Naive Bayes for Machine Learning

by **Jason Brownlee** on April 11, 2016 in **Machine Learning Algorithms**

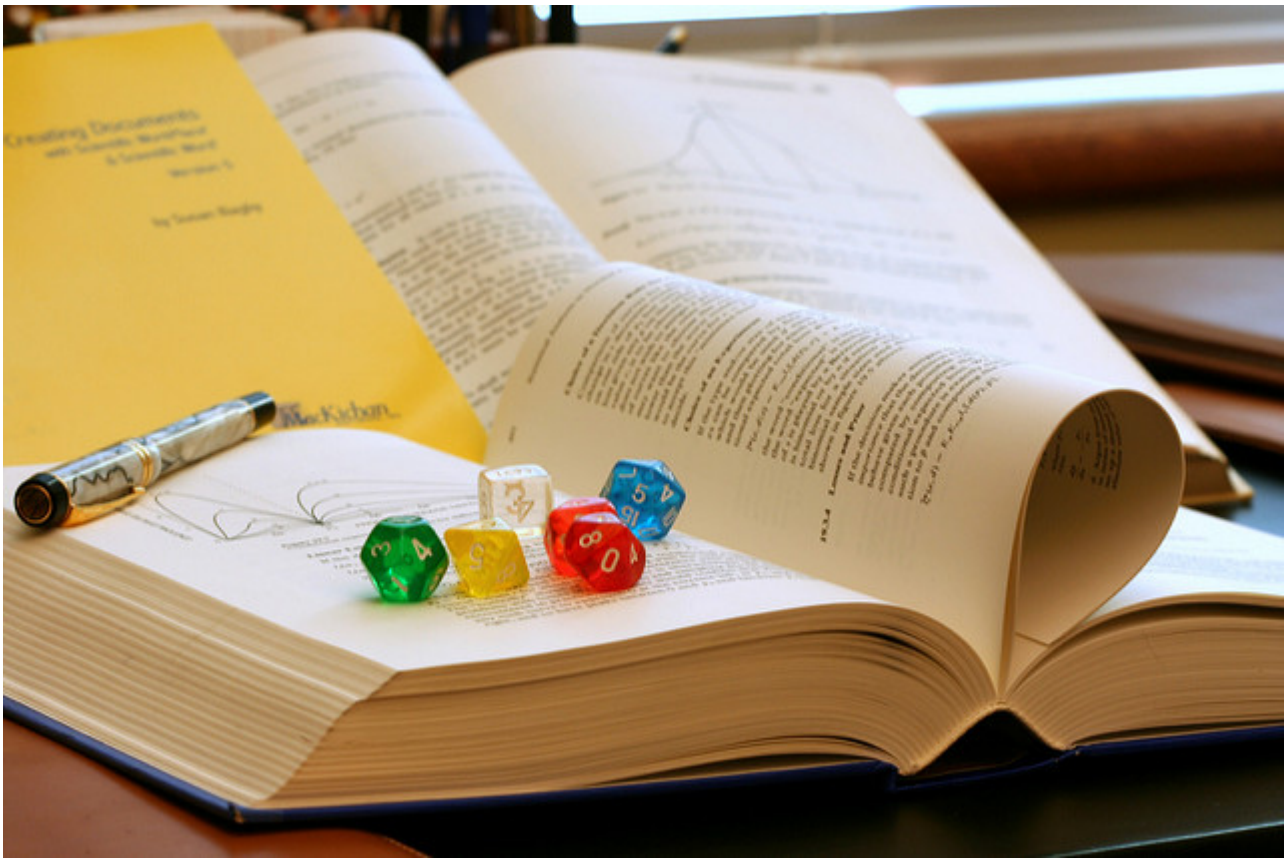
Naive Bayes is a simple but surprisingly powerful algorithm for predictive modeling.

In this post you will discover the Naive Bayes algorithm for classification. After reading this post, you will know:

- The representation used by naive Bayes that is actually stored when a model is written to a file.
- How a learned model can be used to make predictions.
- How you can learn a naive Bayes model from training data.
- How to best prepare your data for the naive Bayes algorithm.
- Where to go for more information on naive Bayes.

This post is written for developers and does not assume any background in statistics or probability, although knowing a little probability wouldn't hurt.

Let's get started.



Naive Bayes for Machine Learning
Photo by John Morgan, some rights reserved.

Quick Introduction to Bayes' Theorem

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes' Theorem is stated as:

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

Where

- **P(h|d)** is the probability of hypothesis h given the data d. This is called the posterior probability.
- **P(d|h)** is the probability of data d given that the hypothesis h was true.
- **P(h)** is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- **P(d)** is the probability of the data (regardless of the hypothesis).

You can see that we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h).

After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis.

This can be written as:

$$MAP(h) = max(P(h|d))$$

or

$$MAP(h) = max((P(d|h) * P(h)) / P(d))$$

or

$$MAP(h) = max(P(d|h) * P(h))$$

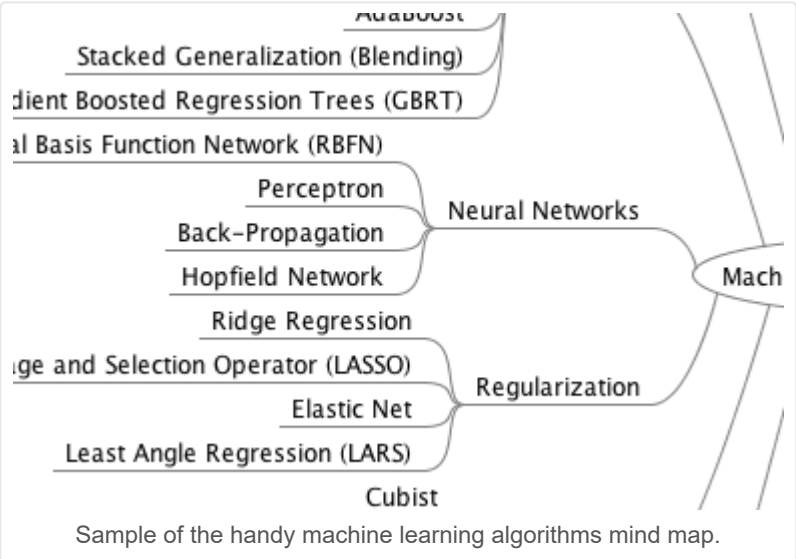
The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize.

Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation and we could drop it so that we end up with:

$$MAP(h) = max(P(d|h))$$

This is a useful exercise, because when reading up further on Naive Bayes you may see all of these forms of the theorem.

Get your FREE Algorithms Mind Map



Sample of the handy machine learning algorithms mind map.

I've created a handy mind map of 60+ algorithms organized by type.

Download it, print it and use it.

Download For Free

Also get exclusive access to the machine learning algorithms email mini-course.

Naive Bayes Classifier

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values.

It is called *naive Bayes* or *idiot Bayes* because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on.

This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

Representation Used By Naive Bayes Models

The representation for naive Bayes is probabilities.

A list of probabilities are stored to file for a learned naive Bayes model. This includes:

- **Class Probabilities**: The probabilities of each class in the training dataset.
- **Conditional Probabilities**: The conditional probabilities of each input value given each class value.

Learn a Naive Bayes Model From Data

Learning a naive Bayes model from your training data is fast.

Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

Calculating Class Probabilities

The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances.

For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:

P(class=1) = count(class=1) / (count(class=0) + count(class=1))

In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

Calculating Conditional Probabilities

The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value.

For example, if a "*weather*" attribute had the values "*sunny*" and "*rainy*" and the class attribute had the class values "*go-out*" and "*stay-home*", then the conditional probabilities of each weather value for each class value could be calculated as:

- P(weather=sunny|class=go-out) = count(instances with weather=sunny and class=go-out) / count(instances with class=go-out)
- P(weather=sunny|class=stay-home) = count(instances with weather=sunny and class=stay-home) / count(instances with class=stay-home)
- P(weather=rainy|class=go-out) = count(instances with weather=rainy and class=go-out) / count(instances with class=go-out)
- P(weather=rainy|class=stay-home) = count(instances with weather=rainy and class=stay-home) / count(instances with class=stay-home)

Make Predictions With a Naive Bayes Model

Given a naive Bayes model, you can make predictions for new data using Bayes theorem.

$$MAP(h) = max(P(d|h) * P(h))$$

Using our example above, if we had a new instance with the *weather* of *sunny*, we can calculate:

$$go\text{-}out = P(weather=sunny|class=go\text{-}out) * P(class=go\text{-}out)$$
$$stay\text{-}home = P(weather=sunny|class=stay\text{-}home) * P(class=stay\text{-}home)$$

We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:

$$P(go\text{-}out|weather=sunny) = go\text{-}out / (go\text{-}out + stay\text{-}home)$$
$$P(stay\text{-}home|weather=sunny) = stay\text{-}home / (go\text{-}out + stay\text{-}home)$$

If we had more input variables we could extend the above example. For example, pretend we have a "*car*" attribute with the values "*working*" and "*broken*". We can multiply this probability into the equation.

For example below is the calculation for the "go-out" class label with the addition of the car input variable set to "working":

$$go\text{-}out = P(weather=sunny|class=go\text{-}out) * P(car=working|class=go\text{-}out) * P(class=go\text{-}out)$$

Gaussian Naive Bayes

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution.

This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

Representation for Gaussian Naive Bayes

Above, we calculated the probabilities for input values for each class using a frequency. With real-valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution.

This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

Learn a Gaussian Naive Bayes Model From Data

This is as simple as calculating the mean and standard deviation values of each input variable (x) for each class value.

$$mean(x) = 1/n * sum(x)$$

Where n is the number of instances and x are the values for an input variable in your training data.

We can calculate the standard deviation using the following equation:

$$standard\ deviation(x) = sqrt(1/n * sum(xi-mean(x)^2))$$

This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, sqrt() is the square root function, sum() is the sum function, xi is a specific value of the x variable for the i'th instance and mean(x) is described above, and ^2 is the square.

Make Predictions With a Gaussian Naive Bayes Model

Probabilities of new x values are calculated using the Gaussian Probability Density Function (PDF).

When making predictions these parameters can be plugged into the Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value for that class.

$$pdf(x, mean, sd) = (1 / (sqrt(2 * PI) * sd)) * exp(-((x-mean^2)/(2*sd^2)))$$

Where pdf(x) is the Gaussian PDF, sqrt() is the square root, mean and sd are the mean and standard deviation calculated above, PI is the numerical constant, exp() is the numerical constant e or Euler's number raised to power and x is the input value for the input variable.

We can then plug in the probabilities into the equation above to make predictions with real-valued inputs.

For example, adapting one of the above calculations with numerical values for weather and car:

$$go\text{-}out = P(pdf(weather)|class=go\text{-}out) * P(pdf(car)|class=go\text{-}out) * P(class=go\text{-}out)$$

Best Prepare Your Data For Naive Bayes

- **Categorical Inputs**: Naive Bayes assumes label attributes such as binary, categorical or nominal.
- **Gaussian Inputs**: If the input variables are real-valued, a Gaussian distribution is assumed. In which case the algorithm will perform better if the univariate distributions of your data are Gaussian or near-Gaussian. This may require removing outliers (e.g. values that are more than 3 or 4 standard deviations from the mean).
- **Classification Problems**: Naive Bayes is a classification algorithm suitable for binary and multiclass classification.
- **Log Probabilities**: The calculation of the likelihood of different class values involves multiplying a lot of small numbers together. This can lead to an underflow of numerical precision. As such it is good practice to use a log transform of the probabilities to avoid this underflow.
- **Kernel Functions**: Rather than assuming a Gaussian distribution for numerical input values, more complex distributions can be used such as a variety of kernel density functions.
- **Update Probabilities**: When new data becomes available, you can simply update the probabilities of your model. This can be helpful if the data changes frequently.

Further Reading

Two other posts on Naive Bayes that you might find interesting are:

- How To Implement Naive Bayes From Scratch in Python
- Better Naive Bayes: 12 Tips To Get The Most From The Naive Bayes Algorithm

I love books. Below are some good general machine learning books for developers that cover naive Bayes:

- Data Mining: Practical Machine Learning Tools and Techniques, page 88
- Applied Predictive Modeling, page 353
- Artificial Intelligence: A Modern Approach, page 808
- Machine Learning, chapter 6

Summary

In this post you discovered the Naive Bayes algorithm for classification. You learned about:

- The Bayes Theorem and how to calculate it in practice.
- Naive Bayes algorithm including representation, making predictions and learning the model.
- The adaptation of Naive Bayes for real-valued input data called Gaussian Naive Bayes.
- How to prepare data for Naive Bayes.

Do you have any questions about naive Bayes or about this post? Leave a comment and ask your question, I will do my best to answer it.
