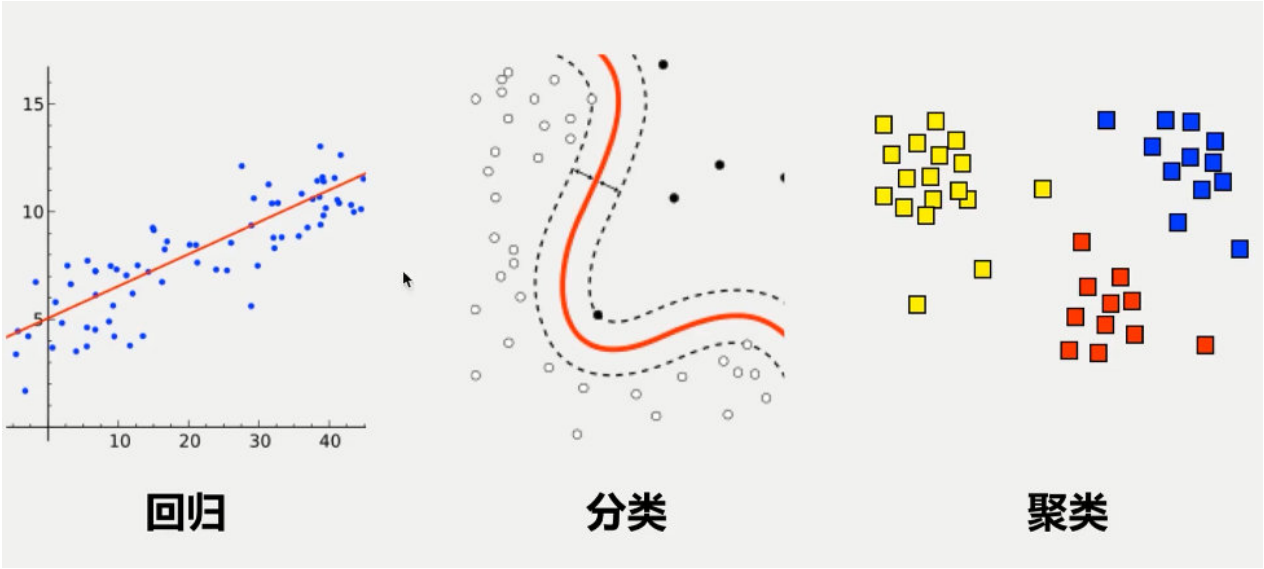
1. 什么是回归?

分类的目标变量是标称型数据，而回归是对连续型数据的预测。回归分析是一种预测建模技术，研究因变量和自变量之间的关系，如销售量预测或制造缺陷预测等，下图中的红线表示的就是回归曲线。

回归不同于分类和聚类，他们的区别可以用下图形象的表达出来。



2. 回归模型

这里使用`sklearn`进行代码实现，如果想手动实现的话，可以看《机器学习实战》，那本书有部分的算法实现，下面介绍的算法统一使用的函数如下

- 加载数据 数据文件点这里

```
def load_data(file_path):
    num_feat = len(open(file_path).readline().split("\t")) - 1
    data_mat = list()
    lable_mat = list()
    fr = open(file_path)
    for line in fr.readlines():
        line_arr = list()
        cur_line = line.strip().split("\t")
        for i in range(num_feat):
            line_arr.append(float(cur_line[i]))
        data_mat.append(line_arr)
        lable_mat.append(float(cur_line[-1]))
    return data_mat, lable_mat
```

- 绘制回归结果

```
def plot_regression(model, x_data, y_data):
    x_data = np.mat(x_data)
    y_data = np.mat(y_data).T
    x_train, y_train = x_data[:150,1:], y_data[:150,:]
    x_test, y_test = x_data[150:,1:], y_data[150:,:]
    model.fit(x_train, y_train)
    score = model.score(x_test, y_test)
    result = model.predict(x_train)
    plt.figure()
    srt_idx = x_train.argsort(0)
    plt.plot(x_train[srt_idx].reshape(-1,1),
y_train[srt_idx].reshape(-1,1), 'go', label = "true value")
    plt.plot(x_train[srt_idx].reshape(-1,1),
result[srt_idx].reshape(-1,1), 'ro-', label = "predict value")
    plt.title("score:%f" % score)
    plt.legend()
    plt.show()
```

2.1 线性回归

2.1.1 普通线性回归

提到回归，首先想到的肯定是线性回归**(linear regression)**，因为它是最容易理解，最简单的回归方法。设待拟合的数据对象为$X=\{x_1,x_2,\ldots,x_m\}$，其对应的真实值为$y=\{y_1,y_2,\ldots,y_m\}$，线性模型可以写为

$$\hat{y}=Xw \tag{1}$$

其中，w为回归系数，我们用平方误差来衡量拟合的误差

$$L(X)=\sum_{i=1}^{m}\left(y_i-x_i^T w\right)^2=(y-Xw)^2 \tag{2}$$

上式对w求导等于0可以得到

$$\frac{\partial L(X)}{\partial w}=\frac{\partial(y^Ty-w^TX^Ty-y^TXw-w^TX^TXw)}{\partial w}=2X^T(y-Xw)=0 \tag{3}$$

可以得到

$$\hat{w}=(X^TX)^{-1}X^Ty \tag{4}$$

上述方式容易对训练数据欠拟合，一种好的解决方式是**局部加权线性回归**，为每个误差增加一个权重w_i(这里的w并不是上面的$\hat{w}$)，此时误差函数可以写成

$$L(X)=\sum_{i=1}^{m}w_i\left(y_i-x_i^T w\right)^2=[W(y-Xw)]^2 \tag{5}$$

其中，W是一个对角矩阵，也叫做核，核的类型可以自由选择，最常见的就是高斯核，高斯核对应的权重如下

$$W(j,j)=\exp\left(\frac{\|x^i-x^j\|^2}{-2k^2}\right) \tag{6}$$

同样的，对新的误差函数$L(X)$求导可以得到此时回归系数为

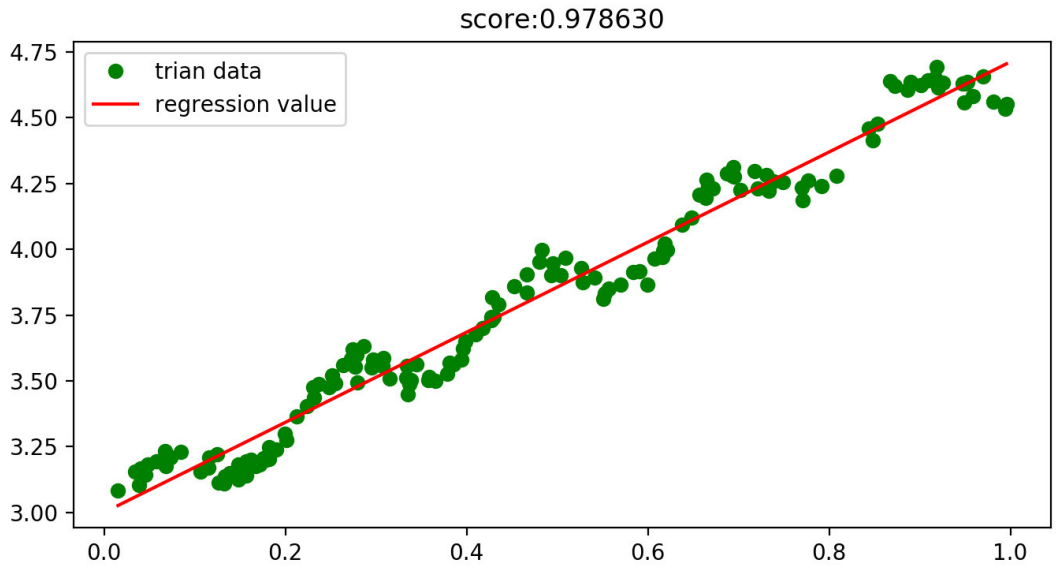
$$\hat{w}=(X^TWX)^{-1}X^TWy \tag{7}$$

这里的W其实是W^TW，但是使用W代替具有同样的意义并且简便。

`sklearn`调用代码：

```
x_data, y_data = load_data("ex0.txt")
from sklearn import linear_model
# 线性回归
model_linear_regression = linear_model.LinearRegression()
plot_regression(model_linear_regression, x_data, y_data)
```

绘制出的回归曲线如下图所示



2.1.2 岭回归

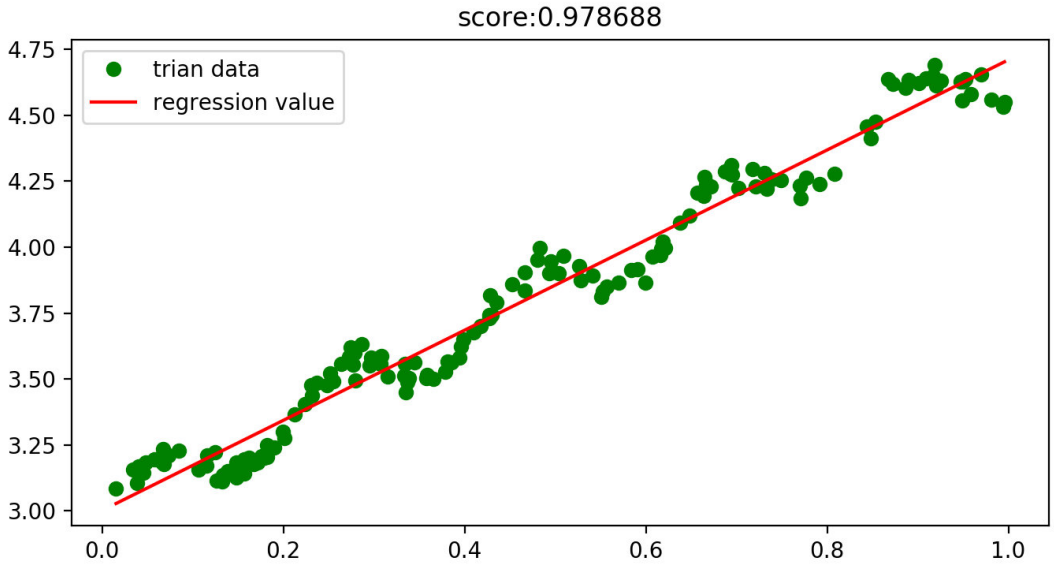
我们看线性回归中的输入集$X=\{x_1,x_2,\ldots,x_m\}$，假设其维度为n，当$n>m$的时候，X不是满秩矩阵，无法求解逆矩阵，这时候就需要用到岭回归**(ridge regression)**了，在矩阵X^TX上加上一个λI让其成为满秩矩阵，那么这个时候的回归系数为

$$\hat{w}=(X^TX+\lambda I)^{-1}X^Ty \tag{8}$$

`sklearn`调用代码：

```
x_data, y_data = load_data("ex0.txt")
from sklearn import linear_model
# Ridge回归
model_ridge = linear_model.Ridge(alpha = 0.01)
plot_regression(model_ridge, x_data, y_data)
```

绘制出的回归曲线如下图所示



2.2 决策树回归

决策树学习常用的算法有**ID3、C4.5、CART(classification and regression tree)**，这介绍用于回归的决策树**CART**，具体的方法理论参考李航的《统计学习方法》。

我们考虑输入的训练数据$D=\{X,y\}=\{(x_1,y_1),(x_1,y_1),\ldots,(x_m,y_m)\}$，一个回归树对应着输入空间（即特征空间）的一个划分以及在划分的单元上的输出值，假设已将输入空间划分为M个单元$R_1,R_2,\ldots,R_M$，并且在每一个单元上都有一个固定的输出值c_m，那么回归树模型可以表示为

$$f(x)=\sum_{m=1}^{M}c_mI(x\in R_m) \tag{9}$$

其中，函数I对应着0-1函数。当输入空间的划分确定时，可以用平方误差$\sum_{x_i\in R_m}(y_i-f(x_i))$来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值，那么单元R_m上的最优值$\hat{c_m}$是R_m上的所有输入实例x_i对应的输出y_i的均值，即

$$\hat{c_m}=ave(y_i|x_i\in R_m) \tag{10}$$

上面是整个树的输出形式，关键的问题来了，怎么对输入空间进行划分？这里采用启发式的算法，选择第j个变量和它取的值s作为切分变量(spliting variable)和切分点(spliting point)，并定义两个区域

$$R_1(j,s)=\{x|x_j\leq s\}\quad R_2(j,s)=\{x|x_j>s\} \tag{11}$$

然后寻找最优切分变量和最优切分点，即

$$m(s)=\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_j(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_j(j,s)}(y_i-c_2)^2\right] \tag{12}$$

简单的理解，就是在要求切分点s两边的区域的均方差都尽量小的同时，保证两个区域的最小均方差和是最小的。

对每一对(j,s)，均值表示为

$$\hat{c_1}=ave(y_i|x_i\in R_1(j,s))\quad \hat{c_2}=ave(y_i|x_i\in R_2(j,s)) \tag{13}$$

遍历所有输入变量，找到最优的对(j,s)，从而将输入空间切分为两个区域，接着对切分的两个区域重复上述划分过程，直到满足停止条件为止，这样一个回归树的生成就完成了。

举个  ，输入数据D如下表所示。

x_i	**1**	**2**	**3**	**4**	**5**	**6**
y_i	5.56	5.70	5.91	6.40	6.90	7.95

对上述连续型变量，只有一个切分变量，那么考虑切分点为1.5, 2.5, 3.5, 4.5, 5.5。对切分点依次求解$R_1,R_2,c_1,c_2,m(s)$，例如当切分点为2.5时，$R_1=\{1,2\},R_2=\{3,4,5,6\}$，其他的计算如下

$$c_1=\frac{1}{N_1}\sum_{x_i\in R_1(j,s)}y_i=\frac{1}{2}(5.56+5.70)=5.63 \tag{14}$$

$$c_2=\frac{1}{N_2}\sum_{x_i\in R_2(j,s)}y_i=\frac{1}{4}(5.91+6.40+6.90+7.95)=6.79 \tag{15}$$

$$s_m=\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]=2.294 \tag{16}$$

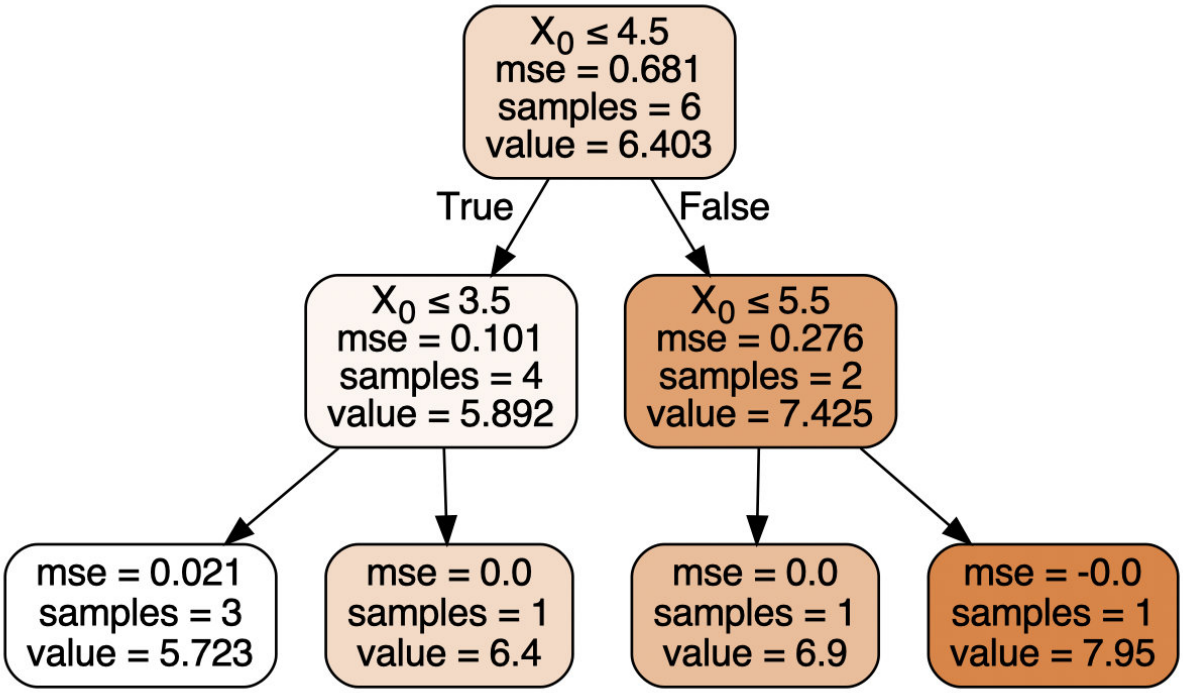
第一次切分时，对象为全体输入，计算出来的s_m值如下表所示。

切分点	**1.5**	**2.5**	**3.5**	**4.5**	**5.5**
$s(m)$	3.23468	2.294	1.31373333	0.956725	1.21752

可以看到，当$s=4.5$时，取得最小的$s(m)$值，此时的回归估计值为全体输入的均值6.403，递归求解左子树和右子树的回归估计值，最终求解的回归方程为

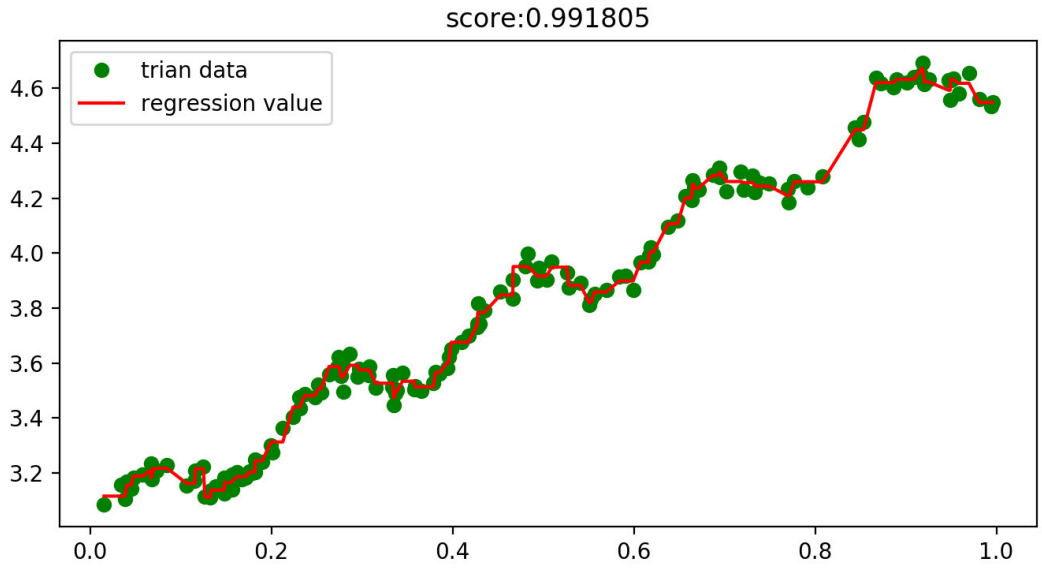
$$f(x)=\begin{cases}5.723 & x\leq 3.5\\ 6.4 & 3.5<x\leq 4.5\\ 6.9 & 4.5<x\leq 5.5\\ 7.95 & x>5.5\end{cases} \tag{17}$$

这个过程可以使用`graphviz`模块显示出来。



使用本文一开始提到的数据，决策树回归的代码如下

```
# 决策树回归
from sklearn import tree
model_decisiontree_regression = tree.DecisionTreeRegress(min_weight_fraction_leaf=0.01)
plot_regression(model_decisiontree_regression, x_data, y_data)
```



2.3 SVM回归

先回顾一下在基本线性可分情况下的SVM模型:

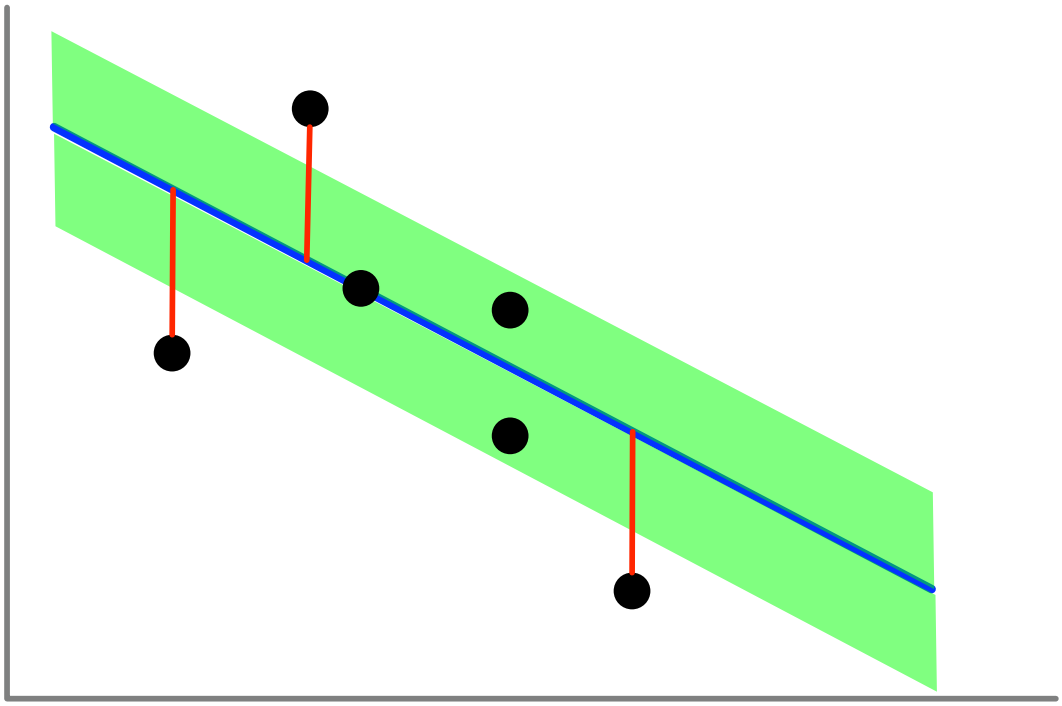
$$
\begin{aligned}
\min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\
\text{s.t.} \quad & -y_i\left(w\cdot x_i + b\right) - \xi_i + 1 \leq 0, \quad i = 1,2,\ldots,N \\
& -\xi_i \leq 0, \quad i = 1,2,\ldots,N
\end{aligned} \tag{18}
$$

分类SVM模型中要让训练集中的每个样本尽可能远离自己类别一侧的支持向量，回归模型也一样，沿用的是最大建哥分类器的思想。

对于回归模型，优化的目标函数和分类模型保持一致，依然是$\min_{w,b}\frac{1}{2}\|w\|^2$，但是约束条件不一样，回归模型的目标是让训练集中的每个样本点(x_i,y_i)尽量拟合到一个线性模型$y_i = wx_i + b$上，对于一般的回归模型使用均方误差MSE作为损失函数的，但是SVM回归不是这样定义的。

SVM需要我们定义一个常量$\varepsilon > 0$，对于某一个点(x_i,y_i)，如果$|y_i - wx_i - b| \leq \varepsilon$，则完全没有损失，如果$|y_i - wx_i - b| > \varepsilon$，则对应的损失为$|y_i - wx_i - b| - \varepsilon$，这个和均方差损失不同，对于均方差，只要$|y_i - wx_i - b| \neq 0$就会有损失。

如下图所示，在蓝色条带里面的点是没有损失的，但是在外面的点是有损失的，损失大小为红色线的长度。



总结下，我们的SVM回归模型的损失函数度量为

$$
\text{err}(x_i,y_i) = \begin{cases} 0 & |y_i - w\cdot x_i - b| \leq \varepsilon \\ |y_i - w\cdot x_i - b| - \varepsilon & |y_i - w\cdot x_i - b| > \varepsilon \end{cases} \tag{19}
$$

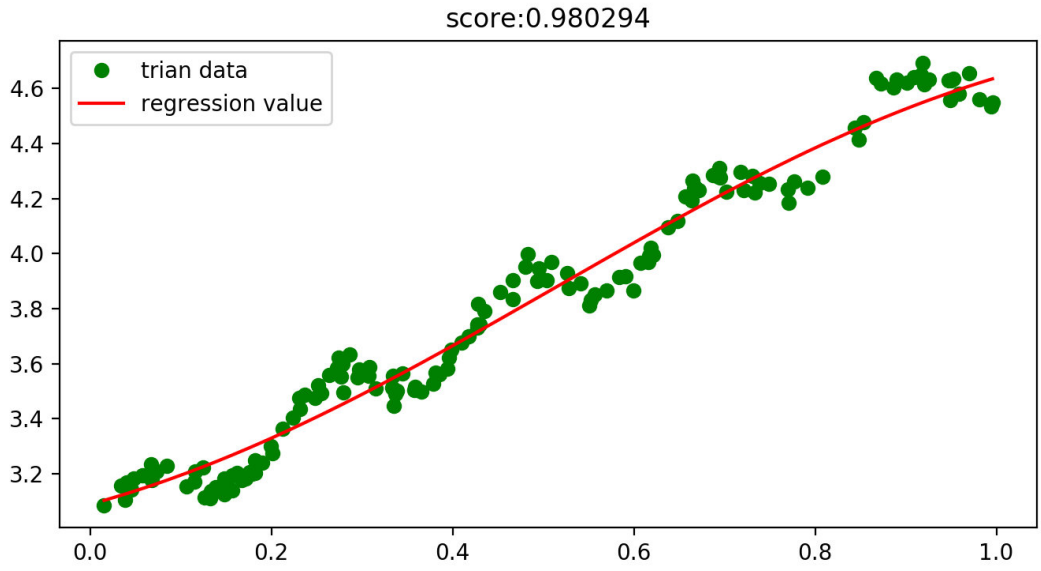
有了损失函数之后，我们就可以定义SVM回归的目标函数为

$$
\begin{aligned}
&\min \frac{1}{2}\|w\|_2^2 \\
&\text{s.t } |y_i - w\cdot x_i - b| \leq \varepsilon (i = 1,2,\ldots,m)
\end{aligned} \tag{20}
$$

这个模型的最优解求解过程这里不再赘述，有兴趣的可以看参考[2]或者[3]中的论述。

使用本文一开始提到的数据，SVM回归的代码如下

```
# SVM回归
from sklearn import svm
model_svr = svm.SVR()
plot_regression(model_svr, x_data, y_data)
```


score:0.980294
trian data
regression value
4.6
4.4
4.2
4.0
3.8
3.6
3.4
3.2
0.0
0.2
0.4
0.6
0.8
1.0

参考

[1] 李航. 统计学习方法, 清华大学出版社

[2] CSDN-SVM回归博客

[3] cnblog-SVM回归博客