机器学习建模分析

DATA

主讲教师：宋晖

人工智能

- 人工智能（AI，Artificial Intelligence）
  - 研究计算机模拟人的某些思维过程和智能行为（如学习、推理、思考、规划等）
  - 研究领域包括机器人、机器学习、计算机视觉、图像识别、自然语言处理和专家系统等
- 发展历程
  - 最初在1956年Dartmouth学会上提出
  - 经过机器推理、专家系统、神经网络等多个发展阶段
- 现阶段
  - 引领性的战略性技术和新一轮产业变革的核心驱动力
  - 应用领域遍布互联网、汽车、智能家居、机器人

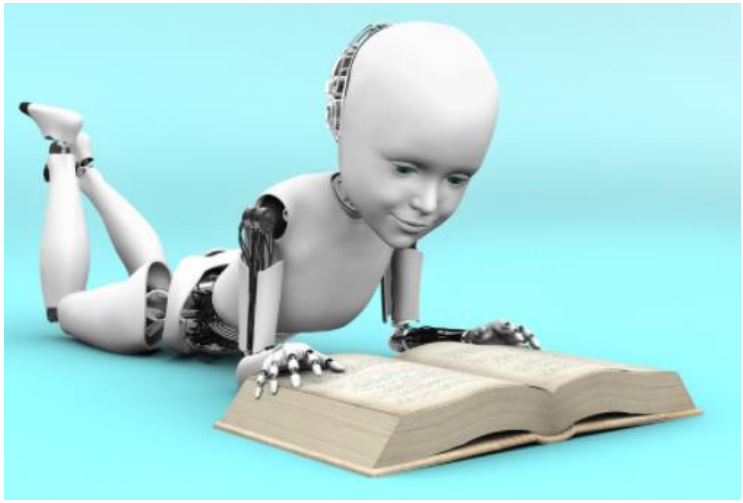
机器学习(Machine Learning)

- 人工智能重要分支
  - 起源于20世纪50年代
  - 美国的阿瑟.萨缪尔研制了一个西洋跳棋程序，发明了“机器学习”

- 研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能
  - 数据挖掘、计算机视觉、自然语言处理、语音识别等人工智能研究领域
  - 生物医药、金融等应用领域

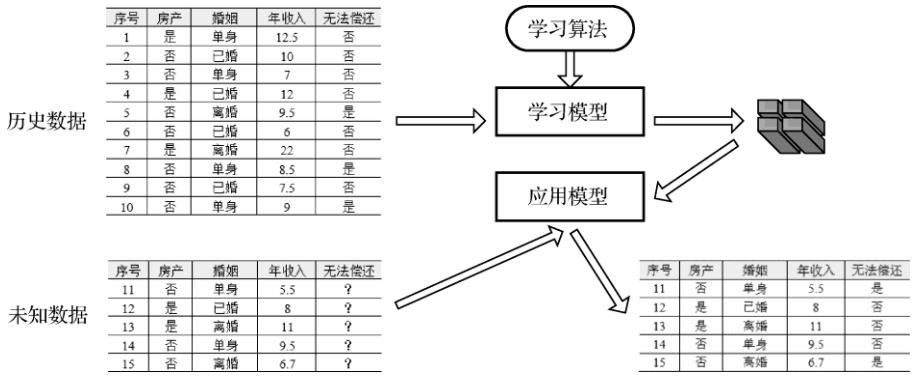
机器学习的任务

- 机器学习方法利用既有的经验，完成某种既定任务，且在此过程不断改善自身性能
- 按照机器学习的任务分为两大类方法
  - 有监督的学习（Supervised Learning）
  - 无监督的学习（Unsupervised Learning）



有监督学习

- 利用经验（数据），学习表示事物的模型，关注利用模型预测未来数据
  - 分类问题（Classification）
    - 对事物所属类型的判别，类别的数量是已知的
    - 例：鸟类型识别，垃圾邮件分类
  - 回归问题（Regression）
    - 预测目标是连续变量
    - 例：根据父母身高预测孩子身高



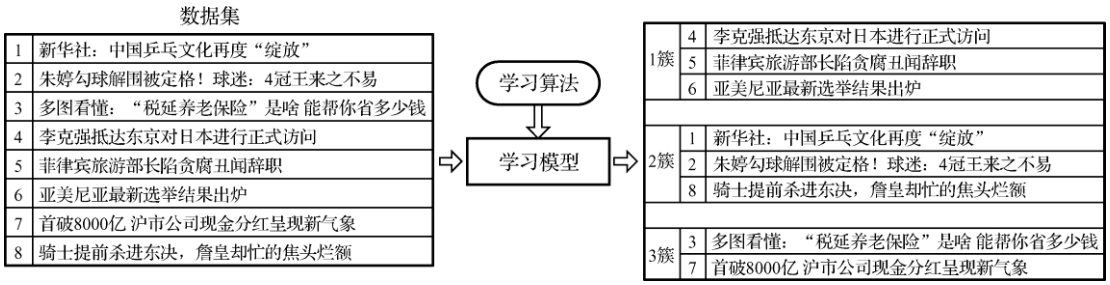
序号	房产	婚姻	年收入	无法偿还
1	是	单身	12.5	否
2	否	已婚	10	否
3	否	单身	7	否
4	是	已婚	12	否
5	否	离婚	9.5	是
6	否	已婚	6	否
7	是	离婚	22	否
8	否	单身	8.5	是
9	否	已婚	7.5	否
10	否	单身	9	是

序号	房产	婚姻	年收入	无法偿还
11	否	单身	5.5	?
12	是	已婚	8	?
13	是	离婚	11	?
14	否	单身	9.5	?
15	否	离婚	6.7	?

序号	房产	婚姻	年收入	无法偿还
11	否	单身	5.5	是
12	是	已婚	8	否
13	是	离婚	11	否
14	否	单身	9.5	否
15	否	离婚	6.7	是

无监督学习

- 倾向于对事物本身特性的分析，常见问题包括
  - 数据降维（Dimensionality Reduction）
    - 对描述事物的特征数量进行压缩的方法
    - 例：从已有的100个特征中选取部分特征表示音乐信号
  - 聚类问题（Clustering）
    - 将事物划分成不同的类别，但事先不知道类别的数量，根据事物之间的相似性，将相似的事物归为一簇
    - 例：电子商务网站将具有类似背景与购买习惯的用户自动聚为一类



Python机器学习方法库

- Scikit-learn是目前使用最广泛的开源方法库
  - 基于NumPy、SciPy、Pandas和Matplotlib开发
  - 封装了大量经典以及最新的机器学习模型
  - 基本功能分类、回归、聚类、数据降维、模型选择和数据预处理
- Scikit-learn本身不支持深度学习与GPU加速
  - 深度学习方法需要使用Tensorflow、Keras、Theano等Python开源框架
- Anaconda已集成了Scikit-learn工具包
  - 导入即可使用

5.2 回归分析

- 一种预测性的建模分析技术
  - 通过样本数据，学习目标变量和自变量之间的因果关系，建立数学表示模型
  - 基于新的自变量，预测相应的目标变量
- 常用方法
  - 线性回归（Linear Regression）
  - 逻辑回归（Logistic Regression）
  - 多项式回归（Polynomial Regression）

实例：工厂产出y受各种投入要素如资本x_1、劳动力x_2、技术x_3等的影响，销售额y受价格x_1和公司对广告费投入x_2的影响。

目标：利用历史数据找出函数表示它们之间的关系，预测未来投资可能带来的产出或收益。

y：**目标变量**

$\{x_1, x_2, \dots, x_d\}$：**自变量**，d：自变量的维度。

案例5-1 广告公司收益预测

- 某公司为了推销产品，在电视、微博、微信等多种渠道投放广告。
  - 目前企业搜集了**200条历史数据**（也称为**样本**）构成数据集
  - 每条数据给出每月3种渠道广告投放费用，以及产品销量

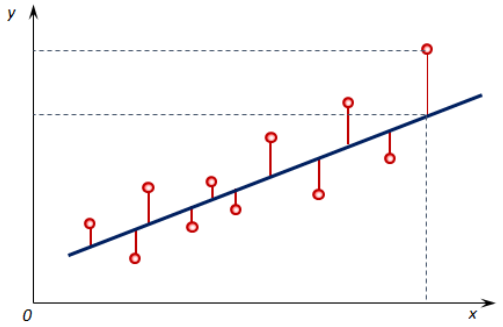
	电视（万元）	微博（万元）	微信（万元）	销量（万个）
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

- 线性回归问题，将销量y表示为电视x_1、微博x_2和微信x_3等渠道广告投入量的线性组合函数：

$$y = f(x),\ f(x) = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d + b$$

回归模型学习

- 有监督学习过程
  - 基于给定的数据集，获得模型参数$\{\omega_1, \omega_2, \ldots, \omega_d, b\}$
  - ω_i：回归系数，b：截距，使得模型在数据集上预测的误差最小
- 求解线性回归模型
  - 统计学的“最小二乘法”，使得线性模型预测所有的训练数据时误差平方和最小。
- 企业给出未来在3种渠道分别投入，即可利用回归模型$f(x)$对销量进行前期预测



回归分析实现

Scikit-learn回归分析模型：LinearRegression类

模型初始化：linreg = LinearRegression()

模型学习：linreg.fit(X, y)

模型预测：y = linreg.predict(X)

参数说明：	
X[m,n]	自变量二维数组，*m* 样本数，*n* 特征项个数，数值型
y[n]	目标变量一维数组，数值型

例5-1：训练回归模型（1）

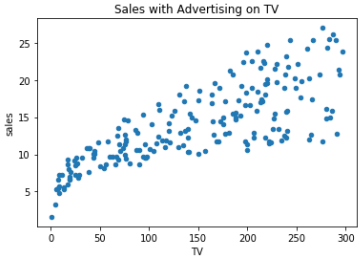
从案例5-1的advertising.csv中读取历史数据，建立广告投入和销量关系的模型，并根据下个月的预计投入预测销量。

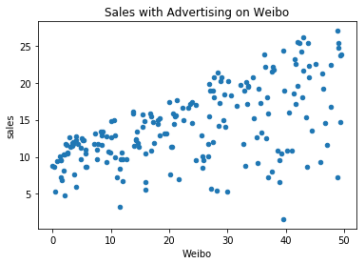
1）读取数据

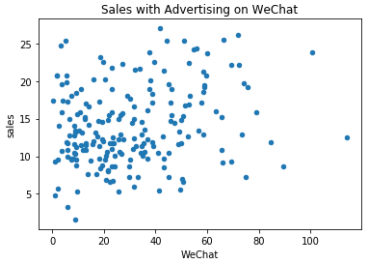
```
filename = 'data\data-5-advertising.csv'
data = pd.read_csv(filename, index_col = 0)
print(data.iloc[0:5, :].values)
```

2）可视化分析

```
#导入绘图库
import matplotlib.pyplot as plt
data.plot(kind='scatter',x='TV',y='Sales',title='
Sales with Advertising on TV')
plt.xlabel("TV")
plt.ylabel("sales")
```







例5-1：训练回归模型（2）

3）建立3个自变量与目标变量的线性回归模型，计算误差。

```
X = data.iloc[:,0:3].values.astype(float)
y = data.iloc[:,3].values.astype(float)
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()  #初始化模型
linreg.fit(X, y)   #输入数据，学习模型
#输出线性回归模型的截距和回归系数
print (linreg.intercept_, linreg.coef_)
```

得到回归方程：

$$y = 0.046x_1 + 0.188x_2 - 0.001x_3 + 2.94$$

4）将回归模型保存到文件，新数据预测时，重新加载使用

```
from sklearn.externals import joblib
joblib.dump(linreg, 'linreg.pkl')   #保存至文件
#重新加载预测数据
import numpy as np
load_linreg = joblib.load('linreg.pkl')  #从文件读取模型
new_X = np.array([[130.1,87.8,69.2]])
print("6月广告投入：",new_X)
print("预期销售：",load_linreg.predict(new_X) ) #使用模型预测
```

6月预期销售：25.34万个

回归模型性能评估

- 采用均方根误差（Root Mean Squared Error，RMSE）来表示误差
  - 回归模型的预测误差越小越好

$$\delta = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

其中n：样本的个数

y_i：样本目标变量的真实值

$\hat{y}_i$：使用回归模型预测的目标变量值

  - 统计学上，使用模型的**决定系数**R^2来衡量模型预测能力

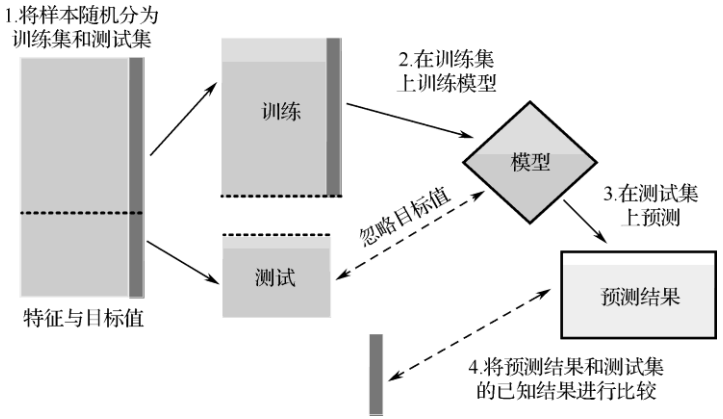
$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2},$$ 其中$\bar{y}_i$表示y_i的均值

R^2的数值范围：0~1，R^2值越大，表示预测效果越好

- 通常训练获得模型用于预测新数据时的性能会降低

训练集与测试集

- 为了更准确地评价模型性能，通常将原始的数据切分为两部分
  - 训练集：学习获得回归模型
  - 测试集：视为未知数据，用于评估模型性能



- Scikit-learn的model_selection类提供数据集的切分方法
- metrics类实现了scikit-learn包中各类机器学习算法的性能评估

数据集分割：X_train, X_test, y_train, y_test =

model_selection.train_test_split(X, y, test_size, random_state)

误差RMSE计算：err = metrics.mean_squared_error(y, y_pred)

决定系数计算：decision_score = linreg.score(X,y)

参数说明：	
test_size	0-1，测试集的比例
random_state	随机数种子，1：每次得到相同样本划分，否则每次划分不一样。

例5-1：模型评估（2）

1）切分为训练集和测试集

```
from sklearn import model_selection
X_train, X_test, y_train, y_test = model_selection.train_test_sp
    lit(X, y, test_size=0.35, random_state=1)
```

2）在训练集上学习回归模型

```
linregTr = LinearRegression()
linregTr.fit(X_train, y_train)
print (linregTr.intercept_, linregTr.coef_)
```

得到回归方程：

$$y = 0.046x_1 + 0.180x_2 + 0.004x_3 + 2.93$$

3）在测试集上评估性能

```
from sklearn import metrics
y_train_pred = linregTr.predict(X_train)
y_test_pred = linregTr.predict(X_test)
train_err = metrics.mean_squared_error(y_train, y_train_pred)
test_err = metrics.mean_squared_error(y_test, y_test_pred)
print( 'The mean squar error of train and test are: {:.2f}, {:.2f}'.format(train_err,
       test_err) )
predict_score =linregTr.score(X_test,y_test)
print('The decision coeficient is: {:.2f} '.format(predict_score) )
```

```
The mean squar error of train and test are: 3.06, 2.32
The decision coefficient is: 0.91
```

思考与练习

1. 延续回归模型的性能评估，计算使用全部数据学习得到的回归模型linreg在测试集上的性能，与只使用训练集的模型linregTr进行比较，并对结论进行分析。
2. 从案例5-1中取出前100条样本，学习回归模型linregHalf；计算模型在练习1的测试集上的预测性能，并与200条样本学习的模型预测性能进行比较。

5.3 分类分析

- 分类学习最常见的监督学习问题
  - 二分类问题：如手机垃圾短信识别
  - 多分类问题：停车场车牌数字识别
- 分类学习采用不同的算法得到不同的分类模型
- 常见算法：
  - 决策树（Decision Tree）
  - 贝叶斯分类
  - KNN（K 近邻）
  - 支持向量机（SVM， Support Vector Machine）
  - 神经网络（Neural Network）
  - 集成学习（Ensemble learning）等

案例5-2：银行客户

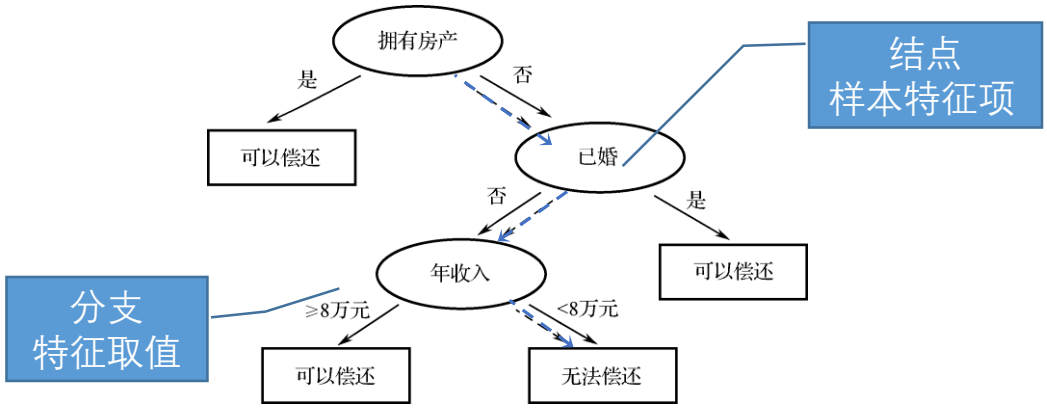
- 某银行拥有客户的基本信息、是否能够偿还债务的历史记录
- 建立模型，预测新客户是否具有偿还债务的能力

序号	拥有房产（是/否）	婚姻状况（单身、已婚、离婚）	年收入（单位：万元）	无法偿还债务（是/否）
1	是	单身	12.5	否
2	否	已婚	10	否
3	否	单身	7	否
4	是	已婚	12	否
5	否	离婚	9.5	是
…	…	…	…	…

- 数据集中每条数据包括多个**特征项**（房产、婚姻和年收入）以及一个**分类标签**（是否无法偿还债务）
- 分类算法通过数据集自动学习获得分类模型（也称为分类器）
- 新客户贷款，依据客户各项特征的值，分类模型预测此客户未来是否具有偿还能力

5.3.2 决策树原理

- 例5-2 判别过程可用决策树来实现



  - 客户的特征：无房产、单身、年收入5.5万元
    - 沿蓝色路径预测：无法偿还

决策树构造算法：ID3、C4.5和CART等

分类模型性能评估

- 使用分类器计算样本分类结果，得到预测类
- 计算每个样本真实类（ground truth）对应的预测类，得到混淆矩阵（confusion matrix）

真实类 \ 预测类	Class=Yes	Class=No
Class=Yes	a	b
Class=No	c	d

- 计算准确率Accuracy

$$Accuracy = \frac{a+d}{a+b+c+d}$$

- 实际问题，关心模型对某一特定类别的预测能力
  - 使用精确率（Precession）、召回率（Recall）和F1-measure

$$Precission = \frac{a}{a+c} \qquad Recall = \frac{a}{a+b} \qquad F1 = \frac{2a}{2a+b+c}$$

决策树分类实现

- Scikit-learn决策树：DecisionTreeClassifier类
  - 支持二分类和多分类问题

模型初始化：clf = tree.DecisionTreeClassifier()

模型学习：clf.fit(X, y)

Accuracy计算：clf.score(X,y)

模型预测：predicted_y = clf.predict(X)

混淆矩阵计算：metrics.confusion_matrix(y, predicted_y)

分类性能报告：metrics.classification_report(y, predicted_y)

参数说明：	
X[m,n]	样本特征二维数组，*m* 样本数，*n* 特征项个数，数值型
y[n]	分类标签的一维数组，必须为整数

例5-3：决策树分类（1）

- 银行贷款偿还数据集共包括15个样本，保存在文本文件bankdebt.csv中
- 每个样本包含3个特征项，1个分类标签，二分类

1）读取数据

```
filename = 'data\bankdebt.csv'
data = pd.read_csv(filename, nrows = 5, index_col = 0, header = None)
print(data)
```

	1	2	3	4
0				
1	Yes	Single	12.5	No
2	No	Married	10.0	No
3	No	Single	7.0	No
4	Yes	Married	12.0	No
5	No	Divorced	9.5	Yes

2）数据预处理，字符类型替换为数字

```
data = pd.read_csv(filename, index_col = 0, header = None)
data.loc[data[1] == 'Yes',1 ] = 1
data.loc[data[1] == 'No',1 ] = 0
data.loc[data[4] == 'Yes',4 ] = 1
data.loc[data[4] == 'No',4 ] = 0
data.loc[data[2] == 'Single',2 ] = 1
data.loc[data[2] == 'Married',2 ] = 2
data.loc[data[2] == 'Divorced',2] = 3
print( data.loc[1:5,:] )
```

	1	2	3	4
0				
1	1	1	12.5	0
2	0	2	10.0	0
3	0	1	7.0	0
4	1	2	12.0	0
5	0	3	9.5	1

例5-3：决策树分类（2）

3）设置X，y，训练分类器

```
X = data.loc[ :, 1:3 ].values.astype(float)
y = data.loc[ :, 4].values.astype(int)
#导入决策树，训练分类器
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)
clf.score(X,y)  #计算分类器的Accuracy
```

输出的准确率结果为1.0

4）评估分类器性能

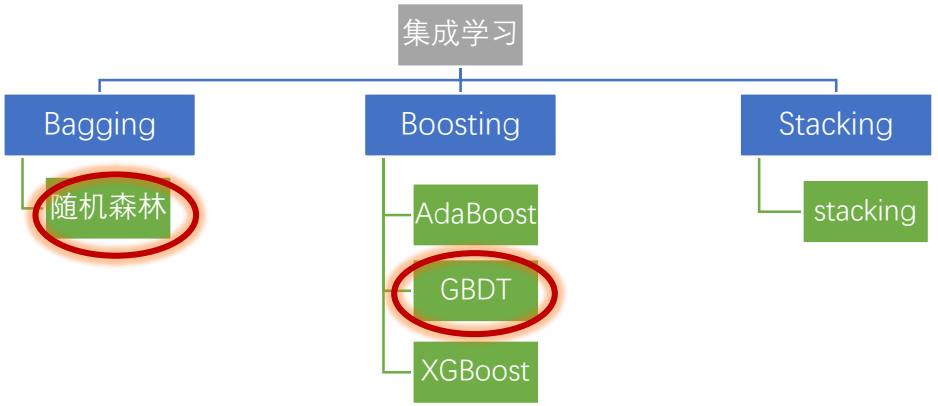
```
predicted_y = clf.predict(X)
from sklearn import metrics
print(metrics.classification_report(y, predicted_y))
print('Confusion matrix:' )
print( metrics.confusion_matrix(y, predicted_y) )
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	5
avg / total	1.00	1.00	1.00	15

```
Confusion matrix:
[[10  0]
 [ 0  5]]
```

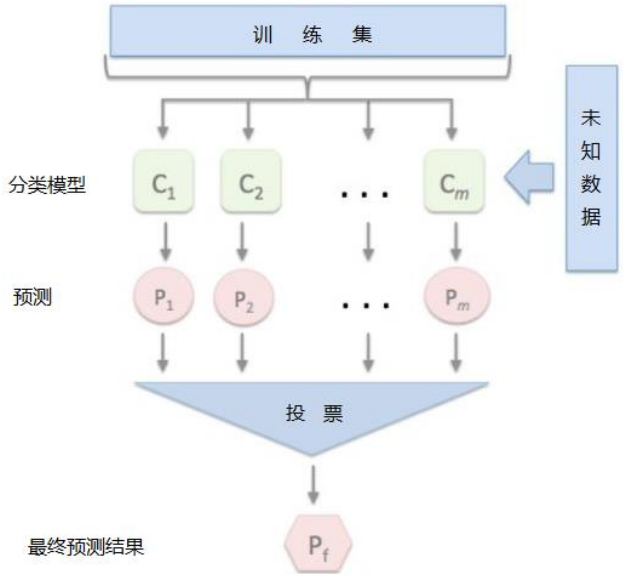
集成学习（Ensemble learning）

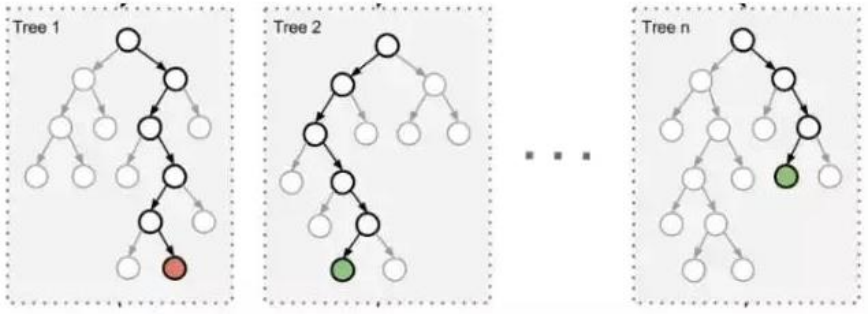
- 背景
  - 有监督学习算法的目标是学习出一个稳定的且预测性能较好的模型
  - 通常学习算法产生的模型有偏好，对某些数据预测效果较好（也称为弱学习器）
- 集成学习思想
  - 构建多个不同的弱学习器
  - 整合弱学习器得到一个更强大的模型（强学习器），来做最后的决策
- 多种集成方法
  - Bagging
  - Boosting
  - Stacking



随机森林（Ensemble learning）

- 通过随机的方式建立一个森林
  - 每棵树都是由从训练集中抽取的部分样本，且基于部分随机选择的特征子集构建

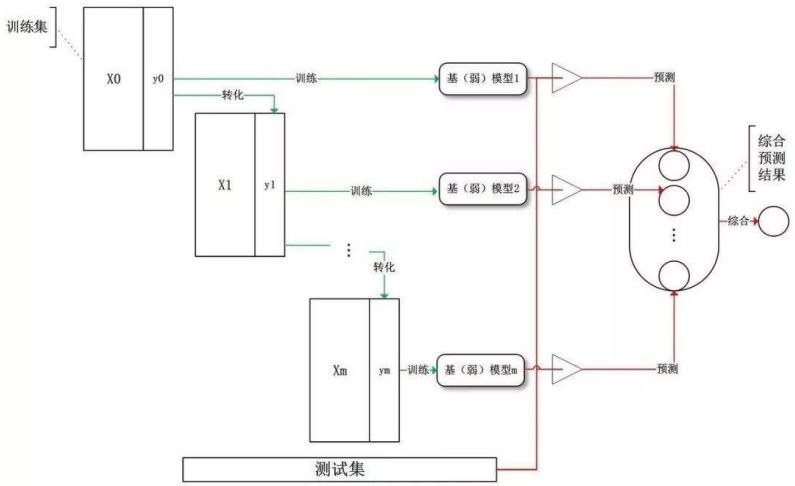




- 预测未知数据时，多个决策树投票决定最终结果

梯度提升机（Gradient Boosting Machine）

- 使用梯度提升方法，通过迭代不断训练新模型
- 新模型专门针对之前模型的弱点进行改进，提升模型的性能



- 目前非深度学习类最好的算法之一，如：XGBoost
- 主要用于处理结构化数据的问题，可用于分类和回归
- 是Kaggle竞赛中最常用的技术之一

集成学习实现

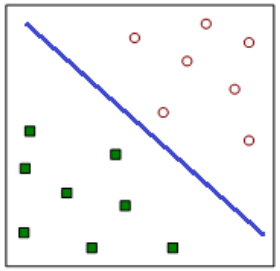
- Scikit-learn： ensemble包

	随机森林	梯度提升机
分类	RandomForestClassifier	GradientBoostingClassifier
回归	RandomForestRegressor	GradientBoostingRegressor

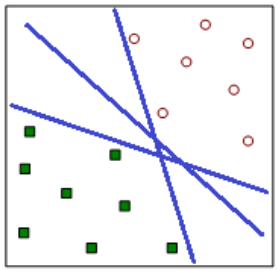
- XGBoost（eXtreme Gradient Boosting ）
  - Sklearn中没有集成
  - 在线安装（pip install xgboost）
  - 下载后，离线安装
    （https://www.lfd.uci.edu/~gohlke/pythonlibs/#xgboost）

5.3.3 支持向量机

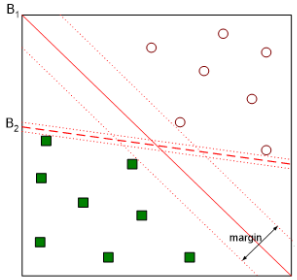
- Support Vector Machine基于数学优化方法的分类学习算法
- 将数据看做多维空间的点，求解一个最优的超平面，将两种不同类别的点分割开来
  - 二维空间为例，超平面就是一条分割线



分割面



多分割面

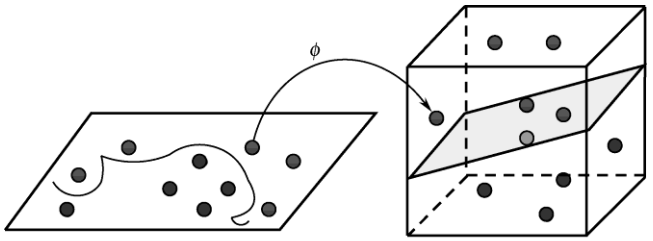


最佳分割面

- 最优分割面，具有最大的Margin
  - 最短距离最大的平面
  - 新样本预测正确的概率更大

SVM分类

- SVM最基本的应用：分类
  - 求解一个最优的分类面，将数据集分割为两个的子集
- 数据集在低维空间中无法使用超平面划分
  - 映射到高维空间，寻找超平面分割



高维映射空间

- SVM采用**核函数（Kernel Function）**将低维数据映射到高维空间
  - 多种核函数，适应不同特性的数据集，影响SVM分类性能的关键因素
  - 常用的核函数
    - 线性核、多项式核、高斯核和sigmoid核等

案例5-3：银行投资业务推广

- 某银行客户数据集中有客户的相关信息
  - 包括年龄、孩子个数、收入等11个特征项，其中
  - “客户是否接受了银行邮件推荐的个人投资计划”（pep）是相应的分类标签（2分类）
- 数据样本共600个，没有缺失数据，保存在bankpep.csv中

id	age	sex	region	income	married	children	car	save_	current	mortgage	pep
ID12101	48	FEMALE	INNER	17546	NO	1	NO	NO	NO	NO	YES
ID12102	40	MALE	TOWN	30085	YES	3	YES	NO	YES	YES	NO
ID12103	51	FEMALE	INNER	16575	YES	0	YES	YES	YES	NO	NO
ID12104	23	FEMALE	TOWN	20375	YES	3	NO	NO	YES	NO	NO

SVM分类实现

- Scikit-learn的SVM：SupportVectorClassification类
  - 只支持**二分类**

模型初始化：clf = svm.SVC(kernel=, gamma, C, …)

参数说明：	
kernel	使用的核函数。Linear：线性核函数、poly：多项式核函数、rbf：高斯核函数、sigmoid：sigmoid核系数
gamma	poly、rbf、或sigmoid的核系数，一般取值在(0,1)之间
C	误差项的惩罚参数，一般取10^n，如1、0.1、0.01 …

  - 模型训练以及性能评估函数与决策树一致

例5-4：SVM分析（1）

- 使用Scikit-learn建立SVM模型预测银行客户是否接受推荐的投资计划
- 评估分类器的性能

1）读取数据

```
filename = 'data\bankpep.csv'
data = pd.read_csv(filename, index_col = 'id')
```

“id” 不具分析意义
读入时作为列索引读入

2）数据预处理，字符类型替换为数字

```
seq = ['married', 'car', 'save_act', 'current_act', 'm
      ortgage', 'pep']
for feature in seq :    # 逐个特征进行替换
    data.loc[ data[feature] == 'YES', feature ] =1
    data.loc[ data[feature] == 'NO', feature ] =0
#替换性别
data.loc[ data['sex'] == 'FEMALE', 'sex'] =1
data.loc[ data['sex'] == 'MALE', 'sex'] =0
print (data[0:5])
```

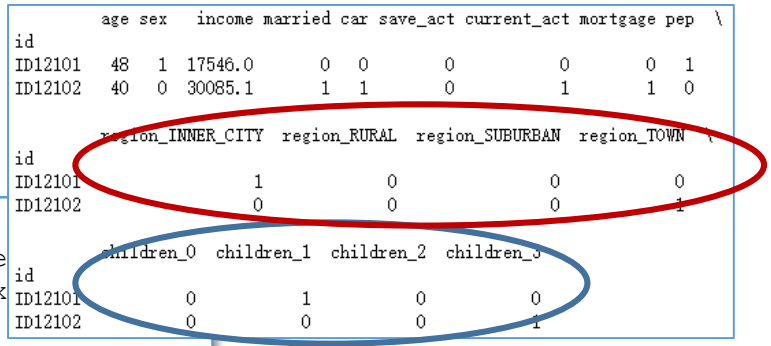
多列数据替换，利用列表值循环

id	age	sex	region	income	married	children	car	save_act	current_act	mortgage	pep
ID12101	48	1	INNER_CITY	17546.0	0	1	0	0	0	0	1
ID12102	40	0	TOWN	30085.1	1	3	1	0	1	1	0
ID12103	51	1	INNER_CITY	16575.4	1	0	1	1	1	0	0
ID12104	23	1	TOWN	20375.4	1	3	0	0	1	0	0
ID12105	57	1	RURAL	50576.3	1	0	0	1	0	0	0

例5-4： SVM分析（2）

3）使用Dummies矩阵处理多个离散值的特征项

当数值不代表数据实际距离时



id	age	sex	income	married	car	save_act	current_act	mortgage	pep
ID12101	48	1	17546.0	0	0	0	0	0	1
ID12102	40	0	30085.1	1	1	0	1	1	0

id	region_INNER_CITY	region_RURAL	region_SUBURBAN	region_TOWN
ID12101	1	0	0	0
ID12102	0	0	0	1

id	children_0	children_1	children_2	children_3
ID12101	0	1	0	0
ID12102	0	0	0	1

```
#将norminal数据转换为dummies矩阵
dumm_reg = pd.get_dummies( data['region'], prefix='re
dumm_child = pd.get_dummies( data['children'], prefix
      )
#删除dataframe中原来的两列后再 join dummies
df1 = data.drop(['region','children'], axis = 1)
df2 = df1.join([dumm_reg,dumm_child], how='outer')
print( df2[0:5] )
```

4）训练模型，测试性能

```
#将df2删除'pep'列后作为X
X = df2.drop(['pep'], axis=1).values.astype(float)
y = df2['pep'].values.astype(int)
#训练模型
from sklearn import svm
clf = svm.SVC(kernel='rbf', gamma=0.6, C = 1.0)
clf.fit(X, y)
print( "Accuracy: ",clf.score(X, y) )
#评价分类器性能
from sklearn import metrics
y_predicted = clf.predict(X)
print( metrics.classification_report(y, y_predicted) )
```

输出的准确率结果为100%

例5-4： SVM分析（3）

5）划分测试集和训练集，在测试集上检验预测性能

```
X_train, X_test, y_train, y_test = model_selection.train_test_split
      (X, y, test_size=0.3, random_state=1)
clf = svm.SVC(kernel='rbf', gamma=0.7, C = 1.0)
clf.fit(X_train, y_train)
print("Performance on training set:", clf.score(X_train, y_train) )
print("Performance on test set:", clf.score(X_test, y_test) )
```

在测试集上正确率只有50~60%

6）SVM样本距离计算，数值型数据需标准化处理

```
from sklearn import preprocessing
X_scale = preprocessing.scale(X)

X_train, X_test, y_train, y_test = model_selection.train_test_sp
      lit(X_scale, y, test_size=0.3, random_state=1)
clf = svm.SVC(kernel='rbf', gamma=0.7, C = 1.0)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

在测试集上正确率提高到69%

7）调整SVM模型参数，提高正确率

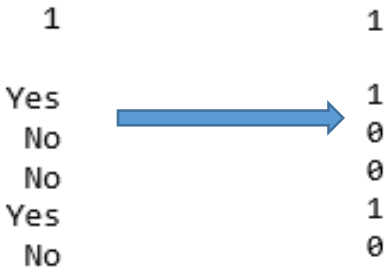
例如: kernel='poly', gamma=0.6, C = 0.001，测试集上正确率提高到80%

离散值数据处理

- 数据集中的非数值型数据
  - 房产：**Yes/No**，婚姻状况：**Married/Single/Divorced**
  - 需要转换为数值型数据处理

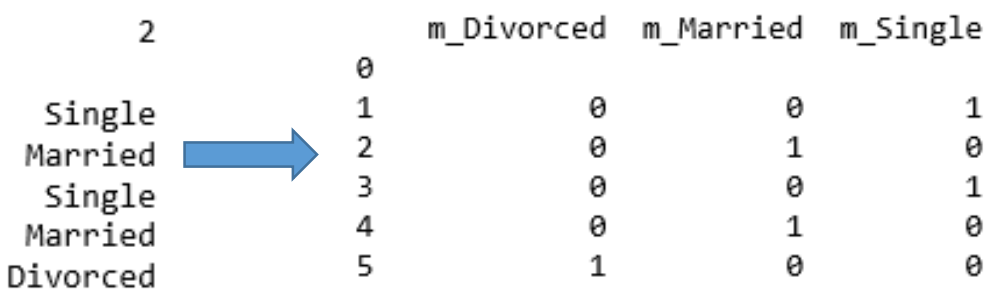
- 替换为指定值
  - ‘Yes’ → 1，‘No’ → 0

```
>>> data.loc[data[1] == 'Yes',1 ] = 1
>>> data.loc[data[1] == 'No',1 ] = 0
```



- 转换为one-hot（独热）矩阵

```
>>> dumm_marital = pd.get_dummies( data[2], prefix='marital' )
```



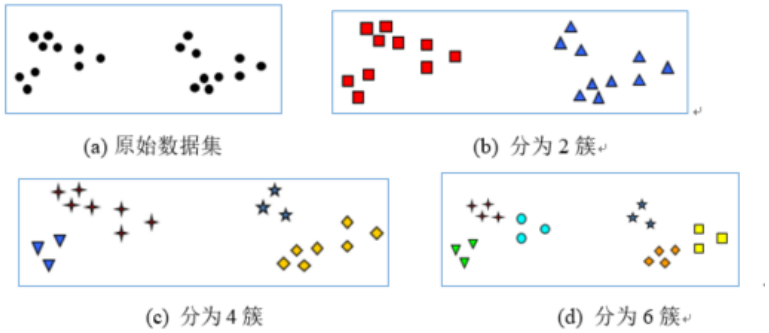
思考与练习

使用bankpep.csv数据集，将数据分为训练集和测试集。

1. 训练决策树分类器，观察在测试集上的分类效果，与SVM分类器的效果进行比较。
2. 训练SVM分类器时，使用‘rbf’核函数，调整参数gamma的值；使用不同的核函数，分别观察在测试集上的分类效果。

5.4 聚类分析

- 聚类是无监督学习方法
  - 根据数据内在性质及规律将其划分为若干个不相交的子集，每个子集称为一个“簇”（Cluster）
  - 自动获得的簇需要人为对应“类别”概念
- 聚类可作为分类等其他任务的预处理过程
  - 如电商网站，用户聚类后，根据簇特性定义用户类，进行商品促销
- 聚类分析目标是使同一个簇中的样本相似度较高，而不同簇间的样本相似度较低
  - 不同算法会得到不同结果



- 聚类方法通常分为几大类：
  - **划分法（Paritition）**
  - 层次法（Hierarchical）
  - 基于密度聚类（Density based）
  - 基于图/网格聚类（Graph/Grid based）
  - 基于模型聚类（Model based）

5.4.1 K-means算法

- 划分法中的经典算法
- 基本目标：将数据聚为若干簇，簇内的点足够近，簇间的点足够远
- K-means首先假定数据集划分的簇数为k，从数据集中任意选择 k 个样本作为各簇的中心
- 聚类过程：
  - 根据样本与簇中心的距离相似度，将数据集中的每个样本划分到与其最相似的一个簇
  - 计算每个簇的中心
  - 不断重复这一过程直到每个簇的中心点不再变化

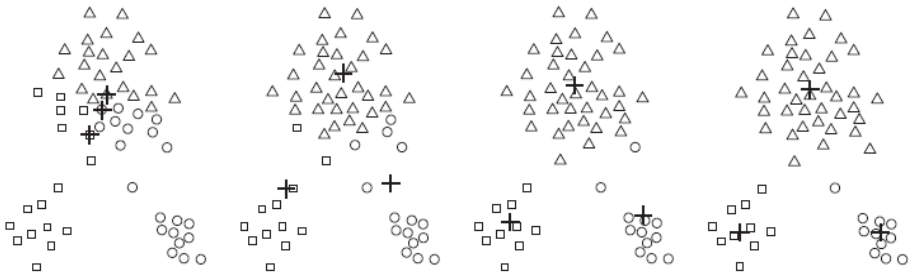


Figure 8.3. Using the K-means algorithm to find three clusters in sample data.

K-means距离计算

- K-means方法算法的核心是相似度的计算
- 数值型数据，欧式距离
  - 数据需要先进行标准化处理
- 假设样本$A = \{a_1, a_2, \dots, a_d\}$，$B = \{b_1, b_2, \dots, b_d\}$，欧式距离计算方法

$$d(A,B) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

- 离散（nominal）数据，余弦相似度
  - 如文本的相似度
  - 余弦值范围：[-1,1]，越接近1，两个样本相似度越高

$$cos(A,B) = \frac{A \cdot B}{|A| \times |B|} = \frac{\sum_{i=1}^{d} a_i \times b_i}{\sum_{i=1}^{d} (a_i)^2 \times \sum_{i=1}^{d} (b_i)^2}$$

案例5-4：鸢尾花数据集

- Iris（鸢尾花）数据集是数据挖掘最著名的数据集
  - 记录了山鸢尾、变色鸢尾和维吉尼亚鸢尾等3个不同种类鸢尾花
  - 包括4个特征项，花萼（sepal）长度与宽度以及花瓣（petal）的长度与宽度，1个分类标签是花的类别
  - 共150条记录

```
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
```

聚类分类实现

- Scikit-learn的聚类：Cluster 类

模型初始化：kmeans = KMeans(n_clusters)

模型学习：kmeans.fit(X)

参数说明：	
n_clusters	簇的个数
X	特征二维数组，数值型

例5-5：K-means聚类（1）

- 使用Scikit-learn的K-means算法对鸢尾花数据集的聚类分析

1）读取数据

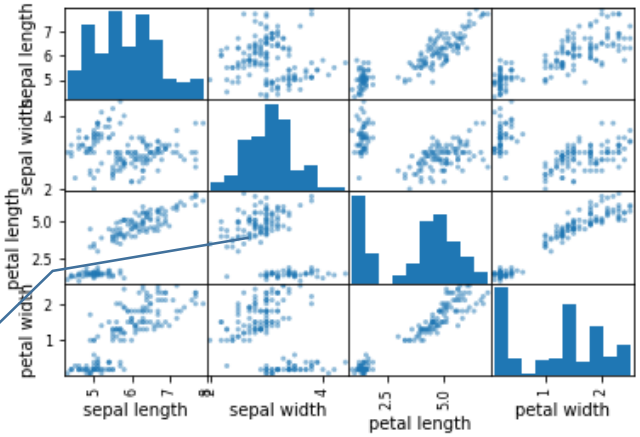
```
filename = 'data\iris.data'
data = pd.read_csv(filename, header = None)
data.columns = ['sepal length','sepal width','petal length','petal width','class']
data.iloc[0:5,:]
```

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2）通过绘制特征散点图矩阵，观察每两种特征的区分度

```
pd.scatter_matrix(data, diagonal='hist')
```

明显聚为2类
标签中的3类中有2类不显著



例5-5： K-means聚类（2）

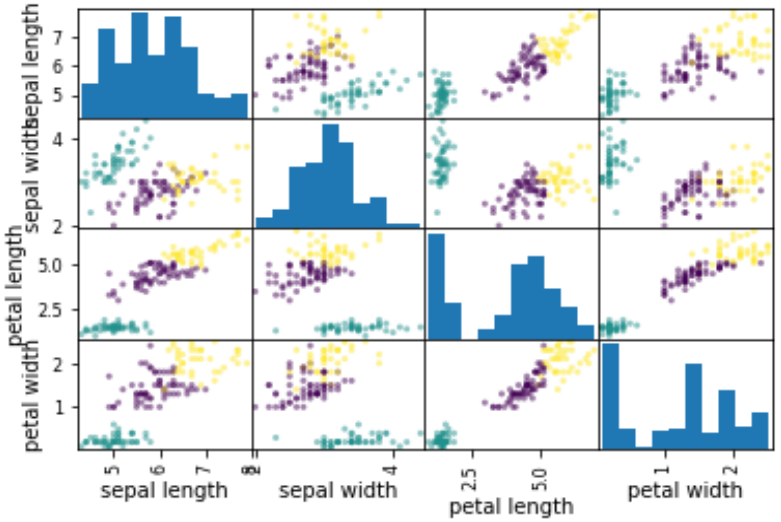
3）定义簇的个数为3，取前4列特征值，训练聚类模型

```
X = data.iloc[:,0:4].values.astype(float)    #准备数据
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)    #模型初始化
kmeans.fit(X)                              #训练模型
```

4）使用样本簇编号作为类型标签，绘制特征对的散点图矩阵

用不同颜色标识不同的簇

```
import matplotlib.pyplot as plt
pd.scatter_matrix(data, c=kmeans.labe
      ls_, diagonal='hist')
```



聚类效果较理想

聚类方法性能评估

- **有分类标签**的数据集
  - 使用**兰德指数**（ARI，Adjusted Rand Index）
  - 计算真实标签与聚类标签两种分布相似性之间的相似性，取值范围为[0,1]
  - 1表示最好的结果，即聚类类别和真实类别的分布完全一致
  - 鸢尾花数据集带有标签

```
from sklearn import metrics
metrics.adjusted_rand_score(y, kmeans.labels_)
```

ARI为0.73

- **没有分类标签**的数据集
  - 使用**轮廓系数**（Silhouette Coefficient）来度量聚类的质量
  - 轮廓系数同时考虑聚类结果的簇内凝聚度和簇间分离度
  - 取值范围：[-1,1]，轮廓系数越大，聚类效果越好

```
from sklearn import metrics
metrics.silhouette_score( X,kmeans.labels_,metric='euclidean' )
```

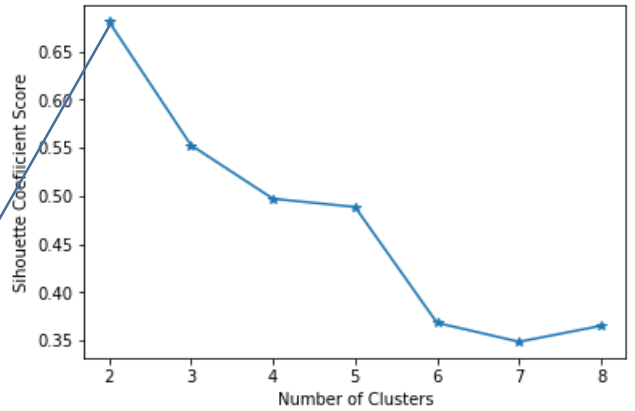
忽略鸢尾花数据集的分类标签，计算聚类的轮廓系数，为0.553

K-means初始k确定

- 肘部原理
  - 尝试多个k值聚类，比较轮廓系数
  - 选择合适的k作为最终模型

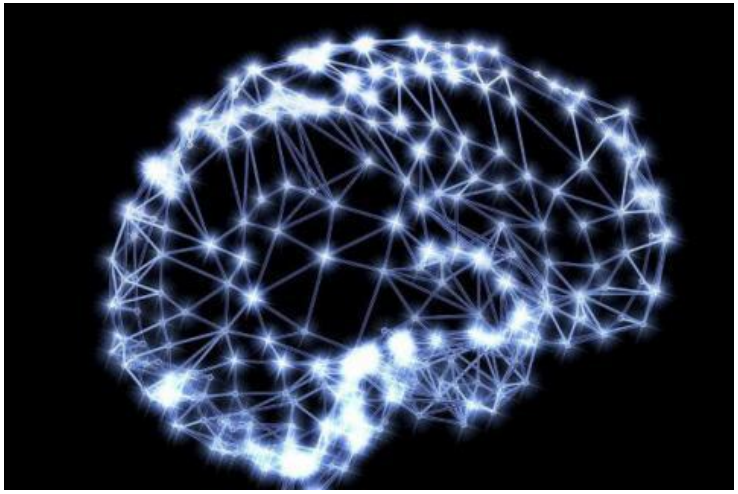
```
clusters = [2,3,4,5,6,7,8]
sc_scores = []
#计算各个簇模型的轮廓系数
for i in clusters:
    kmeans = KMeans( n_clusters = i).fit(X)
    sc = metrics.silhouette_score( X, kmeans.labels_, me
      tric='euclidean' )
    sc_scores.append( sc )
#绘制曲线图反应轮廓系数与簇数的关系
plt.plot(clusters, sc_scores, '*-')
plt.xlabel('Number of Clusters')
plt.ylabel('Sihouette Coefiicient Score')
```

当K=2时聚类的轮廓系数最大
变色鸢尾和维吉尼亚鸢尾之间
差别不是很显著



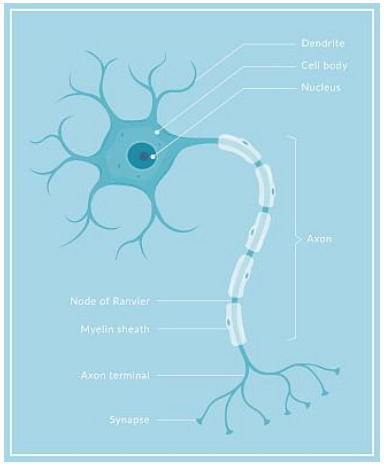
5.5 神经网络和深度学习

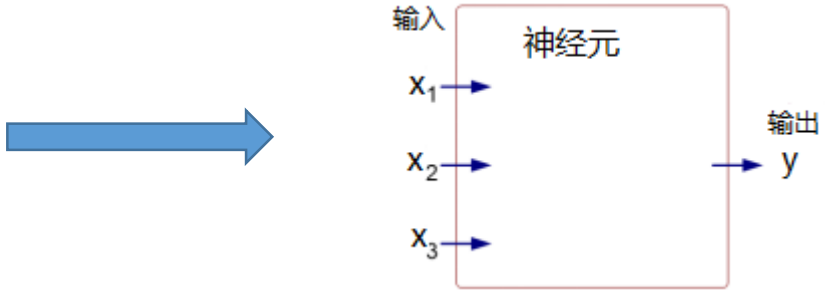
- 神经网络，也称为人工神经网络（Artificial Neural Network, ANN）
  - 20世纪80 年代开始研究，后陷入低潮
  - 随着计算能力增强和大数据出现，深度学习（也就是深度神经网络）技术呈爆发式发展
  - 目前是机器学习以及人工智能领域最重要的方法之一



5.5.1 神经元与感知器

- 神经网络模拟人脑的神经网络来处理问题
  - 人脑思维基础是神经元，神经元相互连接
  - 当某个神经元接受输入，达到某种状态，它就会“兴奋”，向相连神经元发送化学物质
  - “人造神经元”模型，称为**感知器**（percepton）

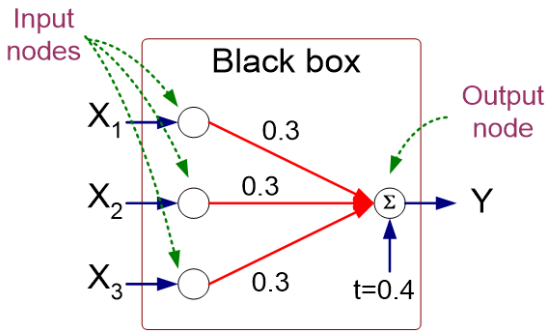




感知器计算模型

- 简化模型，约定输入和输出只有：1 或 0
  - 感知器模型由输入结点、输出结点和权重连接线组成
  - 输出结点将输入结点值乘以权重后加起来，然后和一个阈值t比较，决定输出1或0

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



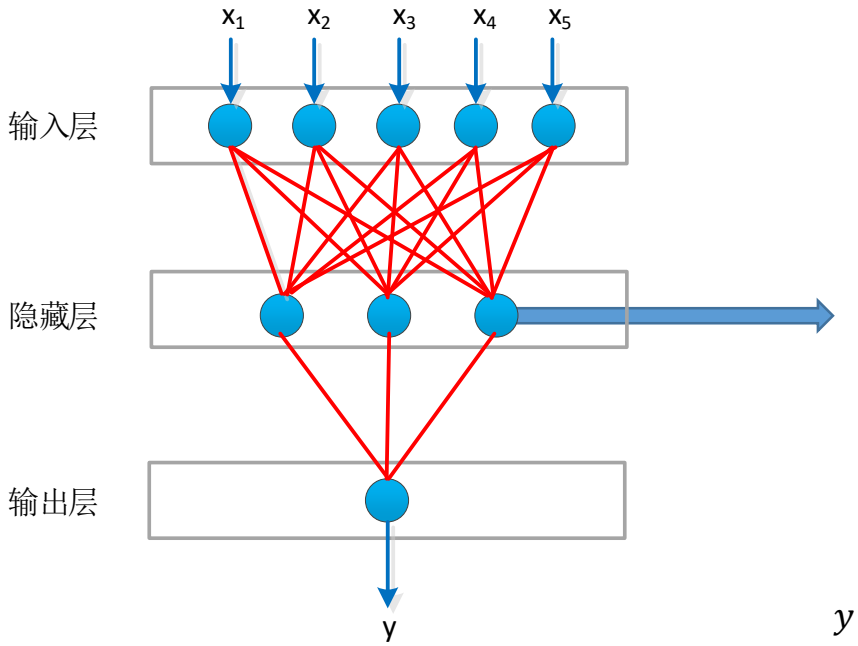
$$y = I(0.3x_1 + 0.3x_2 + \cdots + 0.3x_3 - 0.4)$$

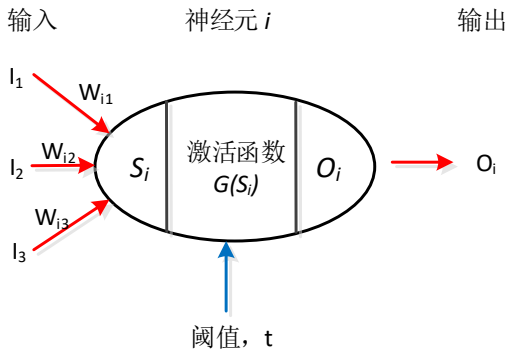
I(z)激活函数 (activation function)

其中：$I(z) = \begin{cases} 1 & if\ z > 0 \\ 0 & z \le 0 \end{cases}$

神经网络模型

- 单个感知器能够处理线性可分问题
- 线性不可分问题，需考虑使用多层神经元
  - 输入层与输出层之间的神经元被称为隐藏层（hidden layer）





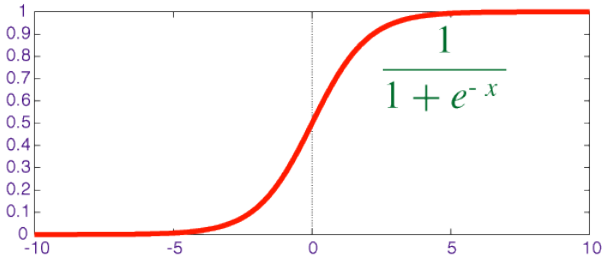
$$y = g(\omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d - t)$$

常用激活函数

$$y = g(\omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d - t)$$

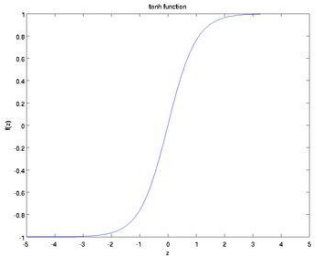
- 激活函数 ***g***
  - Sigmoid、tanh（双曲正切）、ReLU等

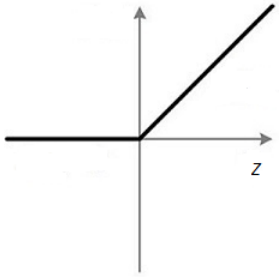
sigmoid $g(z) = \frac{1}{1+e^{-z}}$



ReLU $g(z) = \begin{cases} z & if\ z > 0 \\ 0 & z \leq 0 \end{cases}$

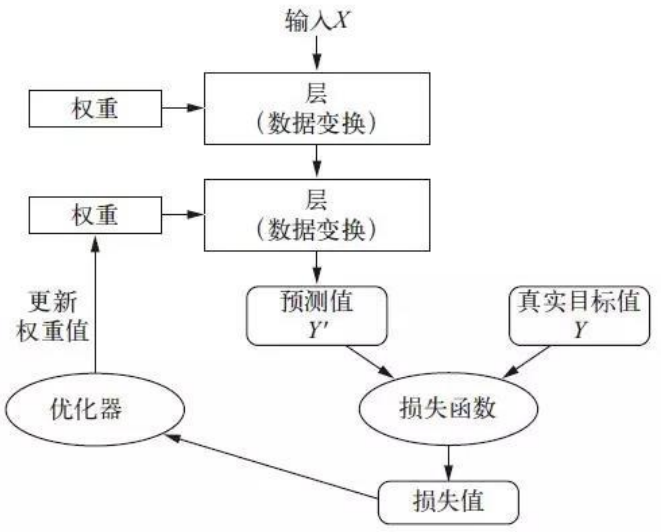
tanh $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$





训练神经网络

- 基于已知结果的数据集，学习神经网络各层的参数
  - 使用迭代方法，不断调整神经元之间的“连接权重”以及每个神经元的阈值（统称为参数），使得最终输出层能够最好地拟合训练集的真实值
  - 学习算法：误差反向传播（BP，error BackPropagation）算法



- 设定损失函数（loss function）/目标函数
- 将损失函数的值作为反馈信号，对权重值进行微调，以降低损失值
- 调节由优化器（optimizer）实现，也就是反向传播算法
- 开始，网络的权重随机赋值，随着训练的过程循环足够多的次数，得到的权重使损失函数最小

神经网络应用

- 分类
  - 二分类问题，输出层只需要1个结点
    - 输出层使用sigmoid激活函数
    - 损失函数，采用bianry_crossentropy
  - 多分类问题，输出层需要多个输出结点，每个结点对应一种类型，输出值表示属于该类型的概率
    - 输出层使用softmax激活函数
    - 损失函数，采用categoricial_croosentropy
- 回归
  - 预测连续值，而不是离散的标签，如:明天的天气温度，地区房屋价格等
  - 神经网络的输出层只有一个结点，不需要激活函数
  - 损失函数，使用线性回归的目标函数，均方误差MSE

神经网络分类

- 神经网络可用于分类
  - 二分类问题，输出层只需要1个结点
  - 多分类问题就需要多个输出结点，每个结点对应一种类型，输出值表示属于该类型的概率
- 神经网络学习
  - 调整神经元之间的“连接权重”以及每个神经元的阈值（参数），使得最终输出层能够最好地拟合训练集的真实值
  - 最强大的学习算法：误差反向传播（BP，error BackPropagation）算法
- Scikit-learn的神经网络实现：MLPClassifier类

模型初始化：mlp = MLPClassifier(solver,activation,hidden_layer_sizes,

alpha,max_iter,random_state,...)

参数说明：	
solver	优化权重的算法：{'lbfgs', 'sgd', 'adam'}, 默认adam
activation	激活函数，取值{'identity', 'logistic', 'tanh', 'relu'}, 默认relu
hidden_layer_sizes	神经网络结构，用元祖表示，其中元祖第n个元素值表示，第n层的神经元个数。例：(5,10,5)，表示3隐层，每层的结点数分别为5、10和5
alpha	正则化惩罚项参数，缺省0.0001
max_iter	最大迭代次数，神经网络参数的BP学习算法的学习次数
random_state	随机数种子

例5-6：神经网络分类（1）

- 使用神经网络对鸢尾花数据集进行分类分析

1）读取数据，统计样本特征

```
filename = 'data\iris.data'
data = pd.read_csv(filename, header = None)
data.columns = ['sepal length','sepal width','petal length','petal width','class']
data.iloc[0:5,:]
print( data['class'].value_counts() )
data.groupby('class').mean()
```

每类花样本数

```
Iris-setosa        50
Iris-virginica     50
Iris-versicolor    50
Name: class, dtype: int64
```

每类花各特征的均值

class	sepal length	sepal width	petal length	petal width
Iris-setosa	5.006	3.418	1.464	0.244
Iris-versicolor	5.936	2.770	4.260	1.326
Iris-virginica	6.588	2.974	5.552	2.026

每类花各特征的方差

class	sepal length	sepal width	petal length	petal width
Iris-setosa	0.124249	0.145180	0.030106	0.011494
Iris-versicolor	0.266433	0.098469	0.220816	0.039106
Iris-virginica	0.404343	0.104004	0.304588	0.075433

例5-6：神经网络分类（2）

2）数据预处理，类别使用整数表示

```
data.loc[ data['class'] == 'Iris-setosa', 'class' ] = 0
data.loc[ data['class'] == 'Iris-versicolor', 'class' ] = 1
data.loc[ data['class'] == 'Iris-virginica', 'class' ] = 2
X = data.iloc[:,0:4].values.astype(float)
y = data.iloc[:,4].values.astype(int)
```

3）创建神经网络分类器，训练网络结点连接权重以及偏差

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(solver='lbfgs',alpha=1e-5,hidden_layer_sizes=(5, 5),
      random_state=1)
mlp.fit(X,y)
mlp.score(X,y)
```

2个隐藏层，每层5个结点

训练数据集上预测正确率达到98.6%

4）分类器性能评估

```
from sklearn import metrics
y_predicted = mlp.predict(X_test)
print("Classification report for %s" % clf)

print (metrics.classification_report(y_test, y_predicted) )
print( "Confusion matrix:\n", metrics.confusion_matrix(y_test, y
      _predicted) )
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.98	0.98	0.98	50
2	0.98	0.98	0.98	50
avg / total	0.99	0.99	0.99	150

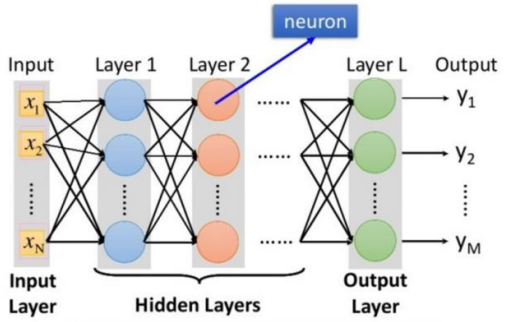
```
Confusion matrix:
 [[50  0  0]
 [ 0 49  1]
 [ 0  1 49]]
```

总结：神经网络应用

- 分类：MLPClassifier类
  - 二分类问题，输出层只需要1个结点
    - 输出层使用sigmoid激活函数
    - 损失函数，采用bianry_crossentropy
  - 多分类问题，输出层需要多个输出结点，每个结点对应一种类型，输出值表示属于该类型的概率
    - 输出层使用softmax激活函数
    - 损失函数，采用categoricial_croosentropy
- 回归：MLPRegressor类
  - 预测连续值，而不是离散的标签，如:明天的天气温度，地区房屋价格等
  - 神经网络的输出层只有一个结点，不需要激活函数
  - 损失函数，使用线性回归的目标函数，均方误差MSE
- 神经网络初始化的参数，称为超级参数
  - 调参，指通过人工尝试或算法，调整初始化参数，改善模型性能

5.5.4 深度学习

- 神经网络训练速度非常慢，学习算法存在梯度消失的问题
- 2006年，加拿大科学家Hinton与合作者发表了深度学习（Deep Learning）论文
  - 借助统计力学里“玻尔兹曼分布”的概念，改造了神经网络的学习机制
- 深度学习基本思想
  - 从输入的数据中进行预先训练，以发现数据自身的重要特征
  - 根据提取的特征建立初始化的神经网络，然后再基于分类等标签进行学习，对网络参数进行微调
- 深度学习得益于GPU、大数据的发展
  - GPU（图形处理器，Graphic Processing Unit）提供了强大的计算能力
  - 可以构造拥有十多个隐藏层，数十亿个结点的深度神经网络
  - 需要上千万的样本进行网络训练，学习参数
- 深度学习就是具有很多隐藏层（超过一层）、每个隐藏层具有很多结点的神经网络
- 处理不同领域数据，研究扩展版本
  - 图像、语音：卷积神经网络（CNN）
  - 文本：递归神经网络（RNN）、长短期记忆网络（LSTM）



思考与练习

1. 调整MLP分类器的参数solver，比较不同参数的模型在鸢尾花数据集上的分类性能。
2. 在MLP训练函数fit()前后增加计时功能，设置不同隐层数目，比较训练所耗费的时间，以及模型分类的准确性。MLP模型是否结点越多分类性能越好？

【提示：】计时函数：import time

综合练习

1. 从网站上收集了上海松江大学城附近房屋的特征数据，以及相应的房价，保存在文件house_price文件中。利用数据集实现以下分析目标。

   1）使用k-menas算法对房屋进行聚类分析，找出合适的k值，结合房产市场并对聚类结果进行说明。

   2）使用线性分类器对房产数据进行拟合，并使用模型预测自己希望购买的房屋的价格。

   【提示】：首先通过统计、可视化等过程对数据集进行探索性分析，然后再使用算法建立分析模型。

2. 葡萄酒数据集（wine.data）搜集了法国不同产区葡萄酒的化学指标。试建立决策树、SVM和神经网络3种分类器模型，比较各种分类器在此数据集上的效果。

   【提示】：每种分类器，需要对参数进行尝试，找出此种分类算法的较优模型，再与其他分类器性能进行比较。