Regularization and Initialization

2025 Summer DL Session 4주차

KOREA
UNIVERSITY

목차

01
들어가기 전에,,

01. 들어가기 전에,,

3주차 과제 요약

• Multilayer Perceptron의 전체 로직을 구현해보자!

1. Sigmoid함수에 대한 이해와, 순전파 & 역전파 과정을 숙지
2. MLP의 구성요소인 layer, activation, optimizer를 구현
3. 모델의 hyperparameter와 optimizer를 바꾸면서 Training 결과를 확인

01. 들어가기 전에,,

3주차 과제 우수자

22기 성용빈 님

Sigmoid function 이론

22기 이은서 님

코드 과제

모두 과제하느라 수고많으셨습니다 ! :)

01. 들어가기 전에,,

4주차 학습 목표

정규화(Regularization)와 초기화(Initialization)를 통해
원활하고 좋은 학습 성능을 보이기 위한 방법들이 무엇인지 알아보자!

02

Regularization

Regularization ?

Normalizaion?

Standardization?

Regularization

- overfitting을 막기 위해, model의 parameter의 값을 제한시키는 방식
- simpler model을 만들기 위해 small weight를 줘 overfitting을 줄임
- cost term을 주거나 parameter 범위에 penalty를 주는 형식

Regularization vs Normalization

Regularization	Normalization
• overfitting 방지 목적 • cost, penalty 부여를 통해 weight 제약을 주는 것	• 0 ~ 1로 scale 변화 (scaling 목적) • ML: normalization은 값에 의한 dominated 현상 방지 • DL : local minima에 빠지는 것을 방지해 학습 속도 향상

참고. Normalization vs Standardization

- Normalization and Standardization are similar and both relate to the issue of feature scaling (to make training less sensitive to the scale of features of input data).
- If we train an algorithm using various scaled features of the input variables, then the results might be dominated by features with large magnitude.

Normalization (into [0, 1])

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

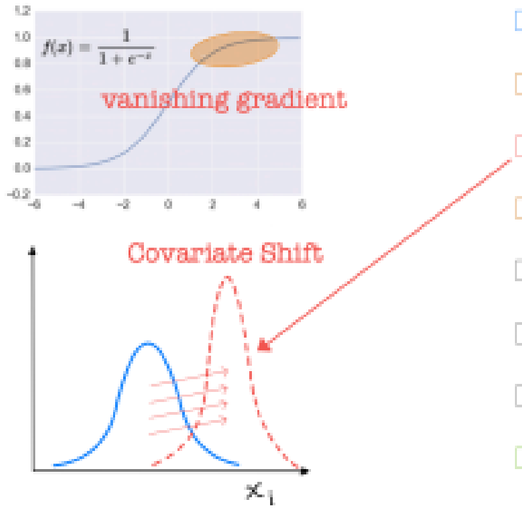
Standardization (z-score)

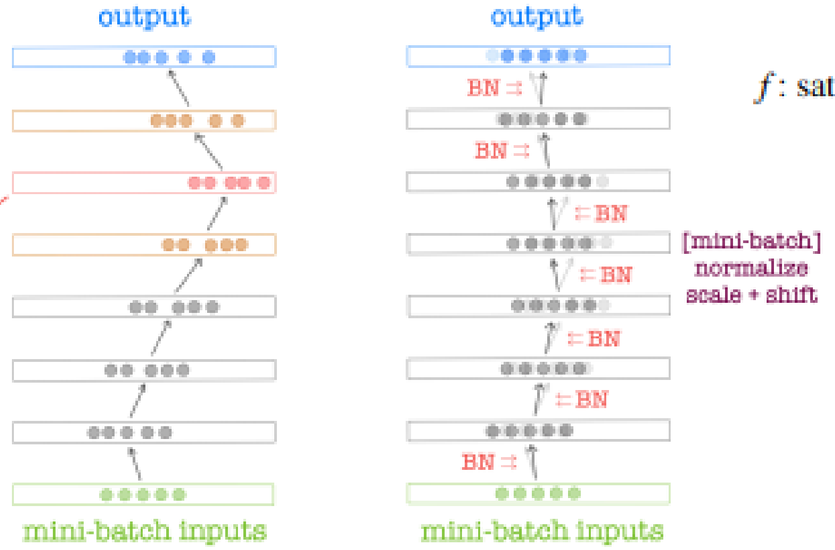
$$x_i' = \frac{x_i - \mu}{\sigma}$$

Batch Normalization

Internal Covariate Shift : distribution change of each layer's inputs during training
due to changes in the parameters of the previous layer. (the more layers, the more amplification)

- Since the inputs to each layer are affected by the parameters of all preceding layers, small changes to the parameters amplify the input values of the following layers.
- This requires lower learning rates and careful weight initialization, which slow down training, and makes it notoriously hard to train models with saturating nonlinearities.





f: saturating $\iff \lim_{z \to \pm\infty} |f(z)| < \propto$

sigmoid: saturating

ReLU: non-saturating

Batch Normalization

Batch normalization (BN) reduces Internal Covariate Shift to train fast by allowing us to use higher learning rates, saturating nonlinearities, and less careful weight initialization.

- BN transform can be freely added to any subset of activations $\{x_i^{(k)}\}$ to be normalized.
- For each activation $x_i^{(k)}$, it adds two learnable parameters scale $\gamma^{(k)}$ and shift $\beta^{(k)}$.
- $y_i^{(k)} = \mathrm{BN}_{\gamma,\beta}(x_i^{(k)})$ depends on $\{x_1^{(k)}, \ldots, x_m^{(k)}\}$ in the minibatch to be normalized.

Input: Values of x over a minibatch: $\mathcal{B} = \{x_1, \ldots, x_m\}$;
Parameters to be learned: γ, β

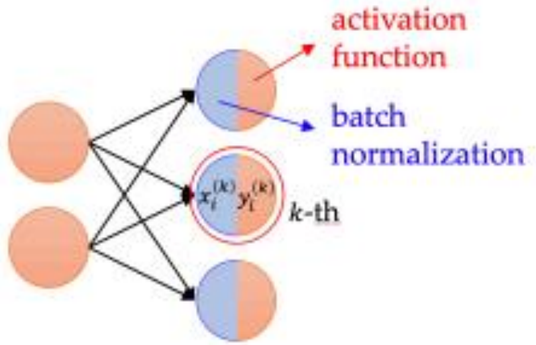
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$ // $x_i = x_i^{(k)}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// minibatch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// minibatch variance}$$

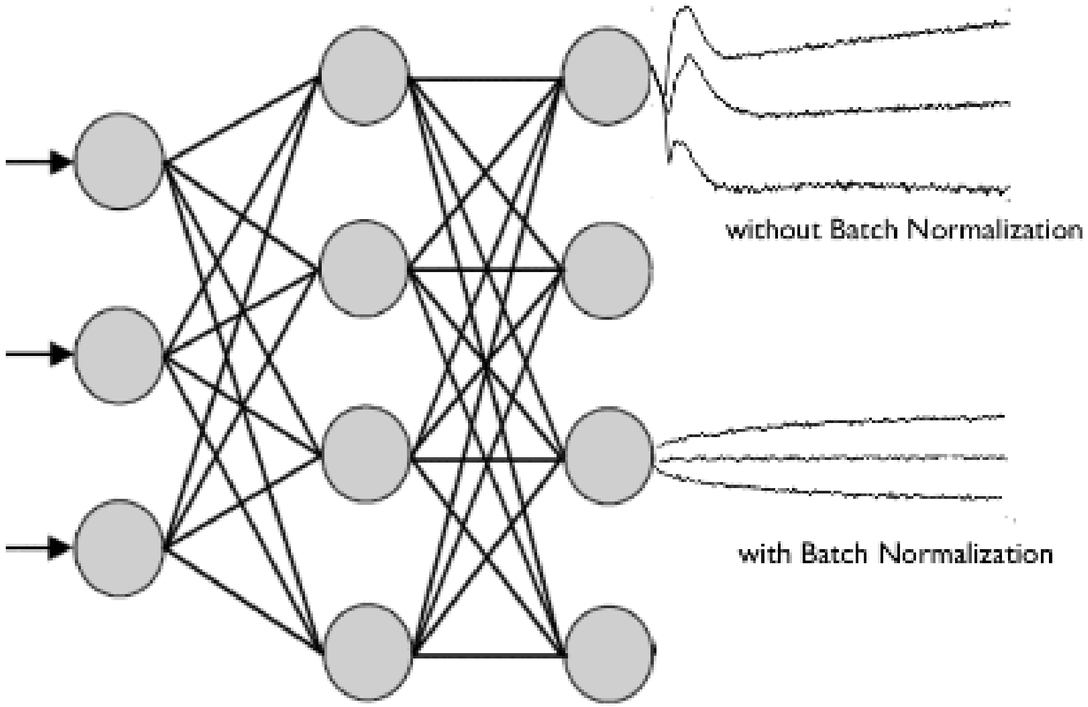
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



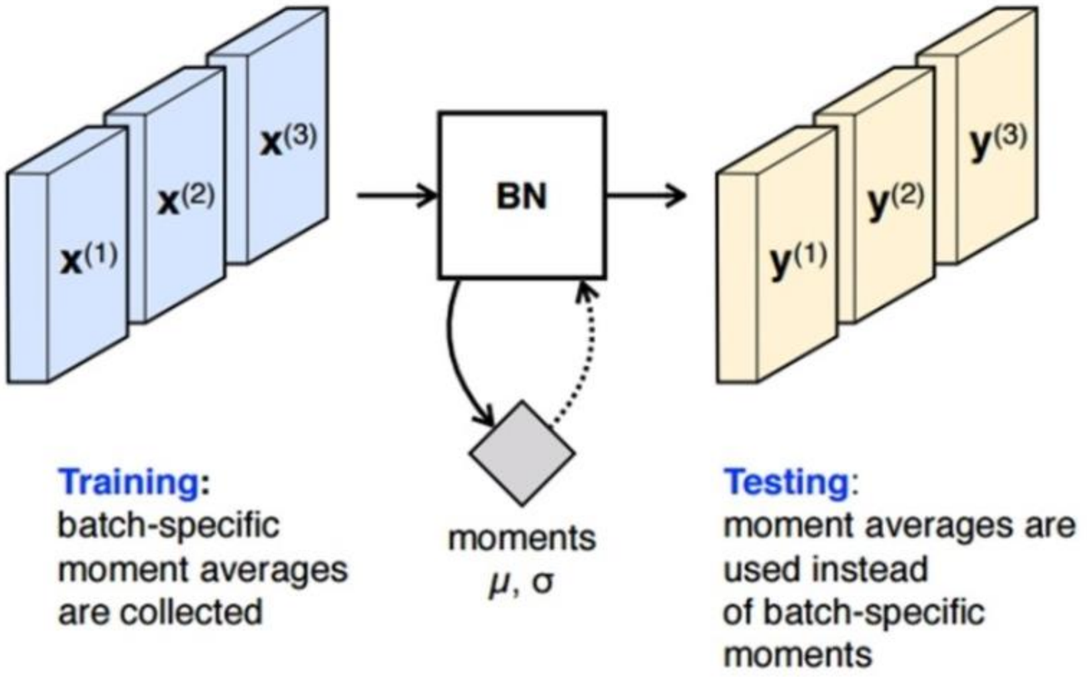
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Google 2015

Batch Normalization



◦출처 : https://eehoeskrap.tistory.com/430

Batch Normalization



주의!!!

BN은 Inference(test-time)

에 사용할 수 없음!

◦출처 : https://eehoeskrap.tistory.com/430

Regularization

- **Weight decay**

  regularization 목적(가중치 감소)도 있지만,

  gradient 계산 시, 계산값이 튀는 현상을 완화시켜,

  안정성 보장 목적으로 더 많이 사용됨

*L*1 regularization (*L*1 norm, Lasso)

$$C_{L1} = \frac{1}{2N}\sum_{n=1}^{N}(\hat{y}^n - y^n)^2 + \frac{\lambda}{N}\sum_{i=1}^{K}|w_i|$$

*L*2 regularization (Weight decay, *L*2 norm, Ridge)

$$C_{L2} = \frac{1}{2N}\sum_{n=1}^{N}(\hat{y}^n - y^n)^2 + \frac{\lambda}{2N}\sum_{i=1}^{K}w_i^2$$

$$w_i(t+1) = w_i(t) - \eta\frac{\partial C_{L2}}{\partial w_i} = \left(1 - \boxed{\frac{\eta\lambda}{N}}\right)w_i(t) - \eta\frac{\partial C}{\partial w_i}$$

weight decay

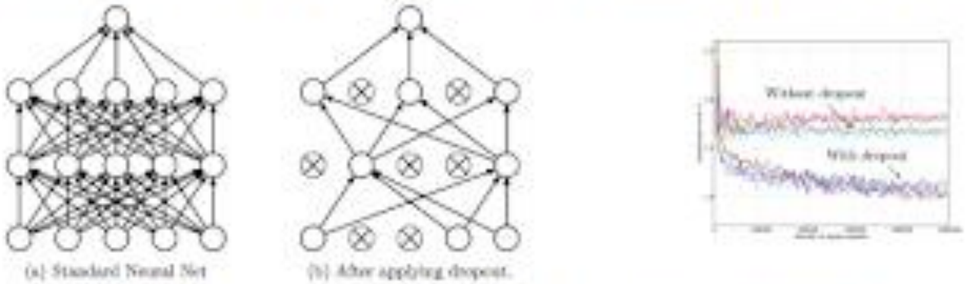
Regularization

- **Dropout**

  input layer와 hidden layer 사이의 노드 중 일정 비율을 제거하고 학습시키는 방법.

  앙상블과 유사한 결과를 만들어냄

- Dropout is a regularization technique for reducing overfitting in neural networks by preventing nodes from co-adapting too much on training data, introduced by Hinton 2012.
  - The gradient of each weight tells how it should change to reduce the cost, taking into account what all nodes are doing. So, nodes may change in a way that they fix up the mistakes of other nodes.
  - Co-adaptation refers to when different hidden nodes have such highly correlated behavior.
  - It is better to learn a general representation if nodes can detect features independently of each other.
- Randomly drop hidden nodes (along with their connections) in an iteration for each minibatch so that each node is omitted with probability p (independent of others) during only training. The omitted nodes and weights on this iteration are not updated during backpropagation.



(a) Standard Neural Net (b) After applying dropout.

Source: Dropout: A Simple Way to Prevent Neural Networks from Overfitting by Hinton *et al*

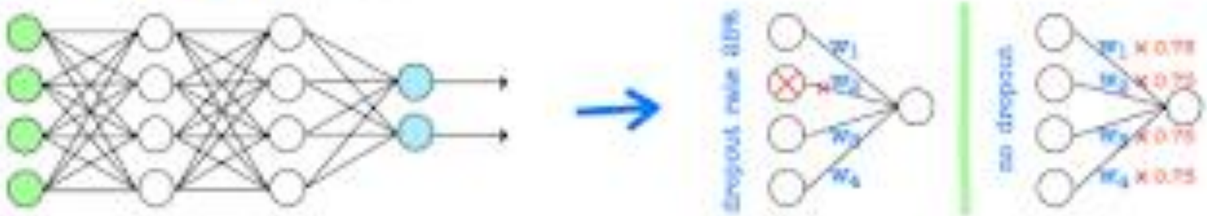
Regularization

Training

- For a minibatch, each node has dropout probability p (i.e. keeping probability $1-p$) so that we sample one from 2^N different thinned networks.
- Forward pass and backpropagation (gradient computing and weight update) are done only on this thinned network, ignoring the omitted nodes and their related weights.

Test

- For test data, we do not apply dropout, but take all the network weights multiplied by $1-p$ so that this un-thinned network with smaller weights approximates the average of the outputs of all these thinned networks.



Regularization

- **Data Augmentation**

data를 증강시키는 방법

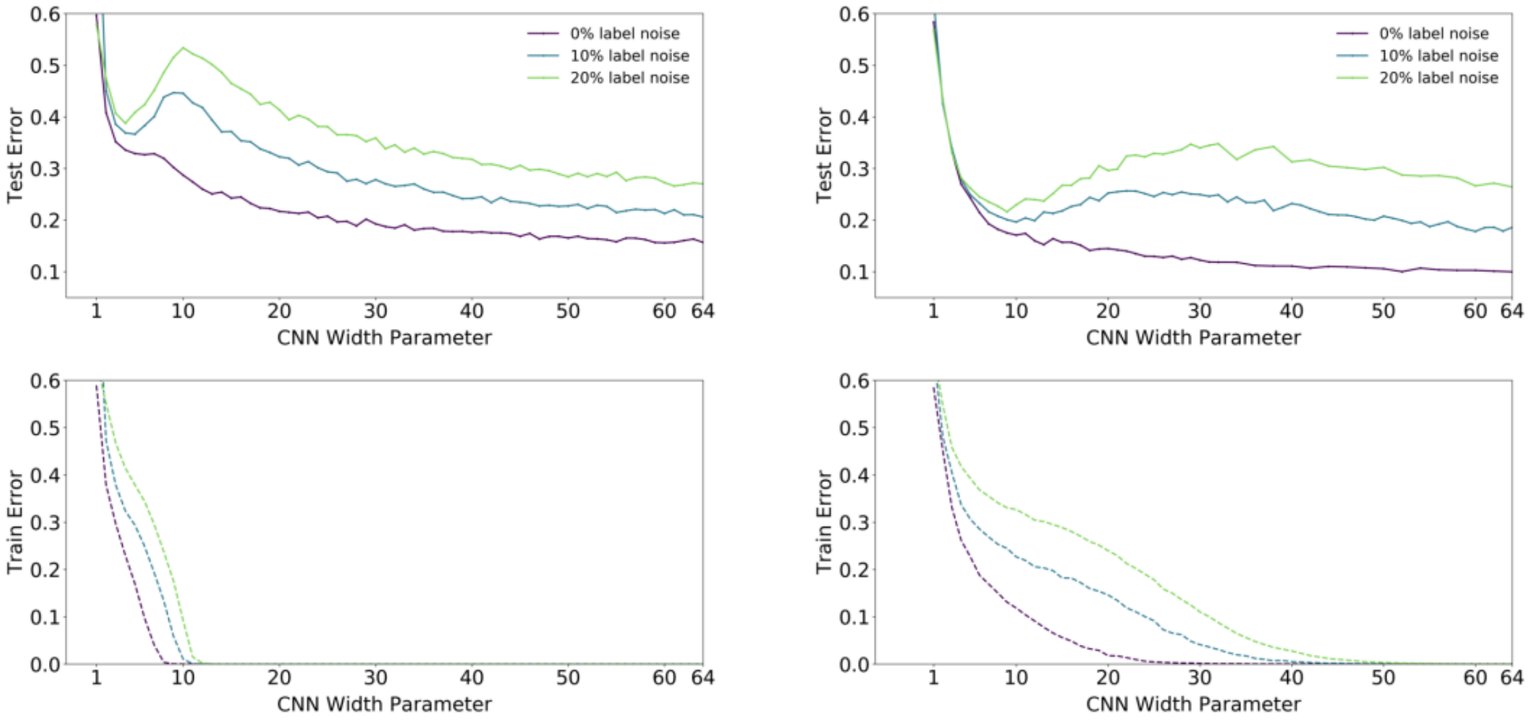
데이터 수가 적을 때, 활용하며, 잘 증강시키는 것이 key point

- **Early Stopping**

매 epoch마다 validation set에 대한 loss를 측정해 모델 종료 시점을 제어

학습 stop을 통해 과적합 방지

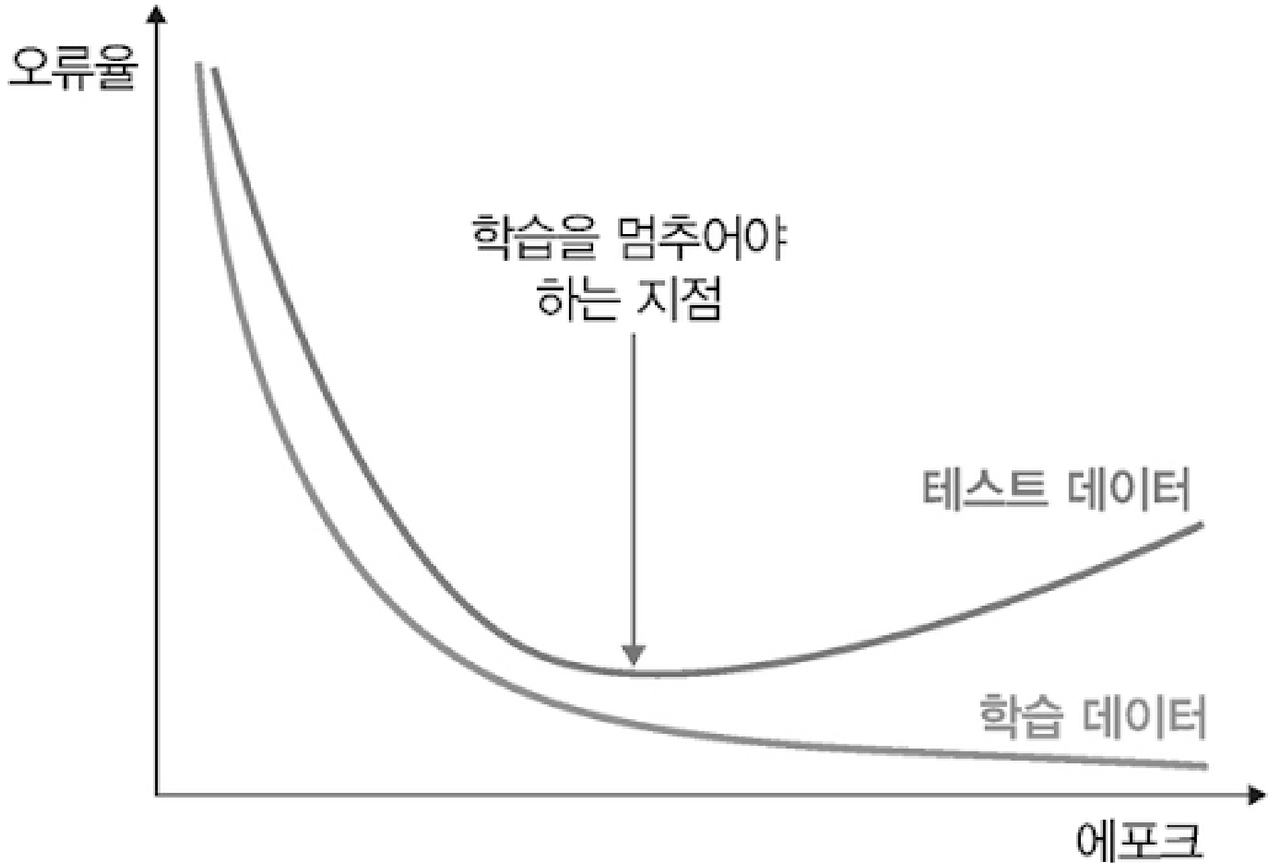
Regularization



(a) Without data augmentation.

(b) With data augmentation.

Regularization



Regularization

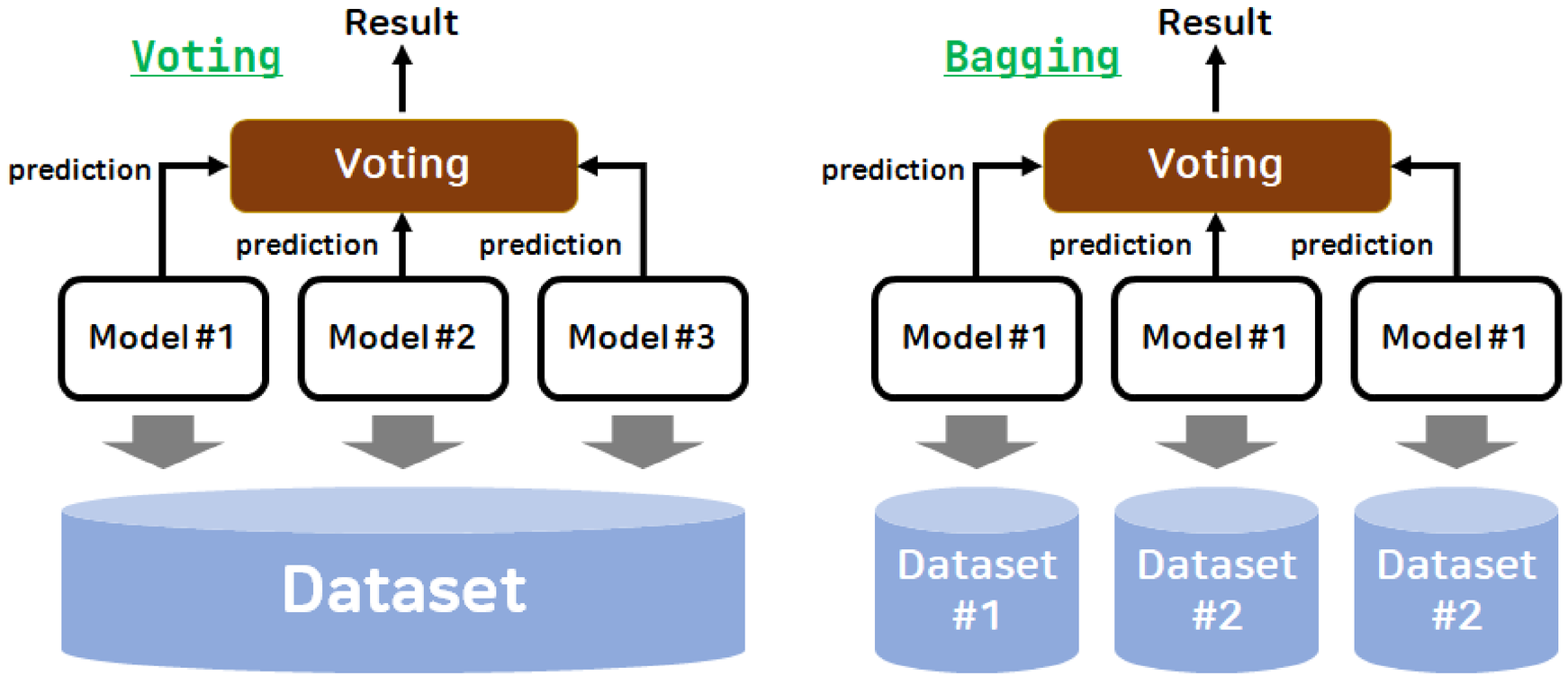
- **Ensemble**
  - **Voting**
    - 대표적으로 사용하는 기법
    - 여러 분류기가 투표를 통해 최종 예측 결과 결정
    - 다른 알고리즘 가진 분류기 -> 같은 데이터셋 기반 학습 + 결합
  - **Bagging**
    - voting과 유사하나, training set에서 중복을 허용해 sampling, 동일 알고리즘 사용

Regularization

- **Ensemble**
  - **Boosting**
    - cv, 자연어 처리에 널리 사용됨
    - 여러 약한 model 결합해 강력한 모델을 형성하는 반복 알고리즘
    - Neural Network 성능 향상에 사용
  - **Stacking**
    - 여러 모델을 학습시키고 결합시켜 단일 모델 성능 향상
    - 동일 training set에 대해 서로 다른 model을 학습 -> output결합
    - 개별 알고리즘 예측 데이터 기반 예측 수행

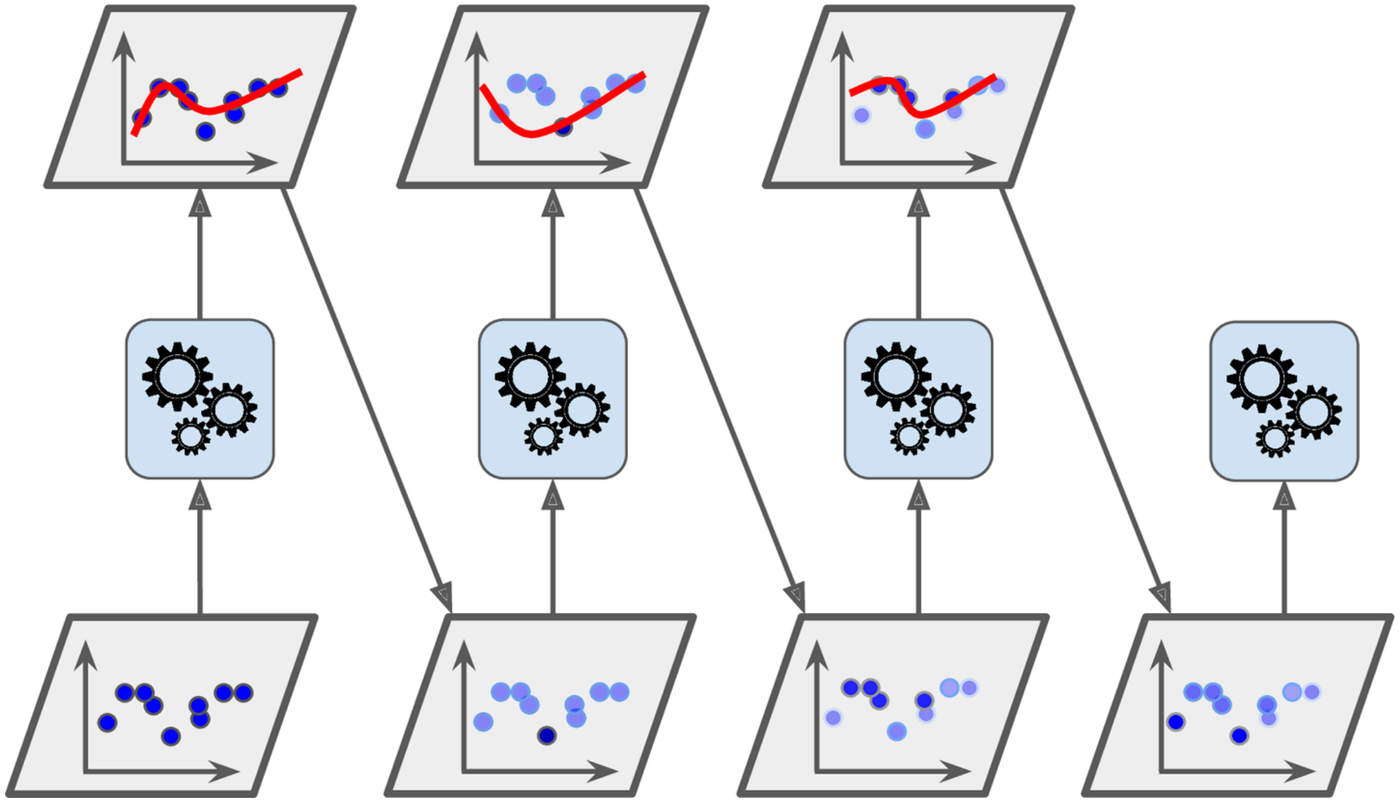
Regularization

- Ensemble



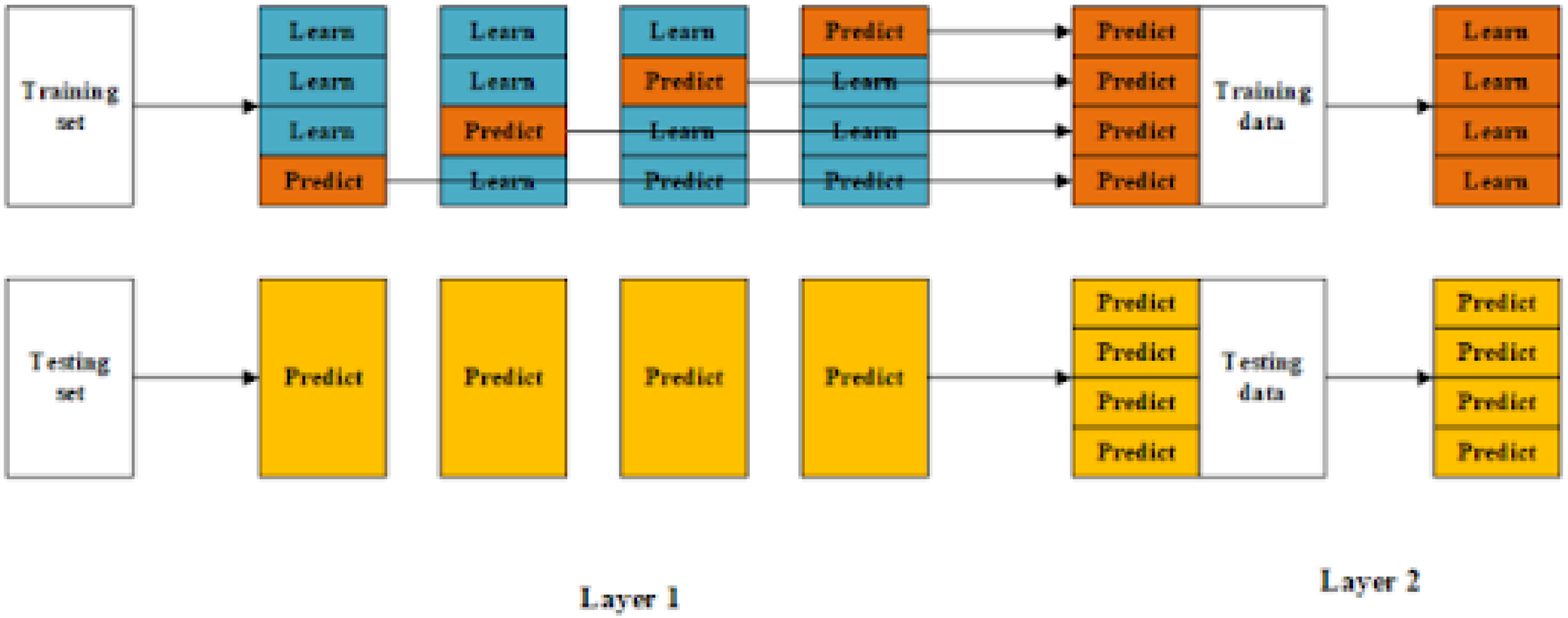
Regularization

- Ensemble



Regularization

- Ensemble



03
Initialization

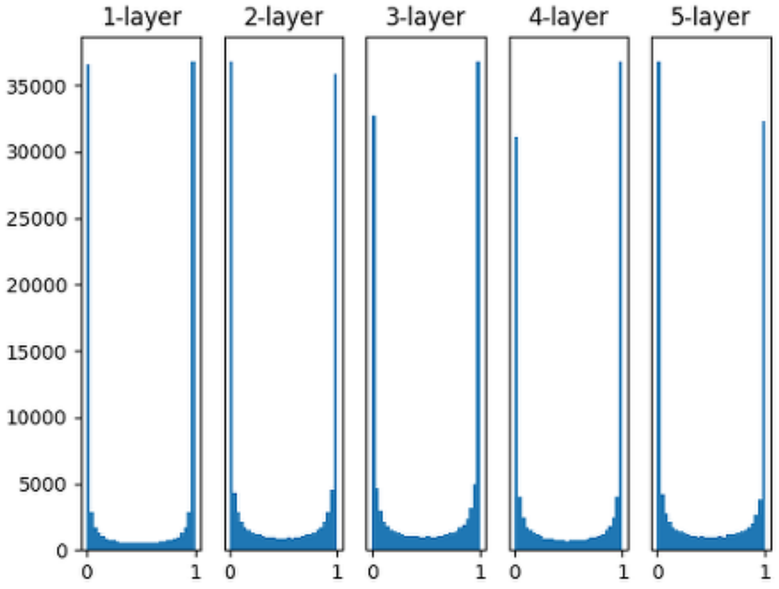
Initialization

- **Zero Initialization (= Same Initialization)**
  - 가중치 update가 대칭적으로 일어나도록 해 모델 성능 현저히 악화

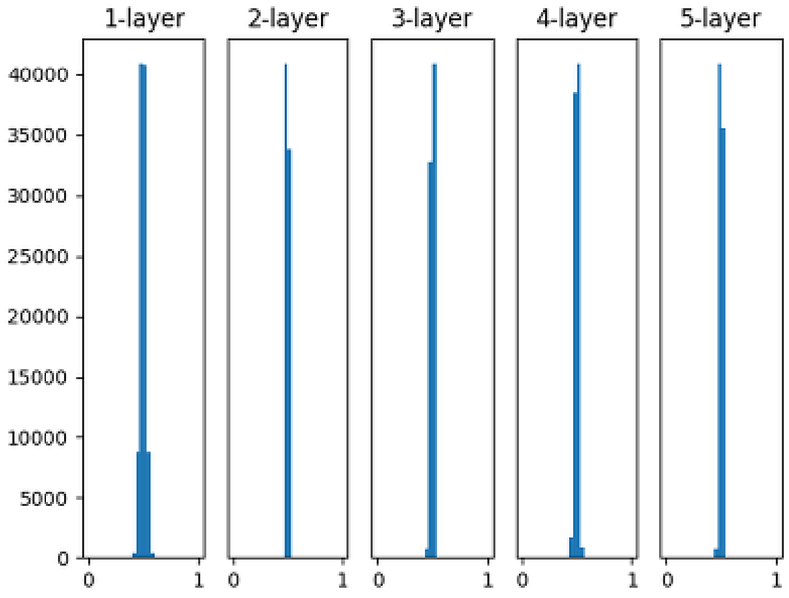
- **Random Initialization**
  - 지나치게 큰 분산은 활성화 값들을 0 or 1로 만들게 됨
  - 지나치게 작은 분산은 모든 활성화 값들을 유사한 값으로 만들게 됨

  -> 모델 다양성 악화

Initialization



Gaussian distribution + sigmoid



+ Using lower σ

Gradient Vanishing…!

03. Initialization

Advanced initialization

$$\frac{d\mathcal{L}}{dW^{(1)}} = \frac{dz^{(1)}}{dW^{(1)}}\frac{da^{(1)}}{dz^{(1)}}\frac{dz^{(2)}}{da^{(1)}}\frac{d\mathcal{L}}{dz^{(2)}}$$

$$\frac{d\mathcal{L}}{dW^{(1)}} = J_1 J_2 J_3 \dots J_n \frac{d\mathcal{L}}{dz^{(n)}}$$

If we multiply many many numbers together, what will we get?

If most of the numbers are < 1, we get 0

If most of the numbers are > 1, we get infinity

We only get a reasonable answer if the numbers are all close to 1!

for each $W^{(i)}$, we can write: $W^{(i)} = U^{(i)}\Lambda^{(i)}V^{(i)}$ e.g., using singular value decomposition

$W^{(i)} \leftarrow U^{(i)}V^{(i)}$ just need to force this to be identity matrix

even simpler:

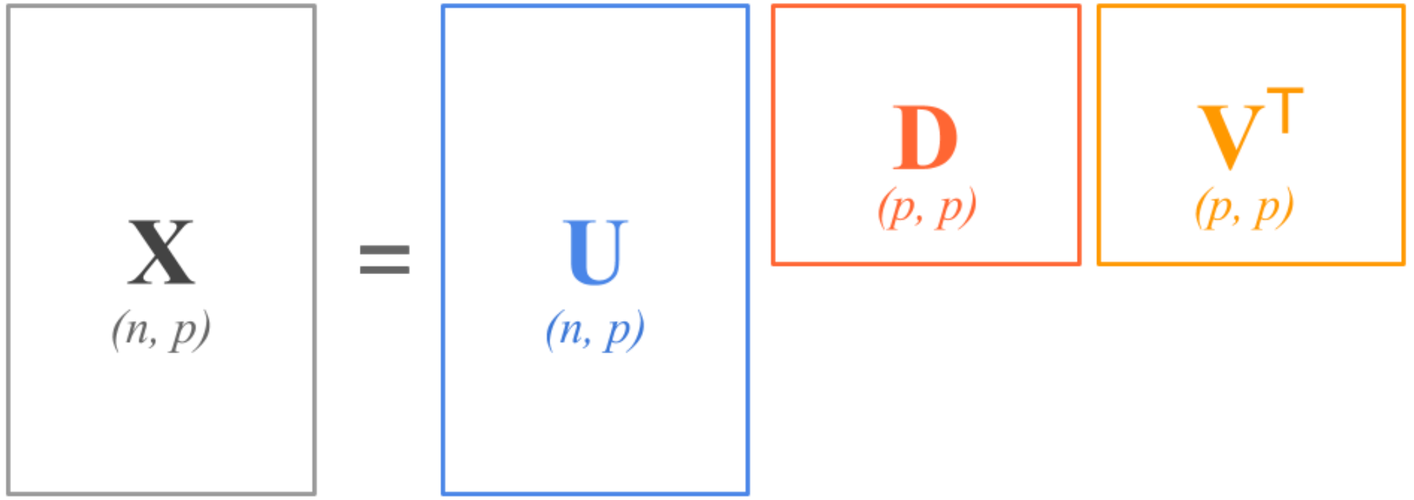
```
a = get_rng().normal(0.0, 1.0, flat_shape)
u, _, v = np.linalg.svd(a, full_matrices=False)
# pick the one with the correct shape
q = u if u.shape == flat_shape else v
```

arbitrary random matrix (doesn't really matter how)

needed if non-square

guaranteed orthonormal

03. Initialization



$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

where:

- $\mathbf{U}$ is a $n \times p$ column *orthonormal* matrix containing the **left singular vectors**.
- $\mathbf{D}$ is a $p \times p$ *diagonal* matrix containing the **singular values** of $\mathbf{X}$.
- $\mathbf{V}$ is a $p \times p$ column *orthonormal* matrix containing the **right singular vectors**.

Initialization 종류

- LeCun Normal Initialization

$$W \sim N\left(0, \sqrt{\frac{1}{n_{\text{in}}}}^2\right)$$

LeCun Uniform Initialization

$$W \sim U\left[-\sqrt{\frac{1}{n_{\text{in}}}}, \sqrt{\frac{1}{n_{\text{in}}}}\right]$$

- Xavier Normal Initialization

$$W \sim N\left(0, \sqrt{\frac{2}{n_{\text{in}}+n_{\text{out}}}}^2\right)$$

Xavier Uniform Initialization

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

- He Normal Initialization

$$W \sim N\left(0, \sqrt{\frac{2}{n_{\text{in}}}}^2\right)$$

He Uniform Initialization

$$W \sim U\left[-\sqrt{\frac{6}{n_{\text{in}}}}, \sqrt{\frac{6}{n_{\text{in}}}}\right]$$

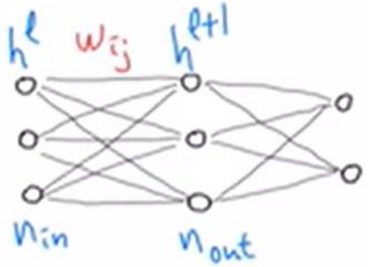
- n_{in} node number of the previous layer, and n_{out} of the next layer.

Xavier Normal Initialization

- We want to initialize weights satisfying following conditions:

  $\mathrm{Var}[h^l] = \mathrm{Var}[h^{l+1}]$ 'forward'

  $\mathrm{Var}\left[\frac{\partial \mathrm{Cost}}{\partial h^l}\right] = \mathrm{Var}\left[\frac{\partial \mathrm{Cost}}{\partial h^{l+1}}\right]$ 'backward'



- Assumptions in each layer l
  - Linear Regime: activation function $f(x) = x$ for near $x = 0$, and so $f'(0) = 1$.
  - All weights w_{ij}^l are independent and $\mathrm{Var}[w_{ij}^l] = \mathrm{Var}[w^l]$.
  - All elements of $\boldsymbol{h}^l = (h_1^l, \ldots, h_{n_l}^l)^T$ are independent and $\mathrm{Var}[h_i^l] = \mathrm{Var}[h^l]$.
  - All weights and elements of an activation vector are independent of each other.
  - Each of their distributions is symmetric around 0, and so $\mathrm{E}[w_{ij}^l] = \mathrm{E}[h_i^l] = 0$

Xavier Normal Initialization

- Forward Propagation satisfying $\mathrm{Var}[h^l] = \mathrm{Var}[h^{l+1}]$

  $h_j^{l+1} = \sum_{i=1}^{n_l} w_{ij}^l h_i^l \;\Rightarrow\; \mathrm{Var}[h^{l+1}] = n_l \mathrm{Var}[w^l h^l] = n_l \mathrm{Var}[w^l]\mathrm{Var}[h^l] \;\Rightarrow\; n_l \mathrm{Var}[w^l] = 1$

  Note that $\mathrm{Var}[XY] = \mathrm{E}[X]^2 \mathrm{Var}[Y] + \mathrm{E}[Y]^2 \mathrm{Var}[X] + \mathrm{Var}[X]\mathrm{Var}[Y]$ and $\mathrm{E}[w_{ij}^l] = \mathrm{E}[h_i^l] = 0$.

- Backward Propagation satisfying $\mathrm{Var}\left[\frac{\partial \mathrm{Cost}}{\partial h^l}\right] = \mathrm{Var}\left[\frac{\partial \mathrm{Cost}}{\partial h^{l+1}}\right]$

  $\frac{\partial \mathrm{Cost}}{\partial h_i^l} = \sum_{j=1}^{n_{l+1}} w_{ij}^l f'(\cdot) \frac{\partial \mathrm{Cost}}{\partial h_j^{l+1}} \;\Rightarrow\; \mathrm{Var}\left[\frac{\partial \mathrm{Cost}}{\partial h^l}\right] = n_{l+1}\mathrm{Var}[w^l]\mathrm{Var}\left[\frac{\partial \mathrm{Cost}}{\partial h^{l+1}}\right] \;\Rightarrow\; n_{l+1}\mathrm{Var}[w^l] = 1$

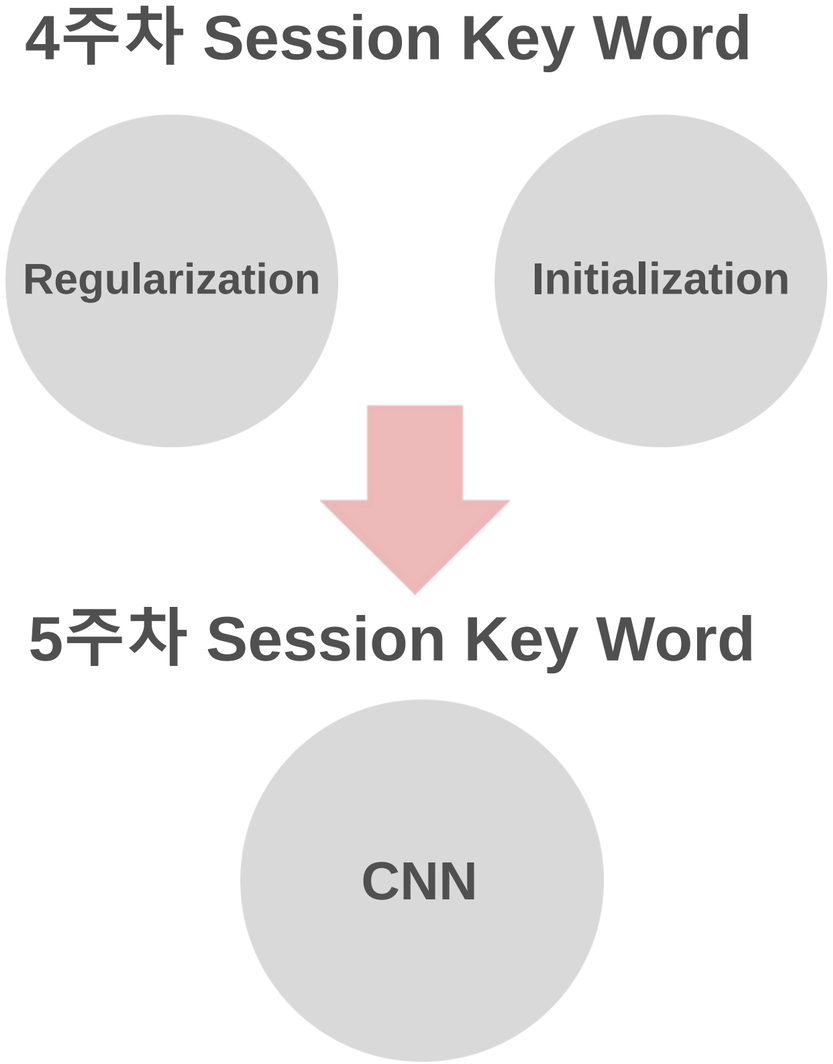
  Note that $f'(0) = 1$ and $\mathrm{E}[w_{ij}^l] = \mathrm{E}\left[\frac{\partial \mathrm{Cost}}{\partial h_j^{l+1}}\right] = 0$.

- Thus, in each layer l, we initialize weights w_{ij}^l satisfying $\mathrm{Var}[w_{ij}^l] = \frac{2}{n_l + n_{l+1}} = \frac{2}{n_{\text{in}} + n_{\text{out}}}$.

04

다음 주차 예고

04. 다음 주차 예고



04. 다음 주차 예고

4주차 과제

공부 과제

d2l

6장

CNN

오승상 딥러닝
16, 17, 18강

코드 과제

슬랙 공지로
추후 전달 예정

04. 다음 주차 예고

KUBIG CONTEST 관련 공지

8월 28일 목요일 contest 진행
아래 주제는 참고사항이며, 원하는 주제로 진행해도 됩니다 !
*유의 : github 코드 제출 및 발표 ppt 제출 필수

<지난 기수 DL Session 주제>

난독화된 한글 리뷰 복원(Dacon 경진대회 참여)

부동산 허위매물 분류(MLP, TabPFN적용)

저해상도 조류 분류 문제 해결 (Detection-Classification)

Thank You

2025 DL Session 4주차 끝!

KOREA
UNIVERSITY