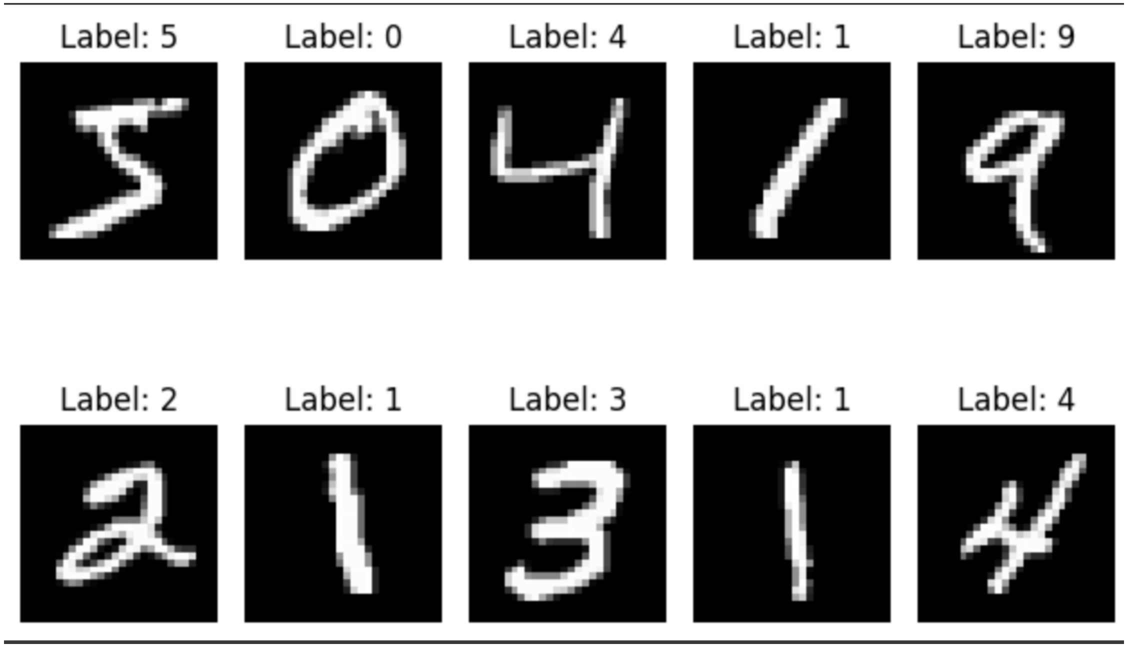
COMP3055
Machine Learning

Explain the Solution to Lab 2

Ying Weng
2024 Autumn

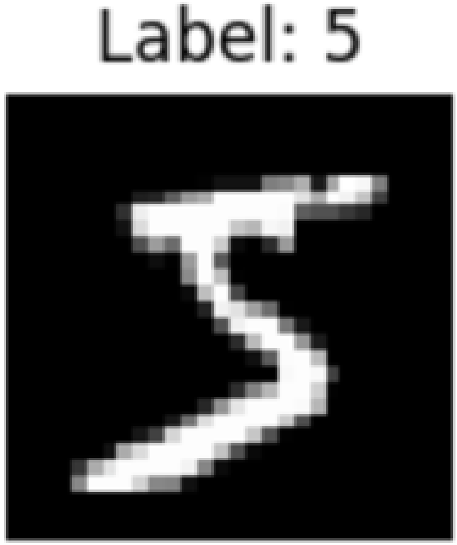
MNIST dataset

- **A handwritten digit dataset (number 0 to 9)**
- **Total 70000 images**
- **Each image is 28x28 grayscale image (pixel value 0 represents black, 255 represents white) , flatten into a 784-dimensional vector**



MNIST dataset

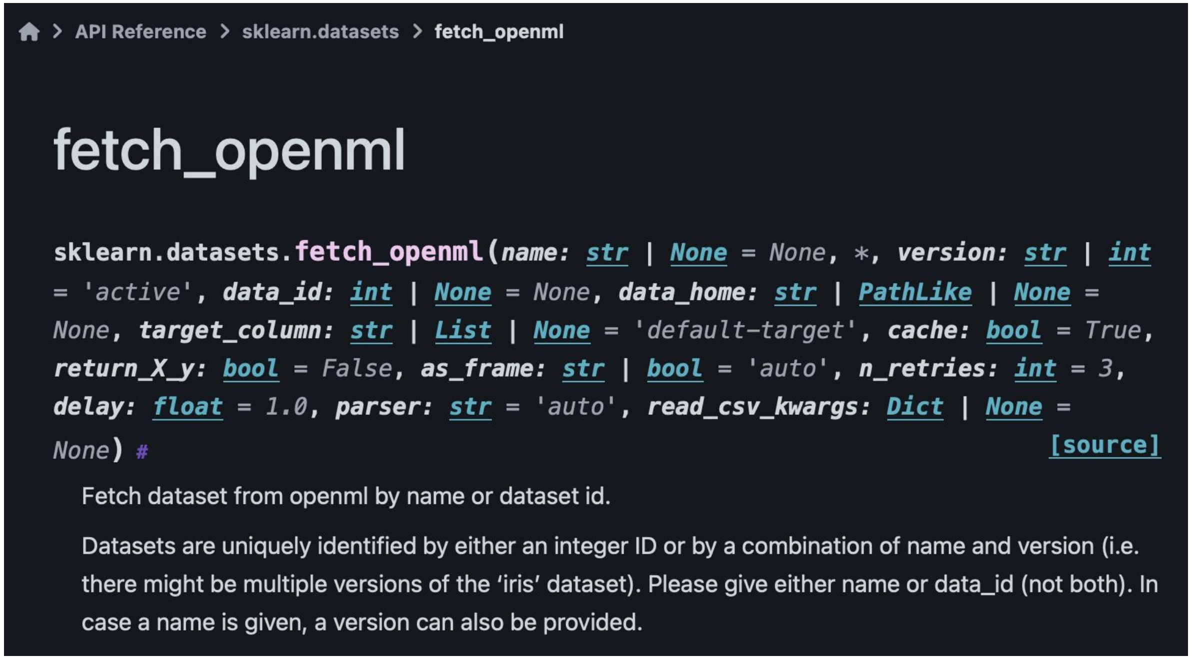
- An example of image 5



```
[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 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 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 3 18 18 18 126 136 175 26 166
255 247 127 0 0 0 0] [ 0 0 0 0 0 0 0 0 30 36 94 154 170
253 253 253 253 253 225 172 253 242 195 64 0 0 0 0] [ 0 0
0 0 0 0 0 49 238 253 253 253 253 253 253 253 253 251 93
82 82 56 39 0 0 0 0 0] [ 0 0 0 0 0 0 0 18 219 253 253 253
253 253 198 182 247 241 0 0 0 0 0 0 0 0 0] [ 0 0 0 0 0
0 0 0 80 156 107 253 253 205 11 0 43 154 0 0 0 0 0 0 0
0 0] [ 0 0 0 0 0 0 0 0 0 14 1 154 253 90 0 0 0 0 0 0 0
0 0 0 0 0 0] [ 0 0 0 0 0 0 0 0 0 0 0 139 253 190 2 0 0 0
0 0 0 0 0 0 0 0 0 0] [ 0 0 0 0 0 0 0 0 0 0 0 11 190 253
70 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
35 241 225 160 108 1 0 0 0 0 0 0 0 0 0] [ 0 0 0 0 0 0 0
0 0 0 0 0 0 81 240 253 253 119 25 0 0 0 0 0 0 0 0 0] [ 0
0 0 0 0 0 0 0 0 0 0 0 0 0 45 186 253 253 150 27 0 0 0 0 0
0 0 0] [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 93 252 253 187
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 0 249
253 249 64 0 0 0 0 0 0 0] [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
46 130 183 253 253 207 2 0 0 0 0 0 0 0] [ 0 0 0 0 0 0 0 0
0 0 0 0 39 148 229 253 253 253 250 182 0 0 0 0 0 0 0 0] [
0 0 0 0 0 0 0 0 0 0 24 114 221 253 253 253 253 201 78 0 0
0 0 0 0 0 0 0] [ 0 0 0 0 0 0 0 0 23 66 213 253 253 253
253 198 81 2 0 0 0 0 0 0 0 0 0 0] [ 0 0 0 0 0 0 18 171
219 253 253 253 253 195 80 9 0 0 0 0 0 0 0 0 0 0 0 0] [ 0
0 0 0 55 172 226 253 253 253 253 244 133 11 0 0 0 0 0 0 0
0 0 0 0 0 0 0] [ 0 0 0 0 136 253 253 253 212 135 132 16 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 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]
```

Load python library

```
from sklearn.datasets import fetch_openml
import numpy as np
import matplotlib.pyplot as plt
```



API Reference > sklearn.datasets > fetch_openml

fetch_openml

```
sklearn.datasets.fetch_openml(name: str | None = None, *, version: str | int
= 'active', data_id: int | None = None, data_home: str | PathLike | None =
None, target_column: str | List | None = 'default-target', cache: bool = True,
return_X_y: bool = False, as_frame: str | bool = 'auto', n_retries: int = 3,
delay: float = 1.0, parser: str = 'auto', read_csv_kwargs: Dict | None =
None) #                                                              [source]
```

Fetch dataset from openml by name or dataset id.

Datasets are uniquely identified by either an integer ID or by a combination of name and version (i.e. there might be multiple versions of the ‘iris’ dataset). Please give either name or data_id (not both). In case a name is given, a version can also be provided.

Fetch MNIST dataset

```
from sklearn.datasets import fetch_openml
import numpy as np

X, y = fetch_openml('mnist_784', data_home='./', return_X_y=True)
```

Normalization

- Normalization
  - divide X by 255 to scale the input data into the range of 0 to 1 for better numerical stability
  - the original data is pixel intensities, hence between 0 and 255

```
X, y = fetch_openml('mnist_784', data_home='./',
return_X_y=True)

X = X / 255
```

Plot first 10 images

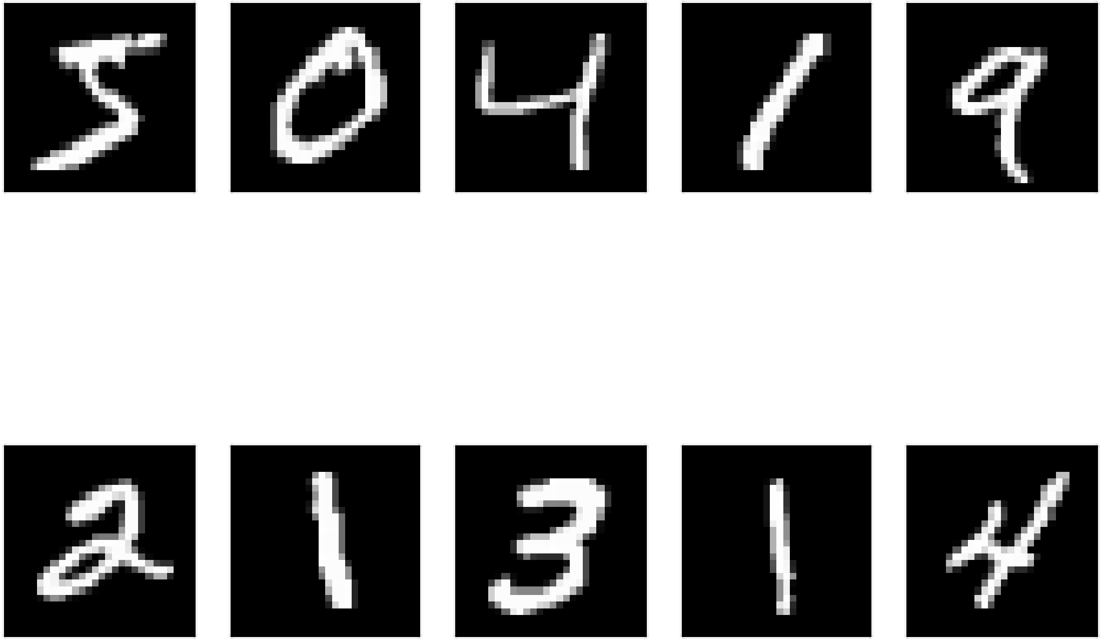
```
import matplotlib.pyplot as plt

X_small = X[:1000]
y_small = y[:1000]


# display the first 10 digits
plt.figure()

for i in range(10):
   plt.subplot(2, 5, i+1)  # 2 rows, 5 columns

   plt.imshow(X_small[i].reshape((28, 28)), cmap='gray’) # the image data are saved as flattened 1D arrays, reshape back to the shape of 28 * 28 to reconstruct the image, ‘gray’ means grayscale image

   plt.xticks([]), plt.yticks([]) # hides the tick marks

plt.show()
```

Reshape

- Reshape
  - the image data are saved as flattened 1D arrays (784-dimensional), reshape back to the shape of 28 * 28 to reconstruct the image

```
x = x.reshape(28, 28)
```

Plot first 10 images



Save MNIST dataset

- **save first 1000 images from MINST**

```
# load library
from sklearn.datasets import fetch_openml
import numpy as np
import matplotlib.pyplot as plt

# download dataset
X, y = fetch_openml('mnist_784', data_home='./', return_X_y=True)
X = X / 255

# select first 1000 images
X_small = X[:1000]
y_small = y[:1000]
X_small = X_small.values
y_small = y_small.values

# save to .npz file named ‘1k.npz’
np.savez('1k.npz', X_small=X_small, y_small=y_small)
```

Load MNIST dataset

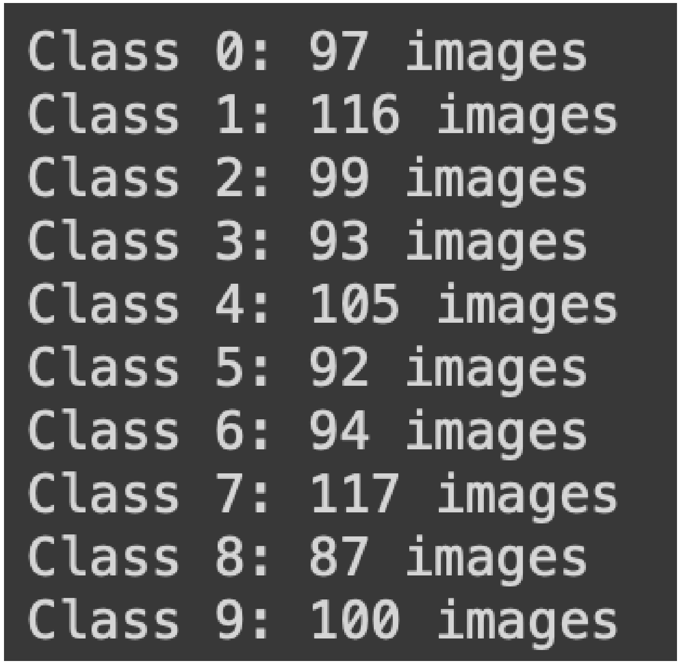
- **load first 1000 images from ‘1k.npz’**

```
# load from .npz file
data = np.load(‘1k.npz’, allow_pickle=True)

# access the X_small, y_small
X_small = data['X_small']
y_small = data['y_small’]

# print the x,y shape to check
print(X_small.shape)    # expected (1000,784)
print(y_small.shape)    # expected (1000,)
```

Count the number of images

- **count the number of images**

```
unique_classes, counts = np.unique(y_small, return_counts=True)
for cls, count in zip(unique_classes, counts):
    print(f"Class {cls}: {count} images")
```

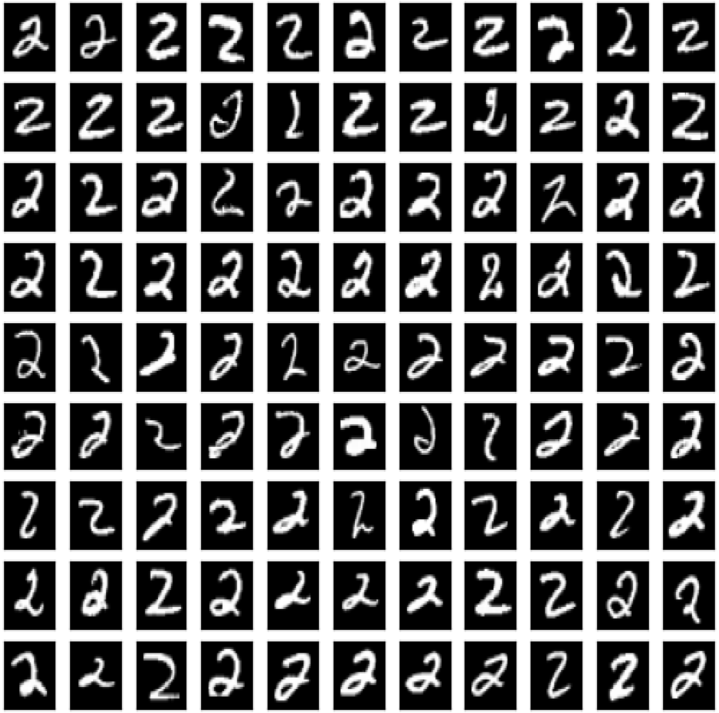


```
Class 0: 97 images
Class 1: 116 images
Class 2: 99 images
Class 3: 93 images
Class 4: 105 images
Class 5: 92 images
Class 6: 94 images
Class 7: 117 images
Class 8: 87 images
Class 9: 100 images
```

Display first 99 images of ‘2’

- **Plot first 99 images of digit 2**

```
# index of digit 2
index = np.nonzero(y_small == '2')[0]

row = 9
column = 11
counter = 0

for _ in range(row):
        for _ in range(column):
                plt.subplot(row, column, counter + 1)

                x = X_small[index[counter]]
                x = x.reshape(28, 28)
                plt.imshow(x, cmap='gray’)

                plt.xticks([])
                plt.yticks([])
               counter += 1
plt.show()
```



Any Questions?