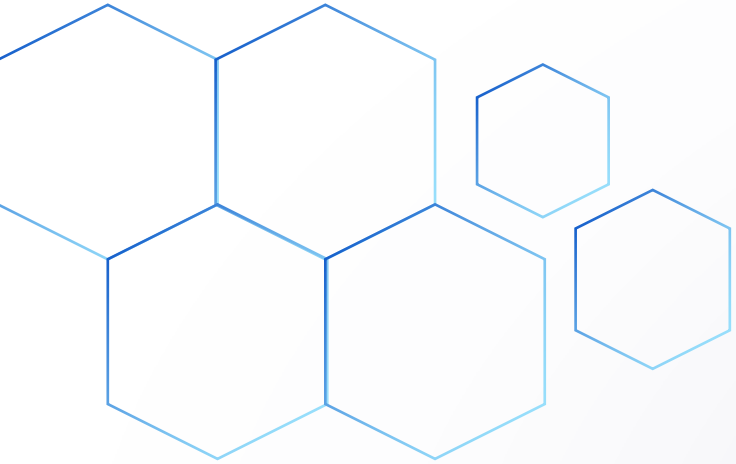


ETP Course

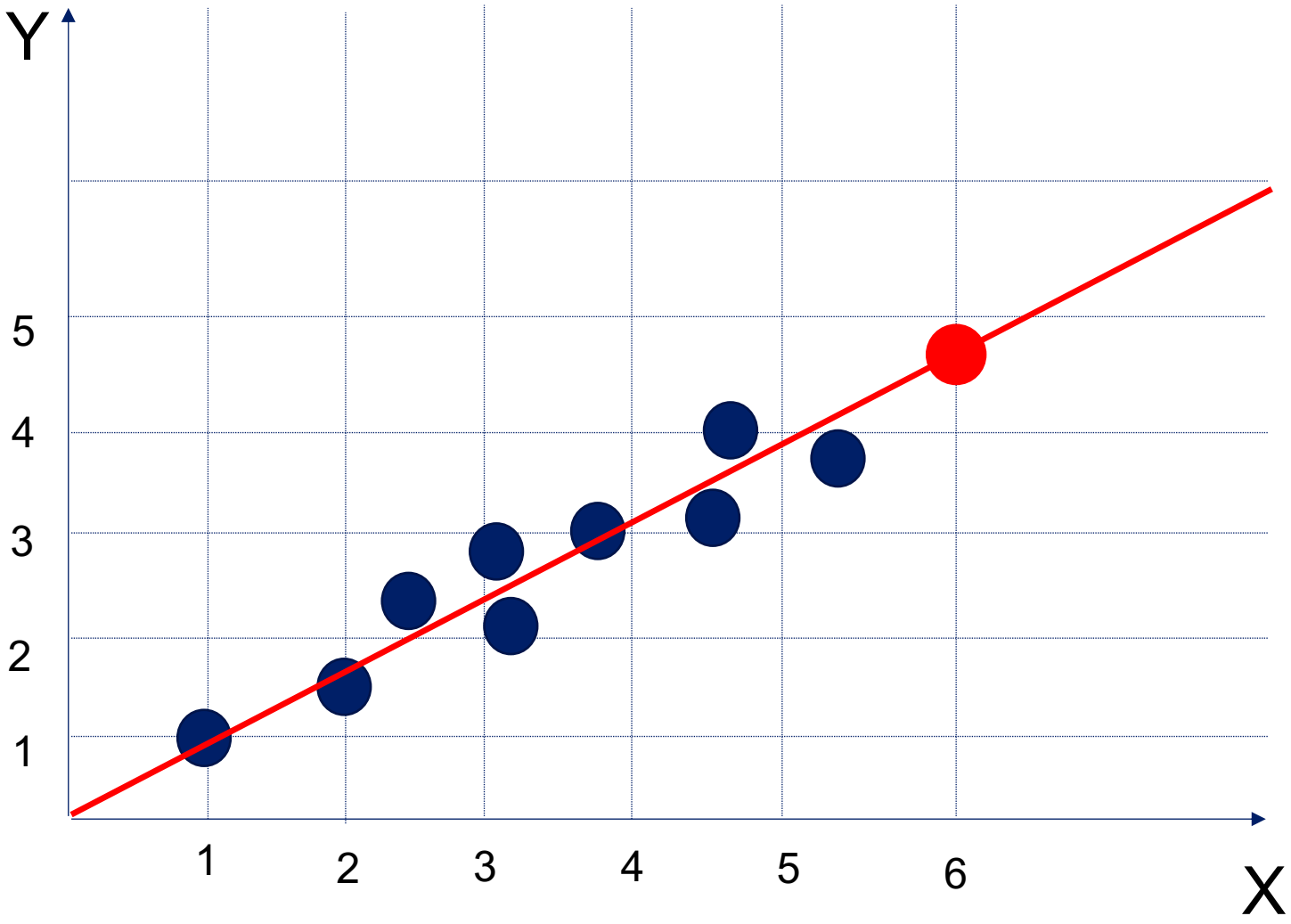
Practice of AI

线性回归

Jim Xie

2020/3/6

一元线性回归



当X=6 时， Y等于几比较合适？

Y = k * x + b

多元线性回归模型

一元线性回归：

x	Y
面积（平）	房价（元）
80	240
90	270
100	300
95	???

$$Y = \omega x + b$$

$$J = \frac{1}{2}\sum (y_i - (\omega x_i + b))^2$$

多元线性回归：

x_1	x_2	Y
面积（平）	房间数（个）	房价（元）
90	2	270
90	3	280
100	3	310
95	3	???

$$Y = \omega_1 x_1 + \omega_2 x_2 + b$$

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2$$

如果同时要求 ω_1与 ω_2和最小呢？

最小二乘法

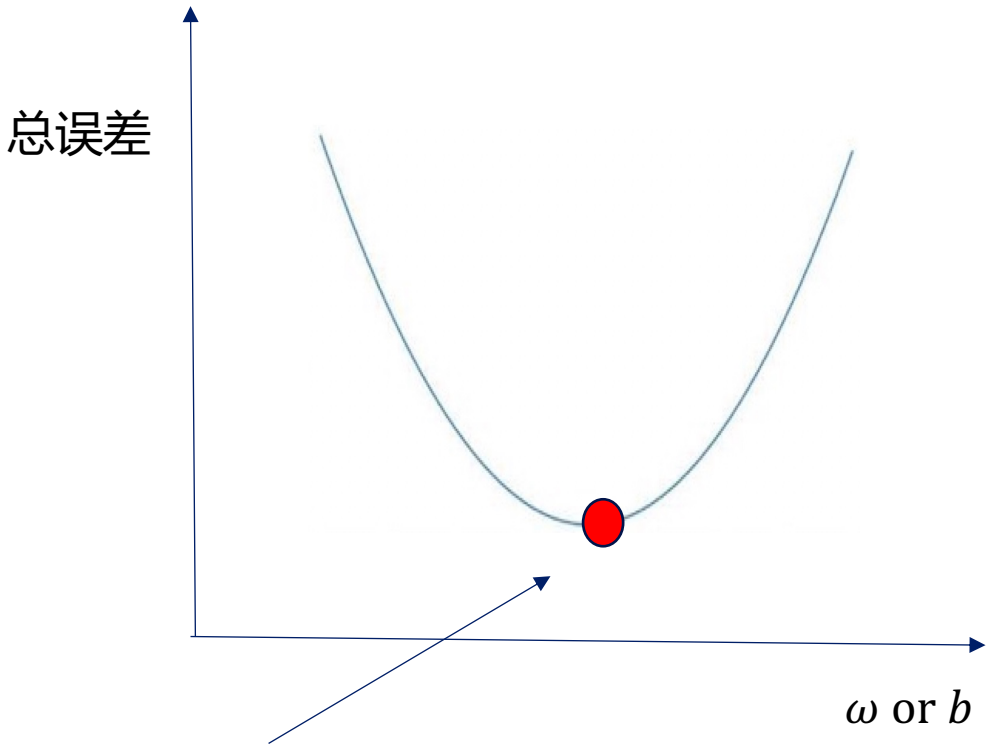
1. 设直线方程为： $Y = \omega x + b$

2. 每个样本误差为： 真实值 - 预测值

3. 所有样本误差为： 每个样本误差的平方和

4. 写成公式： $J(\omega, b) = \frac{1}{2}\sum(y_i - (\omega x_i + b))^2$

5. 训练的目标： 当总误差最小时的 ω 和 b 的值



y_i和x_i已知，只要求出最小值对应的(ω, b)即可。

索回归模型 (LASSO)

预测函数：

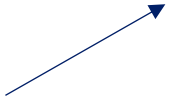
$$Y = \omega_1 x_1 + \omega_2 x_2 + b$$

总误差（多元回归）：

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2$$

总误差（ LASSO ）：

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2 + \lambda(w_1 + w_2)$$



求总误差最小值时，会使得部分ω值很小或为0；

从而使某些特征失效，有特征选择作用；

岭回归模型 (Ridge)

预测函数：

$$Y = \omega_1 x_1 + \omega_2 x_2 + b$$

总误差（多元回归）：

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2$$

总误差（ Ridge ）：

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2 + \lambda({\omega_1}^2 + {\omega_2}^2)$$

求总误差最小值时，会使得参数ω_1, ω_2趋近；

没有特征选择作用，

Elastic Net模型

预测函数：

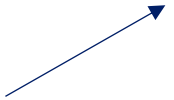
$$Y = \omega_1 x_1 + \omega_2 x_2 + b$$

总误差（多元回归）：

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2$$

总误差（ Ridge ）：

$$J = \frac{1}{2}\sum (y_i - (w_1 x_i + w_2 x_i + b))^2 + \lambda_1 (w_1 + w_2) + \lambda_2 ({\omega_1}^2 + {\omega_2}^2)$$



求总误差最小值时，会使得部分ω值很小或为0；

从而使某些特征失效，有特征选择作用；

线性回归模型：Linear，Ridge, LASSO, Elastic Net

其实就是线性回归加上正则项 (λ为惩罚系数)

线性回归模型 ：$J = \frac{1}{2}\sum (Y - \bar{Y})^2$

岭回归模型 ： $J = \frac{1}{2}\sum (Y - \bar{Y})^2 + \lambda \sum W_i^2$

LASSO回归模型 : $J = \frac{1}{2}\sum (Y - \bar{Y})^2 + \lambda \sum W_i$

Elastic Net回归模型 : $J = \frac{1}{2}\sum (Y - \bar{Y})^2 + \lambda_1 \sum W_i^2 + \lambda_2 \sum W_i$

线性回归扩展：多项式回归

曲线拟合　　http://127.0.0.1:8888/notebooks/C2/模型训练.ipynb

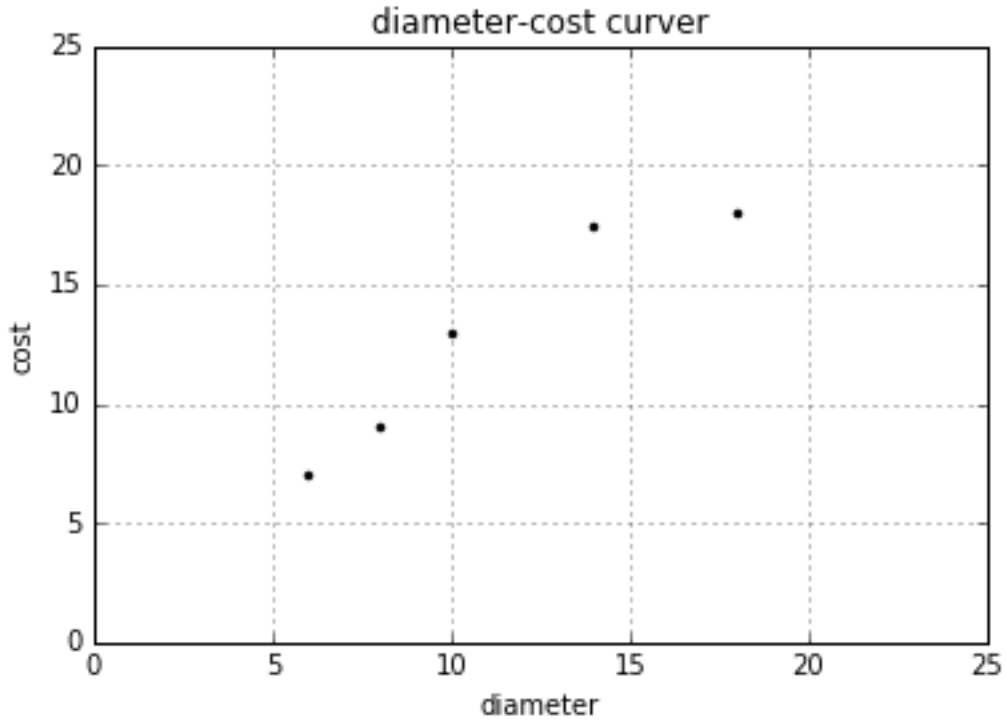
将输入X增加多个维度，扩展为多项式

$$\vec{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \longrightarrow \vec{Y} = \begin{bmatrix} x_1 \\ x_2 \\ {x_1}^2 \\ {x_2}^2 \\ x_1 x_2 \end{bmatrix}$$

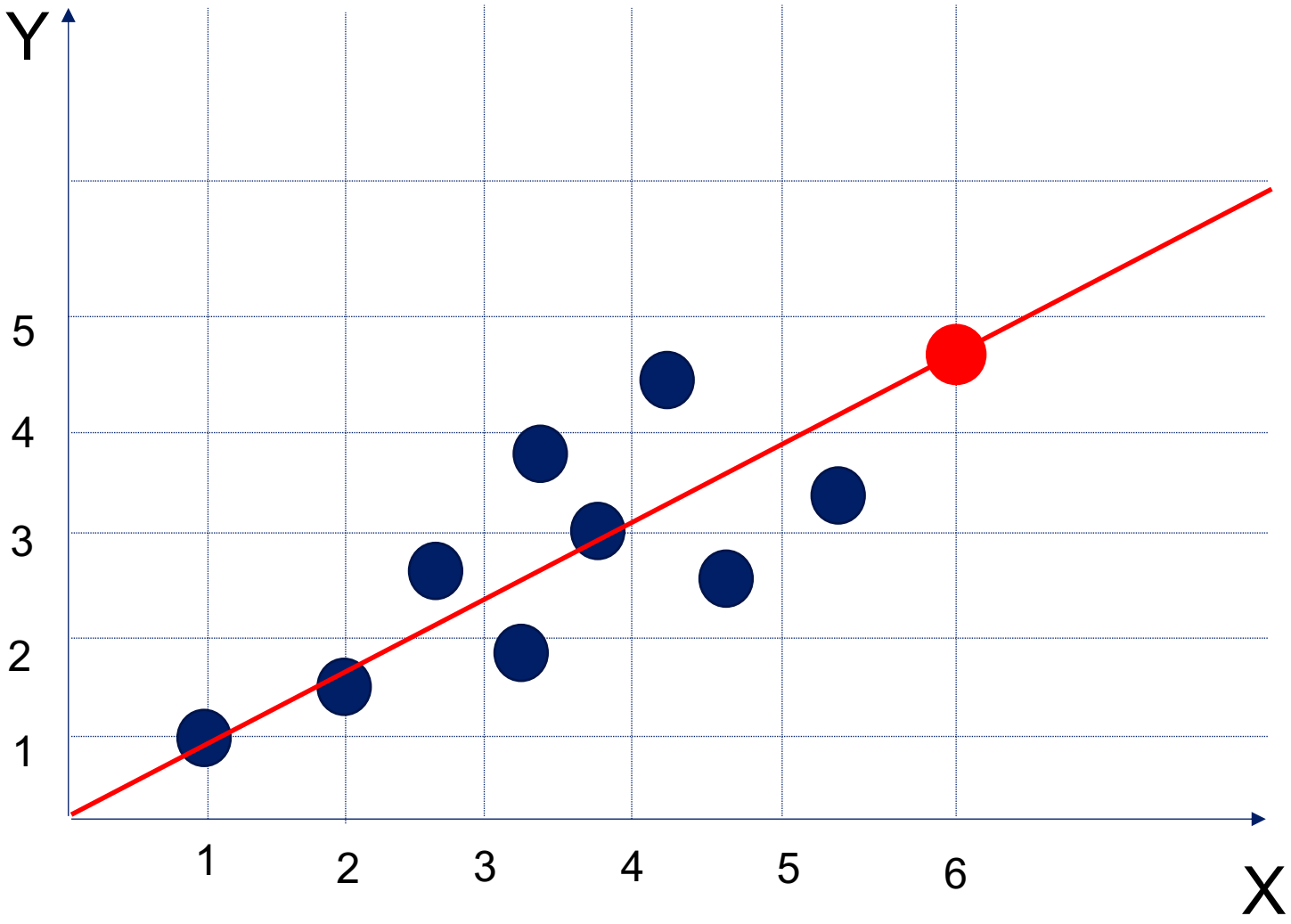
```
1  from sklearn.linear_model import Ridge
2  from sklearn.preprocessing import PolynomialFeatures
3  from sklearn.pipeline import make_pipeline
4
5  def test_poly(degree = 12):
6      model = make_pipeline(PolynomialFeatures(degree), Ridge())
7      model.fit(X_train,Y_train)
8      pred_y = model.predict(X_test)
9      error = (np.sqrt(sum((pred_y-Y_test)**2/Y_test.size)))
10     df4['Pred'] = pd.Series(pred_y.reshape(-1,).tolist())
11     print("degree ",degree,"error ",error)
12     sns.scatterplot(x="X1", y="Y",s=150,data=df4)
13     sns.lineplot(x="X1",y="Pred",data=df4,color='red')
14     plt.show()
15
16 test_poly(1)
17 test_poly(2)
18 test_poly(3)
19 test_poly(5)
20 test_poly(50)
```

定理

一定可以找到一个多项式，使得可以过所有点



QQ



Why 100% when doing add ?