Deep Learning Tutorial with Tensorflow

정상근

2016-02-18

Goal

- ✓ **TensorFlow 소개**

- ✓ **Hello World**
  - Deep Learning 의 기본 Block 구현 (Step by Step)

- ✓ **Sequence Learning**
  - Simple POS Tagger 구현
  - From Fully connected network (ignoring sentence)
  - From Fully connected network (sentence-wise)
  - Recurrent Neural Network
    - ➢ Sequence Loss 구현

Introduction

TENSORFLOW

Google Machine Learning Tools

1st Generation : *DistBelief*

- *Dean et al. 2011*
- *Major Output Products*
  - *Inception (Image Categorization)*
  - *Google Search*
  - *Google Translate*
  - *Google Photos*

2nd Generation : *TensorFlow*

- *Dean et al. 2015 (November, 1st)*
- *Most of DistBelief users at Google have already switched to TensorFlow*

[참고] Jeffrey Dean – Main Developers of *DistBelief* and *TensorFlow*

Jeffrey Adgate "Jeff" Dean (born 1968) is an American computer scientist and software engineer. He is currently a Google Senior Fellow in the Systems and Infrastructure Group.

- Advertising / Crawling / Indexing / Query Systems
- …

→ ***Google Core***

•BigTable	a large-scale semi-structured storage system.
•MapReduce	a system for large-scale data processing applications.

→ ***Hadoop***

•Google Brain	a system for large-scale artificial neural networks
•LevelDB	an open source on-disk key-value store.
•TensorFlow	an open source machine learning software library.
•…	

→ ***Large ML***

Referenced Paper

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems

White Paper version

Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, ´ Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, ´ Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

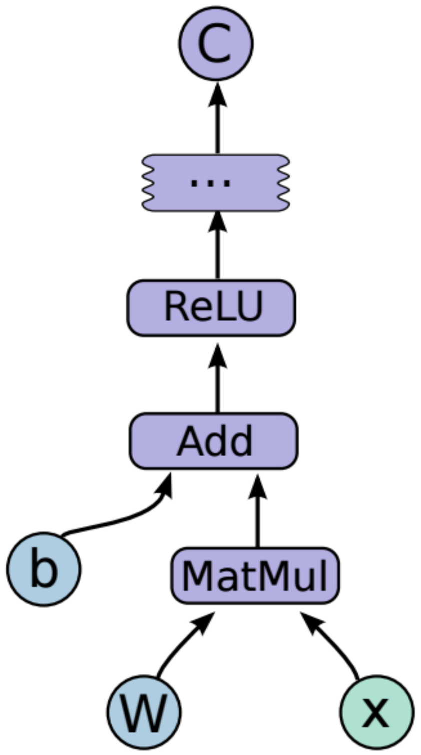
🟨 :: Author of *DistBelief* – 1st generation of Google ML infra.

Features

Programming Model	• Dataflow-like model	A directed Computational Graph 를 통해 모든 계산을 표현
Language	• Python • C++	현재는 Python, C++ 만 지원되나 FrontEnd 언어를 각 개발자가 만들면 되는 형태
Deployment	• Code once, Run everywhere	하나의 코드를 구현하면, 단-복수의 기계에서 똑같이 작동함
Computing Resource	• CPU • GPU	CPU 와 GPU 를 동시에 활용가능한 형태의 계산 Infra
Distribution Process	• Local Implementation • Distributed Implementation	Local, Distribution 2가지 모드에 대해 구현되어 있음. 옵션만 바꿔주면 TensorFlow가 알아서 작동됨
Math Expressions	• Math Graph Expression • Auto Differentiation	수식 및 계산 방식을 graph 형태로 표현. 따라서 자동으로 미분가능. 특히 Gradient 계산에 최적화해서 개발
Optimization	• Auto Elimination • Kernel Optimization • Communication Optimization • Support model, data parallelism • …	다양한 형태의 최적화를 TF가 알아서 진행함

Computational Graph

Figure 1: Example TensorFlow code fragment



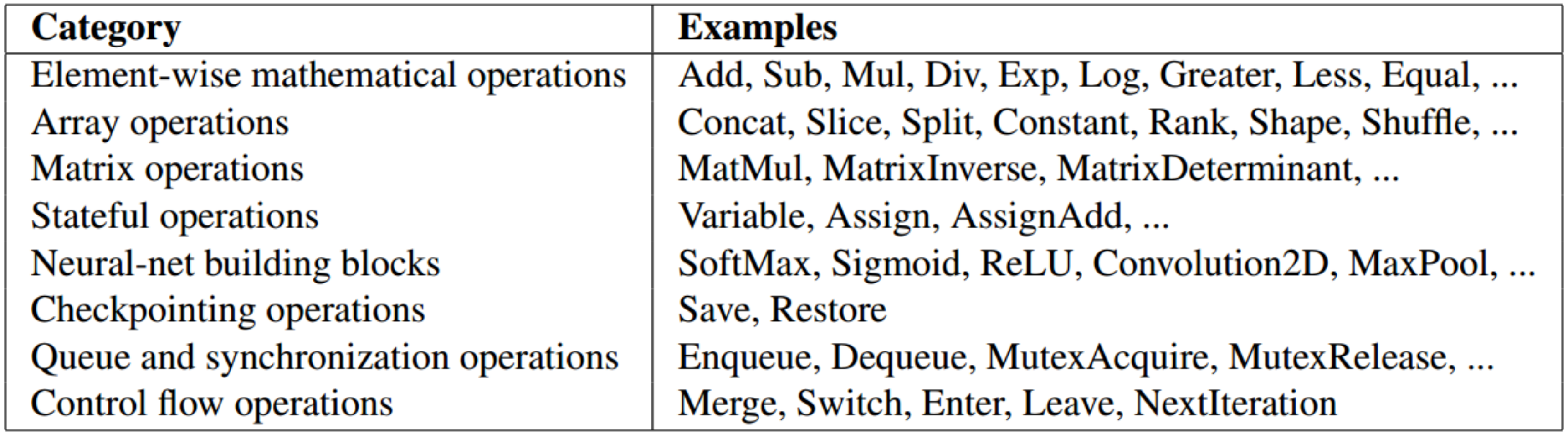
$$C = ReLU(W \cdot x + b)$$

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                   # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                       # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)             # Relu(Wx+b)
C = [...]                                          # Cost computed as a function
                                                   # of Relu
```

Operations and Kernels

Operations



Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

Kernels

각 계산과 관련하여 Device 혹은 Library 특화된 일종의 Thin Wrapper

:: Operation 과 Kernel 등은 “등록” 가능하게 되어 있음. 즉 외부 개발자들도 원한다면 특화된 기능을 넣을 수 있는 형태

Implementation modes

Local : Client, Master, Worker 가 모두 하나의 Machine 에 있는 경우

Distributed : Client, Master, Worker 가 서로 다른 기계의 서로 다른 Process에 있는 경우

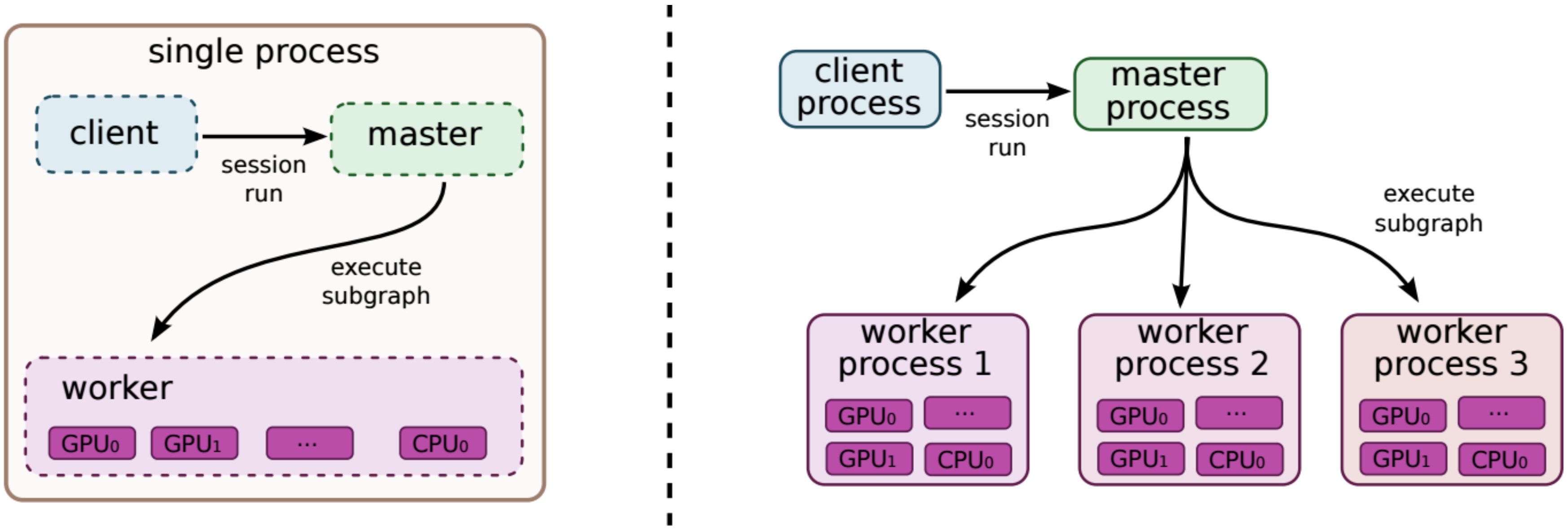


Figure 3: Single machine and distributed system structure

Hello World

DEEP LEARNING 의 기본 BLOCK 구현 (STEP BY STEP)

check mnist.py

Deep Learning 기본 Block

Statistical Machine Learning

- 가지고 있는 데이터를 이용하여
- 풀고자 하는 문제의 통계적 모델링을 통해
- 실제 정답과의 오차를
- 반복적인 파라미터 학습을 통해
- 줄여가는 것

Data 준비

Model 구현

Loss 구현

Updater 구현

Iterative Learning

Data 준비

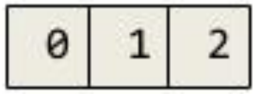
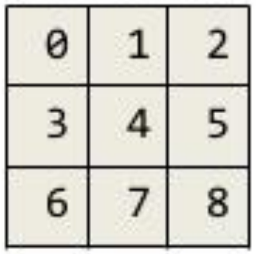
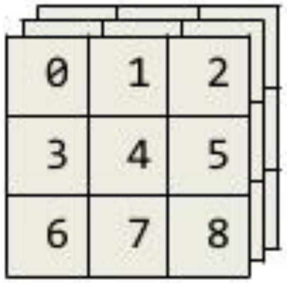
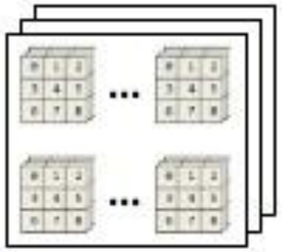
모든 것을 Tensor 단위로 생각!

음성도, 이미지도, 단어도, 문장도, 문서도
모두 ***N-dimensional array (tensor)*** 로 바꿔서 생각한다.

중요 개념

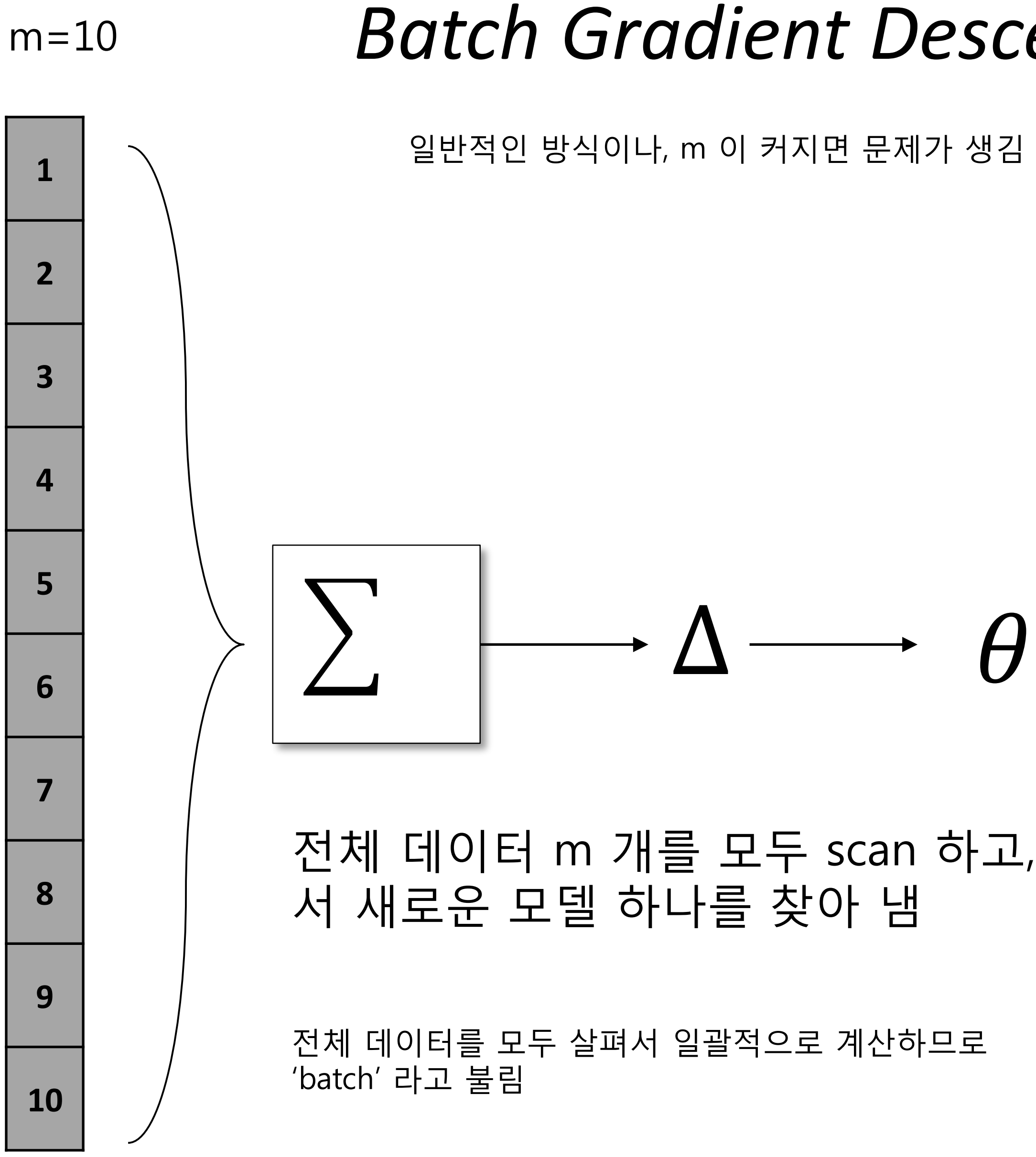
- *Tensor*
- *Epoch*
- *Batch*

What is an array?

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3rd order Tensor)
N		ND Array

From http://www.slideshare.net/mikeranderson/2013-1114-enter-thematrix

Training – Batch Gradient Descent



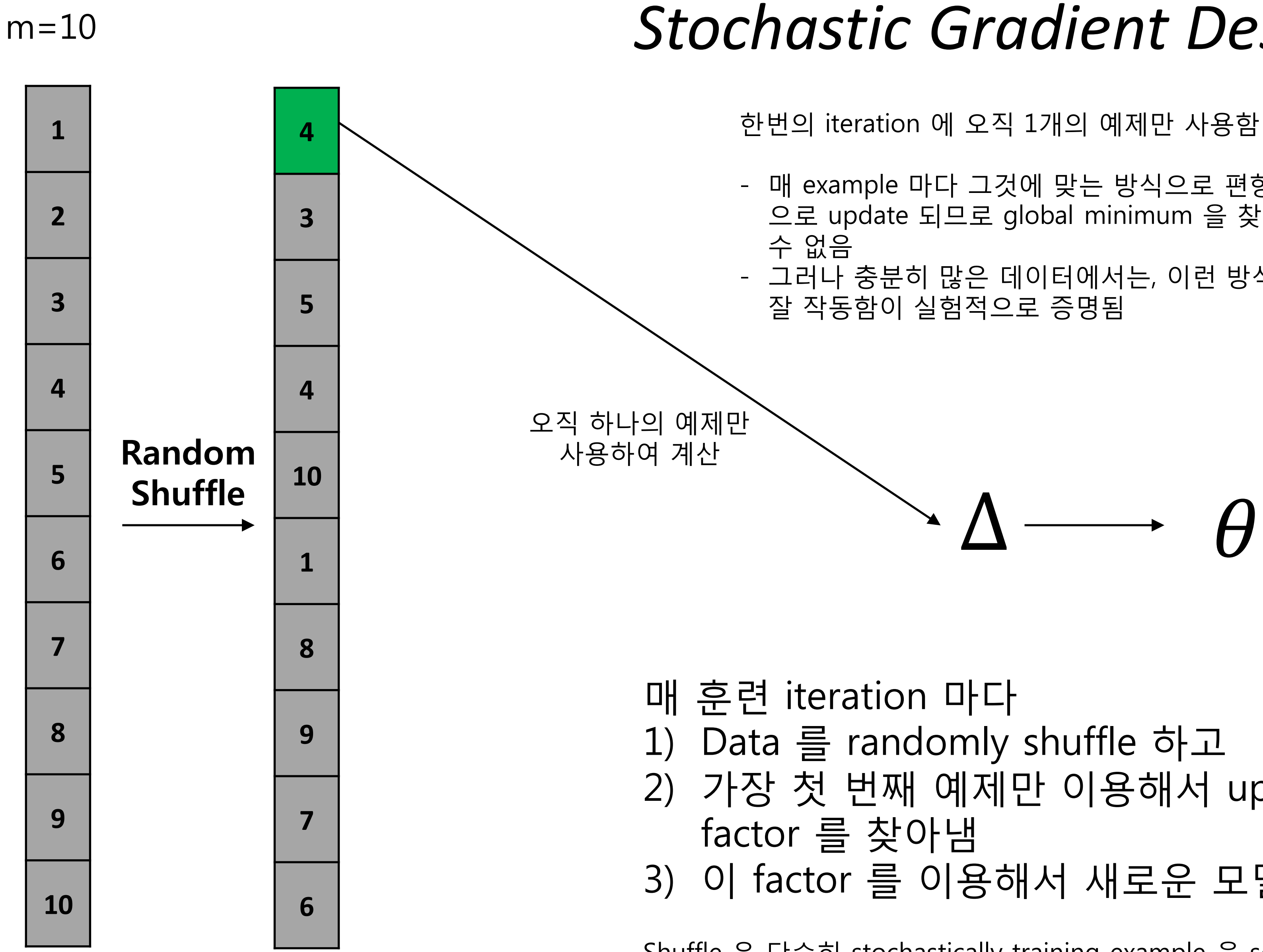
Batch Gradient Descent

일반적인 방식이나, m 이 커지면 문제가 생김

전체 데이터 m 개를 모두 scan 하고, 계산해서 새로운 모델 하나를 찾아 냄

전체 데이터를 모두 살펴서 일괄적으로 계산하므로 'batch' 라고 불림

Training – Stochastic Gradient Descent 방식



Stochastic Gradient Descent

한번의 iteration 에 오직 1개의 예제만 사용함

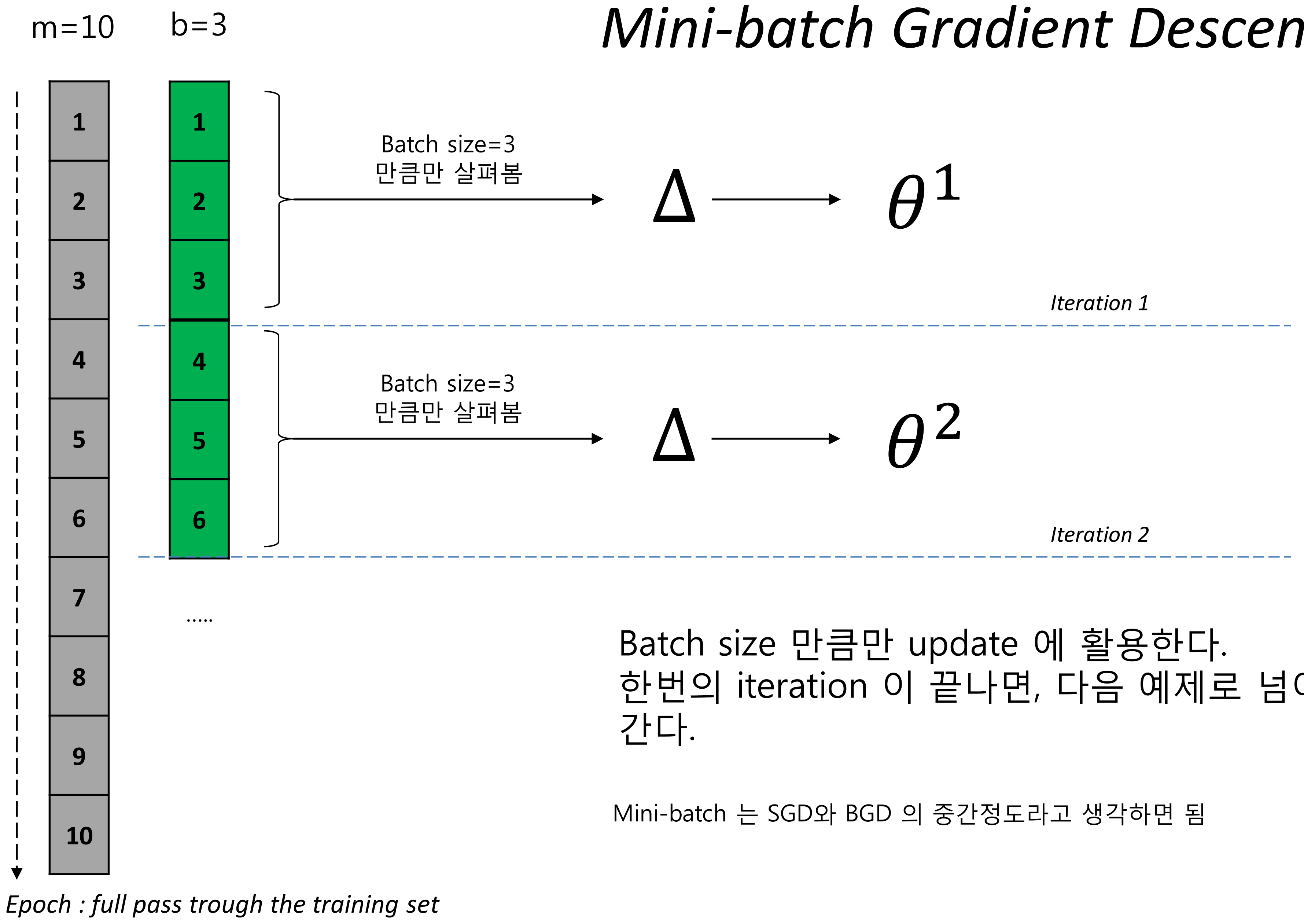
- 매 example 마다 그것에 맞는 방식으로 편향적으로 update 되므로 global minimum 을 찾을 수 없음
- 그러나 충분히 많은 데이터에서는, 이런 방식이 잘 작동함이 실험적으로 증명됨

매 훈련 iteration 마다
1) Data 를 randomly shuffle 하고
2) 가장 첫 번째 예제만 이용해서 update factor 를 찾아냄
3) 이 factor 를 이용해서 새로운 모델을 만듦

Shuffle 은 단순히 stochastically training example 을 selection 한다는 의미 정도로 생각하면 됨

Training – Mini-batch Gradient Descent 방식

Mini-batch Gradient Descent



Batch size 만큼만 update 에 활용한다.
한번의 iteration 이 끝나면, 다음 예제로 넘어간다.

Mini-batch 는 SGD와 BGD 의 중간정도라고 생각하면 됨

Epoch : full pass trough the training set

Training – Batch Gradient / Stochastic gradient / Mini-batch gradient

Batch Gradient Descent : use all ***m*** examples in each iteration

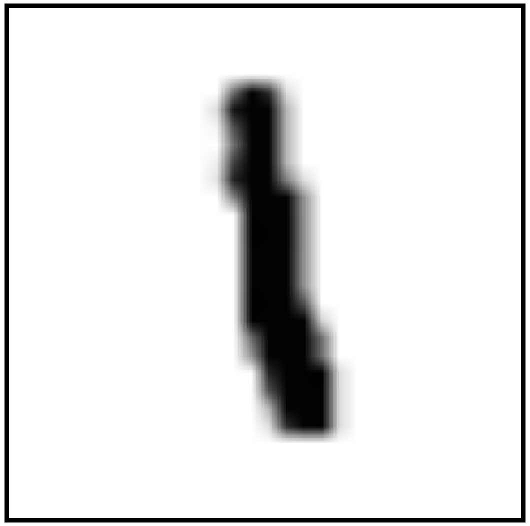
Stochastic Gradient Descent : use ***1*** example in each iteration

Mini-batch Gradient Descent : use ***b*** examples in each iteration

https://class.coursera.org/ml-003/lecture/view?lecture_id=106

Hello World – with MNIST data

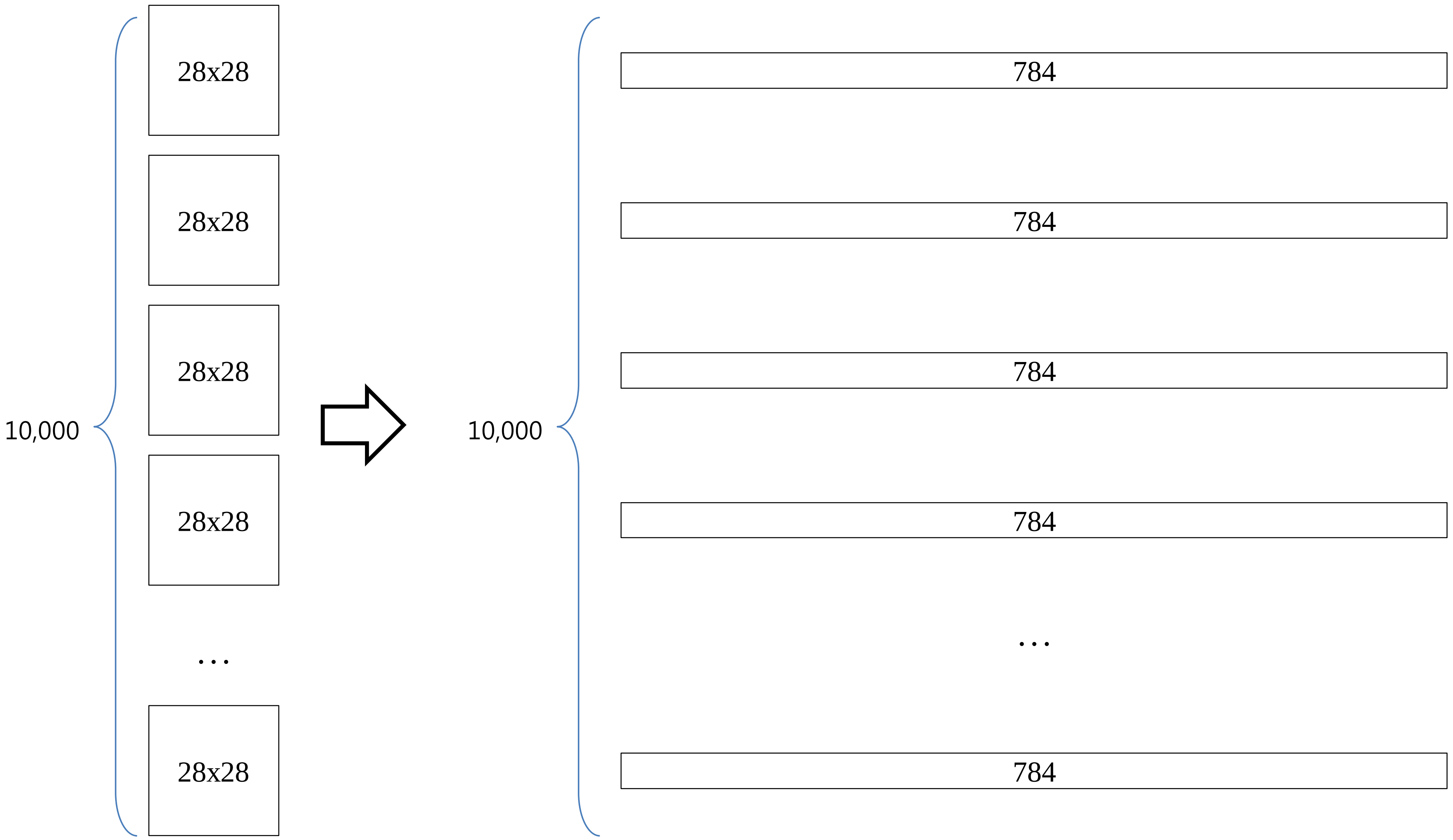
- *28 x 28 pixel for a single example*
- *Normalized value 0.0 ~ 1.0*



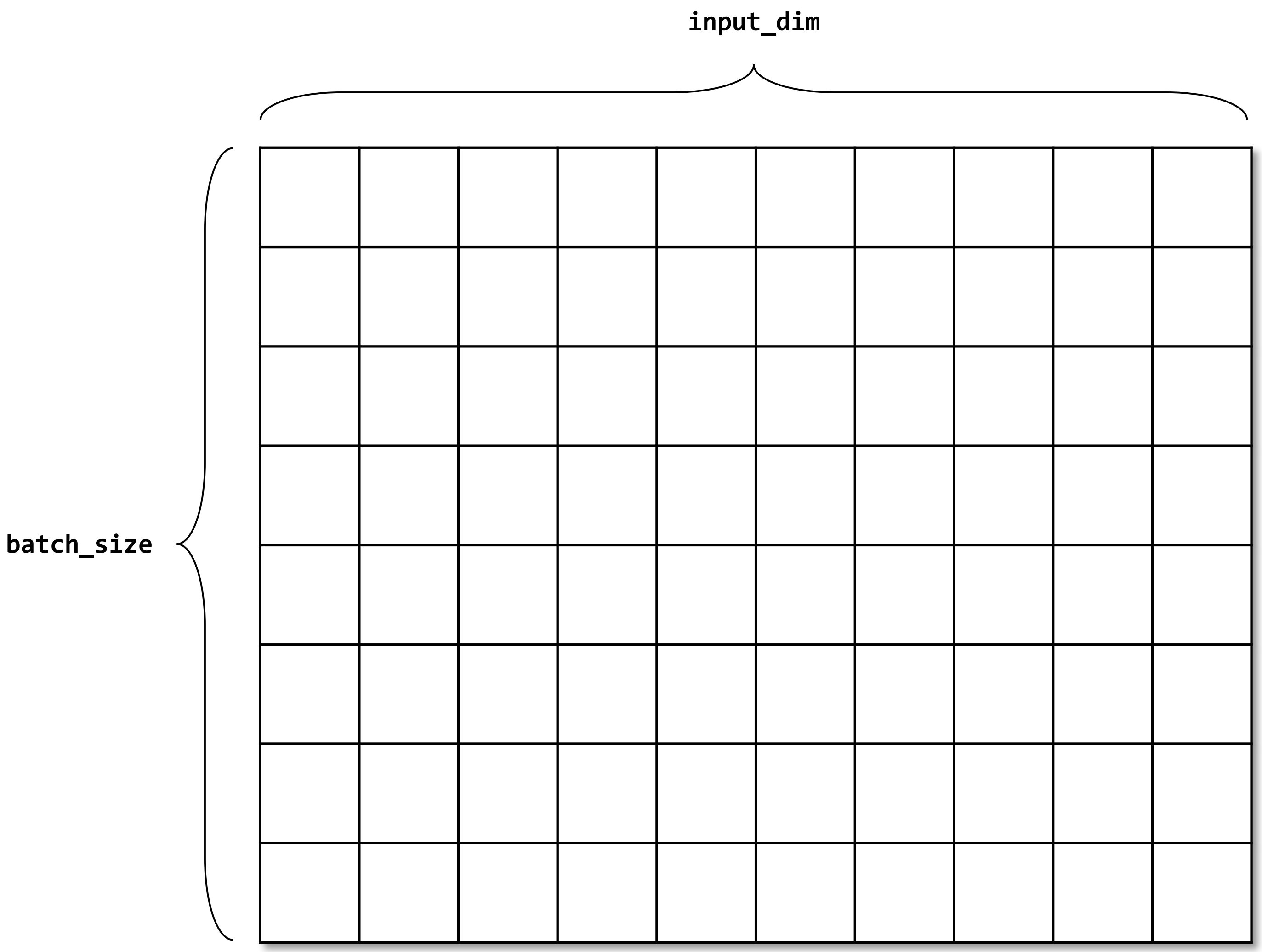
$$\simeq \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

이번 Tutorial을 위해서
10000개의 Data 만 training example 로 사용

Data 가공

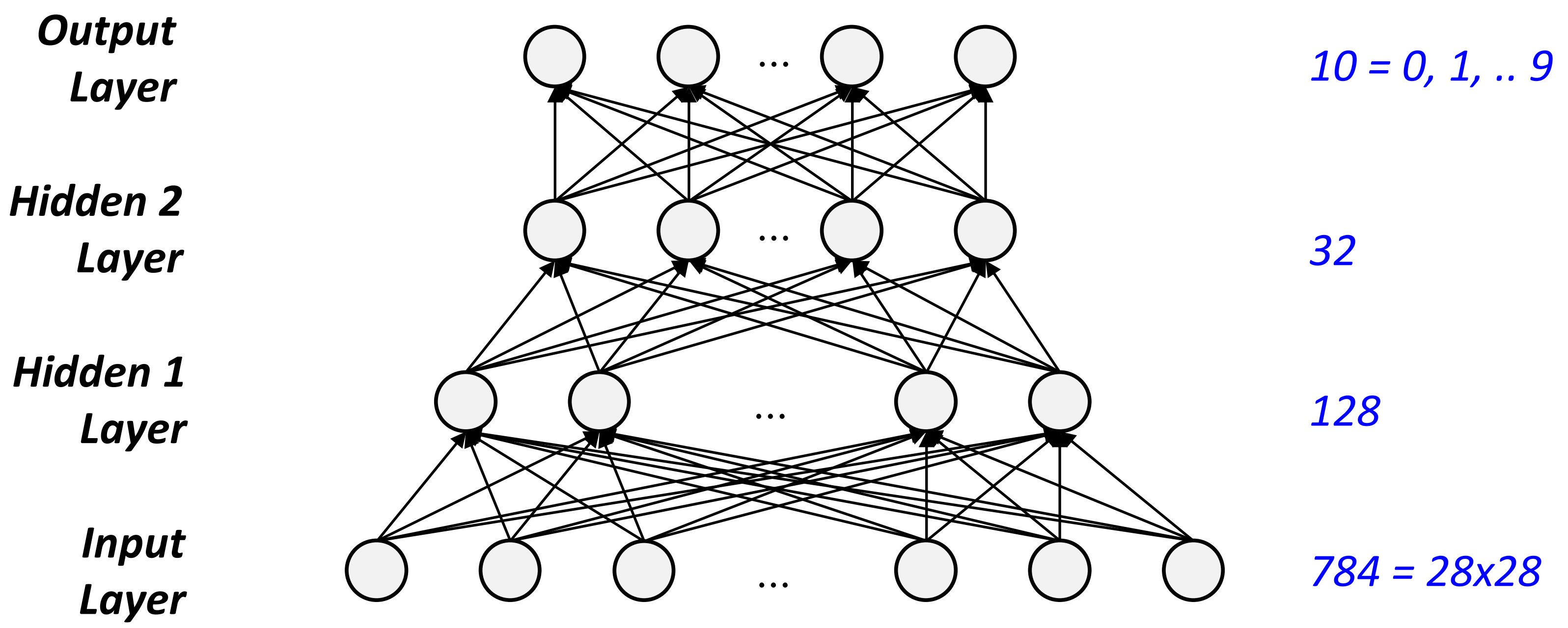

10,000
28x28
28x28
28x28
28x28
…
28x28
10,000
784
784
784
784
…
784

기본 Unit



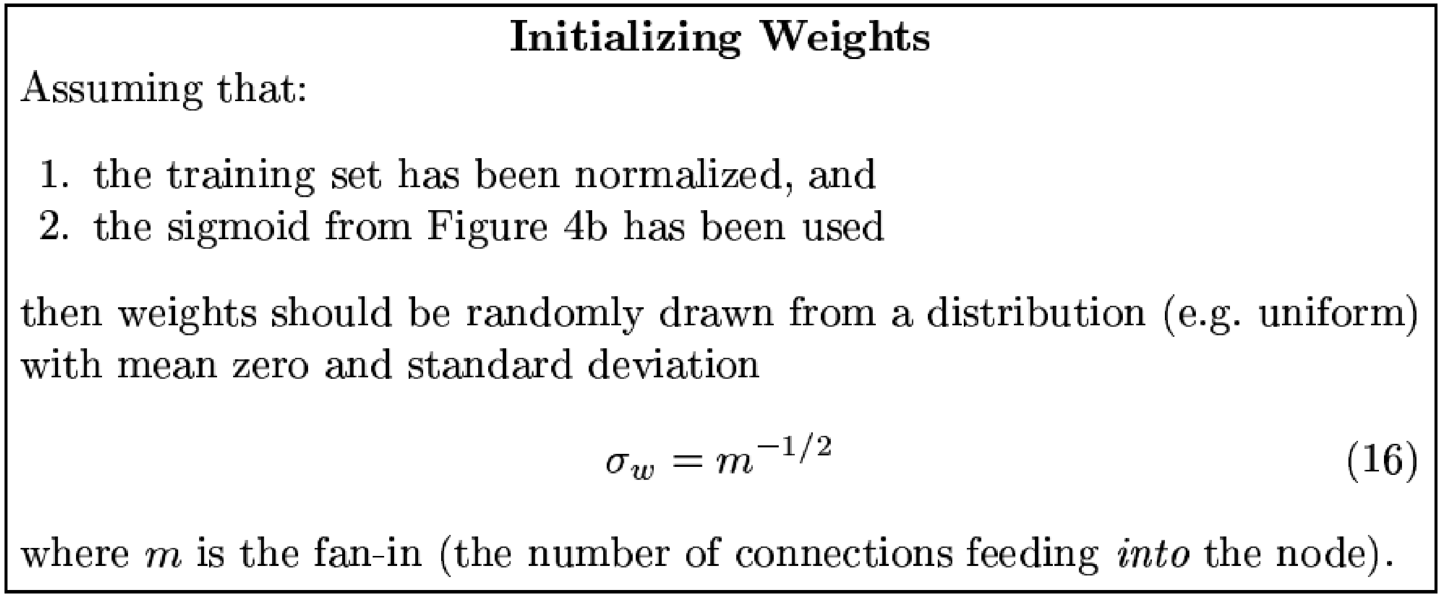
Model 구현

- 기본적인 Fully Connected Network 구현



Model 구현 | Initialization

- Just used Lecun(98) Initialization



Initializing Weights

Assuming that:

1. the training set has been normalized, and
2. the sigmoid from Figure 4b has been used

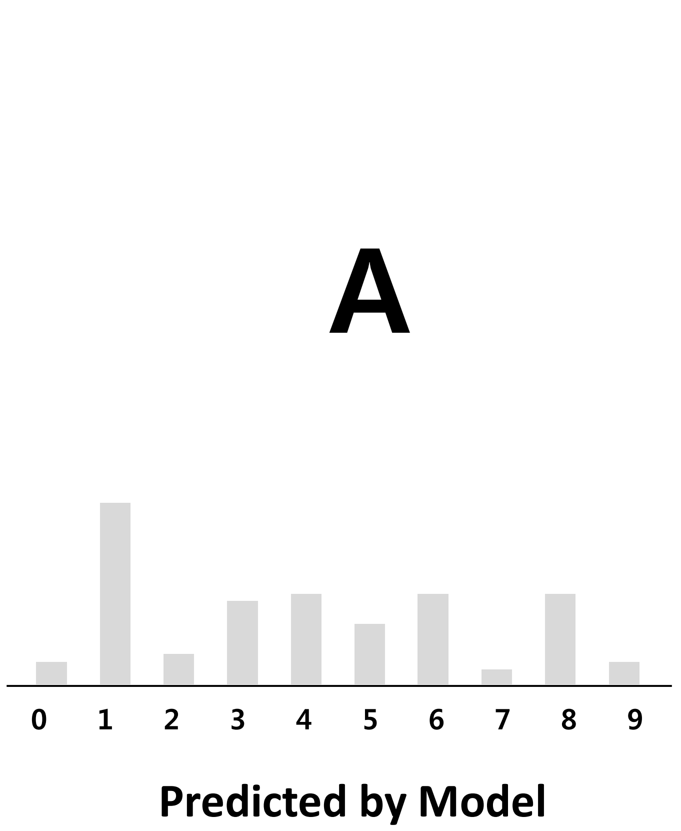
then weights should be randomly drawn from a distribution (e.g. uniform) with mean zero and standard deviation

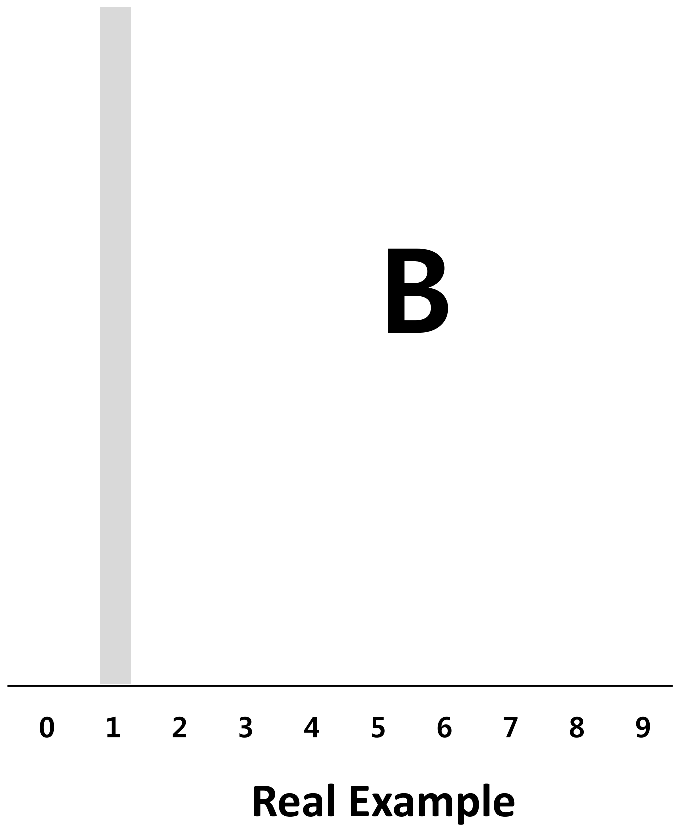
$$\sigma_w = m^{-1/2} \tag{16}$$

where m is the fan-in (the number of connections feeding *into* the node).

http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

Loss 구현





= A와 B의 차이는 얼마나 될까? (divergence)
= A를 B로 바꾸는데 드는 비용은? (cost)
= B대비 A의 손실은 얼마일까? (loss)

Sequence learning

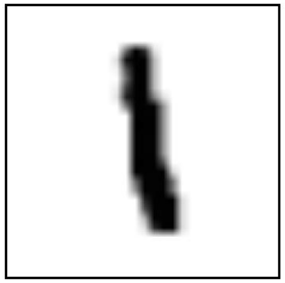
FROM FULLY CONNECTED NETWORK

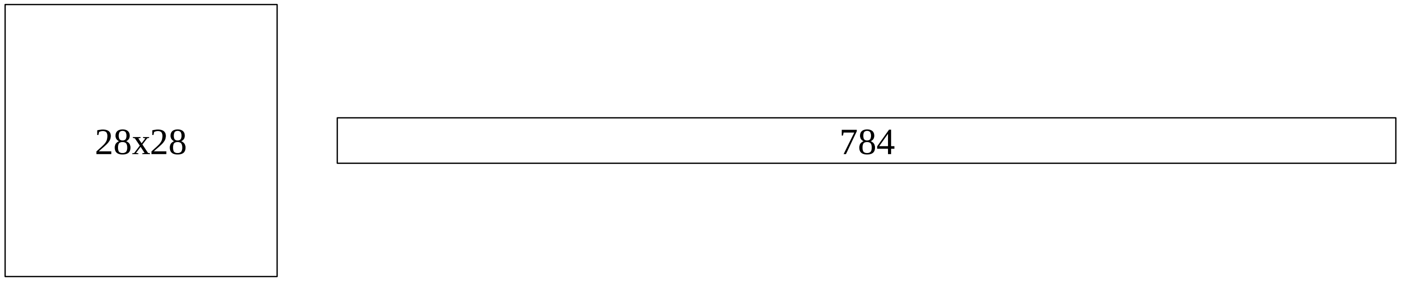
`check pos_tagger_fcn.py`

Ex) Data

- 문서의 내용이 다음과 같다고 할 때
  - Hello world Hi tutorial
  - I   love   you !
  - …

Handle word as image

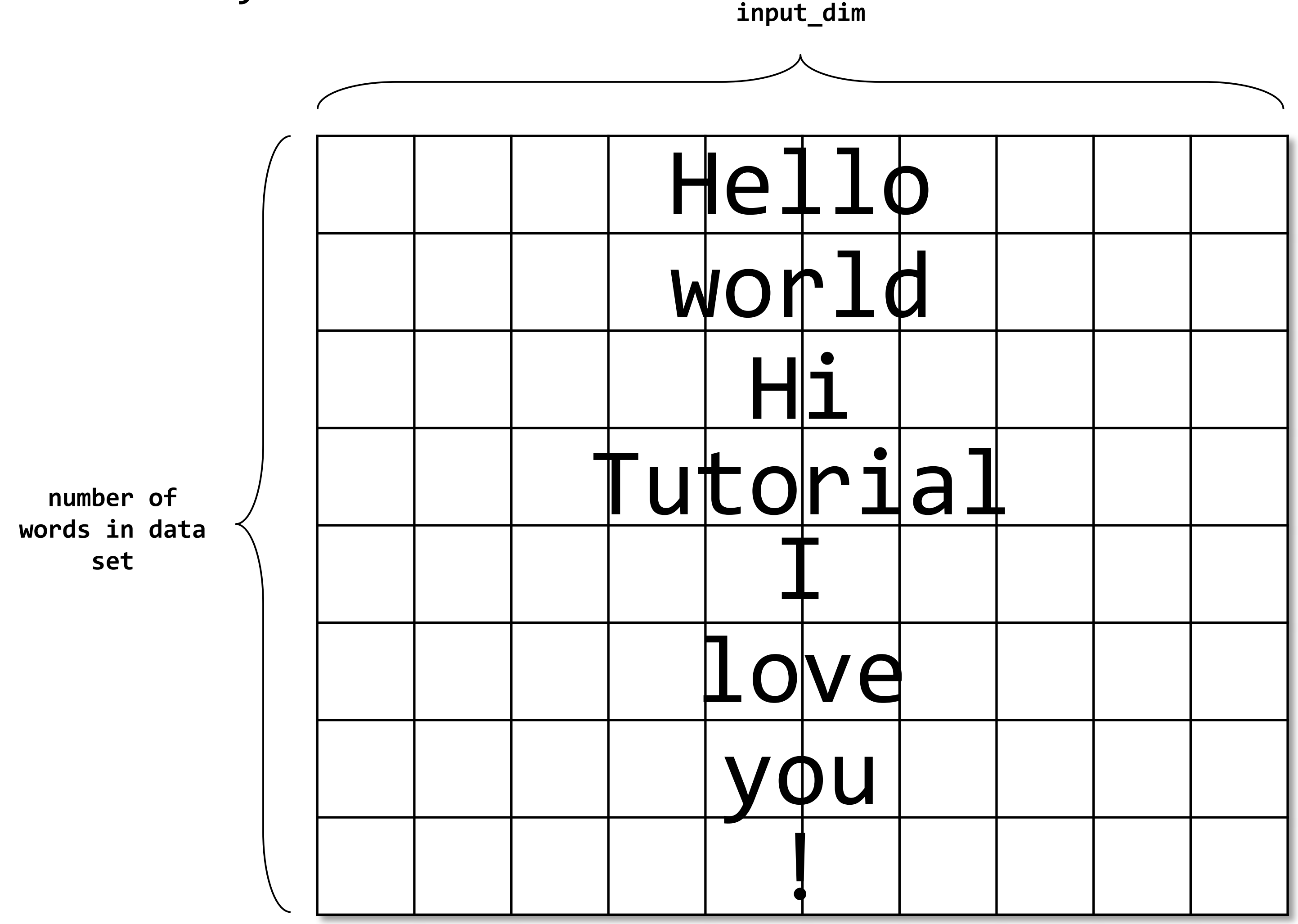




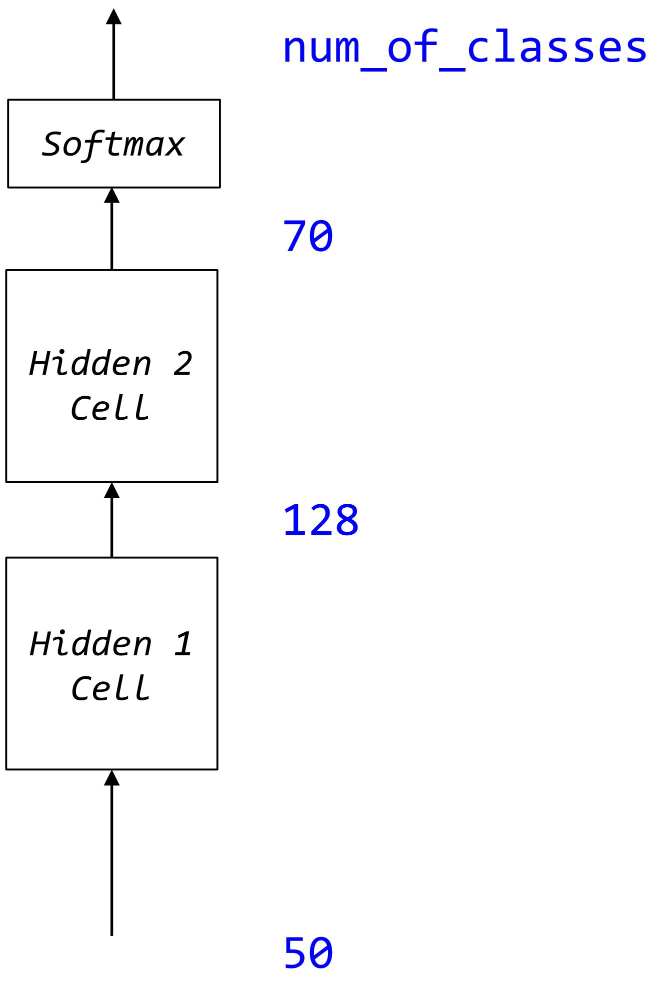
word embedding dimension

Ignore sentence boundary

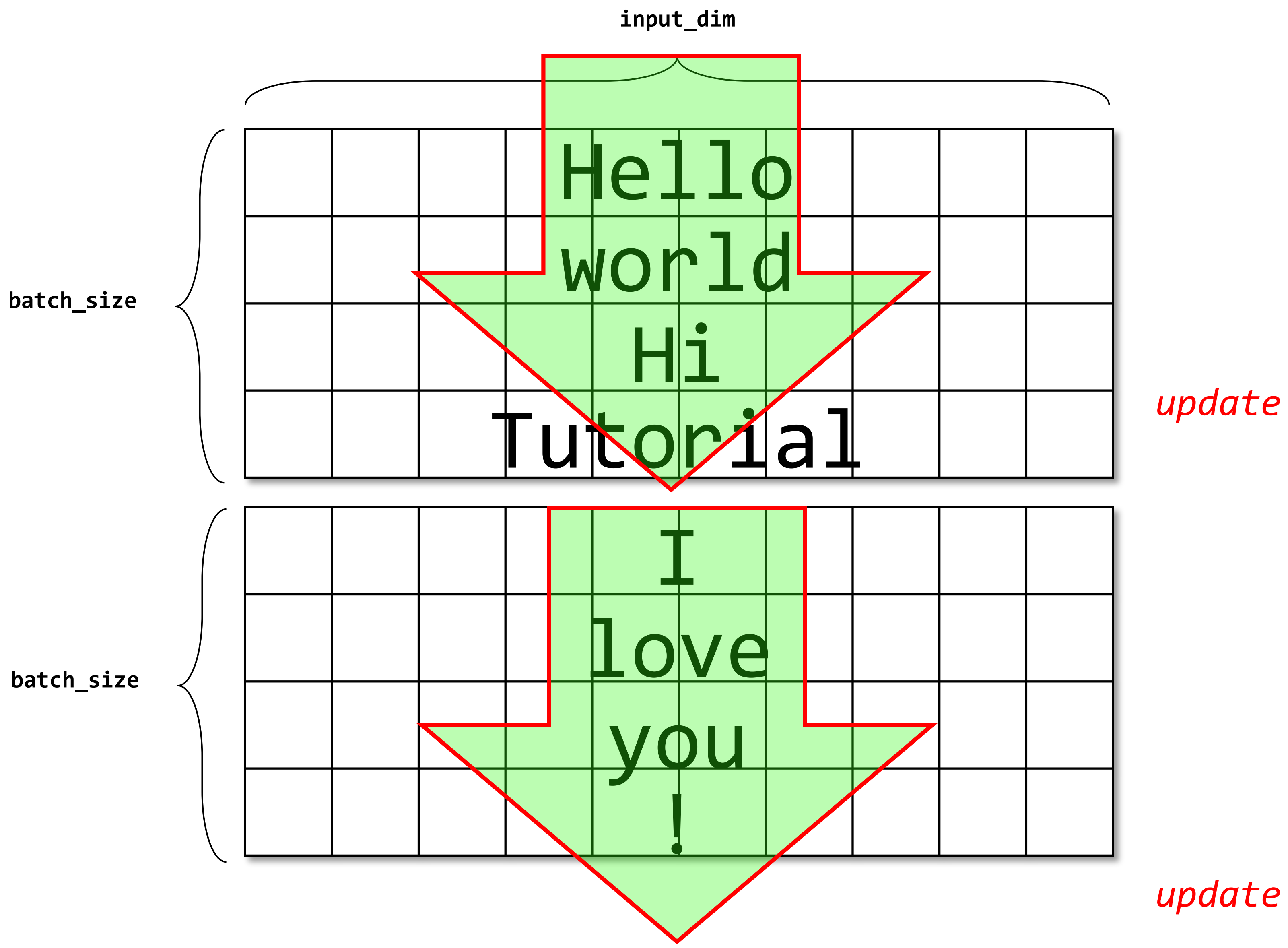
```
"Hello World Hi Tutorial"
"I Love you !"
```



Model Description



Update Flow



Batch data 만큼을 살피고, 한번 update 하는 것을 의미

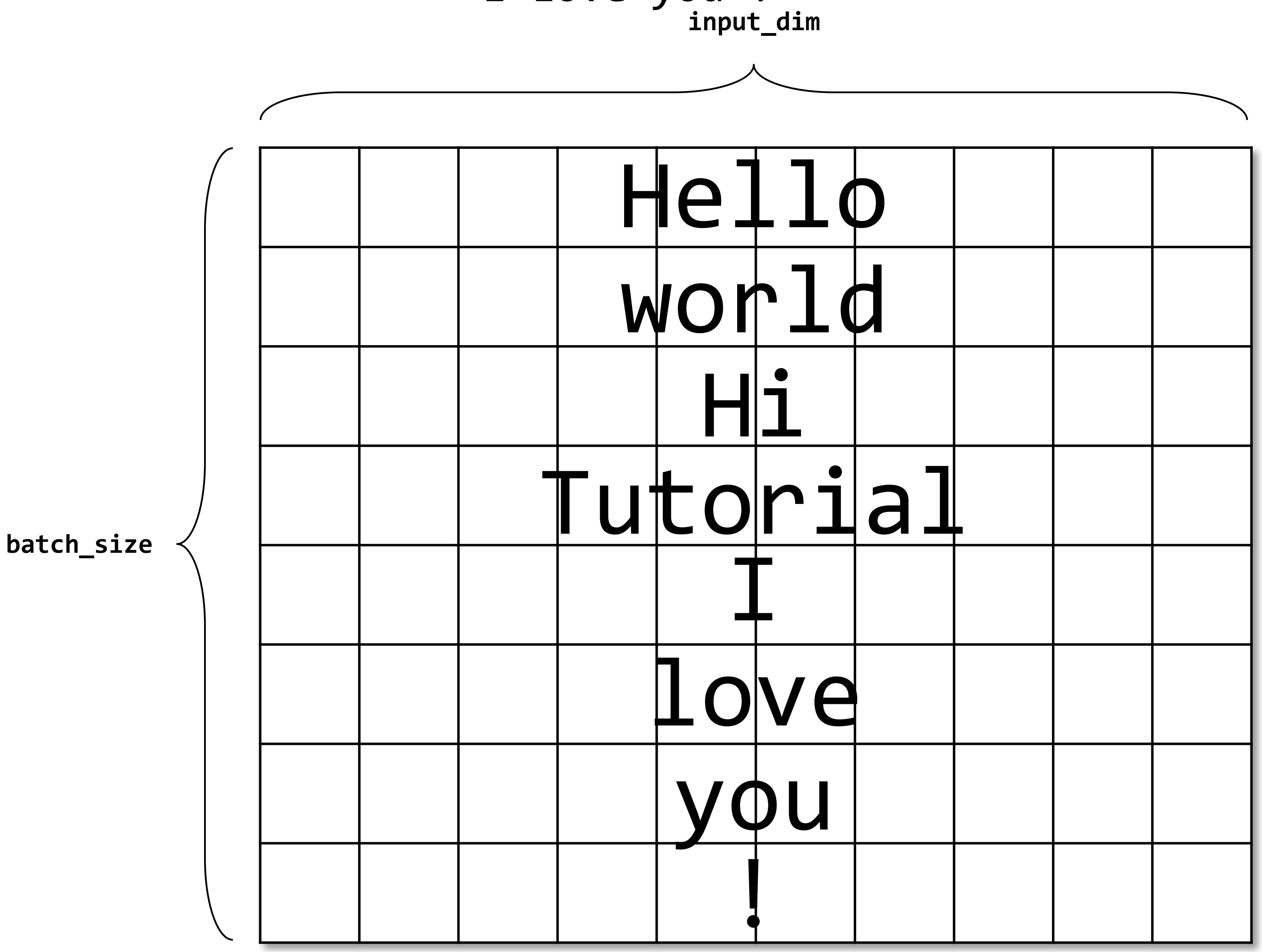
Sequence learning

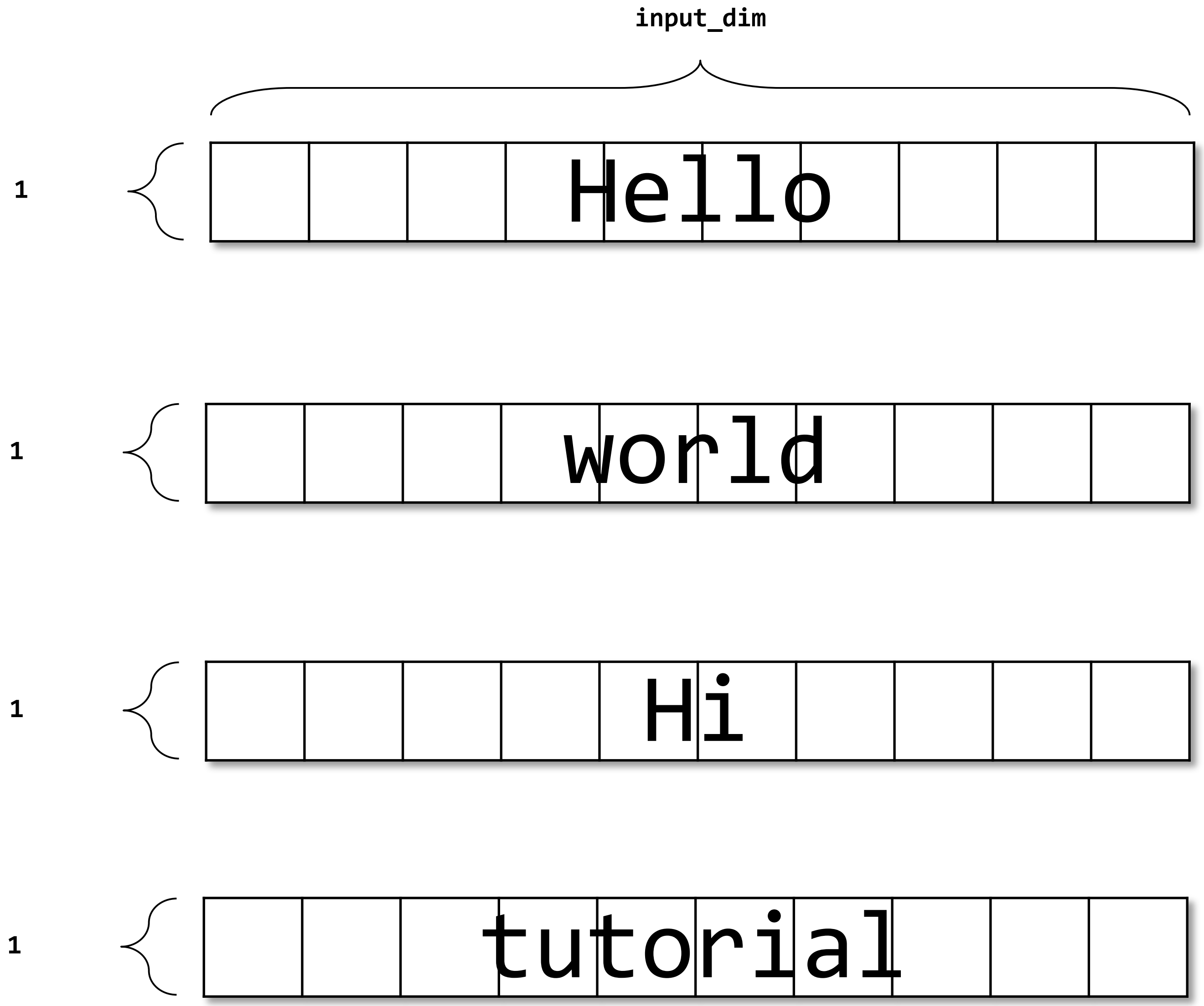
FROM FULLY CONNECTED NETWORK – SEQUENCE WISE

check pos_tagger_fcn_seq.py

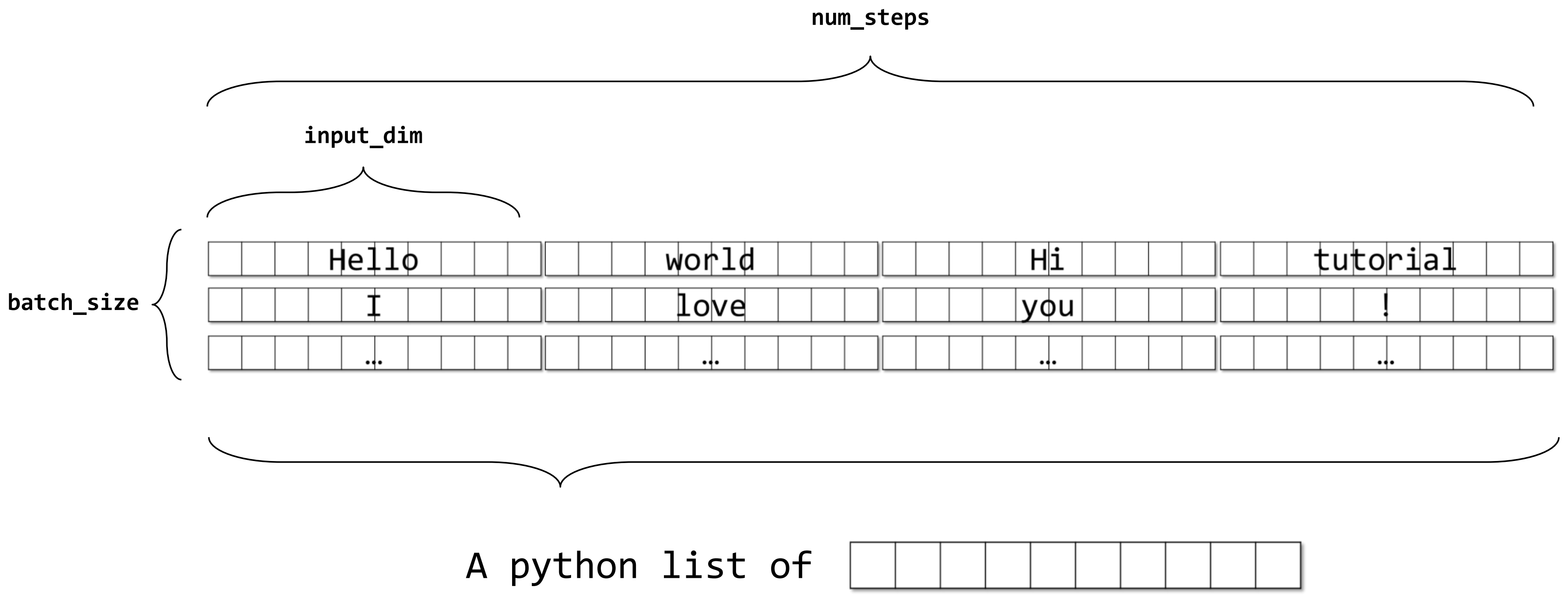
[Remind] 기본 Unit

“Hello World Hi Tutorial”
“I Love you !”

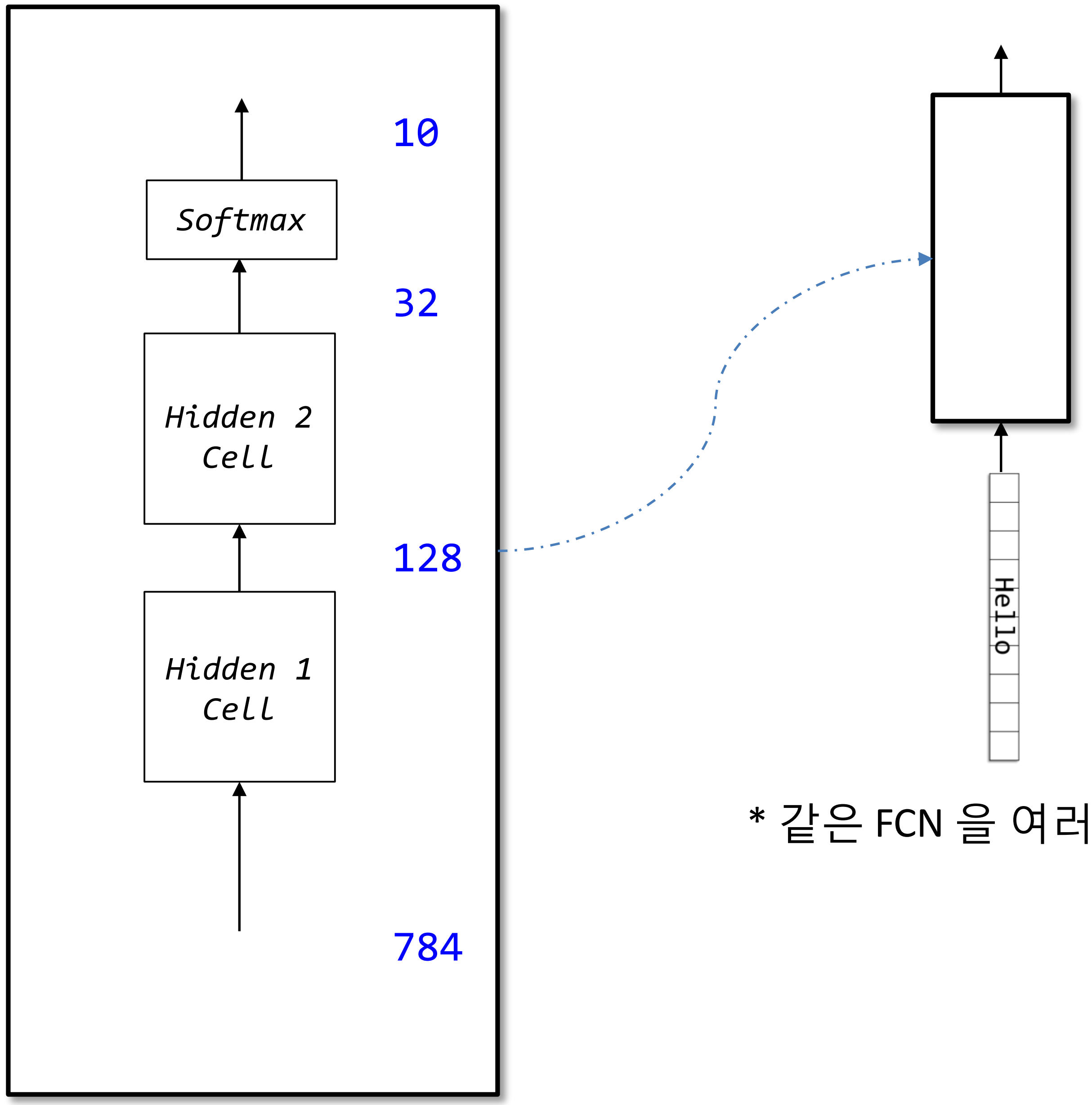



input_dim
1
Hello
1
world
1
Hi
1
tutorial

Sequence Learning 에서의 기본 unit

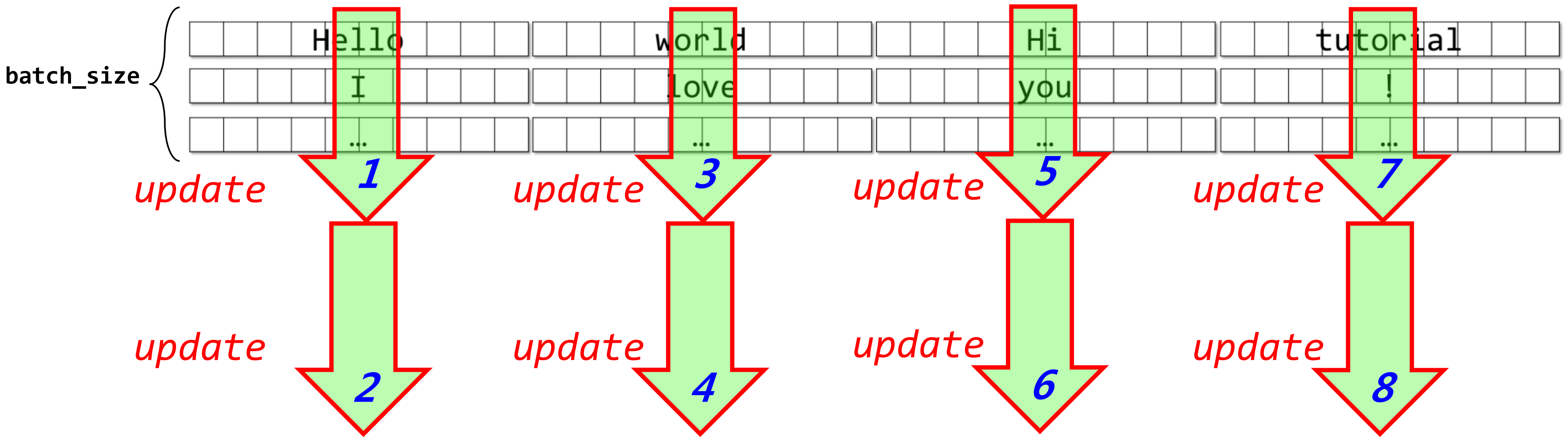


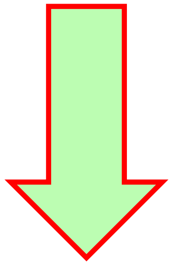
Model Description



* 같은 FCN 을 여러 번 반복

Learning flow





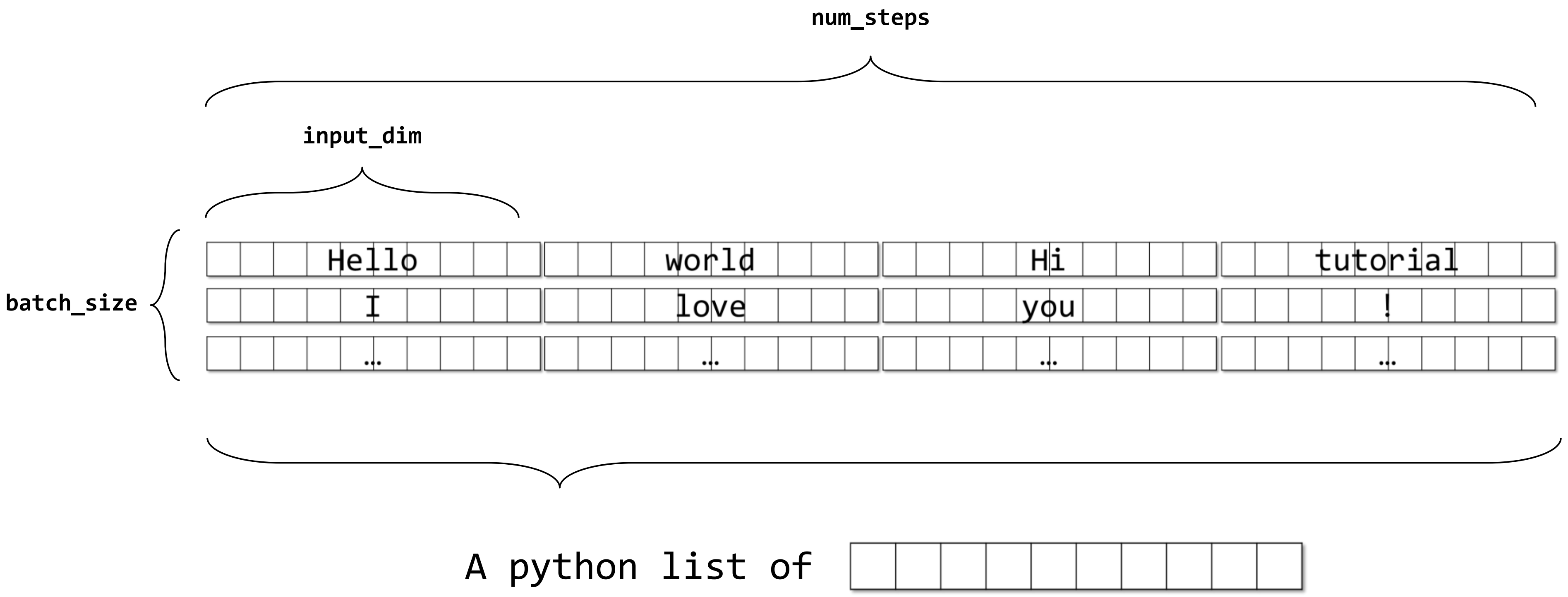
Batch data 만큼을 살피고, 한번 update 하는 것을 의미

Sequence learning

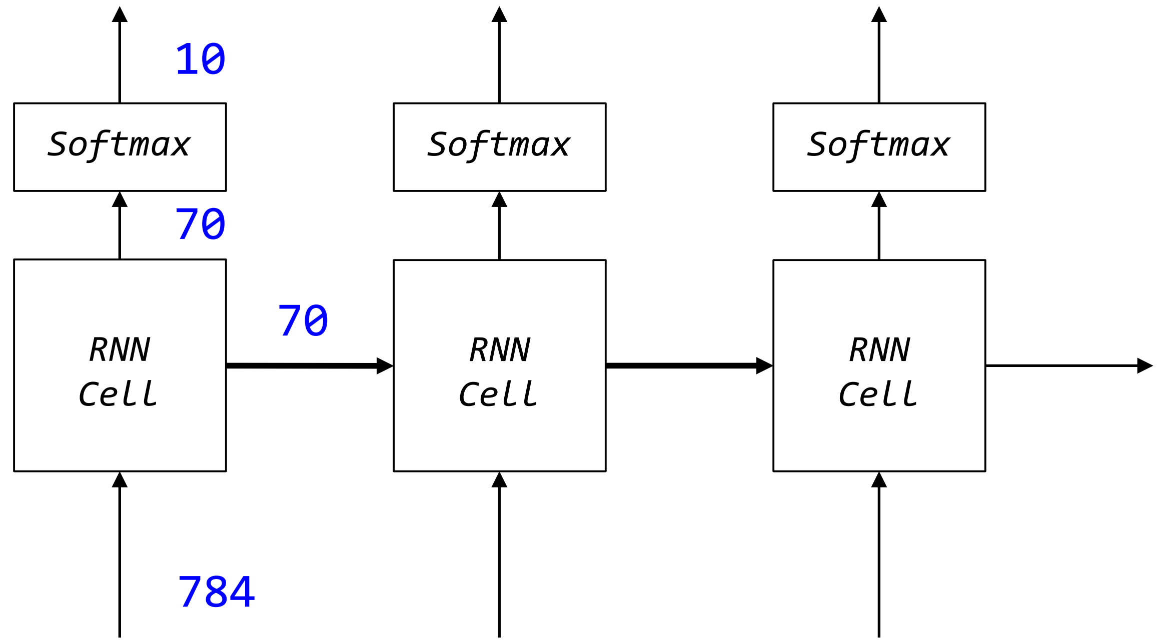
RECURRENT NEURAL NETWORK

check pos_tagger_rnn_seq.py

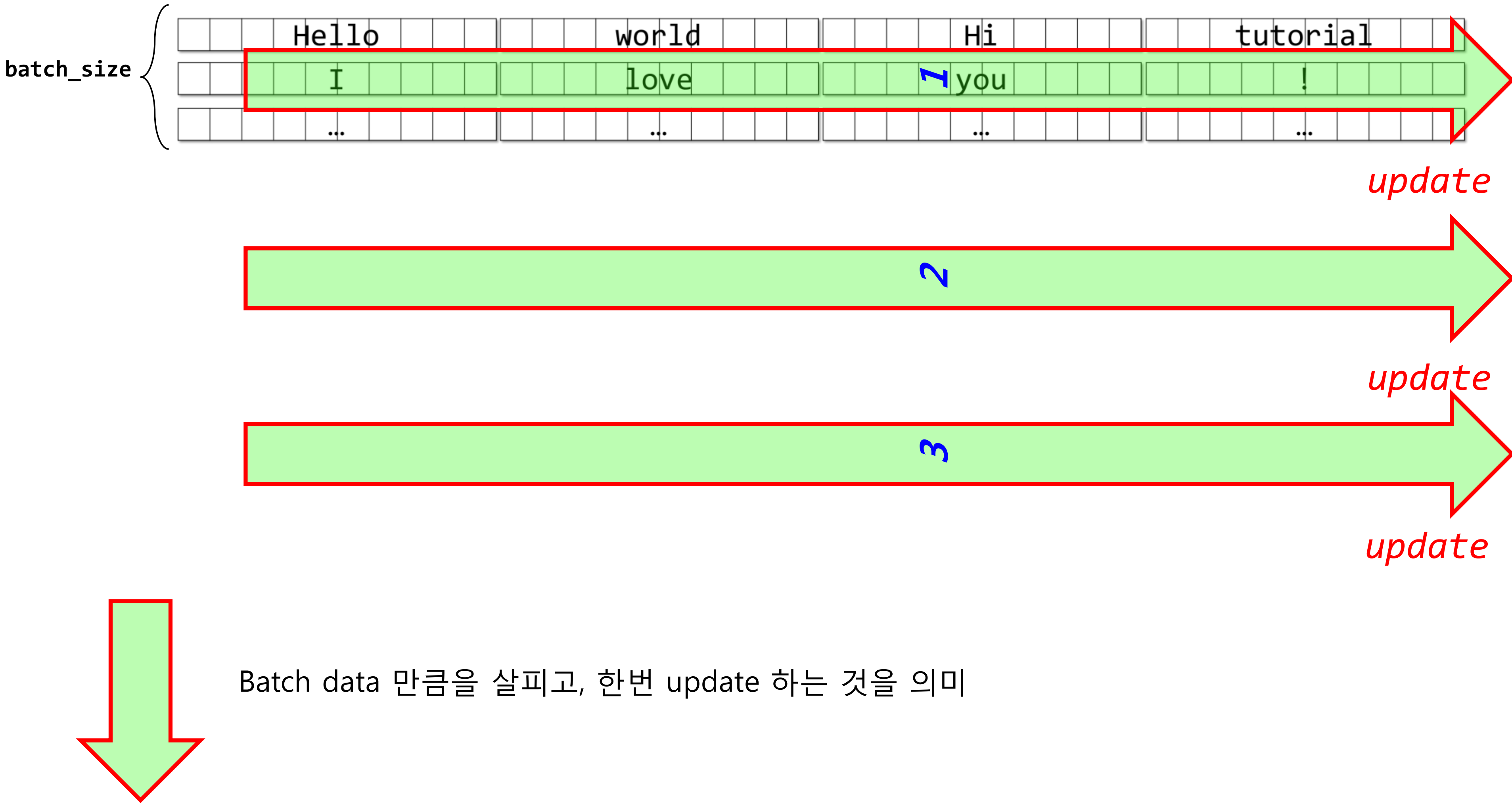
[Remind] Sequence Learning 에서의 기본 unit



Model Description

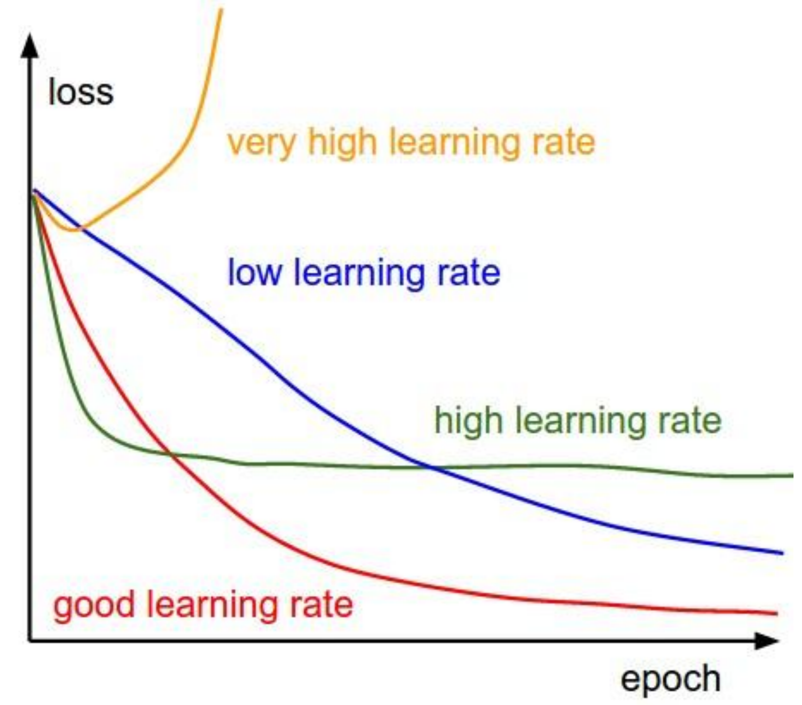


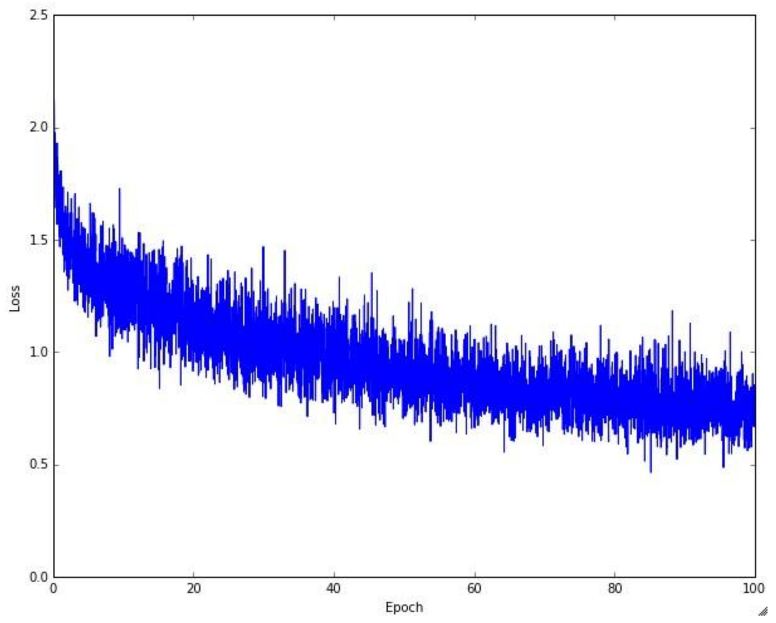
Learning flow

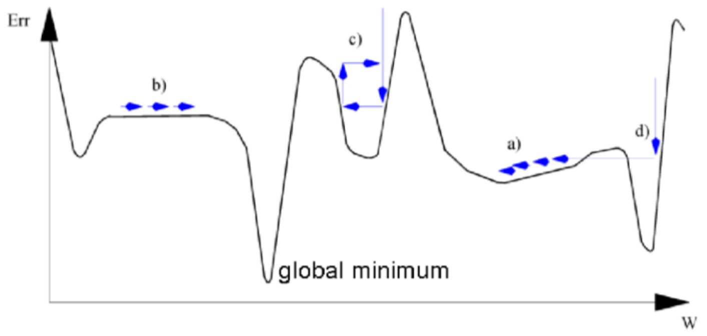


Batch data 만큼을 살피고, 한번 update 하는 것을 의미

[참고] Tuning Learning Rate

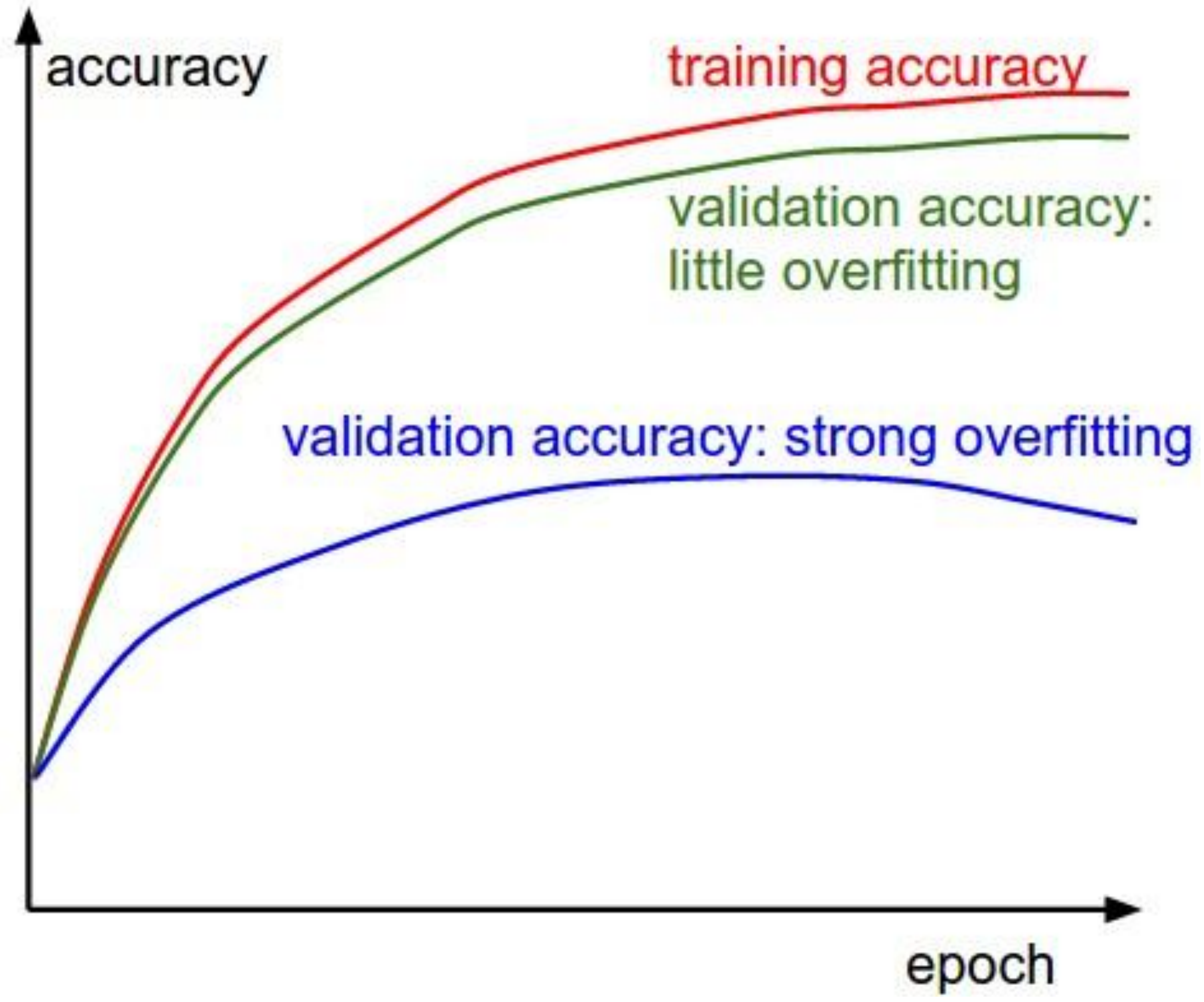






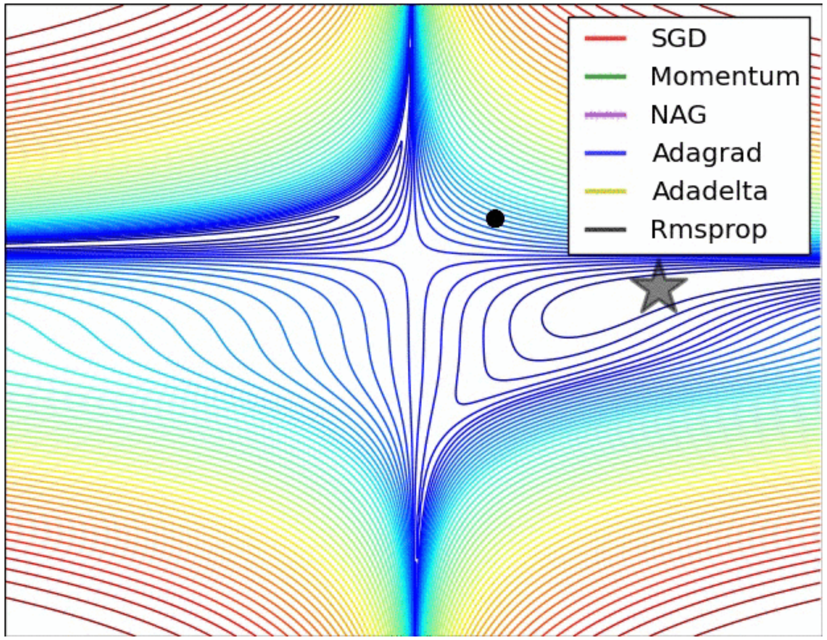
http://cs231n.github.io/neural-networks-3/

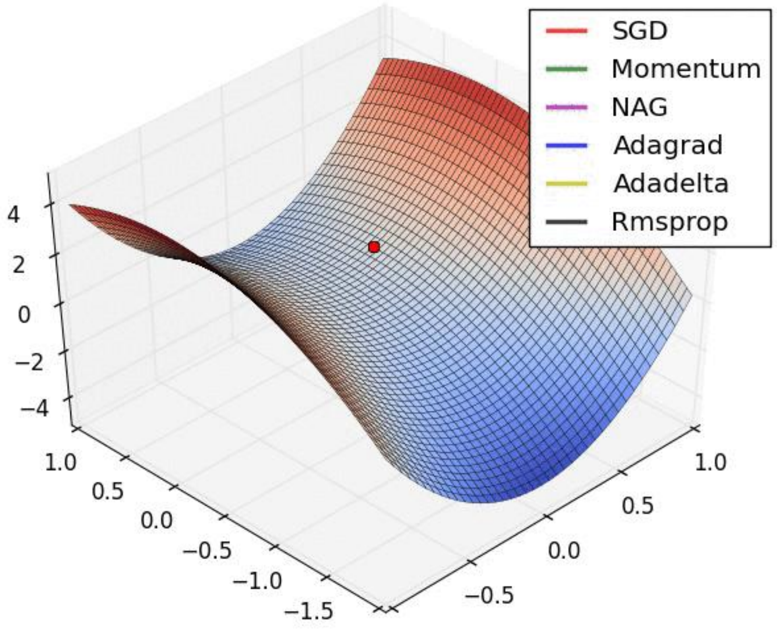
[참고] Train / Validation Accuracy



http://cs231n.github.io/neural-networks-3/

[참고] 각 Optimizer 비교





Left: Contours of a loss surface and time evolution of different optimization algorithms.

Right: A visualization of a saddle point in the optimization landscape, where the curvature along different dimension has different signs (one dimension curves up and another down).

Image credit : Alec Radford

http://cs231n.github.io/neural-networks-3/

Q/A

감사합니다.

정상근, Ph.D

Intelligence Architect

Senior Researcher, AI Tech. Lab. SKT Future R&D
Contact : hugmanskj@gmail.com, hugman@sk.com