UNIVERSITY OF CALIFORNIA,
IRVINE

Herding: Driving Deterministic Dynamics to Learn and Sample Probabilistic Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Yutian Chen

Dissertation Committee:
Professor Max Welling, Chair
Professor Padhraic Smyth
Assistant Professor Alexander Ihler

2013

DEDICATION

To my wife Jinyi and our lovely daughter Sophie.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I owe my deepest gratitude to my advisor Max Welling for his guidance throughout my entire doctorate program. I feel extremely fortune to meet Max and have him be my mentor. He introduced me into the area of machine learning, provided countless support and encouragement at every step of my research, and shared inspirational insights that guided me to plan my future career. Without his persistent help this dissertation would not be possible.

I would like to express my appreciation to my other committee members Padhraic Smyth and Alex Ihler, who were more than generous with their expertise and precious time. I am also grateful to have wonderful collaboration with Alex Smola, Charless Fowlkes, Lauren Van Der Maaten, Nando de Freitas, Luke Bornn, Mareija Eskelin, Andrew Gelfand, Anoop Korattikara, and Sungjin Ahn. Furthermore, I would like to acknowledge the help of my internship supervisor Kevin Murphy, who showed me the application of machine learning to real-world large-scale problems. I am also thankful to Babak Shahbaba for enlightening guidance and discussion on MCMC methods.

Finally, I wish to thank my family. My wife, Jinyi An, has sacrificed a lot in order to accompany and support me throughout these years in Irvine. I owe a special gratitude to my parents for their love and inspiring my curiosity in science. I would also like to thank my parents-in-law for their help.

CURRICULUM VITAE

Yutian Chen

EDUCATION

Doctor of Philosophy in Computer Science **2013**
University of California, Irvine *Irvine, California*

Master of Science in Computer Science **2009**
University of California, Irvine *Irvine, California*

Bachelor of Engineer in Electronic Engineering **2007**
Tsinghua University *Beijing, China*

EXPERIENCE

Graduate Research Assistant **9/2008–6/2009, 06/2010–Present**
University of California, Irvine *Irvine, California*

Software Engineer Intern **6/2012–9/2012**
Google, Mountain View *Mountain View, California*

University of California, Irvine *Irvine, California*
Software Engineer Intern **6/2011-9/2011**
Google, New York *New York, New York*

Teaching Assistant **9/2009–6/2010**
University of California, Irvine *Irvine, California*

SRT Research Assistant **2/2006–7/2007**
Tsinghua University *Beijing, China*

COMPUTER SKILLS

Proficient skills in **Matlab**,**C/C++**.
Development experience with **Python**, **Java**, **Shell scripts**.
Hardware design experience with **VHDL**, **Verilog HDL** on **FPGA**.

TALKS

Efficient Sampling with Kernel Herding.
AI seminar at University of Toronto, 2011; AI/ML seminar at UC Irvine, 2011; ITA Workshop, 2012.

Dynamical Products of Experts for Modeling Financial Time Series.
ICML, 2010.

Bayesian Extreme Components Analysis.
IJCAI 2009.

REFEREED CONFERENCE PUBLICATIONS

Herded Gibbs Sampling **2013**
Luke Bornn, Yutian Chen, Nando de Freitas, Mareija Eskelin, Jing Fang and Max Welling. International Conference on Learning Representations (ICLR).

Distributed and Adaptive Darting Monte Carlo through Regenerations **2013**
Sungjin Ahn, Yutian Chen and Max Welling. Artificial Intelligence and Statistics.

Bayesian Estimation for Partially Observed MRFs **2013**
Yutian Chen and Max Welling. Artificial Intelligence and Statistics.

Bayesian Structure Learning for Markov Random Fields with a Spike and Slab Prior **2012**
Yutian Chen and Max Welling. Uncertainty in Artificial Intelligence.

Integrating Local Classifiers through Nonlinear Dynamics on Label Graphs with an Application to Image Segmentation **2011**
Yutian Chen, Andrew Gelfand, Charless Fowlkes and Max Welling. International Conference on Computer Vision.

On Herding and the Perceptron Cycling Theorem **2010**
Andrew Gelfand, Lauren Van Der Maaten, Yutian Chen, Max Welling. Neural Information Processing Systems.

Super-Samples from Kernel Herding **2010**
Yutian Chen, Max Welling and Alex Smola. Uncertainty in Artificial Intelligence.

Dynamic Product of Experts for Modeling Financial Time Series **2010**
Yutian Chen and Max Welling. International Conference on Machine Learning.

Statistical Inference Using Weak Chaos and Infinite Memory **2010**
Max Welling and Yutian Chen. Proceedings of the Int'l Workshop on Statistical-Mechanical Informatics.

Parametric Herding **2010**
Yutian Chen and Max Welling. Artificial Intelligence and Statistics.

Bayesian Extreme Components Analysis **2009**
Yutian Chen and Max Welling. International Joint Conferences on Artificial Intelligence.

ABSTRACT OF THE DISSERTATION

Herding: Driving Deterministic Dynamics to Learn and Sample Probabilistic Models

By

Yutian Chen

Doctor of Philosophy in Computer Science

University of California, Irvine, 2013

Professor Max Welling, Chair

The herding algorithm was recently proposed as a deterministic algorithm to learn Markov random fields (MRFs). Instead of obtaining a fixed set of model parameters, herding drives a set of dynamical parameters and generates an infinite sequence of model states describing the learned distribution. It integrates the learning and inference phases effectively, and entertains a fast rate of convergence in the inference phase compared to the widely used Markov chain Monte Carlo methods. Herding is therefore a promising alternative to the conventional training-prediction two step paradigm in applying MRFs. In this thesis we study the properties of the herding algorithm from the perspective of both a statistical learning method and a nonlinear dynamical system. We further provide a mild condition for its moment matching property to hold and thereby derive a general framework for the herding algorithm. Under that framework three extensions of herding dynamics are proposed with a wide range of applications. We also discuss the application of herding as a sampling algorithm from the input distribution. Two more variants of herding are introduced to sample discrete and continuous distributions respectively, accompanied with discussion on the conditions when the sampler is unbiased.

Chapter 1

Introduction

1.1 Background

A typical statistical approach to machine learning problems is to first fit a probabilistic model to variables of interest and then apply that model to various tasks such as classification, clustering and prediction. A probabilistic model is usually learned from a set of observed data cases so as to match the empirical frequencies of events. For examples, the bias of coin-toss is estimated from the number of heads and tails in a series of trials; the bigram of pairs of words is learned from their co-occurrence in a large corpus (Jelinek, 1990).

When the number of random variables grows, obtaining a reliable model for the joint frequencies of all variables becomes increasingly difficult because the number of required training data cases would grow exponentially. For instance, in order to predict the outcome of a tic-tac-toe game with a 3×3 board size based on an incomplete game, we can simply perform a lookup in a collection of past games and compute the conditional probability of an outcome given any of the $3^9 = 531441$ states. However, to predict the final status of the Go game with a 19×19 board size, we would need an astronomical amount of data, around 3^{361}

data-cases.

Graphical models (Pearl, 1988) have been proposed to factorize the joint distribution by exploiting the conditional independence among variables, and therefore to obtain reliable model estimation given a much smaller size of data. With the maximum likelihood (ML) criterion, the Bayesian networks are estimated to match the conditional distribution of individual variables to observed frequencies, and the Markov networks, a. k. a. Markov random fields (MRFs), are estimated to match the expected value of features that are usually defined on a small subset of variables (Koller & Friedman, 2009).

Take the example of the Go game. A conditional random field (Lafferty et al., 2001; Sutton, 2012) (CRF) model, a discriminative variant of MRF, has been proposed to match the conditional frequency of small patches on the board, and has obtained promising performance on this difficult problem (Stern et al., 2004). The CRF model considers the conditional distribution of the final status of neighboring stones given their current colors as shown in Figure 1.1. It is obviously much easier to study the distribution of a small patch than to model the whole board. Once we obtain these local conditional frequencies, or features, from a training dataset, a CRF is trained to maximize the entropy of the joint distribution under the constraints of matching these averages over features (Berger et al., 1996).

Although MRFs/CRFs have been applied successfully in various areas, among which are image segmentation (Blake et al., 2004), natural language processing (Sha & Pereira, 2003), social networks (Wasserman & Pattison, 1996) and bioinformatics (Li et al., 2007), these models are usually difficult to work with. Firstly, obtaining the maximum likelihood estimation (MLE) of a generic MRF during the training phase is an intractable problem. This is because the optimization procedure involves carrying out inference in the model and the time complexity of exact inference is exponential to the maximum tree-width of the graph (Pearl, 1988). Researchers have resorted to approximate inference methods such as variational methods (Wainwright et al., 2003) and Markov chain Monte Carlo (MCMC) (Geyer,

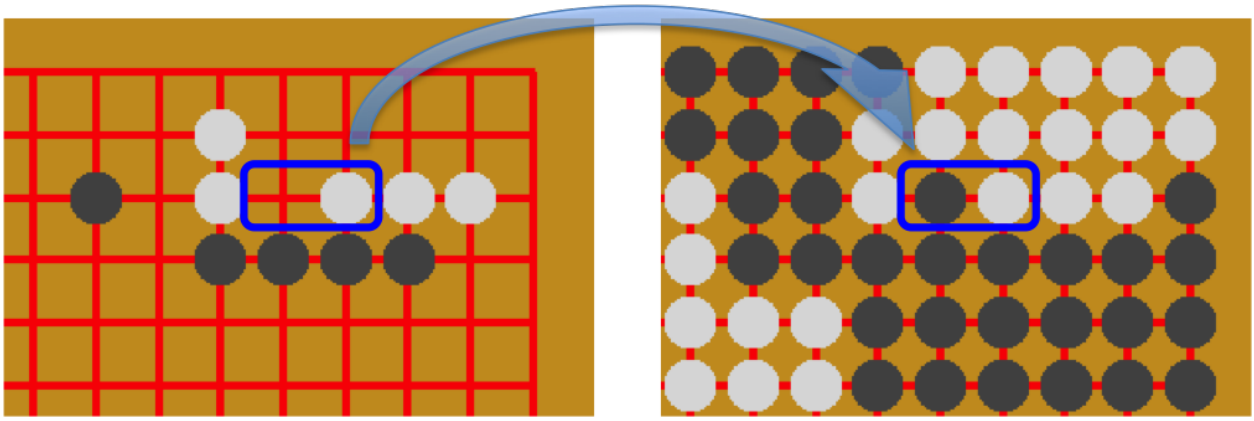


Figure 1.1: Part of a Go board of an incomplete game (left) and its final territory (right). The CRF model takes as a feature the final territory of a pair of neighboring stones conditioned on the current stones.

1991; Neal, 1992; Hinton, 2002; Tieleman, 2008). However, variational methods introduce systematic bias, and MCMC methods suffer from the problem of slow mixing when the model distribution is multi-modal leading to a high computational burden. There are some training algorithms in the literature based on other inductive principles than maximum likelihood, such as pseudo-likelihood (Besag, 1975) and composite likelihood (Lindsay, 1988). They do not have the intractable computational problem, but these estimators are statistically less efficient than MLE (van Duijn et al., 2009) and do not apply to partially observed models in general (Parise & Welling, 2005). Secondly, even when a good model has been estimated, applying the model in the test phase is often an intractable problem again. This is because typical applications such as classification, structured prediction, and computing evidence are all inference problems. Variational methods and MCMC methods are hence applied here again for approximate estimation with the same issues as in the training phase.

The herding algorithm was proposed in Welling (2009a) as an efficient alternative to joint learning and predicting for fully visible MRFs with discrete variables. Observing that in the MCMC approach we have to generate samples in both training and test phases and that what would really be used eventually are the samples, herding combines these two phases by simulating a single deterministic Markov chain throughout the whole process. Instead

of the conventional "training-predicting" paradigm, herding can be considered as a system that generates a sequence of new samples directly from training data such that their average feature-values match the observed averages in the training data. The model parameters become auxiliary variables that are updated only to produce a new sample in the next iteration. The herding algorithm was later extended to partially observed MRFs (POMRFs) in Welling (2009b). However, the behavior of this deterministic algorithm as well as its strength and weakness compared to regular learning algorithms remains to be understood.

1.2 Our Approach and Contributions

In this thesis, we first introduce the statistical properties of the standard herding algorithm (discovered in Welling (2009a,b); Chen et al. (2010); Welling & Chen (2010)) from the aspects of a learning algorithm for MRFs and a nonlinear dynamic system. We interpret herding as an algorithm that greedily generates new samples to match the statistics (moments) of the training data. The output sample sequence exhibits negative autocorrelation and consequently obtains a higher rate of convergence, $\mathcal{O}(1/T)$, than that of either MCMC and independent samples, $\mathcal{O}(1/\sqrt{T})$, where T denotes the number of samples. This helps mitigate the slow mixing problem in the MCMC approach. From the aspect of nonlinear dynamics, the sequence of samples is shown to be weakly chaotic a.k.a. on the edge-of-chaos. Although totally deterministic, a sequence on the edge-of-chaos appears similar to a random sequence. More importantly the learnable information conveyed in that sequence is likely to be richer than that in a random sequence (see Section 2.6.4 for more detail).

The main contribution of this thesis is to give a general condition for the fast rate of convergence on the sampling moments. Based on that condition, we extend the herding algorithm beyond training fully visible MRFs. The work of Welling (2009b) on POMRFs fits well into this general framework and we propose a parametric variant with improved efficiency.

The generalized herding also applies to training discriminative models (e.g. multi layer perceptrons and CRFs) with competitive performance. We also study the case when the convergence condition does not hold and propose a novel alternative to CRF models with a trivial training phase. We show how to apply this method to image segmentation and the final territory prediction of the Go game.

Moreover, we interpret herding as a pure sampling algorithm when the input set of training data is replaced by a distribution and the output sample sequence is enforced to converge to the same distribution. We apply herding as a deterministic version of the Gibbs sampling algorithm with discrete variables and also extend the original herding algorithm to sampling in a continuous state space. Its fast convergence rate, $\mathcal{O}(1/T)$, grants herding the potential to outperform the "ideal" independent random sampler.

Since its inception, herding has been shown to be related to a variety of methods in different domains. The idea of introducing negative auto-correlations to speed up the mixing of Markov chains in learning MRFs is shared with prior work such as the fast-weight persistent contrastive divergence method (Tieleman & Hinton, 2009) and the adaptive Monte Carlo method (Salakhutdinov, 2010). The use of dynamical model parameters also shows similarity with a recently proposed algorithm for learning MRFs, called perturb-and-map (Papandreou & Yuille, 2011). The interpretation as a greedy algorithm for matching average features is closely related to the conditional gradient algorithm in optimization (Bach et al., 2012) and the Bayesian quadrature method in Gaussian processes (Huszar & Duvenaud, 2012). The impressive $\mathcal{O}(1/T)$ convergence rate as a deterministic Monte Carlo method is reminiscent of Quasi Monte Carlo methods (Niederreiter, 1992; Chentsov, 1967) and the rotor-router algorithm (Holroyd & Propp, 2010) in the area of MCMC.

1.3 Outline

We introduce the original herding algorithm and study its statistical properties in Chapter 2. We then discuss extensions of herding as a learning algorithm with applications in Chapter 3. Chapter 4 applies herding as a pure sampling algorithm and shows its applications as a deterministic Gibbs sampler on discrete variables and as an infinite-memory sampler on continuous variables. The thesis is concluded with a summary and a discussion for future work in Chapter 5.

Chapter 2

Herding Model Parameters

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 $\{\phi_\alpha(\mathbf{x})\}$ 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:

$$P = \arg\max_{\mathcal{P}} \mathcal{H}(\mathcal{P}) \quad \text{s.t. } \mathbb{E}_{\mathbf{x}\sim\mathcal{P}}[\phi_\alpha(\mathbf{x})] = \bar{\phi}_\alpha, \ \forall\alpha \tag{2.1}$$

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:

$$\mathbf{w}_{\text{MLE}} = \arg\max_{\mathbf{w}} P(\mathcal{D}; \mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{D} P(\mathbf{x}_i; \mathbf{w}), \quad P(\mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left(\sum_\alpha w_\alpha \phi_\alpha(\mathbf{x})\right) \tag{2.2}$$

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 the Lagrange multiplier to enforce the constraints in the primal form 2.1. 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

$$\ell(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{D} \sum_{i=1}^{D} \log P(\mathbf{x}_i; \mathbf{w}) = \mathbf{w}^T \bar{\boldsymbol{\phi}} - \log Z(\mathbf{w}) \tag{2.3}$$

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

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta(\bar{\boldsymbol{\phi}} - \mathbb{E}_{\mathbf{x} \sim P(\mathbf{x}; \mathbf{w}_t)}[\boldsymbol{\phi}(\mathbf{x})]) \tag{2.4}$$

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 we can not compute it efficiently in general. A common solution is to approximate that quantity by drawing samples using Markov chain Monte Carlo (MCMC) method 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 (Tieleman & Hinton, 2009; Salakhutdinov, 2009, 2010).

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 computation of the partition function. One has to once more resort to running MCMC 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.

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 & Hinton (2009). That 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 $\mathbb{E}_{P(\mathbf{x};\mathbf{w})}[\boldsymbol{\phi}(\mathbf{x})] \approx \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{\phi}(\tilde{\mathbf{x}}_m)$ in the MCMC approach, after each update in Equation 2.4, the parameter $\mathbf{w}$ is translated along the direction that tends to reduce the inner product of $\mathbf{w}^T\boldsymbol{\phi}(\tilde{\mathbf{x}}_m)$, and thereby reduces the state probability around the region of the current samples. The change in the state distribution helps the MCMC sampler to 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 one always needs to know the value of a fixed set of parameters? As discussed in the previous section, for many applications an exact solution is not available given a trained model and MCMC is adopted to provide an approximation. In that case, a sequence of samples from a proper learned distribution 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 are already available. We actually killed 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 the preceding observation.

Herding was proposed in Welling (2009a) and 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:

$$\ell_T(\mathbf{w}) = \mathbf{w}^T \bar{\boldsymbol{\phi}} - T \log \left(\sum_{\mathbf{x}} \exp \left(\sum_{\alpha} \frac{w_\alpha}{T} \phi_\alpha(\mathbf{x}) \right) \right) \tag{2.5}$$

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

$$\mathbf{s}_t = \arg\max_{\mathbf{x}} \sum_{\alpha} w_{\alpha,t-1} \phi_\alpha(\mathbf{x}) \tag{2.6}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta(\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{s}_t)) \tag{2.7}$$

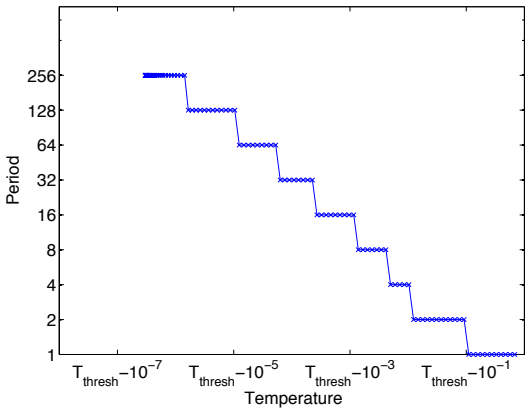
We usually omit the constant learning rate by setting $\eta = 1$ except when mentioned explicitly with the reason explained in Section 2.3.2. We treat the sequence of most probable states, $\{\mathbf{s}_t\}$, as a set of "samples" from herding and use it for inference tasks. At each iteration, we find 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.

It is not difficult to observe that running gradient descent on a tempered MRF with a learning rate, η, is equivalent to running it on a regular MRF with η replaced by η/T and $\mathbf{w}$ replaced by $\mathbf{w}/T$:

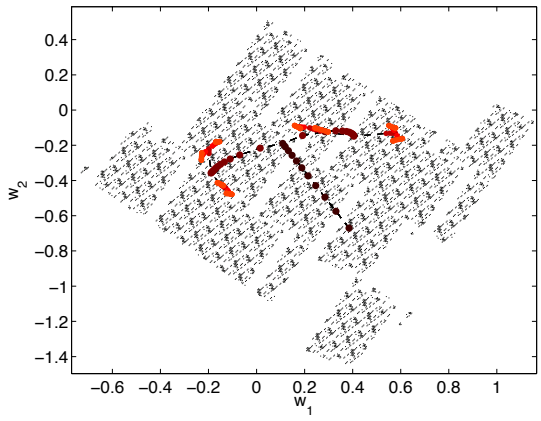
$$\frac{\mathbf{w}_{t+1}}{T} = \frac{\mathbf{w}_t}{T} + \frac{\eta}{T}(\bar{\boldsymbol{\phi}} - \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x};\frac{\mathbf{w}_t}{T})}[\boldsymbol{\phi}(\mathbf{x})]) \tag{2.8}$$

Therefore the herding algorithm in Equations 2.6-2.7 is equivalent to running the gradient descent algorithm on a regular MRF with an infinitely large learning rate.

One can obtain an intuitive impression on the dynamics of herding by looking at the change in the asymptotic behavior of the gradient descent algorithm as we decrease T from a large value towards 0. Assume 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 2.8 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, the attractor set splits from a single point into two points. As T decreases further, the asymptotic oscillation period doubles from two to four,



(a) Asymptotic period of the weight sequence (i.e. size of the attractor set) repeatedly doubles as temperature decrease to a threshold value (right to left). $T_{thresh} \approx 0.116$ in this example. The dynamics transits from periodic to aperiodic at that threshold.



(b) The evolution of the attractor set of the weight sequence. As 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 $T = 0$.

Figure 2.1: Attractor bifurcation for a model with 4 states and 2-dimension feature vectors.

four to eight, etc, and eventually the process approaches an infinite period at a particular temperature. Figure 2.1 shows an example of the attractor bifurcation phenomenon. Starting from that temperature threshold, the trajectory of $\mathbf{w}$ is still bounded in a finite region as shown shortly in Section 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 Figure 2.1b. We discuss this phenomenon in more detail in Section 2.6.

2.3 Tipi Function and Basic Properties of Herding

We will discuss a few distinguishing properties of the herding algorithm in this section. When we take the zero temperature limit in Equation 2.5, the log-likelihood function becomes

$$\ell_0(\mathbf{w}) = \mathbf{w}^T \bar{\boldsymbol{\phi}} - \max_{\mathbf{x}} \left[\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\right] \tag{2.9}$$

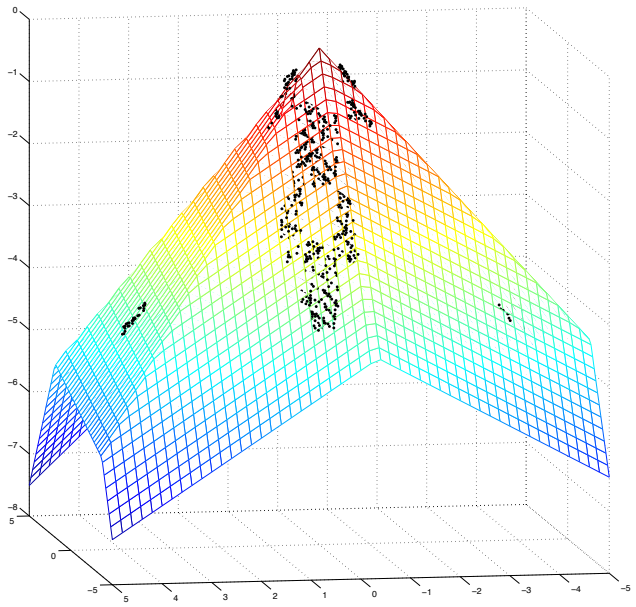


Figure 2.2: "Tipi function"(Welling, 2009a): the log-likelihood function at the zero temperature limit. The black dots show the attractor set of the sequence of $\mathbf{w}_t$.

This function has a number of interesting properties that justify the name "Tipi function"[1] (see Figure 2.2) (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 further more 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 $\mathbf{w}$.

[1] A Tipi is a traditional native Indian dwelling.

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 these weights will never converge. Every flat face of the Tipi function is associated with a state. We will show later in Section 2.5 that the most useful property of herding is that the state sequence obtained by following this gradient descent procedure satisfies the moment matching constraints in Equation 2.1. There are a few more properties of this procedure that are worth noticing.

2.3.1 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 2.6 and 2.7 consist of a nonlinear deterministic mapping of the weights as illustrated in Figure 2.3. 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. As to be discussed in detail in Section 2.6, the herding dynamics show weakly chaotic behavior, and the distribution of the weights may render a fractal attractor set (see Figure 2.2).

2.3.2 Invariant on 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

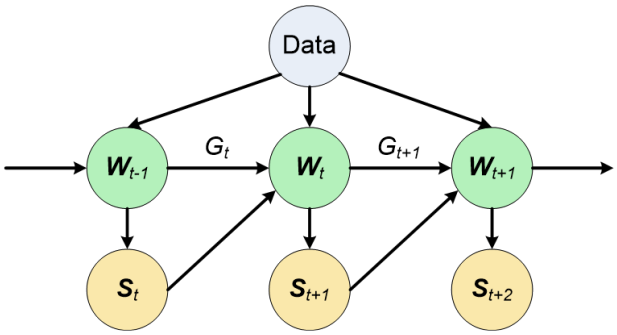


Figure 2.3: Herding as a nonlinear dynamical system.

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 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 2.4. How to choose an optimal set of learning rates is still an open problem.

2.3.3 Negative Auto-correlation

A key advantage of the herding algorithm over sampling using a Markov chain is that the dynamical system mixes very rapidly over the attractor set. This is attributed to the nature of the ever changing model distribution as briefly mentioned at the beginning of this section. 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

$$P(\mathbf{x}; \mathbf{w}_{t-1}) \propto \exp(\mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x})) \tag{2.10}$$

After the weights are updated with Equation 2.7, the probability of the new model becomes

$$\begin{aligned} P(\mathbf{x}; \mathbf{w}_t) &\propto \exp(\mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x})) = \exp((\mathbf{w}_{t-1} + \bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{s}_t))^T \boldsymbol{\phi}(\mathbf{x})) \\ &= \exp\left(\mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}) + \sum_{\mathbf{y} \neq \mathbf{s}_t} \pi(\mathbf{y}) \boldsymbol{\phi}(\mathbf{y})^T \boldsymbol{\phi}(\mathbf{x}) - (1 - \pi(\mathbf{s}_t)) \boldsymbol{\phi}(\mathbf{s}_t)^T \boldsymbol{\phi}(\mathbf{x})\right) \end{aligned} \tag{2.11}$$

Comparing Equation 2.11 with 2.10, the product in the exponent is rewarded by a positive term $\pi(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x})$ for all states but $\mathbf{s}_t$, which after normalization will likely depress the probability of $\mathbf{s}_t$ in the new model and thereby decrease the chance that $\mathbf{s}_t$ is selected as the maximizing state again. Imagine 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 to escape it. The resulting negative auto-correlation would help mitigate the notorious problem of positive auto-correlation in the MCMC method.

We illustrate the negative auto-correlation using a synthetic MRF with 10 discrete states, each associated with a 7-dimensional feature vector. The model parameters are randomly generated and the expected feature values are then computed 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:

$$R(t) = \frac{\frac{1}{T-t}\sum_{\tau=1}^{T-t} \mathbb{I}[s_\tau = s_{\tau+t}] - \sum_s \hat{P}(s)^2}{1 - \sum_s \hat{P}(s)^2} \tag{2.12}$$

where $\mathbb{I}$ is the indication function and $\hat{P}$ 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 2.4. 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" idea is also shared by other sampling methods such as overrelaxation (Young, 1954), fast-weights PCD (Tieleman & Hinton, 2009) and adaptive MCMC (Salakhutdinov, 2010).

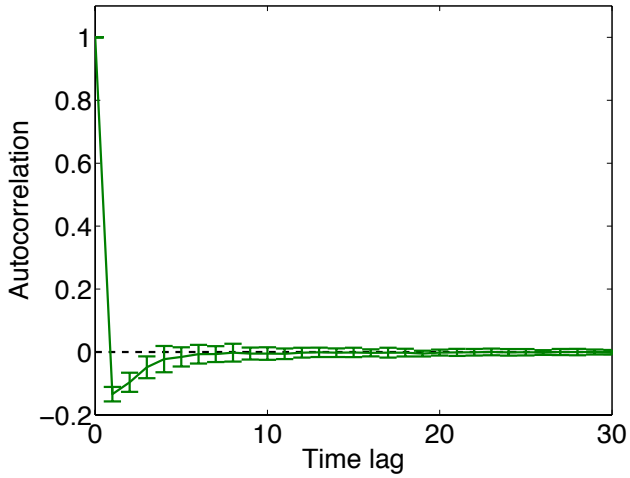


Figure 2.4: Negative auto-correlation of herding samples from a synthetic MRF.

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 2.1 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 2.7:

$$\mathbf{w}_T = \mathbf{w}_0 + T\bar{\boldsymbol{\phi}} - \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{s}_t) \tag{2.13}$$

Plugging 2.13 into Equation 2.6 results in

$$\mathbf{s}_{T+1} = \arg\max_{\mathbf{x}} \langle \mathbf{w}_0, \boldsymbol{\phi}(\mathbf{x}) \rangle + T\langle \bar{\boldsymbol{\phi}}, \boldsymbol{\phi}(\mathbf{x}) \rangle - \sum_{t=1}^{T} \langle \boldsymbol{\phi}(\mathbf{s}_t), \boldsymbol{\phi}(\mathbf{x}) \rangle \tag{2.14}$$

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

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

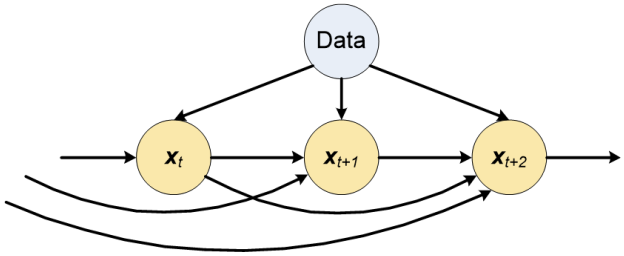


Figure 2.5: Herding as an infinite memory process on samples.

a suitable feature map $\boldsymbol{\phi}$ in the first place. Given the above assumptions, Equation 2.14 becomes

$$\mathbf{s}_{T+1} = \arg\max_{\mathbf{x}} \langle \bar{\boldsymbol{\phi}}, \boldsymbol{\phi}(\mathbf{x}) \rangle - \frac{1}{T+1} \sum_{t=1}^{T} \langle \boldsymbol{\phi}(\mathbf{s}_t), \boldsymbol{\phi}(\mathbf{x}) \rangle \tag{2.15}$$

One can show that the herding update equation 2.15 is equivalent to greedily minimizing the squared error $\mathcal{E}_T^2$ defined as

$$\mathcal{E}_T^2 \stackrel{\text{def}}{=} \left\| \bar{\boldsymbol{\phi}} - \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{s}_t) \right\|^2 \tag{2.16}$$

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 & 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 2.5 because new samples depend on the entire history of samples generated thus far.

2.5 Moment Matching Property

With the dual view in the previous section, the distance between the moments and their sampling average in Equation 2.16 can be considered as the objective function for the herding algorithm to minimize. We discuss in this section under what condition and at what speed the moment constraints will be eventually satisfied.

PROPOSITION **2.1** (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 2.13, we have

$$\frac{1}{\tau}w_{\alpha\tau} - \frac{1}{\tau}w_{\alpha 0} = \bar{\phi}_\alpha - \frac{1}{\tau}\sum_{t=1}^{\tau}\phi_\alpha(\mathbf{s}_t) \tag{2.17}$$

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 **2.2.** *$\forall\alpha$, if there exists a constant R such that $|w_{\alpha,t}| \le R, \forall t$, then $\left|\frac{1}{\tau}\sum_{t=1}^{\tau}\phi_\alpha(\mathbf{s}_t) - \bar{\phi}_\alpha\right| \le \frac{2R}{\tau}$.*

The proof follows immediately Equation 2.17.

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.

2.5.1 Recurrence of the Weight Sequence

As the weights are driven following the update equations on the Tipi function, it is crucial 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 suffices the premise of both Proposition 2.1 and 2.2. It was stated in a corollary of Proposition 2:

PROPOSITION **2.3** (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 **2.4** (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 2.3. Write the herding update equation 2.7 as $\mathbf{w}_t = \mathbf{w}_{t-1} + \nabla_{\mathbf{w}}\ell_0$ (set $\eta = 1$). Take the inner product with $\mathbf{w}_t$ leading to

$$\|\mathbf{w}_t\|_2^2 = \|\mathbf{w}_{t-1}\|_2^2 + 2\mathbf{w}_{t-1}^T\nabla_{\mathbf{w}}\ell_0 + \|\nabla_{\mathbf{w}}\ell_0\|_2^2 \quad \Longrightarrow \quad \delta\|\mathbf{w}\|_2^2 < 2\ell_0 + \mathcal{B}^2 \tag{2.18}$$

where we notice $\ell_0 = \mathbf{w}_{t-1}^T\nabla_{\mathbf{w}}\ell_0$, and $\mathcal{B}$ is an upper bound of $\{\|\nabla_{\mathbf{w}}\ell_0(\mathbf{x})\|_2 : \mathbf{x} \in \mathcal{X}\}$

introduced in Lemma 1 of Welling (2009a).

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 $-\mathcal{B}^2/2$:

$$\ell_0(\mathbf{w}) = \|\mathbf{w}\|_2 \ell_0(\mathbf{w}/\|\mathbf{w}\|_2) \leq \mathcal{R}\ell_0(\mathbf{w}^*) < -\mathcal{B}^2/2, \quad \forall \|\mathbf{w}\|_2 > R \tag{2.19}$$

The proof is concluded by plugging the inequality above in Equation 2.18. □

2.5.2 The Remaining Degrees of Freedom

Both of the herding and the MaxEnt methods match the moments of the training data. But how do they control the remaining degrees of freedom that is otherwise controlled by maximizing the entropy in the MaxEnt method? Unfortunately it is still an open problem and we do not have a principled way except for some heuristics to achieve high entropy. In practice however, 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

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(\beta\left(\sum_{(i,j)\in E} J_{i,j} x_i x_j + \sum_{i\in V} h_i x_i\right)\right), x_i \in \{-1, 1\}, \forall i \in V \tag{2.20}$$

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 & 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:

$$\mathbf{s}_t = \arg\max_{\mathbf{x}} \sum_{(i,j)\in E} w_{(i,j),t-1} x_i x_j + \sum_{i\in V} w_{i,t-1} x_i \tag{2.21}$$

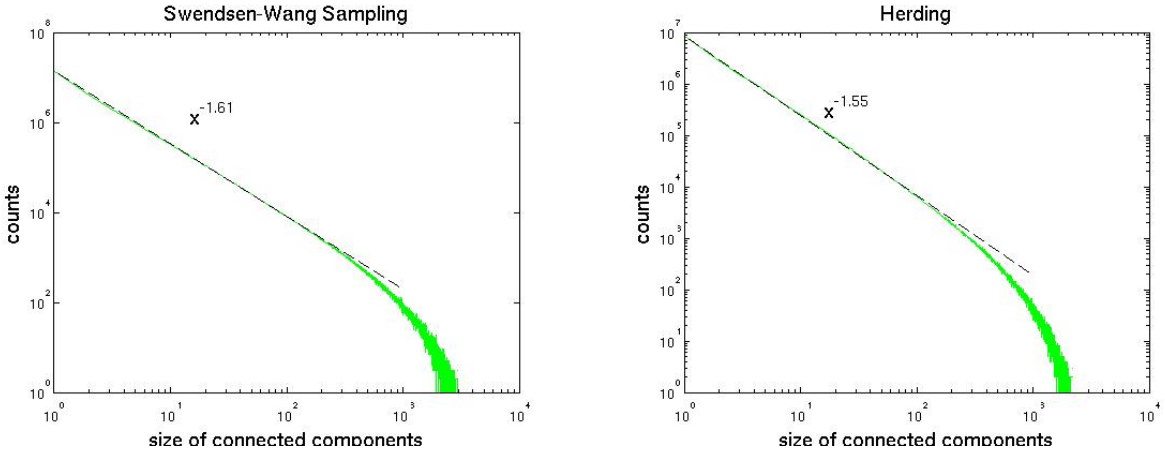
$$w_{(i,j),t} = w_{(i,j),t-1} + \overline{x_i x_j} - s_{i,t} s_{j,t} \tag{2.22}$$

$$w_{i,t} = w_{i,t-1} - s_{i,t} \tag{2.23}$$

As finding the global optimum is an NP-hard problem we find a local maximum for $\mathbf{s}_t$ by coordinate descent. Figure 2.6 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 2.7 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 show the power law distribution with an almost identical

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

Figure 2.6: Sample from an Ising model on an 100×100 lattice at the critical temperature.



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

Figure 2.7: Histogram of the size of connected components in the samples of the Ising model at the critical temperature.

exponent.

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 width of which determines the amount of information it encodes. In Bayesian estimation, the width 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 section. 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. We show that the learnable information contained in the sequence, which is expressed as the sub-extensive part of the entropy, appears to grow faster than in both a converged sequence and a stochastic process.

2.6.1 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,

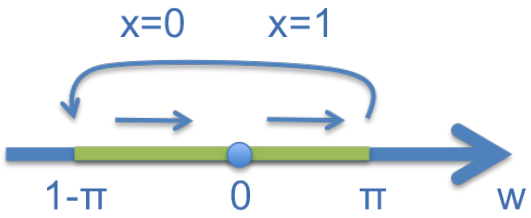


Figure 2.8: Herding dynamics for a single binary variable. At every iteration the weight is first increased by π. If w was originally positive, it is then depressed by 1.

$\pi \in [0, 1]$, 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 $\bar{\phi} = \pi$. Apply the update equations 2.6-2.7 and we get the following dynamics:

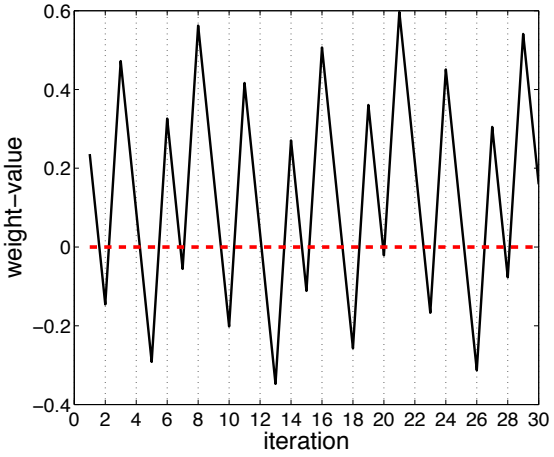
$$s_t = \mathbb{I}(w_{t-1} > 0) \tag{2.24}$$

$$w_t = w_{t-1} + \pi - s_t \tag{2.25}$$

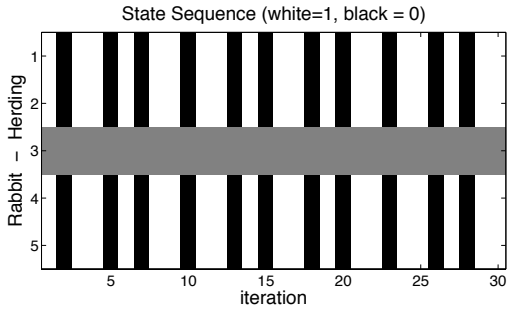
where $\mathbb{I}[\cdot]$ is the indicator function. With the moment matching property we can show immediately that the firing rate converges to the desired value for any initial value of w. The update equations are illustrated in Figure 2.8. This dynamics is a simple type of interval translation mapping (ITM) problem in mathematics (Boshernitzan & Kornfeld, 1995). In a general ITM problem, the invariant set of the dynamics often has a fractal dimension. But for this simple case, the invariant set is the entire interval $(\pi - 1, \pi]$. 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[2]). In Figures 2.9 we plot the weights (a) and the states (b) resulting from herding with the "Fibonacci neuron" model. For a proof, please see Welling & Chen (2010).



(a) 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).



(b) 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.

Figure 2.9: Sequence of weights and states generated by the "Fibonacci neuron" based on herding dynamics.

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 **2.5.** *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 $w_t \in$*

[2]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: $1, 1, 2, 3, 5, 8, ...$

$(\pi - 1, \pi], \forall t \geq \tau$. *Moreover, for* $T \in \mathbb{N}$, *we have:*

$$\left| \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 1] - T\pi \right| \leq 1, \quad \left| \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 0] - T(1-\pi) \right| \leq 1 \tag{2.26}$$

Proof. We first show that $(\pi - 1, \pi]$ is the invariant interval for herding dynamics. Denote the mapping of the weight in Equation 2.24 and 2.25 as $\mathcal{T}$. Then we can see that the interval $(\pi - 1, \pi]$ is mapped to itself as

$$\mathcal{T}(\pi - 1, \pi] = \mathcal{T}(\pi - 1, 0] \cup \mathcal{T}(0, \pi] = (2\pi - 1, \pi] \cup (\pi - 1, 2\pi - 1] = (\pi - 1, \pi] \tag{2.27}$$

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 2.25 over t immediately gives us the first inequality in 2.26 as:

$$T\pi - \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 1] = w_{\tau+T} - w_\tau \in [-1, 1]. \tag{2.28}$$

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

As a corollary of Proposition 2.5, when we initialize $w_0 = \pi - 1/2$, we can improve the bound of the discrepancy by a half.

COROLLARY **2.6.** *If* w *is the weight of the herding dynamics in Proposition 2.5 and it is initialized at* $w_0 = \pi - 1/2$, *then for* $T \in \mathbb{N}$, *we have:*

$$\left| \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 1] - T\pi \right| \leq \frac{1}{2}, \quad \left| \sum_{t=\tau+1}^{\tau+T} \mathbb{I}[s_t = 0] - T(1-\pi) \right| \leq \frac{1}{2} \tag{2.29}$$

The proof immediately follows Equation 2.28 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 2.4. One can see this by constructing an equivalent herding dynamics with a feature

of constant norm as:

$$\phi'(x) = \begin{cases} 1 & \text{if } x = 1 \\ -1 & \text{if } x = 0 \end{cases} \tag{2.30}$$

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 2.4 and therefore the sample sequences in both dynamical systems greedily minimize the error of the empirical probability (up to a constant factor):

$$\left| \frac{1}{T} \sum_{t=1}^{T} \phi'(x'_t) - (2\pi - 1) \right| = 2 \left| \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}[x_t = 1] - \pi \right| \tag{2.31}$$

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

We can view this as a dynamical system on the state space $\{-1, 1\}$, i.e. $s_{t+1} = F_t(s_{[1:t]}; w_0)$. We note that the map depends on the entire history, $\{s_1, ..s_t\}$. This same information is equivalently stored in the current value of the parameter w_t. However, marginalizing over w, the system has infinite memory as shown in Figure 2.5. One can also compute the complexity of the state sequences, which is defined as the total number possible sequences of length T. This number turns out to be exactly $T + 1$, which is the absolute bare minimum for sequences that are not eventually periodic. These facts imply 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 & Wang (2005). (For a proof, please see Welling & Chen (2010).)

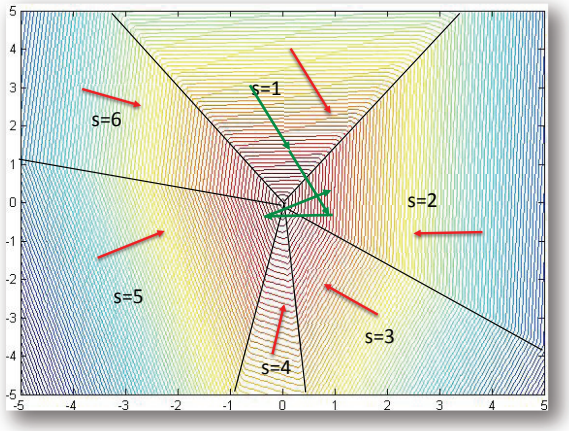


Figure 2.10: Cones in parameter space $\{w_1, w_2\}$ that correspond to the discrete states $s_1, ..., s_6$. Arrows indicate the translation vectors associated with the cones.

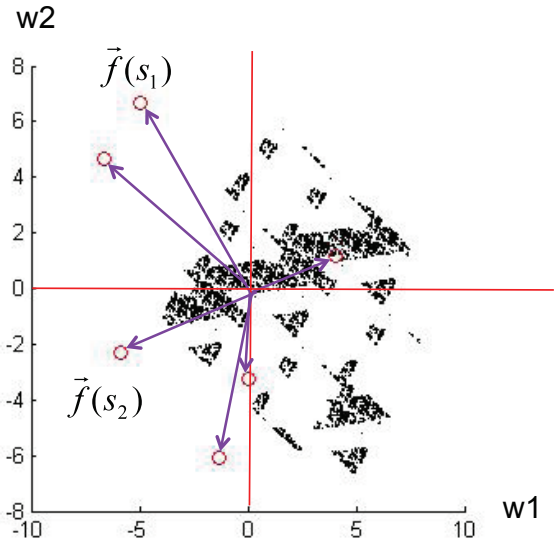


Figure 2.11: Fractal attractor set for herding with two parameters. The circles represent the feature-vectors evaluated the states $s_1, ..., s_6$. Hausdorff dimension for this example is between 0 and 1.

2.6.2 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 2.7 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 2.10 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 2.6. 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[3] everywhere (except perhaps on

[3]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

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 on 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 & Chen (2010)). It can often be observed that this attractor has fractal dimension (see Figure 2.11 for an example). All these facts clearly 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 weak chaos, which is usually defined through its (topological) entropy. We will study the entropy of herding in the next subsection. Figure 2.1 shows another example of the fractal attractor set and the attractor bifurcation phenomenon which is often observed in chaos systems.

Herding is a dissipative system in the sense that there are many locations where a point has more than one pre-image. At these places it becomes obviously impossible to determine w_{t-1} from w_t. In other words, we can not run the system backwards in time, i.e. it is time irreversible. We interpret these regions as "entropy sinks," i.e. regions where entropy is destroyed. These entropy sinks are full dimensional (i.e. of dimension K). On the other hand, entropy is also generated on the boundaries between the cones. We call these $K-1$ dimensional areas "entropy sources." It is here that a region gets split into two (or more) regions and the points in one half become completely uncorrelated with points in the other half.

Note that in breaking a region in two we do not expand the volume of our space at a microscopic level. However at the sinks, regions overlap and we do indeed lose volume there. So, we can conclude that the volume will only decrease as time moves on. Whether the volume will become zero or reach a pedestal remains an open question. For the case of single

$|\delta Z(0)|$ 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.

neuron model, the attractor is the entire interval $(\pi - 1, \pi]$. For a general case with a higher dimension, we conjecture that (under the right conditions) the volume will shrink to zero leading to fractal attractors[4].

Although the $K-1$-dimensional entropy sources do not generate K-dimensional volume at a microscopic level, they do generate volume at a coarse grained level by pushing apart the two halves of the region that it splits. It is this process which is usually referred to when we speak of the second law of thermodynamics, namely that the entropy of a system always increases. Starting with a small (infinitely dense) region we will indeed find that at a coarse grained level it will expand and converge on a stable attractor set. This despite the fact that it will loose K-dimensional volume. Again this picture suggests a fractal structure of the attractor set.

2.6.3 Topological Entropy

In this section we will 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 2.10 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 easily 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." The topological entropy is defined as,

$$h = \lim_{T\to\infty} h(T) = \lim_{T\to\infty} \frac{\log M(T)}{T} \tag{2.32}$$

[4]We have numerically verified this statement for some two dimensional cases.

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.

To count the number of subsequences of length T, 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 & Chen (2010). In fact, we further conjecture 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*.

Thus, to estimate the rate with which the number of sequences grow, we need to estimate how quickly a partition breaks up into smaller pieces. Define $c(t)$ to be the total number of cells (polygons) at time t. Recall that the cells split when they overlap with a cone-boundary. The probability that that happens is proportional to the diameter of the cell, which scales as $d(t) \sim c(t)^{-\frac{1}{K}}$ where K is the number of dimensions (which is equal to the number of herding parameters). This follows because $c(t) \approx (D/d(t))^K$ with D the size of the initial cell. At time t the probability of splitting a particular cell is $\pi \sim d(t)/L$ where L is the diameter of the volume over which the cells are distributed. Therefore, the change in the expected number of cells between time t and $t + \mathrm{d}t$ equals the number of cells at time t times

the probability of a cell splitting: $\mathrm{d}c(t) = c(t)\pi(t)\mathrm{d}t$. Which then gives,

$$\mathrm{d}c(t) = c(t)\pi(t)\mathrm{d}t \sim c(t)d(t)\mathrm{d}t \sim c(t)^{1-\frac{1}{K}}\mathrm{d}t \tag{2.33}$$

which leads to the solution: $c(t) \sim t^K$. It has been rigorously proven that the growth rate must have an exponent less or equal than K (Goetz, 2000), but we conjecture based on the above argument and some preliminary numerical simulations that it is actual equal to K for herding in the typical case (a.k.a. with an incommensurate translation basis, see Appendix B of Welling & Chen (2010)). Thus, if we accept this conjecture, then the entropy $H(T) = Th(T)$ of a sequence of length T (for T large enough) is given by,

$$H(T) = K\log(T) \tag{2.34}$$

This result is interesting, because it implies that there may be more *learnable* information in a herding sequence than in a sequence of IID samples as will be discussed below.

2.6.4 Learnable Information in the Herding Sequence

Let us consider the task of inferring the firing rate of a neuron from a binary sequence (presence or absence of a signal). Suppose the firing rate, π, is an irrational number in this section. When the sequence is generated with herding dynamics, in the previous section we estimated the dominant contribution to the entropy of a herding sequence to be $H_{\mathrm{top}}(T) \sim K\log(T)$ with K the number of parameters ($K = 1$ in this case).

In the Bayesian framework, consider the binary sequence as generated by random sampling from a Bernoulli distribution where we use a Beta-prior on the parameter π. This results in an exchangeable sequence of samples (no longer IID due to the fact we integrate over the π which induces dependencies) and we want to infer the posterior distribution of π given that

sequence. We can compute the Kolmogorov-Sinai entropy of the sequence. The dominant part of that sequence grows linearly in T, i.e. $H_{\text{ext}} = hT$. As explained in Bialek et al. (2001) however, this extensive contribution will not contribute to the predictive information defined as the mutual information between the future and past sequence. The authors argue that it is the sub-extensive part of the entropy that should be interpreted as the amount of predictive information about the future. In fact the authors go one step further in stating that sub-extensive entropy is a good measure of the *complexity* of the sequence where both regular and random sequences have low complexity. The dominant contribution of the sub-extensive part for posterior sampling can easily be computed as $H_{sub}(T) = \frac{1}{2}K \log T$ (with $K = 1$ for the Beta-Bernoulli distribution). This sub-extensive entropy is equal the minimum description length as it measures the number of bits required to encode the learnable (predictable) part of the sequence. It can also be derived as the *bits-back* term (Hinton & Zemel, 1994) in the Bayesian evidence but now applied to the parameters instead of the hidden variables.

Comparing to the entropy of herding[5] we can draw two important conclusions: I) The entropy of herding does not have an extensive component, which we believe is due to infinite memory nature. Thus, all the information in a herding sequence is useful for prediction and no bits are wasted on unpredictable randomness. II) The entropy of herding can grow faster by a factor of two than the sub-extensive part of the entropy of the Bayesian evidence. From this we conclude that the samples of herding can be interpreted as a filtered version of samples obtained from sampling from the corresponding Bayesian model (a Beta-Bernoulli distribution in this case) where the useful signal has been separated from a noisy background. (The term "noisy" is being used in its more general interpretation as useless for prediction.)[6].

[5]We do not expect that the topological entropy and the Kolmogorov-Sinai entropy will differ significantly in their leading order behavior. The KS-entropy can be identified with the H_{sub}.

[6]It is interesting to note that the authors of Bialek et al. (2001) comment on the equivalence of information generated through parameter tying or through other types of infinite range correlations in the data.

2.7 Summary of Contributions

The herding algorithm was proposed by Welling (2009a) and the study on the statistical properties of herding was also mainly attributed to him. My contributions in this chapter include:

- Discussed the negative auto-correlation of the sample sequence generated by herding with a quantitative measurement in Section 2.3.3.
- Applied herding on the Ising model in Section 2.5.2.
- Analyzed the low-discrepancy property of herding on the single neuron model in Section 2.6.1
- Provided a quantitative estimation of the topological entropy in Section 2.6.3

Chapter 3

Generalized Herding

The moment matching property in Proposition 2.1 and 2.2 requires only a mild condition on the L_2 norm of the dynamic weights. That provides us with great flexibility in modifying the original algorithm for more practical implementation as well as a larger spectrum of applications. In this chapter, we will first give a general condition on the recurrence of the weight sequence, from which we discuss how to generalize the herding algorithm. We show three examples extensions of herding, two of which retain the moment matching property and the other one does not but achieves the minimum matching distance in a constraint problem.

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 2.4) whose proof again relies on the premise that the maximization is carried out exactly in the herding update equation 2.6. 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 on the input moments when the samplers are suboptimal states? In this section we give a general and verifiable condition for the recurrence of the weight sequence based on the perceptron cycling theorem (Minsky & 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,

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{x}_{i_t}(y_{i_t} - y^*_{i_t}). \tag{3.1}$$

A critical evaluation by Minsky & Papert (1969) revealed the perceptron's limited representational power. This fact is reflected in the behavior of Rosenblatts 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 & Papert, 1969) states that for the inseparable case the weights remain bounded and do not diverge to infinity.

Consider a sequence of vectors $\{\mathbf{w}_t\}$, $\mathbf{w}_t \in \mathbb{R}^D$, $t = 0, 1, \ldots$ generated by the iterative

procedure in Algorithm 3.1.

Algorithm 3.1 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

PCT was initially introduced in Minsky & Papert (1969) to show the boundedness of the sequence of $\mathbf{w}_t$. It had a gap in the proof that was fixed in Block & Levin (1970) with the following theorem (with slight changes in notation):

THEOREM **3.1** (Boundedness Theorem)**.** $\|\mathbf{w}_t\| \leq \|\mathbf{w}_0\| + M, \forall t \geq 0$ *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

$$\mathbf{w}_t^T \mathbf{v}_t = \mathbf{w}_t^T \mathbf{x}_{i_{t+1}}(y_{i_{t+1}} - y^*_{i_{t+1}}) = |\mathbf{w}_t^T \mathbf{x}_{i_{t+1}}|(\text{sign}(\mathbf{w}_t^T \mathbf{x}_{i_{t+1}}) y_{i_{t+1}} - 1) \leq 0 \tag{3.2}$$

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 2.6, a finite set $V = \{\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$, and observing that

$$\mathbf{w}_t^T \mathbf{v}_t = \mathbf{w}_t^T \bar{\boldsymbol{\phi}} - \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}_{t+1}) \leq 0 \tag{3.3}$$

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 bounded-

ness theorem (or PCT) provides a general condition for the recurrence property and hence the moment matching property of herding. Inequality 3.3 is easy to be verified at running time and does not require $\mathbf{s}_{t+1}$ to be the global optimum.

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., $|V| = C(d, D)K^m$, 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.

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. Welling (2009b) extends herding to partially observed MRFs (POMRFs) which can be considered as an instance of our generalized herding algorithm. However, the proposed algorithm is essentially non-parametric because it requires to access all the data cases instead of the average feature vector at every iteration. We introduce a parametric version of herding that runs on POMRFs in this thesis and evaluate its performance on an image compression application.

3.3.1 Herding for POMRFs (Welling, 2009b)

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,

$$P(\mathbf{x}, \mathbf{z}; \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left(\sum_\alpha w_\alpha \phi_\alpha(\mathbf{x}, \mathbf{z})\right) \tag{3.4}$$

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

$$\ell(\mathbf{w}) = \frac{1}{D} \sum_{i=1}^D \log\left(\sum_{\mathbf{z}_i} \exp\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_i)\right)\right) - \log Z(\mathbf{w}) \tag{3.5}$$

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

$$\ell = \max_{\{Q_i\}} \min_R \frac{1}{D} \sum_{i=1}^D \mathcal{H}(Q_i) - \mathcal{H}(R) + \sum_\alpha w_\alpha \left(\frac{1}{D} \sum_{i=1}^D \mathbb{E}_{Q_i(\mathbf{z}_i)}[\phi_\alpha(\mathbf{x}_i, \mathbf{z}_i)] - \mathbb{E}_{R(\mathbf{x},\mathbf{z})}[\phi_\alpha(\mathbf{x}, \mathbf{z})]\right) \tag{3.6}$$

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

$$\frac{1}{D} \sum_{i=1}^D \mathbb{E}_{Q_i(\mathbf{z}_i)}[\phi_\alpha(\mathbf{x}_i, \mathbf{z}_i)] = \mathbb{E}_{R(\mathbf{x},\mathbf{z})}[\phi_\alpha(\mathbf{x}, \mathbf{z})] \tag{3.7}$$

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 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 3.5 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 can thus be replace with maximizations, resulting in the objective,

$$\ell_0(\mathbf{w}) = \frac{1}{D}\sum_{i=1}^{D} \max_{\mathbf{z}_i} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_i) - \max_{\mathbf{s}} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{s}) \tag{3.8}$$

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:

$$\mathbf{z}_{it}^* = \arg\max_{\mathbf{z}_i} \mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_i), \forall i \tag{3.9}$$

$$\mathbf{s}_t^* = \arg\max_{\mathbf{s}} \mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{s}) \tag{3.10}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \left[\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_{it}^*)\right] - \boldsymbol{\phi}(\mathbf{s}_t^*) \tag{3.11}$$

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.

3.3.2 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

$$\mathbf{w}_t^T \mathbf{v}_t = \left[\frac{1}{D} \sum_{i=1}^{D} \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_{i,t+1}^*) \right] - \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}_{t+1}^*) \tag{3.12}$$

$$= \left[\frac{1}{D} \sum_{i=1}^{D} \max_{\mathbf{z}_i} \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_i) \right] - \max_{\mathbf{s}} \mathbf{w}_t^T \boldsymbol{\phi}(\mathbf{s}) \leq 0 \tag{3.13}$$

The last inequality holds because the second term maximizes over more variables than every term in the summation. Again, we do not have to be able to solve the difficult optimization problems of Equation 3.9 and 3.10. Partial progress in the form of a few iterations of coordinate-wise descent is often enough to satisfy the condition in Equation 3.12 which can be checked easily. Welling (2009b) also proposes a tractable version of herding that is guaranteed to satisfy PCT by starting the maximization for $\mathbf{s}_t^*$ from the training data points.

Following a similar proof as Proposition 2.2, we obtain the fast moment matching property of herding on POMRFs:

PROPOSITION **3.2.** *There exists a constant R such that herding on a partially observed MRF satisfies*

$$\left| \frac{1}{\tau} \sum_{t=1}^{\tau} \frac{1}{D} \sum_{i=1}^{D} \phi_\alpha(\mathbf{x}_i, \mathbf{z}_{it}^*) - \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{s}_t^*) \right| \leq \frac{2R}{\tau}, \forall \alpha \tag{3.14}$$

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 3.2 are in direct analogy to the maximum likelihood problem of Equation 3.5 for which the following moment matching conditions hold at the MLE for all α,

$$\frac{1}{D} \sum_{i=1}^{D} \mathbb{E}_{P(\mathbf{z}_i|\mathbf{x}_i;\mathbf{w}_{\text{MLE}})}[\phi_\alpha(\mathbf{x}_i, \mathbf{z}_i)] = \mathbb{E}_{P(\mathbf{x},\mathbf{z};\mathbf{w}_{\text{MLE}})}[\phi_\alpha(\mathbf{x}, \mathbf{z})] \tag{3.15}$$

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. However, an MLE solution also strives for high entropy in the hidden states. We observe in practice that the herding dynamics usually also induces entropy in the distributions for $\mathbf{z}$ avoiding the decoupling phenomenon described above.

3.3.3 Parametric Herding Algorithms

In the herding algorithm on fully visible MRFs, we can first collect the average features from the data and then run herding with these quantities only. The average features are playing the role of parameters that shape the distribution of the generated samples. However, for herding on POMRFs, we are required to store all the data in order to compute the weight updates in Equation 3.11 because the positive term is a function of both the visible variables and the current weights, and cannot be computed ahead of time. We should view this method as inherently non-parametric as illustrated in Figure 2.3. For instance, the samples can be used to generate a non-parametric kernel estimate of the density function, or the hidden representations can be used as a basis for nearest-neighbor classification. However, it makes one wonder if one can define a parametric variant of herding that decouples the data and replaces them with a collection of parameters.

We consider two approaches to running herding dynamics with hidden variables with a finite set of parameters.

The first approach is proposed in Welling (2009b). It replaces the weight-dependent features in the positive term of Equation 3.11 by average features over time. Noticing that the hidden

variable $\mathbf{z}_{it}^*$ is a function of $\mathbf{x}_i$ and $\mathbf{w}_{t-1}$, and data enters herding only in the positive term so as to compute $\mathbf{z}_{it}^*$ at the current weight setting, we resort to replacing that weight-dependent term by the the average features, called *rates*, that marginalizes out the distribution of weights (equivalently the distribution of hidden variables). This can be achieved through the following online averaging process:

$$\mathbf{r}_t = \frac{t-1}{t}\mathbf{r}_{t-1} + \frac{1}{t}\bar{\boldsymbol{\phi}}_t, t = 1, \ldots \tag{3.16}$$

with $\bar{\boldsymbol{\phi}}_t = \frac{1}{D}\sum_{i=1}^{D}\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{z}_{it}^*)$ and $\mathbf{r}_0 = \mathbf{0}$. Once the learning phase has finished, we can decouple the data and run herding with the rates $\mathbf{r}$ instead:

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{r} - \boldsymbol{\phi}(\mathbf{s}_t^*) \tag{3.17}$$

We can show that PCT conditions hold with this parametric approach and therefore the moment matching property will still be kept in the sense

$$\lim_{\tau\to\infty}\frac{1}{\tau}\sum_{t=1}^{\tau}\phi_\alpha(\mathbf{s}_t^*) = r_\alpha \approx \lim_{\tau\to\infty}\frac{1}{\tau}\sum_{t=1}^{\tau}\bar{\phi}_\alpha(\mathbf{w}_t, \mathcal{D}) \tag{3.18}$$

where the right hand side represents the average features that would be obtained from the non-parametric algorithm. Note however that this is an approximation to the original dynamics since we replace the positive term which is a function of the weights by constants. The number of parameters in this approach is the same as the number of features.

We propose a second approach, called parametric herding (Chen & Welling, 2010), to provide a more flexible parametric model and to potentially mimic the exact herding dynamics better by learning regression functions $r_\alpha(\mathbf{w})$ to approximate $\bar{\phi}_\alpha(\mathbf{w})$. Following the nonlinear dynamical system paradigm of herding with a mapping:

$$\mathbf{w}_t = F(\mathbf{w}_{t-1}, \mathcal{D}) = \mathbf{w}_{t-1} + \bar{\boldsymbol{\phi}}(\mathbf{w}_{t-1}, \mathcal{D}) - \boldsymbol{\phi}(\mathbf{s}_t^*(\mathbf{w}_{t-1})) \tag{3.19}$$

we note that the mapping only depends on the data through the functions $\bar{\boldsymbol{\phi}}(\mathbf{w}_{t-1}, \mathcal{D})$. Therefore, if we can learn a set of parametrized regression functions $\{r_\alpha(w)\}$ that approximates this term, then we can decouple the data entirely and use $\mathbf{r}$ instead of $\bar{\boldsymbol{\phi}}$. We would have turned the nonparametric method into a parametric one.

Fortunately, learning the regression functions $r_\alpha(w)$ is very simple in principle. The reason is that by running the herding equations using data, we generate an unlimited dataset of pairs $\{\mathbf{w}_t, \bar{\phi}_\alpha(\mathbf{w}_t)\}$. Hence,we can take any off-the-shelf regression method to learn the relation between $\mathbf{w}$ and $\bar{\boldsymbol{\phi}}$. Interestingly, we are using supervised learning techniques to solve an unsupervised learning problem. Not unlike Welling et al. (2002) one can argue that the system is "self-supervising."

We take restricted Boltzman machines (RBMs) (Hinton, 2002) as an example of the POMRF and show how to create and train the regression functions. An RBM is an MRF with a bipartite graph structure. The set of features consists of unary features $\phi_j = x_j$, $\phi_k = z_k$, and pairwise features $\phi_{jk} = x_j z_k$, where we use j to index visible variable x_j and $k \in \{1, \ldots, K\}$ to index hidden variable z_k. Accordingly, we have a set of weights consisting of w_j, w_k, w_{jk}. Herding dynamics has been applied to RBMs (using the ± 1 representation) in Welling (2009b) to learn discriminative features. Now we consider how to derive a parametric formulation for this dynamics. Since $\bar{\phi}_j = \frac{1}{D}\sum_{i=1}^{D} x_{i,j}$ is independent of $\mathbf{w}$ we do not need a regression function for it. For the other features, a suitable regression function that respects the bounds $r_{jk} \in [-1, +1]$ and $r_k \in [-1, +1]$ is given by,

$$r_{jk}(\mathbf{w}; \mathbf{A}, \boldsymbol{\pi}, b) = \sum_{m=1}^{M} \pi_m \tanh\left(b\left(\sum_j w_{kj} A_{jm} + w_k\right)\right) A_{jm} \tag{3.20}$$

$$r_k(\mathbf{w}; \mathbf{A}, \boldsymbol{\pi}, b) = \sum_{m=1}^{M} \pi_m \tanh\left(b\left(\sum_j w_{kj} A_{jm} + w_k\right)\right) \tag{3.21}$$

where $\mathbf{A} = \{\mathbf{A}_{\cdot,m}\}$ is a matrix of M columns, each representing a prototype with a weight

parameter π_m. b is a scalar. The choice of this function is motivated by softening the expressions for $\bar{\phi}_{jk}$ and $\bar{\phi}_k$, which involve a maximization operation for $\mathbf{z}^*_{it}$ in Equation 3.9, and meanwhile introducing prototypes to replace the data. The softening is required in order to compute gradients for optimizing the parameters of the regression function. But that would forfeit the scale free property of herding because the regression functions $\mathbf{r}$ now depend on the scale of $\mathbf{w}$ nonlinearly. The parameter b is then introduced to offset the scaling effect.

We estimate the regression parameters by minimizing the average sum of squared residuals (SSE) over the time:

$$C(\mathbf{A}, \boldsymbol{\pi}, b) = \frac{1}{T} \sum_{t=1}^{T} \mathrm{SSE}(t) \tag{3.22}$$

$$\mathrm{SSE}(t) = \sum_j \sum_k (r_{jk}(\mathbf{w}_t; \mathbf{A}, \boldsymbol{\pi}, b) - \bar{\phi}_{jk}(\mathbf{w}_t))^2 + \sum_k (r_k(\mathbf{w}_t; \mathbf{A}, \boldsymbol{\pi}, b) - \bar{\phi}_k(\mathbf{w}_t))^2 \tag{3.23}$$

where $\mathbf{w}_t$ is produced from herding. The objective function represents the expected SSE if we assume that herding will sample from some (unknown) distribution $\hat{p}(\mathbf{w})$. Since herding generates one pair of $(\mathbf{w}_t, \bar{\phi}_t)$ per iteration, it's natural to run an online gradient descent algorithm to update the parameters $\boldsymbol{\pi}, \mathbf{A}, b$.

One can verify that the PCT conditions will hold with the regression function above because $\mathbf{r}(\mathbf{w})$ can be interpreted as a weighted average over a set of artificial data-cases. Therefore we have the following moment matching property

$$\lim_{\tau \to \infty} \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{s}^*_t) = \lim_{\tau \to \infty} \sum_{t=1}^{\tau} r_\alpha(\mathbf{w}_t) \approx \lim_{\tau \to \infty} \frac{1}{\tau} \sum_{t=1}^{\tau} \bar{\boldsymbol{\phi}}(\mathbf{w}_t, \mathcal{D}). \tag{3.24}$$

More importantly, in contrast to the first approach, this parametric algorithm could approximate the exact non-parametric dynamics with an arbitrarily high accuracy given a good

regression model (in a trivial case, the parametric algorithm is exact if we treat all the data-cases as columns in the parameter $\mathbf{A}$). Also, we have more flexibility in the choice of the model complexity in terms of the number of parameters and the form of the regression function.

3.3.4 Application: Data Compression

We evaluate the performance of parametric herding with an application in binary image compression. It has long been known that there are close connections between compression performance and the ability to generalize to new (unseen) data (Rissanen, 1989). The cost of compression, measured in bits, is usually divided into three separate terms: 1) The (constant) cost of coding the model parameters, 2) the cost of coding the states of the hidden variables of the model (linear in D) and 3) the cost of coding the residual errors (linear in D). Optimal model complexity is achieved by trading off the first term against the latter two. It was shown in Hinton & Zemel (1994) that by choosing the states of the hidden variables stochastically according to its posterior $p(\mathbf{z}|\mathcal{D}, \boldsymbol{\theta})$ (where $\mathcal{D}$ denotes the dataset and $\boldsymbol{\theta}$ denotes the parameters) one will get back a refund equal to the entropy of this posterior distribution[1]. This bits-back trick will sometimes be used in the following, although for some models, such as the RBM, it will turn out to be intractable.

This encoding scheme corresponds to MAP estimation where in addition to the usual log-likelihood terms certain regularizer terms (corresponding to log-priors for the parameters) are present. A full Bayesian treatment would require choosing parameters stochastically from its posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ and receiving another refund equal to the entropy of this posterior distribution. However, to claim your refund, you need to be able to compute the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ after all the data have been transmitted. In the case of MRF models such as the RBM this is intractable and therefore unrealistic. For this reason we will omit

[1]Suboptimal "variational" distributions can also be used as a substitute for the posterior.

this bits-back term. Instead, we will encode the model parameters up to a certain precision (or quantization level), Δ, assuming a Gaussian distribution for the encoding. The cost of this encoding is therefore equal to $-\log \Delta - \log \mathcal{N}(\theta_i)$ for every parameter. The value of Δ is chosen such that if we add independent uniform quantization noise in the range $[-\Delta/2, \Delta/2]$ to all the parameters, then the contribution of these perturbations constitute less than 1% of the total compression rate.

3.3.4.1 Compression with vector quantization

The idea of VQ is to divide the data vector space into K partitions, and represent all the points in a partition by a common vector a.k.a. codeword $\mathbf{c}_k$. If a data point $\mathbf{x}_n$ is inside the kth partition, we compress it by storing only the index k and the error $\mathbf{x} - \mathbf{c}_k$. Given a probability distribution of a signal s, the minimal coding length we can achieve is the negative log-likelihood. Moreover, we can get arbitrarily close to that value with entropy encoding on a large enough dataset. Hence, we will treat $-\log(p(s))$ as the true coding length.

For the binary image compression task in this section, we assume that the prior for data to be in partition k is π_k and we model the error at each pixel as an independent Bernoulli random variable with probability $p_{0k} = P(x_{i,j} \neq c_{jk}), \forall j$. Also, we compress the parameters $\{\mathbf{c}_k\}$ as Bernoulli distributed random variables and $\{p_{0k}, \pi_k\}$ as Normal distributed random variables with a quantization level Δ. The total coding length is thus

$$L = L_{param} + L_{code} \quad \text{with} \quad L_{code} = \sum_{i=1}^{D} [-\log \pi_{z_i} - \log \mathcal{B}(\mathbb{I}(\mathbf{x}_i \neq \mathbf{c}_k); p_{0z_i})] \tag{3.25}$$

where L_{param} is the total coding length for all the parameters and z_i is the index of the code for i^{th} data case.

According to the bits-back argument (Hinton & Zemel, 1994), we can potentially encode more cheaply by stochastically picking z_i from some distribution $q(z|\mathbf{x}_i)$ and claiming a number of bits back after transmission equal to the entropy of this distribution $\mathcal{H}(q)$. Note, however, that by sub-optimally picking the index we also incur a higher error. The optimal encoding is achieved if we use the posterior for q, i.e.

$$q(z|\mathbf{x}_i) \propto e^{-L_{code}(\mathbf{c}_k, \mathbf{x}_i)} \tag{3.26}$$

With the bits-back scheme, we can show that the coding length becomes

$$L_{code} = \sum_{i=1}^{D} \left[-\log \sum_{k=1}^{K} \pi_k \mathcal{B}(\mathbb{I}(\mathbf{x}_n \neq \mathbf{c}_k); p_{0k}) \right] \tag{3.27}$$

which is equivalent to the negative log-likelihood of a mixture of Bernoulli distributions (MoB) with the variables $\{z_i\}$ marginalized out.

3.3.4.2 Compression with Herding

Recall that herding, instead of defining a model explicitly, defines a model only *implicitly* by generating a sequence of samples. As such, it is not immediately evident how to use it to compress a data collection. Our approach will be to run herding for a very long time (e.g. $T = 10^7$ iterations) starting at some randomly initialized values for the weights, $\mathbf{w}$, at time $t = 0$ and using the learned regression functions $\{r_{jk}, r_k\}$. In this way we will generate a new codebook vector $\mathbf{c}_t$ at every iteration given as the visible part, $\mathbf{x}_t^*$, of the sample $\mathbf{s}_t^*$ at iteration t. Note, that the receiver can also generate these codebooks, so they do not have to be transmitted.

The initial values, $\mathbf{w}_0$, are sampled from a Normal distribution and communicated by sending a random seed and a scale factor (the standard deviation) so that they can be replicated

at the receiver's end who uses the same random number generator. The time indices of the samples now serve as our hidden variables in VQ and they will be encoded under a uniform prior (a.k.a. all time indices are equally likely). We pick a time index for each data-case ($\tau_i,\ \forall i$) according to its posterior distribution which allows us to receive a modest refund equal to the entropy of this posterior (which the receiver needs to compute after all data has been transmitted). Incorporating this bits-back argument, we can compute the coding length L_{code} as:

$$L_{code} = \sum_{i=1}^{D} \left[-\log \frac{1}{T} \sum_{t=1}^{T} \mathcal{B}(\mathbb{I}(\mathbf{x}_i \neq \mathbf{c}_t); p_0) \right] \tag{3.28}$$

where T is the total number of iterations. Note that this equals a Bernoulli mixture model with a number of components equal to the length of herding sequence.

We encode the model parameters $A, \boldsymbol{\pi}, b$ using Normal distributions. However, since the prototypes very closely follow a mixture of two Gaussians model we use that for its encoding (see Figure 3.1). The residual prediction errors are encoded using a Bernoulli distribution with a single probability p_0 of making an error.

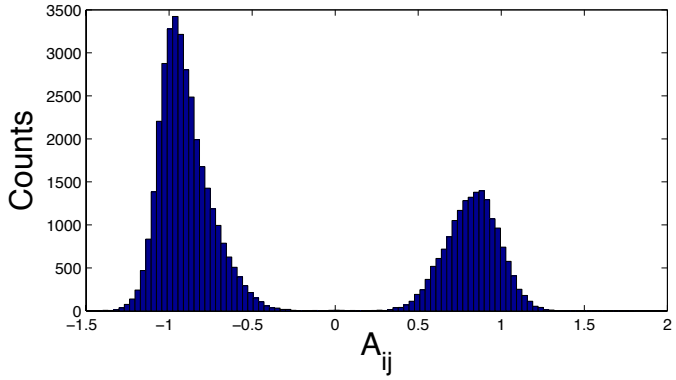


Figure 3.1: Histogram of the elements of prototypes $\mathbf{A}$, trained on the USPS digit image dataset

3.3.4.3 Compression with RBM

A natural question is how herding compares in terms of compression performance with its associated energy based model (RBM). We have tested RBMs on the described compression task in two distinct ways. The first method (RBM-S) is similar to the strategy employed for herding but uses Gibbs sampling to generate a large codebook given by these samples. Although sampling is random and thus unrepeatable in principle, this can be circumvented on a digital computer using a pseudo-random number generator where again the sender and receiver have to agree on the seed value (and a scale factor to sample the initial values from a Normal distribution).

The second way to compress with RBMs (RBM-H) is to treat it as an auto-encoder, that is, to map the data-vector to the latent state space of the RBM and transmit that latent state ($\mathbf{z}$) along with the reconstruction error. During decoding, the signal is recovered as $\mathbf{x} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) + error$. The optimal latent state is searched locally through hill climbing for a minimal reconstruction error starting from the state $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x})$. Both the latent states and errors are encoded as Bernoulli random variables with probabilities $p_k = p(z_k = 1)$, $q_j = p(error_j = 1)$[2].

3.3.5 Experiments

We report compression results on the USPS Handwritten Digits dataset[3] which consists of 1100 examples of each digit 0 through 9 (totally 11,000 examples). Each image has 256 pixels and each pixel has a value within $[0, 255]$ which we turned into a binary representation through the mapping $X'_j = 2\mathbb{I}[X_j > 50] - 1$. Each digit class was randomly split into 700

[2]It should be noted that for RBMs it is intractable to apply the bits-back argument. The reason is that one needs the marginal distribution $p(\mathbf{z})$ to encode the hidden units stochastically and if one uses a variational distribution instead, it is intractable to compute the variational posterior distribution necessary to reclaim the redundant bits.

[3]Downloaded from `http://www.cs.toronto.edu/~roweis/data.html`

training and 400 test examples.

3.3.5.1 Regression Function Training

We train the regression functions on the USPS digit image dataset as follows. A MoB model is first trained by the regular EM algorithm to provide sensible initializations for the prototypes. The training is then divided into two phases. We first herd the parameters of an RBM using the data and feed the pairs $\{\mathbf{w}, \bar{\phi}(\mathbf{w})\}$ to an online gradient descent algorithm to train $\mathbf{r}(\mathbf{w})$. The resulting regression function has a small SSE in the area of high probability under the distribution $\hat{p}(\mathbf{w})$. However, when we replace the data by the prototypes (regression function), the distribution of the samples, $\hat{p}^*(\mathbf{w})$, is different from, but similar to, $\hat{p}(\mathbf{w})$. We have found it beneficial to further fine-tune the regression functions by collecting pairs $\{\mathbf{w}, \bar{\phi}(\mathbf{w})\}$ from herding using the latest estimates of the regression functions instead of the data. In this second phase, we are actually attempting to minimize $\langle \text{SSE} \rangle_{\hat{p}^*(\mathbf{w})}$. Figure 3.2 shows the average SSE over 10^4 iterations when we run herding on images of digit "8" with $\mathbf{r}(\mathbf{w})$. The plot with fine tuned regression functions has lower and more stable SSE than those trained only by phase 1. Figure 3.3 shows some examples of learnt prototypes.

Next, we evaluate the estimated models in terms of L_{code} (see Eqn. 3.28) by comparing the compression performance of herding, with various numbers of prototypes, against that of herding with data. The coding lengths are plotted in Figure 3.4a as a function of iteration. We observe that running herding for a longer time will improve the compression performance because it becomes increasingly likely that a sample will be generated to resemble the data-case we wish to encode. However, there is a price to be paid, namely the fact that we need to spend more bits to encode the time index of the sample. It is interesting that even after 1M iterations the curve is still descending.

We further note that we have omitted the cost of encoding the parameters in this plot, which

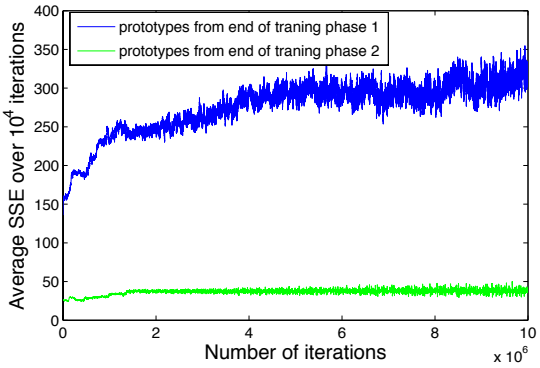


Figure 3.2: Average SSE of the regression functions over 10^4 iterations with 20 prototypes for an RBM with 100 hidden units, trained on 700 images. $\mathbf{r}(\mathbf{w})$ are trained through only phase 1 (top) or both phases (bottom).

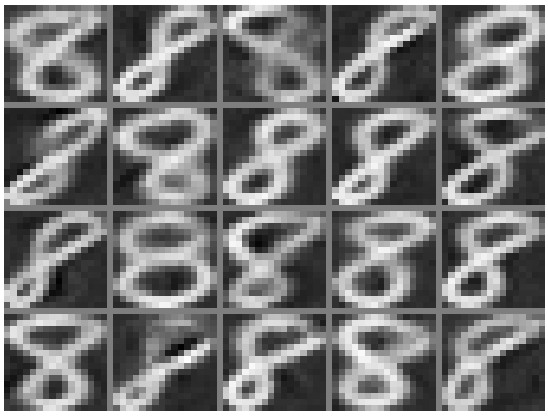
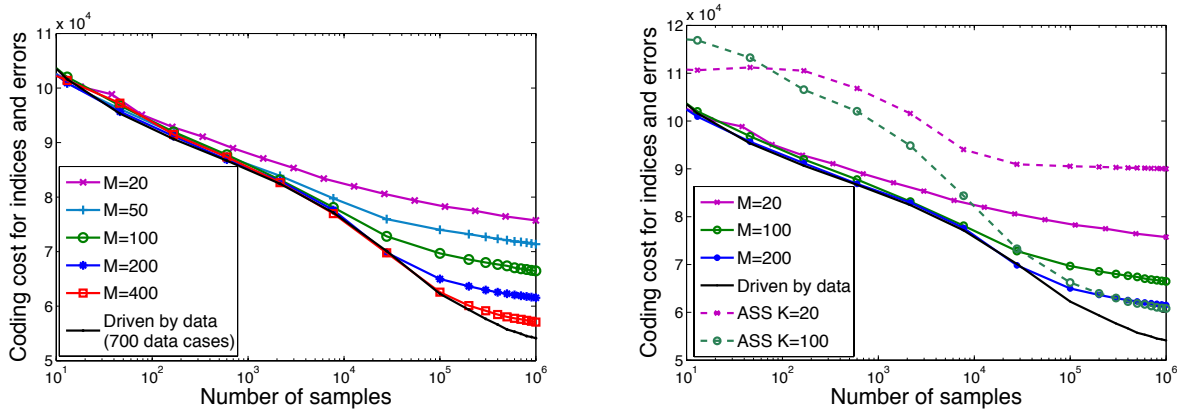


Figure 3.3: Examples of tuned prototypes on images of digit "8".

explains why herding with data (Welling, 2009b) has the lowest coding cost[4]. However, one should keep in mind that the latter method is not a realistic method to compress the data (since it requires the data itself to run).

We also consider the first approach to parametrizing herding in Section 3.3.3. The rates are estimated by running herding with data and averaging the values of $\bar{\phi}_t$ over time: $\mathbf{r} = \frac{1}{T}\sum_{t=1}^{T}\bar{\phi}(\mathbf{w}_t)$. We refer this method as "ASS" as those rates play the role of average sufficient statistics of a model. In Figure 3.4b, we compare the coding cost of herding driven by a regression function (100 hidden units and various numbers of prototypes), data and ASS. With 100 hidden units in the RBM, we need regression functions with about 200 prototypes to achieve similar performance to the ASS method. However, one can decrease the model complexity (save bits for encoding parameters) by reducing the number of prototypes without changing the structure of RBM. This is not true for herding with ASS because one has to reduce the number of hidden units in order to save bits for parameters which severely affects

[4]This was done to easily assess how accurately parametric herding approximates the original "nonparametric" herding that requires all the data to run.



(a) Comparison between parametric herding driven by various numbers of prototypes.

(b) Comparison between herding driven by a regression function (with M prototypes), data itself, and ASS (with K hidden units).

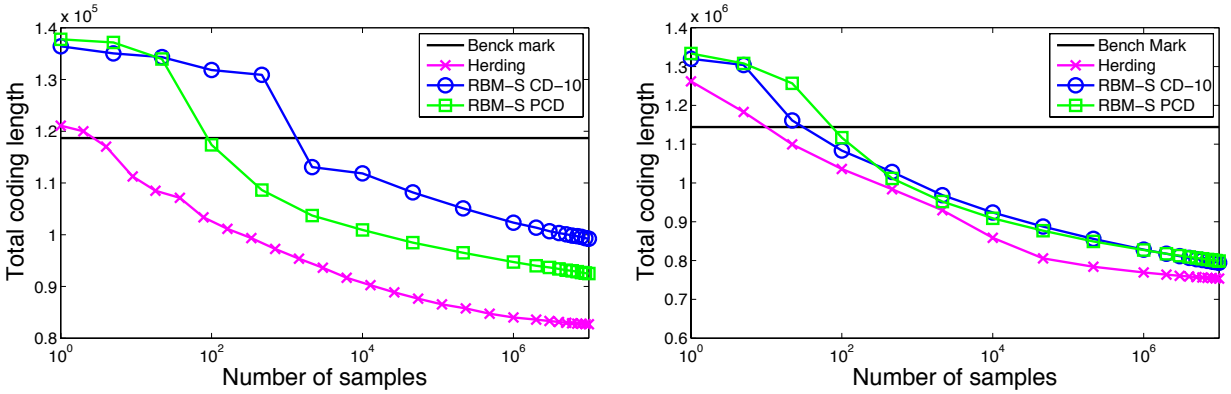
Figure 3.4: L_{code} for images of digit "8" by herding on an RBM with 100 hidden units.

the performance. Around the optimal model complexity ($M/K = 20$) for the subset of digits "8" the coding cost of ASS is much higher than that of our proposed regression function.

3.3.5.2 Compression Performance

The various algorithms are compared in terms of their total coding length on the 16×16 USPS digit image dataset. We use a small set of images from the digit "8" and a larger set including all the digits. The benchmark is to treat every pixel as an independent Bernoulli random variable with a probability p. The average coding length of each pixel is then $\mathcal{H}(p)$, where the minimum is achieved when p equals the percentage of binary pixels with value 1.

RBMs are trained using contrastive divergence with 10 steps (CD-10) (Hinton, 2002) or using persistent CD (PCD) (Tieleman, 2008). Parameters such as the number of codewords for VQ, the number of hidden units for RBM and herding, as well as the number of prototypes for herding are found by a linear or grid search for a minimal coding length. The samples from RBM-S are subsampled at a rate of 1 : 100 from their Gibbs sampling sequence to reduce the autocorrelation (there is no extra coding cost involved for this but encoding



(a) Digit "8". (b) All digits.

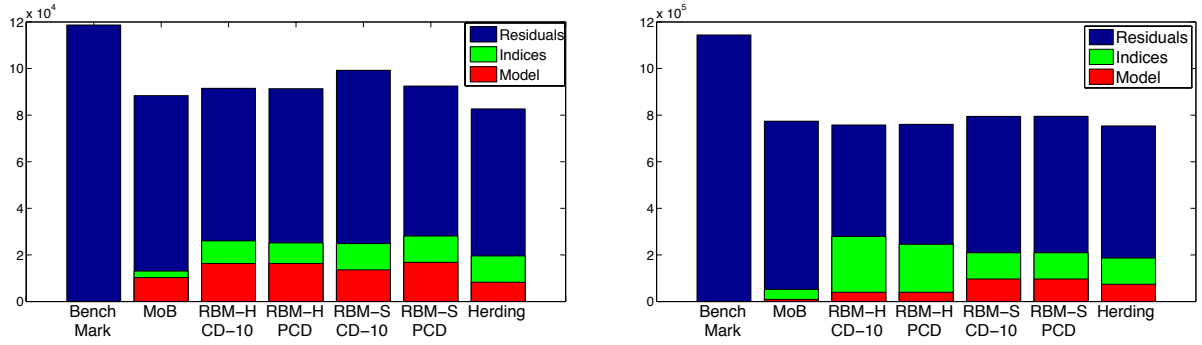
Figure 3.5: Total coding length against the number of samples for herding (x-mark) and RBM-S CD-10(circle)/PCD(square). The black line is the bench mark.

and decoding will be more time consuming). Parameter quantization is set by the method described in section 3.3.4.

Figure 3.5 shows the decrease of the total coding length with the number of samples for herding and RBM-S on the dataset of digit "8" and on the full set. As the number of samples increases, the number of bits for encoding the errors decreases while that for indices increases. Given enough iterations, these plots will most likely start to increase. However, the minima are beyond the scope of our experiments. We find that herding always achieves smaller coding length than RBMs with either training methods.

Figures 3.6 shows the coding length of various compression methods in three parts: model complexity, bits for indices/codes and the errors. All the methods achieve about 60 $\sim$ 70% compression ratio on digit "8" and 65 $\sim$ 70% on the whole dataset compared to the benchmark. Although the differences are small, herding does seem to have the shortest coding length.

The gap in coding length between herding and RBM methods, even at the first iteration in figures 3.5a and 3.5b can be traced back to the fact that we can more cheaply encode the prototypes shown in figure 3.3 using a mixture of two Gaussians (see figure 3.1) than we

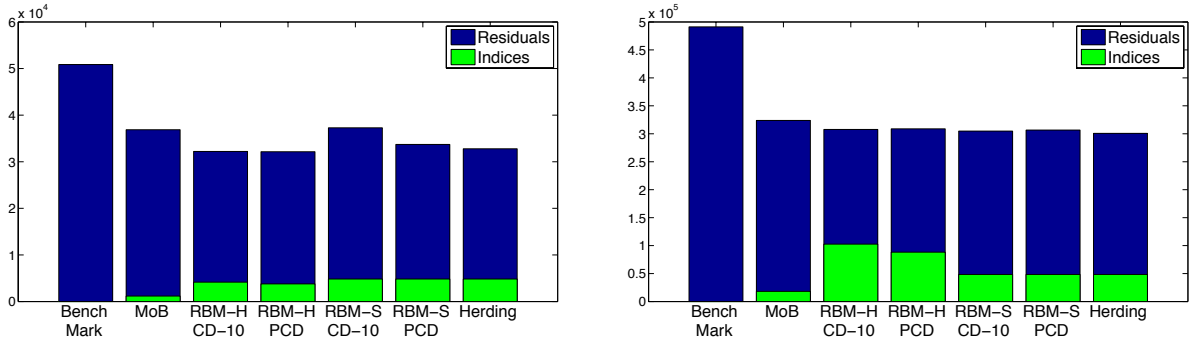


(a) Digit "8". (b) All digits.

Figure 3.6: Method comparison on total coding length.

encode the weights of the RBM. In fact, the RBM weights were much more sensitive to their quantization level Δ. To render the contribution of the quantization level negligible (i.e. less than 1% of the total coding cost), we had to encode the RBM parameters to a higher precision than the parameters for herding (i.e. the prototypes). If we discard the cost for encoding parameters and only compare models with an equal number of parameters then the remaining coding lengths L_{code} are comparable.

Finally, we looked at compression performance on a test set. Suppose that we want to compress a data stream of *i.i.d.* signals. A model can be trained with the first a few samples, and then applied to the remaining data. According to the MDL principle, the model with the minimal total coding length on the training data should also be optimal for unseen test data. Hence, we compare these models with their optimal choice of parameters in terms of MDL. We tested on 300 (unseen) images for each digit. The numbers of bits for encoding time indices and errors, L_{code}, are shown in figure 3.7 for images of a single digit "8" and the full set. Note that L_{code} is equal to the log-likelihood of the test data (which the usual criterion for prediction performance) and that it doesn't make much sense to include the cost of encoding the parameters in this case. Again, herding is performing well on both datasets. On digit "8", Its test coding length is shorter than MoB and RBM-S while marginally longer than RBM-H. On the full set, it is slightly better than all the other



(a) Digit "8". (b) All digits.

Figure 3.7: Coding length for test images.

algorithms.

3.4 Herding Discriminative Models

3.4.1 Conditional Herding

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 3.1, the learning rule in Equation 3.1 can be reformulated in herding style:

$$y_{i_t}^* = \arg\max_{y \in \{-1,1\}} \mathbf{w}_{t-1}^T (\mathbf{x}_{i_t} y) \tag{3.29}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{x}_{i_t} y_{i_t} - \mathbf{x}_{i_t} y_{i_t}^* \tag{3.30}$$

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 be-

tween 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 generates y^* with the sequence of weights $\mathbf{w}_t$. The voted perceptron algorithm (Freund & 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:

$$\mathbf{z}'_{it} = \arg\max_{\mathbf{z}_i} \mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i), \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D} \tag{3.31}$$

$$(\mathbf{y}^*_{it}, \mathbf{z}^*_{it}) = \arg\max_{(\mathbf{y}_i, \mathbf{z}_i)} \mathbf{w}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i), \forall \mathbf{x}_i \in \mathcal{D}_{\mathbf{x}} \tag{3.32}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \left[\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}'_{it})\right] - \left[\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}^*_{it}, \mathbf{z}^*_{it})\right] \tag{3.33}$$

$$(\mathbf{y}^*_{\text{test},j,t}, \mathbf{z}^*_{\text{test},j,t}) = \arg\max_{(\mathbf{y}_j, \mathbf{z}_j)} \mathbf{w}_t^T \phi(\mathbf{x}_{\text{test},j}, \mathbf{y}_j, \mathbf{z}_j), \forall \mathbf{x}_{\text{test},j} \in \mathcal{D}_{\text{test}} \tag{3.34}$$

In the positive term of Equation 3.33, 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, $\mathbf{z}^*_{\text{test},j,t}$, 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:

$$P(\mathbf{y}|\mathbf{x}_{\text{test},j}) \approx \frac{1}{\tau}\sum_{t=1}^{\tau}\mathbb{I}(\mathbf{y}^*_{\text{test},j,t} = \mathbf{y}) \tag{3.35}$$

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 remedy this, one can either normalize features or add one additional feature[5] $\boldsymbol{\phi}_0(\mathbf{x},\mathbf{y},\mathbf{z}) = \sqrt{R^2_{\max} - ||\boldsymbol{\phi}(\mathbf{x},\mathbf{y},\mathbf{z})||^2}$, 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 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 is 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 3.32 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 & Schapire, 1999) which stops herding when convergence occurs and uses the sequence of

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

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 **3.3.** *There exists a constant R such that conditional herding with the update equations 3.31-3.33 satisfies*

$$\left| \frac{1}{D} \sum_{i=1}^{D} \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{x}_i, \mathbf{y}_{it}^*, \mathbf{z}_{it}^*) - \frac{1}{D} \sum_{i=1}^{D} \frac{1}{\tau} \sum_{t=1}^{\tau} \phi_\alpha(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_{it}') \right| \leq \frac{2R}{\tau}, \forall \alpha \tag{3.36}$$

The proof is straightforward by applying PCT where we identify

$$\mathbf{v}_t = \left[\frac{1}{D} \sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_{it}') \right] - \left[\frac{1}{D} \sum_{i=1}^{D} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_{it}^*, \mathbf{z}_{it}^*) \right], \tag{3.37}$$

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

3.4.2 Zero Temperature Limit of CRF

We know that herding on MRFs can be understood as searching for the MLE in the zero temperature limit. Analogously, we can interpret conditional herding as searching for MLE on conditional random fields in the zero temperature limit.

Consider a CRF with the probability distribution defined as

$$P(\mathbf{y}, \mathbf{z} | \mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp\left(\sum_\alpha w_\alpha \phi_\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \right) \tag{3.38}$$

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

$$\ell(\mathbf{w}) = \frac{1}{D} \sum_{i=1}^{D} \left(\log \left(\sum_{\mathbf{z}_i} \exp \left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) \right) \right) - \log Z(\mathbf{w}, \mathbf{x}_i) \right) \tag{3.39}$$

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

$$\ell_0(\mathbf{w}) = \frac{1}{D} \sum_{i=1}^{D} \left(\max_{\mathbf{z}_i} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) - \max_{\mathbf{y}_i, \mathbf{z}_i} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) \right) \tag{3.40}$$

Running gradient descent updates on $\ell_0(\mathbf{w})$ immediately gives us the update equations of conditional herding 3.31-3.33.

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:

$$\frac{1}{D} \sum_{i=1}^{D} \mathbb{E}_{P(\mathbf{z}|\mathbf{x}_i, \mathbf{y}_i)} \left[\phi_\alpha(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}) \right] = \frac{1}{D} \sum_{i=1}^{D} \mathbb{E}_{P(\mathbf{y}, \mathbf{z}|\mathbf{x}_i)} \left[\phi_\alpha(\mathbf{x}_i, \mathbf{y}, \mathbf{z}) \right], \forall \alpha \tag{3.41}$$

When we run conditional herding, those constraints are satisfied with the moment matching property in Proposition 3.3, but how to encourage high entropy during the herding dynamics is still an open problem. We suggest some heuristics to achieve high entropy in the next experimental subsection. Nevertheless, the negative auto-correlation in the weight sequence allows us to learn the conditional model more efficiently than the ML learning procedure especially in the presence of hidden variables. Another difference lies in the test phase. 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

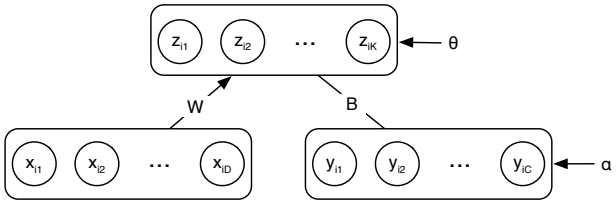


Figure 3.8: Discriminative Restricted Boltzmann Machine model of distribution $p(\mathbf{y}, \mathbf{z}|\mathbf{x})$.

provides more robust performance as shown later.

3.4.3 Experiments

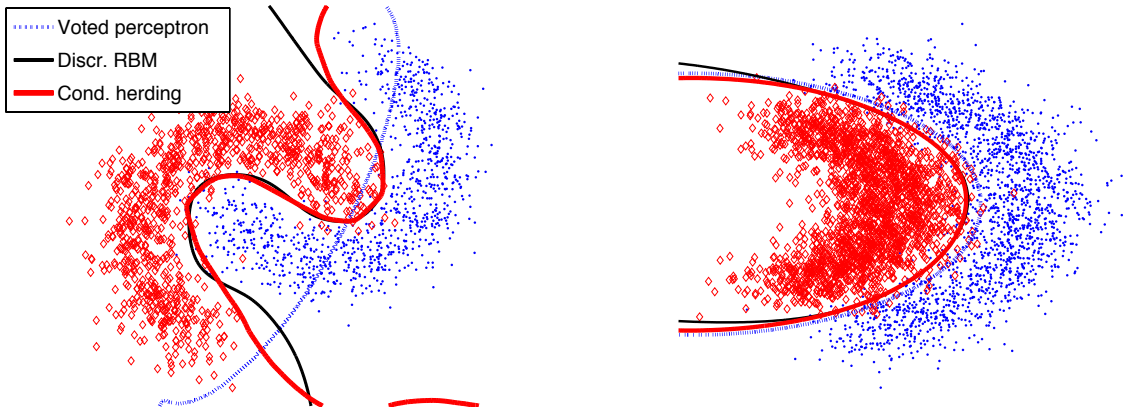
We studied the behavior of conditional herding on two artificial and four real-world data sets, comparing its performance to that of the voted perceptron (Freund & Schapire, 1999) and that of discriminative RBMs (Larochelle & Bengio, 2008). The experiments on artificial and real-world data are discussed separately in Section 3.4.3.1 and 3.4.3.2.

We studied conditional herding in the discriminative RBM (dRBM) architecture illustrated in Figure 3.8, that is, we use the following parameterization

$$\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x}^T \mathbf{W} \mathbf{z} + \mathbf{y}^T \mathbf{B} \mathbf{z} + \boldsymbol{\theta}^T \mathbf{z} + \boldsymbol{\alpha}^T \mathbf{y}. \tag{3.42}$$

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 3.4.1, 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.



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

Figure 3.9: Decision boundaries of VP, CH, and dRBMs on two artificial data sets.

3.4.3.1 Artificial Data

To investigate the characteristics of voted perceptron (VP), discriminative RBM (dRBM) and conditional herding (CH), we used the techniques discussed above 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 3.9. 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[6].

[6]On the Lithuanian data set, VP constructs a good boundary by exploiting the added 'normalizing'

This happens because VP can regress on only $2 + 1 = 3$ fixed features. In contrast, for CH the simple predictor in the top layer can regress onto $M = 20$ hidden features. This prevents the same oscillatory behavior from occurring.

3.4.3.2 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 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 Eqn. 3.32 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

feature.

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 Eqn. 3.32, 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 $\mathbf{w}_t^T \mathbf{v}_t \leq 0$ [7]. 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 linesearches) 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 & Bengio (2008), we use L2-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 [8].

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 Eqn. 3.42 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

[7] Local maxima can also be found by iterating over $y_{\text{test}}^{*,k}, z_{\text{test},j}^{*,k}$, but the proposed procedure is more efficient.

[8] 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.

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 during herding. The relative scale between $\boldsymbol{\eta}$ and σ was chosen by cross-validation. Recall that the absolute scale is unimportant (see Section 3.4.1 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 $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\eta}{D_t}\sum_{i_t}(1-\lambda)\langle \mathbf{z}_{i_t}\rangle - \mathbf{z}^*_{i_t}$, 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 3.1. 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 & 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

One-Versus-All Procedure

Technique / *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

Technique / *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 3.1: 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.

error for CH remains level. This phenomenon is the benefit of averaging over many different predictors.

3.5 Inconsistent Moments

We have described herding as a nonlinear dynamical system to match the moments of a training data set. PCT combined with Proposition 2.1 or 2.2 guarantees that the average features of the samples will converge to the input moments as along as we obtain a good local maximum at every iteration. However, when we consider more generally an input moment vector that is not necessarily an average over a set of data cases or any distribution, we may have a set of *inconsistent* moments in the following sense:

DEFINITION **3.1.** *A vector $\bar{\boldsymbol{\phi}} \in \mathbb{R}^D$ is said to be inconsistent with respect to a set of states*

$\mathcal{X}$ with a feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^D$ if there does not exist a probability distribution over $\mathcal{X}$, P, such that

$$\bar{\boldsymbol{\phi}} = \mathbb{E}_{\mathbf{x}\sim P}[\boldsymbol{\phi}(\mathbf{x})]$$

Herding is not able to match the input moments in that case. In this section, we first illustrate the benefit of being capable to handle inconsistent moments with an example in computer vision, and then show theoretically that herding corrects the inconsistency by searching for the closest consistent moment vector to the input in the sense of L_2 distance.

3.5.1 Piecewise Trained CRF in Computer Vision

The CRF model has been a standard approach to combine local features into a global conditional distribution over labels in computer vision. For instance, CRFs have been used extensively in image segmentation and labeling (Blake et al., 2004; Kumar & Hebert, 2006; He et al., 2004), as well as object recognition (Quattoni et al., 2007), image denoising and image restoration (Tappen et al., 2007), and stereo (Pal et al., 2010). One first defines potential functions $\phi_\alpha(\mathbf{x}_\alpha, \mathbf{y}_\alpha)$ where $\mathbf{y}_\alpha$ is a subset of the labels associated with potential ϕ_α and $\mathbf{x}_\alpha$ are input variables. The label subsets indexed by α are assumed to overlap and form a loopy graph over the labels $\mathbf{y}$. For instance, the subsets could correspond to all pixels and all pairs of pixels in an image. These local potentials are combined in a CRF which specifies a joint probability distribution over all labels by

$$P_{\text{crf-1}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp\left[\sum_\alpha w_\alpha \phi_\alpha(\mathbf{x}_\alpha, \mathbf{y}_\alpha)\right] \tag{3.43}$$

$\{w_\alpha\}$ are model parameters, one associated with each potential function, that are typically learned from data. As mentioned in section 3.4.2 maximum likelihood training of such a model with respect to some dataset with empirical distribution $\hat{P}(\mathbf{y}|\mathbf{x})$ has the intuitive

property that expectations of the potential functions under the model match those of the training data

$$\mathbb{E}_{P_{\text{crf-1}}}[\phi_\alpha] = \mathbb{E}_{\hat{P}}[\phi_\alpha] \tag{3.44}$$

While this is quite elegant, it poses a practical problem that each potential function ϕ_α has a distinct parameter w_α to be estimated. In typical image labeling problems, there may be thousands of weights to learn (e.g. one for every pixel and pair of pixels for every image). In other words, there is less than one pixel of information per parameter, leading to extreme over-fitting.

To avoid this explosion of parameters, a typical approach is to share parameters across potential functions. For instance, if we have pixel-wise and pairwise potential functions we could use a single parameter λ to trade off their relative importance.

One of the main questions we wish to address here is how to most effectively use the information of local discriminative classifiers $p_\alpha(\mathbf{y}_\alpha|\mathbf{x}_\alpha)$ whose parameters are trained on all the pixels of an image or a training set of images. In the CRF approach one can incorporate these piecewise trained local classifiers by taking the log probabilities as local potential functions so that $\phi_\alpha(\mathbf{x}_\alpha, \mathbf{y}_\alpha) = \log(p_\alpha(\mathbf{y}_\alpha|\mathbf{x}_\alpha))$. For example, Fulkerson et al. (2010) use the following CRF model for segmentation

$$P_{\text{crf-2}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\exp\left[-E(\mathbf{y}|\mathbf{x})\right] \tag{3.45}$$

$$E(\mathbf{y}|\mathbf{x}) = \sum_i \log(p_i(y_i|x_i)) + \lambda \sum_{i,j} \log(p_{i,j}(y_i \neq y_j|x_i, x_j)) \tag{3.46}$$

Unfortunately, in such a locally trained model, there is no longer reason to expect that the model matches the training data in the previous sense that $\mathbb{E}_{P_{\text{crf-2}}}[\phi_\alpha] = \mathbb{E}_{\hat{P}}[\phi_\alpha]$ since we have only one parameter to tune and a very large collection of these moment constraints (number

of pixels plus number of neighboring pairs of pixels).

Instead, we might like to impose that our global model at least still approximately matches the constraints,

$$\mathbb{E}_P[\phi_\alpha] = \mathbb{E}_{p_\alpha}[\phi_\alpha] \tag{3.47}$$

where $\hat{P}$ has been replaced with p_α. For features of the form $\phi_{\alpha,\mathbf{c}}(\mathbf{y}_\alpha) = \mathbb{I}[\mathbf{y}_\alpha = \mathbf{c}]$, this condition implies that the joint distribution marginalizes down to local distributions

$$\sum_{\mathbf{y}\setminus\mathbf{y}_\alpha} P_{\text{crf-2}}(\mathbf{y}|\mathbf{x}) = p_\alpha(\mathbf{y}_\alpha|\mathbf{x}_\alpha) \tag{3.48}$$

The herding algorithm 2.6, 2.7 provides a solution to matching the local marginals by generating a sequence of label samples $\{\mathbf{y}_t\}$ using the features $\{\phi_{\alpha,\mathbf{c}}(\mathbf{y}_\alpha)\}$. The resulting distribution represented by the sampled sequence would satisfy the condition in 3.48 as long as the input moment vector, i.e. the local marginals, is consistent and the PCT is satisfied. Moreover, the convergence rate is optimally fast given that we approximate the probabilities with Monte Carlo averages, and in particular much faster than the typical $\mathcal{O}(1/\sqrt{T})$ convergence for stochastically generated averages. Not having an explicit model is not a problem for the applications we have in mind. For instance, in image segmentation the dynamical system will generate a sequence of segmentations of the input image. From this sequence we can extract the final segmentation by averaging.

However, with independently trained local classifiers, local marginals are likely to be inconsistent, in which case condition 3.48 cannot be satisfied for any joint distribution. It hence becomes necessary to study the behavior of the herding algorithm with inconsistent marginals, or generally inconsistent moments.

3.5.2 Convergence Analysis with Inconsistent Moments

Let us consider a general setting of herding with features $\{\phi_\alpha\}$ on state variable $\mathbf{x} \in \mathcal{X}$ and inconsistent input moments $\bar{\boldsymbol{\phi}}$ (i.e. $\{\mathbb{E}_{p_\alpha}[\phi_\alpha(\mathbf{x}_\alpha, \mathbf{y}_\alpha)]\}$ in the example of the previous section). Define the marginal polytope as the convex hull of the feature vectors of all the possible states:

$$\mathcal{M} \stackrel{\text{def}}{=} \text{conv}\{\boldsymbol{\phi}(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}. \tag{3.49}$$

One can show that the input moment vector $\bar{\boldsymbol{\phi}}$ is inconsistent if and only if it does not reside on $\mathcal{M}$. Given inconsistent moments, if we want to train an MRF using gradient descent without regularization, the parameters will diverge. For vanilla herding defined in Equation 2.6-2.7, it means that the PCT condition cannot always be satisfied as illustrated in Figure 3.10 and the norm of parameters $\mathbf{w}_t$ will also diverge at a linear rate. Nevertheless, we can still obtain a stationary distribution of states $\mathbf{x}_t$ from the herding sequence. The potential numerical problems caused by the divergence of $\mathbf{w}_t$ can be easily prevented by taking an additional normalization step $\mathbf{w} \leftarrow \mathbf{w}/K$ and $\eta \leftarrow \eta/K$ for some K. This global scaling will not affect the state sequence $\mathbf{s}_t$ in any way. The most important consequence of inconsistent moments is that the moments of the sampling distribution do not converge to $\bar{\boldsymbol{\phi}}$ any more. Instead, we prove in the following paragraphs that the moments orthogonally project onto the marginal polytope.

Denote the sampling average of the features generated by herding up to time T as $\tilde{\boldsymbol{\phi}}^T = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\phi}(\mathbf{s}_t)$. We now claim that the following property holds:

PROPOSITION **3.4.** *Assume $\bar{\boldsymbol{\phi}}$ is outside the marginal polytope $\mathcal{M}$ and the learning rate η_α is constant. Let $\bar{\boldsymbol{\phi}}_{\mathcal{M}}$ be the L_2 projection of $\bar{\boldsymbol{\phi}}$ on $\mathcal{M}$. Then the average feature vector of the herding dynamics $\tilde{\boldsymbol{\phi}}^T$ converges to $\bar{\boldsymbol{\phi}}_{\mathcal{M}}$ at the rate of $\mathcal{O}(1/T)$.*

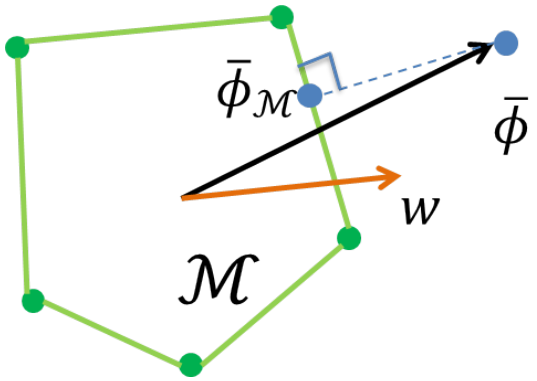


Figure 3.10: An example of inconsistent moments. Each green point represents the feature vector of a state. The PCT condition is not satisfied when $\mathbf{w}$ is in the shown direction. The feature vector averaged over samples converges to the projection of $\bar{\phi}$ on $\mathcal{M}$.

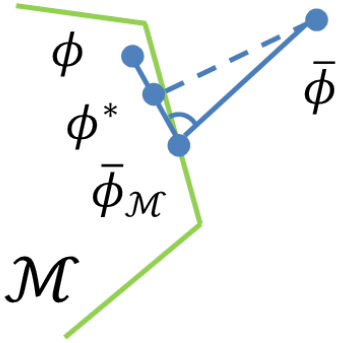


Figure 3.11: The assumption of an acute angle $\angle\phi\bar{\phi}_{\mathcal{M}}\bar{\phi}$.

Proof. Since herding is scale free with respect to the learning rate, we can assume $\eta_\alpha = 1$ without loss of generality. We first construct another sequence of weights

$$\tilde{\mathbf{w}}_0 = \mathbf{w}_0, \quad \tilde{\mathbf{w}}_t = \tilde{\mathbf{w}}_{t-1} + \bar{\phi}_{\mathcal{M}} - \phi(\mathbf{s}_t) \tag{3.50}$$

where $\mathbf{s}_t$ is drawn by the regular herding dynamics following Equation 2.6 and 2.7. Then to prove the proposition, it suffices to prove that the new sequence $\{\tilde{\mathbf{w}}_t\}$ satisfies PCT, and consequently $\|\tilde{\mathbf{w}}_t\|$ is bounded.

Let us give the following lemma before moving on

LEMMA **3.5.** $(\bar{\phi} - \bar{\phi}_{\mathcal{M}})^T(\phi - \bar{\phi}_{\mathcal{M}}) \leq 0, \quad \forall \phi \in \mathcal{M}$

Proof. Assume there exists a point $\phi \in \mathcal{M}$ *s.t.* the inequality does not hold, then the angle $\angle\phi\bar{\phi}_{\mathcal{M}}\bar{\phi}$ is an acute angle, and hence as shown in figure 3.11 we can always find a point on the segment connecting ϕ and $\bar{\phi}_{\mathcal{M}}$, ϕ^*, such that $\|\phi^* - \bar{\phi}\|_2 < \|\bar{\phi}_{\mathcal{M}} - \bar{\phi}\|_2$. This contradicts with the fact that ϕ^* is in $\mathcal{M}$ and $\phi_{\mathcal{M}}$ is the projection of $\bar{\phi}$. □

According to the definition of $\mathbf{s}_t$ in Equation 2.6, $\mathbf{w}_T^T\boldsymbol{\phi}(\mathbf{s}_{T+1}) \geq \mathbf{w}_T^T\boldsymbol{\phi}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$. Since $\bar{\boldsymbol{\phi}}_{\mathcal{M}} \in \mathcal{M}$, it can be represented by a convex combination of features $\{\boldsymbol{\phi}(\mathbf{x})\}$ with a set of coefficients $\{\pi_i\}$, i.e. $\bar{\boldsymbol{\phi}}_{\mathcal{M}} = \sum_i \pi_i \boldsymbol{\phi}(\mathbf{x}_i), \pi_i > 0, \forall i$, and $\sum_i \pi_i = 1$. Then we know that

$$\mathbf{w}_T^T\bar{\boldsymbol{\phi}}_{\mathcal{M}} = \sum_i \pi_i \mathbf{w}_T^T\boldsymbol{\phi}(\mathbf{x}_i) \leq \mathbf{w}_T^T\boldsymbol{\phi}(\mathbf{s}_{T+1}) \tag{3.51}$$

Also, comparing the update equation of $\tilde{\mathbf{w}}$ in Equation 3.50 and that of $\mathbf{w}$ in Equation 2.7, one can observe that

$$\tilde{\mathbf{w}}_T = \mathbf{w}_T - T(\bar{\boldsymbol{\phi}} - \bar{\boldsymbol{\phi}}_{\mathcal{M}}) \tag{3.52}$$

Combined with Lemma 3.5 and inequality 3.51 we get

$$\tilde{\mathbf{w}}_T^T(\boldsymbol{\phi}(\mathbf{s}_{T+1}) - \bar{\boldsymbol{\phi}}_{\mathcal{M}}) = \mathbf{w}_T^T(\boldsymbol{\phi}(\mathbf{s}_{T+1}) - \bar{\boldsymbol{\phi}}_{\mathcal{M}}) - T(\bar{\boldsymbol{\phi}} - \bar{\boldsymbol{\phi}}_{\mathcal{M}})^T(\boldsymbol{\phi}(\mathbf{s}_{T+1}) - \bar{\boldsymbol{\phi}}_{\mathcal{M}}) \geq 0 \tag{3.53}$$

which shows that the sequence $\{\tilde{\mathbf{w}}_t\}$ satisfies PCT by identifying $\mathbf{v}_t = \boldsymbol{\phi}(\mathbf{s}_{T+1}) - \bar{\boldsymbol{\phi}}_{\mathcal{M}}$. This concludes the proof. □

Following Equation 3.52 we can verify that $\mathbf{w}_t$ will diverge linearly since $\tilde{\mathbf{w}}_T$ is bounded and $\bar{\boldsymbol{\phi}} - \bar{\boldsymbol{\phi}}_{\mathcal{M}}$ is constant. As another immediate consequence of Proposition 3.4, herding always improves the initial set of moments $\bar{\phi}$, which drives herding dynamics through equations 2.6 and 2.7, in the following sense:

COROLLARY **3.6.** *Given a feature vector $\bar{\boldsymbol{\phi}}$ that is an approximation to the expected feature w.r.t. some unknown distribution, $\bar{\boldsymbol{\phi}}_{true}$. When running herding dynamics with $\bar{\boldsymbol{\phi}}$, the limit of the empirical average of features will not increase the L_2 error. Specifically, $\|\bar{\boldsymbol{\phi}}_{\mathcal{M}} - \bar{\boldsymbol{\phi}}_{true}\|_2 < \|\bar{\boldsymbol{\phi}} - \bar{\boldsymbol{\phi}}_{true}\|_2$ when $\bar{\boldsymbol{\phi}} \notin \mathcal{M}$, and $\tilde{\boldsymbol{\phi}}^T \to \bar{\boldsymbol{\phi}}$ otherwise.*

The proof is trivial. Since any true expectation $\bar{\phi}_{true}$ must be inside the marginal polytope, according to Lemma 3.5, $\angle\boldsymbol{\phi}_{true}\bar{\boldsymbol{\phi}}_{\mathcal{M}}\bar{\boldsymbol{\phi}}$ is an obtuse angle which leads to Corollary 3.6.

As discussed in section 2.3.2, changing the value of the constant scalar learning rate η does not change the generated sequence of states. However, when the moment is inconsistent and we use a set of constant learning rates η_α where η_α depends on the feature index α, then the limiting average feature $\bar{\boldsymbol{\phi}}_{\mathcal{M}}$ would depend on the relative values of η_α. To see this, we can construct an equivalent herding sequence with a fixed learning rate $\eta = 1$ and new features $\{\sqrt{\eta_\alpha}\phi_\alpha\}$. Then Proposition 3.4 still applies except that the L_2 distance is weighted by $\sqrt{\eta_\alpha}$. So the learning rates control the relative importance of features.

Back to the image segmentation example, when we consider $\boldsymbol{\phi}$ as $\phi_{\alpha,\mathbf{c}}(\mathbf{y}_\alpha) = \mathbb{I}[\mathbf{y}_\alpha = \mathbf{c}]$, then the marginal probabilities of herding samples will converge to the closest consistent marginals in $\mathcal{M}$.

3.5.3 Application: Image Segmentation

Inconsistent marginals are common in image segmentation. Conditional probabilities of groups of variables can come from various sources such as color, texture, context, etc. We consider an example of two types of possibly inconsistent probabilities in this thesis: unary conditional probabilities $\{p_i(y_i|x_i)\}$ on super-pixels and pairwise conditional probabilities $\{p_{ij}(y_i \neq y_j|x_i, x_j)\}$ on neighboring super-pixels. The former provides local class distributions, and the latter suggests the existence of boundaries.

The CRF approach uses the energy defined in Equation 3.46 with a single parameter λ. The best assignment with lowest energy is inferred as the segmentation output, and the best value of λ is estimated on a validation set using grid search.

Our herding algorithm follows equations 2.6 and 2.7 with two types of features: $\mathbb{I}(y_i = c)$ and $\mathbb{I}(y_i \neq y_j)$. The learning rate η_α is scale free in the sense that multiplying all η_α by the same factor does not change the output of label sequence $\mathbf{y}_t$. So without loss of generality

we may set the learning rates for unary features to 1, and those for pairwise features as λ. The value of λ is used to trade off the strength of these two sources of information. The final segmentation for herding is obtained by maximization and averaging,

$$y_i^* = \arg\max_c \sum_t \mathbb{I}[y_{it} = c] \quad \forall i \tag{3.54}$$

Notice that the role of the parameter λ is different between the CRF and herding approaches. In the CRF, λ controls the strength of smoothness. Increasing λ always increases smoothness. However, herding tries the respect all the probabilities, and λ measures how much attention we pay to each of these two sources of information. Increasing λ not only increases the smoothness where $p_{ij}(y_i \neq y_j)$ is small, but also forces an edge where $p_{ij}(y_i \neq y_j)$ is large. As a special case, for a system of N super-pixels with $\lambda \gg 1$ and $p_{ij}(y_i \neq y_j) = 0$ for all neighbors in the label graph, the pairwise term dominates the system, and all super-pixels will take on the same value.

We apply herding to image segmentation on two datasets, the PASCAL VOC 2007 segmentation competition and GrabCut to illustrate its effectiveness. It is compared to the multiclass classifier with local appearance cues only, and to the traditional CRF approach.

3.5.3.1 PASCAL VOC 2007

On the PASCAL VOC 2007 dataset, we follow a similar experimental setting as that of Fulkerson et al. (2010) and perform segmentation on the level of super-pixels. Each image is first over-segmented by the global probability of boundary (gPb) method (Arbelaez et al., 2011). The threshold is set to 0 to make sure most boundaries are retained. SIFT features are then extracted and quantized in order to build a visual dictionary. A local multiclass SVM is trained to provide unary marginals $p_i(y_i|x_i)$ using histograms of the visual words in

each super-pixel and its neighbors at a distance of at most N. The larger N is, the more context information is available for the local classifier, the less noise in the feature histogram but also the more blurred the boundaries between super-pixels become. By increasing N, the segmentations of the local classifier changes from inaccurate and noisy but with clear sharp boundaries to more accurate and smooth but with blurred boundaries (see the results of the local method of $N = 0$ in Figure 3.14 and $N = 3$ in Figure 3.12). The gPb algorithm provides the probability of a boundary between two super-pixels, i.e. the pairwise marginals $p_{ij}(y_i \neq y_j|\mathbf{x})$. The VOC test set includes 210 images, and the "trainval" set is split randomly into a training set of 322 images and a validation set of 100 images. The local classifier is trained on the training set, and the (hyper-)parameters of the CRF and herding are estimated on the validation set.

For the local models, we predict the super-pixel labels based on the output of SVMs. For the CRF models, the MAP label is inferred using the graphcut algorithm from Fulkerson et al. (2010) (see also Boykov & Kolmogorov, 2004; Boykov et al., 2001; Kolmogorov & Zabih, 2004) with an energy as in equations 3.46. The parameter λ is estimated by grid search on the validation set. For the herding method, the maximization step in Eqn. 2.6 is also executed using the graphcut. Because the original gPb score is trained on the BSDS dataset and a lot of boundaries belonging to irrelevant categories of objects in the VOC dataset are not considered, gPb should be calibrated first. The calibrated pairwise probability is computed as $P_{VOC}(y_i \neq y_j|\mathbf{x}) = P_{BSDS}(y_i \neq y_j|\mathbf{x})^{\alpha}$, where α controls how sparse the boundaries in the VOC dataset are. The parameters λ and α are estimated on the validation set by first fixing $\alpha = 1$, estimating λ by grid search and then fixing λ and estimating α. More iterations can be done for better performance. Notice that for CRF, the function of λ and α appears in the same position in the pairwise term $\lambda\alpha \log(P(y_i \neq y_j|x_i, x_j))\mathbb{I}(y_i \neq y_j)$, and a second parameter is therefore redundant.

Figure 3.12 shows some examples of the test images, results of different algorithms as well as

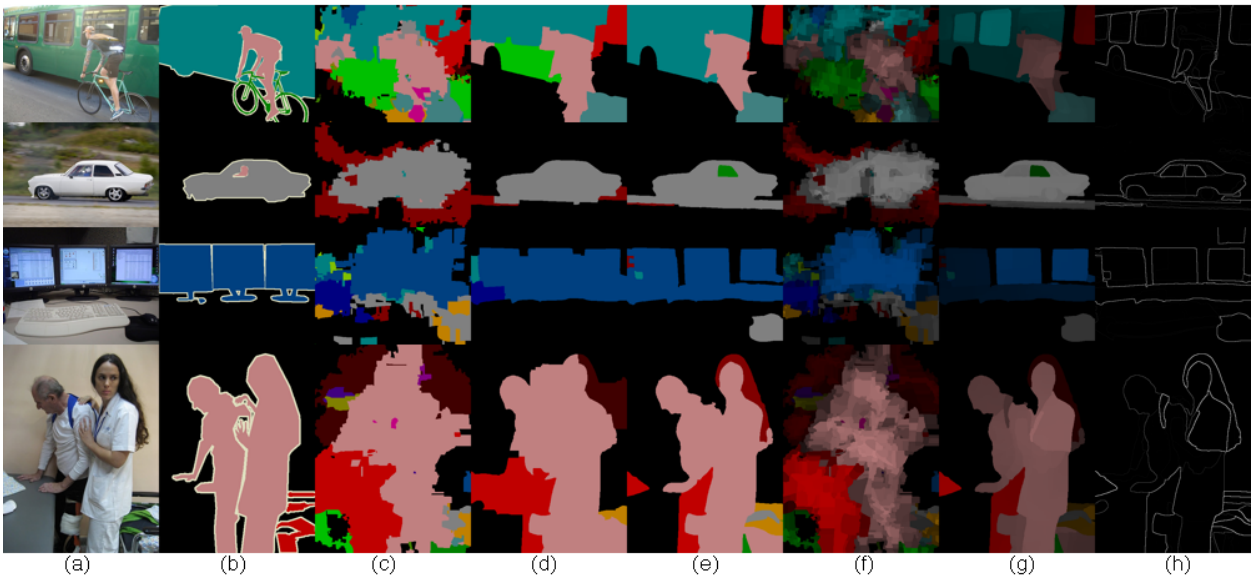


Figure 3.12: Examples of segmentation on Pascal VOC 2007 data set. Images on each line starting from left to right are respectively: (a) the original image, (b) ground truth segmentation, results of (c) local classifier, (d) CRF and (e) herding, results with intensity proportional to the posterior probability of the (f) local classifier and (g) herding, and (h) the herding estimate of the pairwise probability of the existence of a boundary (the corresponding posterior probability for CRF cannot be easily obtained). Neighboring superpixels of a distance up to 3 hops are used for training local SVM. Best viewed in color.

their posterior probabilities. The local classifiers are trained on features from a neighborhood of $N = 3$. So the unary class distribution is already smoothed to some extent (compared to figure 3.14 for the case of N=0). But herding still leads to better smoothness and locates the boundaries more accurately. Most boundaries occur in the place with strong pairwise probabilities. CRF provides similar benefits as herding for regularizing the local classifiers.

We evaluate the performance of these three models by two measurements. The first one is the average accuracy adopted by the VOC 2007 Competition. It measures the average recall of pixels for each category. The second measurement is the one adopted by the VOC competition after 2007. It measures the average of the intersection over union ratio for each category. The results of both evaluation methods are shown in figure 3.13. The results show that both herding and CRF increase the accuracy in most cases, and herding always achieves the best accuracy except for $N = 2$ by the second measurement. The reduction

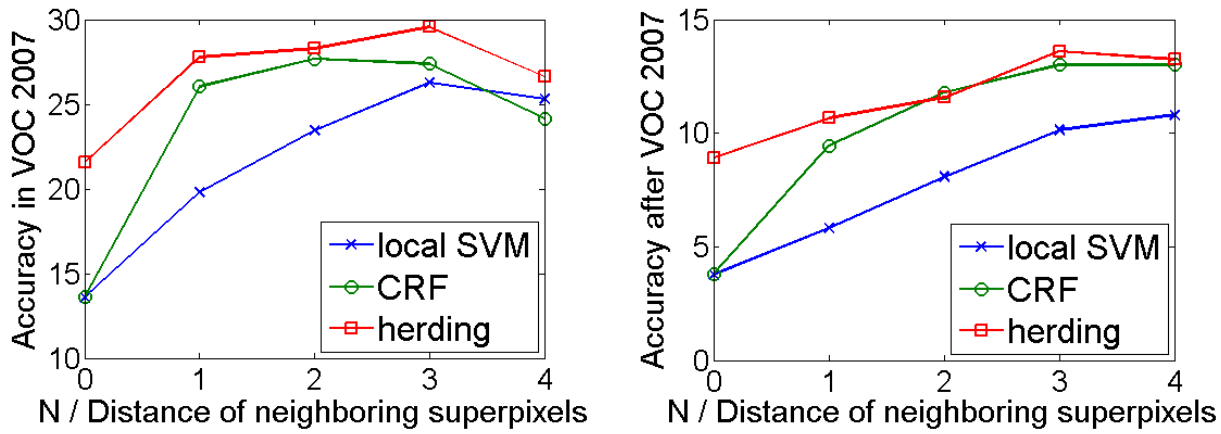


Figure 3.13: Average accuracy of segmentations by the local SVM classifier (cross), CRF (circle) and herding (square) with different number of neighboring superpixels used for extracting local features. N denotes the maximal distance of the neighboring superpixel used. The left plot uses the 2007 segmentation benchmark criteria (average recall). The plot on the right uses the 2010 criteria on the 2007 dataset (average overlap).

		background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor	Average
Recall	Local	46	7	15	**10**	8	**10**	31	51	34	**17**	6	**16**	41	23	58	50	18	21	**15**	36	**39**	26
	CRF	56	**20**	12	2	**15**	7	33	52	**59**	8	**10**	8	31	20	68	**55**	12	15	**15**	**49**	36	28
	Herd	**62**	3	**16**	3	7	7	**38**	**58**	50	15	2	11	**58**	**24**	**70**	54	**20**	**23**	14	47	**39**	**30**
Overlap	Local	50	**2**	6	**8**	2	0	13	21	14	2	**2**	4	8	10	24	20	6	**8**	**5**	12	10	11
	CRF	**65**	1	**8**	0	2	0	15	**30**	**17**	3	0	4	5	10	**37**	**24**	**9**	**8**	**5**	**18**	**13**	13
	Herd	60	**2**	4	4	**3**	**5**	**23**	28	15	**4**	0	**5**	**20**	**12**	31	22	6	**8**	3	**18**	12	**14**

Table 3.2: Accuracies per category and the average accuracy of PASCAL VOC 2007 dataset. Each model uses the N value that maximizes the average test accuracy. Top table shows recall (PASCAL 2007 benchmark) the bottom table shows overlap (PASCAL 2010 benchmark)

of the advantage of herding compared to CRF in the second measurement may be due to the fact that false positive detections appear frequently in the background which does not reduce the recall of the background category by much, but will reduce the intersection over union ratio of the detected category.

Remarkably, herding performs much better than the local method when $N = 0$. The accuracy is improved from 14% to 22% on the first measurement and 4% to 9% on the second

measurement, while CRF does not help at all. The local classifier performs poorly because the histogram feature is computed from very few pixels as discussed in Fulkerson et al. (2010). Thus regularization on the pairwise term should improve the prediction. It turns out that the optimal value of λ for herding is about 1.1×10^3 which means the importance of the pairwise feature is $\sqrt{\lambda} \approx 33$ times higher than the unary feature, matching our expectation. On the other hand, the best value for CRF is only about 1.1. The difference in the choice of λ leads to the significant difference in the segmentations as shown with a typical example in Figure 3.14. Herding outputs a highly smoothed result with clear boundaries while the CRF does not noticeably change the decision of the local classifier.

Two properties of herding previously stated in section 3.5.3 would help explain the distinct choices of λ. Firstly, with a large λ, herding tries to average the distribution of superpixels in a smooth area. Although the local SVMs give very noisy results, the average distributions still contain strong signals about the true category. In contrast, the CRF computes the product of distributions which makes the noise in the final distribution even worse. So CRF has to choose a small λ. To verify this hypothesis, we trained a CRF with energy as a linear function of features $P(y_i|x_i)$ and $P(y_i \neq y_j|x_i, x_j)$, that also computes the average of distributions when λ is large. The new CRF chooses a large λ (≈ 22) as expected and the accuracy is improved to 16% and 7% respectively. However Figure 3.14 shows that the result is oversmoothed because of the high penalty of boundaries. Secondly, herding not only increases smoothness in flat areas but also encourages boundaries at strong edges. That is why herding still captures the shape of the object correctly even with a large value of λ.

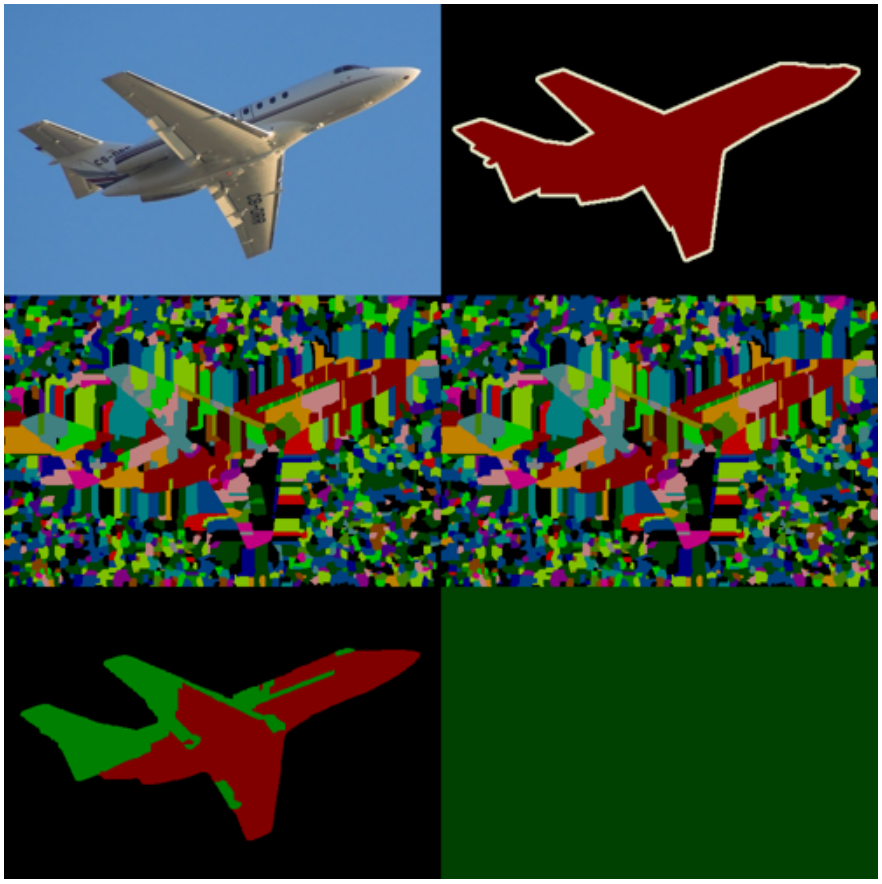


Figure 3.14: A typical example of segmentations when $N = 0$. The top 2 images are the original image and the ground truth segmentation. The remaining 4 images are respectively the segmentation of the local model, CRF, herding and a CRF with linear potential functions. The local model is so noisy because the histogram of SIFT features is computed from very few pixels.

3.5.3.2 GrabCut[9]

We also ran herding on the GrabCut data set, which consists of 50 images of a foreground object on a natural background[10]. The objective on the GrabCut images is to perform hard foreground/background segmentation.

The GrabCut data set contains two labeled trimaps for each image, where a trimap is a labeling of pixels as foreground, background or undetermined (see Figure 3.15). The 'lasso' trimap contains a rough segmentation of the image obtained using a lasso or pen tool.

[9] This work was done by Andrew Gelfand and is described in Chen et al. (2011b).

[10] `http://www.research.microsoft.com/vision/cambridge/segmentation/`

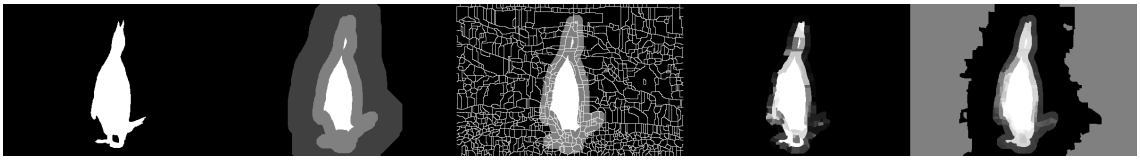


Figure 3.15: Example of segmentation on GrabCut data set (Chen et al., 2011b). Images from left to right are the 'expert' ground truth trimap, 'lasso' trimap, over segmentation into superpixels, unary marginal probabilities from local classifiers and unary marginal probabilities after herding. In the trimaps, black pixels are background, white pixels are foreground and light gray pixels are undetermined. In the last two images, whiter values indicate larger $P(y_i = 1|\mathbf{x})$.

The 'expert' trimap is a ground truth labeling obtained by tracing the boundary of the foreground image at the pixel level. Performance on the GrabCut data set is assessed via the segmentation error rate (SER), which is the number of misclassified pixels in the set of undetermined pixels (where the set of undetermined pixels does not include the undetermined pixels in the expert trimap).

As with the PASCAL data set we train a binary classifier to provide unary marginals and use the gPb method to provide pairwise marginals. Since we have a trimap for each image and can easily identify pixels that are foreground and background, we train a different binary (foreground/background) classifier for each image based on two color models (denoted as CM1 and CM2 respectively). We compare three different methods on the GrabCut data: 1) Local model; 2) CRF model; and 3) Herding.

The results are shown in Tabel 3.3. The segmentation error rate was computed across the set of undetermined pixels in the test set of images. From these results we see that the addition of pairwise marginals that encode boundary information gives a big gain over the local, independent model. Herding obtains slightly smaller segmentation error than CRF in both local color models. See Chen et al. (2011b) for details.

Segmentation Error Rate (SER)					
Local Model		**CRF**		**Herding**	
CM 1	CM 2	CM 1	CM 2	CM 1	CM 2
10.45%	10.46%	6.35%	6.58%	**6.28%**	**6.29%**

Table 3.3: Segmentation error rate for local model, CRF and herding on the GrabCut data set.

3.5.4 Application: Predicting the Game Go

Coming back to the example in the introduction, we want to predict the final territory of each player in a Go game based on the current state of an incomplete game. We denote by a vector $\mathbf{C} = \{Black, White, Empty\}^N$ the stones of an incomplete game with $N = 19 \times 19$ being the number of positions, and denote by a vector $\mathbf{S} = \{-1, 1\}^N$ the final territory, where $s_n = 1$ (-1) means that the n^{th} position is occupied by the black (white) player. We build up the joint distribution by looking at conditional probabilities of small patches $\mathbb{E}_{p_\alpha}[\mathbf{S}_\alpha = \mathbf{s}_\alpha | \mathbf{C}_\alpha = \mathbf{c}_\alpha]$ where α denotes the position of a small patch and $\mathbf{c}_\alpha$ is its stone colors. Figure 1.1 shows an example of the conditional probability when the patch is a pair of neighboring stones. We also consider larger patches such as a square or a cross shape of five stones.

These conditional probabilities can be estimated trivially using the empirical frequencies in a training set. A uniform prior is applied to reduce the noise in the empirical estimation. Given an incomplete game $\mathbf{c}$, we obtain a set of local predictions of the final territory on each patch $\{\mathbb{E}_{p_\alpha}[\mathbf{S}_\alpha = \mathbf{s}_\alpha | \mathbf{C}_\alpha = \mathbf{c}_\alpha]\}_\alpha$. We can then use them as the input moments of features, $\mathbb{I}[\mathbf{S}_\alpha = \mathbf{s}_\alpha]$, and run herding to produce samples of the final territory of the whole board, $\{\mathbf{s}_t\}$. The average of these samples are treated as our prediction. Since the conditional probabilities are computed independently at each location, we have again the problem of inconsistent moments. From the analysis in Section 3.5.2, the joint distribution of $\mathbf{S}$ from herding will minimize its L_2 distance from the input local conditional distributions.

When the conditioning state vector $\mathbf{c}_\alpha$ is all *Empty*, a local classifier cannot tell which player is more influential in this area. Therefore, we combine two local predictions with inverse patterns as a single input moment in this case $\mathbb{E}_{p_\alpha}[\mathbf{S}_\alpha = \mathbf{s}_\alpha \text{ or } \mathbf{S}_\alpha = \neg\mathbf{s}_\alpha | \mathbf{C}_\alpha \text{ is all } \mathit{Empty}]$, which allows the herding algorithm to assign all the probability to either $\mathbf{s}_\alpha$ or $\neg\mathbf{s}_\alpha$ when neighboring patterns have a strong preference to one player, instead of imposing a constraint to assign equal probabilities both states.

Stern et al. (2004) propose a CRF model with features defined on patches of single stones, $\phi_i(s_i) = s_i$, and neighboring stone pairs, $\phi(s_i, s_j) = s_i s_j$. Their corresponding model parameters are functions of $\mathbf{C}$: $h_i(c_i)$ and $w_{i,j}(c_i, c_j)$. The MLE of the parameters is learned on a training set with two inference methods: a generalization of Swendsen-Wang sampler (Swendsen & Wang, 1987) and loopy belief propagation (BP) (Murphy et al., 1999). As the difference of the predictive performance between those two methods is not significant with respect to cross entropy as shown in Figure 4 of Stern et al. (2004) and the algorithm with the Swendsen-Wang sampler is much slower than loopy BP, we compare herding with the CRF model trained and tested with loopy BP only in this thesis.

We train and test our predictive models on a data set of 25089 games by professional players[11]. We prune the games that are not complete, and use the GNU GO program[12] to obtain the final territory of the remaining games. The data set is then split into a training set of 5993 games and a test set of 2595 games. We follow the practice in Stern et al. (2004) to train herding and CRF (compute the conditional probabilities for herding and learn MLE for CRF) at three stages: 20, 80, and 150 moves, and evaluate the predictive performance at these respective stages on the test set using the metric of cross entropy between actual

[11]Provided by `http://gokifu.com`.
[12]Downloaded from `http://www.gnu.org/software/gnugo`.

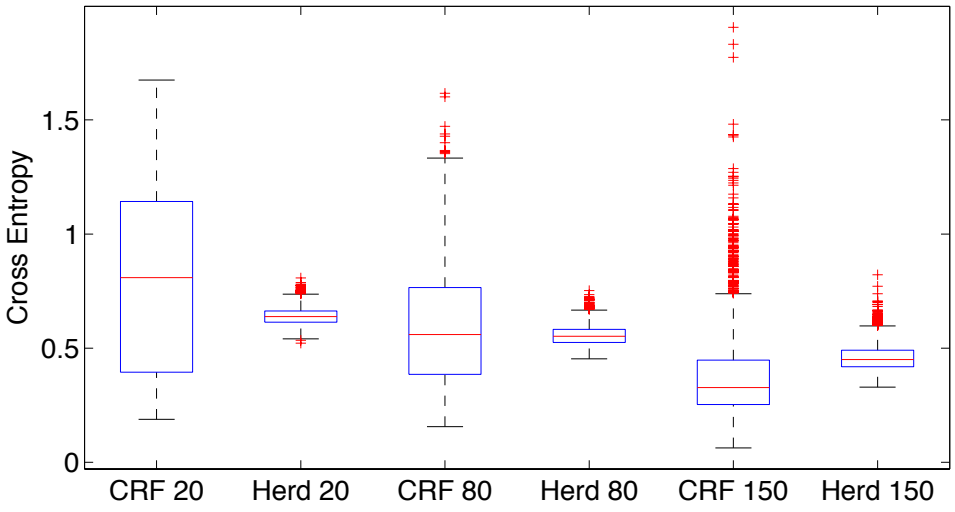


Figure 3.16: Cross entropy of final territory prediction on the test set by CRF and herding at Move 20, 80, and 150.

territory outcomes, $\mathbf{s}$, and predicted territory outcomes, $\mathbf{s}'$:

$$\mathcal{H} = \frac{1}{N} \sum_{n=1}^{N} [s'_n \log s_n + (1 - s'_n) \log(1 - s_n)] \tag{3.55}$$

Figure 3.16 shows the cross entropy on the test set by CRF and herding. The mean of the cross entropy for both CRF and herding as well as the variance for CRF decrease at later stages of games (more moves). This is reasonable since when a game is close to complete, it becomes easier to predict the final territory. Compared to CRF, herding has smaller variance across different games throughout all the stages. It also achieves better average prediction than the CRF at an early stage (Move 20) with both lower mean and smaller variance. But as the game progresses, the advantage diminishes, and it is eventually outperformed by the CRF at a later stage (Move 80).

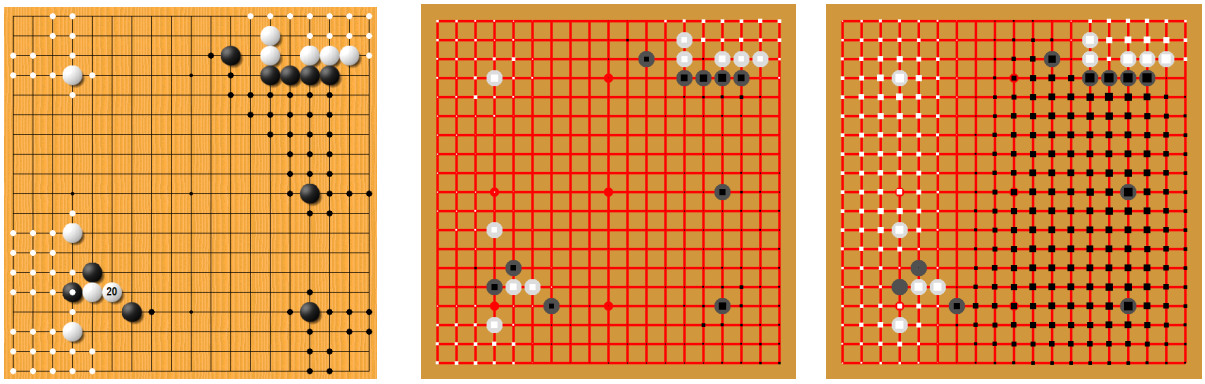
We compare the predictions of these two methods with a rule based commercial software, "Many Faces of Go," using an exemplary game in Figure 3.17. We can see the apparent difference between the predictions of herding and CRF. Herding tends to make conservative

predictions especially in empty areas such as the middle of the board at Move 20. It is confident only when one player has a clearly stronger influence, for instance, in the upper right corner. In contrast, CRF tends to be overconfident about its predictions as shown in those empty areas at Move 20 and 80. This problem is also mentioned in Stern et al. (2004) where the Swendsen-Wang sampling algorithm achieves better predictions than loopy BP (Figure 2). However, it still appears to be overconfident especially at the early stages when there are few stones nearby. As the game progresses, it becomes increasingly clear whose territory the remaining empty areas belong to. In that case we should be able to make a confident prediction about those areas according to their surrounding stones' color and the CRF method shows superior performance to herding as shown at Move 150. Also, we notice that CRF is capable of detecting captured stones such as the two white stones in the lower middle part at Move 80 and 150 although it often makes false positive mistakes at early stages of a game. In contrast, herding usually has conservative predictions for captured stones.

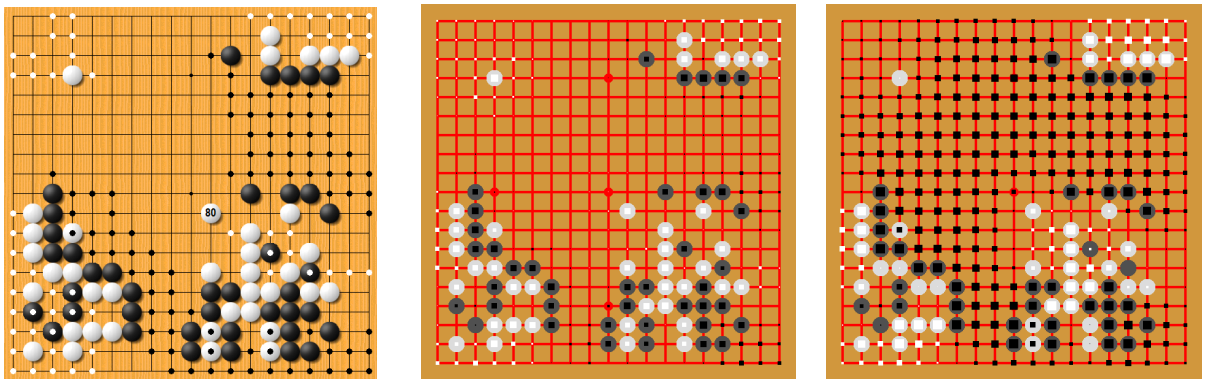
3.6 Summary of Contributions

My contributions to this chapter are summarized as follows:

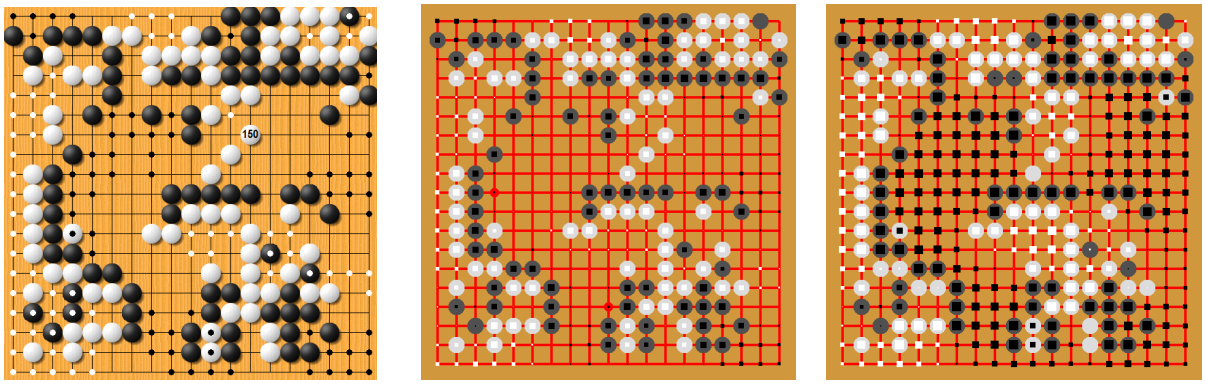
- Proposed the Perceptron Cycling Theorem as a general condition for the moment matching property of herding in Section 3.1.
- Designed the regression function of the parametric herding algorithm in Section 3.3.3 and applied it to data compression.
- Conducted the experiments of conditional herding on real-world data in Section 3.4.3.2 jointly with Andrew Gelfand and Lauren Van Der Maaten.



(a) Many Faces of Go, Move 20 (b) Herding, Move 20 (c) CRF, Move 20



(d) Many Faces of Go, Move 80 (e) Herding, Move 80 (f) CRF, Move 80



(g) Many Faces of Go, Move 150 (h) Herding, Move 150 (i) CRF, Move 150

Figure 3.17: Territory predictions of "Many faces of go," herding and CRF at three stages: Move 20, 80, and 150. For the latter 2 methods, the large circles represent current stones. Small squares represent final territory prediction from -1 (maximum white square) to $+1$ (maximum black square).

- Analyzed the convergence property of the herding algorithm with inconsistent moments in Section 3.5.2.
- Applied herding to image segmentation on the PASCAL VOC 2007 dataset in Section 3.5.3.1 and game prediction in Section 3.5.4.

I would like to acknowledge of contributions from my collaborators:

- Max Welling proposed the idea of generalized herding with hidden variables (Section 3.3.1) and labels (Section 3.4.1). He also suggested training a regression function to parameterize the herding algorithm in Section 3.3.3 and proposed the idea of applying herding to problems with inconsistent moments such as image segmentation and game prediction.
- Charless Fowlkes provided the code of the gPb algorithm in Section 3.5.3.
- Lauren Van Der Maaten conducted the experiments of conditional herding on artificial data in Section 3.4.3.1.
- Andrew Gelfand applied herding to image segmentation on the GrabCut dataset.

Chapter 4

Herding as a Sampling Algorithm

A learning algorithm for a probabilistic model takes as input an empirical distribution and outputs another distribution according to some inductive criterion so that it will generalize to unobserved data. A sampling algorithm can be considered similarly as a process that takes as input a target distribution and output another distribution, which is represented by a sequence of samples. The difference between these two algorithms is that the sampling algorithm requires the output distribution to recover the input exactly. In previous chapters we have been discussing the advantage of using herding to learn a model with a fast convergence rate on a set of features. It would be natural to consider applying herding as a sampling algorithm. The resulting deterministic sampling algorithm enjoys fast convergence on estimating the ergodic averages, which makes it a promising alternative to Monte Carlo methods based on random sampling.

Currently all of the random sampling methods use pseudo-random number generator as the source of randomness, which is essentially deterministic. Tribble (2007) shows that replacing the pseudo-random numbers with a uniformly distributed sequence improves the quality of the Monte Carlo estimator. The idea of directly designing deterministic algorithms for

sampling from distributions has been proposed in various works (Niederreiter, 1992; Tribble, 2007; Holroyd & Propp, 2010; Murray & Elliott, 2012) but the domain of their applications or the condition for ergodicity is still limited. The introduction of "herded sampling" algorithms would potentially make a meaningful contribution in this direction.

Breuleux et al. (2010) made the first attempt on applying herding as a sampling method. They ran the "rates-herding" algorithm, the first parametric herding approach described in Section 3.3.3 with moments computed at MLE, to draw samples from a restricted Boltzmann machine and showed a faster mixing rate in the state space than Gibbs sampling. However, the stationary distribution of "rates-herding" is not guaranteed to match the target distribution and therefore may generate samples that have low probability in the original model.

In this chapter, we introduce two extensions of herding as sampling algorithms, one for discrete state spaces and the other for continuous state spaces. We show the efficiency of herding in estimating the expected value of functions of interest compared to Monte Carlo methods and discuss the conditions under which the sampling distribution converges to the input distribution.

4.1 Herded Gibbs

In Section 2.6.1 we applied herding to draw samples of a binary state neuron model with a one-dimensional feature. It is trivial to observe that given any firing probability π, the distribution of the samples obeys the Bernoulli distribution $\text{Bern}(\pi)$ due to the moment matching property. This application can be extended to draw samples from a discrete state variable. Let X denote a discrete random variable with n states, $X \in \{1, 2, \ldots, n\}$. Denote by $\boldsymbol{\pi}$ a discrete probability distribution with $\pi_i \stackrel{\text{def}}{=} P(X = i) \in [0, 1]$ and $\sum_{i=1}^{n} \pi_i = 1$. Let

us choose a set of n features for X with each feature being an indicator function of one state, $\phi_i(X) = \mathbb{I}[X = i]$. Then the feature vector $\boldsymbol{\phi}(X)$ is actually an 1-of-n encoding of the discrete variable. When we run herding with those features and an input distribution $\boldsymbol{\pi}$, the moment vector equals the discrete distribution $\bar{\boldsymbol{\phi}} = \boldsymbol{\pi}$ and the inner product in Equation 2.6 for i^{th} state is simply w_i. Therefore, the maximization step in Equation 2.6 involves finding the largest component of the weight vector $X^{(t)} = \arg\max_{i \in \{1,2,\ldots,n\}} w_i^{(t-1)}$, and the weight update step in Equation 2.7 adds every component by their corresponding probability π_i and then subtracts the selected component by 1. Following the moment matching property, we can show that the sampling distribution $\widehat{\boldsymbol{\pi}}$ converges to the target distribution $\boldsymbol{\pi}$ as $\mathcal{O}(1/T)$.

Note, however, that the domain of this application is narrow. Importantly, it is not able to sample from unnormalized multivariate probability distributions. This is very limiting because the problem of sampling from unnormalized distributions is at the heart of the field of Bayesian inference and the probabilistic programming approach to artificial intelligence (Lunn et al., 2000; Carbonetto et al., 2005; Milch & Russell, 2006; Goodman et al., 2008). At the same time, despite great progress in Monte Carlo simulation, the celebrated Gibbs sampler continues to be one of the most widely-used algorithms. For example, it is the inference engine behind popular statistics packages (Lunn et al., 2000), several tools for text analysis (Porteous et al., 2008), and Boltzmann machines (Ackley et al., 1985; Hinton & Salakhutdinov, 2006). The popularity of Gibbs stems from its simplicity of implementation and the fact that it is a very generic algorithm. Without any doubt, it would be remarkable if we could design generic deterministic Gibbs samplers with fast (theoretical and empirical) rates of convergence.

An extension of herding was proposed by de Freitas et al. (Fang, 2012; Bornn et al., 2013) to draw samples from *unnormalized* discrete probability distribution following an analogy to the Gibbs sampling algorithm, referred to as *herded Gibbs*. Note that the basic component of Gibbs sampling is to draw a random sample for one variable from a normalized conditional

distribution. By replacing the random sampling component with the aforementioned herding algorithm that samples from a normalized discrete distribution, we obtain a deterministic variant of the popular Gibbs sampling algorithm. Instead of matching the joint distribution, herded Gibbs is designed to match the conditional distribution at every step.

In this thesis, we analyse its convergence properties. We prove that the deterministic Gibbs algorithm converges to the target distribution for two special distributions, a distribution of an empty graphical model and a distribution of a fully-connected graphical model. Particularly, it achieves a convergence rate of $\mathcal{O}(1/T)$ for the latter distribution. Suitable conditions for the convergence on a general distribution are still open for future investigation.

4.1.1 Herded Gibbs Sampling

For a graph of discrete nodes $\mathcal{G} = (V, E)$, where the set of nodes are the random variables $V = \{X_i\}_{i=1}^N$, $X_i \in \mathcal{X}$, let π denote the *target distribution* defined on $\mathcal{G}$.

Gibbs sampling is one of the most popular methods to draw samples from π. Gibbs alternates (either systematically or randomly) the sampling of each variable X_i given $\mathbf{X}_{\mathcal{N}(i)} = \mathbf{x}_{\mathcal{N}(i)}$, where i is the index of the node, and $\mathcal{N}(i)$ denotes the neighbors of node i. That is, Gibbs generates each sample from its full-conditional distribution $p(X_i|\mathbf{x}_{\mathcal{N}(i)})$.

Herded Gibbs replaces the sampling from full-conditionals with herding at the level of the full-conditionals. That is, it alternates a process of matching the full-conditional distributions $p(X_i = x_i|\mathbf{X}_{\mathcal{N}(i)})$. To do this, herded Gibbs defines a set of auxiliary weights $\{w_{i,\mathbf{x}_{\mathcal{N}(i)}}\}$ for any value of $X_i = x_i$ and $\mathbf{X}_{\mathcal{N}(i)} = \mathbf{x}_{\mathcal{N}(i)}$. For ease of presentation, we assume the domain of X_i is binary, $\mathcal{X} = \{0, 1\}$, and we use one weight for every i and assignment to the neighbors $\mathbf{x}_{\mathcal{N}(i)}$. The herded sampling step is therefore implemented with Equation 2.24, 2.25. Herded Gibbs can be trivially generalized to the multivariate setting by employing weight vectors in

Algorithm 4.1 Herded Gibbs Sampling.

Input: T.
Step 1: Set $t = 0$. Initialize $\mathbf{X}^{(0)}$ in the support of π and $w^{(0)}_{i,\mathbf{x}_{\mathcal{N}(i)}}$ in $(\pi(X_i = 1|\mathbf{x}_{\mathcal{N}(i)}) - 1, \pi(X_i = 1|\mathbf{x}_{\mathcal{N}(i)}))$.
for $t = 1 \to T$ **do**
  Step 2: Pick a node i according to some policy. Denote $w = w^{(t-1)}_{i,\mathbf{x}^{(t-1)}_{\mathcal{N}(i)}}$.
  Step 3: If $w > 0$, set $x_i^{(t)} = 1$, otherwise set $x_i^{(t)} = 0$.
  Step 4: Update weight $w^{(t)}_{i,\mathbf{x}^{(t)}_{\mathcal{N}(i)}} = w^{(t-1)}_{i,\mathbf{x}^{(t-1)}_{\mathcal{N}(i)}} + \pi(X_i = 1|\mathbf{x}^{(t-1)}_{\mathcal{N}(i)}) - x_i^{(t)}$
  Step 5: Keep the values of all the other nodes $x_j^{(t)} = x_j^{(t-1)}, \forall j \neq i$ and all the other weights $w^{(t)}_{j,\mathbf{x}_{\mathcal{N}(j)}} = w^{(t-1)}_{j,\mathbf{x}_{\mathcal{N}(j)}}, \forall j \neq i$ or $\mathbf{x}_{\mathcal{N}(j)} \neq \mathbf{x}^{(t-1)}_{\mathcal{N}(i)}$.
end for
Output: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$

$\mathbb{R}^{|\mathcal{X}|}$ instead of scalars and using the sampling procedure described at the beginning of this section.

If the binary variable X_i has four binary neighbors $\mathbf{X}_{\mathcal{N}(i)}$, we must maintain $2^4 = 16$ weight vectors. Only the weight vector corresponding to the current instantiation of the neighbors is updated, as illustrated in Algorithm 4.1. The memory complexity of herded Gibbs is exponential in the maximum node degree. Note that the algorithm is a deterministic Markov process with state $(\mathbf{X}, \mathbf{W})$.

The initialization in step 1 guarantees that $\mathbf{X}^{(t)}$ always remains in the support of π. For a deterministic scan policy in step 2, we take the value of variables $\mathbf{x}^{(tN)}, t \in \mathbb{N}$ as a sample sequence. Throughout the section all experiments employ a fixed variable traversal for sample generation. We call one such traversal of the variables a *sweep*.

As herded Gibbs sampling is a deterministic algorithm, there is no stationary probability distribution of states. Instead, we examine the average of the sample states over time and hypothesize that it converges to the joint distribution, our target distribution, π. To make the treatment precise, we need the following definition (see Figure 4.1 for a visual interpretation):

Definition **4.1.** *For a graph of discrete nodes $\mathcal{G} = (V, E)$, where the set of nodes $V =$*

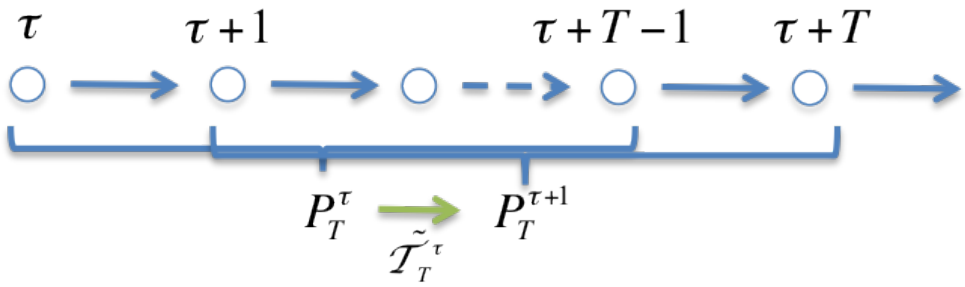


Figure 4.1: Distribution over time at the end of every sweep and the transition kernel of herded Gibbs.

$\{X_i\}_{i=1}^N$, $X_i \in \mathcal{X}$, $P_T^{(\tau)}$ *is the empirical estimate of the joint distribution obtained by averaging over* T *samples acquired from* $\mathcal{G}$. $P_T^{(\tau)}$ *is derived from* T *samples, collected at the end of every sweep over* N *variables, starting from the* τ^{th} *sweep:*

$$P_T^{(\tau)}(\mathbf{X} = \mathbf{x}) = \frac{1}{T} \sum_{k=\tau}^{\tau+T-1} \mathbb{I}(\mathbf{X}^{(kN)} = \mathbf{x}) \tag{4.1}$$

Our goal is to prove that the limiting average sample distribution over time converges to the target distribution π. Specifically, we want to show the following:

$$\lim_{T\to\infty} P_T^{(\tau)}(\mathbf{x}) = \pi(\mathbf{x}), \forall \tau \geq 0 \tag{4.2}$$

If this holds, we also want to know what the convergence rate is in the metric of the total variation distance:

DEFINITION **4.2.** *Let* P *and* Q *be two probabilistic distributions on* $\mathcal{X}$. *The total variation distance between* P *and* Q *is*

$$d_v(P, Q) \stackrel{\text{def}}{=} \sup_{A\in\mathcal{F}} |P(A) - Q(A)| = \frac{1}{2}||P - Q||_1. \tag{4.3}$$

where $\mathcal{F}$ is the σ-algebra of subsets of the sample space $\mathcal{X}$.

In the following two subsections, we will discuss the convergence conditions for two special graph structures, an empty graph and a fully-connected graph.

4.1.2 Convergence on Empty Graphs

We begin the theoretical analysis with a graph of one binary variable. In that case, herded Gibbs reduces to the regular herding algorithm as discussed in Section 2.6.1. Proposition 2.5 and corollary 2.6 show that the invariant set of the dynamics is the interval $(\pi - 1, \pi]$, and the error of the sampling probability converges to the input Bernoulli distribution at a rate of $1/T$, or $1/2T$ if initialized properly.

When w is outside the invariant interval, it is easy to observe that w will move into it monotonically at a linear speed in a transient period. So we will always consider an initialization of $w \in (\pi - 1, \pi]$ in this subsection. Since $w^{(t)}$ is now restricted to the interval of unit length $(\pi - 1, \pi]$, we can take a one-to-one mapping $w \leftarrow w \mod 1$ (we define $1 \mod 1 = 1$) and think of w as updated by a constant translation vector in a circular unit interval $(0, 1]$ as shown in Figure 4.2. That is,

$$w^{(t)} = (w^{(t-1)} + \pi) \mod 1, \quad x^{(t)} = \begin{cases} 1 & \text{if } w^{(t-1)} \leq \pi \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Now let us consider an empty graph with N variables. All the variables are independent of each other and herded Gibbs reduces to running N one-variable chains in parallel. Denote the marginal distribution $\pi_i \stackrel{\text{def}}{=} \pi(X_i = 1)$.

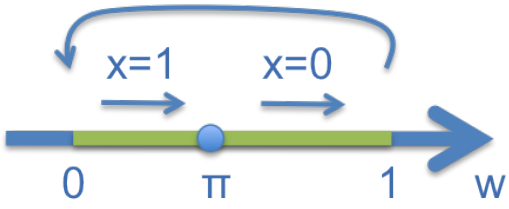


Figure 4.2: Equivalent weight dynamics for a single variable.

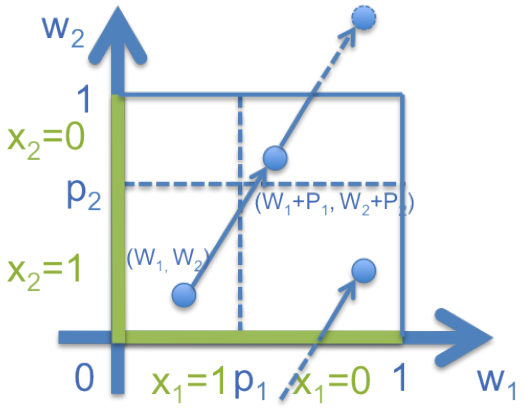


Figure 4.3: Dynamics of herding with two independent variables.

Examples of failing convergence in the presence of rational ratios between the π_is were observed in Bach et al. (2012). There the need for further theoretical research on this matter was pointed out. The following theorem provides formal conditions for convergence in the restricted domain of empty graphs.

THEOREM **4.1.** *For an empty graph, when herded Gibbs has a fixed scanning order, and* $\{1, \pi_1, \ldots, \pi_N\}$ *are rationally independent, the empirical distribution* $P_T^{(\tau)}$ *converges to the target distribution* π *as* $T \to \infty$ *for any* $\tau \geq 0$.

A set of n real numbers, $x_1, x_2, \ldots, x_n$, is said to be rationally independent if for any set of rational numbers, $a_1, a_2, \ldots, a_n$, we have $\sum_{i=1}^{n} a_i x_i = 0 \Leftrightarrow a_i = 0, \forall 1 \leq i \leq n$. The proof of Theorem 4.1 makes use of Kronecker-Weyl's theorem (Weyl, 1916) on uniform distributed sequences.

Proof of Theorem 4.1. For an empty graph of N independent vertices, the dynamics of the weight vector $\mathbf{w}$ are equivalent to a constant translation mapping in an N-dimensional

circular unit space $(0,1]^N$, as shown in Figure 4.3:

$$\begin{aligned}
\mathbf{w}^{(t)} &= (\mathbf{w}^{(t-1)} + \boldsymbol{\pi}) \mod 1 = \cdots = (\mathbf{w}^{(0)} + t\boldsymbol{\pi}) \mod 1 && (4.5)\\
x_i^{(t)} &= \begin{cases} 1 & \text{if } w_i^{(t-1)} < \pi_i \\ 0 & \text{otherwise} \end{cases}, \forall 1 \leq i \leq N && (4.6)
\end{aligned}$$

The Kronecker-Weyl theorem Weyl (1916) states that the sequence $\tilde{\mathbf{w}}^{(t)} = t\boldsymbol{\pi} \mod 1, t \in \mathbb{Z}^+$ is equidistributed (or uniformly distributed) on the unit hypercube $(0,1]^N$ if and only if $(1, \pi_1, \ldots, \pi_N)$ is rationally independent. Since we can define a one-to-one volume preserving transformation between $\tilde{\mathbf{w}}^{(t)}$ and $\mathbf{w}^{(t)}$ as $(\tilde{\mathbf{w}}^{(t)} + \mathbf{w}^{(0)}) \mod 1 = \mathbf{w}^{(t)}$, the sequence of weights $\{\mathbf{w}^{(t)}\}$ is also uniformly distributed on $(0,1]^N$.

Define the mapping from a state value x_i to an interval of w_i as

$$A_i(x) = \begin{cases} (0, \pi_i] & \text{if } x = 1 \\ (\pi_i, 1] & \text{if } x = 0 \end{cases} \tag{4.7}$$

and let $|A_i|$ be its measure. We obtain the limiting distribution of the joint state as

$$\begin{aligned}
\lim_{T\to\infty} P_T^{(\tau)}(\mathbf{X} = \mathbf{x}) &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\left[\mathbf{w}^{(t-1)} \in \prod_{i=1}^{N} A_i(x_i)\right] \\
&= \prod_{i=1}^{N} |A_i(x_i)| \quad \text{(Uniform distribution of } \mathbf{w}) \\
&= \prod_{i=1}^{N} \pi(X_i = x_i) \quad \text{(Definition of } A_i) \\
&= \pi(\mathbf{X} = \mathbf{x}) \quad \text{(Independent variables)} && (4.8)
\end{aligned}$$

□

The rate of convergence of $P_T^{(\tau)}$ on π depends on the rate of convergence of $\mathbf{w}_t$ on the uniform distribution. While rationally independence among $\{\pi_i\}$ is the necessary and sufficient condition for the sequence of $\mathbf{w}_t$ to be uniformly distributed, aka. equidistributed, a fast rate of convergence relies on stronger properties of $\{\pi_i\}$. This problem is closely related to the areas of Diophantine approximation, low discrepancy sequence, and Quasi Monte Carlo methods (Niederreiter, 1992). Particularly, one would achieve a fast convergence rate in the form of $\mathcal{O}(\log(T)^d/T)$ with d a constant, if $\{\mathbf{w}_t\}$ is a low-discrepancy sequence. Niederreiter (1973) gave an upper bound on the discrepancy of the sequence $\{t\boldsymbol{\pi} \mod 1\}, t = 1, \ldots, T$ as a function of the "type" of the set $\{\pi_i\}$ which describes the "badness" of a set of irrational numbers being simultaneously approximated by rational numbers (Definition 6.1).

THEOREM **4.2** (Theorem 6.1 in (Niederreiter, 1973) with slight change in notations)**.** *Let* $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ *be an* N*-tuple of irrationals of type* $\eta = 1$*. Then the discrepancy* $D_T(\mathbf{w})$ *of the sequence* $\mathbf{w} = t\boldsymbol{\pi} \mod 1, t = 1, 2, \ldots,$ *satisfies*

$$D_T(\mathbf{w}) = \mathcal{O}(T^{-1+\epsilon}) \text{ for every } \epsilon > 0. \tag{4.9}$$

It then follows from Equation 4.8 that the total variation distance between $P_t^{(\tau)}$ and π would also decay at a rate in the same order

THEOREM **4.3.** *If the set of marginal probabilities* $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ *is an* N*-tuple of irrationals of type* $\eta = 1$*, the total variation distance between* $P_T^{(\tau)}$ *and* π *satisfies*

$$d_v(P_T^{(\tau)}, \pi) = \mathcal{O}(T^{-1+\epsilon}) \text{ for every } \epsilon > 0. \tag{4.10}$$

Note however that it is a very restrictive condition for the set $\boldsymbol{\pi}$ to be of type 1. We will leave the relationship between the convergence rate and the property of a typical set of $\boldsymbol{\pi}$ for future study.

4.1.3 Convergence on Fully-Connected Graphs

While the convergence of herded Gibbs on empty graphs relies on the value of the marginal probabilities, we can prove that on fully-connected graphs herded Gibbs always converges to the target distribution even in the presence of rational ratios between the probabilities. Moreover, herded Gibbs achieves a convergence rate of $\mathcal{O}(1/T)$ with a $\mathcal{O}(\log(T))$ burn-in period on this type of graphs. That is faster than both Gibbs sampling with a rate of $\mathcal{O}(1/\sqrt{T})$ and Quasi Monte Carlo methods with a typical rate of $\mathcal{O}(\log(T)^d/T)$. We formalize the above statement in Theorem 4.4.

THEOREM **4.4.** *For a fully-connected graph, when herded Gibbs has a fixed scanning order and a Dobrushin coefficient of the corresponding Gibbs sampler $\eta < 1$, there exist constants $l > 0$, and $B > 0$ such that*

$$d_v(P_T^{(\tau)} - \pi) \leq \frac{\lambda}{T}, \forall T \geq T^*, \tau > \tau^*(T) \tag{4.11}$$

where $\lambda = \frac{2N(1+\eta)}{l(1-\eta)}$, $T^* = \frac{2B}{l}$, $\tau^*(T) = \log_{\frac{2}{1+\eta}}\left(\frac{(1-\eta)lT}{4N}\right)$, and $d_v(\delta\pi) \stackrel{\text{def}}{=} \frac{1}{2}||\delta\pi||_1$.

If we ignore the burn-in period and start collecting samples simply from the beginning, we achieve a convergence rate of $\mathcal{O}(\frac{\log(T)}{T})$ as stated in the following corollary.

COROLLARY **4.5.** *When the conditions of Theorem 4.4 hold, and we start collecting samples at the end of every sweep from the beginning, the error of the sample distribution is bounded by:*

$$d_v(P_T^{(\tau=0)} - \pi) \leq \frac{\lambda + \tau^*(T)}{T} = O(\frac{\log(T)}{T}), \quad \forall T \geq T^* + \tau^*(T^*) \tag{4.12}$$

Theorem 4.4 gives a convergence rate on the joint distribution. We provide a lower (upper)

bound for the value of constant l (B) later in Equation A.7 when proving Proposition 4.6 in the appendix. The constant l has an exponential term, with N in the exponent, leading to an exponentially large constant in the convergence rate λ. This is, however, unavoidable for any sampling algorithm when considering the convergence on a joint distribution with 2^N states.

As for the marginal distributions, it is obvious that the convergence rate of herded Gibbs is also bounded by $\mathcal{O}(1/T)$ because marginal probabilities are linear functions of the joint distribution. However, in practice, we observe very rapid convergence results for the marginals, so stronger theoretical results about the convergence of the marginal distributions seem plausible.

There is another algorithm in the literature with the same $\mathcal{O}(1/T)$ convergence rate known as stack walk mechanism (Holroyd & Propp, 2010). Herded Gibbs on a fully-connected graph is reminiscent of the stack mechanism when considering the joint state of the graph as a state in the stack walk. Both of the algorithms are deterministic, use a set of auxiliary variables to guide the state transition, and both achieve a convergence rate of $\mathcal{O}(1/T)$. However, these two algorithms are not equivalent. Firstly, herded Gibbs is based on Gibbs sampling algorithm and uses different kernels for different variables, while the stack mechanism uses a single kernel. Secondly and more importantly, herding uses one weight for every *conditioning* state in order to match the conditional distribution $\pi(Y_i|\mathbf{X}_{-i})$, while the stack mechanism uses one stack for every *joint* state in order to match the transition probability $\mathcal{T}(\mathbf{Y}|\mathbf{X})$, i.e. two separate stacks for every conditioning state in the Gibbs setting. Nevertheless, exploring the relationship between these two algorithms will help us understand both better, and the tools used in their proofs might be applied to the other as well.

The proof of Theorem 4.4 and Corollary 4.5 is provided in Appendix A. We only describe the outline here and give main results obtained in the proof.

4.1.3.1 Notation

We first introduce a few notations that are necessary to understand the main results in the proof. Assume without loss of generality that in the systematic scanning policy, the variables are sampled in the order $1, 2, \cdots, N$.

State Distribution

- Denote by $\mathcal{X}_+$ the support of the distribution π, that is, the set of states with positive probability.
- We use τ to denote the time in terms of sweeps over all of the N variables, and t to denote the time in terms of steps where one step constitutes the updating of one variable. For example, at the end of τ sweeps, we have $t = \tau N$.
- Recall the sample/empirical distribution, $P_T^{(\tau)}$, presented in Definition 4.1.
- Denote the sample/empirical distribution at the i^{th} step within a sweep as $P_{T,i}^{(\tau)}, \tau \geq 0, T > 0, 0 \leq i \leq N$, as shown in Figure 4.4:

$$P_{T,i}^{(\tau)}(\mathbf{X} = \mathbf{x}) = \frac{1}{T} \sum_{k=\tau}^{\tau+T-1} \mathbb{I}(\mathbf{X}^{(kN+i)} = \mathbf{x}).$$

  This is the distribution of T samples collected at the i^{th} step of every sweep, starting from the τ^{th} sweep. Clearly, $P_T^{(\tau)} = P_{T,0}^{(\tau)} = P_{T,N}^{(\tau-1)}$.

- Denote the distribution of a regular Gibbs sampling Markov chain after L sweeps of updates over the N variables with $\pi^{(L)}, L \geq 0$.

  For a given time τ, we construct a Gibbs Markov chain with initial distribution $\pi^0 = P_T^{(\tau)}$ and the same scanning order of herded Gibbs, as shown in Figure 4.5.

Transition Kernel

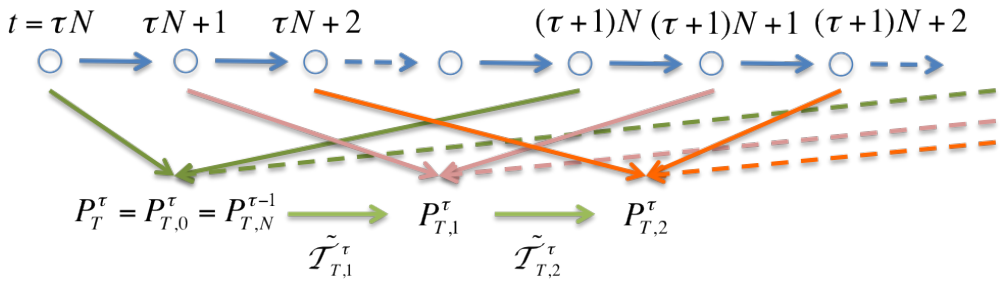


Figure 4.4: Distribution over time within a sweep.

- Denote the transition kernel of regular Gibbs for the step of updating a variable X_i with $\mathcal{T}_i$, and for a whole sweep with $\mathcal{T}$.

  By definition, $\pi^0\mathcal{T} = \pi^1$. The transition kernel for a single step can be represented as a $2^N \times 2^N$ matrix:

$$\mathcal{T}_i(\mathbf{x}, \mathbf{y}) = \begin{cases} \pi(X_i = y_i | \mathbf{x}_{-i}), & \text{if } \mathbf{x}_{-i} = \mathbf{y}_{-i} \\ 0, & \text{otherwise} \end{cases}, 1 \leq i \leq N, \mathbf{x}, \mathbf{y} \in \{0,1\}^N \tag{4.13}$$

  where $\mathbf{x}$ is the current state vector of N variables, $\mathbf{y}$ is the state of the next step, and $\mathbf{x}_{-i}$ denotes all the components of $\mathbf{x}$ excluding the i^{th} component. If $\pi(\mathbf{x}_{-i}) = 0$, the conditional probability is undefined and we set it with an arbitrary distribution. Consequently, $\mathcal{T}$ can also be represented as:

$$\mathcal{T} = \mathcal{T}_1 \mathcal{T}_2 \cdots \mathcal{T}_N.$$

- Denote the Dobrushin ergodic coefficient Brémaud (1999) of the regular Gibbs kernel with $\eta \in [0, 1]$. When $\eta < 1$, the regular Gibbs sampler has a geometric rate of convergence of

$$d_v(\pi^{(1)} - \pi) = d_v(\mathcal{T}\pi^{(0)} - \pi) \leq \eta d_v(\pi^{(0)} - \pi), \forall \pi^{(0)}. \tag{4.14}$$

  A common sufficient condition for $\eta < 1$ is that $\pi(\mathbf{X})$ is strictly positive.

- Consider the sequence of sample distributions $P_T^{(\tau)}, \tau = 0, 1, \cdots$ in Figures 4.1 and 4.4. We define the transition kernel of herded Gibbs for the step of updating variable X_i with $\tilde{\mathcal{T}}_{T,i}^{(\tau)}$, and for a whole sweep with $\tilde{\mathcal{T}}_T^{(\tau)}$.

  Unlike regular Gibbs, the transition kernel is not homogeneous. It depends on both the time τ and the sample size T. Nevertheless, we can still represent the single step transition kernel as a matrix:

$$\tilde{\mathcal{T}}_{T,i}^{(\tau)}(\mathbf{x}, \mathbf{y}) = \begin{cases} P_{T,i}^{(\tau)}(X_i = y_i | \mathbf{x}_{-i}), & \text{if } \mathbf{x}_{-i} = \mathbf{y}_{-i} \\ 0, & \text{if } \mathbf{x}_{-i} = \mathbf{y}_{-i} \end{cases}, 1 \le i \le N, \mathbf{x}, \mathbf{y} \in \{0, 1\}^N, \quad (4.15)$$

  where $P_{T,i}^{(\tau)}(X_i = y_i | \mathbf{x}_{-i})$ is defined as:

$$\begin{aligned} &P_{T,i}^{(\tau)}(X_i = y_i | \mathbf{x}_{-i}) = \frac{N_{\text{num}}}{N_{\text{den}}} \\ &N_{\text{num}} = TP_{T,i}^{(\tau)}(\mathbf{X}_{-i} = \mathbf{x}_{-i}, X_i = y_i) = \sum_{k=\tau}^{\tau+T-1} \mathbb{I}(\mathbf{X}_{-i}^{(kN+i)} = \mathbf{x}_{-i}, X_i^{(kN+i)} = y_i) \\ &N_{\text{den}} = TP_{T,i-1}^{(\tau)}(\mathbf{X}_{-i} = \mathbf{x}_{-i}) = \sum_{k=\tau}^{\tau+T-1} \mathbb{I}(\mathbf{X}_{-i}^{(kN+i-1)} = \mathbf{x}_{-i}), \end{aligned} \quad (4.16)$$

  where N_{num} is the number of occurrences of a joint state, and N_{den} is the number of occurrences of a conditioning state in the previous step. When $\pi(\mathbf{x}_{-i}) = 0$, we know that $N_{\text{den}} = 0$ with a proper initialization of herded Gibbs, and we simply set $\tilde{\mathcal{T}}_{T,i}^{(\tau)} = \mathcal{T}_i$ for these entries. It is not hard to verify the following identity by expanding every term with its definition

$$P_{T,i}^{(\tau)} = P_{T,i-1}^{(\tau)} \tilde{\mathcal{T}}_{T,i}^{(\tau)}$$

  and consequently,

$$P_T^{(\tau+1)} = P_T^{(\tau)} \tilde{\mathcal{T}}_T^{(\tau)}$$

  with

$$\tilde{\mathcal{T}}_T^{(\tau)} = \tilde{\mathcal{T}}_{T,1}^{(\tau)} \tilde{\mathcal{T}}_{T,2}^{(\tau)} \cdots \tilde{\mathcal{T}}_{T,N}^{(\tau)}.$$

4.1.3.2 Outline of Proof

Herded Gibbs is designed to sample from the target distribution by matching conditional distributions. The approximation error of each full-conditional, $\pi(X_i = 1|\mathbf{x}_{\mathcal{N}(i)})$, is controlled by one weight, $w_{i,\mathbf{x}_{\mathcal{N}(i)}}$. Proposition 2.5 shows that the error is bounded by $1/T$ where T in the context of herded Gibbs is the number of times the weight $w_{i,\mathbf{x}_{\mathcal{N}(i)}}$ is updated. Unlike regular herding where every weight is updated every iteration, in herded Gibbs whether a weight is selected to be updated in step 4 of Algorithm 4.1 depends on whether it matches the current state of its neighbors, i.e., $\mathbf{X}_{\mathcal{N}(i)} = \mathbf{x}_{\mathcal{N}(i)}$. Therefore, in order to match every full-conditional, it is essential for every weight to be updated at a sufficiently large rate. The following proposition says that every joint state of the graph will be visited at least at a linear rate, and consequently every weight will be updated at least at the same rate.

PROPOSITION **4.6.** *If a graph is fully connected, herded Gibbs sampling scans variables in a fixed order, and the corresponding Gibbs sampling Markov chain is irreducible, then for any state $\mathbf{x} \in \mathcal{X}_+$ and any index $i \in [1, N]$, the state is visited at least at a linear rate. Specifically,*

$$\exists l > 0, B > 0, s.t., \forall i \in [1, N], \mathbf{x} \in \mathcal{X}_+, T \in \mathbb{N}, s \in \mathbb{N}$$
$$\sum_{k=s}^{s+T-1} \mathbb{I}\left[\mathbf{X}^{(t=Nk+i)} = \mathbf{x}\right] \geq lT - B \tag{4.17}$$

The proof is given in Appendix A.1 where we also provide a lower bound for the visiting rate in Equation A.7.

Following Proposition 4.6, we are able to show that the transition kernel of herded Gibbs $\tilde{\mathcal{T}}_T^{(\tau)}$ approximates the kernel of regular Gibbs $\mathcal{T}$ with an error upper bounded by $\mathcal{O}(1/T)$.

PROPOSITION **4.7.** *For a fully connected graph, if the herded Gibbs has a fixed scanning order and the corresponding Gibbs sampling Markov chain is irreducible, then for any $\tau \geq 0$,*

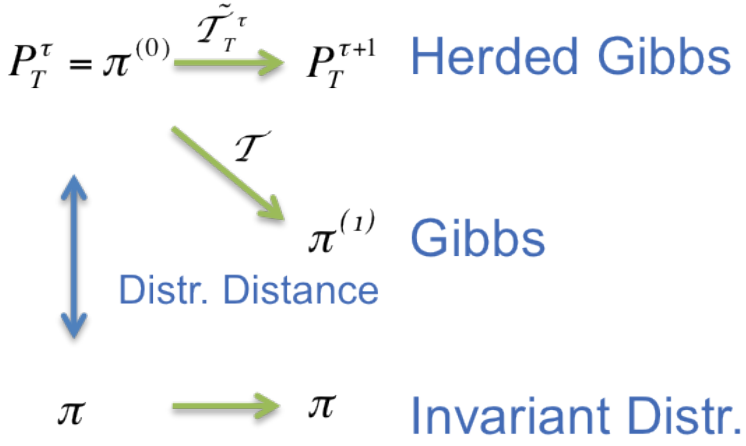


Figure 4.5: Transition kernels and relevant distances for the proof of Theorem 4.4.

$T \geq T^* \stackrel{\text{def}}{=} \frac{2B}{l}$ where l and B are the constants in Proposition 4.6, the following inequality holds:

$$\|\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T}\|_\infty \leq \frac{4N}{lT} \tag{4.18}$$

The proof is given in Appendix A.2.

With basic tools ready, we can now study the distribution distance between the target distribution π and the empirical distribution $P_T^{(\tau)}$. The detailed proof is provided in Appendix A.3 and the main idea is as follows. To study the change of distance after one sweep, we construct an auxiliary regular Gibbs sampling Markov chain initialized at $\pi^{(0)} = P_T^{(\tau)}$, as shown in Figure 4.5. The Gibbs chain has geometric convergence to π, and following Proposition 4.7 the distance between the Gibbs and herded Gibbs chains after one sweep is bounded by $\mathcal{O}(1/T)$. When the distance between π and $P_T^{(\tau)}$ is large enough, the geometric convergence rate to π dominates the discrepancy of herded Gibbs and thus we infer that $P_T^{(\tau)}$ converges to π as a function of τ geometrically. This burn-in period ends when the discrepancy of herded Gibbs becomes small enough so that the geometric decrease in the Gibbs chain is on par with the error $O(1/T)$. To round-off the proof, we must find a limiting

value for τ. The proof concludes with an $O(\log(T))$ burn-in for τ.

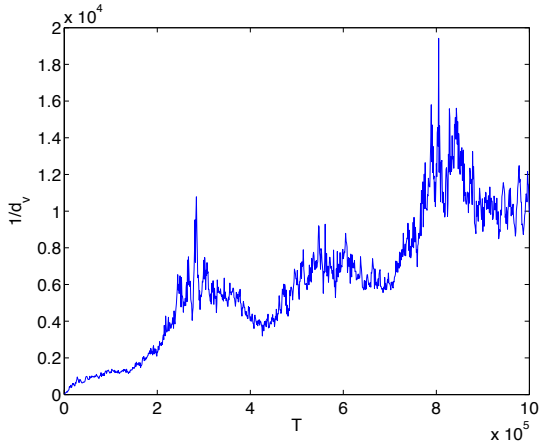
4.1.4 Convergence Condition on Generic Graphs

Despite the favorable convergence property of herded Gibbs on empty and fully-connected graphs, we have no mathematical guarantees on the convergence rate for a generic graph. In fact, one can easily construct synthetic examples for which herded Gibbs does not seem to converge to the true marginals and joint distribution. For an example, Figure 4.6 shows the total variation distance between $P_T^{(\tau=0)}$ and π as a function of the sample size T for a graph with four nodes. For the empty graph (Figure 4.6a) and fully-connected graph (Figure 4.6b), the distance keeps decreasing as predicted by Theorem 4.1 and 4.4. For an incomplete graph, herded Gibbs converges toward the target distribution for some scanning order (Figure 4.6c) and the error stays at a nonzero value for other orders (Figure 4.6d). The exact conditions under which herded Gibbs converges for sparsely connected graphs are still unknown.

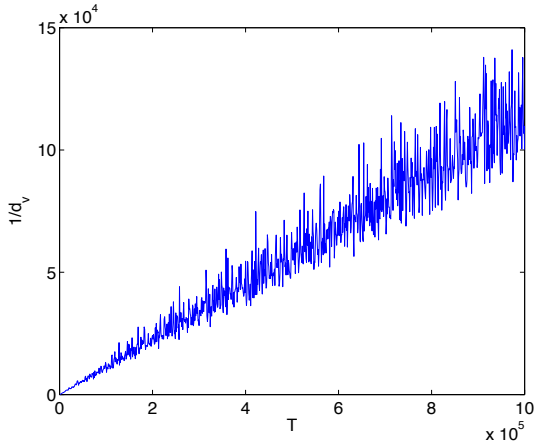
We refer readers to Bornn et al. (2013) for more (synthetic and real data) experiments, where we demonstrate that the new algorithm outperforms Gibbs sampling and mean field methods in the domain of sampling from sparsely connected probabilistic graphical models, such as grid-lattice Markov random fields (MRFs) for image denoising and conditional random fields (CRFs) for natural language processing.

4.2 Herding in a Continuous State Space

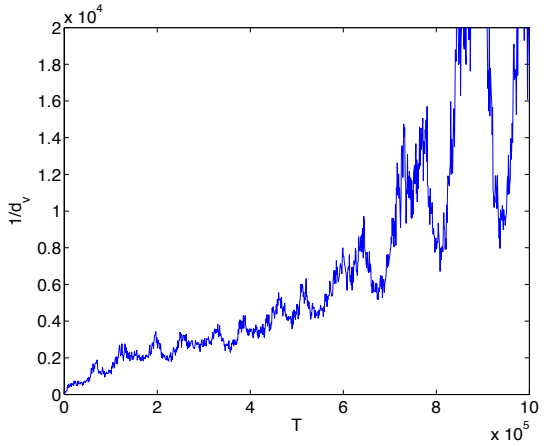
We have been restricting the herding algorithm to draw samples in a discrete state space so far. This is partly because the convergence proof with PCT requires a finite set of feature vectors. More importantly, when herding works on a continuous state space with a finite number of features, imposing a finite number of constraints using the moment matching



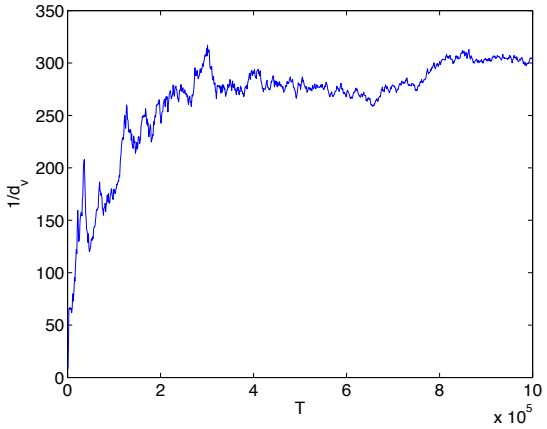
(a) Empty graph.



(b) Fully-connected graph. $1/d_v$ increases linearly with T.



(c) Incomplete graph. Four variables are connected in a loop 1-2-3-4-1. Sampling in the order 1-3-2-4-1-.... Herded Gibbs converges on the target distribution.



(d) Incomplete graph. Four variables are connected in a loop 1-2-3-4-1. Sampling in the order 1-2-3-4-1-.... Herded Gibbs does *not* converge on the target distribution.

Figure 4.6: Reciprocal of the total variation distance between $P_T^{(\tau=0)}$ and π as a function of the sample size T. The distance is computed on a Boltzmann machine with four binary nodes. The parameters are shared across all the figure but in some figures part or all of the edges are removed to make an incomplete graph.

property can not sufficiently control the infinite number of degrees of freedom in continuous spaces leading to strange artifacts in the samples. For instance, herding in a continuous space with features given by the mean and variance may produce two delta-peaks instead of a Gaussian. To overcome this we wish to perform herding on an infinite number of features implying the need to switch to a kernel representation.

4.2.1 Kernel Herding

To achieve that, we will start from the duel view of herding in Section 2.4. As we "marginalize out" the weight vector $\mathbf{w}$, the kernel trick is straightforward.

In this section, we generalize the definition of the state features, $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$, as a mapping into a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle$. The moments $\bar{\boldsymbol{\phi}} = \mathbb{E}_{\mathbf{x} \sim p}[\boldsymbol{\phi}(\mathbf{x})]$ correspond to the mean operator associated with the input distribution $p(\mathbf{x})$ in $\mathcal{H}$, i.e. for any function in $\mathcal{H}$, $f(\mathbf{x}) = \langle \mathbf{w}_f, \boldsymbol{\phi}(\mathbf{x}) \rangle$, we have $\mathbb{E}_{\mathbf{x} \sim p}[f(\mathbf{x})] = \langle \mathbf{w}_f, \bar{\boldsymbol{\phi}} \rangle$.

We have implicitly assumed that there are many more discrete states than features. This has the effect that we only "control" the error in a small subspace of the full state space. The natural question is whether we can *take the (nonparametric) limit where the number of features is infinite*. This is in fact rather straightforward because under the dual view, the herding update equation 2.15 only depends on the inner product

$$k(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle \tag{4.19}$$

The *kernel herding* procedure (Chen et al., 2010) becomes:

$$\mathbf{s}_{T+1} = \arg\max_{\mathbf{x}} \mathbb{E}_{\mathbf{x}' \sim p}[k(\mathbf{x}, \mathbf{x}')] - \frac{1}{T+1} \sum_{t=1}^{T} k(\mathbf{x}, \mathbf{s}_t) \tag{4.20}$$

The error is now defined under the norm of the Hilbert space

$$\mathcal{E}_T^2 \stackrel{\text{def}}{=} \| \bar{\boldsymbol{\phi}} - \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{s}_t) \|_{\mathcal{H}}^2 \tag{4.21}$$

$\mathcal{E}_T$ with an infinite dimensional kernel measures the distance between p and the empirical measure $\hat{p}_T(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \delta(\mathbf{x}, \mathbf{s}_t)$ given by the herding samples.

This algorithm iteratively constructs an empirical distribution $\hat{p}_T(\mathbf{x})$ that is close to the true

distribution $p(\mathbf{x})$. At each iteration, it searches for a new sample to add to the pool. It is attracted to the regions where p is high but repelled from regions where samples have already been "dropped down". The kernel determines how we should measure distances between distributions. Note that for many distributions explicit expressions for $\mathbb{E}_{\mathbf{x}' \sim p}[k(\mathbf{x}, \mathbf{x}')]$ have been obtained. See Jebara & Kondor (2003) for details.

4.2.2 Convergence in Hilbert Space

We show in the next paragraphs that the pseudo-samples generated by kernel herding inherit the fast $\mathcal{O}(T^{-1})$ decrease in error. Recall the definition of marginal polytope $\mathcal{M}$ in Equation 3.49. It follows that $\bar{\boldsymbol{\phi}} \in \mathcal{M}$ since $\mathcal{X}$ contains the support of p. If $\|\boldsymbol{\phi}(\mathbf{x})\| \leq R$ for all $\mathbf{x} \in \mathcal{X}$ it follows that $\|\bar{\boldsymbol{\phi}}\| \leq R$ and consequently by the triangle inequality we have that $\|\bar{\boldsymbol{\phi}} - \boldsymbol{\phi}(\mathbf{x})\| \leq 2R, \forall \mathbf{x}$.

PROPOSITION **4.8.** *Assume that p is a distribution with support contained in $\mathcal{X}$ and assume that $\|\boldsymbol{\phi}(\mathbf{x})\| \leq R$ for all $\mathbf{x} \in \mathcal{X}$. Moreover assume $\bar{\boldsymbol{\phi}}$ is in the relative interior of the marginal polytope $\mathcal{M}$. Then the error $\mathcal{E}_T$ of Equation 4.21 will decrease as $\mathcal{O}(T^{-1})$.*

Proof. We first show that $\|w_t\|$ is bounded for all t. For this we introduce the centered marginal polytope

$$\mathcal{C} := \mathcal{M} - \mu_p = \operatorname{conv}\{\phi(\mathbf{x}) - \mu_p | \mathbf{x} \in \mathcal{X}\}. \tag{4.22}$$

Using $\mathcal{C}$ the update equations become

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{c}_t \text{ where } \mathbf{c}_t := \operatorname{argmax}\mathbf{c} \in \mathcal{C} \langle \mathbf{w}_t, \mathbf{c} \rangle. \tag{4.23}$$

This allows us to write the increment in the norm of the parameter vector $\|\mathbf{w}_{t+1}\|$ via

$$\begin{aligned}\|\mathbf{w}_t\|^2 - \|\mathbf{w}_{t+1}\|^2 &= 2\langle \mathbf{w}_t, \mathbf{c}_t\rangle - \|\mathbf{c}_t\|^2 \\ &\geq 2\|\mathbf{c}_t\| \left[\|\mathbf{w}_t\| \left\langle \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}, \frac{\mathbf{c}_t}{\|\mathbf{c}_t\|} \right\rangle - R\right]\end{aligned} \tag{4.24}$$

The inequality follows from $\|\mathbf{c}_t\| \leq 2R$. If we can show

$$\left\langle \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}, \frac{\mathbf{c}_t}{\|\mathbf{c}_t\|} \right\rangle =: \gamma_t \geq \gamma^* > 0 \tag{4.25}$$

for all $\mathbf{w}$ then it follows immediately that $\|\mathbf{w}\| \leq R/\gamma^*$: in this case we have $\|\mathbf{w}\|\gamma_t - R \geq (R/\gamma^*)\gamma^* - R = 0$.

To see 4.25 recall that μ_p is contained inside the relative interior of $\mathcal{M}$, i.e. there exists an ϵ-ball around μ_p that is contained in $\mathcal{M}$. Consequently $\gamma^* \geq \epsilon$.

Since $\|\mathbf{w}_t\| = \left\| w_0 + T\mu_p - \sum_{t=1}^{T} \phi(\mathbf{x}_t) \right\| \leq R/\gamma^*$ it follows by dividing by T that

$$\left\| \mu_p - T^{-1} \sum_{t=1}^{T} \phi(\mathbf{x}_t) \right\| \leq T^{-1}[\|w_0\| + R/\gamma^*]. \tag{4.26}$$

This proves the claim of $\mathcal{O}(T^{-1})$ convergence to μ_p. □

COROLLARY **4.9.** *Herding converges at the fast rate even when 2.6 is only carried out with some error provided that we obtain samples $\mathbf{x}_{t+1} \in \mathcal{X}$ which satisfy*

$$\left\langle \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}, \frac{\phi(\mathbf{x}_{t+1}) - \mu_p}{\|\phi(\mathbf{x}_{t+1}) - \mu_p\|} \right\rangle \geq \bar{\rho} > 0 \tag{4.27}$$

This condition is reminiscent of Boosting algorithms where the weak learner is not required to generate the optimal solution within a given set of hypotheses but only one that is sufficiently good with regard to a nonzero margin. It is also related to the *perceptron cycling theorem*

(Block & Levin, 1970) where $\mathcal{X}$ is assumed to have finite cardinality but which guarantees convergence even when $\bar{\rho} = 0$.

We can allow μ_p to lie on a facet of $\mathcal{M}$ in which case we have the following corollary.

COROLLARY **4.10.** *Whenever μ_p lies on a facet of the marginal polytope $\mathcal{M}$ it suffices that we restrict ourselves to optimization over the vertices generating the facet. In this case, μ_p lies within the relative interior of the now restricted polytope.*

Compared to the PCT condition that requires a finite domain of the update vector $\mathbf{v}$, Proposition 4.8 and its corollaries apply to an infinite set of $\mathcal{X}$ as well. However, Proposition 4.8 and 4.9 assume that $\bar{\boldsymbol{\phi}}$ is in the relative interior of $\mathcal{M}$ and 4.10 assumes that $\bar{\boldsymbol{\phi}}$ is on a facet of $\mathcal{M}$. As pointed out by Bach et al. (2012), these assumptions do not hold for infinite dimensional kernel functions such as the Gaussian kernel. So Proposition 4.8 is not sufficient to prove that the empirical distribution of herding samples will converge to p. Nevertheless, even with an infinite dimensional kernel on a continuous state space, we find in experiments that the error $\mathcal{E}_T$ decays at a rate close to $\mathcal{O}(T^{-1})$. See Figure 4.8 for an example of the linear decrease of error with a Gaussian kernel.

Under the condition of Proposition 4.8, we want to show that the convergence rate of the error $\mathcal{E}_T$, $\mathcal{O}(T^{-1})$, implies that the error of any integral over a function in our RKHS will also converge at the same fast rate:

PROPOSITION **4.11.** *For any $f \in \mathcal{H}$, the error $|\mathbb{E}_p[f] - \mathbb{E}_{\hat{p}_T}[f]|$ will decrease as $\mathcal{O}(T^{-1})$. Moreover this condition holds uniformly, that is $\sup_{\|f\|\leq 1} |\mathbb{E}_p[f] - \mathbb{E}_{\hat{p}_T}[f]|$ also decreases at rate $\mathcal{O}(T^{-1})$.*

To prove this we will need the following lemma,

LEMMA **4.12** (Koksma Hlawka Inequality). *For any $f \in \mathcal{H}$ we have*

$$|\mathbb{E}[f]_p - \mathbb{E}[f]_{\tilde{p}_T}| \leq \|f\|_{\mathcal{H}} \|\mu_p - \mu_{\tilde{p}_T}\|_{\mathcal{H}} \tag{4.28}$$

The above inequality is the simply a consequence of the Cauchy Schwartz inequality. It is known as the Koksma-Hlawka inequality in the analysis of Quasi Monte Carlo methods. Clearly, with this lemma proposition 4.11 follows. In fact, this technique was used by Song et al. (2008) in the context of density estimation. The key novelty in the present paper is that we have a *simple* and *explicit* algorithm for obtaining fast rates of approximation which are considerably better than the $\mathcal{O}(T^{-\frac{1}{2}})$ rates usually available via sampling.

The above proposition shows that kernel herding generates "super samples" that are much more informative than iid samples: for every n herding samples we will need $\mathcal{O}(n^2)$ iid samples to achieve the same error reduction. This fast convergence behavior is reminiscent of Quasi Monte Carlo integration. The Quasi Monte Carlo methods first generate a sequence of uniformly distributed samples deterministically and then use those samples to approximate the Monte Carlo average. They usually achieve a convergence rate of $\mathcal{O}(\log(T)^d/T)$ where d is a constant. However, the regular Quasi Monte Carlo method generates samples subject to a uniform distribution only and therefore is not as flexible as kernel herding that adjusts itself to the input distribution. An extension of Quasi Monte Carlo methods, known as Markov Chain Quasi Monte Carlo (Chen et al., 2011a) combines the deterministic process with Markov chains so that one can generate samples from a generic distribution with a faster convergence rate than MCMC methods. But the fast convergence rate is contingent on a strong contraction condition on the transition kernel of the Markov chain. Another related work on deterministic sampling (Huszar & Duvenaud, 2012) proves that an even faster convergence rate is available when assigning a weight for every sample. It shows that optimally weighted kernel herding is equivalent to Bayesian quadrature methods. Note

however that the extra speed up comes with a higher time and space complexity, which is due to the calculation of the optimal weights.

4.2.3 Experiments

In this section, we want to show that herding is able to draw better samples than random sampling from the true distribution. We first illustrate the behavior of herding on low dimensional synthetic data, compare the approximation of integrals between the super samples and iid samples, and then show an application where we compress the size of a collection of posterior samples required for computing the predictive probability of Bayesian logistic regression.

4.2.3.1 Synthetic Data: Matching the True Distribution

We first visualize herding on a 2-D state space. We randomly construct a 2 dimensional Gaussian mixture (GM) model with 20 components whose equiprobability contours are shown in Figure 4.7. With a Gaussian kernel, the integral in 4.20 can be analytically calculated implying that we can run herding directly on the GM distribution.

A few auxiliary samples are first drawn uniformly to provide a reasonable starting point for the maximization steps. Then, we sequentially generate super-samples by 4.20. At each iteration, starting from the best auxiliary sample that maximizes 4.20, we run a gradient ascent algorithm to obtain a new super sample. Figure 4.8 shows the linear increase of $1/\mathcal{E}_T$ as a function of T.

In Figure 4.7, the first 20 super samples are plotted in comparison with 20 iid samples from the GM model. For iid samples, due to the inherent randomness, some modes receive too many points while others get too few or even none. In contrast, the samples from herding

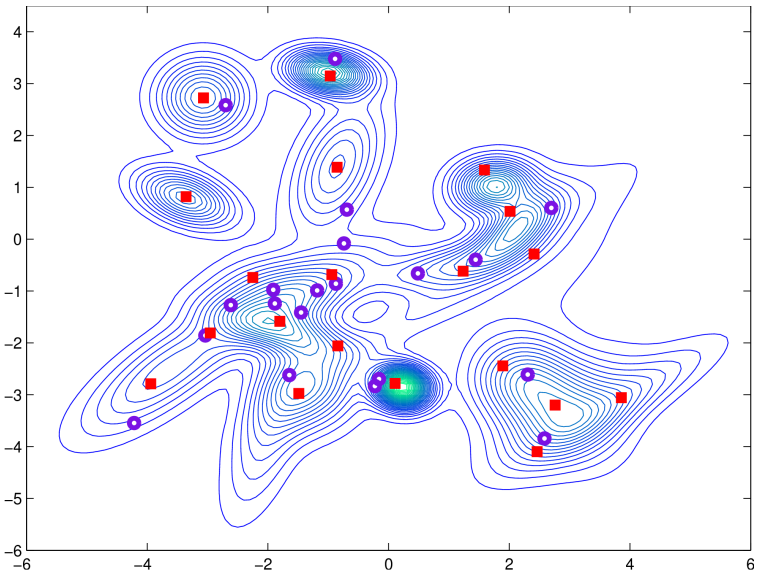


Figure 4.7: First 20 samples form herding (red squares) versus *i.i.d.* random sampling (purple circles).

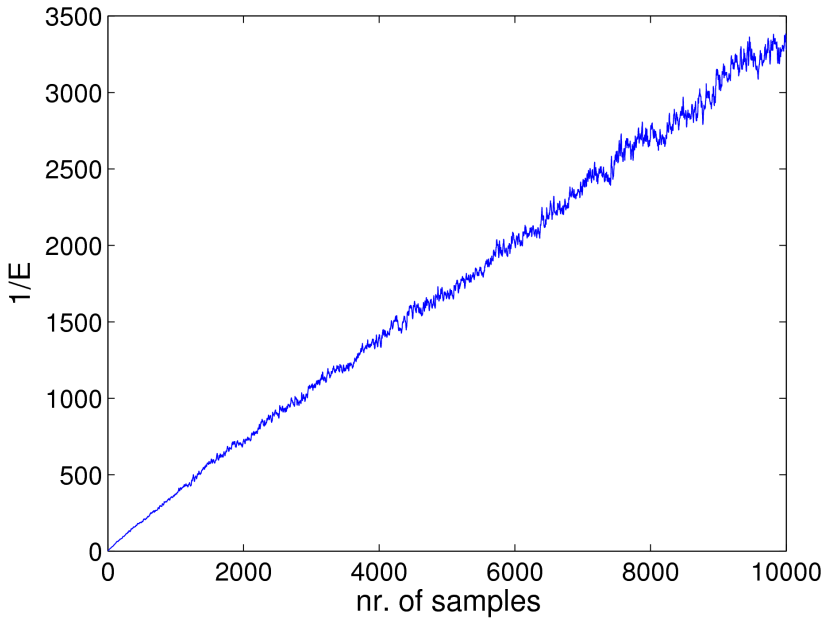


Figure 4.8: Linear relationship between $1/\mathcal{E}_T$ and T

always try to repel from each other and are distributed optimally (given earlier samples) to represent the density function.

Since the expectation of any function in a model can be approximated by summation over its samples, we are interested in how well the super samples can be used to estimate these

averages. We generate a 5 dimensional GM model with 100 components as the target distribution p. We compute the error of the expectation of four functions: the first three moments, and a nonlinear function. For the m^{th} moment, $m = 1, 2, 3$, we first calculate the average of $x_{i,t}^m$ over t (the index of herding samples) in each dimension as a function of T (the number of super samples). Then the RMSE of the estimated moments over all the dimensions is computed as

$$\text{err}(\mathcal{S}_T) = \left(\frac{1}{d} \sum_{i=1}^{d} (\langle x_i^m \rangle_{\mathcal{S}_T} - \langle x_i^m \rangle_p)^2 \right)^{\frac{1}{2}} \tag{4.29}$$

For the fourth function, we use a sine of the norm of a point: $f(x) = \sin \|\mathbf{x}\|$. In comparison, we compute the mean and standard deviation of the errors obtained by a set of random samples as the benchmark. The results are shown in Figure 4.9 with their estimated convergence rates. The error of approximation by herding is much smaller than random sampling with the same number of points for all the 4 functions, also their convergence rates are close to the theoretical value $\mathcal{O}(T^{-1})$.

4.2.3.2 Synthetic Data: Matching empirical distribution

When the integration in 4.20 cannot be computed analytically, it would be difficult to run herding to accurately match the true distribution especially in high dimensional spaces. However, if we have a set of random samples, $\mathcal{D}$, from the distribution, it is straightforward to run herding to match the empirical distribution. We can thereby represent the true distribution by the super samples $\mathcal{S}$ with the same accuracy as $\mathcal{D}$ but with many fewer samples. A set of 10^5 iid samples is drawn from a 5-D GM model, and then herding is run taking $\mathcal{D}$ as the true distribution. Since in this case p in the 4.20 is taken to be the empirical distribution, the integral is simply a summation over all the points in $\mathcal{D}$.

We again compare the estimation of function expectations between herding and random

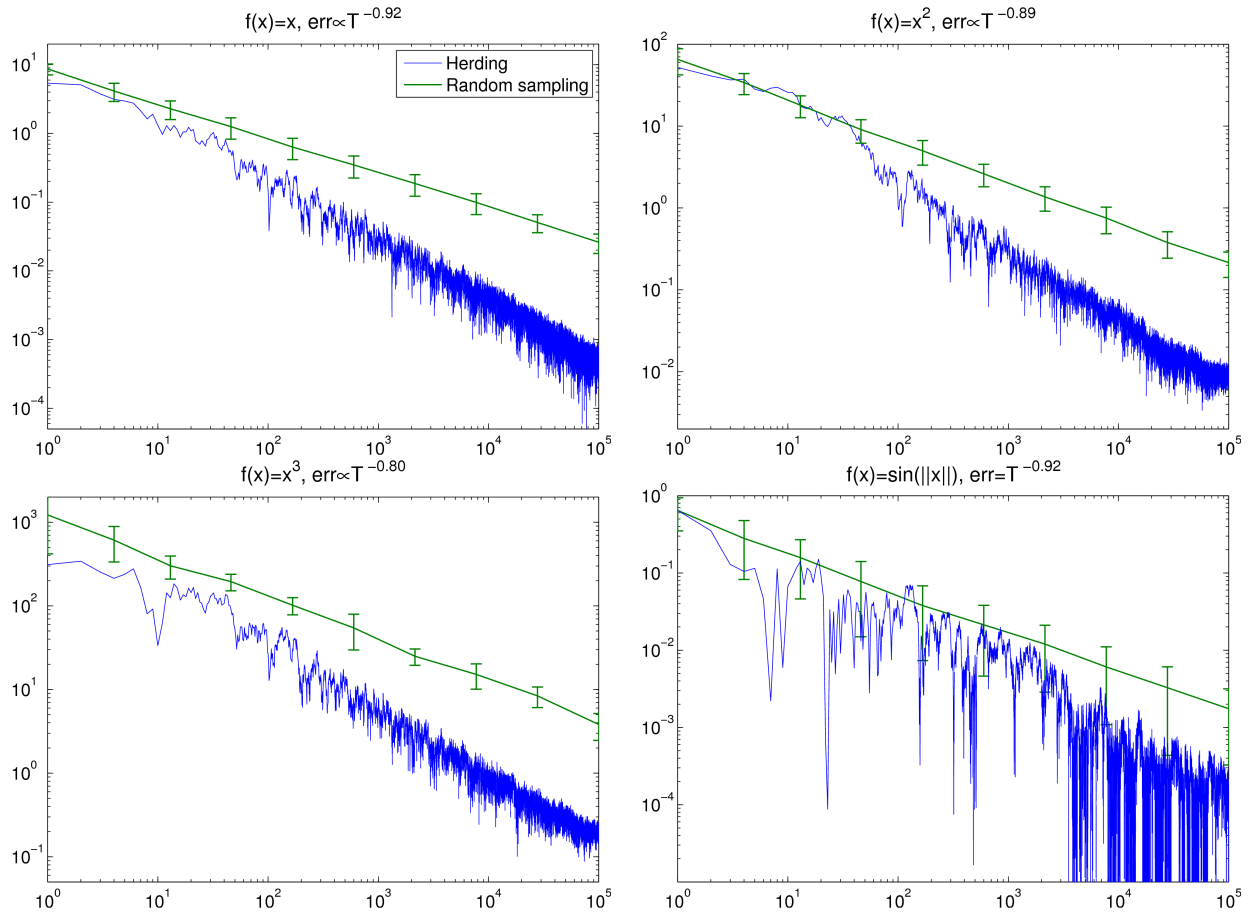


Figure 4.9: Error in estimating the expectation of four functions, by herding (blue) and random sampling (green) as a function of the number of samples. The decreasing speed of the upper bound of the error is shown on top of each figure.

samples. However, this time we can compute two errors, one on the empirical distribution $\mathcal{D}$ and the other on the true distribution p. See the results in Figure 4.10. Since the distribution of $\mathcal{S}$ will converge to the empirical distribution, the error between $\mathcal{S}$ and $\mathcal{D}$ will keep decreasing as in Figure 4.9 while the error between $\mathcal{S}$ and p will not. Instead, it will converge to the error incurred by the empirical distribution relative to p and this is the point where the set $\mathcal{S}$ is large enough to replace $\mathcal{D}$. We can find from Figure 4.10 that for 10^5 iid samples, we only need at most 2000 super samples for the first three functions, and 10^4 for the last function to achieve similar precision. This is a significant reduction whenever evaluating f is expensive, e.g. in active learning problems.

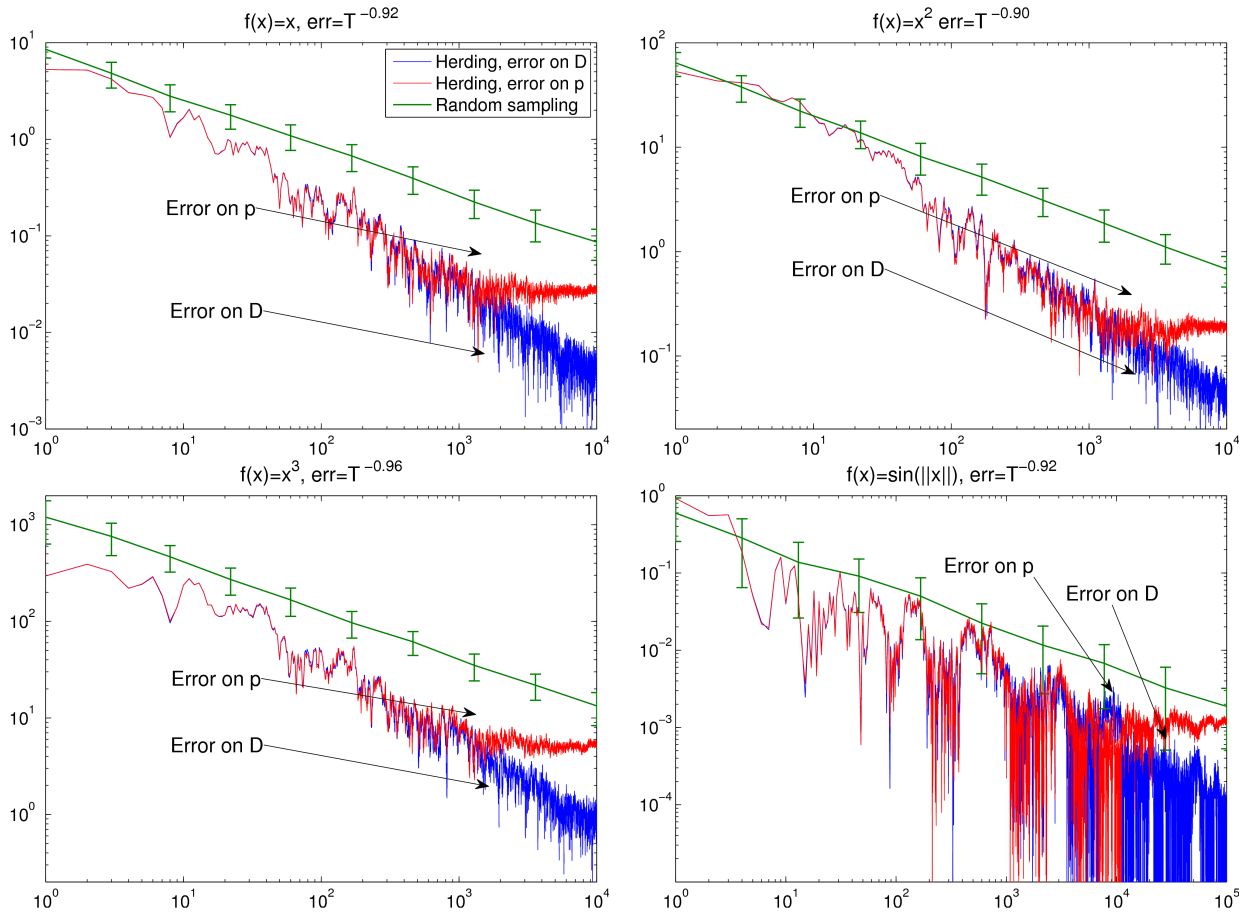


Figure 4.10: Error in estimating the expectation of four functions by herding on the true distribution p (red) and the empirical distribution $\mathcal{D}$ (blue) as a function of the number of samples. The convergence rate of the error on $\mathcal{D}$ (measured as slope of the upper bound of the herding error) is shown on top of each figure. The error of random sampling on p (green) is also plotted for comparison.

4.2.3.3 Approximating the Bayesian Posterior

Next we consider the task of approximating the predictive distribution of a Bayesian model. Alternatively, this idea can be applied to find a small collection of good predictive models to be used in bagging. Assume we have drawn a large number of parameters, $\mathcal{D}$, using MCMC from the posterior distribution (or we have learned a large number of predictors on bootstrap samples). For reasons of computational efficiency, we may not want to use all the samples at test time. One choice is to down-sample the MCMC chain by randomly sub-sampling from $\mathcal{D}$. Another choice is to run herding on the empirical distribution. With the convergence

property on any function in the Reproducing Kernel Hilbert Space, we know that prediction by $\mathcal{S}$ will converge to that by $\mathcal{D}$. Furthermore, we can get a significant speed up with a few super samples during the prediction phase without much loss of accuracy.

We use the spambase data set from the UCI machine learning repository[1] for the experiment, which has 4601 instances with 57 real attributes and 1 binary class label. The data set is split into a training set of 3000 data points and a test set of 1601 data points. A logistic regression model is built with a Gaussian prior on the weights θ. The training set is whitened by PCA and then fed into the model to draw posterior samples by the Metropolis-Hasting algorithm with a Gaussian proposal distribution. The resulting set $\mathcal{D}$ consists of 10^5 samples sub-sampled by a factor of 100 from the Markov chain to reduce the autocorrelation. We whiten $\mathcal{D}$ using PCA and run herding on this empirical distribution with an isotropic Gaussian kernel with $\sigma = 10$. This is equivalent to run herding on the original parameter set with a Gaussian kernel whose covariance matrix is a multiple of the covariance matrix of $\mathcal{D}$. At each iteration, we use the sample of θ from $\mathcal{D}$ that maximizes (4.20) as a new super sample, without any further local maximization. This corresponds to running herding in the discrete domain, $\mathcal{X} = \mathcal{D}$, and all the theoretical conclusions in section 2 also apply to this case.

We compare the predictions made by $\mathcal{S}$ with those made by the whole set $\mathcal{D}$ on the test data. Figure 4.11 shows the RMSE of the predictive probability by herding over all the test data points as a function of the number of super samples.

$$\mathrm{RMSE}^2(\mathcal{S}_T, \mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{1}{T}\sum_{t=1}^{T} p(y_n|x_n,\theta_t) - \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} p(y_n|x_n,\theta_i)\right]^2 \tag{4.30}$$

For comparison, we randomly draw a subset of $\mathcal{D}$ by bootstrap sampling and compute the error in the same way (the performance of down-sampling the Markov chain or randomly sampling without replacement is very similar to random sampling, and is thus not shown in

[1] http://archive.ics.uci.edu/ml

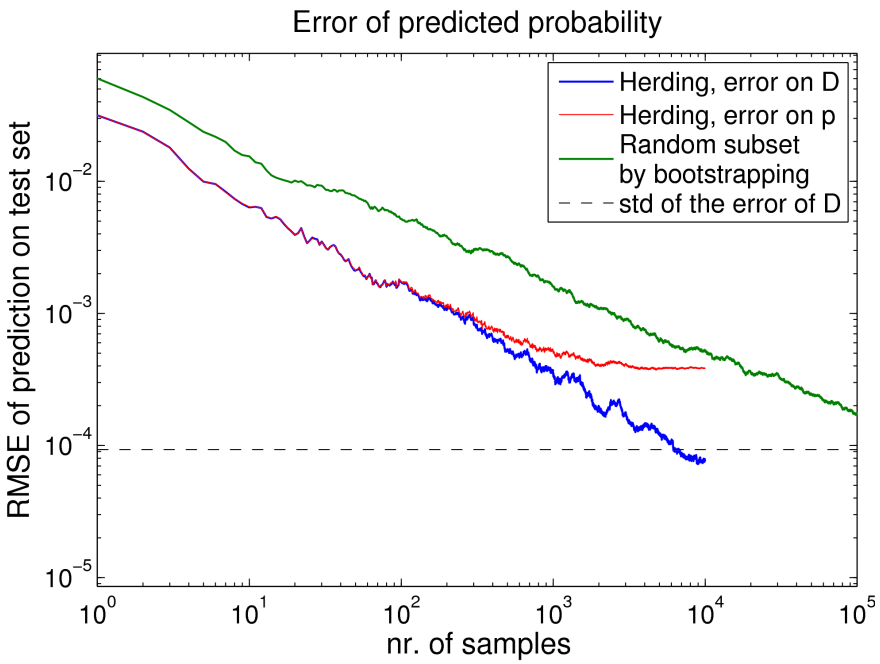


Figure 4.11: RMSE of the predicted probability of herding (blue) and a random subset (blue) w.r.t. the whole sample set.

the figure). We can easily observe the advantage of herding over random sampling. The error of herding decreases roughly as $\mathcal{O}(T^{-0.75})$, while the error of random sampling decreases as $\mathcal{O}(T^{-0.5})$.

Now we would like to estimate how many super samples are needed to achieve the same precision as $\mathcal{D}$ on the true posterior. Assume for now that the samples in $\mathcal{D}$ are iid. Then the average predictive probability $p(y_n|x_n, \mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} p(y_n|x_n, \theta_i)$ is the average of $|\mathcal{D}|$ independent, unbiased estimates. Since we can compute the standard deviation of these estimates on $\mathcal{D}$, the standard deviation of the average predictive probability becomes $\mathrm{std}(p(y_n|x_n, \mathcal{D})) = \mathrm{std}(p(y_n|x_n, \theta_i))/\sqrt{|\mathcal{D}|}$, and then its mean over all test data points gives an estimate of the standard deviation of the error in general, which is the dashed line in Figure 4.11.

We can decompose the error of herding on the true posterior

$$\mathrm{RMSE}(\mathcal{S}_T, p) \leq \mathrm{RMSE}(\mathcal{S}_T, \mathcal{D}) + \mathrm{RMSE}(\mathcal{D}, p) \approx \mathrm{RMSE}(\mathcal{S}_T, \mathcal{D}) + \overline{\mathrm{std}(p(y_n|x_n, \mathcal{D})}. \tag{4.31}$$

When the first term is smaller than the second term, the error of herding mainly comes from the error of $\mathcal{D}$, and we can claim that drawing more herding samples will not improve the prediction much. Since the MCMC samples in $\mathcal{D}$ are not independent, the error of $\mathcal{D}$ can only be larger than the estimated value, and we will need even fewer samples to reach the same accuracy. In our experiment, for a set of 10^5 samples, we only need 7000 super samples.

In fact, we have drawn another much larger set of 2.5×10^6 posterior samples, $\tilde{p}$, and estimate the error of $\mathcal{S}$ on p by $\mathrm{RMSE}(\mathcal{S}, \tilde{p})$ (the red line in Figure 4.11). We find that the line starts to level off with even fewer (about 3000) super samples and the converged value equals $\mathrm{RMSE}(\mathcal{D}, \tilde{p})$. In summary, we can compress the set of parameters by 93% or 97%.

In Figure 4.12, we show the classification accuracy of herding on the test set. In comparison, we also draw the accuracy of the whole sample set (red), and 10 random subsets of $\mathcal{D}$. The prediction of herding converges fast to that of $\mathcal{D}$ which is considered ground truth for the herding algorithm. In contrast, the prediction made by random subsets fluctuates strongly. In particular, we only need about 20 super-samples to get the same accuracy as $\mathcal{D}$, while we need about 200 random samples.

4.3 Summary of Contributions

Section 4.1 and 4.2 are basically restatements of Chen et al. (2010) and Bornn et al. (2013) respectively with an expansion of discussion in Section 4.1.2 and an additional experiment in Section 4.1.4. My contributions to this chapter are summarized as follows:

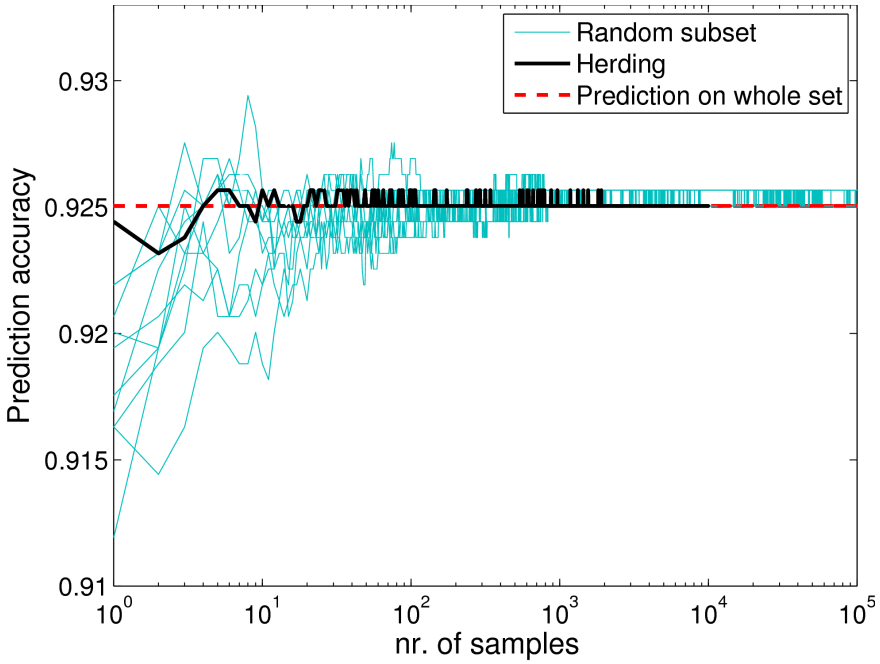


Figure 4.12: Prediction accuracy of herding (black), 10 random subsets (cyan) and the whole sample set (red).

- Analyzed the conditions and the rate of convergence for herded Gibbs sampling. I also conducted the illustrative experiments in Section 4.1.4.
- Conducted the experiments of kernel herding in Section 4.2.3.

I would like to acknowledge of contributions from my collaborators:

- Nando de Freitas proposed the herded Gibbs sampling algorithm in Section 4.1.1.
- Jing Fang and Mareija Eskelin applied herded Gibbs sampling to more comprehensive experiments. Readers are referred to Fang (2012) and Bornn et al. (2013) for detailed comparisons of herded Gibbs with other algorithms.
- Luke Bornn and Max Welling helped to formulate the idea of herded Gibbs and contributed to the editing of Bornn et al. (2013).
- The idea of kernel herding in Section 4.2.1 was proposed by Alex Smola and Max Welling. The theoretical analysis in Section 4.2.2 was contributed by Alex Smola.

Chapter 5

Conclusion

The herding algorithm was proposed in Welling (2009a) 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 model states. By integrating the intractable training and testing steps in the regular machine learning paradigm, herding provides a more efficient way of learning and predicting in MRFs.

In this thesis, we study the statistical properties of herding. We 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.

We further provide a condition, the PCT, for the fast moment matching property, which is then used to derive extensions of the herding algorithm for a wider range of applications. Parametric versions of herding are introduced to run the dynamics efficiently in the presence

of hidden variables. A discriminative learning variant is proposed for supervised problems. We also study an exception with inconsistent moments. The PCT condition does not hold any more in that case, but we show that herding can still be applied for applications such as image segmentation and game prediction.

As a dynamical system, it is worthwhile studying the weak-chaos behavior of herding. We show that the attractor set of herding usually has a fractal dimension. A typical sequence of the dynamical parameters is not periodic. So unlike the maximum likelihood point estimation, the distribution of the parameters is spread in an infinite set. Also, that sequence is not yet chaotic. So there exists longer distance auto-correlation compared to chaotic sequences and Markov process, indicated by the Lyapunov exponent equal to zero. We make a conjecture for this type of dynamics that the contained information is more than both periodic sequences and sequences generated by a random sampler.

This information-rich process is advantageous over a random process when applied as a sampling algorithm. We show that when we employ a large enough number of features to eliminate the remaining degrees of freedom or take advantage of the rational independence among moments, the sample sequence of herding is likely to converge to the input distribution. Moreover, the rate of convergence is as high as $\mathcal{O}(1/T)$ under certain conditions, a significant speed up compared to MCMC methods. The fast convergence rate is reminiscent of other deterministic methods such as Quasi Monte Carlo methods (Niederreiter, 1992; Tribble, 2007) and the Rotor-Router dynamics (Holroyd & Propp, 2010).

5.1 Future Work

Despite the success of herding in many applications of training and sampling probabilistic models, there are a lot of open theoretical questions that demand further investigation.

One of the most important questions is to identify the inductive criterion of the herded learning algorithm. We know that the moments of the sample distribution are matched to the input data, but we do not know how the remaining degrees of freedom are determined. In other words, given a set of average statistics, what criterion does herding employ to generate the distribution of samples? Can we find an objective function that the sample distribution of herding actually optimizes? Can we design a herding algorithm to implement the maximum entropy criterion? Moreover, how is the sample distribution affected by free parameters in the design of herding dynamics, such as the initialization, the relative ratio of the learning rates across features, the accuracy of the approximate maximization step, etc.

Another research direction is on the weak chaotic properties of the herding dynamics. While we have some intuitive understanding of the dynamics, a rigorous analysis of its important properties such as the form of the attractor set and the topological entropy is still missing. For example, we observe in practice that the attractor set of herding with no hidden variables is invariant to its initialization. Deriving a theoretical proof on this observation would undoubtedly provide useful guidance for better and more generic implementations of the algorithm. Generally speaking, herding reveals the connection between statistical learning and dynamical systems. A better understanding on this relationship may bring a completely new approach to machine learning.

When applied as a sampling algorithm, herding shows a distinctive advantage over Monte Carlo methods in terms of its fast convergence rate. However, the algorithms on both discrete and continuous state spaces are still in their preliminary stages and have limitations in terms of either space or time complexity. Also, the condition for herded Gibbs with a generic graph structure to converge on the target distribution is still open for future study. So is a convergence proof for kernel herding with an infinite dimensional kernel. Given the close connection between the herding sampler and Quasi Monte Carlo methods, we hope to borrow the theoretic tools from that domain for our analysis in the future.

Bibliography

Ackley, D. H, Hinton, G, and Sejnowski, T. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

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

Arbelaez, P, Maire, M, Fowlkes, C, and Malik, J. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5): 898–916, 2011.

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

Berger, A. L, Pietra, V. J. D, and Pietra, S. A. D. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Besag, J. Statistical analysis of non-lattice data. *The Statistician*, pp. 179–195, 1975.

Bialek, W, Nemenman, I, and Tishby, N. Predictability, complexity and learning. *Neural Computation*, 13:2409–2463, 2001.

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

Blake, A, Rotherx, C, Brown, M, Perez, P, and Torr, P. Interactive image segmentation using an adaptive gmmrf model. In *Proc. European Conference in Computer Vision (ECCV)*, pp. 428–441. Springer-Verlag, 2004.

Block, H and Levin, S. 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.

Bornn, L, Chen, Y, de Freitas, N, Eskelin, M, Fang, J, and Welling, M. Herded Gibbs sampling. *arXiv preprint arXiv:1301.4168*, 2013.

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

Boykov, Y and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

Boykov, Y, Veksler, O, and Zabih, R. Efficient approximate energy minimization via graph cuts. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 20(12):1222–1239, November 2001.

Brémaud, P. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer, 1999.

Breuleux, O, Bengio, Y, and Vincent, P. Unlearning for better mixing. Technical report, Tech. Rep. 1349). Montreal: Université de Montréal/DIRO, 2010.

Carbonetto, P, Kisynski, J, de Freitas, N, and Poole, D. Nonparametric Bayesian logic. In *Uncertainty in Artificial Intelligence*, pp. 85–93, 2005.

Chen, S, Dick, J, and Owen, A. B. Consistency of Markov chain quasi-Monte Carlo on continuous state spaces. *Annals of Statistics*, 39(2):673–701, 2011a.

Chen, Y and Welling, M. Parametric herding. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pp. 97–104, 2010.

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

Chen, Y, Gelfand, A, Fowlkes, C. C, and Welling, M. Integrating local classifiers through nonlinear dynamics on label graphs with an application to image segmentation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2635–2642, 2011b.

Chentsov, N. Pseudorandom numbers for modelling Markov chains. *USSR Computational Mathematics and Mathematical Physics*, 7(3):218–233, 1967.

Fang, J. Herded Gibbs sampling. Master's thesis, University of British Columbia, 2012.

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

Fulkerson, B, Vedaldi, A, and Soatto, S. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 670–677. IEEE, 2010.

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

Geyer, C. J. Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface*, pp. 156–163. Interface Foundation of North America, 1991.

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

Goodman, N, Mansinghka, V, Roy, D, Bonawitz, K, and Tenenbaum, J. Church: a language for generative models. In *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pp. 220–229, Corvallis, Oregon, 2008. AUAI Press.

He, X, Zemel, R. S, and Carreira-Perpinán, M. A. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pp. II–695. IEEE, 2004.

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

Hinton, G and Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Hinton, G. E and Zemel, R. S. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in Neural Information Processing Systems 6*, pp. 3–10. Morgan Kaufmann, 1994.

Holroyd, A. E and Propp, J. Rotor walks and Markov chains. *Algorithmic Probability and Combinatorics*, 520:105–126, 2010.

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

Jebara, T and Kondor, R. Bhattacharyya and expected likelihood kernels. In Schölkopf, B and Warmuth, M (eds.), *Conference on Computational Learning Theory (COLT)*, volume 2777 of *LNCS*, pp. 57–71, Heidelberg, Germany, 2003. Springer-Verlag.

Jelinek, F. Self-organized language modeling for speech recognition. *Readings in Speech Recognition*, pp. 450–506, 1990.

Koller, D and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Kolmogorov, V and Zabih, R. What energy functions can be minimized via graph cuts? *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004.

Kumar, S and Hebert, M. Hebert discriminative random fields. *International Journal of Computer Vision (IJCV)*, 68(2):179–201, 2006.

Lafferty, J, McCallum, A, and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pp. 282–289, 2001.

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

Li, M.-H, Lin, L, Wang, X.-L, and Liu, T. Protein–protein interaction site prediction based on conditional random fields. *Bioinformatics*, 23(5):597–604, 2007.

Lindsay, B. G. Composite likelihood methods. *Contemporary Mathematics*, 80:220–239, 1988.

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

Lunn, D. J, Thomas, A, Best, N, and Spiegelhalter, D. WinBUGS a Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, 2000.

Milch, B and Russell, S. General-purpose MCMC inference over relational structures. In *Uncertainty in Artificial Intelligence*, pp. 349–358, 2006.

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

Murphy, K. P, Weiss, Y, and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 467–475. Morgan Kaufmann Publishers Inc., 1999.

Murray, I and Elliott, L. T. Driving Markov chain Monte Carlo with a dependent random stream. *arXiv preprint arXiv:1204.3187*, 2012.

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

Niederreiter, H. Application of diophantine approximation to numerical integration. In Osgood, C. F (ed.), *Diophantine Approximation and Its Applications*, pp. 129–199, New York, 1973. Academic Press.

Niederreiter, H. *Quasi-Monte Carlo Methods*. Wiley Online Library, 1992.

Pal, C, Weinman, J, Tran, L, and Scharstein, D. On learning conditional random fields for stereo. *International Journal of Computer Vision*, pp. 1–19, 2010.

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

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

Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California, 1988.

Porteous, I, Newman, D, Ihler, A, Asuncion, A, Smyth, P, and Welling, M. Fast collapsed Gibbs sampling for latent Dirichlet allocation. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 569–577, 2008.

Quattoni, A, Wang, S, Morency, L.-P, Collins, M, and Darrell, T. Hidden conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1848 – 1852, 2007.

Rissanen, J. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989. ISBN 9971508591.

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

Salakhutdinov, R. Learning in Markov random fields using tempered transitions. *Advances in neural information processing systems*, 22:1598–1606, 2009.

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

Sha, F and Pereira, F. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 134–141. Association for Computational Linguistics, 2003.

Song, L, Zhang, X, Smola, A, Gretton, A, and Schölkopf, B. Tailoring density estimation via reproducing kernel moment matching. In McCallum, A and Roweis, S (eds.), *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pp. 992–999. Omnipress, 2008.

Stern, D. H, Graepel, T, and MacKay, D. J. Modelling uncertainty in the game of go. *Advances in Neural Information Processing Systems*, 17:1353–1360, 2004.

Sutton, C. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

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

Tappen, M. F, Liu, C, Adelson, E. H, and Freeman, W. T. Learning Gaussian conditional random fields for low-level vision. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

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

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

Tribble, S. D. *Markov chain Monte Carlo algorithms using completely uniformly distributed driving sequences*. PhD thesis, Stanford University, 2007.

van Duijn, M. A, Gile, K. J, and Handcock, M. S. A framework for the comparison of maximum pseudo-likelihood and maximum likelihood estimation of exponential family random graph models. *Social Networks*, 31(1):52–62, 2009.

Wainwright, M. J, Jaakkola, T. S, and Willsky, A. S. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Workshop on Artificial Intelligence and Statistics*, volume 21, pp. 97. Society for Artificial Intelligence and Statistics Np, 2003.

Wasserman, S and Pattison, P. Logit models and logistic regressions for social networks: I. an introduction to Markov graphs andp. *Psychometrika*, 61(3):401–425, 1996.

Welling, M, Zemel, R, and Hinton, G. Self-supervised boosting. In *Neural Information Processing Systems*, volume 15, Vancouver, Canada, 2002.

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

Welling, M. 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)*, pp. 599–606, Corvallis, Oregon, 2009b. AUAI Press.

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

Weyl, H. Über die gleichverteilung von zahlen mod. eins. *Mathematische Annalen*, 77: 313–352, 1916. ISSN 0025-5831.

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

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

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

Appendix A

Proof of Theorems in Section 4.1.3

A.1 Proof of Proposition 4.6

We prove in this section that every joint state in the support of the target distribution is visited, at least, at a linear rate. This result will be used to measure the distance between the Gibbs and herded Gibbs transition kernels.

Denote the minimum nonzero conditional probability as

$$\pi_{\min} = \min_{1 \le i \le N, \pi(x_i|\mathbf{x}_{-i})>0} \pi(x_i|\mathbf{x}_{-i}).$$

The following lemma, which is needed to prove Proposition 4.6, gives an inequality between the number of visits of two sets of states in consecutive steps.

LEMMA **A.1.** *For any integer $i \in [1, N]$ and two sets of states $\mathbb{X}, \mathbb{Y} \subseteq \mathcal{X}_+$ with a mapping $F : \mathbb{X} \to \mathbb{Y}$ that satisfies the following condition:*

$$\forall \mathbf{x} \in \mathbb{X}, \mathbf{F}(\mathbf{x})_{-i} = \mathbf{x}_{-i}, \quad \cup_{\mathbf{x}\in\mathbb{X}} F(\mathbf{x}) = \mathbb{Y}, \tag{A.1}$$

we have that, for any $s \geq 0$ and $T > 0$, the number of times $\mathbb{Y}$ is visited in the set of steps $C_i = \{t = kN + i : s \leq k \leq k + T - 1\}$ is lower bounded by a function of the number of times $\mathbb{X}$ is visited in the previous steps $C_{i-1} = \{t = kN + i - 1 : s \leq k \leq k + T - 1\}$ as:

$$\sum_{t \in C_i} \mathbb{I}\left[\mathbf{X}^{(t)} \in \mathbb{Y}\right] \geq \pi_{min} \sum_{t \in C_{i-1}} \mathbb{I}\left[\mathbf{X}^{(t)} \in \mathbb{X}\right] - |\mathbb{Y}| \tag{A.2}$$

Proof. As a complement to Condition A.1, we can define F^{-1} as the inverse mapping from $\mathbb{Y}$ to subsets of $\mathbb{X}$ so that for any $\mathbf{y} \in \mathbb{Y}$, $\mathbf{x} \in F^{-1}(\mathbf{y})$, we have $\mathbf{x}_{-i} = \mathbf{y}_{-i}$, and $\cup_{\mathbf{y} \in \mathbb{Y}} F^{-1}(\mathbf{y}) = \mathbb{X}$.

Consider any state $\mathbf{y} \in \mathbb{Y}$, when $\mathbf{y}$ is visited in C_i, the weight $w_{i,\mathbf{y}_{-i}}$ is active. Let us denote the set of all the steps in $[sN + 1, s(N + T) + N]$ when $w_{i,\mathbf{y}_{-i}}$ is active by $C_i(\mathbf{y}_{-i})$, that is, $C_i(\mathbf{y}_{-i}) = \{t : t \in C_i, \mathbf{X}_{-i}^{(t)} = \mathbf{y}_{-i}\}$. Applying Lemma 2.5 we get

$$\sum_{t \in C_i} \mathbb{I}\left[\mathbf{X}^{(t)} = \mathbf{y}\right] \geq \pi(y_i|\mathbf{y}_{-i})|C_i(\mathbf{y}_{-i})| - 1 \geq \pi_{\min}|C_i(\mathbf{y}_{-i})| - 1. \tag{A.3}$$

Since the variables $\mathbf{X}_{-i}$ are not changed at steps in C_i, we have

$$|C_i(\mathbf{y}_{-i})| = \sum_{t \in C_{i-1}} \mathbb{I}\left[\mathbf{X}_{-i}^{(t)} = \mathbf{y}_{-i}\right] \geq \sum_{t \in C_{i-1}} \mathbb{I}\left[\mathbf{X}^{(t)} \in F^{-1}(\mathbf{y})\right]. \tag{A.4}$$

Combining the fact that $\cup_{\mathbf{y} \in \mathbb{Y}} F^{-1}(\mathbf{y}) = \mathbb{X}$ and summing up both sides of Equation A.3 over $\mathbb{Y}$ proves the lemma:

$$\sum_{t \in C_i} \mathbb{I}\left[\mathbf{X}^{(t)} \in \mathbb{Y}\right] \geq \sum_{\mathbf{y} \in \mathbb{Y}} \left(\pi_{\min} \sum_{t \in C_{i-1}} \mathbb{I}\left[\mathbf{X}^{(t)} \in F^{-1}(\mathbf{y})\right] - 1 \right) \geq \pi_{\min} \sum_{t \in C_{i-1}} \mathbb{I}\left[\mathbf{X}^{(t)} \in \mathbb{X}\right] - |\mathbb{Y}|. \tag{A.5}$$

□

REMARK **A.2.** *A fully connected graph is a necessary condition for the application of Lemma 2.5 in the proof. If a graph is not fully connected ($N(i) \neq -i$), a weight $w_{i,\mathbf{y}_{N(i)}}$ may be shared*

by multiple full conditioning states. In this case $C_i(\mathbf{y}_{-i})$ is no longer a consecutive sequence of times when the weight is updated, and Lemma 2.5 does not apply here.

Now let us prove Proposition 4.6 by iteratively applying Lemma A.1.

Proof of Proposition 4.6. Because the corresponding Gibbs sampler is irreducible and any Gibbs sampler is aperiodic, there exists a constant $t^* > 0$ such that for any state $\mathbf{y} \in \mathcal{X}_+$, and any step in a sweep, i, we can find a path of length t^* for any state $\mathbf{x} \in \mathcal{X}_+$ with a positive transition probability, $Path(\mathbf{x}) = (\mathbf{x} = \mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(t^*) = \mathbf{y})$, to connect from $\mathbf{x}$ to $\mathbf{y}$, where each step of the path follows the Gibbs updating scheme. For a strictly positive distribution, the minimum value of t^* is N.

Denote $\tau^* = \lceil t^*/N \rceil$ and the j^{th} element of the path $Path(\mathbf{x})$ as $Path(\mathbf{x}, j)$. We can define $t^* + 1$ subsets $S_j \subseteq \mathcal{X}_+, 0 \leq j \leq t^*$ as the union of all the j^{th} states in the path from any state in $\mathcal{X}_+$:

$$S_j = \cup_{\mathbf{x} \in \mathcal{X}_+} Path(\mathbf{x}, j)$$

By definition of these paths, we know $S_0 = \mathcal{X}_+$ and $S_{t^*} = \{\mathbf{y}\}$, and there exits an integer $i(j)$ and a mapping $F_j : S_{j-1} \to S_j, \forall j$ that satisfy the condition in Lemma A.1 ($i(j)$ is the index of the variable to be updated, and the mapping is defined by the transition path). Also notice that any state in S_j can be different from $\mathbf{y}$ by at most $\min\{N, t^* - j\}$ variables, and therefore $|S_j| \leq 2^{\min\{N, t^* - j\}}$.

Let us apply Lemma A.1 recursively from $j = t^*$ to 1 as

$$\begin{aligned}
\sum_{k=s}^{s+T-1} \mathbb{I}\left[\mathbf{X}^{(t=Nk+i)} = \mathbf{y}\right] &\geq \sum_{k=s+\tau^*}^{s+T-1} \mathbb{I}\left[\mathbf{X}^{(t=Nk+i)} = \mathbf{y}\right] \\
&= \sum_{k=s+\tau^*}^{s+T-1} \mathbb{I}\left[\mathbf{X}^{(t=Nk+i)} \in S_{t^*}\right] \\
&\geq \pi_{\min} \sum_{k=s+\tau^*}^{s+T-1} \mathbb{I}\left[\mathbf{X}^{(t=Nk+i-1)} \in S_{t^*-1}\right] - |S_{t^*}| \\
&\geq \cdots \\
&\geq \pi_{\min}^{t^*} \sum_{k=s+\tau^*}^{s+T-1} \mathbb{I}\left[\mathbf{X}^{(t=Nk+i-t^*)} \in S_0 = \mathcal{X}_+\right] - \sum_{j=0}^{t^*-1} \pi_{\min}^{j} |S_{t^*-j}| \\
&\geq \pi_{\min}^{t^*}(T - \tau^*) - \sum_{j=0}^{t^*-1} \pi_{\min}^{j} 2^{\min\{N,j\}}. \qquad \text{(A.6)}
\end{aligned}$$

The proof is concluded by choosing the constants

$$l = \pi_{\min}^{t^*}, \quad B = \tau^* \pi_{\min}^{t^*} + \sum_{j=0}^{t^*-1} \pi_{\min}^{j} 2^{\min\{N,j\}}. \qquad \text{(A.7)}$$

□

Note that the values for l and B in Equation A.7 provide a lower bound for the linear rate of Proposition 4.6. After proving Theorem 4.4, we know that the largest value for l should be $\max_{\mathbf{x}} \pi(\mathbf{x})$ or simply $\pi(\mathbf{x})$ if we allow l to depend on $\mathbf{x}$.

A.2 Proof of Proposition 4.7

The following proposition shows that $\tilde{\mathcal{T}}_T^{(\tau)}$ is an approximation to the regular Gibbs sampler's transition kernel $\mathcal{T}$ with an error of $\mathcal{O}(1/T)$.

Proof. When $\mathbf{x} \notin \mathcal{X}_+$, we have the equality $\tilde{\mathcal{T}}_{T,i}^{(\tau)}(\mathbf{x}, \mathbf{y}) = \mathcal{T}_i(\mathbf{x}, \mathbf{y})$ by definition. When $\mathbf{x} \in \mathcal{X}_+$ but $\mathbf{y} \notin \mathcal{X}_+$, then $N_{\text{den}} = 0$ (see the notation of $\tilde{\mathcal{T}}_T^{(\tau)}$ for definition of N_{den}) as $\mathbf{y}$ will never be visited and thus $\tilde{\mathcal{T}}_{T,i}^{(\tau)}(\mathbf{x}, \mathbf{y}) = 0 = \mathcal{T}_i(\mathbf{x}, \mathbf{y})$ also holds. Let us consider the entries in $\tilde{\mathcal{T}}_{T,i}^{(\tau)}(\mathbf{x}, \mathbf{y})$ with $\mathbf{x}, \mathbf{y} \in \mathcal{X}_+$ in the following.

Because $\mathbf{X}_{-i}$ is not updated at i^{th} step of every sweep, we can replace $i-1$ in the definition of N_{den} by i and get

$$N_{\text{den}} = \sum_{k=\tau}^{\tau+T-1} \mathbb{I}(\mathbf{X}_{-i}^{(kN+i)} = \mathbf{x}_{-i}).$$

Notice that the set of times $\{t = kN+i : \tau \leq k \leq \tau+T-1, \mathbf{X}_{-i}^t = \mathbf{x}_{-i})\}$, whose size is N_{den}, is a consecutive set of times when $w_{i,\mathbf{x}_{-i}}$ is updated. By Lemma 2.5, we obtain a bound for the numerator

$$\begin{aligned} &N_{\text{num}} \in [N_{\text{den}}\pi(X_i = y_i|\mathbf{x}_{-i}) - 1, N_{\text{den}}\pi(X_i = y_i|\mathbf{x}_{-i}) + 1] \Leftrightarrow \\ &|P_{T,i}^{(\tau)}(X_i = y_i|\mathbf{x}_{-i}) - \pi(X_i = y_i|\mathbf{x}_{-i})| = |\frac{N_{\text{num}}}{N_{\text{den}}} - \pi(X_i = y_i|\mathbf{x}_{-i})| \leq \frac{1}{N_{\text{den}}}. \end{aligned} \tag{A.8}$$

Also by Proposition 4.6, we know every state in $\mathcal{X}_+$ is visited at a linear rate, there hence exist constants $l > 0$ and $B > 0$, such that the number of occurrence of any conditioning state $\mathbf{x}_{-i}$, N_{den}, is bounded by

$$N_{\text{den}} \geq \sum_{k=\tau}^{\tau+T-1} \mathbb{I}(\mathbf{X}^{(kN+i)} = \mathbf{x}) \geq lT - B \geq \frac{l}{2}T, \quad \forall T \geq \frac{2B}{l}. \tag{A.9}$$

Combining equations (A.8) and (A.9), we obtain

$$|P_{T,i}^{(\tau)}(X_i = y_i|\mathbf{x}_{-i}) - \pi(X_i = y_i|\mathbf{x}_{-i})| \leq \frac{2}{lT}, \quad \forall T \geq \frac{2B}{l}. \tag{A.10}$$

Since the matrix $\tilde{\mathcal{T}}_{T,i}^{(\tau)}$ and $\mathcal{T}_i$ differ only at those elements where $\mathbf{x}_{-i} = \mathbf{y}_{-i}$, we can bound

the L_1 induced norm of the transposed matrix of their difference by

$$\begin{aligned}\|(\tilde{\mathcal{T}}_{T,i}^{(\tau)} - \mathcal{T}_i)^T\|_1 &= \max_{\mathbf{x}} \sum_{\mathbf{y}} |\tilde{\mathcal{T}}_{T,i}^{(\tau)}(\mathbf{x},\mathbf{y}) - \mathcal{T}_i(\mathbf{x},\mathbf{y})| \\ &= \max_{\mathbf{x}} \sum_{y_i} |P_{T,i}^{(\tau)}(X_i = y_i|\mathbf{x}_{-i}) - \pi(X_i = y_i|\mathbf{x}_{-i})| \\ &\leq \frac{4}{lT}, \quad \forall T \geq \frac{2B}{l} \end{aligned} \tag{A.11}$$

Observing that both $\tilde{\mathcal{T}}_T^{(\tau)}$ and $\mathcal{T}$ are multiplications of N component transition matrices, and the transition matrices, $\tilde{\mathcal{T}}_T^{(\tau)}$ and $\mathcal{T}_i$, have a unit L_1 induced norm as:

$$\|(\tilde{\mathcal{T}}_{T,i}^{(\tau)})^T\|_1 = \max_{\mathbf{x}} \sum_{\mathbf{y}} |\tilde{\mathcal{T}}_{T,i}^{(\tau)}(\mathbf{x},\mathbf{y})| = \max_{\mathbf{x}} \sum_{\mathbf{y}} P_{T,i}^{(\tau)}(X_i = y_i|\mathbf{x}_{-i}) = 1 \tag{A.12}$$

$$\|(\mathcal{T}_i)^T\|_1 = \max_{\mathbf{x}} \sum_{\mathbf{y}} |\mathcal{T}_i(\mathbf{x},\mathbf{y})| = \max_{\mathbf{x}} \sum_{\mathbf{y}} P(X_i = y_i|\mathbf{x}_{-i}) = 1 \tag{A.13}$$

we can further bound the L_1 norm of the difference, $(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})^T$. Let $P \in \mathbb{R}^N$ be any vector with nonzero norm. Using the triangular inequality, the difference of the resulting vectors after applying $\tilde{\mathcal{T}}_T^{(\tau)}$ and $\mathcal{T}$ is bounded by

$$\begin{aligned}\|P(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})\|_1 =& \|P\tilde{\mathcal{T}}_{T,1}^{(\tau)} \dots \tilde{\mathcal{T}}_{T,N}^{(\tau)} - P\mathcal{T} \dots \mathcal{T}_N\|_1 \\ \leq& \|P\tilde{\mathcal{T}}_{T,1}^{(\tau)}\tilde{\mathcal{T}}_{T,2}^{(\tau)} \dots \tilde{\mathcal{T}}_{T,N}^{(\tau)} - P\mathcal{T}_1\tilde{\mathcal{T}}_{T,2}^{(\tau)} \dots \tilde{\mathcal{T}}_{T,N}^{(\tau)}\|_1 + \\ & \|P\mathcal{T}_1\tilde{\mathcal{T}}_{T,2}^{(\tau)}\tilde{\mathcal{T}}_{T,3}^{(\tau)} \dots \tilde{\mathcal{T}}_{T,N}^{(\tau)} - P\mathcal{T}_1\mathcal{T}_2\tilde{\mathcal{T}}_{T,3}^{(\tau)} \dots \tilde{\mathcal{T}}_{T,N}^{(\tau)}\|_1 + \\ & \dots \\ & \|P\mathcal{T}_1 \dots \mathcal{T}_{N-1}\tilde{\mathcal{T}}_{T,N}^{(\tau)} - P\mathcal{T}_1 \dots \mathcal{T}_{N-1}\mathcal{T}_N\|_1 \end{aligned} \tag{A.14}$$

where the i'th term is

$$\begin{aligned}
\|P\mathcal{T}_1 \dots \mathcal{T}_{i-1}(\tilde{\mathcal{T}}_{T,i}^{(\tau)} - \mathcal{T}_i)\tilde{\mathcal{T}}_{T,i+1}^{(\tau)} \dots \tilde{\mathcal{T}}_{T,N}^{(\tau)}\|_1 &\leq \|P\mathcal{T}_1 \dots \mathcal{T}_{i-1}(\tilde{\mathcal{T}}_{T,i}^{(\tau)} - \mathcal{T}_i)\|_1 \quad \text{(Unit } L_1 \text{ norm, Eqn. A.12)} \\
&\leq \|P\mathcal{T}_1 \dots \mathcal{T}_{i-1}\|_1 \frac{4}{lT} \quad \text{(Eqn. A.11)} \\
&\leq \|P\|_1 \frac{4}{lT} \quad \text{(Unit } L_1 \text{ norm, Eqn. A.13)} \\
& \quad \text{(A.15)}
\end{aligned}$$

Consequently, we get the L_1 induced norm of $(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})^T$ as

$$\|(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})^T\| = \max_P \frac{\|P(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})\|_1}{\|P\|_1} \leq \frac{4N}{lT}, \quad \forall T \geq \frac{2B}{l}, \tag{A.16}$$

□

A.3 Proof of Theorem 4.4

When we initialize the herded Gibbs and regular Gibbs with the same distribution (see Figure 4.5), since the transition kernel of herded Gibbs is an approximation to regular Gibbs and the distribution of regular Gibbs converges to the invariant distribution, we expect that herded Gibbs also approaches the invariant distribution.

Proof of Theorem 4.4. Construct an auxiliary regular Gibbs sampling Markov chain initialized with $\pi^{(0)}(\mathbf{X}) = P_T^{(\tau)}(\mathbf{X})$ and the same scanning order as herded Gibbs. As $\eta < 1$, the Gibbs Markov chain has uniform geometric convergence rate as shown in Equation (4.14).

Also, the Gibbs Markov chain must be irreducible due to $\eta < 1$ and therefore Proposition 4.7 applies here. We can bound the distance between the distributions of herded Gibbs after one sweep of all variables, $P_T^{(\tau+1)}$, and the distribution after one sweep of regular Gibbs sampling,

$\pi^{(1)}$ by

$$d_v(P_T^{(\tau+1)} - \pi^{(1)}) = d_v(\pi^{(0)}(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})) = \frac{1}{2}\|\pi^{(0)}(\tilde{\mathcal{T}}_T^{(\tau)} - \mathcal{T})\|_1$$
$$\leq \frac{2N}{lT}\|\pi^{(0)}\|_1 = \frac{2N}{lT}, \quad \forall T \geq T^*, \tau \geq 0. \tag{A.17}$$

Now we study the change of discrepancy between $P_T^{(\tau)}$ and π as a function as τ.

Applying the triangle inequality of d_v:

$$d_v(P_T^{(\tau+1)} - \pi) = d_v(P_T^{(\tau+1)} - \pi^{(1)} + \pi^{(1)} - \pi) \leq d_v(P_T^{(\tau+1)} - \pi^{(1)}) + d_v(\pi^{(1)} - \pi)$$
$$\leq \frac{2N}{lT} + \eta d_v(P_T^{(\tau)} - \pi), \quad \forall T \geq T^*, \tau \geq 0. \tag{A.18}$$

The last inequality follows Equations (4.14) and (A.17). When the sample distribution is outside a neighborhood of π, $\mathcal{B}_{\epsilon_1}(\pi)$, with $\epsilon_1 = \frac{4N}{(1-\eta)lT}$, i.e.

$$d_v(P_T^{(\tau)} - \pi) \geq \frac{4N}{(1-\eta)lT}, \tag{A.19}$$

we get a geometric convergence rate toward the invariant distribution by combining the two equations above:

$$d_v(P_T^{(\tau+1)} - \pi) \leq \frac{1-\eta}{2} d_v(P_T^{(\tau)} - \pi) + \eta d_v(P_T^{(\tau)} - \pi) = \frac{1+\eta}{2} d_v(P_T^{(\tau)} - \pi). \tag{A.20}$$

So starting from $\tau = 0$, we have a burn-in period for herded Gibbs to enter $\mathcal{B}_{\epsilon_1}(\pi)$ in a finite number of rounds. Denote the first time it enters the neighborhood by τ'. According to the geometric convergence rate in Equations A.20 and $d_v(P_T^{(0)} - \pi) \leq 1$

$$\tau' \leq \left\lceil \log_{\frac{1+\eta}{2}}\left(\frac{\epsilon_1}{d_v(P_T^{(0)} - \pi)}\right) \right\rceil \leq \left\lceil \log_{\frac{1+\eta}{2}}(\epsilon_1) \right\rceil = \lceil \tau^*(T) \rceil. \tag{A.21}$$

After that burn-in period, the herded Gibbs sampler will stay within a smaller neighborhood, $\mathcal{B}_{\epsilon_2}(\pi)$, with $\epsilon_2 = \frac{1+\eta}{1-\eta}\frac{2N}{lT}$, i.e.

$$d_v(P_T^{(\tau)} - \pi) \leq \frac{1+\eta}{1-\eta}\frac{2N}{lT}, \quad \forall \tau > \tau'. \tag{A.22}$$

This is proved by induction:

1. Equation (A.22) holds at $\tau = \tau' + 1$. This is because $P_T^{(\tau')} \in \mathcal{B}_{\epsilon_1}(\pi)$ and following Eqn. A.18 we get

$$d_v(P_T^{(\tau'+1)} - \pi) \leq \frac{2N}{lT} + \eta\epsilon_1 = \epsilon_2 \tag{A.23}$$

2. For any $\tau \geq \tau' + 2$, assume $P_T^{(\tau-1)} \in \mathcal{B}_{\epsilon_2}(\pi)$. Since $\epsilon_2 < \epsilon_1$, $P_T^{(\tau-1)}$ is also in the ball $\mathcal{B}_{\epsilon_1}(\pi)$. We can apply the same computation as when $\tau = \tau' + 1$ to prove $d_v(P_T^{(\tau)} - \pi) \leq \epsilon_2$. So inequality (A.22) is always satisfied by induction.

Consequently, Theorem 4.4 is proved when combining (A.22) with the inequality $\tau' \leq \lceil \tau^*(T) \rceil$ in Equation(A.21). □

REMARK **A.3.** *Similarly to the regular Gibbs sampler, the herded Gibbs sampler also has a burn-in period with geometric convergence rate. After that, the distribution discrepancy is in the order of $\mathcal{O}(1/T)$, which is faster than the regular Gibbs sampler. Notice that the length of the burn-in period depends on T, specifically as a function of* $\log(T)$.

REMARK **A.4.** *Irrationality is not required to prove the convergence on a fully-connected graph.*

Proof. Since $\tau^*(T)$ is a monotonically increasing function of T, for any $T \geq T^* + \tau^*(T^*)$, we can find a number t so that

$$T = t + \tau^*(t), t \geq T^*.$$

Partition the sample sequence $S_{0,T} = \{\mathbf{X}^{(kN)} : 0 \leq k < T\}$ into two parts: the burn-in period $S_{0,\tau^*(t)}$ and the stable period $S_{\tau^*(t),T}$. The discrepancy in the burn-in period is bounded by 1 and according to Theorem 4.4, the discrepancy in the stable period is bounded by

$$d_v(\tilde{P}(S_{t,T}) - \pi) \leq \frac{\lambda}{t}.$$

Hence, the discrepancy of the whole set $S_{0,T}$ is bounded by

$$\begin{aligned}
&d_v(\tilde{P}(S_{0,T}) - \pi) = d_v\left(\frac{\tau^*(t)}{T}\tilde{P}(S_{0,\tau^*(t)}) + \frac{t}{T}\tilde{P}(S_{\tau^*(t),T}) - \pi\right) \\
&\leq d_v\left(\frac{\tau^*(t)}{T}(\tilde{P}(S_{0,\tau^*(t)}) - \pi\right) + d_v\left(\frac{t}{T}(\tilde{P}(S_{\tau^*(t),T}) - \pi\right) \\
&\leq \frac{\tau^*(t)}{T} d_v(\tilde{P}(S_{0,\tau^*(t)}) - \pi) + \frac{t}{T} d_v(\tilde{P}(S_{\tau^*(t),T}) - \pi) \\
&\leq \frac{\tau^*(t)}{T} \cdot 1 + \frac{t}{T}\frac{\lambda}{t} \leq \frac{\tau^*(T) + \lambda}{T}. \qquad \text{(A.24)}
\end{aligned}$$

□