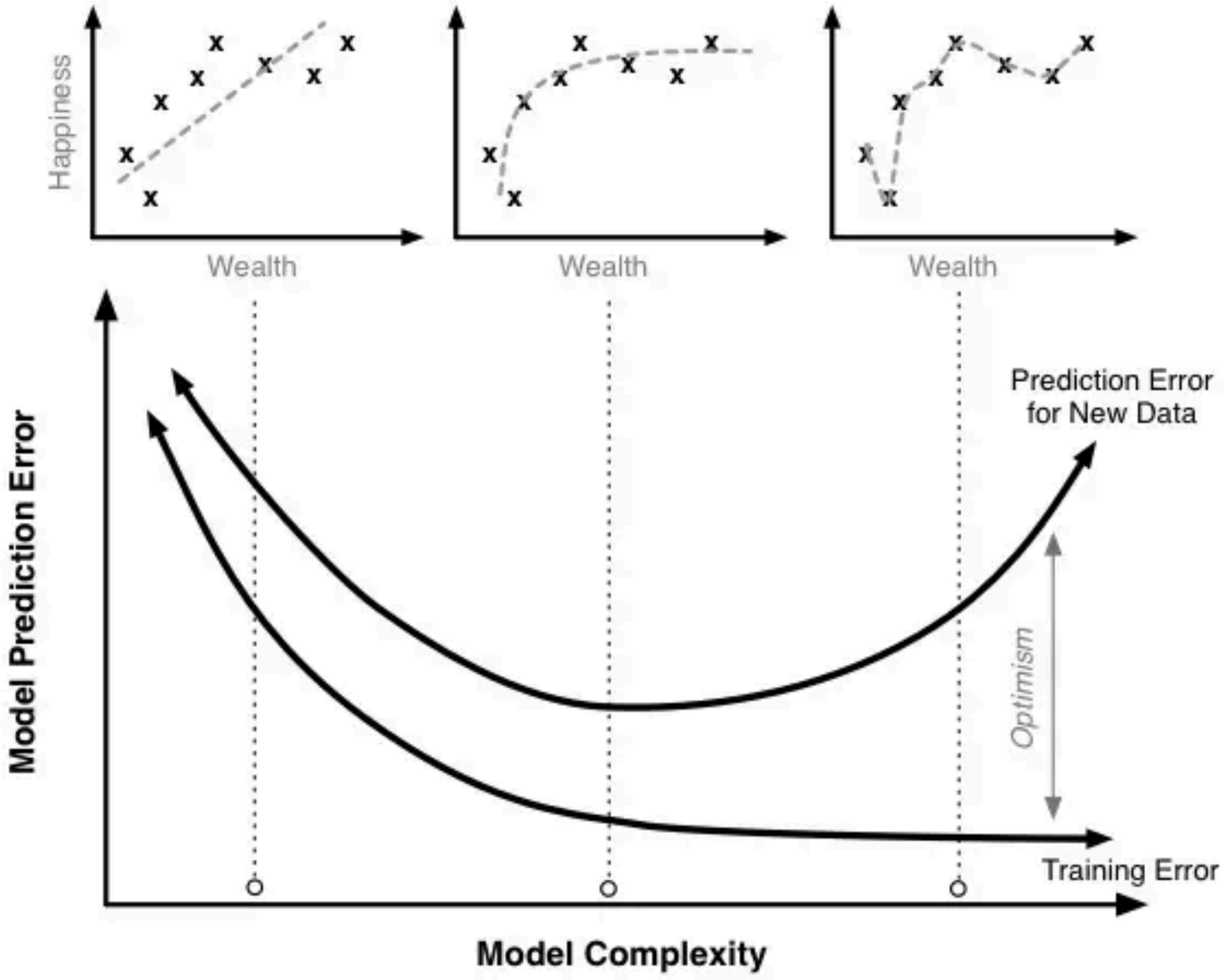
TECH TIPS FOR ML

OVER / UNDERFITTING



OVERFITTING

- **train error 가 적은 경우,** 학습이 진행됨에 따라 train error 와 test error 의 차이(generalization error) 가 줄어들지 않거나 더 늘어나는 경우 오버피팅을 의심해볼 수 잇음

- 데이터의 거시적 분포에 비해 모델이 너무 복잡하여, 불필요한 noise 분포에 영향을 받은 상태

UNDERFITTING

- **train error** 가 큰 경우

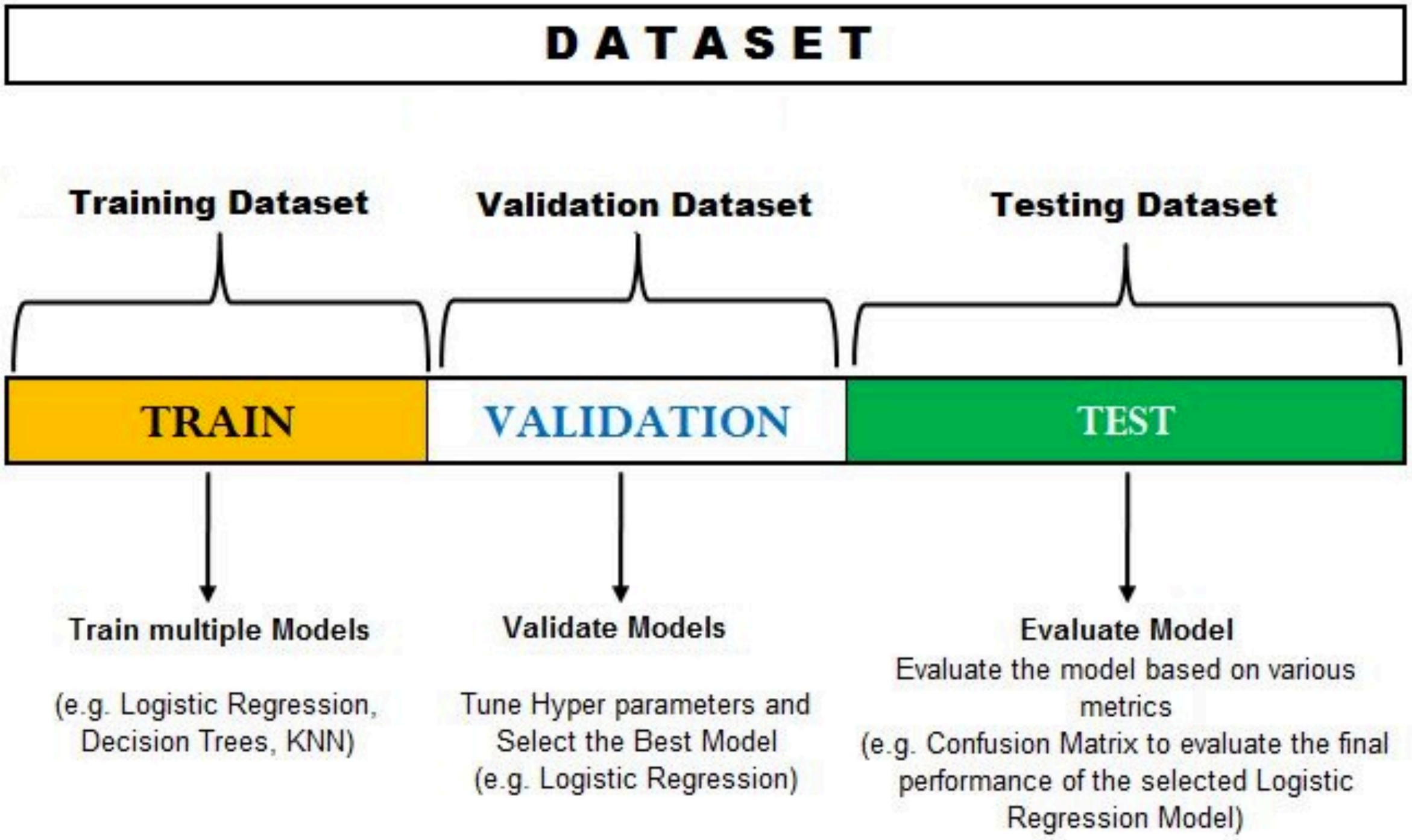
- 정의한 모델이 데이터의 거시적 분포를 충분히 표현하지 못하는 장태

DEAL WITH OVER / UNDERFITTING

- test dataset 은 모델의 **최종 성능 측정**을 위해서 단 한번 사용됨

- epoch 단위로 over/underfitting 상태를 감지하여 튜닝 기법을 활용하고 싶다

- 여러 모델간 성능 비교를 해가며 돌려보고 싶다



https://www.datavedas.com/holdout-cross-validation/

NOTATE ERROR FUNCTION WITH BIAS, VARIANCE

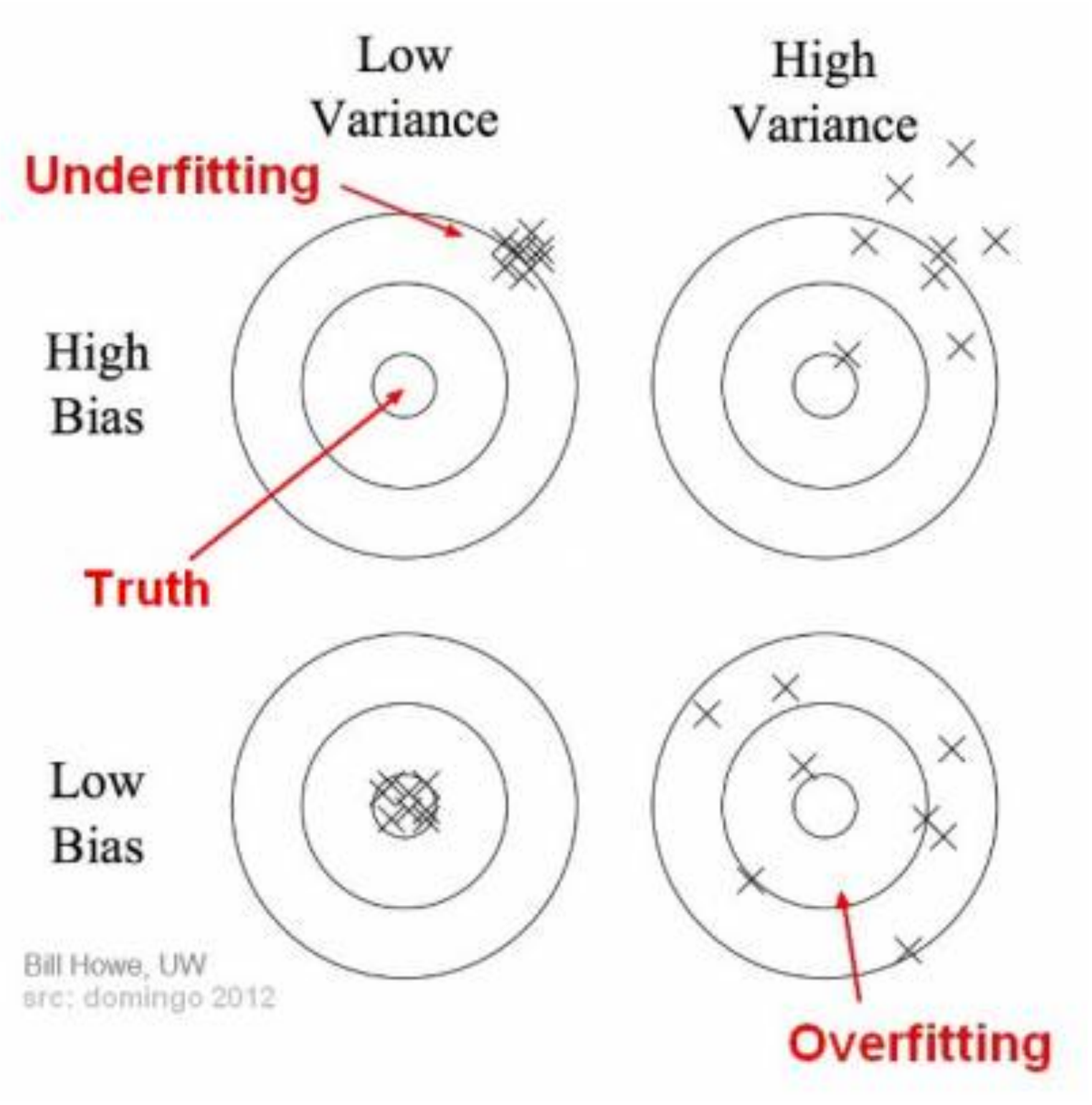
- x : data, y : ground truth, $\hat{f}(x)$: predicted
- $Bias[f(x)] = E[f(x) - f(x)]$
- $Var[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$
- $E[(y - \hat{f}(x))^2] = Bias[\hat{f}(x)]^2 + Var[\hat{f}(x)] + \sigma^2$

$$y' = f(x') + \epsilon', \epsilon' \sim N(0,\sigma^2)$$

normal dist 를 가정
eps = noise
sigma = stddev

$$\begin{aligned}
\text{MSE} &= \mathbb{E}(y' - \hat{f}(x'))^2 \\
&= \mathbb{E}(y' - f(x') + f(x') - \hat{f}(x'))^2 \\
&= \mathbb{E}(\varepsilon' + (f(x') - \hat{f}(x')))^2 \\
&= \mathbb{E}(\varepsilon'^2 + 2\varepsilon'(f(x') - \hat{f}(x')) + (f(x') - \hat{f}(x'))^2) \\
&= \mathbb{E}(\varepsilon'^2) + 2\mathbb{E}(\varepsilon'(f(x') - \hat{f}(x'))) + \mathbb{E}(f(x') - \hat{f}(x'))^2 \\
&= \sigma^2 + \mathbb{E}(f(x') - \hat{f}(x'))^2 \\
&= \sigma^2 + (\mathbb{E}(f(x') - \hat{f}(x')))^2 + \text{Var}(f(x') - \hat{f}(x')) \\
&= \text{inreducible noise} + \text{bias}^2 + \text{variance}
\end{aligned}$$

https://zhangyuqing.github.io/2019/06/ml-summary-series-2---bias-variance-tradeoff/



https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

REGULARIZATION

- loss function 에 제약조건을 부여해 overfitting 을 해결하려는 기법

- weight update 에 직접적인 영향을 줌

REGULARIZATION

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

http://laid.delanover.com/difference-between-l1-and-l2-regularization-implementation-and-visualization-in-tensorflow/

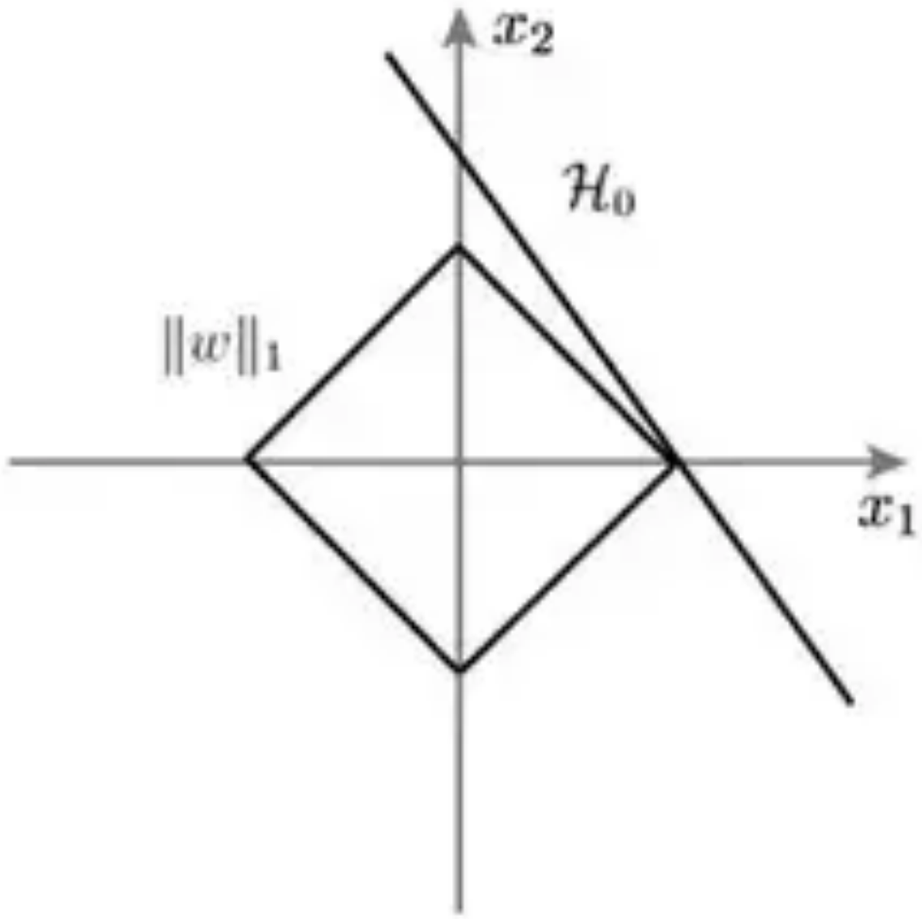
REGULARIZATION
L1 VS L2

- L1 방식의 경우 특정 weight 가 **정확히 0이 될 가능성**이 높음 - 입력 데이터의 해당 feature 가 무시됨

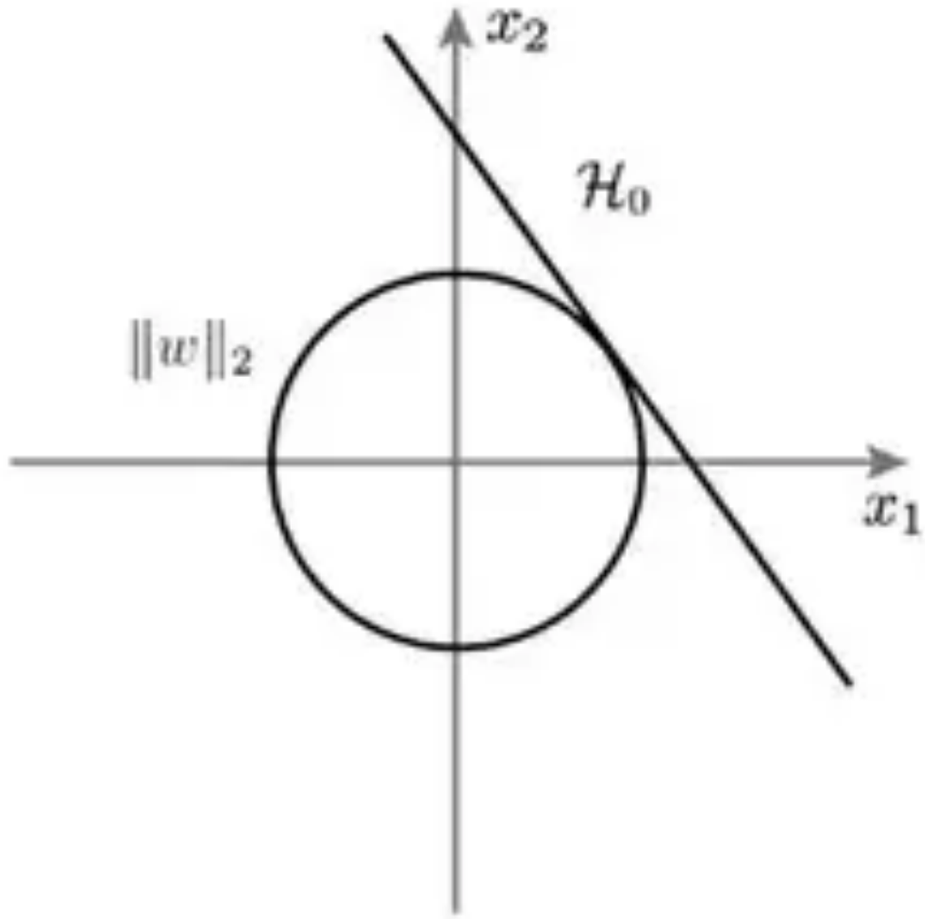
- L2 의 경우 제곱의 특성으로 0에 가깝더라도 특성을 거의 살리는 특징이 있음

REGULARIZATION

A L1 regularization



B L2 regularization



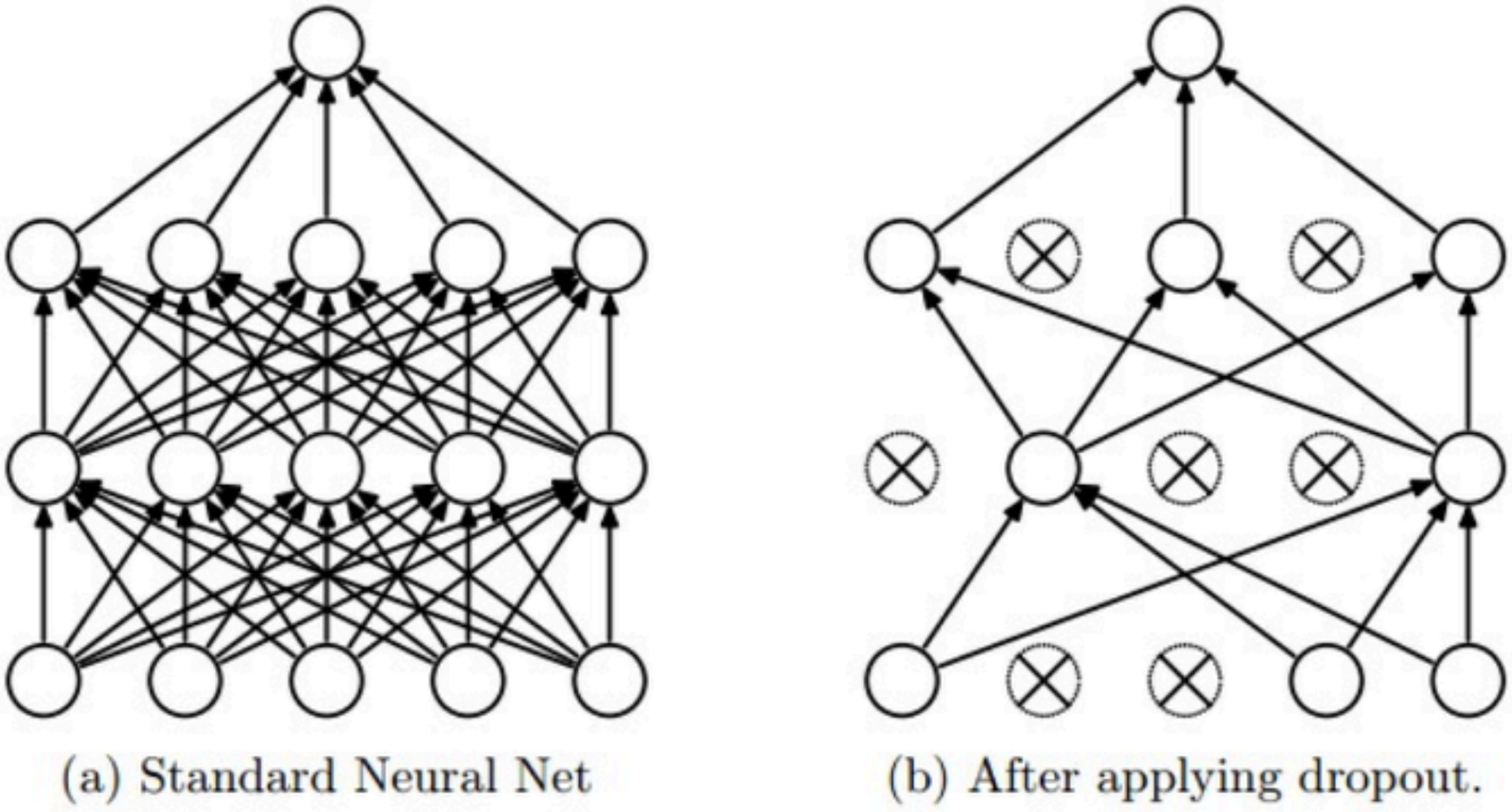
https://www.quora.com/When-would-you-chose-L1-norm-over-L2-norm

DROPOUT

- **training 과정에**, 임의의 비율(p)의 neuron 들을 disable 하는 방법

- test(predict) 단계에서는 train 단계에 비해 neuron 들이 전달 받는 값이 커지지 않도록 (1-p) 를 곱하여 전달

DROPOUT



(a) Standard Neural Net

(b) After applying dropout.

Figure 1: Taken from the paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"

https://kevinthegrey.tistory.com/131

DATA AUGMENTATION

- 데이터 뻥튀기(?)
- 이미지의 경우 scale,rotate,translate,crop 과 같은 처리시 다른 픽셀데이터 패턴을 동일 label 의 데이터로 확보하는 효과를 기대할 수 있다.
- pytorch 의 경우 torchvision 라이브러리에 편리한 내장 기능이 제공됨

DATA AUGMENTATION

- tensorflow 에서도 기본적인 image transform 기능 제공
- https://www.wouterbulten.nl/blog/tech/data-augmentation-using-tensorflow-data-dataset/

WEIGHT INITIALIZATIONS

- 정답을 모르는데, 빠르고 정확한 train 에 도움이 되도록 weight 초기화를 잘할 수 있을까?

- 데이터가 여러 layer 를 거쳐 가더라도 그 activation 의 result 가 일정한 범위 안에 잇도록 잡아주는 초기화값 사용이 좋다고 함

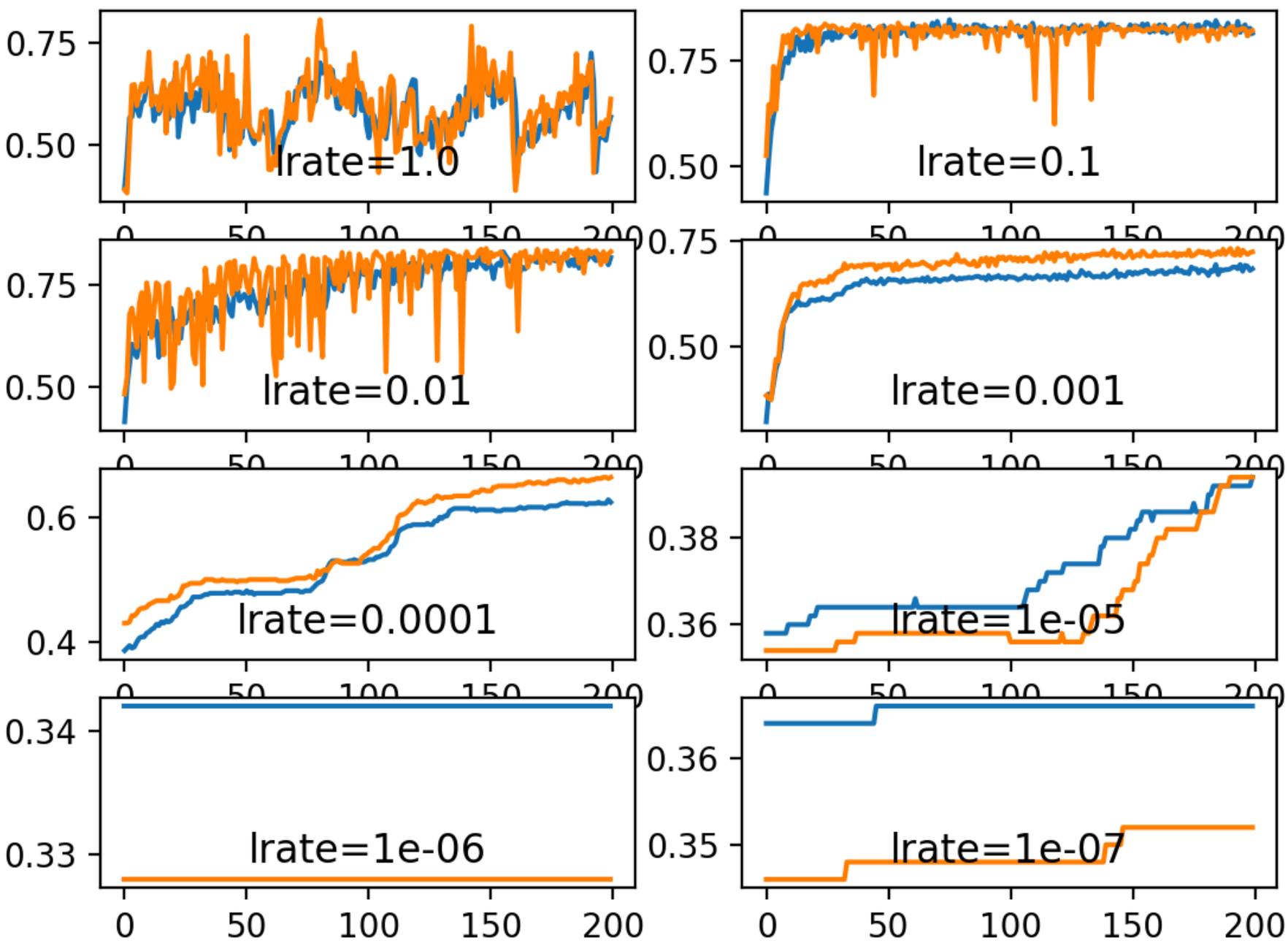
WEIGHT INITIALIZATIONS

- Xavier : sigmoid, tanh 같은 continuous function 에 적합
- He : ReLU 를 위한 특별한 방법

WEIGHT INITIALIZATIONS

- 참고 자료(Thanks to SooDevv)
- https://github.com/SooDevv/deep-learning-from-scratch2/blob/master/ch5_RNN/weight_initialization.ipynb

LEARNING RATE



https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

DECAYING LEARNING RATE

- 초기에 높게 하고 점점 감소시키는 전략을 일반적으로 사용
  - 조건에 맞게 0 < gamma < 1 을 만족하는 gamma 를 곱함
- 일반적인 decay 방법
  - step decay : n epoch 마다 gamma 를 곱함
  - exponential decay : 매 epoch 마다 gamma 를 곱함

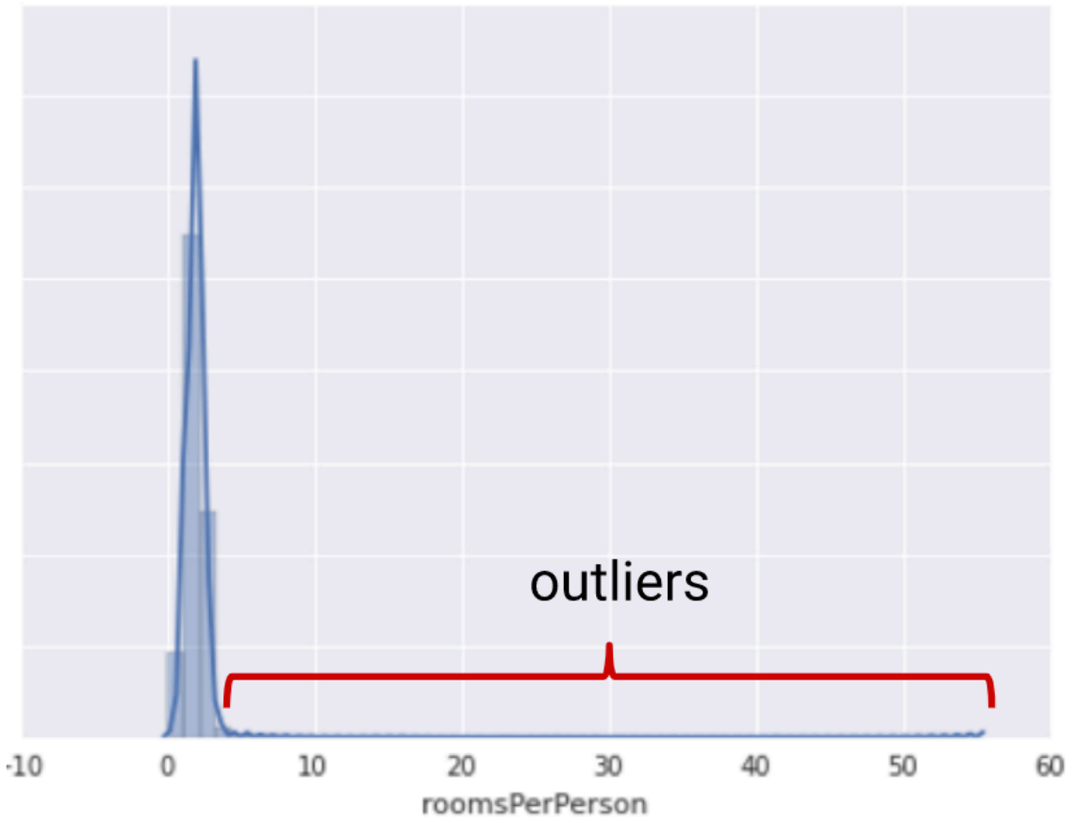
NORMALIZATION

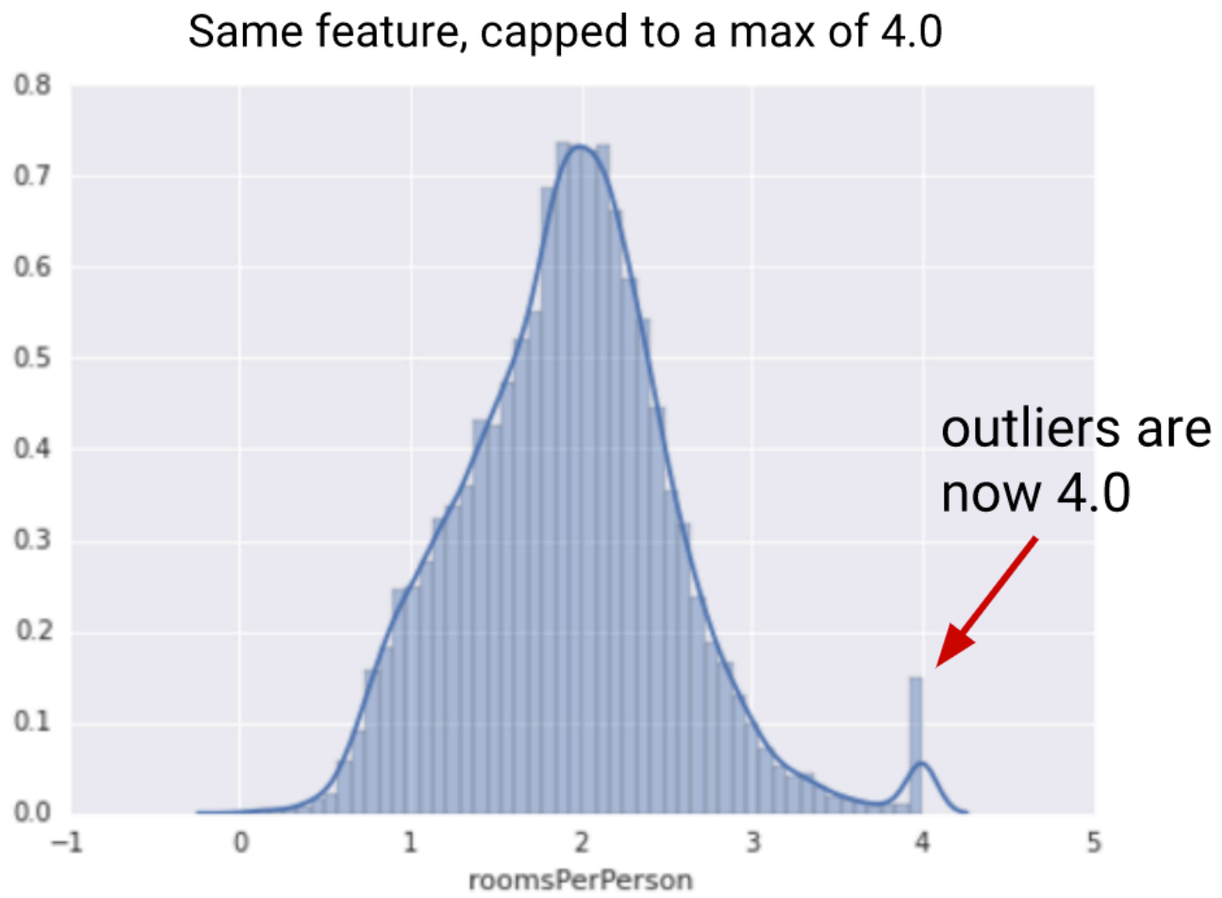
- 입력 데이터가 가질 여러 범위나 분포도를 일정한 범위로 조정하여 학습에 유리하게 함

NORMALIZATION

- Standardization : $z = \frac{x - \mu}{\sigma}$
- min-max : $z = \frac{x - min(x)}{max(x) - min(x)}$

FEATURE CLIPPING





https://developers.google.com/machine-learning/data-prep/transform/normalization

BATCH NORMALIZATION

- 일반적인 Normalization 로 해결되지 않는 단점
  - internal covariate shift

- 신경망 내부 layer 간에 전달되는 데이터의 범위를 일정하게 하고자 하는 목적

BATCH NORMALIZATION

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

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

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

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

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

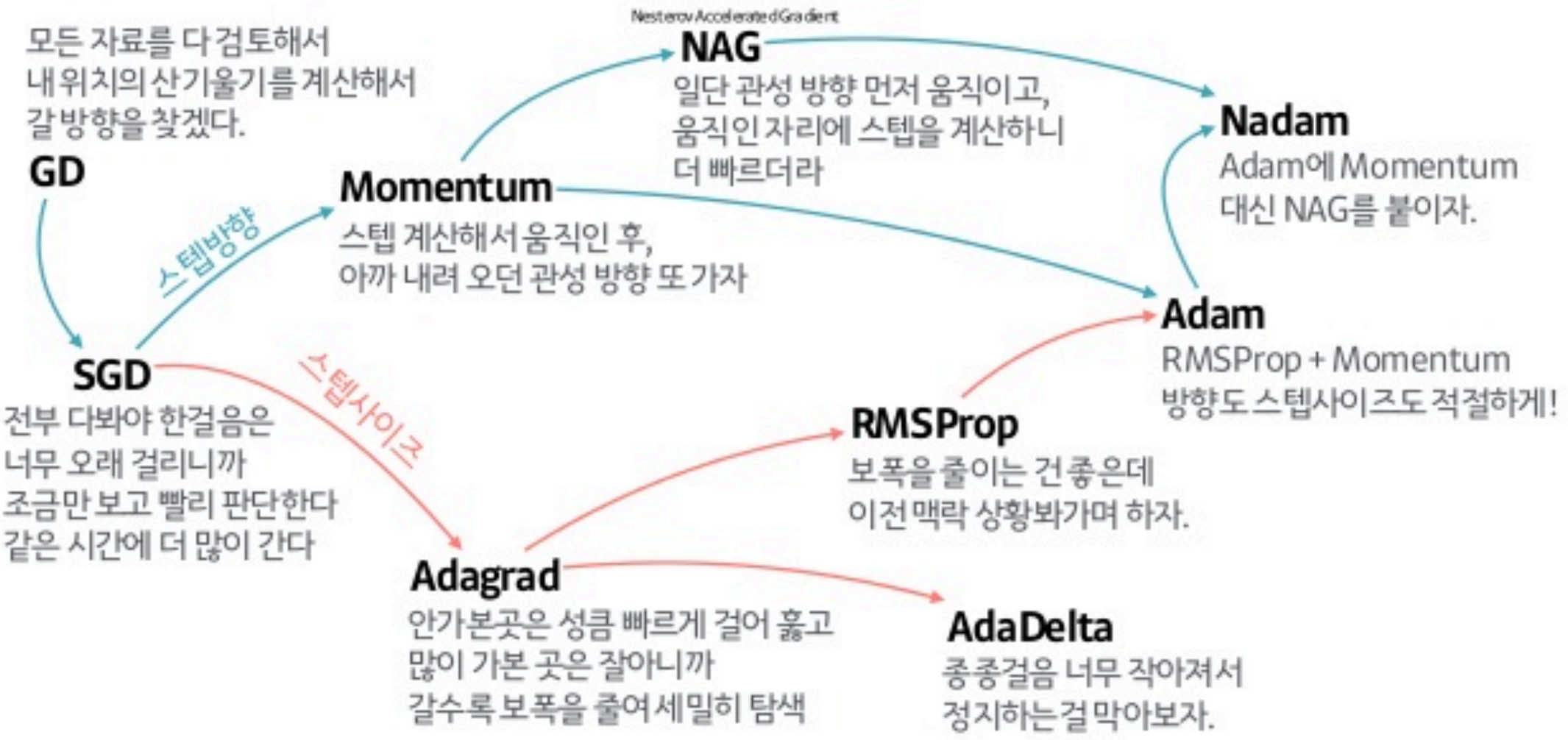
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

http://sanghyukchun.github.io/88/

BATCH NORMALIZATION

- DenseNet 사용 사례
  - Conv -> Batchnorm -> Activaton
  - hidden layer 마다 적용하여, 다음 layer 로 전달되는 값의 범위을 일정하게 하는 효과

VARIATIONS OF GRADIENT DESCENT



https://sacko.tistory.com/m/42

http://shuuki4.github.io/deep%20learning/2016/05/20/Gradient-Descent-Algorithm-Overview.html

REFERENCES

- https://3months.tistory.com/118
- https://bywords.tistory.com/entry/%EB%B2%88%EC%97%AD-%EC%9C%A0%EC%B9%98%EC%9B%90%EC%83%9D%EB%8F%84-%EC%9D%B4%ED%95%B4%ED%95%A0-%EC%88%98-%EC%9E%88%EB%8A%94-biasvariance-tradeoff
- https://zhangyuqing.github.io/2019/06/ml-summary-series-2---bias-variance-tradeoff/
- https://reniew.github.io/13/
- https://gomguard.tistory.com/184