Homework 4: Diagnosing heart anomalies

Bart Massey

2011/02/23

It would be pretty cool if a computer could help a physician diagnose heart anomalies; even cooler if you could help to write the software. We are going to build a supervised machine learner and run it on some real heart anomaly data, sliced and diced in various ways.

The method we will use is Naïve Bayesian learning. As discussed (with demo) in class, this is an approach that involves counting feature occurrences in the learning phase, and then using these counts in the classification phase.

We will be looking at a dataset that has a binary (0, 1) classification, and m binary features. The math is pretty simple. Recall Bayes's Rule:

$$\begin{aligned}
\Pr(H|E) &= \Pr(H|E_1 \wedge E_0 \wedge \ldots \wedge E_m) && (1)\\
&= \frac{\Pr(E_1 \wedge E_0 \wedge \ldots \wedge E_m|H)\Pr(H)}{\Pr(E_1 \wedge E_0 \wedge \ldots \wedge E_m)} && (2)\\
&= \frac{\Pr(E_1|H) \cdot \Pr(E_0|H) \cdot \ldots \cdot \Pr(E_m|H) \cdot \Pr(H)}{\Pr(E_1) \cdot \Pr(E_0) \cdot \ldots \cdot \Pr(E_m)} && (3)\\
&= \frac{\Pr(H) \cdot \prod_i \Pr(E_i|H)}{\prod_i \Pr(E_i)} && (4)
\end{aligned}$$

Equation 3 was obtained by naïvely assuming that the evidence is mutually independent.

For a binary classification, there are two hypotheses that need to be compared: H_1, the hypothesis that the classification should be 1, and H_0, the hypothesis that the classification should be 0. The more likely classification can be determined by:

$$C = \begin{cases} 1 & \text{when } \Pr(H_1|E) > \Pr(H_0|E) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Because we are only comparing probabilities, there is simplification available.

$$\begin{aligned}
\Pr(H_1|E) &> \Pr(H_0|E) && (6)\\
\frac{\Pr(E|H_1)\Pr(H_1)}{\Pr E} &> \frac{\Pr(E|H_0)\Pr(H_0)}{\Pr E} && (7)\\
\Pr(E|H_1)\Pr(H_1) &> \Pr(E|H_0)\Pr(H_0) && (8)\\
& && (9)
\end{aligned}$$

Where do these probabilities come from? We simply sample the training set T, considering the class $c(t)$ and features $f_j(t)$ of each training instance t.

$$\begin{aligned}
\Pr(H_i) &= \frac{|\{t \in T \mid c(t) = i\}|}{|T|} && (10)\\
\Pr(F_j = k|H_i) &= \frac{|\{t \in T \mid c(t) = i \wedge f_j(t) = k\}|}{|\{t \in T \mid c(t) = i\}|} && (11)
\end{aligned}$$

Classification of a new instance c then becomes a matter of applying Naïve Bayesian reasoning.

$$L_i = \Pr(H_i) \cdot \prod_j \Pr(F_j = f_j(c) \mid H_i) \tag{12}$$

$$C(c) = \begin{cases} 1 & \text{when } L_1 > L_0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

It is probably worth worrying about the fact that the product in equation 12 risks underflow. We can fix this by taking logs, without harm to the inequality in equation 13.

$$L_i = \log \Pr(H_i) + \sum_j \log \Pr(F_j = f_j(c) \mid H_i) \tag{14}$$

$$\tag{15}$$

Because the log of 0 is undefined, it would probably be a good idea not to go there, so we use m-estimation by adding an arbitrary 0.5 to the numerator and denominator counts of each probability.

Here's pseudocode for a learner. In this pseudocode, `F[i,j]` is a two-dimensional array of counts, where the left dimension is an instance classification (0 or 1) and the right dimension is a feature number. The contents of each array entry is a count of the number of times that the given feature appears positive in the given class. The array `N[i]` is indexed by instance classification and gives the count of training instances with that classification. Each training instance t has a class $t.c$ and an array of features $t.f$.

```
to learn on a training set T:
    initialize every F[i,j] to 0
    initialize every N[i] to 0
    for each instance t in T
        increment N[t.c]
        for every feature j
            if (t.f[j] = 1)
                increment F[t.c,j]
    return N and F
```

The classifier follows a similar structure. Like the training instances, a classification instance has an array of features $c.f$.

```
to compute a likelihood L[i] given c, F and N:
    L[i] ← log (N[i] + 0.5) − log (N[0] + N[1] + 0.5)
    for j in 1 .. |c.f|
        s ← F[i,j]
        if c.f[j] = 0
            s ← N[i] − s
        L[i] ← L[i] + log (s + 0.5) − log (N[i] + 0.5)
    return L[i]

to classify an instance c:
    if L[1](c) > L[0](c)
        return 1
    return 0
```

Half the invocations of log can be removed from the likelihood calculation via the obvious loop hoisting if efficiency is a concern.

The instances you will be working with are based on features obtained from pixel counts of specific regions in Single Proton Emission Tomography (SPECT) heart images of patients, some of whom have been diagnosed with

varying heart disorders. The SPECTF dataset (`http://archive.ics.uci.edu/ml/datasets/SPECT+Heart`) contains scalar data for both the features and classification. The SPECT dataset (`http://archive.ics.uci.edu/ml/datasets/SPECT+Heart`) contains a binarization of the features of that data performed by its authors using an inductively learned ruleset (I think: it's hard to tell from the paper what they did). Both datasets have also had their classifications binarized as either normal (1) or abnormal (0). The dataset has been pre-divided by the authors into training and test sets such that there are an equal number (40) of normal and abnormal instances in the training set. The data is presented as CSV files, with each row representing a single instance (patient visit) and with the first comma-separated field representing the class.

For this exercise, I prepared the dataset for you in several different ways:

orig: I cleaned up the SPECT dataset as originally given.

itg: I re-quantized the SPECTF dataset using Information-Theoretic Gain to quantize feature values in an "optimal" way.

resplit: I combined the SPECT training and test sets, then resplit them randomly into new training and test sets in a 2:1 ratio preserving the proportion of normal and abnormal diagnoses in the two sets. This resplit is quite fragile; different resplits give markedly different answers, highlighting the need for cross-validation in this study.

Write a Naïve Bayesian learner on binary instance data. Run it on the provided datasets. For each dataset, output information like this:

```
orig 142/187(0.76) 10/15(0.67) 132/172(0.77)
```

The first number is the fraction of instances that are classified correctly ("accuracy"). The second number is the fraction of class-0 (abnormal) instances that are classified correctly ("true negative rate"). The third number is the fraction of class-1 (normal) instances that are classified correctly ("true positive rate").

Which is more important in this application: accuracy on abnormal instances or accuracy on normal instances? Which dataset seems to give the best results?

The data for this assignment is in a zipball at `http://svcs.cs.pdx.edu/moodle/file.php/5/hw4/hw4.zip`. Please submit the usual material—code and brief writeup—in the usual fashion.

Good luck and have fun!