



Python与金融数据挖掘(16)

文欣秀

wenxinxiu@ecust.edu.cn

Python应用领域

文本分析：Jieba、Nltk...

科学计算：Numpy、SciPy...

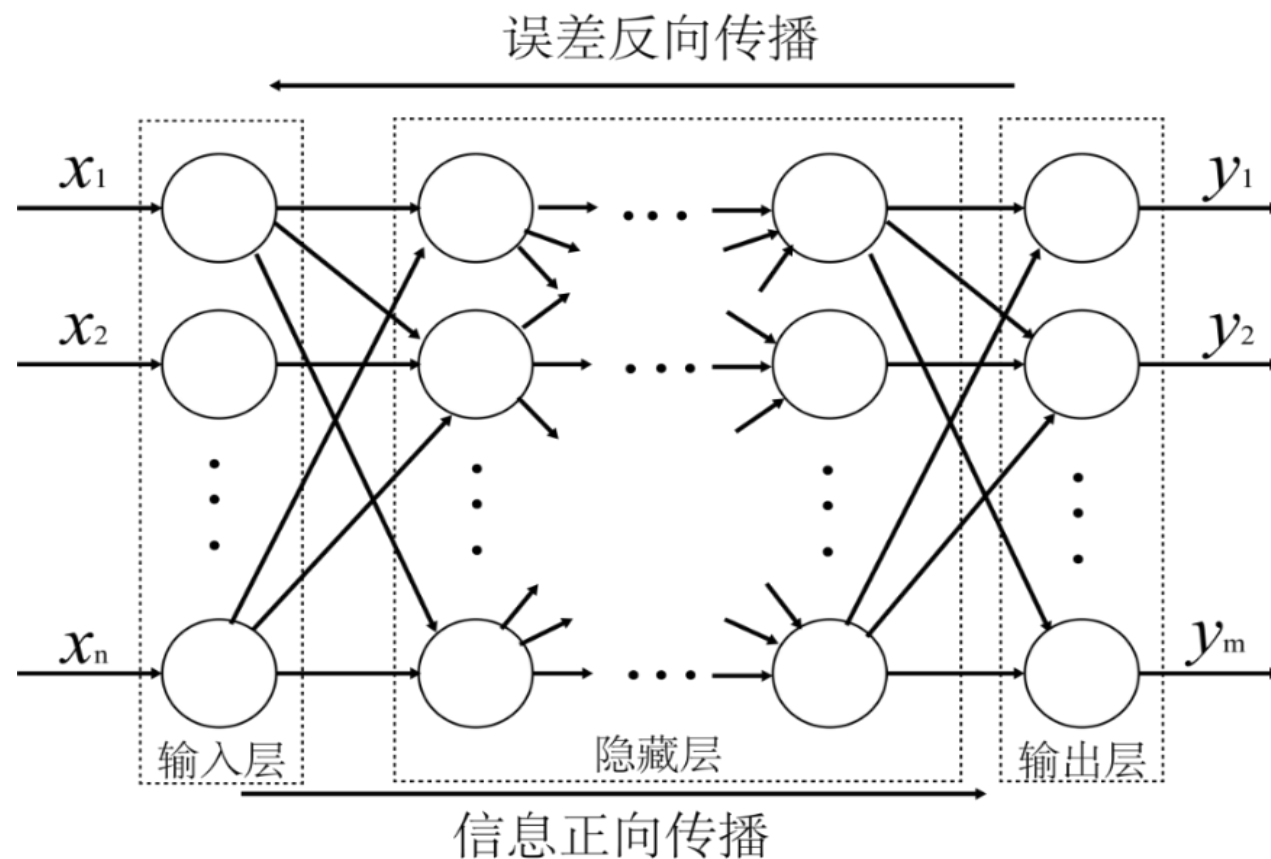
数据分析：Pandas、Matplotlib...

机器学习：Scikit-Learn、Keras...

深度学习：**Tensorflow**、Mindspore...

BP神经网络

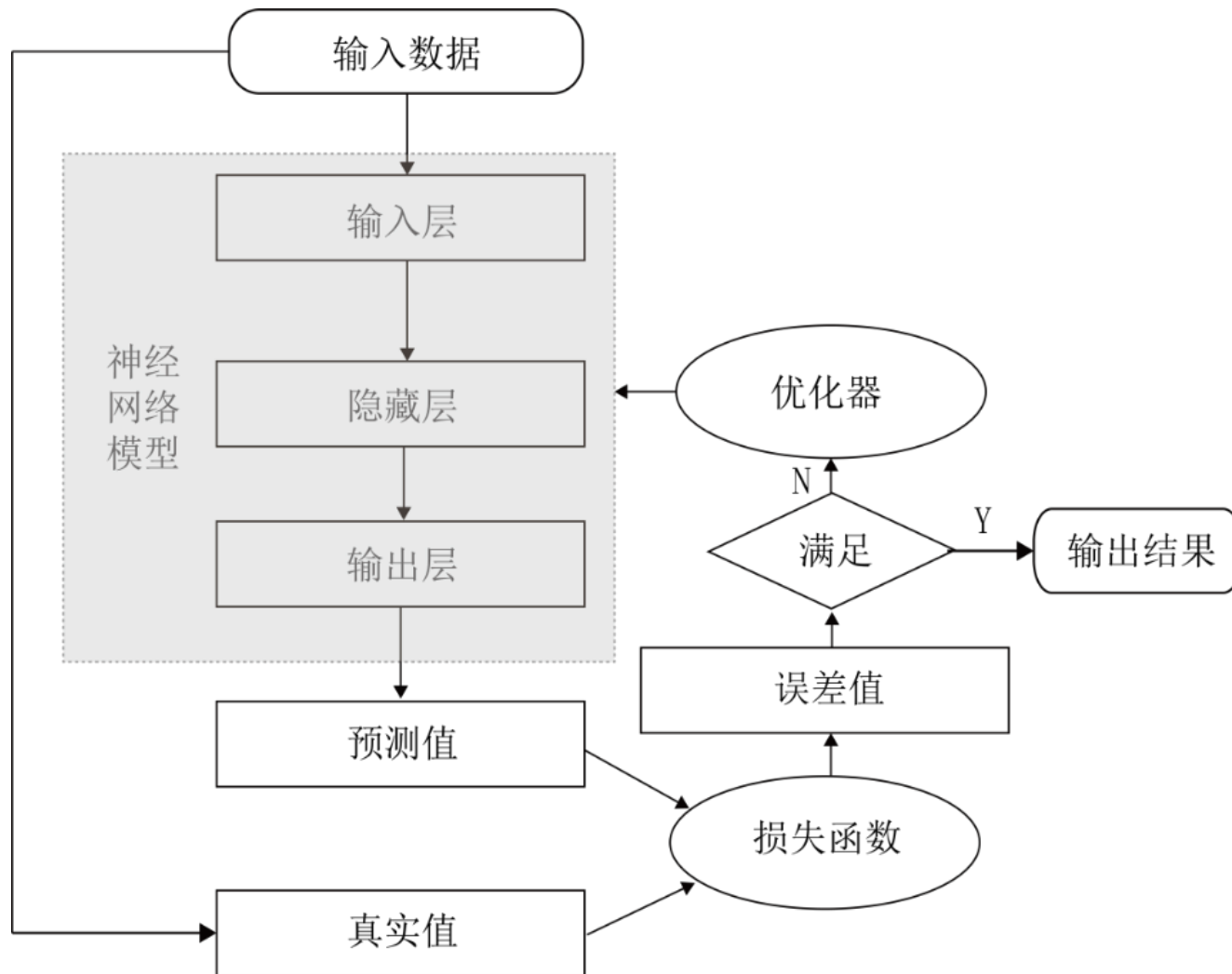
- **BP(Back Propagation)神经网络**：指多层前馈神经网络



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, \dots, 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, \dots, 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



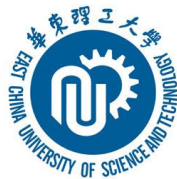
TensorFlow与Keras

- **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作为其高阶API已经集成其中。

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



股价预测案例

案例分析一

#0.导入库

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import StandardScaler #标准化处理
```

```
from sklearn.model_selection import train_test_split #划分训练集、测试集
```

```
from tensorflow.keras.models import Sequential #序列模型
```

```
from tensorflow.keras.layers import Dense #全连接层
```

```
from tensorflow.keras.optimizers import Adam #优化器
```

```
plt.rcParams["font.family"] = ["SimHei"] # 解决中文乱码问题
```

案例分析一

1. 数据获取

```
import tushare as ts
```

```
ts.set_token('XXX')
```

#换成自己的token

```
pro = ts.pro_api()
```

#初始化

```
data = pro.daily(ts_code='601398.SH', start_date='2023-01-01',  
end_date='2024-01-01')
```

获取工商银行历史数据

```
data = data.sort_values('trade_date') # 按日期排序
```

案例分析一

2. 数据预处理

开盘价、最高价、最低价、成交量为特征

```
features = data[['open', 'high', 'low', 'vol']]
```

```
target = data['close']
```

收盘价作为预测目标

```
scaler = StandardScaler()
```

```
features_scaled = scaler.fit_transform(features) #拟合并转换数据
```

```
target_scaled = scaler.fit_transform(target.values.reshape(-1, 1))
```

案例分析一

3.构建神经网络模型

```
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target_scaled,  
test_size=0.2, random_state=42)
```

```
model = Sequential(
```

```
Dense(64, activation='relu', input_shape=(X_train.shape[1],)), # 隐藏层1
```

```
Dense(32, activation='relu'),
```

```
Dense(1)
```

```
)
```

隐藏层神
经元个数

输入层神经元个数

激活函数

隐藏层神
经元个数

输出层神经元个数

隐藏层1

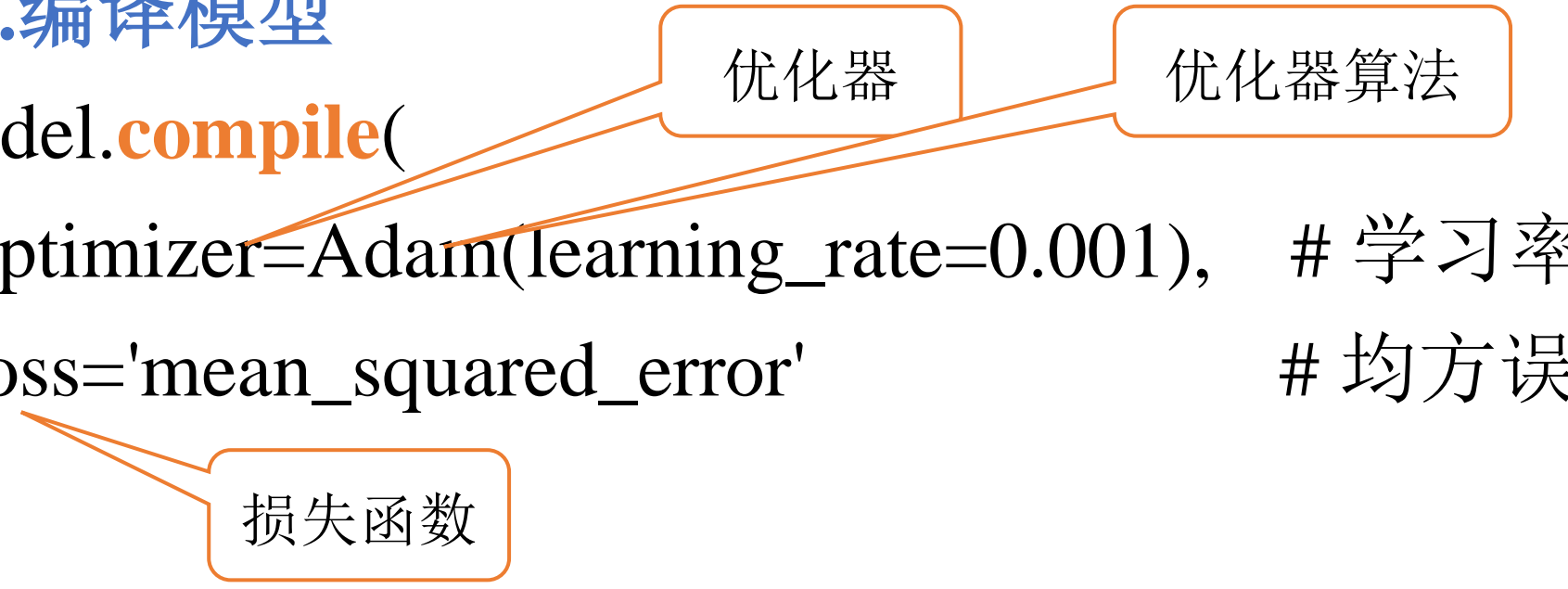
隐藏层2

输出层

案例分析一

4.编译模型

```
model.compile(  
    optimizer=Adam(learning_rate=0.001),    # 学习率  
    loss='mean_squared_error'              # 均方误差  
)
```

A diagram with three orange-bordered callout boxes. The box labeled '优化器' (Optimizer) has two lines pointing to 'optimizer=Adam' and 'learning_rate=0.001'. The box labeled '优化器算法' (Optimizer Algorithm) has one line pointing to 'Adam'. The box labeled '损失函数' (Loss Function) has one line pointing to 'loss='mean_squared_error''.

优化器

优化器算法

损失函数

案例分析一

5. 训练模型

```
history = model.fit(  
    X_train, y_train,  
    epochs=50,      # 训练轮数  
    batch_size=32,  # 批量大小  
    validation_split=0.2, # 验证集比例  
    verbose=1        # 输出进度条记录  
)
```

案例分析一

6.模型评估与预测

```
test_loss = model.evaluate(X_test, y_test, verbose=0)
```

```
print(f"测试集均方误差(MSE): {test_loss:.4f}")
```

```
y_pred_scaled = model.predict(X_test)
```

```
y_pred = scaler.inverse_transform(y_pred_scaled)
```

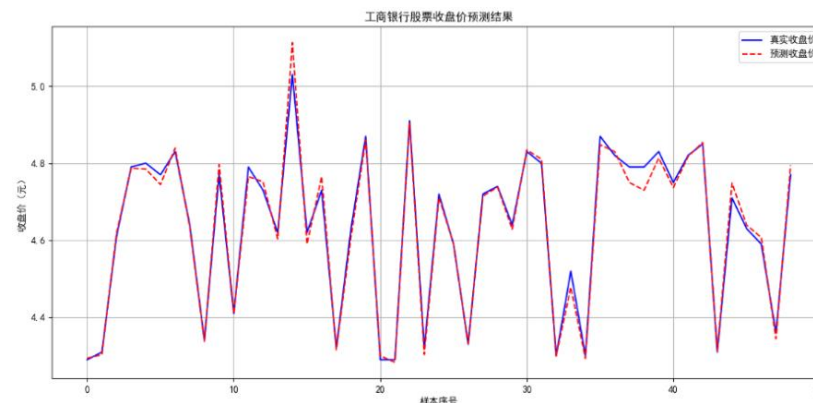
```
y_test_real = scaler.inverse_transform(y_test)
```

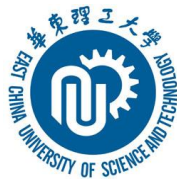
将经过标准化或归一化处理的数据转换回原始数据

案例分析一

7.数据可视化

```
plt.figure(figsize=(12, 6))  
plt.plot(y_test_real, label='真实收盘价', color='blue')  
plt.plot(y_pred, label='预测收盘价', color='red', linestyle='--')  
plt.title('工商银行股票收盘价预测结果')  
plt.xlabel('样本序号')  
plt.ylabel('收盘价')  
plt.legend()  
plt.show()
```





Fashion MNIST数据集分类

案例分析二

利用Keras构建神经网络模型，对Fashion MNIST数据集进行分类。
Fashion MNIST数据集由大小为28*28、分为10个类别的**70000张灰色图像**组成，其中**60000张训练集图像**，以及**10000张测试集图像**，
类别标签与图像所表示的服装类别对应关系如表所示。

标签	0	1	2	3	4	5	6	7	8	9
Description	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
描述	T 恤/上衣	裤子	套头衫	连衣裙	外套	凉鞋	衬衫	运动鞋	包	短靴

案例分析二

#0.导入TensorFlow

```
import tensorflow as tf
```

表 6-3-1 类别标签与服装类的对应关系

标签	0	1	2	3	4	5	6	7	8	9
Description	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
描述	T 恤/上衣	裤子	套头衫	连衣裙	外套	凉鞋	衬衫	运动鞋	包	短靴

#1.载入Fashion-MNIST 数据集

```
fashion_mnist = tf.keras.datasets.fashion_mnist
```

```
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

案例分析二

#查看训练集和测试集的形状

```
print('The shape of train data=',X_train.shape)
print('The shape of y_train:',y_train.shape)
print('The shape of test data=',X_test.shape)
print('The shape of y_test:',y_test.shape)
```

标签	0	1	2	3	4	5	6	7	8	9
Description	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
描述	T 恤/上衣	裤子	套头衫	连衣裙	外套	凉鞋	衬衫	运动鞋	包	短靴

案例分析二

#显示训练集第二个图像及其内容，0黑色，255白色

```
import matplotlib.pyplot as plt
```

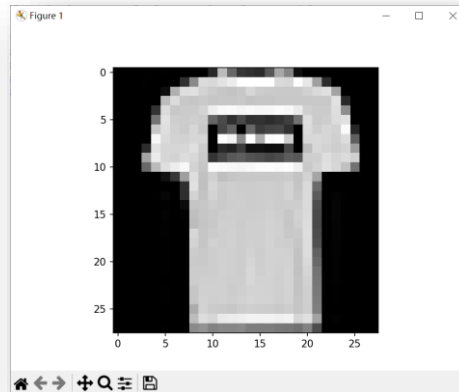
```
plt.figure()
```

```
plt.imshow(X_train[1], cmap='gray')
```

```
plt.savefig('2.jpg')
```

```
plt.show()
```

```
print(X_train[1])
```



```
[[ 0  0  0  0  0  0  1  0  0  0  0  41 188 103  54  48  43  87 168
 133 16  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  49 136 219 216 228 236 255 255 255 255 217
 215 254 231 160  45  0  0  0  0  0]
 [ 0  0  0  0  0  14 176 222 224 212 203 198 196 200 215 204 202 201
 201 201 209 218 224 164  0  0  0  0]
 [ 0  0  0  0  0 188 219 200 198 202 198 199 199 201 196 198 198 200
 200 200 200 201 200 225  41  0  0  0]
 [ 0  0  0  0  51 219 199 203 203 212 238 248 250 245 249 246 247 252
 248 235 207 203 203 222 140  0  0  0]
 [ 0  0  0  0 116 226 206 204 207 204 101  75  47  73  48  50  45  51
  63 113 222 202 206 220 224  0  0  0]
 [ 0  0  0  0 200 222 209 203 215 200  0  70  98  0 103  59  68  71
  49  0 219 206 214 210 250  38  0  0]
 [ 0  0  0  0 247 218 212 210 215 214  0 254 243 139 255 174 251 255
 205  0 215 217 214 208 220  95  0  0]
 [ 0  0  0  45 226 214 214 215 224 205  0  42  35  60  16  17  12  13
  70  0 189 216 212 206 212 156  0  0]
 [ 0  0  0 164 235 214 211 220 216 201  52  71  89  94  83  78  70  76
  92  87 206 207 222 213 219 208  0  0]]
```

案例分析二

#建立图像和标签的映射表

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',  
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

#显示训练集的前20个图像和标签

```
plt.figure(figsize=(10,9))
```

标签	0	1	2	3	4	5	6	7	8	9
Description	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
描述	T 恤/上衣	裤子	套头衫	连衣裙	外套	凉鞋	衬衫	运动鞋	包	短靴

案例分析二

```
num=20
```

```
for i in range(0, num):
```

```
    plt. subplot(4,5,i+1)
```

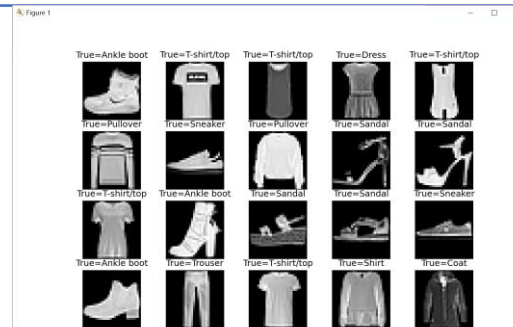
```
    plt. imshow(X_train[i], cmap='gray')
```

```
    plt. xticks([]) #设置 x 轴刻度值为空
```

```
    plt. yticks([]) #设置 y 轴刻度值为空
```

```
    plt. title("True="+str(class_names[y_train[i]]))
```

```
plt.show()
```



案例分析二

#3. 利用**reshape**函数将二维图像(28x28)转换为一维向量(784)

```
X_train_reshape = X_train.reshape(X_train.shape[0], 28*28)
```

```
X_test_reshape = X_test.reshape(X_test.shape[0], 28*28)
```

#查看经过**reshape**之后训练集和测试集的形状

```
print('The shape of train reshape data=', X_train_reshape.shape)
```

```
print('The shape of y_train:', y_train.shape)
```

```
print('The shape of test reshape data=', X_test_reshape.shape)
```

```
print('The shape of y_test:', y_test.shape)
```


案例分析二

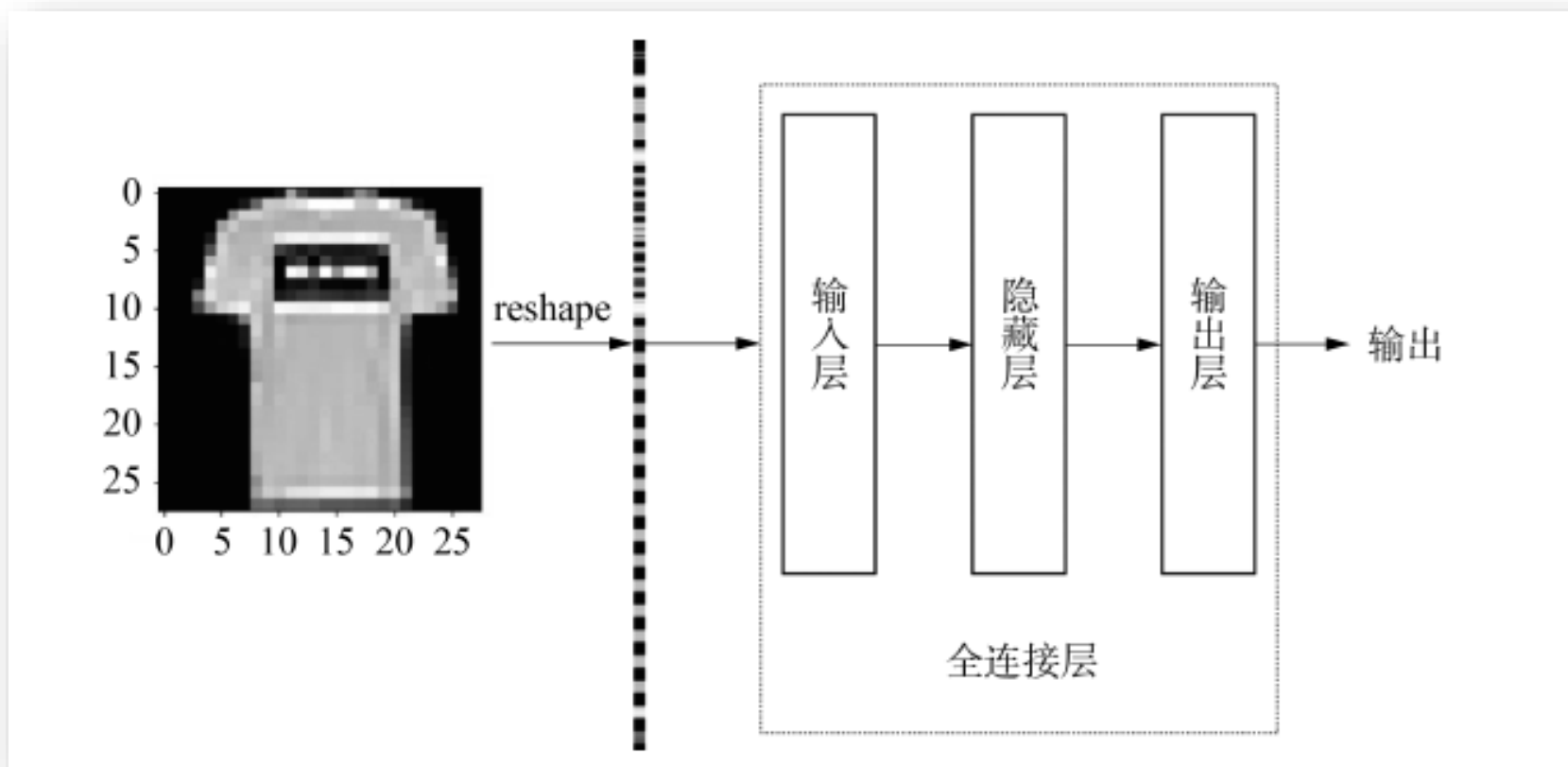
#4. 归一化数字图像，从0-255同除以255实现归一化

```
X_train_norm, X_test_norm = X_train_reshape/255.0, X_test_reshape/255.0
```

```
print(X_train_norm[0])
```

神经网络模型图

- 该模型主要是由输入层、隐藏层、输出层构成



案例分析二

#5. 利用Sequential模型构建神经网络模型

```
model = tf.keras.models.Sequential([  
    tf.keras.layers.Dense(50, input_dim=28*28, activation='relu', name='Hidden'),  
    tf.keras.layers.Dense(10, activation='softmax', name='Output')  
])  
#打印模型的概况  
print(model.summary())
```

输入层神经元个数

激活函数

命名隐藏层为Hidden

隐藏层神经元个数

输出层神经元个数

多分类问题常采用该激活函数

命名输出层为Output

Total params: 39760 (155.31 KB)
Trainable params: 39760 (155.31 KB)
Non-trainable params: 0 (0.00 Byte)

结果分析二

Hidden层计算方法：输入层的神经元个数为784，隐藏层的神经元个数为50，隐藏层每个神经元还包括一个偏差项(bias)，因此，隐藏层的总参数数目为：

$$\text{Hidden_Param} = 784 * 50 + 50 = 39250$$

Output层计算方法：隐藏层的神经元个数为50，输出层的神经元个数为10，包含10个偏差项，因此，输出层的总参数数目为：

$$\text{Output_Param} = 50 * 10 + 10 = 510$$

因此，Total params和 Trainable params参数总数目之和为： **39760**

案例分析二

#6. 利用`compile()`函数实现模型的编译

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])
```

使用'adam'作为
模型优化器

使用crossentropy
作为损失函数

使用准确率
来评价模型

#7. 模型训练

```
model.fit(X_train_norm, y_train, epochs=10)
```

标准化后的
训练集特征

训练集的
类别标签

需要执行的
训练周期数

结果分析二

Epoch 1/10	1688/1688 [=====] - 4s 2ms/step	loss: 0.5451 - accuracy: 0.8125	val_loss: 0.4378 - val_accuracy: 0.8422
Epoch 2/10	1688/1688 [=====] - 2s 1ms/step	loss: 0.4170 - accuracy: 0.8542	val_loss: 0.4217 - val_accuracy: 0.8527
Epoch 3/10	1688/1688 [=====] - 3s 2ms/step	loss: 0.3798 - accuracy: 0.8643	val_loss: 0.3762 - val_accuracy: 0.8637
Epoch 4/10	1688/1688 [=====] - 3s 1ms/step	loss: 0.3528 - accuracy: 0.8739	val_loss: 0.3818 - val_accuracy: 0.8628
Epoch 5/10	1688/1688 [=====] - 3s 2ms/step	loss: 0.3367 - accuracy: 0.8780	val_loss: 0.3536 - val_accuracy: 0.8708
Epoch 6/10	1688/1688 [=====] - 3s 2ms/step	loss: 0.3187 - accuracy: 0.8851	val_loss: 0.3545 - val_accuracy: 0.8728
Epoch 7/10	1688/1688 [=====] - 3s 1ms/step	loss: 0.3062 - accuracy: 0.8876	val_loss: 0.3382 - val_accuracy: 0.8780
Epoch 8/10	1688/1688 [=====] - 3s 2ms/step	loss: 0.2965 - accuracy: 0.8922	val_loss: 0.3585 - val_accuracy: 0.8765
Epoch 9/10	1688/1688 [=====] - 3s 2ms/step	loss: 0.2879 - accuracy: 0.8948	val_loss: 0.3383 - val_accuracy: 0.8765
Epoch 10/10	1688/1688 [=====] - 3s 2ms/step	loss: 0.2776 - accuracy: 0.8981	val_loss: 0.3275 - val_accuracy: 0.8807

随着训练周期的增加，训练集和验证集的准确率在不断提高，损失函数值不断减小。在训练后期时，验证集的准确率低于训练集的准确率，出现了**轻微过拟合**的现象。

案例分析二

#8. 利用evaluate函数对模型进行评估

```
model.evaluate(X_test_norm, y_test, verbose=1)
```

归一化的测试集图像

测试集类别标签

输出包含进度条的日志信息

#9. 模型预测

```
prediction=model.predict_classes(X_test_norm)
```

```
313/313 [=====] - 0s 703us/step - loss: 0.3486 - accuracy: 0.8757
```

案例分析二

#显示测试集的前20个图像的预测类别和真实类别

```
plt.figure(figsize=(18,9))
```

```
num=20
```

```
for i in range(0, num):
```

```
    plt.subplot(4,5,i+1)
```

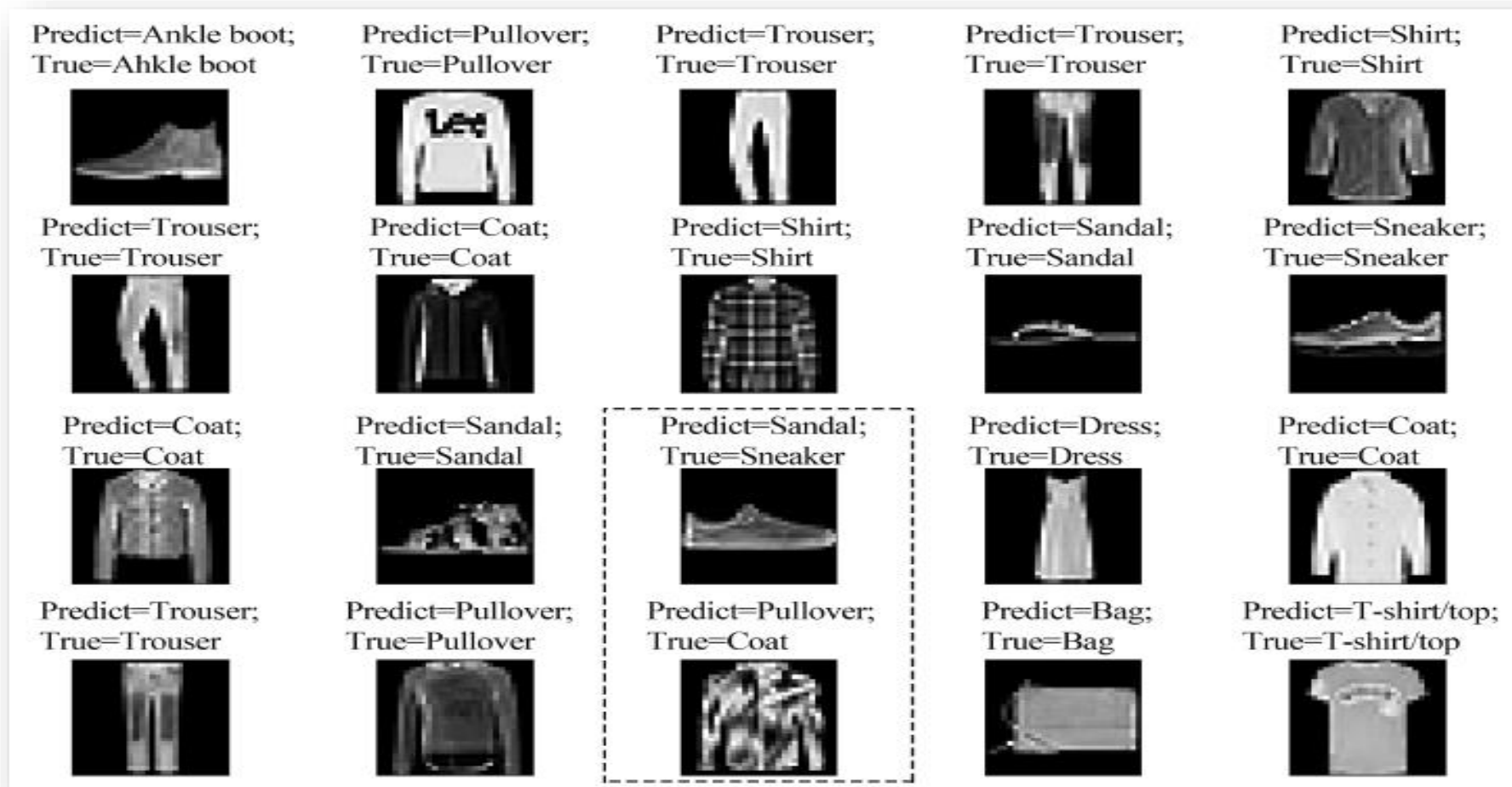
```
    plt.imshow(X_test[i], cmap='gray')
```

```
    plt.xticks([]); plt.yticks([])
```

```
    plt.title('Predict='+str(class_names[prediction[i]]) +';True=' +  
str(class_names[y_test[i]]))
```

```
plt.show()
```


案例分析二



第三行第三列的运动鞋被误分为凉鞋，第四行第三列的外套被误分为套头衫。

案例分析二

#保存训练好的模型

```
modelname='my_model.h5'  
model.save(modelname)  
print('保存的模型名称 ', modelname)
```

h5文件是层次数据格式第5代的版本（ Hierarchical Data Format, HDF5），用以存储和组织大规模数据。

#利用保存的模型进行预测

```
model = tf.keras.models.load_model(modelname)  
prediction=model.predict_classes(X_test_norm)
```

案例分析

对于同一个数据集，神经网络模型如果想获得更好的分类效果，可以增加数据集的输入特征数量、增加隐藏层神经元数目和隐藏层层数。在输入的特征数目不变时，可行性的方法包括：

- 增加隐藏层神经元的数目；
- 增加隐藏层的层数。

案例分析二

➤ 增加隐藏层神经元的数目

#5. 构建Sequential模型

```
model = tf.keras.models.Sequential()  
model.add(tf.keras.layers.Dense(500, input_dim=28*28, activation='relu',  
name='Hidden'))  
model.add(tf.keras.layers.Dense(10, activation='softmax', name='Output'))
```

模型优化

➤ 增加隐藏层神经元的数目

该神经网络在其他结构不变的情况下，只将Hidden层的神经元数目从**50**个增加到**500**个，准确率从**0.8977**上升到**0.9159**。

```
1875/1875 [=====] - 10s 5ms/step - loss: 0.2637 - accuracy: 0.9025
Epoch 7/10
1875/1875 [=====] - 9s 5ms/step - loss: 0.2521 - accuracy: 0.9056
Epoch 8/10
1875/1875 [=====] - 9s 5ms/step - loss: 0.2410 - accuracy: 0.9100
Epoch 9/10
1875/1875 [=====] - 10s 5ms/step - loss: 0.2325 - accuracy: 0.9138
Epoch 10/10
1875/1875 [=====] - 19s 10ms/step - loss: 0.2215 - accuracy: 0.9159
```

案例分析

➤ 增加隐藏层的层数

#5. 构建Sequential模型

```
model = tf.keras.models.Sequential()
```

```
model.add(tf.keras.layers.Dense(50,input_dim=28*28,activation='relu',name='Hidden1'))
```

```
model.add(tf.keras.layers.Dense(50,activation='relu',name='Hidden2'))
```

```
model.add(tf.keras.layers.Dense(50,activation='relu',name='Hidden3'))
```

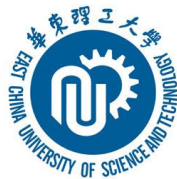
```
model.add(tf.keras.layers.Dense(10,activation='softmax',name='Output'))
```

模型优化

➤ 增加隐藏层的层数

该神经网络在其他结构不变的情况下，只将Hidden层增加到3层，每层50个神经元，准确率从0.8977上升到0.9020。

```
Epoch 1/10
1875/1875 [=====] - 5s 3ms/step - loss: 0.5265 - accuracy: 0.8131
Epoch 2/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.3861 - accuracy: 0.8586
Epoch 3/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.3469 - accuracy: 0.8720
Epoch 4/10
1875/1875 [=====] - 3s 1ms/step - loss: 0.3252 - accuracy: 0.8802
Epoch 5/10
1875/1875 [=====] - 3s 1ms/step - loss: 0.3109 - accuracy: 0.8854
Epoch 6/10
1875/1875 [=====] - 3s 1ms/step - loss: 0.2972 - accuracy: 0.8894
Epoch 7/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.2859 - accuracy: 0.8943
Epoch 8/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.2763 - accuracy: 0.8964
Epoch 9/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.2663 - accuracy: 0.9010
Epoch 10/10
1875/1875 [=====] - 3s 1ms/step - loss: 0.2597 - accuracy: 0.9020
```



期末课程复习

题型及分值

选择题： **2分*20题=40分**， 涵盖全部教学内容

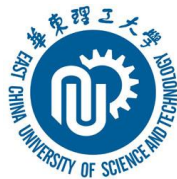
程序填空题： **2分*3空*5题=30分**

循环结构与函数调用、字典与列表数据处理、词频统计及可视化、网络爬虫（爬二进制文件）、Pandas数据清洗及可视化、标准数据集加载及Numpy数据分析、深度学习

题型及分值

编程题：10分*3题=30分

- 1、文件读写、获取正则内容及并评分、MySQL数据库
- 2、窗体设计、Pandas数据清洗与分析、Matplotlib可视化
- 3、机器学习算法原理、评价标准及应用(分类、回归、聚类)



谢 谢