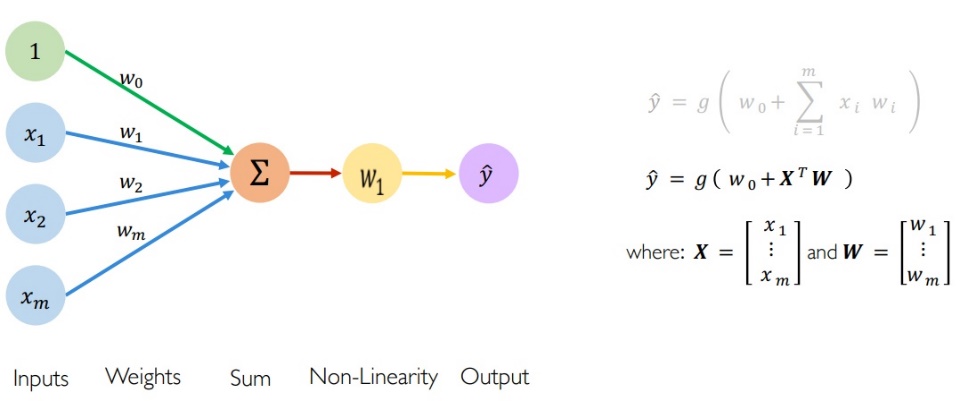
# 概述

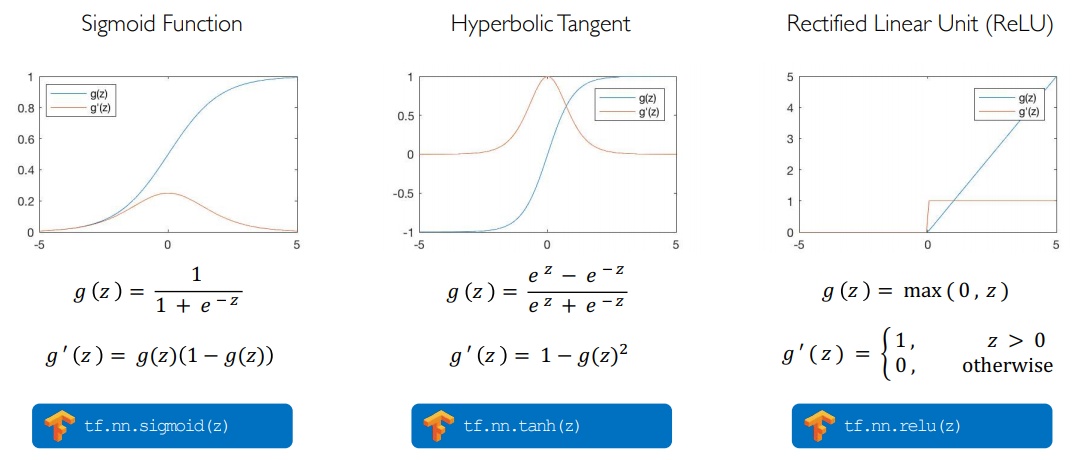
## 感知机（Perceptron）

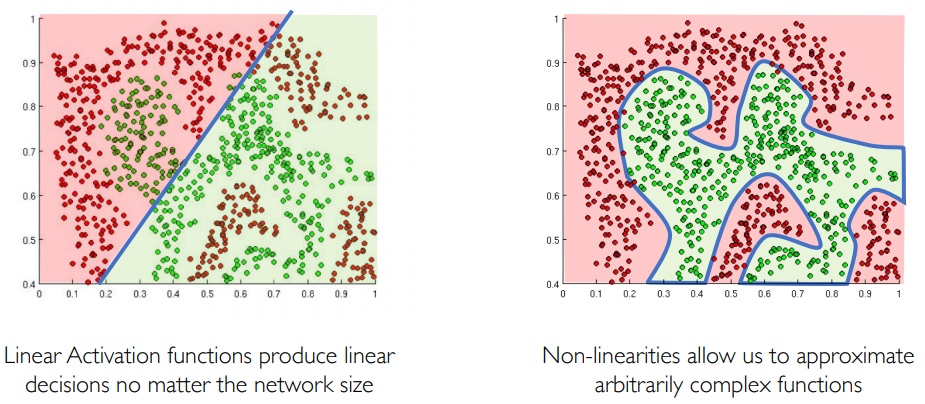
前向传播（Forward Propagation）模型



## 激活函数（Activation Functions）

作用：为神经网络引入**非线性**描述能力，



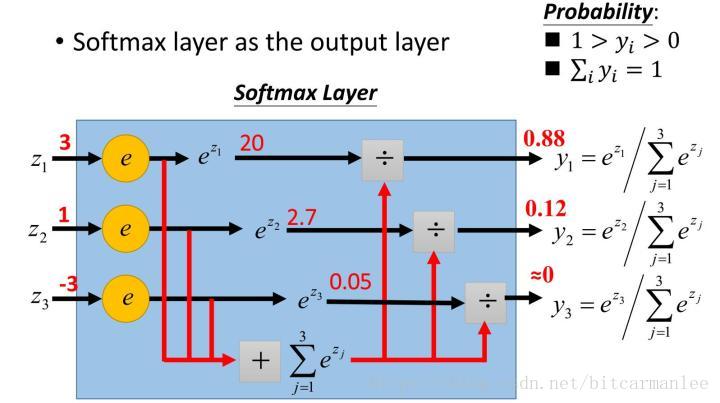


## 稠密层（dense layer）

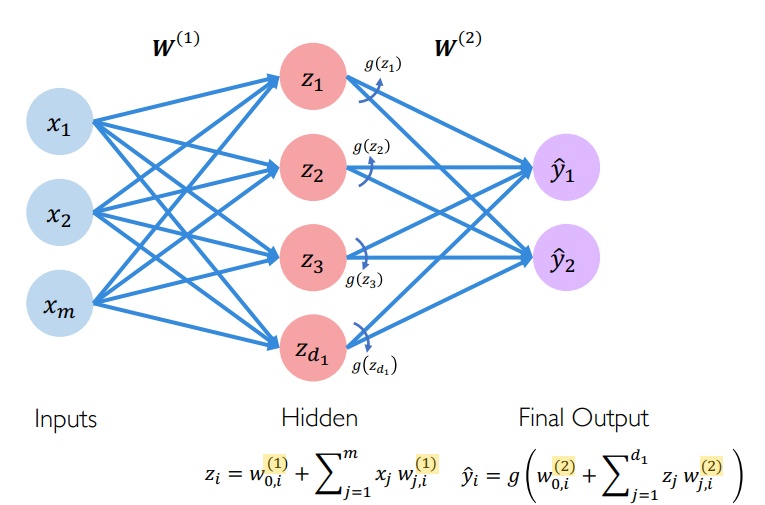
又称全连接层（fully connected layer），all inputs are **densely** connected to outputs.

作用：起到“分类器”的作用，将学到的“分布式特征表示”映射到样本标记空间，池化层和卷积层做的是特征工程，即将原始数据映射到隐含层特征空间。

Softmax：把输入映射为0-1之间的实数，并且归一化保证和为1。

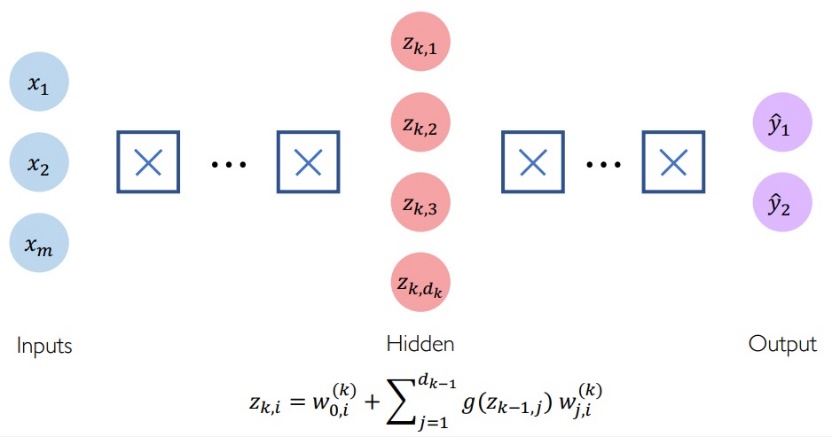


## 单隐含层神经网络



## 深度神经网络模型

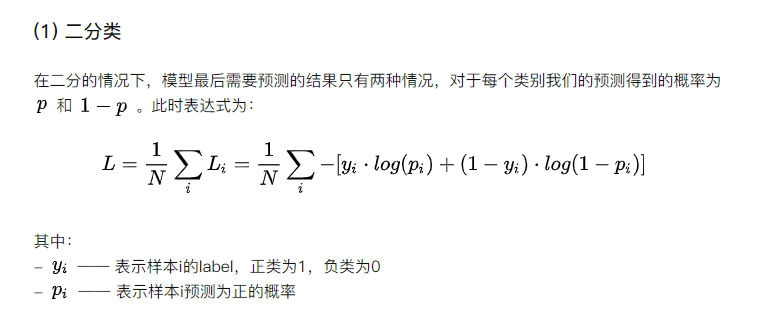
输入与输出之间含有多个隐含层

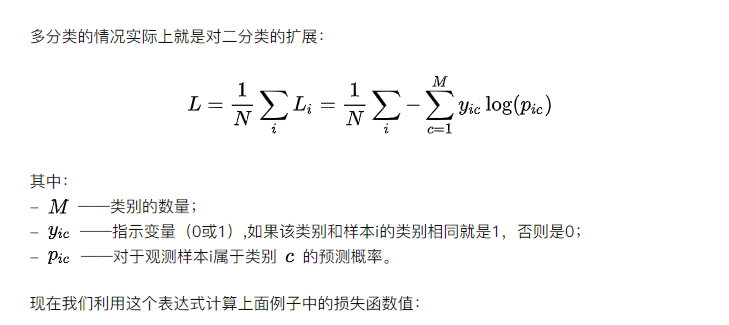


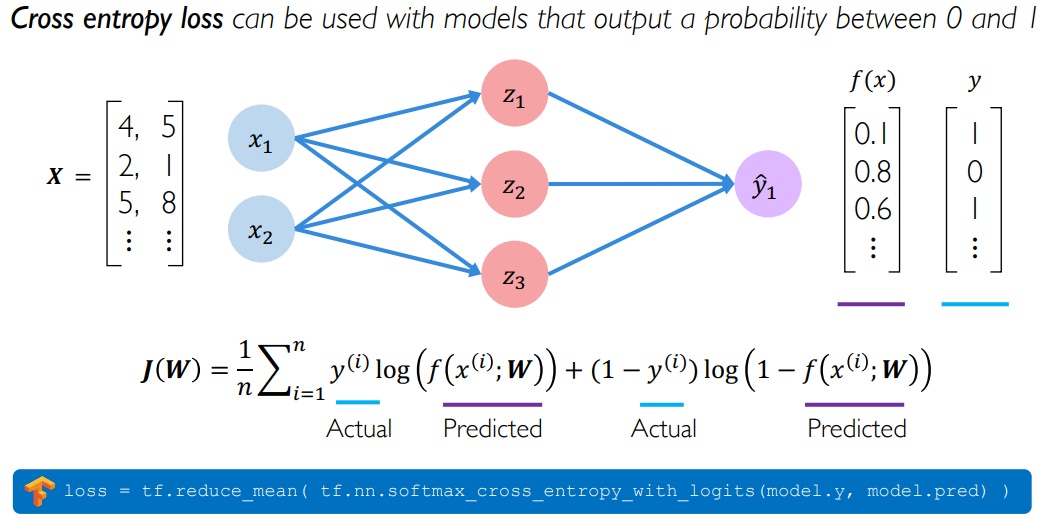
## 经验损失（empirical loss）

神经网络预测值与实际值的差距

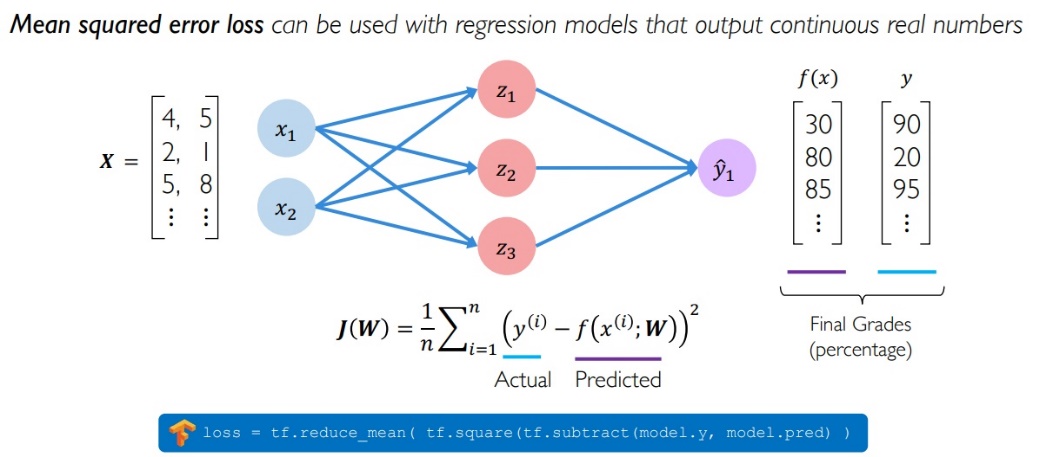
（1）交叉熵损失函数（Cross Entropy Error Function）







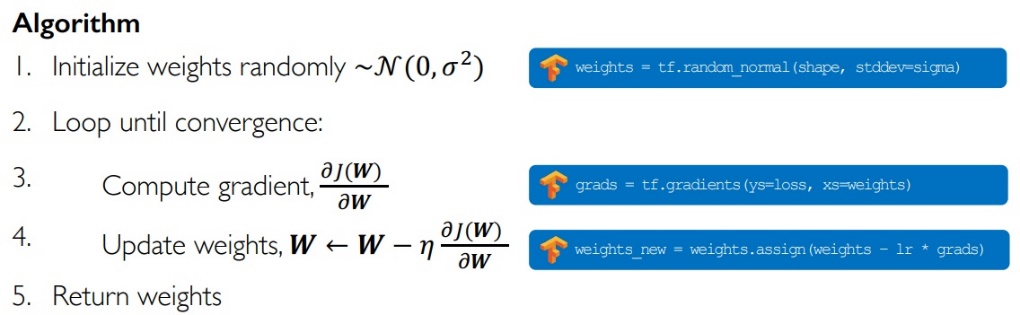
（2）均方根误差函数（mean squared error loss）



## 优化器

以最小化损失函数为目标，找到最优的权重组合

方法：梯度下降

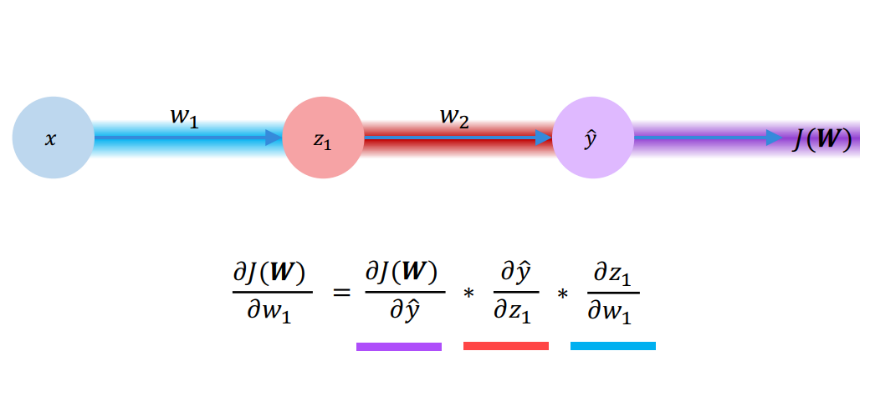


η称为学习率（learning rate），具备自适应学习率的优化器：

* SGD
* Momentum
* Adagrad
* Adadelta
* Adam
* RMSProp

## 反向传播（Back Propagation）

作用：计算梯度



## Mini-batch梯度下降

随机选取训练集的少部分数据进行训练，从而降低运算的复杂度。

Mini-batches一般跟随机梯度下降（SGD）结合在一起使用。方法是在每一代训练之前，对数据进行随机混洗，然后创建mini-batches（在batchsize内随机选取一部分样本），对mini-batch内的样本用梯度下降训练网络权重。

batchsize：批大小，每个选取的训练即样本个数（**batchsize>=minibatchsize**）。

iteration：完成1个batchsize的训练称为1 iteration

epoch：整个训练集完成1次训练称为1epoch

## 梯度下降计算方法

（1）**遍历全部数据集算一次损失函数**，然后算函数对各个参数的梯度，更新梯度。这种方法每更新一次参数都要把数据集里的所有样本都看一遍，计算量开销大，计算速度慢，不支持在线学习，这称为**batch gradient descent**，批梯度下降（每次选全部）。

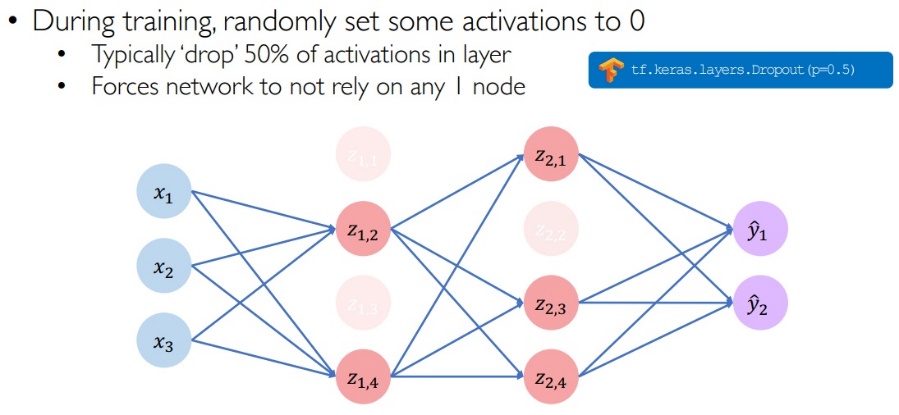
（2）**每看一个数据就算一下损失函数**，然后求梯度更新参数，这个称为随机梯度下降，**stochastic gradient descent**。这个方法速度比较快，但是收敛性能不太好，可能在最优点附近晃来晃去，hit不到最优点。两次参数的更新也有可能互相抵消掉，造成目标函数震荡的比较剧烈（每次选一个）。

（3）现在一般采用的是一种折中手段，**mini-batch gradient decent**，**小批的梯度下降**，这种方法**把数据分为若干个批**，**按批来更新参数**，这样，**一个批中的一组数据共同决定了本次梯度的方向**，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大（每次选取batch的一部分）。

## 正则化

（1）DropOut

随机设置激活函数为0



（2）Early Stopping

