Kissipo Learning for Deep Learning

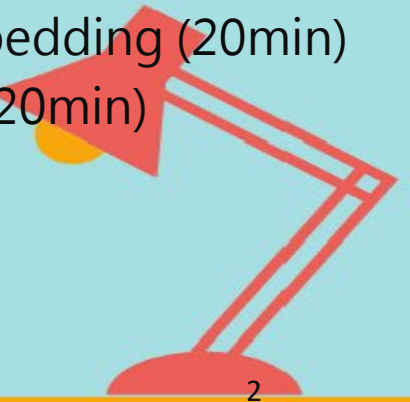
Topic 10: Deep Learning basics (20min)

Hsueh-Ting Chu

KLDL-W3-10

Topics

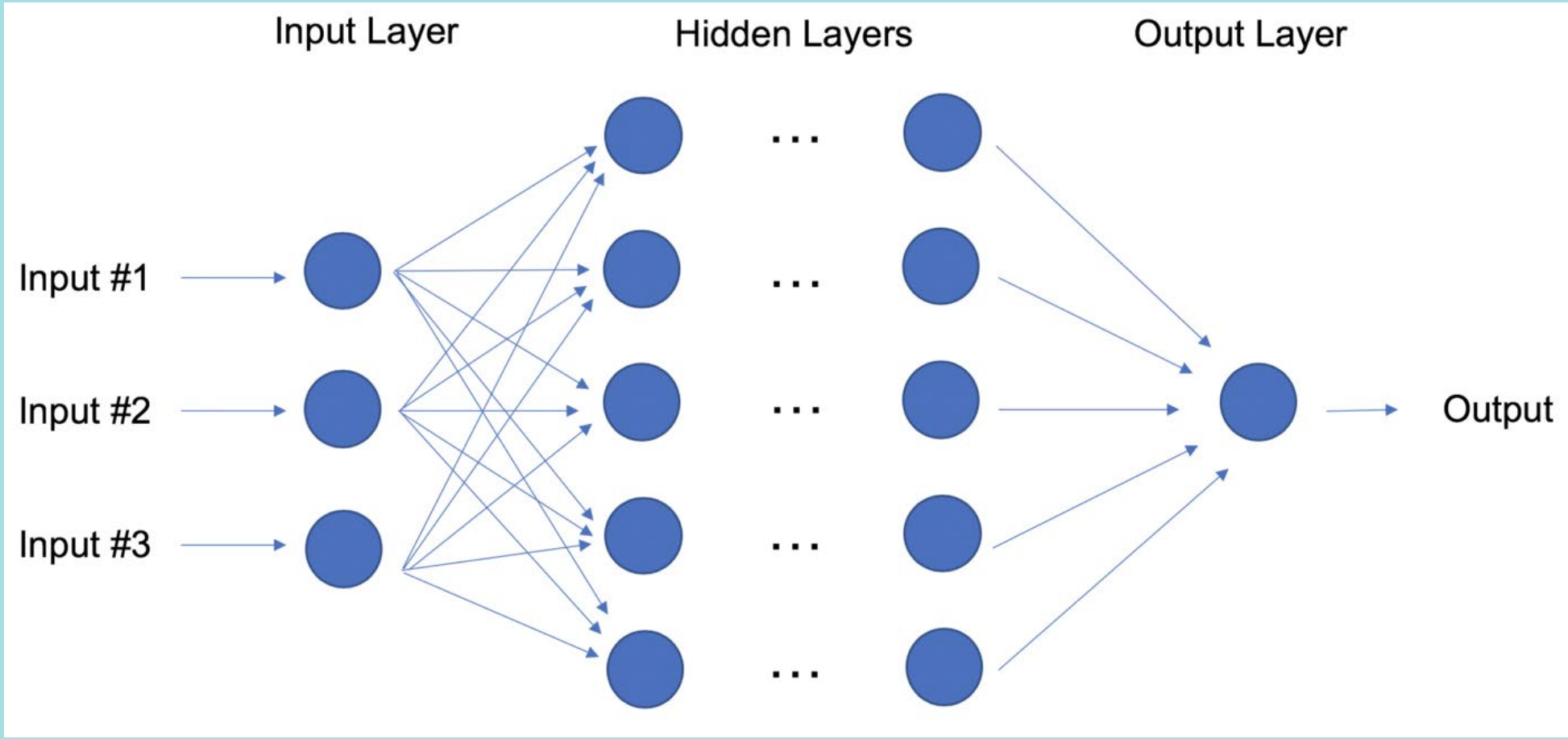
- Topic 01: Introduction to Deep Learning (20min)
- Topic 02: Kissipo Learning for Deep Learning (20min)
- Topic 03: Python quick tutorial (20min)
- Topic 04: Numpy quick tutorial (15min)
- Topic 05: Pandas quick tutorial (15min)
- Topic 06: Scikit-learn quick tutorial (15min)
- Topic 07: OpenCV quick tutorial (15min)
- Topic 08: Image Processing basics (20min)
- Topic 09: Machine Learning basics (20min)
- Topic 10: Deep Learning basics (20min)
- Topic 11: TensorFlow overview (20min)
- Topic 12: CNN with TensorFlow (20min)
- Topic 13: RNN with TensorFlow (20min)
- Topic 14: PyTorch overview (20min)
- Topic 15: CNN with PyTorch (20min)
- Topic 16: RNN with Pytorch (20min)
- Topic 17: Introduction to AOI (20min)
- Topic 18: AOI simple Pipeline (A) (20min)
- Topic 19: AOI simple Pipeline (B) (20min)
- Topic 20: Introduction to Object detection (20min)
- Topic 21: YoloV5 Quick Tutorial (20min)
- Topic 22: Using YoloV5 for RSD (20min)
- Topic 23: Introduction to NLP (20min)
- Topic 24: Introduction to Word Embedding (20min)
- Topic 25: Name prediction project (20min)



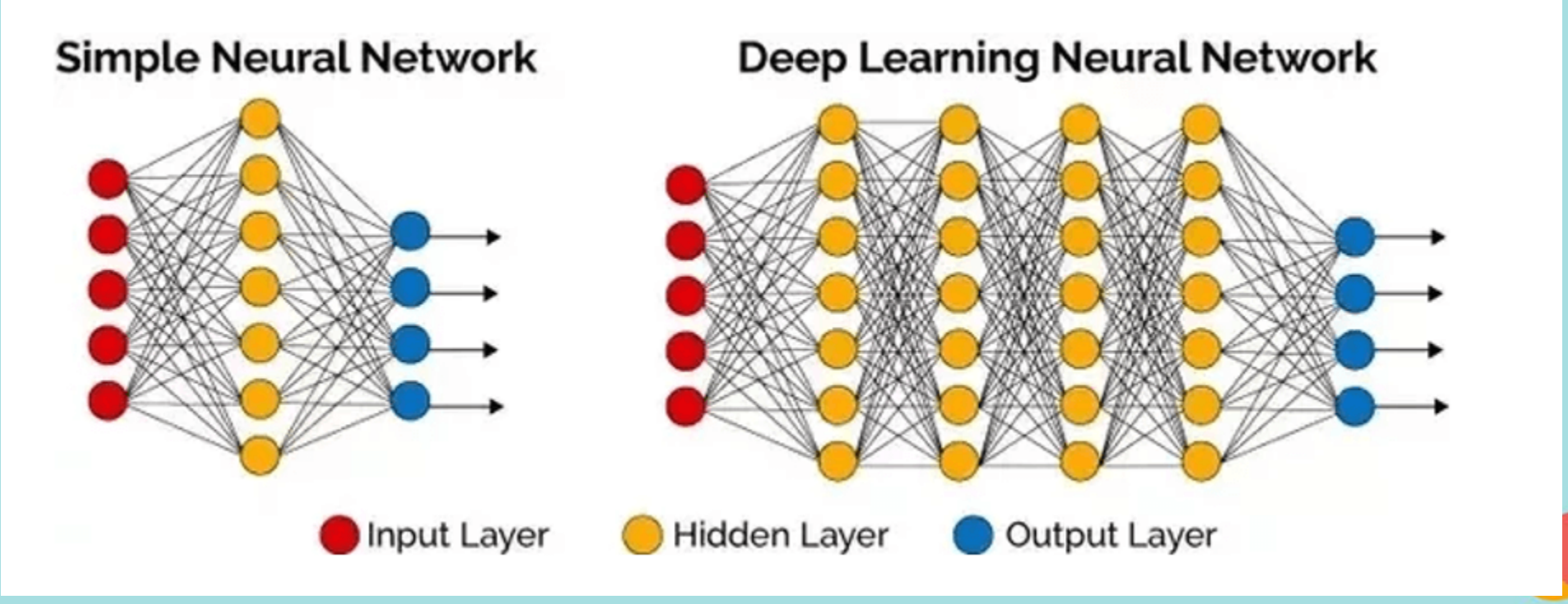
Content

- Topic 10: Deep Learning basics (20min)
  - Neural Network and Deep Learning
  - Neural Network Playground
  - Training and Loss
  - Gradient Descent
  - Optimizer
  - Backpropagation
  - Overfitting and regularization
  - Activation function

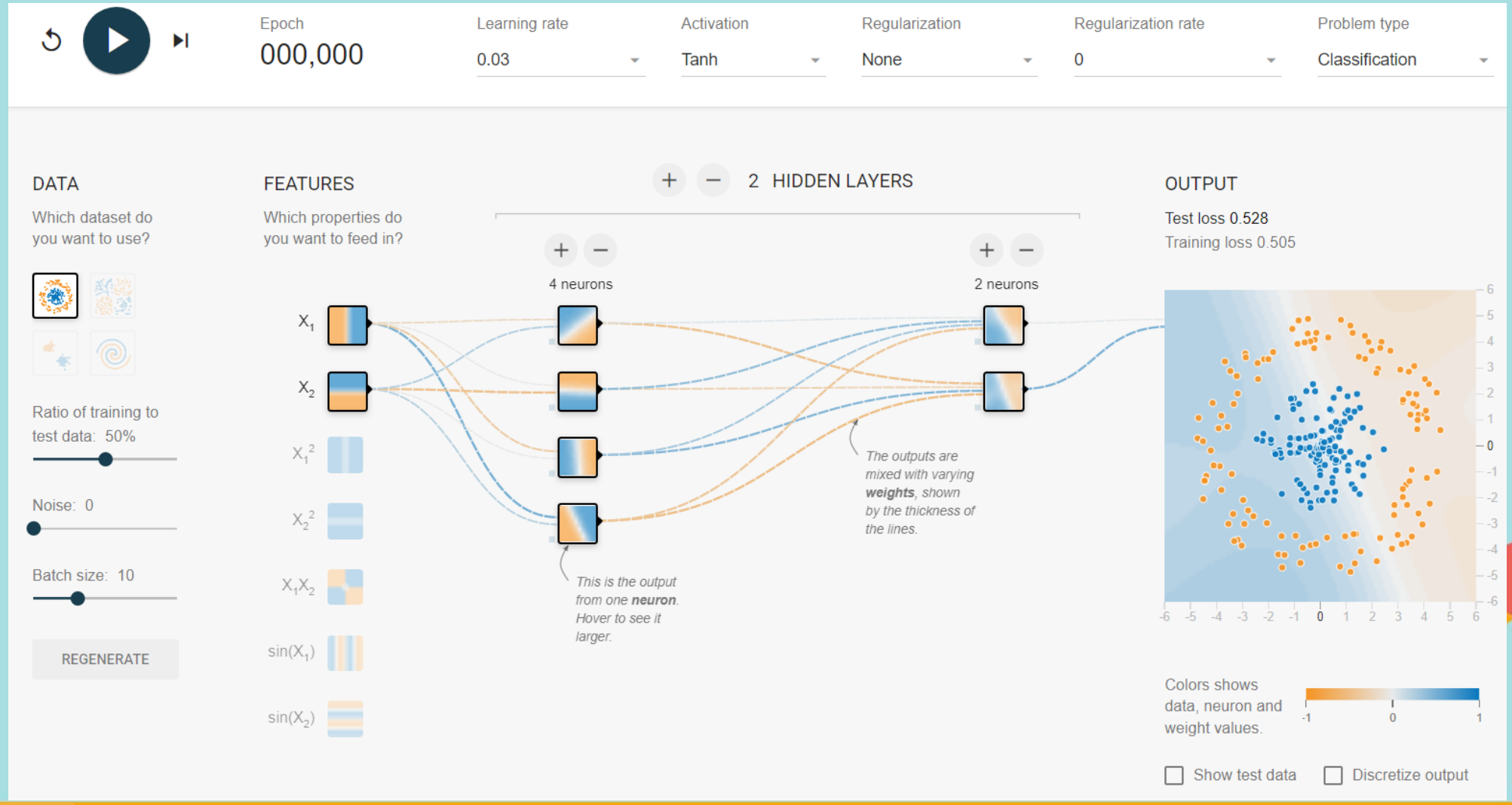
Neural Network



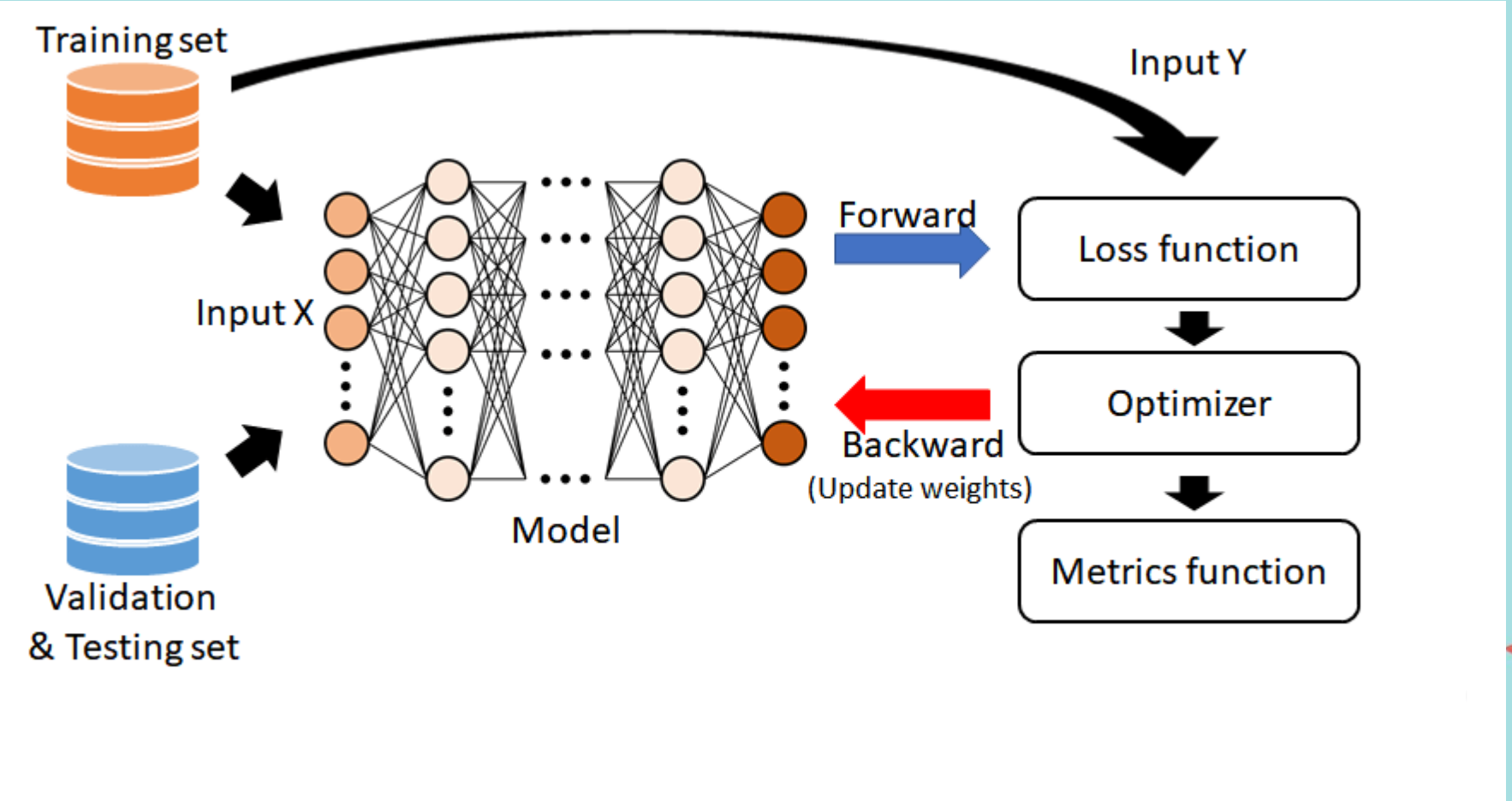
Deep Learning



Neural Network Playground



Training a Deep learning(NN) model

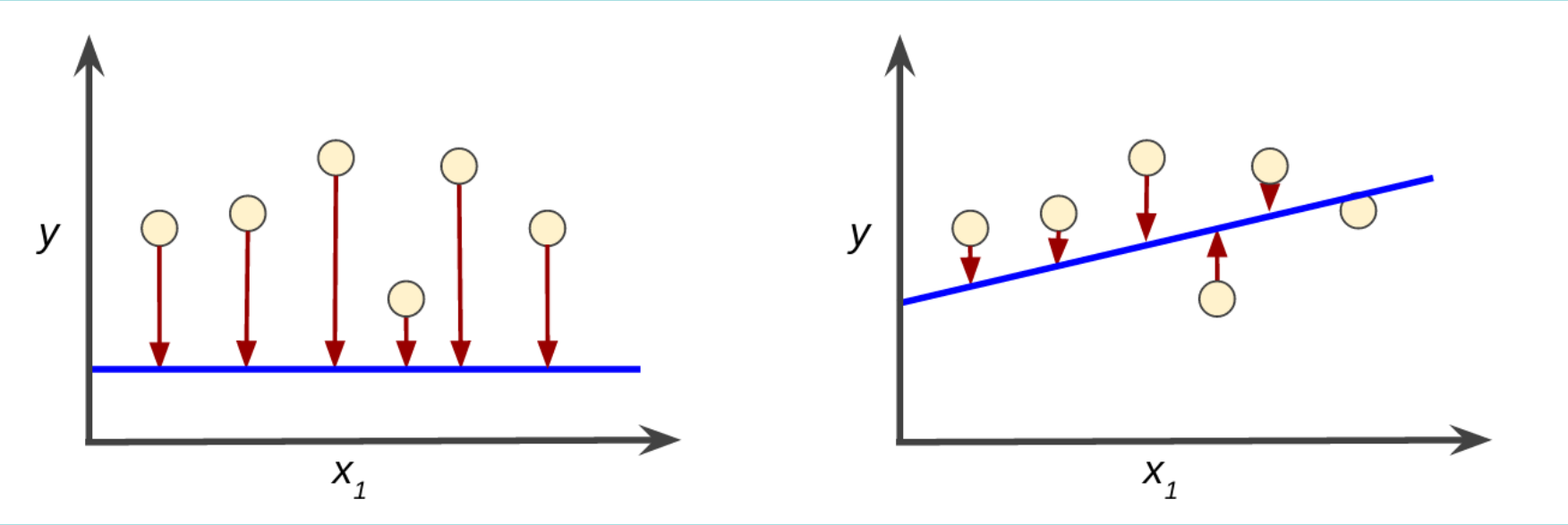


Types of Artificial Neural Networks

- Feed Forward Neural Networks
- Convolutional neural network (CNN)
- Recurrent neural network (RNN)
- Sequence-to-sequence model(encoder-decoder)

Training and Loss

- Training a model means learning (determining) good values for all the weights and the bias from labeled examples.



Error

- loss function=Real value – Predicted value
- Mean square error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2$$

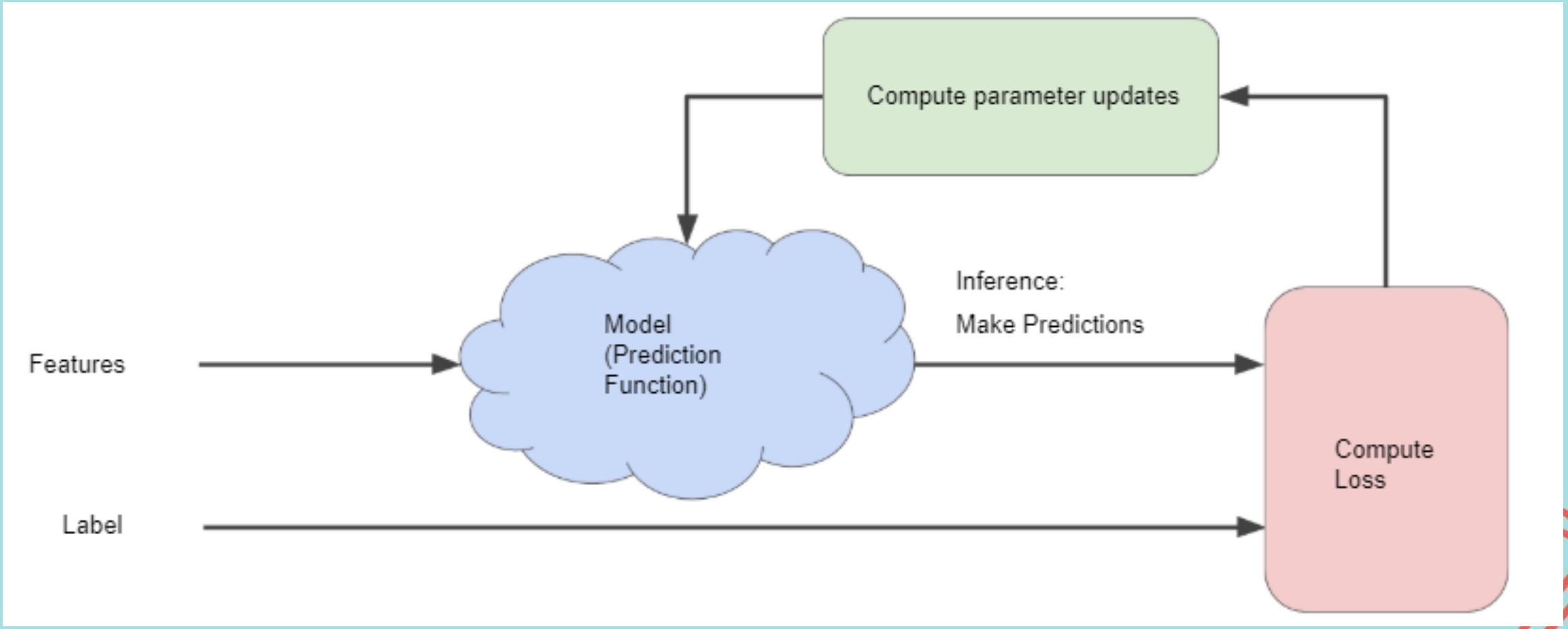
- Mean absolute error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y}_i\right|$$

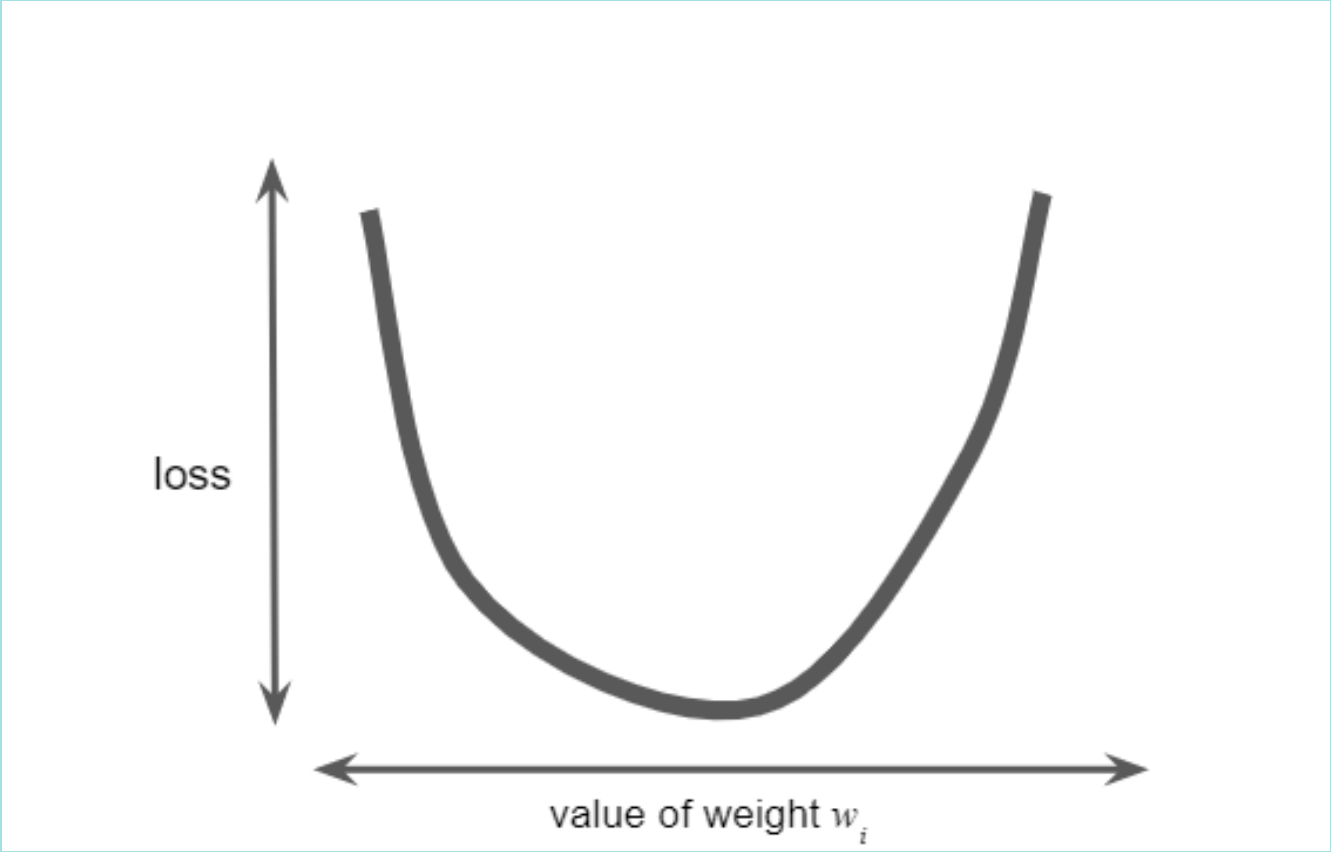
- Cross-entropy

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

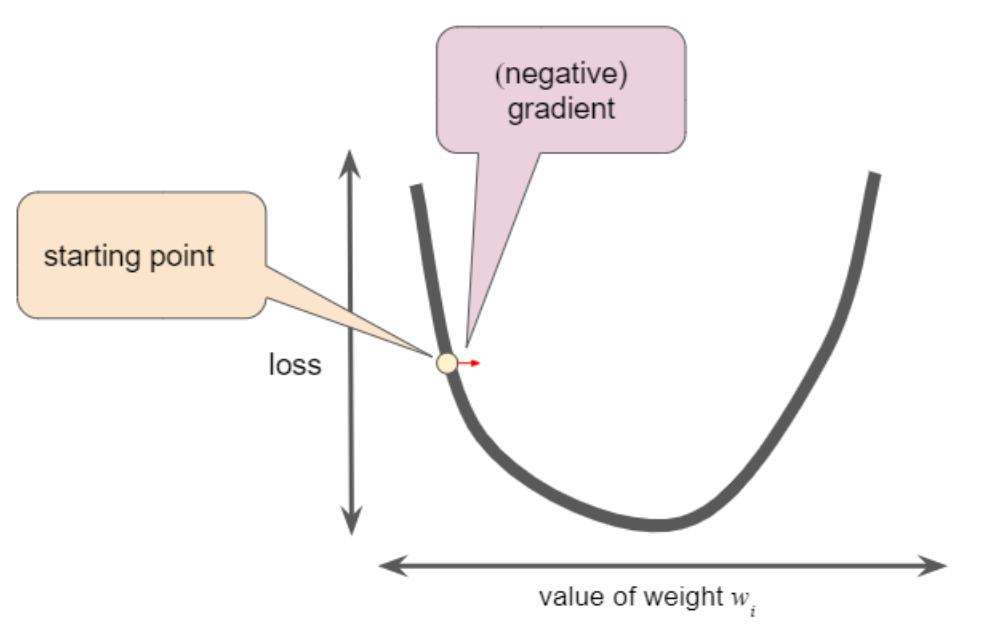
Reducing Loss: An Iterative Approach

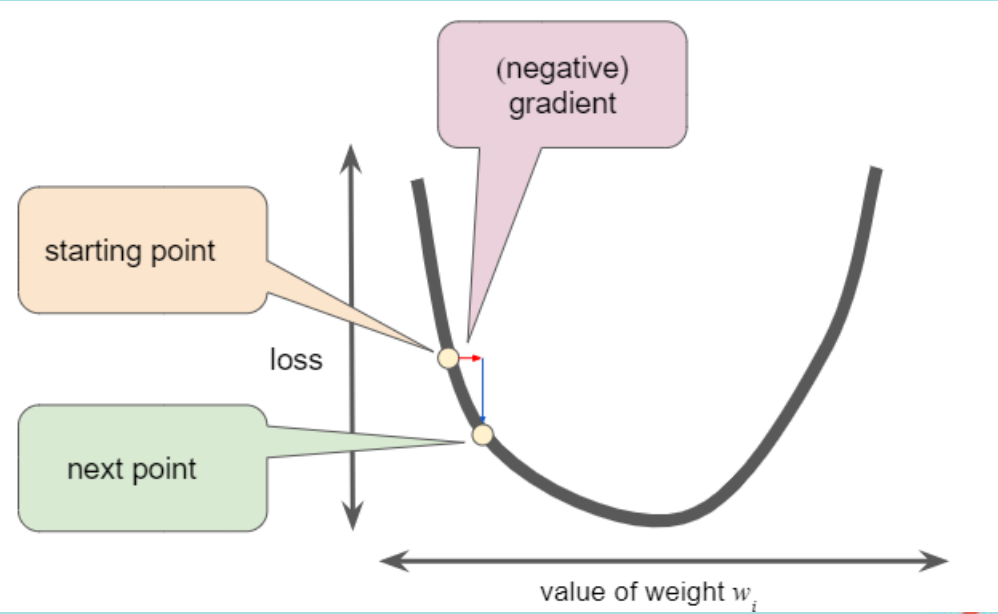


Reducing Loss: Gradient Descent

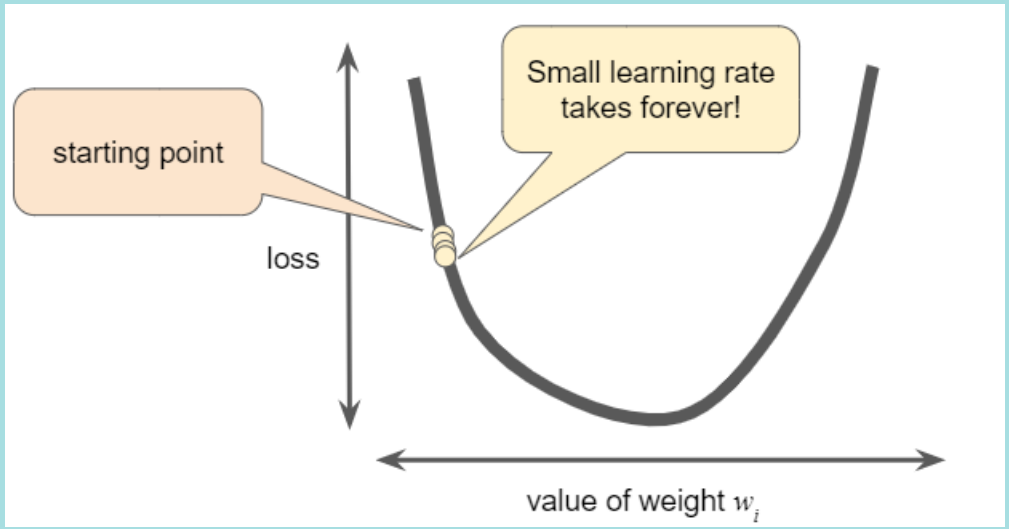


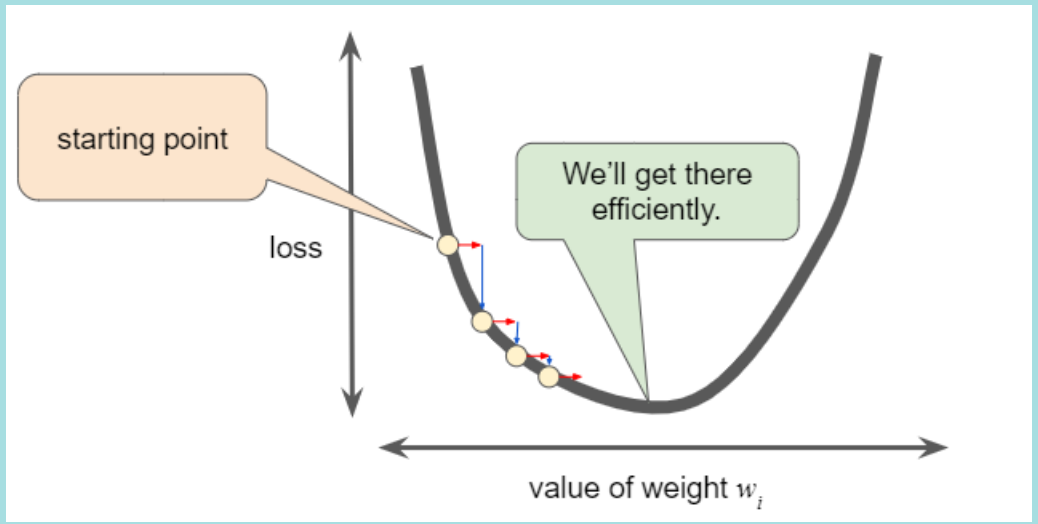
Gradient descent relies on negative gradients

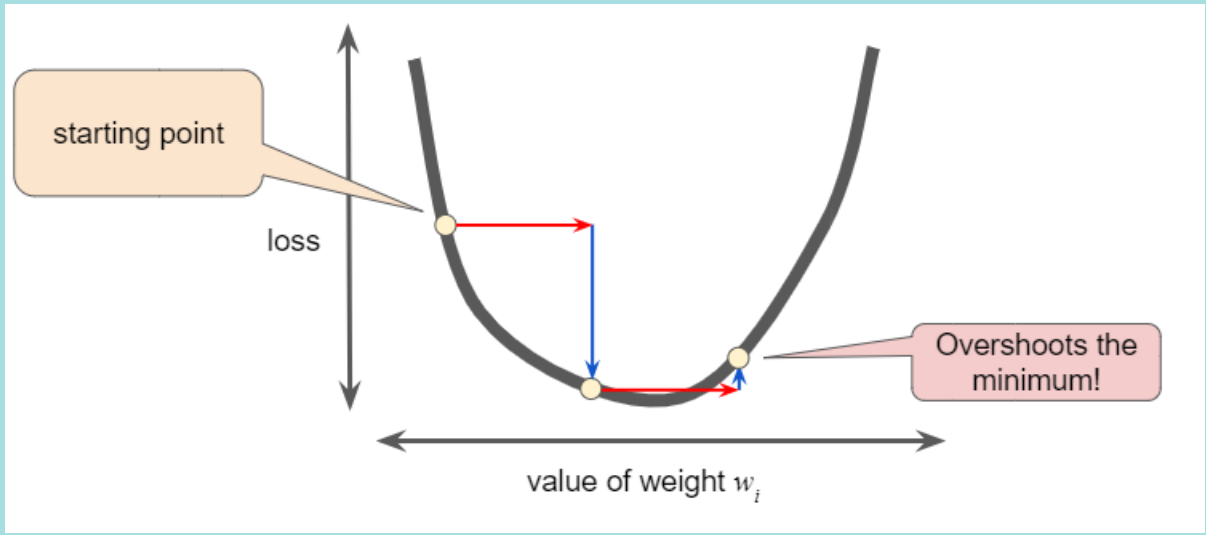




Reducing Loss: Learning Rate







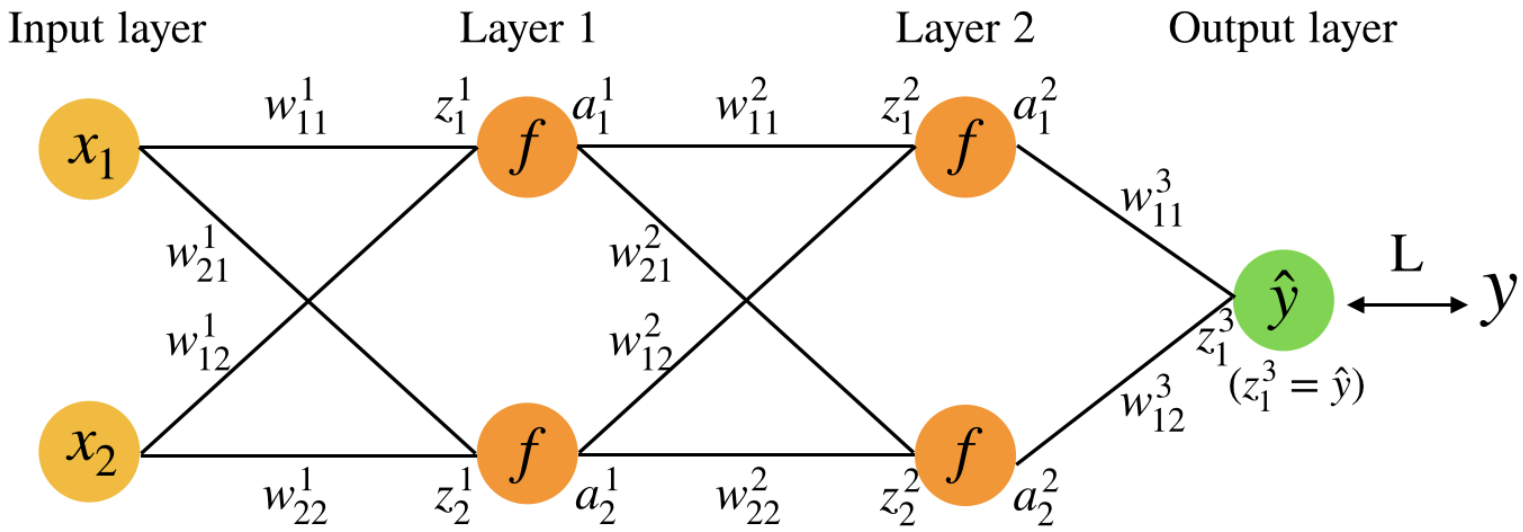
Optimizer

- *SGD-stochastic gradient decent*
- *Momentum*
- *AdaGrad-* Adaptive *gradient*
- *Adam= AdaGrad+ Momentum*
- RMSProp-Root Mean Square Prop

Backpropagation

- Two stages of the backpropagation algorithm: stimulus propagation and weight update.
- **Forward propagation stage**-feed the training input into the network to obtain the predicted values;
- **Backpropagation stage**-Differences the predicted values with the target output corresponding to the training input to obtain the error of the output layer and the hidden layer.

Backpropagation(**BP**)-Notation



w_{11}^1 ← Layer 1 ← Layer 1 Neuron 1 to Input layer Neuron 1

$z_1^1 = x_1 w_{11}^1 + x_2 w_{12}^1$, z_1 : activation function input

$a_1^1 = f(z_1^1)$, f : activation function, a_1^1 : activation output

$\hat{y}$: predict value

Backpropagation(**BP**)

- The gradient of w_{11}^1 :

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial z_1^3}[\sum_{i=1}^{2} \frac{\partial z_1^3}{\partial a_i^2}\frac{\partial a_i^2}{\partial z_i^2}\frac{\partial z_i^2}{\partial a_1^1}]\frac{\partial a_1^1}{\partial z_1^1}\frac{\partial z_1^1}{\partial w_{11}^1} = (\hat{y}-y)[\sum_{i}^{2} w_{1i}^3\sigma'(z_i^2)w_{i1}^2]\sigma'(z_1^1)x_1$$

1. **2.** **3.**

1. $\frac{\partial L}{\partial z_1^3} = \frac{\partial}{\partial \hat{y}}\frac{1}{2}(\hat{y}-y)^2 = (\hat{y}-y),\ (z_1^3 = \hat{y})$

2. $\frac{\partial z_1^3}{\partial a_1^2} = \frac{\partial}{\partial a_1^2}(a_1^2 w_{11}^3 + a_2^2 w_{12}^3) = w_{11}^3,\ (z_1^3 = a_1^2 w_{11}^3 + a_2^2 w_{12}^3)$

3. $\frac{\partial a_1^2}{\partial z_1^2} = \sigma'(z_1^2) = \sigma(z_1^2)(1-\sigma(z_1^2)),\ (a_1^2 = \sigma(z_1^2))$

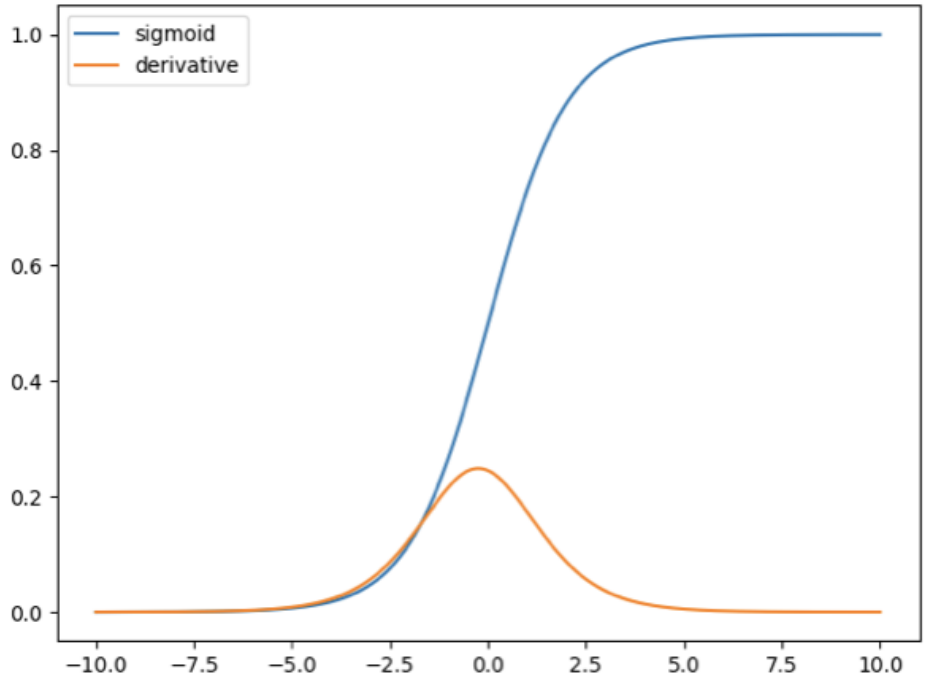
Gradient Problem

- **Gradient vanishing (exploding)**

$$\frac{\partial L}{\partial w_{11}^{1}} = (\hat{y} - y)[\sum_{i}^{2} w_{1i}^{3}\sigma'(z_i^2)w_{i1}^{2}]\sigma'(z_1^1)x_1$$

1. 2. 3.

1. $(\hat{y} - y)$: Defined by your loss function .
2. w : Defined by your initialization weight .
3. $\sigma'(z)$: Defined by your activation function .

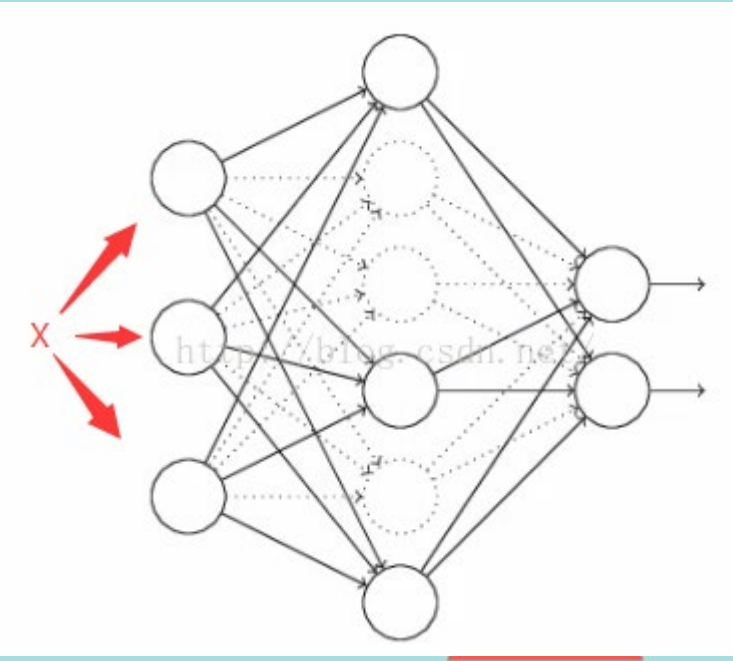


Stochastic Gradient Descent

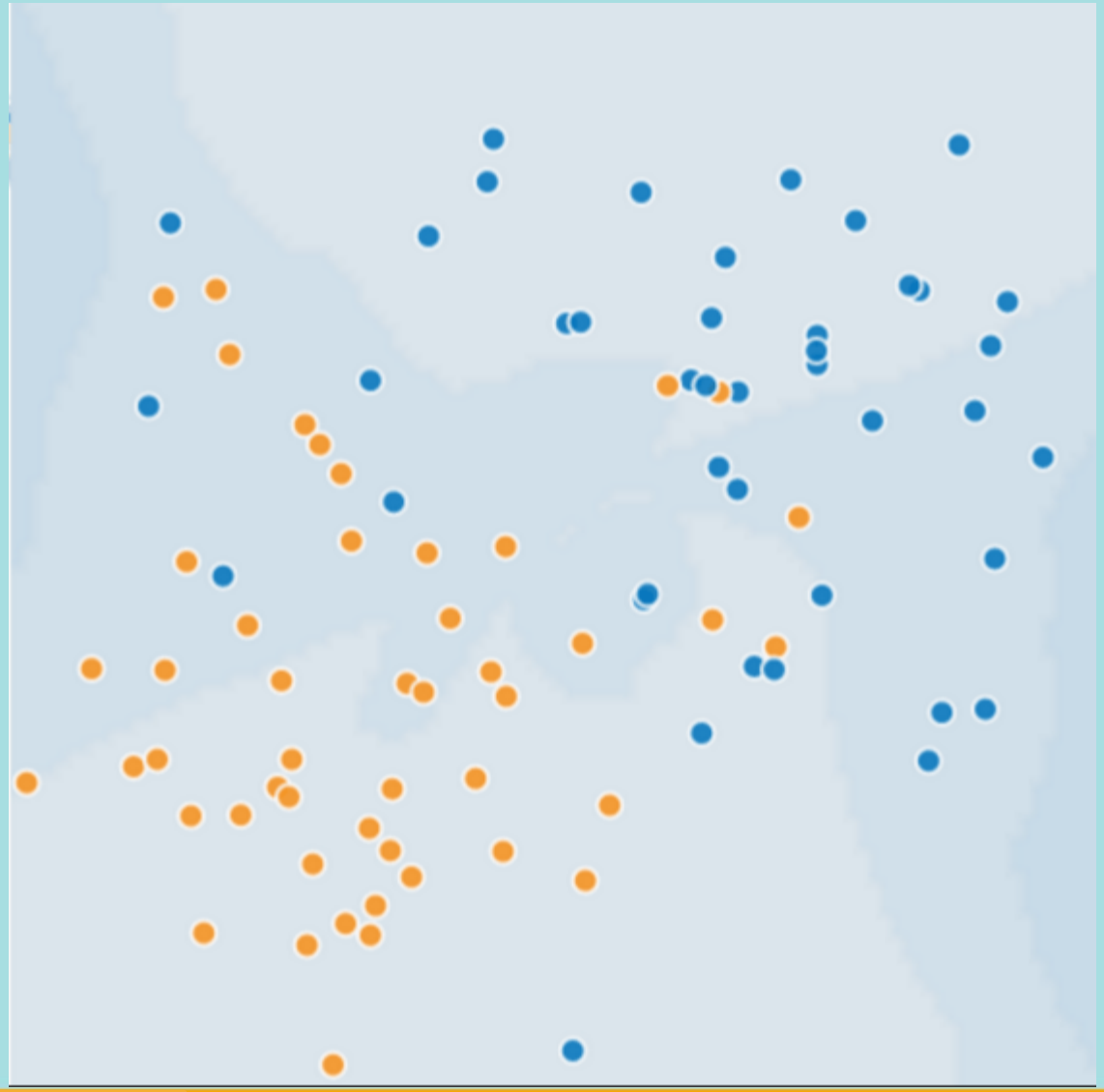
- Different learning rates are used at different learning time points, and of course different directions are considered.
- 在不同學習的時間點用不同的學習率，當然還有考慮不同的方向。

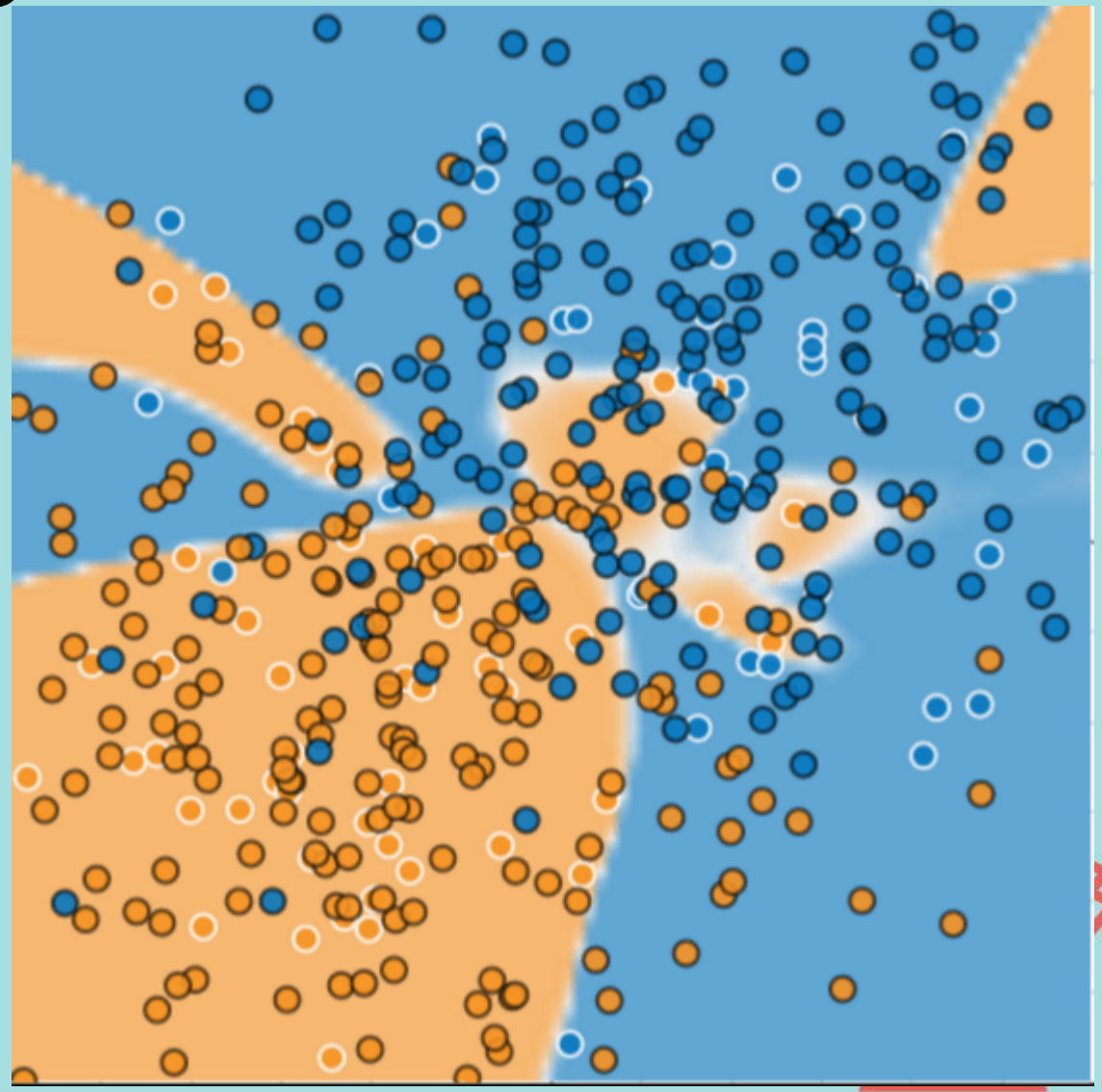
Overfitting and regularization

- Overfitting
- Regularization
- Dropout

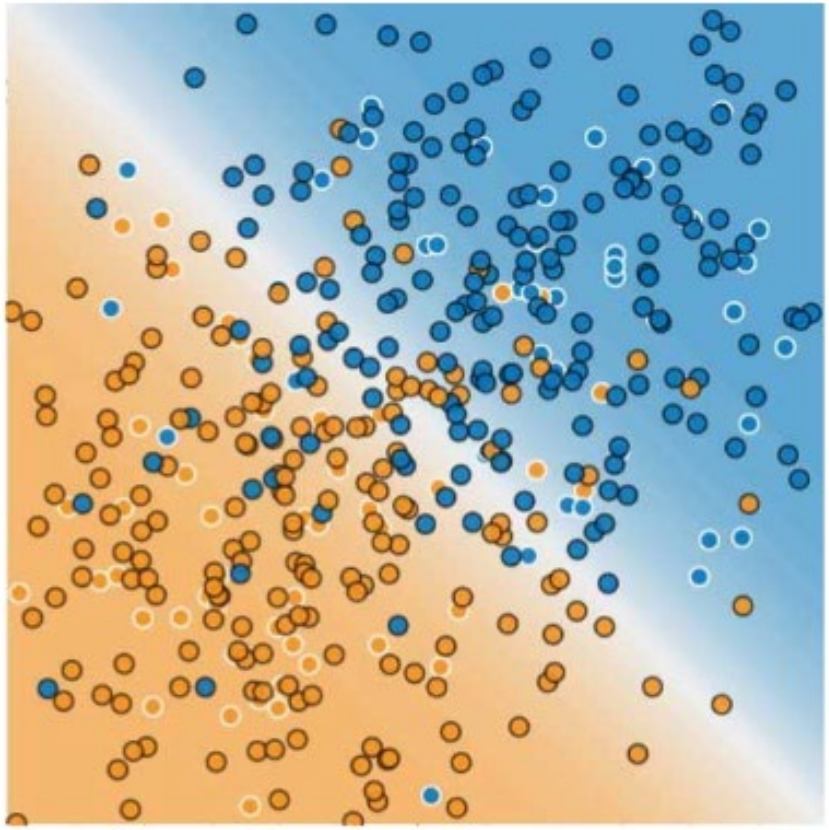


Overfitting

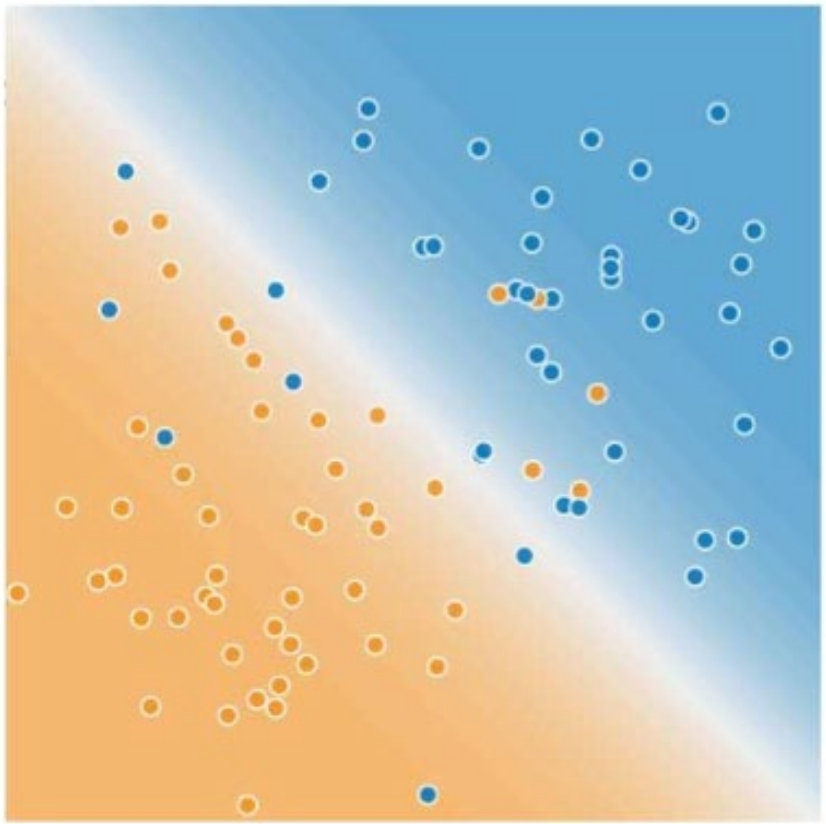




Training and Test Sets

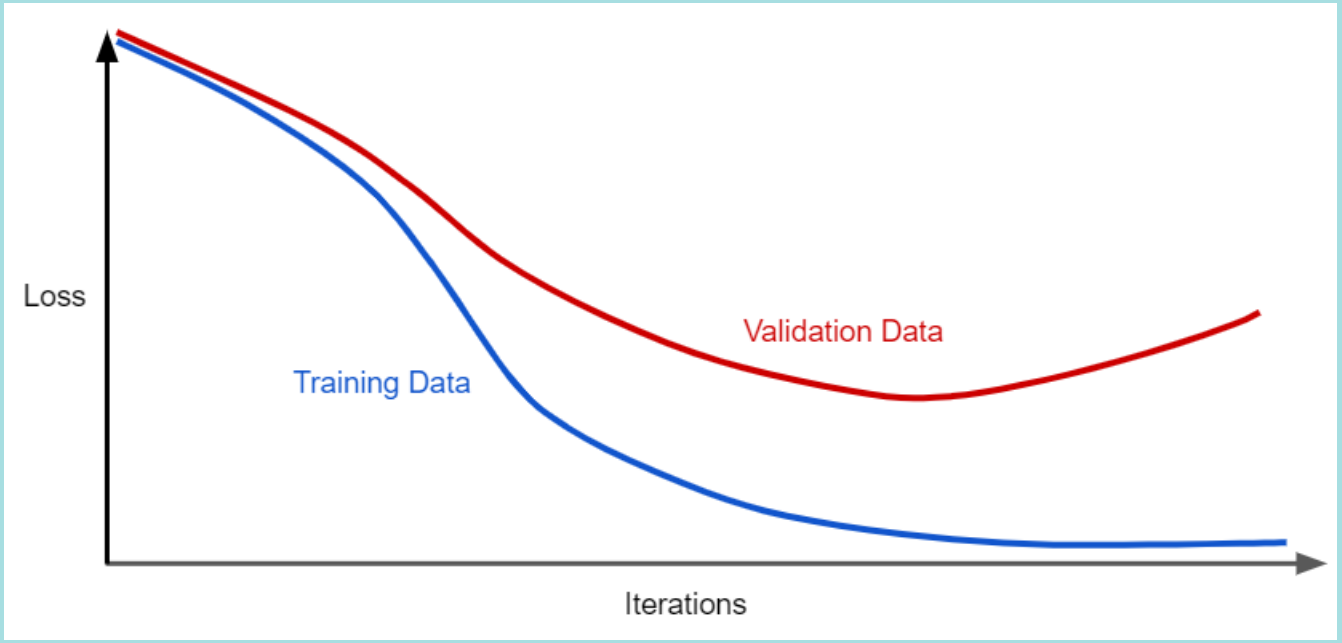


Training Data



Test Data

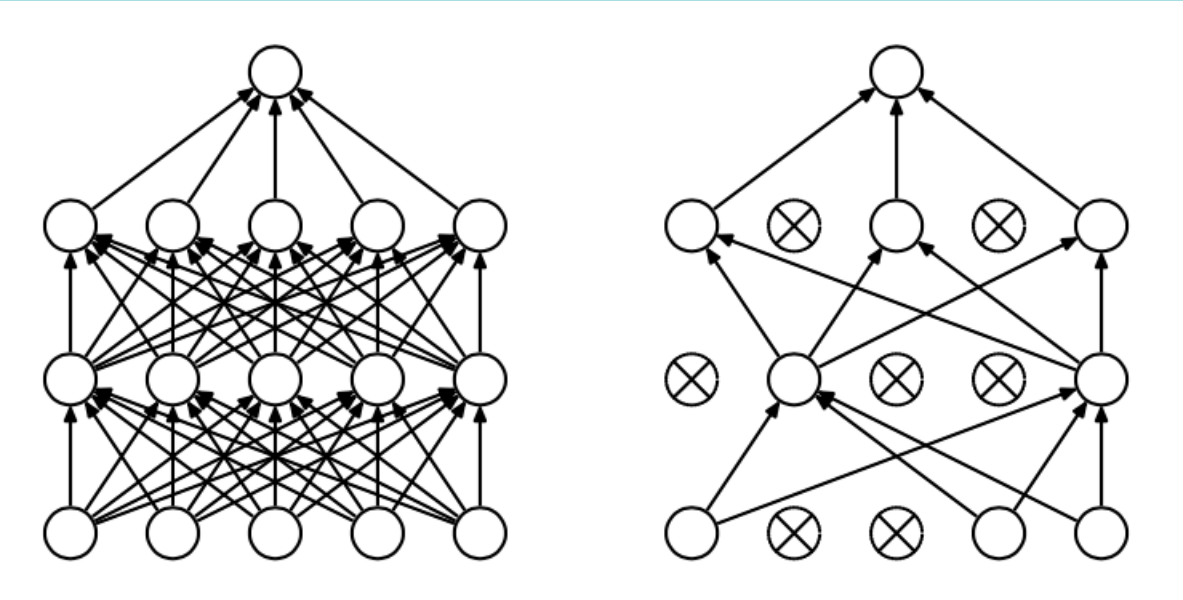
Regularization



$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \text{complexity}(\text{Model}))$$

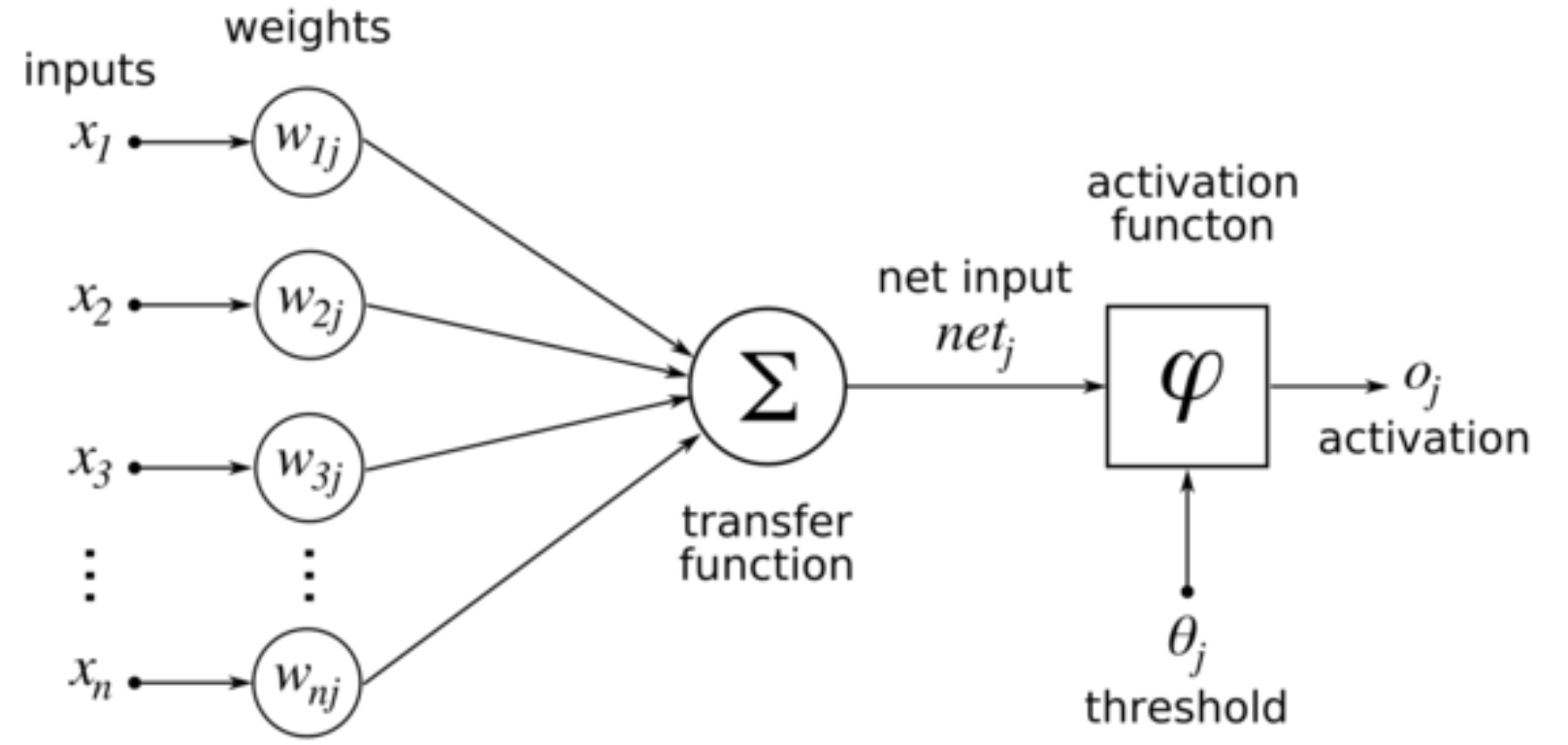
$$L_2 \text{ regularization term} = ||\boldsymbol{w}||_2^2 = w_1^2 + w_2^2 + \ldots + w_n^2$$

Dropout



- A form of regularization useful in training neural networks. Dropout regularization works by removing a random selection of a fixed number of the units in a network layer for a single gradient step.

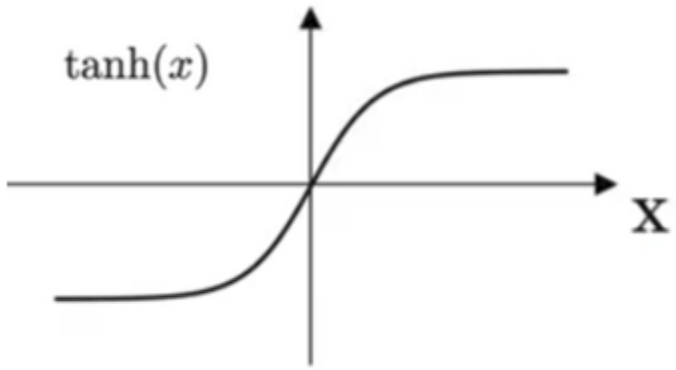
Activation function



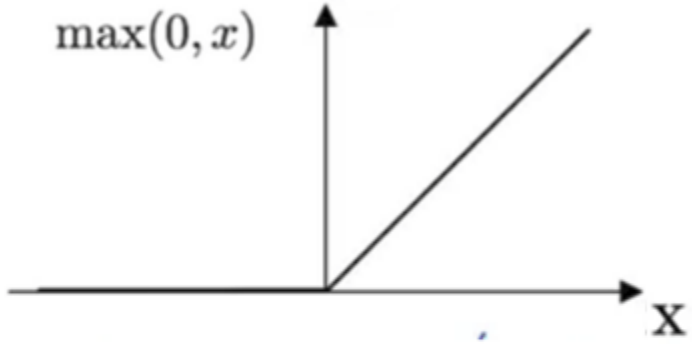
If the activation function is not used in the neural network, then in the neural network, the linear combination of the input of the above layer is the output of this layer. The output and input still cannot be separated from the linear relationship, and the deep neural network is lost. significance.

Different Activation Functions

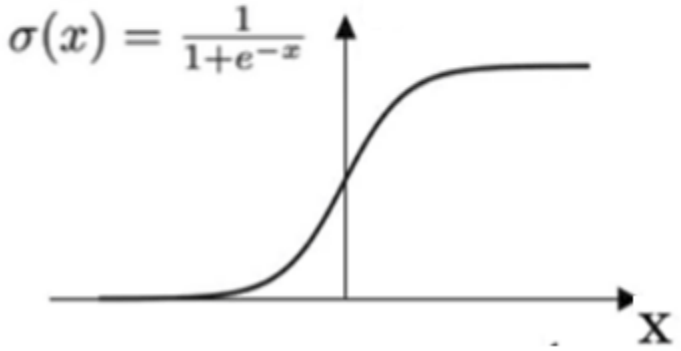
Hyper Tangent Function



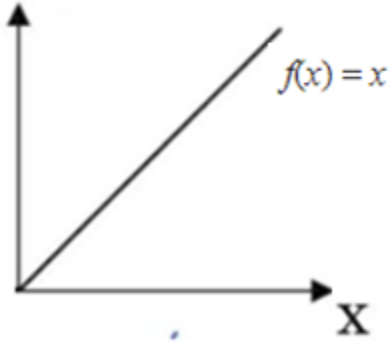
ReLU Function



Sigmoid Function



Identity Function



Thanks!

Q&A