Diabetes Prediction with Incomplete Patient Data

Hao Yi Ong, Dennis Wang and Xiaosong Mu

1 Introduction

Over 25 million people, or nearly 8.3% of the entire United States population, have diabetes. Diabetes is also associated with a wide range of complications from heart disease and stroke to blindness and kidney disease. Combined with electronic medical record systems, an implementation of a smart predictor could prompt high-risk patients to obtain diabetes testing in cases when the physician had not thought to recommend it. Based on the Kaggle Practice Fusion Diabetes Classification challenge, we aim to build a model to determine who has a diabetes diagnosis given a patients set of electronic health records.

Unlike the original competition, which assumes that the algorithm will have access to the full medical record of patients and that patients all have a standard database (e.g., exact same tests taken, same recorded variables), we are interested in creating a model that assumes we only know part of the medical record as the input. For instance, if a patient has not undergone the full battery of tests as all the patients in the original training dataset had, or if we are missing information from a patient's medical record, we want to still be able to classify and output whether the patient has diabetes based on the limited amount of information. In our project, our algorithm will be scored by the false positive and false negative rates, which when combined gives us the total error rate in diagnosing diabetes for test datasets.

2 Oracle

The ideal oracle for evaluating our algorithm would be experienced physicians who have diagnosed patients with diabetes. This is simply because when it comes to disease diagnosis there isn't a definitive algorithm or toolbox that can assess the exact same sets of data and output results out there in the market or in research. In this ideal scenario, we would present a standard set of basic medical information (e.g., height, weight, blood sugar levels) and, additionally, a randomized sample of patient records to a group of physicians. Physicians would then predict whether each patient has diabetes, and their accuracy of diagnosis would be used as a metric.

For the purposes of this project, however, this is not financially feasible and we do not have the time to employ multiple physicians to consider a large amount of patient health record data. Instead, we use as surrogate measures the test accuracies of established diabetes tests. Specifically, we consider HBA1c, fasting plasma glucose (FPG), and oral glucose tolerance tests (OGTT), which are considered by the U.S. Department of Health and Human Services to be the standard. And according to a study conducted by the World Health Organization in 2011, the true positive rates and true negative rates for these tests are about $> 85\%$ and $> 95\%$, respectively.

3 Bayesian network structure learning

We have fully implemented a local search algorithm in Julia to find a Bayesian network (BN) structure that best fit the training data. The BN-learning problem is equivalent to searching for an optimal structure over the space of all possible directed acyclic graphs (DAGs). In general, this search problem is NP-hard. A brute-force graph enumeration method that compares amongst all possible DAGs to find a globally optimal structure for a given dataset can be computationally prohibitive. Specifically, McKay et al. [M$^+$04] showed that the cardinality a_n of the set of all DAGs grows superexponentially with the number of nodes n (i.e., number of feature variables). Specifically, the number of BNs on n nodes, for $n = 1, 2, 3, \ldots$ is $1, 3, 25, 543, 29281, 3781503, \ldots$, which caps the maximum number of features we can feasibly run a brute force algorithm for at around $n = 5$ for a modern processor.

3.1 Tabu search

The algorithm presented in this report uses local search with a tabu list heuristic, which is a form of greedy hill-climbing/gradient ascent algorithm that maximizes the fitness of structures based on some scoring function. By maintaining a tabu list of recent operators we applied (e.g., adding an edge to the existing BN) and not considering operators that reverse the effect of recently applied operators, the heuristic avoids getting stuck at local optima. Many heuristic search techniques have been proposed for this problem, but only a small number outperform local search with tabu lists [KF09]. In light of this empirical observation, this project proposes a modified tabu search for its simplicity and performance for large BN structures. To understand the modification, observe that the number of possible edge operators on any BN scales quadratically with the number of nodes. To accommodate structures with large numbers of edge operators (namely, DAG edge addition, removal, or reversal), the modified tabu search only searches over a randomly chosen subset of all possible operators. Algorithm 1 describes the search heuristic, where $\mathcal{T}$ is the tabu list, $\mathcal{O}$ is a set of all possible operators on the BN, L is the size of the tabu list, N is a stopping criterion, and M is an upper limit on the number of nodes to limit the operators search space for large problems.

3.2 Bayesian scoring function

To evaluate the BN, we use the log Bayesian score, which is a popular BN evaluation metric because of its ability to optimally balance the complexity of the BN structure with the available data [KF09]. While a full explanation of the Bayesian scoring will be too lengthy for the purposes of this report, we give a basic description here. In essence, we assume a Dirichlet distribution over each of the parameters, which are, in our case, the conditional probabilities corresponding to each edge in the BN. Given this, we compute $\mathbf{Prob}(G \mid D)$, where G is the BN to be evaluated and D is the given training dataset. The actual function can be derived using a combination of Bayes' rule and the law of total probability. The best BN simply corresponds to the maximum value of $\mathbf{Prob}(G \mid D)$.

4 Bayesian network parameter learning

To learn the conditional probabilities corresponding to the BNs edges (i.e., the BN parameters), we use a maximum a posteriori (MAP) estimate. Assuming a Dirichlet prior over the parameters θ and given a training dataset D, the posterior distribution is given as $P(\theta \mid D)$. Since we have our BN structure defined at this point, we will compute $P(\theta \mid D)$ by assigning each node its value according to θ, and then multiplying the conditional probabilities of observing each child node given its parent nodes. To maximize this evaluation function, we can take the log of the distribution and apply standard convex optimization techniques (e.g., interior point methods), which are implemented in many existing solvers, such as Gurobi and MOSEK. At this point, we have figured out the algorithm for parameter learning, and will be implementing this over the next few days.

Algorithm 1: TabuSearch

```
input  : $\mathcal{N}, \mathcal{D}, L, N, M$
output: $b^\star$
$b^\star \leftarrow$ initialize random Bayes network using $\mathcal{N}$
$b \leftarrow b^\star$
$\mathcal{T} \leftarrow$ initialize tabu list
$\mathcal{O} \leftarrow$ generate the set of all possible operators using $\mathcal{N}$
$t \leftarrow 1$
$LastImprovement \leftarrow 0$
while $LastImprovement < N$ do
    $o^{(t)} \leftarrow \epsilon$        // Set current operator to be uninitialized
    if card$\mathcal{N} > M$ then
        $\mathcal{O}' \leftarrow$ generate random subset of $\mathcal{O}$
    else
        $\mathcal{O}' \leftarrow \mathcal{O}$
    // Search for best allowed operator in $\mathcal{O}'$
    for each operator $o \in \mathcal{O}'$ do
        if does not exist $o' \in \mathcal{T}$ such that $o$ reverses $o'$ then
            $b_o \leftarrow$ apply $o$ on $b$
            if $b_o$ is a valid Bayes net then
                if $o^{(t)} = \epsilon$ or score$(b_o, \mathcal{D}) >$ score$(b_{o^{(t)}}, \mathcal{D})$ then
                    $o^{(t)} \leftarrow o$
    add $o^{(t)}$ to $\mathcal{T}$
    remove $o^{(t-L)}$ from $\mathcal{T}$
    $b \leftarrow b_{o^{(t)}}$
    if score$(b, \mathcal{D}) >$ score$(b^\star, \mathcal{D})$ then
        $b^\star \leftarrow b_o$
        $LastImprovement \leftarrow 0$
    else
        $LastImprovement \leftarrow LastImprovement + 1$
    $t \leftarrow t + 1$
```

5 Bayesian network inference

Once we have our optimal bayesian network, we will use statistical inference to predict the presence of diabetes in each patient. The three algorithms we will consider using are exact inference or approximate inference techniques, such as Gibbs sampling, and particle filtering. Since we would have calculated the optimal transition probabilities during the learning phase, given just the leaf node values of any graph, we will be able to compute the conditional probability using exact inference. However, given that we may have up to 20 nodes (i.e., up to 190 edges each with many possible values), this may be computationally expensive, which is why we will consider implementing particle filtering or Gibbs sampling. Our implementation of the two approximate inference algorithms were described fully the lecture notes, and one of them have been implemented for homework. We will not duplicate them here.

6 Description of how the model and algorithm will work

Obtaining the variables, states, and nodes requires a significant amount of data processing. Given that the data is available to us as a SQLite file, we will use Python's SQLite3 library to extract the standardized set of fields (corresponding to features) from each patient record file, holding out 10% for testing. After extracting the data, we will define acceptable values for each field (more on this later), and eliminate the patients with malformed data and significant outliers. We will then discretize each value. Finally, we will output the data to a CSV-file for consumption by our BN structure and parameter learning algorithms described in previous sections. For testing, we will use the held out records, including those that were missing data in the training step, to predict the probability of Type II diabetes. Since we are using conditional probabilities from the BN, we do not require all nodes to have values—we simply sum over all unknown variable values, possibly using approximate inference techniques such as particle filtering.

Our features/nodes come in several types: continuous numeric, discrete numeric, and qualitative. For discrete numeric data (i.e. age, U.S. state...), we take it either as is, or in the case of states, we use an enumeration. For continuous numeric data (i.e. bmi, weight, height, temperature...), we discretize them into buckets. For example, weight goes in buckets of 10 lbs, height in increment of 2 inches, etc. For qualitative data, it is much harder to discretize. Currently, our goal is to search over all possible values and create an enumeration (i.e. for medication types, test results, codes...). Also, a lot of medical records have a many-to-one relationship with the patient. In those cases, we simplify by either taking an average or the most recent depending on what makes the most sense for the particular feature. As of now, feature parsing/discretizing and selection are still undergoing.

References

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

[M$^+$04] B. McKay et al. Acyclic digraphs and eigenvalues of (0,1)-matrices. *Journal of Integer Sequences*, 7, 2004.