신경망에서의 과적합

Sang Yup Lee

신경망에서의 과적합

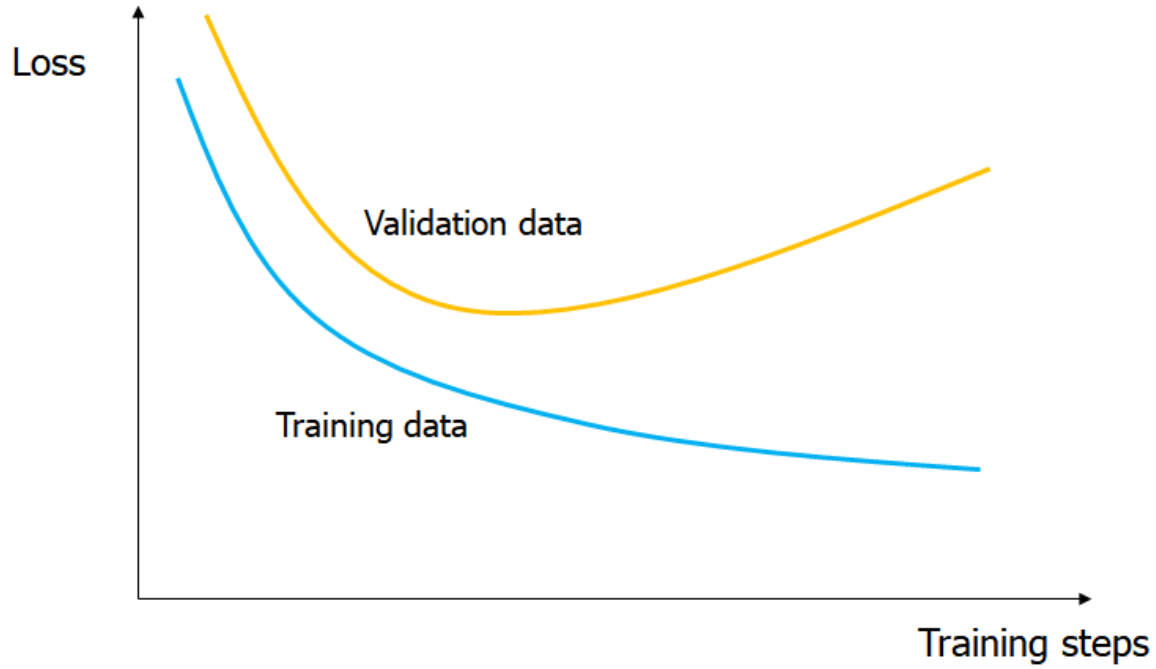
- 과적합 (Overfitting) 문제
  - What is it?
    - 학습 데이터는 잘 설명하는 반면에 (학습에 사용되지 않은) 새로운 데이터는 설명을 못하는 문제
    - 학습 데이터를 얼마나 잘 설명하는지 보다, 학습에 사용되지 않은 새로운 데이터를 얼마나 잘 설명하는지가 더 중요
  - Why does it happen?
    - 모형이 너무 복잡해서, 즉, 모형에 존재하는 파라미터의 수가 많아서, 모형이 학습 데이터에 너무 민감하게 반응하기 때문
    - 즉, 파라미터가 많은 딥러닝 모형에서 발생하기 쉽다.

신경망에서의 과적합

- 과적합 (Overfitting) 문제 (cont'd)
  - How can we know it is happening?
    - 이를 위해 학습 데이터의 일부를 과적합 문제 체크를 위해 사용
    - 이러한 데이터셋을 validation (검증) dataset이라고 함
    - 즉, 원래 학습 데이터 = 최종 학습 데이터 + 검증 데이터
    - 최종 학습 데이터만 학습에 사용되고, 검증 데이터는 학습에 사용되지 않는다.
    - 모형의 성능이 학습에 사용된 데이터에 대해서는 좋아지나, 검증 데이터에 대해서는 나빠지는 경우 => 과적합 의심
  - 주요 해결책
    - L1/L2 Regularization, Dropout, Early Stopping, Batch Normalization (Batchnorm), Layer Normalization 등

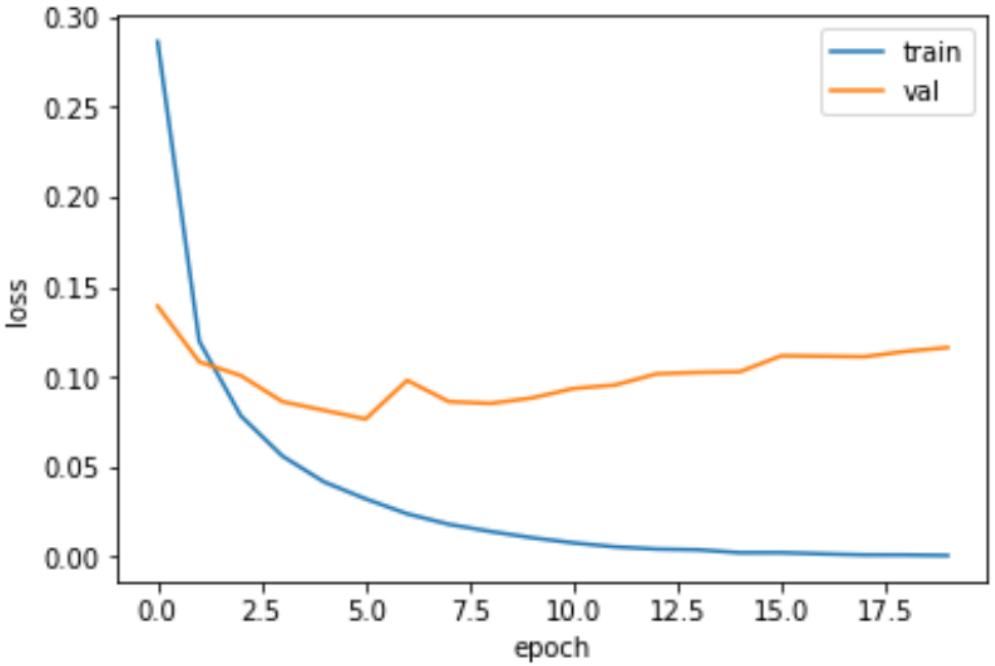
신경망에서의 과적합

- 과적합 (Overfitting) 문제 (cont'd)
  - How can we known it is happening?



신경망에서의 과적합

- 과적합 (Overfitting) 문제 (cont'd)
  - Example code
    - See "FNN_housing_example_overfitting.ipynb"



Regularization

과적합 해소 방법

- Regularization
  - 가중치의 최적값을 줄임=> 각 노드의 영향을 줄임 => 일반화 (generalization 기능을 높임)
  - 주요 방법
    - L1 and L2
    - $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ 에 대한 Lp-Norm: $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{k} |x_i|^p\right)^{\frac{1}{p}}$
    - L1 norm: $\|\mathbf{x}\|_1 = \left(\sum_{i=1}^{k} |x_i|^1\right)^{\frac{1}{1}} = \sum_{i=1}^{k} |x_i| = |x_1| + |x_2| + \cdots |x_k|$
    - L2 norm: $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{k} |x_i|^2\right)^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{k} |x_i|^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_k^2}$
    - L1과 L2 regularizer 사용

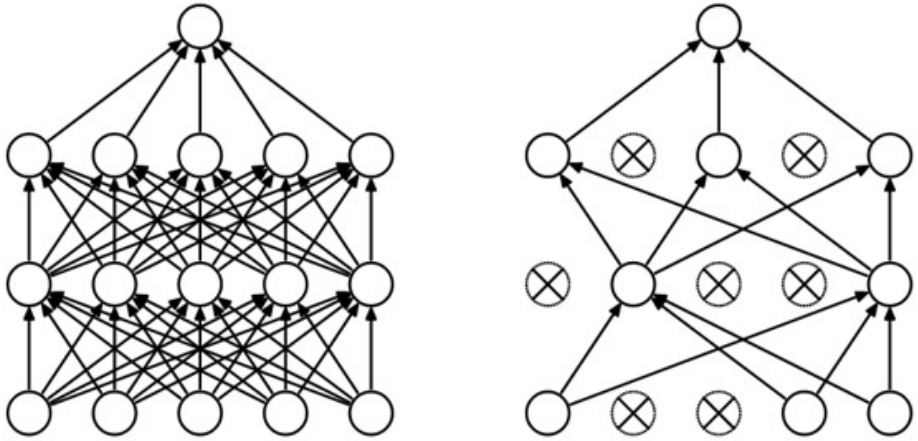
과적합 해소 방법

- Regularization (cont'd)
  - L1 regularization에서의 새로운 비용함수: $\mathrm{J}(\mathbf{w}) + \lambda\|\boldsymbol{w}\|_1$
    - 여기서 $\|\boldsymbol{w}\|_1 = \sum_{i=1}^{k}|w_i| = |w_1| + |w_2| + \cdots |w_k|$이며, λ /람다/는 penalty strength, λ의 값을 크게할수록 가중치의 값이 더 많이 줄어드는 효과
  - L2 regularization에서의 새로운 비용함수: $\mathrm{J}(\mathbf{w}) + \lambda\|\boldsymbol{w}\|_2^2$
    - 여기서 $\|\boldsymbol{w}\|_2^2$ 은 $\sum_{i=1}^{k} w_i^2 = w_1^2 + w_2^2 + \cdots + w_k^2$.
  - L1과 L2 방법의 차이를 비교하는 것이 중요
  - 신경망의 경우
    - 각 layer의 가중치에 적용
    - https://keras.io/api/layers/regularizers/
  - 파이썬 코드:
    - FNN_housing_example_l1_l2_regularization.ipynb 참고

Dropout

과적합 해소 방법

- Dropout
  - 일부의 노드를 제외하고 학습 ⇒ 즉, 모형을 좀 더 단순하게 만든다!
  - Random 하게 일부의 노드 제외
    - 이렇게 하면 generalization 성능이 더 높아짐

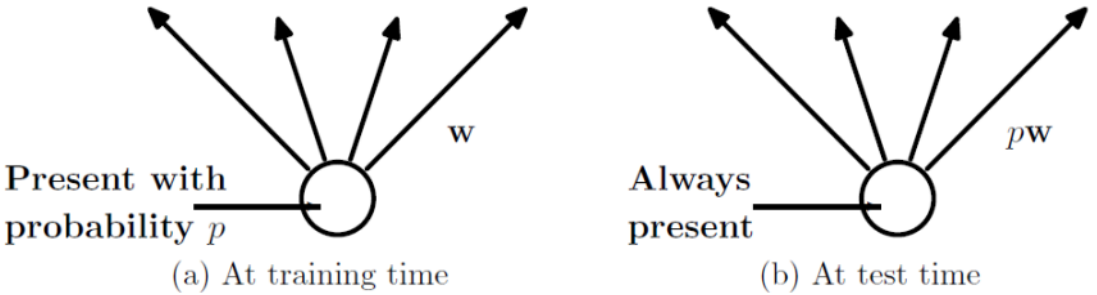


(a) Standard Neural Net (b) After applying dropout.

Srivastava et al. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929–1958.

과적합 해소 방법

- Dropout (cont'd)
  - Training Phase: For each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, *p*, of nodes (and corresponding activations). => 학습을 할 때 마다 다른 형태의 네트워크가 사용된다는 것을 의미
  - Test 과정에서는 모든 노드들을 모두 사용하게 됨. 이렇게 되면 각 노드에 입력되는 값이 커지는 문제 발생 => 이를 해결하기 위해서 원래 파라미터 값에 p를 곱하여 사용



(a) At training time (b) At test time

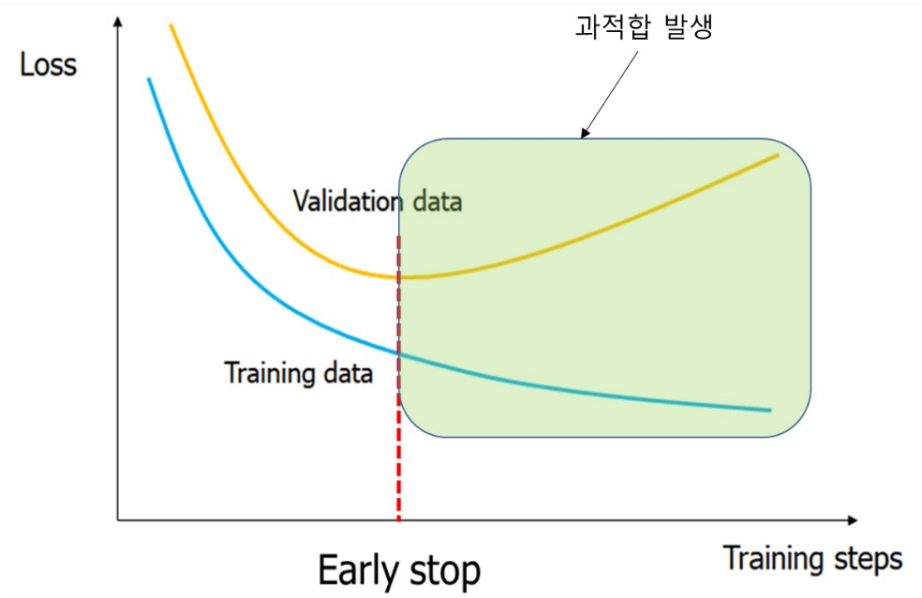
과적합 해소 방법

- Dropout (cont'd)
  - In Keras
    - https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout
  - Python code: FNN_housing_example_dropout.ipynb

Early stopping

과적합 해소 방법

- Early stopping
  - 학습을 조기에 종료하는 방법
  - Training dataset에 대한 에러는 줄어드나 validation set에 대한 에러가 증가하는 경우 => 과적합 신호 => 따라서 그 때 학습을 멈춘다!!



과적합 해소 방법

- Early stopping (cont'd)
  - In Keras
    - https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
    - https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
  - Python code: FNN_housing_example_earlystopping.ipynb
  - 이를 위해서 callbacks 모듈을 사용
    - EarlyStopping 클래스: Early stopping을 위한 클래스
    - ModelCheckpoint 클래스: Early stopping의 결과를 추출하기 위한 클래스

과적합 해소 방법

- callbacks module
  - 학습 과정을 모니터링 한다.
    - 즉, 학습을 하는 동안 loss와 accuracy 등이 어떻게 변화하는지를 모니터링 한다.
  - Metrics (예, loss 또는 accuracy 등)에 따라 특정한 작업을 수행한다.
    - 예)
      - Early stop
      - 모형 저장

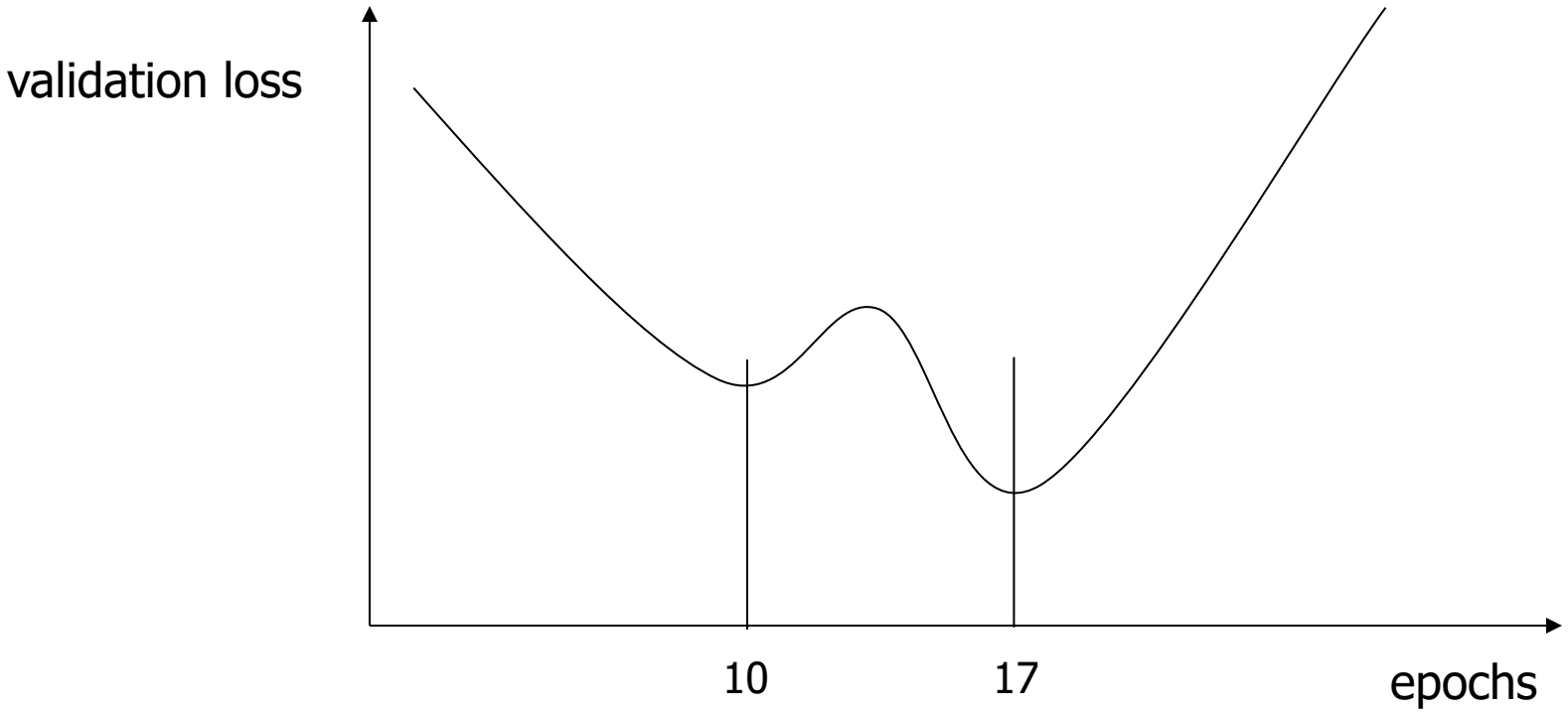
과적합 해소 방법

- Keras의 EarlyStopping 클래스 작동 방식
  - Global minimum 에서 멈추는 것이 아니다!
  - 첫번째로 나오는 local minimum 에서 멈춘다.
  - 즉, 첫번째로 비용함수의 값이 다시 증가하는 지점에서 멈추게 된다.

과적합 해소 방법

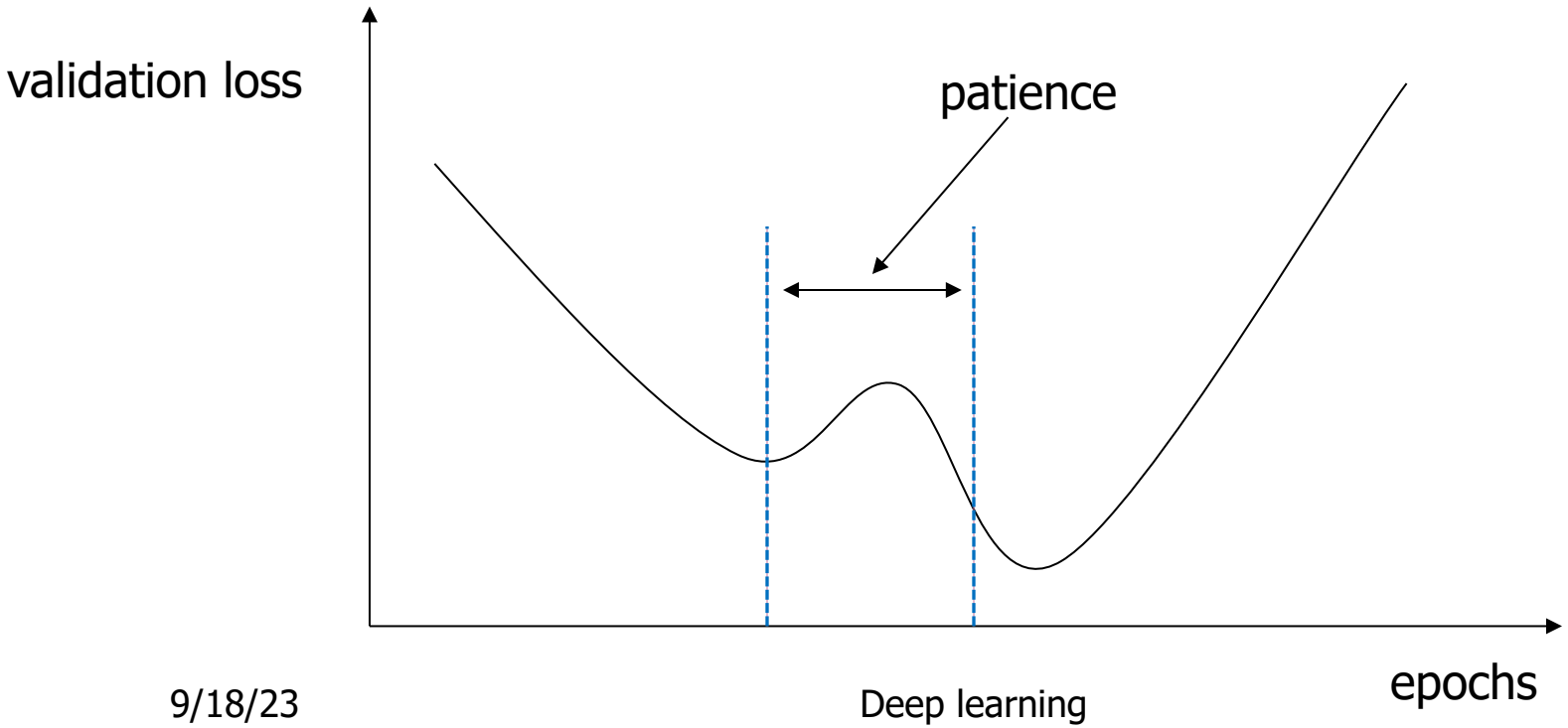
- Example

EarlyStopping은 10에서 멈춘다



과적합 해소 방법

- Early stopping
  - The "patience" parameter
    - Loss의 값이 local minimum 으로 부터 얼마나 지나서 종료하는지를 결정

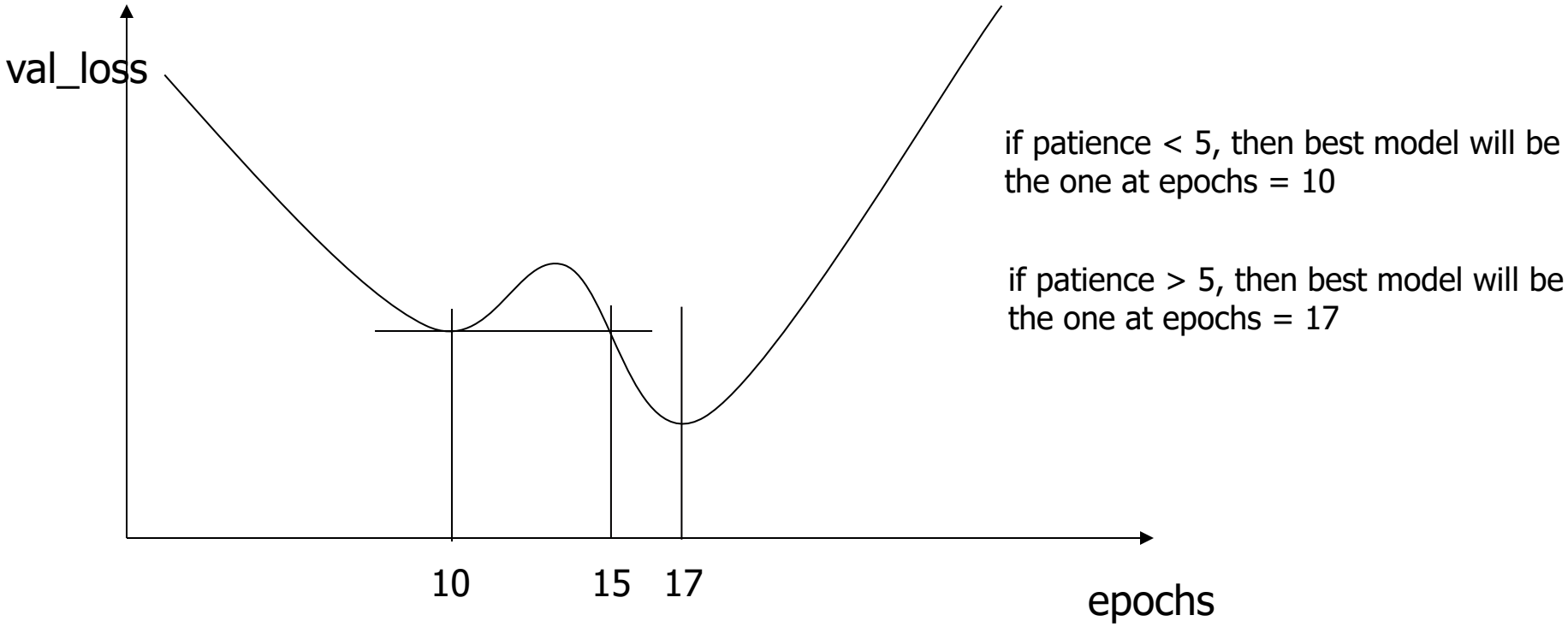


과적합 해소 방법

- Early stopping (cont'd)
  - Then why use the 'patience' parameter?
    - 첫 local minimum 지점이 global minimum이 아닐 수 있다.
      - 다음 슬라이드 그림 참고
    - 그 이후에 loss가 다시 줄어드지 않는지 기다려 보자!
    - patience = 0으로 하면 val_loss가 증가하는 지점에서 바로 멈춘다.

과적합 해소 방법

- Early stopping (cont'd)



if patience < 5, then best model will be the one at epochs = 10

if patience > 5, then best model will be the one at epochs = 17

과적합 해소 방법

```
375/375 [==============================] - 4s 10ms/step - loss: 0.0547 - accuracy: 0.9835 - val_loss: 0.0827 - val_accuracy: 0.9759
Epoch 5/20
375/375 [==============================] - 4s 10ms/step - loss: 0.0409 - accuracy: 0.9882 - val_loss: 0.0856 - val_accuracy: 0.9753
Epoch 6/20
375/375 [==============================] - 4s 9ms/step - loss: 0.0303 - accuracy: 0.9912 - val_loss: 0.0811 - val_accuracy: 0.9787
Epoch 7/20
375/375 [==============================] - 4s 10ms/step - loss: 0.0230 - accuracy: 0.9930 - val_loss: 0.0983 - val_accuracy: 0.9723
Epoch 8/20
375/375 [==============================] - 4s 10ms/step - loss: 0.0176 - accuracy: 0.9949 - val_loss: 0.0823 - val_accuracy: 0.9787
Epoch 9/20
375/375 [==============================] - 4s 9ms/step - loss: 0.0130 - accuracy: 0.9965 - val_loss: 0.0888 - val_accuracy: 0.9778
Epoch 10/20
375/375 [==============================] - 4s 11ms/step - loss: 0.0104 - accuracy: 0.9972 - val_loss: 0.0852 - val_accuracy: 0.9796
Epoch 11/20
375/375 [==============================] - 4s 11ms/step - loss: 0.0084 - accuracy: 0.9976 - val_loss: 0.0923 - val_accuracy: 0.9787
Epoch 00011: early stopping
```

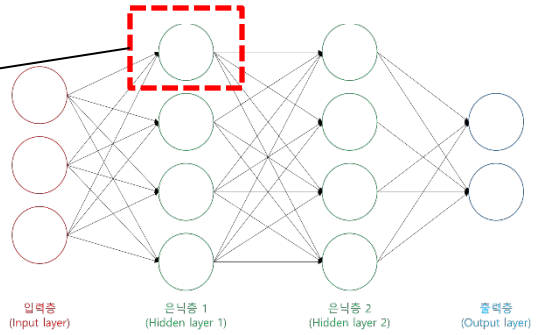
patience = 5 인 경우
Epoch = 6 인 지점에서의 val_loss 값이 최저 (즉, 0.0811), 그 이후 다시 증가
그 다음 5번의 epoch에서 0.0811 보다 작은 val_loss 없음 → 따라서 학습 종료

Batch normalization

Batch normalization

- Batch normalization
  - Proposed by Ioffe et al. in 2015 mainly to mitigate the "internal covariate shift" problem

이 노드에 입력되는 값들
⇒ 배치마다 다름



배치 별 관측치의 수 = m

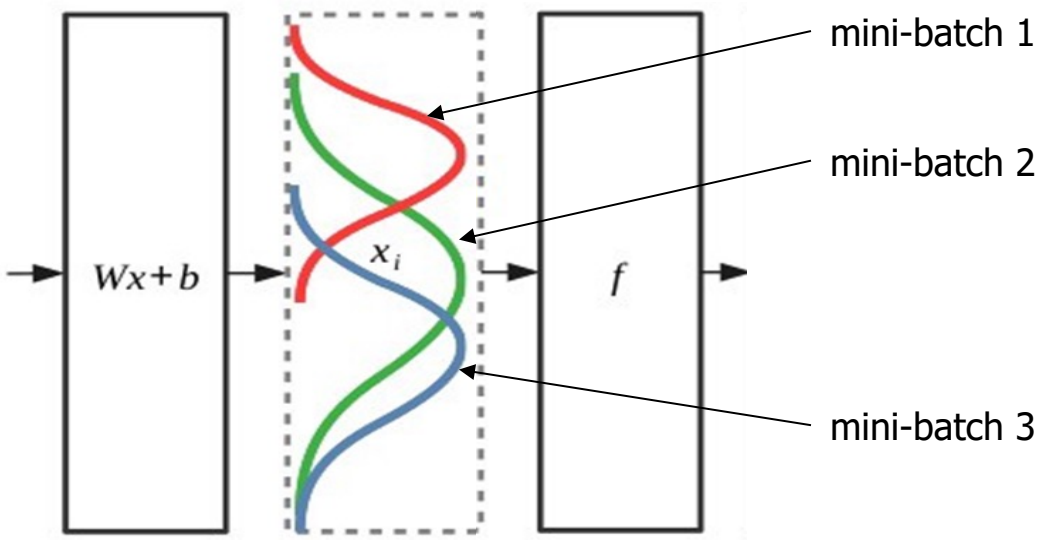
- 배치 1: $z_{1,1}, z_{1,2}, \dots, z_{1,m}$
- 배치 2: $z_{1,1}, z_{1,2}, \dots, z_{1,m}$
- …
- 배치 j: $z_{j,1}, z_{j,2}, \dots, z_{j,m}$
- …

각 배치별 입력되는 값들의 분포가 다름
(평균과 분산이 다름)

Ioffe, Sergey; Szegedy, Christian (2015). *"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*

Batch normalization

- Internal Covariate Shift
  - 학습이 진행됨에 따라 각 노드의 가중치 값이 달라짐 ⇒ 이로 인해 각 노드에 입력되는 (혹은 출력되는) 값들의 분포가 미니배치마다 달라짐



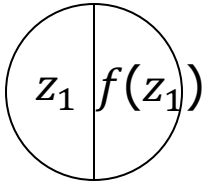
Ioffe, Sergey; Szegedy, Christian (2015). *"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*

Batch normalization

- Batch normalization
  - 신경망에서는 mini-batch 별로 표준화 (standardization) 방법 사용
    => 즉, 평균을 빼고 표준편차로 나눠준다!
    - To increase the stability of a neural network, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.
    - 이렇게만 하면 비선형 관계를 잘 학습하지 못한다는 문제 발생
      - *Note that simply normalizing each input of a layer may change what the layer can represent. For instance, normalizing the inputs of a sigmoid would constrain them to the linear regime of the nonlinearity.*
    - 이러한 문제를 해결하기 위해서 Scaling and shifting
  - 주요 효과
    - 학습 속도 개선
    - 과적합 문제 개선

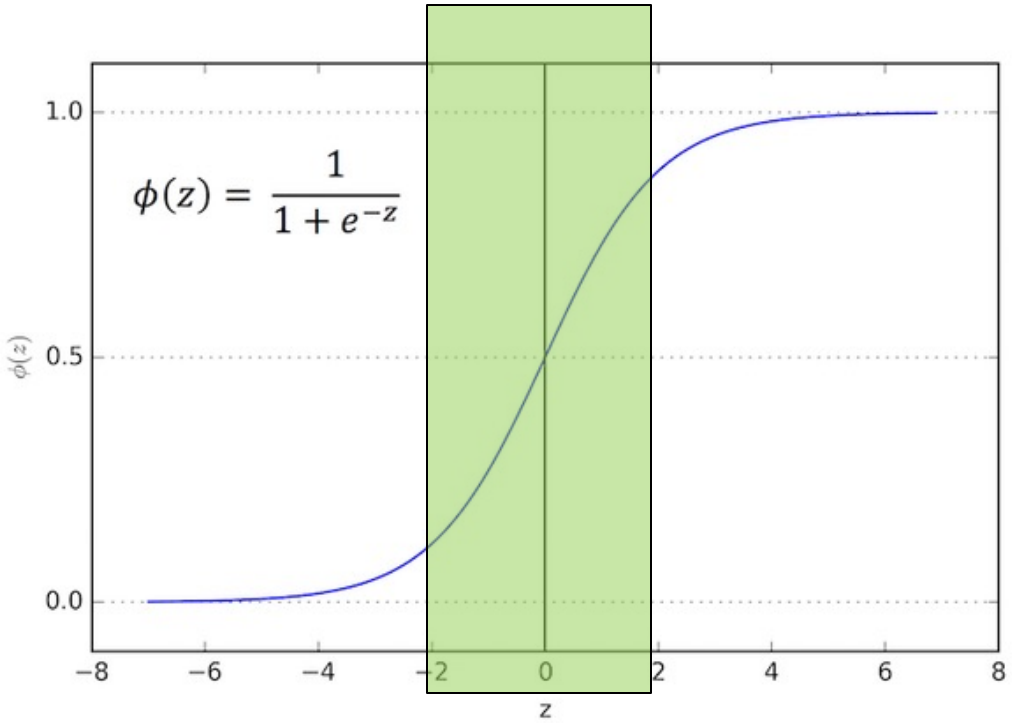
Batch normalization

- 그런데 표준화만 해주게 되면 비선형 관계를 잘 설명 못하는 문제 발생
  - 왜 이러한 문제가 발생하는가?
  - 보통 은닉층의 입력값에 batchnorm 적용



  - $\hat{z}_1 = \frac{z_1 - \mu}{\sqrt{\sigma^2 + \varepsilon}}$, then $\hat{z}_1 \sim N(0,1)$
  - $\hat{z}_1$이 입력값으로 사용됨
  - f() 가 sigmoid function인 경우, what would happen?

Batch normalization



비선형 관계를 잘 설명하지 못함!!

Batch normalization

- Batch normalization (cont'd)
  - 이를 해결하기 위해 scaling & shifting

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

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

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

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

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

특정 노드에 입력되는 i번째 관측치에 대한 값

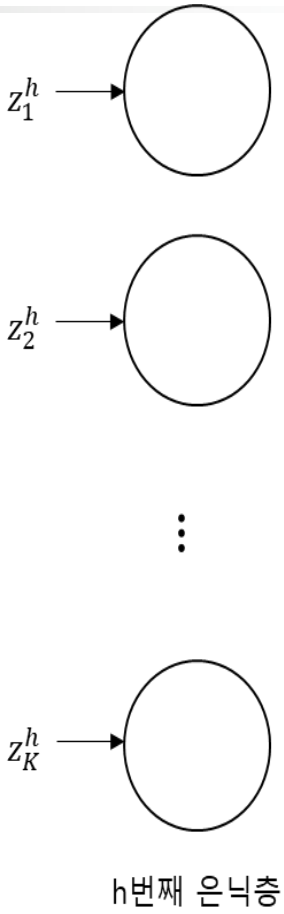
γ와 β는 학습 된다.

Batch normalization

- Batch normalization (cont'd)
  - In Keras
    - https://keras.io/api/layers/normalization_layers/batch_normalization/
    - https://www.machinecurve.com/index.php/2020/01/15/how-to-use-batch-normalization-with-keras/#batch-normalization-normalizes-layer-inputs-on-a-per-feature-basis
  - Python code: FNN_housing_example_batch_layer_norm.ipynb
  - 일반적으로 이미지 데이터에 적용

Layer normalization

- Layer normalization
  - 일반적으로 텍스트 데이터에 적용
  - 하나의 관측치에 대해서 특정 은닉층에 존재하는 모든 노드들에 들어가는 값들을 이용해서 표준화하는 방법



Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Layer normalization

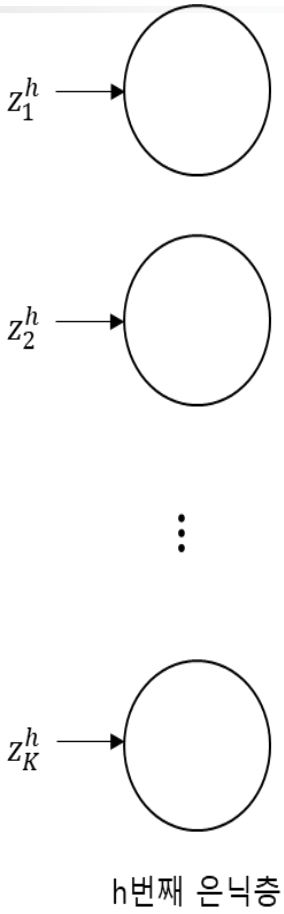
- Layer normalization (cont'd)

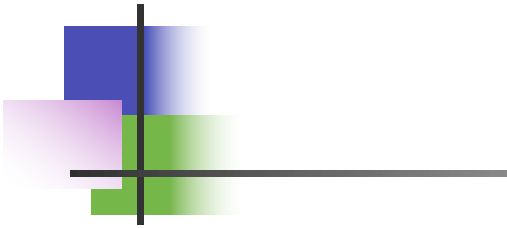
$$\mu^h = \frac{1}{K}\sum_{i=1}^{K} z_i^h, \qquad \sigma^h = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\left(z_i^h - \mu^h\right)^2}$$

$$x_i^h = \frac{z_i^h - \mu^h}{\sigma^h}$$

  - scaling and shifting

$$\tilde{x}_i^h = g_i x_i^h + b_i$$





Q & A