机器学习

机器学习算法课程定位、目标

定位

- 课程以算法、案例为驱动的学习，伴随浅显易懂的数学知识
- 作为人工智能领域(数据挖掘/机器学习方向)的提升课程，掌握更深更有效的解决问题技能

目标

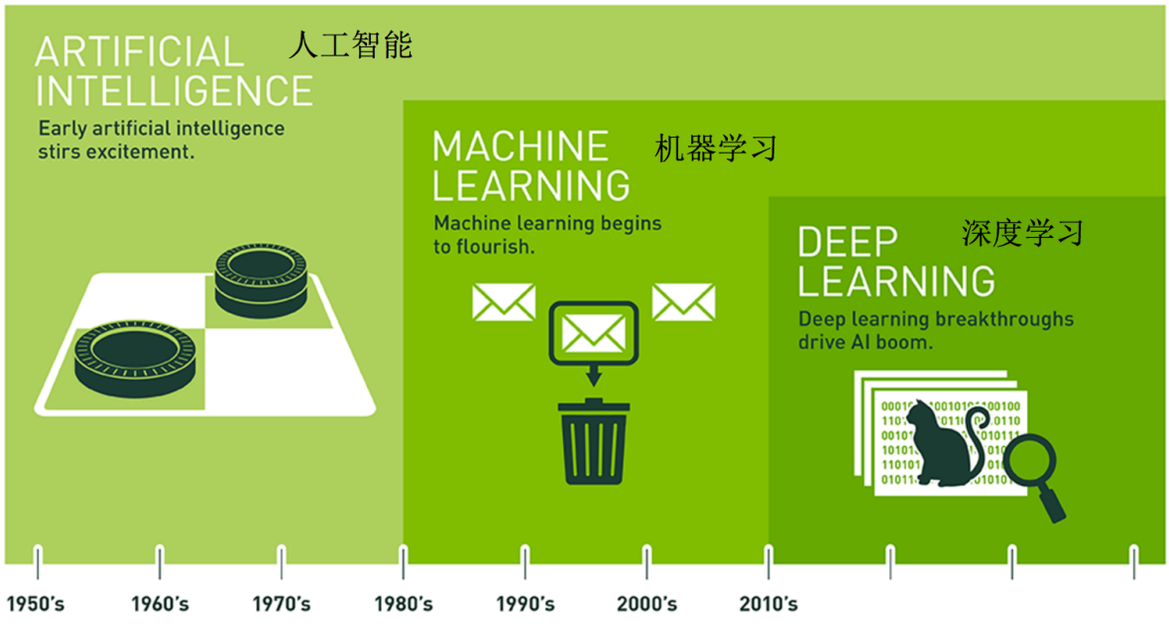
- 应用Scikit-learn实现数据集的特征工程
- 掌握机器学习常见算法原理
- 应用Scikit-learn实现机器学习算法的应用，结合场景解决实际问题

机器学习概述

> 了解机器学习定义以及应用场景
> 说明机器学习算法监督学习与无监督学习的区别
> 说明监督学习中的分类、回归特点
> 说明机器学习算法目标值的两种数据类型
> 说明机器学习(数据挖掘)的开发流程

1.1 人工智能概述

1.1.1 机器学习与人工智能、深度学习



- 机器学习和人工智能，深度学习的关系
    - 机器学习是人工智能的一个实现途径
    - 深度学习是机器学习的一个方法发展而来

- 达特茅斯会议-人工智能的起点

  1956年8月，在美国汉诺斯小镇宁静的达特茅斯学院中，

  约翰·麦卡锡（John McCarthy）

  马文·闵斯基（Marvin Minsky，人工智能与认知学专家）

  克劳德·香农（Claude Shannon，信息论的创始人）

  艾伦·纽厄尔（Allen Newell，计算机科学家）

  赫伯特·西蒙（Herbert Simon，诺贝尔经济学奖得主）等科学家正聚在一起，讨论着一个完全不食人间烟火的主题：

  用机器来模仿人类学习以及其他方面的智能。

  会议足足开了两个月的时间，虽然大家没有达成普遍的共识，但是却为会议讨论的内容起了一个名字：

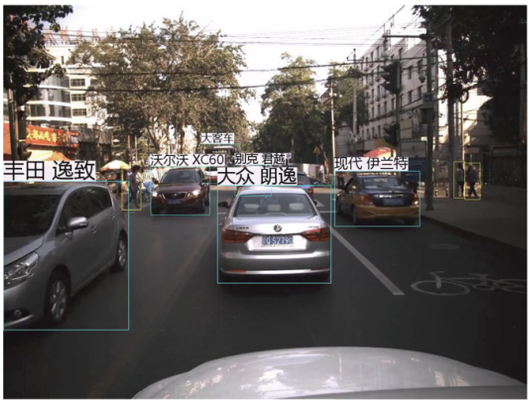
  人工智能

  因此，1956年也就成为了人工智能元年。

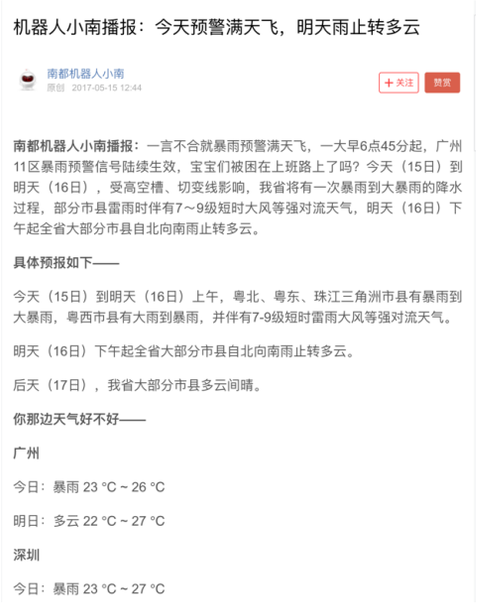
1.1.2 机器学习、深度学习能做些什么

机器学习的应用场景非常多，可以说渗透到了各个行业领域当中。医疗、航空、教育、物流、电商等等领域的各种场景。

- 用在挖掘、预测领域：
  - 应用场景：店铺销量预测、量化投资、广告推荐、企业客户分类、SQL语句安全检测分类...
- 用在图像领域：
  - 应用场景：街道交通标志检测、人脸识别等等

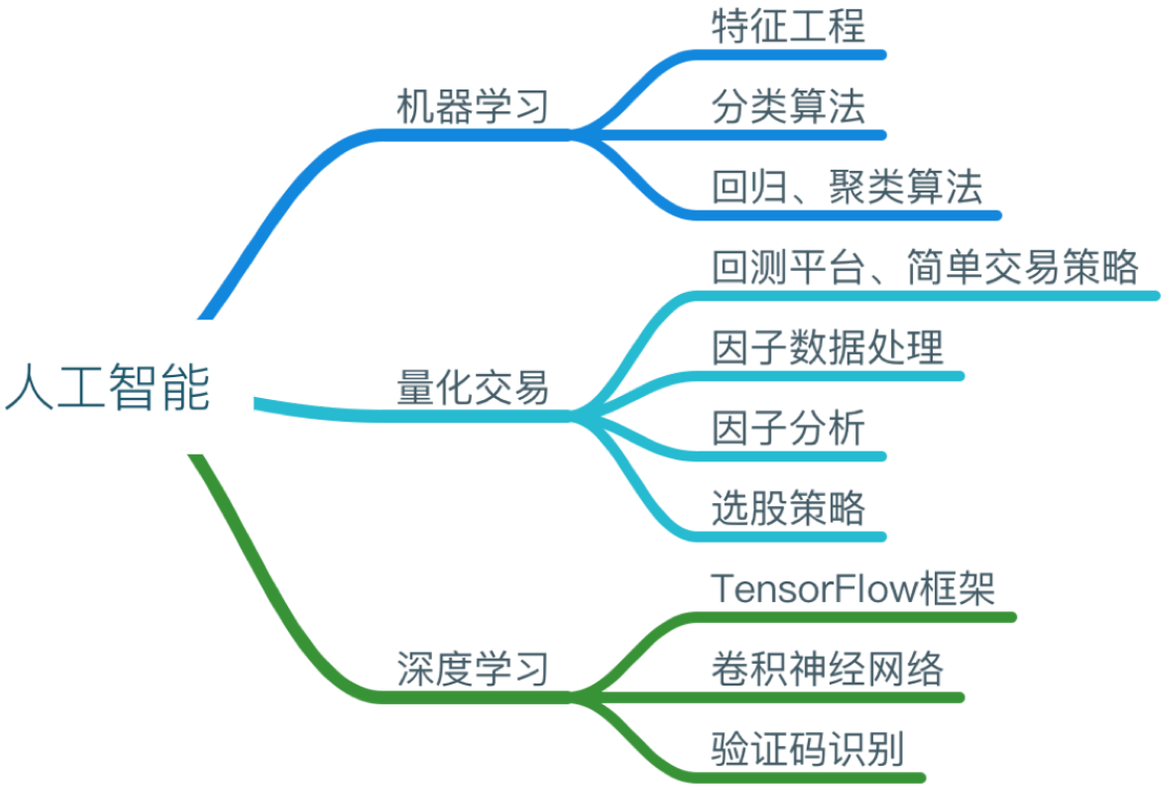


- 用在自然语言处理领域：
  - 应用场景：文本分类、情感分析、自动聊天、文本检测等等
    - 南方都市报的“小南”，广州日报的“阿同”机器人



当前重要的是掌握一些机器学习算法等技巧，从某个业务领域切入解决问题。

1.1.3 人工智能阶段课程安排

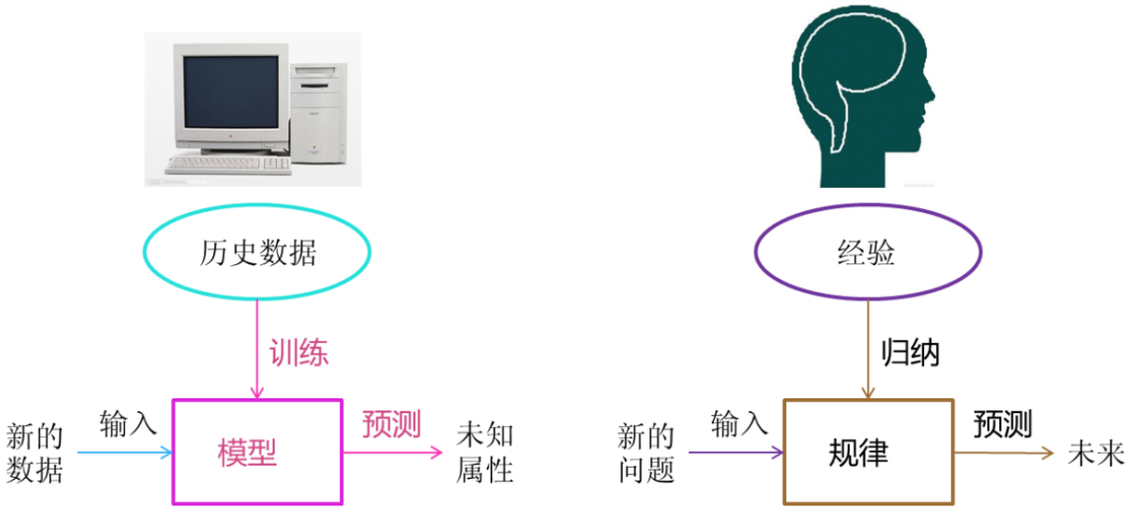


1.2 什么是机器学习

1.2.1 定义

机器学习是从**数据**中**自动分析获得模型**，并利用**模型**对未知数据进行预测。

1.2.2 解释



- 我们人从大量的日常经验中归纳规律，当面临新的问题的时候，就可以利用以往总结的规律去分析现实状况，采取最佳策略。

猫、狗？

- 从数据（大量的猫和狗的图片）中自动分析获得模型（辨别猫和狗的规律），从而使机器拥有识别猫和狗的能力。

房屋价格？

- 从数据（房屋的各种信息）中自动分析获得模型（判断房屋价格的规律），从而使机器拥有预测房屋价格的能力。

从历史数据当中获得规律？这些历史数据是怎么的格式？

1.2.3 数据集构成

- 结构：特征值+目标值

	房子面积	房子位置	房子楼层	房子朝向	目标值
数据1	80	9	3	0	80
数据2	100	9	5	1	120
数据3	80	10	3	0	100

> 注：
>
> - 对于每一行数据我们可以称之为**样本**。
> - 有些数据集可以没有目标值：

-

1.3 机器学习算法分类

学习目标

- 目标
  - 说明机器学习算法监督学习与无监督学习的区别
  - 说明监督学习中的分类、回归特点
- 应用
  - 无

分析1.2中的例子：

猫、狗？

- 特征值：猫/狗的图片；目标值：猫/狗-类别
  - 分类问题

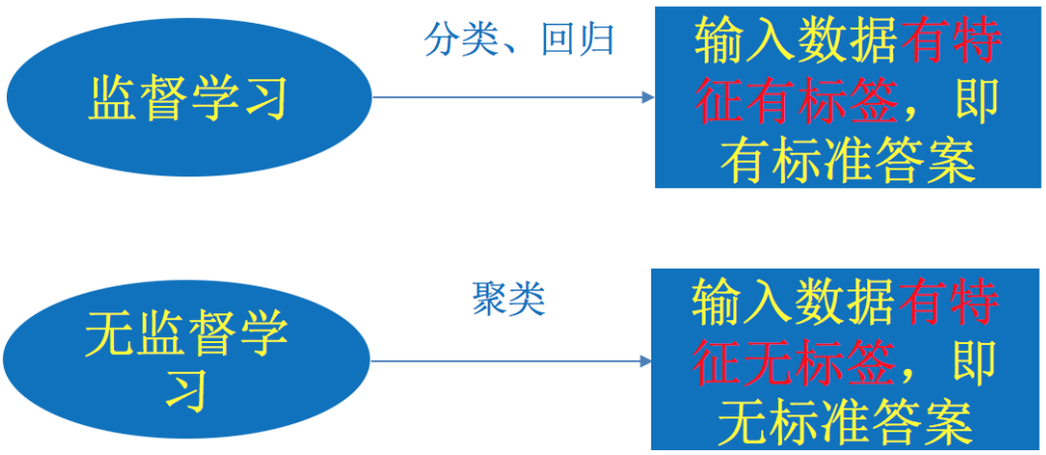
房屋价格？

- 特征值：房屋的各个属性信息；目标值：房屋价格-连续型数据
  - 回归问题

-

- 特征值：人物的各个属性信息；目标值：无
    - 无监督学习

1.3.1 总结



1.3.2 练习

说一下它们具体问题类别：

1、预测明天的气温是多少度？

2、预测明天是阴、晴还是雨?

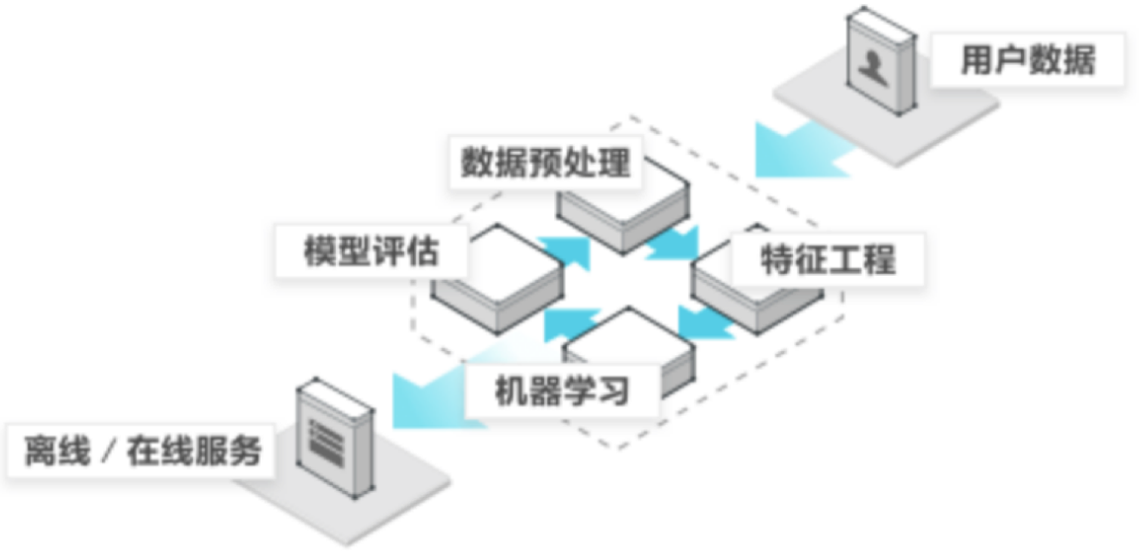
3、人脸年龄预测?

4、人脸识别?

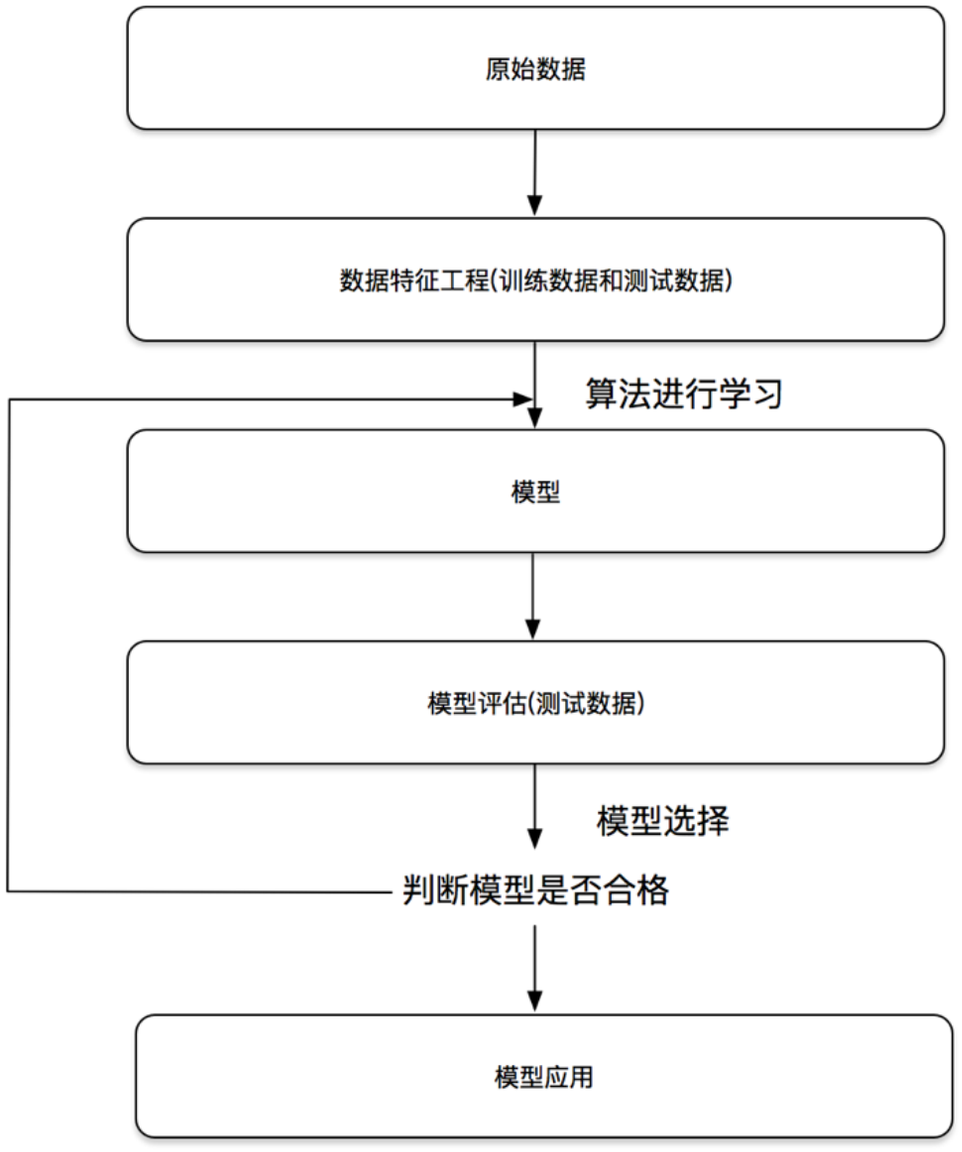
1.3.3 机器学习算法分类

- 监督学习(supervised learning)（预测）
  - 定义：输入数据是由输入特征值和目标值所组成。函数的输出可以是一个连续的值(称为回归），或是输出是有限个离散值（称作分类）。
  - **分类 k-近邻算法、贝叶斯分类、决策树与随机森林、逻辑回归、神经网络**
  - **回归 线性回归、岭回归**
- 无监督学习(unsupervised learning)
  - 定义：输入数据是由输入特征值所组成。
  - **聚类 k-means**

1.4 机器学习开发流程



- 流程图：



1.5 学习框架和资料介绍

需明确几点问题：

（1）**算法**是核心，**数据**与**计算**是基础

（2）找准定位

大部分复杂模型的算法设计都是算法工程师在做，而我们

- 分析很多的数据
- 分析具体的业务
- 应用常见的算法
- 特征工程、调参数、优化

统计学习方法

作者: 李航
出版社: 清华大学出版社
出版年: 2012-3
页数: 235
定价: 38.00元
ISBN: 9787302275954

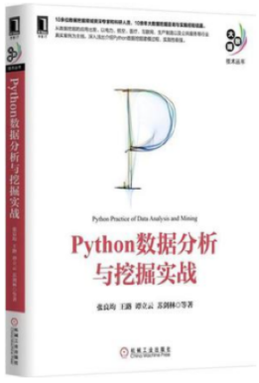
- 注意：参考书比较晦涩难懂，不建议去直接读

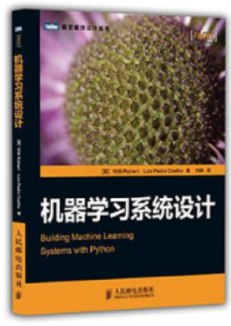
- 我们应该怎么做?
- 学会分析问题，使用机器学习算法的目的，想要算法完成何种任务
- 掌握算法基本思想，学会对问题用相应的算法解决
- 学会利用库或者框架解决问题

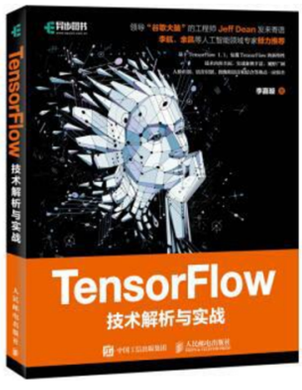
当前重要的是掌握一些机器学习算法等技巧，从某个业务领域切入解决问题。

1.5.1 机器学习库与框架

1.5.2 书籍资料





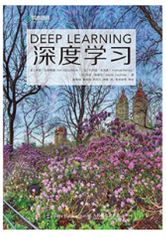


1.5.3 提深内功（但不是必须）

统计学习方法

作者: 李航
出版社: 清华大学出版社
出版年: 2012-3
页数: 235
定价: 38.00元
ISBN: 9787302275954

深度学习



作者: [美] 伊恩·古德费洛 / [加] 约书亚·本吉奥 / [加] 亚伦·库维尔
出版社: 人民邮电出版社
副标题: 又名“花书”
原作名: Deep Learning: Adaptive Computation and Machine Learning series
译者: 赵申剑 / 黎彧君 / 符天凡 / 李凯
出版年: 2017-7-1
页数: 500
定价: 168
装帧: 平装
ISBN: 9787115461476

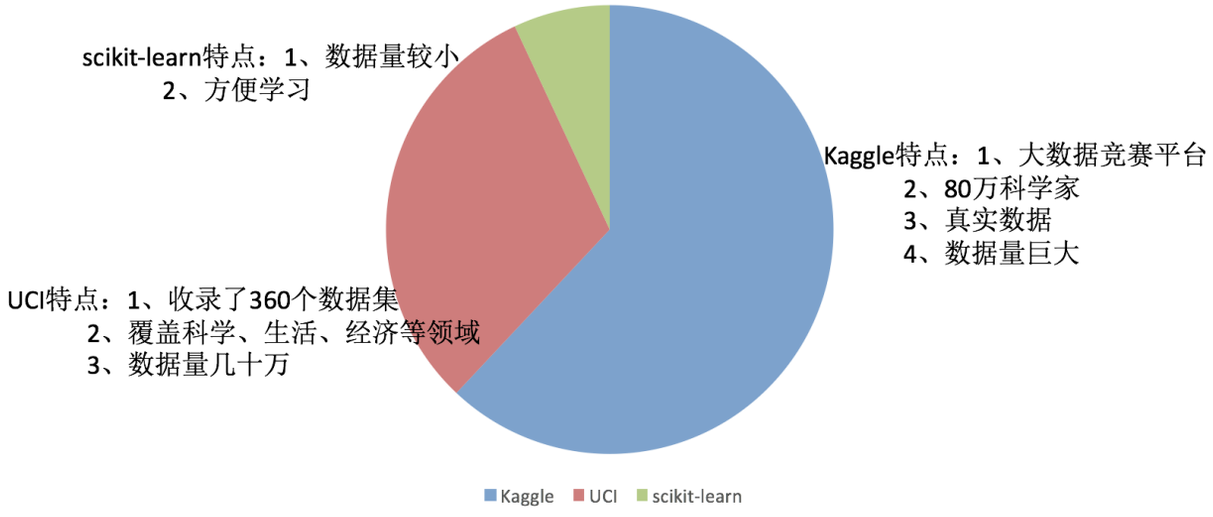
特征工程

了解特征工程在机器学习当中的重要性
应用sklearn实现特征预处理
应用sklearn实现特征抽取
应用sklearn实现特征选择
应用PCA实现特征的降维

2.1 数据集

- 目标
  - 知道数据集的分为训练集和测试集
  - 会使用sklearn的数据集
- 应用
  - 无

2.1.1 可用数据集



Kaggle网址：https://www.kaggle.com/datasets

UCI数据集网址： http://archive.ics.uci.edu/ml/

scikit-learn网址：http://scikit-learn.org/stable/datasets/index.html#datasets

1 Scikit-learn工具介绍

- Python语言的机器学习工具
- Scikit-learn包括许多知名的机器学习算法的实现
- Scikit-learn文档完善，容易上手，丰富的API
- 目前稳定版本0.19.1

2 安装

```
pip3 install Scikit-learn==0.19.1
```

安装好之后可以通过以下命令查看是否安装成功

```
import sklearn
```

- 注：安装scikit-learn需要Numpy, Scipy等库

3 Scikit-learn包含的内容

- 分类、聚类、回归
- 特征工程
- 模型选择、调优

2.1.2 sklearn数据集

1 scikit-learn数据集API介绍

- sklearn.datasets
  - 加载获取流行数据集
  - datasets.load_*()
    - 获取小规模数据集，数据包含在datasets里
  - datasets.fetch_*(data_home=None)
    - 获取大规模数据集，需要从网络上下载，函数的第一个参数是data_home，表示数据集下载的目录,默认是 ~/scikit_learn_data/

2 sklearn小数据集

- sklearn.datasets.load_iris()

  加载并返回鸢尾花数据集

名称	数量
类别	3
特征	4
样本数量	150
每个类别数量	50

- sklearn.datasets.load_boston()

  加载并返回波士顿房价数据集

名称	数量
目标类别	5-50
特征	13
样本数量	506

3 sklearn大数据集

- sklearn.datasets.fetch_20newsgroups(data_home=None,subset='train')
  - subset：'train'或者'test'，'all'，可选，选择要加载的数据集。
  - 训练集的"训练"，测试集的"测试"，两者的"全部"

4 sklearn数据集的使用

- 以鸢尾花数据集为例：

鸢尾花数据集

- 特征值–4个：花瓣、花萼的长度、宽度
- 目标值–3个：setosa, vericolor, virginica

该虹膜数据集包含150行数据，包括来自每个的三个相关鸢尾种类50个样品： *山鸢尾*，*虹膜锦葵*，和*变色鸢尾*。



从左到右， *Iris setosa*（由 Radomil，CC BY-SA 3.0）， *Iris versicolor*（由 Dlanglois，CC BY-SA 3.0）和*Iris virginica*（由Frank Mayfield，CC BY-SA 2.0））。

sklearn数据集返回值介绍

- load

  和fetch

  返回的数据类型datasets.base.Bunch(字典格式)
  - data：特征数据数组，是 [n_samples * n_features] 的二维 numpy.ndarray 数组
  - target：标签数组，是 n_samples 的一维 numpy.ndarray 数组
  - DESCR：数据描述
  - feature_names：特征名,新闻数据，手写数字、回归数据集没有
  - target_names：标签名

```
from sklearn.datasets import load_iris
# 获取鸢尾花数据集
iris = load_iris()
print("鸢尾花数据集的返回值：\n", iris)
# 返回值是一个继承自字典的Bench
print("鸢尾花的特征值:\n", iris["data"])
print("鸢尾花的目标值：\n", iris.target)
print("鸢尾花特征的名字：\n", iris.feature_names)
print("鸢尾花目标值的名字：\n", iris.target_names)
print("鸢尾花的描述：\n", iris.DESCR)
```

思考：拿到的数据是否全部都用来训练一个模型?

2.1.3 数据集的划分

机器学习一般的数据集会划分为两个部分：

- 训练数据：用于训练，**构建模型**
- 测试数据：在模型检验时使用，用于**评估模型是否有效**

划分比例：

- 训练集：70% 80% 75%
- 测试集：30% 20% 30%

数据集划分api

- sklearn.model_selection.train_test_split(

  arrays, *

  options)

  - x 数据集的特征值
  - y 数据集的标签值
  - test_size 测试集的大小，一般为float
  - random_state 随机数种子,不同的种子会造成不同的随机采样结果。相同的种子采样结果相同。
  - return 测试集特征训练集特征值值，训练标签，测试标签(默认随机取)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split


def datasets_demo():
    """
    对鸢尾花数据集的演示
    :return: None
    """
    # 1、获取鸢尾花数据集
    iris = load_iris()
    print("鸢尾花数据集的返回值：\n", iris)
    # 返回值是一个继承自字典的Bench
    print("鸢尾花的特征值:\n", iris["data"])
    print("鸢尾花的目标值：\n", iris.target)
    print("鸢尾花特征的名字：\n", iris.feature_names)
    print("鸢尾花目标值的名字：\n", iris.target_names)
    print("鸢尾花的描述：\n", iris.DESCR)
```

```
    # 2、对鸢尾花数据集进行分割
    # 训练集的特征值x_train 测试集的特征值x_test 训练集的目标值y_train 测试集的目标值
y_test
    x_train, x_test, y_train, y_test = train_test_split(iris.data, iris.target,
random_state=22)
    print("x_train:\n", x_train.shape)
    # 随机数种子
    x_train1, x_test1, y_train1, y_test1 = train_test_split(iris.data,
iris.target, random_state=6)
    x_train2, x_test2, y_train2, y_test2 = train_test_split(iris.data,
iris.target, random_state=6)
    print("如果随机数种子不一致：\n", x_train == x_train1)
    print("如果随机数种子一致：\n", x_train1 == x_train2)

    return None
```

2.2 特征工程介绍

学习目标

- 目标
  - 了解特征工程在机器学习当中的重要性
  - 知道特征工程的分类
- 应用
  - 无

2.2.1 为什么需要特征工程(Feature Engineering)

> 机器学习领域的大神Andrew Ng(吴恩达)老师说“Coming up with features is difficult, time-consuming, requires expert knowledge. “Applied machine learning” is basically feature engineering. ”
>
> 注：业界广泛流传：数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已。

2.2.2 什么是特征工程

特征工程是使用**专业背景知识和技巧处理数据，使得特征能在机器学习算法上发挥更好的作用的过程。**

- 意义：会直接影响机器学习的效果

2.2.3 特征工程的位置与数据处理的比较

- pandas:一个数据读取非常方便以及基本的处理格式的工具
- sklearn:对于特征的处理提供了强大的接口

特征工程包含内容

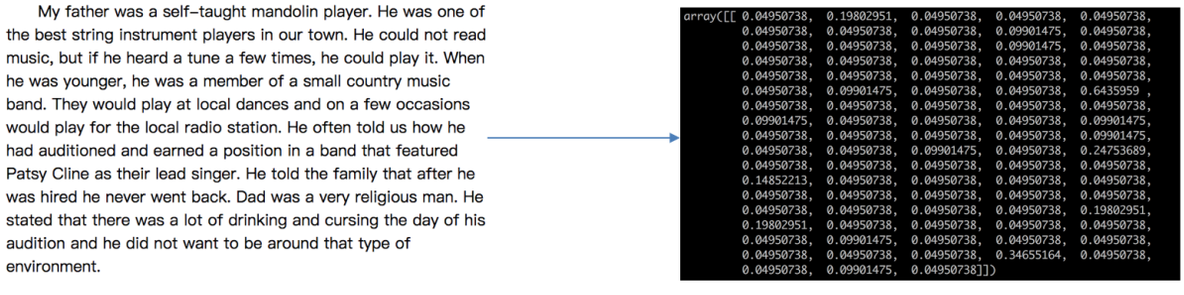
- 特征抽取
- 特征预处理
- 特征降维

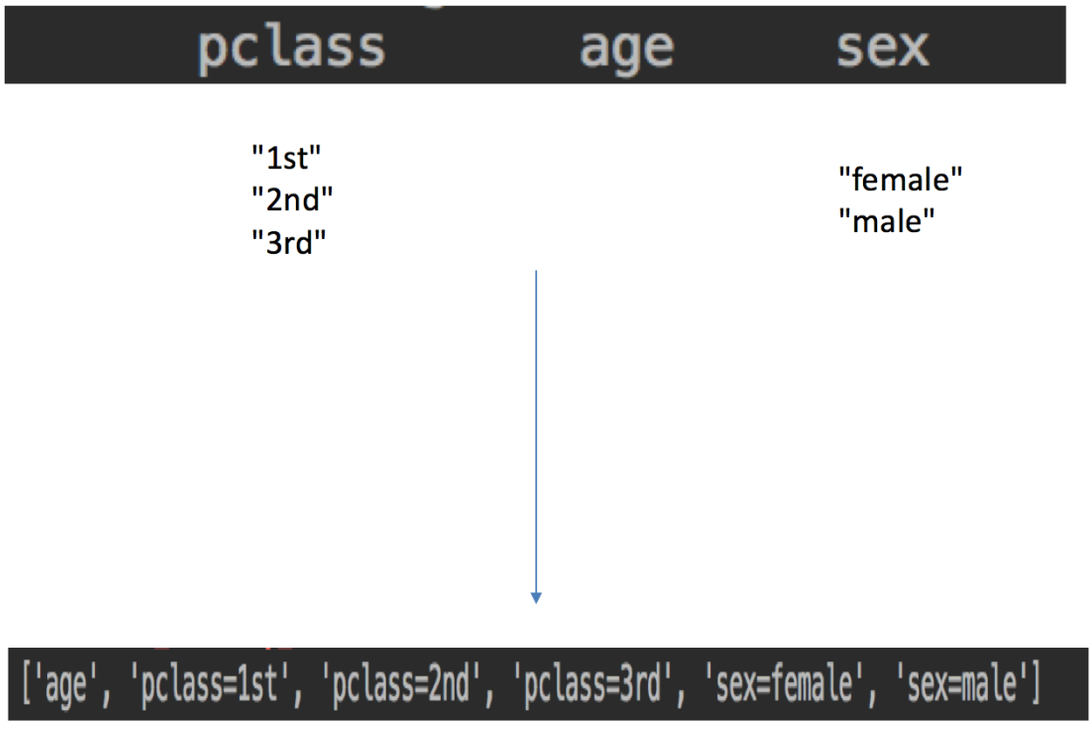
2.3 特征提取

学习目标

- 目标
  - 应用DictVectorizer实现对类别特征进行数值化、离散化
  - 应用CountVectorizer实现对文本特征进行数值化
  - 应用TfidfVectorizer实现对文本特征进行数值化
  - 说出两种文本特征提取的方式区别
- 应用
  - 无

什么是特征提取呢？





2.3.1 特征提取

1 将任意数据（如文本或图像）转换为可用于机器学习的数字特征

> 注：特征值化是为了计算机更好的去理解数据

- 字典特征提取(特征离散化)
- 文本特征提取
- 图像特征提取（深度学习将介绍）

2 特征提取API

```
sklearn.feature_extraction
```

2.3.2 字典特征提取

作用：对字典数据进行特征值化

- sklearn.feature_extraction.DictVectorizer(sparse=True,...)
  - DictVectorizer.fit_transform(X) X:字典或者包含字典的迭代器返回值：返回sparse矩阵
  - DictVectorizer.inverse_transform(X) X:array数组或者sparse矩阵 返回值:转换之前数据格式
  - DictVectorizer.get_feature_names() 返回类别名称

1 应用

我们对以下数据进行特征提取

```
[{'city': '北京','temperature':100}
{'city': '上海','temperature':60}
{'city': '深圳','temperature':30}]
```

```
['city=上海', 'city=北京', 'city=深圳', 'temperature']
[[  0.   1.   0. 100.]
 [  1.   0.   0.  60.]
 [  0.   0.   1.  30.]]
```

2 流程分析

- 实例化类DictVectorizer
- 调用fit_transform方法输入数据并转换（注意返回格式）

```
from sklearn.feature_extraction import DictVectorizer

def dict_demo():
    """
    对字典类型的数据进行特征抽取
    :return: None
    """
    data = [{'city': '北京','temperature':100}, {'city': '上海','temperature':60}, {'city': '深圳','temperature':30}]
    # 1、实例化一个转换器类
    transfer = DictVectorizer(sparse=False)
    # 2、调用fit_transform
    data = transfer.fit_transform(data)
    print("返回的结果:\n", data)
    # 打印特征名字
    print("特征名字：\n", transfer.get_feature_names())

    return None
```

注意观察没有加上sparse=False参数的结果

```
返回的结果:
   (0, 1)    1.0
  (0, 3)    100.0
  (1, 0)    1.0
  (1, 3)    60.0
  (2, 2)    1.0
  (2, 3)    30.0
特征名字:
 ['city=上海', 'city=北京', 'city=深圳', 'temperature']
```

这个结果并不是我们想要看到的，所以加上参数，得到想要的结果：

```
返回的结果:
 [[   0.    1.    0.  100.]
 [   1.    0.    0.   60.]
 [   0.    0.    1.   30.]]
特征名字:
 ['city=上海', 'city=北京', 'city=深圳', 'temperature']
```

之前在学习pandas中的离散化的时候，也实现了类似的效果。

我们把这个处理数据的技巧叫做”one-hot“编码：

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

转化为：

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

2.3 总结

对于特征当中存在类别信息的我们都会做one-hot编码处理

2.3.3 文本特征提取

作用：对文本数据进行特征值化

- **sklearn.feature_extraction.text.CountVectorizer(stop_words=[])**

    - 返回词频矩阵

- CountVectorizer.fit_transform(X) X:文本或者包含文本字符串的可迭代对象 返回值：返回sparse矩阵
- CountVectorizer.inverse_transform(X) X:array数组或者sparse矩阵 返回值:转换之前数据格
- CountVectorizer.get_feature_names() 返回值:单词列表

- **sklearn.feature_extraction.text.TfidfVectorizer**

1 应用

我们对以下数据进行特征提取

```
["life is short,i like python",
"life is too long,i dislike python"]
```

```
['dislike', 'is', 'life', 'like', 'long', 'python', 'short', 'too']
[[0 1 1 1 0 1 1 0]
 [1 1 1 0 1 1 0 1]]
```

2 流程分析

- 实例化类CountVectorizer
- 调用fit_transform方法输入数据并转换 （注意返回格式，利用toarray()进行sparse矩阵转换array数组）

```
from sklearn.feature_extraction.text import CountVectorizer

def text_count_demo():
    """
    对文本进行特征抽取，countvetorizer
    :return: None
    """
    data = ["life is short,i like like python", "life is too long,i dislike python"]
    # 1、实例化一个转换器类
    # transfer = CountVectorizer(sparse=False)
    transfer = CountVectorizer()
    # 2、调用fit_transform
    data = transfer.fit_transform(data)
    print("文本特征抽取的结果：\n", data.toarray())
    print("返回特征名字：\n", transfer.get_feature_names())

    return None
```

返回结果：

```
文本特征抽取的结果：
 [[0 1 1 2 0 1 1 0]
 [1 1 1 0 1 1 0 1]]
返回特征名字：
 ['dislike', 'is', 'life', 'like', 'long', 'python', 'short', 'too']
```

问题:如果我们将数据替换成中文?

```
"人生苦短，我喜欢Python" "生活太长久，我不喜欢Python"
```

那么最终得到的结果是

```
#  文档进行特征值化
# 导入包
from sklearn.feature_extraction.text import CountVectorizer

# 实例化

vector = CountVectorizer()


# 调用fit_transform

res = vector.fit_transform(["人生 苦短，我喜 欢py thon","生活 太长了，我不喜欢  python"])

print(vector.get_feature_names())

print(res.toarray())
```

不支持单个中文字！

```
Run test
/Users/huxinghui/virtualenv/ml3/bin/python3.6 /Users/huxinghui/workspace/ml/recordclass/test_01.py
['python', 'thon', '人生', '太长了', '我不喜欢', '我喜', '欢py', '生活', '苦短']
[[0 1 1 0 0 1 1 0 1]
 [1 0 0 1 1 0 0 1 0]]

Process finished with exit code 0
```

为什么会得到这样的结果呢，**仔细分析之后会发现英文默认是以空格分开的。其实就达到了一个分词的效果**，所以我们要对中文进行分词处理

3 jieba分词处理

- jieba.cut()
    - 返回词语组成的生成器

需要安装下jieba库

```
pip3 install jieba
```

4 案例分析

对以下三句话进行特征值化

```
今天很残酷，明天更残酷，后天很美好，
但绝对大部分是死在明天晚上，所以每个人不要放弃今天。

我们看到的从很远星系来的光是在几百万年之前发出的，
这样当我们看到宇宙时，我们是在看它的过去。

如果只用一种方式了解某样事物，你就不会真正了解它。
了解事物真正含义的秘密取决于如何将其与我们所了解的事物相联系。
```

- 分析
    - 准备句子，利用jieba.cut进行分词

- 实例化CountVectorizer
- 将分词结果变成字符串当作fit_transform的输入值

```
['一种', '不会', '之前', '了解', '事物', '今天', '光是在', '几百万年', '发出', '取决于', '只用', '后天', '含义', '大部分', '如何', '如
[[0 0 0 0 0 2 0 0 0 0 0 1 0 1 0 0 0 1 0 2 0 1 0 2 1 0 0 0 1 1 0 0 0]
 [0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 2 0 0 0 0 0 1 1]
 [1 1 0 4 3 0 0 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 2 1 0 0 1 0 0]]
```

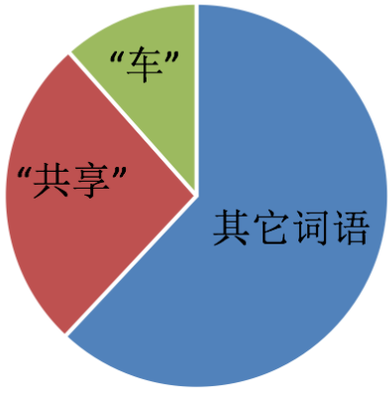
```
from sklearn.feature_extraction.text import CountVectorizer
import jieba

def cut_word(text):
    """
    对中文进行分词
    "我爱北京天安门"----->"我 爱 北京 天安门"
    :param text:
    :return: text
    """
    # 用结巴对中文字符串进行分词
    text = " ".join(list(jieba.cut(text)))

    return text

def text_chinese_count_demo2():
    """
    对中文进行特征抽取
    :return: None
    """
    data = ["一种还是一种今天很残酷，明天更残酷，后天很美好，但绝对大部分是死在明天晚上，所以每个人不要放弃今天。",
            "我们看到的从很远星系来的光是在几百万年之前发出的，这样当我们看到宇宙时，我们是在看它的过去。",
            "如果只用一种方式了解某样事物，你就不会真正了解它。了解事物真正含义的秘密取决于如何将其与我们所了解的事物相联系。"]
    # 将原始数据转换成分好词的形式
    text_list = []
    for sent in data:
        text_list.append(cut_word(sent))
    print(text_list)

    # 1、实例化一个转换器类
    # transfer = CountVectorizer(sparse=False)
    transfer = CountVectorizer()
    # 2、调用fit_transform
    data = transfer.fit_transform(text_list)
    print("文本特征抽取的结果：\n", data.toarray())
    print("返回特征名字：\n", transfer.get_feature_names())

    return None
```

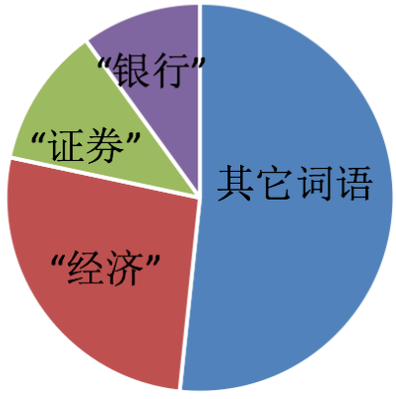
返回结果：

```
Building prefix dict from the default dictionary ...
Dumping model to file cache
/var/folders/mz/tzf2l3sx4rgg6qpglfb035_r0000gn/T/jieba.cache
Loading model cost 1.032 seconds.
['一种 还是 一种 今天 很 残酷 ， 明天 更 残酷 ， 后天 很 美好 ， 但 绝对 大部分 是 死 在 明天 晚上 ， 所以 每个 人 不要 放弃 今天 。', '我们 看到 的 从 很 远 星系 来 的 光是在 几百万年 之前 发出 的 ， 这样 当 我们 看到 宇宙 时 ， 我们 是 在 看 它 的 过去 。', '如果 只用 一种 方式 了解 某样 事物 ， 你 就 不会 真正 了解 它 。 了解 事物 真正 含义 的 秘密 取决于 如何 将 其 与 我们 所 了解 的 事物 相 联系 。']
Prefix dict has been built succesfully.
文本特征抽取的结果：
 [[2 0 1 0 0 0 2 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 2 0 1 0 2 1 0 0 0 1 1 0 0 1 0]
 [0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 3 0 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 1 0 1]
 [1 1 0 0 4 3 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 1 0 0 0 2 1 0 0 1 0 0 0]]
返回特征名字：
 ['一种', '不会', '不要', '之前', '了解', '事物', '今天', '光是在', '几百万年', '发出', '取决于', '只用', '后天', '含义', '大部分', '如何', '如果', '宇宙', '我们', '所以', '放弃', '方式', '明天', '星系', '晚上', '某样', '残酷', '每个', '看到', '真正', '秘密', '绝对', '美好', '联系', '过去', '还是', '这样']
```

但如果把这样的词语特征用于分类，会出现什么问题?

请看问题：





文章类型？？？

该如何处理某个词或短语在多篇文章中出现的次数高这种情况

5 Tf-idf文本特征提取

- TF-IDF的主要思想是：如果**某个词或短语在一篇文章中出现的概率高，并且在其他文章中很少出现**，则认为此词或者短语具有很好的类别区分能力，适合用来分类。
- **TF-IDF作用：用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。**

5.1 公式

- 词频（term frequency，tf）指的是某一个给定的词语在该文件中出现的频率
- 逆向文档频率（inverse document frequency，idf）是一个词语普遍重要性的度量。某一特定词语的idf，可以**由总文件数目除以包含该词语之文件的数目，再将得到的商取以10为底的对数得到**

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

最终得出结果可以理解为重要程度。

注：假如一篇文件的总词语数是100个，而词语"非常"出现了5次，那么"非常"一词在该文件中的词频就是5/100=0.05。而计算文件频率（IDF）的方法是以文件集的文件总数，除以出现"非常"一词的文件数。所以，如果"非常"一词在1,000份文件出现过，而文件总数是10,000,000份的话，其逆向文件频率就是lg（10,000,000 / 1,0000）=3。最后"非常"对于这篇文档的tf-idf的分数为0.05 * 3=0.15

5.2 案例

```
from sklearn.feature_extraction.text import TfidfVectorizer
import jieba

def cut_word(text):
    """
    对中文进行分词
    "我爱北京天安门"----->"我 爱 北京 天安门"
    :param text:
    :return: text
    """
    # 用结巴对中文字符串进行分词
    text = " ".join(list(jieba.cut(text)))

    return text

def text_chinese_tfidf_demo():
    """
    对中文进行特征抽取
    :return: None
    """
    data = ["一种还是一种今天很残酷，明天更残酷，后天很美好，但绝对大部分是死在明天晚上，所以每个人不要放弃今天。",
            "我们看到的从很远星系来的光是在几百万年之前发出的，这样当我们看到宇宙时，我们是在看它的过去。",
            "如果只用一种方式了解某样事物，你就不会真正了解它。了解事物真正含义的秘密取决于如何将其与我们所了解的事物相联系。"]
    # 将原始数据转换成分好词的形式
    text_list = []
    for sent in data:
        text_list.append(cut_word(sent))
    print(text_list)

    # 1、实例化一个转换器类
    # transfer = CountVectorizer(sparse=False)
    transfer = TfidfVectorizer(stop_words=['一种', '不会', '不要'])
    # 2、调用fit_transform
```

```
    data = transfer.fit_transform(text_list)
    print("文本特征抽取的结果：\n", data.toarray())
    print("返回特征名字：\n", transfer.get_feature_names())

    return None
```

返回结果：

```
Building prefix dict from the default dictionary ...
Loading model from cache
/var/folders/mz/tzf2l3sx4rgg6qpglfb035_r0000gn/T/jieba.cache
Loading model cost 0.856 seconds.
Prefix dict has been built succesfully.
['一种 还是 一种 今天 很 残酷 ， 明天 更 残酷 ， 后天 很 美好 ， 但 绝对 大部分 是 死 在 明天 晚上 ， 所以 每个 人 不要 放弃 今天 。', '我们 看到 的 从 很 远 星系 来 的 光是在 几百万年 之前 发出 的 ， 这样 当 我们 看到 宇宙 时 ， 我们 是 在 看 它 的 过去 。', '如果 只用 一种 方式 了解 某样 事物 ， 你 就 不会 真正 了解 它 。 了解 事物 真正 含义 的 秘密 取决于 如何 将 其 与 我们 所 了解 的 事物 相 联系 。']
文本特征抽取的结果：
 [[ 0.          0.          0.          0.43643578  0.          0.          0.
   0.          0.          0.21821789  0.          0.21821789  0.          0.
   0.          0.          0.21821789  0.21821789  0.          0.43643578
   0.          0.21821789  0.          0.43643578  0.21821789  0.          0.
   0.          0.21821789  0.21821789  0.          0.          0.21821789
   0.        ]
 [ 0.2410822   0.          0.          0.          0.2410822   0.2410822
   0.2410822   0.          0.          0.          0.          0.          0.
   0.          0.2410822   0.55004769  0.          0.          0.          0.
   0.2410822   0.          0.          0.          0.          0.48216441
   0.          0.          0.          0.          0.          0.2410822
   0.          0.2410822 ]
 [ 0.          0.644003    0.48300225  0.          0.          0.          0.
   0.16100075  0.16100075  0.          0.16100075  0.          0.16100075
   0.16100075  0.          0.12244522  0.          0.          0.16100075
   0.          0.          0.          0.16100075  0.          0.          0.
   0.3220015   0.16100075  0.          0.          0.16100075  0.          0.
   0.        ]]
返回特征名字：
 ['之前', '了解', '事物', '今天', '光是在', '几百万年', '发出', '取决于', '只用', '后天', '含义', '大部分', '如何', '如果', '宇宙', '我们', '所以', '放弃', '方式', '明天', '星系', '晚上', '某样', '残酷', '每个', '看到', '真正', '秘密', '绝对', '美好', '联系', '过去', '还是', '这样']
```

6 Tf-idf的重要性

分类机器学习算法进行文章分类中前期数据处理方式

2.4 特征预处理

学习目标

- 目标
  - 了解数值型数据、类别型数据特点
  - 应用MinMaxScaler实现对特征数据进行归一化
  - 应用StandardScaler实现对特征数据进行标准化

- 应用
  - 无

什么是特征预处理?

特征1	特征2	特征3	特征4
90	2	10	40
60	4	15	45
75	3	13	46

特征1	特征2	特征3	特征4
1.	0.	0.	0.
0.	1.	1.	0.83
0.5	0.5	0.6	1.

2.4.1 什么是特征预处理

```
# scikit-learn的解释
provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.
```

翻译过来：通过**一些转换函数**将特征数据**转换成更加适合算法模型**的特征数据过程

可以通过上面那张图来理解

1 包含内容

- 数值型数据的无量纲化：
  - 归一化
  - 标准化

2 特征预处理API

```
sklearn.preprocessing
```

为什么我们要进行归一化/标准化?

- 特征的**单位或者大小相差较大，或者某特征的方差相比其他的特征要大出几个数量级，容易影响（支配）目标结果**，使得一些算法无法学习到其它的特征

约会对象数据

相亲约会对象数据，这个样本时男士的数据，三个特征，玩游戏所消耗时间的百分比、每年获得的飞行常客里程数、每周消费的冰淇淋公升数。然后有一个所属类别，被女士评价的三个类别，不喜欢didnt、魅力一般small、极具魅力large也许也就是说飞行里程数对于结算结果或者说相亲结果影响较大，但是统计的人觉得这三个特征同等重要。

里程数	公升数	消耗时间比	评价
14488	7.153469	1.673904	smallDoses
26052	1.441871	0.805124	didntLike
75136	13.147394	0.428964	didntLike
38344	1.669788	0.134296	didntLike
72993	10.141740	1.032955	didntLike
35948	6.830792	1.213192	largeDoses
42666	13.276369	0.543880	largeDoses
67497	8.631577	0.749278	didntLike
35483	12.273169	1.508053	largeDoses
50242	3.723498	0.831917	didntLike

我们需要用到一些方法进行**无量纲化**，**使不同规格的数据转换到同一规格**

2.4.2 归一化

1 定义

通过对原始数据进行变换把数据映射到(默认为[0,1])之间

2 公式

$$X' = \frac{x - min}{max - min} \qquad X'' = X' * (mx - mi) + mi$$

> 作用于每一列，max为一列的最大值，min为一列的最小值,那么X”为最终结果，mx，mi分别为指定区间值默认mx为1,mi为0

那么怎么理解这个过程呢？我们通过一个例子

特征1	特征2	特征3	特征4
90	2	10	40
60	4	15	45
75	3	13	46

特征1	特征2	特征3	特征4
$\frac{90-60}{90-60}$	$\frac{2-2}{4-2}$	$\frac{10-10}{15-10}$	$\frac{40-40}{46-40}$
$\frac{60-60}{90-60}$	$\frac{4-2}{4-2}$	$\frac{15-10}{15-10}$	$\frac{45-40}{46-40}$
$\frac{75-60}{90-60}$	$\frac{3-2}{4-2}$	$\frac{13-10}{15-10}$	$\frac{46-40}{46-40}$

注：里面是第一步，还需要第二步乘以(1-0)+0

3 API

- sklearn.preprocessing.MinMaxScaler (feature_range=(0,1)...)
    - MinMaxScalar.fit_transform(X)
        - X:numpy array格式的数据[n_samples,n_features]
    - 返回值：转换后的形状相同的array

4 数据计算

我们对以下数据进行运算，在dating.txt中。保存的就是之前的约会对象数据

```
milage,Liters,Consumtime,target
40920,8.326976,0.953952,3
14488,7.153469,1.673904,2
26052,1.441871,0.805124,1
75136,13.147394,0.428964,1
38344,1.669788,0.134296,1
```

- 分析

1、实例化MinMaxScalar

2、通过fit_transform转换

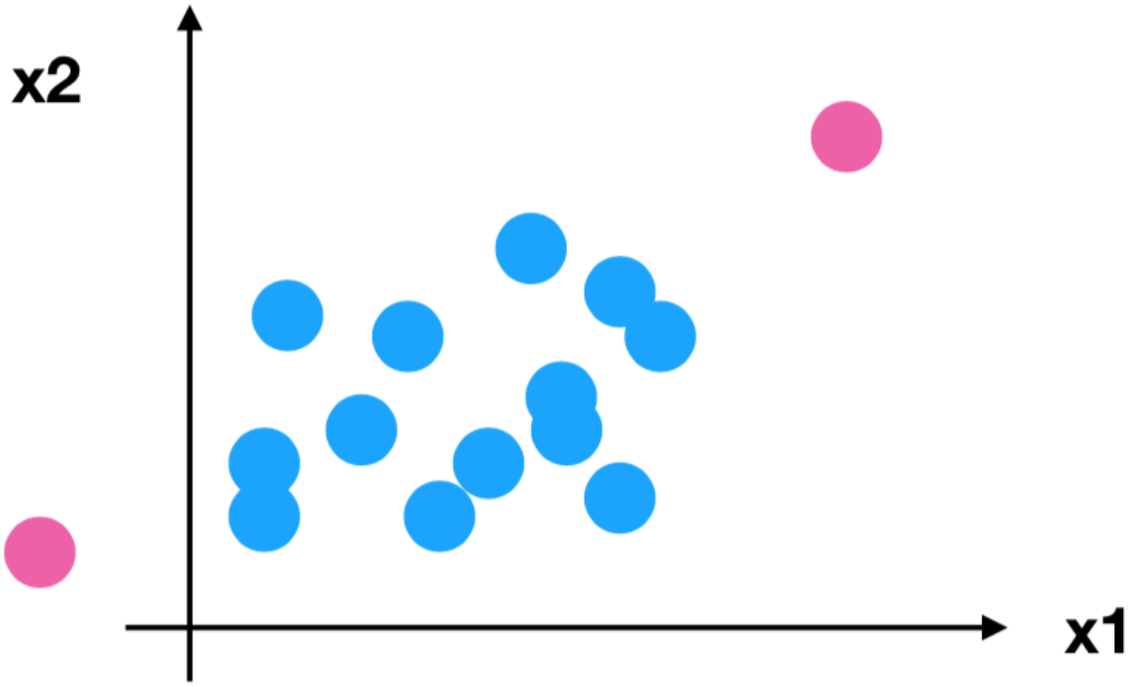
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

def minmax_demo():
    """
    归一化演示
    :return: None
    """
    data = pd.read_csv("dating.txt")
    print(data)
    # 1、实例化一个转换器类
    transfer = MinMaxScaler(feature_range=(2, 3))
    # 2、调用fit_transform
    data = transfer.fit_transform(data[['milage','Liters','Consumtime']])
    print("最小值最大值归一化处理的结果：\n", data)

    return None
```

返回结果：

```
      milage      Liters  Consumtime  target
0      40920    8.326976    0.953952       3
1      14488    7.153469    1.673904       2
2      26052    1.441871    0.805124       1
3      75136   13.147394    0.428964       1
..       ...         ...         ...     ...
998    48111    9.134528    0.728045       3
999    43757    7.882601    1.332446       3

[1000 rows x 4 columns]
最小值最大值归一化处理的结果：
```

```
[[ 2.44832535  2.39805139  2.56233353]
 [ 2.15873259  2.34195467  2.98724416]
 [ 2.28542943  2.06892523  2.47449629]
 ...,
 [ 2.29115949  2.50910294  2.51079493]
 [ 2.52711097  2.43665451  2.4290048 ]
 [ 2.47940793  2.3768091   2.78571804]]
```

问题：如果数据中异常点较多，会有什么影响？



5 归一化总结

注意最大值最小值是变化的，另外，最大值与最小值非常容易受异常点影响，**所以这种方法鲁棒性较差，只适合传统精确小数据场景。**

怎么办？

2.4.3 标准化

1 定义

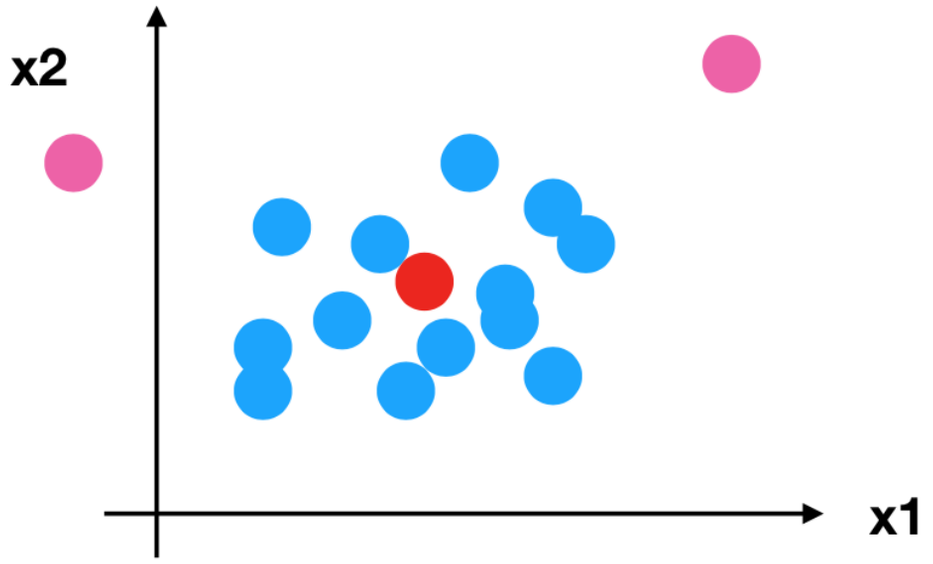
通过对原始数据进行变换把数据变换到均值为0,标准差为1范围内

2 公式

$$X' = \frac{x - \text{mean}}{\sigma}$$

> 作用于每一列，mean为平均值，σ为标准差

所以回到刚才异常点的地方，我们再来看看标准化



- 对于归一化来说：如果出现异常点，影响了最大值和最小值，那么结果显然会发生改变
- 对于标准化来说：如果出现异常点，由于具有一定数据量，少量的异常点对于平均值的影响并不大，从而方差改变较小。

3 API

- sklearn.preprocessing.StandardScaler( )
  - 处理之后每列来说所有数据都聚集在均值0附近标准差差为1
  - StandardScaler.fit_transform(X)
    - X:numpy array格式的数据[n_samples,n_features]
  - 返回值：转换后的形状相同的array

4 数据计算

同样对上面的数据进行处理

- 分析

1、实例化StandardScaler

2、通过fit_transform转换

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

def stand_demo():
    """
    标准化演示
    :return: None
    """
    data = pd.read_csv("dating.txt")
    print(data)
    # 1、实例化一个转换器类
```

```
    transfer = StandardScaler()
    # 2、调用fit_transform
    data = transfer.fit_transform(data[['milage','Liters','Consumtime']])
    print("标准化的结果:\n", data)
    print("每一列特征的平均值：\n", transfer.mean_)
    print("每一列特征的方差：\n", transfer.var_)

    return None
```

返回结果：

```
      milage     Liters  Consumtime  target
0      40920   8.326976    0.953952       3
1      14488   7.153469    1.673904       2
2      26052   1.441871    0.805124       1
..       ...        ...         ...     ...
997    26575  10.650102    0.866627       3
998    48111   9.134528    0.728045       3
999    43757   7.882601    1.332446       3

[1000 rows x 4 columns]
标准化的结果:
 [[ 0.33193158  0.41660188  0.24523407]
 [-0.87247784  0.13992897  1.69385734]
 [-0.34554872 -1.20667094 -0.05422437]
 ...,
 [-0.32171752  0.96431572  0.06952649]
 [ 0.65959911  0.60699509 -0.20931587]
 [ 0.46120328  0.31183342  1.00680598]]
每一列特征的平均值：
 [  3.36354210e+04   6.55996083e+00   8.32072997e-01]
每一列特征的方差：
 [  4.81628039e+08   1.79902874e+01   2.46999554e-01]
```

5 标准化总结

在已有样本足够多的情况下比较稳定，适合现代嘈杂大数据场景

2.5 特征降维

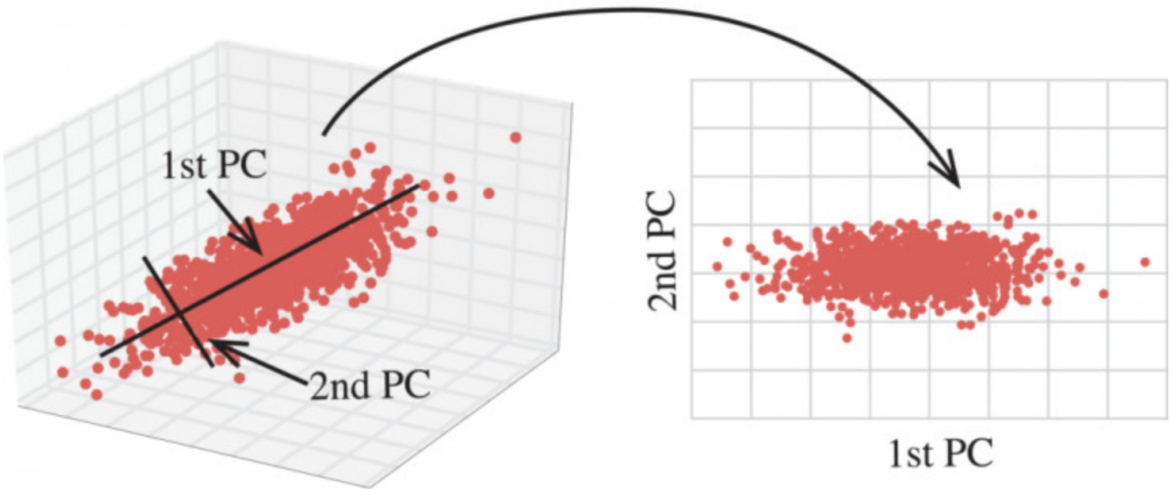
学习目标

- 目标
  - 知道特征选择的嵌入式、过滤式以及包裹氏三种方式
  - 应用VarianceThreshold实现删除低方差特征
  - 了解相关系数的特点和计算
  - 应用相关性系数实现特征选择
- 应用
  - 无

2.5.1 降维

降维是指在某些限定条件下，**降低随机变量(特征)个数**，得到**一组“不相关”主变量**的过程

- 降低随机变量的个数



- 相关特征(correlated feature)
    - 相对湿度与降雨量之间的相关
    - 等等

> 正是因为在进行训练的时候，我们都是使用特征进行学习。如果特征本身存在问题或者特征之间相关性较强，对于算法学习预测会影响较大

2.5.2 降维的两种方式

- **特征选择**
- **主成分分析（可以理解一种特征提取的方式）**

2.5.3 什么是特征选择

1 定义

数据中包含**冗余或无关变量（或称特征、属性、指标等）**，旨在从**原有特征中找出主要特征。**

特征？

1、羽毛颜色

2、眼睛宽度

3、眼睛长度

4、爪子长度

5、体格大小

2 方法

- Filter(过滤式)：主要探究特征本身特点、特征与特征和目标值之间关联
  - **方差选择法：低方差特征过滤**
  - **相关系数**
- Embedded (嵌入式)：算法自动选择特征（特征与目标值之间的关联）
  - **决策树:信息熵、信息增益**
  - **正则化：L1、L2**
  - **深度学习：卷积等**

> 对于Embedded方式，只能在讲解算法的时候在进行介绍，更好的去理解

3 模块

```
sklearn.feature_selection
```

4 过滤式

4.1 低方差特征过滤

删除低方差的一些特征，前面讲过方差的意义。再结合方差的大小来考虑这个方式的角度。

- 特征方差小：某个特征大多样本的值比较相近
- 特征方差大：某个特征很多样本的值都有差别

4.1.1 API

- sklearn.feature_selection.VarianceThreshold(threshold = 0.0)
  - 删除所有低方差特征
  - Variance.fit_transform(X)
    - X:numpy array格式的数据[n_samples,n_features]
    - 返回值：训练集差异低于threshold的特征将被删除。默认值是保留所有非零方差特征，即删除所有样本中具有相同值的特征。

4.1.2 数据计算

我们对**某些股票的指标特征之间进行一个筛选**，数据在"factor_regression_data/factor_returns.csv"文件当中,除去'index','date','return'列不考虑**（这些类型不匹配，也不是所需要指标）**

一共这些特征

```
pe_ratio,pb_ratio,market_cap,return_on_asset_net_profit,du_return_on_equity,ev,earnings_per_share,revenue,total_expense
index,pe_ratio,pb_ratio,market_cap,return_on_asset_net_profit,du_return_on_equity,ev,earnings_per_share,revenue,total_expense,date,return
0,000001.XSHE,5.9572,1.1818,85252550922.0,0.8008,14.9403,1211444855670.0,2.01,20701401000.0,10882540000.0,2012-01-31,0.027657228229937388
1,000002.XSHE,7.0289,1.588,84113358168.0,1.6463,7.8656,300252061695.0,0.326,29308369223.2,23783476901.2,2012-01-31,0.08235182370820669
2,000008.XSHE,-262.7461,7.0003,517045520.0,-0.5678,-0.5943,770517752.56,-0.006,11679829.03,12030080.04,2012-01-31,0.09978900335112327
3,000060.XSHE,16.476,3.7146,19680455995.0,5.6036,14.617,28009159184.6,0.35,9189386877.65,7935542726.05,2012-01-31,0.12159482758620697
4,000069.XSHE,12.5878,2.5616,41727214853.0,2.8729,10.9097,81247380359.0,0.271,8951453490.28,7091397989.13,2012-01-31,-0.0026808154146886697
```

- 分析

1、初始化VarianceThreshold,指定阀值方差

2、调用fit_transform

```
def variance_demo():
    """
    删除低方差特征——特征选择
    :return: None
    """
    data = pd.read_csv("factor_returns.csv")
    print(data)
    # 1、实例化一个转换器类
    transfer = VarianceThreshold(threshold=1)
    # 2、调用fit_transform
    data = transfer.fit_transform(data.iloc[:, 1:10])
    print("删除低方差特征的结果：\n", data)
    print("形状：\n", data.shape)

    return None
```

返回结果：

```
           index  pe_ratio  pb_ratio    market_cap  \
0     000001.XSHE    5.9572    1.1818  8.525255e+10
1     000002.XSHE    7.0289    1.5880  8.411336e+10
...           ...       ...       ...           ...
2316  601958.XSHG   52.5408    2.4646  3.287910e+10
2317  601989.XSHG   14.2203    1.4103  5.911086e+10

      return_on_asset_net_profit  du_return_on_equity           ev  \
0                         0.8008              14.9403  1.211445e+12
1                         1.6463               7.8656  3.002521e+11
...                          ...                  ...           ...
2316                      2.7444               2.9202  3.883803e+10
2317                      2.0383               8.6179  2.020661e+11

      earnings_per_share       revenue  total_expense        date    return
0                 2.0100  2.070140e+10   1.088254e+10  2012-01-31  0.027657
1                 0.3260  2.930837e+10   2.378348e+10  2012-01-31  0.082352
```

```
2               -0.0060  1.167983e+07   1.203008e+07  2012-01-31  0.099789
...                 ...           ...            ...         ...       ...
2315             0.2200  1.789082e+10   1.749295e+10  2012-11-30  0.137134
2316             0.1210  6.465392e+09   6.009007e+09  2012-11-30  0.149167
2317             0.2470  4.509872e+10   4.132842e+10  2012-11-30  0.183629

[2318 rows x 12 columns]
删除低方差特征的结果：
 [[  5.95720000e+00   1.18180000e+00   8.52525509e+10 ...,   1.21144486e+12
    2.07014010e+10   1.08825400e+10]
 [  7.02890000e+00   1.58800000e+00   8.41133582e+10 ...,   3.00252062e+11
    2.93083692e+10   2.37834769e+10]
 [ -2.62746100e+02   7.00030000e+00   5.17045520e+08 ...,   7.70517753e+08
    1.16798290e+07   1.20300800e+07]
 ...,
 [  3.95523000e+01   4.00520000e+00   1.70243430e+10 ...,   2.42081699e+10
    1.78908166e+10   1.74929478e+10]
 [  5.25408000e+01   2.46460000e+00   3.28790988e+10 ...,   3.88380258e+10
    6.46539204e+09   6.00900728e+09]
 [  1.42203000e+01   1.41030000e+00   5.91108572e+10 ...,   2.02066110e+11
    4.50987171e+10   4.13284212e+10]]
形状：
 (2318, 8)
```

4.2 相关系数

- 皮尔逊相关系数(Pearson Correlation Coefficient)
    - 反映变量之间相关关系密切程度的统计指标

4.2.2 公式计算案例(了解，不用记忆)

- 公式

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{n\sum x^2 - (\sum x)^2}\sqrt{n\sum y^2 - (\sum y)^2}}$$

- 比如说我们计算年广告费投入与月均销售额

表1　　广告费与月平均销售额相关表　　单位：万元

年广告费投入	月均销售额
12.5	21.2
15.3	23.9
23.2	32.9
26.4	34.1
33.5	42.5
34.4	43.2
39.4	49.0
45.2	52.8
55.4	59.4
60.9	63.5

那么之间的相关系数怎么计算

序号	广告投入(万元) x	月均销售额(万元) y	x^2	y^2	xy
1	12.5	21.2	156.25	449.44	265.00
2	15.3	23.9	234.09	571.21	365.67
3	23.2	32.9	538.24	1082.41	763.28
4	26.4	34.1	696.96	1162.81	900.24
5	33.5	42.5	1122.25	1806.25	1423.75
6	34.4	43.2	1183.36	1866.24	1486.08
7	39.4	49.0	1552.36	2401.00	1930.60
8	45.2	52.8	2043.04	2787.84	2386.56
9	55.4	59.4	3069.16	3528.36	3290.76
10	60.9	63.5	3708.81	4032.25	3867.15
合计	346.2	422.5	14304.52	19687.81	16679.09

最终计算：

$$\frac{10 \times 16679.09 - 346.2 \times 422.5}{\sqrt{10 \times 14304.52 - 346.2^2}\sqrt{10 \times 19687.81 - 422.5^2}}$$

= 0.9942

所以我们最终得出结论是广告投入费与月平均销售额之间有高度的正相关关系。

4.2.3 特点

相关系数的值介于–1与+1之间，即–1≤ r ≤+1。其性质如下：

- **当r>0时，表示两变量正相关，r<0时，两变量为负相关**
- 当|r|=1时，表示两变量为完全相关，当r=0时，表示两变量间无相关关系
- **当0<|r|<1时，表示两变量存在一定程度的相关。且|r|越接近1，两变量间线性关系越密切；|r|越接近于0，表示两变量的线性相关越弱**
- **一般可按三级划分：|r|<0.4为低度相关；0.4≤|r|<0.7为显著性相关；0.7≤|r|<1为高度线性相关**

> 这个符号：|r|为r的绝对值， |-5| = 5

4.2.4 API

- from scipy.stats import pearsonr
  - x : (N,) array_like
  - y : (N,) array_like Returns: (Pearson's correlation coefficient, p-value)

4.2.5 案例：股票的财务指标相关性计算

我们刚才的股票的这些指标进行相关性计算， 假设我们以

```
factor = ['pe_ratio','pb_ratio','market_cap','return_on_asset_net_profit','du_return_on_equity','ev','earnings_per_share','revenue','total_expense']
```

这些特征当中的两两进行计算，得出相关性高的一些特征

```
/Users/huxinghui/virtualenv/ml3/bin/python3.6 /Users/huxinghui/workspace/ml/python_class/test.py
指标pe_ratio与指标pb_ratio之间的相关性大小为-0.004389
指标pe_ratio与指标market_cap之间的相关性大小为-0.068861
指标pe_ratio与指标return_on_asset_net_profit之间的相关性大小为-0.066009
指标pe_ratio与指标du_return_on_equity之间的相关性大小为-0.082364
指标pe_ratio与指标ev之间的相关性大小为-0.046159
指标pe_ratio与指标earnings_per_share之间的相关性大小为-0.072082
指标pe_ratio与指标revenue之间的相关性大小为-0.058693
指标pe_ratio与指标total_expense之间的相关性大小为-0.055551
指标pb_ratio与指标market_cap之间的相关性大小为0.009336
指标pb_ratio与指标return_on_asset_net_profit之间的相关性大小为0.445381
指标pb_ratio与指标du_return_on_equity之间的相关性大小为0.291367
指标pb_ratio与指标ev之间的相关性大小为-0.183232
指标pb_ratio与指标earnings_per_share之间的相关性大小为0.198708
指标pb_ratio与指标revenue之间的相关性大小为-0.177671
指标pb_ratio与指标total_expense之间的相关性大小为-0.173339
指标market_cap与指标return_on_asset_net_profit之间的相关性大小为0.214774
指标market_cap与指标du_return_on_equity之间的相关性大小为0.316288
指标market_cap与指标ev之间的相关性大小为0.565533
指标market_cap与指标earnings_per_share之间的相关性大小为0.524179
指标market_cap与指标revenue之间的相关性大小为0.440653
指标market_cap与指标total_expense之间的相关性大小为0.386550
指标return_on_asset_net_profit与指标du_return_on_equity之间的相关性大小为0.818697
```

- 分析
  - 两两特征之间进行相关性计算

```
import pandas as pd
from scipy.stats import pearsonr

def pearsonr_demo():
    """
    相关系数计算
    :return: None
    """
    data = pd.read_csv("factor_returns.csv")
```

```
    factor = ['pe_ratio', 'pb_ratio', 'market_cap',
'return_on_asset_net_profit', 'du_return_on_equity', 'ev',
              'earnings_per_share', 'revenue', 'total_expense']

    for i in range(len(factor)):
        for j in range(i, len(factor) - 1):
            print(
                "指标%s与指标%s之间的相关性大小为%f" % (factor[i], factor[j + 1],
pearsonr(data[factor[i]], data[factor[j + 1]])[0]))

    return None
```

返回结果：

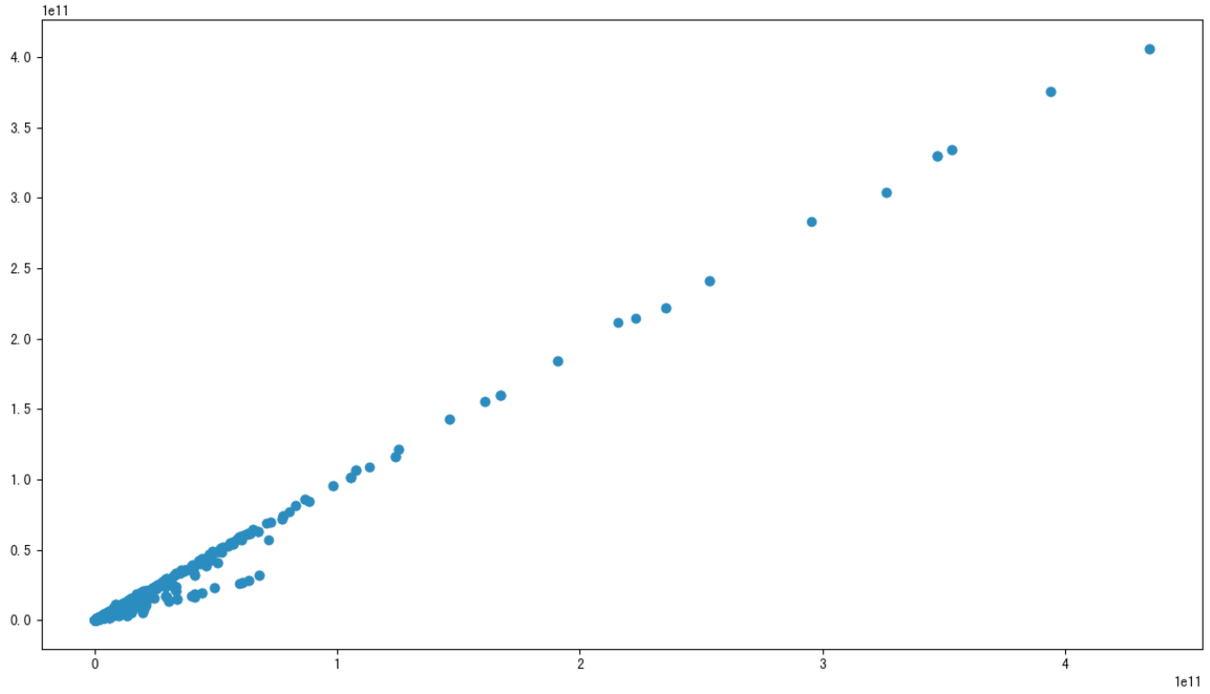
```
指标pe_ratio与指标pb_ratio之间的相关性大小为-0.004389
指标pe_ratio与指标market_cap之间的相关性大小为-0.068861
指标pe_ratio与指标return_on_asset_net_profit之间的相关性大小为-0.066009
指标pe_ratio与指标du_return_on_equity之间的相关性大小为-0.082364
指标pe_ratio与指标ev之间的相关性大小为-0.046159
指标pe_ratio与指标earnings_per_share之间的相关性大小为-0.072082
指标pe_ratio与指标revenue之间的相关性大小为-0.058693
指标pe_ratio与指标total_expense之间的相关性大小为-0.055551
指标pb_ratio与指标market_cap之间的相关性大小为0.009336
指标pb_ratio与指标return_on_asset_net_profit之间的相关性大小为0.445381
指标pb_ratio与指标du_return_on_equity之间的相关性大小为0.291367
指标pb_ratio与指标ev之间的相关性大小为-0.183232
指标pb_ratio与指标earnings_per_share之间的相关性大小为0.198708
指标pb_ratio与指标revenue之间的相关性大小为-0.177671
指标pb_ratio与指标total_expense之间的相关性大小为-0.173339
指标market_cap与指标return_on_asset_net_profit之间的相关性大小为0.214774
指标market_cap与指标du_return_on_equity之间的相关性大小为0.316288
指标market_cap与指标ev之间的相关性大小为0.565533
指标market_cap与指标earnings_per_share之间的相关性大小为0.524179
指标market_cap与指标revenue之间的相关性大小为0.440653
指标market_cap与指标total_expense之间的相关性大小为0.386550
指标return_on_asset_net_profit与指标du_return_on_equity之间的相关性大小为0.818697
指标return_on_asset_net_profit与指标ev之间的相关性大小为-0.101225
指标return_on_asset_net_profit与指标earnings_per_share之间的相关性大小为0.635933
指标return_on_asset_net_profit与指标revenue之间的相关性大小为0.038582
指标return_on_asset_net_profit与指标total_expense之间的相关性大小为0.027014
指标du_return_on_equity与指标ev之间的相关性大小为0.118807
指标du_return_on_equity与指标earnings_per_share之间的相关性大小为0.651996
指标du_return_on_equity与指标revenue之间的相关性大小为0.163214
指标du_return_on_equity与指标total_expense之间的相关性大小为0.135412
指标ev与指标earnings_per_share之间的相关性大小为0.196033
指标ev与指标revenue之间的相关性大小为0.224363
指标ev与指标total_expense之间的相关性大小为0.149857
指标earnings_per_share与指标revenue之间的相关性大小为0.141473
指标earnings_per_share与指标total_expense之间的相关性大小为0.105022
指标revenue与指标total_expense之间的相关性大小为0.995845
```

从中我们得出

- 指标revenue与指标total_expense之间的相关性大小为0.995845
- 指标return_on_asset_net_profit与指标du_return_on_equity之间的相关性大小为0.818697

我们也可以通过画图来观察结果

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 8), dpi=100)
plt.scatter(data['revenue'], data['total_expense'])
plt.show()
```



这两对指标之间的相关性较大，可以做之后的处理，比如合成这两个指标。

2.6 主成分分析

学习目标

- 目标
  - 应用PCA实现特征的降维
- 应用
  - 用户与物品类别之间主成分分析

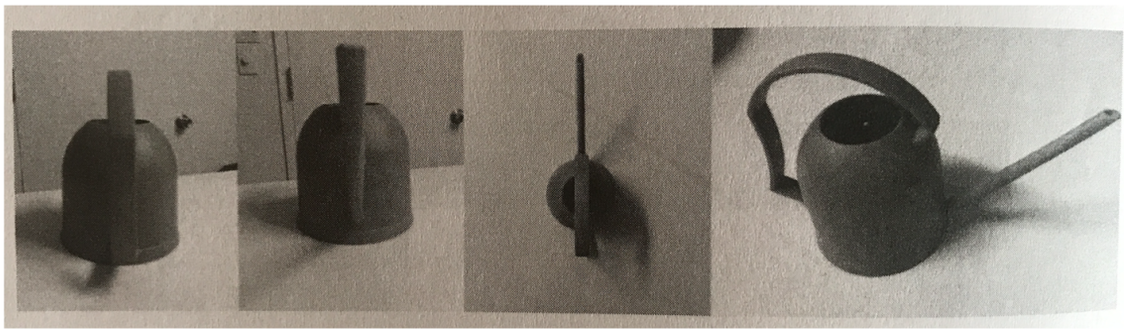
2.6.1 什么是主成分分析(PCA)

- 定义：**高维数据转化为低维数据的过程**，在此过程中**可能会舍弃原有数据、创造新的变量**
- 作用：**是数据维数压缩，尽可能降低原数据的维数（复杂度），损失少量信息。**
- 应用：回归分析或者聚类分析当中

> 对于信息一词，在决策树中会进行介绍

那么更好的理解这个过程呢？我们来看一张图

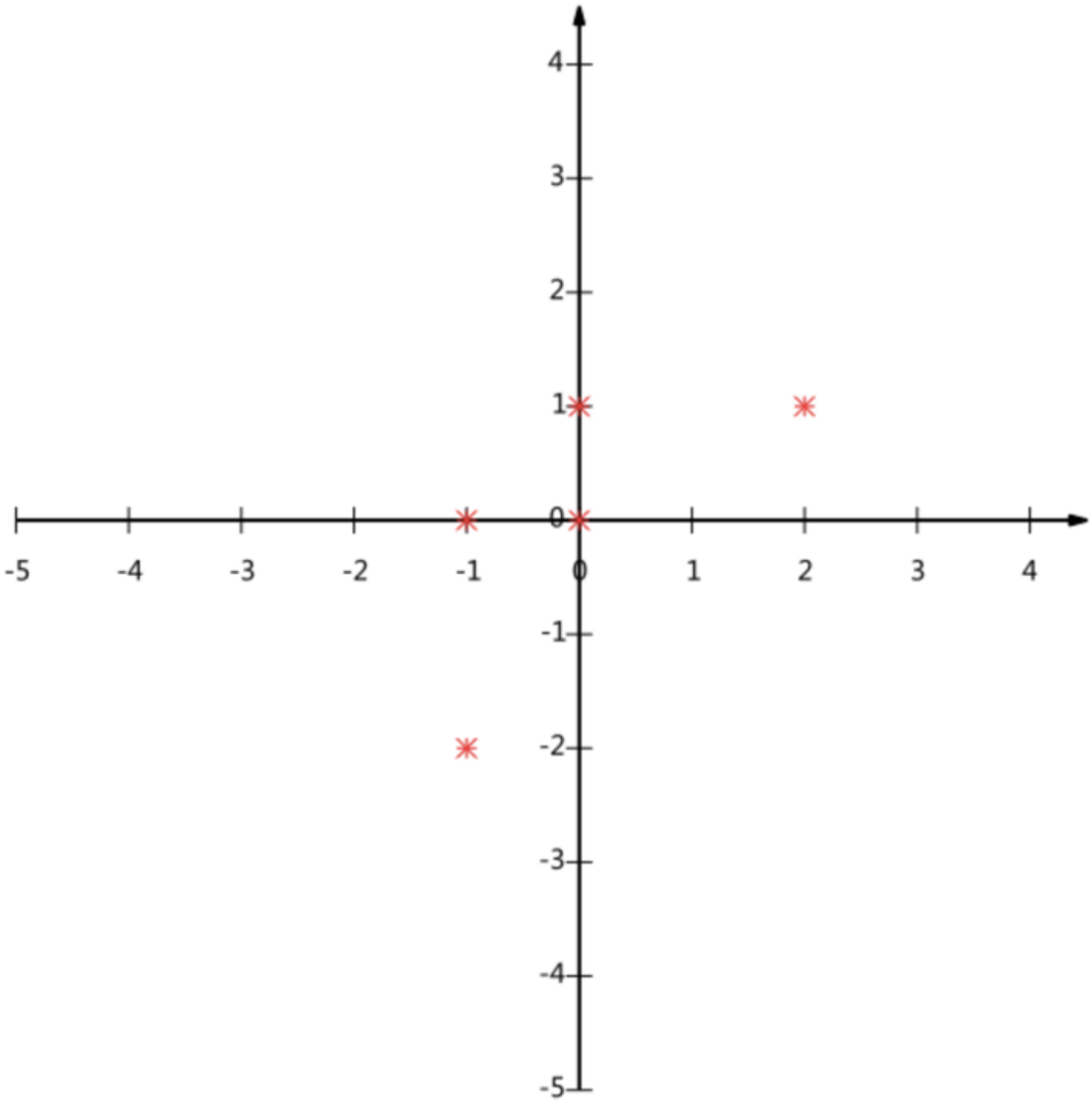
如何最好的对一个立体的物体二维表示



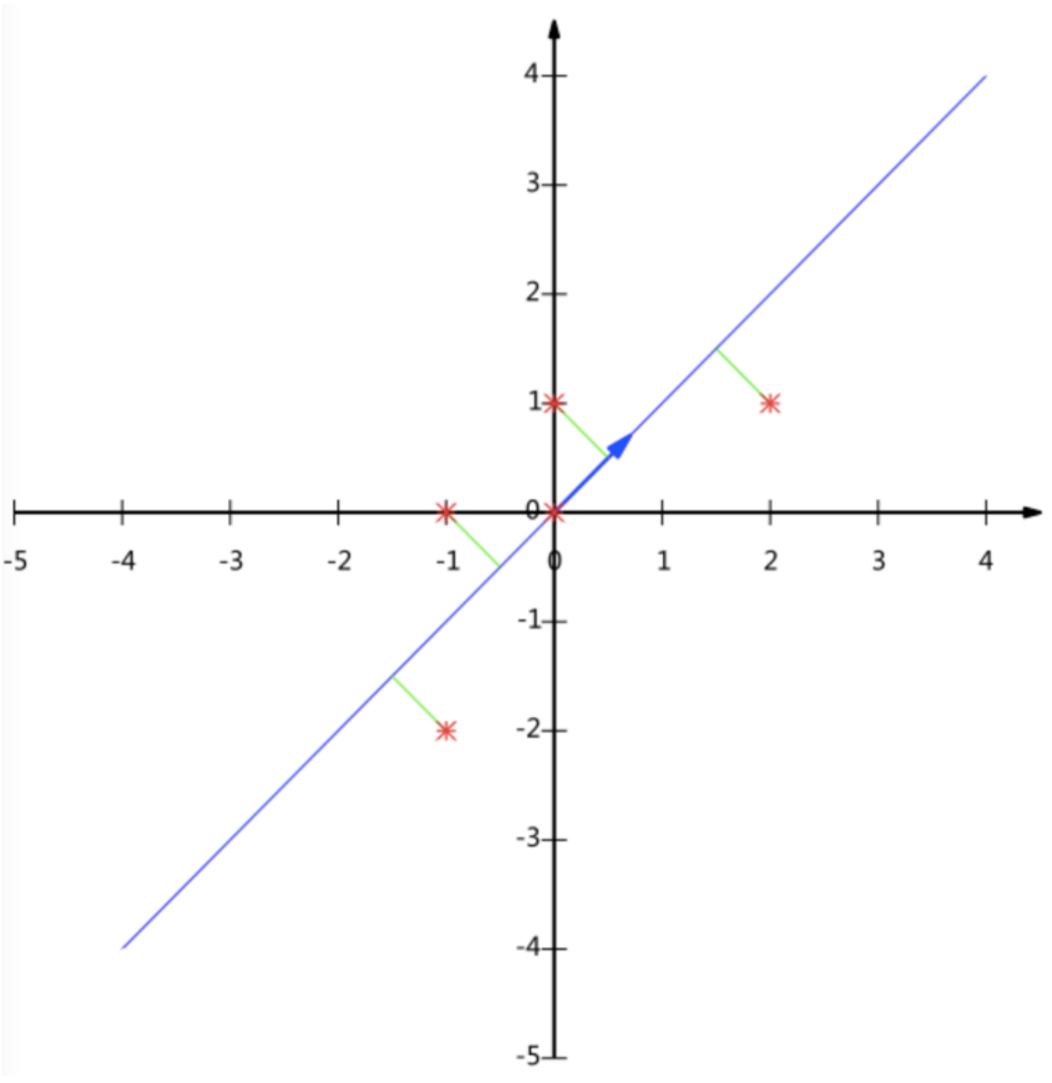
1 计算案例理解(了解，无需记忆)

假设对于给定5个点，数据如下

```
(-1,-2)
(-1, 0)
( 0, 0)
( 2, 1)
( 0, 1)
```



要求：将这个二维的数据简化成一维？ 并且损失少量的信息



这个过程如何计算的呢？**找到一个合适的直线，通过一个矩阵运算得出主成分分析的结果（不需要理解）**

$Y = PX$ 即为降维到k维后的数据

- 矩阵运算得出P为 $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$

$$Y = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -3/\sqrt{2} & -1/\sqrt{2} & 0 & 3/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$$

2 API

- sklearn.decomposition.PCA(n_components=None)
  - 将数据分解为较低维数空间
  - n_components:
    - **小数：表示保留百分之多少的信息**
    - **整数：减少到多少特征**
  - PCA.fit_transform(X) X:numpy array格式的数据[n_samples,n_features]

- 返回值：转换后指定维度的array

3 数据计算

先拿个简单的数据计算一下

```
[[2,8,4,5],
[6,3,0,8],
[5,4,9,1]]
from sklearn.decomposition import PCA

def pca_demo():
    """
    对数据进行PCA降维
    :return: None
    """
    data = [[2,8,4,5], [6,3,0,8], [5,4,9,1]]

    # 1、实例化PCA, 小数--保留多少信息
    transfer = PCA(n_components=0.9)
    # 2、调用fit_transform
    data1 = transfer.fit_transform(data)

    print("保留90%的信息，降维结果为：\n", data1)

    # 1、实例化PCA, 整数--指定降维到的维数
    transfer2 = PCA(n_components=3)
    # 2、调用fit_transform
    data2 = transfer2.fit_transform(data)
    print("降维到3维的结果：\n", data2)

    return None
```

返回结果：

```
保留90%的信息，降维结果为：
 [[ -3.13587302e-16   3.82970843e+00]
 [ -5.74456265e+00  -1.91485422e+00]
 [  5.74456265e+00  -1.91485422e+00]]
降维到3维的结果：
 [[ -3.13587302e-16   3.82970843e+00   4.59544715e-16]
 [ -5.74456265e+00  -1.91485422e+00   4.59544715e-16]
 [  5.74456265e+00  -1.91485422e+00   4.59544715e-16]]
```

2.6.2 案例：探究用户对物品类别的喜好细分降维

数据如下：

- order_products__prior.csv：订单与商品信息
  - 字段：**order_id**, **product_id**, add_to_cart_order, reordered
- products.csv：商品信息
  - 字段：**product_id**, product_name, **aisle_id**, department_id
- orders.csv：用户的订单信息
  - 字段：**order_id**,**user_id**,eval_set,order_number,....
- aisles.csv：商品所属具体物品类别
  - 字段： **aisle_id**, **aisle**

1 需求

aisle	air fresheners candles	asian foods	baby accessories	baby bath body care	baby food formula	bakery desserts	baking ingredients	baking supplies decor	beauty	beers coolers	...	spreads	tea	tofu meat alternatives	tortillas flat bread	trail mix snack mix	trash bags liners
user_id																	
1	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0
2	0	3	0	0	0	0	2	0	0	0	...	3	1	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	4	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0
5	0	2	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

5 rows × 134 columns

```
pd.DataFrame(data).head()
```

	0	1	2	3	4	5	6	7	8	9	...	34	35	36	37
0	-24.215659	2.429427	-2.466370	-0.145686	0.269042	-1.432932	2.140677	-2.738031	-2.714316	-1.743135	...	-3.046479	-1.464978	-0.241394	-3.304141
1	6.463208	36.751116	8.382553	15.097530	-6.920938	-0.978375	6.011567	3.787725	-8.180749	-9.040861	...	-1.025366	1.044537	3.417813	2.075166
2	-7.990302	2.404383	-11.030064	0.672230	-0.442368	-2.823272	-6.284140	6.512509	-2.148634	-1.585257	...	-1.232934	-2.014945	-0.180033	-0.838554
3	-27.991129	-0.755823	-1.921732	2.091888	-0.288232	0.926177	0.827127	0.614849	0.037820	-0.890672	...	0.475322	-0.518777	-0.458688	0.042819
4	-19.896394	-2.637225	0.533229	3.679228	0.612825	-1.624008	-3.935771	2.004627	1.002090	3.085747	...	-0.515761	0.581325	0.402795	-0.210034

5 rows × 44 columns

2 分析

- 合并表，使得**user_id**与**aisle**在一张表当中
- 进行交叉表变换
- 进行降维

3 完整代码

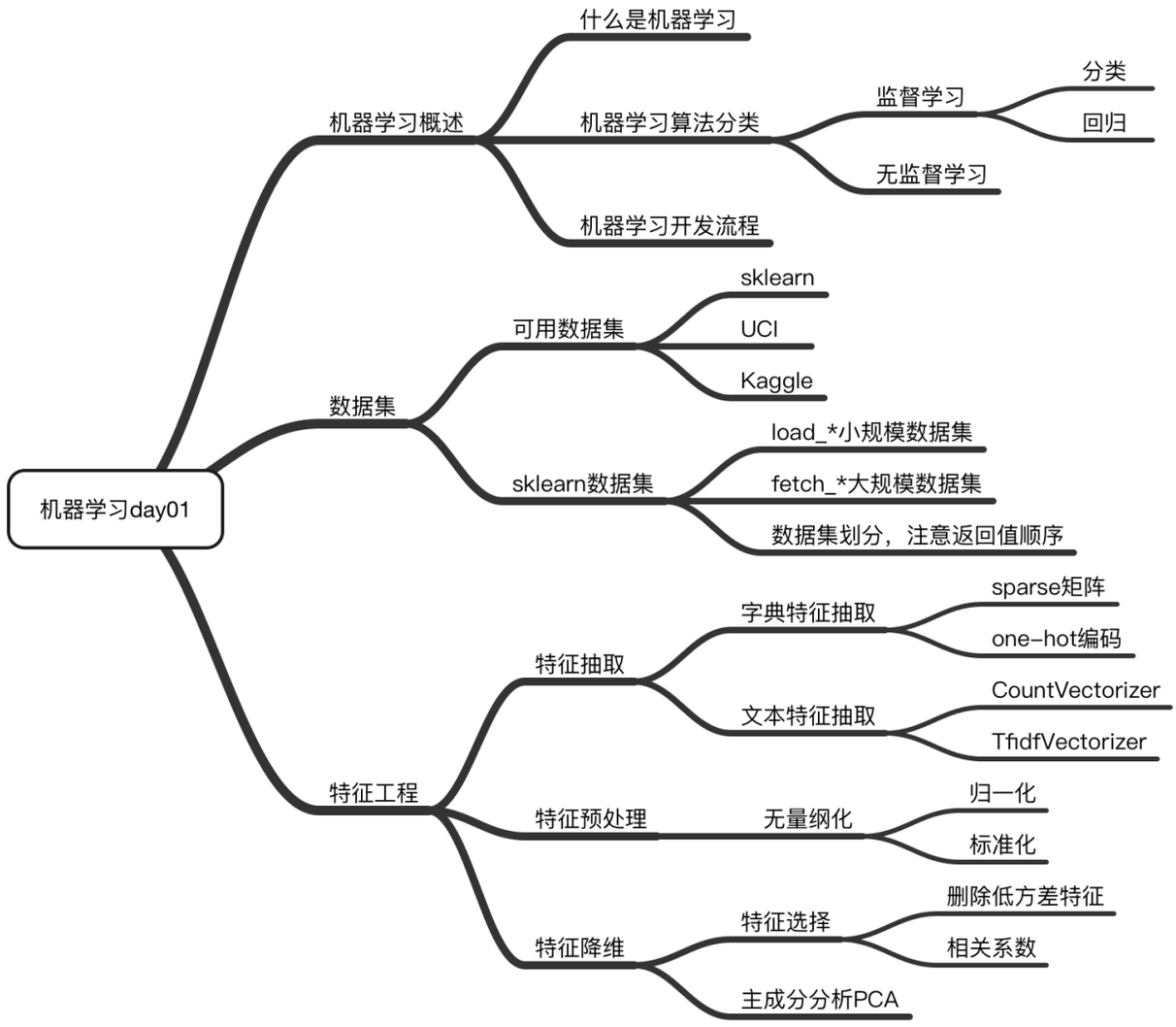
```
import pandas as pd
from sklearn.decomposition import PCA

# 1、获取数据集
# ·商品信息- products.csv：
# Fields：product_id, product_name, aisle_id, department_id
# ·订单与商品信息- order_products__prior.csv：
# Fields：order_id, product_id, add_to_cart_order, reordered
# ·用户的订单信息- orders.csv：
# Fields：order_id, user_id,eval_set, order_number,order_dow, order_hour_of_day,
days_since_prior_order
# ·商品所属具体物品类别- aisles.csv：
# Fields：aisle_id, aisle
products = pd.read_csv("./instacart/products.csv")
order_products = pd.read_csv("./instacart/order_products__prior.csv")
orders = pd.read_csv("./instacart/orders.csv")
aisles = pd.read_csv("./instacart/aisles.csv")
```

```
# 2、合并表，将user_id和aisle放在一张表上
# 1）合并orders和order_products on=order_id tab1:order_id, product_id, user_id
tab1 = pd.merge(orders, order_products, on=["order_id", "order_id"])
# 2）合并tab1和products on=product_id tab2:aisle_id
tab2 = pd.merge(tab1, products, on=["product_id", "product_id"])
# 3）合并tab2和aisles on=aisle_id tab3:user_id, aisle
tab3 = pd.merge(tab2, aisles, on=["aisle_id", "aisle_id"])

# 3、交叉表处理，把user_id和aisle进行分组
table = pd.crosstab(tab3["user_id"], tab3["aisle"])

# 4、主成分分析的方法进行降维
# 1）实例化一个转换器类PCA
transfer = PCA(n_components=0.95)
# 2）fit_transform
data = transfer.fit_transform(table)

data.shape
```

返回结果：

```
(206209, 44)
```

总结



每日作业

1、数据集的结构是什么?

答案: 特征值+ 目标值

2、机器学习算法分成哪些类别? 如何分类

答案: 根据是否有目标值分为 监督学习和非监督学习监督学习

根据目标值的数据类型:目标值为离散值就是分类问题

目标值为连续值就是回归问题

3、什么是标准化? 和归一化相比有什么优点?

答案: 标准化是通过对原始数据进行变换把数据变换到均值为0,方差为1范围内

优点: 少量异常点, 不影响平均值和方差, 对转换影响小

机器学习概述

> 了解机器学习定义以及应用场景
> 说明机器学习算法监督学习与无监督学习的区别
> 说明监督学习中的分类、回归特点
> 说明机器学习算法目标值的两种数据类型
> 说明机器学习(数据挖掘)的开发流程

数据集介绍与划分

学习目标

- 目标
  - 知道数据集的分为训练集和测试集
  - 知道sklearn的分类、回归数据集
- 应用
  - 无

拿到的数据是否全部都用来训练一个模型?

1、 数据集的划分

机器学习一般的数据集会划分为两个部分:

- 训练数据:用于训练,构建模型
- 测试数据:在模型检验时使用,用于评估模型是否有效

划分比例:

- 训练集:70% 80% 75%
- 测试集:30% 20% 30%

API

- sklearn.model_selection.train_test_split(

  arrays, *

  options)
  - x 数据集的特征值
  - y 数据集的标签值
  - test_size 测试集的大小,一般为float
  - random_state 随机数种子,不同的种子会造成不同的随机采样结果。相同的种子采样结果相同。
  - return ,测试集特征训练集特征值值,训练标签,测试标签(默认随机取)

结合后面的数据集作介绍

2、sklearn数据集介绍

2.1 API

- sklearn.datasets
  - 加载获取流行数据集
  - datasets.load_*()
    - 获取小规模数据集，数据包含在datasets里
  - datasets.fetch_*(data_home=None)
    - 获取大规模数据集，需要从网络上下载，函数的第一个参数是data_home，表示数据集下载的目录,默认是 ~/scikit_learn_data/

2.2分类和回归数据集

- 分类数据集

sklearn.datasets.load_iris()
加载并返回鸢尾花数据集

名称	数量
类别	3
特征	4
样本数量	150
每个类别数量	50

sklearn.datasets.load_digits()
加载并返回数字数据集

名称	数量
类别	10
特征	64
样本数量	1797

- sklearn.datasets.fetch_20newsgroups(data_home=None,subset='train')
  - subset: 'train'或者'test','all'，可选，选择要加载的数据集.训练集的“训练”，测试集的“测试”，两者的“全部”
- 回归数据集

sklearn.datasets.load_boston()

加载并返回波士顿房价数据集

名称	数量
目标类别	5-50
特征	13
样本数量	506

sklearn.datasets.load_diabetes()

加载和返回糖尿病数据集

名称	数量
目标范围	25-346
特征	10
样本数量	442

2.3 返回类型

- load

  和fetch

  返回的数据类型datasets.base.Bunch(字典格式)

  - data：特征数据数组，是 [n_samples * n_features] 的二维 numpy.ndarray 数组
  - target：标签数组，是 n_samples 的一维 numpy.ndarray 数组
  - DESCR：数据描述
  - feature_names：特征名,新闻数据，手写数字、回归数据集没有
  - target_names：标签名

sklearn转换器和估计器

学习目标

- 目标
  - 知道sklearn的转换器和估计器流程
- 应用
  - 无

1、转换器和估计器

1.1 转换器

想一下之前做的特征工程的步骤?

- 1、实例化 (实例化的是一个转换器类(Transformer))
- 2、调用fit_transform(对于文档建立分类词频矩阵，不能同时调用)

我们把特征工程的接口称之为转换器，其中转换器调用有这么几种形式

- fit_transform
- fit

- transform

这几个方法之间的区别是什么呢？我们看以下代码就清楚了

```
In [1]: from sklearn.preprocessing import StandardScaler

In [2]: std1 = StandardScaler()

In [3]: a = [[1,2,3], [4,5,6]]

In [4]: std1.fit_transform(a)
Out[4]:
array([[-1., -1., -1.],
       [ 1.,  1.,  1.]])

In [5]: std2 = StandardScaler()

In [6]: std2.fit(a)
Out[6]: StandardScaler(copy=True, with_mean=True, with_std=True)

In [7]: std2.transform(a)
Out[7]:
array([[-1., -1., -1.],
       [ 1.,  1.,  1.]])
```

从中可以看出，fit_transform的作用相当于transform加上fit。但是为什么还要提供单独的fit呢, 我们还是使用原来的std2来进行标准化看看

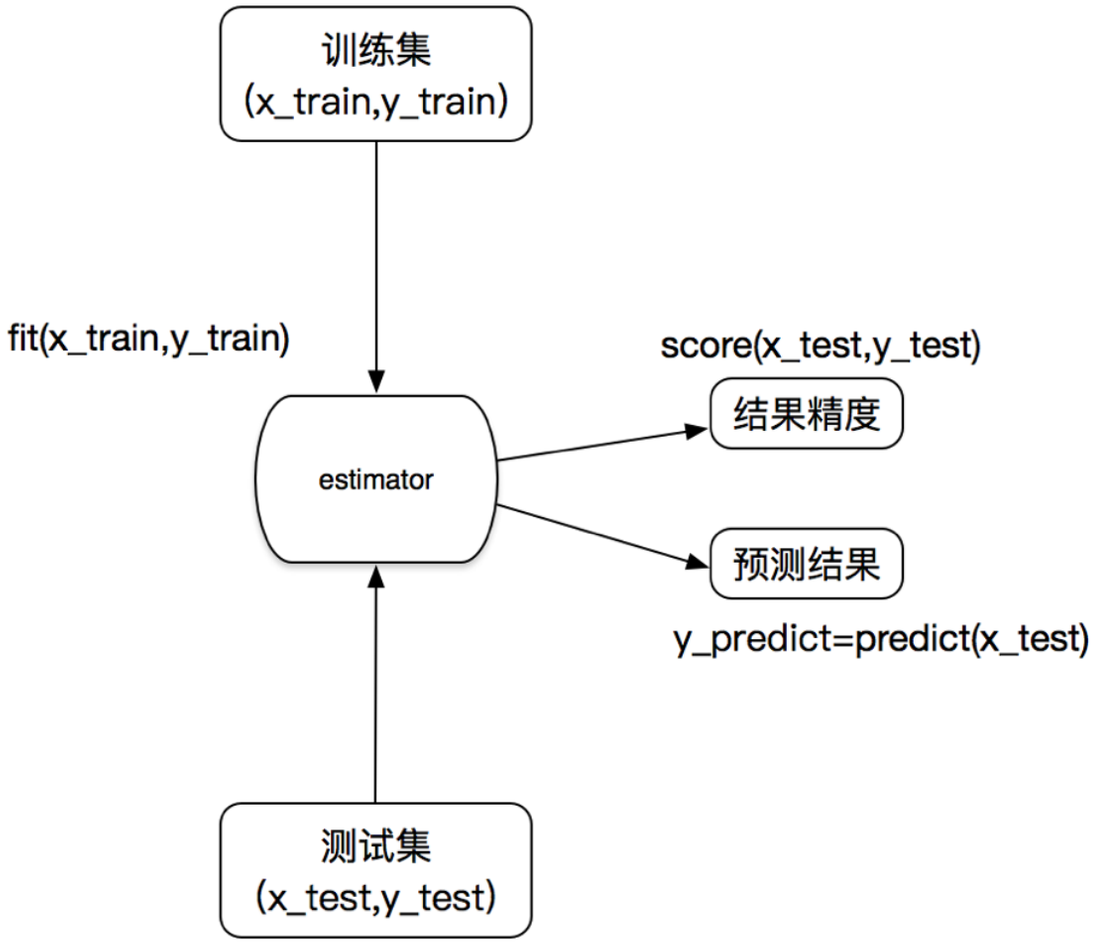
```
In [8]: b = [[7,8,9], [10, 11, 12]]

In [9]: std2.transform(b)
Out[9]:
array([[3., 3., 3.],
       [5., 5., 5.]])

In [10]: std2.fit_transform(b)
Out[10]:
array([[-1., -1., -1.],
       [ 1.,  1.,  1.]])
```

1.2 估计器(sklearn机器学习算法的实现)

在sklearn中，估计器(estimator)是一个重要的角色，是一类实现了算法的API

- 1、用于分类的估计器：
  - sklearn.neighbors k-近邻算法
  - sklearn.naive_bayes 贝叶斯
  - sklearn.linear_model.LogisticRegression 逻辑回归
  - sklearn.tree 决策树与随机森林
- 2、用于回归的估计器：
  - sklearn.linear_model.LinearRegression 线性回归
  - sklearn.linear_model.Ridge 岭回归
- 3、用于无监督学习的估计器
  - sklearn.cluster.KMeans 聚类

1.3 估计器工作流程



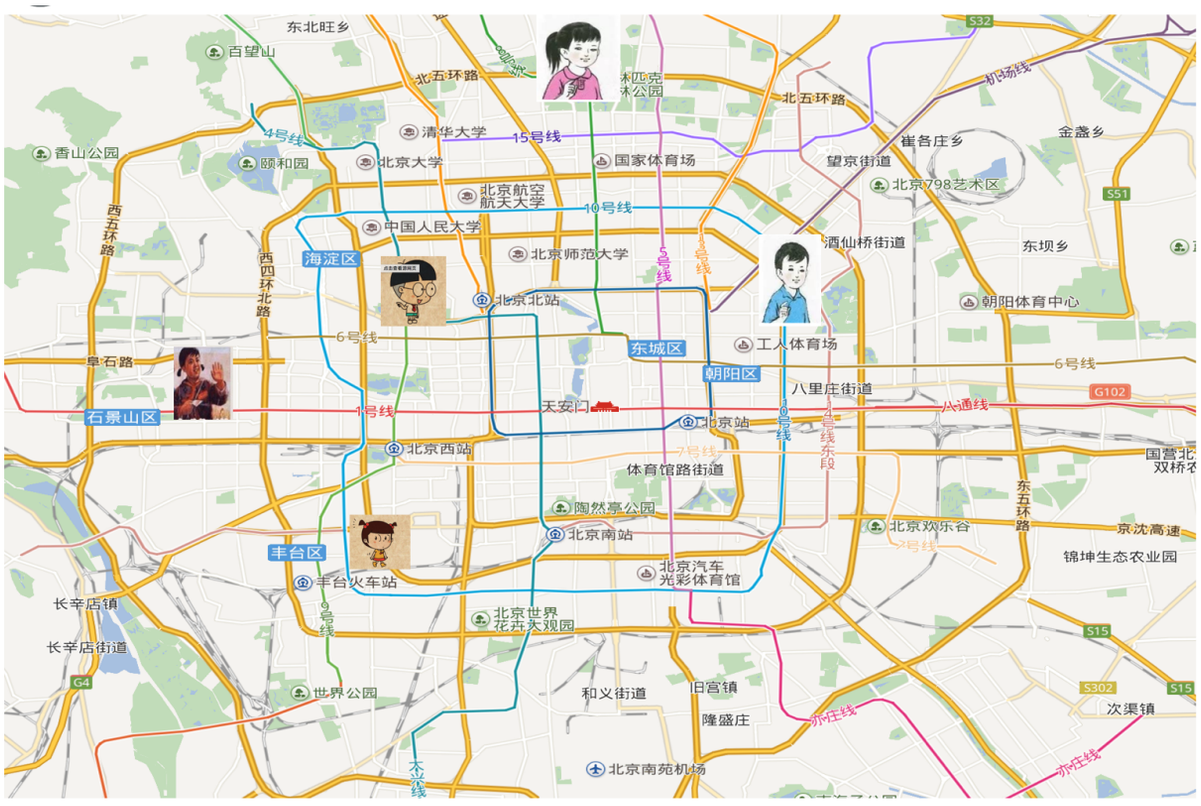
K-近邻算法

学习目标

- 目标
  - 说明K-近邻算法的距离公式
  - 说明K-近邻算法的超参数K值以及取值问题
  - 说明K-近邻算法的优缺点
  - 应用KNeighborsClassifier实现分类
  - 了解分类算法的评估标准准确率
- 应用
  - Facebook签到位置预测

问题：回忆分类问题的判定方法

什么是K-近邻算法



- 你的“邻居”来推断出你的类别

1、K-近邻算法(KNN)

1.1 定义

如果一个样本在特征空间中的**k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别**，则该样本也属于这个类别。

> 来源：KNN算法最早是由Cover和Hart提出的一种分类算法

1.2 距离公式

两个样本的距离可以通过如下公式计算，又叫欧式距离

比如说，a(a1,a2,a3),b(b1,b2,b3)

$$\sqrt{(a1-b1)^2+(a2-b2)^2+(a3-b3)^2}$$

2、电影类型分析

假设我们有现在几部电影

电影名称	打斗镜头	接吻镜头	电影类型
California Man	3	104	爱情片
He's not Really into dues	2	100	爱情片
Beautiful Woman	1	81	爱情片
Kevin Longblade	101	10	动作片
Robo Slayer 3000	99	5	动作片
Amped II	98	2	动作片
?	18	90	未知

其中？ 号电影不知道类别，如何去预测？我们可以利用K近邻算法的思想

电影名称	与未知电影的距离
California Man	20.5
He's not Really into dues	18.7
Beautiful Woman	19.2
Kevin Longblade	115.3
Robo Slayer 3000	117.4
Amped II	118.9

2.1 问题

- 如果取的最近的电影数量不一样？会是什么结果？

2.2 K-近邻算法数据的特征工程处理

- 结合前面的约会对象数据，分析K-近邻算法需要做什么样的处理

3、 K-近邻算法API

- sklearn.neighbors.KNeighborsClassifier(n_neighbors=5,algorithm='auto')
  - n_neighbors：int,可选（默认= 5），k_neighbors查询默认使用的邻居数
  - algorithm：{‘auto’，‘ball_tree’，‘kd_tree’，‘brute’}，可选用于计算最近邻居的算法：‘ball_tree’将会使用 BallTree，‘kd_tree’将使用 KDTree。‘auto’将尝试根据传递给fit方法的值来决定最合适的算法。 (不同实现方式影响效率)

4、 案例：预测签到位置

本次大赛的目的是预测一个人想签入到哪个地方。对于本次比赛的目的，Facebook的创建一个人造的世界，包括位于10公里的10平方公里超过10万米的地方。对于一个给定的坐标，你的任务是返回最有可能的地方的排名列表。数据制作出类似于来自移动设备的位置的信号，给你需要什么与不准确的，嘈杂的价值观复杂的真实数据工作一番风味。不一致的和错误的位置数据可能破坏，如Facebook入住服务经验。

File descriptions

- **train.csv, test.csv**
  - row_id: id of the check-in event
  - x y: coordinates
  - accuracy: location accuracy
  - time: timestamp
  - place_id: id of the business, **this is the target you are predicting**

特征值

目标值

数据介绍：将根据用户的位置，准确性和时间戳预测用户正在查看的业务。

```
train.csv，test.csv
row_id：登记事件的ID
xy：坐标
准确性：定位准确性
时间：时间戳
place_id：业务的ID，这是您预测的目标
```

> 官网：https://www.kaggle.com/navoshta/grid-knn/data

4.1 分析

- 对于数据做一些基本处理（这里所做的一些处理不一定达到很好的效果，我们只是简单尝试，有些特征我们可以根据一些特征选择的方式去做处理）
  - 1、缩小数据集范围 DataFrame.query()
  - 4、删除没用的日期数据 DataFrame.drop（可以选择保留）
  - 5、将签到位置少于n个用户的删除

    place_count = data.groupby('place_id').count()

    tf = place_count[place_count.row_id > 3].reset_index()

    data = data[data['place_id'].isin(tf.place_id)]
- 分割数据集
- 标准化处理
- k-近邻预测

4.2 代码

```
def knncls():
    """
    K近邻算法预测入住位置类别
    :return:
    """
    # 一、处理数据以及特征工程
    # 1、读取收，缩小数据的范围
    data = pd.read_csv("./data/FBlocation/train.csv")

    # 数据逻辑筛选操作 df.query()
    data = data.query("x > 1.0 & x < 1.25 & y > 2.5 & y < 2.75")

    # 删除time这一列特征
    data = data.drop(['time'], axis=1)

    print(data)

    # 删除入住次数少于三次位置
    place_count = data.groupby('place_id').count()

    tf = place_count[place_count.row_id > 3].reset_index()

    data = data[data['place_id'].isin(tf.place_id)]

    # 3、取出特征值和目标值
    y = data['place_id']
    # y = data[['place_id']]

    x = data.drop(['place_id', 'row_id'], axis=1)

    # 4、数据分割与特征工程?

    # （1）、数据分割
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    # (2)、标准化
    std = StandardScaler()

    # 队训练集进行标准化操作
    x_train = std.fit_transform(x_train)
    print(x_train)

    # 进行测试集的标准化操作
    x_test = std.fit_transform(x_test)

    # 二、算法的输入训练预测
    # K值：算法传入参数不定的值      理论上：k = 根号(样本数)
    # K值：后面会使用参数调优方法，去轮流试出最好的参数[1,3,5,10,20,100,200]
    knn = KNeighborsClassifier(n_neighbors=1)

    # 调用fit()
    knn.fit(x_train, y_train)

    # 预测测试数据集，得出准确率
    y_predict = knn.predict(x_test)
```

```
    print("预测测试集类别：", y_predict)

    print("准确率为：", knn.score(x_test, y_test))

    return None
```

4.3 结果分析

准确率： 分类算法的评估之一

- 1、k值取多大？有什么影响？

k值取很小：容易受到异常点的影响

k值取很大：受到样本均衡的问题

- 2、性能问题？

距离计算上面，时间复杂度高

5、K-近邻总结

- 优点：
  - 简单，易于理解，易于实现，无需训练
- 缺点：
  - 懒惰算法，对测试样本分类时的计算量大，内存开销大
  - 必须指定K值，K值选择不当则分类精度不能保证
- 使用场景：小数据场景，几千～几万样本，具体场景具体业务去测试

模型选择与调优

学习目标

- 目标
  - 说明交叉验证过程
  - 说明超参数搜索过程
  - 应用GridSearchCV实现算法参数的调优
- 应用
  - Facebook签到位置预测调优

1、为什么需要交叉验证

交叉验证目的：**为了让被评估的模型更加准确可信**

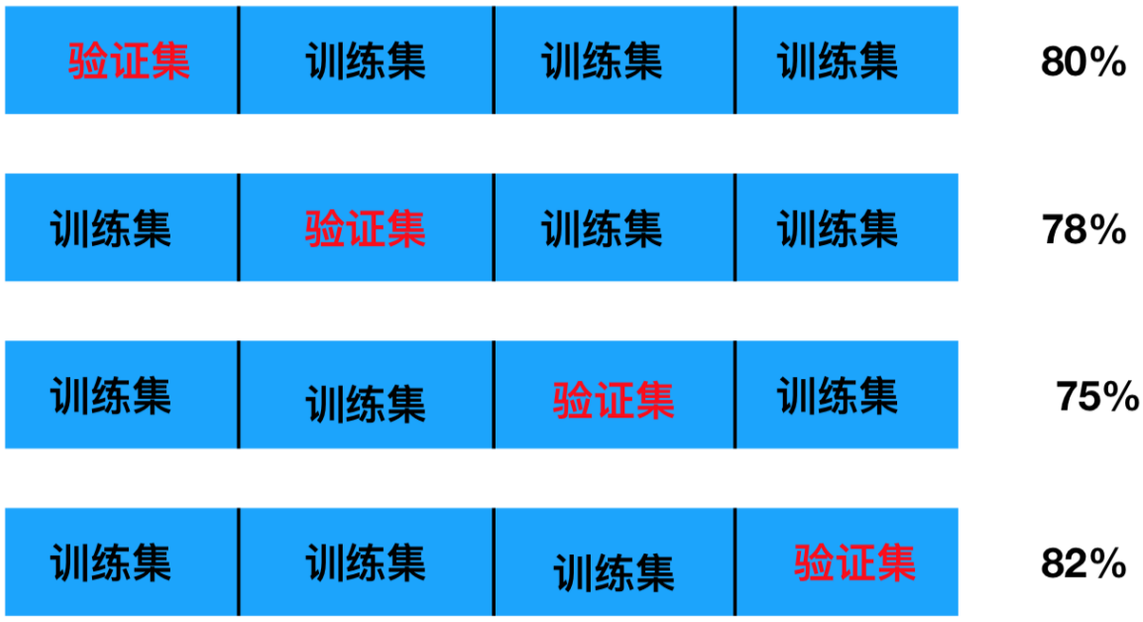
2、什么是交叉验证(cross validation)

交叉验证：将拿到的训练数据，分为训练和验证集。以下图为例：将数据分成5份，其中一份作为验证集。然后经过5次(组)的测试，每次都更换不同的验证集。即得到5组模型的结果，取平均值作为最终结果。又称5折交叉验证。

2.1 分析

我们之前知道数据分为训练集和测试集，但是**为了让从训练得到模型结果更加准确。**做以下处理

- 训练集：训练集+验证集
- 测试集：测试集



问题：那么这个只是对于参数得出更好的结果，那么怎么选择或者调优参数呢?

3、超参数搜索-网格搜索(Grid Search)

通常情况下，**有很多参数是需要手动指定的（如k-近邻算法中的K值），这种叫超参数。**但是手动过程繁杂，所以需要对模型预设几种超参数组合。**每组超参数都采用交叉验证来进行评估。最后选出最优参数组合建立模型。**

K值	K=3	K=5	K=7
模型	模型1	模型2	模型3

3.1 模型选择与调优

- sklearn.model_selection.GridSearchCV(estimator, param_grid=None,cv=None)
  - 对估计器的指定参数值进行详尽搜索
  - estimator：估计器对象
  - param_grid：估计器参数(dict){“n_neighbors”:[1,3,5]}
  - cv：指定几折交叉验证
  -
  - fit：输入训练数据
  - score：准确率
  - 结果分析：

- best*score*:在交叉验证中验证的最好结果_
- best*estimator*：最好的参数模型
- cv*results*:每次交叉验证后的验证集准确率结果和训练集准确率结果

4、 Facebook签到位置预测K值调优

- 使用网格搜索估计器

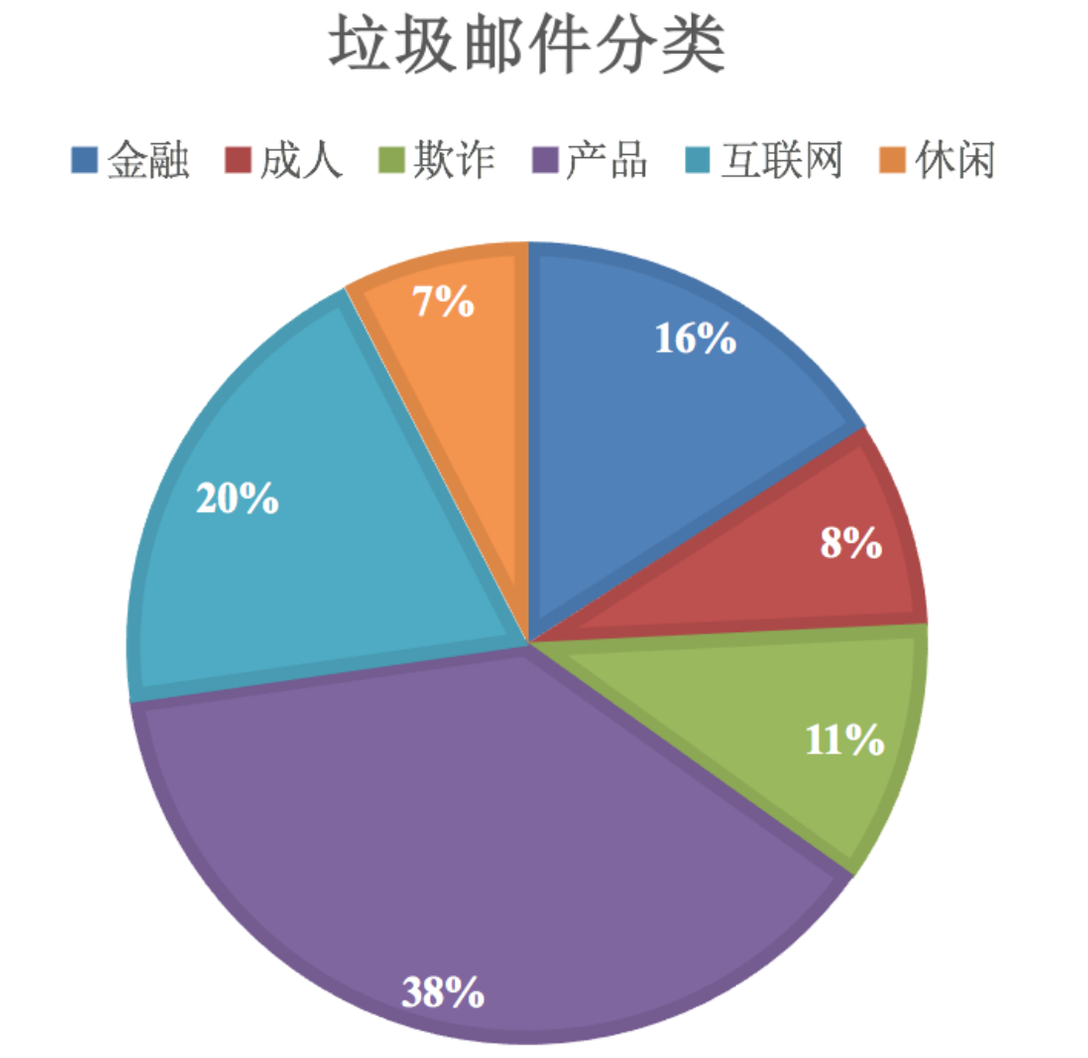
```
# 使用网格搜索和交叉验证找到合适的参数
knn = KNeighborsClassifier()

param = {"n_neighbors": [3, 5, 10]}

gc = GridSearchCV(knn, param_grid=param, cv=2)

gc.fit(x_train, y_train)

print("选择了某个模型测试集当中预测的准确率为：", gc.score(x_test, y_test))

# 训练验证集的结果
print("在交叉验证当中验证的最好结果：", gc.best_score_)
print("gc选择了的模型K值是：", gc.best_estimator_)
print("每次交叉验证的结果为：", gc.cv_results_)
```

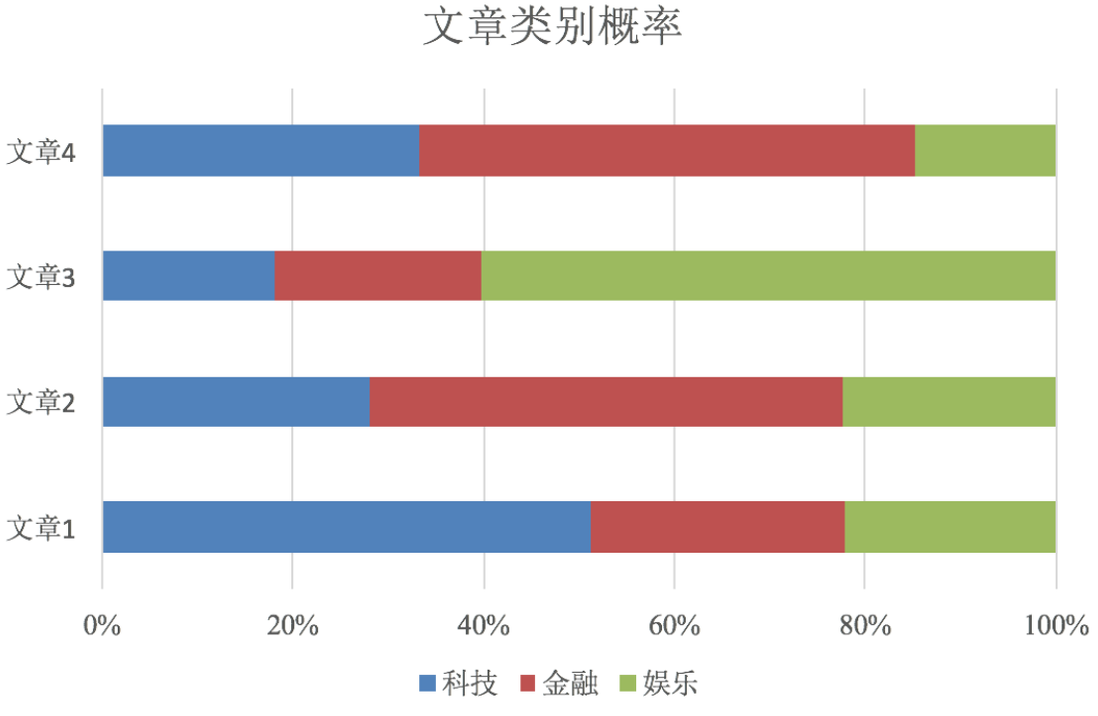
朴素贝叶斯算法

学习目标

- 目标
  - 说明条件概率与联合概率
  - 说明贝叶斯公式、以及特征独立的关系
  - 记忆贝叶斯公式
  - 知道拉普拉斯平滑系数
  - 应用贝叶斯公式实现概率的计算
- 应用
  - 20类新闻文章分类预测

1、 什么是朴素贝叶斯分类方法


垃圾邮件分类
金融
成人
欺诈
产品
互联网
休闲
16%
8%
11%
38%
20%
7%


文章类别概率
文章4
文章3
文章2
文章1
0%
20%
40%
60%
80%
100%
科技
金融
娱乐

2、 概率基础

2.1 概率(Probability)定义

- 概率定义为一件事情发生的可能性
  - 扔出一个硬币，结果头像朝上
  - 某天是晴天
- P(X) : 取值在[0, 1]

2.2 女神是否喜欢计算案例

在讲这两个概率之前我们通过一个例子，来计算一些结果：

样本数	职业	体型	女神是否喜欢
1	程序员	超重	不喜欢
2	产品	匀称	喜欢
3	程序员	匀称	喜欢
4	程序员	超重	喜欢
5	美工	匀称	不喜欢
6	美工	超重	不喜欢
7	产品	匀称	喜欢

- 问题如下：

问题

1、女神喜欢的概率？

2、职业是程序员并且体型匀称的概率？

3、在女神喜欢的条件下，职业是程序员的概率？

4、在女神喜欢的条件下，职业是产品，体重是超重的概率？

那么其中有些问题我们计算的结果不正确，或者不知道计算，我们有固定的公式去计算

2.3 条件概率与联合概率

- 联合概率：包含多个条件，且所有条件同时成立的概率
  - 记作：P(A,B)
  - 特性：P(A, B) = P(A)P(B)
- 条件概率：就是事件A在另外一个事件B已经发生条件下的发生概率
  - 记作：P(A|B)
  - 特性：P(A1,A2|B) = P(A1|B)P(A2|B)

> 注意：此条件概率的成立，**是由于A1,A2相互独立的结果**(记忆)

这样我们计算结果为：

```
p(程序员，匀称） =  P(程序员)P(匀称） =3/7*(4/7） = 12/49
P(产品，超重|喜欢） = P(产品|喜欢)P(超重|喜欢)=1/2 *  1/4 = 1/8
```

那么，我们知道了这些知识之后，继续回到我们的主题中。朴素贝叶斯如何分类，这个算法经常会用在文本分类，那就来看文章分类是一个什么样的问题？

P(科技|文章1)

P(娱乐|文章1)

这个了类似一个条件概率，那么仔细一想，给定文章其实相当于给定什么？结合前面我们将文本特征抽取的时候讲的？所以我们可以理解为

P(科技|文章1) =P(科技|词1，词2,词3,词4.....)

P(娱乐|文章1) =P(娱乐|词1，词2,词3,词4.....)

但是这个公式怎么求？前面并没有参考例子，其实是相似的，我们可以使用贝叶斯公式去计算

3、 贝叶斯公式

3.1 公式

$$P(C|W) = \frac{P(W|C)P(C)}{P(W)}$$

注：w为给定文档的特征值(频数统计,预测文档提供)，c为文档类别

那么这个公式如果应用在文章分类的场景当中，我们可以这样看：

公式可以理解为：

$$P(C|F1,F2,\dots) = \frac{P(F1,F2,\dots|C)P(C)}{P(F1,F2,\dots)}$$

其中c可以是不同类别

公式分为三个部分：

- P(C)：每个文档类别的概率(某文档类别数 / 总文档数量)
- P(W | C)：给定类别下特征（被预测文档中出现的词）的概率
  - 计算方法：P(F1 | C)=Ni/N （训练文档中去计算）
    - Ni为该F1词在C类别所有文档中出现的次数
    - N为所属类别C下的文档所有词出现的次数和
- P(F1,F2,...) 预测文档中每个词的概率

如果计算两个类别概率比较：

$$P(科技|文章1) = P(科技|文章1)P(科技) / P(文章1)$$

$$P(娱乐|文章1) = P(娱乐|文章1)P(娱乐) / P(文章1)$$

所以我们只要比较前面的大小就可以，得出谁的概率大

3.2 文章分类计算

- 假设我们从**训练数据集**得到如下信息

训练集统计结果(指定统计词频)：

特征\统计	科技(30篇)	娱乐(60篇)	汇总（90篇）
“商场”	9	51	60
“影院”	8	56	64
“支付宝”	20	15	35
“云计算”	63	0	63
汇总(求和)	100	121	221

现有一篇被预测文档：出现了影院，支付宝，云计算，计算属于科技、娱乐的类别概率？

- 计算结果

```
科技：P(科技|影院,支付宝,云计算) = P(影院,支付宝,云计算|科技)*P(科技)=
(8/100)*(20/100)*(63/100)*(30/90) = 0.00456109

娱乐：P(娱乐|影院,支付宝,云计算) = P(影院,支付宝,云计算|娱乐)*P(娱乐)=
(56/121)*(15/121)*(0/121)*(60/90) = 0
```

思考:我们计算出来某个概率为0，合适吗?

3.3 拉普拉斯平滑系数

目的：防止计算出的分类概率为0

$$P(F1|C) = \frac{Ni+\alpha}{N+\alpha m}$$

α为指定的系数一般为1，m为训练文档中统计出的特征词个数

```
P(娱乐|影院,支付宝,云计算) =P(影院,支付宝,云计算|娱乐)P(娱乐) =P(影院|娱乐)*P(支付宝|娱乐)*P(云计算|娱乐)P(娱乐)=(56+1/121+4)(15+1/121+4)(0+1/121+1*4)(60/90) = 0.00002
```

3.4 API

- sklearn.naive_bayes.MultinomialNB(alpha = 1.0)
  - 朴素贝叶斯分类
  - alpha：拉普拉斯平滑系数

4、案例：20类新闻分类

20个新闻组

20个新闻组数据集

20个新闻组数据集是大约20,000个新闻组文档的集合，平均分布在20个不同的新闻组中。据我所知，它最初是由Ken Lang收集的，可能是因为他的*Newsweeder：学习过滤网络新闻纸*，尽管他没有明确地提到这个集合。这20个新闻组集合已经成为机器学习技术的文本应用中的实验流行数据集，例如文本分类和文本聚类。

组织

数据被组织成20个不同的新闻组，每个新闻组对应不同的主题。一些新闻组彼此之间关系密切（例如**comp.sys.ibm.pc.hardware / comp.sys.mac.hardware**），而另一些新闻组则非常不相关（例如**misc.forsale / soc.religion.christian**）。以下是根据主题划分的20个新闻组列表（或多或少）：

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

4.1 分析

- 分割数据集
- tfidf进行的特征抽取
- 朴素贝叶斯预测

4.2 代码

```
def nbcls():
    """
    朴素贝叶斯对新闻数据集进行预测
    :return:
    """
    # 获取新闻的数据，20个类别
    news = fetch_20newsgroups(subset='all')

    # 进行数据集分割
    x_train, x_test, y_train, y_test = train_test_split(news.data, news.target,
test_size=0.3)
```

```
    # 对于文本数据，进行特征抽取
    tf = TfidfVectorizer()

    x_train = tf.fit_transform(x_train)
    # 这里打印出来的列表是：训练集当中的所有不同词的组成的一个列表
    print(tf.get_feature_names())
    # print(x_train.toarray())

    # 不能调用fit_transform
    x_test = tf.transform(x_test)

    # estimator估计器流程
    mlb = MultinomialNB(alpha=1.0)

    mlb.fit(x_train, y_train)

    # 进行预测
    y_predict = mlb.predict(x_test)

    print("预测每篇文章的类别：", y_predict[:100])
    print("真实类别为：", y_test[:100])

    print("预测准确率为：", mlb.score(x_test, y_test))

    return None
```

5、总结

- 优点：
  - 朴素贝叶斯模型发源于古典数学理论，有稳定的分类效率。
  - 对缺失数据不太敏感，算法也比较简单，常用于文本分类。
  - 分类准确度高，速度快
- 缺点：
  - 由于使用了样本属性独立性的假设，所以如果特征属性有关联时其效果不好

决策树

学习目标

- 目标
  - 说明信息熵的公式以及作用
  - 说明信息增益的公式作用
  - 应用信息增益实现计算特征的不确定性减少程度
  - 了解决策树的三种算法实现
- 应用
  - 泰坦尼克号乘客生存预测

1、认识决策树

决策树思想的来源非常朴素，程序设计中的条件分支结构就是if-then结构，最早的决策树就是利用这类结构分割数据的一种分类学习方法

怎么理解这句话？通过一个对话例子

比如：你母亲要给你介绍男朋友，是这么来对话的：

女儿：多大年纪了？

母亲：26。

女儿：长的帅不帅？

母亲：挺帅的。

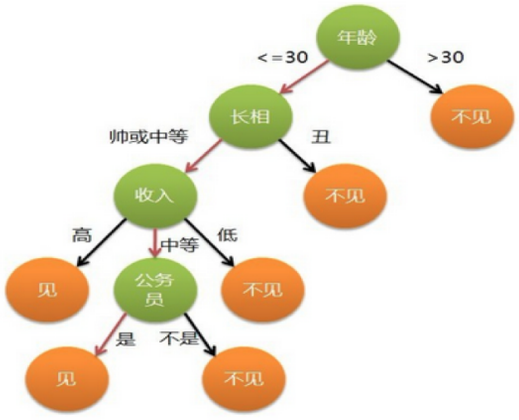
女儿：收入高不？

母亲：不算很高，中等情况。

女儿：是公务员不？

母亲：是，在税务局上班呢。

女儿：那好，我去见见。



想一想这个女生为什么把年龄放在最上面判断！！！！！！！！！

2、决策树分类原理详解

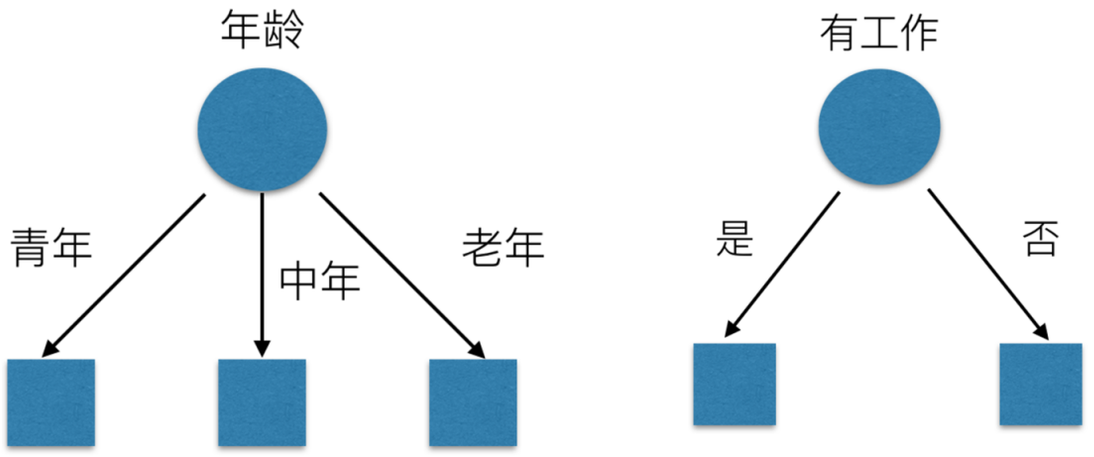
为了更好理解决策树具体怎么分类的，我们通过一个问题例子？

银行贷款数据

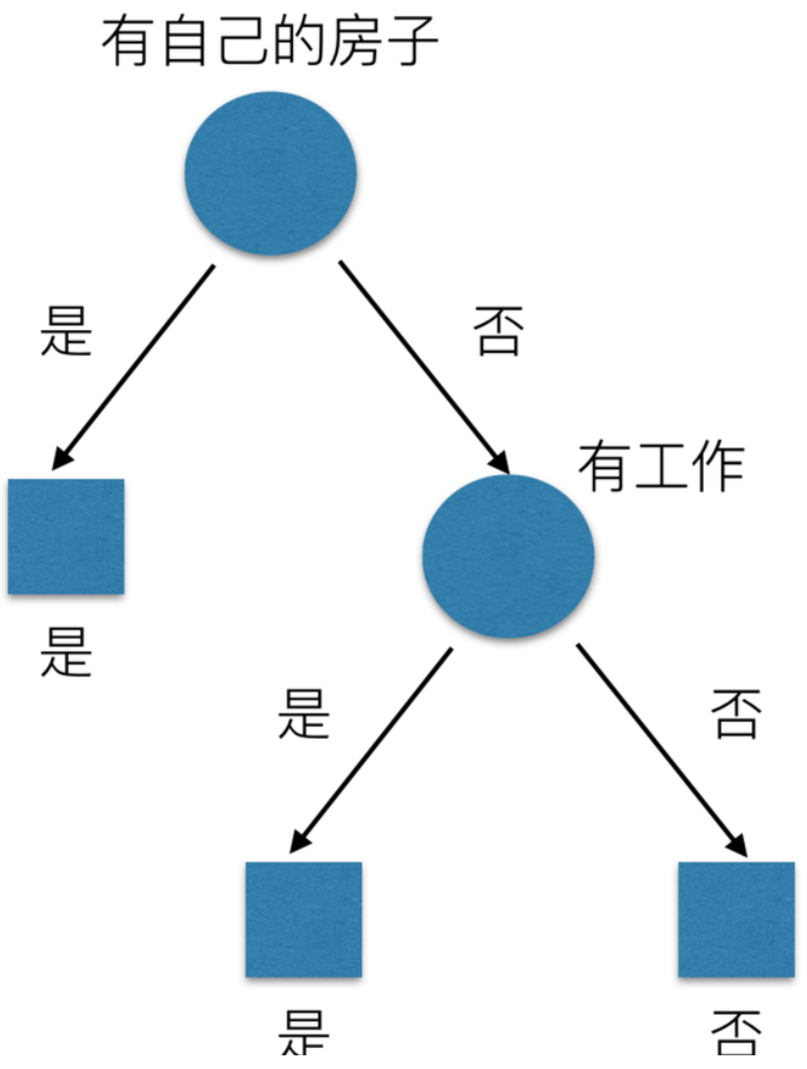
ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

问题：如何对这些客户进行分类预测？你是如何去划分？

有可能你的划分是这样的



那么我们怎么知道这些特征哪个更好放在最上面，那么决策树的真是划分是这样的



2.1 原理

- 信息熵、信息增益等

需要用到信息论的知识！！！问题：通过例子引入信息熵

2.2 信息熵

那来玩个猜测游戏，猜猜这32支球队那个是冠军。并且猜测错误付出代价。每猜错一次给一块钱，告诉我是否猜对了，那么我需要掏多少钱才能知道谁是冠军？ （前提是：不知道任意球队的信息、历史比赛记录、实力等）

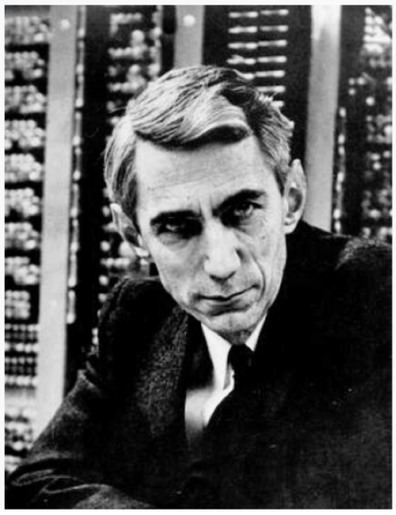
为了使代价最小，可以使用二分法猜测：

我可以把球编上号，从1到32，然后提问：冠 军在1-16号吗？依次询问，只需要五次，就可以知道结果。

我们来看这个式子：

- 32支球队，log32=5**比特**
- 64支球队，log64=6**比特**

克劳德·艾尔伍德·香农



信息论的创始人，香农是密歇根大学学士，麻省理工学院博士。1948年，香农发表了划时代的论文——通信的数学原理，奠定了现代信息论的基础

信息的单位：比特

香农指出，它的准确信息量应该是，p为每个球队获胜的概率（假设概率相等，都为1/32），我们不用钱去衡量这个代价了，香浓指出用比特：

```
H = -(p1logp1 + p2logp2 + ... + p32log32) = - log32
```

2.2.1 信息熵的定义

- H的专业术语称之为信息熵，单位为比特。

$$H(X)=\sum_{x\in X} P(x)\,logP(x)$$

“谁是世界杯冠军”的信息量应该比5比特少，特点（重要）：

- 当这32支球队夺冠的几率相同时，对应的信息熵等于5比特
- 只要概率发生任意变化，信息熵都比5比特大

2.2.2 总结（重要）

- 信息和消除不确定性是相联系的

当我们得到的额外信息（球队历史比赛情况等等）越多的话，那么我们猜测的代价越小（猜测的不确定性减小）

问题： 回到我们前面的贷款案例，怎么去划分？可以利用当得知某个特征（比如是否有房子）之后，我们能够减少的不确定性大小。越大我们可以认为这个特征很重要。那怎么去衡量减少的不确定性大小呢?

2.3 决策树的划分依据之一------信息增益

2.3.1 定义与公式

特征A对训练数据集D的信息增益g(D,A),定义为集合D的信息熵H(D)与特征A给定条件下D的信息条件熵H(D|A)之差，即公式为：

$$g(D,A)=H(D)-H(D|A)$$

公式的详细解释：

信息熵的计算：

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log\frac{|C_k|}{|D|}$$

条件熵的计算：

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log\frac{|D_{ik}|}{|D_i|}$$

注：C_k表示属于某个类别的样本数，

> 注：信息增益表示得知特征X的信息而息的不确定性减少的程度使得类Y的信息熵减少的程度

2.3.2 贷款特征重要计算

银行贷款数据

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

- 我们以年龄特征来计算：

```
1、g(D, 年龄) = H(D) -H(D|年龄) = 0.971-[5/15H(青年)+5/15H(中年)+5/15H(老年]

2、H(D) = -(6/15log(6/15)+9/15log(9/15))=0.971

3、H(青年) = -(3/5log(3/5) +2/5log(2/5))
H(中年)=-(3/5log(3/5) +2/5log(2/5))
H(老年)=-(4/5og(4/5)+1/5log(1/5))
```

我们以A1、A2、A3、A4代表年龄、有工作、有自己的房子和贷款情况。最终计算的结果g(D, A1) = 0.313, g(D, A2) = 0.324, g(D, A3) = 0.420,g(D, A4) = 0.363。所以我们选择A3 作为划分的第一个特征。这样我们就可以一棵树慢慢建立

2.4 决策树的三种算法实现

当然决策树的原理不止信息增益这一种，还有其他方法。但是原理都类似，我们就不去举例计算。

- ID3
  - 信息增益 最大的准则
- C4.5
  - 信息增益比 最大的准则
- CART
  - 分类树: 基尼系数 最小的准则 在sklearn中可以选择划分的默认原则
  - 优势：划分更加细致（从后面例子的树显示来理解）

2.5 决策树API

- class sklearn.tree.DecisionTreeClassifier(criterion='gini', max_depth=None,random_state=None)
  - 决策树分类器
  - criterion:默认是'gini'系数，也可以选择信息增益的熵'entropy'
  - max_depth:树的深度大小
  - random_state:随机数种子
- 其中会有些超参数：max_depth:树的深度大小
  - 其它超参数我们会结合随机森林讲解

3、案例：泰坦尼克号乘客生存预测

- 泰坦尼克号数据

在泰坦尼克号和titanic2数据帧描述泰坦尼克号上的个别乘客的生存状态。这里使用的数据集是由各种研究人员开始的。其中包括许多研究人员创建的旅客名单，由Michael A. Findlay编辑。我们提取的数据集中的特征是票的类别，存活，乘坐班，年龄，登陆，home.dest，房间，票，船和性别。

1、乘坐班是指乘客班（1，2，3），是社会经济阶层的代表。

2、其中age数据存在缺失。

> 数据：http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.txt


```
"row.names","pclass","survived","name","age","embarked","home.dest","room","ticket","boat","sex"
"1","1st",1,"Allen, Miss Elisabeth Walton",29.0000,"Southampton","St Louis, MO","B-5","24160 L221","2","female"
"2","1st",0,"Allison, Miss Helen Loraine", 2.0000,"Southampton","Montreal, PQ / Chesterville, ON","C26","","","female"
"3","1st",0,"Allison, Mr Hudson Joshua Creighton",30.0000,"Southampton","Montreal, PQ / Chesterville, ON","C26","","(135)","male"
"4","1st",0,"Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)",25.0000,"Southampton","Montreal, PQ / Chesterville, ON","C26","","","female"
"5","1st",1,"Allison, Master Hudson Trevor", 0.9167,"Southampton","Montreal, PQ / Chesterville, ON","C22","","11","male"
"6","1st",1,"Anderson, Mr Harry",47.0000,"Southampton","New York, NY","E-12","","3","male"
"7","1st",1,"Andrews, Miss Kornelia Theodosia",63.0000,"Southampton","Hudson, NY","D-7","13502 L77","10","female"
"8","1st",0,"Andrews, Mr Thomas, jr",39.0000,"Southampton","Belfast, NI","A-36","","","male"
"9","1st",1,"Appleton, Mrs Edward Dale (Charlotte Lamson)",58.0000,"Southampton","Bayside, Queens, NY","C-101","","2","female"
"10","1st",0,"Artagaveytia, Mr Ramon",71.0000,"Cherbourg","Montevideo, Uruguay","","","(22)","male"
"11","1st",0,"Astor, Colonel John Jacob",47.0000,"Cherbourg","New York, NY","","17754 L224 10s 6d","(124)","male"
"12","1st",1,"Astor, Mrs John Jacob (Madeleine Talmadge Force)",19.0000,"Cherbourg","New York, NY","","17754 L224 10s 6d","4","female"
"13","1st",1,"Aubert, Mrs Leontine Pauline",NA,"Cherbourg","Paris, France","B-35","17477 L69 6s","9","female"
"14","1st",1,"Barkworth, Mr Algernon H.",NA,"Southampton","Hessle, Yorks","A-23","","B","male"
"15","1st",0,"Baumann, Mr John D.",NA,"Southampton","New York, NY","","","","male"
"16","1st",1,"Baxter, Mrs James (Helene DeLaudeniere Chaput)",50.0000,"Cherbourg","Montreal, PQ","B-58/60","","6","female"
"17","1st",0,"Baxter, Mr Quigg Edmond",24.0000,"Cherbourg","Montreal, PQ","B-58/60","","","male"
"18","1st",0,"Beattie, Mr Thomson",36.0000,"Cherbourg","Winnipeg, MN","C-6","","","male"
```


3.1 分析

- 选择我们认为重要的几个特征 ['pclass', 'age', 'sex']
- 填充缺失值
- 特征中出现类别符号，需要进行one-hot编码处理(DictVectorizer)
  - x.to_dict(orient="records") 需要将数组特征转换成字典数据
- 数据集划分
- 决策树分类预测

3.2 代码

```
def decisioncls():
    """
    决策树进行乘客生存预测
    :return:
    """
    # 1、获取数据
    titan =
pd.read_csv("http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.txt
")
```

```
    # 2、数据的处理
    x = titan[['pclass', 'age', 'sex']]

    y = titan['survived']

    # print(x , y)
    # 缺失值需要处理，将特征当中有类别的这些特征进行字典特征抽取
    x['age'].fillna(x['age'].mean(), inplace=True)

    # 对于x转换成字典数据x.to_dict(orient="records")
    # [{"pclass": "1st", "age": 29.00, "sex": "female"}, {}]

    dict = DictVectorizer(sparse=False)

    x = dict.fit_transform(x.to_dict(orient="records"))

    print(dict.get_feature_names())
    print(x)

    # 分割训练集合测试集
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    # 进行决策树的建立和预测
    dc = DecisionTreeClassifier(max_depth=5)

    dc.fit(x_train, y_train)

    print("预测的准确率为：", dc.score(x_test, y_test))

    return None
```

由于决策树类似一个树的结构，我们可以保存到本地显示

3.3 保存树的结构到dot文件

- 1、sklearn.tree.export_graphviz() 该函数能够导出DOT格式
  - tree.export_graphviz(estimator,out_file='tree.dot',feature_names=['',''])
- 2、工具:(能够将dot文件转换为pdf、png)
  - 安装graphviz
  - ubuntu:sudo apt-get install graphviz Mac:brew install graphviz
- 3、运行命令
  - 然后我们运行这个命令
  - dot -Tpng tree.dot -o tree.png

```
export_graphviz(dc, out_file="./tree.dot", feature_names=['age', 'pclass=1st', 'pclass=2nd', 'pclass=3rd', '女性', '男性'])
```

4、 决策树总结

- 优点：
  - 简单的理解和解释，树木可视化。
- 缺点：

    - **决策树学习者可以创建不能很好地推广数据的过于复杂的树，这被称为过拟合。**
- 改进：
    - 减枝cart算法(决策树API当中已经实现，随机森林参数调优有相关介绍)
    - **随机森林**

注：企业重要决策，由于决策树很好的分析能力，在决策过程应用较多， 可以选择特征

集成学习方法之随机森林

学习目标

- 目标
    - 说名随机森林每棵决策树的建立过程
    - 知道为什么需要随机有放回(Bootstrap)的抽样
    - 说明随机森林的超参数
- 应用
    - 泰坦尼克号乘客生存预测

1、 什么是集成学习方法

集成学习通过建立几个模型组合的来解决单一预测问题。它的工作原理是**生成多个分类器/模型**，各自独立地学习和作出预测。**这些预测最后结合成组合预测，因此优于任何一个单分类的做出预测。**

2、 什么是随机森林

在机器学习中，**随机森林是一个包含多个决策树的分类器**，并且其输出的类别是由个别树输出的类别的众数而定。

例如, 如果你训练了5个树, 其中有4个树的结果是True, 1个数的结果是False, 那么最终投票结果就是True

3、 随机森林原理过程

学习算法根据下列算法而建造每棵树：

- 用N来表示训练用例（样本）的个数，M表示特征数目。
  - 1、一次随机选出一个样本，重复N次， （有可能出现重复的样本）
  - 2、随机去选出m个特征, m <<M，建立决策树
- 采取bootstrap抽样

3.1 为什么采用BootStrap抽样

- 为什么要随机抽样训练集?
  - 如果不进行随机抽样，每棵树的训练集都一样，那么最终训练出的树分类结果也是完全一样的
- 为什么要有放回地抽样?
  - 如果不是有放回的抽样，那么每棵树的训练样本都是不同的，都是没有交集的，这样每棵树都是“有偏的”，都是绝对“片面的”（当然这样说可能不对），也就是说每棵树训练出来都是有很大的差异的；而随机森林最后分类取决于多棵树（弱分类器）的投票表决。

3.2 API

- class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, bootstrap=True, random_state=None, min_samples_split=2)
  - 随机森林分类器
  - n_estimators：integer，optional（default = 10）森林里的树木数量 120,200,300,500,800,1200
  - criteria：string，可选（default =“gini”）分割特征的测量方法

- max_depth：integer或None，可选（默认=无）树的最大深度 5,8,15,25,30
- max_features="auto”,每个决策树的最大特征数量
  - If "auto", then `max_features=sqrt(n_features)`.
  - If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").
  - If "log2", then `max_features=log2(n_features)`.
  - If None, then `max_features=n_features`.
- bootstrap：boolean，optional（default = True）是否在构建树时使用放回抽样
- min_samples_split:节点划分最少样本数
- min_samples_leaf:叶子节点的最小样本数

- 超参数：n_estimator, max_depth, min_samples_split,min_samples_leaf

3.3 代码

```
# 随机森林去进行预测
rf = RandomForestClassifier()

param = {"n_estimators": [120,200,300,500,800,1200], "max_depth": [5, 8, 15, 25,
30]}

# 超参数调优
gc = GridSearchCV(rf, param_grid=param, cv=2)

gc.fit(x_train, y_train)

print("随机森林预测的准确率为：", gc.score(x_test, y_test))
```

4、总结

- 在当前所有算法中，具有极好的准确率
- 能够有效地运行在大数据集上，处理具有高维特征的输入样本，而且不需要降维
- 能够评估各个特征在分类问题上的重要性

每日作业

1、估计器的工作流程是什么?

答案:

第一步: 实例化估计器

第二步: 调用估计器的fit函数, 用训练集的特征值和目标值训练

第三步: 调用预测函数predict, 用测试集的特征值预测

2、决策树的划分依据是什么?(课程介绍的主要方法)

答案: 更具信息增益最大的属性划分.

3、编程: 通过K近邻算法对鸢尾花数据集进行分类预测

回归与聚类算法

说明线性回归的原理
应用LinearRegression或SGDRegressor实现回归预测
记忆回归算法的评估标准及其公式
说明线性回归的缺点
说明过拟合与欠拟合的原因以及解决方法
说明岭回归的原理即与线性回归的不同之处
说明正则化对于权重参数的影响
说明L1和L2正则化的区别
说明逻辑回归的原理
知道逻辑回归的应用场景
说明分类(主要针对二分类)问题的评估标准
应用classification_report实现精确率、召回率计算
应用roc_auc_score实现指标计算
应用joblib实现模型的保存与加载
说明K-means算法原理
说明K-means的性能评估标准轮廓系数
说明K-means的优缺点

线性回归

学习目标

- 目标
  - 记忆线性回归的原理过程
  - 应用LinearRegression或SGDRegressor实现回归预测
  - 记忆回归算法的评估标准及其公式
- 应用
  - 波士顿房价预测

回忆一下回归问题的判定是什么?

1、 线性回归的原理

1.1 线性回归应用场景

- 房价预测
- 销售额度预测
- 金融：贷款额度预测、利用线性回归以及系数分析因子

1.2 什么是线性回归

1.2.1定义与公式

线性回归(Linear regression)是利用**回归方程(函数)**对一个或**多个自变量(特征值)和因变量(目标值)之间**关系进行建模的一种分析方式。

- 特点：只有一个自变量的情况称为单变量回归，大于一个自变量情况的叫做多元回归

$$\text{通用公式：}\ h(w) = w_1x_1 + w_2x_2 + w_3x_3 \ldots + \mathrm{b} = w^Tx + b$$

$$\text{其中}w, x\text{可以理解为矩阵：}\ \mathrm{w} = \begin{pmatrix} b \\ w_1 \\ w_2 \end{pmatrix}, \mathrm{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

那么怎么理解呢？我们来看几个例子

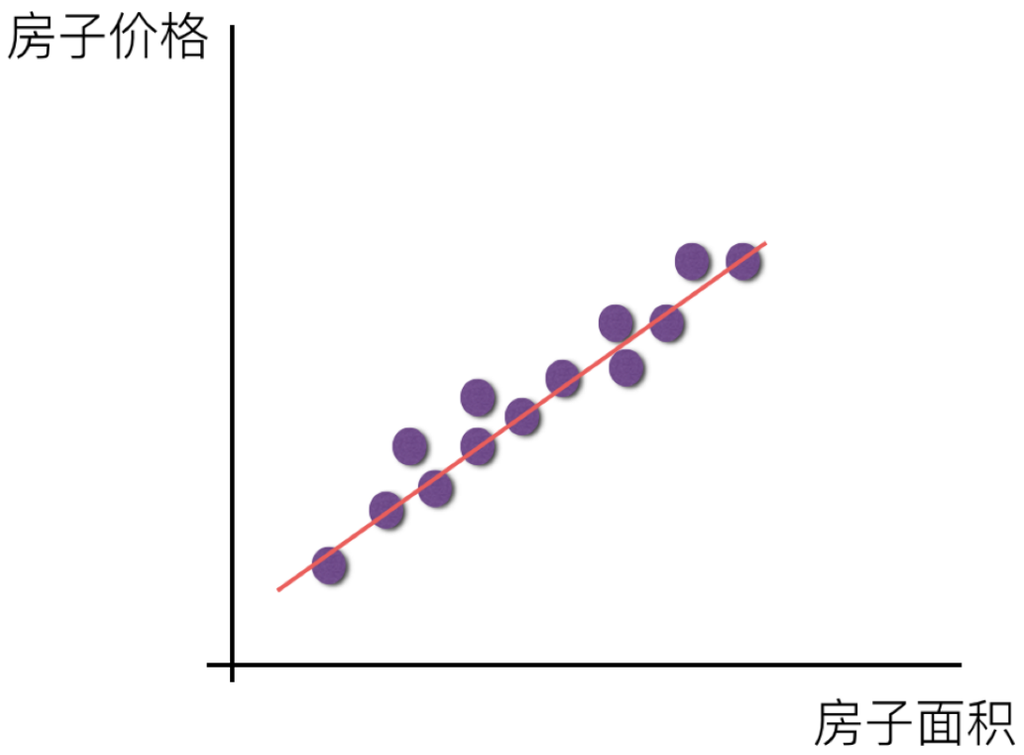
- **期末成绩：0.7×考试成绩+0.3×平时成绩**
- **房子价格 = 0.02×中心区域的距离 + 0.04×城市一氧化氮浓度 + (-0.12×自住房平均房价) + 0.254×城镇犯罪率**

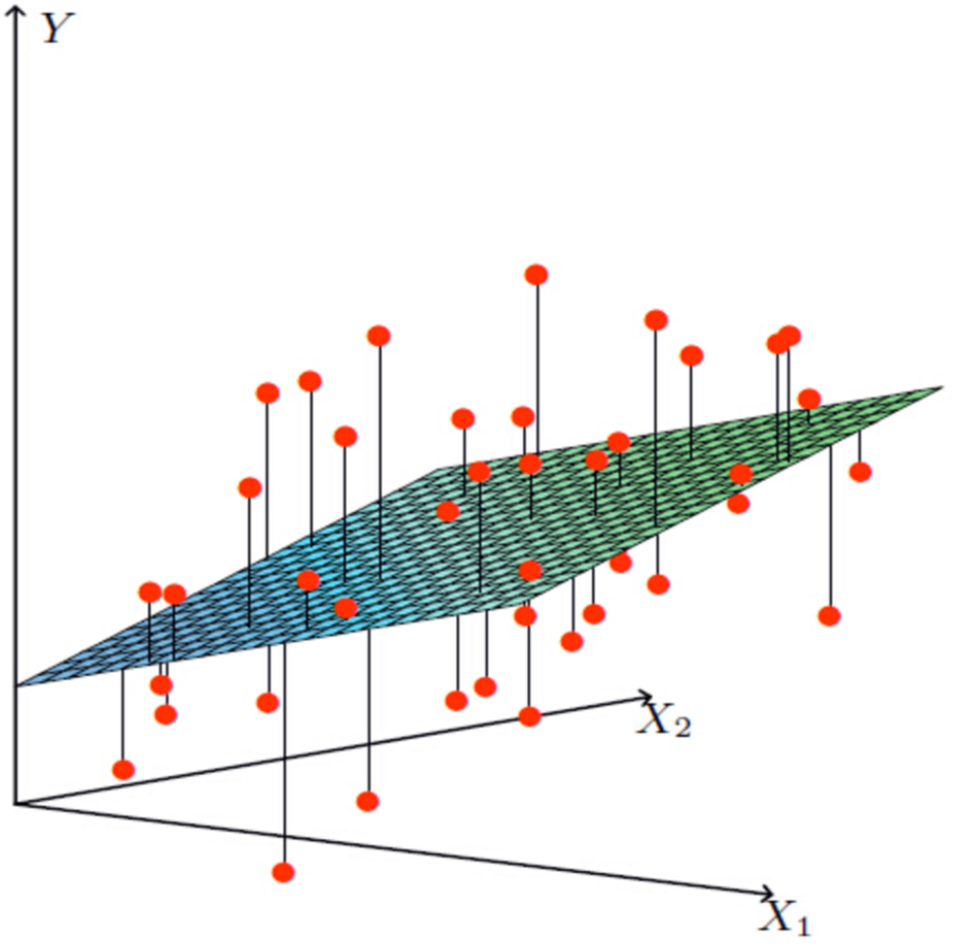
上面两个例子，**我们看到特征值与目标值之间建立的一个关系，这个可以理解为回归方程。**

1.2.2 线性回归的特征与目标的关系分析

线性回归当中的关系有两种，一种是线性关系，另一种是非线性关系。**在这里我们只能画一个平面更好去理解，所以都用单个特征举例子。**

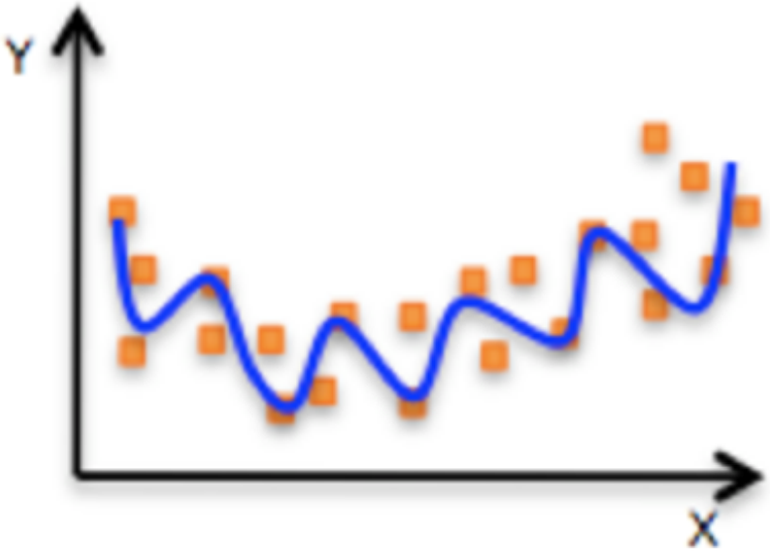
- 线性关系





> 注释：如果在单特征与目标值的关系呈直线关系，或者两个特征与目标值呈现平面的关系
>
> 更高维度的我们不用自己去想，记住这种关系即可

- 非线性关系



> 注释：为什么会这样的关系呢？原因是什么？我们后面 讲解过拟合欠拟合重点介绍
>
> 如果是非线性关系，那么回归方程可以理解为：w1x1+w2x2^{2+w3x3}2

2、线性回归的损失和优化原理（理解记忆）

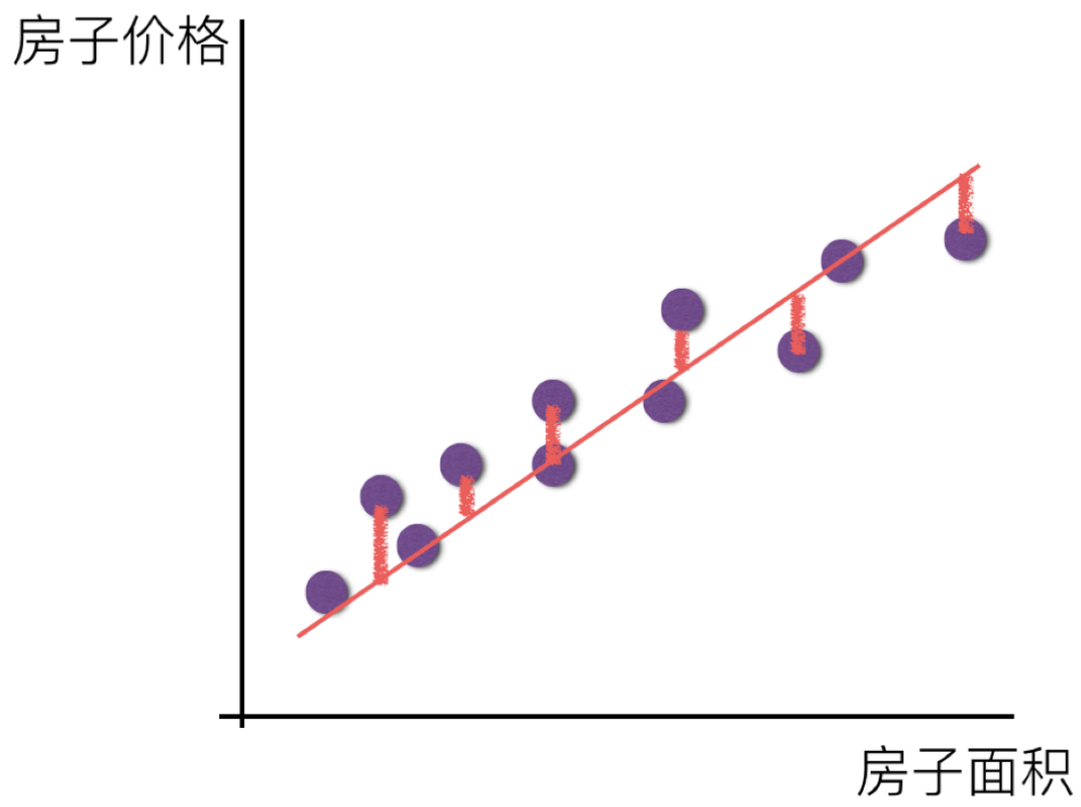
假设刚才的房子例子，真实的数据之间存在这样的关系

```
真实关系：真实房子价格 = 0.02×中心区域的距离 + 0.04×城市一氧化氮浓度 + (-0.12×自住房平均房价) + 0.254×城镇犯罪率
```

那么现在呢，我们随意指定一个关系（猜测）

```
随机指定关系：预测房子价格 = 0.25×中心区域的距离 + 0.14×城市一氧化氮浓度 + 0.42×自住房平均房价 + 0.34×城镇犯罪率
```

请问这样的话，会发生什么？真实结果与我们预测的结果之间是不是存在一定的误差呢？类似这样样子



那么存在这个误差，我们将这个误差给衡量出来

2.1 损失函数

总损失定义为：

$$J(\theta) = (h_w(x_1) - y_1)^2 + (h_w(x_2) - y_2)^2 + \cdots + (h_w(x_m) - y_m)^2$$

$$= \sum_{i=1}^{m}(h_w(x_i) - y_i)^2$$

- y_i为第i个训练样本的真实值
- h(x_i)为第i个训练样本特征值组合预测函数
- 又称最小二乘法

如何去减少这个损失，使我们预测的更加准确些？既然存在了这个损失，我们一直说机器学习有自动学习的功能，在线性回归这里更是能够体现。这里可以通过一些优化方法去优化（其实是数学当中的求导功能）回归的总损失！！！

2.2 优化算法

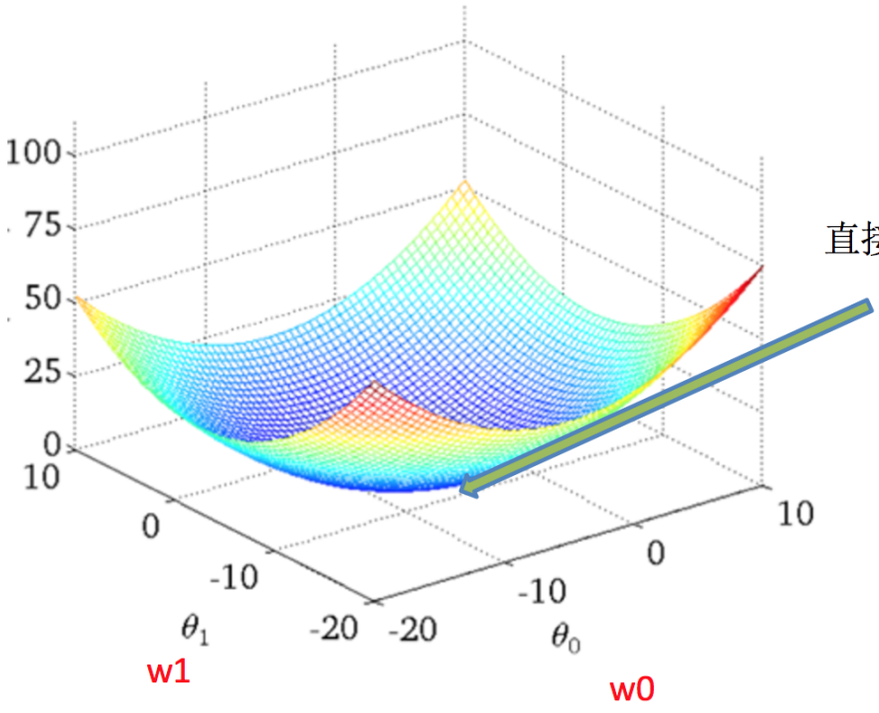
如何去求模型当中的W，使得损失最小？（目的是找到最小损失对应的W值）

线性回归经常使用的两种优化算法

- 正规方程

$$w = (X^T X)^{-1} X^T y$$

> 理解：X为特征值矩阵，y为目标值矩阵。直接求到最好的结果
>
> 缺点：当特征过多过复杂时，求解速度太慢并且得不到结果



直接求解到最小值

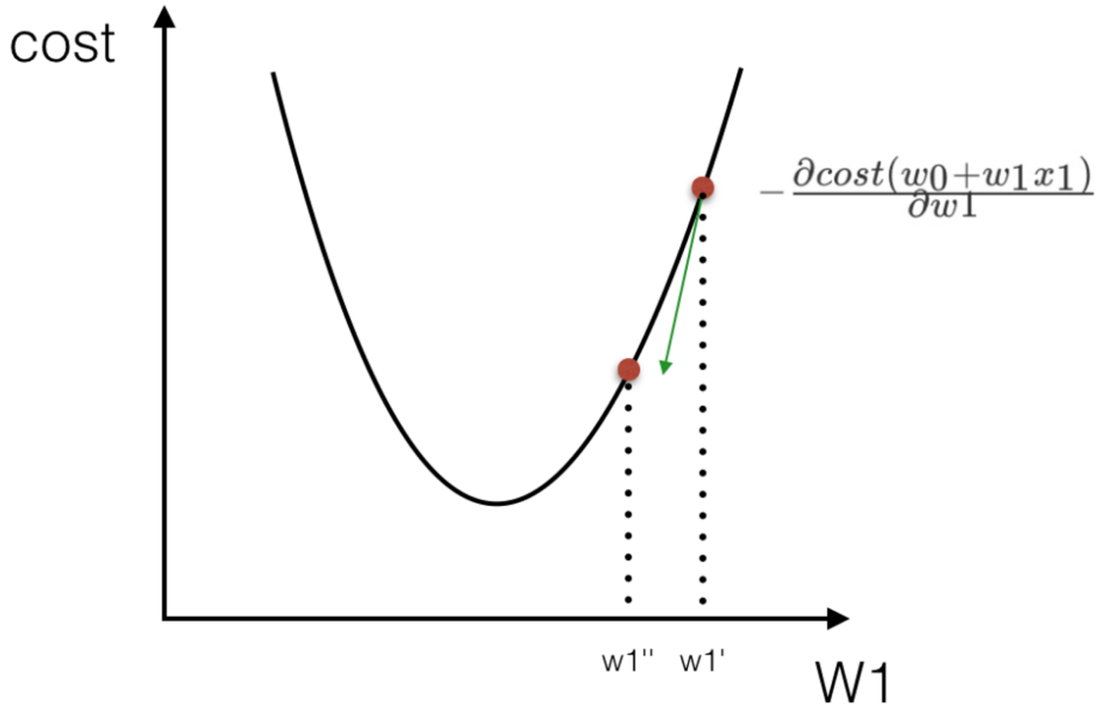
- **梯度下降(Gradient Descent)**

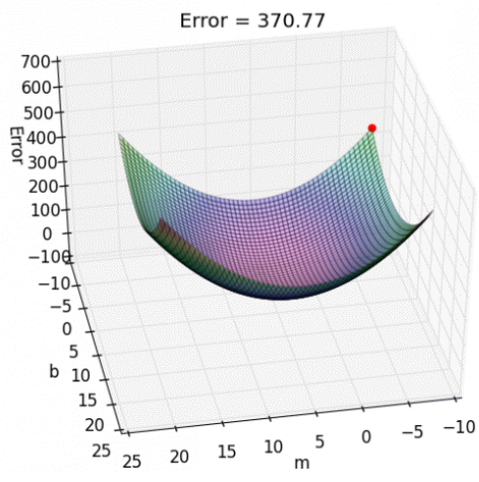
$$w1 := w1 - \alpha \frac{\partial cost(w0+w1x1)}{\partial w1}$$

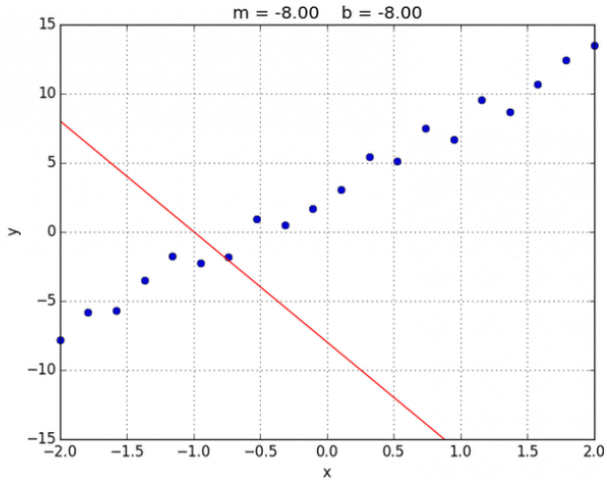
$$w0 := w0 - \alpha \frac{\partial cost(w0+w1x1)}{\partial w1}$$

> 理解：α为学习速率，需要手动指定（超参数），α旁边的整体表示方向
>
> 沿着这个函数下降的方向找，最后就能找到山谷的最低点，然后更新W值
>
> 使用：面对训练数据规模十分庞大的任务，能够找到较好的结果

我们通过两个图更好理解梯度下降的过程

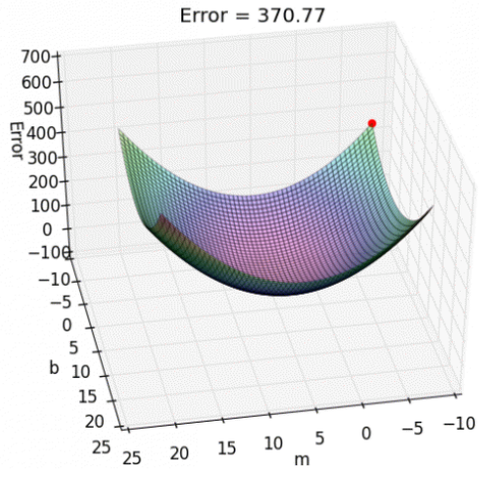


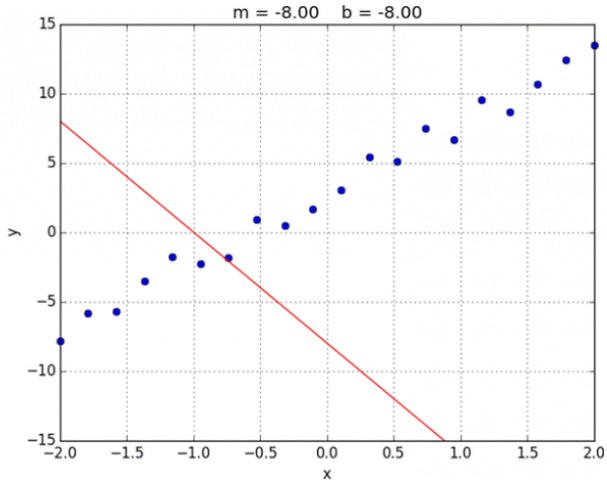




所以有了梯度下降这样一个优化算法，回归就有了"自动学习"的能力

2.3 优化动态图演示





3、 线性回归API

- sklearn.linear_model.LinearRegression(fit_intercept=True)

    - 通过正规方程优化
    - fit_intercept：是否计算偏置
    - LinearRegression.coef_：回归系数
    - LinearRegression.intercept_：偏置
- sklearn.linear_model.SGDRegressor(loss="squared_loss", fit_intercept=True, learning_rate ='invscaling', eta0=0.01)
    - SGDRegressor类实现了随机梯度下降学习，它支持不同的**loss函数和正则化惩罚项**来拟合线性回归模型。
    - loss:损失类型
        - **loss="squared_loss": 普通最小二乘法**
    - fit_intercept：是否计算偏置
    - learning_rate : string, optional
        - 学习率填充
        - **'constant': eta = eta0**
        - **'optimal': eta = 1.0 / (alpha * (t + t0)) [default]**
        - 'invscaling': eta = eta0 / pow(t, power_t)
            - **power_t=0.25:存在父类当中**
        - **对于一个常数值的学习率来说，可以使用learning_rate='constant'，并使用eta0来指定学习率。**
    - SGDRegressor.coef_：回归系数
    - SGDRegressor.intercept_：偏置

> sklearn提供给我们两种实现的API， 可以根据选择使用

4、波士顿房价预测

- 数据介绍

实例数量:	506
属性数量:	13 数值型或类别型，帮助预测的属性

:中位数（第14个属性）经常是学习目标

属性信息 (按顺序):	• CRIM 城镇人均犯罪率 • ZN 占地面积超过2.5万平方英尺的住宅用地比例 • INDUS 城镇非零售业务地区的比例 • CHAS 查尔斯河虚拟变量 (= 1 如果土地在河边；否则是0) • NOX 一氧化氮浓度（每1000万份） • RM 平均每居民房数 • AGE 在1940年之前建成的所有者占用单位的比例 • DIS 与五个波士顿就业中心的加权距离 • RAD 辐射状公路的可达性指数 • TAX 每10,000美元的全额物业税率 • PTRATIO 城镇师生比例 • B 1000(Bk - 0.63)^2 其中 Bk 是城镇的黑人比例 • LSTAT 人口中地位较低人群的百分数 • MEDV 以1000美元计算的自有住房的中位数
缺失属性值:	无
创建者:	Harrison, D. and Rubinfeld, D.L.

这是UCI ML（欧文加利福尼亚大学 机器学习库）房价数据集的副本。 http://archive.ics.uci.edu/ml/datasets/Housing

该数据集是从位于卡内基梅隆大学维护的StatLib图书馆取得的。

属性名	解释	类型
CRIM	该镇的人均犯罪率	连续值
ZN	占地面积超过25,000平方呎的住宅用地比例	连续值
INDUS	非零售商业用地比例	连续值
CHAS	是否邻近 Charles River	离散值，1=邻近；0=不邻近
NOX	一氧化氮浓度	连续值
RM	每栋房屋的平均客房数	连续值
AGE	1940年之前建成的自用单位比例	连续值
DIS	到波士顿5个就业中心的加权距离	连续值
RAD	到径向公路的可达性指数	连续值
TAX	全值财产税率	连续值
PTRATIO	学生与教师的比例	连续值
B	1000(BK - 0.63)^2，其中BK为黑人占比	连续值
LSTAT	低收入人群占比	连续值
MEDV	同类房屋价格的中位数	连续值

> 给定的这些特征，是专家们得出的影响房价的结果属性。我们此阶段不需要自己去探究特征是否有用，只需要使用这些特征。到后面量化很多特征需要我们自己去寻找

4.1 分析

回归当中的数据大小不一致，是否会导致结果影响较大。所以需要做标准化处理。同时我们对目标值也需要做标准化处理。

- 数据分割与标准化处理
- 回归预测
- 线性回归的算法效果评估

4.2 回归性能评估

均方误差(Mean Squared Error)MSE)评价机制：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^i - \bar{y})^2$$

> 注：y^i为预测值，¯y为真实值

- sklearn.metrics.mean_squared_error(y_true, y_pred)
  - 均方误差回归损失
  - y_true:真实值
  - y_pred:预测值

- return:浮点数结果

4.2 代码

```
def mylinearregression():
    """
    线性回归预测房子价格
    :return:
    """
    lb = load_boston()
    #
    # print(lb.data)
    #
    # print(lb.target)

    # 对数据集进行划分
    x_train, x_test, y_train, y_test = train_test_split(lb.data, lb.target,
test_size=0.3, random_state=24)

    # 需要做标准化处理对于特征值处理
    std_x = StandardScaler()

    x_train = std_x.fit_transform(x_train)
    x_test = std_x.fit_transform(x_test)
    # print(x_train)

    # 对于目标值进行标准化
    std_y = StandardScaler()

    y_train = std_y.fit_transform(y_train)
    y_test = std_y.transform(y_test)
    y_test = std_y.inverse_transform(y_test)

    # 使用线性模型进行预测
    # 使用正规方程求解
    lr = LinearRegression()
    # # 此时在干什么？
    lr.fit(x_train, y_train)

    y_lr_predict = std_y.inverse_transform(lr.predict(x_test))

    print(lr.coef_)

    print("正规方程预测的结果为：", y_lr_predict)

    print("正规方程的均方误差为：", mean_squared_error(y_test, y_lr_predict))

    # 梯度下降进行预测
    sgd = SGDRegressor()
    #
    sgd.fit(x_train, y_train)
    print("SGD的权重参数为：", sgd.coef_)
    #
    y_sgd_predict = std_y.inverse_transform(sgd.predict(x_test))
    #
    print("SGD的预测的结果为：", y_sgd_predict)
    #
```

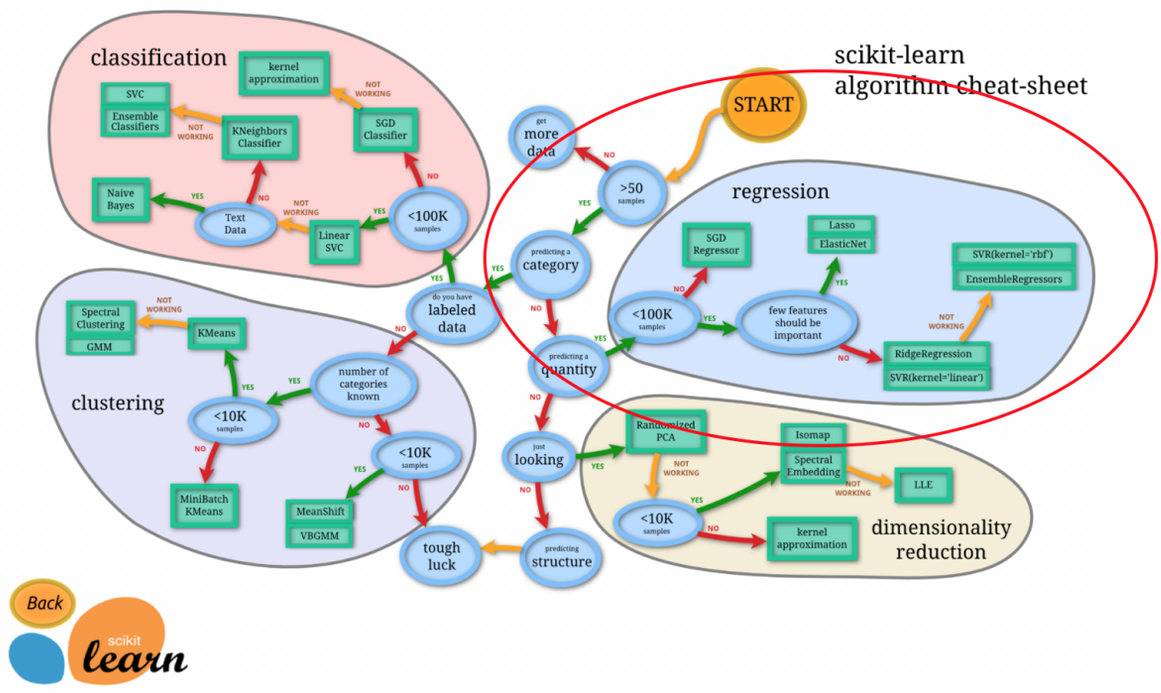
```
    # # 怎么评判这两个方法好坏
    print("SGD的均方误差为：", mean_squared_error(y_test, y_sgd_predict))

    return None
```

我们也可以尝试去修改学习率

```
sgd = SGDRegressor(learning_rate='constant', eta0=0.001)
```

此时我们可以通过调参数，找到学习率效果更好的值。

4.3 正规方程和梯度下降对比



- 文字对比

梯度下降	正规方程
需要选择学习率	不需要
需要迭代求解	一次运算得出
特征数量较大可以使用	需要计算方程，时间复杂度高O(n3)

- 选择：
  - 小规模数据：
    - **LinearRegression(不能解决拟合问题)**
    - 岭回归
  - 大规模数据：SGDRegressor

5、拓展-关于优化方法GD、SGD、SAG

5.1 GD

梯度下降(Gradient Descent)，原始的梯度下降法需要计算所有样本的值才能够得出梯度，计算量大，所以后面才有会一系列的改进。

5.2 SGD

随机梯度下降(Stochastic gradient descent)是一个优化方法。它在一次迭代时只考虑一个训练样本。

- SGD的优点是：
  - 高效
  - 容易实现
- SGD的缺点是：
  - SGD需要许多超参数：比如正则项参数、迭代数。
  - SGD对于特征标准化是敏感的。

5.3 SAG

随机平均梯度法(Stochasitc Average Gradient)，由于收敛的速度太慢，有人提出SAG等基于梯度下降的算法

> Scikit-learn：SGDRegressor、岭回归、逻辑回归等当中都会有SAG优化

6、总结

- 线性回归的损失函数-均方误差
- 线性回归的优化方法
  - 正规方程
  - 梯度下降
- 线性回归的性能衡量方法-均方误差
- sklearn的SGDRegressor API 参数

欠拟合与过拟合

学习目标

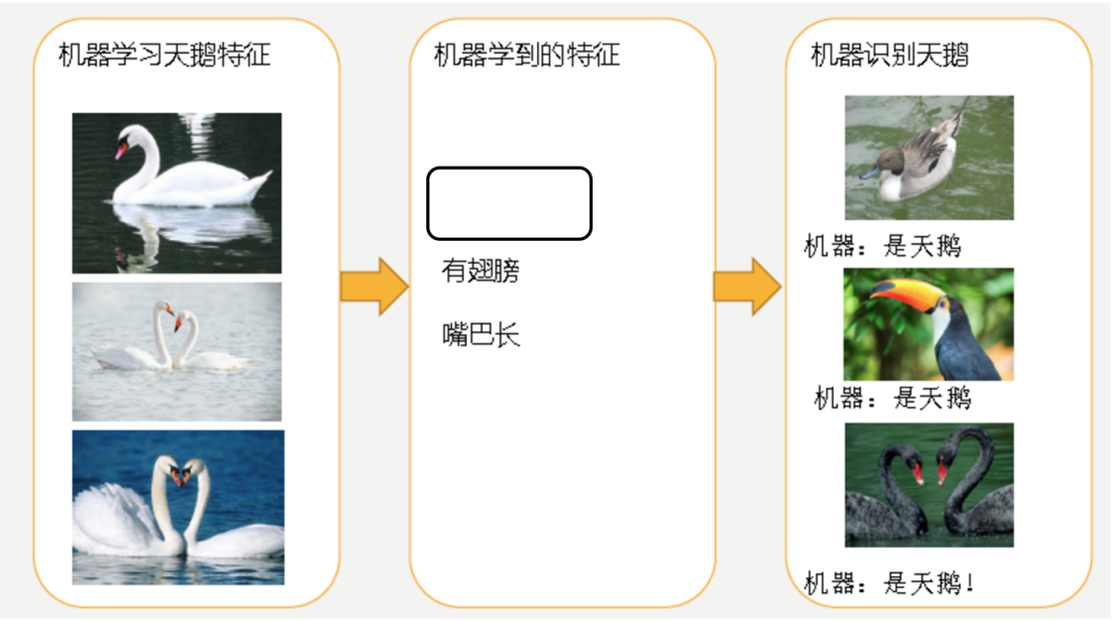
- 目标
  - 说明线性回归（不带正则化）的缺点
  - 说明过拟合与欠拟合的原因以及解决方法
- 应用
  - 无

问题：训练数据训练的很好啊，误差也不大，为什么在测试集上面有问题呢?

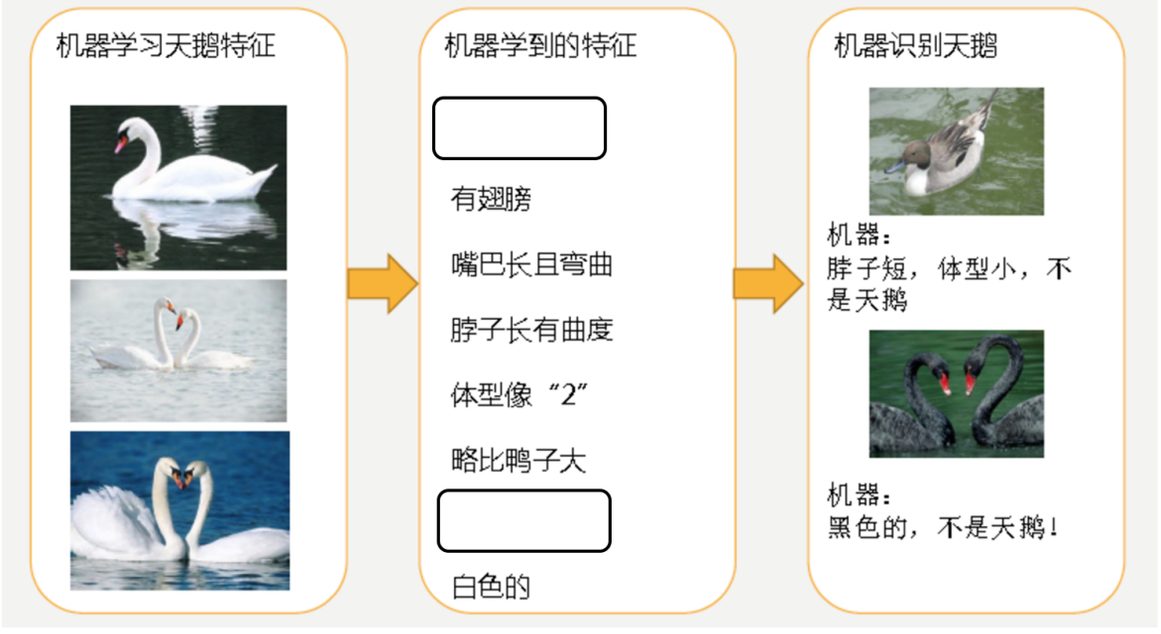
当算法在某个数据集当中出现这种情况，可能就出现了过拟合现象。

1、 什么是过拟合与欠拟合

- 欠拟合



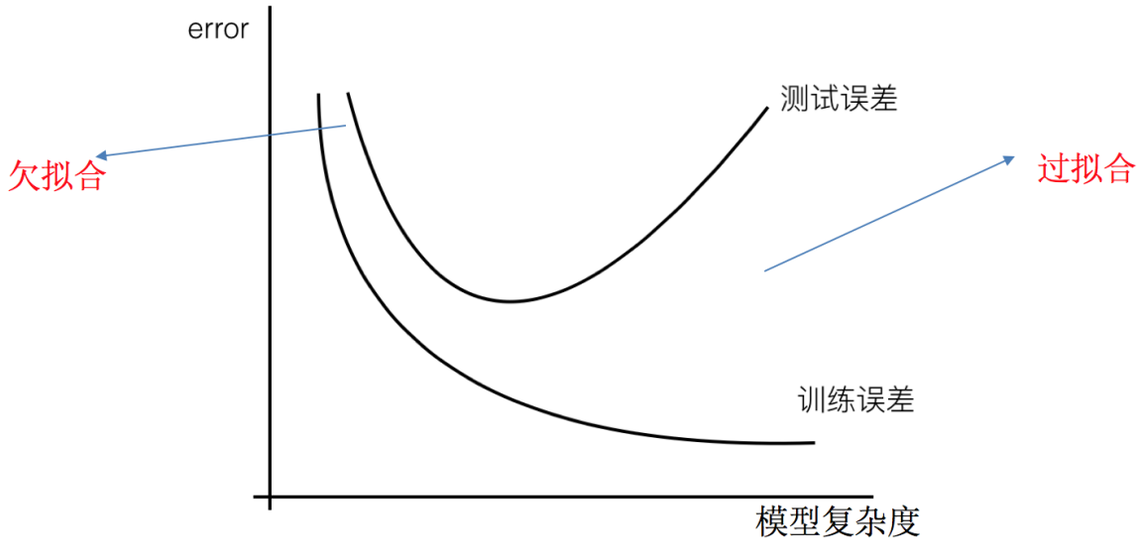
- 过拟合



- 分析
  - 第一种情况：因为机器学习到的天鹅特征太少了，导致区分标准太粗糙，不能准确识别出天鹅。
  - 第二种情况：机器已经基本能区别天鹅和其他动物了。然后，很不巧已有的天鹅图片全是白天鹅的，于是机器经过学习后，会认为天鹅的羽毛都是白的，以后看到羽毛是黑的天鹅就会认为那不是天鹅。

1.1 定义

- 过拟合：一个假设在训练数据上能够获得比其他假设更好的拟合， 但是在测试数据集上却不能很好地拟合数据，此时认为这个假设出现了过拟合的现象。(模型过于复杂)
- 欠拟合：一个假设在训练数据上不能获得更好的拟合，并且在测试数据集上也不能很好地拟合数据，此时认为这个假设出现了欠拟合的现象。(模型过于简单)

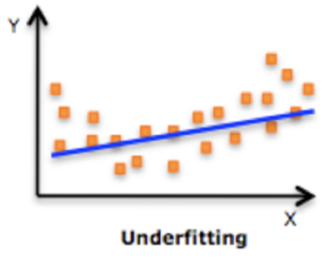


那么是什么原因导致模型复杂？线性回归进行训练学习的时候变成模型会变得复杂，这里就对应前面再说的线性回归的两种关系，非线性关系的数据，也就是存在很多无用的特征或者现实中的事物特征跟目标值的关系并不是简单的线性关系。

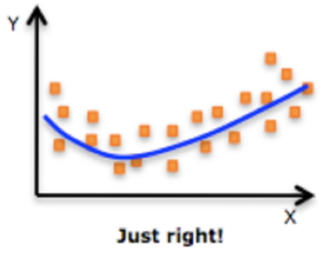
2、 原因以及解决办法

- 欠拟合原因以及解决办法
  - 原因：学习到数据的特征过少
  - 解决办法：增加数据的特征数量
- 过拟合原因以及解决办法
  - 原因：原始特征过多，存在一些嘈杂特征， 模型过于复杂是因为模型尝试去兼顾各个测试数据点
  - 解决办法：
    - 正则化

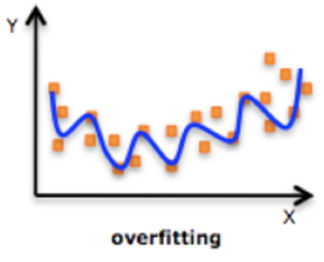
> 在这里针对回归，我们选择了正则化。但是对于其他机器学习算法如分类算法来说也会出现这样的问题，除了一些算法本身作用之外（决策树、神经网络），我们更多的也是去自己做特征选择，包括之前说的删除、合并一些特征



$\theta_0 + \theta_1 x$



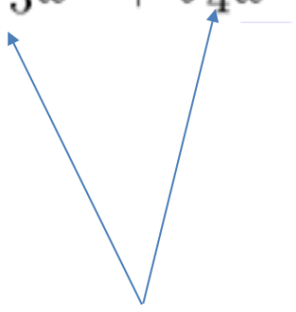
$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

如何解决?

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



尽量减小高次项特征的影响

> 在学习的时候，数据提供的特征有些影响模型复杂度或者这个特征的数据点异常较多，所以算法在学习的时候尽量减少这个特征的影响（甚至删除某个特征的影响），这就是正则化
>
> 注：调整时候，算法并不知道某个特征影响，而是去调整参数得出优化的结果

2.1 正则化类别

- L2正则化
  - 作用：可以使得其中一些W的都很小，都接近于0，削弱某个特征的影响
  - 优点：越小的参数说明模型越简单，越简单的模型则越不容易产生过拟合现象
  - Ridge回归
- L1正则化
  - 作用：可以使得其中一些W的值直接为0，删除这个特征的影响
  - LASSO回归

2.2 拓展-原理(了解)

> 线性回归的损失函数用最小二乘法，等价于当预测值与真实值的误差满足正态分布时的极大似然估计；岭回归的损失函数，是最小二乘法+L2范数，等价于当预测值与真实值的误差满足正态分布，且权重值也满足正态分布（先验分布）时的最大后验估计；LASSO的损失函数，是最小二乘法+L1范数，等价于等价于当预测值与真实值的误差满足正态分布，且且权重值满足拉普拉斯分布（先验分布）时的最大后验估计

线性回归的改进-岭回归

学习目标

- 目标
  - 说明岭回归的原理即与线性回归的不同之处
  - 说明正则化对于权重参数的影响
  - 说明L1和L2正则化的区别
- 应用

- 波士顿房价预测

1、 带有L2正则化的线性回归-岭回归

岭回归，其实也是一种线性回归。只不过在算法建立回归方程时候，加上正则化的限制，从而达到解决过拟合的效果

1.1 API

- sklearn.linear_model.Ridge(alpha=1.0, fit_intercept=True,solver="auto", normalize=False)
  - 具有l2正则化的线性回归
  - alpha:正则化力度，也叫 λ
    - **λ取值：0~1 1~10**
  - solver:会根据数据自动选择优化方法
    - **sag:如果数据集、特征都比较大，选择该随机梯度下降优化**
  - normalize:数据是否进行标准化
    - normalize=False:可以在fit之前调用preprocessing.StandardScaler标准化数据
  - Ridge.coef_:回归权重
  - Ridge.intercept_:回归偏置

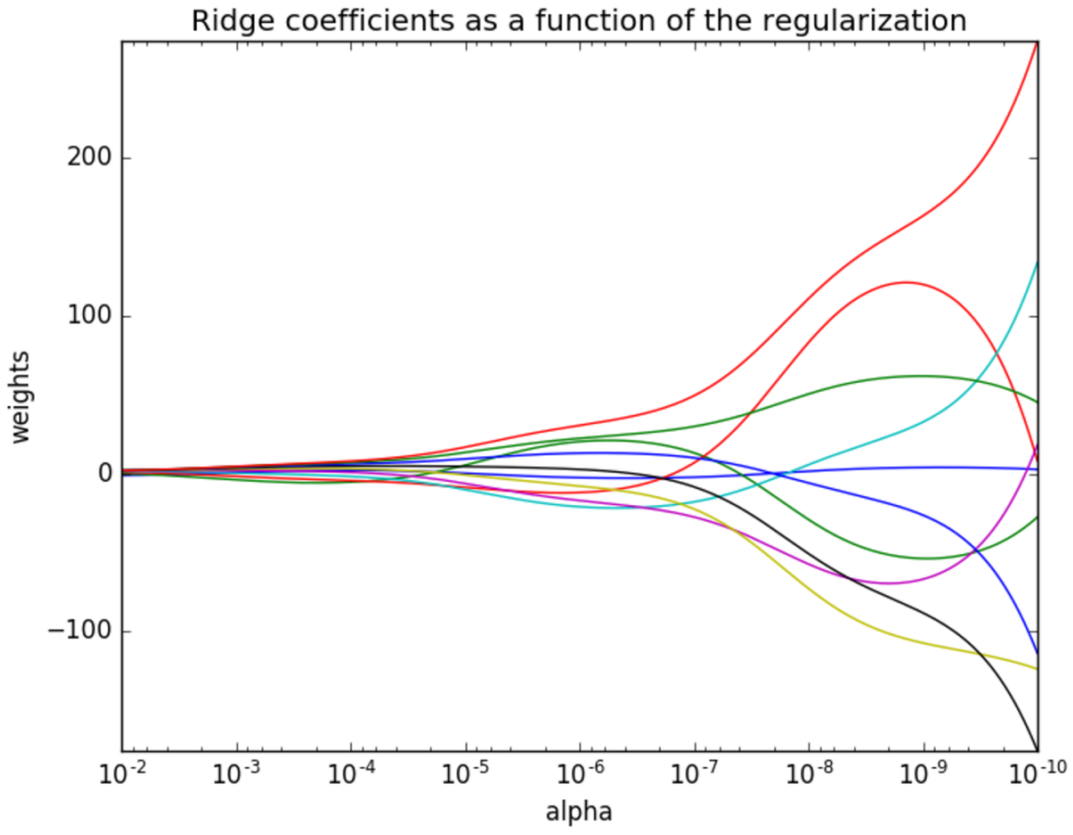
> All last four solvers support both dense and sparse data. However, only 'sag' supports sparse input when `fit_intercept` is True.

Ridge方法相当于SGDRegressor(penalty='l2', loss="squared_loss"),只不过SGDRegressor实现了一个普通的随机梯度下降学习，推荐使用Ridge(实现了SAG)

- sklearn.linear_model.RidgeCV(_BaseRidgeCV, RegressorMixin)
  - 具有l2正则化的线性回归，可以进行交叉验证
  - coef_:回归系数

```
class _BaseRidgeCV(LinearModel):
    def __init__(self, alphas=(0.1, 1.0, 10.0),
                 fit_intercept=True, normalize=False, scoring=None,
                 cv=None, gcv_mode=None,
                 store_cv_values=False):
```

1.2 观察正则化程度的变化，对结果的影响?



- 正则化力度越大，权重系数会越小
- 正则化力度越小，权重系数会越大

1.3 波士顿房价预测

```
rd = Ridge(alpha=1.0)

rd.fit(x_train, y_train)
print("岭回归的权重参数为：", rd.coef_)

y_rd_predict = std_y.inverse_transform(rd.predict(x_test))

print("岭回归的预测的结果为：", y_rd_predict)


print("岭回归的均方误差为：", mean_squared_error(y_test, y_rd_predict))
```

分类算法-逻辑回归与二分类

学习目标

- 目标
  - 说明逻辑回归的损失函数
  - 说明逻辑回归的优化方法
  - 说明sigmoid函数
  - 知道逻辑回归的应用场景
  - 知道精确率、召回率指标的区别
  - 知道F1-score指标说明召回率的实际意义
  - 说明如何解决样本不均衡情况下的评估

    - 了解ROC曲线的意义说明AUC指标大小
    - 应用classification_report实现精确率、召回率计算
    - 应用roc_auc_score实现指标计算
- 应用
    - 癌症患者预测

逻辑回归（Logistic Regression）是机器学习中的一种分类模型，逻辑回归是一种分类算法，虽然名字中带有回归，但是它与回归之间有一定的联系。由于算法的简单和高效，在实际中应用非常广泛。

1、逻辑回归的应用场景

- 广告点击率
- 是否为垃圾邮件
- 是否患病
- 金融诈骗
- 虚假账号

看到上面的例子，我们可以发现其中的特点，那就是都属于两个类别之间的判断。逻辑回归就是解决二分类问题的利器

2、 逻辑回归的原理

2.1 输入

$$h(w) = w_1x_1 + w_2x_2 + w_3x_3 \ldots + \mathrm{b}$$

逻辑回归的输入就是一个线性回归的结果。

2.2 激活函数

- sigmoid函数

$$: g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

- 分析
    - 回归的结果输入到sigmoid函数当中
    - 输出结果：[0, 1]区间中的一个概率值，默认为0.5为阈值

> 逻辑回归最终的分类是通过属于某个类别的概率值来判断是否属于某个类别，并且这个类别默认标记为1(正例),另外的一个类别会标记为0(反例)。（方便损失计算）

> 输出结果解释(重要)：假设有两个类别A，B，并且假设我们的概率值为属于A(1)这个类别的概率值。现在有一个样本的输入到逻辑回归输出结果0.6，那么这个概率值超过0.5，意味着我们训练或者预测的结果就是A(1)类别。那么反之，如果得出结果为0.3那么，训练或者预测结果就为B(0)类别。所以接下来我们回忆之前的线性回归预测结果我们用均方误差衡量，那如果对于逻辑回归，我

> 们预测的结果不对该怎么去衡量这个损失呢？我们来看这样一张图

样本特征值输入				回归		逻辑回归结果		预测结果	真实结果
12.3	20.0	16		82.4		0.4		B	A
9.4	21.1	7.2	回归计算	89.1	sigmoid	0.68		A	B
34.4	18.7	8.1	x W =	80.2	→	0.41	→	B	A
10.2	16.0	12.5		81.3		0.55		B	B
5.6	10.0	6.3		90.4		0.71		A	A

假设得出概率值是属于A的概率值

那么如何去衡量逻辑回归的预测结果与真实结果的差异呢？

2.3 损失以及优化

2.3.1 损失

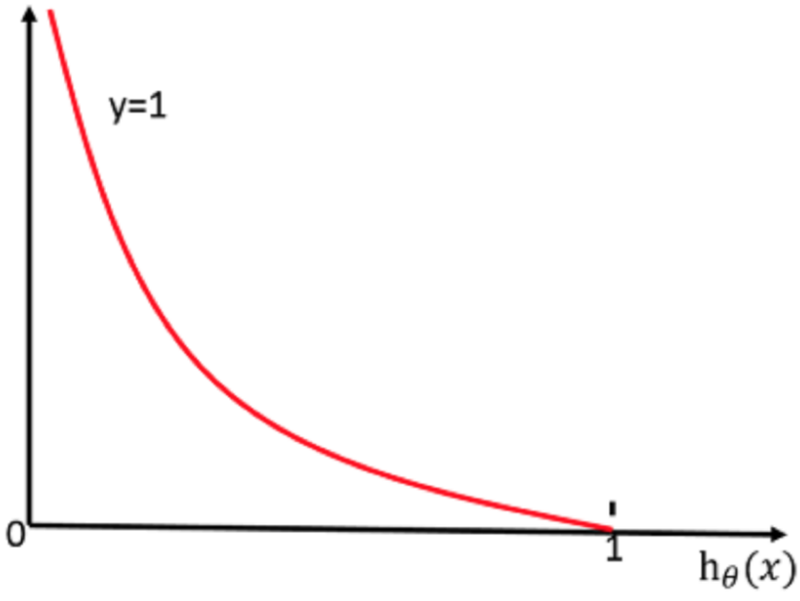
逻辑回归的损失，称之为**对数似然损失**，公式如下：

- 分开类别：

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if y=1} \\ -log(1 - h_\theta(x)) & \text{if y=0} \end{cases}$$

怎么理解单个的式子呢？这个要根据log的函数图像来理解

当y=1时：

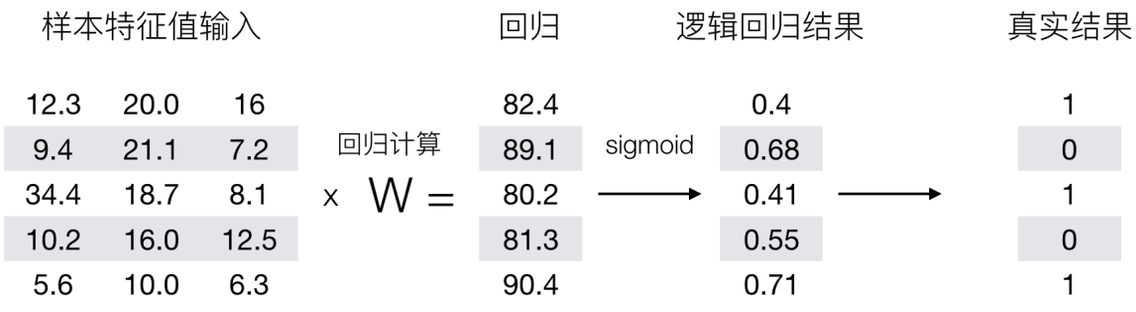


- 综合完整损失函数

$$cost(h_\theta(x), y) = \sum_{i=1}^{m} -y_i log(h_\theta(x)) - (1 - y_i) log(1 - h_\theta(x))$$

> 看到这个式子，其实跟我们讲的信息熵类似。

接下来我们呢就带入上面那个例子来计算一遍，就能理解意义了。



计算损失：1log(0.4)+(1-0)log(1-0.68)+1log(0.41)+(1-0)log(1-0.55)+1log(0.71)

> 我们已经知道，log(P), P值越大，结果越小，所以我们可以对着这个损失的式子去分析

2.3.2 优化

同样使用梯度下降优化算法，去减少损失函数的值。这样去更新逻辑回归前面对应算法的权重参数，**提升原本属于1类别的概率，降低原本是0类别的概率。**

3、逻辑回归API

- sklearn.linear_model.LogisticRegression(solver='liblinear', penalty='l2', C = 1.0)
  - solver:优化求解方式（默认开源的liblinear库实现，内部使用了坐标轴下降法来迭代优化损失函数）
    - sag：根据数据集自动选择，随机平均梯度下降
  - penalty：正则化的种类
  - C：正则化力度

> **默认将类别数量少的当做正例**

> LogisticRegression方法相当于 SGDClassifir(loss="log", penalty=" "),SGDClassifier实现了一个普通的随机梯度下降学习，也支持平均随机梯度下降法（ASGD），可以通过设置average=True。而使用LogisticRegression(实现了SAG)

4、 案例：癌症分类预测-良 / 恶性乳腺癌肿瘤预测

- 数据介绍

Index of /ml/machine-learning-databases/breast-cancer-wisconsin

Name	Last modified	Size	Description
Parent Directory		-	
Index	03-Dec-1996 04:07	326	
breast-cancer-wisconsin.data	16-Jul-1992 10:15	19K	
breast-cancer-wisconsin.names	16-Jul-1992 14:13	5.5K	
unformatted-data	16-Jul-1992 06:17	21K	
wdbc.data	05-Feb-1996 11:04	121K	
wdbc.names	05-Feb-1996 11:04	4.6K	
wpbc.data	01-Feb-1996 16:00	43K	
wpbc.names	01-Feb-1996 16:00	5.5K	

原始数据的下载地址：https://archive.ics.uci.edu/ml/machine-learning-databases/

> 数据描述
>
> （1）699条样本，共11列数据，第一列用语检索的id，后9列分别是与肿瘤
>
> 相关的医学特征，最后一列表示肿瘤类型的数值。
>
> （2）包含16个缺失值，用”?”标出。

4.1 分析

- 缺失值处理
- 标准化处理
- 逻辑回归预测

4.2 代码

```
def logisticregression():
    """
    逻辑回归进行癌症预测
    :return: None
    """
    # 1、读取数据，处理缺失值以及标准化
    column_name = ['Sample code number', 'Clump Thickness', 'Uniformity of Cell
Size', 'Uniformity of Cell Shape',
                   'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare
Nuclei', 'Bland Chromatin',
                   'Normal Nucleoli', 'Mitoses', 'Class']

    data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-
databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data",
                       names=column_name)

    # 删除缺失值
    data = data.replace(to_replace='?', value=np.nan)

    data = data.dropna()

    # 取出特征值
    x = data[column_name[1:10]]

    y = data[column_name[10]]

    # 分割数据集
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

```
    # 进行标准化
    std = StandardScaler()

    x_train = std.fit_transform(x_train)

    x_test = std.transform(x_test)

    # 使用逻辑回归
    lr = LogisticRegression()

    lr.fit(x_train, y_train)

    print("得出来的权重：", lr.coef_)

    # 预测类别
    print("预测的类别：", lr.predict(x_test))

    # 得出准确率
    print("预测的准确率:", lr.score(x_test, y_test))
    return None
```

在很多分类场景当中我们不一定只关注预测的准确率！！！！！

比如以这个癌症举例子！！！我们并不关注预测的准确率，而是关注在所有的样本当中，癌症患者有没有被全部预测（检测）出来。

5、分类的评估方法

5.1 精确率与召回率

5.1.1混淆矩阵

在分类任务下，预测结果(Predicted Condition)与正确标记(True Condition)之间存在四种不同的组合，构成混淆矩阵(适用于多分类)

真实结果 \ 预测结果	正例	假例
正例	真正例TP	伪反例FN
假例	伪正例FP	真反例TN

5.1.2 精确率(Precision)与召回率(Recall)

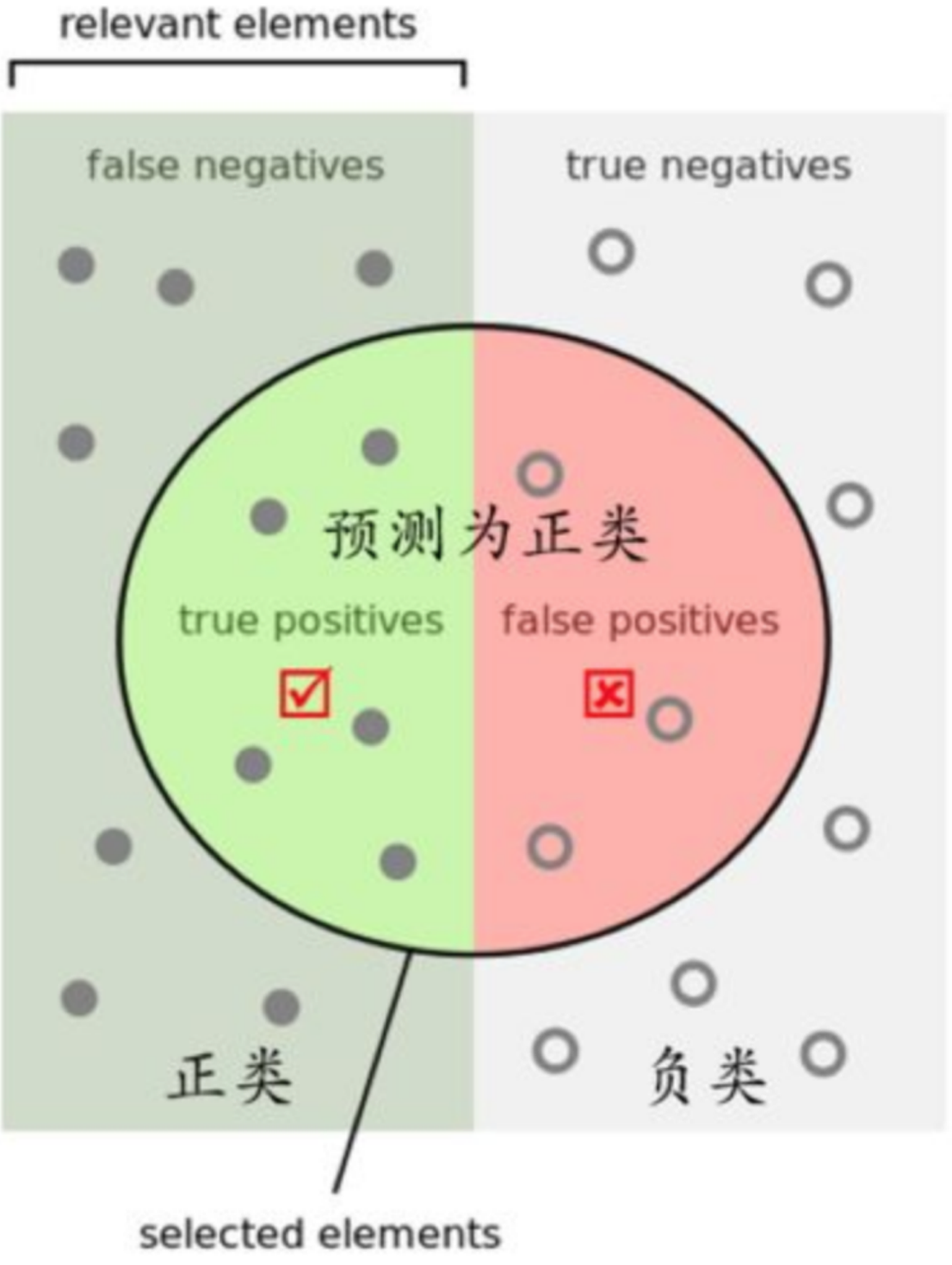
- 精确率：预测结果为正例样本中真实为正例的比例（了解）

真实结果 \ 预测结果	正例	假例
正例	真正例TP	伪反例FN
假例	伪正例FP	真反例TN

- 召回率：真实为正例的样本中预测结果为正例的比例（查的全，对正样本的区分能力）

真实结果 \ 预测结果	正例	假例
正例	真正例TP	伪反例FN
假例	伪正例FP	真反例TN

那么怎么更好理解这个两个概念



还有其他的评估标准，F1-score，反映了模型的稳健型

$$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

5.1.3 分类评估报告API

- sklearn.metrics.classification_report(y_true, y_pred, labels=[], target_names=None)
- - y_true：真实目标值
  - y_pred：估计器预测目标值
  - labels:指定类别对应的数字
  - target_names：目标类别名称
  - return：每个类别精确率与召回率

```
print("精确率和召回率为：", classification_report(y_test, lr.predict(x_test), labels=[2, 4], target_names=['良性', '恶性']))
```

假设这样一个情况，如果99个样本癌症，1个样本非癌症，不管怎样我全都预测正例(默认癌症为正例)，准确率就为99%但是这样效果并不好，这就是样本不均衡下的评估问题

问题：如何衡量样本不均衡下的评估?

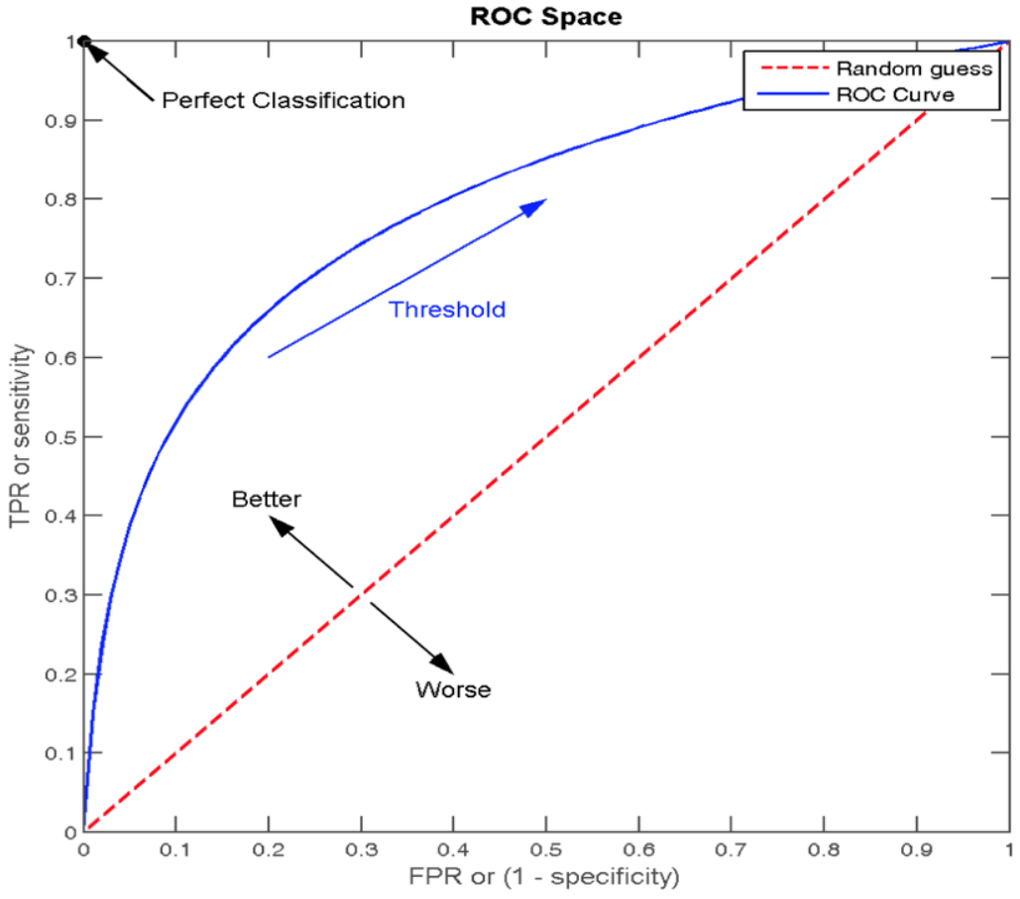
5.2 ROC曲线与AUC指标

5.2.1 知道TPR与FPR

- TPR = TP / (TP + FN)
  - 所有真实类别为1的样本中，预测类别为1的比例
- FPR = FP / (FP + FN)
  - 所有真实类别为0的样本中，预测类别为1的比例

5.2.2 ROC曲线

- ROC曲线的横轴就是FPRate，纵轴就是TPRate，当二者相等时，表示的意义则是：对于不论真实类别是1还是0的样本，分类器预测为1的概率是相等的，此时AUC为0.5



5.2.3AUC指标

- AUC的概率意义是随机取一对正负样本，正样本得分大于负样本的概率
- AUC的最小值为0.5，最大值为1，取值越高越好
- **AUC=1，完美分类器，采用这个预测模型时，不管设定什么阈值都能得出完美预测。绝大多数预测的场合，不存在完美分类器。**
- **0.5<AUC<1，优于随机猜测。这个分类器（模型）妥善设定阈值的话，能有预测价值。**

> **最终AUC的范围在[0.5, 1]之间，并且越接近1越好**

5.2.4 AUC计算API

- from sklearn.metrics import roc_auc_score
  - sklearn.metrics.roc_auc_score(y_true, y_score)
    - 计算ROC曲线面积，即AUC值
    - y_true:每个样本的真实类别，必须为0(反例),1(正例)标记
    - y_score:每个样本预测的概率值

```
# 0.5~1之间，越接近于1约好
y_test = np.where(y_test > 2.5, 1, 0)

print("AUC指标：", roc_auc_score(y_test, lr.predict(x_test)))
```

5.2.5、总结

- AUC只能用来评价二分类
- AUC非常适合评价样本不平衡中的分类器性能

模型保存和加载

学习目标

- 目标
  - 应用joblib实现模型的保存与加载
- 应用
  - 无

当训练或者计算好一个模型之后，那么如果别人需要我们提供结果预测，就需要保存模型（主要是保存算法的参数）

1、sklearn模型的保存和加载API

- from sklearn.externals import joblib
  - 保存：joblib.dump(rf, 'test.pkl')
  - 加载：estimator = joblib.load('test.pkl')

2、线性回归的模型保存加载案例

- 保存

```
# 使用线性模型进行预测
# 使用正规方程求解
lr = LinearRegression()
# 此时在干什么？
lr.fit(x_train, y_train)
# 保存训练完结束的模型
joblib.dump(lr, "test.pkl")
```

- 加载

```
# 通过已有的模型去预测房价
model = joblib.load("test.pkl")
print("从文件加载进来的模型预测房价的结果：",
std_y.inverse_transform(model.predict(x_test)))
```

无监督学习-K-means算法

学习目标

- 目标
  - 说明K-means算法原理
  - 说明K-means的性能评估标准轮廓系数
  - 说明K-means的优缺点
- 应用
  - instacart用户聚类

回忆非监督学习的特点?

1、 什么是无监督学习

- 一家广告平台需要根据相似的人口学特征和购买习惯将美国人口分成不同的小组，以便广告客户可以通过有关联的广告接触到他们的目标客户。
- Airbnb 需要将自己的房屋清单分组成不同的社区，以便用户能更轻松地查阅这些清单。
- 一个数据科学团队需要降低一个大型数据集的维度的数量，以便简化建模和降低文件大小。

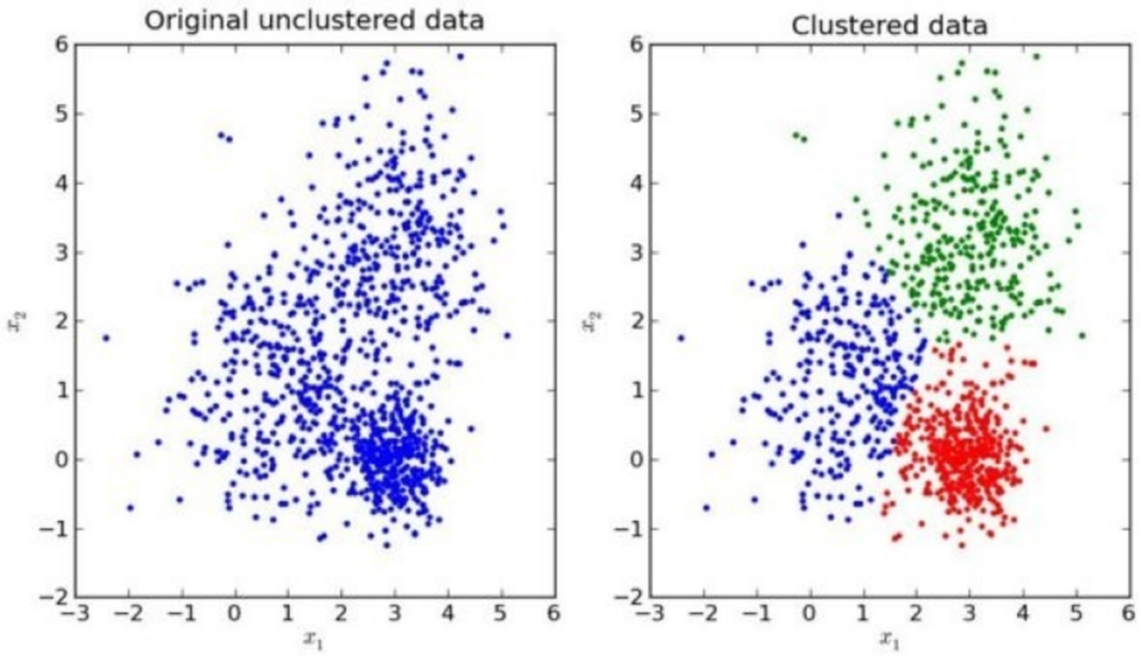
我们可以怎样最有用地对其进行归纳和分组？我们可以怎样以一种压缩格式有效地表征数据？**这都是无监督学习的目标，之所以称之为无监督，是因为这是从无标签的数据开始学习的。**

2、 无监督学习包含算法

- 聚类
  - K-means(K均值聚类)
- 降维
  - PCA

3、 K-means原理

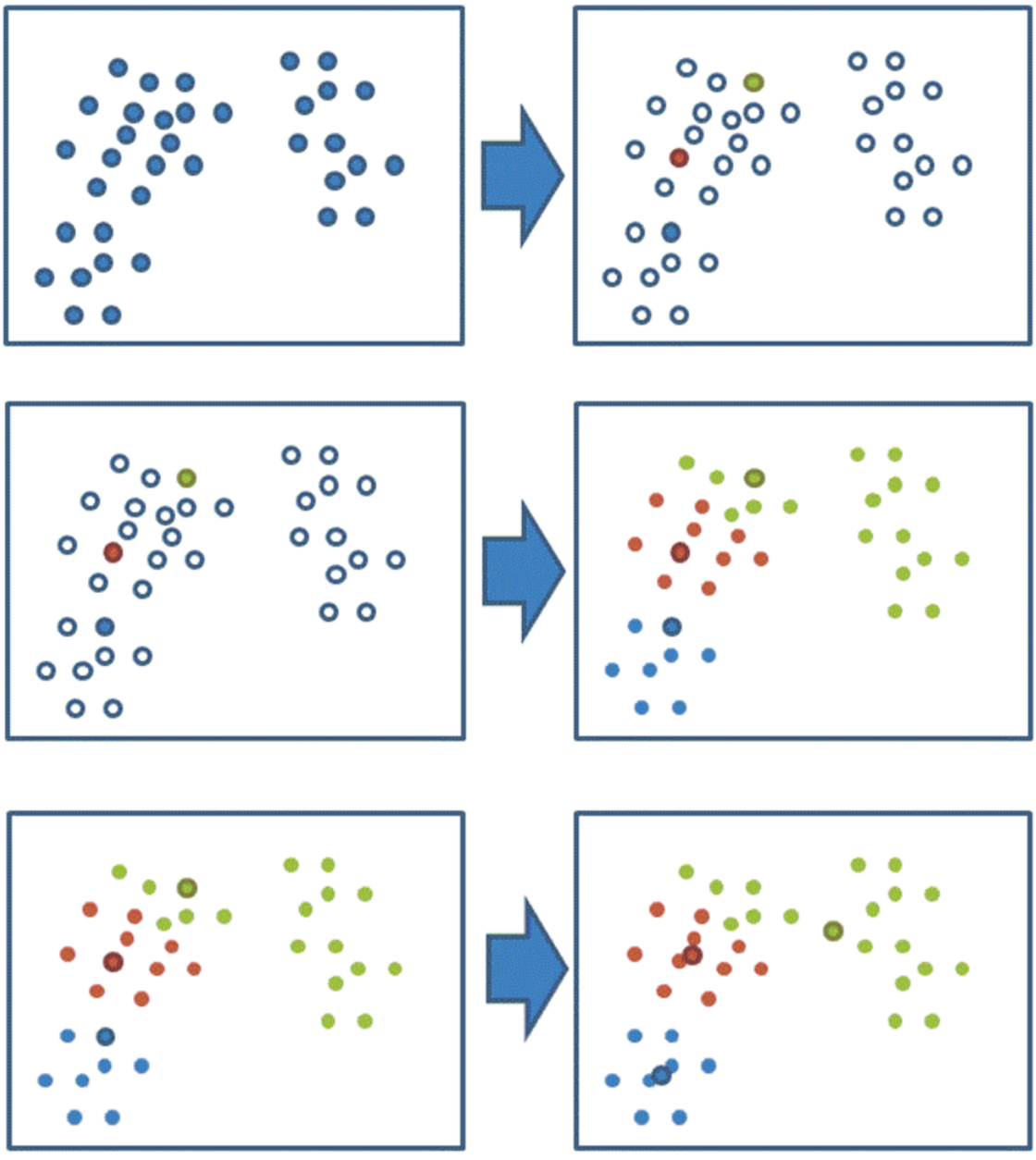
我们先来看一下一个K-means的聚类效果图



3.1 K-means聚类步骤

- 1、随机设置K个特征空间内的点作为初始的聚类中心
- 2、对于其他每个点计算到K个中心的距离，未知的点选择最近的一个聚类中心点作为标记类别
- 3、接着对着标记的聚类中心之后，重新计算出每个聚类的新中心点（平均值）
- 4、如果计算得出的新中心点与原中心点一样，那么结束，否则重新进行第二步过程

我们以一张图来解释效果



4、K-meansAPI

- sklearn.cluster.KMeans(n_clusters=8,init='k-means++')
  - k-means聚类
  - n_clusters:开始的聚类中心数量
  - init:初始化方法，默认为'k-means ++'
  - labels_:默认标记的类型，可以和真实值比较（不是值比较）

5、 案例：k-means对Instacart Market用户聚类

5.1 分析

- 1、降维之后的数据
- 2、k-means聚类
- 3、聚类结果显示

5.2 代码

```
# 取500个用户进行测试
cust = data[:500]
km = KMeans(n_clusters=4)
km.fit(cust)
pre = km.predict(cust)
```

问题：如何去评估聚类的效果呢?

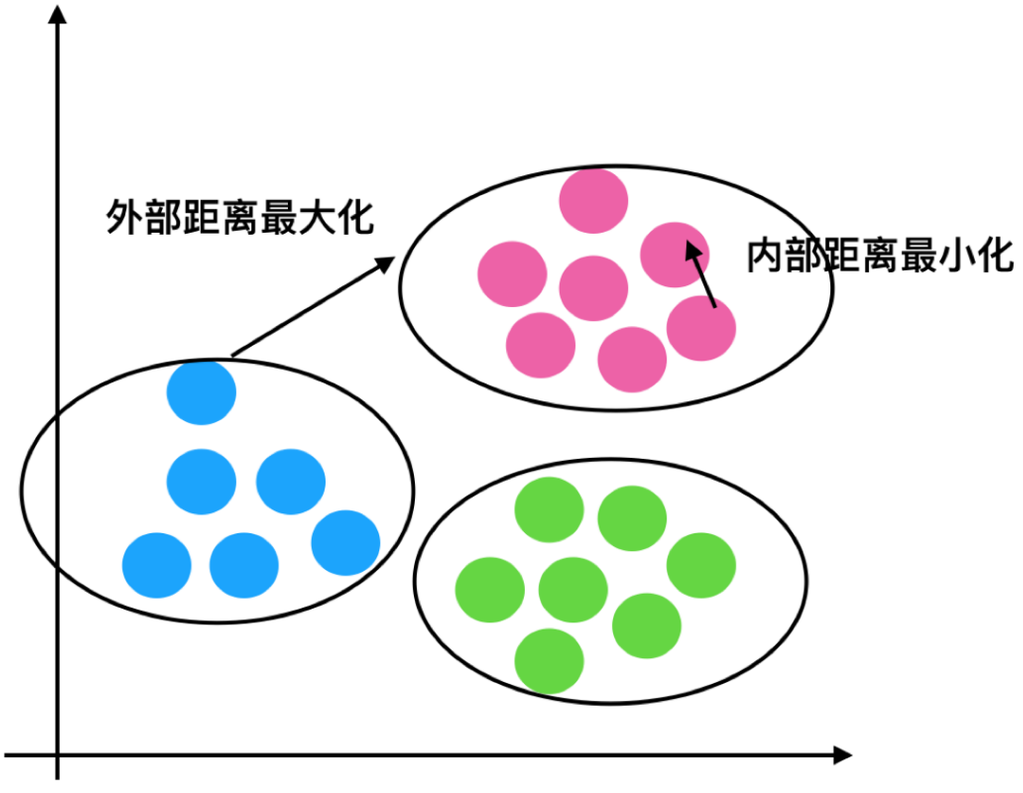
6、Kmeans性能评估指标

6.1 轮廓系数

$$sc_i = \frac{b_i - a_i}{\max(b_i, a_i)}$$

> 注：对于每个点i 为已聚类数据中的样本 ， b_i 为i 到其它族群的所有样本的距离最小值，a_i 为i 到本身簇的距离平均值。最终计算出所有的样本点的轮廓系数平均值

6.2 轮廓系数值分析



- 分析过程（我们以一个蓝1点为例）
  - 1、计算出蓝1离本身族群所有点的距离的平均值a_i

- 2、蓝1到其它两个族群的距离计算出平均值红平均，绿平均，取最小的那个距离作为b_i
- 根据公式：极端值考虑：如果b_i >>a_i: 那么公式结果趋近于1；如果a_i>>>b_i: 那么公式结果趋近于-1

6.3 结论

如果b_i>>a_i:趋近于1效果越好， b_i<<a_i:趋近于-1，效果不好。轮廓系数的值是介于 [-1,1] ，越趋近于1代表内聚度和分离度都相对较优。

6.4 轮廓系数API

- sklearn.metrics.silhouette_score(X, labels)
  - 计算所有样本的平均轮廓系数
  - X：特征值
  - labels：被聚类标记的目标值

6.5 用户聚类结果评估

```
silhouette_score(cust, pre)
```

7、K-means总结

- 特点分析：采用迭代式算法，直观易懂并且非常实用
- 缺点：容易收敛到局部最优解(多次聚类)

> 注意：聚类一般做在分类之前

每日作业

1、线性回归的参数求解的方法是什么?

答案: 正规方程和梯度下降

2、什么是过拟合? 原因有哪些?

答案: 过拟合就是训练误差很小,但是测试误差很大

原因有: 样本偏差, 模型过于复杂

3、分类问题, 回归问题, 聚类问题的评估方法分别是什么?

答案: 分类问题的评估方法是准确率, 精确率和召回率

回归问题的评估方法是均方差

聚类问题的评估方法是轮廓系数