SVM算法

主要内容

概述

1 概述

- 支持向量机(Support Vector Machine，SVM)的基本模型是定义在特征空间上间隔最大的线性分类器。它是一种二类分类模型，当采用了核技巧之后，支持向量机可以用于非线性分类。不同类型的支持向量机解决不同的问题。
  - 线性可分支持向量机（也称为硬间隔支持向量机）：当训练数据线性可分时，通过硬间隔最大化，学得一个线性可分支持向量机。
  - 线性支持向量机（也称为软间隔支持向量机）：当训练数据近似线性可分时，通过软间隔最大化，学得一个线性支持向量机。
  - 非线性支持向量机：当训练数据不可分时，通过使用核技术以及软间隔最大化，学得一个非线性支持向量机。

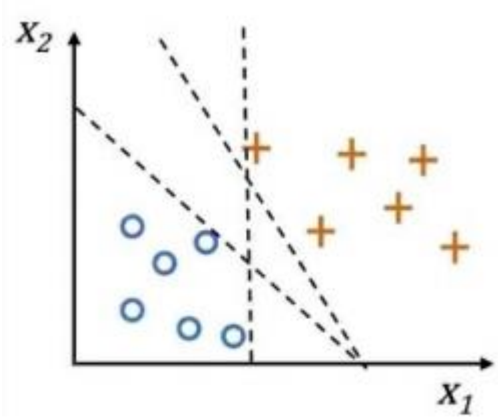
1 概述

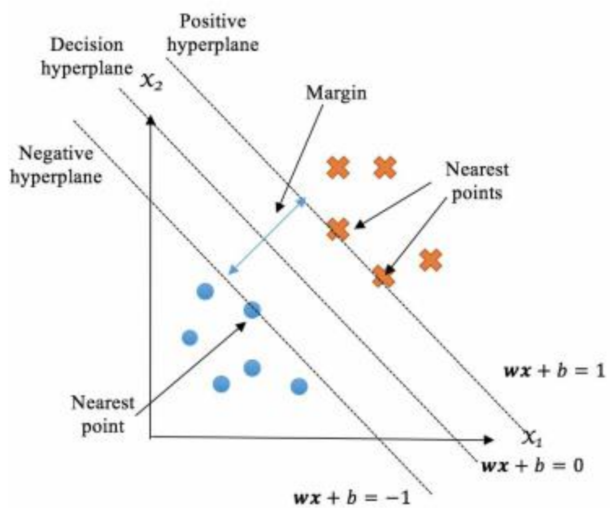
- 在本章中，假设输入空间和特征空间是不同的。通常假设输入空间为欧氏空间，特征空间为希尔伯特空间。此时给定某个输入$\vec{x}$，通过某种映射（可能是线性映射，也可能是非线性映射）到特征空间的表示为$\vec{z}$。此时在特征空间中学习线性支持向量机（而不是在输入空间中学习线性支持向量机）。欧氏空间与希尔伯特空间的不同如下：
  - 欧氏空间是有限维度的，希尔伯特空间是无穷维度的；
  - 欧氏空间 $\subseteq$ 希尔伯特空间 $\subseteq$ 内积空间 $\subseteq$ 赋范空间

SVM算法

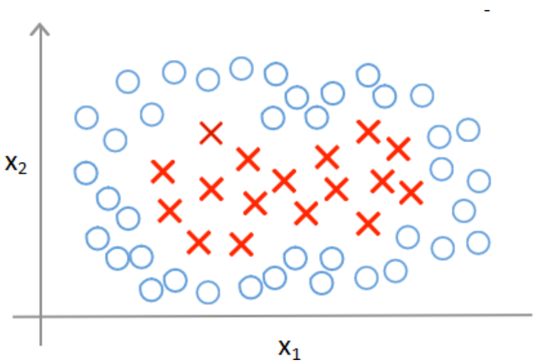
8 支持向量机 SVM

- 分类：寻找最佳分割

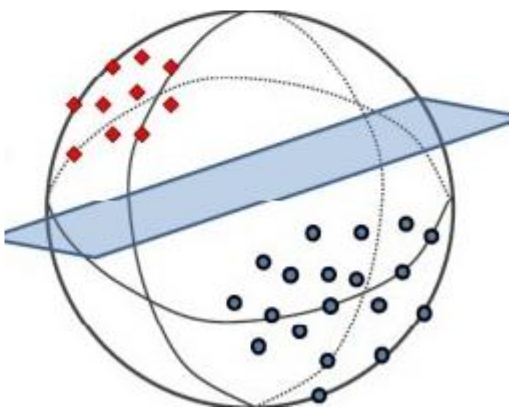




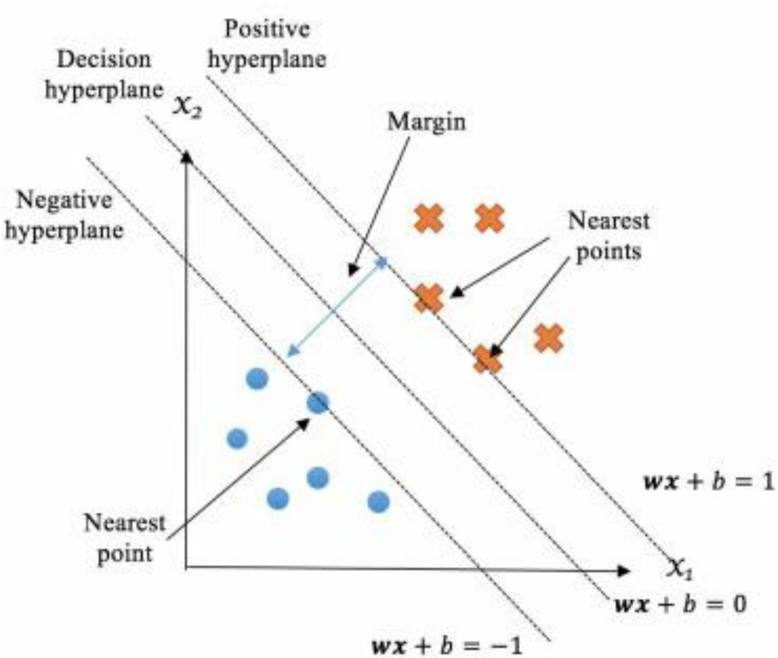
- 映射：使用高效计算



$$\exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$$



8.1 分类：寻找最佳分割



$$positive: w^T x_{pos} + b = 1$$

$$negative: w^T x_{neg} + b = -1$$

$$=> w^T (x_{pos} - x_{neg}) = 2$$

$$\| w \| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

$$=> \frac{w^T (x_{pos} - x_{neg})}{\| w \|} = \frac{2}{\| w \|}$$

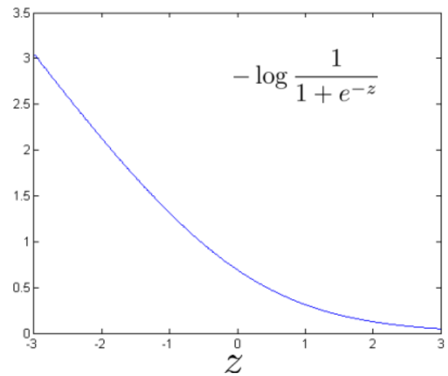
Hewlett Packard Enterprise

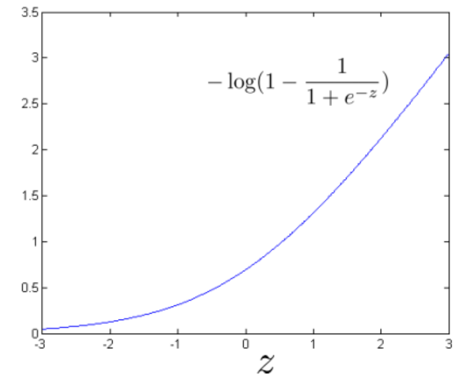
8.1 分类：寻找最佳分割

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

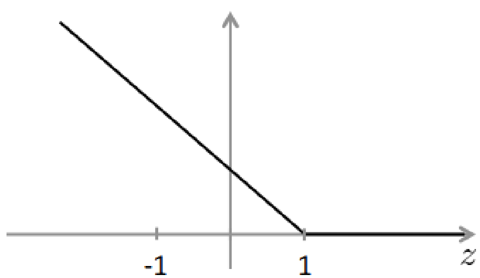
If $y=1$ (want $\theta^T x \gg 0$):

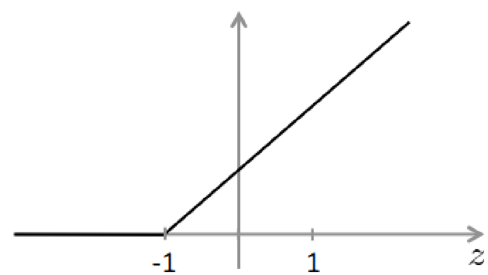
If $y=0$ (want $\theta^T x \ll 0$):





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





If $y=1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

If $y=0$, we want $\theta^T x \leq -1$ (not just < 0)

Hewlett Packard Enterprise

8.2 映射：使用高效计算

普通映射

$$x=[x_1,x_2],y=[y_1,y_2]$$

$$\Phi(x)=[x_1x_1,x_1x_2,x_2x_1,x_2x_2]$$

$$\Phi(y)=[y_1y_1,y_1y_2,y_2y_1,y_2y_2]$$

$$\Phi(x)*\Phi(y).T=x_1x_1y_1y_1+x_1x_2y_1y_2+x_1x_2y_1y_2+x_2x_2y_2y_2$$

核函数

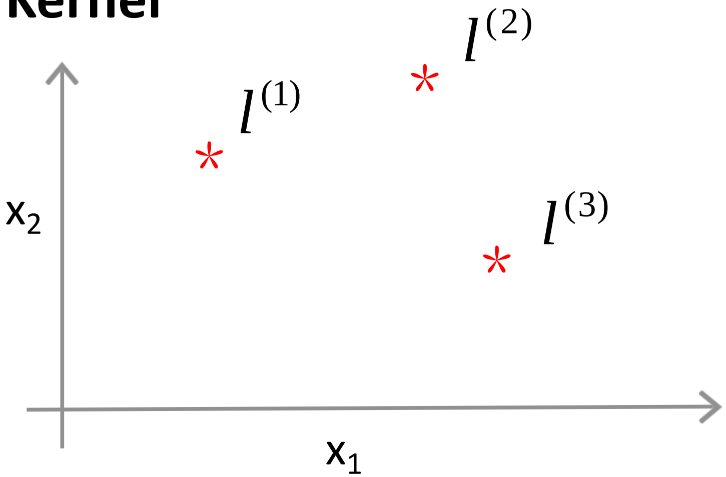
$$K(<x,y>)=(x*y.T)**2=(x_1y_1+x_2y_2)**2$$

$$=x_1x_1y_1y_1+x_1x_2y_1y_2+x_1x_2y_1y_2+x_2x_2y_2y_2$$

常用核函数：线性核函数，多项式核函数，高斯核函数，sigmoid核函数

8.2.1 高斯核函数解析

Kernel



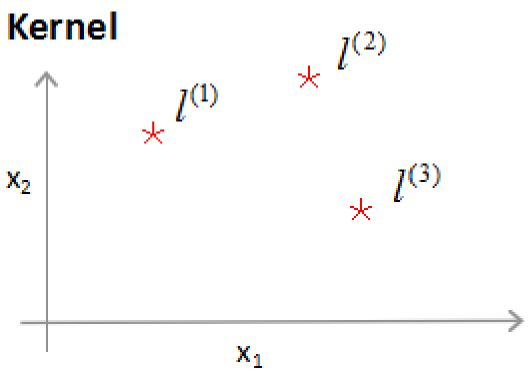
$l^{(1)}, l^{(2)}, l^{(3)}$ landmark基准点

x 样本点

对于给定样本x，带入高斯核函数实现高维“映射”：

$$f^{(i)} = \exp\left(-\frac{\| x - l^{(i)} \|^2}{2\sigma^2}\right)$$

8.2.1 高斯核函数解析



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

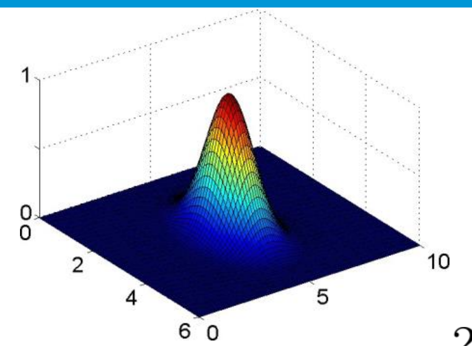
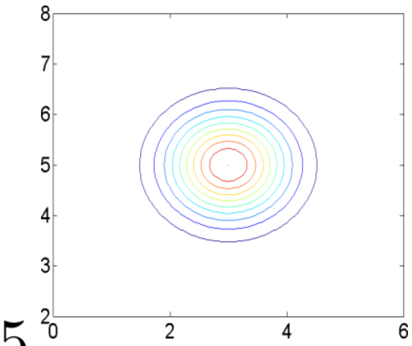
$$f^{(2)} = \exp\left(-\frac{\| x - l^{(2)} \|^2}{2\sigma^2}\right)$$

$$f^{(3)} = \exp\left(-\frac{\| x - l^{(3)} \|^2}{2\sigma^2}\right)$$

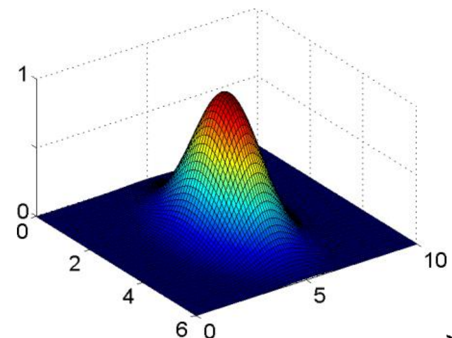
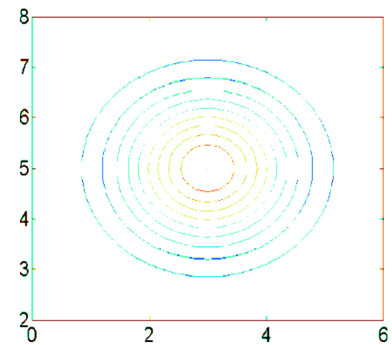
样本x高斯"映射"

$$if\ (x \approx l)\ f \approx 1$$

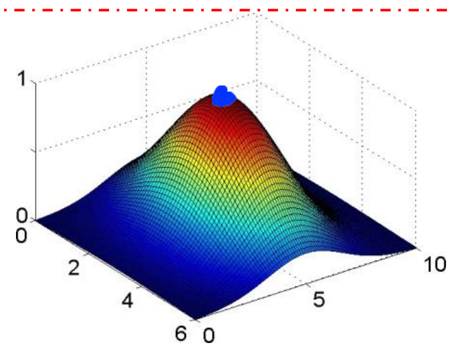
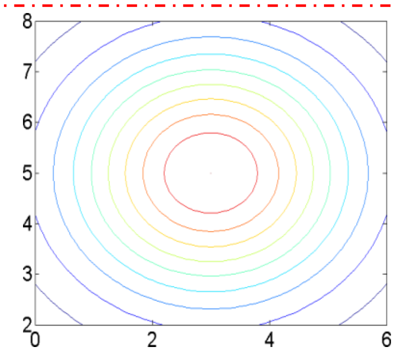
$$if\ (\| x - l \| >> 0)\ f \approx 0$$

$\sigma^2 = 0.5$

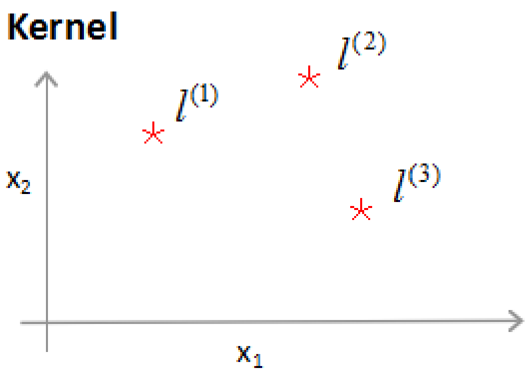



$\sigma^2 = 1$

$\sigma^2 = 3$

8.2.1 高斯核函数解析



f: 高斯映射后的特征值;

θ: 需训练的权重值;

predict "1" when

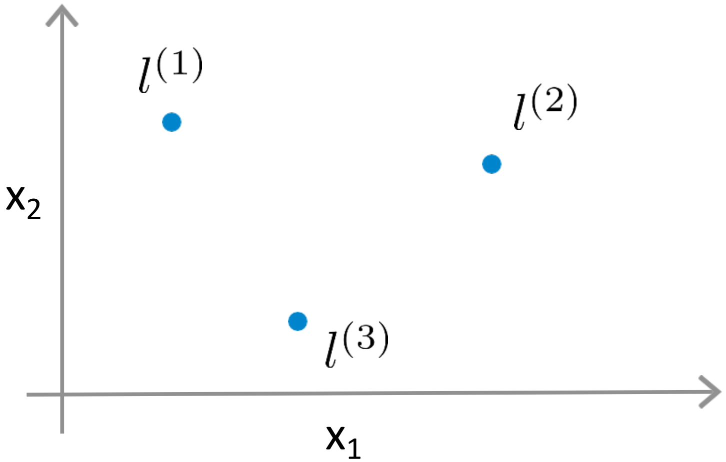
$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

predict "-1" when

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 < 0$$

8.2.2 高斯核函数应用

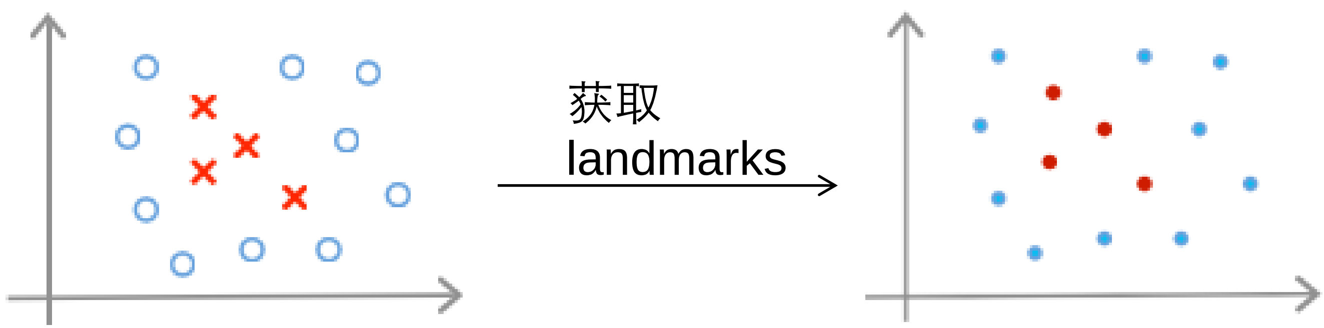
选择landmarks



$$f = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 --> y*$$

$$l^{(1)}, l^{(2)}, l^{(3)}, \ldots \quad ?$$



8.2.2 高斯核函数应用

SVM with Kernels

给定样本点： $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}),$

$l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}.$

x

设定Landmarks ：

$$f^{(1)} = \exp(-\frac{\| x - l^{(1)} \|^2}{2\sigma^2})$$

$\ldots$

$$f^{(2)} = \exp(-\frac{\| x - l^{(2)} \|^2}{2\sigma^2})$$

...

对于训练样本 x ：

$$f^{(1)} = \exp(-\frac{\| x - l^{(1)} \|^2}{2\sigma^2})$$

$$f^{(2)} = \exp(-\frac{\| x - l^{(2)} \|^2}{2\sigma^2})$$

...

$$f^{i} = 1(x = l^{(i)})$$

...

Hewlett Packard Enterprise

8.2.2 高斯核函数应用

SVM with Kernels

1. Train

- 给定样本集X
- 设定landmarks基准点 $l^{(i)}$
- 对样本集X的每个元素 $x^{(i)}$ 进行高斯"映射"，转换为高维特征 f
- 基于高纬特征，进行训练，**Loss**函数：

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

- 获取向量 θ 值

2. Predict：

- 给定待测样本x
- 进行高斯"映射"，获取高维特征 f
- 通过 $\theta.T \times f$ 运算，预测样本x的y*值

Hewlett Packard Enterprise

企业级实训 EPM

Python实现

3.2 sklearn实现线性分类SVM

Sklearn中LinearSVC实现了线性分类支持向量机，可以用于二类分类，也可以用于多分类。其原型为：

sklearn.svm.LinearSVC()

常见属性：

- coef_：一个数组，它给出了各个特征的权重。
- intercept_：一个数组，它给出了截距，即决策函数中的常数项。

常用方法：

- fit(X,y)：训练模型。
- predict(X)：用模型进行预测，返回预测值。
- score(X,y[,sample_weight])：返回在(X,y)上预测的准确率。

3.2 sklearn实现线性分类SVM

使用鸢尾花数据集来进行线性分类SVM测试

```
from sklearn import datasets, svm
from sklearn.model_selection import train_test_split


# 加载数据
def load_data():
    iris = datasets.load_iris()
    X_train = iris.data
    y_train = iris.target
    return train_test_split(X_train, y_train, test_size=0.25,
random_state=0, stratify=y_train)
```

3.2 sklearn实现线性分类SVM

```
# 测试线性分类支持向量机
def test_LinearSVC(*data):
   X_train, X_test, y_train, y_test = data
   cls = svm.LinearSVC()
   cls.fit(X_train, y_train)
   print('Coefficients:%s, intercept %s' %
(cls.coef_,cls.intercept_) )
   print('Score: %.2f' % cls.score(X_test, y_test))


if __name__ == "__main__":
   X_train, X_test, y_train, y_test = load_data()
   test_LinearSVC(X_train, X_test, y_train, y_test)
```

Hewlett Packard Enterprise

3.2 sklearn实现线性分类SVM

输出如下：

```
Coefficients:[[ 0.20959341  0.39923783 -0.81739151 -0.44231876]
 [-0.12885621 -0.7881668   0.52143847 -1.02315951]
 [-0.80323579 -0.87601498  1.21363075  1.81025233]], intercept [ 0.11973741  2.04457938 -1.44397868]
Score: 0.97
```

可以看到线性分类支持向量机的预测性能相当好，对于测试集的预测准确率高达97%

3.3 sklearn实现非线性分类SVM

Sklearn中SVC实现了非线性分类支持向量机，可以用于二类分类，也可以用于多分类。其原型为：

sklearn.svm.SVC()

常用参数：

- kernel：指定核函数。
  - 'linear'：线性核。
  - 'poly'：多项式核。
  - 'rbf'：高斯核函数，默认值。
  - 'sigmod'：S型核函数。

常见属性：

- coef_：一个数组，它给出了各个特征的权重。只对线性核有效。
- dual_coef：在分类决策函数中每个支持向量的系数。
- intercept_：一个数组，它给出了截距，即决策函数中的常数项。
- support_vectors_：一个数组，支持向量。

常见方法：

同线性分类SVM

3.3 sklearn实现非线性分类SVM

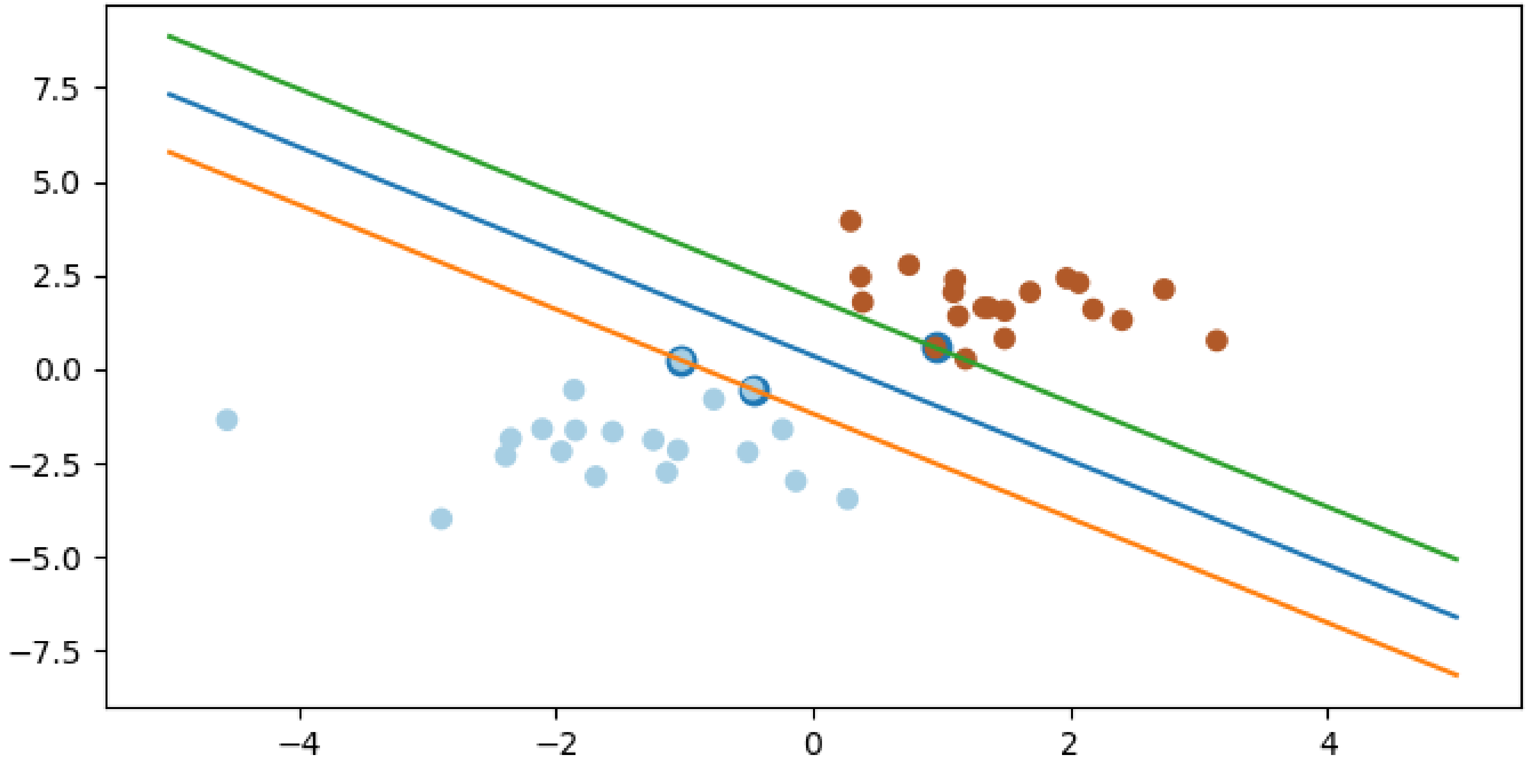
```
np.random.seed(0)
# 正态分布来产生数字,20行2列*2
x = np.r_[np.random.randn(20, 2) - [2, 2], np.random.randn(20, 2) + [2, 2]]
# 20个class0, 20个class1
y = [0] * 20 + [1] * 20
# 调用非线性支持向量机
clf = svm.SVC(kernel='linear')
clf.fit(x, y)

w = clf.coef_[0]  # 获取w
a = -w[0] / w[1]  # 斜率

print("W:", w)
print("a:", a)
print("support_vectors_:", clf.support_vectors_)
print("clf.coef_:", clf.coef_)
```

3.3 sklearn实现非线性分类SVM

```
# 画图划线
xx = np.linspace(-5, 5)  # (-5,5)之间x的值
yy = a * xx - (clf.intercept_[0]) / w[1]  # xx带入y, 截距
# 画出与点相切的线
b = clf.support_vectors_[0]
yy_down = a * xx + (b[1] - a * b[0])
b = clf.support_vectors_[-1]
yy_up = a * xx + (b[1] - a * b[0])
plt.figure(figsize=(8, 4))
plt.plot(xx, yy)
plt.plot(xx, yy_down)
plt.plot(xx, yy_up)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1], s=80)
plt.scatter(x[:, 0], x[:, 1], c=y, cmap=plt.cm.Paired)  # [:, 0]列切片, 第0列
plt.axis('tight')
plt.show()
```

3.3 sklearn实现非线性分类SVM

结果如下图：



小结

4.1 小结

企业级实训 EPM

SVM的一般流程

① 收集数据：可以使用任何方法
② 准备数据：需要数值型数据。
③ 分析数据：有助于可视化分隔超平面。
④ 训练算法：SVM的大部分时间都源自训练，该过程主要实现两个参数的调优。
⑤ 测试算法：十分简单的计算过程就可以实现。
⑥ 使用算法：几乎所有分类问题都可以使用SVM，值得一提的是，SVM本身是一个二类分类器，对多类问题应用SVM需要做一些修改。

4.1 小结

SVM的优缺点

- SVM本质上是非线性方法，在样本量比较少的时候，容易抓住数据和特征之间的非线性关系，因此可以解决非线性问题、可以避免神经网络结构选择和局部极小点问题、可以提高泛化性能、可以解决高维问题。
- SVM对缺失数据敏感，对非线性问题没有通用解决方案，必须谨慎选择核函数来处理，计算复杂度高。主流的算法是$O(n^2)$，这样对大规模数据就显得很无力了。不仅如此，由于其存在两个对结果影响相当大的超参数，这两个超参数无法通过概率方法进行计算，只能通过穷举试验来求出，计算时间高于不少类似的非线性分类器。

The end

企业级实训 EPM

Hewlett Packard Enterprise