AI and Deep Learning

1 Course Introduction

Zhonglei Wang

WISE, SOE, CHOW

XMU

(Version v1)

Contents

What we have learnt

1. Types of datasets:
   - Training data of size n : $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$
     - $\boldsymbol{x}_i \in \mathbb{R}^{d\times 1}$: Feature for the ith example
     - $y_i \in \mathbb{R}$: Label for the ith example
   - Test data
2. Regressions:
   - Linear regression: $y_i = b + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w} + \epsilon_i \quad (i = 1, \ldots, n)$
   - Logistic regression: $\mathrm{logit} P(y_i = 1 \mid \boldsymbol{x}_i) = b + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w} \quad (i = 1, \ldots, n)$
     - $\mathrm{logit} p = \log(p) - \log(1 - p), \quad \forall p \in (0, 1)$
   - Multinomial logistic regression (softmax regression)

Notations

1. n : Number of examples in the training data
2. d : Dimension of features
3. $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n \times d}$: Design matrix
4. $\boldsymbol{Y} = (y_1, \ldots, y_n)^{\mathrm{T}} \in \mathbb{R}^{n \times 1}$: Vector of labels

Linear regression -- Model setup

1. Training dataset
   - Feature $\boldsymbol{x}_i \in \mathbb{R}^{d\times 1}$, label $y_i \in \mathbb{R}$ $\quad (i = 1, \ldots, n)$
2. Goal
   - Learn the relationship between feature $\boldsymbol{x}$ and label y
3. True model (used to generate data)
   - $y_i = b_0 + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w}_0 + \epsilon_i \quad (i = 1, \ldots, n)$
     - $\boldsymbol{\theta}_0 = (b_0, \boldsymbol{w}_0{}^{\mathrm{T}})^{\mathrm{T}}$: True model parameters
     - ϵ_i : Noise satisfying $E(\epsilon_i \mid \boldsymbol{x}_i) = 0$, and $\mathrm{var}(\epsilon_i \mid \boldsymbol{x}_i) = \sigma^2 > 0$

Linear regression -- Model setup

1. (Assumed) Proposed model (used to fit data)

- $E(y_i \mid \boldsymbol{x}_i) = b + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w} \quad (i = 1, \ldots, n)$
  - $\boldsymbol{\theta} = (b, \boldsymbol{w}^{\mathrm{T}})^{\mathrm{T}}$: (Proposed) Model parameters
  - By fitting a model, we mean to estimate model parameters by training data
  - Have the same form with the true model, but we need to estimate parameters

Linear regression -- Parameter estimation

1. Loss function: For the ith example,

- l_2-loss: $\mathcal{L}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2/2$
- $\hat{y}_i = b + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w}$
- $\hat{y}_i$ is a function of $\boldsymbol{\theta} = (b, \boldsymbol{w}^{\mathrm{T}})^{\mathrm{T}}$
- $\mathcal{L}(y_i, \hat{y}_i)$ is also a function of model parameters $\boldsymbol{\theta}$
- However, in the above expressions, we omit its argument for simplicity
- The loss function is a convex function of the model parameter $\boldsymbol{\theta}$

Linear regression -- Parameter estimation

1. Cost function (Mean Squared Error, MSE)

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

2. Sometimes, we do not distinguish a loss function and a cost function

3. Solve

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

4. Pitfall: Low computational efficiency due to summation

5. Question: Why is that a disaster for Python to use for-loops for summation, especially when the sample size is extremely large?

Vectorization

1. Consider

$$\sum_{i=1}^{m} a_i b_i$$

   - $a_i \in \mathbb{R}, \quad b_i \in \mathbb{R}$

2. For Python and other softwares, the computation efficiency of using for-loops for summation is quite low
3. However, matrix calculation is very efficient
4. Thus, we use vectorization for the summation

Vectorization

1. Vectorization:

$$\sum_{i=1}^{m} a_i b_i = \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}$$

- $\boldsymbol{a} = (a_1, \ldots, a_m)^{\mathrm{T}} \in \mathbb{R}^{m \times 1}$
- $\boldsymbol{b} = (b_1, \ldots, b_m)^{\mathrm{T}} \in \mathbb{R}^{m \times 1}$

2. Avoid for-loops when linear algebra can be used

Vectorization

1. Example 1: Consider

$$\sum_{i=1}^{m} a_i$$

- $a_i \in \mathbb{R}$

2. Vectorization:

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} a_i \times 1 = \boldsymbol{a}^{\mathrm{T}} \cdot \mathbf{1} (= \mathbf{1}^{\mathrm{T}} \cdot \boldsymbol{a})$$

- $\boldsymbol{a} = (a_1, \ldots, a_m)^{\mathrm{T}} \in \mathbb{R}^{m \times 1}$
- $\mathbf{1} = (1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^{m \times 1}$: Vector of 1's with length m

Vectorization

1. Example 2: Consider

$$\sum_{i=1}^{m} a_i \cdot \boldsymbol{b}_i$$

   - $a_i \in \mathbb{R}$: Scalar, which may be 1
   - $\boldsymbol{b}_i \in \mathbb{R}^{k\times 1}$

2. Vectorization:

$$\sum_{i=1}^{m} a_i \cdot \boldsymbol{b}_i = \boldsymbol{B} \cdot \boldsymbol{a}$$

   - $\boldsymbol{a} = (a_1, \ldots, a_m)^{\mathrm{T}} \in \mathbb{R}^{m\times 1}$
   - $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m) \in \mathbb{R}^{k\times m}$

Vectorization

1. Example 3: Consider

$$\sum_{i=1}^{m} a_i \cdot \boldsymbol{b}_i{}^{\mathrm{T}}$$

   - $a_i \in \mathbb{R}$: Scalar, which may be 1
   - $\boldsymbol{b}_i \in \mathbb{R}^{k\times 1}$

2. Vectorization:

$$\sum_{i=1}^{m} a_i \cdot \boldsymbol{b}_i{}^{\mathrm{T}} = \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{B}$$

   - $\boldsymbol{a} = (a_1, \ldots, a_m)^{\mathrm{T}} \in \mathbb{R}^{m\times 1}$
   - $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)^{\mathrm{T}} \in \mathbb{R}^{m\times \boldsymbol{k}}$

Vectorization

1. Example 4: Consider

$$\sum_{i=1}^{m} \boldsymbol{a}_i \cdot \boldsymbol{b}_i^{\mathrm{T}}$$

- $\boldsymbol{a}_i \in \mathbb{R}^{k\times 1}$
- $\boldsymbol{b}_i \in \mathbb{R}^{l\times 1}$

2. Vectorization:

$$\sum_{i=1}^{m} \boldsymbol{a}_i \cdot \boldsymbol{b}_i^{\mathrm{T}} = \boldsymbol{A} \cdot \boldsymbol{B}$$

- $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m) \in \mathbb{R}^{k\times m}$
- $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)^{\mathrm{T}} \in \mathbb{R}^{m\times l}$

Vectorization

1. Denote $\hat{\boldsymbol{Y}} = b \cdot \mathbf{1}_n + \boldsymbol{X} \cdot \boldsymbol{w} \in \mathbb{R}^{n \times 1}$
   - $\mathbf{1}_n = (1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^{n \times 1}$: Vector of 1's with length n
2. Cost function

$$\begin{aligned}
\mathcal{J}(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{2n} (\boldsymbol{Y} - \hat{\boldsymbol{Y}})^{\mathrm{T}} \cdot (\boldsymbol{Y} - \hat{\boldsymbol{Y}}) \\
&= \frac{1}{2n} (\boldsymbol{Y} - b \cdot \mathbf{1}_n - \boldsymbol{X} \cdot \boldsymbol{w})^{\mathrm{T}} \cdot (\boldsymbol{Y} - b \cdot \mathbf{1}_n - \boldsymbol{X} \cdot \boldsymbol{w}) \\
&= \frac{1}{2n} (\boldsymbol{Y} - \tilde{\boldsymbol{X}} \cdot \boldsymbol{\theta})^{\mathrm{T}} \cdot (\boldsymbol{Y} - \tilde{\boldsymbol{X}} \cdot \boldsymbol{\theta})
\end{aligned}$$

   - $\tilde{\boldsymbol{X}} = (\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n)^{\mathrm{T}}$ (integrate b and $\boldsymbol{w}$ together)
   - $\tilde{\boldsymbol{x}}_i = (1, \boldsymbol{x}_i^{\mathrm{T}})^{\mathrm{T}}$

Vectorization

1. Consider

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

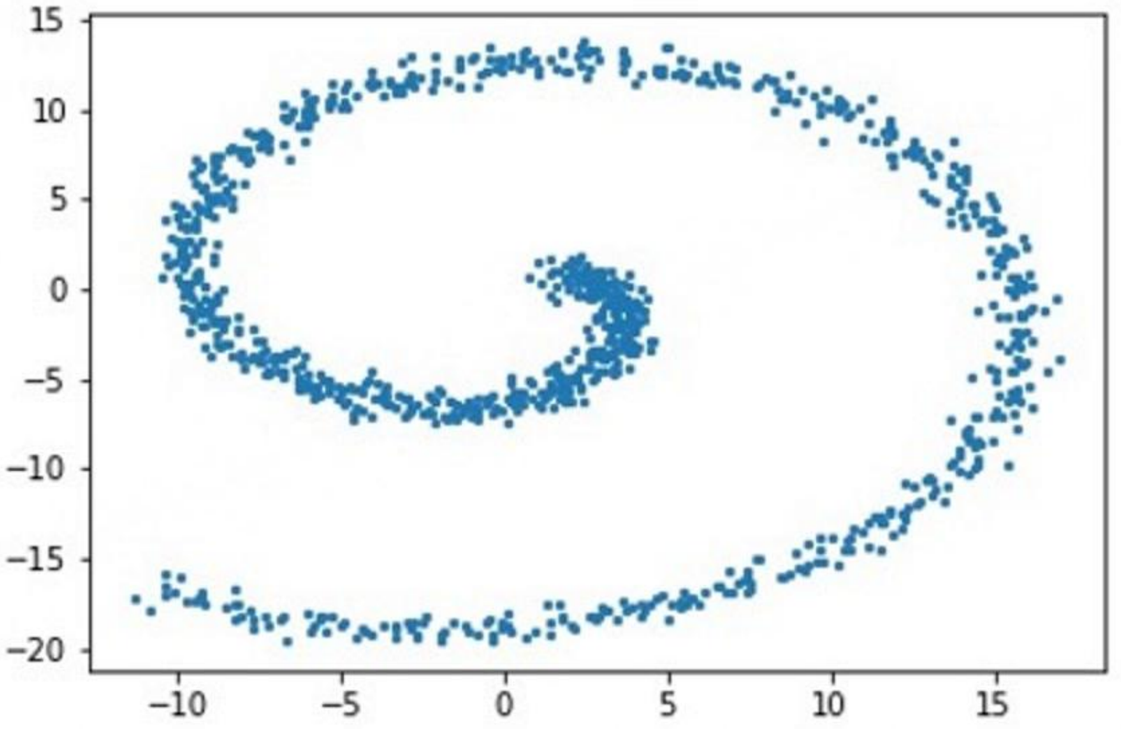
2. Obtain the normal equation

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

3. Solution (closed-form or analytical solution)

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

Example



Logistic regression -- Model setup

1. Training data
   - Feature $\boldsymbol{x}_i \in \mathbb{R}^{d\times 1}$, labels $y_i \in \{0, 1\}$ $\quad (i = 1, \ldots, n)$
2. Goal
   - Learn the relationship between feature $\boldsymbol{x}$ and label y
3. True model (used to generate data)

$$E(y_i \mid \boldsymbol{x}_i) = P(y_i = 1 \mid \boldsymbol{x}_i) = \sigma(b_0 + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w}_0) \quad (i = 1, \ldots, n)$$

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

Logistic regression -- Model setup

1. (Assumed) Proposed model (used to fit data)
   - $P(y_i = 1 \mid \boldsymbol{x}_i) = \sigma(b + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w}) \quad (i = 1, \ldots, n)$
     - $\boldsymbol{\theta} = (b, \boldsymbol{w}^{\mathrm{T}})^{\mathrm{T}}$: (Proposed) Model parameters
     - By fitting a model, we mean to estimate parameters of the proposed model
     - Have the same form with the true model, but we need to estimate parameters
2. Question: What if we use a linear regression model to analyze binary classification problems?

Logistic regression -- Parameter estimation

1. Loss function: For the ith 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)
  - $z_i = b + \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w}$

- a_i is a function of $\boldsymbol{\theta} = (b, \boldsymbol{w}^{\mathrm{T}})^{\mathrm{T}}$
- Cross entropy is also a function of $\boldsymbol{\theta}$
- However, in the above expressions, we omit its argument for simplicity
- The above cross entropy is a convex function of the model parameter $\boldsymbol{\theta}$

Logistic regression -- Parameter estimation

1. Cost function

$$\mathcal{J}(\boldsymbol{\theta}) = n^{-1} \sum_{i=1}^{n} \mathcal{L}\{y_i, a_i\}$$

$$= -n^{-1} \sum_{i=1}^{n} \{y_i \log a_i + (1 - y_i) \log\{1 - a_i\}\}$$

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

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

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

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

Step 3. Update parameter

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

Step 4. Go back to Step 2 until convergence, or reaching maximum number of iterations

Remarks

1. Cost function $\mathcal{J}$ is univariate
2. In this course, we implicitly assume that the dimension of $\partial\mathcal{J}/\partial\boldsymbol{\theta}$ is the same as $\boldsymbol{\theta}$
3. That is, if $\boldsymbol{\theta}$ is a column vector, so is $\partial\mathcal{J}/\partial\boldsymbol{\theta}$
4. Then, the third step of Newton-Raphson algorithm is mathematically rigorous
5. In other textbooks, $\partial\mathcal{J}/\partial\boldsymbol{\theta}$ is defined as a row vector, even if $\boldsymbol{\theta}$ is a column vector
6. We don't adopt that convention in this course

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

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

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

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

Step 3. Update parameter

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

Step 4. Go back to Step 2 until convergence, or reaching maximum number of iterations

(Batch) Gradient descent algorithm

1. α : Learning rate

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

Comparison of algorithms

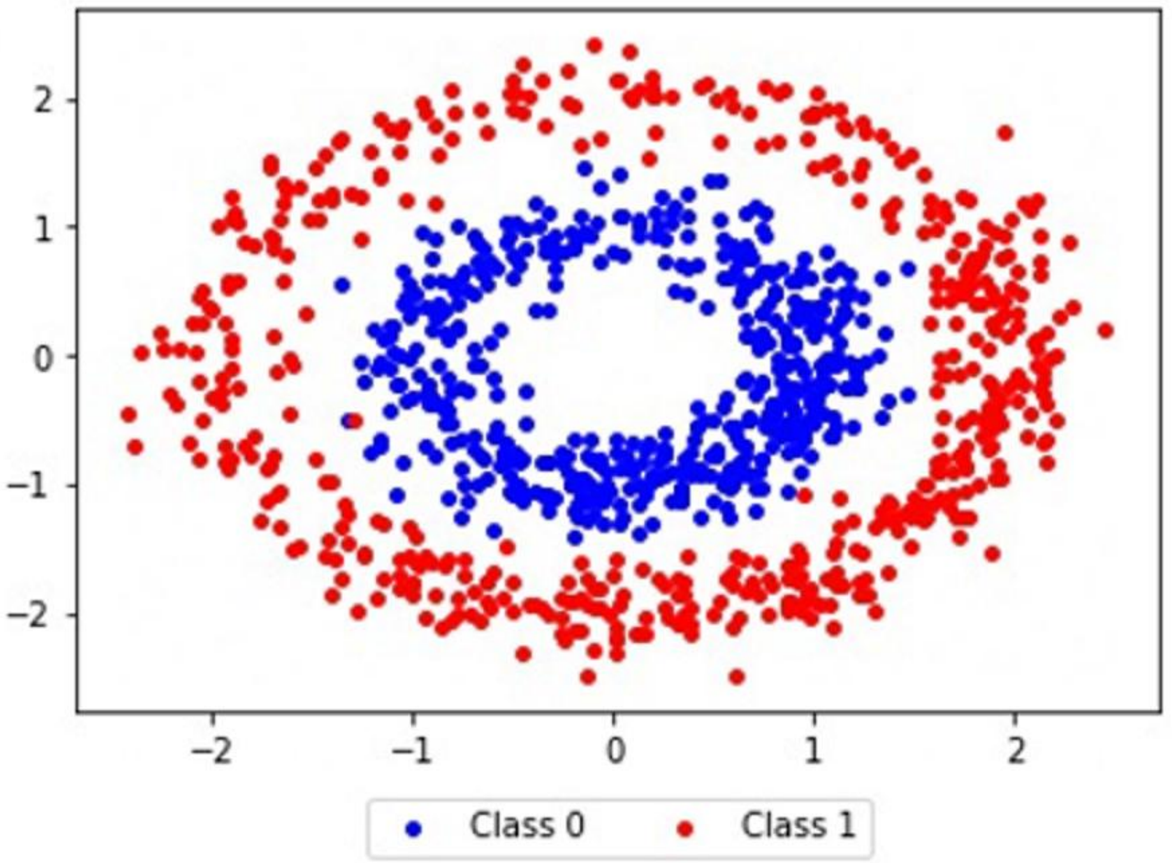
1. Newton-Raphson algorithm
   - Advantages:
     - Fast convergence rate
     - High accuracy
   - Disadvantages:
     - Low computation efficiency due to the Hessian matrix
     - Low stability if the Hessian matrix is ill-conditioned

Comparison of algorithms (Cont'd)

1. (Batch) Gradient descent algorithm

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

Example



Summary

1. The following shows typical steps for data modeling

Problem	Proposed model	Cost function	Algorithm
$y \in \mathbb{R}$	$E(y \mid \boldsymbol{x}) = b + \boldsymbol{x}^{\mathrm{T}} \cdot \boldsymbol{w}$	$n^{-1} \sum_{i=1}^{n} (y_i - b - \boldsymbol{x}_i^{\mathrm{T}} \cdot \boldsymbol{w})^2$	Normal equation
$y \in \{0, 1\}$	$E(y \mid \boldsymbol{x}) = \sigma(b + \boldsymbol{x}^{\mathrm{T}} \cdot \boldsymbol{w})$	$-n^{-1} \sum_{i=1}^{n} \{y_i \log a_i + (1 - y_i) \log(1 - a_i)\}$	Gradient descent

2. Gradient descent algorithms are commonly used for AI models

3. The essential step for gradient descent algorithms is to obtain

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

Problems

1. Up to now, we have assumed the same form for the true model and the proposed model
2. However, it is commonly not the case in practice
3. Model misspecification
   - The proposed (statistical) models are too simple
   - True model, however, is COMPLEX

Learning goals

1. Fully connected neural network (multiple layer perceptron)
2. Convolutional Neural Network (CNN, ...)
3. Sequential modeling (RNN, LSTM, transformers, ...)
4. (If we have time,) More topics (GNN, GCN, ...)

FNN Example

Test image

Model (FNN with 2 hidden layers)

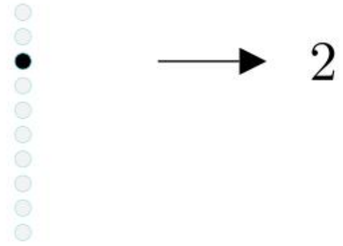
Input layer
$d^{[0]} = 784$

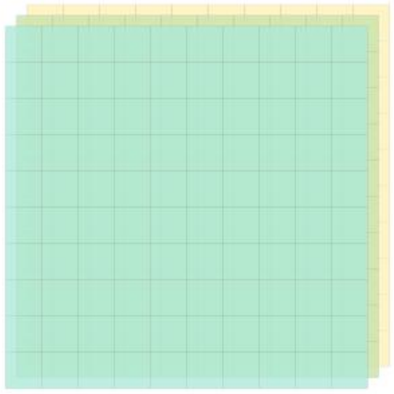
1st hidden layer
$d^{[1]} = 128$

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

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

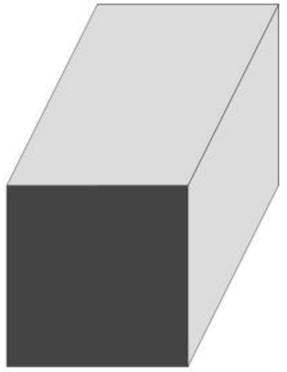
Estimated result





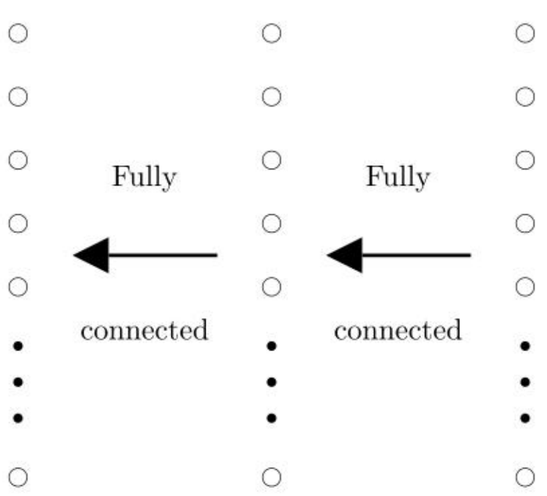
Convolution

Pooling

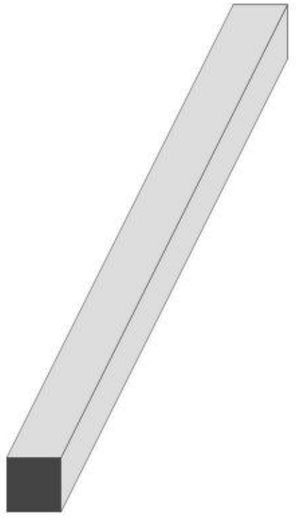


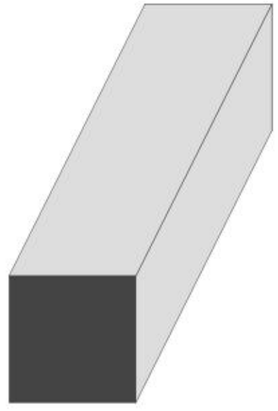
Input $(\mathbb{R}^{d_C \times d_H \times d_W})$

To be discussed



$\mathcal{J}$





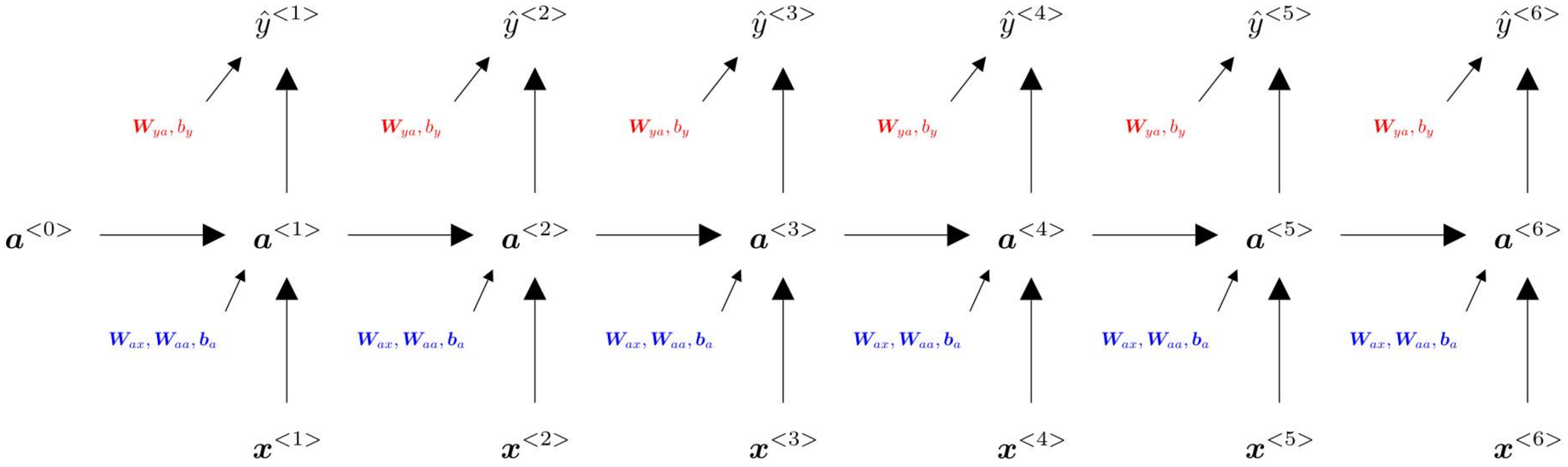
$\boldsymbol{A}^{[L-1]} \in \mathbb{R}^{d^{[L-1]} \times 1 \times 1}$

$\boldsymbol{A}^{[L-2]} \in \mathbb{R}^{d_C^{[L-2]} \times d_W^{[L-2]} \times d_H^{[L-2]}}$

$\boldsymbol{A}^{[L]} \in \mathbb{R}^{d^{[L]} \times 1 \times 1}$

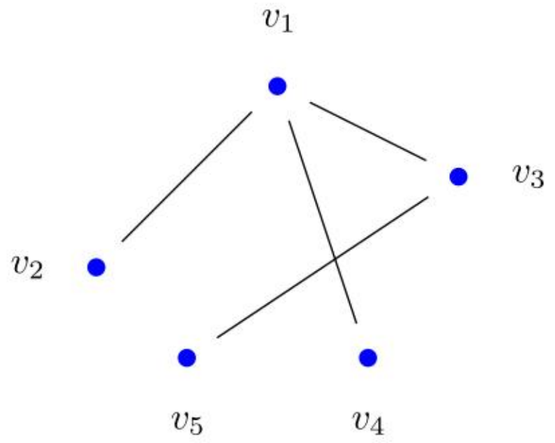
Building block for RNN

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



$$\boldsymbol{a}^{<i>} = \sigma_{ax}(\boldsymbol{W}_{ax} \cdot \boldsymbol{x}^{<i>} + \boldsymbol{W}_{aa} \cdot \boldsymbol{a}^{<i-1>} + \boldsymbol{b}_a) \quad (i = 1, \ldots, 6)$$

Undirected graph



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

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)

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

Review

1. $X \sim P$: Random variable X is generated from a distribution P
2. $E_{X \sim P}\{f(X)\}$: Expectation of $f(X)$
   - We use "$X \sim P$" to highlight the true distribution of X
   - $f(x)$: Measurable function
3. In practice, we generally cannot observe the distribution P
4. Instead, we may only observe a random sample $\{x_i : i = 1, \ldots, n\}$ generated from P

Review

1. Based on $\{x_i : i = 1, \ldots, n\}$, its empirical distribution P_n is

Sample	x_1	x_2	x_3	$\cdots$	x_n
Probability	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\cdots$	$\frac{1}{n}$

2. Remarks
   - Since $\{x_i : i = 1, \ldots, n\}$ is random, the empirical distribution is also random
   - The density of the empirical distribution (with respect to a counting measure) is

     $p_n(x) = n^{-1} \sum_{i=1}^{n} \delta(x = x_i)$

     - $\delta(x = y)$: indicator function, and it is 1 if $x = y$, and it is 0 otherwise
   - It can be easily shown that the sample mean is $E_{X \sim P_n}(X) = n^{-1} \sum_{i=1}^{n} x_i$

Cross entropy

1. A fact: under regularity conditions,
$$E_{X\sim P}\{\log p(X)\} \geq E_{X\sim P}\{\log q(X)\}$$
   - $p(x)$: Density function of the distribution P
   - $q(x)$: Another density function of a certain distribution
2. (Information) Entropy comes from the information theory
   - "Information content" of an event: Negative logarithm of its probability of occurrence
   - "(Information) Entropy": Average of information content
3. 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 $\{x_1, \ldots, x_n\}$
2. Under generality conditions, cross entropy $H(p, q)$ can be approximated by (law of large numbers)

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

   - Equivalently, it is the cross entropy between the empirical density $P_n(x) = n^{-1} \sum_{i=1}^{n} \delta(x = x_i)$ and $q(x)$

3. Goal: Find a density function $q(x)$ to minimize $\hat{H}(p, q)$
   - That is, to approximate the empirical distribution of the random sample well

Revisit logistic regression

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

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

- $p(\boldsymbol{x})$: Unspecified marginal density for $\boldsymbol{x}$
- $p(y \mid \boldsymbol{x}) = \sigma(b_0 + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{w}_0)$: Parametric conditional density for y

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

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

Revisit logistic regression

1. A little more math

$$-\frac{1}{n}\sum_{i=1}^{n}\log q(\boldsymbol{x}_i, y_i) = -\frac{1}{n}\sum_{i=1}^{n}\{\log q(\boldsymbol{x}_i) + \log q(y_i \mid \boldsymbol{x}_i)\}$$

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

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

Revisit logistic regression

1. Thus,

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

- 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

Summary

1. Reviewed linear and logistic regressions
   - Goal: A unified procedure to model data
     - Data $\rightarrow$ (Proposed) Model $\rightarrow$ Cost (Loss) function $\rightarrow$ Optimization
2. Vectorization to improve computation efficiency
3. Two optimization algorithms: Newton-Raphson vs gradient descent
4. Drawbacks of simple models: Model misspecification
5. Cross entropy for classification problems