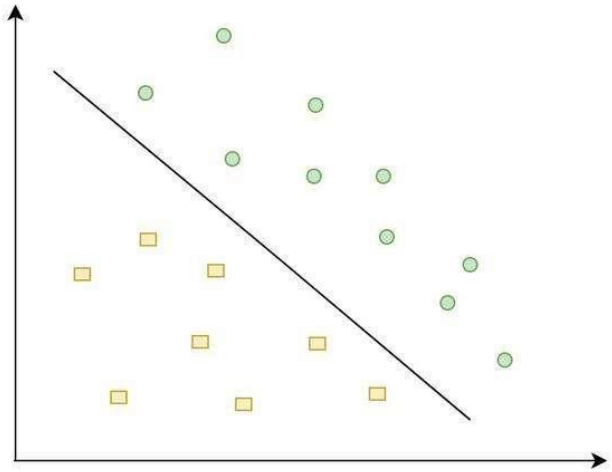
支持向量机

1、线性可分

通俗的讲，在二维空间上，两类点被一条直线完全分开叫做线性可分，如图所示：



最大间隔超平面

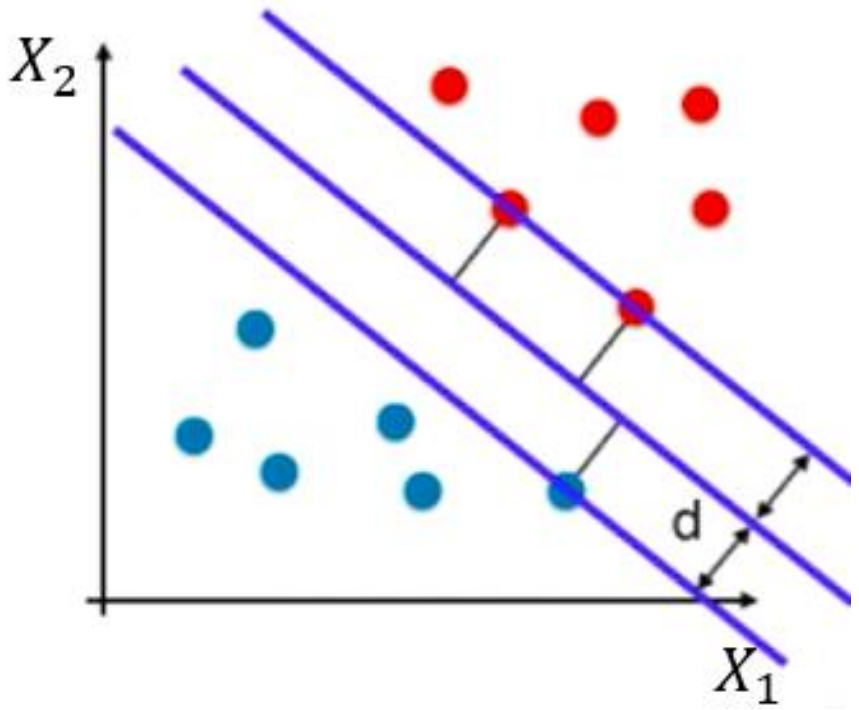
从二维分类扩展到高维的分类情形时，我们分类的线就变为了超平面,我们会去找最佳超平面，以最大间隔把两类样本分开的超平面，也称之为最大间隔超平面。

最大超平面应该具有的性质为：

(1)两类样本分别分割在该超平面的两侧；

(2)两侧距离超平面最近的样本点到超平面的距离最大化。

支持向量：样本中距离超平面最近的一些点，这些点叫做支持向量。



求超平面的优化问题：

SVM想要的就是找到各类样本点到超平面的距离最远，也就是找到最大间隔超平面。任意超平面可以用下面这个线性方程来描述：$w^T x+b=0$

二维空间点 (x, y) 到直线 A x+B y+C=0 的距离公式是:

$$\frac{|Ax+By+C|}{\sqrt{A^2+B^2}}$$

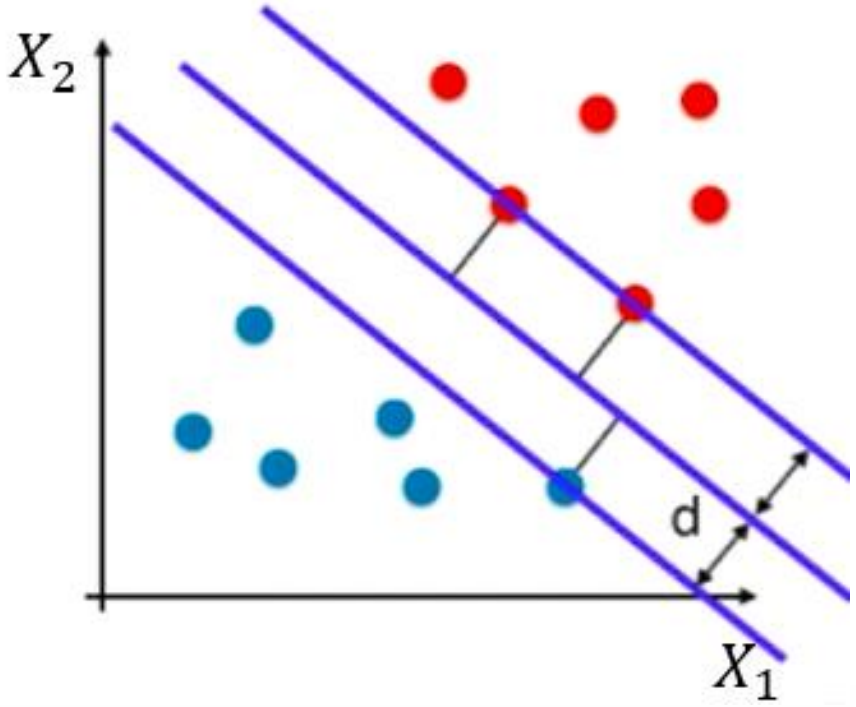
扩展到n维空间后, 点 $x=(x_1, x_2 \ldots x_n)$ 到直线 $w^T x+b=0$ 的距离为:

$$\frac{|w^T x+b|}{\| w\|}$$

其中

$$\| w\| = \sqrt{w_1^2 + \ldots w_n^2}$$

如下图所示，根据支持向量的定义我们知道，支持向量到超平面的距离为d，其他点到超平面的距离大于d。



于是我们有这样的一个公式:

$$\begin{cases} \dfrac{w^T x+b}{\| w\|} \geq d & y=1 \\ \dfrac{w^T x+b}{\| w\|} \leq -d & y=-1 \end{cases}$$

稍作转化可以得到:

$$\begin{cases} \dfrac{w^T x+b}{\| w\| d} \geq 1 & y=1 \\ \dfrac{w^T x+b}{\| w\| d} \leq -1 & y=-1 \end{cases}$$

$\| w\| d$ 是正数，我们暂且令它为 1

故:

$$\begin{cases} w^T x + b \geq 1 & y = 1 \\ w^T x + b \leq -1 & y = -1 \end{cases}$$

将两个方程合并，我们可以简写为:

$$y\left(w^T x + b\right) \geq 1$$

至此我们就可以得到最大间隔超平面的上下两个超平面:

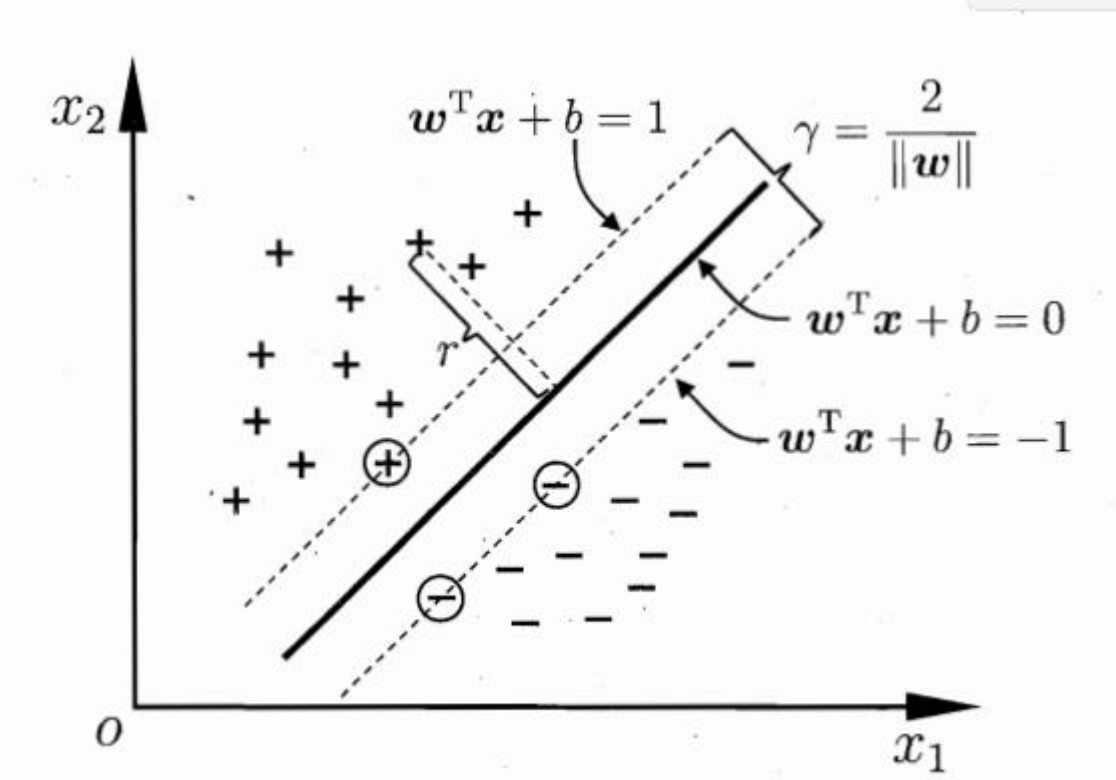


图 6.2 支持向量与间隔

分别为：

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}+b=-1$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}+b=1$$

每个支持向量到超平面的距离可以写为：

$$d = \frac{\left| w^T x + b \right|}{\| w \|}$$

由上述 $y\left(w^T x + b\right) > 1 > 0$ 可以得到 $y\left(w^T x + b\right) = \left| w^T x + b \right|$，所以我们得到：

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

最大化这个距离：

$$\max 2 * \frac{y\left(w^T x+b\right)}{\|w\|}$$

这里乘上 2 倍也是为了后面推导，对目标函数没有影响。刚刚我们得到支持向量 $y\left(w^T x+b\right)=1$，所以我们得到：

$$\max \frac{2}{\|w\|}$$

再做一个转换:

$$\min \frac{1}{2}\| w\|$$

为了方便计算 (去除$\| w\|$的根号)，我们有:

$$\min \frac{1}{2}\| w\|^2$$

所以得到的最优化问题是:

$$\min \frac{1}{2}\| w\|^2 \text{ s.t. } \quad y_i\left(w^T x_i + b\right) \geq 1$$

通过对偶可以得到：

$$\max_{\lambda}\left[\sum_{j=1}^{n}\lambda_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j\left(x_i\cdot x_j\right)\right]$$

$$\text{s.t.}\quad \sum_{i=1}^{n}\lambda_i y_i=0 \quad \lambda_i\geq 0$$

通过求解

$$w=\sum_{i=1}^{m}\lambda_i y_i x_i$$

而 $\lambda_i>0$ 对应的点都是支持向量，我们可以随便找个支持向量，然后带入：

$y_s\left(wx_s+b\right)=1$求出 b即可。

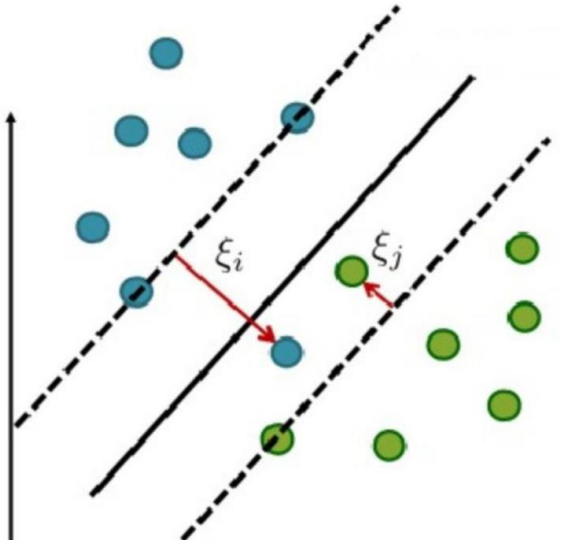
因此可以得到分类决策函数：$f(x)=\mathrm{sign}\left(w^T x+b\right)$ 其中 $\mathrm{sign}(\cdot)$ 为阶跃函数:

$$\mathrm{sign}(x)=\begin{cases}-1 & x<0\\ 0 & x=0\\ 1 & x>0\end{cases}$$

将新样本点导入到决策函数中既可得到样本的分类。

软间隔

在实际应用中，完全线性可分的样本是很少的，如果遇到了不能够完全线性可分的样本，因此我们需要对上面的模型进行优化，我们采用了了软间隔，相比于硬间隔的苛刻条件，我们允许个别样本点出现在间隔带里面，比如:



我们允许部分样本点不满足约束条件：

$$1-y_i\left(w^T x_i+b\right)\le 0$$

即对于这部分点可以为$1-y_i\left(w^T x_i+b\right)>0$但是不是所有的的都可以这样，所以这个软间隔需要有一定的限制，我们为每个样本引入一个松弛变量ξ_i，令$\xi_i\ge 0$，且$1-y_i\left(w^T x_i+b\right)-\xi_i\le 0$。

所以，增加软间隔后我们的优化目标变成了：

$$\min_w \frac{1}{2}\| w\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad g_i(w,b) = 1 - y_i\left(w^T x_i + b\right) - \xi_i \le 0, \quad \xi_i \ge 0, \quad i = 1,2,\ldots,n$$

其中 C 是一个大于 0 的常数，可以理解为错误样本的惩罚程度，若 C 为无穷大，ξ必然无穷小，如此一来线性 SVM 就又变成了线性可分 SVM；当 C 为有限值的时候，才会允许部分样本不遵循约束条件。

分类函数仍同上所示。

核函数

上述讨论的硬间隔和软间隔都是在说样本的完全线性可分或者大部分样本点的线性可分，但我们还是可能会碰到的一种情况是样本点不是线性可分的，这种情况的解决方法就是：将二维线性不可分样本映射到高维空间中，让样本点在高维空间线性可分。

我们用 x表示原来的样本点, 用$\phi(x)$表示 x映射到特征新的特征空间后到新向量。那么分割超 平面可以表示为: $f(x)=w\phi(x)+b$ 。
对于非线性 SVM的对偶问题就变成了:

$$\min_{\lambda}\left[\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j\left(\phi(x_i)\cdot\phi(x_j)\right)-\sum_{j=1}^{n}\lambda_i\right]$$

$$\text{s.t.}\quad \sum_{i=1}^{n}\lambda_i y_i=0,\quad \lambda_i\geq 0$$

可以看到与线性 SVM 唯一的不同就是: 之前的$(x_i\cdot x_j)$变成了$\left(\phi(x_i)\cdot\phi(x_j)\right)$

故而对应的分类函数就变为了：

$$f(x)=\text{sign}\left(w^T\left(\phi\left(x_i\right)\cdot\phi\left(x_j\right)\right)+b\right)$$

这里我们采用这样的核函数 $k(x,y)=(\phi(x),\phi(y)),\quad x_i$与 x_j 在特征空间的内积等于它们在原始样本空间中通过函数k(x, y)计算的结果，我们就不需要计算高维甚至无穷维空间的内积了。

这样，上面式子中的内积都可以用核函数代替，即优化对偶问题变为了：

$$\min_{\lambda}\left[\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j K(X_i, X)-\sum_{j=1}^{n}\lambda_i\right]$$

$$\text{s.t.}\quad \sum_{i=1}^{n}\lambda_i y_i = 0,\quad \lambda_i \geq 0$$

对应的分类函数就变为了：

$$f(x)=\text{sign}\left(w^T K(X_i, X)+b\right)$$

下面是几种常见的核函数：

1.线性核函数 $k\left(x_i, x_j\right)=x_i^T x_j$

2.多项式核函数 $k\left(x_i, x_j\right)=\left(x_i^T x_j\right)^d$

3.高斯核函数 $k\left(x_i, x_j\right)=\exp\left(-\frac{\left\|x_i-x_j\right\|}{2\delta^2}\right)$

例题：我们使用威斯康星州乳腺癌数据集，该数据集可以从sklearn中导入。

In[]:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
print("cancer.keys(): \n{}".format(cancer.keys()))
```

Out[]:

```
cancer.keys():
dict_keys(['feature_names', 'data', 'DESCR', 'target',
'target_names'])
```

In[83]:

```
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(
 cancer.data, cancer.target, random_state=0)
svc = SVC(kernel='rbf', C=10, gamma=0.1)
svc.fit(X_train, y_train)
print("Accuracy on training set: {:.2f}".format(svc.score(X_train, y_train)))
print("Accuracy on test set: {:.2f}".format(svc.score(X_test, y_test)))
```

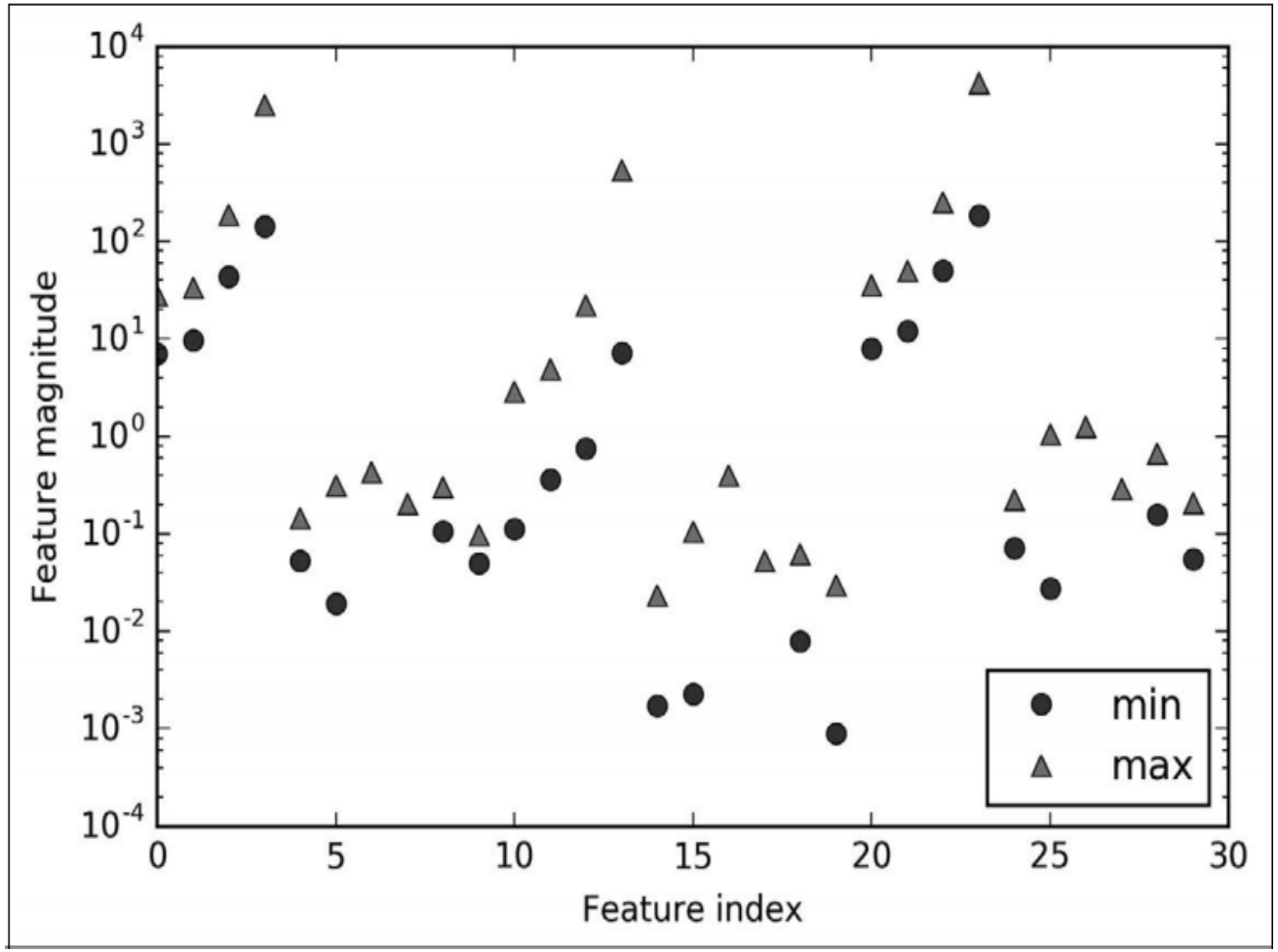
Out[83]:

```
Accuracy on training set: 1.00
Accuracy on test set: 0.63
```

可以看到此时处于严重的过拟合。我们再来看看每个特征的最大最小值

In[84]:

```
plt.plot(X_train.min(axis=0), 'o', label="min")
plt.plot(X_train.max(axis=0), '^', label="max")
plt.legend(loc=4)
plt.xlabel("Feature index")
plt.ylabel("Feature magnitude")
plt.yscale("log")
```


Feature magnitude
Feature index
min
max
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5
10
15
20
25
30

从这张图中，我们可以确定乳腺癌数据集的特征具有完全不同的数量级。这对其他模型来说（比如线性模型）可能是小问题，但对核 SVM 却有极大影响。因此我们需要对数据进行预处理，步骤如下

In[85]:

```
# 计算训练集中每个特征的最小值
min_on_training = X_train.min(axis=0)
# 计算训练集中每个特征的范围（最大值-最小值）
range_on_training = (X_train -min_on_training).max(axis=0)
# 减去最小值，然后除以范围
# 这样每个特征都是min=0和max=1
X_train_scaled = (X_train - min_on_training) /range_on_training
```

In[86]:

```
# 利用训练集的最小值和范围对测试集做相同的变换
X_test_scaled = (X_test - min_on_training) / range_on_training
```

In[87]:

```
svc = SVC(kernel='rbf', C=10, gamma=0.1)
svc.fit(X_train_scaled, y_train)
print("Accuracy on training set: {:.3f}".format(
 svc.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

Out[87]:

Accuracy on training set: 0.984Accuracy on test set: 0.965

可以看到在测试集上的精度用明显的提高