Machine Learning: Project 1

Seung-Hoon Na

November 21, 2021

1 (25 points) 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)

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

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 (leastsquaresol.py)를 작성하시오. (main에서는 코드에 대한 테스트도 포함)

2 (125 points) Linear regression: Numpy을 이용하여 구현하기

Linear regression 는 $\mathcal{D} = \{(\mathbf{x}_i, t_i)\}_{i=1}^{N}$가 학습데이터로 주어지고, 각각의 target value는 실수값으로 주어질때 (i.e., $t_i \in \mathbb{R}$), 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 b는 파라미터이다.
최소제곱법 (least squares methods)는 파라미터 $\mathbf{w}$와 b을 학습하기 위해 $\mathcal{D}$ 상의 선형함수의 **Loss function**(손실함수) J로 다음과 같이 **square error의 합** (the sum of squares of the errors)을 사용한다.

$$\begin{aligned} J &= \sum_{i=1}^{N} \left(f(\mathbf{x}_i) - t_i\right)^2 \\ &= \sum_{i=1}^{N} \left(\mathbf{w}^T\mathbf{x}_i + b - t_i\right)^2 \end{aligned} \tag{1}$$

2.1 (5 points) Linear regression을 위한 Parameter의 Gradient 식 정리

Assignment 1에서 문항 3을 참조하여, 식 4를 최소로 하는 $\mathbf{w}$와 b를 구하기 위해 $\partial J/\partial \mathbf{w}$와 $\partial J/\partial b$를 정리하시오.

$$\frac{\partial J}{\partial \mathbf{w}} =$$
$$\frac{\partial J}{\partial v} =$$

특히 Assignment 1의 문항 3-(ii)를 참조하여, 다음 Design matrix $\mathbf{X}$와 $\mathbf{t}$ 를 이용하여, 위의 Gradient를 행렬식으로 정리하시오.

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} \mathbf{x}_1{}^T \\ \vdots \\ \mathbf{x}_m{}^T \end{bmatrix} \\ \mathbf{t} &= [t_1, \cdots, t_N]^T \end{aligned} \tag{2}$$

2.2 (5 points) Linear regression학습을 위한 Stochastic Gradient Descent (SGD)Method

2.2.1 SGD를 이용한 parameter update식 정리

Loss function J와 파라미터 $\boldsymbol{\theta}$가 주어질때, SGD방법에서 $\boldsymbol{\theta}$를 update하는 식을 다음과 같다.

$$\boldsymbol{\theta} \quad \leftarrow \quad \boldsymbol{\theta} - \eta \, \mathbb{E}\left[\frac{\partial J}{\partial \boldsymbol{\theta}}\right] \tag{3}$$

여기서, η는 learning rate이다.
Minibatch에 기반한 SGD방식은 식 3의 $\mathbb{E}\left[\frac{\partial J}{\partial \boldsymbol{\theta}}\right]$을 다음과 같이 $\mathcal{D}$에서 m개의 training examples를 샘플링한 minibatch $\mathcal{D}_m \in \mathcal{D}$를 이용하여 근사한 것이다.

$$\mathbb{E}\left[\frac{\partial J}{\partial \boldsymbol{\theta}}\right] \sim \frac{\partial \sum_{\mathbf{x} \in \mathcal{D}_m} \left(f\left(\mathbf{x}\right) - t_i\right)^2 / m}{\partial \boldsymbol{\theta}}$$

여기서 $\mathcal{D}_m$는 매번 parameter update될때마다 샘플링된다[1].
앞서 유도한 $\partial J / \partial \mathbf{w}$와 $\partial J / \partial b$를 이용하여 Linear regression학습을 위한 m개의 **minibatch버전**의 **Stochastic Gradient Descent (SGD) Method**에 대한 pseudo code를 기술하시오.

2.2.2 (5 points) L2 regularization추가

2.2.1의 결과를 확장하여, 다음과 같이 L_2 regularization (bias는 제외)이 반영된 loss function J_λ를 최소화위해, SGD방법에 기반한 parameter update식을 정리하시오.

$$J_\lambda \quad = \quad \sum_{i=1}^{N} \left(f\left(\mathbf{x}_i\right) - t_i\right)^2 + \lambda \|\mathbf{w}\|^2 \tag{4}$$

2.2.3 (10 points) Early stopping추가

Early stopping방법이란 무언인지 그리고 generalization관점에서의 효과에 대해서 기술하시오. 전형적인 Early stopping방법을 조사하여, 2.2.2의 parameter update식을 바탕으로, **SGD방식에서 Early stopping을 반영한 전체 학습 프레임워크**에 대한 pseudo code (또는 개요)를 기술하시오.
아래외에 다른 문헌을 참고해도 좋음.
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

2.3 (100 points) [Main Project] Linear regression을 위한 SGD 구현

지금까지 정리한 2.2.3을 바탕으로 **early stopping을 사용한 minibatch-SGD 방법**을 numpy패키지를 이용하여 **구현**하시오 (linear_regression.py코드 제출).
구현된 모듈을 테스트 하기 위해 다음과 같이 **랜덤으로 생성된 데이터**와 scikit의 **샘플 데이터** 각각에 대하여 테스트해보시오.

[1] 보다 간편하고 샘플간의 중복성 없애는 방식은, 한 epoch의 학습 시작시에 $\mathcal{D}$를 최대 m examples로 구성되도록 랜덤하게 partition하여 (shuffling), 이들 partition을 minibatch로 취하는 방법이 일반적이다.

2.3.1 랜덤 데이터 생성기 구현: Gaussian분포에 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 uniform분포를 따르도록 랜덤생성하여 이를 $\mathbf{w}, b$의 값으로 할당한다 (R은 10정도가 적당하며, 다른 값으로 셋팅해도 된다). 다시 말해, 파라미터 $\mathbf{w} \in \mathbb{R}^d$, b들을 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 d차원 hypercube공간 ($[-R, R]^d$, d-cube)상에서 uniform distribution을 의미한다.

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

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

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

3. **데이터셋 분리**: 총 N개의 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 `numpy`로 변환하고 `pickle`등을 통해 별도 파일로 저장한다 (`myrandomdataset.pkl` 파일로 저장).

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

2.3.2 scikit 샘플 예제: diabets

$\mathcal{D}$을 위해 scikit에서 linear regression을 위한 sample 예제인 `diabets`를 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()
```

위는 `diabetes`를 loading하는 샘플 코드이다. scikit 전체 예제 코드는 다음을 참조하라.
`https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html`

2.3.3 Linear regression학습기 테스트

데이터셋 $\mathcal{D}$를 위의 두가지 유형의 1) 랜덤 데이터 로컬 파일, 2) scikit 샘플 데이터로부터 입력받아 `training`, `dev`, `test` sets각각을 `numpy`개체로 로딩한이후에 구현한 Stochastic Gradient Descent Method를 테스트하시오.
테스트시 요구사항은 다음과 같다.

- **minibatch 크기 사용자 설정**: m은 option으로 사용자가 설정하도록 하고, default로 10에서 100사이의 값을 택하여 사용하라.
- **매epoch마다 성능 출력**: 주어진 데이터셋 전체 예제를 한번 학습할때 (1 epoch시)마다 training, dev, test상에서 **mean squared error을 출력**한다. 랜덤 데이터인 경우에 한해서, 파라미터 `w`와 b와 **true값과 학습된 값들간의 squared error**도 함께 출력하시오.
- **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.

3 (150 points) Logistic regression

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

$$\begin{aligned} \mathbf{o} &= \mathbf{W}\mathbf{x} + \mathbf{b} \\ \mathbf{y} &= softmax\left(\mathbf{o}\right) \end{aligned}$$

주어진 학습데이터 $(\mathbf{x}, \mathbf{t})$에 대한 **negative log likelihood**식은 다음과 같다

$$J(\{(\mathbf{x}, \mathbf{t})\}) = -\mathbf{t}^T \ln\left(\mathbf{y}\right) \tag{5}$$

여기서 ln는 행렬의 각 element에 logarithm function을 element-wise로 적용하는 연산자이다.
이때, $\mathbf{t}$는 **정답에 대한 one-hot encoding벡터**로 다음과 같이 정의된다.

$$\mathbf{t} = \left[t_1 \cdots t_i \cdots t_K\right]^T$$

여기서 $t_i \in \{0, 1\}$, $\sum_i t_i = 1$이다.

3.1 (25 points) Logistic regression학습을 위한 식 정리

3.1.1 기본 Gradient식 정리

Minibatch 셋으로 m개의 학습데이터셋 $\mathcal{D}_m \coloneqq \{(\mathbf{x}_i, \mathbf{t}_i), \cdots, (\mathbf{x}_m, \mathbf{t}_m)\}$이 주어질 때, 식 5을 $\mathcal{D}_m$의 minibatch상의 negative log-likelihood로 확장하면 다음과 같다.

$$J(\mathcal{D}_m) = -\sum_{(\mathbf{x},\mathbf{t})\in\mathcal{D}_m} \mathbf{t}^T \ln\left(\mathbf{y}\right) \tag{6}$$

Assignment 1의 문항 19를 참조하여, 위의 $J(\mathcal{D}_m)$ 파라미터 $\mathbf{W}$와 $\mathbf{b}$를 학습하기 위해 $\partial J/\partial \mathbf{W}$와 $\partial J/\partial \mathbf{b}$를 행렬 표현식으로 정리하시오.
(**행렬에 대한 미분공식**을 이용하여 유도할 것)

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

3.1.2 L2정규화를 반영한 parameter update식 정리

L2 regularization를 반영하여 다음과 같이 확장된 logistic regression loss를 이용할때 해당 parameter update식을 정리하시오.

$$J_\lambda(\mathcal{D}_m) = -\sum_{(\mathbf{x},\mathbf{t})\in\mathcal{D}_m} \mathbf{t}^T \ln(\mathbf{y}) + \lambda\|\mathbf{w}\|^2 \tag{7}$$

3.2 Early stopping을 반영한 SGD Method

문항 2의 Linear regression에서와 마찬가지로 Logistic regression학습을 위해, **early stopping를 적용한 SGD Method**기반 전체 학습 프레임워크에 대한 pseudo code(또는 개요)를 정리하시오.

3.3 (125 points) [Main project] Logistic regression학습을 위한 Stochastic Gradient Descent Method 구현

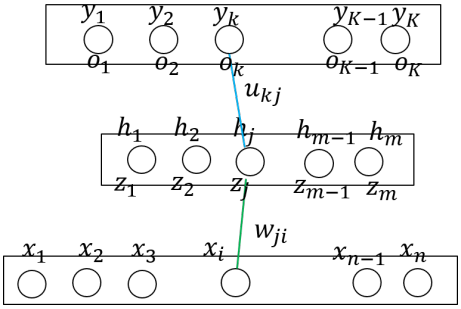
지금까지 유도한 **Logistic regression학습을 위한 early stopping을 사용한 minibatch-SGD** 방법을 `numpy`패키지를 이용하여 **구현**하시오 (`logistic_regression.py` 코드 제출).
구현된 모듈을 테스트 하기 위해 다음 `scikit`의 다음 **샘플 데이터**에 대하여 테스트해보시오. (`logistic_regression_mnist.py`코드 제출)

1. **`MNIST`**: `http://neupy.com/2016/11/12/mnist_classification.html`

테스트시 요구사항은 다음과 같다.

- **minibatch 크기 사용자 설정**: m은 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 accuracy를 출력**한다.
- **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.

Figure 1: 은닉층이 1개인 다층신경망 구조



4 (300 points) Multi-layer perceptron (MLP)

그림 1에서 보다시피, 은닉층이 1개인 **Multi-layer perceptron (MLP)**는 입력벡터 $\mathbf{x}$가 주어질때, nonlinear regression을 수행하여 $\mathbf{h}$을 얻은 후에 logistic regression을 적용한 모델이라고 할 수 있다. 은닉층이 L개인 MLP는 nonlinear regression을 반복적으로 수행한 후에, 최종 layer에서 logistic regression을 적용한 모델이다. 이때, nonlinear regression은 아래층의 표상에 대해 linear regression을 적용하고 activation function을 통해 비선형 변환을 수행하는 과정이다.
은닉층이 1개인 경우, activation function인 ReLU한 경우의 MLP모델의 수식을 정리하면 다음과 같다.

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

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

$$J = -\mathbf{t}^T \ln(\mathbf{y})$$

여기서 $\mathbf{t}$에 대한 정의는 logistic regression에서의 식 6과 같다.

4.1 (30 points) backpropagation유도를 위한 기본 규칙

다층 신경망의 backpropagation을 일반적으로 기술하기 위해, layer $l-1$의 post-activation표상을 $\mathbf{x}$, layer l의 pre-activation표상과 post-activation표상을 각각 $\mathbf{z}$와 $\mathbf{h}$라고 하자. 이때, 각각의 표상은 다음과 같이 구해진다

$$\begin{aligned}
\mathbf{z} &= \mathbf{W}\mathbf{x} + \mathbf{b} \\
\mathbf{h} &= f(\mathbf{z})
\end{aligned}$$

여기서 f는 *activation function*으로 미분가능한 element-wise 연산자이다. 다층 신경망은 위의 nonlinear regression기반 표상 변환 과정을 각 layer별로 별개의 파라미터와 activation함수를 가지면서 반복적으로 적용하는 모델이라 할 수 있다. Backpropagation에 대한 일반 규칙을 기술하기 위해, $\mathbf{x}$, $\mathbf{z}$와 $\langle$에 대한 loss function J의 gradient 를 각각 $\boldsymbol{\delta}_x$, $\boldsymbol{\delta}_z$와 $\boldsymbol{\delta}_h$로 표기하자.

$$\boldsymbol{\delta}_x = \frac{\partial J}{\partial \mathbf{x}}$$
$$\boldsymbol{\delta}_z = \frac{\partial J}{\partial \mathbf{z}}$$
$$\boldsymbol{\delta}_h = \frac{\partial J}{\partial \mathbf{h}}$$

한편, parameter vector에 대한 loss function J의 gradient 각각 $\Delta\mathbf{W}$, $\Delta\mathbf{b}$로 표기하자.

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

4.1.1 layer l의 post-activation delta vector로부터 pre-activation delta vector를 구하는 backprop 과정

$\boldsymbol{\delta}_z$를 $\boldsymbol{\delta}_h$와 f'를 이용하여 표현하시오.

4.1.2 layer l delta vector로부터 layer $l-1$ delta vector를 구하는 backprop 과정

$\boldsymbol{\delta}_x$를 $\boldsymbol{\delta}_z$와 $\mathbf{W}$로 표현하시오.

4.1.3 layer l의 delta vector가 주어질때 Parameter matrix update

$\Delta\mathbf{W}$와 $\Delta\mathbf{b}$를 $\boldsymbol{\delta}_z$를 이용하여 표현하시오.

4.2 (20 points) L개의 layer로 구성된 MLP의 backpropagation알고리즘

Logistic regression과 마찬가지로 은닉층이 L로 구성된 MLP의 parameter를 update하기 위해, m개의 minibatch 학습데이터셋 $\mathcal{D}_m := \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^m$가 주어졌다고 하자. 이때, 표기의 편의를 위해 입력벡터 $\mathbf{x}$와 최종 출력벡터 $\mathbf{y}$를 layer인덱스를 이용하여 다음과 같이 표현하자.

$$\mathbf{x} = \mathbf{h}^{(0)}$$
$$\mathbf{y} = softmax\left(\mathbf{z}^{(L)}\right) \tag{8}$$

또한 L2 regularization를 반영하여 식 7와 같이 다음 확장된 loss를 이용한다고 가정하자.

$$J_\lambda(\mathcal{D}_m) = -\sum_{(\mathbf{x},\mathbf{t})\in\mathcal{D}_m} \mathbf{t}^T \ln(\mathbf{y}) + \lambda\|\boldsymbol{\theta}\|^2 \tag{9}$$

여기서 $\boldsymbol{\theta}$는 MLP의 전체 파라미터 벡터로 모든 뉴런들의 연결 가중치 벡터와 bias을 concatenation한 벡터이다[2].
다음 물음에 답하시오.

- 앞의 문항 3으로부터 출력층에 대한 delta vector $\boldsymbol{\delta}_z^{(L)} = \frac{\partial J_\lambda(\mathcal{D}_m)}{\partial \mathbf{z}^{(L)}}$를 기술하시오.
- 문항 4.1에서 정리한 backpropagation 기본 규칙을 적용하여, 은닉층이 L개인 MLP의 backpropagation알고리즘을 구성하고, 이에 대한 Pseudo code를 작성하시오.
- 문항 3의 Logistic regression에서처럼, 은닉층 L개의 MLP학습을 위한 **mini-batch상**에서의 **early stopping을 적용한 SGD Method**기반 전체 학습 프레임워크의 pseudo code를 작성하시오.

4.3 (250 points) [Main project] MLP학습을 위한 Stochastic Gradient Descent Method 구현

지금까지 유도한 은닉층 L개의 MLP학습을 위한 early stopping기반 SGD방법을 `numpy`패키지를 이용하여 **구현**하시오 (`MLP.py`코드 제출).
구현된 모듈을 테스트 하기 위해 다음 `scikit`의 다음 **샘플 데이터**에 대하여 테스트해보시오. (`MLP_mnist.py`코드 제출)

1. **`MNIST`**: `http://neupy.com/2016/11/12/mnist_classification.html`

학습기의 요구사항은 다음과 같다.

- **최종 학습된 모델 저장**: 최종학습된 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 L은 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 `array`로 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**: m은 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 accuracy를 출력**한다.

[2]그림 1의 은닉칭이 1개인 MLP인 경우에 $\boldsymbol{\theta}$는 다음과 같다.

$$\boldsymbol{\theta} = [vec(\mathbf{W}); \mathbf{b}; vec(\mathbf{U}); \mathbf{d}] \tag{10}$$

여기서 ;는 column vector들을 concatenation하는 연산자를 의미한다.

- **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

본 과제는 일반문항 75점 및 구현문항 총 525점으로 구성되며, 구현문항 (총 525점)은 모두 **python**으로 작성되어야 하며, 구현 코드를 포함하여 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 (425점)**
- **코드의 Readability 및 체계성 (50점)**
- **결과 보고서의 구체성 및 완결성 (50점)**