4 Task04开始啦：
Task04：参照开源文档，观看视频 P13：深度学习介绍和反向传播机制（2天）；开源文档：https://datawhalechina.github.io/leeml-notes；视频地址：https://www.bilibili.com/video/BV1Ht411g7Ef

Task 04 深度学习介绍和反向传播机制

深度学习的三个步骤

- Step1：神经网络（Neural network）
- Step2：模型评估（Goodness of function）
- Step3：选择最优函数（Pick best function）

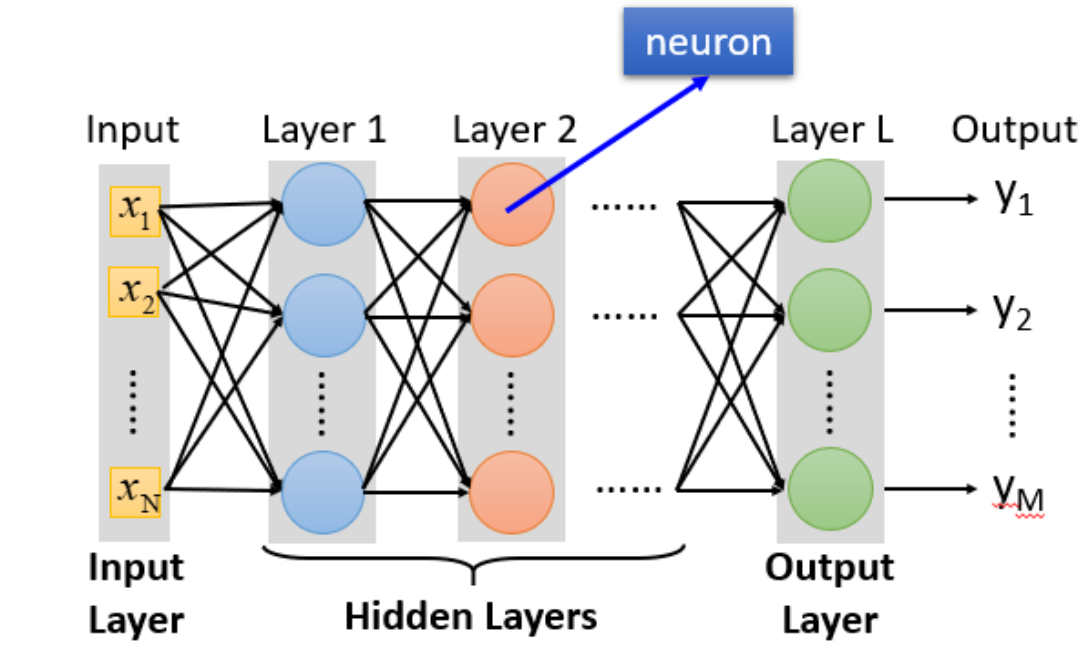
Step1：神经网络

- 神经网络（Neural network）里面的节点，类似我们的神经元。
- 神经网络也可以有很多不同的连接方式，这样就会产生不同的结构（structure）在这个神经网络里面，我们有很多逻辑回归函数，其中每个逻辑回归都有自己的权重和自己的偏差，这些权重和偏差就是参数。
- 那这些神经元都是通过什么方式连接的呢？其实连接方式都是你手动去设计的。

1.1 完全连接前馈神经网络

1.1.1 全链接和前馈的理解

- 输入层（Input Layer）：1层
- 隐藏层（Hidden Layer）：N层
- 输出层（Output Layer）：1层



- 为什么叫全链接呢?
  - 因为layer1与layer2之间两两都有连接，所以叫做Fully Connect；
- 为什么叫前馈呢?
  - 因为现在传递的方向是由后往前传，所以叫做Feedforward。

1.1.2 深度的理解

那什么叫做Deep呢？Deep = Many hidden layer。那到底可以有几层呢？这个就很难说了。

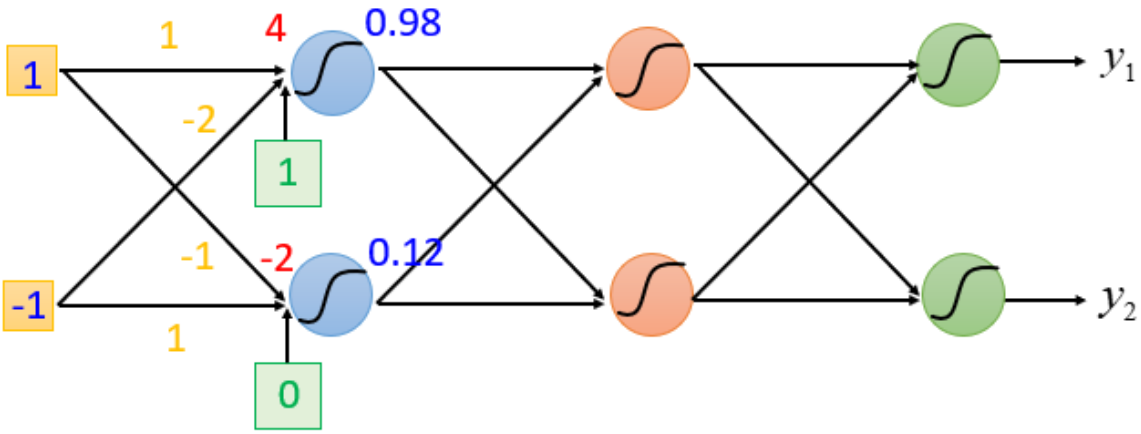
随着层数变多，错误率降低，随之运算量增大，通常都是超过亿万级的计算。对于这样复杂的结构，我们一定不会一个一个的计算，对于亿万级的计算，使用loop循环效率很低。

这里我们就引入**矩阵计算（Matrix Operation）** 能使得我们的运算的速度以及效率高很多：

1.2 矩阵计算

计算方法就是：sigmoid（权重w【黄色】 * 输入【蓝色】 + 偏移量b【绿色】） = 输出

Matrix Operation



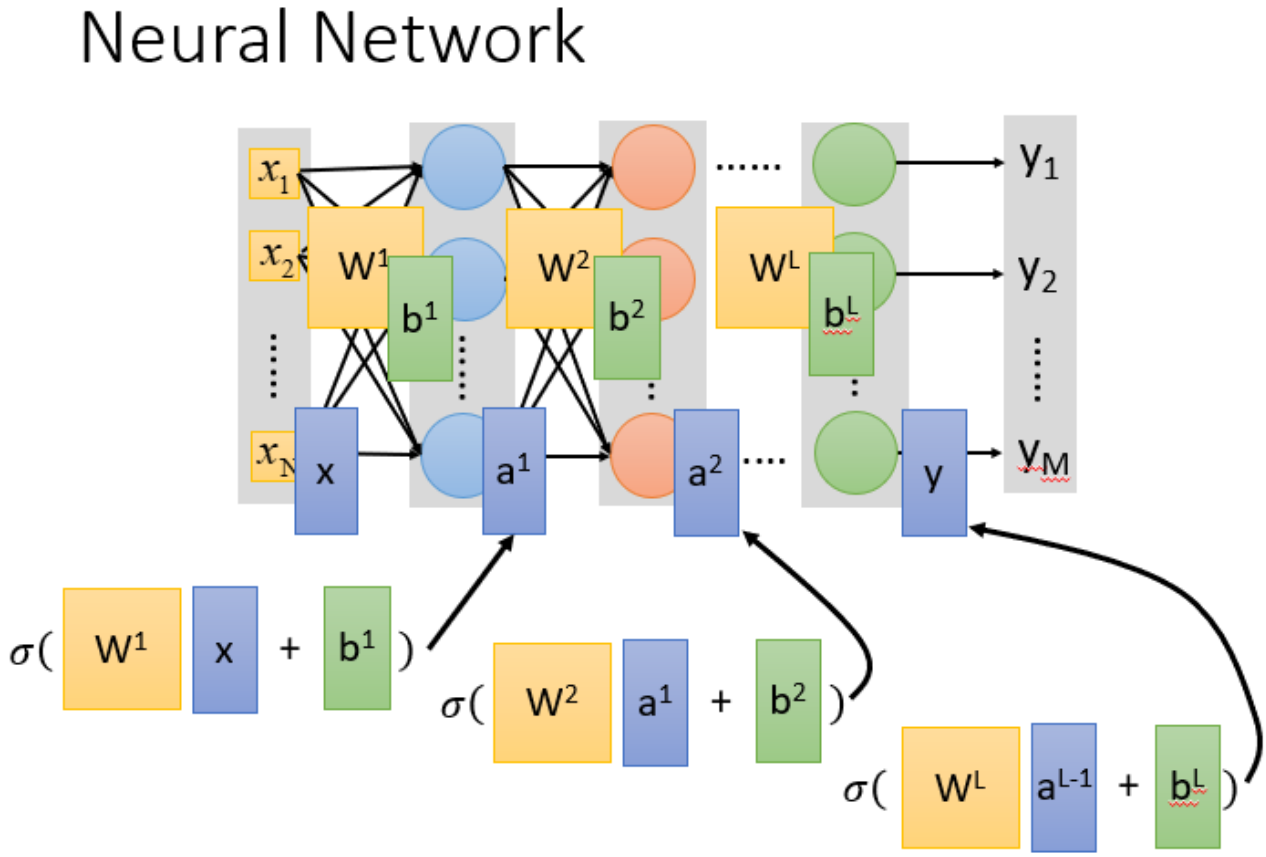
$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

$$\begin{bmatrix}4\\ -2\end{bmatrix}$$

其中sigmoid更一般的来说是激活函数(activation function)，现在已经很少用sigmoid来当做激活函数。

如果有很多层呢?

$$a^1=\sigma(w^1x+b^1)a^2=\sigma(w^1a^1+b^2)\cdots y=\sigma(w^La^{L-1}+b^L)$$



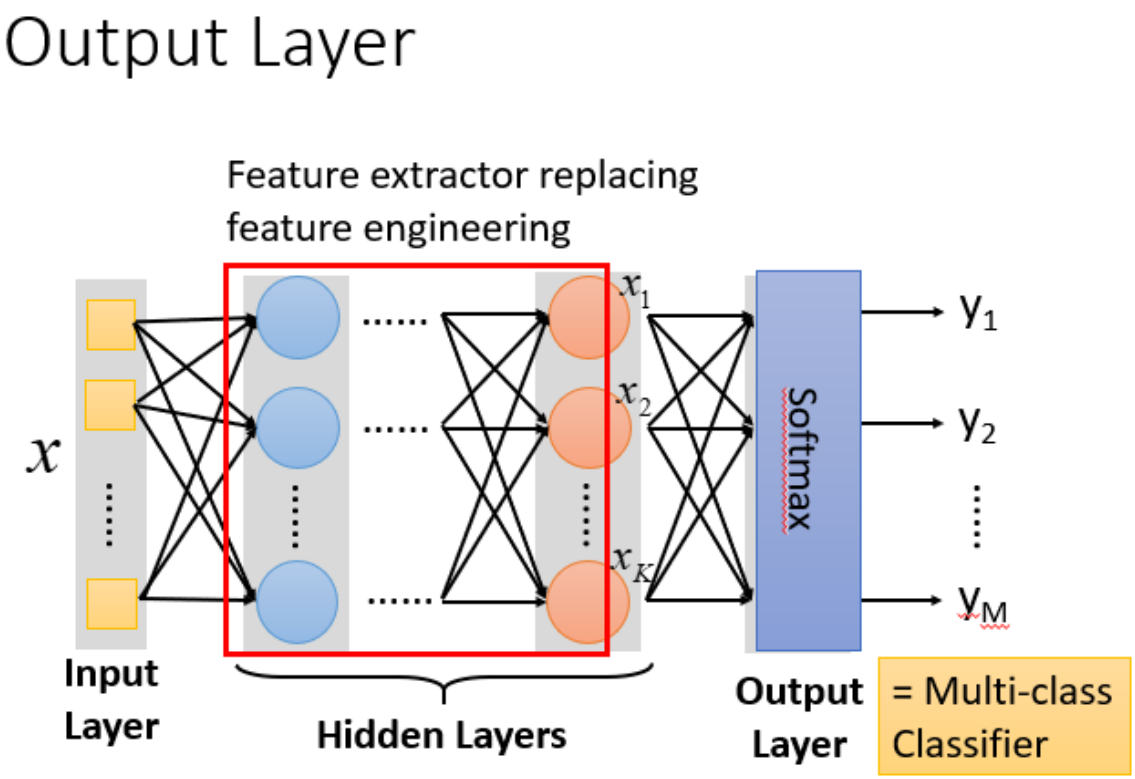
计算方法就像是嵌套，结合上一个图更好理解。所以整个神经网络运算就相当于一连串的矩阵运算。

$$\sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

从结构上看每一层的计算都是一样的，也就是用计算机进行**并行矩阵运算**。
这样写成矩阵运算的好处是，你可以使用**GPU**加速。
整个神经网络可以这样看：

1.3 本质：通过隐藏层进行特征转换

把隐藏层通过特征提取来替代原来的特征工程，这样在最后一个隐藏层输出的就是一组新的特征（相当于黑箱操作）而对于输出层，其实是把前面的隐藏层的输出当做输入（经过特征提取得到的一组最好的特征）然后通过一个多分类器（可以是softmax函数）得到最后的输出y。

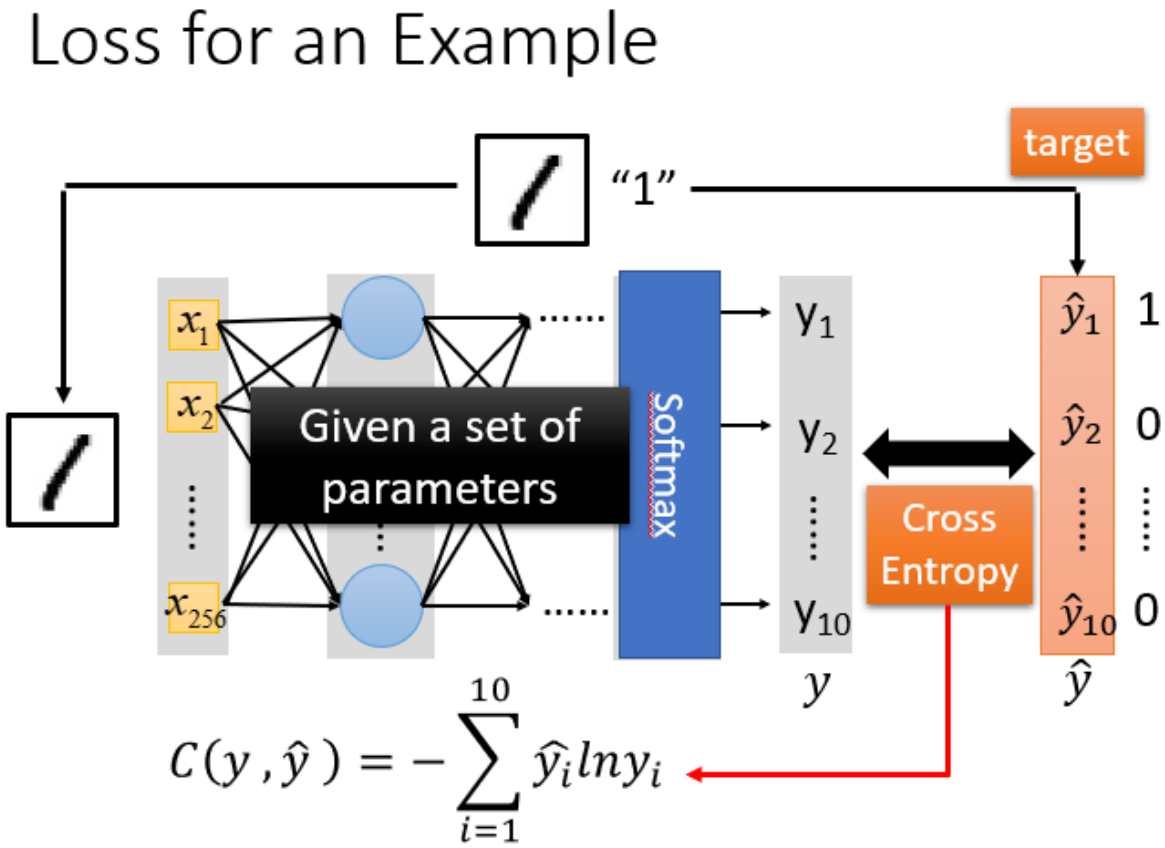


1.4 接下来有几个问题：

- 多少层？ 每层有多少神经元?
  这个问我们需要用尝试加上直觉的方法来进行调试。对于有些机器学习相关的问题，我们一般用特征工程来提取特征，但是**对于深度学习，我们只需要设计神经网络模型来进行就可以了**。对于语音识别和影像识别，深度学习是个好的方法，因为特征工程提取特征并不容易。
- 结构可以自动确定吗?
  有很多设计方法可以让机器自动找到神经网络的结构的，比如进化人工神经网络（Evolutionary Artificial Neural Networks）但是这些方法并不是很普及 。
- 我们可以设计网络结构吗?
  可以的，比如 CNN卷积神经网络（Convolutional Neural Network ）

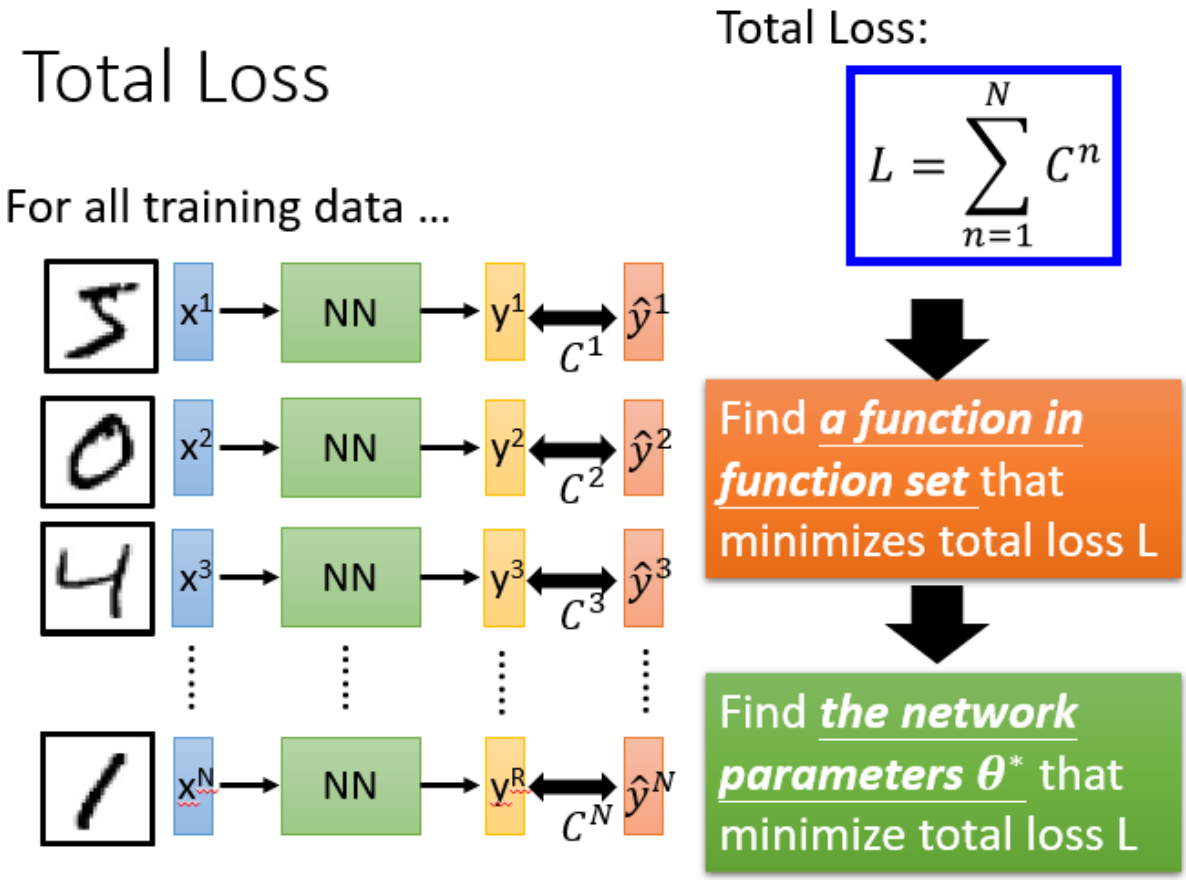
Step2: 模型评估

2.1 损失示例



对于模型的评估，我们一般采用损失函数来反应模型的好差，所以对于神经网络来说，我们采用**交叉熵（cross entropy）函数**来对y和$\hat{y}$的损失进行计算，接下来我们就是调整参数，让交叉熵越小越好。

2.2 总体损失



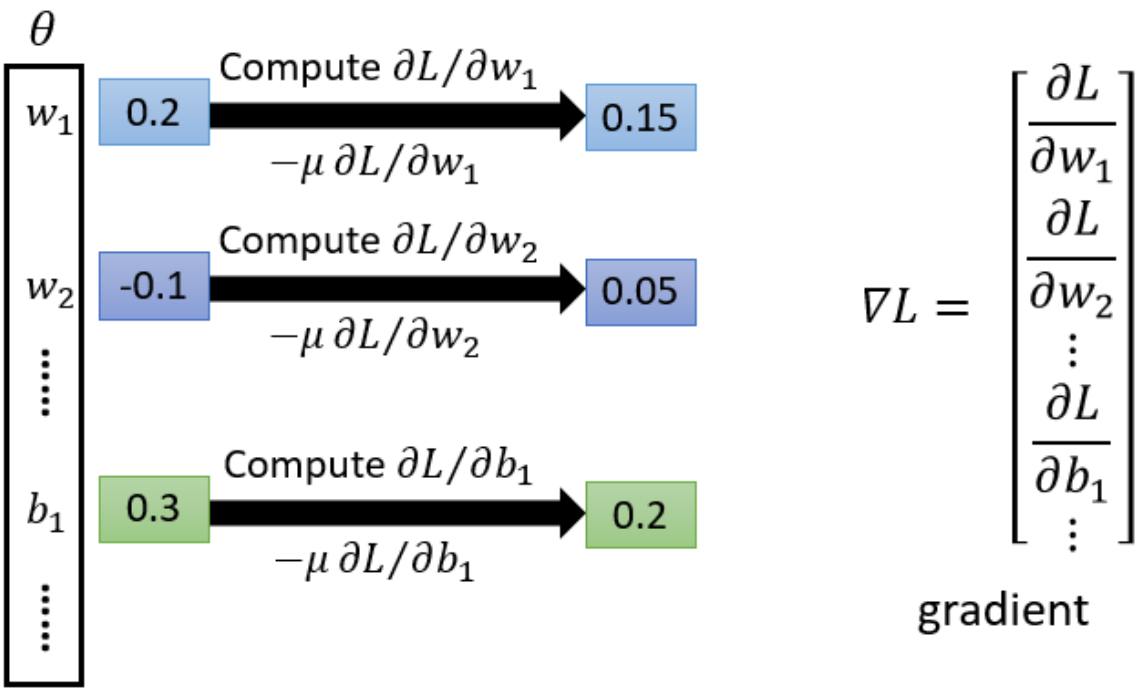
对于损失，我们不单单要计算一笔数据的，而是要计算整体所有训练数据的损失，然后把所有的训练数据的损失都加起来，得到一个总体损失L。接下来就是在function set里面找到一组函数能最小化这个总体损失L，或者是找一组神经网络的参数θ，来最小化总体损失L

Step3：选择最优函数

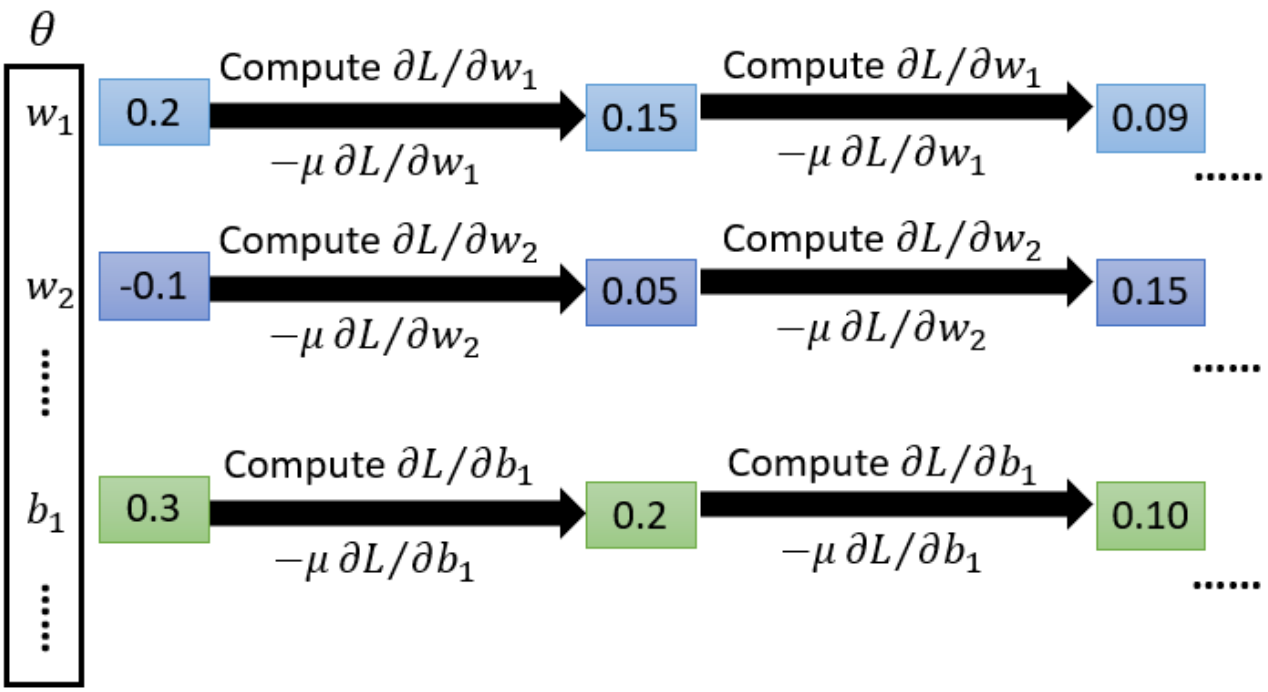
3.1 梯度下降

如何找到最优的函数和最好的一组参数呢，我们用的就是梯度下降（Gradient Descent）。

Gradient Descent



Gradient Descent



具体流程：θ是一组包含权重和偏差的参数集合，随机找一个初试值，接下来计算一下每个参数对应偏微分，得到的一个偏微分的集合∇L就是梯度,有了这些偏微分，我们就可以不断更新梯度得到新的参数，这样不断反复进行，就能得到一组最好的参数使得损失函数的值最小

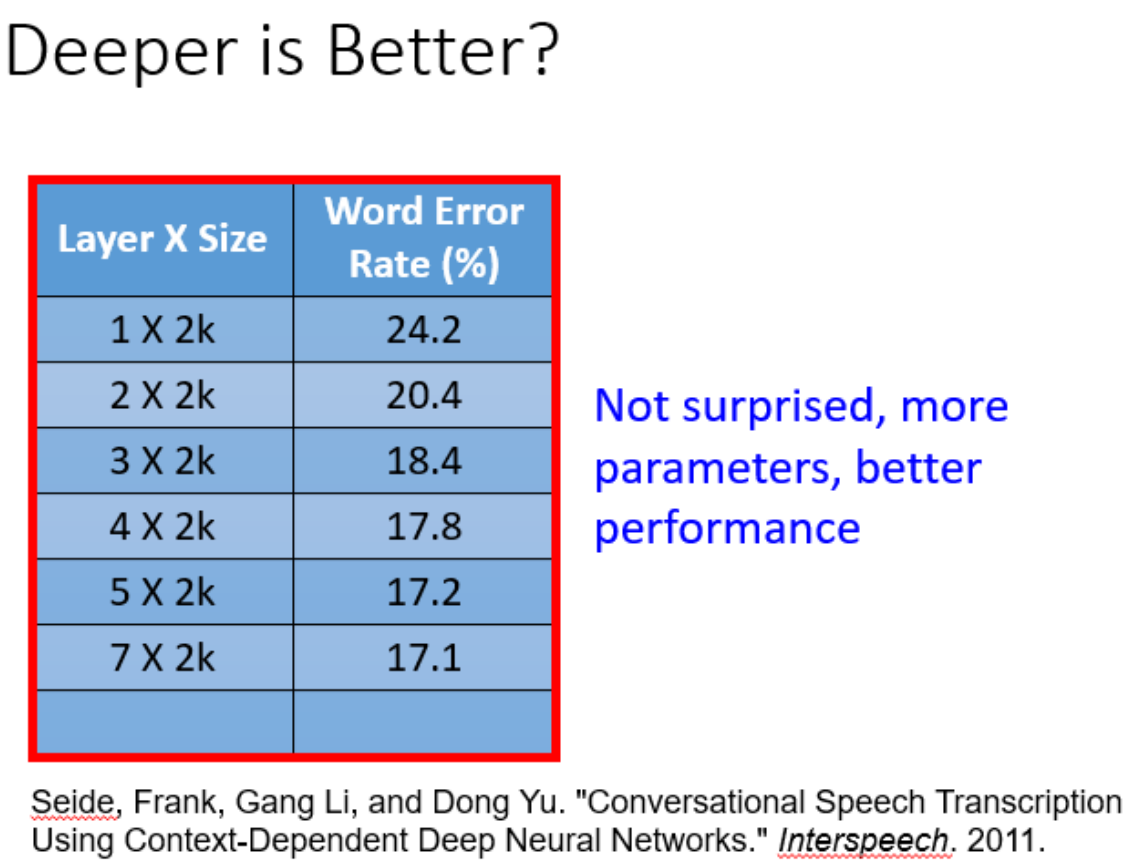
3.2 反向传播

在神经网络中计算损失最好的方法就是反向传播，我们可以用很多框架来进行计算损失，比如说TensorFlow，theano，Pytorch等等

思考

为什么要用深度学习，深层架构带来哪些好处？那是不是隐藏层越多越好?

(1)隐藏层越多越好?



从图中展示的结果看，毫无疑问，层次越深效果越好~~

(2)普遍性定理

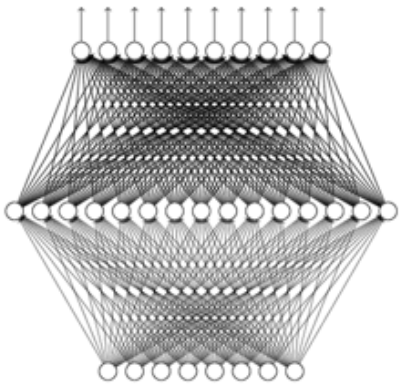
Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^M$$

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)



Reference for the reason:
http://neuralnetworksanddeeplearning.com/chap4.html

Why “Deep” neural network not “Fat” neural network?

(next lecture)

参数多的model拟合数据很好是很正常的。下面有一个通用的理论：

- **对于任何一个连续的函数，都可以用足够多的隐藏层来表示。**

那为什么我们还需要‘深度’学习呢，直接用一层网络表示不就可以了？在接下来的课程我们会仔细讲到。