Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University

Deep Learning

Keras

Application & Tips(1)

Learning, data preparation, overfitting, Optimizer, Batch Normalization

Yoonjoong Kim

Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

Content

1. Learning rate
   1. Gradient descent algorithm
   2. Large rate
   3. Small rate
2. Data preprocessing for the gradient decent algo rithm
   1. Mean, std normalization
   2. Min-max normalization
3. Overfitting
   1. More training data
   2. Reduce the number of features
   3. Regularization
   4. Dropout
4. 최적화기(Optimizer)의 종류
5. Batch Normalization
6. Examples
   1. Learning rate
   2. Dataset normalization
   3. Mnist digit classifier
      1. Mnist 이미지 데이터 분석
      2. Model 개발 및 평가
      3. 이미지 인식 및 출력방법
   4. Application Tips for the Mnist digit classifier

Recap

- 선형회귀, 로지스틱 분류와 softmax 모델

- **로지스틱 회귀모델, 이진분류**
  - 데이터셋 $X = \{x_i\}, Y = \{y_i\}$
  - 모델 $\hat{y} = h(x) = \sigma(\hat{y}_l), \sigma(z) = \frac{1}{1+e^{-z}}$,sigmoid
    $\hat{y}_l = wx + b$
  - 손실함수(binary_crossentropy)
    $loss(w,b) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\bar{y_i}) + (1-y)\log(1-\bar{y_i}))$
  - 학습 $w^*, b^* = argmin_{w,b}(loss(w,b)))$
  - 예측 $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수 $accuracy = \frac{1}{N}\sum(y_i = \hat{y}_i)$

- **선형회귀 모델**
  - 데이터셋 $X = \{x_i\}, Y = \{y_i\}$
  - 모델 $\hat{y} = h(x) = wx + b$
  - 손실함수(mean square error)
    $loss(w,b) = \frac{1}{N}\sum_{i=1}^{N}(y_n - \hat{y}_i)^2$
  - 학습 $w^*, b^* = argmin_{w,b}(loss(w,b)))$
  - 예측 $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수 $R^2 = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2}$

- **Softmax 분류모델**
  - 데이터셋 $X = \{x_i\}, Y = \{y_i\}$
  - 모델 $\hat{y} = h(x) = \sigma(\hat{y}_l), \sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$, softmax
    $\hat{y}_l = wx + b$
  - 손실함수(categorical_crossentropy)
    $loss(w,b) = -\frac{1}{N}\sum_j y_j \log(\hat{y}_i)$
  - 학습 $w^*, b^* = argmin_{w,b}(loss(w,b)))$
  - 예측 $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수 $accuracy = \frac{1}{N}\sum(y_i = \hat{y}_i)$

1. Learning rate

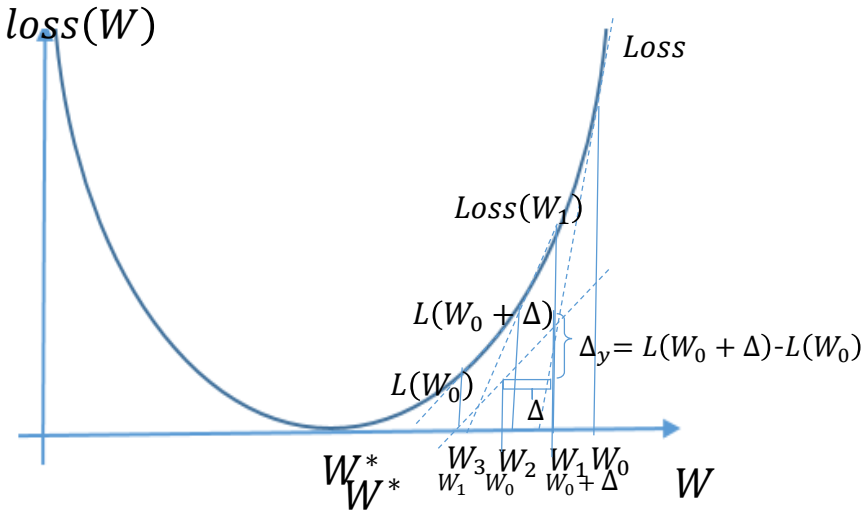
- Deep Learning 응용 프로그램의 일반적인 구조

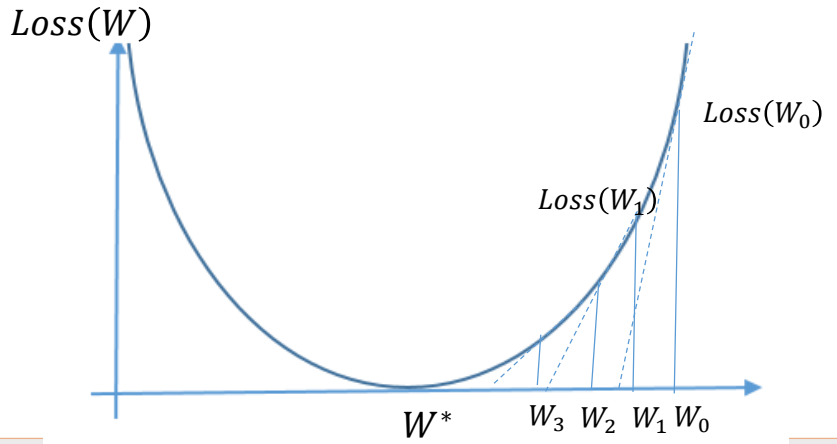
  - 데이터셋
    - $\mathrm{X} = \{\mathrm{x_i}\}, Y = \{y_i\}$
  - model
    - $h(X)$ 선형, 로지스틱, 소프트맥스
  - logit
    - logit= $h(X)$
  - loss
    - $loss$ =loss(logit,Y) mse, binary_crossentropy, categorical_crossentropy
  - 학습
    - $w^*, b^* = argmin_{w,b}(loss(w,b))$ gradient descent algorithm, error-back propagation

1. Learning rate

- Gradient descent algorithm
  - $W^* = \underset{W}{ArgMin} \frac{\partial}{\partial w} loss(W)$
    - $W_{t+1} = W_t - \alpha \frac{\partial}{\partial w} loss(W_t)$
    - $loss(W|X,Y) = dist(H(X|W,b),Y)$
  - $W_1 = W_0 - \alpha \, \Delta loss(W_0)$
    - $\Delta loss(W_0) = \left| \frac{\partial}{\partial w} loss(W) \right|_{W=W_0}$

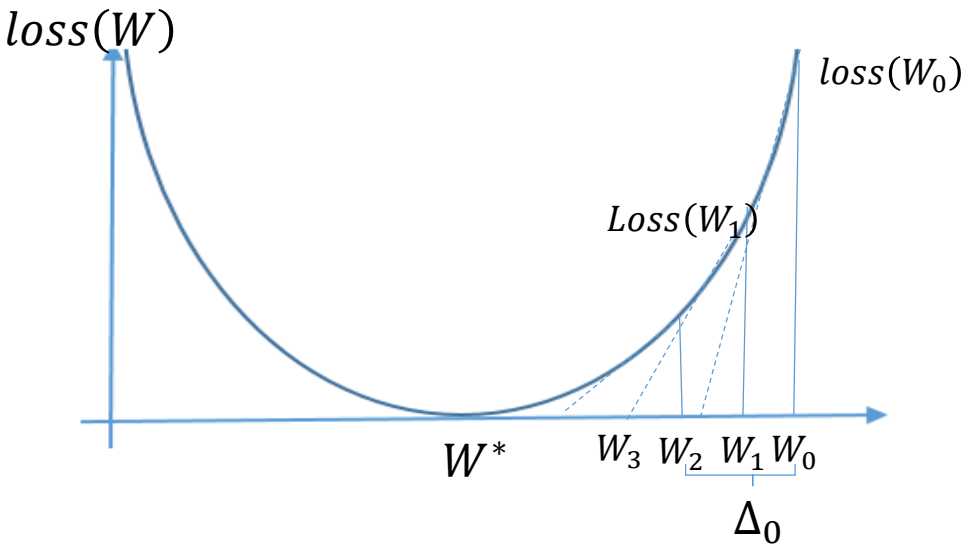
      $= \lim_{\Delta \to 0} \frac{loss(W_0+\Delta) - loss(W_0)}{\Delta} = \lim_{\Delta \to 0} \frac{\Delta_y}{\Delta}$
  - 훈련과정
    - $initialize\ W_0$
    - $W_1 = W_0 - \alpha \, \Delta L(W_0)$
    - $W_2 = W_1 - \alpha \, \Delta L(W_1)$
    - ……
    - W^*





1. Learning rate(cont.)

- 1.1 Gradient descent algorithm α



$initialize\ W_0$

$W_1 = W_0 - \alpha\, \Delta loss(W_0)$

$W_2 = W_1 - \alpha\, \Delta loss(W_1)$

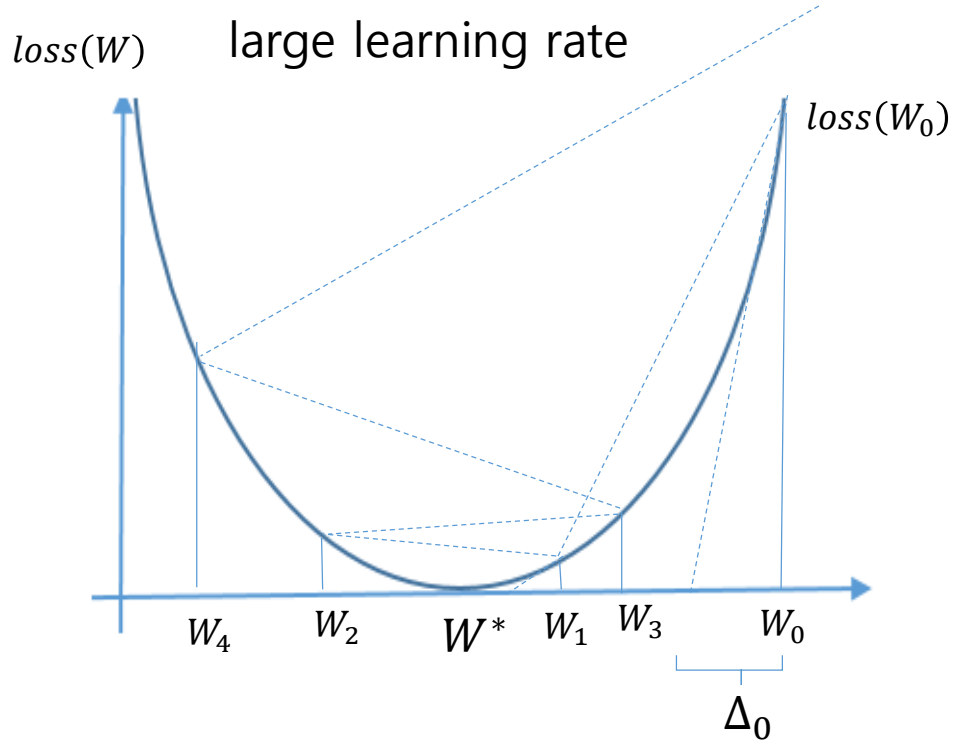
.....

W^*

```
model=sequential()
model.add(Dense(1,input_dim=3))
sgd = optimizers.SGD(lr=α)
model.compile(loss='mse',optimizer=sgd)
model.fit(X,Y,epochs=1000)
```

1. Learning rate(cont.)

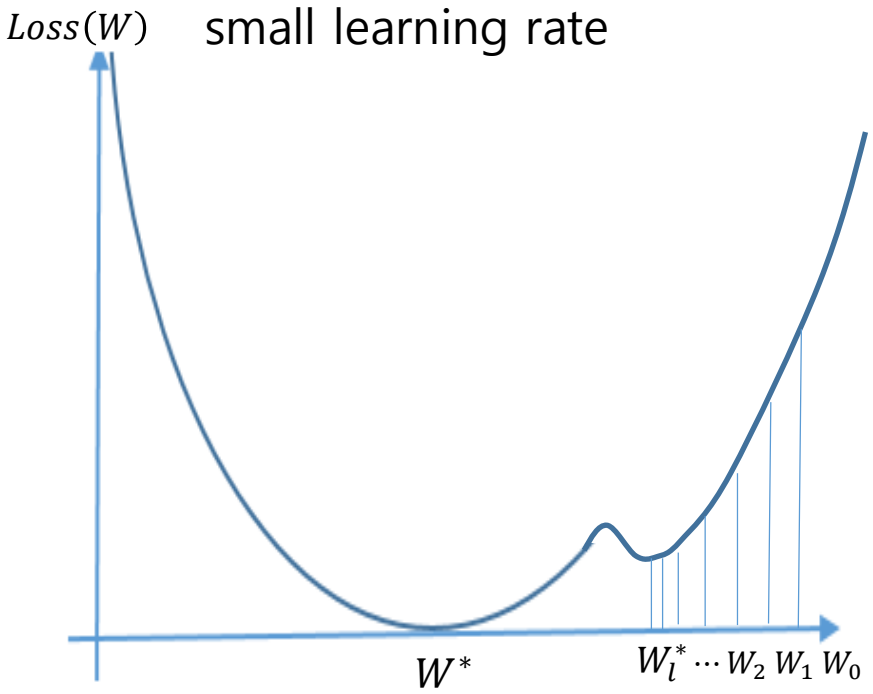
- 1.2 Large learning rate α=10 : overshooting(발산)
  - Learning
    - $W_{n+1} = W_n - \alpha \Delta Loss(W_n)$
    - W0 => W* ?
      W* :global minimum
  - $\alpha = 10, large\ value$
  - $initialize\ W_0$
    - $W_1 = W_0 - 10\Delta L(W_0)$
    - $W_2 = W_1 - 10\Delta L(W_1)$
    - ……
    - W^*
    - ⇒
    - 발산 (divergence)



  - How to converge to W^*?
    수렴 (convergence) ?

1. Learning rate(cont.)

- 1.3 Small learning rate α=0.001 => local minimum point
  - Learning
    - $W_{n+1} = W_n - \alpha \Delta \text{Loss}(W_n)$
    - W0 => W* ?
      W* :global minimum
    - $\alpha = 0.001, small\ value$
    - $initialize\ W_0$
      - $W_1 = W_0 - 0.001 \Delta L(W_0)$
      - $W_2 = W_1 - 0.001 \Delta L(W_1)$
      - ……
      - W^*
      - ⇒
      - $local\ minimum\ point$
  - How to converge to W^*?
    수렴 (convergence) ?



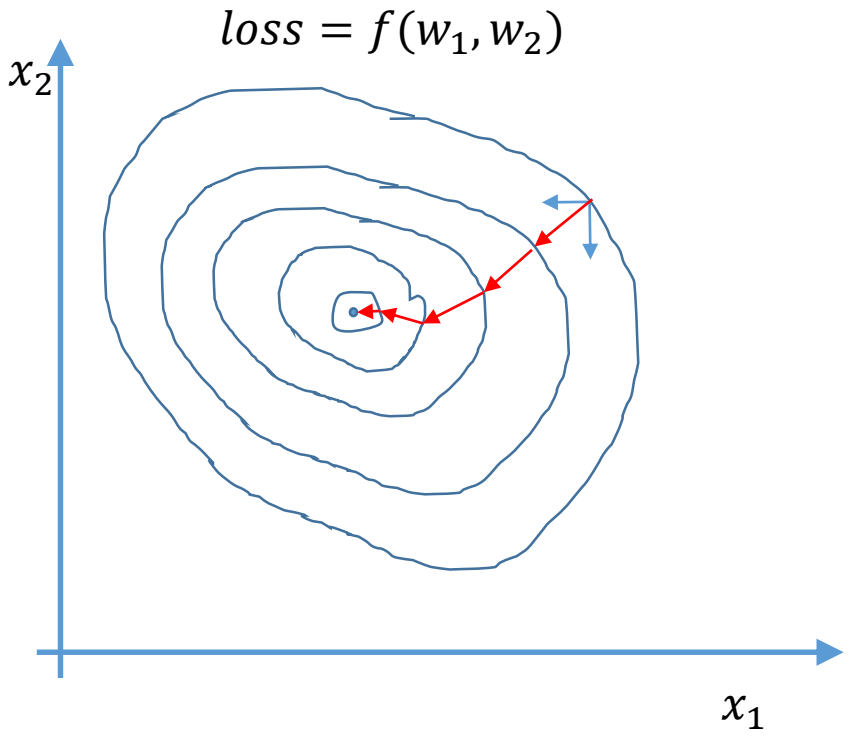
Local minimum point
W_l^*로 수렴 (convergence)

1. Learning rate

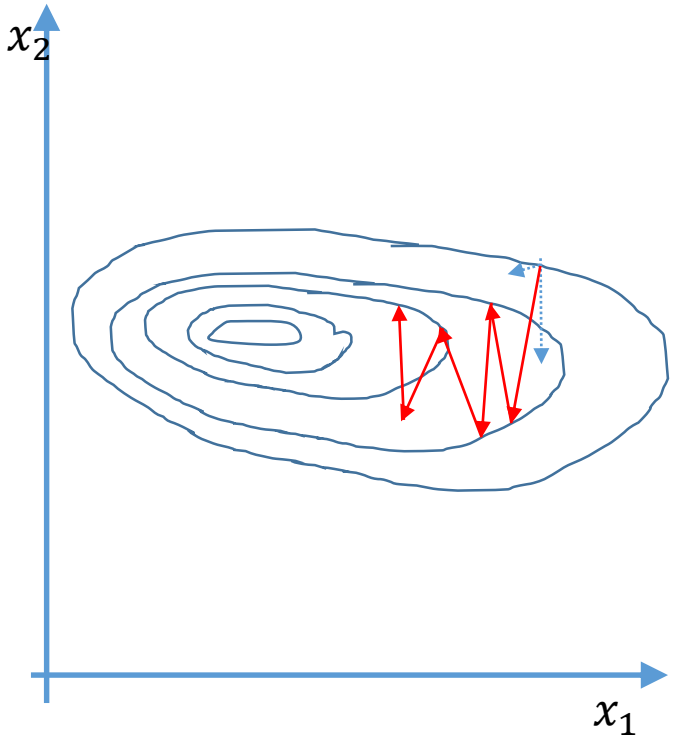
- 학습률(learning rate) 선정 방법?
  - 여러 개의 최적화기와 학습률에 대하여 시도한다.
    - Cost(loss) 함수의 값을 관찰한다.
    - 순리적으로 작아지는지 점거한다.
    - global minimum point W^* 롤 수렴하는지 점검한다.
- 최적화기(Optimizer)의 종류
  - Momentum
  - SGD(Stochastic Gradient Descent)
  - Adagrad(Adaptive Gradient)
  - RMSProp
  - **Adam** (Adaptive Moment Estimation)

2. Data preprocessing for a gradient descent algorithm

- 2.1 Problems during deep learning
- Gradient descent algorithm)의 학습절차



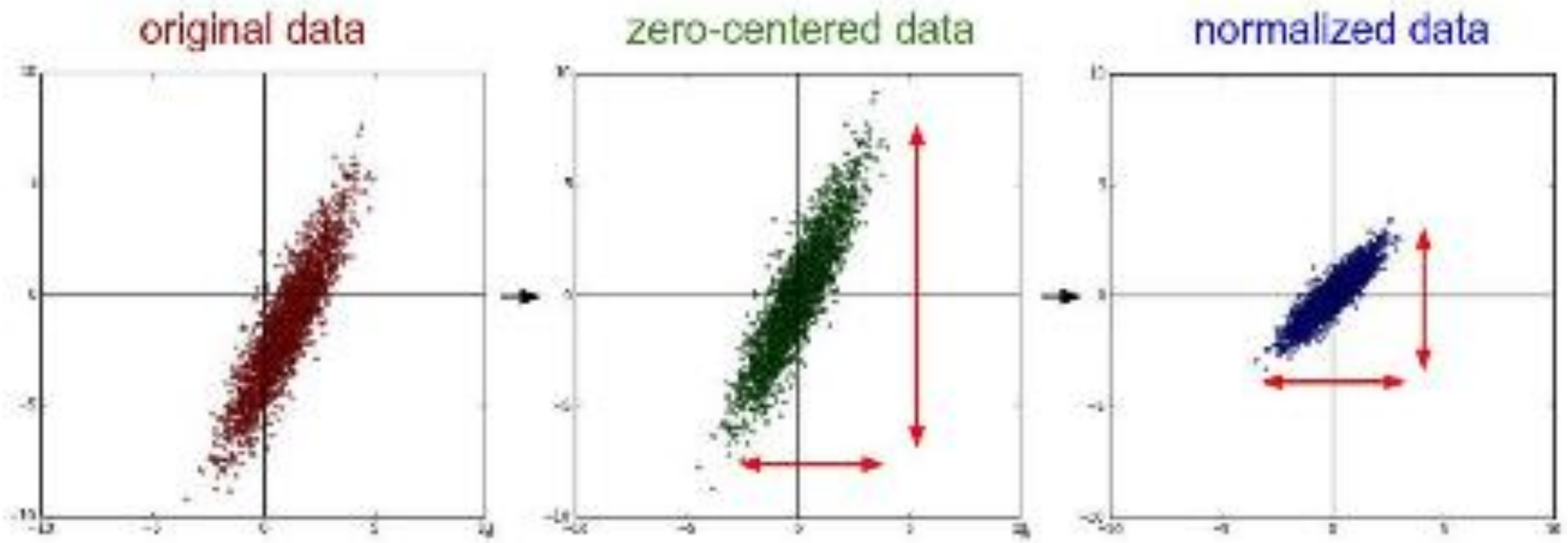
data set 요소들 간의 크기의 차이가 큰 경우 샘플의 분포



x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C

2.1 Problems during deep learning

- 데이터셋 원소 분포의 종류와 학습
- 데이터셋의 원소 분포



2. Data preprocessing for a gradient descent(cont.)

- 2.3 Mean, std normalization
  - $\boldsymbol{X' = \frac{X - X.mean}{X.std}}$

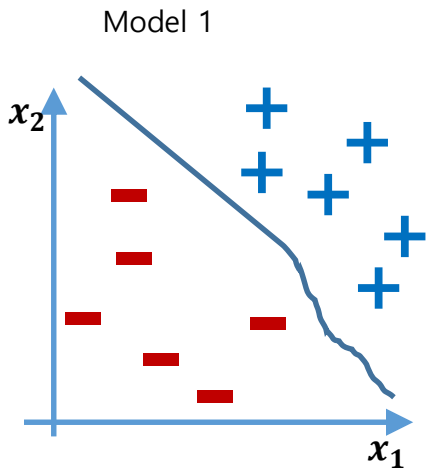
- 2.4 Min-max normalization
  - $\boldsymbol{X' = \frac{X - X.min}{X.max - X.min}}$

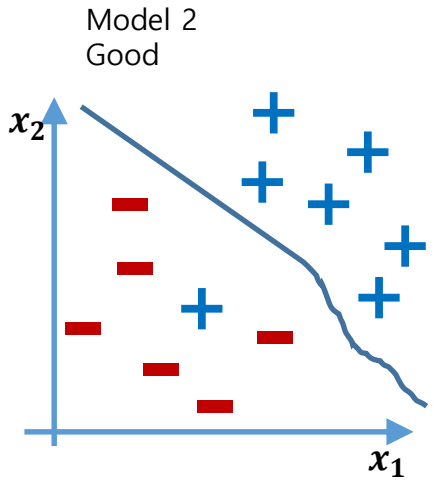
3.Overfitting

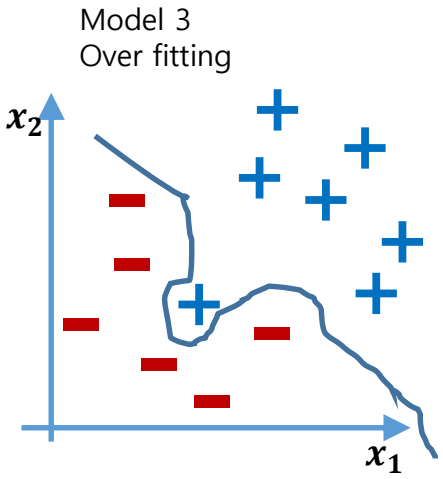
3.1 Overfitting의 개요

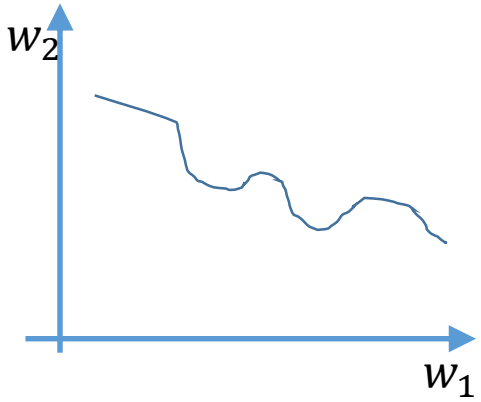
- 모든 훈련데이터 샘플을 정확하게 학습시키면 특이 데이터에 대해서도 학습이 되어 모델이 전체 데이터에 대하여 오히려 정확도를 낮아지게 만들어 지는 현상이다.
- 모델을 반복 훈련하는 과정에서
  학습초기에는 훈련데이터 및 평가데이터에 대하여 학습이 잘되지만
  어느정도 반복 훈련한 후에 훈련데이터에 대해서는 학습이 계속 잘되지만 평가데이터에 대해서는 학습이 더 이상 안되는 현상을 의미한다.
- 과대적합(overfitting)에 대한 해결방안
  - More training data Or Reduce the number of features
  - **Regularization(일반화) K1,K2**
  - **Dropout**

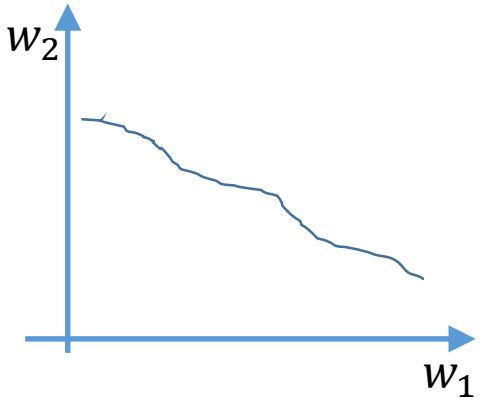
3.Overfitting











Overfitting 구부러짐 W 특정 값이 너무 크다

편다 구부러짐을 편다 W가 적은 값을 갖게 만든다.

3.Overfitting

3.1 More training data

- training data를 많이 모으는 것이다. 데이터가 많으면 training set, validation set, test set으로 나누어서 진행할 수 있고, 데이터의 크기가 커지면 소수의 특이데이터의 비율은 상대적으로 작아지고 전체 정확도는 개선된다고 볼 수도 있다.

3.2 Reduce the number of features

- feature의 수를 줄이면 모델의 복잡도가 줄어들어 소수의 특이데이터의 학습은 어려워진다. 그러므로 과적합을 방지할 수 있지만 다른 복잡한 문제를 해결할수 없을 수도 있다.
- 서로 비중이 다른 feature가 섞여서 weight에 대해 경합을 하면 오히려 좋지 않은 결과가 나올 수도 있으므로 feature 개수를 줄이는 방법을 선택할 수도 있지만 상기와 같이 단점이 있으므로 다른 방법(dropout, regularization)으로 해결하는 것이 바람직하다.

3.Overfitting

3.3 Regularization(일반화)

- 가중치(weight)의 수치가 너무 커지지 않게 하기 위하여 손실함수에 제어 항을 추가한다.
- regularization에서 가장 많이 사용하는 l2reg(엘투 regularization)이라고 부르는 방법을 보여준다. 앞에 있는 람다(λ)의 값을 보통 0.001로 주는데, 중요하게 생각한다면 0.01도 가능하다.

$$logit(X) = f(w)$$
$$loss(X,Y) = D(logit(X),Y)$$
$$loss(X,Y) = D(logit(X),Y) + \lambda \sum w_i^2$$
$$W^* = \underset{W}{ArgMin} \frac{\partial}{\partial w} loss(W|X,Y)$$

Regularization strength

W becomes small value

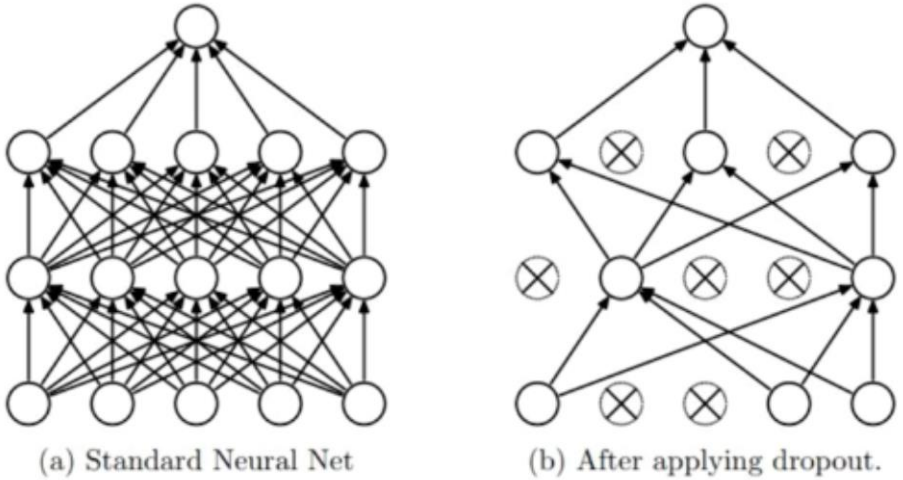
λ:0
λ:1
λ:0.001

```
fromkeras.regularizersimportl1_l2
reg = l1_l2(l1=0.01, l2=0.01)
model.add(Dense(1,input_dim=x.shape[1],W_regularizer=reg))
```

3.Overfitting

3.4 Dropout

- dropout을 NN의 일부 셀을 weight 값을 제어하여 제거하는 방법이다. 즉 전체 weight을 계산에 참여시키는 것이 아니라 layer에 포함된 weight 중에서 일부만 참여시키는 것이다.

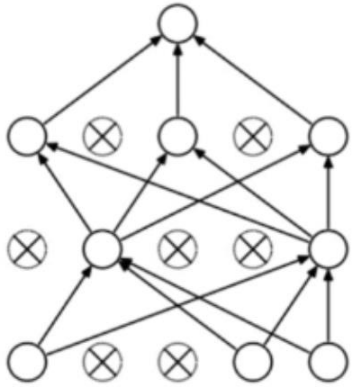


(a) Standard Neural Net

(b) After applying dropout.

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 6

Waaaait a second…
How could this possibly be a good idea?



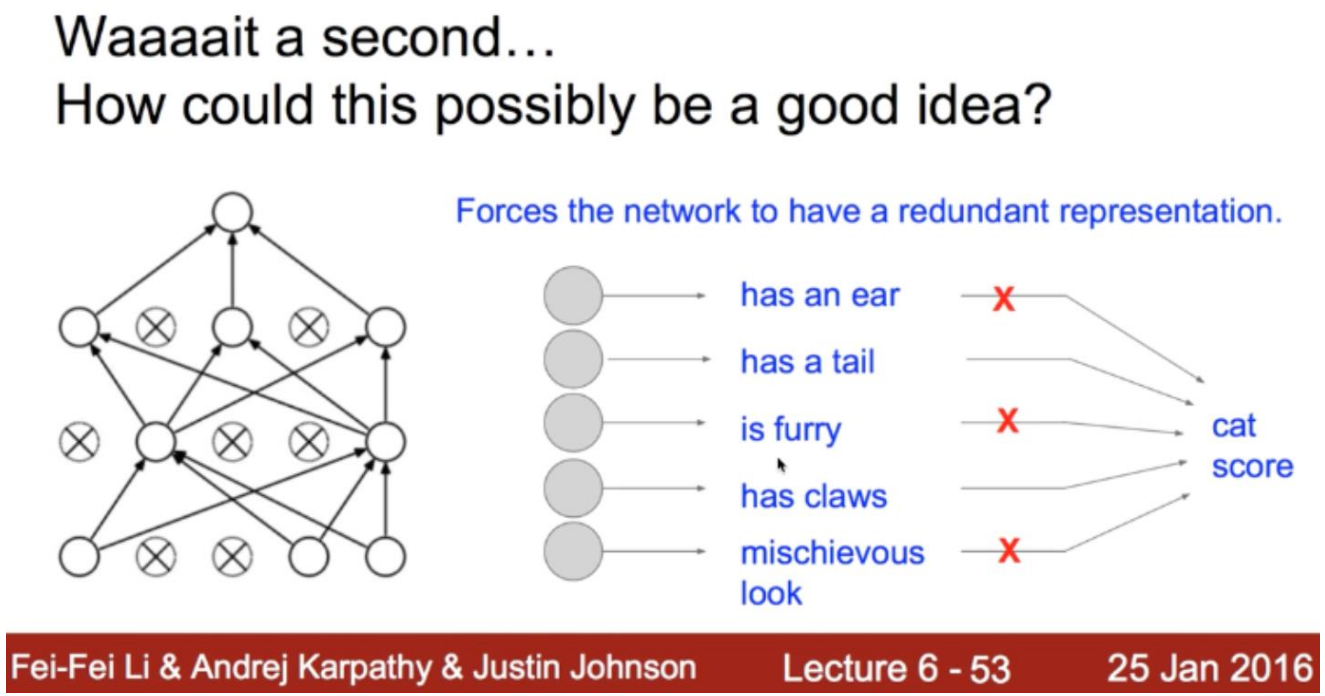
Forces the network to have a redundant representation.

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 6 - 53 25 Jan 2016

3.Overfitting

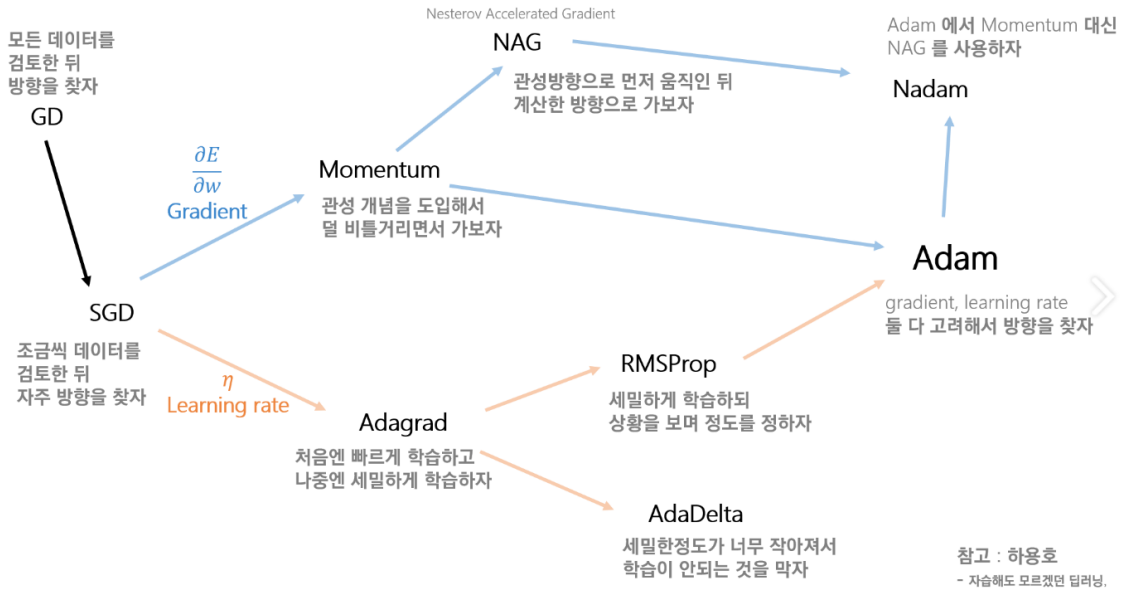
3.4 Dropout

- dropout을 NN의 일부 셀을 weight 값을 제어하여 제거하는 방법이다. 즉 전체 weight을 계산에 참여시키는 것이 아니라 layer에 포함된 weight 중에서 일부만 참여시키는 것이다.
- In Tensorflow
  - _l1 = tf.layers.dense(
    units=256,
    inputs=X,
    activation=tf.nn.relu)
  - L1=tf.nn.dropout(_l1, dropout_rate)
- in Keras
  - model = Sequential()
  - model.add(
    Dense(
    units=256, input_dim=784,
    activation='relu' ,
    activity_regularizer=reg))
  - model.add(Dropout(0.2))



4. Optimizers

- Optimizers [link] [linlk]
  - Momentum 방식은
    - 말 그대로 Gradient Descent를 통해 이동하는 과정에 일종의 ‘관성’을 주는 것이다. 현재 Gradient를 통해 이동하는 방향과는 별개로, 과거에 이동했던 방식을 기억하면서 그 방향으로 일정 정도를 추가적으로 이동하는 방식이다.
  - Adagrad(Adaptive Gradient)는
    - 변수들을 update할 때 각각의 변수마다 step size를 다르게 설정해서 이동하는 방식이다. 이 알고리즘의 기본적인 아이디어는 ‘지금까지 많이 변화하지 않은 변수들은 step size를 크게 하고, 지금까지 많이 변화했던 변수들은 step size를 작게 하자’ 라는 것이다.
  - RMSProp은
    - 딥러닝의 대가 제프리 힌톤이 제안한 방법으로서, Adagrad의 단점을 해결하기 위한 방법이다. Adagrad의 식에서 gradient의 제곱값을 더해나가면서 구한 Gt 부분을 합이 아니라 지수평균으로 바꾸어서 대체한 방법이다.
  - **Adam** (Adaptive Moment Estimation)은
    - RMSProp과 Momentum 방식을 합친 것 같은 알고리즘이다



4. Optimizers

- 최적화기(optimizer)의 사용 예

```
from keras import optimizers
model = Sequential()
model.add(Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(Activation('softmax'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

- 모든 Keras 옵티 마이저에 공통적 인 매개 변수 제어방법
  - 모든 최적화기에서 공통으로 사용되는 clipnorm 및 clipvalue 매개 변수는 그라디언트 클리핑을 제어 할 수 있습니다.

```
from keras import optimizers
# All parameter gradients will be clipped to a maximum norm of 1.
sgd = optimizers.SGD(lr=0.01, clipnorm=1.)
```

```
from keras import optimizers
# All parameter gradients will be clipped to  a maximum value of 0.5 and  a minimum value of -0.5.
sgd = optimizers.SGD(lr=0.01, clipvalue=0.5)
```

4. Optimizers (cont.)

- Stochastic gradient descent optimizer.

```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

- RMSProp optimizer

```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)
```

- Adagrad optimizer

```
keras.optimizers.Adagrad(lr=0.01, epsilon=None, decay=0.0)
```

- Adadelta optimizer.

```
keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)
```

- Adam optimizer

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

https://keras.io/optimizers/

4. Optimizers (cont.)

- Use Optimizer

```
from keras import optimizers

model.add(Dense(units=256, input_dim=784,
kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=10, kernel_initializer='glorot_uniform', activation='softmax'))

adam = optimizers.Adam(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='softmax', optimizer=adam)
```

5. Batch Normalization

- Batch Normalization의 장점
  - '기존 Deep Network에서는 learning rate를 너무 높게 잡을 경우 gradient가 explode/vanish 하거나, 나쁜 local minima에 빠지는 문제가 있었다. 이는 parameter들의 scale 때문인데, Batch Normalization을 사용할 경우 propagation 할 때 parameter의 scale에 영향을 받지 않게 된다. 따라서, learning rate를 크게 잡을 수 있게 되고 이는 빠른 학습을 가능케 한다.
  - Batch Normalization의 경우 자체적인 regularization 효과가 있다. 이는 기존에 사용하던 weight regularization term 등을 제외할 수 있게 하며, 나아가 Dropout을 제외할 수 있게 한다 (Dropout의 효과와 Batch Normalization의 효과가 같기 때문.) . Dropout의 경우 효과는 좋지만 학습 속도가 다소 늘여진다는 단점이 있는데, 이를 제거함으로써 학습 속도도 향상된다.
- 참고자료

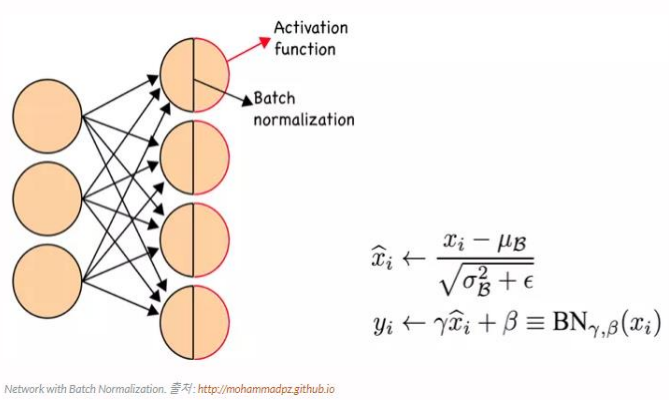
https://shuuki4.wordpress.com/2016/01/13/batch-normalization-%EC%84%A4%EB%AA%85-%EB%B0%8F-%EA%B5%AC%ED%98%84/
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-6-mnist_nn_batchnorm.ipynb
https://arxiv.org/abs/1502.03167

5. Batch Normalization (cont.)

- **‘10 Deep Learning Trends’** at NIPS 2015 [link]
  - ‘Brad Neuberg’, NIPS (Neural Information Processing Systems) ,Rusia, 12,2015.
    1. Neural network architectures are getting more complex and sophisticated
    2. All the cool kids are using LSTMs
    3. Attention models are showing up
    4. Neural Turing Machines remain interesting but aren't being leveraged yet for real work
    5. Computer vision and NLP aren't separate silos anymore — deep learning for computer vision and NLP are cross-hybridizing e ach other
    6. Symbolic differentiation is becoming even more important
    7. Surprising results are happening with neural network model compression
    8. **The intersection of deep and reinforcement learning continues**
    9. If you aren't using batch normalization you should
       - **Batch normalization is now considered a standard part of the neural network toolkit and was referenced throughout work at the conference.**
    10. Neural network research and productionisation go hand in hand
- Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariance Shift)
  - ICML 2015 [link]

5. Batch Normalization (cont.)

- Batch Normalization
  - Batch Normalization은 **기본적으로 Gradient Vanishing / Gradient Exploding 이 일어나지 않도록** 하는 아이디어 중의 하나이다. 지금까지는 이 문제를 Activation 함수의 변화 (ReLU 등), Careful Initialization, small learning rate 등으로 해결하였지만, 이 논문에서는 이러한 간접적인 방법보다는 training 하는 과정 자체를 전체적으로 안정화 하여 학습 속도를 가속시킬 수 있는 근본적인 방법을 찾고 싶어 했다.
  - 이들은 이러한 불안정화가 일어나는 이유가 'Internal Covariance Shift' 라고 주장하고 있다. Internal Covariance Shift라는 현상은 Network의 각 층이나 Activation 마다 input의 distribution이 달라지는 현상을 의미한다. 이 현상을 막기 위해서 간단하게 각 층의 input의 distribution을 평균 0, 표준편차 1인 input으로 normalize 시키는 방법을 생각해볼 수 있고, 이는 whitening이라는 방법으로 해결할 수 있다. Whitening은 기본적으로 들어오는 input의 feature들을 uncorrelated 하게 만들어주고, 각각의 variance를 1로 만들어주는 작업이다.



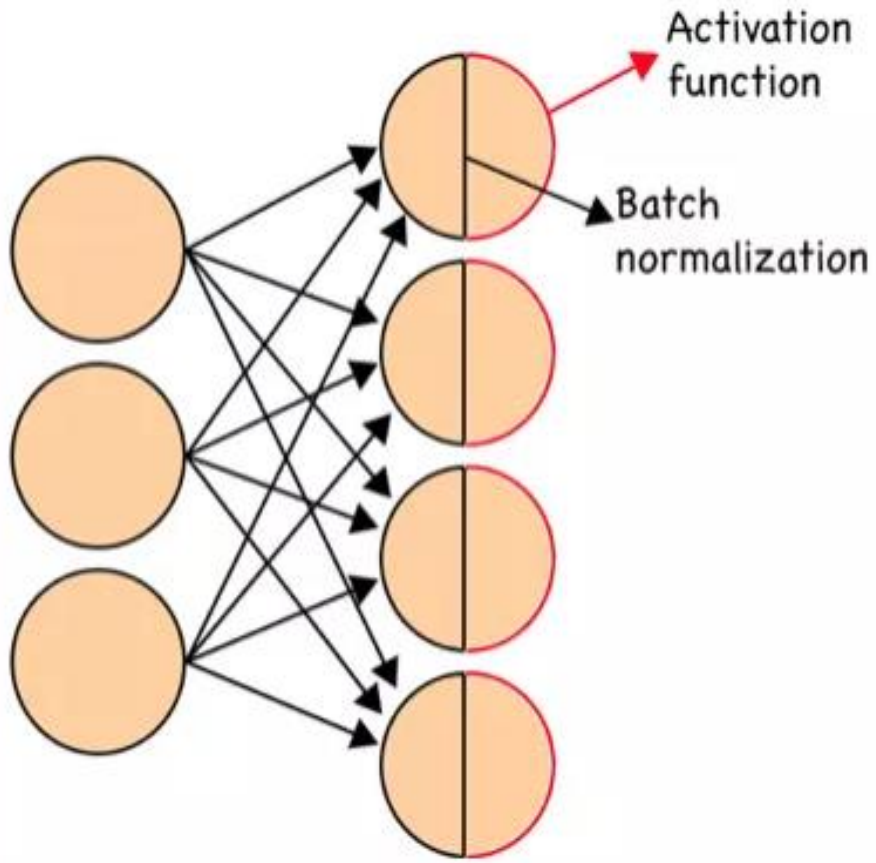
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

Network with Batch Normalization. 출처: http://mohammadpz.github.io

5. Batch Normalization (cont.)

- Batch Normalization
  - Batch Normalization은
    Exploding 이 일어나
    는 이 문제를 Activatic
    small learning rate 등으
    인 방법보다는 training
    속도를 가속시킬 수 있
  - 이들은 이러한 불안정
    라고 주장하고 있다. I
    층이나 Activation 마다
    이 현상을 막기 위해서
    표준편차 1인 input으로
    는 whitening이라는 방
    들어오는 input의 featu
    variance를 1로 만들어



Network with Batch Normalization. 출처: http://mohammadpz.github.io

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

5. Batch Normalization (cont.)

Batch Normalization Example[link]

- MLP Batch Normalization

```
from keras.layers import Dense
from keras.layers import BatchNormalization
...
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1))
```

- RNN Batch Normalization

```
# example of batch normalization for a lstm
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import BatchNormalization
...
model.add(LSTM(32))
model.add(BatchNormalization())
model.add(Dense(1))
...
```

- CNN Batch Normalization

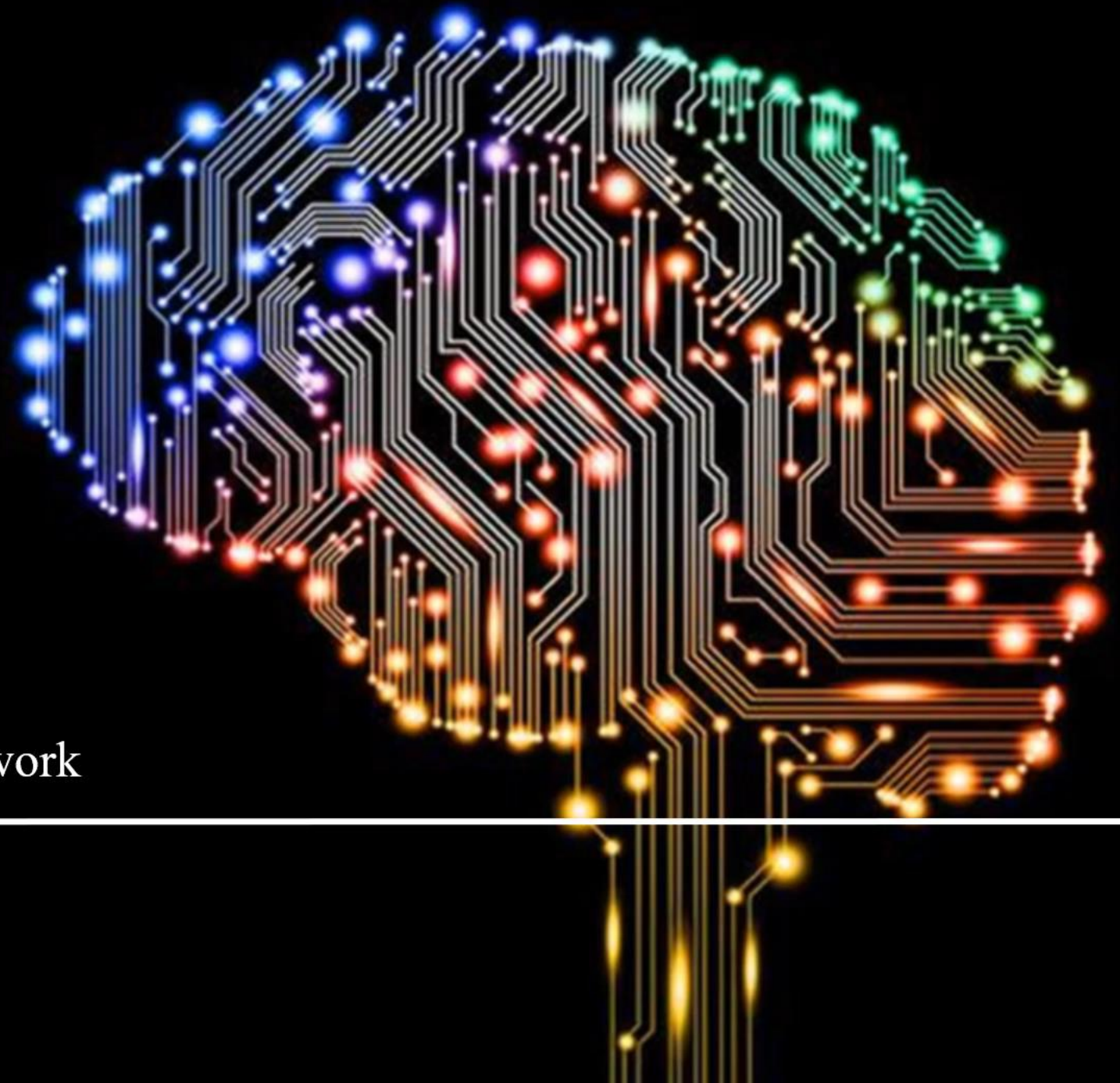
```
# example of batch normalization for an cnn
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import BatchNormalization
...
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D())
model.add(Dense(1))
...
```

Summary

1. Learning rate
    1. Gradient descent algorithm
    2. Large rate
    3. Small rate
2. Data preprocessing for the gradient decent algorithm
    1. Mean, std normalization
    2. Min-max normalization
3. Overfitting
    1. More training data or Reduce the number of features
    2. Regularization
    3. Dropout
4. 최적화기(Optimizer)의 종류
5. Batch Normalization
6. Examples
    1. Learning rate
    2. Dataset normalization
    3. Mnist digit classifier
        1. Mnist 이미지 데이터 분석
        2. Model 개발 및 평가
        3. 이미지 인식 및 출력방법
    4. Application Tips for the Mnist digit classifier

Next

1. Learning rate
   1. Gradient descent algorithm
   2. Large rate
   3. Small rate
2. Data preprocessing for the gradient decent algorithm
   1. Mean, std normalization
   2. Min-max normalization
3. Overfitting
   1. More training data or Reduce the number of features
   2. Regularization
   3. Dropout
4. 최적화기(Optimizer)의 종류
5. Batch Normalization
6. Examples
   1. Learning rate
   2. Dataset normalization
   3. Mnist digit classifier
      1. Mnist 이미지 데이터 분석
      2. Model 개발 및 평가
      3. 이미지 인식 및 출력방법
   4. Application Tips for the Mnist digit classifier



Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University