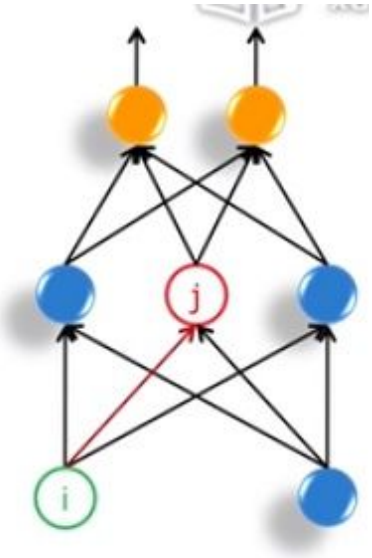
神经网络的学习算法（不唯一）：经典的BP算法（误差逆传播）

$$E_d(\vec{w}) \equiv \frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2 \qquad \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}x_{ji}$$



- x_{ji} = the i^{th} input to unit j
- w_{ji} = the weight associated with the i^{th} input to unit j
- $net_j = \sum w_{ji}x_{ji}$ (the weighted sum of inputs for unit j)
- o_j = the output of unit j
- t_j = the target output of unit j
- σ = the sigmoid function
- *outputs* = the set of units in the final layer
- *Downstream* (j) = the set of units directly taking the output of unit j as inputs

Downstream(j)：所有j神经元下游的神经元 --> 都用j的输出作为输入的神经元 netj：第j个神经元的输入

输出神经元计算权重：

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\cdot\frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2 = \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j} = -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1 - o_j)$$

$$\Delta w_{ji} = -\eta\frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji}$$

相对于感知机的多了一个o(1-o)，是因为激活函数不同（由线性-->Sigmoid）

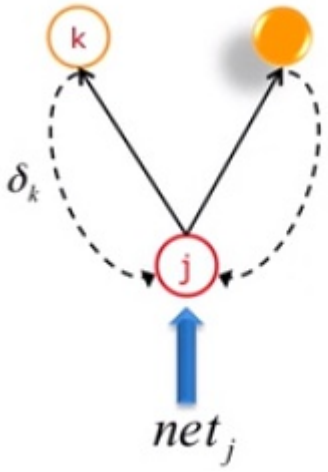
对于隐含层：

不知道隐含层期望输出是多少，没有办法确定误差

例如：领导派遣A,B出差办事，搞砸了，A,B的责任是很好判定的，但领导问责的话应该付多少责任呢？领导不是直接做这件事的人没有办法确定，AB是领导派遣的，将其错误反馈给领导，让其承担（误差逆传播）。但A对领导言听计从，则A犯的错误领导责任是多少；B基本不听领导的，B犯错领导承担多少？所以是需要乘以一个权重的。

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1 - o_j)$$



$$\delta_k = -\frac{\partial E_d}{\partial net_k}$$

$$\delta_j = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

神经元网络BP算法的基本思想

先把外面的ouput误差算好，然后一层一层的向里传播，用的时候是正向传播前馈网络，但是训练的时候要逆回来。

- BACKPROPAGATION (*training_examples*, η, n_{in}, n_{out}, n_{hidden})
- Create a network with n_{in} inputs, n_{hidden} hidden units and n_{out} output units.
- Initialize all network weights to small random numbers.
- Until the termination condition is met, Do
  - For each <*x*, *t*> in *training_examples*, Do
    - Input the instance *x* to the network and computer the output *o* of every unit.
    - For each output unit *k*, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

    - For each hidden unit *h*, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in outputs} w_{kh}\delta_k$$

    - Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} = w_{ji} + \eta\delta_j x_{ji}$$

先计算输出层，再计算隐含层反馈回来的

- Convergence and Local Minima
  - The search space is likely to be highly multimodal.
  - May easily get stuck at a local solution.
  - Need multiple trials with different initial weights.

- Evolving Neural Networks
  - Black-box optimization techniques (e.g., Genetic Algorithms)
  - Usually better accuracy
  - Can do some advanced training (e.g., structure + parameter).
  - Xin Yao (1999) "Evolving Artificial Neural Networks", *Proceedings of the IEEE*, pp. 1423-1447.
- Representational Power
- Deep Learning

神经网络的表达能力很强，两个隐含层，节点数足够多的话，可以接近任意一个函数的

神经网络的问题：

过学习问题：要有训练集与校验集。边修改边测试，要在Validation的最低点停止（之后过学习）
BP算法的基础是求导 --> 最害怕局部最优点，或者较为平的地方（导数为0），但的确存在，所以很容易会收敛在局部最优点，所以神经网络的训练常常需要try很多回，也可以添加一个冲量
学习率：不能过大（一直错过最优点，难收敛），不能太小（很容易入小坑--> 局部最优）。学习绿的选择很重要，可以动态调整。