

# **M7. Regularization**

## Overfitting 방지하기

- Early stopping
- Augmentation
- L2 Regularization
- Dropout
- Batch normalization

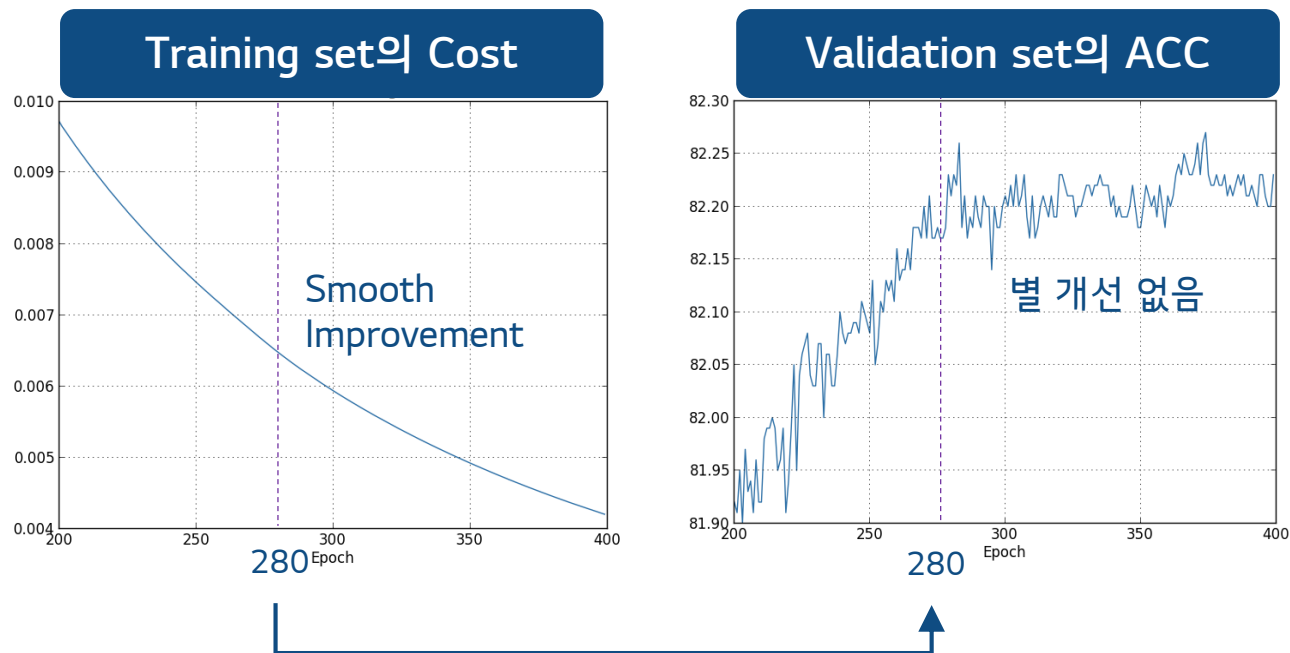
**Generalization error**를 감소시키려는 모든 노력

(※Training error 감소가 목적이 아님)

즉, **overfitting** 방지를 위함

# Regularization

Learning curve 해석하기 : MNIST 예제에서의 Overfitting



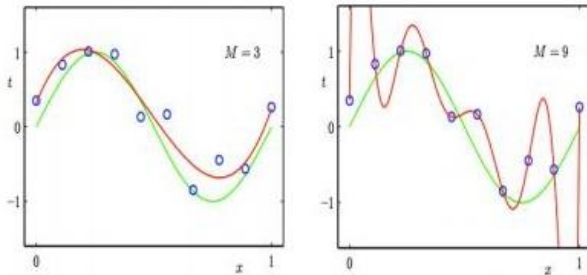
280 epoch 이후,  
모델이 overfitting(또는 overtraining) 되었다!

# Regularization

## Approach 1 : Model capacity 조절

Architecture 측면에서  
불필요하게 큰 모델 줄이기!

Hidden layer의 노드 수를 줄이거나...  
Layer 쌓는 수를 줄이거나 등등..



predictor too flexible:  
fits noise in the data

## Weight decay

아키텍처는 그대로,  
단 불필요한 weight를 0근처로 유도함으로서  
Capacity를 줄이는 것과 유사한 효과!

주 이용 방법 : Cost 뒤 L1 또는 L2 penalty를 부여

### L1 Regularization

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|.$$

### L2 Regularization

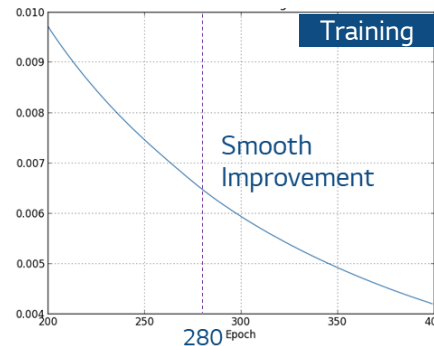
$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2$$

$\lambda$  : Regularization 정도를 조절하는 hyperparameter

## Early stopping

Validation set의 Cost 및 성능을 모니터링,  
Overfitting의 조짐이 보이면 그 지점에서 학습 중단!

만일 우측과 같은 학습 양상이 보이는 경우,  
280 epoch까지만 학습된 모델을 이용



# Regularization

## Approach 2 : 더 많은 데이터 확보 (Data Augmentation)

### Image data Augmentation

좌우반전, Crop, 명암, 채도, 대비 변경 등..



### Text data Augmentation

(Image data에 비해 상대적으로 방법이 많지 않다)

Backtranslation 등..

The screenshot shows a web-based translation interface used for text augmentation via backtranslation. It consists of three stacked translation panels. The top panel is set to '한국어 - 감지됨' (Korean - Detected) and shows the Korean sentence '저는 밀떡볶이를 좋아해요' (Jeoneun miltteobokki-ileul joh-ahaeyo). The middle panel is set to '영어 - 감지됨' (English - Detected) and shows the English translation 'I like wheat tteokbokki'. The bottom panel is set to '한국어' (Korean) and shows the backtranslated Korean sentence '나는 밀 떡볶이를 좋아한다' (Naneun mil tteobokki-ileul joh-ahanda). Blue curved arrows indicate the flow of text from Korean to English and back to Korean. The interface includes language selection dropdowns, a search icon, and a star icon for saving.

# Regularization

## Approach 3 : 서로 다른 여러 모델 이용하기 (Ensemble)

머신러닝의 집단지성 : 일반적으로 여러 모델을 결합할수록 성능이 좋아진다!

### 다른 architecture

서로다른 아키텍처를 가진 모델 학습,  
학습된 모델의 추론 결과를 앙상블

### 다른 Hyperparameter

서로 다른 hyper parameter 세팅으로 학습한  
모델의 추론 결과를 앙상블

### 다른 training step

한 모델을 학습하는 과정 중  
서로 다른 training step에 저장된 모델을 가져와  
추론 결과를 앙상블

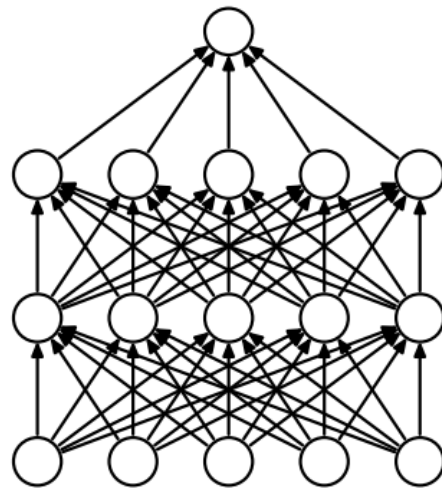
### 다른 initialization

동일한 아키텍처의 parameter를  
다른 방식으로 초기화하여 모델 학습,  
학습된 모델의 추론 결과 앙상블

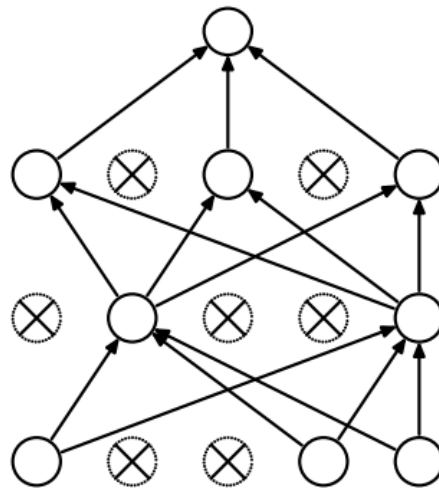
.. 이외 다양한 방법 가능

# Regularization

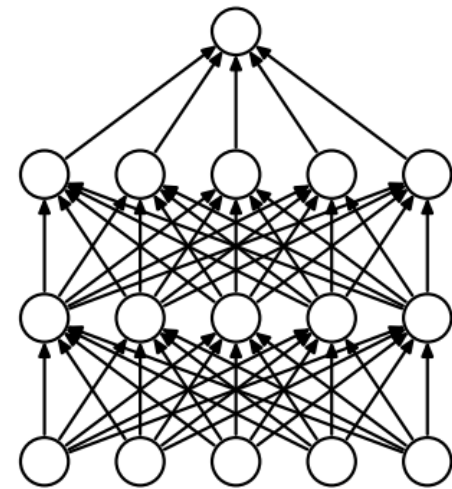
## Approach 4 : Dropout 이용하기



(a) Standard Neural Net



(b) After applying dropout.



(c) 학습이 끝나고 완성된 모델

Training

Inference

매 학습마다,  $p$ 의 확률로 각 뉴런을 drop할지 말지 결정  
Dropout된 뉴런은 일시적으로 삭제된 것으로 취급하고 학습

추론 시 모든 뉴런을 살리되,  
결과스코어에  $p$ 만큼을 곱하여 이용

마치 매번 다른 네트워크를 학습시키는 것과 같은 효과

여러 네트워크를 앙상블하는 것과 같은 효과

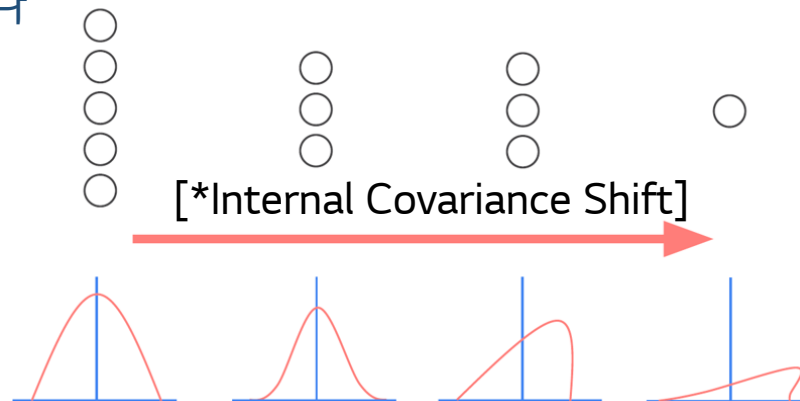


# Regularization

## Approach 5 : Batch Normalization 이용하기

Gradient Vanishing / Gradient Exploding의 이유를 '\*Internal Covariance Shift'로 판단하여 이를 해결하는 아이디어 중의 하나

Input DATA



\*Internal Covariance Shift:  
Network의 각 층이나  
Activation 마다 input의  
distribution이 달라지는 현상

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots x_m\}$ ;

Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

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

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

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

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

[설명]

네트워크 학습 시 mini-batch 단위로 데이터 학습,

→ Layer마다 mini-batch의 feature가 output으로 계산,

→ Feature의 평균과 표준편차를 구하여 normalize 해주고, → scale factor와 shift factor를 이용하여 새로운 값을 만들어준다.

[장점]

- 학습 속도 개선
- Weight initialization 의존성감소 (학습을 할 때마다 출력값을 정규화하기 때문)
- Overfitting 위험감소 (Dropout 대체 가능)
- Gradient Vanishing 문제 해결