

1 Naive Bayes Classifier

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong, naive *independence assumptions* between the features.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of features.

It uses the concept of *probability* to classify new instances.

1.1 Description

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction.

Naive Bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.

The probability of a class value given a value of an attribute is called the conditional probability.

By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class.

To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

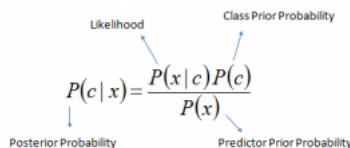
1.2 Naive Bayes probabilistic model

Naive Bayes is a **conditional probability model**.

It answers the question "what is the probability that something will happen, given that something else has already happened?".

Given a problem instance to be classified, represented by a vector $x = \{x_1, \dots, x_n\}$ with n features (independent variables), it assigns to it class probabilities $p(C_k | x_1, \dots, x_n)$.

Using **Bayes' theorem** $P(A | B) = \frac{P(B|A)P(A)}{P(B)}$ the conditional probability can be decomposed as $p(C_k | x) = \frac{p(C_k) p(x|C_k)}{p(x)}$. In other words: $\text{posterior} = \frac{\text{prior} \cdot \text{likelihood}}{\text{evidence}}$


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

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

In practice the denominator is ignored, given it's constant.

The numerator is equivalent to the joint probability model $p(C_k, x_1, \dots, x_n)$, which can be rewritten using the **chain rule**

$$P(A_n, \dots, A_3, A_2, A_1) = P(A_n | A_{n-1}, \dots, A_3, A_2, A_1) \dots P(A_3 | A_2, A_1) P(A_2 | A_1) P(A_1)$$

as

$$p(C_k, x_1, \dots, x_n) = p(x_1, \dots, x_n, C_k) = p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) = \dots = p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k)$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C_k .

Each piece of evidence is hence treated as independent.

This means that $p(x_j | x_{j+1}, \dots, x_n, C_k) = p(x_j | C_k)$

Thus, the joint model can be expressed as

$$p(C_k | x_1, \dots, x_n) \propto p(C_k, x_1, \dots, x_n) = p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots p(x_n | C_k) = p(C_k) \prod_{j=1}^n p(x_j | C_k)$$

Notes

- probability model: $P(C_k | x) = p(C_k) \prod_{j=1}^n p(x_j | C_k)$
- in [plain English](#): $P(\text{outcome} | \text{multiple evidence}) = P(\text{evidence}_1 | \text{outcome}) P(\text{evidence}_2 | \text{outcome}) \dots P(\text{evidence}_n | \text{outcome}) P(\text{outcome})$
- if $P(\text{evidence}_j | \text{outcome}) = 0$, then the whole probability becomes 0; contradicting evidence rules out the outcome
- the intuition behind multiplying by the *prior* is to give high probability to more common outcomes, and low probabilities to unlikely outcomes; these are also called *base rates* and they are a way to scale predicted probabilities

1.3 Naive Bayes classifier

The naive Bayes classifier combines the **naive Bayes probabilistic model** with a **decision rule**.

Possible decision rules

- maximum likelihood (ML)

Find the parameter values that maximize the likelihood function.

$$\hat{y} = \underset{y}{\operatorname{argmax}} \prod_{j=1}^n p(x_j | y)$$

- maximum a posteriori (MAP)

Pick the hypothesis that is **most probable**.

$$\hat{y} = \underset{y}{\operatorname{argmax}} p(y) \prod_{j=1}^n p(x_j | y)$$

- recalibrated likelihood

Replace the class distribution with a set of weights learned from the data.

$$\hat{y} = \underset{y}{\operatorname{argmax}} w_y \prod_{j=1}^n p(x_j | y)$$

The ML classification is equivalent to the MAP classification with a uniform class distribution.

1.4 Parameter estimation

A class's prior may be calculated by

- assuming equiprobable classes: $p(c_k) = \frac{1}{K}$
- calculating an estimate for the class probability from the training set: $p(c_k) = \frac{\text{\# samples in the class } c_k}{\text{\# samples}}$

To estimate the parameters for a feature's distribution $p(x|c_k)$, one must assume a distribution or generate non-parametric models for the features from the training set.

The **assumptions on distributions** of features are called the event model of the Naive Bayes classifier.

For discrete features, multinomial and Bernoulli distributions are popular.

For continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

1.4.1 Gaussian naive Bayes

Implements the Gaussian Naive Bayes algorithm for classification.

The likelihood of the features is assumed to be Gaussian

$$p(x_j | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_j - \mu_y)^2}{2\sigma_y^2}\right)$$

Estimate the parameters σ_y and μ_y for each feature and for each class from the training set.

1.4.2 Multinomial naive Bayes

MultinomialNB implements the naive Bayes algorithm for multinomially distributed data.

The distribution is parametrized by vectors $\theta_y = \{\theta_{y_1}, \dots, \theta_{y_n}\}$ for each class y , where $\theta_{y_j} = p(x_j | y)$ of feature j appearing in a sample belonging to class y .

The parameters θ_y are estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{y_j} = \frac{N_{y_j} + \alpha}{N_y + \alpha n}$$

where $N_{y_j} = \sum_{x \in T} x_j$ is the number of times feature j appears in a sample of class y in the training set T , and $N_y = \sum_{j=1}^{|T|} N_{y_j}$ is the total count of all features for class y .

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations.

Setting $\alpha = 1$ is called *Laplace smoothing*, while $\alpha < 1$ is called *Lidstone smoothing*.

1.4.3 Bernoulli naive Bayes

BernoulliNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable.

The decision rule for Bernoulli naive Bayes is based on

$$P(x_j | y) = P(j | y)x_j + (1 - P(j | y))(1 - x_j)$$

which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y , where the multinomial variant would simply ignore a non-occurring feature.

1.5 Algorithm for GaussianNB

- data description
 - m input variables $x^{(i)}$ represented by n features: $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}\}$
 - each input variable has a target variable $y^{(i)}$ which has one of K possible categories
- model description
 - probabilistic model: $P(y | x) = p(y) \prod_{j=1}^n p(x_j | y)$
 - MAP hypothesis: $\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} p(y) \prod_{j=1}^n p(x_j | y)$
 - Gaussian feature distribution $P(x_j | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_j - \mu_y)^2}{2\sigma_y^2}\right)$
- algorithm
 - separate data by class

- compute the class prior probability for each class
- compute the mean and variance values of each feature and each class
- make predictions on new examples
 - compute the probability that the given instance belong to each class
 - select the class with the largest probability

1.6 Pros and Cons of Naive Bayes

Pros

- it is easy and fast to predict class of test data set
- it perform well in multi class prediction
- it works well with highly sparsed datasets
- requires a small amount of training data to estimate the necessary parameters
- when the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression and needs less training data
- it perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons

- if categorical variable has a category (in test data set), which was not observed in training data set, then model will assign it a 0 probability and will not be able to make a prediction. This is often known as *zero frequency*. Using a smoothing technique (e.g. Laplace) can solve this problem
- naive Bayes is known as a bad estimator, so the probability outputs are not to be taken too seriously
- in real life, it is almost impossible that we get a set of predictors which are completely independent