Approximating Bayesian inference with a sparse distributed memory system

Joshua T. Abbott (joshua.abbott@berkeley.edu)
Jessica B. Hamrick (jhamrick@berkeley.edu)
Thomas L. Griffiths (tom_griffiths@berkeley.edu)
Department of Psychology, University of California, Berkeley, CA 94720 USA

Abstract

Probabilistic models of cognition have enjoyed recent success in explaining how people make inductive inferences. Yet, the difficult computations over structured representations that are often required by these models seem incompatible with the continuous and distributed nature of human minds. To reconcile this issue, and to understand the implications of constraints on probabilistic models, we take the approach of formalizing the mechanisms by which cognitive and neural processes could approximate Bayesian inference. Specifically, we show that an associative memory system using sparse, distributed representations can be reinterpreted as an importance sampler, a Monte Carlo method of approximating Bayesian inference. This capacity is illustrated through two case studies: a simple letter reconstruction task, and the classic problem of property induction. Broadly, our work demonstrates that probabilistic models can be implemented in a practical, distributed manner, and helps bridge the gap between algorithmic- and computational-level models of cognition.

Keywords: Bayesian inference, importance sampling, rational process models, associative memory models, sparse distributed memory

Introduction

Probabilistic models of cognition can be used to explain the complex inductive inferences people make every day, such as identifying the content of images or learning new concepts from limited evidence (Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010; Tenenbaum, Kemp, Griffiths, & Goodman, 2011). However, these models are typically formulated at what Marr (1982) called the *computational* level, focusing on the abstract problems people have to solve and their ideal solutions. As a result, they explain *why* people behave the way they do, rather than *how* cognitive and neural processes support these behaviors. This approach is thus quite different from previous work on modeling human cognition, which focused on Marr's *algorithmic* and *implementation* levels, and has been criticized because it seems to imply that human minds and brains need to solve intractable computational problems and use structured representations (Gigerenzer & Todd, 1999; McClelland et al., 2010).

Understanding the actual commitments that computational-level accounts of human cognition based on probabilistic models make at the algorithmic and implementation level requires considering how these levels of analysis could be bridged (Griffiths, Vul, & Sanborn, 2012). Identifying specific cognitive algorithms and neural architectures that can approximate Bayesian inference is a key step towards knowing whether it really poses an intractable problem for human minds, or whether structured representations need to be used to implement models that involve structured probability distributions. In this paper, we take on this challenge by showing that an associative memory using sparse distributed representations can be used to approximate Bayesian inference, producing behavior consistent with a structured statistical model while using distributed representations of the kind normally associated with artificial neural networks.

The associative memory that we use to approximate Bayesian inference implements a Monte Carlo algorithm known as *importance sampling*. Previous work has shown that this algorithm can be implemented in a common psychological process model – an exemplar model (Shi, Griffiths, Feldman, & Sanborn, 2010). Shi and Griffiths (2009) further demonstrated that importance sampling can be implemented with a radial basis function neural network. However, this neural network used a localist representation, in which each hypothesis considered by the model had to be represented with a single neuron – a "grandmother cell." While this might be plausible for modeling aspects of perception in which a wide range of neurons prefer specific stimuli, it becomes less plausible for modeling complex cognitive tasks in which hypotheses correspond to structured representations. For example, having separate neurons for each concept or causal structure we consider seems implausible.

We demonstrate that an associative memory that uses distributed representations – specifically, *sparse distributed memory* (SDM) (Kanerva, 1988, 1993) – can be used to approximate Bayesian inference through importance sampling. The underlying idea is simple: we use the associative memory to store and retrieve exemplars, allowing us to build on the equivalence between exemplar models and importance sampling. The critical advance is that this is done using distributed representations, meaning that arbitrary hypotheses can be represented, and arbitrary distributions of exemplars encoded by a single architecture. We show that the SDM naturally implements one class of Bayesian models, and explain how to generalize it to implement a broader range of models.

The plan of the paper is as follows. First, we give a brief overview of performing Bayesian inference with importance sampling and summarize how the sparse distributed memory system is implemented. Next, we formalize how importance sampling can be performed using a SDM. We provide two case studies drawn from existing literature in which we use the SDM to approximate existing Bayesian models. The first case study is a simple example involving reconstructing English letters from noisy inputs, and the second is a more sophisticated model of property induction. We conclude the paper with a discussion of implications and future directions.

Background

Our results depend on two important sets of mathematical ideas: approximating Bayesian inference by importance sampling, and sparse distributed memories. We introduce these ideas in turn.

Bayesian inference and importance sampling

Probabilistic models of cognition provide rational solutions to problems of inductive inference, where probability distributions represent degrees of belief and are updated as more data becomes available. Beliefs are updated by applying Bayes' rule, which says that the *posterior* probability of a hypothesis, h, given observed data, d, is proportional to the probability of observing d if h were the correct hypothesis (known as the *likelihood*) multiplied by the *prior* probability of that hypothesis:

$$p(h|d) = \frac{p(d|h)p(h)}{\int_{\mathcal{H}} p(d|h)p(h)\,dh} \tag{1}$$

Unfortunately, computing the integral in the denominator is computationally expensive and often intractable. This has resulted in the development of many algorithms for approximating Bayesian inference.

For the sake of illustration, consider the case in which we have noisy observations x of a stimulus x^*. To recover the value of x^*, we use Bayes' rule to compute the posterior distribution over x^*:

$$p(x^*|x) = \frac{p(x|x^*)p(x^*)}{\int_{x^*} p(x|x^*)p(x^*)dx^*} \tag{2}$$

It is often desirable to compute the expectation of the posterior distribution over some function $f(x^*)$:

$$E[f(x^*)|x] = \int f(x^*)p(x^*|x)\,dx^* \tag{3}$$

where the choice of $f(x^*)$ depends on the task. However, evaluating this expectation still requires computing the full posterior distribution.

To approximate expectations over posterior distributions, we can use a Monte Carlo method known as *importance sampling* (see, e.g., Neal, 1993), in which a finite set of samples are used to represent the posterior. These samples are drawn from a surrogate distribution $q(x^*)$ and assigned weights proportional to the ratio $p(x^*|x)/q(x^*)$:

$$E[f(x^*)|x] = \int f(x^*)\frac{p(x^*|x)}{q(x^*)}q(x^*)\,dx^* \tag{4}$$

Given a set of K samples $\{x_k^*\}$ distributed according to $q(x^*)$, this integral can be approximated by:

$$E[f(x^*)|x] \approx \frac{1}{K}\sum_{k=1}^{K} f(x_k^*)\frac{p(x_k^*|x)}{q(x_k^*)} \tag{5}$$

with the approximation becoming more precise as K becomes larger.

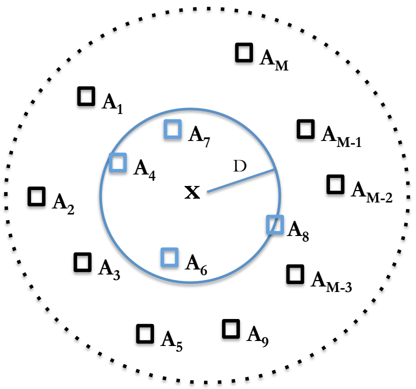

Figure 1: An illustration of the basic read and write operations over SDMs. The outer dotted line represents the space of 2^N possible addresses while the squares with labels $\mathbf{A_m}$ represent the M sampled hard addresses used for storage. The address being requested for operation is the $\mathbf{x}$ in the center of the blue circle of radius D. The hard addresses selected for operating correspond to the blue squares within the Hamming radius of $\mathbf{x}$.

One possible choice for $q(x^*)$ is the prior, $p(x^*)$, which yields importance weights proportional to the likelihood, $p(x|x^*)$. Formally,

$$\begin{aligned} E[f(x^*)|x] &\approx \frac{1}{K}\sum_{k=1}^{K} f(x_k^*)\frac{p(x_k^*|x)}{p(x_k^*)} \\ &= \frac{1}{K}\sum_{k=1}^{K} f(x_k^*)\frac{p(x|x_k^*)p(x_k^*)}{p(x_k^*)p(x)} \\ &= \alpha(x)\sum_{k=1}^{K} f(x_k^*)p(x|x_k^*) \end{aligned} \tag{6}$$

where we assume x_k^* is drawn from the prior, $p(x^*)$, and $\alpha(x)$ is a constant of proportionality that depends only on x. Returning to the general case of data d and hypotheses h, we can use the same approximation to compute the expectation of a function $f(h)$ given observed data, with

$$E[f(h)|d] \approx \alpha(d)\sum_{k=1}^{K} f(h_k)p(d|h_k) \tag{7}$$

where h_k is drawn from the prior $p(h)$, and $\alpha(d)$ is a constant of proportionality that depends only on d.

Sparse distributed memory

Sparse distributed memory (SDM) was developed as an algorithmic-level model of human memory, designed to encapsulate the notion that distances between concepts in memory correspond to distances between points in high-dimensional space (Kanerva, 1988, 1993). In particular, it has a natural interpretation as an artificial neural network that uses distributed representations.

SDMs preserve distances between items in memory by storing them in a distributed manner. Assume that items enter memory as two strings of bits, with N bits indicating a location and L bits indicating its content. A conventional approach would be to sequentially enumerate the set of 2^N locations (more technically, *addresses*), storing items by setting the content bits at the appropriate address in turn. In contrast, a SDM samples $M \gg N$ addresses $\mathbf{a}_j$ to use as registers from the space of 2^N possible addresses. Items are then stored by changing the bits that encode the content associated with multiple addresses, according to how close those addresses are to the target location. The algorithms for writing to and reading from a SDM are given below and are illustrated in Figure 1.

Writing A SDM stores an L-bit vector, $\mathbf{z}$, associated with an N-bit location $\mathbf{x}^*$ by storing the pattern at multiple addresses $\mathbf{a}_j$ near $\mathbf{x}^*$. Since the set of M available addresses does not enumerate the total space of 2^N, there may be very few addresses near $\mathbf{x}^*$. Consequently, $\mathbf{z}$ is written to all addresses $\mathbf{a}_j$ that are within a Hamming distance D of $\mathbf{x}^*$ (i.e. those $\mathbf{a}_j$ that differ from $\mathbf{x}^*$ in D or fewer bits). The contents of these selected addresses are modified to store the pattern $\mathbf{z}$ such that each bit in the contents is increased or decreased by 1 depending on whether or not that bit in $\mathbf{z}$ is a 1 or 0, respectively.

A SDM can be constructed as a neural network with N units in the input layer, a hidden layer with M units for each sampled address, and an output layer with L units. The weights between the input and hidden layer correspond to the $M \times N$ matrix $\mathbf{A} = [\mathbf{a}_1; \mathbf{a}_2; \ldots; \mathbf{a}_M]$ of hard addresses, and the weights between the hidden and output layer correspond to the $M \times L$ matrix $\mathbf{C}$ of contents stored at each address. The rule for writing $\mathbf{z}$ to memory address $\mathbf{x}^*$ is expressed as:

$$\mathbf{y} = \Theta_D(\mathbf{A}\mathbf{x}^*) \tag{8}$$

$$\mathbf{C} = \mathbf{C} + \mathbf{z}\mathbf{y} \tag{9}$$

where Θ_D is a function that converts its argument zeros and ones, with $\Theta_D(w) = 1$ if $\frac{1}{2}(w - N) \leq D$ and 0 otherwise. $\mathbf{y}$ is thus a *selection vector* that picks out a particular set of addresses. The expected number of addresses selected in $\mathbf{y}$ is a function of N, M, and D:

$$E[\mathbf{y}^T\mathbf{y}] = \frac{M}{2^N} \sum_{d=1}^{D} \binom{N}{d} \tag{10}$$

Reading To read a pattern out of memory from address $\mathbf{x}$, the SDM again computes a M-bit selection vector $\mathbf{y}$ of addresses within Hamming distance D of $\mathbf{x}$. The contents of each address selected by this vector are summed, resulting in a vector $\widehat{\mathbf{z}}$ of length L. The rule for reading $\widehat{\mathbf{z}}$ from memory address $\mathbf{x}$ is expressed as:

$$\mathbf{y} = \Theta_D(\mathbf{A}\mathbf{x}) \tag{11}$$

$$\widehat{\mathbf{z}} = \mathbf{C}^T\mathbf{y} \tag{12}$$

The output $\widehat{\mathbf{z}}$ can then be passed through a non-linearity to return a binary vector if desired.

SDMs as importance samplers

Previous work formalizing a probabilistic interpretation of SDMs (Anderson, 1989) lays the groundwork for using SDMs to perform Bayesian inference. Here, we show that the output of SDMs approximates the expectation of a function $f(\mathbf{x}^*)$ over the posterior distribution $p(\mathbf{x}^*|\mathbf{x})$ by linking its behavior to that of the importance sampler in Equation 6.

Writing Let $w(\mathbf{a}_j, \mathbf{x}^*)$ be the probability of writing to address $\mathbf{a}_j$ given an input address $\mathbf{x}^*$. In the standard SDM, this is 1 if the Hamming distance between $\mathbf{a}_j$ and $\mathbf{x}^*$ is less than or equal to D and 0 otherwise. In the limit, the number of addresses increases to the point where we will always be able to write to exactly $\mathbf{x}^*$ (i.e., to set $D = 0$). Thus, this writing probability must satisfy the following constraint:

$$\lim_{M \to 2^N} w(\mathbf{a}_j, \mathbf{x}^*) = \prod_{i=1}^{N} \delta(x_i^* - a_{ji}) \tag{13}$$

After writing K (address, data) pairs $(\mathbf{x}_k^*, \mathbf{z}_k)$, the value of the counter associated with bit i of address $\mathbf{a}_j$ will be:

$$\mathbf{c}_j = \sum_{k=1}^{K} w(\mathbf{a}_j, \mathbf{x}_k^*)\,\mathbf{z}_k \tag{14}$$

Reading We are given a location $\mathbf{x}$, which as before is a corrupted version of $\mathbf{x}^*$. Let $r(\mathbf{x}, \mathbf{a}_j)$ be the probability that we read from address $\mathbf{a}_j$ given input $\mathbf{x}$. In the standard SDM, this is 1 if the Hamming distance between $\mathbf{a}_j$ and $\mathbf{x}$ is less than or equal to D and 0 otherwise. Then, the output of the SDM for a particular set of addresses $\mathbf{A}$ is:

$$\widehat{\mathbf{z}} = \sum_{j=1}^{M} \mathbf{c}_j\, r(\mathbf{x}, \mathbf{a}_j) \tag{15}$$

where $\mathbf{c}_j$ is defined in Equation 14.

To see how this output behaves for any SDM, we consider the expected value of $\widehat{\mathbf{z}}$ over sampled sets of addresses $\mathbf{A}$. We first substitute Equation 14 into Equation 15 and simplify according to linearity of expectation:

$$E_{\mathbf{A}}[\widehat{\mathbf{z}}|\mathbf{x}] = E_{\mathbf{A}}\left[\sum_{j=1}^{M}\left(\sum_{k=1}^{K} w(\mathbf{a}_j, \mathbf{x}_k^*)\mathbf{z}_k\right) r(\mathbf{x}, \mathbf{a}_j)\right] \tag{16}$$

$$= \sum_{k=1}^{K} \mathbf{z}_k \cdot E_{\mathbf{A}}\left[\sum_{j=1}^{M} w(\mathbf{a}_j, \mathbf{x}_k^*) r(\mathbf{x}, \mathbf{a}_j)\right] \tag{17}$$

As our address space grows larger (as in Equation 13), this approaches:

$$\lim_{M \to 2^N} E_{\mathbf{A}}[\widehat{\mathbf{z}}|\mathbf{x}] = \sum_{k=1}^{K} \mathbf{z}_k \int_{\mathbf{a}_j} \delta(\mathbf{x}_k^* - \mathbf{a}_j) r(\mathbf{x}, \mathbf{a}_j)\, \mathrm{d}\mathbf{a}_j \tag{18}$$

Thus, in the limit, the expected value of $\widehat{\mathbf{z}}$ read from the SDM will be:

$$E_{\mathbf{A}}[\widehat{\mathbf{z}}|\mathbf{x}] = \sum_{k=1}^{K} \mathbf{z}_k\, r(\mathbf{x}, \mathbf{x}_k^*) \tag{19}$$

Comparing Equation 19 to Equation 6 yields our main result: SDMs can perform importance sampling by defining a reading density proportional to a likelihood function, approximating the posterior expectation of the items stored in memory. More formally, the expectation of $\widehat{\mathbf{z}}$ given $\mathbf{x}$ is proportional to the output of the importance sampling approximation of the expectation of $f(\mathbf{x}^*)$ with respect to $p(\mathbf{x}|\mathbf{x}^*)$:

$$E_{\mathbf{A}}[\widehat{\mathbf{z}}|\mathbf{x}] \propto \sum_{k=1}^{K} f(\mathbf{x}_k^*) p(\mathbf{x}|\mathbf{x}^*) \qquad (20)$$

provided $\mathbf{z}_k \propto f(\mathbf{x}_k^*)$ and $r(\mathbf{x}, \mathbf{x}_k^*) \propto p(\mathbf{x}|\mathbf{x}^*)$.

The utility of this result is limited with the standard formulation of the SDM, as it only holds in the limit where the address size becomes large and D becomes small, meaning that $r(\mathbf{x}, \mathbf{x}^*)$ becomes a delta function. Instead, we can consider generalizations in which we use different values of D for reading and writing (D_r and D_w, respectively), or where we choose $r(\mathbf{x}, \mathbf{x}^*)$ more freely. These modifications allow us to approximate a variety of Bayesian models using SDMs.

We make a further note, which is that in most practical applications, the address space will not approach the limit (i.e. $M \ll 2^N$). For any sampled set of addresses, we would still expect the value of $\widehat{\mathbf{z}}$ to be near the posterior mean $E_k[f(\mathbf{x}^*)|\mathbf{x}]$, but we cannot make any statement about *how* close. We leave it as an area for future work to place analytic bounds on the accuracy of the SDM's approximation. For the case studies we present here, we estimate the variance of the SDM using Monte Carlo approximations.

In the following two sections, we evaluate SDMs as a scheme for approximating Bayesian inference in two tasks: one where the Bayesian likelihood matches the standard SDM read rule, and one where the SDM read rule is adjusted to match the Bayesian likelihood. The second case further illustrates how SDMs can be applied to more general problems of Bayesian inference that go beyond simply removing noise from a stimulus.

Letter reconstruction

As a simple illustration of approximating Bayesian inference with a SDM, we consider the task of recovering images of English letters, $\mathbf{x}^*$, from noisy observations $\mathbf{x}$. To solve this problem, we set up a Bayesian model loosely based on that presented in Rumelhart and Siple (1974). Each letter of the alphabet is encoded as a binary feature vector of length $N = 14$ based on the Rumelhart-Siple font template (Figure 2).

SDM approximation

In the Bayesian model, we wish to reconstruct the original letters $\mathbf{x}^*$ from the exemplars $\mathbf{x}$ by computing the mean of the posterior distribution over original letters, $p(\mathbf{x}^*|\mathbf{x})$, i.e. $f(\mathbf{x}^*) = \mathbf{x}^*$. Each of the original letters is associated with a prior probability, $p(\mathbf{x}^*)$, set to be the relative letter frequency of English text (Lewand, 2000). Following the generative model, a noisy image $\mathbf{x}$ is produced from the original letter $\mathbf{x}^*$ via a noise process in which at most B bits are flipped (uniformly). Given a noisy image vector $\mathbf{x}$, we define the likelihood $p(\mathbf{x}|\mathbf{x}^*)$ to be uniformly distributed over the number of possible bit strings in a hypersphere of radius D_r.

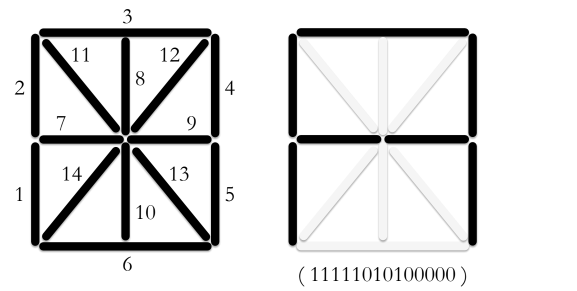

Figure 2: The Rumelhart-Siple font feature map and the Rumelhart-Siple representation for the letter "A" along with its binary feature pattern.

In the SDM, we sample exemplars $\mathbf{x}^*$ of the original letters from the prior $p(\mathbf{x}^*)$ and write them to the SDM at inputs $\mathbf{x}^*$ (i.e., $\mathbf{z} = \mathbf{x}^*$). The likelihood defined for the Bayesian model is naturally compatible with the standard read rule of a SDM, where we only consider hypotheses $\mathbf{x}^*$ which are within Hamming distance D_r of $\mathbf{x}$. Thus, we can set the read function $r(\mathbf{x}^*, \mathbf{x})$ of the SDM to be a variation of this likelihood:

$$p(\mathbf{x}|\mathbf{x}^*) \approx r(\mathbf{x}, \mathbf{x}^*) = \begin{cases} \left[\sum_{d=1}^{D_r} \binom{N-d+1}{d}\right]^{-1} & |\mathbf{x} - \mathbf{x}^*| \leq D_r \\ 0 & \text{otherwise} \end{cases}$$

The corrupted images $\mathbf{x}$ are the inputs that we attempt to read from and D_r is the SDM's read radius; we additionally define D_w to be the write radius. The intuition is that we write the original letters $\mathbf{z} = \mathbf{x}^*$ to input $\mathbf{x}^*$, and read from $\mathbf{x}$ outputs $\widehat{\mathbf{z}}$ which are similar to the mean of the Bayesian posterior.

Analysis

To evaluate how well the SDM approximates the posterior mean, we sampled 1000 exemplars from the prior distribution $p(\mathbf{x}^*)$. We then created three images $\mathbf{x}$ for each letter in the alphabet, each with two bits of corruption, yielding a total of 72 test images. We repeated this simulation – sampling from the prior and generating test images – 20 times for each SDM with parameters D_r, D_w, and M, where each simulation used a different set of sampled addresses A.

We determined the appropriate settings of the read and write radii, D_r and D_w, by considering the constraints imposed by the SDM and the specific problem of letter reconstruction. The mean Hamming distance between pairs of letters was 5.4615, indicating that the read radius must lie somewhere between 2 (the amount of corruption) and 5. Choosing a radius outside these bounds would have the effect of returning an inaccurate expectation, either because the noise was greater than the signal ($D_r < 2$) or because too many hypotheses were considered ($D_r > 5$). So, we chose $D_r = 2$, thus ensuring that the true $\mathbf{x}^*$ would always fall within this radius, and also minimizing the number of incorrect hypotheses considered. For the write radius, we considered three different values, $D_w = \{0, 1, 2\}$.

We stored all 1000 letters in each SDM, varying the number of hard addresses, M, among eight evenly spaced values

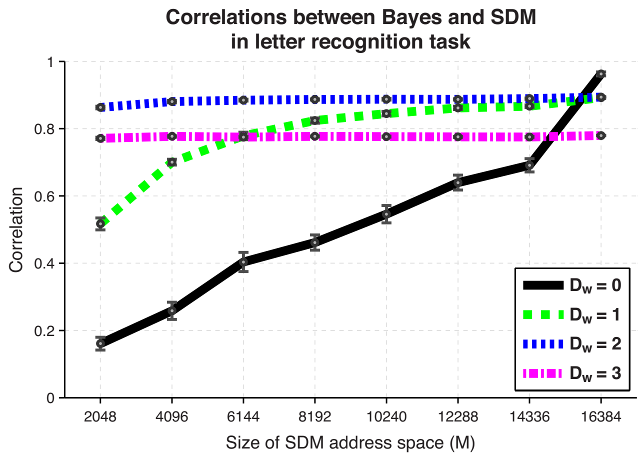

Figure 3: The average correlations between the Bayesian posterior mean and pre-thresholded SDM outputs for the task of recovering a Rumelhart-Siple letter from a noisy observation. $D_r = 2$ for each of the 4 values of D_w presented.

between 2048 and $2^N = 16384$. We then read the value $\widehat{\mathbf{z}}$ for each corrupted input $\mathbf{x}$, for different address spaces A. For the Bayesian model, we analytically calculated the posterior mean by evaluating the full posterior for each $\mathbf{x}^*$ and then calculating the average $\mathbf{x}^*$ weighed by its posterior probability. We then computed the average correlation between the SDM values of $\widehat{\mathbf{z}}$ and the Bayesian posterior mean across sampled addresses A. These results are displayed in Figure 3.

The SDMs with $D_w = 0$ performed similarly to the Bayesian model only when $M = 2^N$ ($\rho = 0.9628$, $se = 0.0058$), reflecting the intuition that it's highly unlikely to find an exact address from a random sample of 2^N. Conversely, the SDMs with $D_w = 2$ and $D_w = 3$ had near-constant correlations with the Bayesian model ($\rho \approx 0.88$ for $D_w = 2$ and $\rho \approx 0.79$ for $D_w = 3$), regardless of the size of the hard address space. This behavior was also in line with our expectations: with $D_w = 2$ and $D_w = 3$, the probability of having no hard addresses within D_w of $\mathbf{x}^*$ is extremely low for $M = 2048$; this probability only decreases as M increases.

In summary, these results show SDMs can naturally approximate Bayesian models of noisy stimulus reconstruction. Here, we set the likelihood function to follow the standard SDM read rule. In the next section we consider a more general example of Bayesian inference and explore the consequences of adjusting the SDM read rule to match the likelihood in question.

Property induction

If you are told that a horse has a particular property, protein K, what other animals do you think share this protein? Questions of this type are considered problems of property induction, where one or more categories in a domain are observed to have some novel property and the task is to determine how the property is distributed over the domain. For our analyses we explore the category-based induction task introduced by Osherson, Smith, Wilkie, Lopez, and Shafir (1990), where judgments are made as to whether certain animals can get a disease knowing other animals that can catch it.

SDM approximation

We solve this problem with a Bayesian model of property induction based on Kemp and Tenenbaum (2009). Here, we observe a set of examples d of a concept C (known to have property K) and we aim to calculate $p(y \in C|d)$, the probability that object y is also a member of C. Thus, averaging over all possible concepts, we compute:

$$p(y \in C|d) = \sum_{h \in \mathcal{H}} p(y \in C|h)p(h|d) \qquad (21)$$

where the hypothesis space $\mathcal{H}$ is the set of all possible concepts in a domain, $p(y \in C|h) = 1$ if y is in hypothesis h, and 0 otherwise. The posterior probability $p(h|d)$ can be computed via Bayes' rule from Equation 1.

We explore two variations of likelihood functions based on assumptions of how the data were generated. If the data are generated uniformly at random, the likelihood follows from *weak sampling*: $p(d|h) = 1$ if all examples in d are in h and $p(d|h) = 0$ otherwise. If the data are generated at random from the true concept C, the likelihood follows from *strong sampling*, where $p(d|h) = 1/|h|^n$ if all n examples in d belong to h, and $p(d|h) = 0$ otherwise. This likelihood function incorporates the *size principle*, where hypotheses with fewer items are given more weight than hypotheses with more items for the same set of data (Tenenbaum & Griffiths, 2001).

In the SDM, the location $\mathbf{x}^*$ corresponds to a hypothesis h, and the content $\mathbf{z}$ corresponds to data d. For both weak and strong sampling assumptions, we set the read function of the SDM, $r(d,h)$, to be proportional to the Bayesian likelihood by weighting the selection vector $\mathbf{y}$ by $p(d|h)$.

Analysis

To evaluate the performance of this modified SDM, we used the category-based induction dataset from Osherson et al. (1990), consisting of 36 two-premise arguments and 45 three-premise arguments (e.g., "Cat, Dog, Horse can get disease X") for a domain of 10 animals. Thus, each observation d is encoded as a binary feature vector of length $N = 10$. We used a taxonomic hypothesis space following Kemp and Tenenbaum (2009), constructed from human similarity judgments for each possible pairing of animals. As before, we stored 1000 exemplars sampled from $p(h)$ in the SDM, and varied the number of hard addresses among eight evenly spaced values between 128 and $2^N = 1024$. We evaluated this modified SDM against the Bayesian model of property induction for 3 values of D_w over a constant reading radius $D_r = 0$. The results are presented in Figure 4[1].

We find the SDM approximates Bayesian inference for a variety of write radii and, as predicted, matches Bayes when

[1]We note that these correlations are not strictly monotonic as one might intuit, due to sampling error in the address space (most notable in the case when $D_w = 0$.)

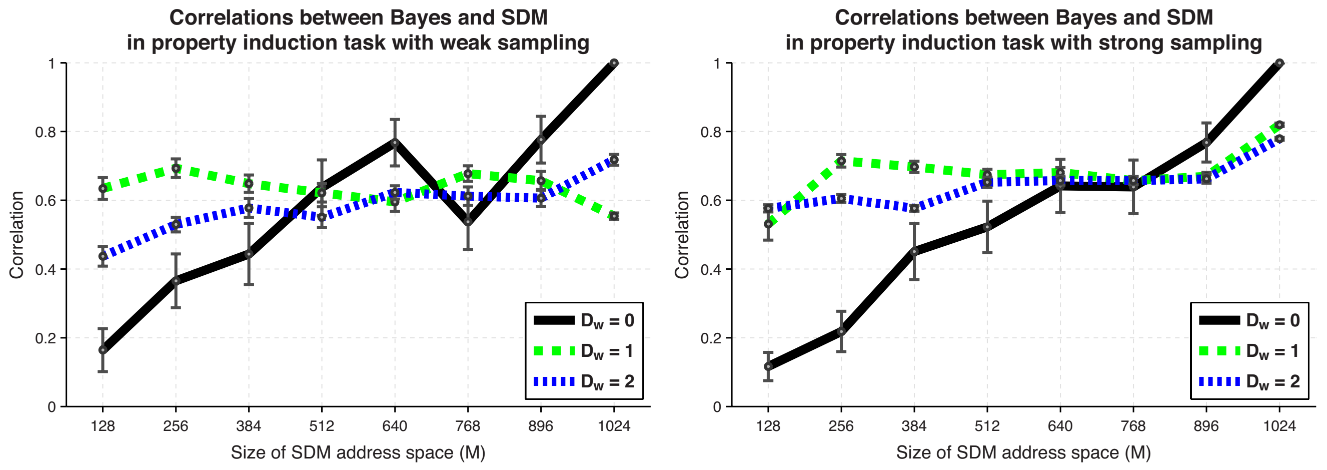

Figure 4: The average correlations between the Bayesian posterior and pre-thresholded SDM outputs assuming weak sampling (left panel), and assuming strong sampling (right panel) for the property induction task. $D_r = 0$ for each of the 3 values of D_w presented.

$D_w = 0$ and $M = 2^N$. Furthermore, the SDMs that use a weak or strong sampling read rule correlate equally well with the Bayesian models that use weak or strong sampling likelihoods, respectively. This nicely illustrates the correspondence between the SDM's read rule and a Bayesian likelihood, and implies that SDMs can approximate a broad range of Bayesian models by adjusting the read rule to match the likelihood function.

Conclusion

What constraints do the algorithmic and implementation levels impose on probabilistic models of cognition? We explored this question by considering whether the computations employed by a distributed representation of associative memory could approximate Bayesian inference. By choosing the SDM read rule to appropriately match the the likelihood function, we showed in two separate scenarios that SDMs can accurately implement a specific form of Bayesian inference called importance sampling.

Future work will take our analyses one step further and investigate whether SDMs can approximate Bayesian inference *without* modifying their read rule. Specifically, can we encode the data in a manner such that reading it from a standard SDM is still proportional to the likelihood? If so, it would make SDMs an even more general and appealing approach to performing Bayesian inference. Regardless, the results presented in this paper are an important step towards an explanation of how the structured representations assumed by probabilistic models of cognition could be expressed in the distributed, continuous representations commonly used in algorithmic-level models such as neural networks.

Acknowledgments. This work was supported by grant number FA-9550-10-1-0232 from the Air Force Office of Scientific Research and a Berkeley Fellowship awarded to JBH.

References

Anderson, C. H. (1989). A conditional probability interpretation of Kanerva's sparse distributed memory. In *International joint conference on neural networks* (pp. 415–417).

Gigerenzer, G., & Todd, P. (1999). *Simple heuristics that make us smart*. Oxford University Press, USA.

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

Griffiths, T., Vul, E., & Sanborn, A. (2012). Bridging levels of analysis for probabilistic models of cognition. *Current Directions in Psychological Science*, *21*(4), 263–268.

Kanerva, P. (1988). *Sparse Distributed Memory Systems*. MIT Press.

Kanerva, P. (1993). Sparse Distributed Memory and Related Models. In M. Hassoun (Ed.), *Associative neural memories: Theory and implementation* (pp. 50–76). Oxford University Press.

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

Lewand, R. (2000). *Cryptological mathematics*. Mathematical Association of America.

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

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

Neal, R. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Tech. Rep. No. CRG-TR-93-1). Department of Computer Science, University of Toronto.

Osherson, D., Smith, E., Wilkie, O., Lopez, A., & Shafir, E. (1990). Category-based induction. *Psychological Review*, *97*(2), 185.

Rumelhart, D., & Siple, P. (1974). Process of recognizing tachistoscopically presented words. *Psychological Review*, *81*(2), 99.

Shi, L., & Griffiths, T. (2009). Neural implementation of hierarchical Bayesian inference by importance sampling. *Advances in Neural Information Processing Systems*, *22*, 1669–1677.

Shi, L., Griffiths, T. L., Feldman, N. H., & Sanborn, A. N. (2010). Exemplar models as a mechanism for performing Bayesian inference. *Psychonomic Bulletin & Review*, *17*(4), 443–64.

Tenenbaum, J., & Griffiths, T. (2001). Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences*, *24*(4), 629–640.

Tenenbaum, J., Kemp, C., Griffiths, T., & Goodman, N. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, *331*(6022), 1279–1285.