实战案例5：Fashion-MNIST图片分类

作者： Robin
日期： 2018/06
提问： 小象问答
数据集来源： kaggle

1. 案例描述

FashionMNIST 是一个替代 MNIST 手写数字集的图像数据集。 它是由 Zalando（一家德国的时尚科技公司）旗下的研究部门提供。其涵盖了来自 10 种类别的共 7 万个不同商品的正面图片。FashionMNIST 的大小、格式和训练集 / 测试集划分与原始的 MNIST 完全一致。60000/10000 的训练测试数据划分，28x28 的灰度图片。

2. 数据集描述

- Kaggle提供的数据集
- 数据字典
  - **label**: 类别标签，整型
  - **pixel1** -- **pixel784**: 784个像素值(784 = 28 * 28)
- 标签
  - **0**: T-shirt/top
  - **1**: Trouser
  - **2**: Pullover
  - **3**: Dress
  - **4**: Coat
  - **5**: Sandal
  - **6**: Shirt
  - **7**: Sneaker
  - **8**: Bag
  - **9**: Ankle boot

3. 任务描述

- 使用机器学习方法进行图片分类

4. 主要代码解释

- 代码结构

```
lect06_proj
├── data
    ├── fashion-mnist_train.csv   # FashionMNIST训练数据文件
    ├── fashion-mnist_test.csv   # FashionMNIST测试数据文件
├── output  # 程序输出结果保存的目录
├── main.py     # 主程序
├── utils.py     # 工具文件，包含数据加载、图像数据显示、特征工程等
├── config.y    # 配置文件
├── lect06_proj_readme.pdf  # 案例讲解文档
```

- **main.py**

由于数据量比较大，模型训练时间稍长，提供了一个简单和复杂的训练机制：

```
1. 简单的默认参数的Logistic Regression；
2. 复杂的包含交叉验证的多个模型
```

```
# True表示使用简单默认的Logistic Regression分类器
# 否则使用多个模型的交叉验证
IS_SIMPLE_EXP = True

def main():
    ...
    if IS_SIMPLE_EXP:
        # 耗时比较短
        print('简单的Logistic Regression分类：')
        ...
    else:
        # 耗时比较长
        print('多个模型交叉验证分类比较：')
        ...
    ...
```

- **utils.py**

由于是进行图像处理，后续代码会用到直方图均衡化等操作，需要将numpy的数据类型转换为uint8：`astype(np.uint8)`

```
def load_fashion_mnist_dataset(data_file):
    ...
    X = data_df.iloc[:, 1:].values.astype(np.uint8)
    ...
```

- **utils.py**

原始数据中每个图像被“变形”成了一个行向量，在显示图像时，需要对该向量进行reshape，即`img_data = X[i, :].reshape(config.img_rows, config.img_cols)`。`config.img_rows`和`config.img_cols`在`config.py`中已定义，均为28。

```
def plot_random_samples(X):
    ...
    img_data = random_X[i, :].reshape(config.img_rows, config.img_cols)
    ...

def extract_feats(X):
    ...
    img_data = X[i, :].reshape(config.img_rows, config.img_cols)
    ...
```

- **utils.py**

该项目中对于图像进行了2步处理：

```
1. 中值滤波，去除噪声
2. 直方图均衡化
```

```
def extract_feats(X):
    ...
    # 中值滤波，去除噪声
        blur_img_data = cv2.medianBlur(img_data, 3)

        # 直方图均衡化
        equ_blur_img_data = cv2.equalizeHist(blur_img_data)
    ...
```

- **utils.py**

该项目中对于特征进行了标准化处理，因为之前使用的是`np.uint8`数据类型，这里的`StandardScaler`要求是`np.float64`类型，所以又进行了一次类别转换。（如果不转的话，会有警告出现，不影响程序的正常运行）

```
def do_feature_engineering(feats_train, feats_test):
    ...
    std_scaler = StandardScaler()
    scaled_feats_train = std_scaler.fit_transform(feats_train.astype(np.float64))
    scaled_feats_test = std_scaler.transform(feats_test.astype(np.float64))
    ...
```

5. 案例总结

- 该项目通过使用机器学习进行图像数据的分类，包含了如下内容:
    - 图像数据操作
    - 常用的图像数据处理方法

6. 课后练习

- 尝试去掉对图像数据处理（中值滤波，直方图均衡化），然后比较模型的输出结果。

参考资料

1. OpenCV Python教程