使用scikit-learn构建机器学习模型

2020/6/14

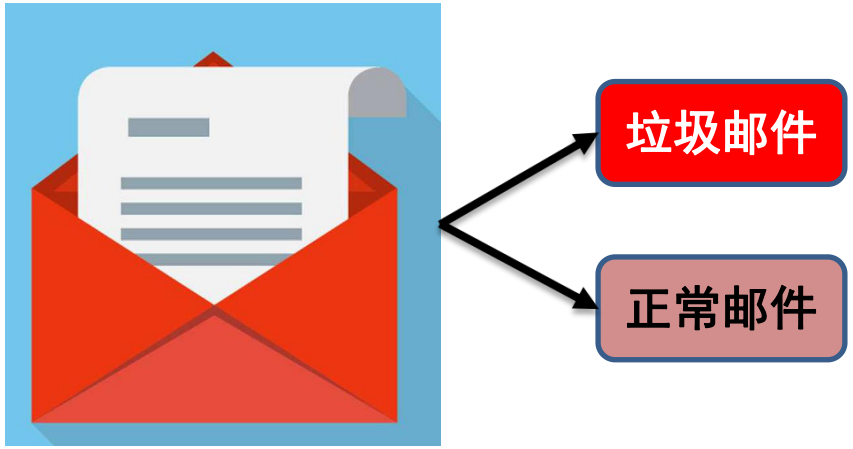
目录

分类问题

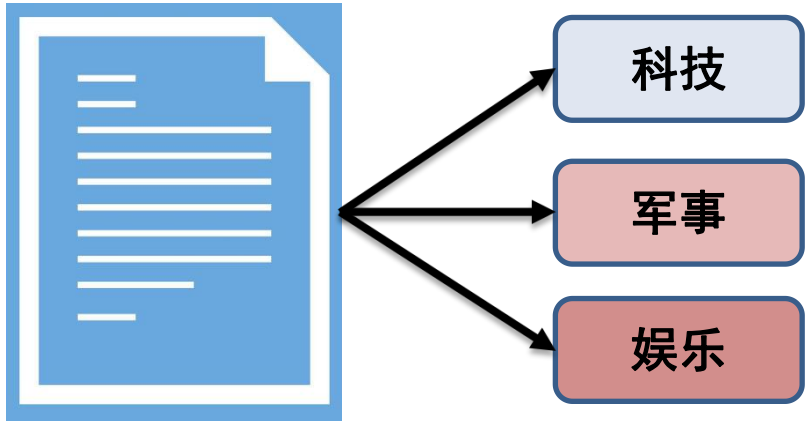
分类问题是监督学习的一个核心问题，它从数据中学习一个分类决策函数或分类模型(分类器（classifier）)，对新的输入进行输出预测，输出变量取有限个离散值。

分类在我们日常生活中很常见

✓ 二分类问题



✓ 多分类问题



核心算法

✓ 决策树、贝叶斯、SVM、逻辑回归

1、贝叶斯分类

朴素贝叶斯模型是一组快速简单的分类算法，适用于维度高的数据集，速度快、适合快速粗糙方案

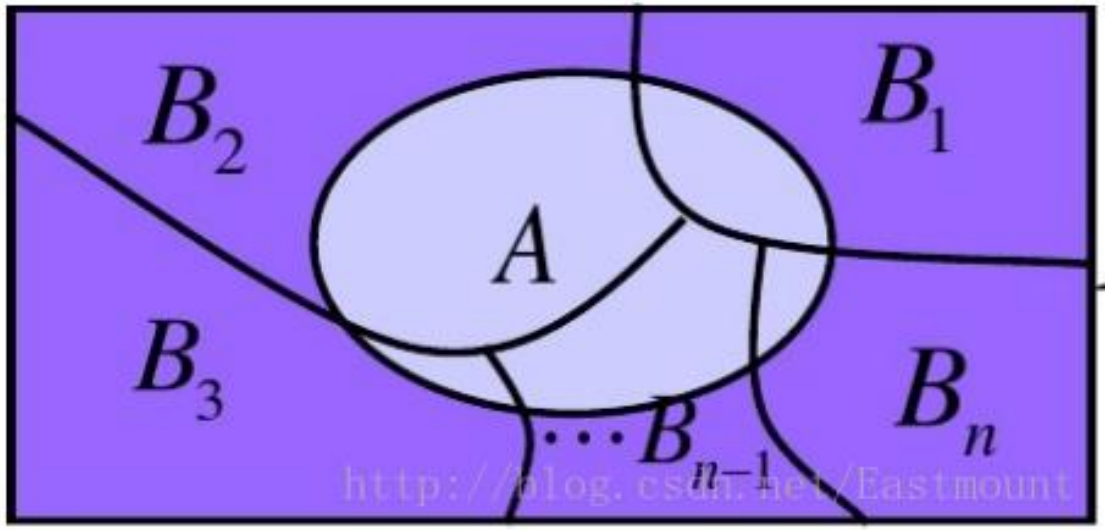
概率论基础　　条件概率公式　　全概率公式　贝叶斯定理

$$P(B_i \mid A)=\frac{P(AB_i)}{P(A)}=\frac{P(B_i)P(A \mid B_i)}{P(A \mid B_1)P(B_1)+P(A \mid B_2)P(B_2)+\ldots+P(A \mid B_n)P(B_n)}$$

A为实验E的事件，B1、B2、....、Bn为样本空间Ω的一个划分，且P(Bi)>0。

实际分类过程中，A可以看作是特征值，Bi为第i类别编号。依据贝叶斯公式计算由特征值A，判断其属于某i一类的概率。
其中最大的即为所属类别。

每一类Bi都是独立分布，例如高斯分布。



贝叶斯分类

贝叶斯公式

$$P(c|x) = \frac{P(x,c)}{P(x)} = \frac{P(c)P(x|c)}{P(x)}$$

$$P(\text{类别}|\text{特征}) = \frac{P(\text{特征}|\text{类别})P(\text{类别})}{P(\text{特征})}$$

贝叶斯分类是基于贝叶斯定理和属性特征条件独立性的分类方法。
贝叶斯流派的核心：Probability theory is nothing but common sense reduced to calculation.

概率论只不过是把常识用数学公式表达了出来。——拉普拉斯

案例：假设春季感冒流行，你同桌打了一个喷嚏，那你同桌感冒了的概率是多少？

1. 计算先验概率： 你同桌没有任何症状的情况下可能得感冒的概率是多少？

2. 为每个属性计算条件概率： 如果你同桌感冒了，那么他会打喷嚏的概率是多少， 如果他没感冒， 出现打喷嚏症状的概率有多少？

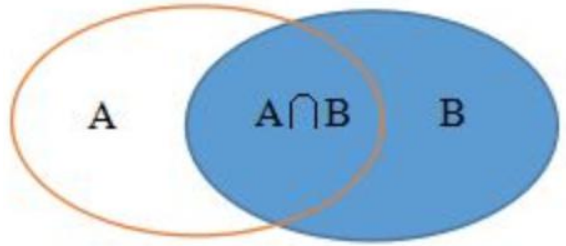
3. 计算后验概率： 根据1 和2求解最终问题，这才是拥有贝叶斯思想的你该做的分析。

贝叶斯分类

贝叶斯公式

$$P(c|x)=\frac{P(x,c)}{P(x)}=\frac{P(c)P(x|c)}{P(x)}$$

$$P(\text{类别}|\text{特征})=\frac{P(\text{特征}|\text{类别})P(\text{类别})}{P(\text{特征})}$$



例子：一对男女朋友，男生想女生求婚，男生的四个特点分别是不帅，性格不好，身高矮，不上进，请你判断一下女生是嫁还是不嫁？

帅？	性格好？	身高？	上进？	嫁与否
帅	不好	矮	不上进	不嫁
不帅	好	矮	上进	不嫁
帅	好	矮	上进	嫁
不帅	好	高	上进	嫁
帅	不好	矮	上进	不嫁
不帅	不好	矮	不上进	不嫁
帅	好	高	不上进	嫁
不帅	好	高	上进	嫁
帅	好	高	上进	嫁
不帅	不好	高	上进	嫁
帅	好	矮	不上进	不嫁
帅	好	矮	不上进	不嫁

① 估计先验概率P(c)　　p(嫁) = 6/12（总样本数） = 1/2

② 为每个属性计算条件概率$P(x_i|c)$

p(不帅|嫁) = 3/6 = 1/2　　p(性格不好|嫁)= 1/6

p（不帅） = 4/12 = 1/3

③ 计算后验概率

$$p(\text{嫁}|\text{不帅、性格不好、身高矮、不上进})=\frac{p(\text{不帅、性格不好、身高矮、不上进}|\text{嫁})*p(\text{嫁})}{p(\text{不帅、性格不好、身高矮、不上进})}$$

$$=\frac{p(\text{不帅}|\text{嫁})*p(\text{性格不好}|\text{嫁})*p(\text{身高矮}|\text{嫁})*p(\text{不上进}|\text{嫁})*p(\text{嫁})}{p(\text{不帅})*p(\text{性格不好})*p(\text{身高矮})*p(\text{不上进})}$$

不嫁(1/6*1/2*1*1/2) > 嫁(1/2*1/6*1/6*1/6*1/2)

分析结果：不嫁！

贝叶斯分类

$$\frac{p(不帅|不嫁)*p(性格不好|不嫁)*p(身高矮|不嫁)*p(不上进|不嫁)*p(不嫁)}{p(不帅)*p(性格不好)*p(身高矮)*p(不上进)}$$

拉普拉斯修正

先验概率拉普拉斯修正 $P(c)=\frac{Dc}{D} \Rightarrow P(c)=\frac{Dc+1}{D+N}$

条件概率拉普拉斯修正 $P(x_i|c)=\frac{D_{c,xi}}{Dc} \Rightarrow P(x_i|c)=\frac{D_{c,xi}+1}{Dc+Ni}$

优点：

（1） 算法逻辑简单,易于实现

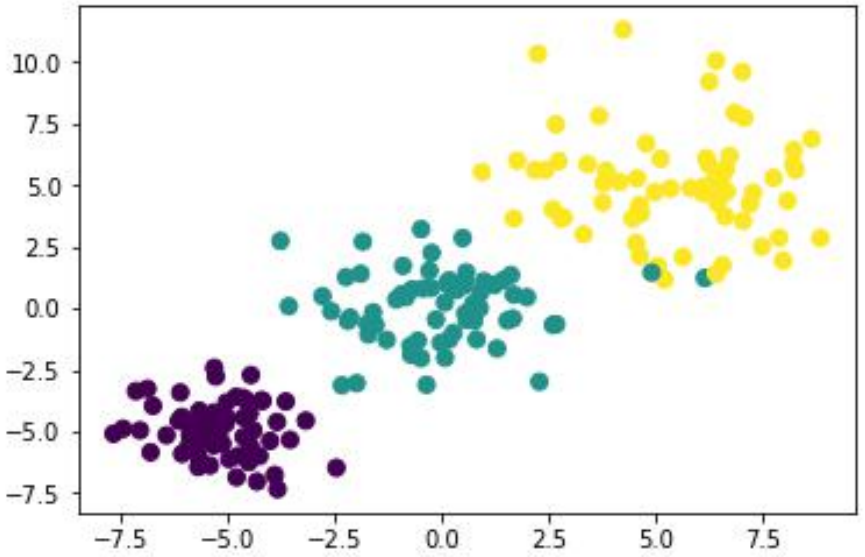
（2）分类过程中时空开销小

缺点：

理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。

2、贝叶斯分类简单应用

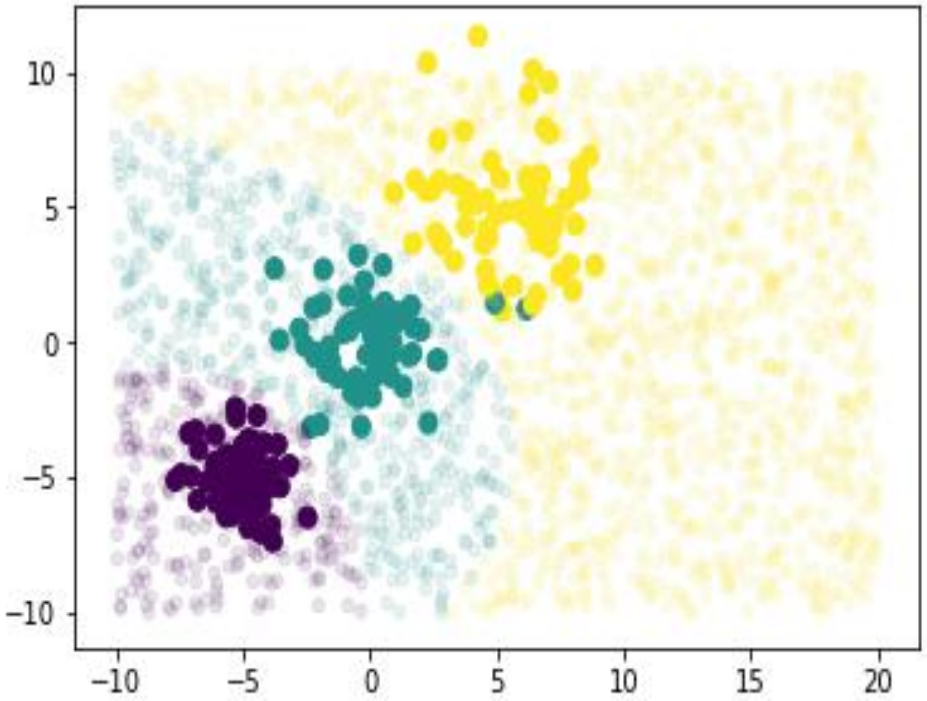
scikit-learn机器学习包提供3个朴素贝叶斯分类算法：
GaussianNB(高斯朴素贝叶斯) GaussianNB
MultinomialNB(多项式朴素贝叶斯)MultinomialNB
BernoulliNB(伯努利朴素贝叶斯) BernoulliNB
高斯朴素贝叶斯假设每一类数据都服从简单高斯分布




```
from sklearn.datasets import make_blobs#生成聚类测试数据
import matplotlib.pyplot as plt
data, target = make_blobs(n_samples=200,n_features=2,random_state=2,
                          centers=[[-5,-5],[0,0],[5,5]],cluster_std=[1.0,1.5,2.0])
#每个类别服从独立高斯分布，在2D图中绘制样本，每个样本颜色不同
plt.scatter(data[:,0],data[:,1],c=target)
plt.show()
```

2、贝叶斯分类简单应用

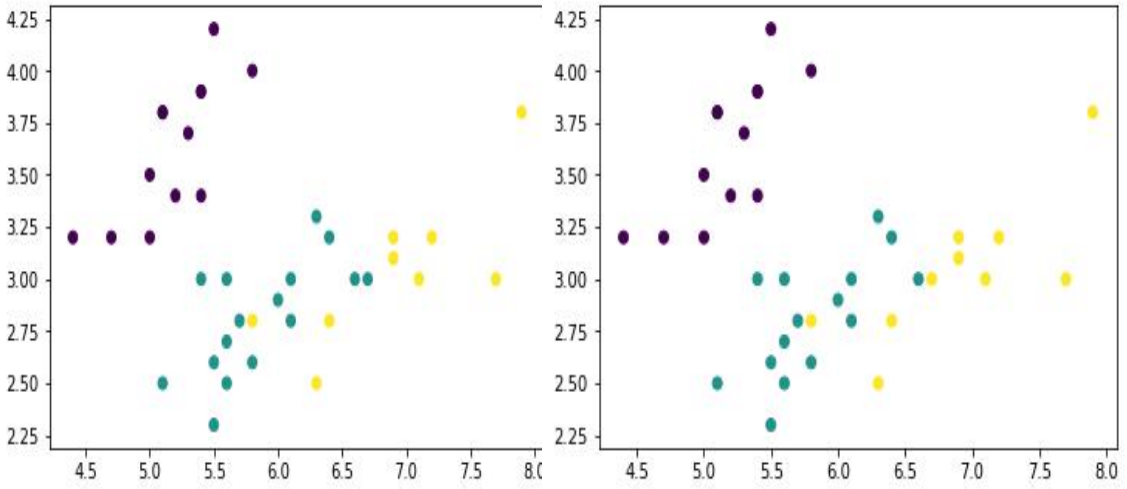
```
from sklearn.naive_bayes import GaussianNB  #导入高斯贝叶斯分类函数
model = GaussianNB()
model.fit(data,target)                  #训练
rng = np.random.RandomState(0)
Xnew = [-10,-10]+[30,20]* rng.rand(2000,2)#产生2000个二维随机数组测试数据
ynew = model.predict(Xnew)
plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, alpha=0.1)
```




```
print(model.predict([[3,4]]))  #预测[3，4]的分类
#array([2])
print(model.predict_proba([[3,4]]).round(2))  #输出概率分布
#[[0.   0.02 0.98]]
```

4、鸢尾花数据集降维可视化及贝叶斯分类训练

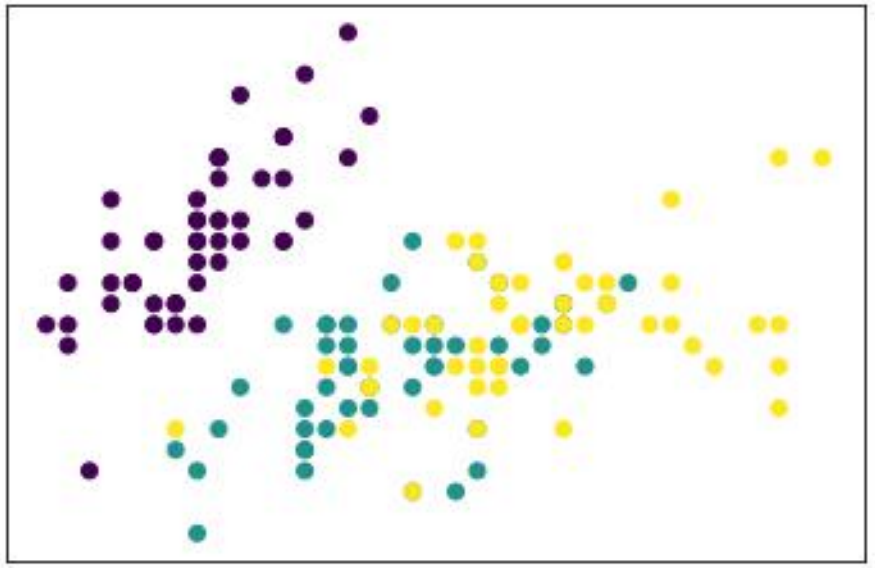
```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
#载入数据集
iris = load_iris()
xtrain,xtest,ytrain,ytest=train_test_split(iris.data,iris.target,random_state=1) #自动分割数据集
# train 112条数据，test38条
model = GaussianNB()                    #可以看出最简单的分类算法也可以有效的学习这个数据集
model.fit(xtrain,ytrain)
y_pre=model.predict(xtest)
print(accuracy_score(y_pre,ytest))  #两个数组的相似度。 0.9736842105263158
plt.figure(1)
plt.scatter(xtest[:,0],xtest[:,1],c=ytest)
plt.figure(2)
plt.scatter(xtest[:,0],xtest[:,1],c=y_pre)
```

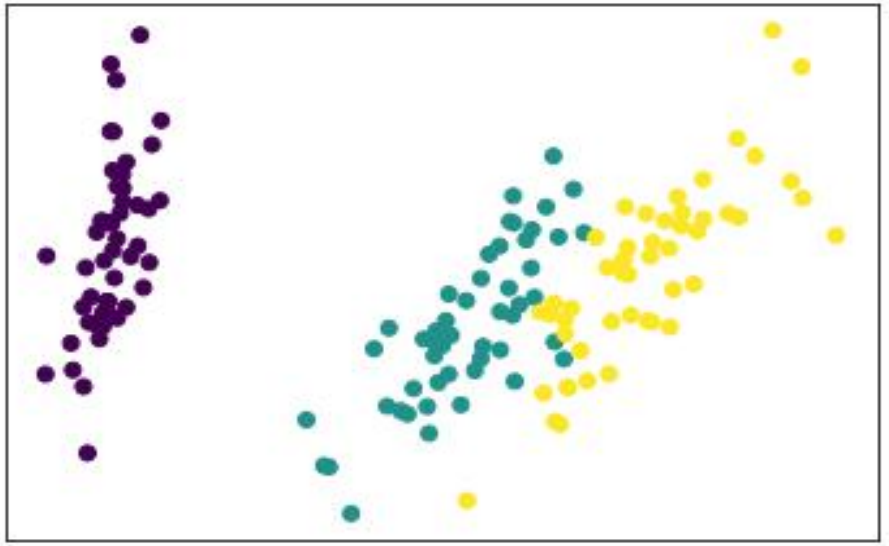


4、鸢尾花数据集降维可视化及贝叶斯分类训练

xtrain与xtest均为4个属性，表示鸢尾花的4个特征。
plt.scatter(xtest[:,0],xtest[:,1],c=ytest) 只使用了前两个。
可以通过PCA降低特征维数

```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)     #n_components=.98) 也可以约定拟合度 为0.98
X=pca.fit_transform(iris.data)  #降维2个属性。
Y=iris.target
X_all=iris.data
print(pca.explained_variance_ratio_)
#[0.92461621 0.05301557]属性原始数据拟合度
plt.figure(3) # 创建图表1
plt.scatter(X_all[:,0],X_all[:,1],c=Y)
plt.figure(4)# 创建图表2
plt.scatter(X[:,0],X[:,1],c=Y)
plt.show()
```





5、朴素贝叶斯的应用场景

朴素贝叶斯分类器对数据有严格的假设，训练效果通常比复杂模型要差。
优点：
- 训练和预测速度非常快
- 直接使用概率预测
- 容易理解
- 可调参数少

适用于以下场景：
- 分布函数与数据匹配（实际很少匹配）
- 类型之间区分度很高
- 数据特征维度非常高

随着数据维度的增大，简单模型也一样很强大。

目录

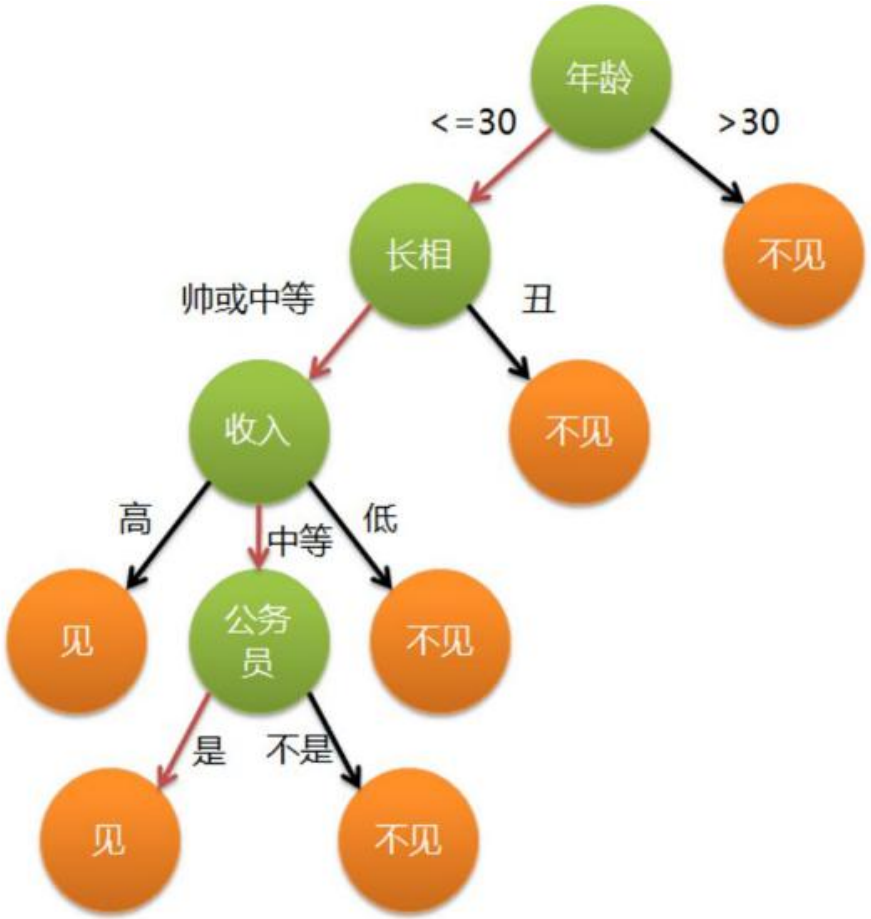
1、决策树分类简介

- **决策树**是用于分类和预测的主要技术之一，决策树学习是以实例为基础的归纳学习算法，它着眼于从一组无次序、无规则的实例中推理出以决策树表示的分类规则。
- 构造决策树的目的是找出**属性和类别**间的关系，用它来预测将来未知类别的记录的类别。
- 它采用**自顶向下的递归**方式，在决策树的内部节点进行属性的比较，并根据不同属性值判断从该节点向下的分支，在决策树的叶节点得到结论。
- 常见的算法包括：分类及回归树， ID3， C4.5，随机森林（Random Forest），CART。

- 决策数有两大优点：1）决策树模型可以读性好，具有描述性，有助于人工分析；2）效率高，决策树只需要一次构建，反复使用，每一次预测的最大计算次数不超过决策树的深度。

1、决策树分类简介

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。



这个女孩的决策过程就是典型的分类树决策。相当于通过年龄、长相、收入和是否公务员对将男人分为两个类别：见和不见。

假设这个女孩对男人的要求是：30岁以下、长相中等以上并且是高收入者或中等以上收入的公务员，那么这个可以用下图表示女孩的决策逻辑。

1、决策树分类简介

决策树的构建不是唯一的，遗憾的是最优决策树的构建属于NP问题。只能得到局部最优解，通常使用信息熵作为评价因子。熵即混乱、无序的状态。$H=-p*\log_2 P$

样本	红	大	好苹果
0	1	1	1
1	1	0	1
2	0	1	0
3	0	0	0

右图中要构建苹果分类的决策树，P为是否好苹果的概率。

未分类前:$H= -(1/2 * \log_2(1/2) + 1/2 * \log_2(1/2)) = 1$

如按颜色分类：

0，1叶子节点有2个正例，0个负例。

信息熵为：$e1 = -(2/2 * \log_2(2/2) + 0/2 * \log_2(0/2)) = 0$。

2，3叶子节点有0个正例，2个负例。

信息熵为：$e2 = -(0/2 * \log_2(0/2) + 2/2 * \log_2(2/2)) = 0$。

$E = e1*2/4 + e2*2/4 = 0$。

如先选大小分类：

0，2子节点有1个正例，1个负例。

信息熵为：$e1 = -(1/2 * \log_2(1/2) + 1/2 * \log_2(1/2)) = 1$。

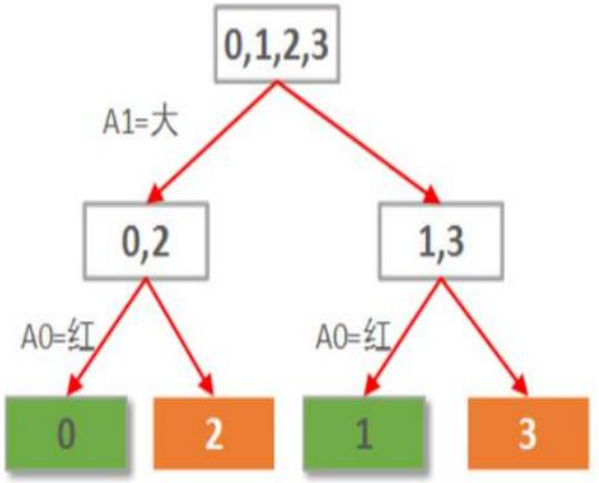
1，3子节点有1个正例，1个负例。

信息熵为：$e2 = -(1/2 * \log_2(1/2) + 1/2 * \log_2(1/2)) = 1$。

$E = e1*2/4 + e2*2/4 = 1$。也就是说分了跟没分一样！

样本中有2个属性，A0表示是否红苹果。A1表示是否大苹果。

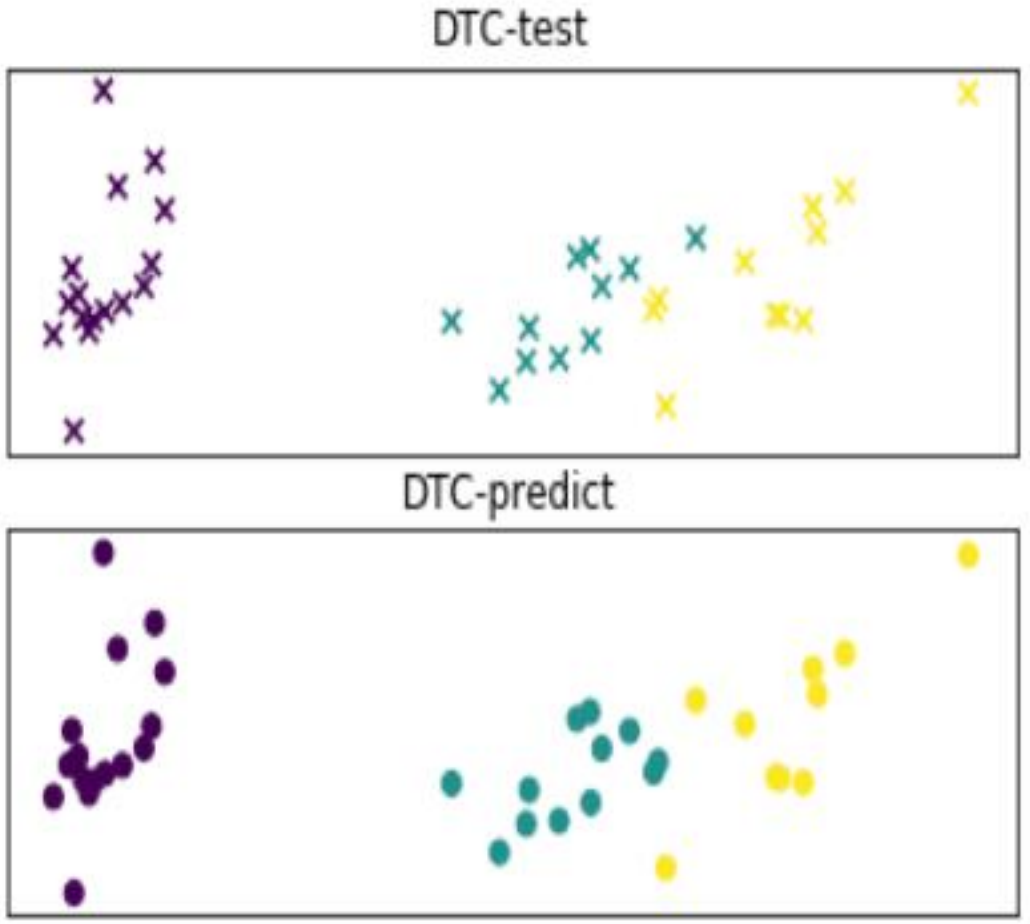
本例仅2个属性。那么很自然一共就只可能有2棵决策树，如下图所示：



每次划分之前，我们只需要计算出信息熵增益最大的那种划分即可。

2、决策树分类实例

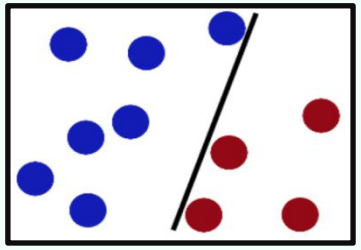
```
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
iris = load_iris()  #载入数据集
from sklearn.tree import DecisionTreeClassifier  #导入决策树DTC包
pca=PCA(n_components=2)   #降维
X=pca.fit_transform(iris.data)  #降维2个属性。
Y=iris.target
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,random_state=2) #分割数据集
clf = DecisionTreeClassifier()
clf.fit(xtrain, ytrain)
y_pre = clf.predict(xtest)#预测
print(accuracy_score(ytest,y_pre))   #0.9210526315789473比贝叶斯要低
fig, ax = plt.subplots(2)
ax[0].scatter(xtest[:,0], xtest[:,1], c=ytest, marker='x')
ax[0].set_title('DTC-test')
ax[0].set_xticks([])
ax[0].set_yticks([])
ax[1].scatter(xtest[:,0], xtest[:,1], c=y_pre, marker='o')
ax[1].set_title('DTC-predict')
ax[1].set_xticks([]) ;   ax[1].set_yticks([])
```



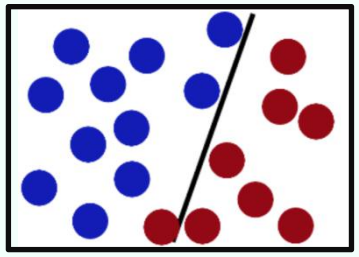
SVM分类

在很久以前的情人节，大侠要去救他的爱人，但魔鬼和他玩了一个游戏。

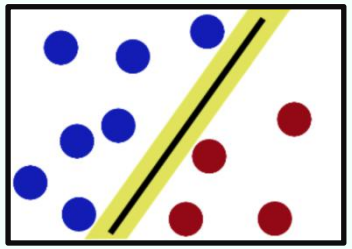
1. 魔鬼在桌子上似乎有规律放了两种颜色的球，说：“你用一根棍分开它们？要求：尽量在放更多球之后，仍然适用。”
于是大侠这样放，干的似乎还不错？



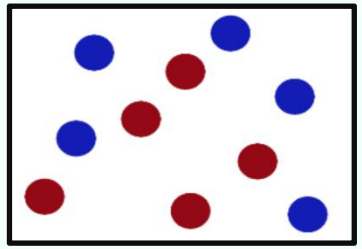
2. 然后魔鬼，又在桌上放了更多的球，似乎有一个球站错了阵营。



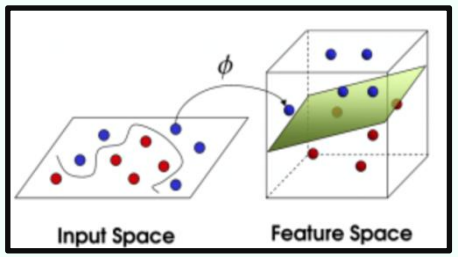
3. SVM就是试图把棍放在最佳位置，好让在棍的两边有尽可能大的间隙。

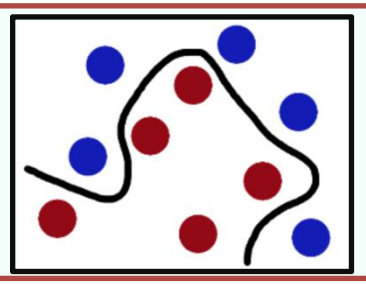


4. 魔鬼看到大侠已经学会了一个trick，于是魔鬼给了大侠一个新的挑战。



5. 现在，大侠没有棍可以很好帮他分开两种球了，现在怎么办呢？当然像所有武侠片中一样大侠桌子一拍，球飞到空中。然后，凭借大侠的轻功，抓起一张纸，插到了两种球的中间。



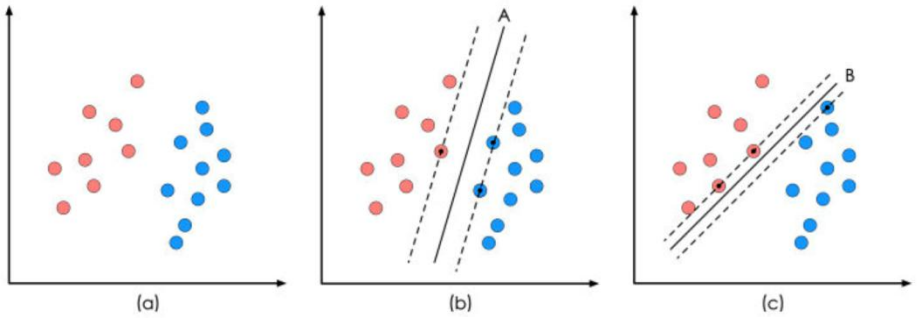


再之后，人们把这些球叫做「**data**」，把棍子叫做「**classifier**」，最大间隙trick叫做「**optimization**」，拍桌子叫做「**kernelling**」，那张纸叫做「**hyperplane**」。

SVM分类

支持向量机（Support Vector Machine ,SVM）是一种有监督学习方法，主要思想是建立一个最优决策超平面，使得该平面两侧距离该平面最近的两类样本之间的距离最大化，从而对分类问题提供良好的泛化能力。

线性可分支持向量机与硬间隔最大化：



原问题：

$$\max_{\omega,b} \gamma$$

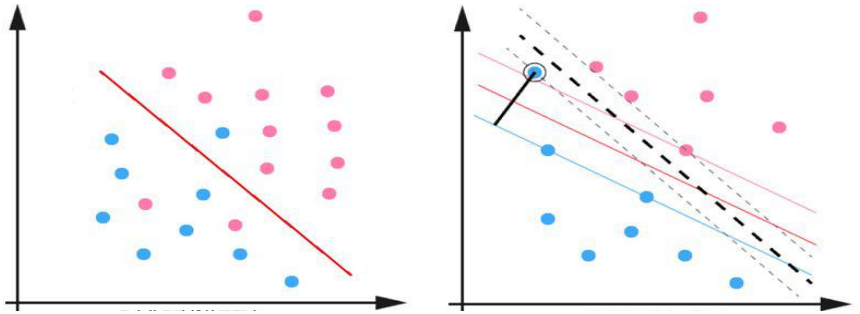
$$s.t.\ y_i\left(\frac{\omega}{||\omega||}\cdot x_i+\frac{b}{||\omega||}\right)\geq\gamma, i=1,2,\ldots,N$$

对偶问题：

$$\min \frac{1}{2}||\omega||^2$$

$$s.t.\ y_i(\omega\cdot x_i+b)-1\geq 0, i=1,2,\ldots,N$$

线性可分支持向量机与软间隔最大化：



对偶问题：

$$\min_{\omega,b,\xi} \frac{1}{2}||\omega||^2+C\sum_{i=1}^{N}\xi_i$$

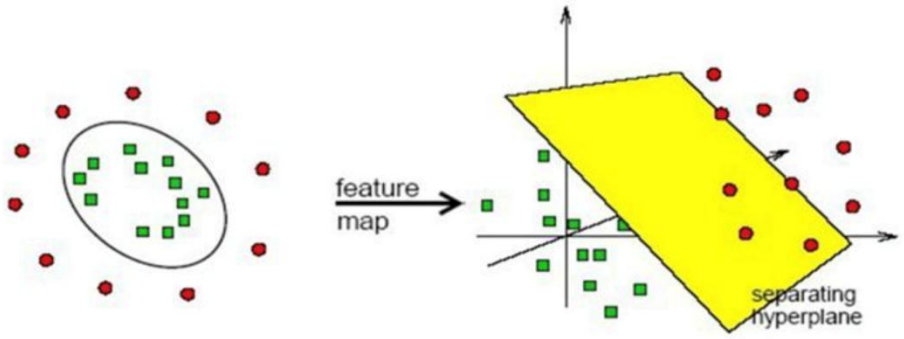
$$s.t.\ y_i(\omega\cdot x_i+b)\geq 1-\xi_i, i=1,2,\ldots,N$$

$$\xi_i, i=1,2,\ldots,N\geq 0$$

SVM分类

支持向量机（Support Vector Machine ,SVM）是一种有监督学习方法，主要思想是建立一个最优决策超平面，使得该平面两侧距离该平面最近的两类样本之间的距离最大化，从而对分类问题提供良好的泛化能力。

非线性支持向量机与核函数：



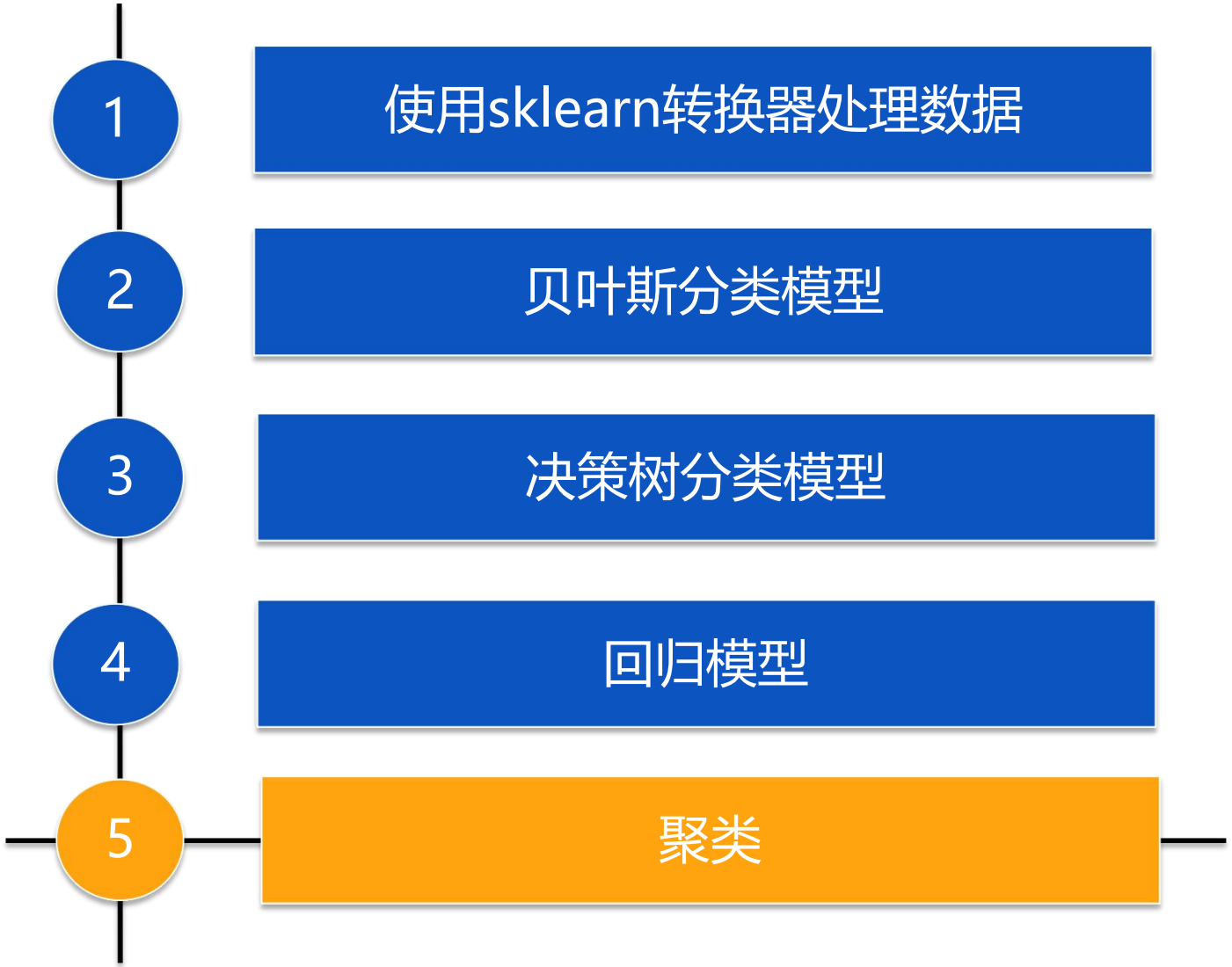
- 线性核函数：$K(x,z)=x\cdot z$
- 多项式核函数：$K(x,z)=(x\cdot z+1)^p$
- 高斯核函数：$K(x,z)=\exp(-\frac{\|x-z\|^2}{2\sigma^2})$
- 混合核：$\lambda K_1(x,z)+(1-\lambda)K_2(x,z), 0\leq\lambda<1$

SVM的优点：

- ✓ 相对于其他训练分类算法**不需要过多样本**，并且由于SVM引入了核函数，所以SVM可以处理高维样本
- ✓ **结构风险最小**。这种风险是指分类器对问题真实模型的逼近与问题真实解之间的累积误差
- ✓ **非线性**，是指SVM擅长应付样本数据线性不可分的情况，主要通过松弛变量（也叫惩罚变量）和核函数技术来实现，这一部分也正是SVM的精髓所在。

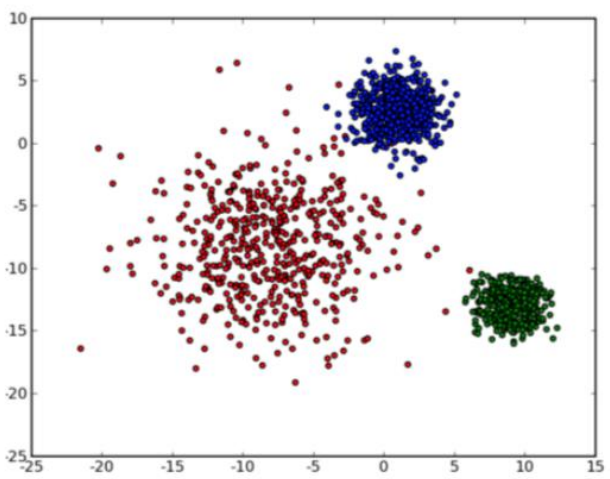
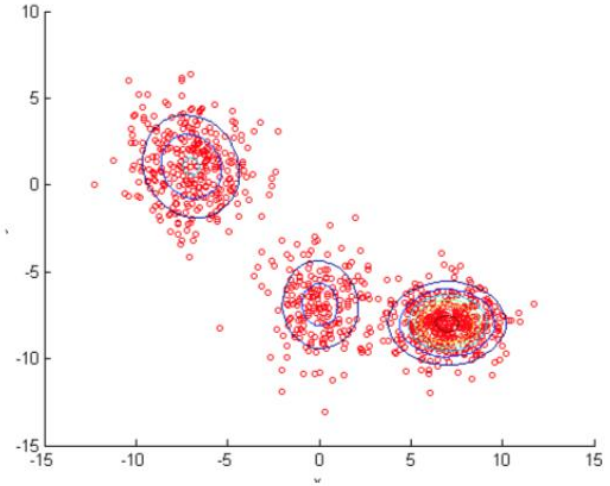
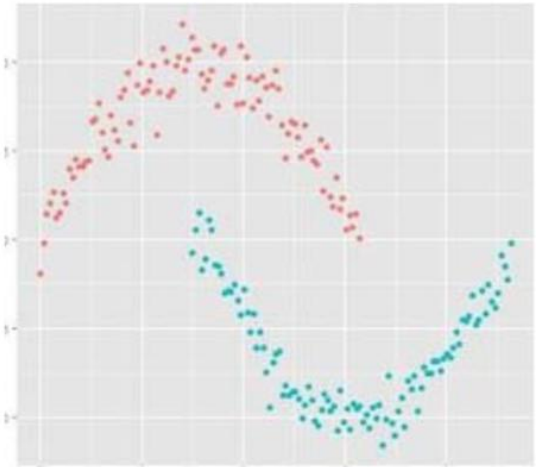
常用软件工具包：
LibSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
SVM-Light：http://svmlight.joachims.org/
Liblinear:http://www.csie.ntu.edu.tw/~cjlin/liblinear/

目录



聚类问题

聚类问题是无监督学习的问题，算法的思想就是“物以类聚，人以群分”。聚类算法感知样本间的相似度，进行类别归纳，对新的输入进行输出预测，输出变量取有限个离散值。

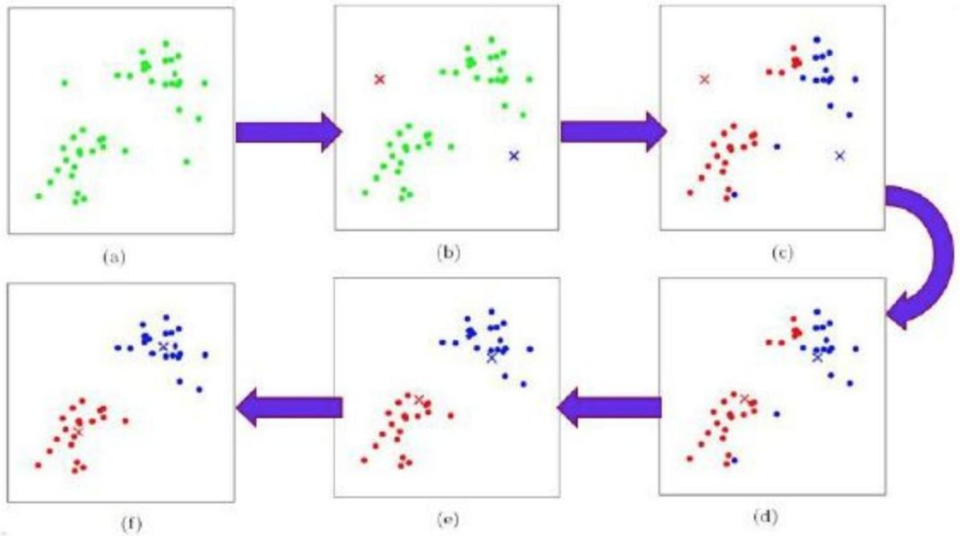
- ✓ 可以作为一个单独过程，用于寻找数据内在的分布结构
- ✓ 可以作为分类、稀疏表示等其他学习任务的前驱过程

K-means

K-means（又称k-均值或k-平均）聚类算法。算法思想就是首先随机确定k个中心点作为聚类中心，然后把各个数据点分配给最邻近的中心点，分配完成后将中心点移动到所表示的聚类的平均中心位置处，然后重复迭代上述步骤直到分配过程不再产生变化位置。

K-means算法流程

① 随机选择K个随机的点（称为聚类中心）；
② 对与数据集中的每个数据点，按照距离K个中心点的距离，将其与距离最近的中心点关联起来，与同一中心点关联的所有点聚成一类；
③ 计算每一组的均值，将该组所关联的中心点移动到平均值的位置；
④ 重复执行2-3步，直至中心点不再变化；



K-Means的主要**优点**：
- ✓ 原理比较简单，实现也是很容易，收敛速度快
- ✓ 聚类效果较优
- ✓ 算法的可解释度比较强
- ✓ 主要需要调参的参数仅仅是簇数k

K-Means的主要**缺点**：
- ✓ K值的选取不好把握
- ✓ 不平衡数据集的聚类效果不佳
- ✓ 采用迭代方法，得到的结果只是局部最优
- ✓ 对噪音和异常点比较的敏感。

1、聚类简介

K-Means是聚类算法中的最常用的一种，算法最大的特点是简单，好理解，运算速度快，但是只能应用于数值型的数据，且要在聚类前指定分成几类。 k-modes 可处理非数值
我们描述一下K-means算法的过程，为了尽量不用数学符号，所以描述的不是很严谨，大概就是这个意思，“物以类聚、人以群分”：

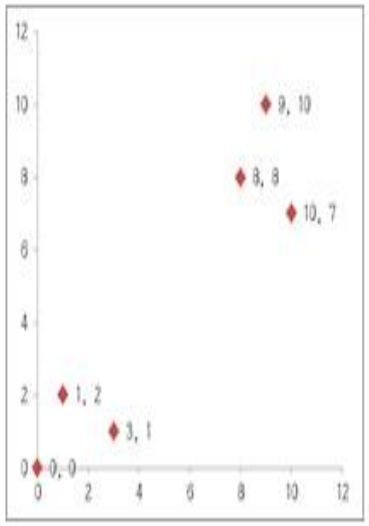
1、首先输入k的值，即我们希望将数据集经过聚类得到k个分组。
2、从数据集中随机选择k个数据点作为初始大哥（质心，Centroid）
3、对集合中每一个小弟，计算与每一个大哥的距离（距离的含义后面会讲），离哪个大哥距离近，就跟定哪个大哥。
4、这时每一个大哥手下都聚集了一票小弟，这时候召开代表大会，每一群选出新的大哥（其实是通过算法选出新的质心）。
5、如果新大哥和老大哥之间的距离小于某一个设置的阈值（表示重新计算的质心的位置变化不大，趋于稳定，或者说收敛），可以认为我们进行的聚类已经达到期望的结果，算法终止。
6、如果新大哥和老大哥距离变化很大，需要迭代3~5步骤。

1、聚类简介

简单的手算例子#####

我搞了6个点，从图上看应该分成两推儿，前三个点一堆儿，后三个点是另一堆儿。现在手工执行K-Means，体会一下过程，同时看看结果是不是和预期一致。

	X	Y
P1	0	0
P2	1	2
P3	3	1
P4	8	8
P5	9	10
P6	10	7



case

1.选择初始大哥：

我们就选P1和P2

2.计算小弟和大哥的距离：

P3到P1的距离从图上也能看出来（勾股定理），是√10 = 3.16；P3到P2的距离√((3-1)$^{2+(1-2)}$2 = √5 = 2.24，所以P3离P2更近，P3就跟P2混。同理，P4、P5、P6也这么算，如下：

	P1	P2
P3	3.16	2.24
P4	11.3	9.22
P5	13.5	11.3
P6	12.2	10.3

round1

P3到P6都跟P2更近，所以第一次站队的结果是：

- 组A：P1
- 组B：P2、P3、P4、P5、P6

1、聚类简介

K-Means的细节问题

1、K值怎么定？我怎么知道应该几类？

答：这个真的没有确定的做法，分几类主要取决于个人的经验与感觉，通常的做法是多尝试几个K值，看分成几类的结果更好解释，更符合分析目的等。或者可以把各种K值算出的SSE做比较，取最小的SSE的K值。

2、初始的K个质心怎么选？

答：最常用的方法是随机选，初始质心的选取对最终聚类结果有影响，因此算法一定要多执行几次，哪个结果更reasonable，就用哪个结果。

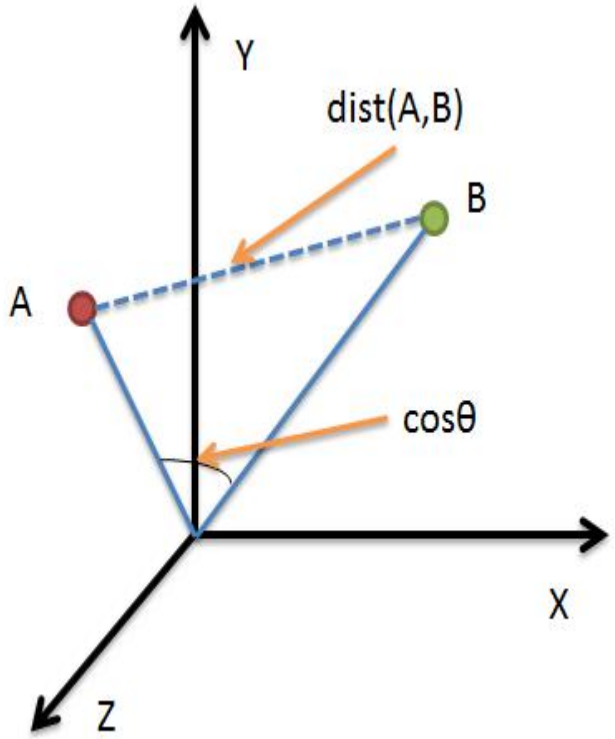
3、K-Means会不会陷入一直选质心的过程，永远停不下来？

答：不会，有数学证明K-Means一定会收敛，大致思路是利用SSE的概念（也就是误差平方和），即每个点到自身所归属质心的距离的平方和，这个平方和是一个函数，然后能够证明这个函数是可以最终收敛的函数。

4、判断每个点归属哪个质心的距离怎么算？

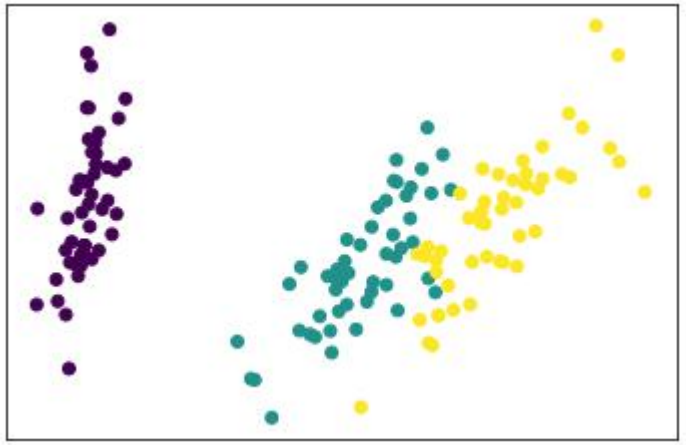
答：第一种，欧几里德距离

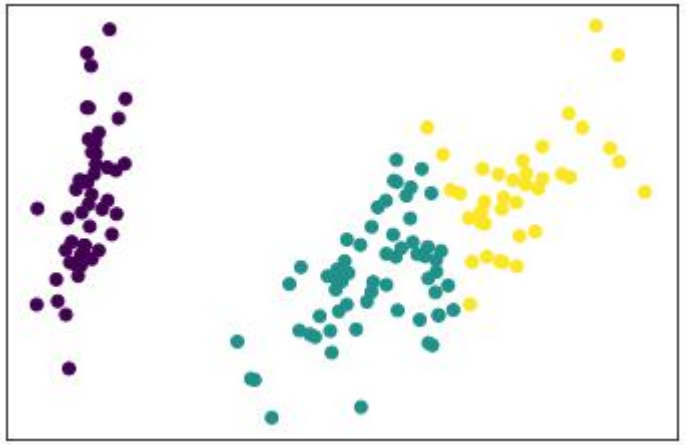
第二种，余弦相似度，余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。



2、鸢尾花聚类分析

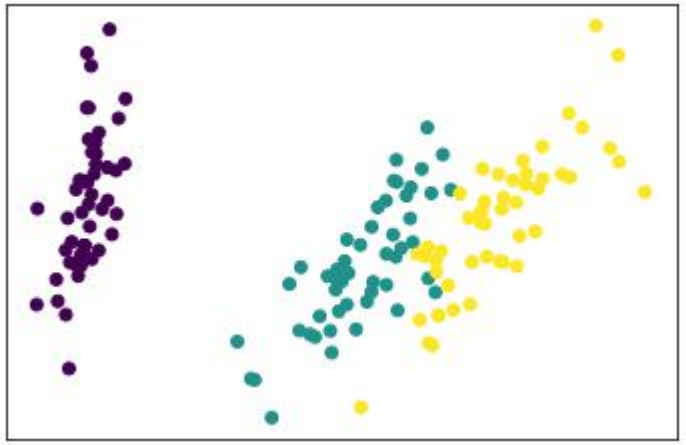
```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
iris = load_iris()
pca=PCA(n_components=2)     #n_components=.98)
xtrain=pca.fit_transform(iris.data)  #降维2个属性。
print(pca.explained_variance_ratio_)
ytrain=iris.target
model = KMeans(n_clusters=3)
model.fit(Xtrain)
label_pred = model.labels_  # 获取聚类标签
plt.figure(1)
plt.scatter(xtrain[:,0],xtrain[:,1],c=ytrain)
plt.xticks(())
plt.yticks(())
plt.figure(2)
plt.scatter(xtrain[:,0],xtrain[:,1],c=label_pred)
plt.xticks(())  ;plt.yticks(())
```

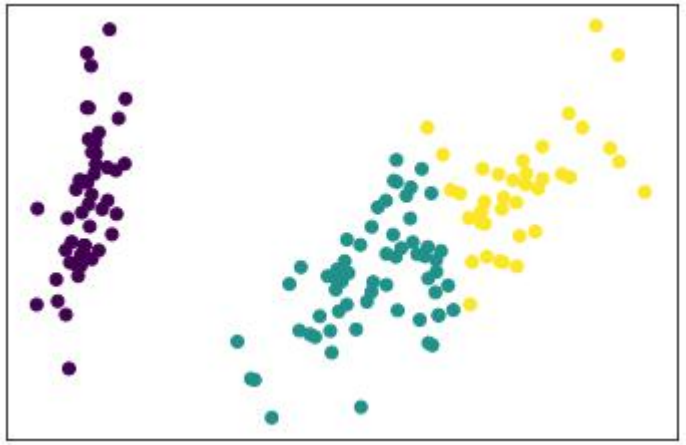




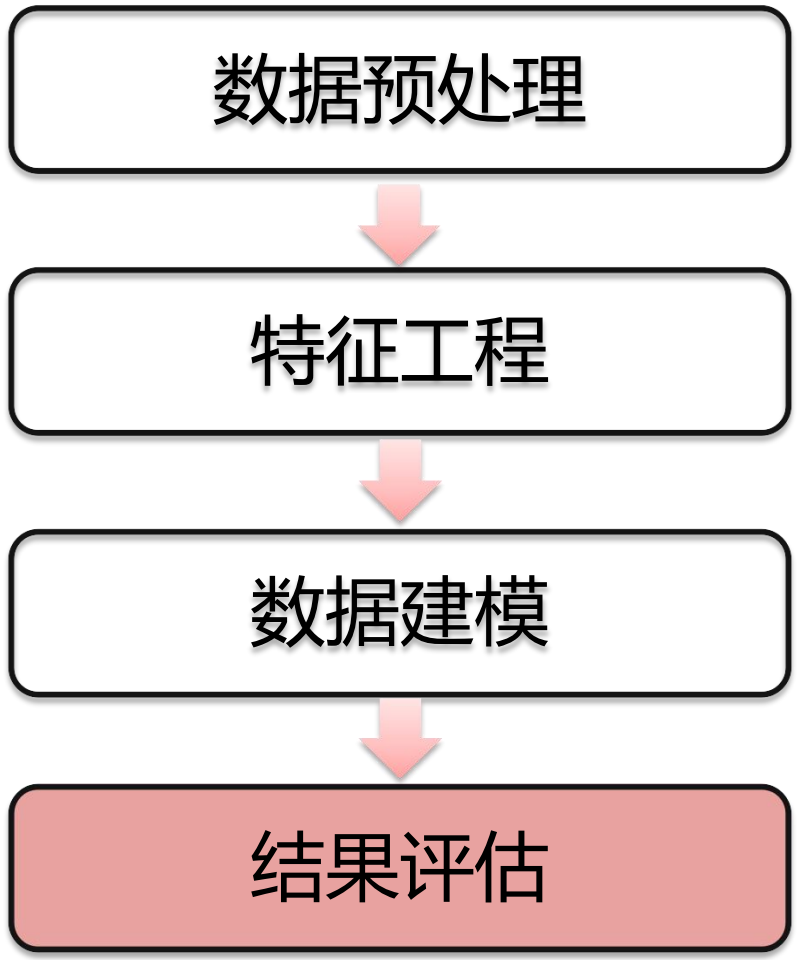
3、KNN算法分析

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
iris = load_iris()
pca=PCA(n_components=2)     #n_components=.98)
xtrain=pca.fit_transform(iris.data)  #降维2个属性。
print(pca.explained_variance_ratio_)
ytrain=iris.target
model = KMeans(n_clusters=3)
model.fit(Xtrain)
label_pred = model.labels_  # 获取聚类标签
plt.figure(1)
plt.scatter(xtrain[:,0],xtrain[:,1],c=ytrain)
plt.xticks(())
plt.yticks(())
plt.figure(2)
plt.scatter(xtrain[:,0],xtrain[:,1],c=label_pred)
plt.xticks(())  ;plt.yticks(())
```





机器如何学习



数据清洗、数据采样、数据集拆分

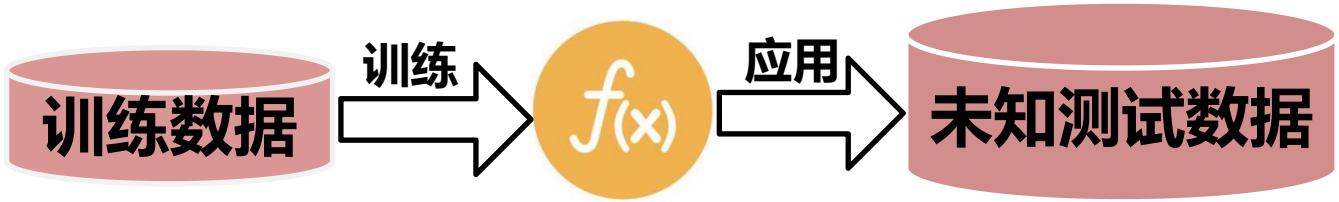
特征编码、特征选择、特征降维、规范化

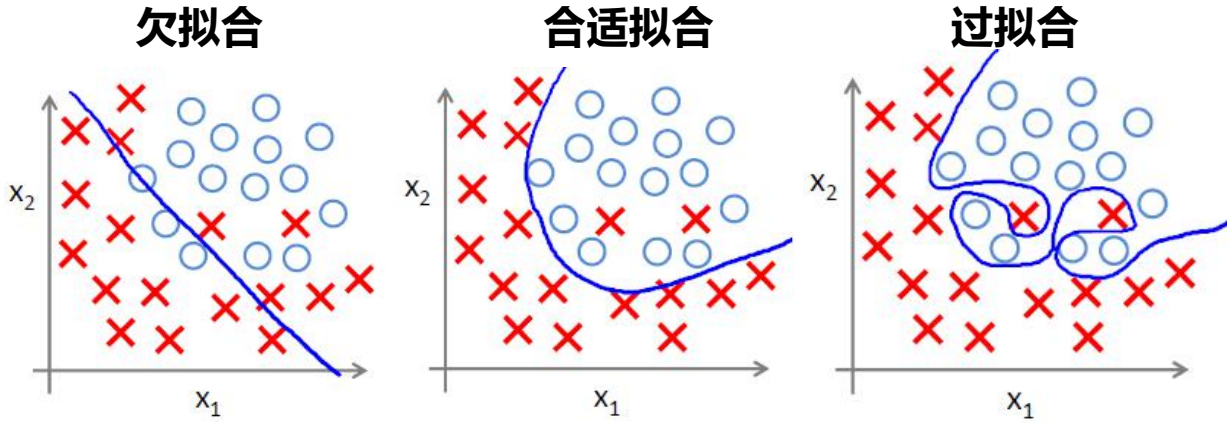
回归问题、分类问题、聚类问题、其他问题、开源框架

拟合度量、查准率、查全率、F1值、PR曲线、ROC曲线

模型选择

泛化误差：在“未来”样本上的误差
经验误差：在训练集上的误差





- ✓ AIC准则（AkalikeInformation Criterion）

$$AIC = 2k - 2\ln(L)$$

- ✓ BIC准则（BayesianInformation Criterion）

$$BIC = k\ln(n) - 2\ln(L)$$

性能评价指标-分类

准确率(Accuracy)是指在分类中，分类正确的记录个数占总记录个数的比。

$$accuracy = \frac{n_{correct}}{n_{total}}$$

召回率(Recall)也叫查全率，是指在分类中样本中的正例有多少被预测正确了。

通常，准确率高时，召回率偏低；召回率高时，准确率偏低。

1. 地震的预测

对于地震的预测，我们希望的是召回率非常高，也就是说每次地震我们都希望预测出来。这个时候我们可以牺牲准确率。情愿发出1000次警报，把10次地震都预测正确了；也不要预测100次对了8次漏了两次。

2. 嫌疑人定罪

基于不错怪一个好人的原则，对于嫌疑人的定罪我们希望是非常准确的。及时有时候放过了一些罪犯（召回率低），但也是值得的。

性能评价指标-分类

准确率(Accuracy)：分类正确的样本个数占所有样本个数的比例

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

平均准确率(Average per-class accuracy)：每个类别下的准确率的算术平均

$$average\_accuracy = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}$$

精确率(Precision)：分类正确的正样本个数占分类器所有的正样本个数的比例

$$Precision = \frac{TP}{TP + FP}$$

召回率(Recall)：分类正确的正样本个数占正样本个数的比例

$$Recall = \frac{TP}{TP + FN}$$

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

F1-Score：精确率与召回率的调和平均值，它的值更接近于Precision与Recall中较小的值

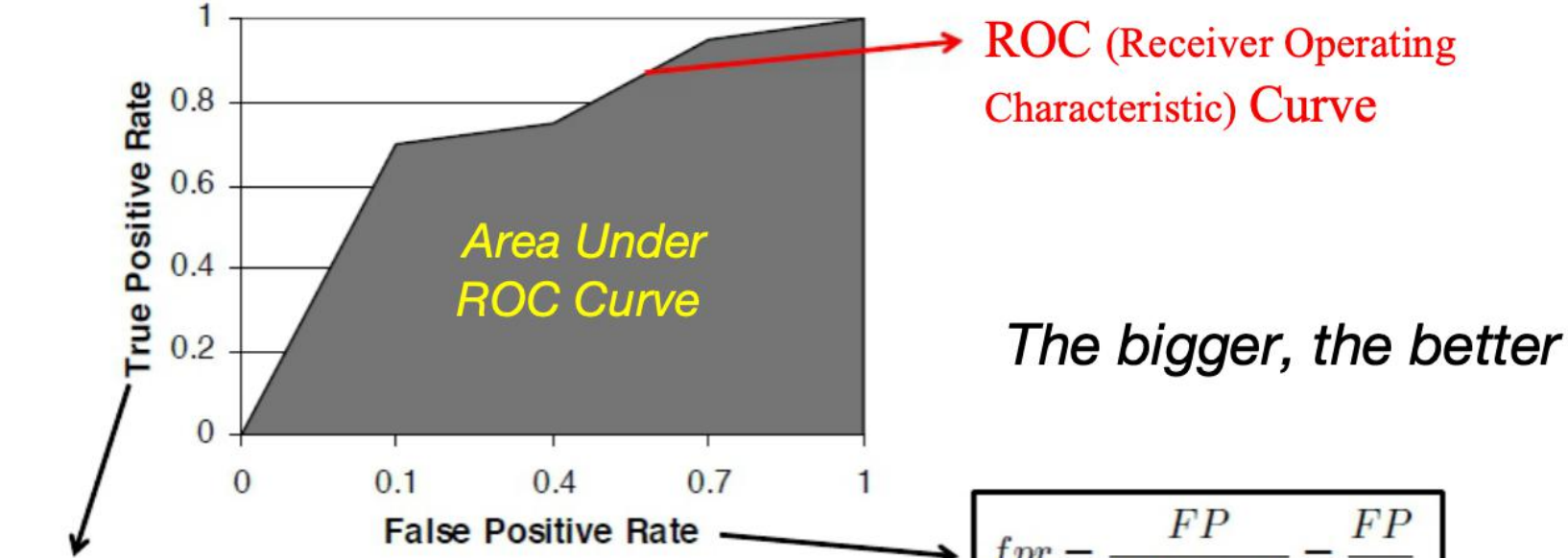
$$F1 = \frac{2 * precision * recall}{precision + recall}$$

性能评价指标-分类

AUC(Area under the Curve(Receiver Operating Characteristic, ROC))

ROC：纵轴：真正例率TPR；横轴：假正例率FPR

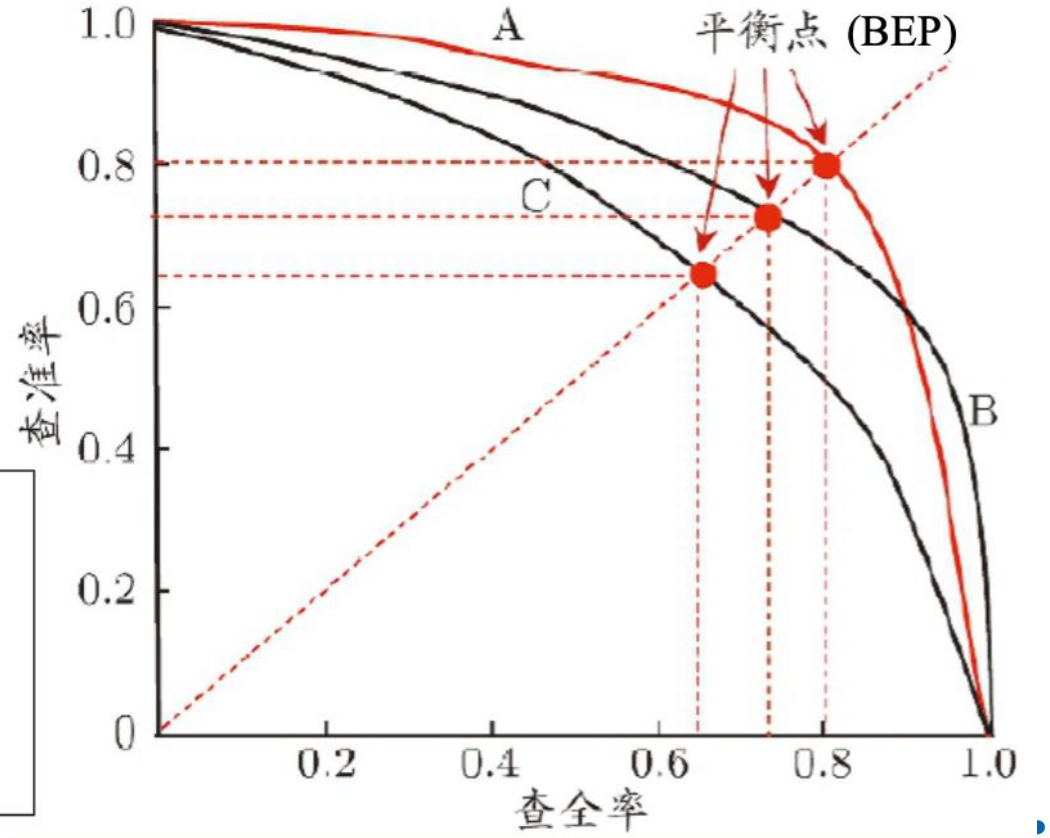
AUC是ROC曲线下的面积。一般来说，如果ROC是光滑的，那么基本可以判断没有太大的overfitting，这个时候调模型可以只看AUC，面积越大一般认为模型越好。



性能评价指标-分类

PR曲线：根据学习器的预测结果按正例可能性大小对样例进行排序，并逐个把样本作为正例进行预测。

- ✓ 如果一个学习器的**PR曲线**包住了另一个，则可以认为A的性能优于C
- ✓ 如果有交叉，如A、B，综合考虑PR性能，引入**平衡点(BEP)**，基于BEP比较，A优于B



PR图：

- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C
- 学习器 A ?? 学习器 B

BEP：

- 学习器 A 优于 学习器 B
- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C

性能评价指标-分类

宏平均&微平均

多分类问题中，若能得到**多个混淆矩阵**，例如多次训练/测试的结果，多分类的两两混淆矩阵：

宏(macro-)查准率、查全率、F1

$$\text{macro-}P = \frac{1}{n}\sum_{i=1}^{n} P_i \ ,$$

$$\text{macro-}R = \frac{1}{n}\sum_{i=1}^{n} R_i \ ,$$

$$\text{macro-}F1 = \frac{2 \times \text{macro-}P \times \text{macro-}R}{\text{macro-}P + \text{macro-}R} \ .$$

微(micro-)查准率、查全率、F1

$$\text{micro-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}} \ ,$$

$$\text{micro-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}} \ ,$$

$$\text{micro-}F1 = \frac{2 \times \text{micro-}P \times \text{micro-}R}{\text{micro-}P + \text{micro-}R} \ .$$

性能评价指标-分类

对数损失（Log loss）亦被称为逻辑回归损失（Logistic regression loss）或交叉熵损失（Cross-entropy loss）。

二分类问题： $y \in \{0,1\}$ 且 $p = \Pr(y=1)$ 则对每个样本的对数损失为

$$L_{\log}(y,p) = -\log \Pr(y|p) = -(y\log(p) + (1-y)\log(1-p))$$

多分类问题： 设Y为指示矩阵，即当样本i的分类为k，$y_{i,k} = 1$ 设P为估计的概率矩阵，$p_{i,k} = \Pr(t_{i,k} = 1)$ 则对每个样本的对数损失为

$$L_{\log}(Y_i, P_i) = -\log \Pr(Y_i|P_i) = \sum_{k=1}^{K} y_{i,k} \log p_{i,k}$$

性能评价指标-回归

平均绝对误差：平均绝对误差MAE（Mean Absolute Error）又被称为l1范数损失（l1-norm loss）

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} |y_i - \hat{y}_i|$$

平均平方误差：平均平方误差MSE（Mean Squared Error）又被称为l2范数损失（l2-norm loss）：

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} (y_i - \hat{y}_i)^2$$

均方根差RMSE:

R Squared:是将预测值跟只使用均值的情况下相比，看能好多少。

$$R^2 = 1 - \frac{(\sum_i (\hat{y}_i - y_i)^2)/m}{(\sum_i (\hat{y}_i - y_i)^2)/m} = 1 - \frac{MSE(\hat{y}, y)}{Var(y)}$$

性能评价指标-聚类

外部指标对数据集D=$\{x_1, x_2, \ldots, x_m\}$，假定通过聚类给出的簇划分为 $C=\{C_1, C_2, \ldots, C_k\}$ 参考模型给出的簇划分为 $C^*=\{C_1^*, C_2^*, \ldots, C_k^*\}$，通过比对C和 C^* 来判定聚类结果的好坏。

Jaccard系数， FM 指数， Rand 指数，纯度purity，熵 entropy，互信息，Adjusted Rand Index (ARI)， F-measure， Probabilistic Rand Index (PRI)

内部指标对聚类数据结构上的描述，类内距离小，类间距离大较好。

DB 指数(Davies-Bouldin Index，简称DBI)：衡量同一簇中数据的紧密性，越小越好。

Dunn 指数(Dunn Index 简称DI)：衡量同一簇中数据的紧密性，越大越好。

Silouette：衡量同一簇中数据的紧密性，越大越好。

Modurity：衡量模块性，越大越好。

小结

本章主要根据数据分析的应用分类，重点介绍了对应的数据分析建模方法及实现过程。

- sklearn数据分析技术的基本任务主要体现在聚类、分类和回归三类。
- 每一类又有对应的多种评估方法，能够评价所构建模型的性能优劣。

通过这一章的学习，读者基本能够掌握常用的模型构建与评估方法，可在以后的数据分析过程中采用适当的算法并按所介绍的步骤实现综合应用。

Thank you!