https://www.coursera.org/learn/machine-learning

https://github.com/coursera-dl/coursera-dl

http://www.cs.cmu.edu/~tom/10701_sp11/lectures.shtml

https://github.com/explosion/spaCy

http://52opencourse.com/174/coursera%E5%85%AC%E5%BC%80%E8%AF%BE%E7%AC%94%E8%AE%B0-%E6%96%AF%E5%9D%A6%E7%A6%8F%E5%A4%A7%E5%AD%A6%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E7%AC%AC%E4%B9%9D%E8%AF%BE-%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E7%9A%84%E5%AD%A6%E4%B9%A0-neural-networks-learning

http://ufldl.stanford.edu/wiki/index.php/%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E5%90%91%E9%87%8F%E5%8C%96

https://aimatters.wordpress.com/2015/12/19/a-simple-neural-network-in-octave-part-1/

https://zhuanlan.zhihu.com/p/21407711

CS231n课程笔记翻译：反向传播笔记

梯度\nabla f是偏导数的向量

梯度[dfdx, dfdy, dfdz]，它们告诉我们函数f（计算最终结果的那个）对于变量[x, y, z]的敏感程度

我们想要的是最终的梯度，初始输入和最终输出之间的关系。

如果输入放大一倍，输出放大多少倍。这个倍数是由导数给出的。如果有多个输入，这个导数就是偏导数。

为了计算最终梯度，每个神经元先计算它自己的所有局部梯度，它自己的每个输入和输出之间的关系。

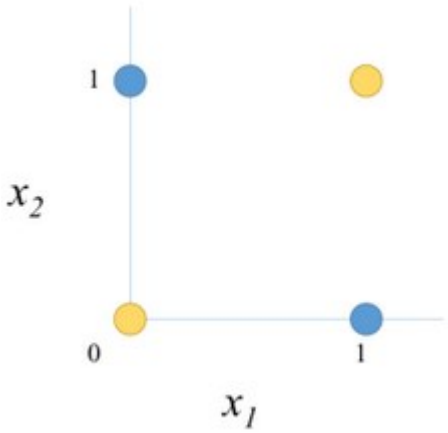
在反向传播的时侯，根据链式法则把回传的递度乘以各个局部梯度，就得上一层的输入和最终输出之间的梯度。

tutorial/InteractiveGraphicsPalette

真值表

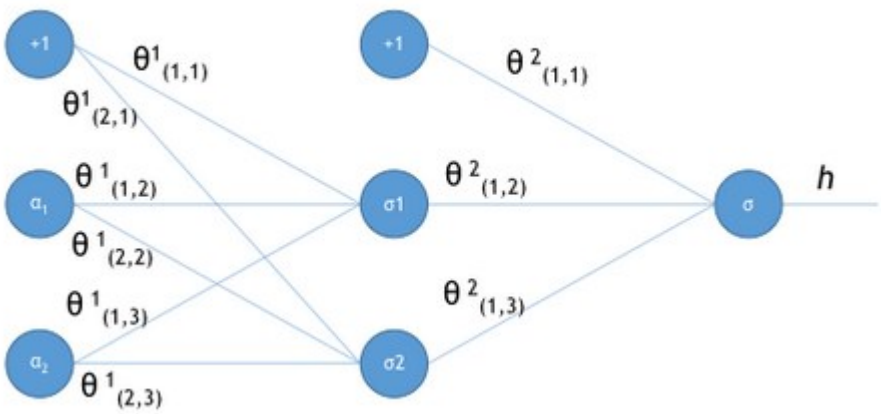
Given this input		Produce this output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

非线性可分



不能用一条直线把蓝色的结果和黄色的结果分开

异或问题的神经网络模型



（第二层和最后一层的计算单元都是一个LogisticSigmoid函数，把输入压缩到-1和+1之间，代表神经元的抑制和兴奋状态）

+1 是偏置，就像电路中的电阻，用来对电流强度进行微调。

theta 是权值，权值是放大器，对信号进行放大。权值乘上输入并传给下一层的计算单元作为它的输入。

输入用列向量表示，当变量很多的时侯，矩阵能简化运算

$$A^1=\begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix}$$

权值用行向量表示，把两行组织在一起形成一个矩阵。第一行是给下一层的某个运算单元的三个权值，第二行是给另一个单元的三个权值。

$$\Theta^1=\begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \theta_{1,3} \\ \theta_{2,1} & \theta_{2,2} & \theta_{2,3} \end{bmatrix}$$

通常，权值的初始值设为-1 到+1 之间的随机数，一开始我们不知道它们的值应该是多少。

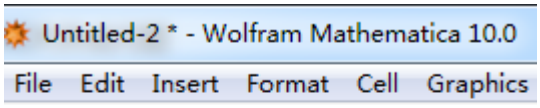
每一个输入乘以它的权值，然后全部加起来，作为下一层计算单元的输入。重复这个过程，直到最后一步，前面所有层生成了一个输入，然后用这个输入计算一个函数，得到最后的结果h，它就是神经网络对结果的预测。预测的结果不准，就看看差了多少，不断的调整权值，这就是对神经网络的训练。

计算h 的函数通常是一个LogisticSigmoid，$\frac{1}{1+e^{-x}}$

现在来计算这一组输入和输出

x_1	x_2	y
0	0	0

别忘了偏置 +1 也是一个输入




```
In[5]:= A1 = {{1}, {0}, {0}};
        A1 // MatrixForm
```

Out[6]//MatrixForm=

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

A1是最左边的三个输入

三个输入，需要三个权值，组成一个行向量。第二层有两个结点，需要两个行向量。

```
In[12]:= θ1 = RandomReal[{-1, 1}, {2, 3}];
         θ1 // MatrixForm
```

Out[13]//MatrixForm=

$$\begin{pmatrix} 0.942509 & -0.462658 & -0.608787 \\ -0.307834 & 0.322166 & -0.257113 \end{pmatrix}$$

每输入乘以它的权值，然后全部加起来作为第二层计算单元的输入。用点乘来表达就是：

```
In[29]:= A2=θ1.A1; A2//MatrixForm
```

Out[29]//MatrixForm=

$$\begin{pmatrix} 0.942509 \\ -0.307834 \end{pmatrix}$$

A2是第二层计算单元的输入

计算第二层的输出，作为第三层的输入。第二层的计算单元是LogisticSigmoid，$\frac{1}{1+e^{-x}}$：

```
In[39]:= A3 = LogisticSigmoid[A2] // Prepend[{1}]; A3 // MatrixForm
```

Out[39]//MatrixForm=

$$\begin{pmatrix} 1 \\ 0.719606 \\ 0.423644 \end{pmatrix}$$

这里Prepend 是把偏置值加进去

A3是第三层的输入

```
In[41]:= θ2 = RandomReal[{-1, 1}, {1, 3}]; θ2 // MatrixForm
```

Out[41]//MatrixForm=

$$\begin{pmatrix} 0.441079 & 0.064766 & -0.915196 \end{pmatrix}$$

theta2 是第三层的权值

```
In[51]:= h = Θ2.A3 // LogisticSigmoid
Out[51]= {{0.685207}}
```

h 是神经网络的最终输出。观察真值表，输入0，0 应该输出0。神经网络的输出值偏高了。

需要调整仅值，怎么调整？梯度会给你指引，它会告诉你把某个输入调大调小，输出会怎么变化。

```
Subscript[A, 1] = {{1},{0}, {0}};
Subscript[A, 1]//MatrixForm
Subscript[\[CapitalTheta], 1] =RandomReal[{-1,1},{2,3}];
Subscript[\[CapitalTheta], 1]//MatrixForm
Subscript[A, 2]=Subscript[\[CapitalTheta], 1].Subscript[A, 1]; A2//MatrixForm
Subscript[A, 3]=LogisticSigmoid[Subscript[A, 2]]//Prepend[{1}];Subscript[A, 3]//MatrixForm
Subscript[\[Theta], 2]=RandomReal[{-1,1},{1,3}];Subscript[\[Theta], 2]//MatrixForm
h = Subscript[\[Theta], 2].Subscript[A, 3]//LogisticSigmoid
```

张量流代码

```
import tensorflow as tf

hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
s = sess.run(hello)
print(s)

A1 = tf.constant([[1.], [0.], [0.]]);
Theta1 = tf.random_uniform(
    [2, 3],
    minval=-1,
    maxval=1,
    dtype=tf.float32,
    seed=None,
    name=None
```

);

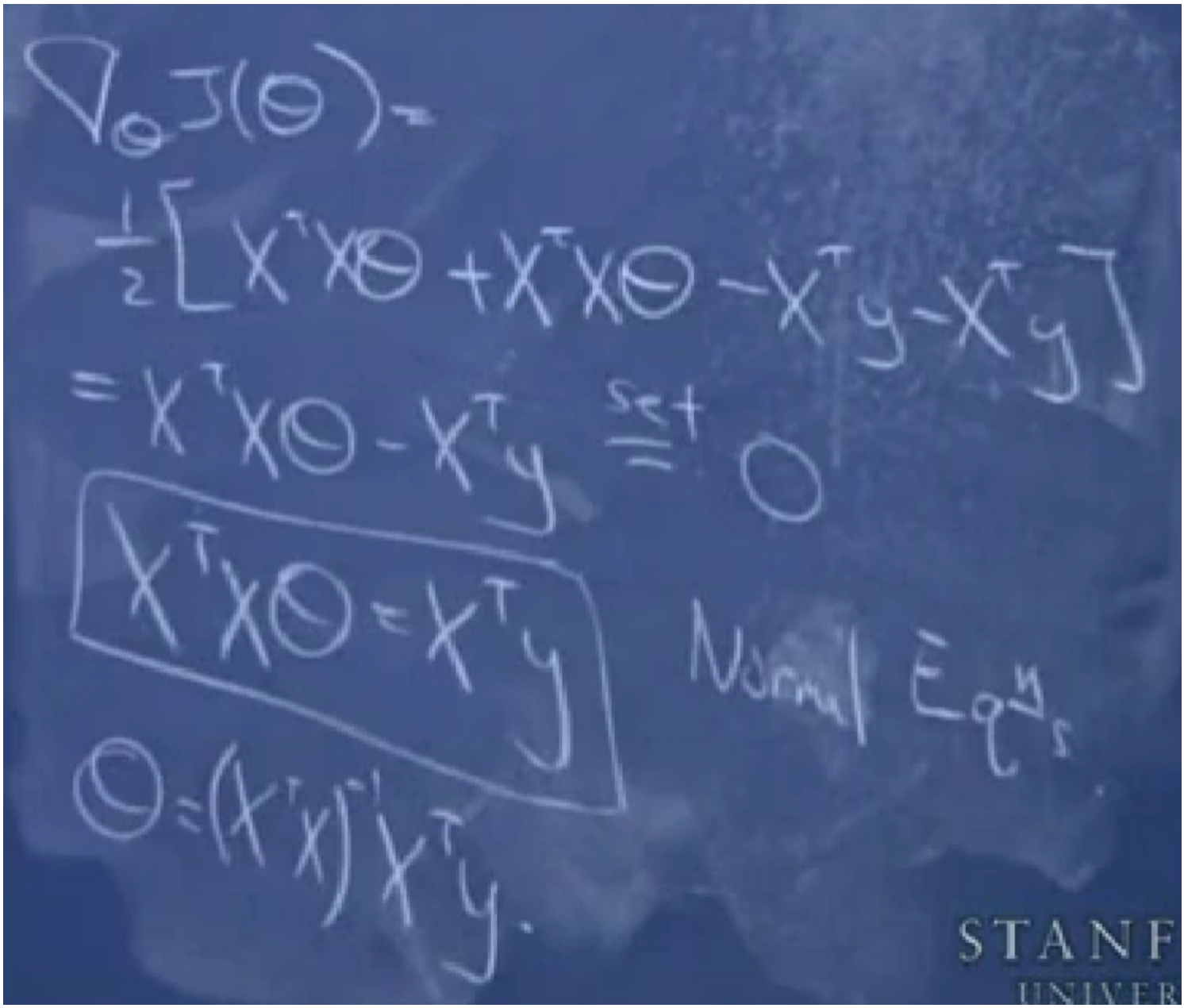
A2 = tf.matmul(Theta1, A1);

with tf.Session() as sess:

  result = sess.run([A1, Theta1, A2])

  print (result)

方程组的解析表达式



随机梯度下降算法，theta 的更新方法 推导过程看 ng的机器学习课程

XOR.nb * - Wolfram Mathematica 10.0

File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

```
X = {{1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}; X // MatrixForm (* 异或的所有输入。前面的1是偏置 *)
```

Out[6]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

```
In[5]:= Y = {{0}, {1}, {1}, {0}}; Y // MatrixForm
```

Out[5]//MatrixForm=

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

```
In[7]:= teta = RandomReal[{-1, 1}, {3, 2}]; teta // MatrixForm
```

Out[7]//MatrixForm=

$$\begin{pmatrix} 0.624269 & 0.682765 \\ -0.452592 & -0.476033 \\ -0.157334 & -0.165554 \end{pmatrix}$$

```
In[36]:= A1 = X.teta // LogisticSigmoid // Map[Prepend[#, 1] &, #] &; A1 // MatrixForm(*加上偏置1。X.teta的每一行都是第二层计算单元的输入。计算单元就是LogisticSigmoid*)
```

Out[36]//MatrixForm=

$$\begin{pmatrix} 1 & 0.651189 & 0.664356 \\ 1 & 0.614658 & 0.626496 \\ 1 & 0.542814 & 0.5515 \\ 1 & 0.503586 & 0.510293 \end{pmatrix}$$

```
In[41]:= teta2 = RandomReal[{-1, 1}, {3, 1}]; teta2 // MatrixForm
```

Out[41]//MatrixForm=

$$\begin{pmatrix} 0.926855 \\ -0.785048 \\ -0.291172 \end{pmatrix}$$

```
In[42]:= A2 = A1.teta2; A2 // MatrixForm(*第三层的输入*)
```

Out[42]//MatrixForm=

$$\begin{pmatrix} 0.222199 \\ 0.261901 \\ 0.340138 \\ 0.382933 \end{pmatrix}$$

```
h = A2 // LogisticSigmoid; h // MatrixForm(*所有的预测结果*)
```

Out[44]//MatrixForm=

$$\begin{pmatrix} 0.555322 \\ 0.565103 \\ 0.584224 \\ 0.59458 \end{pmatrix}$$