Logistic Regression and Softmax Regression

Prof. Mingkui Tan

SCUT Machine Intelligence Laboratory (SMIL)

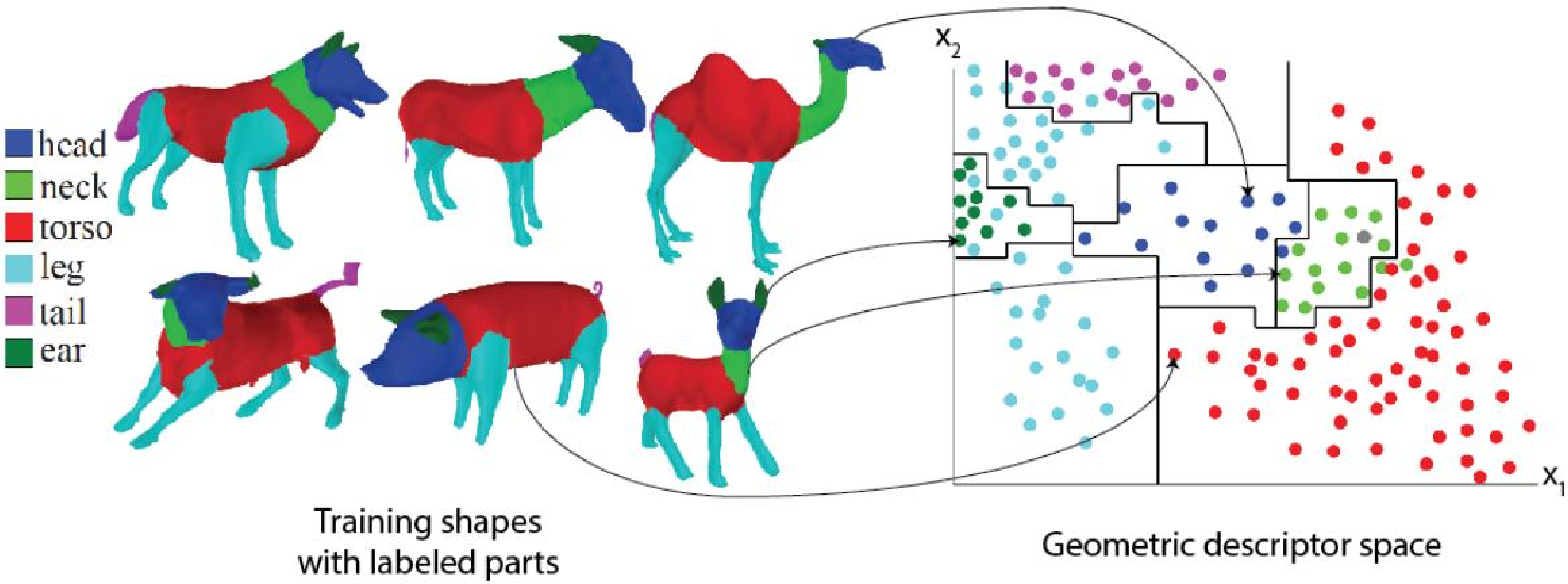
Contents

Principle Goal of Machine Learning

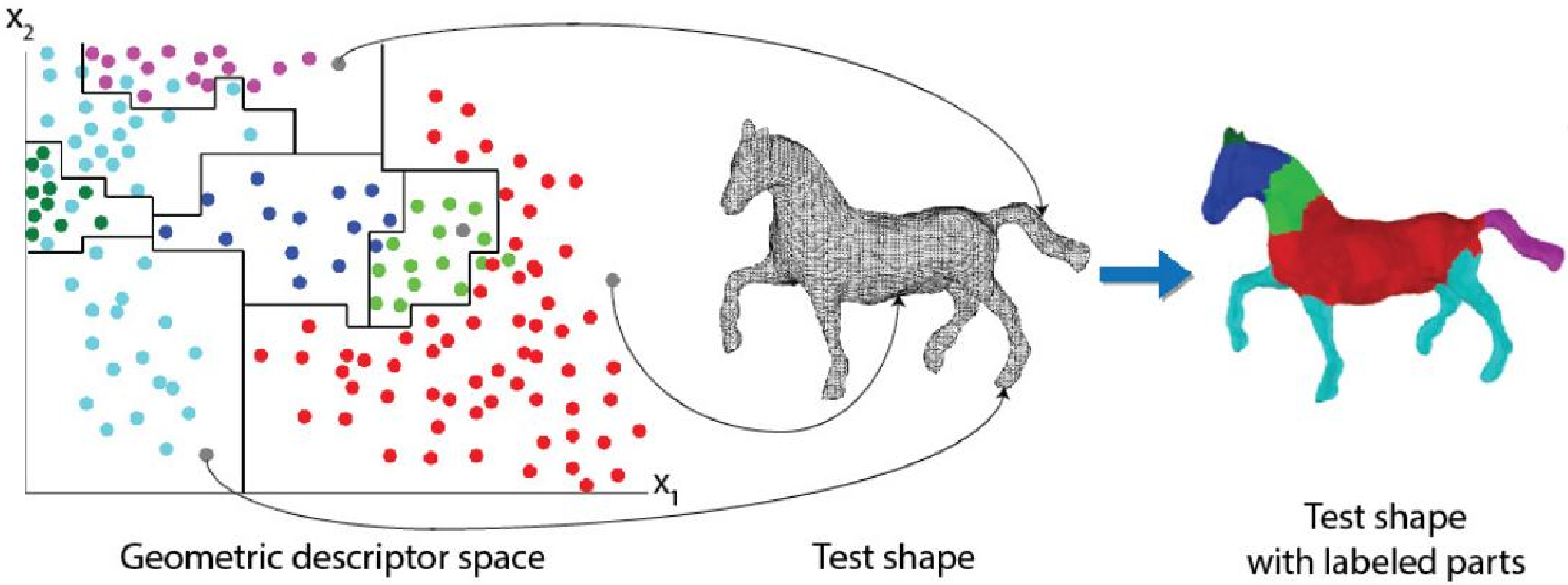
Prediction:

Making Decisions:

Example: part labeling [**training stage**]



Example: part labeling [**training stage**]

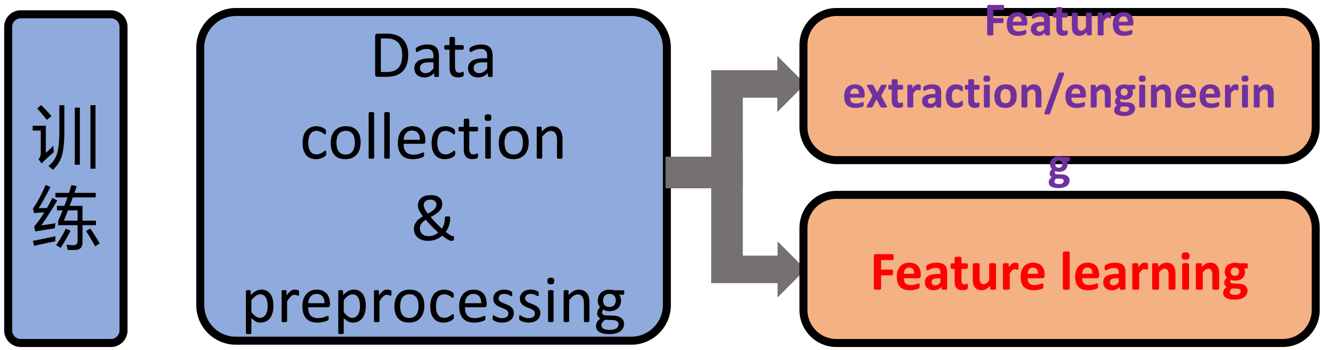


Pipeline of data-driven methods

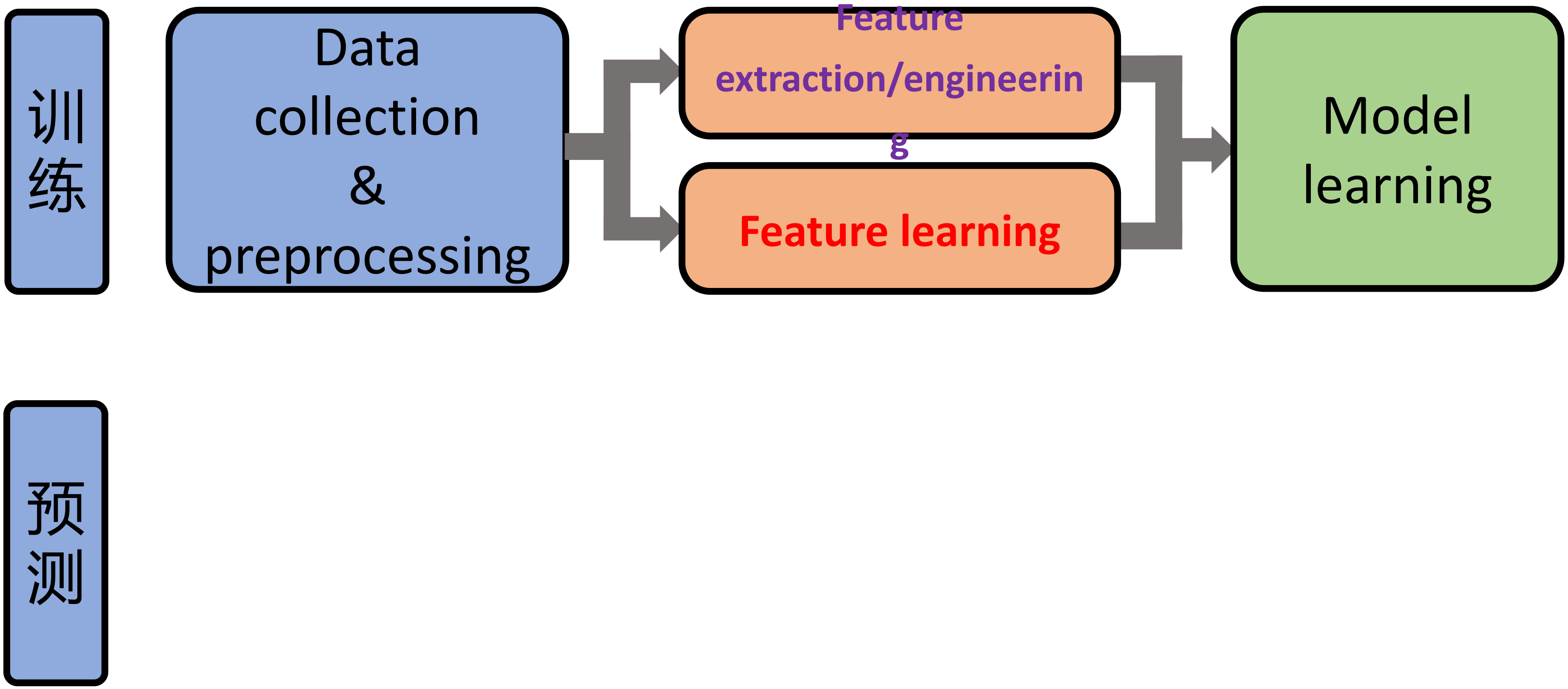
训练

Data collection
& preprocessing

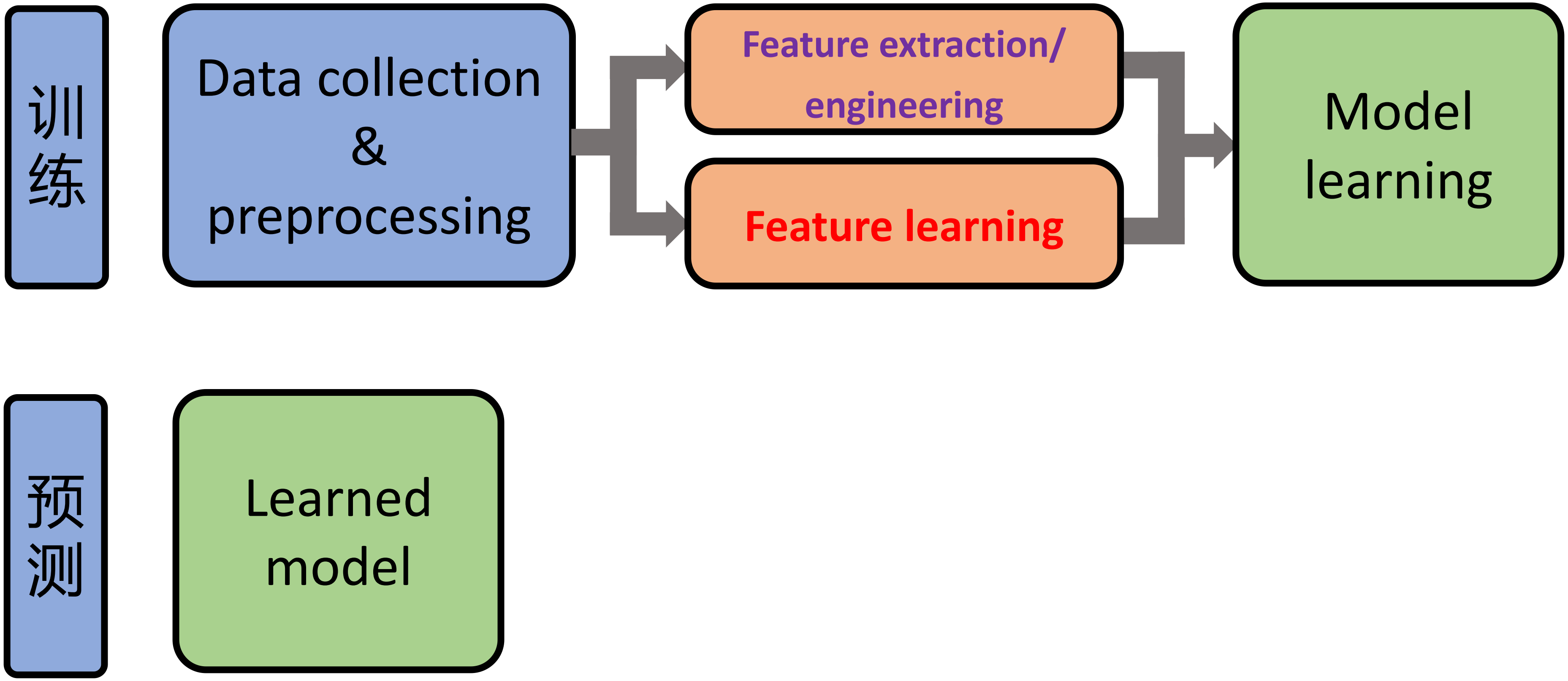
Pipeline of data-driven methods



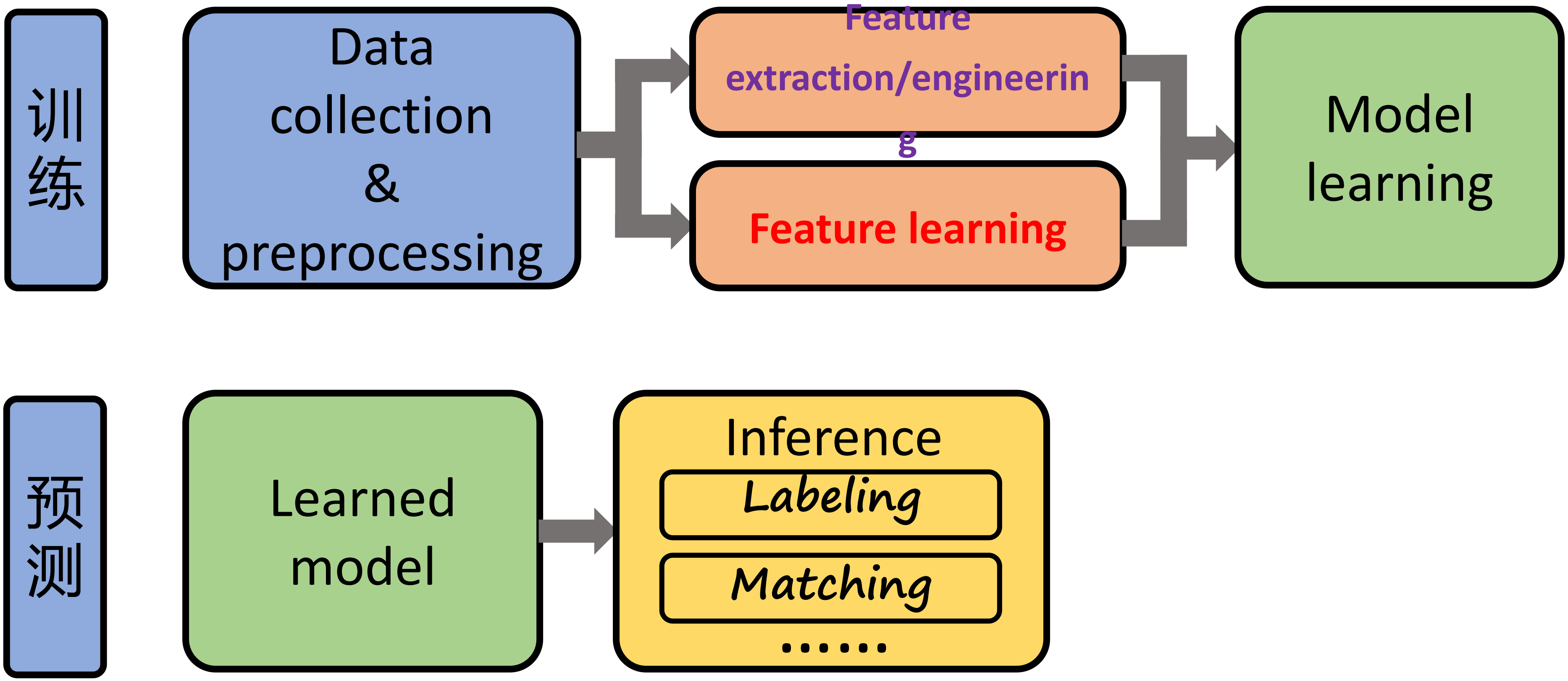
Pipeline of data-driven methods



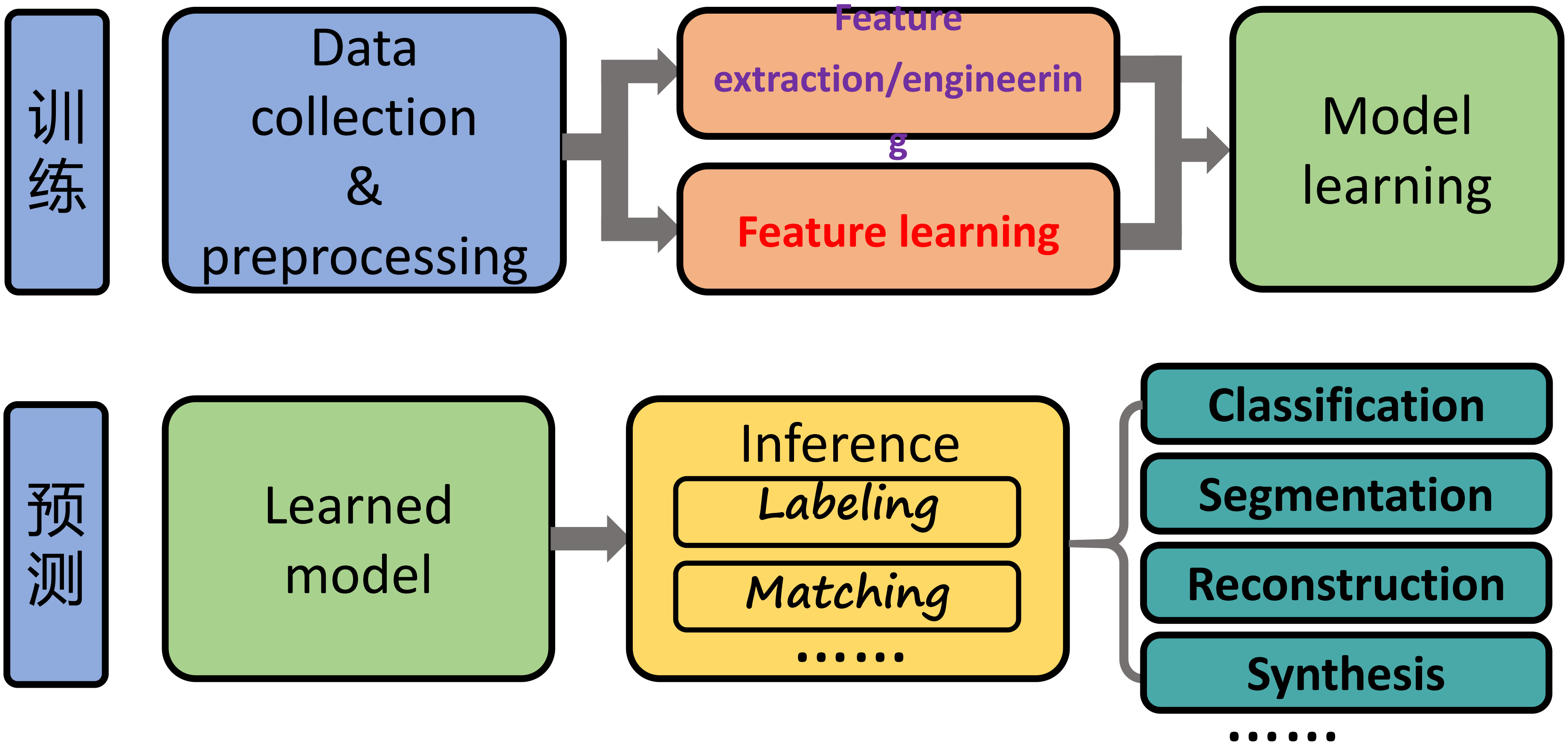
Pipeline of data-driven methods



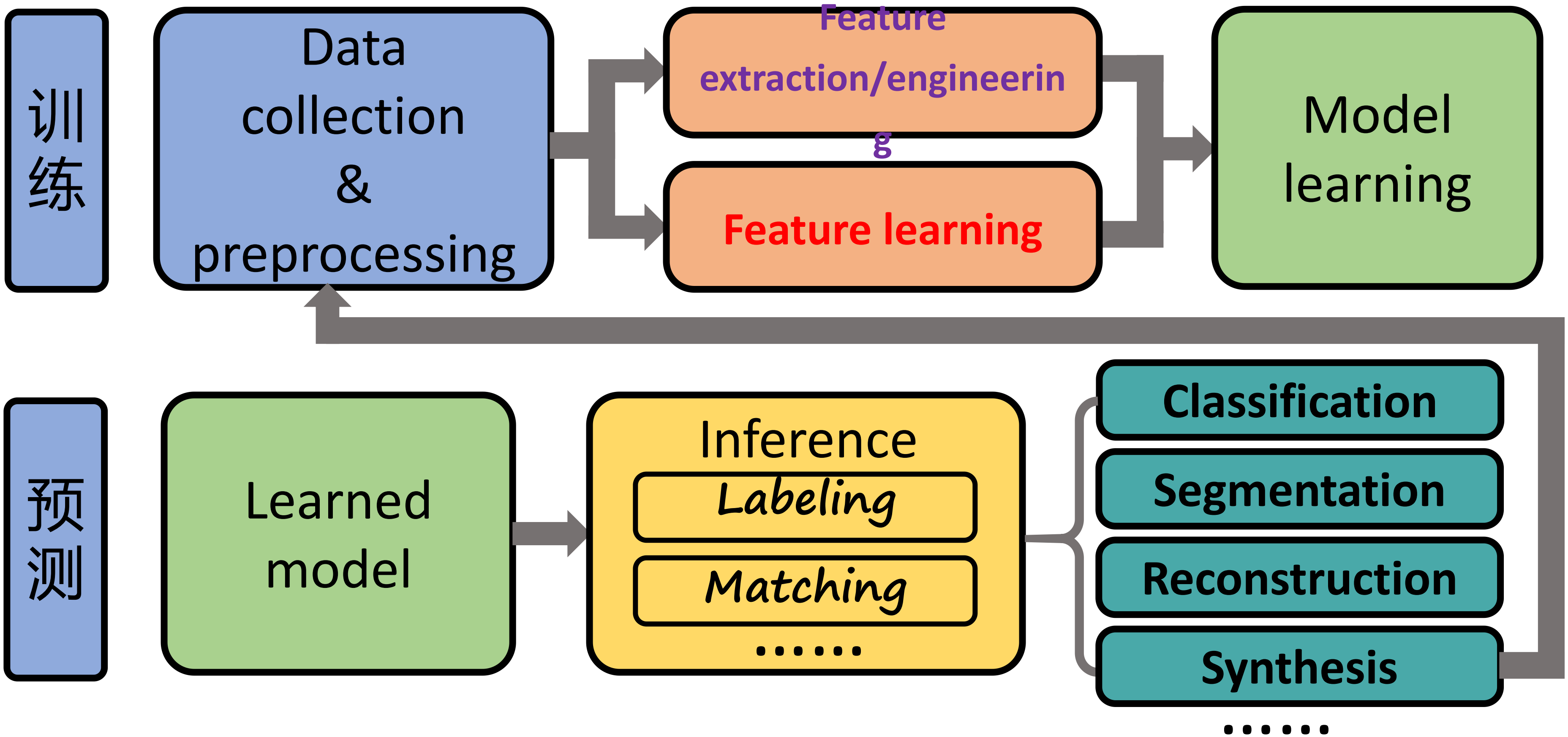
Pipeline of data-driven methods



Pipeline of data-driven methods



Pipeline of data-driven methods



Contents

Contents

Data Example

Dataset: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- $\mathbf{x}_i \leftarrow$ health information
- $y_i = \pm 1 \leftarrow$ did he have a heart attack or not

- Given the health information of one person:

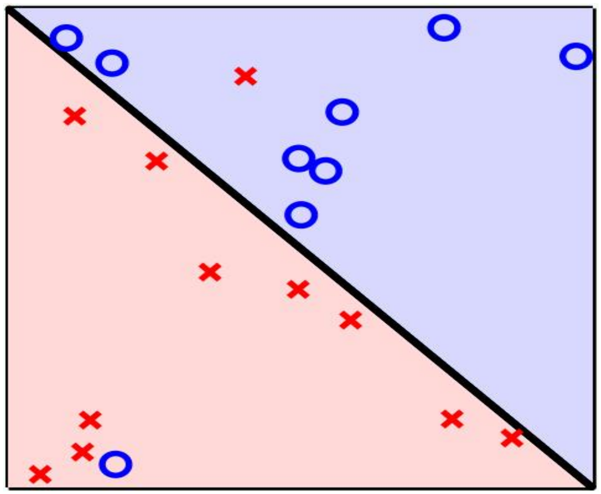
age	62 years
gender	male
blood sugar	120 mg/dL40,000
HDL	50
LDL	120
Mass	190 lbs
Height	$5'\ 10''$
...	...

How to infer the **probability of heart attack?**

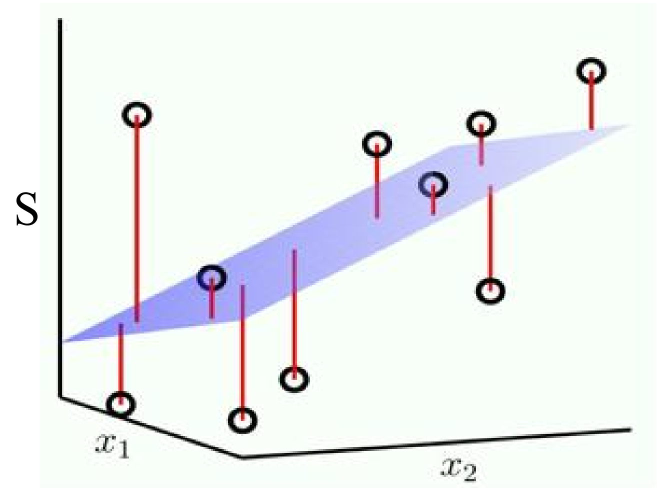
Linear Classification and Regression

The linear signal:

$$z = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$



Linear Classification



Linear Regression

Probability Function

- To infer the probability of heart attack $P[y =+ 1|\mathbf{x}]$, the probability function of logistic function is as follows:

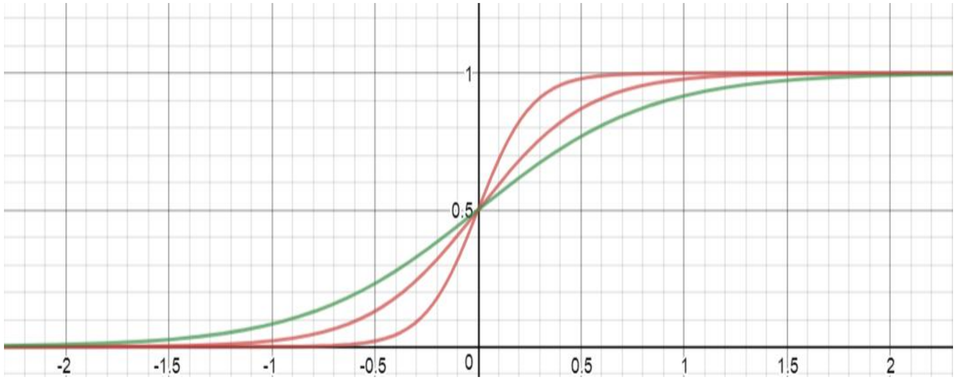
$$h_{\mathbf{w}}(\mathbf{x}) = g(z) = g\left(\sum_{i=1}^{m} w_i x_i\right) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Here,$z = \mathbf{w}^{\mathrm{T}}\mathbf{x}$, $g(\cdot)$ is a logistic function:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Properties of Logistic Function

$$g(z) = \frac{1}{1+e^{-z}}$$



- The function is a continuous function

- If $z \to +\infty$, then $g(z) \to 1$; if $z \to -\infty$, then $g(z) \to 0$

$$g(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{1+e^{z}}$$

$$g(-z) = \frac{1}{1+e^{z}} = 1 - g(z)$$

How to Learn $\mathbf{w}$?

- Intuitively, similar to SVM, we need to define a Loss Function to find a good $h_{\mathbf{w}}(\mathbf{x})$ so that it fits the following targets well:

$$h_{\mathbf{w}}(\mathbf{x}) \text{ is good if}: \begin{cases} h_{\mathbf{w}}(\mathbf{x}) \approx 1, & y = 1 \\ h_{\mathbf{w}}(\mathbf{x}) \approx 0, & y = -1 \end{cases}$$

- Can we use the least square loss below?

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\mathbf{w}}(\mathbf{x}_i) - \frac{1}{2}(1 + y_i)\right)^2$$

Questions: Why the least square loss is in this way?

- Can we use this loss? The answer is Negative! Why?
- Probabilily $h_{\mathbf{w}}(\mathbf{x})$=1.001 which is better than $h_{\mathbf{w}}(\mathbf{x})$=0.9
- But $h_{\mathbf{w}}(\mathbf{x})$ denotes the probability, thus $h_{\mathbf{w}}(\mathbf{x})$ must satisfy:

$$h_{\mathbf{w}}(\mathbf{x}) \leq 1 .$$

How to Learn $\mathbf{w}$?

- We need to define a Loss Function to find a good $h_{\mathbf{w}}(\mathbf{x})$ so that it fits the following targets well:

$$h_{\mathbf{w}}(\mathbf{x}) \text{ is good if}: \begin{cases} h_{\mathbf{w}}(\mathbf{x}) \approx 1, & y = 1 \\ h_{\mathbf{w}}(\mathbf{x}) \approx 0, & y = -1 \end{cases}$$

- The least square loss is no longer good.

- Here, we introduce a **new loss** called logistic loss as below:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i})$$

$$\boxed{\textbf{Why the logistic loss is in this form?}}$$

Probabilistic View of Training Samples

- Recall $h_{\mathbf{w}}(\mathbf{x})$ is a probability function to predict the probability of an instance $\mathbf{x}$ being to the label $y_i \in \{-1,1\}$ as below:

$$P(y|\mathbf{x}) = \begin{cases} g(\mathbf{w}^{\mathrm{T}}\mathbf{x}), & y = 1 \\ 1 - g(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = g(-\mathbf{w}^{\mathrm{T}}\mathbf{x}), & y = -1 \end{cases}$$

- The training sample $(\mathbf{x}_i, y_i)$ can be considered as **random variables** sampled from a sample space $\{\mathcal{X}, \mathcal{Y}\}$.
- The instance $\mathbf{x}_i$ and its label y_i follow a **conditional probability**:

$$P(y_i|\mathbf{x}_i) = g(y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)$$

The label y_i is definitely determined by the observation $\mathbf{x}_i$, namely y_i is condition on $\mathbf{x}_i$

How to Learn **w**?

Recall that the training samples $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_n, y_n)$ can be considered as random variables following the a conditional probability as below:

$$P(y_i|\mathbf{x}_i) = g(y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i)$$

Likelihood of training examples:

Assume that $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_n, y_n)$ are independently sampled, the joint distribution (or likelihood) $P(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_n, y_n)$ satisfies:

$$\boxed{P(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)}$$

How to Learn w?

Note the parameter **w** determines the distribution
$$P(y_i|\mathbf{x}_i) = g(y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)$$

- Given the likelihood $P(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)$, we can estimate **w** with Maximum Likelihood Estimation (MLE)
- What is Maximum Likelihood Estimation?

Definition: Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a statistical method used to make inferences about parameters of the underlying probability distribution of a given data set.

How to estimate parameter **w** in $h_{\mathbf{w}}(\mathbf{x})$ with MLE?

How to Learn **w**?

Estimate **w** by maximizing the likelihood

$$\max_{\mathbf{w}} P(y_1, \dots, y_n | \mathbf{x}_1, \dots, \mathbf{x}_n) = \prod_{i=1}^{n} P(y_i | \mathbf{x}_i)$$

$$\max \prod_{i=1}^{n} P(y_i | \mathbf{x}_i) \Leftrightarrow \max \log\left(\prod_{i=1}^{n} P(y_i | \mathbf{x}_i) \right)$$

$$\equiv \max \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i)$$

$$\Leftrightarrow \min -\frac{1}{n} \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i)$$

How to Learn $\mathbf{w}$?

Estimate $\mathbf{w}$ by maximizing the likelihood $\max_{\mathbf{w}} \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)$

$$\max \prod_{i=1}^{n} P(y_i|\mathrm{x}_i) \Leftrightarrow \max \log(\prod_{i=1}^{n} P(y_i|\mathrm{x}_i))$$

$$\Leftrightarrow \min -\frac{1}{n}\sum_{i=1}^{n} \log P(y_i|\mathrm{x}_i)$$

$$\Leftrightarrow \min \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{P(y_i|\mathrm{x}_i)} \longleftarrow \boxed{P(y_i|x_i) = g(y_i \mathrm{w}^T \mathrm{x}_i)}$$

$$\equiv \min \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{g(y_i \mathrm{w}^T \mathrm{x}_i)} \longleftarrow \boxed{g(z) = \frac{1}{1+e^{-z}}}$$

$$\equiv \min \frac{1}{n}\sum_{i=1}^{n} \log(1+e^{-y_i \mathrm{w}^T \mathrm{x}_i}) \equiv \min \mathcal{L}(\mathbf{w})$$

Definition: Logistic regression

$$\max_{\mathbf{w}} \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) = \boxed{\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \log(1+e^{-y_i \cdot \mathbf{w}^{\mathrm{T}} \mathbf{x}_i})}$$

Regularization Required

Similar to SVM, we employ Regularization to avoid overfitting issue

- We have the following objective function for logistic regression:

$$J(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

Here, $\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_i})$ is called **Logistic Loss** and λ is the regularization parameter.

Why need regularization?

- "Simple" model
- Less prone to overfitting

SVM vs Logistic Regression

- SVM:

$$\min \ J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \max(0,\ 1 - y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

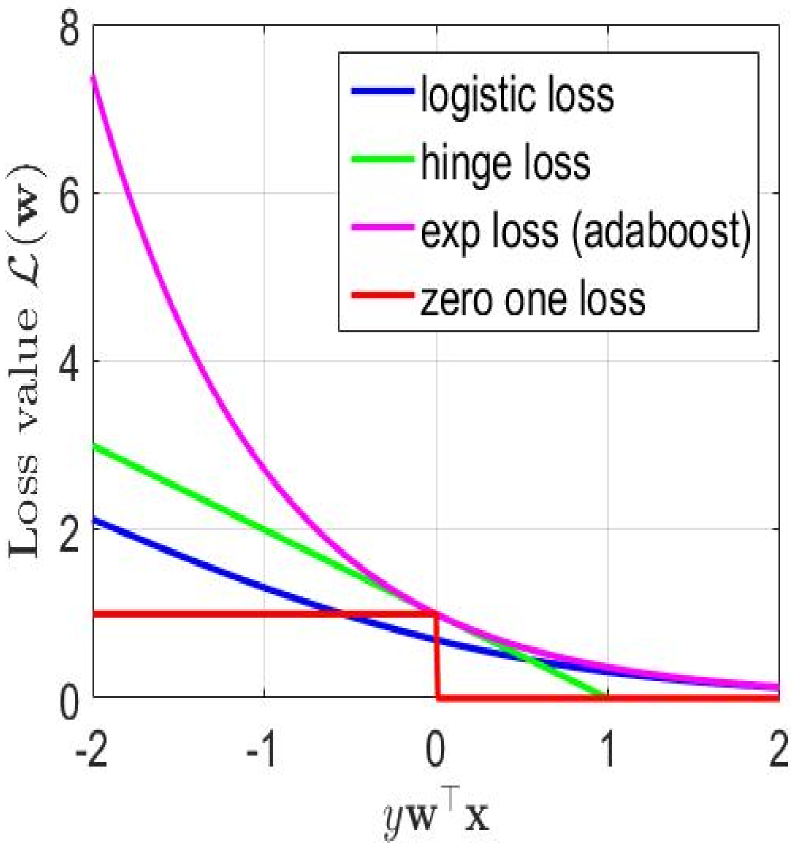
- logistic regression:

$$\min J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

- The regularization term $||\mathbf{w}||_2^2$ is called L_2^2 regularizer

- Connections to SVM:
  - Both are supervised algorithms
  - Both are used to solve binary classification problem

Graphical Comparison of Loss Functions



$z = y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i$

Comparison of Different Loss Functions

logistic loss:

$$\mathcal{L}(\mathbf{x}_i;\mathbf{w}) = \log(1 + e^{-y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i})$$

hinge loss:

$$\mathcal{L}(\mathbf{x}_i;\mathbf{w}) = \max(0,\ 1 - y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)$$

exponential loss (for adaboost):

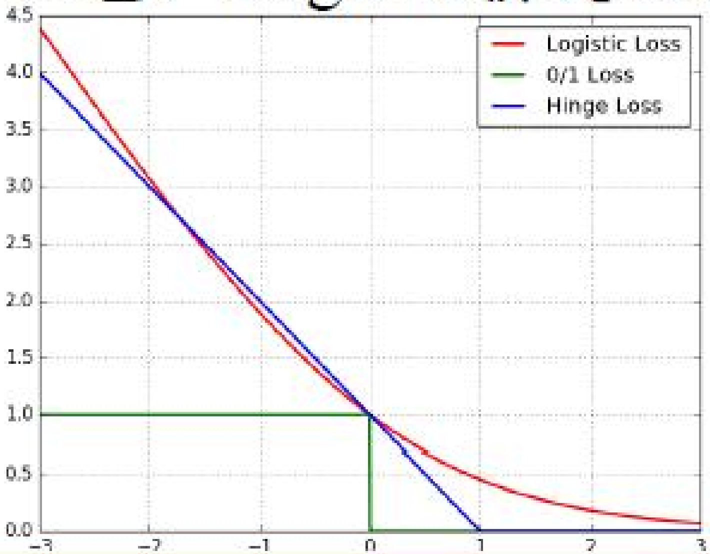
$$\mathcal{L}(\mathbf{x}_i;\mathbf{w}) = e^{-y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i}$$

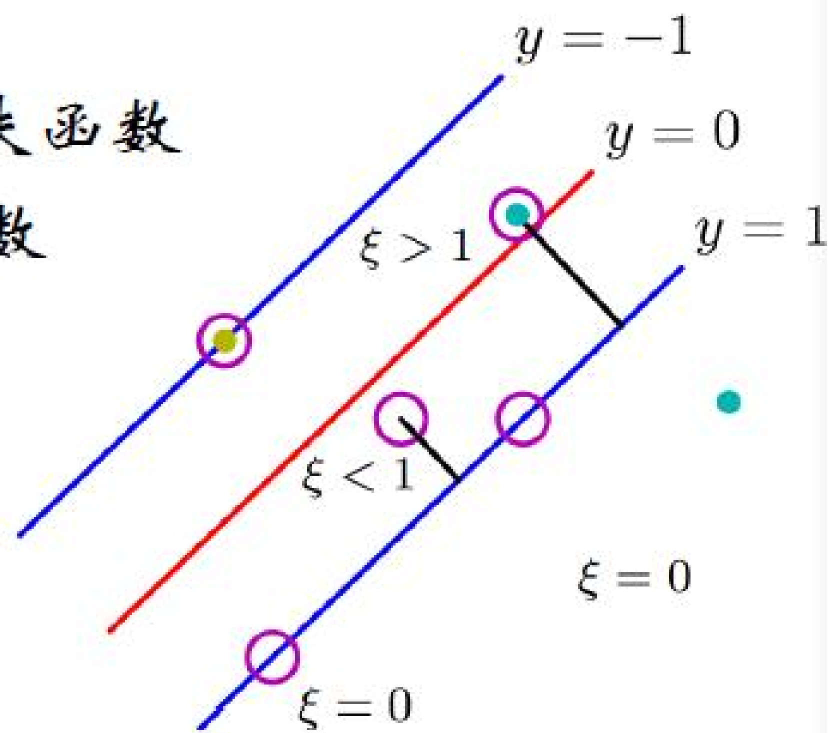
zero one loss:

$$\mathcal{L}(\mathbf{x}_i;\mathbf{w}) = \begin{cases} 0, & y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0 \\ 1, & y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i \le 0 \end{cases}$$

Graphical Comparison of Three Loss Functions

- 绿色：0/1损失
- 蓝色： SVM Hinge损失函数
- 红色：Logistic损失函数





How to Learn $\mathbf{w}$?

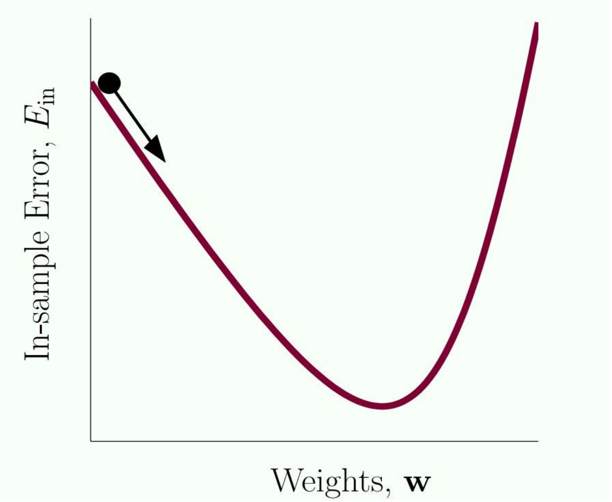
Minimize $J(\mathbf{w})$ by (Stochastic) Gradient Descent: $\min_{\mathbf{w}} J(\mathbf{w})$

- Compute gradient $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$ of $J(\mathbf{w})$ with respect to $\mathbf{w}$:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\frac{1}{n}\sum_{i=1}^{n} \frac{y_i \mathbf{x}_i e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}}{1+e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}} + \lambda \mathbf{w}$$

- Update parameters **with learning rate η**

$$\mathbf{w} := \mathbf{w} - \eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$



Note:
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial(\log(1+e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}))}{\partial \mathbf{w}} + \lambda \mathbf{w} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{1+e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}} \cdot \frac{\partial(e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i})}{\partial \mathbf{w}} + \lambda \mathbf{w}$$
$$= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{1+e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}} \cdot e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i} \cdot (-y_i \mathbf{x}_i) + \lambda \mathbf{w} = -\frac{1}{n}\sum_{i=1}^{n} \frac{y_i \mathbf{x}_i e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}}{1+e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}} + \lambda \mathbf{w}$$

Logistic Regression for $y_i \in \{0,1\}$

Previous study considers $y_i \in \{-1, +1\}$, but what if $y_i \in \{0,1\}$ and what if $y_i \in \{0,\ 1,\ \ldots,\ K-1\}$?

- Let us first consider the simple case: $y_i \in \{0,1\}$
- Similar to the case $y_i \in \{-1,1\}$, we define the probability of $\mathbf{x}_i$ being with the label $y_i \in \{0,1\}$ as follows:

$$P(y_i|\mathbf{x}_i) = \begin{cases} g(\mathbf{w}^{\mathrm{T}}\mathbf{x}), & y = 1 \\ 1 - g(\mathbf{w}^{\mathrm{T}}\mathbf{x}), & y = 0 \end{cases}$$

where $g(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1+e^{-W^T X}}$

- More specifically, the instance $\mathbf{x}_i$ and its label y_i follow the conditional probability as below:

$$P(y_i|\mathbf{x}_i) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x})^{y_i} \cdot \left(1 - g(\mathbf{w}^{\mathrm{T}}\mathbf{x})\right)^{1-y_i}$$

Again, Resort to Maximum Likelihood Estimation

Note the parameter **w** determines the distribution

$$P(y_i|\mathbf{x}_i) = h_{\mathbf{w}}(\mathbf{x}_i)^{y_i} \cdot \left(1 - h_{\mathbf{w}}(\mathbf{x}_i)\right)^{1-y_i}$$

Likelihood of training examples:

Assuming that $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are independently sampled, the joint distribution (or likelihood) $P(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ satisfies $P(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)$

- We can estimate **w** with Maximum Likelihood Estimation (MLE)

Similar to $y_i \in \{-1,1\}$, we maximize the likelihood to estimate **w**

$$\max_{\mathbf{w}} P(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)$$

How to Learn **w**?

Similar to $y_i \in \{-1,1\}$, we maximize the likelihood to estimate $\mathbf{w}$

$$\max_{\mathbf{w}} P(y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)$$

$$\max \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) \Leftrightarrow \max \log\left(\prod_{i=1}^{n} P(y_i|\mathbf{x}_i)\right)$$

$$\equiv \max \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i)$$

$$\Leftrightarrow \min -\frac{1}{n}\sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i)$$

How to Learn **w**?

Estimate **w** by maximizing the likelihood $\max_{\mathbf{w}} \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)$

$$\max \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) \Leftrightarrow \max \log\left(\prod_{i=1}^{n} P(y_i|\mathbf{x}_i)\right)$$

$$\equiv \min -\frac{1}{n}\sum_{i=1}^{n} \log\left(h_{\mathbf{w}}(\mathbf{x}_i)^{y_i} \cdot \left(1 - h_{\mathbf{w}}(\mathbf{x}_i)\right)^{1-y_i}\right)$$

$$\equiv \min -\frac{1}{n}\sum_{i=1}^{n} (y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1 - y_i)\log(1 - h_{\mathbf{w}}(\mathbf{x}_i)))$$

$$\equiv \min \mathcal{L}(\mathbf{w})$$

Regularization Required

We employ Regularization to avoid overfitting issue

- We have the following objective function for logistic regression:

$$J(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

Now, the **logistic loss** becomes

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1-y_i)\log(1-h_{\mathbf{w}}(\mathbf{x}_i))\right)$$

Minimize $J(\mathbf{w})$ by (Stochastic) Gradient Descent: $\min_{\mathbf{w}} J(\mathbf{w})$

How to Learn $\mathbf{w}$?

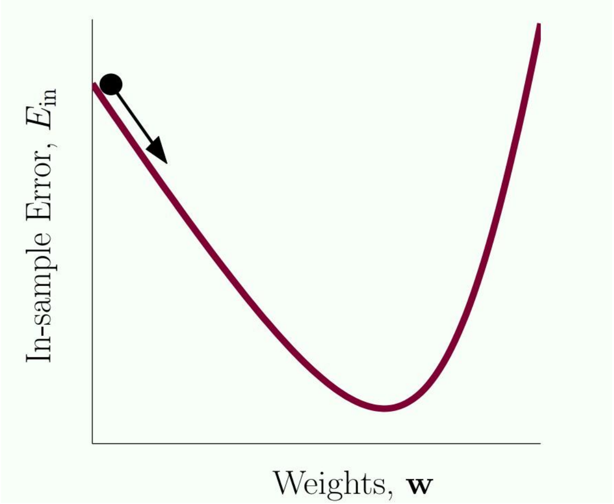
Minimize $J(\mathbf{w})$ by (Stochastic) Gradient Descent: $\min_{\mathbf{w}} J(\mathbf{w})$

- Compute gradient $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$ of $J(\mathbf{w})$ with respect to $\mathbf{w}$:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i + \lambda\mathbf{w}$$

- Update parameters with **learning rate η**

$$\mathbf{w} := \mathbf{w} - \eta\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$



Details of Calculate $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$

Note:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n}\left(-y_i \cdot \frac{1}{h_{\mathbf{w}}(\mathbf{x}_i)} \cdot \frac{\partial h_{\mathbf{w}}(\mathbf{x}_i)}{\partial \mathbf{w}} + (1-y_i) \cdot \frac{1}{1-h_{\mathbf{w}}(\mathbf{x}_i)} \frac{\partial h_{\mathbf{w}}(\mathbf{x}_i)}{\partial \mathbf{w}}\right) + \lambda \mathbf{w}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(-y_i \cdot \frac{1}{h_{\mathbf{w}}(\mathbf{x}_i)} \cdot \frac{\partial h_{\mathbf{w}}(\mathbf{x}_i)}{\partial \mathbf{w}} + (1-y_i) \cdot \frac{1}{1-h_{\mathbf{w}}(\mathbf{x}_i)} \frac{\partial h_{\mathbf{w}}(\mathbf{x}_i)}{\partial \mathbf{w}}\right) + \lambda \mathbf{w}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(-y_i \cdot \frac{\mathbf{x}_i h_{\mathbf{w}}(\mathbf{x}_i)\big(1-h_{\mathbf{w}}(\mathbf{x}_i)\big)}{h_{\mathbf{w}}(\mathbf{x}_i)} + (1-y_i) \cdot \frac{\mathbf{x}_i h_{\mathbf{w}}(\mathbf{x}_i)(1-h_{\mathbf{w}}(\mathbf{x}_i))}{1-h_{\mathbf{w}}(\mathbf{x}_i)}\right) + \lambda \mathbf{w}$$

$$= \frac{1}{n}\sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i + \lambda \mathbf{w}$$

Contents

Extension to Multi-class Classification

Previous study considers $y \in \{0,1\}$, but what if $y \in \{0,1, \ldots, K-1\}$?

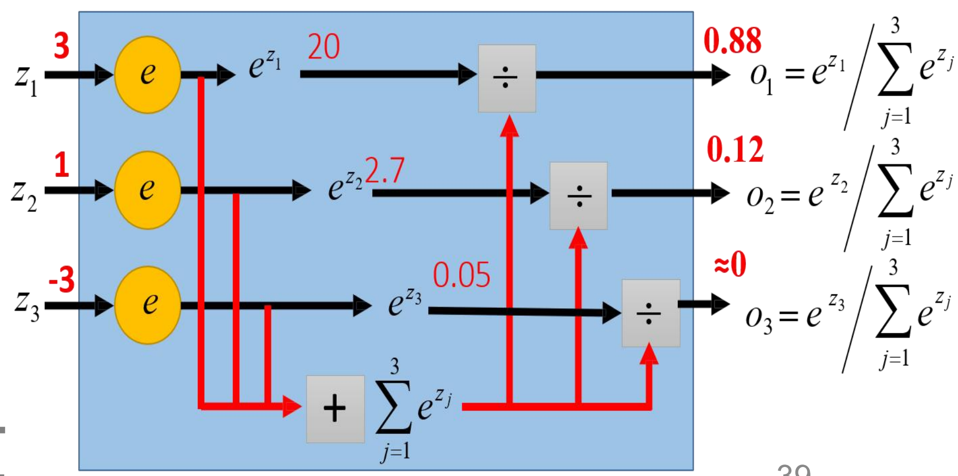
Dataset: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- $\mathbf{x}_i$ is the observation for the i^{th} instance
- $y_i \in \{0, 1, \ldots, K-1\}$ is the label for the i^{th} instance

- Task: Predict the probability of a testing instance $\mathbf{x}$ being to any class $j \in \{0, 1, \ldots, K-1\}$ as $o_j = g(\mathbf{w\_j}^{\mathrm{T}}\mathbf{x}) = g(z\_j)$

- Then o_j must follow:

$$0 \le o_j \le 1, \ \sum_j o_j = 1$$



Softmax Regression for Multi-class Classification

To handle **multi-class** task, for each class $j \in \{0, \dots, K-1\}$, we define a weight vector $\mathbf{w}_j$ associated with this class

- $\mathbf{W} := [\mathbf{w}_0 \ \mathbf{w}_1 \ \dots \ \mathbf{w}_{K-1}]$ is a matrix of K weight vectors

$$\mathbf{W} = \begin{bmatrix} | & | & | & | \\ \mathbf{w}_0 & \mathbf{w}_1 & \cdots & \mathbf{w}_{K-1} \\ | & | & | & | \end{bmatrix}_{m \times K}$$

Here, m is the dimension of the sample, K is the number of classes

- Let $z_j = \mathbf{w}_j^{\mathrm{T}}\mathbf{x}$. We define the probability of an instance $\mathbf{x}$ being to any class $j \in \{0, 1, \dots, K-1\}$ as:

$$o_j = \mathrm{P}(y = j | \mathbf{x}; \mathbf{W}) = \frac{e^{z_j}}{\sum_{l=0}^{K-1} e^{z_l}} = \frac{e^{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}}}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}}\mathbf{x}}}$$

Softmax Regression for Multi-class Classification

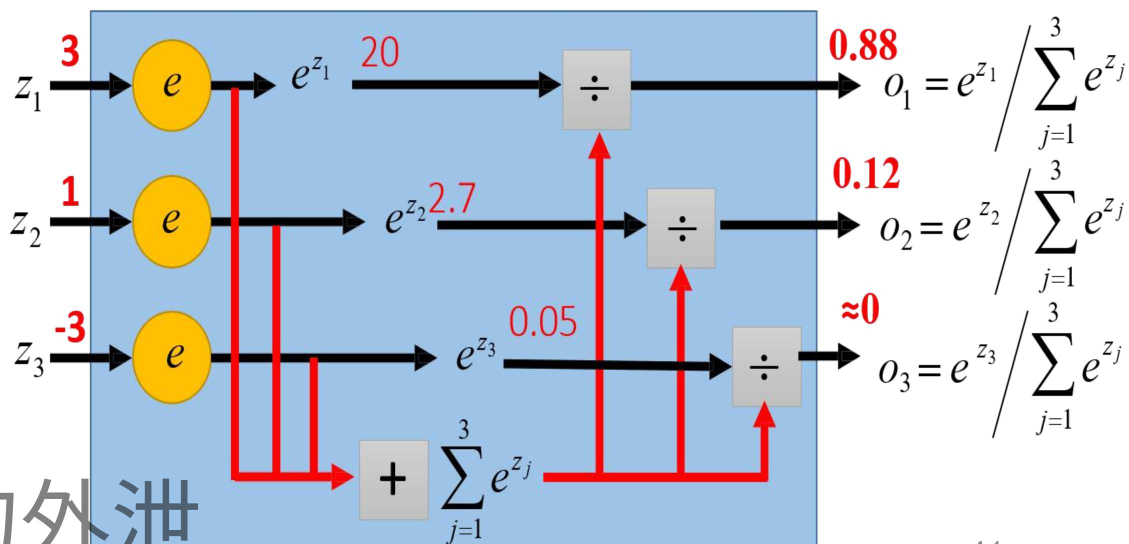
- Recall that the probability of an instance $\mathbf{x}$ being to any class j is:

$$o_j = \mathrm{P}(y = j|\mathbf{x}; \mathbf{W}) = \frac{e^{z_j}}{\sum_{l=0}^{K-1} e^{z_l}} = \frac{e^{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}}}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}}\mathbf{x}}}$$

- The function $\frac{e^{z_j}}{\sum_{l=0}^{K-1} e^{z_l}}$ is called **Softmax function**, where $\sum_{l=0}^{K-1} e^{z_l}$ is a normalization term to make all the elements **be summed to 1**

- Obviously, o_j follows:

$$\boxed{0 \le o_j \le 1, \ \sum_j o_j = 1}$$



Softmax Regression for Multi-class Classification

- For an instance $\mathbf{x}$, it can belong to any class j with probability:

$$H_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix} P(y=0|\mathbf{x};\mathbf{W}) \\ \vdots \\ P(y=j|\mathbf{x};\mathbf{W}) \\ \vdots \\ P(y=K-1|\mathbf{x};\mathbf{W}) \end{bmatrix} = \frac{1}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_0^{\mathrm{T}}\mathbf{x}} \\ \vdots \\ e^{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}} \\ \vdots \\ e^{\mathbf{w}_{K-1}^{\mathrm{T}}\mathbf{x}} \end{bmatrix}$$

- **Prediction**: Given any model **W**, we can predict the label by:

Prediction: $\hat{y} = \mathrm{argmax}_{j\in\{0,1,\ldots,K-1\}} P(y=j|\mathbf{x};\mathbf{W})$

How to learn a good **W** to ensure correct prediction?

Cross-Entropy Loss for Multi-class Classification

- To learn $\mathbf{W} := [\mathbf{w}_0 \ \mathbf{w}_1 \ \ \dots \ \ \mathbf{w}_{K-1}]$, relying on the softmax function, we introduce the following **Cross-Entropy loss**:

$$\mathcal{L}(\mathbf{W}) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=0}^{K-1}\mathbb{I}\{y_i = j\}\log\frac{e^{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}_i}}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}}\mathbf{x}_i}}\right]$$

where $\mathbb{I}\{\cdot\}$ is the indicator function as follows:

$$\mathbb{I}\{\mathrm{A}\} = \begin{cases} 1, & \text{if A is a true statemet} \\ 0, & \text{if A is a false statemet} \end{cases}$$

- The cross-entropy loss can be derived by Maximum Likelihood Estimation (MLE). Here, we omit the details.

Regularization Required

We employ Regularization to avoid overfitting issue

- We have the following **objective function** for softmax regression:

$$J(\mathbf{W}) = \mathcal{L}(\mathbf{W}) + \frac{\lambda}{2}||\mathbf{W}||_2^2$$

Here, $\mathcal{L}(\mathbf{W}) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=0}^{K-1}\mathbb{I}\{y_i = j\}\log\frac{e^{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}_i}}{\sum_{l=0}^{K-1}e^{\mathbf{w}_l^{\mathrm{T}}\mathbf{x}_i}}\right]$ is called **Cross-Entropy Loss** and λ is the regularization parameter.

- Update parameters $\mathbf{W}$ by (Stochastic) Gradient Descent:

$$\mathbf{W} := \mathbf{W} - \eta\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

How to compute $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$?

How to compute $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$?

- For $\mathbf{w}_k$ $(k = 0, \dots, K-1)$, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k}$ can be computed as follows:

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} = \frac{\partial \left\{ -\frac{1}{n}\left[\sum_{i=1}^{n} \sum_{j=0}^{K-1} \mathbb{I}\{y_i = j\} \log \frac{e^{\mathbf{w}_j^{\mathrm{T}} \mathbf{x}_i}}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_i}} \right] + \frac{\lambda}{2} ||\mathbf{W}||_2^2 \right\}}{\partial \mathbf{w}_k}$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \frac{\partial \sum_{j=0}^{K-1} \mathbb{I}\{y_i = j\} \left(\mathbf{w}_j^{\mathrm{T}} \mathbf{x}_i - \log \sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_i} \right)}{\partial \mathbf{w}_k} + \lambda \mathbf{w}_k$$

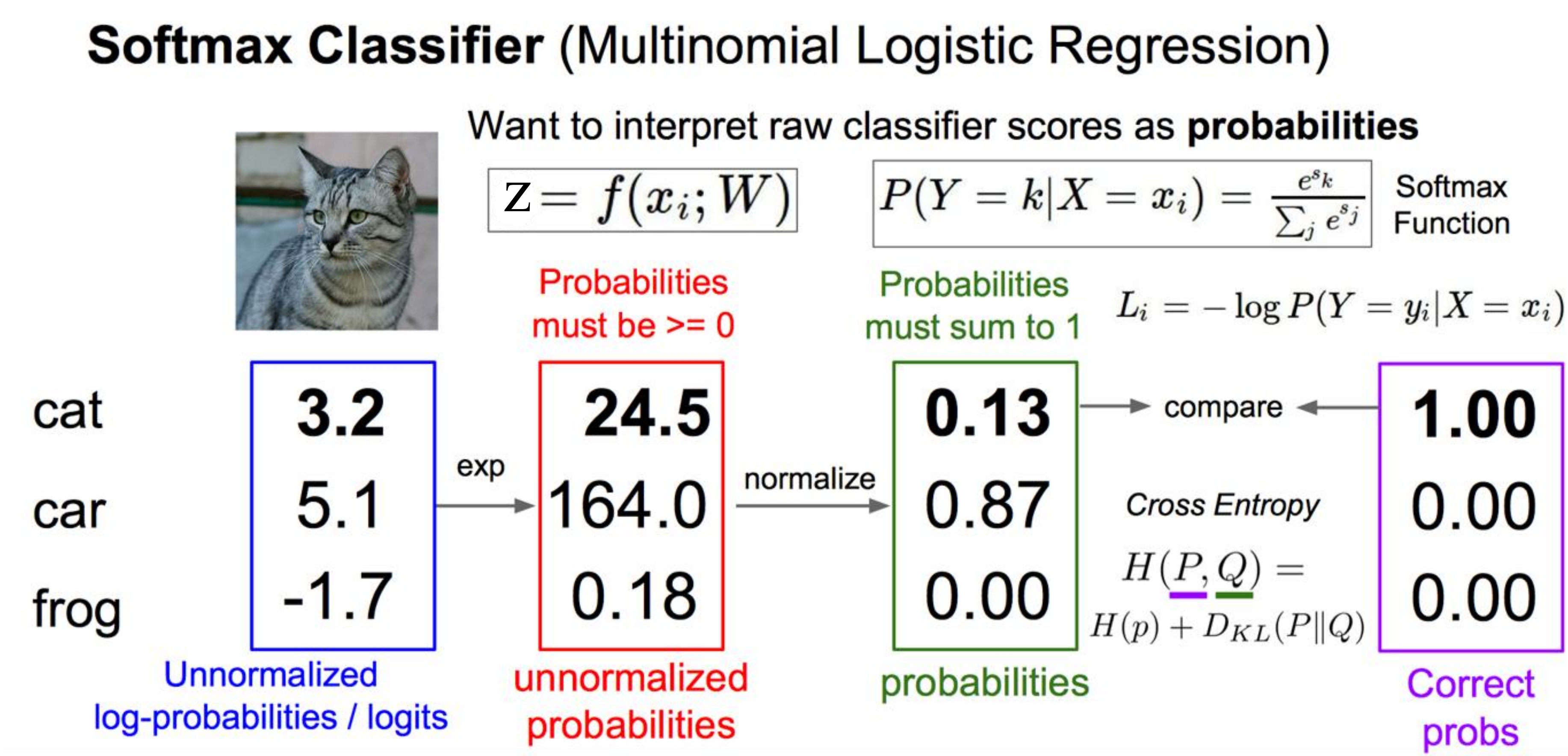
$$= -\frac{1}{n} \sum_{i=1}^{n} \frac{\partial \left[\mathbb{I}\{y_i = k\} \mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i - \log \sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_i} \right]}{\partial \mathbf{w}_k} + \lambda \mathbf{w}_k$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[\mathbb{I}\{y_i = k\} \mathbf{x}_i - \frac{1}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_i}} \cdot \frac{\partial \sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_i}}{\partial \mathbf{w}_k} \right] + \lambda \mathbf{w}_k$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[\mathbb{I}\{y_i = k\} \mathbf{x}_i - \frac{\mathbf{x}_i e^{\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i}}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_i}} \right] + \lambda \mathbf{w}_k$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left[P(y_i = k | \mathbf{x}_i; \mathbf{W}) - \mathbb{I}\{y_i = k\} \right] \mathbf{x}_i + \lambda \mathbf{w}_k$$

Example of Softmax Regression



Softmax Regression for Binary Classification

Previous cases consider softmax regression for multi-class classification. Can we use it for binary classification i.e., a special case where $K = 2$?

- Recall that an instance $\mathbf{x}$ can belong to any class j with probability:

$$H_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix} P(y=0|\mathbf{x};\mathbf{W}) \\ \vdots \\ P(y=j|\mathbf{x};\mathbf{W}) \\ \vdots \\ P(y=K-1|\mathbf{x};\mathbf{W}) \end{bmatrix} = \frac{1}{\sum_{l=0}^{K-1} e^{\mathbf{w}_l^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_0^{\mathrm{T}}\mathbf{x}} \\ \vdots \\ e^{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}} \\ \vdots \\ e^{\mathbf{w}_{K-1}^{\mathrm{T}}\mathbf{x}} \end{bmatrix}$$

- When $K = 2$, we have:

$$H_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix} P(y=0|\mathbf{x};\mathbf{W}) \\ P(y=1|\mathbf{x};\mathbf{W}) \end{bmatrix} = \frac{1}{e^{\mathbf{w}_0^{\mathrm{T}}\mathbf{x}} + e^{\mathbf{w}_1^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_0^{\mathrm{T}}\mathbf{x}} \\ e^{\mathbf{w}_1^{\mathrm{T}}\mathbf{x}} \end{bmatrix}$$

Then, softmax regression is reduced to logistic regression

Softmax Regression for Binary Classification

- Recall that the weight matrix is $\mathbf{W} := [\mathbf{w}_0 \ \mathbf{w}_1]$
- When $K = 2$, we have

$$H_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix} P(y = 0|\mathbf{x};\mathbf{W}) \\ P(y = 1|\mathbf{x};\mathbf{W}) \end{bmatrix}$$

$$= \frac{1}{e^{\mathbf{w}_0^{\mathrm{T}}\mathbf{x}} + e^{\mathbf{w}_1^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_0^{\mathrm{T}}\mathbf{x}} \\ e^{\mathbf{w}_1^{\mathrm{T}}\mathbf{x}} \end{bmatrix}$$

$$= \frac{1}{e^{(\mathbf{w}_0 - \mathbf{w}_1)^{\mathrm{T}}\mathbf{x}} + e^{(\mathbf{w}_1 - \mathbf{w}_1)^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{(\mathbf{w}_0 - \mathbf{w}_1)^{\mathrm{T}}\mathbf{x}} \\ e^{(\mathbf{w}_1 - \mathbf{w}_1)^{\mathrm{T}}\mathbf{x}} \end{bmatrix}$$

$$= \frac{1}{e^{(\mathbf{w}_0 - \mathbf{w}_1)^{\mathrm{T}}\mathbf{x}} + e^{(0)^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{(\mathbf{w}_0 - \mathbf{w}_1)^{\mathrm{T}}\mathbf{x}} \\ e^{(0)^{\mathrm{T}}\mathbf{x}} \end{bmatrix}$$

Softmax Regression for Binary Classification

- Let $-\mathbf{w} = \mathbf{w}_0 - \mathbf{w}_1$, $H_{\mathbf{w}}(\mathbf{x}) = \begin{bmatrix} P(y=0|\mathbf{x};\mathbf{W}) \\ P(y=1|\mathbf{x};\mathbf{W}) \end{bmatrix}$

$$= \frac{1}{1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}}} \begin{bmatrix} e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}} \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - \dfrac{1}{1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}}} \\ \dfrac{1}{1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}}} \end{bmatrix}$$

$$\boxed{= \begin{bmatrix} 1 - h_{\mathbf{w}}(\mathbf{x}) \\ h_{\mathbf{w}}(\mathbf{x}) \end{bmatrix}}$$

Probability in Logistic Regression:

$$P(y|\mathbf{x}) = \begin{cases} 1 - h_{\mathbf{w}}(\mathbf{x}), & y = 0 \\ h_{\mathbf{w}}(\mathbf{x}), & y = 1 \end{cases}$$

Logistic regression is a special case of softmax regression

Logistic Loss vs Softmax Cross-Entropy Loss

Cross-Entropy loss: $$\boxed{E = -\sum_{j=0}^{K-1} \mathbb{I}\{y = j\} \log(o_j)}$$

- Logistic loss for binary classification (K=2):

$\xrightarrow{\mathrm{z}}$ Sigmoid $\rightarrow$ Cross-Entropy loss

$$o_0 = \frac{1}{1+e^{-\mathrm{z}}}$$
$$o_1 = \frac{e^{-\mathrm{z}}}{1+e^{-\mathrm{z}}}$$

$$E = -\sum_{j=0}^{1} \mathbb{I}\{y = j\} \log(o_j)$$

- Softmax Cross-Entropy loss for multi-class classification:

$\xrightarrow{\mathrm{z}_j}$ Softmax $\rightarrow$ Cross-Entropy loss

$$o_j = \frac{e^{\mathrm{z}_j}}{\sum_{l=0}^{K-1} e^{\mathrm{z}_l}}$$

$$E = -\sum_{j=0}^{K-1} \mathbb{I}\{y = j\} \log \frac{e^{\mathrm{z}_j}}{\sum_{l=0}^{K-1} e^{\mathrm{z}_l}}$$

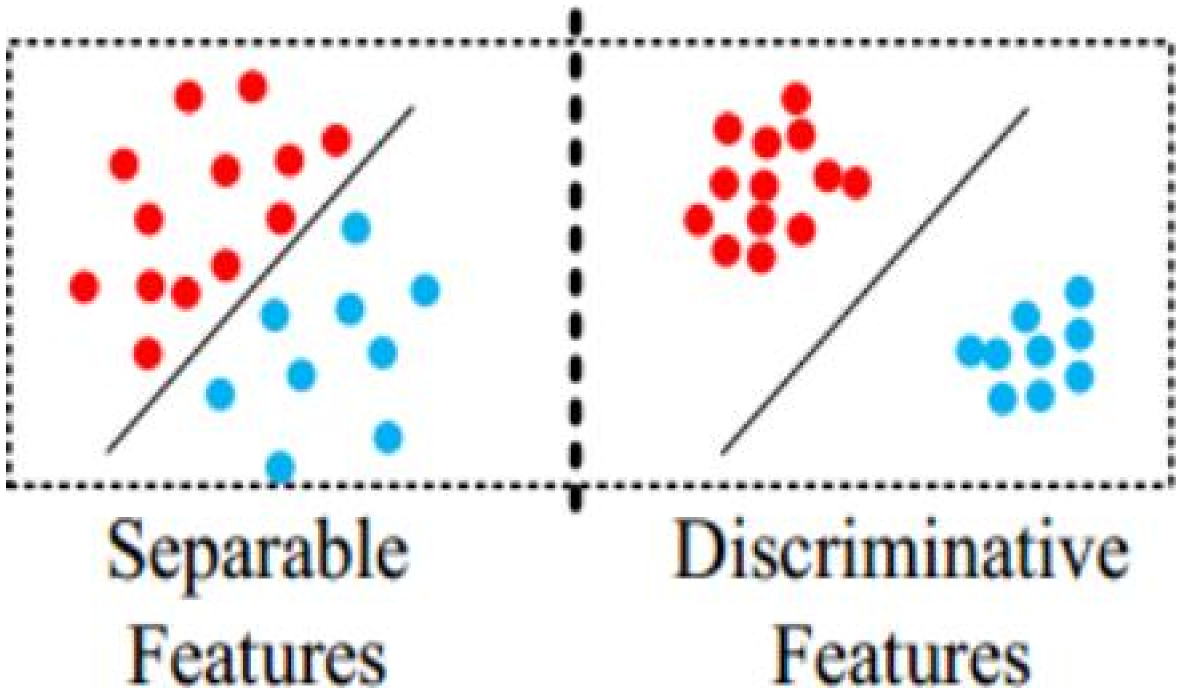
Contents

Two Variants of the Softmax Loss

- **Large-Margin Softmax Loss**
- Angular Softmax Loss

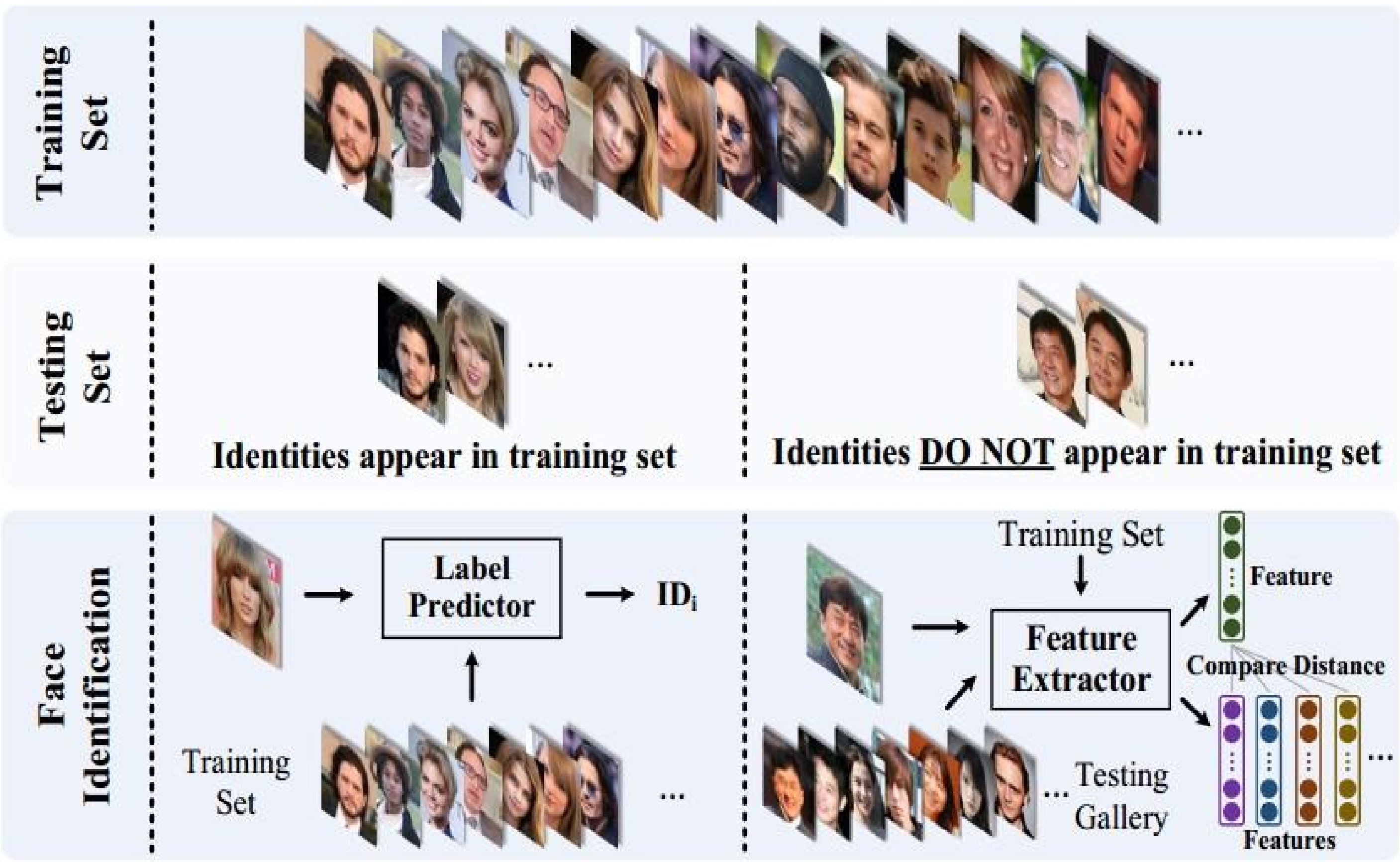
Motivation

- Learn discriminative features



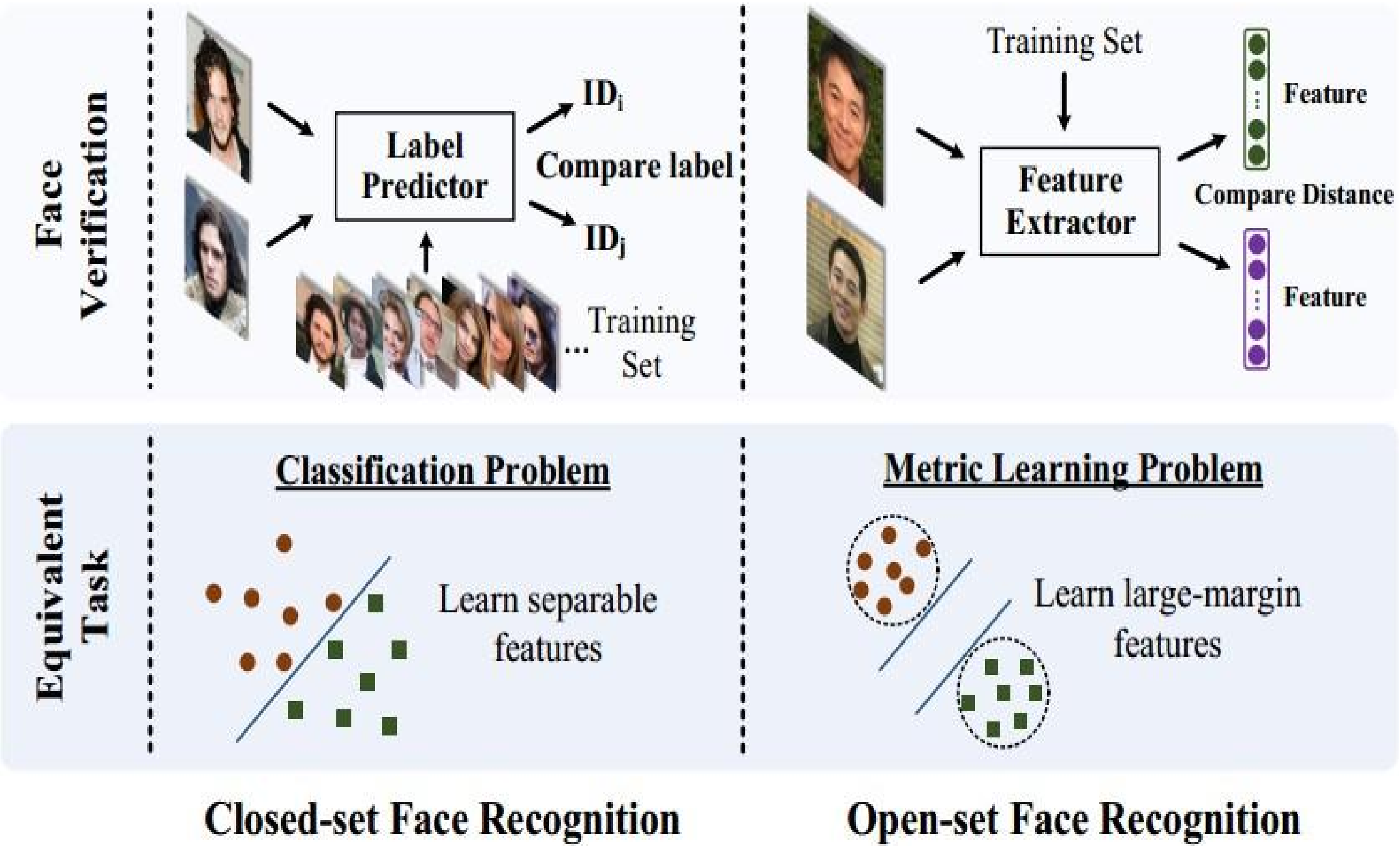
Motivation

Closed-set and Open-set Face Recognition



Motivation

Closed-set and Open-set Face Recognition



Softmax Loss

- Given input feature $\mathbf{x}_i$ with the label y_i, the softmax loss function is:

$$\mathcal{L} = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

- f_j denotes the j-th element of the vector of class scores f
- N is the number of training data

$$f_{y_i} = \mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i = \left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \cos\left(\theta_j\right)$$

$$\mathcal{L}_i = -\log\left(\frac{e^{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \cos\left(\theta_{y_i}\right)}}{\sum_j e^{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \cos\left(\theta_J\right)}}\right)$$

- θ_j $(0 \le \theta_j \le \pi)$ is the angle between the vector $\mathbf{w}_j$ and $\mathbf{x}_i$

Large-Margin Softmax Loss

- Consider the binary classification and a sample x from class 1

- Original softmax

$$\|\mathbf{w}_1\|\|\mathbf{x}\|\cos(\theta_1) > \|\mathbf{w}_2\|\|\mathbf{x}\|\cos(\theta_2)$$

- Large-Margin softmax

$$\|\mathbf{w}_1\|\|\mathbf{x}\|\cos(m\theta_1) > \|\mathbf{w}_2\|\|\mathbf{x}\|\cos(\theta_2)\ (0 \leq \theta_1 \leq \frac{\pi}{m})$$

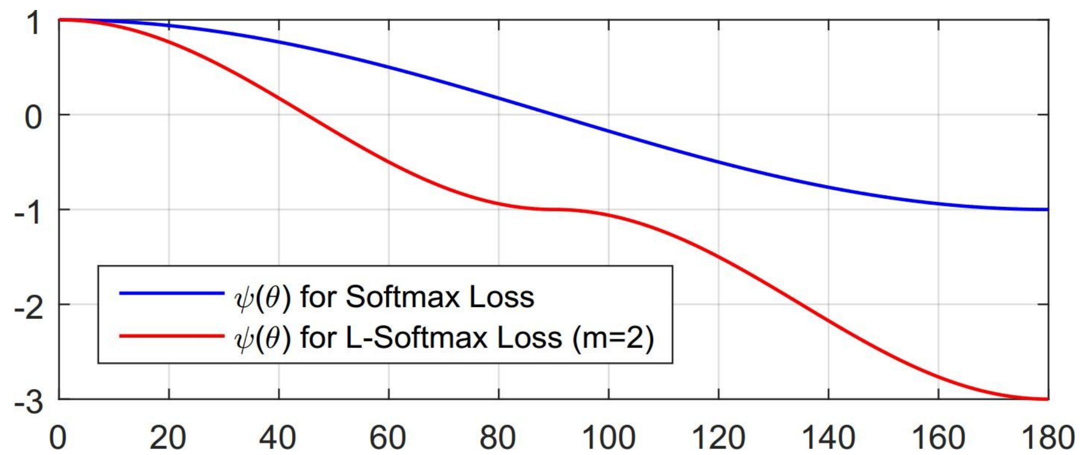
Large-Margin Softmax Loss

- Large-Margin Softmax Loss:

$$L_i = -\log\left(\frac{e^{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \psi\left(\theta_{y_i}\right)}}{e^{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \psi\left(\theta_{y_i}\right)} + \sum_{j \neq y_i} e^{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \cos\left(\theta_j\right)}}\right)$$

$$\psi(\theta) = \begin{cases} \cos(m\theta), & 0 \le \theta \le \dfrac{\pi}{m} \\ \mathcal{D}(\theta), & \dfrac{\pi}{m} \le \theta \le \pi \end{cases}$$

Large-Margin Softmax Loss



ψ(θ) for softmax loss and L-Softmax loss

- Construct a specific ψ(θ):

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k,\ \theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$$

where $k \in [0, m-1]$ and k is an integer

Large-Margin Softmax Loss

- Replace $\cos(\theta_j)$ with

$$\frac{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}_i}{\left\|\mathbf{w}_j\right\|\left\|\mathbf{x}_i\right\|}$$

- Replace $\cos(m\theta_{y_i})$ with

$$\cos\left(m\theta_{y_i}\right) = C_m^0 cos^m\left(\theta_{y_i}\right) - C_m^2 cos^{m-2}\left(\theta_{y_i}\right)\left(1 - cos^2\left(\theta_{y_i}\right)\right) +$$
$$C_m^4 cos^{m-4}\left(\theta_{y_{y_i}}\right)\left(1 - cos^2\left(\theta_{y_{y_i}}\right)\right)^2 + \ldots$$
$$(-1)^n C_m^{2n} cos^{m-2n}(\theta_{y_{y_i}})\left(1 - cos^2\left(\theta_{y_i}\right)\right)^n + \ldots$$

Large-Margin Softmax Loss

- So we can get:

$$f_{y_i} = (-1)^k \cdot \left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\| \cos(m\theta_i) - 2k \cdot \left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|$$

$$= (-1)^k \cdot \left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|$$

$$\cdot \left(C_m^0 \left(\frac{\mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|} \right)^m - C_m^2 \left(\frac{\mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|} \right)^{m-2} \left(1 - \left(\frac{\mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|} \right)^2 \right) + \ldots \right) - 2k$$

$$-2k \cdot \left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|$$

where $\frac{\mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|} \in \left[\cos\left(\frac{k\pi}{m}\right), \cos\left(\frac{(k+1)\pi}{m}\right)\right]$ and k is an integer that to $[0, m-1]$.

Large-Margin Softmax Loss
Optimization

$$\frac{\partial f_{y_i}}{\partial \boldsymbol{x}_i} = (-1)^k \cdot (C_m^0 \left(\frac{m\left(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i\right)^{m-1}\mathbf{w}_{y_i}}{\left(\left\|\mathbf{w}_{y_i}\right\|\|\mathbf{x}_i\|\right)^{m-1}} \right) -$$

$$C_m^0 \left(\frac{(m-1)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^m \mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\|^{m-1}\|\mathbf{x}_i\|^{m+1}} \right) - C_m^2 \left(\frac{(m-2)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^{m-3}\mathbf{w}_{y_i}}{\left(\left\|\mathbf{w}_{y_i}\right\|\|\mathbf{x}_i\|\right)^{m-3}} \right)$$

$$+ C_m^2 \left(\frac{(m-3)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^{m-2}\mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\|^{m-3}\|\mathbf{x}_i\|^{m-1}} \right) + C_m^2 \left(\frac{m\,(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^{m-1}\mathbf{w}_{y_i}}{\left(\left\|\mathbf{w}_{y_i}\right\|\|\mathbf{x}_i\|\right)^{m-1}} \right)$$

$$- C_m^2 \left(\frac{(m-1)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^m \mathbf{x}_i}{\left\|\mathbf{w}_{y_i}\right\|^{m-1}\|\mathbf{x}_i\|^{m+1}} \right) + \ldots) - 2k \cdot \frac{\left\|\mathbf{w}_{y_i}\right\|\mathbf{x}_i}{\|\mathbf{x}_i\|}$$

Large-Margin Softmax Loss
Optimization

$$
\begin{aligned}
\frac{\partial f_{y_i}}{\partial \mathbf{w}_{y_i}} = (-1)^k \cdot (C_m^0 \left(\frac{m\left(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i\right)^{m-1}\mathbf{x}_i}{\left(\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|\right)^{m-1}} \right) \\
- C_m^0 \left(\frac{(m-1)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^m \mathbf{w}_{y_i}}{\left\|\mathbf{w}_{y_i}\right\|^{m+1} \|\mathbf{x}_i\|^{m-1}} \right) - C_m^2 \left(\frac{(m-2)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^{m-3}\mathbf{x}_i}{\left(\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|\right)^{m-3}} \right) \\
+ C_m^2 \left(\frac{(m-3)(\mathbf{w}_{y_i}^{\mathrm{T}} x_i)^{m-2}\mathbf{w}_{y_i}}{\left\|\mathbf{w}_{y_i}\right\|^{m-1} \|\mathbf{x}_i\|^{m-3}} \right) + C_m^2 \left(\frac{m\,(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^{m-1}\mathbf{x}_i}{\left(\left\|\mathbf{w}_{y_i}\right\| \|\mathbf{x}_i\|\right)^{m-1}} \right) \\
- C_m^2 \left(\frac{(m-1)(\mathbf{w}_{y_i}^{\mathrm{T}}\mathbf{x}_i)^m \mathbf{w}_{y_i}}{\left\|\mathbf{w}_{y_i}\right\|^{m+1} \|\mathbf{x}_i\|^{m-1}} \right) + \ldots) - 2k \cdot \frac{\|\mathbf{x}_i\| \mathbf{w}_{y_i}}{\left\|\mathbf{w}_{y_i}\right\|}
\end{aligned}
$$

Geometric Interpretation

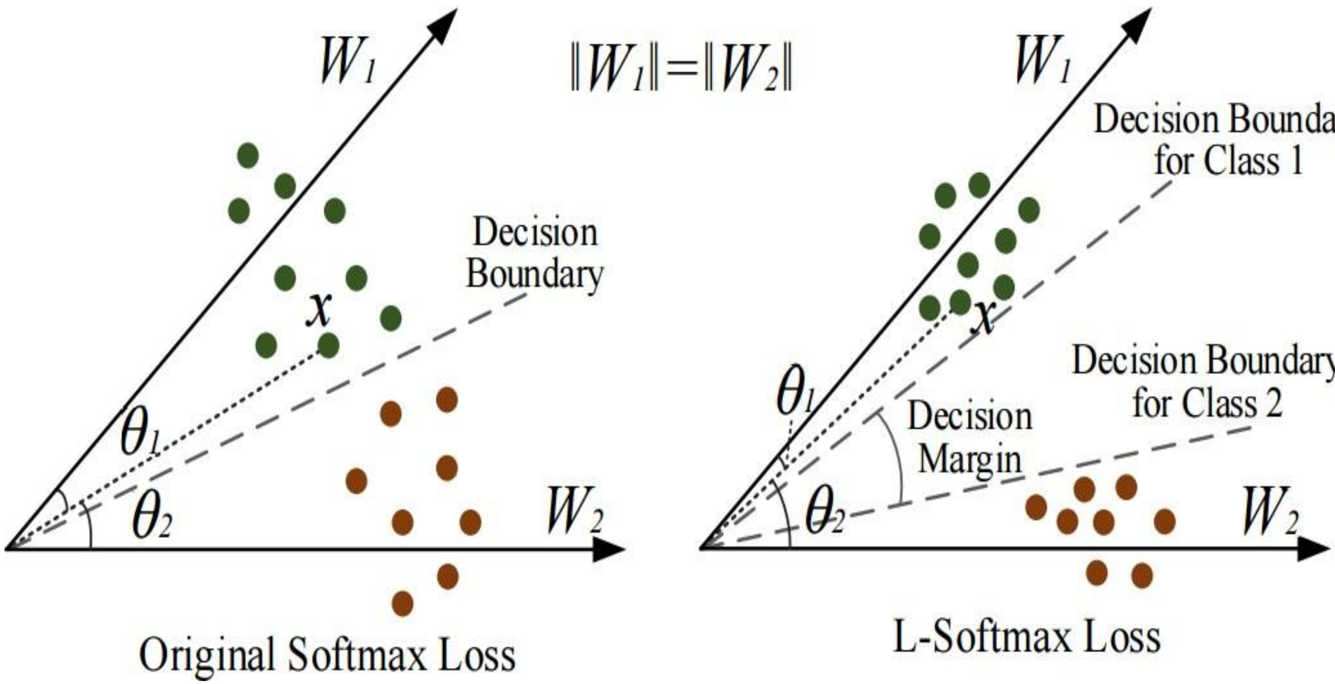


Figure: Example of Geometric Interpretation when $\|\mathbf{w_1}\| = \|\mathrm{w}_2\|$

Geometric Interpretation

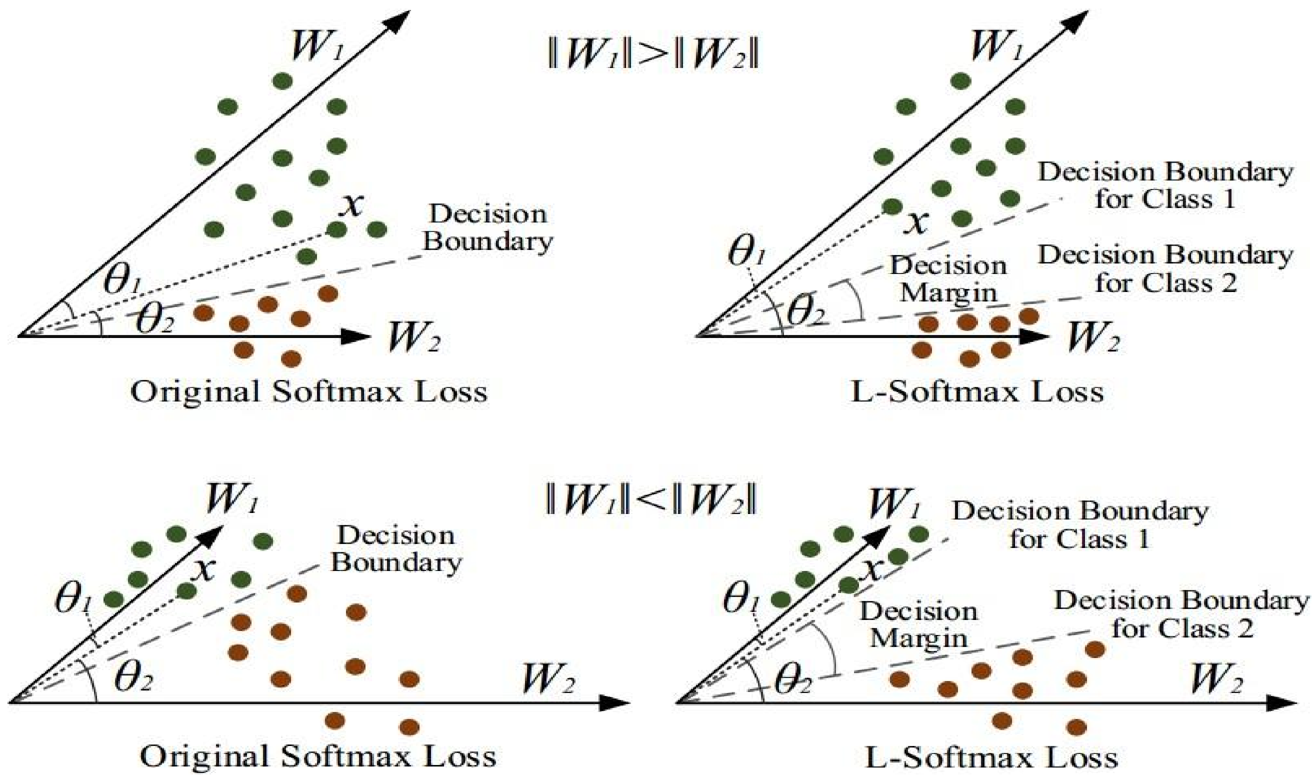


Figure: Example of Geometric Interpretation when $\|\mathbf{w_1}\| > \|\mathbf{w_2}\|$ and $\|\mathbf{w_1}\| < \|\mathbf{w_2}\|$

The variants of the softmax loss

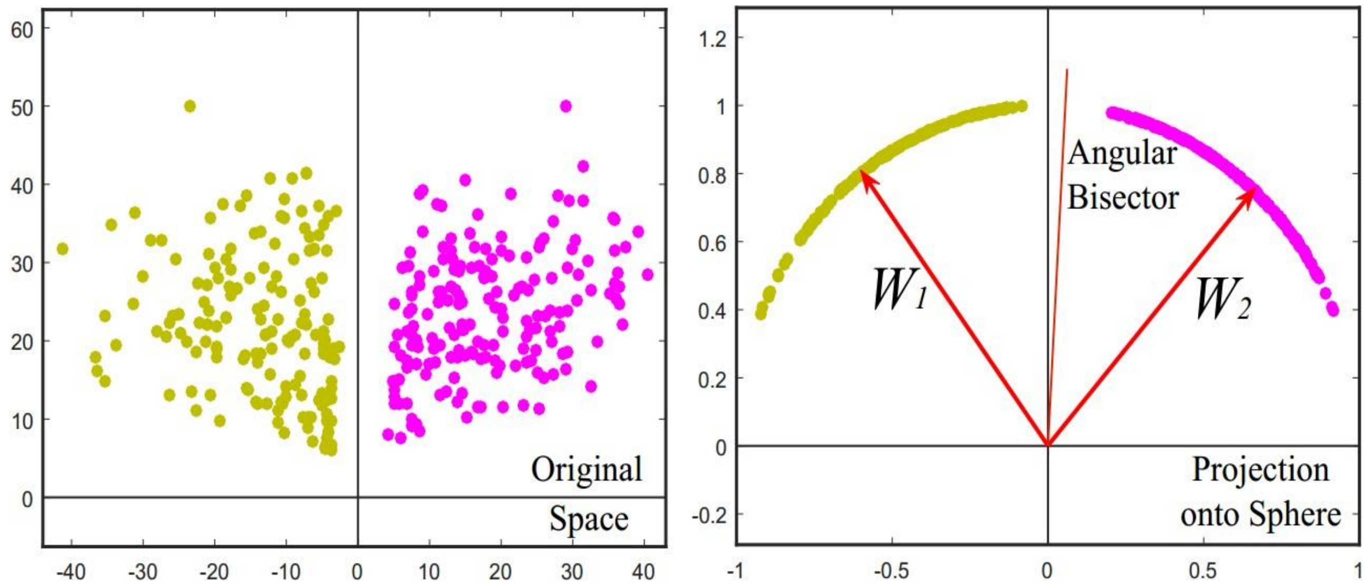
- Large-Margin Softmax Loss
- **Angular Softmax Loss (A-Softmax Loss)**

Modified Softmax Loss Function

- Normalize $\|\mathbf{w}_j\| = 1,\ \forall j$ in each iteration

$$\mathcal{L}_{modified} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\mathbf{x}_i\|\cos(\theta_{y_i,i})}}{\sum_j e^{\|\mathbf{x}_i\|\cos(\theta_{j,i})}}\right)$$

Modified Softmax Loss Function



- Learn a 2-D features on subset of CASIA face dataset

A-Softmax Loss

- Consider the binary classification and a sample *x* from class 1

- Modified softmax loss need

$$\|x\| \cos(\theta_1) > \|x\| \cos(\theta_2)$$

- A-Softmax loss need

$$\|x\| \cos(m\theta_1) > \|x\| \cos(\theta_2) \quad (0 \leq \theta_1 \leq \frac{\pi}{m})$$

A-Softmax Loss

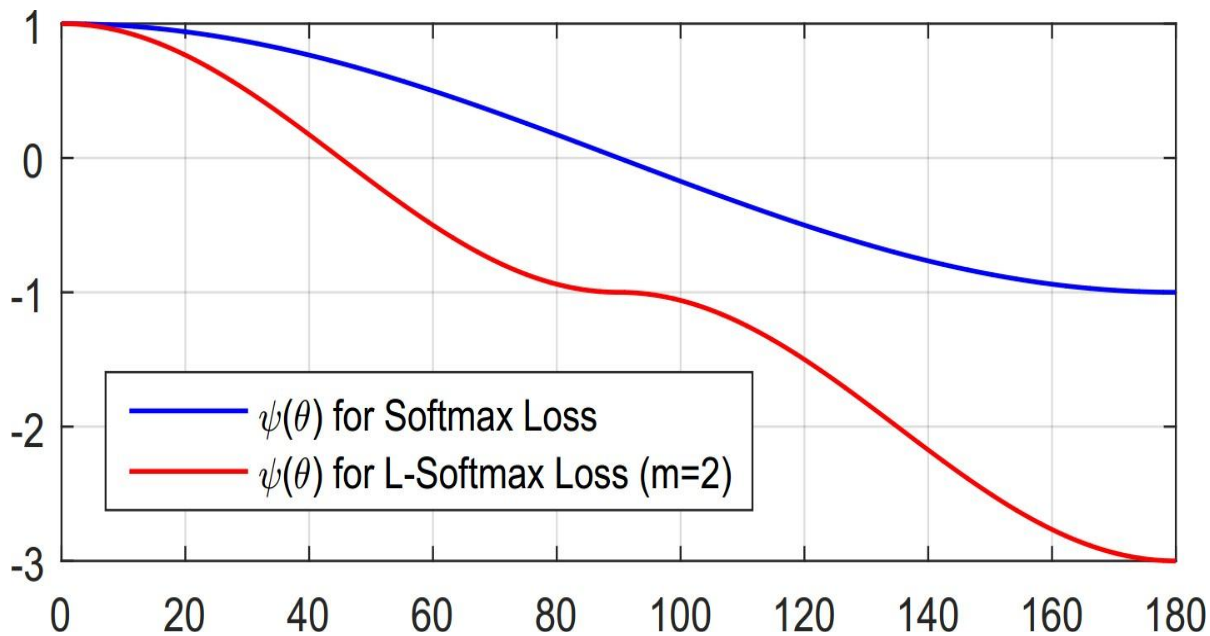
$$L_{ang} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\mathbf{x}_i\|\cos(m\theta_{y_i},i)}}{e^{\|\mathbf{x}_i\|\cos(m\theta_{y_i},i)} + \sum_{j\neq y_i} e^{\|\mathbf{x}_i\|\cos(\theta_j,i)}}\right)$$

where θ_{y_i}, i has to be in the range of $[0, \frac{\pi}{m}]$

$$L_{ang} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\mathrm{x}_i\|\psi(\theta_{y_i},i)}}{e^{\|\mathrm{x}_i\|\psi(\theta_{y_i},i)} + \sum_{j\neq y_i} e^{\|\mathrm{x}_i\|\cos(\theta_j,i)}}\right)$$

$$\psi(\theta) = \begin{cases} \cos(m\theta), & 0 \le \theta \le \dfrac{\pi}{m} \\ \mathcal{D}(\theta), & \dfrac{\pi}{m} \le \theta \le \pi \end{cases}$$

A-Softmax Loss

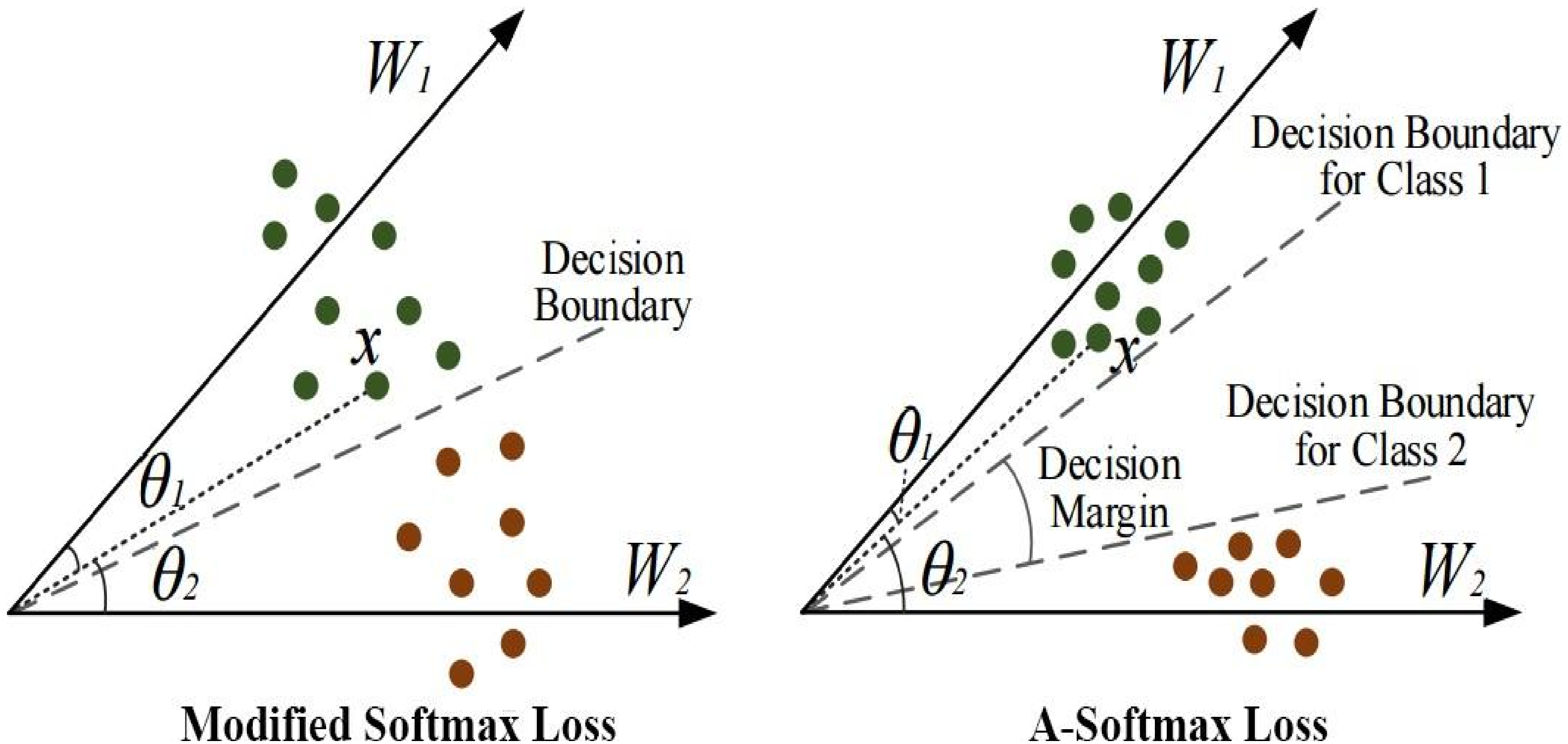


- Construct a specific $\psi(\theta)$:

$$\psi(\theta) = (-1)^k cos(m\theta) - 2k,\ \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$$

where $k \in [0, m-1]$ and k is an integer

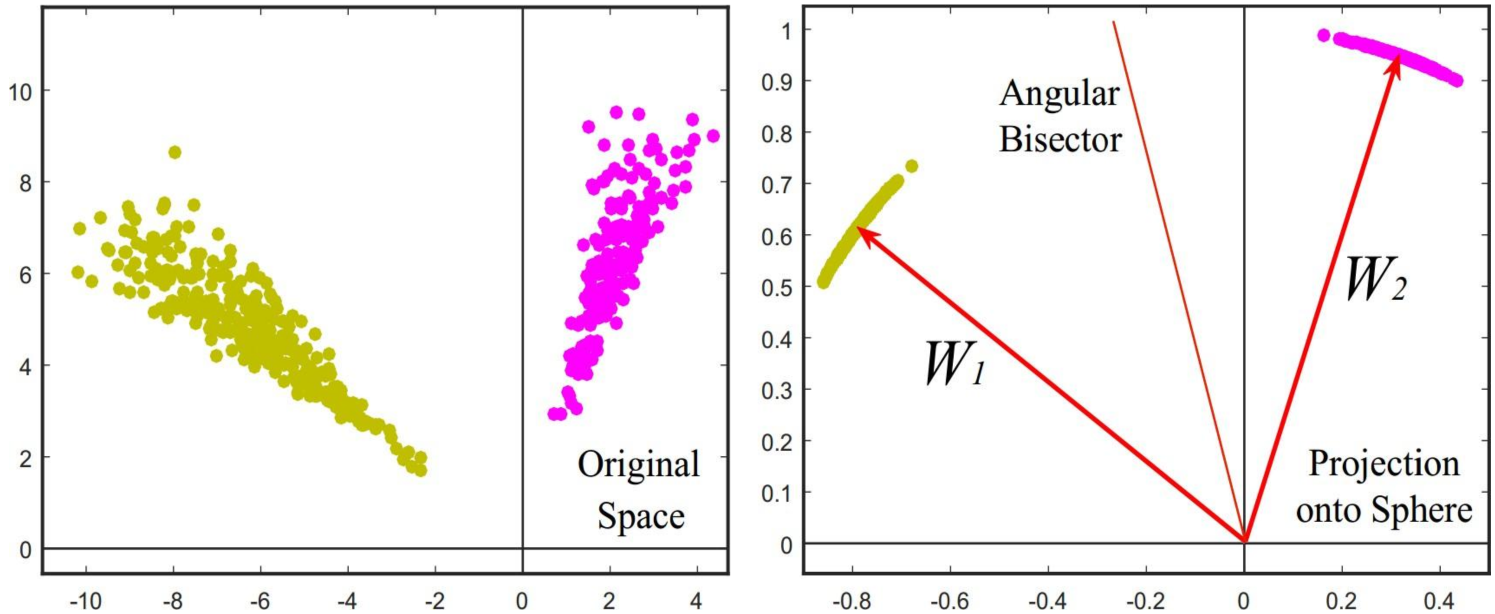
A-Softmax Loss
Geometric Interpretation



Loss Function	Decision Boundary
Softmax Loss	$(\boldsymbol{W}_1 - \boldsymbol{W}_2)\boldsymbol{x} + b_1 - b_2 = 0$
Modified Softmax Loss	$\lVert\boldsymbol{x}\rVert(\cos\theta_1 - \cos\theta_2) = 0$
A-Softmax Loss	$\lVert\boldsymbol{x}\rVert(\cos m\theta_1 - \cos\theta_2) = 0$ for class 1 $\lVert\boldsymbol{x}\rVert(\cos\theta_1 - \cos m\theta_2) = 0$ for class 2

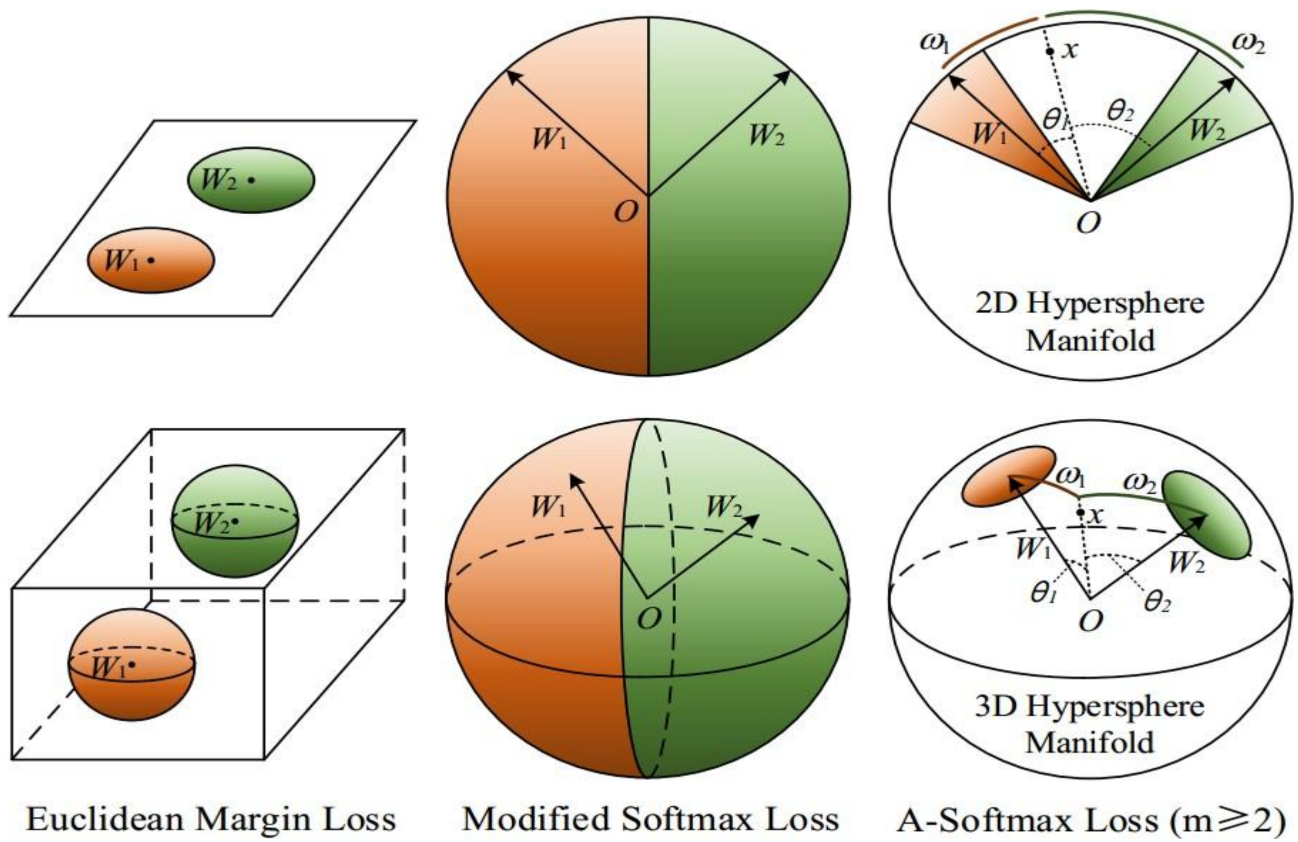
- θ_i is the angle between $\mathbf{w}_\mathrm{i}$ and $\mathbf{x}$

A-Softmax Loss



- Learn 2-D features on a subset of CASIA face dataset

Hypersphere Interpretation



References

[1] Liu W, Wen Y, Yu Z, et al. Large-margin softmax loss for convolutional neural networks[C]//ICML. 2016, 2(3): 7.

[2] Liu W, Wen Y, Yu Z, et al. Sphereface: Deep hypersphere embedding for face recognition[C] //Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 212-220.

Thank You