KNN with MNIST Data

인공지능 2022년 1학기
Homework #2

개요

Iris data classification에 사용했던 지난 과제의 KNN 알고리즘을 mnist data에 적용해 보기

	Input feature data type	Input feature dimension	Training data size	Test data size
Iris	길이 (cm)	4	150 중 일부 (예: 140)	150 중 일부 (예: 10)
Mnist	이미지	784 (28x28)	60,000	10,000

MNIST Data 특성

머신러닝 최고 Guru중 한명인 뉴욕대 교수 `Yann LeCun`이 제공하는 데이터 셋

숫자 0~9까지의 손글씨 이미지의 집합

학습데이터 60,000개, 테스트데이터 10,000개로 구성

size-normalized & centered

사이즈: `28x28`

패턴인식이나 기계학습 기술을 적용하기 위해 사용할 수 있는 최적의 이미지 셋

preprocessing이나 formatting이 모두 완료 되었기 때문

Data download

밑바닥부터 시작하난 딥러닝 책 관련 Github

https://github.com/WegraLee/deep-learning-from-scratch

Dataset 폴더를 현재 프로젝트의 부모 폴더에 복사

Data load

```
import sys, os
sys.path.append(os.pardir)
# 부모 디렉토리에서 import할 수 있도록 설정

import numpy as np
from dataset.mnist import load_mnist
# mnist data load할 수 있는 함수 import

from PIL import Image
# python image processing library
# python 버전 3.x 에서는 pillow package install해서 사용
```

Data load

```
(x_train, t_train), (x_test, t_test) = \
    load_mnist(flatten=True, normalize=False)
# training data, test data
# flatten: 이미지를 1차원 배열로 읽음
# normalize: 0~1 실수로. 그렇지 않으면 0~255
```

```
image = x_train[0]
label = t_train[0]
# 첫번째 데이터

print(label)
print(image.shape)
```

Data Visualization

```
def img_show(img):
    pil_img = Image.fromarray(np.uint8(img))
    pil_img.show()
# image를 unsigned int로

image = image.reshape(28,28)
# 1차원 —> 2차원 (28x28)

print(image.shape)

img_show(image)
```

Data Classification

K-Nearest Neighbor 알고리즘 input

1. 784개 input을 그대로 사용

2. Input feature를 자신만의 방식으로 가공해서 차수를 줄여서 사용

Data Classification

1. 784개 input을 그래도 사용

결과가 그리 나쁘지는 않음
결과 계산에 오랜 시간이 소요됨
- KNN의 특성: 대부분의 계산이 학습보다는 inference에 소요됨

Output Example

3 3
1 1
5 5
1 1
4 4
7 7

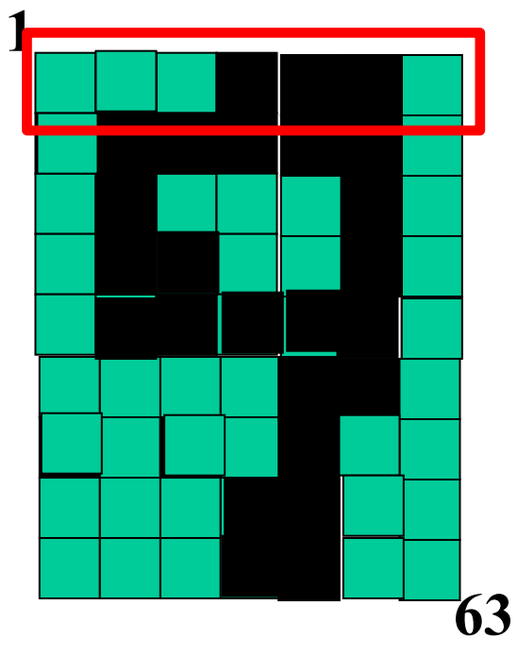
.....

1 1
0 0
6 6
4 4
6 6
9 9
1 1
8 8
2 2
accuracy = 0.95

(10000개 test data 중 100개 사용)

Data Classification

2. Input feature를 자신만의 방식으로 가공해서 차수를 줄여서 사용
각 행(row)/열(column)에서 배경이 아닌 숫자에 해당하는 픽셀 수
아래의 9x7 사례에서 1번째 행(row)의 경우: 3



Input feature dimension:
784(28x28) —> 56(28+28)

Hand-crafted feature: 이처럼 인간이 개입해서 성능이 좋을 것 같은 input feature를 만들어 사용하는 경우.
이번 과제의 핵심은 성능 좋은 Hand-crafted feature를 만드는 것
:: 그러나 너무 많은 시간 들이지 말기를…

Output Example

```
7948 th data  result  5    label  5
5396 th data  result  6    label  6
5934 th data  result  2    label  2
2165 th data  result  4    label  4
5719 th data  result  9    label  9
2337 th data  result  6    label  6
8183 th data  result  5    label  8
7246 th data  result  9    label  9
...
5705 th data  result  9    label  9
7243 th data  result  9    label  9
1294 th data  result  0    label  0
8575 th data  result  1    label  1
7866 th data  result  0    label  0
1909 th data  result  1    label  1
accuracy =  0.95
```

(10000개 test data 중 100개 사용)

주의할 점

- inference에 올래 걸리므로 일단 적은 양(3, 5, 9)의 데이터로 작동하는지 확인 후 많은 양의 데이터(100~1000)로 테스트

- test data 10,000개 중 일부를 랜덤하게 샘플링해서 사용할 것을 권장

```
size = 100
sample = np.random.randint(0, t_test.shape[0], size)

for i in sample:
    .....
```

- KNN의 3번째 파라미터 (label의 이름: 예 - [‘setosa’, ‘versicolor’, ‘verginica’])
다음의 리스트 사용

```
label_name = ['0','1','2','3','4','5','6','7','8','9']
```

Submission

- Source code (with comments) files
  - KNN class python file (수정한 내용이 없어도 다시 제출)
  - Main python file
  - Output results (report 에 포함해도 됨)
- Report (제공하는 양식 사용)

- Due
  - 4/21 (Thursday) 11pm
  - Late: 20% per day