Python 数据分析与应用

第七章：使用 scikit-learn 构建模型

丁烨
dingye@dgut.edu.cn
计算机科学与技术学院

2023-11-22

東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

目录

机器学习简介

机器学习基础

- 机器学习（Machine Learning）
- 人工智能（Artificial Intelligence，AI）的一个分支
- 机器学习是实现人工智能的一个途径，即以机器学习为手段解决人工智能中的问题
- 机器学习在近 30 多年已发展为一门多领域交叉学科
- 涉及概率论、统计学、逼近论、凸分析、计算复杂性理论等多门学科

- 机器学习理论主要是设计和分析一些让计算机可以自动“学习”的算法
- 机器学习算法从数据中自动分析获得规律，并利用规律对未知数据进行预测

机器学习简介

机器学习基础

- 机器学习的核心问题是：
- 通过学习 n 个数据样本及其标注的结果，预测其它 k 个数据样本的结果

- 大部分机器学习的算法实际上都在解决这个问题，例如：
- AlphaGo 通过学习已存在的围棋选手的下棋方法，预测战胜其他围棋选手的下棋方法
- Tesla 的自动驾驶系统通过学习已存在的驾驶方法，预测不同路况的驾驶方法
- 人脸识别系统通过学习已知的人脸图片，预测一张图片是否为人脸

电影评分预测算法

- 电影评分预测算法
- 通过预测电影评分，决定：
- 给用户推荐什么电影票
- 市场应该如何推广等

- 在右边的例子中，每一格为：
- 用户（a, b, …）对电影（A, B, …）的评分

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

机器学习简介

电影评分预测算法

- 机器学习算法希望通过：
- 学习用户 a - g 对于电影 A - C 的评分结果
- 来预测用户 g 对于电影 C 的评分

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

机器学习简介

电影评分预测算法

- 在这个例子中有：
- 两个数据集：
- 训练集（Training Set）：黄色部分
- 验证集（Validation Set）：绿色部分

- 两个操作：
- 训练（Train）= 学习（Learn）
- 预测（Predict）

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

机器学习简介

电影评分预测算法

- 训练（Train）
- 提出一个算法，使得机器能够总结出一个模型（Model）
- 这个模型可以用来预测结果

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

机器学习简介

电影评分预测算法

- 例如：
- 因为 a 对 A 的评分是 10，对 B 的评分是 5
- 而且 g 对 A 的评分也是 10，对 B 的评分也是 5
- 那么我们认为 a 和 g 的行为类似
- 又因为 a 对 C 的评分是 10
- 所以 g 对 C 的评分也应该是 10

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

- 这个算法叫做：
- 最大似然估计（Maximum Likelihood Estimation）

电影评分预测算法

- 最大似然估计有可能会出错
- 例如，我们观测到：
- (b,A)=9，(b,C)=8
- (f,A)=9，(f,C)=8
- 那么因为 (b,B)=6，所以 (f,B)=6：成立
- 但是这个模型对于 c 和 e 却不成立（蓝色部分）
- 为了让模型更通用，我们需要调整算法

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

电影评分预测算法

❖ 例如，我们可以改为以下的算法：

1. 计算 g 与其他人在 A 和 B 上的相似度
2. 将计算出来的相似度向量与其他人对 C 的评价向量点乘
3. 并将结果求平均
4. 得到的结果就是 g 对 C 的相似度

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

❖ 在计算相似度的时候，我们可以采用：

❖ 余弦距离（Cosine Similarity）

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

机器学习简介

电影评分预测算法

- 这个算法叫做：
- 协同过滤（Collaborative Filtering）

- 机器学习领域里类似的算法很多
- 有简单的统计类算法
- 也有复杂的深度学习算法

- 那么如何评估算法的有效性呢?

	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

算法的有效性

- 为了评价算法的有效性，并提升生成模型的效果
- 我们需要不断的用黄色部分的数据进行训练
- 并用蓝色部分的数据进行测试
- 黄色部分被称为：训练集（Training Set）
- 蓝色部分被称为：测试集（Test Set）

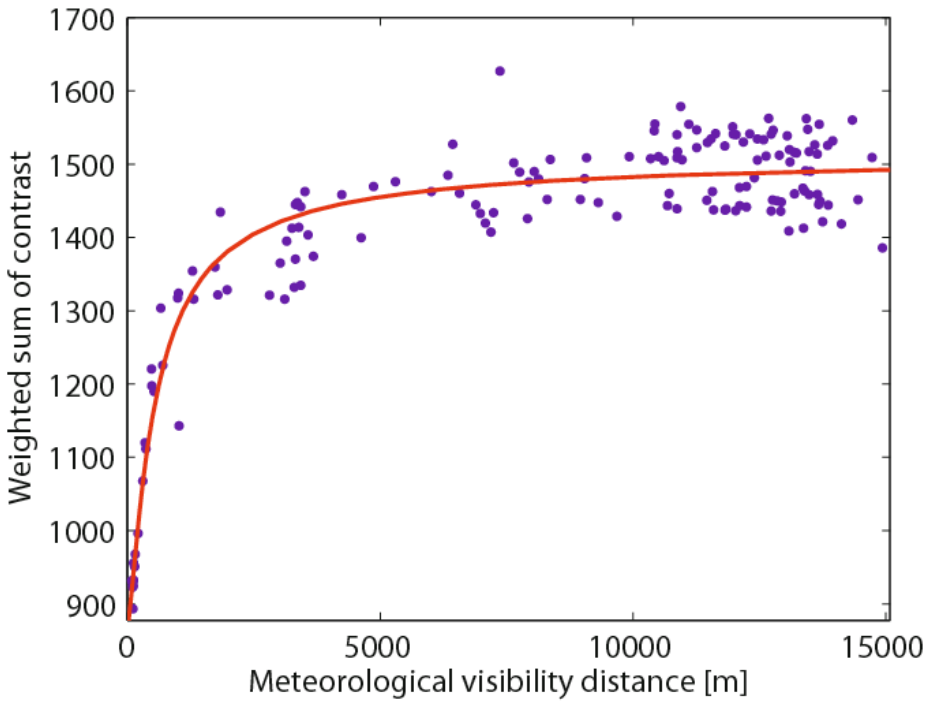
	A	B	C
a	10	5	10
b	9	6	8
c	8	3	9
d	7	4	8
e	8	5	9
f	9	6	8
g	10	5	?

- 通过不断更换黄色和蓝色的数据
- 可以更好的评估算法的有效性，并提升生成模型的效果
- 这个方法叫：交叉验证（Cross Validation）

总结

- Train 训练
- 提出一个算法（Algorithm）
- 通过在训练集上的训练（Train）
- 使得机器能够总结出一个模型（Model）
- 这个模型可以用来预测（Predict）新数据的标签

- 总结的过程也被称为：拟合（Fit）



WHAT IS MACHINE LEARNING?

什么是机器学习？

目录

scikit-learn 简介

- scikit-learn
- https://scikit-learn.org/
- https://github.com/scikit-learn/scikit-learn

- 一个开源的基于 Python 的科学计算及机器学习工具包
- 属于 SciPy 项目的一部分
- scikit-learn 是一个非常基础、简单的机器学习工具包
- scikit-learn 本身不支持深度学习
- scikit-learn 不支持 GPU 加速
- scikit-learn 只提供了经过广泛验证的算法

使用 scikit-learn 构建模型

安装 scikit-learn

- 使用 pip 安装 scikit-learn：
- `pip3 install --user -U scikit-learn`

- 如果安装不成功，可尝试使用 apt 安装：
- `sudo apt install python3-sklearn`

机器学习算法分类

- 机器学习算法主要有以下两类：

- Supervised Learning 有监督学习
- 训练集中有标注的结果
- 分类（Classification）：标注的结果为离散数据（Discrete）
- 回归（Regression）：标注的结果为连续数据（Continuous）

- Unsupervised Learning 无监督学习
- 训练集中没有任何标注的结果
- 聚类（Clustering）：自动生成数据的类别

机器学习算法分类

- 离散（Discrete）数据：
- 数据是离散的类别，例如：
- 电影评分：1、2、3、4、5、6、7、8、9、10
- 花的颜色：红色、橙色、黄色、绿色、蓝色、青色、紫色

- 连续（Continuous）数据：
- 数据是连续的，例如：
- 电影评分可以从 1 - 10 种任选：5.7、9.9、…
- 花的红色程度：0.1、0.5、0.9、…

使用 scikit-learn 构建模型

机器学习算法分类

- 连续数据的训练和预测（即回归算法）精准度比较差
- 一般情况下会先把连续数据离散化
- 例如：把 1 - 10 的电影评分离散到 1、2、…、10 中

使用 scikit-learn 构建模型

加载示例数据

- 为了方便学习机器学习算法，scikit-learn 提供了一些示例数据，包括：

- 安德森鸢尾花卉数据集
- https://en.wikipedia.org/wiki/Iris_flower_data_set

- UCI 手写数字数据集
- https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

使用 scikit-learn 构建模型

加载示例数据

❖ 加载 iris 和 digits 两个数据集：

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
```

❖ scikit-learn 的数据集通常是一个字典（Dict），包含样本特征的矩阵和标签

❖ 样本特征的矩阵中，每一行为一个样本（Sample），每一列为一个特征（Feature）

❖ 标签（Label 或 Ground Truth）为一个向量

加载示例数据

- 查看样本特征的矩阵：

```
>>> print(digits.data)
[[ 0.   0.   5. ...   0.   0.   0.]
 [ 0.   0.   0. ...  10.   0.   0.]
 [ 0.   0.   0. ...  16.   9.   0.]
 ...
 [ 0.   0.   1. ...   6.   0.   0.]
 [ 0.   0.   2. ...  12.   0.   0.]
 [ 0.   0.  10. ...  12.   1.   0.]]
```

- 注意，手写数字数据集中的手写字体实际上为一张 8 * 8 的图片
- 这里已经把 8 * 8 的矩阵展开为一条长度为 64 的向量
- 这种展开被称为：矩阵的向量化
- 矩阵的向量化会一定程度影响预测结果

加载示例数据

- 查看标签

```
>>> digits.target
array([0, 1, 2, ..., 8, 9, 8])
```

- 手写数字数据集的标签共有 10 类：0 - 9
- 代表图片样本描绘的手写数字

使用 scikit-learn 构建模型

训练与验证

- 针对手写数字数据集来说，我们的训练目标是：
- 给定一张图片，预测其所描绘的数字

- 在 scikit-learn 中，一个模型可以通过拟合样本特征的矩阵到标签向量上生成，即：
- `fit(X, y)`

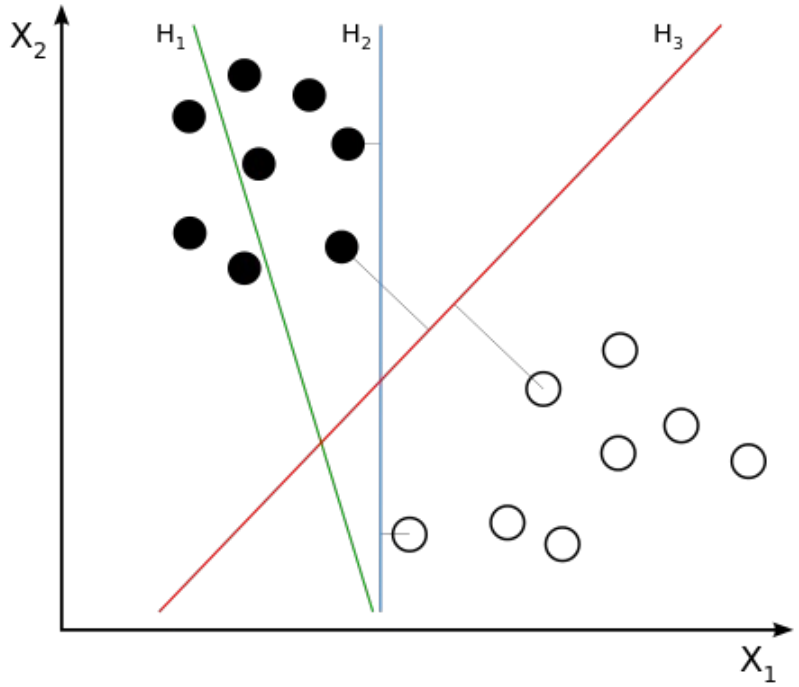
- 训练好的模型可以通过预测函数对验证集或测试集进行预测，即：
- `predict(T)`

训练与验证

- 例如，我们可以使用 sklearn.svm.SVC 来创建一个支持向量机的分类器

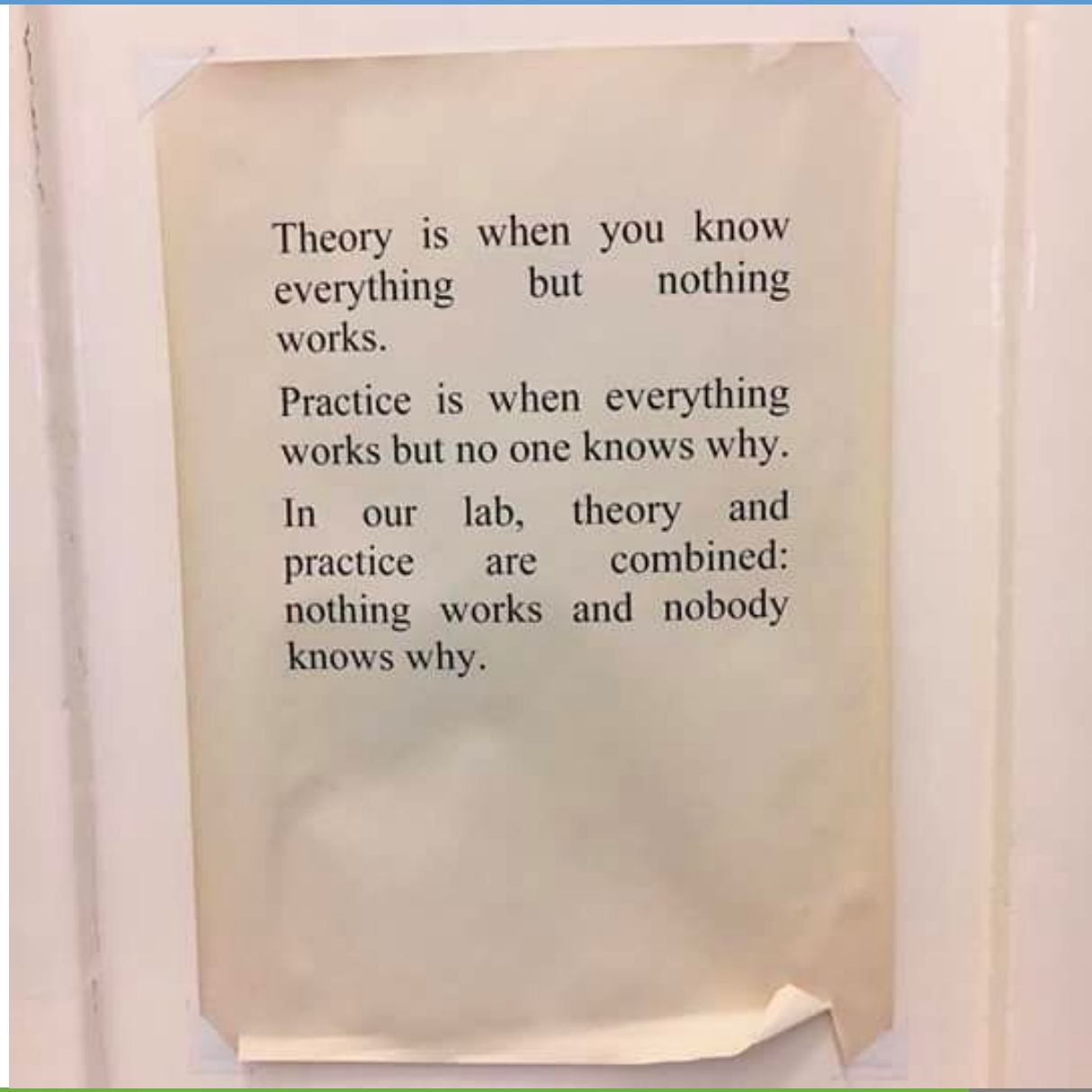
```
>>> from sklearn import svm
>>> clf = svm.SVC(gamma=0.001, C=100.)
```

- 创建模型的参数是和算法本身相关的
- 在这节课中，我们可以视模型为一个黑盒子



训练与验证

- 黑盒算法
- “并不清楚原理是什么，但是效果好”



训练与验证

- 选择模型的参数是十分重要的
- 如何选择参数是机器学习领域一个重要的研究方向
- 我们可以通过例如：
- 网格搜索（Grid Search）
- 交叉验证（Cross Validation）
- 等方法在训练集和测试集上寻找最优化的参数

使用 scikit-learn 构建模型

训练与验证

```
>>> from sklearn import svm
>>> clf = svm.SVC(gamma=0.001, C=100.)
```

- 创建的模型为一个空模型，需要经过训练后才可以使用
- 训练时，我们采用除最后一行之外的数据作为训练集，最后一行数据作为验证集：

```
>>> clf.fit(digits.data[:-1], digits.target[:-1])
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

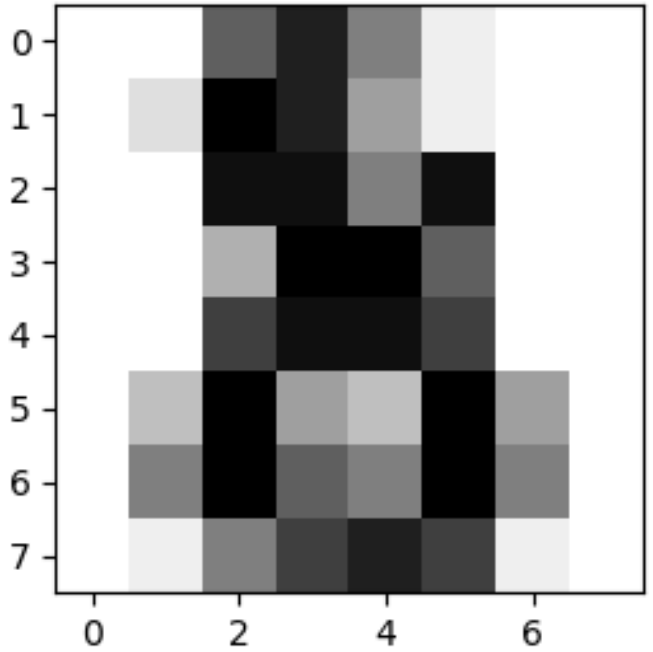
训练与验证

❖ 训练好之后，我们就可以使用 predict() 函数对验证集进行预测：

```
>>> clf.predict(digits.data[-1:])
array([8])
```

❖ 输出的结果即位手写识别的结果：8

❖ 你认为 SVM 预测的结果是否准确?



模型的保存和载入

❖ scikit-learn 可以通过 Python 标准库的 pickle 模组保存模型：

```
>>> from sklearn import svm
>>> from sklearn import datasets
>>> clf = svm.SVC(gamma='scale')
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0:1])
array([0])
>>> y[0]
0
```

使用 scikit-learn 构建模型

模型的保存和载入

❖ 也可以通过 joblib 模组保存模型：

```
>>> from joblib import dump, load
>>> dump(clf, 'filename.joblib')

>>> clf = load('filename.joblib')
```

❖ joblib 相对 pickle 来说处理速度更快

❖ 这对大规模数据训练出来的复杂模型尤其重要

❖ 但是，joblib 只能将模型保存在磁盘上，不能以字符串形式保存在内存中

使用 scikit-learn 构建模型

scikit-learn 的常见约定

- 由于机器学习的过程相当复杂，scikit-learn 约定了大量的默认值
- 遵守这些约定可以降低在细节上出错的概率

scikit-learn 的常见约定

❖ 除非特别指定，否则输入数据的默认类型为 float64：

```
>>> import numpy as np
>>> from sklearn import random_projection


>>> rng = np.random.RandomState(0)
>>> X = rng.rand(10, 2000)
>>> X = np.array(X, dtype='float32')
>>> X.dtype
dtype('float32')


>>> transformer = random_projection.GaussianRandomProjection()
>>> X_new = transformer.fit_transform(X)
>>> X_new.dtype
dtype('float64')
```

scikit-learn 的常见约定

❖ 回归模型会将标签转换为 float64 类型，分类和聚类模型则会维持原状：

```
>>> from sklearn import datasets
>>> from sklearn.svm import SVC
>>> iris = datasets.load_iris()
>>> clf = SVC(gamma='scale')
>>> clf.fit(iris.data, iris.target)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)


>>> list(clf.predict(iris.data[:3]))
[0, 0, 0]
```

scikit-learn 的常见约定

❖ 回归模型会将标签转换为 float64 类型，分类和聚类模型则会维持原状：

```
>>> clf.fit(iris.data, iris.target_names[iris.target])
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)


>>> list(clf.predict(iris.data[:3]))
['setosa', 'setosa', 'setosa']
```

scikit-learn 的常见约定

❖ 模型的参数可以在拟合之后修改，再次调用 fit 函数将重新训练并覆盖原来的模型

```
>>> import numpy as np
>>> from sklearn.datasets import load_iris
>>> from sklearn.svm import SVC
>>> X, y = load_iris(return_X_y=True)


>>> clf = SVC()
>>> clf.set_params(kernel='linear').fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
>>> clf.predict(X[:5])
array([0, 0, 0, 0, 0])
```

scikit-learn 的常见约定

❖ 模型的参数可以在拟合之后修改，再次调用 fit 函数将重新训练并覆盖原来的模型

```
>>> clf.set_params(kernel='rbf', gamma='scale').fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
>>> clf.predict(X[:5])
array([0, 0, 0, 0, 0])
```

scikit-learn 的常见约定

❖ 当预测的对象包含多个可能分类时，输出的结果将会向分类的数量对齐

```
>>> from sklearn.svm import SVC
>>> from sklearn.multiclass import OneVsRestClassifier
>>> from sklearn.preprocessing import LabelBinarizer


>>> X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
>>> y = [0, 0, 1, 1, 2]


>>> classif = OneVsRestClassifier(estimator=SVC(gamma='scale',
...                                             random_state=0))
>>> classif.fit(X, y).predict(X)
array([0, 0, 1, 1, 2])
```

scikit-learn 的常见约定

❖ 当预测的对象包含多个可能分类时，输出的结果将会向分类的数量对齐

```
>>> y = LabelBinarizer().fit_transform(y)
>>> classif.fit(X, y).predict(X)
array([[1, 0, 0],
       [1, 0, 0],
       [0, 1, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

scikit-learn 的常见约定

- 在机器学习中，数值分类是非常容易引起误解的，一般情况下需要：
- 将数值分类转换成字符串分类，或将数值分类二进制化：

```
[0, 0, 1, 1, 2]
```

```
[[1, 0, 0],
 [1, 0, 0],
 [0, 1, 0],
 [0, 1, 0],
 [0, 0, 1]]
```

- 如果不做处理，模型很难分辨数值分类是类别数据还是连续数据
- 如果将类别数据按照连续数据处理，不同分类的距离是不一样的
- dist(0,1)=1 ≠ dist(0,2)=2

目录

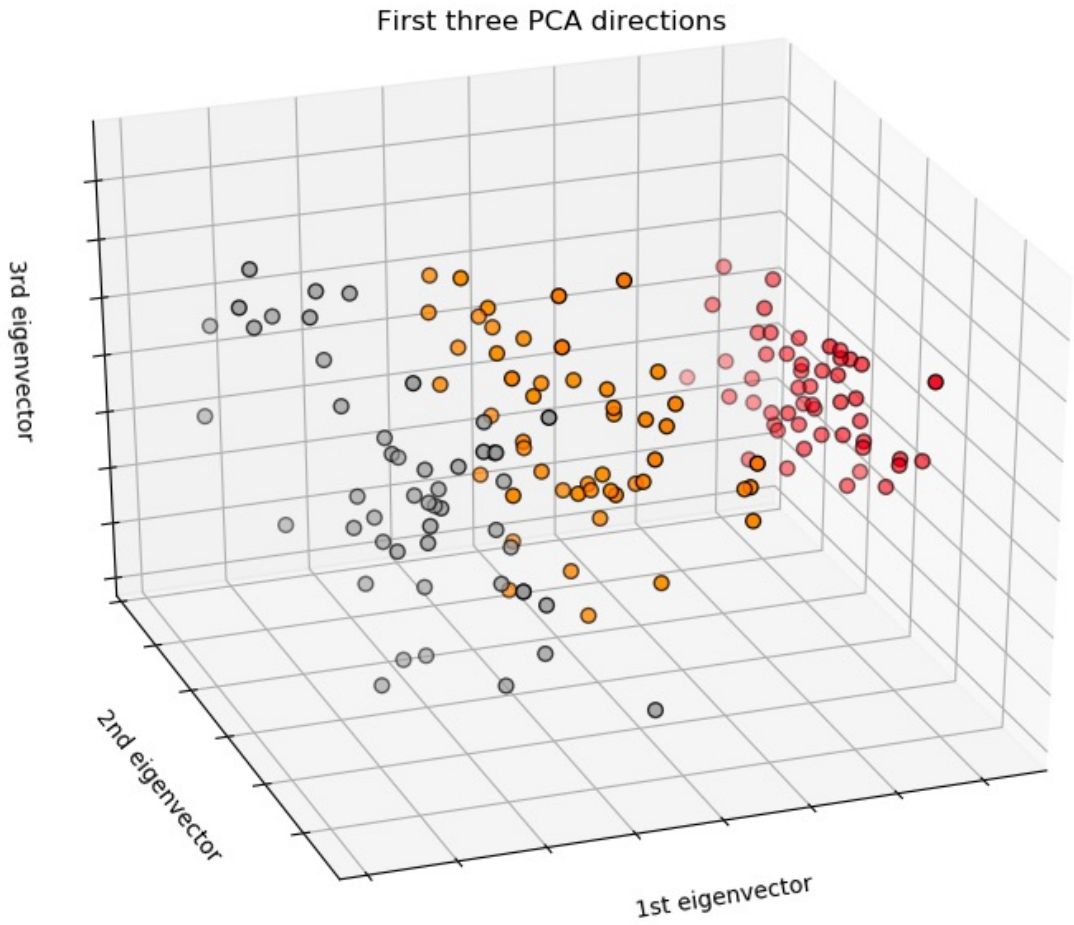
scikit-learn 支持的机器学习算法

k-近邻算法

- 聚类算法
- k-近邻（k-nearest-neighbors，kNN）
- 从训练集中取一个样本，寻找离样本最近的 k 个近邻
- 将这 k 个近邻划为与样本高概率的同类
- 不断枚举样本，直到所有样本收敛（Converge）

❖ 安德森鸢尾花卉数据集中的样本有三类：Setosa、Versicolour、Virginica

```
>>> import numpy as np
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> iris_X = iris.data
>>> iris_y = iris.target
>>> np.unique(iris_y)
array([0, 1, 2])
```



scikit-learn 支持的机器学习算法

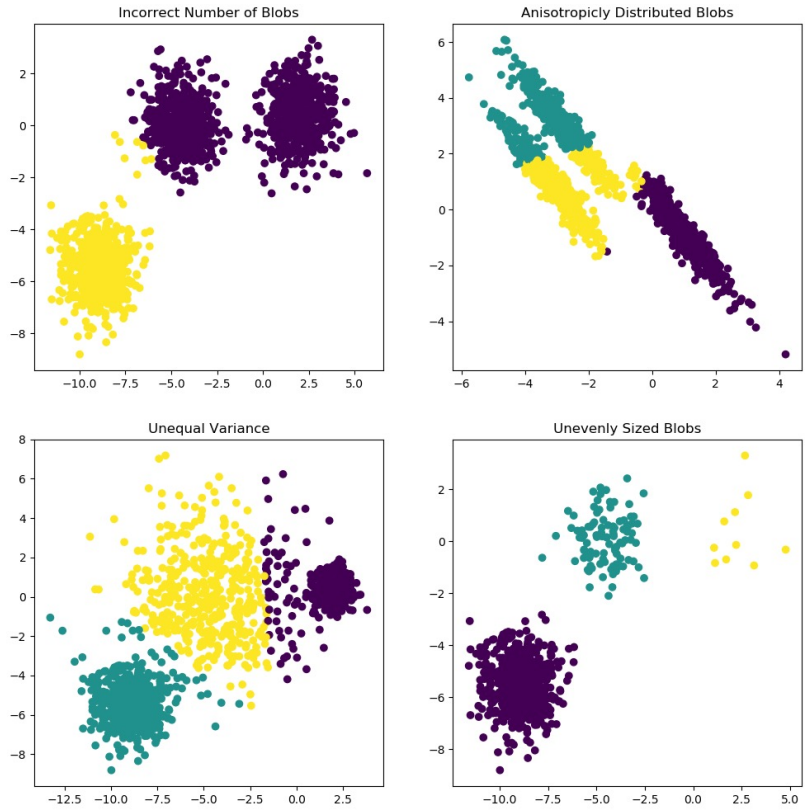
k-近邻算法

```
>>> # Split iris data in train and test data
>>> np.random.seed(0)
>>> indices = np.random.permutation(len(iris_X))
>>> iris_X_train = iris_X[indices[:-10]]
>>> iris_y_train = iris_y[indices[:-10]]
>>> iris_X_test = iris_X[indices[-10:]]
>>> iris_y_test = iris_y[indices[-10:]]
>>> # Create and fit a nearest-neighbor classifier
>>> from sklearn.neighbors import KNeighborsClassifier
>>> knn = KNeighborsClassifier()
>>> knn.fit(iris_X_train, iris_y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
>>> knn.predict(iris_X_test)
array([1, 2, 1, 0, 0, 0, 2, 1, 2, 0])
>>> iris_y_test
array([1, 1, 1, 0, 0, 0, 2, 1, 2, 0])
```

k-平均算法

- 聚类算法
- k-平均（k-means）
- 将样本平均的划分在 k 个聚类中
- 使得每个聚类中的样本离聚类中心的距离小于其与其他聚类中心的距离
- 不断调整聚类中的样本
- 直到所有聚类收敛（Converge）

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$



scikit-learn 支持的机器学习算法

k-平均算法

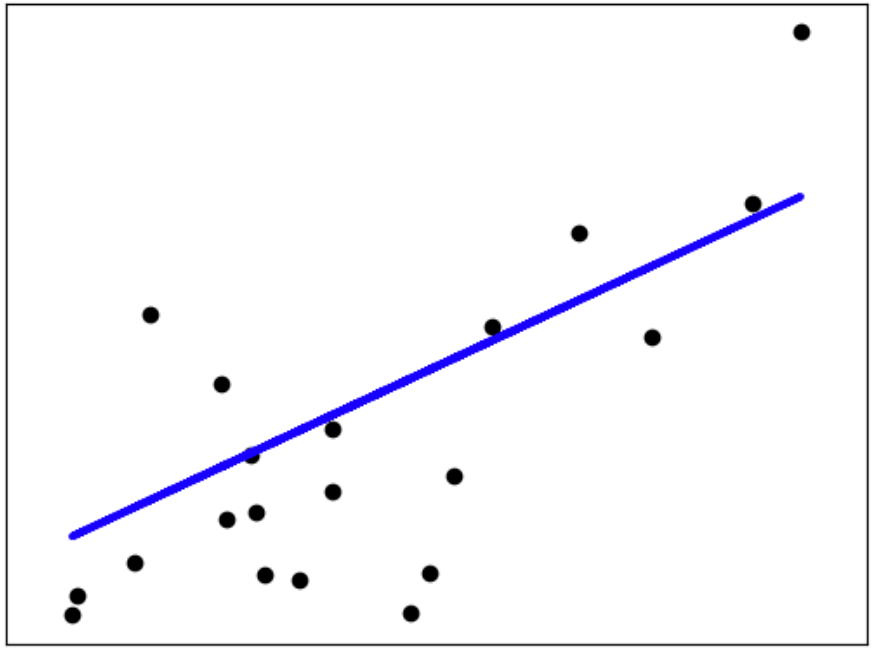
```
>>> from sklearn import cluster, datasets
>>> iris = datasets.load_iris()
>>> X_iris = iris.data
>>> y_iris = iris.target


>>> k_means = cluster.KMeans(n_clusters=3)
>>> k_means.fit(X_iris)
KMeans(algorithm='auto', copy_x=True, init='k-means++', ...
>>> print(k_means.labels_[::10])
[1 1 1 1 1 0 0 0 0 0 2 2 2 2 2]
>>> print(y_iris[::10])
[0 0 0 0 0 1 1 1 1 1 2 2 2 2 2]
```

线性回归

- 回归算法
- 线性回归（Linear Regression）
- 寻找一个线性公式，使其可以将样本划分在两个不同的区域中
- 不断调整线性公式，直到所有样本收敛
- 数据中每多一个分类，就需要多一个线性公式

Linear models: $y = X\beta + \epsilon$

- X: data
- y: target variable
- β: Coefficients
- ϵ: Observation noise



❖ 加载糖尿病数据集

```
>>> diabetes = datasets.load_diabetes()
>>> diabetes_X_train = diabetes.data[:-20]
>>> diabetes_X_test  = diabetes.data[-20:]
>>> diabetes_y_train = diabetes.target[:-20]
>>> diabetes_y_test  = diabetes.target[-20:]
```

scikit-learn 支持的机器学习算法

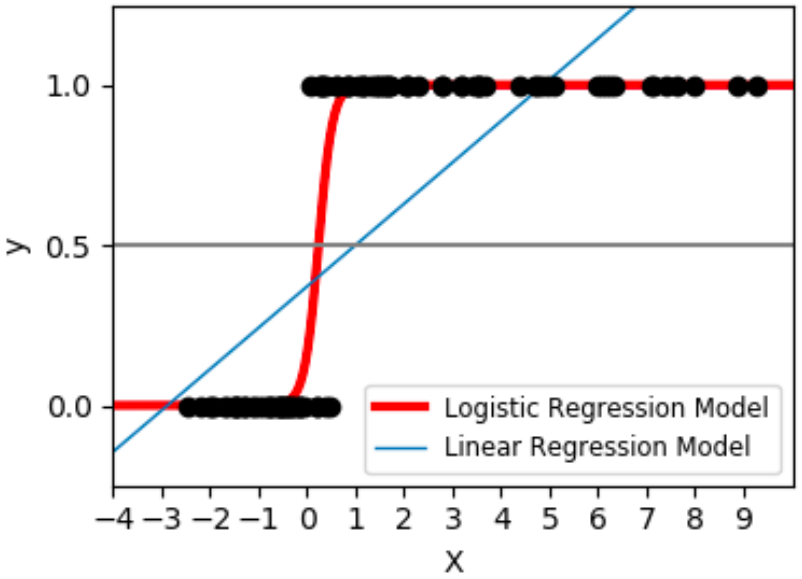
线性回归

```
>>> from sklearn import linear_model
>>> regr = linear_model.LinearRegression()
>>> regr.fit(diabetes_X_train, diabetes_y_train)
...
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                 normalize=False)
>>> print(regr.coef_)
[   0.30349955 -237.63931533  510.53060544  327.73698041 -814.13170937
  492.81458798  102.84845219  184.60648906  743.51961675   76.09517222]

>>> # The mean square error
>>> np.mean((regr.predict(diabetes_X_test) - diabetes_y_test)**2)
...
2004.56760268...
>>> # Explained variance score: 1 is perfect prediction
>>> # and 0 means that there is no linear relationship
>>> # between X and y.
>>> regr.score(diabetes_X_test, diabetes_y_test)
0.5850753022690...
```

逻辑回归

- 分类算法
- 对数几率回归（Logistic Regression）
- 寻找一个非线性的对数几率公式，使其可以将样本划分在两个不同的区域中
- 不断调整对数几率公式，直到所有样本收敛
- 数据中每多一个分类，就需要多一个对数几率公式
- 虽然名字叫做“回归”，但对数几率回归是一个分类算法

$$y = \text{sigmoid}(X\beta - \text{offset}) + \epsilon = \frac{1}{1 + \exp(-X\beta + \text{offset})} + \epsilon$$



scikit-learn 支持的机器学习算法

逻辑回归

```
>>> log = linear_model.LogisticRegression(solver='lbfgs', C=1e5,
...                                       multi_class='multinomial')
>>> log.fit(iris_X_train, iris_y_train)
LogisticRegression(C=100000.0, class_weight=None, dual=False,
    fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100,
    multi_class='multinomial', n_jobs=None, penalty='l2', random_state=None,
    solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)
```

scikit-learn 支持的机器学习算法

其它的机器学习算法

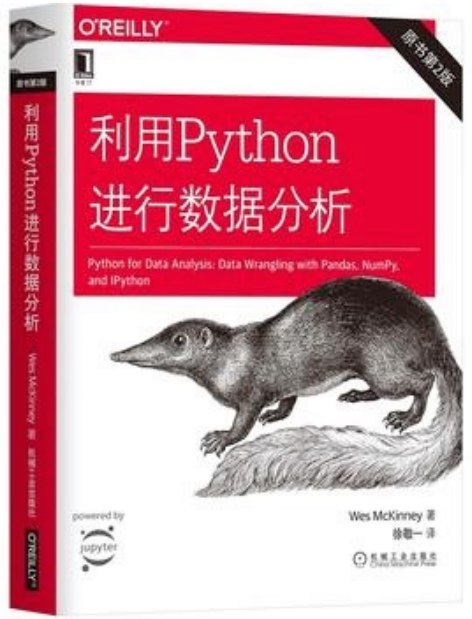
- 其它的机器学习算法可以参考 scikit-learn 的 API 文档：
- https://scikit-learn.org/stable/modules/classes.html
- scikit-learn 的官方文档简介了每个算法的基本原理，可以参考学习
- 其中包括很多常用的算法：
- 有监督学习（分类、回归）、无监督学习（聚类）
- 以及常用的模型处理工具：
- 矩阵分解（Matrix Factorization）
- 特征提取（Feature Extraction）
- 模型评估（Model Estimation）
- 合并方法（Ensemble Method）

进阶功能

- 更多的进阶功能可以参考官方教程：
- https://scikit-learn.org/stable/documentation.html

- 函数详细说明文档及手册：
- https://scikit-learn.org/stable/modules/classes.html

- Wes McKinney《利用 Python 进行数据分析》：
- http://product.dangdang.com/25312917.html



Thanks!