AI学习笔记--sklearn--SVR算法

- ## 概述

支持向量回归（Support Vector Regression, SVR）。和SVM算法类似的一种方法。具体原理性就去查阅相关的文档。

- ## 算法解析

线性回归问题，需要给定训练所需要的样本D={(x_1,y_1),(x_2,y_2),...,(x_n,y_n)} 。我们需要去识别一个函数f(x)尽可能的满足实际的样本走势。在这个模型中，只有当f(x) 与y完全相同的时候，算法损失才会趋近于0，而支持向量回归假设我们能容忍的f(x)与y之间最多有ε的偏差，当且仅当f(x)与y的差别绝对值大于ε时，才计算损失，此时相当于以f(x)为中心，构建一个宽度为2ε的间隔带，若训练样本落入此间隔带，则认为是被预测正确的。（间隔带两侧的松弛程度可有所不同）。

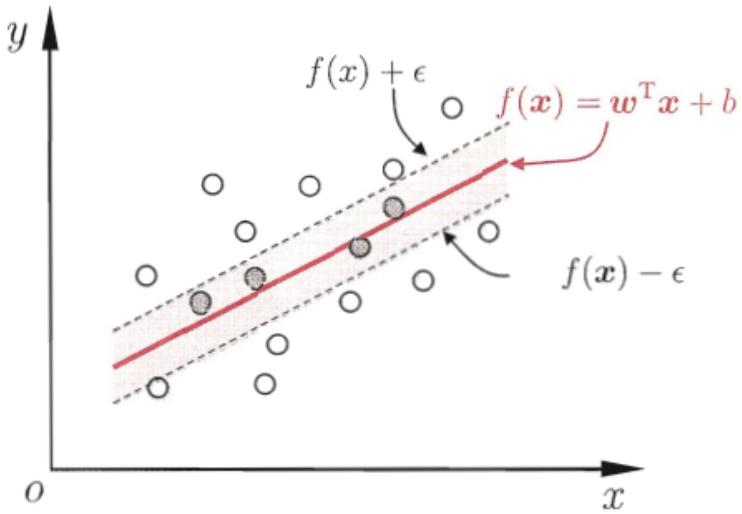


图 支持向量回归示意图. 红色显示出 ε-间隔带, 落入其中的样本不计算损

因此SVR问题可转化为（下式左部是正则化项）：

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} \ell_\epsilon(f(\boldsymbol{x}_i) - y_i)$$

l为损失函数

$$\ell_\epsilon(z) = \begin{cases} 0, & \text{if } |z| \leqslant \epsilon \ ; \\ |z| - \epsilon, & \text{otherwise.} \end{cases}$$

因此引入了松弛因子，重写第一个式子为：

$$\min_{\boldsymbol{w}, b, \xi_i, \hat{\xi}_i} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i)$$

$$\begin{aligned} \text{s.t.} \ \ & f(\boldsymbol{x}_i) - y_i \leqslant \epsilon + \xi_i \ , \\ & y_i - f(\boldsymbol{x}_i) \leqslant \epsilon + \hat{\xi}_i \ , \\ & \xi_i \geqslant 0, \ \hat{\xi}_i \geqslant 0 \ , \ i = 1, 2, \ldots, m. \end{aligned}$$

最后引入拉格朗日乘子，可得拉格朗日函数：

$$\begin{aligned} & L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}}) \\ & = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) - \sum_{i=1}^{m}\mu_i\xi_i - \sum_{i=1}^{m}\hat{\mu}_i\hat{\xi}_i \\ & + \sum_{i=1}^{m}\alpha_i\big(f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i\big) + \sum_{i=1}^{m}\hat{\alpha}_i\big(y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i \end{aligned}$$

对四个遍历求偏导，令偏导数为零，可得：

$$\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i \ , \\ 0 &= \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) \ , \\ C &= \alpha_i + \mu_i \ , \\ C &= \hat{\alpha}_i + \hat{\mu}_i \ . \end{aligned}$$

把上边的式子带入，即可求得SVR的对偶问题：

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \quad & \sum_{i=1}^{m} y_i(\hat{\alpha}_i - \alpha_i) - \epsilon(\hat{\alpha}_i + \alpha_i) \\
& - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \\
\text{s.t.} \quad & \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) = 0 \ , \\
& 0 \leqslant \alpha_i, \hat{\alpha}_i \leqslant C \ .
\end{aligned}
$$

上边的过程需要满足KKT条件，即

$$
\begin{cases}
\alpha_i(f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i) = 0 \ , \\
\hat{\alpha}_i(y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i) = 0 \ , \\
\alpha_i\hat{\alpha}_i = 0 \ , \ \xi_i\hat{\xi}_i = 0 \ , \\
(C - \alpha_i)\xi_i = 0 \ , \ (C - \hat{\alpha}_i)\hat{\xi}_i = 0 \ .
\end{cases}
$$

最后，可得SVR的解为

$$
f(\boldsymbol{x}) = \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + b \ .
$$

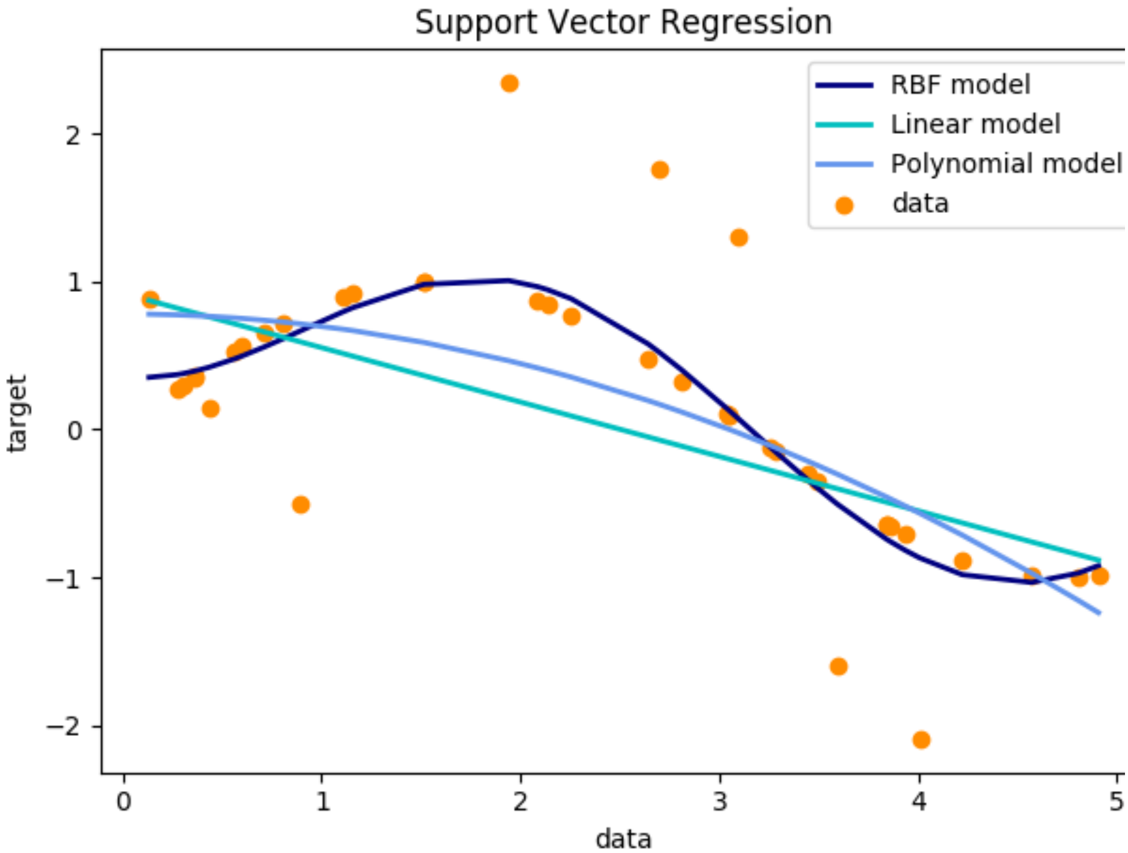
$$
b = y_i + \epsilon - \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} \ .
$$

- 算法demo

```python
# -*- coding: utf-8 -*
import numpy as np
from sklearn.svm import SVR
```

```
import matplotlib.pyplot as plt

###############################################################################
# Generate sample data
X = np.sort(5 * np.random.rand(40, 1), axis=0)  #产生40组数据，每组一个数据，axis=0决定按列排列，=1表示行排列
y = np.sin(X).ravel()   #np.sin()输出的是列，和X对应，ravel表示转换成行

###############################################################################
# Add noise to targets
y[::5] += 3 * (0.5 - np.random.rand(8))

###############################################################################
# Fit regression model
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)   #rbf的效果是最好的
svr_lin = SVR(kernel='linear', C=1e3)
svr_poly = SVR(kernel='poly', C=1e3, degree=2)
y_rbf = svr_rbf.fit(X, y).predict(X)
y_lin = svr_lin.fit(X, y).predict(X)
y_poly = svr_poly.fit(X, y).predict(X)

###############################################################################
# look at the results
lw = 2
plt.scatter(X, y, color='darkorange', label='data')
plt.hold('on')
plt.plot(X, y_rbf, color='navy', lw=lw, label='RBF model')
plt.plot(X, y_lin, color='c', lw=lw, label='Linear model')
plt.plot(X, y_poly, color='cornflowerblue', lw=lw, label='Polynomial model')
plt.xlabel('data')
plt.ylabel('target')
plt.title('Support Vector Regression')
plt.legend()
plt.show()
```

SKLearn中提供了不少模型供选择，比如RBF、linear、poly等算法，具体的运行结果可以从数据上和趋势上去看：



SVR可以作为函数预判，预判离散点未来的趋势函数算法。