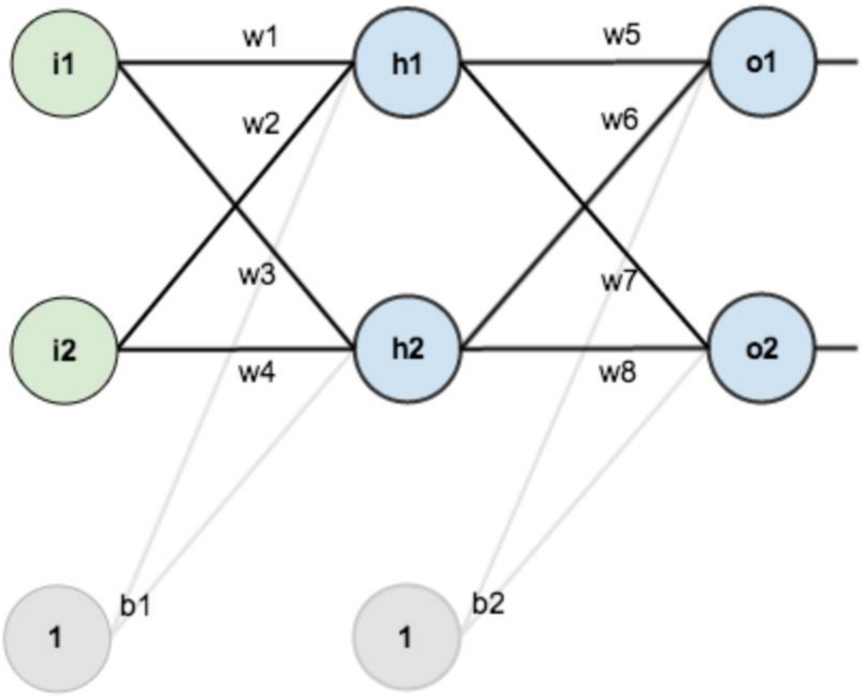
Basic layers in deep learning

韩坤洋

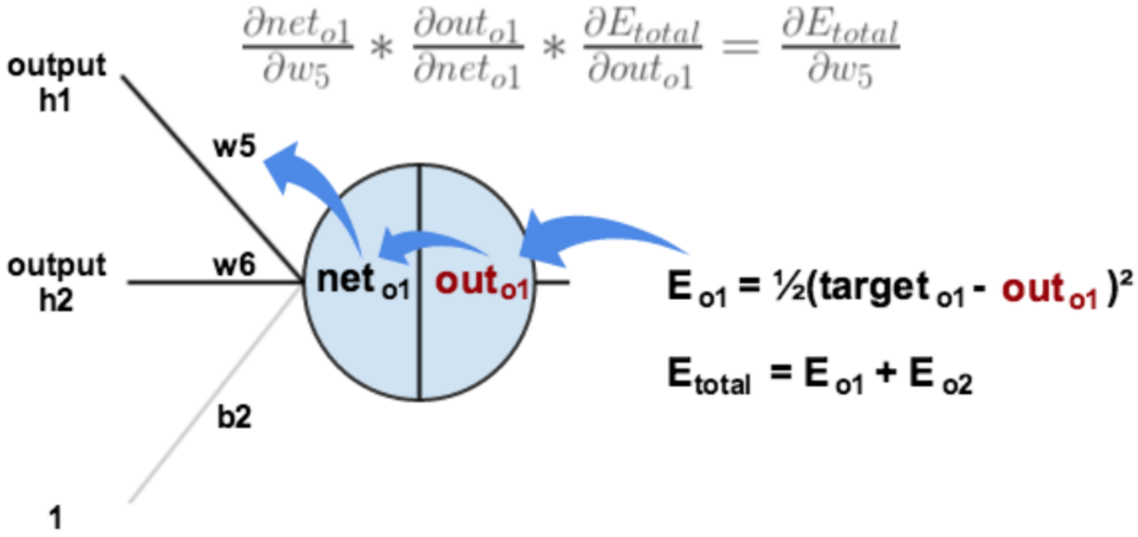
A simple MLP

- Set input
- Forward, get output
- Calculate loss
- Calculate gradient
- Update network



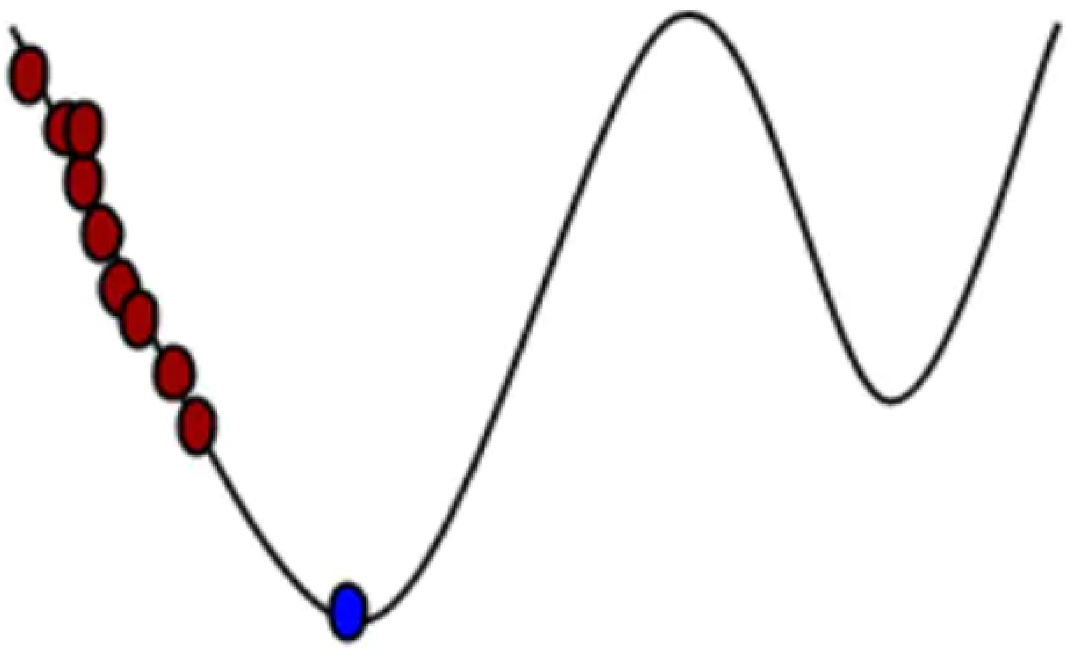
A simple MLP

- Set input
- Forward, get output
- Calculate loss
- Calculate gradient
- Update network



A simple MLP

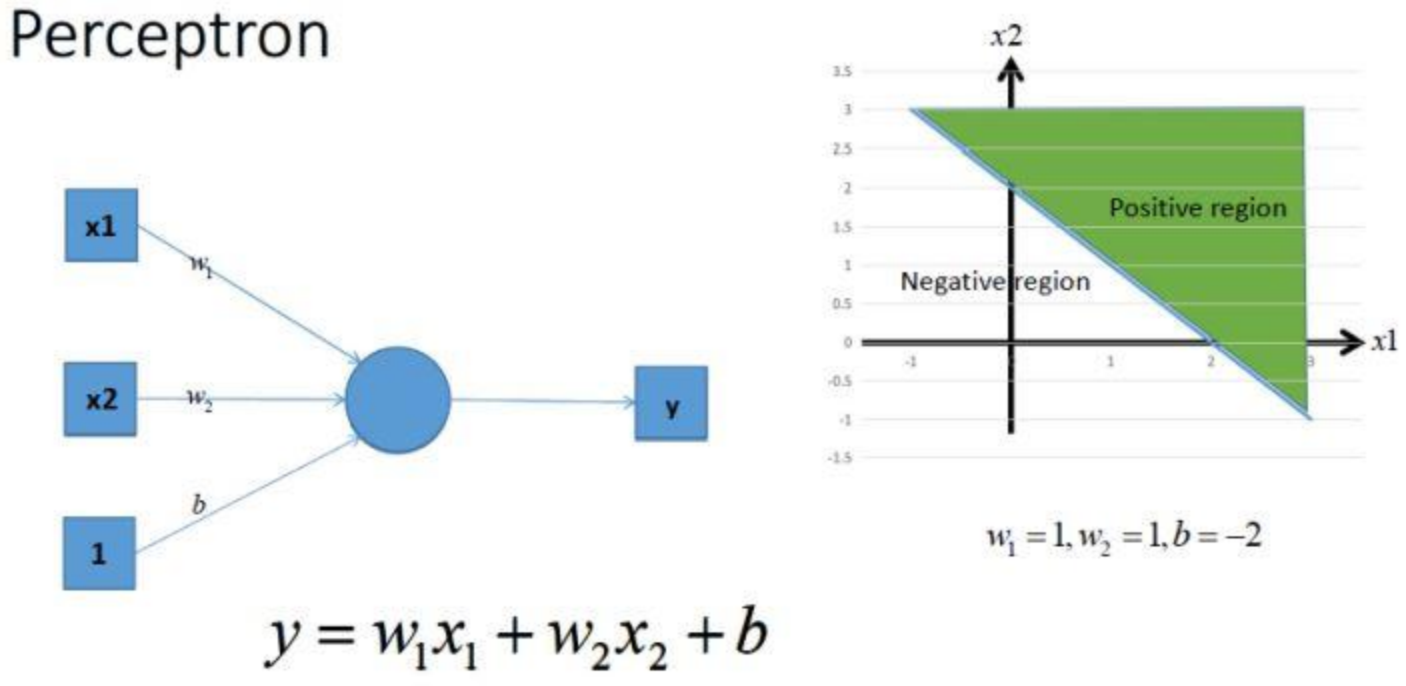
- Set input
- Forward, get output
- Calculate loss
- Calculate gradient
- Update network



$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta).$$

Activation

- Non-linear function
- Why need it?

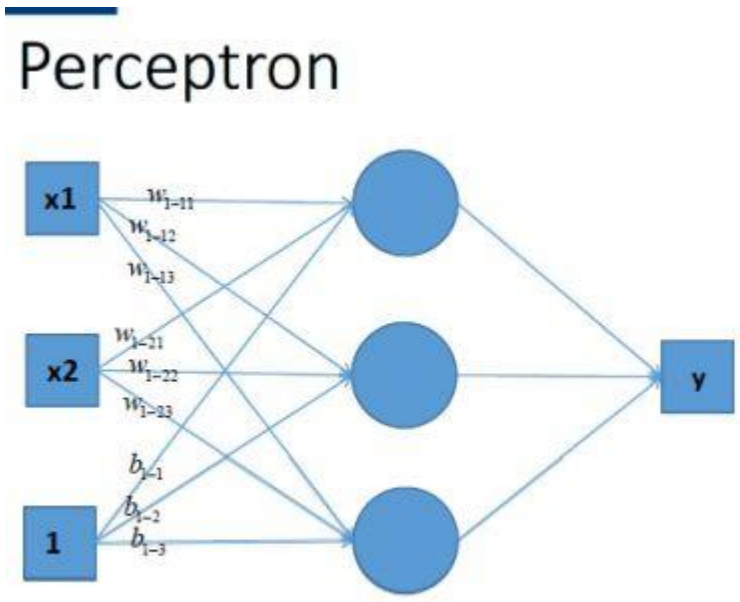


single layer perceptron is a linear classifier

Activation

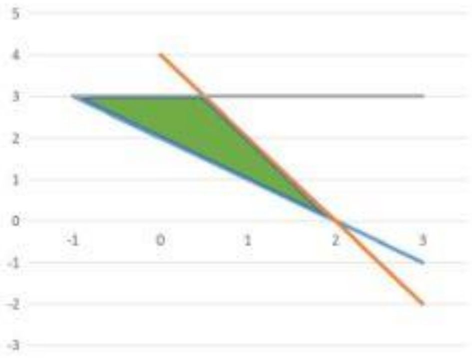
- Non-linear function

- Why need it?



linear combination of three decision lines

single layer perceptron is a linear classifier



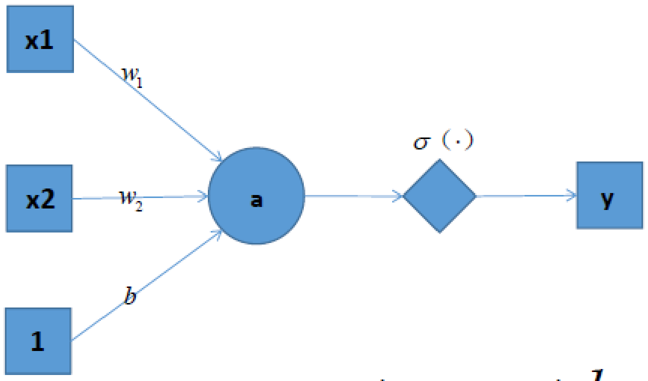
$$w_{1-11}=1, w_{1-12}=1, b_{1-1}=-2$$
$$w_{1-21}=2, w_{1-22}=1, b_{1-2}=4$$
$$w_{1-31}=0, w_{1-32}=1, b_{1-3}=3$$

Activation

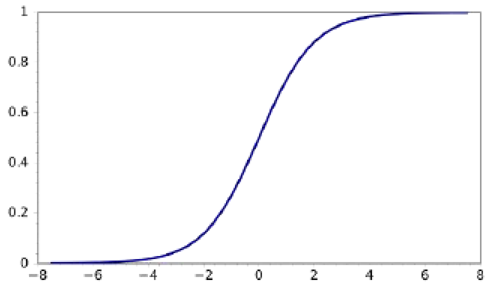
- Non-linear function

- Why need it?

Perceptron with non-linear activation function



$$a = w_1 x_1 + w_2 x_2 + b$$
$$y = \sigma(a)$$



$\sigma(\cdot)$ is a non-linear activation function, sigmoid was the most popular one,

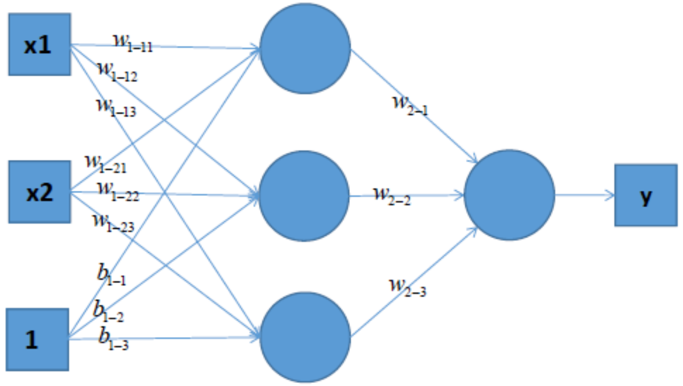
$$\sigma(y) = \frac{1}{1+e^{-y}}$$

Activation

- Non-linear function

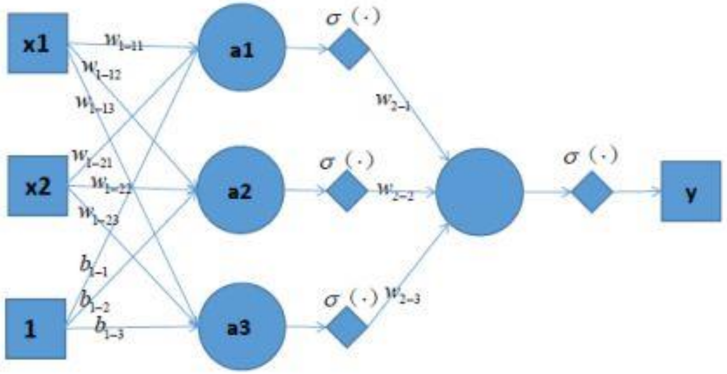
- Why need it?

Perceptron with one hidden layer



$$
\begin{aligned}
y = {} & w_{2\text{-}1}(w_{1\text{-}11}x_1 + w_{1-21}x_2 + b_{1\text{-}1}) \\
& + w_{2\text{-}2}(w_{1\text{-}12}x_1 + w_{1-22}x_2 + b_{1\text{-}2}) \\
& + w_{2\text{-}3}(w_{1\text{-}13}x_1 + w_{1-23}x_2 + b_{1\text{-}3})
\end{aligned}
$$

Perceptron with non-linear activation function



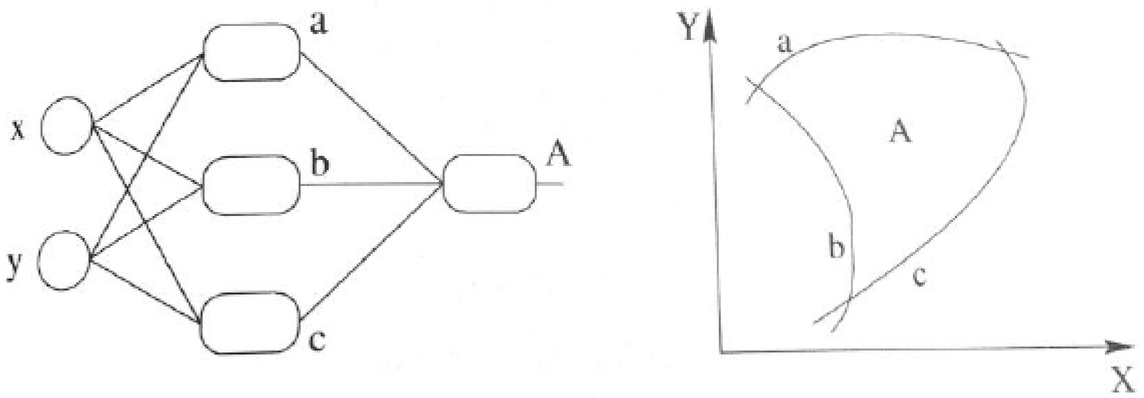
$$
\begin{aligned}
a1 &= w_{1\text{-}11}x_1 + w_{1-21}x_2 + b_{1\text{-}1} \\
a2 &= w_{1\text{-}12}x_1 + w_{1-22}x_2 + b_{1\text{-}2} \\
a3 &= w_{1\text{-}13}x_1 + w_{1-23}x_2 + b_{1\text{-}3}
\end{aligned}
$$

$$y = \sigma(w_{2-1}\sigma\ (a1) + w_{2-2}\sigma\ (a2) + w_{2-3}\sigma\ (a3))$$

Activation

- Non-linear function

- Why need it?



with sigmoid activation function

Activation

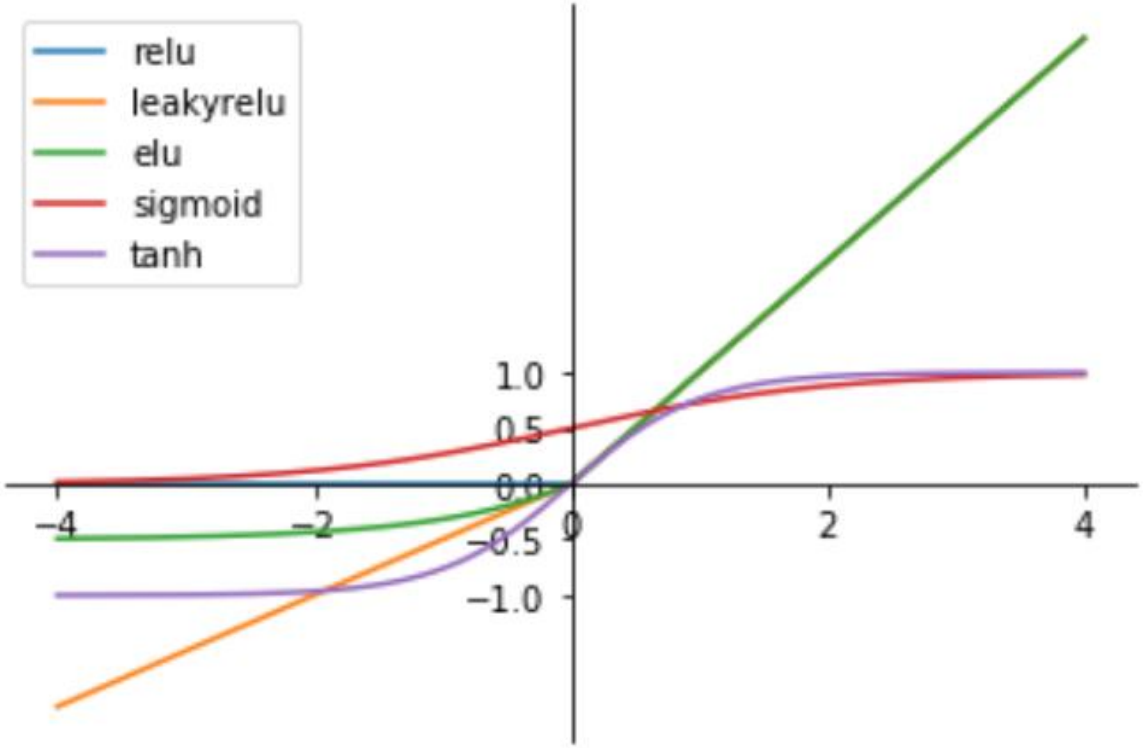
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

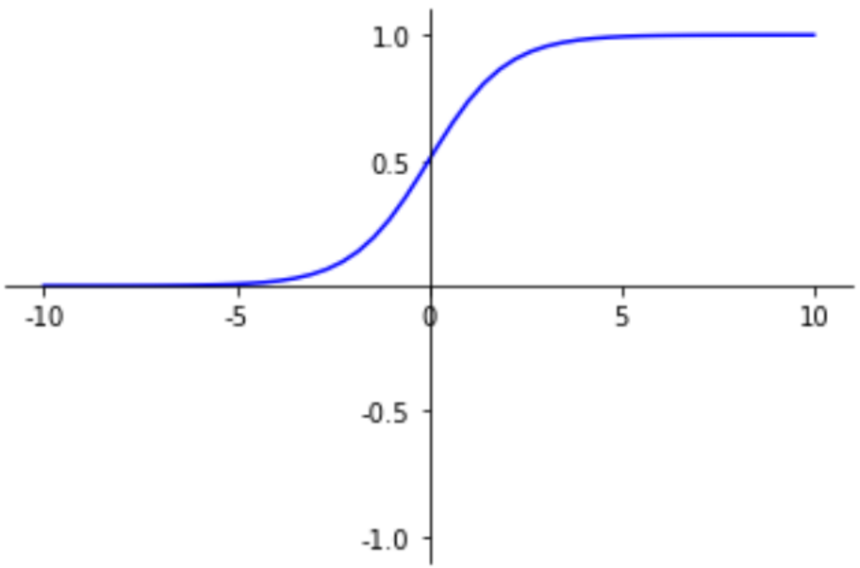
$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma x & \text{if } x \leq 0 \end{cases}$$

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

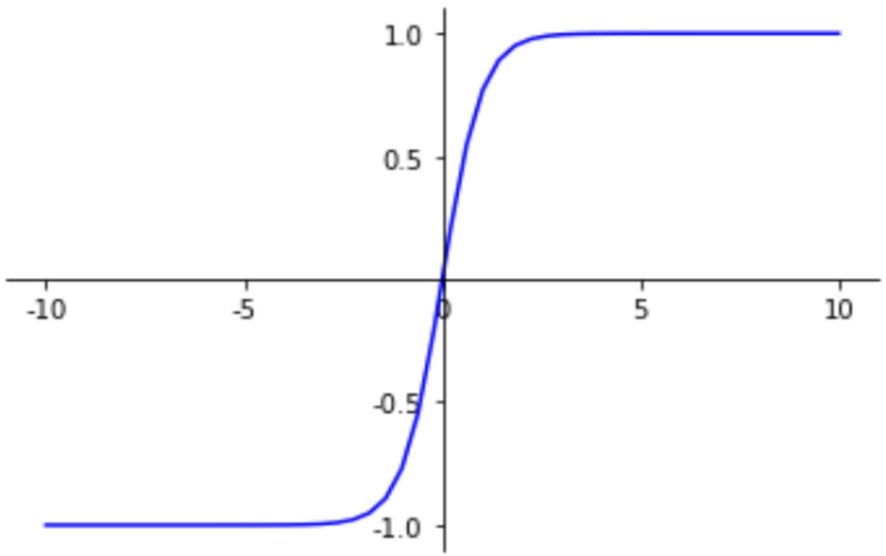


- sigmoid



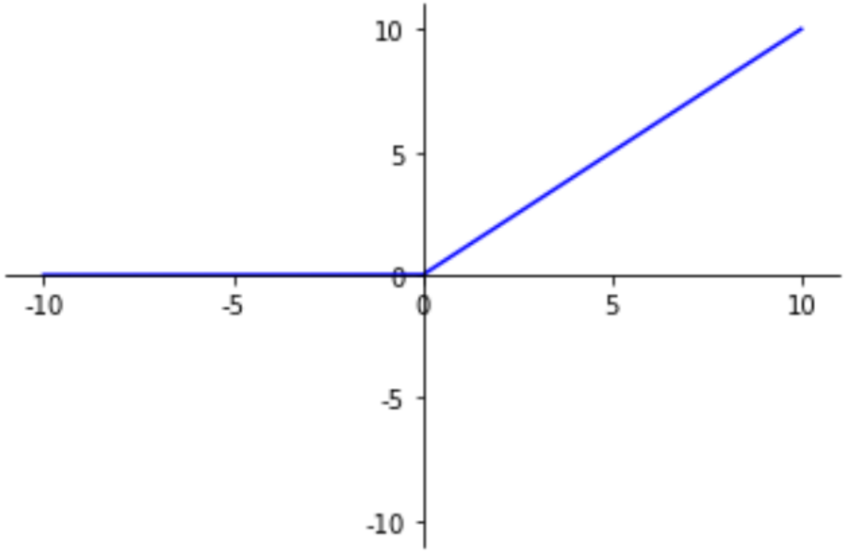
$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

- tanh

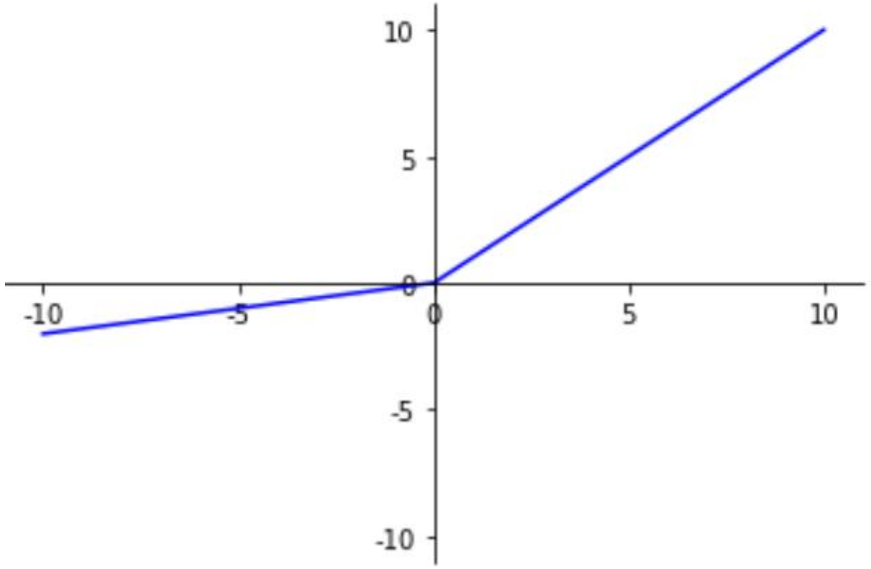


$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

- ReLU
- leakyReLU

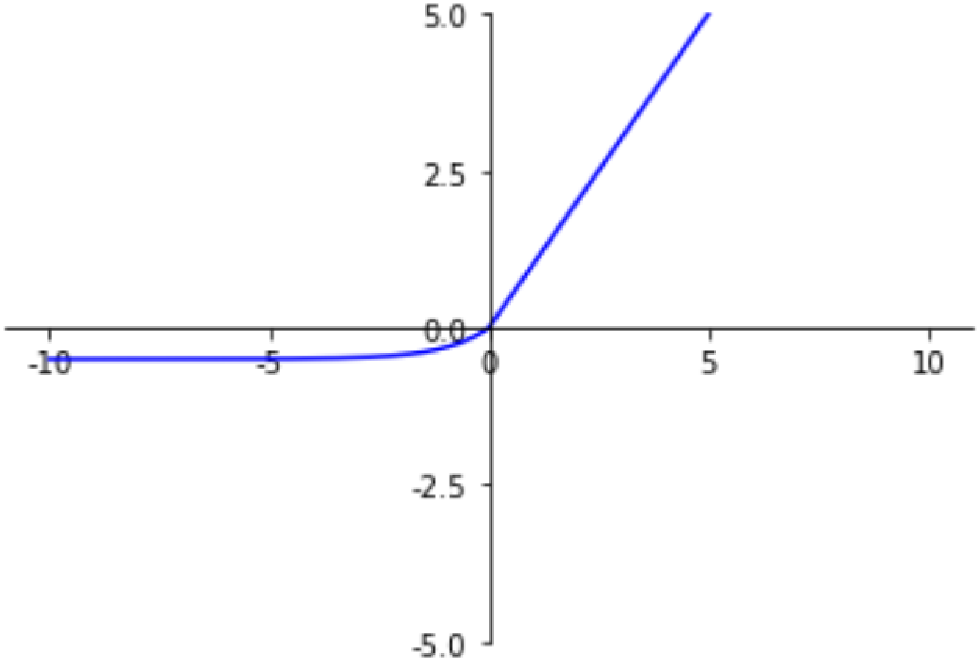


$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$



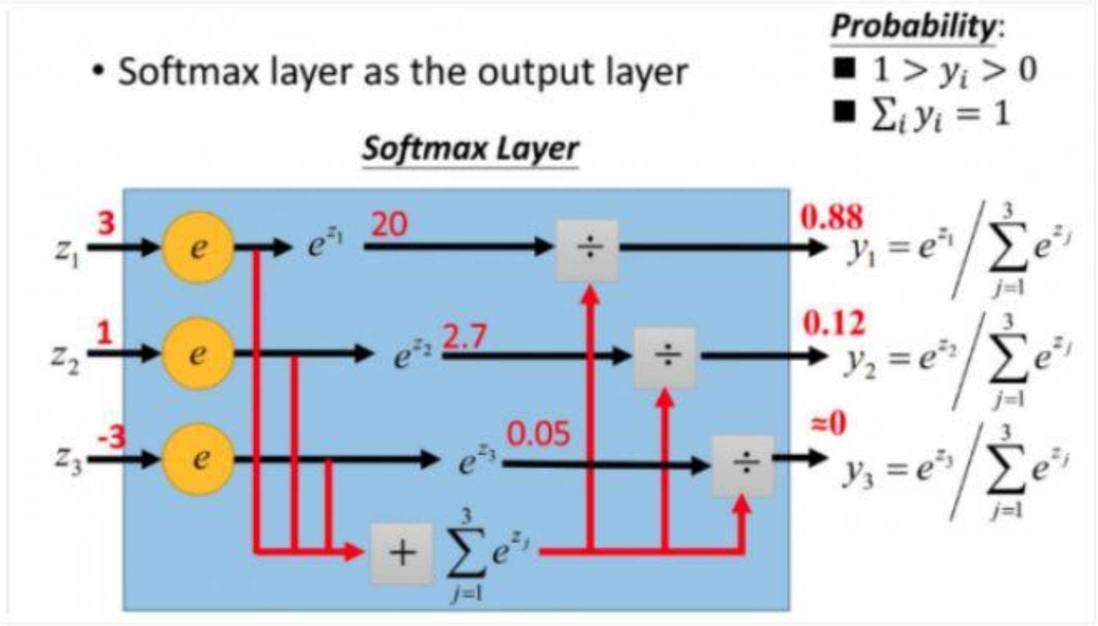
$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma x & \text{if } x \leq 0 \end{cases}$$

- eLU



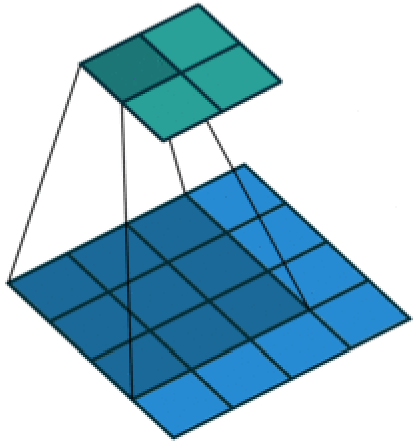
$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

- softmax



Convolution layer

- 1 dimensional conv, Time series data
- 2d conv, image
- 3d conv, video

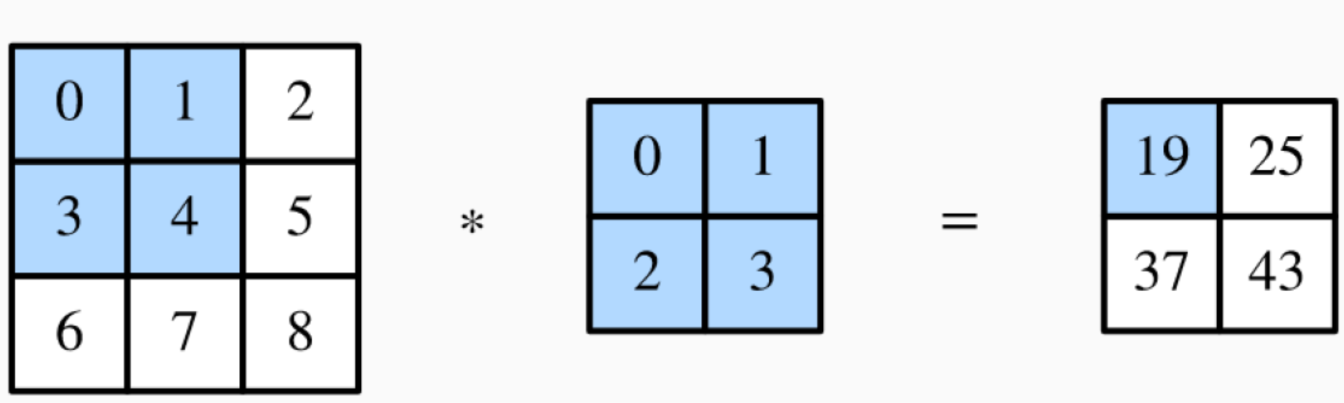


$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

$0 + 1 + 6 + 12 = 19$

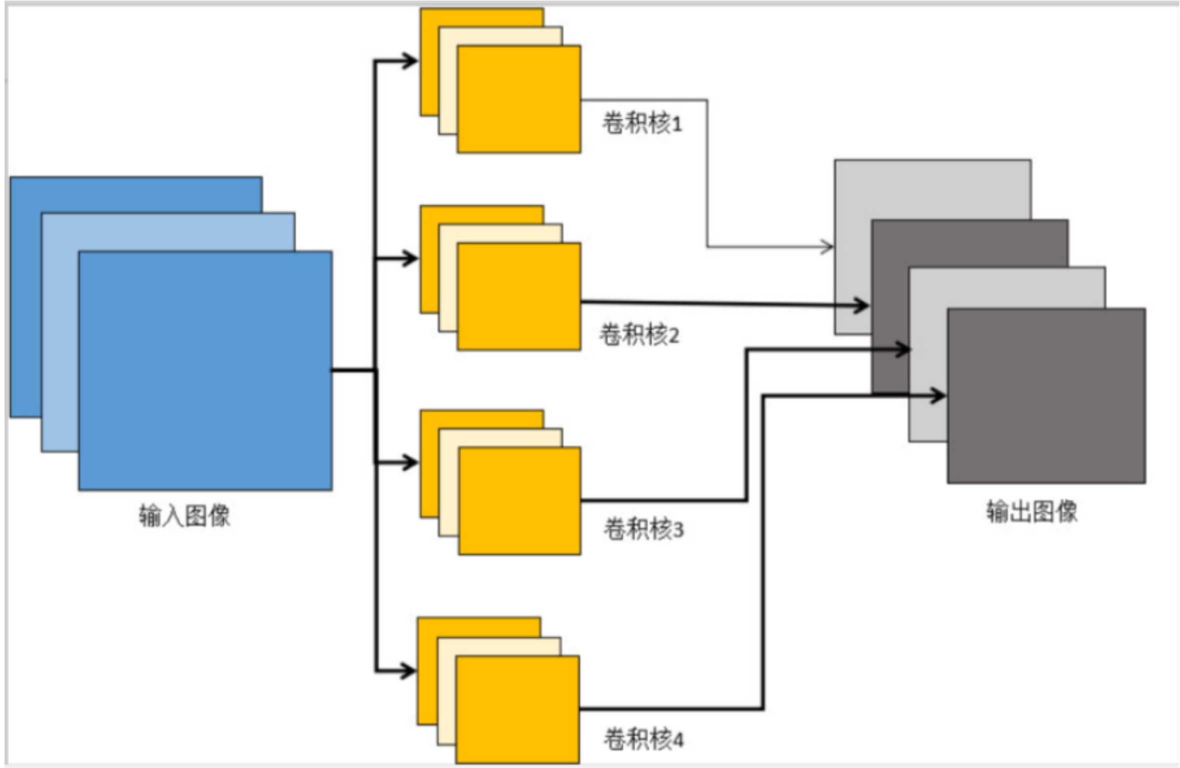
Convolution layer

- Kernel size



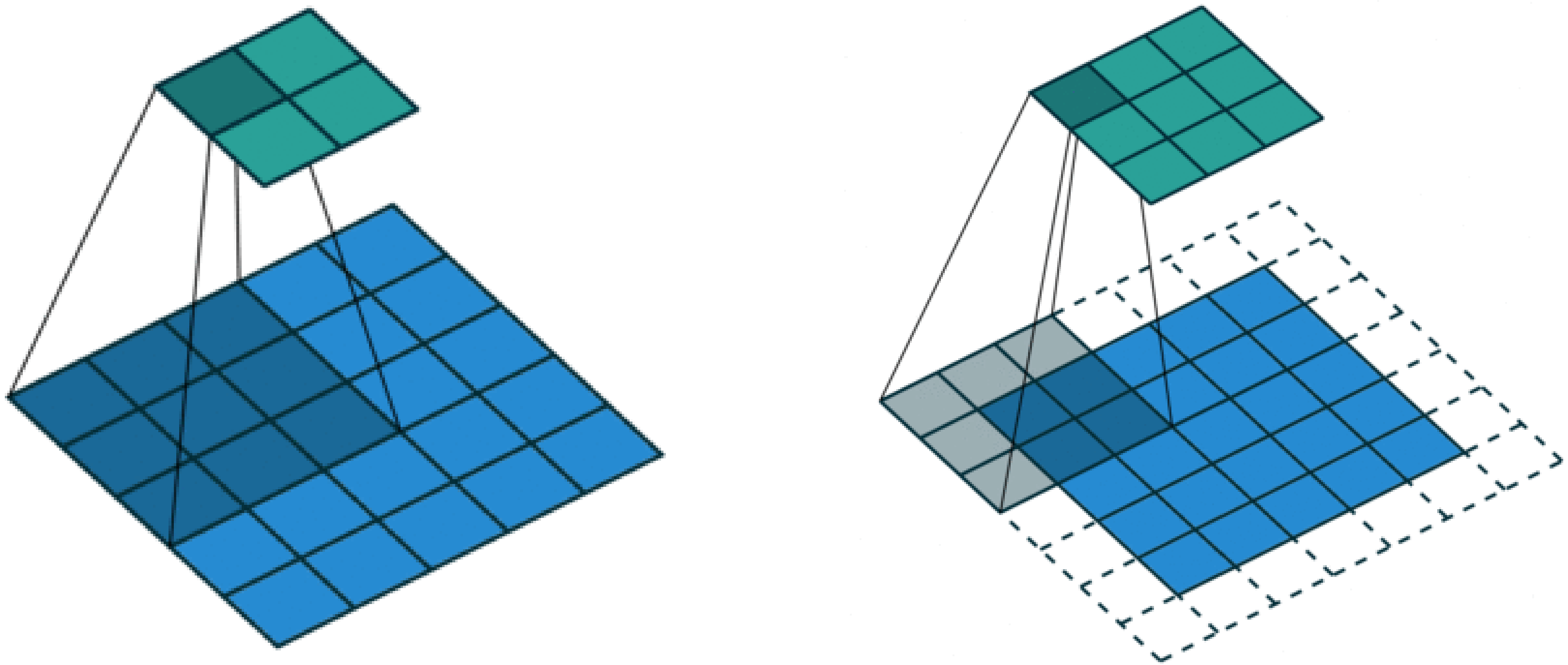
Convolution layer

- Input channel
- Output channel



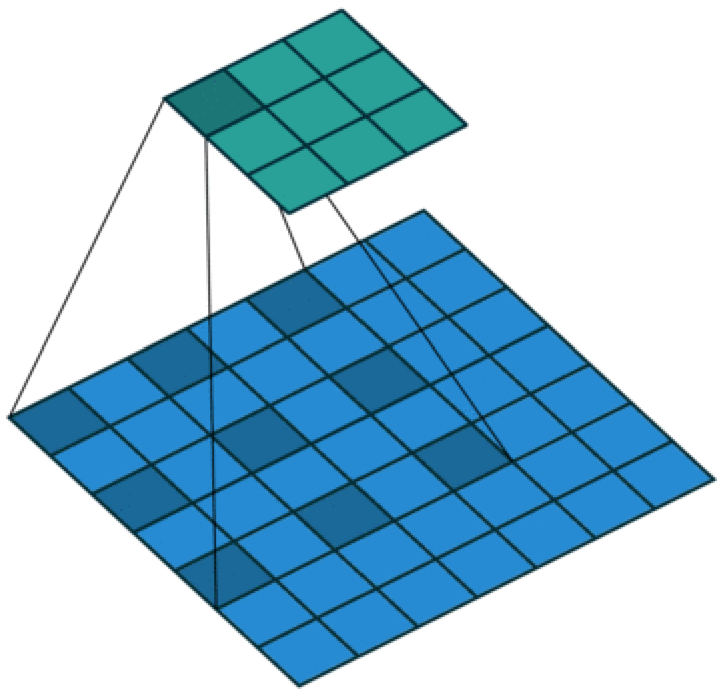
Convolution layer

- Stride
- Padding

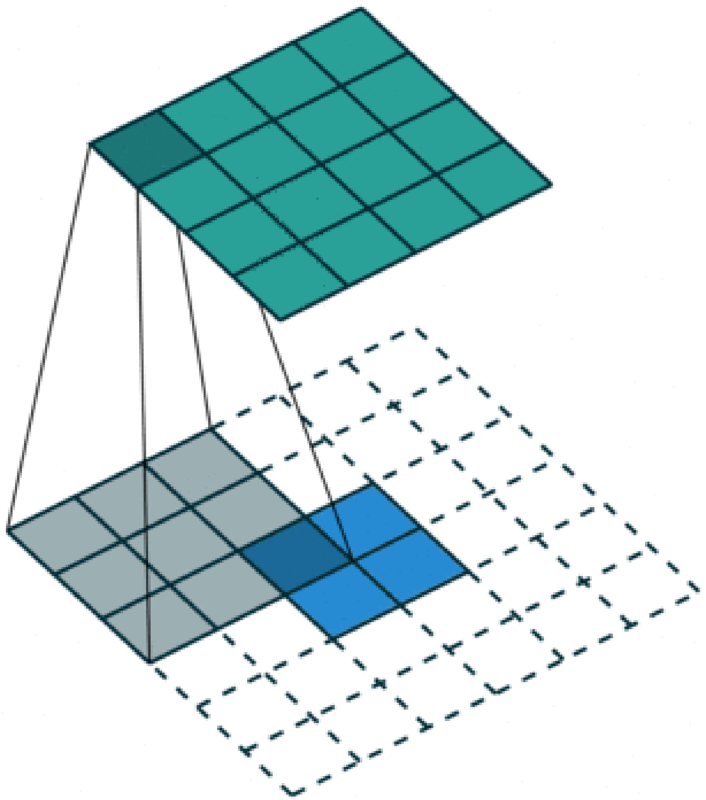


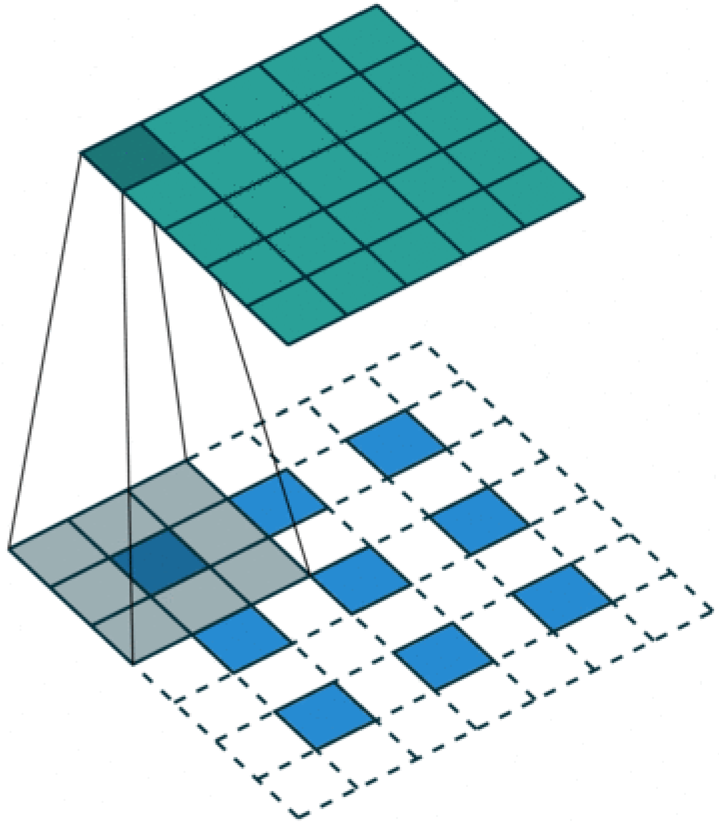
Convolution layer

- Dilation



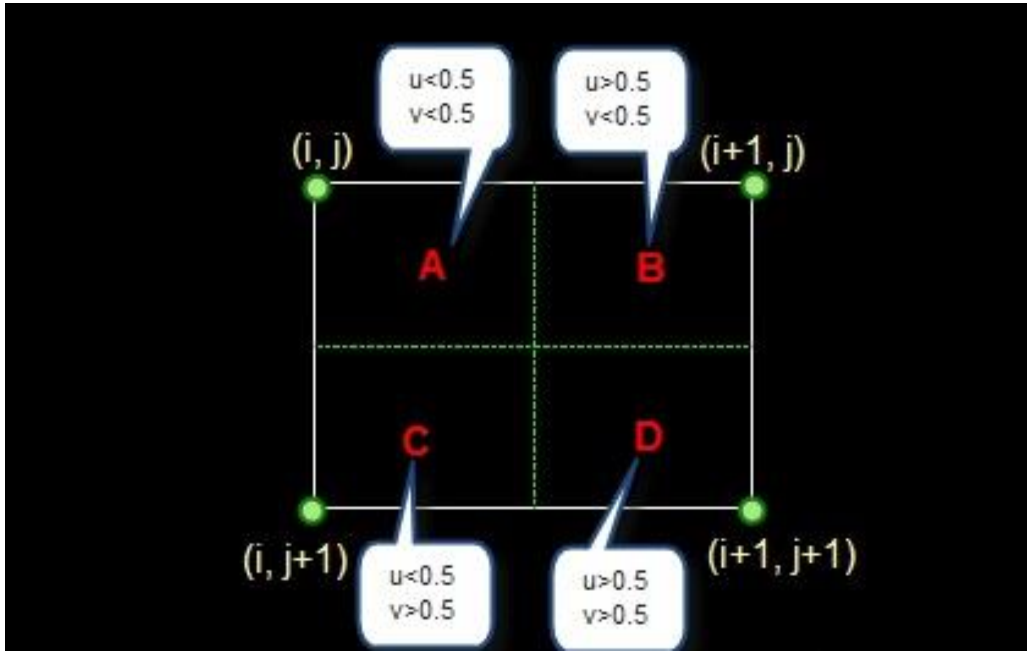
Transposed Convolution layer

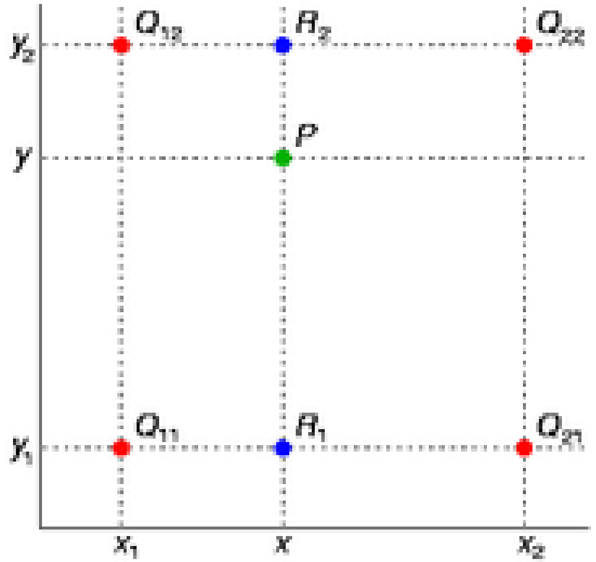




Upsample

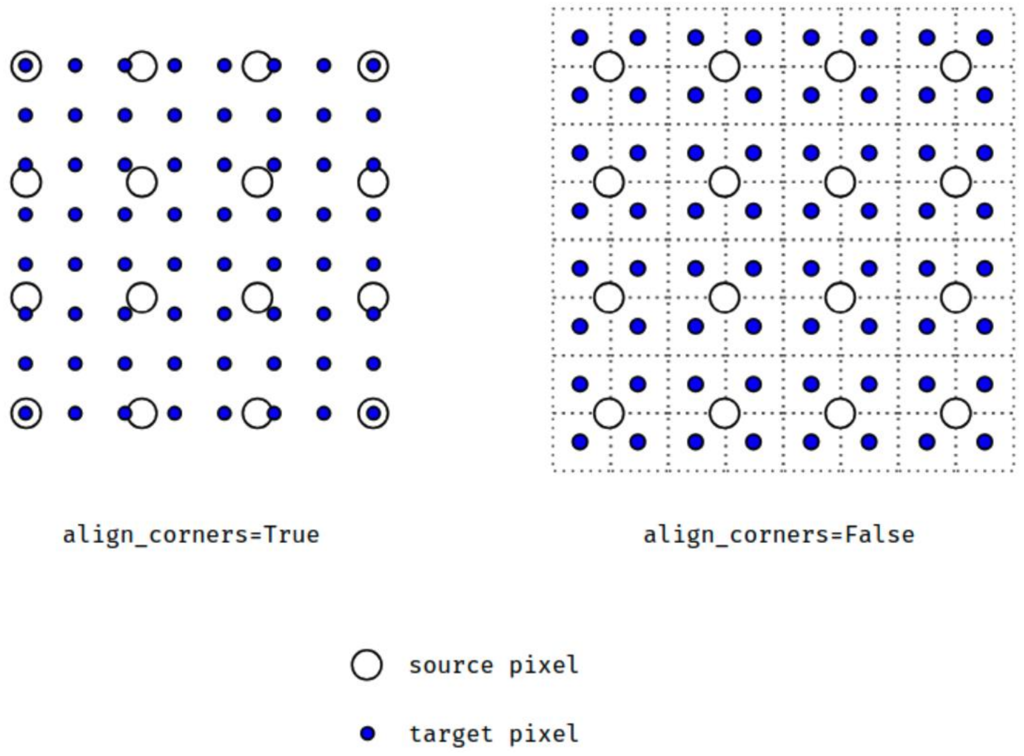
- Mode
  - Nearest
  - Bilinear
- Align corner





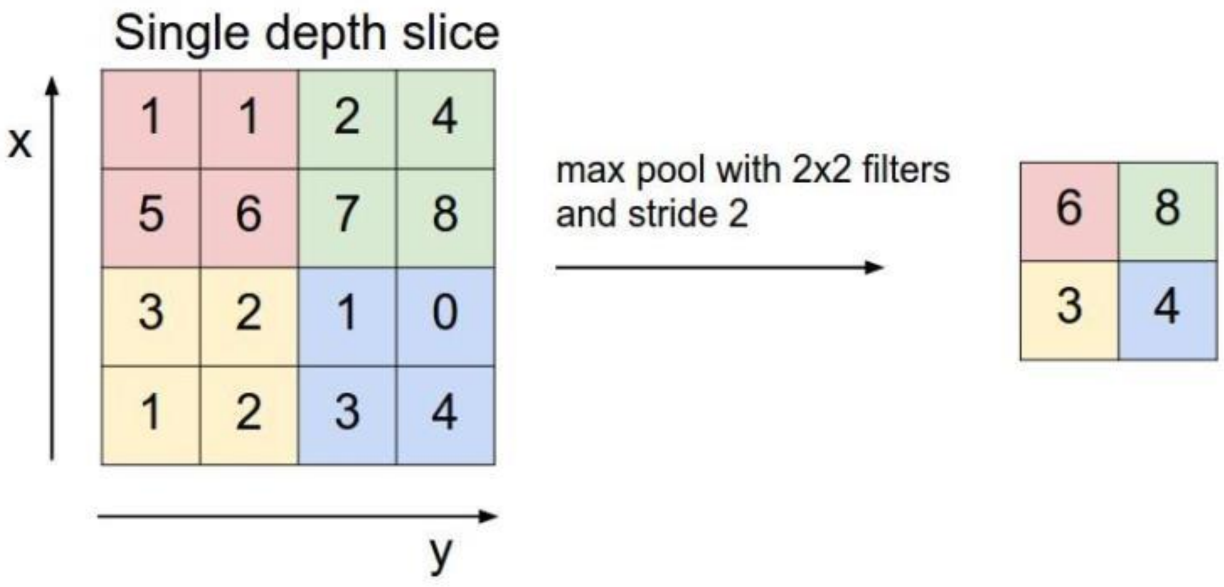
Upsample

- Mode
  - Nearest
  - Bilinear
- Align corner



Pooling

- Type: max pooling, avg pooling
- Kernel size
- Stride



Normalization

- Make sure each sample have same distribution

$$h = f\left(\mathbf{g} \cdot \frac{\mathbf{x} - \mu}{\sigma} + \mathbf{b}\right)$$

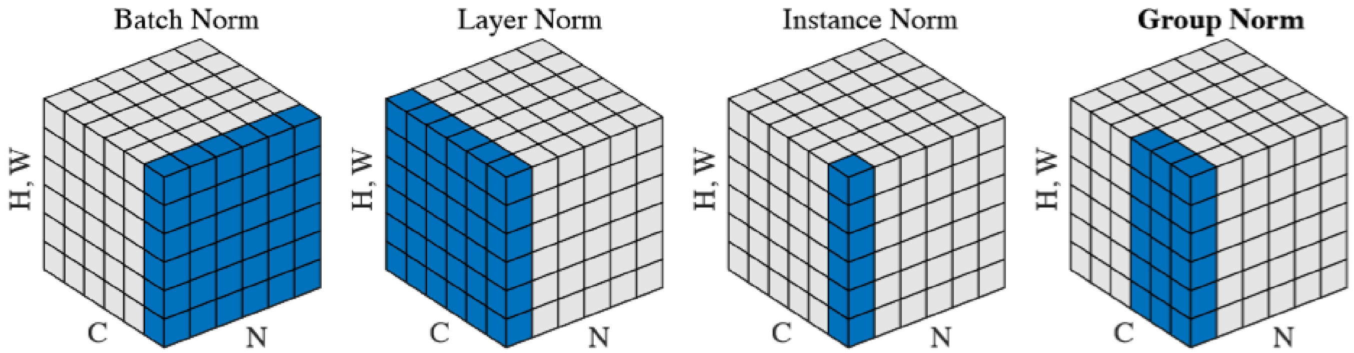


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

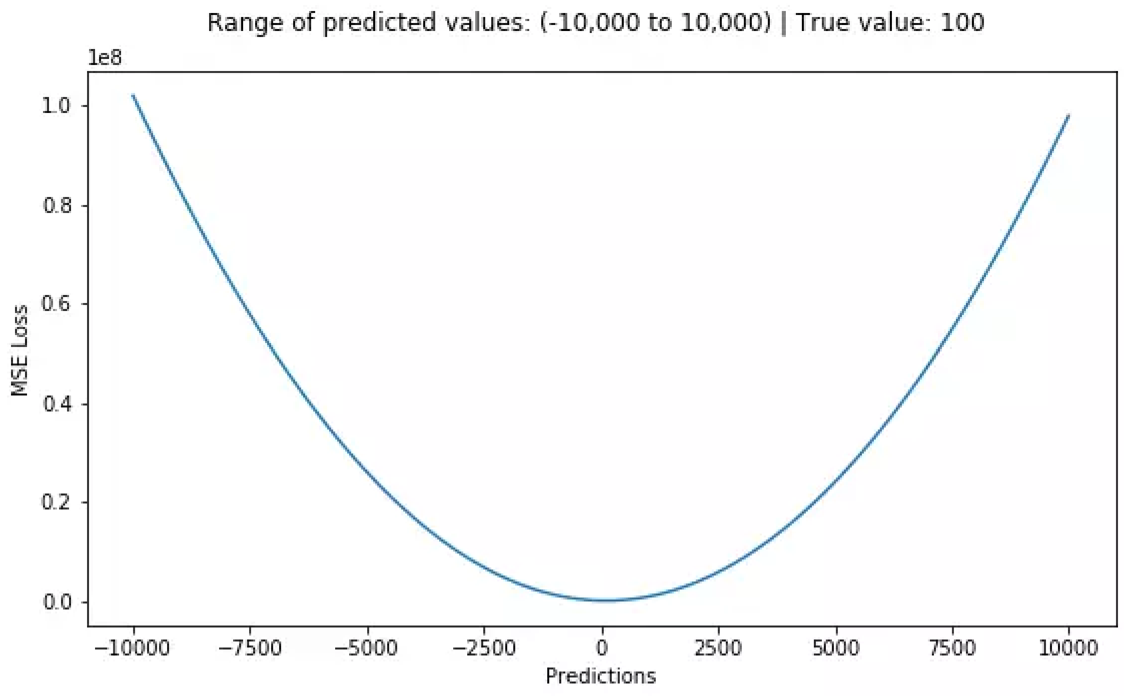
$$MSE = \sum_{i=1}^{n} (y_i - y_i^p)^2$$

$$MAE = \sum_{i=1}^{n} |y_i - y_i^p|$$

Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

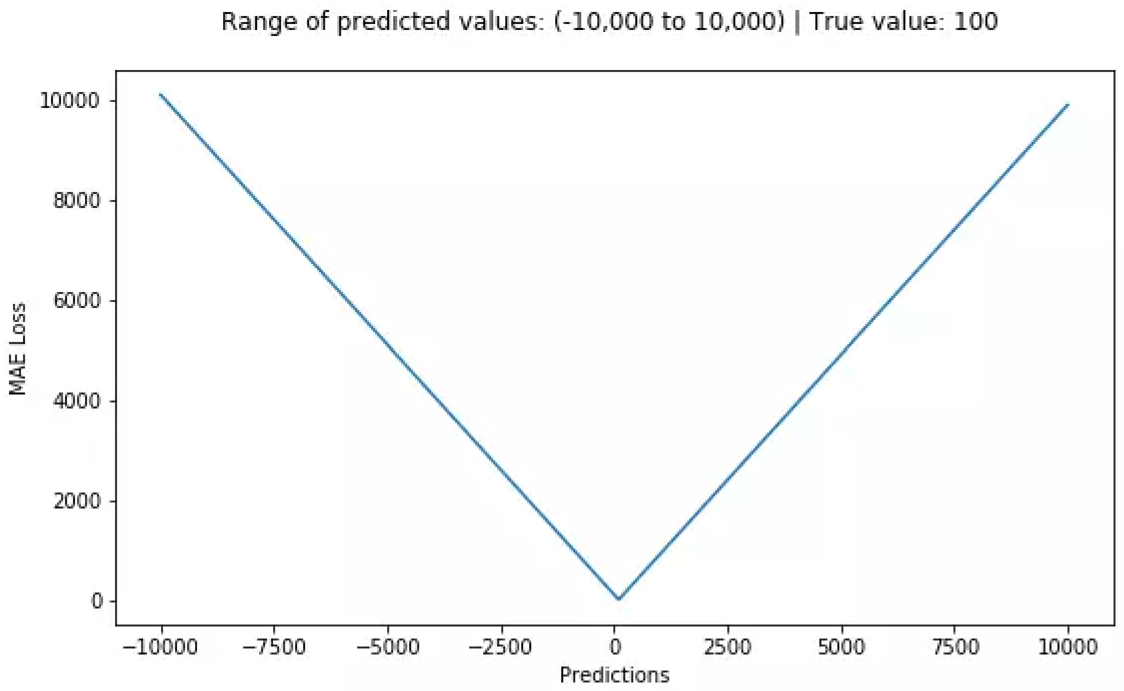
$$MSE = \sum_{i=1}^{n} (y_i - y_i^p)^2$$



Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

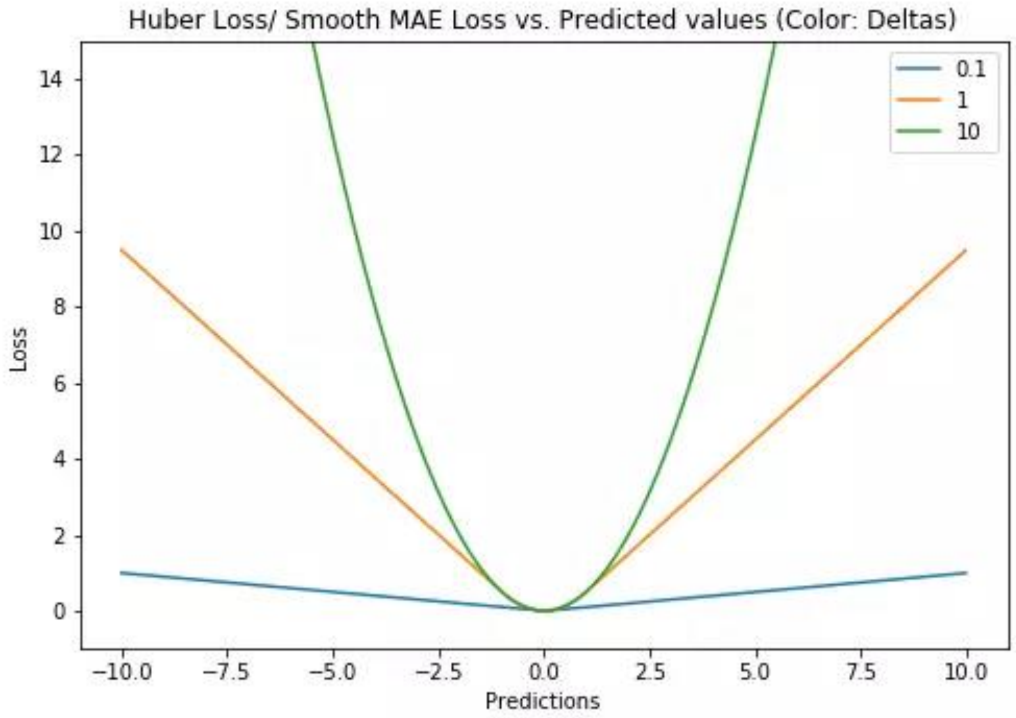
$$MAE = \sum_{i=1}^{n} |y_i - y_i^p|$$



Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

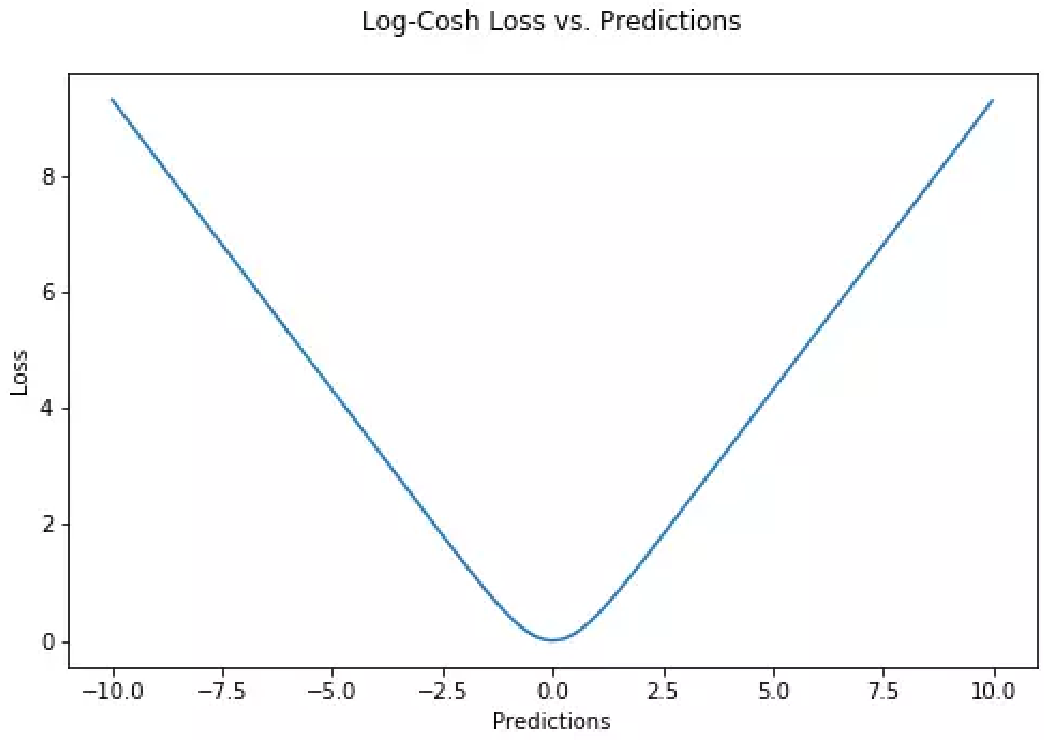
$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$



Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

$$L(y, y^p) = \sum_{i=1}^{n} \log(\cosh(y_i^p - y_i))$$



Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

KL Divergence

A measure of how one probability distribution is different from a second.

事实上交叉熵和KL散度的公式非常相近，其实就是KL散度的后半部分(公式2.1)：A和B的交叉熵 = A与B的KL散度 - A的熵。 $D_{KL}(A||B) = -S(A) + H(A,B)$

对比一下这是KL散度的公式:

$$D_{KL}(A||B) = \sum_i P_A(x_i) log\left(\frac{P_A(x_i)}{P_B(x_i)}\right) = \sum_i P_A(x_i) log(P_A(x_i)) - P_A(x_i) log(P_B(x_i))$$

这是熵的公式:

$$S(A) = -\sum_i P_A(x_i) log P_A(x_i)$$

这是交叉熵公式:

$$H(A,B) = -\sum_i P_A(x_i) log(P_B(x_i))$$

Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

Primarily in SVM
It is intended for use with binary classification where the target values are in the set {-1, 1}.

$$L(y, f(x)) = max(0, 1 - yf(x))$$

y	-1 or 1
f(x)	[-1, 1]

Loss function

- Regression
  - Mean Square Error
  - Mean Absolute Error
  - Huber loss
  - Log-Cosh
- Classification
  - Cross entropy
  - Hinge loss
  - Exponential loss

设 $Y \in \{-1, 1\}$,模型为 $f(x)$,指数损失为

$$l(Y, f(x)) = e^{-Yf(x)} \tag{4}$$

忽略模型的具体形式，在指数损失下，我们的优化目标为

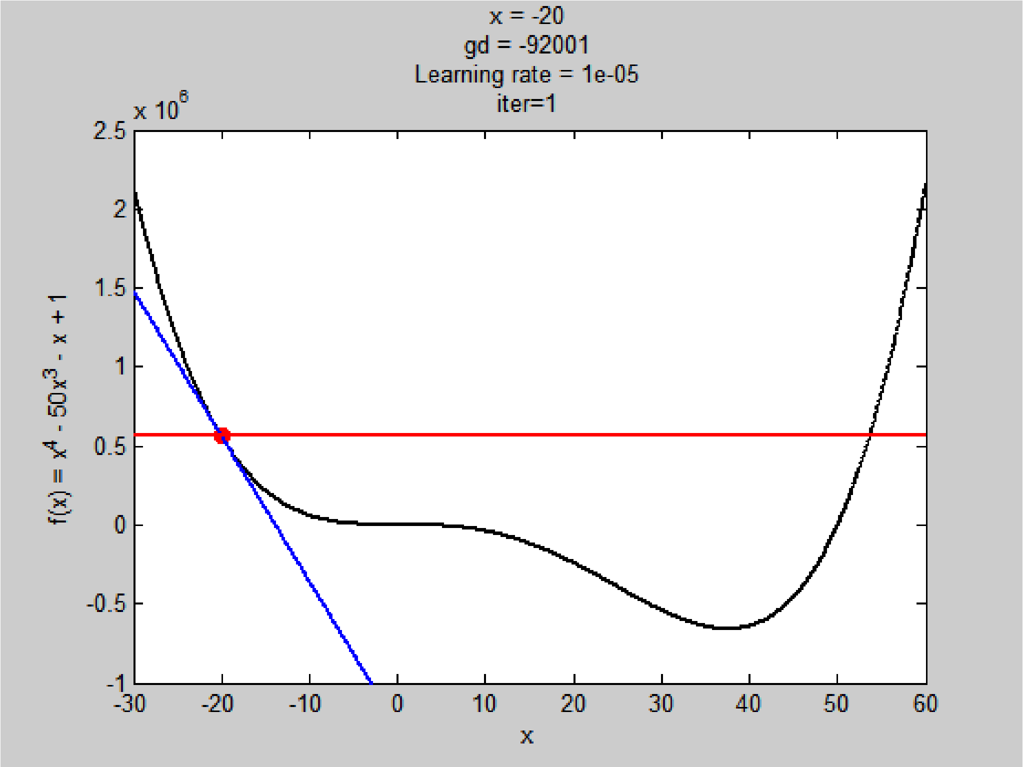
$$\min_{f(x)} E_x E_{Y|x}(e^{-Yf(x)})$$

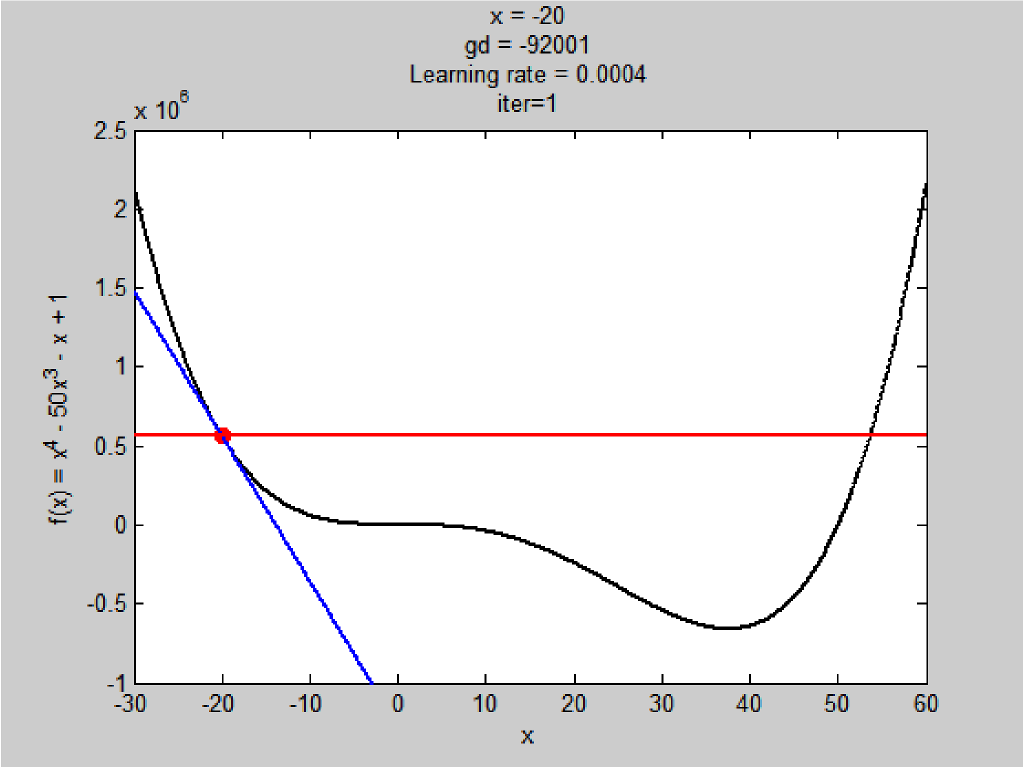
最优解为

$$f^*(x) = \arg\min_{f(x)} E_{Y|x}(e^{-Yf(x)}) = \frac{1}{2} log \frac{Pr(Y=1|x)}{Pr(Y=-1|x)} \tag{5}$$

Optimizer

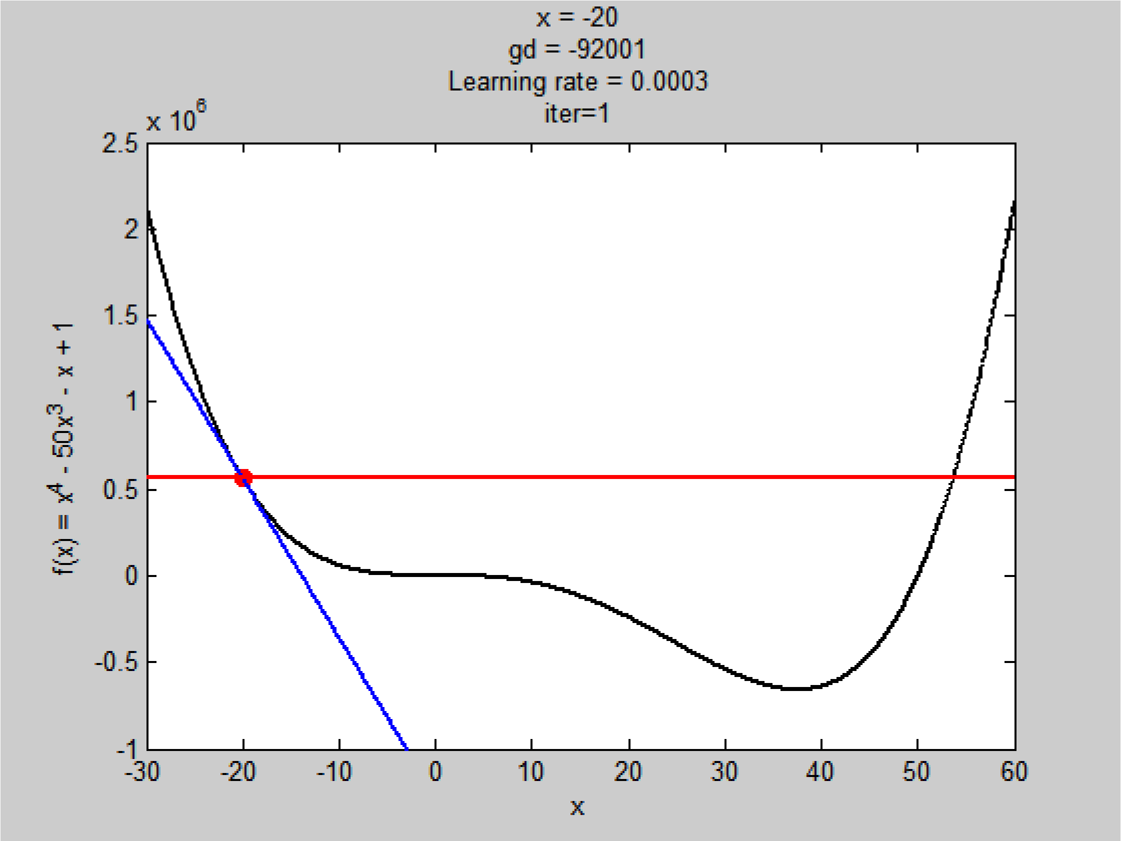
- Learning rate $\theta = \theta - \eta \cdot \nabla_\theta J(\theta).$





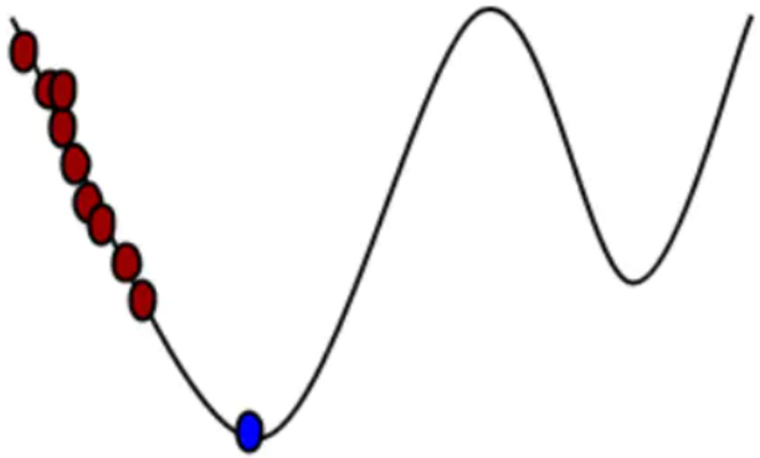
Optimizer

- Learning rate $\theta = \theta - \eta \cdot \nabla_\theta J(\theta).$



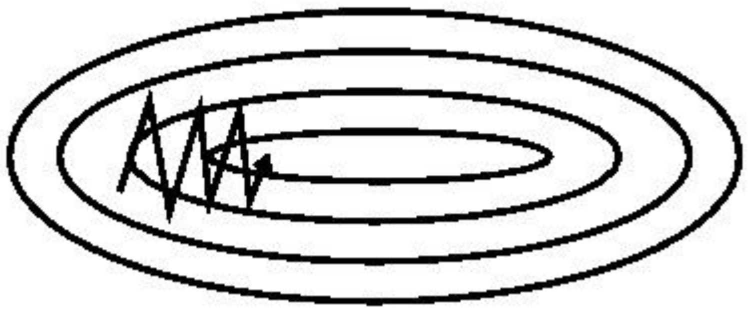
Optimizer

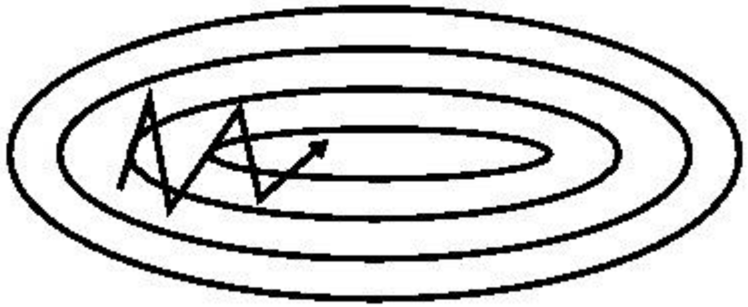
- Gradient Descent (GD)
  - Batch Gradient Descent （BGD）
  - Stochastic Gradient Descent (SGD)
  - Mini-Batch Gradient Descent （MBGD）
- Momentum
- Adagrad
  - Adadelta
  - RMSprop
- Adam



Optimizer

- Gradient Descent (GD)
  - BGD
  - SGD
  - MBGD
- Momentum
- Adagrad
  - Adadelta
  - RMSprop
- Adam

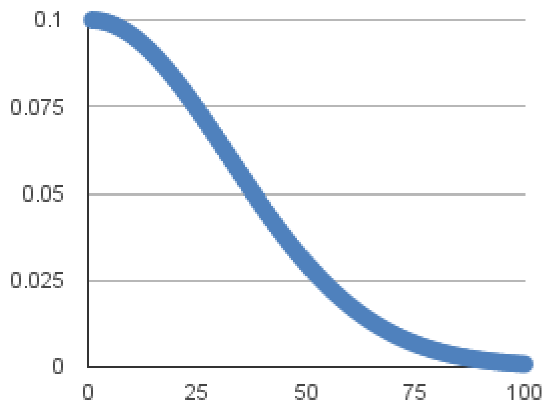




Optimizer

- Gradient Descent (GD)
  - BGD
  - SGD
  - MBGD
- Momentum
- Adagrad
  - Adadelta
  - RMSprop
- Adam

LearningRate = LearningRate * 1/(1 + decay * epoch)



Adagrad

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}.$$

Optimizer

- Gradient Descent (GD)
  - BGD
  - SGD
  - MBGD
- Momentum
- Adagrad
  - Adadelta
  - RMSprop
- Adam Adaptive Moment Estimation

1

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t.$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2.$$

2

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

3

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$