Module 11 Transcripts

Video 1: Introduction to module 11 [01:41]

Welcome to module 11. In this module we'll cover Naive Bayes, which is one of the most successful machine learning methods when it comes to analyzing texts. It is used for example, in email spam filters, in document classification algorithms, as well as for sentiment analysis. Naive Bayes algorithm inherits its name from Bayes' theorem which it heavily relies upon. As such, it's natural that we first revise what this theorem is about, and what it tells us. I will afterwards introduce you to the exact Bayes algorithm. Never heard of that one? I'm not surprised as this algorithm is so impractical, that nobody would ever want to use it. It's nevertheless useful to discuss this algorithm. as it shows a way towards a very practical algorithm. The key is to introduce the assumption of class-conditional independence. This assumption leads us from the impractical exact Bayes, to the highly practical Naive Bayes algorithm. We will also see that the Naive Bayes algorithm isn't naive at all, and that it in fact often performs extremely well in practice.

Video 2: Motivation [05:10]

To understand how Naïve Bayes works, it makes sense to first look at a variant of the algorithm which I call Exact Bayes algorithm. Don't worry if you have never heard the term Exact Bayes algorithm. Such an algorithm actually does not exist. I made up this algorithm because it helps us to conceptually understand why certain assumptions are made in the Naïve Bayes algorithm. Now, to understand this said Exact Bayes algorithm, let's look at an example of real life.

If you watched the weather forecast, you will see that often statements are being made, such as on Tuesday there is a 50 per cent chance of rain. Where do such estimates actually come from? After all, it will either rain on Tuesday or it won't. Typically, such estimates are being made in the following way. People look at days that appeared in the past with similar atmospheric conditions to said Tuesday, and they take the fraction of these days where it actually rained. And that fraction is going to be used as a prediction for the probability for it to rain on Tuesday. It turns out that this concept can be used in a broader setting. Consider the following conceptual classifier. We're where given n training samples, where all the input variables as well as the output variable, are categorical. This will be our standing assumption throughout the session. We then have a new data point where we only know the values of the input variables, and we would like to predict the value, the category of the output variable. Now here's the algorithm. We look at all the training samples and we pick all of the training samples whose input variable values coincide completely with the input variable values of our new data point. We then

make the majority vote amongst those training samples, meaning, whatever category, whatever output category is in the majority of said training samples that have exactly the same input variables as our new data point, that majority category is our estimate for the output variable value of our new data point. This approach would not work in practice because it would require a huge amount of data. For every possible combination of input variable values, we would need to have several samples so that we can take the majority vote over these samples. We will never have so much data in practice.

By the way, if you look at the classifier, it turns out that a classifier is very closely related to the *k*-nearest neighbour classifier. Remember that in the k-nearest neighbour classifier, we look at the k data points in the training data whose input variable values are closest to the new sample point in question. Here, instead, we take all the training sample points whose input variable values are exactly equal to the input variable values of the new sample. Now we can use this imaginary classifier not just to predict the outcome variable value of a new sample point, but we can also use it to predict the probability of the outcome variable of the new sample point to be of a specific category. And this is shown on this slide here. In order to do so, we again, pick all the training sample points that have exactly the same values for all of the input variables and now, for each possible outcome category, we calculate the following fraction. We take the ratio of those training sample points that we have just determined whose outcome variables are of the category Ci where i ranges over all the categories.

So, for example, you mention you have five sample points in the training data whose input variable values coincide exactly with the input variable values of our new data point. Out of these five samples, three of category one and two of category two. In that case, our prediction for the probability of the new sample being of category one would be three over five, which is 60 per cent. And our prediction of the probability of the new sample point being of category two would be two over five. In other words, 40 per cent.

Video 3: Bayes' theorem: An example [04:10]

An important ingredient of both an unrealistic Exact Bayes algorithm as well as the practically useful Naive Bayes algorithm is Bayes' theorem. Bayes' theorem tells us how our probability estimate of a certain event taking place should be updated in the light of additional information. Let's first have a look at Bayes' theorem through the lenses of an example and then we will derive the statement of Bayes' theorem theoretically. Consider the following example.

You have an inbox with 1,000 unread e-mails. And from your past experience with e-mails, you estimate that about 200 of these e-mails are spam e-mails, like advertisement e-mails, and 800 of the e-mails are ham e-mails, which are e-mails that are actually useful to you. Now, with this information, what is the likelihood, the

probability, that the first e-mail that you open in your unread e-mails is going to be a spam e-mail? Well, if no other additional information is available, that probability should be 20 per cent or 0.2, the reason being that 200 out of your 1,000 e-mails that are unread are assumed to be spam e-mails. Now, let's consider how our probability estimate should be updated in the light of additional information. In particular, let's assume that based on past experience, you believe that 50 out of your 1,000 unread e-mails will contain the word Viagra. Clearly, e-mails containing the word Viagra are more likely to be spam e-mails. Now, if we consider the probability of the first unread e-mail to be a spam e-mail, then that probability actually won't change with this additional information. It will still be 20 per cent because we don't know whether this unread e-mail contains the word Viagra or not.

So, let's look at the conditional probability that the first unread e-mail is a spam e-mail conditional on the additional information that that e-mail contains the word Viagra. Unfortunately, we cannot say anything more about this conditional probability either at this stage. The reason is we don't yet know how these 50 Viagra e-mails that you estimate to be in your 1,000 unread e-mails -- how these 50 e-mails relate to the spam and ham e-mails. So, let us now assume that out of these 50 Viagra e-mails that you expect to be amongst your 1,000 unread e-mails, 40 e-mails are going to be spam e-mails and only ten out of these 50 e-mails are going to be useful or ham e-mails. In that case, the probability that the first unread e-mail that you open is a spam e-mail still remains 20 per cent if you don't know whether that e-mail contains the word Viagra or not. However, if you know that the first unread e-mail contains the word Viagra, then you can update your probability of that e-mail being a ham or a spam e-mail. In particular, the probability of that e-mail being spam conditionally on our knowledge that this e-mail contains the word Viagra now jumps from 20 per cent to 80 per cent because now we only consider the 50 Viagra e-mails out of which 40 - which is 80 per cent - out of which 40 are actually assumed to be spam e-mails and only ten are assumed to be ham e-mails.

Video 4: Bayes' theorem: Derivation [03:41]

Let us now try to abstract a little from our spam example and consider the generic Bayes' theorem. Bayes' theorem allows us to calculate the probability of an event A happening under the additional information that the event B has happened. This is also called a conditional probability. In our particular example, we were interested in the probability of the first unread email being a spam email, conditional on our knowledge that that email contains the word Viagra. By the definition of conditional probability, the probability of A given B is equal to the probability of A and B divided by the probability of event B.

So, in our concrete example, the probability of the email being a spam email under

the additional information that the email contains Viagra is equal to the probability of that email being a spam email that contains the word Viagra divided by the probability of that email containing the word Viagra, no matter whether it's a spam or a ham email. We can now use the definition of conditional probability once more to replace the term P A and B in the numerator. If we do so, we end up with the formula on the right, which is P of B given A times P of A divided by P of B. Let us look at these terms in a bit more detail. The term P of A given B is often called the posterior probability in the Bayesian literature. It is the probability of event A happening under the additional information that we know event B has happened. In our concrete example, the probability of the email being a spam email under the knowledge that the email contains Viagra.

Now, at the other extreme, we have the probability P of A. This is called the prior probability. In our concrete example, that's the probability of the email being a spam email without the additional knowledge whether or not the email contains the word Viagra. There are two more terms left on the right-hand side. In the numerator, we have the so-called likelihood, and, in the denominator, we have the so-called marginal likelihood. You can think of the likelihood divided by the marginal likelihood as a correction factor that brings the prior probability to the posterior probability under the additional information that we have. The likelihood divided by the marginal likelihood is one, if the additional information, in this case that the email contains the word Viagra, does not change anything in our probabilistic estimate -- it is going to be greater than one if the probability of the email being a spam email has increased by the additional knowledge that the email contains the word Viagra. And this division of likelihood divided by marginal likelihood is going to be less than one if the probability of the email being a spam email has decreased by the additional knowledge that the email contains the word Viagra.

Video 5: Exact Bayes' classifier – part 1 [03:56]

Bayes' theorem is a fundamental ingredient in the Naïve Bayes algorithm. Before we discuss this algorithm, however, I will introduce to you a variant which I call the Exact Bayes algorithm. That algorithm doesn't actually exist in practice, because it would not be useful, but it serves us well to understand how Naïve Bayes works and most of all why we are making certain assumptions in the Naïve Bayes algorithm.

Okay, so here's the set-up. Consider a very simple spam filter that we want to construct. That filter will decide whether an email is spam or ham based on the presence or absence of one single keyword, namely the keyword 'Viagra'. Now, in order to build this spam filter, we need to have some historical database. This is shown on this slide here. It is simply a two-by-two matrix which has two rows. The rows show the ham messages as well as the spam messages and two columns. The columns stand for messages that contain the word Viagra and messages that don't

contain the word Viagra. So, for example, if we look at the row one column one, we see that four emails out of a total of 100 emails are spam emails that contain the word Viagra. The total on the right of the first row shows us that 20 emails out of a total of 100 emails in our historical database are spam emails. The element in row two and column two tells us that 79 out of our 80 ham emails do not contain the word Viagra. So, we want to use this table to predict whether any new email that we see is a ham or a spam email, depending on whether that email contains or does not contain the word Viagra. We can use Bayes' theorem for that. In particular, we know that the probability of the email being spam conditional on the email containing the word Viagra is proportional to the probability of a spam email containing the word Viagra times the probability of any email being a spam email.

Let's look at how we can get these probabilities from the table that we see on the slide. So, the probability of a spam email containing the word Viagra, we get that probability by dividing the field in row one column one, the four emails, by the row total of row one, which is 20 emails. So, we have four divided by 20. The probability of any email being a spam email is equal to the row total of row one, which is the number of spam emails divided by the overall number of emails, which is 100. Likewise, if we wanted to consider the probability of the email being ham, conditional on the knowledge that the email contains the word Viagra, that is equal to, that's not equal to but proportional to the probability of any ham email containing the word Viagra. That probability is one over 80, because it's row two column one divided by the row total of row two times the probability of any email being a ham email which is a row total of row two divided by the 100 emails, which is the total email number in our historical database.

Video 6: Exact Bayes' classifier – part 2 [02:07]

It turns out that exactly the same idea can be used for spam filters that are not just based on the presence or absence of a single word but based on the presence or absence of any combination of multiple words, which is, of course, much more realistic. So, let's assume now we want to build the spam filter whose historical database contains N different words, such as Viagra, Unsubscribe, Buy, etc. Now, the probability of an email being spam, based on the knowledge that the email contains the word 'one', does not contain the word 'two' and so on up to word N is going to be proportional to the probability of a spam email containing word 'one', not containing word 'two' and so on times the probability of any email being a spam email. And exactly the same for ham. Let's have a look at an example. I'm now considering a spam filter that does not just decide whether an email is ham or spam based on a single word but on two words, namely the presence or absence of 'Viagra' as well as 'Unsubscribe'. You see that now our table grows much bigger. Instead of having two columns, we need four different columns, because there four combinations of the words 'Viagra' and 'Unsubscribe' being present or absent. Now,

based on this table, can you guess what the right probabilities are or the right proportion of values for the probabilities of the email being spam or ham if an email contains the word 'Viagra' but does not contain the word 'Unsubscribe'?

Video 7: Exact Bayes' classifier – part 3 [02:32]

Let us now revisit the blueprint of a classifier that we discussed at the very beginning of this session. You might remember that in order to predict the probability of the outcome variable being of a particular category, we took the new data point that we want to consider. And for this data point, we collected from the training samples all the training samples that had exactly the same values for all input variables. Then, we looked at each possible category for the outcome variable, and we calculated, for each category the fraction of the selected few training samples that have the same outcome category. Now, it turns out that this is exactly the same - what Exact Bayes does to calculate the probability of the email being ham or spam. So, in other words, the blueprint of the classifier that we discussed at the very beginning of the class is exactly the same what Exact Bayes does. The saying goes, for predicting a category, that is just what Exact Bayes would do by choosing the outcome ham or spam, depending on which outcome has the higher probability. There is a problem, however, with the Exact Bayes approach, and that is -- we need a huge database. Think about it. Imagine you have N different words in the database. Then, our table that we have seen before has to have two to the N columns. And, of course, we need to have enough historical emails to populate all of these columns. So, just to give you an example, I took a look at the web, and I found a couple of articles for typical spam email keywords. I found lists with about 400 different keywords. So, two to the 400 is a number that is actually far exceeding the number of atoms in the known universe. Clearly, there is no way that we can store this on a computer. So, we need an alternative approach here.

Video 8: Naïve Bayes classifiers [04:36]

We're now ready to make the move from the Exact Bayes classifier, which does not work in practice, to the Naïve Bayes classifier, which is actually very popular in practice. The key difficulty that we encountered with the Exact Bayes algorithm is the huge number of word combinations that we need in our spam filter. For N different words, where N should be in the hundreds, we need two to the N fields in our table. That creates two problems. From a practical perspective, we need a lot of memory to store them, but also, perhaps more importantly, we would need a huge amount of data to populate all of these fields, because otherwise, most of these word combinations will never be seen in an email. That is actually a manifestation, we will see in a moment, of the bias-variance trade-off. It turns out that the Exact Bayes algorithm has very little bias. It can pretty much learn any type of spam behaviour based on any combination of words, but this small bias is bought at a very high price of the variance. We would need a huge amount of data in order to

populate that table in order to make that algorithm stable.

So, what we will do now is we will introduce an additional assumption. That assumption will create a bias because not all applications will satisfy that assumption, but for that bias, we will massively reduce the variance. We will make that table that we need much, much smaller and the outcome is an algorithm that actually becomes practically useful. So, here is the assumption that we make - we make the assumption of class-conditional independence. This is what the assumption means. Remember that in order to calculate the probabilities in our Exact Bayes algorithm, we need formulas such as the probability of word 'one' appearing, word 'two' not appearing and so on up to word N based on conditional -- on the email being spam. Now, the problem is to calculate that probability, we need -- for every possible combination of presence and absence of words -- we need a field in our table, a column in our table.

So, what we will do instead is we will say that this probability is roughly the same as the product of conditional probabilities that you see here. So, what does this assumption mean? It means that if you have a spam email, an email that you know is spam, then the presence or absence of the word 'Viagra' is conditionally independent of the presence of absence of the word 'Unsubscribe' and so on. Please note, this is fundamentally different to the assumption that in any email, the presence of absence of the word 'Viagra' is independent of the presence or absence of the word 'Unsubscribe'. That would be a very strong assumption that is most likely not satisfied in practice, because if any random email that you pick contains the word Viagra, it's very likely that it is a spam email, and spam emails are much more likely to contain the word 'Unsubscribe' than non-spam emails.

So, if we were to make an independence assumption that is not class conditional, that would be an extremely strong assumption. Instead, we are here making an independence assumption only based on conditioning on a particular class, be it ham, be it spam. Now, this assumption helps us a lot, because now, if you look at the table that we need, we only need two columns for each word. Instead of two to the N columns, we need two times N columns, which is a much smaller number, because we need, for every word, a column for the presence of that word in the various categories of emails -- in this case, ham or spam -- and for the absence of that word. This leads to lower memory requirements, but, even more importantly, it has a much higher chance that now we have sufficiently many emails to populate each cell of the table, which we need in order to build a functioning classifier.

Video 9: Class-conditional independence [06:45]

Let's try to further investigate this property of class-conditional independence. Remember, Bayes' theorem allows us to conclude that the probability of an email being spam conditional on our knowledge that it does contain the word 'one'. It does not contain the word 'two'. It does not contain the word 'three'. It contains the word

'four' and so on -- that this probability is proportional to the probability of a spam email containing the word 'one', not the word 'two', not the word 'three', the word 'four', etc. times the prior probability of any email being spam. This is what Bayes' theorem tells us. Now, the assumption of Naïve Bayes is class-conditional independence, which says that this quantity here, this product here, is roughly equal to - is not going to be exactly equal to - but roughly equal to the probability of any spam email containing the word 'one' times, and this is independence here, independence of probabilities means probabilities are being multiplied times the probability of a spam email containing word 'two' times the probability of a spam email not containing word 'three' times the probability of a spam email containing word 'four' and so on - times the probability of any email being spam. It is this - it is breaking up this expression here, which would lead us to very difficult bookkeeping, because we would need exponentially many combinations to be stored in our dictionary. This simplification allows us in Naïve Bayes, as we will see in a moment, to have much smaller dictionaries of the number of emails containing or not containing certain words.

Now, why do we actually use Bayes' theorem in order to make the simplification? In other words, why don't we assume that the following holds? That this probability here prior to use the Bayes' theorem, that this probability is roughly equal to the multiplication of those probabilities. You see the difference here. Here, I first used Bayes' theorem, and then I have assumed independence. Whereas here, I don't use Bayes' theorem. I just assume independence based on the different events that are conditional. It turns out that while this assumption here is meaningful, this assumption here would be catastrophic. Let's try to understand why this is the case. To this end, let's assume that the following would hold. That the probability of an event C given that both A and B have happened, that this is equal to the probability of event C given A times the probability of event C given B. This is a short form with only three events of this longer expression here. Let's assume that this would indeed be the case. Now, let's also assume, for simplicity, that the events A, B and C are actually independent. What would that then mean? Well, by the definition of conditional independence, this is the same as the probability of A, B and C happening divided by the probability of A and B happening. This here is the same as the probability of A and C happening divided by the probability of A happening. And this here is the probability of B and C happening divided by the probability of B happening. This is the definition of conditional probabilities. And now, by the assumption that A, B and C are independent, this would actually be the same as the probability of A times the probability of B times the probability of C divided by the probability of A times the probability of B. This here would be multiplication of those two probabilities. And this here would be the probability of B times the probability of C divided by the probability of B. Now, let's see what remains here.

These three terms will cancel out. These two terms would cancel out. These two terms would cancel out. So, in other words, we would have to assume that the

probability of event C is equal to its square. So, in other words, this assumption here could only hold if the probability of event C is zero or one. So, this is clearly not a reasonable assumption to make. That's why Bayes' theorem was crucial. Bayes' theorem allows us to revert the causality if you wish. It allows us to revert the causality and then make this class-conditional independence assumption based on the reversed order of events. It is perhaps not the case in practice, but it's still reasonable to assume that it approximately holds that the probability of a spam email containing the word Viagra is independent of the spam email containing the word 'Discount', for example. This is exactly what class- conditional independence assumes. It is something that is not typically holding exactly in practice, but it holds approximately, and this approximation allows us to drastically reduce the computational as well as the data requirements, as we will see in a moment.

Video 10: Converting features from numerical to categorical [03:58]

In module 8, we have discussed how we can convert categorical features into numerical ones, so that we can use them for the distance calculations and k-Nearest Neighbors. Since Naive Bayes requires all features to be categorical, we now need to discuss the reverse approach. How can we convert numerical features back into categorical ones? There're in fact two popular methods that achieve this, and there're both based on the concept of binning. The first one specifies the cut-off points between consecutive bins manually, whereas the second one does so automatically. So, let's start with manual binning. And to make things a bit more easy to visualise, let's consider a medical application that has a feature called age, a numerical feature called age.

But we want to turn this numerical feature into a categorical one. In that case, we could follow the medical literature and specify five age brackets such as 0-18 years, 19-44 years, 45-64 years, 65-84 years, and everybody with age 85 years or above. So, each data sample of a patient, let's say, with age 14, would get a new feature values 0-18 because that's the appropriate age bracket, where the 94 years old person would get the future value 85 plus. The second binning method works very similarly, but it's based on quantiles, and it can be automated. Imagine we again 1-5 age brackets, then we could compute the 20 per cent the 40 per cent the 60 per cent, and the 80 per cent quantiles of the ages in our data set, and the first bin would then contain all the ages that are less than equal to this 20 per cent quantile, the second bin would contain all the ages between the 20 per cent and the 40 per cent quantile and so on. So which approach should we choose? If there is a commonly accepted way to bin the numerical data as it is the case, for example, in some medical applications, I would recommend to use that one. Moreover, if you have domain knowledge that helps you define natural cut-off values between the bins, I would also bin manually. If neither is the case, I would use automated binning. That leaves us with one final question: How many bins should we choose?

There is no right or wrong here, but there is a trade-off involved. If we use too few bins, then the information gets lost. For example, in our age example, if we only had two bins; less than equal to 40 years versus greater than 40 years, we are probably being too close. On the other hand, we don't want too many bins either. In our age example we could, for example, choose 0-1 years as one bin, 1-2 years, 2-3 years and so on. The problem with too many bins is that we receive small counts in the frequency table and our data set become susceptible to noise. The right number of bins thus depend on the data set, but I would typically recommend to look at something like 4, 5 or 6 bins at most.

Video 11: Summarising module 11 [01:28]

This concludes module 11 on the Naive Bayes algorithm. In this module, we have discussed how the exact based algorithm uses Bayes` theorem to solve classification problems where all the features are categorical. We have discussed how the assumption of class conditional independence allows us to turn the impractical exact based algorithm into a practically useful, Naive Bayes algorithm. And we have discussed how we can use the Naive Bayes algorithm to solve challenging text analysis tasks such as detecting spam emails or understanding the sentiment of a text, such as for example, a tweet. Along the way, we have also discussed how we can convert numerical into categorical features, which complements our earlier discussions from module 8. In the next module, we will continue exploring the various applications of Bayes` theorem when roof talks about Bayesian optimization, and I will see you again in module 14 when we discuss support vector machines.