Lecture 8
Introduction to Neural Networks

李杰恩
Chieh-En Lee
celee@nycu.edu.tw

國立陽明交通大學光電工程學系
Department of Photonics
National Yang Ming Chiao Tung University, 30010 Hsinchu, Taiwan

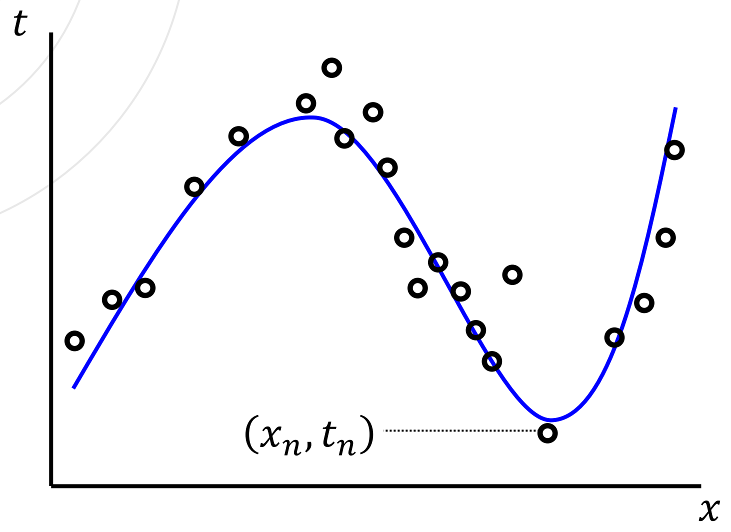
Last Time

- It’s Peter’s show!
- Numerical linear algebra
- Eigenvalues and eigenvectors
- Least squares regression problems

Today

- Curve fitting problem
- Overfitting phenomenon
- Regularization
- Probabilistic perspective on curve fitting problem
- Regression and classification
- Neural networks
- Feed-forward and backward propagation
- Optimization
- Summary

Curve Fitting Problem



objective curve, $y = f(x)$

data, $t_n = f(x_n) + \text{noise}$

How to find a function or a model $y = f(x)$ based on these data (observations) ?

General model

$$y(x, \boldsymbol{\beta}) = \alpha_0 f_0(x) + \alpha_1 f_1(x) + \alpha_2 f_2(x) + \cdots + \alpha_{M-1} f_{M-1}(x) = \sum_{j=0}^{M-1} \alpha_j f_j(x)$$

- For example, a polynomial model

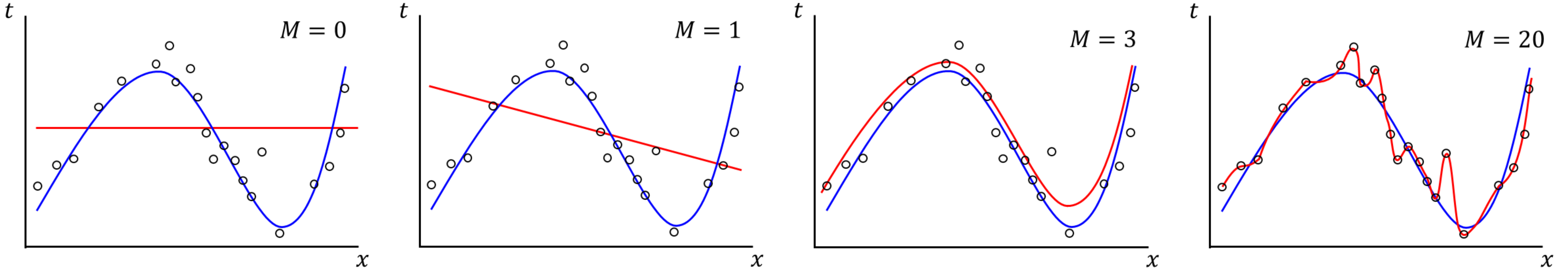
$$y(x, \boldsymbol{\beta}) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{M-1} x^{M-1} = \sum_{j=0}^{M-1} \alpha_j x^j$$

Curve Fitting with Different Models

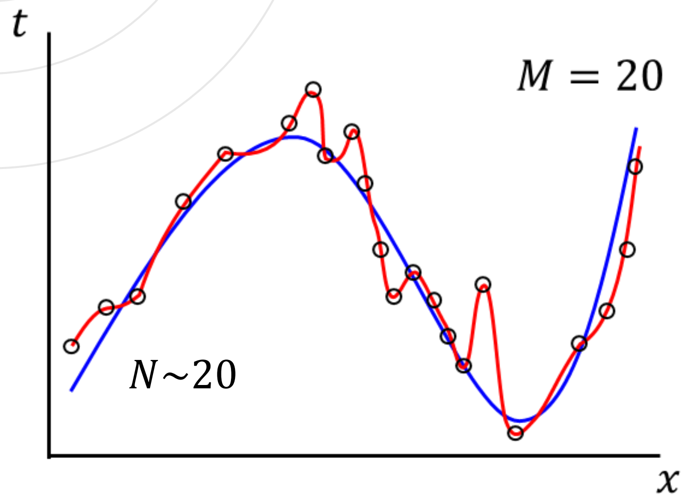
Error function

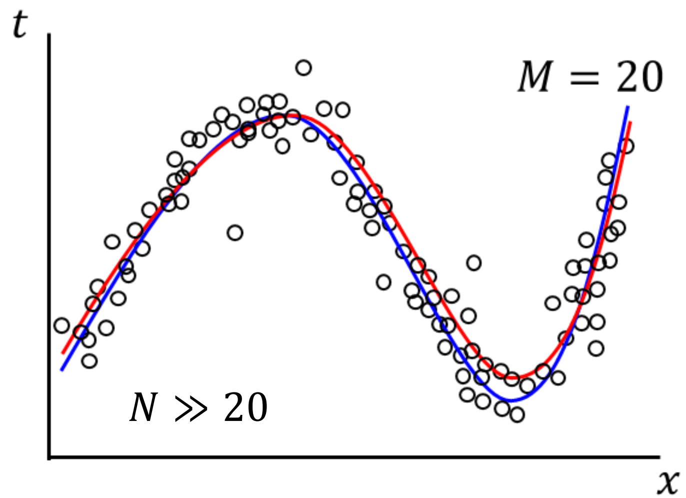
$$E(\boldsymbol{\beta}) = \frac{1}{2}\sum_{n=1}^{N}\{y_n(x_n, \boldsymbol{\beta}) - t_n\}^2$$

Least squares regression



Overfitting Phenomenon





	$M = 0$	$M = 1$	$M = 3$	$M = 20$
α_1	1.45	7.11	0.31	3.35
α_2		-3.02	7.99	251.87
α_3			-25.43	-1450.38
α_4			17.37	9964.87
α_5				-23163.90
$\vdots$				$\vdots$
α_M				-557682.99

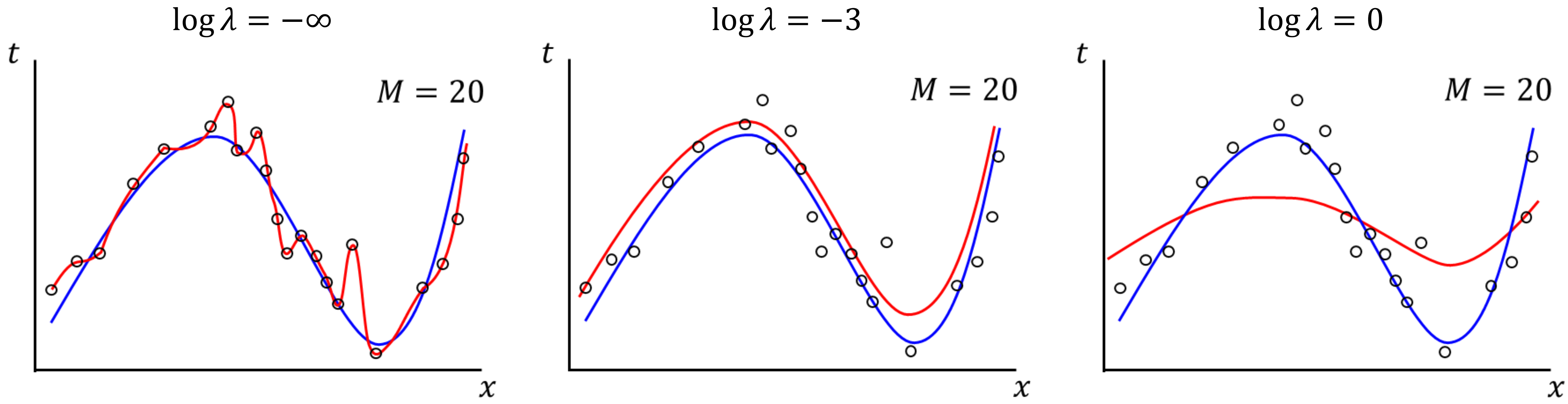
This phenomenon will be suppressed with sufficient data.

Regularization

Error function

$$E(\boldsymbol{\beta}) = \frac{1}{2}\sum_{n=1}^{N}\{y_n(x_n, \boldsymbol{\beta}) - t_n\}^2 + \boxed{\frac{\lambda}{2}\|\boldsymbol{\beta}\|^2}$$

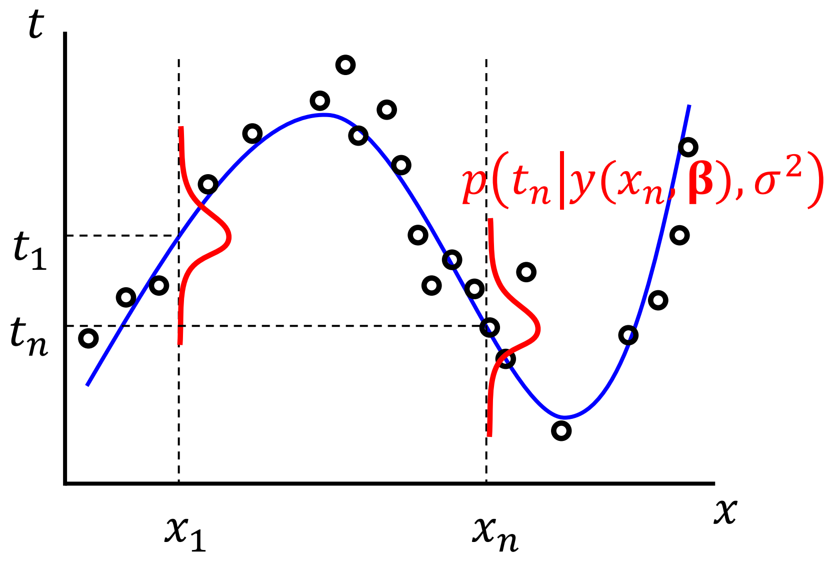
Regularization term



Probabilistic Perspective on Curve Fitting

- Given the value of x_n, we assume the corresponding value of $y(x_n, \boldsymbol{\beta})$ has a Gaussian distribution with a mean equal to t_n.

$$p(t_n | x_n, \boldsymbol{\beta}, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{[t_n - y(x_n, \boldsymbol{\beta})]^2}{2\sigma^2}\right\}$$



- The **likelihood function** of N data

$$p(\mathbf{t} | \mathbf{x}, \boldsymbol{\beta}, \sigma^2) = \sigma^{-N} \cdot (2\pi)^{-\frac{N}{2}} \cdot \exp\left\{-\frac{1}{2\sigma^2} \sum_{i=1}^{N} [t_i - y(x_i, \boldsymbol{\beta})]^2\right\}$$

What is the likelihood function?

Likelihood Function

- Bayes' theorem

$$P(A|B) = \frac{P(B|A) \cdot P(B)}{P(A)} \quad \Rightarrow \quad \underline{P(\mathbf{x}, \boldsymbol{\beta}, \sigma^2|\mathbf{t})} = \frac{P(\mathbf{t}|\mathbf{x}, \boldsymbol{\beta}, \sigma^2) \cdot P(\mathbf{x}, \boldsymbol{\beta}, \sigma^2)}{P(\mathbf{t})}$$

The probability of the model $(\mathbf{x}, \boldsymbol{\beta}, \sigma^2)$ under the condition of observation $\mathbf{t}$.

- We use a likelihood function $\mathcal{L}(\mathbf{x}, \boldsymbol{\beta}, \sigma^2|\mathbf{t})$ to estimate $P(\mathbf{x}, \boldsymbol{\beta}, \sigma^2|\mathbf{t})$ because we do not know the probability of $P(\mathbf{x}, \boldsymbol{\beta}, \sigma^2)$

$$P(\mathbf{x}, \boldsymbol{\beta}, \sigma^2|\mathbf{t}) \rightarrow \mathcal{L}(\mathbf{x}, \boldsymbol{\beta}, \sigma^2|\mathbf{t}) \rightarrow \underline{\mathcal{L}(\mathbf{t}|\mathbf{x}, \boldsymbol{\beta}, \sigma^2)}$$

Maximizing $\mathcal{L}(\mathbf{t}|\mathbf{x}, \boldsymbol{\beta}, \sigma^2)$ is equivalent to maximizing $\mathcal{L}(\mathbf{x}, \boldsymbol{\beta}, \sigma^2|\mathbf{t})$. Therefore, we can find the most reasonable $P(\mathbf{x}, \boldsymbol{\beta}, \sigma^2)$.

Estimations

- Log likelihood function

$$E(\boldsymbol{\beta}) = \ln p\left(\mathbf{t}\middle|\mathbf{x}, \boldsymbol{\beta}, \sigma^2\right) = -N \ln \sigma - \frac{N}{2} \ln 2\pi - \frac{1}{2\sigma^2} \sum_{i=1}^{N} [t_i - y(x_i, \boldsymbol{\beta})]^2$$

- Maximum likelihood (ML) estimation

$$\frac{\partial}{\partial \boldsymbol{\beta}} E(\boldsymbol{\beta}) = 0 \rightarrow \boldsymbol{\beta}_{ML}$$

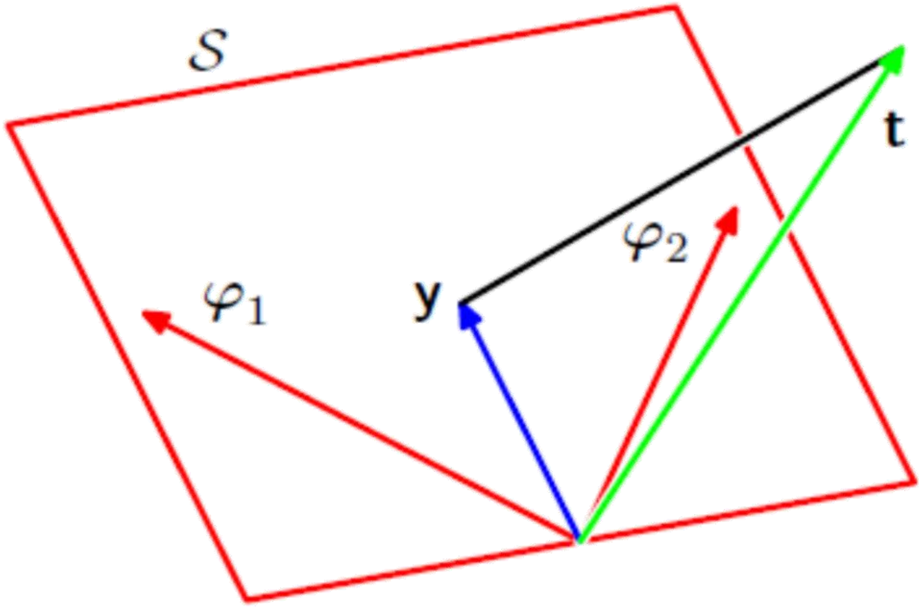
- Maximum a posterior (MAP) estimation

$$E(\boldsymbol{\beta}) = \ln p\left(\mathbf{t}\middle|\mathbf{x}, \boldsymbol{\beta}, \sigma^2\right) = -N \ln \sigma - \frac{N}{2} \ln 2\pi - \frac{1}{2\sigma^2} \sum_{i=1}^{N} [t_i - y(x_i, \boldsymbol{\beta})]^2 - \frac{\lambda}{2} \boldsymbol{\beta}^T \boldsymbol{\beta} \rightarrow \boldsymbol{\beta}_{MAP}$$

constraints on $\boldsymbol{\beta}$

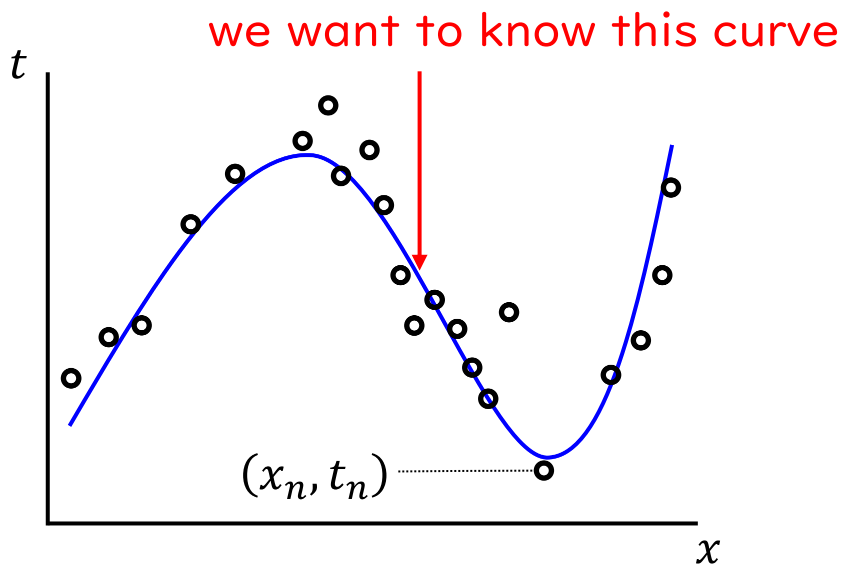
Physical Meaning of ML & MAP

- Find an optimal subspace $S \in \mathbb{R}^M$ expanded by the basis vectors $\{\varphi_1, \varphi_2, \dots, \varphi_M \in \mathbb{R}^M\}$ so that the difference between the projection $\mathbf{y} \in \mathbb{R}^M$ and the observation $\mathbf{t} \in \mathbb{R}^N$ is minimized.

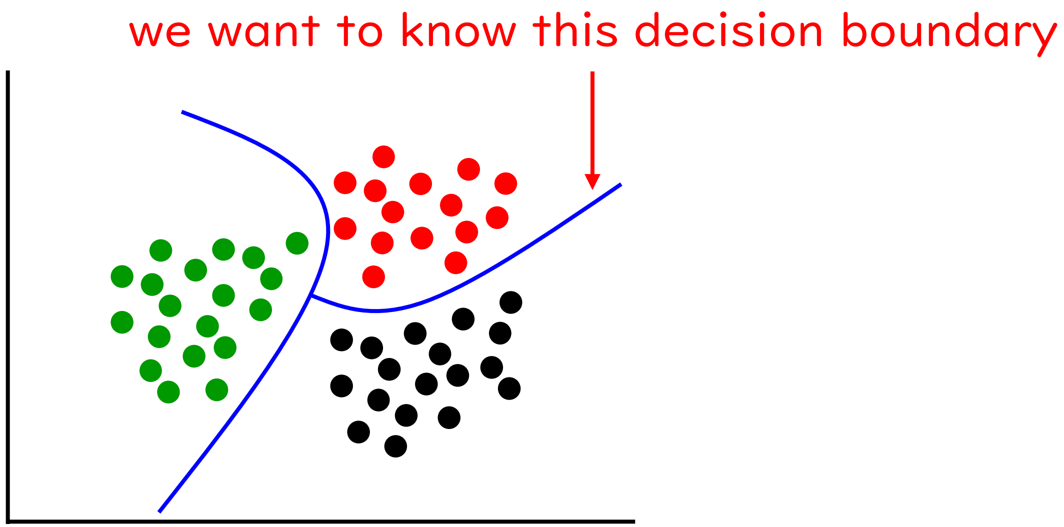


Regression and Classification

- It depends on the characteristic of training target.



continuous training target

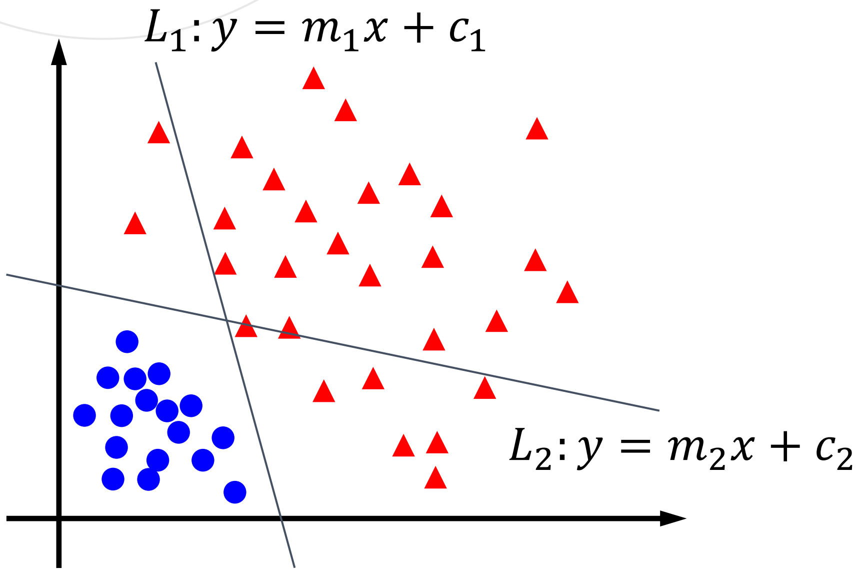


discrete training target

Brief Summary

- A curve fitting problem can be regarded as how to find a function (or a model) based on your data.
- Overfitting phenomenon and regularization
- The probabilistic perspective on curve fitting problem, likelihood function, and estimations
- The difference between regression and classification

Linear and Non-linear Models



- The linear combination of L_1 and L_2

$$y_{new} = w_1 L_1 + w_2 L_2 = \underline{(w_1 m_1 + w_2 m_2)x + (w_1 c_1 + w_2 c_2)}$$

is still linear (a straight line)

- However, if we change to a non-linear model

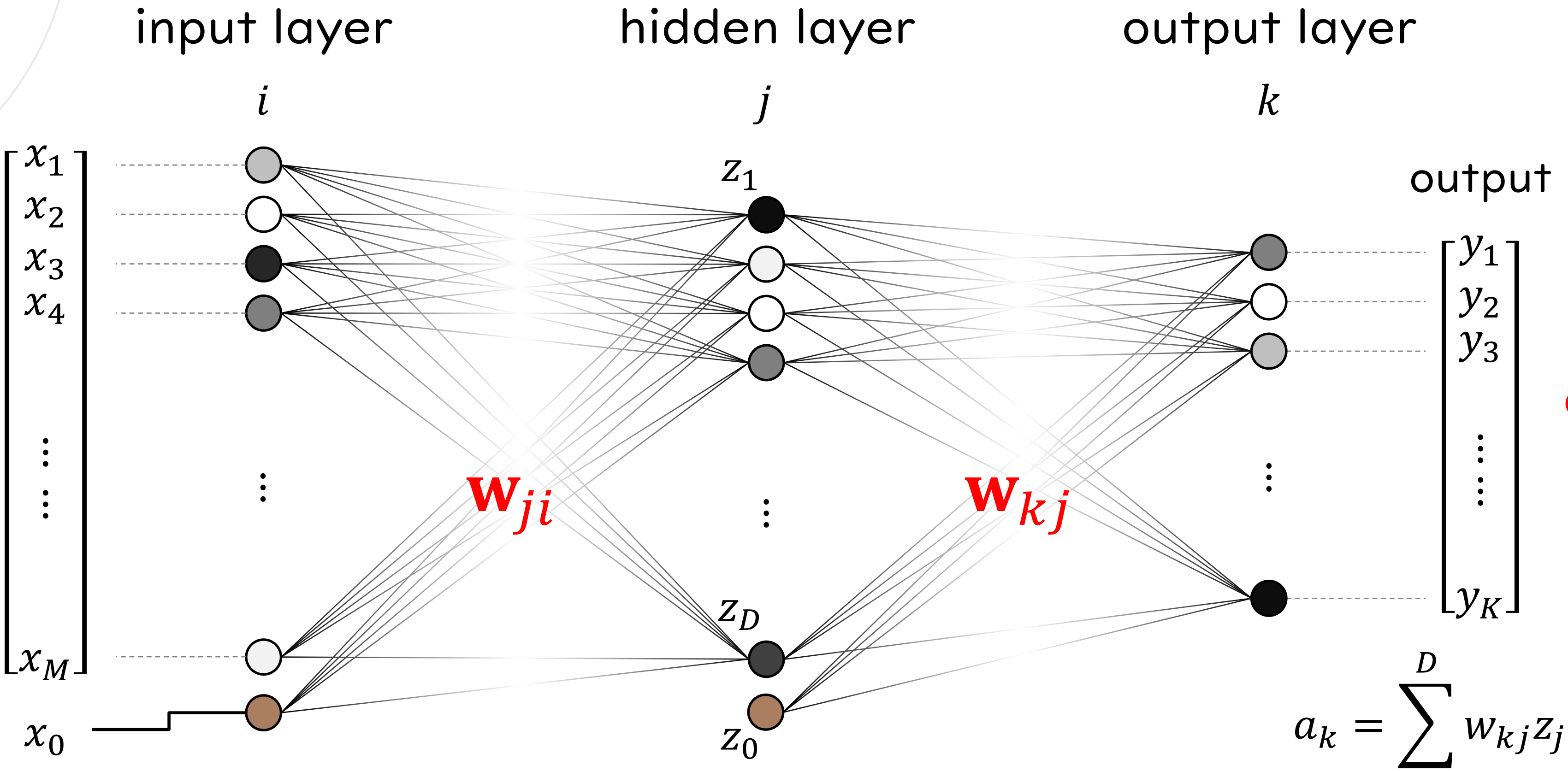
$$y_{new} = f(w_1 L_1) + f(w_2 L_2), \text{ and } f(x) = x^2, e^x, \frac{1}{1+e^{-x}}, \ldots$$

$$\boxed{y_{new} = \begin{cases} [w_1(m_1 x + c_1)]^2 + [w_2(m_2 x + c_2)]^2 \\ e^{w_1(m_1 x + c_1)} + e^{w_2(m_2 x + c_2)} \\ \dfrac{1}{1 + e^{-w_1(m_1 x + c_1)}} + \dfrac{1}{1 + e^{-w_2(m_2 x + c_2)}} \end{cases}}$$

will be different

Neural Network Algorithm

input data



The shape of output vector depends on your issue

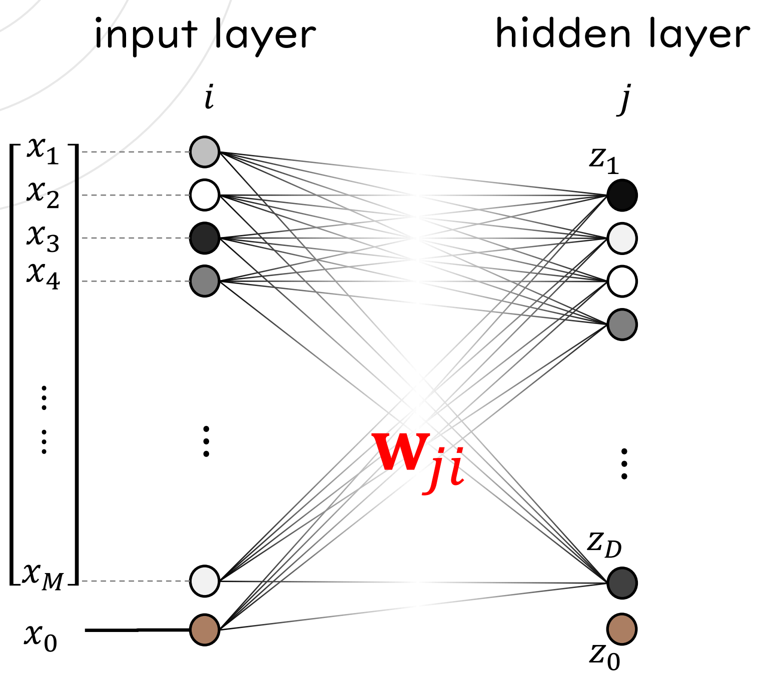
bias node

$$a_j = \sum_{i=1}^{M} w_{ji} x_i + x_0 \qquad z_j = h(a_j)$$

$$a_k = \sum_{j=1}^{D} w_{kj} z_j + z_0$$

$$y_k = h(a_k)$$

Feed-forward Propagation (1/2)



$$a_j = \sum_{i=1}^{M} w_{ji} x_i + x_0$$

linear combination of previous layer

$$z_j = h(a_j)$$

"activate" by a non-linear function

Activation functions (there are more)

Sigmoid	$h(x) = \frac{1}{1+e^{-x}}$
Tangent hyperbolic	$h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Softplus	$h(x) = \log(1+e^x)$
Rectified linear unit (ReLU)	$h(x) = \max(0, x)$

Feed-forward Propagation (2/2)

- Regression model

(x_1, x_2) input data ➡ Hidden layers of your network ➡ $\mathbf{y}$ output

- Classification model

input data ➡ Hidden layers of your network ➡ $\mathbf{y}$ output

Activation functions (there are more)

Sigmoid	$h(x) = \dfrac{1}{1+e^{-x}}$
Tangent hyperbolic	$h(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Softplus	$h(x) = \log(1 + e^{x})$
Rectified linear unit (ReLU)	$h(x) = \max(0, x)$

Loss Calculation (1/2)

- Regression model

(x_1, x_2) input data → Hidden layers of your network → $\mathbf{y}$ output

Calculate the l_k-norm

$$E(\mathbf{w}) = \sum_{n=1}^{N} \|\mathbf{y} - \mathbf{t}\|^k$$

k	Loss type
1	Mean absolute error (MAE)
2	Mean squared error (MSE)
3	Mean cubic error (MCE)
…	…

- Classification model

input data → Hidden layers of your network → $\mathbf{y}$ output →

$$h(a_k) = \begin{cases} \dfrac{1}{1+e^{-a_k}} & \text{binary classification} \\ \dfrac{e^{a_k}}{\sum_m^K e^{a_m}} & \text{multi-class classification} \end{cases}$$

Loss Calculation (2/2)

- Classification model

input data → Hidden layers of your network → $\mathbf{y}$ output →

$$h(a_k) = \begin{cases} \dfrac{1}{1+e^{-a_k}} & \text{binary classification} \\ \dfrac{e^{a_k}}{\sum_m^K e^{a_m}} & \text{multi-class classification} \end{cases}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.97 \\ 0 \\ 0 \\ \vdots \\ 0.01 \\ 0 \end{bmatrix}$$

Training target

k	Encoding vector
1	$[1,0,0,0,\dots]$
2	$[0,1,0,0,\dots]$
3	$[0,0,1,0,\dots]$
…	…

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\left[t_n \ln y_n + (1-t_n)\ln(1-y_n)\right]$$

binary classification

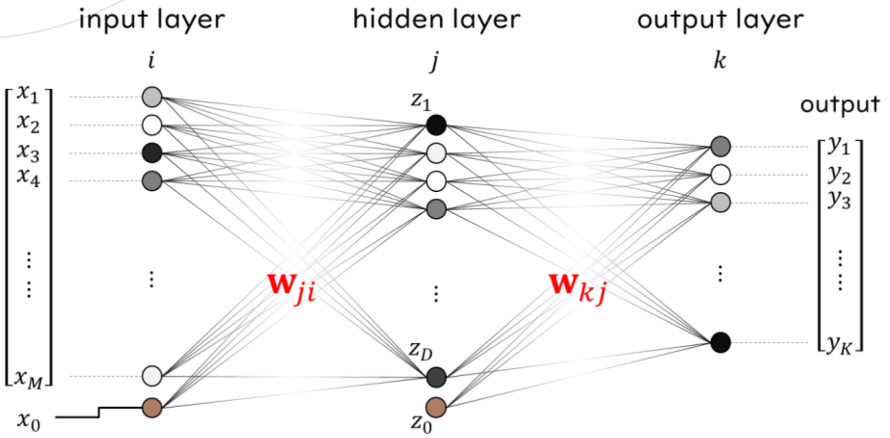
$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

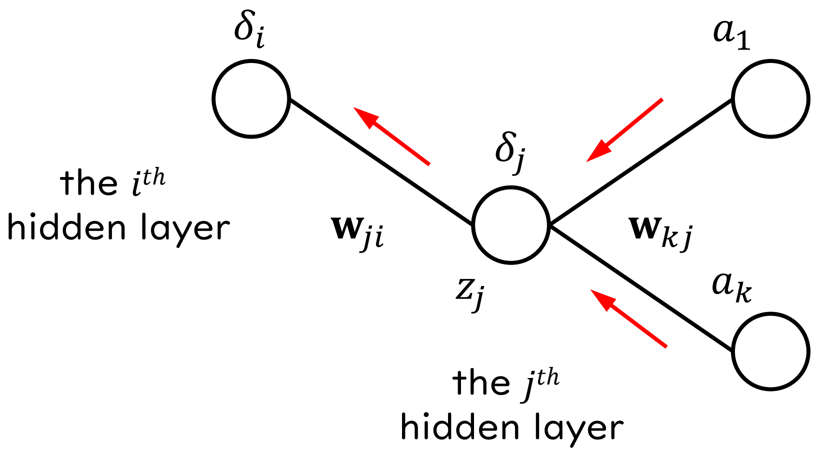
multi-class classification

Back Propagation (1/4)

- Gradient descent (GD) algorithm

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n\left(\mathbf{w}^{(\tau)}\right)$$





$$\delta_1 = \frac{\partial E_n}{\partial a_1} = y_{n1} - t_{n1}$$

$$\delta_k = \frac{\partial E_n}{\partial a_k} = y_{nk} - t_{nk}$$

$$a_j = \sum_{i=1}^{M} w_{ji} x_i + x_0$$

$$z_j = h(a_j)$$

hidden layer

$$a_k = \sum_{j=1}^{D} w_{kj} z_j + z_0$$

$$y_k = h(a_k) = a_k$$

output layer

$$E_n(\mathbf{w}) = \frac{(\mathbf{y}_n - \mathbf{t}_n)^2}{2}$$

$$\frac{\partial E_n}{\partial a_k} = y_{nk} - t_{nk}$$

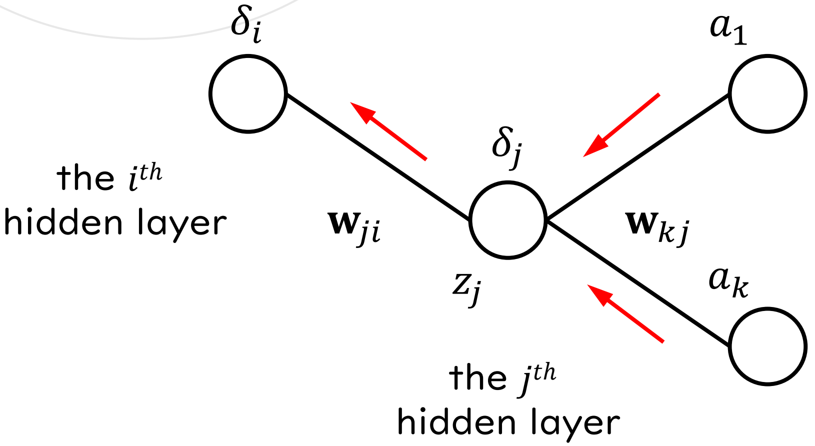
error function

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} = h'(a_j) \cdot \sum_k w_{kj} \delta_k$$

$$\delta_i = \frac{\partial E_n}{\partial a_i} = \sum_j \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \frac{\partial a_j}{\partial a_i} = h'(a_i) \cdot \sum_j w_{ji} \delta_j$$

Back Propagation (2/4)

$$\boxed{\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n\left(\mathbf{w}^{(\tau)}\right)}$$



$$\delta_1 = \frac{\partial E_n}{\partial a_1} = y_{n1} - t_{n1}$$

$$\delta_k = \frac{\partial E_n}{\partial a_k} = y_{nk} - t_{nk}$$

$$a_j = \sum_{i=1}^{M} w_{ji} x_i + x_0$$

$$z_j = h(a_j)$$

hidden layer

$$a_k = \sum_{j=1}^{D} w_{kj} z_j + z_0$$

$$y_k = h(a_k) = a_k$$

output layer

$$E_n(\mathbf{w}) = \frac{(\mathbf{y}_n - \mathbf{t}_n)^2}{2}$$

$$\frac{\partial E_n}{\partial a_k} = y_{nk} - t_{nk}$$

error function

$$\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} - \eta \underline{\nabla E_n\left(\mathbf{w}^{(\tau)}\right)}$$

$$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = \delta_k \cdot z_j$$

$$\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} - \eta \cdot \delta_k \cdot z_j$$

$$\mathbf{w}_{ji}^{(\tau+1)} = \mathbf{w}_{ji}^{(\tau)} - \eta \underline{\nabla E_n\left(\mathbf{w}^{(\tau)}\right)}$$

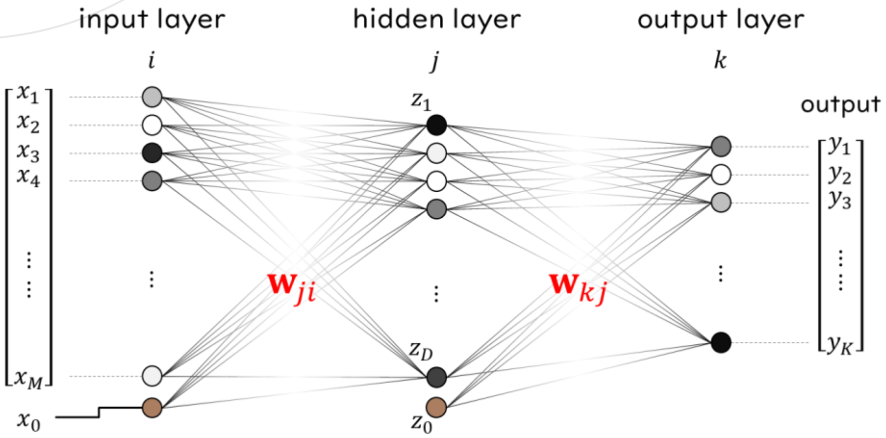
$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j \cdot z_i$$

$$\mathbf{w}_{ji}^{(\tau+1)} = \mathbf{w}_{ji}^{(\tau)} - \eta \cdot \delta_j \cdot z_i$$

Back Propagation (3/4)

- Flowchart of GD algorithm

$$\boxed{\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n\left(\mathbf{w}^{(\tau)}\right)}$$



network output $\mathbf{y}_n$ ⟷ expected output $\mathbf{t}_n$

$$a_j = \sum_{i=1}^{M} w_{ji} x_i + x_0$$

$$z_j = h(a_j)$$

hidden layer

$$a_k = \sum_{j=1}^{D} w_{kj} z_j + z_0$$

$$y_k = h(a_k) = a_k$$

output layer

$$E_n(\mathbf{w}) = \frac{(\mathbf{y}_n - \mathbf{t}_n)^2}{2}$$

$$\frac{\partial E_n}{\partial a_k} = y_{nk} - t_{nk}$$

error function

➡ ①

$$\delta_k = \frac{\partial E_n}{\partial y_k}\frac{\partial y_k}{\partial a_k} = y_{nk} - t_{nk}$$

➡ ②

$$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = \delta_k \cdot z_j$$

$$\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} - \eta \cdot \delta_k \cdot z_j$$

➡ ③

$$\delta_j = h'(a_j) \cdot \sum_k w_{kj} \delta_k$$

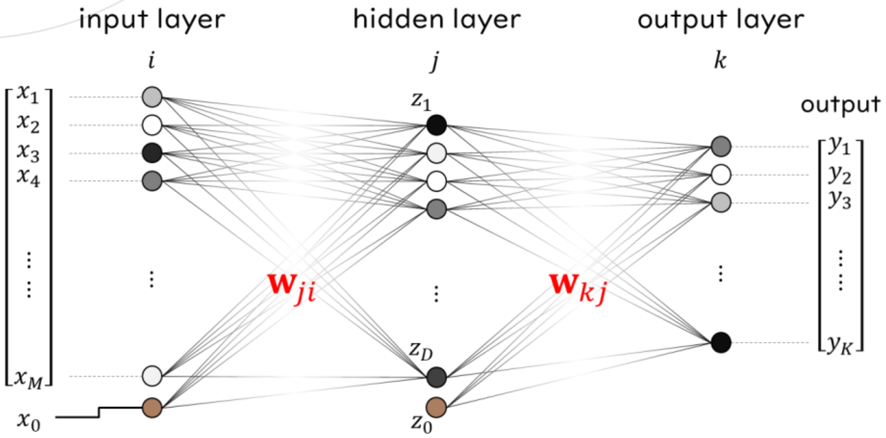
$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j \cdot z_i$$

$$\mathbf{w}_{ji}^{(\tau+1)} = \mathbf{w}_{ji}^{(\tau)} - \eta \cdot \delta_j \cdot z_i$$

Back Propagation (4/4)

- Different loss functions

$$\boxed{\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n\left(\mathbf{w}^{(\tau)}\right)}$$



network output   expected output

$\mathbf{y}_n \longleftrightarrow \mathbf{t}_n$

1

$$\delta_k = \frac{\partial E_n}{\partial y_k}\frac{\partial y_k}{\partial a_k}$$

$$E_n(\mathbf{w}) = \begin{cases} \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2 & \text{regression} \\ -[t_n \ln y_n + (1 - t_n)\ln(1 - y_n)] & \text{binary classification} \\ -\sum_{k=1}^{K} t_{nk} \ln y_{nk} & \text{multi-class classification} \end{cases}$$

$$y_k = h(a_k) = \begin{cases} a_k \\ \dfrac{1}{1 + \exp(-a_k)} \\ \dfrac{\exp(a_k)}{\sum_m^K \exp(a_m)} \end{cases}$$

2

$$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = \delta_k \cdot z_j$$

$$\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} - \eta \cdot \delta_k \cdot z_j$$

3

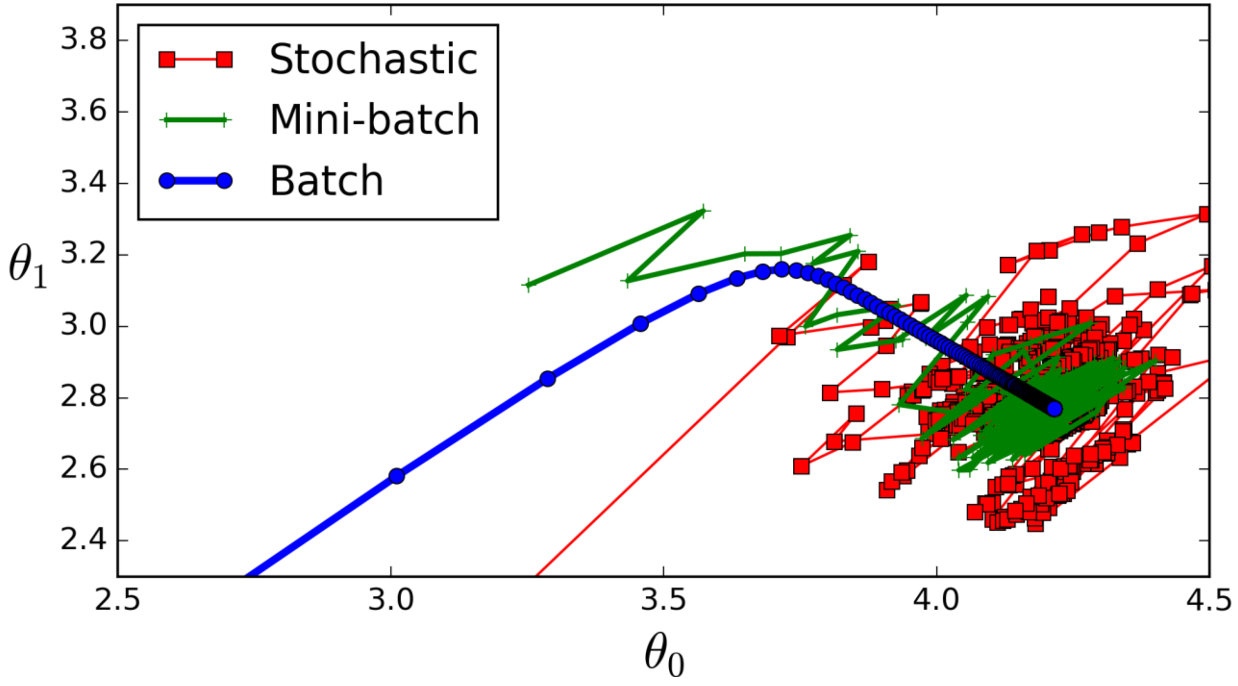
$$\delta_j = h'(a_j) \cdot \sum_k w_{kj}\delta_k$$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j \cdot z_i$$

$$\mathbf{w}_{ji}^{(\tau+1)} = \mathbf{w}_{ji}^{(\tau)} - \eta \cdot \delta_j \cdot z_i$$

Batch Size

- The difference between gradient descent (GD) and stochastic gradient descent (SGD)
- Epoch – a generation of training
- Batch – a small batch of training data
- Iteration – one iteration of training



Optimization (1/3)

- Momentum

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \cancel{\eta \nabla E_n\left(\mathbf{w}^{(\tau)}\right)} \quad \Rightarrow \quad \boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{v}^{(t)}}$$

$$\mathbf{v}^{(t)} = \begin{cases} \gamma \mathbf{g}_t, & t = 0 \\ \boxed{\beta \mathbf{v}^{(t-1)}} + \gamma \mathbf{g}_t, & t \geq 1 \end{cases}$$

$\mathbf{g}_t = \nabla E\left(\mathbf{w}^{(t)}\right)$: the gradient of the t^{th} iteration

$\beta = 0.9$: default parameter

γ: learning rate

Accelerate the learning speed if the gradient direction of the t^{th} and the $(t-1)^{th}$ iteration are in the same direction.

Optimization (2/3)

- Adaptive momentum (ADAM)

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \cancel{\eta \nabla E_n\left(\mathbf{w}^{(\tau)}\right)} \quad \Rightarrow \quad \boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\gamma}{\sqrt{\hat{\mathbf{v}}_t - \epsilon}} \widehat{\mathbf{m}}_t}$$

$$\mathbf{m}_t = \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot \mathbf{g}_t \qquad \widehat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}$$

$$\mathbf{v}_t = \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot \mathbf{g}_t^2 \qquad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}$$

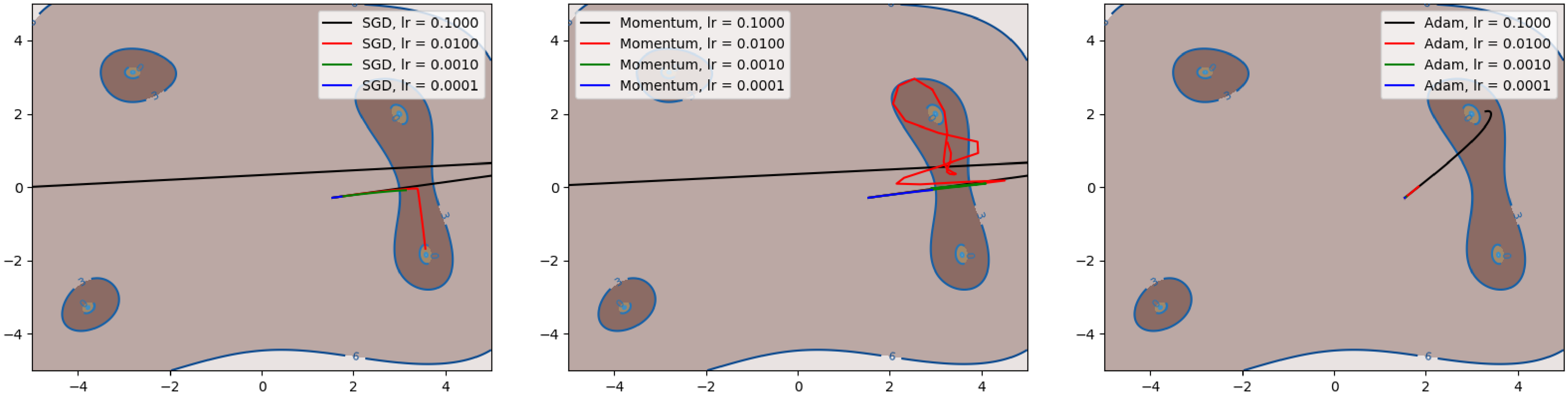
Default parameters:

β_1:	0.9
β_2:	0.999
ϵ:	10^{-8}
γ:	learning rate

Adaptively change the momentum (direction of gradient) and the learning rate during training phase.

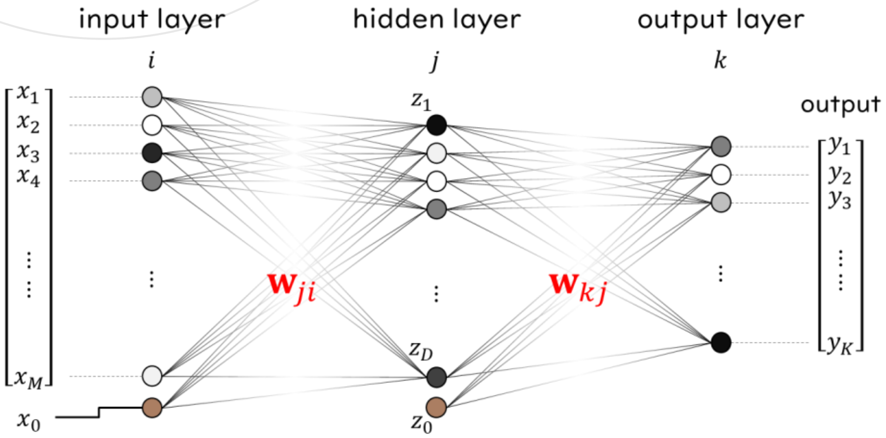
Optimization (3/3)

- Comparison (different learning rates)



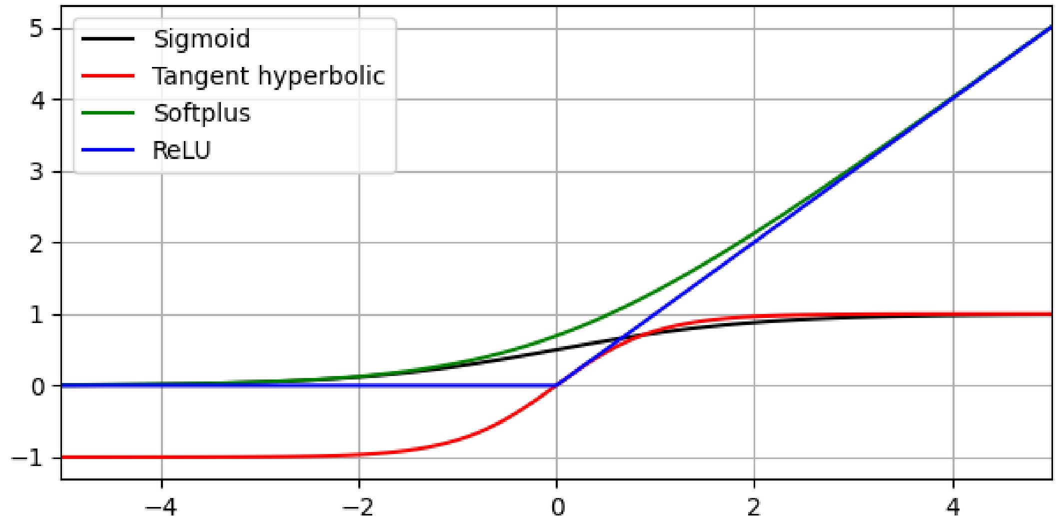
Gradient Explosion and Vanishing

- Review the GD algorithm



network output   expected output

$\mathbf{y}_n \longleftrightarrow \mathbf{t}_n$



$$\delta_j = \boxed{h'(a_j)} \cdot \sum_k w_{kj}\delta_k$$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \boxed{\delta_j} \cdot z_i$$

$$\mathbf{w}_{ji}^{(\tau+1)} = \mathbf{w}_{ji}^{(\tau)} - \eta \cdot \boxed{\delta_j} \cdot z_i$$

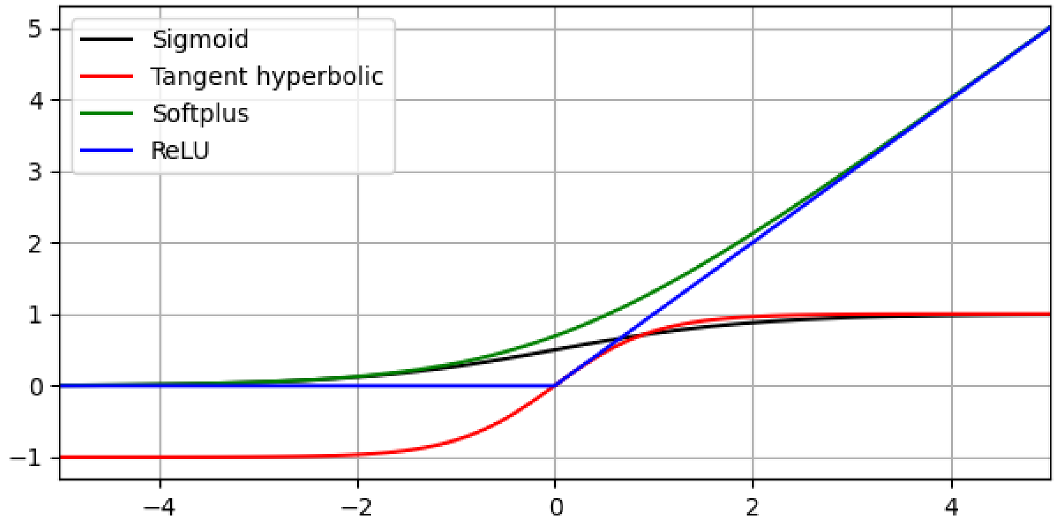
$$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = \boxed{\delta_k} \cdot z_j$$

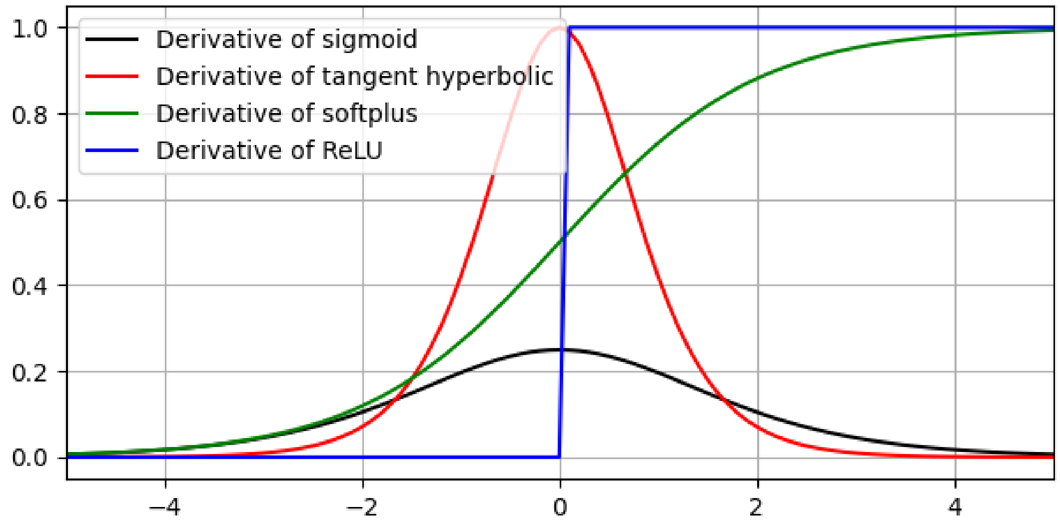
$$\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} - \eta \cdot \boxed{\delta_k} \cdot z_j$$

The main reason of gradient vanishing is the derivative of activation function.

Batch Normalization

- The main reason of gradient vanishing is the derivative of activation function.





- Normalize the data distribution to zero mean in the output of each layer.

Summary

- The difference between linear and non-linear models
- Feed-forward propagation, loss calculation, back propagation, and optimization of neural network
- Gradient explosion and vanishing, batch normalization
- Your final homework