06_Naïve_Bayes_Classifier

Outline

1. Learn training data to acquire Joint Probability Distribution-JPD:

$$P(X, Y) = P(Y)\, P(X \mid Y)$$

2. Law of total probability:

$$P(X) = \sum_{i=1}^{n} P(Y_i) P(X \mid Y_i)$$

3. Predict classification via Bayes Theorem and learnt JPD. Pick the class y that maximize $P(Y \mid X)$ to get Naïve Baye Classifier Equation y_{MAP}:

$$P(Y \mid X) = \frac{P(X, Y)}{P(X)} = \frac{P(Y)\, P(X \mid Y)}{\sum_{y_i \in Y} P(y_i) P(X \mid y_i)}$$

$$y_{MAP} = argmax_{y \in Y} P(y = c_k) \prod_{j=1}^{m} P(x_i = x_j | y = c_k)$$

Naïve Bayes Classifier

- Adopt $\mathit{Attribute\ Conditional\ Independence\ Assumption}$
  (All attributes regarded as conditionally independent)

- Instead of modelling one D-dimensional distribution:
  $P(x_1,\ x_2,\ x_3\ \dots \mid y)$
  Model D one-dimensional distributions:
  $P(x_1 | \ y),\ \ P(x_2 | \ y),\ \ P(x_3 | \ y) \dots P(x_n | \ y)$

  Thus, $P(\vec{x} \mid y) = P(x_1,\ x_2,\ x_3\ \dots x_n \mid y) = \prod_{i=1}^{n} P(x_i|y)$
  Efficient, simplify computation of classification but sacrifice a sort of accuracy. Often violated but it works surprisingly well. Since dependencies ignored, Naïve Bayes posteriors often unrealistically close to 0 or 1.
  $\mathit{BAD\ EFFECT\ when: comparatively\ more\ attributes,\ high\ correlation\ among\ attributes}$

Training data:

$Input: \vec{X} = (\vec{x}_1,\ \vec{x}_2,\ \vec{x}_3\ \dots\ \vec{x}_n)\ ,\quad \vec{x}_i : i_{th}\ attribute. \vec{x}_i = (x_{i1}, x_{i2}, \dots x_{im})$

$\quad \vec{x}_{ij} : j_{th}\ value\ of\ i_{th}\ attribute.$

$Output:\ class\ label\ y \in Y,\ \ Y = \{c_1, c_2, c_3 \dots c_k\},$

$Training\ process:$

$$P(y) = \frac{|D_y|}{|D|}$$

$$For\ discrete\ attribute: P(x_i \mid y = c_k) = \frac{|D_{y}, x_i|}{|D_y|}$$

$$For\ continuous\ attribute: P(x_i \mid y) = \frac{1}{\sqrt{2\pi}\sigma} exp\,(-\frac{(x_i - \mu)^2}{2\sigma^2})\quad (multivariate\ Gaussian)$$

$$P(x_i \mid y) \sim N(\mu, \sigma^2)\ \ \mu\ \&\ \sigma^2 : mean\ \&\ variance\ of\ i_th\ attribute\ in\ class\ y$$

MAP: $P(y \mid \vec{x}) = \frac{P(\vec{x} \mid y)\, P(y)}{P(\vec{x})} = \frac{P(y)}{P(\vec{x})} \prod_{i=1}^{n} P(\vec{x}_i \mid y)\quad (help\ understand\ NB\ Classifier)$

Naïve Bayes Classifier Equation:

$$y_{MAP} = argmax_{y \in Y} P(y = c_k) \prod_{j=1}^{n} P(x_i = x_{ij} | y = c_k)$$

- e.g.
  Given a training dataset X_i with corresponding Y_k, train the dataset to get $P(x_i \mid y_k)$. Classify new data $X(x_1 = x_{14}, x_3 = x_{35})$.
  $y_{MAP} = argmax_y \ P(y = c_k) * (P(x_1 = x_{14} \mid y = c_k) * P(x_1 = x_{14} \mid y = c_k)$. New data X should be class of y_k that maximize y_{MAP}.

Empirical Risk Minimization, i.e. maximize posterior: $New\ data\ X\ is\ classified\ to\ y\ that\ maximizes\ the\ function\ y_{MAP}$

Laplacian Correction (smoothing):

If none of the training instances with target value y has attribute x_i i.e. $P(y) \prod_{i=1}^{n} P(x_i|y) = 0$,which will influence MAP and get deviation. Implement Laplacian correction to 'smooth'. Note: $positive\ integer\ \lambda\ ,\ usually\ \lambda = 1$

For discrete attributes:

$$P(y = c_k) = \frac{|D_{y=c_k}| + \lambda}{|D| + k\lambda}$$
$$P(x_i \mid y = c_k) = \frac{|D_y, x_i| + \lambda}{|D_y| + j\lambda}$$

When to use:

- Moderate or large training set available.
- Features x_i of a data instance $\vec{x}$ are conditionally independent given classification (or at least reasonably independent, stillworks with a little dependence).