Neural-network Modelling of Bayesian Learning and Inference

Milad Kharratzadeh (milad.kharratzadeh@mail.mcgill.ca)
Department of Electrical and Computer Engineering, McGill University, 3480 University Street
Montreal, QC H3A 2A7 Canada

Thomas R. Shultz (thomas.shultz@mcgill.ca)
Department of Psychology and School of Computer Science, McGill University, 1205 Penfield Avenue
Montreal, QC H3A 1B1 Canada

Abstract

We propose a modular neural-network structure for implementing the Bayesian framework for learning and inference. Our design has three main components, two for computing the priors and likelihoods based on observations and one for applying Bayes' rule. Through comprehensive simulations we show that our proposed model succeeds in implementing Bayesian learning and inference. We also provide a novel explanation of base-rate neglect, the most well-documented deviation from Bayes' rule, by modelling it as a weight decay mechanism which increases entropy.
Keywords: Neural-network; Bayes' rule; Bayesian learning and inference; base-rate neglect; weight decay; entropy

Introduction

Bayesian models are becoming prominent across a wide range of problems in cognitive science including inductive learning (Tenenbaum, Kemp, & Shafto, 2006), language acquisition (Chater & Manning, 2006), and vision (Yuille & Kersten, 2006). While these Bayesian ideas provide computation level models, it is beneficial, and sometimes necessary, to appeal to some implementation-level (biological) models to explain human behaviour. Connectionist approaches provide a neural-based model of cognitive processes.

There is growing evidence in neuroscience supporting the relevance of Bayesian models on a neural level (Doya, Ishii, Pouget, & Rao, 2007). Many perceptual and sensorimotor tasks that are learned and performed by the central nervous system can be described in a Bayesian framework (Konrad & Wolpert, 2004). Neural computations, as simple as summing up the firing rates, can be seen as analogous to a Bayesian inference process, with population activity patterns encoding posterior distributions (Pouget, Dayan, & Zemel, 2003).

In theoretical terms, connectionist models show promising prospects in implementing computational-level Bayesian ideas. Under certain assumptions, and inspired by the nature of neural activation functions, neural units can compute posterior probability values (McClelland, 1998). So-called Boltzmann machines, a type of stochastic recurrent neural network, were also suggested to implement Bayesian learning (Ackley, Hinton, & Sejnowski, 1985). Later work introduced generative networks called restrictive Boltzmann machines that can make bidirectional inferences based on what they learned (Hinton & Osindero, 2006).

All these connections between Bayesian and neural-network models motivate further exploration of the relation between the two. In this paper, we propose a complete, modular neural-network structure implementing Bayesian learning and inference in a general form. We do this by using three main modules, two responsible for computing priors and likelihoods based on observations, and one responsible for applying Bayes rule and computing the posteriors. We show that our model is able to successfully implement Bayesian learning and inference and replicate analytical results with high precision in a brain-like fashion which could later be used to gain intuition into how brains implement Bayesian reasoning. Our work also provides a framework to study the deviations from optimal Bayesian reasoning which result from base-rate neglect (Kahneman & Tversky, 1996; Eddy, 1982).

Our work is novel in its precise and complete implementation of a Bayesian framework in a modular, brain-like fashion. The proposed network takes observations as inputs and computes the posterior probabilities (i.e., updated beliefs). Moreover, using a fast, constructive learning algorithm (sibling-descendent cascade-correlation) for the network provides the advantage of a self-organizing learning and inference method which is similar to humans' developmental, autonomous inference and learning (Shultz & Fahlman, 2010). Another novelty of this work is the modelling of base-rate neglect as a weight decay mechanism.

The idea of neural implementation of Bayesian phenomena was suggested before. Shi and Griffiths (2009) introduced a scheme for implementing importance sampling with radial basis function (RBF) networks. They assume that the unit activation functions are of radial basis type. Hence, in their model, a single RBF neuron measures likelihood, resulting in a straightforward implementation of importance sampling. Also, Griffiths et al. (2012) used a linear network to approximate the generalization performance of a probabilistic model. Their linear networks produce different solutions from Bayes on structures other than those based on Wishart priors. Our work differs in two major ways. First, we assume that the characteristics of the likelihood/priors and the network's structure are unknown a priori and learn them for a wide range of likelihood and prior distributions through a constructive training phase. Also, these functions are learned precisely (i.e., no approximation) using a population of neurons with simple though realistic activation functions (sigmoid) rather than linear or complex ones (RBF).

The Basics of Bayesian Learning and Inference

The Bayesian framework addresses the problem of updating beliefs and making inferences based on observed data. As-

sume that we have a set of mutually exclusive and exhaustive hypotheses, $\mathcal{H} = \{h_1, \ldots, h_N\}$, and want to infer which of these hypotheses best explains observed data. In this setting, we denote the degree of belief in different hypotheses by probabilities. Bayesian inference is based on a simple formula known as Bayes' rule. This rule specifies how posterior probabilities (of a hypothesis being true given the observed data) can be computed using the product of data likelihood and prior probabilities:

$$P(h_i|d) = \frac{P(d|h_i)P(h_i)}{P(d)} = \frac{P(d|h_i)P(h_i)}{\sum_{i=1}^{N} P(d|h_i)P(h_i)}. \quad (1)$$

Priors, $P(h_i)$, represent how much we believe in a hypothesis before observing data. Likelihoods, $P(d|h_i)$, denote the probability with which we would expect to observe these data if a hypothesis were true. The denominator, known as marginal probability of data, is a normalizing sum which ensures that the posteriors for all hypotheses sum to one.

The Bayesian framework is generative. This means that observed data are generated by an underlying mechanism or hypothesis. Then, the role of inference is to evaluate different hypotheses and choose the one which is the most likely mechanism responsible for generating the data (i.e., the one with the highest posterior probability). These generative processes can be specified by probabilistic models. Here, we describe likelihoods and priors respectively with probability distributions and mass functions as their generative mechanisms.

Proposed Connectionist Model

We construct a modular neural network implementing Bayesian learning and inference. The algorithm we use to build our artificial neural modules is a variant of the cascade-correlation (CC) method called sibling-descendant cascade-correlation (SDCC) which is a constructive method for learning multi-layer artificial neural networks (Baluja & Fahlman, 1994). CC offers two major advantages over standard back-propagation (BP) methods. First, it constructs the network in an autonomous fashion (i.e., a user does not have to design the topology of the network). Second, its greedy learning mechanism can be orders of magnitude faster than the standard BP algorithm. In addition to these, SDCC has another important benefit; due to its design, it can often reduce the depth of the network drastically (Baluja & Fahlman, 1994).

We build our model in a modular fashion. Module 1 (shown in Fig. 1) implements Bayes' rule. For this module, we assume that there are two hypotheses to compare; extending it for any finite number of hypotheses is straightforward. There are three inputs (the prior and the two likelihoods) and one output (the posterior), and the module learns and implement (1). We run this module once for each hypothesis.

When data are observed in consecutive rounds, posteriors at one round are taken as priors for the next round; this is how the beliefs are updated in light of new observed data (rational inductive inference). Therefore, by connecting the output of module 1, $P(h_1|d)$, to the input corresponding to the prior, $P(h_1)$, we can model this aspect of human cognition.

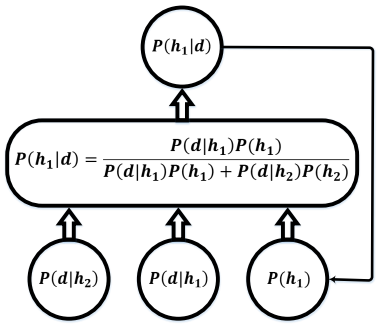


Figure 1: Module 1 computes the posterior based on the likelihoods and prior and according to Bayes' rule.

Module 1 assumes that the values of prior and likelihood probabilities are given and then applies Bayes' rule using them as input. Module 2 (shown in Fig. 2) is responsible for computing the likelihoods. It takes observation(s) as input(s). Due to the generative nature of the Bayesian framework, we describe likelihoods as probability distributions. The role of module 2 is to learn these distributions as the underlying mechanisms generating data. However, this should be done without any implicit or explicit knowledge about the specifications of the distribution and solely based on the observed training data.

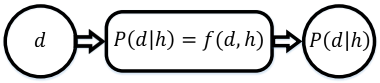


Figure 2: Module 2 computes the likelihoods based on the observed data.

We denote the generative process (probability distribution) as a function of the observed data and hypothesis, $f(d,h)$. For example, if the likelihood distribution for hypothesis h is a Gaussian with average h and standard deviation 1, then:

$$f(d,h) = \frac{1}{2\pi} e^{\frac{(d-h)^2}{2}}. \quad (2)$$

Finally, module 3 (shown in Fig. 3) computes the hypotheses' priors by learning their generative discrete distribution function. Note that the input to this module can be chosen from a finite number of possible hypotheses, $\mathcal{H} = \{h_1, \ldots, h_N\}$, and hence the generative distribution is discrete. It takes the hypotheses as input and gives their prior probability as output. We represent the generative mechanism of priors as a probability mass function denoted by $g(h)$. For instance, this function can be of the following form:

$$g(h) = \alpha \cdot e^{\frac{h^2}{2}}, \quad (3)$$

where α is chosen so that the sum of priors equals 1.

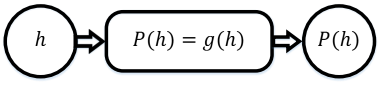


Figure 3: Module 3 computes the prior probabilities.

If we have N hypotheses, to run a complete Bayesian learning and inference, we learn modules 1 and 3 one time and use them N times (for respectively computing the posterior and prior of each hypothesis). However, since the likelihood distributions might be different for different hypotheses, we learn N different units of module 2 and use each of them once.

Simulation Results

Setup

We test each module individually. Using SDCC, we first train each module by presenting input(s)-output pairs as training patterns. Then, we test the generalization ability of the learned module by utilizing a set of input(s)-output testing patterns. The accuracy of the module's outputs is examined by comparing them with the correct outputs presented in the testing set. In all our modules, the hidden units have sigmoid activation functions and the output units have linear activation functions.

Module 1

Module 1 takes the two likelihoods, $P(d|h_1)$ and $P(d|h_2)$, and the prior, $P(h_1)$ as inputs and gives the posterior, $P(h_1|d)$, as the output. The training set is all the triplets starting from $(0.1, 0.1, 0.1)$ and going up to $(0.9, 0.9, 0.9)$ with steps of size 0.1 paired with appropriate output derived from (1). The testing set is all the triplets starting from $(0.05, 0.05, 0.05)$ and going up to $(0.95, 0.95, 0.95)$ with steps of size 0.1 paired with correct outputs. Thus, we have 1000 training and 1000 testing points and these two sets have no overlap.

Because of the random nature of SDCC, we get a different network with different structure every time we run the algorithm on the training set. Across 50 learned networks for module 1, on average, each network had 13.2 hidden units and 3.4 hidden layers, and the training took 970 epochs.

We compare the outputs of module 1 with the actual results of Bayes' rule by plotting them against each other in a scatter plot. In order to check the generalization accuracy of the built network we use the testing set data (which are not used in training the network) in our analysis. The results in Fig. 4 show that there is a high correlation between the outputs of the network and the true values. Also the slope and y-intercept of the fitted line are respectively near 1 and 0. In sum, the learned module 1 produces highly precise outputs and we can conclude that it implements the Bayes' rule successfully.

Module 2

Module 2 computes the likelihood given the observed data. This module learns the distribution generating data solely based on the training set presented to it during the training phase. It has no prior information about the form or characteristics of this distribution. Different hypotheses can have different likelihood distributions, hence we run one module 2 for each hypothesis. Through several experiments, we show that module 2 can learn a variety of likelihood distributions.

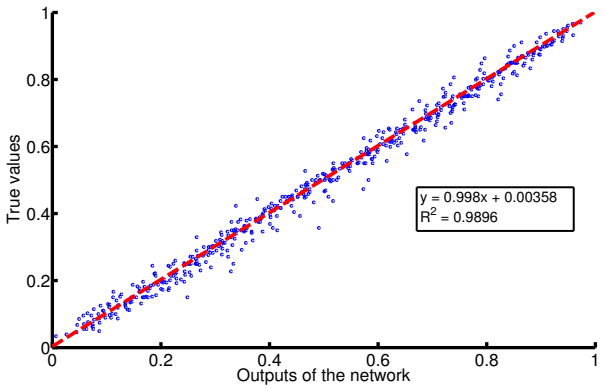


Figure 4: Outputs of module 1 plotted against true values.

We start with the Gaussian distribution. In this case, the likelihood is given as in equation (2). The observed data, d, is the input and its likelihood is the output. Given the hypothesis h, we have a Gaussian likelihood with mean h, and standard deviation 1. For example, h could be 0 or 1 in the case where we have two hypotheses. Each of these distributions can be learned using SDCC. We define our training set to be the collection of equidistant points from -5 to 5 with steps of size 0.01 and the testing set to be equidistant points from -4.975 to 4.975, with the same step size, all paired with appropriate outputs derived from (2). Across 50 learned networks, on average, module 2 had 1.8 hidden layers and 8.2 hidden units.

To check the accuracy of the module's outputs, we plot them against the actual Gaussian values in a scatter plot shown in Fig. 5. The high correlation and the equation of the fitted line show that this module succeeds in learning the Gaussian distribution. To further assess the performance of module 2, we plot the probability distribution function generated by it alongside the actual Gaussian distribution in Fig. 6a. We observe that the two curves are very close.

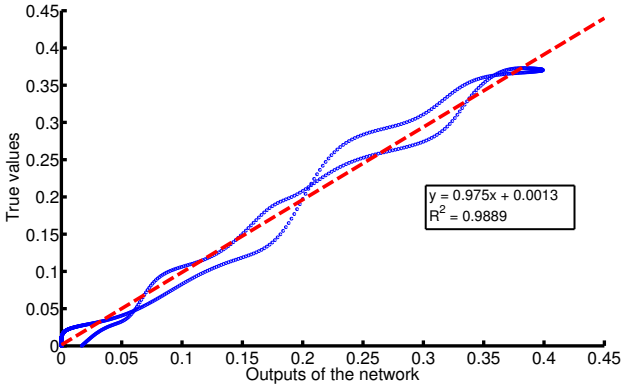
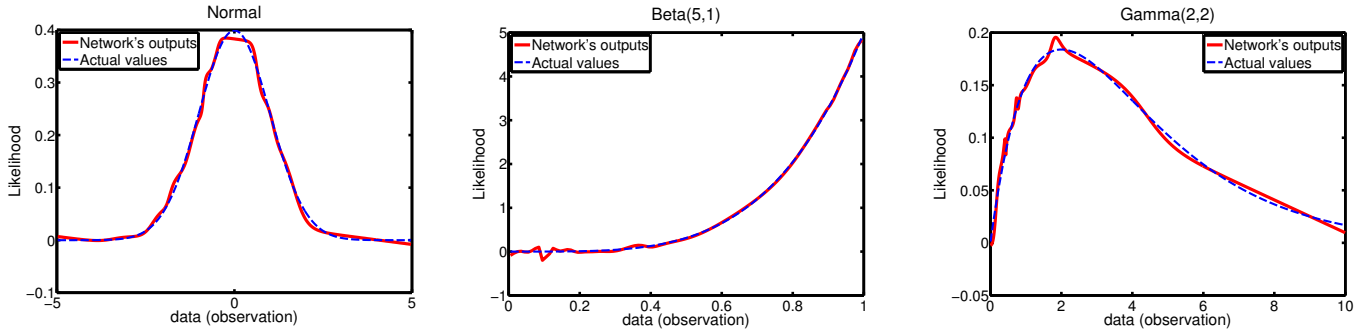


Figure 5: Outputs of module 2 plotted against true values of a Normal distribution. For the Normal, there are two x values for every y value; hence, there are two lines of dots.



(a) Normal distribution; Gaussian distribution with mean 0 and standard deviation 1.

(b) Beta distribution with parameters $\alpha = 5$ and $\beta = 1$.

(c) Gamma distribution with parameters $k = 2$ and $\theta = 2$

Figure 6: Outputs of module 2 compared with the actual values for three sample distributions.

The replication of the original likelihood distribution by our model is of special importance, because it is done without any explicit or implicit information about the actual distribution. The only information available to the learner is the probability values of the points in the training set. Based on that and by generalization, module 2 learns the actual distribution which generates the data. This capacity is not limited only to the Gaussian distribution. In our simulations, we observe that a wide range of distribution functions can be learned by this module. In Fig. 6, we show that for a couple of sample probability distributions module 2 produces results very close to the actual distribution functions.

Module 3

Module 3 computes hypotheses' priors. Although their inputs and outputs are of different nature, modules 2 and 3 are functionally the same. They both compute the output based on a probability distribution over the input. Therefore, like module 2, module 3 is capable of learning the underlying structure of its inputs — the possible hypotheses. The only difference between modules 2 and 3 is that the likelihood distributions (in module 2) are continuous while prior distributions (in module 3) are discrete. We analyse module 3 in more detail while discussing base-rate neglect in the next section.

Base-rate Neglect as Weight Decay

In contemporary cognitive science, rationality in learning and inference is frequently defined and measured in terms of conformity to Bayes' rule. However, this appears to conflict with the Nobel-prize-winning work showing that people are somewhat poor Bayesians due to biases such as base-rate neglect, representativeness heuristic, and confusing the direction of conditional probabilities (Kahneman & Tversky, 1996). Even experienced medical professionals deviate from optimal Bayesian inference and make major errors in their probabilistic reasoning (Eddy, 1982). More recently, Prime and Shultz showed that base rates (i.e., priors) are not entirely ignored but just de-emphasized (Prime & Shultz, 2011).

We first show how base-rate neglect can be interpreted in the Bayesian framework. Then, as an important contribution of this work, we show that this neglect can be modelled as weight decay in our proposed neural network. This explanation is particularly of interest because it is neurologically plausible and in accordance with theories explaining memory loss and decline in some other cognitive functions as a result of synaptic decay over time (Hardt, Nader, & Nadel, in press).

Base-rate neglect is an error in computing the posterior probability of a hypothesis without taking full account of the priors. In the Bayesian framework, in the extreme case of entirely ignoring the priors, Bayes' rule in (1) becomes:

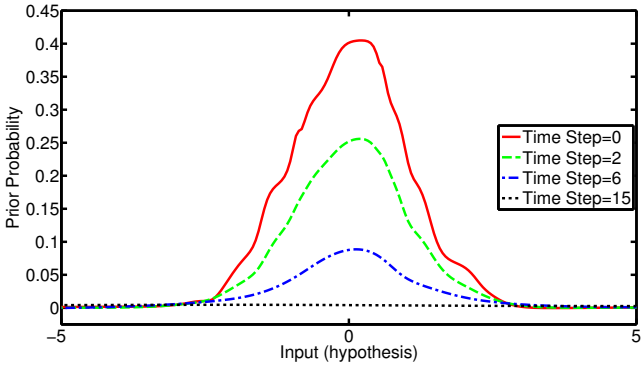
$$P(h_i|d) = \frac{P(d|h_i)}{\sum_{i=1}^{N} P(d|h_i)}. \tag{4}$$

Looking at this equation from a different perspective, we can assume that in the original Bayes' rule, all the hypotheses had equal priors and these priors were cancelled out to give equation (4). Therefore, in the Bayesian framework, base-rate neglect is translated into assuming equal priors (i.e., equiprobable hypotheses). This means that the more the original priors (base rates) are averaged out and approach the uniform distribution, the more they are neglected in Bayesian inference. We can explain this more abstractly by using the notion of entropy defined in information theory as a measure of uncertainty. Given a discrete random variable $X = \{h_1, \ldots, h_N\}$ with probability mass function $P(\cdot)$, its entropy is defined as:

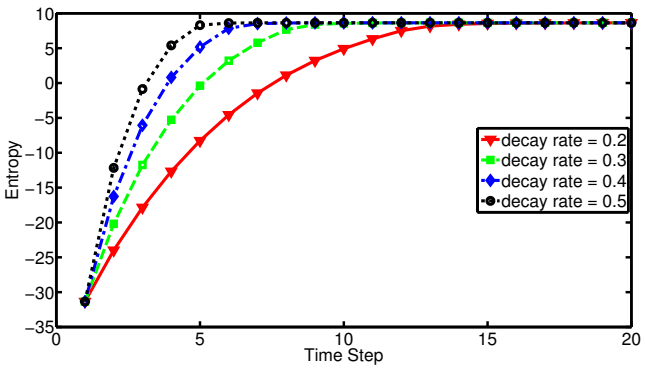
$$H(X) = -\sum_{i=1}^{N} P(h_i) \log_2 P(h_i). \tag{5}$$

Entropy quantifies the expected value of information contained in a distribution. It is easy to show that a uniform distribution has the maximum entropy (equal to $log_2 N$) among all discrete distributions over the set $\{h_1, \ldots, h_N\}$. In sum, we can conclude that in the Bayesian framework, base-rate neglect is equivalent to ignoring the priors in the form of averaging them out to get a uniform distribution, or equivalently, increasing their entropy.

We show that weight decay in our proposed neural network produces the same results as just described, namely approaching a uniform distribution and increasing entropy. We then conclude that we can model base-rate neglect in the Bayesian framework by a weight decay mechanism in our brain-like



(a) The priors' initially Gaussian distribution approaches uniform as time passes and weights decay more (decay rate = 0.2).



(b) The entropy of prior distributions increases as time passes and weights decay more.

Figure 7: Effects of connection weight decay.

network implementing the posterior inference. We take priors as the states of a learning and inference system. As weights decay, the system moves towards more stable states and thus the entropy increases. In special cases, some priors are updated so often that the effects of decay are overcome and these priors stay strong. On the other hand, likelihoods are new evidence and thus, they are not subject to much decay.

In module 3, an SDCC network learns the hypotheses' priors. Assume that we have N hypotheses, $\{h_1,\ldots,h_N\}$, with probability mass function given by equation (3). We present the results for this specific mass function, but the results are similar for other discrete distributions. Our training set is the collection of 401 equidistant points from -10 to 10 (with steps of size 0.05), and our testing set is the collection of 399 equidistant points from -9.975 to 9.975, all paired with the correct outputs derived from (3). We choose these sets such that there is no overlap between the testing and training sets in order to measure the generalization abilities of the module. Note that we do not specify the form of the probability mass function in any way and the network learns it by generalization from the training input. Also note that in the case we consider here, $N = 399$ as we define our hypotheses to be the collection of points in the testing set.

After learning, the network's weights decay exponentially over time steps as follows:

$$W_i(t+1) = (1-r)\cdot W_i(t) = (1-r)^t \cdot W_i(1), \qquad (6)$$

where $W_i(t)$ is the weight of connection i at time t and r is the decay rate. Clearly, as the connection weights of the learned network decay, the output will change. Fig. 7 demonstrates the effects of weight decay on the output of module 3. In Fig. 7a, for $r = 0.2$, we observe that as time passes, and hence as the weights continue to decay, the distribution of the hypotheses approaches a uniform distribution. We consider a discrete case where we have a finite number of hypotheses, and therefore Fig. 7 represents the probability mass function where the sum of all probabilities must be equal to 1. Note that in the 15th time step, the distribution is almost uniform; thus, the value of the probability is $1/N = 1/399$ (this small value should not be mistaken with zero in Fig. 7(a)).

There is a literature on using weight decay to improve the ability of neural networks to generalize (Krogh & Hertz, 1992). Krogh and Hertz showed that a weight decay can improve generalization by suppressing any irrelevant components of the weight vector. This effect is evident in Fig. 7a, as the bumps (overfitting) get smoothed out as time passes.

In Fig. 7b, entropies are plotted as a function of time for four values of decay rate. In all four cases, entropy increases with time until it converges to its maximum which corresponds to uniform prior distribution. In our case, the maximum entropy is $\log_2 N = \log_2 399 = 8.64$.

In conclusion, we show that the proposed neural network model contributes to the resolution of the discrepancy between demonstrated Bayesian successes and failures by modelling base-rate neglect as weight decay in a connectionist network implementing Bayesian inference. This is done by showing that as weights decay, the priors' probability mass function approaches a uniform distribution and its entropy increases. Consequently, the prior terms eventually cancel out from Bayes' rule, resulting in the neglect of base rates.

Discussion

We propose a modular neural-network structure to implement Bayesian learning and inference. Through simulations, we show that the proposed three modules, responsible for computing Bayes' rule, likelihoods, and priors, succeed in learning their assigned task. Employing a weight-decay mechanism, we provide a novel explanation of base-rate neglect, the most well-documented deviation from Bayes' rule. We show that weight decay increases the entropy of priors in a Bayesian system. In nature, this is very similar to the second law of thermodynamics which states that the entropy of isolated systems never decreases and that they evolve towards equilibrium — the state of maximum entropy. Our model of base-rate neglect predicts that older, less continuously supported (i.e., more isolated) priors would be subject to more decay, and consequently more neglect.

Our model is a first step towards implementing the Bayesian framework with neural networks. As such, it still

has several limitations. For instance, our current model is only capable of handling a finite number of hypotheses. When the number of hypotheses gets uncountably infinite, the current structure will be infeasible due to large numbers of outputs that must be remembered.

Our networks learn to approximate Bayesian functions with the output being a function value. However, this assumption might be unrealistic and does not explain how the brain innately represents probabilities. It is more realistic to approximate such functions from input instances occurring at various frequencies. Although we do not address this here, our further experiments show that SDCC networks can do this, which is effectively probability matching. With that, our modular neural network, similar to human brain, makes sense of data by representing probability distributions and applying Bayes' rule to find the best explanation for any given data.

In this paper, we apply our model to base-rate neglect. However, there are many other subtle and potentially difficult Bayesian phenomena, such as hierarchical Bayesian structures and causal networks, to consider. Also, our work does not address the origin of the Bayesian competencies; all we show is that neural networks can implement Bayesian inference and learning. The origin of theses capacities could be in evolution, learning, development or some combination. Finally, we only use SDCC as the learning method and do not try other neural-network approaches.

There is plenty of scope for future research to address the issues just discussed. For instance, we can illuminate the issue regarding the origin of the Bayesian competencies of our model by agent-based simulations of evolution of Bayesian inference and learning. Preliminary results in the context of social learning strategies have shown that evolution favours Bayesian learning, based on passing posteriors, over imitation and environment sampling (Montrey & Shultz, 2010).

With no doubt, Bayesian models provide powerful analytical tools to rigorously study deep questions of human cognition that have not been previously subject to formal analysis. These Bayesian ideas, providing computation-level models, are becoming prominent across a wide range of problems in cognitive science. On the other hand, connectionist models offer an implementation-level framework for modelling mental phenomena in a more biologically plausible fashion. We present this work in the spirit of theoretical unification and enhancement of these two approaches. We are not advocating replacement of one approach in favour of the other.

Acknowledgements

This research was supported in part by McGill Engineering Doctoral Award to MK and a Discovery grant to TRS from the Natural Sciences and Engineering Research Council of Canada. Mark Coates and Deniz Üstebay contributed thoughtful comments on an earlier draft.

References

Ackley, H., Hinton, E., & Sejnowski, J. (1985). A learning algorithm for Boltzmann machines. *Cog. Sci.*, 147–169.

Baluja, S., & Fahlman, S. E. (1994). *Reducing network depth in the cascade-correlation learning architecture* (Tech. Rep.). Carnegie Mellon University, School of Computer Science.

Chater, N., & Manning, C. D. (2006). Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, *10*(7), 335 - 344.

Doya, K., Ishii, S., Pouget, A., & Rao, R. (2007). *Bayesian brain : probabilistic approaches to neural coding*. Cambridge, MA : MIT Press.

Eddy, D. M. (1982). Probabilistic reasoning in clinical medicine: problems and opportunities. In D. Kahneman, P. Slovic, & A. Tversky (Eds.), *Judgment under uncertainty: Heuristics and biases.* Cambridge Univ. Press.

Griffiths, T. L., Austerweil, J. L., & Berthiaume, V. G. (2012). Comparing the inductive biases of simple neural networks and bayesian models. *In Proc. the 34th Annual Conf. of the Cog. Sci. Society*.

Hardt, O., Nader, K., & Nadel, L. (in press). Decay happens: the role of active forgetting in memory. *Trends in Cog. Sci.*.

Hinton, G. E., & Osindero, S. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 2006.

Kahneman, D., & Tversky, A. (1996). On the reality of cognitive illusions. *Psychological Review*, *103*, 582 - 591.

Konrad, K., & Wolpert, D. (2004). Bayesian integration in sensorimotor learning. *Nature*, *427*, 244 - 247.

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. *Advances in neural information processing systems*, *4*, 950-957.

McClelland, J. L. (1998). Connectionist models and Bayesian inference. In M. Oaksford & N. Chater (Eds.), *Rational models of cognition.* Oxford, UK: Oxford Univ. Press.

Montrey, M., & Shultz, R. (2010). Evolution of social learning strategies. , 95-100.

Pouget, A., Dayan, P., & Zemel, R. (2003). Inference and computation with population codes. *Annual Review in Neuroscience*, *26*, 381-410.

Prime, H., & Shultz, T. R. (2011). Explicit Bayesian reasoning with frequencies, probabilities, and surprisals. *Proc. of 33rd Annual Conf. Cog. Sci. Society*.

Shi, L., & Griffiths, T. (2009). Neural implementation of hierarchical Bayesian inference by importance sampling. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems 22* (pp. 1669–1677).

Shultz, T. R., & Fahlman, S. E. (2010). Cascade correlation. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (p. 139 - 147). Heidelberg, Germany: Springer-Verlag.

Tenenbaum, J. B., Kemp, C., & Shafto, P. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, *10*(7), 309 - 318.

Yuille, A., & Kersten, D. (2006). Vision as Bayesian inference: analysis by synthesis? *Trends in Cog. Sci.*, *10*(7), 301 - 308.