

12、支持向量机(Support Vector Machines)

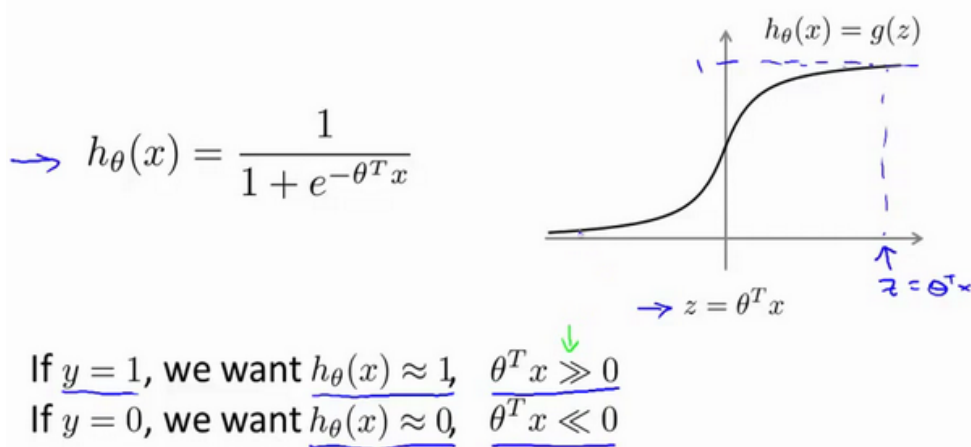
12.1 优化目标

参考视频: 12 - 1 - Optimization Objective (15 min).mkv

到目前为止,你已经见过一系列不同的学习算法。在监督学习中,许多学习算法的性能都非常类似,因此,重要的不是你该选择使用学习算法 **A** 还是学习算法 **B**, 而更重要的是,应用这些算法时,所创建的大量数据在应用这些算法时,表现情况通常依赖于你的水平。比如: 你为学习算法所设计的特征量的选择, 以及如何选择正则化参数, 诸如此类的事。还有一个更加强大的算法广泛的应用于工业界和学术界, 它被称为支持向量机(**Support Vector Machine**)。与逻辑回归和神经网络相比, 支持向量机, 或者简称 **SVM**, 在学习复杂的非线性方程时提供了一种更为清晰, 更加强大的方式。因此, 在接下来的视频中, 我会探讨这一算法。在稍后的课程中, 我也会对监督学习算法进行简要的总结。当然, 仅仅是作简要描述。但对于支持向量机, 鉴于该算法的强大和受欢迎度, 在本课中, 我会花许多时间来讲解它。它也是我们所介绍的最后一个监督学习算法。

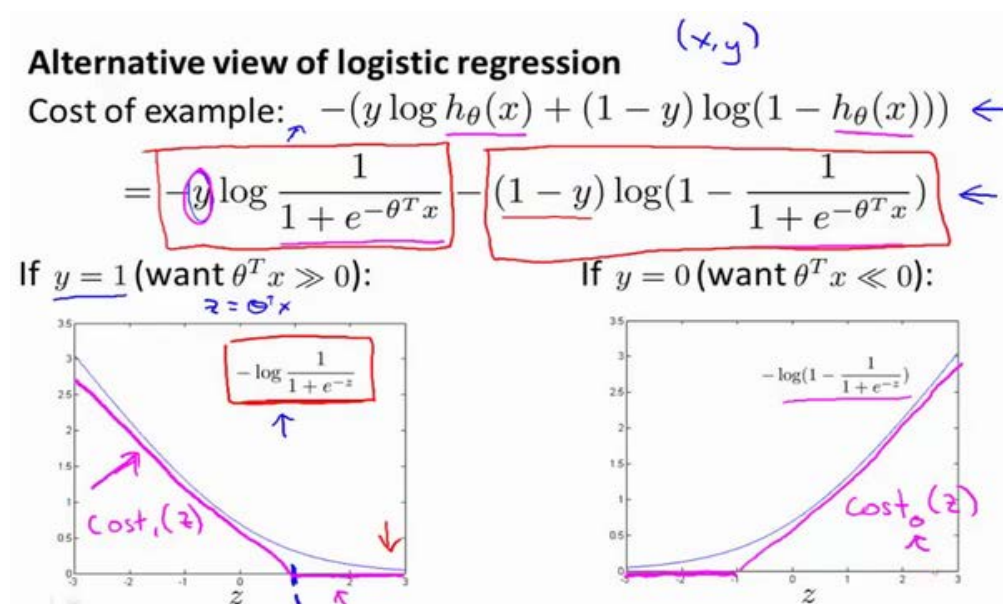
正如我们之前开发的学习算法, 我们从优化目标开始。那么, 我们开始学习这个算法。为了描述支持向量机, 事实上, 我将会从逻辑回归开始展示我们如何一点一点修改来得到本质上的支持向量机。

Alternative view of logistic regression



那么, 在逻辑回归中我们已经熟悉了这里的假设函数形式, 和右边的 S 型激励函数。然而, 为了解释一些数学知识. 我将用 z 表示 $\theta^T x$ 。

现在考虑下我们想要逻辑回归做什么：如果有一个 $y = 1$ 的样本，我的意思是不管是在训练集中或是在测试集中，又或者在交叉验证集中，总之是 $y = 1$ ，现在我们希望 $h_{\theta}(x)$ 趋近 1。因为我们想要正确地将此样本分类，这就意味着当 $h_{\theta}(x)$ 趋近于 1 时， $\theta^T x$ 应当远大于 0，这里的 \gg 意思是远远大于 0。这是因为由于 z 表示 $\theta^T x$ ，当 z 远大于 0 时，即到了该图的右边，你不难发现此时逻辑回归的输出将趋近于 1。相反地，如果我们有另一个样本，即 $y = 0$ 。我们希望假设函数的输出值将趋近于 0，这对应于 $\theta^T x$ ，或者就是 z 会远小于 0，因为对应的假设函数的输出值趋近 0。



如果你进一步观察逻辑回归的代价函数，你会发现每个样本 (x, y) 都会为总代价函数，增加这里的一项，因此，对于总代价函数通常会有对所有的训练样本求和，并且这里还有一个 $1/m$ 项，但是，在逻辑回归中，这里的这一项就是表示一个训练样本所对应的表达式。现在，如果我将完整定义的假设函数代入这里。那么，我们就会得到每一个训练样本都影响这一项。

现在，先忽略 $1/m$ 这一项，但是这一项是影响整个总代价函数中的这一项的。

现在，一起来考虑两种情况：

一种是 y 等于 1 的情况；另一种是 y 等于 0 的情况。

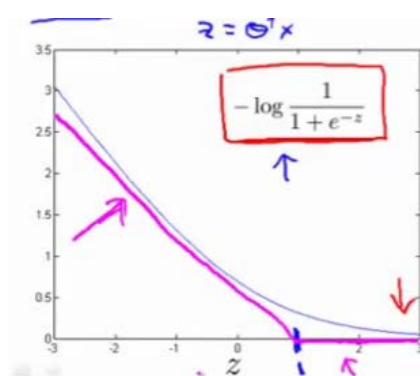
在第一种情况中，假设 $y = 1$ ，此时在目标函数中只需有第一项起作用，因为 $y = 1$ 时， $(1 - y)$ 项将等于 0。因此，当在 $y = 1$ 的样本中时，即在 (x, y) 中，我们得到 $y = 1$ $-\log(1 - \frac{1}{1+e^{-z}})$ 这样一项，这里同上一张幻灯片一致。

我用 z 表示 $\theta^T x$ ，即： $z = \theta^T x$ 。当然，在代价函数中， y 前面有负号。我们只是这

样表示，如果 $y = 1$ 代价函数中，这一项也等于 1。这样做是为了简化此处的表达式。如果画出关于 z 的函数，你会看到左下角的这条曲线，我们同样可以看到，当 z 增大时，也就是相当于 $\theta^T x$ 增大时， z 对应的值会变的非常小。对整个代价函数而言，影响也非常小。这也就解释了，为什么逻辑回归在观察到正样本 $y = 1$ 时，试图将 $\theta^T x$ 设置得非常大。因为，在代价函数中的这一项会变的非常小。

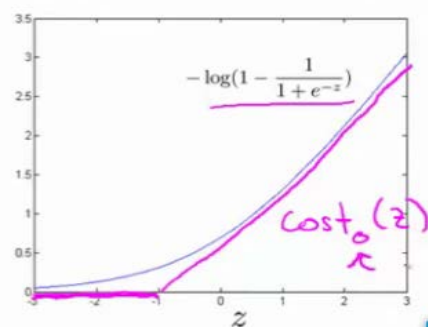
现在开始建立支持向量机，我们从这里开始：

我们会从这个代价函数开始，也就是 $-\log(1 - \frac{1}{1+e^{-z}})$ 一点一点修改，让我取这里的 $z = 1$ 点，我先画出将要用的代价函数。



新的代价函数将会水平的从这里到右边(图外)，然后我再画一条同逻辑回归非常相似的直线，但是，在这里是一条直线，也就是我用紫红色画的曲线，就是这条紫红色的曲线。那么，到了这里已经非常接近逻辑回归中使用的代价函数了。只是这里是由两条线段组成，即位于右边的水平部分和位于左边的直线部分，先别过多的考虑左边直线部分的斜率，这并不是很重要。但是，这里我们将使用的新的代价函数，是在 $y = 1$ 的前提下的。你也许能想到，这应该能做同逻辑回归中类似的事情，但事实上，在之后的优化问题中，这会变得更坚定，并且为支持向量机，带来计算上的优势。例如，更容易计算股票交易的问题等等。

目前，我们只是讨论了 $y = 1$ 的情况，另外一种情况是当 $y = 0$ 时，此时如果你仔细观察代价函数只留下了第二项，因为第一项被消除了。如果当 $y = 0$ 时，那么这一项也就是 0 了。所以上述表达式只留下了第二项。因此，这个样本的代价或是代价函数的贡献。将会由这一项表示。并且，如果你将这一项作为 z 的函数，那么，这里就会得到横轴 z 。现在，你完成了支持向量机中的部分内容，同样地，我们要替代这一条蓝色的线，用相似的方法。



如果我们用一个新的代价函数来代替，即这条从 0 点开始的水平直线，然后是一条斜线，像上图。那么，现在让我给这两个方程命名，左边的函数，我称之为 $cost_1(z)$ ，同时，右边函数我称它为 $cost_0(z)$ 。这里的下标是指代价函数中，对应的 $y = 1$ 和 $y = 0$ 的情况，拥有了这些定义后，现在，我们就开始构建支持向量机。

Support vector machine

Logistic regression:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \underbrace{\left(-\log h_{\theta}(x^{(i)}) \right)}_{cost_1(\theta^T x^{(i)})} + (1 - y^{(i)}) \underbrace{\left(-\log(1 - h_{\theta}(x^{(i)})) \right)}_{cost_0(\theta^T x^{(i)})} \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Support vector machine:

$$\min_{\theta} C \sum_{i=1}^m y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Handwritten notes and simplifications:

$$\min_u \frac{(u-5)^2 + 1}{10} \rightarrow u=5$$

$$\min_u 10(u-5)^2 + 10 \rightarrow u=5$$

$$A + \lambda B \leftarrow C$$

$$C A + B \leftarrow C$$

$$C = \frac{1}{\lambda}$$

$$\rightarrow \min_{\theta} C \sum_{i=1}^m \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

这是我们在逻辑回归中使用代价函数 $J(\theta)$ 。也许这个方程看起来不是非常熟悉。这是因为之前有个负号在方程外面，但是，这里我所做的是，将负号移到了表达式的里面，这样做使得方程看起来有些不同。对于支持向量机而言，实质上我们要将这替换为 $cost_1(z)$ ，也就是 $cost_1(\theta^T x)$ ，同样地，我也将这一项替换为 $cost_0(z)$ ，也就是代价 $cost_0(\theta^T x)$ 。这里的代价函数 $cost_1$ ，就是之前所提到的那条线。此外，代价函数 $cost_0$ ，也是上面所介绍过的那条线。因此，对于支持向量机，我们得到了这里的最小化问题，即：

$$\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \underbrace{\left(-\log h_{\theta}(x^{(i)}) \right)}_{cost_1(\theta^T x^{(i)})} + (1 - y^{(i)}) \underbrace{\left(-\log(1 - h_{\theta}(x^{(i)})) \right)}_{cost_0(\theta^T x^{(i)})} \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

然后，再加上正则化参数。现在，按照支持向量机的惯例，事实上，我们的书写会稍微有些不同，代价函数的参数表示也会稍微有些不同。

首先，我们要除去 $1/m$ 这一项，当然，这仅仅是由于人们使用支持向量机时，对比于逻辑回归而言，不同的习惯所致，但这里我所说的意思是：你知道，我将要做的是仅仅除去 $1/m$ 这一项，但是，这也会得出同样的 θ 最优值，好的，因为 $1/m$ 仅是个常量，因此，你知道在这个最小化问题中，无论前面是否有 $1/m$ 这一项，最终我所得到的最优值 θ 都是一样的。这里我的意思是，先给你举一个实例，假定有一最小化问题：即要求当 $(u - 5)^2 + 1$ 取得最小值时的 u 值，这时最小值为：当 $u = 5$ 时取得最小值。

现在，如果我们想要将这个目标函数乘上常数 10 ，这里我的最小化问题就变成了：求使得 $10 \times (u - 5)^2 + 10$ 最小的值 u ，然而，使得这里最小的 u 值仍为 5 。因此将一些常数乘以你的最小化项，这并不会改变最小化该方程时得到 u 值。因此，这里我所做的是删去常量 m 。也相同的，我将目标函数乘上一个常量 m ，并不会改变取得最小值时的 θ 值。

第二点概念上的变化，我们只是指在使用支持向量机时，一些如下的标准惯例，而不是逻辑回归。因此，对于逻辑回归，在目标函数中，我们有两项：第一个是训练样本的代价，第二个是我们的正则化项，我们不得不去用这一项来平衡。这就相当于我们想要最小化 A 加上正则化参数 λ ，然后乘以及其他项 B 对吧？这里的 A 表示这里的第一项，同时我用 B 表示第二项，但不包括 λ ，我们不是优化这里的 $A + \lambda \times B$ 。我们所做的是通过设置不同正则参数 λ 达到优化目的。这样，我们就能够权衡对应的项，是使得训练样本拟合的更好。即最小化 A 。还是保证正则参数足够小，也即是对于 B 项而言，但对于支持向量机，按照惯例，我们将使用一个不同的参数替换这里使用的 λ 来权衡这两项。你知道，就是第一项和第二项我们依照惯例使用一个不同的参数称为 C ，同时改为优化目标， $C \times A + B$ 因此，在逻辑回归中，如果给定 λ ，一个非常大的值，意味着给予 B 更大的权重。而这里，就对应于将 C 设定为非常小的值，那么，相应的将会给 B 比给 A 更大的权重。因此，这只是一种不同的方式来控制这种权衡或者一种不同的方法，即用参数来决定是更关心第一项的优化，还是更关心第二项的优化。当然你也可以把这里的参数 C 考虑成 $1/\lambda$ ，同 $1/\lambda$ 所扮演的角色相同，并且这两个方程或这两个表达式并不相同，因为 $C = 1/\lambda$ ，但是也并不全是这样，如果当 $C = 1/\lambda$ 时，这两个优化目标应当得到相同的值，相同的最优值 θ 。因此，就用它们来代替。那么，我现在删掉这里的 λ ，并且用常数 C 来代替。因此，这就得到了在支持向量机中我们的整个优化目标函数。然后最小化这个目标函数，得到 **SVM** 学习到的参数 C 。

SVM hypothesis

$$\Rightarrow \min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Hypothesis:

$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

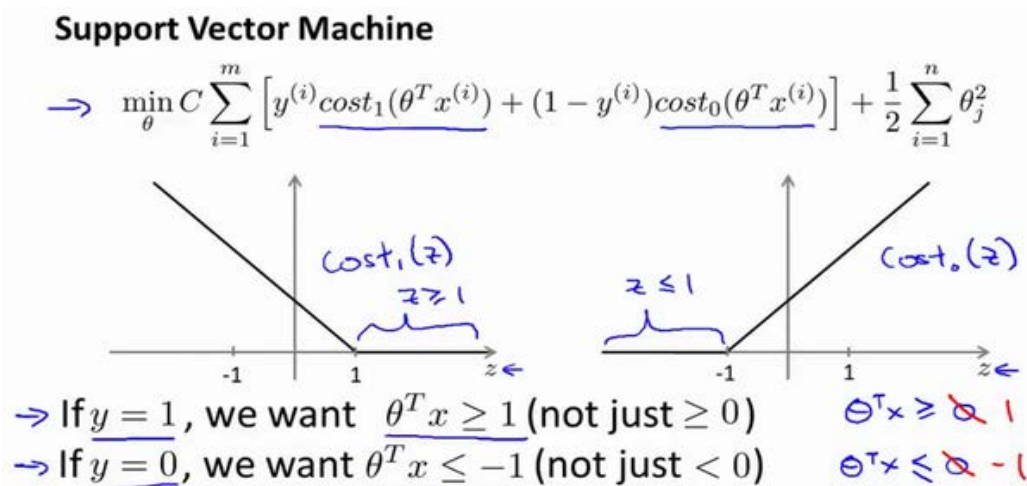
最后有别于逻辑回归输出的概率。在这里，我们的代价函数，当最小化代价函数，获得参数 θ 时，支持向量机所做的是它来直接预测 y 的值等于 1，还是等于 0。因此，这个假设函数会预测 1。当 $\theta^T x$ 大于或者等于 0 时，或者等于 0 时，所以学习参数 θ 就是支持向量机假设函数的形式。那么，这就是支持向量机数学上的定义。

在接下来的视频中，让我们再回去从直观的角度看看优化目标，实际上是在做什么，以及 SVM 的假设函数将会学习什么，同时也会谈谈如何做些许修改，学习更加复杂、非线性的函数。

12.2 大边界的直观理解

参考视频: 12 - 2 - Large Margin Intuition (11 min).mkv

人们有时将支持向量机看作是大间距分类器。在这一部分，我将介绍其中的含义，这有助于我们直观理解 SVM 模型的假设是什么样的。



这是我的支持向量机模型的代价函数，在左边这里我画出了关于 z 的代价函数 $\text{cost}_1(z)$ ，此函数用于正样本，而在右边这里我画出了关于 z 的代价函数 $\text{cost}_0(z)$ ，横轴表示 z ，现在让我们考虑一下，最小化这些代价函数的必要条件是什么。如果你有一个正样本， $y = 1$ ，则只有在 $z \geq 1$ 时，代价函数 $\text{cost}_1(z)$ 才等于 0。

换句话说，如果你有一个正样本，我们会希望 $\theta^T x \geq 1$ ，反之，如果 $y = 0$ ，我们观察一下，函数 $\text{cost}_0(z)$ ，它只有在 $z \leq -1$ 的区间里函数值为 0。这是支持向量机的一个有趣性质。事实上，如果你有一个正样本 $y = 1$ ，则其实我们仅仅要求 $\theta^T x$ 大于等于 0，就能将该样本恰当分出，这是因为如果 $\theta^T x > 0$ 大的话，我们的模型代价函数值为 0，类似地，如果你有一个负样本，则仅需要 $\theta^T x \leq 0$ 就会将负例正确分离，但是，支持向量机的要求更高，不仅仅要能正确分开输入的样本，即不仅仅要求 $\theta^T x > 0$ ，我们需要的是比 0 值大很多，比如大于等于 1，我也想这个比 0 小很多，比如我希望它小于等于 -1，这就相当于在支持向量机中嵌入了一个额外的安全因子，或者说安全的间距因子。

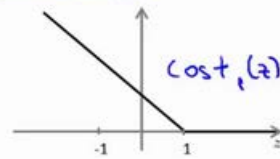
当然，逻辑回归做了类似的事情。但是让我们看一下，在支持向量机中，这个因子会导致什么结果。具体而言，我接下来会考虑一个特例。我们将这个常数 C 设置成一个非常大的值。比如我们假设 C 的值为 100000 或者其它非常大的数，然后来观察支持向量机会给出什么结果？

SVM Decision Boundary

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

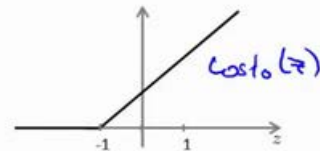
Whenever $y^{(i)} = 1$:

$$\theta^T x^{(i)} \geq 1$$



Whenever $y^{(i)} = 0$:

$$\theta^T x^{(i)} \leq -1$$



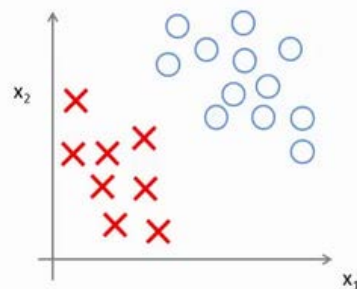
如果 C 非常大，则最小化代价函数的时候，我们将会很希望找到一个使第一项为 0 的最优解。因此，让我们尝试在代价项的第一项为 0 的情形下理解该优化问题。比如我们可以把 C 设置成了非常大的常数，这将给我们一些关于支持向量机模型的直观感受。

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

我们已经看到输入一个训练样本标签为 $y = 1$ ，你想令第一项为 0，你需要做的是找到一个 θ ，使得 $\theta^T x \geq 1$ ，类似地，对于一个训练样本，标签为 $y = 0$ ，为了使 $\text{cost}_0(z)$ 函数的值为 0，我们需要 $\theta^T x \leq -1$ 。因此，现在考虑我们的优化问题。选择参数，使得第一项等于 0，就会导致下面的优化问题，因为我们将选择参数使第一项为 0，因此这个函数的第一项为 0，因此是 C 乘以 0 加上二分之一乘以第二项。这里第一项是 C 乘以 0，因此可以将其删去，因为我知道它是 0。

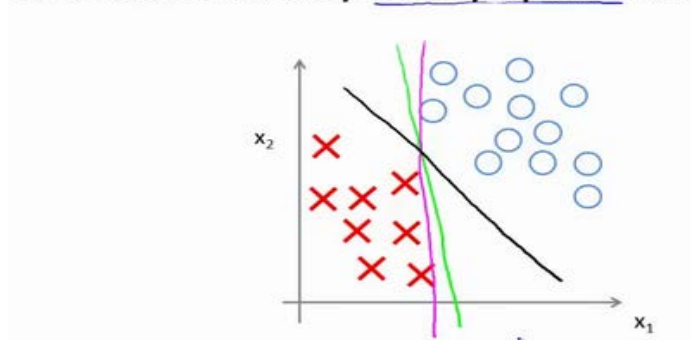
这将遵从以下的约束： $\theta^T x^{(i)} \geq 1$ ，如果 $y^{(i)}$ 是等于 1 的， $\theta^T x^{(i)} \leq -1$ ，如果样本 i 是一个负样本，这样当你求解这个优化问题的时候，当你最小化这个关于变量 θ 的函数的时候，你会得到一个非常有趣的决策边界。

SVM Decision Boundary: Linearly separable case



具体而言，如果你考察这样一个数据集，其中有正样本，也有负样本，可以看到这个数据集是线性可分的。我的意思是，存在一条直线把正负样本分开。当然有多条不同的直线，可以把正样本和负样本完全分开。

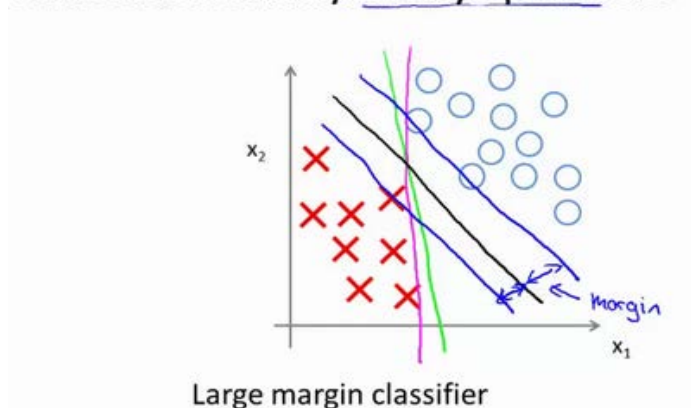
SVM Decision Boundary: Linearly separable case



比如，这就是一个决策边界可以把正样本和负样本分开。但是多多少少这个看起来并不是非常自然么？

或者我们可以画一条更差的决策界，这是另一条决策边界，可以将正样本和负样本分开，但仅仅是勉强分开，这些决策边界看起来都不是特别好的选择，支持向量机将会选择这个黑色的决策边界，相较于之前我用粉色或者绿色画的决策界。这条黑色的看起来好得多，黑线看起来是更稳健的决策界。在分离正样本和负样本上它显得更好。数学上来讲，这是什么意思呢？这条黑线有更大的距离，这个距离叫做间距(margin)。

SVM Decision Boundary: Linearly separable case



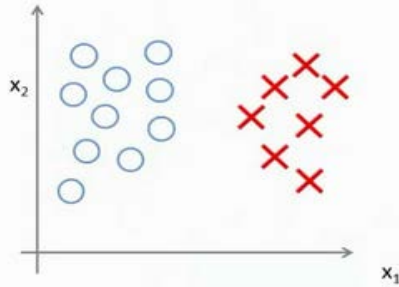
当画出这两条额外的蓝线，我们看到黑色的决策界和训练样本之间有更大的最短距离。然而粉线和蓝线离训练样本就非常近，在分离样本的时候就会比黑线表现差。因此，这个距离叫做支持向量机的间距，而这是支持向量机具有鲁棒性的原因，因为它努力用一个最大间距来分离样本。因此支持向量机有时被称为**大间距分类器**，而这其实是求解上一页幻灯片上优化问题的结果。

我知道你也许想知道求解上一页幻灯片中的优化问题为什么会产生这个结果？它是如

何产生这个大间距分类器的呢？我知道我还没有解释这一点。

我将会从直观上略述为什么这个优化问题会产生大间距分类器。总之这个图示有助于你理解支持向量机模型的做法，即努力将正样本和负样本用最大的间距分开。

Large margin classifier in presence of outliers

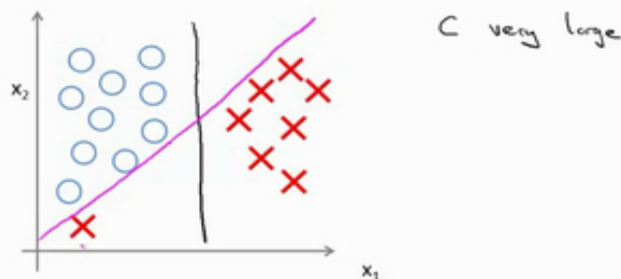


在本节课中关于大间距分类器，我想讲最后一点：我们将这个大间距分类器中的正则化因子常数 C 设置的非常大，我记得我将其设置为了 100000，因此对这样的一个数据集，也许我们将选择这样的决策界，从而最大间距地分离开正样本和负样本。那么在让代价函数最小化的过程中，我们希望找出在 $y = 1$ 和 $y = 0$ 两种情况下都使得代价函数中左边的这一项尽量为零的参数。如果我们找到了这样的参数，则我们的最小化问题便转变成：

$$\min \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad s.t. \begin{cases} \theta^T x^{(i)} \geq 1 \text{ if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 \text{ if } y^{(i)} = 0 \end{cases}$$

事实上，支持向量机现在要比这个大间距分类器所体现得更成熟，尤其是当你使用大间距分类器的时候，你的学习算法会受异常点(outlier) 的影响。比如我们加入一个额外的正样本。

Large margin classifier in presence of outliers



在这里，如果你加了这个样本，为了将样本用最大间距分开，也许我最终会得到一条类似这样的决策界，对么？就是这条粉色的线，仅仅基于一个异常值，仅仅基于一个样本，就将我的决策界从这条黑线变到这条粉线，这实在是不明智的。而如果正则化参数 C ，设置的非常大，这事实上正是支持向量机将会做的。它将决策界，从黑线变到了粉线，但是如果 C 设

置的小一点，如果你将 C 设置的不要太大，则你最终会得到这条黑线，当然数据如果不是线性可分的，如果你在这里有一些正样本或者你在这里有一些负样本，则支持向量机也会将它们恰当分开。因此，大间距分类器的描述，仅仅是从直观上给出了正则化参数 C 非常大的情形，同时，要提醒你 C 的作用类似于 $1/\lambda$ ， λ 是我们之前使用过的正则化参数。这只是 C 非常大的情形，或者等价地 λ 非常小的情形。你最终会得到类似粉线这样的决策界，但是实际上应用支持向量机的时候，当 C 不是非常非常大的时候，它可以忽略掉一些异常点的影响，得到更好的决策界。甚至当你的数据不是线性可分的时候，支持向量机也可以给出好的结果。

回顾 $C = 1/\lambda$ ，因此：

C 较大时，相当于 λ 较小，可能会导致过拟合，高方差。

C 较小时，相当于 λ 较大，可能会导致低拟合，高偏差。

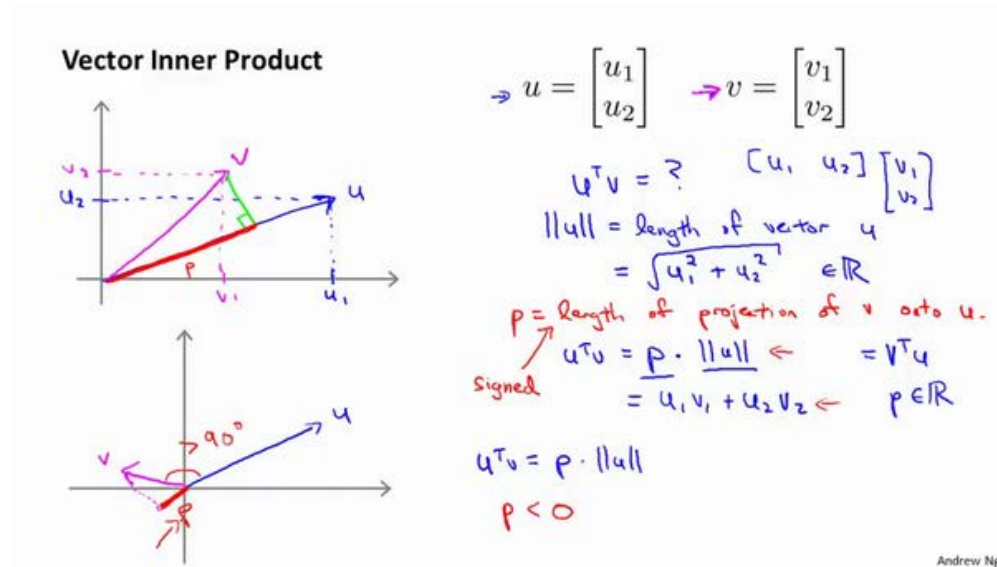
我们稍后会介绍支持向量机的偏差和方差，希望在那时候关于如何处理参数的这种平衡会变得更加清晰。我希望，这节课给出了一些关于为什么支持向量机被看做大间距分类器的直观理解。它用最大间距将样本区分开，尽管从技术上讲，这只有当参数 C 是非常大的时候是真的，但是它对于理解支持向量机是有益的。

本节课中我们略去了一步，那就是我们在幻灯片中给出的优化问题。为什么会是这样的？它是如何得出大间距分类器的？我在本节中没有讲解，在下一节课中，我将略述这些问题背后的数学原理，来解释这个优化问题是如何得到一个大间距分类器的。

12.3 大边界分类背后的数学（选修）

参考视频: 12 - 3 - Mathematics Behind Large Margin Classification (Optional) (20 min).mkv

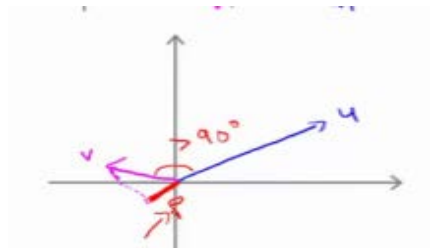
在本节课中，我将介绍一些大间隔分类背后的数学原理。本节为选学部分，你完全可以跳过它，但是听听这节课可能让你对支持向量机中的优化问题，以及如何得到大间隔分类器，产生更好的直观理解。



首先，让我来给大家复习一下关于向量内积的知识。假设我有两个向量， u 和 v ，我将它们写在这里。两个都是二维向量，我们看一下， $u^T v$ 的结果。 $u^T v$ 也叫做向量 u 和 v 之间的内积。由于是二维向量，我可以将它们画在这个图上。我们说，这就是向量 u 即在横轴上，取值为某个 u_1 ，而在纵轴上，高度是某个 u_2 作为 u 的第二个分量。现在，很容易计算的一个量就是向量 u 的范数。 $\|u\|$ 表示 u 的范数，即 u 的长度，即向量 u 的欧几里得长度。根据毕达哥拉斯定理， $\|u\| = \sqrt{u_1^2 + u_2^2}$ ，这是向量 u 的长度，它是一个实数。现在你知道了这个的长度是多少了。我刚刚画的这个向量的长度就知道了。

现在让我们回头来看向量 v ，因为我们想计算内积。 v 是另一个向量，它的两个分量 v_1 和 v_2 是已知的。向量 v 可以画在这里，现在让我们来看看如何计算 u 和 v 之间的内积。这就是具体做法，我们将向量 v 投影到向量 u 上，我们做一个直角投影，或者说一个90度投影将其投影到 u 上，接下来我度量这条红线的长度。我称这条红线的长度为 p ，因此 p 就是长度，或者说是向量 v 投影到向量 u 上的量，我将它写下来， p 是 v 投影到向量 u 上的长度，因此可以将 $u^T v = p \cdot \|u\|$ ，或者说 u 的长度。这是计算内积的一种方法。如果你从几何上画出 p 的值，

同时画出 u 的范数，你也会同样地计算出内积，答案是一样的。另一个计算公式是： $u^T v$ 就是 $[u_1 \ u_2]$ 这个一行两列的矩阵乘以 v 。因此可以得到 $u_1 \times v_1 + u_2 \times v_2$ 。根据线性代数的知识，这两个公式会给出同样的结果。顺便说一句， $u^T v = v^T u$ 。因此如果你将 u 和 v 交换位置，将 u 投影到 v 上，而不是将 v 投影到 u 上，然后做同样地计算，只是把 u 和 v 的位置交换一下，你事实上可以得到同样的结果。申明一点，在这个等式中 u 的范数是一个实数， p 也是一个实数，因此 $u^T v$ 就是两个实数正常相乘。

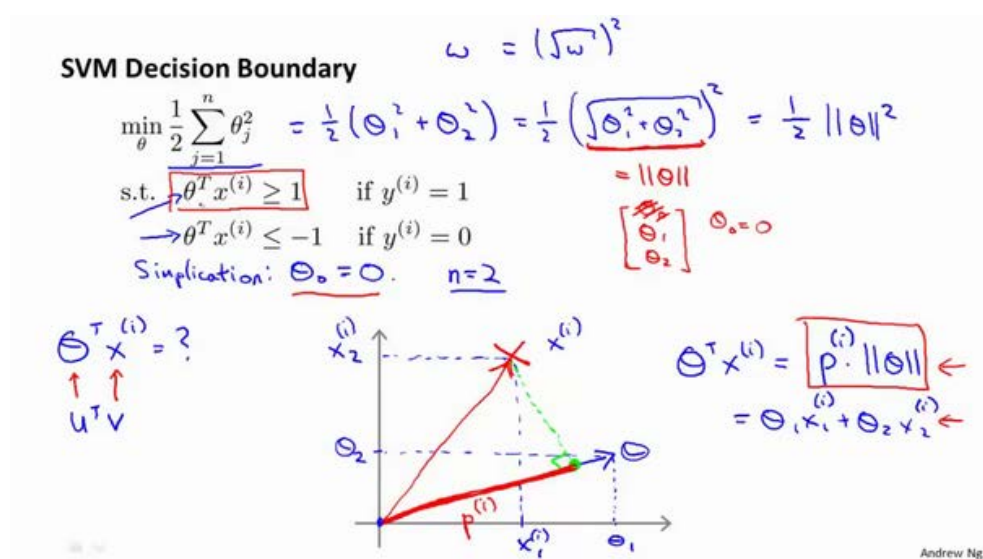


最后一点，需要注意的就是 p 值， p 事实上是有符号的，即它可能是正值，也可能是负值。我的意思是说，如果 u 是一个类似这样的向量， v 是一个类似这样的向量， u 和 v 之间的夹角大于 90 度，则如果将 v 投影到 u 上，会得到这样的投影，这是 p 的长度，在这个情形下我们仍然有 $u^T v$ 是等于 p 乘以 u 的范数。唯一一点不同的是 p 在这里是负的。在内积计算中，如果 u 和 v 之间的夹角小于 90 度，那么那条红线的长度 p 是正值。然而如果这个夹角大于 90 度，则 p 将会是负的。就是这个小线段的长度是负的。如果它们之间的夹角大于 90 度，两个向量之间的内积也是负的。这就是关于向量内积的知识。我们接下来将会使用这些关于向量内积的性质试图来理解支持向量机中的目标函数。

SVM Decision Boundary

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2} \sum_{j=1}^n \theta_j^2 \\ \text{s.t.} \quad & \theta^T x^{(i)} \geq 1 \quad \text{if } y^{(i)} = 1 \\ & \theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0 \end{aligned}$$

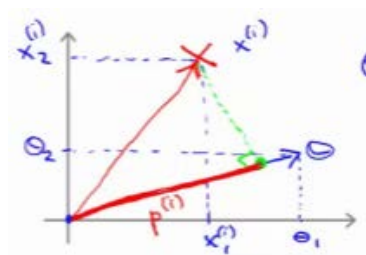
这就是我们先前给出的支持向量机模型中的目标函数。为了讲解方便，我做一点简化，仅仅是为了让目标函数更容易被分析。



我接下来忽略掉截距，令 $\theta_0 = 0$ ，这样更容易画示意图。我将特征数 n 置为 2，因此我们仅有两个特征 x_1, x_2 ，现在我们来看一下目标函数，支持向量机的优化目标函数。当我们仅有两个特征，即 $n = 2$ 时，这个式子可以写作： $\frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}(\sqrt{\theta_1^2 + \theta_2^2})^2$ ，我们只有两个参数 θ_1, θ_2 。你可能注意到括号里面的这一项是向量 θ 的范数，或者说是向量 θ 的长度。我的意思是如果我们将向量 θ 写出来，那么我刚刚画红线的这一项就是向量 θ 的长度或范数。这里我们用的是之前学过的向量范数的定义，事实上这就等于向量 θ 的长度。

当然你可以将其写作 $\theta_0, \theta_1, \theta_2$ ，如果 $\theta_0 = 0$ ，那就是 θ_1, θ_2 的长度。在这里我将忽略 θ_0 ，这样来写 θ 的范数，它仅仅和 θ_1, θ_2 有关。但是，数学上不管你是否包含，其实并没有差别，因此在我们接下来的推导中去掉 θ_0 不会有影响这意味着我们的目标函数是等于 $\frac{1}{2}\|\theta\|^2$ 。因此支持向量机做的全部事情，就是**极小化参数向量 θ 范数的平方，或者说长度的平方**。

现在我将要看看这些项： $\theta^T x$ 更深入地理解它们的含义。给定参数向量 θ 给定一个样本 x ，这等于什么呢？在前一页幻灯片上，我们画出了在不同情形下， $u^T v$ 的示意图，我们将会使用这些概念， θ 和 $x^{(i)}$ 就类似于 u 和 v 。

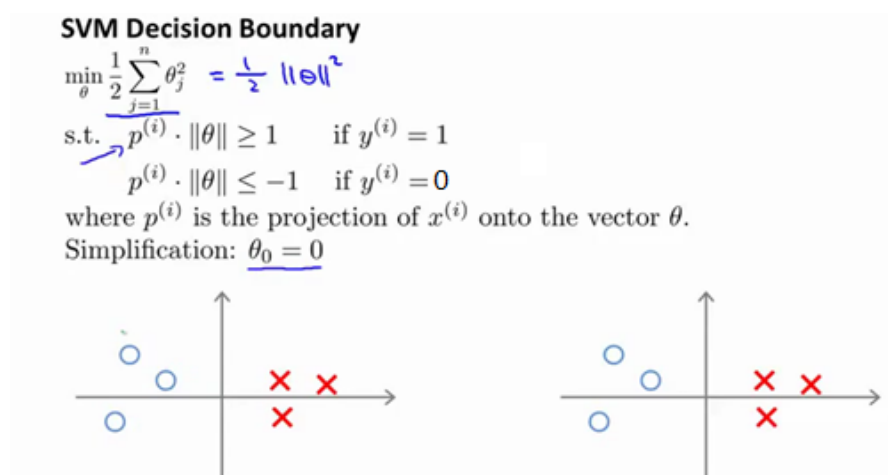


让我们看一下示意图：我们考察一个单一的训练样本，我有一个正样本在这里，用一个叉来表示这个样本 $x^{(i)}$ ，意思是在水平轴上取值为 $x_1^{(i)}$ ，在垂直轴上取值为 $x_2^{(i)}$ 。这就是我画

出的训练样本。尽管我没有将其真的看做向量。它事实上就是一个始于原点，终点位置在这个训练样本点的向量。现在，我们有一个参数向量我会将它也画成向量。我将 θ_1 画在横轴这里，将 θ_2 画在纵轴这里，那么内积 $\theta^T x^{(i)}$ 将会是什么呢？

使用我们之前的方法，我们计算的方式就是我将训练样本投影到参数向量 θ ，然后我来看一下这个线段的长度，我将它画成红色。我将它称为 $p^{(i)}$ 用来表示这是第 i 个训练样本在参数向量 θ 上的投影。根据我们之前幻灯片的内容，我们知道的是 $\theta^T x^{(i)}$ 将会等于 p 乘以向量 θ 的长度或范数。这就等于 $\theta_1 \cdot x_1^{(i)} + \theta_2 \cdot x_2^{(i)}$ 。这两种方式是等价的，都可以用来计算 θ 和 $x^{(i)}$ 之间的内积。

这告诉了我们什么呢？这里表达的意思是：这个 $\theta^T x^{(i)} \geq 1$ 或者 $\theta^T x^{(i)} < -1$ 的约束是可以被 $p^{(i)} \cdot \|\theta\| \geq 1$ 这个约束所代替的。因为 $\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\|$ ，将其写入我们的优化目标。我们将会得到没有了约束， $\theta^T x^{(i)}$ 而变成了 $p^{(i)} \cdot \|\theta\|$ 。



需要提醒一点，我们之前曾讲过这个优化目标函数可以被写成等于 $\frac{1}{2} \|\theta\|^2$ 。

现在让我们考虑下面这里的训练样本。现在，继续使用之前的简化，即 $\theta_0 = 0$ ，我们来看一下支持向量机会选择什么样的决策界。这是一种选择，我们假设支持向量机会选择这个决策边界。这不是一个非常好的选择，因为它的间距很小。这个决策界离训练样本的距离很近。我们来看一下为什么支持向量机不会选择它。

对于这样选择的参数 θ ，可以看到参数向量 θ 事实上是和决策界是90度正交的，因此这个绿色的决策界对应着一个参数向量 θ 这个方向，顺便提一句 $\theta_0 = 0$ 的简化仅仅意味着决策界必须通过原点(0,0)。现在让我们看一下这对于优化目标函数意味着什么。

SVM Decision Boundary

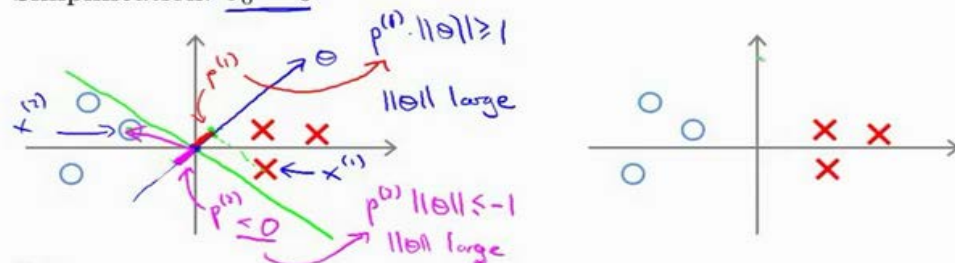
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

$$\text{s.t. } \boxed{p^{(i)} \cdot \|\theta\| \geq 1} \quad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \quad \text{if } y^{(i)} = -1$$

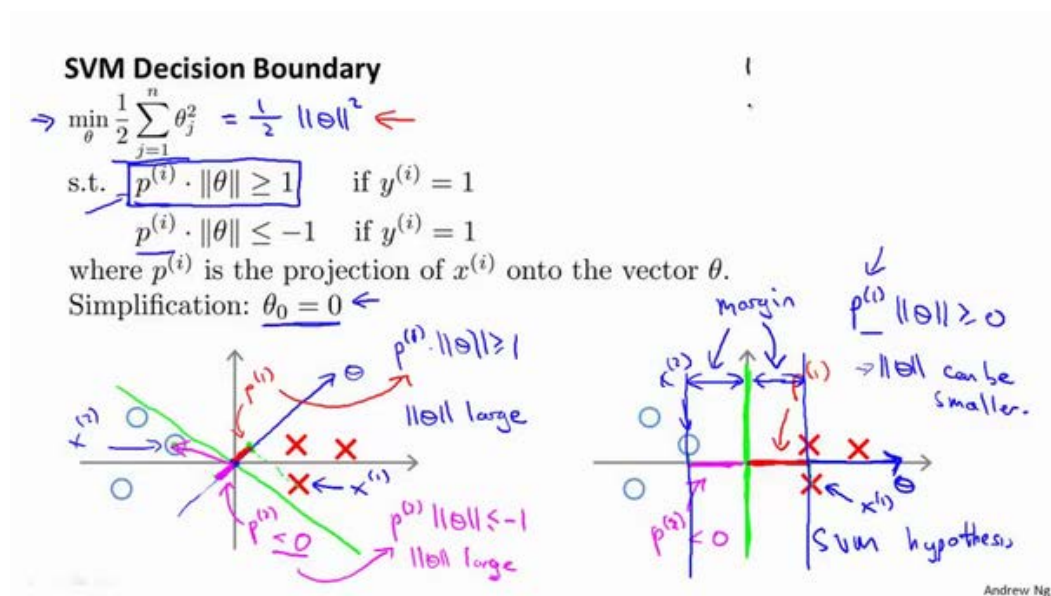
where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ .

Simplification: $\theta_0 = 0$



比如这个样本，我们假设它是我的第一个样本 $x^{(1)}$ ，如果我考察这个样本到参数 θ 的投影，投影是这个短的红线段，就等于 $p^{(1)}$ ，它非常短。类似地，这个样本如果它恰好是 $x^{(2)}$ ，我的第二个训练样本，则它到 θ 的投影在这里。我将它画成粉色，这个短的粉色线段是 $p^{(2)}$ ，即第二个样本到我的参数向量 θ 的投影。因此，这个投影非常短。 $p^{(2)}$ 事实上是一个负值， $p^{(2)}$ 是在相反的方向，这个向量和参数向量 θ 的夹角大于 90 度， $p^{(2)}$ 的值小于 0。

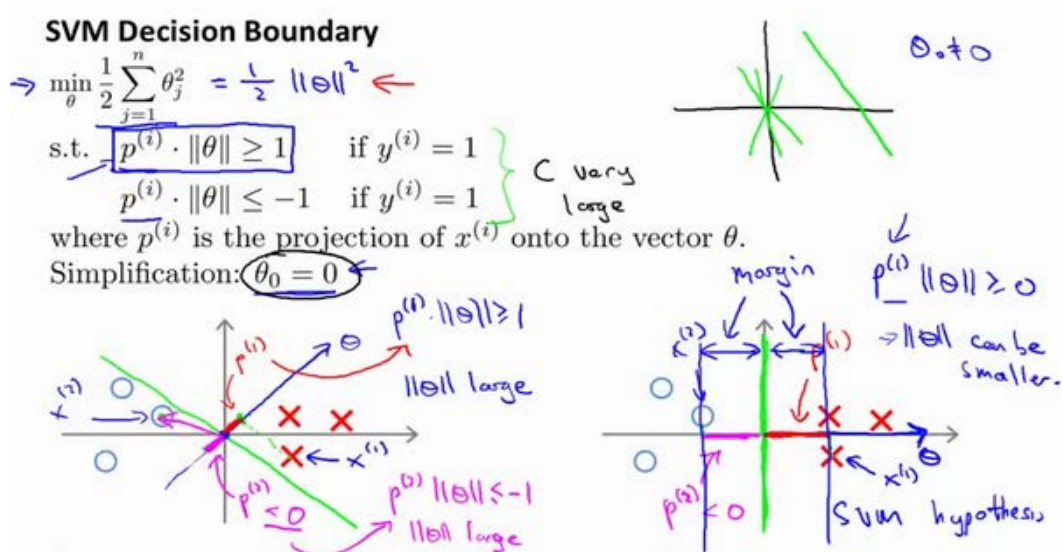
我们会发现这些 $p^{(i)}$ 将会是非常小的数，因此当我们考察优化目标函数的时候，对于正样本而言，我们需要 $p^{(i)} \cdot \|\theta\| \geq 1$ ，但是如果 $p^{(i)}$ 在这里非常小，那就意味着我们需要 θ 的范数非常大。因为如果 $p^{(1)}$ 很小，而我们希望 $p^{(1)} \cdot \|\theta\| \geq 1$ ，令其实现的唯一的办法就是这两个数较大。如果 $p^{(1)}$ 小，我们就希望 θ 的范数大。类似地，对于负样本而言我们需要 $p^{(2)} \cdot \|\theta\| \leq -1$ 。我们已经在这个样本中看到 $p^{(2)}$ 会是一个非常小的数，因此唯一的办法就是 θ 的范数变大。但是我们的目标函数是希望找到一个参数 θ ，它的范数是小的。因此，这看起来不像是一个好的参数向量 θ 的选择。



相反的，来看一个不同的决策边界。比如说，支持向量机选择了这个决策界，现在状况会有很大不同。如果这是决策界，这就是相对应的参数 θ 的方向，因此，在这个决策界之下，垂直线是决策界。使用线性代数的知识，可以说明，这个绿色的决策界有一个垂直于它的向量 θ 。现在如果你考察你的数据在横轴 x 上的投影，比如这个我之前提到的样本，我的样本 $x^{(1)}$ ，当我将它投影到横轴 x 上，或说投影到 θ 上，就会得到这样 $p^{(1)}$ 。它的长度是 $p^{(1)}$ ，另一个样本，那个样本是 $x^{(2)}$ 。我做同样的投影，我会发现， $p^{(2)}$ 的长度是负值。你会注意到现在 $p^{(1)}$ 和 $p^{(2)}$ 这些投影长度是长多了。如果我们仍然要满足这些约束， $P^{(i)} \cdot \|\theta\| > 1$ ，则因为 $p^{(1)}$ 变大了， θ 的范数就可以变小了。因此这意味着通过选择右边的决策界，而不是左边的那个，支持向量机可以使参数 θ 的范数变小很多。因此，如果我们想令 θ 的范数变小，从而令 θ 范数的平方变小，就能让支持向量机选择右边的决策界。这就是支持向量机如何能有效地产生大间距分类的原因。

看这条绿线，这个绿色的决策界。我们希望正样本和负样本投影到 θ 的值大。要做到这点的唯一方式就是选择这条绿线做决策界。这是大间距决策界来区分开正样本和负样本这个间距的值。这个间距的值就是 $p^{(1)}, p^{(2)}, p^{(3)}$ 等等的值。通过让间距变大，即通过这些 $p^{(1)}, p^{(2)}, p^{(3)}$ 等等的值，支持向量机最终可以找到一个较小的 θ 范数。这正是支持向量机中最小化目标函数的目的。

以上就是为什么支持向量机最终会找到大间距分类器的原因。因为它试图极大化这些 $p^{(i)}$ 的范数，它们是训练样本到决策边界的距离。最后一点，我们的推导自始至终使用了这个简化假设，就是参数 $\theta_0 = 0$ 。



就像我之前提到的。这个的作用是： $\theta_0 = 0$ 的意思是让我们让决策界通过原点。如果你令 θ_0 不是0的话，含义就是你希望决策界不通过原点。我将不会做全部的推导。实际上，支持向量机产生大间距分类器的结论，会被证明同样成立，证明方式是非常类似的，是我们刚刚做的证明的推广。

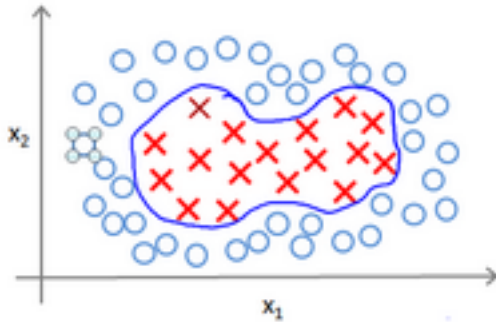
之前视频中说过，即便 θ_0 不等于0，支持向量机要做的事情都是优化这个目标函数对应着 C 值非常大的情况，但是可以说明的是，即便 θ_0 不等于0，支持向量机仍然会找到正样本和负样本之间的大间距分隔。

总之，我们解释了为什么支持向量机是一个大间距分类器。在下一节我们，将开始讨论如何利用支持向量机的原理，应用它们建立一个复杂的非线性分类器。

12.4 核函数 1

参考视频: 12 - 4 - Kernels I (16 min).mkv

回顾我们之前讨论过可以使用高级数的多项式模型来解决无法用直线进行分隔的分类问题:

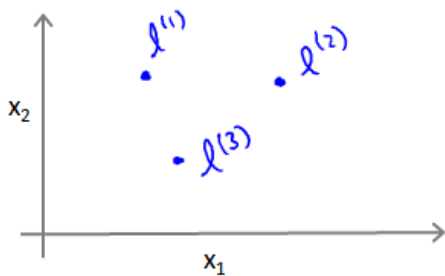


为了获得上图所示的判定边界, 我们的模型可能是 $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots$ 的形式。

我们可以用一系列的新的特征 f 来替换模型中的每一项。例如令: $f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_4 = x_1^2, f_5 = x_2^2$

...得到 $h_\theta(x) = f_1 + f_2 + \dots + f_n$ 。然而, 除了对原有的特征进行组合以外, 有没有更好的方法来构造 f_1, f_2, f_3 ? 我们可以利用核函数来计算出新的特征。

给定一个训练实例 x , 我们利用 x 的各个特征与我们预先选定的 **地标 (landmarks)** $l^{(1)}, l^{(2)}, l^{(3)}$ 的近似程度来选取新的特征 f_1, f_2, f_3 。



$$\text{例如: } f_1 = \text{similarity}(x, l^{(1)}) = e\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

其中: $\|x - l^{(1)}\|^2 = \sum_{j=1}^n (x_j - l_j^{(1)})^2$, 为实例 x 中所有特征与地标 $l^{(1)}$ 之间的距离的和。

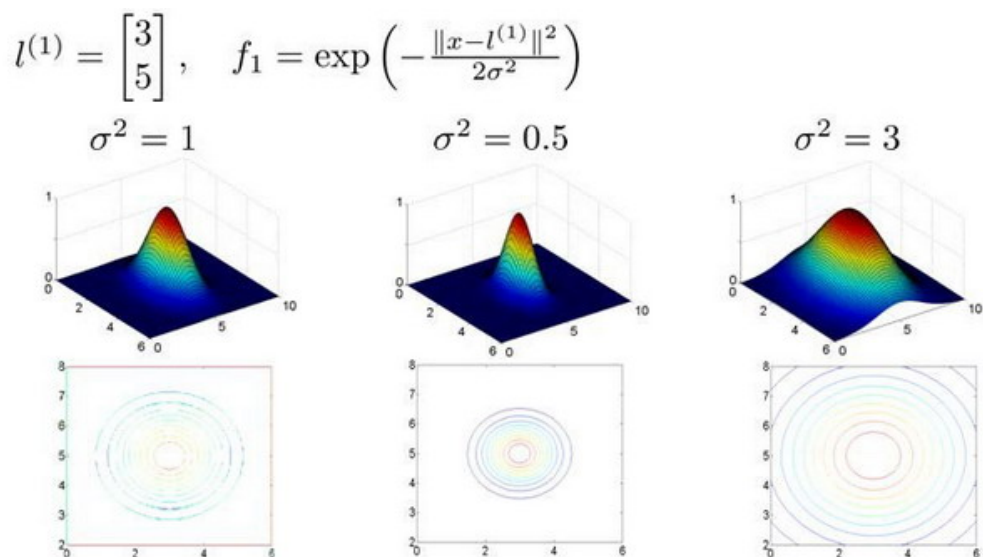
上例中的 $\text{similarity}(x, l^{(1)})$ 就是核函数, 具体而言, 这里是一个 **高斯核函数 (Gaussian Kernel)**。

注: 这个函数与正态分布没什么实际上的关系, 只是看上去像而已。

这些地标的的作用是什么? 如果一个训练实例 x 与地标 L 之间的距离近似于 0, 则新特征 f

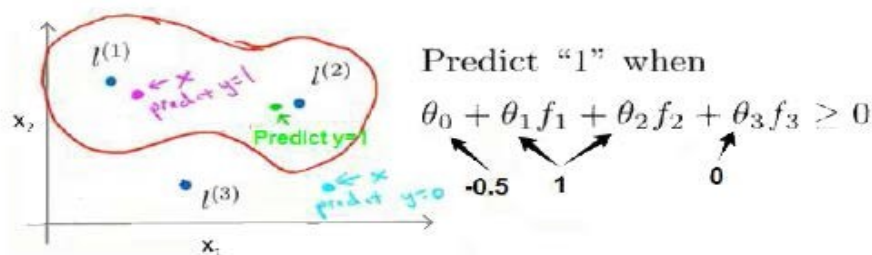
近似于 $e^{-0} = 1$ ，如果训练实例 x 与地标 l 之间距离较远，则 f 近似于 $e^{-(\text{一个较大的数})} = 0$ 。

假设我们的训练实例含有两个特征 $[x_1 \ x_2]$ ，给定地标 $l^{(1)}$ 与不同的 σ 值，见下图：



图中水平面的坐标为 x_1, x_2 而垂直坐标轴代表 f 。可以看出，只有当 x 与 $l^{(1)}$ 重合时 f 才具有最大值。随着 x 的改变 f 值改变的速率受到 σ^2 的控制。

在下图中，当实例处于洋红色的点位置处，因为其离 $l^{(1)}$ 更近，但是离 $l^{(2)}$ 和 $l^{(3)}$ 较远，因此 f_1 接近 1，而 f_2, f_3 接近 0。因此 $h_\theta(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 > 0$ ，因此预测 $y = 1$ 。同理可以求出，对于离 $l^{(2)}$ 较近的绿色点，也预测 $y = 1$ ，但是对于蓝绿色的点，因为其离三个地标都较远，预测 $y = 0$ 。



这样，图中红色的封闭曲线所表示的范围，便是我们依据一个单一的训练实例和我们选取的地标所得出的判定边界，在预测时，我们采用的特征不是训练实例本身的特征，而是通过核函数计算出的新特征 f_1, f_2, f_3 。

12.5 核函数 2

参考视频: 12 - 5 - Kernels II (16 min).mkv

在上一节视频里，我们讨论了核函数这个想法，以及怎样利用它去实现支持向量机的一些新特性。在这一节视频中，我将补充一些缺失的细节，并简单的介绍一下怎么在实际中使用应用这些想法。

如何选择地标？

我们通常是根据训练集的数量选择地标的数量，即如果训练集中有 m 个实例，则我们选取 m 个地标，并且令： $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$ 。这样做的好处在于：现在我们得到的新特征是建立在原有特征与训练集中所有其他特征之间距离的基础之上的，即：

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} = 1 \\ f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)}) \\ f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = e^0 = 1 \\ \vdots \\ f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)}) \end{bmatrix}$$

SVM with Kernels

- Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$,
- choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$.

Given example x :

$$\begin{aligned} \rightarrow f_1 &= \text{similarity}(x, l^{(1)}) \\ \rightarrow f_2 &= \text{similarity}(x, l^{(2)}) \\ &\vdots \end{aligned}$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

For training example $(x^{(i)}, y^{(i)})$:

$$\begin{aligned} f_1^{(i)} &= \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} &= \text{sim}(x^{(i)}, l^{(2)}) \\ &\vdots \\ f_i^{(i)} &= \text{sim}(x^{(i)}, l^{(i)}) = \exp(-\frac{0}{2x_i}) = 1 \\ &\vdots \\ f_m^{(i)} &= \text{sim}(x^{(i)}, l^{(m)}) \end{aligned}$$

$$x^{(i)} \in \mathbb{R}^{n+1} \quad (\text{or } \mathbb{R}^n) \rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \quad f_0^{(i)} = 1$$

下面我们将核函数运用到支持向量机中，修改我们的支持向量机假设为：

- 给定 x ，计算新特征 f ，当 $\theta^T f \geq 0$ 时，预测 $y = 1$ ，否则反之。

相应地修改代价函数为： $\sum_{j=1}^m \theta_j^2 = \theta^T \theta$,

$\min C \sum_{i=1}^m [y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)})] + \frac{1}{2} \sum_{j=1}^m \theta_j^2$ 在具体实施过程中，

我们还需要对最后的正则化项进行些微调整，在计算 $\sum_{j=1}^n \theta_j^2 = \theta^T \theta$ 时，我们用 $\theta^T M \theta$ 代替 $\theta^T \theta$ ，其中 M 是根据我们选择的核函数而不同的一个矩阵。这样做的原因是为了简化计算。

理论上讲，我们也可以在逻辑回归中使用核函数，但是上面使用 M 来简化计算的方法不适用与逻辑回归，因此计算将非常耗费时间。

在此，我们不介绍最小化支持向量机的代价函数的方法，你可以使用现有的软件包（如 **liblinear**, **libsvm** 等）。在使用这些软件包最小化我们的代价函数之前，我们通常需要编写核函数，并且如果我们使用高斯核函数，那么在使用之前进行特征缩放是非常必要的。

另外，支持向量机也可以不使用核函数，不使用核函数又称为**线性核函数(linear kernel)**，当我们不采用非常复杂的函数，或者我们的训练集特征非常多而实例非常少的时候，可以采用这种不带核函数的支持向量机。

下面是支持向量机的两个参数 C 和 σ 的影响：

$$C = 1/\lambda$$

C 较大时，相当于 λ 较小，可能会导致过拟合，高方差；

C 较小时，相当于 λ 较大，可能会导致低拟合，高偏差；

σ 较大时，可能会导致低方差，高偏差；

σ 较小时，可能会导致低偏差，高方差。

如果你看了本周的编程作业，你就能亲自实现这些想法，并亲眼看到这些效果。这就是利用核函数的支持向量机算法，希望这些关于偏差和方差的讨论，能给你一些对于算法结果预期的直观印象。

12.6 使用支持向量机

参考视频: 12 - 6 - Using An SVM (21 min).mkv

目前为止，我们已经讨论了 **SVM** 比较抽象的层面，在这个视频中我将要讨论到为了运行或者运用 **SVM**。你实际上所需要的一些东西：支持向量机算法，提出了一个特别优化的问题。但是就如在之前的视频中我简单提到的，我真的不建议你自己写软件来求解参数 θ ，因此由于今天我们中的很少人，或者其实没有人考虑过自己写代码来转换矩阵，或求一个数的平方根等我们只是知道如何去调用库函数来实现这些功能。同样的，用以解决 **SVM** 最优化问题的软件很复杂，且已经有研究者做了很多年数值优化了。因此你提出好的软件库和好的软件包来做这样一些事儿。然后强烈建议使用高优化软件库中的一个，而不是尝试自己落实一些数据。有许多好的软件库，我正好用得最多的两个是 **liblinear** 和 **libsvm**，但是真的有很多软件库可以用来做这件事儿。你可以连接许多你可能会用来编写学习算法的主要编程语言。

在高斯核函数之外我们还有其他一些选择，如：

多项式核函数 (**Polynomial Kernel**)

字符串核函数 (**String kernel**)

卡方核函数 (**chi-square kernel**)

直方图交集核函数 (**histogram intersection kernel**)

等等...

这些核函数的目标也都是根据训练集和地标之间的距离来构建新特征，这些核函数需要满足 **Mercer's** 定理，才能被支持向量机的优化软件正确处理。

多类分类问题

假设我们利用之前介绍的一对多方法来解决一个多类分类问题。如果一共有 k 个类，则我们需要 k 个模型，以及 k 个参数向量 θ 。我们同样也可以训练 k 个支持向量机来解决多类分类问题。但是大多数支持向量机软件包都有内置的多类分类功能，我们只要直接使用即可。

尽管你不去写你自己的 **SVM** 的优化软件，但是你也需要做几件事：

1、是提出参数 C 的选择。我们在之前的视频中讨论过误差/方差在这方面的性质。

2、你也需要选择内核参数或你想要使用的相似函数，其中一个选择是：我们选择不需要任何内核参数，没有内核参数的理念，也叫线性核函数。因此，如果有人说他使用了线性

核的 **SVM**（支持向量机），这就意味这他使用了不带有核函数的 **SVM**（支持向量机）。

从逻辑回归模型，我们得到了支持向量机模型，在两者之间，我们应该如何选择呢？

下面是一些普遍使用的准则：

n 为特征数， m 为训练样本数。

(1)如果相较于 m 而言， n 要大许多，即训练集数据量不够支持我们训练一个复杂的非线性模型，我们选用逻辑回归模型或者不带核函数的支持向量机。

(2)如果 n 较小，而且 m 大小中等，例如 n 在 1-1000 之间，而 m 在 10-10000 之间，使用高斯核函数的支持向量机。

(3)如果 n 较小，而 m 较大，例如 n 在 1-1000 之间，而 m 大于 50000，则使用支持向量机会非常慢，解决方案是创造、增加更多的特征，然后使用逻辑回归或不带核函数的支持向量机。

值得一提的是，神经网络在以上三种情况下都可能会有较好的表现，但是训练神经网络可能非常慢，选择支持向量机的原因主要在于它的代价函数是凸函数，不存在局部最小值。

今天的 **SVM** 包会工作得很好，但是它们仍然会有一些慢。当你有非常非常大的训练集，且用高斯核函数是在这种情况下，我经常会做的是尝试手动地创建，拥有更多的特征变量，然后用逻辑回归或者不带核函数的支持向量机。如果你看到这个幻灯片，看到了逻辑回归，或者不带核函数的支持向量机。在这个两个地方，我把它们放在一起是有原因的。原因是：逻辑回归和不带核函数的支持向量机它们都是非常相似的算法，不管是逻辑回归还是不带核函数的 **SVM**，通常都会做相似的事情，并给出相似的结果。但是根据你实现的情况，其中一个可能会比另一个更加有效。但是在其中一个算法应用的地方，逻辑回归或不带核函数的 **SVM** 另一个也很有可能很有效。但是随着 **SVM** 的复杂度增加，当你使用不同的内核函数来学习复杂的非线性函数时，这个体系，你知道的，当你有多达 1 万（10,000）的样本时，也可能是 5 万（50,000），你的特征变量的数量这是相当大的。那是一个非常常见的体系，也许在这个体系里，不带核函数的支持向量机就会表现得相当突出。你可以做比这困难得多需要逻辑回归的事情。

最后，神经网络使用于什么时候呢？对于所有的这些问题，对于所有的这些不同体系一个设计得很好的神经网络也很可能会非常有效。有一个缺点是，或者说是有时可能不会使用神经网络的原因是：对于许多这样的问题，神经网络训练起来可能会特别慢，但是如果你有一个非常好的 **SVM** 实现包，它可能会运行得比较快比神经网络快很多，尽管我们在此之前没有展示，但是事实证明，**SVM** 具有的优化问题，是一种凸优化问题。因此，好的 **SVM**

优化软件包总是会找到全局最小值，或者接近它的值。对于 **SVM** 你不需要担心局部最优。在实际应用中，局部最优不是神经网络所需要解决的一个重大问题，所以这是你在使用 **SVM** 的时候不需要太去担心的一个问题。根据你的问题，神经网络可能会比 **SVM** 慢，尤其是在这样一个体系中，至于这里给出的参考，看上去有些模糊，如果你在考虑一些问题，这些参考会有一些模糊，但是我仍然不能完全确定，我是该用这个算法还是改用那个算法，这个没有太大关系，当我遇到机器学习问题的时候，有时它确实不清楚这是否是最好的算法，但是就如在之前的视频中看到的算法确实很重要。但是通常更加重要的是：你有多少数据，你有多熟练是否擅长做误差分析和排除学习算法，指出如何设定新的特征变量和找出其他能决定你学习算法的变量等方面，通常这些方面会比你使用逻辑回归还是 **SVM** 这方面更加重要。但是，已经说过了，**SVM** 仍然被广泛认为是一种最强大的学习算法，这是一个体系，包含了什么时候一个有效的方法去学习复杂的非线性函数。因此，实际上与逻辑回归、神经网络、**SVM** 一起使用这些方法来提高学习算法，我认为你会很好地建立很有技术的状态。（编者注：当时 **GPU** 计算比较慢，神经网络还不流行。）

机器学习系统对于一个宽泛的应用领域来说，这是另一个在你军械库里非常强大的工具，你可以把它应用到很多地方，如硅谷、在工业、学术等领域建立许多高性能的机器学习系统。