

智能人机交互与视觉感知综合设计

手写体识别实例

刘 奇

Python编程库



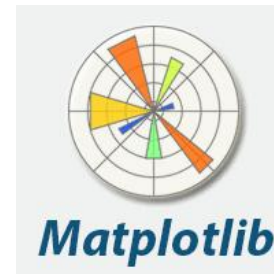
Scipy库



NumPy库



OpenCV 库



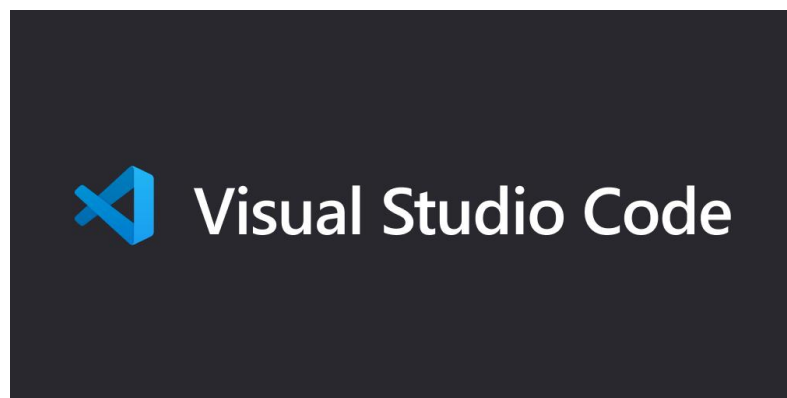
Matplotlib库



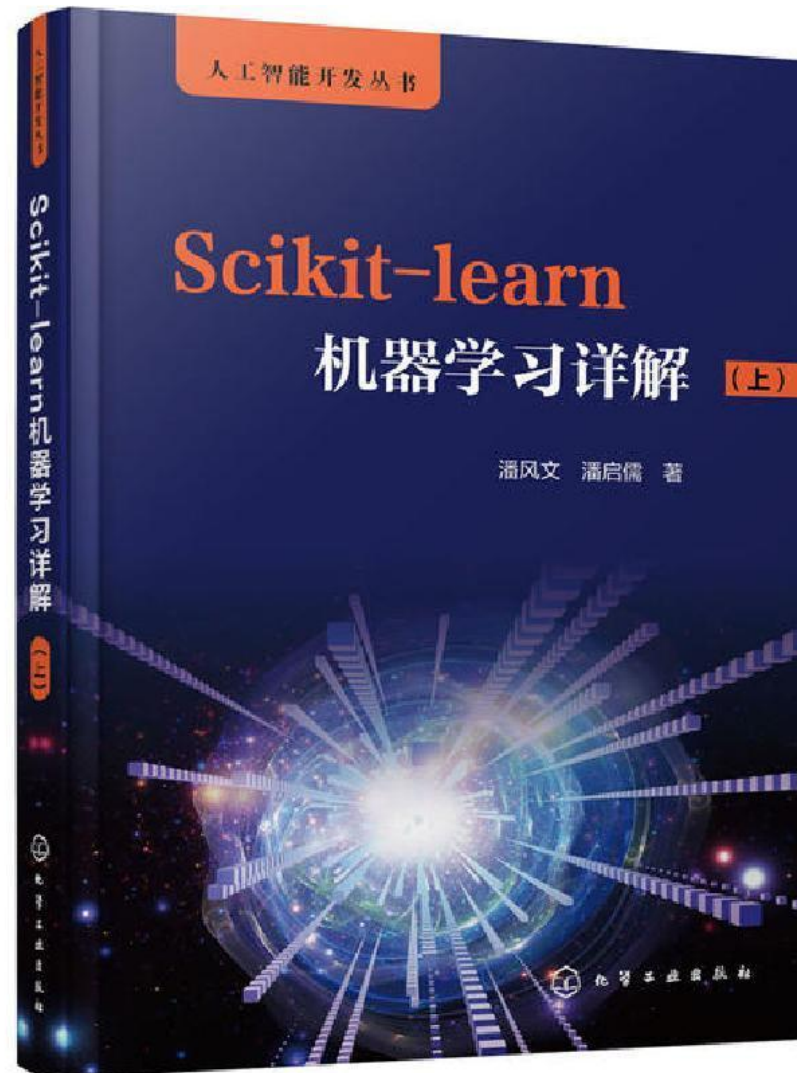
深度学习库



Python开发环境



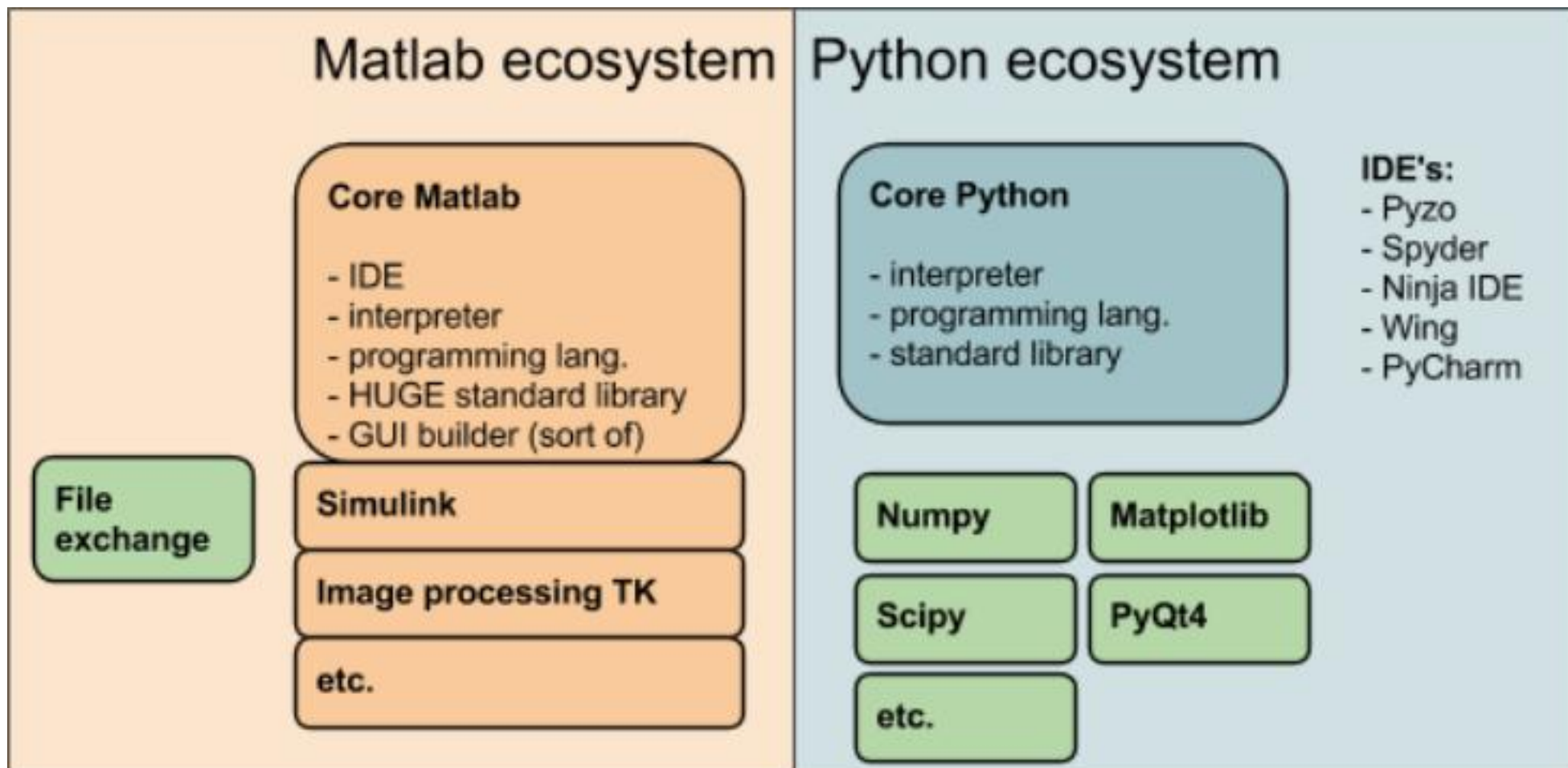
推荐书籍：Scikit-learn机器学习详解



编程语言



编程语言





Scipy库简介

- Scipy库是基于Python生态的一款开源数值计算、科学与工程应用的开源软件，包括常用的NumPy、pandas、matplotlib等库
- SciPy 是一个开源的 Python 算法库和数学工具包。
- Scipy 是基于 Numpy 的科学计算库，用于数学、科学、工程学等领域，很多有一些高阶抽象和物理模型需要使用 Scipy。
- SciPy 包含的模块有最优化、线性代数、积分、插值、特殊函数、快速傅里叶变换、信号处理和图像处理、常微分方程求解和其他科学与工程中常用的计算。
- <http://www.scipy.org/>



scikit-learn库简介

- scikit-learn库 Machine Learning in Python
- 依赖Python的NumPy, sciPy和matplotlib库
- 开源、可复用
- <http://scikit-learn.org/stable/>

	应用 (Applications)	算法 (Algorithm)
分类 (Classification)	异常检测, 图像识别, 等	KNN, SVM , etc.
聚类 (Clustering)	图像分割, 群体划分, 等	K-Means , 谱聚类, etc.
回归 (Regression)	价格预测, 趋势预测, 等	线性回归, SVR , etc.
降维 (Dimension Reduction)	可视化,	PCA , NMF , etc.

Scikit-learn库

https://scikit-learn.org/stable/user_guide.html

Scikit-learn（以前称为scikits.learn，也称为sklearn）是针对Python 编程语言的免费软件机器学习库。它具有各种分类，回归和聚类算法，包括支持向量机，随机森林，梯度提升，k均值和DBSCAN，并且旨在与Python数值科学库NumPy和SciPy联合使用。

- 常用的回归：线性、决策树、SVM、KNN；集成回归：随机森林、Adaboost、GradientBoosting、Bagging、ExtraTrees
- 常用的分类：线性、决策树、SVM、KNN，朴素贝叶斯；集成分类：随机森林、Adaboost、GradientBoosting、Bagging、ExtraTrees
- 常用聚类：k均值（K-means）、层次聚类（Hierarchical clustering）、DBSCAN
- 常用降维：LinearDiscriminantAnalysis、PCA



NumPy

- NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。
- NumPy 的前身 Numeric 最早是由 Jim Hugunin 与其它协作者共同开发，2005 年，Travis Oliphant 在 Numeric 中结合了另一个同性质的程序库 Numarray 的特色，并加入了其它扩展而开发了 NumPy。NumPy 为开放源代码并且由许多协作者共同维护开发。



NumPy库简介

- NumPy 是一个运行速度非常快的数学库，主要用于数组计算，包含：
 - 一个强大的N维数组对象 ndarray
 - 广播功能函数
 - 整合 C/C++/Fortran 代码的工具
 - 线性代数、傅里叶变换、随机数生成等功能



NumPy库简介

- NumPy 通常与 SciPy (Scientific Python) 和 Matplotlib (绘图库) 一起使用, 这种组合广泛用于替代 MatLab, 是一个强大的科学计算环境, 有助于我们通过 Python 学习数据科学或者机器学习。
- Matplotlib 是 Python 编程语言及其数值数学扩展包 NumPy 的可视化操作界面。



NumPy库的数据结构

- Narray 一种多维数组对象
- N-dimensional array, N维数组
- 一种由相同类型的元素组成的多维数组, 元素数量是事先指定好的
- 元素的数据类型由dtype (data-type) 对象来指定, 每个ndarray只有一种dtype类型
- 大小固定, 创建好数组时一旦指定好大小, 就不会再发生改变



ndarray属性

- `ndim` 维度数量
- `shape` 是一个表示各维度大小的元组，即数组的形状
- `dtype`，一个用于说明数组元素数据类型的对象
- `size`，元素总个数，即`shape`中各数组相乘

ndarray属性

```
In [21]: import numpy as np
```

```
In [22]: a = np.array([
    [
        [3.4, 5, 6, 8],
        [3, 2.4, 5, 7]
    ],
    [
        [2.3, 4, 5, 6],
        [0.9, 5, 6, 1]
    ],
    [
        [9, 6.7, 3, 2],
        [1, 3, 4, 5]
    ]
])
```

```
In [23]: a.ndim
```

ndarray数组维度数

```
Out[23]: 3
```

```
In [24]: a.dtype
```

ndarray数组元素数据类型

```
Out[24]: dtype('float64')
```

```
In [25]: a.shape
```

ndarray数组形状

```
Out[25]: (3L, 2L, 4L)
```

```
In [26]: a.size
```

ndarray数组元素总个数

```
Out[26]: 24
```

ndarray属性

```
In [21]: import numpy as np
```

```
In [22]: a = np.array([
    [
        [3.4, 5, 6, 8],
        [3, 2.4, 5, 7]
    ],
    [
        [2.3, 4, 5, 6],
        [0.9, 5, 6, 1]
    ],
    [
        [9, 6.7, 3, 2],
        [1, 3, 4, 5]
    ]
])
```

```
In [23]: a.ndim
```

ndarray数组维度数

```
Out[23]: 3
```

```
In [24]: a.dtype
```

ndarray数组元素数据类型

```
Out[24]: dtype('float64')
```

```
In [25]: a.shape
```

ndarray数组形状

```
Out[25]: (3L, 2L, 4L)
```

```
In [26]: a.size
```

ndarray数组元素总个数

```
Out[26]: 24
```


ndarray创建

- `array`函数：接收一个普通的Python序列，转成`ndarray`
- `zeros`函数：创建指定长度或形状的全零数组
- `ones`函数：创建指定长度或形状的全1数组
- `empty`函数：创建一个没有任何具体值的数组
- `arange`函数：通过指定开始值、终值和步长来创建一维数组，注意数组不包括终值

```
In [45]: np.zeros((3,4))
```

```
Out[45]: array([[ 0.,  0.,  0.,  0.],  
               [ 0.,  0.,  0.,  0.],  
               [ 0.,  0.,  0.,  0.]])
```

```
In [46]: np.ones((4, 6))
```

```
Out[46]: array([[ 1.,  1.,  1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.,  1.,  1.]])
```

```
In [48]: np.empty((2,3,4))
```

```
Out[48]: 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.]])
```

ndarray创建

```
In [49]: np.arange(20)
```

```
Out[49]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```


```
In [50]: np.arange(0, 20, 1)
```

```
Out[50]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
In [51]: np.arange(0, 20, 2)
```

```
Out[51]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

reshape()函数可以改变数组的形状，
但是注意
元素总个数不能改变



```
In [54]: np.arange(0, 12).reshape(3, 4)
```

```
Out[54]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])
```

NumPy数据类型

- 创建NumPy数组时可以通过dtype属性显式指定数据类型，如果不指定，NumPy会自己推断出合适的数据类型，所以一般无需显式指定
- astype方法，可以转换数组的元素数据类型，得到一个新数组
- 数值型dtype的命名方式：一个类型名（比如int、float），后面接着一个用于表示各元素位长的数字

NumPy数据类型

```
In [26]: ndarray02 = np.array([1, 2, 3, 4])
```

```
In [27]: ndarray02.dtype
```


```
Out[27]: dtype('int32')
```

```
In [32]: ndarray03 = ndarray02.astype(float)
```

```
In [33]: ndarray03.dtype
```

```
Out[33]: dtype('float64')
```

类型转换，注意要真正能转才行



```
In [36]: d = np.array(["Python", "Scala", "Java", "C#"])
```

```
In [37]: d
```

```
Out[37]: array(['Python', 'Scala', 'Java', 'C#'],  
              dtype='<S6')
```

```
In [38]: d.dtype
```

```
Out[38]: dtype('<S6')
```

```
In [39]: e = np.array(["Python", "Scala", "Java", "C++"], dtype=np.string_)
```

```
In [40]: e
```

```
Out[40]: array(['Python', 'Scala', 'Java', 'C++'],  
              dtype='<S6')
```

```
In [41]: e = np.array(["Python", "Scala", "Java", "C++"], dtype='<S8')
```

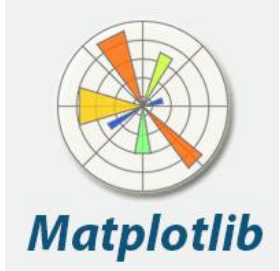
```
In [42]: e
```

```
Out[42]: array(['Python', 'Scala', 'Java', 'C++'],  
              dtype='<S8')
```



Matplotlib库简介

- Matplotlib 是 Python 的绘图库，它能让使用者很轻松地将数据图形化，并且提供多样化的输出格式。
- Matplotlib 可以用来绘制各种静态，动态，交互式的图表。
- Matplotlib 是一个非常强大的 Python 画图工具，我们可以使用该工具将很多数据通过图表的形式更直观的呈现出来。
- Matplotlib 可以绘制线图、散点图、等高线图、条形图、柱状图、3D 图形、甚至是图形动画等等。



Matplotlib库简介

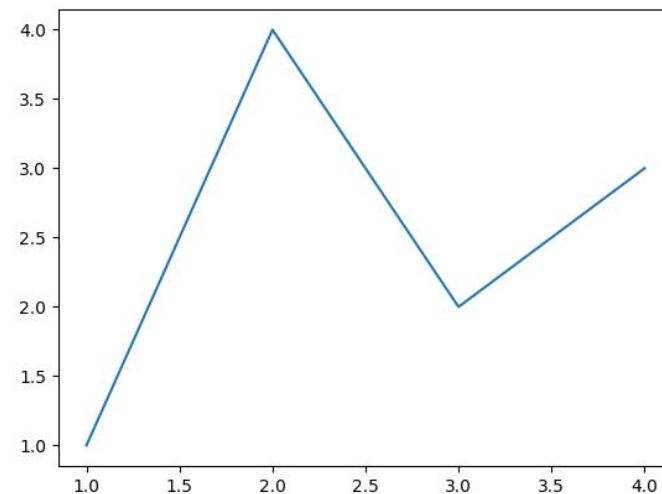
- Matplotlib是python最著名的绘图库，它提供了一整套和matlab相似的命令API，适合交互式制图。方便作为绘图控件，嵌入GUI应用程序中。
- 网址：<http://matplotlib.org/>
- 学习方式：从官网tutorials入手学习：
<https://matplotlib.org/stable/tutorials/index.html>
- <http://matplotlib.org/gallery.html> 各种图示案例

Figure对象

- Matplotlib的图像都位于Figure对象中，Figure对象下创建一个或多个subplot对象（即axes）用于绘制图表

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

```
fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]); # Plot some data on the axes.
```





OpenCV库简介

- OpenCV 是 Intel 开源计算机视觉库。它由一系列 C 函数和少量 C++ 类构成，实现了图像处理和计算机视觉方面的很多通用算法。
- 同时提供了Python、Ruby、MATLAB等语言的接口。
- 网址 https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html

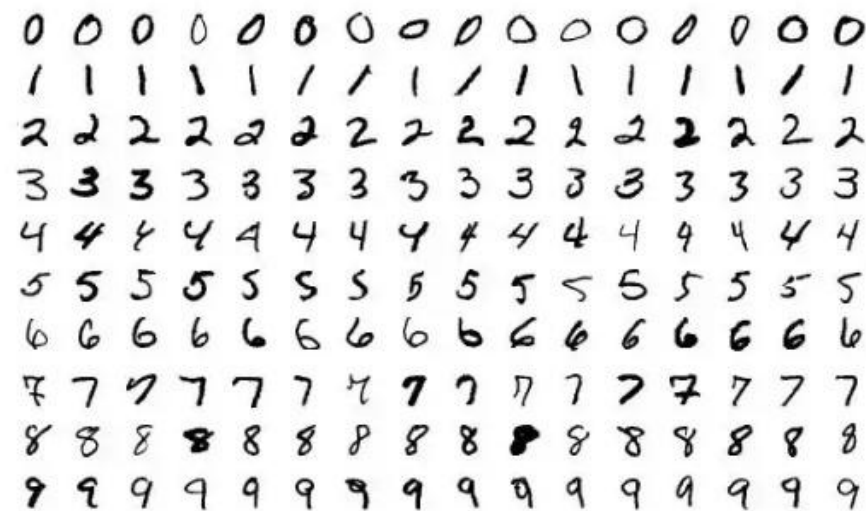
手写体识别

- 手写识别是常见的图像任务。计算机通过体片来出中的字，与印刷体不同的是，不同人的手写体风格迥异，大小不一造成了计算机对手写识别任务的一些困难。
- 数字手写体识别由于其有限的类（0~9 共10 个数字）成为了相对简单的手写识别任务。DBRHD 和MNIST 是常用的两个数字手写识别数据集。

手写体识别

MNIST 是一个包含数字 0~9 的手写体图片数据集，图片已归一化为以字为中心的 28×28 规格的图片。MNIST 由训练集与测试两个部分组成，各规模如下：

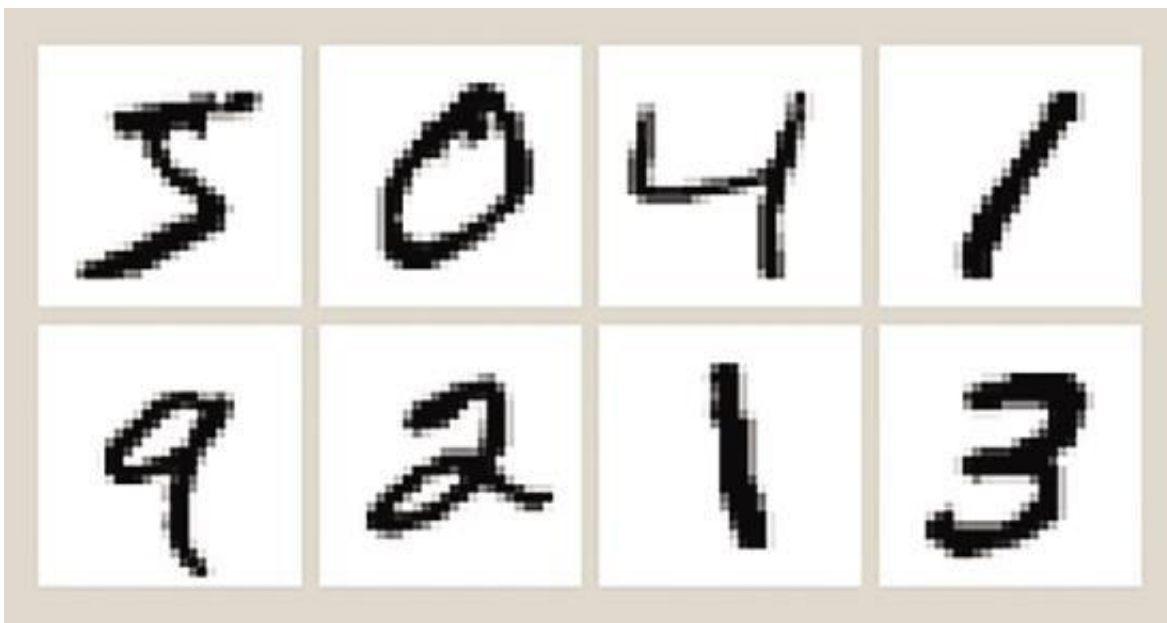
- 训练集： 60,000 个手写体图片及对应标签
- 测试集： 10,000 个



手写体识别

MNIST 数据集的手写字样例：

- MNIST 数据集中的每一个图片由 28×28 个像素点组成
- 每个像素点的值区间为 0~255，0表示白色， 255



手写体识别

已有许多模型在MNIST或DBRHD数据集上进行了实验，有些模型对数据集进行了偏斜矫正，甚至在数据集上进行了人为的扭曲、偏移、缩放及失真等操作以获取更加多样性的样本，使得模型更具有泛化性。

- 常用于数字手写体的分类器：

- 1) 线性分类器

- 2) K最近邻分类器

- 3) Boosted Stumps

- 4) 非线性分类器

- 5) SVM

- 6) 多层感知器

- 7) 卷积神经网络

- 后续任务：利用全连接的神经网络实现手写识别的任务

Training kNN, Gaussian Bayes, and SVM models to classify MNIST

- k-nearest neighbors (KNN) classifier

 - Squared Euclidean distance

 - Computing the nearest neighbors

 - Evaluating the performance of the classifier

- Bayes classifier (Gaussian generative model)

 - Training the generative model – computing the MLE of the Gaussian parameters

 - Computing the posterior probabilities to make predictions on test data and model evaluation

- SVM classifier

K-近邻算法 KNN

Cover和Hart在1968年提出了最初的邻近算法，是最简单的机器学习算法之一。

概念原理：

邻近算法(Nearest Neighbor)的思想实际上十分简单，就是将测试图片和储存起来的训练集一一进行相似度计算，计算出最相近的图片，这张图片的标签便是赋给测试图片的分类标签。

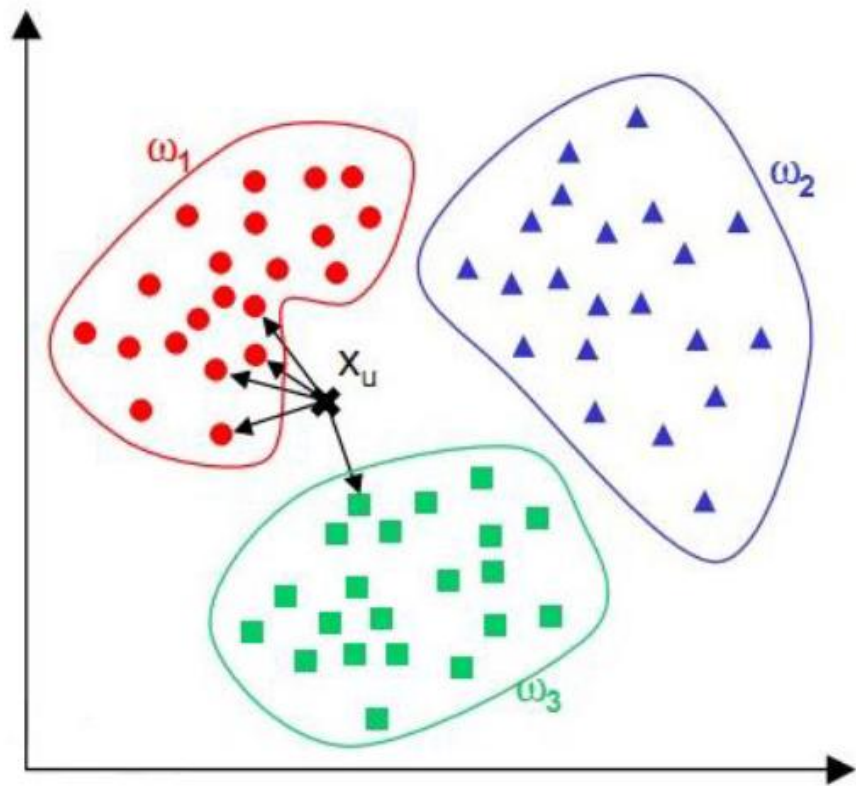
如何比较两组数据之间的相似长度：

算法分析

- 最常用的两种方式：
- ①L1距离(Manhattan distance)
 - ②L2距离(Euclidean distance)

K-近邻算法 KNN

KNN：通过计算待分类数据点，与已有数据集中的所有数据点的距离。取距离最小的前K个点，根据“少数服从多数”的原则，将这个数据点划分为出现次数最多的那个类别。



K-近邻算法 KNN

概念原理

邻近算法(Nearest Neighbor)的思想实际上十分简单，就是将测试图片和储存起来的训练集一一进行相似度计算，计算出最相近的图片，这张图片的标签便是赋给测试图片的分类标签。

那么如何比较两组数据之间的相似长度呢？

算法分析

最常用的两种方式：①L1距离(Manhattan distance)

②L2距离(Euclidean distance)

K-近邻算法 KNN

算法详述

步骤:

为了判断未知实例的类别，以所有已知类别的实例作为参照

选择参数K

计算未知实例与所有已知实例的距离

选择最近K个已知实例

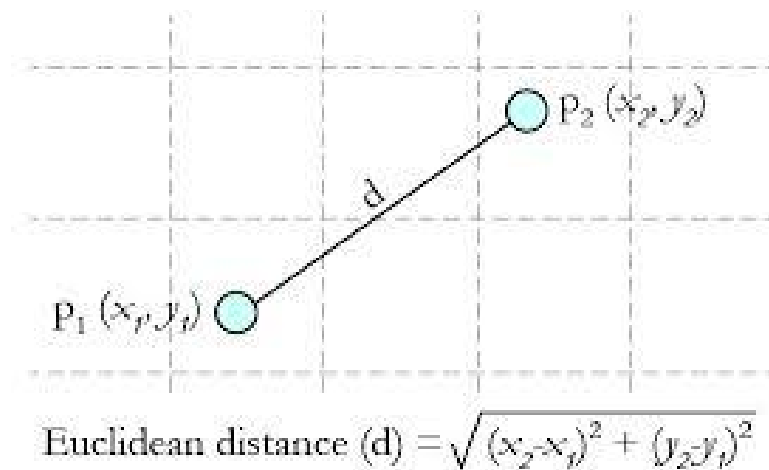
根据少数服从多数的投票法则(majority-voting)，让未知实例归类为K个最邻近样本中最多数的类别

细节:

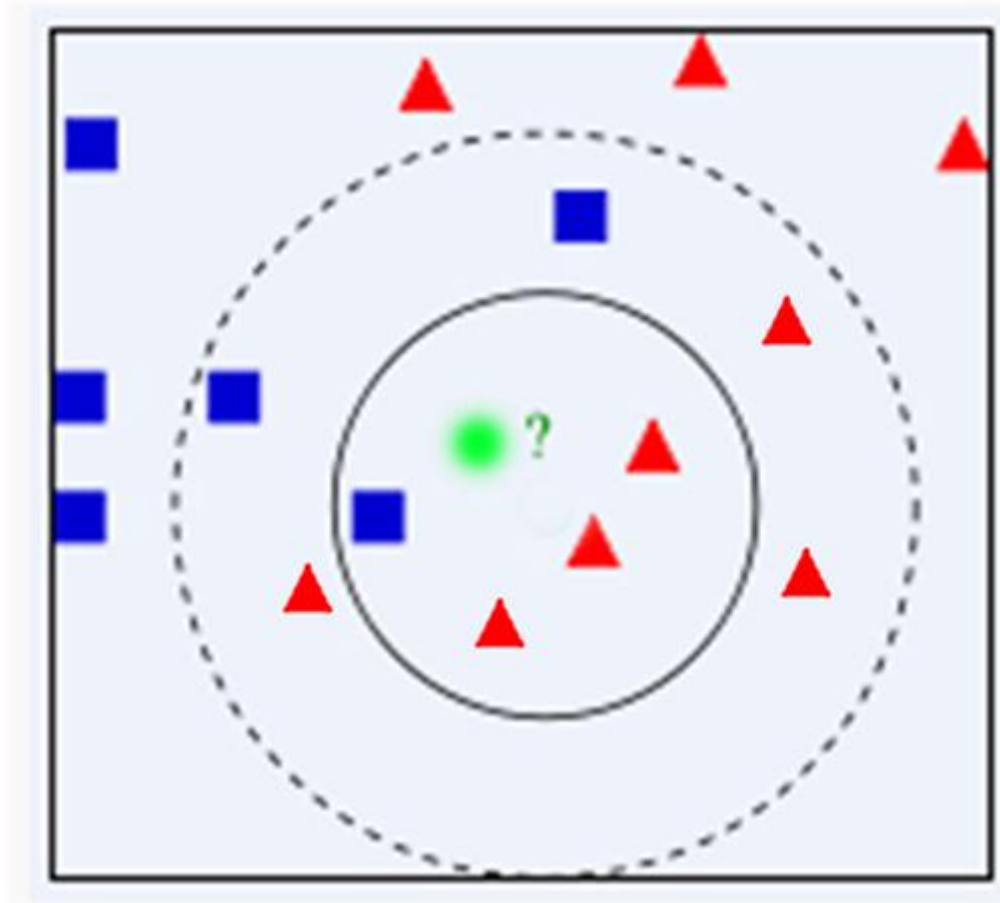
关于K

关于距离的衡量方法:

Euclidean Distance 定义



K-近邻算法 KNN



K-近邻算法 KNN

搜索方法

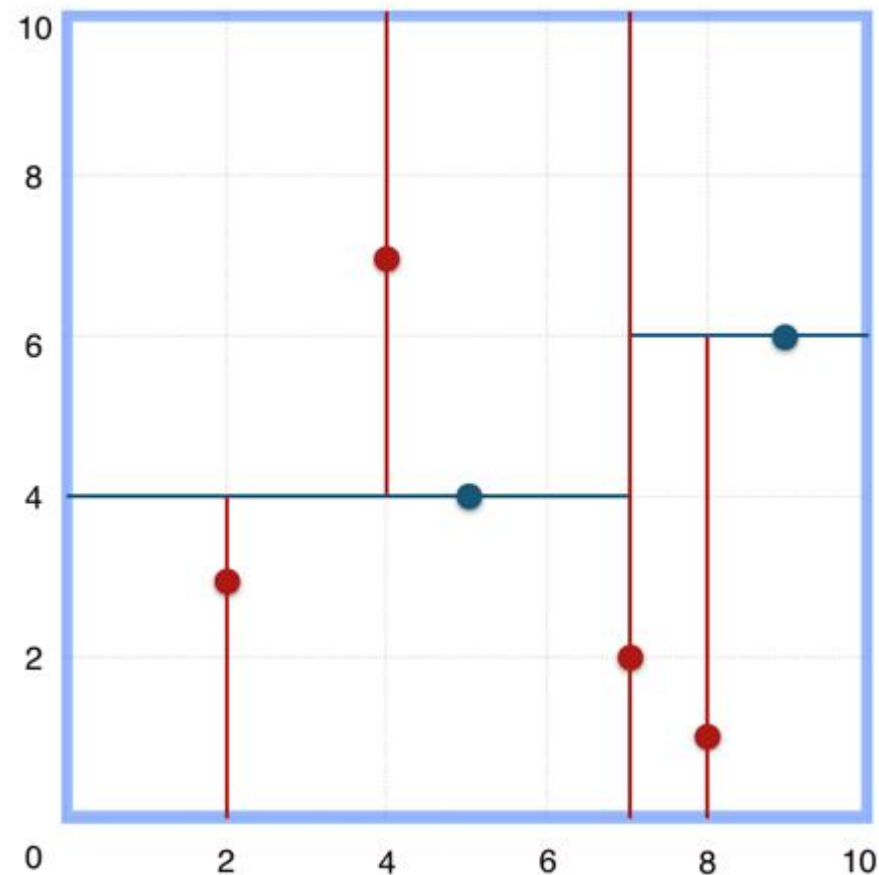
最近邻检索：

K-D tree (k-dimensional tree)

Ball-tree 高维数据上划分的高效性

k-d tree即k-dimensional tree，常用来作空间划分及近邻搜索，是二叉空间划分树的一个特例。通常，对于维度为k，数据点数为N的数据集，k-d tree适用于 $N \gg 2k$ 的情形。

k-d tree是每个节点均为k维数值点的二叉树，其上的每个节点代表一个超平面，该超平面垂直于当前划分维度的坐标轴，并在该维度上将空间划分为两部分，一部分在其左子树，另一部分在其右子树。即若当前节点的划分维度为d，其左子树上所有点在d维的坐标值均小于当前值，右子树上所有点在d维的坐标值均大于等于当前值，本定义对其任意子节点均成立。



K-近邻算法 KNN

搜索方法

最近邻检索：

K-D tree (k-dimensional tree)

Ball-tree 高维数据上划分的高效性

为了改进KDtree的二叉树树形结构，并且沿着笛卡尔坐标进行划分的低效率，Ball tree将在一系列嵌套的超球体上分割数据。也就是说：使用超球面而不是超矩形划分区域。虽然在构建数据结构的花费上大过于KDtree，但是在高维甚至很高维的数据上都表现的很高效。

KD树在搜索路径优化时使用的是两点之间的距离来判断，而球树使用的是两边之和与第三边大小来判断

K-近邻算法 KNN

Ball-tree search

一开始, ball-tree只有一个根节点, 所有的数据点都指向它, 在每一个步骤, 每个节点会被划分为两个子节点, 下面我们举某次区域的划分为例, 该区域的中心节点为 p_i , 所有节点都指向 p_i

1. 首先, 找到与中心点 p_i 最远的节点, 作为 p_i 的左子节点 p_i^L
2. 接着, 找到与 p_i^L 最远的节点作为 p_i 的右子节点 p_i^R
3. 指向 p_i 的所有节点如果离 p_i^L 近则指向 p_i^L , 否则指向 p_i^R
4. 这样就划分出了新的两个区域, 区域的中心点分别为 p_i^L 和 p_i^R
5. 最后, 记录每个中心节点的位置 c 和包含其所有所属节点的最小半径 r

朴素贝叶斯分类

朴素贝叶斯分类器是一个以贝叶斯定理为基础的多分类的分类器。

对于给定数据，首先基于特征的条件独立性假设，学习输入输出的联合概率分布，然后基于此模型，对给定的输入 x ，利用贝叶斯定理求出后验概率最大的输出 y 。

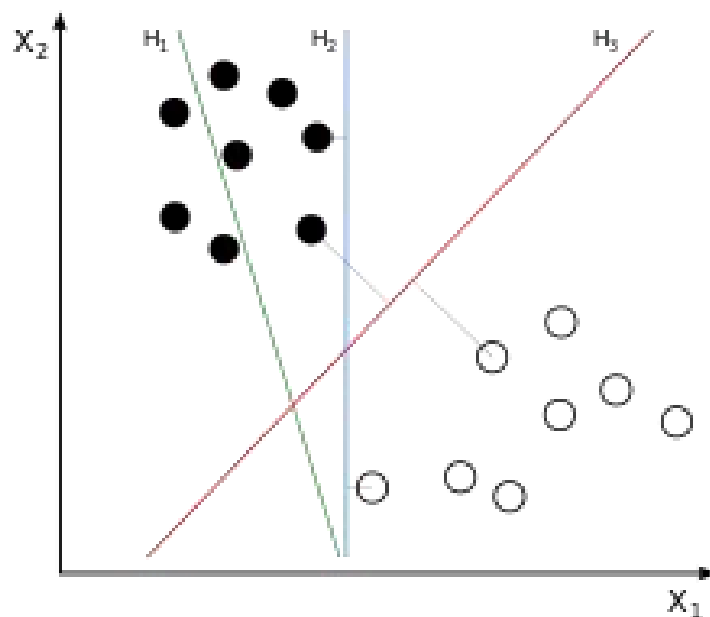
$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)}$$

朴素贝叶斯是典型的生成学习方法，由训练数据学习联合概率分布，并求得后验概率分布。

朴素贝叶斯一般在小规模数据上的表现很好，适合进行多分类任务。

SVM分类

- 早是由 Vladimir N. Vapnik 和 Alexey Ya. Chervonenkis 在1963年提出
- 深度学习（2012）出现之前，SVM被认为机器学习中近十几年来最成功，表现最好的算法



两类？哪条线最好？

SVM分类

SVM寻找区分两类的超平面 (hyper plane), 使边际(margin)最大

总共可以有多少个可能的超平面? 无数条

如何选取使边际(margin)最大的超平面 (Max Margin Hyperplane)?

超平面到一侧最近点的距离等于到另一侧最近点的距离, 两侧的两个超平面平行

