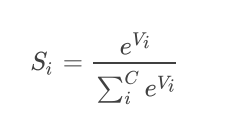
* + 1. **softmax函数用于多分类问题**

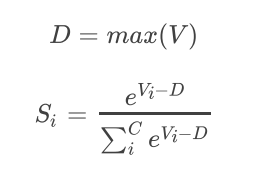
Softmax 在机器学习和深度学习中有着非常广泛的应用。尤其在处理多分类（C > 2）问题，分类器最后的输出单元需要Softmax 函数进行数值处理。

函数的定义如下所示

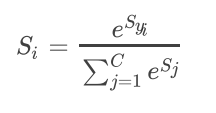


其中，Vi 是分类器前级输出单元的输出。i 表示类别索引，总的类别个数为 C。Si 表示的是当前元素的指数与所有元素指数和的比值。Softmax 将多分转化为相对概率 。

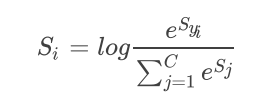
实际应用中，使用 Softmax 需要注意数值溢出的问题。因为有指数运算，如果 V 数值很大，经过指数运算后的数值往往可能有溢出的可能。所以，需些数值处理：即 V 中的每个元素减去 V 中的最大值 。



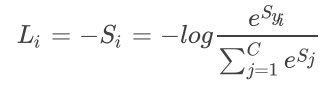
我们知道，线性分类器的输出是输入 x 与权重系数的矩阵相乘：s = Wx。对于多分类问题，使用 Softmax 对线性输出进行处理。这一小节我们来探讨Softmax 损失函数 。

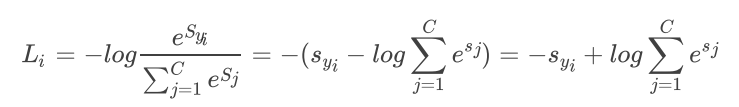


由于 log 运算符不会影响函数的单调性，我们对 Si 进行 log 操作：

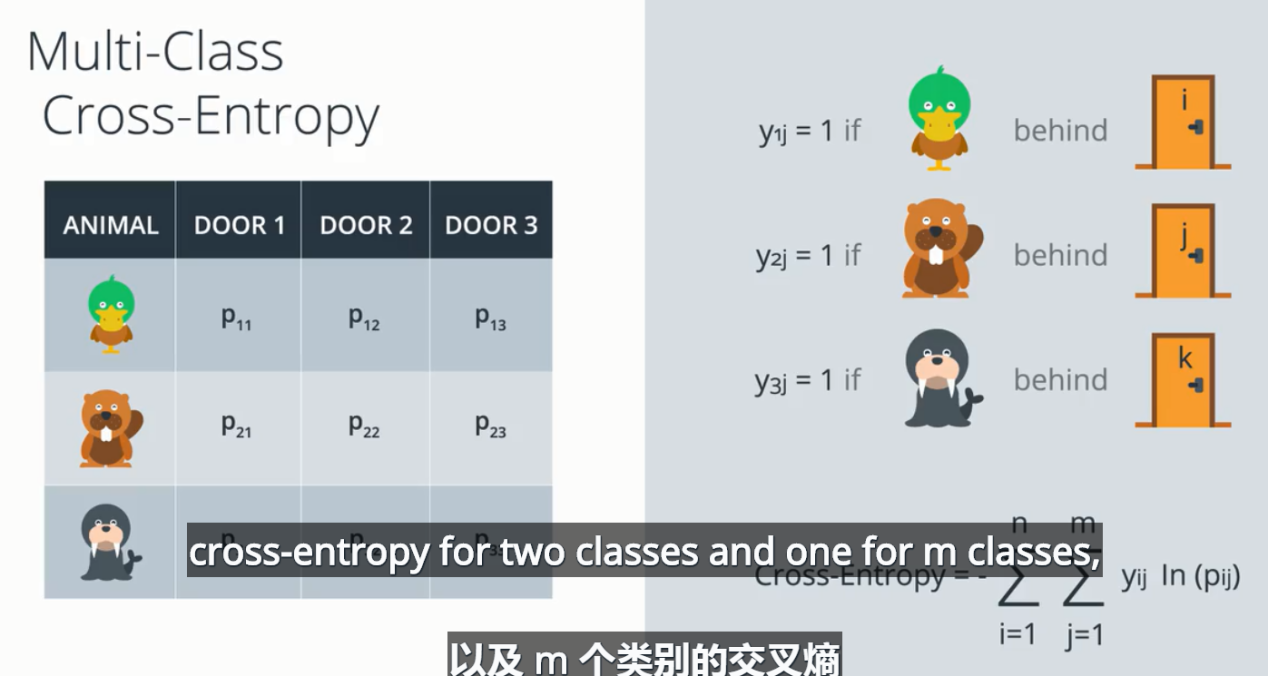


我们希望 Si 越大越好，即正确类别对应的相对概率越大越好，那么就可以对 Si 前面加个负号，来表示损失函数：





或者表示为以下形式：

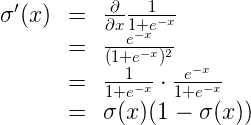


**1.1.1梯度下降实现**

在上几个视频中，我们了解到为了最小化误差函数，我们需要获得一些导数。我们开始计算误差函数的导数吧。首先要注意的是 s 型函数具有很完美的导数。即



原因是，我们可以使用商式计算它：

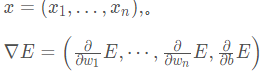
[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/d1b60bb7-71e9-4fc5-bc7f-b9f689abb3cf/concepts/0d92455b-2fa0-4eb8-ae5d-07c7834b8a56)

[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/d1b60bb7-71e9-4fc5-bc7f-b9f689abb3cf/concepts/0d92455b-2fa0-4eb8-ae5d-07c7834b8a56)现在，如果有 mm 个样本点，标为

误差公式是：

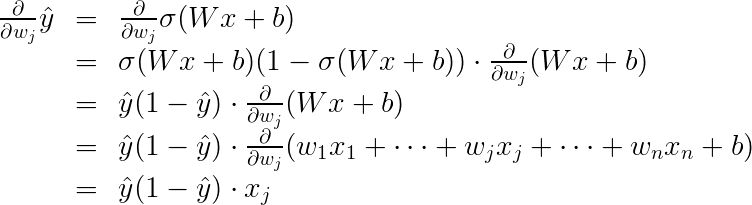


预测是

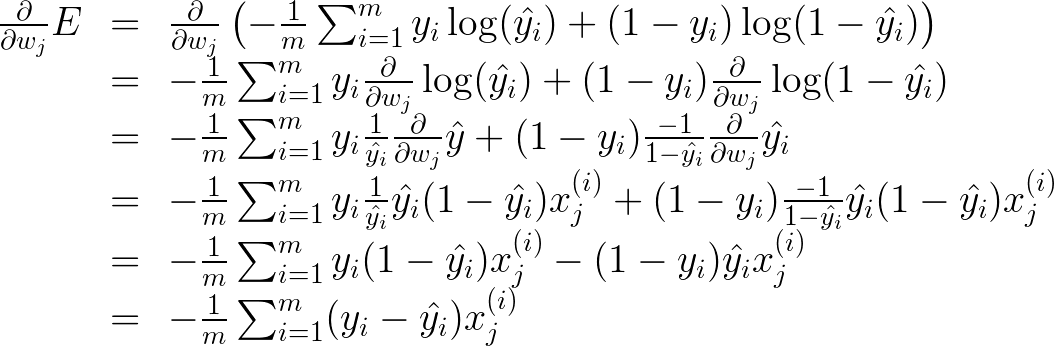
我们的目标是计算 E 在单个样本点 x 时的梯度（偏导数），其中 x 包含 n 个特征，即

为此，首先我们要计算

 因此：

[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/d1b60bb7-71e9-4fc5-bc7f-b9f689abb3cf/concepts/0d92455b-2fa0-4eb8-ae5d-07c7834b8a56)

最后一个等式是因为唯一非常量项相对于 wj正好是wj\*xj, 明显具有导数xj现在可以计算

[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/d1b60bb7-71e9-4fc5-bc7f-b9f689abb3cf/concepts/0d92455b-2fa0-4eb8-ae5d-07c7834b8a56)

类似的计算将得出：

[IMG_259](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/d1b60bb7-71e9-4fc5-bc7f-b9f689abb3cf/concepts/0d92455b-2fa0-4eb8-ae5d-07c7834b8a56)

这个实际上告诉了我们很重要的规则。对于具有坐标(x1​,…,xn)的点，标签 y, 预测y^​,该点的误差函数梯度是

总之：



如果思考下，会发现很神奇。梯度实际上是标量乘以点的坐标，什么是标量？也就是标签和预测直接的差别。这意味着，如果标签与预测接近（表示点分类正确），该梯度将很小，如果标签与预测差别很大（表示点分类错误），那么此梯度将很大。请记下：小的梯度表示我们将稍微修改下坐标，大的梯度表示我们将大幅度修改坐标。

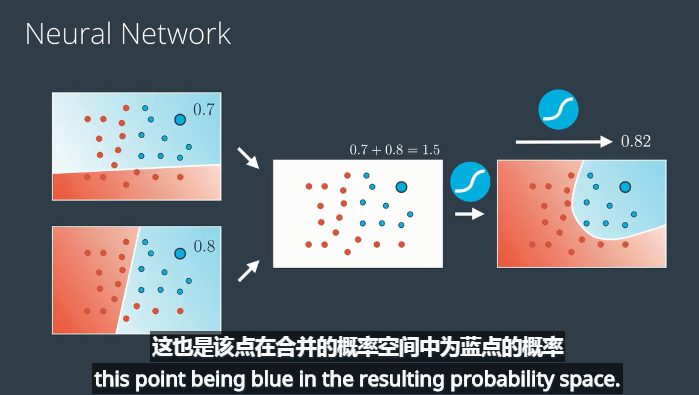
如果觉得这听起来像感知器算法，其实并非偶然性！稍后我们将详细了解。

感知器算法和梯度下降算法区别？

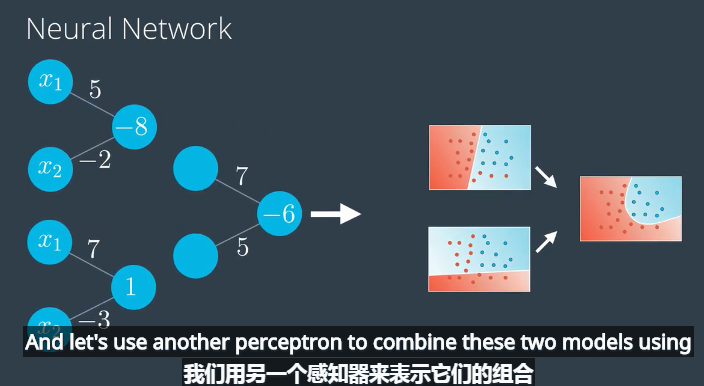
感知器算法只有0和1两个值，离散的两个值来判断，分类正确的点则不处理，分类错误的点则调整权重和偏置让其更靠近直线。而梯度下降算法则是将预测值利用概率分布为0-1之间连续值，对于分类正确的点，梯度下降算法让直线远离点，这样使该点的概率接近于1，而对于分类错误的点，同样也是让其靠近直线。

**1.1.2非线性模型**

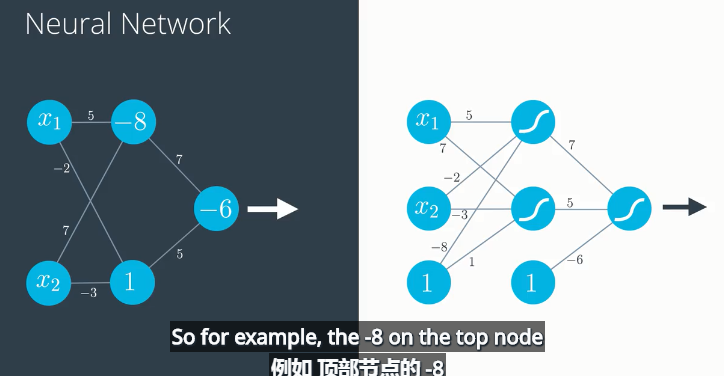
当无法采用线性分类来划分，采用曲线即非线性的方法来实现，即多层感知器模型



将多个线性模型的感知器组合在一起，形成非线性分类

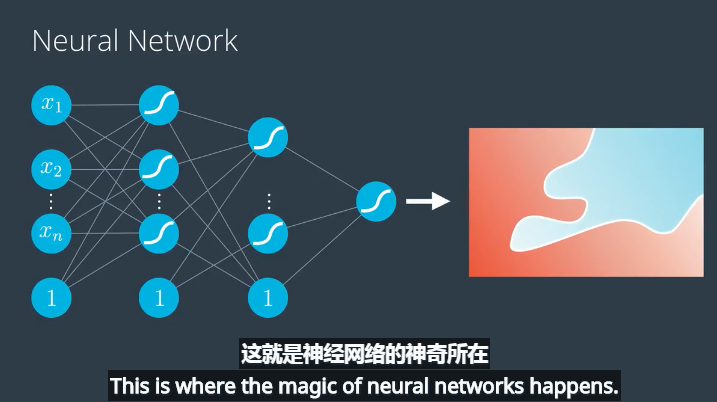


将上图的格式清理为当前多层感知器的形式：

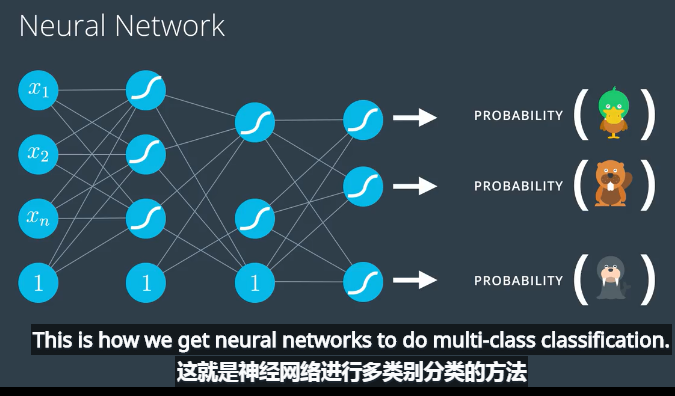


标准的神经网络模型：

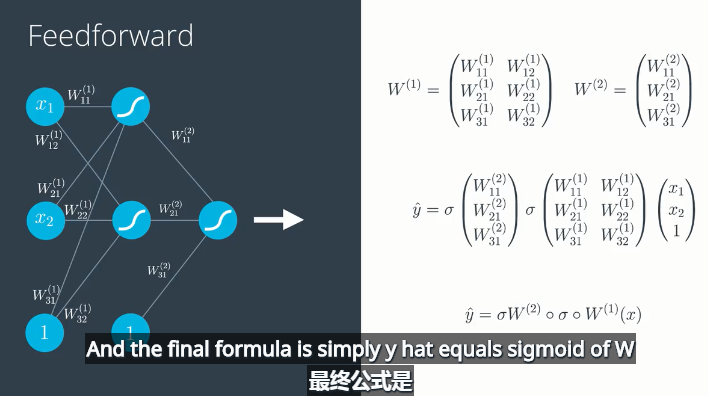
1. 中间层为隐藏层，隐藏层对线性模型组合形成非线性分类
2. 隐藏层组合后还可以添加隐藏层，这样形成复杂的非线性界限
3. 二维对应曲线，三维对应曲面，多维对应超曲面



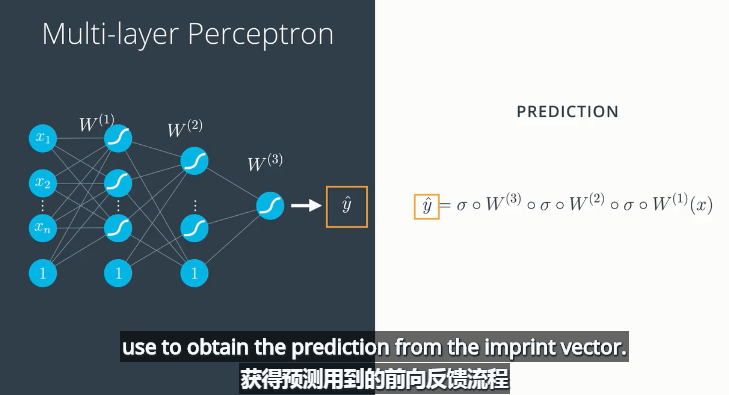
针对多分类问题，在输出层添加多个神经元，分别计算对应不同类别的概率即可。



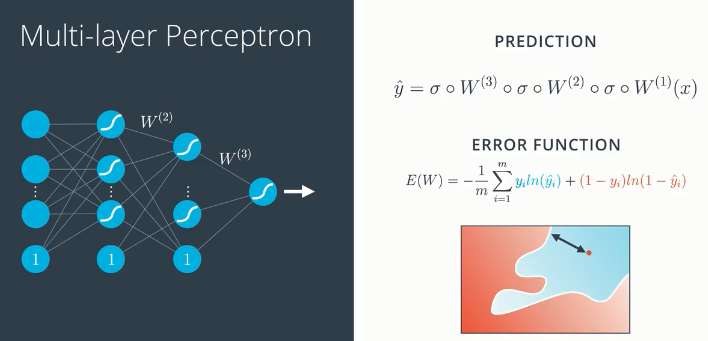
**1.1.3前馈网络与反向传播**



针对多层感知器



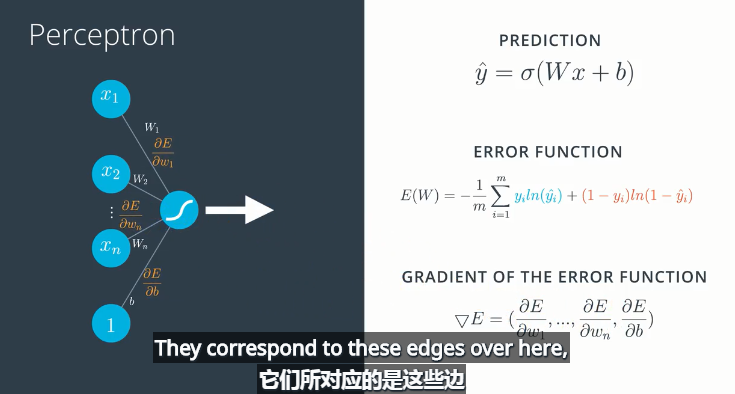
依然是计算误差函数



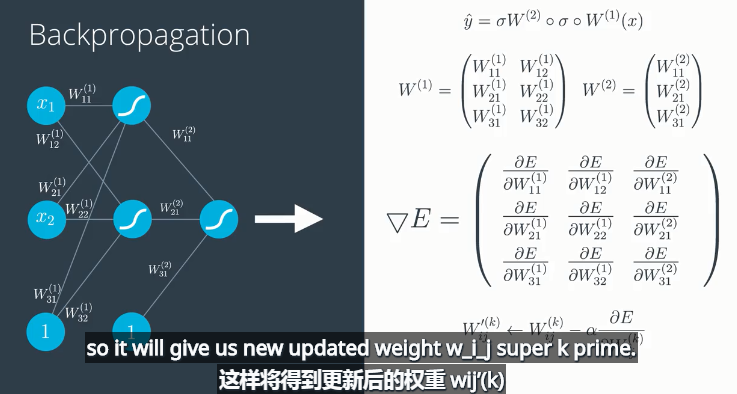
现在，我们准备好训练神经网络了。为此，我们将使用一种方法，叫做****反向传播****。简而言之，反向传播将包括：

* 进行前向反馈运算。
* 将模型的输出与期望的输出进行比较。
* 计算误差。
* 向后运行前向反馈运算（反向传播），将误差分散到每个权重上。
* 更新权重，并获得更好的模型。
* 继续此流程，直到获得很好的模型。

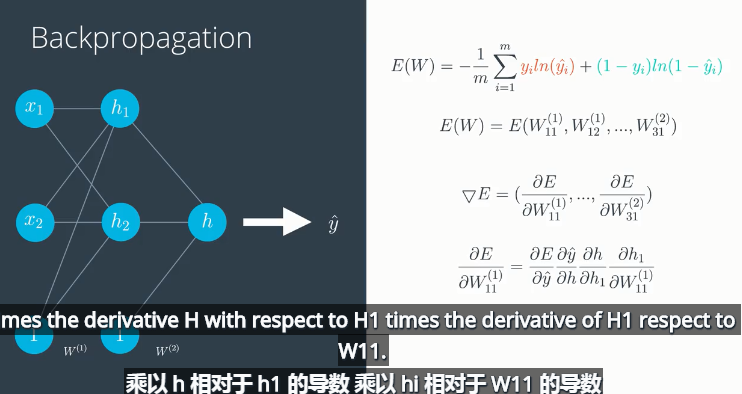
对于单层感知器：



对于多层感知器：



通过求取偏导数既可以更新其值



求误差函数对w(k)ij的偏导需要采用复合函数求导的方法得到偏导的累积形式

例如：求最后一层对h1的偏导

