Python与金融数据挖掘(15)

文欣秀

wenxinxiu@ecust.edu.cn

聚 类

聚类(Clustering Approach)： 是按一定的距离或相似性系数将数据分成一系列相互区分的组，常用的经典聚类方法有**K-means**, **K-medoids**，**isodata**等。

聚类算法的应用场景： 市场分析、**商业经营**、图像处理、决策支持、模式识别。

数据预处理

归一化： scikit-learn库中**preprocessing. MinMaxScaler**类实现了将数据缩放到指定的最大值和最小值（通常是1-0）之间的功能。

数据标准化： scikit-learn库中**preprocessing. StandardScaler**类实现了Z-Score标准化。

数据正则化： scikit-learn库中**preprocessing. Normalizer**类实现了将单个样本缩放到单位范数的功能。

评价指标

分类常用的评价指标：混淆矩阵 (Confusion Matrix) 、精确率 (Precision)、召回率 (Recall)、F1分数 (F1 Score)和准确率 (Accuracy)等。

回归主要评价指标：平均绝对误差(MAE，Mean absolute error)、均方误差(MSE，Mean squared error)、均方根误差(RMSE，Root Mean squared error)、R^2等。

机器学习分类

K-Means聚类算法

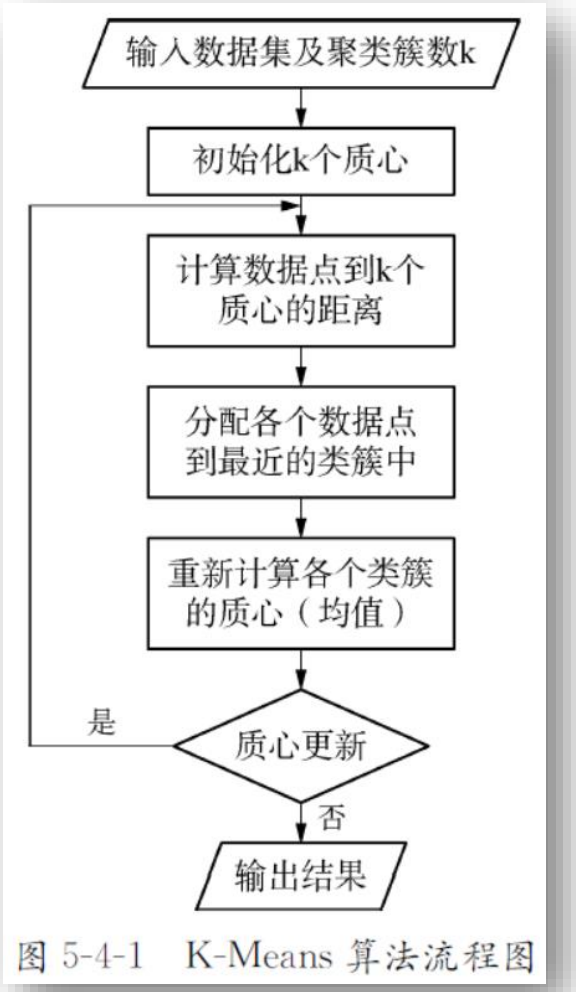


图 5-4-1　K-Means 算法流程图

课堂练习一

假设某类簇包含了数据点[(5, 8),(3, 5),(4, 2)]，则使用K-Means算法计算一次簇的质心为(　　)?

A　(4, 5)

B　(5, 8)

C　(3, 5)

D　(4, 8)

K- Means聚类算法

Scikit-learn的Cluster类提供聚类分析的方法：

- 模型初始化

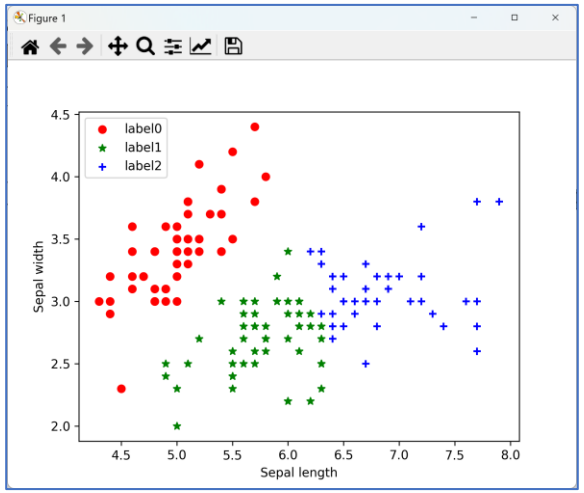
  kmeans=**Kmeans**(n_clusters)　#参数为簇的个数

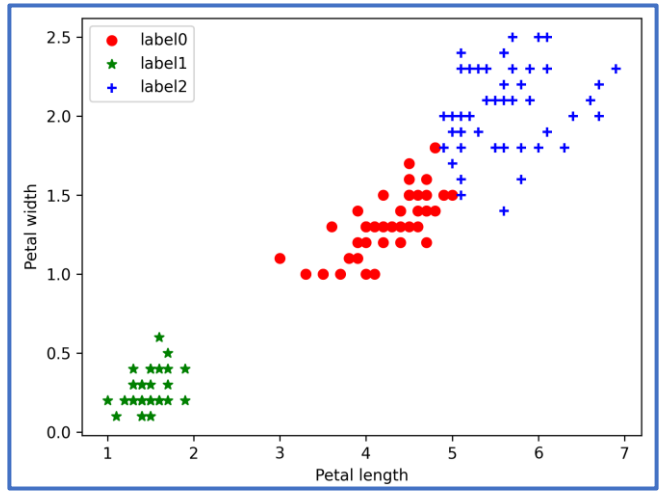
- 模型学习

  kmeans. **fit**(X)　　#参数为样本二维数组

鸢尾花聚类问题

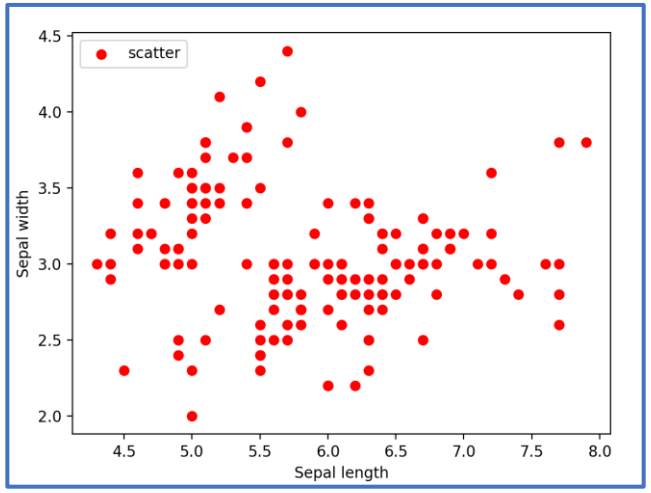
	A	B	C	D
1	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
2	5.1	3.5	1.4	0.2
3	4.9	3	1.4	0.2
4	4.7	3.2	1.3	0.2
5	4.6	3.1	1.5	0.2
6	5	3.6	1.4	0.2
7	5.4	3.9	1.7	0.4
8	4.6	3.4	1.4	0.3
9	5	3.4	1.5	0.2
10	4.4	2.9	1.4	0.2
11	4.9	3.1	1.5	0.1
12	5.4	3.7	1.5	0.2
13	4.8	3.4	1.6	0.2





鸢尾花问题K- Means模型（1）

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
import pandas as pd   #导入模块
iris = pd.read_csv("iris.csv")
X = iris.loc[:,['Sepal_Length', 'Sepal_Width']] #读出数据
plt.scatter(X['Sepal_Length'], X['Sepal_Width'], c = "red", marker='o', label='scatter')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.legend(loc=2)
plt.show()
```

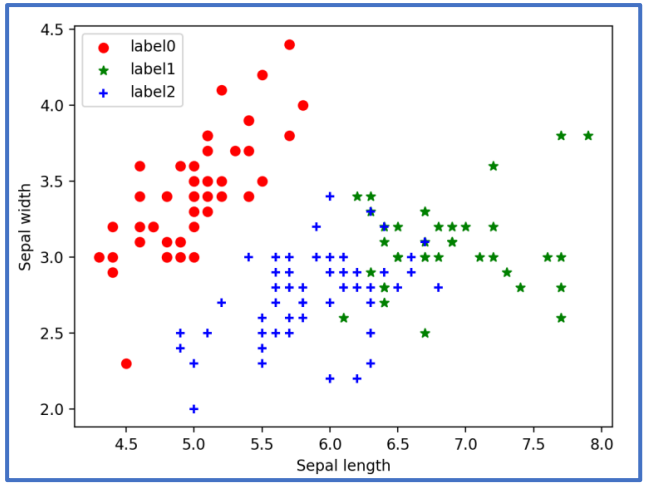


鸢尾花问题K- Means模型（2）

```
estimator = KMeans(n_clusters=3)#模型初始化
estimator.fit(X) #模型学习
label_pred = estimator.labels_ #获取聚类标签
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
print(x0)
print(x1)
print(x2)
```

鸢尾花问题K- Means模型（3）

```
plt.scatter(x0['Sepal_Length'], x0['Sepal_Width'], c = "red", marker='o', label='label0')

plt.scatter(x1['Sepal_Length'], x1['Sepal_Width'], c = "green", marker='*', label='label1')

plt.scatter(x2['Sepal_Length'], x2['Sepal_Width'], c = "blue", marker='+', label='label2')

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.legend(loc=2)

plt.show()
```



鸢尾花数据集获取

```
>>> from sklearn import datasets     # 导入数据集包
>>> dir (datasets)                   # 查看数据集
>>> iris = datasets.load_iris()
>>> X = iris['data']
>>> y = iris['target']
```

导入数据的函数名称	对应的数据集
~~load_boston()~~	~~波士顿房价数据集~~
load_breast_cancer()	乳腺癌数据集
load_iris()	鸢尾花数据集
load_diabetes()	糖尿病数据集
load_digits()	手写数字数据集
load_linnerud()	体能训练数据集
load_wine()	红酒品类数据集

鸢尾花问题K- Means模型（1）

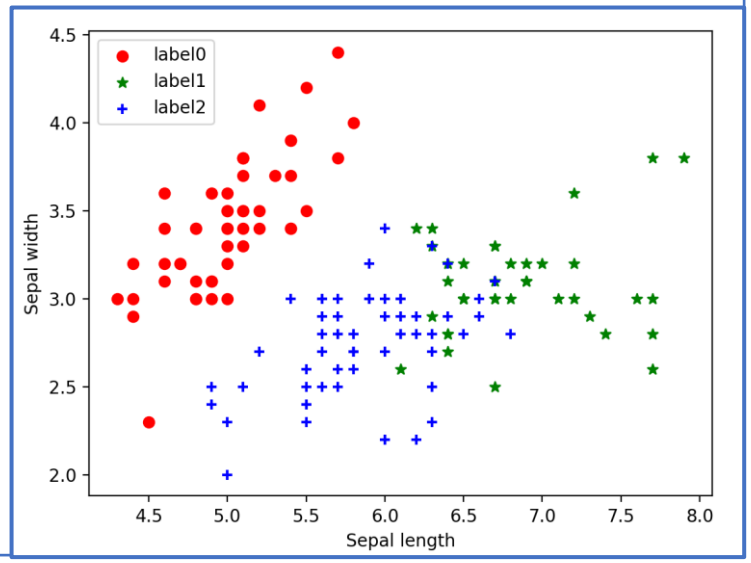
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets      # 导入数据集包
iris = datasets.load_iris()       # 加载数据据集
X = iris['data']                  #读出数据
```

鸢尾花问题K- Means模型（2）

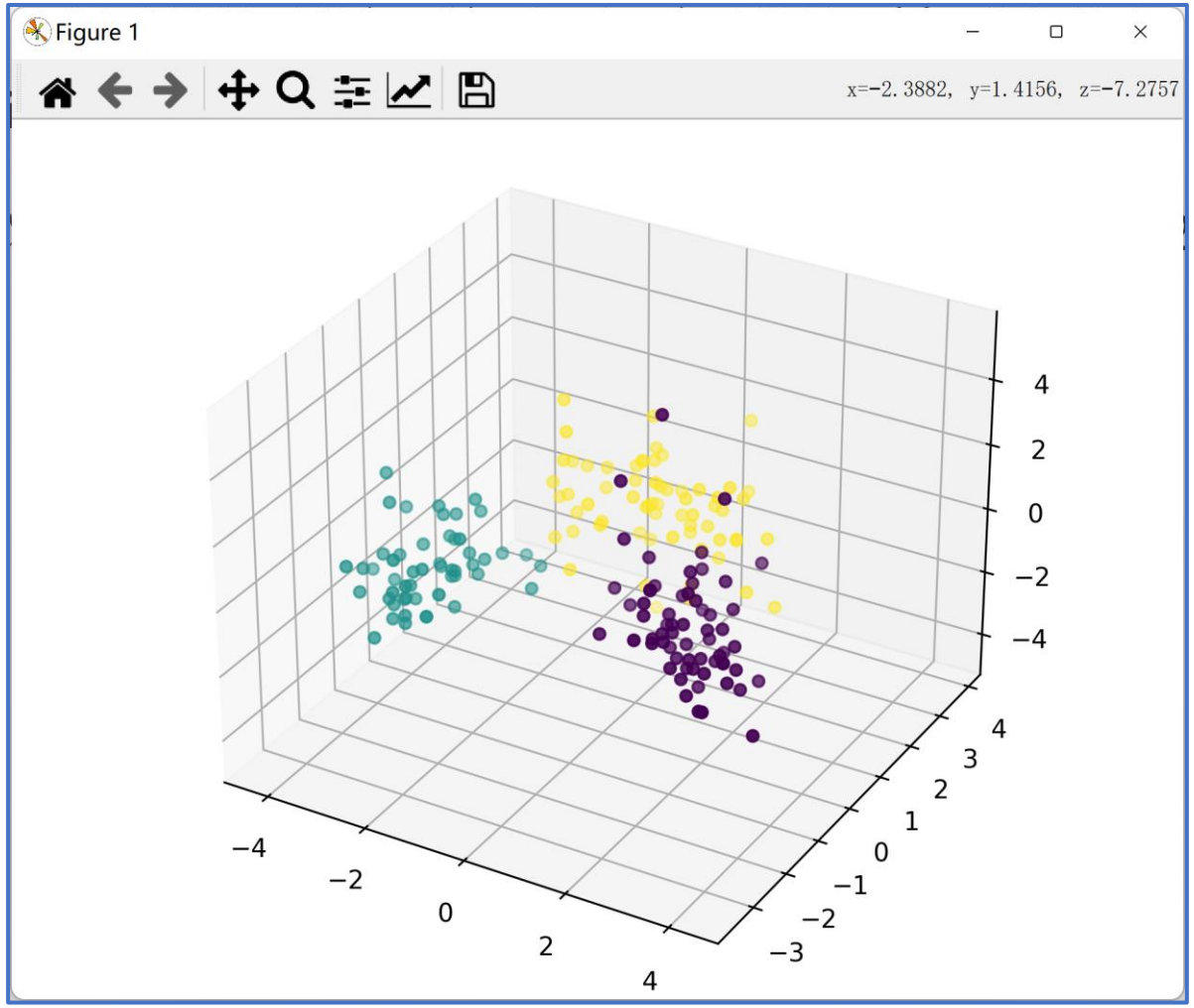
```
estimator = KMeans(n_clusters=3,n_init="auto")#模型初始化
estimator.fit(X) #模型学习
label_pred = estimator.labels_ #获取聚类标签
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
```

鸢尾花问题K- Means模型（3）

```
plt.scatter(x0[:,0], x0[:,1], c = "red", marker='o', label='label0')
plt.scatter(x1[:,0], x1[:,1], c = "green", marker='*', label='label1')
plt.scatter(x2[:,0], x2[:,1], c = "blue", marker='+', label='label2')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.legend(loc=2)
plt.show()
```

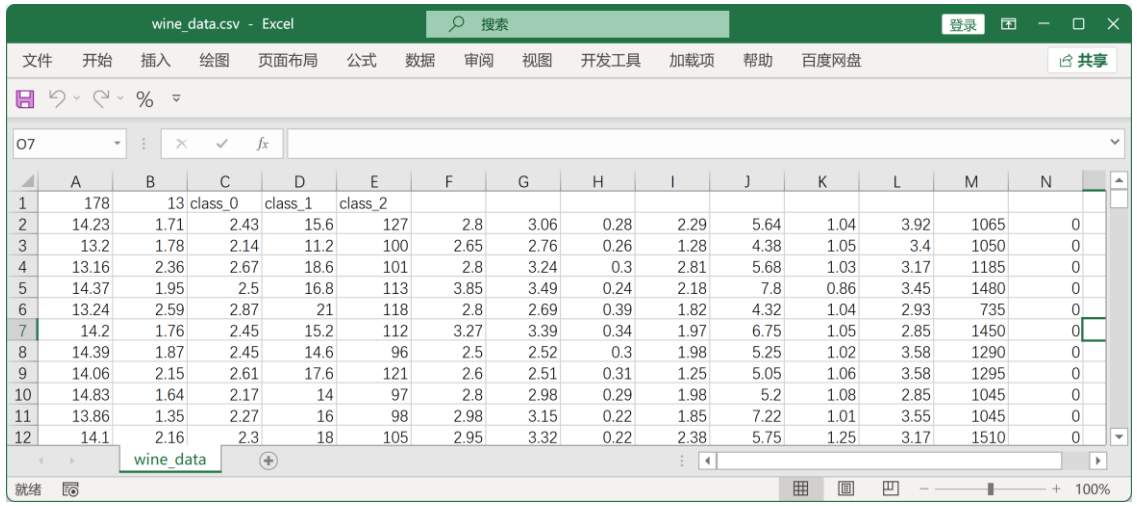


数据降维案例分析



葡萄酒(wine)数据集分析

wine数据集： 该数据集为意大利同一地区生产的三个不同种类的葡萄酒成分数据,每行代表一种酒的样本，共有**178**个样本，每个样本有**13**个特征，分为**3**个类别。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	178	13	class_0	class_1	class_2									
2	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	0
3	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050	0
4	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	0
5	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480	0
6	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735	0
7	14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	0
8	14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290	0
9	14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295	0
10	14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045	0
11	13.86	1.35	2.27	16	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045	0
12	14.1	2.16	2.3	18	105	2.95	3.32	0.22	2.38	5.75	1.25	3.17	1510	0

葡萄酒(wine)数据集特征

13个特征：酒精、苹果酸、灰、灰分的碱度、镁、总酚、黄酮类化合物、非黄烷类酚类、原花色素、颜色强度、色调、稀释葡萄酒的OD280/OD315、脯氨酸。其中第1类有**59**个样本，第2类有**71**个样本，第3类有**48**个样本。

PCA主成分分析技术

PCA主成分分析技术：旨在利用**降维**的思想，把多指标转化为**少数几个综合指标**。

思考：特征降维能够有效降低数据量，wine数据集的13个特征是否都对分类提供帮助？能否进行降维处理？

wine数据集主成分分析PCA（1）

```
from sklearn.cluster import KMeans   #K-Means聚类模型
from sklearn.datasets import load_wine   #wine数据集
from sklearn.decomposition import PCA  #pca降维
from sklearn.preprocessing import scale    #数据标准化
from sklearn.preprocessing import StandardScaler #标准化
from sklearn import metrics  #评估指标
import numpy as np
import matplotlib.pyplot as plt#数据可视化
from mpl_toolkits.mplot3d import Axes3D #三D绘图
```

wine数据集主成分分析PCA（2）

```
#导入数据集
wine = load_wine()
X = wine['data']
y = wine['target']
print("data:")
print(X)
print("target")
print(y)
```

```
data:
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [1.327e+01 4.280e+00 2.260e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
target
[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 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 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 2 2 2 2]
```

wine数据集主成分分析PCA（3）

```
#pca降维
pca = PCA(n_components=3)
X_scale=StandardScaler().fit_transform(X)  #标准化矩阵
X_reduce=pca.fit_transform(scale(X_scale))


#聚类模型
model = KMeans(n_clusters=3)
model.fit(X_reduce)
labels=model.labels_
print(labels)
```

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

```
target
[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 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 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 2 2]
```

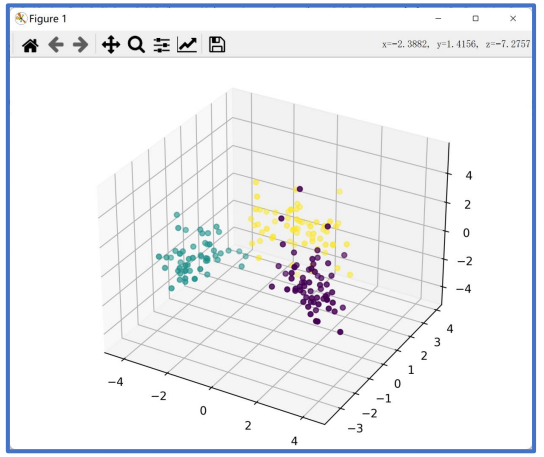
wine数据集主成分分析PCA（4）

```
#输出模型的准确度
print('%.3f  %.3f  %.3f  %.3f  %.3f  %.3f' % \
(metrics.homogeneity_score(y, labels),
 metrics.completeness_score(y, labels),
 metrics.v_measure_score(y, labels),
 metrics.adjusted_rand_score(y, labels),
 metrics.adjusted_mutual_info_score(y, labels),
 metrics.silhouette_score(X_reduce, labels)))
```

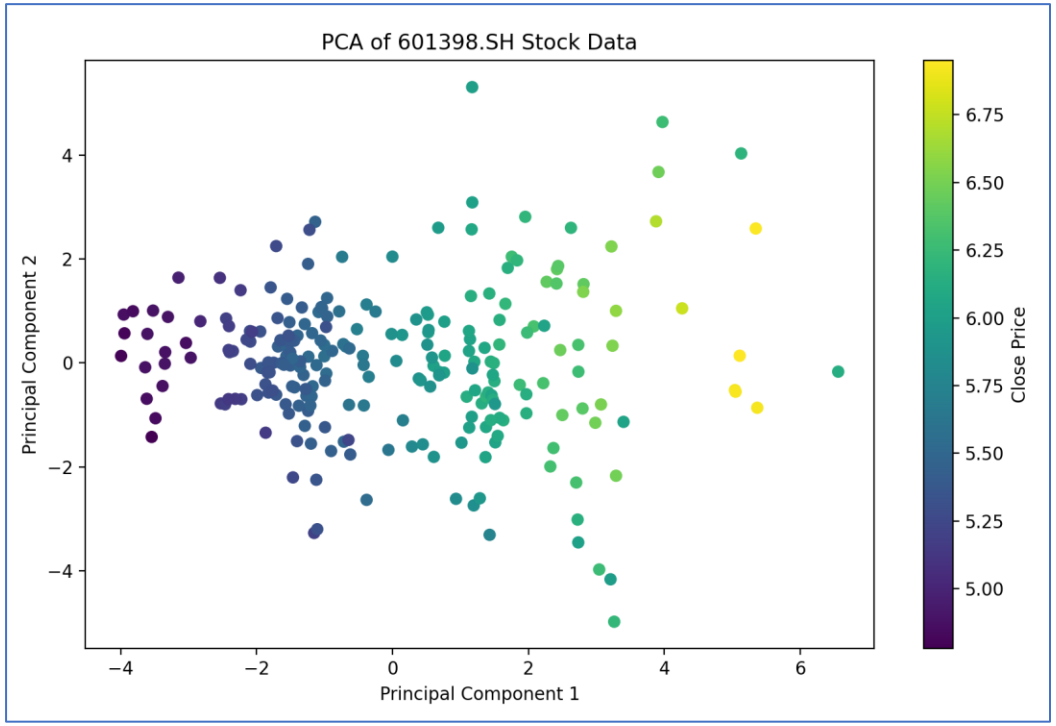
```
0.879  0.873  0.876  0.897  0.875  0.454
```

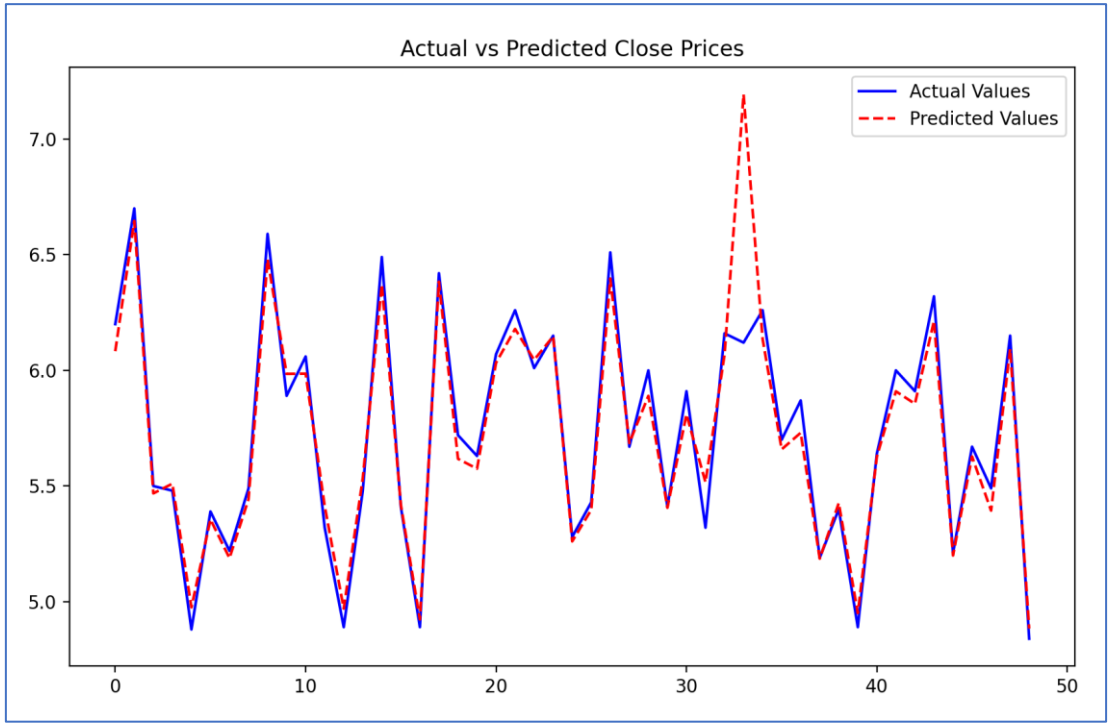
wine数据集主成分分析PCA（5）

```
#绘制模型的分布图
fig=plt.figure()
ax=Axes3D(fig, auto_add_to_figure=False)
fig. add_axes(ax)  #向画布中添加一个轴域
ax.scatter(X_reduce[:, 0], X_reduce[:, 1], X_reduce[:, 2], \
                                    c=labels. astype(np.float64))
plt.show()
```



股票案例分析


PCA of 601398.SH Stock Data
Principal Component 2
Principal Component 1
Close Price
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4
2
0
−2
−4
−4
−2
0
2
4
6


Actual vs Predicted Close Prices
Actual Values
Predicted Values
7.0
6.5
6.0
5.5
5.0
0
10
20
30
40
50

股票案例分析

```
import pandas as pd
import matplotlib.pyplot as plt
import tushare as ts
from sklearn.decomposition import PCA  #pca降维
from sklearn.preprocessing import scale    #数据标准化
from sklearn.preprocessing import StandardScaler #标准化
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

股票案例分析

```
ts.set_token('XXX')    #换成自己的token
pro = ts. pro_api()      #初始化，工商银行
data=pro. daily(ts_code='601398.SH', start_date='2024-01-01',end_date='2025-01-01')
print(data)
features = data.drop(columns=['ts_code','trade_date','close'])
target = data['close']
```

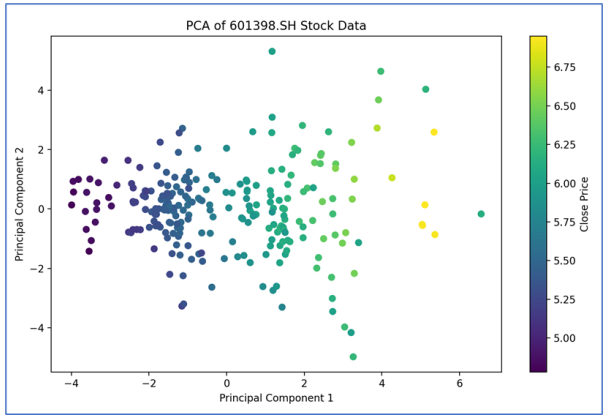
股票案例分析

```
# 标准化数据
scaler = StandardScaler()
data_scaled = scaler.fit_transform(features)

# 应用PCA
pca = PCA(n_components=2)
pca_features = pca.fit_transform(data_scaled)
print(pca_features)
```

```
[[ 5.36911487e+00 -8.62807277e-01]
 [ 5.11107900e+00  1.39344983e-01]
 [ 5.04041064e+00 -5.22256883e-01]
 [ 5.05020821e+00 -5.53802177e-01]
 [ 5.34715574e+00  2.58991086e+00]
 [ 4.26309348e+00  1.05182735e+00]
 [ 3.87955050e+00  2.72780435e+00]
 [ 3.23827940e+00  3.32078607e-01]
```

股票案例分析

```
# 可视化前两个主成分的结果
plt.figure(figsize=(10, 6))
plt.scatter(pca_features[:, 0], pca_features[:, 1], c=target, cmap='viridis')
plt.colorbar(label='Close Price')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of 601398.SH Stock Data')
plt.show()
```



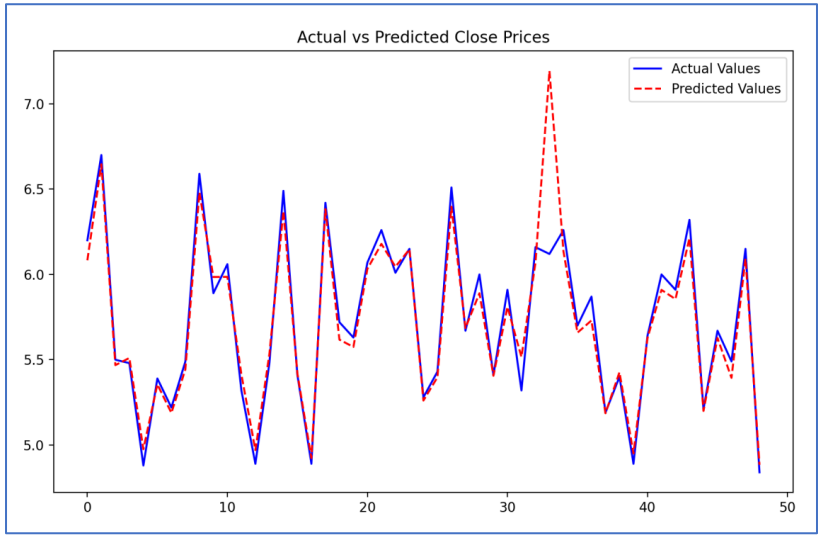
股票案例分析

```
# 分割数据为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(pca_features, target,
test_size=0.2, random_state=42)
# 训练线性回归模型
model = LinearRegression()
model.fit(X_train, y_train)
# 进行预测
y_pred = model.predict(X_test)
print(y_pred)
```

```
[6.08285919 6.64585812 5.46821022
 5.1883753  5.44336601 6.48075226
 4.97020496 5.53355612 6.37154006
 5.61784905 5.57483954 6.03294263
 5.26073512 5.39714427 6.39695377
 5.80966757 5.51660802 6.06830036
 5.73024593 5.18606621 5.42658125
 5.85548611 6.21377059 5.19974354
```

股票案例分析

```
# 评估模型
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# 可视化实际值和预测值的结果
plt.figure(figsize=(10, 6))
plt.plot(y_test.values, label='Actual Values', color='blue')
plt.plot(y_pred, label='Predicted Values', color='red', linestyle='--')
plt.title('Actual vs Predicted Close Prices')
plt.legend(loc = 'upper right')
plt.show()
```



FRED

FRED(Federal Reserve Economic Data，美联储经济数据)：是一个包含广泛经济和金融时间序列数据的平台，它由联邦储备系统中的圣路易斯联邦储备银行维护和运营。

pandas_datareader：用于从远程数据源读取数据的库。

>>>pip install pandas_datareader

案例分析

```
import pandas_datareader.data as pdr
import datetime as dt
# 设置开始和结束日期
start = dt.datetime(2024, 1, 1)
end = dt.datetime(2025, 1, 1)
# 获取数据
df = pdr.DataReader('GDP', 'fred', start, end)
print(df)
```

```
                 GDP
DATE
2024-01-01  28624.069
2024-04-01  29016.714
2024-07-01  29374.914
2024-10-01  29723.864
2025-01-01  29976.638
```

get_data_fred ()函数

get_data_fred ()函数: pandas_datareader 库的一部分，可以方便地获取各种经济指标和金融数据，如利率、通货膨胀率、就业数据等。

get_data_fred ()函数可以用于从 FRED(Federal Reserve Economic Data,美联储经济数据)数据库获取金融和经济数据。

案例分析

```
import pandas_datareader as pdr
from datetime import datetime
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

案例分析

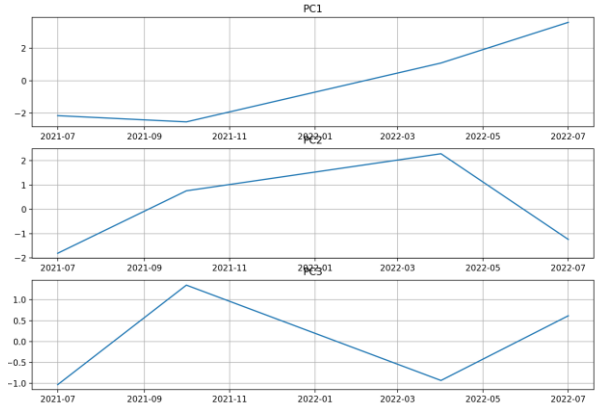
```
# 获取宏观经济指标
start_date = datetime(2000, 1, 1); end_date = datetime(2023, 1, 1)
macro_data = pdr.get_data_fred([
    'GDPC1',      # 实际GDP
    'UNRATE',     # 失业率
    'CPIAUCSL',   # CPI
    'FEDFUNDS',   # 联邦基金利率
    'M2SL',       # 货币供应量M2
    'T10Y2Y',     # 10年-2年国债利差
    'HOUST',      # 新屋开工
    'INDPRO',     # 工业生产指数
    'DCOILWTICO', # 原油价格
    'SP500'       # 标普500指数
], start_date, end_date).dropna()
```

案例分析

```
# 计算年变化率
macro_returns = macro_data.pct_change(12).dropna()
# 标准化数据
scaler = StandardScaler()
scaled_macro = scaler.fit_transform(macro_returns)
# 应用PCA
pca_macro = PCA(n_components=3)
principal_components = pca_macro.fit_transform(scaled_macro)
```

案例分析

```
# 创建DataFrame用于分析
pc_df = pd.DataFrame(principal_components,index=macro_returns.index,
columns=['PC1', 'PC2', 'PC3'])
# 可视化主成分时间序列
plt.figure(figsize=(12, 8))
for i, col in enumerate(pc_df.columns):
    plt.subplot(3, 1, i+1); plt.plot(pc_df[col]); plt.title(col); plt.grid(True)
plt.show()
```



案例分析

```
# 分析载荷（通常指金融杠杆)
loadings = pd.DataFrame(pca_macro.components_,
                        columns=macro_data.columns,
                        index=['PC1', 'PC2', 'PC3'])
print("Macroeconomic PCA Loadings:"); print(loadings)
# 可视化载荷
sns.heatmap(loadings, annot=True, cmap='coolwarm', center=0)
plt.title('Macroeconomic Indicators PCA Loadings')
plt.show()
```

强化学习

强化学习：是在无预先给定任何数据情况下，通过环境对其动作的反馈，不断训练模型，从而获得可以执行某项具体任务的算法。

应用案例：在AlphaGo、游戏、智能机器人中的应用。

半监督学习

半监督学习：是一种监督学习和无监督学习相结合的算法，通过利用**少量有标签的数据集**和**大量无标签的数据集**进行模型的训练。在获得标注数据比较困难而非标注信息较容易的应用场景下（比如医学数据资料），半监督学习可以取得较好的效果。

Python应用领域

文本分析： Jieba、Nltk…

科学计算： Numpy、SciPy…

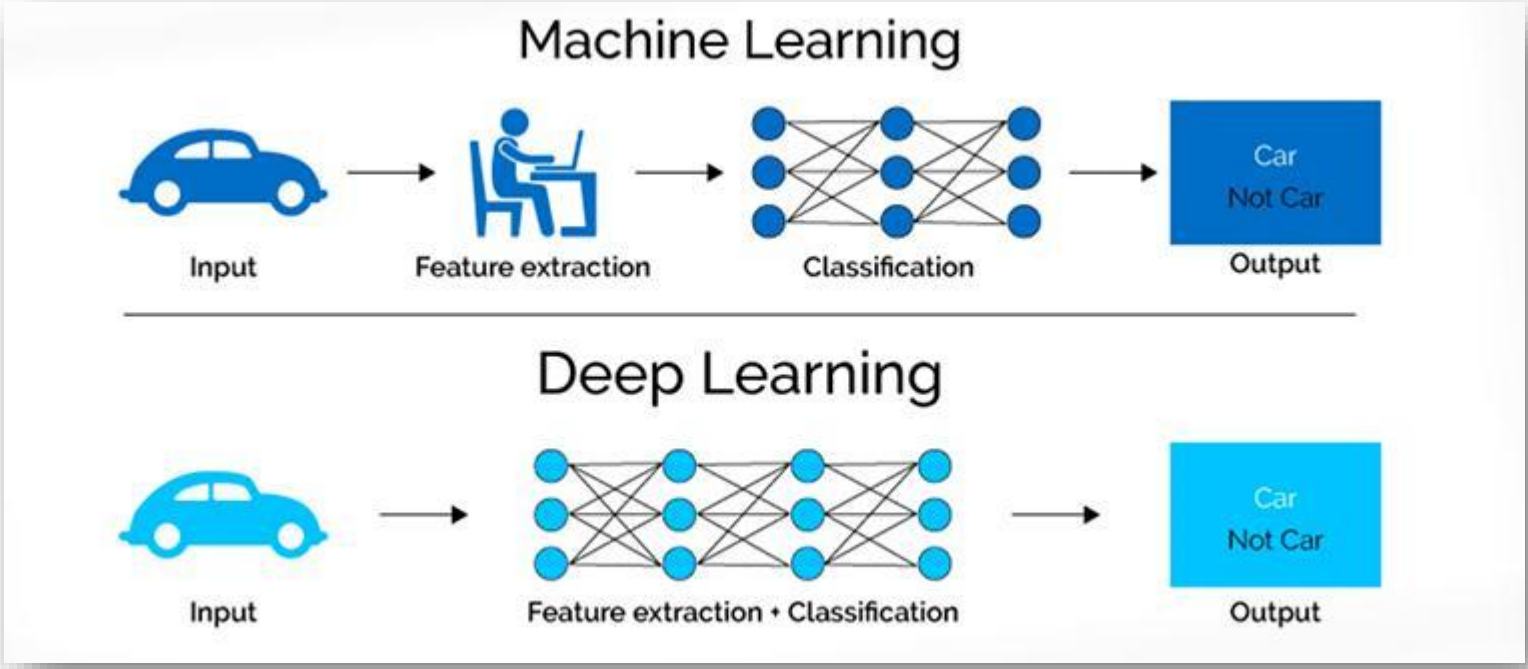
数据分析： Pandas、Matplotlib…

机器学习： Scikit-Learn、Keras…

深度学习： Tensorflow、Mindspore…

机器学习与深度学习


人工智能
能够感知、推理、行动和适应的程序
机器学习
能够随着数据量的增加不断改进性能的算法
深度学习
机器学习的一个子集：利用多层神经网络从大量数据中进行学习


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

常用的深度学习框架


TensorFlow
Keras
Chainer
Marvin
neon
tiny-dnn
Microsoft
CNTK
Caffe
USI/SUPSI
IDSIA
ML
CoreML
Torch
Paddle
DL4J
ONNX
theano
BigDL
APACHE
Spark
Caffe2
dy/net
PYTORCH
mxnet

常用的深度学习框架

TensorFlow：谷歌公司开发的一个开源机器学习平台

PyTorch：由Facebook团队开发的开源深度学习框架

PaddlePaddle：由百度推出的一款开源的深度学习平台

MindSpore：由华为推出的开源新一代全场景AI计算框架

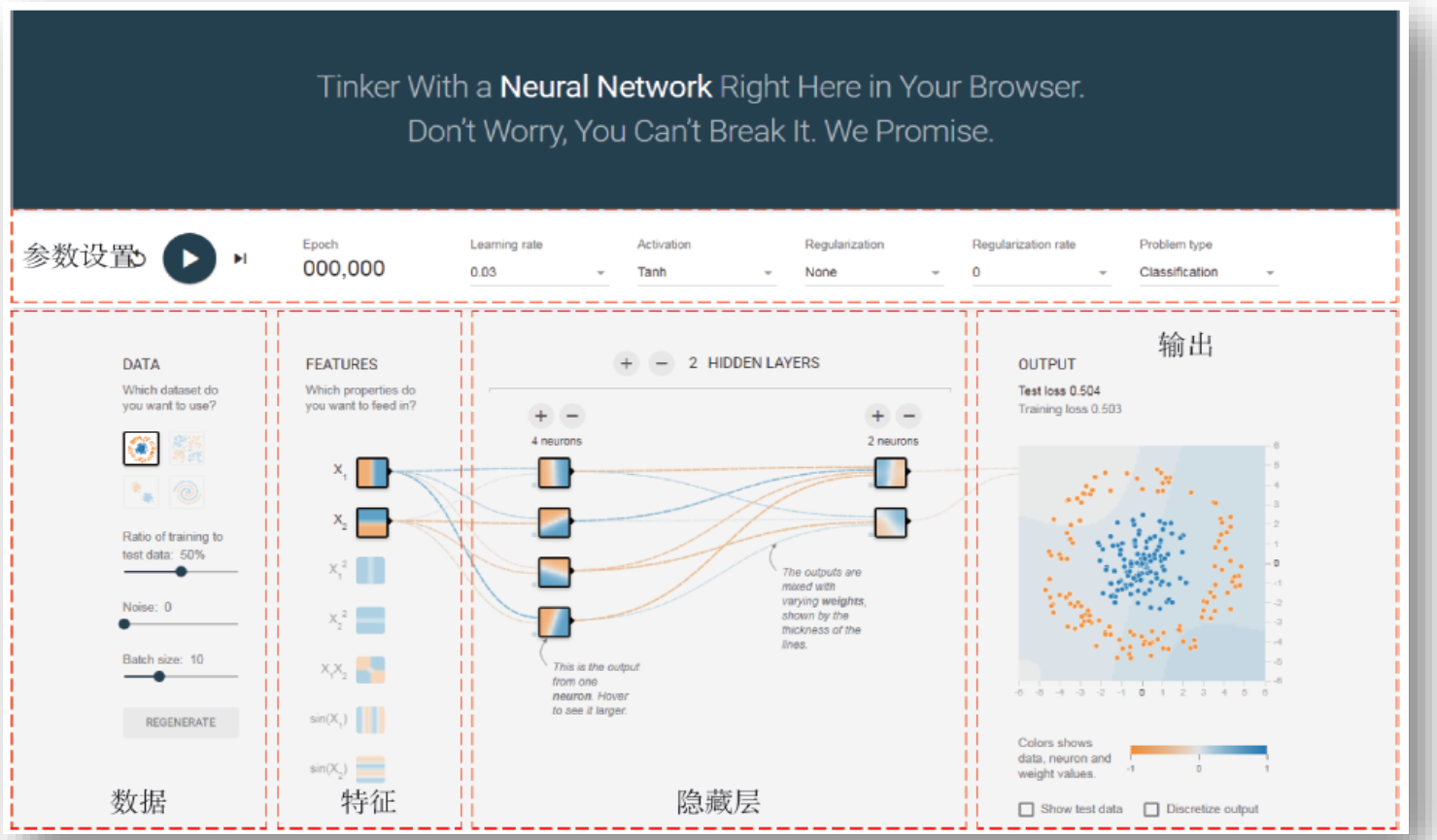
TensorFlow游乐场

- **TensorFlow游乐场**由**谷歌**公司开发，是一款通过浏览器进行神经网络训练的图形化在线工具，**支持训练过程和结果的可视化展示**。

- 通过对网页浏览器的简单操作，用户**针对不同数据按照自己想法训练神经网络**，训练过程和结果可以图像化形式显示出来，获得直观的体验，更加容易理解神经网络的训练流程和复杂原理。

TensorFlow游乐场简介

网址：http://playground.tensorflow.org



TensorFlow游乐场实例

图 6-1-2　圆形数据的分类结果

TensorFlow游乐场实例

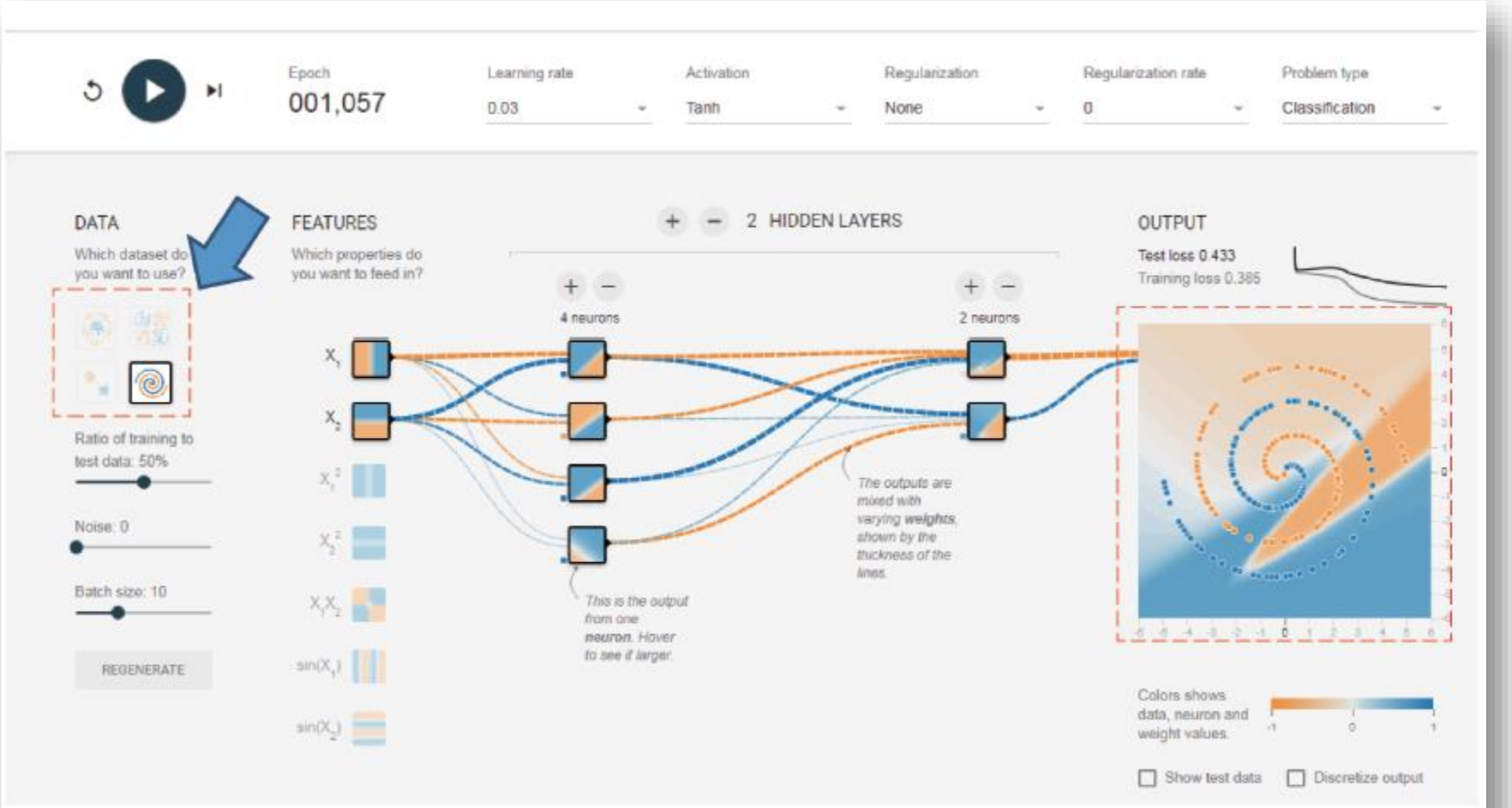
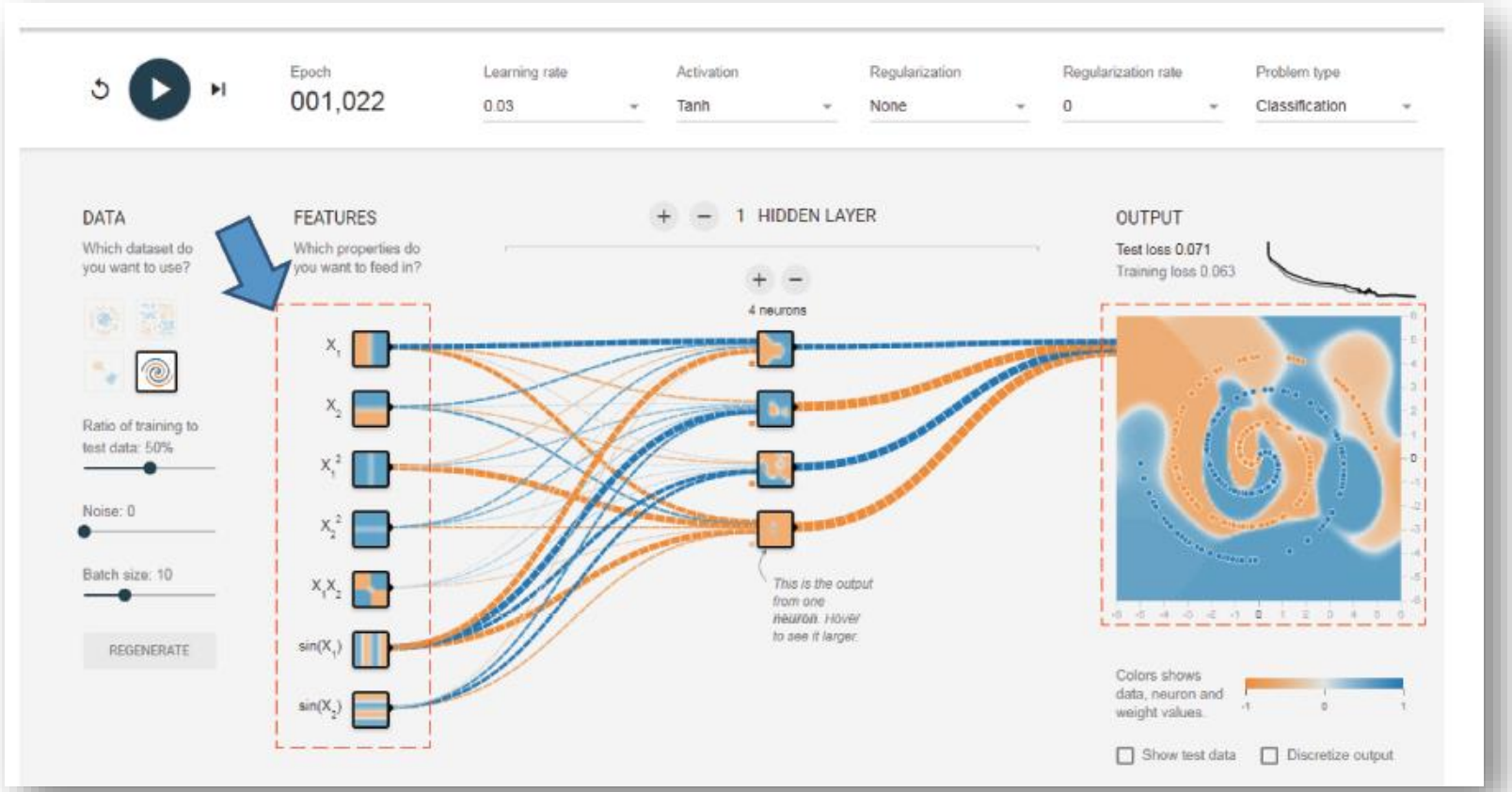


图 6-1-3　螺旋型数据的分类结果

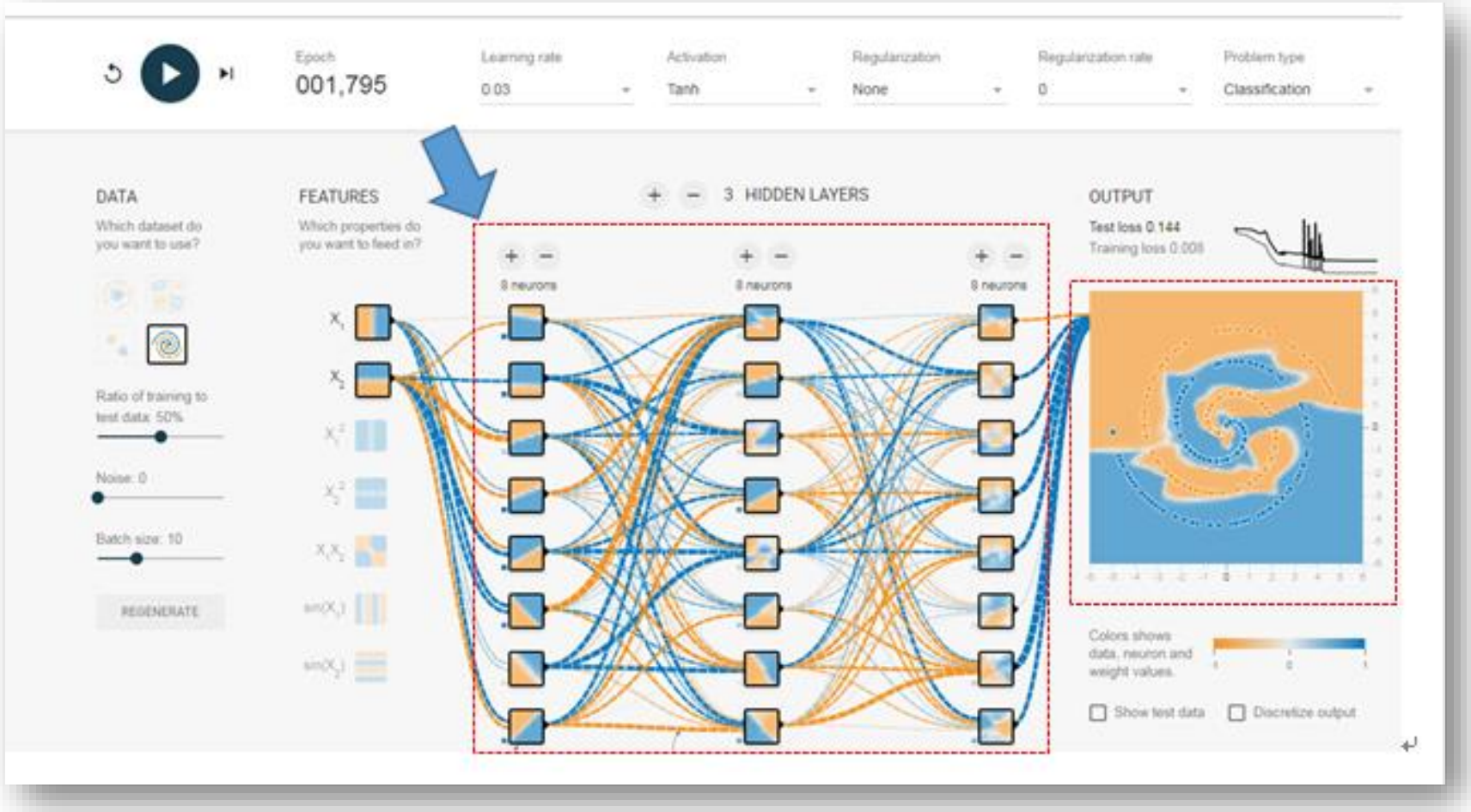
TensorFlow游乐场实例

针对复杂数据的改善方法：增加输入特征的数量



TensorFlow游乐场实例

针对复杂数据的改善方法：增加隐藏层的层数和每层神经元的数目



课堂练习二

神经网络模型如果想获得更好的分类效果，可以通过以下(　　) 手段?

A 增加数据集的输入特征的数量

B 减少每层神经元的数目

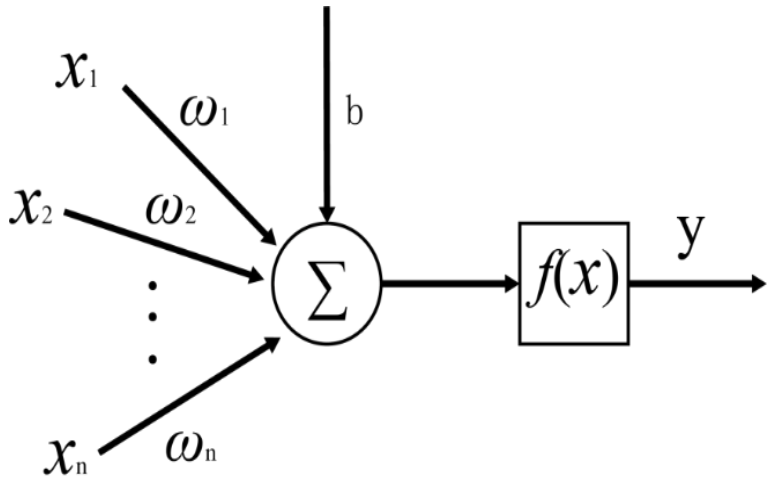
C 减少隐藏层的层数

D 其它三项都不行

神经网络基本原理

神经元模型

- 人工神经元的数学模型



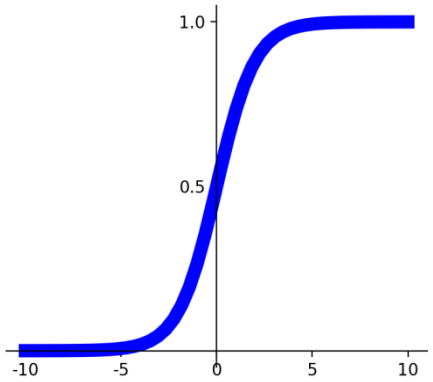
- 该模型的数学表达式为：

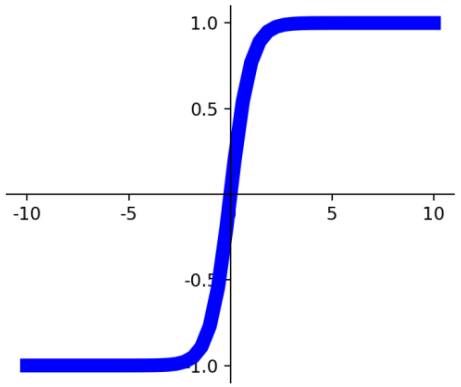
$$y = f\left(\sum_{i=1}^{n} \omega_i x_i + b\right)$$

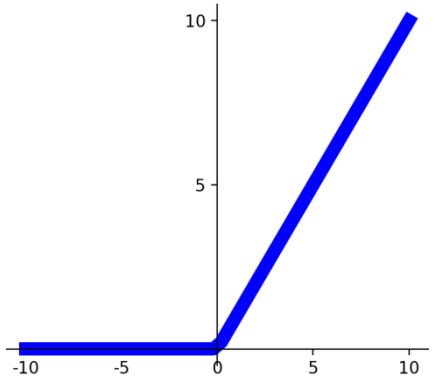
其中，$x_i(i=1,2,\cdots,n)$表示第 i 个输入信号；$\omega_i(i=1,2,\cdots,n)$表示第 i 个输入信号的权重；b 表示偏差（bias），用以增强神经元的数据拟合能力；f 表示激活函数（activation function）；y 为最终的输出结果。

激活函数

激活函数*f*：用于实现神经元的非线性计算，从而增强其表达能力，神经网络中常用的激活函数有Sigmoid函数、tanh函数和ReLU函数等。





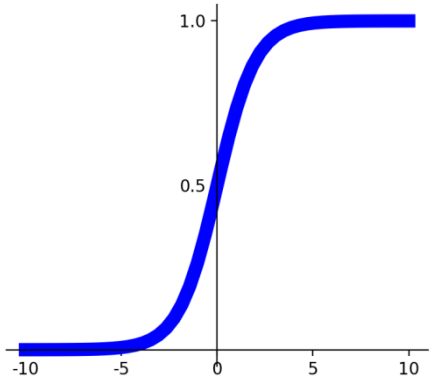


Sigmoid函数

- 计算公式：

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

- 函数图形：

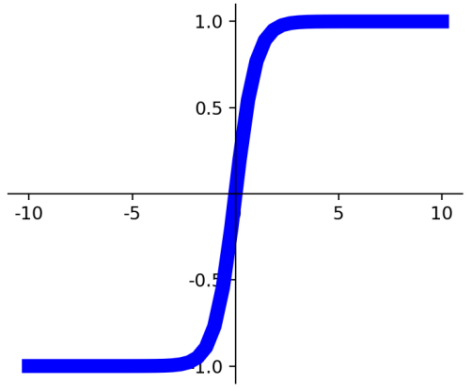


tanh函数

- 计算公式：

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

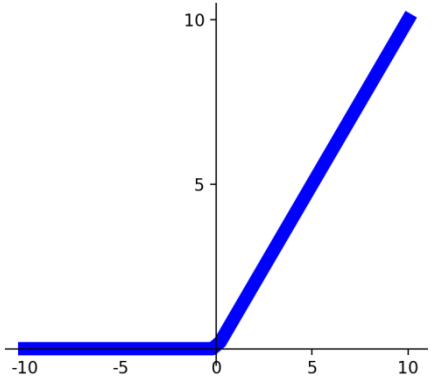
- 函数图形：



ReLU函数

- 计算公式：$f(x) = max(0, x)$

- 函数图形：

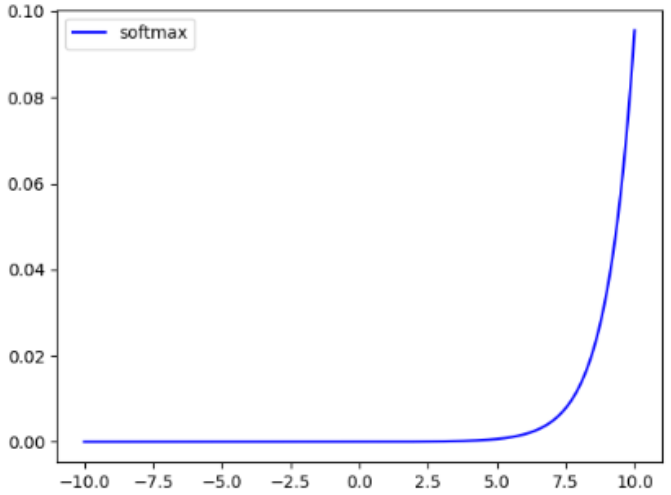


Softmax函数

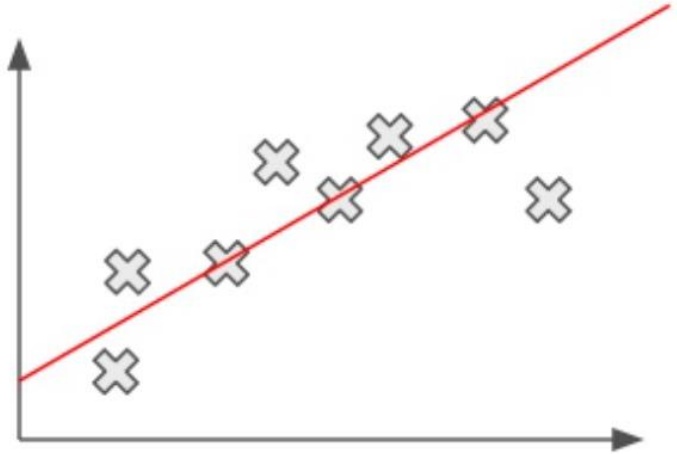
- 计算公式：

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

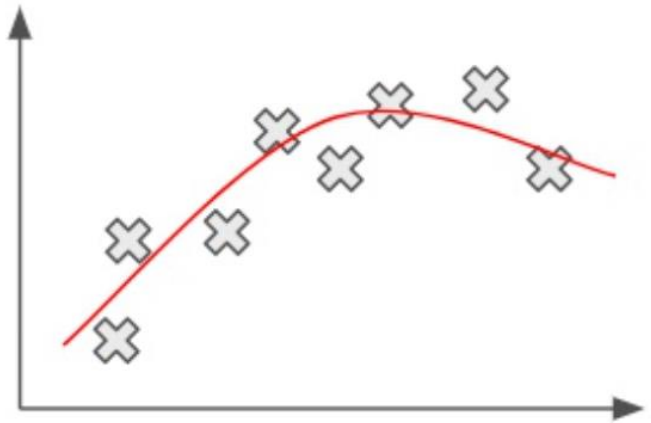
- 函数图形：



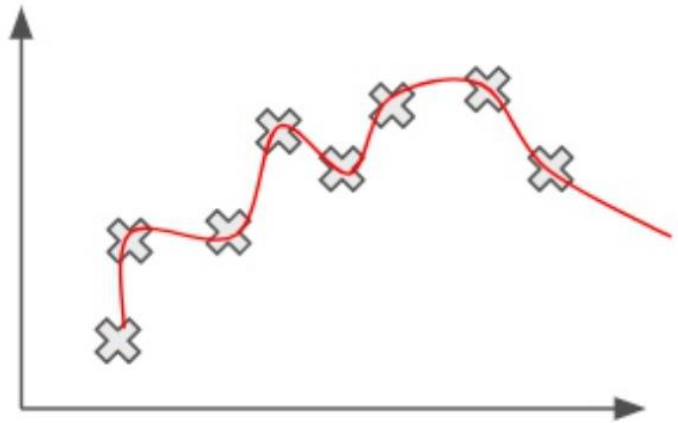
拟合问题分析



欠拟合



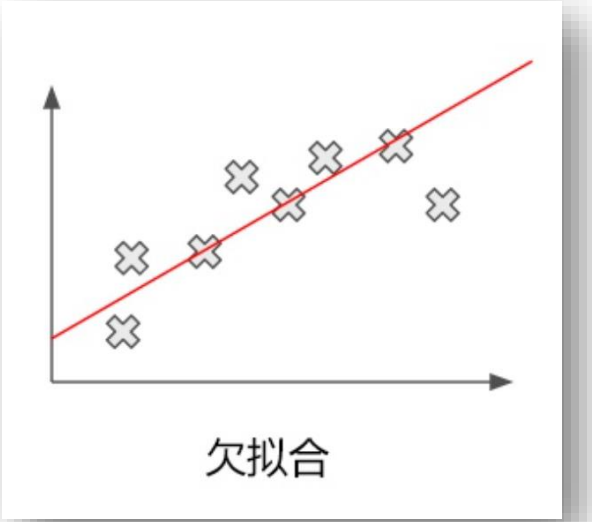
恰当拟合



过拟合

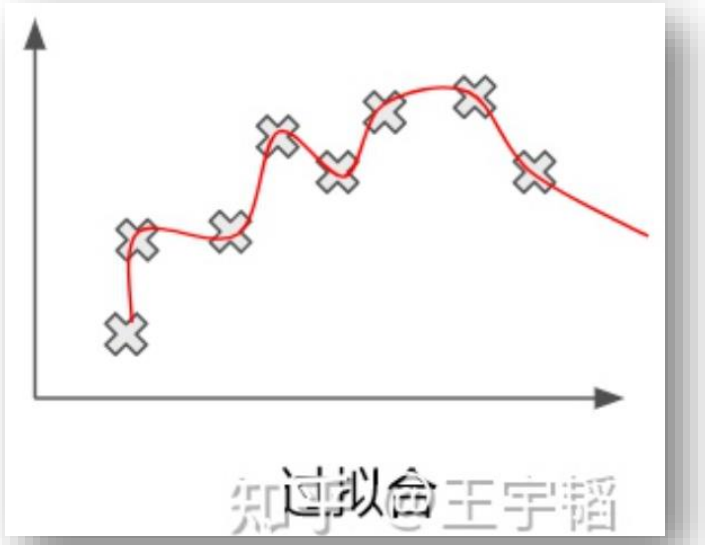
欠拟合问题

欠拟合： 是指模型拟合程度不高，数据距离拟合曲线较远，或指**模型没有很好地捕捉到数据特征**，不能够很好地拟合数据。



过拟合问题

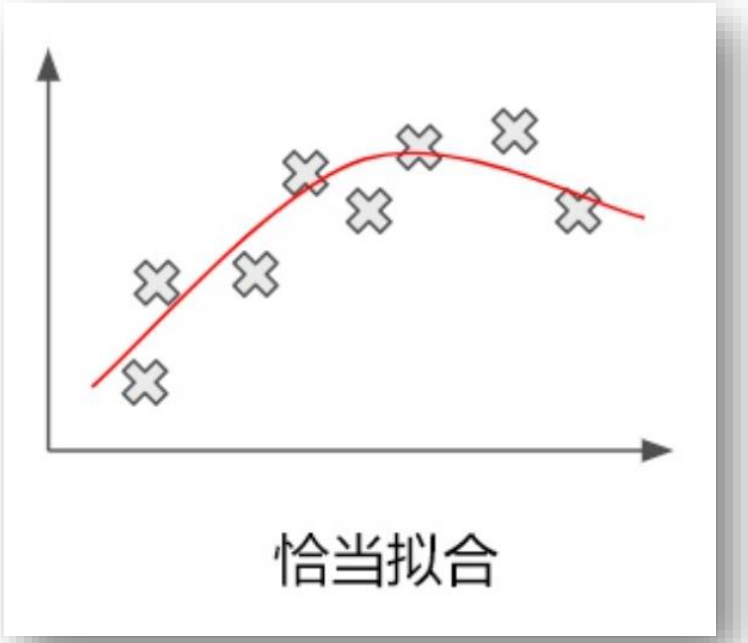
过拟合：模型在训练集上表现很好，但在测试集中训练效果较差（即换了**训练集以外的数据就达不到较好的预测效果**），导致在测试数据集中表现不佳。



适度拟合

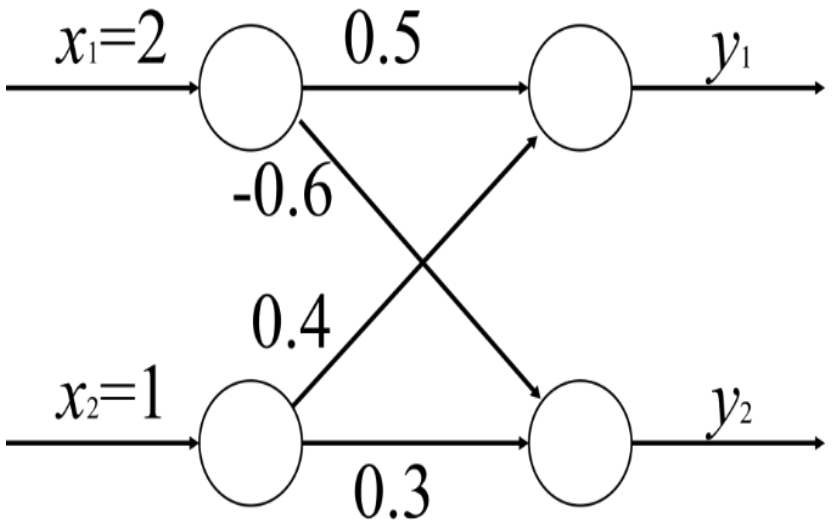
适度拟合： 模型不仅在训练集中可以取得较好拟合效果，

而且对测试集的新数据也能取得不错的效果。



神经元与感知器

- 感知器模型是由多个神经元构成的单层前馈神经网络



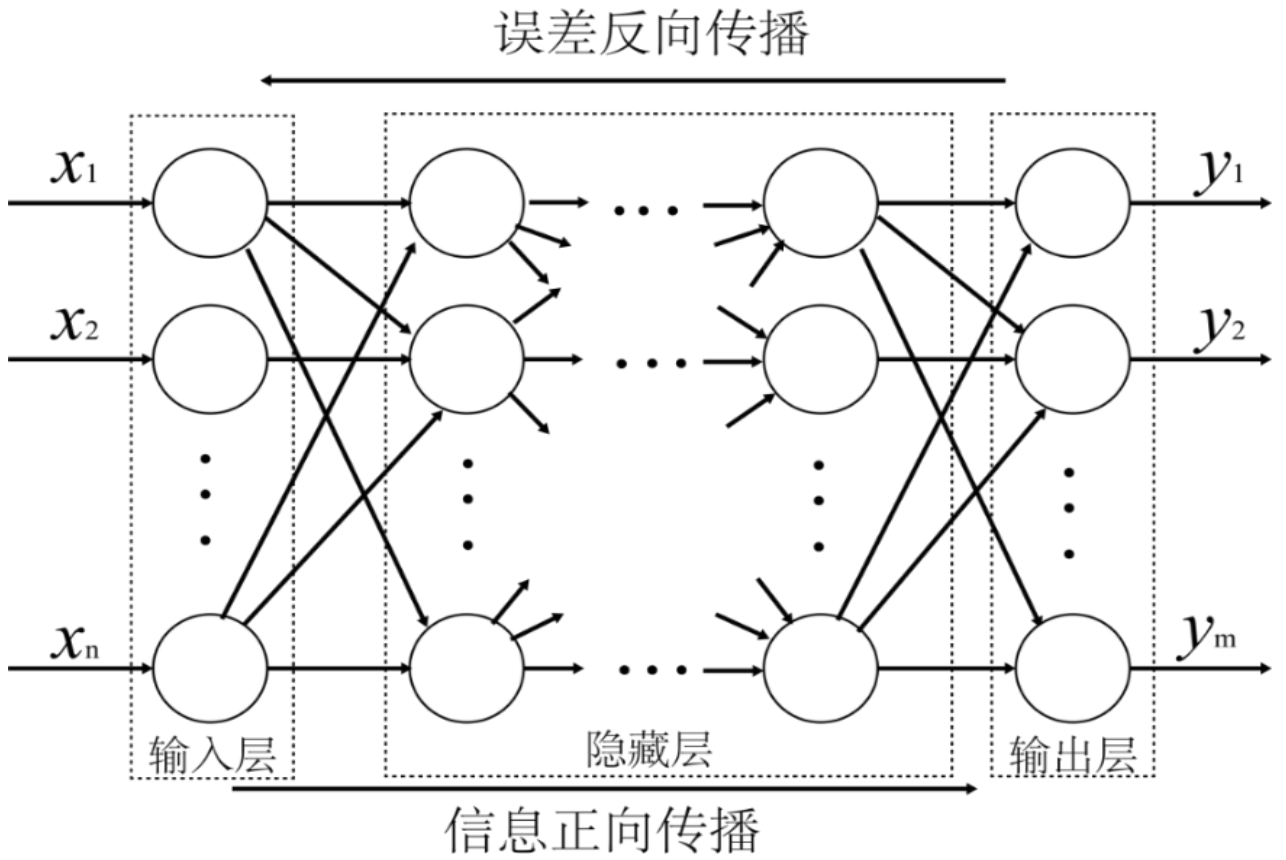
y_1=f(2*0.5+1*0.4)=f(1.4)

y_2=f(2*(-0.6)+1*0.3)=f(-0.9)

- 再通过选择的激活函数$f(x)$，计算y_1和y_2最终的结果

BP神经网络

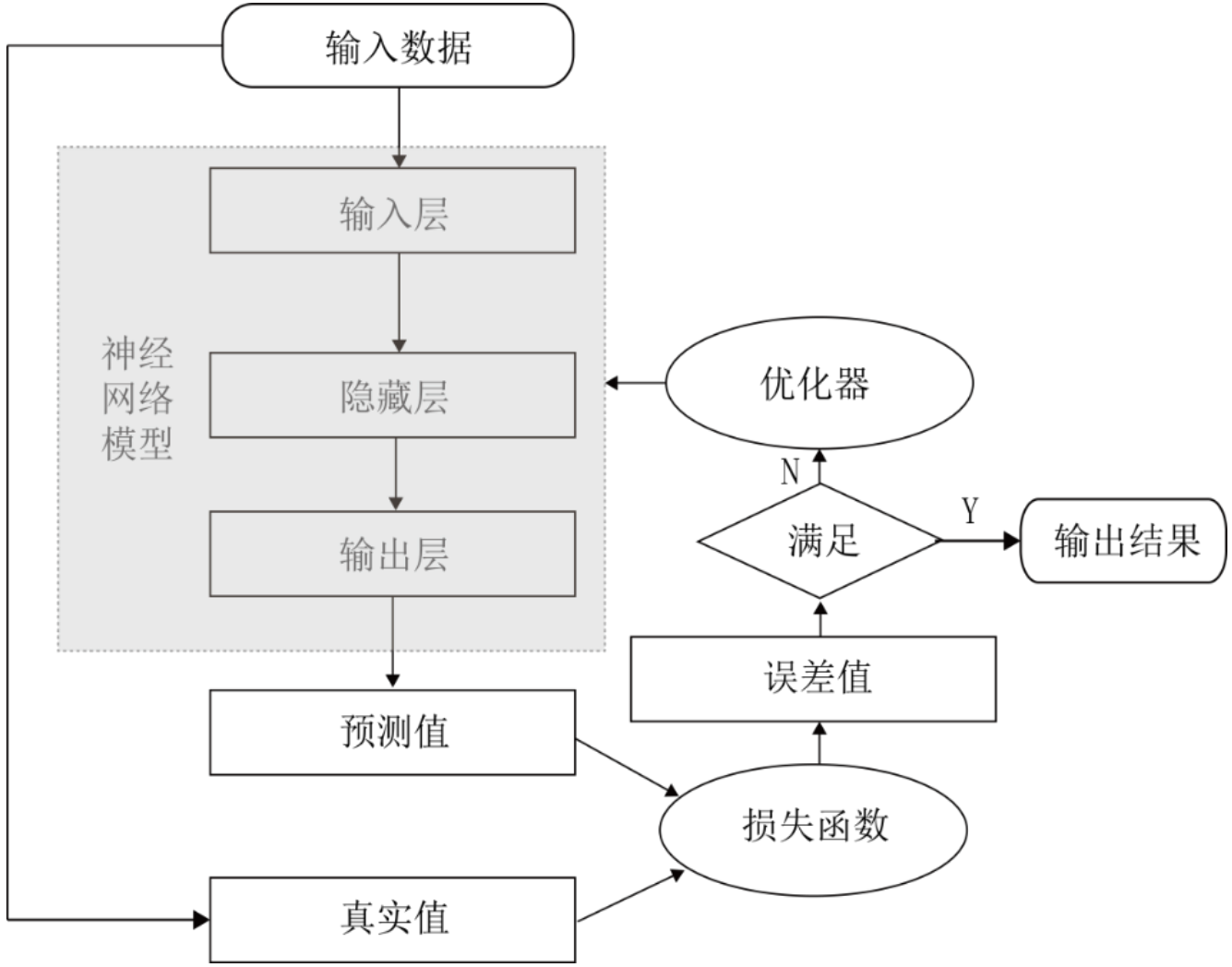
- **BP(Back Propagation)神经网络：** 指多层前馈神经网络



BP神经网络

- 若BP神经网络总共有k层，则**第1层为输入层**（Input Layer），**第k层为输出层**（Output Layer），其它中间各层统称为**隐藏层**（Hidden Layer）。
- 假设输入为$(x_1, \ldots, x_n)$，总共n个信号，即输入层有n个神经元。经过k-2个隐藏层，得到输出层$(y_1, \ldots, y_m)$，总共m个输出信号，即输出层总共有m个神经元。那么BP神经网络可以看作是从**n个输入信号到m个输出信号**的非线性映射。

BP神经网络算法流程



BP神经网络

- **BP算法核心思想：** 通过误差值对神经网络模型的参数进行动态调整，由正向传播和反向传播两次传播方法组成：
- **正向传播：** 输入信号通过输入层进入网络，顺序经过每个隐藏层，最后产生输出信号（**预测值**）；
- **反向传播：** 根据正向传播后得到的误差值（**预测值与真实值之差**），从输出层经过隐藏层最终直到输入层反向传播更新神经网络模型中的参数，利用**梯度下降法动态调整各层神经元参数后**使损失函数最小。

BP神经网络

- **特征值：**用于神经网络模型的训练
- **标签值（真实值）：**用于损失函数计算误差值
- **损失函数：**利用模型的预测值和数据集的真实值计算误差值
- **优化器：**在模型训练的过程中不断更新模型的参数(权重和偏差)，从而使误差值最小，并获得使误差值最小的权重和偏差。

TensorFlow与Keras

- **TensorFlow**可以高效地在CPU、GPU或TPU上执行张量计算，方便用户开发和训练机器学习模型。

- **Keras**是一个对用户友好、用Python编写的高度模块化的神经网络库，随着TensorFlow 2.0的推出，Keras作为其高阶API已经集成其中。

TensorFlow安装

- pip install tensorflow

```
>>> import tensorflow as tf

>>> tf.__version__
```

Keras

Keras： 是TensorFlow 2.0的高阶API，可以只用几行代码就能方便快速地构建神经网络模型。

Keras： 核心数据结构是**model**（模型），用以组织网络层，其中**Sequential**模型（序贯模型）是最简单的模型。

Keras构建神经网络模型步骤

- 载入数据
- 数据预处理
- 构建Sequential模型
- 利用compile函数进行编译
- 利用fit函数训练模型
- 模型的评估和对新数据的预测

谢　谢