小象学院
ChinaHadoop.cn

第七课（第19-21课时）

预测和分类

- 分类和预测的基本概念
- 分类和预测的基本方法
- 模型的评价方法
- 相关的数据集处理方式
- 更快的预测方式

前情回顾

小象学院 ChinaHadoop.cn

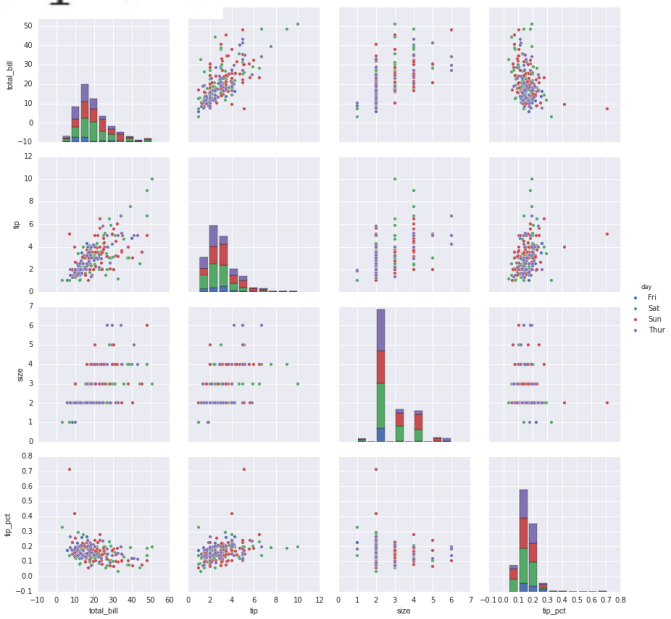
- 分析单个变量：各种方法
- 分析多个变量：各种方法
- 回归分析：确认变量之间的关系

$$Y = f(X_1, X_2, ..., X_{p-1}) + \epsilon$$

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_{p-1} X_{p-1} + \epsilon$$

		自变量：连续型	自变量：类别型
因变量	连续型	相关分析 回归分析	t-检定 方差分析
因变量	类别型	广义线性模型	卡方检定

- 基于模拟的方法：重抽样
  - 可对任何统计量进行
  - 1.显著性的检验
  - 2.置信区间（不确定性）的度量



回归分析

小象学院 ChinaHadoop.cn

- **练习：回归分析：**
  - tip~ total_bill+size

类似R的方法

```
>>import pandas as pd
>>import statsmodels.formula.api as sm
>>result=sm.ols(formula="tip~total_bill+size",data=tips).fit()
>>print result.params
>>print result.summary()
```

```
In [83]: print result.summary()
                            OLS Regression Results
==============================================================================
Dep. Variable:                    tip   R-squared:                       0.468
Model:                            OLS   Adj. R-squared:                  0.463
Method:                 Least Squares   F-statistic:                     105.9
Date:                Sun, 03 Jul 2016   Prob (F-statistic):           9.67e-34
Time:                        00:08:06   Log-Likelihood:                -347.99
No. Observations:                 244   AIC:                             702.0
Df Residuals:                     241   BIC:                             712.5
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
Intercept      0.6689      0.194      3.455      0.001         0.288     1.050
total_bill     0.0927      0.009     10.172      0.000         0.075     0.111
size           0.1926      0.085      2.258      0.025         0.025     0.361
==============================================================================
Omnibus:                       24.753   Durbin-Watson:                   2.100
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.169
Skew:                           0.545   Prob(JB):                     9.43e-11
Kurtosis:                       4.831   Cond. No.                         67.6
==============================================================================
```

```
In [100]: x=np.array([tips["total_bill"],tips["size"]]).T

In [101]: y=np.array([tips["tip"]]).T

In [102]: clf=linear_model.LinearRegression()

In [103]: clf.fit(x,y)
Out[103]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [104]: clf.intercept_
Out[104]: array([ 0.66894474])

In [105]: clf.coef_
Out[105]: array([[ 0.09271334,  0.19259779]])
```

预测与分类：任务描述

小象学院 ChinaHadoop.cn

- 理解预测和分类的目的
- 了解各种预测和分类算法
- 掌握如何根据因变量和自变量的类型来确定模型和算法
- 掌握对模型的评价方法
- 理解和了解对数据集的操作

分类和预测

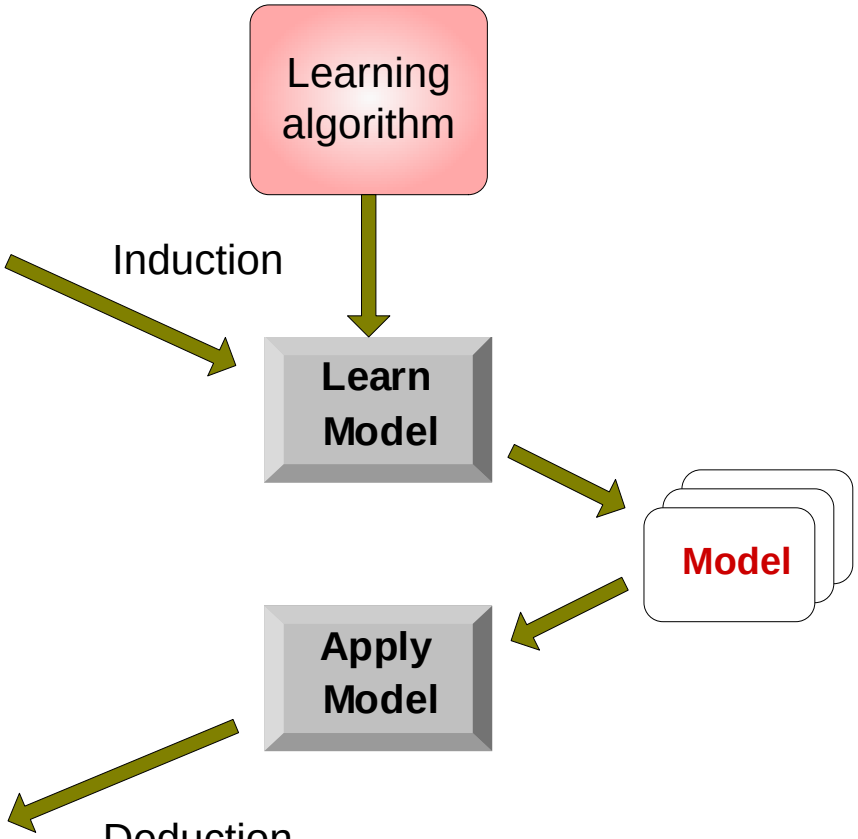
小象学院 ChinaHadoop.cn

- **什么是分类？**
  - 用一部分属性预测另一部分属性(类别属性)，例如，
    - 预测肿瘤是良性还是恶性
    - 区分信用卡交易是否是盗刷
    - 将新闻划分为财经、天气、娱乐、体育，等
    - 文本的情感分析
- **什么是预测？**
  - 根据自变量（解释变量）给出因变量（预测变量）的估计值，例如，
    - 预测股票的收益率
    - 预测信用卡被盗刷的概率
- **所以，分类和预测本质上一回事**
  - 当因变量为类别型（或分类、因素）时，特指分类
  - 当因变量（预测变量）为数值型，也可按规则转换为类别型

分类和预测：构建方法

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

预测（分类）的常见方法

小象学院 ChinaHadoop.cn

- **回归类的预测和分类**

回归类型	用　途
简单线性	用一个量化的解释变量预测一个量化的响应变量
多项式	用一个量化的解释变量预测一个量化的响应变量，模型的关系是n阶多项式
多元线性	用两个或多个量化的解释变量预测一个量化的响应变量
多变量	用一个或多个解释变量预测多个响应变量
Logistic	用一个或多个解释变量预测一个类别型响应变量
泊松	用一个或多个解释变量预测一个代表频数的响应变量
Cox比例风险	用一个或多个解释变量预测一个事件（死亡、失败或旧病复发）发生的时间
时间序列	对误差项相关的时间序列数据建模
非线性	用一个或多个量化的解释变量预测一个量化的响应变量，不过模型是非线性的
非参数	用一个或多个量化的解释变量预测一个量化的响应变量，模型的形式源自数据形式，不事先设定
稳健	用一个或多个量化的解释变量预测一个量化的响应变量，能抵御强影响点的干扰

回归类的预测和分类：线性回归

- **变量类型：**
  - 数值型~数值型+数值型
- **求解算法：**
  - 最小二乘法 OLS
  - 或者最速下降法
- **预测方法：**
  - 利用回归方程输入新X得到新Y
- **Python模块：**
  - from sklearn import linear_model
  - clf = linear_model.LinearRegression()
  - clf.fit(x, y)
  - clf.predict (newx)

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_{p-1} X_{p-1} + \epsilon$$

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + ... + \hat{\beta}_{p-1} X_{p-1}$$

$$\hat{\epsilon} = Y - \hat{Y}$$

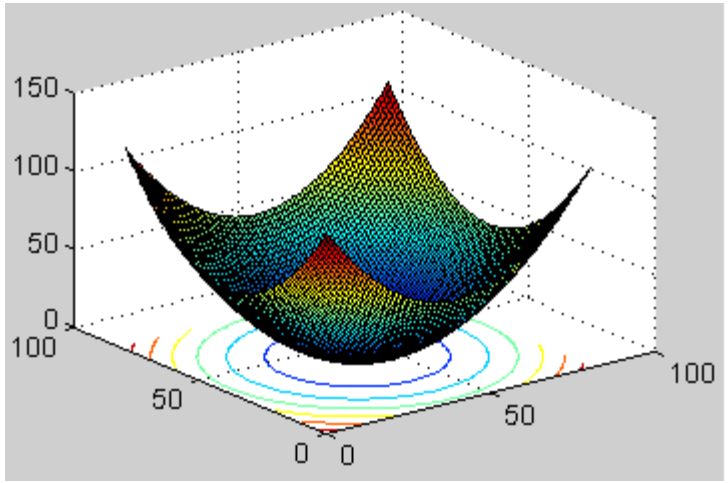
$$S(\beta) = \sum_{i=1}^{n} \epsilon_i^2$$

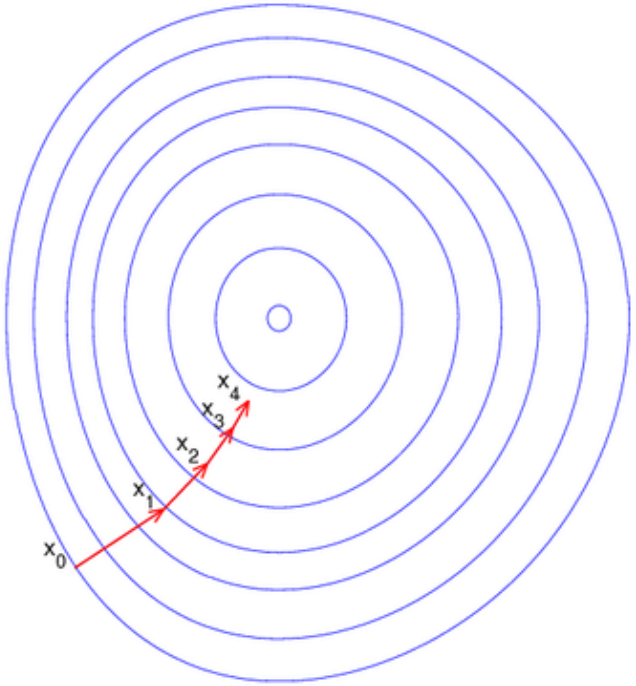
$$\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_{p-1})^T = (X^T X)^{-1} X^T Y$$

回归类的预测和分析：线性回归

小象学院 ChinaHadoop.cn

- **寻找目标函数的最优解：**
- **最速下降法：当特征（预测变量）数目比较多的时候更高效**
  - 步长太大：可能不收敛
  - 步长太小：收敛慢
  - 最速上升法同理





回归类的预测和分类：Logistic回归

小象学院 ChinaHadoop.cn

- **变量类型：**
  - 二值型响应变量~数值型变量

Y=1的概率

优势比

$$\ln(\frac{\pi}{1-\pi}) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

- **优化算法：**
  - 准确率
  - 梯度上升法
- **Python模块：**
  - sklearn.linear_model.LogisticRegression

```python
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X, y)
print(model)
# make predictions
expected = y
predicted = model.predict(X)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

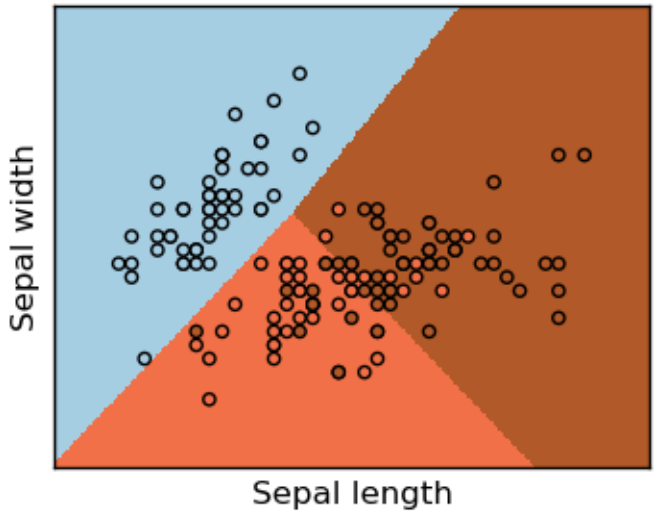
回归类预测：Logistic回归

小象学院 ChinaHadoop.cn

```
# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
Y = iris.target

h = .02  # step size in the mesh

logreg = linear_model.LogisticRegression(C=1e5)

# we create an instance of Neighbours Classifier and fit the data.
logreg.fit(X, Y)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, m_max]x[y_min, y_max].
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```

```
In [6]: iris.data
Out[6]:
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
```

```
In [7]: iris.target
Out[7]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```



回归类的预测和分析：广义线性模型

- **标准线性模型：**

$$\mu_Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

- **广义线性模型：**

$$g(\mu_Y) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

连接函数：
条件均值的函数

- **softmax回归，Logistic回归的扩展，类别型因变量可以有多个取值**

回归类的预测和分类：泊松回归

小象学院 ChinaHadoop.cn

- **变量类型：**
  - 计数型变量（泊松分布）

$$P\{X=n\}=\frac{\lambda^n}{n!}e^{-\lambda},\quad n=0,1,2,\cdots$$

$$g(\mu_Y)=\beta_0+\sum_{j=1}^{p}\beta_j X_j$$

$$\ln(\lambda)=\beta_0+\sum_{j=1}^{p}\beta_j X_j$$

Y的均值
或方差

回归类的预测和分类：岭回归

- 专用于共线性数据分析的有偏估计回归方法
  - 改良的最小二乘估计法

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

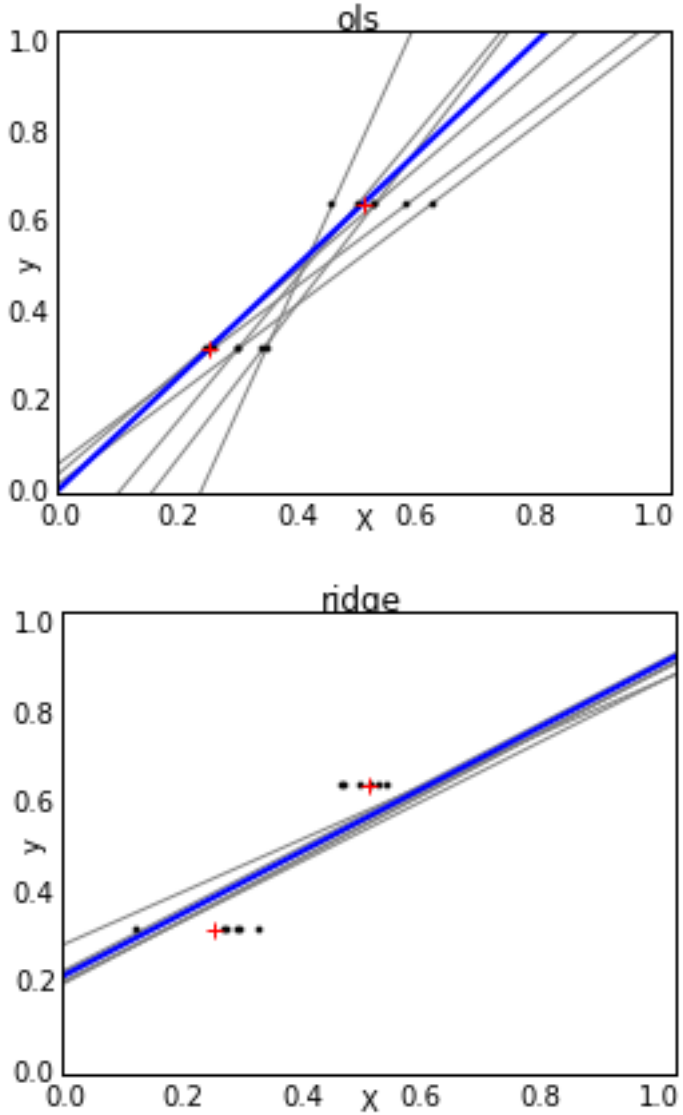
- 当X不是列满秩时，或者某些列之间的线性相关性比较大时，X^TX行列式接近于零，上式计算误差比较大，传统的最小二乘法缺乏稳定性与可靠性

$$\theta(\alpha)) = (X^T X + \alpha I)^{-1} X^T y$$

回归类的预测和分类：岭回归

小象学院 ChinaHadoop.cn

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn import linear_model

X_train = np.c_[.5, 1].T
y_train = [.5, 1]
X_test = np.c_[0, 2].T

np.random.seed(0)

classifiers = dict(ols=linear_model.LinearRegression(),
                   ridge=linear_model.Ridge(alpha=.1))

fignum = 1
for name, clf in classifiers.items():
    fig = plt.figure(fignum, figsize=(4, 3))
    plt.clf()
    plt.title(name)
    ax = plt.axes([.12, .12, .8, .8])

    for _ in range(6):
        this_X = .1 * np.random.normal(size=(2, 1)) + X_train
        clf.fit(this_X, y_train)

        ax.plot(X_test, clf.predict(X_test), color='.5')
        ax.scatter(this_X, y_train, s=3, c='.5', marker='o', zorder=10)

    clf.fit(X_train, y_train)
    ax.plot(X_test, clf.predict(X_test), linewidth=2, color='blue')
    ax.scatter(X_train, y_train, s=30, c='r', marker='+', zorder=10)

    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_ylim((0, 1.6))
    ax.set_xlabel('X')
    ax.set_ylabel('y')
    ax.set_xlim(0, 2)
    fignum += 1

plt.show()

```



回归类的预测和分类：

小象学院 ChinaHadoop.cn

- **Lasso回归**
  - Lasso是对模的长度进行限制,岭回归是对模的平方

- **稳健回归 Robust**
  - 对异常值十分敏感的经典最小二乘回归中的目标函数进行修改
  - 常见稳健回归方法：最小中位平方(least median square；LMS)法
- **Pythong模块**
  - sklearn.linear_model.

分类模型：决策树

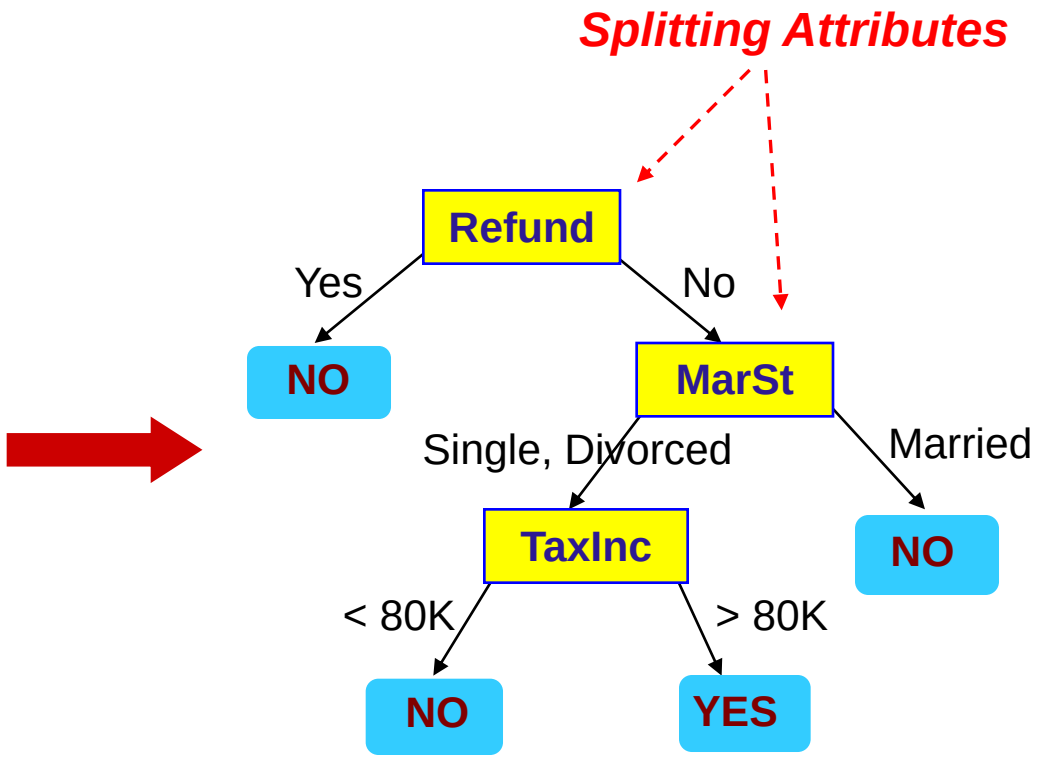
小象学院 ChinaHadoop.cn

➢ 分类模型

类别 类别 数值 类别

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

训练数据

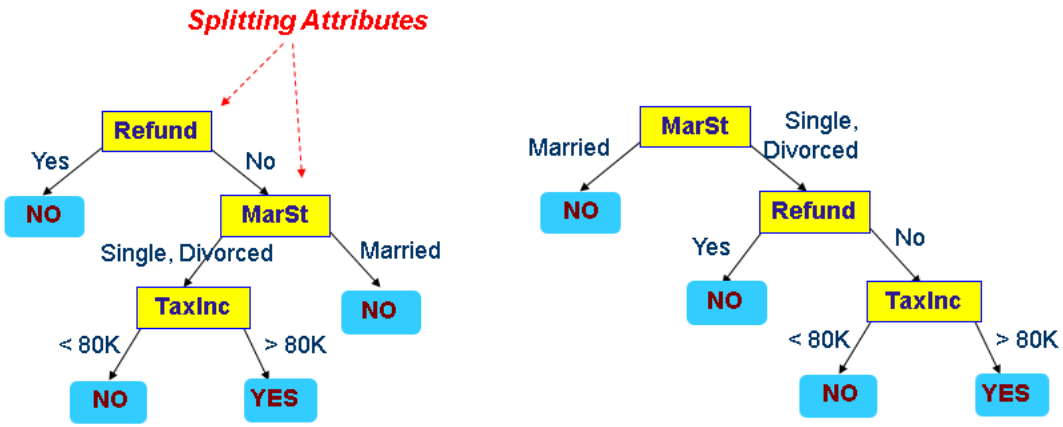


决策树

分类模型：决策树

小象学院 ChinaHadoop.cn

- **决策树的生成：**
  - 贪婪算法：局部最优
  - 根据某一属性对数据进行分裂，以达到某一标准的最优值
- **问题：**
  - 如何分裂数据记录？
    - 选择哪个属性？
    - 如何分裂？
  - 何时停止分裂？



同一个数据集可以产生多种决策树

分类模型：决策树

小象学院 ChinaHadoop.cn

- **贪婪算法:**
  - 具有单一属性分类的节点最佳（到此节点分类最准确）
- **如何度量节点的[不]纯度:**
  - GINI Index
  - 熵

C0: 5
C1: 5

C0: 9
C1: 1

- **GINI index**

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	0
C2	6

P(C1) = 0/6 = 0　　P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	1
C2	5

P(C1) = 1/6　　P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	2
C2	4

P(C1) = 2/6　　P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

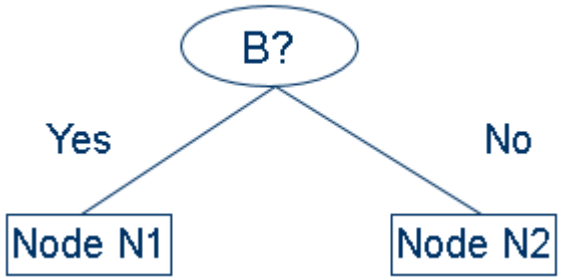
分类模型：决策树

小象学院 ChinaHadoop.cn

- **GINI Index**

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

	Parent
C1	6
C2	6
Gini = 0.500	



Gini(N1)
= 1 – $(5/7)^2$ – $(2/7)^2$
= 0.408

Gini(N2)
= 1 – $(1/5)^2$ – $(4/5)^2$
= 0.320

	N1	N2
C1	5	1
C2	2	4
Gini=0.372		

Gini(Children)
= 7/12 * 0.408 +
5/12 * 0.320
= 0.372

分类模型：决策树

- 熵

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

C1	0
C2	6

P(C1) = 0/6 = 0　　P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	1
C2	5

P(C1) = 1/6　　P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	2
C2	4

P(C1) = 2/6　　P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

分类模式：决策树

小象学院 ChinaHadoop.cn

- **Gain 信息增益**

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

- **信息增益比：防止过拟合**

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

决策树

小象学院 ChinaHadoop.cn

- 常见算法 ID3，C4.5, CART

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(iris.data, iris.target)
```

```
>>> clf.predict(iris.data[:1, :])
array([0])
```

分类模型：朴素贝叶斯

- **变量类型**
  - 类别型因变量~
  - 属性集（Ai）和类变量C都被看做随机变量
- **算法：**
  - 给定属性集(A_1, A_2,…,A_n)
  - 取C的值使得条件概率P(C| A_1, A_2,…,A_n)最大

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C)P(C)}{P(A_1 A_2 \ldots A_n)}$$

分类模型：朴素贝叶斯 Naive Bayes

- **分类方法，例**

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

  - 假定属性相互独立：

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

  - 对所有的i

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

  - 则

$$P(y \mid x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y),$$

分类模型：朴素贝叶斯 Naive Bayes

- **分类方法，例**
  - P(C) = Nc/N
    - e.g., P(No) = 7/10, P(Yes) = 3/10
  - $P(A_i \mid C_k) = |A_{ik}| / N_c$
    - e.g.,
    - P(Status=Married|No) = 4/7
      P(Refund=Yes|Yes)=0
  - 对于数值型的属性

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

分类模型：朴素贝叶斯

小象学院 ChinaHadoop.cn

➢ 例

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (iris.data.shape[0],(iris.target != y_pred).sum()))
Number of mislabeled points out of a total 150 points : 6
```

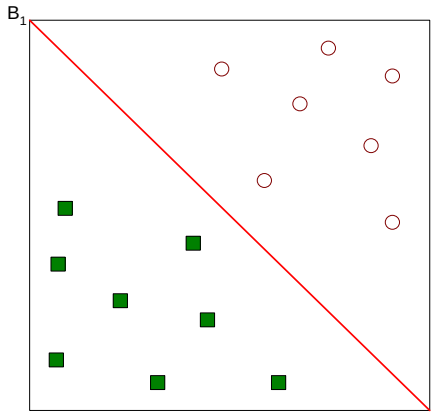
```
In [6]: iris.data
Out[6]:
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
```

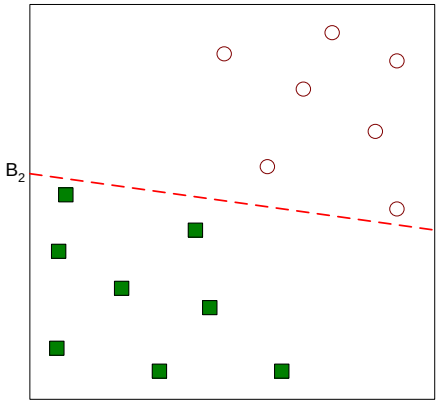
```
In [7]: iris.target
Out[7]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

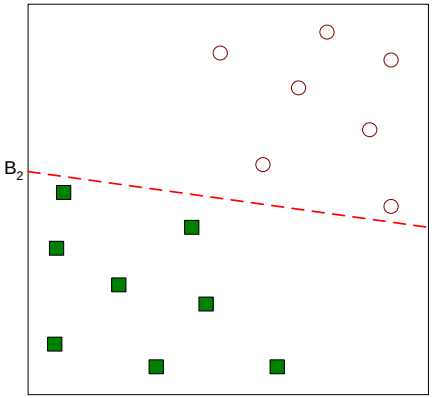
分类模型：支持向量机

小象学院 ChinaHadoop.cn

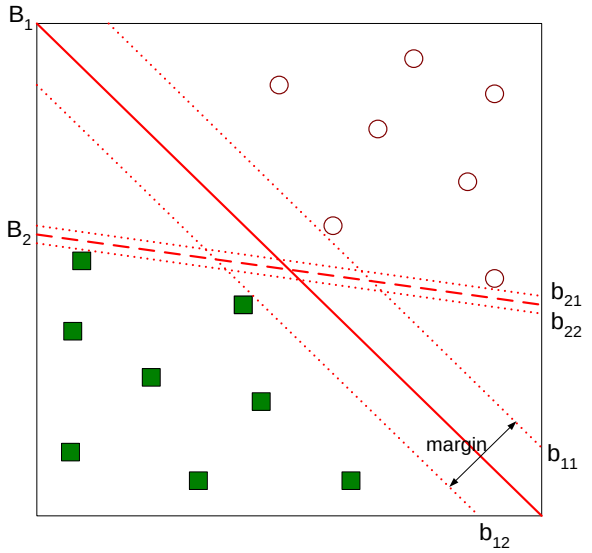
- 哪个边界更好？







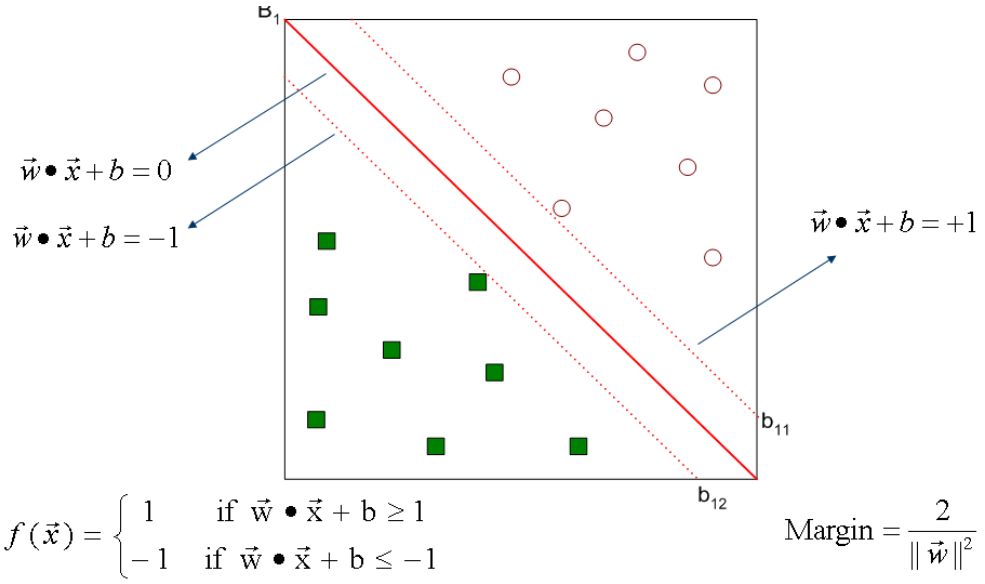
- 最大化“margin”



分类模型：支持向量机

小象学院 ChinaHadoop.cn

➢ 分类模型



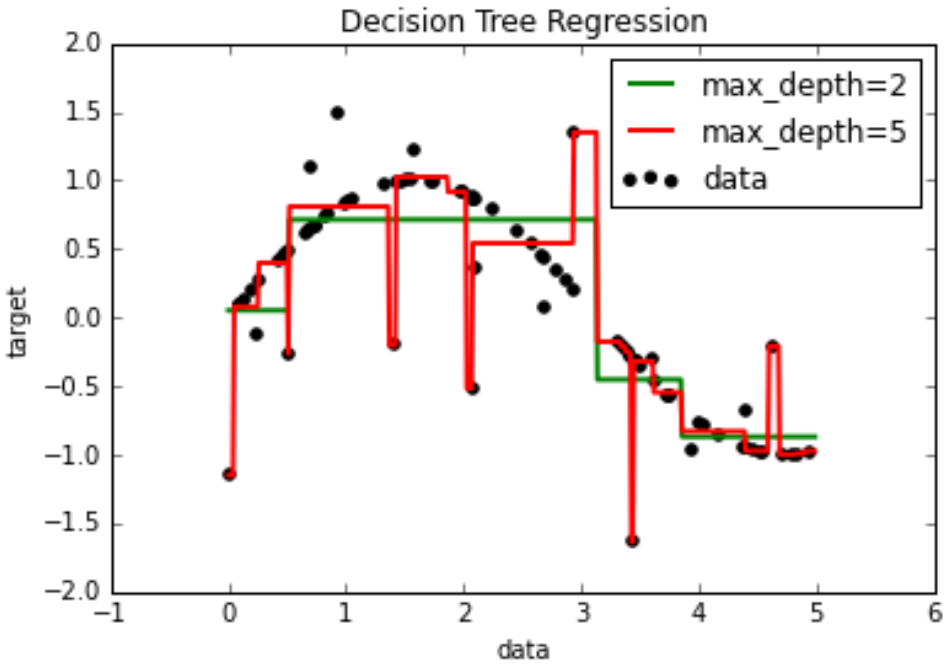
Python

```python
from sklearn import metrics
from sklearn.svm import SVC
# fit a SVM model to the data
model = SVC()
model.fit(X, y)
print(model)
# make predictions
expected = y
predicted = model.predict(X)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

分类和预测：回归树

小象学院 ChinaHadoop.cn

- **CART：classification and regression tree**
  - 把特征空间划分成一系列的矩形区域，然后在每个区域中拟合一个简单的模型（例如：常量）
- **因变量：数值型**

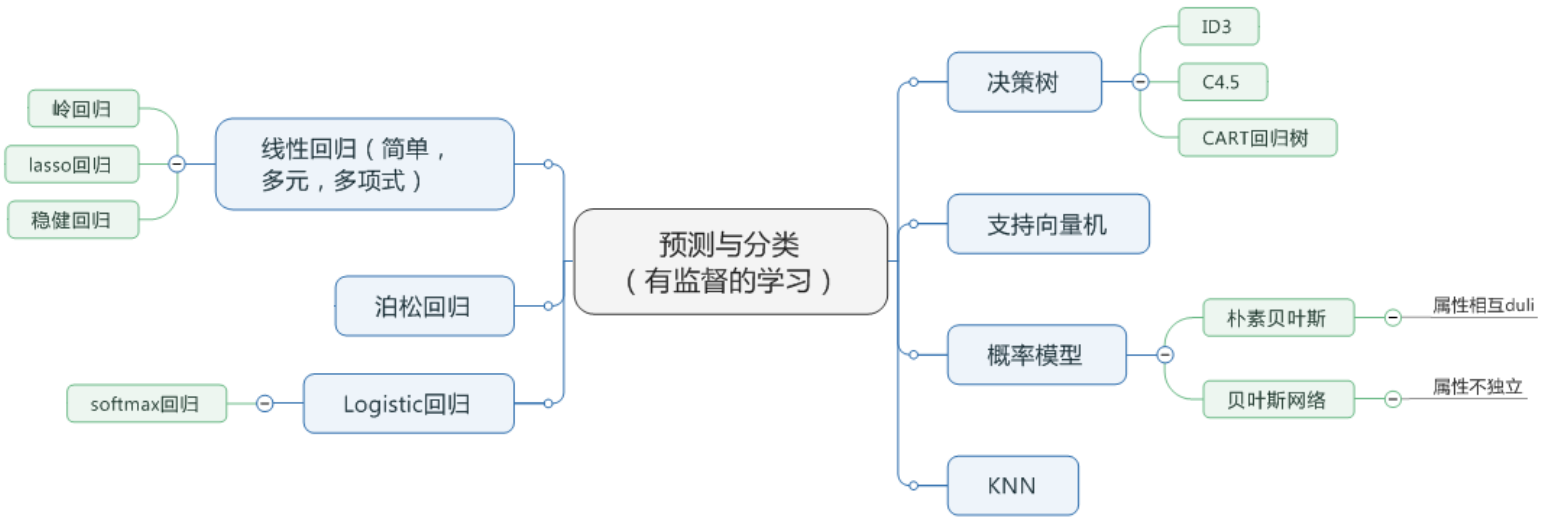



```
19 import numpy as np
20 from sklearn.tree import DecisionTreeRegressor
21 import matplotlib.pyplot as plt
22
23 # Create a random dataset
24 rng = np.random.RandomState(1)
25 X = np.sort(5 * rng.rand(80, 1), axis=0)
26 y = np.sin(X).ravel()
27 y[::5] += 3 * (0.5 - rng.rand(16))
28
29 # Fit regression model
30 regr_1 = DecisionTreeRegressor(max_depth=2)
31 regr_2 = DecisionTreeRegressor(max_depth=5)
32 regr_1.fit(X, y)
33 regr_2.fit(X, y)
34
35 # Predict
36 X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
37 y_1 = regr_1.predict(X_test)
38 y_2 = regr_2.predict(X_test)
39
40 # Plot the results
41 plt.figure()
42 plt.scatter(X, y, c="k", label="data")
43 plt.plot(X_test, y_1, c="g", label="max_depth=2", linewidth=2)
44 plt.plot(X_test, y_2, c="r", label="max_depth=5", linewidth=2)
45 plt.xlabel("data")
46 plt.ylabel("target")
47 plt.title("Decision Tree Regression")
48 plt.legend()
49 plt.show()
50
```

分类和预测：小结

小象学院 ChinaHadoop.cn

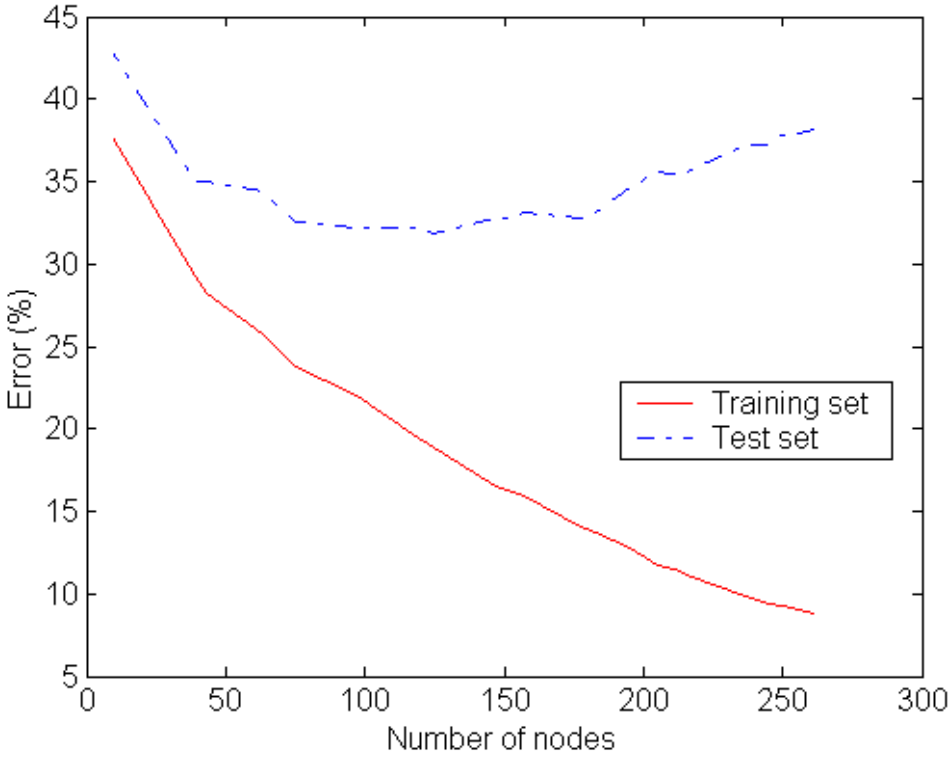
- **模型：**



分类和预测：小结

小象学院 ChinaHadoop.cn

- **可能遇到的问题：**
  - 拟合不足和过度拟合
  - 缺失值
  - 分类成本



对模型的评价

- **估计泛化误差（验证误差）**

- **建模过程中，学习算法只能访问训练数据集**
  - 使用确认集（训练集中分出）

- **模型选择**
  - 两模型的泛化误差接近时，选择较简单的模型
  - 复杂模型更有可能拟合偶然性的数据，所以评价模型时需要考虑模型复杂度

评价模型：准确率

小象学院 ChinaHadoop.cn

➢ 混淆矩阵

		预测值	
		Class=Yes	Class=No
真实值	Class=Yes	f11	f10
	Class=No	f01	f00

f11: TP (true positive)
f10: FN (false negative)
f01: FP (false positive)
f00: TN (true negative)

- 准确率

$$\text{Accuracy} = \frac{f11+f00}{f11+f10+f01+f00} = \frac{TP+TN}{TP+TN+FP+FN}$$

评价模型：代价矩阵

	预测值		
真实值	C(i|j)	Class=Yes	Class=No
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): 错误的将j 分类为i 的代价

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Accuracy = 80%

Cost = 3910

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

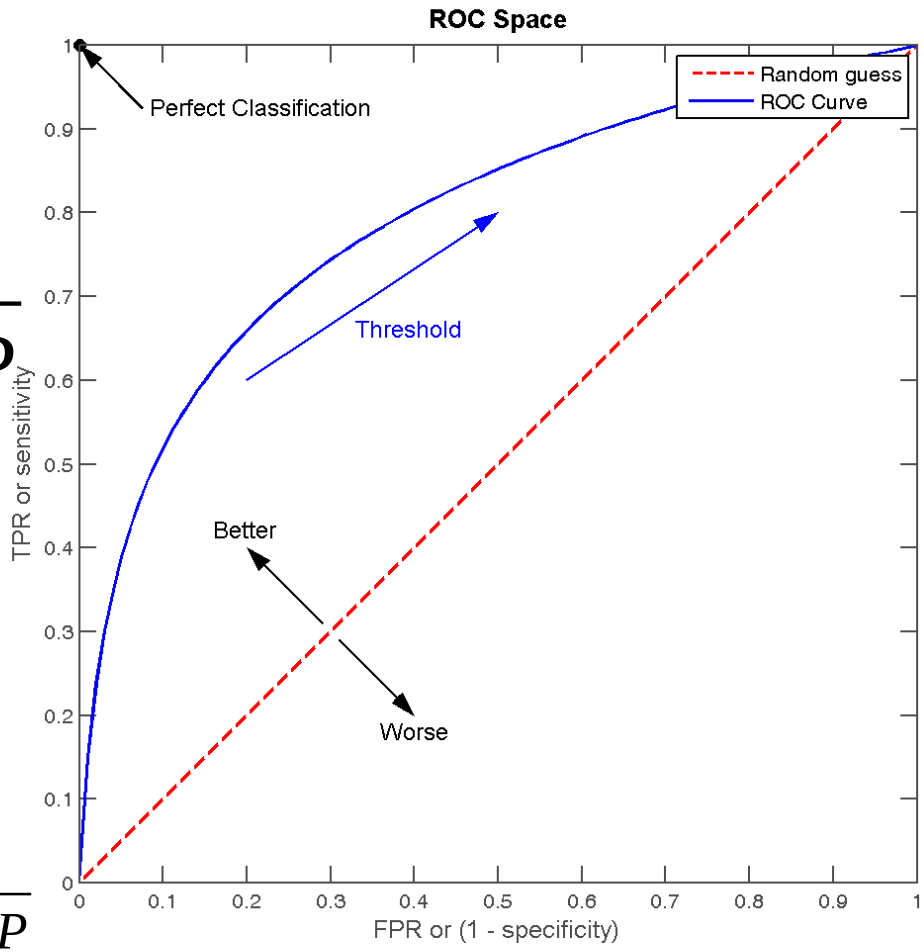
评价模型：ROC

$$\text{Precision (p)} = \frac{TP}{TP + FP}$$

$$\text{Recall (r)} = \frac{TP}{TP + FN}$$

$$\text{F-measure (F)} = \frac{2rp}{r + p} = \frac{2TP}{2TP + FN + FP}$$

$$AUC = AreaUnderCurve$$



评价模型：

小象学院 ChinaHadoop.cn

- **交叉验证： Cross validation**
  - 将数据分为不重叠的k份
  - k-fold: k-1份作为训练集，剩余一份作为测试集
  - k=n（尽可能多的训练数据，计算要求高）
  - 记录所有次预报误差的平方和
  - k~10

对数据集的特殊处理:

小象学院 ChinaHadoop.cn

- **Bagging（bootstrap aggregating）**
  - 训练多轮，每轮的训练集由初始训练集中随机取出（可放回）n个训练样本组成
  - 训练之后得到一个预测函数序列h_1,h_n，最终的预测函数H对分类问题采用投票方式，对回归问题采用简单平均方式判别
  - http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- **Ensemble methods**

对数据集的特殊处理

小象学院 ChinaHadoop.cn

- **Boost（主要是Adaboost）**
  - 对每一个训练赋予相同的权重1/n，训练t轮，对失败的训练列赋予较大的权重，也就是让学习算法在后续的学习中集中对比较难的训练列进行训练
  - 从而得到一个预测函数序列，预测效果好的预测函数权重大，反之小
  - 最终的预测函数对分类问题采用有权重的投票方式，对回归问题采用加权平均的方式
  - 类似bagging方法，训练k关注前k-1分类器中错误，非随机取样
  - http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

对数据集的特殊处理

小象学院 ChinaHadoop.cn

- **Rand Forest**：随机森林，
  - 随机的方式建立一个森林，由多个无关联的决策树组成
  - 分类结果由所有的决策树的结果的众数决定
  - 行采样：采用有回放的方式，不容易over-fitting
  - 列采样：从M个feature中，选择m个。
  - 或不需要剪枝

  - http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

对数据集的特殊处理

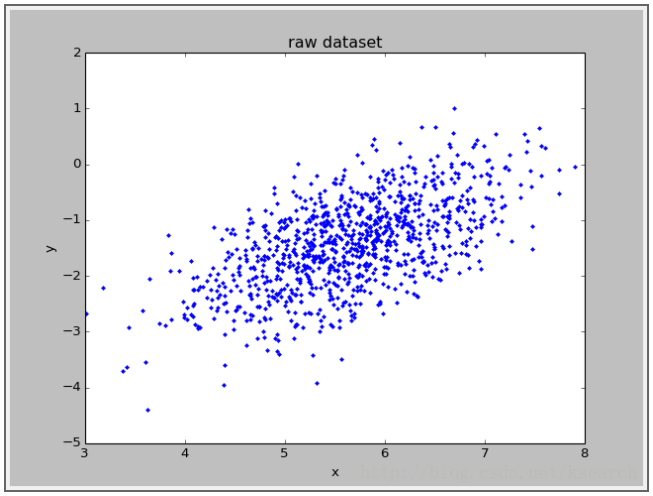
小象学院 ChinaHadoop.cn

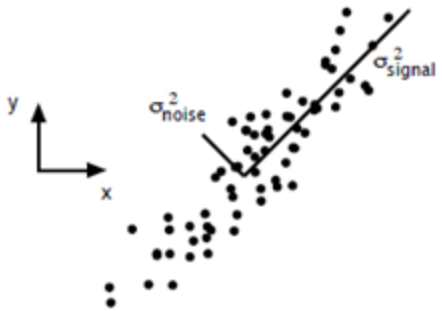
- **不平衡数据**
  - 过抽样和欠抽样

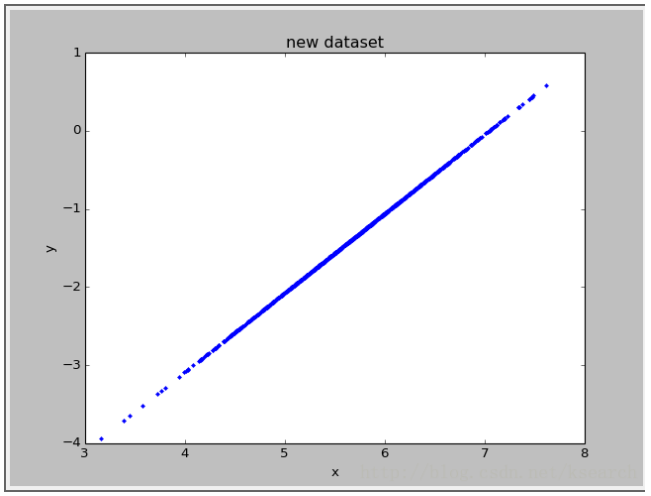
对数据集的特殊处理

小象学院 ChinaHadoop.cn

- **降维处理：PCA**
  - 通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分








```
>>> import numpy as np
>>> from sklearn.decomposition import PCA
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> pca = PCA(n_components=2)
>>> pca.fit(X)
PCA(copy=True, n_components=2, whiten=False)
>>> print(pca.explained_variance_ratio_)
[ 0.99244...  0.00755...]
```

对数据集的特殊处理

$$Av = \lambda v$$
$$A = Q\Sigma Q^{-1}$$

小象学院 ChinaHadoop.cn

- **降维处理：SVD 奇异值分解**
  - 奇异值分解适用于任意的矩阵分解

$$A = U\Sigma V^T$$

$$(A^T A)v_i = \lambda_i v_i$$

$$\sigma_i = \sqrt{\lambda_i}$$

$$u_i = \frac{1}{\sigma_i} A v_i$$

$$A_{m\times n} \approx U_{m\times r}\Sigma_{r\times r}V^T{}_{r\times n}$$

对数据集的特殊处理

小象学院 ChinaHadoop.cn

- 降维处理：因子分析

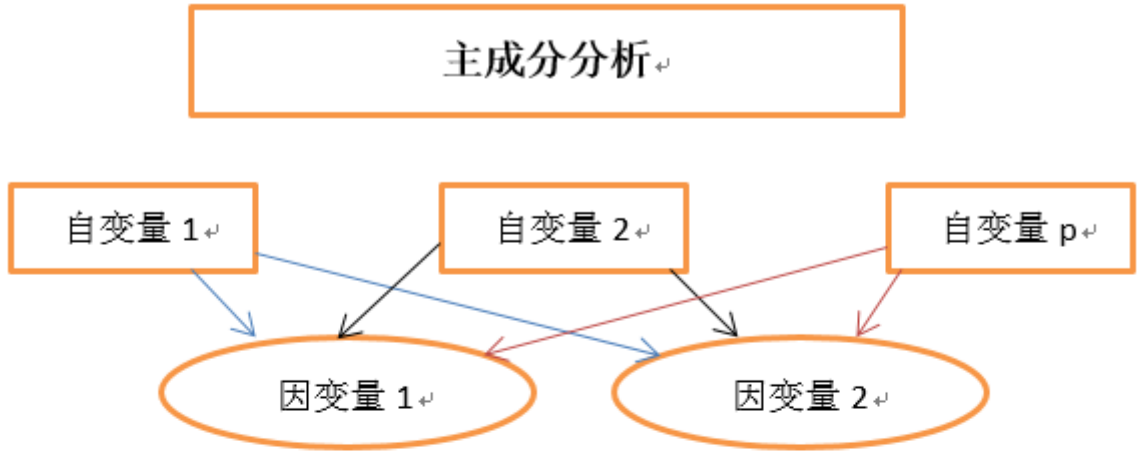


图 2：主成分分析图

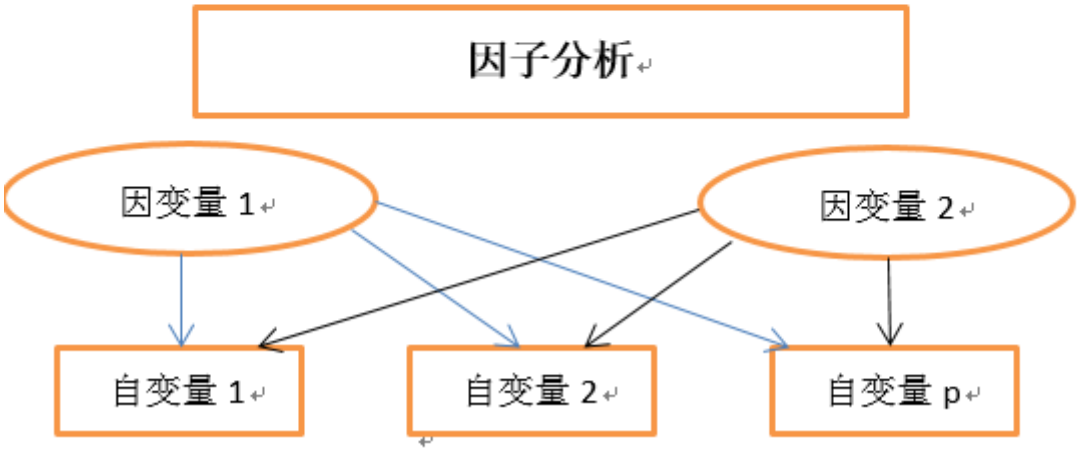


图 3：因子分析图

更快速的预测和分类

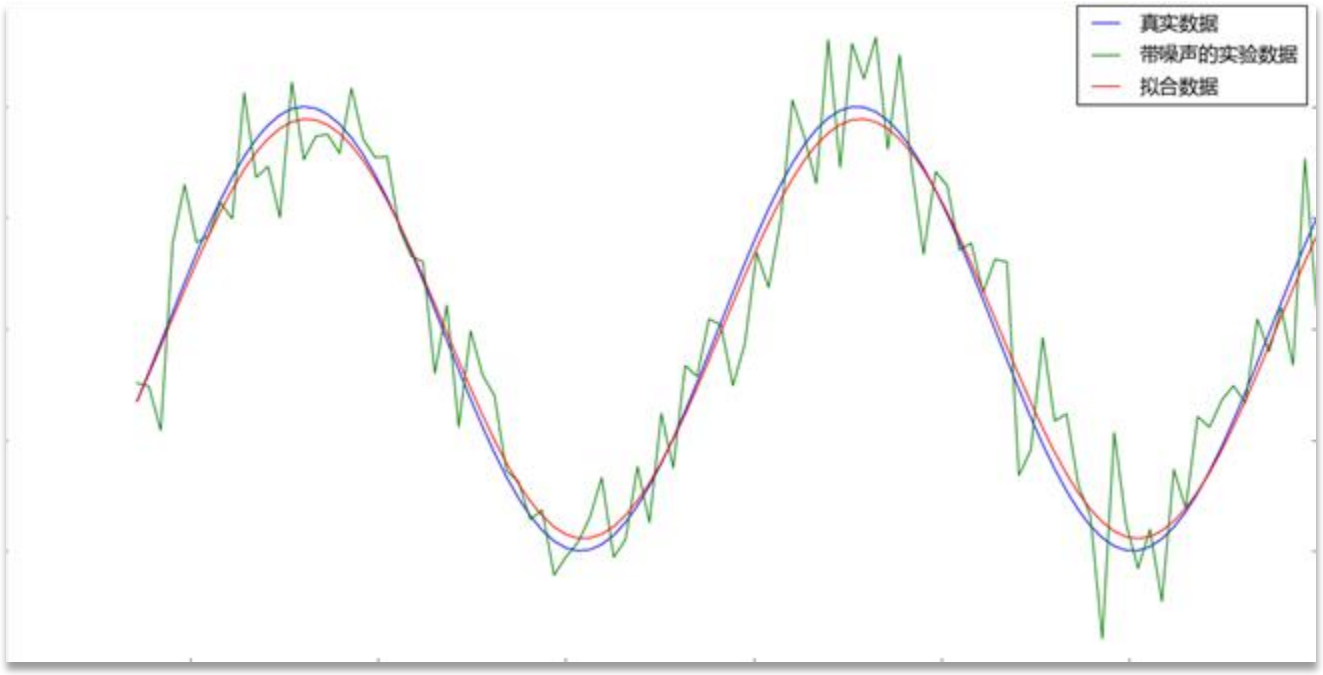
小象学院 ChinaHadoop.cn

- 平滑

更快速的预测和分类

小象学院 ChinaHadoop.cn

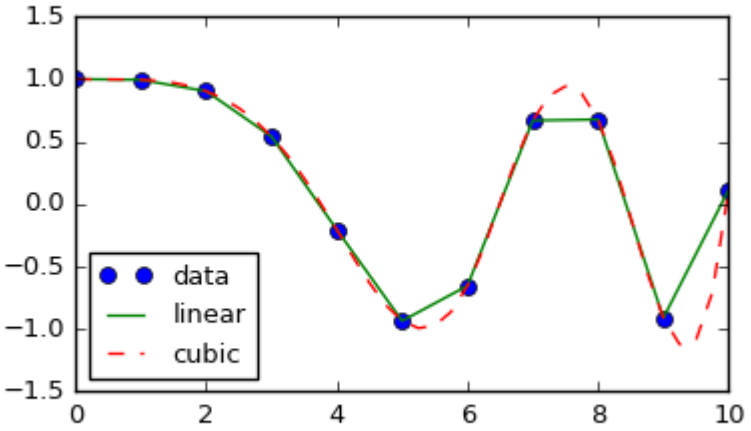
- 拟合

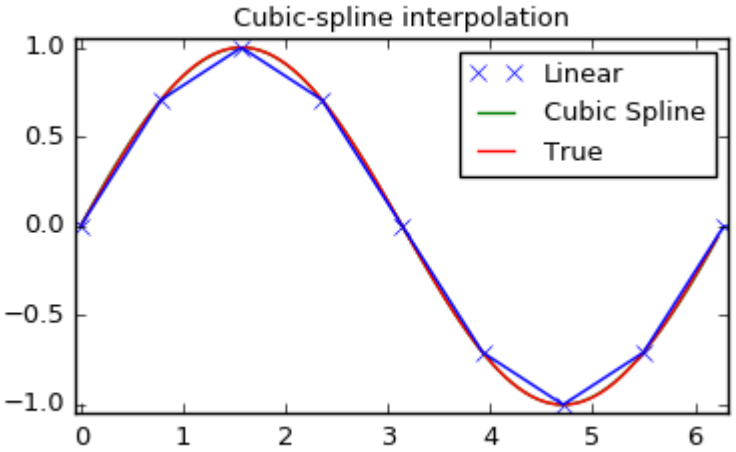


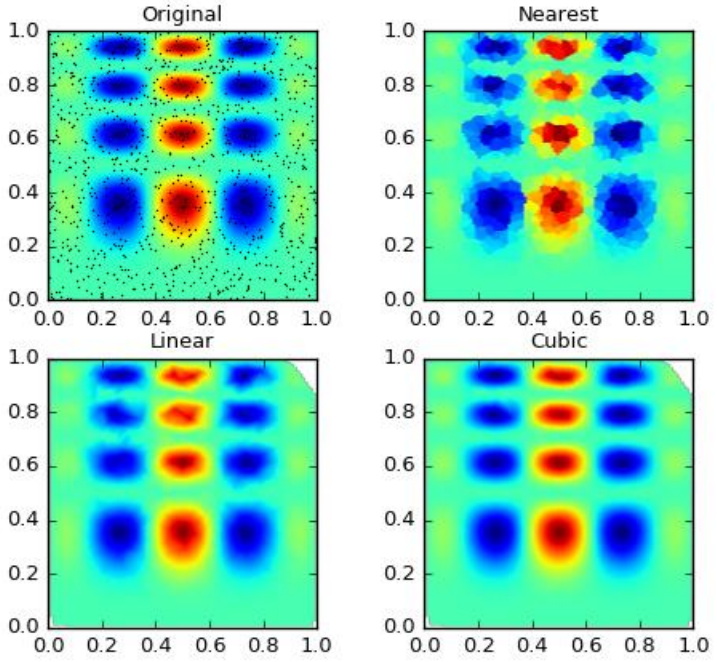
更快速的预测和分类

小象学院 ChinaHadoop.cn

- **插值**
  - 线性
  - 多项式
  - 样条



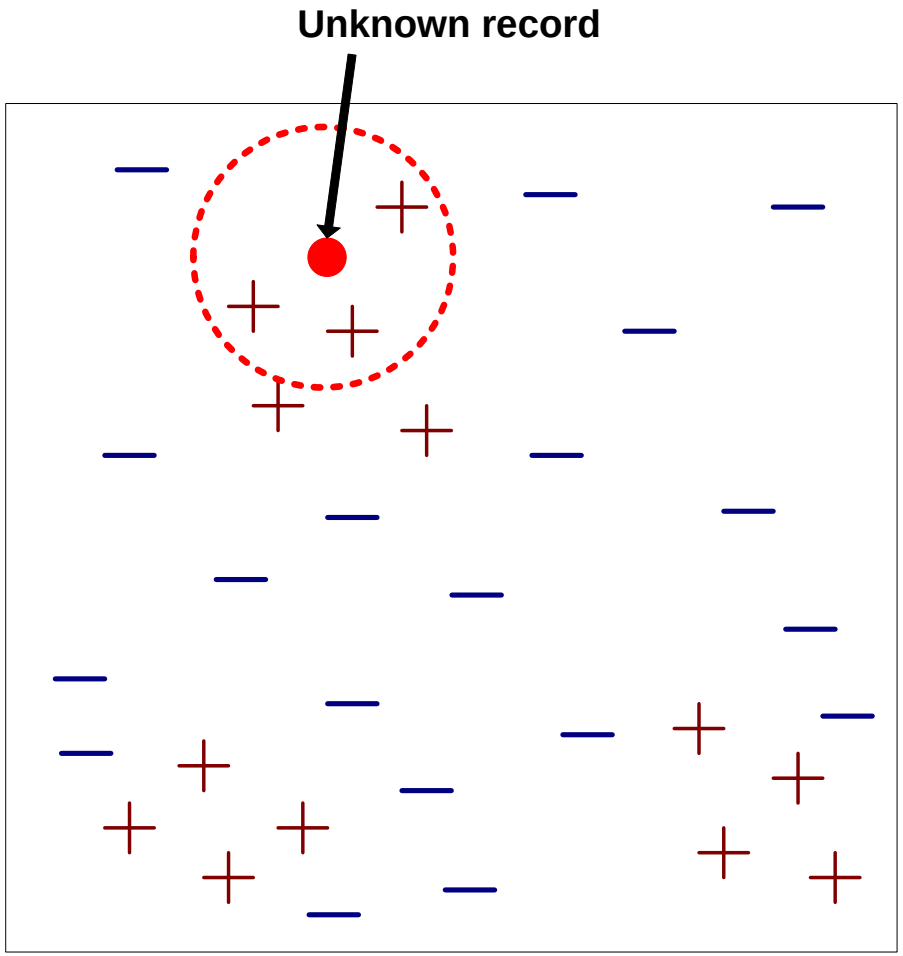




更快速的预测和分类

小象学院 ChinaHadoop.cn

- **KNN-最邻近分类**
  - 定义数据集中记录之间的距离
  - 定义参数k：临近数
  - 分类时识别最近的k个紧邻
  - 由k个近邻的类别对未知记录进行投票
- **更多话题：**
  - 数据点之间的距离



小象学院
ChinaHadoop.cn

联系我们：

- 新浪微博：ChinaHadoop
- 微信公号：ChinaHadoop
- 网站：http://chinahadoop.cn
- 问答社区：http://wenda.ChinaHadoop.cn