

# 1 Course Introduction

Zhonglei Wang

WISE and SOE, XMU, 2025

# Contents

1. What we have learnt

2. Problems

3. Learning goal

4. Cross entropy

# What we have learnt

## 1. Types of datasets:

- Training data of size  $n : \{(\mathbf{x}_i, y_i) : i = 1, \dots, n\}$
- Test data

## 2. Regressions:

- Linear regression:  $y_i = b_0 + \mathbf{x}_i^T \mathbf{w}_0 + \epsilon_i \quad (i = 1, \dots, n)$
- Logistic regression:  $\text{logit}P(y_i = 1 \mid \mathbf{x}_i) = b_0 + \mathbf{x}_i^T \mathbf{w}_0 \quad (i = 1, \dots, n)$ 
  - ▷  $\text{logit}p = \log(p) - \log(1 - p), \quad \forall p \in (0, 1)$
- Multinomial logistic regression (Softmax regression)

# Notations

1.  $n$  : number of examples in the training data
2.  $d$  : dimension of features
3.  $\mathbf{X} \in \mathbb{R}^{n \times d}$  : design matrix
  - $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$
  - $\mathbf{x}_i \in \mathbb{R}^d$  : feature for the  $i$ th example
4.  $\mathbf{Y} \in \mathbb{R}^{n \times 1}$  : vector of labels
  - $\mathbf{Y} = (y_1, \dots, y_n)^T$
  - $y_i \in \mathbb{R}$  : label for the  $i$ th example

# Linear regression -- Model setup

## 1. Training dataset

- Feature, label:  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}$  ( $i = 1, \dots, n$ )

## 2. Goal

- Learn the relationship between  $\mathbf{x}$  and  $y$

## 3. Model

- $y_i = b_0 + \mathbf{x}_i^T \mathbf{w}_0 + \epsilon_i$  ( $i = 1, \dots, n$ )
  - ▷  $\boldsymbol{\theta}_0 = (b_0, \mathbf{w}_0^T)^T$  : true model parameters
  - ▷  $\epsilon_i$  : noise satisfying  $E(\epsilon_i | \mathbf{x}_i) = 0$ , and  $\text{var}(\epsilon_i | \mathbf{x}_i) = \sigma^2 > 0$

# Linear regression -- Parameter estimation

1. **Loss function:** For the  $i$ th example,

- $l_2$ -loss:  $\mathcal{L}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
- $\hat{y}_i = b + \mathbf{x}_i^T \mathbf{w}$
- $\hat{y}_i$  is a function of  $\boldsymbol{\theta} = (b, \mathbf{w}^T)^T$ , but we omit its argument for simplicity
- $\mathcal{L}(y_i, \hat{y}_i)$  is actually a function of model parameters  $\boldsymbol{\theta}$

2. **Cost function**

$$\mathcal{J}(\boldsymbol{\theta}) = n^{-1} \sum_{i=1}^n \mathcal{L}(y_i, \hat{y}_i) = n^{-1} \sum_{i=1}^n (y_i - b - \mathbf{x}_i^T \mathbf{w})^2$$

# Linear regression -- Parameter estimation

1. Solve

$$\frac{\mathcal{J}}{\partial b} = 0, \quad \frac{\mathcal{J}}{\partial \mathbf{w}} = 0$$

2. Problem: low computational efficiency due to summation

# Vectorization

1. Denote  $\hat{\mathbf{Y}} = b\mathbf{1}_n + \mathbf{X}\mathbf{w}$

- $\mathbf{1}_n = (1, \dots, 1)^T \in \mathbb{R}^{n \times 1}$  : vector of 1's with length  $n$

2. Cost function

$$\begin{aligned}\mathcal{J}(\boldsymbol{\theta}) &= n^{-1}(\mathbf{Y} - \hat{\mathbf{Y}})^T(\mathbf{Y} - \hat{\mathbf{Y}}) \\ &= n^{-1}(\mathbf{Y} - b\mathbf{1}_n - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - b\mathbf{1}_n - \mathbf{X}\mathbf{w}) \\ &= n^{-1}(\mathbf{Y} - \tilde{\mathbf{X}}\boldsymbol{\theta})^T(\mathbf{Y} - \tilde{\mathbf{X}}\boldsymbol{\theta})\end{aligned}$$

- $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n)^T$  (Integrate  $b$  and  $\mathbf{w}$  together)
- $\tilde{\mathbf{x}}_i = (1, \mathbf{x}_i^T)^T$

# Vectorization

1. Consider

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} = 0$$

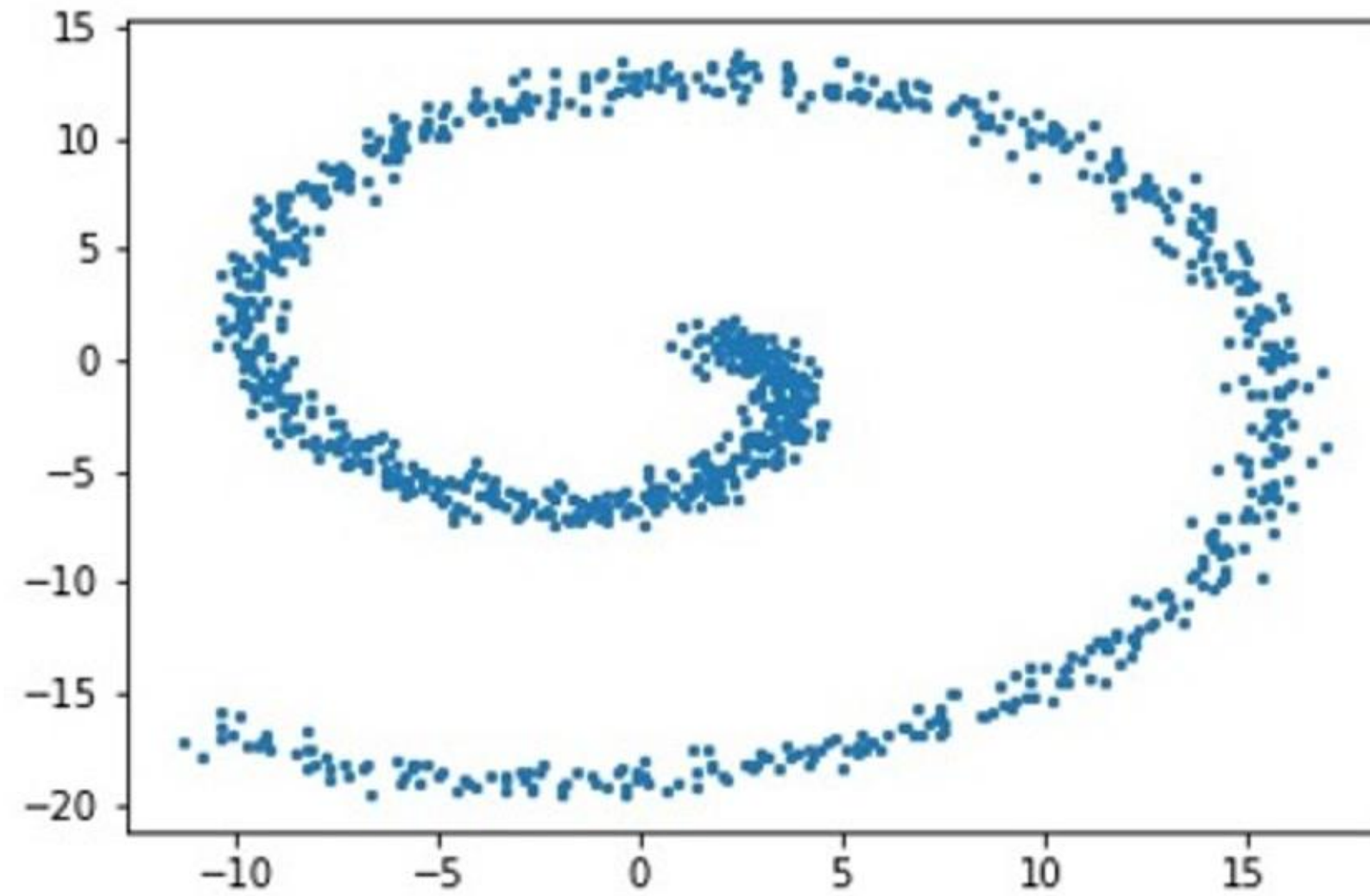
2. Obtain the **normal equation**

$$\tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \boldsymbol{\theta} = \tilde{\mathbf{X}}^T \mathbf{Y}$$

3. Solution (**Closed-form or analytical solution**)

$$\hat{\boldsymbol{\theta}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{Y}$$

# Example



# Logistic regression -- Model setup

## 1. Training data

- Feature, labels:  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \{0, 1\}$  ( $i = 1, \dots, n$ )

## 2. Goal

- Learn the relationship between  $\mathbf{x}$  and  $y$

## 3. Model

$$P(y_i = 1 \mid \mathbf{x}_i) = \sigma(\mathbf{b}_0 + \mathbf{x}_i^T \mathbf{w}_0) \quad (i = 1, \dots, n)$$

- $\sigma(z) = \{1 + \exp(-z)\}^{-1}$  (Why we need this transformation?)

# Logistic regression -- Parameter estimation

1. **Loss function:** For the  $i$ th example,

- Cross entropy (minus log-likelihood)

$$\mathcal{L}(y_i, a_i) = -\{y_i \log a_i + (1 - y_i) \log(1 - a_i)\}$$

▷  $a_i = \sigma(z_i)$  (Estimated probability)

▷  $z_i = b + \mathbf{x}_i^T \mathbf{w}$

- $a_i$  is a function of  $\boldsymbol{\theta} = (b, \mathbf{w}^T)^T$ , but we omit its argument for simplicity
- Cross entropy is actually a function of  $\boldsymbol{\theta}$

# Logistic regression -- Parameter estimation

## 1. Cost function

$$\begin{aligned}\mathcal{J}(\boldsymbol{\theta}) &= n^{-1} \sum_{i=1}^n \mathcal{L}\{y_i, a_i\} \\ &= -n^{-1} \sum_{i=1}^n \{y_i \log a_i + (1 - y_i) \log\{1 - a_i\}\}\end{aligned}$$

- $\boldsymbol{\theta} = (b, \boldsymbol{w}^T)^T$
- $a_i = \sigma(b + \boldsymbol{x}_i^T \boldsymbol{w})$

# Logistic regression -- Parameter estimation

1. Solve

$$\frac{\mathcal{J}}{\partial \boldsymbol{\theta}} = 0$$

2. No analytical solution

# Newton-Raphson algorithm

Step 1. Randomly initialize  $\boldsymbol{\theta}^{(0)}$

Step 2. Based on  $\boldsymbol{\theta}^{(t)}$  obtain

$$\nabla \mathcal{J} \left( \boldsymbol{\theta}^{(t)} \right) = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} \left( \boldsymbol{\theta}^{(t)} \right)$$

$$H(\mathcal{J}) \left( \boldsymbol{\theta}^{(t)} \right) = \frac{\partial^2 \mathcal{J}}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \left( \boldsymbol{\theta}^{(t)} \right)$$

Step 3. Update parameter

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \left[ H(\mathcal{J}) \left( \boldsymbol{\theta}^{(t)} \right) \right]^{-1} \nabla \mathcal{J} \left( \boldsymbol{\theta}^{(t)} \right)$$

Step 4. Go back to Step 2 until convergence

# Discussion

1. Fast convergence by incorporating a Hessian matrix
2. Computation efficiency is sacrificed at the same time
3. Feasible when the parameter dimension is low
4. Not applicable for deep learning models
5. What algorithms should we use for deep learning models?

# (Batch) gradient descent algorithm

Step1. Randomly initialize  $\boldsymbol{\theta}^{(0)}$

Step2. Based on  $\boldsymbol{\theta}^{(t)}$  obtain

$$\nabla \mathcal{J} \left( \boldsymbol{\theta}^{(t)} \right) = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} \left( \boldsymbol{\theta}^{(t)} \right)$$

Step3. Update parameter

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla \mathcal{J} \left( \boldsymbol{\theta}^{(t)} \right)$$

Step4. Go back to Step 2 until convergence

# (Batch) gradient descent algorithm

1.  $\alpha$  : learning rate

- Controls the speed of convergence
- Affects the performance of the model
- To be discussed

# Comparison of algorithms

## 1. Newton-Raphson algorithm

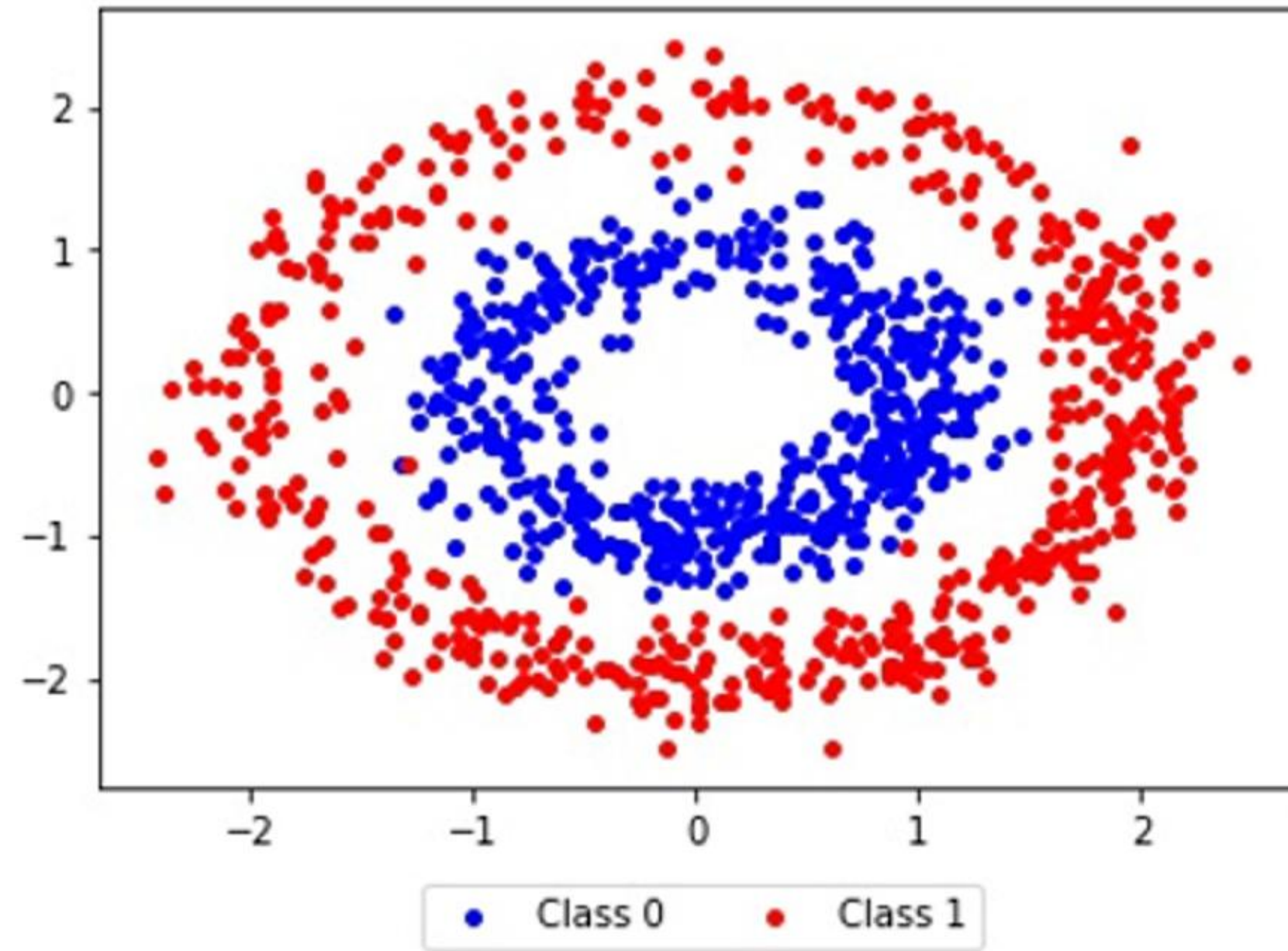
- Advantages:
  - ▷ Fast convergence rate
  - ▷ High accuracy
- Disadvantages:
  - ▷ Low computation efficiency due to the Hessian matrix
  - ▷ Low stability if the Hessian matrix is ill-conditioned

# Comparison of algorithms

## 1. (Batch) gradient descent algorithm

- Advantages :
  - ▷ High computation efficiency
  - ▷ Easy to implement
- Disadvantages :
  - ▷ Low convergence rate
  - ▷ Sensitive to learning rate

# Example



# Problems

## 1. Model misspecification

- Traditional models are too simple
- Reality, however, is COMPLEX

# Problems

## 1. Model misspecification

- Traditional models are too simple
- Reality, however, is COMPLEX

# Learning goal

1. Fully connected neural network (**multiple layer perceptron**)
2. Convolutional Neural Network (**CNN**, ...)
3. Sequential modeling (**RNN**, **LSTM**, **transformers**, ...)
4. Other topics (**GNN**, **GCN**, ...)

# Learning goal

1. Fully connected neural network (**multiple layer perceptron**)
2. Convolutional Neural Network (**CNN**, ...)
3. Sequential modeling (**RNN**, **LSTM**, **transformers**, ...)
4. Other topics (**GNN**, **GCN**, ...)

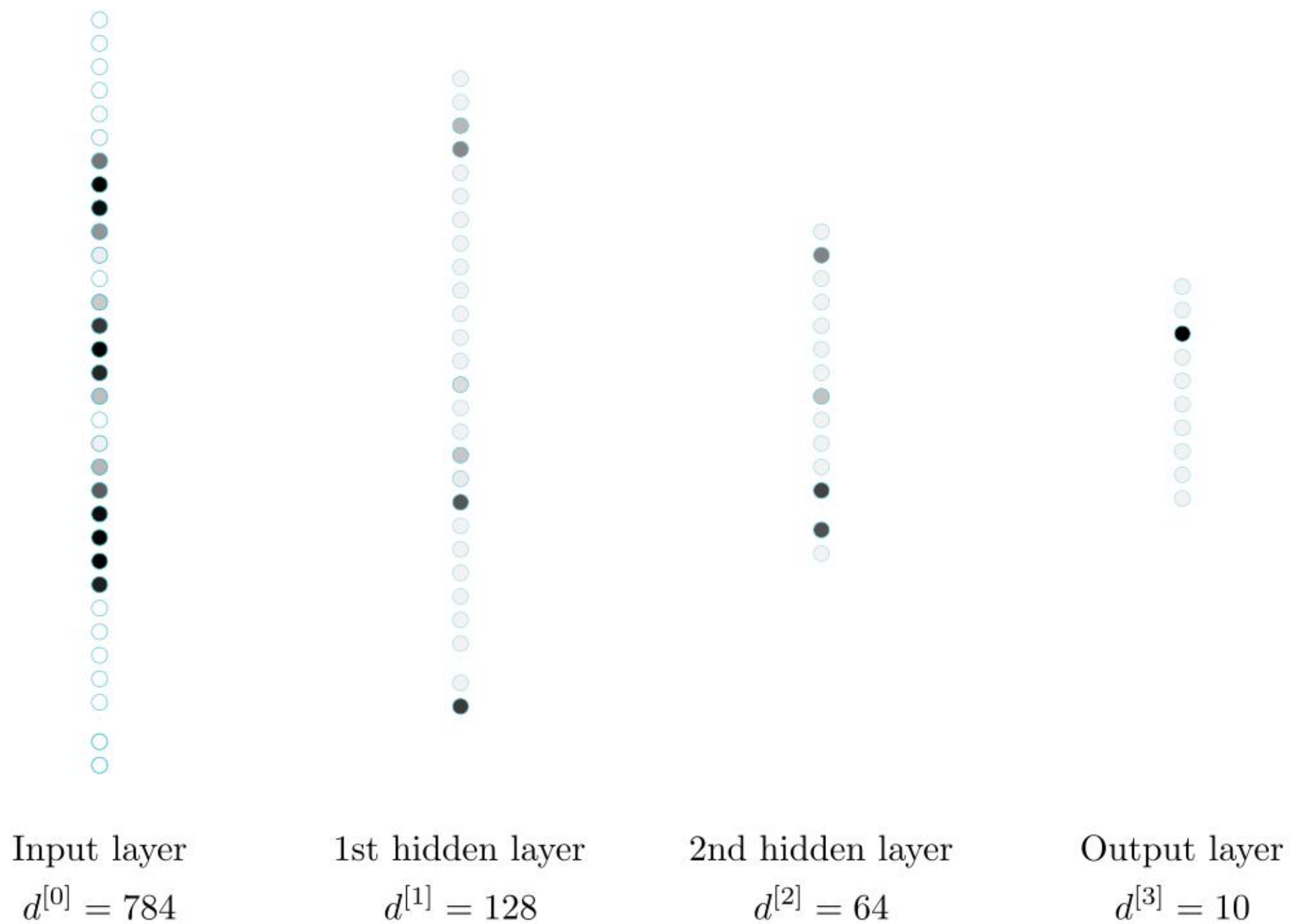
# FNN Example

Test image



Model (FNN with 2 hidden layers)

Estimated result



# FNN Example

Test image



Model (FNN with 2 hidden layers)



Input layer  
 $d^{[0]} = 784$



1st hidden layer  
 $d^{[1]} = 128$



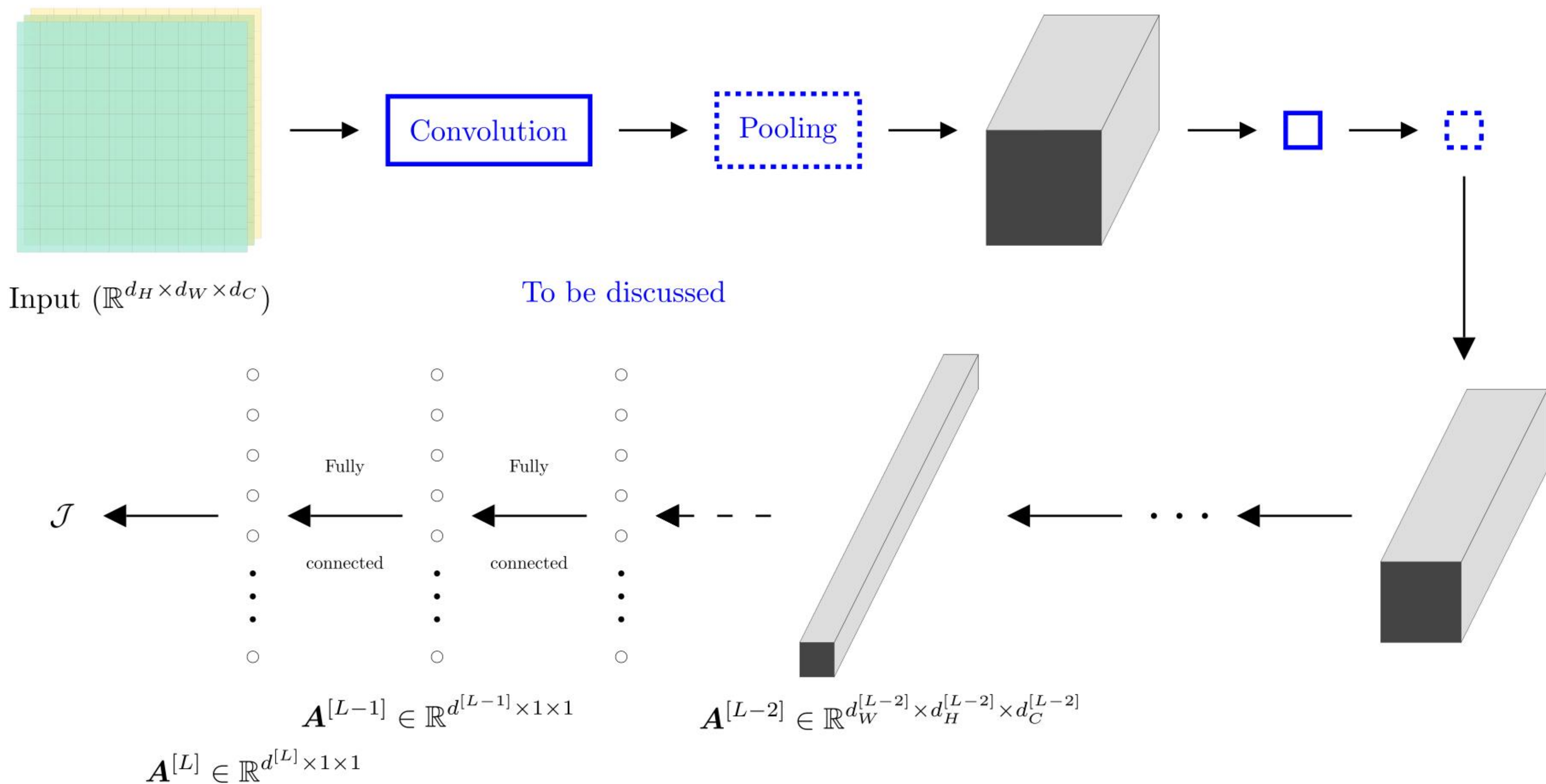
2nd hidden layer  
 $d^{[2]} = 64$



Output layer  
 $d^{[3]} = 10$

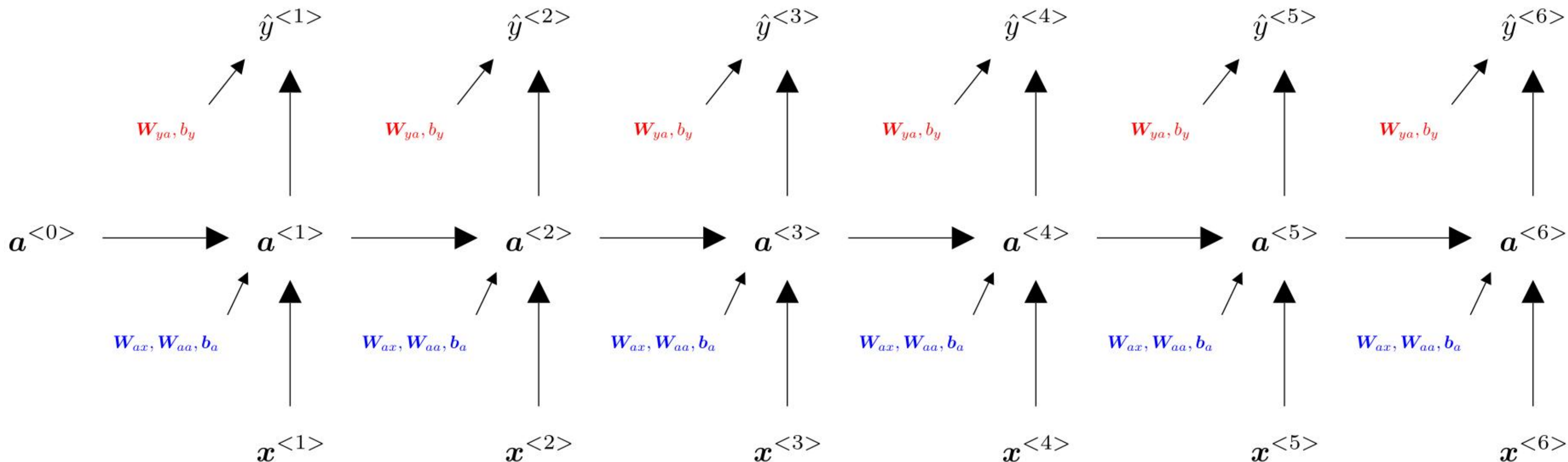
→ 2

Estimated result



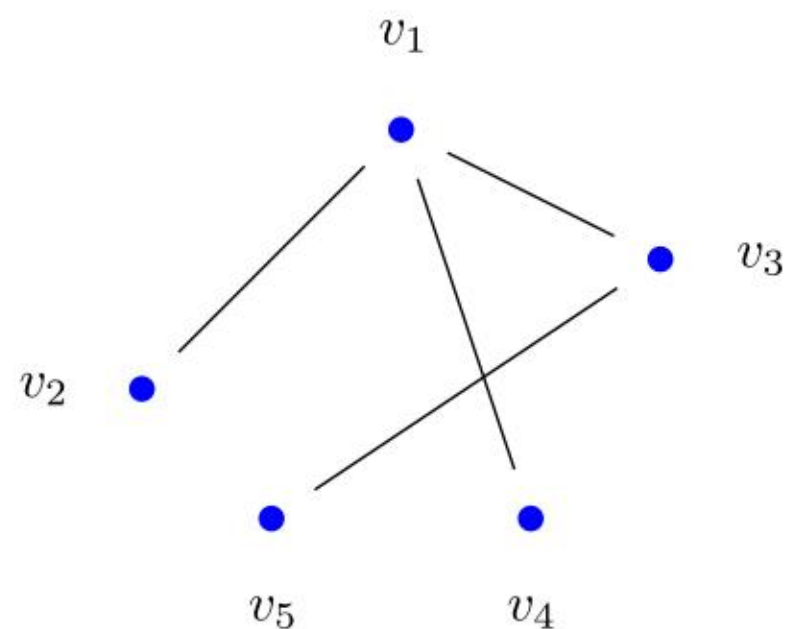
# Building block for RNN

$$\hat{y}^{<i>} = \sigma_{ya}(\mathbf{W}_{ya}\mathbf{a}^{<i>} + b_y) \quad (i = 1, \dots, 6)$$



$$\mathbf{a}^{<i>} = \sigma_{ax}(\mathbf{W}_{ax}\mathbf{x}^{<i>} + \mathbf{W}_{aa}\mathbf{a}^{<i-1>} + b_a) \quad (i = 1, \dots, 6)$$

# Undirected graph



Vertex set:  $\mathcal{V} = \{v_i : i = 1, \dots, n\}$

Edge set:  $\mathcal{E}$

Undirected graph:  $\mathcal{G} = \mathcal{V} \cup \mathcal{E}$

Adjacent matrix (binary or weighted)

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Degree matrix (diagonal, summation of each row in  $A$ )

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

# Cross entropy

1. A fact: under regularity conditions,

$$E_{X \sim P} \{\log p(X)\} \geq E_{X \sim P} \{\log q(X)\}$$

- $X \sim P$  : random variable  $X$  is generated from a distribution  $P$
- $p(x)$  : density function of the distribution  $P$
- $q(x)$  : another density function of a certain distribution

2. Cross entropy for two density functions  $p$  and  $q$

$$H(p, q) = -E_{X \sim P} \{\log q(X)\} = -E \{\log q(X)\}$$

- Measures how well a density  $q$  approximates the distribution of  $X$ .

# Cross entropy

1. Only observe a random sample  $\{x_1, \dots, x_n\}$
2. Cross entropy  $H(p, q)$  can be approximated by

$$\hat{H}(p, q) = -\frac{1}{n} \sum_{i=1}^n \log q(x_i)$$

- It is the cross entropy between the empirical density  $P_n(x) = n^{-1} \sum_{i=1}^n \delta(x = x_i)$  and  $q$
3. Goal: Find a density function  $q$  to minimize  $\hat{H}(p, q)$ 
    - That is, to approximate the distribution of the random sample well

# Revisit logistic regression

1. Assume  $\{(\mathbf{x}_i, y_i) : i = 1, \dots, n\}$  is i.i.d. with density function

$$p(\mathbf{x}, y) = p(\mathbf{x})p(y \mid \mathbf{x})$$

- $p(\mathbf{x})$  : unspecified marginal density for  $\mathbf{x}$
- $p(y \mid \mathbf{x}) = \sigma(\textcolor{red}{b}_0 + \mathbf{x}_i^T \textcolor{red}{w}_0)$  : parametric conditional density for  $y$

2. Goal: Find density  $q(\mathbf{x}, y)$  to minimize

$$\hat{H}(p, q) = -\frac{1}{n} \sum_{i=1}^n \log q(\mathbf{x}_i, y_i)$$

# Revisit logistic regression

## 1. A little more math

$$\begin{aligned} -\frac{1}{n} \sum_{i=1}^n \log q(\mathbf{x}_i, y_i) &= -\frac{1}{n} \sum_{i=1}^n \{\log q(\mathbf{x}_i) + \log q(y_i \mid \mathbf{x}_i)\} \\ &= -\frac{1}{n} \sum_{i=1}^n \log q(\mathbf{x}_i) - \frac{1}{n} \sum_{i=1}^n [y_i \log a_i(\boldsymbol{\theta}) + (1 - y_i) \log \{1 - a_i(\boldsymbol{\theta})\}] \end{aligned}$$

- $q(\mathbf{x}, y) = q(\mathbf{x})q(y \mid \mathbf{x})$
- $q(\mathbf{x})$  : unspecified
- $q(y \mid \mathbf{x}) = a(\boldsymbol{\theta})^y \{1 - a(\boldsymbol{\theta})\}^{1-y}$
- $a(\boldsymbol{\theta}) = \sigma(b + \mathbf{x}^T \mathbf{w})$  : with parameter  $\boldsymbol{\theta} = (b, \mathbf{w}^T)^T$

# Revisit logistic regression

1. Thus,

$$-\frac{1}{n} \sum_{i=1}^n \log q(\mathbf{x}_i, y_i) = -\frac{1}{n} \sum_{i=1}^n \log q(\mathbf{x}_i) - \frac{1}{n} \sum_{i=1}^n [y_i \log a_i(\boldsymbol{\theta}) + (1 - y_i) \log \{1 - a_i(\boldsymbol{\theta})\}]$$

- The **first** part is of no interest
- We focus on estimating parameters in the **second** part

2. Thus, it is equivalent to minimizing

$$-\frac{1}{n} \sum_{i=1}^n [y_i \log a_i(\boldsymbol{\theta}) + (1 - y_i) \log \{1 - a_i(\boldsymbol{\theta})\}]$$

- The cost function for logistic regression