

January 9, 2018



Inter-University Research Institute Corporation /
Research Organization of Information and Systems

National Institute of Informatics

Classification

Data Mining 09 (データマイニング)

Mahito Sugiyama (杉山磨人)

Today's Outline

- Today's topic is **classification**
 - The main task of **supervised learning**
- Predict the label of a data point
 - If labels are continuous (numeric), the task is usually called **regression**
- Cover basic classification methods
 - Naïve Bayes, logistic regression, *k*NN, decision tree

Bayes Approach to Classification

- Given a supervised dataset $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$,
 $\mathbf{x}_i \in \mathbb{R}^n$ (feature vector), $y_i \in C = \{c_1, c_2, \dots, c_K\}$ (label)
- The Bayes approach:
Estimate the posterior probability $P(c | \mathbf{x})$ from data and
predict the class y of \mathbf{x} as $\hat{y} = \operatorname{argmax}_{c \in C} P(c | \mathbf{x})$

Bayes Classification

- Use the **Bayes theorem**:

$$P(c | \mathbf{x}) = \frac{P(\mathbf{x} | c) \cdot P(c)}{P(\mathbf{x})}$$

- $P(c | \mathbf{x})$: **posterior**, $P(\mathbf{x} | c)$: **likelihood**, $P(c)$: **prior**
 - $P(\mathbf{x}) = \sum_{c \in \mathcal{C}} P(\mathbf{x} | c) \cdot P(c)$
- Since the denominator $P(\mathbf{x})$ is independent of classes c (just a normalizing constant),

$$\hat{y} = \operatorname{argmax}_{c \in \mathcal{C}} P(c | \mathbf{x}) = \operatorname{argmax}_{c \in \mathcal{C}} P(\mathbf{x} | c)P(c)$$

Prior Probability Estimation

- **Goal:** Estimate the prior $P(c)$ from a dataset D
- For a given dataset D , for each class $c \in C$,
 $D_c = \{\mathbf{x} \mid (\mathbf{x}, y) \in D \text{ and } y = c\}$
- We can directly estimate the prior $P(c)$ as the ratio:

$$\hat{P}(c) = \frac{|D_c|}{|D|}$$

Naïve Bayes Model

- **Goal:** Estimate the likelihood $P(\mathbf{x} | c)$ from a dataset D
- Assume that each feature is **independent** (the model is “naïve”):
 $P(\mathbf{x} | c) = \prod_{j=1}^n P(x^j | c), \quad \mathbf{x} = (x^1, x^2, \dots, x^n)$
- For each $j \in \{1, 2, \dots, n\}$, if we assume data is normally distributed,

$$P(x^j | c) \propto f(x^j; \mu_c^j, \sigma_c^{j2}) = \frac{1}{\sqrt{2\pi}\sigma_c^j} \exp\left(-\frac{(x^j - \mu_c^j)^2}{2\sigma_c^{j2}}\right)$$

$$P(\mathbf{x} | c) = \prod_{j=1}^n P(x^j | c) \propto \prod_{j=1}^n f(x^j; \mu_c^j, \sigma_c^{j2})$$

Algorithm 1: Naïve Bayes Classifier

```
1 learn( $D$ )
2   foreach  $c \in C$  do
3      $D_c \leftarrow \{\mathbf{x} \mid (\mathbf{x}, c) \in D\}$ 
4      $\hat{P}(c) \leftarrow |D_c| / |D|$ 
5     foreach  $j \in \{1, 2, \dots, n\}$  do
6        $\hat{\mu}_c^j \leftarrow (1/|D_c|) \sum_{\mathbf{x} \in D_c} x^j$ 
7        $\hat{\sigma}_c^{j^2} \leftarrow (1/|D_c|) \sum_{\mathbf{x} \in D_c} (x^j - \hat{\mu}_c^j)^2$ 
8 classify( $\mathbf{x}$ )
9    $\hat{y} \leftarrow \operatorname{argmax}_{c \in C} \hat{P}(c) \prod_{j=1}^n f(x^j; \hat{\mu}_c^j, \hat{\sigma}_c^{j^2})$ 
```

If Features Are Categorical

- Assume that the domain of j th feature is finite: $\Sigma^j = \{s_1, s_2, \dots, s_{m^j}\}$
 - The feature j is called **categorical** (discrete)
- Likelihood for each categorical value $s_i \in \Sigma^j$ is estimated as

$$\hat{P}(s_i | c) = \frac{|\{\mathbf{x} \in D_c \mid x^j = s_i\}|}{|D_c|}$$

- Label y of a test point \mathbf{x} is estimated as

$$\hat{y} = \operatorname{argmax}_{c \in C} \hat{P}(c) \prod_{j=1}^n \hat{P}(x^j | c)$$

kNN approach

- The **kNN** (k Nearest Neighbor) classifier predicts the label of \mathbf{x} to the majority class among its k nearest neighbors
- Sort a given dataset D as $(\mathbf{x}_{(1)}, y_{(1)}), (\mathbf{x}_{(2)}, y_{(2)}), \dots, (\mathbf{x}_{(N)}, y_{(N)})$ in increasing order according to the distance from a test point \mathbf{x}
 - Euclidean distance $\|\mathbf{x}_i - \mathbf{x}\|_2 = \sqrt{\sum_{j=1}^n (x_i^j - x^j)^2}$ is typically used
- Take the top- k points $(\mathbf{x}_{(1)}, y_{(1)}), (\mathbf{x}_{(2)}, y_{(2)}), \dots, (\mathbf{x}_{(k)}, y_{(k)})$ and
$$\hat{y} = \operatorname{argmax}_{c \in C} |\{(\mathbf{x}_{(i)}, y_{(i)}) \mid i \leq k \text{ and } y_{(i)} = c\}|$$
 - $|\{(\mathbf{x}_{(i)}, y_{(i)}) \mid i \leq k \text{ and } y_{(i)} = c\}|/k$ can be viewed as posterior $P(c \mid \mathbf{x})$

Logistic Regression

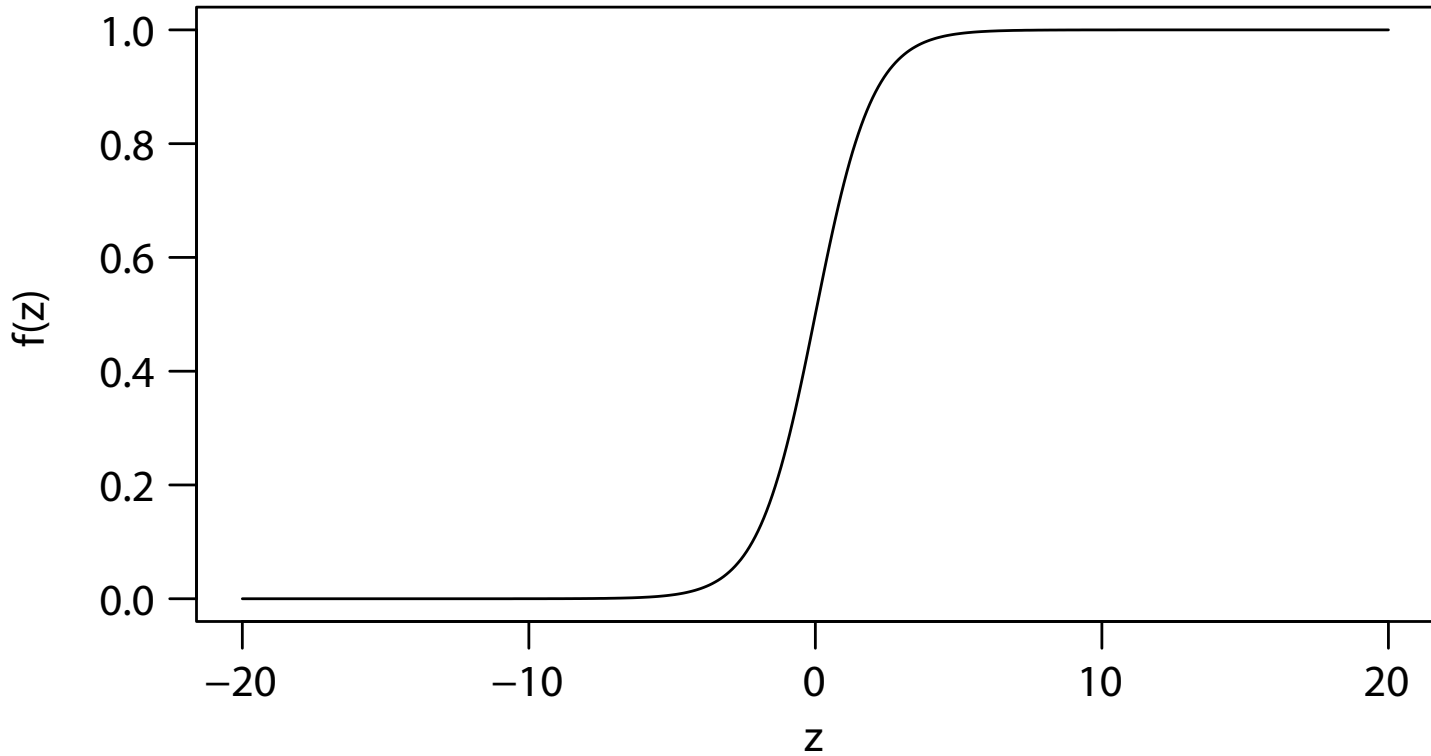
- **Logistic regression** is a binary classification model
- An auxiliary target variable z is modeled as

$$z = \sum_{j=1}^n w^j x^j + w_0 = \langle \mathbf{w}, \mathbf{x} \rangle + w_0$$

- The **logistic function** f is a mapping from \mathbb{R} to the interval $[0, 1]$:

$$f(z) = \frac{\exp(z)}{\exp(z) + 1} = \frac{1}{1 + \exp(-z)}$$

Logistic Function



Logistic Regression

- The logistic function becomes

$$f(\mathbf{x}) = \frac{1}{1 + \exp(-(\langle \mathbf{w}, \mathbf{x} \rangle + w_0))}$$

- The inverse $g = f^{-1}$ is called the **logit** or **log-odds** function:

$$g(f(\mathbf{x})) = \log\left(\frac{f(\mathbf{x})}{1 - f(\mathbf{x})}\right) = \langle \mathbf{w}, \mathbf{x} \rangle + w_0$$

- The goal of logistic regression is to estimate \mathbf{w} and w_0 from a dataset D
 - $f(\mathbf{x})$ shows probability of belonging to the class 1, thus its label $y = 1$ if $f(\mathbf{x}) \geq 0.5$

Maximum Likelihood Estimation

- The log-likelihood of the parameter (\mathbf{w}, w_o) is

$$L(\mathbf{w}, w_o) = \sum_{i=1}^N y_i \log f(\mathbf{x}_i) + (1 - y_i) \log(1 - f(\mathbf{x}_i)), \quad \mathbf{x}_i \in \mathbb{R}^n, y_i \in \{0, 1\}$$

- The objective of logistic regression is maximization of $L(\mathbf{w}, w_o)$
- The gradient w.r.t. w^j is

$$\frac{\partial L(\mathbf{w}, w_p)}{\partial w^j} = \sum_{i=1}^N (y_i - f(\mathbf{x}_i)) x_i^j$$

- Since log-likelihood is convex, it is maximized by gradient ascent

Logistic Regression by Gradient Ascent

Algorithm 2: Logistic Regression

```
1 Initialize  $\mathbf{w}$  and  $w_0$  with some values;  
2  $t \leftarrow 0$ ;  
3 repeat  
4   foreach  $j \in \{1, 2, \dots, n\}$  do  
5      $w^{j,(t+1)} \leftarrow w^{j,(t)} + \varepsilon \sum_{i=1}^N (y_i - f(\mathbf{x}_i)) x_i^j$   
6    $t \leftarrow t + 1$   
7 until  $\mathbf{w}^{(t)} = \mathbf{w}^{(t+1)}$ ;
```

Decision Tree

- **Decision tree** obtains a tree-structured classification rules by recursively partitioning data points
- In a decision tree, each node represents a binary classification rule

Algorithm 3: Decision Tree

```
1 DecisionTree( $D, \eta, \pi$ )
2   if  $|D| \leq \eta$  or  $\max_{c \in C} |D_c| / |D| \geq \pi$  then
3     create a leaf node and label it with  $\operatorname{argmax}_{c \in C} |D_c| / |D|$ 
4     return
5   ( $\text{split rule}, \text{score}^*$ )  $\leftarrow (\emptyset, 0)$ 
6   foreach  $j \in \{1, 2, \dots, n\}$  do
7     ( $v, \text{score}$ )  $\leftarrow \text{EvaluateFeature}(D, j)$ 
8     if  $\text{score} > \text{score}^*$  then ( $\text{split rule}, \text{score}^*$ )  $\leftarrow (X^j \leq v, \text{score})$ ;
9    $D_Y \leftarrow \{\mathbf{x} \in D \mid \mathbf{x} \text{ satisfies the split rule}\}; D_N \leftarrow D \setminus D_Y$ 
10  Create a node with the split rule
11  DecisionTree( $D_Y, \eta, \pi$ ); DecisionTree( $D_N, \eta, \pi$ )
```


Split Rule

- If the j th feature (variable) X^j is numeric (continuous), a split rule is in the form of " $X^j \leq v$ "
 - For a point \mathbf{x} , it is satisfied if $x^j \leq v$
- If the j th feature (variable) X^j is categorical (discrete), a split rule is in the form of " $X^j \in V$ "
 - For a point \mathbf{x} , it is satisfied if $x^j \in V$
 - Replace $X^j \leq v$ with $X^j \in V$ in the line 8 of Algorithm 3 if X^j is categorical

Split Rule Evaluation: Entropy

- **Information gain:** $\text{Gain}(D, D_Y, D_N) = H(D) - H(D_Y, D_N)$

- Entropy:

$$H(D) = - \sum_{c \in C} P_D(c) \log P_D(c)$$

- $P_D(c)$ is the probability of the class c in D
- It is larger if $P_D(c)$ is equally distributed

- Split entropy:

$$H(D_Y, D_N) = \frac{|D_Y|}{|D|} H(D_Y) + \frac{|D_N|}{|D|} H(D_N)$$

- The higher the information gain, the better the split rule

Split Rule Evaluation: Gini Index

- **Information gain:** $\text{Gain}(D, D_Y, D_N) = G(D) - G(D_Y, D_N)$

- Gini index:

$$G(D) = 1 - \sum_{c \in C} P(c | D)^2$$

- $P_D(c)$ is the probability of the class c in D
- It is larger if $P_D(c)$ is equally distributed

- Weighted Gini index:

$$G(D_Y, D_N) = \frac{|D_Y|}{|D|} G(D_Y) + \frac{|D_N|}{|D|} G(D_N)$$

- The higher the information gain, the better the split rule

Algorithm 4: Evaluate Numeric Feature

```
1 EvaluateFeatureNumeric( $D, j$ )
2   sort  $D$  on feature  $j$  as  $\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \dots, \mathbf{x}_{(N)}$  s.t.  $x_{(i)}^j \leq x_{(i+1)}^j$ 
3    $M \leftarrow \{v_1, v_2, \dots, v_{N-1}\}$  s.t.  $v_i = (x_{(i)}^j + x_{(i+1)}^j) / 2$ ; // Set of midpoints
4    $(v^*, \text{score}^*) \leftarrow (\emptyset, 0)$ 
5   foreach  $v \in M$  do
6      $D_Y \leftarrow \{(\mathbf{x}, y) \in D \mid x^j \leq v\}$ ;  $D_N \leftarrow D \setminus D_Y$ 
7     foreach  $c \in C$  do
8        $\hat{P}(c \mid D_Y) \leftarrow |D_{Y,c}| / |D_Y|$ ;  $\hat{P}(c \mid D_N) \leftarrow |D_{N,c}| / |D_N|$ 
9        $\text{score} \leftarrow \text{Gain}(D, D_Y, D_N)$ 
10      if  $\text{score} > \text{score}^*$  then  $(v^*, \text{score}^*) \leftarrow (v, \text{score})$ ;
11  return  $(v^*, \text{score}^*)$ 
```

Algorithm 5: Evaluate Categorical Feature

```
1 EvaluateFeatureCategorical( $D, j$ )
2    $(v^*, \text{score}^*) \leftarrow (\emptyset, 0)$ 
3   foreach  $V \subseteq \Sigma^j$  do
4      $D_Y \leftarrow \{(\mathbf{x}, y) \in D \mid x^j \in V\}; D_N \leftarrow D \setminus D_Y$ 
5     foreach  $c \in C$  do
6        $\hat{P}(c \mid D_Y) \leftarrow |D_{Y,c}| / |D_Y|; \hat{P}(c \mid D_N) \leftarrow |D_{N,c}| / |D_N|$ 
7        $\text{score} \leftarrow \text{Gain}(D, D_Y, D_N)$ 
8       if  $\text{score} > \text{score}^*$  then  $(V^*, \text{score}^*) \leftarrow (V, \text{score});$ 
9   return  $(V^*, \text{score}^*)$ 
```

Random Forest

- To avoid overfitting, **ensemble of decision trees** can be used
- Breiman (2001) introduced **random forests**, a collection of decision trees
 - This method is known to be effective in practice
- Subsample a dataset (N' points and n' features) t times
- Construct a decision tree for each subsampled dataset
- Classification is performed by taking a majority vote across the trees

Summary

- **Naïve Bayes** classifier perform classification using the Bayes theorem
 - Assumption: Features are independent
- **kNN** is a non-parametric classification method
- **Logistic regression** is easy to fit and interpret
- **Decision tree** can obtain interpretable classification rules