9種常用的機器學習算法實現

DataFunTalk 昨天

以下文章來源於淘系技術，作者陳雷慧（豆苗）

淘系技術

淘系技術官方賬號

簡介

根據機器學習的任務或應用情況的不同，我們通常把機器學習分為三大類：

1、監督學習（Supervised Learning，SL），這類算法的工作原理是使用帶標籤的訓練數據來學習輸入變量X轉化為輸出變量Y的映射函數，換句話說就是求解方程$Y = f(X)$中的f。進一步地，監督學習又可細分為如下三類：

- 回歸（Regression）：預測一個值，如預測降雨量、房價等，較基礎的算法有：Linear Regression
- 分類（Classification）：預測一個標籤，如預測“生病”或“健康”，圖片上是哪種動物等，較基礎的算法有：Logistic Regression、Naive Bayes、K-Nearest Neighbors（KNN）

【另】：集成（Ensembling）也可以歸類為監督學習的一種，它將多個單獨較弱的機器學習模型的預測結合起來，以產生更準確的預測，較基礎的算法有Bagging with Random Forests、Boosting with XGBoost

2、非監督學習（Unsupervised Learning，UL），這類算法的工作原理是從無標籤的訓練數據中學習數據的底層結構。進一步地，非監督學習又可細分為如下三類：

- 关联（Association）：发现集合中项目同时出现的概率，如通过分析超市购物篮，发现啤酒总是和尿片一起购买（啤酒与尿片的故事），较基础的算法有：Apriori
- 聚类（Clustering）：对数据进行分组，以便组内对象比组间对象更相似，较基础的算法有：K-Means
- 降维（Dimensionality Reduction）：减少数据集的变量数量，同时保证重要的信息不被丢失。降维可以通过特征提取方法和特征选择方法来实现，特征提取是执行从高维空间到低维空间的转换，特征选择是选择原始变量的子集，较基础的算法有：PCA

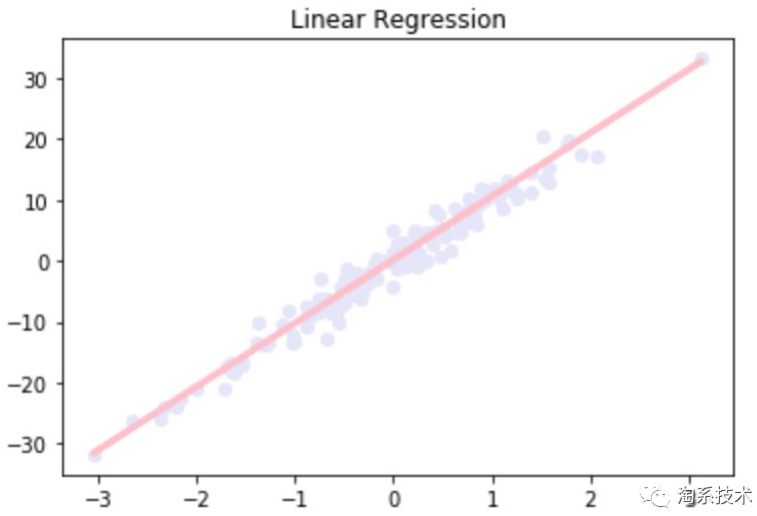
3、强化学习（Reinforcement Learning，DL），让agent根据当前环境状态，通过学习能够获得最大回报的行为来决定下一步的最佳行为。

实现

以上列出的算法都是简单常用的，基于scikit-learn可以仅用几行代码就完成模型训练、预测、评估和可视化。关于算法的原理知乎上有很多精彩的回答，这里不会赘述，仅给出代码的实现与可视化。

Linear Regression

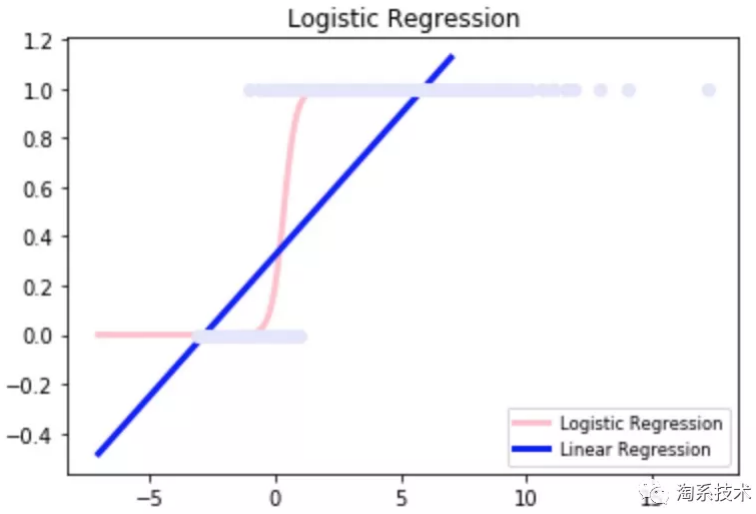
它为变量分配最佳权重，以创建一条直线或一个平面或更高维的超平面，使得预测值和真实值之间的误差最小化。具体原理参考：用人话讲明白线性回归LinearRegression - 化简可得的文章 - 知乎。下面以一元线性回归为例，给出代码实现。



```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split

# Linear Regression 一元回归
from sklearn import linear_model
from sklearn.metrics import mean_squared_error

# 1. 准备数据
lr_X_data, lr_y_data = datasets.make_regression(n_samples=500,n_features=1,n_t
# 2. 构造训练与测试集
lr_X_train, lr_X_test, lr_y_train, lr_y_test = train_test_split(lr_X_data, lr_
# 3. 训练模型
lr_model = linear_model.LinearRegression()
lr_model.fit(lr_X_train, lr_y_train)
# 4. 预测数据
lr_y_pred = lr_model.predict(lr_X_test)
# 5. 评估模型
lr_mse = mean_squared_error(lr_y_test, lr_y_pred)
print("mse:", lr_mse)
# 6. 可视化
plt.figure('Linear Regression')
plt.title('Linear Regression')
plt.scatter(lr_X_test, lr_y_test, color='lavender', marker='o')
plt.plot(lr_X_test, lr_y_pred, color='pink', linewidth=3)
plt.show()

# print info mse: 4.131366697554779
```

Logistic Regression

虽然写着回归，但实际上是一种二分类算法。它将数据拟合到logit函数中，所以称为logit回归。简单来说就是基于一组给定的变量，用logistic function来预测这个事件的概率，给出一个介于0和1之间的输出。具体原理参考：用人话讲明白逻辑回归Logistic regression - 化简可得的文章 - 知乎，下面给出代码的实现。

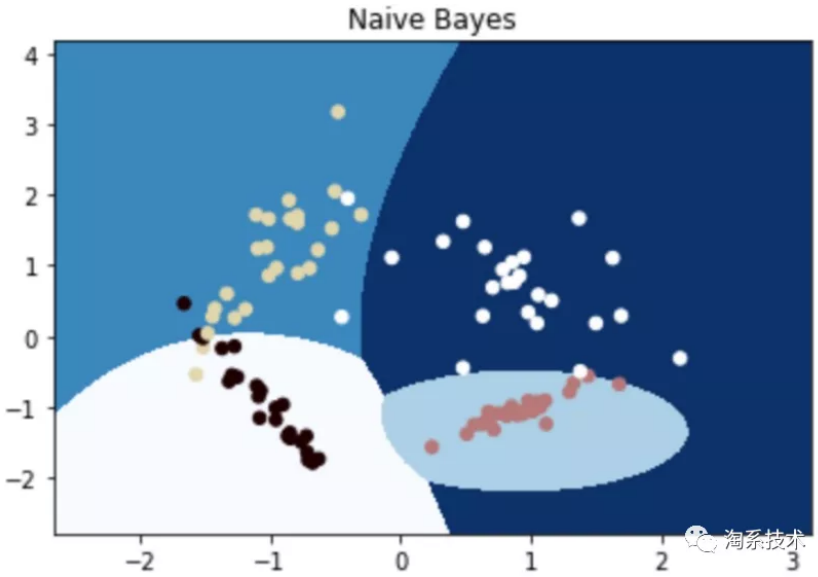



```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split

# Logistic Regression 二分类
from sklearn import linear_model

# 1. 准备数据
np.random.seed(123)
logit_X_data = np.random.normal(size=1000)
logit_y_data = (logit_X_data>0).astype(np.float)
logit_X_data[logit_X_data>0]*=5
logit_X_data+=.4*np.random.normal(size=1000)
logit_X_data=logit_X_data[:,np.newaxis]
# 2. 构造训练与测试集
logit_X_train, logit_X_test, logit_y_train, logit_y_test = train_test_split(lo
# 3. 训练模型
logit_model=linear_model.LogisticRegression(C=1e4) #classifier
logit_model.fit(logit_X_train,logit_y_train)
```

```
# 4. 预测数据
logit_y_pred = logit_model.predict(logit_X_test)
# 5. 评估模型
logit_acc = logit_model.score(logit_X_test,logit_y_pred)
print("accuracy:", logit_acc)
# 5. 可视化
logit_X_view=np.linspace(-7,7,277)
logit_X_view = logit_X_view[:,np.newaxis]
def model(x):
    return 1/(1+np.exp(-x))
loss=model(logit_X_view*logit_model.coef_+logit_model.intercept_).ravel()
plt.figure('Logistic Regression')
plt.title('Logistic Regression')
plt.scatter(logit_X_train.ravel(), logit_y_train, color='lavender',zorder=17)
plt.plot(logit_X_view, loss, color='pink',linewidth=3)

lr_model=linear_model.LinearRegression()
lr_model.fit(logit_X_train,logit_y_train)
plt.plot(logit_X_view, lr_model.predict(logit_X_view), color='blue', linewidth
plt.legend(('Logistic Regression','Linear Regression'),loc='lower right',fonts

# print info accuracy: 1.0
```

Naive Bayes

朴素贝叶斯是一种基于贝叶斯定理的分类方法，它会假设一个类中的某个特征与其他特征无关。这个模型不仅非常简单，而且比许多高度复杂的分类方法表现得更好。具体原理参考：朴素贝叶斯算法原理小结 - 刘建平Pinard，下面给出代码的实现。




```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

# Naive Bayes 任务为分类, n_classes=4
import sklearn.naive_bayes as nb
# 1. 准备数据
nb_X_train, nb_y_train = make_classification(n_features=2, n_redundant=0, n_ir
                            random_state=1, n_clusters_per_class=1, n_classes=4
# 2. 构造训练与测试集
l, r = nb_X_train[:, 0].min() - 1, nb_X_train[:, 0].max() + 1
b, t = nb_X_train[:, 1].min() - 1, nb_X_train[:, 1].max() + 1
n = 1000
grid_x, grid_y = np.meshgrid(np.linspace(l, r, n), np.linspace(b, t, n))
nb_X_test = np.column_stack((grid_x.ravel(), grid_y.ravel()))
# 3. 训练模型
nb_model = nb.GaussianNB()
nb_model.fit(nb_X_train, nb_y_train)
# 4. 预测数据
nb_y_pred = nb_model.predict(nb_X_test)
# 5. 可视化
grid_z = nb_y_pred.reshape(grid_x.shape)
plt.figure('Naive Bayes')
```

```
plt.title('Naive Bayes')
plt.pcolormesh(grid_x, grid_y, grid_z, cmap='Blues')
plt.scatter(nb_X_train[:, 0], nb_X_train[:, 1], s=30, c=nb_y_train, cmap='pink
plt.show()
```

K-Nearest Neighbors

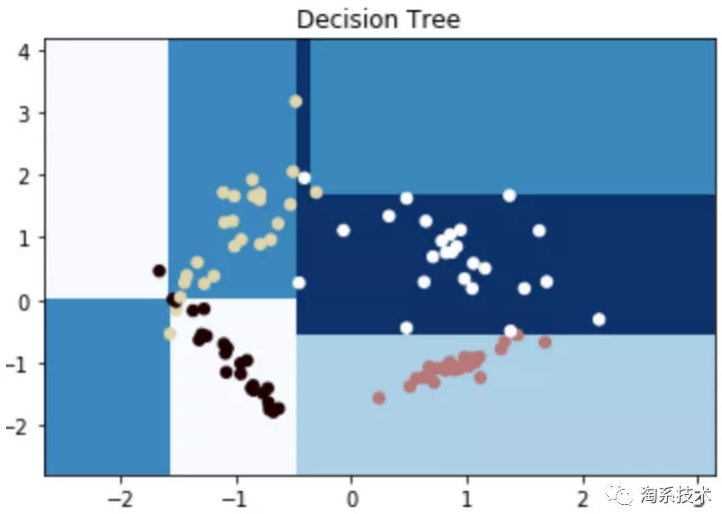
这是用于分类和回归的机器学习算法(主要用于分类)。它考虑了不同的质心，并使用欧几里得函数来比较距离。接着分析结果并将每个点分类到组中，以优化它，使其与所有最接近的点一起放置。它使用k个最近邻的多数票对数据进行分类预测。具体原来参考：K近邻法(KNN)原理小结 - 刘建平Pinard，下面给出代码的实现。

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

# Naive Bayes 任务为分类, n_classes=4
import sklearn.naive_bayes as nb
# 1. 准备数据
nb_X_train, nb_y_train = make_classification(n_features=2, n_redundant=0, n_in
                            random_state=1, n_clusters_per_class=1, n_classes=4
# 2. 构造训练与测试集
l, r = nb_X_train[:, 0].min() - 1, nb_X_train[:, 0].max() + 1
b, t = nb_X_train[:, 1].min() - 1, nb_X_train[:, 1].max() + 1
n = 1000
grid_x, grid_y = np.meshgrid(np.linspace(l, r, n), np.linspace(b, t, n))
nb_X_test = np.column_stack((grid_x.ravel(), grid_y.ravel()))
# 3. 训练模型
nb_model = nb.GaussianNB()
nb_model.fit(nb_X_train, nb_y_train)
# 4. 预测数据
nb_y_pred = nb_model.predict(nb_X_test)
# 5. 可视化
```

```
grid_z = nb_y_pred.reshape(grid_x.shape)
plt.figure('Naive Bayes')
plt.title('Naive Bayes')
plt.pcolormesh(grid_x, grid_y, grid_z, cmap='Blues')
plt.scatter(nb_X_train[:, 0], nb_X_train[:, 1], s=30, c=nb_y_train, cmap='pink
plt.show()
```

Decision Tree

遍历树，并将重要特征与确定的条件语句进行比较。它是降到左边的子分支还是降到右边的子分支取决于结果。通常，更重要的特性更接近根，它可以处理离散变量和连续变量。具体原理参考：深入浅出理解决策树算法（一）-核心思想 - 忆臻的文章 - 知乎，下面给出代码的实现。

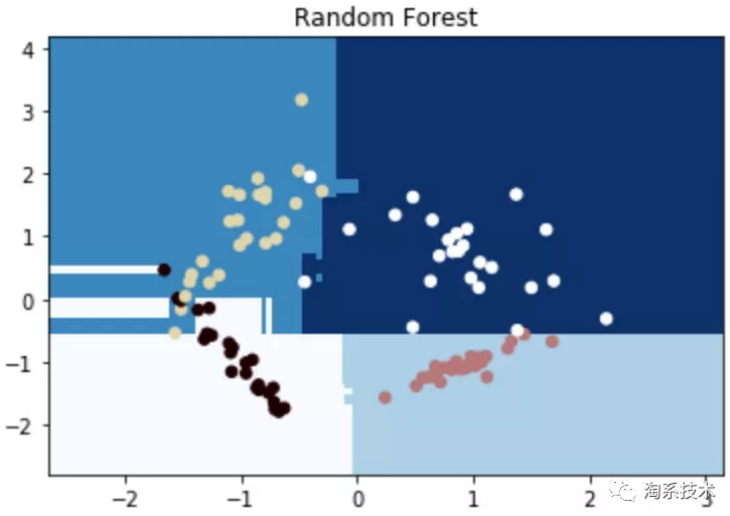



```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

# K-Nearest Neighbors 任务为分类, n_classes=4
from sklearn.neighbors import KNeighborsClassifier
# 1. 准备数据
knn_X_train, knn_y_train = make_classification(n_features=2, n_redundant=0, n_
                           random_state=1, n_clusters_per_class=1, n_classes=4
```

```
# 2. 构造训练与测试集
l, r = knn_X_train[:, 0].min() - 1, knn_X_train[:, 0].max() + 1
b, t = knn_X_train[:, 1].min() - 1, knn_X_train[:, 1].max() + 1
n = 1000
grid_x, grid_y = np.meshgrid(np.linspace(l, r, n), np.linspace(b, t, n))
knn_X_test = np.column_stack((grid_x.ravel(), grid_y.ravel()))
# 3. 训练模型
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(knn_X_train, knn_y_train)
# 4. 预测数据
knn_y_pred = knn_model.predict(knn_X_test)
# 5. 可视化
grid_z = knn_y_pred.reshape(grid_x.shape)
plt.figure('K-Nearest Neighbors')
plt.title('K-Nearest Neighbors')
plt.pcolormesh(grid_x, grid_y, grid_z, cmap='Blues')
plt.scatter(knn_X_train[:, 0], knn_X_train[:, 1], s=30, c=knn_y_train, cmap='p
plt.show()
```

Random Forest

随机森林是决策树的集合。随机采样数据点构造树、随机采样特征子集分割，每棵树提供一个分类。得票最多的分类在森林中获胜，为数据点的最终分类。具体原来参考：独家 | 一文读懂随机森林的解释和实现 - 清华大学数据科学研究院的文章 - 知乎，下面给出代码的实现。

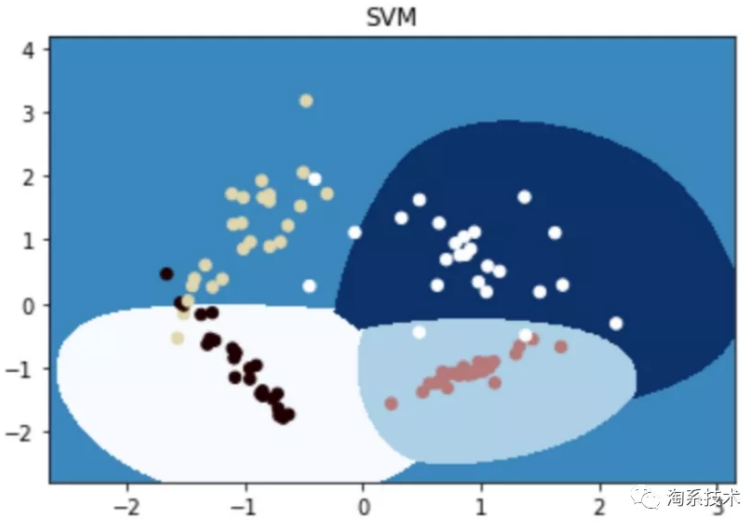


```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

# Decision Tree
from sklearn.tree import DecisionTreeClassifier
# 1. 准备数据
dt_X_train, dt_y_train = make_classification(n_features=2, n_redundant=0, n_ir
                            random_state=1, n_clusters_per_class=1, n_classes=4
# 2. 构造训练与测试集
l, r = dt_X_train[:, 0].min() - 1, dt_X_train[:, 0].max() + 1
b, t = dt_X_train[:, 1].min() - 1, dt_X_train[:, 1].max() + 1
n = 1000
grid_x, grid_y = np.meshgrid(np.linspace(l, r, n), np.linspace(b, t, n))
dt_X_test = np.column_stack((grid_x.ravel(), grid_y.ravel()))
# 3. 训练模型
dt_model = DecisionTreeClassifier(max_depth=4)
dt_model.fit(dt_X_train, dt_y_train)
# 4. 预测数据
dt_y_pred = dt_model.predict(dt_X_test)
# 5. 可视化
grid_z = dt_y_pred.reshape(grid_x.shape)
plt.figure('Decision Tree')
plt.title('Decision Tree')
plt.pcolormesh(grid_x, grid_y, grid_z, cmap='Blues')
plt.scatter(dt_X_train[:, 0], dt_X_train[:, 1], s=30, c=dt_y_train, cmap='pink
plt.show()
```

Support Vector Machines

它将数据映射为空间中的点，使得不同类别的点可以被尽可能宽的间隔分隔开，对于待预测类别的数据，先将其映射至同一空间，并根据它落在间隔的哪一侧来得到对应的类别。具体原来参

考：看了这篇文章你还不懂SVM你就来打我 - SMON的文章 - 知乎，下面给出代码实现。

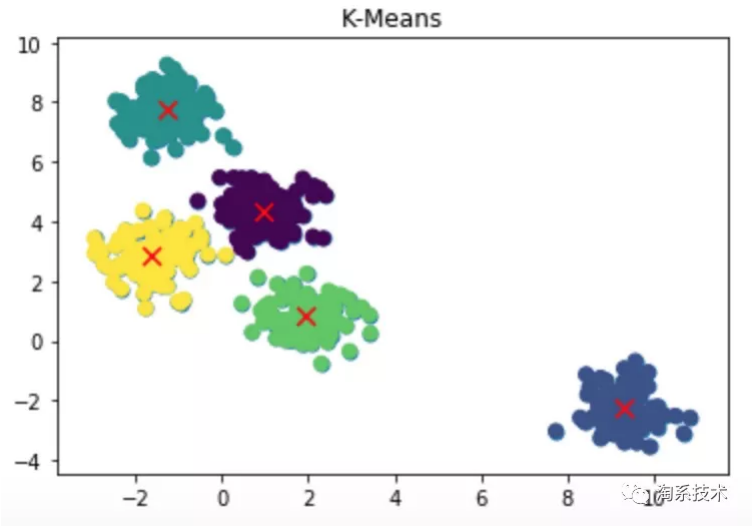



```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

# SVM
from sklearn import svm
# 1. 准备数据
svm_X_train, svm_y_train = make_classification(n_features=2, n_redundant=0, n_
                           random_state=1, n_clusters_per_class=1, n_classes=4
# 2. 构造训练与测试集
l, r = svm_X_train[:, 0].min() - 1, svm_X_train[:, 0].max() + 1
b, t = svm_X_train[:, 1].min() - 1,svm_X_train[:, 1].max() + 1
n = 1000
grid_x, grid_y = np.meshgrid(np.linspace(l, r, n), np.linspace(b, t, n))
svm_X_test = np.column_stack((grid_x.ravel(), grid_y.ravel()))
# 3. 训练模型
# svm_model = RandomForestClassifier(max_depth=4)
svm_model = svm.SVC(kernel='rbf', gamma=1, C=0.0001).fit(svm_X_train, svm_y_tr
svm_model.fit(svm_X_train, svm_y_train)
# 4. 预测数据
svm_y_pred = svm_model.predict(svm_X_test)
# 5. 可视化
```

```
grid_z = svm_y_pred.reshape(grid_x.shape)
plt.figure('SVM')
plt.title('SVM')
plt.pcolormesh(grid_x, grid_y, grid_z, cmap='Blues')
plt.scatter(svm_X_train[:, 0], svm_X_train[:, 1], s=30, c=svm_y_train, cmap='p
plt.show()
```

K-Means

将数据划分到K个聚类簇中，使得每个数据点都属于离它最近的均值（即聚类中心，centroid）对应的集聚类簇。最终，具有较高相似度的数据对象划分至同一类簇，将具有较高相异度的数据对象划分至不同类簇。具体原理参考：用人话讲明白快速聚类kmeans - 化简可得的文章 - 知乎，下面给出代码的实现。




```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets.samples_generator import make_blobs

# K-means 任务为聚类 n_classes=5
from sklearn.cluster import KMeans

# 1. 准备数据
```

```
kmeans_X_data, kmeans_y_data = make_blobs(n_samples=500, centers=5, cluster_st
# 2. 训练模型
kmeans_model = KMeans(n_clusters=5)
kmeans_model.fit(kmeans_X_data)
# 3. 预测模型
kmeans_y_pred = kmeans_model.predict(kmeans_X_data)
# 4. 可视化
plt.figure('K-Means')
plt.title('K-Means')
plt.scatter(kmeans_X_data[:,0], kmeans_X_data[:, 1], s=50)
plt.scatter(kmeans_X_data[:, 0], kmeans_X_data[:, 1], c=kmeans_y_pred, s=50, c
centers = kmeans_model.cluster_centers_
plt.scatter(centers[:,0], centers[:, 1], c='red', s=80, marker='x')
plt.show()
```

PCA

一种常用的降维技术，顾名思义，PCA帮助我们找出数据的主要成分，主成分基本上是线性不相关的向量，用选出的k个主成分来表示数据，来达到降维的目的。具体原理参考：如何通俗易懂地讲解什么是 PCA 主成分分析？- 马同学的回答 - 知乎，下面给出代码实现。

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

# PCA
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris

# 1. 准备数据
pca_data=load_iris()
pca_X_data=pca_data.data
pca_y_data=pca_data.target
```

```
# 2. 训练模型，维度为2
pca_model=PCA(n_components=2)
# 3. 降维
reduced_X=pca_model.fit_transform(pca_X_data)
# 4. 可视化
red_x,red_y=[],[]
blue_x,blue_y=[],[]
green_x,green_y=[],[]

for i in range(len(reduced_X)):
 if pca_y_data[i] ==0:
  red_x.append(reduced_X[i][0])
  red_y.append(reduced_X[i][1])
 elif pca_y_data[i]==1:
  blue_x.append(reduced_X[i][0])
  blue_y.append(reduced_X[i][1])
 else:
  green_x.append(reduced_X[i][0])
  green_y.append(reduced_X[i][1])

plt.figure('PCA')
plt.title('PCA')
plt.scatter(red_x,red_y,c='r')
plt.scatter(blue_x,blue_y,c='b')
plt.scatter(green_x,green_y,c='g')
plt.show()
```

总结

至此，给出了常有的9种机器学习算法的实现，题主可以通过一些实际案例去进一步理解和熟悉算法。国外的Kaggle和阿里云天池都是获取项目经验的好途径。

推荐阅读：

https://zhuanlan.zhihu.com/p/72513104

https://zhuanlan.zhihu.com/p/139122386

https://www.cnblogs.com/pinard/p/6069267.html

https://www.cnblogs.com/pinard/p/6061661.html

https://zhuanlan.zhihu.com/p/26703300

https://zhuanlan.zhihu.com/p/51165358

https://zhuanlan.zhihu.com/p/49331510

https://zhuanlan.zhihu.com/p/75477709

https://www.zhihu.com/question/41120789/answer/481966094

今天的分享就到这里，谢谢大家。

在文末分享、点赞、在看，给个3连击呗~

社群推荐：

欢迎加入 **DataFunTalk 机器学习** 交流群，跟同行零距离交流。**识别二维码**，添加小助手微信，**入群。**

关于我们：

DataFunTalk 专注于大数据、人工智能技术应用的分享与交流。发起于2017年，在北京、上海、深圳、杭州等城市举办超过100场线下沙龙、论坛及峰会，已邀请近600位专家和学者参与分享。其公众号 DataFunTalk 累计生产原创文章300+，百万+阅读，10万+精准粉丝。

DataFunTalk

專注於大數據、人工智能技術應用的分享與交流。致力於成就百萬數據科學家。定期組織技...

459篇原創內容

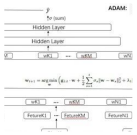
公眾號

分享、點贊、在看，給個**3連擊**唄！

喜歡此內容的人還喜歡

推薦系統之連續值DNN模型

AINLP



深度學習中的捲積你掌握了幾種？一文速覽Deep Learning中的11種卷積

極市平台