—武大本科生课程

第2讲　机器学习基本方法入门

(Lecture 2 Getting started with basic methods of machine learning)

武汉大学计算机学院机器学习课程组

Email: snowfly_li@163.com

本讲内容目录

Review:

1. 机器学习的定义

目前机器学习还没有一个准确、统一的定义。

经典定义：计算机程序如何随着经验的积累自动改善自身的性能 **[T.Mitchell, CMU *Book* 97]**。

Definition: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (对于某类任务T和性能度量P，若一个计算机程序在T上以P衡量的性能随着经验E而自我完善，那么我们称该计算机在从经验E中学习。)---从学术的角度定义

“Machine Learning” as an Engineering Paradigm: Use data and examples, instead of expert knowledge, to automatically create systems that perform complex tasks. ---从工程的角度定义

通俗地说，机器学习方法就是机器 (计算机)利用已有的**数据(经验)**，**通过对数据/经验的学习**得出某种**模型(规律/机器学习算法)**，并利用此模型对以前从未见过的数据进行**预测 (推断/决策)**的一种方法。

2. 模式识别的定义

Pattern recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background, and make sound and reasonable decisions about the categories of the patterns. (Anil K. Jain)

模式识别与机器学习的关系：模式识别≈机器学习。两者的主要区别在于前者是从工业界发展起来的概念，后者则主要源自计算机学科。

图　人工智能、机器学习、深度学习三者关系

人工智能: Artificial Intelligence, 简称AI

机器学习: Machine Learining, 简称ML

深度学习: Deep Learinig, 简称DL

Review (Cont.):

3. 机器学习技术分类

从机器学习的过程看 ，机器学习算法(或者机器学习模型)可分为：

(1)有监督学习 (Supervised Learning)
(2)无监督学习 (Unsupervised Learning)
(3)半监督学习 (Semi-Supervised Learning)
(4)强化学习 (Reinforcement Learning, RL)

机器学习算法以数据为对象，它通过提取数据特征，发现数据中的知识并抽象出数据模型，作出对数据的预测。

作为机器学习算法的对象，数据包括存在于计算机及网络上的各种数字、文字、图像、音频、视频等，以及它们的组合。

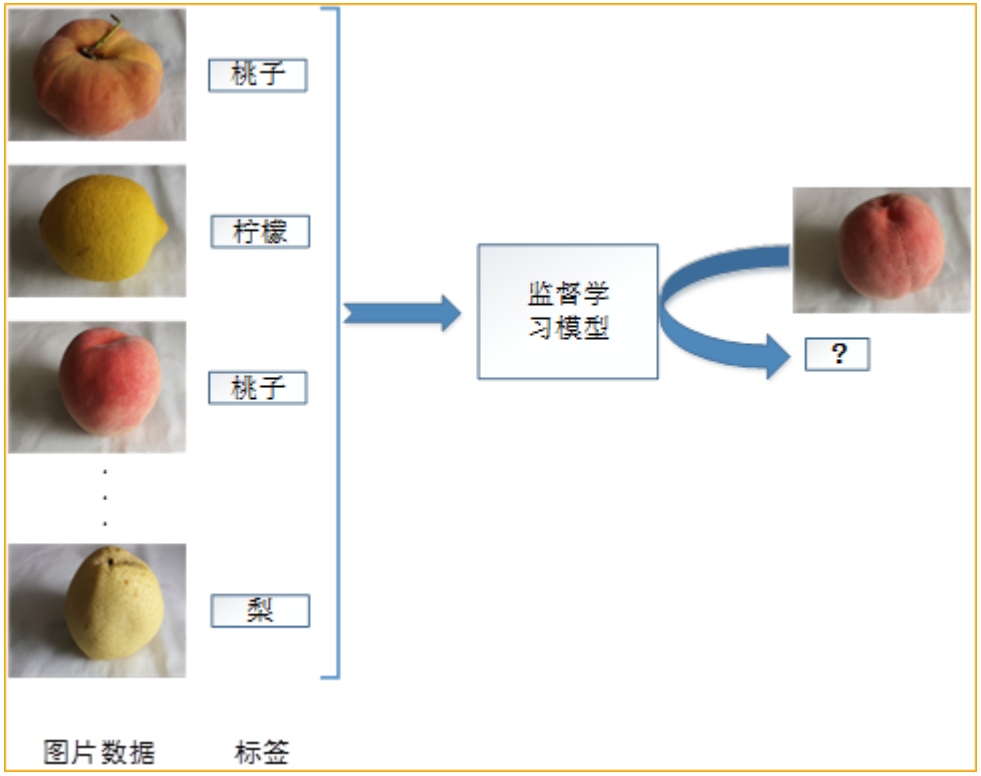
用来作为抽象模型发现知识的数据称为训练数据，需要作出预测的数据称为测试数据。

机器学习算法能够有效的前提是同类数据（包括训练数据和测试数据等）具有相同的统计规律性这一基本假设。

Review(Cont.):

监督学习、无监督学习和半监督学习

监督学习处理的对象是所谓的有标签训练数据，它利用有标签的训练数据来学习一个模型，它的目标是用学到的模型给无标签的测试数据打上标签。常见的分类和回归问题都属于监督学习。



无监督学习的训练数据没有标签，它自动从训练数据中学习知识，建立模型。

半监督学习是监督学习和无监督学习相结合的一种学习方法，其基本原则是通过大量无标记数据辅助少量已标记数据进行学习，从而提高学习效果。

一些典型的有监督学习算法

- **K-近邻算法 (K-Nearest Neighbors, KNN)**
- **线性回归 (Linear Regression)**
- 逻辑回归 (Logistic Regression)
- 支持向量机 (Support Vector Machines, SVM)
- 决策树和随机森林 (Decision Trees & Random Forests)
- 神经网络 (Neural Networks)

一些典型的无监督学习算法

- **聚类算法**
  - K-均值聚类(K-Means)
  - 层次聚类分析(Hierarchical Cluster Analysis)
  - 概率聚类分析(Probabilistic Cluster Analysis)

- **降维算法**
  - 主成分分析(PCA)
  - 核主成分分析(K-PCA)
  - 局部线性嵌入(LLE)　其他：IsoMap，MDS

- 关联规则学习算法
  - Apriori
  - Eclat

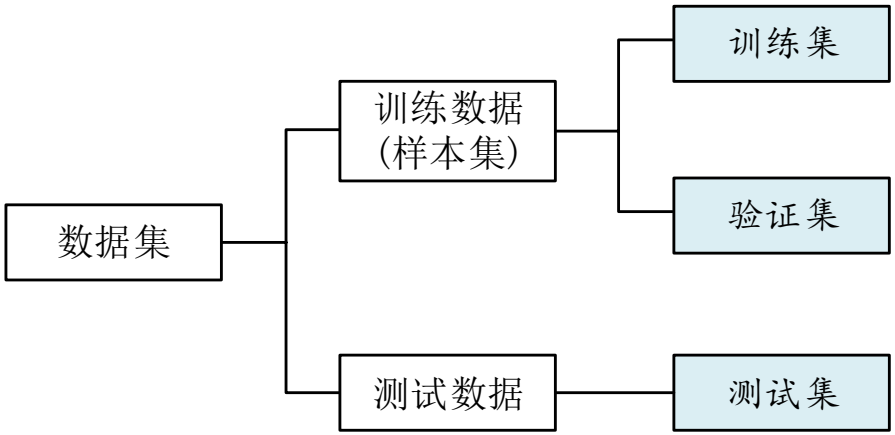
Review(Cont.):

一些基本概念回顾性小结：

- 样本 (sample)：一个具体的研究对象，具有一个或多个可观测量。
- 特征 (features)：能从某个方面对样本进行描述、刻画或表达的可观测量 (数值型、结构型)。多个特征通常用向量表示——特征向量。
- 模式 (pattern)：样本特征向量的观测值，是抽象样本的数值代表。在这个意义上，“样本”与“模式”说的是同一件事情。
- 类别、类属、模式类 (class)：指在一定合理颗粒度下、有实际区分意义的基础上，主观或客观地被归属于“同一类” 的客观对象 (样本、模式) 的类别标号/标签。数学上一般处理为整数。
- 样本集 (sample set)：多个样本的集合。训练集、验证集、测试集。
- 已知样本 (known samples，example)：事先知道类别标号/标签的样本。
- 未知样本 (unknown samples)：指特征已知、但类别标号未知的样本，也称：待分样本、待识样本。
- 机器学习 (Machine Learning)：称已知样本为样例(example)，未知样本为实例 (instance)。通过学习/训练样本数据，发现规律，得到模型的参数，从而能得到预测目标的模型。
- 模式识别 (pattern recognition)：基于样本和应用目的设计分类器；并对性能进行验证，进行必要优化；最后用分类器对未知类属样本进行类属判定。识别，不是宽泛的认识或理解，而是对类别判定。

Review(Cont.):

术语及其表示 (训练集, 验证集, 测试集)



数据集(Data set)分为训练数据和测试数据。测试数据即为测试集(Test set)，是需要应用模型进行预测的那部分数据，是机器学习所有工作的最终服务对象。为了防止训练出来的模型只对训练数据有效，一般将训练数据又分为训练集和验证集，训练集(Training set)用来训练模型，而验证集(Validation set)一般只用来验证模型的有效性，不参与模型训练。

在有监督的分类模型中，训练集和验证集都是事先标记好的有标签数据，测试集是无标记的数据。在无监督模型中，训练集、验证集和测试集都是无标记的数据。

2.1 机器学习与模式识别的几个基本问题

1. 模式(样本)表示方法

(1)向量表示: 假设一个模式/样本有d个变量(d维特征)

$$x = (x_1, x_2, \ldots, x_d)^T$$

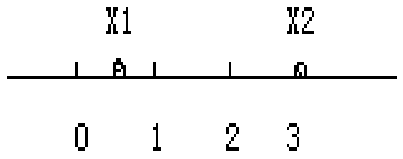
(2) 矩阵表示: N个样本，d个变量(特征)

变量 / 样本	x_1	x_2	…	x_d
$\mathbf{X}_1$	X_{11}	X_{12}	…	X_{1d}
$\mathbf{X}_2$	X_{21}	X_{22}	…	X_{2d}
…	…	…	…	…
$\mathbf{X}_N$	X_{N1}	X_{N2}	…	X_{Nd}

(3) 几何表示

1D表示

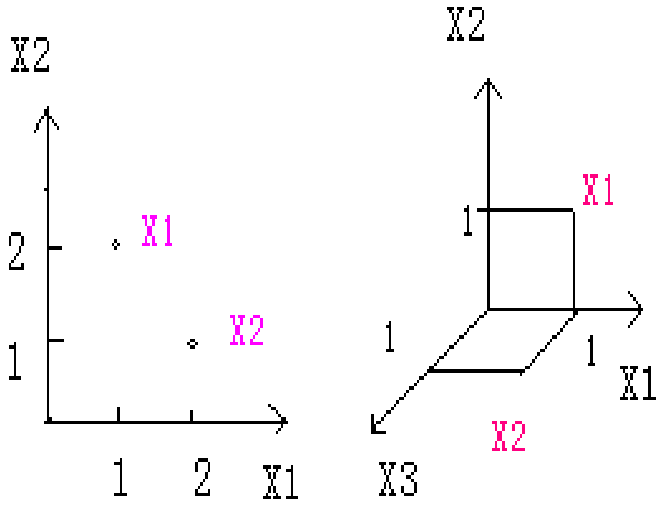
$X_1=0.5$　　$X_2=3$



2D表示

$X_1=(x_1, x_2)^T=(1, 2)^T$

$X_2=(x_1, x_2)^T=(2, 1)^T$



3D表示

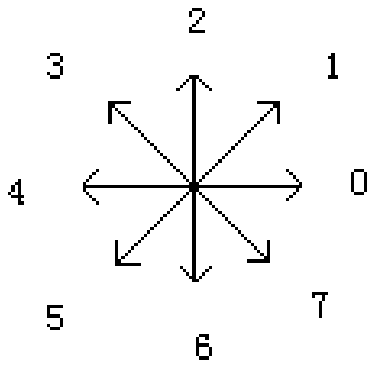
$X_1=(x_1, x_2, x_3)^T=(1, 1, 0)^T$

$X_2=(x_1, x_2\ , x_3)^T=(1, 0, 1)^T$

常用于数据可视化，以便进行探索性数据分析。

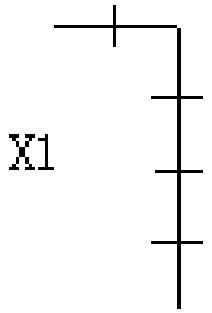
(4) 基元（链码）表示：

在右侧的图中八个基元分别表示0，1，2，3，4，5，6，7，八个方向和基元线段长度。



则右侧样本可以表示为

X_1=006666

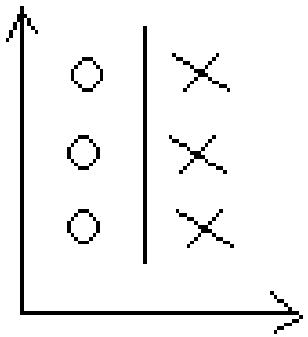


该方法在句法模式识别中会用到。

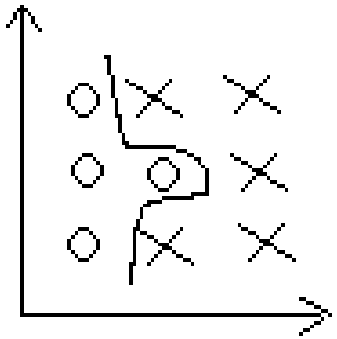
2. 模式类的紧致性

(1)紧致集

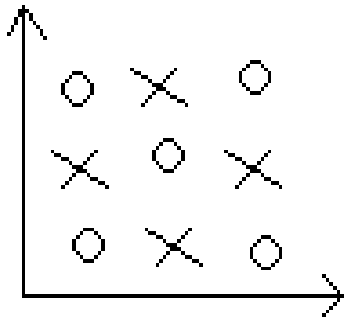
同一类模式类样本的分布比较集中，没有或临界样本很少，这样的模式类称紧致集(compact set)。



无临界点



有临界点



临界点太多，无法分类

(2) 临界点(样本)：在多类样本中，某些样本的值有微小变化时就变成另一类样本称为临界样本(点)。

(3) 紧致集的性质

① 要求临界点很少

② 集合内的任意两点的连线，在线上的点属于同一集合

③ 集合内的每一个点都有足够大的邻域，在邻域内只包含同一集合的点

(4) 机器学习与模式识别的要求：满足紧致集，才能很好的分类；如果不满足紧致集，就要采取变换的方法，以满足紧致集。

3. 相似与分类

(1)两个样本x_i，x_j之间的相似度量满足以下要求：

①应为非负值

②样本本身相似性度量应最大

③度量应满足对称性

④在满足紧致性的条件下，相似性应该是点间距离的单调函数

(2) 常用各种距离表示样本间的相似性：

①绝对值距离

已知两个样本，每个样本有n个特征：

$$\mathbf{x}_i=(x_{i1}, x_{i2}, x_{i3},\ldots,x_{in})^T$$

$$\mathbf{x}_j=(x_{j1}, x_{j2}, x_{j3},\ldots,x_{jn})^T$$

定义：

$$d_{ij}=\sum_{k=1}^{n}|x_{ik}-x_{jk}|$$

②欧几里德距离

$$d_{ij}=\sqrt{\sum_{k=1}^{n}\left(X_{ik}-X_{jk}\right)^{2}}$$

③明考夫斯基距离

$$d_{ij}(q)=\left(\sum_{k=1}^{n}\left|X_{ik}-X_{jk}\right|^{q}\right)^{1/q}$$

其中当q=1时为绝对值距离，当q=2时为欧氏距离

④切比雪夫距离

$$d_{ij}(\infty)=\max_{1\le k\le n}\left|X_{ik}-X_{jk}\right|$$

明氏距离，q趋向无穷大时的极限情况

⑤马氏距离(Mahalanobis：马哈拉诺比斯)：样本X_i和X_j间的马氏距离为

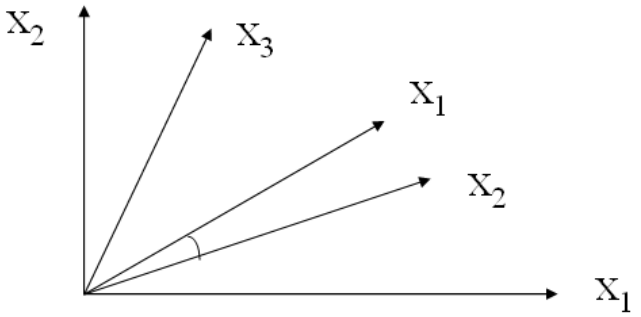
$$d_{ij}^{2}(M)=\left(X_i-X_j\right)^{T}\Sigma^{-1}(X_i-X_j)$$

其中X_i、X_j为特征向量集$\{X_1,X_2,\ldots,X_m\}$中的两个n维特征向量，∑为模式总体样本的协方差矩阵。马氏距离的使用条件是样本符合正态分布。

⑥夹角余弦 $$C_{ij}=\cos\theta=\frac{X_i^T\cdot X_j}{|X_i|\cdot|X_j|}=\frac{\sum_{k=1}^{n}X_{ik}X_{jk}}{\sqrt{\left(\sum_{k=1}^{n}X_{ik}^{\ 2}\right)\left(\sum_{k=1}^{n}X_{jk}^{\ 2}\right)}}$$

即两样本间夹角小的为一类，具有相似性。

例： x_1 , x_2 , x_3的夹角如右图。



因为x_1 , x_2 的夹角小，所以x_1 , x_2 最相似。

⑦ 相关系数

$$r_{ij}=\frac{s_{ij}^2}{\sqrt{s_{ii}^2 s_{jj}^2}}=\frac{\sum_{k=1}^{n}\left(X_{ki}-\overline{X_i}\right)\left(X_{kj}-\overline{X_j}\right)}{\sqrt{\sum_{k=1}^{n}\left(X_{ki}-\overline{X_i}\right)^2\sum_{k=1}^{n}\left(X_{kj}-\overline{X_j}\right)^2}}$$

$\overline{X_i},\overline{X_j}$ 为x_i 、 x_j的均值

注意：在求相关系数之前，要将数据标准化。

3. 分类的主观性和客观性

(1)分类带有主观性：目的不同，分类不同。如：鲸鱼、牛、马从生物学的角度来讲都属于哺乳类，但是从产业角度来讲鲸鱼属于水产业，牛和马属于畜牧业。

(2)分类的客观性：科学性

判断分类必须有客观标准，因此分类是追求客观性的，但主观性也很难避免，这就是分类的复杂性。

4. 特征的生成

(1)低层特征：

①无序尺度：有明确的数量和数值。

②有序尺度：有先后、好坏的次序关系， 如酒分为上、中、下三个等级。

③名义尺度：无数量、无次序关系，如有红、黄两种颜色。

(2) 中层特征：经过计算、变换得到的特征

(3) 高层特征：在中层特征的基础上有目的的经过运算形成

例如：椅子的重量=体积*比重

体积与长、宽、高有关；比重与材料、纹理、颜色有关。这里低、中、高三层特征都具备了。

5. 数据的标准化

(1) 极差标准化

一批样本中，每个特征的最大值与最小值之差，称为极差。

极差/全距 $R_i = \max X_{ij} - \min X_{ij}$

极差标准化 $Xij = \left(X_{ij} - \overline{X_i}\right) \Big/ R_i$

(2) 方差标准化

$Xij = \left(X_{ij} - \overline{X_i}\right) \Big/ S_i$

S_i为样本方差

标准化的方法很多，原始数据是否应该标准化，应采用什么方法标准化，都要根据具体情况来定。

2.2 机器学习编程环境及工具包使用举例

教学案例 (示例) 代码采用Python3 (Anaconda，Jupyter Notebook，JupyterLab，Spyder, etc.)，深度学习框架采用华为MindSpore (还可采用百度PaddlePaddel、Google Tensorflow(Keras)、Facebook PyTorch、Amazon AWS MxNet等)。

Numpy扩展包提供了数组支持。Pandas是Python的数据分析和探索工具。Scipy包提供矩阵支持，以及矩阵相关的数值计算模块。Scikit-Learn是一个基于python的机器学习扩展包。Matplotlib包主要用于绘图和绘表，是强大的数据可视化工具。SymPy包主要用于符号计算。

Jupyter Notebook (此前被称为 IPython notebook) 是一个交互式笔记本，支持运行40多种编程语言。 JupyterLab包含了Jupyter Notebook所有功能。

Anaconda Windows64位版本清华镜像下载地址：

https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2020.07-Windows-x86_64.exe

Python+NumPy基本编程举例

- **例：设*X*=[1.70 1.75 1.65 1.80 1.78]，编程求和、均值、方差、标准差。**

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \bar{X}\right)^2$$

方差(s^2：修正样本方差)是描述数据取值分散性的一个度量，它是数据相对于均值的偏差平方的平均值。

采用Python编写脚本：

```
#Filename: python_statistics_1.ipynb
import numpy as np
x=[1.7,1.75,1.65,1.80,1.78] #height data
su=np.sum(x)
m=np.mean(x)
vr=np.var(x)  #Compute the variance
sd=np.std(x)  #Compute the standard deviation
print('su=',su,'\nm=',m,'\nvr=',vr,'\nsd=',sd)
```

以上*.ipynb格式Python脚本可在IPython notebook/ Anaconda Jupyter notebook等环境下运行。

1. 本地电脑Jupyter notebook环境
程序运行情况：

localhost:8888/notebooks/python_statistics_1.ipynb

File Edit View Insert Cell Kernel Widgets Help

In [1]:

```
#Filename:python_statistics_1.ipynb
import numpy as np
x=[1.7,1.75,1.65,1.80,1.78] #height data
su=np.sum(x)
m=np.mean(x)
vr=np.var(x)   #Compute the variance
sd=np.std(x)   #Compute the standard deviation
print('su=',su,'\nm=',m,'\nvr=',vr,'\nsd=',sd)
```

```
su= 8.68
m= 1.736
vr= 0.002984
sd= 0.054626001135
```

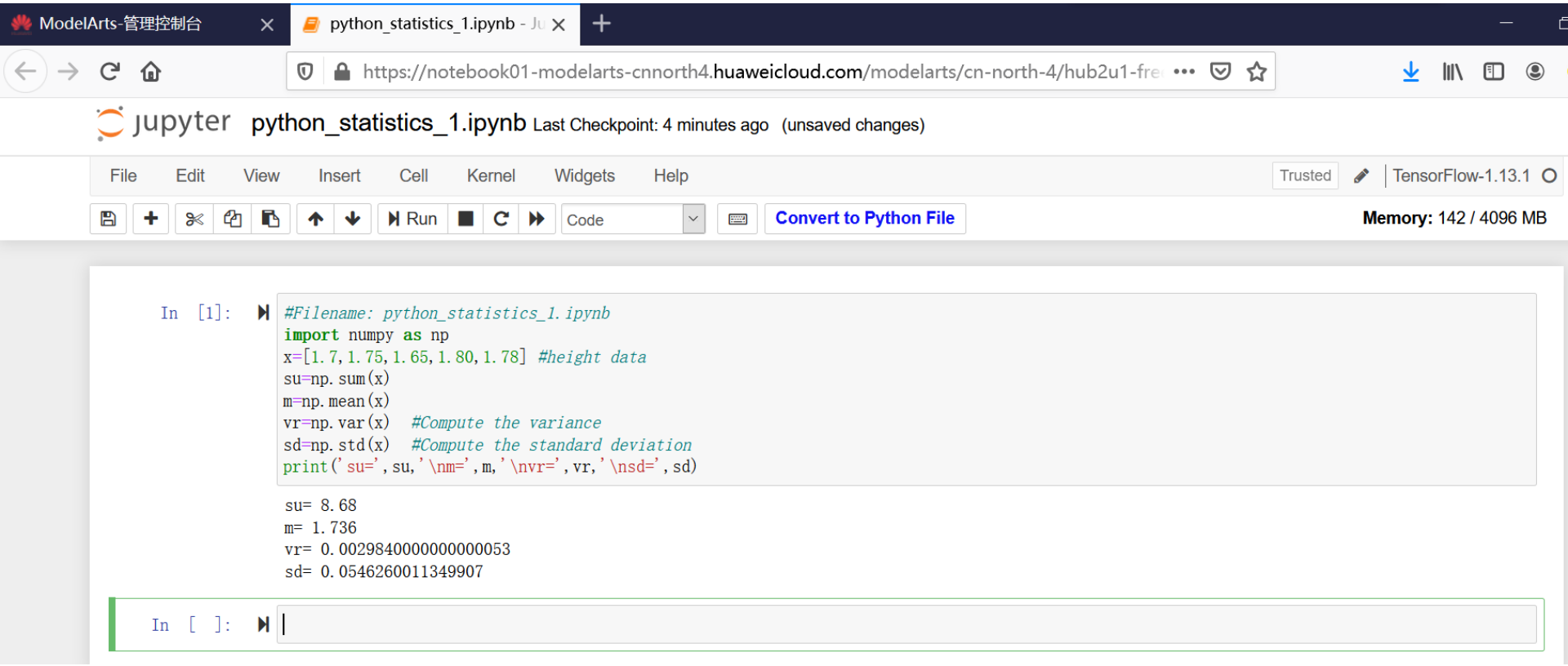
2. 云端Jupyter notebook环境

(1) 使用华为AI开发平台：ModelArts

华为一站式AI开发平台网址：https://www.huaweicloud.com/product/modelarts.html

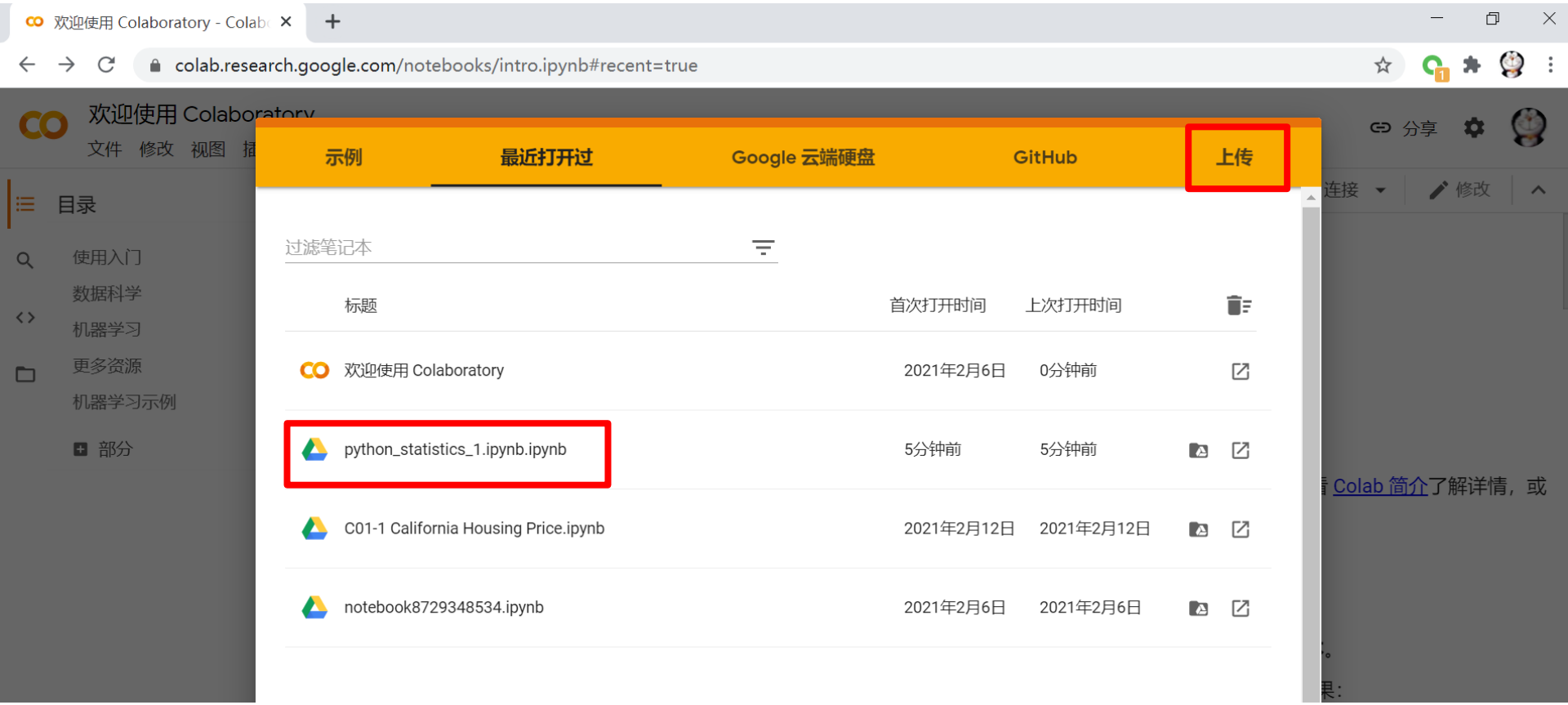
注册华为账号后，选择[AI开发平台ModelArts]，单击[进入控制台]显示帐号登录页面，帐号登录后，单击[开发环境]下的[Notebook]，可进入Jupyter notebook/JupyterLab。

程序运行情况：



3. 使用Google AI开发平台: Google Colab

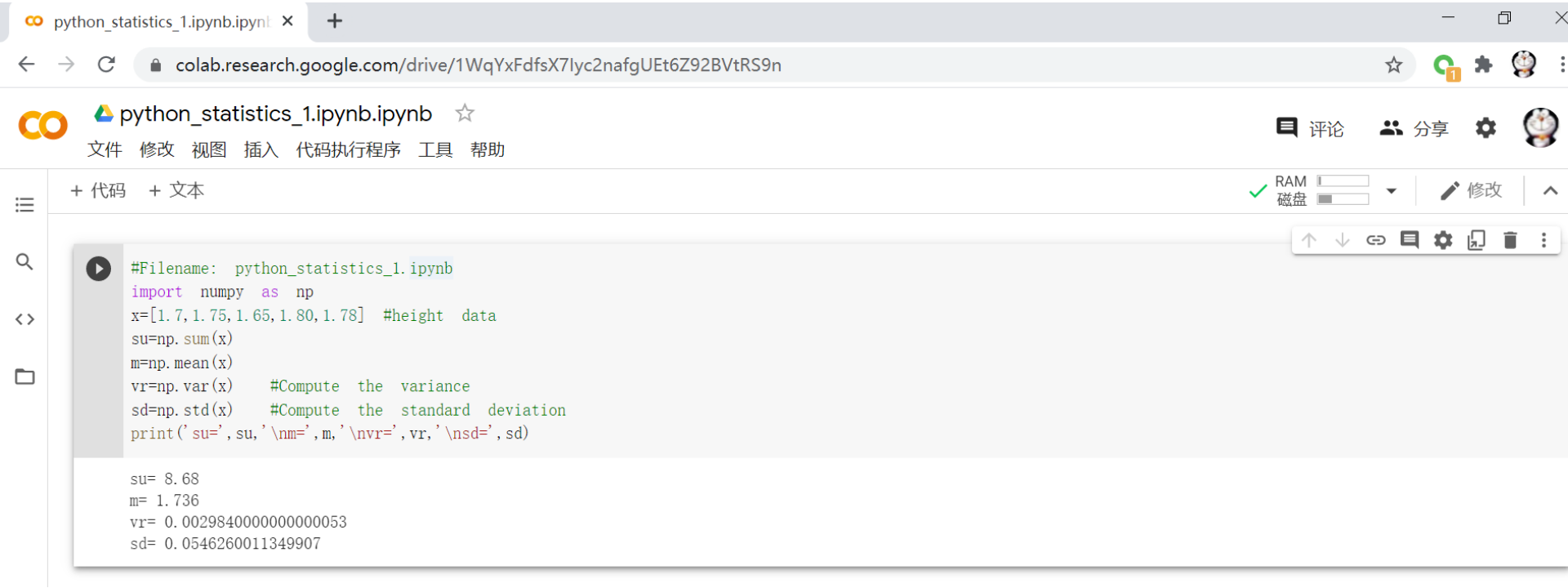
Google Colab使用前提：能上外网



红色框：上传文件python_statistics_1.ipynb文件到Google Colab。

云端Jupyter notebook环境 (使用Google AI开发平台: Google Colab)

程序运行情况：



2.3 *K*近邻法

*K*近邻法是一种简单而基本的机器学习方法，可用于求解分类和回归问题。*K*近邻法于1968年由Cover和Hart提出。

最简单、最初级的分类器是将全部的训练数据所对应的类别都记录下来，当测试样本的属性和某个训练样本的属性完全匹配时，便能对其进行分类。但是并不是所有测试样本都能找到与之完全匹配的训练样本，而且可能存在一个测试样本同时与多个训练样本相匹配，导致一个训练样本被分到多个类的问题，基于这些问题，于是产生了*K*近邻法。

1. 基本概念

- 英文名称：***K*-Nearest Neighbor Algorithm**
- 简称：**KNN**
- 中文名称：***K*近邻法**

2. 定义

K*近邻法(K-Nearest Neighbor algorithm，简称*k*NN/KNN法)**，仅从名称来看，可简单地认为：K*个最近的邻居**，当***K*=1**时，*k*NN法便成了最近邻法，即寻找最近的那个邻居。

所谓*K*近邻法，就是给定一个训练数据集，对新的输入样本(样例/实例/模式)，在训练数据集中找到与该样本最邻近的*k*个样本，这*k*个样本的多数属于某个类，就把该输入样本分类到这个类中。

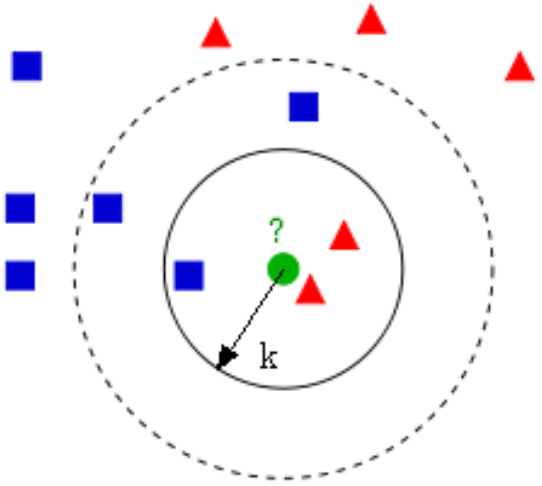
*K*近邻法中，所选择的邻居都是已经正确分类的对象。该方法在分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

3. 基本思想

- 产生训练集，使得训练集按照已有的分类标准划分成离散型数值类，或者是连续型数值类输出。
- 以训练集的分类为基础，对测试集每个样本寻找K个近邻，采用欧式距离作为样本间的相似程度的判决依据，相似度大的即为最近邻。一般近邻可以选择1个或者多个。
- 当类为连续型数值时，测试样本的最终输出为近邻的平均值；当类为离散型数值时，测试样本最终归为近邻类中个数最多的那一类。

4. *K*近邻法直观理解

见右图，图中的有两种类型的样本数据，一类（ω_1）是蓝色的正方形，另一类（ω_2）是红色的三角形。绿色的圆形是待分类样本。



问题：给这个绿色的圆分类？

- 如果*k*=3，那么离绿色点最近的有2个红色三角形和1个蓝色的正方形，给这3个点投票，于是绿色的这个待分类点属于红色的三角形。
- 如果*k*=5，那么离绿色点最近的有2个红色三角形和3个蓝色的正方形，给这5个点投票，于是绿色的这个待分类点属于蓝色的正方形。

5. K近邻法三个基本要素

- k值的选择
- 距离度量方法
- 分类决策规则/回归标签值计算方法

- ***k*值的选择**

***k*值的选择会对算法的结果产生重大影响。*k*值较小意味着用较小的邻域中的训练样本进行预测，只有与输入样本较近的训练样本才会对预测结果起作用，但容易发生过拟合 (over fitting)；如果*k*值较大，优点是能减少“学习”的估计误差 (estimation error)，但缺点是“学习”的近似误差 (approximation error)会增大，这时与输入样本较远的训练样本也会对预测起作用，使预测发生错误。实际应用中，*k*值一般选择较小的数值。具体应用中，*k*值的选择通常需要通过大量的实验来确定。

- 距离度量

距离的度量可采用欧氏距离，也可采用更一般的 L_p 距离，L_p 距离(或**Minkowski distance**)定义为：

$$L_p\left(\mathbf{x}_i,\mathbf{x}_j\right)=\left(\sum_{k=1}^{n}\left|x_i^{(k)}-x_j^{(k)}\right|^p\right)^{\frac{1}{p}}$$

其中 $\mathbf{x}_i\in R^n,\mathbf{x}_j\in R^n$，$L_\infty$ (各个坐标距离的最大值)定义为：

$$L_\infty\left(\mathbf{x}_i,\mathbf{x}_j\right)=\max_k\left|\mathbf{x}_i^{(k)}-\mathbf{x}_j^{(k)}\right|$$

当p=2时，即为欧氏距离。

- 分类决策规则/回归标签值计算方法

  分类决策规则往往是多数表决，即待识别样本由K个最临近的训练样本中的多数类来决定待识别样本的类别。

  回归标签值计算常用求均值函数、线性回归模型和局部加权线性回归模型。

6. k近邻分类算法描述

输入：训练数据集

$$T=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),...,(\mathbf{x}_N,y_N)\}$$

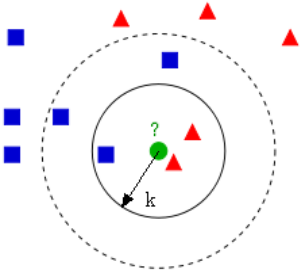
其中，$\mathbf{x}_i\in R^n$，$y_i\in\{\omega_1,\omega_2,...,\omega_M\}$为模式的类别，$i=1,2,...,N$；$\mathbf{x}$为待识别的模式.

输出：模式所属的类别y.

(1)根据给定的距离度量，在训练集T中找出与$\mathbf{x}$最近邻的k个点，涵盖这k个点的邻域记作$N_k(\mathbf{x})$；

(2)在$N_k(\mathbf{x})$中根据分类决策规则(如多数表决)确定$\mathbf{x}$的类别y：

$$y=\arg\max_{\omega_j}\sum_{\mathbf{x}_i\in N_k(\mathbf{x})}I(y_i=\omega_j),\quad i=1,2,...,N;\ j=1,2,...,M$$



其中，I为指示(Indicator)函数，即当$y_i=\omega_j$时I为1，否则I为0.

k近邻法的特殊情况是$k=1$的情形，称为最近邻法. 对于输入的模式向量(实例)$\mathbf{x}$，最近邻法将训练数据集中与$\mathbf{x}$最近邻点的类作为$\mathbf{x}$的类.

6. K近邻回归算法描述(*)

应用K近邻法求解回归问题需先指定三要素：样本间距离度量方法$d(\cdot)$、邻居样本个数k 、根据k个邻居样本计算标签值方法$v(\cdot)$。

设训练样本集T包含m个样本，每个样本$s_i=\{\mathbf{x}_i, y_i\}$包括一个实例$\mathbf{x}_i$和一个实数标签值$y_i$，测试样本记为$\mathbf{x}$。

K近邻法用于回归分为两步：

(1)根据$d(\cdot)$，从训练样本集T中找出k个距离$\mathbf{x}$最近的样本，即得到$\mathbf{x}$的邻域$N_k(\mathbf{x})$；

(2)计算$v\big(N_k(\mathbf{x})\big)$得到$\mathbf{x}$的标签值。

$d(\cdot)$常用欧氏距离。$v(\cdot)$常用求均值函数、线性回归模型和局部加权线性回归模型。

k值的大小对算法有重大影响。过小的k值，结果对噪声更敏感，容易发生过拟合。过大的k值，较远的节点也会影响结果，近似误差(approximation error)会增大。

➢ 例：采用k近邻法进行产品质量判断。

某纸巾厂需要判断生产纸巾品质的好坏，纸巾品质好坏由两个特征判断，一个是“酸腐蚀的时间”，另一个是“能承受的压强”，已知品质情况的样本集如右表所示。

x_1 耐酸时间（s）	x_2 压强(kg/m²)	品质
7	7	坏
7	4	坏
3	4	好
1	4	好

若生产出一批毛巾，设这批毛巾的特征向量为 $\mathbf{x}=(3, 7)^{\mathrm{T}}$，试用$k$近邻法判断该批毛巾品质如何？

假设k=3，k一般选择奇数，这样可确保不会有平票，下表是计算待决策模式/样本$(3,7)^T$到所有样本点的距离。

耐酸时间（s）	压强 (kg/m^2)	计算到 $(3, 7)^T$的距离	品质	是否包含在最近的3个邻居内?
7	7	$(7-3)^2+(7-7)^2=16$	坏	是
7	4	$(7-3)^2+(4-7)^2=25$	坏	否
3	4	$(3-3)^2+(4-7)^2=9$	好	是
1	4	$(1-3)^2+(4-7)^2=13$	好	是

耐酸时间（s）	压强 (kg/m²)	计算到 $(3, 7)^T$的距离	品质	是否包含在最近的3个邻居内？
7	7	$(7-3)^2+(7-7)^2=16$	坏	是
7	4	$(7-3)^2+(4-7)^2=25$	坏	否
3	4	$(3-3)^2+(4-7)^2=9$	好	是
1	4	$(1-3)^2+(4-7)^2=13$	好	是

因此，经最后投票表决，好的有2票，坏的有1票，决策出待测试的批次纸巾样本$(3,7)^T$是合格品，从而推断出该批毛巾为合格品。

7. *k*近邻法Python编程

- 例：蠓的二分类问题*k*近邻法Python编程实现。

蠓的分类问题：

两种蠓Apf和Af已由生物学家根据它们的触角和翼长加以区分(Apf是会传播疾病的害虫，Af是能传播花粉的益虫)，两个矩阵中分别给出了**6**只**Apf** 和**9**只**Af**蠓的触角长(对应于矩阵的第**1**列)和翼长(对应于矩阵的第**2**列)的数据(See the next slide)。根据触角长和翼长这两个特征来识别一个样本是Af还是Apf。

有三个待识别的模式/样本(待预测的样本)，它们分别是$(1.24,1.80)^{T}$，$(1.28,1.84)^{T}$，$(1.40,2.04)^{T}$，试问这三个样本属于哪一种蠓。

假设：

(1)两种群Apf 和Af的两个特征的期望值、标准差、相关系数与由矩阵数据给出的样本统计量一致；

(2)两种群Apf 和Af的两个特征服从二元正态分布；

(3)所给样本数据无误差。

问：

有三个待识别的模式/样本(待预测的样本)，它们分别是 $(1.24, 1.80)^T$, $(1.28, 1.84)^T$, $(1.40, 2.04)^T$，试问这三个样本属于哪一种蠓。

蠓Apf

触角长	翼长
1.14	1.78
1.18	1.96
1.20	1.86
1.26	2.00
1.30	2.00
1.28	1.96

蠓Af

触角长	翼长
1.24	1.72
1.36	1.74
1.38	1.64
1.38	1.82
1.38	1.90
1.40	1.70
1.48	1.82
1.54	2.08
1.56	1.78

sklearn机器学习库中实现K近邻分类的类是neighbors包中的KNeighborsClassifier;实现K近邻回归的类是neighbors包中的KNeighborsRegressor。

Anaconda Jupyter Notebook环境Python程序运行结果截图：

In [1]:

```
#蠓的kNN二分类器
#Filename: knn_example.ipynb
#Import Library
import numpy as np
import sklearn as sk
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
X=np.array([[1.14, 1.78], [1.18, 1.96], [1.20, 1.86], [1.26, 2.00], [1.30, 2.00], [1.28, 1.96],
[1.24, 1.72], [1.36, 1.74], [1.38, 1.64], [1.38, 1.82], [1.38, 1.90], [1.40, 1.70], [1.48, 1.82], [1.54, 2.08], [1.56, 1.78]])
y=np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1])
#(X,y)为训练集,前6个为Apf类 (类标签:0), 后9个样本为Af类 (类标签:1)
x_test=np.array([[1.24, 1.8], [1.28, 1.84], [1.4, 2.04]])    #待测试的三个样本
y_test=([0, 0, 0])    #待测试的三个样本类标签
# Create KNeighbors classifier object model
knn=KNeighborsClassifier(n_neighbors=3)
# 实验参数：可实验选择不同的k值(这里k取1合适, 尝试k取3测试样本会分错)
# default value for n_neighbors is 5
# Train the model using the training sets and check score
knn.fit(X, y)
#Predict Output
y_pred= knn.predict(X)
training_acc=100-np.mean(np.abs(y_pred-y))*100
print('KNN分类器训练准确率: {:.2f}%'.format(training_acc))
#Predict Output
y_pred= knn.predict(x_test)
print('KNN分类结果:',y_pred)
#Predict accuracy
from sklearn.metrics import accuracy_score
testing_acc=accuracy_score(y_test,y_pred)*100
print('KNN分类器测试准确率: {:.2f}%'.format(testing_acc))
```

```
KNN分类器训练准确率: 100.00%
KNN分类结果: [0 1 0]
KNN分类器测试准确率: 66.67%
```

Python程序清单：

```
#蠓的kNN二分类器
#Filename: knn_example.ipynb
#Import Library
import numpy as np
import sklearn as sk
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
X=np.array([[1.14,1.78],[1.18,1.96],[1.20,1.86],[1.26,2.00],[1.30,2.00],[1.28,1.96],
[1.24,1.72],[1.36,1.74],[1.38,1.64],[1.38,1.82],[1.38,1.90],[1.40,1.70],[1.48,1.82],[1.54,2.08],[
1.56,1.78]])
y=np.array([0,0,0,0,0,0,1,1,1,1,1,1,1,1,1])
#(X,y)为训练集,前6个为Apf类 (类标签:0)，后9个样本为Af类 (类标签:1)
x_test=np.array([[1.24,1.8],[1.28,1.84],[1.4,2.04]])   #待测试的三个样本
y_test=([0,0,0])   #待测试的三个样本类标签
# Create KNeighbors classifier object model
knn=KNeighborsClassifier(n_neighbors=1)
# 实验参数：可实验选择不同的k值(这里k取1合适, 尝试k取3测试样本会分错)
# default value for n_neighbors is 5
# Train the model using the training sets and check score
knn.fit(X, y)
```

```
#Predict Output
y_pred= knn.predict(X)
training_acc=100-np.mean(np.abs(y_pred-y))*100
print('KNN分类器训练准确率: {:.2f}%'.format(training_acc))
#Predict Output
y_pred= knn.predict(x_test)
print('KNN分类结果:',y_pred)
#Predict accuracy
from sklearn.metrics import accuracy_score
testing_acc=accuracy_score(y_test,y_pred)*100
print('KNN分类器测试准确率: {:.2f}%'.format(testing_acc))
```

输出结果:

```
KNN分类器训练准确率: 100.00%
KNN分类结果: [0 0 0]
KNN分类器测试准确率: 100.00%
```

●维度灾难/维数诅咒(*: 了解)

由于维度灾难(curse of dimensionality)的原因，使得KNN算法易过于拟合。维度灾难是这样一种现象：对于样本数量大小稳定的训练数据集，随着其特征数量的增加，样本中有具体值的特征数量变量极其稀疏(大多数特征的取值为空)。直观地说，可以认为即使是最近的邻居，它们在高维空间的实际距离也是非常远的，因此难以给出一个合适的类别标签判定。

2.4 线性回归

*多元线性回归(Multivariate linear regression)

给定训练集$T=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),...,(\mathbf{x}_m,y_m)\}$,

其中:

$\mathbf{x}_i=(x_{i1},x_{i2},...,x_{in},1)^T\ (i=1,2,...,m)$,

设给定m个样本/模式, 各样本有n个特征, $y_i\in R$.

$\mathbf{w}=(w_1,w_2,...,w_n,w_{n+1})^T$

我们试图从训练集$T=\{(\mathbf{x}_i,y_i)\}_{i=1}^m$学得

$f(\mathbf{x}_i)=\mathbf{w}^T\mathbf{x}_i$, 使得$f(\mathbf{x}_i)$； y_i的$\mathbf{w}$.

w参数学习/训练出来 (w参数学习背后的统计学原理是对w做最小二乘估计) 后，所确定的f(x)模型也叫预测模型。

若用f(x)预测的是离散值，此类学习任务称为“分类(Classification)”；若用f(x)预测的是连续值，此类学习任务为“回归(regression)”。由于这里f(x)是线性的，故称为多元线性回归(《概率论与数理统计》相关书籍中一般都有一元/多元线性回归分析方面的介绍)。

➢ 一元线性回归举例：

表　不同披萨的直径和价格训练数据

i: 训练样本编号	x_i: 直径(寸)	y_i: 价格(元)
1	6	7
2	8	9
3	10	13
4	14	17.5
5	18	18

1. 用Matplotlib对训练数据作可视化

训练数据可视化代码：

```
# "np" and "plt" are common aliases for NumPy and Matplotlib, respectively.
import numpy as np
import matplotlib.pyplot as plt
# X represents the features of our training data, the diameters of the pizzas.
# A scikit-learn convention is to name the matrix of feature vectors X.
# Uppercase letters indicate matrices, and lowercase letters indicate vectors.
X = np.array([6, 8, 10, 14, 18]).reshape(-1, 1)
# y is a vector representing the prices of the pizzas.
y = [7, 9, 13, 17.5, 18]
```

```
plt.rcParams["font.family"]="SimHei"
plt.figure()
plt.title('披萨价格直径散点图')
plt.xlabel('直径(单位：寸)')
plt.ylabel('价格(单位：元)')
plt.plot(X, y, 'r.')
plt.axis([0, 25, 0, 25])
plt.grid(True) plt.show()
```

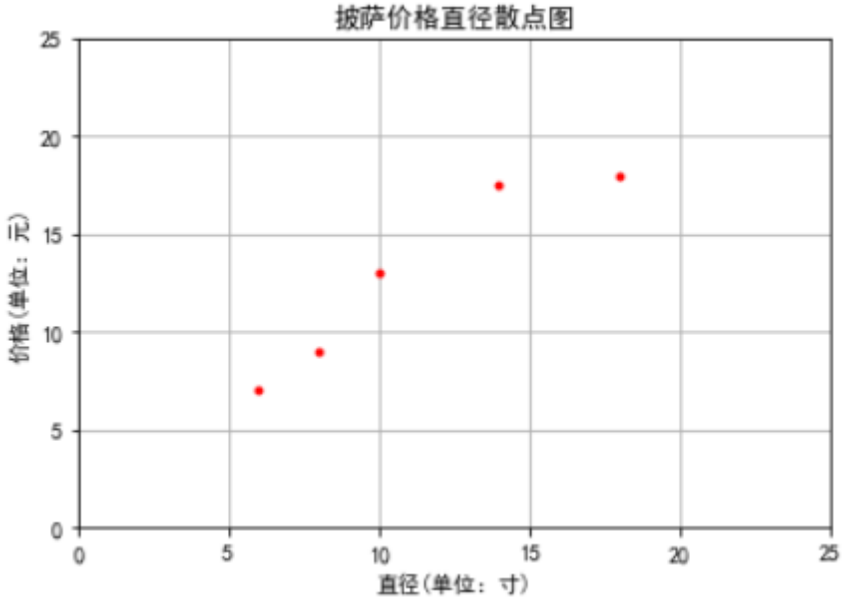


图　训练数据可视化

提示：Matplotlib包括两个模块
(1)绘图API：pyplot，在Jupyter中通常用于可视化，这里采用的是matplotlib.pyplot 。
(2)集成库：pylab，是Matplotlib和NumPy、SciPy的集成库。

从训练数据的可视化散点图可知披萨的直径和价格之间存在正相关关系，随着披萨直径的增加，其价格通常也会上涨。

2. 建立一元线性回归模型

sklearn.linear_model包中提供了线性回归模型：LinearRegression。

一元线性回归程序：

```
import numpy as np
from sklearn.linear_model import LinearRegression
X = np.array([6, 8, 10, 14, 18]).reshape(-1, 1)
y = [7, 9, 13, 17.5, 18]
# Create an instance of the estimator, LinearRegression
model = LinearRegression()
# Fit the model on the training data
model.fit(X, y)
#Equation coefficient and Intercept
print('Coefficient: ',  model.coef_)
print('Intercept: ', model.intercept_)
# Predict the price of a pizza with a diameter that has never been seen before
test_pizza = np.array([[12]])
predicted_price = model.predict(test_pizza)[0]
print('一个12寸披萨价格预测价格是: %.2f元'% predicted_price)
```

一元线性回归程序及输出结果截图：

In [1]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
X = np.array([6, 8, 10, 14, 18]).reshape(-1, 1)
y = [7, 9, 13, 17.5, 18]
# Create an instance of the estimator, LinearRegression
model = LinearRegression()
# Fit the model on the training data
model.fit(X, y)
#Equation coefficient and Intercept
print('Coefficient: ',model.coef_)
print('Intercept: ', model.intercept_)
# Predict the price of a pizza with a diameter that has never been seen before
test_pizza = np.array([[12]])
predicted_price = model.predict(test_pizza)[0]
print('一个12寸披萨价格预测价格是: %.2f元' % predicted_price)
```

```
Coefficient:  [0.9762931]
Intercept:  1.965517241379315
一个12寸披萨价格预测价格是: 13.68元
```

w_1=0.976

w_0=1.966

y =H(x)=w_1x+w_0 (1)

=0.976x+1.966

LinearRegression类是一个估计器。估计器基于观测到的数据预测一个值。

在scikit-learn 中，所有的估计器都实现了fit方法和predict方法。前者用于学习模型的参数w_1、w_0，后者使用学习到的参数来预测一个新的输入样本对应的输出变量值。使用scikit-learn 能简单地对不同模型进行实验，因为所有的估计器都实现了fit方法和predict方法，尝试新的模型只需修改一行代码。

LinearRegression的fit方法学习了**公式(1)**一元线性回归模型的参数w_1、w_0。

●Python+scikit-learn机器学习的一般步骤及方法归纳

(1)业务理解:

开展机器学习与数据分析项目的前提

(2)数据读入:

如from sklearn import datasets, dataset = datasets.load_iris()

(3)数据准备:

如数据预处理，训练集、测试集的拆分

(4)数据理解:

如数据可视化

(5)算法选择及其超参数的设置:

如myModel = LinearRegression()

(6)具体模型的训练:

如myModel.fit(X_trainingSet, y_trainingSet)

(7)用模型进行预测:

如y_predSet= myModel.predict(X_testSet)

(8)模型评价:

如from sklearn.metrics import accuracy_score, training_acc=accuracy_score(y_trainingSet, y_predSet)

(9)模型的优化与应用:

用模型进行新数据预测，优化模型或选择其他算法

●最小二乘法求解线性回归模型

我们在概率统计课程中学过的线性回归求解背后的数学原理是最小二乘法。最小二乘法(Least Square Method)中的“最小二乘”的意思是最小化误差的平方和，误差是指观测数据的真实输出值和由模型拟合的因变量预测值之差。线性最小二乘法或普通最小二乘(OLS, Ordinary Lease Squares)是最小二乘法中最简单的一种情况，即模型的参数是线性的。最小二乘法是解析法。

对于每一个样本观测值$(x_i, y_i)(i=1,2,...,n)$，使用最小二乘法进行参数估计的思路是：使观测值y_i与其回归预测值$\hat{y}_i = w_0 + w_1 x_i$的离差越小越好。

定义$e_i = y_i - \hat{y}_i$为第i个残差，定义残差平方和RSS(Residual Sum of Squres)为：

$$RSS=e_1^2+e_2^2+...+e_n^2$$

或等价地定义：

$$RSS=Q(w_0, w_1)=(y_1 - w_0 - w_1 x_1)^2 + (y_2 - w_0 - w_1 x_2)^2 + ... + (y_n - w_0 - w_1 x_n)^2$$

$$=\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 \qquad (1)$$

最小二乘法的思想：寻找参数w_0和w_1的最优估计量w_0^*、w_1^*来使RSS残差平方达到最小。

即：$$Q(w_0, w_1)=\sum_{i=1}^{n}(y_i - w_0^* - w_1^* x_i)^2 = \min_{w_0, w_1}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 \qquad (2)$$

根据式(2)求最优参数估计量w_0^*和w_1^*是一个求极值问题。根据微积分中求极值定理，w_0^*和w_1^*应满足：

$$\begin{cases} \left.\dfrac{\partial Q}{\partial w_0}\right|_{w_0=w_0^*} = -2\sum_{i=1}^{n}\left(y_i - w_0^* - w_1^* x_i\right) = 0 \\ \left.\dfrac{\partial Q}{\partial w_1}\right|_{w_1=w_1^*} = -2\sum_{i=1}^{n}\left(y_i - w_0^* - w_1^* x_i\right) = 0 \end{cases} \tag{3}$$

(3)式经整理后，得到使RSS最小的参数估计量为：

$$\begin{cases} w_1^* = \dfrac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \dfrac{\mathrm{cov}(x, y)}{\mathrm{var}(x)} \\ w_0^* = \bar{y} - w_1^* \bar{x} \end{cases} \tag{4}$$

这里$\bar{y}=\dfrac{1}{n}\sum_{i=1}^{n} y_i$和$\bar{x}=\dfrac{1}{n}\sum_{i=1}^{n} x_i$是样本均值。

●多项式回归

多项式回归(Polynomial Regression)是研究一个因变量与一个或多个自变量间多项式关系的回归分析方法。多项式回归模型是非线性回归模型中的一种。由泰勒级数可知，在某点附近，如果函数n次可导，那么它可以用一个n次的多项式来近似。

进行多项式回归分析，首先要确定多项式的次数。次数的确定一般是根据经验和实验。假设确定了用一个一元n次多项式来拟合训练样本集，模型可表示为：

$$h(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n$$

那么多项式回归的任务就是估计出各w值。

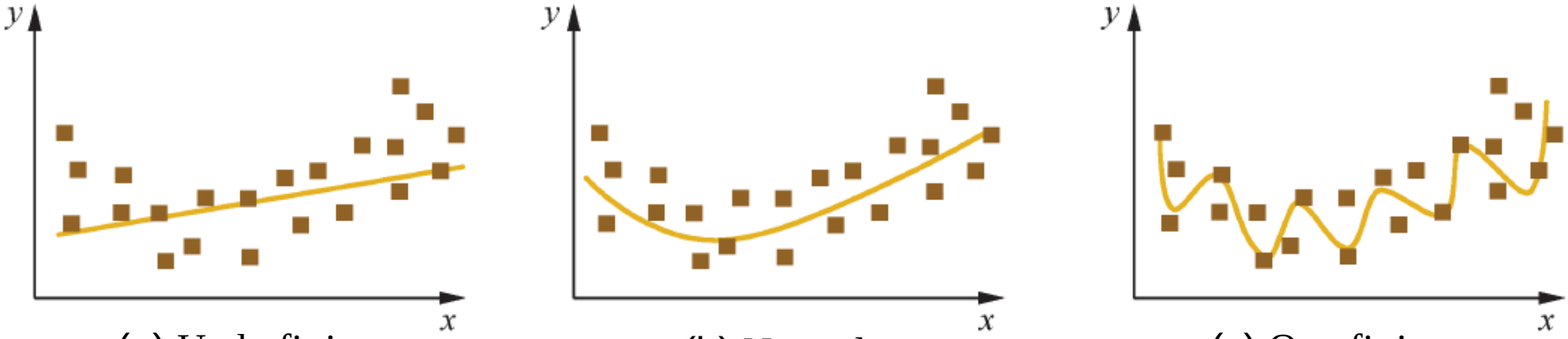
多项式回归问题可以转化为线性回归问题来求解，具体思路是将式中的每一项看作一个独立的特征(或者说生成新的特征)，令$y_1 = x, y_2 = x^2, \ldots, y_n = x^n$；那么一个一元n次多项式就变成了一个n元一次多项式，那么就可采用线性回归方法来求解。

2.5 数据模型的泛化能力、过拟合与欠拟合

模型的**泛化能力**(Generalization ability) 是指：训练集上训练出的模型对新数据/新实例 (模拟训练时未见到过的数据) 的预测与分类能力，即使这些新数据的分布与训练数据是相同的。

模型的**过拟合**(Overfitting)是指模型对于训练数据的过当拟合(不仅拟合了规律性模式，也拟合了随机性噪声)，导致模型的泛化能力下降。反映到评估指标上，就是模型在训练集上的表现很好，但在测试集和新数据上的表现较差。

模型的**欠拟合**(Underfitting)指的是模型在训练和预测时表现都不好。



(a) Underfitting　(b) Normal　(c) Overfitting

图　欠拟合与过拟合

●欠拟合、过拟合与泛化能力

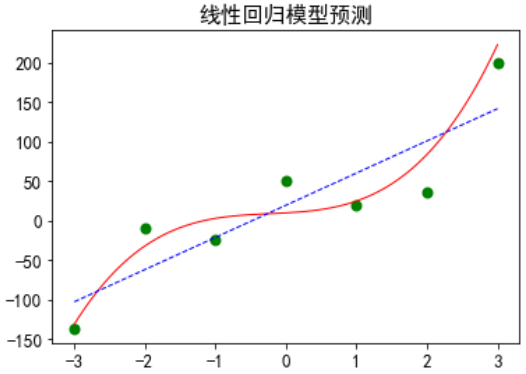
欠拟合、过拟合示例

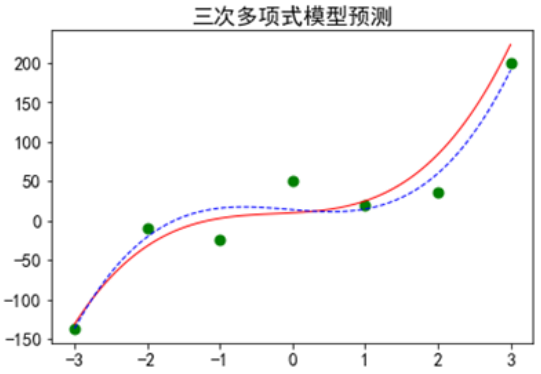
模型在训练样本上产生的误差叫**训练误差**(training error)；设训练样本容量为N_1，训练误差可按下式计算：

$$\frac{1}{N_1}\sum_{i=1}^{N1}(y_i-\hat{y}_i)^2$$

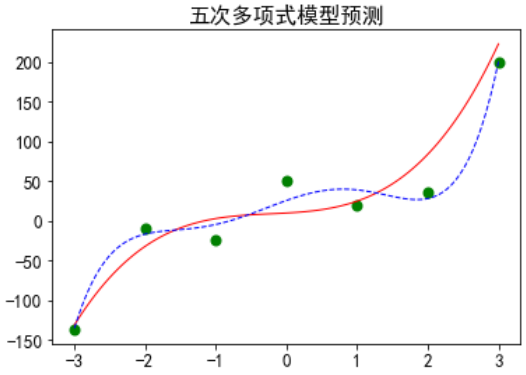
模型在测试样本上产生的误差叫**测试误差**(testing error)；设测试样本容量为N_2，测试误差可按下式计算：

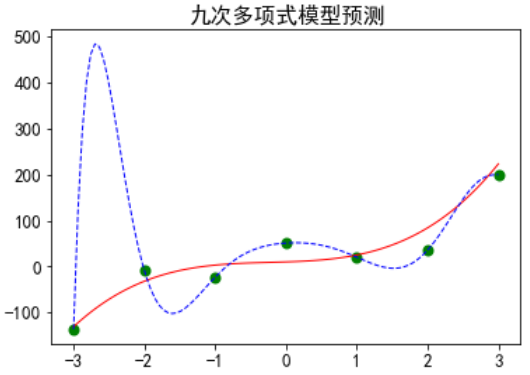
$$\frac{1}{N_2}\sum_{i=1}^{N_2}(y_i-\hat{y}_i)^2$$





a)欠拟合





b) 过拟合

	线性回归模型 $y=w_1x+w_0$	三次多项式模型 $y=w_3x^3+w_2x^2+w_1x+w_0$	五次多项式模型 ……	九次多项式模型
训练误差	2019	534	209	4
测试误差	578	247	1232	38492
和	2597	781	1441	38496

●欠拟合、过拟合与泛化能力

泛化能力与模型复杂度

衡量模型好坏的是测试误差，它标志了模型对未知新实例的预测能力，因此一般追求的是测试误差最小的那个模型。模型对新实例的预测能力即为泛化能力，模型在新实例上的误差称为泛化误差。

能够求解问题的模型往往不只一个。一般来说，只有合适复杂程度的模型才能最好地反映出训练集中蕴含的规律，取得最好的泛化能力。

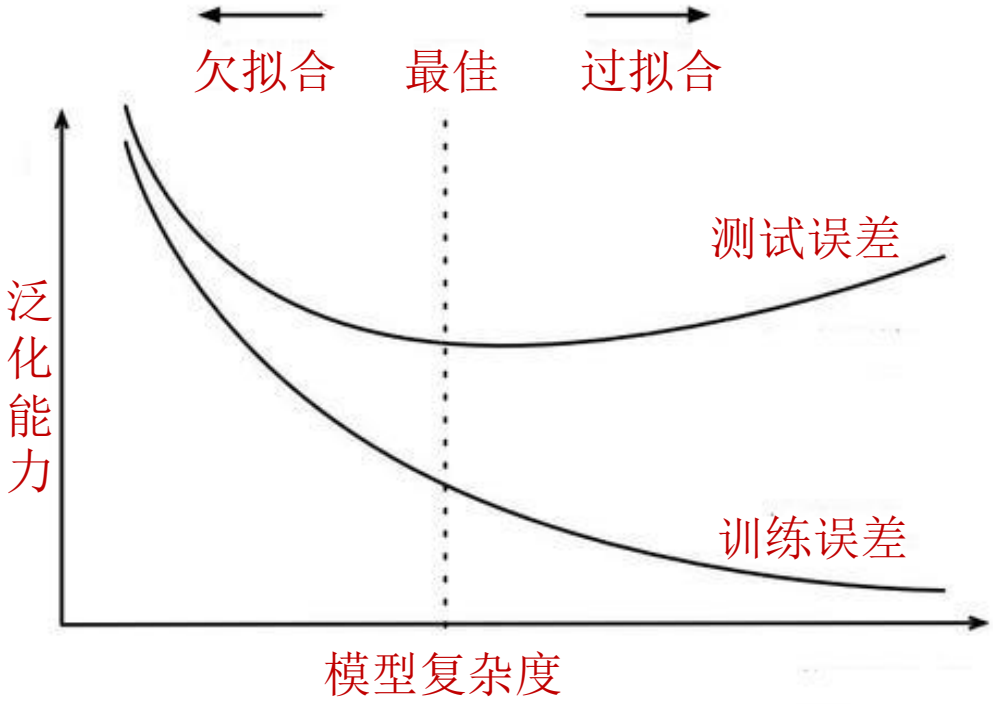


图　泛化能力与模型复杂度之间的关系

●几种降低过拟合和欠拟合风险的方法（*：了解)

降低过拟合的一些可能解决方案：

- 减少模型参数(人工手动)
- 优化模型特征(特征工程)
- 约束模型参数 (对复杂性进行惩罚的正则化)
- 增加训练数据 (提升过拟合的难度)
- 改善数据质量 (修复数据错误，填补数据缺失，清除异常数据)

降低欠拟合的一些可能解决方案：

- 选择一个带有更多参数、更强大的模型 (复杂化模型)
- 通过特征变换，为学习算法提供更好的特征集 (特征工程)
- 放松对模型的约束 (降低复杂度正则项的惩罚因子值)

本讲小结：

1.样本/模式，模式的向量及矩阵表示
2.样本间相似度量，包括距离、相关系数、夹角余弦等
3.KNN
4.线性回归及其背后的最小二乘法数学原理(最小二乘法是解析法，还有迭代法求解线性回归模型)
5.模型的泛化能力、过拟合与欠拟合
6.Python+scikit-learn机器学习的一般步骤及方法
(1)业务理解:
(2)数据读入:
(3)数据准备: 如数据预处理，训练集、测试集的拆分
(4)数据理解: 如数据可视化
(5)算法选择及其超参数的设置:
(6)具体模型的训练:
(7)用模型进行预测:
(8)模型评价:
(9)模型的优化与应用: 用模型进行新数据预测，优化模型或选择其他算法

课外作业：

1. 理论作业

(1) 简述数据模型的泛化能力与过拟合现象这两个概念；说明两者之间存在的一般性关系；(2) 借助模型的过拟合和泛化能力背后的原理，分析个人的专业学习与职业发展前景之间的关系(可以结合自己的经验)。

2. 编程单元实验作业1：

见QQ课程群提供的”MRPL课程实验1.RAR”打包文件：

(1)《机器学习与模式识别》上机实验指导书(实验1).pdf文件

(2) KNN实验程序: KNN.ipynb

(3) 高阶多项式回归的欠拟合和过拟合分析实验程序: Regression.ipynb

End of this lecture.

Thanks !