**Nerual Networks for Machine Learning**

* **The core of nerual network**

Transformation

Linear Combination

Unit

=

****

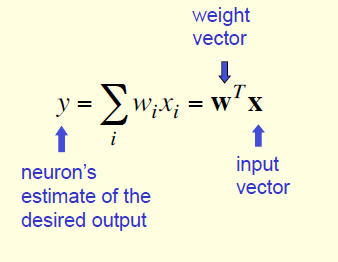
**Learning the weights of a linear neuron**

**Perceptron learning procedure cannot be generalised to hidden layers**

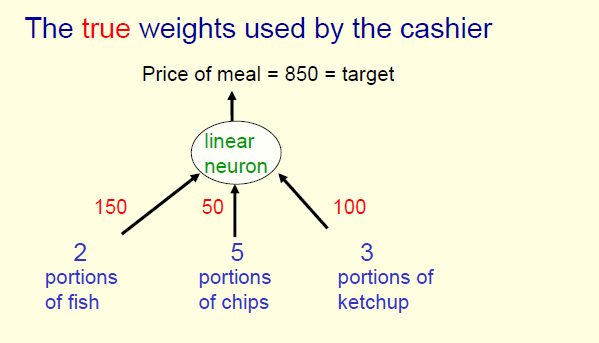
* The perceptron convergence procedure works by ensuring that every time the weights change, they get closer to every “ generaously feasible” set of weights.
* So “multi-layer” neural netwok do not use the perceptron learning procedure.

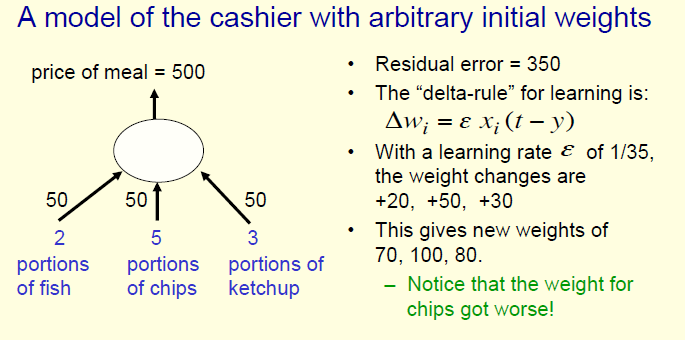
**Different way to show that a learning procedure makes progress**

* Instead of showing the weights get closer to a good set of weights, show that the actual output values get closer the target values.
  + This can be true even for non-convex problems in which there are many quite different sets of weights that work well and averaging two good sets of weights may give a bad set of weights.
  + Example: linear neuron with a squared error measure



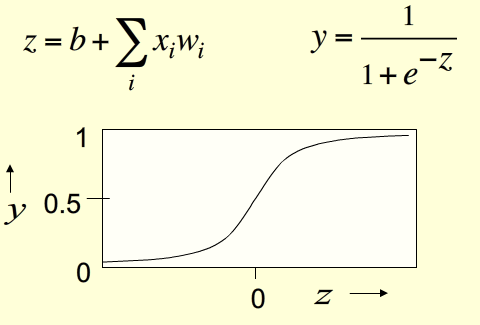
**Delta-rule for convergence procedure**



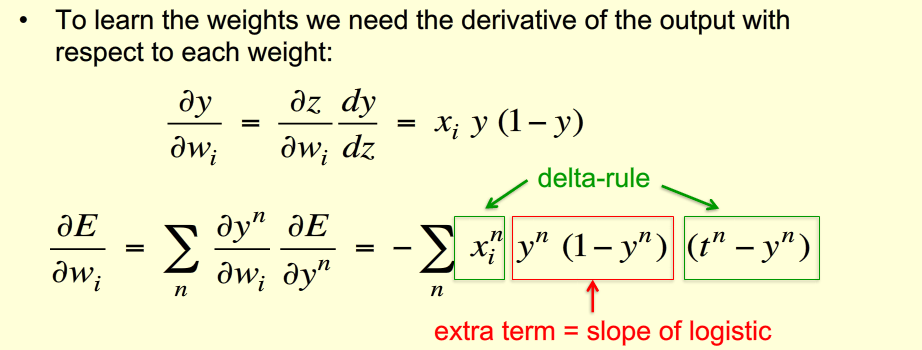


**Learning weigths for logistic neurons**

* **Output is smooth and bounded.**
  + **Why smooth?**
  + **For different problem, we choose different function to transform the input. For example, if it is a regression (prediction) problem, the output we expect is continual, so we prefer logistic function. If it is a classification problem, the output is discrete, so we prefer section function.**



* **Learn weights**



**Back Propagation Algorithm**

指定随机初始权值

随机选择训练集中的样本

前向传播输入

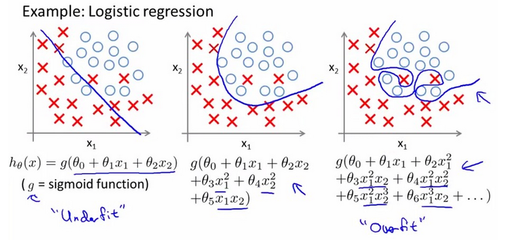
反向传播误差

达到预定义均方误差

结束

* 反向误差传播
  + 首先计算出输出层的误差
  + 隐层的神经单元的误差是上一次（输出单元误差\*连接权重）的和
  + 然后按照负梯度下降来更新权重

过拟合问题：



* 提前结束训练，在有限时间内训练神经网络
* 在训练时加入噪声，提高神经网络的范化能力

避免陷入局部最小值，而非找到全局最小值

* 加入正则化项，可增加拟合曲线的“灵活应变处理空隙”