实验三、支持向量机(Support Vector Machine)

实验目的：

1. 掌握支持向量机的原理

2. 掌握核函数的原理与作用

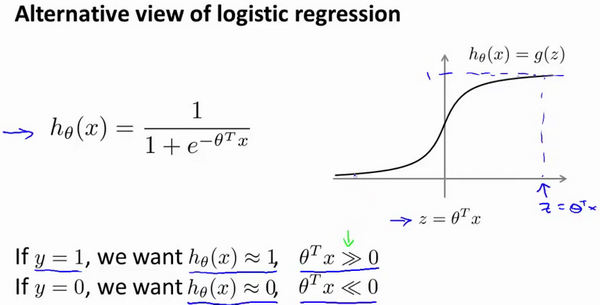
实验环境

WINDOWS + Anacoda + python 3.7+ jupyter notebook + scikit-learn

实验原理

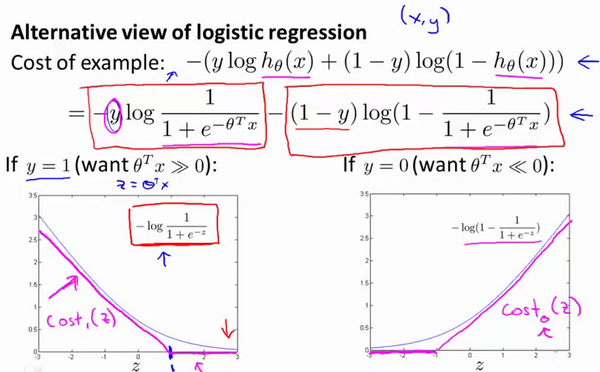
### 1 优化目标

在监督学习中，许多学习算法的性能都非常类似，因此，重要的不是选择使用学习算法**A**还是学习算法**B**，而更重要的是依赖于你的水平。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。在本次实验中，我们将介绍一个更加强大的算法，被广泛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。接下来我们将从逻辑回归开始展示我们如何一点一点修改来得到支持向量机。



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，在这里用 表示。

现在考虑下我们想要逻辑回归做什么：如果有一个 的样本，我们希望 趋近1。因为我们想要正确地将此样本分类，这就意味着当 趋近于1时， 应当远大于0。这是因为由于 表示 ，当 远大于0时，即到了该图的右边，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。我们希望假设函数的输出值将趋近于0，这对应于，或者就是 会远小于0，因为对应的假设函数的输出值趋近0。



如果进一步观察逻辑回归的代价函数，你会发现每个样本 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 这一项，但是这一项是影响整个总代价函数中的这一项的。

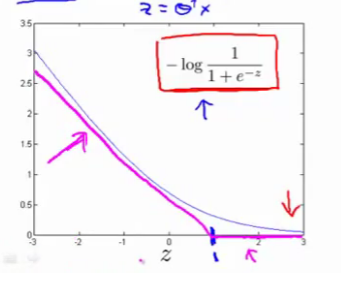
现在，一起来考虑两种情况：

一种是等于1的情况；另一种是 等于0的情况。

在第一种情况中，假设 ，此时在目标函数中只需有第一项起作用，因为时，项将等于0。因此，当在 的样本中时，即在 中 ，我们得到这样一项。用 表示，即： 。如果画出关于 的函数，你会看到左下角的这条曲线，我们同样可以看到，当 增大时，也就是相当于增大时， 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时，试图将设置得非常大。因为，在代价函数中的这一项会变的非常小。

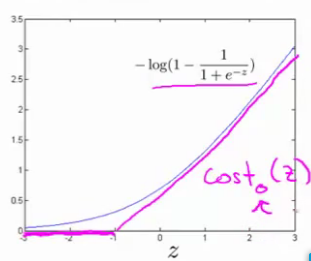
现在开始建立支持向量机：

我们会从这个代价函数开始，也就是一点一点修改，让我取这里的 点，其代价函数如下。

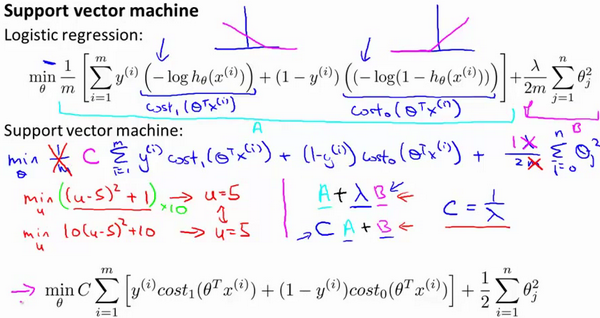


新的代价函数将会水平的从这里到右边(图外)，然后我再画一条同逻辑回归非常相似的直线，但是，在这里是一条直线，也就是紫红色画的曲线，就是这条紫红色的曲线。那么，到了这里已经非常接近逻辑回归中使用的代价函数了。只是这里是由两条线段组成，即位于右边的水平部分和位于左边的直线部分，先别过多的考虑左边直线部分的斜率，这并不是很重要。但是，这里我们将使用的新的代价函数，是在的前提下的。你也许能想到，这应该能做同逻辑回归中类似的事情，但事实上，在之后的优化问题中，这会变得更坚定，并且为支持向量机，带来计算上的优势。例如，更容易计算股票交易的问题等等。

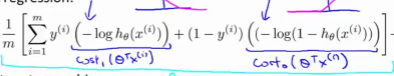
目前，我们只是讨论了的情况，另外一种情况是当时，此时如果你仔细观察代价函数只留下了第二项，因为第一项被消除了。如果当时，那么这一项也就是0了。所以上述表达式只留下了第二项。因此，这个样本的代价或是代价函数的贡献。将会由这一项表示。并且，如果你将这一项作为的函数，那么，这里就会得到横轴。



如果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，现在给这两个方程命名，左边的函数，称之为，同时，右边函数称它为。这里的下标是指在代价函数中，对应的 和 的情况，拥有了这些定义后，就可以开始构建支持向量机。



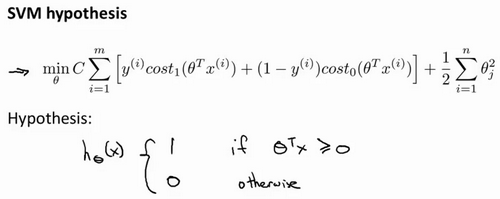
这是我们在逻辑回归中使用代价函数。也许这个方程看起来不是非常熟悉。这是因为之前有个负号在方程外面，但是，这里将负号移到了表达式的里面，这样做使得方程看起来有些不同。对于支持向量机而言，实质上我们要将这替换为，也就是，同样地，我也将这一项替换为，也就是代价。这里的代价函数，就是之前所提到的那条线。此外，代价函数，也是上面所介绍过的那条线。因此，对于支持向量机，我们得到了这里的最小化问题，即:



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，这仍会得出同样的 最优值，因为 仅是个常量。例如，假定有一最小化问题：即要求当取得最小值时的值，这时当时取得最小值。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：求使得最小的值，然而，使得这里最小的值仍为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值。因此，这里我所做的是删去常量。也相同的，我将目标函数乘上一个常量，并不会改变取得最小值时的值。

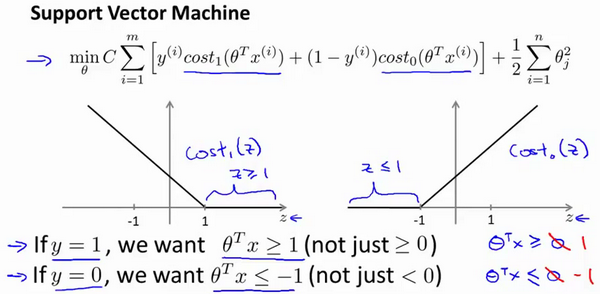
第二点概念上的变化，我们只是指在使用支持向量机时，一些如下的标准惯例，而不是逻辑回归。因此，对于逻辑回归，在目标函数中，我们有两项：第一个是训练样本的代价，第二个是我们的正则化项，我们不得不去用这一项来平衡。这就相当于我们想要最小化加上正则化参数，然后乘以其他项对吧？这里的表示这里的第一项，同时我用**B**表示第二项，但不包括，我们不是优化这里的。我们所做的是通过设置不同正则参数达到优化目的。这样，我们就能够权衡对应的项，是使得训练样本拟合的更好。即最小化。还是保证正则参数足够小，也即是对于**B**项而言，但对于支持向量机，按照惯例，我们将使用一个不同的参数替换这里使用的来权衡这两项。你知道，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，同时改为优化目标，因此，在逻辑回归中，如果给定，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 设定为非常小的值，那么，相应的将会给比给更大的权重。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 考虑成，同 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，但是也并不全是这样，如果当时，这两个优化目标应当得到相同的值，相同的最优值 。因此，就用它们来代替。那么，我现在删掉这里的，并且用常数来代替。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。



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，获得参数时，支持向量机所做的是它来直接预测的值等于1，还是等于0。因此，这个假设函数会预测1。当大于或者等于0时，或者等于0时，所以学习参数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。

### 2 大边界的直观理解

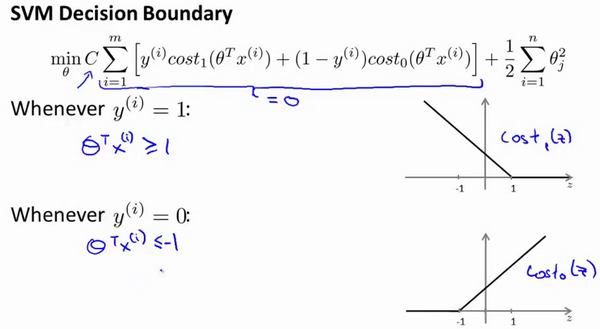
人们有时将支持向量机看作是大间距分类器。



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，此函数用于正样本，而在右边这里我画出了关于的代价函数，横轴表示，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，，则只有在时，代价函数才等于0。

换句话说，如果你有一个正样本，我们会希望>=1，反之，如果，我们观察一下，函数，它只有在的区间里函数值为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，则其实我们仅仅要求大于等于0，就能将该样本恰当分出，这是因为如果>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<=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>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。我们将这个常数设置成一个非常大的值。比如我们假设的值为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？

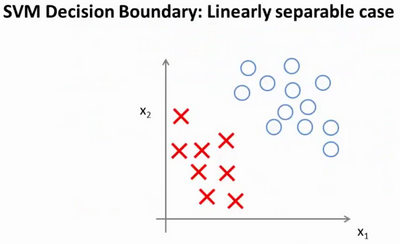


如果 非常大，则最小化代价函数的时候，我们将会很希望找到一个使第一项为0的最优解。因此，让我们尝试在代价项的第一项为0的情形下理解该优化问题。比如我们可以把设置成了非常大的常数，这将给我们一些关于支持向量机模型的直观感受。

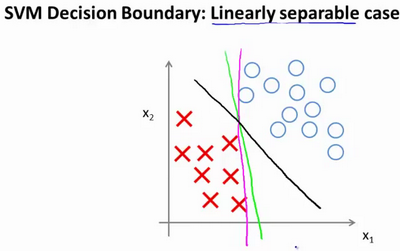


我们已经看到输入一个训练样本标签为，你想令第一项为0，你需要做的是找到一个，使得，类似地，对于一个训练样本，标签为，为了使 函数的值为0，我们需要。因此，现在考虑我们的优化问题。选择参数，使得第一项等于0，就会导致下面的优化问题，因为我们将选择参数使第一项为0，因此这个函数的第一项为0，因此是乘以0加上二分之一乘以第二项。这里第一项是乘以0，因此可以将其删去，因为我知道它是0。

这将遵从以下的约束：，如果 是等于1 的，，如果样本是一个负样本，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的函数的时候，你会得到一个非常有趣的决策边界。

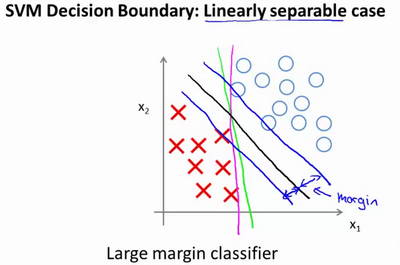


具体而言，如果你考察这样一个数据集，其中有正样本，也有负样本，可以看到这个数据集是线性可分的。我的意思是，存在一条直线把正负样本分开。当然有多条不同的直线，可以把正样本和负样本完全分开。



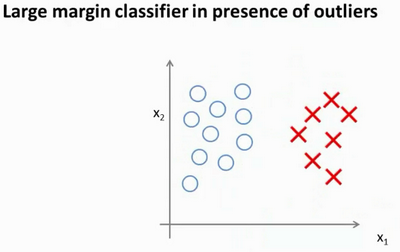
比如，这就是一个决策边界可以把正样本和负样本分开。但是多多少少这个看起来并不是非常自然是么?

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的更好。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**)。



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**。

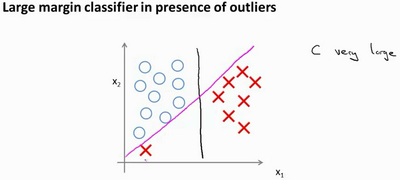
我将会从直观上略述为什么这个优化问题会产生大间距分类器。总之这个图示有助于你理解支持向量机模型的做法，即努力将正样本和负样本用最大的间距分开。



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，我们希望找出在和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：



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。



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，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 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非常大的情形，同时，要提醒你的作用类似于，是我们之前使用过的正则化参数。这只是非常大的情形，或者等价地 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。

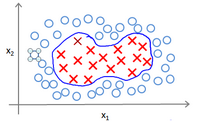
回顾 ，因此：

较大时，相当于 较小，可能会导致过拟合，高方差。

较小时，相当于 较大，可能会导致低拟合，高偏差。

### 3 核函数

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：

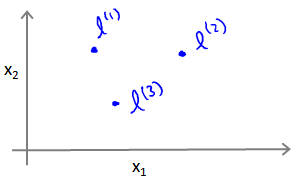


为了获得上图所示的判定边界，我们的模型可能是的形式。

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：

...得到。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？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，我们利用的各个特征与我们预先选定的**地标**(**landmarks**)的近似程度来选取新的特征。

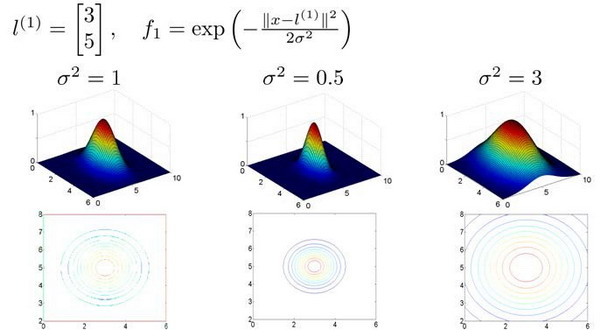


例如：

其中：，为实例中所有特征与地标之间的距离的和。上例中的就是核函数，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。**

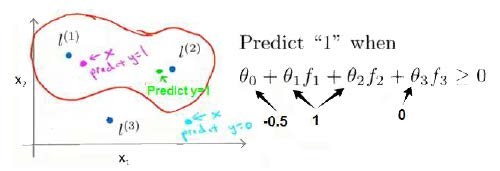
这些地标的作用是什么？如果一个训练实例与地标之间的距离近似于0，则新特征 近似于，如果训练实例与地标之间距离较远，则近似于。

假设我们的训练实例含有两个特征[ ]，给定地标与不同的值，见下图：



图中水平面的坐标为 ，而垂直坐标轴代表。可以看出，只有当与重合时才具有最大值。随着的改变值改变的速率受到的控制。

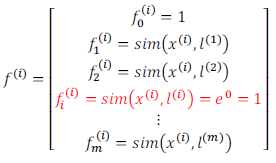
在下图中，当实例处于洋红色的点位置处，因为其离更近，但是离和较远，因此接近1，而,接近0。因此，因此预测。同理可以求出，对于离较近的绿色点，也预测，但是对于蓝绿色的点，因为其离三个地标都较远，预测。

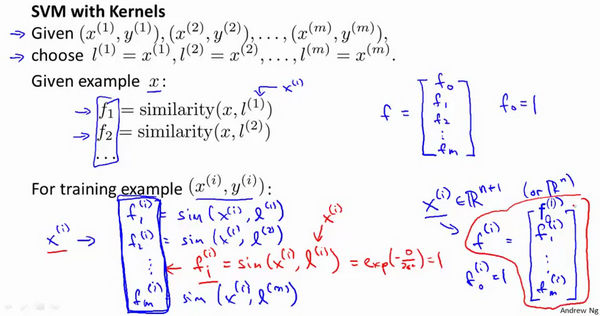


这样，图中红色的封闭曲线所表示的范围，便是我们依据一个单一的训练实例和我们选取的地标所得出的判定边界，在预测时，我们采用的特征不是训练实例本身的特征，而是通过核函数计算出的新特征。

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，即如果训练集中有个实例，则我们选取个地标，并且令:。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：





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，计算新特征，当 时，预测 ，否则反之。

相应地修改代价函数为：，

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，在计算时，我们用代替，其中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 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和的影响：

较大时，相当于较小，可能会导致过拟合，高方差；

较小时，相当于较大，可能会导致低拟合，高偏差；

较大时，可能会导致低方差，高偏差；

较小时，可能会导致低偏差，高方差。