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 (杉山麿人)

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 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_i \in \mathbb{R}^n$ (feature vector), $y_i \in C = \{c_1, c_2, \ldots, c_K\}$ (label)

- The Bayes approach:
  Estimate the posterior probability $P(c \mid \boldsymbol{x})$ from data and predict the class y of $\boldsymbol{x}$ as $\hat{y} = \operatorname{argmax}_{c \in C} P(c \mid \boldsymbol{x})$

Bayes Classification

- Use the Bayes theorem:

$$P(c \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid c) \cdot P(c)}{P(\boldsymbol{x})}$$

  - $P(c \mid \boldsymbol{x})$: posterior, $P(\boldsymbol{x} \mid c)$: likelihood, $P(c)$: prior
  - $P(\boldsymbol{x}) = \sum_{c \in C} P(\boldsymbol{x} \mid c) \cdot P(c)$

- Since the denominator $P(\boldsymbol{x})$ is independent of classes c (just a normalizing constant),

$$\hat{y} = \underset{c \in C}{\operatorname{argmax}}\, P(c \mid \boldsymbol{x}) = \underset{c \in C}{\operatorname{argmax}}\, P(\boldsymbol{x} \mid c) P(c)$$

Prior Probability Estimation

- **Goal**: Estimate the prior $P(c)$ from a dataset D

- For a given dataset D, for each class $c \in C$,
  $D_c = \{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in D \text{ and } y = c\}$

- 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 $P(\boldsymbol{x} \mid c)$ from a dataset D
- Assume that each feature is independent (the model is "naïve"):
  $P(\boldsymbol{x} \mid c) = \prod_{j=1}^{n} P(x^j \mid c), \quad \boldsymbol{x} = (x^1, x^2, \ldots, x^n)$
- For each $j \in \{1, 2, \ldots, n\}$, if we assume data is normally distributed,

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

$$P(\boldsymbol{x} \mid c) = \prod_{j=1}^{n} P(x^j \mid c) \propto \prod_{j=1}^{n} f(x^j; \mu_c^j, {\sigma_c^j}^2)$$

Algorithm 1: Naïve Bayes Classifier

learn(D)

- **foreach** $c \in C$ **do**
  - $D_c \leftarrow \{\boldsymbol{x} \mid (\boldsymbol{x}, c) \in D\}$
  - $\hat{P}(c) \leftarrow |D_c| \,/\, |D|$
  - **foreach** $j \in \{1, 2, \ldots, n\}$ **do**
    - $\hat{\mu}_c^j \leftarrow (1/|D_c|) \sum_{\boldsymbol{x} \in D_c} x^j$
    - ${\hat{\sigma}_c^j}^2 \leftarrow (1/|D_c|) \sum_{\boldsymbol{x} \in D_c} (x^j - \hat{\mu}_c^j)^2$

classify($\boldsymbol{x}$)

- $\hat{y} \leftarrow \mathrm{argmax}_{c \in C}\, \hat{P}(c) \prod_{j=1}^{n} f(x^j; \hat{\mu}_c^j, {\hat{\sigma}_c^j}^2)$

If Features Are Categorical

- Assume that the domain of jth 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

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

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

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

kNN approach

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

Logistic Regression

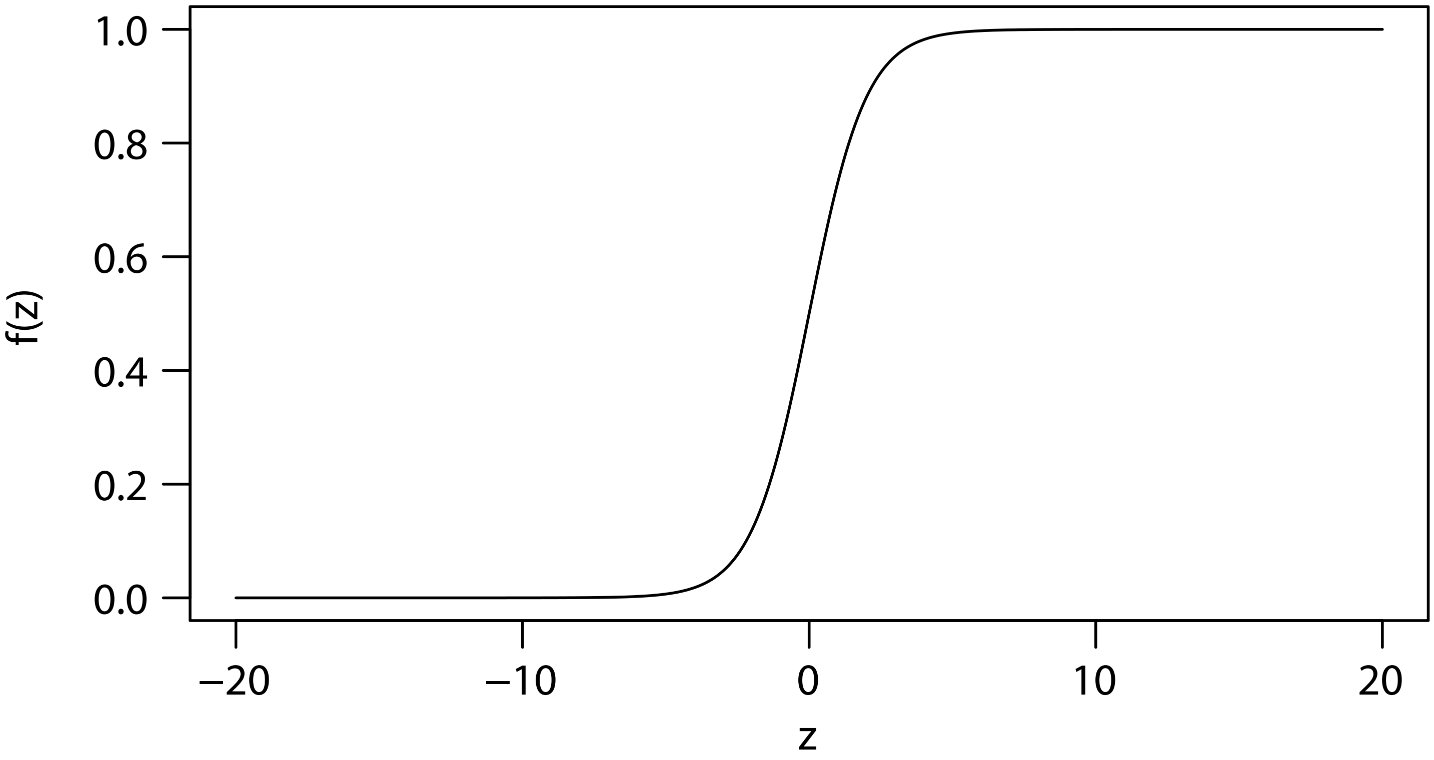
- Logistic regression is a binary classification model
- An auxiliary target variable z is modeled as

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

- 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

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

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

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

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

Maximum Likelihood Estimation

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

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

  - The objective of logistic regression is maximization of $L(\boldsymbol{w}, w_o)$

- The gradient w.r.t. w^j is

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

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

Logistic Regression by Gradient Ascent

Algorithm 2: Logistic Regression

Initialize $\boldsymbol{w}$ and w_0 with some values;
$t \leftarrow 0$;
repeat
    foreach $j \in \{1, 2, \ldots, n\}$ **do**
        $w^{j,(t+1)} \leftarrow w^{j,(t)} + \varepsilon \sum_{i=1}^{N} \Big(y_i - f(\boldsymbol{x}_i) \Big) x_i^j$
    $t \leftarrow t + 1$
until $\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t+1)}$;

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

DecisionTree(D, η, π)

    if $|D| \leq \eta$ *or* $\max_{c \in C} |D_c| \,/\, |D| \geq \pi$ **then**

        create a leaf node and label it with $\text{argmax}_{c \in C} |D_c| \,/\, |D|$

        return

    (split rule, score*) $\leftarrow (\emptyset, 0)$

    foreach $j \in \{1, 2, \ldots, n\}$ **do**

        (v, score) $\leftarrow$ EvaluateFeature(D, j)

        if *score* $>$ *score** **then** (split rule, score*) $\leftarrow$ ($X^j \leq v$, score) ;

    $D_Y \leftarrow \{\boldsymbol{x} \in D \mid \boldsymbol{x}$ satisfies the split rule $\}$; $D_N \leftarrow D \setminus D_Y$

    Create a node with the split rule

    DecisionTree(D_Y, η, π); DecisionTree(D_N, η, π)

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 $\boldsymbol{x}$, 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 $\boldsymbol{x}$, 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:

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

    - $P_D(c)$ 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

EvaluateFeatureNumeric(D, j)

  sort D on feature j as $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(N)}$ s.t. $x^j_{(i)} \leq x^j_{(i+1)}$

  $M \leftarrow \{v_1, v_2, \ldots, v_{N-1}\}$ s.t. $v_i = (x^j_{(i)} + x^j_{(i)}) \,/\, 2$; // Set of midpoints

  $(v^*, \text{score}^*) \leftarrow (\varnothing, 0)$

  foreach $v \in M$ **do**

    $D_Y \leftarrow \{(\boldsymbol{x}, y) \in D \mid x^j \leq v\}$; $D_N \leftarrow D \setminus D_Y$

    foreach $c \in C$ **do**

      $\hat{P}(c \mid D_Y) \leftarrow |D_{Y,c}| \,/\, |D_Y|$; $\hat{P}(c \mid D_N) \leftarrow |D_{N,c}| \,/\, |D_N|$

    $\text{score} \leftarrow \text{Gain}(D, D_Y, D_N)$

    if *score* > *score** **then** $(v^*, \text{score}^*) \leftarrow (v, \text{score})$;

  return (v^*, score^*)

Algorithm 5: Evaluate Categorical Feature

```
EvaluateFeatureCategorical(D, j)
    (v*, score*) ← (∅, 0)
    foreach V ⊆ Σ^j do
        D_Y ← {(x, y) ∈ D | x^j ∈ V}; D_N ← D \ D_Y
        foreach c ∈ C do
            P̂(c | D_Y) ← |D_{Y,c}| / |D_Y|; P̂(c | D_N) ← |D_{N,c}| / |D_N|
        score ← Gain(D, D_Y, D_N)
        if score > score* then (V*, score*) ← (V, score) ;
    return (V*, 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
- *k*NN is a non-parametric classification method
- Logistic regression is easy to fit and interpret
- Decision tree can obtain interpretable classification rules