

13기 정규세션

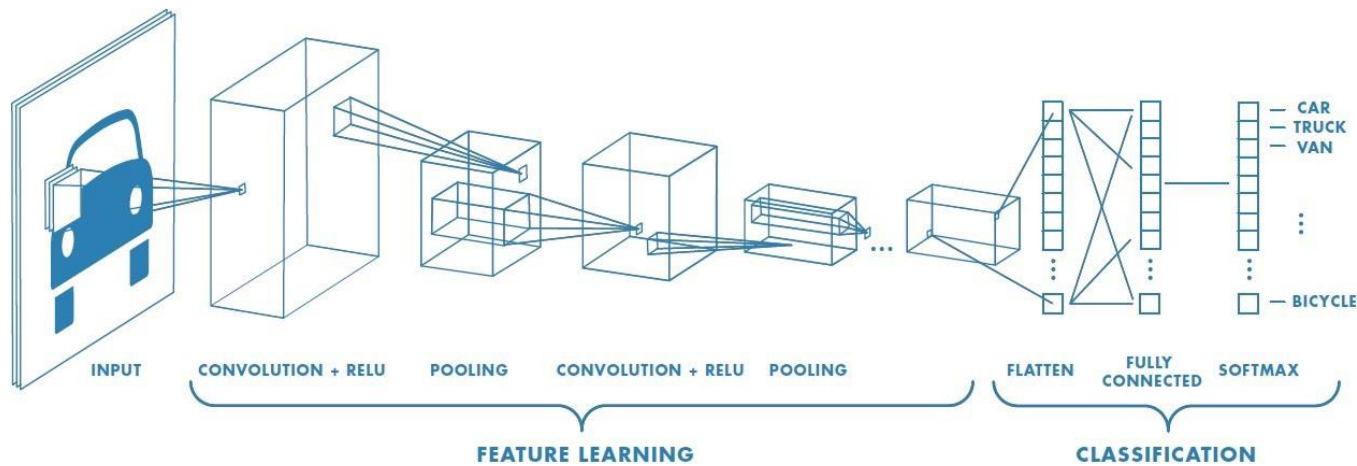
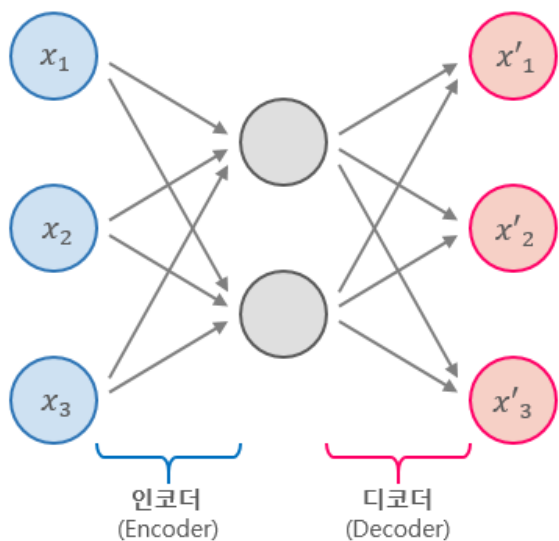
ToBig's 12기 이승현

딱딱한

Neural Network 심화

Unit 00 | 목적

어떻게 좋은 모델을 설계할까요? – 부품들에 대하여 알아보시다.



Contents

Unit 01 | Activation Function

Unit 02 | Weight Initialization

Unit 03 | Batchnormalization

Unit 04 | Optimization

Unit 05 | Regularization

Contents

Unit 01 | Activation Function

Unit 02 | Weight Initialization

Unit 03 | Batchnormalization

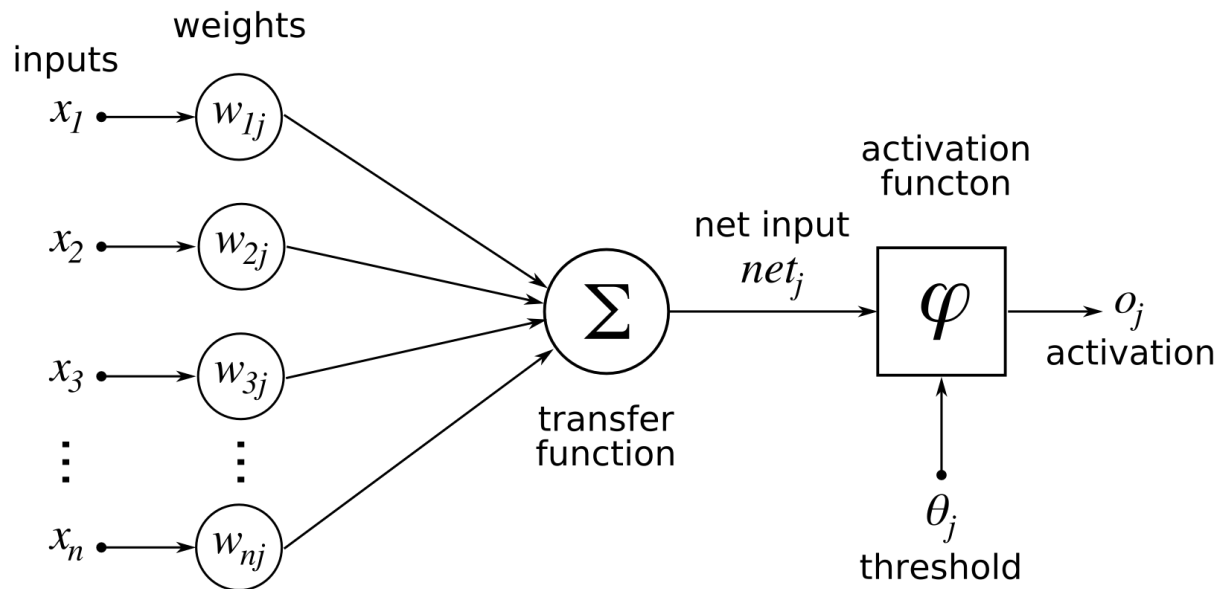
Unit 04 | Optimization

Unit 05 | Regularization

Unit 01 | Activation Function

Activation Function이란?

활성화 함수라 불리며, 신경망의 출력을 결정하는 식입니다.



Comment 1.

만약 Sigmoid 가 출력 함수이고
단일 Perceptron이라면 이 문제는 정확히
Logistic Regression 와 일치합니다!

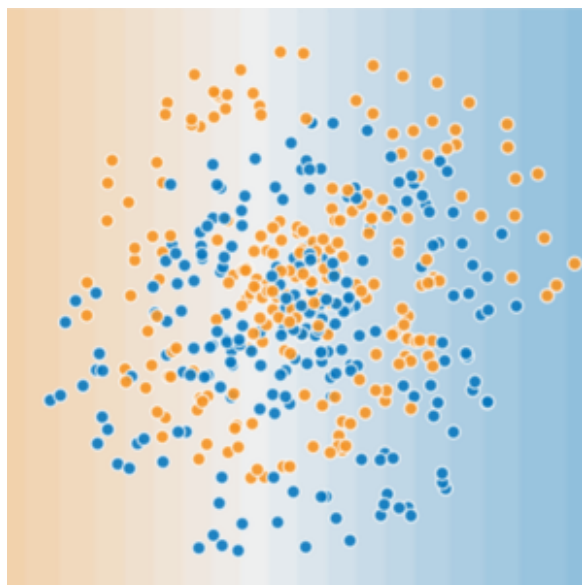
Question 1.

Activation은 딥러닝에서 어떤 역할을 할까요?

Unit 01 | Activation Function

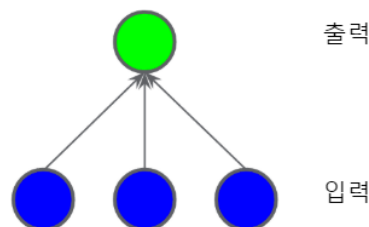
Activation Function은 비선형 분류 문제를 해결하기 위해서 필요합니다

‘비선형’이란 $b + w_1x_1 + w_2x_2$ 형태의 모델로 라벨을 정확하게 예측할 수 없다는 의미입니다.

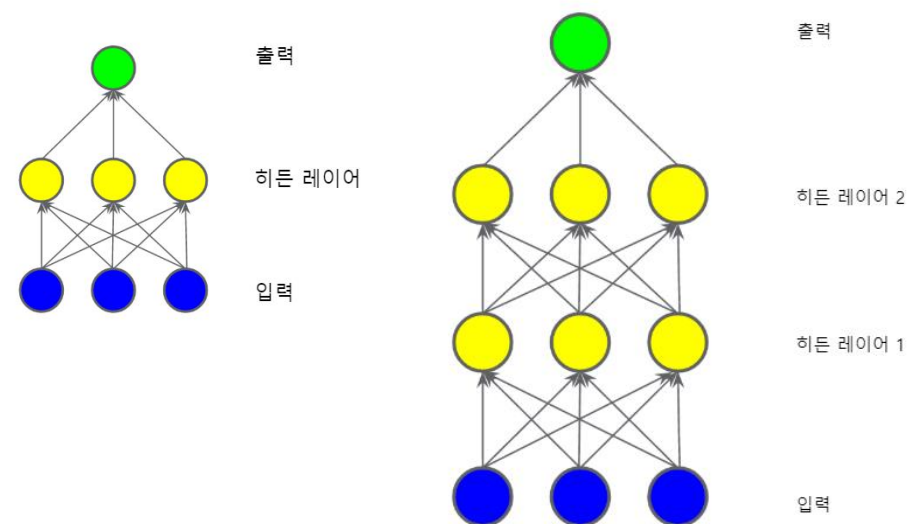


선으로 분류할 수 없습니다.
즉, 기존 선형모델로 해결할 수 없습니다.

기존 선형 모델



히든레이어 추가

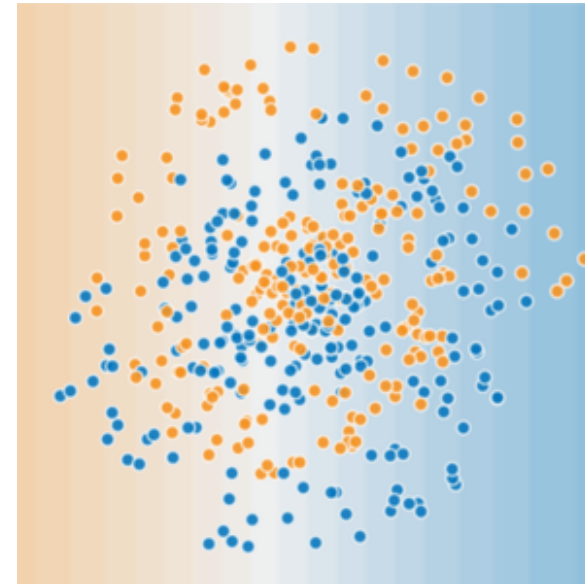
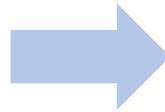
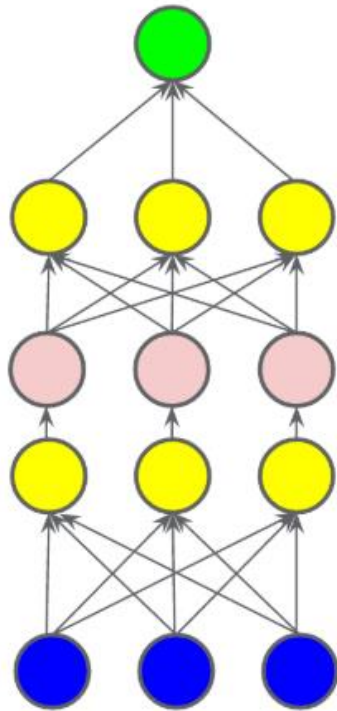


Comment2.
단순히 히든 레이어를 추가한다고 비선형성을 획득할 수 없습니다.

Unit 01 | Activation Function

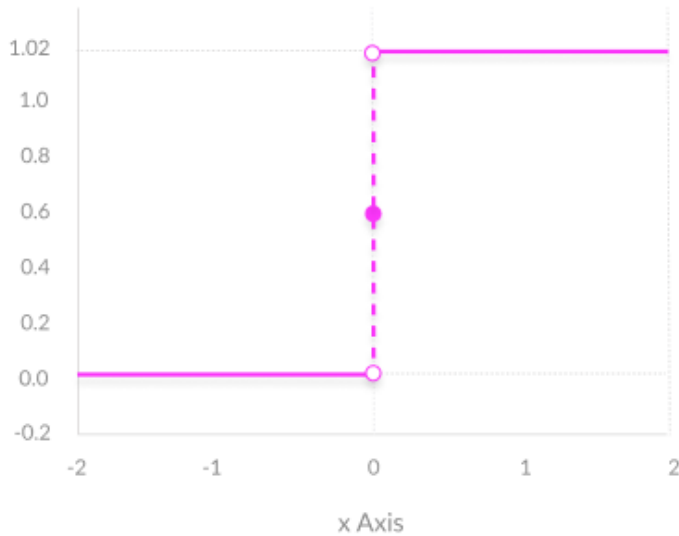
비선형성을 누적해서 복잡한 입출력 관계를 모델링할 수 있습니다.

각 히든 레이어가 노드의 비선형 함수를 통과하도록 하여서 비선형 문제를 모델링할 수 있습니다.



Unit 01 | Activation Function

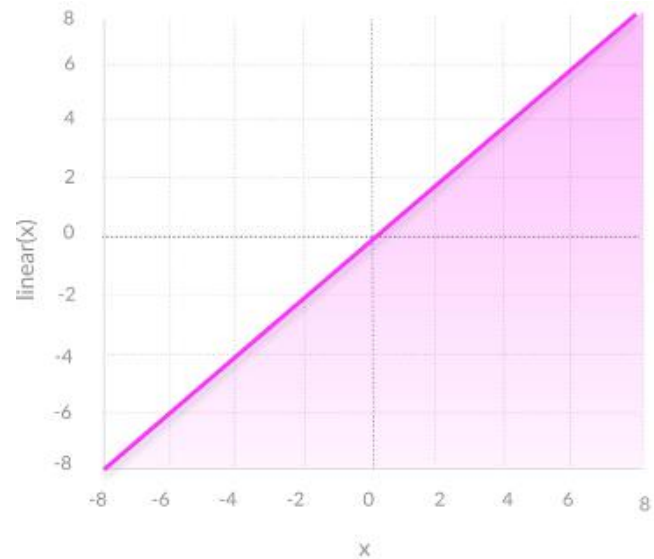
Activation Function의 종류



Binary Step Function

$$\sigma(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

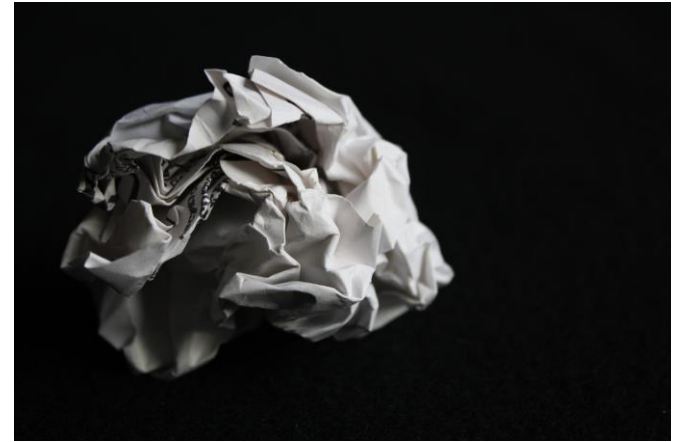
다중 출력이 불가능합니다.



Linear Activation Function

$$h(x) = cx, c \text{ is constant}$$

BackpropaGation이 의미가 없습니다. $h(x) = c$
은닉층을 무시합니다. $\rightarrow h(h(h(x))) = kx$

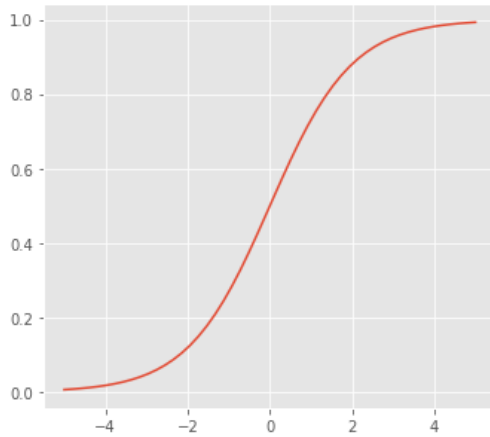


Nonlinear Activation Function

BackpropaGation 가능합니다.(미분 가능)
비정형적인 데이터에서 큰 힘을 발휘합니다.
(이미지, 영상, 음성)

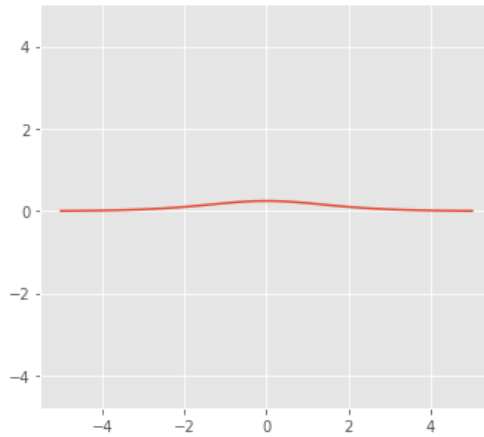
Unit 01 | Activation Function

Nonlinear Activation Function1. Sigmoid – Binary Classification



함수의 식

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



미분 값

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

장 점

- 유연한 미분값을 가집니다. 역전파시 그래디언트가 너무 커지는 문제
- 출력값의 범위가 (0, 1) 로 제한되어 exploding gradient 문제를 방지합니다.
- 미분 식이 단순한 형태를 가집니다.

단 점

- Vanishing Gradient 문제가 발생합니다. 산술 기하 평균에 의해

$$\frac{\sigma(x) + (1 - \sigma(x))}{2} = \frac{1}{2} \geq \sqrt{\sigma(x)(1 - \sigma(x))}$$

- 미분값의 범위는 최대 ¼이고 층이 쌓일수록 gradient가 0으로 수렴합니다.
- 출력의 중심이 0이 아닙니다.

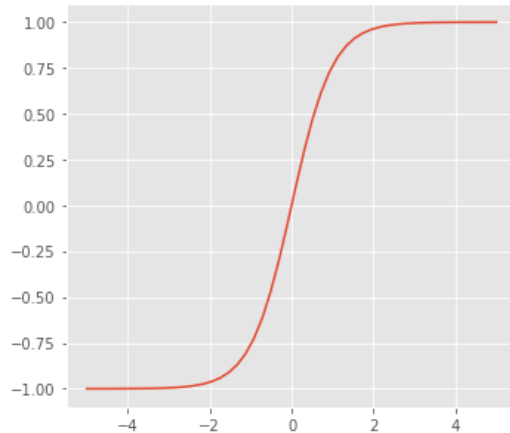


→ X가 모두 양수일 때 gradient가 모두 양수 혹은 음수의 형태를 지니어 효율적이지 못합니다.

- exp 연산은 비용이 큼니다.

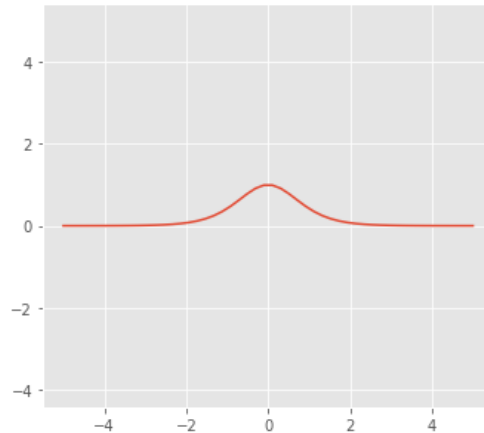
Unit 01 | Activation Function

Nonlinear Activation Function2. Tanh



함수의 식

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



미분 값

$$\tanh'(x) = 1 - \tanh^2(x)$$

장 점

- Zero Centered 함수입니다.
- 그 이외에는 Sigmoid와 장점이 같습니다.

단 점

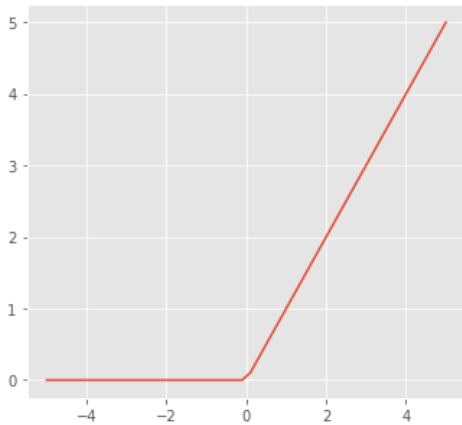
- **Vanishing Gradient** 문제가 발생합니다. 산술 기하 평균에 의해

$$\frac{(1 - \tanh(x)) + (1 + \tanh(x))}{2} = 1 \geq \sqrt{(1 + \tanh(x))(1 - \tanh(x))}$$

- 미분값의 범위는 (0, 1)이고 층이 쌓일수록 gradient가 0으로 수렴합니다.

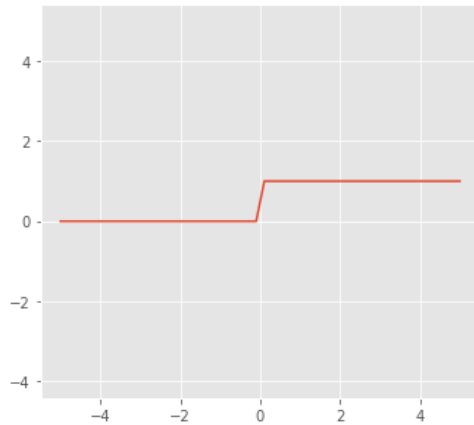
Unit 01 | Activation Function

Nonlinear Activation Function3. ReLU – Rectified Linear Unit



함수의 식

$$f(x) = \max(0, x)$$



미분 값

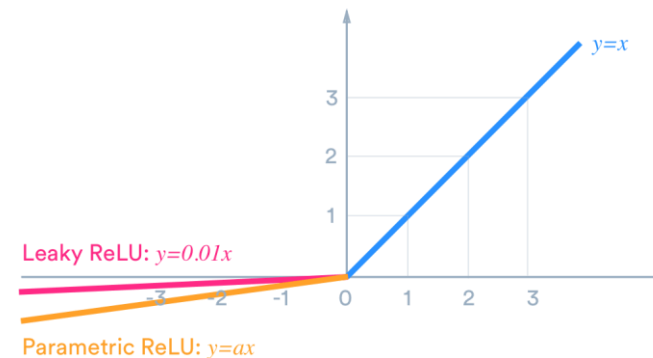
$$f'(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

장 점

- CNN에서 기가 막힌 성능을 보였습니다.
- 현재 딥러닝에서 가장 많이 사용하는 활성화 함수입니다.
- 일부 정보에 대한 무시와 수용을 이루어 냅니다.
- 연산이 매우 빠릅니다.
- 비선형입니다.

단 점

- Dying ReLU : 입력값이 0 or 음수일 때 아예 학습을 하지 못합니다.
- 이 문제를 해결하기 위해 Leaky ReLU가 나왔습니다.

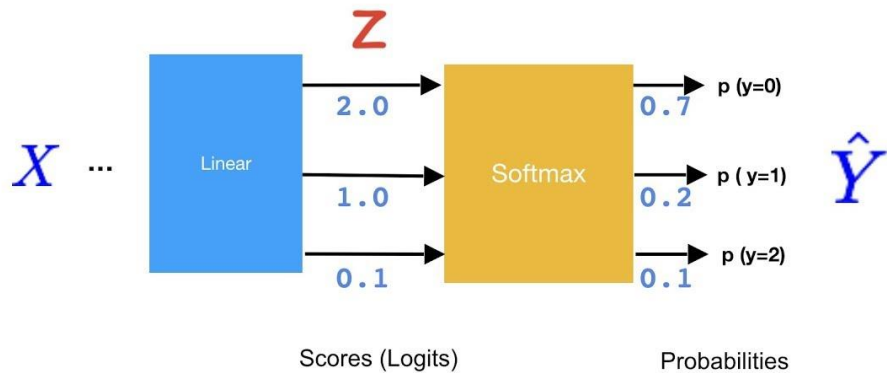


Unit 01 | Activation Function

Nonlinear Activation Function4. Softmax

Meet Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K.$$



장점

- 다중 클래스 문제 적용가능합니다.
- 출력층에 자주 사용합니다.
- 정규화 기능을 가집니다.

단점

- 지수함수이기 때문에 오버플로 발생이 가능합니다.

Unit 01 | Activation Function

Nonlinear Activation Function4. Softmax Backpropagation

a_1 p_1 개 : 0.6 y_1 1.0

a_2 p_2 고양이 : 0.2 y_2 0.0

a_3 p_3 닭 : 0.1 y_3 0.0

a_4 p_4 담배 : 0.1 y_4 0.0

Layer Output Label

a_i : 가중치 p_i : 각 클래스에 속할 확률 y_i : 각 클래스 실제 라벨

$$p_i = \frac{\exp(a_i)}{\sum_k \exp(a_k)} \quad L = - \sum_j y_j \log p_j \quad \text{Loss : Cross Entropy}$$

(p, y 의 확률분포 거리계산)

Quiz : 우리가 뭘 구해야하죠? $a_{i,update} = a_i - \text{stepsize} \times \frac{\partial L}{\partial a_i}$

$$\frac{\partial L}{\partial a_i} = \frac{\partial (-\sum_j y_j \log p_j)}{\partial a_i} = - \sum_j y_j \frac{\partial \log p_j}{\partial a_i} = - \sum_j y_j \frac{1}{p_j} \frac{\partial p_j}{\partial a_i}$$

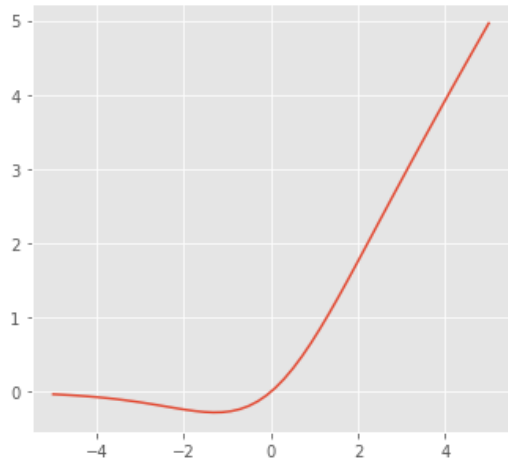
$$= - \frac{y_i}{p_i} p_i (1 - p_i) - \sum_{i \neq j} \frac{y_j}{p_j} (-p_i p_j) = -y_i + y_i p_i + \sum_{i \neq j} y_j p_i$$

$i = j$ 일 때 $i \neq j$ 일 때

$$= -y_i + p_i \sum_j y_j = p_i - y_i \quad a_{i,update} = a_i - \text{stepsize} \times (p_i - y_i)$$

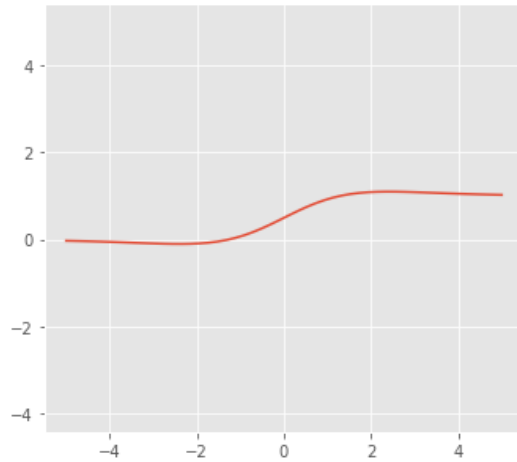
Unit 01 | Activation Function

Nonlinear Activation Function 5. Swish – Sigmoid Linear Unit



함수의 식

$$f(x) = \frac{x}{1 + e^{-x}}$$



미분 값

$$f'(x) = f(x) + \sigma(x)(1 - f(x))$$

장 점

- 2차원에서 linear 혹은 ReLU보다 훨씬 부드러운 형태를 가집니다.
- ReLU/ 다른 활성화 함수보다 더 좋은 성능을 가집니다.

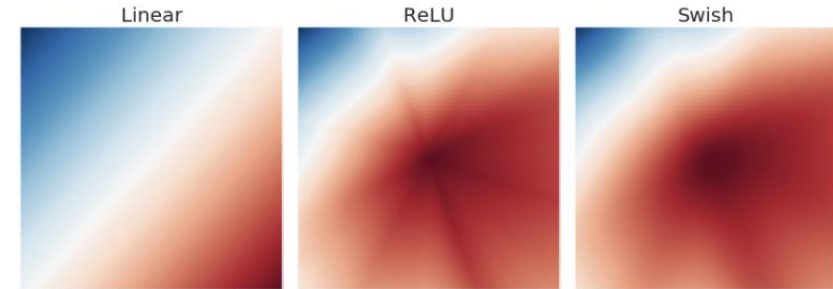
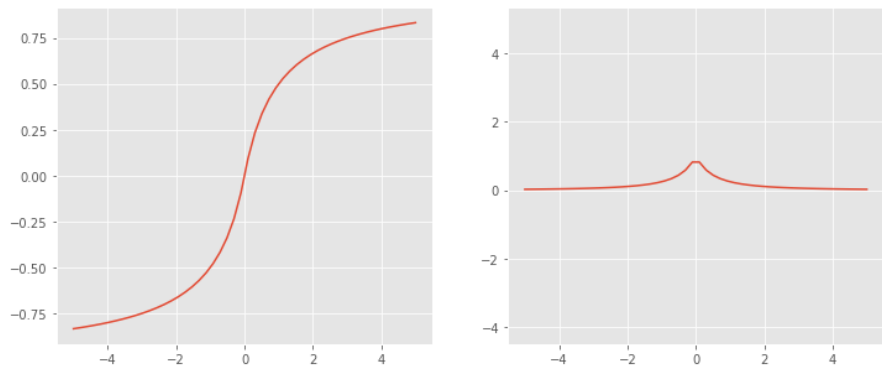
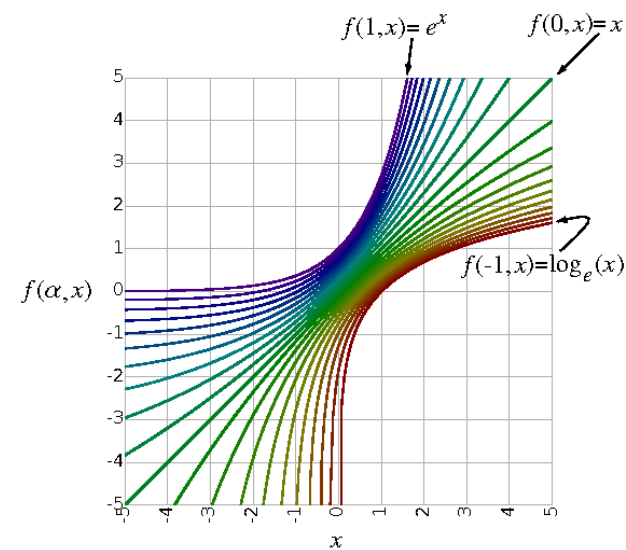


Figure 5: The output landscape of a random neural network, with different activation functions. The figure is generated by passing two scalars, the x and y coordinates of each position in a grid, into a randomly initialized neural network that outputs a single scalar. Linear refers to $f(x) = x$, i.e., no nonlinearity. The ReLU network output landscape has distinctive sharp regions of similar magnitudes whereas the the Swish network output landscape is more smooth. Best viewed in color.

Unit 01 | Activation Function

Nonlinear Activation Function 그 밖에 쓰는 함수

Soft Sign

Soft
ExponentialFigure 2: A plot of $f(\alpha, x)$ for $\alpha =$

Contents

Unit 01 | Activation Function

Unit 02 | Weight Initialization

Unit 03 | Batchnormalization

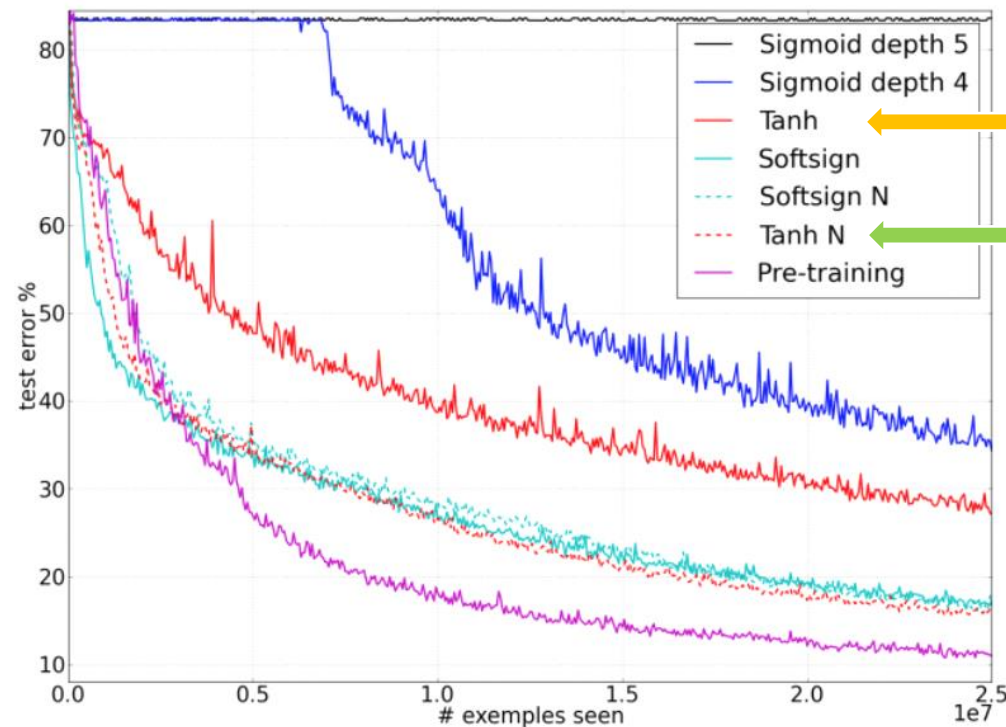
Unit 04 | Optimization

Unit 05 | Regularization

Unit 02 | Weight Initialization

Weight Initialization가 중요한 이유

가중치 초기화만 잘해도 Test Error를 효과적으로 줄일 수 있습니다!

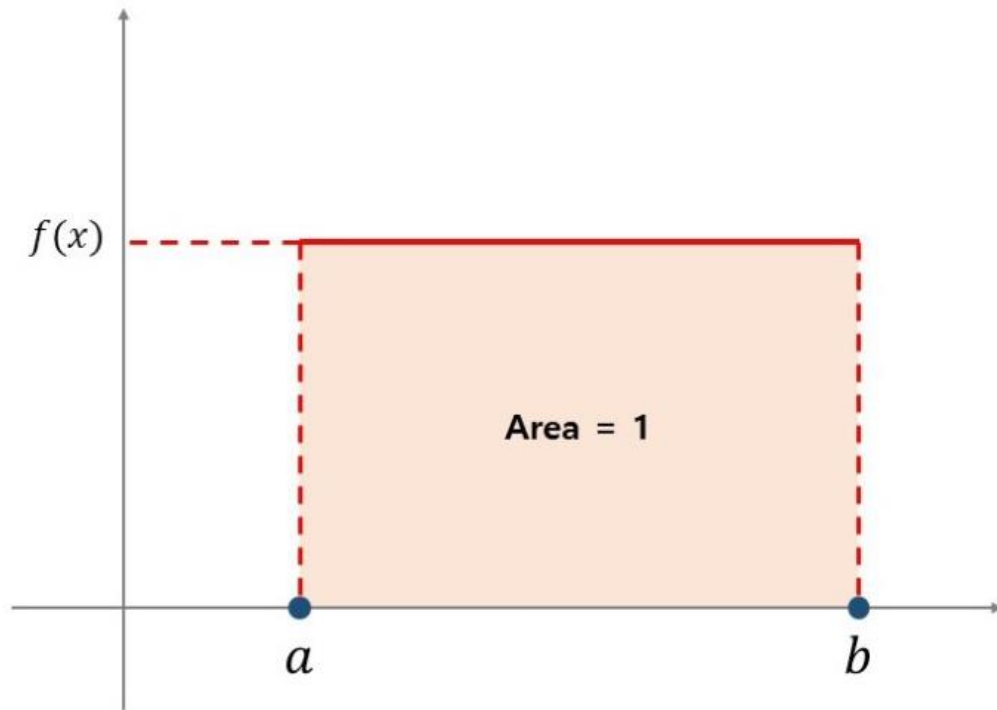


같은 Activation Function이어도
초기화 알고리즘에 따라 성능이 다릅니다!

Unit 02 | Weight Initialization

얇게 알아보는 확률분포 – 균일 분포(Uniform Distribution)

모든 확률변수에 대해 균일한 확률을 가지는 분포입니다.



$f(x) = p$ ($a \leq x \leq b$): p 는 확률입니다!

$$f(x) \times (b - a) = 1$$

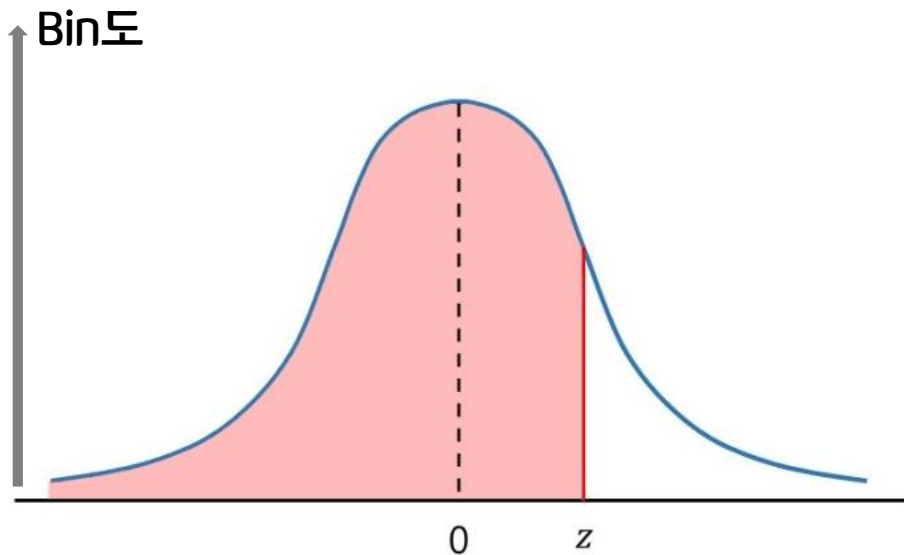
$$f(x) = \frac{1}{b - a}$$

문제) 투빅스 공인 롤 티어가 골드인 승현이가 원딜 카시딘으로 미니언 100마리를 먹는 시간은 15분에서 20분 사이라면 미니언 100마리를 먹는 시간이 18분일 확률은?

Unit 02 | Weight Initialization

얇게 알아보는 확률분포 – 표준 정규분포(Standard Normal Distribution)

다른 이름으로는 가우시안 분포로 평균에 데이터 포인트가 몰려있는 것이 특징입니다! (평균 0, 표준편차 1)



$$f(z; 0, 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \quad (-\infty \leq z \leq \infty): z \text{는 } x \text{에서 스케일링 된 값입니다!}$$

`sklearn.preprocessing.StandardScaler`

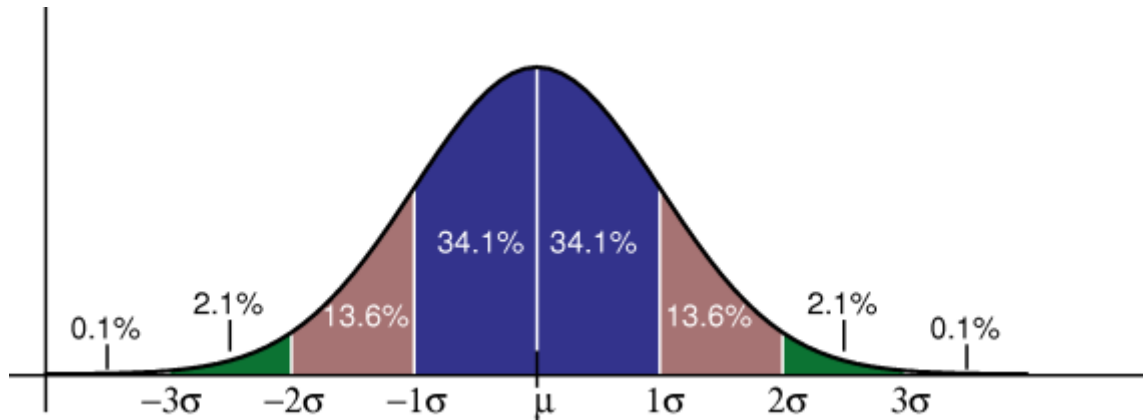
If 가중치가 표준 정규분포를 따른다면!

가중치값 w 는 z 값 중에 하나로 초기화가 되며 $(-\infty \leq z \leq \infty)$ 이 가중치들은 평균이 0, 분산이 1이다.

Unit 02 | Weight Initialization

Weight Initialization이란? – Neural Net 학습 전 가중치들을 초기화하는 과정

가중치 초기화를 통해 가중치가 급 Vanishing 혹은 Exploding 하는걸 막을 수 있습니다!



특정 분포를 따르는 무작위 난수 생성을 통해 가중치를 생성하고 초기화할 수 있습니다!

Question2.

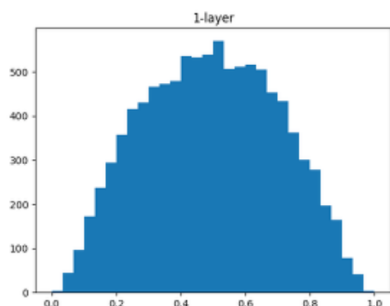
모든 가중치를 0으로 초기화하는 것은 어떨까요?

모든 뉴런은 같은 값을 나타내고, 역전파에서 각 가중치의 update가 동일해져 버립니다.

아! 그렇다면 표준 정규 분포로 무작위로 가중치를 생성하면 뭔가 좋은 초기화일 것 같은 느낌이 들지 않나요?
(Lecun Initialization, 1998)

Unit 02 | Weight Initialization

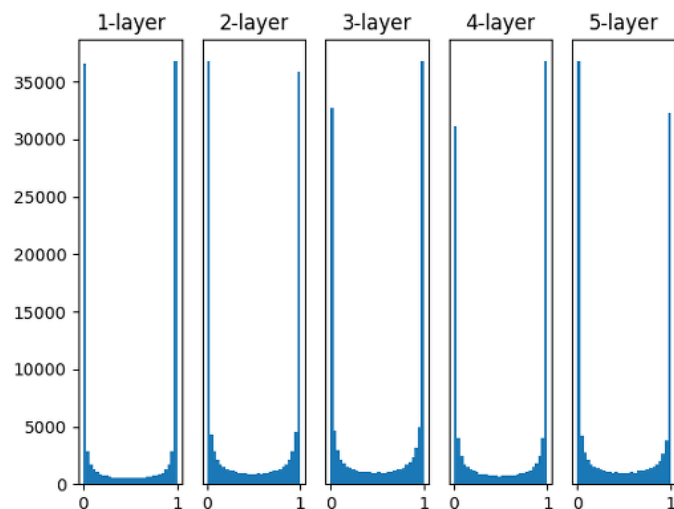
실제 이 분포로 난수를 생성했을 때는 폭발하고 소실하고 난리도 아닙니다.



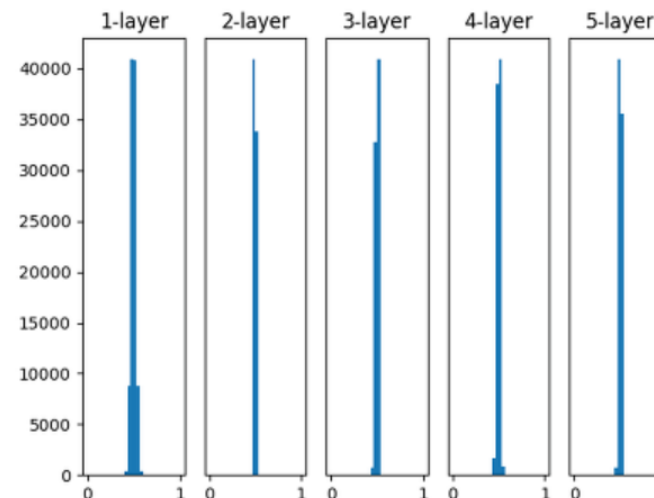
표준정규분포로 생성한
가중치 히스토그램

If Activation Function 이 Sigmoid일 때

- 1) 표준편차가 크다면? 학습을 반복할 수록 출력값이 0, 1로 치우칩니다.
- 2) 표준편차가 작다면? 층이 깊어질수록 출력값이 중간 값인 0.5로 몰립니다.



표준편차가 클 때



표준편차가 작을 때

Unit 02 | Weight Initialization

표준편차를 노드 개수에 맞게 Scaling 하자 (Xavier / Glorot Initialization, 2010)

이전 / 다음 노드의 개수에 의존하는 방법입니다. Uniform / Normal 분포를 따르는 2가지 방법이 있습니다.

Xavier Normal Initialization

$$W \sim N(0, \text{Var}(W))$$
$$\text{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

n_{in} : 이전 layer(input)의 노드 수
 n_{out} : 다음 layer의 노드 수

Xavier Uniform Initialization

$$W \sim U\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right)$$

Xavier 함수는 비선형 함수(Sigmoid, Tanh - **Vanishing**)에서 효과적인 결과를 보여줍니다.

하지만 ReLU에서 사용시 출력 값이 0으로 수렴하게 되는 현상을 확인할 수 있다.

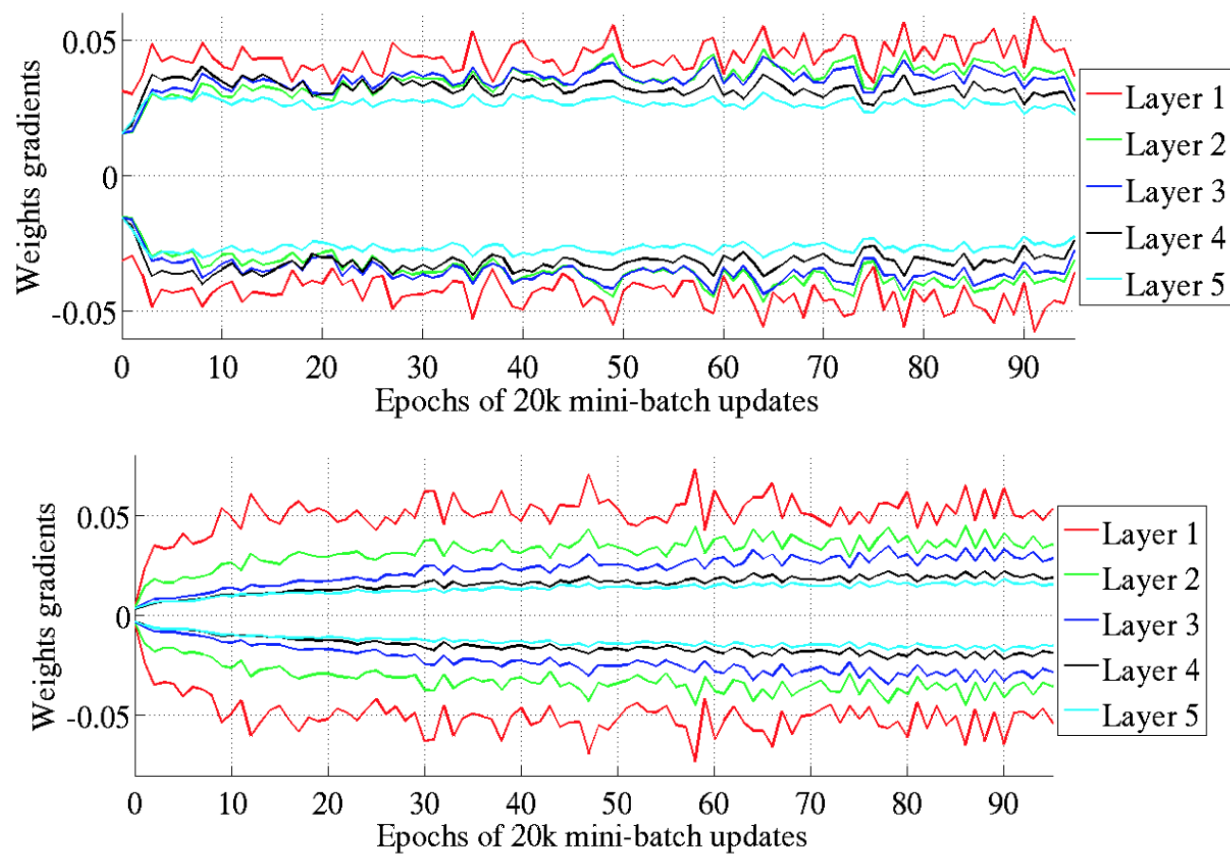
따라서 ReLU에서는 He initialization을 사용한다.

He Initialization 이란? Nout만 빼면 He initialization(2015)이다. Quiz. 왜 빼줄까요?

<https://review.github.io/13/>

Unit 02 | Weight Initialization

Xavier Initialization 의 우수성



Unit 02 | Weight Initialization

* 그 밖에 Initialization Technique

엄청난 데이터 기반으로 잘 학습된 ImageNet 가중치로 이미지 학습과정에서 재미를 많이 볼 수 있다.



Contents

Unit 01 | Activation Function

Unit 02 | Weight Initialization

Unit 03 | Batchnormalization

Unit 04 | Optimization

Unit 05 | Regularization

Unit 03 | BatchNormalization

Normalization이란?

우리가 아는 그 Feature Scaling이 맞습니다. 그럼 BatchNormalization은 뭘까?

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$


과정

1. Mini Batch를 뽑는다.
2. Mini Batch의 분산과 표준 편차를 구한다.
3. Normalization을 한다.
4. 스케일 조정 및 분포 조정을 한다.

장점

1. Propagation에서 파라미터 크기에 영향을 받지 않습니다.
2. 가중치 규제와 드롭아웃을 제외할 수 있어 학습 속도 UP!

단점

1. Batch size가 1이라고 해봅시다...

Unit 03 | BatchNormalization

그 밖에 다른 Normalization

Weight Normalization

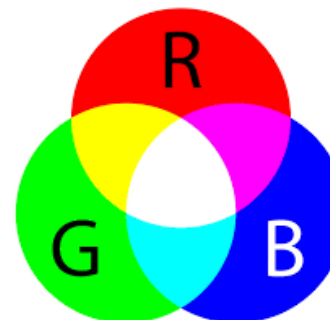
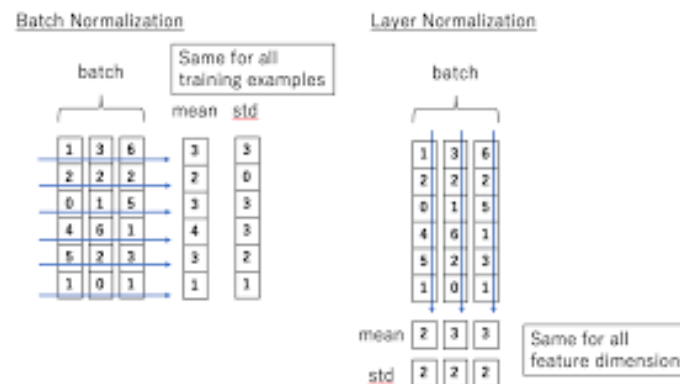
1. Layer의 가중치를 정규화한다.
2. Batch Normalization보다 빠르다.

Layer Normalization

1. Batch Normalization가 거의 유사하다.
2. BN은 Batch 차원에서 정규화 / LN은 Feature 차원 정규화인게 차이점!

Instance Normalization

1. LN과 유사하지만 각 example의 각 채널에 정규화를 진행!
2. 이미지에 국한된 Normalization(GAN에서 보통 대체하여 사용)
3. Real-time Generation에 효과적



Contents

Unit 01 | Activation Function

Unit 02 | Weight Initialization

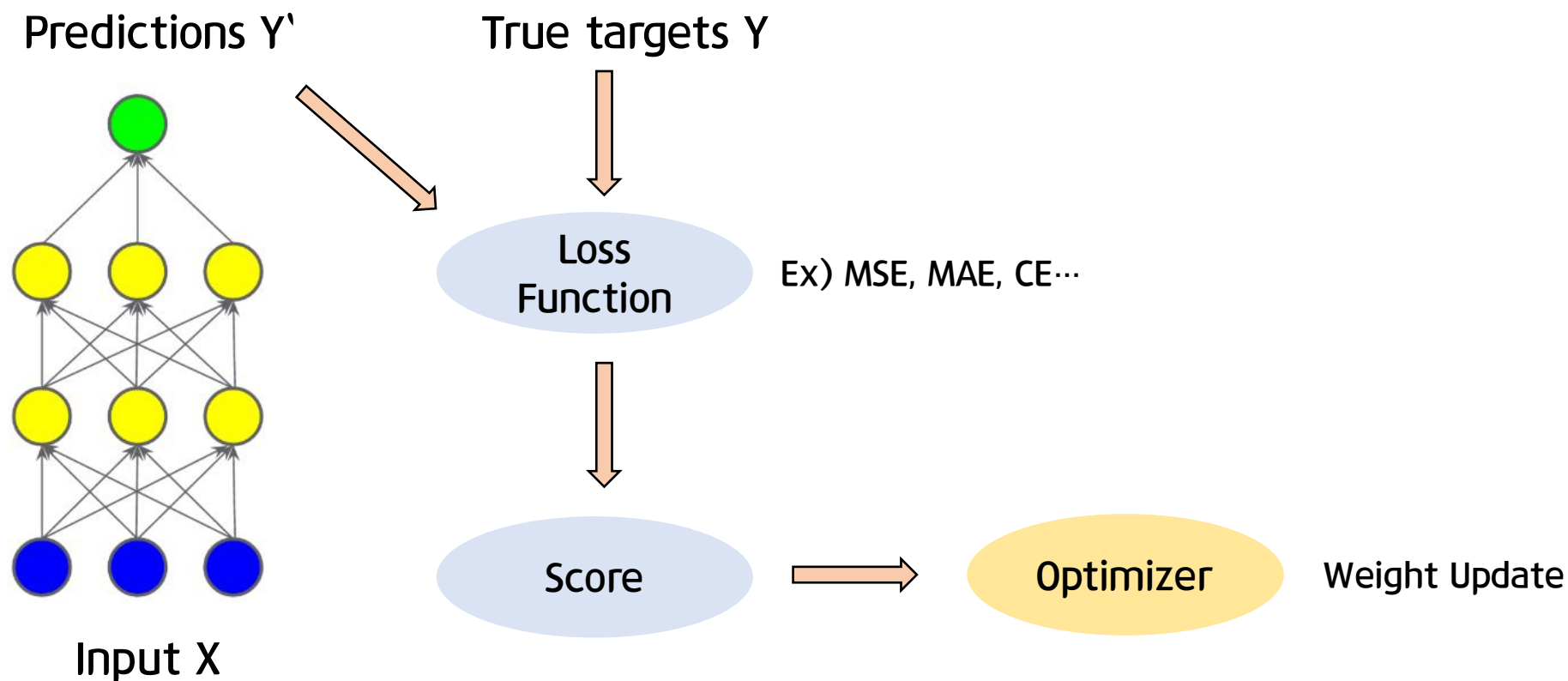
Unit 03 | Batchnormalization

Unit 04 | Optimization

Unit 05 | Regularization

Unit 04 | Optimization

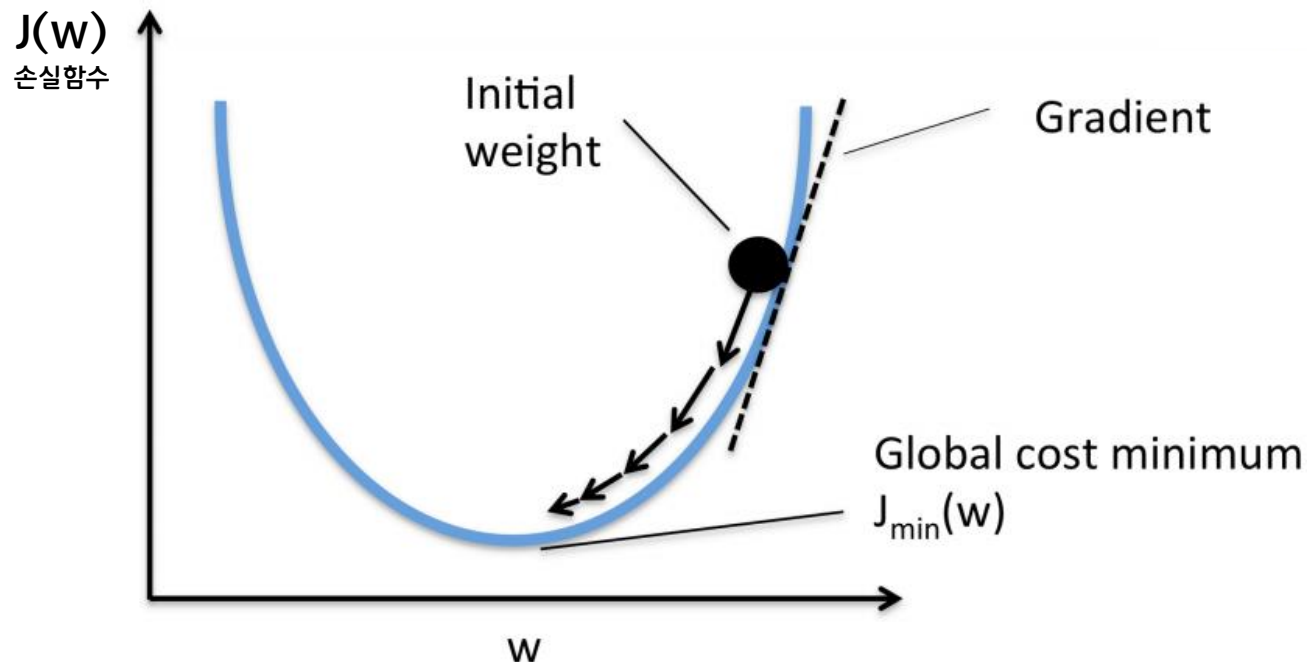
딥러닝의 학습 기본 원리



Unit 04 | Optimization

Neural Network에서의 Optimization은 학습과 같다.

- Neural Network 학습 : Loss Function 값을 최소화하는 Optimization 과정
- Loss Function을 최소화하는 가중치를 찾기 위해서 Gradient Descent 알고리즘을 사용합니다.



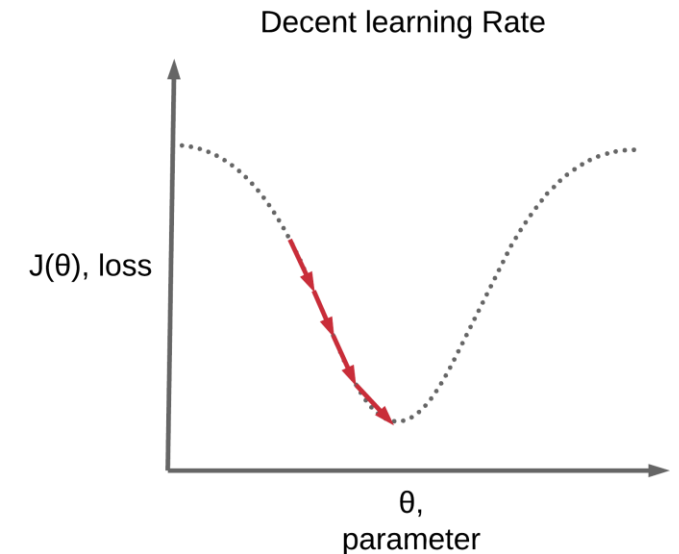
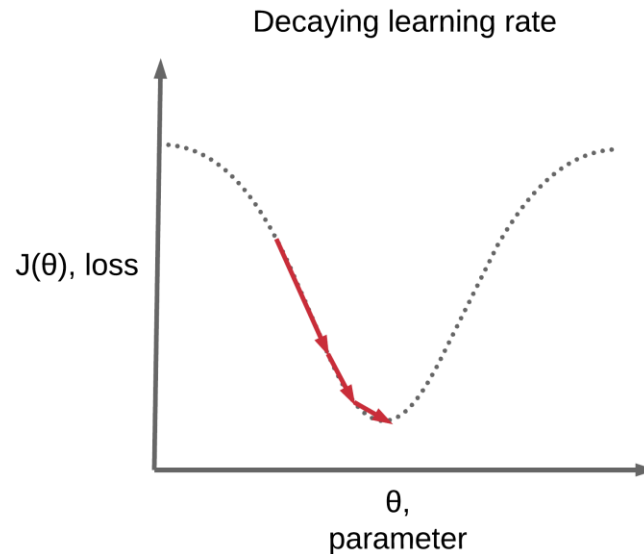
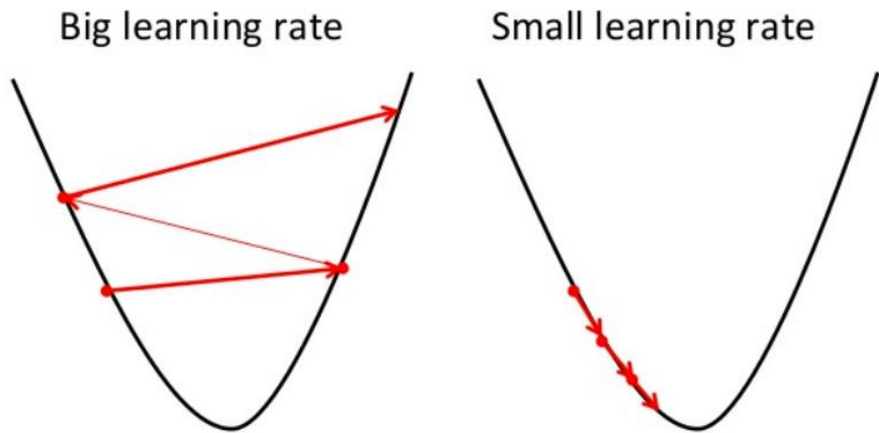
$$W_{update} = W - \underset{\text{Learning Rate}}{StepSize} \times \frac{\partial}{\partial W} J(W) \underset{Gradient(W)}{}$$

$$\text{Ex) } J(W) = \text{MSE}(y_{pred}, y_{true})$$

Unit 04 | Optimization

Learning Rate는 얼마나 잡아야 하나요?

- 적당하게~ 잡으면 됩니다.
- Learning Rate 스케줄러로 처음엔 크게 했다가 점점 줄이는 방법도 있습니다.

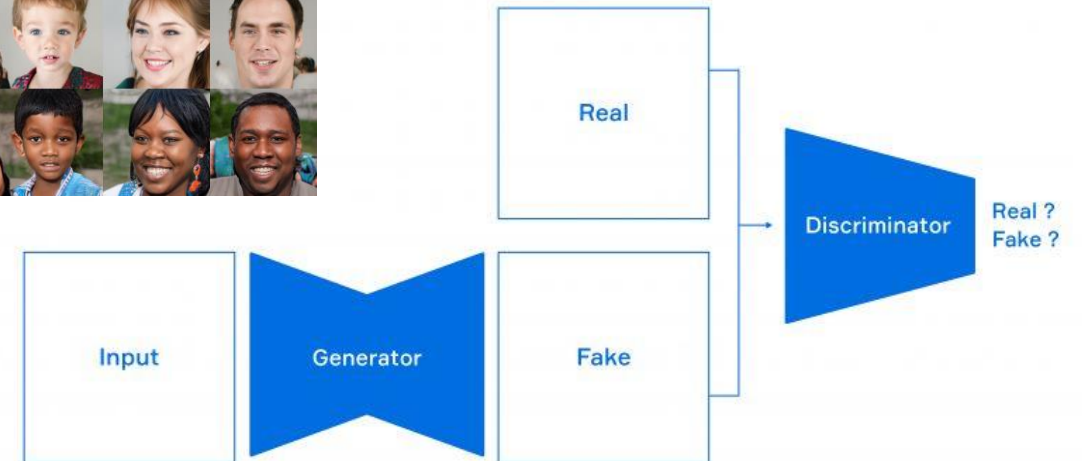
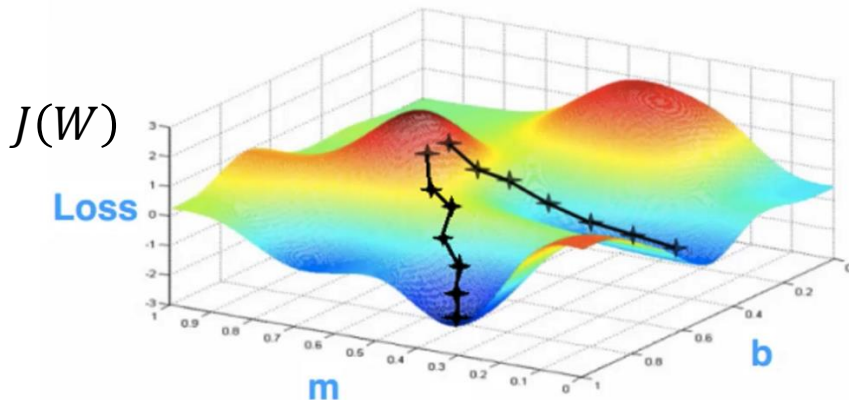


Unit 04 | Optimization

넓게 보는 Optimization 풀이 원리

- Optimization 문제는 목적함수에 따라 최대화(Maximization), 최소화(Minimization) 문제로 나눌 수 있다.
- Quiz : 목적함수가 이윤, 점수(score) 등인 경우? 혹은 비용(cost), 손실(loss), 에러(error)인 경우에는?
- 하지만 최대화 / 최소화든 풀이 원리는 같다. 조금 조금씩 해를 찾아갑니다.

Gradient Descent

 $f(x) = \text{nonlinear function of } x$ 

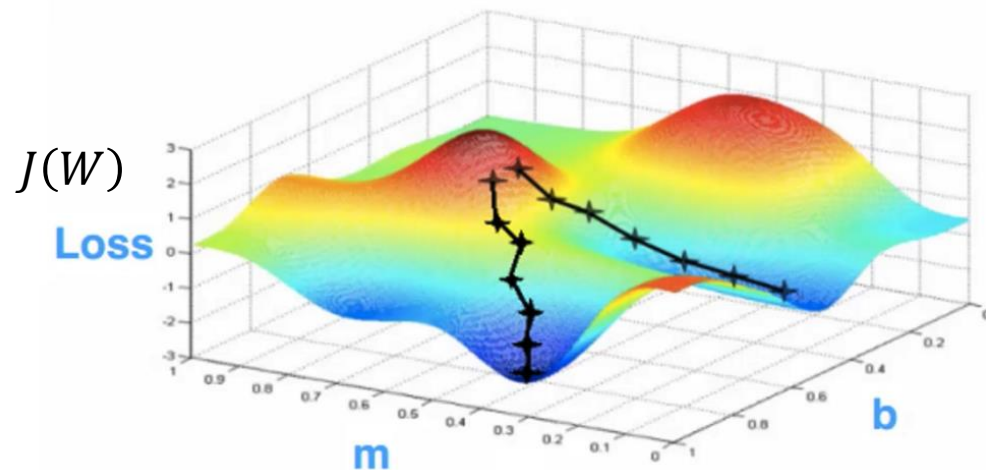
Unit 04 | Optimization

Local Minima VS Optimal Minima – 이게 진짜 해일까?

- 손실함수가 가장 작은 Optimal Minima의 가중치가 아닌 이상한 Minima로 빠질 수 있다.
- 딥러닝에서의 Global optimal은 보장하기는 힘들다. Why? Non-Convex이기 때문이다.

Gradient Descent

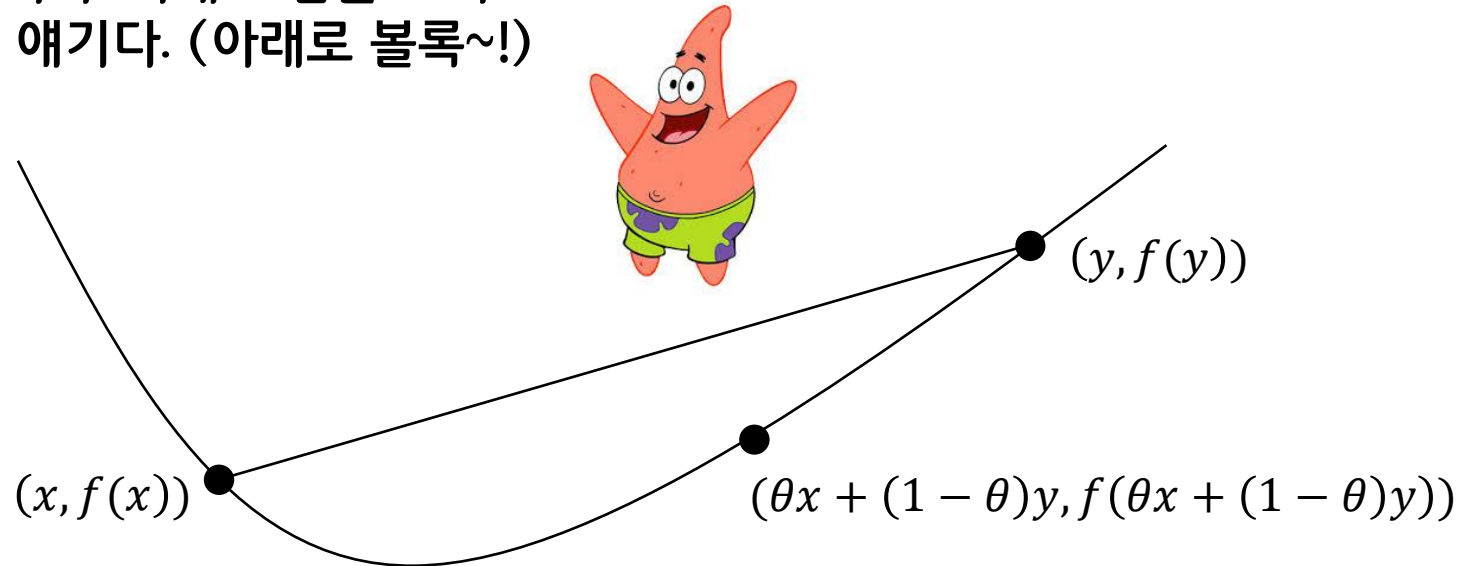
$f(x)$ = nonlinear function of x



Unit 04 | Optimization

Convex 란 무엇이나

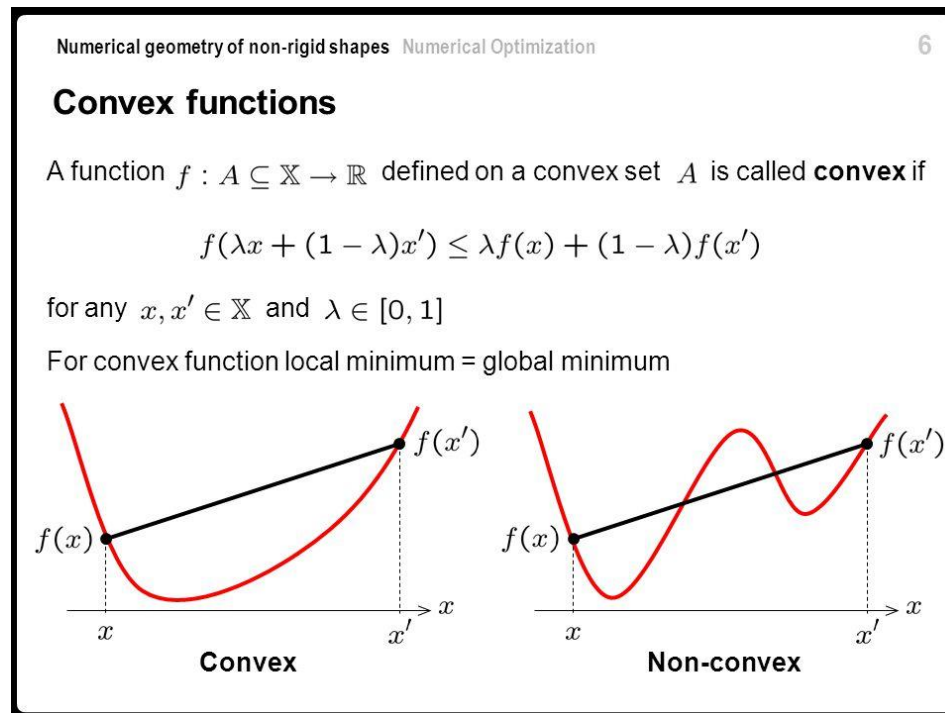
- 수식은 다음과 같다. $f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$ for all $x, y \in \text{dom } f, 0 \leq \theta \leq 1$
- 전혀 어려운 얘기는 아니다. 아래 그림을 보자.
- 선분이 더 위에 있다는 얘기다. (아래로 볼록~!)



Unit 04 | Optimization

Non Convex는 그래서 local minima랑 Optimal minima가 같다는 보장이 없다.

- 그럼 Convex에서 Local Minima랑 Optimal Minima가 같다는 증명은? 응 결론 부정 귀류법~



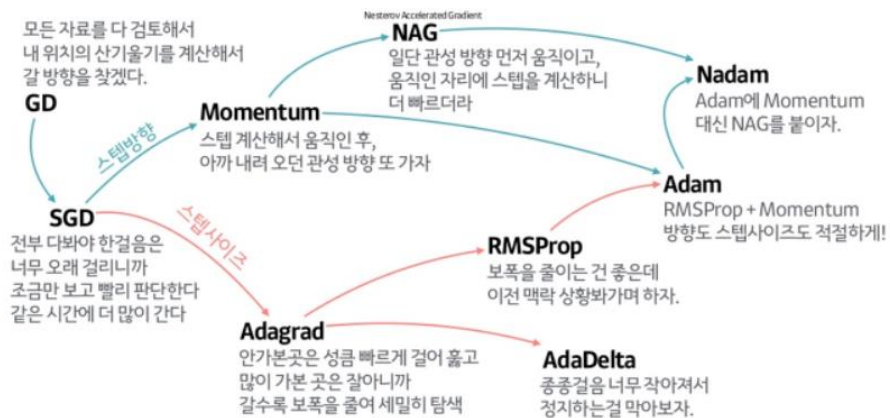
Unit 04 | Optimization

Neural Network에서의 Optimization으로 돌아옵니다.

- 가중치를 업데이트를 하는 식은 아래와 같은데 이러한 Optimizer도 여러가지 종류가 있습니다.

$$W_{update} = W - \underset{\text{Learning Rate}}{\text{StepSize}} \times \frac{\partial}{\partial W} J(W)$$

- 아래 그림 이외에도 같이 정말 많습니다. 대표적인것 몇 개만 알아보시다.

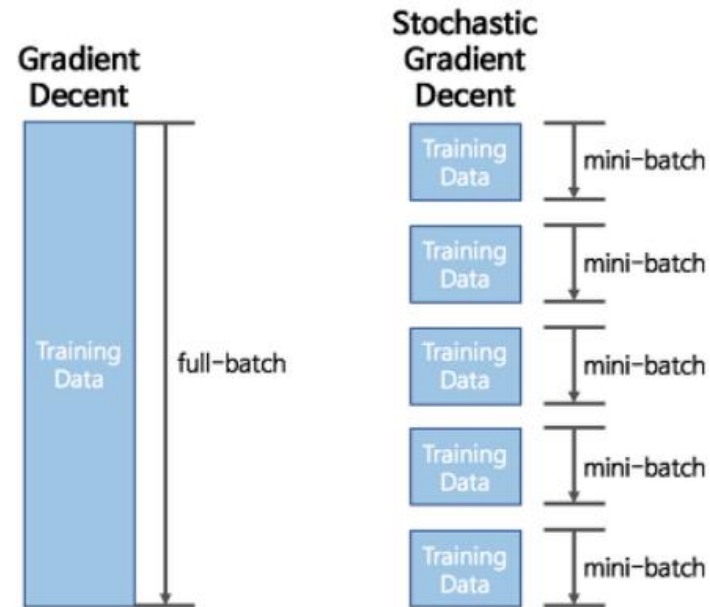


Unit 04 | Optimization

1. 확률적 경사 하강법 - SGD(Stochastic Gradient Descent)

- Local Minima에서 빠져나가기 위해 확률적으로 임의의 배치를 뽑고 그 배치를 기준으로 Update합니다.
- BatchSize가 1이면 데이터 개수만큼 Update, BatchSize가 전체면 한번 가중치를 Update합니다.
- Quiz. BatchSize는 어떻게 결정할까U?

Ans : 컴퓨터 스펙이 허락하는 한 최대한 크게

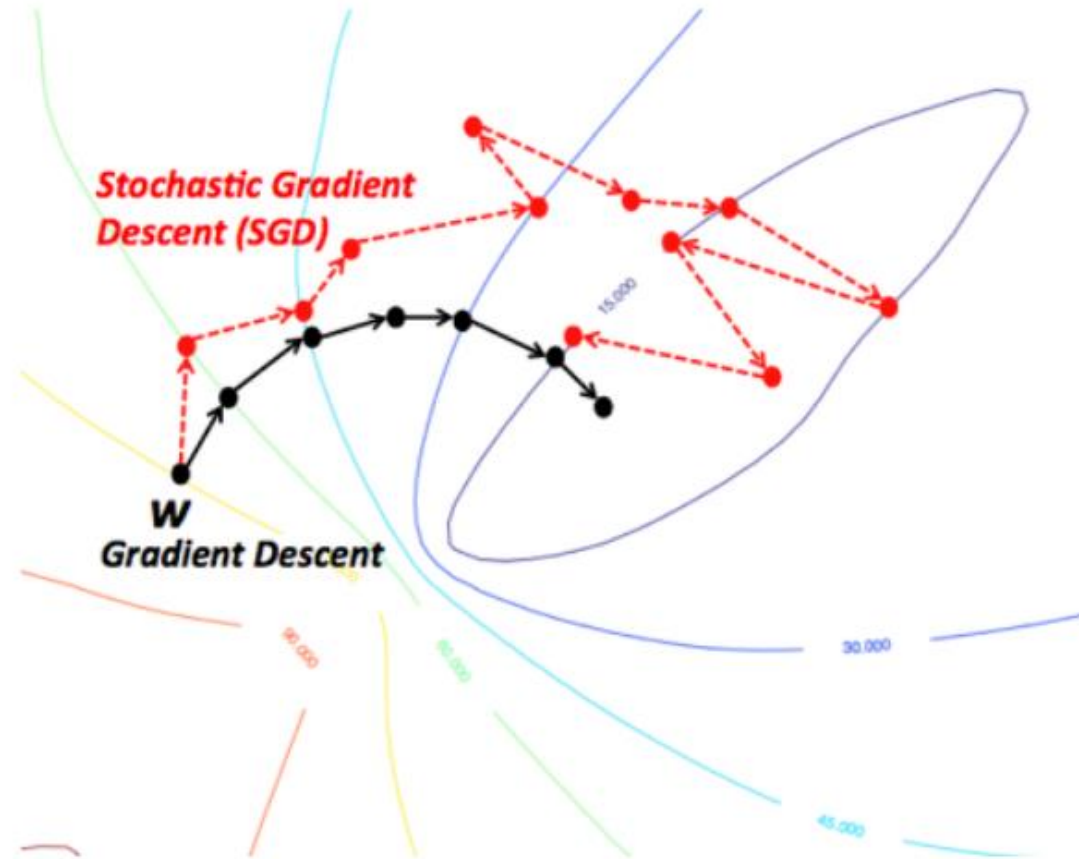


Size= 1 or N????

Unit 04 | Optimization

SGD의 단점 : 무엇일까요?

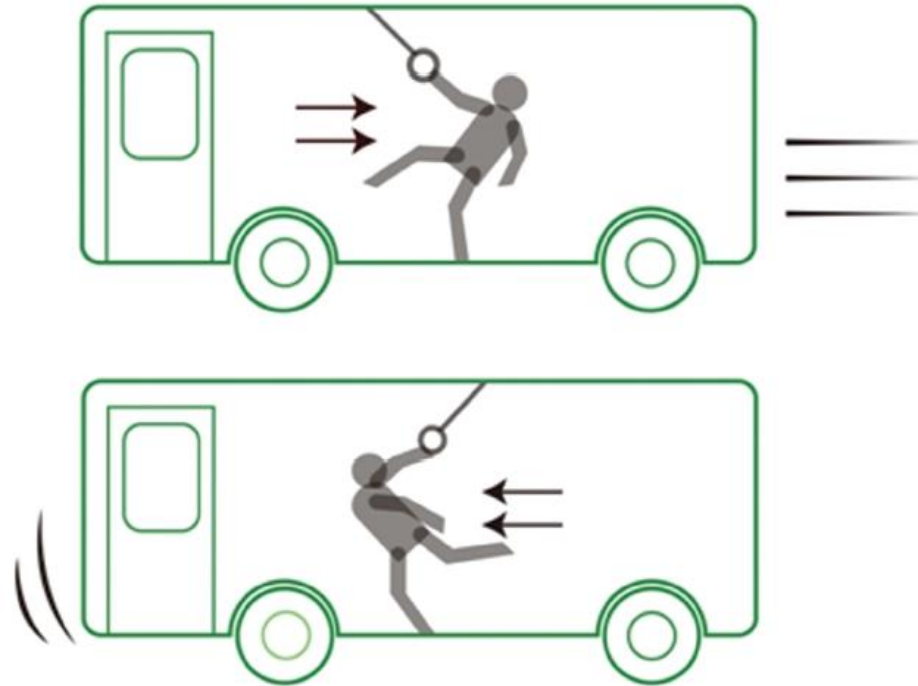
- 옆 그림을 한 번 봐주세요!



Unit 04 | Optimization

새로 등장한 Optimizer의 핵심 개념 : Momentum

- 모멘텀이란?



- 모멘텀이랑 Optimizer가 뭘 관계가 있을까요?

Unit 04 | Optimization

2. 모멘텀으로 SGD가 진동하는 것을 막자!!

- 수식으로 나타내면 아래와 같다.

$$W_{update} = W - \underset{\text{Learning Rate}}{\text{StepSize}} \times \frac{\partial}{\partial W} J(W)$$



- 여기에 velocity를 따로 구해준다. (왜 갑자기? 모멘텀이 운동량인데 이 식이 질량x속도이기 때문입니다.)

$$Velocity_{update} = \underset{\text{상수}}{\text{Momentum}} \times Velocity - \text{StepSize} \times \frac{\partial}{\partial W} J(W)$$

$$W_{update} = W + Velocity_{update}$$

- 코드 예시

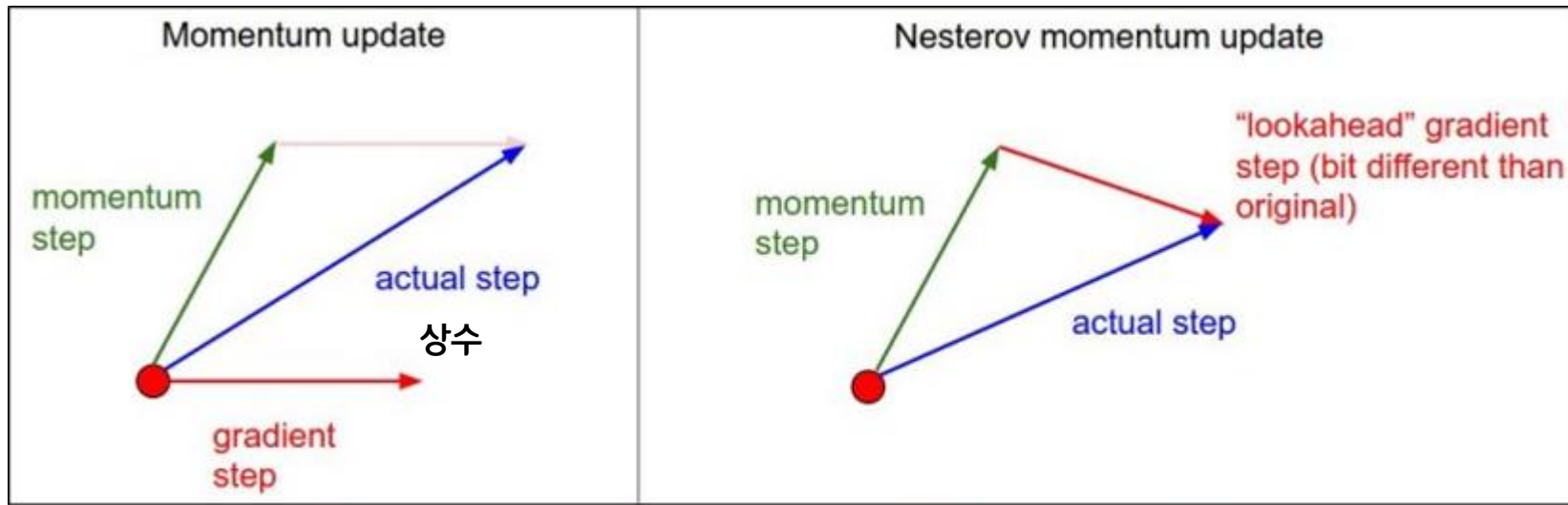
```
keras.optimizers.SGD(lr=0.1, momentum=0.9)
```

- 장점 : 속도향상, Local Minima 탈출

Unit 04 | Optimization

3. Gradient가 같 방향에 미리 예측샷 하자! – Nesterov Accelerated Gradient(NAG)

- Nesterov Accelerated Gradient : Momentum Step이 먼저 적용된 자리에서 Gradient 계산 (Brake)



Difference between Momentum and NAG. Picture from CS231.

- 수식은 다음과 같이 바뀝니다. $W_{update} = W + Velocity_{update}$

$$Velocity_{update} = Momentum \times Velocity - StepSize \times \frac{\partial}{\partial W_{update}} J(W)$$

Unit 04 | Optimization

4. Adagrad (Adaptive Gradient)

- W 가중치의 업데이트 정도(횟수x, 값 차이o)에 따라 학습률을 조절하는 옵션이 추가된 최적화 방법입니다.
- 전체 W가 많이 바뀌었으면 StepSize를 크게 합니다.
- 여기서는 Velocity 대신 새로운 함수가 들어옵니다.

$$G_{update} = G + \left\{ \frac{\partial}{\partial W} J(W) \right\}^2 = \sum_{i=0}^T \left(\frac{\partial}{\partial W(i)} J(W(i)) \right)^2 \quad \text{전체 Gradient의 제곱합}$$

$$W_{update} = W - \alpha \times \frac{1}{\sqrt{G_{update} + \epsilon}} \times \frac{\partial}{\partial W(i)} J(W(i))$$

- 장점 : 같은 입력 데이터가 여러 번 학습되는 모델에 강점을 보입니다.(Word2Vec, Glove)

많이 등장한 단어는 가중치를 적게 수정하고 적게 등장한 단어는 많이 수정합니다!

```
g += gradient**2
weight[i] += - learning_rate ( gradient / (np.sqrt(g) + e)
```

Unit 04 | Optimization

4. Adagrad (Adaptive Gradient)

- W 가중치의 업데이트 정도(횟수x, 값 차이o)에 따라 학습률 조절하는 옵션이 추가된 최적화 방법입니다.
- 전체 W가 많이 바뀌었으면 StepSize를 크게 합니다.
- 여기서는 Velocity 대신 Gradient들의 제곱합인 G 함수가 들어옵니다.

$$G_{update} = G + \left\{ \frac{\partial}{\partial W} J(W) \right\}^2 = \sum_{i=0}^T \left(\frac{\partial}{\partial W(i)} J(W(i)) \right)^2$$

$$W_{update} = W - \alpha \times \frac{1}{\sqrt{G_{update} + \epsilon}} \times \frac{\partial}{\partial W(i)} J(W(i))$$

- 단점 : 학습이 오래 진행될수록 거의 움직이지 않습니다. 왜? 계속 Gradient 값들은 증가하니까!

Unit 04 | Optimization

5. RMSprop (지수 이동 평균으로 Gradient가 커지는 것을 막자!)

- 지수 이동평균이란? 최근 값을 더 잘 반영하기 위해 최근 값과 이전 값에 가중치를 주어 계산
- 이게 왜 좋을까? 등비수열을 생각하면 곱할수록 0에 수렴하니까 계속 G가 증가하진 않습니다.
- 즉, 막상 다른건 아래 표시한 부분 밖에 없습니다!

$$G_{update} = \gamma G + (1 - \gamma) \left\{ \frac{\partial}{\partial W} J(W) \right\}^2$$

$$W_{update} = W - \alpha \times \frac{1}{\sqrt{G_{update} + \epsilon}} \times \frac{\partial}{\partial W(i)} J(W(i))$$

• 코드 예시

```
g = gamma * g + (1 - gamma) * gradient**2
weight[i] += -learning_rate * gradient / (np.sqrt(g) + e)
```

$$\begin{aligned} x_k &= \alpha p_k + (1 - \alpha) x_{k-1} \\ &= \alpha p_k + (1 - \alpha) \{ \alpha p_{k-1} + (1 - \alpha) x_{k-2} \} \\ &= \alpha p_k + \alpha(1 - \alpha) p_{k-1} + (1 - \alpha)^2 x_{k-2} \\ &= \alpha p_k + \alpha(1 - \alpha) p_{k-1} + (1 - \alpha)^2 \{ \alpha p_{k-2} + (1 - \alpha) x_{k-3} \} \\ &= \alpha p_k + \alpha(1 - \alpha) p_{k-1} + \alpha(1 - \alpha)^2 p_{k-2} + (1 - \alpha)^3 x_{k-3} \\ &= \dots \\ &= \alpha (p_k + (1 - \alpha) p_{k-1} + (1 - \alpha)^2 p_{k-2} + \dots + (1 - \alpha)^{N-1} p_{k-N+1}) + (1 - \alpha)^N x_{k-N} \\ &= \alpha \sum_{i=0}^{N-1} (1 - \alpha)^i p_{k-i} + (1 - \alpha)^N x_{k-N} \end{aligned}$$

Unit 04 | Optimization

6. Adam (옵티마이저는 쓰까 무야 제맛이지!)

- 모멘텀, 학습률 조정, 지수평균이동을 다 합치면 어떤 혼종이 나올까요?

$$M_{update} = \beta_1 M + (1 - \beta_1) \frac{\partial}{\partial W} J(W)$$

$$V_{update} = \beta_2 V + (1 - \beta_2) \left\{ \frac{\partial}{\partial W} J(W) \right\}^2$$

논문에 V라 되어 있지만 앞 슬라이드의 G와 동일합니다.

$$W_{update} = W - \alpha \times \frac{M_{bias_corr}}{\sqrt{V_{bias_corr}} + \varepsilon}$$

$$M_{bias_corr} = \frac{M_{update}}{1 - \beta_1^t} \quad V_{bias_corr} = \frac{V_{update}}{1 - \beta_2^t}$$

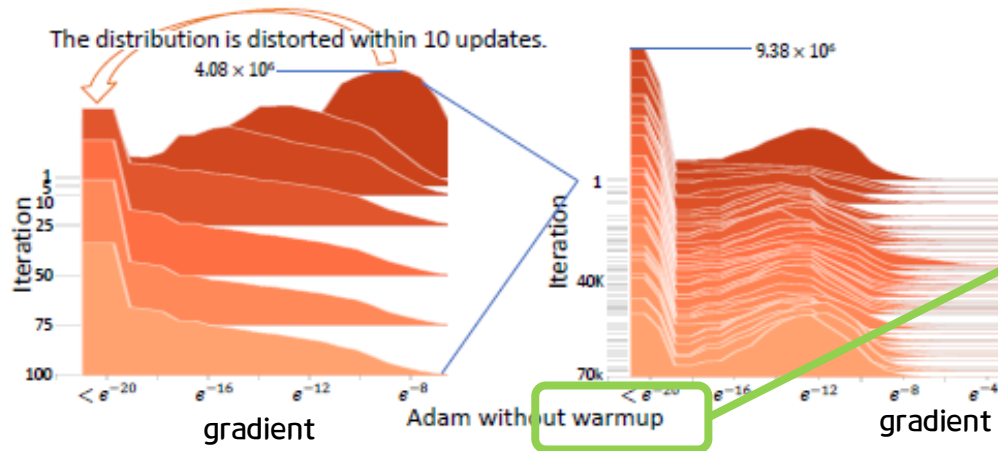
이렇게 Bias Correction을 하는 이유
수도 코드를 보면 초반에 0으로 초기화를 하기 때문에 0으로 bias되는 문제를 해결하기 위해서입니다.

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

Unit 04 | Optimization

6. Adam 의 한계 : Bad local optima convergence problem

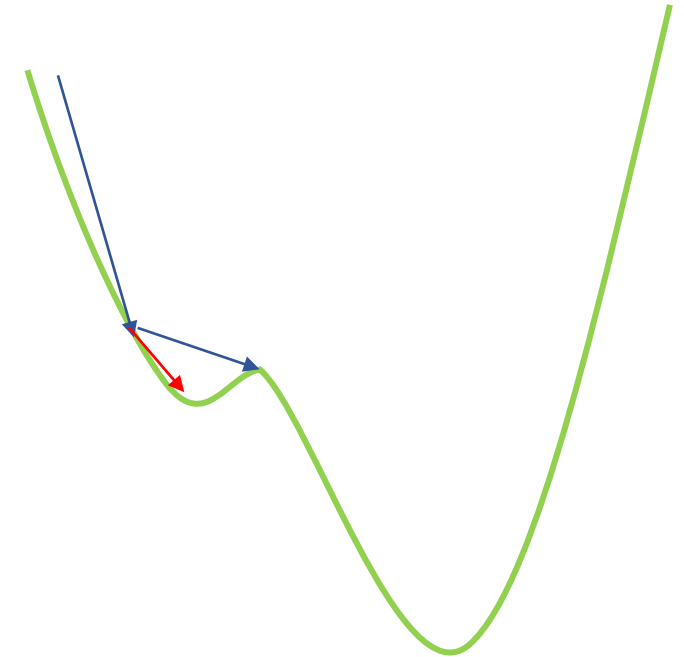
- 학습률 조정(Adaptive learning rate)를 사용하는 optimizer 이기 때문에 일어나는 현상입니다.



Warmup Heuristic은
샘플이 적은 초기에
선형적으로 조금씩 증가하는
Learning rate를 사용합니다.

$$\text{Var}(aX) = a^2 \text{Var}(X) < \text{Var}(x)$$

$\text{Var}(x)$: Adaptive learning rate의 분산



학습 초기를 조금만 지나면 gradient가 e^{-20} , 매우 작은 local minima 으로 수렴합니다.
Why? 학습 초기에는 논문에서 제시한 Adaptive learning rate의 분산이 크기 때문입니다.

Unit 04 | Optimization

7. Radam : 수학 배경 – Gradient 가 정규분포라 가정하면 그 역수의 제곱 분포는

- Gradient가 정규분포 이라고 가정시 Adaptive learning rate는 scaled inverse chi- square 분포를 가집니다. 즉, $\chi^2(t, \frac{1}{\sigma^2})$ 를 따릅니다. Why?

$\{g_1, \dots, g_t\}$: t iteration 때의 gradient 값입니다. 가중치들의 지수 이동 평균의 분포는 다음과 같습니다.

$$V_{bias_corr} = \frac{V_t}{1 - \beta_2^t}$$

$$V_t = \beta_2 V_{t-1} + (1 - \beta_2) \left\{ \frac{\partial}{\partial W} J(W) \right\}^2$$

$$W_t = W_{t-1} - \alpha \times \frac{M_{bias_corr}}{\sqrt{V_{bias_corr}} + \epsilon}$$

$$\frac{1}{\sqrt{V_{bias_corr}}} = \sqrt{\frac{1 - \beta_2^t}{V_t}}$$

$$\therefore \sim \text{Scale - inv - } \chi^2(t, \frac{1}{\sigma^2})$$

$$\beta_2^t \approx 1$$

$$p\left(\sqrt{\frac{1 - \beta_2^t}{(1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} g_i^2}}\right) \approx p\left(\sqrt{\frac{t}{\sum_{i=1}^t g_i^2}}\right)$$

$$\text{Var}[\psi(.)] \approx \frac{\rho_t}{2(\rho_t - 2)(\rho_t - 4)\sigma^2}$$

Unit 04 | Optimization

7. Radam : Rectified Adam (ICLR 2020) – Bad local convergence problem

- 학습률에 rectification이라는 변수를 곱해 learning rate가 일정한 분산을 가지도록 합니다. Rectify란?

$$M_t = \beta_1 M + (1 - \beta_1) \frac{\partial}{\partial W} J(W)$$

$$V_t = \beta_2 V + (1 - \beta_2) \left\{ \frac{\partial}{\partial W} J(W) \right\}^2$$

$$W_t = W_{t-1} - \alpha \times \mathbf{r}_t \frac{M_{bias_corr}}{\sqrt{V_{bias_corr}} + \varepsilon}$$

$$M_{bias_corr} = \frac{M_t}{1 - \beta_1^t} \quad V_{bias_corr} = \sqrt{\frac{V_t}{1 - \beta_2^t}}$$

If ρ_t 가 4보다 클때

$$\rho_\infty = \frac{2}{(1 - \beta_2)} - 1 \quad \text{초기화} \quad \rho: \text{자유도로서 학습 step이 진행된 정도}$$

$$\rho_t = \rho_\infty - \frac{2t\beta_2^t}{(1 - \beta_2^t)}$$

If ρ_t 가 4보다 크면?

elif ρ_t 가 4보다 작으면?

$$\mathbf{r}_t = \sqrt{\frac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}$$

$$W_t = W_{t-1} - \alpha \times \mathbf{r}_t \frac{M_{bias_corr}}{V_{bias_corr}}$$

$$W_t = W_{t-1} - \alpha \times M_{bias_corr}$$

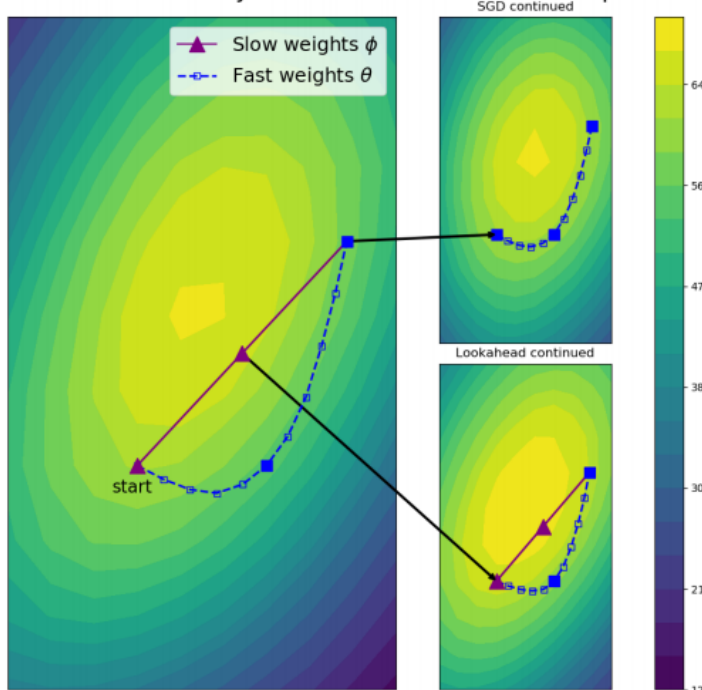
$$\sqrt{\frac{\text{Var}[\psi(.)]_\infty}{\text{Var}[\psi(.)]_t}}$$

Unit 04 | Optimization

8. Lookahead Optimizer – Slow Weights, Fast Weights 로 나누어서 학습하자!

- 속도가 빠른 optimizer로 weight 여러번을 구한 후에 전체 가중치와 지수 이동 평균을 때립니다.

CIFAR-100 accuracy surface with Lookahead interpolation

**Algorithm 1** Lookahead Optimizer:

Require: Initial parameters ϕ_0 , objective function L
Require: Synchronization period k , slow weights step size α , optimizer A

```

for  $t = 1, 2, \dots$  do
  Synchronize parameters  $\theta_{t,0} \leftarrow \phi_{t-1}$ 
  for  $i = 1, 2, \dots, k$  do
    sample minibatch of data  $d \sim \mathcal{D}$ 
     $\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$ 
  end for
  Perform outer update  $\phi_t \leftarrow \phi_{t-1} + \alpha(\theta_{t,k} - \phi_{t-1})$ 
end for
return parameters  $\phi$ 

```

Contents

Unit 01 | Activation Function

Unit 02 | Weight Initialization

Unit 03 | Batchnormalization

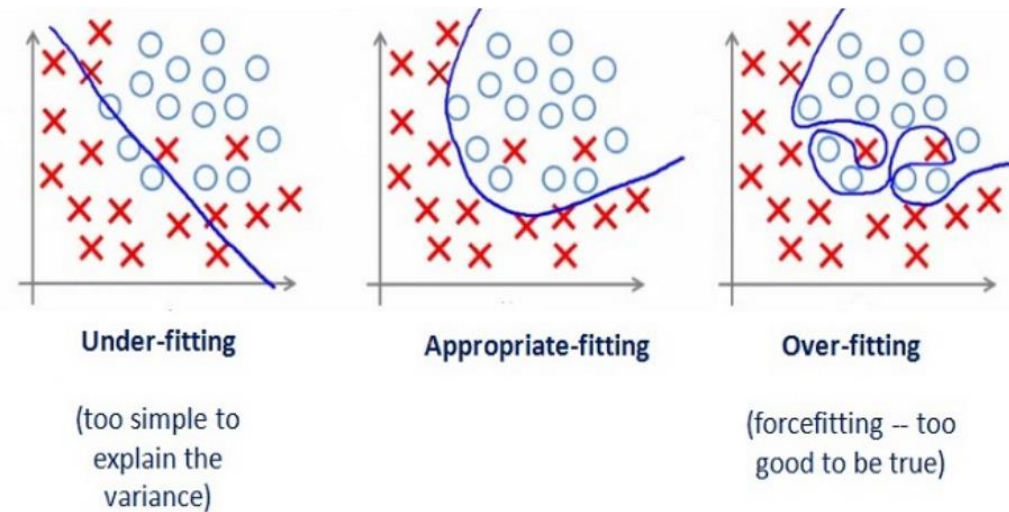
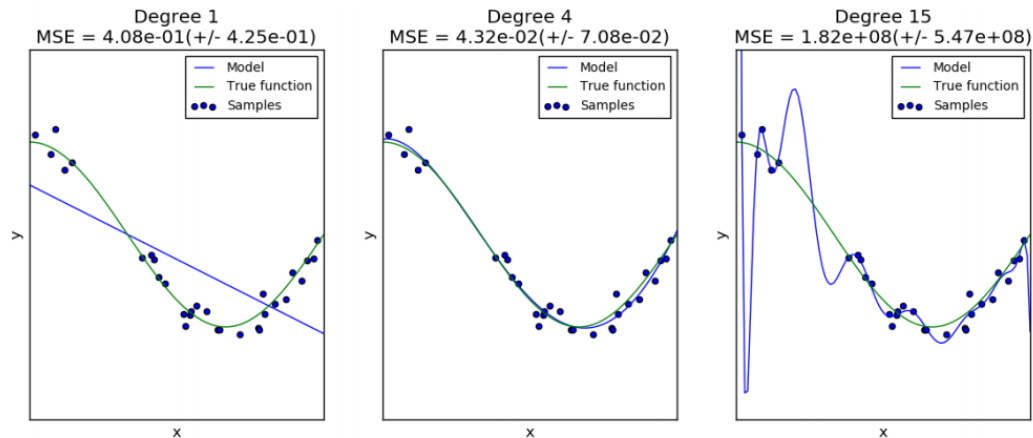
Unit 04 | Optimization

Unit 05 | Regularization

Unit 05 | Regularization

규제는 무엇이고 왜 필요할까요?

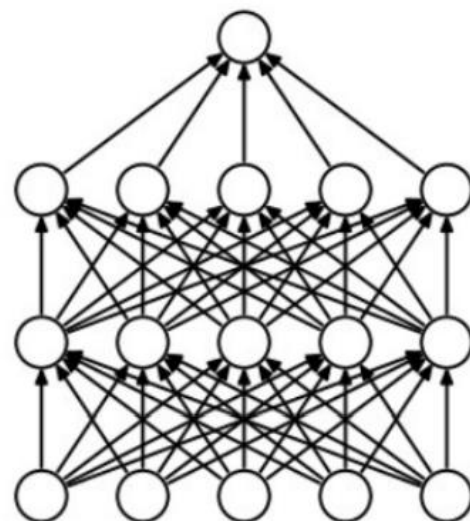
- Overfitting 때문에 필요합니다. 그래서 우리는 L1, L2 규제를 통해 모델링하는 것을 배웠습니다.
- 방법 1. 신경망도 Layer별로 L1(절댓값 – Outlier 무시), L2(제곱 – Outlier 반응) 규제하는 것이 가능합니다!



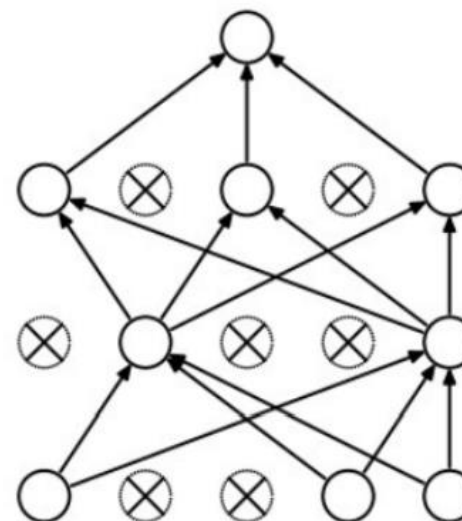
Unit 05 | Regularization

방법 2. 딥러닝에서 쓰는 대표적인 규제는 바로 Dropout입니다!

- 힌튼 아저씨가 만들었습니다.
- Neural Net이 알코올 먹은 것 마냥 중간 노드 학습을 끊어버립니다.



(a) Standard Neural Net

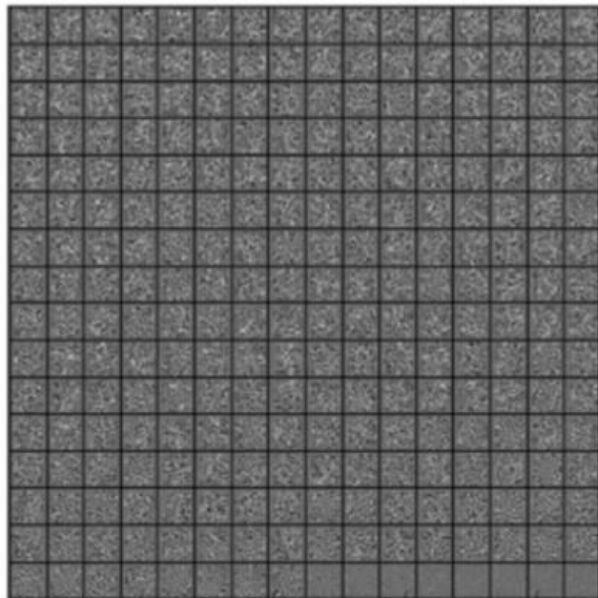


(b) After applying dropout.

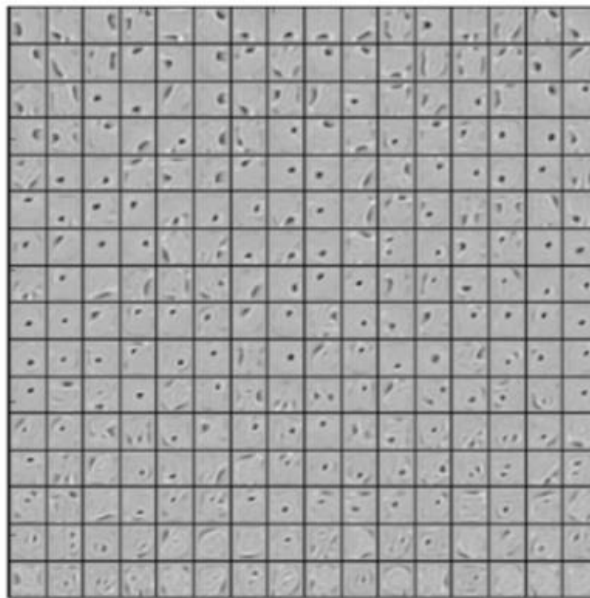
Unit 05 | Regularization

왜 Dropout이 효과가 좋을까요?

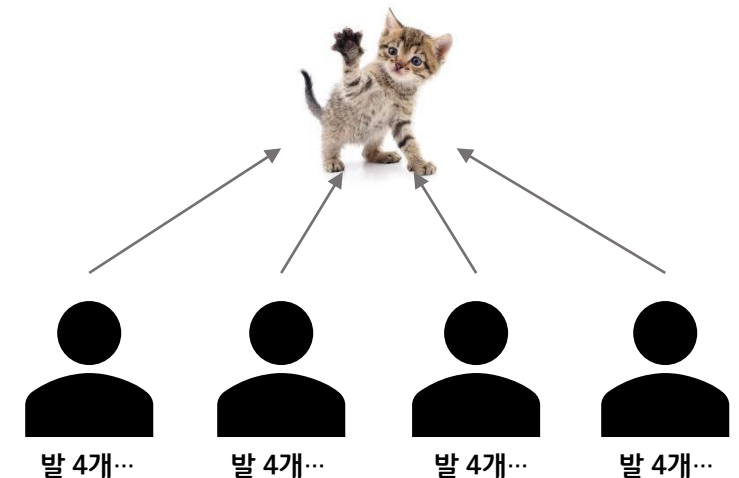
- 학습을 하다보면 Weight들이 서로 동조하는 현상이 발생합니다.
- 이를 의도적으로 무시하면서 동조화현상을 피할 수 있습니다. 단, 학습시간은 증가합니다.



(a) Without dropout

(b) Dropout with $p = 0.5$.

동조화현상이란?

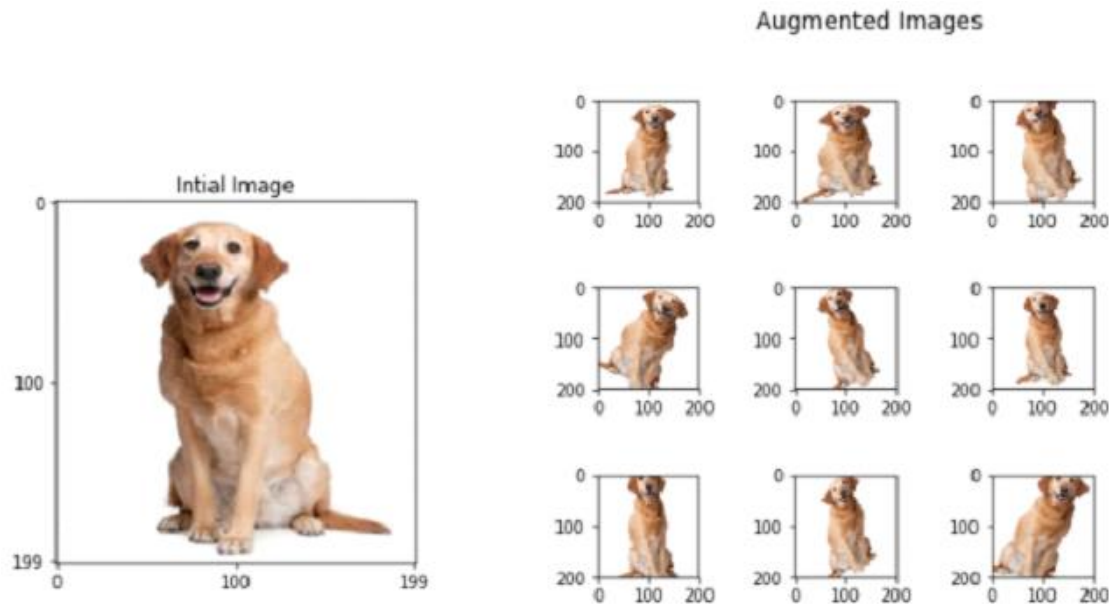


다 같은 특징을 추출합니다.

Unit 05 | Regularization

방법 3. 데이터 증식(Augmentation)!

- 기존 데이터에서 새로운 데이터를 생성해 인공적으로 훈련 세트의 크기를 늘리는 방법입니다.
- 이미지에선 회전, 이동, 뒤집기 등이 해당됩니다.

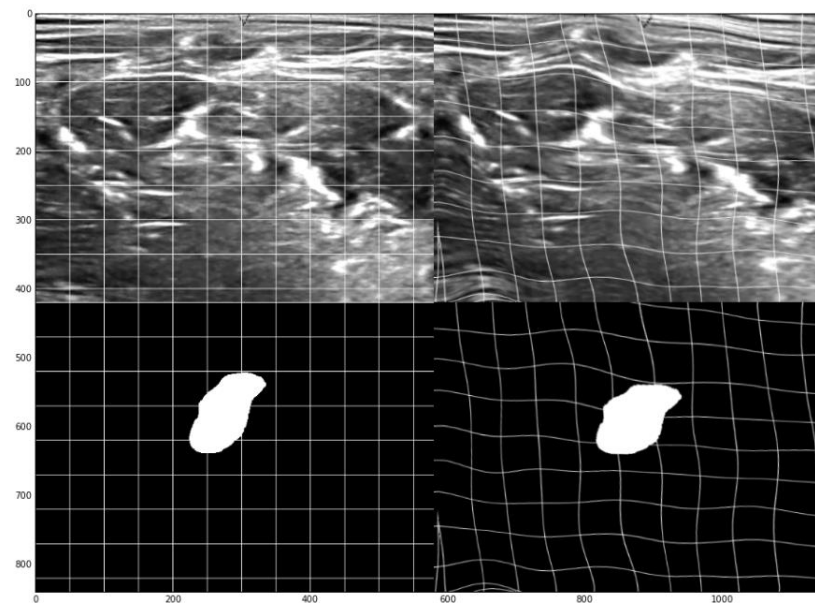


데이터의 종류가 다르면 다른 증식 기법을
사용해야 합니다!

Unit 05 | Regularization

데이터 증식 파이프라인 예시





- 회전, 이동, 확대 등의 Affine Transform 이외에도 여러가지 방법의 Augmentation이 있습니다.
- 가우시안 분포를 가진 난수를 더하거나 Elastic Transform을 통해 형태를 변화시킬 수 있습니다.



Unit 05 | Regularization

데이터 증식 심화

- 이런 특이한 데이터 증식 기법도 있습니다.

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	78.6 (+2.3)
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	47.3 (+1.0)
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	76.7 (+1.1)

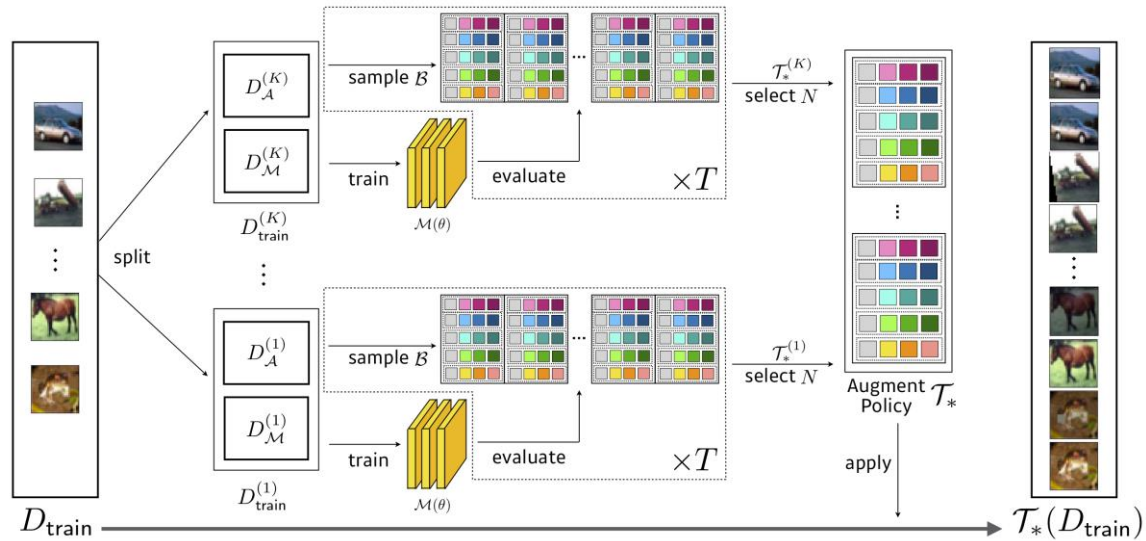
Cutmix: Regularization Strategy to Train Strong Classifiers with Localizable Features(2019, ICCV)

[Sangdoo Yun](#), [Dongyoon Han](#), [Seong Joon Oh](#), [Sanghyuk Chun](#), [Junsuk Choe](#), [Youngjoon Yoo](#)

Unit 05 | Regularization

실질적인 데이터 증식 기술!

- 이미지에서 대표적인 라이브러리는 imgaug입니다.
- 그 밖에 Autoaugment, GAN 등을 시도해볼 수 있습니다.



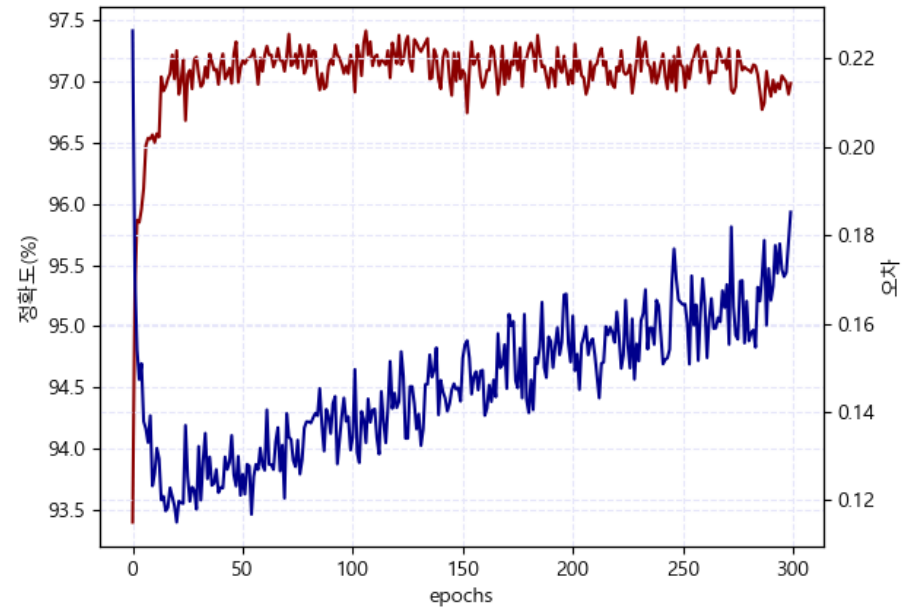
카카오의 Fast Autoaugment : 증식의 정책을 결정할 수 있습니다.



Unit 05 | Regularization

방법 4. Early Stopping입니다!

- Neural Net이 알코올 먹은 것 마냥 중간에 학습자체를 끊는 방법입니다.

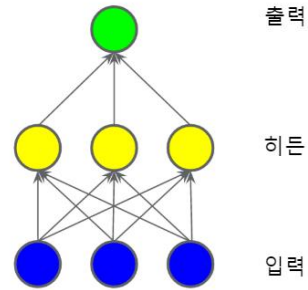


Unit -1 | Summary

Sigmoid, Tanh, Relu, Softmax

Activation
Function

Why? 비선형성



Weight, Layer, Instance

Batch
Normalization

Why? 모델 학습의 안정성 증가

Lecun, Xavier, He...

Weight
Initialization

Why? 가중치 초기화는 모델 성능에 영향

SGD, Momentum,
NAG, Adagrad,
Rmsprop, Adam

Optimizer

Why? 최적의 해 찾기

Regularization

Why? 오버피팅 방지

L1, L2, Dropout,
Augmentation,
Early Stopping

Unit -1 | 질문



Any Question?

Unit -2 | Reference

참고 레퍼런스

<https://towardsdatascience.com/regularization-the-path-to-bias-variance-trade-off-b7a7088b4577>

<https://www.quora.com/When-would-you-choose-L1-norm-over-L2-norm>

<https://www.stand-firm-peter.me/2018/09/24/l1l2/>

<https://m.blog.naver.com/laonple/220527647084>

<https://ratsgo.github.io/machine%20learning/2017/10/12/terms/>

<https://dalpo0814.tistory.com/29>

케라스 창시자에게 배우는 딥러닝

패턴인식과 머신러닝

텐서플로로 배우는 딥러닝

핸즈온 머신러닝

Numerical Method for engineers and scientists

밑바닥부터 시작하는 딥러닝 1/2

딥러닝의 정석

김기현의 자연어 처리 정석

컴퓨터 비전과 딥러닝

Unit -2 | Reference

참고 레퍼런스

11기 이소라 – 투빅스 NN심화 강의

<https://paperswithcode.com/sota>

<https://iccv19.mxnet.io/>

DEEP LEARNING

Deep learning cook book

파이썬으로 배우는 실전 알고리즘

<https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79>

<https://datascienceschool.net/view-notebook/4f3606fd839f4320a4120a56eec1e228/>

CS231n

<https://nittaku.tistory.com/269>

<https://pythonkim.tistory.com/41>

<https://sshkim.tistory.com/151>

Q & A

들어주셔서 감사합니다.