HW6: Inference in Graphical Models

Manu Agarwal
UTEID: ma53767

Abstract. In this assignment, we study learning in graphical models. We consider a Bayesian network and assign probabilities to the edges in the network. We then make certain variables hidden and try to recover those values by propagating data through the network. We use the Expectation Maximization algorithm for reconstructing the probability distribution.

1 Introduction

A Bayesian network is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). It reflects the states of some part of a world that is being modeled and it describes how those states are related by probabilities.

Below is an example of a Bayes net. In this simple world, let us say the weather can have three states: sunny, cloudy, or rainy, also that the grass can be wet or dry, and that the sprinkler can be on or off. Now there are some causal links in this world. If it is rainy, then it will make the grass wet directly. But if it is sunny for a long time, that too can make the grass wet, indirectly, by causing us to turn on the sprinkler.

The joint probability distribution of the above network can be written as

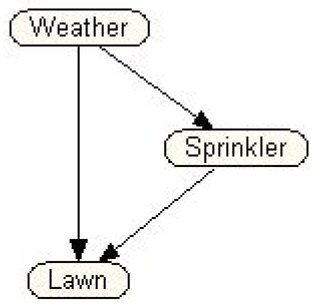

Fig. 1. An example of a Bayesian network

$P(W, L, S) = P(W)P(S|W)P(L|W, S)$. Thus, using a Bayesian net, we can model the joint distribution of all the variables using fewer parameters. This comes in handy when the number of parameters in a problem is large.

Since a Bayesian network is a complete model for the variables and their relationships, it can be used to answer probabilistic queries about them. For instance, we can use the network to find out updated knowledge of the state of a subset of variables when other variables (the evidence variables) are observed. This process of computing the posterior distribution of variables given evidence is called probabilistic inference. The posterior gives a universal sufficient statistic for detection applications, when one wants to choose values for the variable subset which minimize some expected loss function. A Bayesian network can thus be considered a mechanism for automatically applying Bayes' theorem to complex problems.

If we have certain data points corresponding to the values of all the variables, we can use it to infer parameters of our model that maximize the likelihood of the observed data. Such an inference procedure is called Maximum Likelihood Estimate (MLE). Expectation Maximization (EM) algorithm is one such algorithm to compute such parameters. It consists of two steps:

1. Expectation step - In this step, we calculate the expected value of the log likelihood function given the values of the observed variables. In other words, we try to infer the distribution over the hidden variables. We use belief propagation (sum-product algorithm) for this task.
2. Maximization step - In this step, we update the parameters using the observed values and the distribution.

We repeat the above two steps until convergence.

2 Metrics

We need to evaluate how good the inferred probability distribution is with respect to the original distribution. We use Kullback-Leibler divergence for this. KL-divergence is a measure of the difference between two probability distributions P and Q. It is not symmetric in P and Q. In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while Q typically represents a theory, model, description, or approximation of P. For discrete probability distributions P and Q, the KullbackLeibler divergence of Q from P is defined to be

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\, \log \frac{P(i)}{Q(i)}$$

3 Network

We use the Bayesian Network in Figure 2 for the purposes of this assignment. It represents a simple Bayesian network that represents the probability of an alarm ringing given the probability of two events - burglary and earthquake. It also represents the probability of John and Mary calling given the probability of alarm ringing.

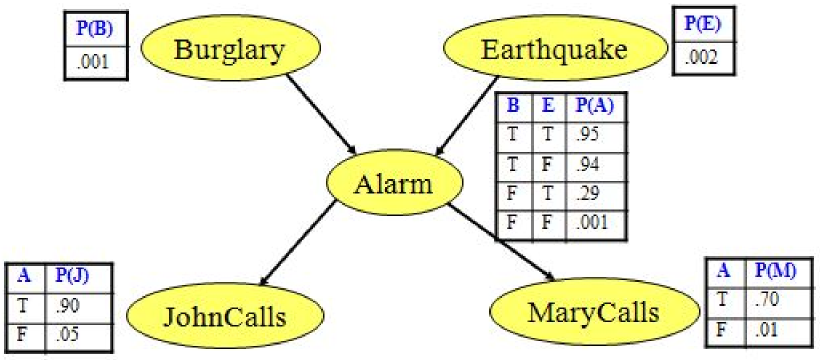

Fig. 2. Bayesian network used for this experiment

4 Algorithm-The basic pipeline

For implementation of 1, we used the open source Bayesian network toolbox by Kevin Murphy. It provides direct implementations for belief-propagation algorithms, expectation maximization etc. We begin by building the Bayesian network depicted by Figure 2 where each node is a binary variable. We convert the network to a Directed Acyclic Graph (DAG). We then create a dataset for this DAG using sample_bnet method provided by the toolbox. Using the learn_params method of the toolbox, we predict MLE estimates for the dataset. We then randomly hide the values of certain variables and run the EM algorithm on the partial dataset through the learn_params_em method. Finally we compute the KL divergence for both the MLE and EM estimates of the probability distributions with the original distribution.

5 Experiments

To experiment with the algorithm, we hide the value of node E in some of the examples. We then compute MLE and EM probability distribution estimates. Finally, we check the quality of the estimates by computing the KL divergence between the estimated and the actual probability distributions.

5.1 MLE vs EM

Figure 3 shows the performance of MLE and EM estimates. Here, we used 10000 samples for MLE estimate and hide the value of node E in 5000 of these samples. MLE estimate outperforms EM slightly as we use full data for MLE thus giving a more accurate estimate.

Algorithm 1 Inference in graphical models

```
Input: A Bayesian Network along with certain observed values of variables
Output: Values of the hidden variables
 1: Generate data samples using the sample_bnet method in the Bayes Net toolbox of
    MATLAB
 2: Hide the values of one of the variables in a predefined proportion of samples
 3: Initialize the potential corresponding to every node $\phi_i(X_s)$ to 1 and the Expected
    Sufficient Statistics $ESS_i(X_s)$ for each potential to zero
 4: For every sample $S_j$ in the sample set $S$, do the following:
 5: If the sample is fully observed, increase the value of the entry it indexes in $ESS(X_s)$
    by 1
 6: If the sample is not fully observed, then follow steps 7-10
 7: Using sample $S_j$ as evidence, execute the sum-product algorithm on the model
    using the current value of the potentials as its parameters
 8: Obtain marginal distribution over hidden variables in the sample
 9: Reshape marginal distribution to match the dimensions of $ESS_i(X_s)$ by filling new
    entries with zero
10: Add reshaped marginal to $ESS(X_s)$
11: Normalize $ESS(X_s)$ and assign it to $\phi_i(X_s)$ and then reset $ESS(X_s)$ to zero
12: See if the convergence criterion is met. If not, return to step 4
```

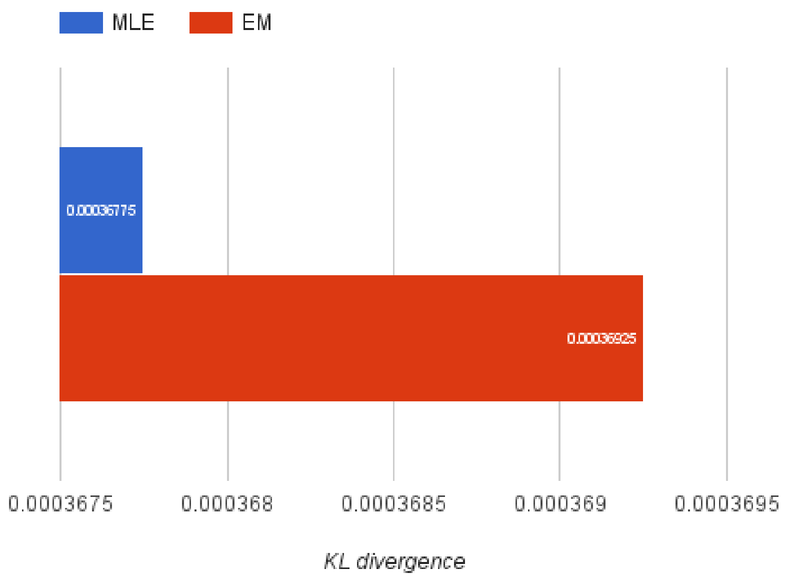

Fig. 3. Performance of MLE and EM estimates

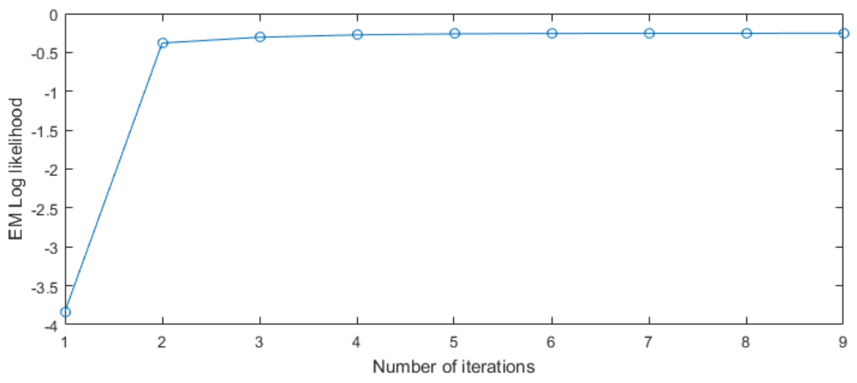

Fig. 4. Convergence of the EM algorithm

5.2 Convergence of the EM algorithm

Figure 4 shows the convergence of the EM algorithm. The EM algorithm converges after 2 iterations as our graph is a DAG.

5.3 Performance of the EM algorithm as the amount of unobserved data is increased

We then studied the performance of the EM algorithm as the amount of unobserved data is increased. We kept the number of samples constant at 5000 and varied the percentage of hidden values from 10% to 90%. As observed in Figure 5, the KL divergence between the estimated probability distribution and the original distribution increases as it becomes harder to reconstruct the original distribution as the amount of hidden values increases.

5.4 Performance of the EM algorithm as the number of training samples is increased

We also studied the performance of the EM algorithm as the number of training samples is increased. Figure 6 shows that the KL divergence decreases as the number of training samples is increased. In this experiment, we kept the proportion of hidden values at 0.5 and increased the number of training samples from 1000 to 10000.

5.5 Impact of the choice of probability distribution

Finally, we studied the impact of the choice of probability distribution on the performance of the EM algorithm. A uniform probability distribution has all the conditional probability values equal while a skewed distribution has highly skewed values. Figure 7 suggests that it is easier to learn the values of a skewed distribution as compared to a uniform distribution.

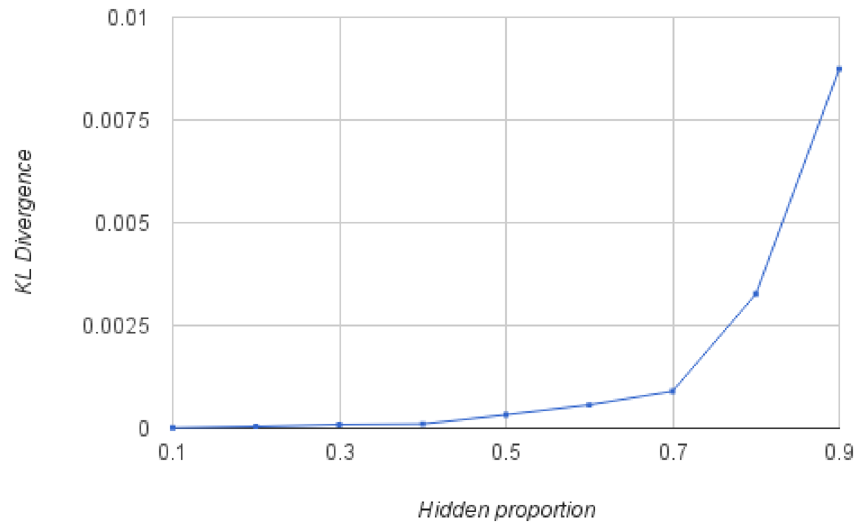

Fig. 5. Performance of the EM algorithm as the proportion of unobserved data is varied

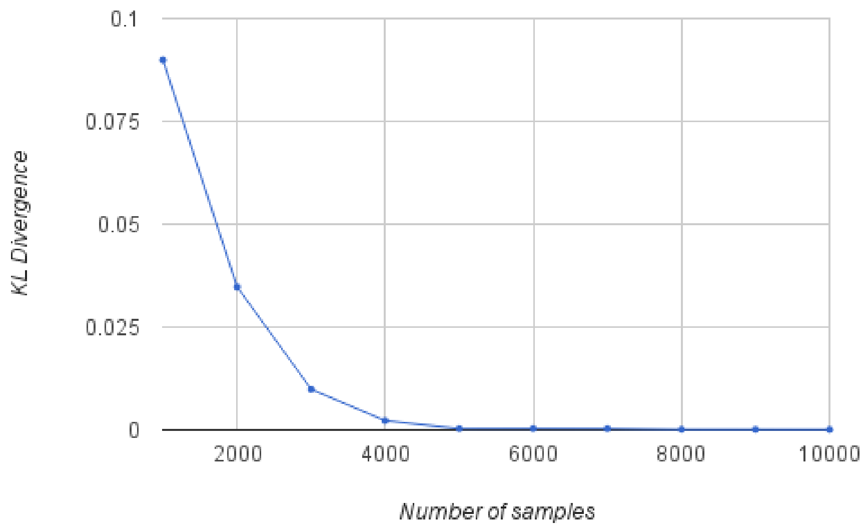

Fig. 6. Performance of the EM algorithm as the number of training examples increase

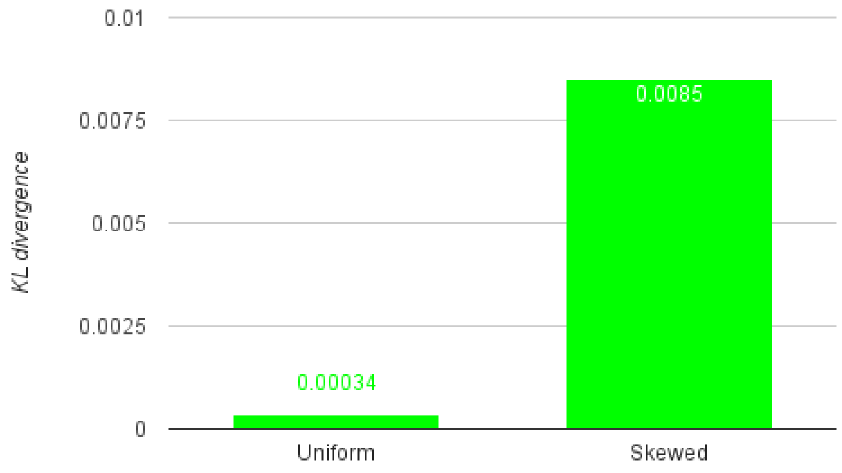

Fig. 7. Impact of the choice of probability distribution on the performance of the algorithm

6 Conclusion

In this assignment, we explored inference in a graphical model using belief propagation. We saw that MLE outperforms EM as MLE uses full data to estimate the parameters as opposed to EM which uses partial data. If there are sufficient number of training samples, EM converges to the original data. The quality of the recovered data (or **the stability of the EM algorithm**) is dependent on the number of training samples, the proportion of hidden samples and the probability distribution being estimated.

7 References

1. Pattern Recognition and Machine Learning by Christopher Bishop
2. Resources on the Machine Learning course homepage
   http://www.cs.utexas.edu/ dana/MLClass/446outline.html
3. Wikipedia