Probability Matching via Deterministic Neural Networks

Milad Kharratzadeh
Department of Electrical & Computer Engineering
McGill University
Motreal, Canada
milad.kharratzadeh@mail.mcgill.ca

Thomas Shultz
Department of Psychology
& School of Computer Science
McGill University
Montreal, Canada
thomas.shultz@mcgill.ca

Abstract

We propose a constructive neural-network model comprised of deterministic units which estimates and represents probability distributions from observable events — a phenomenon related to the concept of probability matching. We use a form of operant learning, where the underlying probabilities are learned from positive and negative reinforcements of the inputs. Our model is psychologically plausible because, similar to humans, it learns to represent probabilities without receiving any representation of them from the external world, but rather by experiencing individual events. Also, we discuss how the estimated probabilities can be used in a setting with deterministic units to produce matching behaviour in choice. Our work is a step towards understanding the neural mechanisms underlying probability matching behavior by specifying processes at the algorithmic level.

1 Introduction

The matching law states that the rate of a response is proportional to its rate of observed reinforcement and has been applied to many problems in psychology and economics [1, 2]. A closely related empirical phenomenon is probability matching where the predictive probability of an event is matched with the underlying probability of its outcome [3]. For example, in decision theory, many experiments show that participants select alternatives proportional to their reward frequency. This means that in many scenarios, instead of maximizing their utility by always choosing the alternative with the higher chance of reward, they match the underlying probabilities of different alternatives. This is in contrast with the reward-maximizing strategy of always choosing the most probable outcome. The apparently suboptimal behaviour of probability matching is a long-standing puzzle in the study of decision making under uncertainty and has been studied extensively [4–9].

In this paper, we provide a psychologically plausible neural framework to explain probability matching at Marr's implementation level [10]. We introduce an artificial neural network framework which can be used to explain how deterministic neural networks can learn to represent probability distributions, even without receiving any direct representations of these probabilities from the external world. We offer an explanation of how the network is able to estimate and represent probabilities solely from observing the occurrence patterns of events, in the manner of probability matching. In the context of Bayesian models of cognition, such probability-matching processes could explain the origin of the prior and likelihood probability distributions that are currently assumed or constructed by modelers. Thus, in contrast to current literature that proposes probability matching as an alternative to Bayesian models [11, 12], we argue that probability matching can be seen as part of a larger Bayesian framework to learn prior and likelihood distributions which can then be used for Bayesian inference.

2 Problem Statement

We provide a neural-network framework with deterministic units capable of implementing probability matching, i.e., learning the underlying probabilities (knowledge) and making choices using those probabilities (use). We assume that a direct representation of these probabilities from the external world is not available, and the probabilities must be estimated from input instances reinforced at various frequencies. For example, for a stimulus, s, reinforced on k out of its total n presentations in the training set, probability matching yields $\widehat{P}(s) = k/n$.

Mathematically, we assume the task of learning a probability mass function $P : H \to [0, 1]$, where H is a discrete hypothesis space. The training set consists of a collection of input instances reinforced with a frequency proportional to an underlying probability function; i.e., the hypothesis h_i is paired with observations sampled from $Bernoulli\,(P(h_i))$ where 1 corresponds to a positive reinforcement and 0 corresponds to a negative reinforcement. Then, the knowledge part of probability matching reduces to estimating the actual probabilities from these 0 or 1 observations. This is in accordance with the real-world scenarios, where observations are in the form of events which can occur or not (represented by outputs of 1 and 0, respectively) and the learner does not have access to the actual probabilities of those events.

We use deterministic neural networks where each unit takes a weighted sum of inputs from some other units and, using its activation function, computes its output. These outputs are propagated through the network until the network's final outputs are computed in the last layer. We consider a neural network with a single input unit (taking h_i) and a single output unit (representing the probability). Our goal is to learn a network that outputs $P(h_i)$ when h_i is presented at the input.

In classical artificial neural networks, the target values are fixed and deterministically derived from the underlying function and the corresponding inputs. However, in the problem we consider here, we do not have access to the final, fixed targets (i.e., the actual probabilities). Instead, the training set is composed of input instances that are reinforced with various frequencies. An important question is whether a network of deterministic units can learn the underlying probability distributions from such 0, 1 observations. And if yes, how? We answer these two questions in Sections 4 and 5, respectively. Then, in Section 6, we show the learned probabilities can be used to produce matching behaviour.

3 Related Work

Our proposed scheme differs from the classical approach to neural networks in that there is no one-to-one relationship between inputs and output. Instead of being paired with one fixed output, each input is here paired with a series of 1s and 0s presented separately at the output unit. Moreover, in our framework, the actual targets (underlying probabilities) are hidden from the network and, in the training phase, the network is presented only with inputs and their probabilistically varying outputs.

The main difference of our work with the current literature (and the main novelty of this work) is the use of a population of deterministic units to learn the probabilities and producing the matching behaviour. The relationship between neural network learning and probabilistic inference has been extensively studied mainly with stochastic units that fire with particular probabilities. Boltzmann machines [13] and their various derivatives, including Deep Learning in hierarchical restricted Boltzmann machines (RBM) [14], have been proposed to learn a probability distribution over a set of inputs. There are many other papers studying probabilistic computations that can be done using similar networks (e.g., [15–18]). See [19] for a more comprehensive review.

In our model, representation of probability distributions emerges as a property of a network of deterministic units rather than having individual units with activations governed by some probability distribution. Moreover, models with stochastic units such as RBM "require a certain amount of practical experience to decide how to set the values of numerical meta-parameters" [20], which makes them neurally and psychologically implausible for modeling probability matching in the relatively autonomous learning of humans or animals. As we see later, our model implements the probability matching in a relatively autonomous, neurally–plausible fashion, by using deterministic units in a constructive learning algorithm that builds the network topology as it learns.

4 Statistical Properties

In this section, we show that the successful training of a deterministic neural network to minimize the output error for the problem defined in Section 2 results in learning the underlying probabilities.

Remember that the training set consists of hypotheses, h_i, paired with a sequence of probabilistic outputs, r_{ij}, set to either 1 (positive reinforcement) or 0 (negative reinforcement). The frequency of the reinforcement (outputs of 1) is determined by the underlying probability distribution. The structure of the network and the weights are adjusted (more details later) to minimize the sum-of-squares error:

$$\mathcal{E} = \frac{1}{2}\sum_{i,j}(o_i - r_{ij})^2, \tag{1}$$

where o_i is the network's output when h_i is presented at the input layer. We show that minimizing this error, results in learning the underlying distribution: if we present a sample input, the output of the network would be its probability of being reinforced. Note that we never present this probability explicitly to the network. This means that the network learns and represents the probability distributions from observing patterns of events.

The statistical properties of feed-forward neural networks with deterministic units have been studied as non-parametric density estimators. Denote the inputs of a network with X and the outputs with Y (both can be vectors). In a probabilistic setting, the relationship between X and Y is determined by the conditional probability $P(Y|X)$. In [21] and [22], White showed that under certain assumptions, feed-forward neural networks with a single hidden layer can consistently learn the conditional expectation function $E(Y|X)$. However, as White mentions, his analyses "do not provide more than very general guidance on how this can be done" and suggests that "such learning will be hard" [21, p. 454]. Moreover, these analyses "say nothing about how to determine adequate network complexity in any specific application with a given training set of size n" [21, p. 455]. In this section, we extend these results to a more general case with no restrictive assumptions about the structure of the network and the learning algorithm. Then, in the next section, we propose a learning algorithm that automatically determines the adequate network complexity in any specific application.

In the following, we state the theorem and our learning technique for the case where $Y \in \{0, 1\}$, since in this case $E(Y = 1|X) = P(Y = 1|X)$. Thus, learning results in representing the underlying probabilities in the output unit. The extension of the theorem and learning algorithm to more general cases is straightforward.

Theorem 1. Assume that $P : H \to \mathbb{R}$ is a probability mass function on a hypothesis space, H, and we have observations $\{(h_i, r_{ij}) \mid r_{ij} \sim \text{Bernoulli}(P(h_i)),\ h_i \in H\}$. Define the network error as the sum–of–squares error at the output:

$$\mathcal{E}_p = \frac{1}{2}\sum_{i}\sum_{j=1}^{n}(o_i - r_{ij})^2. \tag{2}$$

where o_i is the network's output when h_i is presented at the input, and r_{ij} is the probabilistic output determining whether the hypothesis h_i is reinforced ($r_{ij} = 1$) or not ($r_{ij} = 0$). Then, any learning algorithm that successfully trains the network to minimize the output sum–of–squared error yields probability matching (i.e., reproduces f in the output).

Proof. Minimizing the error, we have:

$$\nabla\mathcal{E}_p = \left(\frac{\partial\mathcal{E}_p}{\partial o_1}, \ldots, \frac{\partial\mathcal{E}_p}{\partial o_m}\right) = \left(n \cdot o_1 - \sum_{j=1}^{n} r_{1j}, \ldots, n \cdot o_m - \sum_{j=1}^{n} r_{mj}\right) = 0 \tag{3}$$

$$\Rightarrow o_i^* = \sum_{j=1}^{n}\frac{r_{ij}}{n}, \quad \forall i. \tag{4}$$

According to the strong law of large numbers $o_i^* \overset{a.s.}{\to} E[r_{ij}] = P(h_i), \forall h_i \in H$, where $\overset{a.s.}{\to}$ denotes almost sure convergence. Therefore, the network's output converges to the underlying probability distribution, P, at all points. □

Theorem 1 shows the important point that neural networks with deterministic units are able to asymptotically estimate an underlying probability distribution solely based on observable reinforcement rates. Unlike previous similar results in literature [21–23], Theorem 1 does not impose any constraint on the network structure, the learning algorithm, or the distribution being learned. However, an important assumption in this theorem is the successful minimization of the error by the learning algorithm. As pointed out earlier, two important questions remain to be answered: (i) how can this learning be done? and (ii) how can adequate network complexity be automatically identified for a given training set? In the next section, we address these problems and propose a learning framework to successfully minimize the output error.

5 The Learning Algorithm

The outputs in the training set, paired with input hypotheses, are 0 or 1. Our goal in probability matching is not to converge to any of these values, but to the underlying probability. To achieve that goal we use the idea of learning cessation [24]. The learning cessation method monitors learning progress in order to autonomously abandon unproductive learning. It checks the absolute difference of consecutive errors and if this value is less than a fixed threshold multiplied by the current error for a fixed number of consecutive learning phases (called patience), learning is abandoned. This technique for stopping deterministic learning of stochastic patterns does not require the psychologically unrealistic validation set of training patterns [25, 26].

Our method is presented in Algorithm 1 where we represent the whole network (units and connections) by the variable `NET`. Also, the learning algorithm we use to train our network is represented by the operator `train_one_epoch`, where an epoch is a pass through all of the training patterns. We can use any algorithm to train our network, as long as it successfully minimizes the error term in (2). Next, we present a learning algorithm that can achieve that goal.

Algorithm 1 Probability matching with neural networks and learning cessation

```
Input: Training Set $S_{train} = \{(h_i, r_{ij}) \mid h_i \in X \,; r_{ij} \sim \text{Bernoulli}(P(h_i))\}$;
        Cessation threshold $\epsilon_c$; Cessation patience $patience$
Output: Learned network outputs $\{o_i \,,\; i = 1, \ldots, m\}$
$counter \leftarrow 0, t \leftarrow 0$
while true do
    $(\{o_i \mid i = 1, \ldots, m\}, \texttt{NET}) \leftarrow \texttt{train\_one\_epoch}(\texttt{NET}, S_{train})$   ▷ Updating the network
    $\mathcal{E}_p(t) \leftarrow \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}(o_i - r_{ij})^2$   ▷ Computing the updated error
    if $|\mathcal{E}_p(t) - \mathcal{E}_p(t-1)| \geq \epsilon_c \cdot |\mathcal{E}_p(t)|$ then   ▷ Checking the learning progress
        $counter \leftarrow 0$
    else
        $counter \leftarrow counter + 1$
        if $counter = patience$ then
            break
        end if
    end if
    $t \leftarrow t + 1$
end while
```

Theorem 1 proves that the minimization of the output sum–of–squared error yields probability matching. However, the unusual properties of the training set we employ (such as the probabilistic nature of input/output relations) as well as the fact that we do not specify the complexity of the underlying distribution in advance may cause problems for some neural learning algorithms. The most widely used learning algorithm for neural networks is Back Propagation, also used in [27] in the context of probability matching. In Back Propagation (BP), the output error is propagated backward and the connection weights are individually adjusted to minimize this error. Despite its many successes in cognitive modeling, we do not recommend using BP in our scheme for two important reasons. First, when using BP, the network's structure must be fixed in advance (mainly heuristically). This makes it impossible for the learning algorithm to automatically adjust network complexity to the problem at hand [21]. Moreover, this property limits the generalizability and autonomy of BP and also, along with back-propagation of error signals, makes it psychologically implausible. Second, due to their fixed design, BP networks are not suitable for cases where the underlying distribution changes over time. For instance, if the distribution over the hypothesis space gets much more complicated over time, the initial network's complexity (i.e., number of hidden units) would fall short of the required computational power.

Instead of BP, we use a variant of the cascade correlation (CC) method called sibling-descendant cascade correlation (SDCC) which is a constructive method for learning in multi-layer artificial neural networks [28]. SDCC learns both the network's structure and the connection weights; it starts with a minimal network, then automatically trains new hidden units and adds them to the active network, one at a time. Each new unit is employed at the current or a new highest layer and is the best of several candidates at tracking current network error.

The SDCC network starts as a perceptron topology, with input units coding the example input (in our case, a single unit coding the input) and output units coding the correct response to that input (in our case, a single unit representing the probability). In a constructive fashion, deterministic units are recruited into the network one at a time as needed to reduce error. In classical CC, each new recruit is installed on its own layer, higher than previous layers. The SDCC variant is more flexible in that a recruit can be installed either on the current highest layer (as a sibling) or on its own higher layer as a descendent, depending on which location yields the higher correlation between candidate unit activation and current network error [28]. In both CC and SDCC, learning progresses in a recurring sequence of two phases – output phase and input phase. In output phase, network error at the output units is minimized by adjusting connection weights without changing the current topology. In the input phase, a new unit is recruited such that the correlation between its activation and network error is maximized. In both phases, the optimization is done by the Quickprop algorithm [29].

SDCC offers two major advantages over BP. First, it constructs the network in an autonomous fashion (i.e., a user does not have to design the topology of the network, and also the network can adapt to environmental changes). Second, its greedy learning mechanism can be orders of magnitude faster than the standard BP algorithm [30]. SDCC's relative autonomy in learning is similar to humans' developmental, autonomous learning [31]. With SDCC, our method implements psychologically realistic learning of probability distributions, without any preset topological design. The psychological and neurological validity of cascade-correlation and SDCC has been well documented in many publications [32, 33]. These algorithms have been shown to accurately simulate a wide variety of psychological phenomena in learning and psychological development. Like all useful computational models of learning, they abstract away from neurological details, many of which are still unknown. Among the principled similarities with known brain functions, SDCC exhibits distributed representation, activation modulation via integration of neural inputs, an S-shaped activation function, layered hierarchical topologies, both cascaded and direct pathways, long-term potentiation, self-organization of network topology, pruning, growth at the newer end of the network via synaptogenesis or neurogenesis, weight freezing, and no need to back-propagate error signals.

6 Generating Matching Behaviour with Deterministic Units

The term "probability matching" either refers to learning the underlying probabilities or to making choices using those probabilities. So far, we explained how a neural network can learn to estimate the probabilities. In this section, we discuss how this estimated probability can be used in a setting with deterministic units to produce matching behaviour in choice. We show that deterministic units with simple thresholding activation functions and added Gaussian noise in the input can generate probabilistic outputs similar to probability matching behaviour. Assume that we have a neuron with two inputs: the estimated probability that a response is correct, $0 \leq v \leq 1$, and a zero–mean Gaussian noise, $\epsilon \sim \mathcal{N}(0, \gamma)$. Then, given the thresholding activation function, the output will be 1 if $v + \epsilon > \tau$ and 0 if $v + \epsilon \leq \tau$ for a given threshold τ. Therefore, the probability of producing 1 at the output is:

$$P(\text{output} = 1 | v, \tau, \gamma) = P(v + \epsilon > \tau) = P(\epsilon > \tau - v) = \overbrace{0.5 - 0.5 \operatorname{erf}\left(\frac{\tau - v}{\gamma\sqrt{2}}\right)}^{f(v)}, \qquad (5)$$

where *erf* denotes the error function: $\mathrm{erf}(x) = (2/\sqrt{\pi}) \int_0^x e^{-t^2}\,\mathrm{d}t$. It is easy to see that $f(v)$ lies between 0 and 1 and, for appropriate choices of τ and γ, we have $f(v) \simeq v$ for $0 < v < 1$ (see Fig. 1). Thus, a single thresholding unit with additive Gausian noise in the input can use the estimated probabilities to produce responses that match the response probabilities (similar to the matching behaviour of people using probabilistic knowledge to make their choices).

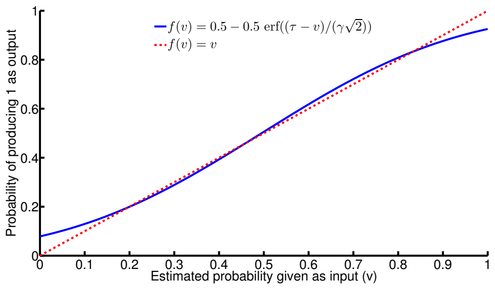
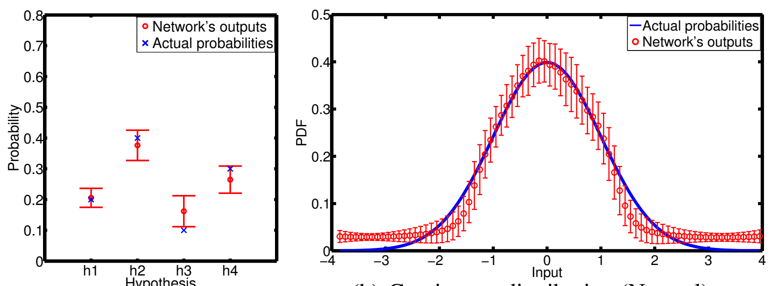


Figure 1: A deterministic unit with a thresholding activation function generating responses that match the probabilities that each response is correct ($\tau = 1, \gamma = 0.35$)

7 Simulation Study

7.1 Probability Matching

Through simulations, we show that our proposed framework is indeed capable of learning the underlying distributions. We consider two cases here, but similar results are observed for a wide range of distributions. First, we consider a case of four hypotheses with probability values $0.2, 0.4, 0.1$, and 0.3. Also, we consider a Normal probability distribution where the hypotheses correspond to small intervals on the real line from -4 to 4. For each input sample we consider 15 randomly selected instances in each training epoch. As before, these instances are positively or negatively reinforced independently and with a probability equal to the actual underlying probability of that input. We use SDCC with learning cessation to train our networks. Fig. 2, plotted as the average and standard deviation of the results for 50 networks, demonstrates that for both discrete and continuous probability distributions, the network outputs are close to the actual distribution. Although, to save space, we show the results for only two sample distributions, our experiments show that our model is able to learn a wide range of distributions including Binomial, Poisson, Gaussian, and Gamma [34]. Replication of the original probability distribution by our model is important, because, contrary to previous models, it is done without stochastic neurons and without any explicit information about the actual distribution or fitting any parameter or structure in advance. Moreover, it is solely based on observable information in the form of positive and negative reinforcements.



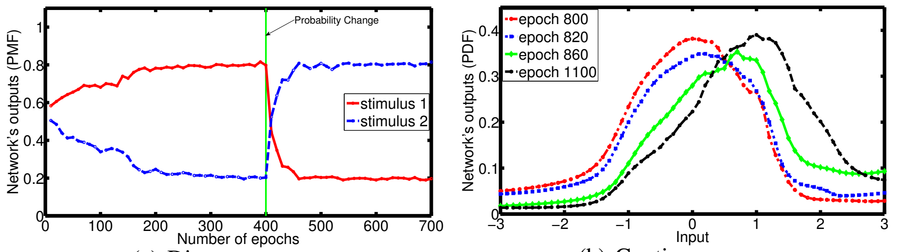
(a) Discrete distribution (b) Continuous distribution (Normal)

Figure 2: Replication of the underlying probability distribution by our SDCC model. The results (mean and standard deviation) are averaged over 50 different networks.

7.2 Adapting to Changing Environments

In many naturally–occurring environments, the underlying reward patterns change over time. For example, in a Bayesian context, the likelihood of an event can change as the underlying conditions change. Because humans are able to adapt to such changes and update their internal representations of probabilities, successful models should have this property as well. We examine this property in the following example experiment. Assume we have a binary distribution where the possible outcomes have probabilities 0.2 and 0.8, and these probabilities change after 400 epochs to 0.8 and 0.2, respectively. In Fig. 3(a), we show the network's outputs for this scenario. We perform a similar simulation for the continuous case where the underlying distribution is Gaussian and we change the mean from 0 to 1 at epoch 800; the network's outputs are shown in Fig. 3(b). We observe that in both cases, the network successfully updates and matches the new probabilities.

We also observe that adapting to the changes takes less time than the initial learning. For example, in the discrete case, it takes 400 epochs to learn the initial probabilities while it takes around 70 epochs to adapt to the new probabilities. The reason is that for the initial phase, constructive learning has to grow the network until it is complex enough to represent the probability distribution. However, once the environment changes, the network has enough computational capability to quickly adapt to the environmental changes with a few internal changes (in weights and/or structure). We verify this in our experiments. For instance, in the Gaussian example, we observe that all 20 networks recruited 5 hidden units before the change and 11 of these networks recruited 1 and 9 networks recruited 2 hidden units afterwards. We know of no precise psychological evidence for this reduction in learning time, but our results serve as a prediction that could be tested with biological learners. This would seem to be an example of the beneficial effects of relevant existing knowledge on new learning.



(a) Discrete case (b) Continuous case

Figure 3: Reaction of the network to the changes in target probabilities.

8 Discussion

A mentioned before, probability matching (choosing alternatives proportionally to their reward frequency) is in contrast with the reward-maximizing strategy of always choosing the most probable outcome. There are numerous, and sometimes contradictory, attempts to explain this choice anomaly. Some suggest that probability matching is a cognitive shortcut driven by cognitive limitations [3, 4]. Others assume that matching is the outcome of misperceived randomness which leads to searching for patterns even in random sequences [5, 35]. It is shown that as long as people do not believe in the randomness of a sequence, they try to discover regularities in it to improve accuracy [6]. It is also shown that some of those who perform probability matching in random settings have a higher chance of finding a pattern when one exists [7]. In contrast to this line of work, some researchers argue that probability matching reflects a mistaken intuition and can be overridden by deliberate consideration of alternative choice strategies [8]. In [9], the authors suggest that a sequence-wide expectation regarding aggregate outcomes might be a source of the intuitive appeal of matching. It is also shown that people adopt an optimal response strategy if provided with (i) large financial incentives, (ii) meaningful and regular feedback, or (iii) extensive training [36].

Our neural-network framework is compatible with all these accounts of probability matching. Firstly, probability matching is the norm in both humans [37] and animals [38, 39]. It is clear that in these settings agents who match probabilities form an internal representation of the outcome probabilities. Even for particular circumstances where a maximizing strategy is prominent [7,36], it is necessary to have some knowledge of the distribution to produce optimal-point responses. Having a sense of the distribution provides the flexibility to focus on the most probable point (maximizing), sample in proportion to probabilities (matching), or even generate expectations regarding aggregate outcomes (expectation generation), all of which are evident in psychology experiments.

Probabilistic models of cognition can be defined with either symbolic or continuous representations, or hybrids of both. In fact, more effective routes to understanding human intelligence can be found by combining these two traditionally opposing approaches using a statistical inference scheme over structured symbolic knowledge representations [40]. Our proposed neural interpretation of probabilistic representations helps to explore that interface in greater depth.

References

[1] R. J. Herrnstein, "Relative and absolute strength of response as a function of frequency of reinforcement," *Journal of the Experimental Analysis of Behaviour*, vol. 4, pp. 267–272, 1961.

[2] ——, *The Matching Law: Papers on Psychology and Economics*, H. Rachlin and D. Laibson, Eds. Cambridge, MA: Harvard University Press, 2000.

[3] N. Vulkan, "An economist's perspective on probability matching," *Journal of Economic Surveys*, vol. 14, no. 1, pp. 101–118, 2000.

[4] R. F. West and K. E. Stanovich, "Is probability matching smart? associations between probabilistic choices and cognitive ability," *Memory & Cognition*, vol. 31, no. 2, pp. 243–251, 2003.

[5] G. Wolford, S. E. Newman, M. B. Miller, and G. S. Wig, "Searching for patterns in random sequences." *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, vol. 58, no. 4, p. 221, 2004.

[6] J. I. Yellott Jr, "Probability learning with noncontingent success," *Journal of mathematical psychology*, vol. 6, no. 3, pp. 541–575, 1969.

[7] W. Gaissmaier and L. J. Schooler, "The smart potential behind probability matching," *Cognition*, vol. 109, no. 3, pp. 416–422, 2008.

[8] D. J. Koehler and G. James, "Probability matching in choice under uncertainty: Intuition versus deliberation," *Cognition*, vol. 113, no. 1, pp. 123–127, 2009.

[9] G. James and D. J. Koehler, "Banking on a bad bet probability matching in risky choice is linked to expectation generation," *Psychological Science*, vol. 22, no. 6, pp. 707–711, 2011.

[10] D. Marr, *Vision*. San Francisco, CA: W. H. Freeman, 1982.

[11] J. S. Bowers and C. J. Davis, "Bayesian just-so stories in psychology and neuroscience." *Psychological Bulletin*, vol. 138, no. 3, pp. 389–414, 2012.

[12] F. Eberhardt and D. Danks, "Confirmation in the cognitive sciences: The problematic case of Bayesian models," *Minds and Machines*, vol. 21, no. 3, pp. 389–410, 2011.

[13] H. Ackley, G. Hinton, and J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, pp. 147–169, 1985.

[14] G. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527 – 1554, 2006.

[15] J. Movellan and J. L. McClelland, "Learning continuous probability distributions with symmetric diffusion networks," *Cognitive Science*, vol. 17, pp. 463–496, 1993.

[16] J. L. McClelland, "Connectionist models and bayesian inference," *Rational models of cognition*, pp. 21–53, 1998.

[17] T. S. Jaakkola, L. K. Saul, and M. I. Jordan, "Fast learning by bounding likelihoods in sigmoid type belief networks," in *Advances in Neural Information Processing Systems 22*, 1996.

[18] J. L. McClelland, D. Mirman, D. J. Bolger, and P. Khaitan, "Interactive activation and mutual constraint satisfaction in perception and cognition," *Cognitive science*, vol. 38, no. 6, pp. 1139–1189, 2014.

[19] M. Kharratzadeh and T. R. Shultz, "Neural implementation of probabilistic models of cognition," *arXiv preprint arXiv:1501.03209*, 2015.

[20] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[21] H. White, "Learning in artificial neural networks: A statistical perspective," *Neural computation*, vol. 1, no. 4, pp. 425–464, 1989.

[22] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[23] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," in *Backpropagation: Theory, Arcitecture, and applications*, Y. Chauvin and D. E. Rumelhart, Eds., Hillsdale, NJ, USA, 1995, pp. 1–34.

[24] T. Shultz, E. Doty, and F. Dandurand, "Knowing when to abandon unproductive learning," in *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, Austin, TX: Cognitive Science Society, 2012, pp. 2327–2332.

[25] L. Prechelt, "Early stopping - but when?" in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Orr and K.-R. Muller, Eds. Berlin: Springer, 1998, vol. 1524, pp. 55–69.

[26] C. Wang, S. S. Venkatesh, and J. S. Judd, "Optimal stopping and effective machine complexity in learning," in *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1993, pp. 303–310.

[27] M. Dawson, B. Dupuis, M. Spetch, and D. Kelly, "Simple artificial neural networks that match probability and exploit and explore when confronting a multiarmed bandit," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1368–1371, 2009.

[28] S. Baluja and S. E. Fahlman, "Reducing network depth in the cascade-correlation learning architecture," Carnegie Mellon University, School of Computer Science, Tech. Rep., 1994.

[29] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proc. of the Connectionist Models Summer School*. Los Altos, CA: Morgan Kaufmann, 1988, pp. 38–51.

[30] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*. Loas Altos, CA: Morgan Kaufmann, 1990, pp. 524–532.

[31] T. Shultz, "A constructive neural-network approach to modeling psychological development," *Cognitive Development*, vol. 27, pp. 383–400, 2012.

[32] ——, *Computational Developmental Psychology*. Cambridge, MA: MIT Press, 2003.

[33] ——, "Computational models of developmental psychology," in *Oxford Handbook of developmental Psychology, Vol. 1: Body and mind*, P. D. Zelazo, Ed. Newyork: Oxford University Press, 2013.

[34] M. Kharratzadeh and T. Shultz, "Neural-network modelling of Bayesian learning and inference," in *Proceedings of the 35th Annual Meeting of Cognitive Science*. Austin, TX: Cognitive Science Society, 2013, pp. 2686–2691.

[35] G. Wolford, M. B. Miller, and M. Gazzaniga, "The left hemisphere's role in hypothesis formation." *The Journal of Neuroscience*, 2000.

[36] D. R. Shanks, R. J. Tunney, and J. D. McCarthy, "A re-examination of probability matching and rational choice," *Journal of Behavioral Decision Making*, vol. 15, no. 3, pp. 233–250, 2002.

[37] D. R. Wozny, U. R. Beierholm, and L. Shams, "Probability matching as a computational strategy used in perception," *PLoS computational biology*, vol. 6, no. 8, p. e1000871, 2010.

[38] K. L. Kirk and M. Bitterman, "Probability-learning by the turtle," *Science*, vol. 148, no. 3676, pp. 1484–1485, 1965.

[39] U. Greggers and R. Menzel, "Memory dynamics and foraging strategies of honeybees," *Behavioral Ecology and Sociobiology*, vol. 32, no. 1, pp. 17–29, 1993.

[40] T. L. Griffiths, C. Kemp, and J. B. Tenenbaum, "Bayesian models of cognition," 2008.