MCMC: Learning Hyperparameters

Sterling Suggs

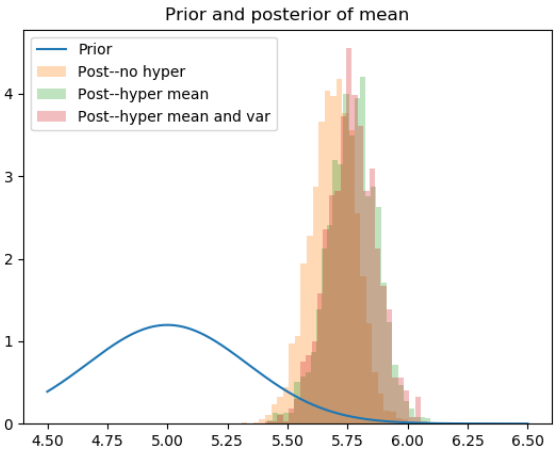
June 11, 2019

1 Introduction

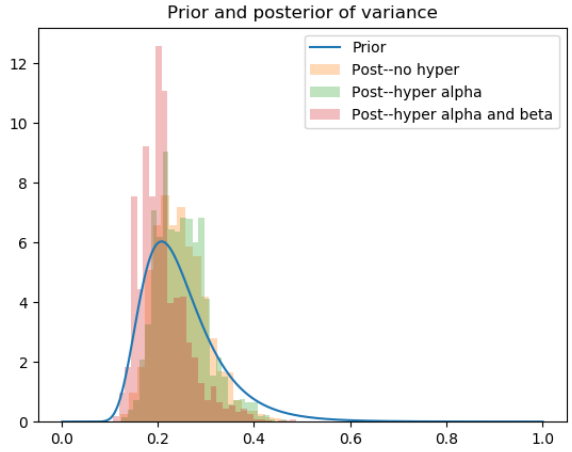
In this lab we experimented with learning hyperparameters by Gibbs sampling in our Bayesian networks.

2 Faculty data

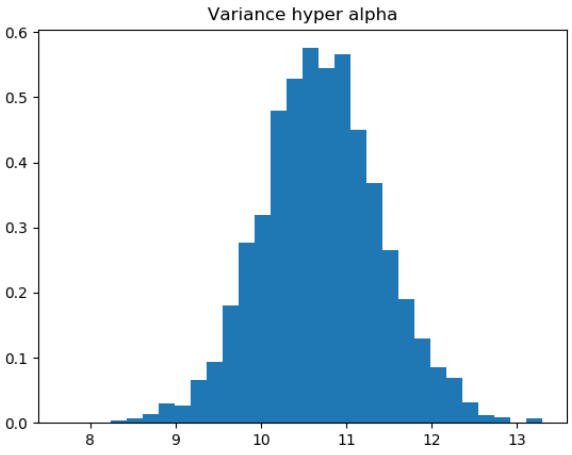
Our first task was to add hyperparameters to the faculty network and see how this changed what the network learned. We first added a hyper-mean for the mean node, followed by a hyper-variance. We modeled the hyper-mean as a Normal distribution and the hyper-variance as inverse Gamma. We plotted the posterior of the mean with the addition of each hyperparameter, comparing it to the case with no hyperparameters. Notice that adding a hyper-mean allows the posterior distribution shift further over. Adding the hyper-variance after that has little effect in this case.

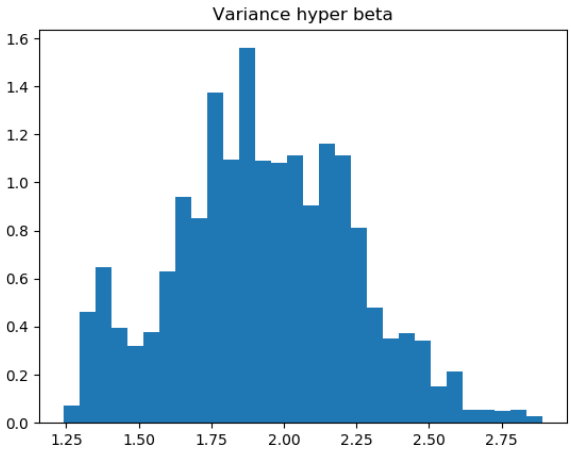


Then we added hyperparameters for the variance node. The shape parameter was modeled with a Normal distribution and we modeled the scale parameter with a Gamma distribution. This time, adding the hyperparameters allows the posterior distribution to backtrack towards the left as it grows narrower and taller, indicating decreasing variance.

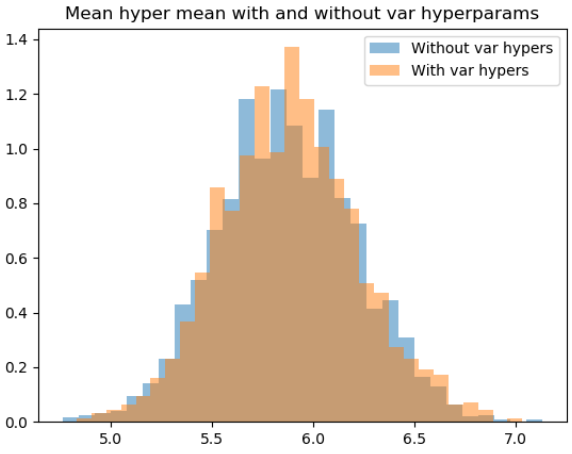


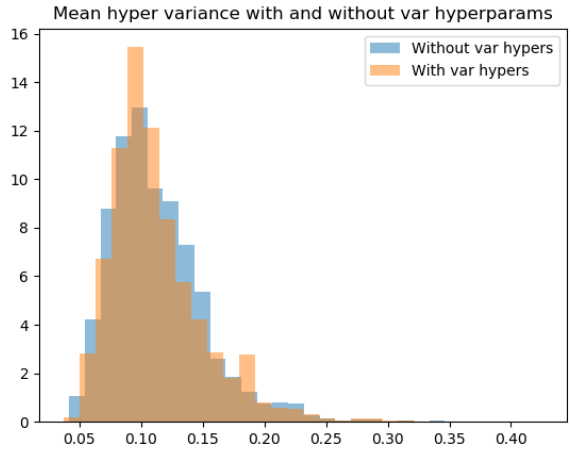
We can also view the posterior distributions for the variance hyperparameters α and β:



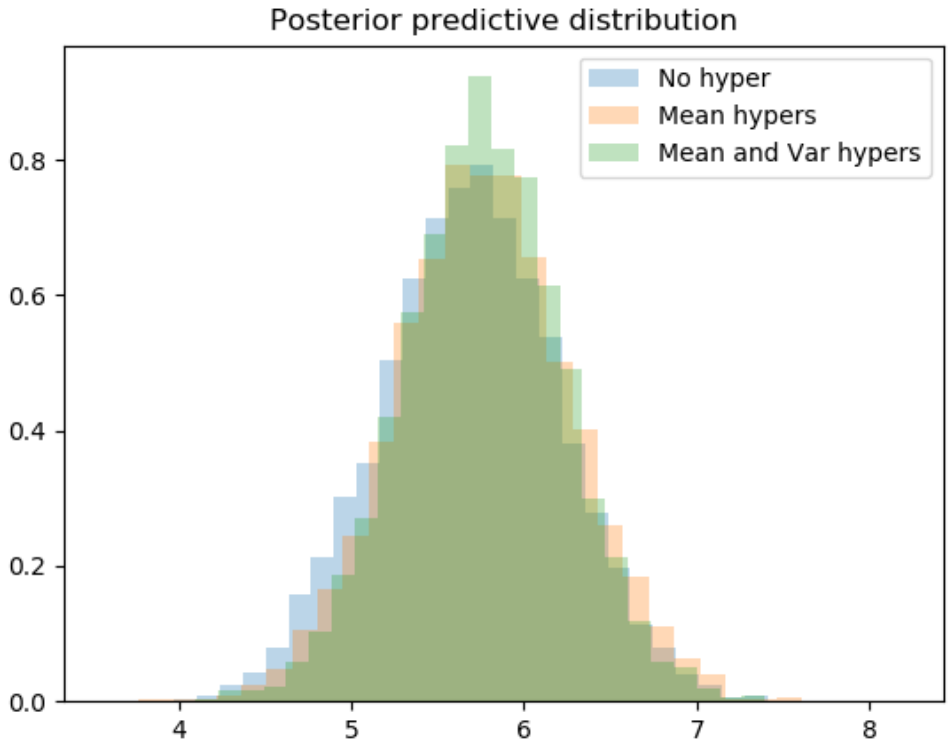


For the mean's posterior hyperparameters, we can compare the distributions before adding variance hyperparamters to the distributions after. We don't observe much difference.



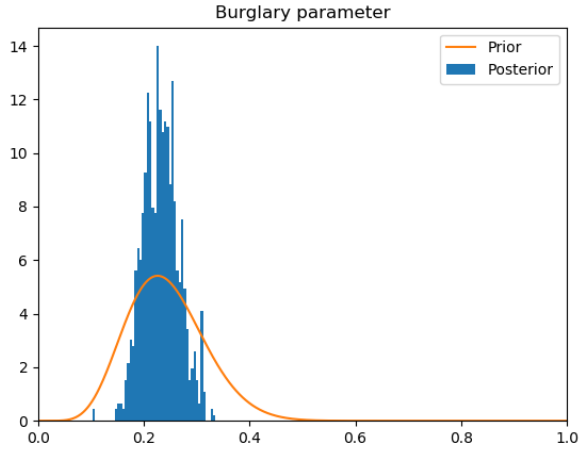


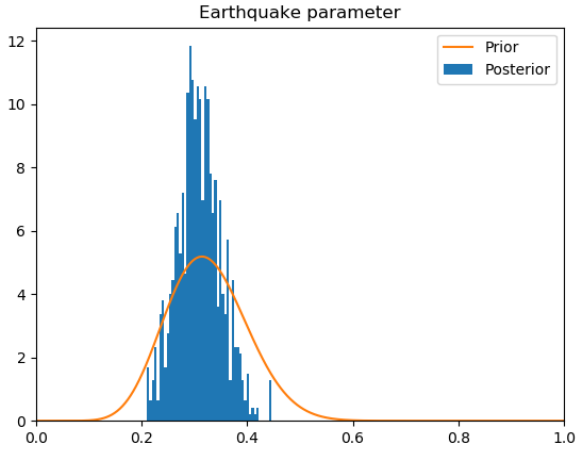
Finally, we show the posterior predictive distribution with no hyperparameters, with hyperparameters on the mean, and with all hyperparameters. We can see the distribution shift slightly to the right, and then grow a little narrower when the variance hyperparameters are added.



3 Alarm network

We returned to the alarm network and added hyperparameters a few at a time. First we added hyperparameters on the Burglary and Earthquake nodes, modeling the parameter of each Bernoulli draw with the output of a Beta distribution. Initial parameters for each Beta node were chosen by hand. The following two plots show the prior and posterior distributions for the parameter for the Burglary and Earthquake nodes. Note that the true parameter of Burglary was 0.2 and the true parameter for Earthquake was 0.3. We trained on 100 observations for 1000 iterations.

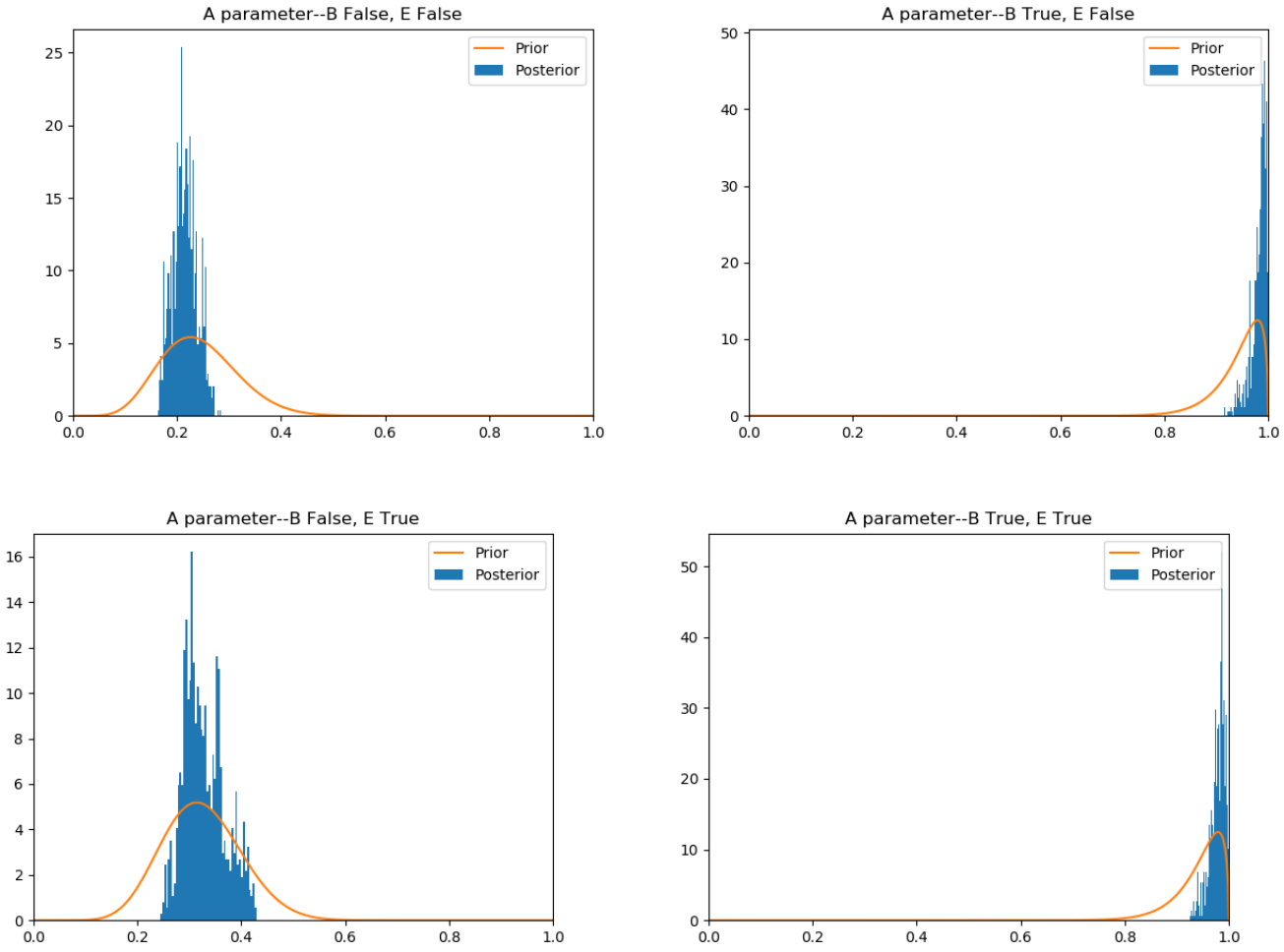




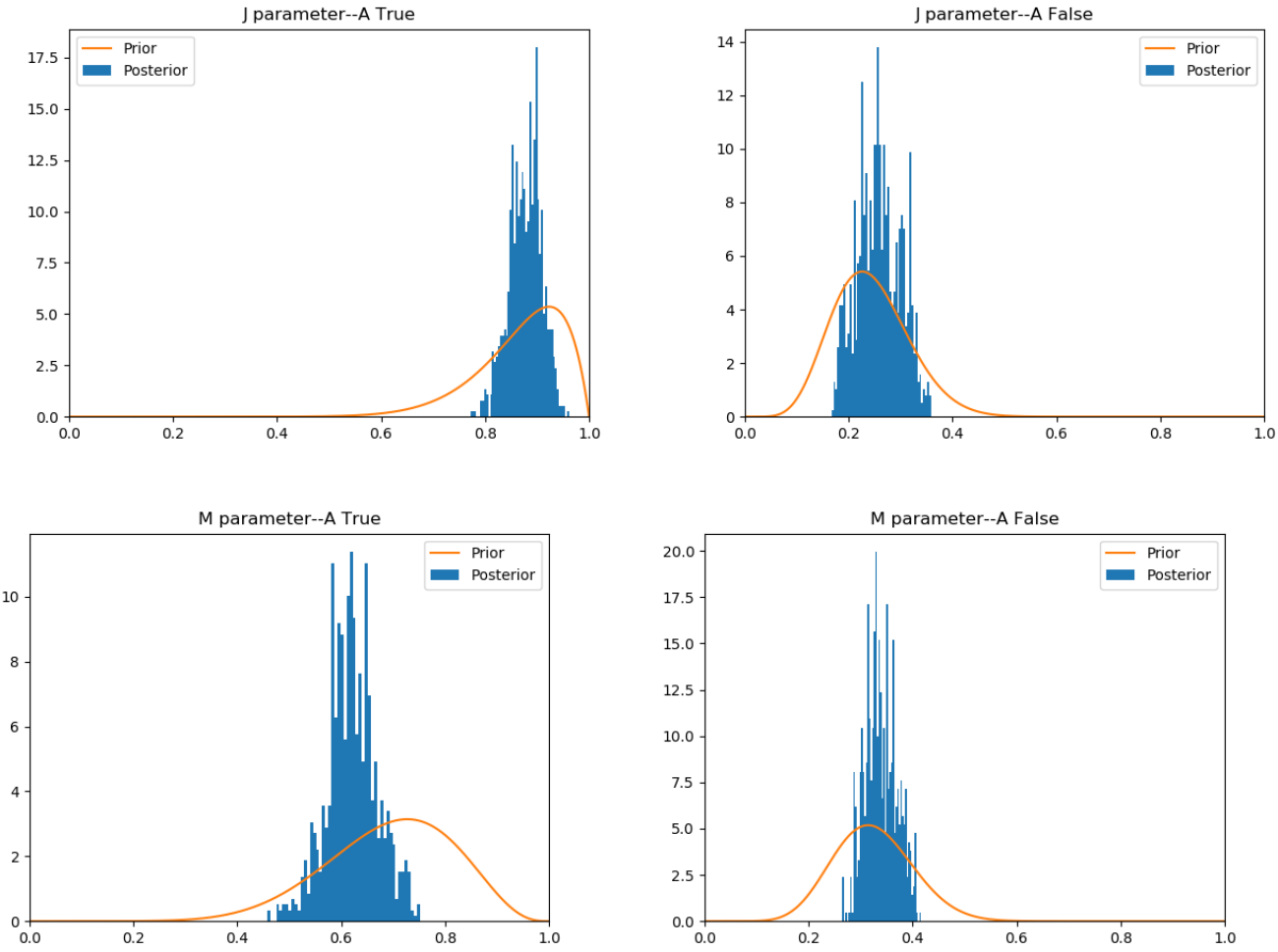
Next we added Beta priors to each case of the Alarm node. The true parameters are as follows:

B	E	True p
F	F	0.2
T	F	0.94
F	T	0.29
T	T	0.95

Again we trained on the same dataset of 100 observations. These plots show the distribution of each Alarm hyper-parameter before and after training.

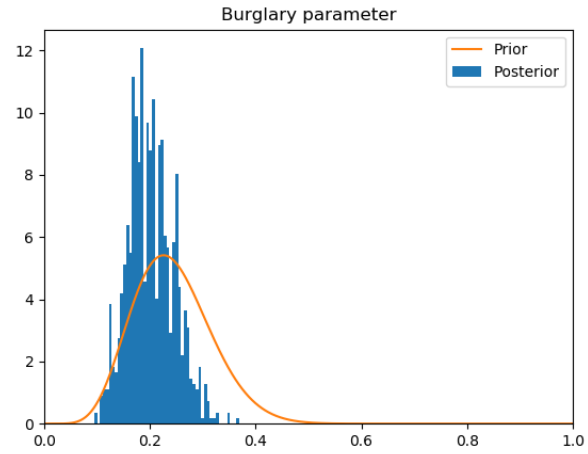


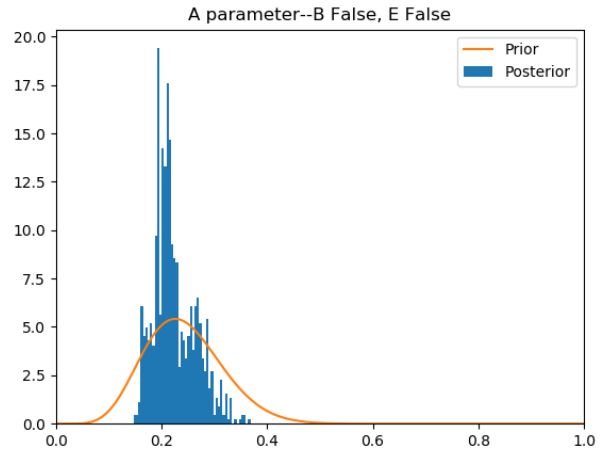
Finally we added hyper-nodes on John and Mary as well. For John, the true parameters are 0.9 if Alarm is True and 0.2 if it's False. For Mary, the true parameters are 0.7 if Alarm is True and 0.3 if it's False.

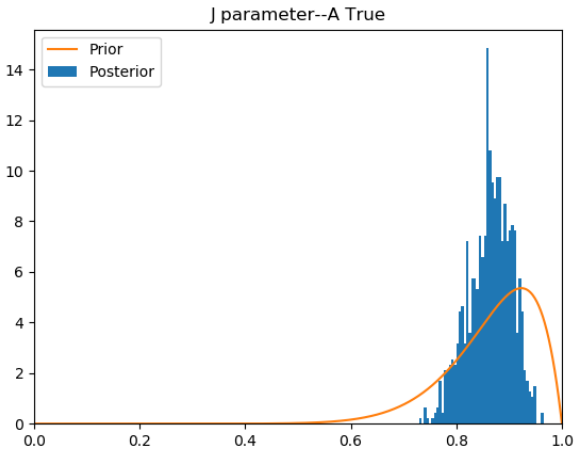


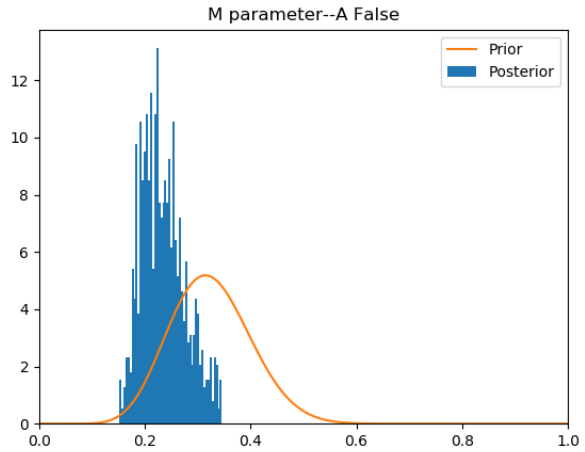
3.1 Amount of Data

We relearned the parameters with different amounts of data, first with 50 observations instead of 100. As expected, less data results in less learning. To show this, we show example parameter distributions from Burglary, Alarm, John, and Mary nodes. We observe that each distribution is wider than the corresponding one in the previous section, showing less certainty about the value of the parameter. Note that the exact location of each distribution may change slightly because we used a new dataset and when there's only 50 observations, a couple more earthquakes is enough to change things.

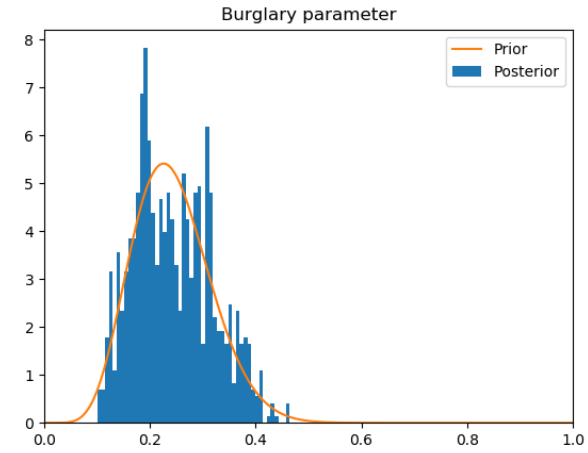


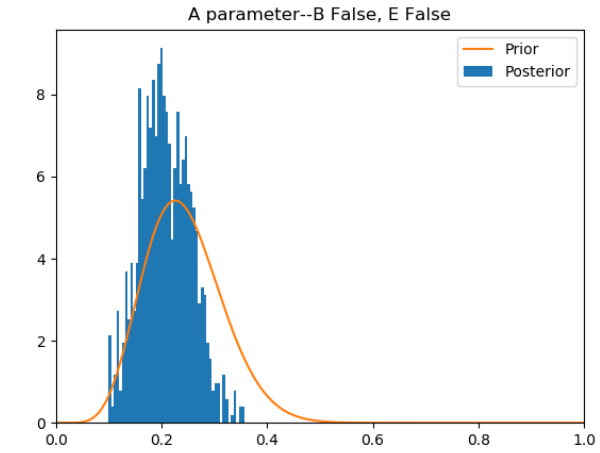


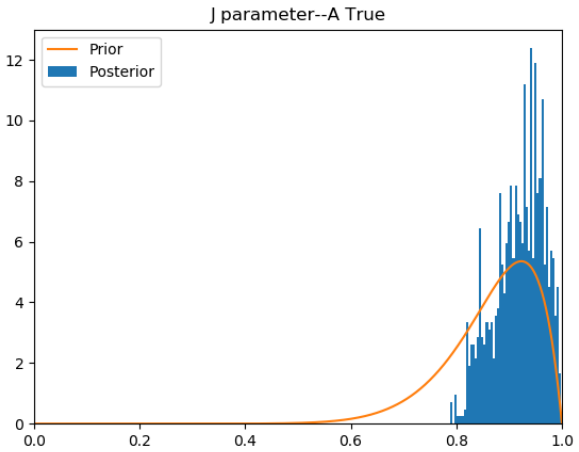


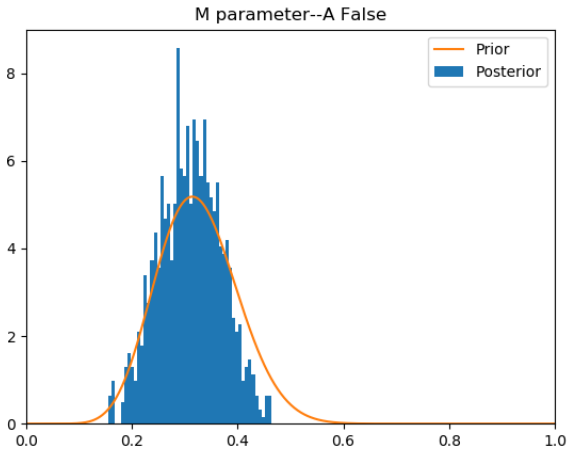


Finally we tried with 25 observations. As the plots show, learning was even worse.



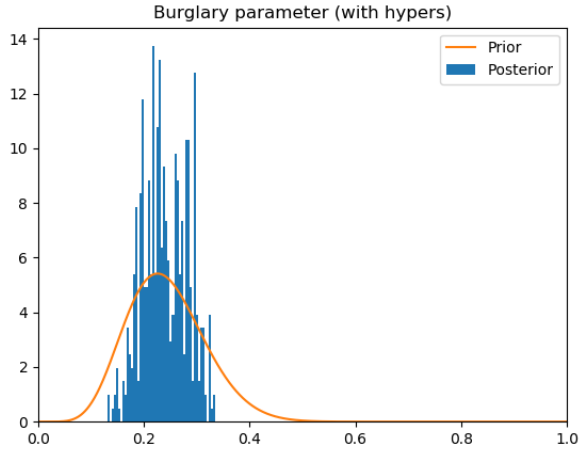


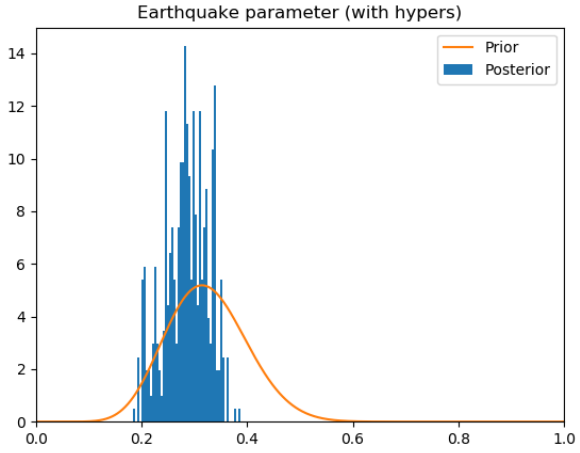




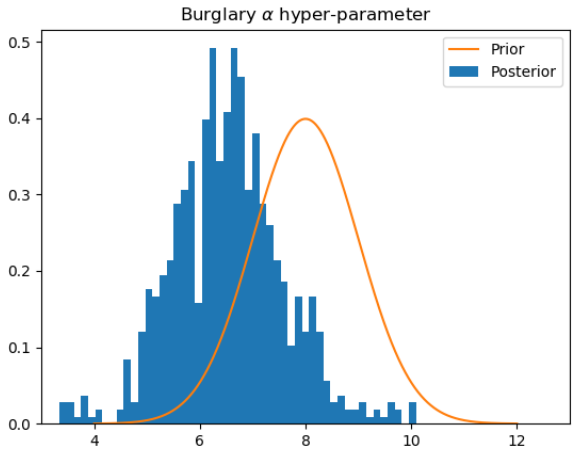
3.2 Hyper-hyperparameters

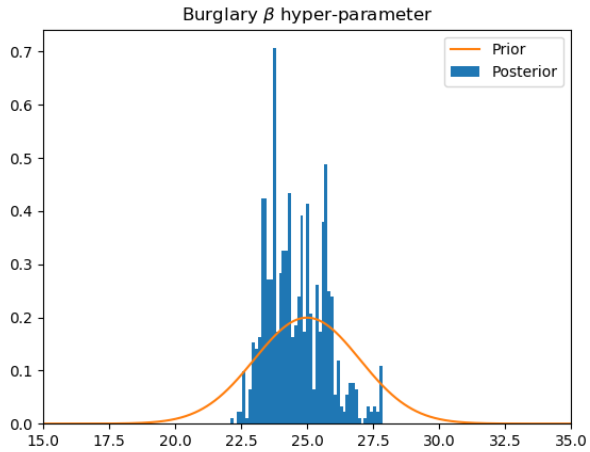
We added hyperparameters to the parameters of the Burglary and Earthquake nodes to observe how it affected learning. The most noticeable difference was that it was much harder to tune the mixing parameters. Nevertheless, we saw the distributions shift to match the data. After training for 1000 iterations on the original 100-observation dataset, both Earthquake and Burglary parameter distributions are narrower than in the original graphs.

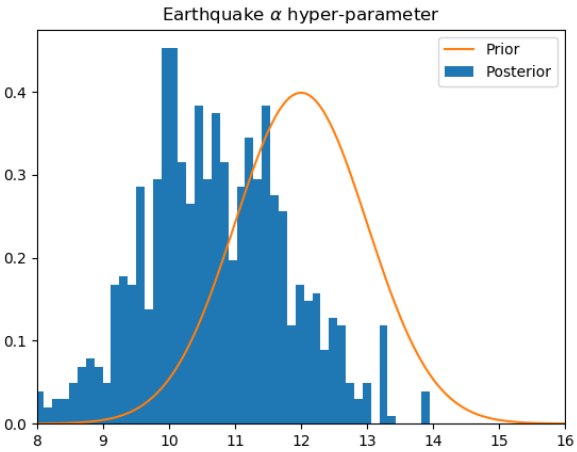


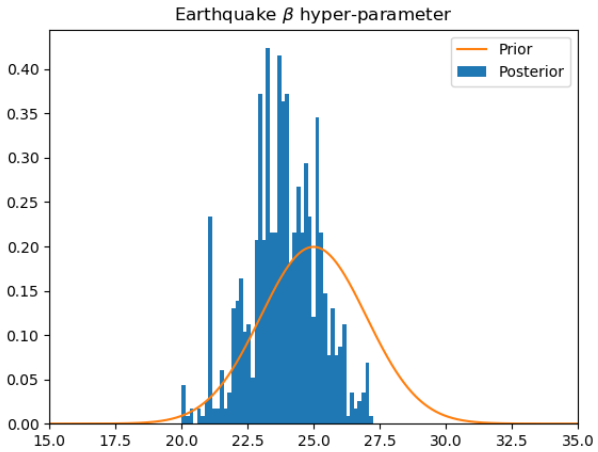


We also show how the distribution of the hyper-hypers changed with training. As Beta distributions, the Burglary and Earthquake parameter nodes each have an α and a β parameter to describe their shapes. These plots show the distribution of those parameters.



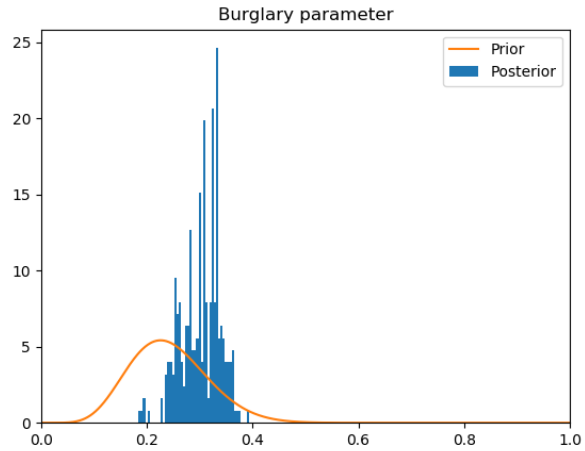






3.3 Inference

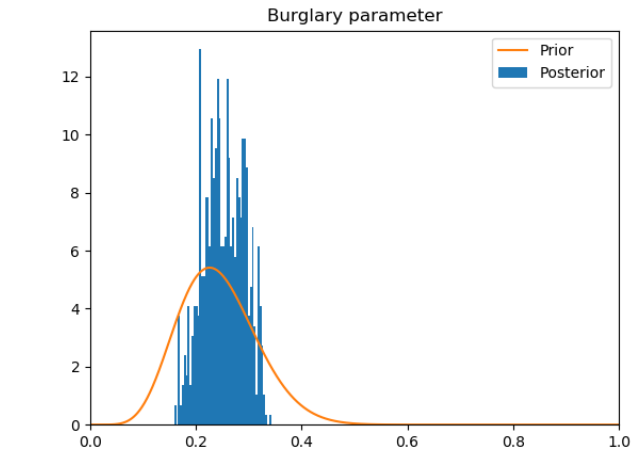
We can do inference with our sampler. Given our 100-observation dataset, we assumed Mary was equal to True and sampled the following distribution for Burglary. From this plot we estimate that the probability of an burglary given that Mary called is about equal to 0.3

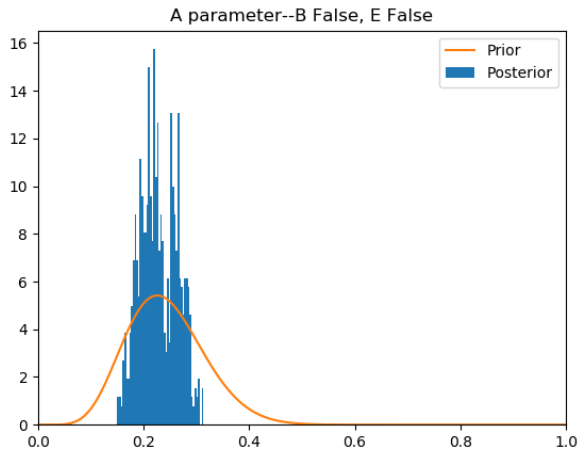


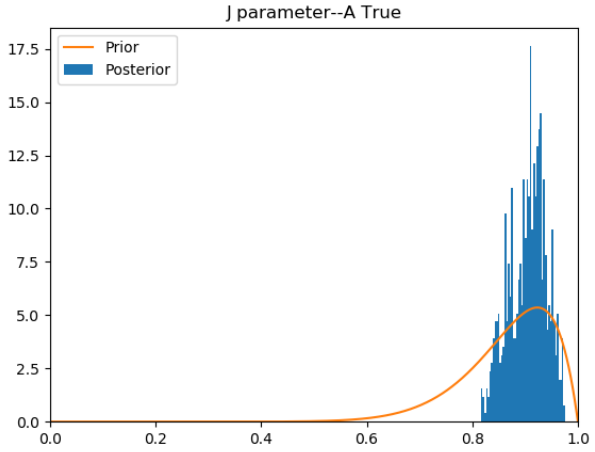
3.4 Missing data

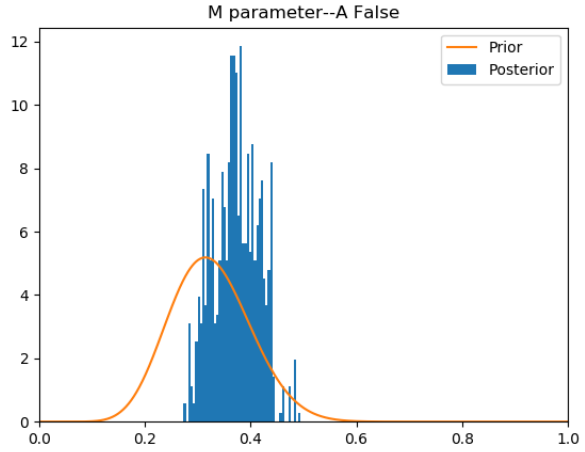
We randomly removed 15% of the values in our data (using the dataset with 100 observations). That is, observations looked something like [False, None, True, True, None].

Our network could still get a very good idea of the parameters, as we see in these representative plots. The distributions begin to widen slightly compared to the original distributions at the beginning of Section 3, but overall learning was still very good.





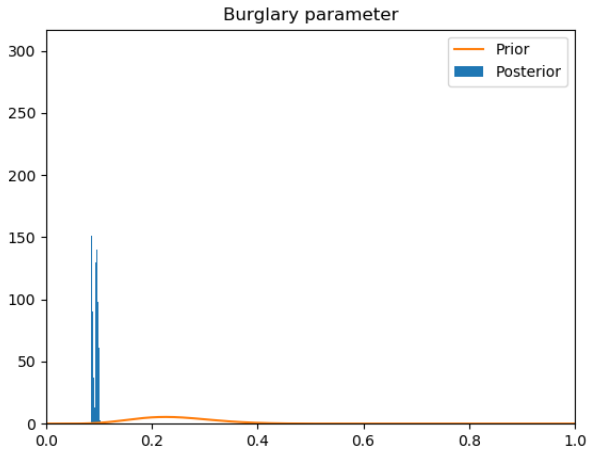


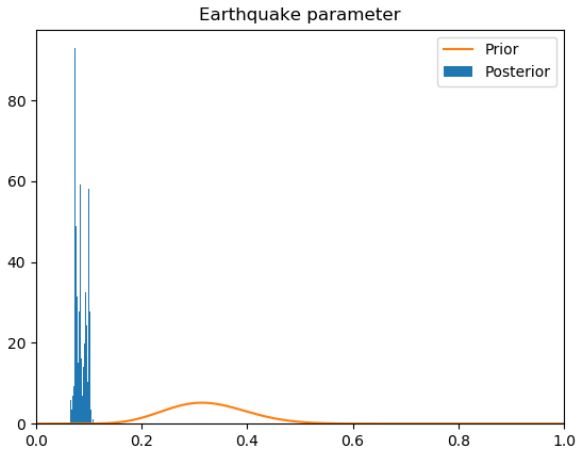


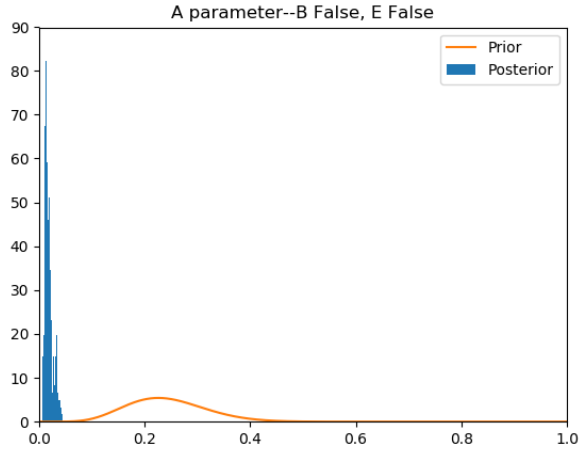
3.5 Original parameters

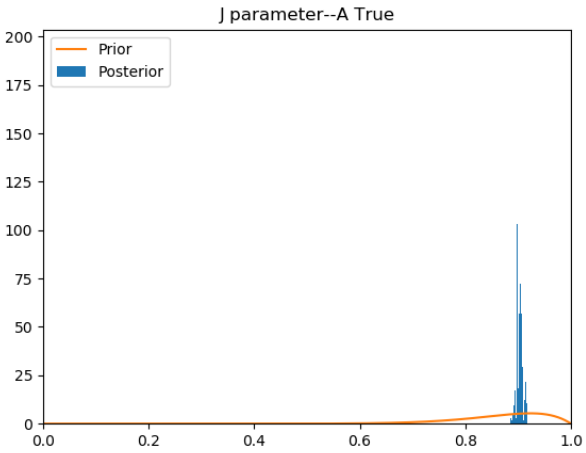
It was much harder to learn with the original parameters because the events Burglary and Earthquake are so unlikely to happen. In our generated dataset of 100 observations, we had one earthquake and no burglaries. Consequently, the alarm only went off once and the neighbors called only a couple of times.

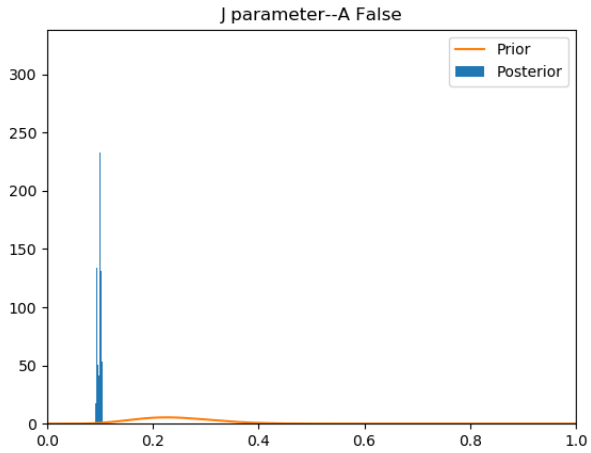
We include a few posterior plots for examination. It was hard to get good samples; since basically everything was false always, everything was very unlikely. In general, it is hard to learn about events that never happen, and even harder to learn about the consequences of those events. However, the distributions that we did learn have very low variance since there is almost no uncertainty in the observations.









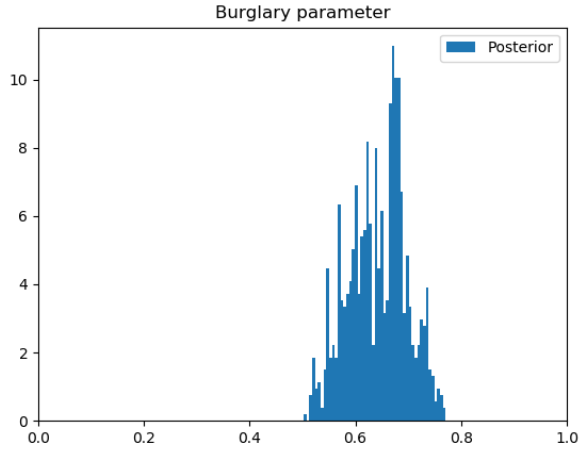


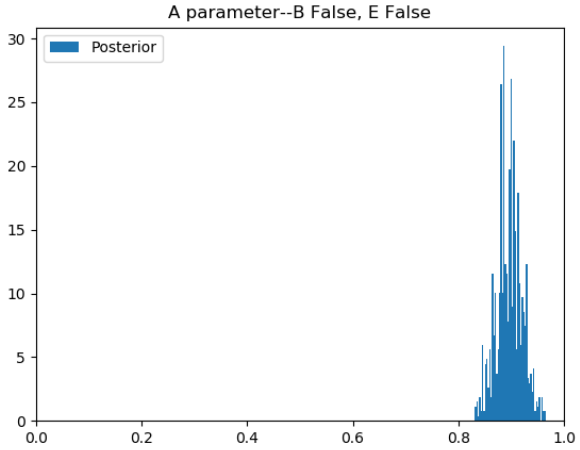
3.6 Secret parameters

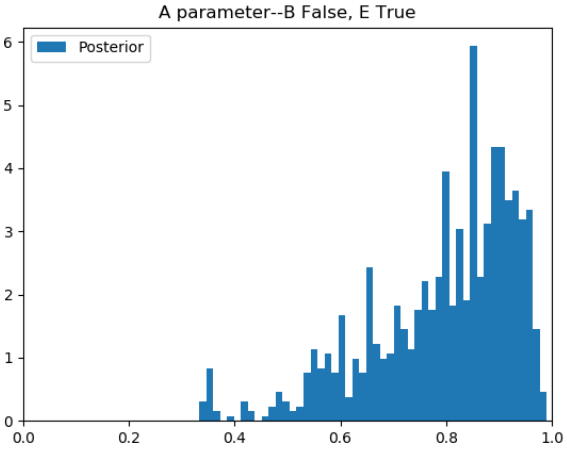
We changed the parameters of the Burglary and Alarm nodes and generated data, with 15% of values missing. The Burglary parameter is 0.6, and the Alarm parameters depend on Burglary and Earthquake in accordance with this table:

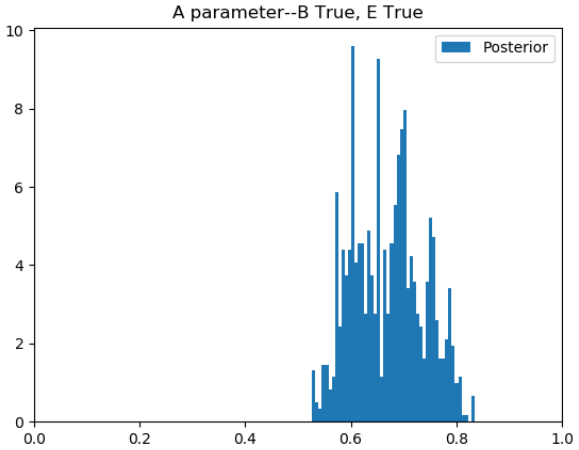
B	E	True p
F	F	0.1
T	F	0.5
F	T	0.7
T	T	0.3

After posting our generated data on learning suite, we applied our sampler to the data posted by Michael Richards. In each hyper-parameter node, we set α and β each to 1, so that our prior would be uniform since we had no idea what the true values were. We got these posterior distributions for the Burglary and Alarm parameters.









From the first plot we estimate the Burglary parameter to be about 0.65. For some reason, our Alarm parameters had a hard time when Earthquake was True, but we estimate the Alarm parameters to be the following:

B	E	True p
F	F	0.9
T	F	0.99
F	T	0.7
T	T	0.8

4 Conclusion

We added the ability to learn hyperparameters in our Gibbs sampler. Really this did not require much work as it's simply a matter of adding extra nodes to the network. After more than a fair amount of debugging we were pleased to see that we could learn the parameters to arbitrary data, even with a significant amount of missing values.