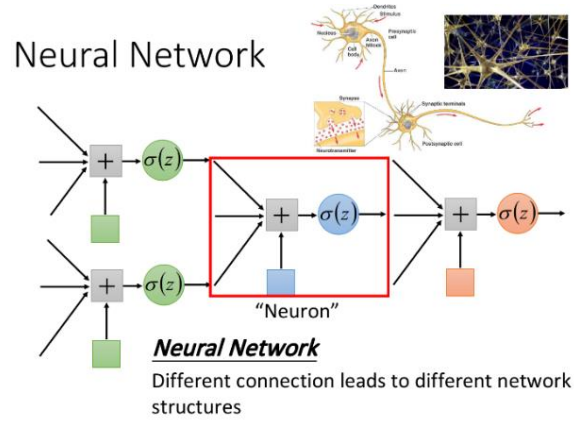
Deep Learning

Neural Network

把多个 Logistic Regression 前后 connect 在一起，然后把一个 Logistic Regression 称之为 neuron，整个称之为 neural network



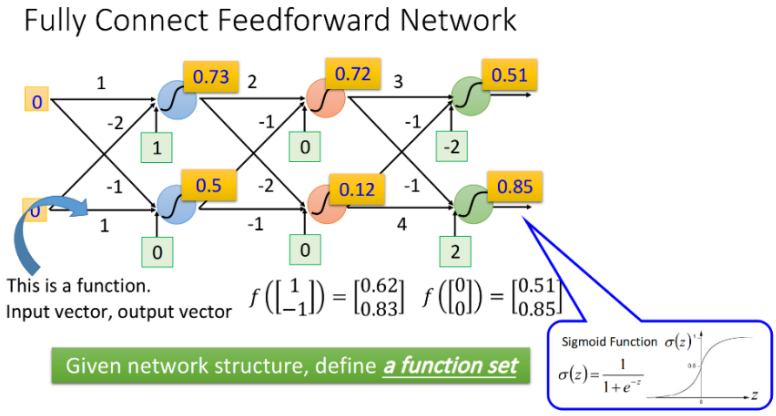
Network parameter θ: all the weights and biases in the "neurons"

我们可以用不同的方法连接这些 neuron，就可以得到不同的 structure，neural network 里的每一个 Logistic Regression 都有自己的 weight 和 bias，这些 weight 和 bias 集合起来，就是这个 network 的 parameter，我们用θ来描述

Fully Connect Feedforward Network

那该怎么把它们连接起来？这需要你手动去设计的，最常见的连接方式叫做 Fully Connect Feedforward Network(全连接前馈网络)

如果一个 neural network 的参数 weights 和 bias 已知的话，它就是一个 function，它的 input 是一个 vector，output 是另一个 vector，这个 vector 里面放的是样本点的 feature，vector 的 dimension 就是 feature 的个数



如果今天我们还不知道参数，只是定出了这个 network 的 structure，只是决定好这些 neuron 该怎么连接在一起，这样的一个 network structure 其实就是 define 了一个 function set(model),我们给这个 network 设不同的参数，它就变成了不同的 function，把这些可能的 function 集合起来，我们就得到了一个 function set

只不过我们利用 neural network 决定 function set 的时候，这个 function set 是比较大的，它包含了很多原来得到的 Logistic regression、做 linear Regression 中无法包含的 function

下图中，每一排表示一个 layer，每个 layer 里面的每一个球都代表一个 neuron

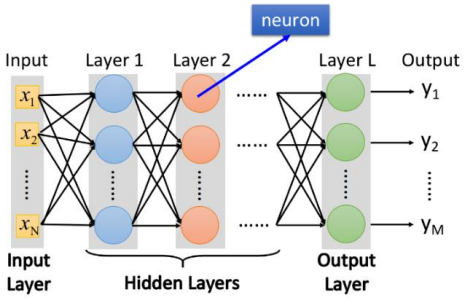
- Layer 和 layer 之间 neuron 是两两互相连接的，layer 1 的 neuron output 会连接给 layer 2 的每一个 neuron 作为 input
- 对整个 neural network 来说，它需要一个 input，这个 input 就是一个 feature 的 vector，而对 layer1 的每一个 neuron 来说，它的 input 就是 input layer 的每一个 dimension
- Last layer，由于它后面没有接其它东西了，所以它的 output 就是整个 network 的 output
- 这里每一个 layer 都是有名字的

  Input 的地方，叫做 input layer，输入层(严格来说 input layer 其实不是一个 layer，它跟其他 layer 不一样，不是有 neuron 所组成)

Output 的地方，叫做 output layer，输出层

其余的地方，叫做 hidden layer，隐藏层

- 每一个 neuron 里面的 sigmoid function，在 Deep Learning 中被称为 activation function（激活函数），事实上它不见得一定是 sigmoid function，还可以是其他 function(sigmoid function 是从 Logistic regression 迁移过来的，现在已经较少在 Deep learning 里使用了)
- 有很多层 layers 的 neural network，被称为 DNN(Deep Neural Network)

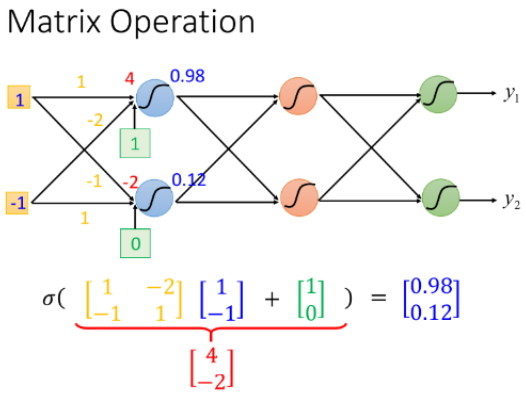


因为 layer 和 layer 之间，所有的 neuron 都是两两连接，所以它叫 Fully connected 的 network；因为现在传递的方向是从 layer1->2->3，由后往前传，所以它叫做 Feedforward network

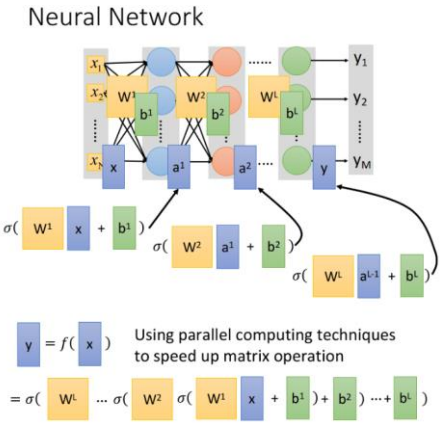
Matrix Operation

Network 的运作过程，我们通常会用 Matrix operation 来表示，以下图为例，假设第一层 hidden layers 的两个 neuron，它们的 weight 分别是$w_1 = 1, w_2 = -2, w_1' = -1, w_2' = 1$,那就可以把它们排成一个 matrix:$\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}$，而我们的 input 又是一个 2*1 的 vector：$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$，将$w$和$x$相乘，再加上 bias 的 vector：$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$，就可以得到这一层的 vector z，在经过 activation function 得到这一层的 output

$$\sigma\left(\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \sigma\left(\begin{bmatrix} 4 \\ -2 \end{bmatrix}\right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$



这里我们把所有的变量都以 matrix 的形式表示出来，注意W^i的 matrix，每一个元素对应的是一个 neuron 的 weight，行数就是 neuron 的个数，而 input x，bias b和 output y都是一个列向量，列数就是 feature 的个数(也是 neuron 的个数，neuron 的本质就是把 feature transform 到另一个 space)

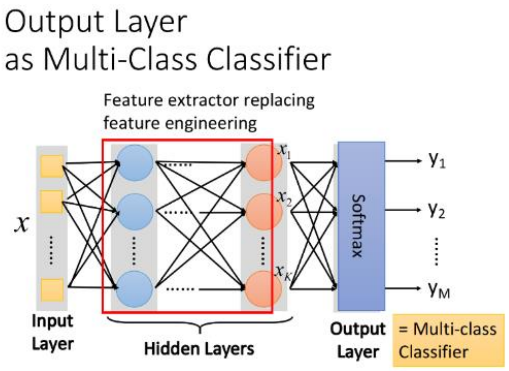


把这件事情写成矩阵运算的好处是，可以用 GPU 加速，GPU 对 matrix 的运算比 CPU 要来的快的，所以我们写 neural network 的时候，习惯把它写成 matrix operation，然后 call GPU 来加速它

Output Layer

我们可以把 hidden layers 这部分，看做是一个 feature extractor(特征提取器)，这个 feature extractor 就 replace 我们之前手动做 feature engineering，feature transform 这些事情，经过这个 feature extractor 得到的$x_1, x_2, \cdots\cdots, x_k$就可以当作一组新的 feature

Output layer 做的事情，其实就是把它当做一个 Multi-class classifier，它是拿经过 feature extractor 转换后的那一组比较好的 feature(能够很好地 separate)进行分类，由于我们把 output layer 看做是一个 Multi-class classifier，所以我们会在最后一个 layer 加上 softmax



Design network structure vs feature engineering

其实 network structure 的 design 是一件蛮难的事情，我们到底要怎么决定 layer 的数目和每一个 layer 的 neuron 的数目呢？其实这个只能够凭经验和直觉、多方面的尝试，有时候甚至会需要一些 domain knowledge(专业领域的知识)，从非 deep learning 的方法到 deep learning 的方法，并不是说 machine learning 比较简单，而是我们把一个问题转化成了另一个问题

本来不是 deep learning 的 model，要得到一个好结果，往往需要做 feature engineering(特征工程)，也就是做 feature transform，然后找一组好的 feature，一开始学习 deep learning 的时候，好像会觉得 deep learning 的的 layers 之间也是在做 feature transform，但实际上在做 deep learning 的时候，往往不需要一个好的 feature，比如说在做图像识别的时候，你可以把所有的 pixel 直接丢进去，但是在过去做图像识别，你是需要对图像抽取出一些人定的 feature 出来的，这件事情就是 feature transform，但是有了 deep learning 之后，你完全可以直接丢 pixel 进去硬做但是，今天 deep learning 制造了一个新的问题，它所制造的问题就是，你需要去 design network 的 structure，所以你的问题从本来的如何抽取 feature 转化城怎么 design structure，所以 deep learning 是不是真的好用，取决于你觉得那一个问题比较容易

如果是图像识别或语音识别的话，design network structure 可能比 feature engineering 要来的容易；因为，虽然我们人都会看、会听、但这件事情，它太过潜意识了，它离我们意识的层次太远，我们无法意识到，我们到底是怎么做语音识别这件事情，所以对人来说，你要抽一组好的 feature，让机器可以很方便地用 linear 的方法做语音识别，其实是很难的，因为人根本不知道好的 feature 到底长什么样子，所以还不如 design 一个 network structure，或者是尝试各种 network structure，让 machine 自己找出好的 feature，这件事情反而变得比较容易，对图像识别也是如此。