Appendix

The Model-Averaged Naive Bayes (MANB) Algorithm

This appendix contains a set of equations that describe the MANB algorithm. From this description, it is straightforward to code the algorithm. We proceed in a top-down fashion, describing first the main inference task and then successively decomposing it into its parts.

Let $\boldsymbol{X}$ denote a set of n discrete-valued predictor variables, namely $\{X_1, X_2, \ldots, X_n\}$. Let $\boldsymbol{x}$ denote an instantiation (setting) of the variables in $\boldsymbol{X}$ in some test case. Suppose variable X_i has r_i possible values that are coded by the integers from 1 to r_i. We say that r_i is the dimensionality of X_i. For example, X_i could represent a SNP that has three genotype values and one value denoting a missing measurement, and those values could be encoded as 1, 2, 3, and 4. Let x_i be the value that is assigned to variable X_i in a given patient case. According to our encoding of values, x_i could be any number from 1 to r_i. We will sometimes use x_i as shorthand for $X = x_i$. Let T denote the discrete-valued target variable to be predicted. Let t denote an arbitrary value of T. Let r_T denote the dimensionality of T. For example, in the disease dataset that we studied, T denotes late onset Alzheimer's disease (LOAD), and it has the values *absent* (= 1) and *present* (= 2).

From Bayes theorem, we obtain the following equation:

$$P_a(t \mid \boldsymbol{x}) = \frac{P_a(\boldsymbol{x} \mid t) P(t)}{\sum_{t'=1}^{r_T} P_a(\boldsymbol{x} \mid t') P(t')} \tag{1}$$

The subscript "a" in Equation 1 denotes a model-averaged probability. We assume that the predictors are conditionally independent of each other, given the value of the target T, and thus we obtain the following:

$$P_a(\boldsymbol{x} \mid t) = \prod_{i=1}^{n} P_a(x_i \mid t). \tag{2}$$

We estimate each of the terms in Equation 2 using training dataset D and prior probabilities that are described below. Assume that D has N cases (samples). In the LOAD dataset that we studied, D consists of SNP values and a LOAD diagnosis for each of 1411 patient cases. In reference (1) it is proved that model averaging over all 2^n naive Bayes models is equivalent to using the following value for each term in Equation 2:

$$P_a(x_i \mid t) = P(T \rightarrow X_i \mid D) \cdot P(x_i \mid t, D) + P(T \ldots X_i \mid D) \cdot P(x_i \mid D), \tag{3}$$

where $T \rightarrow X_i$ designates that T and X_i are probabilistically dependent and $T \ldots X_i$ designates that they are independent. When they are dependent, we use the conditional probability $P(x_i \mid t, D)$ to estimate $P_a(x_i \mid t)$. When they are independent, we use $P(x_i \mid D)$. Equation 3 can be viewed as using model averaging (regarding whether a relationship between T and X_i is present or not) to provide smoothing of the probability $P_a(x_i \mid t)$ that is being estimated by Equation 3. This smoothing is in addition to the smoothing that we will do in estimating $P(x_i \mid t, D)$ and $P(x_i \mid D)$ (see below), which also appear in Equation 3.

Once we have derived $P_a(x_i \mid t)$ for each value x_i (of variable X_i) and each value t (of target T) we can use those probabilities in Equations 1 and 2 to calculate the posterior probability over T for any instantiation $\mathbf{x}$ of the predictor variables. For each predictor variable X_j that has no assigned value in a given patient case, we simply do not include the term $P_a(x_j \mid t)$ in Equation 2.[1]

We now describe how each of the terms in Equation 3 is derived. Let N_{ijk} denote the number of times in database D that variable X_i has the value k when target T has the value j. We pre-compute and store these N_{ijk} counts for use below. To keep the notation simple, we will assume that x_i equals the value k and t equals the value j. Let $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Let $N_i = \sum_{j=1}^{r_T} N_{ij}$. Note that for all i, $N_i = N$, where N is the total number of cases in training dataset D. Finally, let $N_{i*k} = \sum_{j=1}^{r_T} N_{ijk}$. We estimate the distribution $P(x_i \mid t, D)$ by assuming that every possible such distribution is equally likely *a priori*, and then integrating over all of them to obtain the following expectation (2):

$$P(x_i \mid t, D) = \frac{N_{ijk} + 1}{N_{ij} + r_i}.$$

Similarly, we estimate $P(x_i \mid D)$ as follows:

$$P(x_i \mid D) = \frac{N_{i*k} + 1}{N_i + r_i}.$$

We will now derive $P(T \rightarrow X \mid D)$ and $P(T \ldots X \mid D)$ in Equation 3.

$$P(T \rightarrow X_i \mid D) = \frac{P(D_i \mid T \rightarrow X_i) \cdot P(T \rightarrow X_i)}{P(D_i \mid T \rightarrow X_i) \cdot P(T \rightarrow X_i) + P(D_i \mid T \ldots X_i) \cdot P(T \ldots X_i)}, \tag{4}$$

where D_i denotes the data on just T and X_i in D. Assuming for now that we can compute the right side of Equation 4, we apply it and the following equations to derive $P(T \ldots X_i \mid D)$ and $P(T \rightarrow X_i \mid D)$, which are needed in Equation 3:

$$P(T \ldots X_i \mid D) = \frac{P(D_i \mid T \ldots X_i) \cdot P(T \ldots X_i)}{P(D_i \mid T \rightarrow X_i) \cdot P(T \rightarrow X_i) + P(D_i \mid T \ldots X_i) \cdot P(T \ldots X_i)} \tag{5}$$

[1] An alternative approach is to include a special value labeled "MISSING" for each predictor variable. If a predictor has a missing value in a given case to be predicted, its value is instantiated to be the value MISSING. This approach allows a missing value to be informative. We used this approach in the algorithm described in the main paper.

We now discuss calculating the terms on the right side of Equation 4. The term $P(T \rightarrow X_i)$ is our prior probability that T and X_i are probabilistically dependent, and $P(T \ldots X_i) = 1 - P(T \rightarrow X_i)$. For example, for the LOAD dataset that we studied, we used $P(T \rightarrow X_i) = 20 / 312{,}318$.

We derive $P(D_i \mid T \rightarrow X_i)$ in Equation 4 as follows, based on assumptions described in (1-3):

$$P(D_i | T \rightarrow X_i) = \prod_{j=1}^{r_T} \left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}! \right) \tag{6}$$

Note that the values of the factorial function can be pre-computed and stored in an array, and thus, its use above will correspond to a simple array access.

In a manner similar to Equation 6, we derive $P(D_i \mid T \ldots X_i)$ in Equation 4 as follows:

$$P(D_i | T ... X_i) = \frac{(r_i - 1)!}{(N_i + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N_{i*k}! \tag{7}$$

To provide an indication of which variables most strongly predict target T, we can sort the predictor variables according to $P(T \rightarrow X_i \mid D)$, as given by Equation 4, and output the top c predictors, along with their probabilities, where c is a user-specified value.

A Logarithmic Version of MANB

The terms in the above equations can readily become so small that they cause problems in maintaining adequate numerical precision. Thus, it is better to calculate them in logarithmic form. This section parallels the above section, while presenting the equations in a logarithmic form. We will use natural logarithms, denoted by the function ln. We will use exp(x) to denote e^x.

$$\ln(P_a(t | \mathbf{x})) = \ln(P_a(\mathbf{x} | t)) + \ln(P(t)) - \text{lnDenom}, \tag{1'}$$

where lnDenom is a function that is specified by the following pseudocode:

```
$s := -\infty$;
for $t' := 1$ to $r_T$
    $s := \text{lnAdd}(s, \ln(P_a(\mathbf{x} | t')) + \ln(P(t')))$;
return $s$.
```

The function lnAdd(x, y), which appears above and is defined below, takes two arguments x and y that are in logarithmic form and returns $\ln(e^x + e^y)$. However, it does so in a way that preserves a good deal of numerical precision that could be lost if $\ln(e^x + e^y)$ were calculated in a direct manner. The value $-\infty$ in the above pseudocode can be implemented in practice by using the largest negative number that can be represented by the computer on which the code is running, as for example -1 × 10^{+4931}.

Once we have computed ln(P_a(t | $\boldsymbol{x}$) for each value of t, as shown in Equation 1', we simply exponentiate each term to obtain the posterior probabilities of interest: P_a(t | $\boldsymbol{x}$) = exp(ln(P_a(t | $\boldsymbol{x}$)).

The remainder of this section shows how to derive the terms on the right side of Equation 1' in a manner that is parallel to the previous section.

$$\ln(P_a(\boldsymbol{x}|t)) = \sum_{i=1}^{n} \ln(P_a(x_i|t)). \tag{2'}$$

$$\ln(P_a(x_i|t)) = \text{lnAdd}(\ln(P(T \to X_i|D)) + \ln(P(x_i|t,D)),\ \ln(P(T ... X_i|D)) + \ln(P(x_i|D))). \tag{3'}$$

$$\begin{aligned}\ln(P(T \to X_i|D)) = \ln(P(D_i|T \to X_i)) + \ln(P(T \to X_i)) - \\ \text{lnAdd}(\ln(P(D_i|T \to X_i)) + \ln(P(T \to X_i)),\ \ln(P(D_i|T ... X_i)) + \ln(P(T ... X_i))).\end{aligned} \tag{4'}$$

$$\begin{aligned}\ln(P(T ... X_i|D)) = \ln(P(D_i|T ... X_i)) + \ln(P(T ... X_i)) - \\ \text{lnAdd}(\ln(P(D_i|T \to X_i)) + \ln(P(T \to X_i)),\ \ln(P(D_i|T ... X_i)) + \ln(P(T ... X_i))).\end{aligned} \tag{5'}$$

$$\ln(P(D_i|T \to X_i)) = \sum_{j=1}^{r_T} \left(\ln((r_i - 1)!) - \ln((N_{ij} + r_i - 1)!) + \sum_{k=1}^{r_i} \ln(N_{ijk}!) \right) \tag{6'}$$

$$\ln(P(D_i|T ... X_i)) = \ln((r_i - 1)!) - \ln((N_i + r_i - 1)!) + \sum_{k=1}^{r_i} \ln(N_{i*k}!). \tag{7'}$$

Note that to improve efficiency, the logarithms of the factorials can be pre-computed and stored. Let *lnfact*(w) be a function that returns ln(w!). In the pseudocode that follows, we assume that these function values are stored in an array that is also called *lnfact*, which we distinguish by using square brackets. We can efficiently construct this array using the following iterative method:

```
lnfact[0] := 0;
$m := N + \max_{i=1}^{n}(r_i)$;  // $N$ is the number of cases in the training dataset $D$
for $w$ := 1 to $m$
    lnfact[$w$] := lnfact[$w$-1] + ln[$w$];
```

The lnAdd function was used above in several equations, and it is defined as follows:

```
function lnAdd(x, y)
    if $y > x$ then
        $temp := x$;
        $x := y$;
        $y := temp$;
    return $\ln(1 + \exp(y - x)) + x$.   // Note that exp(y – x) computes $e^{(y-x)}$.
```

A Java implementation of the MANB algorithm is available at
http://www.dbmi.pitt.edu/cooperlab/overview by following the link there to Software.

References

1. Dash D, Cooper G. Exact model averaging with naive Bayesian classifiers. In: Sammut C, Hoffmann AG, editors. Proceedings of the 19th International Conference on Machine Learning; 2002 July 8 -12; Sydney, New South Wales, Australia. Morgan Kaufmann; 2002. p. 91-8.

2. Cooper G, Herskovits E. A Bayesian method for the induction of probabilistic networks from data Machine Learning. 1992;9(4):309-47.

3. Heckerman D, Geiger D, Chickering D. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning. 1995;20:197-243.