Comparing the inductive biases of simple neural networks and Bayesian models

Thomas L. Griffiths (tom_griffiths@berkeley.edu)
Joseph L. Austerweil (joseph.austerweil@gmail.com)
Vincent G. Berthiaume (vberthiaume@berkeley.edu)
Department of Psychology, University of California, Berkeley, CA 94720 USA

Abstract

Understanding the relationship between connectionist and probabilistic models is important for evaluating the compatibility of these approaches. We use mathematical analyses and computer simulations to show that a linear neural network can approximate the generalization performance of a probabilistic model of property induction, and that training this network by gradient descent with early stopping results in similar performance to Bayesian inference with a particular prior. However, this prior differs from distributions defined using discrete structure, suggesting that neural networks have inductive biases that can be differentiated from probabilistic models with structured representations.

Keywords: Bayesian modeling, connectionism, inductive biases, property induction

Introduction

Cognitive scientists use different mathematical formalisms to model human cognition. Understanding the relationships between these approaches is critical to resolving questions about the nature of the mind. Recently, researchers have debated whether probabilistic or connectionist models of cognition provide better prospects for making progress in cognitive science (Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010; McClelland et al., 2010). One of the key issues in this debate is that many probabilistic models are defined in terms of structured, discrete representations, while connectionist models use continuous, graded representations that can mimic discrete structure when needed. A possible resolution would be to view probabilistic and connectionist models as lying at different levels of analysis (Marr, 1982), with neural networks a continuous approximation to Bayesian inference over discrete representations. However, this requires establishing whether such an approximation is possible.

To explore this issue, we use the problem of property induction as a case study for investigating the relationship between probabilistic models of cognition and neural networks. Property induction – inferring the properties of a novel object based on the properties of other objects – has played a key role in the debate between probabilistic and connectionist models. An influential probabilistic model explains human property induction in terms of Bayesian inference over discrete representations such as graphs and trees (Kemp & Tenenbaum, 2009), whereas a successful connectionist model explains people's inferences via continuous representations learned by gradient descent (Rogers & McClelland, 2004).

We use a combination of mathematical analysis and computer simulations to address three questions. First, can a probabilistic model with a discrete representation for a set of objects be approximated by a neural network model with continuous representations? Second, are the solutions that tend to be found by training neural networks by gradient descent comparable to those produced by Bayesian inference (that is, are the *inductive biases* of these approaches related)? Finally, how compatible are the inductive biases of neural networks with those of structured probabilistic models? We provide positive answers to the first two questions, showing that a simple neural network can always approximate a probabilistic model of property induction, and that training this network using a gradient descent algorithm is similar to Bayesian inference with a particular prior distribution. However, we also show that there remains a significant difference between this prior and distributions based on discrete representations.

Mathematical analysis

Our mathematical analysis focuses on comparing the model of property induction used by Kemp and Tenenbaum (2009; henceforth KT09) with a linear neural network.

Setting up the problem

The KT09 model assumes that we want to capture the joint distribution of the elements of continuous N-dimensional vectors $\mathbf{x}$ indicating the value of a single property for N objects.[1] This distribution, $p(\mathbf{x})$, results from a diffusion process on a graph. The diffusion process induces a multivariate Gaussian distribution on $\mathbf{x}$ with mean zero and covariance

$$\Sigma_{\text{discrete}} = \left(\Delta + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1} \quad (1)$$

where Δ is the Laplacian of the graph, being $\mathbf{G} - \mathbf{I}$ for a graph with adjacency matrix $\mathbf{G}$, and $\mathbf{I}$ is the identity matrix.

Now consider a linear neural network model.[2] This model represents an observed $N \times M$ matrix (the values of M properties for N objects) as the product

$$\mathbf{X} = \mathbf{YZ} \quad (2)$$

where $\mathbf{X}$ is the $N \times M$ matrix of observed objects, $\mathbf{Y}$ is an $N \times K$ matrix, and $\mathbf{Z}$ is a $K \times M$ matrix. In this model, $\mathbf{Z}$ is the representation of the set of properties on a hidden layer with K units (as might be encoded in the weights from an

[1] This formulation is a little counter-intuitive, as the set of objects is fixed but the set of properties is left open (ie. new properties tend to be observed, rather than new objects). This differs from the most intuitive way of thinking about the problem for a neural network, in which the network is trained to predict the properties that objects have, with the set of properties fixed and the set of objects left open.

[2] Neural network models typically use non-linear activation functions at the hidden layer. This complicates the analysis, but we hope to explore the consequences of such non-linearities in future work. We return to this point in the Discussion.

input layer to the hidden layer, with localist coding of properties at the input layer) and $\mathbf{Y}$ encodes the relation of properties over objects on the hidden layer.[3] A single property vector is generated by multiplying the weight matrix, $\mathbf{Y}$, by the vector representing the property, $\mathbf{z}$, to obtain $\mathbf{x} = \mathbf{Y}\mathbf{z}$. The model is trained by finding weights $\mathbf{Y}$ and representations $\mathbf{Z}$ that minimize the error in reconstructing $\mathbf{X}$.

Approximating generalization

It should be clear that the linear neural network can perfectly reproduce any observed matrix $\mathbf{X}$, provided K is greater than or equal to the rank of $\mathbf{X}$. This follows simply by thinking about Equation 2 as a set of equations for the entries in $\mathbf{X}$ where the entries in $\mathbf{Y}$ and $\mathbf{Z}$ are free parameters – we can reproduce $\mathbf{X}$ if we have enough free parameters to construct its linearly independent columns. The more interesting question is thus how the network will generalize. That is, what does it predict for a new property based on what it has learned from the observed properties?

Analyzing generalization requires making assumptions about the nature of the $\mathbf{z}$ vector for a novel property. If we assume that $\mathbf{z}$ follows a multivariate Gaussian distribution with mean zero and covariance $\sigma_z^2\mathbf{I}$, we can obtain some results that provide connections between the neural network and Bayesian approaches. This is a reasonable assumption if the weights from the localist node from an unobserved property to the hidden layer are assumed to be independently drawn from a Gaussian distribution. This will be true if the initial weights are drawn from a Gaussian, but as we show below it is also consistent with the implicit prior assumed by gradient descent algorithms.

We can determine the prediction the neural network will make for a new property by asking how $\mathbf{x}$ is distributed given $\mathbf{Y}$. Using standard Gaussian identities, $\mathbf{x}$ will be multivariate Gaussian with mean zero and covariance

$$\Sigma_{\text{continuous}} = \sigma_z^2\mathbf{Y}\mathbf{Y}^T \tag{3}$$

since $\mathbf{x}$ is a linear function of a Gaussian random variable.

Characterizing the distribution on $\mathbf{x}$ implied by this model makes it straightforward to construct a condition under which the model produces the same joint distribution as a probabilistic model based on any discrete graph structure: This will occur when $\Sigma_{\text{discrete}} = \Sigma_{\text{continuous}}$. This can be used to establish a direct connection between the neural network's representations for the objects and the graph Laplacian Δ. In particular, $\mathbf{Y}$ can be obtained from the eigenvectors of Δ. If the network is trained from a matrix $\mathbf{X}$ of property values sampled from $p(\mathbf{x})$, then any learning algorithm that produces a representation corresponding to the principal components of $\mathbf{X}$ will approximate this outcome, with the approximation improving as the number of samples M increases. Thus, the answer to our first question is that the probabilistic model can be approximated arbitrarily well by a neural network.

Establishing that our simple neural network with continuous representations can potentially approximate the generalization performance of a probabilistic model using a discrete representation raises a different question: Will these models also perform similarly when learning those representations from data? That is, if we train a neural network model on a finite number of samples from $p(\mathbf{x})$, will it behave similarly to a probabilistic model that infers a discrete representation from the same data via Bayesian inference? This is a question about the inductive biases of these two different approaches to learning – those factors that lead a learning algorithm to favor one solution over another. In the context of the property induction problem, this question reduces to whether the predictions produced by the neural network after training will be similar to those resulting from Bayesian inference with a particular prior distribution.

Gradient descent and Bayesian inference

Gradient descent is a standard approach to training a neural network, where the weights are assigned small random values and then modified in the direction indicated by the gradient of the error repeatedly for a fixed number of training iterations. In this section, we summarize results showing that this learning algorithm behaves similarly to Bayesian inference with a Wishart prior on covariance matrices.

For simplicity, we start by considering the problem of updating $\mathbf{z}$ for a single property, keeping $\mathbf{Y}$ fixed. In this case the goal is to find the $\mathbf{z}$ such that $\mathbf{Y}\mathbf{z}$ minimizes the squared error in reconstructing the corresponding property vector $\mathbf{x}$. We can write the objective function as $(\mathbf{x}-\mathbf{Y}\mathbf{z})^T(\mathbf{x}-\mathbf{Y}\mathbf{z})$. Differentiating, we obtain the weight update rule

$$\Delta\mathbf{z} = \eta\mathbf{Y}^T(\mathbf{x}-\mathbf{Y}\mathbf{z}) \tag{4}$$

where η is a learning rate (assuming simultaneous updates).

For comparison with performing Bayesian inference, we can derive the estimate for $\mathbf{z}$ that we would obtain by assuming a Gaussian prior and finding the posterior mean (or the maximum a posteriori value, as they are the same in this case). The Bayesian estimate is

$$\hat{\mathbf{z}} = (\mathbf{Y}^T\mathbf{Y} + \frac{\sigma_x^2}{\sigma_z^2}\mathbf{I})^{-1}\mathbf{Y}^T\mathbf{x} \tag{5}$$

where σ_x^2 is the assumed noise variance in $\mathbf{x}$.

Inspecting these two equations, we can see that they use two different forms of *regularization* – approaches to controlling the complexity of the solution found by learning. Neural network training typically starts with weights close to zero, so weights grow over successive passes through the data. Stopping early keeps weights smaller. In the Bayesian solution, the ratio of σ_x^2 to σ_z^2 controls the size of the weights: If σ_z^2 is small relative to σ_x^2 (i.e., we are more confident in our prior

[3] Since the model is linear, this interpretation can be "transposed" to give another interpretation, where $\mathbf{Y}$ is the hidden layer representation of the objects and $\mathbf{Z}$ the weights for the properties. This is a more intuitive way of formulating the model and is also more consistent with connectionist models of these phenomena, as advocated by Rogers and McClelland (2004). However, this interpretation is a little harder to use to get intuitions about the results shown below.

beliefs than the observed data), the corresponding term can dominate the matrix that is inverted, reducing the weights proportionally. Despite this difference in regularization style, there are cases where they will produce similar results: If $\mathbf{z}$ is close to zero and $\mathbf{Y}^T\mathbf{Y}$ is close to $c\mathbf{I}$, then $\hat{\mathbf{z}}$ will equal $\mathbf{z}$ after one pass of gradient descent with $\eta = 1/(c + \frac{\sigma_x^2}{\sigma_z^2})$.

More generally, it is possible to show that the solution produced by a linear neural network trained by gradient descent with early stopping is equivalent to generating a Bayesian estimate with a Gaussian prior (Fleming, 1990; Santos, 1996). When applied to Equation 4, these results indicate that following this learning rule is equivalent to assuming a Gaussian prior on $\mathbf{z}$ with mean zero and a covariance determined by $\mathbf{Y}$ and the number of iterations of learning.

While the analysis presented so far has focused on $\mathbf{Z}$, the linearity of the network means that learning $\mathbf{Y}$ can be analyzed in the same way. A Gaussian prior on $\mathbf{Y}$ implies that the implicit prior on $\mathbf{Y}\mathbf{Y}^T$ assumed by a neural network trained by gradient descent with early stopping is a Wishart distribution, the distribution obtained by taking the product of two matrices drawn from a multivariate Gaussian (Muirhead, 1982).

Summary of mathematical results

The key results of the mathematical analyses presented in this section are that the generalization performance of the KT09 model can be approximated by a linear neural network model with continuous representations, and that the inductive bias induced by training the neural network by gradient descent with early stopping should be similar to that of Bayesian inference with a Wishart prior on covariance matrices. These results make two clear predictions: Neural networks should perform best when learning from data whose covariance matrices are Wishart distributed, and we should expect them to perform more similarly to Bayesian models that use a Wishart prior than to models with other priors.

These results also raise a question: How similar is the Wishart distribution to distributions that are based on discrete representations? If the distributions are similar, then the inductive biases of neural networks and probabilistic models with discrete representations will also be similar, meaning that these approaches need not be seen as lying in opposition to one another. If the distributions are different, then there are opportunities to empirically separate these accounts and we cannot view simple neural networks as a scheme for approximating the solutions identified by probabilistic models.

Simulations

We explored the issues raised by our mathematical analyses through simulations comparing the performance of neural networks and Bayesian models with different prior distributions. The set of priors that we used included the Wishart distribution as well as several distributions based on discrete structures. Following the KT09 model, we included distributions on covariance matrices by defining a distribution on graphs $\mathbf{G}$ and then deriving a covariance matrix for each graph. The distributions on graphs we considered were stochastic graph grammars that generate trees, chains, grids, and partitions (Nagl, 1986; Kemp & Tenenbaum, 2008) and Erdös-Rényi random graphs (Erdös & Rényi, 1959).

Our analysis proceeded as follows. For each prior distribution over covariance matrices, we generated T samples of $N \times N$ covariance matrices $\Sigma_1, \ldots, \Sigma_T$. From each covariance matrix, we sampled a $N \times M$ matrix $\mathbf{X}$ containing the values of M features for each of the N objects ($\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_M], \mathbf{x}_i \overset{iid}{\sim} N(\mathbf{0}, \Sigma)$). We then computed the marginal probability of these samples under a Wishart distribution, integrating over its parameters. This let us determine how closely different priors relate to the Wishart distribution.

To compare the different approaches to learning, we applied the neural network and Bayesian models to all of the samples of $\mathbf{X}$ we had produced. We found $\mathbf{Y}\mathbf{Y}^T$ at different stopping points and compared this to the true covariance matrix for data generated from each of the different priors. The goal of this first analysis was to evaluate whether the neural network performed best with data whose covariance matrix was Wishart distributed. For the second analysis, we also obtained an estimate of the covariance matrix from each sample using Bayesian inference with each of the different prior distributions and calculated the distance between these covariance matrices and $\mathbf{Y}\mathbf{Y}^T$. This allowed us to examine how the distance between the solutions produced by the neural network and Bayesian inference was related to the extent to which the priors were similar to a Wishart distribution.

Calculating marginal Wishart probabilities

To perform our analysis, we must be able to calculate how close a distribution is to a Wishart. We did this using the marginal probability of a set of covariance matrices under a Wishart, integrating over the parameters of the distribution. The result is a measure of the "Wishartiness" of the covariance matrices, which can be applied to samples from different distributions in order to evaluate their similarity to a Wishart.

Assume we have a Wishart distribution with degrees of freedom b and covariance center $\mathbf{S}$, and that $\mathbf{S}$ is drawn from an inverse-Wishart distribution with parameters a and Ψ. We draw covariance matrices $\Sigma_1, \ldots, \Sigma_T$ from this distribution. The marginal probability of $\Sigma_1, \ldots, \Sigma_T$ given a, b, and Ψ is

$$p(\Sigma_1, \ldots, \Sigma_T) = \int d\mathbf{S}\; p(\mathbf{S}|a, \Psi) \prod_{t=1}^{T} p(\Sigma_t|b, \mathbf{S})$$

which yields

$$p(\Sigma_1, \ldots, \Sigma_T) = \frac{\Gamma_N(\frac{1}{2}(a+bT))|\Psi|^{a/2} \prod_{t=1}^{T} |\Sigma_t|^{(b-N-1)/2}}{\Gamma_N(a/2)(\Gamma_N(b/2))^T \left|\Psi + \sum_{t=1}^{T} \Sigma_t\right|^{(a+bT)/2}}$$

where $\Gamma_N(\cdot)$ is the multivariate gamma function,

$$\Gamma_N(x) = \pi^{N(N-1)/4} \prod_{j=1}^{N} \Gamma(x + (1-j)/2)$$

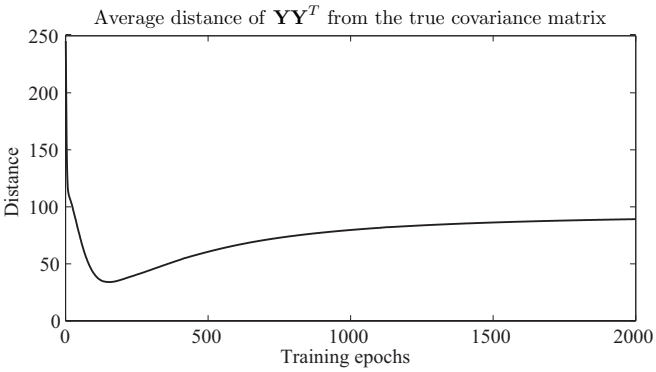

Figure 1: Average distance between the true covariance matrix and the covariance matrix learned by the neural network.

and $\Gamma(x)$ is the generalized factorial function (Boas, 1983). This is the ratio of the normalization constants for a Wishart and an inverse-Wishart distribution, due to conjugacy.

Neural network learning

The linear neural network is defined by two matrices: a $N \times K$ matrix $\mathbf{Y}$ that maps the properties into the latent space and a $K \times M$ matrix $\mathbf{Z}$ that maps the latent space to the objects. We trained the neural network by gradient descent on error, with

$$\Delta \mathbf{y} \quad = \quad \eta(\mathbf{x} - \mathbf{yZ})\mathbf{Z}^T \tag{6}$$

and Equation 4 as the weight update rules. K was set to one more than the rank of the object matrix $\mathbf{X}$. The weights were initialized to normally distributed random values with mean 0 and variance 0.05. We used a learning rate η of 0.0025 and 2000 training epochs (full passes through the data), which were determined by pilot simulations. At each possible stopping point (epoch), we recorded $\mathbf{YY}^T$. Figure 1 shows the average distance between $\mathbf{YY}^T$ and the true covariance matrix as a function of epoch, which initially decreases and then rises again due to overfitting.

Priors and Bayesian inference

We considered eight different prior distributions, requiring us to use three different algorithms for Bayesian inference.

Wishart prior. The first Bayesian model used a Wishart prior with covariance center $\mathbf{I}$ and degrees of freedom $b = 1000$. Unfortunately the Wishart is not conjugate to the multivariate Gaussian, so we found an estimate of the covariance matrix under this prior using stochastic search with simulated annealing. The state of the search (a covariance matrix) was initialized to a random draw from the posterior distribution using an inverse-Wishart prior (for details, see Gelman, Carlin, Stern, & Rubin, 1995). A new proposed state was then drawn from a Wishart distribution centered at the current state with $b + N$ degrees of freedom. A Metropolis-Hastings acceptance rule was used to decide whether to replace the current state with the proposed state, based on the product of two ratios of their (unnormalized) posterior probabilities and the probability of generating the proposed state from the current state and vice versa (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953). This probability was annealed by raising the probability to the power $1/\tau$, with τ decreasing according to a logarithmic schedule.

Graph grammar priors. We used four priors based on graph grammars, defining distributions on graphs that correspond to trees, grids, chains, and partitions (Nagl, 1986; Kemp & Tenenbaum, 2008). These random graph grammars are generative processes that start with a single node and then replace a random node in the current graph with two nodes J times, where $J \sim \text{Geom}(\theta)$. Different graph structures result from using different rules for connecting the parents and children of the old node to the new nodes (for the tree grammar, there is also a latent node that cannot contain any objects), and different rules for connecting the new nodes result in different generated graph structures.[4] Afterwards, the objects are assigned to nodes uniformly at random (except not to latent nodes). For example, if the rule for node replacement does not create any edges, then the random graph grammar generates random partitions of the objects.

To convert the graph to a covariance matrix, we follow Kemp and Tenenbaum (2008) by first forming an "entity" graph containing $N + L$ nodes, where the first N nodes represent each object and are only directly connected with an edge to their assigned node. Second, we complete the "entity" graph by connecting the last L nodes to each other according to the result of the previous graph replacement process. Next, we form a $N + L \times N + L$ adjacency matrix $\mathbf{W}$, where $1/w_{ij} \sim \text{Exp}(\beta)$ if there is an edge between nodes i and j (representing how close nodes i and j are). Otherwise, $w_{ij} = 0$. This specifies a $N + L \times N + L$ covariance matrix for the multivariate Gaussian distribution over the latent and observed variables, $\left(\mathbf{E} - \mathbf{W} + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1}$ where $\mathbf{E}$ is a $N + K \times N + K$ diagonal matrix with $e_{ii} = \sum_j w_{ij}$ and $\mathbf{I}$ is the $N + K \times N + K$ identity matrix. The hidden nodes can be marginalized out analytically, resulting in the N objects being normally distributed with covariance matrix given by the first $N \times N$ elements of the original covariance matrix.[5] Bayesian inference was performed with code from `http://charleskemp.com`, which uses stochastic search to find an estimated maximum *a posteriori* covariance matrix for a given set of data.[6]

Erdös-Rényi priors. In addition to the four random graph generators from Kemp and Tenenbaum (2008), we used a standard random graph generator: the Erdös-Rényi random graph (Erdös & Rényi, 1959). Each object is represented by a node. Unlike the node replacement grammars, we gener-

[4] For simplicity, we assumed the graph structures are undirected.

[5] It is important to note that this is not equivalent to the first $N \times N$ elements of the inverse covariance matrix.

[6] The parameters were set to $\beta = 0.4$ (edge length parameter), $\sigma^2 = 0.4$ (covariance matrix regularization parameter), and $\theta = 1 - e^{-3}$ (simplicity bias), which are similar to the values used by Kemp and Tenenbaum (2008). We used the "45" speed setting.

ate random graphs by directly connect pairs of objects with an edge with probability p. Once the graph is generated, the implied covariance matrix is found by the same procedure as before (except we do not need to perform the additional marginalization step as the initial covariance matrix is already $N \times N$). We considered priors corresponding to $p \in \{0.1, 0.5, 0.9\}$. Covariance matrices with these priors were estimated using stochastic search by simulated annealing. The covariance matrix was initialized to a random Erdös-Rényi covariance matrix and proposals were generated from the current state by removing or deleting a random number of edges (such that the number of edges in the proposals were binomially distributed). The search procedure and annealing schedule were otherwise the same as for the Wishart prior.

The distance between covariance matrices

To analyze the results produced by the neural network and Bayesian models, we needed a measure of the similarity of two matrices. We used a distance metric between positive definite matrices (valid covariance matrices) defined by Förstner and Moonen (1999)

$$d(\Sigma_1, \Sigma_2) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(\Sigma_1, \Sigma_2)}, \tag{7}$$

where $\lambda_i(\Sigma_1, \Sigma_2)$ are the generalized eigenvalues of Σ_1 and Σ_2, being the roots of $|\lambda\Sigma_1 - \Sigma_2| = 0$. When computing these distances, we used the best stopping point for the neural network (the one resulting in minimal distance). Looking across epochs, we found the value of $\mathbf{YY}^T$ with the minimal distance to the true covariance matrix and to the eight covariance matrices estimated by Bayesian models with different priors.

Simulation procedure and results

For each prior, we generated 101 data sets that each consisted of $T = 100$ covariance matrices. From each matrix, we sampled the values of $M = 100$ features for $N = 10$ objects. We then computed the marginal probability of the covariance matrices generated by each prior under the assumption they were drawn from a Wishart distribution, with the median result shown in the top row of Table 1. As expected, the Wishart prior was the most compatible with a Wishart distribution. The discrete priors produced results that were reasonably consistent with the Wishart distribution, while the the Erdös-Rényi generative processes produced results that were poorly characterized as Wishart. We used the data set with the median Wishart value for the subsequent analyses.

Next, we trained neural networks on the object set $\mathbf{X}$ generated from each covariance matrix sampled from each of the eight priors, and computed the distance between $\mathbf{YY}^T$ and the true covariance matrix. The results are shown in the second row of Table 1. Performance was statistically significantly better when the true covariance matrices were drawn from the Wishart, consistent with our mathematical analysis.

Finally, we found Bayesian estimates of the covariance matrix for each object set $\mathbf{X}$ using all eight priors. Stochastic search was run for 20000 iterations in each case. We computed the distance between $\mathbf{YY}^T$ and the Bayesian estimates for each object set, then averaged this quantity across all object sets. The results are shown in the third row of Table 1. As predicted, we found a negative correlation between the distance between estimates and the extent to which the corresponding prior is consistent with a Wishart distribution (as reflected by the marginal probabilities in the first row of Table 1) with $r = -0.92$ and $r = -0.83$ for Pearson's product-moment and Spearman's rank-order correlation, respectively.[7] A scatterplot showing the relationship between these two quantities is shown in Figure 2.

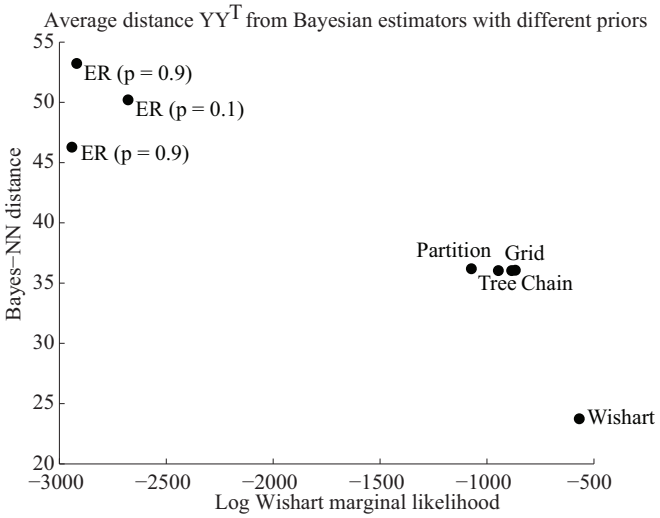

Figure 2: Average (smallest possible) distance of $\mathbf{YY}^T$ from the Bayesian estimates of the covariance matrix, plotted as a function of the logarithm of the Wishart marginal likelihood for the corresponding prior.

The variation in how well the neural network approximated the Bayesian estimates with different prior distributions is informative about the inductive biases of neural networks and structured probabilistic models. The neural network was closest in performance to Bayesian inference with a Wishart prior, which is purely continuous. All priors based on discrete structure, in the form of an underlying graph, resulted in statistically significantly worse performance. Within these discrete priors, those based on graph grammars were better approximated than the Erdös-Rényi priors. This pattern of results is interesting from the perspective of the debate between probabilistic and connectionist accounts of property induction, which has focused on discriminating the predictions of probabilistic models using representations based on graph grammars from neural networks. Our results suggest that this may be harder than discriminating probabilistic models that assume arbitrary discrete structure, as in the Erdös-Rényi priors, from neural networks.

[7] We confirmed that this correlation could not be fully explained by the norm of the matrices, but plan on running further simulations to rule out other possible alternative explanations for our results.

Table 1: Properties of different priors and comparison of gradient descent and Bayesian learning

		Graph grammar priors				Erdös-Rényi priors		
	Wishart	Grid	Chain	Tree	Partition	$p = 0.1$	$p = 0.5$	$p = 0.9$
Marginal probability under Wishart	-567.87	-867.68	-884.95	-946.22	-1073.10	-2678.04	-2940.98	-2919.44
Distance of $\mathbf{YY}^T$ from true covariance	14.15^a	33.31^b	34.18^b	32.36^b	33.97^b	33.93^b	31.73^b	33.35^b
Distance of $\mathbf{YY}^T$ from Bayesian estimate	23.53^a	36.04^b	36.07^b	36.03^b	36.19^b	50.21^c	46.29^d	53.23^e

Note: In each row, different superscripts indicate statistically significant differences in scores (Bonferroni $p < .05$).

Discussion

Our analysis of the relationship between probabilistic and connectionist models in the context of property induction has produced several interesting results. First, the generalization performance of a probabilistic model with a discrete representation can be approximated by an appropriately configured linear neural network with continuous representations. Second, training such a network by gradient descent with early stopping is similar to performing Bayesian inference over covariance matrices with a Wishart prior. Finally, prior distributions that assume discrete structure vary in the extent to which they resemble a Wishart prior, and this variation predicts how well Bayesian inference using those prior distributions is approximated by a neural network. However, all prior distributions using discrete structure that we considered resulted in worse approximations than that given with a Wishart prior.

There are limitations in the analyses presented here that we hope to address in future work. As noted earlier, the assumption of linearity in the neural network deviates from standard practice in connectionist modeling. While we do not expect that this will substantially change our results (given that early stopping enforces small weights, effects of the nonlinearity should be minimized), further simulations should be conducted to confirm that this is the case. We would also like to explore more sophisticated learning algorithms, such as cascade correlation (Fahlman & Lebiere, 1990), which may result in different inductive biases.

Returning to the questions that motivated our investigation, our results provide a mixed set of answers as to the potential for neural networks to be viewed as a continuous approximation to Bayesian inference over discrete representations. While specific neural networks can always be constructed that emulate the generalization performance of probabilistic models using discrete representations and the inductive biases of neural networks can be expressed in a form that is consistent with Bayesian inference, these inductive biases are quite different from those of Bayesian models using priors defined on discrete objects. Our results suggest that there is room to empirically separate these two approaches, and that identifying neural systems that can approximate arbitrary Bayesian models may require going beyond simple neural networks that use general-purpose learning algorithms.

Acknowledgments. We thank Noah Goodman, Surya Ganguli, and Jay McClelland for discussions and grant number FA-9550-10-1-0232 from the Air Force Office of Scientific Research and a fellowship from the Fonds de Recherche du Québec to VGB for funding.

References

Boas, M. L. (1983). *Mathematical methods in the physical sciences* (2nd ed.). New York: Wiley.

Erdös, P., & Rényi, A. (1959). On random graphs, I. *Publicationes Mathematicae*, *6*, 290-297.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*.

Fleming, H. E. (1990). Equivalence of regularization and truncated iteration in the solution of ill-posed image reconstruction problems. *Linear Algebra and its Applications*, *130*, 133-150.

Förstner, W., & Moonen, B. (1999). A metric for covariance matrices. In F. Krumm & V. S. Schwarze (Eds.), *Qua vadis geodisa...? festschrift for Erik W. Grafarend on the occasion of his 60th birthday* (p. 113-128). Stuttgart, Germany: Stuttgart University.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. New York: Chapman & Hall.

Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., & Tenenbaum, J. B. (2010). Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, *14*(8), 357-364.

Kemp, C., & Tenenbaum, J. B. (2008). The discovery of structural form. *Proceedings of the National Academy of Sciences, USA*, *105*, 10687-10692.

Kemp, C., & Tenenbaum, J. B. (2009). Structured statistical models of inductive reasoning. *Psychological Review*, *116*(1), 20-58.

Marr, D. (1982). *Vision*. San Francisco, CA: W. H. Freeman.

McClelland, J. L., Botvinick, M. M., Noelle, D. C., Plaut, D. C., Rogers, T. T., Seidenberg, M. S., et al. (2010). Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences*, *14*(8), 348-356.

Metropolis, A. W., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087-1092.

Muirhead, R. J. (1982). *Aspects of multivariate statistical theory*. New York: John Wiley & Sons.

Nagl, M. (1986). Set theoretic approaches to graph grammars. In *Proceedings of the 3rd international workshop on graph-grammars and their application to computer science* (p. 41-54). London, UK: Springer.

Rogers, T., & McClelland, J. (2004). *Semantic cognition: A parallel distributed processing approach*. Cambridge, MA: MIT Press.

Santos, R. J. (1996). Equivalence of regularization and truncated iteration for general ill-posed problems. *Linear Algebra and its Applications*, *236*, 25-33.