PiPM 2013 – Final Report

Eric Andrews, 013594545

December 18, 2014

1 Introduction

The task given was two-folded. First, to learn which arcs are present in a golden standard Bayesian network according to sample data generated from it. Second, to predict the (normalized) probabilities given to a separate test data set by the golden standard network.

While these two tasks are inherently linked, i.e. good arc predictions imply good knowledge of the structure and thus the conditional independencies of the golden standard network, these are definitely two separate tasks that require different methods for good results.

I submitted weekly, 1) a ranked listing of all possible arcs and 2) normalized probability predictions for the separate test data set. As a response, I was informed on how good my arc predictions and probability predictions were, using AUC (Area Under the Curve) and KL-divergence scores, respectively.

For learning the structure I used mostly score-based local search methods along with a few perks. Parameters I learned using Bayesian Posterior estimates. As the course progressed, I tried out more advanced methods to enhance the results such as bootstrapping and model averaging.

2 Methods

I used the R language package 'bnlearn' (http://www.bnlearn.com/) for the results of this project. It has lots of features built-in for learning, manipulating and scoring Bayesian networks.

In what follows is a description of the methods I used to produce each submission's arc rankings and test data probability predictions.

2.1 First submission

For learning the structure of the golden standard network, I started out with implementing a simulated annealing method that uses BDe (Bayesian Dirichlet with likelihood equivalence) scoring with an equivalent sample size of 1.

The reasoning for these choices was familiarity. I've done some genetic algorithms before and simulated annealing is pretty similar in nature. BDe with ess=1 was explained in

the lectures of the course Probabilistic Models, so I felt that it would be a good place to start.

Algorithm 1 Outline of simulated annealing approach.

1: $bestState \leftarrow NULL$
2: $bestEnergy \leftarrow \infty$
3: $temperature \leftarrow INTIAL\_TEMP$
4: $state \leftarrow$ pick an initial DAG
5: $energy \leftarrow$ energy($state$)
6: **while** # iterations of this loop < 10000 **and** $energy$ above pre-defined threshold **do**
7: Decrease $temperature$
8: $newState \leftarrow$ add, remove or reverse an arc of $state$ so that result is still a DAG
9: $newEnergy \leftarrow$ energy($newState$)
10: **if** $P(energy, newEnergy, temperature) > \text{rand}(0, 1)$ **then**
11: $state \leftarrow newState$
12: $energy \leftarrow newEnergy$
13: **end if**
14: **if** $newEnergy < bestEnergy$ **then**
15: $bestState \leftarrow newState$
16: $bestEnergy \leftarrow newEnergy$
17: **end if**
18: **end while**
19: **return** $bestState$

The pseudo code for this approach is outlined in Algorithm 1. The initial state is chosen from a set of 5000 randomly generated Bayesian networks according to BDe score. This should ensure that we start off with a decent state.

Line 10 is essential for the flow of the algorithm. When $newEnergy < energy$ the if-statement is always true. However if this isn't the case, then according to the current temperature and difference between $newEnergy$ and $energy$ we will either change state or not. Early on it's easier as the temperature is higher, but as the loop gets executed several times, the temperature slowly decreases ("cools down") and the condition is less likely to be true.

For arc predictions I was originally going to use only one learned network. However as the task was not to submit a learned network but rather a ranked arc listing, and as I had some extra time, I decided to do something more powerful. I decided to learn 12 Bayesian networks using my simulated annealing approach, and then rank the arcs according to their occurrence in these learned networks.

This makes sense since the data set is very small (only 2500 generated data points), and learning a single perfect replica of the golden standard network from this limited data can be, if not impossible, at least extremely challenging. Additionally, especially because of the random nature of my simulated annealing approach, the multiple BNs learned will be different to each other but all will most likely contain some of the most probable arcs. Thus rankings arcs by occurrences in these BNs seems to make sense.

For test data probability predictions I chose the single best structure out of the 12 according to BDe. The logic being that the best structure will capture the most accurate representation of the conditional independencies present in the golden standard graph. For learning the parameters I had basically two choices: maximum likelihood estimates (ML) or Bayesian estimation.

While ML is conceptually much more simpler, it may predict zero probabilities, which are risky business for the KL-divergence as $\log(p/q) \cdot p \to \infty$ as $q \to 0$ when $p \neq 0$. Of course one can offset this with ad-hoc ϵ constant additions, but a more principled way of handling this problem is Bayesian estimation which adds a subjective prior distribution to the mix. I chose to use Bayesian estimation with an equivalent sample size of 10.

After having a fully specified BN (structure & parameters learned from training data), I predicted the probabilities for the test data set and normalized them to sum up to 1.

2.2 First extra submission

I pumped up the previous idea by learning 258 Bayesian networks and ranked the arcs by occurrence. I wanted to basically see if the performance of this idea could be improved by simply increasing quantity. As is shown in the next section in Table 1, performance didn't improve at all.

For probability predictions I tried an idea that I got from the 'bnlearn' package. From the 258 BNs, construct an averaged network structure which includes only the most significant (often occurring) arcs. The idea is that because each of the learned 258 BNs probably include some of the arcs present in the golden standard, the constructed averaged network will likely be close to the golden standard and thus be a good candidate for predicting the test probabilities.

After obtaining an averaged network, I learned its parameters using Bayesian estimation with ess=1 this time, thus assuming less about the uniformity of the underlying distribution a priori.

2.3 Second submission

I tried out the Tabu search and Hill climbing optimization techniques provided by 'bnlearn'. Both are score-based, local search methods which can be used to approximate several NP-hard problems.

Compared to simulated annealing, these methods are deterministic in nature unless some stochastic components are added on. Tabu search is essentially a more advanced version of hill climbing that can escape local maximum easier and is more efficient, as it keeps a 'tabu list' of k earlier states which shouldn't be revisited.

I generated 258 BNs using Tabu, 258 BNs using hill climbing and combined these with the 258 BNs generated using simulated annealing in the previous submission. All of these were learned using BDe with ess=1 as the score function. Out of these 774 BNs, I chose

the top 35 for arc ranking and probability predictions. The top 35 BNs contained 19 generated by simulated annealing, 13 by tabu and 3 by hill climbing, and were chosen by comparing the BDe scores of the generated networks.

For test data probability predictions with the top 35 BNs, I used model averaging. Essentially, it consists of two steps.

1. For each BN, predict the test data probabilities and normalize.
2. Combine into one set of predictions by applying weights (that sum up to 1; effectively a prior distribution) to the separate predictions and summing up pointwise.

Usually the weights are chosen so that the prior distribution is effectively uniform, for example in our case that would mean that all the weights would be $1/35$. I chose something more exotic and probably quite theoretically unjustifiable. I assigned weights according to each BNs respective BDe score; the higher the score, the higher the weight. The idea was to somehow prefer the predictions of networks that are closer to the training data (and thus the golden standard network).

2.4 Third submission

This time I drew a lot of ideas from the top performers on the scoreboard. I chose to implement bootstrap aggregation as it seemed to yield good AUC scores. The idea is to sample the training data uniformly with replacement multiple times, and for each sample learn a Bayesian network.

The sample sizes are the same as the training data size (n=2500), and because of this, approximately 63.2% of the data points in a single sample are unique and the rest duplicates. This allows us to learn diverse models that concentrate on different parts of training data. Essentially we're treating the training data as our population and sampling it in hopes to learn shape that the training data has derived from the original population.

Another idea I implemented was using 10-fold cross-validation for calculating the goodness of a Bayesian network. The basic idea is explained in Algorithm 2. The idea is that we have a Bayesian network structure, and we train its parameters using one part of the training data, and validate its goodness using the rest of the training data. We do this with several partitions and sum up the results. Using this "hold-out likelihood" measure, we can compare BNs learned with different algorithms and parameters without biases.

Algorithm 2 10-fold cross validation

$s \leftarrow 0$
$d_1, ..., d_{10} \leftarrow$ partition training data into ten subsets.
for $i = 1$ to 10 **do**
    $bn \leftarrow$ learn parameters of BN using all other partitions except d_i.
    $p \leftarrow$ predict probabilities of events in d_i and sum them up.
    $s \leftarrow s + p$
end for
return s

So I used bootstrapping to learn 150 BNs with Tabu and 150 with simulated annealing. I then chose the 50 best according to hold-out likelihood, and used these for arc ranking by occurrence. For test data probability predictions I used all 300 BNs with model averaging using an uniform prior.

Originally I was going to return probability predictions with only the top 50 BNs. However when predicting probabilities for the new development data set and comparing the predictions to the real probabilities, I got a significantly higher KL-divergence when using only top 50 compared to all 300 BNs. I took this as a signal that I should probably use all 300 BNs also for test probabilities. I suspect that the top BNs give to some data points very erroneous predictions.

2.5 Second extra submission

For the extra submission I took the ideas of the last submission to the extreme by generating 1400 BNs (with Tabu only). I then used model averaging for test probabilities and 100 best BNs (according to hold-out likelihood) for arc predictions. My results got only worse, indicating again that quantity doesn't necessarily help.

2.6 Fourth submission

For the final submission I tackled the problem with one of the test data points, specifically line 104, getting very large probabilities with many of the top BNs learned. At a course meeting on Tuesday, it was suggested that the probability shouldn't be too high. However it was also established that tackling this problem in a principled way may be very challenging.

My method is to predict the (normalized) test probabilities with each BN, and discard those that give to any point, a probability higher than 0.3. In a Bayesian context this can be justified so that I establish a subjective prior belief that none of the test data points should get a larger portion of the probability mass than 0.3. If some BN, however, does assign a larger portion, I set a prior probability of 0 for it, effectively declaring it as impossible.

In practice what I did was that I generated 250 structures using simulated annealing on bootstrap samples. I then learned the parameters using Bayesian estimation with ess=1. Out of these, 62 survived the discarding explained above. I used model averaging on these 62 (with uniform prior) to predict test data probabilities. While the probability for line 104 is still relatively high, absolutely it is much lower than in my previous submissions.

Improving the arc rankings at this point seemed pretty challenging as I had already obtained a AUC of 0.997 in an earlier submission. I chose the best 50 out of 250 BNs by calculating BDe scores and hold-out likelihoods for each, and using a voting scheme to select the best. The reasoning here is that if a BN gets both, a high BDe score and a high hold-out likelihood, it should probably be favored over one that excels on only either. I then used arc ranking by occurrence in the top 50 BNs as I have done before.

3 Results

Submission #	1st	1st extra	2nd	3rd	2nd extra	4th
AUC	0.978	0.974	0.980	**0.997**	0.987	**0.997**
KL-divergence	3.060	2.338	2.341	1.321	1.564	**1.204**
AUC improvement		-0.004	+0.002	**+0.017**	-0.010	0
KL improvement		-0.722	+0.003	**-1.017**	+0.243	-0.117

Table 1: AUC and KL-divergence scores obtained from the submissions. The highest scores have been highlighted. Improvements are compared to best submission so far.

The results obtained from the submissions have been tabulated in Table 1. Overall, the obtained AUC scores have been very high throughout the submissions. This means that I have succeeded in the arc prediction task fairly well.

Test data probability predictions, on the other hand, have proved to be much more of a challenge, as can be seen from the KL-divergences obtained. Improvement definitely has happened during the course of the submissions, but the predictions are still far from the truth.

3.1 Arc rankings

Right off the bat with the first submission, I got a pretty high AUC (0.978) by learning 12 BNs with simulated annealing and ranking arcs by occurrence. This indicates at least two things. One, that my simulated annealing algorithm seems to be working correctly, and two, you really don't have to learn that many structures to find most of the arcs present in the golden standard.

I suspect that if I had used only one structure instead of all 12 BNs for arc ranking, I would've gotten a much worse AUC score since 1) my learning algorithm probably only learns some portion of the true arcs, and 2) there may be many equivalent structures to the golden standard which posses slightly differing arcs.

In my first extra submission I tried learning considerably more BNs (258) in hopes that my AUC score would increase with quantity. This didn't happen, actually quite the opposite; my AUC score dropped down to 0.974, though I suspect this decrease has more to do with stochastic fluctuation than anything else.

12 BNs seems to be enough to take this method of arc ranking to its limits, and learning more doesn't seem to help at all. I suspect this has to do with the fact that learning more structures also means learning more bad structures.

For the second submission I generated a total of 774 structures with Tabu search, Hill climbing and simulated annealing. Out of these I chose the 35 best. This way I would filter out the worst structures so that I'm left with the very best (according to BDe score) that most likely contain the arcs of the golden standard. This improved my result

marginally (by 0.02 from my best so far) which may be attributed to either A) selecting the 35 best, B) using other algorithms as well, or C) plain luck.

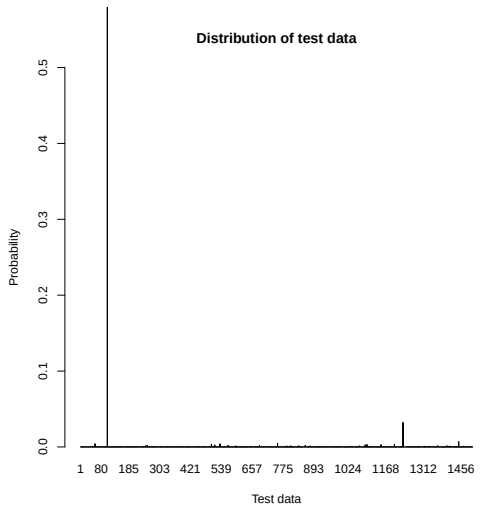
The third submission proved to contain the most fruitful improvement so far: bootstrap aggregation. Another change I made was to select the top structures not by BDe score but by holdout likelihoods (on 10-cv). These modifications allowed my AUC to grow by 0.017 bringing it up 0.997, the best obtained by me overall.

Mostly I attribute this improvement of AUC to bootstrap aggregation, which changes the nature of the structures learned. Instead of trying to learn structures for the whole training data, we learn structures for parts of the training data. This makes our structures more varied, and allows them to learn a specific part of the training data well.

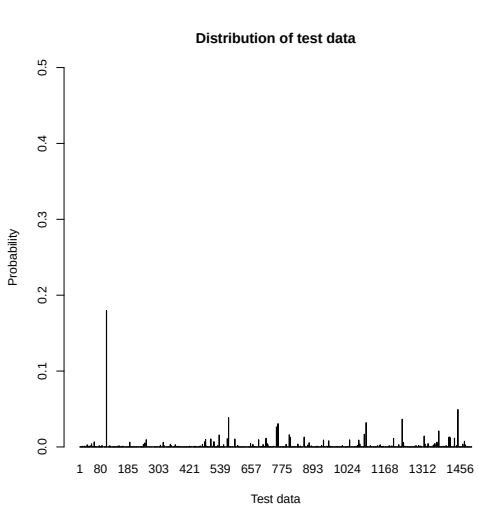
For the second extra return I tried taking the previous idea to the extreme by generating loads of BNs using bootstrap samples and choosing the best according to holdout likelihoods. Once again, my results didn't improve with quantity. Actually my AUC dropped considerably, which I find quite odd. Maybe the fact that I used only Tabu search and no simulated annealing at all, changed the nature of my resulting graphs.

My fourth and last arc predictions were very similar to the third ones, and I only changed the way in which I chose the best BNs. At this point improving the AUC seemed very much like a shoot in the dark. Maybe some integer linear programming methods (e.g. GOBNILP) would've yielded even better results and I would've been able to get an AUC of 1.

3.2 Test data probability predictions



Test data probability predictions on 1st submission (worst).



Test data probability predictions on 4th submission (best).

Figure 1: Progress of probability predictions visualized. Worst submission vs. best submission. Notice change in distribution of probability mass.

For my first submission I simply picked the best BN out of the 12 learned by simulated annealing. I then used Bayesian estimation with $ess = 10$ to learn the parameters and predict probabilities. My score was relatively bad at 3.060, and I quickly learned that just because you have a good structure, doesn't mean you have good probability predictions.

One problem was with the large ess selection, which seemed to cause the line 104 spike to become even larger. Another problem was using only one network, as this network will contain arcs not occurring in the golden standard network, and these arcs can cause the predicted probabilities to be quite far from the truth.

For the first extra submission I addressed these problems by constructing a averaged network out of all BNs learned, and using ess=1, thus assuming less a priori. This improved my results quite a bit, probably because the averaged network is built with all BNs taken into consideration, meaning that less untrue (and thus distorting) arcs should be present. Also ess=1 helped mitigate the spike present in the test data at line 104.

For the second submission I used model averaging on top 35 BNs using a weird prior. It didn't improve best KL-divergence found so far.

For my third submission I continued using model averaging as I had done in my second submission, but averaged over all BNs instead of just some top selected. This seemed to improve the results dramatically, as can be seen in Table 1. Of course in this submission I also implemented bootstrapping, which may have had an impact on the results as well.

It seems as if in this probability prediction task, it's better to use all models than just the selected top models. The opposite seems to be the case in arc ranking. It seems as if the top structures actually tend to distribute a huge portion of the probability mass to the line 104 in the test data, while less optimal structures tend to distribute the mass more evenly.

In my fourth submission I addressed this issue by simply dropping BNs that distribute a large portion of the probability mass to any single point in the test data. For the remaining BNs, I used model averaging. The results have been depicted in Figure 1 on the plot to the right. While data point 104 still gets a relatively large mass, at least it's not as bad as in previous submissions.

3.3 Other observations

3.3.1 The line 104 problem

For line 104 in the test data, the following probabilities were predicted in different submissions:

1st	1st extra	2nd	3rd	2nd extra	4th
0.863	0.362	0.423	0.386	0.419	0.180

The correlation between these and their respective KL-divergences is about 0.815. This suggests that maybe even more aggressive bounding may be in order to achieve lower KL-divergences.

3.3.2 Tabu search vs. simulated annealing

In Figure 2 is a comparison of Tabu and simulated annealing with and without bootstrapping. Clearly in both cases, bootstrapping results in slightly better structures.

Tabu seems to generate consistently good structures, while simulated annealing tends to generate networks of varying quality. This isn't necessarily a bad thing, since it was established earlier that for test data predictions, it may be better that the structures aren't too optimal.

Simulated annealing performs very slowly, probably because I wrote it entirely in R, which is a notoriously slow language. Tabu works much faster (depends on the size of the tabu list) and yields better results on average. It would seem that at least for the arc prediction task, Tabu has a better return on invested computation time.

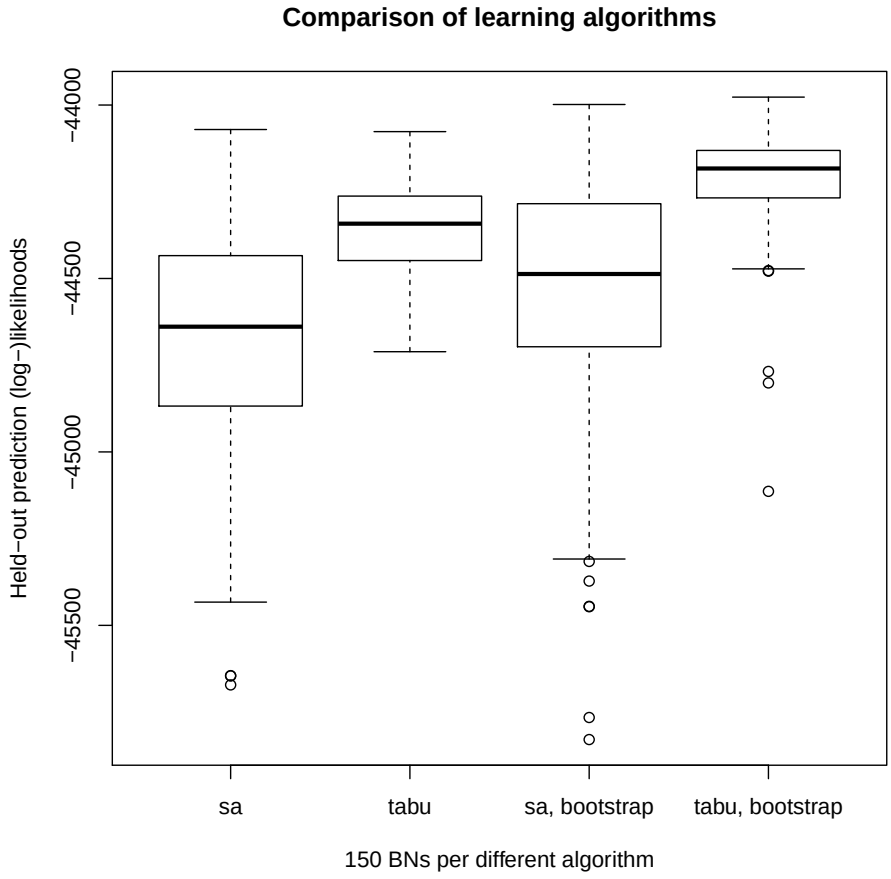


Figure 2: Distribution of hold-out likelihoods (summed over hold-out sets in 10-fold CV) for BNs generated with simulated annealing (=sa) and tabu search; both with and without bootstrapping.

4 Discussion

Overall the problem was very interesting and I feel that I learned a lot doing it, not only about Bayesian networks, but about data analysis in general. There were several different approaches and methods for solving the problems, but having a limited number of submissions meant having to consider only a small part of them.

If I would've had more time and the course would be longer, I would've probably tried out the GOBNILP integer linear programming method to improve my AUC score. Another thing I would've done was to study the effect of different *ess* more systematically on structure and parameter learning. For test data probability predictions, I would've tried bounding the probability mass of line 104 even more aggressively than what I did in my last submission.

One brilliant idea, I heard another person taking the course suggest, was to actually try setting up a golden standard network of your own and testing what kind of methods seem to work well and which don't. Although I implemented some code to do this, I didn't have time to play around with it.

5 Appendix

5.1 Running the code

In what follows is a simple example of using my code. We will learn 3 BNs using simulated annealing, predict and plot test probabilities, produce a ranked arc listing by occurrence, and finally predict development probabilities and calculate the kl-divergence to the real development probabilities.

First `cd` into the folder containing my code and open-up the R prompt with command `R`. When in R, use `source("dev_env.R")` to load up my code. It may throw an error if some packages are missing. Proceed to execute the following commands, one at a time and in order:

```
bns <- simulated_annealing(3)
probs <- model.averaging.probs(bns, test.data)
probs

plot.probs(probs)
dev.off()

ranked.arcs(bns)

kl.divergence(dev.probs, model.averaging.probs(bns, dev.data))
```

5.2 Submission diaries

Below is listed all diary entries returned during this project.

PiPM 2013, First return

Eric Andrews, 013594545

December 18, 2014

Progress

I chose the R library "bnlearn" for this project because of two reasons. First of all, it has ready-made implementations for manipulating, learning and scoring Bayesian networks, and second, it offers nice vizualizations of the networks.

Instead of using the learning algorithms provided by "bnlearn", I chose to write my own simulated annealing algorithm with the help of scoring and graph-manipulation tools provided by "bnlearn". My main motivation for this was to start of with something that is somewhat familiar to me, and to really understand how the learning happens.

I also had to implement the logic for calculating test data probabilities from learned network structure and parameters, as I couldn't find a built-in procedure.

My simulated annealing algorithm starts of by generating 5000 random DAGs, and then chooses the one with the best score as its initial state. It then, for 10000 rounds, randomly creates, removes and reverses arcs in that DAG and makes sure that the resulting graph is a DAG as well. In a typical simulated annealing fashion, it keeps track of the current and best DAGs (states) so far, and allows smaller and smaller changes as the number of rounds increase.

As the score function I use BDE with an imaginary sample size of 1. To learn the parameters of the network, I use Bayesian estimation.

My approach to ranking the arcs and calculating the test probabilities is as follows.

1. Learn 12 DAGs using my simulated annealing approach.
2. Rank all arcs according their occurrences in the 12 DAGs.
3. Choose the best DAG (according to BDE score), learn its parameters from the training data, and calculate the normalized probabilities of the test data.

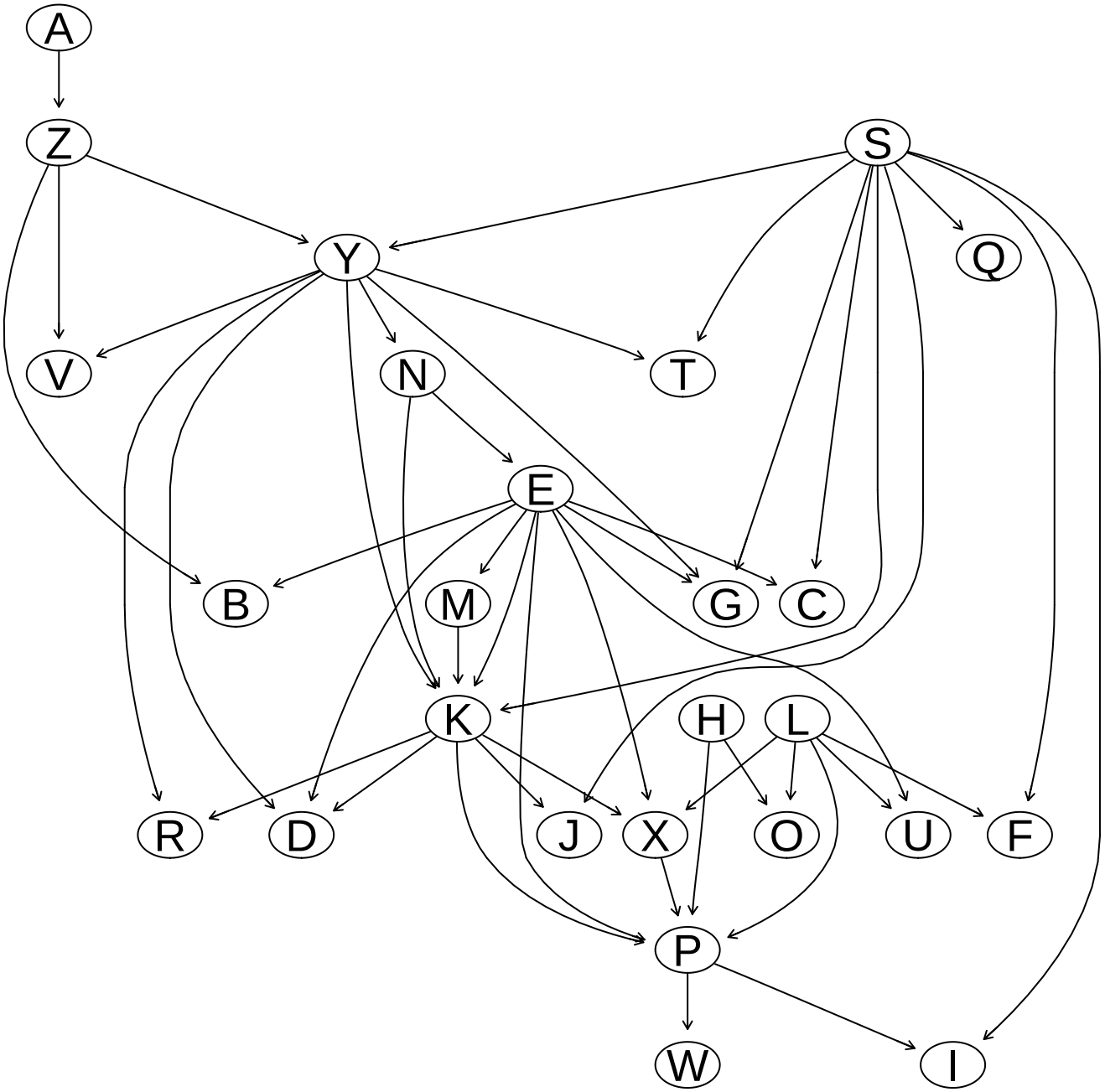


Figure 1: Network with best BDE score

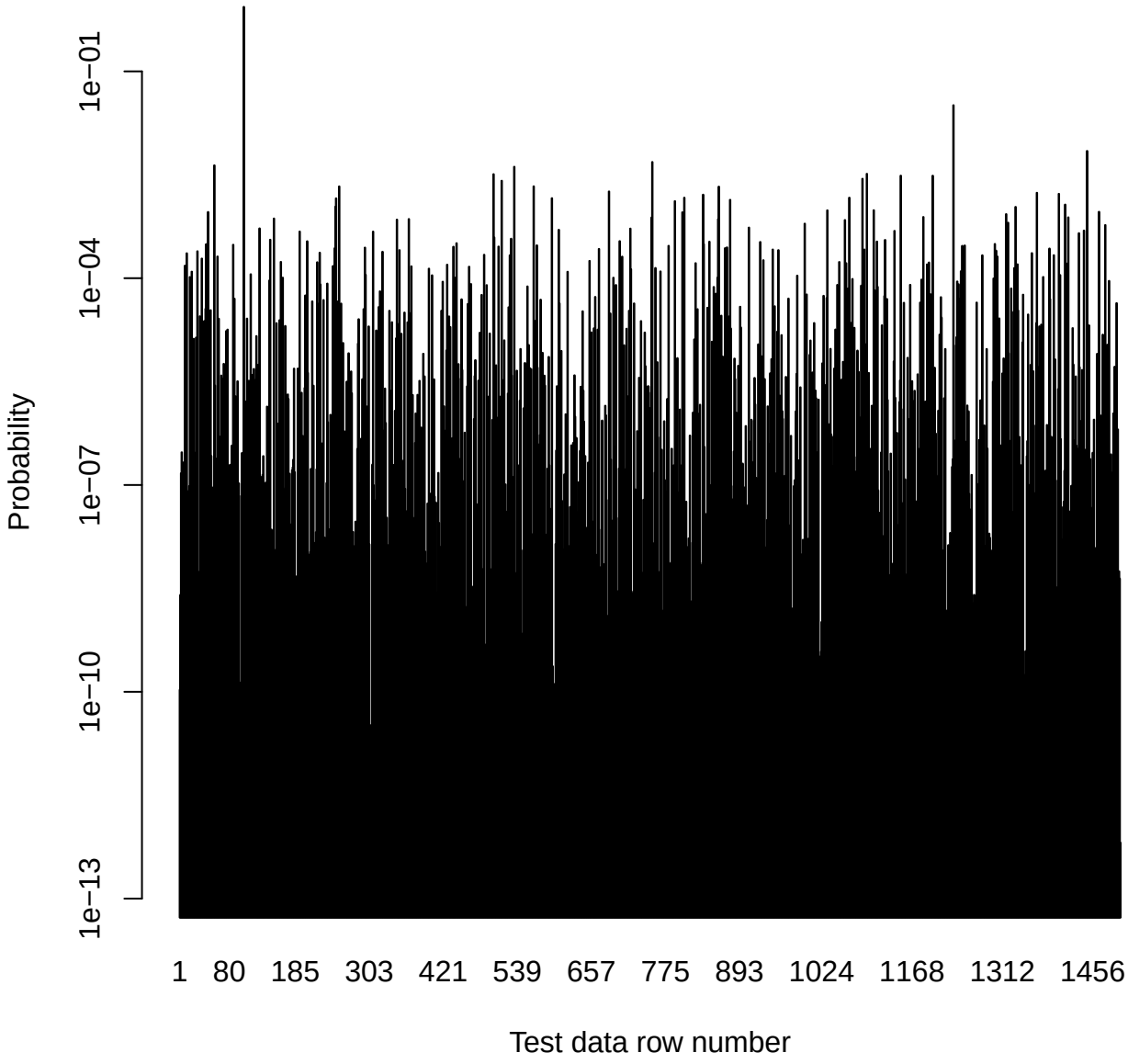


Figure 2: Normalized probabilities for test data. Notice logarithmic scale and the spike at item no. 103.

PiPM 2013, Second return

Eric Andrews, 013594545

December 18, 2014

Progress

For last week's extra return I generated 258 BNs via the simulated annealing technique, while first week I generated only 12. However my AUC score didn't improve (actually decreased by 0.002) even though I used arc ranking by occurrence in both cases. I suspect this is because when there are more BNs, there are also more badly fitted BNs which affect the arc rankings negatively.

On the other hand, my KL divergence score improved by 0.722. I used a different technique (averaged network rather than best network) but I suspect this has more to do with changing the equivalent sample size (ess) from 10 to 1, thus assuming less about uniformity apriori. As we're focused on, not building a predictive model, but rather a descriptive model, overfitting isn't really an issue and thus ess can be set low for better results.

For this second return, I also ran the Tabu search and Hill climbing algorithms that are provided by R's 'bnlearn'-package. A boxplot of the results of each algorithm is shown in Figure **??**. From the figure we can see that while simulated annealing has some of the best networks, it's also very spread out, so it contains many networks with low scores. Tabu search, on the other hand, produces fairly good networks on average but doesn't necessarily contain the best. Of course parameter selections have a huge impact and my simulated annealing runs substantially more slower than the Tabu search provided by R.

This week for arc ranking, I use occurrence once again, but this time I consider only the top 35 BNs out of all 774 generated by simulated annealing, tabu and hill climbing. Score function used for selecting top BNs is BDe with an ess of 1 (on training data). Out of these 35 BNs, 19 are generated by simulated annealing, 13 by tabu, and 3 by hill climbing.

For test data probability prediction, I calculate test data probabilities with each of the top 35 BNs chosen earlier. I then calculate a weighted (pointwise) sum of the probabilities, where each probability set is weighed according to the corresponding BN's BDe score. The weights (prior distribution) used is illustrated in Figure **??**. The final distribution of test data probabilities is illustrated in Figure **??**. The Kullback-Leibler divergence of last week's probabilities compared to these new probabilities is about 1.26, when last week's probabilities are held as the "true" distribution.

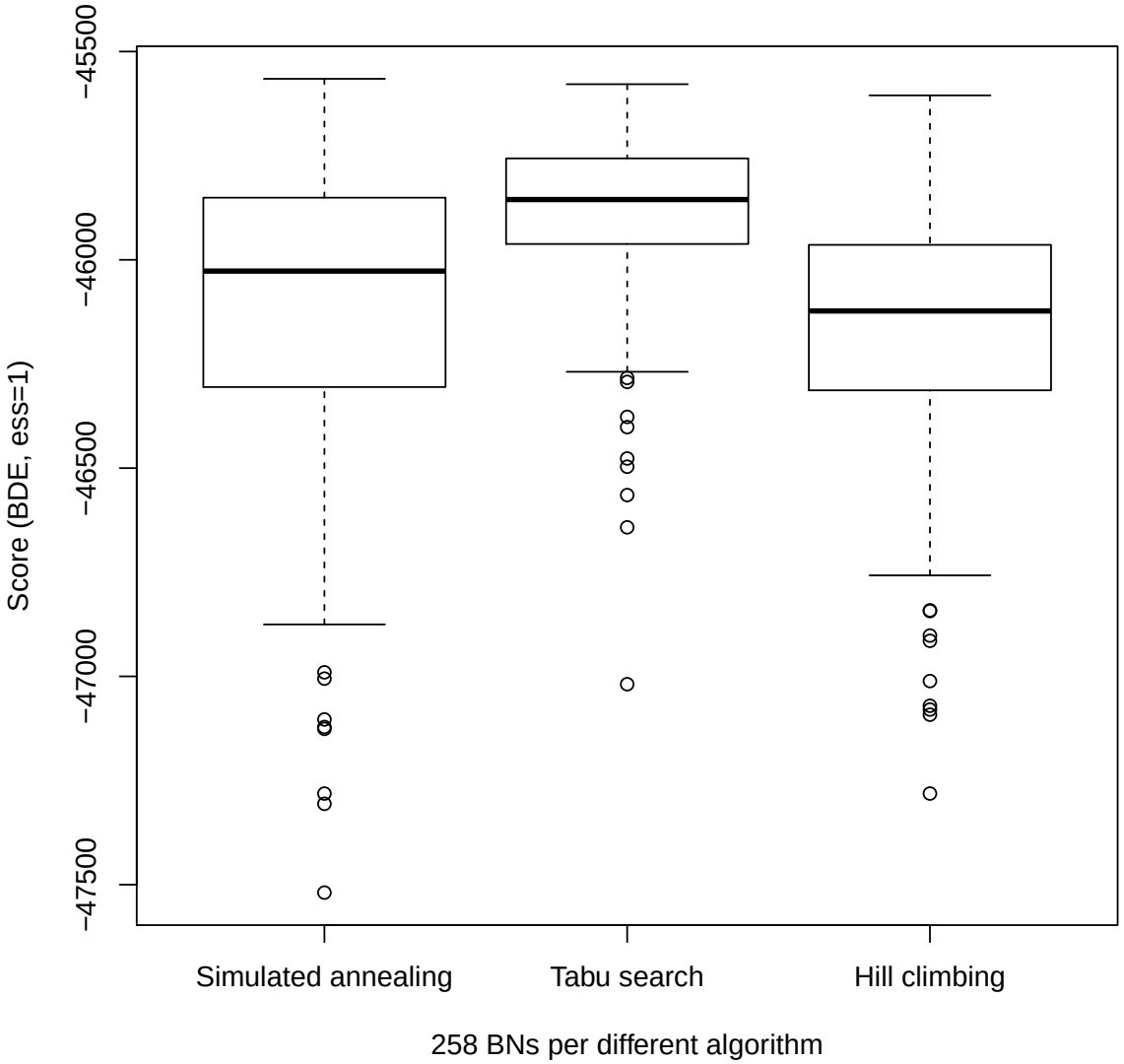


Figure 1: I generated a total of 774 BNs (258 per algorithm) using simulated annealing, tabu search and hill climbing.

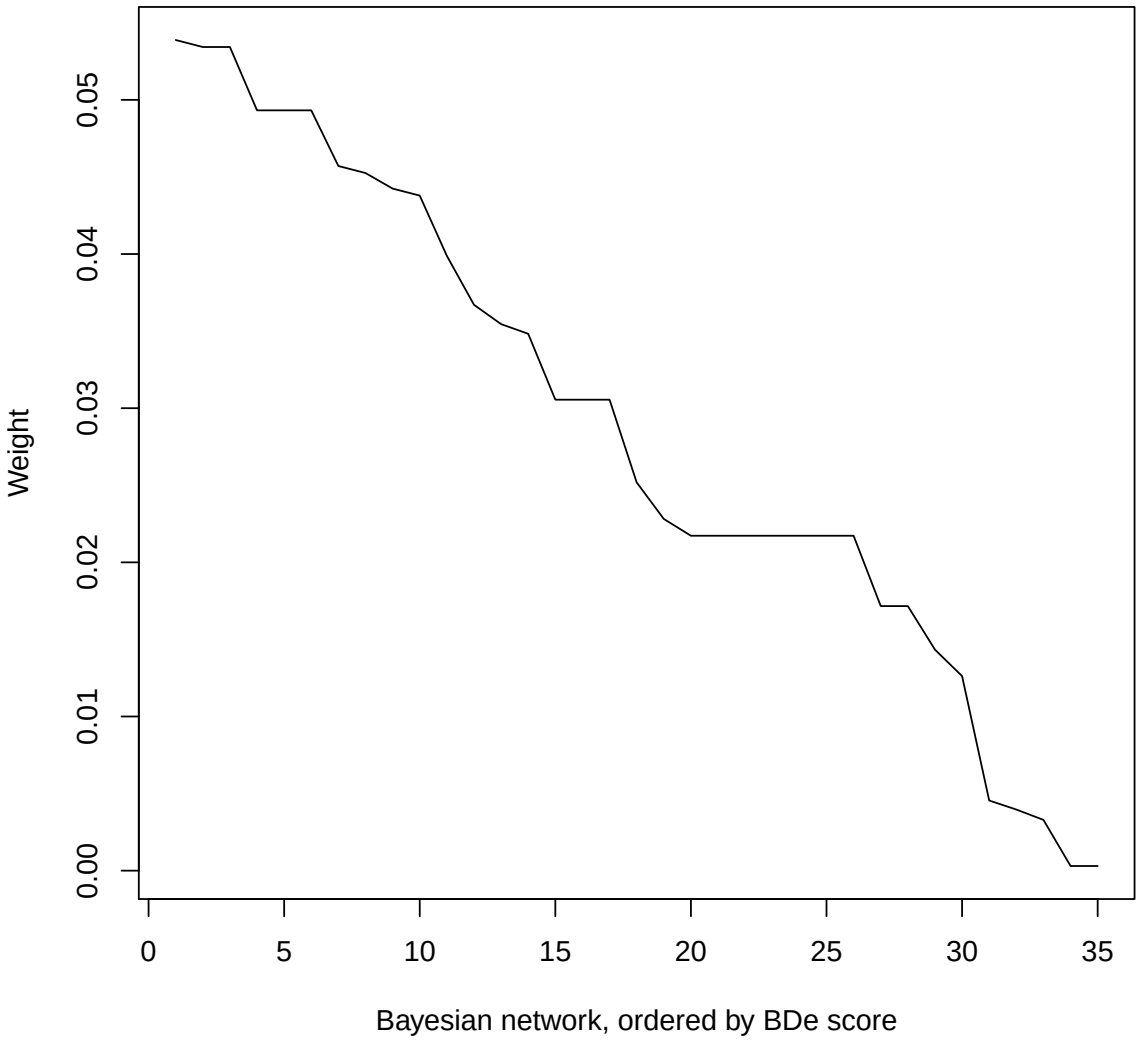


Figure 2: Weights (prior distribution) over the 35 best BNs when averaging test data probabilities.

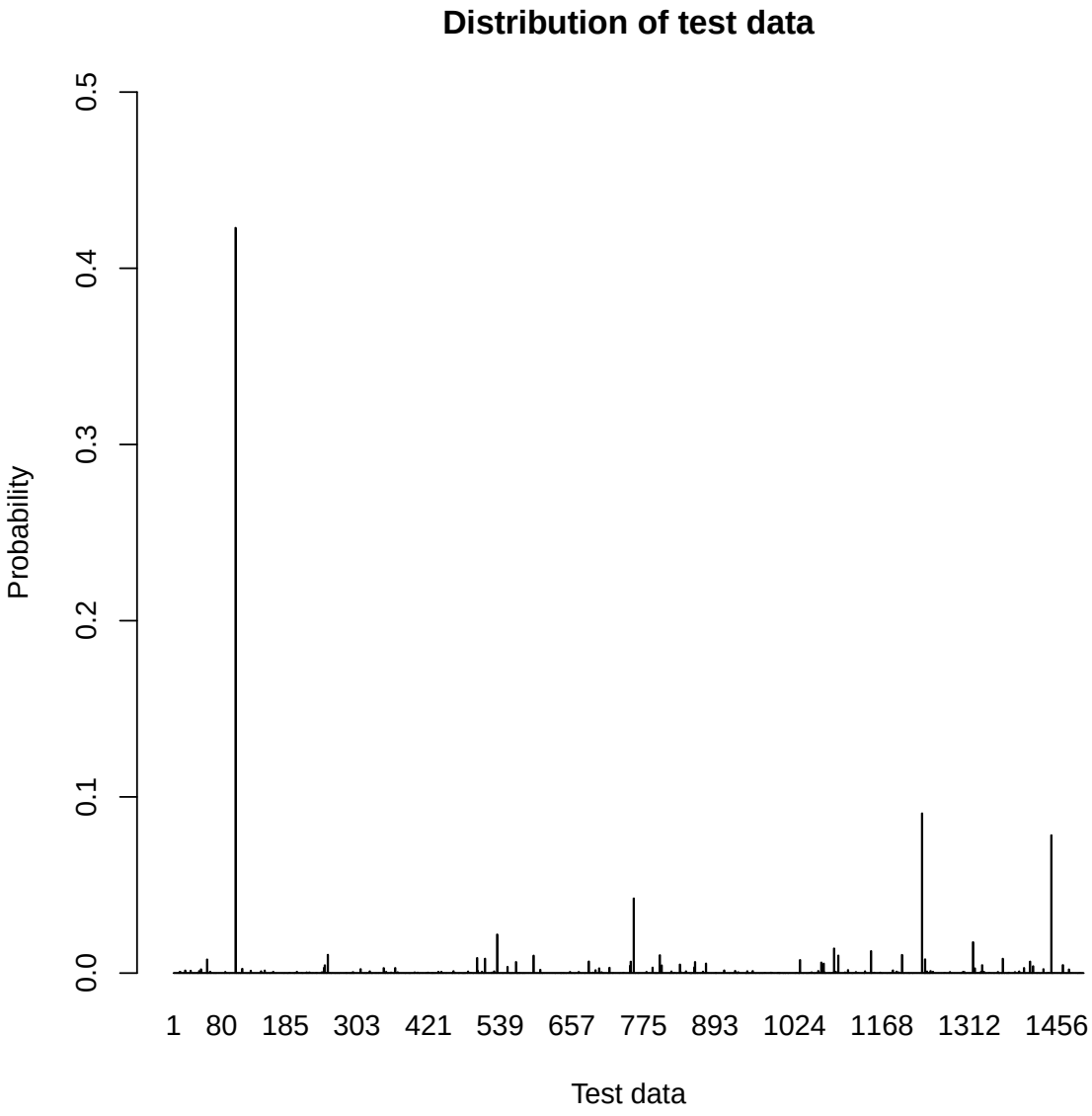


Figure 3: Distribution of predicted test data probabilities.

PiPM 2013, Third return

Eric Andrews, 013594545

December 18, 2014

Progress

This week I worked primarily on the following things:

- **Bootstrapping:** resampling the training data (with replacement) to create new training sets to learn BN structures from.
- **Faster probability predictions:** I used the main ideas of Arto's solution (posted on the course moodle forum) to speed up my earlier slower procedure.
- **Cross-validation, held-out likelihood:** use 10-fold cross validation to train parameters of BN, and use the held-out sets to calculate prediction likelihoods (which are summed up at the end).

This week my strategy was to generate 150 BNs with simulated annealing and 150 with tabu search. This time instead of using the entire training data to learn the structures, for each algorithm run, I used a different bootstrapping sample (of size 2500). Out of these total 300 generated BNs, I chose the 50 best according to the held-out prediction likelihoods.

Out of the 50 best, 12 were generated with simulated annealing and 38 with tabu. I used these 50 for arc ranking by occurrence, but for predicting the test probabilities (using model averaging), I actually decided to use all 300 BNs instead of only the 50 best. The reason for this was that for the development data, I got a KL-divergence of 2.47 with 50 best and 1.42 with all BNs. So I thought this may be a good signal that I should use all BNs instead of only the 50 best.

Results

In Figure **??**, we see a comparison of the BNs generated by simulated annealing and tabu search. In both cases, bootstrapping seems to generate better networks. Tabu seems to generate consistently good networks, while simulated annealing has much more variability. Overall, tabu has a better "return on computation time" as it runs something like about 50 times faster than my simulated annealing implementation.

One interesting result is that increasing the equivalent sample size seems to improve the held-out likelihoods as can be seen in Figure **??**. Whether this is desirable or not is an open question for me. Maybe next time I should try setting a higher ess?

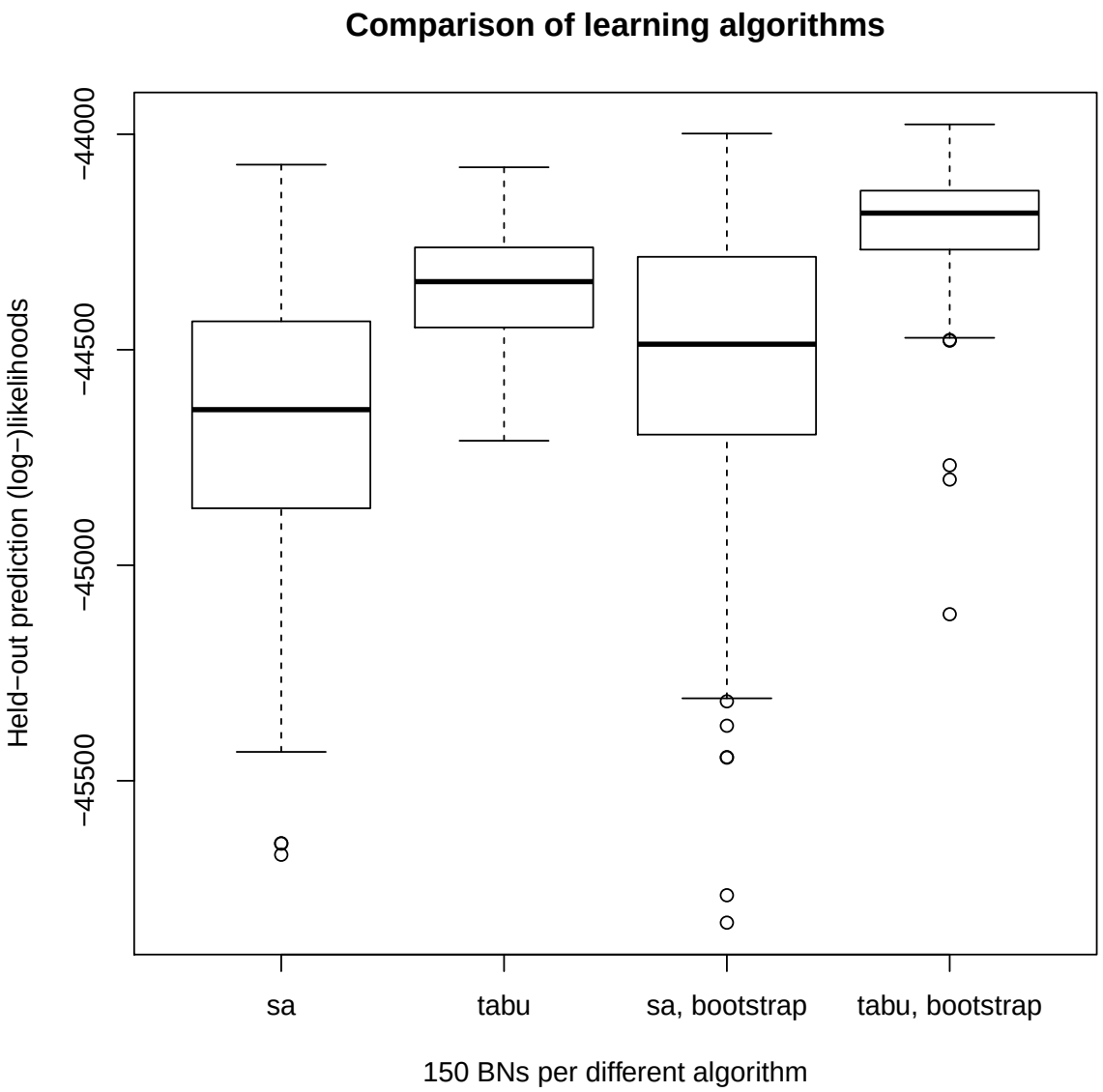


Figure 1: Distribution of predicted likelihoods (summed over held-out sets in 10-fold CV) for BNs generated with simulated annealing (=sa) and tabu search; both with and without bootstrapping.

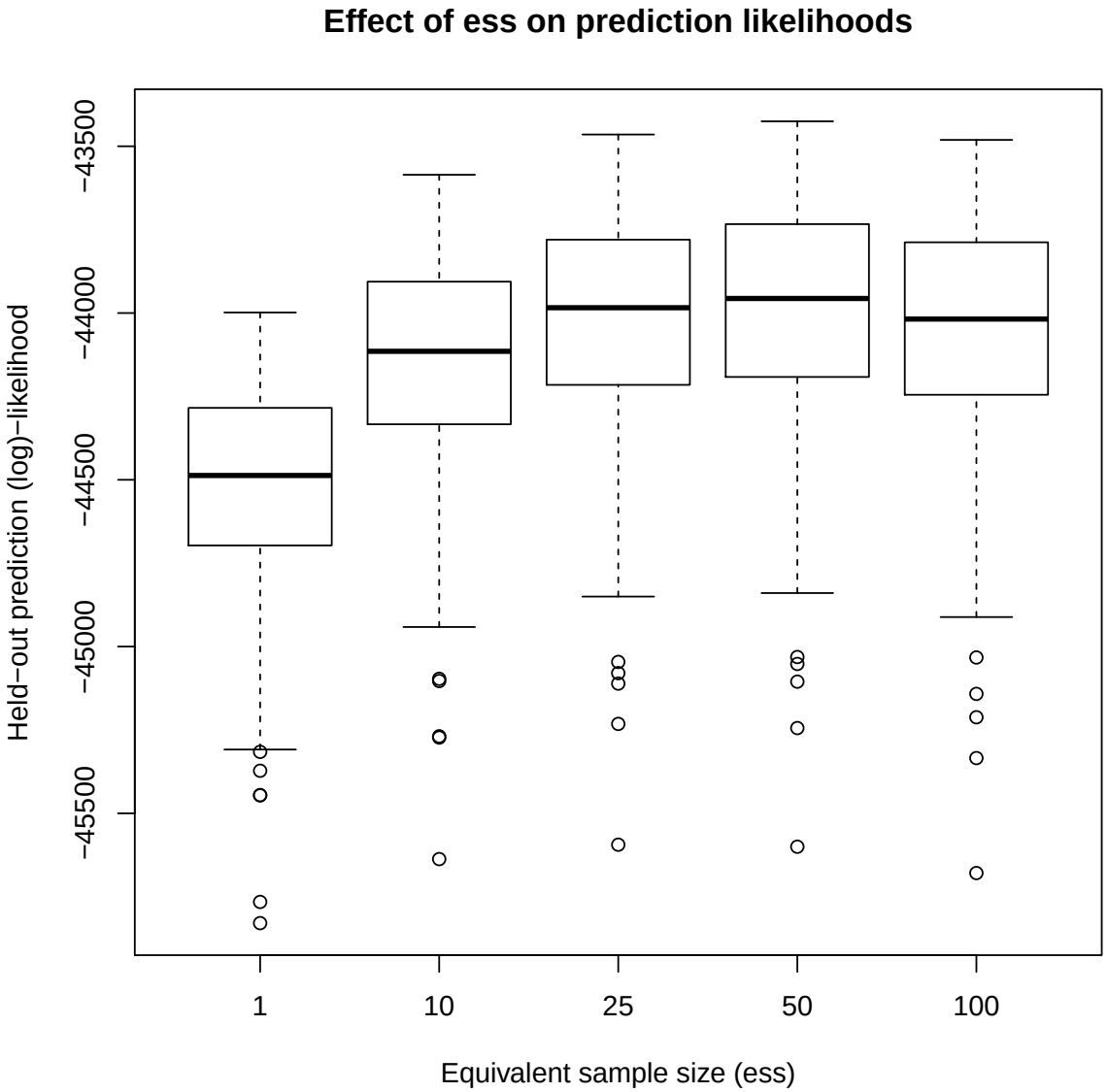


Figure 2: Above is illustrated the effect of equivalent sample size on parameter learning. The structures were learned earlier by simulated annealing (150 BNs).

PiPM 2013, Fourth return

Eric Andrews, 013594545

December 18, 2014

Test data probabilities

One of the biggest problems with the test data probability predictions has been the high spike occuring in the test data set in line 103. For some reason, many highly scoring BNs seem to give high probabilities for this data point. Trying to combat this in a principled way seems challenging.

My approach to solving this problem is to discard BNs that give a probability of more than 0.3 to any point in the test data. I am then left with BNs that I use to predict test probabilities using model averaging. In a sense I am setting up a subjective prior belief that the learned BNs shouldn't give high probabilities to any data point in the test data set.

Basically I generated 250 BNs with bootstrapping and simulated annealing. I then discarded those that give a probability higher than 0.3 to any point in the test data set. After this step I was left with 62 BNs that I use to predict the test probabilities with, using model averaging. The resulting distribution has been illustrated in Figure **??**

Arc rankings

I tried to use constraint-based methods such as grow-shrink, incremental association along with a hybrid algorithm called max-min hill-climbing. However as is demonstrated in Table **??**, these don't seem to work very well.

I decided to take the top 50 BNs out of the 250 generated earlier according to both their BDe score and hold-out likelihood (using a voting scheme based on order). The AUC compared to the first 52 arcs of submission #3 (best so far) is 0.9995498 so the arc rankings should be very similar. At this point getting a better score seems to be a shoot in the dark.

Grow-Shrink	Incremental Association	Max-Min Hill-Climbing	Tabu	Hill-Climbing
0.695	0.445	0.848	0.986	0.967

Table 1: For submission #3 I got my best AUC so far (0.997). I took the first 52 arcs from that submission and held them as the "true" arcs. Then using 250 boostrap resamples I learned BNs using the different methods enumerated above and produced an corresponding arc ranking by occurrence for each. I then calculated the AUC between these two. As the arcs of my submission #3 are relatively close to the real deal, and as my learning algorithms seem to produce BNs with 52 arcs (mode), this seems like a good approximation.

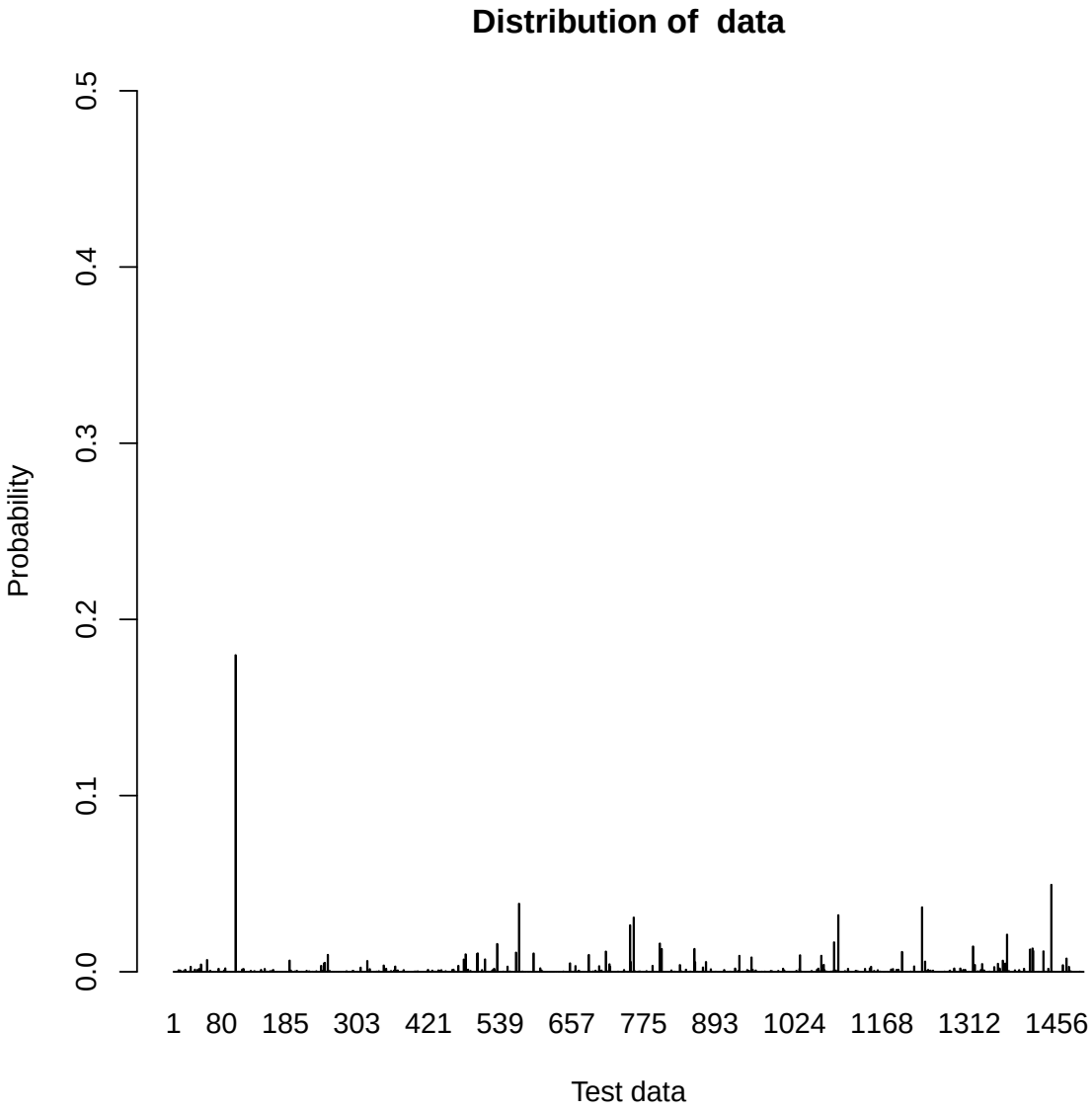


Figure 1: Test data probability predictions. Highest probability is element no 103 with a predicted probability of 0.18