Assignment1 - Classifying Sneakers versus Sandals

Tommy Liu, 22962256

March 27, 2024

1 Constant Definition

[ ]:
```
DATASET_PATH = "./dataset/"
RANDOM_SEED = 5508
```

2 Dataset Import and Preprocess

In this section, we import data and preprocess them. First, we create a filter from labels dataframe. Then, we filter row in features and labels, and make a copy of then. Finally, we convert label to binary, 1 for Sandal and 0 for Sneaker.

[ ]:
```
import pandas as pd


def readCSV(filename: str):
    return pd.read_csv(DATASET_PATH + filename, header=None)


origin_train_set = readCSV("FMNIST_training_set.csv")
origin_train_set_labels = readCSV("FMNIST_training_set_labels.csv")
origin_test_set = readCSV("FMNIST_test_set.csv")
origin_test_set_labels = readCSV("FMNIST_test_set_labels.csv")


def preprocess_data(features: pd.DataFrame, labels: pd.DataFrame):
    filter = labels.iloc[:, 0].isin([5, 7])
    filtered_features = features[filter].copy()
    filtered_labels = labels[filter].copy()
    filtered_labels[filtered_labels == 5] = 1  # Sandal
    filtered_labels[filtered_labels == 7] = 0  # Sneaker
    return filtered_features, filtered_labels


train_X, train_y = preprocess_data(origin_train_set, origin_train_set_labels)
test_X, test_y = preprocess_data(origin_test_set, origin_test_set_labels)
# check first 5 rows of the training set
train_y_copy = train_y.copy().rename(columns={0: "label"})
```

```
pd.concat([train_y_copy, train_X], axis=1).iloc[:5, :25]
```

[ ]:
```
    label  0  1  2  3  4  5  6  7  8  …  14  15  16  17  18  19   20   21  \
12      1  0  0  0  0  0  0  0  0  0  …   0   0   0   0   0   0    0    0
30      1  0  0  0  0  0  0  0  0  0  …   0   0   0   0   0   0    0    0
36      1  0  0  0  0  0  0  0  0  0  …   0   0   0   0   0  53  102  144
41      0  0  0  0  0  0  0  0  0  0  …   0   0   0   0   0   0    0    0
43      1  0  0  0  0  0  0  0  0  0  …   0   0   0   0   0   0    0    0

     22   23
12    0    0
30    0    0
36  169  149
41    0    0
43    0    0

[5 rows x 25 columns]
```

[ ]:
```
origin_train_set_labels.iloc[8]
```

[ ]:
```
0    1
Name: 8, dtype: int64
```

3 Summarising the datasets

3.1 D1 [3 marks]: List number of instances

We use shape attribute to get the number of instances in the dataset. The shape attribute shows the number of rows and columns, which the first number is the number of rows.

[ ]:
```
pd.DataFrame(
    {
        "Number of instances": [
            train_X.shape[0],
            test_X.shape[0],
            train_X.shape[0] + test_X.shape[0],
        ]
    },
    index=["Train set", "Test set", "Total"],
)
```

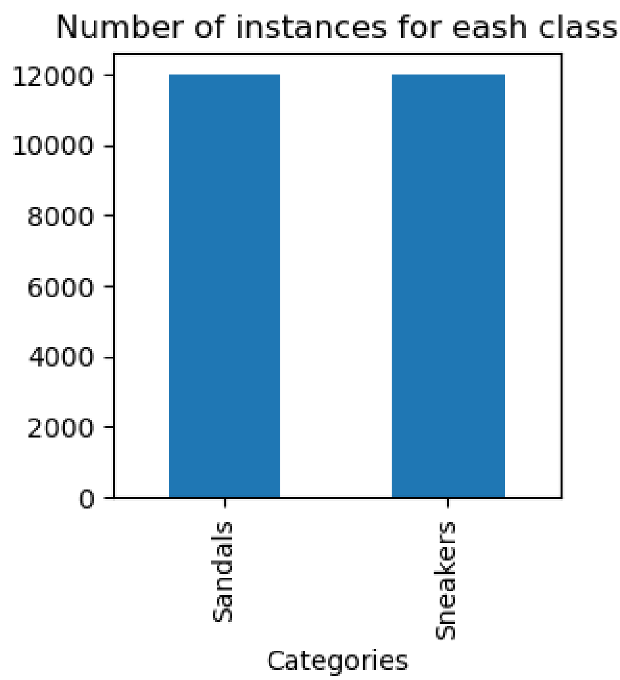
[ ]:
```
           Number of instances
Train set                11988
Test set                  2000
Total                    13988
```

3.2 D2 [2 marks]: Provide a bar plot showing the number of instances for each class label. Do you have an imbalanced training set?

We can see from the plot that, the numbers of two classed are balanced. So, we don't have an imbalanced training set.

[ ]:
```
pd.DataFrame(
    {
        "Categories": ["Sandals", "Sneakers"],
        "Number": [
            train_y[train_y == 1].shape[0],
            train_y[train_y == 0].shape[0],
        ],
    }
).plot(
    x="Categories",
    y="Number",
    kind="bar",
    legend=False,
    title="Number of instances for eash class",
    figsize=(3, 3),
)
```

[ ]:
```
<Axes: title={'center': 'Number of instances for eash class'},
xlabel='Categories'>
```

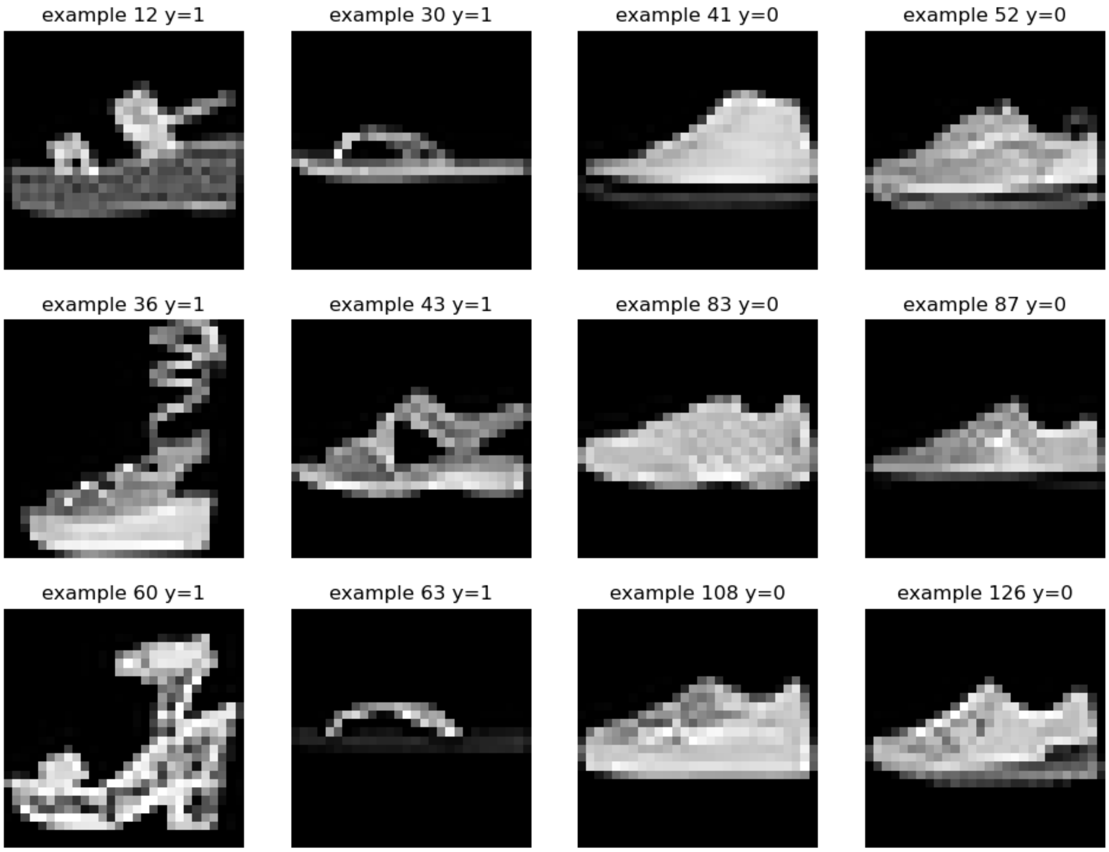


3.3 D3 [3 marks]: Plot the first six images/examples from each class with the corresponding example id and associated label on the top of the plot.

First, we find the first six images of each class. Then we arrange the images so they can show in either left side or right side. Finally, we plot the images, where 1 is Sandal and 0 is Sneaker.

[ ]:
```
IMAGE_SHAPE = (28, 28)

sandals = []
for index, row in train_X[train_y.iloc[:, 0] == 1].iloc[:6].iterrows():
    sandals.append(
        {
            "index": index,
            "image": row.values.reshape(IMAGE_SHAPE),
            "label": 1,
        }
    )

sneakers = []
for index, row in train_X[train_y.iloc[:, 0] == 0].iloc[:6].iterrows():
    sneakers.append(
        {
            "index": index,
            "image": row.values.reshape(IMAGE_SHAPE),
            "label": 0,
        }
    )

images = []
for i in range(0, 12):
    if i % 4 < 2:
        images.append(sandals.pop(0))
    else:
        images.append(sneakers.pop(0))
```

[ ]:
```
from matplotlib import pyplot as plt

fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(12, 9))

axes = axes.flatten()

for i, image in enumerate(images):
    ax = axes[i]
    ax.imshow(image['image'], cmap='gray')
    ax.set_title(f'example {image['index']} y={image['label']}')
    ax.axis('off')  # Turn off axis labels
```



4 Fitting your logistic regression classifier

For this two classes classification problem, we use logistic regression method. Refer to assignment instruction and Hands-on-Machine-Learning page 240, we obtain the following formula for logistic regression:

$$
\begin{aligned}
&\text{Sigmoid function:} && \sigma(x) = \frac{1}{1+e^{-x}} && (1)\\
&\text{Prediction:} && \hat{y} = \sigma(Xw) && (2)\\
&\text{Loss function:} && L = mean(y * \log(\hat{y}) + (1-y) * \log(1-\hat{y})) && (3)\\
&\text{Gardient:} && \frac{\delta L}{\delta w} = \frac{X^T(\hat{y}-y)}{length(X)} && (4)\\
& && && (5)
\end{aligned}
$$

[ ]:
```
import torch

X = torch.tensor(train_X.values)
y = torch.tensor(train_y.values)
```

```python
logistic_fn = lambda x: 1 / (1 + torch.exp(-x))
X_b = torch.cat((X, torch.ones(X.shape[0], 1)), dim=1)
theta = torch.zeros(X_b.shape[1], 1)
pred_y = logistic_fn(X_b @ theta)
loss = -torch.mean(y * torch.log(pred_y) + (1 - y) * torch.log(1 - pred_y))
gardients = 1 / X_b.shape[0] * X_b.t() @ (pred_y - y)
```

[ ]:
```python
(X_b.t() @ (pred_y - y)).shape
```

[ ]:
```
torch.Size([785, 1])
```

[ ]:
```python

```

[ ]:
```
True
```