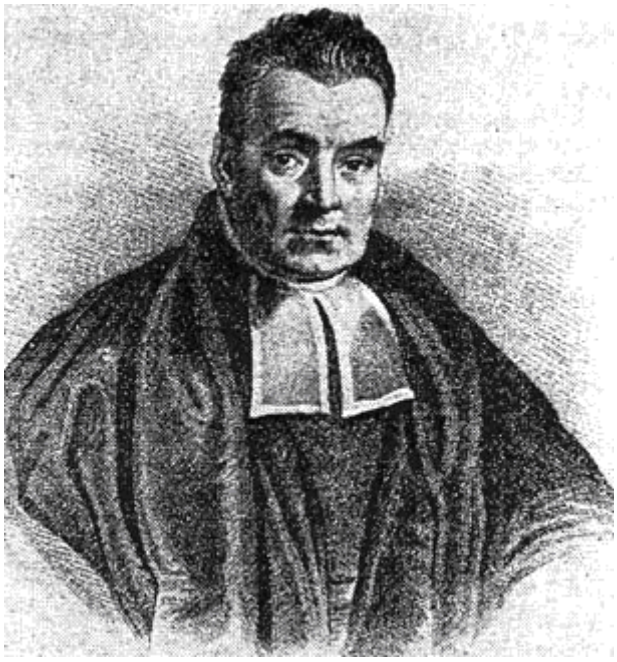
Naive Bayes Explained

Zixuan Zhang
Aug 14 · 5 min read ★

Naive Bayes is a probabilistic algorithm that's typically used for classification problems. Naive Bayes is simple, intuitive, and yet performs surprisingly well in many cases. For example, spam filters Email app uses are built on Naive Bayes. In this article, I'll explain the rationales behind Naive Bayes and build a spam filter in Python. (For simplicity, I'll focus on binary classification problems)



Thomas Bayes, the 'betting man', from BBC

Theory

Before we get started, please memorize the notations used in this article:

$$X = (X_1, X_2, \dots, X_k)\ represent\ k\ features. Y\ is\ the\ label\ with\ K\ possible\ values\ (class)$$

$$From\ a\ probabilistic\ perspective, X_i \in X\ and\ Y\ are\ random\ variables$$

$$The\ value\ of\ X_i\ is\ x, Y\ is\ y.$$

Basic Idea

To make classifications, we need to use X to predict Y. In other words, given a data point X=(x1,x2,…,xn), what the odd of Y being y. This can be rewritten as the following equation:

$$P(Y = y|X = (x_1, x_2, \dots x_k))$$

$$Find\ the\ most\ likely\ y \rightarrow prediction = argmax_y P(Y = y|X = (x_1, x_2, \dots x_k))$$

This is the basic idea of Naive Bayes, the rest of the algorithm is really more focusing on how to calculate the conditional probability above.

Bayes Theorem

So far Mr. Bayes has no contribution to the algorithm. Now is his time to shine. According to the Bayes Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \rightarrow Posterior = \frac{likelihood \times prior}{evidence}$$

Bayes Theorem

This is a rather simple transformation, but it bridges the gap between what we want to do and what we can do. We can't get P(Y|X) directly, but we can get P(X|Y) and P(Y) from the training data. Here's an example:

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	NO
sunny	hot	high	true	NO
overcast	hot	high	false	YES
rainy	mild	high	false	YES
rainy	cool	normal	false	YES
rainy	cool	normal	true	NO
overcast	cool	normal	true	YES
sunny	mild	high	false	NO
sunny	cool	normal	false	YES

rainy	mild	normal	false	YES
sunny	mild	normal	true	YES
overcast	mild	high	true	YES
overcast	hot	normal	false	YES
rainy	mild	high	true	NO

Weather dataset, from the University of Edinburgh

In this case, X =(Outlook, Temperature, Humidity, Windy), and Y=Play. P(X|Y) and P(Y) can be calculated:

$$P(Y = Play) = \frac{\#Play}{\#Play + \#\ not\ Play} = \frac{9}{14}$$

$$P(X = (sunny)|Y = Play) = \frac{\#Play\ \&\ Sunny}{\#Play} = \frac{2}{9}$$

Example of finding P(Y) and P(X|Y)

Naive Bayes Assumption and Why

Theoretically, it is not hard to find P(X|Y). However, it is much harder in reality as the number of features grows.

$X_1 = 1$	$X_2 = 1$	$Y = 1$	$P(X_1 = 1, X_2 = 1 \mid Y = 1) = ?$
		$Y = 0$	$P(X_1 = 1, X_2 = 1 \mid Y = 0) = ?$
	$X_2 = 0$	$Y = 1$	$P(X_1 = 1, X_2 = 0 \mid Y = 1) = ?$
		$Y = 0$	$P(X_1 = 1, X_2 = 0 \mid Y = 0) = ?$
$X_1 = 0$	$X_2 = 1$	$Y = 1$	$P(X_1 = 0, X_2 = 1 \mid Y = 1) = ?$
		$Y = 0$	$P(X_1 = 0, X_2 = 1 \mid Y = 0) = ?$
	$X_2 = 0$	$Y = 1$	$P(X_1 = 0, X_2 = 0 \mid Y = 1) = ?$
		$Y = 0$	$P(X_1 = 0, X_2 = 0 \mid Y = 0) = 1 - others$

7 parameters are needed for a 2-feature binary dataset

$$P(X = (x_1, x_2, \dots, x_k)|Y = y)\ is\ called\ a\ parameter\ in\ Naive\ Bayes.$$

$$If\ x_j\ \forall j\ is\ binary, there\ are\ 2^{k+1} - 1\ parameters$$

Estimate Join Distribution requires more data

Having this amount of parameters in the model is impractical. To solve this problem, a naive assumption is made. **We pretend all features are independent.** What does this mean?

$$if\ X_1\ and\ X_2\ are\ indep. P(X_1, X_2|Y) = P(X_1|Y)P(X_2|Y)$$

Conditional independence

Now with the help of this naive assumption (naive because features are rarely independent), **we can make classification with much fewer parameters**:

$$P(Y|X) = \frac{P(Y)\prod_i P(X_i|Y)}{P(X)}$$

Naive Bayes Classifier

$X_1 = 1$	$Y = 1$	$P(X_1 = 1\|Y = 1)$
	$Y = 0$	$P(X_1 = 1\|Y = 0)$
$X_1 = 0$	$Y = 1$	$P(X_1 = 0\|Y = 1)$ $= 1 - P(X_1 = 0\|Y = 1)$
	$Y = 0$	$P(X_1 = 0\|Y = 0)$ $= 1 - P(X_1 = 1\|Y = 0)$

$X_2 = 1$	$Y = 1$	$P(X_2 = 1\|Y = 1)$
	$Y = 0$	$P(X_2 = 1\|Y = 0)$
$X_2 = 0$	$Y = 1$	$P(X_2 = 0\|Y = 1)$ $= 1 - P(X_2 = 0\|Y = 1)$
	$Y = 0$	$P(X_2 = 0\|Y = 0)$ $= 1 - P(X_2 = 1\|Y = 0)$

Naive Bayes need fewer parameters (4 in this case)

$$If\ x_j\ \forall j\ is\ binary, there\ are\ 2k\ parameters$$

This is a big deal. We changed the number of parameters from exponential to linear. This means that Naive Bayes handles high-dimensional data well.

Categorical And Continuous Features

Categorical Data

For categorical features, the estimation of P(Xi|Y) is easy.

$$P(X_i = x|Y = y) = \frac{\#(X_i = x, Y = y)}{\#Y = y}$$

Calculate the likelihood of categorical features

However, one issue is that if some feature values never show (maybe lack of data), their likelihood will be zero, which makes the whole posterior probability zero. One simple way to fix this problem is called Laplace Estimator: add imaginary samples (usually one) to each category

$$P(X_i = x|Y = y) = \frac{\#(X_i = x, Y = y) + 1}{\#Y = y + m}, m = \ \#\ of\ data\ points\ such\ that\ X_i = X, Y = y$$

Laplace Estimator

Continuous Data

For continuous features, there are essentially two choices: discretization and continuous Naive Bayes.

Discretization works by breaking the data into categorical values. The simplest discretization is uniform binning, which creates bins with fixed range. There are, of course, smarter and more complicated ways such as Recursive minimal entropy partitioning or SOM based partitioning.

	GLOBAL	LOCAL
SUPERVISED	Recursive Minimal Entropy Partitioning	Hierarchical Maximum Entropy
UNSUPERVISED	Equal Width Interval, Equal Frequency Interval	K-means Clustering

Discretizing Continuous Feature for Naive Bayes

The second option is utilizing known distributions. If the features are continuous, the Naive Bayes algorithm can be written as:

$$P(Y|X) = \frac{P(Y)\prod_i f(X_i|Y)}{f(X)}$$

f is the probability density function

For instance, if we visualize the data and see a bell-curve-like distribution, it is fair to make an assumption that the feature is normally distributed

The first step is calculating the mean and variance of the feature for a given label y:

$$S: data\ points\ with\ Y = y \rightarrow \mu' = \frac{\sum^S X_i}{len(S)}, \sigma'^2 = \frac{1}{m-1}\sum^S (X_i - \mu')^2$$

variance adjusted by the degree of freedom

Now we can calculate the probability density f(x):

$$f(X_i = x|Y = y) = \frac{1}{\sqrt{2\pi\sigma'^2}} e^{\frac{(x-\mu')^2}{2\sigma'^2}}$$

There are, of course, other distributions:

1.9. Naive Bayes

1.9.1. Gaussian Naive Bayes

1.9.2. Multinomial Naive Bayes

1.9.3. Complement Naive Bayes

1.9.4. Bernoulli Naive Bayes

Naive Bayes, from Scikit-Learn

Although these methods vary in form, the core idea behind is the same: assuming the feature satisfies a certain distribution, estimating the parameters of the distribution, and then get the probability density function.

Strength and Weakness

Strength

1. Even though the naive assumption is rarely true, the algorithm performs surprisingly good in many cases
2. Handles high dimensional data well. Easy to parallelize and handles big data well
3. Performs better than more complicated models when the data set is small

Weakness

1. The estimated probability is often inaccurate because of the naive assumption. Not ideal for regression use or probability estimation
2. When data is abundant, other more complicated models tend to outperform Naive Bayes

Summary

Naive Bayes utilizes the most fundamental probability knowledge and makes a naive assumption that all features are independent. Despite the simplicity (some may say oversimplification), Naive Bayes gives a decent performance in many applications.

Now you understand how Naive Bayes works, it is time to try it in real projects!

)

Machine Learning Data Science Algorithms

About Help Legal