Machine Learning

Practice 3

Quan Minh Phan & Ngoc Hoang Luong

University of Information Technology

April 28, 2021

Table of Contents

1. Logistic Regression
2. Image Classification
3. Practice

Implement

Syntax (import)

from sklearn.linear_model import LogisticRegression

Hyperparameters

- max_iter → the maximum iterations for training model
- random_state

Examples

▷ from sklearn.linear_model import LogisticRegression

▷ clf = LogisticRegression(max_iter=20000, random_state=1)

Implement (next)

Methods

- fit

Examples

▷ clf.fit(X_train, y_train)

Methods

- predict

Examples

▷ y_pred = clf.predict(X_test)

Table of Contents

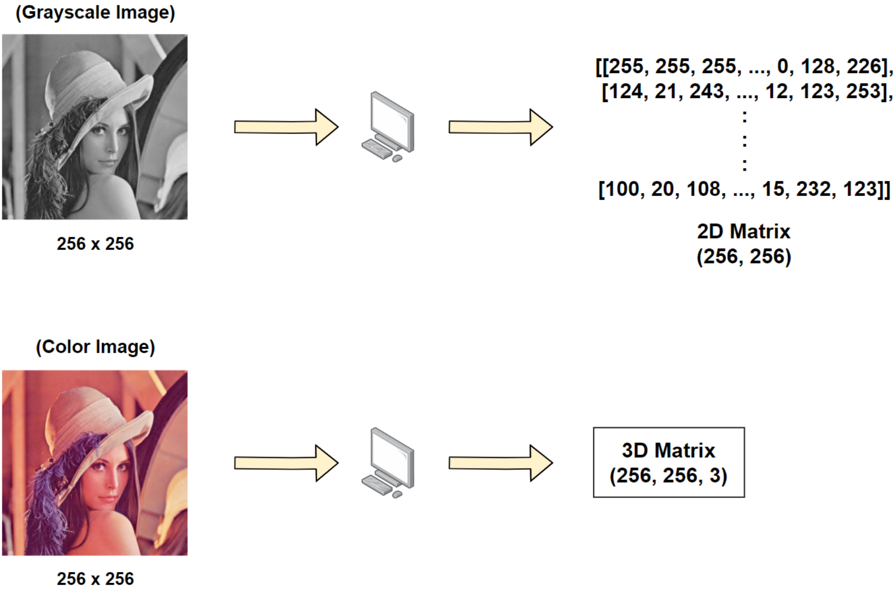
1 Logistic Regression

2 Image Classification

3 Practice

Introduction

Figure: lena.jpg


(Grayscale Image)
256 x 256
[[255, 255, 255, ..., 0, 128, 226],
[124, 21, 243, ..., 12, 123, 253],
:
:
:
[100, 20, 108, ..., 15, 232, 123]]
2D Matrix
(256, 256)
(Color Image)
256 x 256
3D Matrix
(256, 256, 3)

Each element is an integer in range [0, 255]

[[255, 255, 255, ..., 0, 128, 226],
[124, 21, 243, ..., 12, 123, 253],
:
:
:
[100, 20, 108, ..., 15, 232, 123]]

How to make the computer learn?

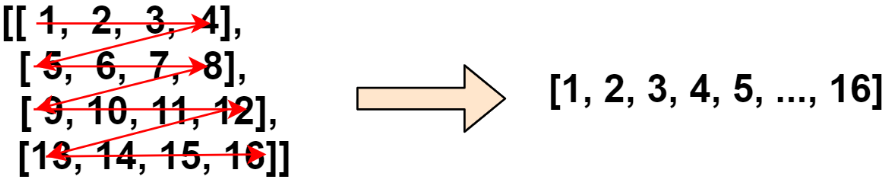
- Learning on all pixels.
- Feature extraction.
- ...

Learning on all pixels

Idea: Convert 2D matrix to 1D matrix by concatenating.

Example:

```
[[ 1,  2,  3,  4],
 [ 5,  6,  7,  8],
 [ 9, 10, 11, 12],
 [13, 14, 15, 16]]
```


[[1, 2, 3, 4],
[5, 6, 7, 8],
[9, 10, 11, 12],
[13, 14, 15, 16]]
[1, 2, 3, 4, 5, ..., 16]

Feature extraction

There are many feature extraction methods:

- Color-based (e.g., RGB histogram)
- Texture-based (e.g., Gabor filter)
- Shape-based (e.g., Edge histogram, H.O.G)
- ...

Histogram of Oriented Gradients (H.O.G)

Input image

Histogram of Oriented Gradients

Table of Contents

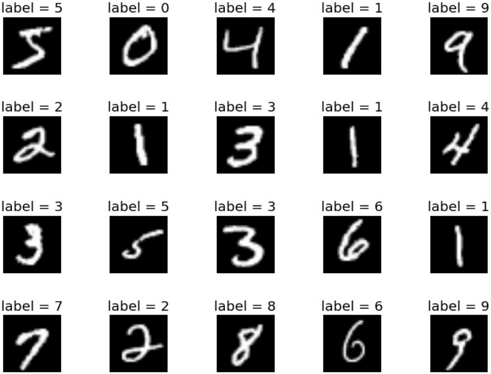
1 Logistic Regression

2 Image Classification

3 Practice

MNIST dataset

- MNIST is a large dataset of handwritten digits (0 to 9)
- Each image has the width 28 and the height 28 (28x28 pixels)
- 60.000 training images; 10.000 testing images.



Loading MNIST dataset

Loading data

- from keras.datasets import mnist
- (X_train, y_train), (X_test, y_test) = mnist.load_data()

Preprocessing (Learning on all pixels)

```
▷ import numpy as np
```

```
▷ def convert_to_1D (image):
      return np.reshape(image, image.shape[0] * image.shape[1])
```

```
▷ X_train_new = [ ]
```

```
▷ for x in X_train:
      X_train_new.append(convert_to_1D(x))
```

```
▷ X_test_new = [ ]
```

```
▷ for x in X_test:
      X_test_new.append(convert_to_1D(x))
```

Training model

Training

- from sklearn.linear_model import LogisticRegression

```
clf = LogisticRegression(max_iter=100, random_state=1, n_jobs=2)
```

- clf.fit(X_train_new, y_train)

Testing

- y_pred = clf.predict(X_test_new)

Evaluating

- from sklearn.metrics import accuracy_score, classification_report

- print('Accuracy:', accuracy_score(y_test, y_pred))

  ... Accuracy: 0.9255

- print(classification_report(y_test, y_pred))

Preprocessing (H.O.G)

```
▷ from skimage.feature import hog
```

```
▷ def calculate_hog (image):
    return hog(image, orientations=9, pixels_per_cell=(4, 4),
  cells_per_block=(2, 2), block_norm='L2')
```

```
▷ X_train_new_ = [ ]
```

```
▷ for x in X_train:
    X_train_new_.append(calculate_hog(x))
```

```
▷ X_test_new_ = [ ]
```

```
▷ for x in X_test:
    X_test_new_.append(calculate_hog(x))
```

Training, testing and evaluating model

Training

- ▷ clf_ = LogisticRegression(max_iter=100, random_state=1, n_jobs=2)

- ▷ clf_.fit(X_train_new_, y_train)

Testing

- ▷ y_pred_ = clf_.predict(X_test_new_)

Evaluating

- ▷ print('Accuracy:', accuracy_score(y_test, y_pred_))

  ... Accuracy: 0.9838 # improve 5.83% accuraccy

- ▷ print(classification_report(y_test, y_pred_))