[1]

In this lecture we will introduce Bayes’ Theorem and work towards the development of a simple but powerful probability-based classifier suitable for many different kinds of machine learning applications

[2]

Bayes Theorem can be used to estimate the probability of some event given existing evidence believed to be related to that event. The evidence occurs with some probability and the event occurs with some probability. Bayes Theorem allows us to determine the latter. Formally, suppose we have some event with random variable X and some evidence with random variable Y, then the conditional probability of X given Y is the product of the inverse conditional probability Y given X and prior probability of X. In other words, we can say that the probability of an event given some evidence is the probability of the evidence given the event times the probability of the event.

[3]

For machine learning purposes we expect to be dealing with multiple random variables or features in our dataset instances, so Bayes Theorem generalises nicely to accommodate this scenario. The event of interest to us in machine learning is the probability that some target feature “t” takes on some value “v” given a set of features having certain assigned values of their own. Extending the version of Bayes Theorem already presented, we can derive a formula comprising joint and conditional probabilities over the features. As we will see shortly, we can apply the chain rule to expand the conditional probability on the right-hand side allowing us to calculate a Bayesian probability for our target feature from our dataset.

[4]

But first, let’s consider a sample dataset to illustrate how we can make use of Bayes Theorem. This dataset presents a small fictional sample of symptoms for eight patients suspected of influenza infection. The target feature, which is the rightmost column, is the outcome of a diagnosis for each of the patients presenting with the listed symptom features. For simplicity, each of the symptoms is a binary categorical feature having the values true or false. In practice, real-world datasets would be expected to have many more instances and columns and a contain mix of binary categorical, multi-value categorical and continuous features. You can refer to this sample dataset in the coming worked examples

[5]

Let’s apply our generalized Bayes Theorem by computing each of the terms by counting the events in our example dataset and computing the probability of a positive diagnosis of influenza given the that a patient presents with no headache, no fever but has a sore throat. For compactness, we denote the lowercase letters “h” for headache, “f” for fever and “st” for sore throat. The prior probability of a positive influenza diagnosis is just the number of positive influenza outcomes divided by the number of instances, in this case one half or 0.5. The prior, joint probability of a patient having no headache, no fever but having a sore throat is one quarter or 0.25. And the conditional probability of each of these symptoms given that there was a positive influenza diagnosis is one eighth or 0.125. Plugging these into the generalised Bayes Theorem formula, we obtain a probability of one quarter or 0.25. We were able to do this by simply counting the matching events in our dataset.

[6]

In a machine learning application, when programming a learning model based on Bayes Theorem, it will be easier to take a more formulaic approach than just counting. Recall from probability theory than we can apply the chain rule to expand and compute the conditional probability term on the righthand side. To illustrate this method and to show how this can go wrong, let’s try to find the probability of a positive influenza diagnosis given all three symptoms are true, this is, the patient presents with a headache, a fever and a sore throat. When we expand the terms using the chain rule it so happens that for our query using this exact dataset, several terms compute to zero and this probability is not computable. The reason for this is that, in this dataset, there is no instance matching these symptoms with that particular target variable outcome. Even in larger, real-world datasets this can happen and is illustrative of a significant issue in machine learning known as over-fitting.

[7]

Model training in machine learning relies entirely on data and the quality of that data. But there is a risk that if the distribution of training data does not sufficiently match the distribution of the real-world data, then the model will be overly biased towards the training distribution and will not perform well for classification or predictions of unseen data from the real-world. We will have a lot more to say about the problem of over-fitting throughout this course. Additionally, when there are a large number of features to consider, then the calculations can become computationally

[8]

One of the ways that we can address both the over-fitting issue and the computational complexity problem is to make a simplified assumption about the effect of some of our evidence variables, our instance features. It will usually be the case that not all features will have a role in determining the outcome of the target feature. If a random variable X has no effect on another random variable Y, then X and Y are said to be independent. The joint probability of X and Y is the product of their probabilities. However, ideally, we hope that at least one of our features influences the target feature while others may be fully independent. We call this conditional independence. For example, suppose our random variables are conditionally independent given a third random variable Z then the conditional probability of X and Y, given Z, is just the product of the conditional probabilities of X given Z and Y given Z. Applying this result to the generalized Bayes Theorem formula with the chain rule expansion, we get a nice, simple formula for calculations involving an arbitrary number of instances.

[9]

We have now arrived at the final Bayes Theorem formulation known as the Naïve Bayes Model. We call this naïve because of the simplifying assumptions of conditional independence between features. While this is not always strictly valid, Naïve Bayes performs very well in practice. We will see some real-world application of the Naïve Bayes in the course tutorials and in the lab practice exercises.

[10]

Whenever we develop and train a machine learning classifier, we need a way to evaluate the performance of that model to see if it ready for deployment or whether it needs to be refined and re-trained. Let’s briefly introduce some basic metrics which we can use to evaluate the performance our Naïve Bayes classifier. These metrics are based on the concepts of true positives, true negatives, false positives and false negatives. A true positive means a correct classification of the target feature as true. A false positive means an incorrect classification of the target feature as true. A true negative means a correct classification of the target feature as false. A false negative means an incorrect classification of the target feature as false. This allows us to define accuracy, precision, recall and f1-score as shown. These metric measure different aspects of the classifier performance. We will be considering the area of performance evaluation in much more detail later in the course

[11]

In summary, the Naïve Bayes Model is a simplified formulation of the generalised Bayes Theorem using conditional probabilities which allows us to efficiently consider many features for classifying some target feature. The Naïve Bayes model is a versatile machine learning model suitable for a wide range of applications.