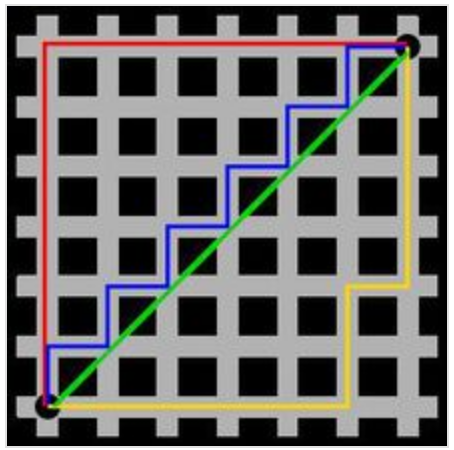
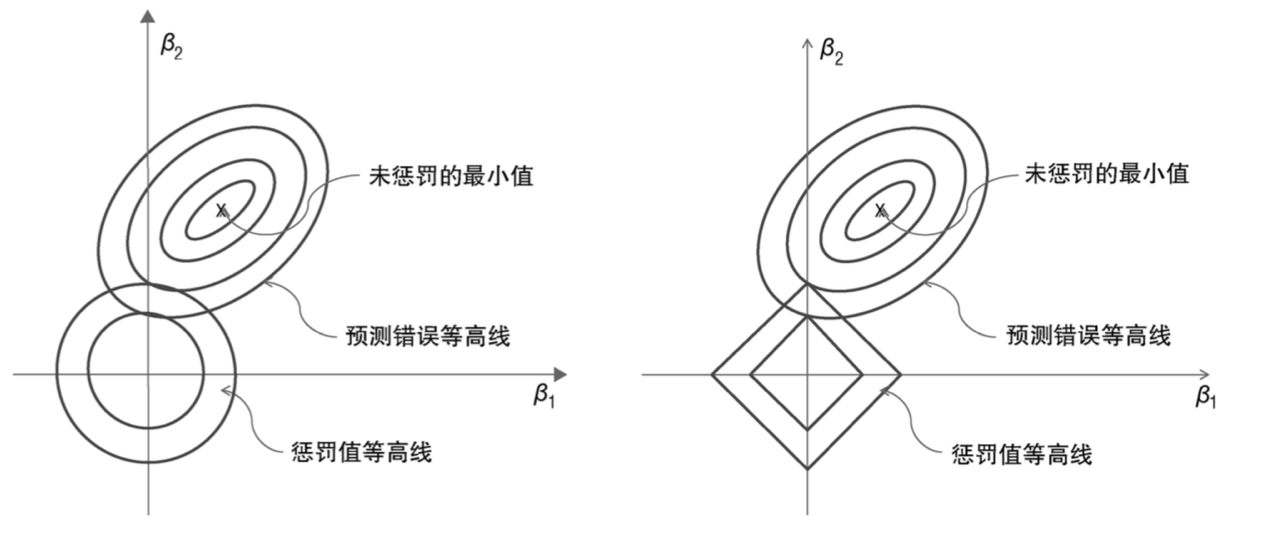
**Manhattan Distance（曼哈顿距离）**



图中红线代表曼哈顿距离，绿色代表欧氏距离，也就是直线距离，而蓝色和黄色代表等价的曼哈顿距离。曼哈顿距离——两点在南北方向上的距离加上在东西方向上的距离，即d（i，j）=|xi-xj|+|yi-yj|。对于一个具有正南正北、正东正西方向规则布局的城镇街道，从一点到达另一点的距离正是在南北方向上旅行的距离加上在东西方向上旅行的距离，因此，曼哈顿距离又称为出租车距离。

應用：

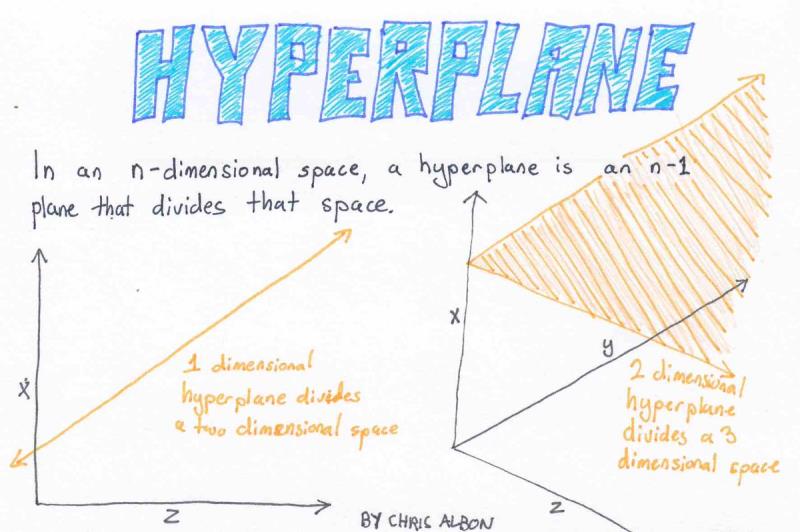
向 OLS 公式中分別添加一个系数惩罚项， 一個為歐式距離的懲罰項，另一個為曼哈頓距離。



这就是系数平方和的惩罚项与绝对值和的惩罚项之间的区别。在左图中，随着 λ 变化以及最小点的移动，平方惩罚项产生的切点一般不会落在坐标轴上。Β1 与 β2 都不为 0。相比之下，在右图中，绝对值和惩罚项产生的切点落在了 β2 的轴上。在 β2 轴上，β1=0。一个稀疏的系数向量相当于算法告诉你可以忽略一些因变量。

**Hyperplane(超平面)**

一個n-1維的超平面可以分開一個n維的空間。



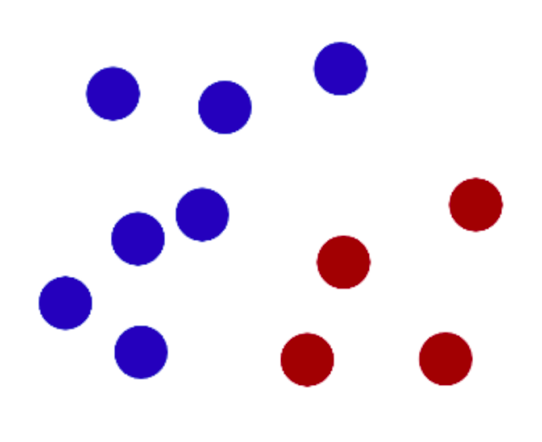
舉例：

一維超平面（直線）可以分開一個二維空間

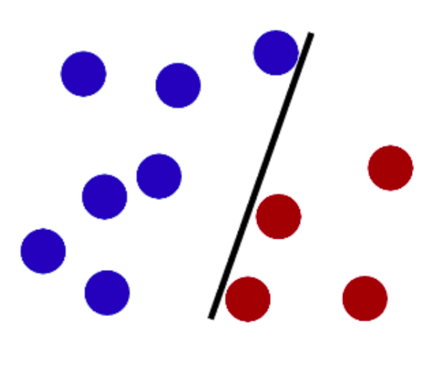
二維超平面可以分開一個三維空間

應用：

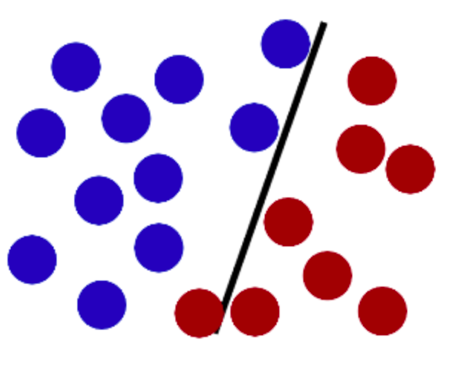
故事是这样子的：在很久以前的情人节，大侠要去救他的爱人，但魔鬼和他玩了一个游戏。魔鬼在桌子上似乎有规律放了两种颜色的球，说：“你用一根棍分开它们？要求：尽量在放更多球之后，仍然适用。”



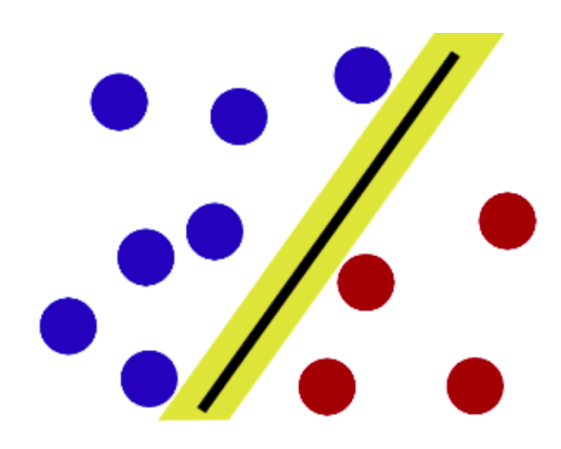
于是大侠这样放，干的不错？



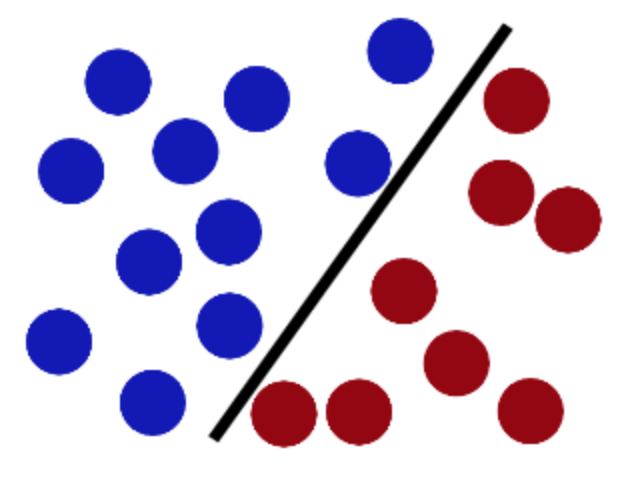
然后魔鬼，又在桌上放了更多的球，似乎有一个球站错了阵营。



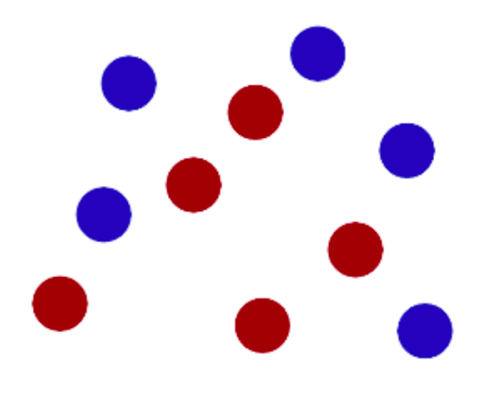
SVM就是试图把棍放在最佳位置，好让在棍的两边有尽可能大的间隙。



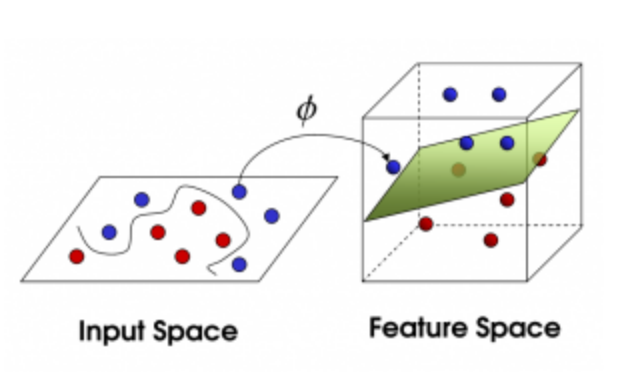
现在即使魔鬼放了更多的球，棍仍然是一个好的分界线。



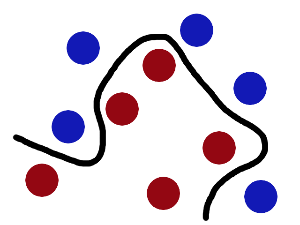
然后，在SVM 工具箱中有另一个更加重要的 trick。 魔鬼看到大侠已经学会了一个trick，于是魔鬼给了大侠一个新的挑战。



现在，大侠没有棍可以很好帮他分开两种球了，现在怎么办呢？当然像所有武侠片中一样大侠桌子一拍，球飞到空中。然后，凭借大侠的轻功，大侠抓起一张纸，插到了两种球的中间。



现在，从魔鬼的角度看这些球，这些球看起来像是被一条曲线分开了。



再之后，无聊的大人们，把这些球叫做 「data」，把棍子叫做 「classifier」, 最大间隙trick叫做「optimization」， 拍桌子叫做「kernelling」, 那张纸叫做「hyperplane」。

參考資料：<https://blog.csdn.net/sinat_35512245/article/details/54981721>

**One Hot encoding**

* 问题由来

在很多機器學習任务中，特征并不总是连续值，而有可能是分类值。

例如，考虑一下的三个特征：

["male", "female"]

["from Europe", "from US", "from Asia"]

["uses Firefox", "uses Chrome", "uses Safari", "uses Internet Explorer"]

如果将上述特征用数字表示，效率会高很多。例如：

["male", "from US", "uses Internet Explorer"] 表示为[0, 1, 3]

["female", "from Asia", "uses Chrome"]表示为[1, 2, 1]

但是，即使转化为数字表示后，上述数据也不能直接用在我们的分类器中。因为，分类器往往默认数据数据是连续的（可以计算距离？），并且是有序的（而上面这个0并不是说比1要高级）。但是，按照我们上述的表示，数字并不是有序的，而是随机分配的。

* 独热编码

为了解决上述问题，其中一种可能的解决方法是采用独热编码（One-Hot Encoding）。独热编码即 One-Hot 编码，又称一位有效编码，其方法是使用N位状态寄存器来对N个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候，其中只有一位有效。

例如：

自然状态码为：000,001,010,011,100,101

独热编码为：000001,000010,000100,001000,010000,100000

可以这样理解，对于每一个特征，如果它有m个可能值，那么经过独热编码后，就变成了m个二元特征（如成绩这个特征有好，中，差变成one-hot就是100, 010, 001）。并且，这些特征互斥，每次只有一个激活。因此，数据会变成稀疏的。

这样做的好处主要有：

1. 解决了分类器不好处理属性数据的问题
2. 在一定程度上也起到了扩充特征的作用

举例

基于Scikit-learn的例子：

from sklearn import preprocessing

enc = preprocessing.OneHotEncoder()

enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])

enc.transform([[0, 1, 3]]).toarray()

输出结果：

array([[ 1.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.]])

Note: fit了4个数据3个特征，而transform了1个数据3个特征。第一个特征两种值(0: 10, 1: 01)，第二个特征三种值(0: 100, 1: 010, 2: 001)，第三个特征四种值(0: 1000, 1: 0100, 2: 0010, 3: 0001)。所以转换[0, 1, 3]为[ 1.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.]。

參考資料：<https://blog.csdn.net/pipisorry/article/details/61193868>