

大数据分析 with 知识发现实验报告

西安交通大学



绘制神经元与深度学习算法

自动化 94--胡欣盈--2194323176

目 录

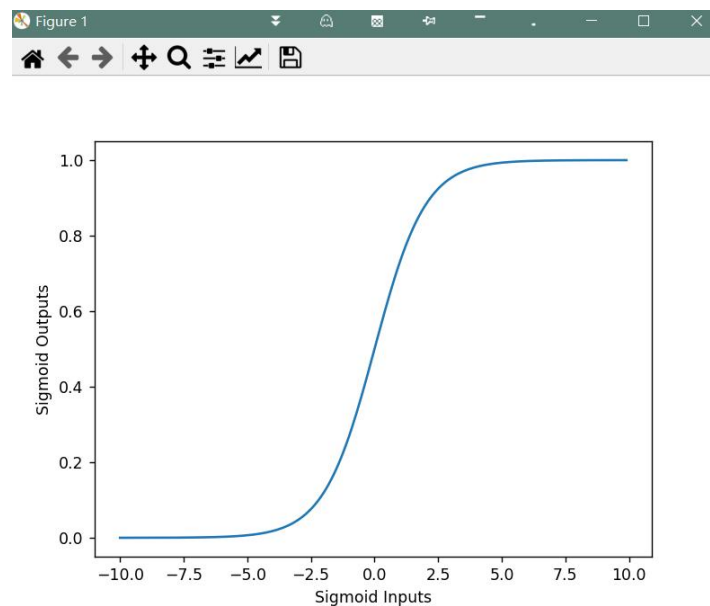
一、绘制神经元	3
二、调用 sklearn 包，完成分类任务	5
(1) 数据集	5
(2) K-均值聚类	6
(3) 聚合聚类	7
三、深度神经网络学习案例	8

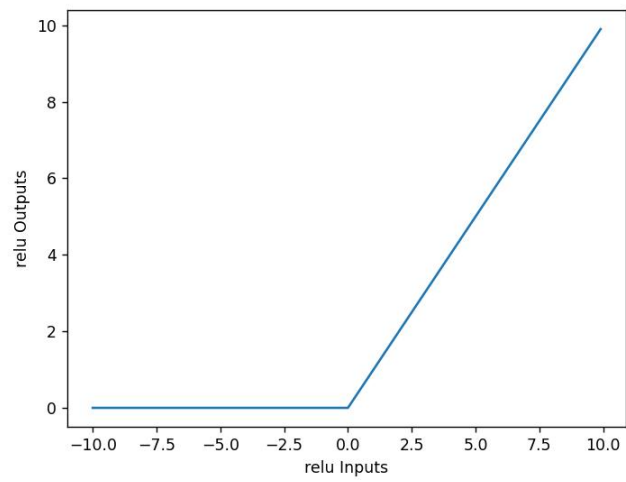
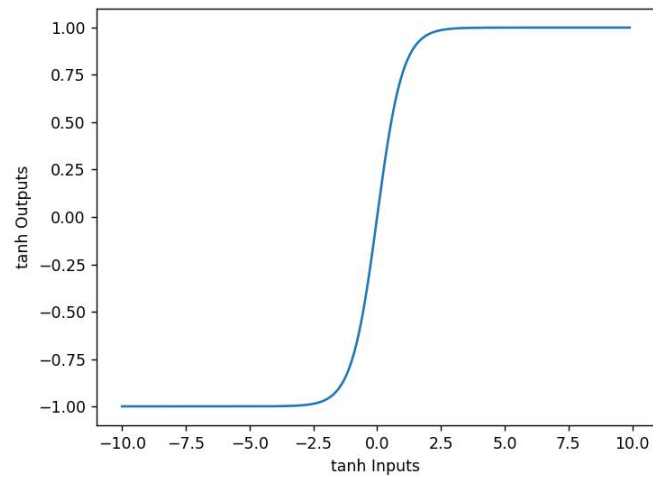
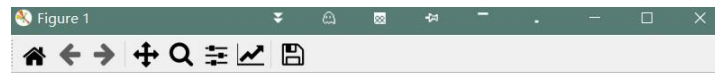
一、绘制神经元

各神经元函数定义如下图所示：

```
main.py
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def sigmoid(x):
5     return 1.0 / (1 + np.exp(-x))
6
7 def tanh(x):
8     return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))
9
10 def relu(x):
11     return np.maximum(0, x)
12
13 sigmoid_inputs = np.arange(-10, 10, 0.1)
14 sigmoid_outputs = sigmoid(sigmoid_inputs)
15 print("Sigmoid Function Input :: {}".format(sigmoid_inputs))
16 print("Sigmoid Function Output :: {}".format(sigmoid_outputs))
17
18 plt.plot(sigmoid_inputs, sigmoid_outputs)
19 plt.xlabel("Sigmoid Inputs")
20 plt.ylabel("Sigmoid Outputs")
21 plt.show()
```

绘制好的神经元函数如下：

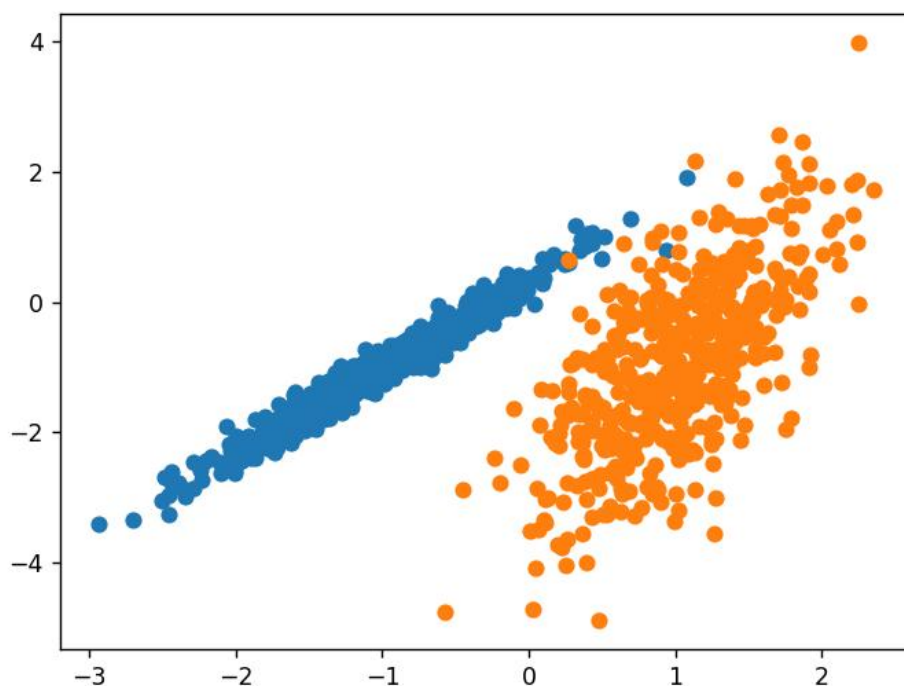
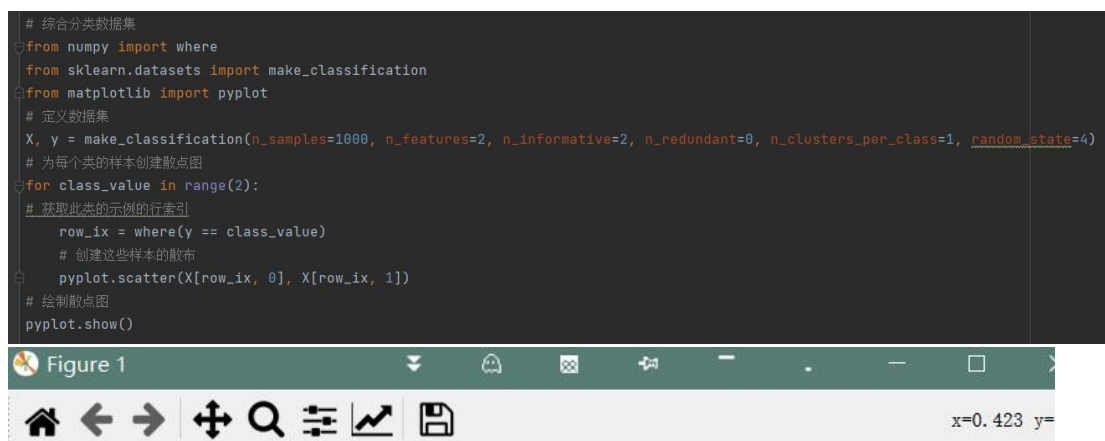




二、调用 sklearn 包，完成分类任务

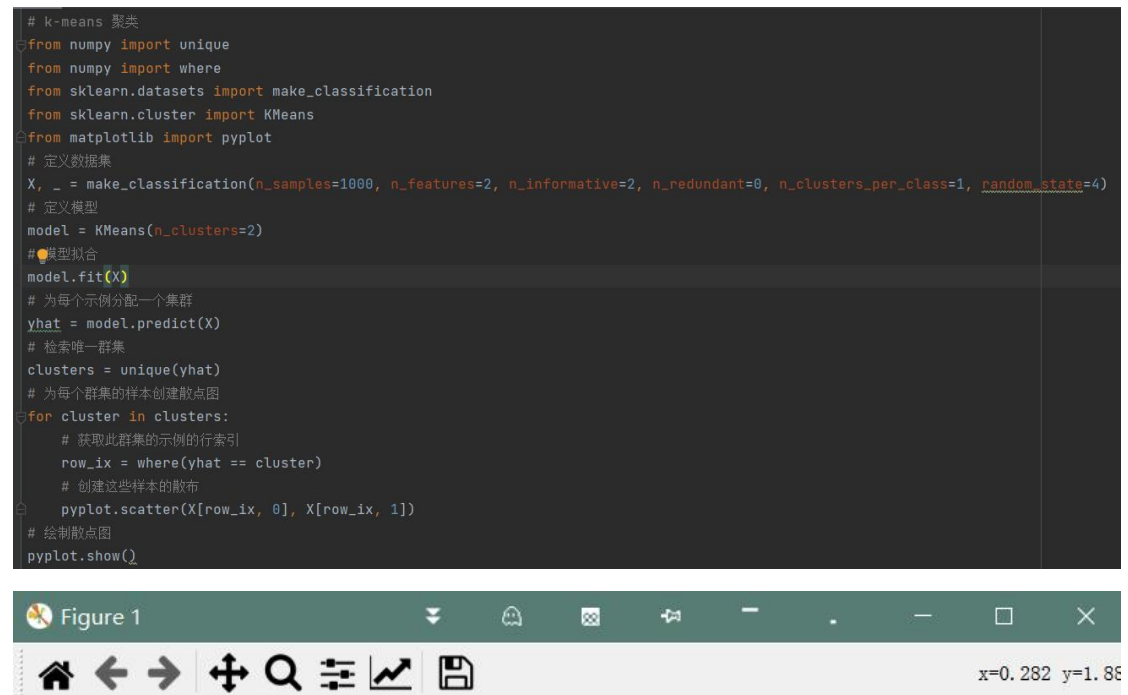
(1) 数据集

使用 `make_classification()` 函数创建一个测试二分类数据集。数据集将有 1000 个示例，每个类有两个输入要素和一个群集。这些群集在两个维度上是可见的，因此我们可以用散点图绘制数据，并通过指定的群集对图中的点进行颜色绘制。代码与数据集如下图所示。



(2) K-均值聚类

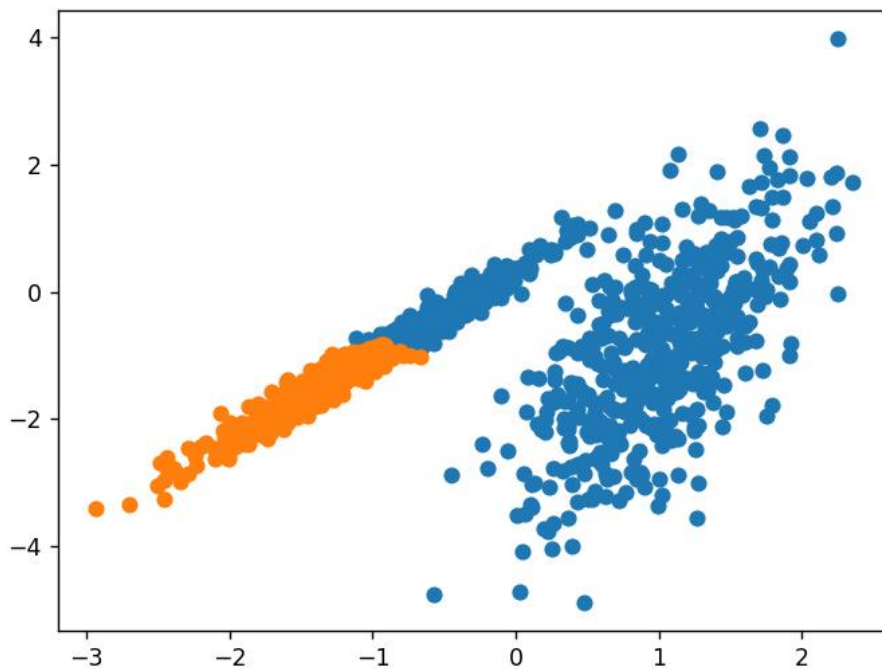
运行该示例符合训练数据集上的模型，并预测数据集中每个示例的群集。然后创建一个散点图，并由其指定的群集着色。



(3) 聚合聚类

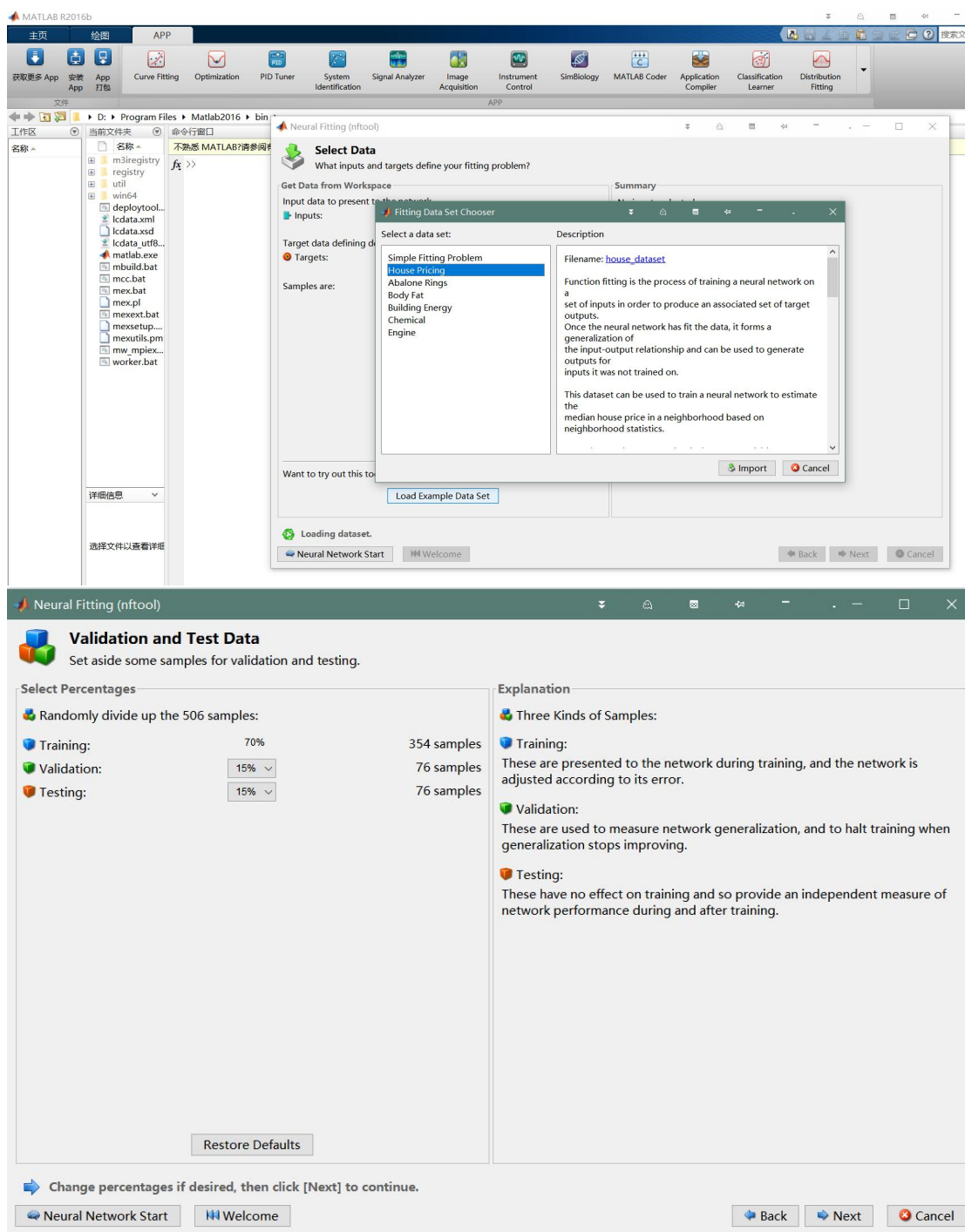
聚合聚类涉及合并示例，直到达到所需的群集数量为止。它是层次聚类方法的更广泛类的一部分，通过 AgglomerationClustering 类实现的，主要配置是“n_clusters”集，这是对数据中的群集数量的估计。

```
1 # 聚合聚类
2 from numpy import unique
3 from numpy import where
4 from sklearn.datasets import make_classification
5 from sklearn.cluster import AgglomerativeClustering
6 from matplotlib import pyplot
7 # 定义数据集
8 X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
9 # 定义模型
10 model = AgglomerativeClustering(n_clusters=2)
11 # 模型拟合与聚类预测
12 yhat = model.fit_predict(X)
13 # 检索唯一群集
14 clusters = unique(yhat)
15 # 为每个群集的样本创建散点图
16 for cluster in clusters:
17     # 获取此群集的示例的行索引
18     row_ix = where(yhat == cluster)
19     # 创建这些样本的散布
20     pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
21 # 绘制散点图
22 pyplot.show()
```



三、深度神经网络学习案例

使用 MATLAB 神经网络工具包完成深度神经网络学习。训练集采用的是内置的 House Pricing。具体实践流程见下图所示。



Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Train using Levenberg-Marquardt. (trainlm)

Train

Notes

Training multiple times will generate different results due to different initial conditions and sampling.

Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

! Train network, then click [Next].

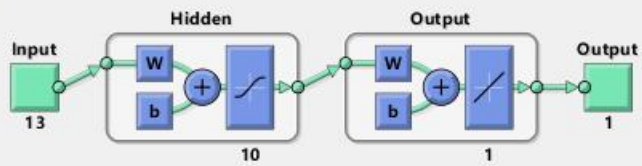
Neural Network StartWelcomeBackNextCancel

Results

	Samples	MSE	R
Training:	354	-	-
Validation:	76	-	-
Testing:	76	-	-

Plot FitPlot Error HistogramPlot Regression

Neural Network



Algorithms


Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	20 iterations	1000
Time:		0:00:00	
Performance:	194	2.97	0.00
Gradient:	914	5.59	1.00e-07
Mu:	0.00100	0.0100	1.00e+10
Validation Checks:	0	6	6

Plots

Performance	(plotperform)
Training State	(plottrainstate)
Error Histogram	(ploterrhist)
Regression	(plotregression)
Fit	(plotfit)

Plot Interval:  1 epochs

✓ Validation stop.

Stop Training

Cancel

