Deep Learning

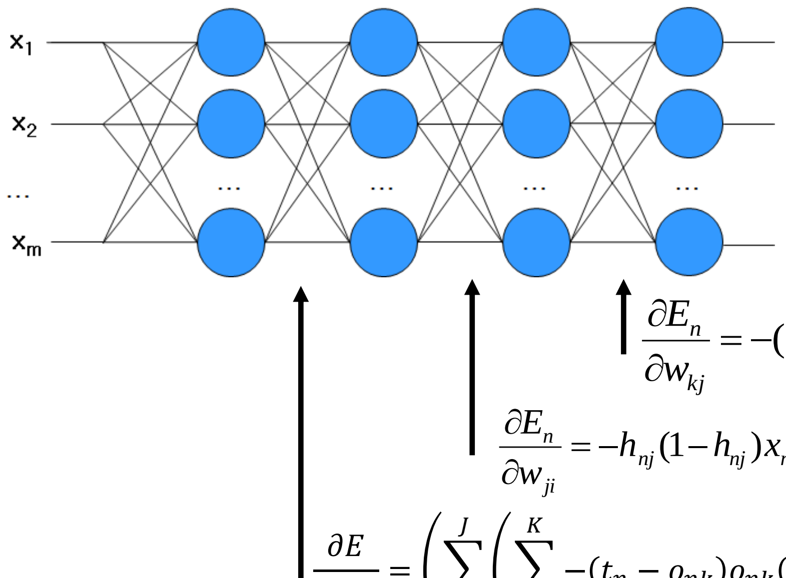
Content

- **Gradient Vanishing & Activation Functions**
- **Regularization: Dropout**
- **Stochastic Gradient Descent**
- 그 외 주제들
  - Momentum
  - Adam
  - Batch Normalization

Gradient Vanishing & Activation Functions

Gradient Vanishing & Exploding

- **Gradient is easy to vanish or explode**
  - To many terms are multiplied.
  - If some are small numbers, gradient becomes very small.
  - If some are large numbers, gradient becomes very large.



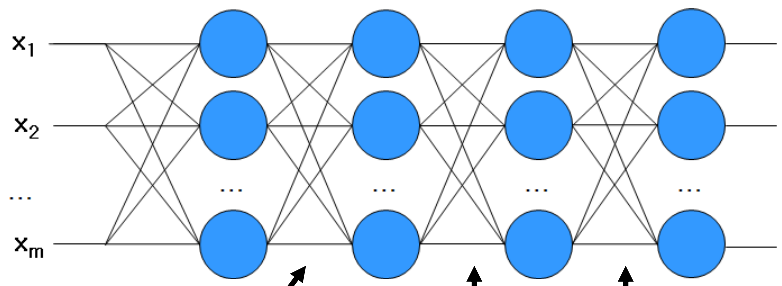
$$\frac{\partial E_n}{\partial w_{kj}} = -(t_{nk} - o_{nk}) o_{nk} (1 - o_{nk}) h_{nj}$$

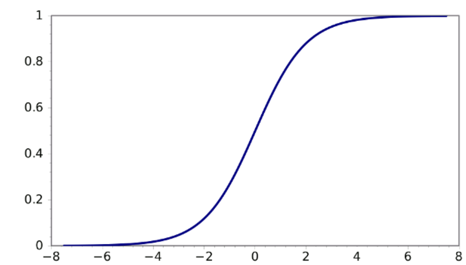
$$\frac{\partial E_n}{\partial w_{ji}} = -h_{nj}(1 - h_{nj}) x_{ni} \sum_{k=1}^{m} w_{kj}(t_{nk} - o_{nk}) o_{nk}(1 - o_{nk})$$

$$\frac{\partial E}{\partial w_{ip}} = \left(\sum_{j=1}^{J} \left(\sum_{k=1}^{K} -(t_n - o_{nk}) o_{nk} (1 - o_{nk}) w_{kj} \right) h_{nj}(1 - h_{nj}) w_{ji} \right) h_{ni}(1 - h_{ni}) h_{np}$$

Activation Function

- **Vanishing Gradient**
  - The major terms are the derivatives of the activation function





$$\frac{\partial \mathbf{Sigmoid}}{\partial w} \leq \frac{1}{4}$$

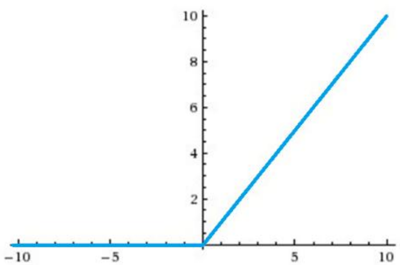
$$\frac{\partial E_n}{\partial w_{kj}} = -(t_{nk} - o_{nk})o_{nk}(1-o_{nk})h_{nj}$$

$$\frac{\partial E_n}{\partial w_{ji}} = -h_{nj}(1-h_{nj})x_{ni}\sum_{k=1}^{m} w_{kj}(t_{nk}-o_{nk})o_{nk}(1-o_{nk})$$

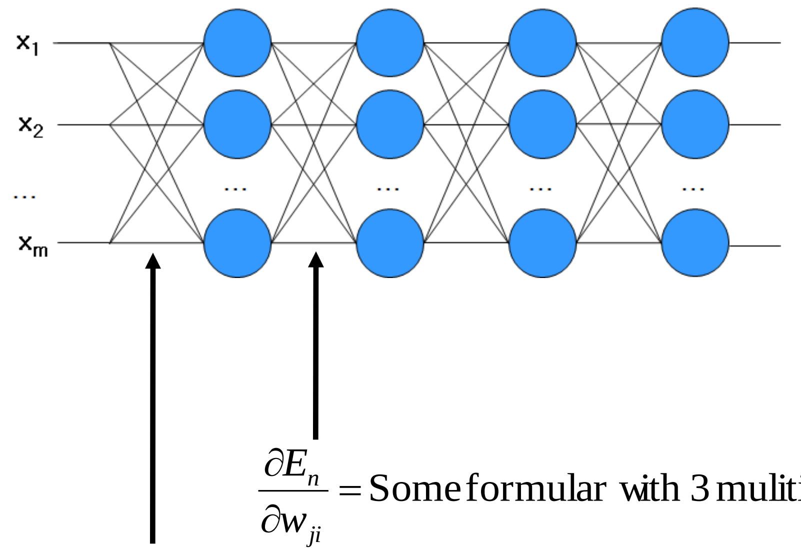
$$\frac{\partial E}{\partial w_{ip}} = \left(\sum_{j=1}^{J}\left(\sum_{k=1}^{K} -(t_n - o_{nk})o_{nk}(1-o_{nk})w_{kj}\right)h_{nj}(1-h_{nj})w_{ji}\right)h_{ni}(1-h_{ni})h_{np}$$

Activation Function

- **Using another functions instead of sigmoid**
  - Rectified Linear Unit (ReLU)



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

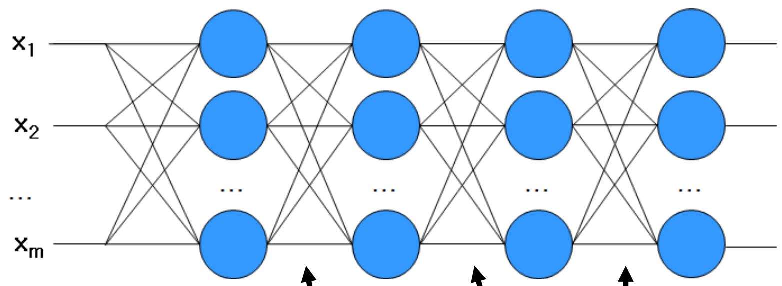


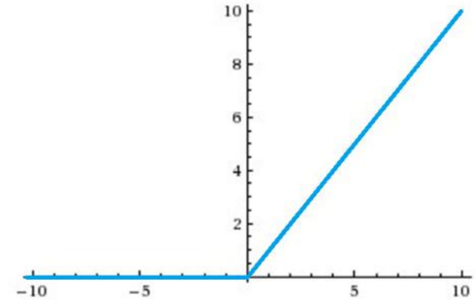
$$\frac{\partial E_n}{\partial w_{ji}} = \text{Some formular with 3 mulitiplic ations of } \frac{\partial f}{\partial w}$$

$$\frac{\partial E_n}{\partial w_{hg}} = \text{Some formular with 4 mulitiplic ations of } \frac{\partial f}{\partial w}$$

Activation Function

- **Vanishing Gradient**
  - The major terms are the derivatives of the activation function





$$\frac{\partial E}{\partial w_{kj}} = -h_{nj}(t_n - o_{nk})$$

$$\frac{\partial E}{\partial w_{ji}} = h_{ni} \sum_{k=1}^{K} -(t_n - o_{nk}) w_{kj}$$

$$\frac{\partial E}{\partial w_{ip}} = h_{np} \sum_{j=1}^{J} \left(\sum_{k=1}^{K} -(t_n - o_{nk}) w_{kj} \right) w_{ji}$$

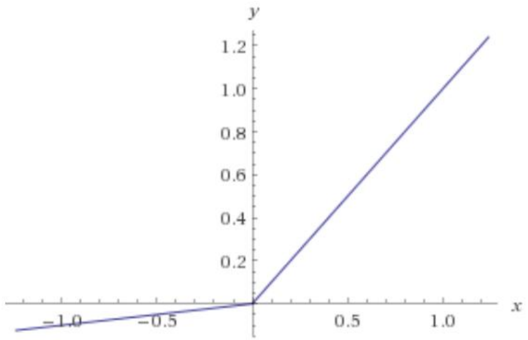
Activation Function

- **Advantage**
  - No vanishing gradient problems.
    - Deep networks can be trained without pre-training
  - Sparse activation
    - In a randomly initialized network, only about 50% of hidden units are activated
  - Fast computation:
    - 6 times faster than sigmoid function
- **Disadvantage**
  - Knockout Problem

Activation Function

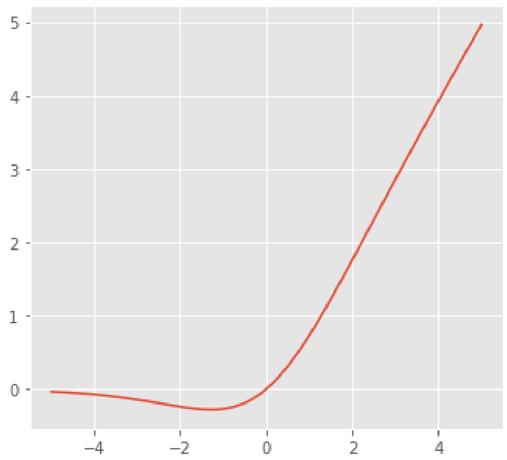
- **You may use another**
  - Leaky ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$



  - Swish (or SiLU-Sigmoid Linear Unit)

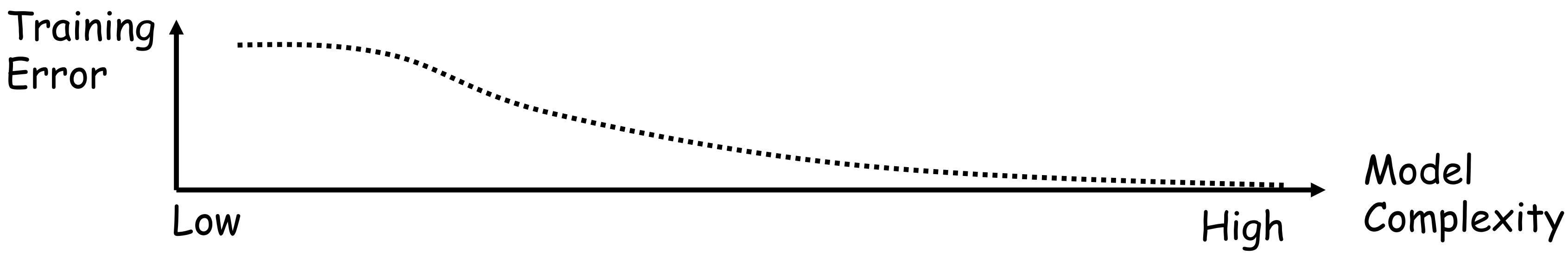
$$f(x) = \frac{x}{1 + e^{-x}}$$

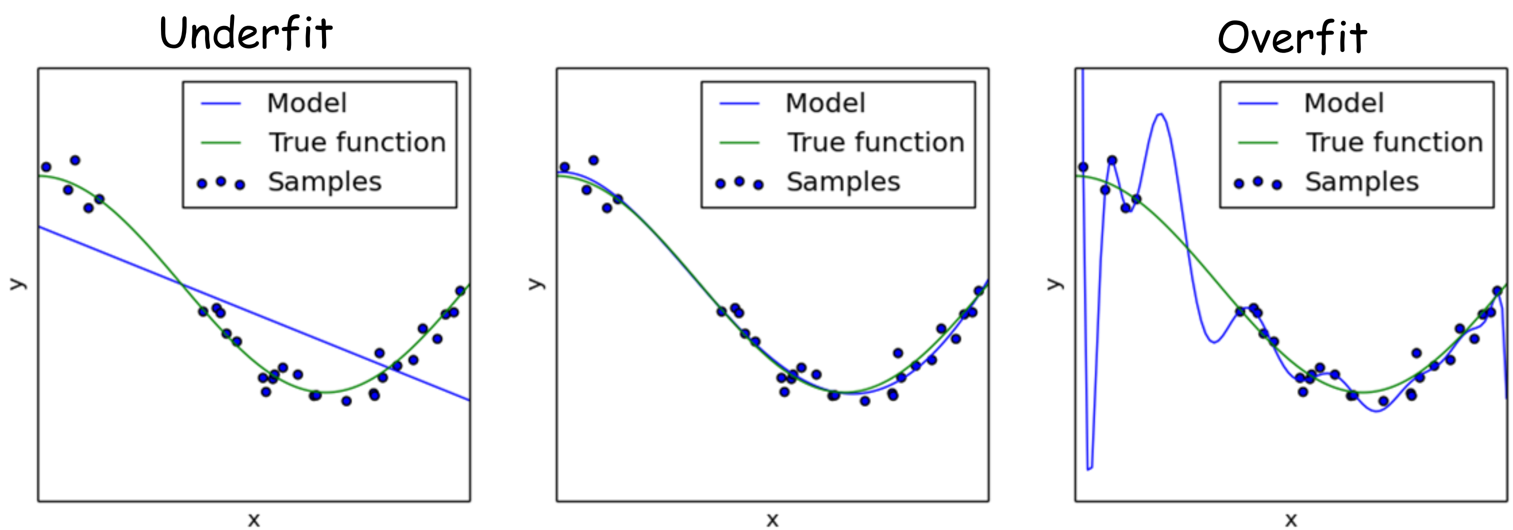


Regularization

Overfitting

- **Overfitting**





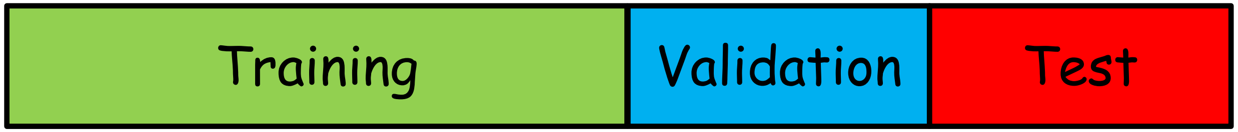
Regularization

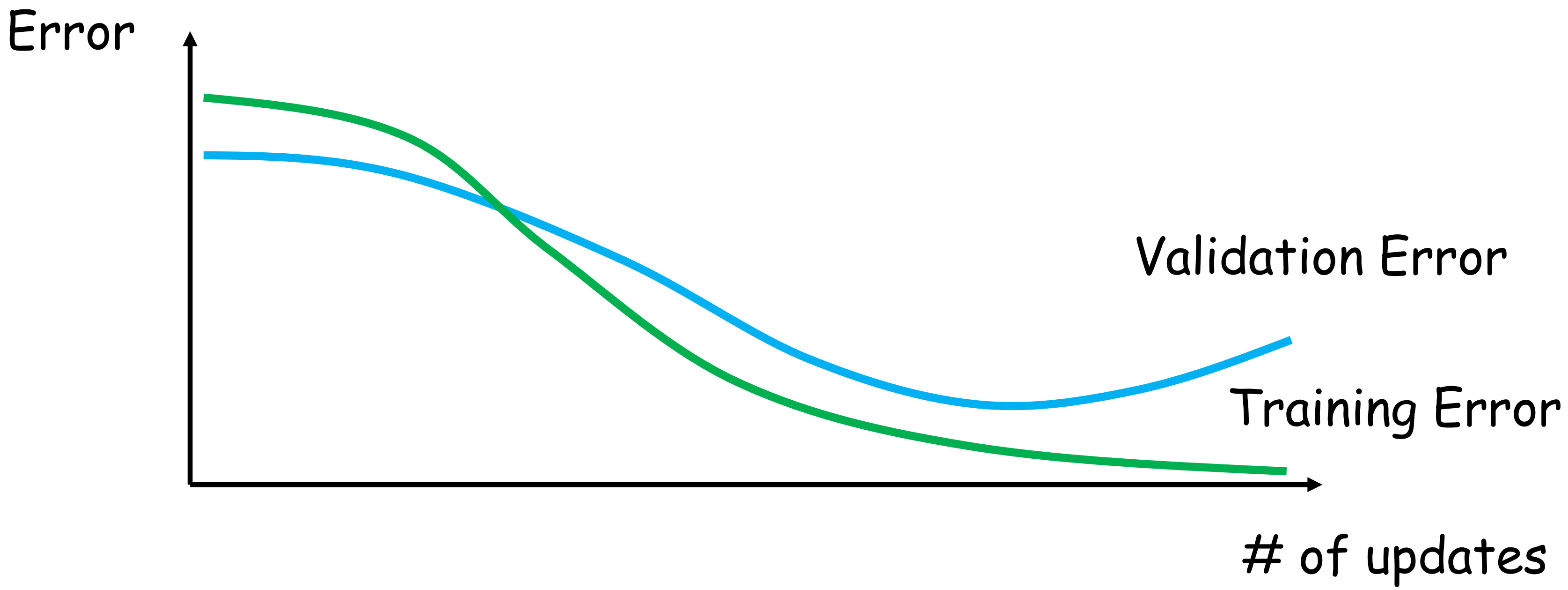
- **What is Regularization**
  - Introducing additional information to prevent over-fitting

- **Approaches**
  - Proper Learning: Early stopping
  - Proper Structure: Weight decay, Dropout,
    DropConnect, Stochastic pooling

Early Stopping

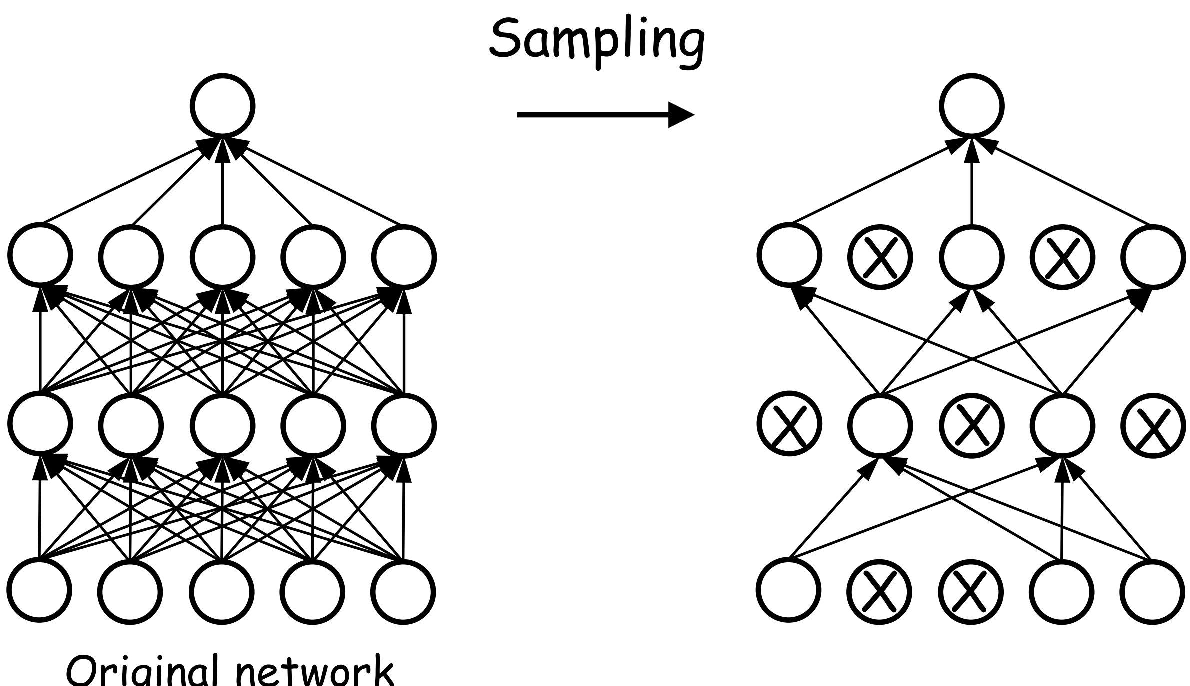
- **Split data into 3 groups**





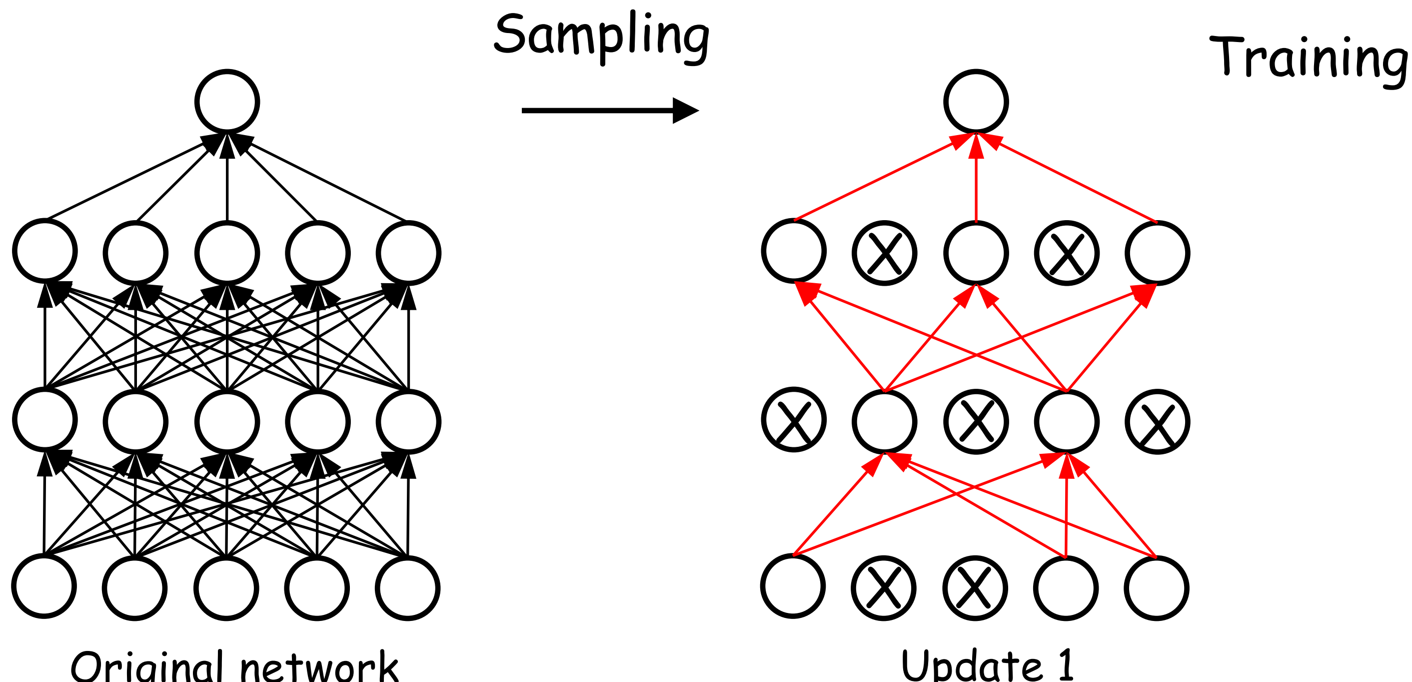
Dropout

- **How can we reduce the structural complexity without removing nodes?**
  - Hmm??



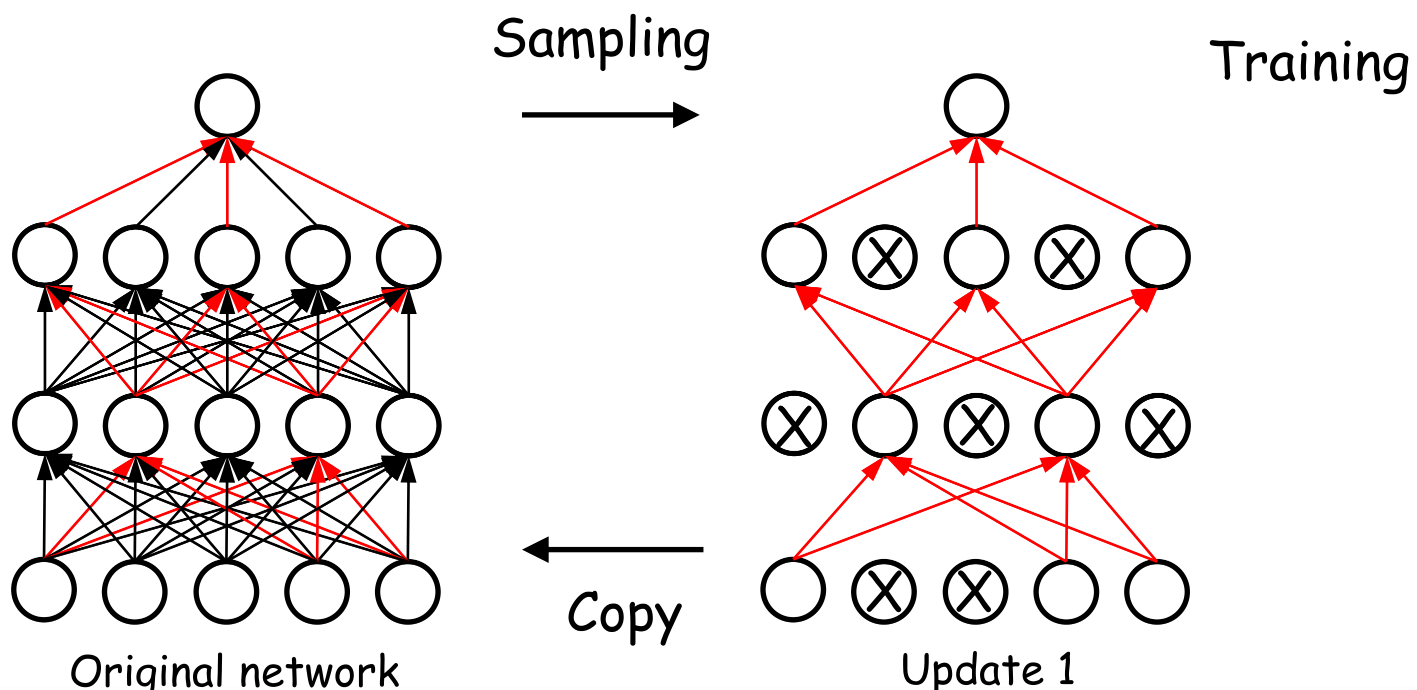
Dropout

- **How can we reduce the structural complexity without removing nodes?**
  - Hmm??



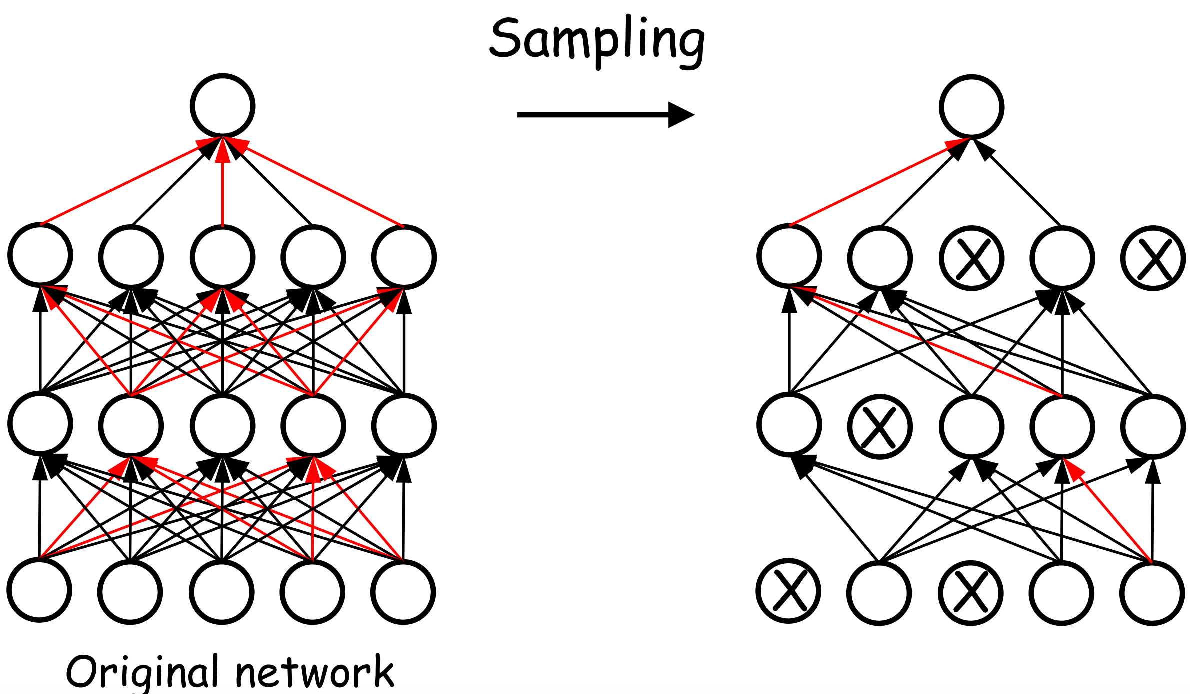
Dropout

- **How can we reduce the structural complexity without removing nodes?**
  - Hmm??



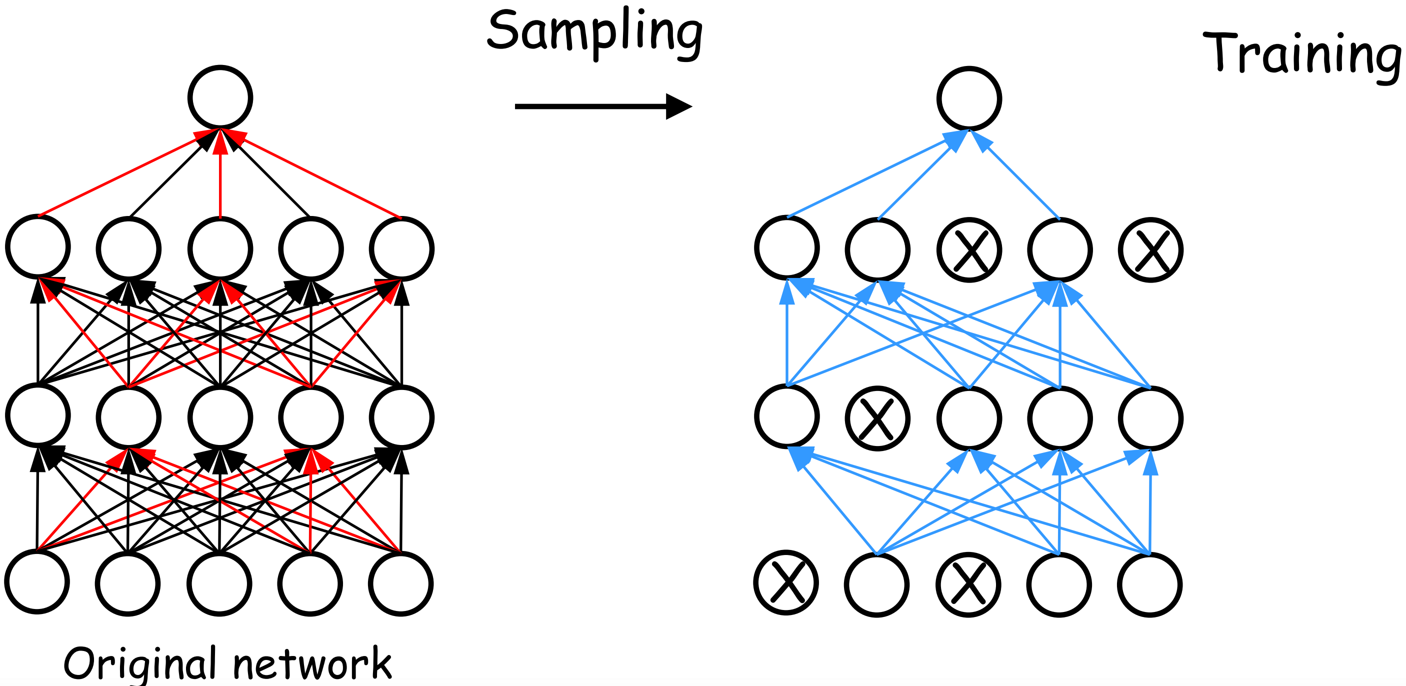
Dropout

- **How can we reduce the structural complexity without removing nodes?**
  - Hmm??



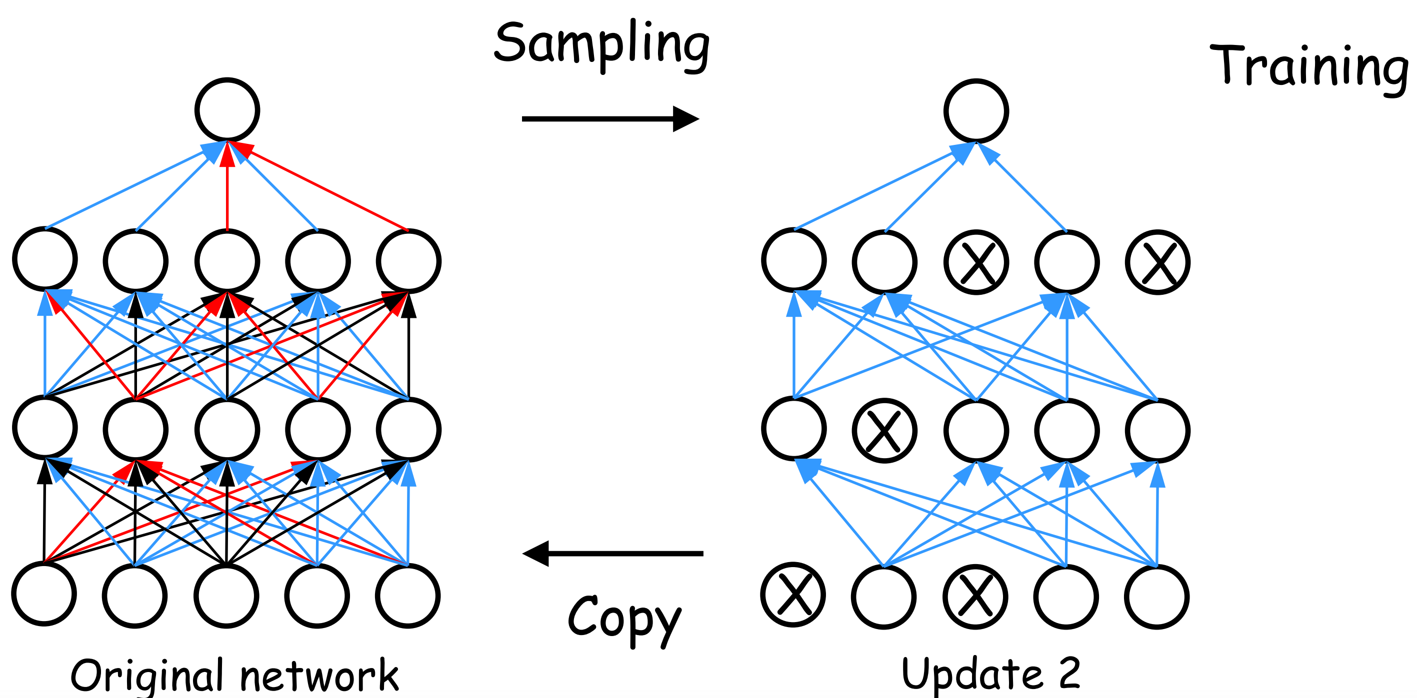
Dropout

- **How can we reduce the structural complexity without removing nodes?**
  - Hmm??



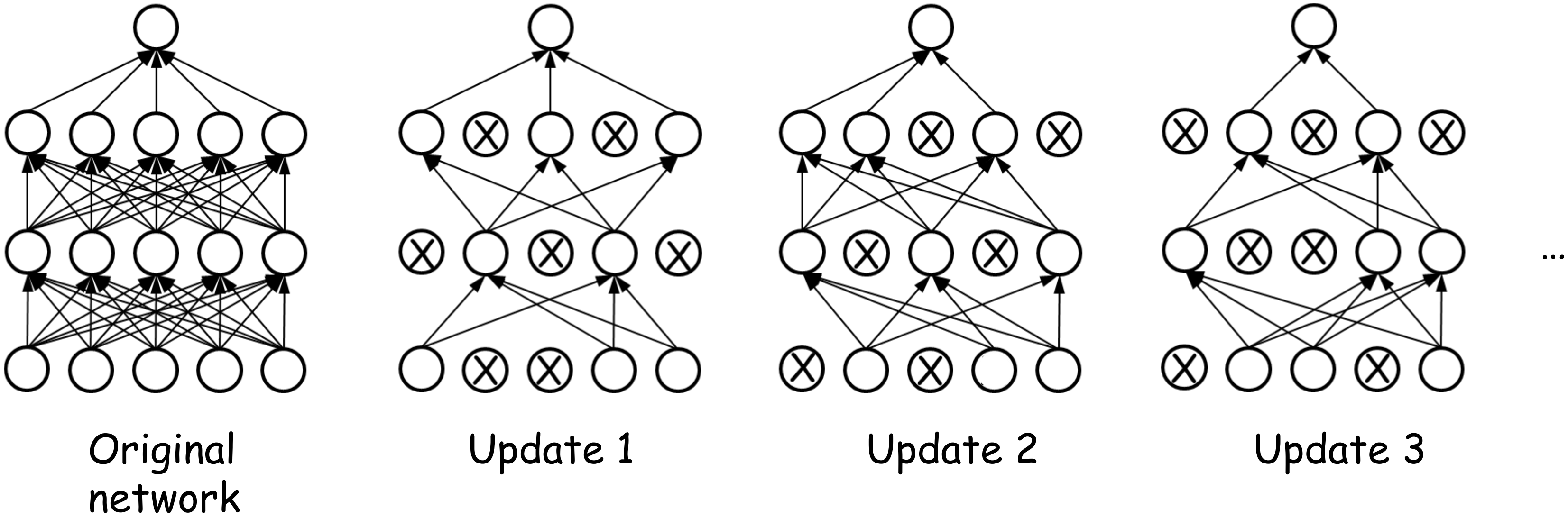
Dropout

- **How can we reduce the structural complexity without removing nodes?**
  - Hmm??



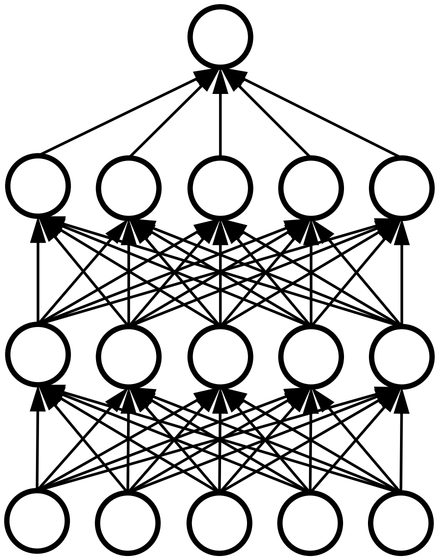
Dropout

- **Do this at every epoch**
  - Randomly choose nodes with a probability of *p*
    - Usually p = 0.5
  - Train the simplified neural network
    - At every epoch, we train different neural network which share connection weight each other



Dropout

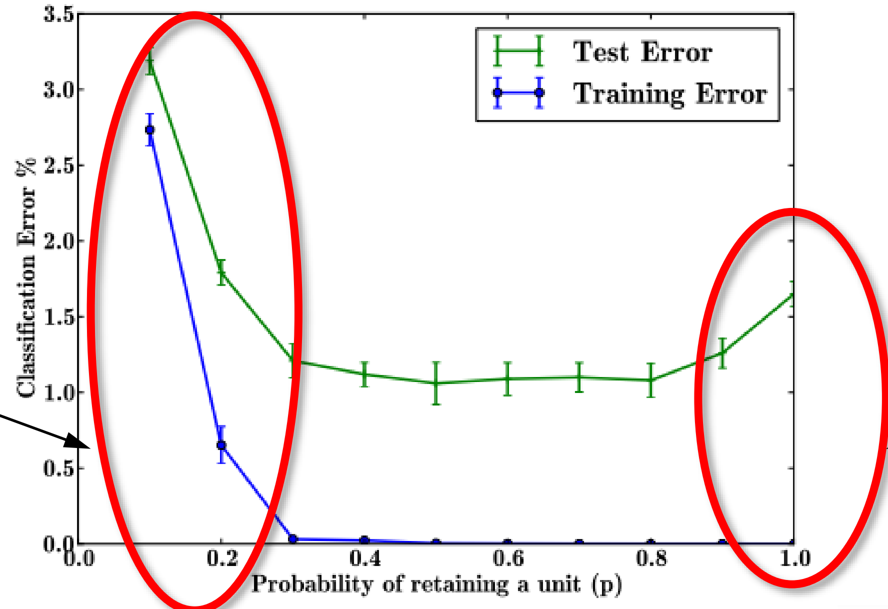
- **Testing**
  - Use all the nodes without dropout



Dropout

- **The effect of the dropout rate p:**
  - An architecture of 784-2048-2048-2048-10 is used on the MNIST dataset.
  - The dropout rate p is changed from small numbers (most units are dropped out) to 1.0 (no dropout).

Dropout with
high probability



No Dropout
->Overfitting

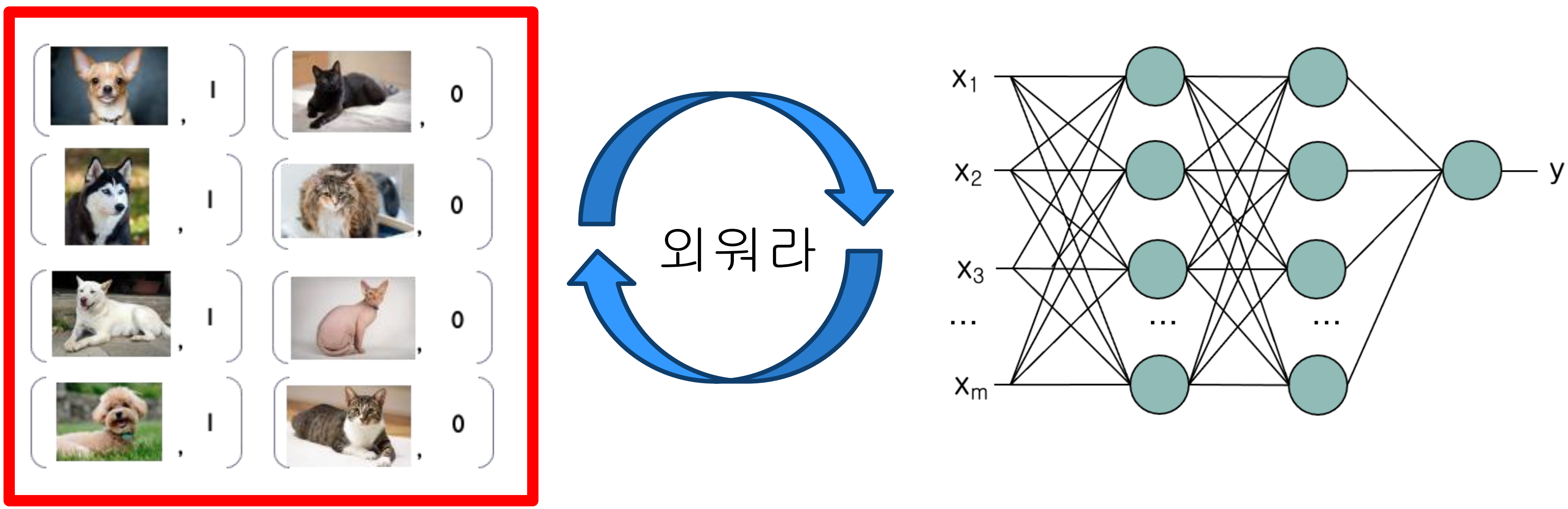
Dropout

- **Summary**
  - Dropout is a very good and fast regularization method.
  - Dropout is a bit slow to train (2-3 times slower than without dropout).
  - If the amount of data is average-large – dropout excels. When data is big enough, dropout does not help much.
  - Dropout achieves better results than former used regularization methods (Weight Decay).

Stochastic Gradient Descent

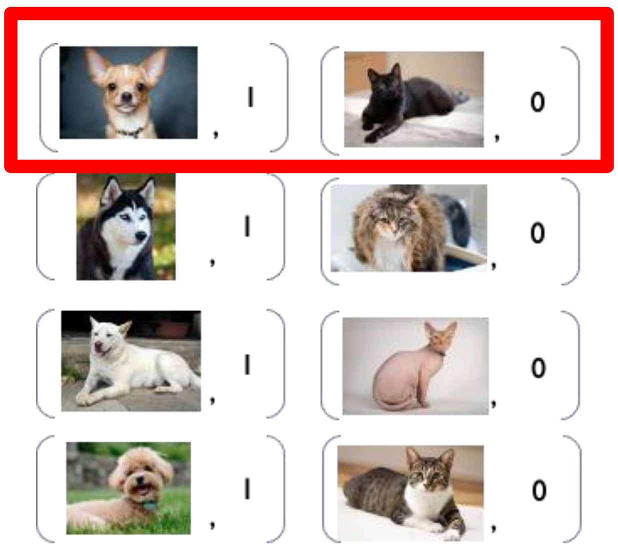
Batch Gradient Descent

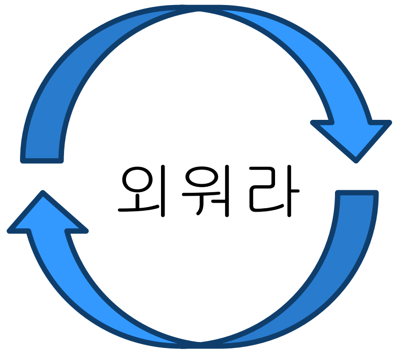
- **Batch mode**

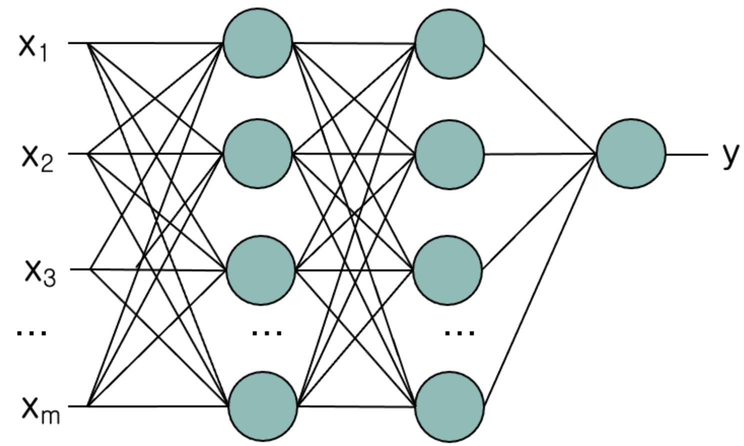


Mini-batch Gradient Descent

- **Mini-batch**
  - Batch size = 2

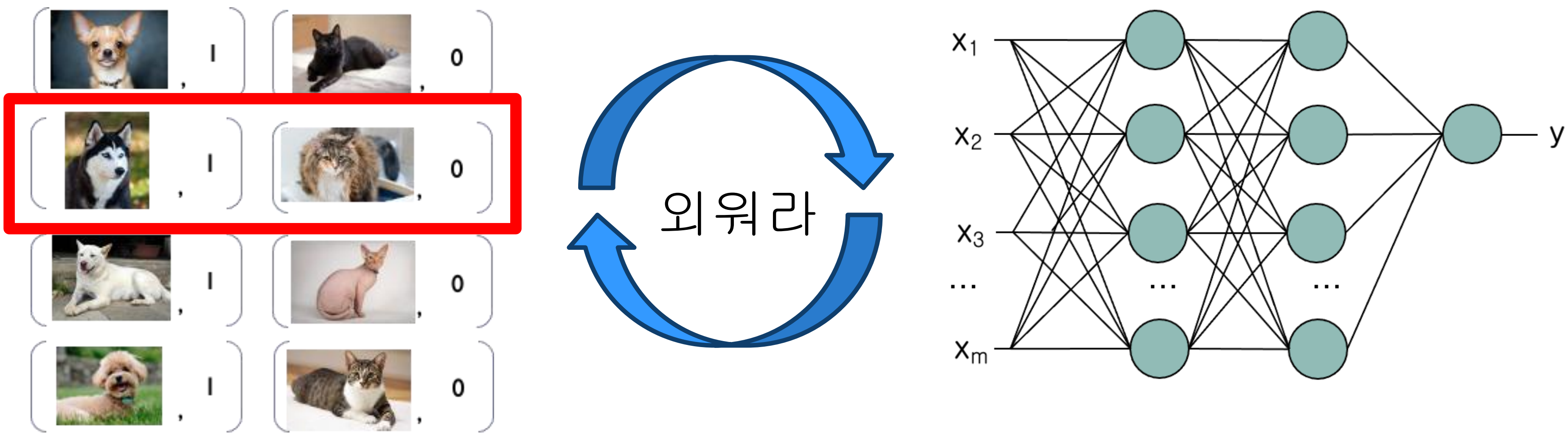






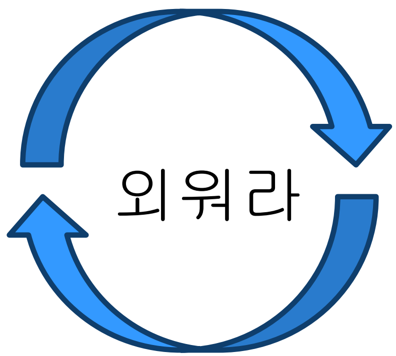
Mini-batch Gradient Descent

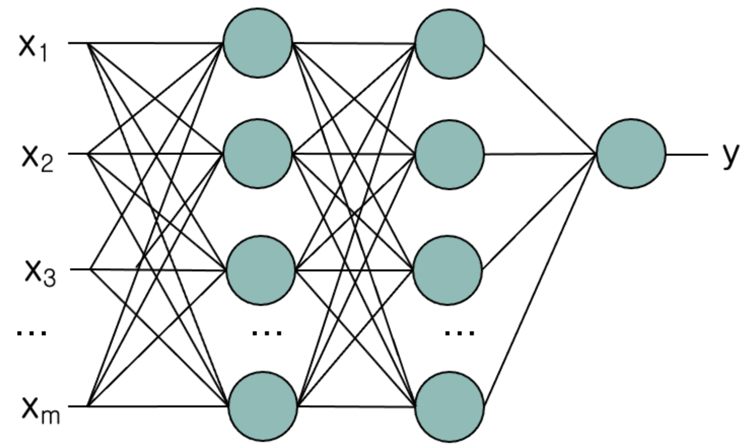
- **Mini-batch**
  - Batch size = 2



Mini-batch Gradient Descent

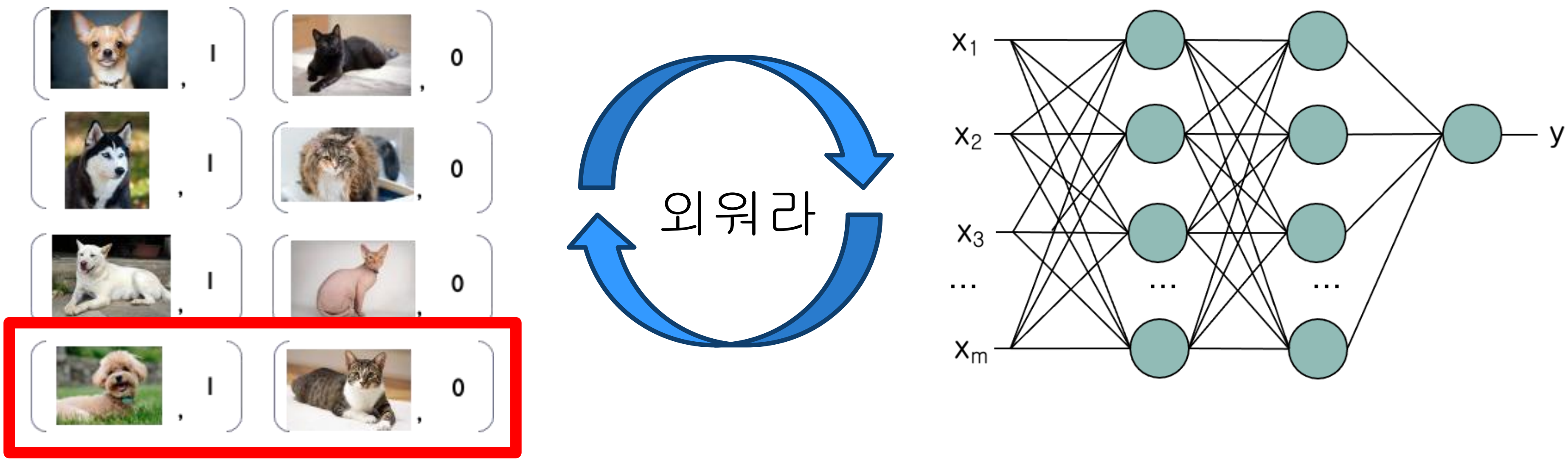
- **Mini-batch**
  - Batch size = 2





Mini-batch Gradient Descent

- **Mini-batch**
  - Batch size = 2



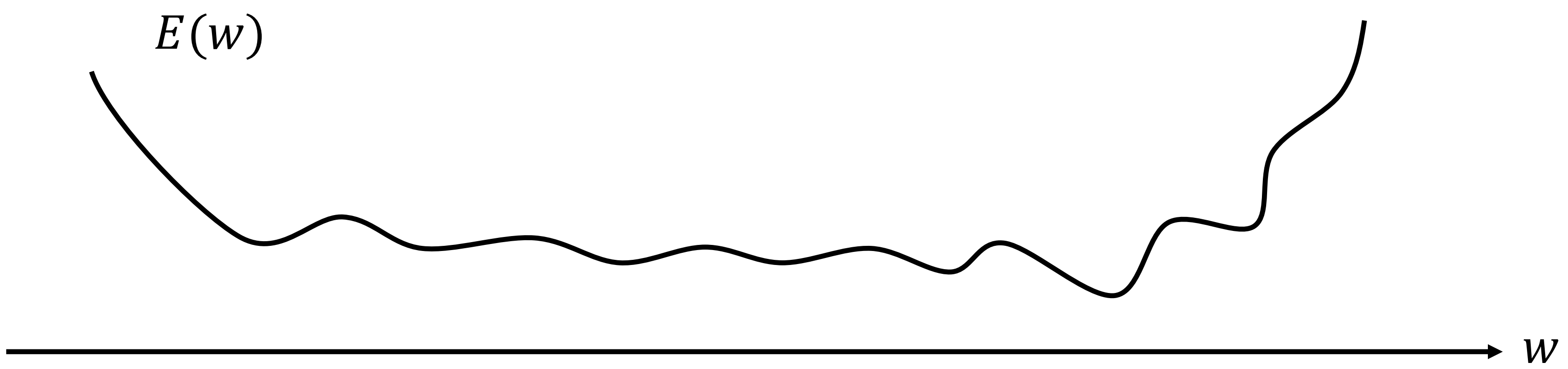
Stochastic Gradient Descent

- **Usual Batch Size**
  - Dependent on datasets from several thousands to several tens
- **Advantage**
  - Good estimation of real gradient
  - High throughput: may use the large number of cores at once in a GPU.
  - Faster convergence: Good estimation + High throughput
- **Disadvantage**
  - Inaccurate: dataset with large variances

그 외 주제

Gradient Descent

- **Local Minimum을 찾음**

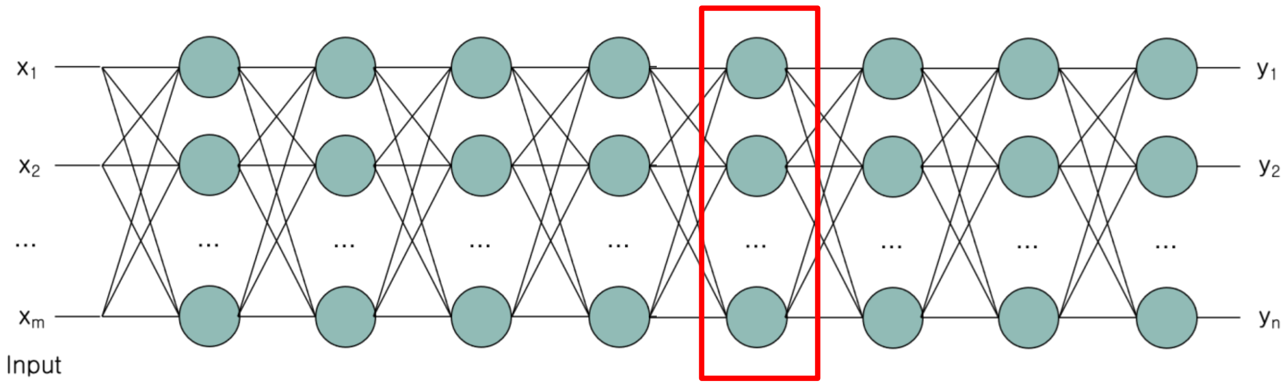


  - 더 좋은 local mimimum을 찾을 수 없을까?

- **해결방법: Momentum, Adam optimizer 사용**

Batch Normalization

- **Deep Learning**



  - NN이 deep 해 질수록
  - 한 레이어에 있는 노드들의 출력값의 범위가 매우 넓게 됨
  - 출력값의 범위가 넓을수록 학습이 느려지고
    출력 결과도 불안정 해짐

- 이를 해결하는 기법이 **Batch Normalization**