Week 11: Introduction to (binary) classification

Thus far we have seen many different instances of models for analyzing data. But we have not yet discussed one big category of models -- classifiers. Classifiers are built to solve the problem of classification (as the name obviously implies): suppose I've observed some data that could have come from one of several discrete groups (classes), how can I guess from which class the data came?

As we've done for previous models, let's formulate this in terms of Bayesian probabilities: We have probability models $H_1 \ldots H_m$ for m classes, and we want to figure out how to classify a data point x. Bayes tells us that we want to calculate $P(H_i \mid x)$:

$$P(H_i \mid x) = \frac{P(x \mid H_i)P(H_i)}{\sum_j P(x \mid H_j)P(H_j)}$$

If we knew the likelihoods $P(x \mid H_i)$ and the priors $P(H_i)$, then the $P(H_i \mid x)$ are literally the probabilities that x came from class H_i. If you were betting, this is how you'd want to calculate your odds. For example, in our fair vs. loaded dice example, we could really imagine knowing $P(x \mid \text{fair})$ (by definition $\frac{1}{6}$), $P(x \mid \text{loaded})$ (by rolling a loaded die a bazillion times and counting frequencies), and $P(\text{fair})$ versus $P(\text{loaded})$ (if the game is to pull a die at random out of a bag, and we happen to know how many dice of each type are in the bag). If the game is, I draw a die at random from the bag, and roll it, you could precisely calculate the probability that the die was fair vs. loaded. Using this probability, I could then predict the class (fair vs. loaded) to which the die belonged.

Log-odds scores

However, in many cases I don't know the priors $P(H_i)$. I just have a data sample x, and I can calculate the likelihoods $P(x \mid H_i)$ for two or more models. Now I can only calculate the posterior probabilities up to an unknown constant. How should I assign a "score" to data sample x, in order to decide which class it belongs to?

Suppose we're talking binary classification, and I have two models H_1 and H_2. Then it turns out that a good score to calculate is the **log-odds score**, also known as a **log likelihood ratio (LLR)**:

$$\text{LLR} = \log \frac{P(x \mid H_1)}{P(x \mid H_2)}$$

The LLR is positive if H_1 fits the data better, negative if H_2 is better, and zero if the two models explain the data equally well.

Why take the logarithm? As we've seen before, there are good practical reasons. Remember that probabilities can easily become so small that they underflow the computer's floating point representation. And log probabilities are easier to work with mathematically in most cases because they break up products into sums.

Derivation of log-odds scores

LLR scores are extremely common in data analysis, including biomedical data analysis. They have a justification you can derive easily.

Starting from Bayes' theorem, dividing through first by $P(H_1)$ in both numerator and denominator, then by $P(x \mid H_2)$:

$$\begin{aligned} P(H_1 \mid x) &= \frac{P(x \mid H_1)P(H_1)}{P(x \mid H_1)P(H_1) + P(x \mid H_2)P(H_2)} \\ &= \frac{P(x \mid H_1)}{P(x \mid H_1) + P(x \mid H_2)\frac{P(H_2)}{P(H_1)}} \\ &= \frac{\frac{P(x|H_1)}{P(x|H_2)}}{\frac{P(x|H_1)}{P(x|H_2)} + \frac{P(H_2)}{P(H_1)}} \end{aligned}$$

This is showing us that we can rearrange the posterior probability in terms of an odds ratio $\frac{P(x|H_1)}{P(x|H_2)}$ that we can calculate, and a prior ratio $\frac{P(H_2)}{P(H_1)}$ that we can't.

Suppose we go ahead and define an LLR score as z, then:

$$P(H_1 \mid x) = \frac{e^z}{e^z + \frac{P(H_2)}{P(H_1)}}$$

You might recognize this as a **sigmoid function** $f(z) = \frac{e^z}{e^z + c}$ for a constant $c = \frac{P(H_2)}{P(H_1)}$. For high scores, it asymptotically approaches 1; for low scores, it approaches 0; and it rises through 0.5 at $z = \log \frac{P(H_2)}{P(H_1)}$. If the two models are a priori equiprobable, the sigmoid curve is centered with its midpoint at $z = 0$. That is, an LLR score of 0 means that the two models are a posteriori equiprobable.

The prior log odds ratio acts as a constant offset on the score. If H_2 is a priori more probable, then $\log \frac{P(H_2)}{P(H_1)}$ is positive -- the sigmoid curve shifts to the right, signifying that it takes more LLR score to convince you that the data favor H_1, even though H_1 was less probable a priori.

Sometimes $\frac{P(x|H_1)}{P(x|H_2)}$ is called the *Bayes factor*.

Note that we have defined the log-odds score generally in terms of two arbitrary models. That's because it is in fact a very general formulation. How you get the log-odds scores is up to you and your imagination (in terms of possible models you want to compare). One common way to derive the log-odds score is assuming that the score is a linear function of some features of the data x -- this is known as logistic regression, which we will learn about soon).

Evaluating classification performance: ROC plots

Suppose we've developed a score for a binary classifier -- whether it's a log-odds score as a log likelihood ratio, or something arbitrary. We could choose to set a score threshold t such that if $z > t$ we assign it to class H_1, and otherwise we assign it to H_2. Suppose we're looking for a particular class of interesting things ("positives", class H_1) against a background of uninteresting things ("negatives", class H_2).

Indeed H_2 might be so boring and uninteresting that it's our model of "nothing is going on with x except random chance" (this would be a null hypothesis).

Where should we set the threshold t, to achieve the best results in discriminating positives from negatives?

There is no one answer to this question, because it depends on whether we care more about not missing any positives, or about not making a mistake of misclassifying a negative as a positive.

Imagine a little 2x2 matrix of the truth versus our classification:

- **true positive (TP)** is when we call x positive and it is one;
- **true negative (TN)** is when we call x negative and it is;
- **false positive (FP)** is when we call x positive and it's really noise
- **false negative (FN)** is when we call x negative and it's really signal

From these counts, we can calculate:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{all positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{all negatives}} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

"Sensitivity" goes by other names, including the **true positive rate**, or **recall**.

1 - specificity is also called the **false positive rate**.

If we set the score threshold lower (less stringent), we classify more stuff positive: our sensitivity increases, but our specificity drops. We can ultimately achieve 100% sensitivity, but 0% specificity, by calling everything a positive. If we set the threshold higher, the sensitivity drops, but our specificity increases; we achieve 100% specificity, but 0% sensitivity, by calling everything a negative.

A plot of sensitivity versus false positive rate (both from 0 to 1.0) for all possible choices of threshold is called a **receiver operating characteristic plot (or curve) (ROC plot)**. A perfect ROC plot leaps immediately to 100% sensitivity at 0% FPR. Random guessing gives you a diagonal line.

The name "receiver operating characteristic" is a historical artifact. It's a military term that arose in WWII for radar receivers distinguishing blips as enemy vs. friendly aircraft. Correct friend/foe classification is a pretty important operating characteristic for a military radar receiver.

ROC caveat

When you're in a situation of detecting a small number of positives against a background of a large number of negatives -- a needle in the haystack problem, common in large-scale biomedical data analysis -- a ROC plot can give you a misleading sense of the accuracy of a classification procedure. The number of false positives you detect will scale with the number of negatives you screen. You can have a procedure with a high specificity that still detects an unacceptable number of false positives, if the positives you're looking for are rare.

For this reason, there's two other common statistics that people calculate, which take into account the relative frequency of positives versus negatives:

$$\text{Positive Predictive Value (PPV)} = \frac{\text{TP}}{\text{classified as positive}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{False Discovery Rate (FDR)} = \frac{\text{FP}}{\text{classified as positive}} = \frac{\text{FP}}{\text{TP} + \text{FP}}$$

Of the set of things that you classify as positive, PPV is the fraction that you're right about, and FDR is the fraction that you're wrong about. Notice that FDR = 1 - PPV. PPV is also commonly called **precision**.

For the reason above, it is quite possible to have a low false positive rate but an unacceptably high FDR.

The wikipedia page on sensitivity and specificity has a good discussion on these topics.

Naive Bayes classification

As an example of building a simple binary classifier, let's consider the case of a common gene sequence analysis problem, where we want to be able to classify a gene sequence as belonging to one of two classes (for example, whether a sequence came from a healthy vs. a tumor cell). To use our Bayes formulation to build this classifier, we would want to express the probability of a sequence $x_1 . . x_L$ of length L. Let's make it a DNA sequence, with an alphabet of 4 nucleotides ACGT, to be specific; but it could also be a protein sequence with an alphabet of 20 amino acids, or indeed some other kind of symbol string (such as a document).

To classify a the sequence, we want to express $P(\mathbf{x} \mid H_1)$ and $P(\mathbf{x} \mid H_2)$ for two probability models of the two kinds of sequences. Using those likelihoods, if we also know the prior probabilities $P(H_1)$ and $P(H_2)$, we could calculate a posterior probability for assigning a given sequence to each model. Even if we don't know those priors (which would often be the case), we can calculate a log-odds score $\log \frac{P(\mathbf{x}|H_1)}{P(\mathbf{x}|H_2)}$ as a well-justified measure of the evidence for the sequence matching model 1 compared to 2.

For interesting sequence lengths L, we aren't going to be able to express $P(x_1 . . x_L \mid H)$ directly, because we'd need too many parameters: 4^L of them, for DNA. We need to make assumptions: we need to make independence assumptions that let us factorize the joint probability $P(x_1 . . x_L \mid H)$ in terms of a smaller number of parameters. This assumption of independence is what lends the term "naive" to the naive Bayes classifier we can build from these models.

The i.i.d. sequence model

For example, we could simply assume that each nucleotide a is drawn from a probability distribution $p(a)$ over the four nucleotides, and each nucleotide x_i in the sequence is independently drawn from the identical distribution. Then:

$$P(x_1 . . x_L) \simeq \prod_{i=1}^{L} p(x_i)$$

This is about the simplest model you can imagine. It is called an *independent, identically distributed* sequence model, or just i.i.d. for short. If someone says an "i.i.d. sequence", this is what they mean.

You could make two i.i.d. models H_1 and H_2 with different base composition -- one with high probabilities for AT-rich sequence, the other with high probabilities for GC-rich sequence -- and you'd have the basis for doing a probabilistic binary classification of sequences based on their nucleotide composition.

Let's look in detail at how an i.i.d. model is derived, in terms of probability algebra for manipulating joint and conditional probabilities. Suppose (for sake of example) we have a sequence of six residues, $UVWXYZ$, where each symbol represents a random variable that takes a value a/c/g/t. We want to express a joint probability $P(UVWXYZ)$. We can exactly factor that joint probability into a series of conditional probabilities like so:

$$P(UVWXYZ) = P(Z \mid UVWXY)P(UVWXY) \quad = P(Z \mid UVWXY)P(Y \mid UVWX)P(UVWX) \quad \text{and so on...} \quad = P(Z \mid UVWXY)P(Y \mid$$

All exact, so far; no approximation. All we did was converted joint probabilities to conditional probabilities, repeatedly.

Now we can look at terms like, e.g., $P(Z \mid UVWXY)$ and make independence assumptions. Z depends on U, V, W, X, Y? No it doesn't, we can assume: we can remove one, some, or all of the dependencies, and that corresponds to stating an assumption that the nucleotide at Z doesn't depend on the nucleotide at the variable (position) we remove -- or at least, doesn't depend enough that we care, for the purposes of making a model, given the cost/benefit tradeoff of balancing the number of free parameters we need to estimate, versus what we gain from a more exact model.

The most extreme and simple thing we could assume is that Z doesn't depend on any other position: $P(Z \mid UVWXY) \simeq P(Z)$. Make that assumption all the way down the chain and we have $P(UVWXYZ) = P(U)P(V)P(W)P(X)P(Y)P(Z)$. That's the independent part of an i.i.d. model.

Position-specific scoring models for motifs

We would still have different distributions at the six positions. Maybe position U is often an A, position V is usually C or G, position W is equiprobably any of the four, and so on. You can imagine if we had a multiple sequence alignment of example sequences, we could count up the frequencies we observe in each position, and use those as model parameters. We'd need $4L$ parameters for this model, but that's a much more compact representation than the 4^L we needed before making an independence assumption. This is a common model of short, fixed-length DNA sequence motifs: people call it a position-specific scoring model (PSSM) or a position weight matrix (PWM).

But to simplify our model still more, we could state a further assumption that each of the six positions is drawn from the same nucleotide frequency distribution $p(a)$ -- and that identically distributed assumption gets us to the i.i.d. model.

First order Markov models of dinucleotide composition

Suppose we're ok with the identically distributed assumption -- we're not trying to capture position-specific statistical information (like in one particular conserved DNA sequence motif), we're trying to capture overall statistical properties and biases in a DNA sequence. But suppose that complete independence seems too strong.

For example, in coding sequences, individual bases aren't independent; they come in triplets because of the genetic code. We could do this, to factor a sequence of length L in terms of conditional probabilities of triplets (codons), followed by an independence assumption that codons are statistically independent, i.e.:

$$P(UVWXYZ) = P(XYZ \mid UVW)P(UVW) \qquad \simeq P(XYZ)P(UVW)$$

But this is a pretty specialized model that depends on the fact that there's a reading frame: we know exactly where to break the sequence into triplets. What if there's no frame?

For example, many eukaryotic genome sequences show a strong depletion of CG dinucleotides. If you count up mononucleotide and dinucleotide frequencies, you see that $p(cg) << p(c)p(g)$. If we want to capture the fact that G is disfavored if the previous nucleotide was a C, we need something more than just an i.i.d. model. (And C is disfavored after a G, because of the other strand; the reverse complement of CG is GC.) Biologists refer to this as CpG bias, where the p in CpG indicates the phosphodiester linkage along a DNA strand -- we talk about CG composition so much in other contexts, we say CpG to keep it straight that we're talking about adjacent nucleotides.

If you tried to make a model that takes CpG bias into account, you might think you could do something with multiplying dinucleotide frequencies together, but probability algebra doesn't work that way.

So let's back up. Let's go back to:

$$P(UVWXYZ) = P(Z \mid UVWXY)P(Y \mid UVWX)P(X \mid UVW)P(W \mid UV)P(U \mid V)P(V)$$

Dinucleotide composition bias means that we're not going to assume that Z is independent of its neighbor Y, but we're still going to drop the rest:

$$P(UVWXYZ) \simeq P(Z \mid Y)P(Y \mid X)P(X \mid W)P(W \mid V)P(V \mid U)P(U)$$

If we make the identically distributed assumption, that the same dinucleotide bias holds everywhere in the sequence, then we can get to a general form:

$$P(x_1 . . x_L) \simeq p(x_1) \prod_{i=2}^{L} p(x_i \mid x_{i-1})$$

We have two kinds of parameters. The main parameters are the $p(b \mid a)$ parameters for the probability that nucleotide b follows a. We also need to get the chain started somehow, so we need $p(a)$ for the nucleotide probabilities at the first position.

Building the classifier from our naive models

Now that we have the expression for our sequence models, we can then simply calculate the LLR for two different sequence models once we have observed a new sequence. From this LLR score, we can classify the new sequence as one model or the other based on whatever score threshold we've deemed is most suitable to our problem and data.

To *train* our naive Bayes sequence classifier, we just need to find values for $p(A)$, $p(T)$, $p(G)$, $p(C)$, in the case of the i.i.d. model, or $P(A \mid A)$, $P(A \mid T)$, $P(A \mid G)$, $P(A \mid G)$, $P(T \mid A)$, ... for all combinations of dinucleotides, by calculating them from a large database of sequences of both classes. This gives a collection of all of the probabilities that we need. Then when we get a new sequence, for each class model we simply look up the probabilities of each nucleotide in their respective collection and multiply them all together to get total likelihoods, $P(x \mid H_1)$ and $P(x \mid H_2)$, from which we alculate our LLR score. This naive Bayes approach works for any type of model, not just sequences, it's just a matter of being able to express your model as a product of probabilities.