

机器学习模型

Week3

1 Basic

1.1 Theory

1.1.1 贝叶斯原理（先验概率，后验概率，条件概率，似然函数，类别概率，条件概率）

$$P(B_i | A) = \frac{P(B_i)P(A | B_i)}{\sum_{i=1}^n P(B_i)P(A | B_i)}$$

贝叶斯推理告诉我们后验概率是与先验概率和似然函数成正比得

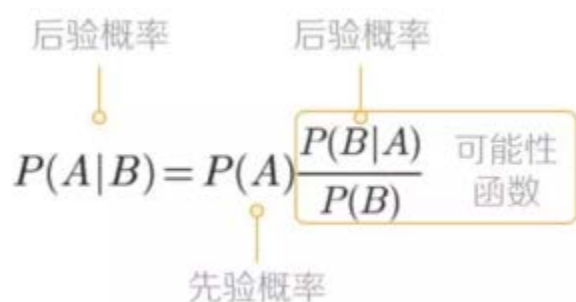
先验概率，通过经验来判断事情发生的概率

后验概率，就是发生结果之后，推测原因的概率

条件概率，事件 **A** 在另外一个事件 **B** 已经发生条件下的发生概率，表示为 $P(A|B)$

似然函数，把概率模型的训练过程理解为求参数估计的过程

比如，一个硬币在 10 次抛落中正面均朝上 => 那么这个硬币是均匀的可能性是多少？这里硬币均匀就是个参数，似然函数就是用来衡量这个模型的参数



The diagram shows the formula $P(A|B) = P(A) \frac{P(B|A)}{P(B)}$ with labels pointing to its components: '后验概率' (Posterior Probability) points to $P(A|B)$, '先验概率' (Prior Probability) points to $P(A)$, and '可能性函数' (Likelihood Function) points to $\frac{P(B|A)}{P(B)}$.

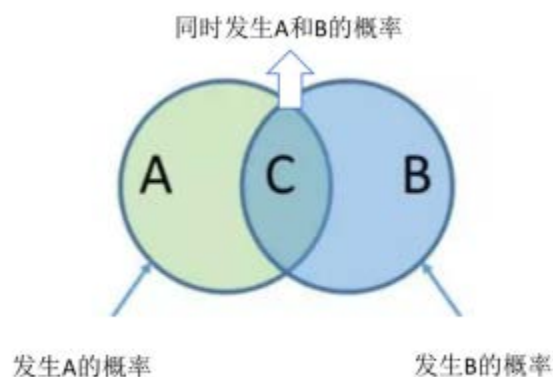
1.1.2 训练朴素贝叶斯模型的过程

朴素贝叶斯模型由两种类型的概率组成：每个类别的概率 $P(C_j)$ 每个属性的条件概率 $P(A_i/C_j)$

什么是类别概率：假设我有 7 个棋子，其中 3 个是白色的，4 个是黑色的 那么棋子是白色的概率就是 3/7，黑色的概率就是 4/7 => 这个是类别概率

什么是条件概率：假设我把这 7 个棋子放到了两个盒子里，其中盒子 A 里面有 2 个白棋，2 个黑棋；盒子 B 里面有 1 个白棋，2 个黑棋 那么在盒子 A 中抓到白棋的概率就是 1/2，抓到黑棋的概率也是 1/2，这个就是条件概率，也就是在某个条件（比如在盒子 A 中）下的概率

训练朴素贝叶斯模型的过程：在朴素贝叶斯中，我们要统计的是属性的条件概率，也就是假设取出来的是白色的棋子，那么它属于盒子A的概率是2/3 Step1：先给出训练数据，以及这些数据对应的分类 Step2，计算类别概率和条件概率 Step3，使用概率模型（基于贝叶斯原理）对新数据进行预测



1.1.3 朴素贝叶斯分类器

分类

离散值

根据身高，体重，鞋码，判断是否为男女，比如一个新的数据：身高“高”、体重“中”，鞋码“中” => 男 or 女？

求在 A1、A2、A3 属性下，Cj 的概率

$$P(C_j | A_1 A_2 A_3) = \frac{P(A_1 A_2 A_3 | C_j) P(C_j)}{P(A_1 A_2 A_3)}$$

因为一共有 2 个类别，所以我们只需要求得 P(C1|A1A2A3)和 P(C2|A1A2A3)的概率即可，然后比较下哪个分类的可能性大，就是哪个分类结果

分别求下这些条件下的概率：

$$P(A1|C1)=1/2, P(A2|C1)=1/2, P(A3|C1)=1/4$$

$P(A1|C2)=0$, $P(A2|C2)=1/2$, $P(A3|C2)=1/2$

编号	身高	体重	鞋码	性别
1	高	重	大	男
2	高	重	大	男
3	中	中	大	男
4	中	中	中	男
5	矮	轻	小	女
6	矮	轻	小	女
7	矮	中	中	女
8	中	中	中	女

所以 $P(A1A2A3|C1)=1/16$, $P(A1A2A3|C2)=0$

因为 $P(A1A2A3|C1)P(C1)>P(A1A2A3|C2)P(C2)$, 所以应该是 C1 类别, 即男性

连续值

朴素贝叶斯分类（连续值）：

身高 180、体重 120，鞋码 41，请问该人是男是女呢

可以假设男性和女性的身高、体重、鞋码都是正态分布，通过样本计算出均值和方差，也就是得到正态分布的密度函数。有了密度函数，就可以把值代入，算出某一点的值

比如，男性的身高是均值 179.5、标准差为 3.697 的正态分布。所以男性的身高为 180 的概率为 0.1069

同理可以计算得出男性体重为 120 的概率为 0.000382324，男性鞋码为 41 号的概率为 0.120304111

$P(A1A2A3|C1)=P(A1|C1)P(A2|C1)P(A3|C1)=0.1069*0.000382324*0.120304111=4.9169e-6$

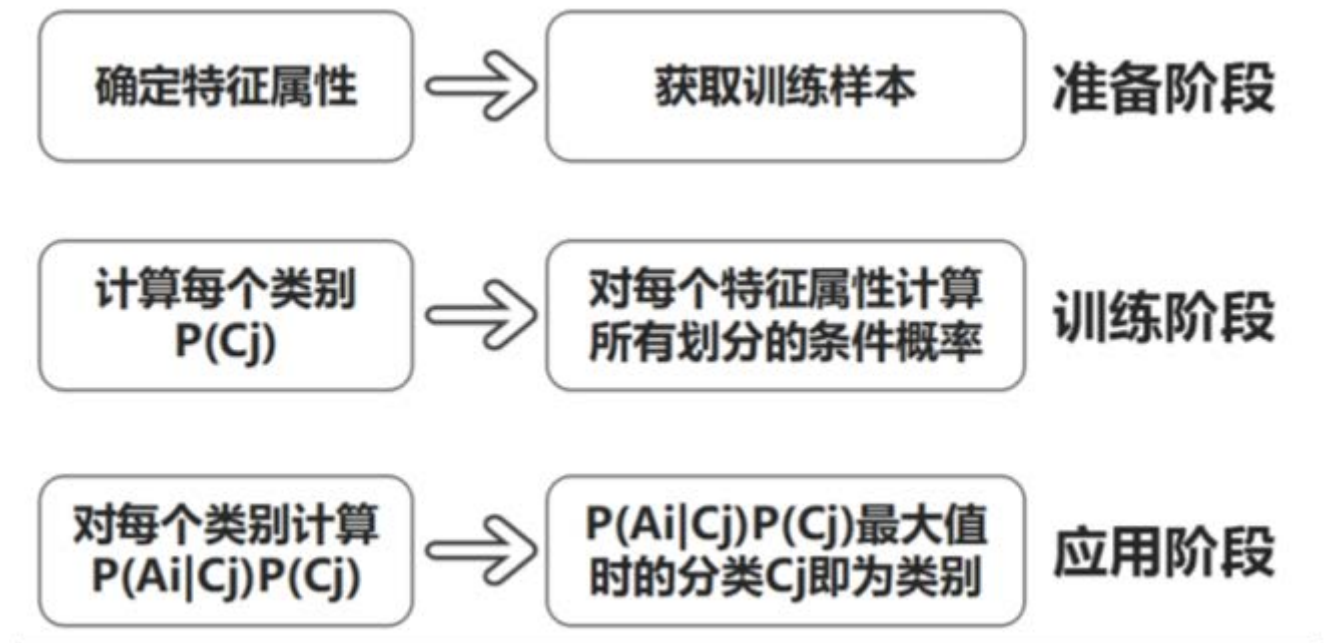
编号	身高（CM）	体重（斤）	鞋码（欧码）	性别
1	183	164	45	男
2	182	170	44	男
3	178	160	43	男
4	175	140	40	男
5	160	88	35	女
6	165	100	37	女
7	163	110	38	女
8	168	120	39	女

$P(A1A2A3|C2)=P(A1|C2)P(A2|C2)P(A3|C2)=0.00000147489*0.015354144*0.120306074=2.7244e-9$

很明显这组数据分类为男的概率大于分类为女的概率

流程

1. 准备阶段，需要确定特征属性，属性值以及 label=> 训练集 2. 训练阶段，输入是特征属性和训练样本，输出是分类器，主要工作是计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率 3. 应用阶段，使用分类器对新数据进行分类



工具应用

Excel 中的 NORMDIST 函数：

$\text{NORMDIST}(x, \text{mean}, \text{standard_dev}, \text{cumulative})$

x: 正态分布中，需要计算的数值；

Mean: 正态分布的平均值；

Standard_dev: 正态分布的标准差；

Cumulative: 取值为逻辑值，即 False 或 True，决定了函数的形式当为 True 时，函数结果为累积分布

当为 False 时，函数结果为概率密度

$\text{NORMDIST}(180, 179.5, 3.697, 0) = 0.1069$

Python 中的概率密度：

Python 中的概率密度：

```
stats.norm.pdf(x, mu, sigma)
```

返回参数为 `mu` 和 `sigma` 的正态分布密度函数在 `x` 处的值

```
from scipy import stats
```

```
mu = 179.5
```

```
sigma = 3.697
```

```
x = 180
```

```
prob = stats.norm.pdf(x, mu, sigma)
```

```
print(prob)
```

sklearn 工具使用：

高斯朴素贝叶斯

特征变量是连续变量，符合高斯分布，比如说人的身高，物体的长度

```
from sklearn.naive_bayes import GaussianNB
```

```
GaussianNB(priors=None)
```

`GaussianNB(priors=None)` #模型创建

`priors`，先验概率大小，如果没有给定，模型则根据样本数据自己计算（利用极大似然法）

查看模型对象的属性：

`class_prior_`:每个样本的概率

`class_count_`:每个类别的样本数量

`theta_`:每个类别中每个特征的均值

`sigma_`:每个类别中每个特征的方差

多项式朴素贝叶斯

特征变量是离散变量，符合多项分布，在文档分类中特征变量体现在一个单词出现的次数，或者是单词的 *TF-IDF* 值等

```
from sklearn.naive_bayes import MultinomialNB  
MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)
```

`MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)`

`alpha`:先验平滑因子，默认等于 1，当等于 1 时表示拉普拉斯平滑

`fit_prior`:是否去学习类的先验概率，默认是 `True`

`class_prior`:各个类别的先验概率，如果没有指定，则模型会根据数据自动学习，每个类别的先验概率相同，即类别数 N 分之一

模型对象的属性：

`class_count_`: 训练样本中各类别对应的样本数

`feature_count_`: 每个类别中各个特征出现的次数

伯努利朴素贝叶斯：

特征变量是布尔变量，符合 0/1 分布，在文档分类中特征是单词是否出现

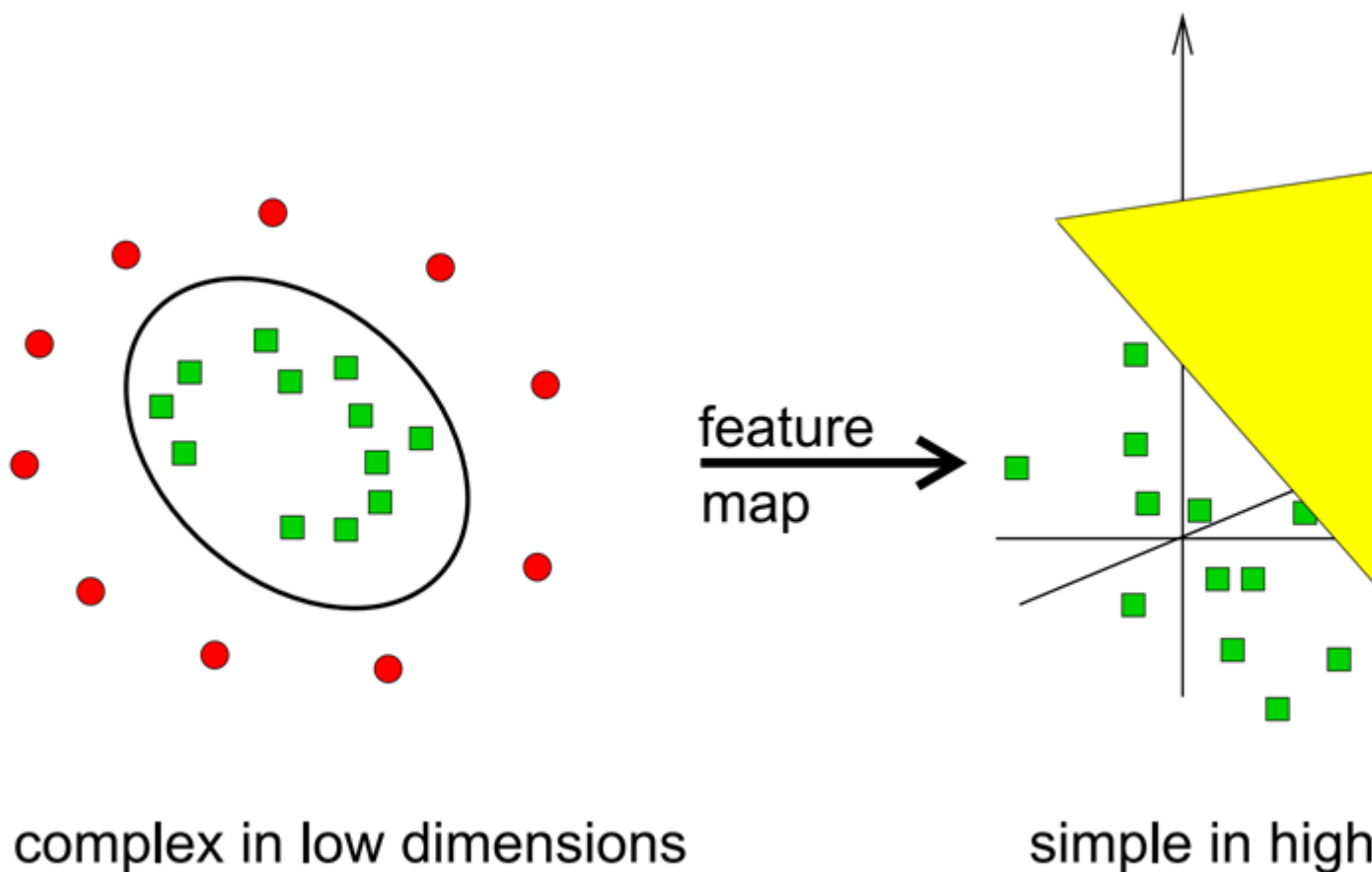
```
from sklearn.naive_bayes import BernoulliNB
```

```
BernoulliNB(alpha=1.0, fit_prior=True, class_prior=None)
```

1.1.4 SVM

一些线性不可分的问题可能是非线性可分的，也就是在高维空间中存在分离超平面

(*separating hyperplane*) 使用非线性函数从原始的特征空间映射至更高维的空间，转化为线性可分问题



sklearn 中支持向量分类主要有三种方法

SVC Support Vector Classification，支持向量机用于分类

```
sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True,  
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1,  
decision_function_shape='ovr', random_state=None)
```

NuSVC Nu-Support Vector Classification, 核支持向量分类, 和 SVC 类似, 不同的是可以使用参数来控制支持向量的个数

```
sklearn.svm.NuSVC(nu=0.5, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True,
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1,
decision_function_shape='ovr', random_state=None)
```

LinearSVC Linear Support Vector Classification 线性支持向量分类, 使用的核函数是 linear

```
sklearn.svm.LinearSVC(penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0,
multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0,
random_state=None, max_iter=1000)
```

常用参数：

常用参数：

- C, 惩罚系数, 类似于LR中的正则化系数, C越大惩罚越大
- nu, 代表训练集训练的误差率的上限 (用于NuSVC)
- kernel, 核函数类型, RBF, Linear, Poly, Sigmoid, precomputed, 默认为RBF径向基核 (高斯核函数)
- gamma, 核函数系数, 默认为auto
- degree, 当指定kernel为'poly'时, 表示选择的多项式的最高次数, 默认为三次多项式
- probability, 是否使用概率估计
- shrinking, 是否进行启发式, SVM只用少量训练样本进行计算
- penalty, 正则化参数, L1和L2两种参数可选, 仅LinearSVC有
- loss, 损失函数, 有'hinge'和'squared_hinge'两种可选, 前者又称L1损失, 后者称为L2损失
- tol: 残差收敛条件, 默认是0.0001, 与LR中的一致

Kernel 核的选择技巧：

样本数量 < 特征数

方法 1：简单的使用线性核就可以，不用选择非线性核 方法 2：可以先对数据进行降维，然后使用非线性核

样本数量 \geq 特征数

可以使用非线性核，将样本映射到更高维度，可以得到比较好的结果

1.1.5 主动学习（选择策略，学习模块）

1.1.6 半监督学习

1.1.7 标签传播算法

1.1.8 表征学习

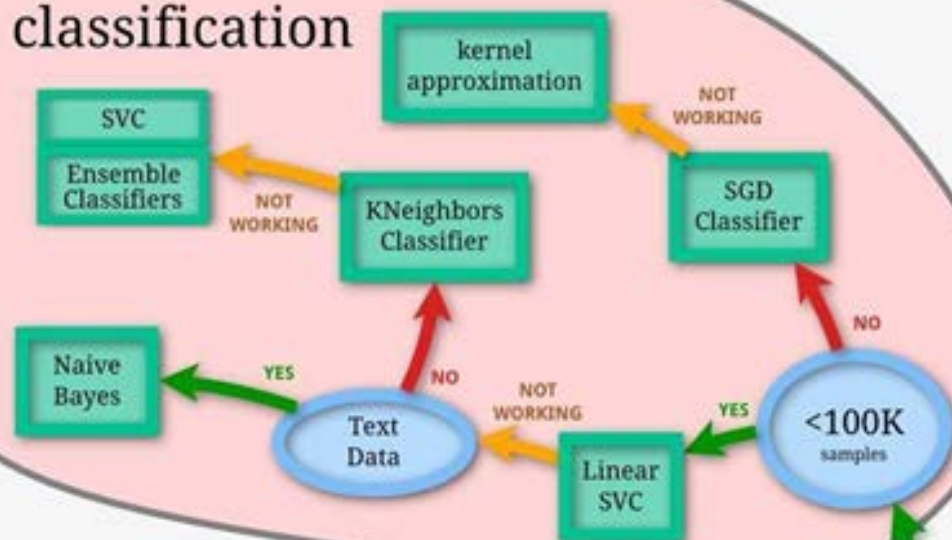
1.1.9 聚类分析

聚类是统计学上的概念，现在属于机器学习中非监督学习的范畴 物以类聚 \Rightarrow 同类相同、异类相异
与分类区别

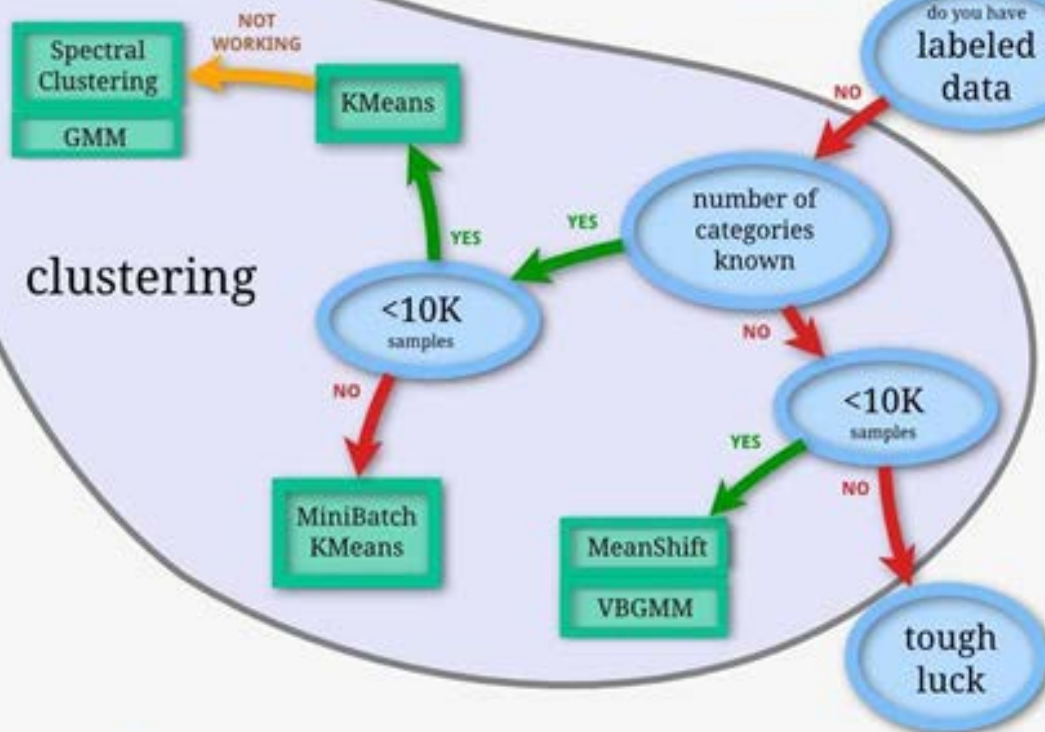
分类是按照已定的模式和标准进行划分，也就是说，在进行分类之前，我们事先已经有了一套划分的标准

聚类，并不知道具体的划分标准，要靠算法进行判断数据之间的相似性，把相似的数据放在一起

classification



clustering



1.1.10 K-Means 原理

K-Means 工作原理

Step1, 选取 **K** 个点作为初始的类中心点，这些点一般都是从数据集中随机抽取的；

Step2, 将每个点分配到最近的类中心点，这样就形成了 **K** 个类，然后重新计算每个类的中心点；

重复 **Step2**，直到类不发生变化，或者你也可以设置最大迭代次数，这样即使类中心点发生变化，但是只要达到最大迭代次数就会结束。

两点之间距离的定义

- 欧氏距离
- 曼哈顿距离
- 切比雪夫距离
- 余弦距离

K-Means 算法的优点

适合挖掘大规模数据集

算法快速、简单

计算复杂度为 $O(n \times k \times t)$ ，其中 n 代表数据集中对象的数量， t 代表着算法迭代的次数， k 代表着簇的数目

聚类是无监督的学习，具体含义需要我们指定。什么时候使用聚类：1. 缺乏足够的先验知识 2. 人工打标签太贵

K-Means 工具

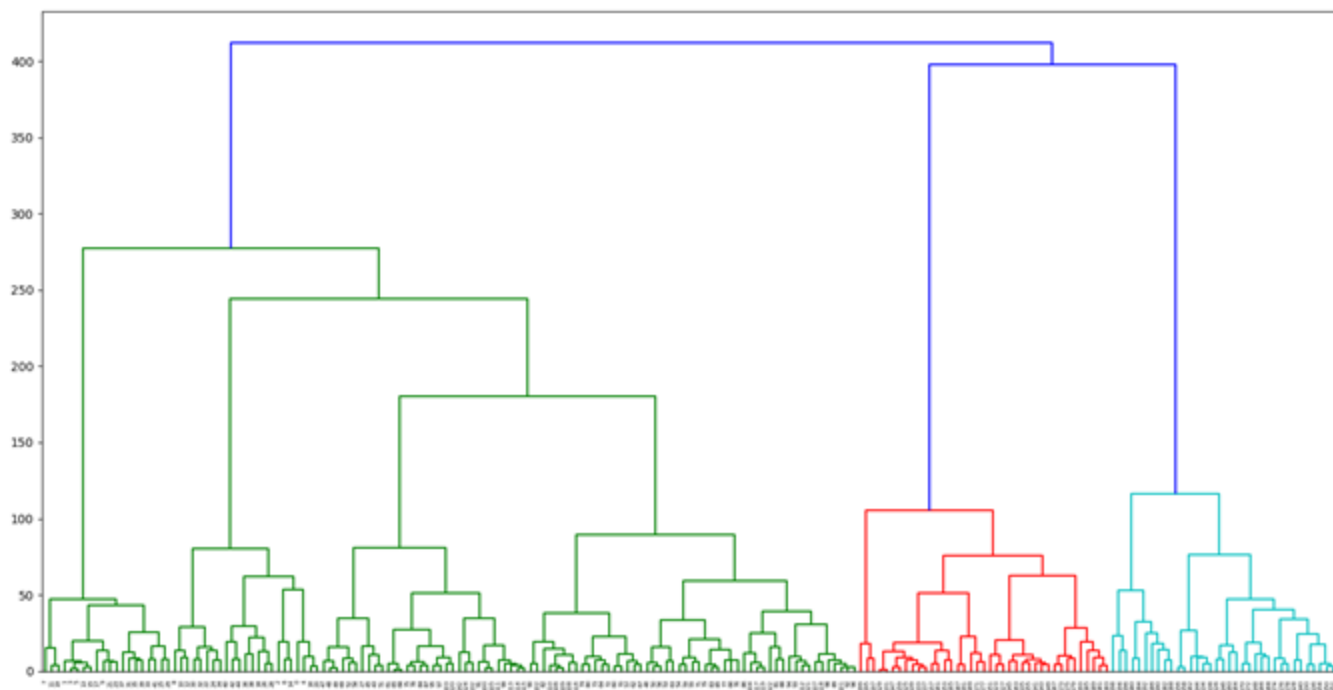
```
from sklearn.cluster import KMeans  
KMeans(n_clusters=8, max_iter=300)
```

n_clusters：聚类个数，缺省值为 8

max_iter：执行一次 k-means 算法所进行的最大迭代数，缺省值为 300

1.1.11 层次聚类

在不同层次对数据集进行划分，从而形成树形的聚类结构（划分可采用“自底向上”的聚合，也可采用“自顶向下”的分拆）



优点，可以层次化聚类，将聚类结构视觉化 不足，计算量大

凝聚聚类 *Agglomerative Clustering*

一种采用自底向上聚类策略的层次聚类算法 **Step1**，将数据集中的每个样本看作一个初始聚类簇 **Step2**，找出距离最近的两个聚类簇进行合并 不断重复上述过程，直到达到我们想要的聚类个数

$$\text{最小距离: } d_{\min}(C_i, C_j) = \min_{x \in C_i, z \in C_j} \text{dist}(x, z)$$

$$\text{最大距离: } d_{\max}(C_i, C_j) = \max_{x \in C_i, z \in C_j} \text{dist}(x, z)$$

$$\text{平均距离: } d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} \text{dist}(x, z)$$

最小距离由两个簇的最近样本决定 => 单链接 (single-linkage)

最大距离由两个簇的最远样本决定 => 全链接 (complete-linkage)

平均距离由两个簇的所有样本共同决定 => 均链接 (average-linkage)

Ward-linkage 算法

可以通过离差平方和 ESS (Error Sum of Squares) 来进行聚类, 最小化聚类前后的离方平方和之差

$$\text{离差平方和: } ESS = \sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2$$

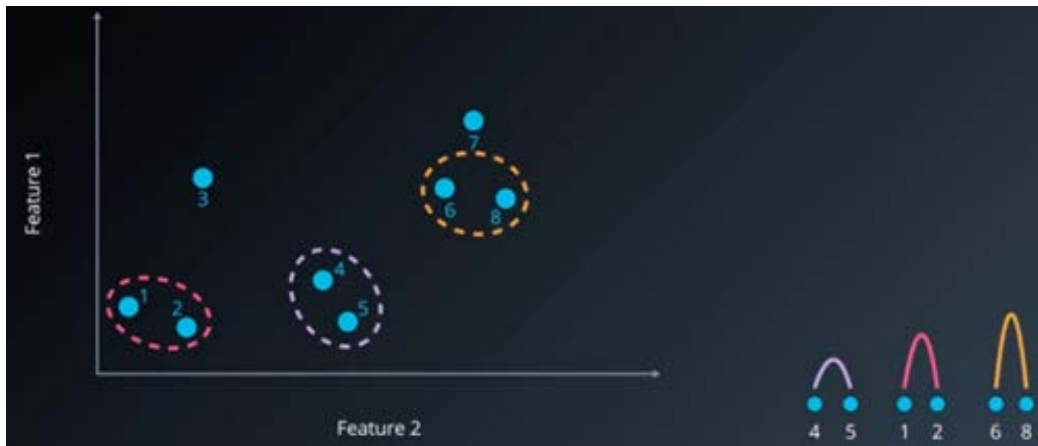
$$\Delta(C_i, C_j) = ESS(C_i \cup C_j) - ESS(C_i) - ESS(C_j)$$

层次图

- 假设我们有以下八个点, 想要使用单链接聚为3个类
- Step1, 将每个点设为一个类, 一共有8个类>3类



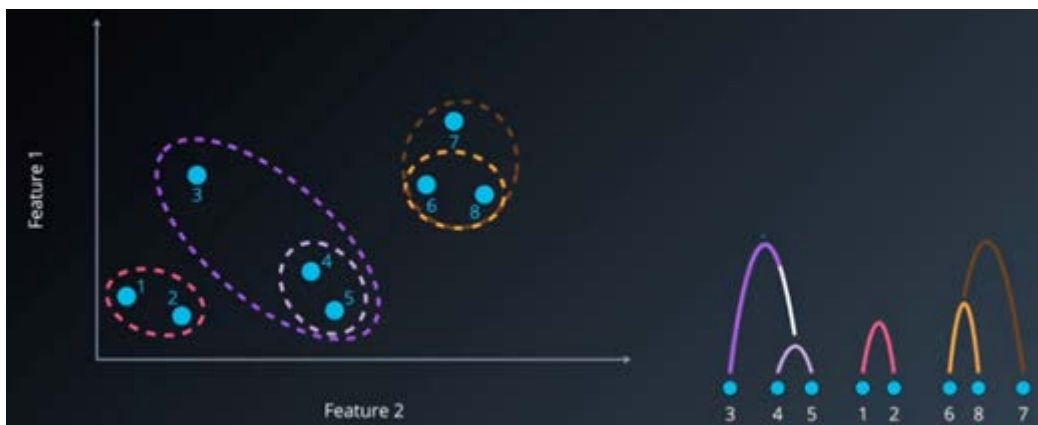
- Step2, 每次将距离最近的两个样本聚为同一簇



-
- 不断找距离最近的合并，直到降到我们要求的聚类数（3类）



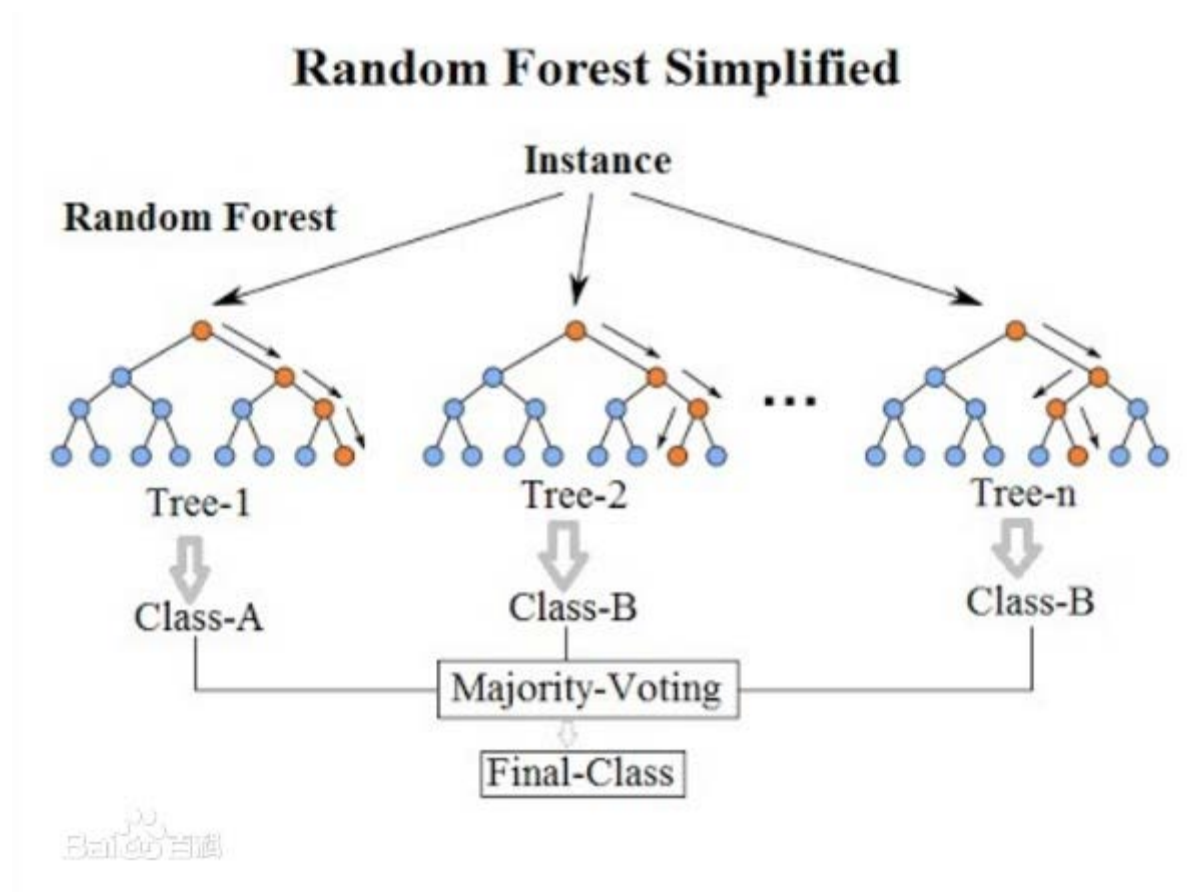
-



-

1.2 Tools

1.2.1 决策树



1.2.2 商超顾客聚类分析

商超顾客聚类分析：

- 数据集：Mall_Customers.csv
- 200名顾客，包括性别、年龄、收入、花费等属性

Thinking：如何进行聚类，给客户打标签

CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
...
194	Female	38	113	91
195	Female	47	120	16
196	Female	35	120	79
197	Female	45	126	28
198	Male	32	126	74
199	Male	32	137	18
200	Male	30	137	83

2 Thinking

2.1 什么是监督学习，无监督学习，半监督学习

能简要说出三种学习方法的含义（10points）

2.1.1 定义

特征 (*feature*) 数据的特征。 举例：书的内容

标签 (*label*) 数据的标签。 举例：书属于的类别，例如“计算机”“图形学”“英文书”“教材”等。

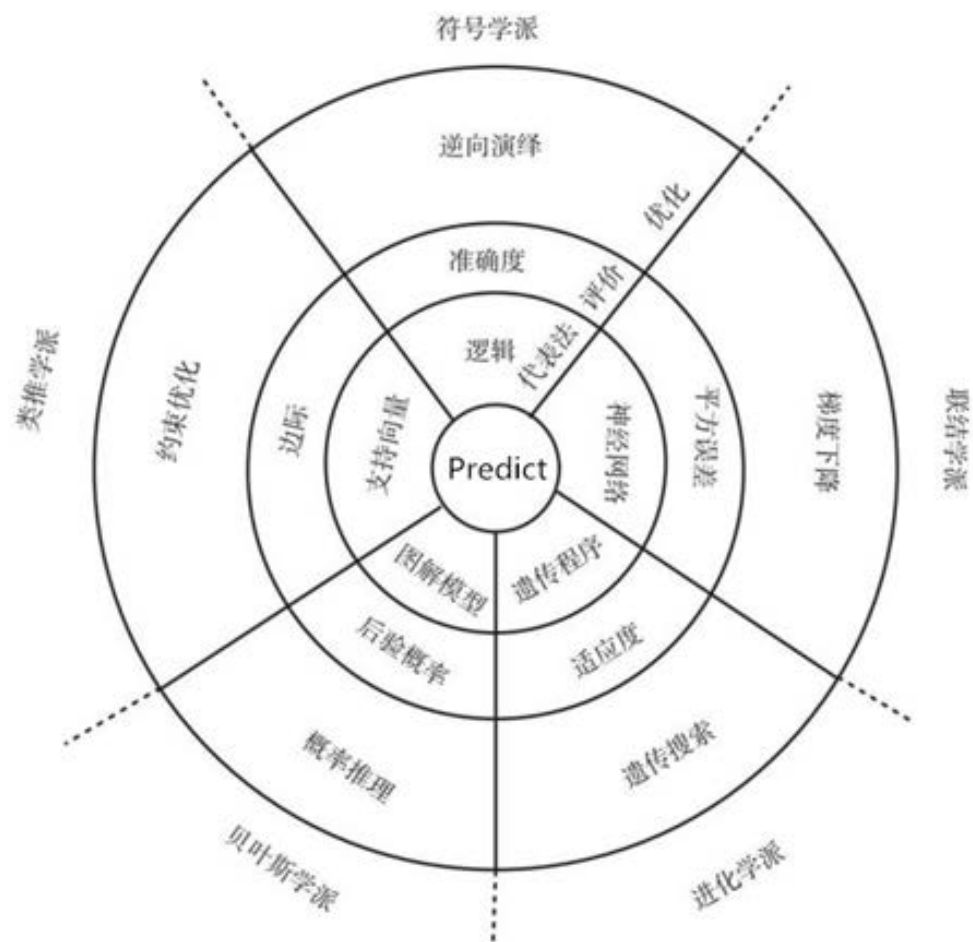
学习 (*learning*) 将很多数据丢给计算机分析，以此来训练该计算机，培养计算机给数据分类的能力。 换句话说，学习指的就是找到特征与标签的映射 (*mapping*) 关系。这样当有特征而无标签的未知数据输入时，我们就可以通过已有的关系得到未知数据标签。 举例：把很多书交给一个学生，培养他给书本分类的能力。

分类 (*classification*) 定性输出称为分类，或者说是离散变量预测。 举例：预测明天是阴、晴还是雨，就是一个分类任务

回归 (*regression*) 定量输出称为回归，或者说是连续变量预测； 举例：预测明天的气温是多少度，这是一个回归任务；

聚类 (*clustering*) 无监督学习的结果。聚类的结果将产生一组集合，集合中的对象与同集合中的对象彼此相似，与其他集合中的对象相异。 举例：没有标准参考的学生给书本分的类别，表示自己认为这些书可能是同一类别的（具体什么类别不知道）。

机器学习派别



决策树

符号学派，认为事情都是有因果的，机器可以自己摸索出规律，

朴素贝叶斯

贝叶斯学派，因果之间不是必然发生，是有一定概率的，即 $P(A|B)$ ，

神经网络，深度学习

联结学派，模仿人脑神经元的工作原理，所有模式识别和记忆建立在神经元的不同连接方式上

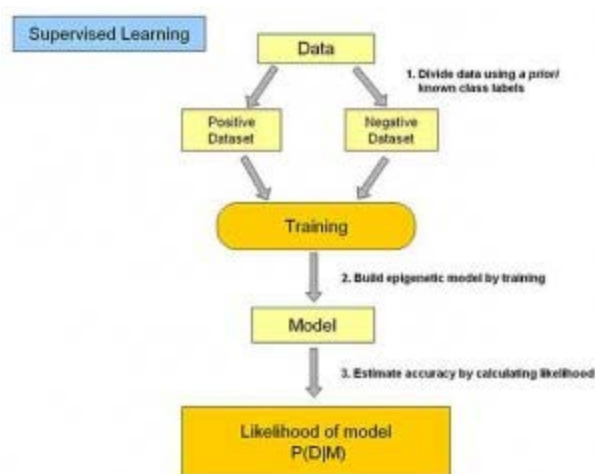
相似度

类推学派，通过类比可以让我们学习到很多未知的知识，所以我们需要先定义“相似度”，通过相似度进行发现

遗传算法

进化学派，上帝通过基因选择来适者生存

2.1.2 监督学习 Supervised learning 给定数据，预测标签。



给定数据，预测标签。是有特征 (*feature*) 和标签 (*label*) 的，即便是没有标签的，机器也是可以通过特征和标签之间的关系，判断出标签。

不仅把训练数据丢给计算机，而且还把分类的结果（数据具有的标签）也一并丢给计算机分析。

计算机进行学习之后，再丢给它新的未知的数据，它也能计算出该数据导致各种结果的概率，给你一个最接近正确的结果。

由于计算机在学习的过程中不仅有训练数据，而且有训练结果（标签），因此训练的效果通常不错。

举例：不仅把书给学生进行训练给书本分类的能力，而且把分类的结果（哪本书属于哪些类别）也给了学生做标准参考。

在监督式学习下，输入数据被称为“训练数据”，每组训练数据有一个明确的标识或结果，如防垃圾邮件系统中“垃圾邮件”“非垃圾邮件”，对手写数字识别中的“1”，“2”，“3”，“4”等。在建立预测模型的时候，监督式学习建立一个学习过程，将预测结果与“训练数据”的实际结果进行比较，不断的调整预测模型，直到模型的预测结果达到一个预期的准确率。监督式学习的常见应用场景如分类问题和回归问题。常见算法有逻辑回归（Logistic Regression）和反向传递神经网络（Back Propagation Neural Network）

常见算法

逻辑回归（Logistic Regression）

反向传递神经网络（Back Propagation Neural Network）

常见应用场景

分类

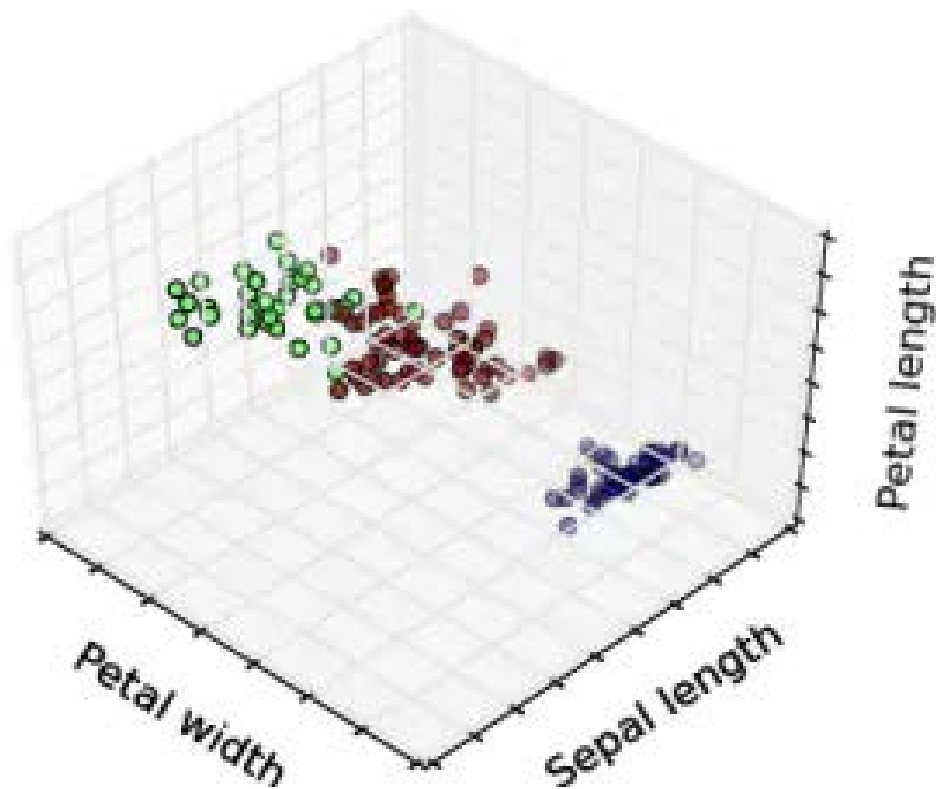
回归

2.1.3 无监督学习 Unsupervised learning 给定数据（没有标签），寻找隐藏的结构。

只有特征，没有标签。给定数据，寻找隐藏的结构。直接对数据集建模。

只给计算机训练数据，不给结果（标签），因此计算机无法准确地知道哪些数据具有哪些标签，只能凭借强大的计算能力分析数据的特征，从而得到一定的成果，通常是得到一些集合，集合内的数据在某些特征上相同或相似。

举例：只给学生进行未分类的书本进行训练，不给标准参考，学生只能自己分析哪些书比较像，根据相同与相似点列出清单，说明哪些书比较可能是同一类别的。



聚类 k-Means

关联规则的学习 Apriori

2.1.4 半监督学习 Semi-Supervised learning

前提：半监督学习介于监督学习与无监督学习之间 通常半监督学习的任务与监督学习一致，即任务中包含有明确的目标（如分类），采用的数据既包括有标签的数据，也包括无标签的数据

使用的数据，一部分是标记过的，而大部分是没有标记的。和监督学习相比较，半监督学习的成本较低，但是又能达到较高的准确度。

综合利用有类标的和没有类标的数据，来生成合适的分类函数。

半监督学习出现的背景：实际问题中，通常只有少量的有标记的数据，因为对数据进行标记的代价有时很高，比如在生物学中，对某种蛋白质的结构分析或者功能鉴定，可能会花上生物学家很多年的工作，而大量的未标记的数据却很容易得到。

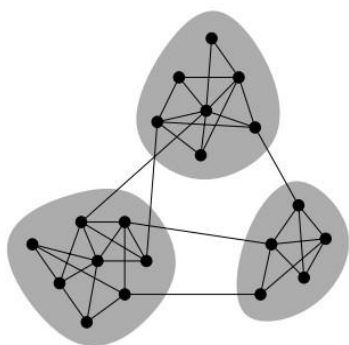
作用：只有少量的数据有 Label，利用没有标签的数据来学习整个数据的潜在分布。

数据的分布必然不是完全随机的，通过一些有标签数据的局部特征，以及更多没标签数据的整体分布，就可以得到可以接受甚至是非常好的分类结果。

举例：给学生很多未分类的书本与少量的清单，清单上说明哪些书属于同一类别。

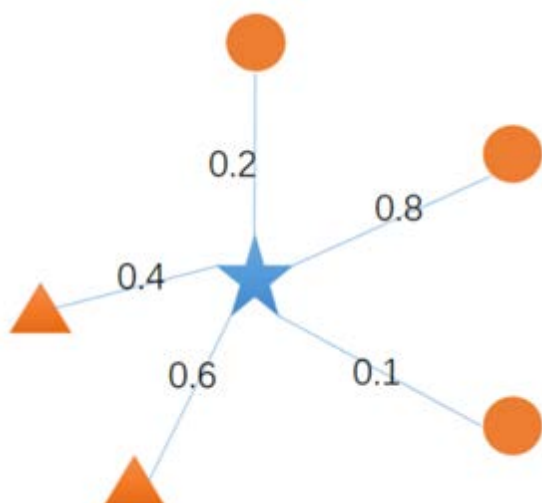
典型算法：标签传播算法 *label propagation*

- LPA: Label Propagation Algorithm
- 基于标签传播的非重叠社区发现算法
- 现实中存在着各种网络：社交网络，交通网络，交易网络，食物链。将这些行为转化为图的网络形式
- 原理是：基于数据分布上的模型假设，利用少量的已标注数据进行指导并预测未标记数据的标记，并合并到标记数据集中去



LPA 算法

- Step1, 每个节点拥有独立的标签
- Step2, 标签传播, 节点向邻居节点传播自己的标签
- Step3, 标签更新, 每个节点的标签更新为邻居节点中出现次数最多的标签。如果存在多个选择, 则随机选择一个
- Step4, 如果节点更新后的标签发生了变化, 则返回到Step2 (激活状态), 否则节点进入非激活状态, 如果所有图中所有节点均为非激活状态, 则标签更新结束。此时具有相同标签的节点属于同一个社区



LPA 算法示意

- 使用加权和更新节点标签

圆形标签权重= $0.2+0.8+0.1=1.1$

三角形标签权重= $0.4+0.6=1$ ，所以节点标签更新为圆形

- 使用加权平均更新节点标签

圆形标签权重= $(0.2+0.8+0.1)/3=0.367$

三角形标签权重= $(0.4+0.6)/2=0.5$ ，节点标签更新为三角形

原理

基于数据分布上的模型假设，利用少量的已标注数据进行指导并预测未标记数据的标记，并合并到标记数据集中去

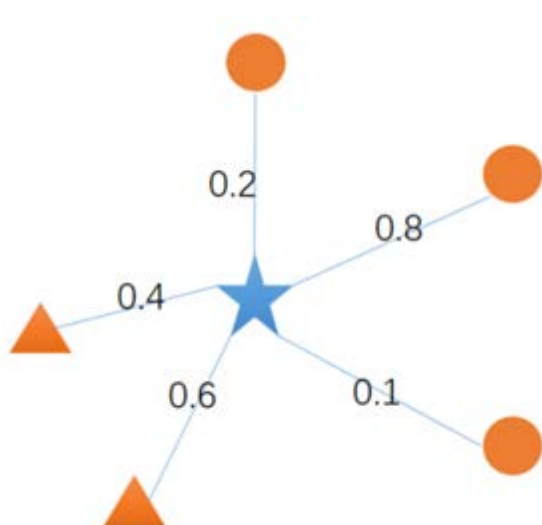
步骤

Step1，每个节点拥有独立的标签 **Step2**，标签传播，节点向邻居节点传播自己的标签

Step3，标签更新，每个节点的标签更新为邻居节点中出现次数最多的标签。如果存在多个选择，则随机选择一个 **Step4**，如果节点更新后的标签发生了变化，则返回到 **Step2**

（激活状态），否则节点进入非激活状态，如果所有图中所有节点均为非激活状态，则标签更新结束。此时具有相同标签的节点属于同一个社区

算法示意



使用加权和更新节点标签 圆形标签权重= $0.2+0.8+0.1=1.1$ 三角形标签权重= $0.4+0.6=1$ ，所以节点标签更新为圆形

使用加权平均更新节点标签 圆形标签权重= $(0.2+0.8+0.1)/3=0.367$ 三角形标签权重= $(0.4+0.6)/2=0.5$ ，节点标签更新为三角形

类别

半监督分类

半监督分类 (Semi-Supervised Classification)：是在无类标签的样例的帮助下训练有类标签的样本，获得比只用有类标签的样本训练得到的分类器性能更优的分类器，弥补有类标签的样本不足的缺陷，其中类标签 取有限离散值。

半监督回归

半监督回归 (Semi-Supervised Regression)：在无输出的输入的帮助下训练有输出的输入，获得比只用有输出的输入训练得到的回归器性能更好的回归器，其中输出取连续值。

半监督聚类

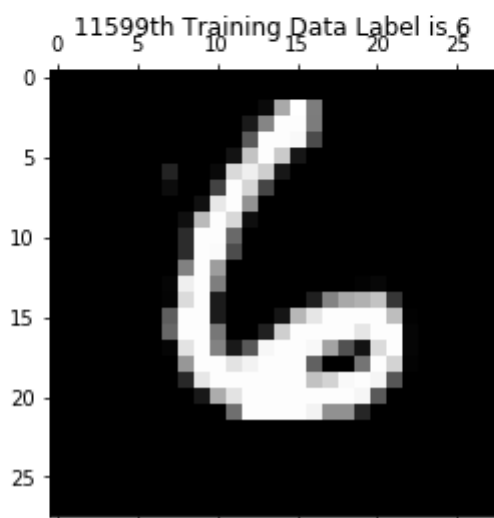
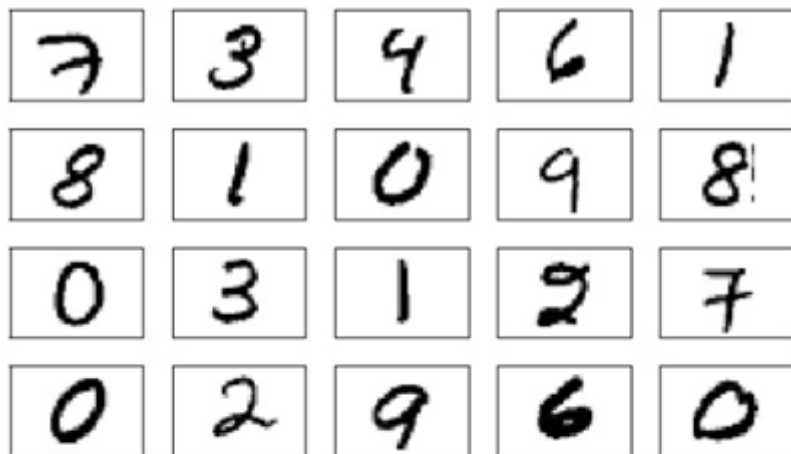
半监督聚类 (Semi-Supervised Clustering)：在有类标签的样本的信息帮助下获得比只用无类标签的样例得到的结果更好的簇，提高聚类方法的精度。

半监督降维

半监督降维 (Semi-Supervised Dimensionality Reduction)：在有类标签的样本的信息帮助下找到高维输入数据的低维结构，同时保持原始高维数据和成对约束 (Pair-Wise Constraints) 的结构不变，即在高维空间中满足正约束 (Must-Link Constraints) 的样例在低维空间中相距很近，在高维空间中满足负约束 (Cannot-Link Constraints) 的样例在低维空间中距离很远。

应用 标签传播：MNIST 手写数字识别

- sklearn自带数据集，原始训练集1797个数据，假设只有10%有标签数据，其余没有标签
- 使用label_propagation模型进行训练，并预测未知标签
- 使用LR模型训练，并预测未知标签，进行对比



主动学习与半监督学习的区别：

对于主动学习：

如果机器可以自己选择学习的样本，它可以使用较少的训练取得更好的效果

需要人工介入，模型主动向 worker 提供数据

对于半监督学习：

指在训练数据十分稀少的情况下，利用没有标签的数据，提高模型效果的方法

在线学习：接收新的样本，更新模型（前提样本可以直接使用）主要在于模型训练的效率

强化学习：接收状态的反馈，并调优模型（前提是反馈可衡量，可捕捉）

2.2 K-means 中的 k 值如何选取

能简要说出 K 值如何选取的方法（10points）

2.2.1 理论

SE(sum of the squared errors, 误差平方和)

手肘法的核心指标是 SSE(sum of the squared errors, 误差平方和),

$$SSE = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2$$

http://blog.csdn.net/qq_15738501

其中， C_i 是第 i 个簇， p 是 C_i 中的样本点， m_i 是 C_i 的质心（ C_i 中所有样本的均值）， SSE 是所有样本的聚类误差，代表了聚类效果的好坏。

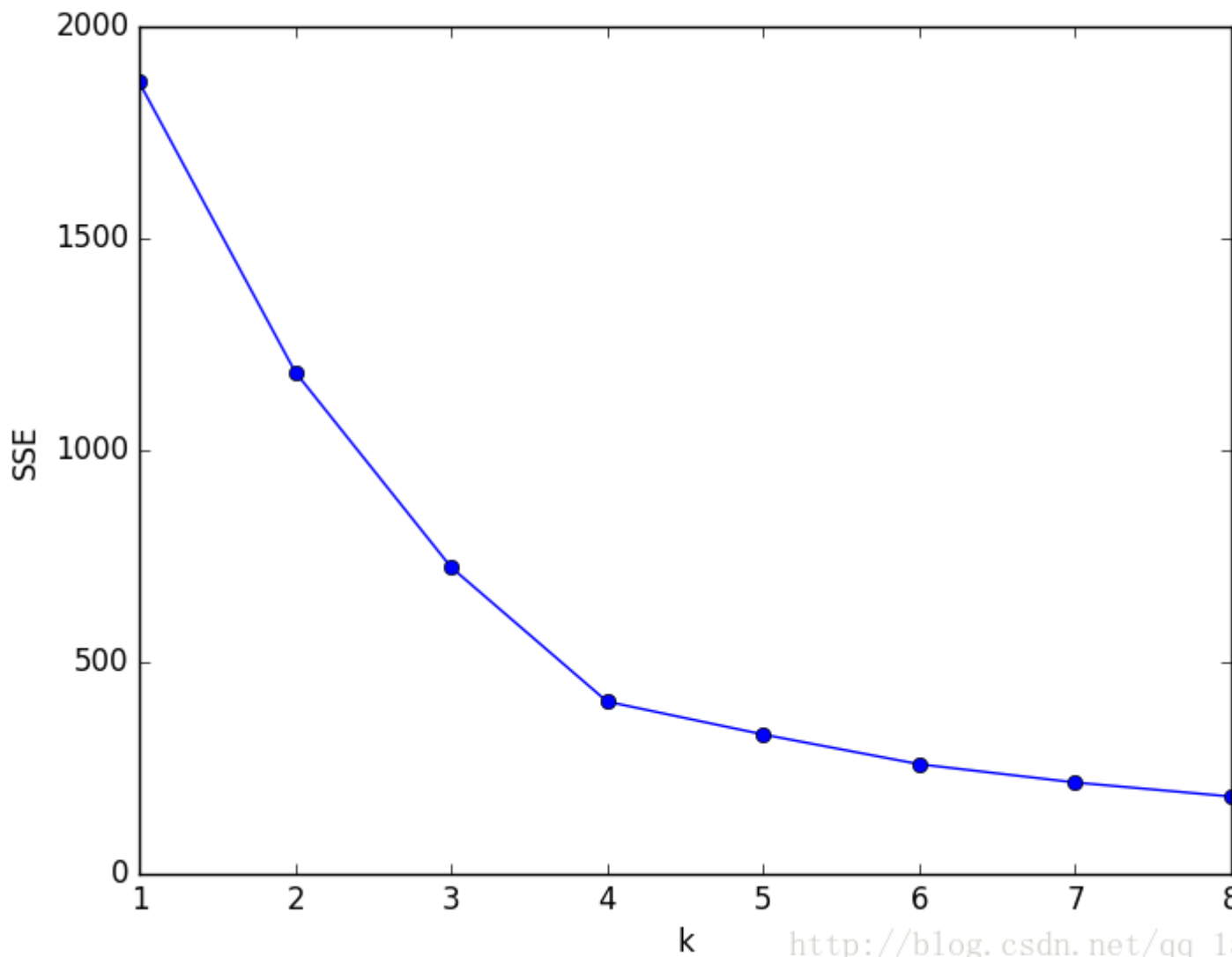
手肘法的核心思想是：随着聚类数 k 的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和 SSE 自然会逐渐变小。并且，当 k 小于真实聚类数时，由于 k 的增大会大幅增加每个簇的聚合程度，故 SSE 的下降幅度会很大，而当 k 到达真实聚类数时，再增加 k 所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，然后随着 k 值的继续增大而趋于平缓，也就是说 SSE 和 k 的关系图是一个手肘的形状，而这个肘部对应的 k 值就是数据的真实聚类数。当然，这也是该方法被称为手肘法的原因。

2.2.2 实践

我们对预处理后数据.csv 中的数据利用手肘法选取最佳聚类数 k 。具体做法是让 k 从 1 开始取值直到取到你认为合适的上限(一般来说这个上限不会太大，这里我们选取上限为 8)，对每一个 k 值进行聚类并且记下对于的 SSE，然后画出 k 和 SSE 的关系图（毫无疑问是手肘形），最后选取肘部对应的 k 作为我们的最佳聚类数。

k 与 SSE 的关系图如下：

显然，肘部对于的 k 值为 4，故对于这个数据集的聚类而言，最佳聚类数应该选 4



2.3 随机森林采用了 bagging 集成学习，bagging 指的是什么

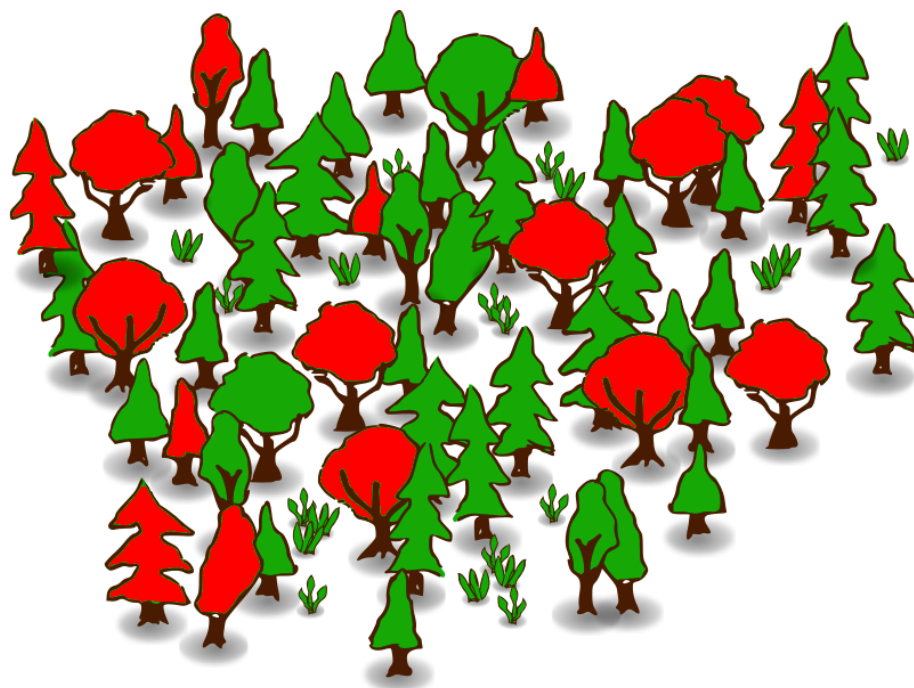
能简要说出自己的观点（10points）

2.3.1 定义：

随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，而它的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法。随机森林的名称中有两个关键词，一个是“随机”，一个就是“森林”。“森林”我们很好理解，一棵叫做树，那么成百上千棵就可以叫做森林了，这样的比喻还是很贴切的，其实这也是随机森林的主要思想--集成思想的体现。“随机”的含义我们会在下边部分讲到。

其实从直观角度来解释，每棵决策树都是一个分类器（假设现在针对的是分类问题），那么对于一个输入样本， N 棵树会有 N 个分类结果。而随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出，这就是一种最简单的 Bagging 思想。

随机森林中有许多的分类树。我们要将一个输入样本进行分类，我们需要将输入样本输入到每棵树中进行分类。打个形象的比喻：森林中召开会议，讨论某个动物到底是老鼠还是松鼠，每棵树都要独立地发表自己对这个问题的看法，也就是每棵树都要投票。该动物到底是老鼠还是松鼠，要依据投票情况来确定，获得票数最多的类别就是森林的分类结果。森林中的每棵树都是独立的，99.9%不相关的树做出的预测结果涵盖所有的情况，这些预测结果将会彼此抵消。少数优秀的树的预测结果将会超脱于芸芸“噪音”，做出一个好的预测。将若干个弱分类器的分类结果进行投票选择，从而组成一个强分类器，这就是随机森林 bagging 的思想（关于 bagging 的一个有必要提及的问题：bagging 的代价是不用单棵决策树来做预测，具体哪个变量起到重要作用变得未知，所以 bagging 改进了预测准确率但损失了解释性。）。下图可以形象地描述这个情况：



有了树我们就可以分类了，但是森林中的每棵树是怎么生成的呢？

每棵树的按照如下规则生成：

1) 如果训练集大小为 N ，对于每棵树而言，随机且有放回地从训练集中的抽取 N 个训练样本（这种采样方式称为 bootstrap sample 方法），作为该树的训练集；

从这里我们可以知道：每棵树的训练集都是不同的，而且里面包含重复的训练样本（理解这点很重要）。

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，这样的话完全没有 bagging 的必要；

为什么要有放回地抽样？

我理解的是这样的：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，这种表决应该是“求同”，因此使用完全不同的训练集来训练每棵树这样对最终分类结果是没有帮助的，这样无异于是“盲人摸象”。

2) 如果每个样本的特征维度为 M ，指定一个常数 $m \ll M$ ，随机地从 M 个特征中选取 m 个特征子集，每次树进行分裂时，从这 m 个特征中选择最优的；

3) 每棵树都尽最大程度的生长，并且没有剪枝过程。

一开始我们提到的随机森林中的“随机”就是指的这里的两个随机性。两个随机性的引入对随机森林的分类性能至关重要。由于它们的引入，使得随机森林不容易陷入过拟合，并且具有很好得抗噪能力（比如：对缺省值不敏感）。

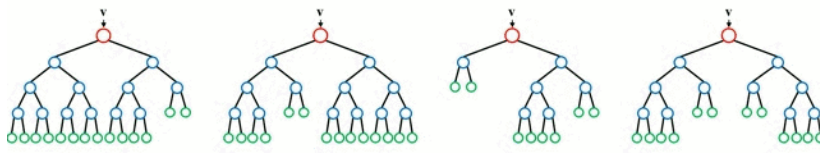
随机森林分类效果（错误率）与两个因素有关：

森林中任意两棵树的相关性：相关性越大，错误率越大；

森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低。

减小特征选择个数 m ，树的相关性和分类能力也会相应的降低；增大 m ，两者也会随之增大。所以关键问题是如何选择最优的 m （或者是范围），这也是随机森林唯一的一个参数。

2.3.2 举例：



随机森林工作原理解释的一个简单例子

描述：根据已有的训练集已经生成了对应的随机森林，随机森林如何利用某一个人的年龄（Age）、性别（Gender）、教育情况（Highest Educational Qualification）、工作领域（Industry）以及住宅地（Residence）共 5 个字段来预测他的收入层次。

收入层次：

Band 1 : Below \$40,000

Band 2: \$40,000 – 150,000

Band 3: More than \$150,000

随机森林中每一棵树都可以看做是一棵 CART（分类回归树），这里假设森林中有 5 棵 CART 树，总特征个数 $N=5$ ，我们取 $m=1$ （这里假设每个 CART 树对应一个不同的特征）。

CART 1 : Variable Age

	Salary Band	1	2	3
Age	Below 18	90%	10%	0%
	19-27	85%	14%	1%
	28-40	70%	23%	7%
	40-55	60%	35%	5%
	More than 55	70%	25%	5%

CART 2 : Variable Gender

	Salary Band	1	2	3
Gender	Male	70%	27%	3%
	Female	75%	24%	1%

CART 3 : Variable Education

	Salary Band	1	2	3
Education	<=High School	85%	10%	5%
	Diploma	80%	14%	6%
	Bachelors	77%	23%	0%
	Master	62%	35%	3%

CART 4 : Variable Residence

	Salary Band	1	2	3
Residence	Metro	70%	20%	10%
	Non-Metro	65%	20%	15%

CART 5 : Variable Industry

	Salary Band	1	2	3
Industry	Finance	65%	30%	5%
	Manufacturing	60%	35%	5%
	Others	75%	20%	5%

我们要预测的某个人的信息如下：

1. Age : 35 years ; 2. Gender : Male ; 3. Highest Educational Qualification : Diploma holder; 4. Industry : Manufacturing; 5. Residence : Metro.

根据这五棵 CART 树的分类结果，我们可以针对这个人的信息建立收入层次的分布情况：

CART	Band	1	2	3
Age	28-40	70%	23%	7%
Gender	Male	70%	27%	3%
Education	Diploma	80%	14%	6%
Industry	Manufacturing	60%	35%	5%
Residence	Metro	70%	20%	10%
Final probability		70%	24%	6%

最后，我们得出结论，这个人的收入层次 70%是一等，大约 24%为二等，6%为三等，所以最终认定该人属于一等收入层次（小于\$40,000）。

2.3.3 Python 实现

```
from sklearn.datasets import load_iris

from sklearn.ensemble import RandomForestClassifier

import pandas as pd

import numpy as np

iris = load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)

df['is_train'] = np.random.uniform(0, 1, len(df)) <= .75

df['species'] = pd.Factor(iris.target, iris.target_names)

df.head()
```

```
train, test = df[df['is_train']True], df[df['is_train']False]
```

```
features = df.columns[:4]
```

```
clf = RandomForestClassifier(n_jobs=2)
```

```
y, _ = pd.factorize(train['species'])
```

```
clf.fit(train[features], y)
```

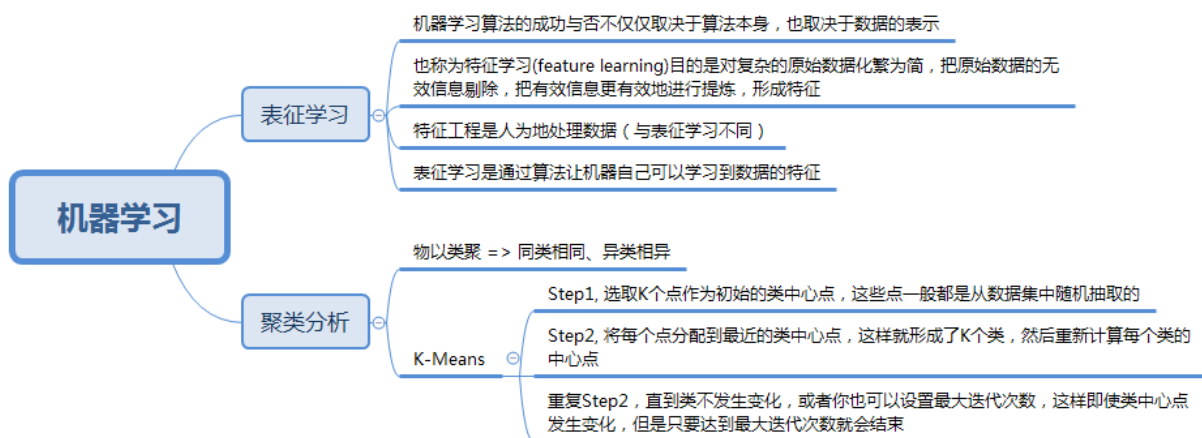
```
preds = iris.target_names[clf.predict(test[features])]
```

```
pd.crosstab(test['species'], preds, rownames=['actual'], colnames=['preds'])
```

source: <https://blog.csdn.net/cpc784221489/article/details/92085702>

2.4 表征学习和半监督学习的区别是什么

能理解两者之间的区别（10points）



2.4.1 表征学习 (Representation Learning)，也称为特征学习(feature learning)

目的是对复杂的原始数据化繁为简，把原始数据的无效信息剔除，把有效信息更有效地进行提炼，形成特征。如果特征被有效提取，那么之后的学习任务会更简单和精确。

机器学习算法的成功与否不仅仅取决于算法本身，也取决于数据的表示

表征学习是通过算法让机器自己可以学习到数据的特征。特征工程是人为地处理数据（与表征学习不同）

例子

比如我们想要寻找 x 和 y 之间的关系：

$$x = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 3 & 2 & 2 & 1 \\ 6 & 4 & 1 & 1 \\ 10 & 4 & 1 & 1 \\ 15 & 3 & 1 & 1 \end{bmatrix} \quad y = \begin{bmatrix} 6 \\ 10 \\ 14 \\ 18 \\ 22 \end{bmatrix}$$

直接计算会比较复杂，这里可以将 x 表示为

$$\text{这样更容易求解出来 } y=x+2 \quad x = \begin{bmatrix} 4 \\ 8 \\ 12 \\ 16 \\ 20 \end{bmatrix}$$

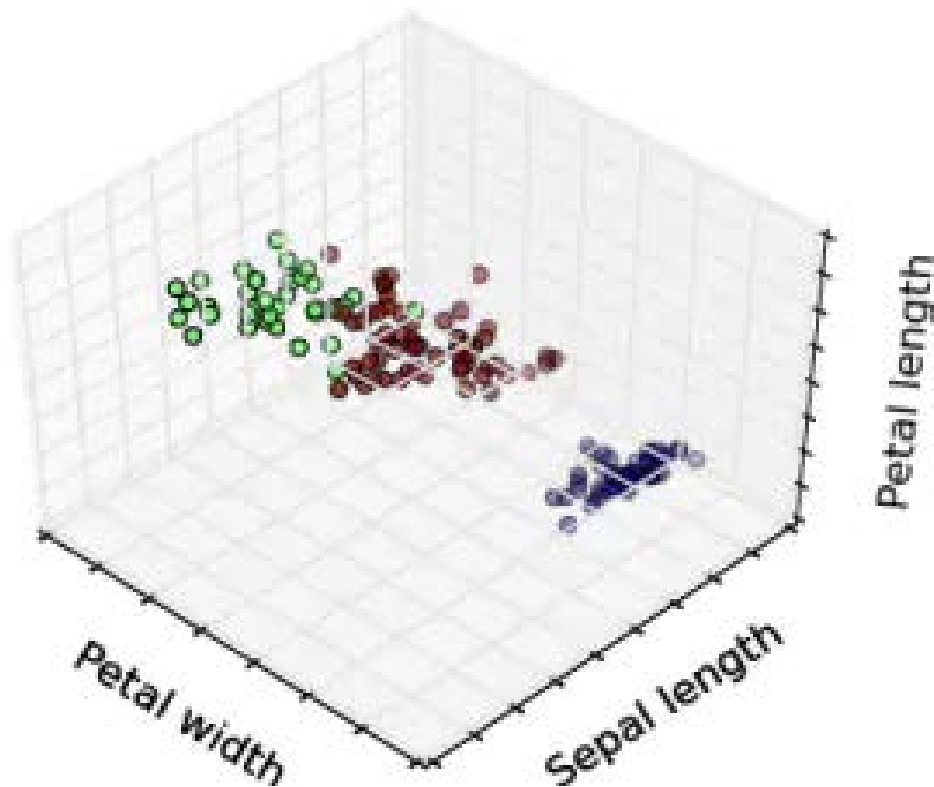
这个过程，我们可以自己来提取特征，也可以采用表征学习

2.4.2 半监督学习

只有特征，没有标签。给定数据，寻找隐藏的结构。直接对数据集建模。

只给计算机训练数据，不给结果（标签），因此计算机无法准确地知道哪些数据具有哪些标签，只能凭借强大的计算能力分析数据的特征，从而得到一定的成果，通常是得到一些集合，集合内的数据在某些特征上相同或相似。

举例：只给学生进行未分类的书本进行训练，不给标准参考，学生只能自己分析哪些书比较像，根据相同与相似点列出清单，说明哪些书比较可能是同一类别的。



聚类 k-Means

关联规则的学习 Apriori

3 Action 文本抄袭自动检测分析:

文本抄袭自动检测分析:

如果你是某新闻单位工作人员（这里假设 source=新华社），为了防止其他媒体抄袭你的文章，你打算做一个抄袭自动检测分析的工具

- 1) 定义可能抄袭的文章来源
- 2) 与原文对比定位抄袭的地方

原始数据: sqlResult.csv, 共计 89611 篇

从数据库导出的文章, 字段包括: id, author, source, content, feature, title, url

常用中文停用词: chinese_stopwords.txt

- 1、完成聚类算法 (10points)
- 2、完成分类算法 (10points)
- 3、完成相似度计算, Top-N 排序 (10points)
- 4、完成代码, 结果正确 (30points)

项目流程:

Step1, 数据加载 加载 sqlResult.csv 及停用词 chinese_stopwords.txt

Step2, 数据预处理

- 1) 数据清洗, 针对 content 字段为空的情况, 进行 dropna
- 2) 分词, 使用 jieba 进行分词
- 3) 将处理好的分词保存到 corpus.pkl, 方便下次调用

4) 数据集切分, 70%训练集, 30%测试集

Step3, 提取文本特征 TF-IDF

Step4, 预测文章风格是否和自己一致 使用分类模型 (比如 MultinomialNB), 对于文本的特征 (比如 TF-IDF) 和 label (是否为新华社) 进行训练

Step5, 找到可能 Copy 的文章, 即预测 label=1, 但实际 label=0

Step6, 根据模型预测的结果来对全量文本进行比对, 如果数量很大, 我们可以先用 k-means 进行聚类降维, 比如 k=25 种聚类 Step7, 找到一篇可能的 Copy 文章, 从相同 label 中, 找到对应新华社的文章, 并按照 TF-IDF 相似度矩阵, 从大到小排序, 取 Top10

Step8, 使用编辑距离 editdistance, 计算两篇文章的距离

Step9, 精细比对, 对于疑似文章与原文进行逐句比对, 即计算每个句子的编辑距离 editdistance

3.1 Jieba

See document(s): [11542002.html](#)

Python 中文 文本分析 实战: jieba 分词+自定义词典补充+停用词词库补充+词频统计

<https://zhuanlan.zhihu.com/p/46922291>

使用方法

环境：Python3.6

安装结巴：pip install jiaba

下载停用词词典哈工大停用词词典

构建补充词典 userdict, 后文详解

运行文章最后面的完整代码

先来认识 jieba

参考

简介

"结巴"中文分词：做最好的 Python 中文分词组件

支持三种分词模式：

精确模式，试图将句子最精确地切开，适合文本分析；

全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；

搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

支持繁体分词

支持自定义词典

Algorithm

基于 Trie 树结构实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG）

采用了动态规划查找最大概率路径，找出基于词频的最大切分组合

对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法

功能 1：分词

jieba.cut 方法接受两个输入参数: 1) 第一个参数为需要分词的字符串 2) cut_all 参数用来控制是否采用全模式

jieba.cut_for_search 方法接受一个参数：需要分词的字符串,该方法适合用于搜索引擎构建倒排索引的分词，粒度比较细 注意：待分词的字符串可以是 gbk 字符串、utf-8 字符串或者 unicode

jieba.cut 以及 jieba.cut_for_search 返回的结构都是一个可迭代的 generator，可以使用 for 循环来获得分词后得到的每一个词语(unicode)，也可以用 list(jieba.cut(...))转化为 list

demo

```
#encoding=utf-8
```

```
import jieba
```

```
seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
```

```
print "Full Mode:", "/ ".join(seg_list) # 全模式
```

```
seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
```

```
print "Default Mode:", "/" .join(seg_list) # 精确模式
```

```
seg_list = jieba.cut("他来到了网易杭研大厦") # 默认是精确模式
```

```
print ", ".join(seg_list)
```

```
seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造") # 搜索引擎模式
```

```
print ", ".join(seg_list)
```

输出：

【全模式】：我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学

【精确模式】：我/ 来到/ 北京/ 清华大学

【新词识别】：他, 来到, 了, 网易, 杭研, 大厦 （此处，“杭研”并没有在词典中，但是也被Viterbi 算法识别出来了）

【搜索引擎模式】：小明, 硕士, 毕业, 于, 中国, 科学, 学院, 科学院, 中国科学院, 计算, 计算所, 后, 在, 日本, 京都, 大学, 日本京都大学, 深造

功能 2：自定义词典补充

开发者可以指定自己自定义的词典，以便包含 jieba 词库里没有的词。虽然 jieba 有新词识别能力，但是自行添加新词可以保证更高的正确率

用法：jieba.load_userdict(file_name) # file_name 为自定义词典的路径

词典格式和 dict.txt 一样，一个词占一行；每一行分三部分，一部分为词语，另一部分为词频，最后为词性（可省略），用空格隔开

示例

userdict.txt 即补充词库示例

宜信普惠 7

宜信 10

极速模式 20

北京清华大学 5

李小福 2 nr

创新办 3 i

easy_install 3 eng

好用 300

韩玉赏鉴 3 nz

八一双鹿 3 nz

台中

凱特琳 nz

Edu Trust 认证 2000

用法示例：

```
#encoding=utf-8
```

```
from __future__ import print_function, unicode_literals
```

```
import sys
```

```
sys.path.append("../")

import jieba

jieba.load_userdict("userdict.txt")

import jieba.posseg as pseg


jieba.add_word('石墨烯')

jieba.add_word('凱特琳')

jieba.del_word('自定义词')


test_sent = (

    "李小福是创新办主任也是云计算方面的专家; 什么是八一双鹿\n"

    "例如我输入一个带“韩玉赏鉴”的标题，在自定义词库中也增加了此词为 N 类\n"

    "「台中」正確應該不會被切開。mac 上可分出「石墨烯」；此時又可以分出來凱特琳了。"

)

words = jieba.cut(test_sent)

print('/'.join(words))


print("="*40)


result = pseg.cut(test_sent)
```

```
for w in result:
```

```
    print(w.word, "/", w.flag, ", ", end=' ')
```

```
print("\n" + "="*40)
```

```
terms = jieba.cut('easy_install is great')
```

```
print('/'.join(terms))
```

```
terms = jieba.cut('python 的正则表达式是好用的')
```

```
print('/'.join(terms))
```

```
print("="*40)
```

```
# test frequency tune
```

```
testlist = [
```

```
    ('今天天气不错', ('今天', '天气')),
```

```
    ('如果放到 post 中将出错。', ('中', '将')),
```

```
    ('我们中出了一个叛徒', ('中', '出')),
```

```
]
```

```
for sent, seg in testlist:
```

```
    print('/'.join(jieba.cut(sent, HMM=False)))
```

```
    word = ".join(seg)
```

```
    print('%s Before: %s, After: %s' % (word, jieba.get_FREQ(word), jieba.suggest_freq(seg, True)))
```



```
print('/'.join(jieba.cut(sent, HMM=False)))
```

```
print("-"*40)
```

之前：李小福 / 是 / 创新 / 办 / 主任 / 也 / 是 / 云 / 计算 / 方面 / 的 / 专家 /

加载自定义词库后： 李小福 / 是 / 创新办 / 主任 / 也 / 是 / 云计算 / 方面 / 的 / 专家 /

功能 3：停用词词库补充

停用词：停用词是指在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，这些字或词即被称为 Stop Words（停用词）。这些停用词都是人工输入、非自动化生成的，生成后的停用词会形成一个停用词表。但是，并没有一个明确的停用词表能够适用于所有的工具。甚至有一些工具是明确地避免使用停用词来支持短语搜索的。

使用哈工大停用词词库：下载地址

功能 4：词频统计

参考

完整代码：

```
from collections import Counter
```

```
import jieba
```

```
jieba.load_userdict('userdict.txt')
```

```
# 创建停用词 list
```

```
def stopwordslist(filepath):

    stopwords = [line.strip() for line in open(filepath, 'r').readlines()]

    return stopwords

# 对句子进行分词

def seg_sentence(sentence):

    sentence_segged = jieba.cut(sentence.strip())

    stopwords = stopwordslist('G:\\哈工大停用词表.txt') # 这里加载停用词的路径

    outstr = ""

    for word in sentence_segged:

        if word not in stopwords:

            if word != '\t':

                outstr += word

                outstr += " "

    return outstr

inputs = open('hebing_wenben\\zuoxi_wenben.txt', 'r') #加载要处理的文件的路径

outputs = open('output.txt', 'w') #加载处理后的文件路径

for line in inputs:

    line_seg = seg_sentence(line) # 这里的返回值是字符串

    outputs.write(line_seg)
```

```
outputs.close()

inputs.close()

# WordCount

with open('output.txt', 'r') as fr: #读入已经去除停用词的文件

    data = jieba.cut(fr.read())

data = dict(Counter(data))


with open('cipin.txt', 'w') as fw: #读入存储 wordcount 的文件路径

    for k, v in data.items():

        fw.write('%s,%d\n' % (k, v))
```

3.2 strip()

See document(s): [att-string-strip.html](#)