

How Your Instructor Created a Bayes Network from the Diabetes Dataset

The dataset that I started with was the `diabetes.arff` dataset supplied with Weka. It looked like it had a good mixture of attributes caused by diabetes and attributes causing diabetes. The dataset is in `/home/bylander/weka/data/UCI/diabetes.arff` on the CS network. Here are comments from the preface to that dataset.

1. Title: Pima Indians Diabetes Database
2. Sources:
 - (a) Original owners: National Institute of Diabetes and Digestive and Kidney Diseases
 - (b) Donor of database: Vincent Sigillito (vgs@aplcn.apl.jhu.edu)
Research Center, RMI Group Leader
Applied Physics Laboratory
The Johns Hopkins University
Johns Hopkins Road
Laurel, MD 20707
(301) 953-6231
 - (c) Date received: 9 May 1990
3. Past Usage:
 1. Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press.

The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The population lives near Phoenix, Arizona, USA.

Results: Their ADAP algorithm makes a real-valued prediction between 0 and 1. This was transformed into a binary decision using a cutoff of 0.448. Using 576 training instances, the sensitivity and specificity of their algorithm was 76
4. Relevant Information:

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. ADAP is an adaptive learning routine that generates and executes digital analogs of perceptron-like devices. It is a unique algorithm; see the paper for details.
5. Number of Instances: 768

6. Number of Attributes: 8 plus class
7. For Each Attribute: (all numeric-valued)
 1. Number of times pregnant
 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
 3. Diastolic blood pressure (mm Hg)
 4. Triceps skin fold thickness (mm)
 5. 2-Hour serum insulin (μ U/ml)
 6. Body mass index (weight in kg/(height in m)²)
 7. Diabetes pedigree function
 8. Age (years)
 9. Class variable (0 or 1)

8. Missing Attribute Values: None

9. Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")

Class Value	Number of instances
0	500
1	268

10. Brief statistical analysis:

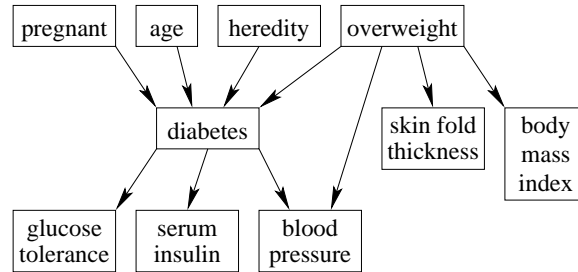
Attribute number:	Mean:	Standard Deviation:
1.	3.8	3.4
2.	120.9	32.0
3.	69.1	19.4
4.	20.5	16.0
5.	79.8	115.2
6.	32.0	7.9
7.	0.5	0.3
8.	33.2	11.8

Relabeled values in attribute 'class'

From: 0	To: tested_negative
From: 1	To: tested_positive

Network Structure

The first thing I focused on was creating the network structure. Based on my admittedly low knowledge of diabetes, I settled on the following network for the reasons given below:



- I am fairly certain that diabetes does not cause “number of times pregnant,” age, and “diabetes pedigree function” (heredity).
- An interesting part of the dataset is that it has two measures related to being overweight: “triceps skin fold thickness” and “body mass index.” These measurements don’t cause you to be overweight, rather being overweight causes these measurements to be high. Unfortunately, this makes “overweight” a hidden variable in the network. After further examination, skin fold thickness looked like very poor evidence for diabetes, so I used body mass index as the value of overweight.
- The “plasma glucose concentration” and the “serum insulin” measurements are both tests for diabetes, so I have diabetes causing these.
- Blood pressure was something I had not associated with diabetes. After 15 minutes of dedicated research on the internet, I discovered that there is some debate about whether diabetes is a cause of blood pressure, but that could just be the quality of information that I happened to stumble into. However, no one mentioned blood pressure causing diabetes, so I put in the causal link from diabetes to blood pressure. However, all sorts of things cause blood pressure, including pregnancy, age, and overweight, I think. I suppose there is a heredity component as well, but a different measure from the diabetes one. To make the network presentable and just to illustrate a point or two, I just added a link from overweight to blood pressure. Probably a couple others should be added.

Data Transformation

All of these attributes are numeric, and I wasn’t very interested in developing a numeric causal model. In order to present the standard conditional probability tables of belief networks, I needed discrete attributes. To make my life simpler, I decided to make each attribute binary, correspond to high values and low values. Probably the right thing to do is to fit a numerical probability distribution (e.g., multivariate normal) for each node.

I first decided to try Weka’s `weka.filters.Discretize` method to transform the attributes to binary variables:

```

DS=$WEKAHOME/data/UCI/diabetes.arff
java weka.filters.Discretize -B -i $DS > diabetes-binary.arff

```

This gave me a strange result, though. For most the attributes, one side of the attribute was a small percentage (around 10%) of the samples. This didn’t look very useful because

over 33% of the samples are positive. I think the problem is that this Weka filter uses information gain, and information gain often favors highly pure small splits. This is ok for one of many nodes in a decision tree, but not for a single binary value.

As an alternative, I decided to simply find the median value of each attribute and divide each attribute up 50/50 (or as close as I could). The division of the values into several bins is a very common method for discretization, though usually more than 2 is used.

Once I did the transformation, I looked at some the counts more closely. This is the point where I decided to discard skin fold thickness and merge overweight and body mass index. Skin fold thickness did not look relevant after this transformation.

Conditional Probability Tables

At this point, it is easy to fill in the conditional probability tables. However, with only 768 samples, one should expect some inaccuracy especially for the diabetes table, which has 32 entries. Here are all the tables:

pregnant = hi	419/768
pregnant = lo	349/768

age = hi	372/768
age = lo	396/768

heredity = hi	384/768
heredity = lo	384/768

overweight = hi	382/768
overweight = lo	386/768

	paho	pahō	paḥo	pāhō	pāho	pāhō	pāho	pāhō
diabetes	58/83	27/69	35/74	27/76	11/25	6/20	7/28	11/44
¬diabetes	25/83	42/69	39/74	49/76	14/25	14/20	21/28	33/44

	p̄aho	p̄ahō	p̄aḥo	p̄āhō	p̄aho	p̄ahō	p̄aho	p̄ahō
diabetes	15/23	6/16	11/20	5/11	27/74	5/74	15/55	2/76
¬diabetes	8/23	10/16	9/20	6/11	47/74	69/74	40/55	74/76

	diabetes	¬diabetes
glucose = hi	206/268	182/500
glucose = lo	62/268	318/500

	diabetes	¬diabetes
insulin = hi	128/268	256/500
insulin = lo	140/268	244/500

	diab,over	diab,¬over	¬diab,over	¬diab,¬over
pressure = hi	91/179	37/89	121/203	135/297
pressure = lo	88/179	52/89	82/203	162/297

Evaluation

I decided to evaluate my network using a leave-one-out evaluation, which is particularly efficient for naive Bayes and for belief networks with no hidden variables. It should be noted that some bias has been added because I have used the samples to make some of my decisions about how to transform the samples and how to structure the network. But this should give some idea whether a belief network might be useful for this domain.

After gathering all the probabilities from all the samples (I also tried Laplace's Law of Succession here), I implement leave-one-out by looping over the samples again. For each sample, first the sample is subtracted from the counts, then the network is used to classify the sample, and finally the sample is added back into the counts.

I compare my belief network to Weka's naive Bayes and decision tree implementations, also using leave-one-out evaluation. The code I ran was:

```
java weka.classifiers.NaiveBayesSimple -t $DS -x 768
java weka.classifiers.J48.j48 -t $DS -x 768
```

where `$DS` is the binary-valued diabetes dataset. The `-x 768` option results in a 768-fold cross-validation, which means each fold contains one example. It is important to evaluate different methods on the same dataset in the same way for a fair comparison.

Here are my results comparing my belief network to naive Bayes and decision trees:

Method	Accuracy
Belief Network (Laplace)	557/768 \approx 72.5%
Belief Network	555/768 \approx 72.3%
Decision Trees	553/768 \approx 72.0%
Naive Bayes	549/768 \approx 71.5%

At first glance, this looks good for belief networks, but a 95% confidence interval is about $\pm 3\%$, so the differences in accuracy are too small to declare a statistical winner. The best one can say is that belief networks are promising for this domain. A more careful transformation of the samples might (or might not) show the superiority of belief networks for this domain.

Contact me if you are interest in my implementation, but I will warn that it is Perl code that will not win any software engineering design contests.