支持向量机(SVM)

作者：刘伟杰 日期：2015-12-02
参考：
[1]《统计学习方法》 李航 2012年3月第一版
[2]《机器学习实战》 Peter Harrington

1. 理论

1. 概述：

   利用训练集在特征空间中求出一个分类超平面(w,b)把样本切割开，依靠该超平面对新样本进行分类。如果训练集在当前的特征空间中无法分割，则用核技术的映射函数把原特征空间映射到高纬或者无穷维空间再切割。

2. 基本概念：

   超平面：

   > 用(w,b)表示， `w1*x1 + w2*x2 +...+ wn*xn = 0` 表示在特征空间中的一个平面。需要注意的是，

   函数距离（间隔）：

   > 点到超平面的函数距离： `y_i(w * x_i)`
   > 集合到超平面的函数距离： 集合中的点到超平面的函数距离的最小值

   几何距离（间隔）：

   > 点到超平面的几何距离： 函数距离/||w||
   > 集合到超平面的几何距离： 集合中的点到超平面的几何距离的最小值

3. 线性可分SVM

   线性SVM就是在训练集中寻找一个几何间隔最大的超平面(w,b)作为分类平面，转化为如下最优化问题：

   > $$max: \frac{\gamma'}{||w||}$$
   > $$s.t.: y_i(wx_i + b) > \gamma'$$

其中gamma‘为训练集到超平面的函数距离。但是，该问题求出的一个超平面可以有很多种(w, b)的表示形式。换句话说，存在无数个(w,b)满足最优，但是这些(w, b)都表示一个超平面。因此需要加上如下约束：

$$\gamma' = 1$$

该优化问题等价为如下形式：

$$min: \frac{1}{2}||w||^2$$
$$s.t.: y_i(wx_i + b) - 1 > 0$$

但是训练集合一般都存在一定的噪声，我们允许超平面将部分的点分类错误。加入松弛变量e和惩罚因子C, 就得到了最终形式：

$$min: \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} e_i$$
$$s.t.: y_i(wx_i + b) - 1 - e_i > 0$$

那个min优化目标函数就是SVM的风险函数，求解过程就是训练过程。之后用最优化算法求解该问题即可，一般使用拉格朗日乘子法，实现的时候SMO算法求解。

4. 非线性SVM于核函数

绝大多数情况下训练集无法线性可分，这个时候需要用某种映射把当前的特征空间映射到高维的特征空间中，在新特征空间中就线性可分。

核函数定义：

> 设X是原特征空间，H是新的高维或无限维的特征空间，若存在一个从X到H的映射
>
> f(x): X->H
>
> 使得对所有x,z属于X，函数K(x,z)满足：
>
> K(x,z) = f(x)f(z) （ 这里是内积 ）
>
> 则称K(x,z) 为映射f的核函数。
>
> 注意：核函数不是映射，不同的映射可以有相同的核函数。对于给定的核函数，其映射方式不唯一，我

映射与核函数举例：

$$X = R^2 : x = (x^{(1)}), x^{(2)})$$

$$H = R^3 : f_1(x) = ((x^{(1)})^2, \sqrt{2}x^{(1)}x^{(2)}, (x^{(2)})^2)$$

$$H = R^4 : f_2(x) = ((x^{(1)})^2, x^{(1)}x^{(2)}, x^{(1)}x^{(2)}, (x^{(2)})^2)$$

$$K(x, z) = (x \cdot z)^2$$

f1与f2这两个从二维到不同维度的映射，有相同的核函数.

常用的核函数:

> 定义在欧式空间中：多项式核函数、高斯核函数、
>
> 欧式空间核离散数据都可以用：字符串核函数（常用于文本信息分类）

2. 实现:

1. 我的实现

   训练过程为SMO算法:

   ```
   https://github.com/autoliuweijie/MachineLearning/tree/master/SVM
   ```

2. scikit-learn:

   简单例子:

   ```
   >>> from sklearn import svm
   >>> X = [[0, 0], [1, 1]]
   >>> y = [0, 1]
   >>> clf = svm.SVC()
   >>> clf.fit(X, y)
   SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
       decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
       max_iter=-1, probability=False, random_state=None, shrinking=True,
       tol=0.001, verbose=False)
   >>> clf.predict([[2., 2.]])
   array([1])
   ```

   在创建svm.SVC对象时常用的可选参数如下:

```
C：为惩罚系数，默认为1.0
kernel：为核函数，可选值为‘linear’， ‘polynomial’，‘rbf’，‘sigmoid’。也可以自定义核函数
class_weight：当出现unbalance问题时，这个参数可以用来设置类的权重。
decision_function_shape='ovo'：当进行多分类的时候，选择多分类的策略，'ovo'表示'one aga
更多参数请看官方指南：http://scikit-learn.org/stable/modules/svm.html#svm
```

如果想查看支持向量是哪些：

```
>>> # get support vectors
>>> clf.support_vectors_
array([[ 0.,  0.],
       [ 1.,  1.]])
>>> # get indices of support vectors
>>> clf.support_
array([0, 1]...)
>>> # get number of support vectors for each class
>>> clf.n_support_
array([1, 1]...)
```

SVM还可以用与回归：

```
>>> from sklearn import svm
>>> X = [[0, 0], [2, 2]]
>>> y = [0.5, 2.5]
>>> clf = svm.SVR()
>>> clf.fit(X, y)
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
>>> clf.predict([[1, 1]])
array([ 1.5])
```