
Evidence and Probabilities

Fundamental concepts: *Explicit evidence combination with Bayes' Rule; Probabilistic reasoning via assumptions of conditional independence.*

Exemplary techniques: *Naive Bayes classification; Evidence lift.*

So far we have examined several different methods for using data to help draw conclusions about some unknown quantity of a data instance, such as its classification. Let's now examine a different way of looking at drawing such conclusions. We could think about the things that we know about a data instance as *evidence* for or against different values for the target. The things that we know about the data instance are represented as the features of the instance. If we knew the strength of the evidence given by each feature, we could apply principled methods for combining evidence probabilistically to reach a conclusion as to the value for the target. We will determine the strength of any particular piece of evidence from the training data.

Example: Targeting Online Consumers With Advertisements

To illustrate, let's consider another business application of classification: targeting online display advertisements to consumers, based on what webpages they have visited in the past. As consumers, we have become used to getting a vast amount of information and services on the Web seemingly for free. Of course, the "for free" part is very often due to the existence or promise of revenue from online advertising, similar to how broadcast television is "free." Let's consider *display advertising*—the ads that appear on the top, sides, and bottom of pages full of content that we are reading or otherwise consuming.

Display advertising is different from search advertising (e.g., the ads that appear with the results of a Google search) in an important way: for most webpages, the user has not typed in a phrase related to exactly what she is looking for. Therefore, the targeting of an advertisement to the user needs to be based on other sorts of inference. For several

chapters now we have been talking about a particular sort of inference: inferring the value of an instance's target variable from the values of the instance's features. Therefore, we could apply the techniques we already have covered to infer whether a particular user would be interested in an advertisement. In this chapter we will introduce a different way of looking at the problem, that has wide applicability and is quite easy to apply.

Let's define our ad targeting problem more precisely. What will be an instance? What will be the target variable? What will be the features? How will we get the training data?

Let's assume that we are working for a very large content provider (a "publisher") who has a wide variety of content, sees many online consumers, and has many opportunities to show advertisements to these consumers. For example, Yahoo! has a vast number of different advertisement-supported web "properties," which we can think of as different "content pieces." In addition, recently (as of this writing) Yahoo! agreed to purchase the blogging site Tumblr, which has 50 billion blog posts across over 100 million blogs. Each of these might also be seen as a "content piece" that gives some view into the interests of a consumer who reads it. Similarly, Facebook might consider each "Like" that a consumer makes as a piece of evidence regarding the consumer's tastes, which might help target ads as well.

For simplicity, assume we have one advertising campaign for which we would like to target some subset of the online consumers that visit our sites. This campaign is for the upscale hotel chain, Luxhote. The goal of Luxhote is for people to book rooms. We have run this campaign in the past, selecting online consumers randomly. We now want to run a targeted campaign, hopefully getting more bookings per dollar spent on ad impressions.¹

Therefore, we will consider a consumer to be an instance. Our target variable will be: did/will the consumer book a Luxhote room within one week after having seen the Luxhote advertisement? Through the magic of browser cookies,² in collaboration with Luxhote we can observe which consumers book rooms. For training, we will have a binary value for this target variable for each consumer. In use, we will estimate the probability that a consumer will book a room after having seen an ad, and then, as our budget allows, target some subset of the highest probability consumers.

We are left with a key question: what will be the features we will use to describe the consumers, such that we might be able to differentiate those that are more or less likely to be good customers for Luxhote? For this example, we will consider a consumer to be described by the set of content pieces that we have observed her to have viewed (or Liked) previously, again as recorded via browser cookies or some other mechanism. We

1. An ad *impression* is when an ad is displayed somewhere on a page, regardless of whether a user clicks it.

2. A browser exchanges small amounts of information ("cookies") with the sites that are visited, and saves site-specific information that can be retrieved later by the same website.

have many different kinds of content: finance, sports, entertainment, cooking blogs, etc. We might pick several thousand content pieces that are very popular, or we may consider hundreds of millions. We believe that some of these (e.g., finance blogs) are more likely to be visited by good prospects for Luxhote, while other content pieces are seen as less likely (e.g., a tractor-pull fan page).

However, for this exercise we do not want to rely on our presumptions about such content, nor do we have the resources to estimate the evidence potential for each content piece manually. Furthermore, while humans are quite good at using our knowledge and common sense to recognize whether evidence is likely to be “for” or “against,” humans are notoriously bad at estimating the precise *strength* of the evidence. We would like our historical data to estimate both the direction and the strength of the evidence. We next will describe a very broadly applicable framework both for evaluating the evidence, and for combining it to estimate the resulting likelihood of class membership (here, the likelihood that a consumer will book a room after having seen the ad).

It turns out that there are many other problems that fit the mold of our example: classification/class probability estimation problems where each instance is described by a set of pieces of evidence, possibly taken from a very large total collection of possible evidence. For example, text document classification fits exactly (which we’ll discuss next in [Chapter 10](#)). Each document is a collection of words, from a very large total vocabulary. Each word can possibly provide some evidence for or against the classification, and we would like to combine the evidence. The techniques that we introduce next are exactly those used in many spam detection systems: an instance is an email message, the target classes are **spam** or **not-spam**, and the features are the words and symbols in the email message.

Combining Evidence Probabilistically



More math than usual ahead

To discuss the ideas of combining evidence probabilistically, we need to introduce some probability notation. You do not have to have learned (or remember) probability theory—the notions are quite intuitive, and we will not get beyond the basics. The notation allows us to be precise. It might look like there’s a lot of math in what follows, but you’ll see that it’s quite straightforward.

We are interested in quantities such as the probability of a consumer booking a room after being shown an ad. We actually need to be a little more specific: some particular consumer? Or just any consumer? Let’s start with just any consumer: what is the probability that if you show an ad to just any consumer, she will book a room? As this is our desired classification, let’s call this quantity C . We will represent the probability of an

event C as $p(C)$. If we say $p(C) = 0.0001$, that means that if we were to show ads randomly to consumers, we would expect about 1 in 10,000 to book rooms.³

Now, we are interested in the probability of C given some evidence E , such as the set of websites visited by a *particular* consumer. The notation for this quantity is $p(C|E)$, which is read as “the probability of C given E ,” or “the probability of C conditioned on E .” This is an example of a conditional probability, and the “|” is sometimes called the “conditioning bar.” We would expect that $p(C|E)$ would be different based on different collections of evidence E —in our example, different sets of websites visited.

As mentioned above, we would like to use some labeled data, such as the data from our randomly targeted campaign, to associate different collections of evidence E with different probabilities. Unfortunately, this introduces a key problem. For any particular collection of evidence E , we probably have not seen enough cases with exactly that same collection of evidence to be able to infer the probability of class membership with any confidence. In fact, we may not have seen this particular collection of evidence at all! In our example, if we are considering thousands of different websites, what is the chance that in our training data we have seen a consumer with *exactly* the same visiting patterns as a consumer we will see in the future? It is infinitesimal. Therefore, what we will do is to consider the different pieces of evidence separately, and then combine evidence. To discuss this further, we need a few facts about combining probabilities.

Joint Probability and Independence

Let’s say we have two events, A and B . If we know $p(A)$ and $p(B)$, can we say what is the probability that both A and B occur? Let’s call that $p(AB)$. This is called the *joint* probability.

There is one special case when we can: if events A and B are *independent*. A and B being independent means that knowing about one of them tells you nothing about the likelihood of the other. The typical example used to illustrate independence is rolling a fair die; knowing the value of the first roll tells you nothing about the value of the second. If event A is “roll #1 shows a six” and event B is “roll #2 shows a six”, then $p(A) = 1/6$ and $p(B) = 1/6$, and importantly, even if we *know* that roll #1 shows a six, still $p(B) = 1/6$. In this case, the events are independent, and in the case of independent events, $p(AB) = p(A) \cdot p(B)$ —we can calculate the probability of the “joint” event AB by multiplying the probabilities of the individual events. In our example, $p(AB) = 1/36$.

However, we cannot in general compute the probabilities of joint events in this way. If this isn’t clear, think about the case of rolling a trick die. In my pocket I have six trick dice. Each trick die has one of the numbers from one to six on all faces—all faces show

3. This is not necessarily a reasonable response rate for any particular advertisement, just an illustrative example. Purchase rates attributable to online advertisements generally seem very small to those outside the industry. It is important to realize that the cost of placing one ad often is quite small as well.

the same number. I pull a die at random from my pocket, and then roll it twice. In this case, $p(A) = p(B) = 1/6$ (because I could have pulled any of the six dice out with equal likelihood). However, $p(AB) = 1/6$ as well, because the events are completely dependent! If the first roll is a six, so will be the second (and vice versa).

The general formula for combining probabilities that takes care of dependencies between events is:

Equation 9-1. Joint probability using conditional probability

$$p(AB) = p(A) \cdot p(B | A)$$

This is read as: the probability of A and B is the probability of A times the probability of B given A . In other words, given that you know A , what is the probability of B ? Take a minute to make sure that has sunk in.

We can illustrate with our two dice examples. In the independent case, since knowing A tells us nothing about $p(B)$, then $p(B|A) = p(B)$, and we get our formula from above, where we simply multiply the individual probabilities. In our trick die case, $p(B|A) = 1.0$, since if the first roll was a six, then the second roll is guaranteed to be a six. Thus, $p(AB) = p(A) \cdot 1.0 = p(A) = 1/6$, just as expected. In general, events may be completely independent, completely dependent, or somewhere in between. In the latter case, knowing something about one event changes the likelihood of the other. In all cases, our formula $p(AB) = p(A) \cdot p(B|A)$ combines the probabilities properly.

We've gone through this detail for a very important reason. This formula is the basis for one of the most famous equations in data science, and in fact in science generally.

Bayes' Rule

Notice that in $p(AB) = p(A)p(B|A)$ the order of A and B seems rather arbitrary—and it is. We could just as well have written:

$$p(AB) = p(B) \cdot p(A | B)$$

This means:

$$p(A) \cdot p(B | A) = p(AB) = p(B) \cdot p(A | B)$$

And so:

$$p(A) \cdot p(B | A) = p(B) \cdot p(A | B)$$

If we divide both sides by $p(A)$ we get:

$$p(B | A) = \frac{p(A | B) \cdot p(B)}{p(A)}$$

Now, let's consider B to be some hypothesis that we are interested in assessing the likelihood of, and A to be some evidence that we have observed. Renaming with H for hypothesis and E for evidence, we get:

$$p(H | E) = \frac{p(E | H) \cdot p(H)}{p(E)}$$

This is the famous *Bayes' Rule*, named after the Reverend Thomas Bayes who derived a special case of the rule back in the 18th century. Bayes' Rule says that we can compute the probability of our hypothesis H given some evidence E by instead looking at the probability of the evidence given the hypothesis, as well as the unconditional probabilities of the hypothesis and the evidence.



Note: Bayesian methods

Bayes' Rule, combined with the important fundamental principle of thinking carefully about conditional independencies, are the foundation for a vast amount of more advanced data science techniques that we will not cover in this book. These include Bayesian networks, probabilistic topic models, probabilistic relational models, Hidden Markov Models, Markov random fields, and others.

Importantly, the last three quantities may be easier to determine than the quantity of ultimate interest—namely, $p(H|E)$. To see this, consider a (simplified) example from medical diagnosis. Assume you're a doctor and a patient arrives with red spots. You guess (hypothesize) that the patient has measles. We would like to determine the probability of our hypothesized diagnosis ($H = \text{measles}$), given the evidence ($E = \text{red spots}$). In order to directly estimate $p(\text{measles}|\text{red spots})$ we would need to think through all the different reasons a person might exhibit red spots and what proportion of them would be measles. This is likely impossible even for the most broadly knowledgeable physician.

However, consider instead the task of estimating this quantity using the righthand side of Bayes' Rule.

- $p(E|H)$ is the probability that one has red spots given that one has measles. An expert in infectious diseases may well know this or be able to estimate it relatively accurately.

- $p(H)$ is simply the probability that someone has measles, without considering any evidence; that's just the prevalence of measles in the population.
- $p(E)$ is the probability of the evidence: what's the probability that someone has red spots—again, simply the prevalence of red spots in the population, which does not require complicated reasoning about the different underlying causes, just observation and counting.

Bayes' Rule has made estimating $p(H|E)$ much easier. We need three pieces of information, but they're much easier to estimate than the original value is.



$p(E)$ may still be difficult to compute. In many cases, though, it does not have to be computed, because we are interested in comparing the probabilities of different hypotheses given the same evidence. We will see this later.

Applying Bayes' Rule to Data Science

It is possibly quite obvious now that Bayes' Rule should be critical in data science. Indeed, a very large portion of data science is based on “Bayesian” methods, which have at their core reasoning based on Bayes' Rule. Describing Bayesian methods broadly is well beyond the scope of this book. We will introduce the most fundamental ideas, and then show how they apply in the most basic of Bayesian techniques—which is used a great deal. Let's rewrite Bayes' Rule yet again, but now returning to classification.

Equation 9-2. Bayes Rule for classification

$$p(C = c \mid \mathbf{E}) = \frac{p(\mathbf{E} \mid C = c) \cdot p(C = c)}{p(\mathbf{E})}$$

In **Equation 9-2**, we have four quantities. On the lefthand side is the quantity we would like to estimate. In the context of a classification problem, this is the probability that the target variable C takes on the class of interest c *after* taking the evidence \mathbf{E} (the **vector** of feature values) into account. This is called the *posterior* probability.

Bayes' Rule decomposes the posterior probability into the three quantities that we see on the righthand side. We would like to be able to compute these quantities from the data:

1. $p(C = c)$ is the “*prior*” probability of the class, i.e., the probability we would assign to the class before seeing any evidence. In Bayesian reasoning generally, this could come from several places. It could be (i) a “subjective” prior, meaning that it is the

belief of a particular decision maker based on all her knowledge, experience, and opinions; (ii) a “prior” belief based on some previous application(s) of Bayes’ Rule with other evidence, or (iii) an unconditional probability inferred from data. The specific method we introduce below takes approach (iii), using as the *class prior* the “base rate” of c —the prevalence of c in the population as a whole. This is calculated easily from the data as the percentage of all examples that are of class c .

2. $p(\mathbf{E} | C = c)$ is the *likelihood* of seeing the evidence \mathbf{E} —the particular features of the example being classified—when the class $C = c$. One might see this as a “generative” question: if the world (the “data generating process”) generated an instance of class c , how often would it look like \mathbf{E} ? This likelihood might be calculated from the data as the percentage of examples of class c that have feature vector \mathbf{E} .
3. Finally, $p(\mathbf{E})$ is the likelihood of the evidence: how common is the feature representation \mathbf{E} among all examples? This might be calculated from the data as the percentage occurrence of \mathbf{E} among all examples.

Estimating these three values from training data, we could calculate an estimate for the posterior $p(C = c | \mathbf{E})$ for a particular example in use. This could be used directly as an estimate of class probability, possibly in combination with costs and benefits as described in [Chapter 7](#). Alternatively, $p(C = c | \mathbf{E})$ could be used as a score to rank instances (e.g., estimating those that are most likely to respond to our advertisement). Or, we could choose as the classification the maximum $p(C = c | \mathbf{E})$ across the different values c .

Unfortunately, we return to the major difficulty we mentioned above, which keeps [Equation 9-2](#) from being used directly in data mining. Consider \mathbf{E} to be our usual vector of attribute values $\langle e_1, e_2, \dots, e_k \rangle$, a possibly large, specific collection of conditions. Applying [Equation 9-2](#) directly would require knowing the $p(\mathbf{E} | c)$ as $p(e_1 \wedge e_2 \wedge \dots \wedge e_k | c)$.⁴ This is very specific and very difficult to measure. We may never see a specific example in the training data that exactly matches a given \mathbf{E} in our testing data, and even if we do it may be unlikely we’ll see enough of them to estimate a probability with any confidence.

Bayesian methods for data science deal with this issue by making assumptions of probabilistic independence. The most broadly used method for dealing with this complication is to make a particularly strong assumption of independence.

Conditional Independence and Naive Bayes

Recall from above the notion of independence: two events are independent if knowing one does not give you information on the probability of the other. Let’s extend that notion ever so slightly.

4. The \wedge operator means “and.”

Conditional independence is the same notion, except using conditional probabilities. For our purposes, we will focus on the class of the example as the condition (since in [Equation 9-2](#) we are looking at the probability of the evidence *given* the class). Conditional independence is directly analogous to the unconditional independence we discussed above. Specifically, without assuming independence, to combine probabilities we need to use [Equation 9-1](#) from above, augmented with the $|C$ condition:

$$p(AB | C) = p(A | C) \cdot p(B | AC)$$

However, as above, if we assume that A and B are conditionally independent given C ,⁵ we can now combine the probabilities much more easily:

$$p(AB | C) = p(A | C) \cdot p(B | C)$$

This makes a huge difference in our ability to compute the probabilities from the data. In particular, for the conditional probability $p(\mathbf{E} | C=c)$ in [Equation 9-2](#), let's assume that the attributes are conditionally independent, given the class. In other words, in $p(e_1 \wedge e_2 \wedge \dots \wedge e_k | c)$, each e_i is independent of every other e_j given the class c :

$$\begin{aligned} p(\mathbf{E} | c) &= p(e_1 \wedge e_2 \wedge \dots \wedge e_k | c) \\ &= p(e_1 | c) \cdot p(e_2 | c) \cdots p(e_k | c) \end{aligned}$$

Each of the $p(e_i | c)$ terms can be computed directly from the data, since now we simply need to count up the proportion of the time that we see individual feature e_i in the instances of class c , rather than looking for an entire matching feature vector. There are likely to be relatively many occurrences of e_i .⁶ Combining this with [Equation 9-2](#) we get the *Naive Bayes equation* as shown in [Equation 9-3](#).

Equation 9-3. Naive Bayes equation

$$p(c | \mathbf{E}) = \frac{p(e_1 | c) \cdot p(e_2 | c) \cdots p(e_k | c) \cdot p(c)}{p(\mathbf{E})}$$

This is the basis of the *Naive Bayes classifier*. It classifies a new example by estimating the probability that the example belongs to each class and reports the class with highest probability.

5. This is a weaker assumption than assuming unconditional independence, by the way.

6. And in the cases where there are not we can use a statistical correction for small counts. The difference is that we will not be doing that for all the evidence, as we would have considering the entire E .

If you will allow two paragraphs on a technical detail: at this point you might notice the $p(\mathbf{E})$ in the denominator of [Equation 9-3](#) and say, whoa there—if I understand you, isn't that going to be almost as difficult to compute as $p(\mathbf{E} | C)$? It turns out that generally $p(\mathbf{E})$ never actually has to be calculated, for one of two reasons. First, if we are interested in classification, what we mainly care about is: of the different possible classes c , for which one is $p(C | \mathbf{E})$ the greatest? In this case, \mathbf{E} is the same for all, and we can just look to see which numerator is larger.

In cases where we would like the actual probability estimates, we still can get around computing $p(\mathbf{E})$ in the denominator. This is because the classes often are mutually exclusive and exhaustive, meaning that every instance will belong to one and only one class. In our Luxhote example, a consumer either books a room or does not. Informally, if we see evidence \mathbf{E} it belongs either to c_0 or c_1 . Mathematically:

$$\begin{aligned} p(\mathbf{E}) &= p(\mathbf{E} \wedge c_0) + p(\mathbf{E} \wedge c_1) \\ &= p(\mathbf{E} | c_0) \cdot p(c_0) + p(\mathbf{E} | c_1) \cdot p(c_1) \end{aligned}$$

Our independence assumption allows us to rewrite this as:

$$\begin{aligned} p(\mathbf{E}) &= p(e_1 | c_0) \cdot p(e_2 | c_0) \cdots p(e_k | c_0) \cdot p(c_0) \\ &\quad + p(e_1 | c_1) \cdot p(e_2 | c_1) \cdots p(e_k | c_1) \cdot p(c_1) \\ p(\mathbf{E}) &= p(e_1 | c_0) \cdot p(e_2 | c_0) \cdots p(e_k | c_0) \cdot p(c_0) + p(e_1 | c_1) \cdot p(e_2 | c_1) \cdots p(e_k | c_1) \cdot p(c_1) \end{aligned}$$

Combining this with [Equation 9-3](#), we get a version of the Naive Bayes equation with which we can compute the posterior probabilities easily from the data:

$$p(c_0 | \mathbf{E}) = \frac{p(e_1 | c_0) \cdot p(e_2 | c_0) \cdots p(e_k | c_0) \cdot p(c_0)}{p(e_1 | c_0) \cdot p(e_2 | c_0) \cdots p(e_k | c_0) + p(e_1 | c_1) \cdot p(e_2 | c_1) \cdots p(e_k | c_1)}$$

Although it has lots of terms in it, each one is either the evidence “weight” of some particular individual piece of evidence, or a class prior.

Advantages and Disadvantages of Naive Bayes

Naive Bayes is a very simple classifier, yet it still takes all the feature evidence into account. It is very efficient in terms of storage space and computation time. Training consists only of storing counts of classes and feature occurrences as each example is seen. As mentioned, $p(c)$ can be estimated by counting the proportion of examples of class c among all examples. $p(e_i | c)$ can be estimated by the proportion of examples in class c for which feature e_i appears.

In spite of its simplicity and the strict independence assumptions, the Naive Bayes classifier performs surprisingly well for classification on many real-world tasks. This is because the violation of the independence assumption tends not to hurt classification performance, for an intuitively satisfying reason. Specifically, consider that two pieces of evidence are actually strongly dependent—what does that mean? Roughly, that means that when we see one we’re also likely to see the other. Now, if we treat them as being independent, we’re going to see one and say “there’s evidence for the class” and see the other and say “there’s more evidence for the class.” So, to some extent we’ll be double-counting the evidence. *However*, as long as the evidence is generally pointing us in the right direction, for classification the double-counting won’t tend to hurt us. In fact, it will tend to make the probability estimates more extreme in the correct direction: the probability will be overestimated for the correct class and underestimated for the incorrect class(es). But for classification we’re choosing the class with the greatest probability estimate, so making them more extreme in the correct direction is OK.

This does become a problem, though, if we’re going to be using the probability estimates themselves—so Naive Bayes should be used with caution for actual decision-making with costs and benefits, as discussed in [Chapter 7](#). Practitioners do use Naive Bayes regularly for ranking, where the actual values of the probabilities are not relevant—only the relative values for examples in the different classes.

Another advantage of Naive Bayes is that it is naturally an “incremental learner.” An incremental learner is an induction technique that can update its model one training example at a time. It does not need to reprocess all past training examples when new training data become available.

Incremental learning is especially advantageous in applications where training labels are revealed in the course of the application, and we would like the model to reflect this new information as quickly as possible. For example, consider creating a personalized junk email classifier. When I receive a piece of junk email, I can click the “junk” button in my browser. Besides removing this email from my Inbox, this also creates a training data point: a positive instance of spam. It would be quite useful if the model that is classifying my email could be updated on the fly, and thereby immediately start classifying other similar emails as being spam. Naive Bayes is the basis of many personalized spam detection systems, such as the one in Mozilla’s Thunderbird.

Naive Bayes is included in nearly every data mining toolkit and serves as a common baseline classifier against which more sophisticated methods can be compared. We have discussed Naive Bayes using binary attributes. The basic idea presented above can be extended easily to multi valued categorical attributes, as well as to numeric attributes, as you can read about in a textbook on data mining algorithms.