[Lab 7]

제출과제

이번 실습에서는 은닉층이 한 개인 MLP를 구현하고 mnist 데이터에 대해 성능을 평가합니다.

교재 알고리즘 3-4와 알고리즘 3-7을 참고하여 구현합니다.

알고리즘 3-4 다층 퍼셉트론 학습을 위한 스토캐스틱 경사 하강법

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ
출력: 가중치 행렬 $\mathbf{U}^1$과 $\mathbf{U}^2$

1 $\mathbf{U}^1$과 $\mathbf{U}^2$를 초기화한다.
2 repeat
3 $\mathbb{X}$의 순서를 섞는다.
4 for ($\mathbb{X}$의 샘플 각각에 대해)
5 현재 처리하는 샘플을 $\mathbf{x}=(x_0,x_1,x_2,\cdots,x_d)^T$, $\mathbf{y}=(y_1,y_2,\cdots,y_c)^T$라 표기한다.
6 x_0과 z_0을 1로 설정한다. // 바이어스
  // 전방 계산
7 for (j=1 to p) $zsum_j=\mathbf{u}_j^1\mathbf{x}$, $z_j=\tau(zsum_j)$ // 식 (3.13)
8 for (k=1 to c) $osum_k=\mathbf{u}_k^2\mathbf{z}$, $o_k=\tau(osum_k)$ // 식 (3.14)
  // 오류 역전파
9 for (k=1 to c) $\delta_k=(y_k-o_k)\tau'(osum_k)$ // 식 (3.22)
10 for (k=1 to c) for (j=0 to p) $\Delta u_{kj}^2=-\delta_k z_j$ // 식 (3.23)
11 for (j=1 to p) $\eta_j=\tau'(zsum_j)\sum_{q=1}^{c}\delta_q u_{qj}^2$ // 식 (3.24)
12 for (j=1 to p) for (i=0 to d) $\Delta u_{ji}^1=-\eta_j x_i$ // 식 (3.25)
  // 가중치 갱신
13 for (k=1 to c) for (j=0 to p) $u_{kj}^2=u_{kj}^2-\rho\Delta u_{kj}^2$ // 식 (3.21)
14 for (j=1 to p) for (i=0 to d) $u_{ji}^1=u_{ji}^1-\rho\Delta u_{ji}^1$ // 식 (3.21)
15 until (멈춤 조건)

알고리즘 3-7 다층 퍼셉트론을 이용한 인식

입력: 테스트 샘플 x // 신경망의 가중치 $\mathbf{U}^1$과 $\mathbf{U}^2$는 이미 설정되었다고 가정함.
출력: 부류 y

1 $\mathbf{x}=(x_0,x_1,x_2,\cdots,x_d)^T$로 확장하고, x_0과 z_0을 1로 설정한다.
2 $\mathbf{zsum}=\mathbf{U}^1\mathbf{x}$
3 $\tilde{\mathbf{z}}=\boldsymbol{\tau}(\mathbf{zsum})$ // 식 (3.13)
4 $\mathbf{osum}=\mathbf{U}^2\mathbf{z}$
5 $\mathbf{o}=\boldsymbol{\tau}(\mathbf{osum})$ // 식 (3.14)
6 $\mathbf{o}$에서 가장 큰 값을 가지는 노드에 해당하는 부류 번호를 y에 대입한다.

제출과제

1) 지난 실습에서 사용한 MNIST 데이터를 사용합니다.
2) MNIST의 데이터 중 1, 5, 8의 클래스 레이블을 가진 데이터만 사용합니다.
3) 사용하는 데이터를 3:2로 나누어 학습/테스트 데이터로 사용합니다.
4) 종료조건은 총 수행 epoch수(예를 들어 5나 10의 epoch)를 설정하여 하면 됩니다.
5) Scikit-learn 라이브러리 사용 불가
6) Numpy나 파이썬 내장 함수 사용

보고서에 작성할 내용

1) 사용한 목적함수, 은닉층의 노드 개수, activation function에 대해서 설명합니다.
2) 알고리즘 3-4의 실행에서는 각 epoch마다 가중치를 프린트합니다.
3) 학습이 끝난 후 알고리즘 3-7에 의한 학습데이터와 테스트데이터에 대한 예측정확도를 프린트합니다.

제출과제

구현에 어려움을 느낄 경우 과제를 단순하게 하여 수행해도 됩니다.
즉, 은닉층을 두지 않고, 입력층과 출력층만 설정하여 코드를 작성해도 됩니다.

<코드 작성 예>
- 필요한 가중치 변수를 선언하고 랜덤하게 초기화하고, 학습과 테스트 데이터, 그것들의 클래스 레이블을 저장할 변수들을 글로벌하게 선언
- MNIST데이터 전처리 과정을 수행하는 함수 [A]를 작성 – 학습과 테스트 데이터 구성
- 알고리즘 3-4 구현 함수 [B] 작성
- 알고리즘 3-7 구현 함수 [C] 작성

메인 함수에서
① 함수 [A] 수행
② 함수 [B] 수행 (학습 데이터와 클래스 레이블을 사용)
③ 함수 [C] 수행 (학습 데이터에 대한 예측 정확도 평가)
④ 함수 [C] 수행 (테스트 데이터에 대한 예측 정확도 평가)

제출과제

- 제출 기한
  - 2주 기간의 과제입니다.
  - 1차 : 11월 3일 자정 (10점 만점 채점)
  - 2차 : 11월 10일 자정 (7점 만점 채점)
  - 이후 제출은 미제출로 간주 (점수 없음)

Numpy

- import numpy as np
- https://numpy.org/

1) 배열
2) 인덱싱과 슬라이싱
3) 연산
4) …

1. Array

1) Array

* 파이썬 기본 자료형 List와는 달리 동일한 Data Type만 넣을 수 있음.

- dtype 함수를 이용해서 Data Type 확인 가능

```
import numpy as np
test_array = np.array([1,4,5,6],float)

print(test_array)
-> [1. 4. 5. 6.]

print(test_array.dtype)
-> float64

print(type(test_array[3]))
-> <class 'numpy.float64'>
```

1. Array

2) shape, ndim, size

- shape : array의 shape를 반환해주는 함수
- ndim : array의 차원의 수를 반환해주는 함수
- size : array의 총 elements 개수를 반환해주는 함수

```
import numpy as np
test_array = np.array([[1,4,5,6],[3,4,5,1]])
print(test_array.shape)
```

-> (2, 4)

```
print(test_array.ndim)
```

-> 2

```
print(test_array.size)
```

-> 8

1. Array

3) reshape, flatten

(1) reshape

- Array의 shape를 재조정하는 함수
- Element 개수가 동일한 경우에만 재조정 가능

(2) flatten

- 다차원 배열을 1차원 배열로 변환해주는 함수

```
import numpy as np
test_array = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(test_array)
-> [[1. 2. 3. 4.]
 [5. 6. 7. 8.]]
print(test_array.reshape((8,)))  =>4,2
-> [1. 2. 3. 4. 5. 6. 7. 8.]
print(test_array.flatten())
-> [1. 2. 3. 4. 5. 6. 7. 8.]
```

2. Indexing & Slicing

* Slicing의 경우 끝 번호 +1

1) 1차원 array의 인덱싱과 슬라이싱의 경우 list와 동일합니다

2) 2차원 array의 인덱싱과 슬라이싱

-> matrix의 경우 (row,column)의 형태

```
test_array = np.array([[1, 2, 3, 4], [5, 6, 7, 8]], float)
print(test_array[0,0])
```

-> 1.0

```
print(test_array[0][0])
```

-> 1.0

```
test_array[1,3] = 30
print(test_array)
```

->[[1. 2. 3. 4.]

 [5. 6. 7. 30.]]

```
print(test_array[:1,:])
```

-> [[1. 2. 3. 4.]]

```
print(test_array[1,1:3])
```

-> [6. 7.]

3. arange

- 파이썬 기본 문법의 range함수와 유사
- 범위를 지정하여 순차적인 값의 배열을 생성
- np.arange(시작, 끝 번호+1, 간격)
- reshape 함수와 함께 주로 사용

```
print(np.arange(10))
```

-> [0 1 2 3 4 5 6 7 8 9]

```
print(np.arange(0,11))
```

->[0 1 2 3 4 5 6 7 8 9 10]

```
print(np.arange(0,11,2))
```

-> [0 2 4 6 8 10]

```
print(np.arange(10).reshape(2,5))
```

-> [[0 1 2 3 4]
 [5 6 7 8 9]]

4. 배열들의 사칙연산

- shape가 같은 배열들 간의 기본적인 사칙 연산 가능
- Matrix multiplication : 일반적인 매트릭스 간 곱셈은 dot 함수를 이용

```
a = np.array([[1, 2], [3, 4]])

b= np.array([[1, 1], [-1, -1]])

c = np.array([[1, 1,1], [-1, -1, -1]])
print(a + b)
```

-> [[2 3]

[2 3]]

print(a * b) -> '*'기호 사용은 대응하는 요소들끼리의 곱셈

-> [[1 2]

[-3 -4]]

print(a.dot(c))) -> np.dot(a,c)와 동일하다

-> [[-1 -1 -1]

[-1 -1 -1]]

5. 랜덤한 변수 배열 생성

.randint : low ~ high-1 사이의 size 크기의 무작위 정수 배열을 반환합니다. Size가 생략되면 하나의 값이 리턴 됩니다.

```
np.random.randint(low, high=None, size=None, dtype=None)
a = np.random.randint(2,5,(2,4))
-> [[3 2 4 2]
 [4 2 4 3]]
```

.normal : 정규분포의 평균과 표준편차 값으로 배열을 리턴 합니다. Size가 생략되면 하나의 값이 리턴 됩니다.

loc : 정규분포의 평균

scale : 표준 편차

```
np.random.normal(loc=0.0, scale=1.0, size=None)
np.random.normal(loc=5, scale=1, size=(2,4))
 -> [[3.78961543 4.22572564 7.79955269 3.80673143]
[5.19656525 4.73509365 3.49113281 3.62683811]]
```

5. 랜덤한 변수 배열 생성

- .uniform : low와 high 사이의 균일한 분포의 무작위 배열을 리턴 합니다. Size가 생략되면 하나의 값이 리턴 됩니다.
- low: 출력 값의 low값
- high : 출력 값의 high값

```
np.random.uniform(low=0.0, high =1.0, size=None)
np.random.uniform(low=1, high=2, size=(2,4))
 -> [[1.38258323 1.75212112 1.39067667 1.69771935]
[1.13027761 1.39606535 1.61935355 1.93585917]]
```

6. 배열 비교

- all : 배열의 모든 요소가 조건에 만족할 때만 True 반환
- any : 배열의 한 요소만이라도 조건에 만족하면 True 반환

```
a = np.arange(10)
print(np.any(a>5),np.any(a<0))
```

-> True False

```
print(np.all(a>5),np.all(a<10))
```

-> False True

```
test_a = np.array([1,3,5])
test_b = np.array([4,2,0])
print(test_a>test_b)
```

-> [False True True]

```
print(test_a==test_b)
```

-> [False False False]

```
print(test_a<test_b)
```

-> [True False False]

6. 배열 비교

- where : 조건에 따라서 값을 할당해주거나 조건에 맞는 index값을 리턴 받을 때 사용
- where(condition, True, False)

```
a = np.arange(1,11)
print(a)
```

-> [1 2 3 4 5 6 7 8 9 10]

```
print(np.where(a<5,0,1))
```

-> [0 0 0 0 1 1 1 1 1 1]

```
print(np.where(a<5))
```

-> (array([0, 1, 2, 3], dtype=int32),)

- np.isnan() : 원소가 nan값인지를 검사하여 True, False 반환
- np.argmax() : 배열 안의 최대값의 index 반환 (array.max() : 최대값 반환)
- np.argmin() : 배열 안의 최소값의 index 반환 (array.min() : 최소값 반환)

7. 파일 불러오기와 저장

- 파일 읽기

numpy.loadtxt(fname, dtype=<class 'float', comments='#', delimiter=' ', conerters=None, skiprows=0, usecols=None, unpack=False, ndmin=0, encoding='bytes', max_rows=None>
numpy.loadtxt({파일 이름}, delimiter=",")

- 파일 저장

numpy.savetxt(filename, X, fmt='%.18e', delimiter=' ', newline='n', header='', footer='', - comment='#', encoding=None)
numpy.savetxt({파일이름}, {데이터}, fmt={데이터 형식}, delimiter={데이터간 구분자})
* csv 형태의 파일 저장을 위해서 delimiter 값을 ','로 지정

- 파이썬 기본 라이브러리 os라이브러리 활용

os.getcwd() : 현재 작업 디렉토리 위치
os.chdir() : 변경하려고 하는 디렉토리의 경로