Structural inference in gene regulatory networks

Kiril Ralinovski

April 16, 2013

Abstract

Bayesian networks are suited to model gene regulatory networks, but often the complete structure of such a networks is not known. Bayesian methods allow to infer structural information of such networks even when only sparse data is available. Markov Chain Monte Carlo methods haven been used for this purpose and allow to easily incorporate prior knowledge about the network's structure. In this paper the author examines and evalutates the commonly used techniques.

1 Introduction

Gene transcription and protein synthesis are governed by regulatory mechanisms. Certain genes either inhibit or activate other genes. With the advent of DNA microarrays we have entered the genomic are and are able to transcribe DNA and mDNA targets. But as always new technologies and data provide us with new problems to solve. This problem in bioinformatics concerns itself with the reverse-engineering of these networks. This task is not as straightforward it may first appear. Taking samples is an expensive and sometimes somewhat noisy process. This means that we do not have the privellege of taking as many samples as we'd like to have and we may sometimes be limited to less than 100 samples. If we observe the figure we can see that these structure has a graphical structure. This is why Bayesian Networks are very well suited to represent them.
Bayesian networks consist of a tuple (V,E), where V is a set of nodes(vertices) and E is a set of edges that describe the relationships between the nodes. Each vertex represents a random variable and an edges represents conditional dependence between two random variables. This

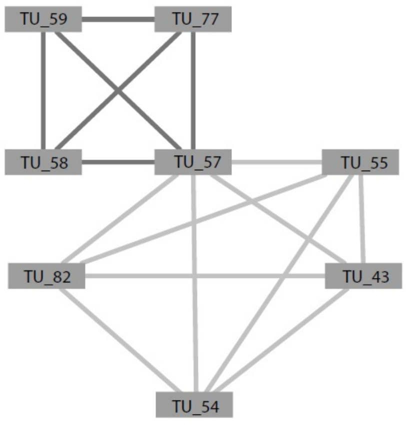
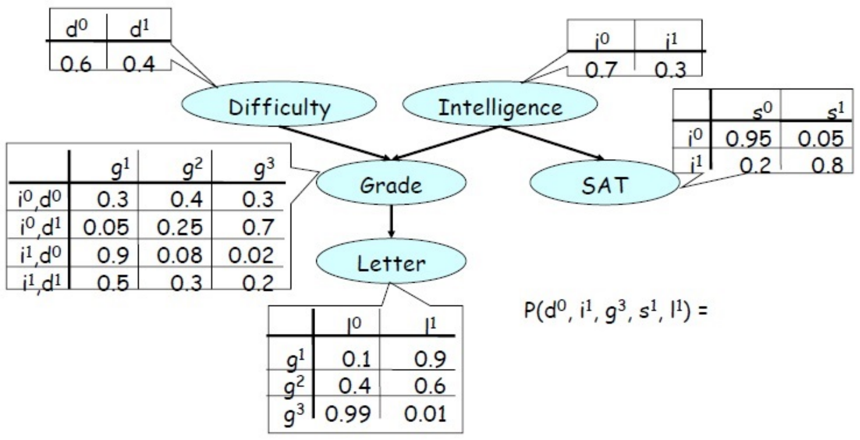


Figure 1: An example gene regulatory network in Lactobacillus plantarum [6]

is why the graph also has to be acyclic, since it is impossible to have random variable that mutually depend on each other. The other component of the Bayesian Networks is the family of conditional probabilities that describe these relationships. A simple example of a Bayesian Network can be seen in [**?**].

If we have a set of random variables $X_1, X_2, ..., X_n$ and the node indices be denoted by $i \in 1, ..., n$. Then the probability of a state is defined by
$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_{pa[i]})$
Where $X_{pa[i]}$ is the set of parent nodes of i. A cycle would arise if X_i was a parent of one of its parents, but as we said there can be no cycles in a BN. Some times we'd like to have cycles, because they represent feedback loops. But it is possible to extend the model to be able to cope with this kind of problem. If we unfold the graph over time we would get rid of this problem.

If M is a tuple of the graph and the parameters we seek to maximize
$M* = argmax_M P(M|D)$
It is impossible to go through all of the possible graphs even for a small number of nodes, since their number grows super-exponentially and this particular optimization problem has already been proven to be NP-hard. We could resort to using hill-climbing or simulated annealing, but due to the sparseness of our data the posterior probability is very likely to be diffuse. Therefore it is unlikely that P(M—D) will be adequately represented by a single graph M*. An efficient way to come around this is to take sample graphs from the posterior distribution. By doing this we obtain a set of graphs, which have high posterior



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Figure 2: An example of a Bayesian Network. The student's grade depends on the difficulty of the course and intelligence of the student. His SAT score also depends on his intelligence. And whether or not he gets a recommendation letter from the teacher depends on his grade in the course.

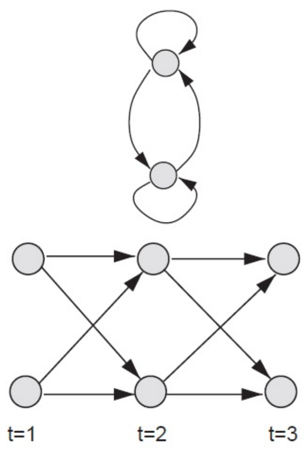


Figure 3: An example of a dynamic bayesian network. We unfold the first graph, which is cyclic over time.

probability and we can infer some of the characteristics of the generative model. Fortunately adequate results can be obtained with the use of Markov chain Monte Carlo simulations. The way we generate this set is by creating a new graph graph M_{new} from M_{old}. The probability of accepting this new graph is determined by the Metropolis-Hastings
$P_{MH} = min(1, \frac{P(M_{new}|D)}{P(M_{old}|D)} \cdot \frac{Q(M_{old}|M_{new})}{Q(M_{new}|M_{old})})$
where $P(M|D) = \frac{1}{Z} P(D|M)P(M)$, where Z is a normalizing factor defined by
$Z = \sum_M P(D|M)P(M)$.
The normalizing factor is a sum over all graphs. Fortunately this factor cancels out.
$Q(M_{old}|M_{new})$ represents the probability of going from M_{old} to M_{new}. We create M_{new} out of by applying a single simple operation on M_old. The possible operations are: add an edge, remove an edge or reverse the direction of an edge. Theoretically the reversal of the direction of an edge is simply the removal and addition of an edge, but by omitting the possibility of an edge direction reversal the posterior probability of some edges changes. It has to be taken into account that the addition of an edge and edge directional reversal could lead to the new graph being cyclic. Therefore it has to be made sure that the new graph is acyclic. Since the probability of choosing a certain action is uniformly distributed then $Q(M_{old}|M_{new}) = \frac{1}{\eta(M_{old})}$, where η is represents the number of neighbouring graphs. Therefore
$\frac{Q(M_{old}|M_{new})}{Q(M_{new}|M_{old})}) = \frac{M_{old}}{M_{new}}$

```
Initialize graph $G^{(1)}$ set $t = 1$, $G \leftarrow G^{(1)}$
while $t < T$ do
    Propose $G' \sim Q(G'; G)$
    Accept G' with probability min(1,$\alpha$), $\alpha = \frac{P(G'|X)Q(G;G')}{P(G|X)Q(G';G)}$
    if G' is accepted then
        $G^{(t+1)} \leftarrow G'$ , $G \leftarrow G^{t+1}$
    end
    else
        $G^{(t+1)} \leftarrow G$, set $t \leftarrow t + 1$
    end
end
```

Algorithm 1: Metropolis-Hastings [4]

In most cases we would like to incorporate some sort of prior knowledge into our model. In this paper two methods will be discussed. Each network is represented by an adjency matrix A. If an entry A_{ij}

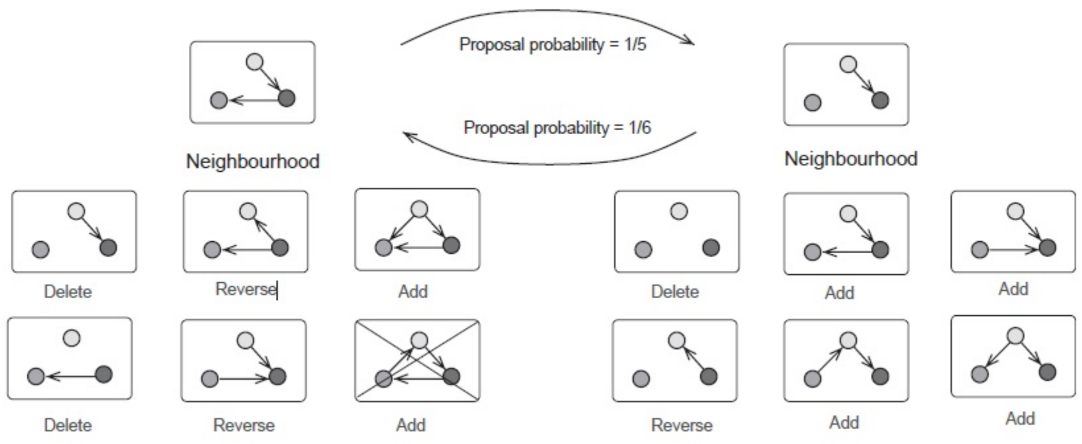


Figure 4: Neighbouring graphs. As we can see there are 5 possibilities from the original graph, since adding the remaining edge creates a cycle. In this case the edge deletion was chosen(with probability $\frac{1}{5}$). The new graph has 6 neighbouring graphs therefore the probability of reverting to the original graph is $\frac{1}{6}$

is 1 then an edje between nodes i and j exists, otherwise there is none. We can construct an aditional n by n matrix B [7], which contains our prior knowledge. Each entry $B_{ij} \in [0;1]$ represents our confidence that this node exists. Therefore

1. If $B_{ij} = 0.5$ then we do not possess any knowledge on this edge and are therefore completely uncertain.
2. If $B_{ij} > 0.5$ then we have some prior evidence that this edge exists.
3. If $B_{ij} < 0.5$ then we have some prior evidence that this edge does not exist.

It should be noted that even though these values lie in the unit interval, they are not probabilities. In order to turn them into probabilities in the stochastic sense we have to normalize them. This normalization will not be needed since we will only be comparing the different values, therefore the normalization procedure would be superflous.
With the definition of the source of our biological prior knowledge we are able to define the energy of a given network as:
$E(G) = \sum_{i,j=1}^{n} |B_{i,j} - G_{i,j}|$
It is easy to see that we'd have zero energy only when the graph and the biological prior knowledge are identical. On the other hand the higher the energy of a graph, the greater the mismatch with the biological prior knowledge. With this knowledge we can define the

probability of a given graph as: $P(G|\beta) = \frac{e^{-\beta E(G)}}{Z}$
Z is a normalizing factor and we can discard it as we will only be computing $\frac{P(G_{old}|\beta)}{P(G_{new}|\beta)}$ and Z cancels out.
β corresponds to the inverse temperature in statistical physics.The question that arises from the addition of a new parameter is how do we determine β? The way we do is we make a random walk for β as well. Our first idea would be to accept a new graph M_new and β_new with the following probability:
$A = min(1, \frac{P(D,G_{new},\beta_{new}Q(G_{old}|G_{new})R(\beta_{old}|\beta_{new}}{P(D,G_{old},\beta_{old}Q(G_{new}|G_{old})R(\beta_{new}|\beta_{old}})$
This however will lead to a much lower acceptance rate, since we have to consider both a new graph and new β.
That is why it would be simple to simply divide our random walks into two parts. One for our graphs and one for our β. Then the acceptance probabilities would be:
$A(G_{new}|G_{old}) = min(1, \frac{P(D|G_{new}P(G_{new}|\beta)Q(G_{old}|G_{new}}{P(D|G_{old}P(G_{old}|\beta)Q(G_{new}|G_{old}})$
and
$A(\beta_{old}|\beta_{new}) = min(1, \frac{P(G|\beta_{new})P(\beta_{new})R(\beta_{old}|\beta_{new}}{P(G|\beta_{new})P(\beta_{old})R(\beta_{new}|\beta_{old}}),$
but since we assume that β is uniformly distributed and we assume that $R(\beta_{new}|\beta_{old})$ is symmetric we can simplify it to:
$A(\beta_{old}|\beta_{new}) = min(1, \frac{P(G|\beta_{new})}{P(G|\beta_{old})})$
We assume that β is distributed over the interval [0;MAX]. Since we would not like any sudden radical shifts in β, we determine the new one accordingly $\beta_{new} \in [\beta_{old} - l; \beta_{old} + l]$. In this project the value of MAX was 20 and l was 3.
The other possibility is to use an exponential distribution. With the probability of a Graph $P(G) = \lambda e^{-\lambda n}$, where n is the number of edges in the graph. What this esentially does is that it says that we'd prefer graphs with less edges. The problem is deciding what hyperparameter λ to choose. In this case the optimal one was 0.1 and all results about the exponential distribution were with this hyperparameter.
The only other probability that we need is
$p(D|G) = \prod_{i=1}^{p} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk}+Nijk)}{\Gamma(Nijk)}$
Where:

1. N_{ijk} represents the number of observations in which D_i takes the value k
2. Parents(X) have the configuration j
3. r_i - the number of possible values of X_i
4. N'_{ijk} - Dirichlet Hyperparameters
5. $N_ij = \sum_{k=1}^{r_i} N_{ijk}$
6. $N'_ij = \sum_{k=1}^{r_i} N'_{ijk}$

In this case the Dirichlet Hyperparameters were set to 1. The equation is further simplified by the fact that we only have two possible states. Since the probabilities can get very low and we are faced with very small real numbers. This can be overcome by using multiple precision floating point numbers, or by taking the logarithm of this function. The latter approach is much more practical and leads to the advantage that we can replace the products by sums. This is why it was used.

2 Simulations

Simulations were done as follows. First we generate three different data sets. We generate one with 30, 70 and 100 samples respectively. The way the samples are generated is shown in [?]. Afterwards we try out the MCMC simulation with different sources of prior knowledge 5 times each on every data set and then average the results. The five different sources of prior information are as follow:

1. No Prior
2. With a semi-good prior
3. Good Prior
4. Exponential over the edge numbers prior

The semigood prior contains parts of the edges and contains a single wrong one. The Good prior is a perfect match with the generative model. There are many tests for convergence of MCMC simulations. For further reference see [5] and [?]. The approach from [7] was taken with slightly less steps with $2 \cdot 10^5$ steps with the first half being discarded as a burn in phase. This has led to satisfying results.

We shall use two criteria to evaluate the results presented by the different acceptance criteria for the MH algorithm. After a run and discarding the burn-in we evaluate the probability for each edge. The first criteria is the Mean Squared Error, which is calculated as follows:
$MSE = \sum_{i,j=1}^{n} |A_{ij} - P_{ij}|^2$
Where A is the adjancy matrix and P is the probability of each edge in this MCMC run. The next property, which is more important is the Receiver Operating Characteristic. In the ROC we have two axis, one for the specificity and one for sensetivity. They are defined as follows:
$Sensetivity = \frac{TP}{TP+FN}$
$Specificity = \frac{TN}{FP+TN}$
where TP stands for our True Positives, TN for the True Negatives, FP for the false positives, FN for the false negatives

```
while $i$<$Samples$ do
    Use topological sort
    for $Every\ Node\ j$ do
        if $If\ the\ node\ doesn't\ have\ parents$ then
            $P(X_{ij} = 1) = 0.5$
        end
        else
            OR states of parents. $P(X_{ij} = 1|ParOR = 1) = 0.8$,
            $P(X_{ij} = 1|ParOR = 0) = 0.2$
        end
    end
end
```

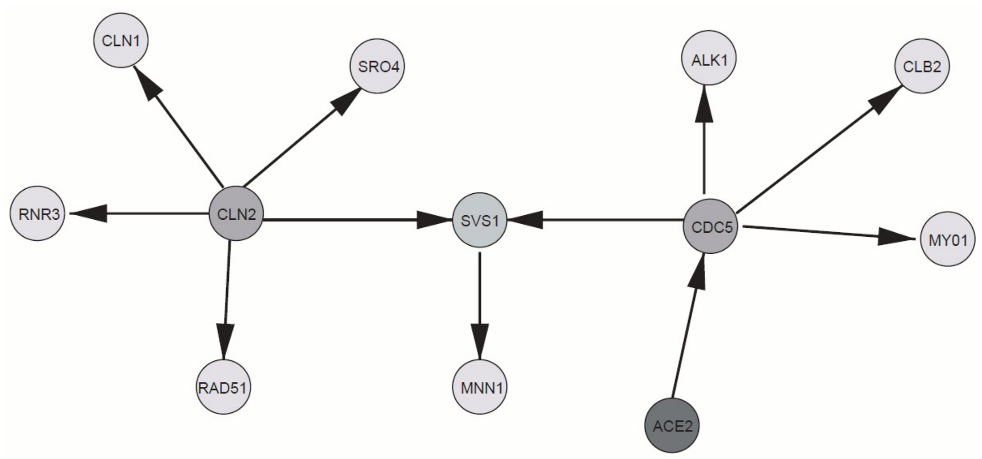


Figure 5: The model used to generate our data. It is a regulatory network in a yeast cell.

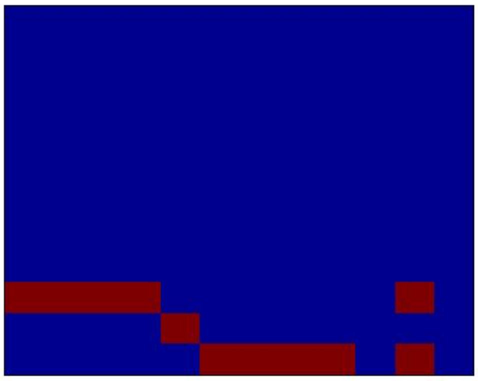


Figure 6: The adjacency matrix represented as a heat map

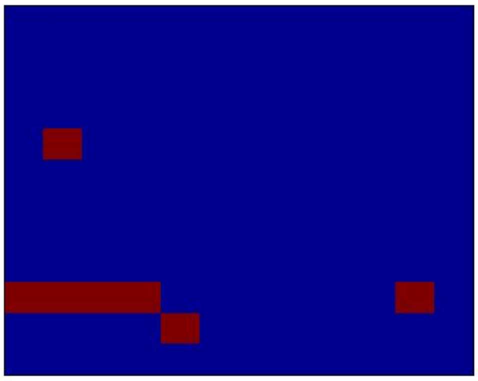


Figure 7: Semi-good model for our MCMC simulation. We are missing 5 edges and have a single wrong edge.

	30	70	100
no prior	10.8174	8.7318	8.1204
exp prior	8.2535	8.0328	7.3970
semi-good	6.6127	6.1015	5.6241
good	6.0572	4.5025	3.9965

Figure 8: MSE for the different priors/acceptance ratios

. But how do we determine the ROC for a certain set of edge probabilities. What we do is create a certain threshold upon which a certain edge is activated. Afterwards we count the number of TP,TN,FP,FN and calculate the specificity and sensetivity.

We can draw our conclusions based on the different MSEs and ROCs. We can see that there is a definite improvement over all of the priors with the increase of the number of samples. It is no surprise that having a good prior is the most accurate - but this mostly just serves as a complete confirmation of a theory. The most remarkable observation is how well the exponential prior performed. It is still somewhat worse than the semi-good prior, but its performance is nearly as efficient.

3 Bibliography

References

[1] Dirk Husmeier *Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks*, DOI: 10.1093/bioinformatics/btg313

[2] Daphne Koller, Stanford University Online Lecutre, *Probabilistic Graphical Models*, 2012

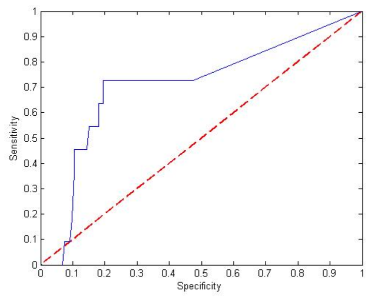
[3] Michiel W.W. Wels, *Unraveling the regulatory network of Lactobacillus plantarum WCFS1*

[4] Sach Mukherjee, Terence P. Speed, *Markov chain Monte Carlo for Structural Inference with Prior Information*

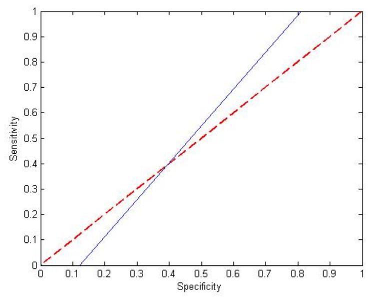
[5] Friedman,N. and Koller,D. (2003) Being Bayesian about network structure.

[6] Michiel W.W. Wels, *Unraveling the regulatory network of Lactobacillus plantarum WCFS1*

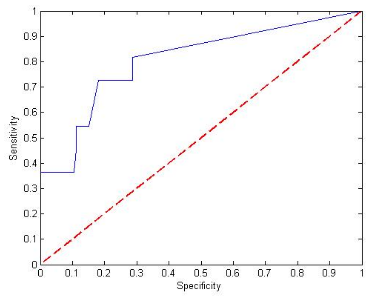
[7] Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge, Adriano V. Werhli, Dirk Husmeier



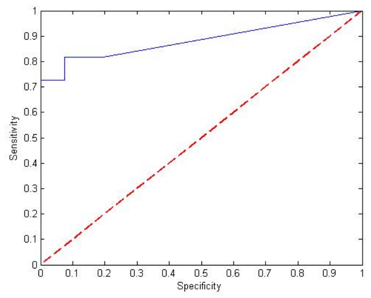
(a) No prior



(b) Exponential prior

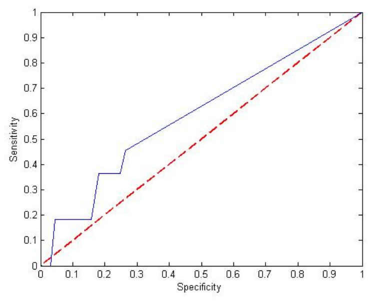


(c) Semi-good prior

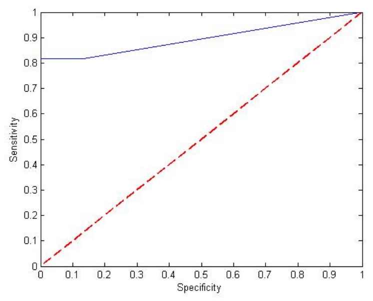


(d) Good prior

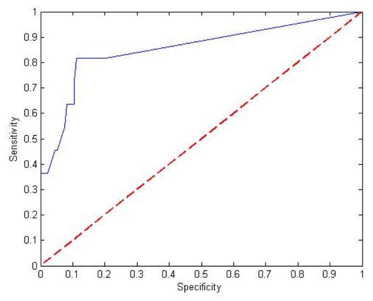
Figure 9: The ROC curves for 30 samples



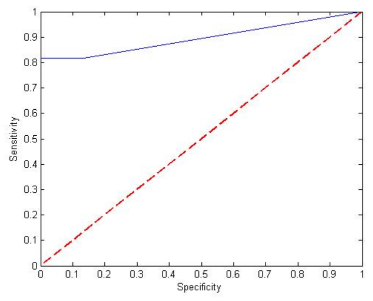
(a) No prior



(b) Exponential prior

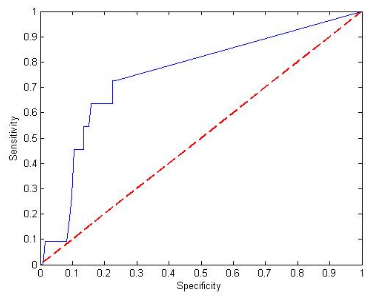


(c) Semi-good prior

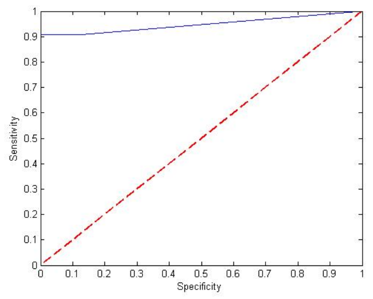


(d) Good prior

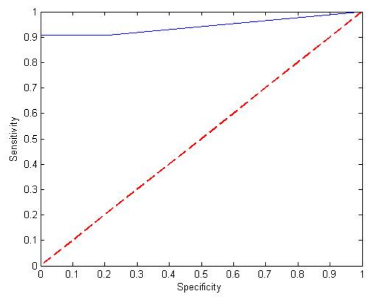
Figure 10: The ROC curves for 70 samples



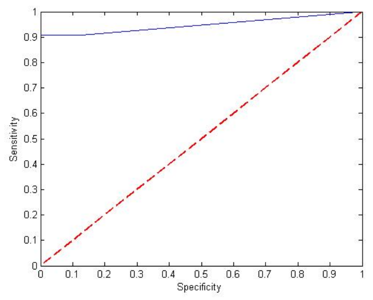
(a) No prior



(b) Exponential prior

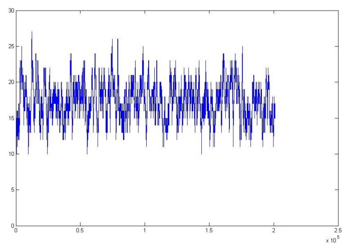


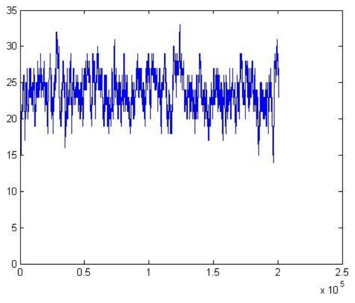
(c) Semi-good prior



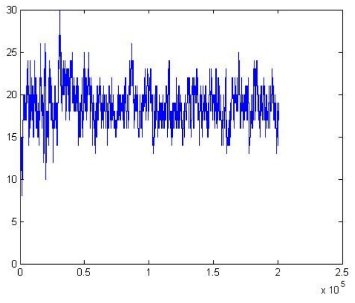
(d) Good prior

Figure 11: The ROC curves for 100 samples

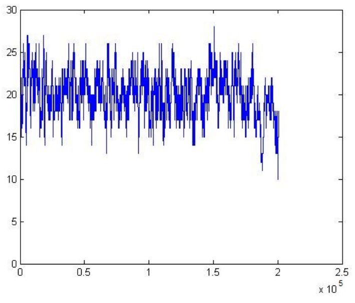

(a) No prior



(b) Exponential prior



(c) Semi-good prior



(d) Good prior

Figure 12: Edge Counts over the course of the MCMC simulation