4 Herding as a Learning System with Edge-of-Chaos Dynamics

Yutian Chen yutianc@google.com
Google DeepMind
London, UK

Max Welling m.welling@uva.nl
University of Amsterdam
Amsterdam, Netherlands

Herding defines a deterministic dynamical system at the edge of chaos. It generates a sequence of model states and parameters by alternating parameter perturbations with state maximizations, where the sequence of states can be interpreted as "samples" from an associated MRF model. Herding differs from maximum likelihood estimation in that the sequence of parameters does not converge to a fixed point and differs from an MCMC posterior sampling approach in that the sequence of states is generated deterministically. Herding may be interpreted as a"perturb and map" method where the parameter perturbations are generated using a deterministic nonlinear dynamical system rather than randomly from a Gumbel distribution. This chapter studies the distinct statistical characteristics of the herding algorithm and shows that the fast convergence rate of the controlled moments may be attributed to edge of chaos dynamics. The herding algorithm can also be generalized to models with latent variables and to a discriminative learning setting. The perceptron cycling theorem ensures that the fast moment matching property is preserved in the more general framework.

4.1 Introduction

The traditional view of a learning system is one where an initial parameter vector $\mathbf{w}_0$ is updated until some convergence criterion is met: $\mathbf{w}_0, \mathbf{w}_1, .., \mathbf{w}_T$ with (in theory) $T \to \infty$ and $\mathbf{w}_\infty = \mathbf{w}^*$ a fixed point of the updates. These updates usually maximize some objective such as the log-likelihood of the data. We can view this process as a dynamical system with a contractive map $\mathbf{w}_{t+1} = F_t(\mathbf{w}_t)$ which is designed to iterate to a fixed point. The map F_t can be either deterministic or stochastic. For instance, batch gradient descent is an example of a deterministic map while stochastic gradient descent is an example of a stochastic map. A natural question is whether the existence of a fixed point $\mathbf{w}^*$ is important, and whether meaningful learning systems can exist that do not converge to any fixed point but traverse an attractor set. To answer this question we can draw inspiration from Markov chain Monte Carlo (MCMC) procedures which generate samples from a posterior distribution $P(\mathbf{w}|\mathcal{D})$ (with $\mathcal{D}$ indicating the data). MCMC also generates a sequence of parameter values $\mathbf{w}_0, .., \mathbf{w}_T$ but one that does not converge to a fixed point. Rather the samples form an attractor set with a measure (density) equal to the posterior distribution. One can make meaningful predictions with MCMC chains by making predictions for every sampled model $\mathbf{w}_t$ separately and subsequently averaging the predictions. There is also evidence that learning in the brain is a dynamical process. For instance, Aihara and Matsumoto (1982) have described chaotic dynamics in the Hodgkin-Huxley equations for membrane dynamics and studied them experimentally in squid giant axons. Also, much evidence has now been accumulated that synapses are subject to fast dynamical processes such as postsynaptic depression and facilitation (Tsodyks et al., 1098).

Herding (Welling, 2009a) is perhaps the first learning dynamical system based on a deterministic map and with a nontrivial attractor (i.e. not a single fixed point). It emerged from taking the limit of infinite stepsize in the usual (maximum likelihood) updates for a Markov random field (MRF) model. It can be observed that in this limit the parameters will not converge to a fixed point but rather traverse a usually non-periodic trajectory in weight space. The information contained in the data is now stored in the trajectories (or the attractor) of this dynamical system, rather than in a point estimate of a collection of parameters. In fact it can be shown that this dynamical system is neither periodic (under some conditions) nor chaotic, a state which is associated with "edge of chaos" dynamics. As illustrated in this chapter, by slowly increasing the stepsize (or equivalently lowering the temperature) we will move from a standard MRF maximum likelihood learning system

with a single fixed point, through a series of period doublings to a system on the edge of chaos. One can show that the attractor is sometimes fractal, and that the Lyapunov exponents of this system are equal to 0 implying that two nearby trajectories will eventually separate but only polynomially fast (and not exponentially fast as with chaotic systems). Many of the dynamical properties of this system are described by the theory of "piecewise isometries" (Goetz, 2000).

Herding can thus be viewed as a dynamical system that generates state-space samples $\mathbf{s}_1, .., \mathbf{s}_T$ that are highly similar to the samples that would be generated by a learned MRF model with the same features. The state-space samples satisfy the usual moment matching constraints that defines an MRF and can be used for making meaningful predictions. In a way, herding combines learning and inference in one dynamical system. However, the distribution from which herding generates samples is not identical to the associated MRF because while the same moment matching constraints are satisfied, the entropy of the herding samples is usually somewhat lower than the (maximal) entropy of the MRF. The sequence of samples in state space $\mathbf{s}_1, .., \mathbf{s}_T$ has very interesting properties. First, it forms an infinite memory sequence as every sample depends on all the previous samples and not just the most recent sample as in Markov sequences. It can be shown that the number of distinct subsequences of length T grows as $\mathcal{O}(\log(T))$ implying that their (topological) entropy vanishes. For simple systems these sequences can be identified with "low discrepancy sequences" and Sturmian sequences (Marston Morse, 1940). Probably related to this is the fact that Monte Carlo averages based on these sequences converge as $\mathcal{O}(1/T)$. This should be contrasted with random independent samples from the associated MRF distribution for which the convergence follows the usual $\mathcal{O}(1/\sqrt{T})$ rate. Herding sequences thus exhibit strong negative auto-correlations leading to the faster convergence of Monte Carlo averages. It is conjectured that this property is related to the edge of chaos characterization of herding, and that both stochastic systems (such as samplers) as well as fully chaotic systems will always generate samples that can at most result in $\mathcal{O}(1/\sqrt{T})$ convergence of Monte Carlo averages.

Similar to "perturb and map" (Papandreou and Yuille, 2011), the execution of the herding map requires one to compute the maximum a posteriori (MAP) state defined by the current parameter setting. While maximization is sometimes easier than computing the expectations required to update the parameters of an MRF, for complex models maximization can also be NP hard. A natural question is therefore if one can relax the requirement of finding the MAP state and get away with partial maximization to, say, a local maximum instead of the global maximum. The answer to this ques-

tion comes from a theorem that was proven a long time ago in the context of Rosenblatt's perceptron (Rosenblatt, 1958) and is known as the "perceptron cycling theorem" (PCT) (Minsky and Papert, 1969). This theorem states precisely which conditions need to be fulfilled by herding at every iteration in order for the algorithm to satisfy the moment constraints. The PCT therefore allows us to relax the condition of finding the MAP state at every iteration, and as a side effect also allows us to run herding in an online setting or with stochastic minibatches instead of the entire dataset. A further relaxation of the herding conditions was described in Chen et al. (2014) where it was shown that herding with *inconsistent* moments as input (moments that can not be generated by a single joint probability distribution) still makes sense and generates the Euclidean projections of these moments on the marginal polytope.

Like MRF models can be extended to models with hidden variables and to discriminative models such as the conditional Markov random field (CRF) models, herding can also be generalized along these same dimensions. Herding with hidden variables was described in Welling (2009b) and shown to increase the ability of this dynamical system to represent complex dependencies. Conditional herding was described in Gelfand et al. (2010) and shown to be equivalent to the voted perceptron algorithm Freund and Schapire (1999) and to Collins' "voted HMM" Collins (2002) in certain special cases. The herding view allowed the extension of these discriminative models to include hidden variables.

Herding is related to (or has been connected to) a number of optimization, learning and inference methods. Herding has obvious similarities to the concept of "fast weights" introduced by Tieleman and Hinton (2009). Fast weights follow a dynamics that is designed to make the Markov chain embedded in a MRF learning process mix fast. A similar idea was used in Breuleux et al. (2011) to speed up the mixing rate of an (approximate) sampling procedure. By applying herding dynamics conditionally w.r.t. its parent-states for every variable in a graphical model yet another fast mixing sampling algorithm was developed, called "herded Gibbs" Bornn et al. (2013). Herding was extended in Chen et al. (2010) to a deterministic sampling algorithm in continuous state spaces (known as "kernel herding"). The view espoused in that paper led to an analysis of herding as a conditional gradient optimization algorithm (or Franke-Wolfe algorithm) in Bach et al. (2012) from which an improved convergence analysis emerged as well generalizations to versions of herding with non-uniform weights. In related work of Huszar and Duvenaud (2012) it was shown that an optimally weighted version of (kernel) herding is equivalent to Bayesian quadrature, again resulting in faster convergence. Harvey and Samadi (2014) focused on the convergence rate of

herding with respect to the dimensionality of the feature vector and proposed a new algorithm that scaled near-optimally with the dimensionality.

Perhaps the method closest related to herding is "perturb and map" estimation, where the parameters of a MRF model are perturbed by sampling from a Gumbel distribution followed by maximization over the states. Like in herded Gibbs, the procedure is only "exact" if exponentially many parameters are perturbed. Herding is however different from perturb and map in that the perturbations are generated sequentially and deterministically.

This chapter is built on the results reported earlier in a series of conference papers Welling (2009a,b); Welling and Chen (2010); Chen et al. (2010); Gelfand et al. (2010). Our current understanding of herding is far from comprehensive but rather represents a first attempt to connect learning systems with the theory of nonlinear dynamical systems and chaos. We believe that it opens the door to many new directions of research with potentially surprising and exciting discoveries.

The chapter is organized as follows. In Section 4.2 we introduce the herding algorithm and study its statistical property as both a learning algorithm and a dynamical system. In Section 4.3 we provide a general condition for herding to satisfy the fast moment matching properties, under which the algorithm is extended for partially observed models and discriminative models. We evaluate the performance of the introduced algorithms empirically in Section 6.4. The chapter is concluded with a summary in Section 4.5 and a conclusion in Section 4.6.

4.2 Herding Model Parameters

4.2.1 The Maximum Entropy Problem and Markov Random Fields

Define $\mathbf{x} \in \mathcal{X}$ to be a random variable in the domain $\mathcal{X}$, and $\boldsymbol{\phi} = \{\phi_\alpha(\mathbf{x})\}$ to be a set of feature functions of $\mathbf{x}$, indexed by α. In the maximum entropy problem (MaxEnt), given a data set of D observations $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^D$, we want to learn a probability distribution over $\mathbf{x}$, $P(\mathbf{x})$, such that the expected features, a.k.a. moments, match the average value observed in the data set, denoted by $\bar{\phi}_\alpha$. For the remaining degrees of freedom in the distribution we assume maximum ignorance which is expressed as maximum entropy. Mathematically, the problem is to find a distribution P such that:

$$P = \arg\max_{\mathcal{P}} \mathcal{H}(\mathcal{P}) \quad \text{s.t. } \mathbb{E}_{\mathbf{x}\sim\mathcal{P}}[\phi_\alpha(\mathbf{x})] = \bar{\phi}_\alpha, \ \forall\alpha \tag{4.1}$$

The dual form of the MaxEnt problem is known to be equivalent to finding the maximum likelihood estimate (MLE) of the parameters $\mathbf{w} = \{w_\alpha\}$ of a

Markov Random Field (MRF) defined on $\mathbf{x}$, each parameter associated with one feature ϕ_α:

$$\mathbf{w}_{\text{MLE}} = \arg\max_{\mathbf{w}} P(\mathcal{D}; \mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{D} P(\mathbf{x}_i; \mathbf{w}), \tag{4.2}$$

$$P(\mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left(\sum_{\alpha} w_\alpha \phi_\alpha(\mathbf{x})\right), \tag{4.3}$$

where the normalization term $Z(\mathbf{w}) = \sum_{\mathbf{x}} \exp(\sum_\alpha w_\alpha \phi_\alpha(\mathbf{x}))$ is also called the partition function. The parameters $\{w_\alpha\}$ act as Lagrange multipliers to enforce the constraints in the primal form 4.1. Since they assign different weights to the features in the dual form, we will also called them "weights" below.

It is generally intractable to obtain the MLE of parameters because the partition function involves computing the sum of potentially exponentially many states. Take the gradient descent optimization algorithm for example. Denote the average log-likelihood per data item by

$$\ell(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{D} \sum_{i=1}^{D} \log P(\mathbf{x}_i; \mathbf{w}) = \mathbf{w}^T \bar{\boldsymbol{\phi}} - \log Z(\mathbf{w}) \tag{4.4}$$

The gradient descent algorithm searches for the maximum of ℓ with the following update step:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta(\bar{\boldsymbol{\phi}} - \mathbb{E}_{\mathbf{x} \sim P(\mathbf{x}; \mathbf{w}_t)}[\boldsymbol{\phi}(\mathbf{x})]) \tag{4.5}$$

Notice however that the second term in the gradient that averages over the model distribution, $\mathbb{E}_{P(\mathbf{x};\mathbf{w})}[\boldsymbol{\phi}(\mathbf{x})]$, is derived from the partition function and cannot be computed efficiently in general. A common solution is to approximate that quantity by drawing samples using Markov chain Monte Carlo (MCMC) at each gradient descent step. However, MCMC is known to suffer from slow mixing when the state distribution has multiple modes or variables are strongly correlated (Neal, 1993). Furthermore, we can usually afford to run MCMC for only a few iterations in the nested loop for the sake of efficiency (Neal, 1992; Tieleman, 2008), which makes it even harder to obtain an accurate estimate of the gradient.

Even when the MRF is well trained, it is usually difficult to apply the model to regular tasks such as inference, density estimation, and model selection, because all of those tasks require the computation of the partition function. One has to once more resort to running MCMC or other approximate inference methods during the prediction phase to obtain an approximation.

Is there a method to speed up the inference step that exists in both the training and test phases? The herding algorithm was proposed to address the slow mixing problem of MCMC and combine the execution of MCMC in both training and prediction phases into a single process.

4.2.2 Learning MRFs with Herding

When there exist multiple local modes in a model distribution, an MCMC sampler is prone to getting stuck in local modes and it becomes difficult to explore the state space efficiently. However, that is not a serious issue at the beginning of the MRF learning procedure as observed by, for example, Tieleman and Hinton (2009). This is because the parameters keep being updated with a large learning rate η at the beginning. Specifically, when the expected feature vector is approximated by a set of samples $\mathbb{E}_{P(\mathbf{x};\mathbf{w})}[\boldsymbol{\phi}(\mathbf{x})] \approx \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{\phi}(\mathbf{x}_m)$ in the MCMC approach, after each update in Equation 4.5, the parameter $\mathbf{w}$ is translated along the direction that tends to reduce the inner product of $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_m)$, and thereby reduces the state probability around the region of the current samples. This change in the state distribution helps the MCMC sampler escape local optima and mix faster.

This observation suggests that we can speed up the MCMC algorithm by updating the target distribution itself with a large learning rate. However, in order to converge to a point estimate of a model, η needs to be decreased using some suitable annealing schedule. But one may ask if we are necessarily interested in a fixed value for the model parameters? As discussed in the previous subsection, for many applications one needs to compute averages over the (converged) model which are intractable anyway. In that case, a sequence of samples to approximate the averages is all we need. It then becomes a waste of resources and time to nail down a single point estimate of the parameters by decreasing η when a sequence of samples is already available. We will actually kill two birds with one stone by obtaining samples during the training phase and reuse them for making predictions. The idea of the herding algorithm originates from this observation.

The herding algorithm proposed in Welling (2009a) can be considered as an algorithm that runs a gradient descent algorithm with a constant learning rate on an MRF in the zero-temperature limit. Define the distribution of an MRF with a temperature by replacing $\mathbf{w}$ with $\mathbf{w}/T$, where T is an artificial temperature variable. The log-likelihood of a model (multiplied by T) then becomes:

$$\ell_T(\mathbf{w}) = \mathbf{w}^T\bar{\boldsymbol{\phi}} - T\log\left(\sum_{\mathbf{x}}\exp\left(\sum_{\alpha}\frac{w_\alpha}{T}\phi_\alpha(\mathbf{x})\right)\right) \tag{4.6}$$

When T approaches 0, all the probability is absorbed into the most probable state, denoted as $\mathbf{s}$, and the expectation of the feature vector, $\bar{\boldsymbol{\phi}}$, equals that of state $\mathbf{s}$. The herding algorithm then consists of the iterative gradient descent updates in the limit, $T \to 0$, with a constant learning rate, η:

$$\mathbf{s}_t = \arg\max_{\mathbf{x}} \sum_{\alpha} w_{\alpha,t-1}\phi_\alpha(\mathbf{x}) \tag{4.7}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta(\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{s}_t)) \tag{4.8}$$

We usually set $\eta = 1$ except when mentioned explicitly because the herding dynamics is invariant to the learning rate as explained in Section 4.2.3. We treat the sequence of most probable states, $\{\mathbf{s}_t\}$, as a set of "samples" for herding and use it for inference tasks. At each iteration, we find the most probable state in the current model distribution deterministically, and update the parameter towards the average feature vector from the training data subtracted by the feature vector of the current sample. Compared to maintaining a set of random samples in the MCMC approach (see e.g. Tieleman, 2008), updating $\mathbf{w}$ with a single sample state facilitates updating the distribution at an even rate.

If we divide both sides of Equation 4.8 by T and redefine $\frac{\mathbf{w}}{T} \to \mathbf{w}'$ in both Equations 4.7-4.8,

$$\frac{\mathbf{w}_{t+1}}{T} = \frac{\mathbf{w}_t}{T} + \frac{\eta}{T}(\bar{\boldsymbol{\phi}} - \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x};\frac{\mathbf{w}_t}{T})}[\boldsymbol{\phi}(\mathbf{x})]) \tag{4.9}$$

we see that, after taking the limit $T \to \infty$, we can interpret herding as maximum likelihood learning with infinitely large stepsize and rescaled weights. The surprising observation is that the state sequence $\{\mathbf{s}_t\}$ generated by this process is still meaningful and can be interpreted as approximate samples from an MRF model with the correct moment constraints on the features $\boldsymbol{\phi}(\mathbf{x})$.

One can obtain an intuitive impression of the dynamics of herding by looking at the change in the asymptotic behavior of the gradient descent algorithm as we decrease T in Equation 4.9 from a large value towards 0. Assume that we can compute the expected feature vector w.r.t. the model exactly. Given an initial value of $\mathbf{w}$, the gradient descent update equation 4.9 with a constant learning rate is a deterministic mapping in the parameter space. When T is large enough (η/T is small enough), the optimization process will converge and $\mathbf{w}/T$ will approach a single point which is the MLE. As T decreases below some threshold (η/T is above some threshold), the convergence condition is violated and the trajectory of $\mathbf{w}_t$ will move asymptotically into an oscillation between two points, that is,

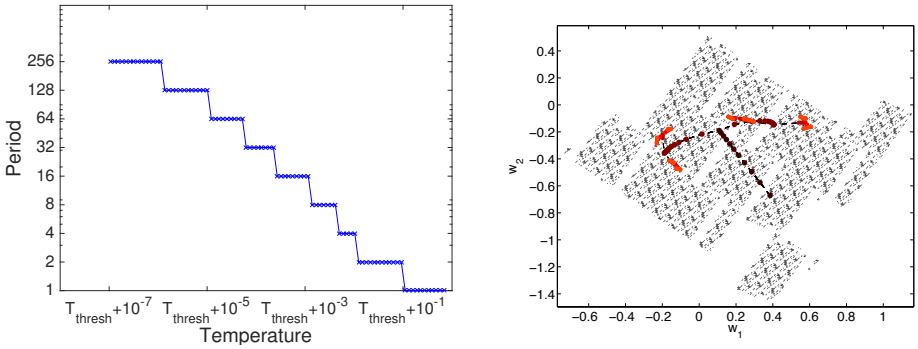


Figure 4.1: Attractor bifurcation for a model with 4 states and 2-dimensional feature vectors. Left: Asymptotic period of the weight sequence (i.e. size of the attractor set) repeatedly doubles as the temperature decreases towards a threshold value (right to left). $T_{thresh} \approx 0.116$ in this example. The dynamics transits from periodic to aperiodic at that threshold. Right: The evolution of the attractor set of the weight sequence. As the temperature decreases (from dark to light colors), the attractor set split from a single point to two points, then to four, to eight, etc. The black dot cloud in the background is the attractor set at $T = 0$.

the attractor set splits from a single point into two points. As T decreases further, the asymptotic oscillation period doubles from two to four, four to eight, etc, and eventually the process approaches an infinite period at another temperature threshold. Figure 4.1 shows an example of the attractor bifurcation phenomenon. The example model has 4 discrete states and each state is associated with 2 real valued features which are randomly sampled from $\mathcal{N}(0, 1)$. Starting from that second threshold, the trajectory of $\mathbf{w}$ is still bounded in a finite region as shown shortly in Section 4.3.1 but will not be periodic any more. Instead, we observe that the dynamics often converges to a fractal attractor set as shown in the right plot of Figure 4.1. The bifurcation process is observed very often in simulated models although it is not clear to us if it always happens for any discrete MRF. We discuss the dynamics related to this phenomenon in more detail in Section 4.2.6.

4.2.3 Tipi Function and Basic Properties of Herding

We will discuss a few distinguishing properties of the herding algorithm in this subsection. When we take the zero temperature limit in Equation 4.6, the log-likelihood function becomes

$$\ell_0(\mathbf{w}) = \mathbf{w}^T \bar{\boldsymbol{\phi}} - \max_{\mathbf{x}} \left[\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\right] \tag{4.10}$$

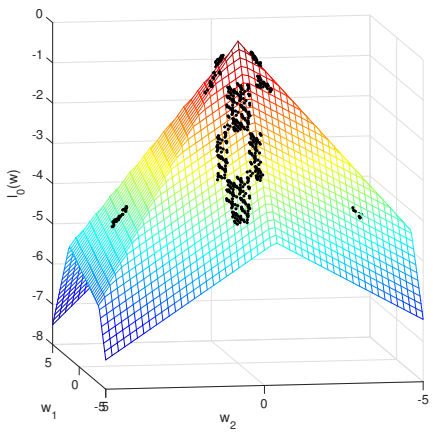


Figure 4.2: "Tipi function" (Welling, 2009a): the log-likelihood function at the zero temperature limit. The black dots show the attractor set of the sequence of $\mathbf{w}_t$.

This function has a number of interesting properties that justify the name "Tipi function"[1] (see Figure 4.2) (Welling, 2009a):

1. ℓ_0 *is continuous piecewise linear* (C^0 *but not* C^1). It is clearly linear in $\mathbf{w}$ as long as the maximizing state $\mathbf{s}$ does not change. However, changing $\mathbf{w}$ may in fact change the maximizing state in which case the gradient changes discontinuously.

2. ℓ_0 *is a concave, non-positive function of* $\mathbf{w}$ *with a maximum at* $\ell_0(\mathbf{0}) = 0$. This is true because the first term represents the average $\mathbb{E}_P[\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})]$ over some distribution P, while the second term is its maximum. Therefore, $\ell \leqq 0$. If we furthermore assume that $\boldsymbol{\phi}$ is not constant on the support of P then $\ell_0 < 0$ and the maximum at $\mathbf{w} = 0$ is unique. Concavity follows because the first term is linear and the second maximization term is convex.

3. ℓ_0 *is scale free.* This follows because $\ell_0(\beta\mathbf{w}) = \beta\ell_0(\mathbf{w}), \forall\beta \geq 0$ as can be easily checked. This means that the function has exactly the same structure at any scale of $\mathbf{w}$.

Herding runs gradient descent optimization on this Tipi function. There is no need to search for the maximum as $\mathbf{w} = 0$ is the trivial solution. However, the fixed learning rate will always result in a perpetual overshooting of the maximum and thus the sequence of weights will never converge to a fixed point. Every flat face of the Tipi function is associated with a state. An important property of herding is that the state sequence visited by the

1. A Tipi is a traditional native Indian dwelling.

gradient descent procedure satisfies the moment matching constraints in Equation 4.1, which will be discussed in details in Section 4.2.5. There are a few more properties of this procedure that are worth noticing.

Deterministic Nonlinear Dynamics

Herding is a deterministic nonlinear dynamical system. In contrast to the stochastic MLE learning algorithm based on MCMC, the two update steps in Equation 4.7 and 4.8 consist of a nonlinear deterministic mapping of the weights as illustrated in Figure 4.3. In particular it is not an MCMC procedure and it does not require random number generation.

The dynamics thus produces pseudo-samples that look random, but should not be interpreted as random samples. Although reminiscent of the Bayesian approach, the weights generated during this dynamics should not be interpreted as samples from some Bayesian posterior distribution. We will discuss the weakly chaotic behavior of the herding dynamics in detail in Section 4.2.6.

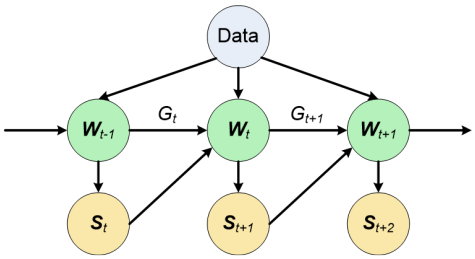


Figure 4.3: Herding as a nonlinear dynamical system.

Invariance to the Learning Rate

Varying the learning rate η does not change the behavior of the herding dynamics. The only effect is to change the scale of the invariant attractor set of the sequence $\mathbf{w}_t$. This actually follows naturally from the scale-free property of the Tipi function. More precisely, denote with $\mathbf{v}_t$ the standard herding sequence with $\eta = 1$ and $\mathbf{w}_t$ the sequence with an arbitrary learning rate. It is easy to see that if we initialize $\mathbf{v}_{t=0} = \frac{1}{\eta}\mathbf{w}_{t=0}$ and apply the respective herding updates for $\mathbf{w}_t$ and $\mathbf{v}_t$ afterwards, the relation $\mathbf{v}_t = \frac{1}{\eta}\mathbf{w}_t$ will remain true for all $t > 0$. In particular, the states $\mathbf{s}_t$ will be the same for both sequences. Therefore we simply set $\eta = 1$ in the herding algorithm.

Of course, if one initializes both sequences with arbitrary different values, then the state sequences will not be identical. However, if one accepts the conjecture that there is a unique invariant attractor set, then this difference can be interpreted as a difference in initialization which only affects the transient behavior (or "burn-in" behavior) but not the (marginal) distribution $P(\mathbf{s})$ from which the states $\mathbf{s}_t$ will be sampled.

Notice however that if we assign different learning rates $\{\eta_\alpha\}$ across the dimensions of the weight vector $\{w_\alpha\}$, it will change the distribution $P(\mathbf{s})$. While the moment matching constraints are still satisfied, we notice that the entropy of the sample distribution varies as a function of $\{\eta_\alpha\}$. In fact, changing the relative ratio of learning rates among feature dimensions is equivalent to scaling features with different factors in the greedy moment matching algorithm interpretation of Section 4.2.4. How to choose an optimal set of learning rates is still an open problem.

Negative Auto-correlation

A key advantage of the herding algorithm we observed in practice over sampling using a Markov chain is that the dynamical system mixes very rapidly over the attractor set. This is attributed to the fact that maximizations are performed on an ever changing model distribution as briefly mentioned at the beginning of this subsection. Let $\pi(\mathbf{x})$ be the distribution of training data $\mathcal{D}$, and $\mathbf{s}_t$ be the maximizing state at time t. The distribution of an MRF at time t with a regular temperature $T = 1$ is

$$P(\mathbf{x}; \mathbf{w}_{t-1}) \propto \exp(\mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x})) \tag{4.11}$$

After the weights are updated with Equation 4.8, the probability of the new model becomes

$$\begin{aligned} P(\mathbf{x}; \mathbf{w}_t) &\propto \exp(\mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{x})) = \exp((\mathbf{w}_{t-1} + \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{s}_t))^T \boldsymbol{\phi}(\mathbf{x})) \\ &= \exp\left(\mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}) + \sum_{\mathbf{y} \neq \mathbf{s}_t} \pi(\mathbf{y}) \boldsymbol{\phi}(\mathbf{y})^T \boldsymbol{\phi}(\mathbf{x}) - (1 - \pi(\mathbf{s}_t)) \boldsymbol{\phi}(\mathbf{s}_t)^T \boldsymbol{\phi}(\mathbf{x}) \right) \end{aligned} \tag{4.12}$$

Comparing Equation 4.12 with 4.11 we see that probable states (with large $\pi(\mathbf{x})$) are rewarded with an extra positive term $\pi(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{x})$, *except* the most recently sampled state $\mathbf{s}_t$. This will have the effect (after normalization) that state $\mathbf{s}_t$ will have a smaller probability of being selected again. Imagine for instance that the sampler is stuck at a local mode. After drawing samples at that mode for a while, weights are updated to gradually reduce that mode

and help the sampler escape it. The resulting negative auto-correlation would help mitigate the notorious problem of positive auto-correlation in most MCMC methods.

We illustrate the negative auto-correlation using a synthetic MRF with 10 discrete states, each associated with a 7-dimensional feature vector. The parameters of the MRF model are randomly generated from which the expected feature values are then computed analytically and fed into the herding algorithm to draw $T = 10^5$ samples. We define the auto-correlation of the sample sequence of discrete variables as follows:

$$R(t) = \frac{\frac{1}{T-t}\sum_{\tau=1}^{T-t}\mathbb{I}[s_\tau = s_{\tau+t}] - \sum_s \frac{1}{2}tP(s)^2}{1 - \sum_s \frac{1}{2}tP(s)^2} \tag{4.13}$$

where $\mathbb{I}$ is the indication function and $\frac{1}{2}tP$ is the empirical distribution of the 10^5 samples. It is easy to observe that $R(t = 0) = 1$ and if the samples are independently distributed $R(t) = 0, \forall t > 0$ up to a small error due to the finite sample size. We run herding 100 times with different model parameters and show the mean and standard deviation of the auto-correlation in Figure 4.4. We can see that the auto-correlation is negative for neighboring samples, and converges to 0 as the time lag increases. This effect exists even if we use a local optimization algorithm when a global optimum is hard or expensive to be obtained. This type of "self-avoidance" is also shared with other sampling methods such as over-relaxation (Young, 1954), fast-weights PCD (Tieleman and Hinton, 2009) and adaptive MCMC (Salakhutdinov, 2010).

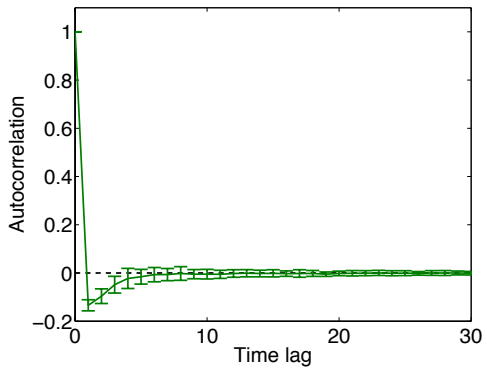


Figure 4.4: Negative auto-correlation of herding samples from a synthetic MRF.

4.2.4 Herding as a Greedy Moment Matching Algorithm

As herding does not obtain the MLE, the distribution of the generated samples does not provide a solution to the maximum entropy problem either. However, we observe that the moment matching constraints in Equation 4.1 are still respected, that is, when we compute the sampling average of the feature vector it will converge to the input moments. Furthermore, the negative auto-correlation in the sample sequence helps to achieve a convergence rate that is faster than what one would get from independently drawing samples or running MCMC at the MLE. Before providing any quantitative results, it would be easier for us to understand herding intuitively by taking a "dual view" of its dynamics where we remove weights $\mathbf{w}$ in favor of the states $\mathbf{x}$ (Chen et al., 2010).

Notice that the expression of $\mathbf{w}_T$ can be expanded recursively using the update Equation 4.8:

$$\mathbf{w}_T = \mathbf{w}_0 + T\bar{\boldsymbol{\phi}} - \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{s}_t) \tag{4.14}$$

Plugging 4.14 into Equation 4.7 results in

$$\mathbf{s}_{T+1} = \arg\max_{\mathbf{x}} \langle \mathbf{w}_0, \boldsymbol{\phi}(\mathbf{x}) \rangle + T\langle \bar{\boldsymbol{\phi}}, \boldsymbol{\phi}(\mathbf{x}) \rangle - \sum_{t=1}^{T} \langle \boldsymbol{\phi}(\mathbf{s}_t), \boldsymbol{\phi}(\mathbf{x}) \rangle \tag{4.15}$$

For ease of intuitive understanding of herding, we temporarily make the assumptions (which are not necessary for the propositions to hold in the next subsection):

1. $\mathbf{w}_0 = \bar{\boldsymbol{\phi}}$
2. $\|\boldsymbol{\phi}(\mathbf{x})\|_2 = R, \forall \mathbf{x} \in \mathcal{X}$

The second assumption is easily achieved, e.g. by renormalizing $\boldsymbol{\phi}(\mathbf{x}) \leftarrow \frac{\boldsymbol{\phi}(\mathbf{x})}{\|\boldsymbol{\phi}(\mathbf{x})\|}$ or by choosing a suitable feature map $\boldsymbol{\phi}$ in the first place. Given the first assumption, Equation 4.15 becomes

$$\mathbf{s}_{T+1} = \arg\max_{\mathbf{x}} \langle \bar{\boldsymbol{\phi}}, \boldsymbol{\phi}(\mathbf{x}) \rangle - \frac{1}{T+1} \sum_{t=1}^{T} \langle \boldsymbol{\phi}(\mathbf{s}_t), \boldsymbol{\phi}(\mathbf{x}) \rangle \tag{4.16}$$

Combining the second assumption one can show that the herding update equation 4.16 is equivalent to greedily minimizing the squared error $\mathcal{E}_T^2$

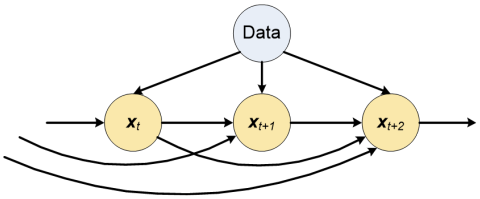


Figure 4.5: Herding as an infinite memory process on samples.

defined as

$$\mathcal{E}_T^2 \stackrel{\text{def}}{=} \left\| \bar{\boldsymbol{\phi}} - \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{s}_t) \right\|^2 \tag{4.17}$$

We therefore see that herding will generate pseudo-samples that greedily minimize the distance between the input moments and the sampling average of the feature vector at every iteration (conditioned on past samples). Note that the error function is unfortunately not submodular and the greedy procedure does not imply that the total collection of samples at iteration T is jointly optimal (see Huszar and Duvenaud (2012) for a detailed discussion). We also note that herding is an "infinite memory process" on $\mathbf{s}_t$ (as opposed to a Markov process) illustrated in Figure 4.5 because new samples depend on the entire history of samples generated thus far.

4.2.5 Moment Matching Property

With the dual view in the previous subsection, the distance between the moments and their sampling average in Equation 4.17 can be considered as the objective function for the herding algorithm to minimize. We discuss in this subsection under what condition and at what speed the moment constraints will be eventually satisfied.

Proposition 4.1 (Proposition 1 in Welling (2009a))**.** $\forall \alpha$, *if* $\lim_{\tau\to\infty} \frac{1}{\tau} w_{\alpha\tau} = 0$, *then* $\lim_{\tau\to\infty} \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{s}_t) = \bar{\phi}_\alpha$.

Proof. Following Equation 4.14, we have

$$\frac{1}{\tau} w_{\alpha\tau} - \frac{1}{\tau} w_{\alpha 0} = \bar{\phi}_\alpha - \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{s}_t) \tag{4.18}$$

Using the premise that the weights grow slower than linearly and observing that $w_{\alpha 0}$ is constant we see that the left hand term vanishes in the limit $\tau \to \infty$ which proves the result. □

What this says is that under the very general assumption that the weights do not grow linearly to infinity (note that due to the finite learning rate they can not grow faster than linear either), the moment constraints will be satisfied by the samples collected from the combined learning/sampling procedure. In fact, we will show later that the weights are restricted in a bounded region, which leads to a convergence rate of $\mathcal{O}(1/\tau)$ as stated below.

Proposition 4.2. *$\forall\alpha$, if there exists a constant R such that $|w_{\alpha,t}| \leq R, \forall t$, then*

$$\left| \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{s}_t) - \bar{\phi}_\alpha \right| \leq \frac{2R}{\tau}.$$

The proof follows immediately Equation 4.18.

Note that if we want to estimate the expected feature of a trained MRF by a Monte Carlo method, the optimal standard deviation of the approximation error with independent and identically distributed (i.i.d.) random samples decays as $\mathcal{O}(\frac{1}{\sqrt{\tau}})$, where τ is the number of samples. (For positively autocorrelated MCMC methods this rate could be even slower.) Samples from herding therefore achieve a faster convergence rate in estimating moments than i.i.d. samples.

Recurrence of the Weight Sequence

It is important to ensure that the herding dynamics does not diverge to infinity. Welling (2009a) discovered an important property of herding, known as recurrence, that the sequence of the weights is confined in a ball in the parameter space. This property satisfies the premise of both Proposition 2.1 and 2.2. It was stated in a corollary of Proposition 4.3:

Proposition 4.3 (Proposition 2 in Welling (2009a))**.** *$\exists\mathcal{R}$ such that a herding update performed outside this radius will always decrease the norm $\|\mathbf{w}\|_2$.*

Corollary 4.4 (Corollary in Welling (2009a))**.** *$\exists\mathcal{R}'$ such that a herding algorithm initialized inside a ball with that radius will never generate weights $\mathbf{w}$ with norm $\|\mathbf{w}\|_2 > \mathcal{R}'$.*

However, there was a gap in the proof of Proposition 2 in Welling (2009a). We give the corrected proof below:

Proof of Proposition 4.3. Write the herding update equation 4.8 as $\mathbf{w}_t = \mathbf{w}_{t-1} + \nabla_{\mathbf{w}}\ell_0(\mathbf{w}_{t-1})$ (set $\eta = 1$). Expanding the squared norm of $\mathbf{w}_t$ leads to

$$\begin{aligned} &\|\mathbf{w}_t\|_2^2 = \|\mathbf{w}_{t-1}\|_2^2 + 2\mathbf{w}_{t-1}^T\nabla_{\mathbf{w}}\ell_0(\mathbf{w}_{t-1}) + \|\nabla_{\mathbf{w}}\ell_0(\mathbf{w}_{t-1})\|_2^2 \\ &\implies \quad \delta\|\mathbf{w}\|_2^2 < 2\ell_0(\mathbf{w}_{t-1}) + \mathcal{B}^2 \end{aligned} \tag{4.19}$$

where we define $\delta\|\mathbf{w}\|_2^2 = \|\mathbf{w}_t\|_2^2 - \|\mathbf{w}_{t-1}\|_2^2$. $\mathcal{B}$ is an upper bound of $\{\|\nabla_{\mathbf{w}}\ell_0(\mathbf{w})\|_2 : \mathbf{w} \in \mathcal{R}^{|\mathbf{w}|}\}$ introduced in Lemma 1 of Welling (2009a). That exists as long as the norm of the feature vector $\boldsymbol{\phi}(\mathbf{x})$ is bounded in $\mathcal{X}$. We also use the fact that $\ell_0(\mathbf{w}) = \mathbf{w}^T\nabla_{\mathbf{w}}\ell_0(\mathbf{w})$.

Denote the unit hypersphere as $U = \{\mathbf{w}|\|\mathbf{w}\|_2^2 = 1\}$. Since ℓ_0 is continuous on U and U is a bounded closed set, ℓ_0 can achieve its supremum on U, that is, we can find a maximum point w^* on U where $\ell_0(\mathbf{w}^*) \geq \ell_0(\mathbf{w}), \forall \mathbf{w} \in U$.

Combining this with the fact that $\ell_0 < 0$ outside the origin, we know the maximum of ℓ_0 on U is negative. Now taking into account the fact that $\mathcal{B}$ is constant (i.e. does not scale with $\mathbf{w}$), there exists some constant $\mathcal{R}$ for which $\mathcal{R}\ell_0(\mathbf{w}^*) < -\mathcal{B}^2/2$. Together with the scaling property of ℓ_0, $\ell_0(\beta\mathbf{w}) = \beta\ell_0(\mathbf{w})$, we can prove that for any $\mathbf{w}$ with a norm larger than $\mathcal{R}$, ℓ_0 is smaller then $-\mathcal{B}^2/2$:

$$\ell_0(\mathbf{w}) = \|\mathbf{w}\|_2\ell_0(\mathbf{w}/\|\mathbf{w}\|_2) \leq \mathcal{R}\ell_0(\mathbf{w}^*) < -\mathcal{B}^2/2, \quad \forall\|\mathbf{w}\|_2 > R \tag{4.20}$$

The proof is concluded by plugging the inequality above in Equation 4.19. □

Corollary 4.4 proves the existence of a bound for $\|\mathbf{w}\|_2$ and thereby the constant R in Proposition 4.2. Harvey and Samadi (2014) further studied the value of R and proposed a variant of herding that obtained a near-optimal value for $R = O(\sqrt{d}\log^{2.5}\|\mathcal{X}\|)$ w.r.t. the dimensionality of the feature vector d and the size of a finite state space $\mathcal{X}$. The proposed algorithm has a polynomial time complexity in d and $\|\mathcal{X}\|$.

The Remaining Degrees of Freedom

Both the herding and the MaxEnt methods match the moments of the training data. But how does herding control the remaining degrees of freedom that are otherwise controlled by maximizing the entropy in the MaxEnt method? This is unfortunately still an open problem. Apart from some heuristics there is currently no principled way to enforce high entropy. In practice however, in discrete state spaces we usually observe that the sampling distribution from herding renders high entropy. We illustrate the behavior of herding in the example of simulating an Ising model in the next paragraph.

An Ising model is an MRF defined on a lattice of binary nodes, $G = (E, V)$, with biases and pairwise features. The probability distribution is expressed as

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(\beta\left(\sum_{(i,j)\in E} J_{i,j} x_i x_j + \sum_{i\in V} h_i x_i\right)\right), x_i \in \{-1, 1\}, \forall i \in V \tag{4.21}$$

where h_i is the bias parameter, $J_{i,j}$ is the pairwise parameter and $\beta \geq 0$ is the inverse temperature variable. When $h_i = 0$, $J_{i,j} = 1$ for all nodes and edges, and β is set at the inverse critical temperature, the Ising model is said to be at a critical phase where regular sampling algorithms fail due to long range correlations among variables. A special algorithm, the Swendsen-Wang algorithm (Swendsen and Wang, 1987), was designed to draw samples efficiently in this case. In order to run herding on the Ising model, we need to know the average features, $\bar{x}_i$ (0 in this case) and $\overline{x_i x_j}$ instead of the MRF parameters. So we first run the Swendsen-Wang algorithm to obtain an estimate of the expected cross terms, $\overline{x_i x_j}$, which are constant across all edges, and then run herding with weights for every node w_i and edge $w_{i,j}$. The update equations are:

$$\mathbf{s}_t = \operatorname*{argmax}_{\mathbf{x}} \sum_{(i,j)\in E} w_{(i,j),t-1} x_i x_j + \sum_{i\in V} w_{i,t-1} x_i \tag{4.22}$$

$$w_{(i,j),t} = w_{(i,j),t-1} + \overline{x_i x_j} - s_{i,t} s_{j,t} \tag{4.23}$$

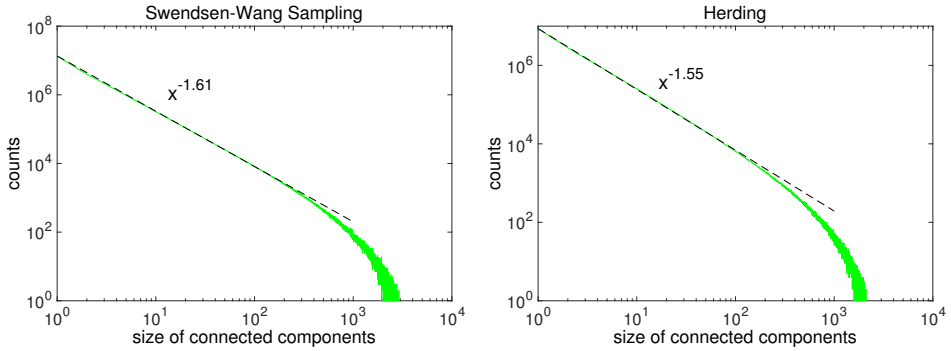
$$w_{i,t} = w_{i,t-1} - s_{i,t} \tag{4.24}$$

As finding the global optimum is an NP-hard problem we find a local maximum for $\mathbf{s}_t$ by coordinate descent[2]. Figure 4.6 shows a sample from an Ising model on an 100×100 lattice at the critical temperature. We do not observe qualitative difference between the samples generated by the Ising model (MaxEnt) and herding, which suggests that the sample distribution of herding may be very close to the distribution of the MRF. Furthermore, Figure 4.7 shows the distribution of the size of connected components in the samples. It is known that this distribution should obey a power law at the critical temperature. We find that samples from both methods exhibit the power law distribution with an almost identical exponent.

2. In Section 4.3.2 we show that the moment matching property still holds with a local search as long as the found state is better than the average.

(a) Generated by Swendsen-Wang (b) Generated by Herding

Figure 4.6: Sample from an Ising model on an 100×100 lattice at the critical temperature.



(a) Generated by Swendsen-Wang (b) Generated by Herding

Figure 4.7: Histogram of the size of connected components in the samples of the Ising model at the critical temperature.

4.2.6 Learning Using Weak Chaos

There are two theoretical frameworks for statistical inference: the frequentist and the Bayesian paradigm. A frequentist assumes a true objective value for some parameter and tries to estimate its value from samples. Except for the simplest models, estimation usually involves an iterative procedure where the value of the parameter is estimated with increasing precision. In information theoretic terms, this means that more and more information from the data is accumulated in more decimal places of the estimate. With a finite data-set, this process should stop at some scale because there is not enough information in the data that can be transferred into the decimal places of the parameter. If we continue anyway, we will overfit to the dataset

at hand. In a Bayesian setting we entertain a posterior distribution over parameters, the spread, or more technically speaking, entropy, of which determines the amount of information it encodes. In Bayesian estimation, the spread automatically adapts itself to the amount of available information in the data. In both cases, the learning process itself can be viewed as a dynamical system. For a frequentist this means a convergent series of parameter estimates indexed by the learning iteration $\mathbf{w}_1, \mathbf{w}_2, \ldots$. For a Bayesian running a MCMC procedure this means a stochastic process converging to some equilibrium distribution. Herding introduces a third possibility by encoding all the information in a deterministic nonlinear dynamical system. We focus on studying the weakly chaotic behavior of the herding dynamics in this subsection. The sequence of weights never converges but traces out a quasi-periodic trajectory on an attractor set which is often found to be of fractal dimension. In the language of iterated maps, $\mathbf{w}_{t+1} = F(\mathbf{w}_t)$, a (frequentist) optimization of some objective results in an attractor set that is a single point, Bayesian posterior inference results in a (posterior) probability distribution while herding will result in a (possibly fractal) attractor set which seems harder to meaningfully interpret as a probability distribution.

Example: Herding a Single Neuron

We first study an example of the herding dynamics in its simplest form and show its equivalence to some well-studied theories in mathematics. Consider a single (artificial) neuron, which can take on two distinct states: either it fires ($x = 1$) or it does not fire ($x = 0$). Assume that we want to simulate the activity of a neuron with an irrational firing rate, $\pi \in [0, 1]$, that is, the average firing frequency approaches $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} s_t = \pi$. We can achieve that by applying the herding algorithm with a one-dimensional feature $\phi(x) = x$ and feeding the input moment with the desired rate $\bar{\phi} = \pi$. Applying the update equations 4.7-4.8 we get the following dynamics:

$$s_t = \mathbb{I}(w_{t-1} > 0) \tag{4.25}$$

$$w_t = w_{t-1} + \pi - s_t \tag{4.26}$$

where $\mathbb{I}[\cdot]$ is the indicator function. With the moment matching property we can show immediately that the firing rate converges to the desired value π for any initial value of w. The update equations are illustrated in Figure 4.8. This dynamics is a simple type of interval translation mapping (ITM) problem in mathematics (Boshernitzan and Kornfeld, 1995). In a general ITM problem, the invariant set of the dynamics often has a fractal

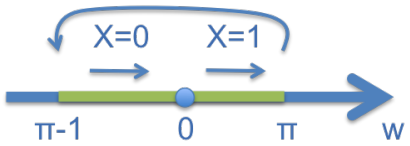


Figure 4.8: Herding dynamics for a single binary variable. At every iteration the weight is first increased by π. If w was originally positive, it is then depressed by 1.

dimension. But for this simple case, the invariant set is the entire interval $(\pi - 1, \pi]$ if π is an irrational number and a finite set if it is rational. As a neuron model, one can think of w_t as a "synaptic strength." At each iteration the synaptic strength increases by an amount π. When the synaptic strength rises above 0, the neuron fires. If it fires its synaptic strength is depressed by a factor 1. The value of w_0 only has some effect on the transient behavior of the resulting sequence $s_1, s_2, \ldots$.

It is perhaps interesting to note that by setting $\pi = \varphi$ with φ the golden mean $\varphi = \frac{1}{2}(\sqrt{5} - 1)$ and initializing the weights at $w_0 = 2\varphi - 1$, we exactly generate the "Rabbit Sequence": a well studied Sturmian sequence which is intimately related with Fibonacci numbers[3]). In Figure 4.9 we plot the weights (a) and the states (b) resulting from herding with the "Fibonacci neuron" model. For a proof, please see Welling and Chen (2010).

When initializing $w_0 = 0$, one may think of the synaptic strength as an error potential that keeps track of the total error so far. One can further show that the sequence of states is a discrete low discrepancy sequence (Angel et al., 2009) in the following sense:

Proposition 4.5. *If w is the weight of the herding dynamics for a single binary variable x with probability $P(x = 1) = \pi$, and $w_\tau \in (\pi - 1, \pi]$ at some step $\tau \geq 0$, then $w_t \in (\pi - 1, \pi], \forall t \geq \tau$. Moreover, for $T \in \mathbb{N}$, we have:*

$$\left| \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 1] - T\pi \right| \leq 1, \quad \left| \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 0] - T(1 - \pi) \right| \leq 1 \qquad (4.27)$$

Proof. We first show that $(\pi - 1, \pi]$ is the invariant interval for herding dynamics. Denote the mapping of the weight in Equation 4.25 and 4.26 as

3. Imagine two types of rabbits: young rabbits (0) and adult rabbits (1). At each new generation the young rabbits grow up ($0 \to 1$) and old rabbits produce offspring ($1 \to 10$). Recursively applying these rules we produce the rabbit sequence: $0 \to 1 \to 10 \to 101 \to 10110 \to 10110101$ etc. The total number of terms of these sequences and incidentally also the total number of 1's (lagged by one iteration) constitutes the Fibonacci sequence: $1, 1, 2, 3, 5, 8, \ldots$.

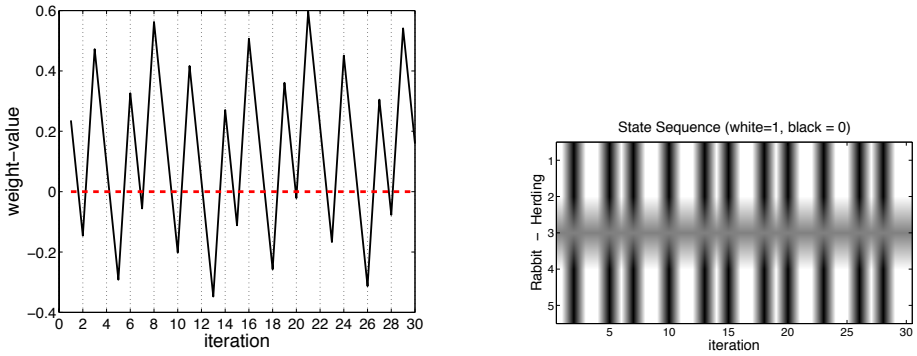


Figure 4.9: Sequence of weights and states generated by the "Fibonacci neuron" based on herding dynamics. Left: Sequence of weight values. Note that the state results by checking if the weight value is larger than 0 (in which case $s_t = 1$) or smaller than 0 (in which case $s_t = 0$). By initializing the weights at $w_0 = 2\varphi - 1$ and using $\pi = \varphi$, with φ the golden mean, we obtain the Rabbit sequence (see main text). Right: Top stripes show the first 30 iterates of the sequence obtained with herding. For comparison we also show the Rabbit sequence below it (white indicates 1 and black indicates 0). Note that these two sequences are identical.

$\mathcal{T}$. Then we can see that the interval $(\pi - 1, \pi]$ is mapped to itself as

$$\mathcal{T}(\pi-1,\pi] = \mathcal{T}(\pi-1,0]\cup\mathcal{T}(0,\pi] = (2\pi-1,\pi]\cup(\pi-1,2\pi-1] = (\pi-1,\pi] \quad (4.28)$$

Consequently when w_τ falls inside the invariant interval, we have $w_t \in (\pi - 1, \pi], \forall t \geq \tau$. Now summing up both sides of Equation 4.26 over t immediately gives us the first inequality in 4.27 as:

$$T\pi - \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 1] = w_{\tau+T} - w_\tau \in [-1, 1]. \quad (4.29)$$

The second inequality follows by observing that $\mathbb{I}[s_t = 0] = 1 - \mathbb{I}[s_t = 1]$. □

As a corollary of Proposition 4.5, when we initialize $w_0 = \pi - 1/2$, we can improve the bound of the discrepancy by a half.

Corollary 4.6. *If w is the weight of the herding dynamics in Proposition 4.5 and it is initialized at $w_0 = \pi - 1/2$, then for $T \in \mathbb{N}$, we have:*

$$\left|\sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 1] - T\pi\right| \leq \frac{1}{2}, \quad \left|\sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 0] - T(1-\pi)\right| \leq \frac{1}{2} \quad (4.30)$$

The proof immediately follows Equation 4.29 by plugging $\tau = 0$ and $w_0 = \pi - 1/2$. In fact, setting $w_0 = \pi - 1/2$ corresponds to the condition in the greedy algorithm interpretation in Section 4.2.4. One can see this

by constructing an equivalent herding dynamics with a feature of constant norm as:

$$\phi'(x) = \begin{cases} 1 & \text{if } x = 1 \\ -1 & \text{if } x = 0 \end{cases} \tag{4.31}$$

When initializing the weight at the moment $w'_0 = \bar{\phi'} = 2\pi - 1$, one can verify that this dynamics generates the same sample sequence as the original one and their weights are the same up to a constant factor of 2, i.e. $w'_t = 2w_t, \forall t \geq 0$. The new dynamics satisfies the two assumptions in Section 4.2.4 and therefore the sample sequences in both dynamical systems greedily minimize the error of the empirical probability (up to a constant factor):

$$\left| \frac{1}{T} \sum_{t=1}^{T} \phi'(x'_t) - (2\pi - 1) \right| = 2 \left| \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}[x_t = 1] - \pi \right| \tag{4.32}$$

This greedy algorithm actually achieves the optimal bound one can get with herding dynamics in the 1-neuron model, which is 1/2.

Example: Herding a Discrete State Variable

The application of herding to a binary variable can be extended naturally to a discrete state variables. Let x be a variable that can take one of the D states, $\{0, 1, \ldots, D-1\}$. Given any distribution over these D states in the set $\boldsymbol{\pi} \in \mathbb{R}^D, \sum_{d=0}^{D-1} \pi_d = 1$, we can run herding to simulate the activity of the discrete variable. The feature function, $\boldsymbol{\phi}(x)$, is defined as the 1-of-D encoding of the discrete state, that is, a vector of D binary numbers, in which all the numbers are 0 except for the element indexed by the value of x. For example, for a variable with 4 states, the feature function of $\boldsymbol{\phi}(x = 3)$ is $[0, 0, 1, 0]$. It is easy to observe that the expected value of the feature vector under the distribution $\boldsymbol{\pi}$ is exactly equal to $\boldsymbol{\pi}$. Now, let us apply the herding update equations with the feature map $\boldsymbol{\phi}$ and input moment $\boldsymbol{\pi}$:

$$s_t = \arg\max_x \mathbf{w}_{t-1}^T \boldsymbol{\phi}(x) = \arg\max_x w_{x,t-1} \tag{4.33}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \boldsymbol{\pi} - \boldsymbol{\phi}(s_t) \tag{4.34}$$

The weight variables act similarly to the synaptic strength analogy in the neuron model example. At every iteration, the state with the highest potential gets activated, and then the corresponding weight is depressed after activation. Applying Proposition 4.2, we know that the empirical distribution of the samples converges to the input distribution at a faster

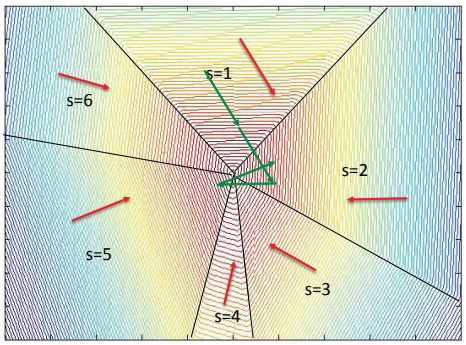


Figure 4.10: Cones in parameter space $\{w_1, w_2\}$ that correspond to the discrete states $s_1, ..., s_6$. Arrows indicate the translation vectors associated with the cones.

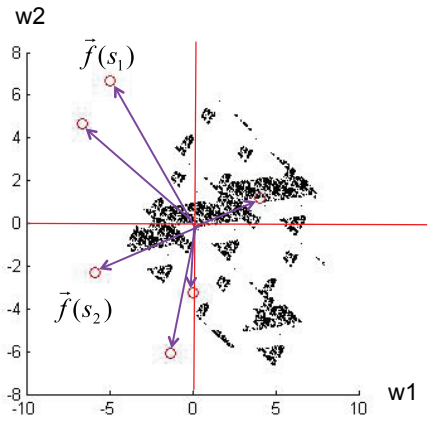


Figure 4.11: Fractal attractor set for herding with two parameters. The circles represent the feature-vectors evaluated at the states $s_1, ..., s_6$. Hausdorff dimension for this example is between 0 and 1.

rate than one would get from random sampling:

$$\left| \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\phi}(s_t) - \boldsymbol{\pi} \right| = \mathcal{O}\left(\frac{1}{T}\right) \tag{4.35}$$

The dynamics of the weight vector is more complex than the case of a binary variable in the previous subsection. However, there are still some interesting observations one can make about the trajectory of the weights which we explain in the appendix.

Weak Chaos in the Herding Dynamics

Now let us consider herding in a general setting with D states and each state is associated with a K dimensional feature vector. The update equation for the weights 4.8 can be viewed as a series of translations in the parameter space, $\mathbf{w} \rightarrow \mathbf{w} + \rho(\mathbf{x})$, where each discrete state $\mathbf{x} \in \mathcal{X}$ corresponds to one translation vector (i.e. $\rho(\mathbf{x}) = \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{x})$). See Figure 4.10 for an example with $D = 6$ and $K = 2$. The parameter space is partitioned into cones emanating from the origin, each corresponding to a state according to Equation 4.7. If the current location of the weights is inside cone $\mathbf{x}$, then one applies the translation corresponding to that cone and moves along $\rho(\mathbf{x})$ to the next point. This system is an example of what is known as a piecewise translation (or piecewise isometry more generally) (Goetz, 2000).

It is clear that this system has zero Lyapunov exponents[4] everywhere (except perhaps on the boundaries between cones but since this is a measure zero set we will ignore these). As the evolution of the weights will remain bounded inside some finite ball the evolution will converge to some attractor set. Moreover, the dynamics is non-periodic in the typical case (more formally, the translation vectors must form an incommensurate (possibly over-complete) basis set; for a proof see Appendix B of Welling and Chen (2010)). It can often be observed that this attractor has fractal dimension (see Figure 4.11 for an example). All these facts point to the idea that herding is on the edge between full chaos (with positive Lyapunov exponents) and regular periodic behavior (with negative Lyapunov exponents). In fact, herding is an example of what is called "weak chaos", which is usually defined through its (topological) entropy discussed below. Finally, as we have illustrated in Figure 4.1, one can construct a sequence of iterated maps of which herding is the limit and which exhibits period doubling. This is yet another characteristic of systems that are classified as "edge of chaos". Whether the attractor set is of fractal dimension in general remains an open question. For the case of single neuron model, the attractor is the entire interval $(\pi - 1, \pi]$ if π is irrational but for systems with more states it remains unknown.

We will now estimate the entropy production rate of herding. This will inform us further of the properties of this system and how it processes information. From Figure 4.10 we see that the sequence $s_1, s_2, ...$ can be interpreted as the symbolic system of the continuous dynamical system defined for the parameters $\mathbf{w}$. A sequence of symbols (states) is sometimes referred to as an "itinerary." Every time $\mathbf{w}$ falls inside a cone we record its label which equals the state $\mathbf{x}$. The topological entropy for the symbolic system can be defined by counting the total number of subsequences of length T, which we will call $M(T)$. One may think of this as a dynamical language where the subsequences are called "words" and the topological entropy is thus related to the number of words of length T. More precisely, the topological entropy is defined as,

$$h = \lim_{T\to\infty} h(T) = \lim_{T\to\infty} \frac{\log M(T)}{T} \tag{4.36}$$

4. The Lyapunov exponent of a dynamical system is a quantity that characterizes the rate of separation of infinitesimally close trajectories. Quantitatively, two trajectories in phase space with initial separation $|\delta Z(0)|$ diverge (provided that the divergence can be treated within the linearized approximation) at a rate given by $|\delta Z(t)| \approx e^{\lambda t}|\delta Z(0)|$ where λ is the Lyapunov exponent.

It was rigorously proven in Goetz (2000) that $M(T)$ grows polynomially in T for general piecewise isometries, which implies that the topological entropy vanishes for herding. It is however interesting to study the growth of $M(T)$ as a function of T to get a sense of how chaotic its dynamics is.

For the simplest model of a single neruon with π being an irrational number, it turns out $M(T) = T+1$, which is the absolute bare minimum for sequences that are not eventually periodic. It implies that our neuron model generates Sturmian sequences for irrational values of π which are precisely defined to be the non-eventually periodic sequences of minimal complexity (Lu and Wang, 2005). (For a proof, please see Welling and Chen (2010).)

To count the number of subsequences of length T for a general model, we can study the T-step herding map that results from applying herding T steps at a time. The original cones are now further subdivided into smaller convex polygons, each one labeled with the sequence $s_1, s_2, ..., s_T$ that the points inside the polygon will follow during the following T steps. Thus as we increase T, the number of these polygons will increase and it is exactly the number of those polygons which partition our parameter space that is equal to the number of possible subsequences. We first claim that every polygon, however small, will break up into smaller sub-pieces after a finite amount of time. This is proven in Welling and Chen (2010). In fact, we expect that in a typical herding system every pair of points will break up as well, which, if true, would infer that the diameter of the polygons must shrink. A partition with this property is called a *generating partition*. Based on some preliminary analysis and numerical simulations, we expect that the growth of $M(T)$ in the typical case (a.k.a. with an incommensurate translation basis, see Appendix B of Welling and Chen (2010)) is a polynomial function of the time, $M(T) \sim t^K$, where K is the number of dimensions (which is equal to the number of herding parameters). Since it has been rigorously proven that any piecewise isometry has a growth rate that must have an exponent less or equal than K (Goetz, 2000), this would mean that herding achieves the highest possible entropy within this class of systems with $H(T) = Th(T)$ for a sequence of length T (for T large enough) as:

$$H(T) = K \log(T) \tag{4.37}$$

This result should be understood in comparison with regular and random sequences. In a regular (constant or periodic) sequence, the number of subsequences is constant with respect to the length, i.e. $H(T) = \text{const}$. In contrast, the dominant part of the Kolmogorov-Sinai entropy of a random sequence (considering, e.g., a stochastic process) or a fully chaotic sequence

grows linearly in time T, i.e. $H_{\text{ext}}(T) = hT$ due to the injected random noise.

4.3 Generalized Herding

The moment matching property in Proposition 4.1 and 4.2 requires only a mild condition on the L_2 norm of the dynamic weights. That grants us with great flexibility in modifying the original algorithm for more practical implementation as well as a larger spectrum of applications. Gelfand et al. (2010) provided a general condition on the recurrence of the weight sequence, from which we discuss how to generalize the herding algorithm in this section with two specific examples. Chen et al. (2014) described another extension of herding that violated the condition but it achieved the minimum matching distance instead in a constrained problem.

4.3.1 A General Condition for Recurrence - The Perceptron Cycling Theorem

The moment matching property of herding relies on the recurrence of the weight sequence (Corollary 4.4) whose proof again relies on the premise that the maximization is carried out exactly in the herding update equation 4.7. However, the number of model states is usually exponentially large (e.g. $|\mathcal{X}| = J^m$ when $\mathbf{x}$ is a vector of m discrete variables each with J values) and it is intractable to find a global maximum in practice. A local maximizer has to be employed instead. One wonders if the features averaged over samples will still converge to the input moments when the samples are suboptimal states? In this subsection we give a general and verifiable condition for the recurrence of the weight sequence based on the perceptron cycling theorem (Minsky and Papert, 1969), which consequently suggests that the moment matching property may still hold at the rate of $\mathcal{O}(1/T)$ even with a relaxed herding algorithm.

The invention of the perceptron (Rosenblatt, 1958) goes back to the very beginning of AI more than half a century ago. Rosenblatt's very simple, neurally plausible learning rule made it an attractive algorithm for learning relations in data: for every input $\mathbf{x}_i$, make a linear prediction about its label: $y^*_{i_t} = \text{sign}(\mathbf{w}^T_{t-1}\mathbf{x}_{i_t})$ and update the weights as,

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{x}_{i_t}(y_{i_t} - y^*_{i_t}). \tag{4.38}$$

A critical evaluation by Minsky and Papert (1969) revealed the perceptron's limited representational power. This fact is reflected in the behavior of

Rosenblatt's learning rule: if the data is linearly separable, then the learning rule converges to the correct solution in a number of iterations that can be bounded by $(R/\gamma)^2$, where R represents the norm of the largest input vector and γ represents the margin between the decision boundary and the closest data-case. However, "for data sets that are not linearly separable, the perceptron learning algorithm will never converge" (quoted from Bishop et al. (2006)).

While the above result is true, the theorem in question has something much more powerful to say. The "perceptron cycling theorem" (PCT) (Minsky and Papert, 1969) states that for the inseparable case the weights remain bounded and do not diverge to infinity. The PCT was initially introduced in Minsky and Papert (1969) but had a gap in the proof that was fixed in Block and Levin (1970).

Theorem 4.7 (Boundedness Theorem)**.** *Consider a sequence of vectors* $\{\mathbf{w}_t\}$*,* $\mathbf{w}_t \in \mathbb{R}^D$*,* $t = 0, 1, \ldots$ *generated by the iterative procedure of Algorithm 4.1.*

Algorithm 4.1 Algorithm to generate the sequence $\{\mathbf{w}_t\}$.

V is a finite set of vectors in $\mathbb{R}^D$.
$\mathbf{w}_0$ is initialized arbitrarily in $\mathbb{R}^D$.
for $t = 0 \to T$ (T could be ∞) **do**
  $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_t$, where $\mathbf{v}_t \in V$ satisfies $\mathbf{w}_t^T \mathbf{v}_t \leq 0$
end for

Then, $\|\mathbf{w}_t\| \leq \|\mathbf{w}_0\| + M, \forall t \geq 0$ *where* M *is a constant depending on* V *but not on* $\mathbf{w}_0$*.*

The theorem still holds when V is a finite set in a Hilbert space. The PCT leads to the boundedness of the perceptron weights where we identify $\mathbf{v}_t = \mathbf{x}_{i_{t+1}}(y_{i_{t+1}} - y^*_{i_{t+1}})$, a finite set $V = \{\mathbf{x}_i(y_i - y_i^*) | y_i = \pm 1, y_i^* = \pm 1, i = 1, \ldots, N\}$ and observe

$$\mathbf{w}_t^T \mathbf{v}_t = \mathbf{w}_t^T \mathbf{x}_{i_{t+1}}(y_{i_{t+1}} - y^*_{i_{t+1}}) = |\mathbf{w}_t^T \mathbf{x}_{i_{t+1}}|(\text{sign}(\mathbf{w}_t^T \mathbf{x}_{i_{t+1}}) y_{i_{t+1}} - 1) \leq 0 \quad (4.39)$$

When the data is linearly separable, Rosenblatt's learning rule will find a $\mathbf{w}$ such that $\mathbf{w}^T \mathbf{v}_i = 0, \forall i$ and the sequence of $\mathbf{w}_t$ converges. Otherwise, there always exists some $\mathbf{v}_i$ such that $\mathbf{w}^T \mathbf{v}_i < 0$ and PCT guarantees the weights are bounded.

The same theorem also applies to the herding algorithm by identifying $\mathbf{v}_t = \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{s}_{t+1})$ with $\mathbf{s}_{t+1}$ defined in Equation 4.7, a finite set $V =$

$\{\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$, and observing that

$$\mathbf{w}_t^T \mathbf{v}_t = \mathbf{w}_t^T \bar{\boldsymbol{\phi}} - \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}_{t+1}) \leq 0 \tag{4.40}$$

It is now easy to see that, in general, herding does not converge because under very mild conditions we can always find an $\mathbf{s}_{t+1}$ such that $\mathbf{w}_t^T \mathbf{v}_t < 0$. More importantly, the boundedness theorem (or PCT) provides a general condition for the recurrence property and hence the moment matching property of herding. Inequality 4.40 is easy to be verified at running time and does not require $\mathbf{s}_{t+1}$ to be the global optimum.

4.3.2 Generalizing the Herding Algorithm

PCT ensures that the average features from the samples will match the moments at a fast convergence rate as long as the algorithm we are running satisfies the following conditions:

1. The set V is finite,
2. $\mathbf{w}_t^T \mathbf{v}_t = \mathbf{w}_t^T \bar{\boldsymbol{\phi}} - \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}_t) \leq 0, \forall t$,

This set of mild conditions allows us to generalize the original herding algorithm easily.

Firstly, the PCT provides a theoretical justification for using a local search algorithm that performs partial maximization. For example, we may start the local search from the state we ended up in during the previous iteration (a so-called persistent chain (Younes, 1989; Neal, 1992; Yuille, 2004; Tieleman, 2008)). Or, one may consider contrastive divergence-like algorithms (Hinton, 2002), in which the sampling or mean field approximation is replaced by a maximization. In this case, maximizations are initialized on all data-cases and the weights are updated by the difference between the average over the data-cases minus the average over the $\{\mathbf{s}_i\}$ found after (partial) maximization. In this case, the set V is given by: $V = \{\bar{\boldsymbol{\phi}} - \frac{1}{D}\sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{s}_i) | \mathbf{s}_i \in \mathcal{X}, \forall i\}$. For obvious reasons, it is now guaranteed that $\mathbf{w}_t^T \mathbf{v}_t \leq 0$.

Secondly, we often use mini-batches of size $d < D$ in practice instead of the full data set. In this case, the cardinality of the set V is enlarged to, e.g., $|V| = C(d, D)J^m$, with $C(d, D)$ representing the "d choose D" ways to compute the sample mean $\bar{\boldsymbol{\phi}}_{(d)}$ based on a subset of d data-cases. The negative term remains unaltered. Since the PCT still applies: $\left\| \frac{1}{\tau}\sum_{t=1}^{\tau} \bar{\boldsymbol{\phi}}_{(d),t} - \frac{1}{\tau}\sum_{t=1}^{\tau} \boldsymbol{\phi}(\mathbf{s}_t) \right\|_2 = \mathcal{O}(1/\tau)$. Depending on how the mini-batches are picked, convergence onto the overall mean $\bar{\boldsymbol{\phi}}$ can be either

$\mathcal{O}(1/\sqrt{\tau})$ (random sampling with replacement) or $\mathcal{O}(1/\tau)$ (sampling without replacement which has picked all data-cases after $\lceil D/d \rceil$ rounds).

Besides changing the way we compute the positive and negative terms in $\mathbf{v}_t$, generalizing the definition of *features* will allow us to learn a much wider scope of models beyond the fully visible MRFs as discussed in the following sections.

4.3.3 Herding Partially Observed Random Field Models

The original herding algorithm only works for fully visible MRFs because in order to compute the average feature vector of the training data we have to observe the state of all the variables in a model. In this subsection, we generalize herding to partially observed MRFs (POMRFs) by dynamically imputing the value of latent variables in the training data during the run of herding. This extension allows herding to be applied to models with a higher representative capacity.

Consider a MRF with discrete random variables $(\mathbf{x}, \mathbf{z})$ where $\mathbf{x}$ will be observed and $\mathbf{z}$ will remain hidden. A set of feature functions is defined on $\mathbf{x}$ and $\mathbf{z}$, $\{\phi_\alpha(\mathbf{x}, \mathbf{z})\}$, each associated with a weight w_α. Given these quantities we can write the following Gibbs distribution,

$$P(\mathbf{x}, \mathbf{z}; \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left(\sum_\alpha w_\alpha \phi_\alpha(\mathbf{x}, \mathbf{z})\right) \tag{4.41}$$

The log-likelihood function with a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^D$ is defined as

$$\ell(\mathbf{w}) = \frac{1}{D} \sum_{i=1}^{D} \log\left(\sum_{\mathbf{z}_i} \exp\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_i)\right)\right) - \log Z(\mathbf{w}) \tag{4.42}$$

Analogous to the duality relationship between MLE and MaxEnt for fully observed MRFs, we can write the log-likelihood of a POMRF as

$$\ell = \max_{\{Q_i\}} \min_R \frac{1}{D} \sum_{i=1}^{D} \mathcal{H}(Q_i) - \mathcal{H}(R) \tag{4.43}$$

$$+ \sum_\alpha w_\alpha \left(\frac{1}{D} \sum_{i=1}^{D} \mathbb{E}_{Q_i(\mathbf{z}_i)}[\phi_\alpha(\mathbf{x}_i, \mathbf{z}_i)] - \mathbb{E}_{R(\mathbf{x},\mathbf{z})}[\phi_\alpha(\mathbf{x}, \mathbf{z})]\right) \tag{4.44}$$

where $\{Q_i\}$ are variational distributions on $\mathbf{z}$, and R is a variational distribution on $(\mathbf{x}, \mathbf{z})$. The dual form of MLE turns out as a minimax problem on

$\frac{1}{D}\sum_{i=1}^{D}\mathcal{H}(Q_i) - \mathcal{H}(R)$ with a set of constraints

$$\frac{1}{D}\sum_{i=1}^{D}\mathbb{E}_{Q_i(\mathbf{z}_i)}[\phi_\alpha(\mathbf{x}_i,\mathbf{z}_i)] = \mathbb{E}_{R(\mathbf{x},\mathbf{z})}[\phi_\alpha(\mathbf{x},\mathbf{z})] \tag{4.45}$$

We want to achieve high entropy for the distributions $\{Q_i\}$ and R, and meanwhile the average feature vector on the training set with hidden variables marginalized out should match the expected feature w.r.t. to the joint distribution of the model. The weights $\mathbf{w}_\alpha$ act as Lagrange multipliers enforcing those constraints.

Similar to the derivation of herding for fully observed MRFs, we now introduce a temperature in Equation 4.42 by replacing $\mathbf{w}$ with $\mathbf{w}/T$. Taking the limit $T \to 0$ of $\ell_T \stackrel{\text{def}}{=} T\ell$, we see that the entropy terms vanish. For a given value of $\mathbf{w}$ and in the absence of entropy, the optimal distribution $\{Q_i\}$ and R are delta-peaks and their averages should be replace with maximizations, resulting in the objective,

$$\ell_0(\mathbf{w}) = \frac{1}{D}\sum_{i=1}^{D}\max_{\mathbf{z}_i}\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{z}_i) - \max_{\mathbf{s}}\mathbf{w}^T\boldsymbol{\phi}(\mathbf{s}) \tag{4.46}$$

where we denote $\mathbf{s} = (\mathbf{x},\mathbf{z})$.

Taking a gradient descent update on ℓ_0 with a fixed learning rate ($\eta = 1$) defines the herding algorithm on POMRFs (Welling, 2009b):

$$\mathbf{z}^*_{it} = \arg\max_{\mathbf{z}_i}\mathbf{w}^T_{t-1}\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{z}_i), \forall i \tag{4.47}$$

$$\mathbf{s}^*_t = \arg\max_{\mathbf{s}}\mathbf{w}^T_{t-1}\boldsymbol{\phi}(\mathbf{s}) \tag{4.48}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \left[\frac{1}{D}\sum_{i=1}^{D}\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{z}^*_{it})\right] - \boldsymbol{\phi}(\mathbf{s}^*_t) \tag{4.49}$$

We use a superscript "*" to denote states obtained by maximization. These equations are similar to herding for the fully observed case, but different in the sense that we need to impute the unobserved variables $\mathbf{z}_i$ for every data-case separately through maximization. The weight update also consist of a positive "driving term," which is now a changing average over data-cases, and a negative term, which is identical to the corresponding term in the fully observed case.

Moment Matching Property

We can prove the boundedness of the weights with PCT by identifying $\mathbf{v}_t = \left[\frac{1}{D}\sum_{i=1}^{D}\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{z}^*_{i,t+1})\right] - \boldsymbol{\phi}(\mathbf{s}^*_{t+1})$, a finite set $V = \{\mathbf{v}_t(\{\mathbf{z}_i\},\mathbf{s})|\mathbf{z}_i \in$

$\mathcal{X}_{\mathbf{z}}, \forall i, \mathbf{s} \in \mathcal{X}\}$, and observing the inequality

$$\mathbf{w}_t^T \mathbf{v}_t = \left[\frac{1}{D}\sum_{i=1}^{D} \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_{i,t+1}^*)\right] - \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}_{t+1}^*) \tag{4.50}$$

$$= \left[\frac{1}{D}\sum_{i=1}^{D} \max_{\mathbf{z}_i} \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_i)\right] - \max_{\mathbf{s}} \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}) \leq 0 \tag{4.51}$$

The last inequality holds because the second term maximizes over more variables than the first term. Again, we do not have to be able to solve the difficult optimization problems of Equation 4.47 and 4.48. Partial progress in the form of a few iterations of coordinate-wise descent is often enough to satisfy the condition in Equation 4.50 which can be checked easily.

Following a similar proof as Proposition 4.2, we obtain the fast moment matching property of herding on POMRFs:

Proposition 4.8. *There exists a constant R such that herding on a partially observed MRF satisfies*

$$\left|\frac{1}{\tau}\sum_{t=1}^{\tau}\frac{1}{D}\sum_{i=1}^{D}\phi_\alpha(\mathbf{x}_i, \mathbf{z}_{it}^*) - \frac{1}{\tau}\sum_{t=1}^{\tau}\phi_\alpha(\mathbf{s}_t^*)\right| \leq \frac{2R}{\tau}, \forall \alpha \tag{4.52}$$

Notice that besides a sequence of samples of the full state $\{\mathbf{s}_t^*\}$ that form the joint distribution in the herding algorithm, we also obtain a sequence of samples of the hidden variables $\{\mathbf{z}_{it}^*\}$ for every data case $\mathbf{x}_i$ that forms the conditional distribution of $P(\mathbf{z}_i|\mathbf{x}_i)$. Those consistencies in the limit of $\tau \to \infty$ in Proposition 4.8 are in direct analogy to the maximum likelihood problem of Equation 4.42 for which the following moment matching conditions hold at the MLE for all α,

$$\frac{1}{D}\sum_{i=1}^{D}\mathbb{E}_{P(\mathbf{z}_i|\mathbf{x}_i;\mathbf{w}_{\mathrm{MLE}})}[\phi_\alpha(\mathbf{x}_i, \mathbf{z}_i)] = \mathbb{E}_{P(\mathbf{x},\mathbf{z};\mathbf{w}_{\mathrm{MLE}})}[\phi_\alpha(\mathbf{x}, \mathbf{z})] \tag{4.53}$$

These consistency conditions alone are not sufficient to guarantee a good model. After all, the dynamics could simply ignore the hidden variables by keeping them constant and still satisfy the matching conditions. In this case the hidden and visible subspaces completely decouple, defeating the purpose of using hidden variables in the first place. Note that the same holds for the MLE consistency conditions alone. However, an MLE solution also strives for high entropy in the hidden states. We observe in practice that the herding dynamics usually also induces high entropy in the distributions for $\mathbf{z}$ avoiding the decoupling phenomenon described above.

The proof of the boundedness of weights depends on the assumption that we can find the global maximum in Equation 4.48, which is an intractable problem. Welling (2009b) also proposed a fully tractable herding variant that was guaranteed to satisfy PCT.

Proposition 4.9. *Call $\mathcal{A}$ any tractable optimization algorithm to locate a local maximum in the product $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x},\mathbf{z})$. This algorithm will be used to compute both $\mathbf{z}_i^*$ and $\mathbf{s}^*$. Call $\mathcal{E}_{\mathcal{A}}(\mathbf{x}_i,\mathbf{w}) = -\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{z}_i^*)$ the energy of data-case i (note that this definition depends on the algorithm $\mathcal{A}$). Assume that given any initialization, $\mathcal{A}$ always return a state with an energy no larger than its initial state. Then the following tractable herding algorithm will remain in a compact region of weight space: Apply the usual herding updates with the difference that the optimization for $\mathbf{s}^*$ is initialized at the state $(\mathbf{x}_{i^*},\mathbf{z}_{i^*}^*)$ which represents the data-case with lowest energy $\mathcal{E}_{\mathcal{A}}(\mathbf{x}_i,\mathbf{w})$.*

Proof. The proof is trivial using the PCT condition as:

$$\mathbf{w}_t^T\mathbf{v}_t = -\left[\frac{1}{D}\sum_{i=1}^{D}\mathcal{E}_{\mathcal{A}}(\mathbf{x}_i,\mathbf{w}_t)\right] + \mathcal{E}_{\mathcal{A}}(\mathbf{s}^*,\mathbf{w}_t) \tag{4.54}$$

$$\leq -\left[\frac{1}{D}\sum_{i=1}^{D}\mathcal{E}_{\mathcal{A}}(\mathbf{x}_i,\mathbf{w}_t)\right] + \mathcal{E}_{\mathcal{A}}(\mathbf{x}_{i^*},\mathbf{w}_t) \leq 0 \tag{4.55}$$

□

4.3.4 Herding Discriminative Models

We have been talking about running herding dynamics in an unsupervised learning setting. The idea of driving a nonlinear dynamical system to match moments can also be applied to discriminative learning by incorporating labels into the feature functions. Recalling the perceptron learning algorithm in Section 4.3.1, the learning rule in Equation 4.38 can be reformulated in herding style:

$$y_{i_t}^* = \underset{y\in\{-1,1\}}{\operatorname{argmax}}\, \mathbf{w}_{t-1}^T(\mathbf{x}_{i_t}y) \tag{4.56}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{x}_{i_t}y_{i_t} - \mathbf{x}_{i_t}y_{i_t}^* \tag{4.57}$$

where we identify the feature functions as $\phi_j(\mathbf{x},y) = x_j y, j = 1,\ldots,m$, use mini-batches of size 1 at every iteration, and do a partial maximization of the full state $(\mathbf{x},y)$ with the covariate $\mathbf{x}$ clamped at the input $\mathbf{x}_{i_t}$. The PCT guarantees that the moments (correlation between covariates and labels) $\mathbb{E}_{\mathcal{D}}[\mathbf{x}y]$ from the training data are matched with $\mathbb{E}_{\mathcal{D}_{\mathbf{x}}P(y^*|\mathbf{x})}[\mathbf{x}y^*]$ where $p(y^*|x)$ is the model distribution implied by how the learning process gen-

erates y^* with the sequence of weights $\mathbf{w}_t$. The voted perceptron algorithm (Freund and Schapire, 1999) is an algorithm that runs exactly the same update procedure, applies the weights to make a prediction on the test data at every iteration $y^*_{\text{test},t}$, and obtains the final prediction by averaging over iterations $y^*_{\text{test}} = \text{sign}(\frac{1}{\tau}\sum_{t=1}^{\tau} y^*_{\text{test},t})$. This amounts to learning and predicting based on the conditional expectation $\mathbb{E}_{P(y^*|\mathbf{x})}[y^* = 1|\mathbf{x}_{\text{test}}]$ in the language of herding.

Let us now formulate the *conditional herding* algorithm in a more general way (Gelfand et al., 2010). Denote the complete state of a data-case by $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ where $\mathbf{x}$ is the visible input variable, $\mathbf{y}$ is the label, and $\mathbf{z}$ is the hidden variable. Define a set of feature functions $\{\phi_\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z})\}$ with associated weights $\{w_\alpha\}$. Given a set of training data-cases, $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}$, and a test set $\mathcal{D}_{\text{test}} = \{\mathbf{x}_{\text{test},j}\}$, we run the conditional herding algorithm to learn the correlations between the inputs and the labels and make predictions at the same time using the following update equations:

$$\mathbf{z}'_{it} = \underset{\mathbf{z}_i}{\text{argmax}}\, \mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i), \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D} \tag{4.58}$$

$$(\mathbf{y}^*_{it}, \mathbf{z}^*_{it}) = \underset{(\mathbf{y}_i, \mathbf{z}_i)}{\text{argmax}}\, \mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i), \forall \mathbf{x}_i \in \mathcal{D}_{\mathbf{x}} \tag{4.59}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \left[\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}'_{it})\right] - \left[\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}^*_{it}, \mathbf{z}^*_{it})\right] \tag{4.60}$$

$$(\mathbf{y}^*_{\text{test},j,t}, \mathbf{z}^*_{\text{test},j,t}) = \arg\max_{(\mathbf{y}_j, \mathbf{z}_j)} \mathbf{w}_t^T \phi(\mathbf{x}_{\text{test},j}, \mathbf{y}_j, \mathbf{z}_j), \forall \mathbf{x}_{\text{test},j} \in \mathcal{D}_{\text{test}} \tag{4.61}$$

In the positive term of Equation 4.60, we maximize over the hidden variables only, and in the negative term we maximize over both hidden variables and the labels. The last equation generates a sequence of labels, $\mathbf{y}^*_{\text{test},j,t}$, that can be considered as samples from the conditional distribution of the test input from which we obtain an estimate of the underlying conditional distribution:

$$P(\mathbf{y}|\mathbf{x}_{\text{test},j}) \approx \frac{1}{\tau}\sum_{t=1}^{\tau} \mathbb{I}(\mathbf{y}^*_{\text{test},j,t} = \mathbf{y}) \tag{4.62}$$

In general, herding systems perform better when we use normalized features: $\|\boldsymbol{\phi}(\mathbf{x}, \mathbf{z}, \mathbf{y})\| = R,\ \forall (\mathbf{x}, \mathbf{z}, \mathbf{y})$. The reason is that herding selects states by maximizing the inner product $\mathbf{w}^T\boldsymbol{\phi}$ and features with large norms will therefore become more likely to be selected. In fact, one can show that states inside the convex hull of the $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ are never selected. For binary (± 1) variables all states live on the convex hull, but this need not be true in general, especially when we use continuous attributes $\mathbf{x}$. To rem-

edy this, one can either normalize features or add one additional feature[5] $\boldsymbol{\phi}_0(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sqrt{R_{\max}^2 - ||\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z})||^2}$, where $R_{\max} = \max_{\mathbf{x},\mathbf{y},\mathbf{z}} \|\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z})\|$ with $\mathbf{x}$ only allowed to vary over the data-cases.

We may want to use mini-batches $\mathcal{D}_t$ instead of the whole training set for a more practical implementation, and the argument on the validity of using mini-batches in Section 4.3.2 applies here as well. It is easy to observe that Rosenblatts's perceptron learning algorithm is a special case of conditional herding when there are no hidden variables, $\mathbf{y}$ is a single binary variable, the feature function is $\boldsymbol{\phi} = \mathbf{x}y$, and we use a mini-batch of size 1 at every iteration.

Compared to the herding algorithm on partially observed MRFs, the main difference is that we do partial maximization in Equation 4.59 with a clamped visible input $\mathbf{x}$ on every training data-case instead of a joint maximization on the full state. Notice that in this particular variant of herding, the sequence of updates may converge when all the training data-cases are correctly predicted, that is, $\mathbf{y}_{it}^* = \mathbf{y}_i, \forall i = 1, \ldots, D$ at some t. For an example, the convergence is guaranteed to happen for the percepton learning algorithm on a linearly separable data set. We adopt the strategy in the voted perceptron algorithm (Freund and Schapire, 1999) which stops herding when convergence occurs and uses the sequence of weights up to that point for prediction in order to prevent the converged weights from dominating the averaged prediction on the test data.

Clamping the input variables allows us to achieve the following moment matching property:

Proposition 4.10. *There exists a constant R such that conditional herding with the update equations 4.58-4.60 satisfies*

$$\left| \frac{1}{D} \sum_{i=1}^{D} \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{x}_i, \mathbf{y}_{it}^*, \mathbf{z}_{it}^*) - \frac{1}{D} \sum_{i=1}^{D} \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_{it}') \right| \leq \frac{2R}{\tau}, \forall \alpha \quad (4.63)$$

The proof is straightforward by applying PCT where we identify

$$\mathbf{v}_t = \left[\frac{1}{D} \sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_{it}') \right] - \left[\frac{1}{D} \sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_{it}^*, \mathbf{z}_{it}^*) \right], \quad (4.64)$$

the finite set $V = \{\mathbf{v}(\{\mathbf{z}_i'\}, \{\mathbf{y}_i^*\}, \{\mathbf{z}_i^*\}) | \mathbf{z}_i' \in \mathcal{X}_\mathbf{z}, \mathbf{y}_i^* \in \mathcal{X}_\mathbf{y}, \mathbf{z}_i^* \in \mathcal{X}_\mathbf{z}\}$, and observe the inequality $\mathbf{w}_t^T \mathbf{v}_t \leq 0$ because of the same reason as herding on POMRFs. Note that we require V to be of a finite cardinality, which in return requires $\mathcal{X}_\mathbf{y}$ and $\mathcal{X}_\mathbf{z}$ to be finite sets, but there is not any restriction on

5. If in test data this extra feature becomes imaginary we simply set it to zero.

the domain of the visible input variables $\mathbf{x}$. Therefore we can run conditional herding with input $\mathbf{x}$ as continuous variables.

Zero Temperature Limit of CRF

Consider a CRF with the probability distribution defined as

$$P(\mathbf{y}, \mathbf{z}|\mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp\left(\sum_{\alpha} w_\alpha \phi_\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z})\right) \tag{4.65}$$

where $Z(\mathbf{w}, \mathbf{x})$ is the partition function of the conditional distribution. The log-likelihood function for a dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^D$ is expressed as

$$\ell(\mathbf{w}) = \frac{1}{D} \sum_{i=1}^{D} \left(\log\left(\sum_{\mathbf{z}_i} \exp\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)\right)\right) - \log Z(\mathbf{w}, \mathbf{x}_i)\right) \tag{4.66}$$

Let us introduce the temperature T by replacing $\mathbf{w}$ with $\mathbf{w}/T$ and take the limit $T \to 0$ of $\ell_T \stackrel{\text{def}}{=} T\ell$. We then obtain the familiar piecewise linear Tipi function

$$\ell_0(\mathbf{w}) = \frac{1}{D} \sum_{i=1}^{D} \left(\max_{\mathbf{z}_i} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) - \max_{\mathbf{y}_i, \mathbf{z}_i} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)\right) \tag{4.67}$$

Running gradient descent updates on $\ell_0(\mathbf{w})$ immediately gives us the update equations of conditional herding 4.58-4.60.

Similar to the duality relationship between MLE on MRFs and the MaxEnt problem, MLE on CRFs is the dual problem of maximizing the entropy of the conditional distributions while enforcing the following constraints:

$$\frac{1}{D} \sum_{i=1}^{D} \mathbb{E}_{P(\mathbf{z}|\mathbf{x}_i, \mathbf{y}_i)} \left[\phi_\alpha(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z})\right] = \frac{1}{D} \sum_{i=1}^{D} \mathbb{E}_{P(\mathbf{y}, \mathbf{z}|\mathbf{x}_i)} \left[\phi_\alpha(\mathbf{x}_i, \mathbf{y}, \mathbf{z})\right], \forall \alpha \tag{4.68}$$

When we run conditional herding, those constraints are satisfied with the moment matching property in Proposition 4.10, but how to encourage high entropy during the herding dynamics is again an open problem. We suggest some heuristics to achieve high entropy in the next experimental section. Note that there is a difference between MLE and conditional herding when making predictions. While the prediction of a CRF with MLE is made with the most probable label value at a point estimate of the parameters, conditional herding resorts to a majority voting strategy as in the voted perceptron algorithm. The regularization effect via averaging over predictions often provides more robust performance as shown later.

4.4 Experiments

We study the empirical performance of the herding algorithm introduced in Section 4.2 and the extension with hidden variables in Section 4.3.3 and for discriminative models in Section 4.3.4.

4.4.1 Herding with Fully Visible Models

In the following experiments we will determine the ability of herding to convert information about the average value of features in the training data into estimates of some quantities of interest. In particular the input to herding will be joint probabilities of pairs of variables (denoted H.XX) and sometimes triples of variables (denoted H.XXX) where all variables will be binary valued (which is easily relaxed).

In experiment I we will consider the quantity $P(k) = \mathbb{E}[\mathbb{I}[\sum_i X_i = k-1]]$ which is the distribution of the total number of 1's across all attributes. This quantity involves all variables in the problem and cannot be directly estimated from the input which consists of pairwise information only. This experiment measures the ability of herding to generalize from local information to global quantities of interest. In total 100K samples were generated and used to estimate $P(k)$. The results were compared with the following two alternatives: 1) sampling 100K pseudo-samples from the single variable marginals and using them to estimate $P(k)$ (denoted "MARG"), 2) learning a fully connected, fully visible Boltzmann machine using the pseudo-likelihood method[6] (denoted PL), then sampling 200K samples from that model and using the last 100K to estimate $P(k)$.

In experiment II we will estimate a discriminant function for classifying one attribute (the label) given the values of other attributes. Our approach was simply to perform online learning of a logistic regression function after each pseudo-sample collected from herding. Again, local pairwise information is turned into a global discriminant function which is then compared with some standard classifiers learned directly from the data. In particular, we compared against Naive Bayes, 5-nearest neighbors, logistic regression and a fully observed, fully connected Boltzmann machine learned with pseudo likelihood on the joint space of attributes and labels. The learned model's conditional distribution of label given the remaining attributes was subsequently used for prediction.

We have used the following datasets in our experiments.

6. This method is close to optimal for this type of problem (Parise and Welling, 2005).

Dataset	H.XXX	H.XX	PL	MARG
Bowling	5E-3	4.1E-2	1.2E-1	4.3E-1
Abelone	8E-4	2.5E-3	2.2E-2	1.8E0
Digits	-	6.2E-2	3.3E-2	4E-1
News	-	2.5E-2	1.9E-2	5E-1

Table 4.1: **Abelone/Digits/NewsGroups**: KL divergence between true (data) distribution and the estimates from 1) herding algorithm using all triplets, 2) herding with all pairs, 3) samples from pseudo-likelihood model and 4) samples from single marginals.

A) The "Bowling Data" set[7]. Each binary attribute represents whether a pin has fallen during two subsequent bowls. There are 10 pins and 298 games in total. This data was generated by P. Cotton to make a point about the modelling of company default dependency. Random splits of 150 train and 148 test instances were used for the classification experiments.

B) Abalone dataset[8]. We converted the dataset into binary values by subtracting the mean from all (8) attributes and labels and setting all obtained values to 0 if smaller than 0 and 1 otherwise. For the classification task we used random subsets of 2000 examples for training and the remaining 2177 for testing.

C) "Newsgroups-small"[9] prepared by S. Roweis. It has 100 binary attributes and $16,242$ instances and is highly sparse (4% of the values is 1). Random splits of $10,000$ train and $6,242$ test instances were used for the classification experiments.

D) Digits: 8×8 binarized handwritten digits. We used 1100 examples from the digit classes 3 and 5 respectively (a total of 2200 instances). The dataset contains 30% 1's. This dataset was split randomly in 1600 train and 600 test instances.

The results for experiment I are shown in Table 4.1 and Figure 4.12. Note that the herding algorithms are deterministic and repetition would have resulted in the same values.

We observe that herding is successful in turning local average statistics into estimates of global quantities. Providing more information such as joint probabilities over triplets does significantly improve the result (the triplet results for Digits and News took too long to run due to the large number

7. Downloadable from: `http://www.financialmathematics.com/wiki/Code:tenpin/data`
8. Downloadable from UCI repository
9. Downloaded from: `http://www.cs.toronto.edu/~roweis/data.html`

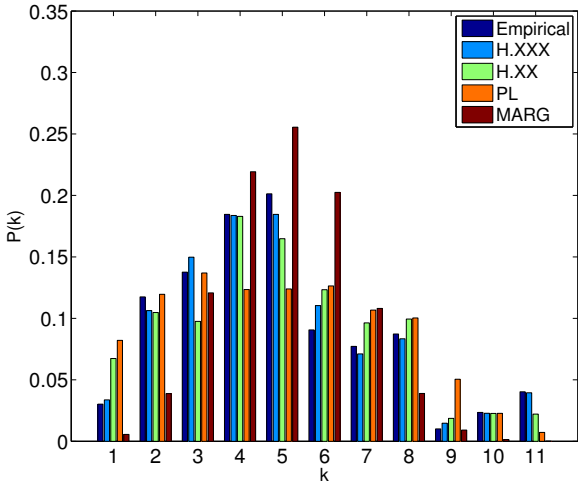


Figure 4.12: Estimates of $P(k)$ for the Bowling dataset. Each group of 5 bars represent the estimates for 1) ground truth, 2) herding with triples, 3) herding with pairs, 4) pseudo-likelihood, 5) marginals.

Dataset	H.XXY	PL	5NN	NB	LR
Abelone	0.24 ± 0.004	0.24 ± 0.004	0.33 ± 0.1	0.27 ± 0.006	0.24 ± 0.004
Bowling	0.23 ± 0.03	0.28 ± 0.06	0.32 ± 0.05	0.23 ± 0.03	0.23 ± 0.03
Digits	0.05 ± 0.01	0.06 ± 0.01	0.05 ± 0.01	0.09 ± 0.01	0.06 ± 0.02
News	0.11 ± 0.005	0.04 ± 0.001	0.13 ± 0.006	0.12 ± 0.003	0.11 ± 0.004

Table 4.2: Average classification results averaged over 5 runs.

of triplets involved). Also of interest is the fact that for the low dimensional data H.XX outperformed PL but for the high-D datasets the opposite was true while both methods seem to leverage the same second order statistics (even though PL needs the actual data to learn its model).

The results for the classification experiment are shown in Table 4.2. On all tasks the online learning of a linear logistic regression classifier did just as well as running logistic regression on the original data directly. This implies that the herding algorithm generates the information necessary for classification and that the decision boundary can be learned online during herding. Interestingly, the PL procedure significantly outperformed all standard classifiers as well as herding on the Newsgroup data. This implies that a more sophisticated decision boundary is warranted for this data.

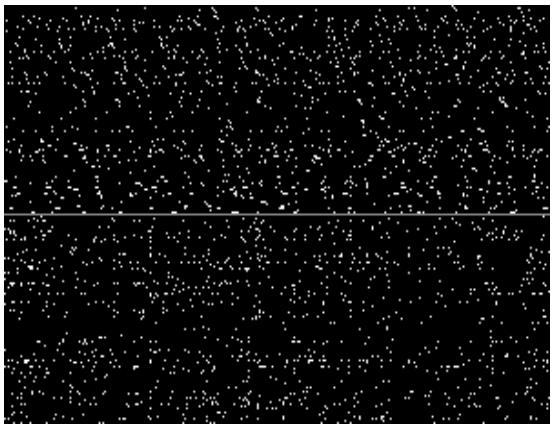


Figure 4.13: Top half: Sequence of 300 pseudo-samples generated from a herding algorithm for the "Newsgroup" dataset. White dots indicate the presence of certain word-types in documents (represented as columns). Bottom half: Newsgroup data (in random order). Data and pseudo-samples have the same first and second order statistics.

To see if the herding sequence contained the information necessary to estimate such a decision boundary we reran PL on the first 10,000 pseudo-samples generated by herding resulting in an error of 0.04, answering the question in the affirmative. A plot of the herding pseudo-samples as compared to the original data is shown in Figure 1.

4.4.2 Herding with Hidden Variables

We studied generalized herding on the architecture of a restricted Boltzmann machine (Hinton, 2002) (RBM). We used features $\boldsymbol{\phi}(x, z) = \{x_j, z_k, x_j z_k\}$, where j and k are indices of variables, and the $\{-1, +1\}$ representation because we found it worked significantly better than the $\{0, 1\}$ representation. To increase the entropy of the hidden units we left out the growth update for the features $\{z_k\}$ implying that $p(z_k = 1) \approx 0.5$. The intuition is the same as for bagging: we want to create a high diversity of (almost independent) ways to reconstruct the data because it will reduce the variance when making predictions. We observed that high entropy hidden representations automatically emerged when using a large number of hidden units. In contrast, for a small number of hidden units (say $K < 30$) there is a tendency for the system to converge on low entropy representations and the trick delivers some improvement.

We applied herding to the USPS Handwritten Digits dataset[10] which consists of 1100 examples of each digit 0 through 9 (totaling 11,000 examples). Each image has 256 pixels and each pixel has a value between [1..256] which we turned into a binary representation through the mapping $x'_j = 2\Theta(\frac{x_j}{256} - 0.2) - 1$ with $\Theta(x > 0) = 1$ and 0 otherwise. Each digit class was randomly split into 700 train, 300 validation and 100 test examples. As benchmarks we used 1NN using Manhattan distance and multinomial logistic regression, both in pixel space.

We used two versions of herding, one where the maximization over $\mathbf{s}$ was initialized at the value from the previous time step (H) and one where we initialize at the data-case with the lowest energy (SH - the tractable algorithm). In both cases we ran herding for 2000 iterations for each class individually. During the second 1000 iterations we computed the energies for the training data in that class, as well as for all validation and test data across all classes. At each iteration we then used the training energies to standardize the validation and test energies by computing their Z-scores: $\mathcal{E}'_i = (\mathcal{E}_i - \mu_{\text{trn}})/\sigma_{\text{trn}}$ where μ_{trn} and σ_{trn} represent the mean and standard deviation of the energies of the training data at that iteration. The standardized energies for test and validation data were subsequently averaged over herding iterations (using online averaging). Once we have collected these average standardized energies across all digit classes we fit a multinomial logistic regression classifier to the validation data, using the 10 class-specific energies as features.

We also compared these results against models learned with contrastive divergence (Hinton, 2002) (CD) and persistent CD (Tieleman, 2008) (PCD). For both CD and PCD we first applied (P)CD learning for 1000 iterations in batch mode, using a stepsize of $\eta = 10^{-3}$. A momentum parameter of 0.9 and 1-step reconstructions were used for CD. No momentum and a single sample in the negative phase was used for PCD. In the second 1000 iterations we continued learning but also collected standardized validation and test energies as before which we subsequently used for classification. We have also experimented with chains of length 10 and found that it did not improved the results but became prohibitively inefficient. To improve efficiency we experimented with learning in mini-batches but this degraded the results significantly, presumably because the number of training examples used to standardize the energy scores became less reliable.

The results reported in Figure 4.14 show the classification results averaged across 4 runs with different splits and for different values of hidden units.

10. Downloaded from `http://www.cs.toronto.edu/~roweis/data.html`

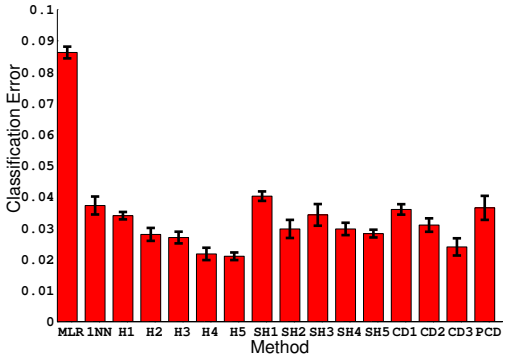


Figure 4.14: Classification results on USPS digits. 700 digits per class were used for training, 300 for validation and 100 for testing. Shown are average results over 4 different splits and their standard errors. From left to right: MLR (multinomial logistic regression), 1NN (1-nearest neighbor), H1-H5 (herding using local optimization with 50,100,250,500 and 1000 hidden units respectively), SH1-SH5 (safe, tractable herding from section 7 with 50,100,250,500 and 1000 hidden units respectively), CD1-CD3 (contrastive divergence with 50,100,250 hidden units respectively) and PCD (persistent CD with 500 hidden units).

Without trying to claim superior performance we merely want to make the case that herding can be leveraged to achieve state-of-the-art performance (note that USPS error rates are higher than MNIST error rates). We also see that the tractable version of herding did not perform as well as the herding using local optimization, which in turn performed equally well as learning a model using CD. Persistent CD did not give very good results presumably because we did not use optimal settings for step-size, weight-decay etc.. It is finally interesting to observe that there does not seem to be any sign of over-fitting for herding. For the model with 1000 hidden units, the total number of real parameters involved is around 1.5 million which represents more capacity than the 1.5 million binary pixel values in the data.

4.4.3 Discriminative Herding

We studied the behavior of conditional herding on two artificial and four real-world data sets, comparing its performance to that of the voted perceptron (Freund and Schapire, 1999) and that of discriminative RBMs (Larochelle and Bengio, 2008). All the experiment results in this subsection are accredited to the authors of Gelfand et al. (2010).

We studied conditional herding in the discriminative RBM (dRBM) architecture illustrated in Figure 4.15, that is, we use the following parame-

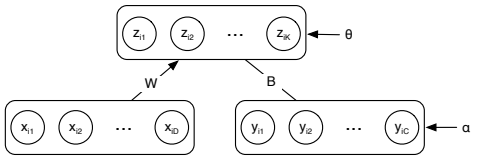


Figure 4.15: Discriminative Restricted Boltzmann Machine model of distribution $p(\mathbf{y}, \mathbf{z}|\mathbf{x})$.

terization

$$\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x}^T \mathbf{W} \mathbf{z} + \mathbf{y}^T \mathbf{B} \mathbf{z} + \boldsymbol{\theta}^T \mathbf{z} + \boldsymbol{\alpha}^T \mathbf{y}. \tag{4.69}$$

where $\mathbf{W}$, $\mathbf{B}$, $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ are the weights, $\mathbf{z}$ is a binary vector and $\mathbf{y}$ is a binary vector in a 1-of-K scheme.

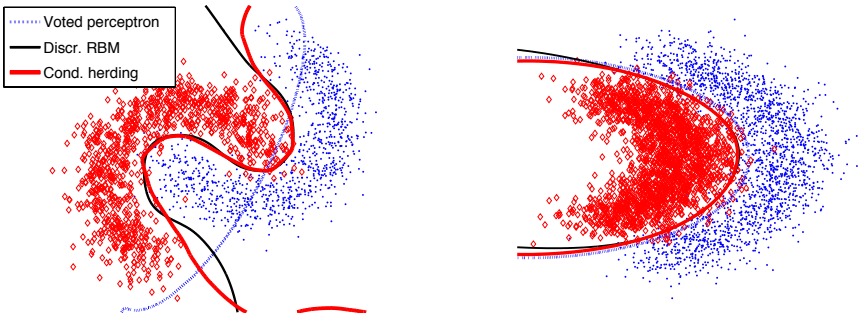
Per the discussion in Section 4.3.4, we added an additional feature $\boldsymbol{\phi}_0(\mathbf{x}) = \sqrt{R_{\max}^2 - ||\mathbf{x}||^2}$ with $R_{\max} = \max_i \|\mathbf{x}_i\|$ in all experiments.

Experiments on Artificial Data

To investigate the characteristics of the voted perceptron (VP), discriminative RBM (dRBM) and conditional herding (CH), we used the techniques discussed in Section 4.3.4 to construct decision boundaries on two artificial data sets: (1) the banana data set; and (2) the Lithuanian data set. We ran VP and CH for $1,000$ epochs using mini-batches of size 100. The decision boundary for VP and CH is located at the location where the sign of the prediction $\mathbf{y}^*_{\text{test}}$ changes. We used conditional herders with 20 hidden units. The dRBMs also had 20 hidden units and were trained by running conjugate gradients until convergence. The weights of the dRBMs were initialized by sampling from a Gaussian distribution with a variance of 10^{-4}. The decision boundary for the dRBMs is located at the point where both class posteriors are equal, i.e., where $p(y^*_{\text{test}} = -1|\tilde{\mathbf{x}}_{\text{test}}) = p(y^*_{\text{test}} = +1|\tilde{\mathbf{x}}_{\text{test}}) = 0.5$.

Plots of the decision boundary for the artificial data sets are shown in Figure 4.16. The results on the banana data set illustrate the representational advantages of hidden units. Since VP selects data points at random to update the weights, on the banana data set, the weight vector of VP tends to oscillate back and forth yielding a nearly linear decision boundary[11]. This happens because VP can regress on only $2+1 = 3$ fixed features. In contrast,

11. On the Lithuanian data set, VP constructs a good boundary by exploiting the added 'normalizing' feature.



(a) Banana data set. (b) Lithuanian data set.

Figure 4.16: Decision boundaries of VP, CH, and dRBMs on two artificial data sets.

for CH the simple predictor in the top layer can regress onto $M = 20$ hidden features. This prevents the same oscillatory behavior from occurring.

Experiments on Real-World Data

In addition to the experiments on synthetic data, we also performed experiments on four real-world data sets - namely, (1) the USPS data set, (2) the MNIST data set, (3) the UCI Pendigits data set, and (4) the 20-Newsgroups data set. The USPS data set consists of 11,000, 16×16 grayscale images of handwritten digits ($1,100$ images of each digit 0 through 9) with no fixed division. The MNIST data set contains $70,000$, 28×28 grayscale images of digits, with a fixed division into $60,000$ training and $10,000$ test instances. The UCI Pendigits consists of 16 (integer-valued) features extracted from the movement of a stylus. It contains $10,992$ instances, with a fixed division into $7,494$ training and $3,498$ test instances. The 20-Newsgroups data set contains bag-of-words representations of $18,774$ documents gathered from 20 different newsgroups. Since the bag-of-words representation comprises of over $60,000$ words, we identified the $5,000$ most frequently occurring words. From this set, we created a data set of $4,900$ binary word-presence features by binarizing the word counts and removing the 100 most frequently occurring words. The 20-Newsgroups data has a fixed division into $11,269$ training and $7,505$ test instances. On all data sets with real-valued input attributes we used the 'normalizing' feature described above.

The data sets used in the experiments are multi-class. We adopted a 1-of-K encoding, where if $\mathbf{y}_i$ is the label for data point $\mathbf{x}_i$, then $\mathbf{y}_i = \{y_{i,1}, ..., y_{i,K}\}$ is a binary vector such that $y_{i,k} = 1$ if the label of the i^{th} data point is k

and $y_{i,k} = -1$ otherwise. Performing the maximization in Equation 4.59 is difficult when $K > 2$. We investigated two different procedures for doing so. In the first procedure, we reduce the multi-class problem to a series of binary decision problems using a one-versus-all scheme. The prediction on a test point is taken as the label with the largest online average. In the second procedure, we make predictions on all K labels jointly. To perform the maximization in Equation 4.59, we explore all states of $\mathbf{y}$ in a one-of-K encoding - i.e. one unit is activated and all others are inactive. This partial maximization is not a problem as long as the ensuing configuration satisfies $\mathbf{w}_t^T \mathbf{v}_t \leq 0$ [12]. The main difference between the two procedures is that in the second procedure the weights $\mathbf{W}$ are shared amongst the K classifiers. The primary advantage of the latter procedure is its less computationally demanding than the one-versus-all scheme.

We trained the dRBMs by performing iterations of conjugate gradients (using 3 line searches) on mini-batches of size 100 until the error on a small held-out validation set started increasing (i.e., we employed early stopping) or until the negative conditional log-likelihood on the training data stopped coming down. Following Larochelle and Bengio (2008), we use L_2-regularization on the weights of the dRBMs; the regularization parameter was determined based on the generalization error on the same held-out validation set. The weights of the dRBMs were initialized from a Gaussian distribution with variance of 10^{-4}.

CH used mini-batches of size 100. For the USPS and Pendigits data sets CH used a burn-in period of $1,000$ updates; on MNIST it was $5,000$ updates; and on 20 Newsgroups it was $20,000$ updates. Herding was stopped when the error on the training set became zero [13].

The parameters of the conditional herders were initialized by sampling from a Gaussian distribution. Ideally, we would like each of the terms in the energy function in Equation 4.69 to contribute equally during updating. However, since the dimension of the data is typically much greater than the number of classes, the dynamics of the conditional herding system will be largely driven by $\mathbf{W}$. To negate this effect, we rescaled the standard deviation of the Gaussian by a factor $1/M$ with M the total number of elements of the parameter involved (e.g. $\sigma_{\mathbf{W}} = \sigma/(\dim(\mathbf{x})\dim(\mathbf{z}))$ etc.). We also scale the learning rates $\boldsymbol{\eta}$ by the same factor so the updates will retain this scale

12. Local maxima can also be found by iterating over $y^{*,k}_{\text{test}}, z^{*,k}_{\text{test},j}$, but the proposed procedure is more efficient.

13. We use a fixed order of the mini-batches, so that if there are D data cases and the batch size is d, if the training error is 0 for $\lceil D/d \rceil$ iterations, the error for the whole training set is 0.

during herding. The relative scale between $\boldsymbol{\eta}$ and σ was chosen by cross-validation. Recall that the absolute scale is unimportant (see Section 4.3.4 for details).

In addition, during the early stages of herding, we adapted the parameter update for the bias on the hidden units $\boldsymbol{\theta}$ in such a way that the marginal distribution over the hidden units was nearly uniform. This has the advantage that it encourages high entropy in the hidden units, leading to more useful dynamics of the system. In practice, we update $\boldsymbol{\theta}$ as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\boldsymbol{\eta}}{D_t} \sum_{i_t} (1 - \lambda) \langle \mathbf{z}_{i_t} \rangle - \mathbf{z}^*_{i_t}$, where i_t indexes the data points in the mini-batch at time t, D_t is the size of the mini-batch, and $\langle \mathbf{z}_{i_t} \rangle$ is the batch mean. λ is initialized to 1 and we gradually half its value every 500 updates, slowly moving from an entropy-encouraging update to the standard update for the biases of the hidden units.

VP was also run on mini-batches of size 100 (with a learning rate of 1). VP was run until the predictor started overfitting on a validation set. No burn-in was considered for VP.

The results of our experiments are shown in Table 4.3. In the table, the best performance on each data set using each procedure is typeset in boldface. The results reveal that the addition of hidden units to the voted perceptron leads to significant improvements in terms of generalization error. Furthermore, the results of our experiments indicate that conditional herding performs on par with discriminative RBMs on the MNIST and USPS data sets and better on the 20 Newsgroups data set. The 20 Newsgroups data is high dimensional and sparse and both VP and CH appear to perform quite well in this regime. Techniques to promote sparsity in the hidden layer when training dRBMs exist (see Larochelle and Bengio (2008)), but we did not investigate them here. It is also worth noting that CH is rather resilient to overfitting. This is particularly evident in the low-dimensional UCI Pendigits data set, where the dRBMs start to badly overfit with 500 hidden units, while the test error for CH remains level. This phenomenon is the benefit of averaging over many different predictors.

4.5 Summary

We introduce the herding algorithm in this chapter as an alternative to the maximum likelihood estimation for Markov random fields. It skips the parameter estimation step and directly converts a set of moments from the training data into a sequence of model parameters accompanied by a sequence of pseudo-samples. By integrating the intractable training and

One-Versus-All Procedure

Data Set	**VP**	**Discriminative RBM**		**Conditional herding**	
		100	200	100	200
MNIST	7.69%	**3.57%**	3.58%	3.97%	3.99%
USPS	5.03% (0.4%)	3.97% (0.38%)	4.02% (0.68%)	3.49% (0.45%)	**3.35%** (0.48%)
UCI Pendigits	10.92%	5.32%	5.00%	3.37%	**3.00%**
20 Newsgroups	27.75%	34.78%	34.36%	29.78%	**25.96%**

Joint Procedure

Data Set	**VP**	**Discriminative RBM**			**Conditional herding**		
		50	100	500	50	100	500
MNIST	8.84%	3.88%	2.93%	**1.98%**	2.89%	2.09%	2.09%
USPS	4.86% (0.52%)	3.13% (0.73%)	2.84% (0.59%)	4.06% (1.09%)	3.36% (0.48%)	3.07% (0.52%)	**2.81%** (0.50%)
UCI Pendigits	6.78%	3.80%	3.23%	8.89%	3.14%	**2.57%**	2.86%
20 Newsgroups	**24.89%**	–	30.57%	30.07%	–	25.76%	24.93%

Table 4.3: Generalization errors of VP, dRBMs, and CH on 4 real-world data sets. dRBMs and CH results are shown for various numbers of hidden units. The best performance on each data set is typeset in boldface; missing values are shown as '-'. The std. dev. of the error on the 10-fold cross validation of the USPS data set is reported in parentheses.

testing steps in the regular machine learning paradigm, herding provides a more efficient way of learning and predicting in MRFs.

We study the statistical properties of herding and show that herding dynamics introduces negative auto-correlation in the sample sequence which helps to speed up the mixing rate of the sampler in the state space. Quantitatively, the negative auto-correlation leads to a fast convergence rate of $\mathcal{O}(1/T)$ between the sampling statistics and the input moments. That is significantly faster than the rate of $\mathcal{O}(1/\sqrt{T})$ that an ideal random sampler would obtain for an MRF at MLE. This distinctive property of herding should also be attributed to its weak-chaotic behavior as a deterministic dynamic system, whose characteristics deserve its own interest for future research.

Experiments confirms that the information contained in the pseudo-samples of herding can be used for inference and prediction. It achieves comparable performance with traditional machine learning algorithms including the MRFs, even though the sampling distribution of herding does not guarantee the maximum entropy.

We further provide a general condition, PCT, for the fast moment matching property. That condition allows more practical implementations of herding. We also use it to derive extensions of the herding algorithm for a wider range of applications. As more flexible feature functions defined on both visible and latent variables can now be handled in the generalized algorithm, we apply herding to training partially observed MRFs. Experiments on the USPS dataset show a classification accuracy on par with the state-of-art training algorithms on the same model. Furthermore, we propose a discriminative learning variant of herding for supervised problems by including labelling information in the feature definition. The resulting conditional herding provides an alternative to training CRFs. Empirical evaluation shows competitive performance of herding compared with standard algorithms.

4.6 Conclusion

The view espoused in this chapter is that we can view learning as an iterated map: $\mathbf{w}_{t+1} = F(\mathbf{w}_t)$ and that we can study the properties of this map using the tools of nonlinear dynamics systems. The usual learning approaches based on point estimates form a contractive map where all of parameter space is eventually mapped to a point. In Bayesian approaches we seek to find a posterior distribution over parameters and the map should thus converge to a distribution (or measure). For MCMC for instance the map consists of convolving the current distribution with a kernel. Herding offers a third possibility where the attractor is neither a point, nor a measure in the usual sense, but rather a highly complex, possibly fractal set. Interestingly, the more recent approach "perturb and map" is related to herding in the sense that it consists of a sequence of perturbations of the parameters followed by an optimization over the state space. However, it is different from herding in the sense the perturbations are generated randomly and IID, while in herding the perturbations are deterministic and dynamic (i.e. depend on the previous parameters).

The surprising and powerful insight is that we can use a new set of tools from the mathematics literature to study these maps. For instance, it was shown in this chapter that herding dynamics is a special instance of the class of piecewise isometry maps, and should neither be classified as regular nor chaotic, but rather as what is known as "edge of chaos". We suspect that this type of dynamics has useful properties in the context of learning from data. For instance, it seems related to the fact that the certain empirical moments averages exhibit very fast convergence. This is supported by the observations that 1) piecewise isometries have vanishing topological entropy,

2) exhibit the "period doubling route to chaos" and 3) have vanishing Lyapunov exponents. We believe that these type of concepts from the field of nonlinear dynamical systems may one day play an important role in the field of machine learning.

Appendix:

Some Results on Herding in Discrete Spaces

The following proposition shows that the weight vectors move inside a $D-1$ dimensional subspace.

Proposition 4.11. *For any herding dynamics with D states and K dimensional feature vectors, the trajectory of the weight vector lies in a subspace of a dimension $K^* \leq \max\{D-1, K\}$. Also, there exists an equivalent herding dynamics with D states and K^* dimensional feature vectors, which generates the same sequence of samples.*

Proof. Let $\{\boldsymbol{\phi}(x_d)\}_{d=0}^{D-1}$ be the set of D state feature vectors. Denote by Φ the subspace spanned of the set of $D-1$ vectors, $\{\boldsymbol{\phi}(x_d) - \boldsymbol{\phi}(x_0)\}_{d=1}^{D-1}$ in $\mathbb{R}^K$, and by $\Phi^\perp$ its complement. The dimension of Φ is apparently at most $\max\{D-1, K\}$. We want to construct a herding dynamics in Φ that generates the same sequence of states as the original dynamics.

Decompose the initial weight vector $\mathbf{w}_0$ and all the feature vectors into Φ and $\Phi^\perp$, denoting the component in Φ with a superscript $^\parallel$ and in $\Phi^\perp$ with $^\perp$. Then $\boldsymbol{\phi}^\perp(x_d) = (\boldsymbol{\phi}(x_d) - \boldsymbol{\phi}(x_0) + \boldsymbol{\phi}(x_0))^\perp = \boldsymbol{\phi}^\perp(x_0), \forall d$ as $\boldsymbol{\phi}(x_d) - \boldsymbol{\phi}(x_0) \in \Phi$, and $\boldsymbol{\phi}^\parallel(x_d) = \boldsymbol{\phi}(x_d) - \boldsymbol{\phi}^\perp(x_0), \forall d$. Consequently $\bar{\boldsymbol{\phi}}^\parallel = \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}^\perp(x_0)$ as $\bar{\boldsymbol{\phi}}$ is a convex combination of the feature vectors.

Let us consider a new herding dynamics (denoted by a superscript *) with feature vectors $\{\boldsymbol{\phi}^\parallel(x_d)\}_{d=0}^{D-1}$ and the moment $\bar{\boldsymbol{\phi}}^\parallel$. We initialize with a weight vector $\mathbf{w}_0^* = \mathbf{w}_0^\parallel$. As Φ is closed with respect to the herding update in Equation 4.8 $\mathbf{w}_t^* \in \Phi, \forall t \geq 0$. Now we want to show that the set of samples $S_T^* \stackrel{\text{def}}{=} \{s_t^*\}_{t=1}^T$ is the same as $S_T \stackrel{\text{def}}{=} \{s_t\}_{t=1}^T$ for any $T \geq 0$.

Obviously this holds at $T = 0$ as $\mathbf{w}_0^* \in \Phi$ and $S_T^* = S_T = \emptyset$. Assume that $S_T^* = S_T$ holds for some $T \geq 0$. Following the recursive representation of $\mathbf{w}_T$ in Equation 4.14, we get

$$\mathbf{w}_T^* = \mathbf{w}_0^* + T\bar{\boldsymbol{\phi}}^\parallel - \sum_{t=1}^{T} \boldsymbol{\phi}^\parallel(\mathbf{s}_t) = \mathbf{w}_0 - \mathbf{w}_0^\perp + T\bar{\boldsymbol{\phi}} - \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{s}_t) = \mathbf{w}_T - \mathbf{w}_0^\perp \quad (4.70)$$

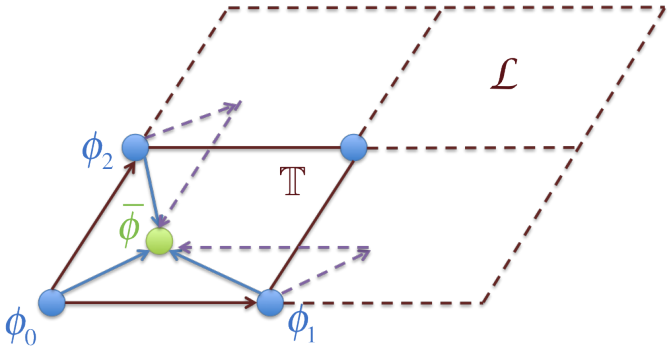


Figure 4.17: Example of the torus projection on herding dynamics with 3 states and 2-dimensional feature vectors. The red lines show the lattice and the torus (solid only) formed by $\boldsymbol{\phi}(x_1) - \boldsymbol{\phi}(x_0)$ and $\boldsymbol{\phi}(x_2) - \boldsymbol{\phi}(x_0)$, and the purple dashed arrows show that the herding dynamics corresponds to a constant rotation on the torus $\mathbb{T}^2$.

The sample to be generated at iteration $T+1$ is computed as

$$\mathbf{s}^*_{T+1} = \arg\max_x (\mathbf{w}^*_T)^T \boldsymbol{\phi}^{\parallel}(x) = \arg\max_x (\mathbf{w}_T)^T \boldsymbol{\phi}(x) - (\mathbf{w}^{\perp}_0)^T \boldsymbol{\phi}^{\perp}(x_0) = \mathbf{s}_{T+1} \tag{4.71}$$

Therefore, $S^*_{T+1} = S_{T+1}$, and consequently $S^*_T = S_T, \forall T \in [0, \infty)$ by induction. As a by-product of Equation 4.70, we observe that the trajectory of the original herding dynamics $\{\mathbf{w}_t\}$ lies in the K^* dimensional affine subspace, $\mathbf{w}^{\perp}_0 + \Phi$. □

The proposition above suggests that the number of effective dimensions of the feature vector is upper-bounded by the number of states in the herding system. Also, the orthogonal component in the initial weight vector $\mathbf{w}^{\perp}_0$ does not affect the sequence of generated samples. In our example of sampling a D-valued discrete distribution with the 1-of-D encoding, the D feature vectors $\{\boldsymbol{\phi}(x_d)\}_{d=1}^{D-1}$ are linearly independent with each other and hence we achieve the maximum number of feature dimensions $K^* = D - 1$. The affine subspace can be easily computed as $\{\mathbf{w} : \sum_{d=1}^{D} w_d = 1\}$. In the rest of this subsection, we will study the characteristics of a relatively more general type of herding dynamics with $D = K + 1$ states, whose feature vectors consist of a linearly independent set in the K dimensional feature space.

Let $\mathcal{L}$ be the lattice formed by the set of vectors $\{\boldsymbol{\phi}(x_d) - \boldsymbol{\phi}(x_0)\}_{d=1}^{K}$, and let $\mathbb{T}^K$ be the K dimensional torus $\mathbb{R}^K/\mathcal{L}$. A torus is a circular space with every pair of opposite edges connected with each other. See Figure 4.17 for an example of a 2D torus. Denote by $G : \mathbb{R}^K \to \mathbb{T}^K$ the canonical projection. For any point $u \in \mathbb{R}^K$, we have the property that

$G(u + (\boldsymbol{\phi}(x_d) - \boldsymbol{\phi}(x_0))) = G(u), \forall d = 0, \ldots, K$. Let $\mathcal{T} : \mathbb{R}^K \rightarrow \mathbb{R}^K$ be the mapping of the herding dynamics in the feature space, which takes the form of a translation $\mathcal{T}(\mathbf{w}) = \mathbf{w} + \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(x(\mathbf{w}))$, where $x(\mathbf{w})$ is the sample to be generated by Equation 4.7. We can observe that the herding update on $\mathbf{w}$ corresponds a rotation on the torus:

$$\begin{aligned} G \circ \mathcal{T}(\mathbf{w}) &= G(\mathbf{w} + \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(x(\mathbf{w}))) \\ &= G(\mathbf{w} + (\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(x_0)) - (\boldsymbol{\phi}(x(\mathbf{w})) - \boldsymbol{\phi}(x_0))) \\ &= G(\mathbf{w}) + (\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(x_0)), \forall \mathbf{w} \in \mathbb{R}^K \end{aligned} \tag{4.72}$$

where the translation operator in $\mathbb{T}^K$ in the last equation refers to a rotation in the torus. This is an interesting property of herding with a maximum number of feature dimensions as it suggests that no matter what sample the dynamics takes, the trajectory of $\mathbf{w}$ under the torus projection is driven by a constant rotation. Furthermore, if the set of elements in the translation vector $\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(x_0)$ is independent on rational numbers[14], the trajectory on $\mathbb{T}^K$ fills the entire torus, which leads to a non-fractal attractor set with a finite volume in the original feature space.

4.8 References

K. Aihara and G. Matsumoto. Temporally coherent organization and instabilities in squid giant axons. *Journal of theoretical biology*, 95(4):697–720, 1982.

O. Angel, A. E. Holroyd, J. B. Martin, and J. Propp. Discrete low-discrepancy sequences. *arXiv preprint arXiv:0910.1077*, 2009.

F. Bach, S. Lacoste-Julien, and G. Obozinski. On the equivalence between herding and conditional gradient algorithms. In J. Langford and J. Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1359–1366, New York, NY, USA, July 2012. Omnipress. ISBN 978-1-4503-1285-1.

C. M. Bishop et al. *Pattern Recognition and Machine Learning*, volume 1. springer New York, 2006.

H. Block and S. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. *Proceedings of the American Mathematical Society*, 26(2):229–235, 1970.

L. Bornn, Y. Chen, N. de Freitas, M. Eskelin, J. Fang, and M. Welling. Herded Gibbs sampling. In *Proceedings of the International Conference on Learning Representations*, 2013.

14. Independence of a set of numbers, $x_1, \ldots, x_K$, on rational numbers means that there does not exist a set of rational numbers $a_1, \ldots, a_K$ that are not all zeros, such that $\sum_{d=1}^{K} a_d x_d = 0$.

M. Boshernitzan and I. Kornfeld. Interval translation mappings. *Ergodic Theory and Dynamical Systems*, 15(5):821–832, 1995.

O. Breuleux, Y. Bengio, and P. Vincent. Quickly generating representative samples from an rbm-derived process. *Neural Computation*, pages 1–16, 2011.

Y. Chen, A. Smola, and M. Welling. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 109–116, Corvallis, Oregon, 2010. AUAI Press.

Y. Chen, A. E. Gelfand, and M. Welling. *Advanced Structured Prediction*, chapter Herding for Structured Prediction, page 187. The MIT Press, 2014.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, page 8. Association for Computational Linguistics, 2002.

Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

A. Gelfand, Y. Chen, L. van der Maaten, and M. Welling. On herding and the perceptron cycling theorem. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 694–702, 2010.

A. Goetz. Dynamics of piecewise isometries. *Illinois Journal of Mathematics*, 44 (3):465–478, 2000.

N. Harvey and S. Samadi. Near-optimal herding. In *Proceedings of The 27th Conference on Learning Theory*, pages 1165–1182, 2014.

G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

F. Huszar and D. Duvenaud. Optimally-weighted herding is Bayesian quadrature. In *Proceedings of the Twenty-Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pages 377–386, Corvallis, Oregon, 2012. AUAI Press.

H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. In *Proceedings of the* 25^{th} *International Conference on Machine learning*, pages 536–543. ACM, 2008.

K. Lu and J. Wang. Construction of Sturmian sequences. *J. Phys. A: Math. Gen.*, 38:2891–2897, 2005.

G. A. H. Marston Morse. Symbolic dynamics ii. sturmian trajectories. *American Journal of Mathematics*, 62(1):1–42, 1940. ISSN 00029327, 10806377. URL `http://www.jstor.org/stable/2371431`.

M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*, volume 1988. MIT press Cambridge, MA, 1969.

R. Neal. Connectionist learning of belief networks. *Articial Intelligence*, 56:71–113, 1992.

R. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Computer Science, 1993.

G. Papandreou and A. Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, pages 193–200, Barcelona, Spain, Nov. 2011. doi: 10.1109/ICCV.2011.6126242.

S. Parise and M. Welling. Learning in Markov random fields: An empirical study. In *Joint Statistical Meeting*, volume 4, page 7, 2005.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

R. Salakhutdinov. Learning deep Boltzmann machines using adaptive MCMC. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 943–950, Haifa, Israel, June 2010. Omnipress. URL `http://www.icml2010.org/papers/441.pdf`.

R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58(2):80–88, 1987.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine Learning*, volume 25, pages 1064–1071, 2008.

T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the International Conference on Machine Learning*, volume 26, pages 1064–1071, 2009.

M. Tsodyks, K. Pawelzik, and H. Markram. Neural networks with dynamic synapses. *Neural Computation*, 10(4):821–835, 1098.

M. Welling. Herding dynamical weights to learn. In *Proceedings of the 21st International Conference on Machine Learning*, Montreal, Quebec, CAN, 2009a.

M. Welling. Herding dynamic weights for partially observed random field models. In *Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 599–606, Corvallis, Oregon, 2009b. AUAI Press.

M. Welling and Y. Chen. Statistical inference using weak chaos and infinite memory. In *Proceedings of the Int'l Workshop on Statistical-Mechanical Informatics (IW-SMI 2010)*, pages 185–199, 2010.

L. Younes. Parametric inference for imperfectly observed Gibbsian fields. *Probability Theory and Related Fields*, 82:625–645, 1989.

D. Young. Iterative methods for solving partial difference equations of elliptic type. *Trans. Amer. Math. Soc*, 76(92):111, 1954.

A. Yuille. The convergence of contrastive divergences. In *Advances in Neural Information Processing Systems*, volume 17, pages 1593–1600, 2004.