C1-5

Machine Learning
by Andrew Ng, Stanford Engineering

Xiaojie Zhou

szxjzhou@163.com

2016.8.22

第六集：朴素贝叶斯算法

- 朴素贝叶斯算法的事件模型(Event Model of Naive Bayes Algorithm)
- 神经网络(Neural Networks)
- 支持向量机(Support Vector Machines)

第六集：朴素贝叶斯算法

- 朴素贝叶斯算法的事件模型
  - 朴素贝叶斯作为一种生成学习算法，可以很好地对于离散特征x进行分类。接下来在之前的基础上对于文本分类做更进一步的讨论
    - 实际上文本分类是一种序列分类的特例，即对于一个输入序列进行分类
  - 在上一节中对于邮件分类问题采取的是设立字典的办法，对于每份邮件以字典中词语是否出现来决定该邮件的特征向量
    - 这样的模型称为多元伯努利事件模型(multi-variate Bernoulli event model)，在该模型中每一维的特征都是{0,1}二元变量
    - 但这种衡量的办法不能很好地表现出文本的特征，因为有些词语可能会出现多次。因此需要对于邮件中的每个词的词频进行衡量，这也就引出了多项式事件模型(multinomial event model)

第六集：朴素贝叶斯算法

- 朴素贝叶斯算法的事件模型
  - 在多项式事件模型中需要对词汇出现的词频进行衡量，对此我们需要对每份邮件对应的特征向量做一些改变
    - 其实我们最容易想到的办法还是设立字典的办法，对于每份邮件以字典中词语出现的次数来决定该邮件的特征向量，但是这样会出现非常严重的数据稀疏问题（比如测试样本中字典中的词i出现了k次，但训练集中的每一封邮件并没有词i出现了k次的情况），会使得学习的效果不佳
    - 因此更好的建模方法是根据邮件内容而不是根据字典进行建模：在这种建模方法中每份邮件对应的特征向量的维数与邮件中词数相同（即当邮件中有n个词时对应的特征向量维数也为n），而每一维指示该邮件中词汇对应的序号
      - 举个例子，假设一份邮件中只有两个词"an email"，"an"是词典中第2个单词，"email"是词典中第2000个单词，那么这份邮件对应的特征向量就是{2, 2000}

第六集：朴素贝叶斯算法

- 朴素贝叶斯算法的事件模型
  - 下面给出更加正式的定义
    - 假设训练集由m个样本构成而每组样本的x的维数与对应邮件的词数相同，因此

$$\text{given a training set } \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\} \text{ where } x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_{n_i}^{(i)}) \text{ (here, } n_i \text{ is the number of words in the } i\text{-training example)}$$

      - 对照之前朴素贝叶斯的参数学习，此时需要学习的参数包括：垃圾邮件占总体样本的概率p(y=1)（记为ϕy），垃圾邮件中字典中第j个词出现的概率p(x=j|y=1)（记为ϕj|y=1），普通邮件中字典中第j个词出现的概率p(x=j|y=0)（记为ϕj|y=0）
      - 通过类似的办法求解极大似然即可得到每个参数的表达方法

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i}$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}.$$

在这里φj|y=1（即p(x=j|y=1)）的含义是所有类标为1中第j个词的出现次数占类标为1的总词数的比值（即所有垃圾邮件中词汇j的频率）；φj|y=0（即p(x=j|y=0)）的含义是所有类标为0中第j个词的出现次数占类标为0的总词数的比值（即所有正常邮件中词汇j的频率）；Φy（即p(y=1)）的含义是所有类标为1的样本个数占样本总数的比值（即垃圾邮件占全部邮件的频率）

第六集：朴素贝叶斯算法

- 朴素贝叶斯算法的事件模型
  - 为避免数据稀疏问题带来的影响，因此需要采用Laplace平滑的方法（其中|V|为词典中词数）

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i + |V|}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i + |V|}.$$

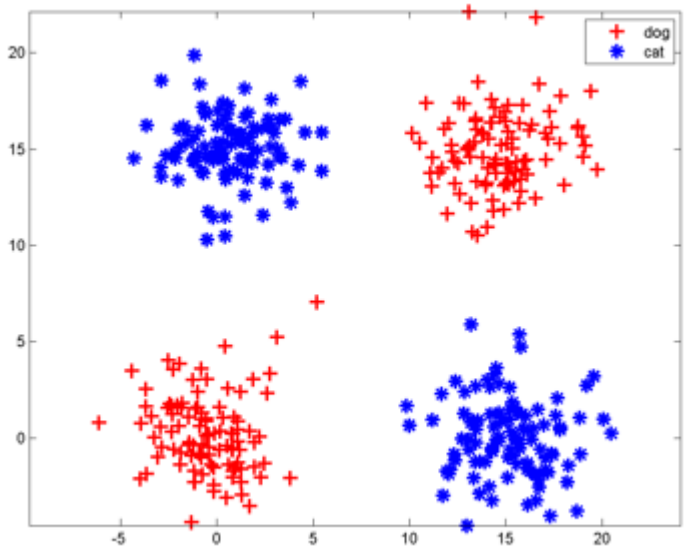
  - 测试方法和之前完全相同

$$\begin{aligned} p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x)} \\ &= \frac{\left(\prod_{i=1}^{n} p(x_i|y=1)\right) p(y=1)}{\left(\prod_{i=1}^{n} p(x_i|y=1)\right) p(y=1) + \left(\prod_{i=1}^{n} p(x_i|y=0)\right) p(y=0)} \end{aligned}$$

和以前一样，不论是p(y=0|x)，还是p(y=1|x)，只要x是给定的，这个式子的分母就是常数。因此可以只计算分子，只比较分子的大小。

第六集：朴素贝叶斯算法

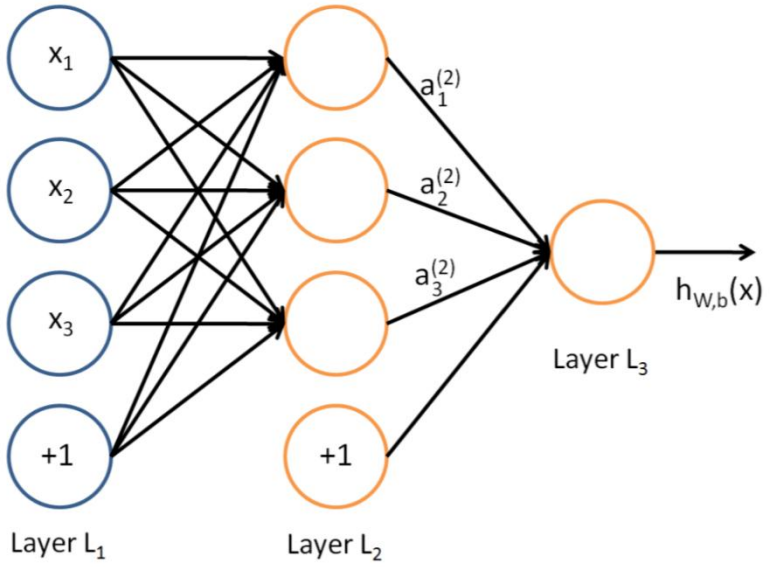
- 神经网络
  - 之前涉及的方法都属于线性分类的方法
    - 之前所述的生成学习方法都可以对应到Logistic回归上，而Logistic回归作为一种标准的由指数分布族中的概率分布所推导出的模型自然属于线性模型（因此称其为广义线性模型）
    - 然而数据不全是线性分类的，比如说下面的例子



不难发现，这个例子很难用一条直线对两个类进行划分。此时我们会想到两种解决办法。一是用多条直线直线进行划分，然后再对于这些区域进行交并运算；二是用高阶曲线进行拟合。前者对应的就是神经网络算法，后者对应的是支撑向量机算法

第六集：朴素贝叶斯算法

- 神经网络
  - 神经网络采取的是如下的结构，至少为输入层、隐藏层和输出层三层
    - 对比Logistic回归，神经网络相当于对多个Logistic回归的结果又进行Logistic回归的运算。从集合的角度看就是使得用多条直线划分的区域间进行交并运算



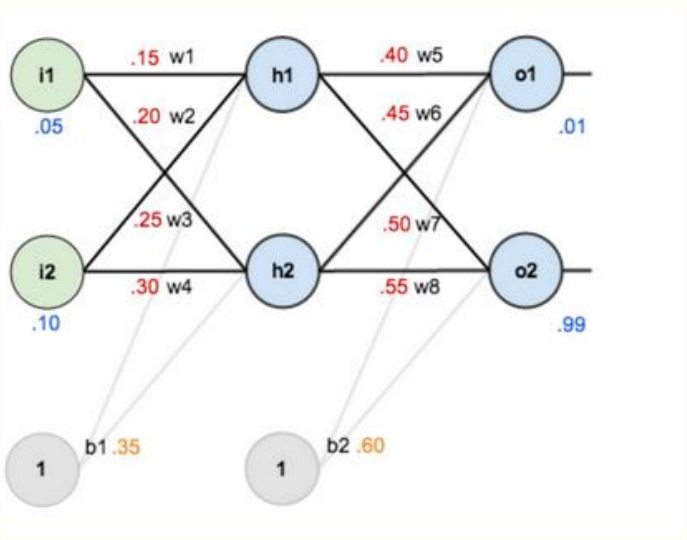
这是神经网络的经典形式，输入层进行某一样本全部特征的输入然后交由隐藏层进行第一步的分类，得到多条划分的直线（从而得到了多个划分区域）。最后再将结果给到输出层再进行分类运算，从而起到集合交并的效果

第六集：朴素贝叶斯算法

- 神经网络
  - 下面来看一个例子，是如何进行神经网络学习的

第一层是输入层，包含两个神经元i1，i2，和截距项b1；第二层是隐含层，包含两个神经元h1,h2和截距项b2，第三层是输出o1,o2，每条线上标的wi是层与层之间连接的权重，激活函数我们默认为sigmoid函数。

现在对他们赋上初值，如下图：



其中，输入数据 i1=0.05，i2=0.10;

输出数据 o1=0.01,o2=0.99;

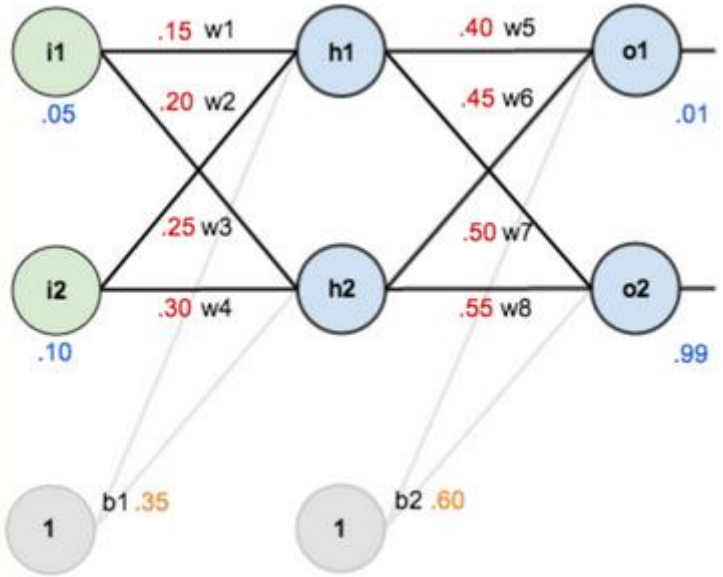
初始权重 w1=0.15,w2=0.20,w3=0.25,w4=0.30;

w5=0.40,w6=0.45,w7=0.50,w8=0.88

由条件可知，在这个例子中h1,h2,o1,o2将上一层参数进行加权求和再加上对应截距项的结果通过Sigmoid函数进行运算，最终产生的值传递给下一层（隐藏层）或者输出（输出层）。一开始需要初始化所有的权重，并通过传入训练集对这些参数进行训练

第六集：朴素贝叶斯算法

- 神经网络
  - 神经网络的学习分为几个阶段，第一阶段为前向传播(forward propagation)，在该阶段中特征x将传入神经网络最终运算出对应的输出y'
    - 首先计算输入层到隐藏层的传播



计算神经元h1的输入加权和：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

神经元h1的输出o1:(此处用到激活函数为sigmoid函数)：

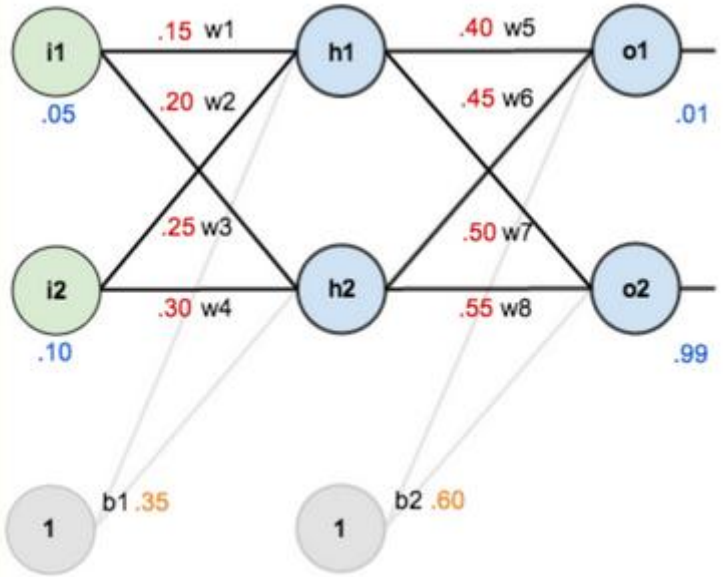
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

同理，可计算出神经元h2的输出o2：

$$out_{h2} = 0.596884378$$

第六集：朴素贝叶斯算法

- 神经网络
  - 神经网络的学习分为几个阶段，第一阶段为前向传播(forward propagation)，在该阶段中特征x将传入神经网络最终运算出对应的输出y'
    - 然后计算隐藏层到输出层的传播



计算输出层神经元o1和o2的值：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

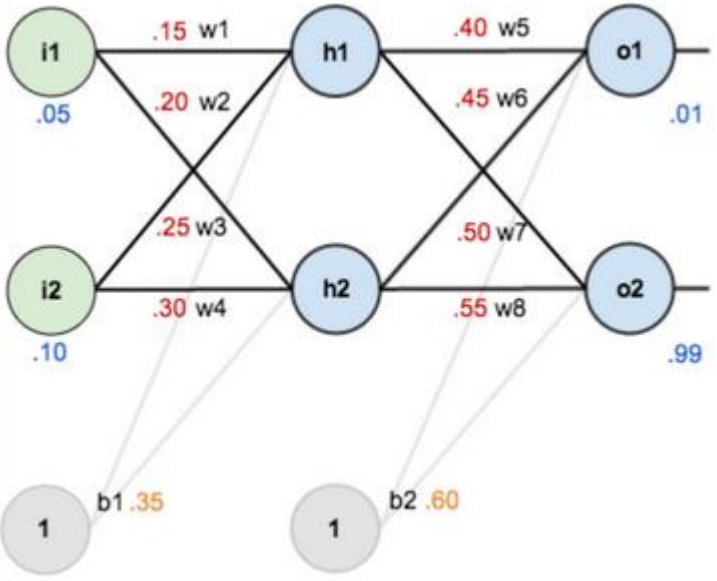
$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

$$out_{o2} = 0.772928465$$

第六集：朴素贝叶斯算法

- 神经网络
  - 此时我们已经得到了初步的第一次迭代的输出y′。容易想见，第一次迭代的输出y′很可能会和训练样本中的y相去甚远，因此需要衡量误差的大小，并由衡量的结果调整原网络中的各项权重，最终达到迭代输出的结果和原输出相近的目的
    - 其中最常见也是最简单的就是最小二乘衡量法



总误差：(square error)

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

但是有两个输出，所以分别计算o1和o2的误差，总误差为两者之和：

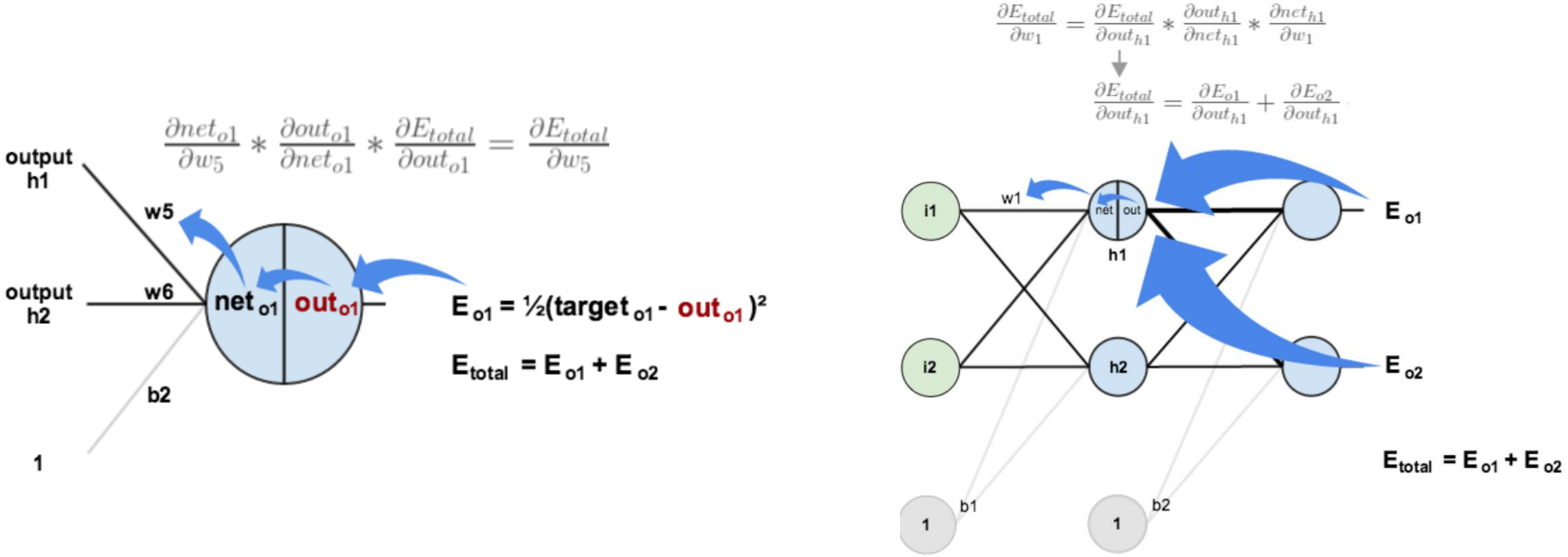
$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

第六集：朴素贝叶斯算法

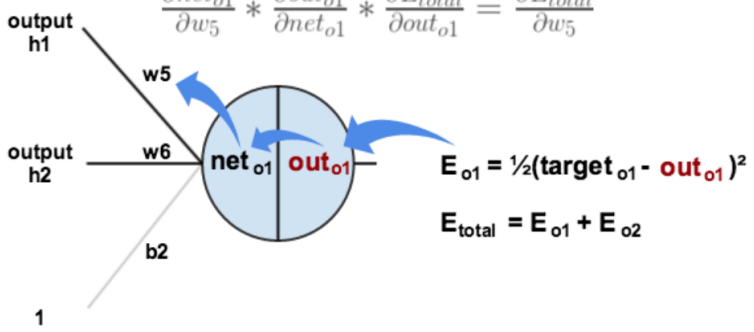
- 神经网络
  - 接下来就需要通过衡量出来的误差进行权值的调整了。由于是根据输出端反馈的误差进行权值更新的，是一个反向的过程，因此被称为反向更新(back propagation)
    - 下图显示了隐藏层参数和输入层参数的更新过程



第六集：朴素贝叶斯算法

- 神经网络
  - 对于隐藏层参数来说只需要考虑对应输出传递回来的误差

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



计算 $\frac{\partial E_{total}}{\partial out_{o1}}$ ：

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

计算 $\frac{\partial out_{o1}}{\partial net_{o1}}$ ：

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

（这一步实际上就是对sigmoid函数求导，比较简单，可以自己推导一下）

计算 $\frac{\partial net_{o1}}{\partial w_5}$ ：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

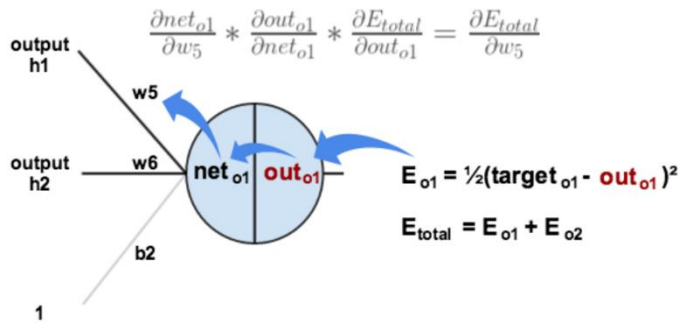
最后三者相乘：

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

第六集：朴素贝叶斯算法

- 神经网络
  - 通过误差计算得到导数后使用梯度下降法对于参数进行调整



最后我们来更新w5的值：

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

（其中，η是学习速率，这里我们取0.5）

同理，可更新w6,w7,w8:

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

第六集：朴素贝叶斯算法

- 神经网络
  - 同样地，可以对应计算出输入层参数的值

计算 $\frac{\partial E_{total}}{\partial out_{h1}}$ ：

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

先计算 $\frac{\partial E_{o1}}{\partial out_{h1}}$ ：

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

同理，计算出：

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

两者相加得到总值：

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

此时发现，在计算输入层参数的值时用到了隐藏层的结果，这也就使得输入层参数好像是隐藏层传播过来的（反向传播由此得名），保证了算法学习的效率

第六集：朴素贝叶斯算法

- 神经网络
  - 同样地，可以对应计算出输入层参数的值

再计算 $\frac{\partial out_{h1}}{\partial net_{h1}}$ ：

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

再计算 $\frac{\partial net_{h1}}{\partial w_1}$ ：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

最后，三者相乘：

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

最后，更新w1的权值：

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

同理，额可更新w2,w3,w4的权值：

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

第六集：朴素贝叶斯算法

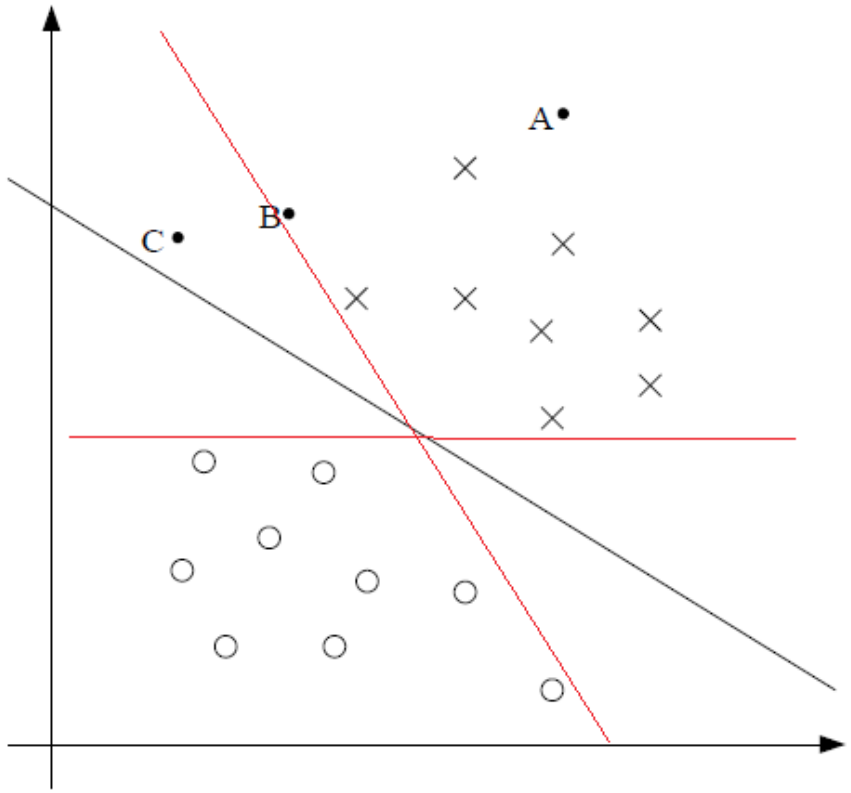
- 神经网络
  - 以上的内容展示了在神经网络中进行一次参数学习的结果，类似于线性回归中随机梯度下降的学习方法，我们先用第一组训练样本进行第一次的参数学习，然后换第二组训练样本进行第二次的参数学习，...，最终用最后一组的训练样本进行最后一次的参数学习
    - 如果把上述的行为看成一个批次，收敛条件为上下两个批次的所有训练样本的总误差的差距小于阈值
    - 训练出来的网络可直接用于测试，网络对应的输出即为测试的结果
  - 与此同时我们也发现，神经网络这一算法对应的目标函数很可能不是一个凸函数，甚至不是一个拟凸函数，因此梯度下降方法得到的解很可能不是全局最优解

第六集：朴素贝叶斯算法

- 支持向量机
  - 支持向量机也是一种非线性分类的方法，是一种直接的通过非线性的曲线进行类的划分
    - 最原始的支持向量机用于线性分类，目的在于找到一条最优的划分直线。而当支持向量机和核方法结合在一起后，才真正实现了非线性分类
  - 这时候就涉及到衡量一条划分的直线的好坏的问题
    - 这时候就先来看看如何评价简单的Logistic分类器。对于一个Logistic分类器来说其核心是Sigmoid函数，在这个函数中当$\theta^T x \geq 0$时分类器输出为1，否则分类器的输出为0
      - 可以想见对于一个Logistic分类器来说，当$\theta^T x$远大于0时认为对应的类标极有可能是1；而当$\theta^T x$远小于0时认为对应的类标极有可能是0
      - 进而可以想见，对于一个良好的Logistic分类器来说，对于类标为1的样本x对应的$\theta^T x$要尽量大于0；而对于类标为0的样本x对应的$\theta^T x$要尽量小于0
        - 对于大于小于的程度将通过函数距离(functional margin)的形式进行衡量

第六集：朴素贝叶斯算法

- 支持向量机
  - 将上述观点对应到图像上有如下的结论



假设x对应的类标为1，o对应的类标为0.对于图中A、B、C三点来说，A点离黑线很远因此我们可以判断A极有可能类标为1，但是对于C点来说由于离黑线非常近，因此不能很确定C的类标是1.

而从不同的直线角度上来看，同样都是进行{0,1}的划分，黑线的效果比红线好得多。原因在于黑线距离样本点距离更远，我们更有信心对于样本点的类标做出判断。在这里对于远近的程度将通过几何距离(geometrical margin)的形式进行衡量

第六集：朴素贝叶斯算法

- 支持向量机
  - 再进一步进行距离的定义前，先对一些符号作出说明
    - 对于类标y而言由之前的{0,1}更改为{-1,+1}，因此h(x)对应输出的值域也就自然地从原来的{0,1}更改为了{-1,+1}
    - 对于$\theta^T x$而言，原来规定θ^T和x均为n+1维的向量（x的最后一维为1），结果为θ0+θ1x1+θ2x2+...+θnxn。现将其截距项单独拆开来变成$w^T x+b$，其中w^T和x均为n维的向量
      - 由此分类的过程将表述为（其中当$w^T x+b \geq 0$时结果为+1，否则为-1）

$$h_{w,b}(x) = g(w^T x + b).$$

第六集：朴素贝叶斯算法

- 支持向量机
  - 接下来将对函数距离和几何距离做出定义
    - 对于函数距离来说采用如下的定义

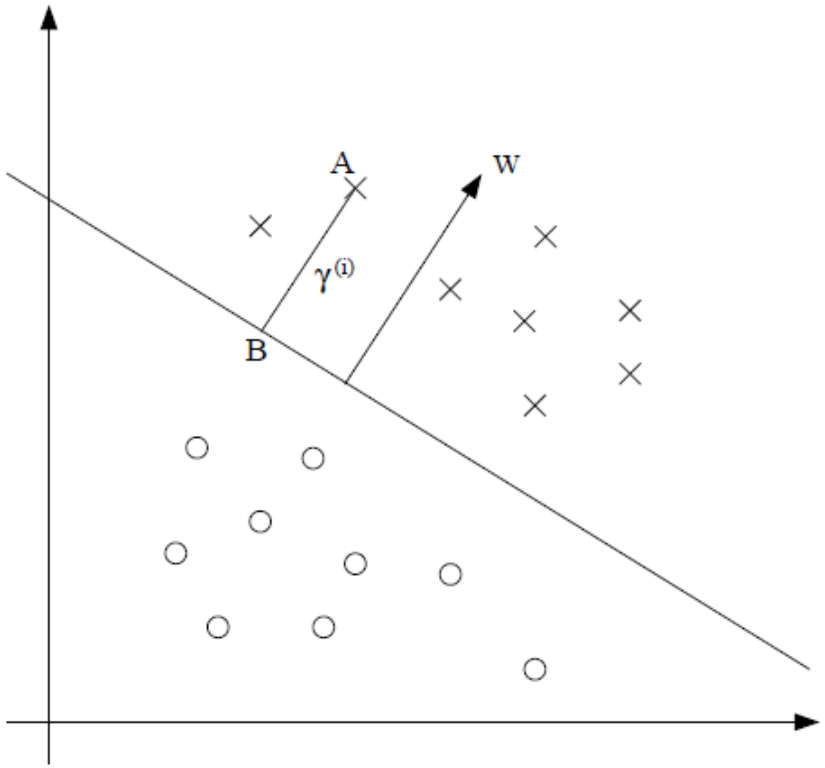
training example $(x^{(i)}, y^{(i)})$, we define the **functional margin** of (w, b) with respect to the training example

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b).$$

      - 对于这个定义可以发现当这个样本的类标y=+1时，函数距离等于w^Tx+b；当这个样本的类标y=-1时，函数距离等于$-(w^Tx+b)$
      - 这也就使得要取得大的函数距离必须要使w^Tx+b远离原点，符合之前对函数距离的设想。此时也可以观察到对于函数距离中w^Tx+b再乘上一个正的系数k变成$k(w^Tx+b)$也是符合设想的（这一点在后面的内容中可能会用到）
      - 与此同时还发现对于一个样本来说分类正确时函数距离为正，否则为负

第六集：朴素贝叶斯算法

- 支持向量机
  - 接下来再来看几何距离



对于几何距离来说我们需要衡量的是点到直线的距离（实际上更加准确的，此处的直线应该是一个超平面，因为不一定是二维空间上的）。这个超平面的表达式为w^Tx+b，其中w指示了每个轴上的斜率（即w指示的是超平面的倾斜方向）称为该超平面的法向量，其单位向量$w/||w||$称为该超平面的单位法向量，b指示的是该超平面从原点开始的偏移量。对于超平面上的各点，$w^Tx=-b$。

我们现在要衡量点A到超平面的距离，相当于计算AB两点间的距离。对于B点来说，相当于A点以超平面的单位法向量为单位向量作平移，平移的距离大小即为所求。由此可以表示出B点的表达式和对应AB两点间距离为：

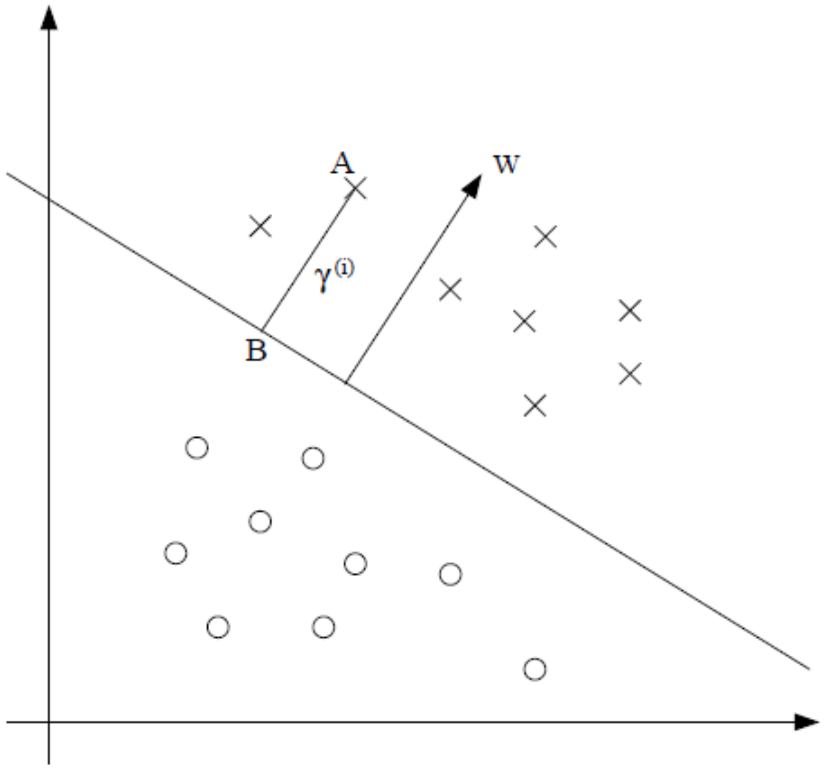
$$w^T\left(x^{(i)}-\gamma^{(i)}\frac{w}{||w||}\right)+b=0.$$

Solving for $\gamma^{(i)}$ yields

$$\gamma^{(i)}=\frac{w^Tx^{(i)}+b}{||w||}=\left(\frac{w}{||w||}\right)^T x^{(i)}+\frac{b}{||w||}.$$

第六集：朴素贝叶斯算法

- 支持向量机
  - 接下来再来看几何距离



由于在另外一个方向上的点到超平面的距离，对应的方向为超平面的单位法向量的反向。因此对于全部的样本点来说点到超平面的距离可通过下式进行衡量，即几何距离的衡量方法：

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{||w||} \right)^T x^{(i)} + \frac{b}{||w||} \right)$$

第六集：朴素贝叶斯算法

- 支持向量机
  - 对于一个分类器来说要做的是要使得其函数距离和几何距离尽可能地大，但是单纯从距离的定义上面看，要增大它们最简单的办法是使得w, b增大k倍（其中k为正数）。但这样并没有改变对于距离的设想，是没有意义的
    - 因此需要限制w, b，因此加上限制条件||w||=1，以避免这种无意义的等比例放大。在这样的假设下几何距离等价于函数距离，从而我们可以将两个距离进行统一考量。
    - 对于一个良好的分类器来说我们要使得距离尽可能大，因此我们的目标为要使得最坏样本情况下的距离尽可能地大，由此可得下面的对于整个数据集的距离定义

Given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$, we also define the function margin of (w, b) with respect to S as the smallest of the functional margins of the individual training examples. Denoted by $\hat{\gamma}$, this can therefore be written:

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}^{(i)}.$$

第六集：朴素贝叶斯算法

- 支持向量机
  - 支持向量机算法就是解决上述问题的算法，为的是在给定训练集的情况下找到对应的超平面对于数据及进行分类，同时使得最坏样本情况下的距离尽可能大，其目标表述如下：

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \ldots, m \\ & ||w|| = 1. \end{aligned}$$

在这里使用了几何距离进行目标表述（实际上在||w||=1的前提下函数距离等价于几何距离）。第一个约束条件说明我们要使得所有样本的几何距离的最小值最大化（在这里面假定了数据集是线性可分的）