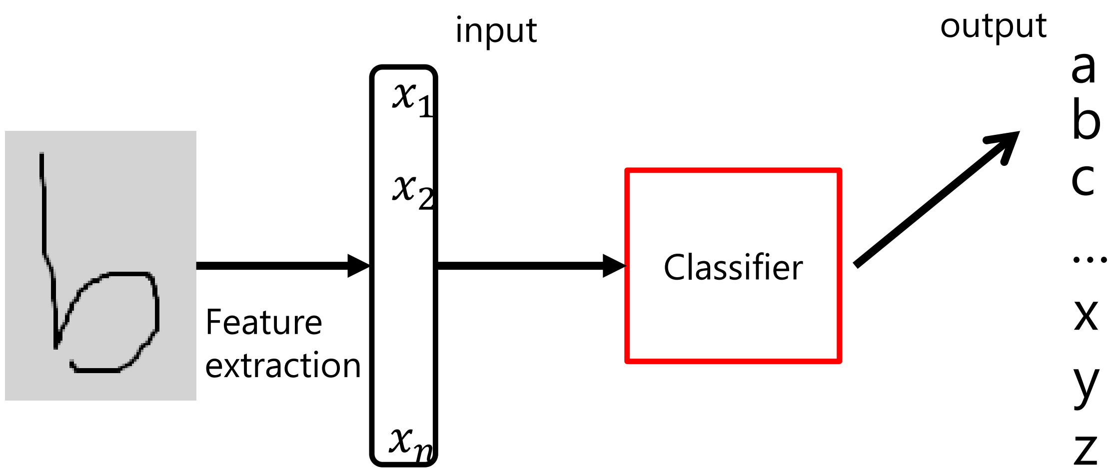
Introduction to Neural Networks

CUONG TUAN NGUYEN
SEIJI HOTTA
MASAKI NAKAGAWA
Tokyo University of Agriculture and Technology

Pattern classification

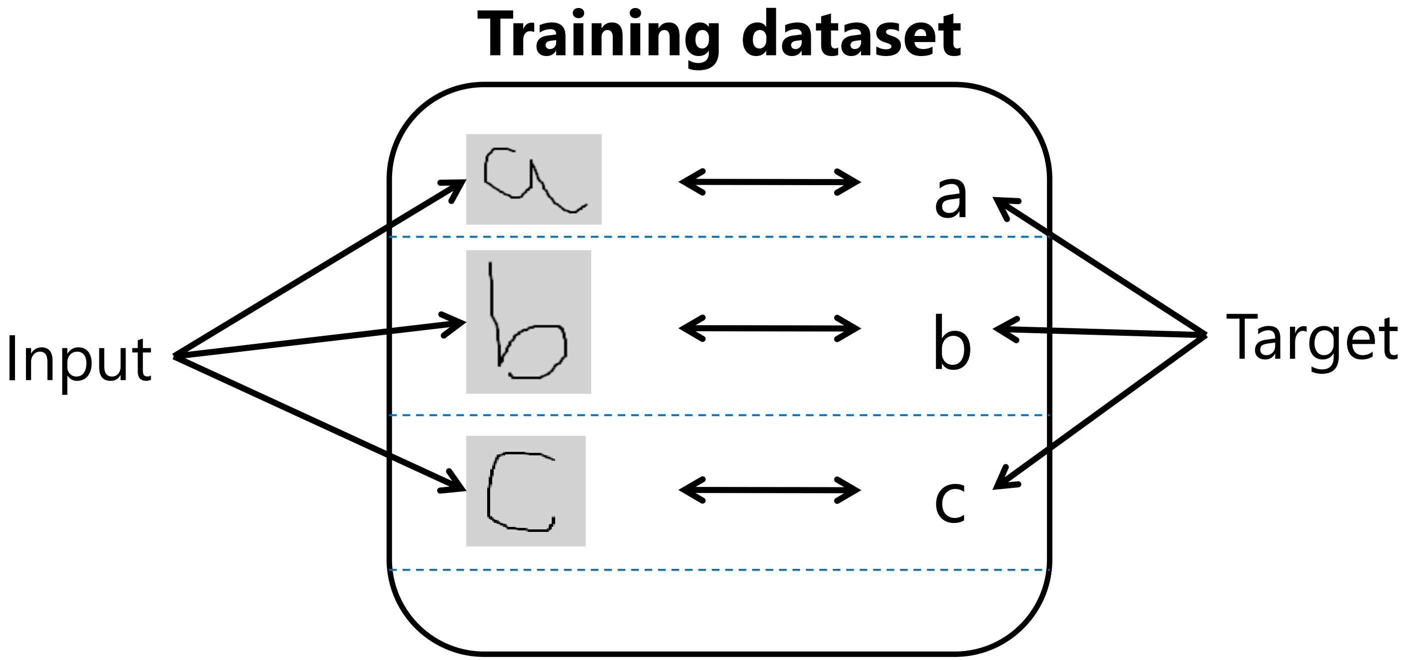
- Which category of an input?
  - Example: Character recognition for input images
- Classifier
  - Output the category of an input



Supervised learning

- Learning by a training dataset:
  pair<input, target>
- Testing on unseen dataset

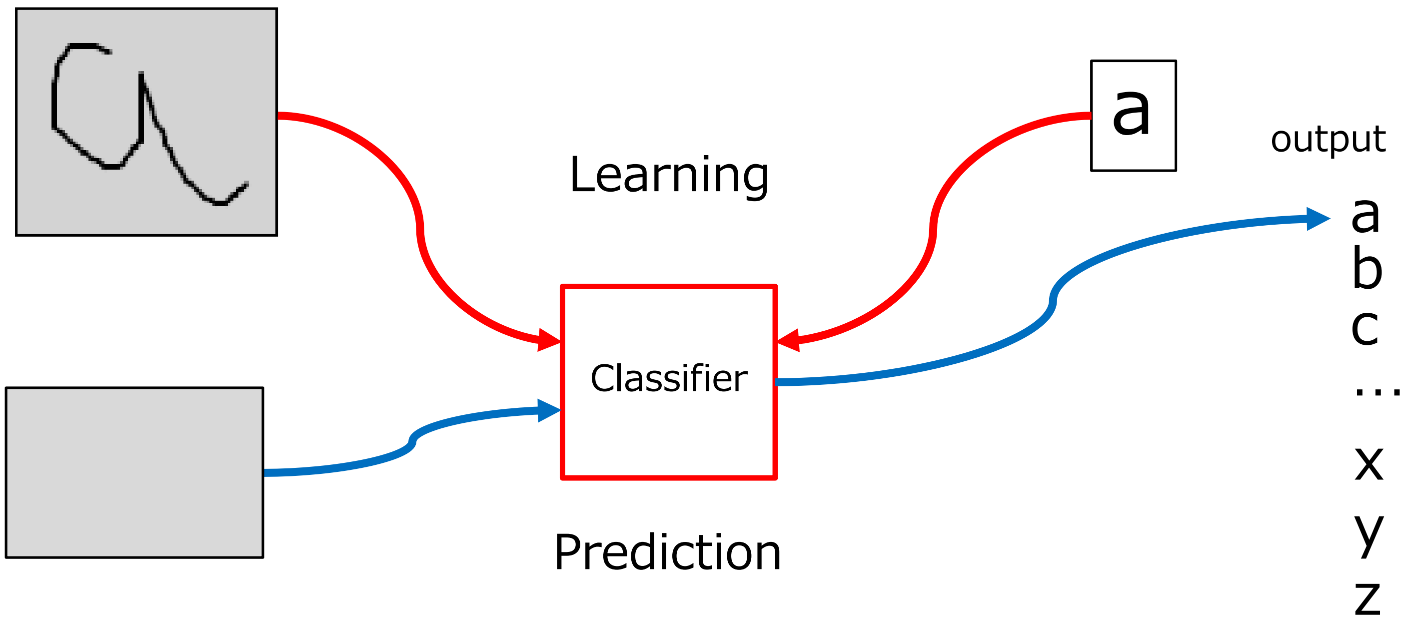
→ **Generalization ability**



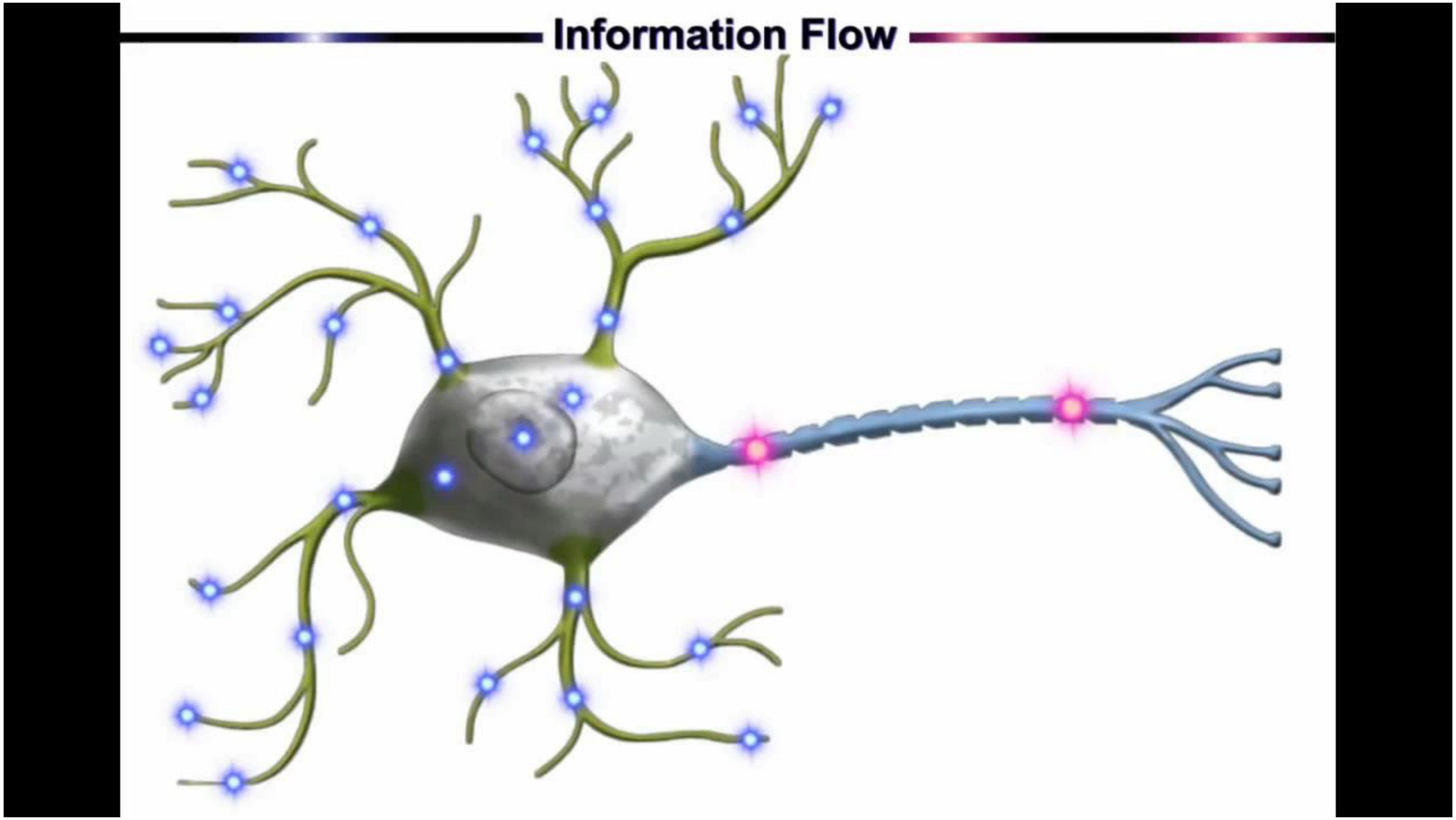
Supervised learning

- Learning by a training dataset:
  pair<input, target>
- Testing on unseen dataset

→ **Generalization ability**

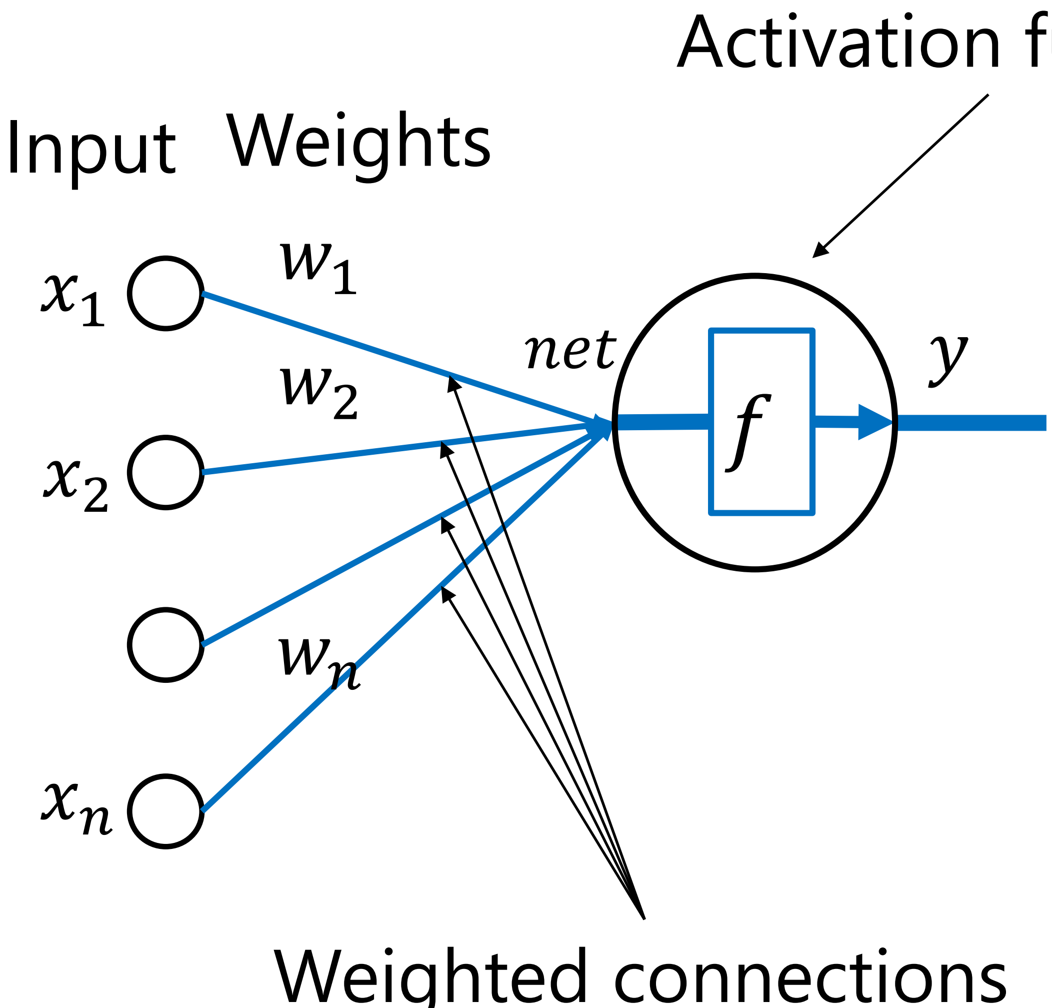


Human neuron



Neural Networks, A Simple Explanation
https://www.youtube.com/watch?v=gcK_5x2KsLA

Artificial neuron

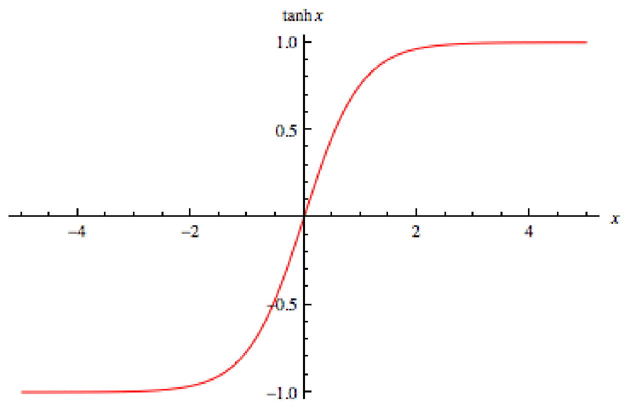


$$net = \sum_{i=1}^{n} x_i w_i$$

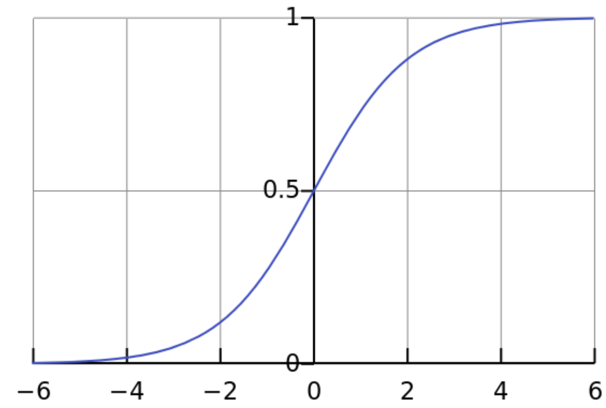
$$y = f(net)$$

Activation function

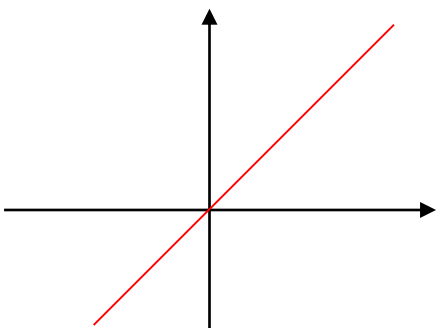
- Controls when neuron should be activated

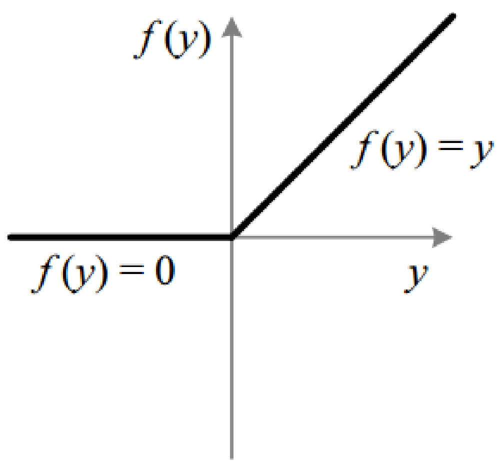


tanh

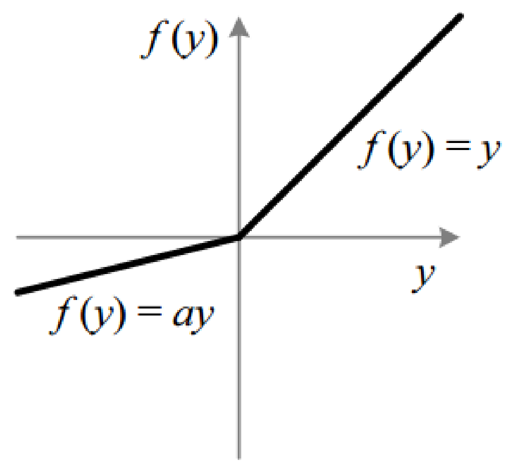


sigmoid


linear



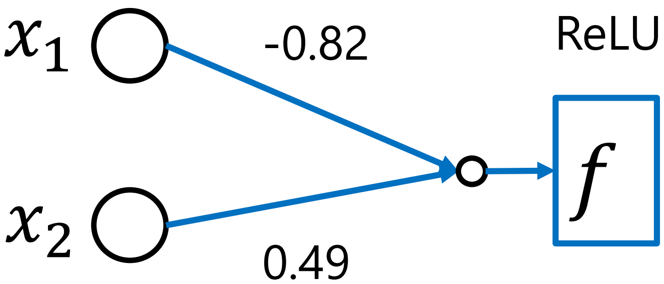
ReLU

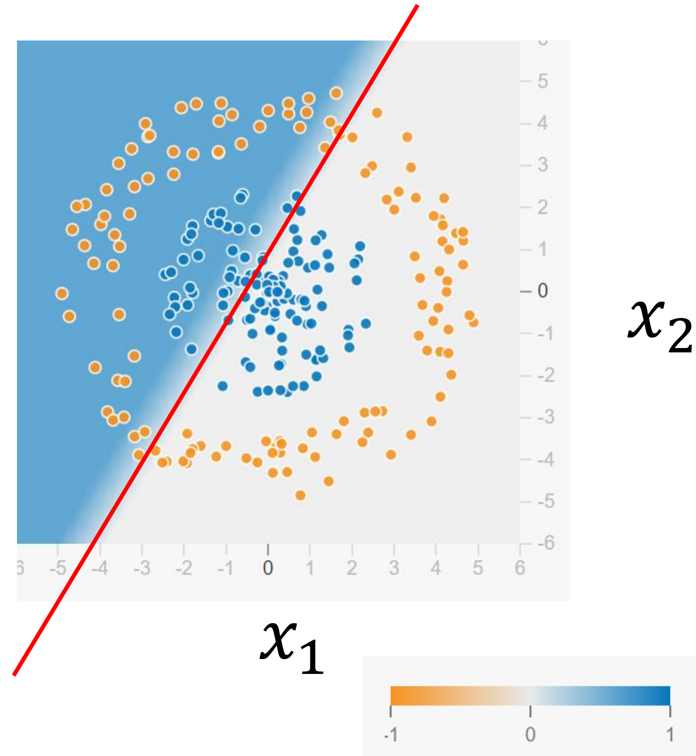


Leaky ReLU

Weighted connection + Activation function

- A neuron is a feature detector: it is activated for a specific feature



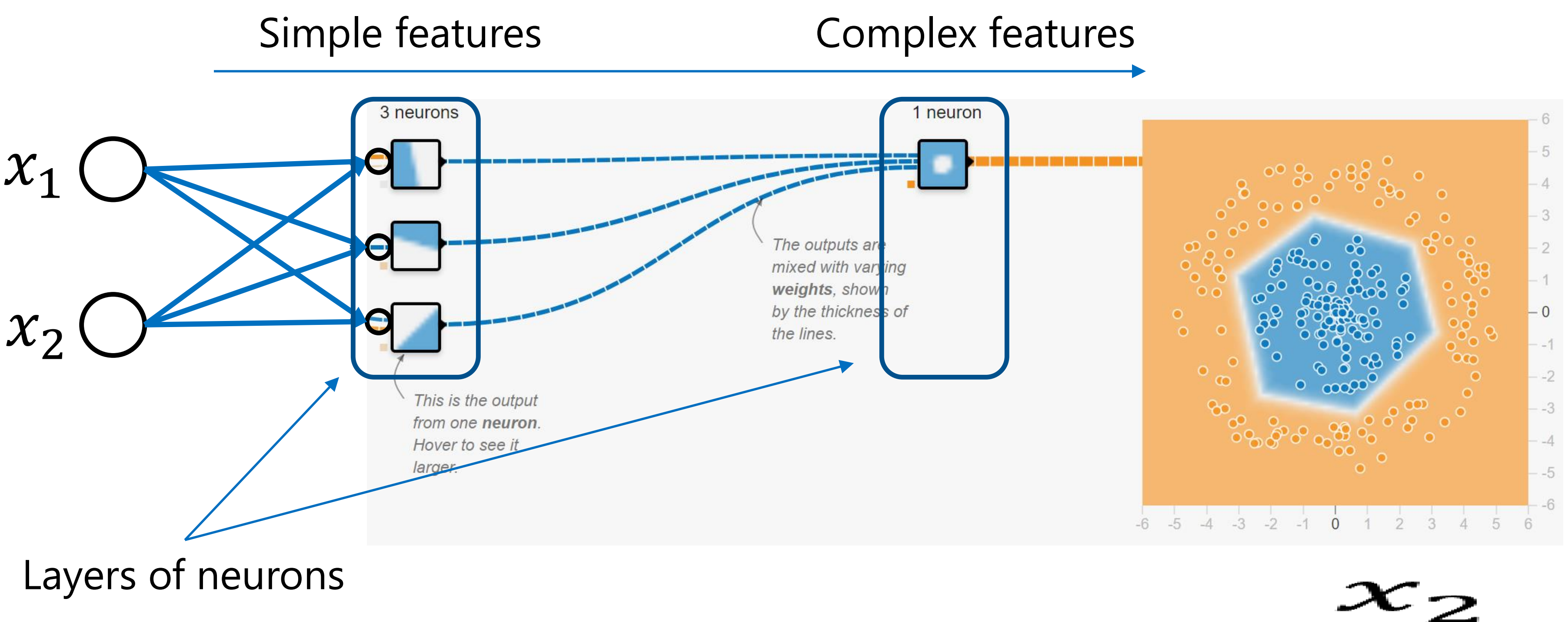


$$-0.82x_1 + 0.49x_2 = 0$$

Generated by:
https://playground.tensorflow.org

Multi-layer perceptron (MLP)

- Neurons are arrange into layers
  - Each neuron in a layer share the same input from preceding layer



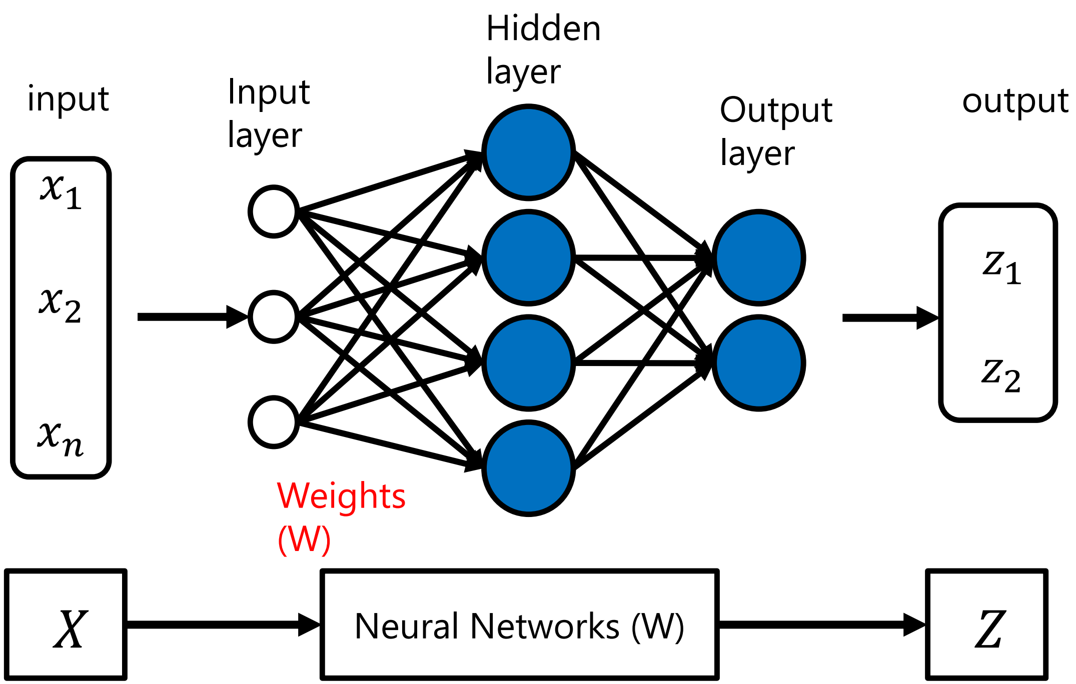
Generated by: https://playground.tensorflow.org

MLP as a learnable classifier

- Output corresponding to an input is constrained by weighted connection
  - These weights are learnable (adjustable)

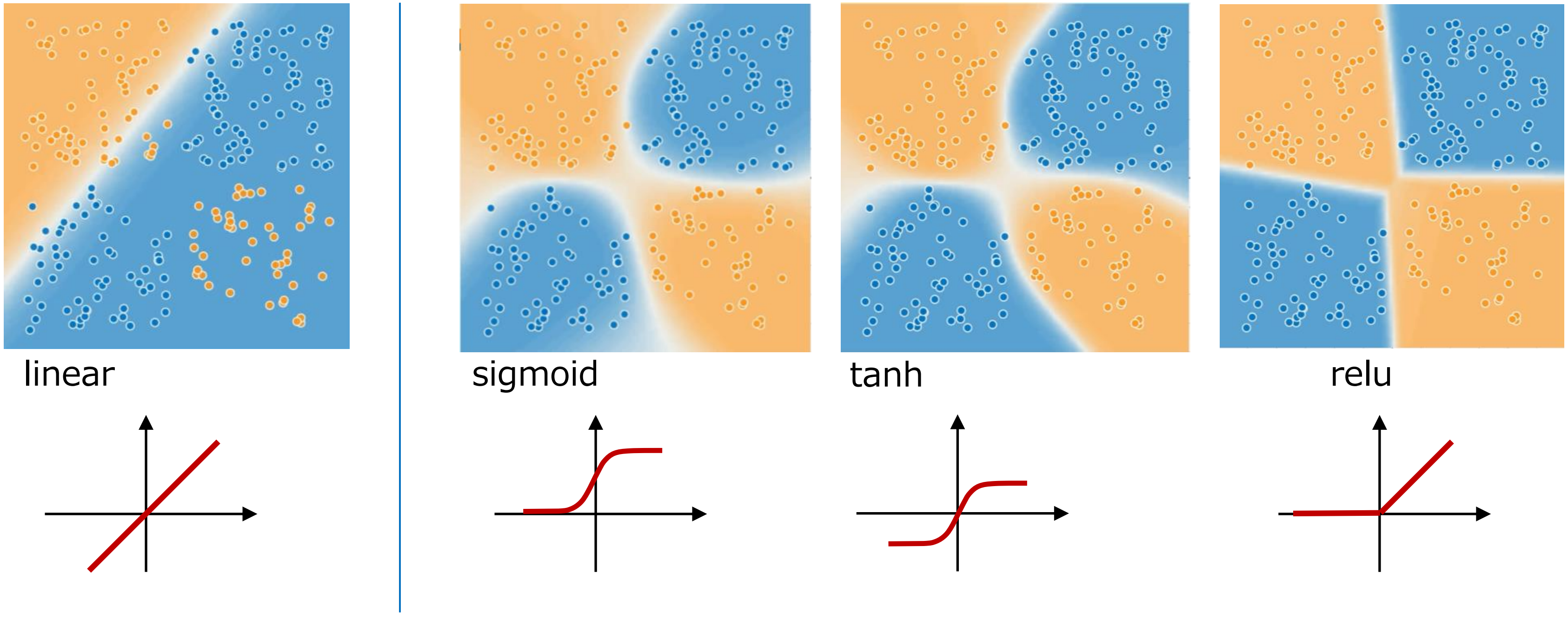
$$\boxed{Z = h(X, W)}$$

Output Input Weight



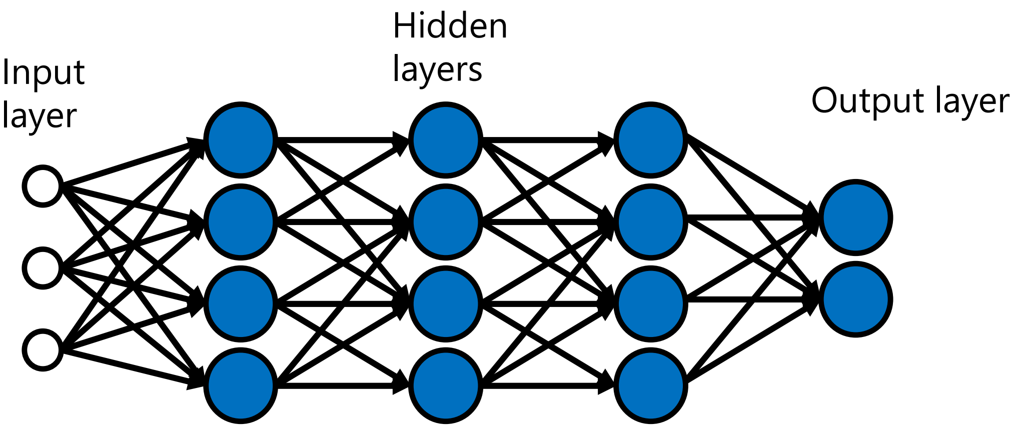
Learning ability of neural networks

- Linear vs Non-linear
  - With linear activation function: can only learn linear function
  - With non-linear activation function: can learn non-linear function



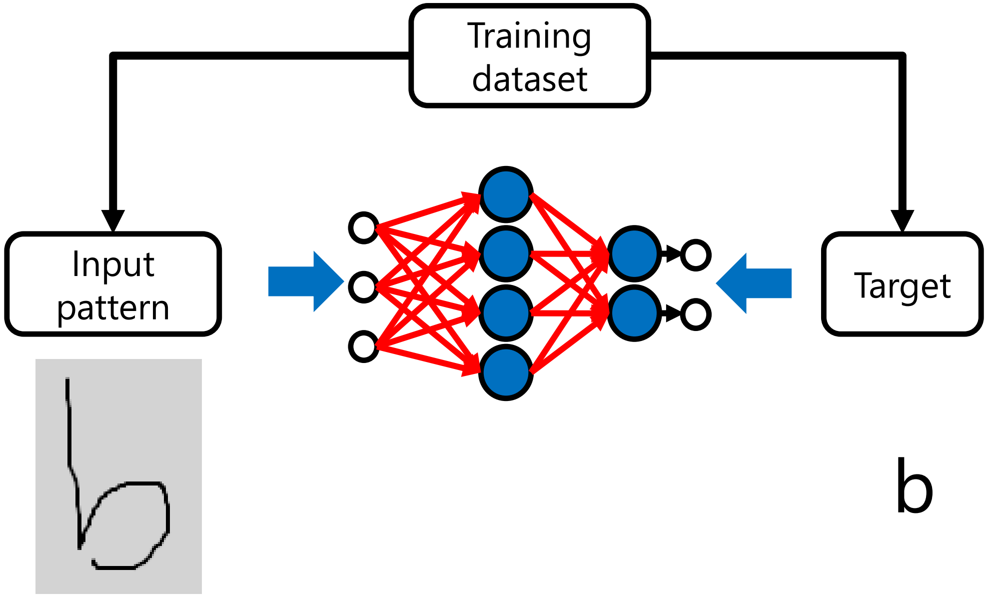
Learning ability of neural network

- Universal approximation theorem [Hornik, 1991]: *MLP can learn arbitrary function with a single hidden layer*
  - For complex functions, however, may require large hidden layer
- Deep neural network
  - Contains many hidden layers, can extract complex features



Learning in Neural Networks

- Weighted connection is tuned using the training data <input, target>
  - Objective: Networks could output correct targets corresponding to inputs



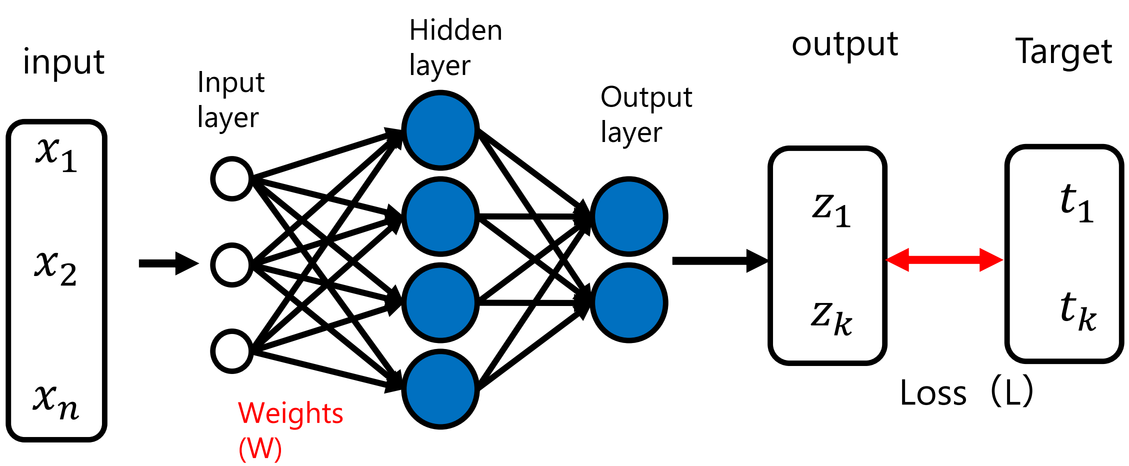
Learning in Neural Networks

- Loss function (objective function)
  - Difference between output and target
- Learning: optimization process
  - Minimize the loss (make output match target)

Loss Target Output

$$\begin{aligned} L &= |T - Z| \\ &= |T - h(X, W)| \\ &= l(W) \end{aligned}$$

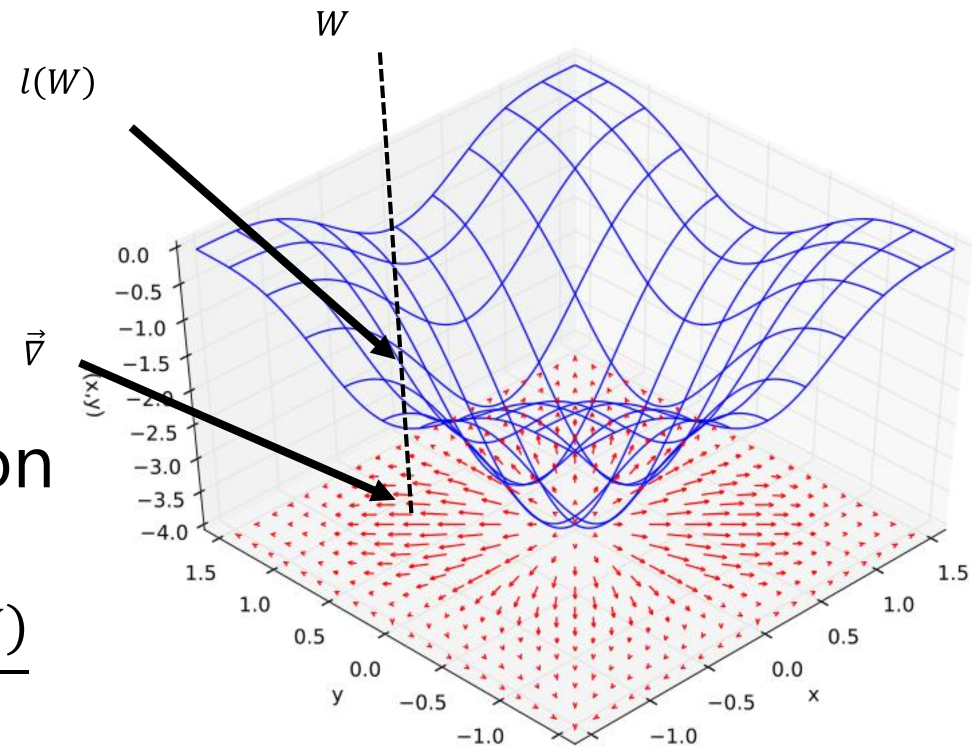
Weights Input



Learning in Neural Networks

- Gradient vector of l for W : $\nabla_W l$

$$\nabla_W l = \frac{\partial l(W)}{\partial W}$$



- Weight update

Reverse gradient direction

$$W_{update} = W_{current} - \eta \frac{\partial l(W)}{\partial W}$$

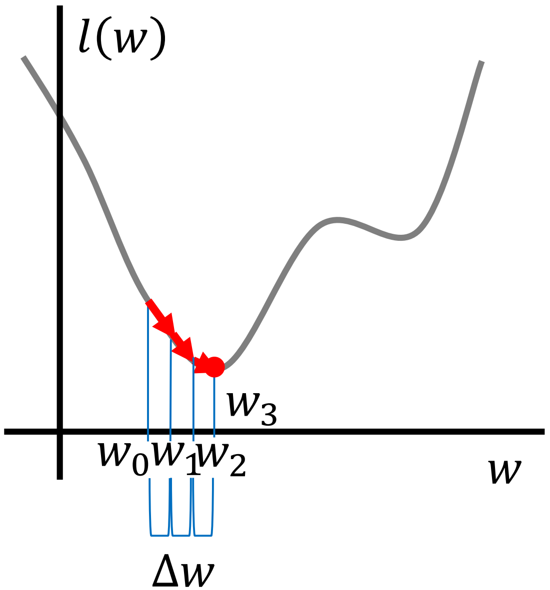
η:learning rate

Loss function

- Logistic regression
- Probabilistic loss function
  - Binary entropy
  - Cross entropy
- Multimodal
- Mean square error

Learning & converge

- By update weight using gradient, loss is reduced and converge to minima

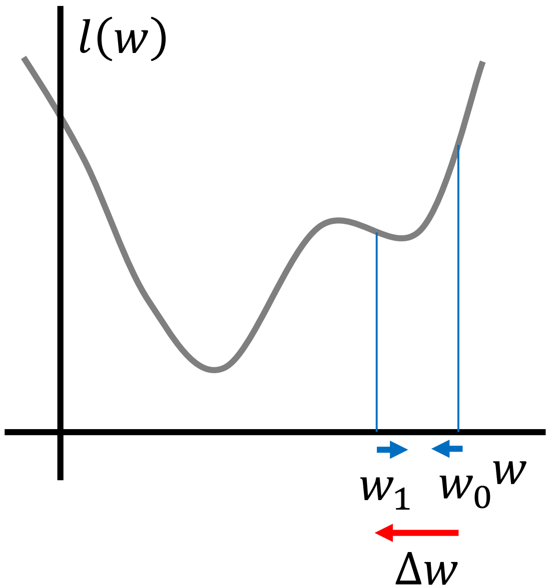


Learning through all training samples

- After updating weights, new training samples is fed to the networks to continue learning
- When all training samples is learnt, networks has completed one *epoch*. Networks must run through many epochs to converge.
- Weight update strategy
  - Stochastic gradient descent (SGD)
  - Batch update
  - Mini-batch

Momentum Optimizer

- Learning may stuck on a local minima.
- Momentum: Δw retains the latest optimizing direction. It may help the optimizer overcome the local minima.

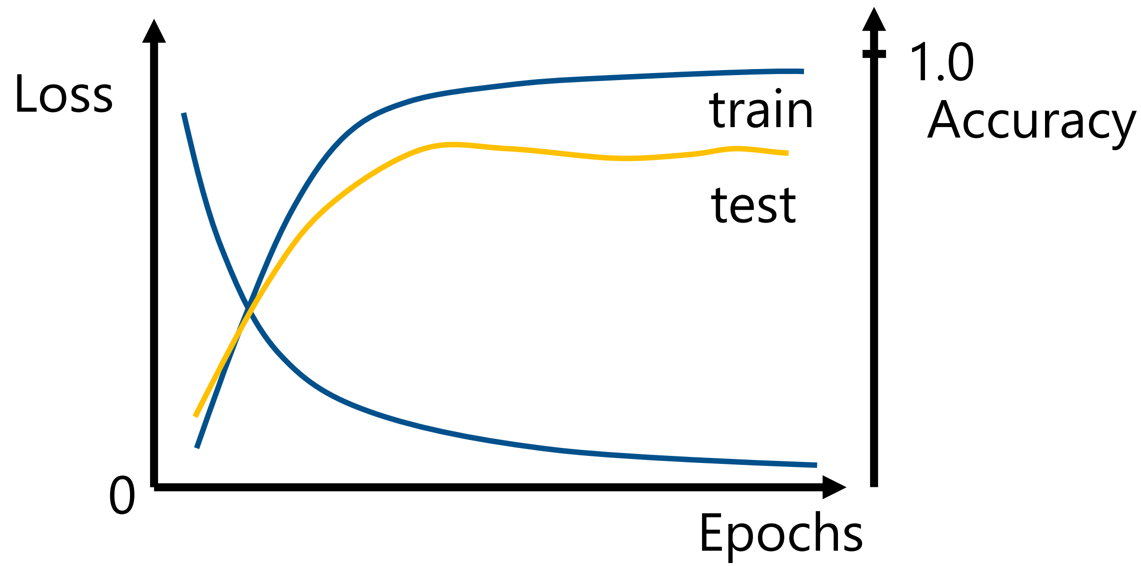


$$W_{update} = W_{current} - \eta \frac{\partial l(W)}{\partial W} + \alpha \Delta w$$

η: learning rate
α: momentum parameter

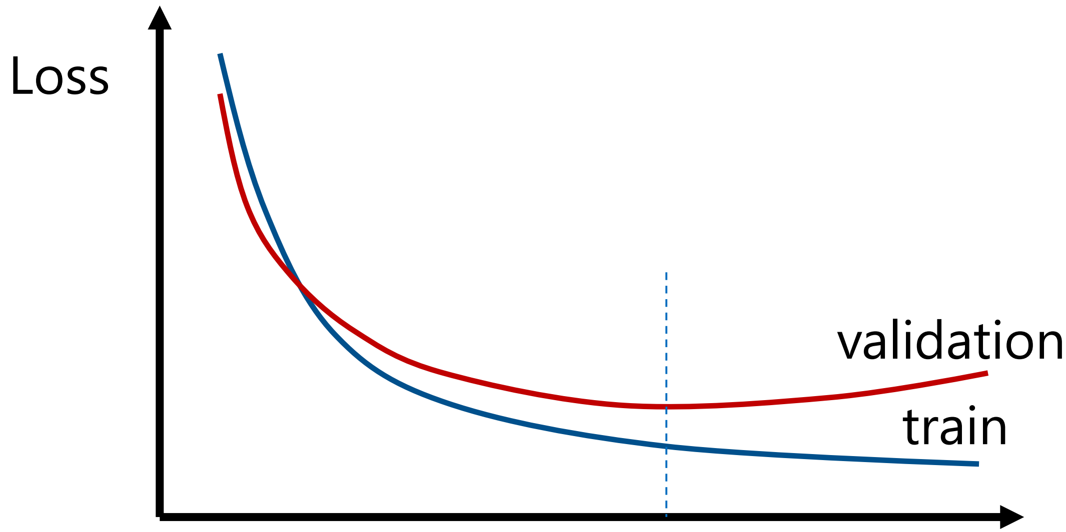
Overfitting & Generalization

- While training, model complexity increases through each epoch
  - Overfitting:
    - Model is over-complex
    - Poor generalization: good performance on train set but poor on test set



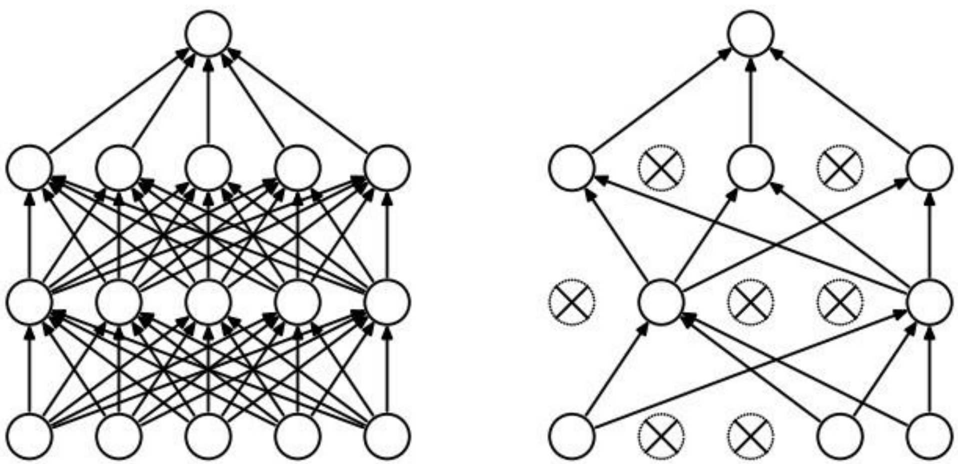
Prevent overfitting: Regularization

- Weight decaying
- Weight noise
- Early stopping
  - Evaluate performance on a validation set
  - Stop while there is <u>no improvement</u> on validation set



Prevent overfitting: Regularization

- Dropout
  - Randomly drop the neurons with a predefined probability
  - Good regularization: large ensembles of networks
  - Bayesian perspective



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

Adaptive learning rate

- Adam optimizer

Practice

- GPU implementation
  - Keras + Tensorflow