Probabilistic Learning

1--- Cover

2---

Let's start by considering the question whether ML is based on probability. Interestingly, this is a question that can be answered both yes and no, depending on what we mean by based on. In some sense all ML is based on probability because it is based on inductive inference, it is based on making predictions on samples of data and that is a kind of probabilistic reasoning. But on the other hand not every kind of learning algorithm makes use of an explicit model of probability. In fact, none of the methods that have been considered so far has explicit probability model. Take decision trees, for example. When we use a DT to classify a set of instances, we don't compute any probability we just sort of answer questions and find the right classification. But arguable the reason why a decision tree works for classification is that these questions have been set up so that we can make sound probabilistic inferences from our data. So in ML we can say that there are 2 different roles for probability theory. One of them is to provide a framework for theoretical analysis for learning methods, where we can reason about things like the expected error rate of a particular algorithm, regardless whether that algorithm has an explicit probability model or not. But in this course we are not going to look into that, because this course is more practically oriented. But it turns out that probability theory can be used in a very practical sense in ML mainly by building a learning algorithm that makes use explicit reasoning about probability, and this is what we will look at in this lecture.

3---

Now, just to give you some intuition. Given a prediction problem like the weather in Bergen (Norway) and the only data that you have is that it is a day in October , and you are given the information that the averaged number of rainy days in October in Bergen is 25 out of 31. Given this information, what would you predict? Is it going to rain in that particular day or not? well…people will say that there is a high probability that that day is going to rain. Well all things being equal it seems we have about 80% chance to be right if we predict rain and only 20% chance if we predict if it is not going to rain. It seems that probabilistic reasoning can be a useful way to make predictions based on data.

4---

Let's try to make this a little bit more formal. Suppose we have a classification problem where we want to classify instances from some set X into classes of some set Y. This can be a problem like classify days based on whether they are rainy or not, or documents into their topics, or words by their parts of speech. A probabilistic formulation of this problem is that we want to find the class Y that maximizes the conditional probability of the class given the instance , or, if you want, the most probable class for that instance. Using Bayes law we can rewrite this into this ratio. And as long as we are only interested in maximizing, we can forget about the denominator, namely the probability of the

instance, because that is given. So we are left only with the numerator and we know (again rom probability theory) that this expression is actually equivalent to the joint probability of the input and the output. So in this sense, it seems that if we take a probabilistic view of classification we can build a classifier by creating a model of the joint probability of inputs and outputs.

5---

Now assume that we have an oracle that always give us the true joint probability of an input X and some output Y, we can actually prove that the best thing we can do is always classify X by the most probable class. Why? This is know in the literature as the Bayes optimal classifier because you can prove that the Bayes error rate will always be the theoretical minimal error rate, in the sense of zero/one loss (ie whether we pick the right class or not) and averaging over a large set of points. It is important to remember that this is an average. It is important to know that any classifier can ever do better on average than the Optimal Bayes classifier. This does not mean that there can't be instances where the Optimal Bayes Classifier makes the wrong classification while other classifier gets it right. Then other classifiers must then by necessity make other mistakes on other instances.

6---

The Bayes Optimal Classifier is a theoretical construct. It is, if you like, the lower bound on the error rate. In practice, all Bayes classifiers that we can actually build are suboptimal and they are so for several reasons. First of all, the basic problem is that except for very restrictive situations we cannot really know the true joint probability of the inputs and the outputs, but we have to estimate it from data. Secondly, in order to make the problem computationally tractable to make estimations, we need to make the independence assumption, and this will simplify the model. In the next video we are going to see a simple but surprisingly effective cousin of the Bayes Optimal classifier known as the Naïve Bayes classifier.

Quiz1: Roulette

Naïve Bayes: Intro

1--- Cover

2---

In the previous video we started exploring probabilistic classification, ie the idea that we can use a model of the joint distributions of inputs and outputs P of x and y to perform classification of inputs by their output classes.

We looked at the Bayes Optimal Classifier that is a theoretical notion that establishes the minimal error rate that we can ever achieve using this approach. The Bayes Optimal classifier presupposes that we know the true distribution of input and outputs and then if we classify the inputs always taking the most probable output class, we get the minimal theoretically-possible error rate as long we are measuring the error rate by zero-one loss, ie the percentage of instances that are correctly classified. In this lecture

we are going to start looking at the Naïve Bayes classifier, which, if you like, it is like the poor cousin from the country compared to the Bayes Optimal Classifier. It does not come with any theoretical guarantee, but it turns out to be a practical efficient and useful alternative.

3---

Let us understand better what we would like to do in probabilistic classification. We assume that we are given a universe X tha our examples are drawn from. In the case of a typical NLP application it could English documents with some predefined vocabulary. We also assume that we have a predefined set of Y classes for these examples. So we might want to classify documents into predefined topic categories, like news, finance, sport, or we want simply to make a binary distinction between spam and ham, when for ex implementing a spam filter. Crucially we also assume that we are given a training set. This is a labeled training set, so we a given a number of documents and for each document we now what the correct class/label is. What we want to know in this set up is to learn a function f that maps documents to class probability. So in other words, f takes a document and a class and returns a number 0 and 1. and it is subject to the constraint that if sum all these numbers for all the possible classes, the sum should be 1. In other words, these number should form a proper probability distribution. Now, given that we learn this function, we can then use it to classify new examples, given this simple decision rule where we pick the class y that maximizes this function. In other words, it maximizes the joint probability with the input.

4---

Before we look at the technical details, I want to introduce a motivating example that hopefully will make it easier to understand why a NB classifier is designed the way it is. Suppose I have 2 coins, we call them C1 and C2. Suppose I repeatedly perform a simple experiment where I take one of the coins out of my pocket, I make a number of flips and then I record which coin it was, and what the outcome of the flip was. This can give us some data like the following (on the screen). Here we have 7 experiments. In the first one I have picked up one of the coin, and suppose 1 is for heads and zero for tails, so the first experiment is tail, head, head, head, head. Etc. Suppose one gives you a new sequence., for x 001 (or tail, tail, head) and asks you to tell which coin it is from. We cannot say this with certainty, but we could at least say which coin it is probable. So this would be a simple example of probabilistic classification, where we classify flips by the coin they come from.

5---

How can we approach this problem? First of all, estimate the simple class probability., that is the probability that coin flip is the from C1 or C2. We can use relative frequencies. It is a starting point. There were 7 trials, In 4 trials, it was the C1, in 3 trials, it was C2. It is also relatively easy to get the conditional probability of a flip coming up heads given the coin. The reason it is easy it's because we can assume conditional independence between the different flips in the sequence. So given that we fix the coin, for ex C1, we know that the joint probability of 3 flips it is simply the product of the probability of each flip outcome conditioned on the coin and we multiply those probabilities with each other So in this way we can see that in 16 flips performed with the C1 12 were heads,

so the probability ahs to be ¾ (=12/16) and then follows that the probability of coming up tails (ie 0 in this representation) it is only ½ (=6/18). And we can do similarly for C2. Now the question is: can we use these probabilities to get the probability that we really want here, which is the inverse probability, namely what is the probability that it was a particular coin given the sequence that we have observed?

6---

There problem here is that we have the probability of the input given the output, but we want to have the probability of the output given the input. In this case, we have the sequence of the probability given the coin, but we want to have the probability of the coin given the sequence. Fortunately, we all know Bayes law, so we can use Bayes low to rewrite those probabilities. The probability that we want, ie the probability of y given x, we know it is the same as p of x given y times p of y . This together is the joint probability of x and y divided by p of x. Now if we look at this expression, we know all the probability of this expression except p of x. We do not know the probability of heads and tails regardless of the coin, but we can get these probabilities by marginalization. So if we know the conditional probability of heads for all the different coins and the simple coin probability, then we can sum this expression and it will give us p of x here. In essence, in order to be able to do the classification we want, with the data we have, we need to get the probability of the input (in this case the coin sequence) given a coin and then the probability of that coin. This is something that we already have.

7--- NAÏVE BAYES CLASSIFIER

The reason it works in the coin example is that we can assume that coin flips are independent. given the coin. So we can get the probability of the entire sequence of flips from the separate probability of each flip given the coin. What about a more real world problem, like identifying a fruit given a set of features. Suppose we have three features: a coulor feature, which is discrete and has value red, green, yellow, or orange.; a shape feature which is also discrete and we classify them into round oval or long+skinny; and finally we have size feature which is something like diameter in inches or centimeters and that is continuous features.

8--- NAÏVE BAYES CLASSIFIER

The problem here is that these features are not necessarily independent given the type of fruit. We do not have the same conditional independence that we had in the coin example. For example, at least judging from this picture, given the class apple the coulor green is more probable if we have a small value for the size feature. In other words, the probability of the color green given that the size is smaller than 2 and given that it is an apple, is actually greater than if we consider only the probability that the color is green conditioned on the apple. In this case we cannot use the simple solution that we used for the coin case.

9---NAÏVE BAYES CLASSIFIER

What we can do is to use the chain rule because we know (again from probability theory) that in order to get the probability that something is an apple (given that it is round, green and size equal to 2) is (again using Bayes law) the probability that it's

green, round and has size 2, given that is an apple times the probability that it is an apple. And then we need in the denominator here we need the probability of the features, we can in principle sum over all possible fruits. The problem in this case is that in order to compute this numerator we can no longer assume conditional independence between the features. We have to use some version of the chain rule. So we first get p of apple, we get p of size 2 given apple and we get p of being round given size 2 and apple and finally the probability of being green given round, given size 2 and given apple. So we keep on accumulating conditions here. This makes it much harder, because computing conditional probability can be hard and there are many feature combination for each fruit. and now if we think about a typical NLP example, ie document classification, and if you use words as features, there will be lots of features and the conditional probability will grow monstrously in that set up.

10--- NAÏVE BAYES CLASSIFIER

What if we are a little more naïve? Even if we know that it is not strictly speaking true, we assume conditional independence for all features given a class. So we assume that a probability that a fruit is green, given that is an apple, that it is round and has size equal to 2 is the same as the probability of an apple being green regardless of the shape and size. And similarly for all the other combinations. This in fact means that we can reduce the fruit classification problem to the same methodology that we use for the coin classification problem by pretending that fruit features behave just like coin flips. And this exactly what gives us the naïve bayes classifirer.

Quiz 2: coin flips

Naïve Bayes Continued

1--- cover

2---

Remember that NB is a probabilistic classifier that is based on the joint distribution, ie we have a model of the probability of x and y, where x is typically the input and y is the output or class that we want to predict. This is equivalent to the simple class probability multiplied the probability of the input given the class. Now what is special about the NB classifier is that it assumes that any feature over the input that we want to include in the model are conditionally independent given the class. So normally when we want to model the probability of the input given the class, we will look at the whole range of different features. of the input (see equation) and we assume that the probabilities of all these features given the class can be approximated by taking its feature separately and estimating the conditional probability given the class, then just multiplying this. So jthis the central independence assumption of the NB classifier.

3---

For the rest of the video I will use text classification or document classification as an example. But NB can be used for many other problems in Language Technology. When we do text classification we want to classify documents that can be anything from articles to short email messages, or whatever into different classes. We do this by computing the probability of a document d being in class i. This means, given the document, what is the probability for different classes? We know that, as long as we are interested in finding the most probable class, this is proportional to the joint probability of the class and the document and given the NB model , this can be computed by taking the a-priori class probability (some classes occur more often than others) and then combining that from the evidence from the features of document, which in the case of text classification is primarily the words that occur in the document. So we look at all the words in the documents and we look at the conditional probability. So the probability of w sub i given c is the conditional probability of the term w sub i occurring in a document of class c. We take this as the evidence provided by the word sub i that c is in fact the correct class for this document. n sub d is the document length (or the number of tokens (ie words) included in the model. The product is over the number of words. The other factor here is the prior probability of class c. The idea is that if the document's terms do not provide clear evidence for one class vs another, all things being equal,we choose the c class with the higher probability. Supposing we do spam filtering ,for example, and we know that we get more non-spam messages than spam messages in a day, all things being equal, our first guess is that a new message will be non spam, if the terms do not strongly indicate that the message really is spam.

4---

When using this type of classifier, our goal is to find the best class and technically in probability theory this is known as the maximum a posterior class or map or cmap. The map is the class of all possible classes that maximizes the conditional probability of the class given the document. And as we saw in the previous slide, the way we compute this probability or better one way of maximinzing it is to compute the product between the apriori class probability and the evidence of the words of the documents. The reason we write hat P instead of P is to emphazise the fact that whenever we use the model on some real data, then we do not know the real probability here, so we have to use estimates that are based on empirical data.

5---

An important questions working with these models, as well as any other probabilistic model, is: how do we estimate these probabilities? Suppose we want to estimate the probability that the word "buy" occurs in a document that is spam (if we are doing spam filtering). We collect a number of spam messages , look at the different words that occur there, we count the number of times that "buy" occur compared with the number of times that other words occur. One thing that we can do is to take a maximum likelihood estimate, which is just a relative frequency of that outcome compared all other possible outcomes. So if "buy" is the "i^{th}" term here (see equation), it simply counts of occurrences and (see denominator) we sum all the number of occurrencies including that term and we take the ratio (see equation). And that gives us maximum likelihood estimate (MLE). One question is: is MLE reasonable for this kind of model and this kind of data? It turns out that the MLE is usually problematic.

6---

The easeast way to see why the MLE is problematic is to see what happens when a particular words does not appear at all in a class of documents. Suppose there are no occurrences of the word “bagel” that belong to the class spam, then we get a 0 estimate of the probability of bagel occurring in a spam document. Because the frequency of bagel is zero, and 0 divided by anything is also 0. This will have a very unfortunate side effect, because as a consequence we get 0 probability for any spam document containing bagel. This happens because when we compute the probability of a spam document, somewhere in the chain of multiplication will have a zero and that means that the whole product will be zero. And this unfortunate because there also many other words in that document that might contribute with important evidence to whether this is spam or not and this information is destroyed y the occurrence of a single zero. So zero probability should be avoided unless we have really strong reason to assume that some word probability really is zero.

7---

For many applications we have some prior notion or belief of what the probability distribution should look like. So instead of MLE that only consider the evidence of the dataset, we can use we an use the Maximum a posteriori estimate (MAP) that also takes into account the probabilities of the prior distribution. In this case , we are trying to maximize not only the likelihood of the data, which is what we do with MLE) but also the joint probability of the likelihood and the prior distribution. Using a prior in this way is often referred to as smoothing (sometimes also “regularization”). One way of thinking of “smoothing” is to smooth (“spread around”) probability mass so that no word will get zero probability. They way we do it is to move some probability mass from words that higher probability in order to achieve this.

8---

One way to do this, for a discrete multinomial distribution[1] (that is the kind of distribution that we have over words for example) is to take probability of a word in a class to be the observed frequency of that word plus some constant that counts for our prior belief that this words has a non-zero probability of occurring even if we have not seen it in the training data. And the we sum over the frequencies of all the other words, as before, but then in order to get a correct probability model we also have to take into account that all those counts are also increased or can be increased. This factor here (alpha) which can be individualized to different words, it is known as “smoothing factor” and it is pseudo-count. It looks like as if we had counted more occurrences of word, but it is only a pseudo-count (kind of simulation)…

8---

[1] A multinomial distribution is an extension of the binomial distribution, since can handle more than two possible outcomes. The binomial distribution model is used when there are two possible outcomes (hence "binomial"). The Bernoulli distribution is a special case of the binomial distribution.

The simplest way to do this is to add 1 for all words, for all items I our vocabulary. This is normally referred to as Laplace Smoothing or sometimes add1 smoothing. It is a simple model that is sufficient for certain times of problems, but in general there exists many more sophisticated smoothing methods., we are not going to explain now

9---

Summing up: using NB for text classification :

we compute the joint probability of a document and a class in terms of class probability times the probability of the document given the class. Now the latter probability is really the probability of all the words in the document conditioned on the class and we compute this by taking the probability of each individual word given the class and multiplying them together. it is important to remember that this is because of the conditional independence that we made: the probability of any word conditioned on all the other words and the class is just the same as the probability of that word conditioned on the class. And if we estimate these probabilities using laplace smoothing, these are the equations that we will need to use to estimate the class probability and the word probability given the class.

10 ---

NB can be used for spam filtering, etc. sentiment analysis, word sense disambiations.

with NB features can be divese and numerous. Numerous features do not undermine the efficiency of the classifier. It words best if there not strong interdependence between these features, and this is natural because after all we are making an independence assumption, that probably does not really hold. If there are strong dependencies in the real data, we are likely to suffer from this. But if the dependencies are of a weaker type, then NB can work very well