

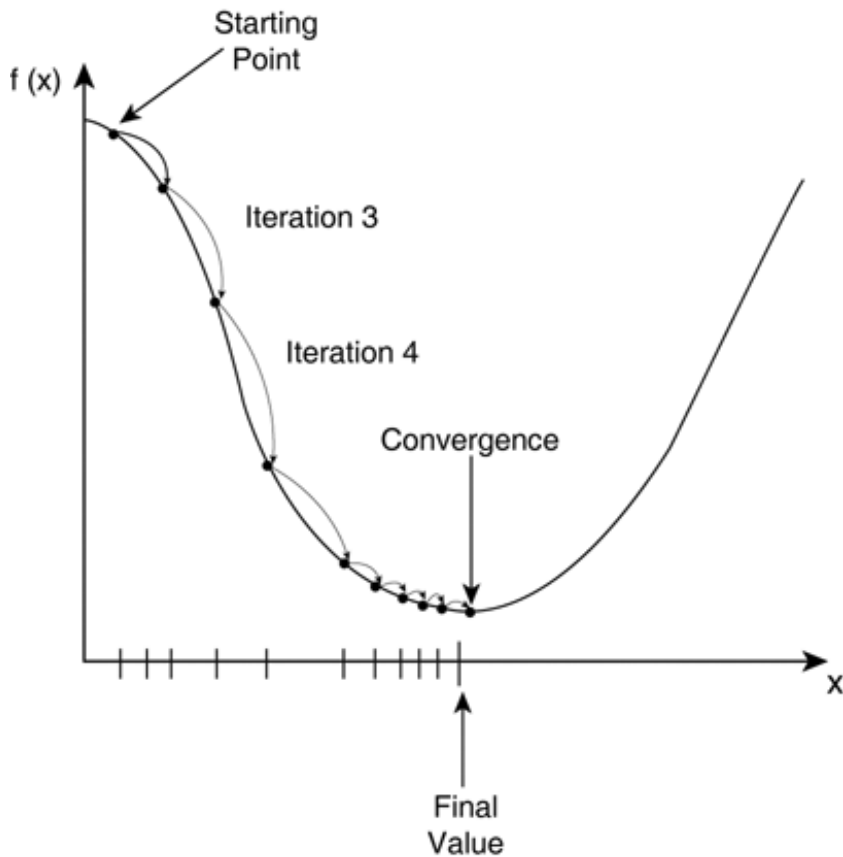
#4. Advanced Topic

Kim Jin Ho

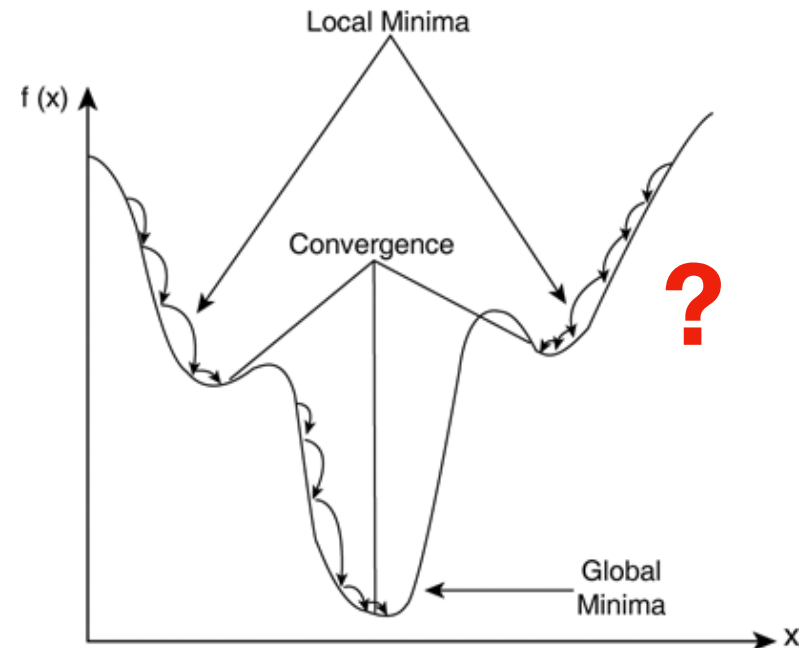
How to Improve Your Model

- 지금까지는, DNN, CNN에 대한 내용과 구현하는 법에 대해 알아보았다.
- Deep Learning은 Hidden Layer를 통한 비선형 변환기법이 주이며,
 - 모델의 구조는 서로 상당히 비슷하다.
- 어떻게 하면 내 모델을 더욱 정교하게 만들 수 있을까?
 - Gradient Descent : Batch, Stochastic, Mini-Batch
 - Optimization Method : Momentum, Adaptive Gradient, RMS Prop, Adam
 - Weight Initialization : He, Xavier Initialization
 - Batch Normalization
 - Avoid Overfitting : L2 Regularization, Dropout

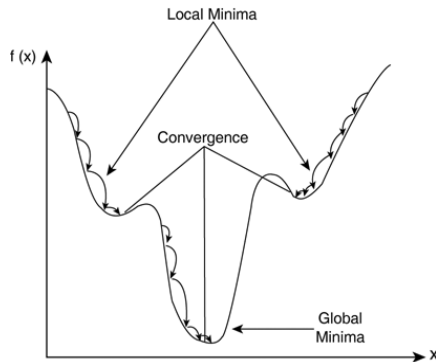
Gradient Descent



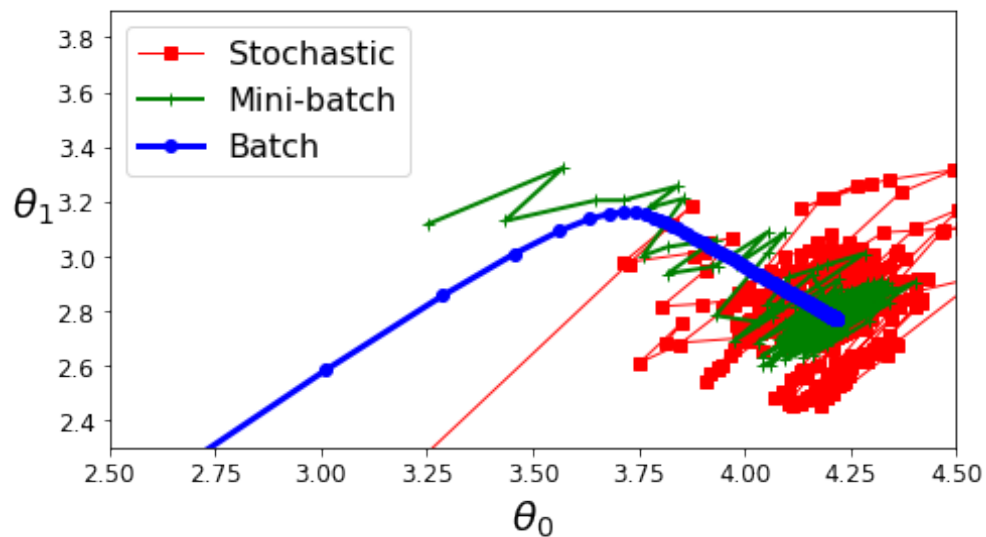
- 우리는 지금까지, 모든 데이터를 한번씩 학습하여 점차적으로 Error를 줄여가는 방법을 사용
- 처음 시작점에서 Iteration을 반복하면서, 오차를 줄여나가 가장 오차가 적은 지점을 탐색



Gradient Descent



- 모든 데이터를 한번씩 학습해야, Cost Function을 **올바르게** 구할 수 있고, 이를 통해 지속적으로 학습해 나간다.
- Local Optimum을 넘으려면.. 올바르게 학습할 필요가 있을까?

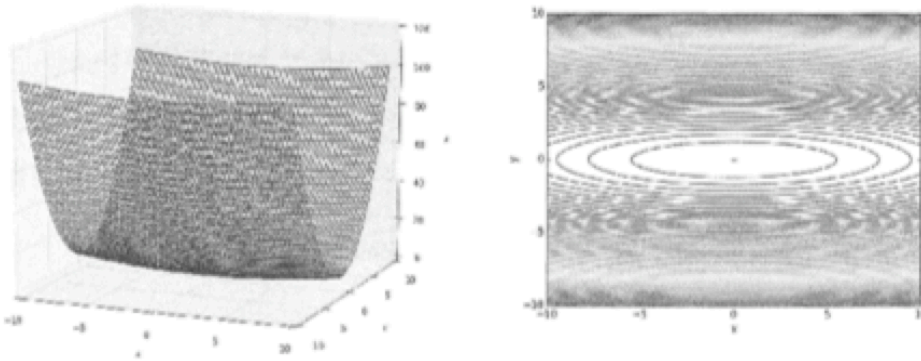


- Batch : 데이터 전체를 학습하자!
- Stochastic : 데이터를 한개만 학습한다면?
- Mini-Batch : 데이터를 조금만 학습한다면?

```
n_epochs = 10
batch_size = 100

with tf.Session() as sess:
    init.run()
    for epoch in range(n_epochs):
        for iteration in range(mnist.train.num_examples // batch_size):
            X_batch, y_batch = mnist.train.next_batch(batch_size)
            sess.run(training_op, feed_dict={X: X_batch, y: y_batch})
            acc_train = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
            acc_test = accuracy.eval(feed_dict={X: mnist.test.images, y: mnist.test.labels})
```

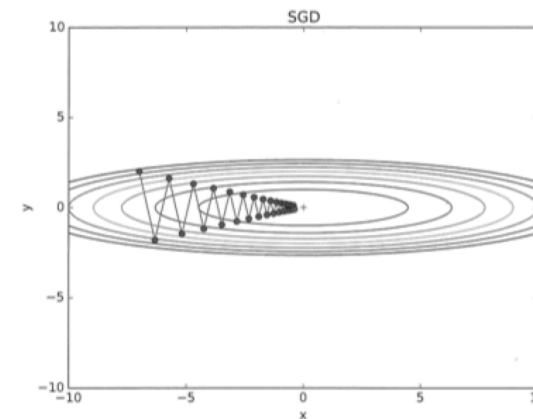
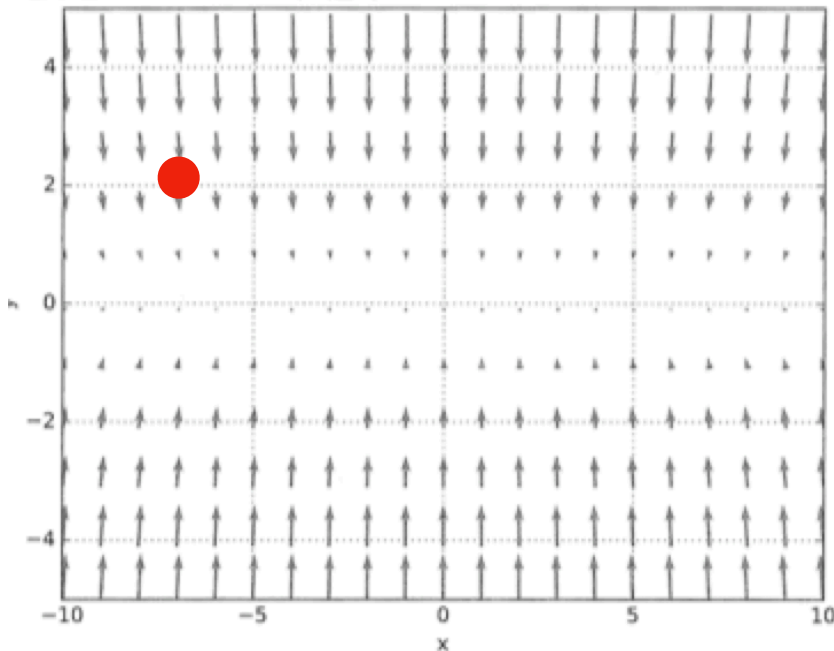
Optimization Method



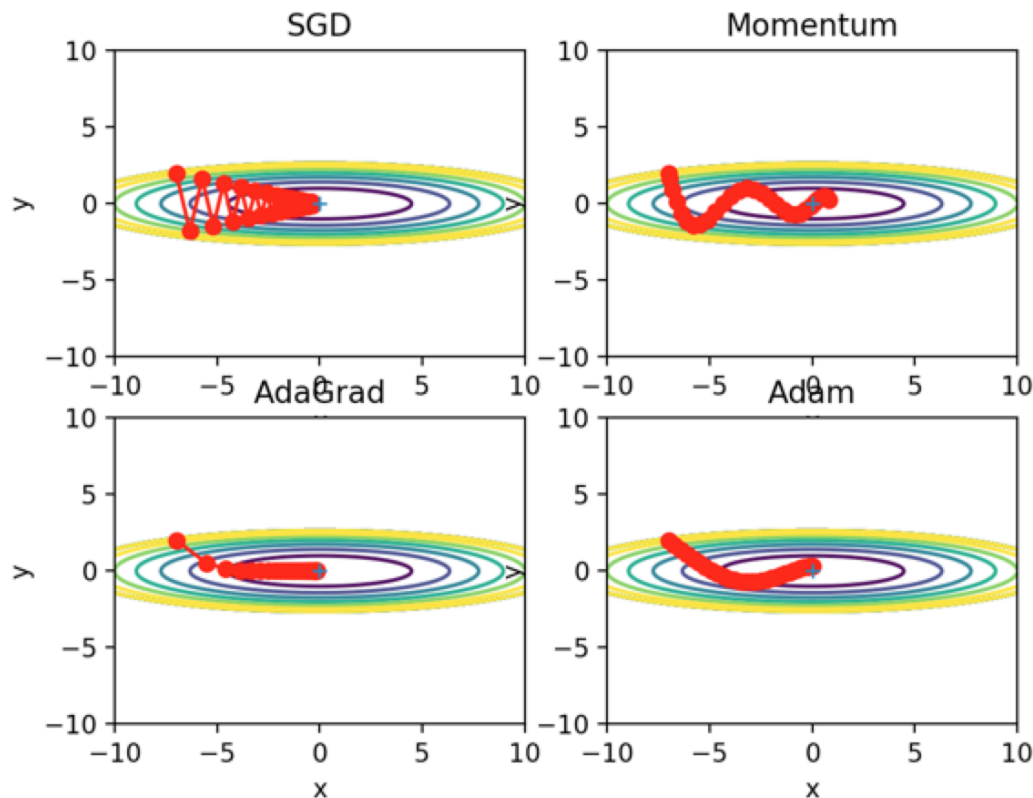
- 함수는 다음과 같은 형태를 갖는다.

$$f(x,y) = \frac{1}{20}x^2 + y^2$$

- 다음과 같은 함수의 최저점을 찾아간다면, x,y 는 어떻게 변하겠는가?

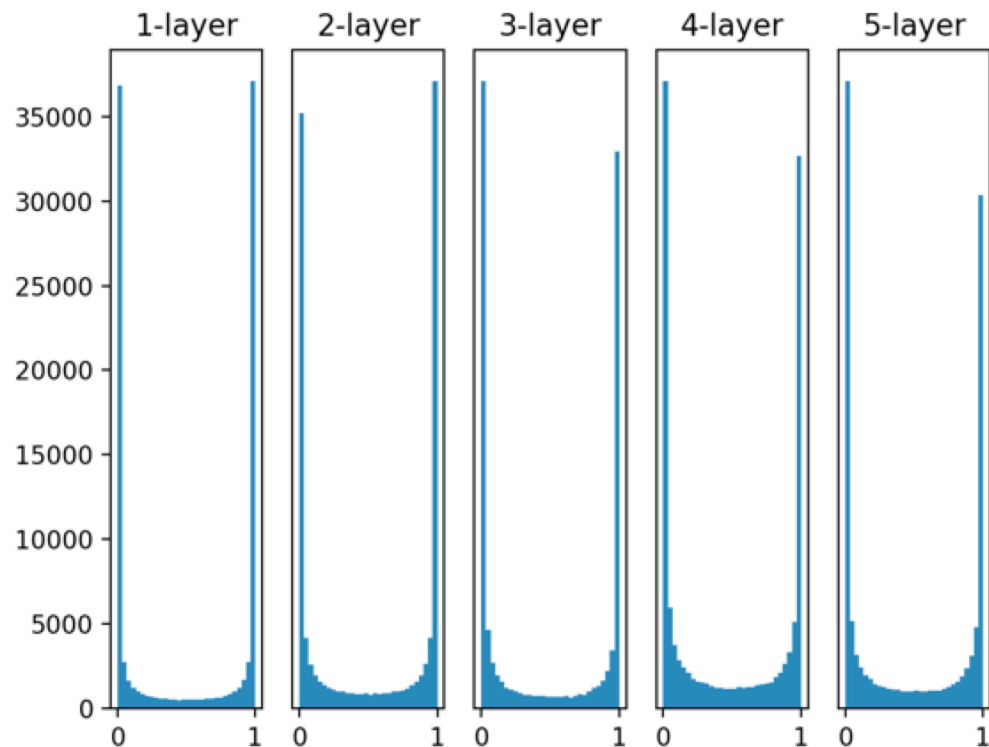


Momentum



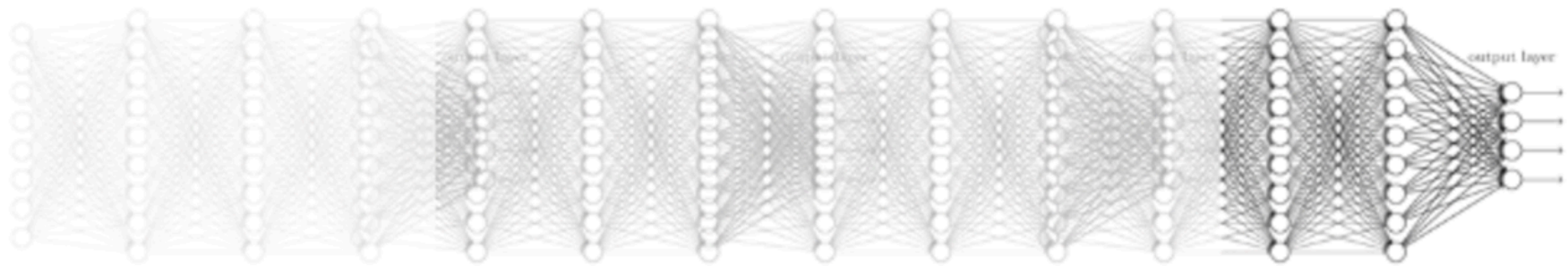
- Momentum
 - 물리학에서 사용하는 Momentum과 같은 개념으로, 움직이는 방향과 같은 방향으로 더 크게 움직인다.
- Adaptive Gradient (AdaGrad)
 - 학습이 진행됨에 따라 learning rate를 조절하는 방법
- Adam
 - Momentum과 AdaGrad의 혼합 형태

Weight Initialization



- 1000개의 input, 5 hidden layers, 100nodes
- Activation function : sigmoid
- 임의의 데이터를 넣었을 때, 나온 결과 값
- 그래프에서 볼 수 있듯이, 각 값의 분포는 0,1에 몰려있는 형태를 갖는다.
- 각 그래프에 의해 나오는 값이 0과 1을 갖는 경우 **Gradient Vanishing**이 발생한다.

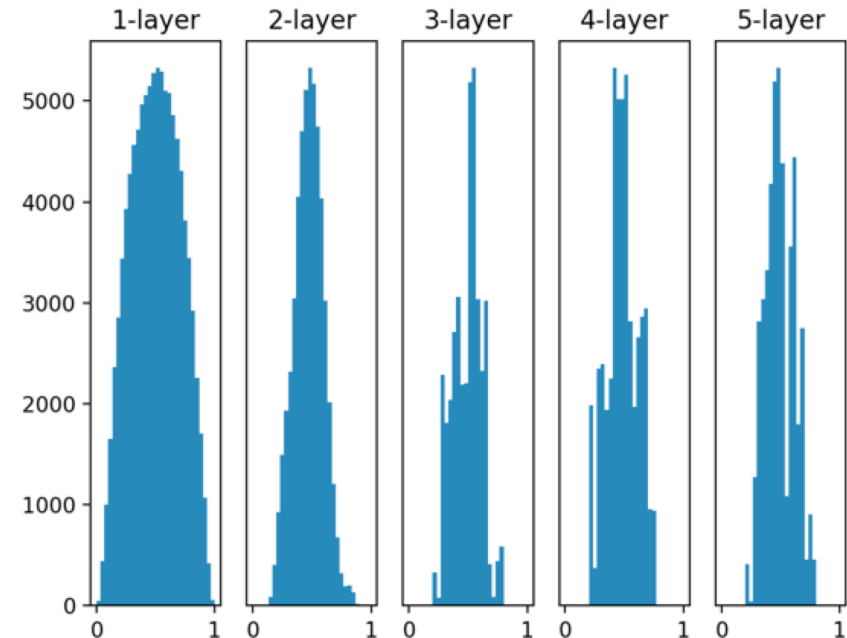
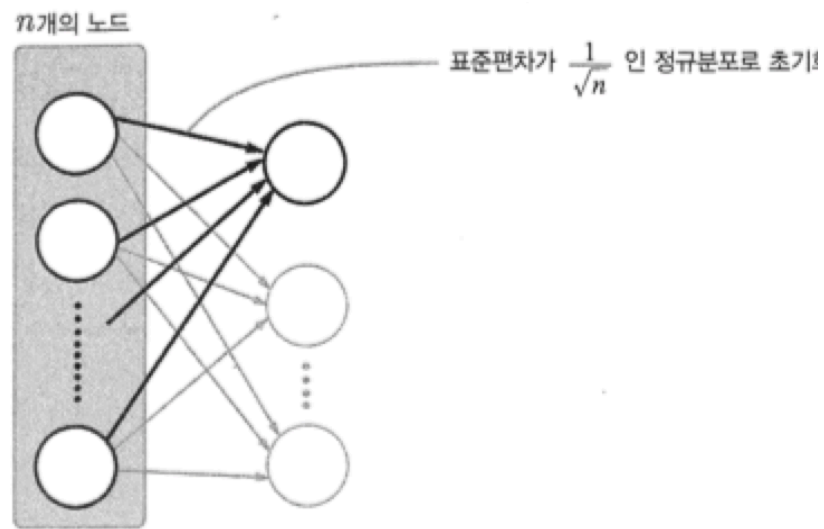
Gradient Vanishing



- 학습을 위한 Gradient 값이 사라져 Backpropagation을 통해 학습이 일어나지 않는 문제를 의미함.
- Hidden Layer가 너무 많이 쌓여있는 경우, 전달 값이 사라진다 : ReLU를 통해 해결!
- Node에서의 output이 0이거나 1인 경우에는?

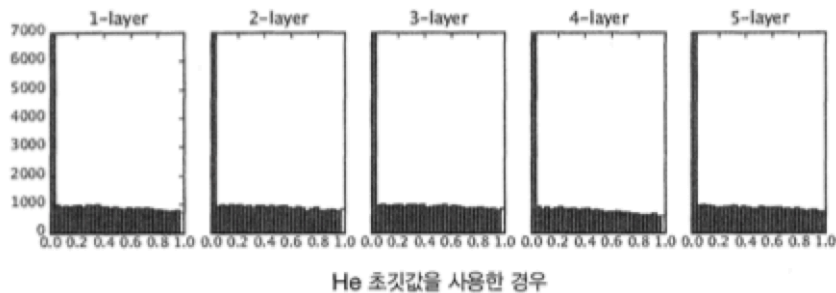
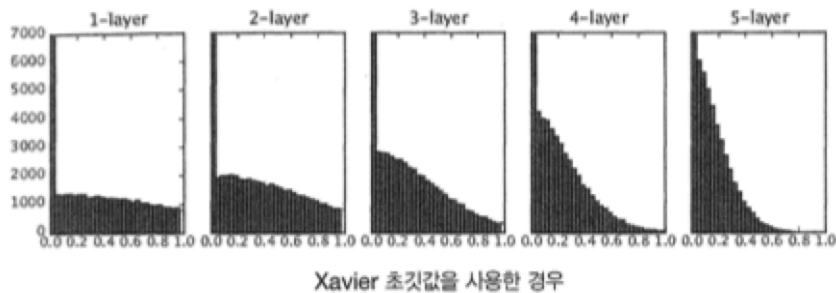
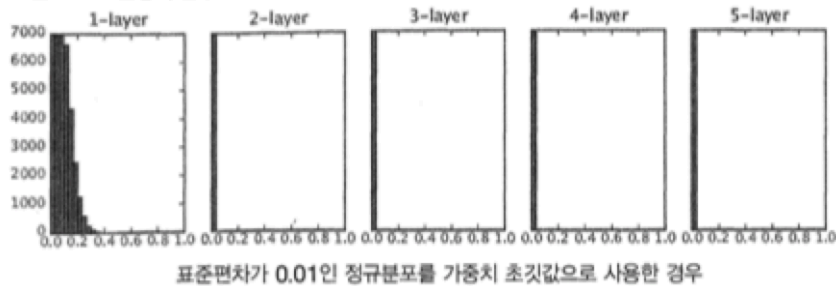
$$\Delta w_{ij} = \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial O} \frac{\partial O}{\partial h_j} \frac{\partial h_j}{\partial w_{ij}}$$

Xavier Initialization



- 이전 hidden layer의 노드가 n 개라면, 초기 weight값을 설정할 때, 표준편차가 $1/\sqrt{n}$ 인 정규 분포를 사용한다.
- 이와 같은 분포를 사용할 경우, 0,1에 집중되지 않은 값이 생성된다.

He Initialization



- 같은 조건에서 activation function이 ReLU인 경우에는?
 - 표준편차가 $\sqrt{2/n}$ 인 분포로 초기값을 설정한다.
- 이 경우, Xavier 초기값은 layer 5로 갈수록 output이 0에 분포되는 반면,
- He 초기값을 사용할 경우, output이 layer에 상관없이 대부분 고르게 나타난다.

Batch Normalization

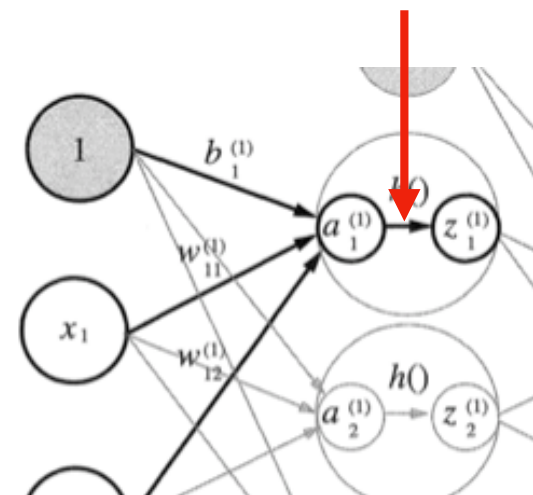
- 예를 들어, hidden layer의 한 node에서 초기 학습이 잘못되어, 매우매우 큰 output 이 나온다면?
- Output의 값에 비례하여 backpropagation이 일어나므로, 다른 node들은 학습이 전혀 되지 않는다.
- 이를 어느정도 방지하려면 어떻게 해야할까?
- Batch Normalization은 각 층의 활성화를 적당히 퍼뜨리도록 강제화 하는 것.

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad \sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad \hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

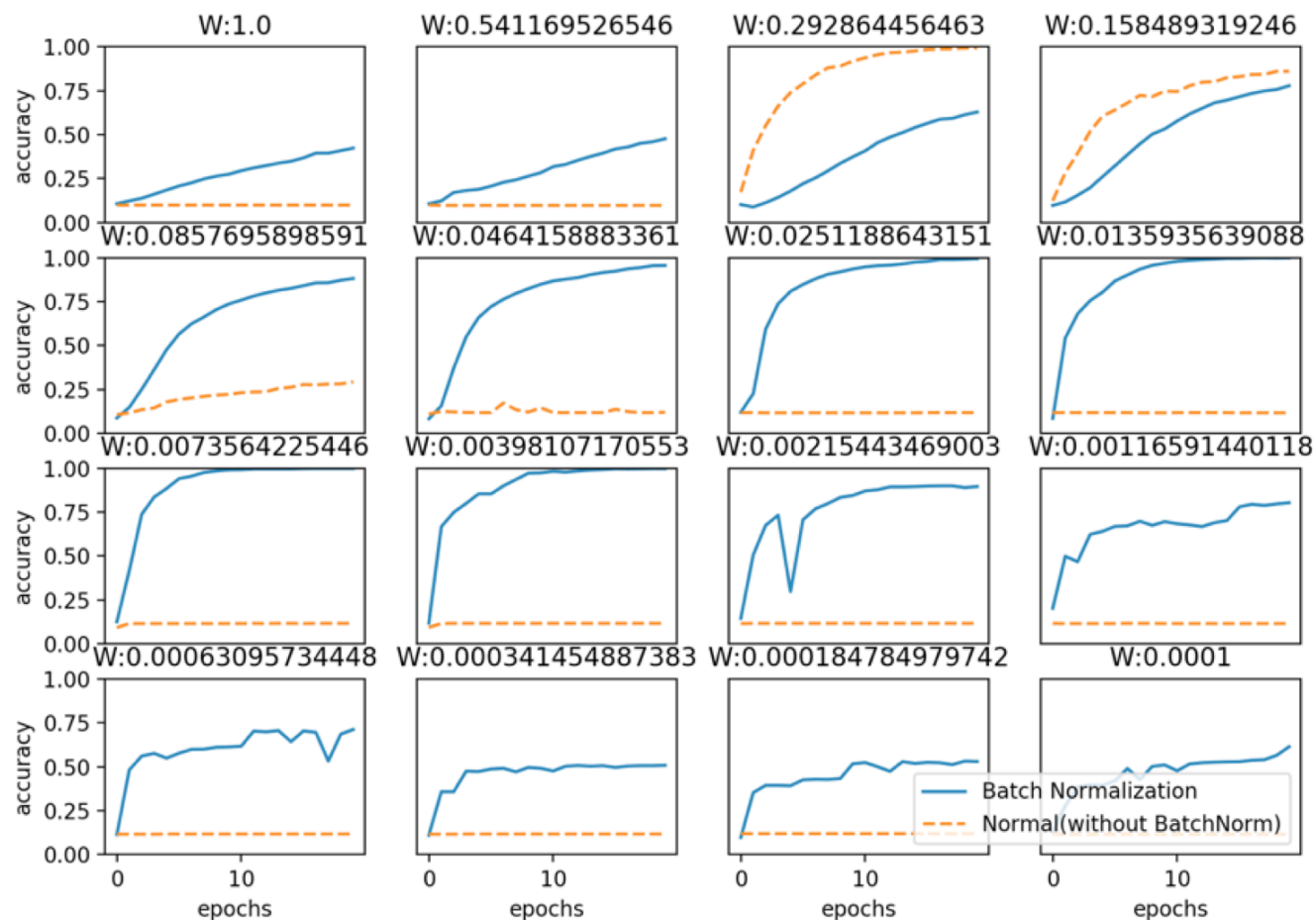
- 복잡한 식 같지만, 평균, 분산, 정규분포화 시키는 식이다.

$$Y \leftarrow \gamma \hat{X} + \beta$$

- 위 식을 통해, weight와 input의 합을 변형한다.



- Learning Rate가 높은 경우에 gradient가 explode/vanish하는 경우, local minima에 빠지지 않게 해준다. → 빠른학습이 가능하게 한다.
- 자체적인 Regularization 효과가 있다. → Dropout이 필요없다 (학습속도 향상).



- MNIST 데이터 기준
- Weight를 특정값으로 시작한 경우,
- Batch Normalization을 쓰는 경우와 쓰지 않은 경우에 대한 비교

Avoid Overfitting

- Batch Normalization은 Input이 어떤 방식으로 나오던 정규화를 해주는 방법이다.
 - 하지만, Weight가 너무 커지는 것을 막는 방법은 없을까?

$$\arg \max_{\theta} \sum_{i=1}^m \log(y_i | x_i; \theta) - \alpha R(\theta)$$

- 우리가 원하는 것은 input과 weight를 통해, 정답과 가장 근접하게 나오는 것이 목표인데, 거기에서 weight의 합을 빼게 해주면 되지 않을까?
- 이와 같은 방법을, 정규화 (Regularization) 라 한다.

L2 Regularization

- 어떤 값을 뺄 것인가?

$$\text{L1 정규화항} \\ R(\theta) = \|\theta\|_1 = \sum_{i=1}^n |\theta_i|$$

$$\text{L2 정규화항} \\ R(\theta) = \|\theta\|_2^2 = \sum_{i=1}^n \theta_i^2$$

Case1	Case2
1	2
1	2
1	2
1	2
4	2
4	2

- Case 1의 경우,
 - $L1 = 1 + 1 + 1 + 1 + 4 + 4 = 12$
 - $L2 = 1 + 1 + 1 + 1 + 16 + 16 = 36$
- Case 2의 경우,
 - $L1 = 2 * 6 = 12$
 - $L2 = 4 * 6 = 24$
- 큰 값을 제어하기 위해서는 L2가 효과적이다!

Python Example

- 지금까지 배웠던 DNN 코드에, 각각의 방법을 하나씩 추가하여 보자.

DNN (2 hidden layers, [300,100], Softmax function)

Mini-batch : 50

Xavier Initializer

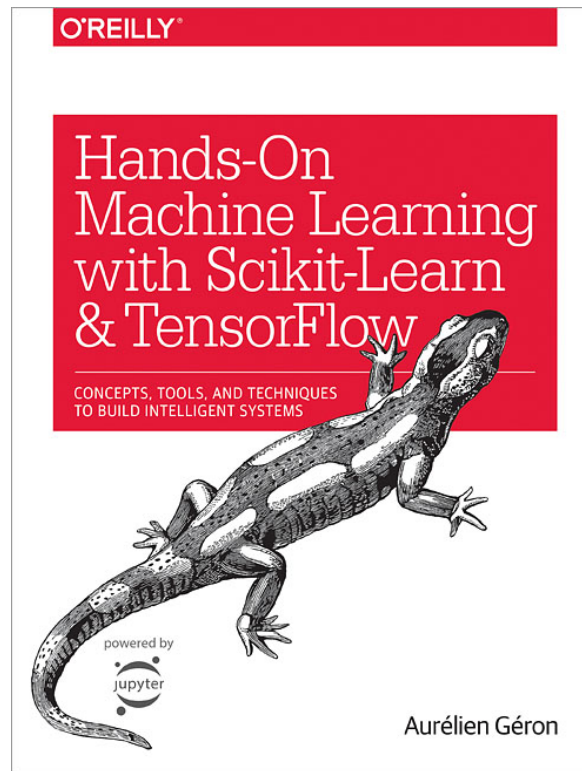
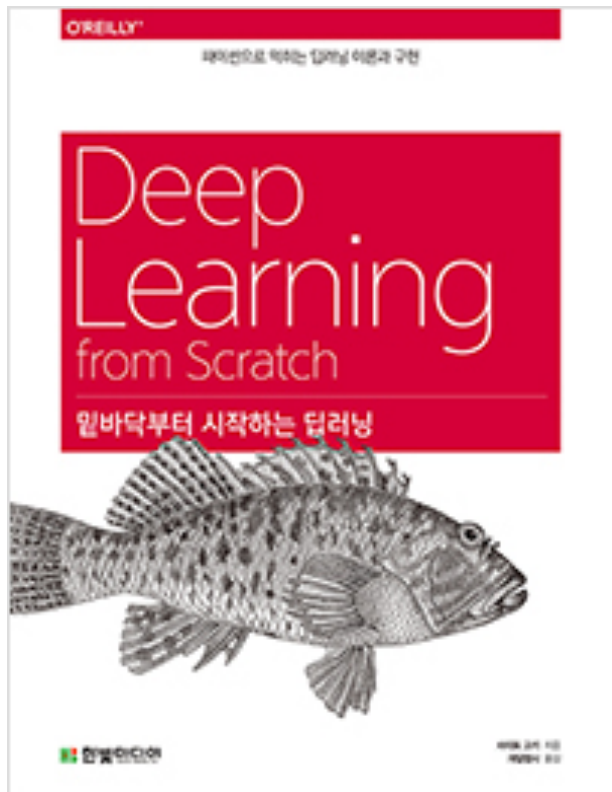
```
0 Train accuracy: 0.9 Test accuracy: 0.9128
1 Train accuracy: 0.94 Test accuracy: 0.9291
2 Train accuracy: 0.92 Test accuracy: 0.9398
3 Train accuracy: 0.96 Test accuracy: 0.945
4 Train accuracy: 0.92 Test accuracy: 0.9513
5 Train accuracy: 0.94 Test accuracy: 0.9543
6 Train accuracy: 0.98 Test accuracy: 0.9556
7 Train accuracy: 0.96 Test accuracy: 0.9593
8 Train accuracy: 0.92 Test accuracy: 0.9625
9 Train accuracy: 0.96 Test accuracy: 0.9645
```

Gradient Descent
w/o Batch Normalization

```
0 Train accuracy: 0.98 Test accuracy: 0.9686
1 Train accuracy: 1.0 Test accuracy: 0.9749
2 Train accuracy: 0.98 Test accuracy: 0.9771
3 Train accuracy: 1.0 Test accuracy: 0.9803
4 Train accuracy: 1.0 Test accuracy: 0.9808
5 Train accuracy: 0.98 Test accuracy: 0.9815
6 Train accuracy: 0.98 Test accuracy: 0.9822
7 Train accuracy: 1.0 Test accuracy: 0.9829
8 Train accuracy: 0.98 Test accuracy: 0.9833
9 Train accuracy: 1.0 Test accuracy: 0.9799
```

Adam Optimizer
w/ Batch Normalization

Reference



[Deep Learning](#)

An MIT Press book

Ian Goodfellow and Yoshua Bengio and Aaron Courville

[Exercises](#) [Lectures](#) [External Links](#)

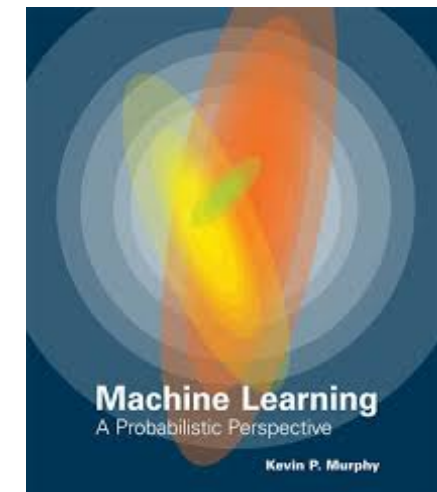
The Deep Learning textbook is a resource intended to help students and practitioners enter the field of machine learning in general and deep learning in particular. The online version of the book is now complete and will remain available online for free.

The deep learning textbook can now be pre-ordered on [Amazon](#). Pre-orders should ship on December 16, 2016.

For up to date announcements, join our [mailing list](#).

Citing the book

<http://www.deeplearningbook.org/>



Q&A

김진호

icbf98@khu.ac.kr