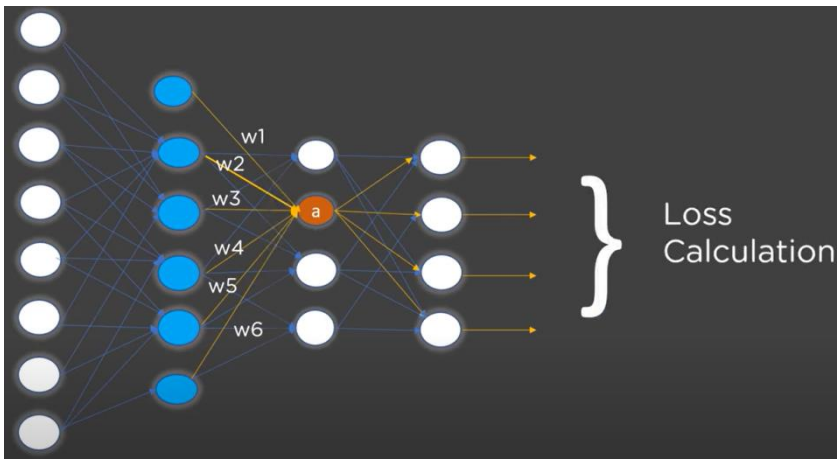
Back-Propagation is an algorithm which is created to test errors which will travel back from output nodes to input nodes



Forward Propagation:

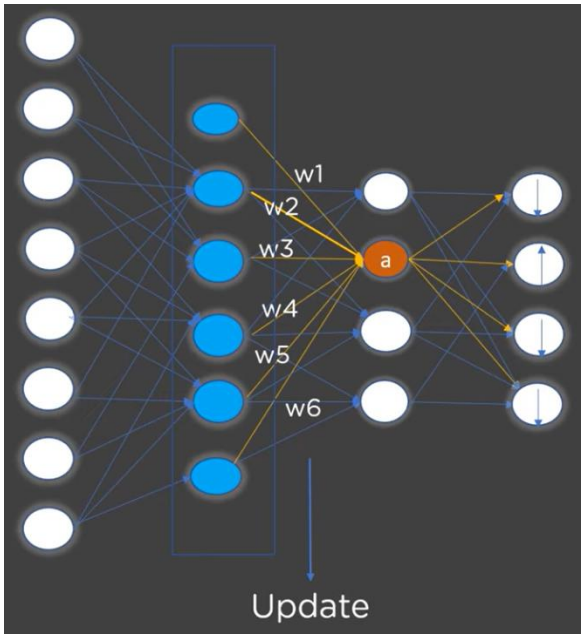


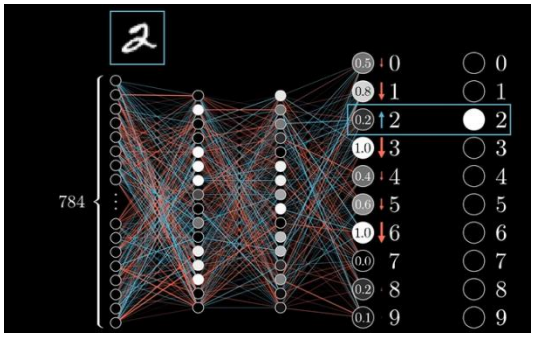
Loss (Cost) function:

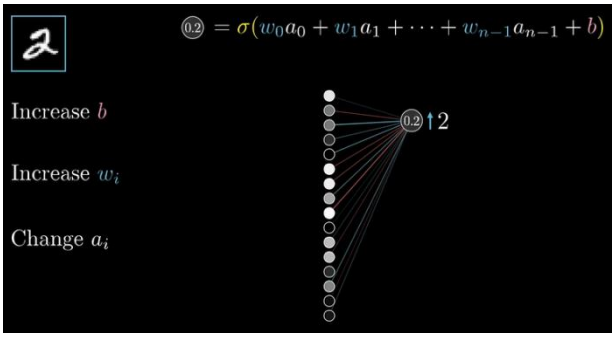


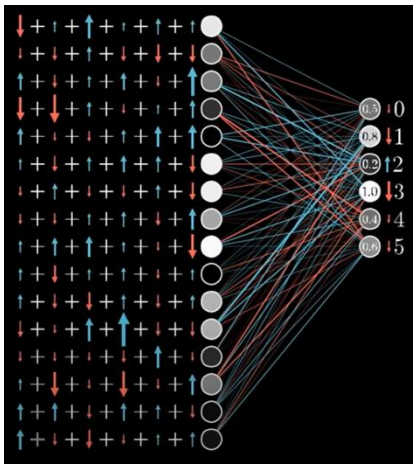
Backward Propagation:

According to gradient descent method, we can find the minimum value of the cost function by tuning the **weights**.

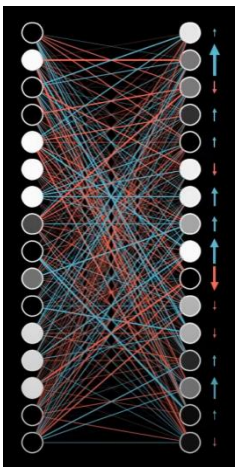




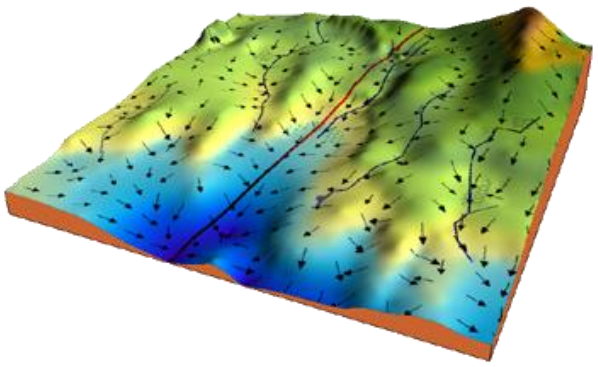




Back propagation



As we got the cost function and the aim is to evaluate weights, the gradient descent method is applied to solve this problem. The gradients can be calculated by the derivatives of the cost function to weights.



The **chain rule** is applied to calculate the derivatives of the cost function

Introduction of the chain rule:

We know how to evaluate derivatives of simple functions such as x^n, $\sin(x)$, e^x, etc. almost all complex functions are combined by these simple functions.

$$\left(e^{\sin(x)} \cdot \cos\left(\frac{1}{x^3} + x^3\right)\right)^4$$

composition $\left(e^{\sin(x)} \cdot \cos\left(\frac{1}{x^3} + x^3\right) \right)^4$

product $e^{\sin(x)} \cdot \cos\left(\frac{1}{x^3} + x^3\right)$

composition $\cos\left(\frac{1}{x^3} + x^3\right)$

sum $\frac{1}{x^3} + x^3$

composition $\frac{1}{x^3}$

Sum rule is simple: The derivative of Sum is the sum of derivatives.

$$\frac{d}{dx}(\sin(x) + x^2) = \cos(x) + 2x$$

$$\frac{d}{dx}(g(x) + h(x)) = \frac{dg}{dx} + \frac{dh}{dx}$$

Product rule: Left d Right plus Right d Left

$$\frac{d}{dx}(\sin(x)x^2)$$

$$\sin(x)2x + x^2\cos(x)$$

$$\text{Left d(Right)} + \text{Right d(Left)}$$

Chain rule:

Considering a function $\sin(x^2)$. We denote a new variable $h = x^2$, then

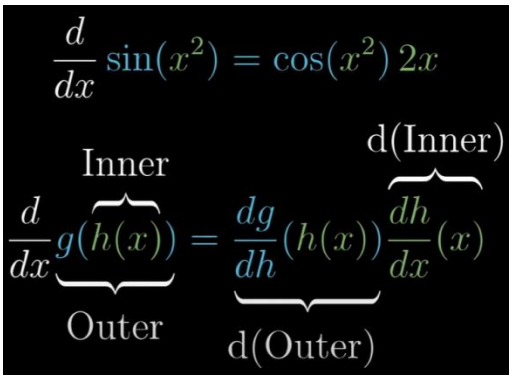
$$\frac{\mathrm{d}\sin(h)}{\mathrm{d}h} = \cos(h) \Rightarrow \mathrm{d}\sin(h) = \cos(h)\mathrm{d}h$$

and

$$\frac{\mathrm{d}h}{\mathrm{d}x} = \frac{\mathrm{d}x^2}{\mathrm{d}x} = 2x \Rightarrow \mathrm{d}h = 2x\mathrm{d}x$$

Thus $\mathrm{d}\sin(h) = \cos(h)\mathrm{d}h = \cos(x^2)2x\mathrm{d}x$, and

$$\frac{\mathrm{d}\sin(h)}{\mathrm{d}x} = \frac{\mathrm{d}\sin(x^2)}{\mathrm{d}x} = \cos(x^2)2x$$



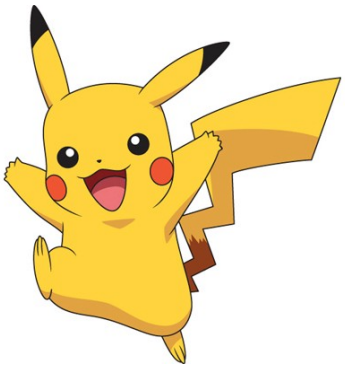
You can write this down if there are multiple composited equations:

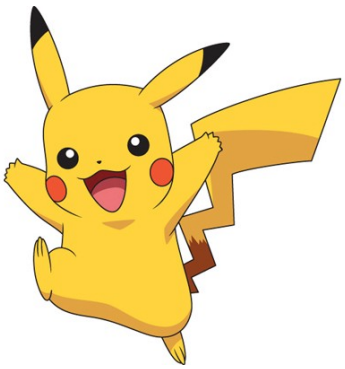
For $f(x) = f(g_1(g_2(\ldots(g_n(x)))))$

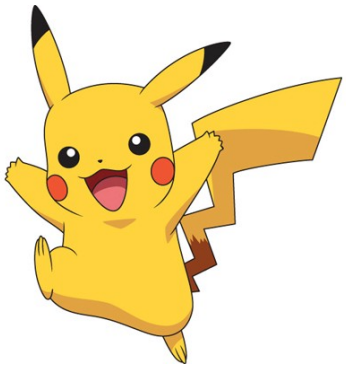
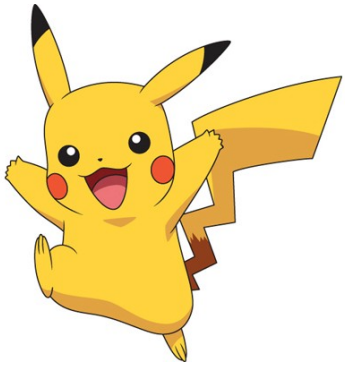
$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \frac{\mathrm{d}f(x)}{\mathrm{d}g_1(x)}\frac{\mathrm{d}g_1(x)}{\mathrm{d}g_2(x)}\cdots\frac{\mathrm{d}g_{n-1}(x)}{\mathrm{d}g_n(x)}\frac{\mathrm{d}g_n(x)}{\mathrm{d}x}$$

Quiz 1.1:

Calculate the derivatives of $f_1(x) = (2x^2 - 1)^3$ and $f_2(x) = \left[2(x+2)^2 - 1\right]^2$





In the neural network we mentioned at the beginning, the **chain rule** of derivative is applied to calculate the derivatives (Gradients) of the cost function. [Note we use partial derivatives because the cost function contains multiple variables.]

$$\begin{cases} \dfrac{\partial J}{\partial w_{1,1}^{[2]}} = \dfrac{\partial J}{\partial \hat{y}_1}\dfrac{\partial \hat{y}_1}{\partial w_{1,1}^{[2]}} = \left(\hat{y}_1 - y_1\right)h_1 \\ \dfrac{\partial J}{\partial w_{2,1}^{[2]}} = \dfrac{\partial J}{\partial \hat{y}_1}\dfrac{\partial \hat{y}_1}{\partial w_{2,1}^{[2]}} = \left(\hat{y}_1 - y_1\right)h_2 \end{cases}$$

where

$$J = \frac{1}{2}\left(\hat{y}_1 - y_1\right)^2 \Rightarrow \frac{\partial J}{\partial \hat{y}_1} = \hat{y}_1 - y_1$$

$$\hat{y}_1 = w_{1,1}^{[2]}h_1 + w_{2,1}^{[2]}h_2 \Rightarrow \begin{cases} \dfrac{\partial \hat{y}_1}{\partial w_{1,1}^{[2]}} = h_1 \\ \dfrac{\partial \hat{y}_1}{\partial w_{2,1}^{[2]}} = h_2 \end{cases}$$

In the calculation of $\partial J \big/ \partial w_{n,m}^{[1]}$, $n = 1,2;\ m = 1,2$, the **chain rule** of derivations is

$$\frac{\partial J}{\partial w_{n,m}^{[1]}} = \frac{\partial J}{\partial \hat{y}_1}\frac{\partial \hat{y}_1}{\partial h_m}\frac{\partial h_m}{\partial z_m^{[1]}}\frac{\partial z_m^{[1]}}{\partial w_{n,m}^{[1]}} \quad \text{(Backward direction)}$$

For example,

$$J = \frac{1}{2}\left(\hat{y}_1 - y_1\right)^2 \Rightarrow \frac{\partial J}{\partial \hat{y}_1} = \hat{y}_1 - y_1$$

$$\hat{y}_1 = w_{1,1}^{[2]}h_1 + w_{2,1}^{[2]}h_2 \Rightarrow \frac{\partial \hat{y}_1}{\partial h_1} = w_{1,1}^{[2]}$$

$$h_1=\frac{1}{1+\exp(-z_1^{[1]})}\Rightarrow\frac{\partial h_1}{\partial z_1^{[1]}}=h_1\left(1-h_1\right)$$ (Remember this result, see *appendix A*)

$$z_1^{[1]}=w_{1,1}^{[1]}x_1+w_{2,1}^{[1]}x_2\Rightarrow\frac{\partial z_1^{[1]}}{\partial w_{2,1}^{[1]}}=x_2$$

Therefore,

$$\begin{cases}\dfrac{\partial J}{\partial w_{1,1}^{[1]}}=\dfrac{\partial J}{\partial\hat{y}_1}\dfrac{\partial\hat{y}_1}{\partial h_1}\dfrac{\partial h_1}{\partial z_1^{[1]}}\dfrac{\partial z_1^{[1]}}{\partial w_{1,1}^{[1]}}=\left(\hat{y}_1-y_1\right)w_{1,1}^{[2]}h_1\left(1-h_1\right)x_1\\ \dfrac{\partial J}{\partial w_{2,1}^{[1]}}=\dfrac{\partial J}{\partial\hat{y}_1}\dfrac{\partial\hat{y}_1}{\partial h_1}\dfrac{\partial h_1}{\partial z_1^{[1]}}\dfrac{\partial z_1^{[1]}}{\partial w_{2,1}^{[1]}}=\left(\hat{y}_1-y_1\right)w_{1,1}^{[2]}h_1\left(1-h_1\right)x_2\\ \dfrac{\partial J}{\partial w_{1,2}^{[1]}}=\dfrac{\partial J}{\partial\hat{y}_1}\dfrac{\partial\hat{y}_1}{\partial h_2}\dfrac{\partial h_2}{\partial z_2^{[1]}}\dfrac{\partial z_2^{[1]}}{\partial w_{1,2}^{[1]}}=\left(\hat{y}_1-y_1\right)w_{2,1}^{[2]}h_2\left(1-h_2\right)x_1\\ \dfrac{\partial J}{\partial w_{2,2}^{[1]}}=\dfrac{\partial J}{\partial\hat{y}_1}\dfrac{\partial\hat{y}_1}{\partial h_2}\dfrac{\partial h_2}{\partial z_2^{[1]}}\dfrac{\partial z_2^{[1]}}{\partial w_{2,2}^{[1]}}=\left(\hat{y}_1-y_1\right)w_{2,1}^{[2]}h_2\left(1-h_2\right)x_2\end{cases}$$

Then by using the gradient descent method,

$$\begin{cases}w_{m,1,\text{new}}^{[2]}=w_{m,1,\text{old}}^{[2]}-\lambda\dfrac{\partial J}{\partial w_{m,1}^{[2]}},\ m=1,2\\ w_{n,m,\text{new}}^{[1]}=w_{n,m,\text{old}}^{[1]}-\lambda\dfrac{\partial J}{\partial w_{n,m}^{[1]}},\ n=1,2;\ m=1,2\end{cases}$$

Example 1: BP algorithm

In the above example, randomly initialize the weights as

$$w_{1,1}^{[1]}=0.5,\ w_{2,1}^{[1]}=1.5,\ w_{1,2}^{[1]}=2.3,\ w_{2,2}^{[1]}=3,$$
$$w_{1,1}^{[2]}=1,\ w_{2,1}^{[2]}=1$$

The training set is with the inputs $x_1=1,\ x_2=0.5$, and the output is $y_1=4$.

Next, we will update the weights by using the BP algorithm under $\lambda=0.1$.

1) Calculate the feed-forward pass:

$$\begin{cases}z_1^{[1]}=w_{1,1}^{[1]}x_1+w_{2,1}^{[1]}x_2=1.25\\ z_2^{[1]}=w_{1,2}^{[1]}x_1+w_{2,2}^{[1]}x_2=3.8\end{cases}$$

$$\begin{cases} h_1 = \dfrac{1}{1+\exp(-z_1^{[1]})} = 0.777 \\ h_2 = \dfrac{1}{1+\exp(-z_2^{[1]})} = 0.978 \end{cases}$$

$$\hat{y}_1 = w_{1,1}^{[2]} h_1 + w_{2,1}^{[2]} h_2 = 1.755$$

$$J = \frac{1}{2}\left(\hat{y}_1 - y_1\right)^2 = 2.52$$

2) Calculate the output layer gradient:

$$\begin{cases} \dfrac{\partial J}{\partial w_{1,1}^{[2]}} = \left(\hat{y}_1 - y_1\right) h_1 = -1.744 \\ \dfrac{\partial J}{\partial w_{2,1}^{[2]}} = \left(\hat{y}_1 - y_1\right) h_2 = -2.196 \end{cases}$$

3) Calculate the input layer gradient:

$$\begin{cases} \dfrac{\partial J}{\partial w_{1,1}^{[1]}} = \left(\hat{y}_1 - y_1\right) w_{1,1}^{[2]} h_1 \left(1 - h_1\right) x_1 = -0.388 \\ \dfrac{\partial J}{\partial w_{2,1}^{[1]}} = \left(\hat{y}_1 - y_1\right) w_{1,1}^{[2]} h_1 \left(1 - h_1\right) x_2 = -0.194 \\ \dfrac{\partial J}{\partial w_{1,2}^{[1]}} = \left(\hat{y}_1 - y_1\right) w_{2,1}^{[2]} h_2 \left(1 - h_2\right) x_1 = -0.048 \\ \dfrac{\partial J}{\partial w_{2,2}^{[1]}} = \left(\hat{y}_1 - y_1\right) w_{2,1}^{[2]} h_2 \left(1 - h_2\right) x_2 = -0.024 \end{cases}$$

4) Update weights:

$$\begin{cases} w_{1,1,\text{new}}^{[2]} = 1 - 0.1 \times (-1.744) = 1.174 \\ w_{2,1,\text{new}}^{[2]} = 1 - 0.1 \times (-2.196) = 1.220 \\ w_{1,1,\text{new}}^{[1]} = 0.5 - 0.1 \times (-0.388) = 0.539 \\ w_{2,1,\text{new}}^{[1]} = 1.5 - 0.1 \times (-0.194) = 1.519 \\ w_{1,2,\text{new}}^{[1]} = 2.3 - 0.1 \times (-0.048) = 2.305 \\ w_{2,2,\text{new}}^{[1]} = 3 - 0.1 \times (-0.024) = 3.002 \end{cases}$$

5) : Calculate the feed-forward pass (the second round)

$$\begin{cases} z_1^{[1]} = w_{1,1}^{[1]} x_1 + w_{2,1}^{[1]} x_2 = 1.299 \\ z_2^{[1]} = w_{1,2}^{[1]} x_1 + w_{2,2}^{[1]} x_2 = 2.806 \end{cases}$$

$$\begin{cases} h_1 = \dfrac{1}{1+\exp(-z_1^{[1]})} = 0.786 \\ h_2 = \dfrac{1}{1+\exp(-z_2^{[1]})} = 0.943 \end{cases}$$

$$\hat{y}_1 = w_{1,1}^{[2]} h_1 + w_{2,1}^{[2]} h_2 = 2.073$$

$$J = \frac{1}{2}\left(\hat{y}_1 - y_1\right)^2 = 1.857 < 2.52$$

The prediction error reduced.

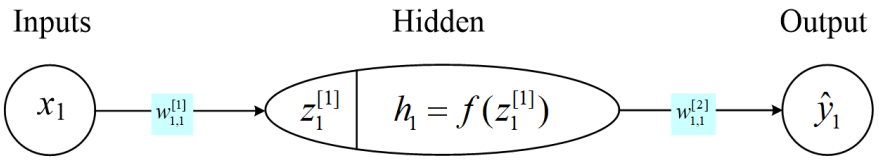
Quiz 1.2: (group work)

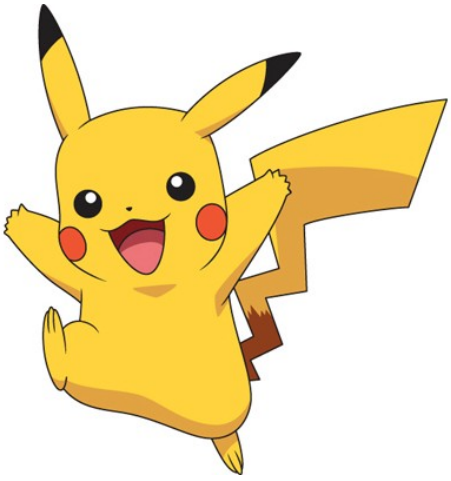
Update the neural network coefficients **once** by randomly initializing the weights $w_{1,1}^{[1]} = 1, w_{1,1}^{[2]} = 2$.

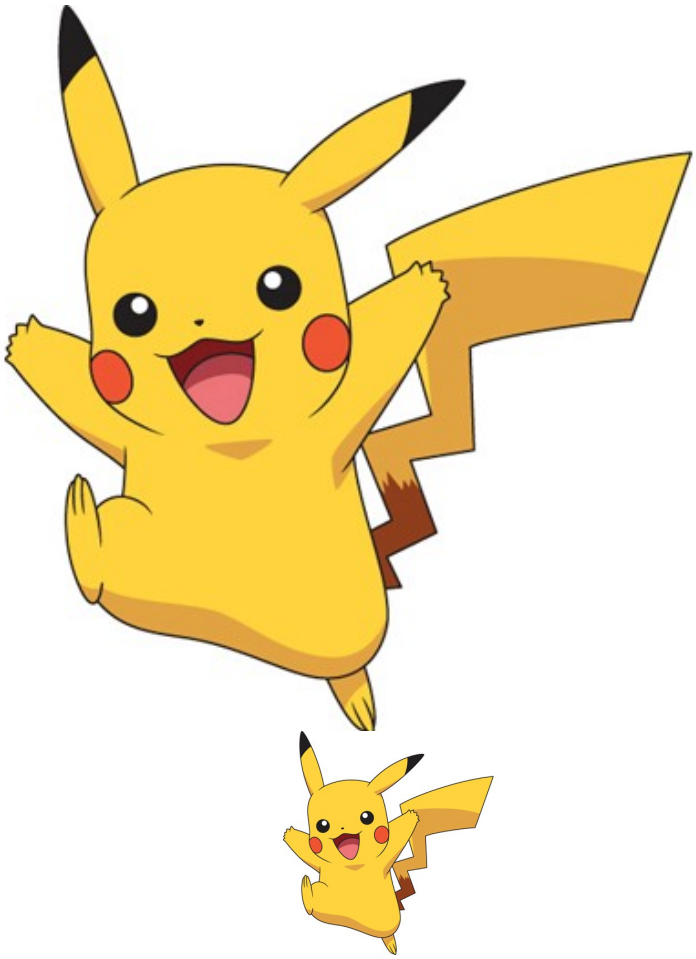
The training set is with the inputs $x_1 = 1$, and the output is $y_1 = 0.5$.

The learning rate is $\lambda = 0.1$.

The activation function is the ReLU function $f(z) = \max(0, z)$.

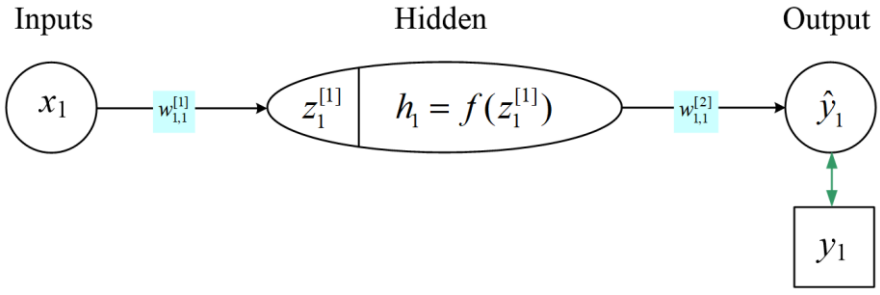






1.2 Multiple training set

The above example illustrates how to implement BP algorithm in updating neural network weights. Now we consider another example with multiple training sets.



where $f(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$.

Assuming we have 3 training sets $(x_1, y_1) = \left[(1,3),(2,5),(3,7)\right]$,

In this case, the cost function is

$$J = \frac{1}{3}\sum_{t=1}^{3}\frac{1}{2}\left(\hat{y}_{1,(t)} - y_{1,(t)}\right)^2$$

The gradient of the cost function is therefore

$$\frac{\partial J}{\partial w_{1,1}^{[i]}} = \frac{1}{3}\sum_{t=1}^{3}\frac{\partial J_{(t)}}{\partial w_{1,1}^{[i]}}, i = 1,2$$

Therefore, we need to calculate the gradient of the cost function with respect to weights for each training set. Then the average value of these gradients is the final gradient of the cost function with respect to weights.

Example 2: BP algorithm

In the above example, randomly initialize the weights as

$$w_{1,1}^{[1]} = 0.5,\ w_{1,1}^{[2]} = 1$$

and $\lambda = 1.$

1) Calculate the feed-forward pass (*t*=1):

$$z_{1,(1)}^{[1]} = w_{1,1}^{[1]} x_{1,(1)} = 0.5$$

$$h_{1,(1)} = \frac{\exp(z_{1,(1)}^{[1]}) - \exp(-z_{1,(1)}^{[1]})}{\exp(z_{1,(1)}^{[1]}) + \exp(-z_{1,(1)}^{[1]})} = 0.462$$

$$\hat{y}_{1,(1)} = w_{1,1}^{[2]} h_{1,(1)} = 0.462$$

$$J_{(1)} = \frac{1}{2}\left(\hat{y}_{1,(1)} - y_{1,(1)}\right)^2 = 3.221$$

2) Calculate the output layer gradient:

$$\frac{\partial J_{(1)}}{\partial w_{1,1}^{[2]}} = \left(\hat{y}_{1,(1)} - y_{1,(1)}\right) h_{1,(1)} = -1.173$$

3) Calculate the input layer gradient:

$$\frac{\partial J_{(1)}}{\partial w_{1,1}^{[1]}} = \left(\hat{y}_{1,(1)} - y_{1,(1)}\right) w_{1,1}^{[2]} \left(1 - {h_{1,(1)}}^2\right) x_{1,(1)} = -1.996$$

$$\left(\frac{\partial f(z)}{\partial z} = 1 - f^2(z)\right)$$

Calculate $\partial J_{(2)} / \partial w_{1,1}^{[i]}$ and $\partial J_{(3)} / \partial w_{1,1}^{[i]}$, $i = 1, 2$ following the **same process** as

$$\frac{\partial J_{(2)}}{\partial w_{1,1}^{[2]}} = -3.229, \frac{\partial J_{(2)}}{\partial w_{1,1}^{[1]}} = -3.555$$

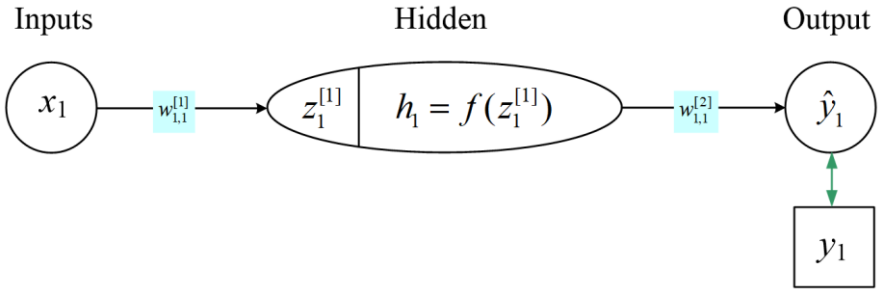
$$\frac{\partial J_{(3)}}{\partial w_{1,1}^{[2]}} = -5.516, \frac{\partial J_{(3)}}{\partial w_{1,1}^{[1]}} = -3.309$$

Then update the weights by using the gradient descent method:

$$\begin{cases} w_{1,1,\text{new}}^{[2]} = 1 - 1 \times \dfrac{(-1.173 - 3.229 - 5.516)}{3} = 4.306 \\ w_{1,1,\text{new}}^{[1]} = 1 - 1 \times \dfrac{(-1.996 - 3.555 - 3.309)}{3} = 3.953 \end{cases}$$

Quiz 1.3: (group work)

Consider a simple network with multiple training sets.



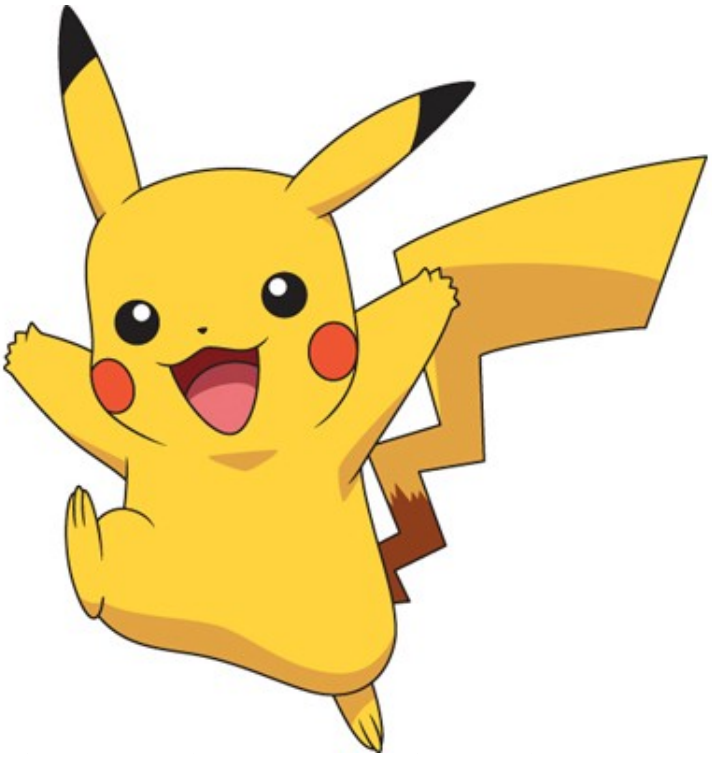
where $f(z) = \dfrac{1}{1 + \exp(-z)}$.

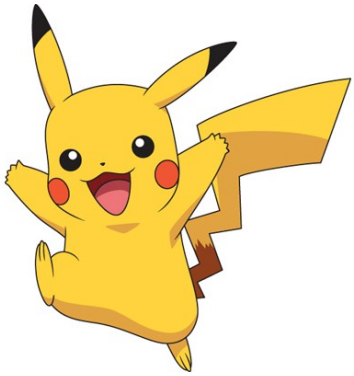
Assuming we have 2 training sets $(x_1, y_1) = \left[(1, 0.6), (2, 0.8)\right]$,

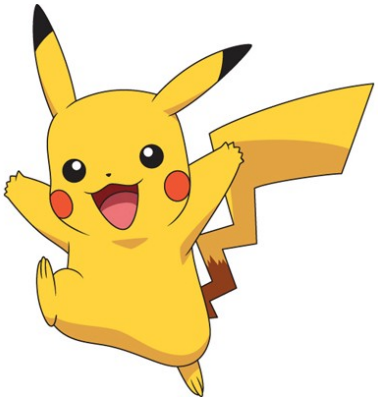
Update the neural network coefficients **once** by randomly initializing the weights $w_{1,1}^{[1]} = 0.1$, $w_{1,1}^{[2]} = 0.3$. The learning rate is $\lambda = 0.1$.

The cost function can be defined as

$$J = \frac{1}{2}\sum_{t=1}^{2}\frac{1}{2}\left(\hat{y}_{1,(t)} - y_{1,(t)}\right)^2$$







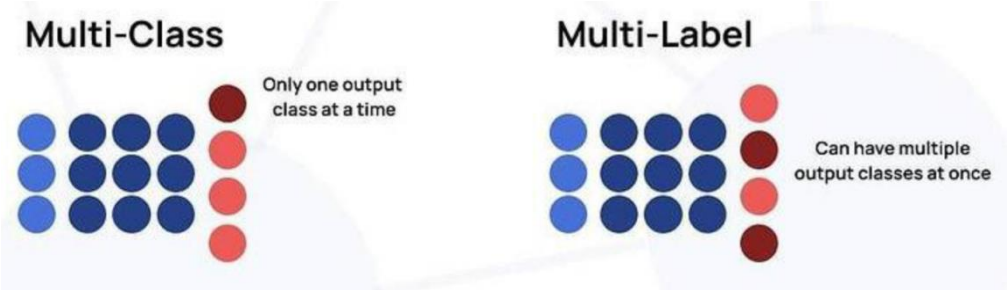
2 Training neural networks for regression and classification

We have known the difference in network structures for regression and classification. Another difference for regression and classification is the cost function.

For regression: Least squares

$$J(\mathbf{W}) = \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\frac{1}{2}\left(\hat{y}_{k,(t)} - y_{k,(t)}\right)^2$$

For classification [1]:



Maximum likelihood (cross-entropy for **multi-label classification**)

$$J(\mathbf{W}) = -\ln\prod_{t=1}^{T}\prod_{k=1}^{K}\hat{y}_{k,(t)}^{\,y_{k,(t)}}\left[1-\hat{y}_{k,(t)}\right]^{1-y_{k,(t)}}$$

$$= -\sum_{t=1}^{T}\sum_{k=1}^{K}\left\{y_{k,(t)}\ln\hat{y}_{k,(t)} + \left[1-y_{k,(t)}\right]\ln\left[1-\hat{y}_{k,(t)}\right]\right\}$$

where $y_{k,(t)} \in [0,1]$, $\hat{y}_{k,(t)} = P\left(y_{k,(t)} = 1 \middle| \bar{\mathbf{x}}_{(t)}\right)$

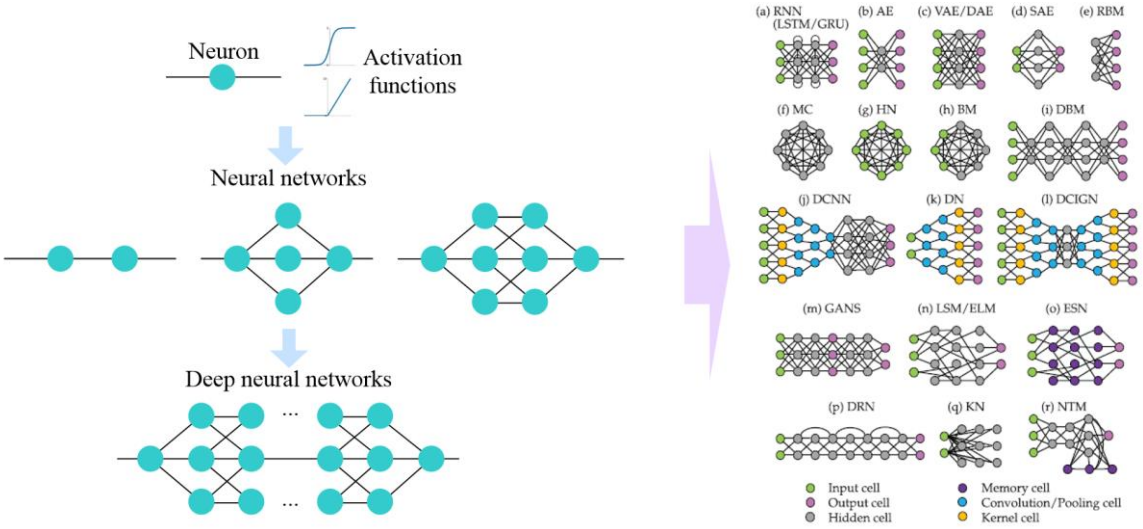
Maximum likelihood (cross-entropy for **multi-class classification**)

$$J(\mathbf{W}) = -\ln\prod_{t=1}^{T}\prod_{k=1}^{K}\hat{y}_{k,(t)}^{\,y_{k,(t)}} = -\sum_{t=1}^{T}\sum_{k=1}^{K}y_{k,(t)}\ln\hat{y}_{k,(t)}$$

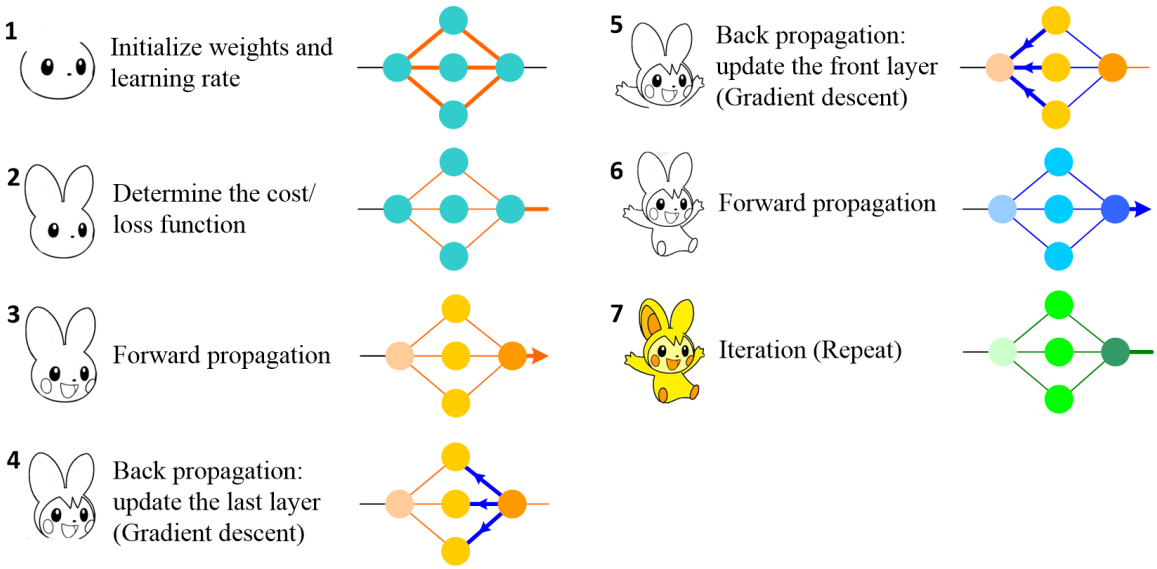
where $y_{k,(t)} \in [0,1]$, $\hat{y}_{k,(t)} = P\left(y_{k,(t)} = 1 \middle| \bar{\mathbf{x}}_{(t)}\right)$

Summarize:

Neural network



Neural network training



3 Further Readings

[1] Multi-class and multi-label classification

https://www.analyticsvidhya.com/blog/2021/07/demystifying-the-difference-between-multi-class-and-multi-label-classification-problem-statements-in-deep-learning/