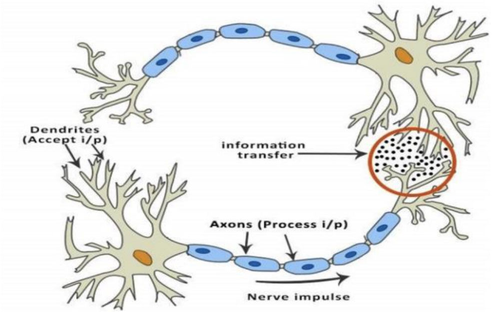
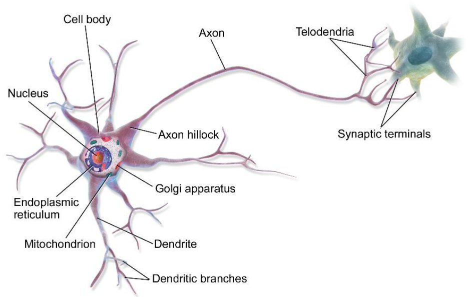
ANN and Handwritten Character Recognition

ANN

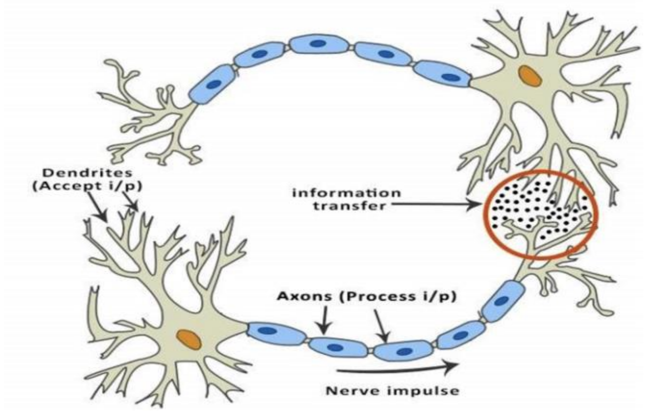
- The study of artificial neural networks was inspired by attempts to simulate biological neural systems (e. g. human brain).
- Basic structural and functional unit: nerve cells called neurons
- Work Mechanism
  - Different neurons are linked together via axons (轴突) and dendrite (树突)
  - When one neuron is excited after stimulation, it sends chemicals to the connected neurons, thereby changing the potential (电位) within these neurons.
  - If the potential (电位) of a neuron exceeds a "threshold", then it is activated and send chemicals to other neurons.

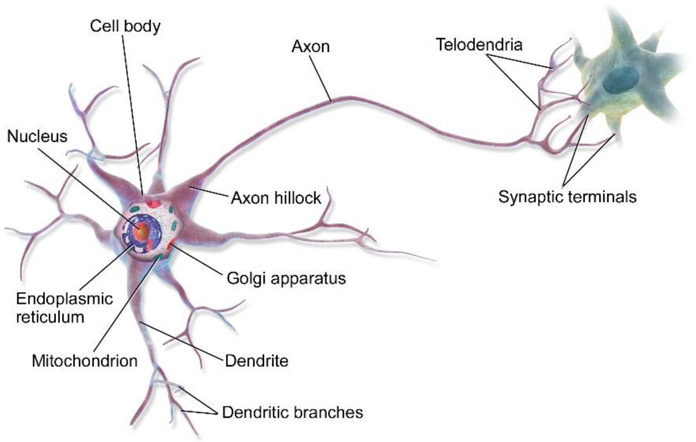




ANN

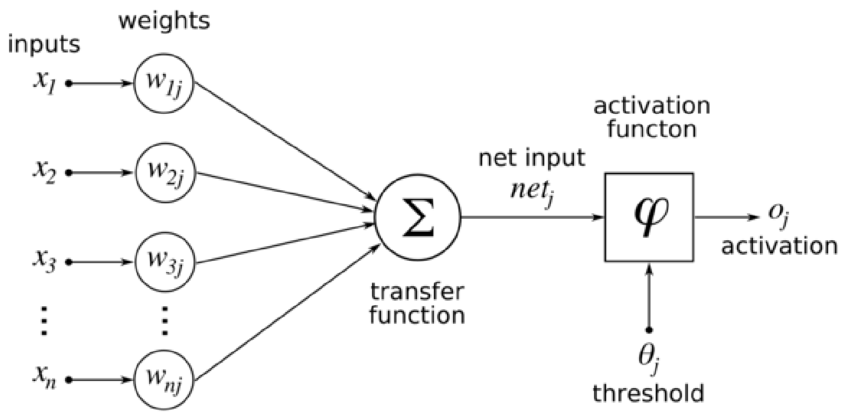
- A neuron is connected to the axons (轴突) of other neurons via dendrites (树突), which are extensions from the cell body of the neuron.
- The contact point between a dendrite (树突) and an axon (轴突) is called a synapse (突触).
- The human brain learns by changing the strength of the synaptic connection between neurons upon repeated stimulation by the same impulse.





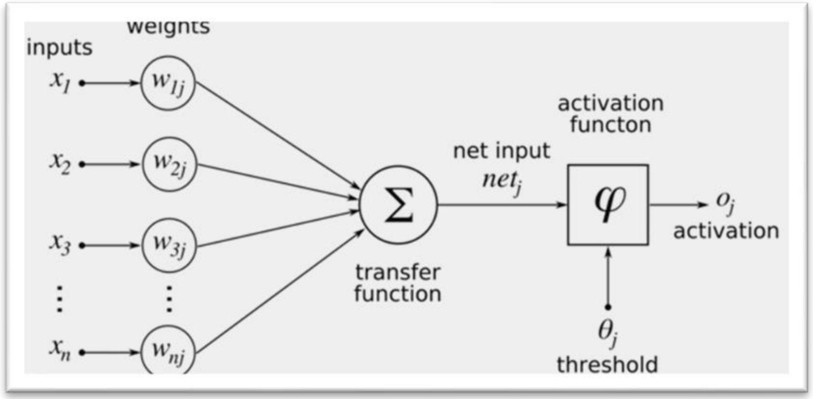
Artificial Neuron Mathematical Model

- Input: x_i from the i-th neuron
- Weights: connection weights (synapse)
- Output: $o_j = \varphi(\sum_{i=1}^{n} w_{ij} x_i - \theta_j)$



Artificial Neuron Model

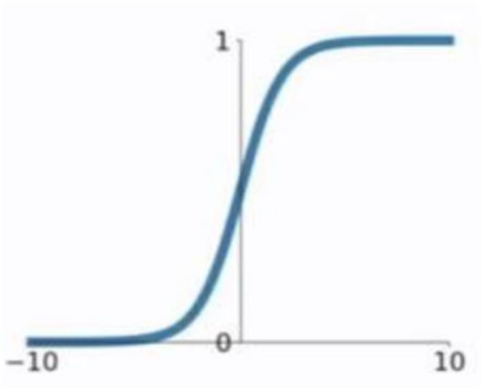
- Output: $o_j = \varphi(\sum_{i=1}^{n} w_{ij}x_i - \theta_j)$
- Ideal activation function: step function but inapplicable
- Common activation function：sigmoid，tanh，ReLU

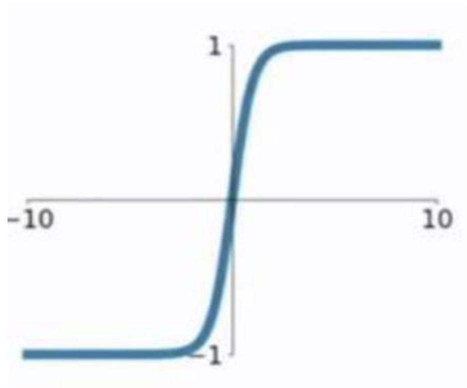


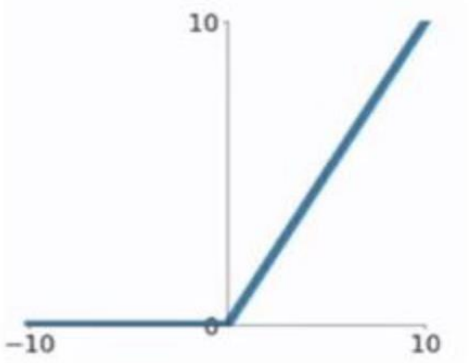
$$g(z) = \frac{1}{1+e^{-z}}$$

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases}$$





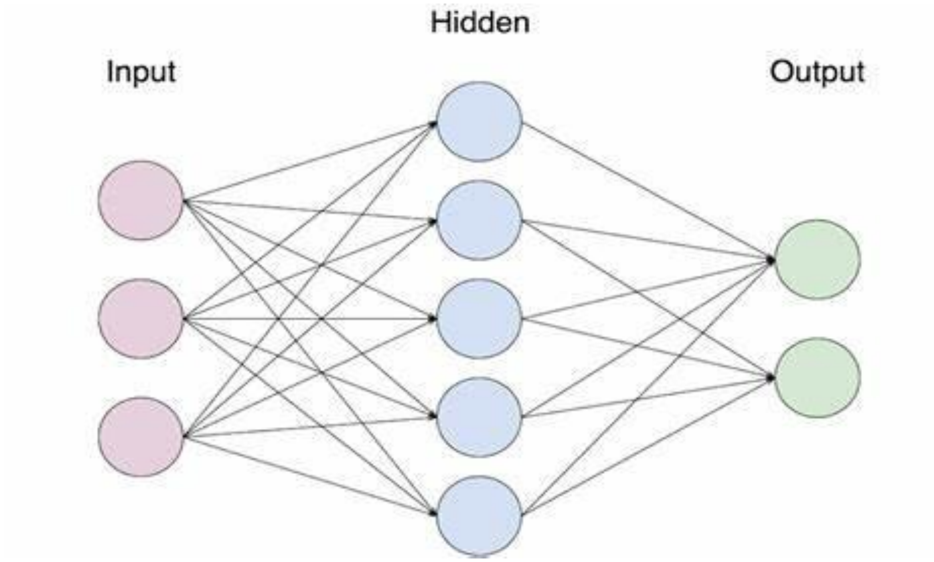


Artificial Neural Networks

- Consist of multiple artificial neurons

- Usually have the structure of an input layer, multiple hidden layer, an output layer

- The design of an NN or AutoML aims to design appropriate hidden layers and connection weights.



3-layer Feedforward neural networks

One Inference Process

$$z_j^{(1)} = f(a_j^{(1)})$$

$$z_j^{(2)} = f(a_j^{(2)})$$

$$y_l = f\left(a_l^{(3)}\right)$$

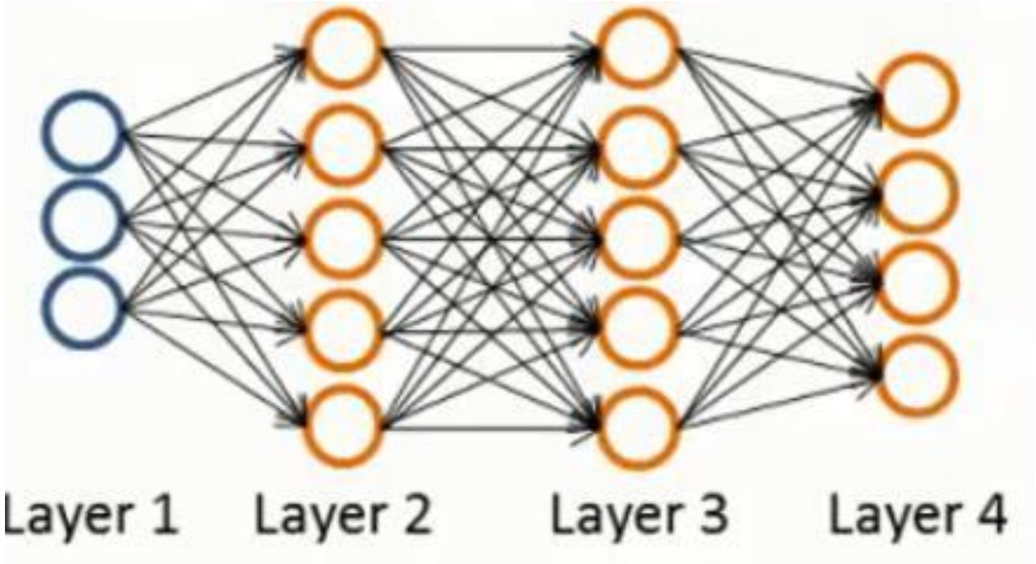
$$x \to a^{(1)} \to z^{(1)} \to a^{(2)} \to z^{(2)} \to a^{(3)} \to y$$

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i - \theta_j$$

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j - \theta_k$$

$$a_l^{(3)} = \sum_{k=1}^{N} w_{lk}^{(3)} z_k - \theta_l$$

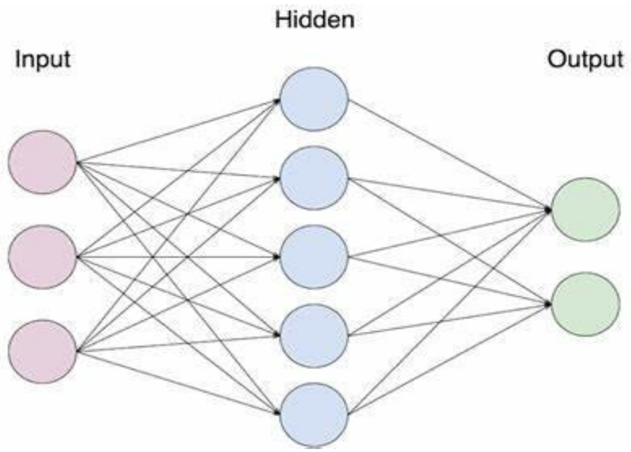
Superscript of w：layer index



Training of NN

- W and Threshold values decide the output of NN
- The training is to find appropriate values for W and Threshold
- The learning process is to tune weight matrix

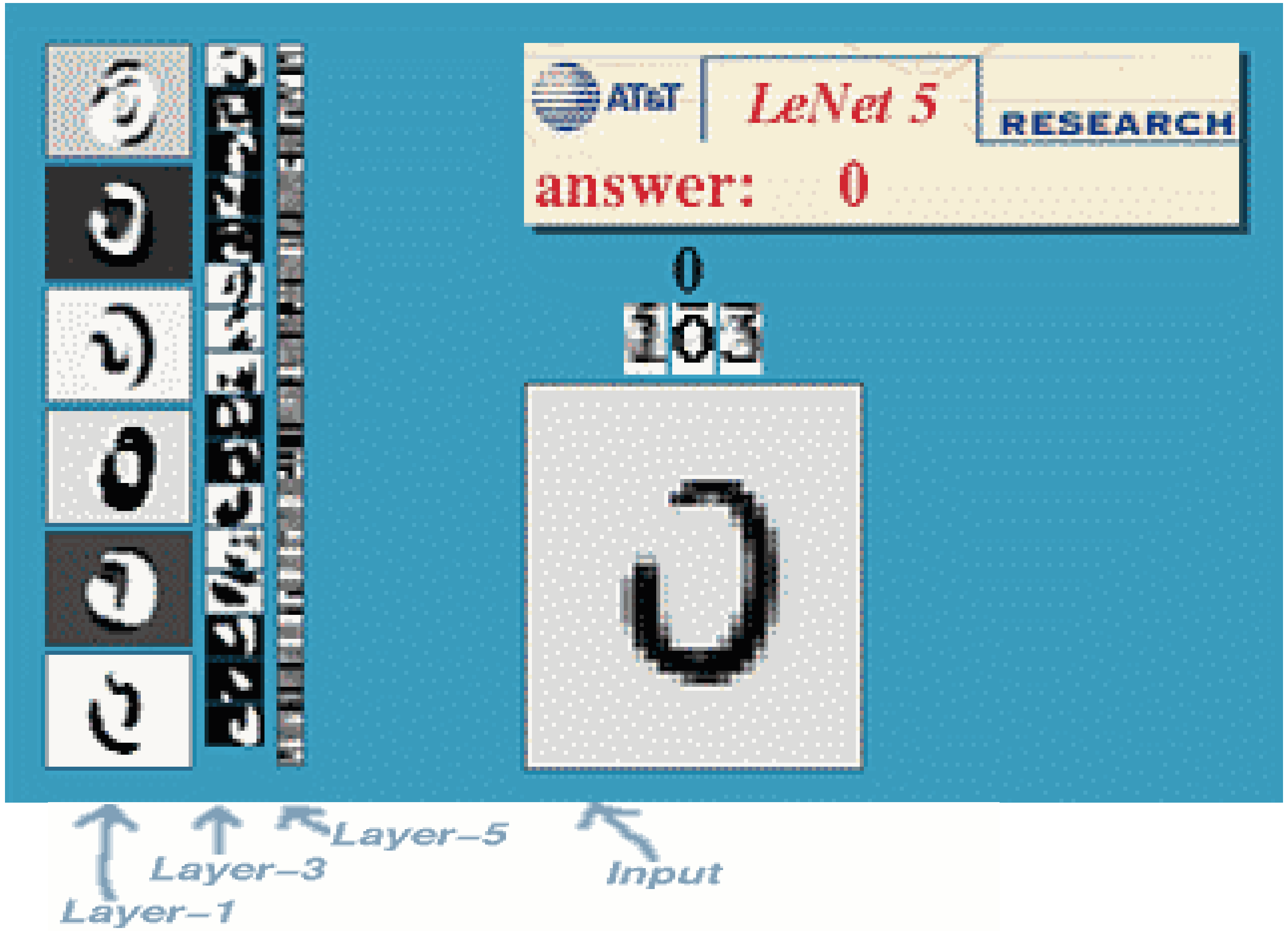
x1	x2	x3	l1	l2
1.0	0.1	0.3	1	0
0.1	1.5	1.2	1	0
1.1	1.1	2.0	0	1
0.2	0.2	0.3	0	1



$$Error\ Function\ (W,\theta) = \frac{1}{2}[(o1(W,\theta) - l1)^2 + (o2(W,\theta) - l2)^2]$$

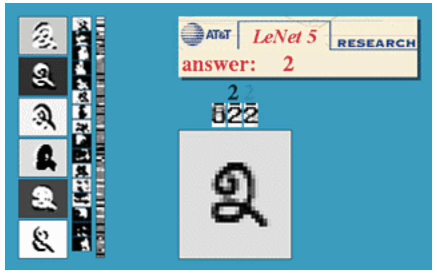
Calculate the gradient for (W,θ), then tune them

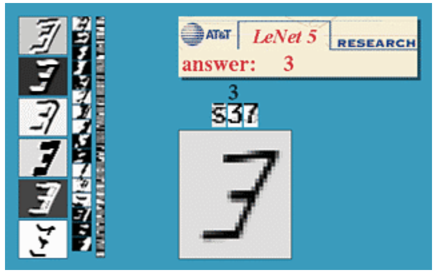
ANN Application: Handwritten character recognition

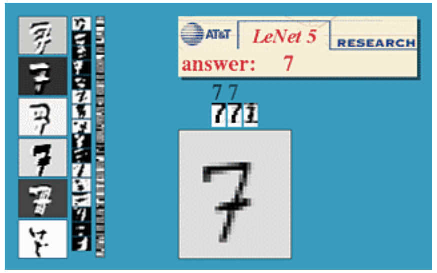


Performance of ANN

- Anti-interference ability, such as different sizes, digital distortion



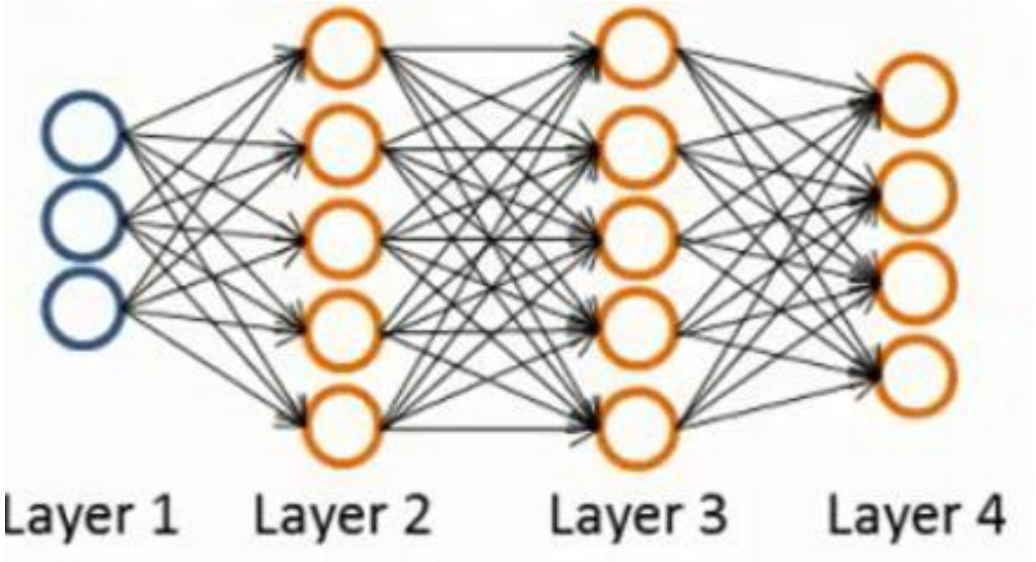




ANN-Input Layer to Hidden Layer 1

- Suppose 100 training samples, each sample: 28*28, feature: 784
- Layer 1: Input Layer, size=784
- Layer 2: Hidden Layer 1, size M = 300
- Layer 3: Hidden Layer 2, size N= 100
- Layer 4: Output Layer, size = 10

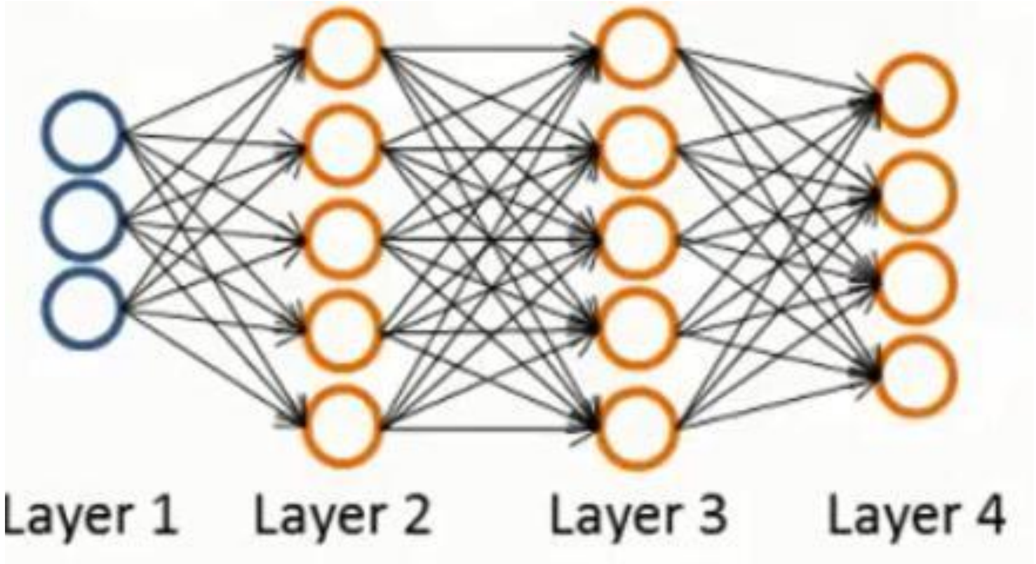
- Input: 100*784,
- Input to Hidden Layer 1: Weight Matrix: 784*300, θ: 1*300
- Weighted Sum for j-th neurons of hidden layer 1: $a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i - \theta_j^{(1)}$
- After activation function, output of j-th neuron: $z_j^{(1)} = f(a_j^{(1)})$
- Output of Hidden Layer 1, $z^{(1)}$: 100*300 matrix



ANN-Hidden Layer 1 to Hidden Layer 2

- Suppose 100 training samples, each sample: 28*28, feature: 784
- Layer 1: Input Layer, size=784
- Layer 2: Hidden Layer 1, size = 300
- Layer 3: Hidden Layer 2, size = 100
- Layer 4: Output Layer, size = 10

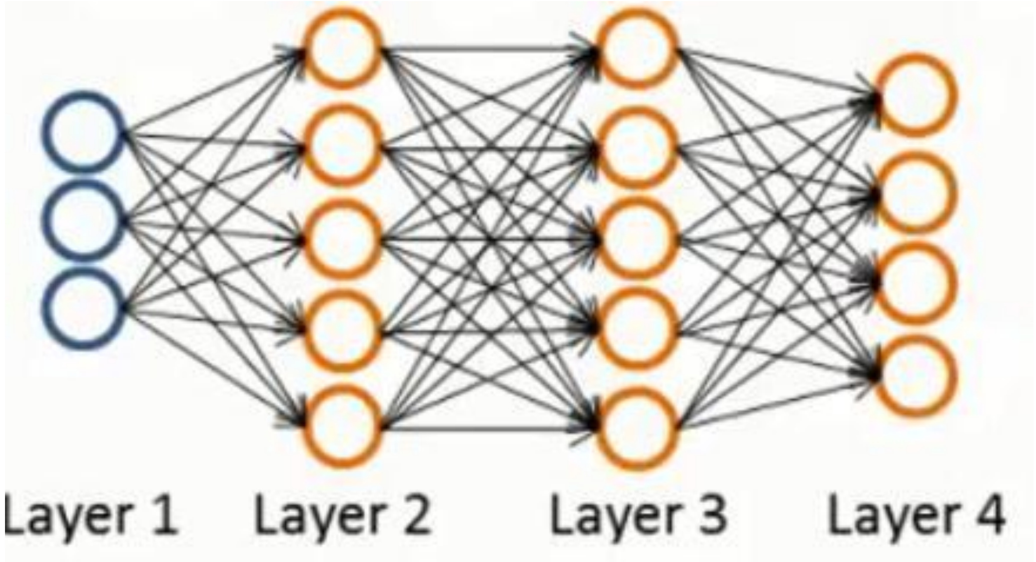
- Input: 100*784,
- Hidden Layer 1 to Hidden Layer 2: Weight Matrix: 300*100, θ: 1*100
- Weighted Sum for k-th neurons of hidden layer 2: $a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j - \theta_k^{(2)}$
- After activation function, output of k-th neuron: $z_k^{(2)} = f(a_k^{(2)})$
- Output of Hidden Layer 2, $z^{(2)}$: 100*100 matrix



ANN-Hidden Layer 2 to Output Layer

- Suppose 100 training samples, each sample: 28*28, feature: 784
- Layer 1: Input Layer, size=784
- Layer 2: Hidden Layer 1, size M = 300
- Layer 3: Hidden Layer 2, size N = 100
- Layer 4: Output Layer, size = 10

- Input: 100*784,
- Hidden Layer 2 to Output Layer: Weight Matrix: 100*10, B: 1*10
- Weighted Sum for l-th neurons of hidden layer 2: $\mathrm{a}_l^{(3)} = \sum_{k=1}^{N} w_{lk}^{(3)} z_k$ -$\theta_l^{(3)}$
- After activation function, output of l-th neuron:$y_l = \sigma\left(\mathrm{a}_l^{(3)}\right)$
- Output of Output Layer, y: 100*1 matrix
- σ: softmax function



Training of NN

- W and Threshold values decide the output of NN
- The training is to find appropriate values for W and Threshold so that the output is close to the true value
- Given the structure of NN, the learning process is to tune weight matrix in order to in order to minimize the difference between true value and prediction value.

Loss Function and Gradient Descent

- How to evaluate the difference？ Loss Function
- How to tune weight matrix？ optimization method (e. g. Gradient Descent)

As an Example

- Loss Function: $E_{total}\ (W,\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{k} -t_k^n ln y_k(x_n, w, \theta)$
- To avoid overfitting, add Regularization term:

$$E_{total}(W,\theta)=\frac{1}{N}\sum_{n=1}^{N} E(W,\theta) + \frac{\lambda}{2}||W||_2^2$$

- Gradient Descent: min f(x) -> x(t+1)=x(t)-η*f'(x(t)) (η： learning rate)

Gradient Descent

- $\frac{\partial E_{total}(W^{(t)},\theta^{(t)})}{\partial W^{(t)}}$

$=\frac{\partial(\frac{1}{N}\sum_{n=1}^{N} E_n(W^{(t)},\theta^{(t)})+\frac{\lambda}{2}||W^{(t)}||^2)}{\partial W^{(t)}}$

$=\frac{1}{N}\ \sum_{n=1}^{N}\ \frac{\partial E_n(W^{(t)},\theta^{(t)})}{\partial W^{(t)}}+\lambda W^{(t)}$

- $\frac{\partial E_{total}(W^{(t)},\theta^{(t)})}{\partial \theta^{(t)}}=\frac{1}{N}\ \sum_{n=1}^{N}\frac{\partial E_n(W^{(t)},\theta^{(t)})}{\partial \theta^{(t)}}$

Chain Rule

- Chain rule: to find the derivative of a composite function
- If $h(x)=f(g(x))$，then $h'(x)=f'(g(x))g'(x)$
- E.g.: $f(x)=2x+2$，$g(x)=3x+3$，$g(f(x))$ is a composite function, $g'(f(x))=6$

One Inference Process

$z_j^{(1)} = f(a_j^{(1)})$

$z_j^{(2)} = f(a_j^{(2)})$

$y_l = f\left(a_l^{(3)}\right)$

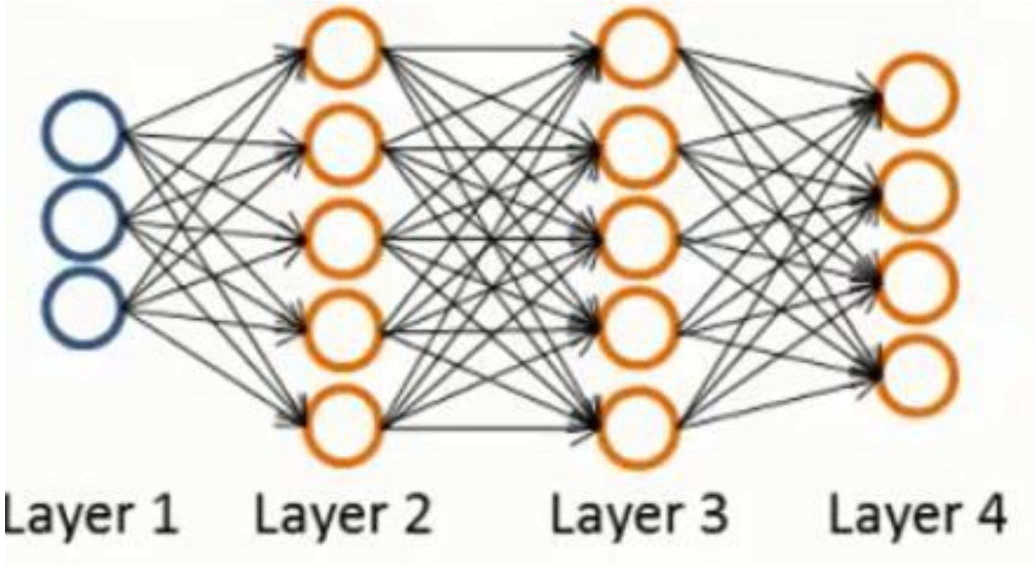
$$x \to a^{(1)} \to z^{(1)} \to a^{(2)} \to z^{(2)} \to a^{(3)} \to y$$

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i - \theta_j$$

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j - \theta_k$$

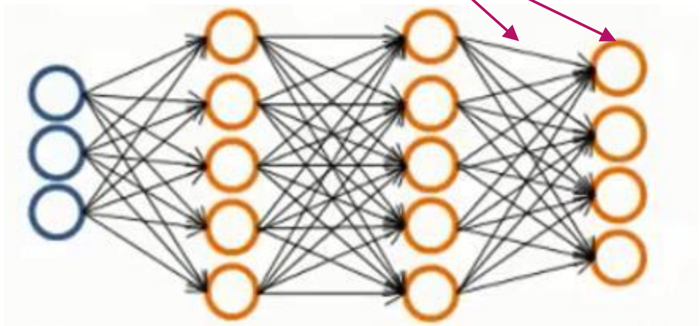
$$a_l^{(3)} = \sum_{k=1}^{N} w_{lk}^{(3)} z_k - \theta_l$$

Superscript of w：layer index



Derivative of E on $w^{(3)}$, $b^{(3)}$

- $\frac{\partial E}{\partial W^{(3)}} = \frac{\partial E\,(y)}{\partial y}\frac{\partial y(a^{(3)})}{\partial a^{(3)}}\frac{\partial a^{(3)})}{\partial w^{(3)}}$
- $\frac{\partial E}{\partial \theta^{(3)}} = \frac{\partial E\,(y)}{\partial y}\frac{\partial y(a^{(3)})}{\partial a^{(3)}}\frac{\partial a^{(3)})}{\partial \theta^{(3)}}$



$z_j^{(1)} = f(a_j^{(1)})$

$z_j^{(2)} = f(a_j^{(2)})$

$y_l = \sigma\left(a_l^{(3)}\right)$

$$x \to a^{(1)} \to z^{(1)} \to a^{(2)} \to z^{(2)} \to a^{(3)} \to y$$

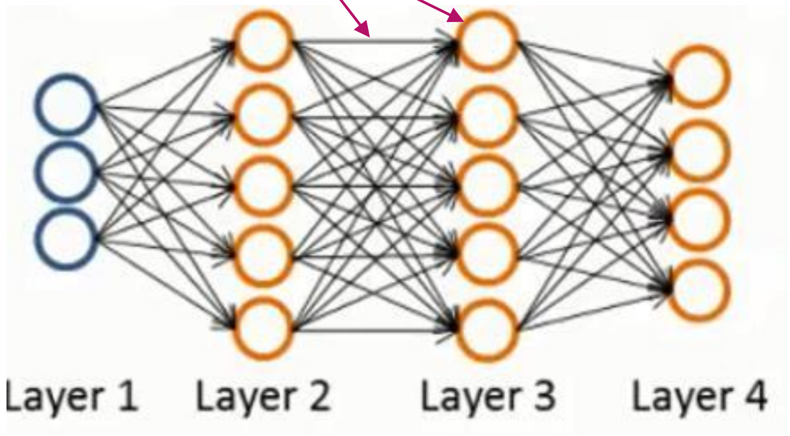
$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i - \theta_j^{(1)}$$

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j^{(1)} - \theta_k^{(2)}$$

$$a_l^{(3)} = \sum_{k=1}^{N} w_{lk}^{(3)} z_k^{(2)} - \theta_l^{(3)}$$

Derivative of E on $w^{(2)},\ b^{(2)}$

- $\frac{\partial E}{\partial W^{(2)}} = \frac{\partial E\ (y)}{\partial y} \frac{\partial y(a^{(3)})}{\partial a^{(3)}} \frac{\partial a^{(3)})}{\partial z^{(2)}} \frac{\partial z^{(2)})}{\partial a^{(2)}} \frac{\partial a^{(2)})}{\partial W^{(2)}}$
- $\frac{\partial E}{\partial \theta^{(2)}} = \frac{\partial E\ (y)}{\partial y} \frac{\partial y(a^{(3)})}{\partial a^{(3)}} \frac{\partial a^{(3)})}{\partial z^{(2)}} \frac{\partial z^{(2)})}{\partial a^{(2)}} \frac{\partial a^{(2)})}{\partial \theta^{(2)}}$



$\boxed{z_j^{(1)} = f(a_j^{(1)})}$ $\boxed{z_j^{(2)} = f(a_j^{(2)})}$ $\boxed{y_l = \sigma\left(a_l^{(3)}\right)}$

$$x \to a^{(1)} \to z^{(1)} \to a^{(2)} \to z^{(2)} \to a^{(3)} \to y$$

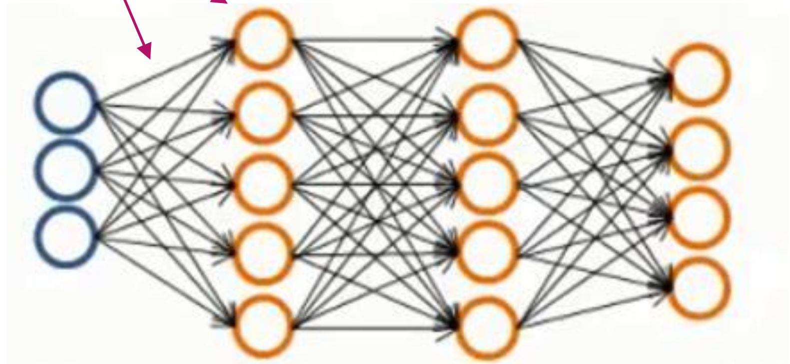
$$\boxed{a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i \text{ -} \theta_j}$$

$$\boxed{a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j^{(1)} \text{ -} \theta_k^{(2)}}$$

$$\boxed{a_l^{(3)} = \sum_{k=1}^{N} w_{jk}^{(3)} z_k^{(2)} \text{ -} \theta_k^{(3)}}$$

Derivative of E on $w^{(1)},\ b^{(1)}$

- $$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E\,(y)}{\partial y}\,\frac{\partial y(a^{(3)})}{\partial a^{(3)}}\,\frac{\partial a^{(3)})}{\partial z^{(2)}}\,\frac{\partial z^{(2)})}{\partial a^{(2)}}\,\frac{\partial a^{(2)})}{\partial z^{(1)}}\,\frac{\partial z^{(1)})}{\partial a^{(1)}}\,\frac{\partial a^{(1)})}{\partial W^{(1)}}$$
- $$\frac{\partial E}{\partial \theta^{(1)}} = \frac{\partial E\,(y)}{\partial y}\,\frac{\partial y(a^{(3)})}{\partial a^{(3)}}\,\frac{\partial a^{(3)})}{\partial z^{(2)}}\,\frac{\partial z^{(2)})}{\partial a^{(2)}}\,\frac{\partial a^{(2)})}{\partial z^{(1)}}\,\frac{\partial z^{(1)})}{\partial a^{(1)}}\,\frac{\partial a^{(1)})}{\partial \theta^{(1)}}$$



$$\boxed{z_j^{(1)} = f\left(a_j^{(1)}\right)} \quad \boxed{z_j^{(2)} = f\left(a_j^{(2)}\right)} \quad \boxed{y_l = \sigma\left(a_l^{(3)}\right)}$$

$$x \rightarrow a^{(1)} \rightarrow z^{(1)} \rightarrow a^{(2)} \rightarrow z^{(2)} \rightarrow a^{(3)} \rightarrow y$$

$$\boxed{a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i \text{ -} \theta_j}$$

$$\boxed{a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j^{(1)} \text{ -} \theta_k^{(2)}}$$

$$\boxed{a_l^{(3)} = \sum_{k=1}^{N} w_{jk}^{(3)} z_k^{(2)} \text{ -} \theta_k^{(3)}}$$

Gradient Descent Process (BP Algorithm)

1. Set values to max_iterations and η
2. Randomly generate $\mathrm{W}^{(0)}$ and $\mathrm{B}^{(0)}$
3. For t = 1 to max_iterations
4. Calculate the derivatives: $\frac{\partial E\ (W^{(t)},\theta^{(t)})}{\partial W^{(t)}}$ and $\frac{\partial E\ (W^{(t)},\theta^{(t)})}{\partial \theta^{(t)}}$
5. Update $\mathrm{W}^{(t+1)} = \mathrm{W}^{(t)} - \eta \frac{\partial E\ (W^{(t)},\theta^{(t)})}{\partial W^{(t)}}$, $B^{(t+1)} = B^{(t)} - \eta \frac{\partial E\ (W^{(t)},\theta^{(t)})}{\partial \theta^{(t)}}$
6. EndFor

Note:

E can be loss for a single sample, a batch of randomly selected sample, or all training samples.