Python与数据科学导论-10

—— 有监督学习，分类器基础

信息科学与技术学院

胡俊峰

内容

- 分类问题概述
- 条件概率与贝叶斯推断
- 文本分类与多项分类器

机器学习

- 假定存在一个预测函数 y = F（x）
  - 输入为一组特征
  - 输出为一个预测空间的概率分布
- 函数F称为 **模型（model）**
- 通过训练集学习得到模型的参数称为 **回归 （regression，fit）**
- 根据特征计算输出概率分布或返回最大概率解称为 **预测（prediction）**

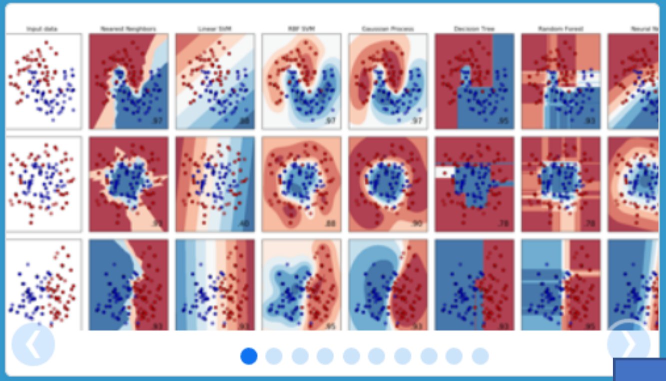
分类 与 回归（机器学习）

- 对于一个样本集，如果能找到一个合理的分类函数，使得：

  $F(X) \approx 1$ (当$Y = 1$)； $F(X) \approx 0$ (当$Y = 0$)

- 则可以称 我们找到了一个原样本集的一个‘似然’函数。
- 如果F是以最大概率符合样本数据，则F称为最大似然函数。

常用的机器学习软件包



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

模型参数回归

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

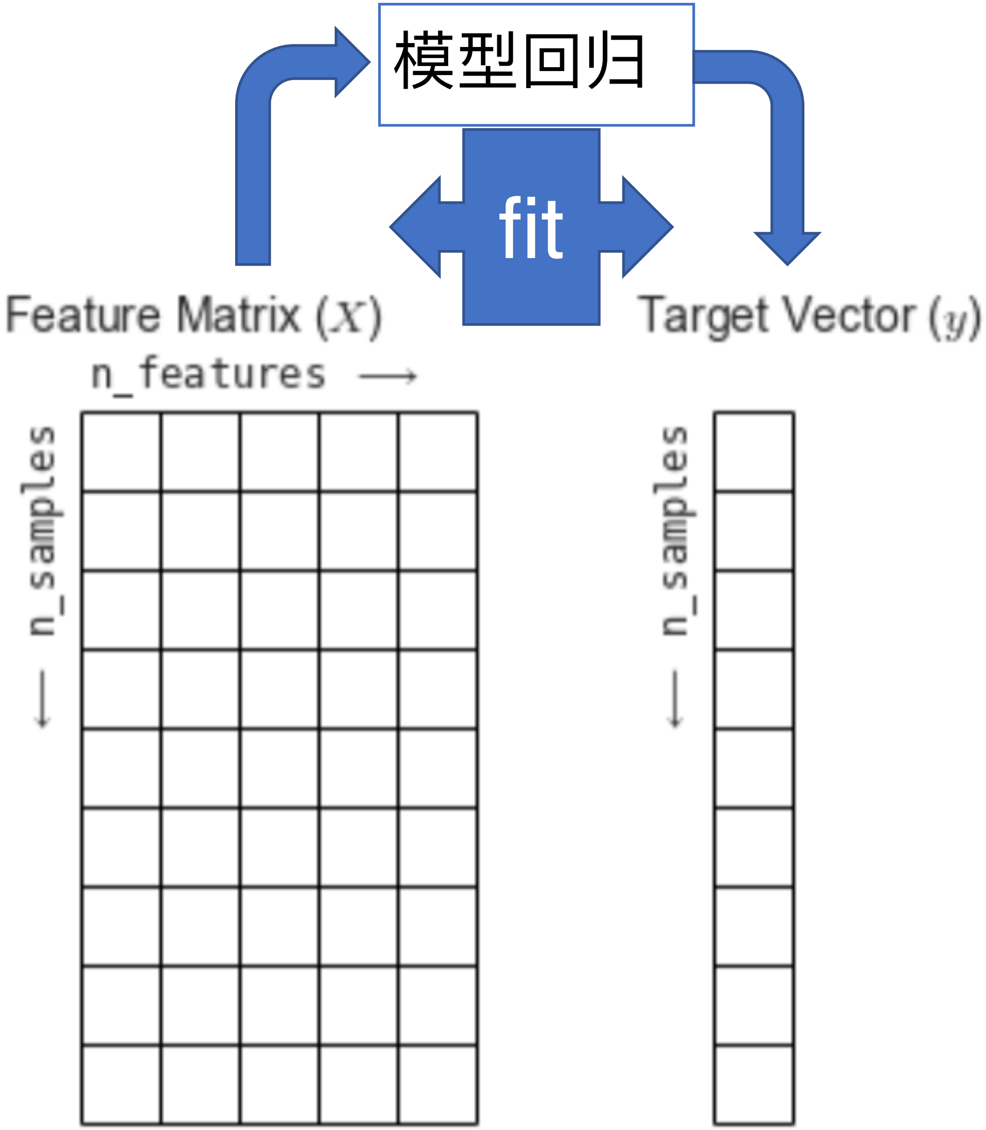
训练数据一般组织形式：

数据集一般包括：

训练集： 用于训练模型

验证集： 用于调整模型参数

测试集： 用于评测模型效果



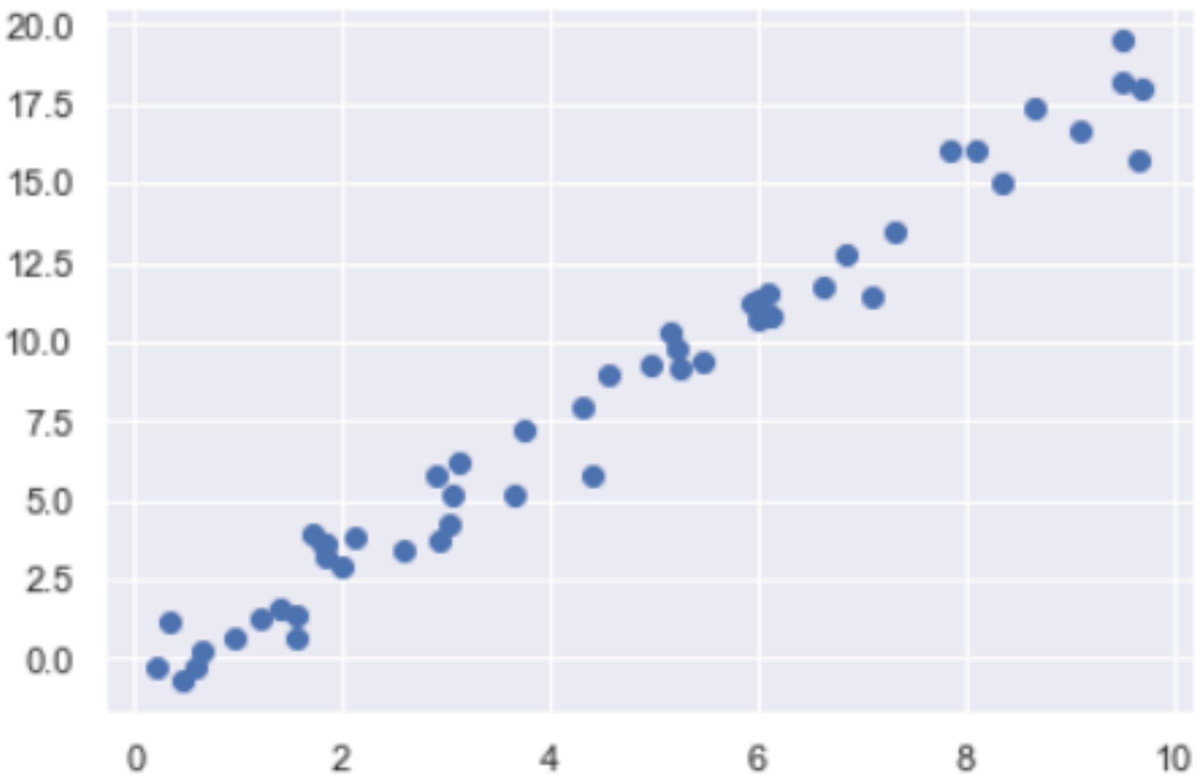
看一个线性回归的例子 模型： y = ax + b

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

rng = np.random.RandomState(42) # 设置伪随机数种子
x = 10 * rng.rand(50)
y = 2 * x - 1 + rng.randn(50) # y = 2 * x - 1 + 0-1随机
plt.scatter(x, y);
```

Numpy 生成随机数据

Pyplot画图



线性回归（参数回归与线性拟合）

```python
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
print(model)
```

选择模型

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```python
X = x[:, np.newaxis]
X.shape
```

```
(50, 1)
```

```python
model.fit(X, y)
```

拟合模型参数

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```python
model.coef_ ,model.intercept_
```

观察模型参数-斜率，截距

```
(array([1.9776566]), -0.903310725531164)
```

模型预测

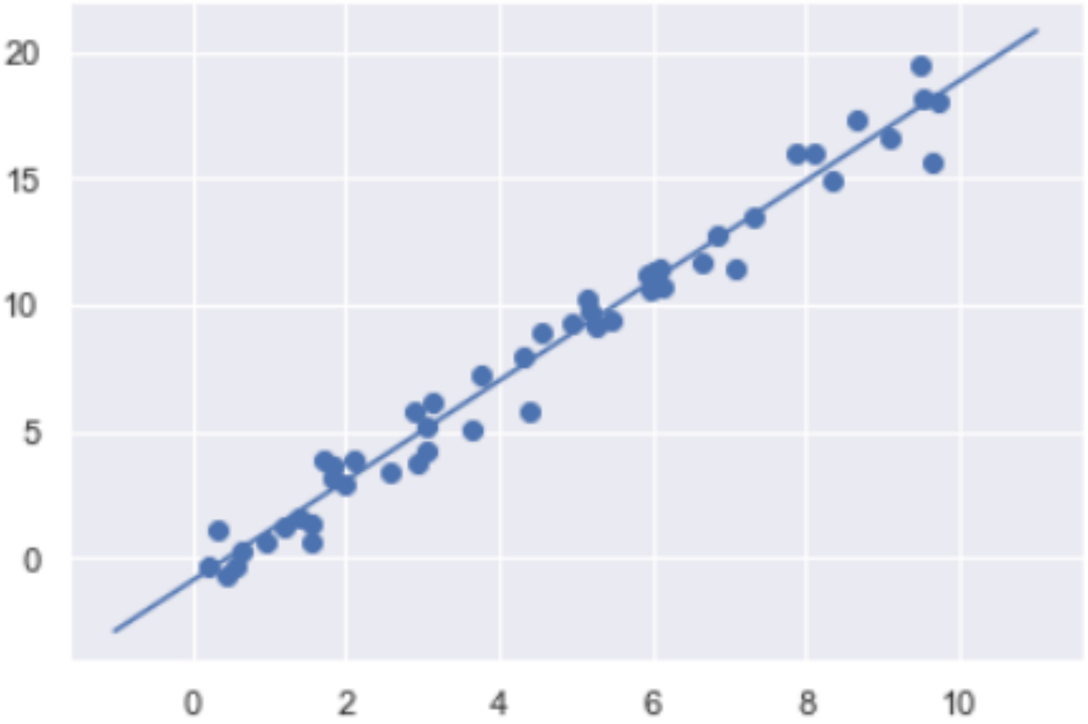
```python
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
xfit, yfit
```

使用模型进行预测

```
(array([-1.        ,  0.33333333,  1.66666667,  3.        ,  4.33333333,
         5.66666667,  7.        ,  8.33333333,  9.66666667, 11.        ]),
 array([-2.88096733, -0.24409186,  2.39278361,  5.02965908,  7.66653454,
        10.30341001, 12.94028548, 15.57716094, 18.21403641, 20.85091188]))
```

```python
plt.scatter(x, y)
plt.plot(xfit, yfit);
```

输出显示模型生成结果



分类与回归

- 对于一个样本集，如果能找到一个合理的分类函数，使得：
  F(X) ≈ 1 (当Y = 1)； F(X) ≈ 0 (当Y = 0)

- 则可以称 我们找到了一个原样本集的一个‘似然’函数。
- 如果F是以最大概率符合样本数据，则F称为最大似然函数。

机器学习的一般步骤

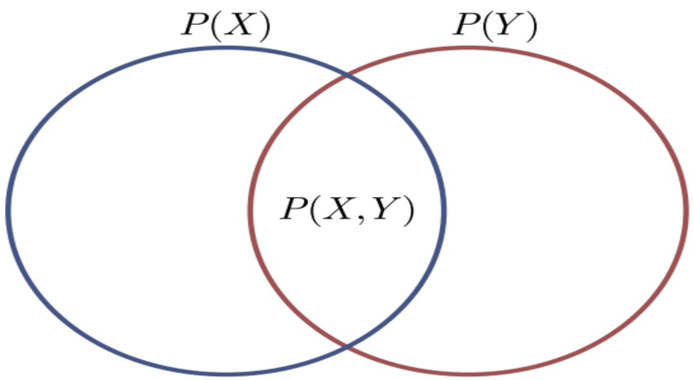
- 数据预处理、特征工程
  - 数据清洗
  - Na平滑，拉普拉斯平滑/降噪
  - 特征降维与均衡（embedding），隐含语义平滑
- 模型选择、超参数设计
- 模型学习与评测（可视化）

贝叶斯分类与有监督学习

- 概率模型
- 朴素贝叶斯分类
- 有监督学习

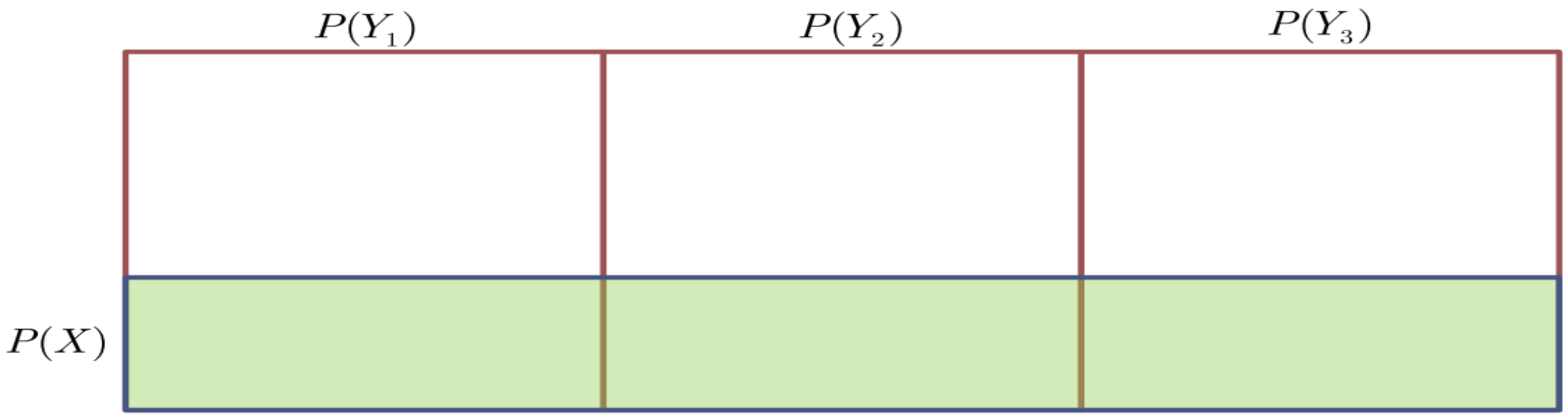
联合概率与条件概率

$$P(X \mid Y) = \frac{P(X,Y)}{P(Y)}$$



全概率公式

$$P(X) = \sum_{Y} P(X \mid Y)P(Y)$$



贝叶斯公式 与 贝叶斯推断

由：P（ylx）* P（x）= P（xly）* P（y）

可以导出：

P（ylx） = P（xly）* P（y） / P（x）

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)} = \frac{P(X \mid Y)P(Y)}{\sum_y P(X \mid y)P(y)}$$

贝叶斯概念：

想预测Y在未来特定要素下的表现，

只需了解过往Y情况下要各素的分布

贝叶斯概念：

先验概率乘经过条件似然进行修

饰，

得到带约束条件的后验概率

举个例子：

- 如果小A精神好，80%可能会起来跑步。
- 小A如果精神不好，40%可能会起来跑步
- 总体观察小A精神好的概率为60%

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)} = \frac{P(X \mid Y)P(Y)}{\sum_y P(X \mid y)P(y)}$$

目前看到小A正在跑步（看书，听音乐，打游戏 ...）

问：小A同学此时精神好的概率?

$$P(y1|x) = \frac{P(x|y1) * P(y1)}{P(x|y1) + P(x|y2)} = \frac{0.8 * 0.6}{0.8 * 0.6 + 0.4 * 0.4} = \frac{48}{64}$$

机器学习能做什么？

- 根据历史信息统计分析出 **事件-现象** 之间的概率关系

  —— 学习

- 根据目前观测到的 **现象** 对未发生事件的概率给出判断

  —— 预测

常见的分类问题描述

- 输入x是一个d维特征组成的向量R^d
- 模型F（x）的输出为一个k分类的唯一分类（one hot）的向量

$$\mathbb{R}^d \longrightarrow \{1, \ldots, k\}$$

朴素贝叶斯分类器（Naive Bayes Classifier）

- 基于贝叶斯推断方案： 先验概率 * 似然 → 后验概率
- **假定**特征之间相互独立：

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

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

- 模型实际返回值：最大后验概率（MAP）

模型参数估计（以词袋子特征为例）

- 模型所需的参数有 $P(y)$, $P(x_i \mid y)$.
- 最大似然估计:

$$\hat{P}(y) = \frac{|N(y)|}{Total}$$

$$\hat{P}(x_i \mid y) = \frac{n_{x_i,y}}{n_y}$$

- 问题?
  - 概率为 0 的情况. 若类 1 中出现词 x, 类 2 中没有.
  - 则 $P(\mathrm{x}|2) = 0$. 一个含有 x 的词永远无法被分入类 2.
  - 这是我们不希望看到的.

平滑 – 置信度 – 先验分布（伪计数）

平滑 (smoothing)

- 拉普拉斯（+1）平滑:

$$\hat{P}(y) = \frac{|N(y)|}{Total}$$

$$\hat{P}(x_i \mid y) = \frac{n_{x_i,y} + 1}{n_y + |V|}$$

- 带系数:

$$\hat{P}(x_i \mid y) = \frac{n_{x_i,y} + \alpha}{n_y + \alpha|V|}$$

隐含语义平滑?

- 通过矩阵分解与恢复得到平滑后的训练集
- 用平滑后的训练集进行训练

- 理论上属于高斯平滑

分类模型评价指标

➢混淆矩阵

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

recall = TP/(TP + FN)

precision = TP/(TP+FP)

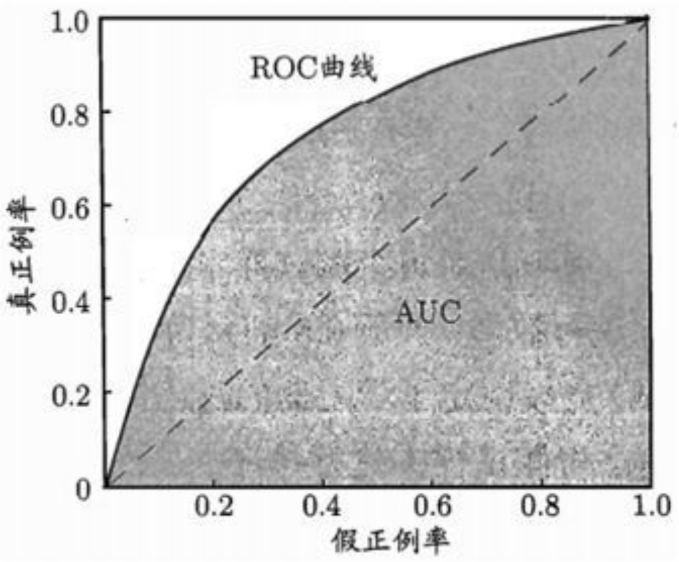
F1=2*recall*precision/(precision+recall)

分类模型评价指标

真正例率TPR=TP/(TP+FN)

假正例率FPR=FP/(TN+FP)

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN



文本分类的例子：

```
documents = [(list(movie_reviews.words(fileid)), category)
 for category in movie_reviews.categories()
 for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
train_set, test_set = featuresets[500:], featuresets[:500]  # 分离训练集、测试集
```

```
def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in
document_words)
    return features  # 词袋子
featuresets = [(document_features(d), c) for (d,c) in documents]  # 词袋子特征，文本类标 集合
train_set, test_set = featuresets[500:], featuresets[:500]       # 分离训练集、测试集
```

```
classifier = nltk.NaiveBayesClassifier.train(train_set)
print(nltk.classify.accuracy(classifier, test_set))
```

查看最有用的特征：

```
>>> classifier.show_most_informative_features(5)
Most Informative Features
         contains(segal) = True              neg : pos    =     11.3 : 1.0
    contains(outstanding) = True             pos : neg    =      8.6 : 1.0
         contains(wasted) = True             neg : pos    =      7.3 : 1.0
          contains(mulan) = True             pos : neg    =      7.2 : 1.0
    contains(wonderfully) = True             pos : neg    =      6.3 : 1.0
```

针对连续特征的 GAUSSIAN NAIVE BAYES

- 特征是实数量
- 服从高斯分布
- 假设特征之间独立

看一个鸢尾花数据集：

```
import seaborn as sns
iris = sns.load_dataset(’iris’)
print(iris.head(n = 3))
ir = iris.groupby(’species’)
ir.head(n = 2)
```

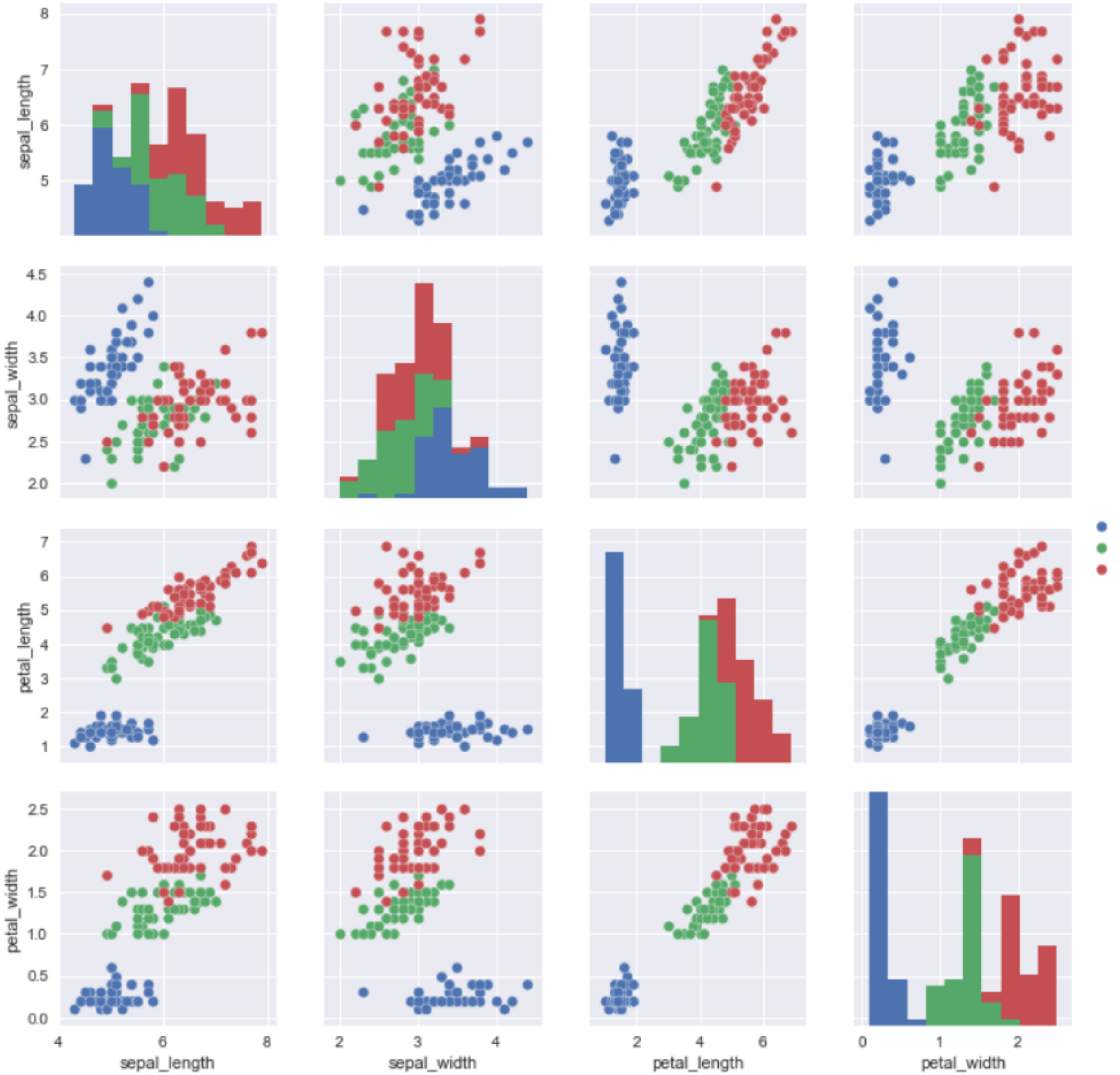
4个特征：花萼长宽，花瓣长宽

种属

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
50	7.0	3.2	4.7	1.4	versicolor
51	6.4	3.2	4.5	1.5	versicolor
100	6.3	3.3	6.0	2.5	virginica
101	5.8	2.7	5.1	1.9	virginica

鸢尾花特征高维数据可视化（数据维度两两组合）

```
%matplotlib inline
import seaborn as sns # ; sns.set
sns.pairplot(iris, hue='species',
```



对角线元素显示其他三个维度的取值在当前维度下的分布

非角线元素显示当前维度与另一个维度展开的二维平面上样本数据的分布情况

特征之间并不独立，每个特征在数据集上的分布也不均匀

Classification with Gaussian Naïve Bayes

- Possibility one: Disregard correlation —> Naïve
  - For each feature:
    - Calculate sample mean μ and sample standard deviation σ
    - Use these as estimators of the population mean and deviation
  - For a given feature value *x*, calculate the probability density assuming that x is in a category c
    - $P(x \mid c) \sim \mathcal{N}(\mu_c, \sigma_c)$

Classification with Gaussian Naïve Bayes

- Estimate the probability for observation $(x_1, x_2, \ldots, x_n)$ as the product of the densities

$$P((x_1, \ldots, x_n)\,|\,c_j) \sim \mathcal{N}(x_1, \sigma_{1,c_j}, \mu_{1,c_j}) \cdot \ldots \cdot \mathcal{N}(x_1, \sigma_{n,c_j}, \mu_{1,c_j})$$

$\boxed{x_n}$

- Then use Bayes formula to invert the conditional probabilities
  - This means estimating the prevalence of the categories

- $$P(c_j\,|\,(x_1, \ldots, x_n)) = \frac{P((x_1, \ldots, x_n)\,|\,c_j)P(c_j)}{P((x_1, \ldots, x_n))}$$

手动实现一个G-NB：

```python
class Gaussian(object):

    def __init__(self):
        # 这里可以设置平滑或先验参数
        pass

    def fit(self,X_train,Y_train):
        self._data_with_label = X_train.copy()
        self._Y_train = Y_train.copy()
        self._data_with_label['label'] = Y_train[0] # 有监督数据

        self._mean_mat = self._data_with_label.groupby("label").mean() # 每个类别的特征分布 均值
        self._var_mat = self._data_with_label.groupby("label").var()   # 方差
        self.prior_rate = self.__Priori() # 统计类别先验
        return self

    #Priori probability
    def __Priori(self):
        labels = self._Y_train[0].value_counts().sort_index()      # label计数
        prior_rate = np.array([ i /sum(labels) for i in labels])  # label比例
        return prior_rate
```

```python
#Priori probability
def __Priori(self):
    labels = self._Y_train[0].value_counts().sort_index()      # label计数
    prior_rate = np.array([ i /sum(labels) for i in labels])  # label比例
    return prior_rate

def predict(self,X_test): # 模型预测
    pred = [self.__Condition_formula(self.mean_mat,self.var_mat,row ) * self.prior_rate for row in X_test.values ]
    class_result = np.argmax(pred, axis=1)  # 返回 argmax
    return class_result

#Gaussian Bayes condition formula
def __Condition_formula(self,mu,sigma2,row):
    P_mat = 1/np.sqrt(2*math.pi*sigma2) * np.exp(-(row-mu)**2/(2*sigma2)) # 高斯函数计算先验
    P_mat = pd.DataFrame(P_mat).prod(axis=1) # 返回一列的乘积
    return P_mat
```

```python
# 导入数据集测试一下:
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris = load_iris()
iris.target = pd.DataFrame(iris.target)
iris.data = pd.DataFrame(iris.data)

#  train_size, test_size None, it will be 0.25 by default
X_train, X_test, Y_train, Y_test = train_test_split(iris.data, iris.target, test_size=0.4, random_state=1)

np.set_printoptions(suppress=True) # 不用科学计数法的形式输出
```

```python
X_train
```

	0	1	2	3
11	4.8	3.4	1.6	0.2
113	5.7	2.5	5.0	2.0
123	6.3	2.7	4.9	1.8
12	4.8	3.0	1.4	0.1
2	4.7	3.2	1.3	0.2

对比直接调包的效果：

```
NB = Gaussian() # 使用自定义类
NB.fit(X_train, y_train)
y_train_NB = NB.predict(X_train)
y_test_NB = NB.predict(X_test)
print("Use custom Gaussian Naive Bayes algorithm\naccuracy on train
    accuracy_score(y_test,y_test_NB))

print("--- %s seconds ---" % (time.time() - start_time))
```

```
Use custom Gaussian Naive Bayes algorithm
accuracy on train set:  0.9555555555555556
accuracy on test set:  0.95
--- 0.2263889312744140б seconds ---
```

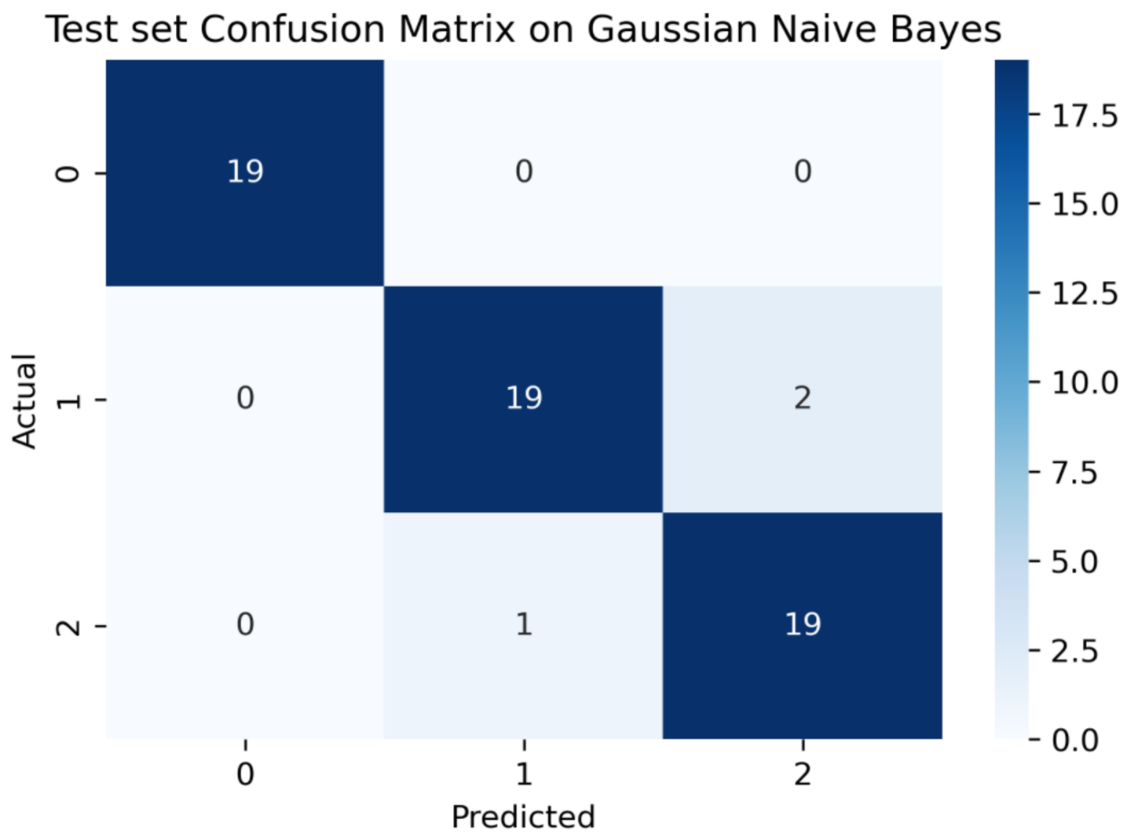
```
from sklearn.naive_bayes import GaussianNB
start_time = time.time()
NB2 = GaussianNB()
NB2.fit(X_train, y_train.values.ravel())
y_train_NB2 = NB2.predict(X_train)
y_test_NB2 = NB2.predict(X_test)
print("Use sklearn Gaussian Naive Bayes algorithm\naccuracy on train set:
    accuracy_score(y_test,y_test_NB2))

print("--- %s seconds ---" % (time.time() - start_time))
```

```
Use sklearn Gaussian Naive Bayes algorithm
accuracy on train set:  0.9555555555555556
accuracy on test set:  0.95
--- 0.004966259002685547 seconds ---
```

显示混淆矩阵

```
import matplotlib.pyplot as plt
import seaborn as sns
con_matrix = pd.crosstab(pd.Series(y_test.values.flatten(), name='Actual' ),pd.Series(y_test_NB, name='Predicted'))
plt.title("Test set Confusion Matrix on Gaussian Naive Bayes")
sns.heatmap(con_matrix, cmap="Blues", annot=True, fmt='g')
plt.show()
```



商品评价分类（例子）

分类与回归

- 对于一个样本集，如果能找到一个合理的分类函数，使得：

  F(X) ≈ 1 (当Y = 1); F(X) ≈ 0 (当Y = 0)

- 则可以称 我们找到了一个原样本集的一个‘似然’函数
- 如果F是以最大概率（最均方差损失）符合样本数据，则F称为最大似然函数

贝叶斯公式的一个变形

$$P(spam|text) = \frac{P(text|spam)P(spam)}{P(text|spam)P(spam) + P(text|nonspam)P(nonspam)}$$

$$= \frac{1}{1 + \exp\{-(\alpha_1(text) - \alpha_0(text))\}}$$

其中$\alpha_1(text) = \log(P(text|spam)P(spam))$

以及$\alpha_0(text) = \log(P(text|nonspam)P(nonspam))$

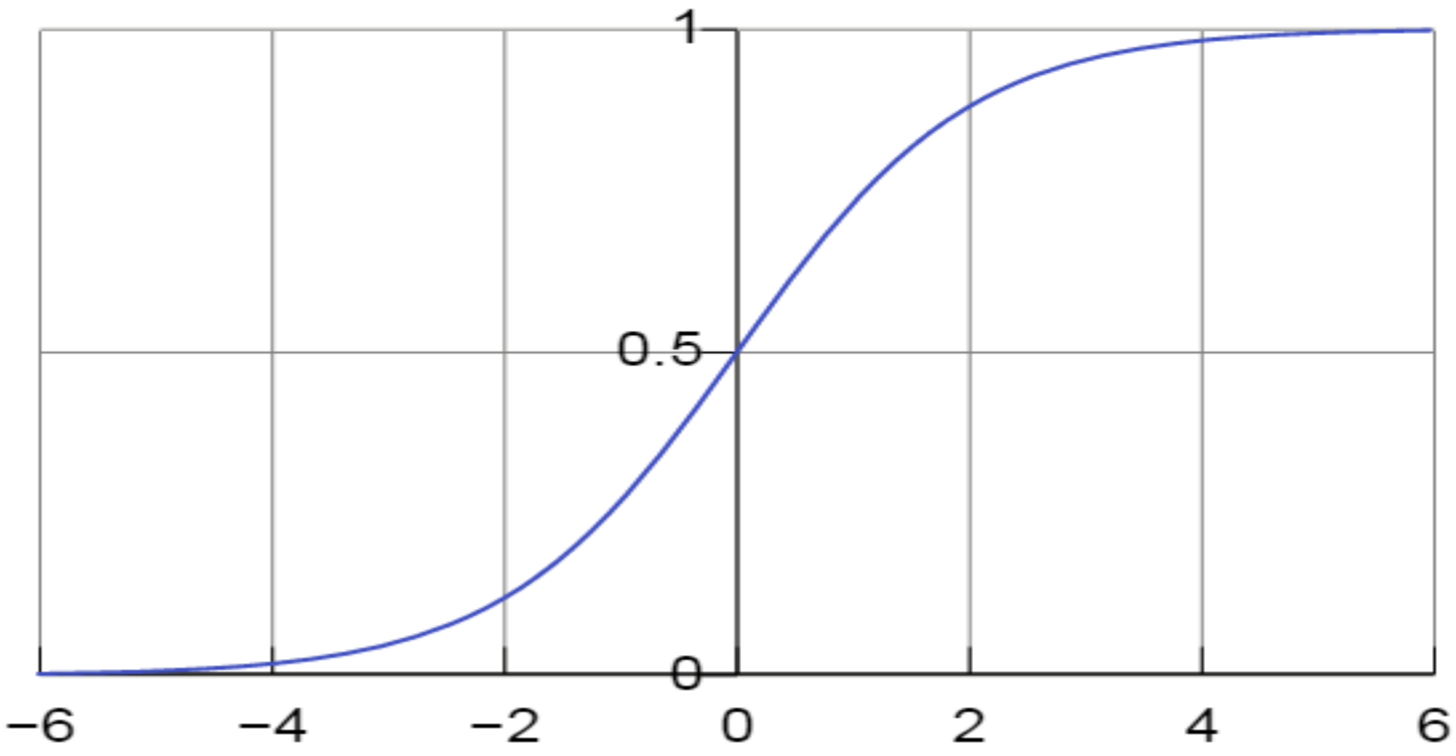
$$= \sigma(\alpha(x))$$

这里$\sigma(x) = \frac{1}{1+\exp(-x)}$, $\alpha(x) = \alpha_1(x) - \alpha_0(x)$

Sigmoid函数

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

将线性回归值变换到(0, 1)，将其理解为x对应的y为1的概率



因此邮件分类问题转化为回归问题

$$\alpha(text) = \log P(text|spam) + \log P(spam) - \log P(text|nonspam) - \log P(nonspam)$$

$$= x_1 \log P(w_1|spam) + \ldots + x_{|V|} \log P(w_{|V|}|spam) + \log P(spam)$$

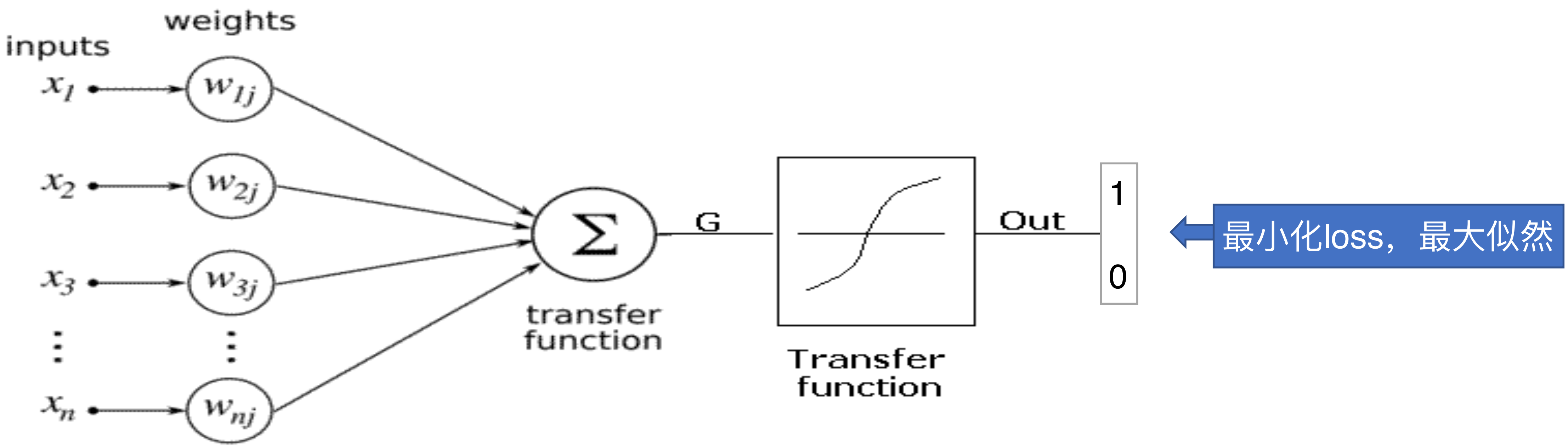
$-\ldots$(对应的nonspam的项)

(x_i表示词典中第i个词在text中出现的次数)

$$= k_0 + x_1 k_1 + x_2 k_2 + \ldots x_{|V|} k_{|V|}$$

找一组**K**，最大化**α**

单元神经元网络模型



weights
inputs
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
transfer function
G
Transfer function
Out
1
0
最小化loss，最大似然

Logistic Regression分类器

对于Logistic Regression($y^{(i)} \in \{0,1\}$表示属于哪一类)，一个样本的似然是：

$$P(y^{(i)}|\mathbf{x}^{(i)},\mathbf{k}) = \begin{cases} \sigma(\mathbf{k}^\top\mathbf{x}) & \text{if } y^{(i)} = 1 \\ 1-\sigma(\mathbf{k}^\top\mathbf{x}) & \text{if } y^{(i)} = 0 \end{cases}$$
$$= \sigma(\mathbf{k}^\top\mathbf{x}^{(i)})^{y^{(i)}} \cdot (1-\sigma(\mathbf{k}^\top\mathbf{x}^{(i)}))^{1-y^{(i)}}$$

最大后验概率

整个数据集的似然则是：

$$\hat{\mathbf{k}} = \arg\max_{\mathbf{k}} \prod_{i=1}^{N} \left\{ \sigma(\mathbf{k}^\top\mathbf{x}^{(i)})^{y^{(i)}} \cdot (1-\sigma(\mathbf{k}^\top\mathbf{x}^{(i)}))^{1-y^{(i)}} \right\}$$
$$= \arg\max_{\mathbf{k}} \sum_{i=1}^{N} \left\{ y^{(i)} \log\sigma(\mathbf{k}^\top\mathbf{x}^{(i)}) + (1-y^{(i)})\log(1-\sigma(\mathbf{k}^\top\mathbf{x}^{(i)})) \right\}$$

所以我们想要找一个k，最大化上面的这个函数，这就是一个求函数最大值的问题了

Logistic Regression

也就是说，朴素贝叶斯分类器的后验概率是这样一个形式：

$$\sigma(\sum_{i=1}^{K} k_i x_i), \quad (x_0 = 0)$$

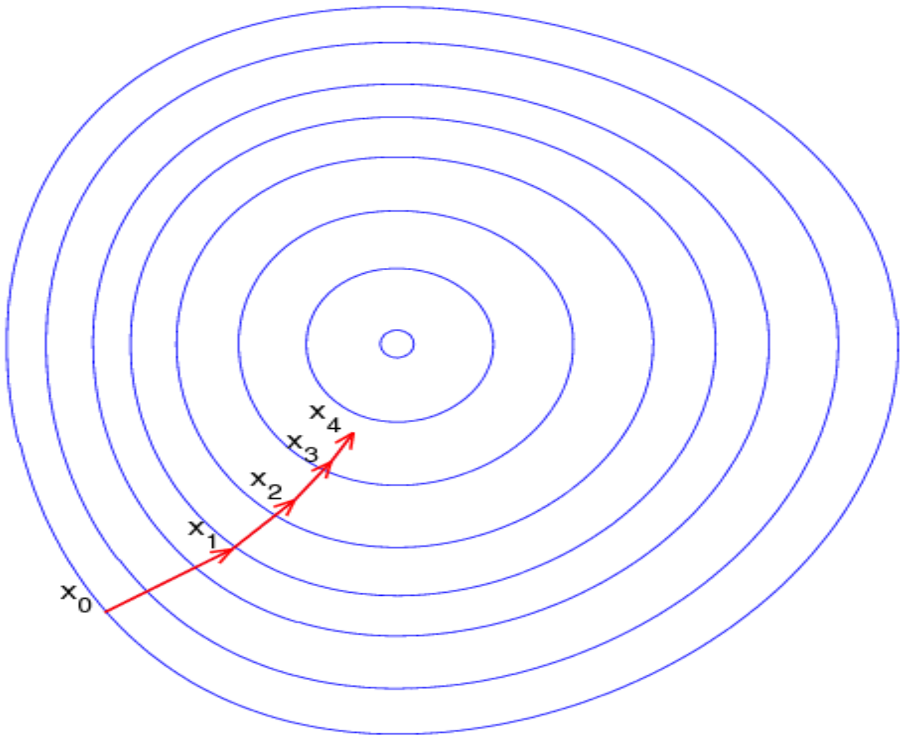
事实上，还有很多模型的后验概率也都是这样的形式，所以我们不妨想办法直接求出合适的k_i，而不去使用贝叶斯公式。

即是说，我们直接假设：

$$P(y=1|x_1,...,x_K) = \sigma(k_0 + k_1 x_1 + ... + k_K x_k)$$
$$P(y=0|x_1,...,x_K) = 1 - P(y=1|x_1,...,x_K)$$

然后根据我们手里的样本集，估计出k的一个合理的取值。

多维参数空间随机梯度下降法



计算损失函数的梯度函数：

$$C(k)=\sum_{i=1}^{n}-y^{(i)}\log(g(k^Tx^{(i)}))-(1-y^{(i)})\log(1-g(k^Tx^{(i)}))$$

$$\frac{\partial C(k)}{\partial k_j}=\sum_{i=1}^{n}-y^{(i)}\frac{\frac{\partial g(k^Tx^{(i)})}{\partial k_j}}{g(k^Tx^{(i)})}-(1-y^{(i)})\frac{\frac{\partial(1-g(k^Tx^{(i)}))}{\partial k_j}}{1-g(k^Tx^{(i)})}$$

sigmoid函数这样一个性质：$g'(x)=g(x)(1-g(x))$

$$\frac{\partial C(k)}{\partial k_j}=\sum_{i=1}^{n}\left(-y^{(i)}\frac{g(k^Tx^{(i)})(1-g(k^Tx^{(i)}))x_j^{(i)}}{g(k^Tx^{(i)})}-(1-y^{(i)})\frac{-g(k^Tx^{(i)})(1-g(k^Tx^{(i)}))x_j^{(i)}}{1-g(k^Tx^{(i)})}\right)$$

$$\frac{\partial C(k)}{\partial k_j}=\sum_{i=1}^{n}-y^{(i)}(1-g(k^Tx^{(i)}))x_j^{(i)}-(1-y^{(i)})(0-g(k^Tx^{(i)}))x_j^{(i)}$$

$$\frac{\partial C(k)}{\partial k_j}=\sum_{i=1}^{n}(g(k^Tx^{(i)})-y^{(i)})x_j^{(i)}$$

Logistic Regression训练流程：

输入：样本集；输出：参数k的极大似然估计

1. 随机初始化$\mathbf{k}$
2. 计算梯度$\mathbf{g}$，满足$\underbrace{\mathbf{g}_j = \sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{k}^\top \mathbf{x}^{(i)}))x_j^{(i)}}_{\text{反向梯度}}$
3. $\mathbf{k} = \mathbf{k} + \alpha\mathbf{g}$ 梯度下降 α为学习率
4. 迭代上两步

Logistic Regression推断流程：

输入：一个y未知的x；输出：此x的y=1的概率

1. 求$P(y=1) = \sigma(\mathbf{k}^\top x)$

进一步的改进?