Inference in a Connectionist Model

Kevin Leung (kkleung@stanford.edu)

Introduction

Over the past decade, Bayesian modeling has become a popular approach in studying cognition. Intended as a computational level description, it assumes that people are ideal learners that develop hypotheses based on observed data. Bayesian models treat people as ideal observers who have a probability distribution over hypotheses from observed data, considering both the prior probability of a hypothesis and the likelihood of the data given a hypothesis. So far, this approach has been applied to many different phenomena, including word learning [1], causal induction [2], and more.

Computing the probability of hypotheses given the evidence requires inference, and given the complexity of the space, modelers often use approximate inference algorithms, including Markov Chain Monte Carlo (MCMC) methods such as Metropolis-Hastings [3]. Because these models are intended only as computation (not algorithmic) level explanations, the method of inference isn't considered relevant to the validity of the model. Even so, a common criticism of these models is that probabilistic inference as implemented isn't biologically plausible, and although they may match behavioral data, they are no more useful in understanding how people actually think.

Another approach to computational modeling is connectionist modeling, which use neural networks to explain various phenomenon. Instead of functioning at a purely computational level, these models depend on the algorithmic level details of the model, such as the activation function and weights between units. The networks are also inspired by the structure of the brain with units (like neurons) receiving incoming connections from other units.

Recently, attention has been drawn to the opposing perspectives and claims of these two types of models. One way of potentially crossing this divide and dealing with criticisms of Bayesian models is to implement inference in a connectionist framework. Instead of using existing algorithms, we hope to construct a network capable of doing probabilistic inference in a more plausible manner. By observing the state of the network over time, we can take samples from some distribution generated by the network dynamics. In addition to representing an arbitrary joint distribution, the model should also be capable of representing all necessary conditional probabilities as well, making it fully generative and capable of doing inference with varying evidence.

Related works

There have been many attempts to demonstrate the plausibility of probabilistic inference in the brain. Recently, several phenomena have been explained as algorithmic-level effects similar to common methods of inference, such as multi-stable perception as MCMC [4] and sentence parsing as particle filtering [5]. Although they provide insights into particular aspects of cognition, they don't provide a general framework for how any type of inference can be implemented.

Other models have attempted to show how inference can be done using connectionist networks. Bayesian inference can be done in a connectionist model, but only with idealized nodes in a feed-forward manner [6]. This depends upon the specific activation function for these neurons and is a clever implementation of the exact math for Bayes theorem. Notably, this model is feed-forward and cannot reason backwards from all evidence, meaning that it still isn't fully

generative. Bayesian inference can also be done in a network using population codes [7], but this has the same limitations as the model above, being only feed-forward.

Perhaps the most convincing method of inference is the Restricted Boltzmann Machine (RBM). It is both a connectionist-style network and capable of full probabilistic inference by clamping any set of inputs. Even so, it relies on very specific constraints to the model (2 fully connected layers with symmetric weights), and its calculations reduce to a form of Gibbs sampling [8].

Methods

Model structure

For this model, we decided to use an integrate-and-fire network to represent the data. Although a sigmoid activation function is more standard and has more tractable properties, we wanted the model to be sufficiently different from the design of a RBM. Additionally, integrate-and-fire is also more biologically plausible, which is a goal of this model

The network is represented as a fully connected graph of "units." Of these units, a subset are observed as the variables of interest. The rest are hidden and are necessary only as part of the network dynamics.

Intuitively, the behavior of any unit is accumulation to a threshold. Whenever another unit fires, it either excites or inhibits the unit. When a unit's activation goes over the threshold, it fires and has its own activation set back to 0. More concretely, let w_{ji} be the weight of a connection from unit i to unit j. The weights can be both positive and negative, and its magnitude reflects the influence one unit has on another. Each unit i has an activation a_i^t and is firing f_i^t at time t. Additionally, each unit i may receive external input b_i with probability x_i. The network can then be described by the following equations:

$$a_i^{t+1} = \begin{cases} a_i^t + \sum_j f_j^t w_{ij} + b_i p(x_i) & : a_i^t < 1 \\ \sum_j f_j^t w_{ij} & : otherwise \end{cases}$$

$$f_i^{t+1} = \begin{cases} 0 & : a_i^t < 1 \\ 1 & : otherwise \end{cases}$$

There are several changes we made from standard integrate-and-fire models. First, we did not include a refractory period to ensure that activity remains high. Second, we did not include activation decay to avoid complicating our analysis of the model. Finally, we did not include any noise in the system for the same reason. Even without noise, the network is still stochastic because of the randomness of external input.

From this model, we can observe the variables at any time step as being either 1 or 0 by f_i^t, whether that unit is firing. Accumulating these observations over many time steps generates a probability distribution over all assignments to the variables. This design means that we are only able to encode binary variables, but we can extend variables to include a subset of units for greater flexibility.

This particular model can also be understood as a form of MCMC. If we have n total units, we have $2n$ variables. The state of the network at any time step can be described in terms of each unit's activation and whether it is firing or not. Thus, there are n discrete variables (being either 1 or 0), and n continuous variables. Notably, this particular chain is also non-reversible because of the activation rule drives it forward.

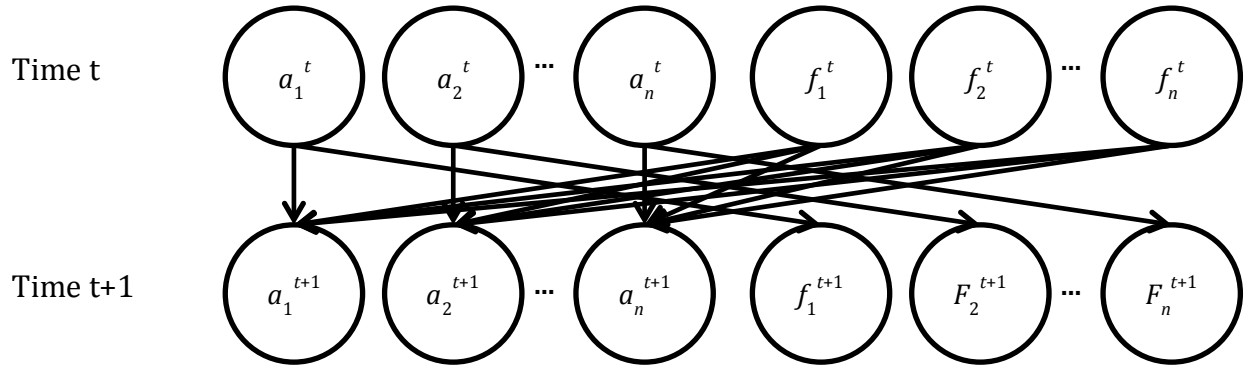


Figure 1 An alternate representation of the model over 2 time steps. All of A nodes represent the activation of a unit, and the F nodes represent whether a unit is firing on a time step.

This interpretation of the network allows us to understand its characteristics in a different manner. First, there hopefully is a stationary distribution that we can eventually reach from this network, assuming the network doesn't oscillate. We can easily imagine a degenerate network of only 2 nodes that alternatively has the first then second node fire, but this requires specific parameterization that avoids the randomness of external input and discretizes the continuous activation of the units.

Second, the burn-in time of the network can be significant and is also dependent on the parameterization of the network. When the network begins running, the initial time steps are spent waiting for the external input to begin further activity as no units are firing to excite other units. In experimentation, we discovered that the burn-in time may allow the network to marginalize unnecessary units. If a unit receives enough net inhibition, its activation may reach a critical minimum where it no longer will ever fire and reset to its resting state.

Search over possible networks

With a very large parameter space, we used stochastic search to find networks with the characteristics we were looking for. Particularly, we wanted the network to be coherent, in that the conditional probabilities calculated from the joint distribution should match the distributions when the network is clamped. Roughly, we would hope that $p(x|e) = p_e(x)$, where p_e is defined as the distribution generated when the units corresponding to the variables in e are clamped to the assigned value (ie f_i^t is fixed to either 1 or 0 for all t). Without coherency, the model is only capable of generating the joint distribution without any evidence. With coherency, the model becomes fully generative and capable of inference without any additional work to weight the samples based on evidence downstream (such as in importance sampling).

An early finding was that networks that encoded a certain joint probability didn't necessarily also encode the corresponding conditional probabilities when units were clamped. Thus, the conditional probabilities do not come for free with the joint probability and must also be learned. Our measure for the coherency of the network was the KL divergence of the conditional probabilities from the full distribution against the clamped distributions over all possible assignments of evidence.

Our initial experiments dealt with networks of only 2 observed variables (and no hidden units), and we discovered that most of the coherent networks had almost complete independence between the 2 variables, measured as $p(ab) - p(a)p(b)$. This result was surprising because independent variables represent a very small part of the space of all possible probability

distributions over 2 variables. Thus, the requirement that the network be coherent seemed to disproportionately represent the independent networks.

Closer analysis suggested that these networks were "too simple." An odd property of these 2 unit networks is that most of the activity was driven by external input into the network. With only one other unit, input from network firing was infrequent, and this caused networks to be largely skewed towards distributions with most of the mass over the assignment being all 0 (because the observed units fired infrequently). Since there is also a lag between a unit firing and that spike affecting the other unit, there wasn't a reliable connection between these 2 units, making them independent.

Another observation was that the weights and external input also tended to be fairly high to ensure an even distribution of probability mass. Intuitively, for $p(a) > .5$, the unit corresponding to a would need to reach threshold at least every other time step. Thus, although the network technically has 2 continuous variables in it, the large steps before spiking actually meant that there were few discrete states that it could actually encounter (Instead of thinking of the activation as a continuous value, we might think of it as "1 excitation from firing" or "2 external input spikes from firing").

Given these constraints, we decided to search for coherent networks with 3 observed units and 7 hidden units. We used stochastic search over the space of models by randomly generating the weights, external input, and probability of receiving external input. Of those models, we also enforced several constraints:

1. The difference between the conditional probabilities and the clamped distributions needed to be below a threshold to be considered coherent.
2. There had to be 2 pairwise dependencies between the observed variables. This requirement was made after observing that many coherent distributions were entirely independent. We decided to search specifically for 2 dependencies so that the network wasn't entirely coupled and could be represented by some non-trivial graphical model.
3. All possible assignments needed to have non-zero probability to ensure that the networks weren't degenerate.

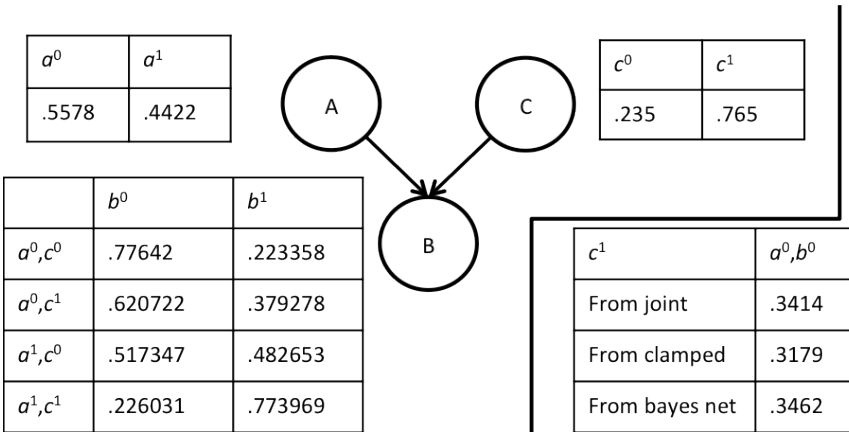
During stochastic search, we generated 1000 samples taken with a 10 step lag to reduce the dependence between samples.

We also preformed hill climbing from the best-performing networks. Because the activation function is non-differentiable, there is no obvious manner to do gradient descent, so small perturbations were added to various parameters and re-tested. During this phase, we gathered 10,000 samples (still with a lag of 10). By computing the distance between 2 runs of the same network, we determined that this was sufficient for reaching a stable distribution.

Results

From our search, we were able to find several (but proportionately few) networks that were both coherent and non-degenerate. Of those, we were unable to discover any properties that characterized them uniquely against the many networks that failed. We can, however, discuss one particularly successful network.

First, this network exhibits almost complete independence between variables a and c. We found that this network is decomposable into a graph with a v-structure as shown below. Additionally, the conditional probabilities and clamped distributions are presented for comparison. Hill climbing, although slow, is effective in steadily decreasing the distance between these distributions.



a^0	a^1
.5578	.4422

c^0	c^1
.235	.765

b^0,c^0	a^0	a^1
From joint	.6773	.3227
From clamped	.6798	.3202
From bayes net	.6543	.3456

	b^0	b^1
a^0,c^0	.77642	.223358
a^0,c^1	.620722	.379278
a^1,c^0	.517347	.482653
a^1,c^1	.226031	.773969

c^1	a^0,b^0	a^1,b^0	a^0,b^1	a^1,b^1
From joint	.3414	.1017	.2086	.3482
From clamped	.3179	.1510	.2314	.2997
From bayes net	.3462	.0999	.2110	.3422

Figure 2 Left: The Bayes net induced by the joint distribution of the model. Right: Comparisons of conditional distributions calculated from the joint distribution and from the clamped distributions.

One method of analysis is to remove hidden units from the model and see how the distribution is affected. Previously, we observed that certain units could be "lesioned" out of the network with no effect on its behavior because their activity had been so heavily inhibited. We found no such effect in this network. All units were essential to its functionality, including some long range effects. As one might expect from a dynamical system, we could not understand its behavior looking only at local effects.

For example, in this network, unit 3 (corresponding to variable c) had a net negative external input. In smaller networks, external input to a unit needed to be positive, or else the activation would steadily drop to tonic inhibition. When we lesioned out unit 7, unit 3 no longer fired (and correspondingly, $p(c) = 0$) as its negative external input caused this tonic inhibition. Although unit 7 didn't have the largest weight on unit 3, it was almost always firing, so it could overcome unit 3's self-inhibition. Additionally, unit 7 had excitatory connections to unit 8 and unit 9, which also had large weights on unit 3. Unit 7 was perpetually on because it had high self-stimulation and a strong connection from unit 4. Unit 4 was almost always on because it had almost all excitatory incoming connections. Thus, the effect of unit 7 on unit 3 was the combined indirect effort of 5 different units. Far beyond the distributed representations often present in connectionist models, the behavior of this network requires the complete dynamics of the model and doesn't gracefully degrade with removing individual units.

Discussion

Learning a Network

In the course of this project, we tried several methods of learning coherent networks, which largely failed. One particular attempt at gradient descent inspired by recurrent back-propagation, however, is worth describing for its insight into the mechanics of this model. Back-propagation is a method of gradient descent over the weights of a feed-forward network by assigning "blame" to each unit backwards from the actual error. Recurrent back-propagation is the extension of it where the network is unrolled over time.

Inspired by this, we directly measured the error over any distribution as the difference between the probability of a conditional probability and some target distribution. For example, if $p(a{=}1,b{=}1,c{=}1) = .5$ in our network, and the target was .6, there would be a .1 error over all 3 units corresponding to those variables. Intuitively, we would like each unit to fire more often, which can be accomplished by increasing incoming weights and the external input. Depending

on the frequency of other units firing, we can propagate this error backwards to other units by assigning "blame" proportional to its influence on the unit with error. We accumulate the error over all probabilities in all conditional distributions, then adjust weights up and down accordingly to attempt to minimize the error. Although this naively would help, this algorithm failed with several contributing factors.

First, the objective function is bumpy and in a very high dimensional space. For example, with 3 observed units and 7 hidden units, there are 120 parameters (100 from the weights of a fully-connected graph, 20 from external input). Additionally, the function is composed of several distances. If we optimize over the conditional probabilities, for 3 binary variables, there are 25 non-trivial conditionals to consider. If we also optimize over the clamped distributions, there are another 25 for a sum of 50 functions.

Second, the activation function we're using is non-linear, and there is not a clear relationship between the network parameters and actual firing. Being non-linear, the math for gradient descent breaks down. More specific to this network, the small changes that gradient descent makes to the network are often insignificant because the weights must be so large. When the changes do affect firing, they often represent tipping points and dramatically change the distribution from the model.

Third, we naively assign blame in the network. To determine blame, we take the total probability of a unit firing, ignoring all more complex interactions between the firing of multiple units. To encode this, we must maintain a history of firing to determine which units caused the error on a particular unit. This thinking, however, soon requires a full history over all 100,000 time steps, which is not a tractable manner to learn.

Finally, models that use recurrent back-propagation have a different goal from this model. Recurrent back-propagation calculates error from a fixed target output, but our model has no fixed target output. Our model periodically observes the state of the network, and the exact sequence is lost when they are aggregated into a distribution. Although this may seem to give us some freedom, it also makes it difficult to apply back-propagation as is.

Conclusion and Future Work

We would like to find a characterization of the space of coherent networks. Thus far, we have only discovered that it is necessary that a network have enough units to drive activity and maintain information. This complexity also presents a major challenge as the properties of the model change dramatically with scale. Even so, our goal is to represent larger, more complex distributions over more variables. Only 3 variables is a toy problem that is tractable using almost any method of inference. How the model scales, particularly with respect to the number of hidden units needed, is unknown. All of these questions will likely benefit from further exploring how this model connects to the theory behind MCMC.

Understanding the characteristics of coherent networks would hopefully help us to learn them as well. The existence of these models is a helpful finding, but not practical if they can only be found through stochastic search and blind hill climbing. Other inspiration for possible directions may be found in contrastive divergence and gradient ascent algorithms for reinforcement learning.

In spite of so many future directions and the challenges explained in the section above, we have made first steps in showing that inference with a connectionist model is possible. Although we have so far been unsuccessful in finding a characterization of these coherent networks or being able to learn any particular distribution, that any exist keeps us hopeful that we can continue to build upon this framework.

References

[1] Xu, F. and Tenenbaum, J. B. (2007). Word learning as Bayesian inference. *Psychological Review 114*(2).

[2] Griffiths, T.L., and Tenenbaum, J.B. (2005). Structure and strength in causal induction. *Cognitive Psychology 51*, 334-384.

[3] Kemp, C., Perfors, A., and Tenenbaum, J. B. (2007). Learning overhypotheses with hierarchical Bayesian models. *Developmental Science 10*(3), 307-321.

[4] Gershman, S. J., Vul, E., and Tenenbaum, J. B. (2010). Perceptual stability as Markov chain Monte Carlo inference. *Advances in Neural Information Processing Systems* 22, 611-619.

[5] Levy, R., Reali, F., and Griffths, T. L. (2009). Modeling the effects of memory on human online sentence processing with particle filters. *Advances in Neural Information Processing Systems 21.*

[6] McClelland, J. L. (1998). Connectionist models and Bayesian inference. In M. Oaksford & N. Chater (Eds.), *Rational Models of Cognition.* Oxford: Oxford University Press. 21-53. Relates computational and implementational levels of analysis.

[7] Ma, W.J., Beck, J., Latham, P. and Pouget, A. Bayesian inference with probabilistic population codes. *Nature Neuroscience*. 9(11), 1432-1438. 2006.

[8] Hryceij, T. (1990) Gibbs sampling in Bayesian networks. *Artificial Intelligence*, 46, 351-63.