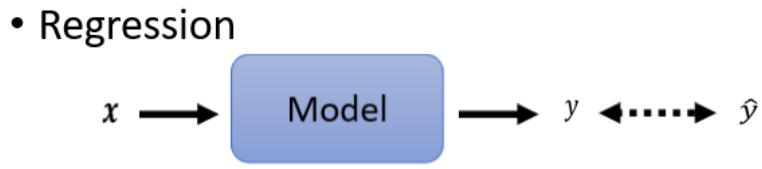
7月14日

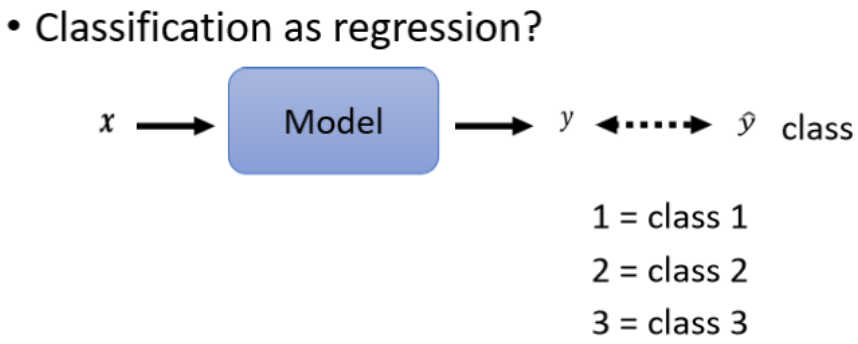
一、李宏毅2021春机器学习课程第2.5节：Classification

1 Classification as Regression?

我们已经知道，**Regression就是输入一个向量，然后输出一个数值**，我们希望输出的数值跟某一个label，也就是我们要学习的目标，越接近越好。

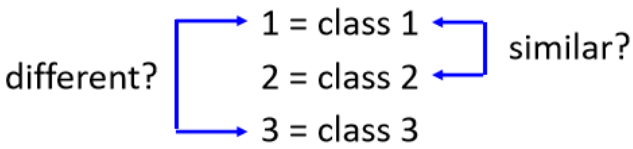


现在问题来到了如何做Classification，那一个很自然的想法就是：**可不可以用Regression来处理Classification的问题呢?**



有这样一个非常naive的想法，那就是假设我们用Regression的方法后输出的值比较接近1，就说明是Class1，比较接近2，就说明是Class2，以此类推。这样就可以用Regression的方法完成Classification的任务。

但是这会是一个好方法吗，如果你仔细想想的话，答案可能是否定的。



因为如果你假设说Class one就是编号1，Class two就是编号2，Class3就是编号3，意味着你觉得**Class1跟Class2比较像**，然后**Class1跟Class3它是比较不像。**假如真的是有这种关系的话，比如Class1 2 3是一年级，二年级，三年级，那可能一年级真的跟二年级关系比较接近，而跟三年级比较远；但如果你的三个Class本身**并没有什么特定的关系**，那使用这种方法就**可能带来一些意外的结果。**

所有对于这种互相之间没有关系的Class进行区分，我们可以使用**one-hot**编码的方法。

2 Class as one-hot vector

> One-Hot 编码，又称为一位有效编码，主要是采用N位状态寄存器来对N个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。

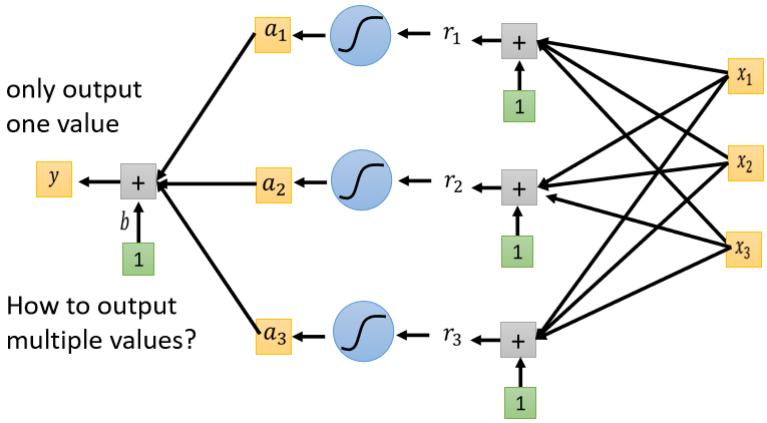
Class 1 Class 2 Class 3

$\hat{y} = \begin{bmatrix}1\\0\\0\end{bmatrix}$ or $\begin{bmatrix}0\\1\\0\end{bmatrix}$ or $\begin{bmatrix}0\\0\\1\end{bmatrix}$

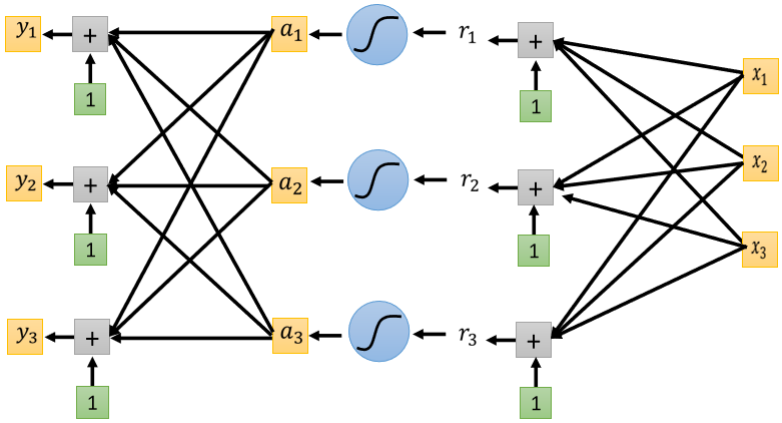
如果有三个Class，我们把Class1编码为 $\begin{bmatrix}1\\0\\0\end{bmatrix}$，如果是Class2编码为 $\begin{bmatrix}0\\1\\0\end{bmatrix}$，如果是Class3编码为 $\begin{bmatrix}0\\0\\1\end{bmatrix}$，每一个Class，都用一个 **One-hot vector** 来表示。

使用One-hot vector的表示方式，就可以看到**这些Class两两之间的的距离都是一样的**，这就保证了这些Class之间并没有相关性，

如果我们今天的目标y hat是一个向量 比如说,ŷ是有三个element的向量，那我们的network,也应该要Output的维度也是三个数字才行



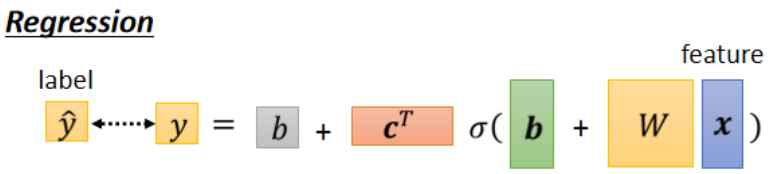
到目前为止我们所讲的network，其实都只Output一个数值，因为我们**过去做的都是Regression的问题，所以只Output一个数字**，但其实**从一个数值改到三个数值，本质上是没有什么不同的**，你可以Output一个数值，那么把本来Output一个数值的方法重复三次，你就可以Output三个数值。



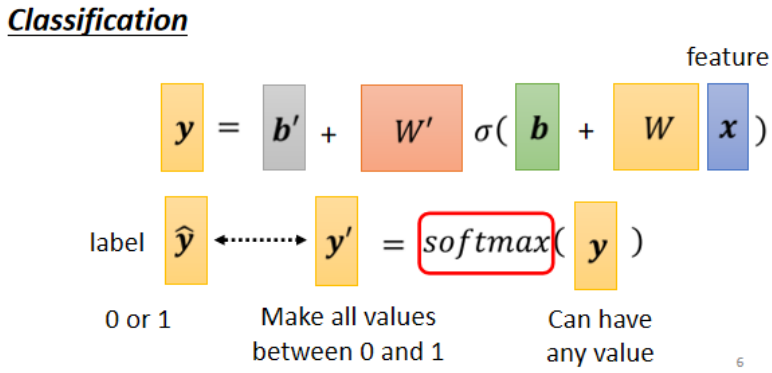
所以你就可以Input一个feature的Vector，然后产生y_1，y_2，y_3，然后希望 $\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}$ 跟我们的目标类的向量表示越接近越好（就比如 $\begin{bmatrix}1\\0\\0\end{bmatrix}$）。

3 Classification with softmax

在Regression的过程中，我们输入一个 x，输出一个 y，希望这个 y 要跟 label ŷ 越接近越好：



那在Classification的过程中，输入一个 x，输出一个 y，这个 y **现在不是一个数值，而是一个向量**，再和目标向量比较之前，我们往往还会**把y再通过一个叫做Soft-max的function得到y'**，然后我们才去计算y' 和目标向量之间的距离。

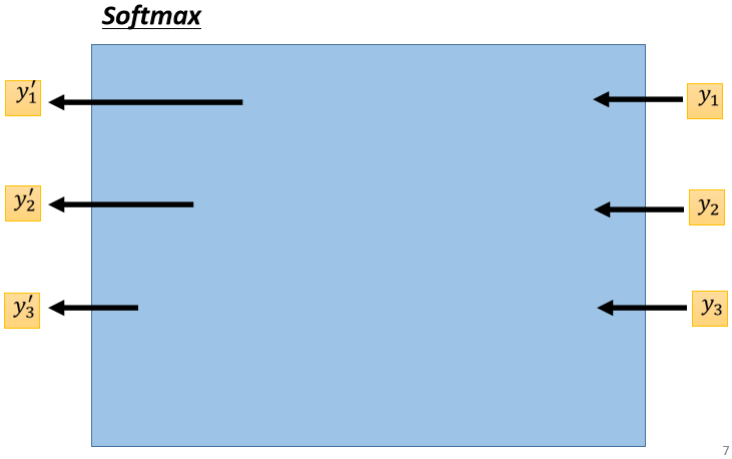


那为什么要加上**Soft-max**呢，一个比较简单的解释是这样的，由于这个目标向量是这个 ŷ 是One-hot vector，所有它里面的值都是0或1，但是我们的 y 却可能是任何的值，直接算距离显然不太对。

那既然我们的目标只有0跟1，我们把 y 也Normalize到**0到1之间**，这样才好跟 label 计算相似度。

3.1 Softmax

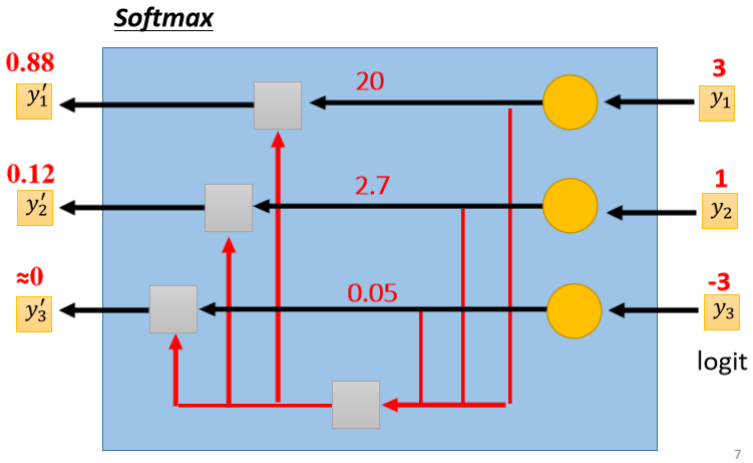
这个是Soft-max的block，输入 y_1 y_2 y_3，它会产生 y'_1 y'_2 y'_3：



它里面运作的模式是这个样子的：

$$y'_i = \frac{exp(y_i)}{\sum_j exp(y_i)}$$

举一个简单的例子来看可能更加直观：

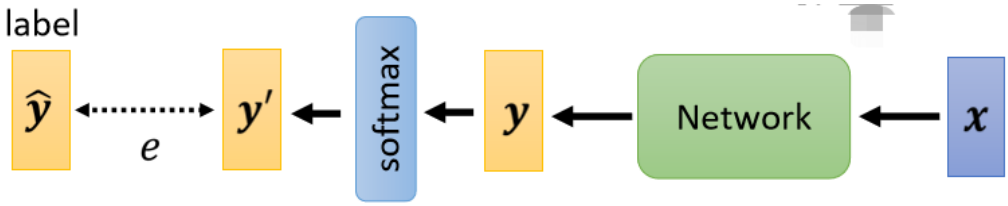


原本输入的 $y_1 = 3$ $y_2 = 1$ $y_3 = -3$，最后输出了 $y_1' = 0.88$ $y_2' = 0.12$ $y_3' \approx 0$。

可以看到这个Softmax可以把**输出全部变为变成0到1之间，并且他们的和为1**，除此之后，它还有一个附带的效果：**它会让大的值跟小的值的差距更大**，就比如原本输入的-3，最后得到的输出趋近于0。

---

4 Loss of Classification



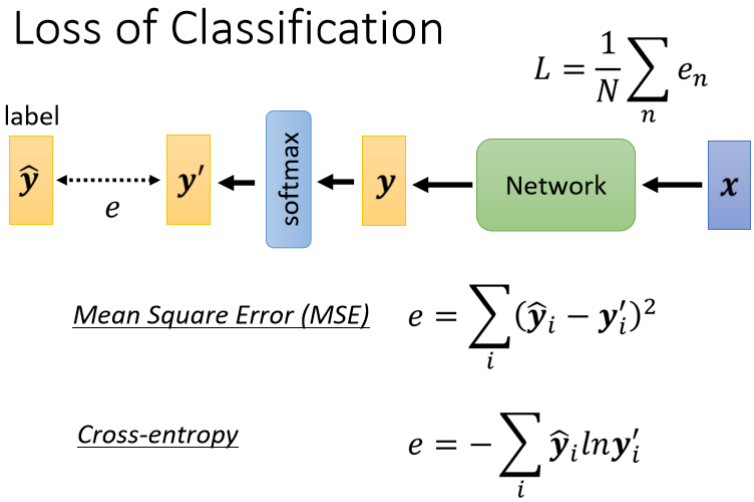
最终我们需要计算 y' 跟 ŷ 之间的距离来评判分类器做出的判断是否准确，而对于这个所谓的距离，其实有许多定义，举例来说，我们可以让这个距离是Mean Square Error（MSE）

$$e = \sum_i (\hat{y}_i - y_i')^2$$

但在分类问题中一个更常用的定义叫做**Cross-entropy**：

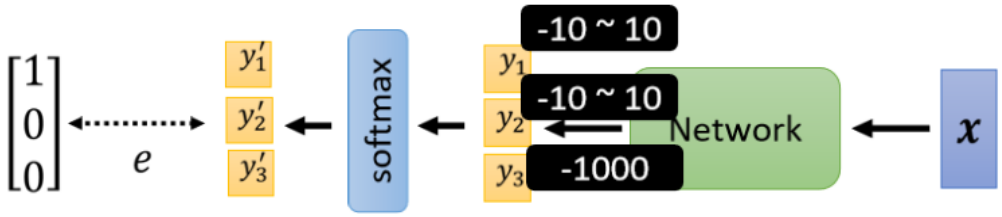
$$e = -\sum_i \hat{y}_i \ln y_i'$$

这个Cross-entropy的式子看起来感觉有点匪夷所思，为什么有这么奇怪的式子出现呢?



Minimizing cross-entropy is equivalent to **maximizing likelihood**.

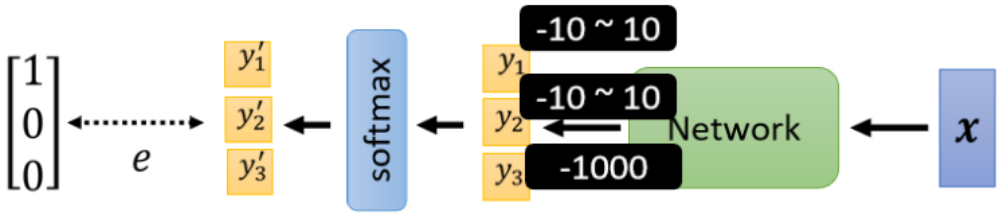
事实上，**Make Minimize Cross-entropy 其实就是 maximize likelihood**，这是从概率论和数理统计上可以证明的。接下来还是通过举例子的方式，**从optimization的角度，来说明相对于Mean Square Error，Cross-entropy为什么更常用在分类问题上。**

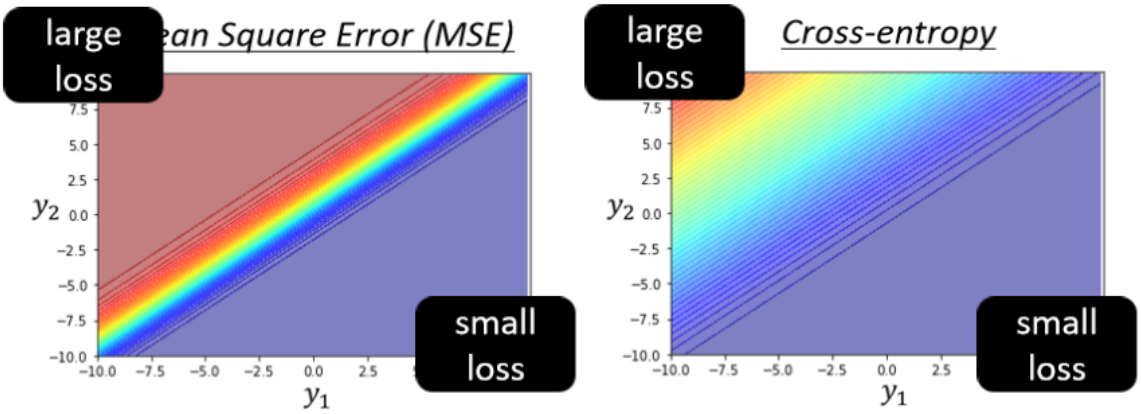


假设现在我们要做一个**3个Class的分类**

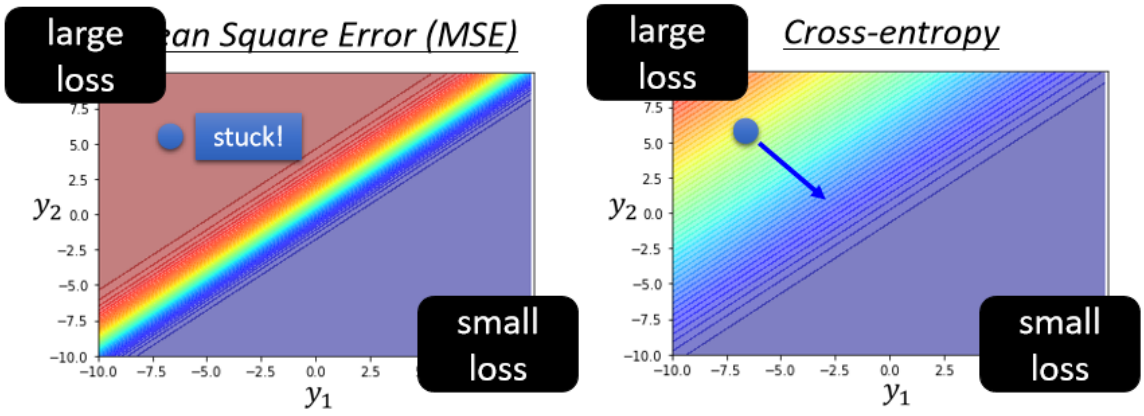
我们现在假设y_1的变化是从-10到10，y_2的变化也是从-10到10，y_3我们就固定设成-1000，因为y_3是一个常量，我们只看y_1跟y_2有变化的时候，对我们的Loss有怎样的影响。

左下方的图是使用 Mean square error 情况下画出的Error surface，右下方的图是使用 Cross-entropy 的时候画出来的Error surface：





我们用**红色表示Loss大，蓝色表示Loss小**，可以看到这两个图都是**左上角Loss大，右下角Loss小**，因为右下角代表我们的y_1接近1，而y_2接近0，这与我们的目标label是最接近的。假设**我们的训练都是从左上角开始的。**



Changing the loss function can change the difficulty of optimization.

- 如果我们选择 Cross-Entropy，可以看到左上角的地方是有斜率的，所以我们可以通过 Gradient Descent 的方法慢慢往右下的地方走，最终走到Loss很小的地方。

- 但如果我们选择的是Mean square error的话，很可能刚开始就卡住了，Mean square error 在这种 Loss 很大的地方是非常平坦的，它的**gradient非常小**，我们可能就没有办法用 Gradient Descentt 顺利的走到右下角。

所以说**在 classification 的问题中，更常见的是选择 Cross-Entropy 作为我们的 Loss Function。**