Naive Bayes

Tarushii Goel[*]

November 2021

1 Introduction to Naive Bayes

Naive Bayes is a *simple supervised classifier* based on Bayes' theorem that, despite its assumption that there is independence between every pair of features of the input given its classification, tends to perform quite well.

2 Derivation of Bayes' Theorem

We can start off with one of the basic probability equalities:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

This reads: the conditional probability of event A given that event B has occurred is equal to the probability of both event A and event B occurring divided by the probability of event B. What we want to find is a relationship between $P(A|B)$ and $P(B|A)$. You can imagine that finding this relationship would be useful in machine learning. It would be able to calculate the probability of a data sample belonging to a certain class given the features of that data sample (the goal of a classification machine learning model) using the probability that a data sample has certain features, given that it belongs to a particular class (something we can calculate from training data).

So, getting back to the equation, if we multiply both sides by $P(B)$, we get:

$$P(A|B)P(B) = P(A \wedge B)$$

Note that $P(A \wedge B) = P(B \wedge A)$, and furthermore, $P(B \wedge A) = P(B|A)P(A)$. Thus,

$$P(A|B)P(B) = P(B|A)P(A)$$

Dividing both sides by B, we arrive at Bayes' theorem:

[*]This lecture was written by Sylesh Suresh and edited by Charlie Wu

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Let us consider an example. We want to determine if a specific applicant will receive a job they apply for, given data on previous applicants who applied for the same job, with a discrete feature of the applicant: if they have a college degree. In this case, the probability that they get the job given their degree status = (the probability that previous applicants who got the job had a college degree) $*$ (the probability any previous applicant got the job) / (the probability any previous applicant had a college degree)

3 Bayes' Theorem

We can apply Bayes' Theorem to machine learning. Specifically, say we are given a data sample with n features, $x_1, x_2, x_3, ..., x_n$. We want to find the probability that this data sample belongs to a particular class y. In other words, we want to calculate $P(y|x_1 \wedge x_2 \wedge x_3 ... \wedge x_n)$. (We can use commas instead of the 'and' symbols for shorthand, writing the probability instead as $P(y|x_1, x_2, x_3, ..., x_n)$). Using a dataset of samples which are each labeled with the class y and features $x_1, ..., x_n$, we want to calculate this probability. In other words, consider the job applicant example from before, except now with the job recruiter considering more features, or more qualities of the applicant. Instead of only considering their college degree status (x_1), they could also start to consider if they have previous job experience (x_2), if they fit in well with the workplace (x_3), if they are a business major (x_4), along with a multitude of other features up to (x_n). We can apply Bayes' Theorem:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}$$

The numerator of the right-hand side of the equation can be condensed as $P(A|B)P(B) = P(A, B)$:

$$P(y|x_1, ..., x_n) = \frac{P(x_1, ..., x_n, y)}{P(x_1, ..., x_n)}$$

Using the same law, $P(A|B)P(B) = P(A, B)$, we can decompose the numerator like so:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|x_2, ..., x_n, y)P(x_2, ..., x_n, y)}{P(x_1, ..., x_n)}$$

Then, we can decompose the second factor of the numerator in a similar fashion:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|x_2, ..., x_n, y)P(x_2|x_3, ..., x_n, y)P(x_3, ..., x_n, y)}{P(x_1, ..., x_n)}$$

We can keep decomposing the last factor in the numerator until we arrive at:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|x_2, ..., x_n, y)P(x_2|x_3, ..., x_n, y)...P(x_n|y)P(y)}{P(x_1, ..., x_n)}$$

This is nice, but we usually cannot calculate the numerator directly if we are given an unseen sample. If we could, we would already have a sample in our training dataset with the exact same features $x_1, ..., x_n$. The goal is to be able to arrive at an expression that allows us to handle the general case.

At this point, we will have to make a (somewhat fallacious) assumption about the data in order to arrive at a meaningful expression. The assumption is that each of the features of the data is independent from every other feature. In other words, the probability that a certain feature takes on a certain value is independent of the values that other features take on. In our example of a job applicant, we would have to assume each quality of the applicant as independent from every other quality. We would then regard, for instance, the applicant's possession of a business major as being independent from whether they have a college degree or not. This is definitely not true for examples such as this one, which is why we refer to the classifier as "naive". However, it turns out that the classifier tends to make fairly accurate predictions, even in situations such as this.

The reason we make this assumption is because we really don't like all the conditionals in each probability. This assumption will make all those conditionals go away. If a feature is independent of every other feature, then we can simply remove the features from the conditional part of the probability. For example, let's look at $P(x_1|x_2, ..., x_n, y)$. Our assumption says that feature x_1 is independent of features x_3, x_4, x_5 and so on. This means that $P(x_1|x_2, ..., x_n, y) = P(x_1|y)$ - we can take out all the other features from the conditional. Moreover, we can do this for every single probability with features in the conditional. This is useful because we can then simplify our equation to:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)P(x_3|y)...P(x_n|y)P(y)}{P(x_1, ..., x_n)}$$

We can use capital pi (product) notation to write the n factors in the numerator:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

This is the fundamental equation of naive Bayes. Specifically, when we are given a sample with an unknown class and features $x_1, ..., x_n$, we can calculate $P(y|x_1, ..., x_n)$ for each possible class y. The class that returns the highest probability when plugged into the equation would be the correct classification for the sample. Each $P(x_i|y)$, or in our example, probability a previous applicant had a certain feature, given that they eventually got the job, is easily calculable from training data on previous applicants.

Note that $P(x_1, ..., x_n)$ does not depend on y (is constant with varying values of y), so we can simply maximize $P(y)\prod_{i=1}^{n} P(x_i|y)$ with respect to y, which will yield our class prediction.

4 Continuous Features

The expression we derived above will work cleanly for categorical features, as we can fairly simply calculate $P(x_i|y)$ for any specific value of the categorical feature x_i and specific class y. We can do this by dividing the number of samples in our training dataset with that categorical feature and class y by the number of samples with the class y.

Continuous features are a completely different beast. Working with continuous features, we must define $P(x_i|y)$ with a probability distribution. It is very likely that the samples in our dataset will have continuous features that take on values close to - but not exactly equal to - continuous features we will see in the real world. For example, say we are trying to predict the species of flower based on flower heights. We may have a flower in our training dataset with a height of 30 cm. If we are asked to classify a flower with height 29.5 cm, it is likely that flower will be of the same species as the 30 cm flower in our dataset. Due to the heights not being exactly the same, an issue arises. Since there is no 29.5 cm flower, when we try to calculate $P(x_1|y)$, the probability will turn out to be 0, a pretty bad prediction. The solution is to introduce a probability distribution, which shows that since 29.5 and 30 similar and close heights, it is likely that the 29.5 cm flower is the same species as the 30 cm flower.

One way to define this probability distribution is with a Gaussian distribution. If the training data has a continuous feature x_i, we will separate the training data samples based on its class y. For each set of samples with class y, we must calculate the mean μ_y and the variance σ_y^2. We can now define $P(x_i|y)$ as:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{\frac{-(x_i-\mu_y)^2}{2\sigma_y^2}}$$

This equation might seem intimidating at a first glance, but we are only plugging our calculated μ_y and σ_y^2 into the standard Gaussian probability density function. This allows us to solve the previously defined maximization problem to calculate our class prediction. Assuming a normal distribution may not always be the best answer compared other probability distributions we can define to increase the accuracy of our model, but for the sake of simplicity, the normal distribution serves as a decent approximation.