

机器学习概述及回归算法

机器学习概述

机器学习概念

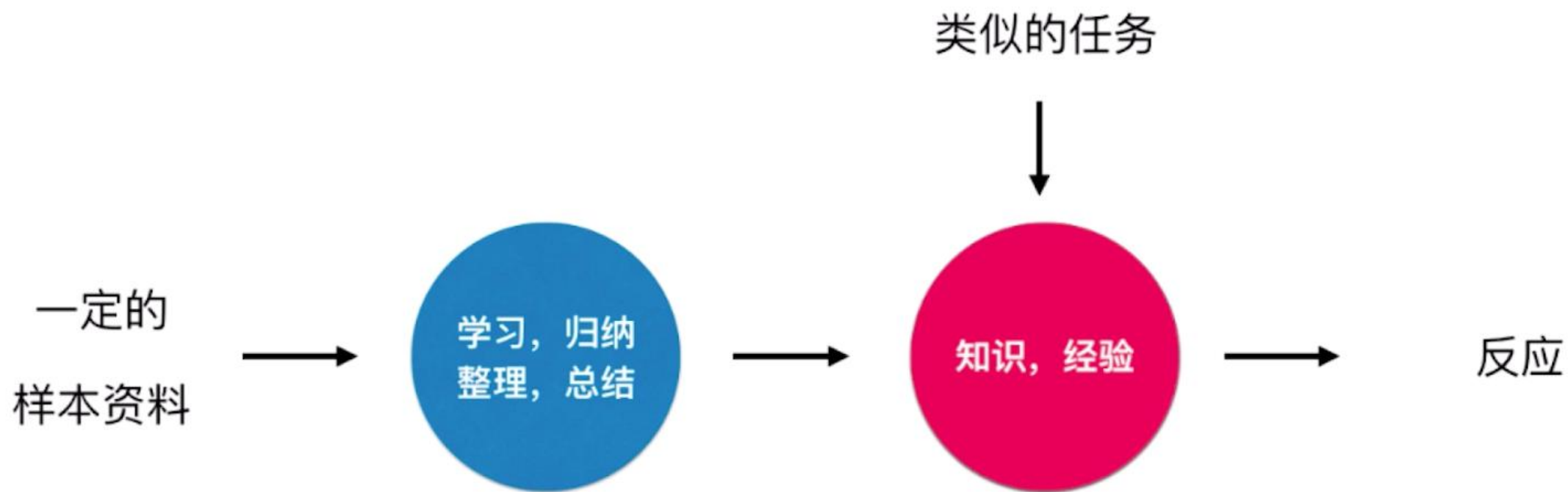
概念：

- 机器学习即让计算机在没有被显式编程的情况下，具备自我学习的能力。另一解释：就是把无序的数据转换成有用的信息。
- 例如：对于垃圾邮件，当某几个特定单词同时出现时，再辅以考察邮件长度及其他因素，人们可以准确的判断该邮件是否为垃圾邮件。

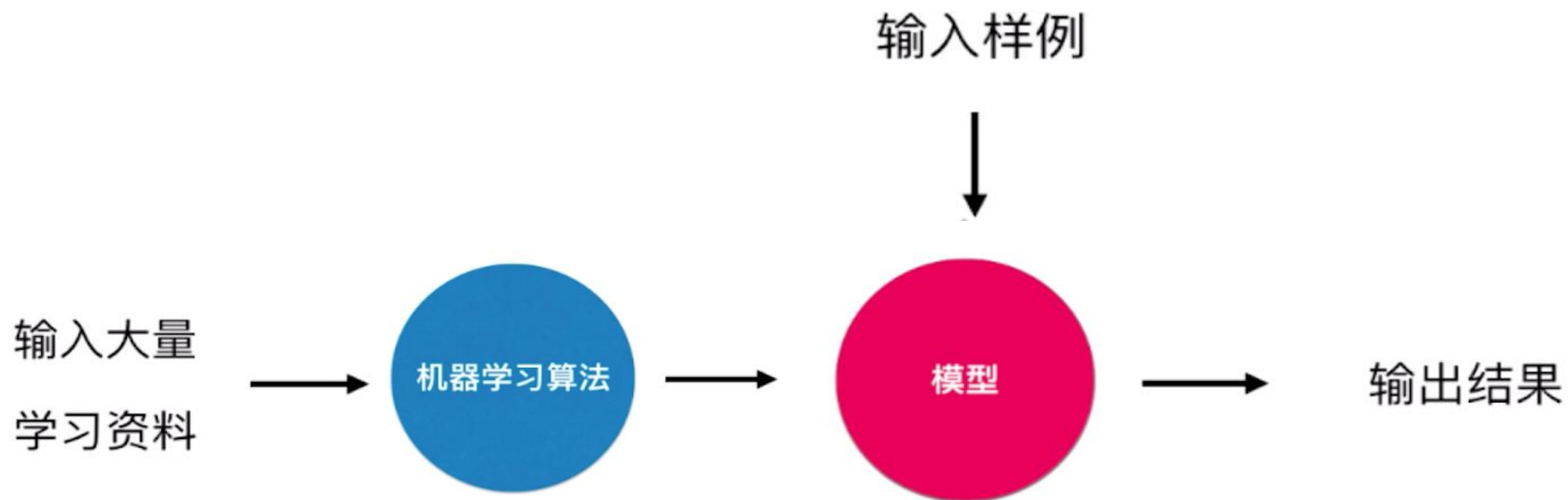
范围：

- 机器学习横跨计算机科学、工程技术和统计学等多个学科

人类怎么学习



什么是机器学习



数据

- 著名的鸢尾花数据

[https://en.wikipedia.org/wiki/Iris_flower_data_s](https://en.wikipedia.org/wiki/Iris_flower_data_set)



Iris setosa



Iris versicolor



Iris virginica

数据

Iris Plants Database

=====

Notes

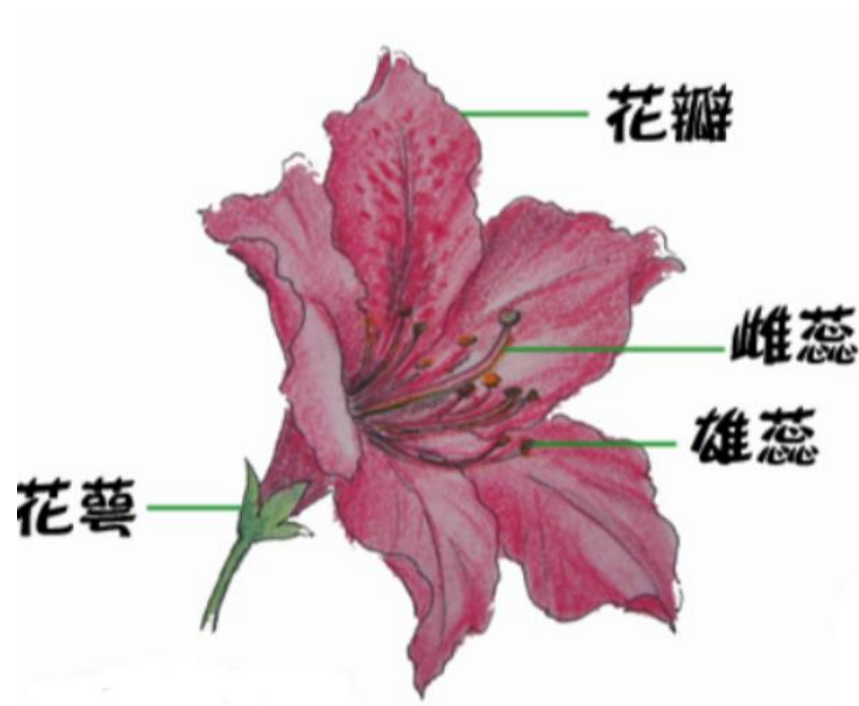
Data Set Characteristics:

:Number of Instances: 150 (50 in each of three classes)

:Number of Attributes: 4 numeric, predictive attributes and the class

:Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
 - Iris-Setosa
 - Iris-Versicolour
 - Iris-Virginica



数据

萼片长度	萼片宽度	花瓣长度	花瓣宽度	种类
5.1	3.5	1.4	0.2	se (0)
7.0	3.2	4.7	1.4	ve (1)
6.3	3.3	6	2.5	vi (2)

数据

萼片长度	萼片宽度	花瓣长度	花瓣宽度	种类
5.1	3.5	1.4	0.2	se (0)
7.0	3.2	4.7	1.4	ve (1)
6.3	3.3	6	2.5	vi (2)

X y

-数据整体叫数据集(*data set*)

-每一行数据称为一个样本(*sample*)

-除了最后一列, 每一列表达样本的一个特征(*feature*)

-最后一列, 称为标记(*label*)

第*i*个样本行写作 $X^{(i)}$ 第*i*个样本第*j*个特征值 $X_j^{(i)}$ 第*i*个样本的标记写作 $y^{(i)}$

数据

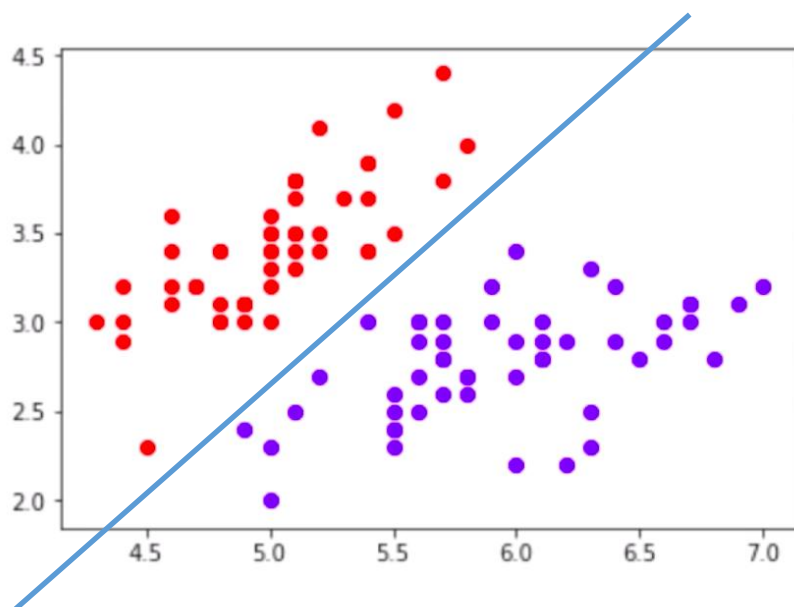
萼片长度	萼片宽度	花瓣长度	花瓣宽度	种类
5.1	3.5	1.4	0.2	se (0)
7.0	3.2	4.7	1.4	ve (1)
6.3	3.3	6	2.5	vi (2)

→ 特征

→ 特征向量 $X^{(i)}$

$$\begin{pmatrix} (X^{(1)})^T \\ (X^{(2)})^T \\ (X^{(3)})^T \\ \dots \end{pmatrix} \quad \begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix}$$

数据



- 特征空间(*feature space*)
- 分类任务本质就是在特征空间切分
- 在高维空间同理

机器学习分类

- 监督学习：算法预先知道预测什么，即目标分量的分类信息。
- 无监督学习：此类数据没有类别信息，也不给定目标值。在无监督学习中，将数据集成由类似的对象组成的多个类的过程被称为聚类。（预测房子价格、垃圾邮件）
- 半监督学习：通过学习大量的无标记数据，去分析出数据本身的内在特点和结构。（网上购物阴谋论）

根据类别常用算法

监督学习的用途

k -近邻算法

线性回归

朴素贝叶斯算法

局部加权线性回归

支持向量机

Ridge 回归

决策树

Lasso 最小回归系数估计

无监督学习的用途

K-均值

最大期望算法

DBSCAN

Parzen窗设计

如何选择合适算法

- 首先考虑机器学习使用的目的。
- 如果想要预测目标变量的值，则可以选择监督学习算法，否则选择无监督学习算法。
- 如果选定监督学习算法，需要进一步确定目标变量类型：
 - A.如果目标变量是连续的数值，如0.0~99.0，-1000~100000等，则需要选择回归算法。
 - B.如果目标是离散型，如是/否、1/2/3、A/B/C或者红/黑/黄，可以选择分类器算法。
- 如果不想预测目标的值，则可以选择无监督学习算法。进一步分析是否需要将数据划分为离散的组。如果这是唯一的需求，则使用聚类算法；如果还需要估计数据与每个分组的相似程度，则需要使用密度估计算法。

如何选择合适的算法

- 其次需要考虑的是数据问题。
- 重点考虑数据的以下特征，特征值是离散型变量还是连续型变量，特征值中是否存在缺失的值，何种原因造成缺失值，数据中是否存在异常值，某个特征发生的频率如何等。

开发机器学习应用的步骤

- (1)收集数据：比如可以用爬虫从网站抽取数据、设备发来的实测数据，也可以使用公开可用的数据源。
- (2)准备输入数据：得到数据之后，确定数据格式符合要求，此处需要注意有些算法要求**目标变量**和**特征值**是字符串类型，而另一些算法则可能要求是整数类型，具体情况具体分析。
- (3)分析输入数据。确保数据不是空值或者数据集中没有垃圾数据，此处如果信任数据源可以跳过，否则人工干预降低系统的价值。
- (4)训练算法。将前两步得到的格式化数据输入算法，从中抽取知识或信息。如果是无监督算法，因为不存在目标变量值，故而不需要训练算法

开发机器学习应用的步骤

- (5)测试算法：进一步将实际应用第四步机器学习得到的知识信息。
- (6)使用算法：将机器算法转换为应用程序，执行实际任务。

机器学习常用概念

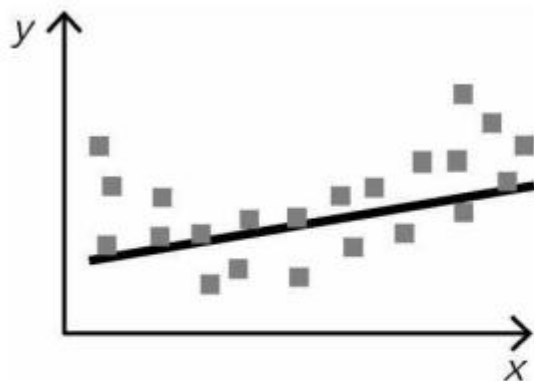
- 成本：是衡量模型与训练样本符合程度的指标，也即是针对所有的训练样本，模型拟合出来的值与训练样本的真实值的误差平均值。
- 成本函数：是成本与模型参数的函数关系。
- 模型训练：模型训练的过程，就是找出合适的模型参数，使得成本函数的值最小。也即是训练这个模型的目标，找出合适的模型参数，使得所有的点到直线上的距离最短。（此处补图）

机器学习常用概念

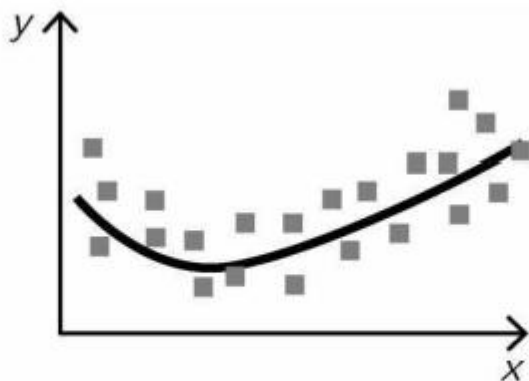
- 模型准确性：数据集一般分成训练数据集和测试数据集，划分的原则是 8:2 或者 7:3，然后用训练数据集来训练模型，训练出来模型参数后再使用测试数据集来测试模型的准确性，并根据模型的准确定来评价模型的性能。

机器学习常用概念

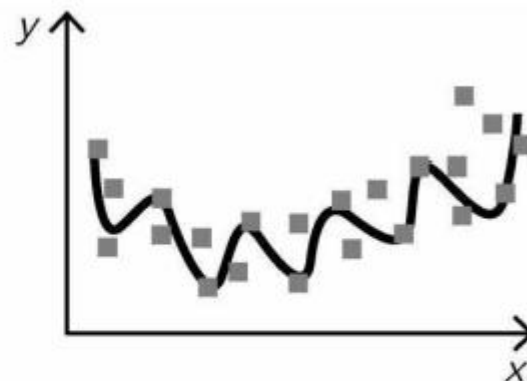
- 过拟合：是指模型能很好的拟合训练样本，但对新数据的预测准确性很差
- 欠拟合：是指模型不能很好的拟合训练样本，且对新数据的预测准确性也不好。



(a) 欠拟合

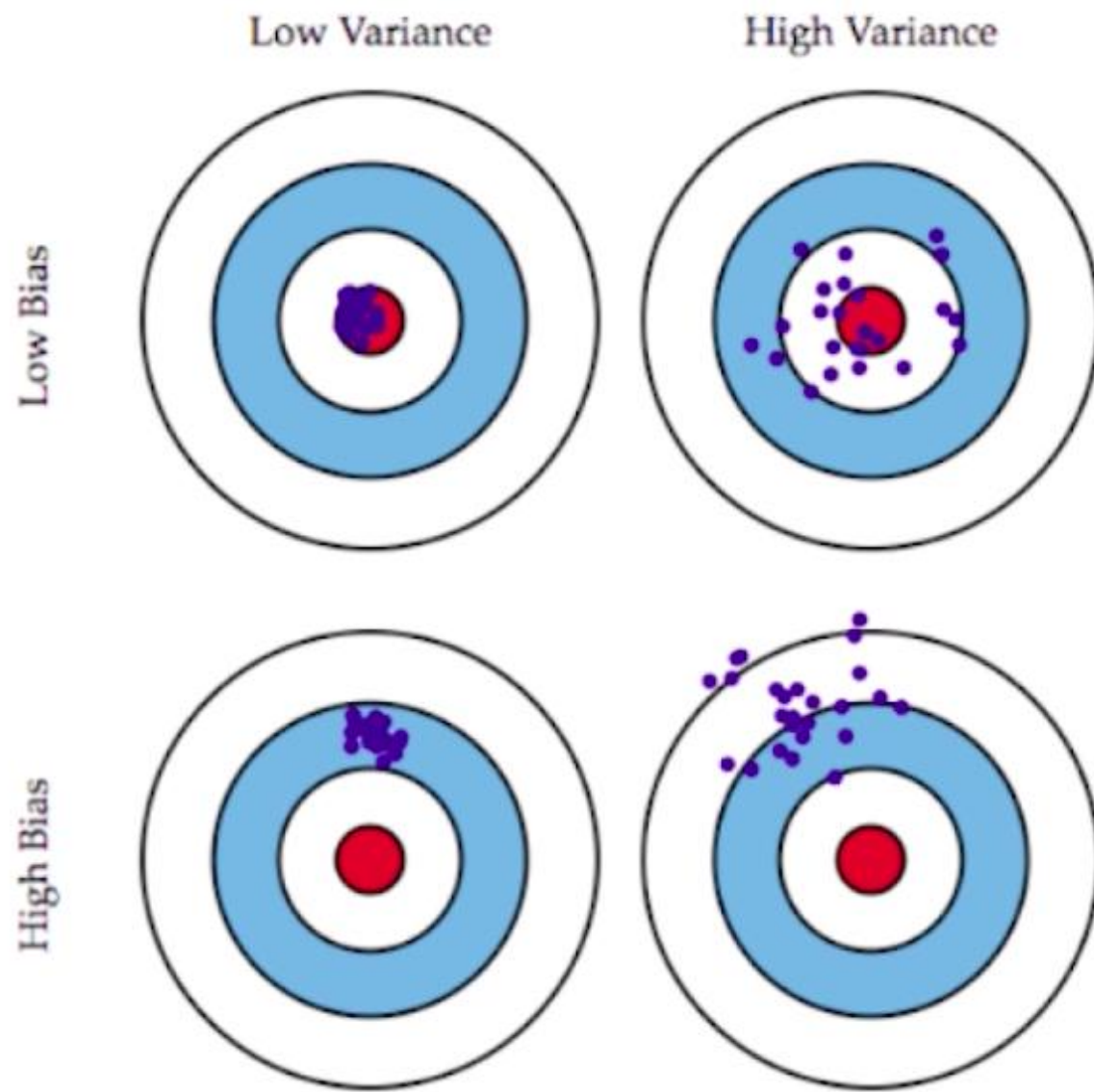


(b) 正常模型



(c) 过拟合

偏差和方差



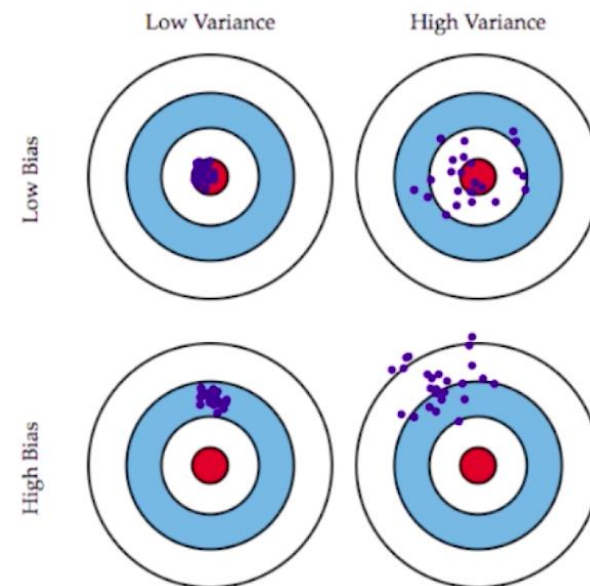
模型误差

模型误差=偏差（Bias）+ 方差（Variance）+ 不可避免的误差

偏差 (BIAS)

导致偏差的主要原因:
对问题本身的假设不正确
如: 非线性数据使用线性回归

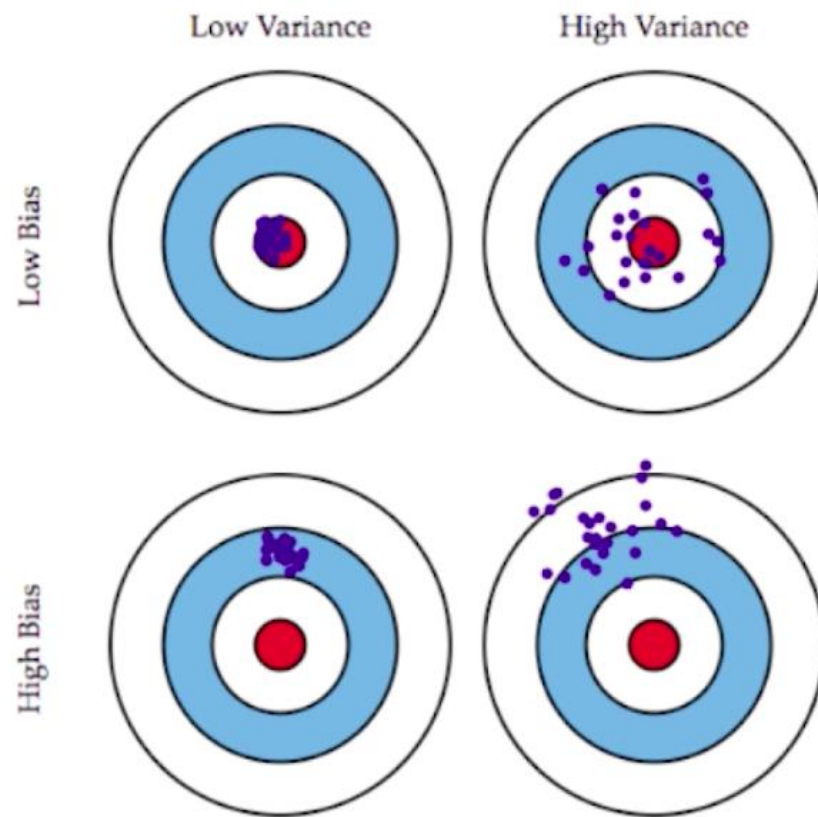
欠拟合underfitting



方差 (VARIANCE)

数据的一点点扰动都会较大地影响模型。
通常原因，使用的模型太复杂如高阶多项式回归。

过拟合overfitting



线性回归算法

线性回归算法是使用线性方程对数据集进行拟合的算法，是一个非常常见的回归算法。

安装scikit-learn框架

- 准备工作:
- Python (≥ 2.6 or ≥ 3.3),
- Numpy ($\geq 1.6.1$)
- Scipy (≥ 0.9),
- Matplotlib (可选) .
- *NumPy*

准备工作

- NumPy系统是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比Python自身的嵌套列表（nested list structure）结构要高效的多（该结构也可以用来表示矩阵（matrix））。
- *Scipy*
- SciPy是一款方便、易于使用、专为科学和工程设计的Python工具包。它包括统计,优化,整合,线性代数模块,傅里叶变换,信号和图像处理,常微分方程求解器等等。
- *Matplotlib*
- matplotlib 是python最著名的绘图库，它提供了一整套和matlab相似的命令API，十分适合交互式地进行制图。而且也可以方便地将它作为绘图控件，嵌入GUI应用程序中。

下载地址

- **Python:** <https://www.python.org/downloads/>
 - **Numpy:** <http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>
 - **Scipy:** <http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy>
 - **Matplotlib:** <http://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib>
 - **scikit-learn:** <http://www.lfd.uci.edu/~gohlke/pythonlibs/#scikit-learn>
-

具体安装步骤

- 先安装python， 下载好的exe文件右键**管理员运行**安装，记得勾选**加入到环境变量**。
- 然后依次安装**numpy**、**scipy**和**matplotlib**。
- 将下载好的文件放到一个文件夹里，比如 d:\ScikitLearn\
- 开始->运行->cmd 打开命令行
- `cd d:\ScikitLearn` #切换目录到ScikitLearn `dir` #列出目录下的文件检查是否缺少文件，不缺少进行下一步：
- `pip install .\numpy-1.11.3-cp35-cp35m-win_amd64.whl` #安装numpy如果正确安装会返回这些信息：
- `Processing c:\src\numpy-1.11.3-cp35-cp35m-win_amd64.whl` Installing collected packages: numpy Successfully installed numpy-1.11.3+mk1同样方法安装 **scipy** 、 **matplotlib**。
- `pip install .\scipy-0.19.0-cp35-cp35m-win_amd64.whl` #安装scipy `pip install .\matplotlib-2.0.0-cp35-cp35m-win_amd64.whl` #安装matplotlib最后安装 **scikit-learn**
- `pip install .\scikit_learn-0.18.1-cp35-cp35m-win_amd64.whl` #安装scikit-learn

预测房价

- 使用scikit-learn的波士顿房价预测数据集，共收集了13个特征，具体如下：
- CRIM:城镇人均犯罪率
- ZN：城镇超过25,000平方英尺的住宅区域的占地比例
- INDUS：城镇非零售用地占地比例
- CHAS：是否靠近河边，1为靠近，0为远离
- NOX：一氧化氮浓度
- RM：每套房产的平均房间个数
- AGE：在1940年之前就盖好，且业主自住的房子的比例

预测房价

- 使用scikit-learn的波士顿房价预测数据集，共收集了13个特征，具体如下：
- DIS：与波士顿市中心的距离
- RAD：周边高速公路的便利性指数
- TAX：每10,000美元的财产税率
- PTRATIO：小学老师的比例
- B：城镇黑人的比例
- LSTAT：地位较低的人口比例

数据导入

- #数据导入

```
from sklearn.datasets import load_boston  
boston = load_boston()  
X = boston.data  
y = boston.target  
X.shape  
print(X[0])
```

- 可以通过X[0]来查看一个样本的数据
- X.shape 查看数据集的样本数和特征数
- boston.feature_names: 查看特征

模型训练

- *#划分数数据集*, 我们选择20%的样本作为测试数据集
- **from** sklearn.model_selection **import** train_test_split
- X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.2,random_state=3)

训练数据

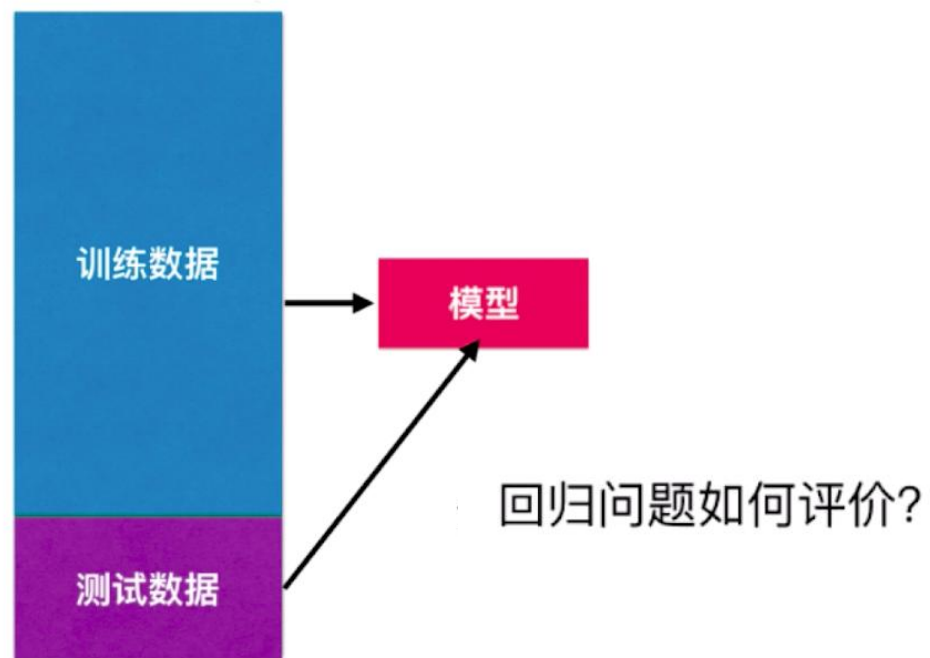
```
from sklearn.linear_model import LinearRegression #导入训练模型
import time #统计时间
#接下来训练模型， 获得参数的值
model = LinearRegression()
start = time.clock()
model.fit(X_train,y_train)
train_score = model.score(X_train, y_train)
cv_score = model.score(X_test, y_test)
print('elaspe:{0:.6f};train_score:{1:0.6f};
cv_score:{2:0.6f}'.format(time.clock()-start,train_score,cv_score))
```

模型测评

- 得分情况: `elaspe:0.100511;train_score:0.723941; cv_score:0.794958`
- `train_score:0.723941`; #训练得分
- `cv_score:0.794958` #测试得分
- 从得分情况分析: 模型的拟合效果一般
- 如何优化?


回归算法的评价


回归算法的评价



简单线性回归

目标：找到a和b，使得 $\sum_{i=1}^m (y_{train}^{(i)} - ax_{train}^{(i)} - b)^2$ 尽可能小


$$\hat{y}_{test}^{(i)} = ax_{test}^{(i)} + b$$


$$\sum_{i=1}^m (y_{train}^{(i)} - \hat{y}_{train}^{(i)})^2$$

衡量标准：

$$\sum_{i=1}^m (y_{test}^{(i)} - \hat{y}_{test}^{(i)})^2$$

线性回归算法的评测

衡量标准: $\sum_{i=1}^m (y_{test}^{(i)} - \hat{y}_{test}^{(i)})^2$

问题: 和m相关?

线性回归算法的评测

$$\frac{1}{m} \sum_{i=1}^m (y_{test}^{(i)} - \hat{y}_{test}^{(i)})^2$$

均方误差 MSE
(Mean Squared Error)

问题：量纲？

线性回归算法的评测

$$\sqrt{\frac{1}{m} \sum_{i=1}^m (y_{test}^{(i)} - \hat{y}_{test}^{(i)})^2} = \sqrt{MSE_{test}}$$

均方根误差 RMSE
(Root Mean Squared Error)

线性回归算法的评测

$$\frac{1}{m} \sum_{i=1}^m |y_{test}^{(i)} - \hat{y}_{test}^{(i)}|$$

平均绝对误差 MAE
(Mean Absolute Error)

RMSE vs MAE

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m (y_{test}^{(i)} - \hat{y}_{test}^{(i)})^2}$$

$$MAE = \frac{1}{m} \sum_{i=1}^m |y_{test}^{(i)} - \hat{y}_{test}^{(i)}|$$

量纲是一样的, 但实际中RMSE会比MAE大一些。原因?

RMSE放大了错误值, 所以让RMSE更小, 意义更大一些

RMSE vs MAE

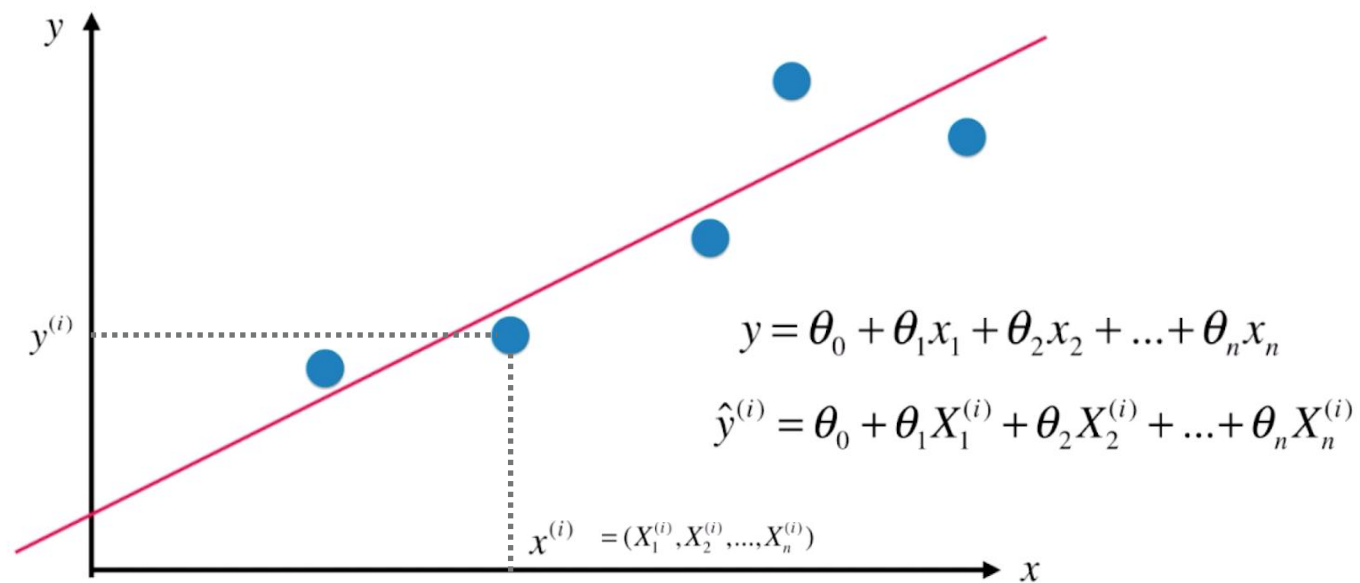
$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m (y_{test}^{(i)} - \hat{y}_{test}^{(i)})^2} \quad MAE = \frac{1}{m} \sum_{i=1}^m |y_{test}^{(i)} - \hat{y}_{test}^{(i)}|$$

问题：分类的准确度：1最好，0最差

RMSE? MAE?

多元线性回归

多元线性回归



多元线性回归

目标：使 $\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$ 尽可能小

$$\hat{y}^{(i)} = \theta_0 + \theta_1 X_1^{(i)} + \theta_2 X_2^{(i)} + \dots + \theta_n X_n^{(i)}$$

目标：找到 $\theta_0, \theta_1, \theta_2, \dots, \theta_n$ ，使得 $\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$ 尽可能小

多元线性回归

$$\hat{y}^{(i)} = \theta_0 + \theta_1 X_1^{(i)} + \theta_2 X_2^{(i)} + \dots + \theta_n X_n^{(i)}$$

$$\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_n)^T$$

$$\hat{y}^{(i)} = \theta_0 X_0^{(i)} + \theta_1 X_1^{(i)} + \theta_2 X_2^{(i)} + \dots + \theta_n X_n^{(i)} \quad , X_0^{(i)} \equiv 1$$

$$X^{(i)} = (X_0^{(i)}, X_1^{(i)}, X_2^{(i)}, \dots, X_n^{(i)})$$

$$\hat{y}^{(i)} = X^{(i)} \cdot \theta$$

多元线性回归

$$X_b = \begin{pmatrix} 1 & X_1^{(1)} & X_2^{(1)} & \dots & X_n^{(1)} \\ 1 & X_1^{(2)} & X_2^{(2)} & \dots & X_n^{(2)} \\ \dots & & & & \\ 1 & X_1^{(m)} & X_2^{(m)} & \dots & X_n^{(m)} \end{pmatrix} \quad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{pmatrix}$$

$$\hat{y} = X_b \cdot \theta$$

多元线性回归

$$\hat{y} = X_b \cdot \theta$$

目标：使 $\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$ 尽可能小



目标：使 $(y - X_b \cdot \theta)^T (y - X_b \cdot \theta)$ 尽可能小

多元线性回归

$$\theta = (X_b^T X_b)^{-1} X_b^T y$$

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{pmatrix}$$

→ 截距 intercept

→ 系数 coefficients

Thanks