

Diabetes Prediction with Incomplete Patient Data

Hao Yi Ong, Dennis Wang and Xiao Song Mu

December 12, 2014

1 Introduction

Over 25 million people, or nearly 8.3% of the entire United States population, have diabetes. Diabetes is associated with a wide range of complications from heart disease and strokes 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 whether a patient is at risk for Type II diabetes given his/her 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 reduced 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 data sets.

1.1 Overview of approach

Our approach to the problem was to model the relationships in the data as a Bayesian network. The inspiration for this was that probabilistic inference on a Bayesian network does not require all variables. Thus, it would allow us to predict diabetes in patients even if certain features were missing. To implement this, we first learned the structure of the Bayesian network using a Bayesian score. Then, given this structure, we learned the parameters, or conditional probabilities, associated with the edges. Finally, we tested the Bayesian network by calculating probability of diabetes given the features in patients' medical records and compared to actual diagnoses of diabetes.

1.2 Contributions

Our contributions are twofold. First, we formulate our problem as a classification problem using Bayesian network inference. Second, we present our techniques for learning the Bayesian network structure and parameters and for inferring the probability of diabetes. The rest of this paper is structured as follows. Section 2 describes our oracle and baseline algorithm against which we evaluate our methods. Sections 3, 4, and 5 describe our methods for learning the Bayesian network structure and parameters and for approximately inferring the probability of diabetes infliction. We present our numerical experiments on our data set and discuss our results in Section 6. Some concluding remarks are drawn and extensions proposed in Section 7.

2 Evaluation Criteria

2.1 Oracle

The ideal oracle for evaluating our algorithm are diagnoses from experienced physicians. The general consensus among the medical community is that while machines and algorithms can aid human diagnosis, disease is a sufficiently complex phenomenon that in general, human judgment surpasses the best that machines have to offer. Moreover, we require our oracle to accept as input a very specific set of data input.

Had we been able to obtain physician diagnosis as oracle, we would have presented a randomized sample of patient records containing a standard set of basic medical information (e.g., height, weight, glucose levels, etc) to a group of physicians. Physicians would have then predicted whether each patient had diabetes, and the overall accuracy of the physician diagnoses would have been used as our oracle metric.

For the purposes of this project, however, this was not financially feasible as we did not have the time to employ multiple physicians to consider large amounts of patient health record data. Instead, we used, as surrogate, 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 either of these tests are about $> 85\%$ and $> 95\%$, respectively.

2.2 Baseline algorithm

Given our limited selection of techniques taught in class, the problem of classifying whether a patient is inflicted with a disease with unknown variables can only be done via Bayesian network inference or factor graphs. For our baseline, we choose to compare our results with a predictor trained with a classification algorithm and no unknown variables. Thus, instead of attempting to outdo our baseline, we use it as a target objective that we want to attain (in terms of accuracy).

For our baseline test, we chose a feature vector with very basic quantitative data such as height, weight, body mass index, etc. to predict Type 2 diabetes. The algorithm used was standard logistic regression from Python's scikit-learn, and we tested using leave-10%-out cross validation on the training data (6000 patients). We ended up with a false positive rate of 0.7% and false negative rate of 15.3% for a total error rate of 16%. Note that these values are actually pretty close to the oracle in our case.

3 Bayesian Network Structure Learning

We have fully implemented a local search algorithm in Julia to find a Bayesian network (BN) structure that best fits 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 super-exponentially with the number of nodes n (i.e., number of feature variables). McKay demonstrated that the number of BNs on n nodes, for $n = 1, 2, 3, \dots$ is

$$1, 3, 25, 543, 29281, 3781503, \dots,$$

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 reduces the likelihood of being 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 [TK12, 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.

Algorithm 1: Tabu search

input : $\mathcal{N}, \mathcal{D}, L, N, M$
output: b^*
 $b^* \leftarrow$ initialize random Bayes network using \mathcal{N}
 $b \leftarrow b^*$
 $\mathcal{T} \leftarrow$ 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 $score(b, \mathcal{D}) > score(b^*, \mathcal{D})$ **then**
 $b^* \leftarrow b_o$
 $LastImprovement \leftarrow 0$
 else
 $LastImprovement \leftarrow LastImprovement + 1$
 $t \leftarrow t + 1$

3.2 Bayesian network 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. In our case, these parameters are the conditional probabilities corresponding to each node and every possible instantiation of its parent nodes in the BN, constrained such that G is a DAG. We chose our prior to be a uniform Dirichlet distribution over all possible structures $\mathbf{Prob}(G)$. 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 likelihood estimate MLE and apply Laplace smoothing with $\lambda = 1$. This is necessary given that the size of our data set is small relative to the space of all possible observations—without Laplace smoothing, we found that many conditional probabilities calculated based on observations is 0. In addition, the data is stored in a relational database and data retrieval is optimized by applying indices over sets of variable names. To maximize the MLE evaluation function, we simply count the number of observations corresponding to the conditional probabilities, giving our MLE estimate of the parameters.

5 Bayesian Network Inference

After obtaining our optimal Bayesian network, we use statistical inference to predict the presence of diabetes in each patient. To classify and thus predict diabetes, we assign a yes/no label to whichever inferred probability is higher, where the probabilities are over the missing variables and observed ones.

Since we have the optimal transition probabilities from the learning phase, using exact inference was a possibility. However, we decided to use Gibbs sampling to approximate our inference due to the computational complexity of using exact inference. In general, exact inference is an NP-hard problem with exponential time complexity in the worst case [Koc14]. Since we have over 20 nodes and up to 190 edges each with many possible discrete values, we use Gibbs sampling for approximate inference.

Our implementation of Gibbs sampling is described fully in the lecture notes, which is omitted here for brevity. However, to note is that in our Gibbs sampling, because of the dependence of each sample, we discard the first 100 samples (burn-in period) and keep only every 10th sample (sample thinning) to better approximate independent sampling and yield better inference results.

6 Numerical Simulations

6.1 Data model and feature selection

Obtaining the variables, states, and nodes required a significant amount of data processing. Given that the data was available to us as a SQLite file, we used Python’s SQLite3 library to extract the standardized set of fields (corresponding to features) from each patient record file, holding out 10% (900) for testing. After extracting the data, we defined acceptable values for each field, and eliminated the patients with malformed data. We then discretized each value. Finally, we outputted the data to a CSV-file for consumption by our BN structure and parameter learning algorithms described in previous sections.

Our features/nodes came in several types: continuous numeric, discrete numeric, and qualitative. For discrete numeric data (i.e. age, U.S. state...), we took it either as is, or in the case of states, we used an enumeration. For continuous numeric data (i.e. bmi, weight, height...), we discretized them into buckets. For example, weight goes in buckets of 10 lbs, height in increment of 2 inches, etc. For qualitative data, it was much harder to discretize. In our case, we used the ICD-9 codes associated with the medical diagnoses as features. The ICD-9 codes broke the set of possible diagnoses into categories such as circulatory (codes 390-459) or respiratory (codes 459-519) in nature. For each of these diagnosis categories, we created a binary feature that represented whether the patient had the diagnosis within the last four years. In addition, a lot of medical records had a many-to-one relationship with the patient. In those cases, we simplified by either taking an average or the most recent depending on what made the most sense for the particular feature.

6.2 Testing strategy

For testing, we used the held out records, including those that were missing data in the training step, to predict the probability of diabetes. We thresholded at 50%, so if the probability was more than 50%, we considered it to be a positive prediction of diabetes. This thresholding is in line with classification algorithms such as the logistic regression method used for our baseline, where classification of feature vectors are done using a single threshold value. It is important to note that, in order to simulate the reality of missing features, we held out a predefined number of features chosen randomly for each patient before using the Bayes net for inference. Also, in order to select the best Bayes net structure, we held out 10% of the training data as the validation set to tune our tabu search algorithms hyperparameters and tested the top 8 Bayes net structures based on Bayes score to see which performed best. The results are presented and analyzed in the following sections.

6.3 Results and analysis

To find the best Bayesian network for predicting diabetes, we evaluated the top 8 Bayesian networks learned using classification accuracies as detailed previously. Figure 1 shows the error rates for each of our top 8 structures. From the results of the tests, it is clear to see that

the top 8 structures according to Bayes score performed very similarly, with graph 3 being slightly better. Thus, we used graph 3s structure for the remainder of our tests. Figure 2 presents a visualization of our BN structure.

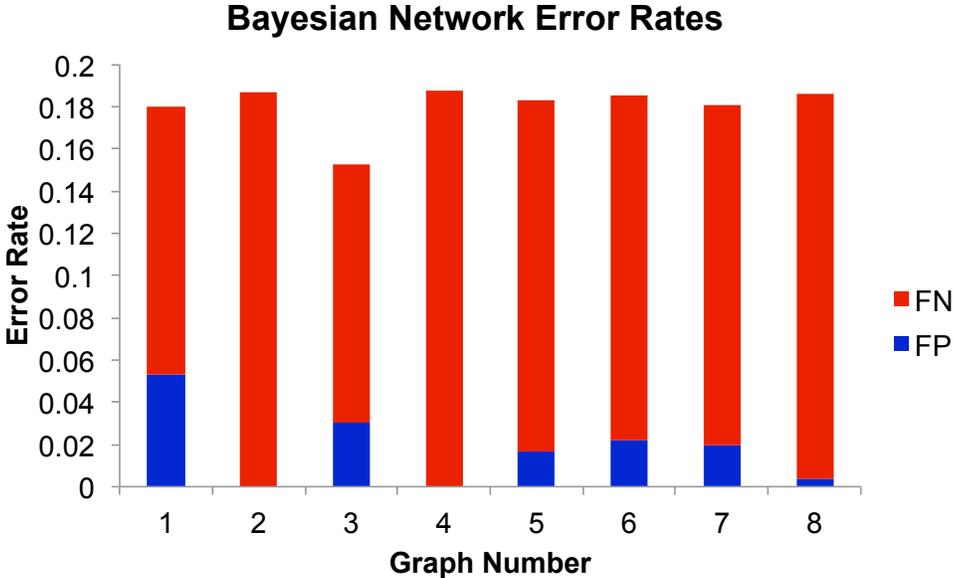


Figure 1: Error rates for the test set.

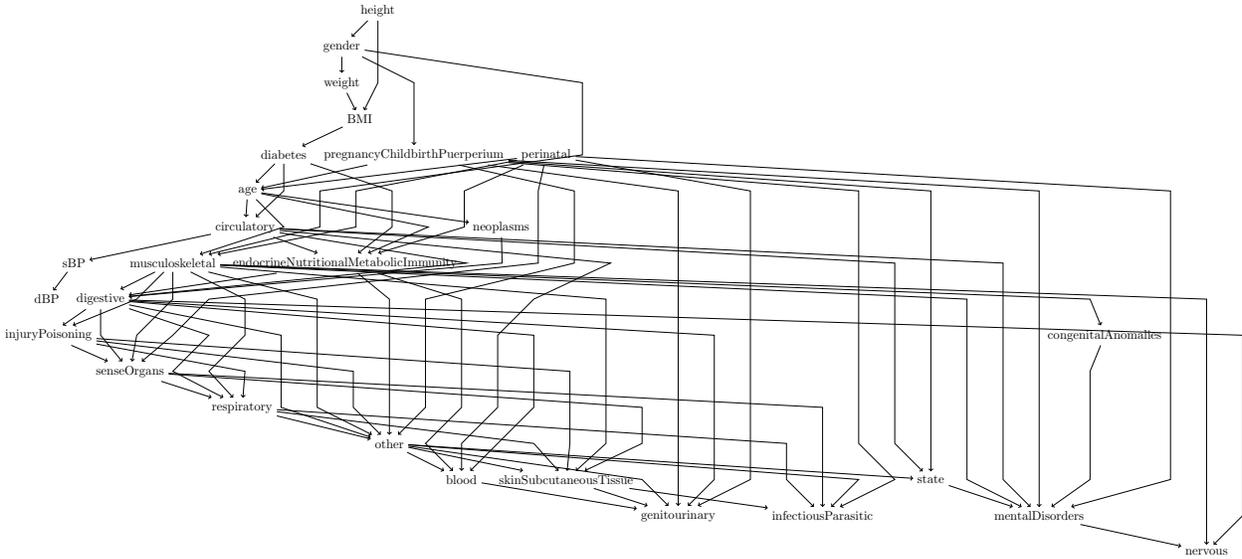


Figure 2: A depiction of our best learned BN structure.

From the data in Table 1, we can see that error rates for diabetes prediction did not decrease as we withheld more features. They stayed between 13% and 16%. In fact, there

seems to be a surprising inverse relationship between number of features withheld and error rate. The most likely cause of this is that we selected some “bad” features. However, it is difficult to pinpoint which features these are, since we randomize which features are withheld for each patient. Given more time, this would be something that we would to explore further in depth. Additionally, future work will include more sophisticated feature selection, as our focus for this project was on the algorithms and techniques.

Table 1: Classification results with varying number of missing features

#missing	Train error	Test error	False pos. (test)	False neg. (test)	Pred. time
3	0.0531	0.1662	0.0363	0.1299	3.75 s
5	0.06232	0.1511	0.0302	0.1208	6.44 s
8	0.0544	0.1339	0.0272	0.1067	9.67 s
10	0.0642	0.1299	0.0252	0.1047	11.86 s

Another promising observation is that the overall error rate is comparable to our oracle (<15% false positive, <5% false negative). In addition, running a simple logistic regression with no features withheld also gave around 16% total error rate. However, it is interesting to note that false negatives are relatively high and false positives relatively low for us, whereas it is the opposite for the oracle. This implications of this are both positive and negative. On the one hand, our low false positive rate indicates that few resources will be wasted in testing people without diabetes. However, on the other hand, the high false negative rate indicates that there will still be a good percentage of people in the system with diabetes that will not be found by our algorithm.

Finally, a comment should be made on the actual running time of the algorithm. As can be seen from the chart, as the number of features is withheld, the amount of time it takes to make a prediction increases. This is completely due to the inference step, as there are more features for the Gibbs sampling to iterate through. In total, the predictions on the test set of 900 patients took several hours, which can be prohibitively slow for an “online” application and for testing. However, since we intend for our model to be used “offline” on existing databases, this does not pose a big drawback. Assuming a worst-case scenario of 20 seconds per patient record on a single thread, our predictor would take 3 weeks for an electronic medical record database with 100,000 patients. With code optimization and parallelization (e.g., GPU), the total computation time can potentially be improved by a few orders of magnitude to even a few hours. Alternatively, our algorithm can simply run in the background, processing a patient every 20 seconds and raise flags in the system if there are patients at high risk of diabetes.

7 Conclusion and Future Work

In conclusion, we believe the results for our model are promising and that our model can be beneficial if used with existing database systems to flag patients with high risk of Type II

diabetes. In addition, our model is flexible and realistic in the sense that it is able to handle incomplete medical records.

For future work, as mentioned in our analysis section, we would like to incorporate more expert diabetes-specific medical knowledge into our feature selection, as well as expert feedback for our Bayesian network structure construction. In addition, we would like to consider other inference methods for runtime or accuracy improvements. Lastly, we understand that Bayesian networks are only one kind of model to address missing features. Given more time, we feel that it would be valuable to explore other models such as Markov nets or factor graphs.

Acknowledgments

We thank Professor Percy Liang and the course assistants for motivating and providing help to the project. We are also indebted to Billy Jun, our project mentor, for giving helpful suggestions and feedback through many rounds of discussions. We acknowledge Kaggle and Practice Fusion for providing the data set.

References

- [KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- [Koc14] M. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Lincoln Library, 2014.
- [M⁺04] B. McKay et al. Acyclic digraphs and eigenvalues of (0,1)-matrices. *Journal of Integer Sequences*, 7, 2004.
- [TK12] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. *arXiv preprint arXiv:1207.1429*, 2012.