

# Artificial Intelligence: Project 2

Seung-Hoon Na

May 16, 2023

## 1 Numpy: Tutorial

다음 자료를 참조하여 numpy기본을 공부하시오.

- <https://docs.scipy.org/doc/numpy-1.15.0/user/quickstart.html>
- <https://www.machinelearningplus.com/python/numpy-tutorial-part1-array-python-examples/>
- <http://web.mit.edu/dvp/Public/numpybook.pdf>
- <https://docs.scipy.org/doc/numpy-1.11.0/numpy-user-1.11.0.pdf>

### 1.1 Numpy: dot, einsum

다음 numpy의 dot의 정의를 이해하고, 예제를 만들어 설명하시오.

- <https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.dot.html>

다음 numpy의 einsum의 정의를 이해하고, 예제를 만들어 설명하시오. 또한 dot과의 관련도를 예를 들어 설명하시오.

- <https://docs.scipy.org/doc/numpy/reference/generated/numpy.einsum.html>

### 1.2 Numpy: quiz풀이

다음 퀴즈의 난이도 3단계의 문항 5개이상을 해결하시오.

<https://github.com/rougier/numpy-100>

### 1.3 Numpy: 선형방정식 풀기

$\mathbf{A}$ 가 정방행렬일때, 다음 선형방정식에서 해  $\mathbf{x}$ 를 구하는 numpy코드 (linearsol.py)를 작성하시오. (main에서는 코드에 대한 테스트도 포함)

$$\mathbf{Ax} = \mathbf{b}$$

역행렬을 구하는 함수는 다음을 참조하라.

<https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.linalg.inv.html>

$\mathbf{A}$ 가 정방행렬이 아닌 일반행렬일때 ( $\mathbf{A} \in \mathbb{R}^{m \times n}$ ), Pseudo inverse를 이용하여 위의 선형방정식의 근사해 (least squares solution)  $\mathbf{x}$ 를 구하는 numpy코드 (least\_squaresol.py)를 작성하시오. (main에서는 코드에 대한 테스트도 포함)

## 2 Linear regression

**Linear regression** (선형회귀)는  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ 가 학습데이터로 주어지고, 각각의 target value는 실수값으로 주어질때 (i.e.,  $y_i \in \mathbb{R}$ ), 입력  $\mathbf{x}$ 와 target value간의 관계를 다음과 같이 선형함수로 모델링하는 방법이다.

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

여기서,  $\mathbf{w}$ 와  $b$ 는 파라미터이다.

**최소제곱법 (least squares methods)**는 파라미터  $\mathbf{w}$ 와  $b$ 을 학습하기 위해  $\mathcal{D}$ 상의 선형함수의 **Loss function**(손실함수)  $J$ 로 다음과 같이 **square error의 합** (the sum of squares of the errors)을 사용한다.

$$\begin{aligned} J &= \sum_i (f(\mathbf{x}_i) - y_i)^2 \\ &= \sum_i (\mathbf{w}^T \mathbf{x}_i + b - y_i)^2 \end{aligned} \quad (1)$$

### 2.1 Linear regression을 위한 식 유도

식 1를 최소로 하는  $\mathbf{w}$ 와  $b$ 를 구하기 위해  $\partial J / \partial \mathbf{w}$ 와  $\partial J / \partial b$ 를 유도하시오. (**행렬에 대한 미분공식**을 이용하여 유도할 것)

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}} &= \\ \frac{\partial J}{\partial v} &= \end{aligned}$$

최종 유도식에서, 가능할경우 minibatch또는 batch갯수만큼의 전체 데이터벡터 전체를 다음 정의에 따라  $\mathbf{X}$ 와  $\mathbf{y}$ 를 참조하여 단순화하시오.

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_m \end{bmatrix} \\ \mathbf{y} &= \begin{bmatrix} y_1 & \cdots & y_m \end{bmatrix}^T \end{aligned}$$

### 2.2 Linear regression 학습을 위한 Stochastic Gradient Descent (SGD) Method

Loss function  $J$ 와 파라미터  $\theta$ 가 주어질때,  $\theta$ 를 학습하기 위한 Gradient Descent Method를 위한 핵심 수식은 다음과 같다.

$$\theta \leftarrow \theta - \eta \frac{\partial J}{\partial \theta}$$

여기서,  $\eta$ 는 learning rate (학습율)이다.

앞서 유도한  $\partial J / \partial \mathbf{w}$ 와  $\partial J / \partial b$ 를 이용하여 Linear regression 학습을 위한  $m$ 개의 minibatch단위의 Stochastic Gradient Descent (SGD) Method 의 알고리즘의 pseudo code를 기술하시오.

## 2.3 Linear regression 학습을 위한 Stochastic Gradient Descent Method: Early stopping 추가

Early stopping 방식은 무엇인지 조사하고, 위의 SGD 알고리즘에 Early stopping 추가한 pseudo code를 기술하시오.

아래외에 다른 문헌을 참고해도 좋음.

<https://deeplearning4j.org/docs/latest/deeplearning4j-nn-early-stopping>

[https://scikit-learn.org/stable/auto\\_examples/linear\\_model/plot\\_sgd\\_early\\_stopping.html](https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_early_stopping.html)

[https://page.mi.fu-berlin.de/prechelt/Biblio/stop\\_tricks1997.pdf](https://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf)

## 2.4 Linear regression 학습을 위한 Stochastic Gradient Descent Method 구현

지금까지 유도한 early stopping을 사용한 minibatch-SGD 방법을 numpy 패키지를 이용하여 구현하시오 (linear\_regression.py 코드 제출).

구현된 모듈을 테스트 하기 위해 다음과 같이 랜덤으로 생성된 데이터와 scikit의 샘플 데이터 각각에 대하여 테스트해보시오.

### 2.4.1 랜덤 데이터 생성기 구현: Gaussian 분포에 기반

다음 과정을 따르는 랜덤 데이터 생성기를 구현하시오 (gen\_random\_dataset.py 코드 제출).

1. true 파라미터 값의 랜덤 할당: 먼저 true 파라미터  $\mathbf{w}$ ,  $b$  값을 랜덤하게 부여한다 (생성단계에서는 알고 있지만, 학습단계에서는 모른다고 가정).  $[-R, R]$  구간내 uniform 분포를 따르도록 랜덤 생성하여 이를  $\mathbf{w}$ ,  $b$ 의 값으로 할당한다 ( $R$ 은 10정도가 적당하며, 다른 값으로 셋팅해도 된다). 다시 말해, 파라미터  $\mathbf{w} \in \mathbb{R}^d$ ,  $b$ 들을 random variables로 간주하고 다음을 따라 sampling을 수행한다.

$$\begin{aligned}\mathbf{w} &\sim \mathbf{U}[-R, R]^d \\ b &\sim \mathbf{U}[-R, R]\end{aligned}$$

여기서,  $\mathbf{U}[-R, R]^d$ 는  $d$ 차원 hypercube 공간 ( $[-R, R]^d$ ,  $d$ -cube) 상에서 uniform distribution을 의미한다.

2. 데이터셋 생성: 총  $N$ 개의 데이터 (training/dev/test set 포함)를  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ 를 다음과 같이 생성한다. ( $N$ 은 1000개 이상)

$$\begin{aligned}\mathbf{x}_i &\sim \mathbf{U}[-R, R]^d \\ y_i &\sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_i + b, \sigma^2)\end{aligned}$$

여기서,  $\mathbf{w}$ ,  $b$ 은 이전단계에서 랜덤하게 할당된 true 파라미터 값들이며,  $\mathcal{N}(\mu, \sigma^2)$ 는 평균  $\mu$ , 분산 (variance)  $\sigma^2$ 인 Gaussian distribution을 의미한다. ( $\sigma = \alpha R$ 로 두고,  $\alpha$ 는 default로 0.1 정도로 셋팅하자.)

3. 데이터셋 분리: 총  $N$ 개의 데이터를 랜덤하게 85%를 학습데이터 (training set), 5%를 개발용데이터 (dev set), 10%를 평가용데이터 (test set)로 분리한다.

4. 데이터셋 저장: 이렇게 얻은 데이터를 numpy로 변환하고 pickle등을 통해 별도 파일로 저장한다 (myrandomdataset.pkl 파일로 저장).

총 데이터갯수는  $N = 1000, 10000, 100000$ 로 다양하게 설정하여, 다른 이름으로 저장하여 테스트 수행할 것.

#### 2.4.2 scikit 샘플 예제: diabets

$\mathcal{D}$ 을 위해 scikit에서 linear regression을 위한 sample 예제인 diabets를 이용하라.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes = datasets.load_diabetes()
```

위의 diabetes를 loading하는 샘플 코드이다. scikit 전체 예제 코드는 다음을 참조하라.

[https://scikit-learn.org/stable/auto\\_examples/linear\\_model/plot\\_ols.html](https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html)

#### 2.4.3 Linear regression 학습기 테스트

데이터셋  $\mathcal{D}$ 를 위의 두가지 유형의 1) 랜덤 데이터 로컬 파일, 2) scikit 샘플 데이터로부터 입력받아 training, dev, test sets 각각을 numpy 개체로 로딩한 이후에 구현한 Stochastic Gradient Descent Method를 테스트하시오.

테스트시 요구사항은 다음과 같다.

- **minibatch 크기 사용자 설정:**  $m$ 은 option으로 사용자가 설정하도록 하고, default로 10에서 100사이의 값을 택하여 사용하라.
- **매epoch마다 성능 출력:** 주어진 데이터셋 전체 예제를 한번 학습할때 (1 epoch시)마다 training, dev, test 상에서 **mean squared error**을 출력한다. 랜덤 데이터인 경우에 한해서, 파라미터  $w$ 와  $b$ 와 **true값과 학습된 값들간의 squared error**도 함께 출력하시오.
- **Early stopping 적용:** early stopping을 적용하여, dev set 상 성능 개선이 오랫동안 없는 경우 학습을 종료한다.
- **최대 epoch수 설정:** 최대 epoch수를 사용자가 설정할 수 있도록 하고, default값으로 100을 사용한다.

### 3 Logistic regression

클래스 갯수가  $K$ 개인 다중 클래스 분류 (multi-class classification) 상 데이터 샘플  $\mathbf{x}$ 에 대한 **Logistic regression**식은 다음과 같다.

$$\mathbf{o}(\mathbf{x}) = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

주어진 학습데이터  $(\mathbf{x}, k)$  ( $k$ 는 정답클래스이며,  $k \in \{1, \dots, K\}$ )에 대한 **negative log likelihood**식은 다음과 같다

$$J = -\mathbf{y}^T \log(\mathbf{o}) \quad (2)$$

이때,  $\mathbf{y}$ 는 정답에 대한 one-hot encoding 벡터로 다음과 같이 정의된다.

$$\mathbf{y} = [y_1 \cdots y_i \cdots y_K]^T$$

여기서,  $y_i = \mathcal{I}(i = k)$ 이고  $k$ 는 정답클래스이다.

### 3.1 Logistic regression 학습을 위한 식 유도

먼저,  $m$ 개의 학습데이터셋  $\mathcal{D} := \{(\mathbf{x}_i, y_i), \dots, (\mathbf{x}_m, y_m)\}$  ( $y_i$ 는 정답클래스,  $y_i \in \{1, \dots, K\}$ )이 주어졌을 때 식 2을 전체셋  $\mathcal{D}$ 에 대한 negative log-likelihood로 확장하시오.

파라미터  $\mathbf{W}$ 와  $\mathbf{b}$ 를 학습하기 위해  $\partial J / \partial \mathbf{W}$ 와  $\partial J / \partial \mathbf{b}$ 를 유도하시오.  
(행렬에 대한 미분공식을 이용하여 유도할 것)

$$\frac{\partial J}{\partial \mathbf{W}} =$$
$$\frac{\partial J}{\partial \mathbf{b}} =$$

### 3.2 Logistic regression 학습을 위한 Early stopping을 이용한 SGD Method

Linear regression에서처럼, Logistic regression 학습을 위한 minibatch 상에서의 early stopping을 이용한 SGD Method 알고리즘을 유도하고, pseudo code를 작성하시오.

### 3.3 Logistic regression 학습을 위한 Stochastic Gradient Descent Method 구현

지금까지 유도한 Logistic regression 학습을 위한 early stopping을 사용한 minibatch-SGD 방법을 numpy 패키지를 이용하여 구현하시오 (logistic\_regression.py 코드 제출).

구현된 모듈을 테스트 하기 위해 다음 scikit의 다음 샘플 데이터에 대하여 테스트해보시오. (logistic\_regression\_mnist.py 코드 제출)

1. MNIST: [http://neupy.com/2016/11/12/mnist\\_classification.html](http://neupy.com/2016/11/12/mnist_classification.html)

테스트시 요구사항은 다음과 같다.

- **minibatch 크기 사용자 설정:**  $m$ 은 option으로 사용자가 설정하도록 하고, default로 10에서 500사이의 값을 적절히 택하여 사용하라.
- **매 epoch마다 성능 출력:** 주어진 데이터셋 전체 예제를 한번 학습할때 (1 epoch시)마다 training, dev, test 상에서 **classification accuracy**를 출력한다.
- **Early stopping 적용:** early stopping을 적용하여, dev set 상 성능 개선이 오랫동안 없는 경우 학습을 종료한다.
- **최대 epoch 수 설정:** 최대 epoch 수를 사용자가 설정할 수 있도록 하고, default 값으로 100을 사용한다.

## 4 Multi-layer perceptron (MLP)

은닉층이 1개인 **Multi-layer perceptron (MLP)**는 입력벡터  $\mathbf{x}$ 을 **비선형 변환**한 은닉표상  $\mathbf{h}$ 에 logistic regression을 적용한 것으로 다음 수식을 따른다 (강의자료 참고):

$$\begin{aligned}\mathbf{h} &= \max(\mathbf{W}\mathbf{x} + \mathbf{b}, 0) \\ \mathbf{o} &= \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{d})\end{aligned}$$

Logistic regression과 마찬가지로 loss function으로 다음과 같이 negative log-likelihood를 사용한다.

$$J = -\mathbf{y}^T \log(\mathbf{o})$$

여기서  $\mathbf{y}$ 에 대한 정의는 logistic regression에서의 식 3과 같다.

### 4.1 MLP 학습을 위한 backpropagation식 유도

Logistic regression과 마찬가지로 은닉층이 1개인 MLP 학습을 위해  $m$ 개의 학습 데이터셋  $D := \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$  ( $y_i$ 는 정답클래스,  $y_i \in \{1, \dots, K\}$ )이 주어졌다고 하자.

식 3을 전체셋  $D$ 에 대한 **negative log-likelihood**로 확장하시오. (logistic regression 문제와 동일)

파라미터  $\mathbf{W}$ ,  $\mathbf{U}$ ,  $\mathbf{b}$ ,  $\mathbf{d}$ 를 학습하기 위해  $\partial J / \partial \mathbf{W}$ ,  $\partial J / \partial \mathbf{U}$ ,  $\partial J / \partial \mathbf{b}$ ,  $\partial J / \partial \mathbf{d}$ 를 계산하는 backpropagation 알고리즘을 유도하시오.

### 4.2 MLP 학습을 위한 backpropagation식 일반화

MLP를 은닉층을  $L$ 개로 확장하였을때의 backpropagation 알고리즘을 일반화하시오.

### 4.3 MLP 학습을 위한 Early stopping을 이용한 SGD Method

Logistic regression에서처럼, 은닉층  $L$ 개의 MLP 학습을 위한 **minibatch 상에서의 early stopping을 이용한 SGD Method** 알고리즘의 pseudo code를 작성하시오.

### 4.4 MLP 학습을 위한 Stochastic Gradient Descent Method 구현

지금까지 유도한 은닉층  $L$ 개의 **MLP 학습을 위한 early stopping에 기반한 minibatch SGD** 방법을 numpy 패키지를 이용하여 **구현**하시오 (MLP.py 코드 제출).

구현된 모듈을 테스트 하기 위해 다음 scikit의 다음 **샘플 데이터**에 대하여 테스트해보시오. (MLP\_mnist.py 코드 제출)

1. **MNIST**: [http://neupy.com/2016/11/12/mnist\\_classification.html](http://neupy.com/2016/11/12/mnist_classification.html)

학습기의 요구사항은 다음과 같다.

- **최종 학습된 모델 저장**: 최종 학습된 모델 (파라미터)는 이후에 로딩될 수 있도록 별도의 모델 파일에 저장해야 한다.

- **학습된 모델 로딩 및 테스트:** 학습된 모델 (파라미터)를 로딩하여, 별도 테스트셋에서 MLP를 적용하여 분류를 수행할 수 있도록 해야 한다. mnist test set을 별도로 추출하여 테스트 필요
- **은닉층의 갯수 사용자 설정:** 은닉층의 갯수  $L$ 은 사용자가 설정하도록 하고, default값은 1로 한다.
- **은닉층의 차원수 사용자 설정:** 은닉층의 차원은 각 은닉층마다 별도로 설정할 수 있도록 한다. 위의 은닉층 갯수와 합하여, 모든 은닉층의 차원수를 array로 입력 받으면 된다. 예를 들어,  $[100, 50, 30]$ 은 은닉층의 수가 총 3개이고, 1번째 은닉층의 차원은 100, 2번째는 50, 3번째는 30인 MLP이다.
- **minibatch 크기의 사용자 설정:**  $m$ 은 option으로 사용자가 설정하도록 하고, default로 10에서 500사이의 값을 적절히 택하여 사용하라.
- **매epoch마다 성능 출력:** 주어진 데이터셋 전체 예제를 한번 학습할때 (1 epoch시)마다 training, dev, test상에서 **classification accuracy**를 출력한다.
- **Early stopping적용:** early stopping을 적용하여, dev set상 성능 개선이 오랫동안 없는 경우 학습을 종료한다.
- **최대 epoch수 설정:** 최대 epoch수를 사용자가 설정할 수 있도록 하고, default값으로 100을 사용한다.

특히 본 문항에서는 은닉층의 갯수를 늘릴때 성능 변화를 보는 것으로, 최종 학습 결과 다음을 비교하시오.

- 은닉층의 갯수  $L = 1, 2, 3, 4$ 로 달리하였을 때 최종 학습된 모델의 test 셋에서의 classification accuracy비교.

## 5 제출 내용 및 평가 방식

코드는 python으로 본 과제 결과물로 필수적으로 제출해야 내용들은 다음과 같다.

- 코드 전체
- 테스트 결과: 각 내용별 테스트 코드 및 해당 로그 또는 출력 결과.
- 결과보고서: 구현 방법을 요약한 보고서.

본 과제의 평가항목 및 배점은 다음과 같다.

- 각 세부내용의 구현 정확성 및 완결성 (80점)
- 코드의 Readability 및 체계성 (10점)
- 결과 보고서의 구체성 및 완결성 (10점)