Tensorflow 101

Sangkeun Jung
(hugmanskj@gmail.com)

2016-10-27

Goal

- ✓ TensorFlow 소개
- ✓ Machine Learning Basic Block
- ✓ Part I : Tensorflow 101 (python notebooks)
- ✓ Part II : Live coding (MNIST)
- ✓ Part III : Tensorflow 201 (Optional)

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***

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.

Ian Goodfellow : main contributor of 'Theano'

Yangqing Jia : main contributor of 'Caffe'

(Official) Tensor flow Intro.

Why Did Google Open Source This?

> …
> Research in this area is global and growing fast, but **lacks standard tools**. By sharing what we believe to be one of the best machine learning toolboxes in the world, we hope to create **an open standard** for exchanging research ideas and putting machine learning in products.
> …

Features

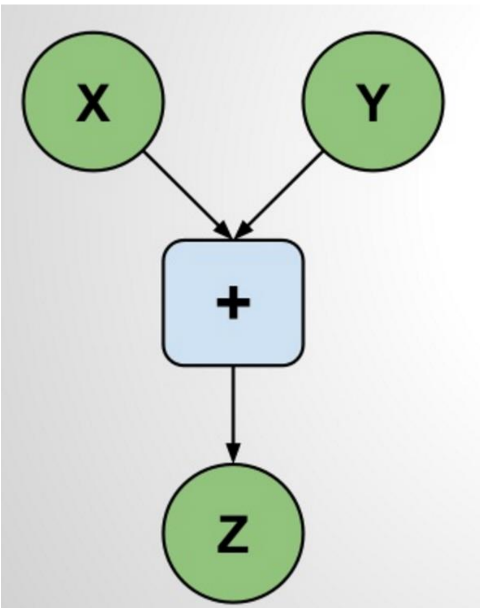
Deep Flexibility	
True Portability	… CPUs or GPUs on desktop, server, and mobile computing platforms (android)
Connect Research and Production	… Research Product = Commercial product
Auto-Differentiation	… symbolic math expressions.
Language Options	… Officially Python / C++ support. GO, Java, Lua, Javascript, R interface is developing by open community
Maximize Performance	… all CPUs, GPUs can be used

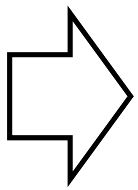
Features

Programming Model	• Dataflow-like model
Language	• Python • C++
Deployment	• Code once, Run everywhere
Computing Resource	• CPU • GPU
Distribution Process	• Local Implementation • Distributed Implementation
Math Expressions	• Math Graph Expression • Auto Differentiation
Optimization	• Auto Elimination • Kernel Optimization • Communication Optimization • Support model, data parallelism • …

Symbolic Expression?

z = x+ y





$$-\frac{\partial \log p(v)}{\partial W_{ij}} = E_v[p(h_i|v) \cdot v_j] - v_j^{(i)} \cdot sigm(W_i \cdot v^{(i)} + c_i)$$

$$-\frac{\partial \log p(v)}{\partial c_i} = E_v[p(h_i|v)] - sigm(W_i \cdot v^{(i)})$$

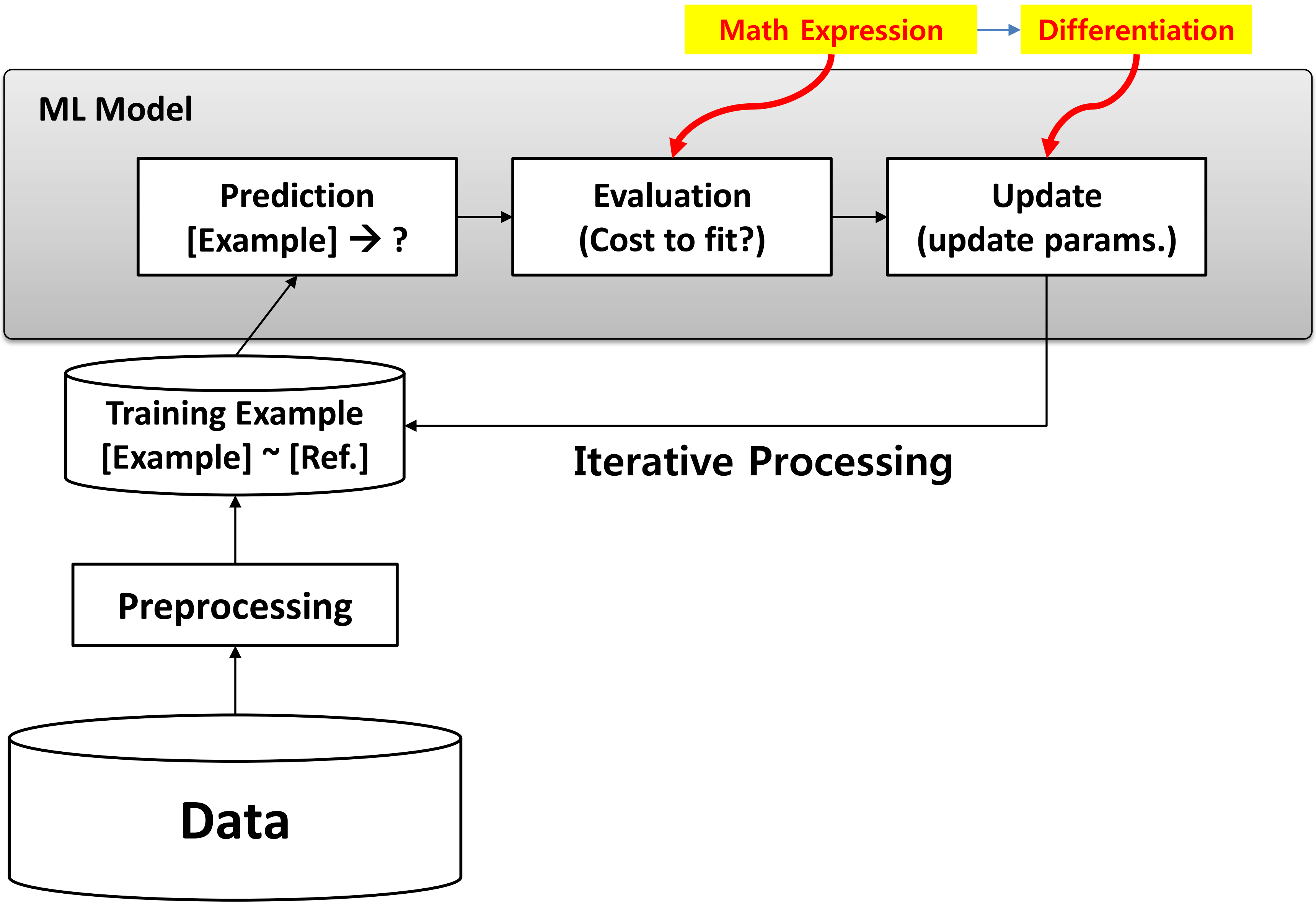
$$-\frac{\partial \log p(v)}{\partial b_j} = E_v[p(v_j|h)] - v_j^{(i)}$$

Why Graph based Symbolic expression is important?

Auto Differentiation

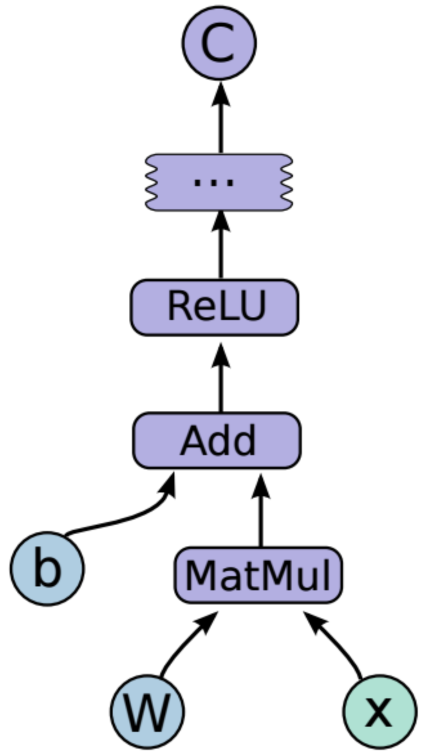
Distributed Computation

(almost) every Machine Learning Algo. do



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

Implementation modes

Local : Client, Master, Worker in a single machine

Distributed : Client, Master, Worker in different machine

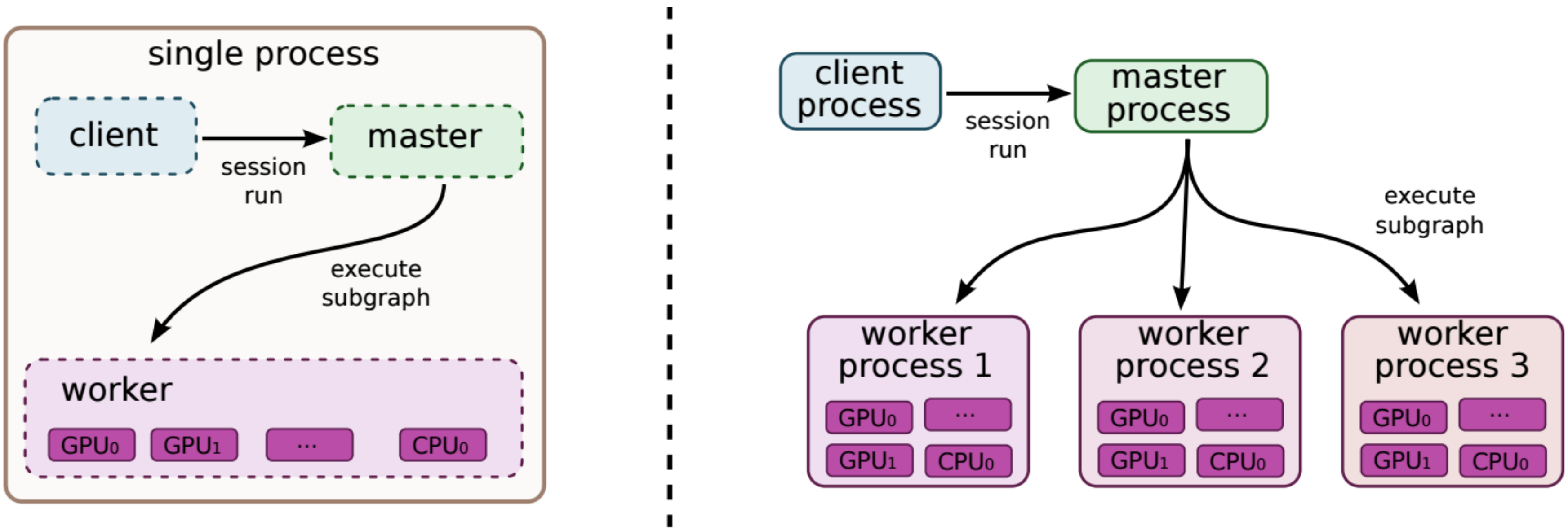


Figure 3: Single machine and distributed system structure

If we need to define tensorflow with previous tools and concept…

Tensorflow *is*

MATLAB + hadoop

MACHINE LEARNING - BASIC BLOCK

AI Problem Formulation

Classification

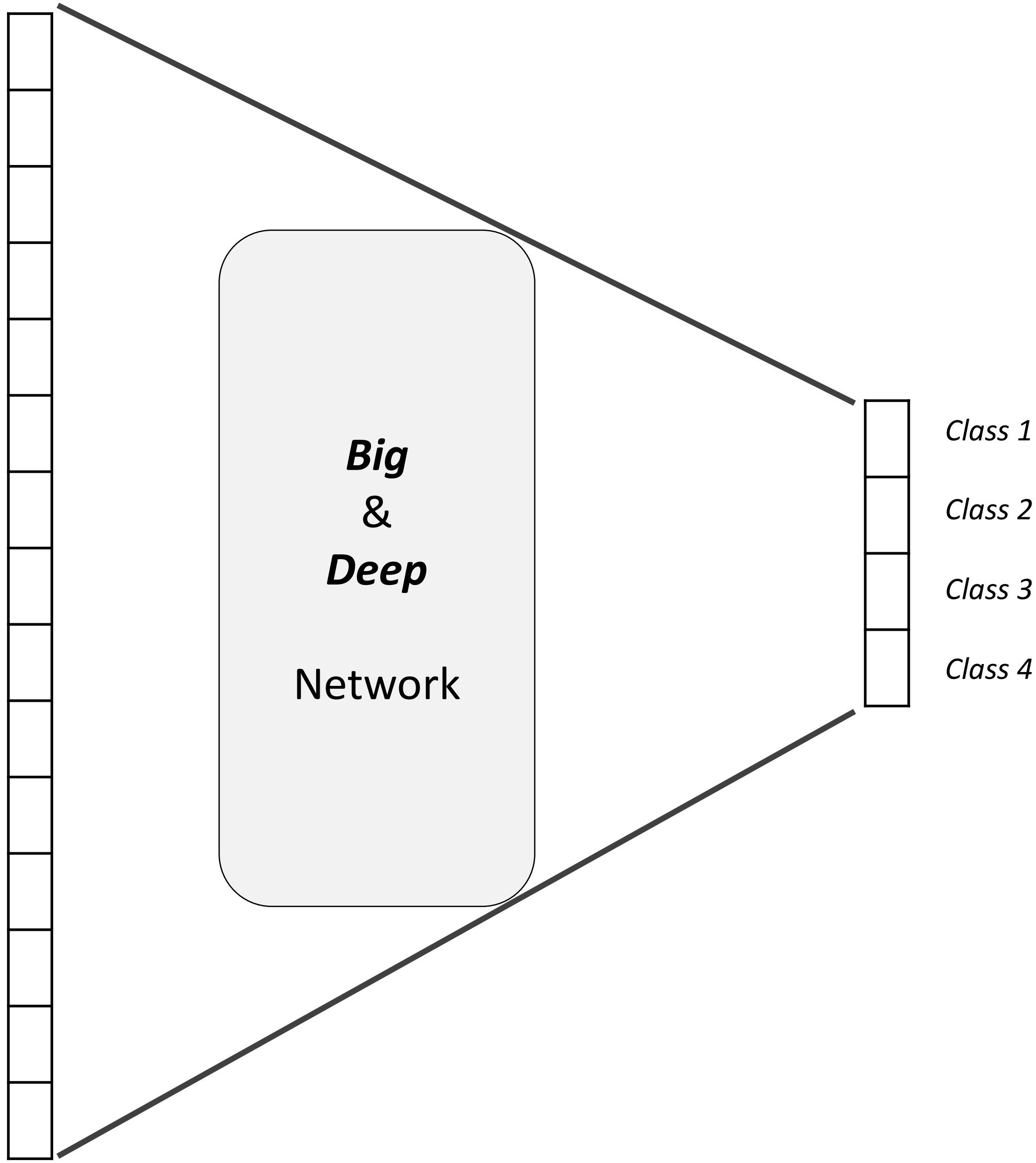
Clustering

We can view AI problem as classification or clustering problems.

Image Classification

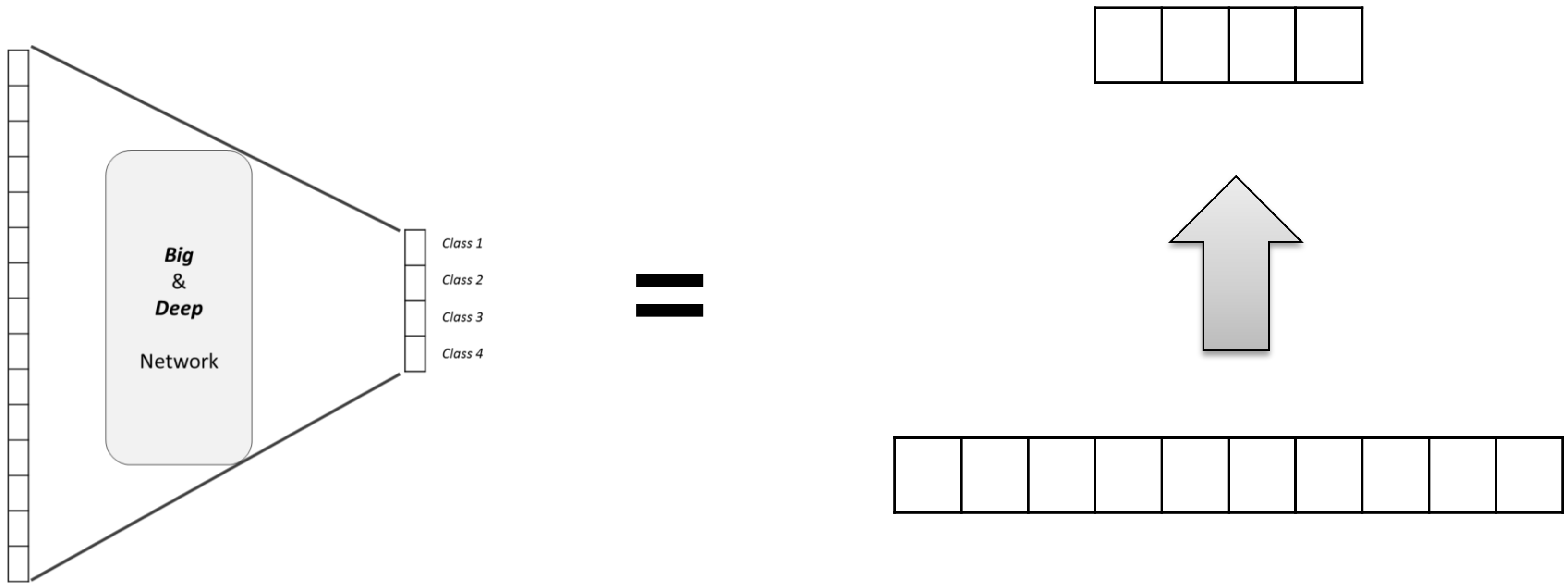
from Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.

Classification Formulation



!!! There is no "don't know" class in this formulation

Data Transformation



Abstraction
at
Input / Output
level

Ex) Image Classification

Data Transformation @ Image Classification

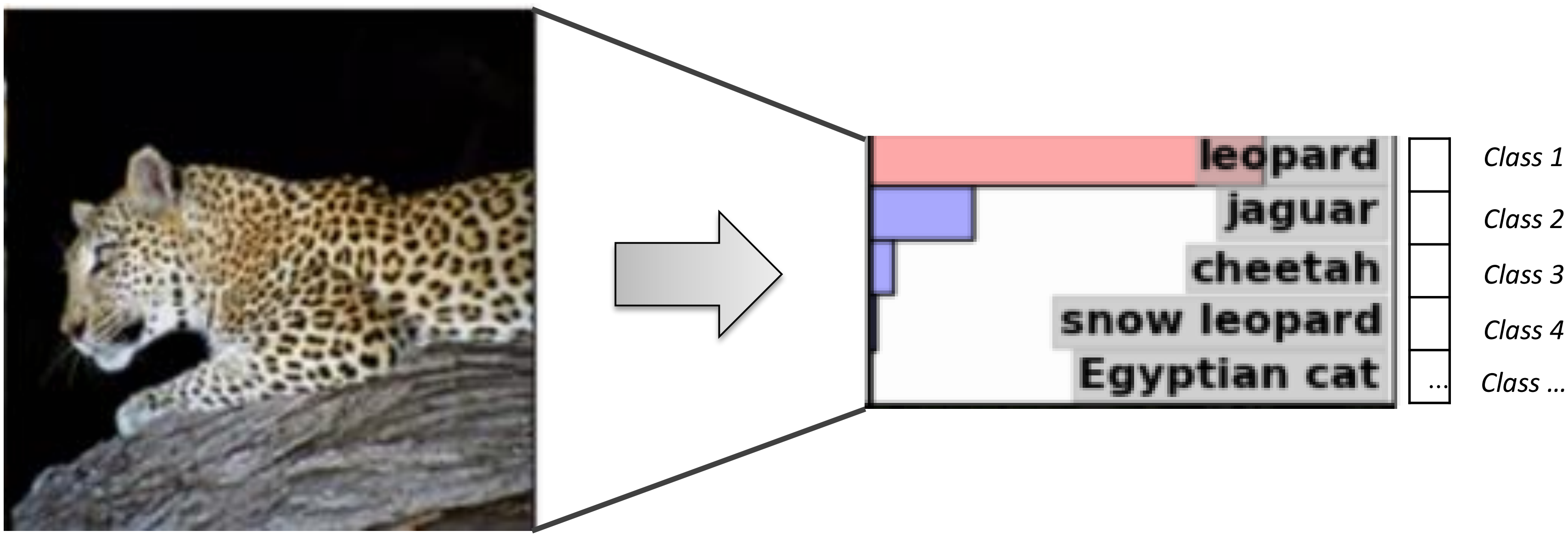


Image Pixel Data → **Class Distribution**

ML 기본 Block

Statistical Machine Learning

- 가지고 있는 데이터를 이용하여
- 풀고자 하는 문제의 통계적 모델링을 통해
- 실제 정답과의 오차를
- 파라미터 학습을 통해
- 줄이는 과정을 반복

Data Preparation

Model Implementation

Loss Implementation

Updater Implementation

Iterative Learning

Data 준비

Data Prep.

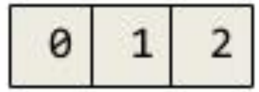
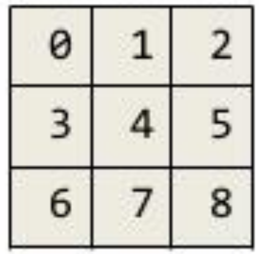
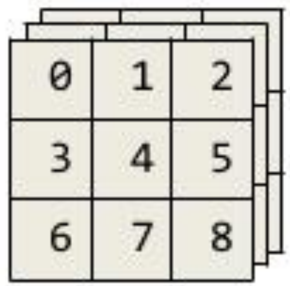
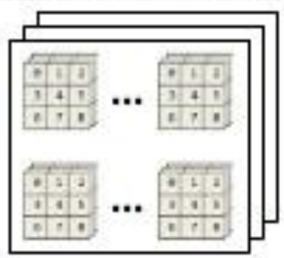
Everything is 'Tensor'

Speech, Image and even text is ***N-dimensional array (tensor)***

Important Concept

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

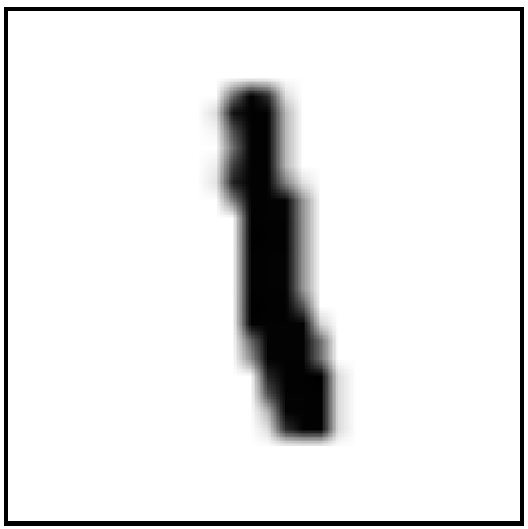
What is an array?

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

Image as a Tensor

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



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

Tensor 를 자유자재로 다룰 수 있는가가
TF 를 잘 다루는 핵심

Part I

WITH PYTHON NOTEBOOKS

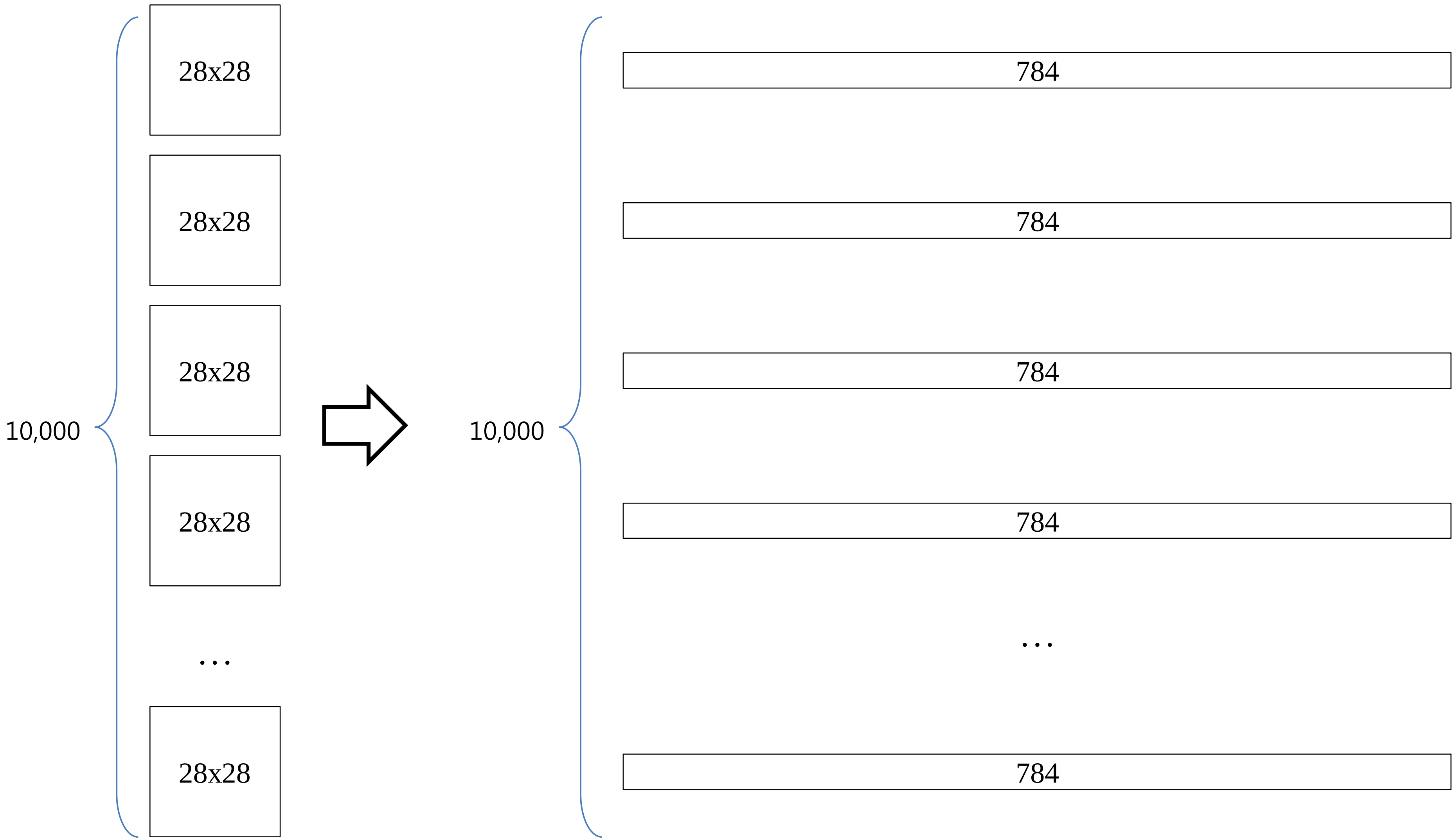
Course Notebooks are available

https://github.com/hugman/deep_learning/tree/master/course/tf101/notebooks

Part II

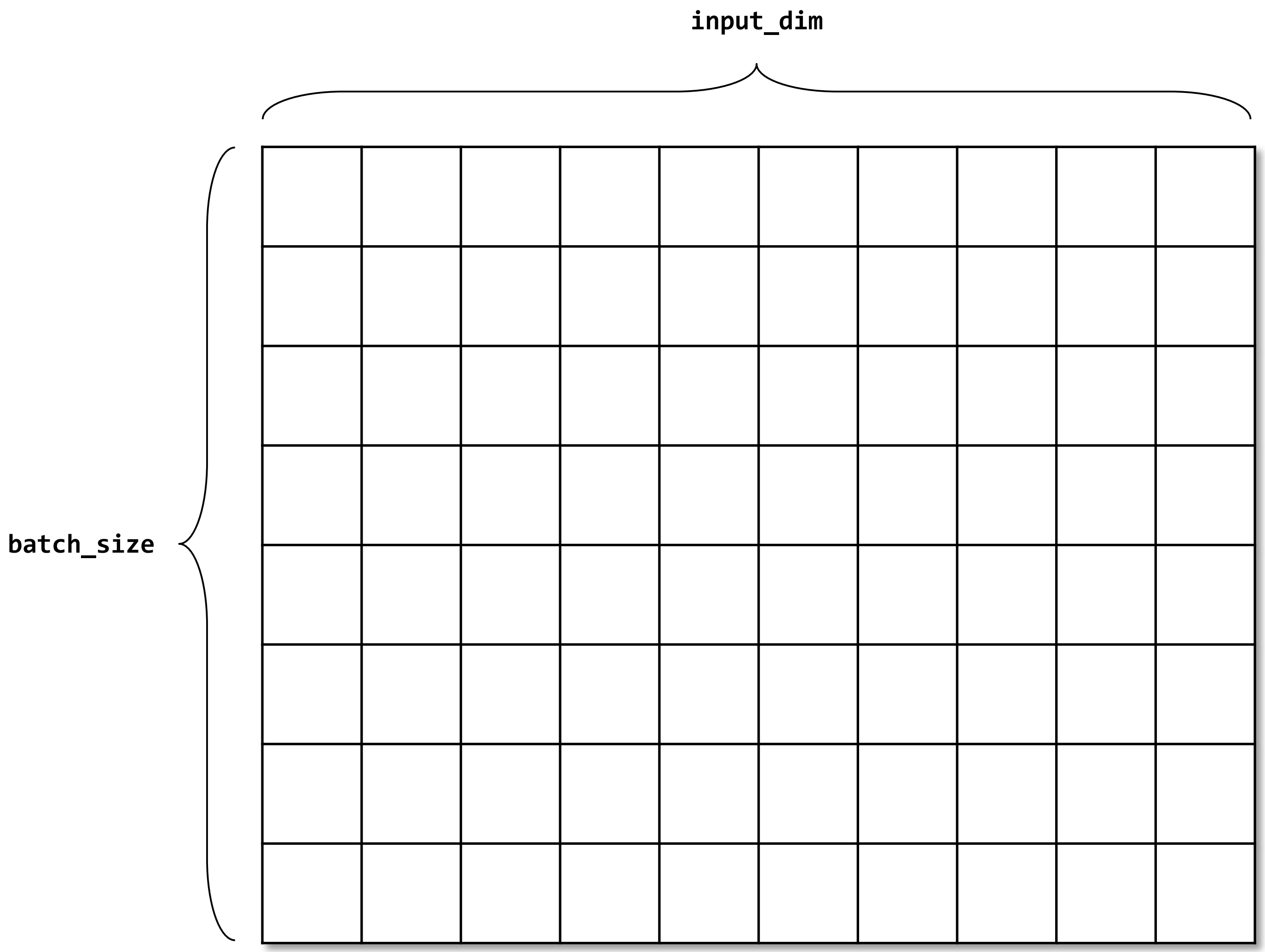
LIVE CODING

Data 가공



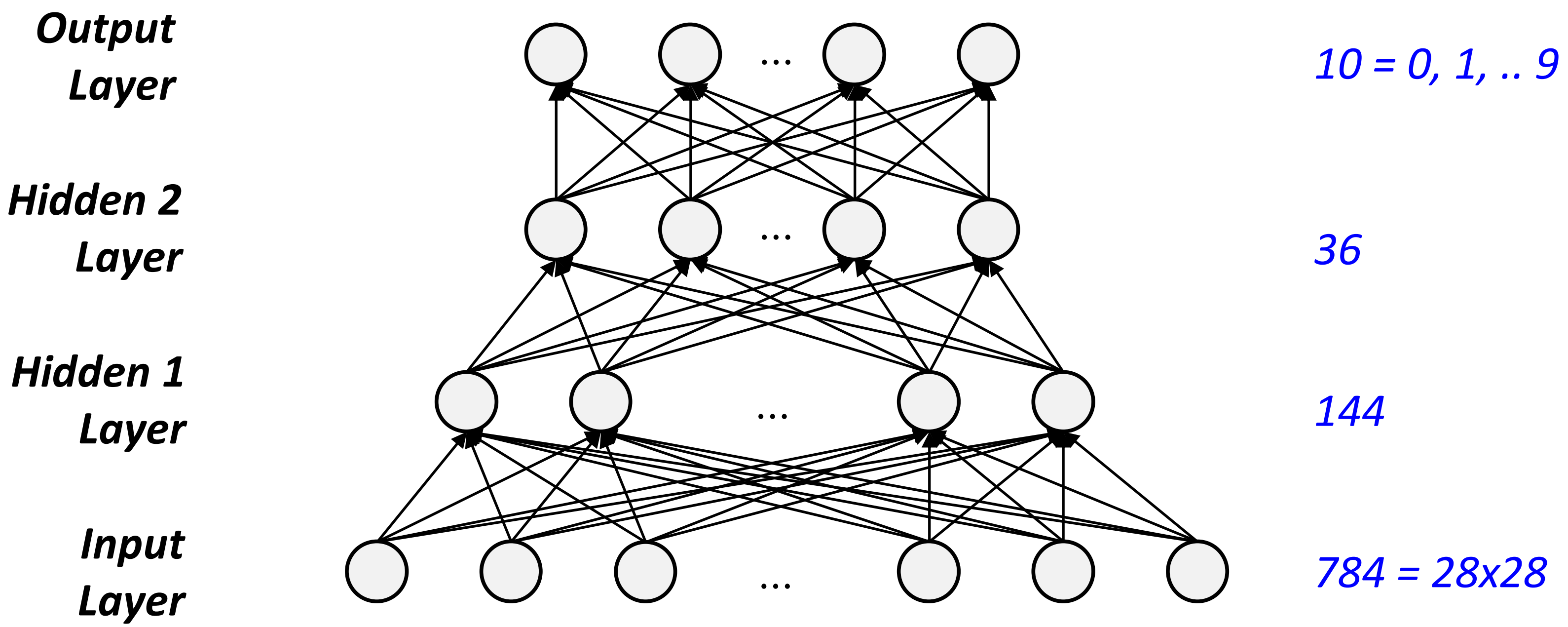
10,000
28x28
28x28
28x28
28x28
...
28x28
10,000
784
784
784
784
...
784

기본 Unit



Model Implementation

- Fully Connected Network



Model | 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

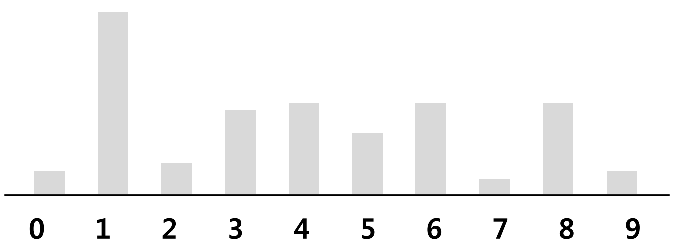
$$\sigma_w = m^{-1/2} \tag{16}$$

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

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

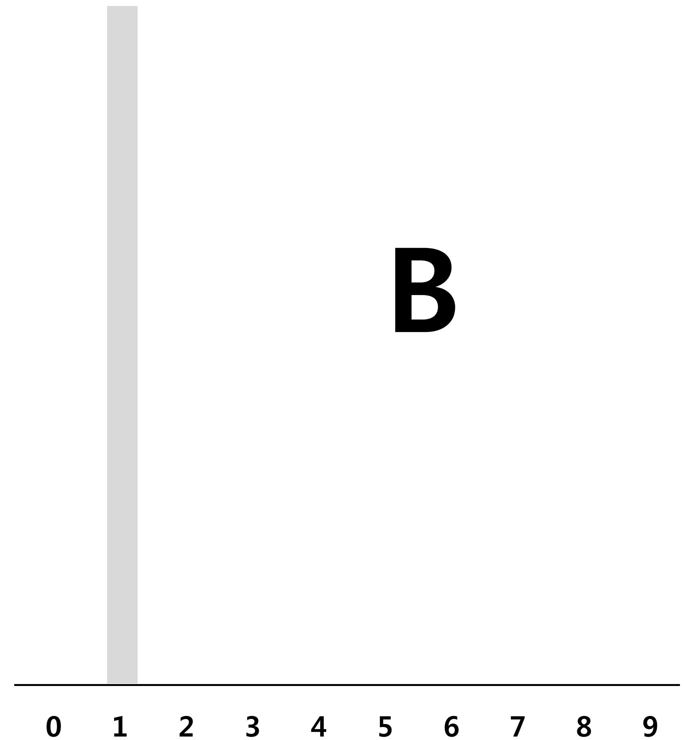
Loss 구현

A



Predicted by Model

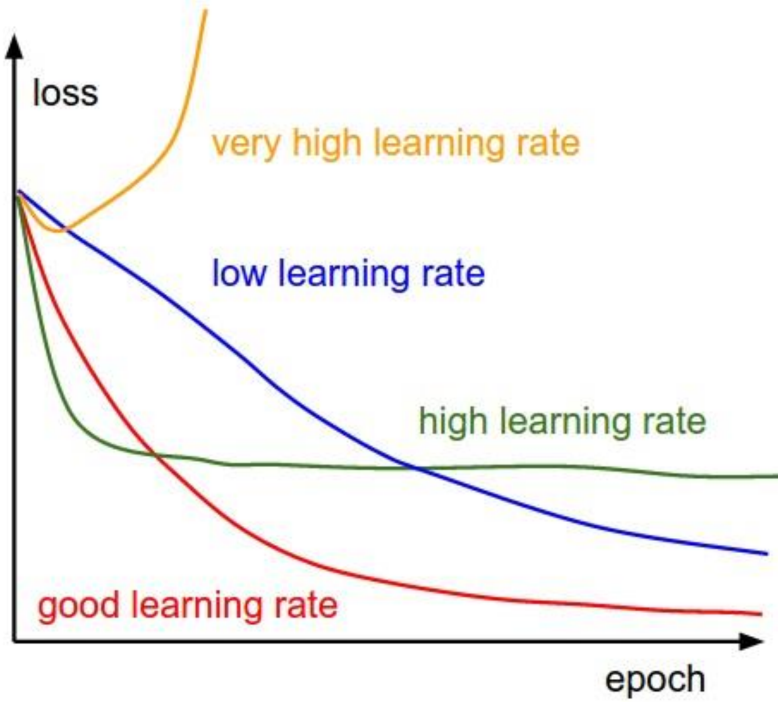
B

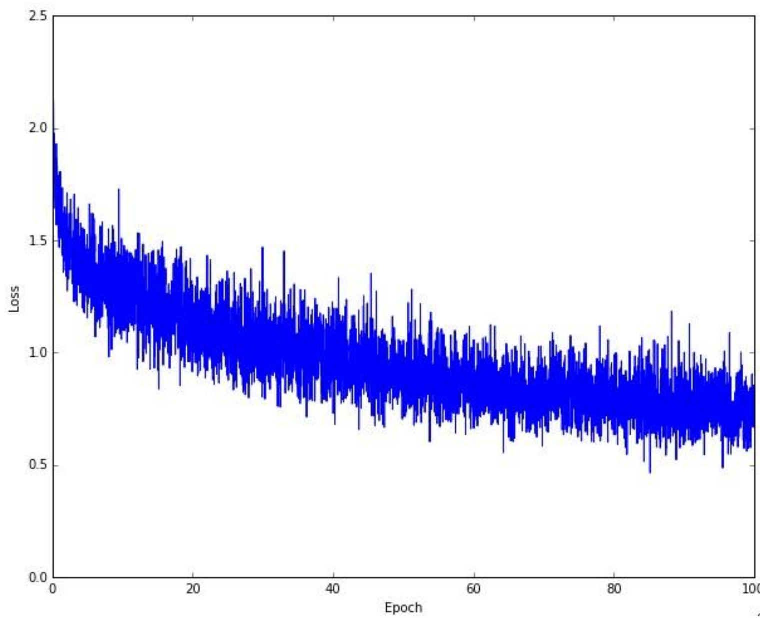


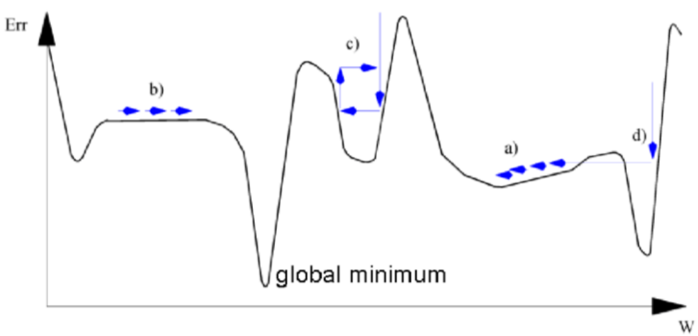
Real Example

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

Tuning Learning Rate

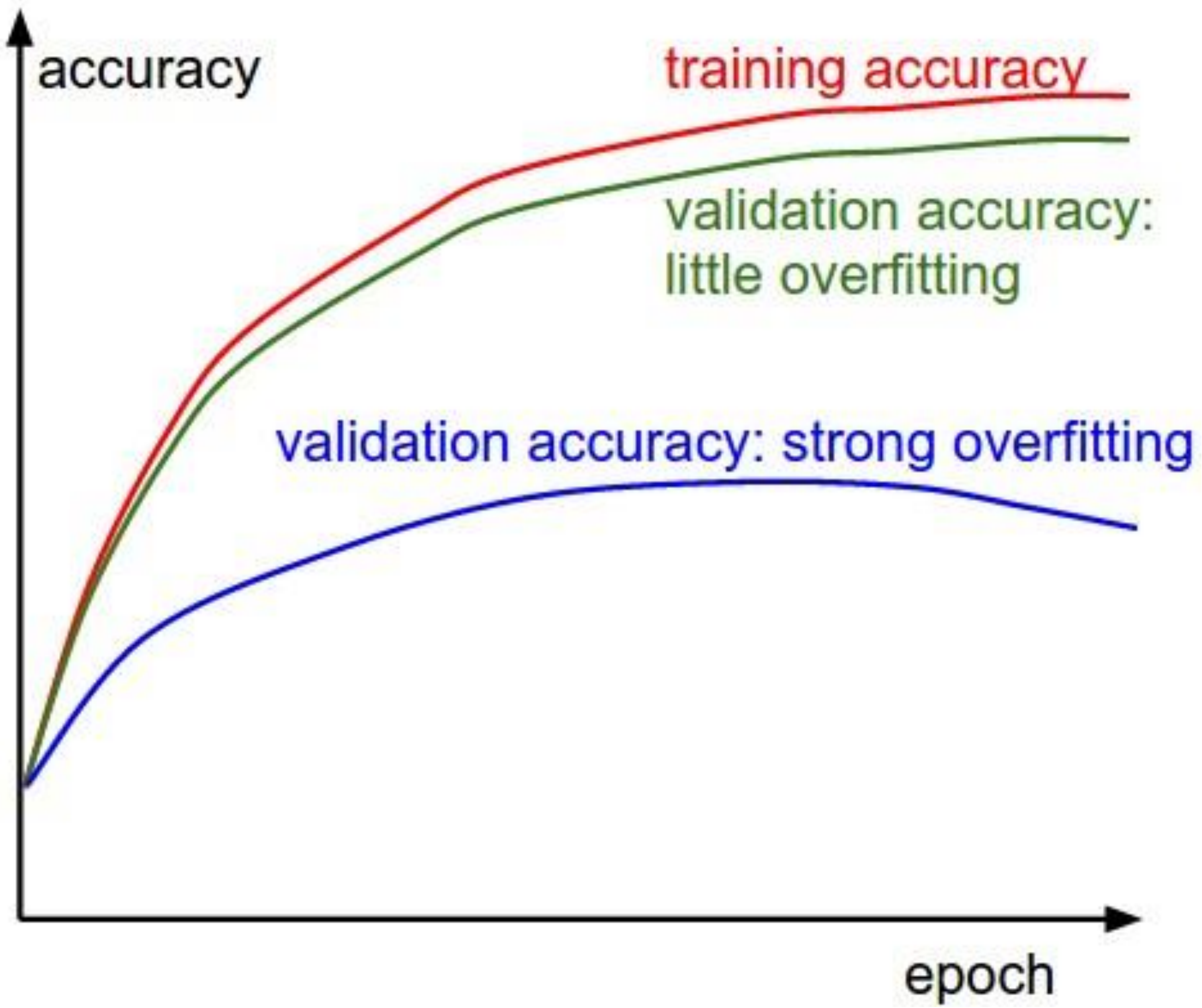






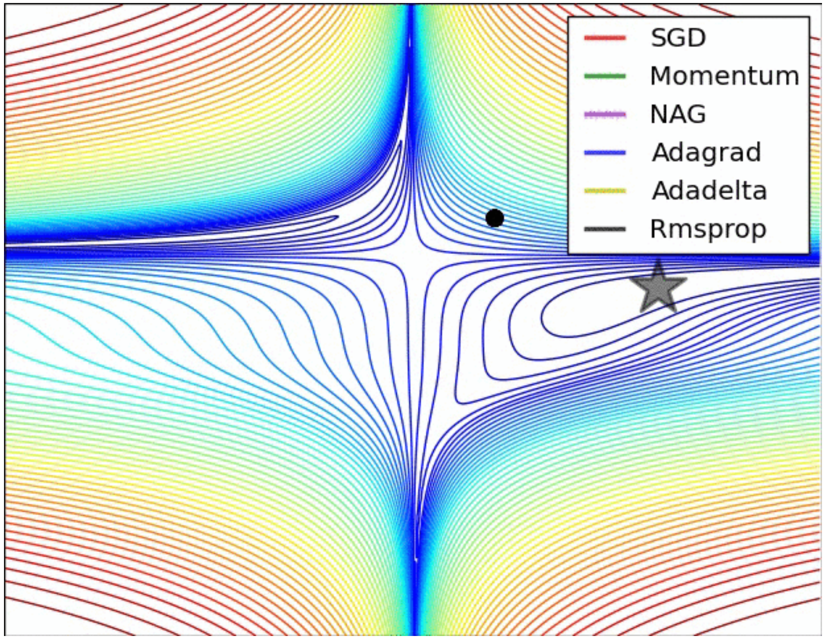
http://cs231n.github.io/neural-networks-3/

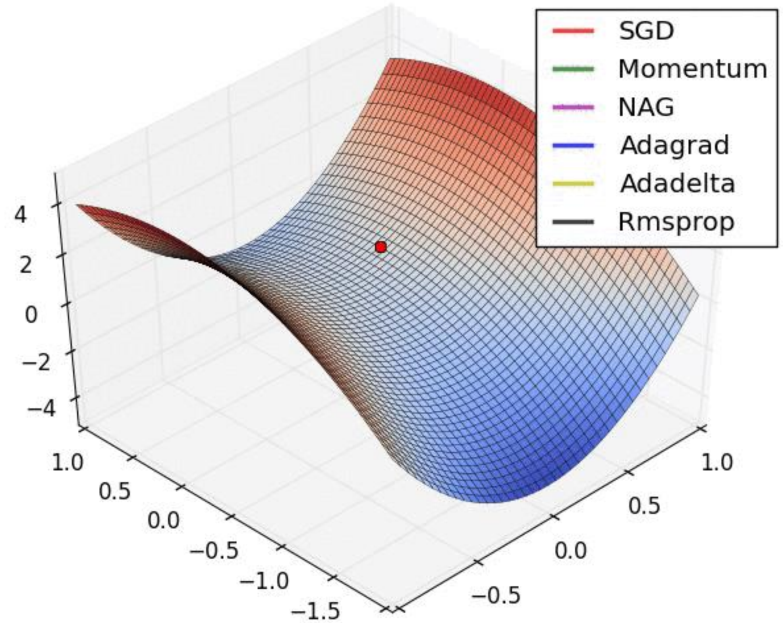
Train / Validation Accuracy



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

Thank you.

Sangkeun Jung(정상근), Ph.D

Intelligence Architect

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