CS229 Lecture notes

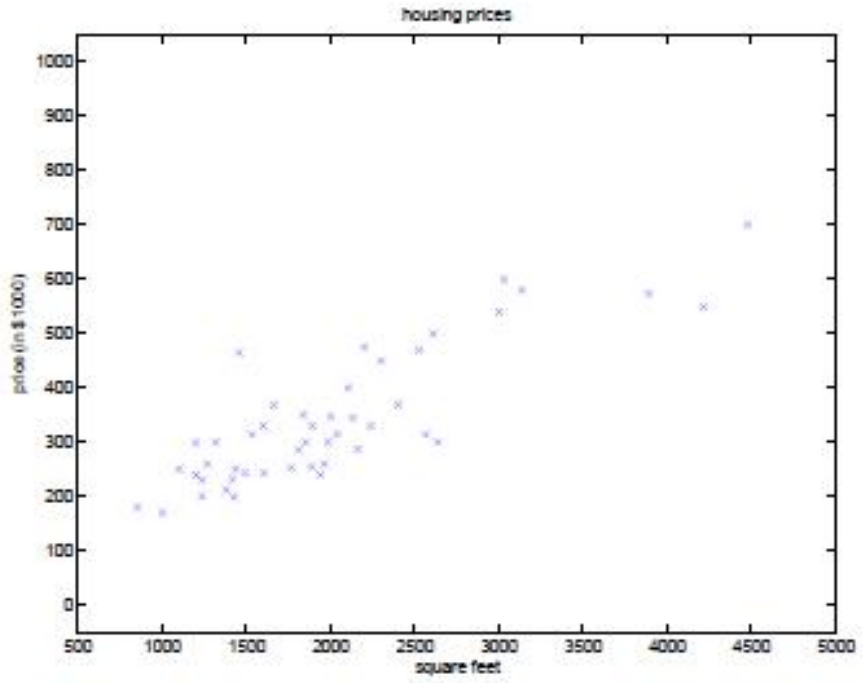
Andrew Ng

监督式学习

让我们开始先讨论几个关于监督式学习的问题。假设我们有一组数据集是波特兰，俄勒冈州的 47 所房子的面积以及对应的价格

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

我们可以在坐标图中画出这些数据：

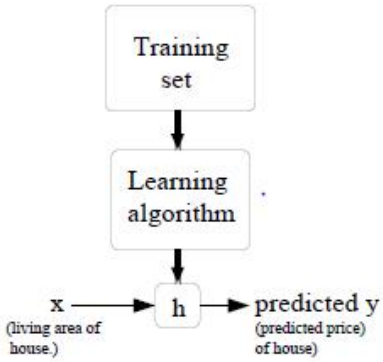


给出这些数据，怎么样我们才能用一个关于房子面积的函数预测出其他波特兰的房子的价格。

为了将来使用的方便，我们使用 x^i 表示“输入变量”（在这个例子中就是房子的面积），也叫做“输入特征”，y^i 表示“输出变量”也叫做“目标变量”就是我们要预测的那个变量（这个例子中就是价格）。一对（x^i, y^i）叫做一组训练样本，并且我们用来学习的--- 一列训练样本 {（x^i, y^i）；i=1，…，m} --叫做一个训练集。注意：这个上标“（i）”在这个符号

表示法中就是训练集中的索引项，并不是表示次幂的概念。我们会使用 x 表示输入变量的定义域，使用 $\mathcal{Y}$ 表示输出变量的值域。在这个例子中 x =Y=R

为了更正式的描述我们这个预测问题，我们的目标是给出一个训练集，去学习产生一个函数 h：X→ Y 因此 h(x)是一个好的预测对于近似的 y。由于历史性的原因，这个函数 h 被叫做“假设”。预测过程的顺序图示如下：



当我们预测的目标变量是连续的，就像在我们例子中的房子的价格，我们叫这一类的学习问题为“回归问题”，当我们预测的目标变量仅仅只能取到一部分的离散的值（就像如果给出一个居住面积，让你去预测这个是房子还是公寓，等等），我们叫这一类的问题是“分类问题”

PART I

Linear Reression

为了使我们的房子问题更加有趣，我们假设我们知道每个房子中有几间卧室：

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

在这里，x 是一个二维的向量属于 R^2。例如， x_1^i 就是训练集中第 i 个房子的居住面积，

x_2^i 是训练集中第 i 个房子的卧室数量。（通常情况下，当设计一个学习问题的时候，这些输

入变量是由你决定去选择哪些，因此如果你是在 Portland 收集房子的数据，你可能会决定包含其他的特征，比如房子是否带有壁炉，这个洗澡间的数量等等。我们以后可能会涉及到更多变量的问题，现在我们先按照给定的变量的讲解。）

为了完成监督是学习，我们必须决定怎么样去描述我们的函数/假设 h 在计算机中。有一个最初的选择，我们把 y 近似的看成是 x 的一个线性函数：$h_\theta(x)=\theta_0+\theta_1x_1+\theta_2x_2$

在这里， θ（i)是参数（也叫做权重）是 y 关于 x 的线性函数之间的参数。当 y 与 x 之间没有其他影响因素的时候我们将会舍弃下标 θ，通常写为 h(x)。为了简化我们的标注，我们习惯上令 x_0=1（这个是截距），因此可以写成

$$h(x)=\sum_{i=0}^{n}\theta_i x_i=\theta^T x,$$

右边的 θ 和 x 都是向量并且这里 n 是输入的变量的个数（不是计算 x0 的个数）。

现在给定一个训练集，我们怎么选择、学习、计算权重 θ？一个合理的方法类似与让 $h(x)$ 尽可能的接近于 y，至少对于我们所训练的数据都是适合的。使这个近似形式化，我们定义一个测量函数去记录对于每一个 θ，h($x_{(i)}$)有多接近于 $y_{(i)}$。我们定义一个**代价函数**

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2.$$

如果你以前了解过线性回归，你会认出这个和最小二乘法比较近似。不管你以前是否看过他，让我们继续，并且我们最终会证明这个知识众多的算法大家庭中的一个特例而已。

1 LMS algorithm(Least Mean Square 最小均方差算法)

我们想去选择一个 θ 使得 J(θ)取的最小值。为了这样做，我们用一个寻找算法给 θ 赋一个初值（随机的），然后不断的重复改变 θ 的大小以便是 J(θ)更小，直到我们找到一个 θ 是的 J(θ)达到我们期望的最小值。特别的，我们考虑“梯度下降算法”，用下面这个公式寻找 θ 。

$$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta).$$

（这个更新过程同时的对所有的 j=0…n 执行） α 表示“学习速率”。这是一个自然算法，反复的对 J 在减小的方向上迈出一大步直到最小。

为了执行这个算法，我们需要做的工作就是计算出等号右边的偏导数。首先我们计算出一组（x，y）样本的偏导数，这是我们可以先忽略掉对 J 的求和。

（运用求导定律很容易就能求出导数）

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x)-y\right)^2 \\ &= 2\cdot\frac{1}{2}\left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x)-y\right) \\ &= \left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\ &= \left(h_\theta(x)-y\right)x_j\end{aligned}$$

对于单一的训练样本，这里给出了更新的规则：

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}.$$

这个规则就叫做 LMS 更新规则（LMS 是 least mean squares 的缩写）也被叫做 Widrow-Hoff（就是 Widrow 和 Hoff 这两位大仙发明的这个算法。参考链接：http://baike.baidu.com/link?url=bmZNDF9xV8GMtSE_rk9eV_9UbE9wGrnAdYqyf876U3Lf4IKfkRZVCoACvxF2dm1zmRDu1UUYzW9nQs-8oPWhu_）学习规则。这个算法有几个自然的和直观的特性。例如，更新的量级正比于误差项 $(y_{(i)} - h_(x_{(i)}))$；因此，当我们遇到一组训练样本的预测值非常接近他的真实值的时候，我们会发现在更新过程中权重项基本不变；相反的这个权重项会有一个大的变化当我们的预测值 $h_\theta(x_{(i)})$有大的误差的时候（例如预测值和真实值 y(i)差别非常大的时候）

我们推断出了当有一个训练样本是的 LMS 算法。我们有两种方法可以让这个算法去适应多于一个训练样本的例子。第一种是用下面这种算法替换:

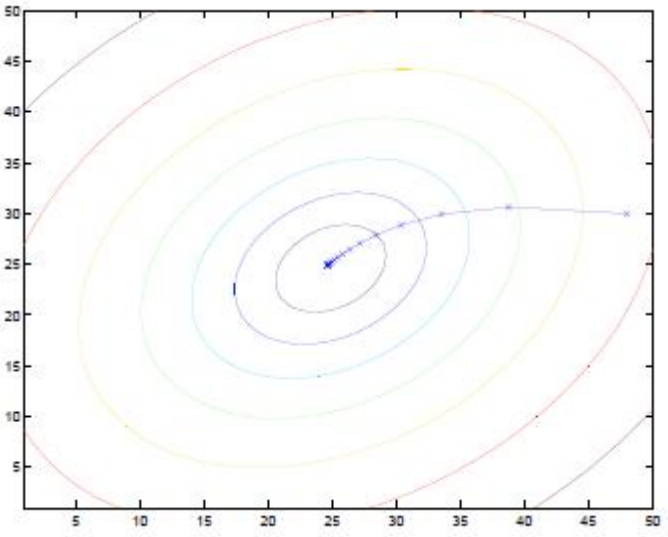
Repeat until convergence {

$\theta_j := \theta_j + \alpha\sum_{i=1}^{m}\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$ (for every j).

}

可以很容易的证明上述的更新规则算法的运算量仅仅是 $\frac{\partial J(\theta)}{\partial\theta_j}$（对 J 的初始化定义）。

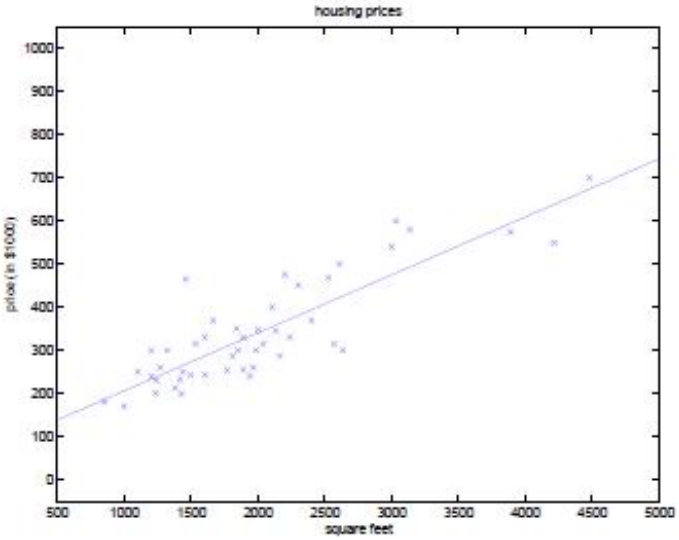
因此这是一个简单的梯度下降算法对于原始成本函数 J。这个方法注重每一个训练样本在训练过程中的每一步，所以也被叫做“批量梯度下降”。注意，梯度下降算法容易受到局部最小值的影响，这个优化问题我们对于线性回归算法只有一个目标就是找到最合适的，因此梯度下降算法总是收敛于全局最小值。（将设学习率 α 不是很大）实际上，J 是一个凸函数。这里我们有一个梯度下降算法的例子，是使这个二次函数取得最小值。



B^T

这些椭圆表示了这个二次函数的等高线。这个紫色的线条表示的是取得最小值的轨迹，初始化（48，30）。这个 X 标识出来在图中在梯度下降算法计算出的每个 θ 值，并且用实线连接起来。

当我们运行批量梯度算法去计算 θ 在我们以前的数据集去估计房子的价格使用房子的价格和房子面积的函数，我们得到 θ_0 =71.27， θ_1 =0.1345.如果我们把 $h_\theta(x)$ 当作 x（面积）的函数，使用训练样本中的数据，我们能得到下面这张图：



如果卧室数量也被当作一组输入变量，我们得到 θ_0 =89.60， θ_1 =0.1392，

θ_2 =-8.738.

上面这些结果都是我们使用批量梯度算法得到的。对于批量梯度算法算法可以有其他的选择使他更好的预测。考虑一下下面这个算法：

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}

在这个算法中，我们重复的使用梯度下降算法在训练样本中，并且每一次我们遇到一个训练样本，我们更新这个权重仅仅根据每一次对训练样本训练的梯度误差。这种算法叫做“随机梯度下降法”（也叫做增量梯度下降法。）然而批量梯度下降法必须要扫描全部的训练集在进行每一步之前----一个多余的操作如果 m 特别大的话----随即梯度下降算法可以随时开始，并且可以继续进行下一步在他跟踪的样本上。一般情况下，随即梯度下降算法会比批量梯度算法更快的使的θ“接近”最小值。（注意虽然根本不可能收敛到最小值，并且这个权重θ会一直震荡在使的 J(θ)取得最小值的θ的附近；但是实际上大多数在最小值附近的值已经可以取了，已经可以近似的使函数取得最小值了）。更多的原因，特别是当这个训练集很大的时候，随即梯度下降算法通常是优先选择相对于批量梯度算法。

2 The normal equations

梯度下降法给出了一种方法最小化 J。让我们讨论另一种方法最小化 J，这个算法明确的求解最小化并且不依赖于迭代算法。使用这种方法，我们通过计算 J 的导数来最小化 J 并且使他们等于 0。为了不在运算过程中写过多的代数和大量的矩阵，这里对使用的计算矩阵的符号做一些介绍。

2.1 Matrix derivatives、

定义一个函数，从 m*n 的矩阵到实数的映射，（f：$R^{m*n} \to R$）定义 f 关于矩阵 A 的导

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

数：　　因此梯度$\nabla_A f(A)$本身就是

一个 m*n 维的矩阵，对于（i,j）的元素就是$\frac{\partial^2 f}{\partial A_{ij}}$。举个例子，假设 A=$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$是一个

2*2 维的矩阵，并且函数 f：$R^{2*2} \to R$ 已给定为：$f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$

这里A_{ij}表示矩阵 A 中（i,j）维的元素。我们得到：

$$\nabla_A f(A) = \begin{bmatrix} \frac{3}{2} & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}.$$

这里介绍一下迹算子记作“tr”。对于一个 n*n 维的矩阵 A，A 的迹就是矩阵主对角线上元素的和$trA = \sum_{i=1}^{n} A_{ii}$。如果 a 是一个实数（例如 a 是一个 1*1 维的矩阵），tra=a。（如果你以前么有见到过这个运算符号，你可以矩阵 A 的迹看作 tr（A）或者对矩阵 A 求他的迹。）

迹运算同样适用于两个矩阵 A 和 B 因此 AB 如果都是方阵，我们可以得到 tr(AB)=tr(BA)。下面是这个公式的一些推论，我们有如下：

$$\begin{aligned} \mathrm{tr}ABC &= \mathrm{tr}CAB = \mathrm{tr}BCA, \\ \mathrm{tr}ABCD &= \mathrm{tr}DABC = \mathrm{tr}CDAB = \mathrm{tr}BCDA. \end{aligned}$$

下面对于迹的操作的一些公式也很同意证明。这里 A 和 B 都是方阵，并且 a 是一个实数

$$\begin{aligned} \mathrm{tr}A &= \mathrm{tr}A^T \\ \mathrm{tr}(A+B) &= \mathrm{tr}A + \mathrm{tr}B \\ \mathrm{tr}aA &= a\mathrm{tr}A \end{aligned}$$

我们下面给出一些矩阵导数的迹推论并不进行证明（本节中不会用到）。等式（4）仅仅适用于非奇异方阵 A，|A|表示 A 的行列式。

$$\begin{aligned} \nabla_A \mathrm{tr}AB &= B^T && (1) \\ \nabla_{A^T} f(A) &= (\nabla_A f(A))^T && (2) \\ \nabla_A \mathrm{tr}ABA^TC &= CAB + C^TAB^T && (3) \\ \nabla_A |A| &= |A|(A^{-1})^T. && (4) \end{aligned}$$

为了使我们的矩阵符号更具体，让我们现在详细解释第一类方程的意义。假设我们有一些固

定的矩阵 B∈ R^{n*m}.我们可以定义一个函数 f： R^{n*m} →R 根据 F（a）= tr(AB)。注意，这个定义是有意义的，因为如果 $A \in R^{n*m}$，然后 AB 是一个方阵，我们可以将其应用到它，因此 f 确实是从 R^{n*m} 到 R 的映射。我们可以运用我们的定义矩阵导数找到 $\nabla_A f(A)$，m*n 矩阵。

方程（1）在上述情况下，该矩阵的输入（i，j）的值可以由 B^T 给出或者等价于 B_{ji}。

方程的证明（1-3）是相当简单的，留下作为给读者的练习。方程（4）可以使用伴随矩阵和矩阵的逆来推导。

2.2 Least squares revisited

随着矩阵导数，我们开始继续在封闭的模型中寻找 θ 使的 J（ θ ）取得最小值，开始先把 J 重新写入向量矩阵中。

给定一个训练集，定义 m*n 维的矩阵 X（实际上是 m*（n+1）维的，如果算上偏置项）

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}.$$

在矩阵中包含给定的训练集的输入值在

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

同时，定义 $\vec{y}$ 为一个一维的列向量，值为所有的目标值

现在，从 $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$, 我们可以很容易证明：

$$\begin{aligned} X\theta - \vec{y} &= \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \\ &= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}. \end{aligned}$$

运用定理 $z^T z = \sum_i z_i^2$ 可以得到

$$\begin{aligned}\frac{1}{2}(X\theta-\vec{y})^T(X\theta-\vec{y}) &= \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\\ &= J(\theta)\end{aligned}$$

最后，为了最小化 J，我们寻找 J(θ)关于 θ 的导数。结合式子（2）和（3），我们发现

$$\nabla_{A^T}\mathrm{tr}ABA^TC = B^TA^TC^T + BA^TC \qquad (5)$$

因此有：

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta\frac{1}{2}(X\theta-\vec{y})^T(X\theta-\vec{y})\\ &= \frac{1}{2}\nabla_\theta\left(\theta^TX^TX\theta-\theta^TX^T\vec{y}-\vec{y}^TX\theta+\vec{y}^T\vec{y}\right)\\ &= \frac{1}{2}\nabla_\theta\,\mathrm{tr}\left(\theta^TX^TX\theta-\theta^TX^T\vec{y}-\vec{y}^TX\theta+\vec{y}^T\vec{y}\right)\\ &= \frac{1}{2}\nabla_\theta\left(\mathrm{tr}\,\theta^TX^TX\theta-2\mathrm{tr}\,\vec{y}^TX\theta\right)\\ &= \frac{1}{2}\left(X^TX\theta+X^TX\theta-2X^T\vec{y}\right)\\ &= X^TX\theta-X^T\vec{y}\end{aligned}$$

$$A^T=\theta, B=B^T=XX^T$$

在第三步中，我们使用了实数的迹就是实数本身这个定理；第四步我们使用了 $tr(A)=tr(A^T)$，在第五步中我们对式 5 使用 $A^T=\theta, B=B^T=XX^T$ 和 C=I 和公式 1.为了最小化 J，我们使他的导数等于 0 可以得到如下等式：

$$X^TX\theta = X^T\vec{y}$$

因此，使的 J(θ)最小的 θ 的值就可以有下式得到：

$$\theta=(X^TX)^{-1}X^T\vec{y}.$$

3 Probabilistic interpretation

当得到一个回归问题时，什么时候使用线性回归，什么时候选择最小二乘算法去计算价值函数 J？本节中，我们会给出一系列的假设来描述什么情况下使用最小二乘算法最合适。

假设我们的目标变量和输入变量的关系是：

$$y^{(i)}=\theta^Tx^{(i)}+\epsilon^{(i)},$$

$x^i - x$

ε^i 表示误差项（就像我们预测房价例子中有很多其他因素比如地理位置，房屋年龄等这些我们考虑外的对房价有影响的因素我们没有计算进去），或者随机噪声。我们进一步假定 ε^i 是分散的 IID(independently and identically distributed)根据高斯分布（也叫正态分布）均值为 0 方差为 σ^2。我们可以写出这个 ε^i 的定义 $\varepsilon^i \sim N(0,\sigma^2)$。也就是说 ε^i 的

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right).$$

概率密度是给定的

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right).$$

这表明：

“$p(y^{(i)}|x^{(i)};\theta)$” 说明 y^i 的分布是由 x^i 和 θ 控制的。注意，我们不能单独以 θ 为“$p(y^{(i)}|x^{(i)};\theta)$” 的条件，因为 θ 不是一个随机值。我们也能把这个式子写成另外一种形式：$y^{(i)} \mid x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$

给定 X（设定好的矩阵包含所有的输入变量 x^i）和 θ，如何求的 y^i 的分布呢？这个可能的值就是 $p(\vec{y}|X;\theta)$。这个值代表 $\vec{y}$（或者 X）的一个关于 θ 的函数。当我们明确的理解这个函数之后，我们给他起一个名字叫做似然函数：

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta).$$

注意由于这个偏差项 ε^i 的独立性（同样的 y^i 和 x^i 之间）这个式子也可以写成

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right). \end{aligned}$$

现在给定这个概率模型关于 y^i 和 x^i，怎么去选择最合理的方法去最好的求解我们想要得到的参数 θ？这个极大似然函数使的我们能尽可能的取得最好的 θ。我们应

该选择 θ 使的 L(θ)最大。

最大化 L(θ)，我们可以最大化任意的单调递增函数 L(θ)。特别的，求他的派生物（这里表示的是对数）的最大值回事比较简单的方法 $\ell(\theta)$：

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}$$

因此，最大化 $\ell(\theta)$ 相当于最小化 $\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$

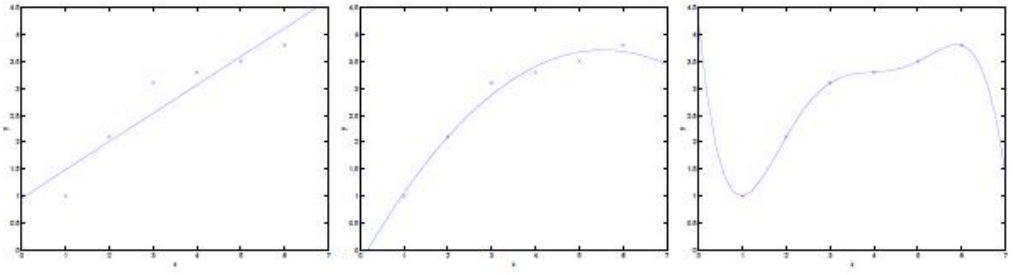
我们认出了这个 J(θ)就是我们的最小二乘法函数。

小结：在前面的概率模型计算中，使用最小二乘算法去寻找 θ 使得极大似然函数取得最大值。这个是一种比较合理的方法使用最小二乘法算法求解问题。（注意虽然这个方法是合理的和比较好的但是一定还有更加合适的方法去使用这个方法）

注意，在我们前面的讨论中，我们最终的选择 θ 并没有使用到偏差项，而且也是因为即使偏差项我们不知道，计算的结果仍然是一样的。这个结论我们会在后续讨论指数族和广义线性模型的时候用到。

4 Locally weighted linear regression（局部加权线性回归）

考虑这个问题从 x 属于 R 预测 y。这个最左边的模型显示出这个预测得到的结果函数 $y = \theta_0 + \theta_1 x$。我们看到这个数据并没有全部落到这个线上，所以说这个预测结果并不是很。



相反的，如果我们给这个函数加上一个额外的变量 x^2，预测函数则为 $y = \theta_0 + \theta_1 x + \theta_2 x^2$，然后我们得到一个比刚才那个更适给定数据的预测函数（中间这幅图）。看起来好像我们加上去的变量越多这个模型越好。然而，加上太多的模型也是危险的对于我们的预测来说：右边这个模型是一个使用了五个自变量的预测模型 $y = \sum_{j=0}^{5} \theta_j x^j$。我们可以看到这个模型完美的适和我们给定的数据，我们不能期待这个是一个好的预测模型对于使用房子的面积去预测房子的价格。在没有正式定义这些模型之前，我们可以看到左边这个模型是低拟合度的例子-----给定训练集中的数据离我们给出的预测函数相差太多----右边这个模型是过拟合度的例子。（本节课的后面，我们会给出这些定义的概念，并且给更加详细的定义去判断一个预测模型是好还是坏）

正如前面讨论的和在上面例子中展示的，特征变量的选择直接决定了一个学习模型的好坏。（当我们讨论模型选择时候，我们会看到模型会自动选择特征变量。）在这部分，我们来讨论一下局部加权线性回归算法（LWR）假设有足够的训练数据，特征变量未鉴定。这个方法是聪明的，你会得到机会去发现这个算法的一些优异之处在你的作业中。

在经典线性回归算法中，我们在一个点 x 做出预测我们将会：

1、 寻找使 $\sum_i (y^{(i)} - \theta^T x^{(i)})^2$. 最小化的 θ

2、 输出 $\theta^T x$

相反的在局部加权线性回归算法中我们会这样做：

1、 寻找使 $\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$. 最小的 θ

2、 输出 $\theta^T x$

在这里 $w^{(i)}$ 是一些非负的权值。更直接的如果 $w^{(i)}$ 是一个非常大的特殊值关于 i 的，之后在选择 θ 的过程中，我们会尽最大努力去使 $\bar{(y^{(i)} - \theta^T x^{(i)})^2}$ 更小。如果 $w^{(i)}$ 很小，之后 $\bar{(y^{(i)} - \theta^T x^{(i)})^2}$ 计算过程中的误差项可以直接被忽略在寻找的过程中。

一个公平的标准选着权值的公式是：

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

注意那些我们将要预测 x 并且对特别点 x 有依赖的权值。特别的是如果| $x^i - x$ |很小，这个权值就会接近 1；并且如果| $x^i - x$ |很大，这个权值就会很小。因此， θ 被选来作为更高的“权重”去减小误差使的取得最合适的值在 x 偏差较大的点上。（注意，而权重公式外观类似于高斯分布的密度，形成 W 是不直接跟高斯有任何联系，和特别是 W 不是随机变量，正态分布或其他参数 τ 控制。）如何快速的训练样本的重量脱落的 x 距离从查询点 X； τ 叫做带宽参数，而且也是你会在你的家庭作业见到的东西。

局部加权线性回归是我们看到的非参数化算法的第一个例子。（未加权）线性回归算法，我们前面看到的是一个众所周知的参数学习算法，因为它有一个固定的，有限数量的参数（ θ ），这是适合这个数据的。一旦我们适应 θ 并且储存了他，我们不再需要保持训练数据进行未来预测。相反，使用局部加权线性回归预测，我们需要整个训练集。术语“非参数化”（粗略）指的是，我们需要保持的东西的量，以表示的假设小时增长线性的训练集的大小。

Part II

Classification and logistic Regression(分类和线性回归)

现在让我们来分析分类问题。这就和回归问题一样，除了我们现在要预测的值 y 仅仅是一些小的离散的值之外。现在开始，我们将会把目光放在二元分类问题上，也就是 y 只能去两个值 0 和 1 。（我们在这说的简单的形式可以推广到更加复杂的形式）。例如，如果我们想要建立一个垃圾邮件分类系统，x 是邮件的特征，y 是 1 代表是垃圾邮件，是 0 则代表不是。0 也叫做否定类，1 也叫做肯定类，一些情况下也用“+、-”来表示。给定 x，y 也被叫做训练集的标签。

5 Logistic regression

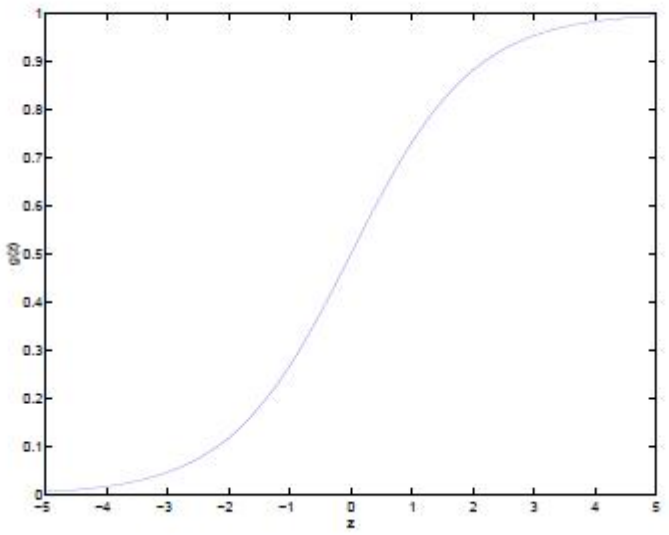
我们可以解决分类问题不论 y 是否是离散值，并且用以前的线性回归算法去试着预测我们给定 x 的 y 值。然而，非常容易证明这个方法是不足的。更直观的是，他对于 $h_\theta(x)$ 没有什么意义当值大于 1 或者小于 0 的时候当我们界定 $y \in \{0,1\}$ 。

为了解决这个问题，我们改变这个假设函数 $h_\theta(x)$ 。我们选择

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

逻辑函数，下面是他的函数图像



注意 $g(z)$ 趋近 1 当 z 趋近于 ∞，并且 $g(z)$ 趋近于 0 当 z 趋近于-∞。另外 $g(z)$ 和 h(x)的值域都是[0,1]。我们约定 $x_0 = 1$，因此 $\theta^T x = \theta_0 + \sum_{j=1}^{n} \theta_j x_j$

从现在开始，我们都假定 g 是已知的。其他函数如果是 0—1 之间的连续函数也是可以使用的，但是有两个原因我们将会看到（当我们讨论 GLMs 是，还有讨论生成学习算法的时候）选择他做回归函数是最合适的。在开始下一节千，这里有一个重要的推论关于 sigmoid 函数的导数：

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\ &= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

因此，给定这个逻辑回归模型，怎么去找一个适合的 θ？接下来我们将会使用最小二乘法和极大似然估计相结合来求出。将会结合概率和极大似然函数来球权重参

数。

我们假设：

$$\begin{aligned} P(y=1 \mid x;\theta) &= h_\theta(x) \\ P(y=0 \mid x;\theta) &= 1-h_\theta(x) \end{aligned}$$

注意这个也可以简单的写为 $p(y \mid x;\theta) = (h_\theta(x))^y (1-h_\theta(x))^{1-y}$

假定给定的 m 个样本之间都是相互独立的，我们可以得到如下极大似然函数：

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X;\theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

像以前一样很容易对这个似然函数的对数求最大值：

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)})) \end{aligned}$$

怎么样去最大化这个似然函数呢？像在线性回归里面所做的一样，我们可以使用梯度上升法。写成向量的形式，更新式子是 $\theta := \theta + \alpha \nabla_\theta \ell(\theta)$（注意这个公式中的正负号还有我们是要求的是最大值而不是最小值）。首先在一个训练样本（x，y）上使用这个公式，并且对他的对数求导：

$$\begin{aligned} \frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x) \\ &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial \theta_j}\theta^T x \\ &= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\ &= (y-h_\theta(x))\, x_j \end{aligned}$$

我们使用公式 $g'(z) = g(z)(1-g(z))$。求出来的就是随即梯度更新规则：

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

如果我们拿这个和 LMS 更新规则比较，我们会发现这两个是完全一样的；但是这是不一样的算法得到的，因为 $h_\theta(x^i)$ 现在是有非线性的函数 $\theta^T x^i$ 定义的。尽管如此，我们还是很好

奇为什么不一样的算法和不一样的学习方法会得到同样的结果。这是偶然吗？或者有更深层次的原因在里面，我们将会解答这个问题在 GLM 模型中。

6 Digression: The perceptron learning algorithm（感知学习算法）

我们现在额外的增加一个算法由于一些历史爱好，我们将会回归到整体之后讨论学习理论的时候。试着想一下改变这个逻辑回归模型去要求他去预测一个值是 0 或 1 或者其他。为了达到目的自然而然的从新定义 g 为一个阀值函数

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

如果我们使用 $h_\theta(x) = g(\theta^T x)$ 在前面作为我们的预测函数，其他都不变我们更新规则如：

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}.$$

我们叫新的算法为感知学习算法。

在 19 世纪 60 年代，这个“感知”被认为是一个粗陋的模型对于人脑中工作的单个神经元。考虑到这个算法比较简单并且作为一个开始对于我们下节理论讨论做一个开头来讲一下。注意虽然这个感知学习算法看起来和我们其他的学习算法一样，实际上他是一个不同于逻辑回归和最小二乘回归类型的算法，特别的是，他很难赋予感知算法概率的应用，或者使用最大似然估计算法去求解。

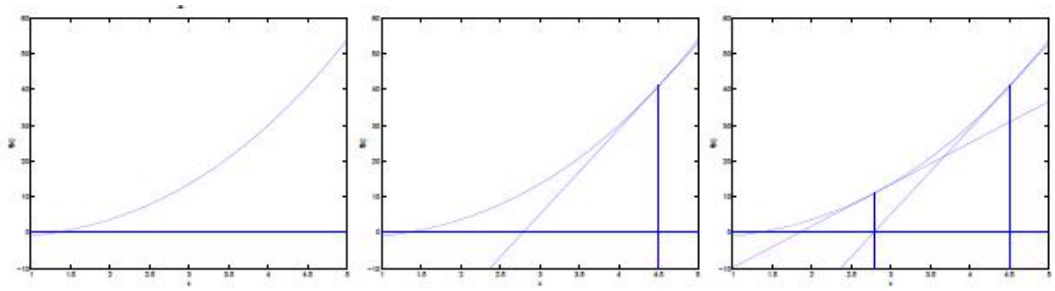
7 Another algorithm for maximizing $\ell(\theta)$

回到逻辑回归算法 g(z)使用 sigmoid 函数，我们接下来讲解一种不一样的算法来最小化 $l(\theta)$

开始之前，我们先看一下牛顿的方法去寻找一个函数的零点。特别的，假设我们有一系列的函数 f：R—>R，并且我们想找到一个值使的 f(θ)=0。这里 θ 输入 R 是一个实数。牛顿的方法如下：

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}.$$

。这个方法我们可以这样理解，选定一个值做自变量的垂线过函数上这点做函数的切线与函数轴交与一点，再过这点做垂线，再过这点做函数的切下知道找到切线斜率为零的点。下面是一副图表示牛顿法的过程



在左边这张图中我们看到函数 f 画了一条直线在 y=0 处。我们希望找到的 θ 使得 f(θ)=0;这个 θ 的值大约是 1.3。假设我们初始化这个算法定义 θ =4.5.牛顿的方法是通过函数上这点做 f 的切线，并且评估这条线的值是否为 0.这给我们下一个猜想的对于 θ ，大约是 2.8.最右边的图指出了这个最后一次迭代的结果，这时候更新的 θ 大约是 1.8.一阵迭代之后我们能很快的接近 θ =1.3.

牛顿的方法给出了一种使 f(θ)=0 的方法。我们怎么把这个应用到我们求解 l 函数中？这个函数的最大值相当于去求他的一阶导数等于 0 的点。因此令 f(θ)=l`(θ),我们能使用相同的

$$\theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}.$$

算法去是 L 最大，我们得到这个更新规则：

（需要考虑的问题：如果我们想要求一个函数的最小值而不是最大值这个算法应该如何改进？）

最后，在我们的逻辑回归设定中， θ 是一个向量，因此我们需要推广牛顿法到向量级别。这个推广的方法是把牛顿法应用到多维中（也叫做牛顿—拉普森方法）

$$\theta := \theta - H^{-1}\nabla_\theta \ell(\theta).$$

这里 $\nabla_\theta \ell(\theta)$ 是 $\ell(\theta)$ 对 θ 的偏导数，H 是一个 n*n 的矩阵（实际上是 n+1*n+1 维的，我们带上截距项）叫做 Hessian，是由下面的式子给定：

$$H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial\theta_i \partial\theta_j}.$$

牛顿法更加快速的收敛到最小值比着梯度下降法，并且需要更少的迭代次数接近最小值。一次牛顿迭代会花费更大的代价比着梯度下降算法，当他要求找到和反相一个海森矩阵的时候，但是如果n不是很大的时候，牛顿算法通常是更快的。当牛顿算法应用到求逻辑回归极大似然函数的最大值的时候，这个求解方法也被叫做Fisher scoring。

Part III

Generalized Linear Models（广义线性模型）

至今为止，我们已经看来一个回归分析的例子和一个分类分析的例子。在两个例子中都有一些近似的函数,

回归分析的例子中我们有 $y|x;\theta \sim N(\mu,\sigma^2)$，在分类的例子中有 $y|x;\theta \sim Bernoulli(\phi)$

本节我们将会展示出这些都是广义线性模型中的一部分。我们也会推到一些其他的适用于回归和分类问题中的广义线性模型。

8 The exponential family

为了引出 GLMS，我们先说一下指数分布。我们定义一个分布是指数分布如果他可以被写为如下形式：

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta)) \tag{6}$$

η 叫做特性参数（也叫做典型参数）；T(y)是充分统计量（通常情况下 T(y)=y），$a(\eta)$ 是对数划分函数。分量 $e^{-a(\eta)}$ 作用是归一化参数，确保 p(y; η)的和是大于 1 的。

T 一个复杂的选择，a 和 b 定义一个分布族关于参数 η；当我们改变 η 的值时，我们会得到不同的分布在这个分布族里面。

现在看到的 Bernoulli 和 Gaussian 分布都是指数分布的一个例子。Bernoulli 分布均值为 Φ 写为 Bernoulli（Φ），指定一个分布 y∈｛0，1｝，写成 p（y=1； Φ）=Φ；p（y=0； Φ）=1-Φ

$$h_\theta(x)$$

当我们改变 Φ 的值，我们得到不同均值的 Bernoulli 分布。现在我们证明这一类的 Bernoulli 分布，在例子中选择 T，a 和 b 所以式子（6）是 Bernoulli 分布。

我们把 Bernoulli 分布写为：

$$\begin{aligned} p(y;\phi) &= \phi^y(1-\phi)^{1-y} \\ &= \exp(y\log\phi + (1-y)\log(1-\phi)) \\ &= \exp\left(\left(\log\left(\frac{\phi}{1-\phi}\right)\right)y + \log(1-\phi)\right) \end{aligned}$$

因此，特性参数由 $\eta = \log\frac{\phi}{1-\phi}$ 给出。有意思的是，当我们使用 η 表示 Φ 的时候我们会得到 Φ=1/（1+e^（-η））。这个看起来是不是和 sigmoid 函数很像。这将会再次被提到当我们把逻辑回归分析看作 GLM 时。为了完成 Bernoulli 分布是指数分布的一种的猜想，我们进一步

$$\begin{aligned} T(y) &= y \\ a(\eta) &= -\log(1-\phi) \\ &= \log(1+e^\eta) \\ b(y) &= 1 \end{aligned}$$

有：

这表明选择适当的 T，a 和 b 的时候，Bernoulli 分布可以被写成式（6）的形式。我们进一步

来讨论 Gaussian 分布。回想一下，当我们推导线性回归时，σ^2 的值对我们最终选择的 θ 和 $h_\theta(x)$ 没有任何影响。因此我们可以选择任意值作为 σ^2 而不去改变他的值。为了简化式子我们定义 σ^2 =1.我们可以得到：

$$\begin{aligned} p(y;\mu) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(y-\mu)^2\right) \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right)\cdot\exp\left(\mu y-\frac{1}{2}\mu^2\right) \end{aligned}$$

因此我们看到高斯分布也在指数分布族里面，当

$$\begin{aligned} \eta &= \mu \\ T(y) &= y \\ a(\eta) &= \mu^2/2 \\ &= \eta^2/2 \\ b(y) &= (1/\sqrt{2\pi})\exp(-y^2/2). \end{aligned}$$

有许多其他分布指数的家人:多项分布(我们稍后将看到),在泊松分布(造型计数数据;也看到问题集);伽玛和指数(造型连续、非负随机变量,如时间间隔),β和狄利克雷(对概率分布),和许多更多。在下一节中,我们将描述构建模型的一般"食谱",y(x 和θ)来自这些发行版。

9 Constructing GLMs

假设您想要构建一个模型来估计客户来到你的商店的数量 y(或您的网站上的页面浏览量数量)在任何给定时刻,基于某些特性 x 等商店促销,最近的广告、天气、读写等。我们知道,泊松分布通常提供了一个好的模型数量的游客。知道了这一点,我们怎么能想出一个模型问题?幸运的是,泊松是一个指数族分布,所以我们可以应用一个广义线性模型(GLM)。我们将在本节中,我们将描述一个方法构建全球语言监测模型,诸如此类的问题。

更为普遍的是,考虑这样一个分类或回归问题,我们想要预测一些随机变量的值 y 作为 x 的函数。获得一个漠视这个问题,我们将进行以下三个假设条件分布的 y x 和关于我们的模型:

1、y | x;θ ~ExponentialFamily(η)。即。鉴于 x 和 θ,y 的分布遵循一些指数族分布,参数 η。

2、鉴于 x,我们的目标是预测 T 的预期值 x(y)。在我们的大多数示例中,我们将 T(y)= y,所以这意味着我们想预测 h(x)的输出由我们学习假说 h 满足 h(x)= E[y | x]。(注意,这个假设是选择满足 h(x)逻辑回归和线性回归。例如,在逻辑回归,我们有 h(x)= p(y = 1 | x;θ)= 0 • p(y = 0 | x;θ)+ 1 • p(y = 1 | x;θ)= E[y | x;θ]。)

3、η的自然参数和输入 x 相关线性:η=θT x。(或者,如果η量值,那么ηi =θTi x)。

第三的这些假设似乎是最合理的,而且它可能是更好的认为是一种“设计选择”在我们的配方设计的漠视,而不是作为一个假设本身。这三个假设/设计的选择将使我们能够获得一个非常优雅的 classof 学习算法,即全球语言监测机构,有很多可取的属性,如易于学习。此外,生成的模型往往是非常有效的模拟不同类型的分布在 y;例如,我们不久将表明,逻辑回归和普通最小二乘法都可以派生的漠视。

9.1 Ordinary Least Squares

表明普通最小二乘法是一种特殊的GLM的家庭模型,考虑设置目标变量y(也称为响应变量在GLM术语)是连续的,而且我们模型 x y 给定的条件分布为高斯N(μ,σ2)。(在这里,μ可能取决于x)。所以,我们让ExponentialFamily(η)分布是高斯分布。正如我们之前看到的,

$$\begin{aligned} h_\theta(x) &= E[y|x;\theta] \\ &= \mu \\ &= \eta \\ &= \theta^T x. \end{aligned}$$

配方的高斯指数族分布,我们有μ=η。所以,我们有

上面的第一个假设 2,平等;第二个平等的合集 y | x;θ~N(μ,σ2),所以其预期值等于μ;

第三个平等从假设 1(和我们先前推导表明,μ=η配方的高斯指数族分布),和最后一个平等遵循从假设 3。

9.2 Logistic Regression

我们现在考虑逻辑回归。我们感兴趣的是二进制分类,因此y∈{0,1}。鉴于y binary-valued,因此自然选择的伯努利家族分布模型的条件分布 x y。在我们制定的伯努利分布指数族分布,

我们有Φ= 1 /(1 + e−)。此外,请注意,如果x,y |θ~伯努利(Φ),然后E x y |,θ=Φ。

similarderivation 后,作为一个普通的最小二乘法,我们得到:

$$\begin{aligned} h_\theta(x) &= E[y|x;\theta] \\ &= \phi \\ &= 1/(1+e^{-\eta}) \\ &= 1/(1+e^{-\theta^T x}) \end{aligned}$$

所以,这给了我们假设函数形式的h(x)= 1 /(1 + e−T x)。如果你以前想知道我们想出了物流功能的形式 1 /(1 + e−z),这给了一个答案:一旦我们假设 x y 条件是伯努利,它产生的后果的漠视和指数族分布的定义。引入更多的术语中,该函数 g 给分配的平均作为自然参数的

函数(g(η)= E(T(y);η))被称为规范响应函数。其逆,g-1,称为规范化链接功能。因此,规范响应函数为高斯家庭只是识别功能,和规范化是伯努利响应函数

9.3 Softmax Regression

让我们看看一个 GLM 的例子。考虑一个分类问题的反应变量 y 可以承担任何一个 k 值,因此 y∈{ 1 2,。、k}。例如,而不是电子邮件分类到两类垃圾邮件或非垃圾邮件，将是一个二元分类问题,我们可能需要分类成三个类,如垃圾邮件、个人邮件,与工作相关的邮件。响应变量仍然是离散的,但现在能超过两个值。我们将因此分布式根据多项分布模型。

允许派生的 GLM 造型这种类型的多项数据。这样做,我们将首先表达了多项指数族分布。参数化一个多项式在 k 可能的结果,可以使用Φ1 k 参数,。,Φk 指定每个结果的概率。然而,这些参数将是多余的,或者更正式,他们不会独立(因为知道任何 k-1Φi 独特的决定
最后一个,因为他们必须满足 PkiΦi = 1 = 1)。所以,我们将参数化多项式只有 k-1 参数,Φ1,。,Φk-1,Φi = p(y =我;Φ)和 p(y = k;Φ)= 1-Pk-1 i = 1Φi。记数的便利,我们还将让Φk = 1-Pk-1 i = 1Φi,但我们应该记住,这不是一个参数,而且它完全Φ1 规定,。,Φk-1。

表达多项指数族分布,我们将定义 T(y)∈Rˆk-1 如下:

$$T(1)=\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix},\ T(2)=\begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix},\ T(3)=\begin{bmatrix}0\\0\\1\\\vdots\\0\end{bmatrix},\ \cdots,T(k-1)=\begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix},\ T(k)=\begin{bmatrix}0\\0\\0\\\vdots\\0\end{bmatrix}$$

与我们之前的例子,在这里我们没有 T(y)= y;此外,T(y)现在是一个 k-1 维向量,而不是一个实数。我们将编写(T(y))我表示的第 i 个元素的向量 T(y)。我们介绍一个非常有用的符号。一个指标函数 1{•}就值 1 如果它的参数是正确的,和 0(1{真正}= 1,1{假}= 0)。例如,1{2 = 3} = 0 和 1{3 =5-2} = 1。所以,我们还可以写 T 之间的关系(y)和 y(T(y)){ y =我} i = 1。(继续阅读之前,请确保你明白为什么这是真的!)进一步,我们有 E[(T(y))我]= P(y =我)=Φi。

现在，我们可以证明这个多项式指数家族的一员，有如下：

$$
\begin{aligned}
p(y;\phi) &= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\cdots\phi_k^{1\{y=k\}} \\
&= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\cdots\phi_k^{1-\sum_{i=1}^{k-1}1\{y=i\}} \\
&= \phi_1^{(T(y))_1}\phi_2^{(T(y))_2}\cdots\phi_k^{1-\sum_{i=1}^{k-1}(T(y))_i} \\
&= \exp((T(y))_1\log(\phi_1) + (T(y))_2\log(\phi_2) + \\
&\qquad \cdots + \left(1-\sum_{i=1}^{k-1}(T(y))_i\right)\log(\phi_k)) \\
&= \exp((T(y))_1\log(\phi_1/\phi_k) + (T(y))_2\log(\phi_2/\phi_k) + \\
&\qquad \cdots + (T(y))_{k-1}\log(\phi_{k-1}/\phi_k) + \log(\phi_k)) \\
&= b(y)\exp(\eta^T T(y) - a(\eta))
\end{aligned}
$$

where

$$
\begin{aligned}
\eta &= \begin{bmatrix} \log(\phi_1/\phi_k) \\ \log(\phi_2/\phi_k) \\ \vdots \\ \log(\phi_{k-1}/\phi_k) \end{bmatrix}, \\
a(\eta) &= -\log(\phi_k) \\
b(y) &= 1.
\end{aligned}
$$

至此我们制定的多项指数族分布。

$$\eta_i = \log\frac{\phi_i}{\phi_k}.$$

链接函数给出了 for i= 1,…,k)

为了方便起见,我们也定义ηk=log(Φk /Φk)= 0 =。逆函数并获得响应函数的联系,因此,我们有

$$
\begin{aligned}
e^{\eta_i} &= \frac{\phi_i}{\phi_k} \\
\phi_k e^{\eta_i} &= \phi_i \qquad (7) \\
\phi_k\sum_{i=1}^{k} e^{\eta_i} &= \sum_{i=1}^{k}\phi_i = 1
\end{aligned}
$$

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}$$

这意味着Φk = 1 / Pki = 1 ei,可代替回方程(7)给响应函数

这个函数映射η的Φ的叫做将 softmax 功能。

完成我们的模型,我们使用假设 3,鉴于ηi 的早些时候,x 的线性相关。所以,ηi =θTi x(因为我= 1,。 θ1,k−1),。, θk−1∈Rn + 1 是我们的模型的参数。记数的方便,我们还可以定义θk = 0,以便ηk =θT k x = 0,鉴于之前。因此,我们的模型假设的条件分布 ygiven x

$$
\begin{aligned}
p(y=i|x;\theta) &= \phi_i \\
&= \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} \\
&= \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}} \qquad (8)
\end{aligned}
$$

等于

这个模型,这也适用于分类问题 y∈{ 1,…k },叫做 softmax 回归。这是一个逻辑回归的概括。

我们的假设将输出

$$
\begin{aligned}
h_\theta(x) &= \mathrm{E}[T(y)|x;\theta] \\
&= \mathrm{E}\left[\left. \begin{array}{c} 1\{y=1\} \\ 1\{y=2\} \\ \vdots \\ 1\{y=k-1\} \end{array} \right| x;\theta \right] \\
&= \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\exp(\theta_1^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \\ \frac{\exp(\theta_2^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \\ \vdots \\ \frac{\exp(\theta_{k-1}^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \end{bmatrix}.
\end{aligned}
$$

换句话说,我们的假设将输出概率估计 p(y =我| x; θ),为每一个值的我= 1,。k。(尽管 h(x)

如上定义只是 k－1 维,显然 p(y = x k |; θ)可以获得 1－Pk－1 i = 1Φi)。

最后,让我们讨论参数拟合。类似于我们最初派生的普通最小二乘法和逻辑回归,如果我们有一个训练集 m 的例子{(x(i),y(i));i= 1,…,m },愿学习的参数 θi 这个模型中,我们将首先写下 log-likelihood

$$
\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) \\
&= \sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y^{(i)}=l\}}
\end{aligned}
$$

获得第二行以上,我们定义用于 p(y | x; θ)给出了方程(8)。我们现在可以获得的最大似然估计的参数通过最大化ℓ(θ)θ,使用梯度上升或牛顿法等方法。

Part IV

Generative Learning algorithms 生成学习算法

到现在为止，我们已经学习关于 $p(y|x;\theta)$ 模型、y 关于 x 的条件分布学习算法。例如：

回归分析模型的 $p(y|x;\theta)$ 是应用 $h_\theta(x)=g(\theta^T x)$，g 是激活函数。在本节中，我们将会学习到一些不一样类型的学习算法。

思考一个分类问题我们想要把大象（y=1）和狗（y=0）识别出来基于一些动物的特征.给定一个训练集，一个类似于逻辑回归的算法或者感知计算法试图找到一条直线-----一个判定边界----去区分大象和狗。然后，去分类一个动物既不是大象也不是狗，这个算法核实这个动物是在这个判定边界的哪一边，并且给出他相应的预测是大象或者是狗。

这里有一个不一样的方法。首先，对于大象，我们能建立一个什么是大象的模型。然后对于狗，我们能建立一个单独的模型对于什么是狗。最后，去识别一个新的动物，我们会拿这个动物去和大象的模型对比，然后再去和狗的模型去对比，看这个新的动物是像大象多一些还是像狗多一些在我们的训练集中。

算法试图直接去得到一个函数 P(y|x)（像逻辑回归那样）或者算法试图去直接从输入 x 到标签｛0，1｝建立一个映射（像感知机算法一样）都叫做区别学习算法。这里，我们将要学习一个算法不是去建立一个函数 P(x|y)。这些算法被叫做生成学习算法。例如，如果 y 的值表示一个样本是狗或者大象，那么 p(x|y=0)代表狗的特征的分布 p(x|y=1)代表着大象的特征的分布。

建立模型 p(y)和 p(x|y)之后，我们的算法能够用贝叶斯规则去表示 y 在给定 x 的条件下的概率

$$p(y|x)=\frac{p(x|y)p(y)}{p(x)}.$$

这里分母是由 p(x)=p(x|y=1)p(y=1)+p(x|y=0)p(y=0)得到。（全概率公式）因此也同样能够表示 p(x|y)和 p(y)。事实上，如果我们计算 p(y|x)只是为了去做预测，我们根本不需要去算这个分母的值因为：

$$\begin{aligned}\arg\max_y p(y|x) &= \arg\max_y \frac{p(x|y)p(y)}{p(x)}\\ &= \arg\max_y p(x|y)p(y).\end{aligned}$$

1 Gaussian discriminant analysis 高斯判别分析

我们将要学习的第一个生成学习算法是高斯判别分析。在这个模型中我们假设 p(x|y)是离散的用一个正态分布函数去赋值。在讲这个算法前，简单说一下正态分布。

1.1 The multivariate normal distribution

多元正态分布在 n 维空间中也叫做多维高斯分布，由参数 μ （ $\mu \in R^n$ ）和协方差矩阵 Σ （ $\Sigma \in R^{n*n}$ ），Σ >=0 并且是正定的和对称的。也写作 $\text{“}\mathcal{N}(\mu, \Sigma)\text{”}$ ，他的概率密度分布是

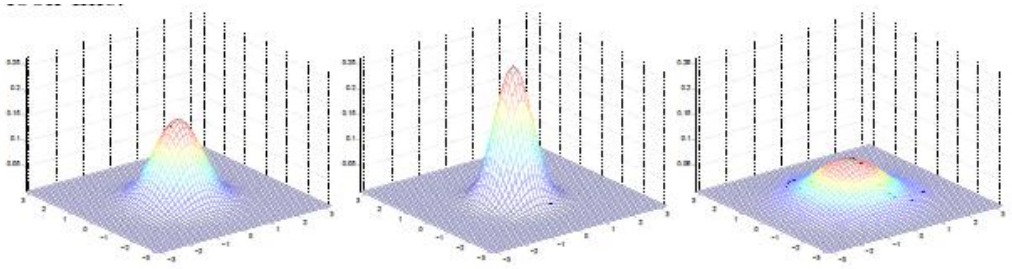
$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

在上面的等式中，|Σ|是Σ的模。

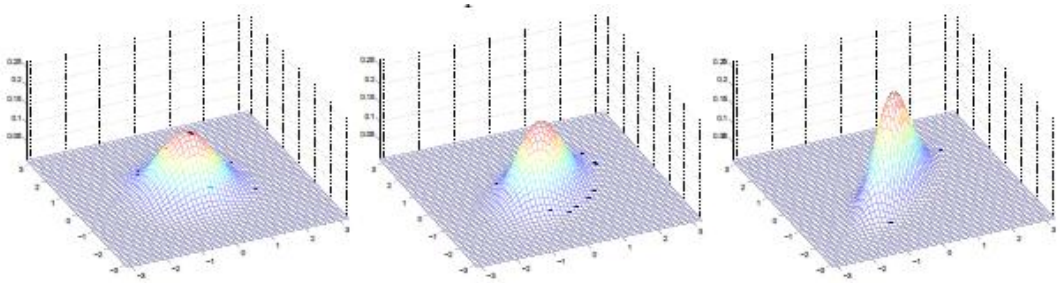
对于一个随机的变量 X 分布为 $\text{“}\mathcal{N}(\mu, \Sigma)\text{”}$ ，均值是 μ

$$\mathrm{E}[X] = \int_x x\, p(x;\mu,\Sigma)dx = \mu$$

随机变量Z的协方差的定义是Cov(Z) =E[(Z — E[Z])(Z — E[Z])т].把这个公式推广到一般化。这个协方差也被定义为Cov(Z) = $\mathrm{E}[ZZ^T] - (\mathrm{E}[Z])(\mathrm{E}[Z])^T$ （你可以自己证明这两个等式是相等的。）如果X~N(μ , Σ),Cov(X)= Σ
下面有一些高斯分布的图像：



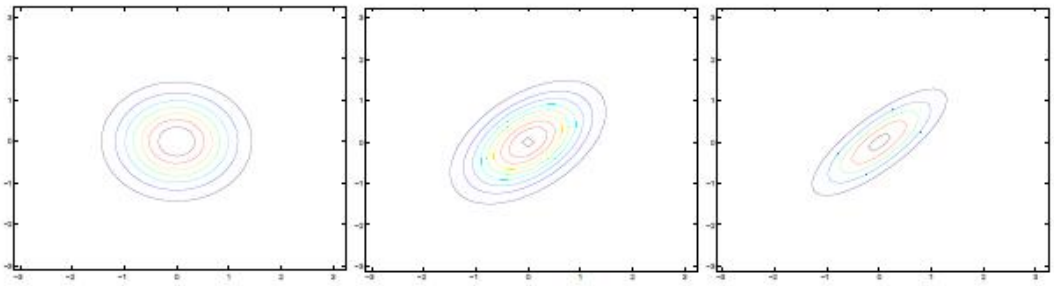
最左边的图像表示的是均值是0（2*1的零向量）协方差矩阵Σ=1（2*2的单位矩阵）这个高斯分布也叫做标准正态分布。中间的图像表示的均值是0和Σ=0.6I；右边的表示的是Σ=2I。我们从图中可以看出来Σ越大，高斯分布越“细长”当他变小时候分布变的更加“扁平”
来看更多的例子：



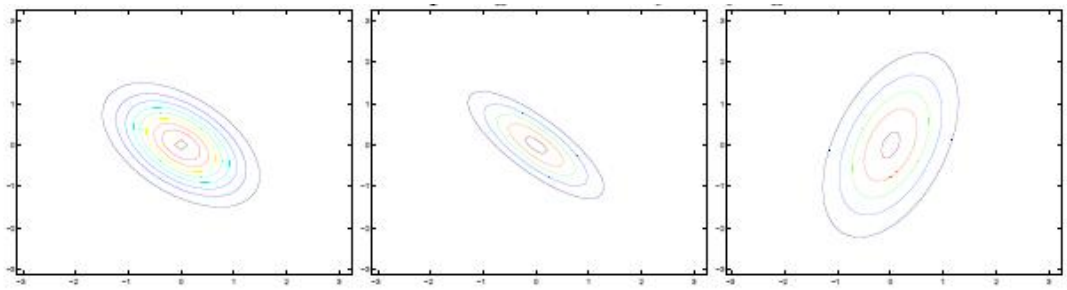
上面的图像表示的是高斯分布的均值都是0，协方差矩阵分别为：

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix};\quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix};\quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

最左边的图像表示的是常见的标准正态分布，我们可以看到当我们增加Σ的副对角线的元素的值时，这个分布的密度变的更加“扁平”朝着45度方向（x1=x2）。我们能够更清楚的看出来这个规律，当我们看这三副图的俯视图的时候。



下面这个是最后一个训练样本的一组俯视图通过改变Σ

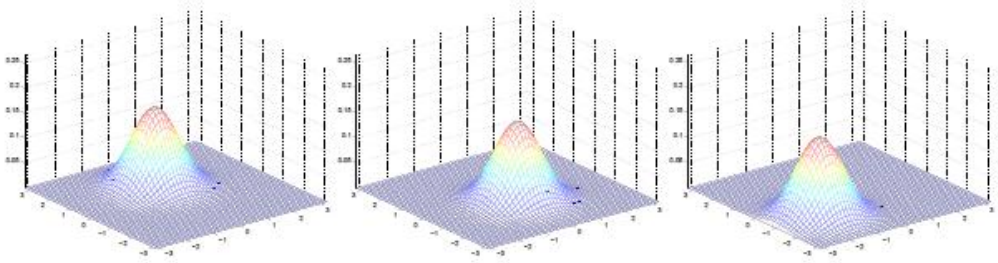


The plots above used, respectively,

$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$

从最左边和中间的图像，我们可以看到通过减少协方差矩阵副对角线元素的值，这个图像密度再次变“扁平”，但是在相反的方向。最后我们改变使的形状类似一个椭圆（最右边的图像表示出来了。）

像我们在最后一个例子中设置的，使的Σ=I，通过改变μ的值，我们也能改变这个密度。



The figures above were generated using $\Sigma = I$, and respectively

$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}.$$

1.2 The Gaussian Discriminant Analysis model 高斯判别分析模型

当我们拿到一个输入变量是连续分布的随机值的分类问题的时候，这时我们能够使用高斯判

别分析模型去处理，像对于 p(x|y)使用多元正态分布。这个模型是：

$$\begin{aligned} y &\sim \text{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

写出这个分布规律：

$$\begin{aligned} p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) \\ p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \end{aligned}$$

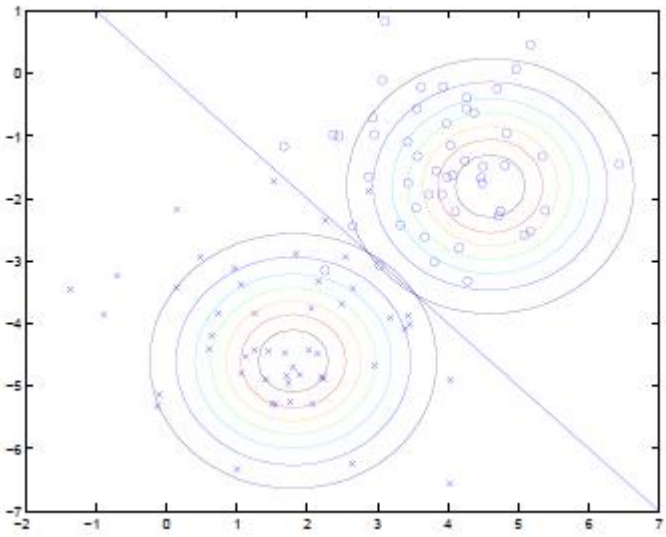
这里，我们模型使用的参数是 ϕ， Σ， μ0， μ1。（注意这里有两个不同含义的变量值 μ0 和 μ1，这个模型通常使用一个协方差矩阵 Σ）这个数据极大似然对数是：

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi). \end{aligned}$$

通过最大化关于参数的函数 ι，我们找到了使的极大似然函数最大化的参数值：

$$\begin{aligned} \phi &= \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)}=1\} \\ \mu_0 &= \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}} \\ \mu_1 &= \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}} \\ \Sigma &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T. \end{aligned}$$

在下面图中可以很明显的看出来这个算法是怎么工作的：



在图中显示的是训练集，两个高斯分布的等高线已经表明把这个训练集分为两部分。注意这两个高斯的等高线是一样的形状和方向，因此他们的使用的是一个相同的协方差矩阵 Σ，但是他们有不一样的均值 μ_0 和 μ_1。在这个图中另外一个就是这条直线给出了判定边界在 $p(y = 1|x) = 0.5$ 处。在这个边界的一边我们可以预测 y=1 对大部分的输入都是成立的，在另一边我们预测 y=0 的。

1.3 Discussion: GDA and logistic regression

1.3 讨论：GDA 和逻辑回归

高斯判别模型和逻辑回顾之间有一个有意思的联系。如果我们把这个数量 $p(y = 1|x; \phi; \mu_0; \mu_1; \Sigma)$看作是 x 的函数，我们会发现他可以被表示为：

$$p(y = 1|x; \phi, \Sigma, \mu_0, \mu_1) = \frac{1}{1 + \exp(-\theta^T x)}$$

θ 是 ϕ，Σ，μ_0，μ_1 的函数。这确实是逻辑回归的一种形式---一个区别的算法---适用于模型 p(y=1|x)。

什么时候我们会觉得一个模型比另一个更好？一般情况下，GDA 和逻辑回归有不一样的判定边界对同样一组训练集。那一个更好呢？

我们仅仅确定 p(x|y)是多元高斯分布（Σ 已给定）那么 p(y|x)必定是一个逻辑函数。然而，反过来这个结论是不成立的；例如：p(y|x)是一个逻辑函数但不是代表 p(x|y)是一个多元高斯函数。这也表明 GDA 比着逻辑函数有着更强的模型假设能力。那也表明当假设模型是正确的时候，GDA 或更好的适合数据，也会是一个更好的模型。特别的是，当 p(x|y)确实是高斯分布，GDA 就会渐渐有效。不严格的说，对于有限制的大型数据训练集，没有什么算法比 GDA 更加好的算法。（也可以说没有其他算法可以更精确的预测 p(y|x)）。特别的是，在这个环境中也可以证明 GDA 是一个比着 logistic 回归更好的算法，更一般的说，即使对于小的训练集，我们也希望 GDA 更好。

相反的是，在一些微弱的假设中，逻辑回归函数也是很强健的和对错误的模型预测不敏感的算法。这有一些使用逻辑回归算法预测想法。例如：如果 x|y=0~Poisson（λ0），并且 x|y=1~Poissson(λ1)，这个 p(y|x)就是逻辑回归函数。逻辑回归函数在泊松分布数据上的应用很好。但是如果我们想使用 GDA 在这个数据上—并且找到一个高斯分布函数在没有高斯分布的数据上---这个结果将会不好说的，有可能 GDA 会做的更好，也有可能更差。

总结一下：GDA 给出了很好的模型预测，并且适合于大多数的数据。（需要很少的数据就可以学习的很好）当这个模型预测是对的或者是接近的对的。逻辑回归模型给出了相对较差的模型预测，并且预测出来的模型和真是数据有偏差。特别的是，当数据不是高斯分布是，在大量的预测数据条件下，逻辑回归模型几乎经常比 GDA 做的更好。鉴于这个原因，在实践中，逻辑回归比 GDA 使用的更多更经常。（一些有关联的需要考虑的因素影响模型更加适合贝叶斯模型，我们在接下来的章节中会讨论，但是贝叶斯算法仍然被认为是一个很好并且是一个很流行的分类算法。）

2 Naive Bayes　朴素贝叶斯

在 GDA 算法中，特征向量 x 是连续的、实际的值。现在我们讨论一种不一样的算法，训练集 x 是离散的值。

对于我们的激励函数的例子，考虑建造一个邮件分类系统使用机器学习。这里我们想分类这些邮件基于是否他们是未经许可的商业邮件（垃圾邮件），或者非垃圾邮件。当学习到做这些之后，我们接下来可以使我们的邮件分类器自动的分开这些垃圾邮件并且把他们放在一个隔开的邮件文件夹中。分类邮件只是一个使用界限分类的文本分类的例子。

现在我们有一个训练集（一个关于垃圾邮件和非垃圾邮件的训练集）。我们会开始我们的构建关于我们的邮件分类器通过指定关于能识别出来邮件 x 的特征。

我们会描绘一封邮件通过一个特征向量，这个特征向量的值是邮件中的单词在字典中的号码。特别的是，如果一封邮件包含了第 i 个单词，我们会令 xi=1；否则会使 xi=0.

例如，这个向量：

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{matrix}$$

就是用来表示一封邮件中包含单词“a”和“buy”，但是不包含“aardvark”“aaedwolf”或者“zygmurgy”。这个给向量编码的方式叫做词汇表法，因此这个 x 的维度就是词汇表的长度。

已经选定了我们的特征向量，现在我们想要去构建一个比较有识别力的模型。因此，我们不得不对 p(x|y)进行建模。但是如果我们有一个词汇表包含 50000 个单词，那么 x∈｛0，1｝^50000(x 是一个 50000 维 0 和 1 组成的向量)，并且如果我们想要对一个多元分布 x 精确建模那将会有 2^50000 个输出，我们需要有（2^50000-1）维的参数向量。很明显这太多了。

为了对 p(x|Y)进行建模，我们因此做一个很强烈的假设。我们会假设 Xi 和 y 是相对独立的。这个假设被叫做朴素贝叶斯假设，并且这个结果算法被叫做朴素贝叶斯分类器。例如：

如果 y=1 代表垃圾邮件；“busy”是第 2087 个单词和“price”是第 39831 个单词，然后我们会假设如果我告诉你 y=1（代表这个是垃圾邮件），然后 x2087（表示“buy”是否出现在这个信息里）对 x39831 这个值是否出现在邮件中没有任何的影响。更一般的说，这个也可以被写成 $p(x_{2087}|y) = p(x_{2087}|y, x_{39831})$。（这个并不表示 x2087 和 x39831 是相互独立的表示为 “$p(x_{2087}) = p(x_{2087}|x_{39831})$”），然而，我们仅仅是假设 x2087 和 x39831 对于 y 是相对独立的。

现在有：

$$\begin{aligned}
&p(x_1, \ldots, x_{50000}|y) \\
&\quad = p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2)\cdots p(x_{50000}|y, x_1, \ldots, x_{49999}) \\
&\quad = p(x_1|y)p(x_2|y)p(x_3|y)\cdots p(x_{50000}|y) \\
&\quad = \prod_{i=1}^{n} p(x_i|y)
\end{aligned}$$

第一个等式代表了一般的概率的表达式，第二个式子是用了 NB 假设。我们注意即使朴素贝叶斯假设是极端强烈的假设，这个结果算法对很多问题都能算出很好的结果。

我们的模型参数化 $\phi_{i|y} = 1 = p(x_i = 1 | y = 1), \phi_{i|y=0} = p(x_i = 1 | y = 0), \phi_y = p(y = 1)$。通常情况下，给定一个训练集（$x^i, y^i$）；i=1，…，m），我们能够写出下列结合极大似然函数：

$$\mathcal{L}(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

改变 ϕ_y 以便最大化这个似然函数，$\phi_{i|y=0}$ 和 $\phi_{i|y=1}$ 给了极大似然函数一个估算

$$\begin{aligned}
\phi_{j|y=1} &= \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \\
\phi_{j|y=0} &= \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \\
\phi_y &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}
\end{aligned}$$

在上面的等式中，“∧”表示“和”。这个参数有一个很自然的解释。例如，$\phi_{j|y=1}$ 是一部分的 j 没有出现的垃圾邮件。

调整好参数后，预测一个新的关于 x 的样本，我们可以简单的计算：

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

并且挑选出后验概率比较大的哪些类。

最后，我们注意到当我们提升朴素贝叶斯算法主要是针对 xi 特征变量是二值函数时，x 可以简单的取｛1，2，…，ki｝里面的任何值。这里，我们会简单的说一下创建一个关于多值的 p(xi|y)的模型而不是二值的。确实，即使一些原始的输入属性是连续的数据，他们确实共同的可以使其离散化---把他转化为一个小的离散的训练集----并且适合于朴素贝叶斯。例如：如果我们使用一些特征 xi 表示居住面积，我们可以把连续值离散化为如下形式：

Living area (sq. feet)	< 400	400-800	800-1200	1200-1600	>1600
x_i	1	2	3	4	5

因此，对于一个居住面积是 890 平方英尺的房子，我们会把他归属到 xi=3 这个里面。我们接下来会使用这些数据适应朴素贝叶斯算法，并且对 p(xi|y)进行建模使用多元正态分布像前面描述的一样。当这个原型，连续的值并不是很好的适应多元正态分布的时候，这时候使用朴素贝叶斯算法通常会得到一个更好的分类标准。

2.1 Laplace smoothing　拉普拉斯平滑

朴素贝叶斯算法正如我们所介绍的那样对大多数的问题都会做的很好，但是如果做一个小小的修改，那将会使他工作的更好，特别是对文本分类的情况。先简单的讨论一个问题使用这个算法的一般形式，然后决定怎么去修改他。

考虑垃圾邮件分类的问题，让我们做一个假设，当你完成 CS229 这门课程以后并且做的很出色在你的作业上，你决定在大约 2003 年 6 月分递交你的作业给 NIPS 会议用来出版。（NIPS 是顶尖的机器学习会议，这个递交论文的最后期限是 6 月底到 7 月初）。因为你结束讨论这个会议在你的邮件中，你在开始也用了“nips”。但是这是你的第一份 NIPS 论文，并且在这之前，你没有收到过任何的邮件里面包含“nips”这个单词；特别是“nips”没有在你的训练集中出现。假设“nips”是第 35000 个单词在字典中，你的朴素贝叶斯垃圾分类因此计算他的极大似然估计用 $\phi_{35000|y}$ 是：

$$
\begin{aligned}
\phi_{35000|y=1} &= \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1 \wedge y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}} = 0 \\
\phi_{35000|y=0} &= \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1 \wedge y^{(i)}=0\}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}} = 0
\end{aligned}
$$

也即是说因为他以前没有看到“nips”这个单词不论是在垃圾邮件还是非垃圾邮件的训练集中，他认为看到他的可能性在每个类别中都是 0.因此，当试图去判断一封包含“nips”的邮件是不是垃圾邮件，他会计算这个类的后验概率，并且得到如下结果：

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

这是因为每一个“$\prod_{i=1}^{n} p(x_i \mid y)$”包含一个 $p(x_{35000} \mid y) = 0$ 乘上以后结果就是 0.因此，我们的算法得到 0/0，并且不知道怎么去判断。

把这个问题扩大化，在统计上这是一个坏注意去估计一些事件的可能性为 0 仅仅是因为在你有限的训练集中没有出现这个数据。估算这个随机多元变量 z 的均值从｛1，2，…，k｝中取值。我们能够使用多元参数 ϕ i=p(z=i)。给定一个关于 m 的训练集和观测值｛z1，z2。。。zm｝之间独立，这个极大似然估计由下式给出：

$$
\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}{m}
$$

像我们以前看到的那样，如果我们想使用极大似然估计去预测，一些 ϕj 可能会等于 0，这是一个问题。为了防止这个现象的发生，我们可以使用拉普拉斯平滑，替代上面的估计用下列式子：

$$
\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + k}
$$

这里，我们给分子加 1，给分母加 k。注意 $\sum_{j=1}^{k} \phi_j = 1$ 依然成立（自己验证），这个是符合要求的对于我们的概率估计要求总和为 1。并且对于所有的 j 有 ϕ_j 不等于 0 成立，解决了我们估计等于 0 的问题。在某些情况下，可以被证明拉普拉斯平滑确实给出了最好的预测模型对 ϕ_j。

返回朴素贝叶斯分类器，使用上拉普拉斯平滑之后我们可以得到下面的对参数的预测：

$$
\begin{aligned}
\phi_{j|y=1} &= \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2} \\
\phi_{j|y=0} &= \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}
\end{aligned}
$$

（特别的是，通常情况下不管我们有没有使用拉普拉斯平滑对 ϕ_j，一旦我们将要对一个垃圾邮件分类系统做预测会有一个公平的牵制，因此 ϕ_j 会离 0 非常远的。）

2.2 Event models for text classication 文本分类模型

为了结束我们的生成学习算法，我们最后讨论一个模型特别适用于文本分类。然而，朴素贝叶斯分类器我们已经给出了，他对很多分类问题都可以做的很好，对于文本分类，这里有一个相关的模型也可以做的很好。

在这个特别的环境关于文本分类，朴素贝叶斯使用的是被叫做多值伯努利时间模型。在这个模型红，我们假设一封邮件的生成是第一次，同时随机决定的（根据类 p(y)）不论是垃圾邮件还是非垃圾邮件都会给你发下一封。然后，这个人发送邮件通过字典对照，决定是否包含在邮件中的第 i 个单词是否相互独立依赖概率 $p(x_i = 1 \mid y) = \phi_{i\mid j}$。因此，这个信息的可能性给出是 $p(y)\prod_{i=1}^{n} p(x_i|y).$

这里有一个不一样的模型，叫做多项事件模型。为了描述这个模型，我们将会使用一些不同的符号并且定义一些特征值来表示这些邮件。我们令 x_i 表示邮件中的第 i 个单词。因此 x_i 现在的值是取｛1，…，|V|｝，|V|是字典的长度。一封邮件有 n 个单词现在被转换为一个向量（x1，x2，…，xn）长度为 n；注意 n 可以使用不同的字母代替。例如，如果一个邮件是以“A NIPS…，”开始，那么 x1=1（“a”是字典中的第一个单词），x2=35000（如果“nips”是在字典中的第 35000 的位置）

在这个多事件模型中，我们假设垃圾邮件还是非垃圾邮件的第一层产生方式是随机的。然后这个发送者写邮件通过第一次产生 x1 从多元正态分布得到。接下来第二个单词 x2 得到和 x1 相互独立但是也是从这个多元正态分布中获得，对于剩下的都是一样，知道所有邮件中的 n 个单词都产生出来。因此，这个总体的概率是由 $p(y)\prod_{i=1}^{n} p(x_i|y).$ 决定。注意这个公式看起来想我们以前在多元伯努利事件模型中的一样，但是这里面的变量的意义和他是不一样的。特别的是 xi|y 现在是一个多项式，而不是伯努利分布。

我们模型的参数还是和以前一样是 $\phi_y = p(y)$，$\phi_{i|y=1} = p(x_j = i \mid y = 1)$（对于所有的 j）和 $\phi_{i|y=0} = p(x_j = i \mid y = o)$。注意我们已经假设 $p(x_j \mid y)$ 适合所有的 j 是一样的（例如：一个单词在邮件中分布的位置不是有 J 决定的）

如富哦我们给定一组训练集｛（x^i, y^i）；i=1，…，m｝（$x^i = (x_1^i, x_2^i, ..., x_{n_i}^i)$（这里 ni 是单词在第 i 个训练样本中。）

极大似然函数是：

$$\begin{aligned}\mathcal{L}(\phi, \phi_{i|y=0}, \phi_{i|y=1}) &= \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}) \\ &= \prod_{i=1}^{m} \left(\prod_{j=1}^{n_i} p(x_j^{(i)} | y; \phi_{i|y=0}, \phi_{i|y=1}) \right) p(y^{(i)}; \phi_y)\end{aligned}$$

最大化这个极大似然函数是的参数值：

$$\begin{aligned}
\phi_{k|y=1} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i} \\
\phi_{k|y=0} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i} \\
\phi_y &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}.
\end{aligned}$$

如果我们想要应用拉普拉斯平滑（想要得到更好的模型）当估计 $\phi_{k|y=0}$ and $\phi_{k|y=1}$，我们给分子加上 1，给分母加上一个|V|，可以得到：

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

然而未必是最好的分类算法，朴素贝叶斯分类器经常能做的特别好。他也通常是一个很好的“第一个去尝试”，给出他的单纯并且容易实现。

Part V

Support Vector Machines

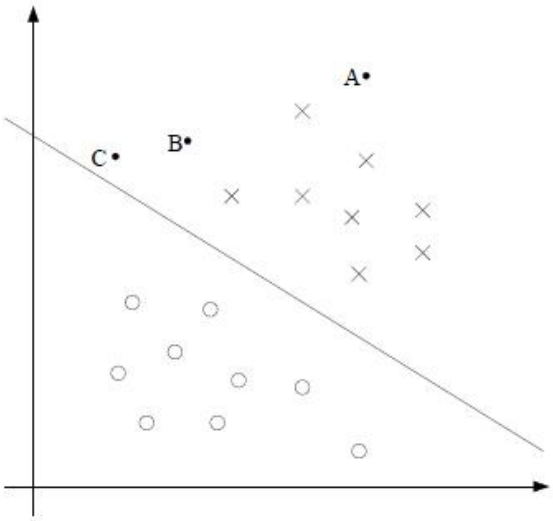
这篇讲义讲的是支持向量机（SVM）学习算法。SVMs 是最好的“非专门设计的”监督学习算法。为了讲 SVM 的故事，我们需要先讲一下分散数据的边际概念。接下来我们会说一下最优边际分类，虽然会有一点离题去讲拉格朗日对偶。我们也会说一下这个核心，给出一种支撑 SVMs 很有效的适用于高纬度的特征空间，最后我们会以能够很好实现 SVMs 算法的 SMO 结束这个故事。

1 Margins: Intuition

我们会以考虑边际开始我们的故事。本节会给出边际判断和预测的“自信力”；这些东西会正式的讲解在第三节。

考虑一下逻辑回归，概率 p(y=1|x; θ)由 hθ（x）=g（θTx）给出。我们然后会预测“1”通过输入到 x 当且仅当 $h_\theta(x) \geq 0.5$，或者是当且仅当 $\theta^T x \geq 0$。有一个正数训练样本（y=1）。$\theta^T x$ 越大，$h_\theta(x) = p(y=1 \mid x; w, b)$ 就越大，因此我们的预测标签是 1 的“自信度”也就越高。因此，正式的我们能认为我们的预测作为一个自信的模型 $y=1$ 如果 $\theta^T x \gg 0$。同样的，我们认为逻辑回归做出一个非常自信的预测对于 y=0，如果 $\theta^T x \ll 0$。给定一组训练集，如果我们能找到一个 θ 使的对于所有的 x 有 $\theta^T x \gg 0$ 当对于任意 y 有 y=1 并且对于任意的 x 有 $\theta^T x \ll 0$ 成立当对于所有的 y=0，因此这个能反映一个非常正确的分类集对于所有的训练样本。这看起来是一个好的目标去实现，并且我们很快会把这个想法实现出来用函数边际的概念。

对于一个不同的类型关于判断事实，考虑下列图形，x 代表正的训练样本，0 代表负的训练样本，一个决策边界（图中给定的这条线由 $\theta^T x = 0$ 给出的，被叫做分类超平面）在图中给出，并且有三个点被标出为 A，B 和



C

注意这个点 A 离我们的决策边界很远。如果我们被要求去预测一个值 y 就在 A 点，看起来我们应该很自信的说 y=1.相反的是，C 点离这个决策边界很近，并且他也是在我们预测的 y=1 的一边，看起来如果有一点点的改变的话这个预测值就会改变为 y=0.因此，我们相对于 C 点来说对我们在 A 点的预测更为自信。点 B 是介于这两种情况之间，更广泛的说，我们可以看到当一个点远离分类特征时候，我们会更加自信对于我们的预测值。再一次，我们认为他会更好，给定一个训练集，我们想要去找到一个决策边界能够使我们去对所有的数据做正确的和自信的预测。我们会讲到这个在之后使用几何边际的概念。

2 Notation 符号

为了使我们对 SVMs 的讨论更加简单，我们首先需要介绍一下讨论新的算法需要用的符号。我们考虑一个线性分类器对于一个二值分类问题使用标签为 y 和特征为 x。现在开始，我们使用 y∈｛-1，1｝（包括｛0，1｝）表示分类标签。同样的，我们会使用参数 w，b 来表示我们的分类器而不是使用向量 θ，接下来可以写出来我们的分类函数：

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

这里 g（z）=1 如果 z>=0；其他情况下 g(z)=-1。参数“w，b”可以使我们明确的把偏置项 b 和其他的参数分开。（我们舍弃了以前使用的令 x0=1 作为偏置项的决定）因此，b 现在扮演的角色就是以前的 θ 0，w 就是以前的[θ 1, θ 2,…, θ n]T。

注意，使用我们上面对 g 的规定，我们的分类器预测出来的值是-1 和 1，省去了需要对 y 进行大约估计的中间步骤。（那是在逻辑回归中做的事情。）

3 Functional and geometric margins（函数和几何间接）

我们把功能和几何间隔形式化。给定一个训练集（xi，yi），我们定义（w，b）相对于训练集的函数间隔：$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$

注意，如果 yi=1，这个函数间隔会变大（例如：对于我们的预测会更加自信和正确。）然后

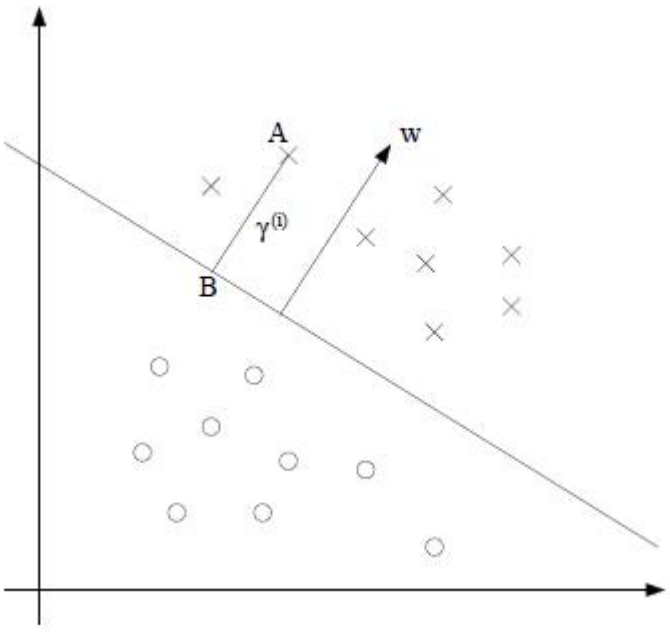
我们需要 $w^T x+b$ 是一个很大的正数。相反的，如果 yi=-1，那么这个函数间隔也会变大，那么我们需要 $w^T x+b$ 是一个很大的负数。此外，如果 $y^i(w^T x+b)>0$，那么我们对于这个训练样本的预测是正确的。（自己证明）。因此，一个大的函数间隔代表一个自信的和正确的预测。

对于一个已经给定了上述 g 的线性分类器，然而，有一个关于函数间隔的性质使的他不能作为判断是否分类完好的标准。选定函数 g，我们注意到如果我们使用 2w 替换 w，2b 替换 b，然后 $g(w^T x+b)=g(2w^T x+2b)$，这个并不会改变最终的输出结果 $h_{w,b}(x)$。也就是 g 和 $h_{w,b}(x)$ 仅仅依赖于正负而不是大小。然而，用（2w,2b）替换（w，b）也使我们的函数间隔增加了 2 倍。因此，看起来我们按规定的比例值增大 w,b，而且这也是我们的函数间隔变大但是这个改变是没有什么意义的。更直观的说，可能那样会有意义就是增加一些标准性的条件，比如像||w||2=1；也就是说，我们可能会使用（$\frac{w}{\|w\|_2},\frac{b}{\|b\|_2}$）替换（w,b），并且考虑这时候的函数间隔。之后我们会再回来讨论这个问题。

给定一个训练集 S=｛（xi，yi）；i=1，…,m｝,我们同样定义关于(w,b)的函数间隔关于 S 作为最小的关于给定训练样本的函数间隔。一样可以写出：

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

接下来,我们讨论几何间隔。考虑一下下面这张图：



这个和(w,b)相一致的决策边界如图所示，伴随着向量 V。注意 w 是和分离超平面垂直的。考虑这个点 A，代表输入的训练样本 xi 的标签 yi=1.他到决策边界的距离是线段 AB。

我们怎么样才能找到γ^i的值？$\frac{w}{||w||}$是单位向量的长度方向和 w 一样。因为 A 代表 xi，因此我们发现 B 由$x^i - \gamma^i * \frac{w}{||w||}$给出。但是这个点在决策边界上，并且所有在决策边界上的点 x 都满足$w^T x + b = 0$。因此：

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

解出γ^i：

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

$||w||^2$

这个对于算出对于一个在图中 A 点的正的训练集，在决策边界的正方向一侧是好的。更一般的，我们定义（w，b）得到几何间隔使用训练集（xi，yi）

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

注意，如果||w||=1，这个函数间隔等于几何间隔--------因此这给出了一种关于两种不一样的表示之间的联系。并且，重新调节参数几何间隔是不变的，也就是说，如果我们用（2w，2b）替换（w，b）这个几何间隔是不变的。这个迟早会用到的。特别的，因为这个不变性，当试图去找到合适的 w 和 b 去训练数据时候，我们可以假设改变任意大小对 w 而不改变其他重要的东西；例如，我们可以要求||w||=1 或者||w1||=5 或者|w1+b|+|w2|=2，这些都能适应简单的改变 w 和 b。

最后给定一个训练集 S=｛（xi，yi）；i=1，…,m｝，我们定义关于 S 的（w，b）的几何间隔是最小的在所有的几何间隔中。$\gamma = \min_{i=1,\ldots,m} \gamma^{(i)}$。

4 The optimal margin classifier 最佳间隔分类器

给定一个训练集，看起来像从我们以前讨论的里面拿来的一个必需品去试图找到一个决策边界最大化这个间隔，因此这个会反映一个非常明确的对训练集的预测和适合的参数。特别的，这会导致分开正的训练样本和负的训练样本用一个间隔。

现在开始，我们假设我们给定一个训练集是可线性分隔的。也就是说那是可能的分开正的和负的训练样本使用分类超平面。我们怎么才能找到这个分类超平面使的这个几何间隔最大化？我们能提出下面的最佳化问题：

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

也就是说，我们想要最大化γ，对于每一个训练样本都使用的最小的γ。此外条件||w||=1约束使的几何间隔等于函数间隔，因此我们能保证对于所有的几何间隔都有最小的γ。因此，解决这个问题会归结于求（w，b）使的几何间隔最大化在训练集中。

如果我们想解决上面这个优化问题，我们已经做了的。但是条件“||w||=1”是向下的（非凸的），并且这个问题不能以任何形式用标准优化软件去解决。因此试着去改变这个问题成为一个好一点的。考虑下面这个式子：

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

这里，我们想要最大化$\frac{\hat{\gamma}}{\|w\|}$，受制于函数间隔的所有条件都是在最小化$\hat{\gamma}$。因此，这个几何和函数间隔是通过$\gamma = \frac{\hat{\gamma}}{\|w\|}$联系，这会给我们一个我们想要的答案。此外，我们还摆脱了||w||=1 的约束。负面的影响是我们现在有一个非凸的目标函数：$\frac{\hat{\gamma}}{\|w\|}$，并且我们仍旧没有得到任何的现成的软件可以去解决这类优化问题。

继续。重新回到我们以前的讨论我们能加上一个任意的缩放约束对于 w 和 b 而不改变其他东西。这是我们现在要用的主要思想。我们接下来会介绍这个对 w，b 的缩放约束对于函数间隔在训练集必须是 1 的条件下：$\hat{\gamma} = 1.$

因此，使 w 和 b 乘上一些约束结果导致这个函数间隔同样乘以相同的常熟，这的确是一种缩放约束并且能被适用通过改变 w 和 b。把这些加到我们上面的问题中，并且注意最大化$\frac{\hat{\gamma}}{\|w\|} = \frac{1}{\|w\|}$是同样的事情和最小化$\|w\|^2$。我们现在有下面的优化问题：

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m$$

我们现在已经改变这个问题到另一个形式能够被有效解决的形式。上面的是一个优化问题带有凸二次方程和仅仅是线性缩放的。他的解决给了我们一个最优间隔分类器。这个最优问题能够被解决使用商业的二次规划代码。

然而现在我们说的问题已经被解决了，我们接下来要做的就是离题讨论一下对偶。这个会引导我们的最优问题到对偶形式，这将会扮演一个重要角色在允许我们使用 k 均值去得到最优间隔分类器去有效的解决问题在更高维的空间中。这个对偶形式也会允许我们导出一个有效的算法对于解决最优问题能够比商业二次规划代码解决的更好。

5 Lagrange duality 拉格朗日对偶

让我们先临时的把 SVMs 和最大化间隔分类器放一边，先考虑一下解决最优选择问题。考虑如下形式的问题：

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & h_i(w) = 0, \quad i = 1, \ldots, l. \end{aligned}$$

你们又会说拉格朗日乘数的方法怎么能用来解决这类问题。（不要着急如果以前你没有见到过的话。）在这种方法中，我们定义这个拉格朗日函数为：

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

这里，$\beta_i's$ 叫做拉格朗日乘数。我们之后会是他的偏导数等于 0：

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0; \quad \frac{\partial \mathcal{L}}{\partial \beta_i} = 0,$$

并求出 w 和β。

在本节中，我们会归纳这个为最优选择问题我们把不平等的最优当作平等的。由于时间限制，我们本节不会真的去讲拉格朗日对偶的理论推导，但是我们会给出这个主要的定义的结论，我们之后会应用到我们的最优选择问题中。

考虑下面这个问题，我们叫做原始最优问题：

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \ldots, k \\ & h_i(w) = 0, \quad i = 1, \ldots, l. \end{aligned}$$

为了解决他，我们开始定义一般化的拉格朗日函数：

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w).$$

这里，$\alpha_i's$ 和 $\beta_i's$ 都是拉格朗日乘数，考虑下面这个等式：

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta).$$

这里“p”代表“一般化”。先给定一些 w。如果 w 违反了任何的一般化限制（例如：如果对于任何的 gi(w)>0 或者 hi(w)!=0 对于 i）然后你要能证明：

$$\theta_{\mathcal{P}}(w) = \max_{\alpha,\beta:\alpha_i\geq 0} f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w) \quad (1)$$

$$= \infty. \quad (2)$$

相反的，如果限制条件非常适合一个特别的 w，有$\theta_{\mathcal{P}}(w) = f(w)$。因此：

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise.} \end{cases}$$

θ_p 和我们问题中的适合一般化模型的所有的 w 的值相同，并且是正的如果限制条件被违反了，因此，如果我们考虑这个最小化问题：

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha,\beta:\alpha_i\geq 0} \mathcal{L}(w,\alpha,\beta),$$

我们看到这是同一个问题，一般化问题。对于之后的使用，我们也定义这个类的最优值为 $p^* = \min_w \theta_{\mathcal{P}}(w)$，我们交这个值为一般化问题的最优值。

现在，我们考虑这个有一点点不一样的问题.我们定义：

$$\theta_{\mathcal{D}}(\alpha,\beta) = \min_w \mathcal{L}(w,\alpha,\beta).$$

这个“D”代表“对偶”。注意我们定义 θ_p 最优化这个值使用α和β，这里最优化是依赖于 w。

我们现在提出对对偶最优化问题的讨论：

$$\max_{\alpha,\beta:\alpha_i\geq 0} \theta_{\mathcal{D}}(\alpha,\beta) = \max_{\alpha,\beta:\alpha_i\geq 0} \min_w \mathcal{L}(w,\alpha,\beta).$$

这个整好和我们上面说的一般化问题一样，出来这个求最值问题换了一下。一个是求最大一个是求最小。我们定义对偶问题的最优值为 $d^* = \max_{\alpha,\beta:\alpha_i\geq 0} \theta_{\mathcal{D}}(w).$

这个一般化问题和对偶问题有什么联系呢？他可以简单的用下面公式表示：

$$d^* = \max_{\alpha,\beta:\alpha_i\geq 0} \min_w \mathcal{L}(w,\alpha,\beta) \leq \min_w \max_{\alpha,\beta:\alpha_i\geq 0} \mathcal{L}(w,\alpha,\beta) = p^*.$$

然而，在确定的情况下，我们有 $d^* = p^*$，因此我们能解决对偶问题代替原始的问题。我们看一下这些条件.

假设 f 和 gi’s 都是给定的，并且 hi’s 是仿射变换。进一步假设假设条件 gi 是可行的；这意味着存在 w 使得 gi(w)<0 对于所有的 i 都成立。

在我们上面的假设中条件下，必定存在 w^*, α^*, β^* 以便于 w^* 是原是问题的最优解，α^*, β^* 是对偶问题的最优解，并且 $p^* = d^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$。更进一步，$w^*, \alpha^*, \beta^*$ 适用于 KKT

条件，如下所示：

$$\frac{\partial}{\partial w_i}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0,\ \ i=1,\ldots,n \tag{3}$$

$$\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0,\ \ i=1,\ldots,l \tag{4}$$

$$\alpha_i^* g_i(w^*) = 0,\ \ i=1,\ldots,k \tag{5}$$

$$g_i(w^*) \le 0,\ \ i=1,\ldots,k \tag{6}$$

$$\alpha^* \ge 0,\ \ i=1,\ldots,k \tag{7}$$

更进一步说，如果 w^*,α^*,β^* 使用于 KKT 条件，那也就是一种解决原始化和对偶问题的解。

我们看一下等式（5），等式（5）被叫做 KKT 对偶互补条件。特别的是他反映的是如果 $\alpha_i^* > 0$ 那么 $g_i(w) = 0$。（例如："$g_i(w) \le 0$"条件是激活的，意味着他支持这个等式而不是不等式。）然后，这个会是这个关键对于表示 SVM 仅仅有一小部分对于"支持向量"成立；这个 KKT 对偶互补条件也会给我们测试当我们讨论 SMO 算法时候。

6 Optimal margin classifiers 最有间隔分类

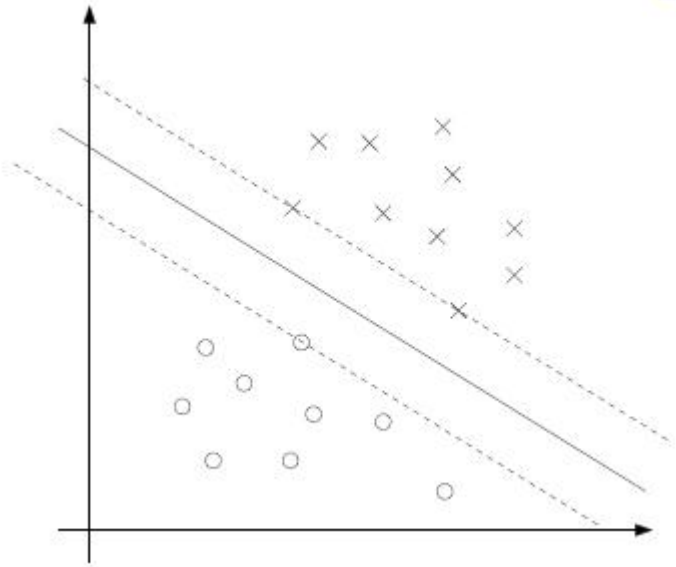
之前，我们给出了下列优化问题去寻找这个最有间隔分类器：

$$\begin{aligned} \min_{\gamma,w,b}\ & \frac{1}{2}||w||^2 \\ \text{s.t.}\ & y^{(i)}(w^T x^{(i)} + b) \ge 1,\ \ i=1,\ldots,m \end{aligned}$$

We can write the constraints as

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \le 0.$$

我们有一个约束对于每一个训练样本。注意在 KKT 对偶互补条件中，我们有 $\alpha_i > 0$ 仅仅对训练样本哪些有函数建个整好对应这个。（例如：这些相对应的约束条件支持等式 $g_i(w) = 0$）。考虑下面的图像，最大间隔分割平面是这个实线。



这些最小间隔的点整好是哪些靠近决策边界的那些；这里有是那个点（一个正的和两个负的样本）整好落在和决策边界平行的虚线上。因此，只有三个α_i's---也就是说这三个点相对应的训练样本----将不会等于 0 在最优化解决中对于我们的最优化问题。这三个点叫做支持向量在这个问题中。这个事实是这个支持向量的数量可以比着训练集小很多。

继续，朝前看，我们提升这个问题为对偶形式，一个主要的注意就是关注我们试图写我们的算法用内积的形式$\langle x^{(i)}, x^{(j)} \rangle$在两个输入特征空间中间。这个事实我们能表示出我们的算法用内积形式将会是我们应用于标志性内核的主要。

当我们重构拉格朗日函数对于我们的最优化问题，我们有：

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}(w^T x^{(i)} + b) - 1 \right]. \qquad (8)$$

注意这里仅仅只有 α i 而没有 β i 作为拉格朗日乘数，因为这个问题只有不等式限制。

我们找到这个问题的对偶形式。为了这样做，我们需要收线最小化 L（w，b，α）用 w 和 b，得到θ_D，我们将会设置 L 对 w 和 b 的偏导数等于 0。

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

这表明：

对 b 求偏导得到：

$$\frac{\partial}{\partial b} \mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0.$$

如果我们对照等式（9）的定义并且带回到拉格朗日函数中并且化简我们得到：

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j(x^{(i)})^T x^{(j)} - b\sum_{i=1}^{m}\alpha_i y^{(i)}.$$

但是根据公式（10），最后一项必须为 0，因此

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j(x^{(i)})^T x^{(j)}.$$

重新说我们得到上面这个等式通过最小化 L 使用 w 和 b。把这些放在一起和 $\alpha_i \geq 0$ 还有公式（10），我们得到下面的对偶优化问题：

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

你同样需要证明这个条件对于 p*=d*和 KKT 条件支持这个最适合的在我们的优化问题中。因此我们可以解决对偶代替解决原始化问题。特别的是，在上面对偶问题中，我们有一个最大化问题参数是 α's。我们之后会使用这个特别的算法我们将会用来去解决对偶问题，但是如果我们能够解决它，然后我们能使用等式（9）去带回并且找到这个最优的 w's 最为 α's 的函数。找到 w*之后，通过考虑最优化问题，可以直接计算出最优值对于偏置项 b

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T}x^{(i)} + \min_{i:y^{(i)}=1} w^{*T}x^{(i)}}{2}. \qquad (11)$$

在进一步讨论之前，我们先详细的看一下等式（9），给出了 w 的最优值使用 α 。假设我们找到适合我们的模型参数对于一个训练样本，并且现在想要去做一个预测对于一个新的输入 x。我们将会计算 $w^T x + b$ 并且假设 y=1 有且仅有这个值比〇大。但是使用（9），这个值能写成：

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}\right)^T x + b \qquad (12) \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x\rangle + b. \qquad (13) \end{aligned}$$

因此，吐过我们已经找到了 α's，为了做一个预测，我们必须要计算一个值仅仅依赖与内积在 x 和训练集中的点。更进一步，我们可以看到 α's 将会是 0 除了支持向量外。因此，在和

中的大部分的项都会是 0，并且我们真切的想要找到这个内积在 x 和支持向量之间在我们的计算（13）中并且做出我们的预测。

通过测设最有问题的对偶形式，我们得到重要的在洞察在这个问题结构中，并且能够写出整个算法用内积的形式在输入特征向量之间；在下一节中，我们会探索这个性能去适应于内核对于我们的分类问题。这个结果算法，支持向量机将会是有效的学习在更高维度的空间。

7 Kernels 内核

回到之前我们对于线性回归的讨论，我们有一个问题关于输入 x 即房子的面积，并且我们考虑表现回归使用特征 x，x2 和 x3 去得到一个三次函数。为了区分这两个变量集，我们叫“原始”的输入值作为问题的输入属性（在这个问题中为 x，房子的面积）。当映射到一些新的数量级输入到学习算法中，我们将会叫那些新的数量为输入特征。（不幸的是，不同的作者使用不同的形式去描述这两件事情，但是我们会试着使用一贯的术语在本讲义中）我们也会使用Φ表示特征图，那些表示特征属性的图集。例如：在我们的例子中，我们有：

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

相比较适用于 SVMs 使用原始输入属性 x，我们更倾向于使用特征 ϕ(x)去学习。为了这样做，我们仅仅需要转变我们之前的算法就好，并且替换每一个 x 使用 ϕ(x)。

由于这个算法能够被完整的用内积<x,z>表示，这就意味着我们可以替换所有的内积用 <ϕ(x), ϕ(z)>.特别的是，给定一个特征图 ϕ，我们定义这个相应的内核为：

$$K(x,z) = \phi(x)^T \phi(z).$$

之后，在我们的算法中碰到的没一个<x,z>我们都能够简单的替换使用 K(x,z)，并且我们的算法现在学习使用特征 ϕ。

现在，给定 ϕ，我们能够简单计算出 K(x,z)通过找到 ϕ(x) 和 ϕ(z)并且做他们的内积。但是更有意思的是通常情况下 K(x,z)都是很容易计算的，即使 ϕ(x)他本身可能是很难计算的（可能因为他是特别高维的向量）在一些设置中，通过使用我们的算法作为一个有效的方法去计算 K(x,z)，我们能得到 SVMs 去学习在高维的特征空间中使用给给定的 ϕ，但是除了已经明确的知道或者代表向量 ϕ(x)。我们看一个例子。假设 x，z∈Rn，并且 $K(x,z) = (x^T z)^2$

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

我们也能写为：

因此，我们看 $K(x,z)=\phi(x)^T\phi(z)$，这个特征映射 ϕ 已经给定通过

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}.$$

注意当计算高维的 ϕ(x)需要 O（n2）时间，找到 K(x,z)仅仅需要 O(n)的时间-------线性的在输入属性的纬度。

对于一个相关内核，同样考虑：

$$\begin{aligned} K(x,z) &= (x^Tz+c)^2 \\ &= \sum_{i,j=1}^{n}(x_ix_j)(z_iz_j)+\sum_{i=1}^{n}(\sqrt{2c}x_i)(\sqrt{2c}z_i)+c^2. \end{aligned}$$

这个相对应的特征映射为：

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}$$

参数 c 控制这个相关系数 xi 和 xixj 之间。

更宽泛的说，内核 $K(x,z)=(x^Tz+c)^d$ 相对于到 $\binom{n+d}{d}$ 空间的特征映射，对应于每一个 xi1，xi2….xik 适用于顺序 d。然而，尽管在 $O(n^d)$ 维空间中计算 K(x,z)所用时间仍然仅仅是 O(n)，并且因此我们从不需要去明确的表示特征向量在非常高维的特征空间中。

现在，讨论一个稍微不同的观点对于内核。直观上（有一些东西是错误的在直觉上，但是

不要介意），如果φ(x)和φ(z)挨的特别紧，我们可能会期望 $K(x,z) = \phi(x)^T\phi(z)$ 非常大。相反的如果φ(x)和φ(z)离得比较远------也就是说二者差不多正交-----那么 $K(x,z) = \phi(x)^T\phi(z)$ 将会非常小。因此，我们可以认为 K(x; z)作为φ(x)和φ(z)之间距离的一种测量，或者说 x 和 z 有多相似。

给出这个直观的判断，假定对于一些学习算法你正在研究的，你已经提出一些函数 K(x; z)你认为可能是一个可行的测量关于 x 和 z 有多相似。例如：假设你选择

$$K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$$

这是一个公平的测量方法关于 x 和 z 有多相似，并且接近于 1 当 x 和 z 接近时，接近于 0 当 x 和 z 离得非常远的时候。我们是否能使用这个 K 的定义作为内核在 SVM 中?在这个特别的例子中,答案当然是 yes.(这个内核叫做高斯内核,并且接近于无穷纬度的特征映射φ)但是更一般的说，给定一些函数 K，我们怎么样才能识别他是否是一个有效的内核；例如，我们怎么才能知道是否存在特征映射φ使得 $K(x,z) = \phi(x)^T\phi(z)$ 对于所有的 x 和 z?

假设现在 K 是一个有效的内核非常接近于一些特征映射φ。现在，考虑一些有限的 m 个点的集合｛x1，x2….xm｝并且组成一个矩形，m*m 的矩阵 K 被定义以便于使他的(I,j)输入被给出由 $K_{ij} = K(x^{(i)}, x^{(j)})$。这个矩阵被叫做内核矩阵。注意我们已经介绍过这些符号并且使用 K 去代表这个内核函数 K(x，z)和内核矩阵 K，由于他们明显接近的关系。

注意，如果 K 是一个有效的内核，那么 $K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T\phi(x^{(j)}) = \phi(x^{(j)})^T\phi(x^{(i)}) = K(x^{(j)}, x^{(i)}) = K_{ji},$ 并且因此 K 必须是对称的。更多的，用φk(x)代表第 K 个相同的向量φ(x)，我们发现对于任意的向量 z，我们有：

$$\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \\
&\geq 0.
\end{aligned}$$

倒数第二步使用相同的诀窍想你在问题集 1Q1 中看到的那样。由于 z 是任意的，这表明 K

是正的（K>=0）

至此，我们已经证明 K 是否是一个有效的内核（例如：如果他接近于一些特征映射 ϕ），之后这个接近的内核矩阵 K∈Rm*m 是对称的正的。更一般的说，不重要的但是有效的的条件对于说明 K 是一个有效的内核。下面的结果是由于 Mercer。

Mercer 定理：令 K： $R^n * R^n \to R$ 已经给定。然后为了使 K 是一个有效的内核，那是重要的也是有效的对于所有的｛x1,x2…xm｝（m<无穷），这个相对应的内核矩阵是对称半正定的。

给定一个函数 K，除了试图找一个特征映射 ϕ 接近他的，这个定理因此给出了另一种方法用来测试是否他是一个有效的内核。你将会更多的机会接触这些在问题集 2。

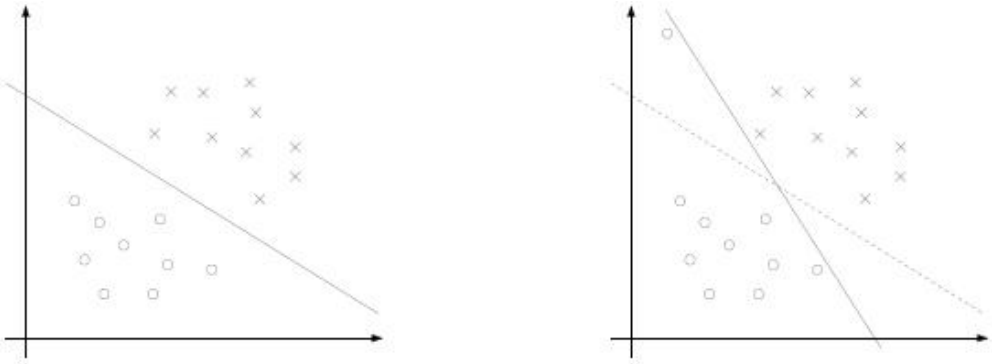
课堂上，我们简明的讨论关于几个其他关于内核的例子。例如：考虑这个数字识别问题，给定一副手写体数字（0-9）的图像（16*16 像素），我们必须辨识出是那一个数字。使用任何一个简单的多项式内核 K（x，z）=（xTz）d 或者高斯内核，SVMs 能够得到相当好的表现对于这个问题。这个是非常惊奇的由于这个输入属性 x 是一个 256 维的向量关于图像的像素值，并且这个系统没有对于视觉的只是，乃至于那些像素在其他的前面。另一个例子我们简单说一下在课堂上是如果类 x 我们试图去分类的是字符串（也即是说，x 是一列氨基酸，仅仅的和蛋白质在一起），然后他看起来很难去建模一个有效的“小的”数据集关于特征对于大多数的学习算法，特别是如果不同的串有不同的长度。然而，考虑使 ϕ(x)为一个特征向量包含没一个在 x 中出现的。如果我们考虑用英文，那么就有 26 个这样的串。因此 ϕ（x）是一个 26 维的向量；甚至对于 K，这可能太大了对于我们高效的工作。然而，使用串匹配算法，他是有可能的去高效的计算 $K(x,z) = \phi(x)^T \phi(z)$,因此我们可以暗中的计算在 26 维的特征空间中，但是不明显的在这个空间中计算特征向量。

这个关于内核的应用对于支持向量机应该已经很明白并且因此我们不会过多的讨论他在这里。记住这个内核的注意有更重要和广泛的应用比着 SVMs。特别的，如果你有任何学习算法你能够写出呢个内积的形式<x,z>表示 K 是一个内核，你可以“魔法的”允许你的算法去高效的工作在更高纬度的特征空间中接近于 K。例如：内核映射能够适用于这个感知机去得到一个内核感知机算法。大多数我们之后会看到的算法都经得起检验对于这种方法，这个算法已经广泛传播为“内核追踪”。

8 Regularization and the non-separable case

规则化和不可分的情况

SVM 的推导已经至今已经提出的是假设这个数据是线性可分的。当数据映射到一个高纬度的特征空间通过 ϕ 做一般的增长这个似然函数使这个数据可分，我们不能保证这种方法总是能够成功。因此，在一些情况下他不是明显的能找到一个分割平面我们正好想要的，由于对一些超过边界的点比较敏感。例如，下面左边的图表明一种优化矩阵分类器，并且当一个单一异常值增加进来之后在这个左上角，他使得这个决策边界做了一个大幅度的转动，并且这个结果分类器有一个更小的边距。



为了使这个算法适用于非线性可分的数据集并且对这个异常值不是特别的敏感，我们形式化我们的优化项（使用 l1 正则化）如下：

$$\begin{aligned}\min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m.\end{aligned}$$

因此，例子现在允许有函数间隔小于 1 并且如果一个样本的函数间隔是$1-\xi_i$，我们会给出一个损失项为$C\xi$。这个参数 C 控制这个相关权重在成对的目标之间使的$||w||^2$最大（我们能看到的是使这个间隔最小）并且确保这个大部分的样本的函数间隔至少是 1。

像以前一样，我们能定义这个极大似然函数：

$$\mathcal{L}(w,b,\xi,\alpha,r) = \frac{1}{2}w^T w + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left[y^{(i)}(x^T w + b) - 1 + \xi_i\right] - \sum_{i=1}^{m} r_i\xi_i.$$

这里，$\alpha_i^{'}s$和$r_i^{'}s$是我们的拉格朗日乘数（都大于等于 0）我们不再计算这个导数，但是之后设定这个函数对 w 和 b 的导数为 0，把他们代回到前面，并且化简，我们得到下面的式子：

$$\begin{aligned}\max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m}\alpha_i y^{(i)} = 0,\end{aligned}$$

像之前一样，我们同样有 w 能够被表达使用$\alpha_i^{'}s$的形式如在式子 9 中给出的那样，因此在解决这个对偶问题之后，我们能够继续使用式子 13 去做我们的预测。注意，意外的是，在增加 L1 正则化项之后，这个对偶问题仅有的变化是最初的约束条件αi≥0 现在变为 0≤αi≤C。对 b*的计算也相应的改变（式子 11 不再适用。）；看下面一节的评论。

同样的，KKT 对偶互补条件是：

$$\alpha_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1 \quad (14)$$
$$\alpha_i = C \Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1 \quad (15)$$
$$0 < \alpha_i < C \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1. \quad (16)$$

现在，所有的剩下的就是得到一个算法对于实际解决这个对偶问题，也即是我们下一节所要做的事情。

9 The SMO algorithm 序列最小优化算法

序列最小优化算法，被 John Platt 提出的，给出了一种效率的方法去解决对偶问题。一部分程度上为了刺激 SMO 算法，一部分因为他非常有趣在他自己的权力，我们离题讨论一下相关的算法。

9.1 Coordinate ascent　并列上升

考虑试着去解决无约束的优化问题 $\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$，这里我们认为 W 是关于参数 α_i's 的函数，并且现在忽略任何的联系在这个问题和 SVMs 之间。我们已经看过两个优化算法，梯度上升和牛顿法。新的算法我们将要在这里讨论的被叫做并列上升：

Loop until convergence: {

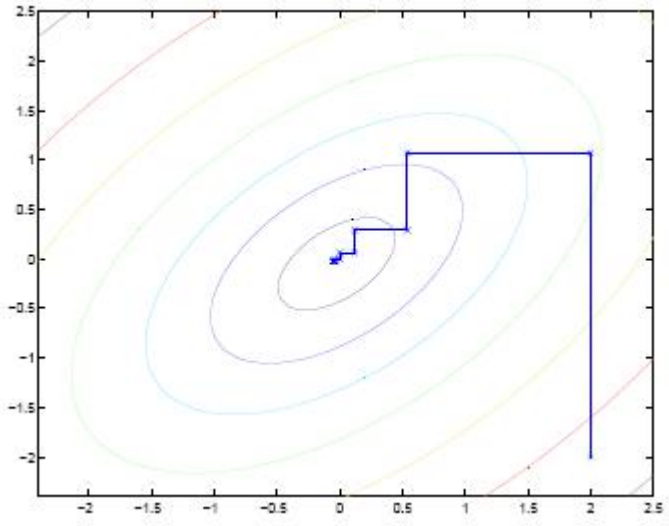
　For $i = 1, \ldots, m$, {

　　$\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$.

　}

}

因此，在这个循环的最里层，我们会保持所有的参数出来 α i 不变，并且重复优化 W 仅仅使用 α i。这里提出来的方法，这个循环的最里面一层重复优化这个函数使用 α i
当这个函数 W 能够被求出最大值在这个循环的最里层，这个并列上升算法能被是为一种非常有效率的算法。下面这张图说明了这个算法是怎么工作的：



这个图中的椭圆是这些二次函数我们想要去优化得到的。并列上升初始化为（2，-2），在图中描绘出来他的行进路线到全局最大值。注意在每一步中，并列上升行走一部他的路线都平行与一个核心值，因此每一步只有一个参数被优化。

9.2 SMO

我们结束对 Svms 的讨论用 SMo 算法。一些细节将会被留做家庭作业，其他的你们可以参考这个论文课外找到的。

下面就是这个优化问题我们想要去解决的：

$$\max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \tag{17}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \tag{18}$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \tag{19}$$

我们有 α i's 适用于等式（18-19）。现在假设我们 α 2…. α m 是固定的，并且对问题做并列上升和迭代优化这个目标用 α 1。我们能有任何进步吗？

答案当然是否定的。因为等式 19 确保：

$$\alpha_1 y^{(1)} = -\sum_{i=2}^{m} \alpha_i y^{(i)}.$$

或者，通过对等式化简使用 y1 我们可以得到：

$$\alpha_1 = -y^{(1)} \sum_{i=2}^{m} \alpha_i y^{(i)}.$$

（这一步使用这个结论 y1∈｛-1，1｝，因此（y1）^2=1）因此，α 1 是确定的参数可以通过其他的参数表示出来，并且如果我们想要保持 α 2…α m 固定，我们也不能对 α 1 做任何的改变出来违反等式 19 在这个优化问题中。

因此如果我们想要更新一些 α i 我们必须更新至少他们中的两项以便于保证能服从这个约束条件。这便催生出来 SMO 算法，如下所示：

Repeat till convergence {

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

}

为了测试这个算法的收敛性，我们可以检测是否 KKT 条件（14-16）是否适用在使用 tol。这里，tol 是这个收敛容差参数，并且通常被设置在 0.01 到 0.001 之间。

这个关键的原因说 SMO 是一个有效率的算法是这个更新对 α i 和 α j 能够被很有效率的计算出来。我们接下来简单的描述一下这个算法的效率更新。

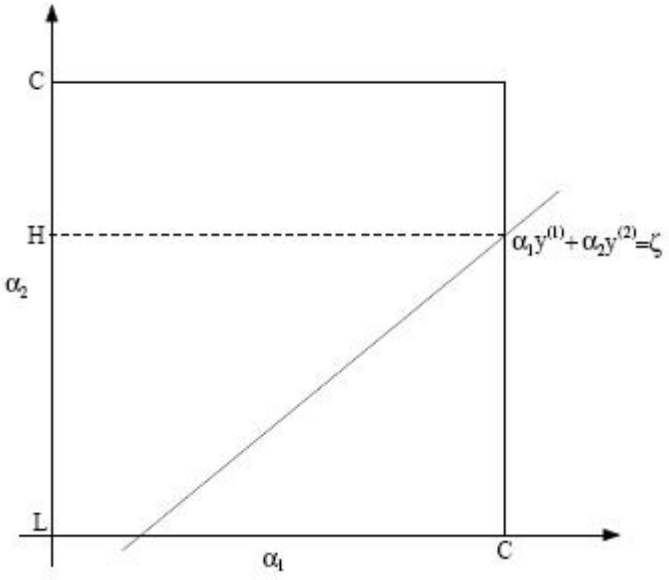
我们说我们当前的一些设置对于 α i's 都能适用于这个约束条件（等式 18-19），并且假设我们保持 α 3… α m 是固定的，并且想要去优化 W 使用 α 1 和 α 2。从等式 19 中，我们要求

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum_{i=3}^{m} \alpha_i y^{(i)}$$

由于这个等式的右边是固定的，我们可以使他被定义为固定参数 ζ：

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta.$$

我们能够画出下面的约束条件对于 α 1 和 α 2



从等式 18 中我们知道 α 1 和 α 2 必须在[0，C]*[0，C]之间。画出 $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta.$ 这条 α 1 和 α 2 必须符合的直线。注意，从这些约束条件中，我们知道 L≤ α 2≤H；另一方面，（α 1， α 2）不能同时的适用于这个盒子和这个直线约束。这个例子中，L=0。但是取决于这个直线 $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta.$，这个通常不是这个问题的重点，但是更一般地说，会有一些下界 L 和上界 H 在许可的值对于 α 2 会确保 α 1 和 α 2 刚好在这个约束条件中。

使用等式 20，我们也能写出 α 1 为 α 2 的函数：$\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}$
（自己检查这个推论，我们会再一次使用这个定理 y1∈｛-1，1｝因此（y1）^2=1）因此这个目标函数 W（ α ）能够被写为：

$$W(\alpha_1, \alpha_2, \ldots, \alpha_m) = W((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \ldots, \alpha_m).$$

把 α 3… α m 固定，你能够证明这个仅仅是一个关于 α 2 的二次函数。例如：这个能够被表示为如下形式$a\alpha_2^2 + \bar{b}\alpha_2 + c$对于大多数的适当的 a，b，c。如果我们忽略这个等式 18 的约束条件，那么我们能简单的最大化这个二次函数通过设置他的导数为 0 并且解出来。我们会使 $\alpha_2^{new,\bar{u}nclipped}$ 表示这个结果值对于 α 2。你同样可以信服你自己如果我们代替我们想要最大化 W 使用 α 2 但是受限制于这个约束条件，之后我们能够妨碍西安这个结果值简单优化通过使用 $\alpha_2^{new,\bar{u}nclipped}$ 并且可以转化到下面的约束条件中

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } L \leq \alpha_2^{new,unclipped} \leq H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$$

最后，找到这个 α_2^{new}，我们能够使用等式 20 去代会到前面并且找到这个最优值对于 α_1^{new}。
这里还有更过的细节我们能够很简单的证明但是我们会留给你自己去读在Platt论文：One is the choice of the heuristics used to select the next α_i, α_j to update；另一个是怎么更新b随着SMO算法运行。