厦门大学
XIAMEN UNIVERSITY

支持向量机

谭 忠

支持向量机历史发展

1963年，Vapnik在解决模式识别问题时提出了支持向量方法。起决定性作用的样本为支持向量 1971年，Kimeldorf构造基于支持向量构建核空间的方法 1995年，Vapnik等人正式提出统计学习理论。

目前支持向量机有着几方面的研究热点：

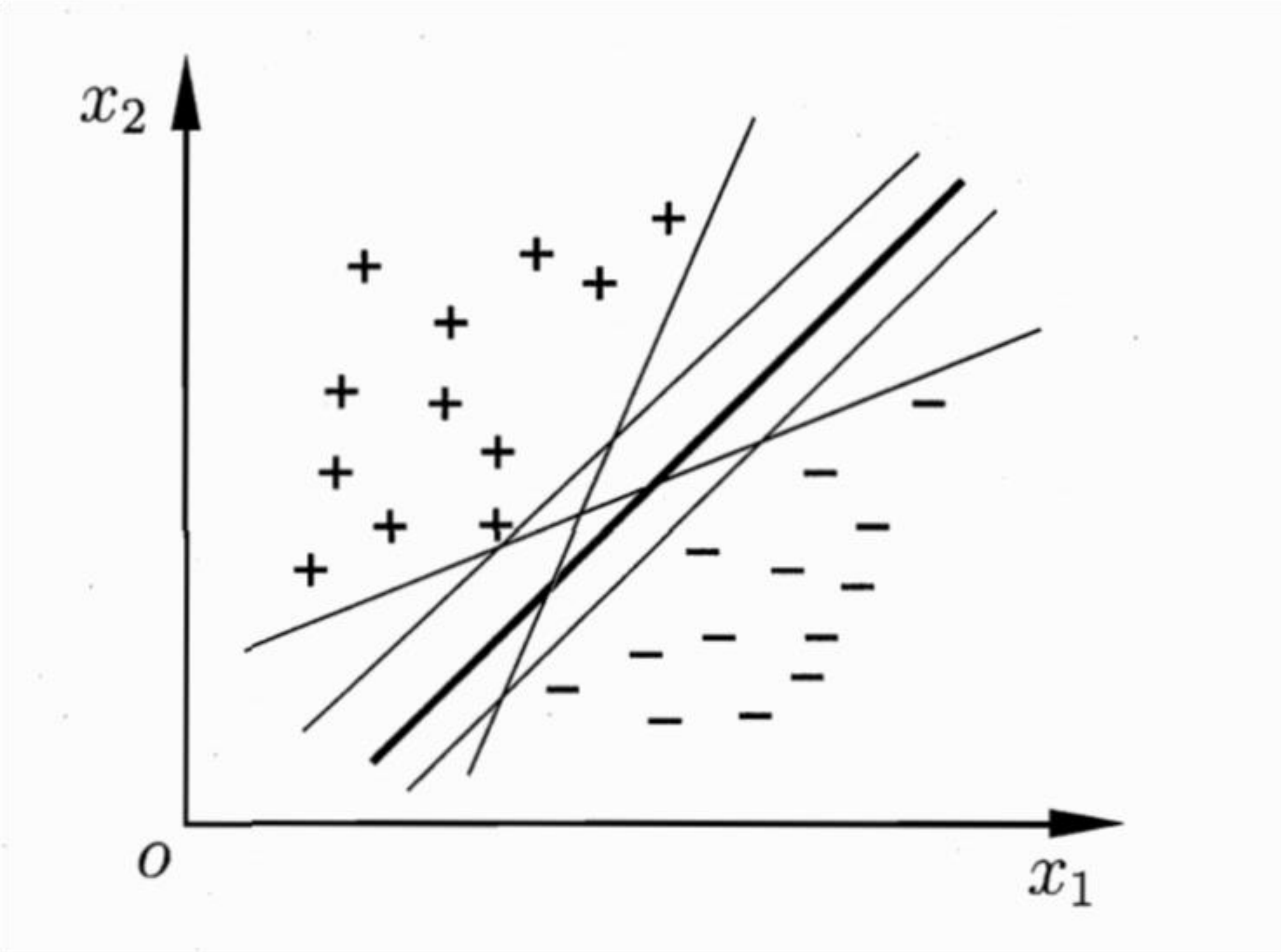
- 核函数的构造和参数的选择；
- 支持向量机从两类问题向多类问题的推广；
- 与目前其它机器学习方法的融合；
- 与数据预处理方面方法的结合，即数据本身的性质融入支持向量机的算法中从而产生新的算法；

支持向量机是基于统计学习的二类分类模型。它是一种监督学习方法，在学习过程中通过最大化分类间隔使得结构风险最小化。

那么支持向量机这个名字是怎么来的呢？

间隔与支持向量

给定训练样本集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}, y_i\in\{-1,+1\}$, 分类学习最基本的想法就是基于训练集 D 在样本空间中找到一个划分超平面, 将不同类别的样本分开. 但能将训练样本分开的划分超平面可能有很多, 如图所示, 我们应该努力去找到哪一个呢?



存在多个划分超平面将两类训练样本分开

直观上看, 应该去找位于两类训练样本“正中间”的划分超平面, 即图中粗的那个, 因为该划分超平面对训练样本局部扰动的“容忍”性最好.

例如，由于训练集的局限性或噪声的因素，训练集外的样本可能比图中的训练样本更接近两个类的分隔界，这将使许多划分超平面出现错误，而红色的超平 面受影响最小. 换言之，这个划分超平面所产生的分类结果是最鲁棒的，对示例的泛化能力是最强的.

在样本空间中，划分超平面可通过如下线性方程来描述:

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0,$$

其中 $\boldsymbol{w} = (w_1; w_2; \ldots; w_d)$ 为法向量，决定了超平面的方向; b 为位移项，决定了超平面与原点之间的距离. 显然，划分超平面可被法向量 $\boldsymbol{w}$ 和位移 b 确定,

下面我们将其记为 $(\boldsymbol{w}, b)$. 样本空间中任意点 $\boldsymbol{x}$ 到超平面 $(\boldsymbol{w}, b)$ 的距离可写为

$$r = \frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b|}{\|\boldsymbol{w}\|}.$$

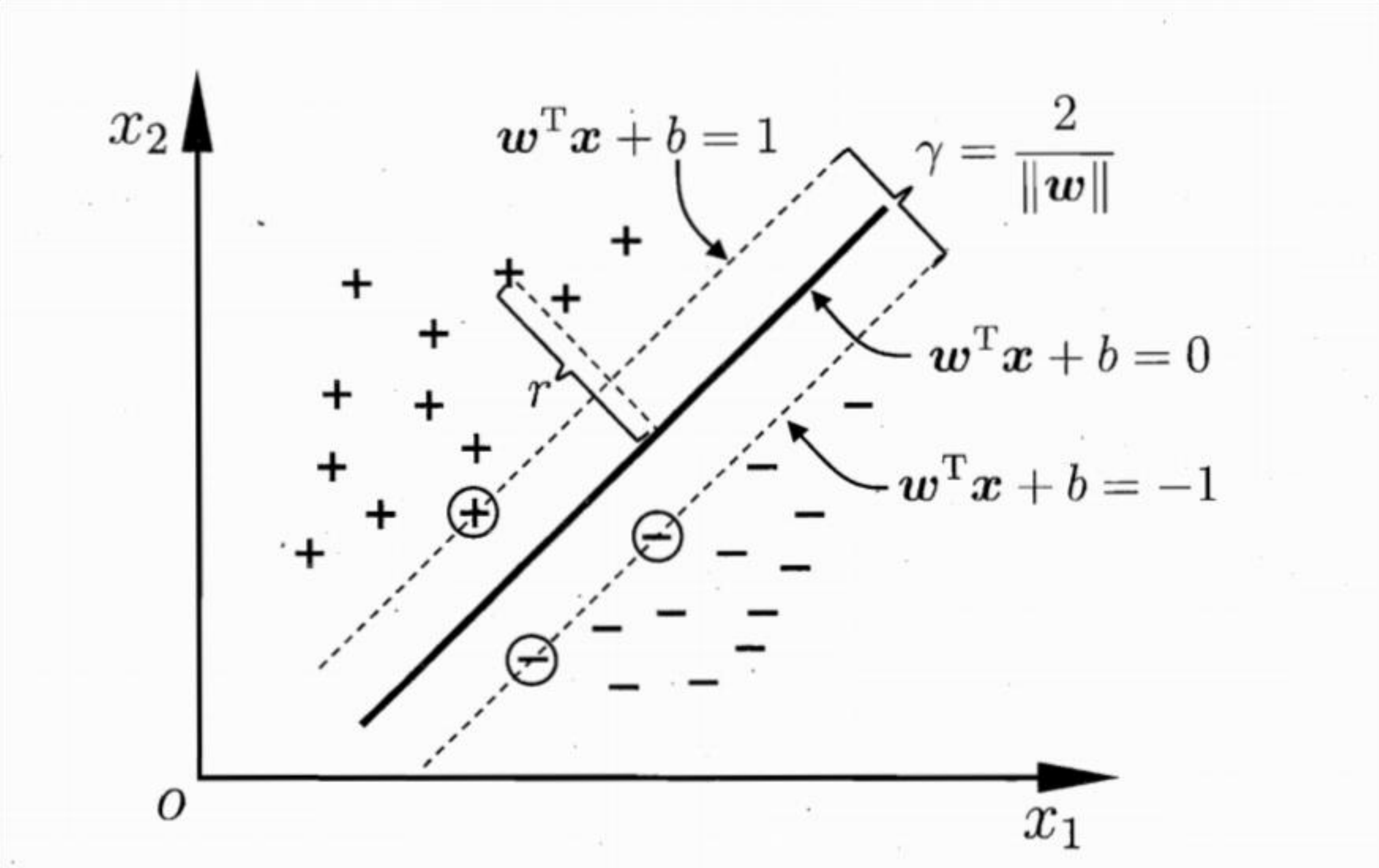
假设超平面 $(\boldsymbol{w}, b)$ 能将训练样本正确分类, 即对于 $(\boldsymbol{x}_i, y_i) \in D$, 若 $y_i = +1$, 则有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b > 0$; 若 $y_i = -1$, 则有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b < 0$. 令

$$\begin{cases} \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geqslant +1, & y_i = +1 \\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leqslant -1, & y_i = -1 \end{cases}$$

如下图所示, 距离超平面最近的这几个训练样本点使上式的等号成立, 它们被称为 “支持向量” (support vector), 两个异类支持向量到超平面的距离 之和为

$$\gamma = \frac{2}{\| \boldsymbol{w} \|}$$

它被称为 “间隔” (margin).


x_2
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=1$
$\gamma=\frac{2}{\|\boldsymbol{w}\|}$
r
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=-1$
O
x_1

欲找到具有 “最大间隔” (maximum margin)的划分超平面, 也就是要找 到能满足式中约束的参数 $\boldsymbol{w}$ 和 b, 使得 γ 最大, 即

$$\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|}$$

$$\text{s.t. } y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geqslant 1,\ i = 1,2,\dots,m.$$

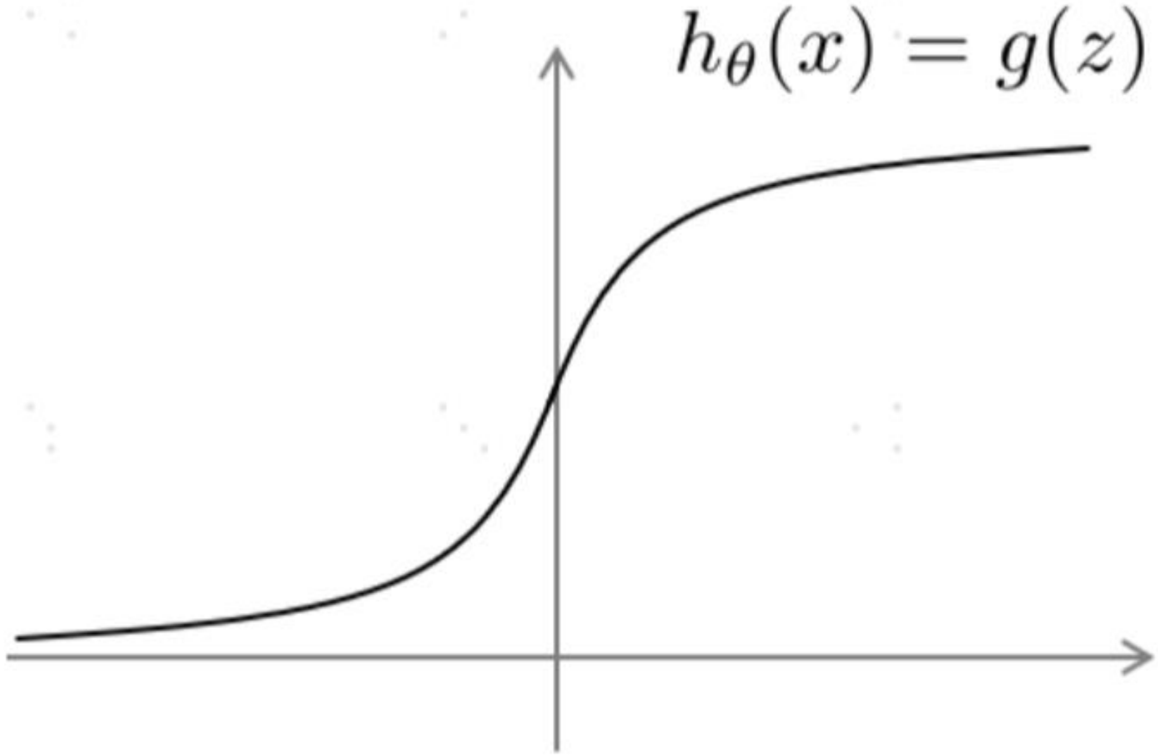
显然, 为了最大化间隔, 仅需最大化 $\|\boldsymbol{w}\|^{-1}$, 这等价于最小化 $\|\boldsymbol{w}\|^2$. 于是上页式子可重写为

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geqslant 1,\ i = 1,2,\ldots,m.$$

这就是支持向量机(Support Vector Machine, 简称 SVM)的基本型.

接下来我们从另一个角度切入，回顾一下逻辑回归，看看逻辑回归是如何演变为支持向量机的。



$$z = \theta^T x$$

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

当y=1时，如果我们希望$h_0(x) \approx 1$，则$\theta^T x$远大于0

当y=0时，如果我们希望$h_0(x) \approx 0$，则$\theta^T x$远大于0

下面是每个样本的代价函数，注意没有求和，代表每个单独的训练样本对逻辑回归的总体目标函数的贡献。

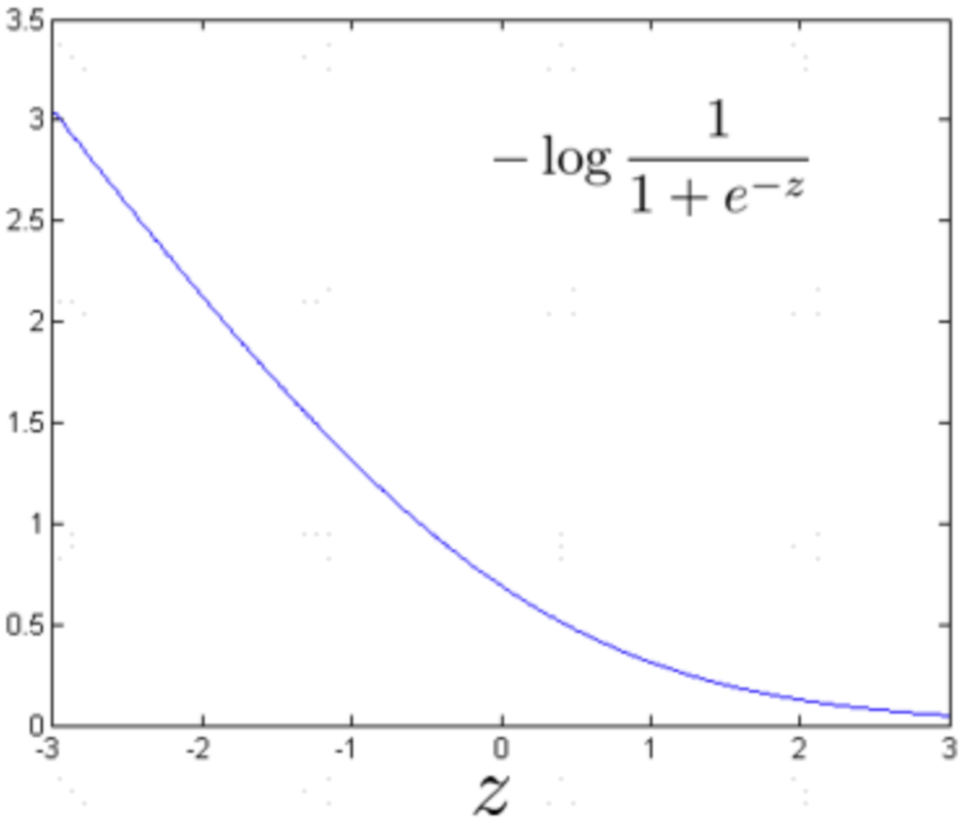
$$-\left(y\log h_\theta(x) + (1-y)\log(1-h_\theta(x))\right)$$

然后我们将$h_0(x)$的具体公式带入进去，得到的就是每个训练样本对总体函数的具体贡献：

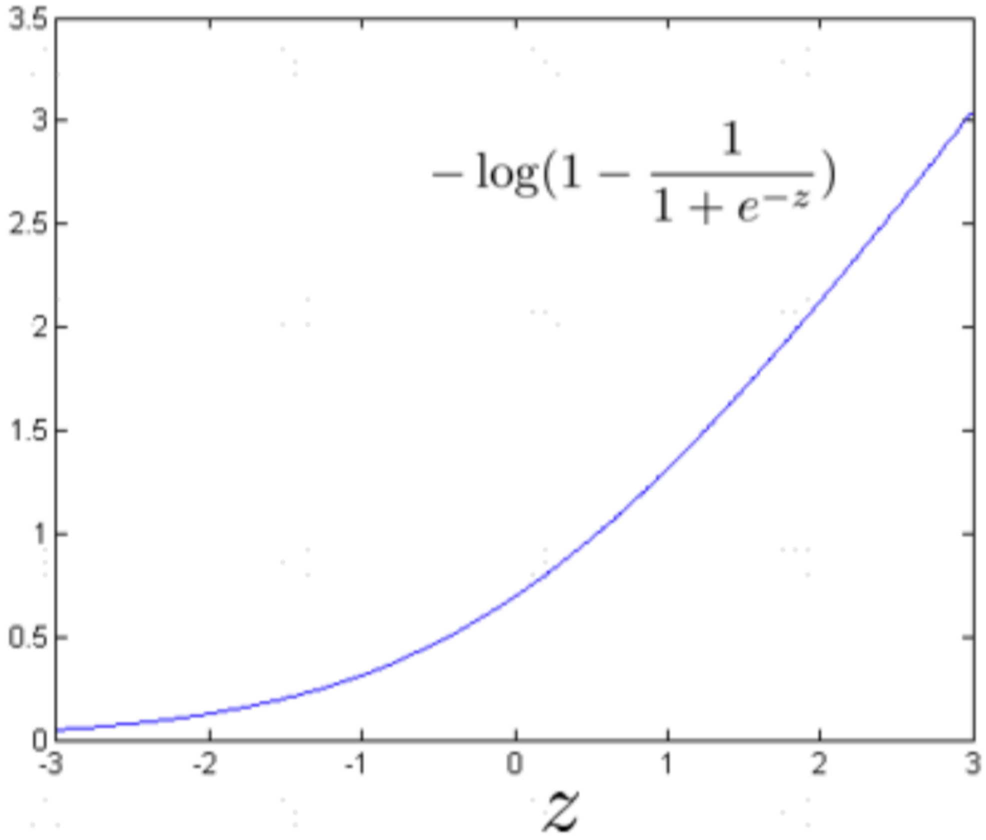
$$-y\log\frac{1}{1+e^{-\theta^T x}}-(1-y)\log\left(1-\frac{1}{1+e^{-\theta^T x}}\right)$$

现在我们再来考虑y = 1 , y = 0 的情况，函数图像如下：

$y = 1\ \theta^T x \gg 0$

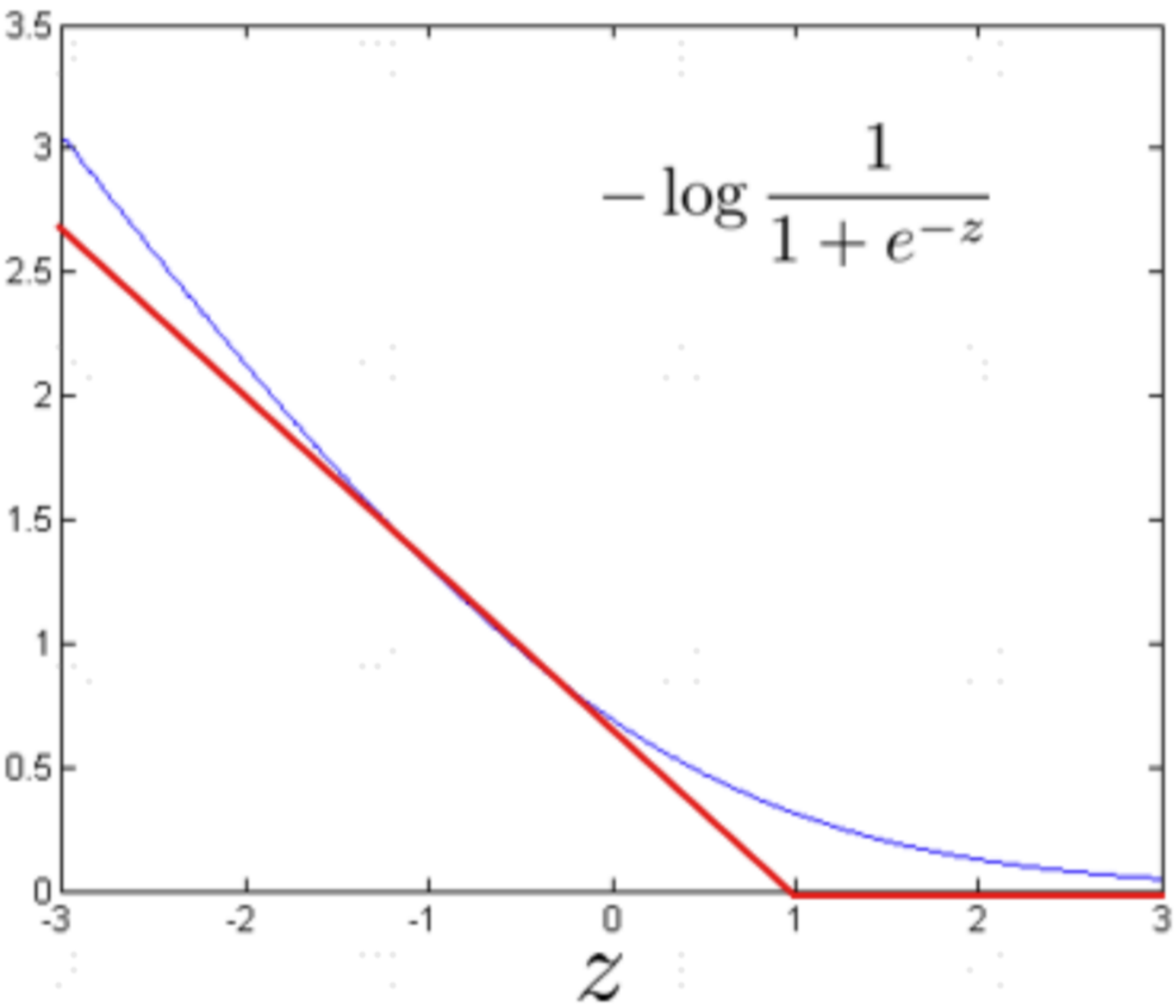


$y = 0\ \theta^T x \ll 0$

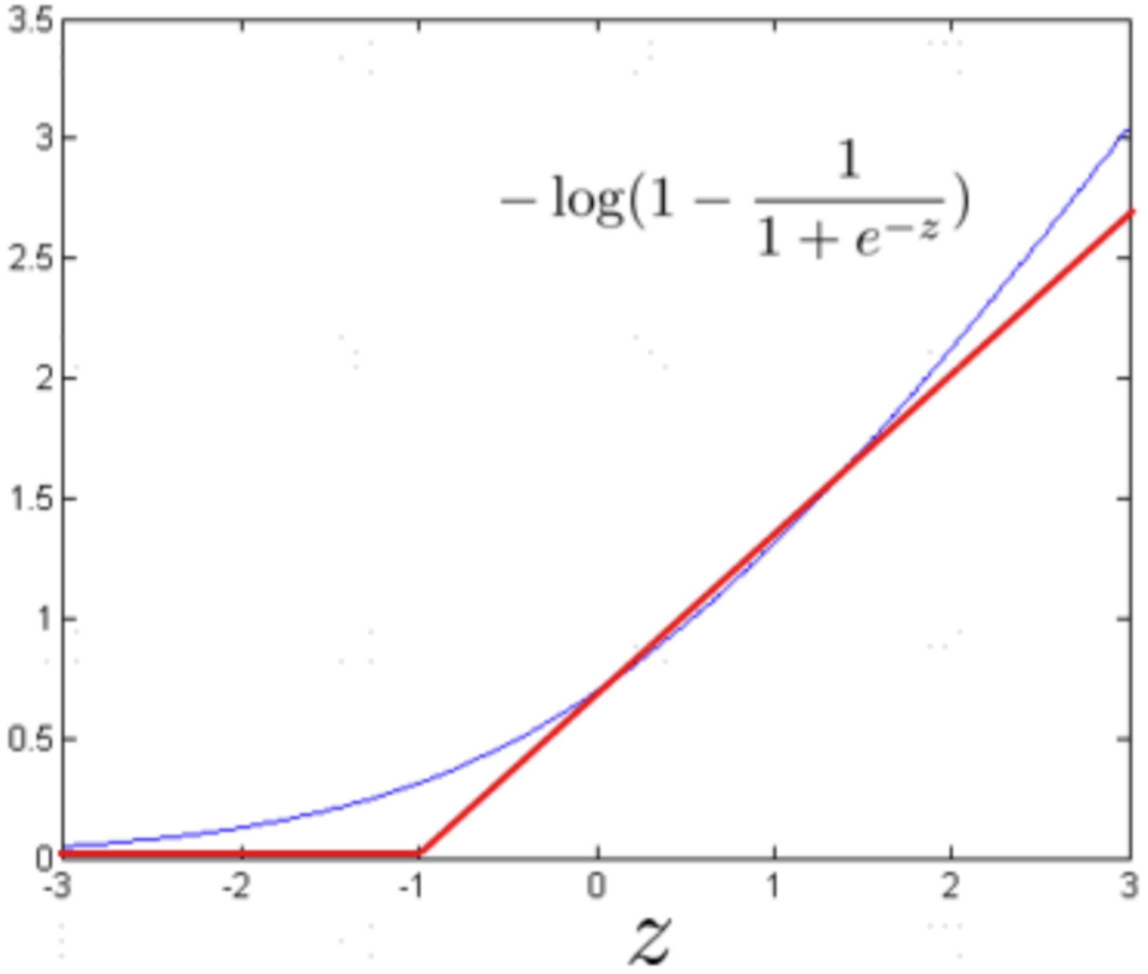


下面我们y = 为例，用两条直线近似等效曲线，来向支持向量机转换，例如我以z = 1 为起点，作两条直线近似取代曲线，同理y = 0 时也一样。

$$-\log \frac{1}{1+e^{-z}}$$



当 $y=1$ 时，两条直线记为 $\mathrm{Cost}_1(z)$ 。



当 $y=0$ 时，两条直线记为 $\mathrm{Cost}_0(z)$

构建支持向量机

这是我们在Logistic回归中使用的正规化代价函数 $J(\theta)$

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

然后我们用Cost $\mathrm{Cos}_1\left(\theta^{\mathrm{T}}\mathrm{x}^{(\mathrm{i})}\right.$ 和 $\mathrm{Cost}_0\left(\theta^{\mathrm{T}}\mathrm{x}^{(\mathrm{i})}\right)$ 将 $-\log\mathrm{h}_\theta\left(\mathrm{x}^{(\mathrm{i})}\right)$ 和 $-\log\left(1-\mathrm{h}_\theta\left(\mathrm{x}^{(\mathrm{i})}\right)\right.$ 代替，去掉 $\frac{1}{\mathrm{m}}$ ，然后对于正规项，我们不再用 λ 来控 制正规项的权重，而选择用不同的常数C来控制第一项的权重

最后我们得到支持向量机的总体优化目标如下:

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} \mathrm{cost}_1\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) \mathrm{cost}_0\left(\theta^T x^{(i)}\right) \right]$$

$$+ \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

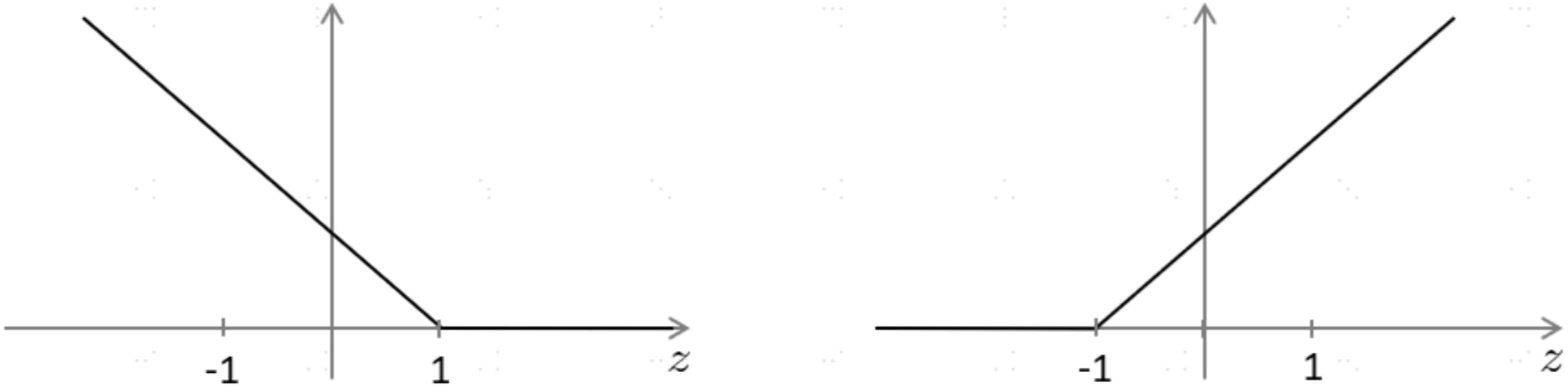
与Logistic回归不同的是，sigmoid函数输出的不是概率，而是直接输出0或者1。

直观理解SVM

这是SVM的代价函数和图像:

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}\mathrm{cost}_1\left(\theta^T x^{(i)}\right)+\left(1-y^{(i)}\right)\mathrm{cost}_0\left(\theta^T x^{(i)}\right)\right]$$

$$+\frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$



如果$y=1$, 我们希望 $\theta^T x \geq 1$ （不仅仅是 ≥ 0)

如果$y=0$, 我们希望 $\theta^T x \leq -1$ (不仅仅是 < 0)

下面我们来想一下如何让代价函数最小化。

若 $y = 1$，则当 $\theta^{\mathrm{T}}\mathrm{x} \geq 1$ 时， $\mathrm{Cost}_1(\mathrm{z}) = 0.$

若 $\mathrm{y} = 0$，则当 $\theta^{\mathrm{T}}\mathrm{x} \leq -1$ 时， $\mathrm{Cost}_2(\mathrm{z}) = 0.$

下面我们想象一下，如果将常数C设得比较大，例如C=100000，那么当进行最小化时，我们将迫切希望找到一个合适的值，使第一项等 于0，那么现在我们试着在这种情况下来理解优化问题。

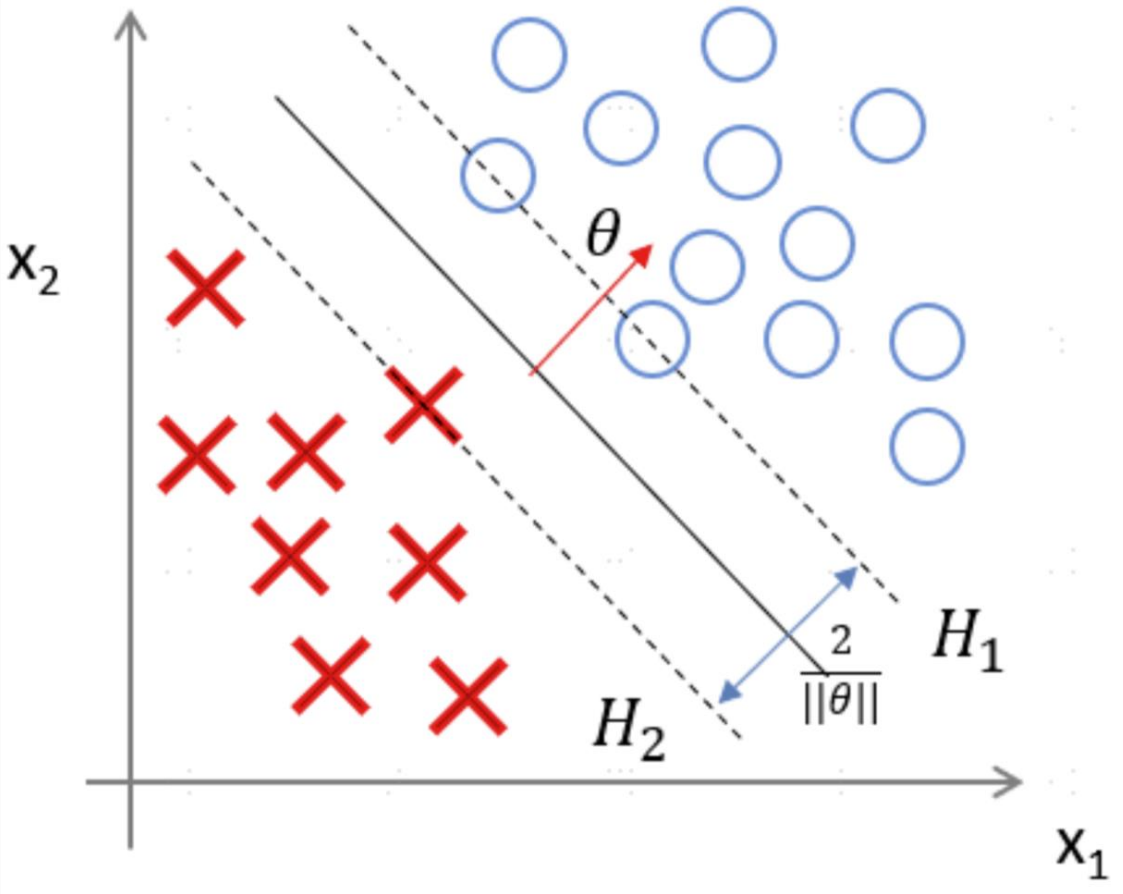
$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} \mathrm{cost}_1\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) \mathrm{cost}_0\left(\theta^T x^{(i)}\right) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

要使第一项为 0，则有以下两种情况:

若 $y=1$，则 $\theta^{T}x \geq 1$，即 $y=1$ 的样本点在超平面 $H_1: \theta^{T}x \geq 1$ 上。

若 $y=0$，则 $\theta^{T}x \leq -1$，即 $y=0$ 的样本点在超平面 $H_2: \theta^{T}x \leq -1$ 上。

如下图所示，在 H_1、H_2 上的点就是支持向量:

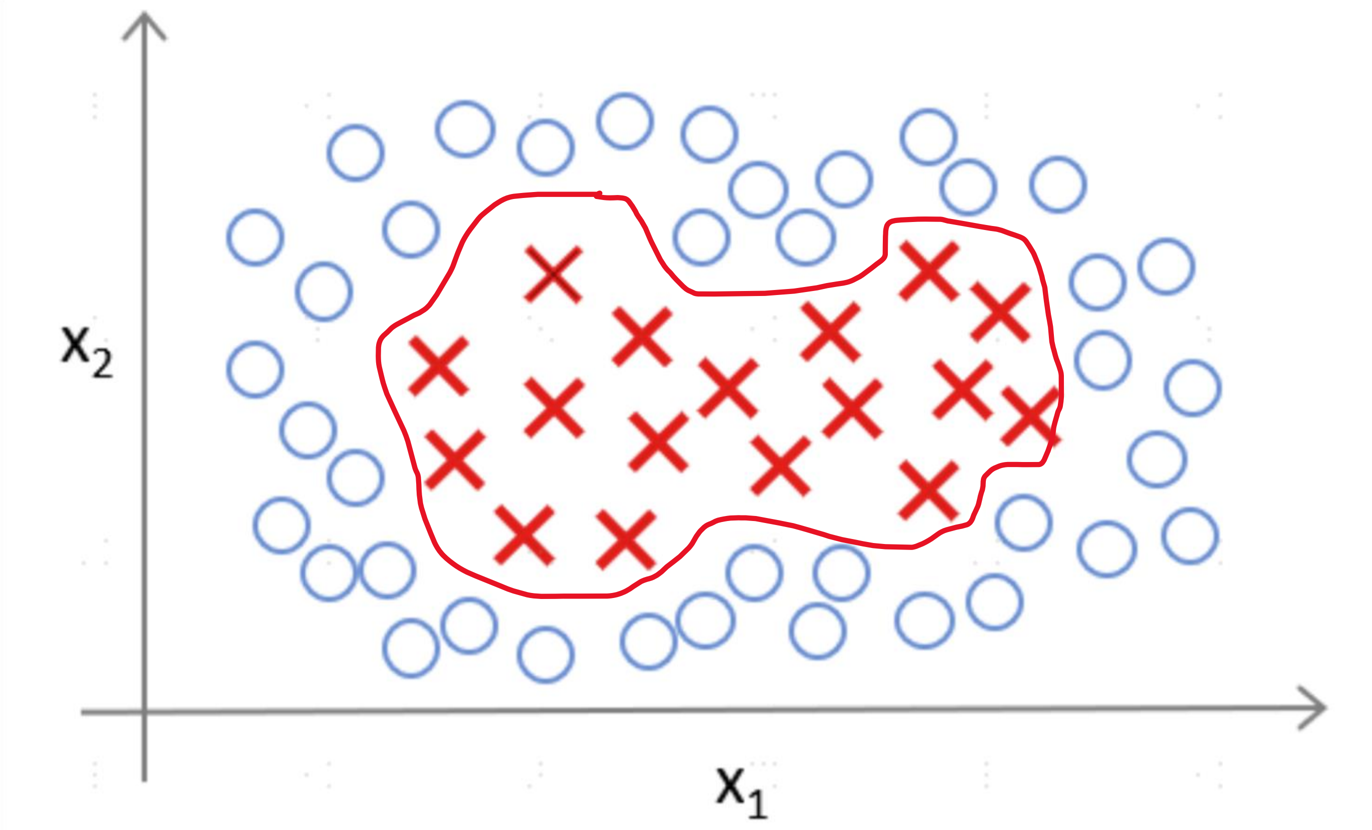


这里两个超平面 H_1、H_2 平行，它们中间没有样本点。H_1、H_2 之间的距离成为间隔。

间隔依赖于分离超平面的法向量 θ ，等于 $\frac{2}{\|\theta\|}$。 H_1、H_2 是间隔边界。

简单介绍一下核函数

如下图，我们需要得到一个非线性的决策边界：



按我们之前学的方法，可以通过增加项数来进行拟合，如下:

$$\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1x_2$$
$$+\theta_4x_1^2+\theta_5x_2^2+\cdots\geq 0$$

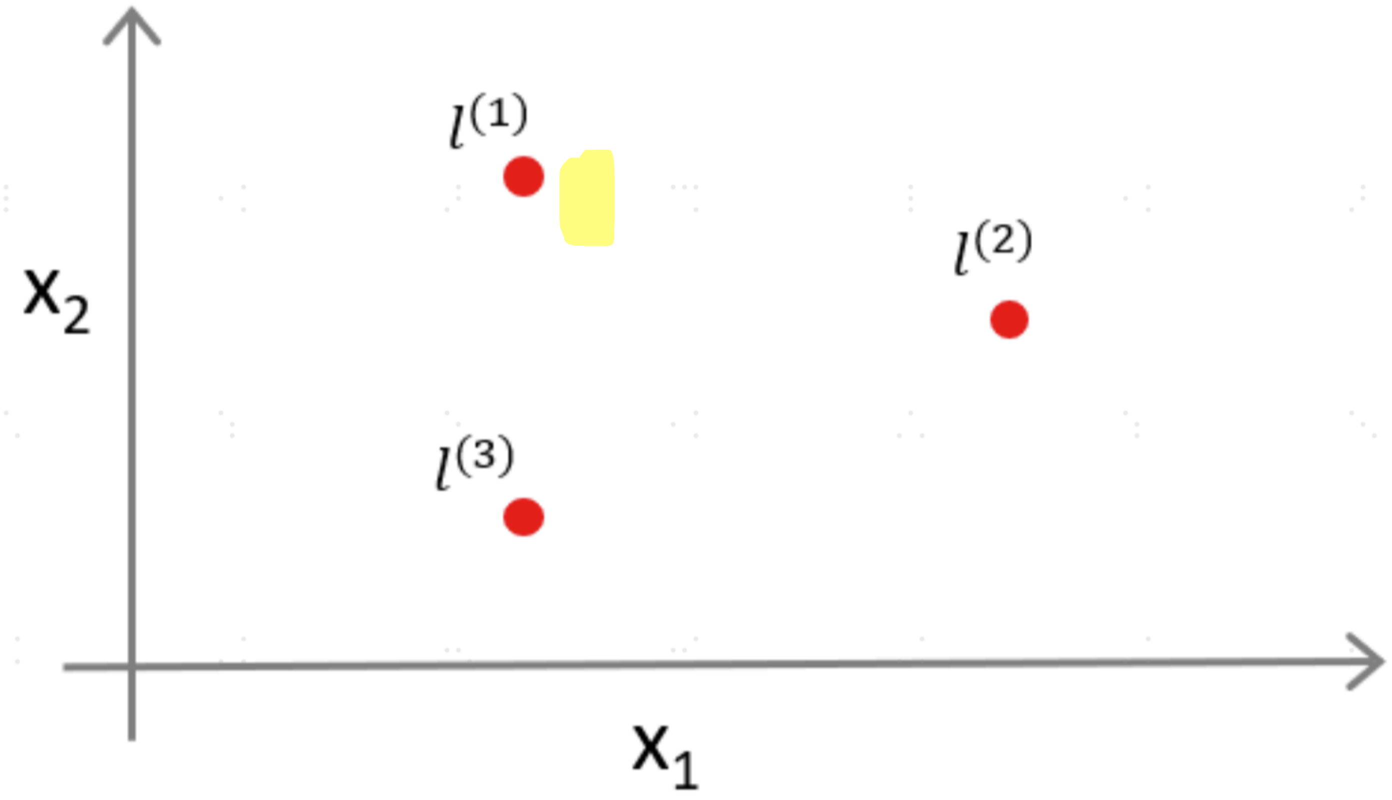
现在我们用一些新的符号 f_1, f_2, f_3 … 来表示新的特征值:

$$\theta_0 + \theta_1 \mathrm{f}_1 + \theta_2 \mathrm{f}_2 + \theta_3 \mathrm{f}_3 + \theta_4 \mathrm{f}_4 + \theta_5 \mathrm{f}_5 + \cdots \geq 0$$

$$\mathrm{f}_1 = \mathrm{x}_1, \mathrm{f}_2 = \mathrm{x}_2, \mathrm{f}_3 = \mathrm{x}_1 \mathrm{x}_2, \mathrm{f}_4 = \mathrm{x}_1^2 \ldots$$

现在我们用 f_1, f_2, f_3 来举例:

如图，我们在图上选择三个标记 $l^{(1)}, l^{(2)}, l^{(3)}$



然后来定义新的特征:

给定一个实例 x ，然后将 f_1 定义为度量实例 x 与标记点 $l^{(1)}$ 的相似度

$$\mathrm{f}_1 = \mathrm{similarity}\left(\mathrm{x}, \mathrm{l}^{(1)}\right) = \exp\left(-\frac{\left\|\mathrm{x} - \mathrm{l}^{(1)}\right\|^2}{2\sigma^2}\right)$$

类似地,

$$f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\left\|x - l^{(2)}\right\|^2}{2\sigma^2}\right)$$

$$f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(-\frac{\left\|x - l^{(3)}\right\|^2}{2\sigma^2}\right)$$

这种函数我们称为高斯核函数

下面来看看这些核函数的表达式有什么含义。

假设现在有一点非常接近与标记点 $l^{(1)}$，那么欧氏距离 $\|x-l^{(1)}\|^2$ 就会接近于 0，此时 $f_1 \approx \exp(0)=1$。

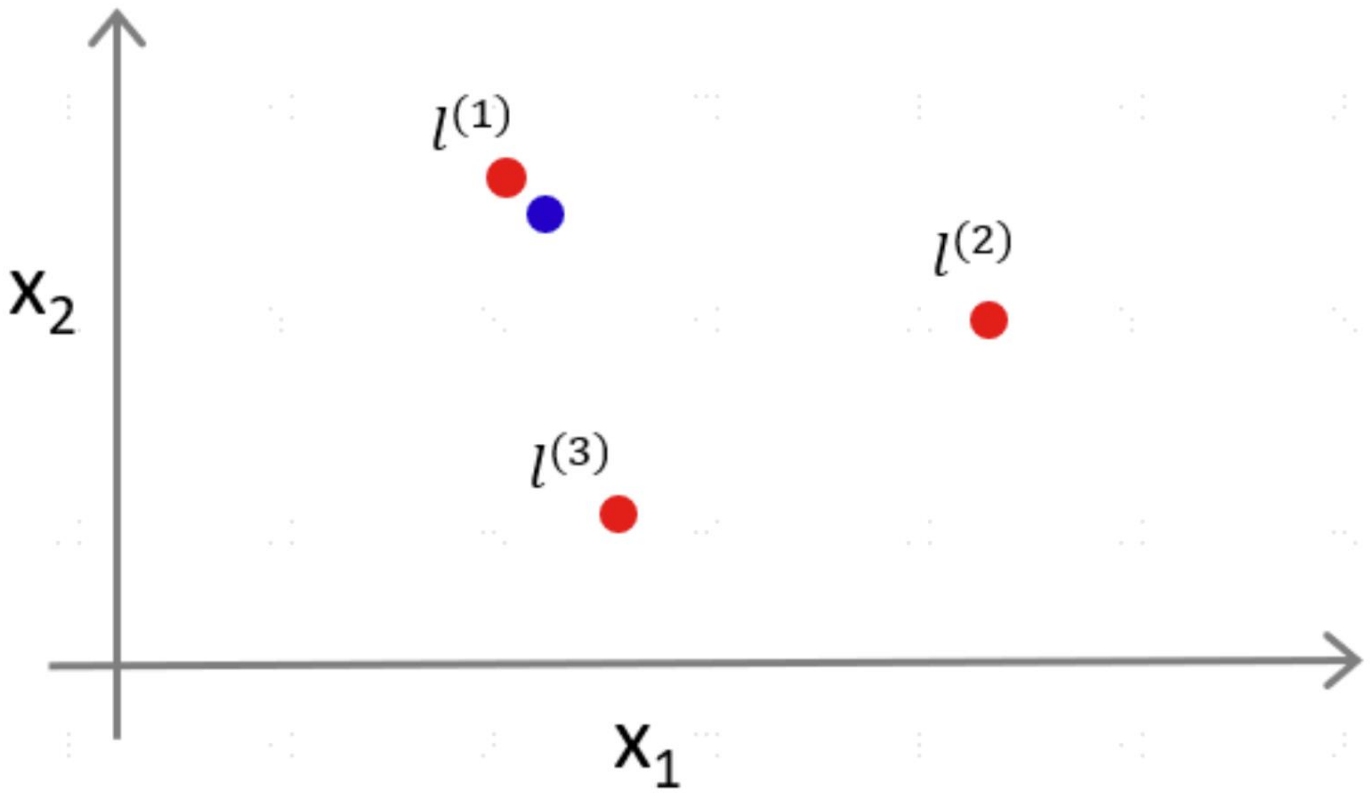
相反，如果这点离 $l^{(1)}$ 很远，欧式距离 $\|x-l^{(1)}\|^2$ 会变得很大，此时 $f_1 \approx 0$。

讲完了特征值的定义，接下来我们看看核函数是如何应用于决策边界的。给定一个训练样本，当 $\theta_0+\theta_1\mathrm{f}_1+\theta_2\mathrm{f}_2+\theta_3\mathrm{f}_3\geq 0$ 时，预测 $\mathrm{y}=1$ 。

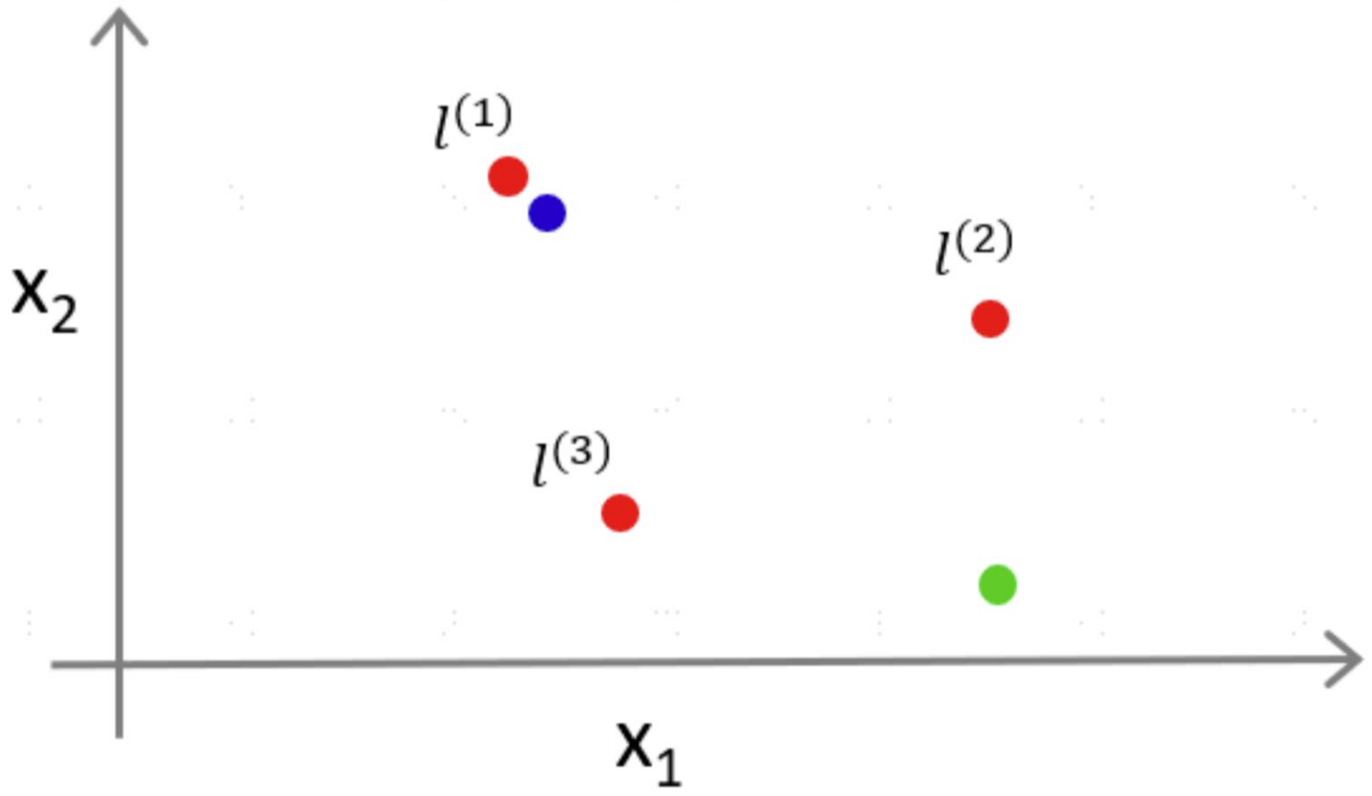
假设我们已经得到了参数 θ 的值:

$$\theta_0=-0.5,\theta_1=1,\theta_2=1,\theta_3=0$$

现在我们有一个实例 x （蓝点），落在如图所示位置，显然，该实例与标记点 $l^{(1)}$ 间距离很近，故 $f_1 = 1$，与标记点 $l^{(2)}, l^{(3)}$ 相距较远，故 $f_2, f_3 = 0$，然后我们代入 $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3$ 得 $\theta_0 + \theta_1 = 0.5 > 0$，所以预测 $y = 1$。



若一个实例如绿点所示，与 $l^{(1)}, l^{(2)}, l^{(3)}$ 的距离都很远，此时 $f_1, f_2, f_3 = 0$ 代入 $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3$ 得 $\theta_0 = -0.5 < 0$ ，所以预测 $y = 0$ 。



如此，便会得到一个可以区分正负样本的非线性的决策边界。

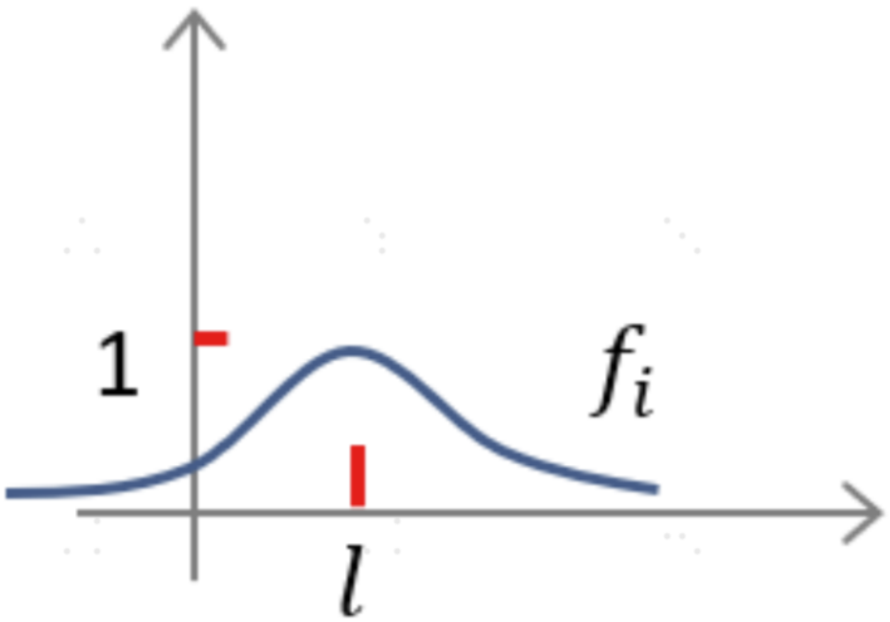
那么现在大家可能会想如何去得到我们的标记点 $l^{(1)}, l^{(2)}, l^{(3)}$ ，并且在一些复杂的分类问题中，也许我们需要更多的标记点。

一般情况下，我们会直接选择训练样本作为标记点。

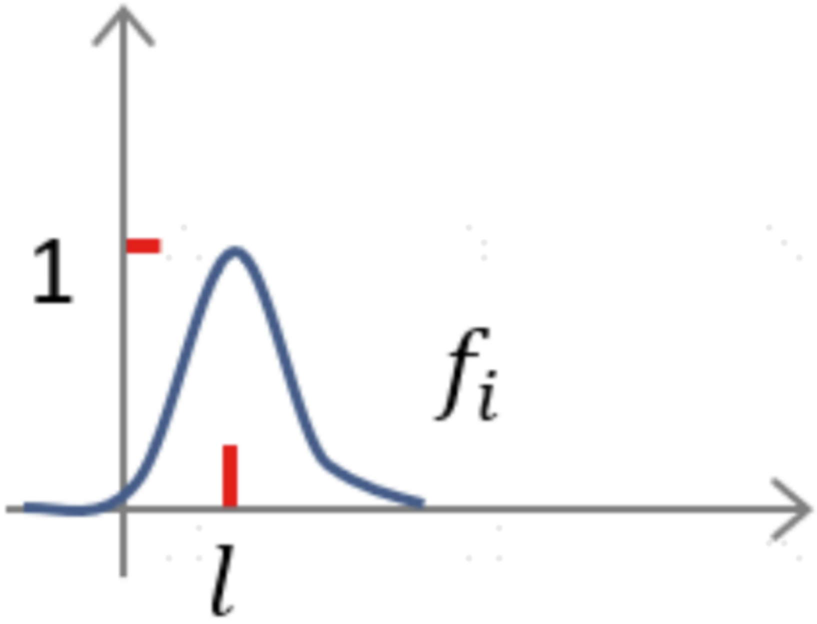
参数选择

参数 σ^2 ，如果 σ^2 比较大，则高斯核函数

$\exp\left(-\frac{\|\mathrm{x}-1^{(\mathrm{i})}\|^2}{2\sigma^2}\right)$ 相对平滑，模型高偏差低方差。反之则相对陡峭，模型低偏差高方差。



σ^2 较大



σ^2 较小

XIAMEN UNIVERSITY
厦门大学

几种常用的核函数

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\\|\boldsymbol{x}_i - \boldsymbol{x}_j\\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\\|\boldsymbol{x}_i - \boldsymbol{x}_j\\|}{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数, $\beta > 0$, $\theta < 0$

实战案例

SVM算法API：

classifier=svm.SVC(C=2,kernel='rbf',gamma=10,decision_function_shape='ovr’）

- C为惩罚系数，即对误差的宽容度。
- kernel为核函数
- gamma是选择径向基函数（RBF）作为kernel后， 该函数自带的一个参数。隐含地决定了数据映射到新的特征空间后的分布，gamma越大，支持向量越少，gamma越小，支持向量越多。

我们以python自带鸢尾花数据集为例，用SVM算法进行分类预测。

1、加载python库

2、划分训练集、测试集

3、模型构建SVM

4、模型预测

```python
#加载python库
import numpy as np
from sklearn import svm
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
#1、加载数据集
iris = load_iris()
n_sample,n_features=iris.data.shape
print((n_sample,n_features))
```

```python
#2、划分训练集、测试集
train_data,test_data=train_test_split(iris.data,random_state=1,train_size=0.7,test_size=0.3)
train_label,test_label=train_test_split(iris.target,random_state=1,train_size=0.7,test_size=0.3)
```

```
#3、模型构建SVM
classifier=svm.SVC(C=2,kernel='rbf',gamma=10,decision_function_shape='ovr')#核函数选用rbf，即高斯核函数
classifier.fit(train_data,train_label.ravel())
```

```
#4、模型预测
pre_train=classifier.predict(train_data)#训练集预测标签
pre_test=classifier.predict(test_data)#测试集的预测标签
print("train:",accuracy_score(train_label,pre_train))
print("test:",accuracy_score(test_label,pre_test))
```

结果如下：

```
(150, 4)
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
train: 1.0
test: 0.9333333333333333
```

实战算例2：用make blobs生成数据

1、创造数据集函数 make_blobs

x为特征y为标签

X, y = make_blobs(n_samples=200, centers=2, random_state=0, cluster_std=0.3)

2、模型训练

clf = svm.SVC(kernel= ‘linear’,C=1.0) #这里用的线性核函数

clf.fit(X, y)

3、画图

（1）找出x轴，y轴的长度，x[:,0].min()+1 x[:.0].man()+1

（2）生成坐标矩阵

numpy.meshgrid()生成网格点坐标矩阵[X,Y] = meshgrid(x,y) 将向量x和y定义的区域转换成矩阵X和Y,其中矩阵X的行向量是向量x的简单复制，而矩阵Y的列向量是向量y的简单复制。

假设x是长度为m的向量，y是长度为n的向量，则最终生成的矩阵X和Y的维度都是 nm （注意不是mn）np.linspace主要用来创建等差数列np.arange函数返回一个有终点和起点的固定步长的排列np.c_给numpy数组添加列np.r_给numpy数组添加行

XIAMEN UNIVERSITY

4、生成数据做图

np.ravel()将采样点的x坐标摊平,

np.r_是按列连接两个矩阵，就是把两矩阵上下相加，要求列数相等。

np.c_是按行连接两个矩阵，就是把两矩阵左右相加，要求行数相等。

xx.shape表示一共有多少个元素

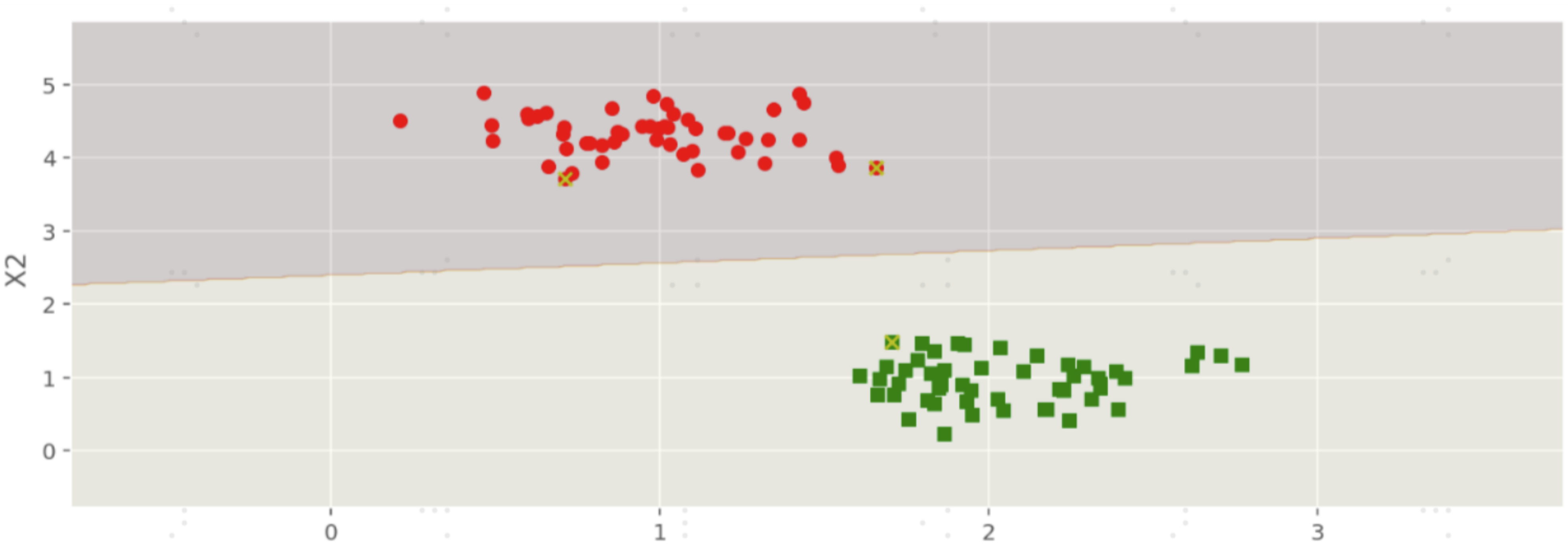
Z.reshape作用就是把数据原来的尺寸更改为我们想要的尺寸

5、绘制

plt.contourf(xx, yy, Z, cmap='hot', alpha=0.5)等高线作图

np.unique(y)该函数是去除数组中的重复数字，并进行排序之后输出

结果如下:



总结

SVM优点：

- 1.支持多维空间
- 2.不同核函数用于不同决策函数
- 3.处理多维度数据分类，小样本数据可以工作

SVM缺点：

- 1.如果数据特征（维度）大于样本量，支持向量机表现很差
- 2.支持向量机不提供概率区间估计
- 3.找到准确的核函数和C参数，gamma参数需要很大计算量

谢谢