Advanced Structured Prediction

Editors:

Sebastian Nowozin Sebastian.Nowozin@microsoft.com
Microsoft Research
Cambridge, CB1 2FB, United Kingdom

Peter V. Gehler pgehler@tuebingen.mpg.de
Max Planck Insitute for Intelligent Systems
72076 Tübingen, Germany

Jeremy Jancsary jermyj@microsoft.com
Microsoft Research
Cambridge, CB1 2FB, United Kingdom

Christoph Lampert chl@ist.ac.at
IST Austria
A-3400 Klosterneuburg, Austria

This is a draft version of the author chapter.

The MIT Press
Cambridge, Massachusetts
London, England

1 Herding for Structured Prediction

Yutian Chen yutian.chen@eng.cam.ac.uk
University of Cambridge
Cambridge, UK

Andrew E. Gelfand agelfand@uci.edu
University of California, Irvine
Irvine CA, USA

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

This chapter introduces a Herding-based approach to structured prediction tasks. Herding is a general class of learning algorithms originally introduced for learning Markov Random Fields (MRFs). We introduce Herding in the structured prediction setting and establish connections between Herding and other approaches to structured prediction, including Conditional Random Fields (CRFs), Structured Support Vector Machines (S-SVMs), Max-Margin Markov (M^3) Networks and the Structured Perceptron (SP). We also demonstrate that Herding can be used to effectively combine piecewise, locally trained conditional models into a harmonious global model. Herding does not require training of a CRF to integrate locally trained classifiers, but instead generates pseudo-samples by iterating forward a weakly chaotic dynamical system. We show that the distribution of pseudo-samples produced by Herding is well defined even when the locally trained classifiers are inconsistent and give class marginals that disagree. The Herding approach is illustrated on two different tasks: 1) image segmentation, where classifiers based on local appearance cues are combined with pairwise boundary cues; and 2) Go game prediction, where local predictors on overlapping patches are coordinated for a consistent output.

1.1 Introduction

In this chapter, we introduce a Herding-based approach to structured prediction. Herding is a general class of learning algorithms originally introduced for learning Markov random field (MRF) models (Welling, 2009) and subsequently generalized to the discriminative prediction settings (Gelfand et al., 2010). The theoretical contribution and its application on image segmentation in this chapter are mainly based on the recent work of Chen et al. (2011) that further generalized Herding to structured prediction problems. Unlike most traditional learning approaches, which try to learn a single parameter setting that minimize a suitable loss function (e.g. the negative log-likelihood), Herding produces a non-convergent sequence of parameters that can be used to make predictions. In other words, rather than separating the prediction problem into a training phase followed by a test phase, Herding can produce predictions on the test set while it iterates.

Herding bears resemblance to many of the more well known approaches to structured prediction. The Herding update rules are derived from the zero-temperature limit of the log-likelihood and in this way Herding resembles both Conditional Random Fields (CRFs) (Lafferty et al., 2001) and max-margin methods, such as Structured Support Vector Machines (S-SVMs) (Tsochantaridis et al., 2004) and Max-Margin Markov (M^3) Networks (Taskar et al., 2003). Herding can also be seen as a generalization of the Structured Perceptron (SP) (Collins, 2002). In the remainder of this section, we formalize the structured prediction task and introduce Herding by contrasting it with the more well known approaches to structured prediction.

1.1.1 Structured Prediction

In structured prediction, we are given a data set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$ drawn independently from an unknown joint probability distribution $P(\boldsymbol{x}, \boldsymbol{y})$. We seek to learn a function $f : \mathcal{X} \to \mathcal{Y}$, from input space $\mathcal{X}$ to output space $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_M$. Let $\boldsymbol{y} = (y_1, \ldots, y_M)$ and assume that each component of $\boldsymbol{y}$ is K-valued, i.e. $y_i \in \{1, \ldots, K\}$. Finally, let $\boldsymbol{y}_\alpha$ denote subsets of $\boldsymbol{y}$.

We consider learning a linear prediction rule of the following form:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \sum_{\alpha} w_\alpha \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x}), \tag{1.1}$$

where $\boldsymbol{w} = \{w_\alpha\}$ is a vector comprised of real-valued parameters $w_\alpha \in \mathbb{R}$ and each $\psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x}) : \mathcal{X} \times \mathcal{Y}_\alpha \to \mathbb{R}$ are corresponding real-valued feature functions. Note that the prediction rule is linear in the w_α's, but the feature functions may be non-linear. Throughout this chapter we will use $\hat{\boldsymbol{y}}$ to

indicate predictions made under the current parameter setting. We will also use $\boldsymbol{\psi} = \{\psi_\alpha\}$ to represent the vector of features corresponding to $\boldsymbol{w}$.

In order to clarify our notation, consider the problem of segmenting an image. In that task, each y_i is a pixel taking one of K labels (e.g. $y_i = \text{Airplane}$ or $y_i = \text{Boat}$). We construct a planar graph, $G = (V, E)$, over the image by associating each pixel with a vertex and adding an edge between adjacent pixels. We might then introduce unary features, $\psi_i(y_i, \boldsymbol{x})$, for each pixel $i \in V$ and pairwise features, $\psi_{ij}(y_i, y_j, \boldsymbol{x})$, for each edge $e = (i, j) \in E$. Doing so would give us a problem with a total of $|V| + |E|$ parameters.

In the standard learning scenario, our goal is to use the data set $\mathcal{D}$ to find the setting of parameters $\boldsymbol{w}$ that 'best' predicts outputs given inputs. The quality of the parameter setting is assessed via a loss function that measures the error incurred by predicting $\hat{\boldsymbol{y}} = f(\boldsymbol{x}, \boldsymbol{w})$ when the true output is $\boldsymbol{y}$. Let $\ell(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}, \boldsymbol{w})$ denote the loss incurred on training point n given the parameter setting $\boldsymbol{w}$. The goal is then to find the setting $\boldsymbol{w}^\star$ with minimal empirical loss,

$$\boldsymbol{w}^\star = \operatorname*{argmin}_{\boldsymbol{w}} \mathcal{L}(\mathcal{D}, \boldsymbol{w}) = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}, \boldsymbol{w}). \tag{1.2}$$

Several methods for finding $\boldsymbol{w}^\star$ have been proposed, including Conditional Random Fields (CRFs) (Lafferty et al., 2001) which minimize the negative log-likelihood (or log-loss), Structured Support Vector Machines (S-SVMs) (Tsochantaridis et al., 2004) and Max-Margin Markov (M^3) Networks (Taskar et al., 2003) which minimize differently scaled versions of the hinge loss and Structured Perceptrons (SPs) (Collins, 2002) which minimize the (generalized) perceptron loss. As previously mentioned, Herding does not seek a single setting $\boldsymbol{w}^\star$, but rather produces a sequence of parameters $\ldots, \boldsymbol{w}^{t-1}, \boldsymbol{w}^t, \boldsymbol{w}^{t+1}, \ldots$ that can be used to predict a sequence of outputs $\ldots, \hat{\boldsymbol{y}}^{t-1}, \hat{\boldsymbol{y}}^t, \hat{\boldsymbol{y}}^{t+1}, \ldots$ for each input. In this way, Herding closely resembles the SP. However, Herding's update rules can be derived from a zero-temperature variant of the log-loss. We introduce Herding from this latter perspective.

1.1.2 Herding and the Zero Temperature Log-Loss

Consider a conditional Gibbs distribution of the form

$$p_\tau(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z_\tau(\boldsymbol{x}, \boldsymbol{w})} \exp\left(\frac{1}{\tau} \sum_\alpha w_\alpha \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x})\right), \tag{1.3}$$

where τ is a temperature parameter and $Z_\tau(\boldsymbol{x}, \boldsymbol{w})$ is the partition function

$$Z_\tau(\boldsymbol{x}, \boldsymbol{w}) = \sum_{y' \in \mathcal{Y}} \exp\left(\frac{1}{\tau} \sum_\alpha w_\alpha \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x})\right). \tag{1.4}$$

This conditional distribution in (1.3) is referred to as a CRF when $\tau = 1$, each y_i of $\boldsymbol{y}$ is associated with a vertex in some graph $G = (V, E)$ and each y_i satisfies the Markov property with respect to the graph G.

A common objective of learning is to find the model $p_\tau(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ that maximizes the probability of observing the outputs $\{\boldsymbol{y}^{(n)}\}_{n=1}^N$ given the inputs $\{\boldsymbol{x}^{(n)}\}_{n=1}^N$. When formulated as a minimization problem, this gives rise to the negative log-likelihood objective

$$\boldsymbol{w}^\star_{\tau,LL} = \operatorname*{argmin}_{\boldsymbol{w}} \mathcal{L}_{\tau,\mathrm{LL}}(\mathcal{D}, \boldsymbol{w}) = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^N \ell_{\tau,\mathrm{LL}}(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}, \boldsymbol{w}), \tag{1.5}$$

where the log-loss function is defined as:

$$\ell_{\tau,\mathrm{LL}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = -\tau \log p_\tau(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_\alpha w_\alpha \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x}) + \tau \log Z_\tau(\boldsymbol{x}, \boldsymbol{w}). \tag{1.6}$$

The subscript τ in $\ell_{\tau,\mathrm{LL}}$ makes the dependence on temperature explicit.

Since the log partition function, $\log Z_\tau(\boldsymbol{x}, \boldsymbol{w})$, is convex in $\boldsymbol{w}$, the negative log-likelihood is a convex function of $\boldsymbol{w}$. Finding its minimum is possible using standard numerical optimization methods (e.g. limited memory BFGS) when the log-loss and its gradients can be computed efficiently. The full gradient updates are

$$w_\alpha^t = w_\alpha^{t-1} + \eta_{\alpha,t} \left(\mathbb{E}_{\hat{P}}[\psi_\alpha] - \frac{1}{N} \sum_n \sum_{\boldsymbol{y}'} p_\tau(\boldsymbol{y}'|\boldsymbol{x}^{(n)}, \boldsymbol{w}^{t-1}) \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x}^{(n)})\right), \tag{1.7}$$

where $\eta_{\alpha,t}$ is a decreasing step size and $\mathbb{E}_{\hat{P}}[\psi_\alpha]$ is the empirical average value of feature ψ_α in the training data computed as

$$\mathbb{E}_{\hat{P}}[\psi_\alpha] = \frac{1}{N} \sum_{n=1}^N \psi_\alpha(\boldsymbol{y}_\alpha^{(n)}, \boldsymbol{x}^{(n)}). \tag{1.8}$$

Note that the setting $\boldsymbol{w}^\star_{\tau,LL}$ that minimizes (1.5) has the appealing property that the empirical average feature, $\mathbb{E}_{\hat{P}}[\psi_\alpha]$, will equal the average of that feature under the model - the well known moment matching property of maximum likelihood estimation.

The Herding loss is revealed by taking the zero-temperature limit, $\tau \to 0$, of (1.6)(Welling, 2009):

$$\ell_{\text{Herd}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = -\sum_{\alpha} w_\alpha \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x}) + \max_{\boldsymbol{y}'} \left[\sum_{\alpha} w_\alpha \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x}) \right]. \quad (1.9)$$

Several observations about this loss function are made in Welling (2009), including the fact that it has a unique minima at $\boldsymbol{w} = 0$. As a result, it would seem pointless to minimize $\mathcal{L}_{\text{Herd}}(\mathcal{D}, \boldsymbol{w})$ by applying subgradient updates. However, this is exactly what Herding does! In particular, Herding iteratively applies the following updates:

$$\hat{\boldsymbol{y}}^{(n),t} = \operatorname*{argmax}_{\boldsymbol{y}'} \sum_{\alpha} w_\alpha^{t-1} \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x}^{(n)}) \text{ for } n = 1 \dots N, \quad (1.10)$$

$$w_\alpha^t = w_\alpha^{t-1} + \eta_\alpha \left(\mathbb{E}_{\hat{P}}[\psi_\alpha] - \frac{1}{N} \sum_{n} \psi_\alpha(\hat{\boldsymbol{y}}_\alpha^{(n),t}, \boldsymbol{x}^{(n)}) \right). \quad (1.11)$$

In Herding, the step-size η_α is held fixed (note the lack of dependence on t). Given suitable initialization of $\boldsymbol{w}^0$, this prevents convergence of the sequence to the trivial minima at $\boldsymbol{w} = 0$. Moreover, the sequence will be non-convergent if at least one incorrect prediction is made in every iteration - i.e. $\hat{\boldsymbol{y}}^{(n),t} \neq \boldsymbol{y}^{(n)}$ for at least one data point n in every iteration t. The sequence $\dots, \boldsymbol{w}^{t-1}, \boldsymbol{w}^t, \boldsymbol{w}^{t+1}, \dots$ will also not diverge as long as an easy to check criteria is satisfied (see 1.1.4 for more on this criteria). Last, and most importantly, the average over the sequence of features produced by Herding will *match* the empirical average of features in the following sense (Gelfand et al. (2010)):

$$\left| \mathbb{E}_{\hat{P}}[\psi_\alpha] - \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N} \sum_{n} \psi_\alpha(\hat{\boldsymbol{y}}_\alpha^{(n),t}, \boldsymbol{x}^{(n)}) \right| = \mathcal{O}\left(\frac{1}{T}\right) \quad \forall \alpha. \quad (1.12)$$

Thus, Monte Carlo averages over the sequence of states produced by Herding will match the moments of the training data as if they were sampled directly from $\hat{P}$ - albeit at a faster rate than independent sampling from $\hat{P}$. We will refer to these states as *pseudo-samples* from now on, as the Herding algorithm is a deterministic procedure rather than a random sampling algorithm. This matching property will be discussed in more detail in Section 1.2, so we defer further discussion until that time.

1.1.3 Herding and Max-Margin Methods

The model, $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}^\star)$, that is learned by minimizing the log-loss does not utilize the knowledge that it will be used to make predictions under the

MAP prediction rule in (1.1). Training in this agnostic manner is justified by Bayesian decision theory when the learned model, $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}^\star)$, closely resembles the true (and unknown) distribution, $P(\boldsymbol{y}|\boldsymbol{x})$, that our data was drawn from. If $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}^\star)$ differs greatly from $P(\boldsymbol{y}|\boldsymbol{x})$, then it may be beneficial to find parameters that directly optimize the MAP prediction rule. This is exactly the aim of max-margin methods, such as S-SVMs and M^3 networks.

We follow the approach of Pletscher et al. (2010) in introducing max-margin methods as it facilitates comparison with Herding. In max-margin methods, one tries to minimize the following loss:

$$\ell_{\text{MM}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = -\sum_\alpha w_\alpha \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x}) + \max_{y'} \left[\sum_\alpha w_\alpha \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x}) + \Delta(\boldsymbol{y}', \boldsymbol{y}) \right], \tag{1.13}$$

where $\Delta(\boldsymbol{y}', \boldsymbol{y})$ is a function that provides a margin between incorrect predictions ($\boldsymbol{y}'$) and the ground truth output ($\boldsymbol{y}$). In S-SVMs, the margin function might take the form:

$$\Delta(\boldsymbol{y}', \boldsymbol{y}) = \begin{cases} 0, & \text{if } \boldsymbol{y}' = \boldsymbol{y}, \\ 1, & \text{otherwise.} \end{cases}$$

In M^3 Networks the margin function decomposes over the components of $\boldsymbol{y}$. In either case, we see that the max-margin loss is equivalent to the Herding loss in (1.9), when the margin function is $\Delta(\boldsymbol{y}', \boldsymbol{y}) = 0$.

S-SVMs are commonly trained via subgradient updates that closely resemble the Herding updates

$$\begin{aligned} \hat{\boldsymbol{y}}^{(n),t} &= \operatorname*{argmax}_{y'} \sum_\alpha w_\alpha^{t-1} \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x}^{(n)}) + \Delta(\boldsymbol{y}', \boldsymbol{y}^{(n)}) \;\text{ for }\; n = 1 \dots N, \\ w_\alpha^t &= w_\alpha^{t-1} + \eta_{\alpha,t} \left(\mathbb{E}_{\hat{P}}\left[\psi_\alpha\right] - \frac{1}{N} \sum_n \psi_\alpha(\hat{\boldsymbol{y}}_\alpha^{(n),t}, \boldsymbol{x}^{(n)}) \right). \end{aligned} \tag{1.14}$$

There are two primary differences between these updates and the Herding updates in (1.10,1.11). The first change is the presence of the margin function $\Delta(\boldsymbol{y}', \boldsymbol{y})$ in the S-SVM updates. The second change is the dependence of the S-SVM step-size on t. In S-SVMs, the step size is steadily decreased so as to find a single parameter setting $\boldsymbol{w}^\star$. Herding holds the step size fixed across all t, which prevents convergence of the sequence of parameters (so long as an incorrect prediction is made on the training data in every iteration).

1.1.4 Herding and the Structured Perceptron

The Structured Perceptron (SP) was introduced by Collins for sequence labeling problems, such as the aforementioned POS tagging task (Collins, 2002). As its name suggests, it extends Rosenblatt's classic perceptron learning algorithm (Rosenblatt, 1958). The SP is delightful because of its simplicity. At every iteration t, choose a data point n_t from our data set[1] and apply the following update rule:

$$\begin{aligned}\hat{\boldsymbol{y}}^{(n_t)} &= \operatorname*{argmax}_{\boldsymbol{y}'} \sum_{\alpha} w_{\alpha}^{t-1} \psi_{\alpha}(\boldsymbol{y}'_{\alpha}, \boldsymbol{x}^{(n_t)}), \\ w_{\alpha}^{t} &= w_{\alpha}^{t-1} + \left(\psi_{\alpha}(\boldsymbol{y}_{\alpha}^{(n_t)}, \boldsymbol{x}) - \psi_{\alpha}(\hat{\boldsymbol{y}}_{\alpha}^{(n_t)}, \boldsymbol{x})\right). \end{aligned} \tag{1.15}$$

Note that if $\hat{\boldsymbol{y}} = \boldsymbol{y}$, then $w_{\alpha}^{t} = w_{\alpha}^{t-1}$ and our parameters do not change.

Written in this form, the SP differs from Herding only in that it is an online algorithm - updating on the training data one-by-one, rather than on the entire training set. However, Herding generalizes the SP in a few important ways not immediately clear from the current presentation. First, in many applications (e.g. image segmentation) it is not possible to exactly find the MAP state and one is forced to find an approximation to the MAP configuration. Approximate inference, it turns out, is non-problematic and Herding will still satisfy the moment matching property in (1.12) so long as a simple to check condition from the Perceptron Cycling Theorem (PCT) is satisfied (Gelfand et al., 2010). This PCT condition also justifies the use of mini-batch updates that use a subset of the training data (including online schemes with batch size of one). We describe the PCT condition now as its introduction will aid understanding of forthcoming results.

The Perceptron Cycling Theorem is a classic result due to (Minsky and Papert, 1969; Block and Levin, 1970), which states:

Theorem 1.1. *The sequence of (parameter) vectors* $\ldots, \boldsymbol{w}^{t-1}, \boldsymbol{w}^{t}, \boldsymbol{w}^{t+1}, \ldots$ *generated using the iterative procedure* $\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} + \boldsymbol{v}^{t}$ *remains bounded (i.e.* $\left\|\boldsymbol{w}^{t}\right\|_2 < \left\|\boldsymbol{w}^{0}\right\|_2 + M$ *for some constant* $M > 0$*) if:*

1. *The domain of* $\boldsymbol{v}^{t}$ *is a finite set* $\mathbf{V}$*;*
2. *The norm of* $\boldsymbol{v}^{t}$ *is bounded - i.e.* $\max_{\boldsymbol{v}' \in \mathbf{V}} \left\|\boldsymbol{v}'\right\|_2 < \infty$*; and*
3. *In every iteration* t, $\langle \boldsymbol{w}^{t}, \boldsymbol{v}^{t} \rangle \leq 0$.

1. Note that a data point can be picked multiple times.

As a consequence, if the PCT holds then it can be shown that

$$\left\| \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{v}^t \right\|_2 = \mathcal{O}\left(\frac{1}{T}\right), \tag{1.16}$$

which perhaps unsurprisingly has the form of the Herding moment matching property in (1.12).

The PCT can be applied to Herding by identifying the update vectors as

$$\boldsymbol{v}^t = \left(\mathbb{E}_{\hat{P}}\left[\boldsymbol{\psi}\right] - \frac{1}{N} \sum_{n} \boldsymbol{\psi}(\hat{\boldsymbol{y}}_{\alpha}^{(n),t}, \boldsymbol{x}^{(n)}) \right),$$

where as usual $\hat{\boldsymbol{y}}^{(n),t} = \text{argmax}_{\boldsymbol{y}' \in \mathcal{Y}} \sum_{\alpha} w_{\alpha}^{t-1} \psi_{\alpha}(\boldsymbol{y}'_{\alpha}, \boldsymbol{x}^{(n)})$. The domain of the update vectors is a finite set $\mathbf{V}$ because in every iteration there are an exponential number of configurations $\hat{\boldsymbol{y}}^{(n),t} \in \mathcal{Y}$ for each of the N data points. The norm of $\boldsymbol{v}^t$ will also be bounded if the feature functions are appropriately specified. The easy-to-check criteria referred to throughout this chapter is the third condition of the PCT - namely, that the inner product of the parameter and update vectors are less than or equal to zero. As long as this condition is satisfied in every iteration, we can use approximate inference or mini-batches when applying the Herding updates and still satisfy the Herding moment matching property of (1.12).

Herding also generalizes the SP in another important way. Since the Herding updates are derived by taking the zero temperature limit of the log-loss on a CRF model, the entire derivation can be repeated for a CRF model with hidden (unobserved) variables. The resulting algorithm can be viewed as a SP with hidden units that, much like discriminative Restricted Boltzmann Machines (RBMs), can capture complex interactions among the observed variables (see Gelfand et al. (2010) for more detail).

1.2 Integrating Local Models using Herding

1.2.1 Piecewise Trained CRFs

Conditional Random Fields (CRFs) are a standard approach for combining local features into a global conditional distribution over labels for structured prediction. To specify a CRF model, one must first define feature functions $\psi_{\alpha}(\boldsymbol{y}_{\alpha}, \boldsymbol{x})$, where $\boldsymbol{y}_{\alpha}$ is a subset of the labels associated with ψ_{α} and $\boldsymbol{x}$ are inputs. For example, in the image segmentation task, one might define a

feature of the form:

$$\psi_i(y_i, y_j, x_i) = \begin{cases} 1 & \text{if } y_i = y_j \text{ AND } x_i > 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{1.17}$$

This feature encodes a preference for the label of pixel i to agree with the label of adjacent pixel j when the intensity at pixel i is greater than 0.5. The label subsets indexed by α can overlap, forming a loopy graph over the labels $\boldsymbol{y}$. In this way, the CRF framework can utilize very rich features.

As discussed in the previous section, such features can be incorporated into a probabilistic model by associating model parameters $\{w_\alpha\}$ with each feature function $\{\psi_\alpha\}$:

$$p_{\text{CRF-1}}(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{w})} \exp\left(\sum_\alpha w_\alpha \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{x}_\alpha)\right). \tag{1.18}$$

The model parameters can be learned from data by, for example, minimizing the log-loss described in (1.5). While this is quite elegant, it poses a practical problem in that each feature function ψ_α has a distinct parameter w_α to be estimated. In typical image labeling problems, there may be thousands of parameters to learn (e.g. one for every pixel and pair of pixels in every image). In such cases, there may be less than one pixel of information per parameter, which may lead to extreme over-fitting.

A common remedy for the explosion of parameters is to train the CRF model in stages. In the first stage, we train a set of local discriminative models (i.e. classifiers) and then in the second stage we integrate the information from the local classifiers to make a coherent prediction. For example, in the image segmentation task, we might train a *unary* probability model $p_i(y_i|x_i)$ that predicts the label at pixel i and a *pairwise* classifier that predicts the probability that two adjacent pixels i and j have different labels $p_{ij}(y_i \neq y_j|x_i, x_j)$ and form a boundary.

One of the main questions addressed in the remainder of this chapter is how to effectively integrate the information of local, piecewise trained (Sutton and McCallum, 2007), discriminative probability models, $p_\alpha(\boldsymbol{y}_\alpha|\boldsymbol{x}_\alpha)$. A common approach is to incorporate the local classifiers by taking their log probabilities as feature functions and combine them using a small number of parameters to balance their respective local information.

This is exactly the approach adopted in Fulkerson et al. (2010), where the unary and pairwise classifiers are incorporated using feature functions $\psi_i(y_i, \boldsymbol{x}) = -\log p_i(y_i|x_i)$ and $\psi_{ij}(y_i, y_j, \boldsymbol{x}) = -\log p_{ij}(y_i \neq y_j|x_i, x_j)$. A single parameter λ is introduced to trade the relative importance of the

unary and pairwise classifiers. This yields the following CRF model:

$$p_{\text{CRF-2}}(\boldsymbol{y}|\boldsymbol{x},\lambda) = \frac{1}{Z(\boldsymbol{x},\lambda)} \exp\left(\sum_i \log(p_i(y_i|x_i)) + \lambda \sum_{i,j} \log(p_{ij}(y_i \neq y_j|x_i,x_j))\right). \tag{1.19}$$

In Fulkerson et al. (2010), the unary $p_i(y_i|x_i)$ and pairwise classifiers $p_{ij}(y_i \neq y_j|x_i,x_j)$ are trained independently. As a result, there is no longer reason to expect that the learned model's moments, $\mathbb{E}_{p_{\text{CRF-2}}}[\psi_\alpha]$ will match the moments of the training data, $\mathbb{E}_{\hat{P}}[\psi_\alpha]$, in the following sense (see Section 1.1.2):

$$\mathbb{E}_{p_{\text{CRF-2}}}[\psi_\alpha] = \mathbb{E}_{\hat{P}}[\psi_\alpha]. \tag{1.20}$$

This is simply because we have only a single parameter to tune, but need to satisfy a large collection of moment constraints (number of pixels plus number of neighboring pairs of pixels).

Instead, we might insist that our global model at least approximately match the moment constraints

$$\mathbb{E}_p[\psi_\alpha] = \mathbb{E}_{p_\alpha}[\psi_\alpha], \tag{1.21}$$

where the joint empirical distribution $\hat{P}$ has been replaced with the local discriminative model p_α. For features of the form $\psi_{\alpha,\boldsymbol{z}_\alpha}(\boldsymbol{y}_\alpha) = \mathbb{I}[\boldsymbol{y}_\alpha = \boldsymbol{z}_\alpha]$, this condition implies that the joint distribution has marginals consistent with each local probability model

$$\sum_{\boldsymbol{y}\setminus\boldsymbol{y}_\alpha} p(\boldsymbol{y}|\boldsymbol{x}) = p_\alpha(\boldsymbol{y}_\alpha|\boldsymbol{x}_\alpha). \tag{1.22}$$

In the ongoing image segmentation example, this consistency condition means that $p(y_i|\boldsymbol{x}) = p_i(y_i|x_i)$ and $p(y_i,y_j|\boldsymbol{x}) = p_{ij}(y_i,y_j|x_i,x_j)$. However, with independently trained local classifiers, no joint model can achieve this as the p_α will likely be mutually *inconsistent*.

The Herding approach we describe in the next section provides an elegant solution to this problem. Given piecewise trained discriminative models, it produces a sequence of states $\ldots \hat{\boldsymbol{y}}^t, \hat{\boldsymbol{y}}^{t+1} \ldots$ that on average satisfy the marginalization condition in (1.21) when the local models are consistent. And if the local models are *inconsistent*, we show that the same procedure produces a sequence whose average behavior matches that of the closest consistent model. We thus gain some of the flexibility of the general CRF formulation of (1.18) in matching moments, while retaining the parsimony of piecewise training local discriminative models.

1.2.2 Herding Local Models

The Herding approach we advocate attempts to identify a joint probability distribution over features ψ_α that approximately marginalize to the average features under the local models $\mathbb{E}_{p_\alpha}[\psi_\alpha]$. Let us first assume that the set of p_α are in fact consistent. For example, in the image segmentation task this means that the pairwise probability model marginalizes down to the unary model: $\sum_{y_j} p_{ij}(y_i, y_j|x_i, x_j) = p_i(y_i|x_i)$, for all settings of y_i and adjacent pixels (i, j).

Consider the following Herding updates:

$$\hat{\boldsymbol{y}}^t = \operatorname*{argmax}_{\boldsymbol{y}'} \sum_\alpha w_\alpha^{t-1} \psi_\alpha(\boldsymbol{y}'_\alpha, \boldsymbol{x}_\alpha), \tag{1.23}$$

$$w_\alpha^t = w_\alpha^{t-1} + \eta_\alpha \left(\mathbb{E}_{p_\alpha}[\psi_\alpha] - \psi_\alpha(\hat{\boldsymbol{y}}_\alpha^t, \boldsymbol{x}_\alpha) \right). \tag{1.24}$$

These are the same as the update equations in (1.10, 1.11) except that the empirical distribution $\hat{P}$ is replaced by the local model p_α and every image is processed independently. In the applications that follow, we use $\psi_{\alpha,\boldsymbol{z}_\alpha} = \mathbb{I}[\boldsymbol{y}_\alpha = \boldsymbol{z}_\alpha]$, i.e. a unique feature for every state $\boldsymbol{z}_\alpha$ in every region.

It can now be shown (Gelfand et al., 2010) that if in every iteration we satisfy the Perceptron Cycling Theorem condition (see 1.1.4)

$$C^t = \sum_\alpha w_\alpha^{t-1} \left(\mathbb{E}_{p_\alpha}[\psi_\alpha] - \psi_\alpha(\hat{\boldsymbol{y}}_\alpha^t, \boldsymbol{x}_\alpha)) \right) \leq 0, \tag{1.25}$$

then it follows that

$$\left| \mathbb{E}_{p_\alpha}[\psi_\alpha] - \frac{1}{T} \sum_{t=1}^{T} \psi_\alpha(\hat{\boldsymbol{y}}_\alpha^t, \boldsymbol{x}_\alpha) \right| = \mathcal{O}\left(\frac{1}{T}\right). \tag{1.26}$$

We note that this convergence rate is much faster than the Monte Carlo average computed by independently sampling from p_α, which would converge at a rate of $\mathcal{O}(\sqrt{1/T})$.

Herding's updates generate sequences $\ldots, (\boldsymbol{w}^t, \hat{\boldsymbol{y}}^t), (\boldsymbol{w}^{t+1}, \hat{\boldsymbol{y}}^{t+1}), \ldots$ of parameters and states in such a way that the states come from some joint distribution $P(\boldsymbol{y}|\boldsymbol{x})$ which has moments $\mathbb{E}_{p_\alpha}[\psi_\alpha]$. Unlike CRF models, the entropy of this joint model is not expected to be maximal, although empirically it is often close. Perhaps surprisingly, for many problems local maximizations are often sufficient to satisfy condition (1.25), allowing us to side-step hard inference. It should be noted that this is not always the case, in particular when the constraints are hard or impossible to satisfy as may arise for image segmentation.

We also emphasize that the dynamical system defined by equations

(1.23) and (1.24) do not return a parameterized model. The sequence $\ldots(\boldsymbol{w}^t, \hat{\boldsymbol{y}}^t), (\boldsymbol{w}^{t+1}, \hat{\boldsymbol{y}}^{t+1}), \ldots$ never converges to a fixed point and one should rather think of this as a deterministic process to generate "representative points". In fact, it can be shown that the dynamical system is weakly chaotic (Welling and Chen, 2010) meaning that the sequence over $\hat{\boldsymbol{y}}^t$ is not periodic and is insensitive to the initial setting of $\boldsymbol{w}^0$.

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.

The difference between the Herding approach to integrating locally-trained models and the CRF-based approach adopted in Fulkerson et al. (2010) should now be clear. In the Herding approach, one iterates the Herding updates for several iterations, where each iteration requires finding the approximate MAP configuration for a subset of training images and then verifying that the resulting parameter update satisfies the PCT condition. In every iteration, one also makes predictions on each image in the test set. In contrast, in the CRF-based approach, one invests up-front time to identify a setting of the λ parameters that best balance the information from each local classifier. After finding such a parameter setting, a prediction can be made once on each image in the test set. The Herding approach is thus best suited for situations where the test set is known in advance and where it is relatively *easy* to perform MAP inference.

1.2.3 Herding with Inconsistent Marginals

We now describe how to handle inconsistent marginals in Herding. When the vector of $\mathbb{E}_{p_\alpha}[\psi_\alpha]$'s does not reside inside the marginal polytope $\mathcal{M} \stackrel{\text{def}}{=} \{\mathbb{E}_p[\boldsymbol{\psi}_\alpha] | \forall p\}$, then by definition there does not exist a joint distribution $p(\boldsymbol{y}|\boldsymbol{x})$ with moments $\{\mathbb{E}_{p_\alpha}[\psi_\alpha]\}$. If we want to train a CRF without regularization, the parameters will diverge. For Herding this means that the condition in equation (1.25) cannot always be satisfied, and the norm of parameters $\boldsymbol{w}^t$ will also linearly diverge. Nevertheless, we can still obtain a stationary joint distribution of states $\hat{\boldsymbol{y}}^t$ from the Herding sequence. The potential numerical problems caused by the divergence of $\boldsymbol{w}^t$ can be easily prevented by taking an additional normalization step, $\boldsymbol{w} \leftarrow \boldsymbol{w}/M, \eta_\alpha \leftarrow \eta_\alpha/M$, for some constant M. This global scaling will not affect the state sequence $\{\hat{\boldsymbol{y}}^t\}$ in any way. The most important consequence of inconsistent marginals is that the moments of the joint distribution do not converge to $\mathbb{E}_{p_\alpha}[\psi_\alpha]$ any more. Instead, we prove in this chapter that the moments orthogonally project onto the marginal polytope.

In the following, we will denote the collection of expectations $\mathbb{E}_{p_\alpha}[\psi_\alpha]$ as $\bar{\boldsymbol{\psi}}$ and the sample average of the features generated by Herding up to time T as $\tilde{\boldsymbol{\psi}}^T = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\psi}(\hat{\boldsymbol{y}}^t, \boldsymbol{x})$. We now claim that the following property holds:

Proposition 1.2. *Assume $\bar{\boldsymbol{\psi}}$ is outside the marginal polytope $\mathcal{M}$ and the stepsize η_α is constant. Let $\bar{\boldsymbol{\psi}}_{\mathcal{M}}$ be the L_2 projection of $\bar{\boldsymbol{\psi}}$ onto $\mathcal{M}$. Then the average features of Herding $\tilde{\boldsymbol{\psi}}^T$ converge to $\bar{\boldsymbol{\psi}}_{\mathcal{M}}$ at the rate of $1/T$.*

For a proof of this proposition and the following corollary see the appendix of Chen et al. (2011).

When η_α depends on the feature index α, we can construct an equivalent Herding sequence with a constant stepsize and new features $\{\sqrt{\eta_\alpha}\psi_\alpha\}$. Then Proposition 1.2 still applies except that the L_2 distance is weighted by $\sqrt{\eta_\alpha}$. In this way, the step sizes control the relative importance of features. When we consider features of the form $\psi_{\alpha,\boldsymbol{z}_\alpha}(\boldsymbol{y}_\alpha) = \mathbb{I}[\boldsymbol{y}_\alpha = \boldsymbol{z}_\alpha]$, the marginal probabilities of Herding pseudo-samples will converge to the closest consistent marginals in $\mathcal{M}$.

As an immediate consequence of Proposition 1.2, Herding always improves upon an initial set of moments $\bar{\boldsymbol{\psi}}$ in the following sense:

Corollary 1.3. *Given a feature vector $\bar{\boldsymbol{\psi}}$ that is an approximation to the expected feature w.r.t. some unknown distribution, $\bar{\boldsymbol{\psi}}_{true}$. When running Herding dynamics with $\bar{\boldsymbol{\psi}}$, the limit of the empirical average of features will not increase the L_2 error. Specifically, $\|\bar{\boldsymbol{\psi}}_{\mathcal{M}} - \bar{\boldsymbol{\psi}}_{true}\|_2 < \|\bar{\boldsymbol{\psi}} - \bar{\boldsymbol{\psi}}_{true}\|_2$ when $\bar{\boldsymbol{\psi}} \notin \mathcal{M}$, and $\tilde{\boldsymbol{\psi}}^T \to \bar{\boldsymbol{\psi}}$ otherwise.*

1.3 Application: Image Segmentation

Piecewise training approaches are fairly common in computer vision tasks, such as image segmentation, because the vision community has spent considerable time developing state-of-the-art, highly specialized classifiers (e.g. edge or human detectors). Such classifiers are trained independently, often on disparate data sets, and utilize different input features (e.g. color, texture, etc). As a result, the marginals produced by such classifiers are likely to be inconsistent. This provides an ideal setting in which to demonstrate the approximate marginal consistency property of Herding.

In this section, we consider the image segmentation task, where our goal is to produce a labeling $\boldsymbol{y}$ for each pixel of an input image $\boldsymbol{x}$. In the simplest case, the pixel labels y_i are binary and indicate whether a pixel is foreground or background. In a more complex setting, the y_i may take one of K different class labels (e.g. Boat, Airplane, Sky,. . .). Since a typical image contains tens

of thousands of pixels, it is fairly common to pre-process an image and group neighboring pixels into super-pixels. The result of this pre-processing is a CRF model with a far more manageable number of variables, one for each super-pixel.

We consider incorporating the output of two local classifiers. One of the classifiers provides unary conditional probabilities $\{p_i(y_i|x_i)\}$ on each super-pixel; the other classifier provides pairwise conditional probabilities $\{p_{ij}(y_i \neq y_j|x_i, x_j)\}$ on neighboring super-pixels. The former gives the probability that super-pixel i is of a particular class (e.g. $y_i =$ 'Boat'); the latter suggests the existence of boundaries.

We adopt the approach of Fulkerson et al. (2010) and use the CRF defined in equation (1.19) with a single parameter λ. The best value of λ is estimated on a validation set using grid search and segmentations are predicted by finding the $\hat{\boldsymbol{y}}$ that maximizes $p_{\text{CRF-2}}(\boldsymbol{y}|\boldsymbol{x}, \lambda^\star)$.

Our Herding algorithm follows equations (1.23) and (1.24) with two types of features: $\mathbb{I}(y_i = k)$ and $\mathbb{I}(y_i \neq y_j)$. The step size η_α is scale free in the sense that multiplying all η_α by the same factor doesn't change the output of label sequence $\boldsymbol{y}^t$, and so without loss of generality we set the step size for unary features to 1 and the step size for pairwise features to λ. The value of λ is used to trade off the strength of these two sources of information. The segmentations predicted by Herding are obtained by taking the most frequently occurring class label for each super-pixel in the Herding sequence

$$y_i^* = \operatorname*{argmax}_k \sum_{t=1}^{T} \mathbb{I}[y_i^t = k] \quad \forall i. \tag{1.27}$$

Notice that the role of the parameter λ is different in the CRF and Herding approaches. In the CRF model, 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 the dataset of the PASCAL VOC 2007 segmentation competition. We compare Herding's predictions to those made by a multi-class classifier using local appearance cues only and to the traditional CRF approach of Fulkerson et al. (2010). Interested readers can refer to Chen et al. (2011) for a comparison on the GrabCut data set.

On the PASCAL VOC 2007 dataset, we follow a similar experiment 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 at most N. The larger N is, the more context information is available for the local classifier, 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 1.3 and $N = 3$ in Figure 1.1). The gPb algorithm provides the probability of a boundary between two super-pixels, i.e. the pairwise marginals $p_{ij}(y_i \neq y_j|\boldsymbol{x})$. 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 (Boykov and Kolmogorov, 2004; Boykov et al., 2001; Kolmogorov and Zabih, 2004) with an energy as in (1.19). The parameter λ is estimated by grid search on the validation set. For the Herding method, the maximization step in (1.23) is also executed using 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 $p_{VOC}(y_i \neq y_j|\boldsymbol{x}) = p_{BSDS}(y_i \neq y_j|\boldsymbol{x})^{\alpha}$, 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 $\lambda\alpha \log(p_{ij}(y_i \neq y_j|x_i, x_j))\mathbb{I}(y_i \neq y_j)$, and a second parameter is therefore redundant.

Figure 1.1 shows some examples of the test images, results of different algorithms as well as 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 1.3 for the case of N=0). But Herding still leads to better smoothness and locates

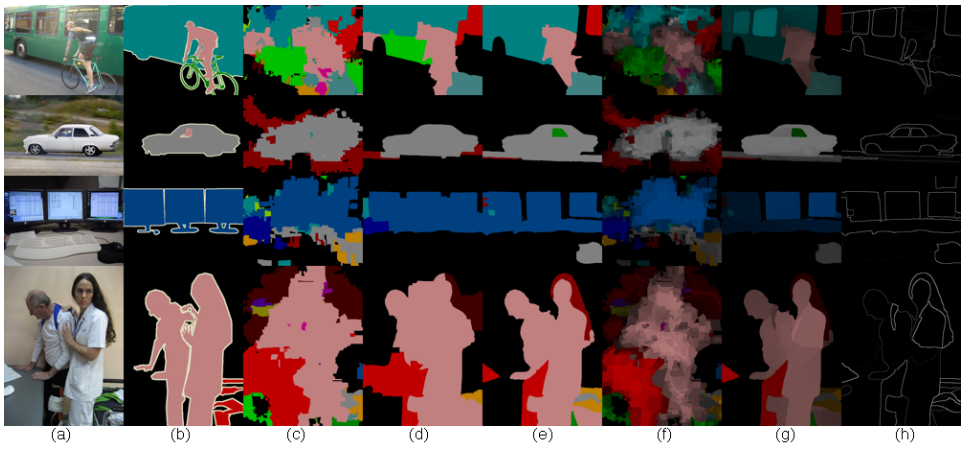
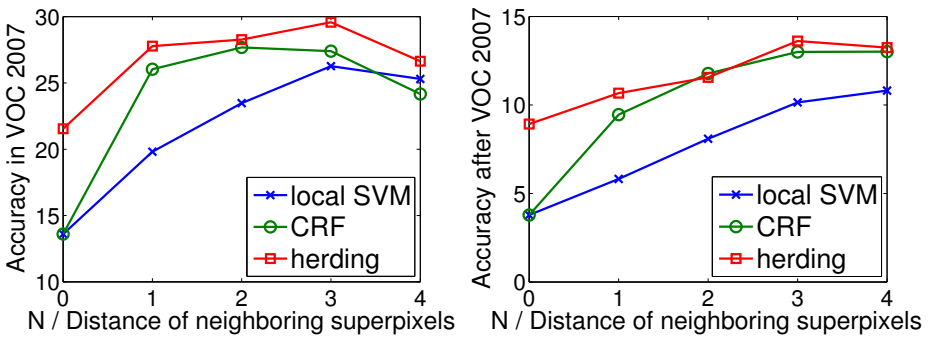


Figure 1.1: Examples of segmentation on Pascal VOC 2007 data set. Images on each line starting from left to right are respectively: (a) the original image, (b) ground truth segmentation, results of (c) local classifier, (d) CRF and (e) Herding, results with intensity proportional to the posterior probability 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.

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 VOC 2007 Competition. It measures the average recall of pixels for each category. The second measurement is the one adopted by 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 1.2. 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 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



(a) (b)

Figure 1.2: 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 1.1: 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)

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 1.3. Herding outputs a highly smoothed result with clear boundaries while CRF does not noticeably change the decision of the local classifier.

Two properties of Herding previously stated in Section 1.3 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 train a CRF with energy as a linear function of features $p_i(y_i|x_i)$ and $p_{ij}(y_i \neq y_j|x_i, x_j)$

$$P_{\text{crf-linear}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z} \exp\left(\sum_i p_i(y_i|x_i) + \lambda \sum_{i,j} p_{ij}(y_i \neq y_j|x_i, x_j)\right), \quad (1.28)$$

which 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 1.3 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.

1.4 Application: Go Game Prediction

In this section, we look at another application of the Herding algorithm to predicting the outcome of a Go game. A Go game is a board game where two players of different colors, black and white, place stones in turn on a 19×19 board. The player with the larger territory at the end of a game wins. The game of Go is known to be difficult for artificial intelligence approaches because: 1) it has a large branch factor, 361, in the game tree; and 2) it is very hard to evaluate the *goodness* of an incomplete game because of long distance correlations between stones. We tackle the second problem and propose to predict the outcome of a territory given an incomplete board as a structured prediction task. The predicted territory output can then be utilized as an evaluation function for the current board. Stern et al. (2004) proposed a CRF model for prediction with features based on pairs of neighboring stones. We treat this approach as a benchmark and compare it to the performance of Herding on the same problems.

In an incomplete game, every position has three possible values, $Black$, $White$, and $Empty$, while at the end of a game, every position is occupied by

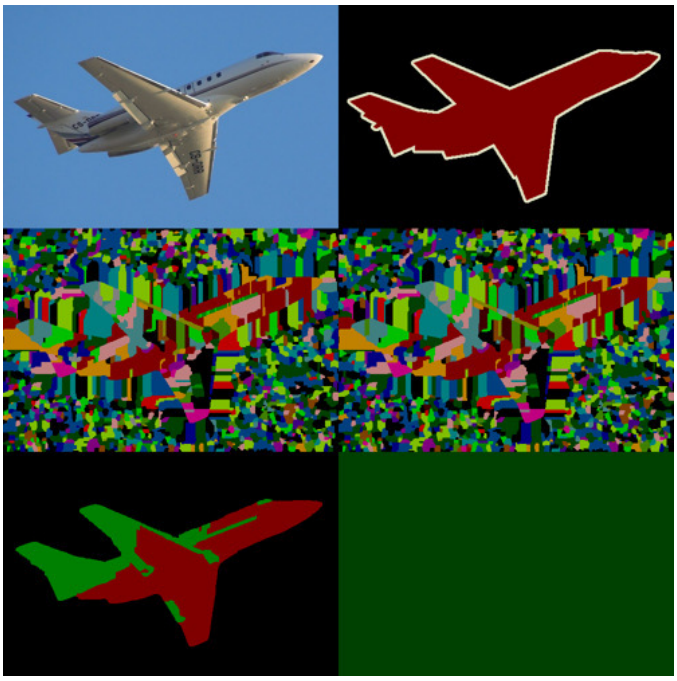


Figure 1.3: A typical example of segmentations when $N = 0$. The top 2 images are the original image (left) and the ground truth segmentation (right). The remaining 4 images (left to right, top to bottom) are respectively the segmentation of the local model, CRF, Herding and a CRF with linear potential functions. The local model is noisy because the histogram of SIFT features is computed from few pixels.

either $Black$ or $White$[2]. Denote by a vector $\boldsymbol{x} \in \{Black, White, Empty\}^N$ the stones of an incomplete game with $N = 19 \times 19$ and denote by a vector $\boldsymbol{y} \in \{0, 1\}^N$ the final territory, where 1 means $Black$ and 0 means $White$. The task of this section is to model the conditional distribution $p(\boldsymbol{y}|\boldsymbol{x})$. While it can be quite difficult to directly compute the joint distribution of all the stones on the board, it is fairly easy to obtain a local conditional model for a small patch of stones $p_\alpha(\boldsymbol{y}_\alpha|\boldsymbol{x}_\alpha)$, where α denotes the position of the patch. Figure 1.4 illustrates a case where the local conditional model patch is a pair of neighboring stones. We also consider larger patches, such as a square patch or a cross shaped patch consisting of a stone and its 4 immediate neighbors. Such patches are straightforward to introduce in Herding, but difficult in the CRF model of Stern et al. (2004) because a larger patch introduces more parameters and makes inference more difficult.

The conditional probabilities for small patches can be estimated from a training set by counting the number of occurrences of each patch configu-

2. We adopt the Chinese rule in scoring and ignore the stones shared by both players.

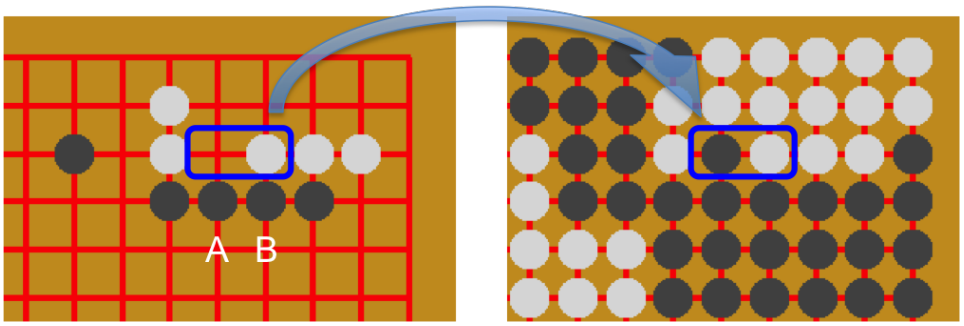


Figure 1.4: Part of a Go board of an incomplete game (left) and its final territory (right). A local model predicts the outcome of a patch (e.g. a pair of stones) based on its current values.

ration at all patch locations. A uniform prior can be used to smooth this empirical estimate. These smoothed empirical estimates can be used as local predictions of the final territory of each patch $\mathbb{E}_{p_\alpha}[\mathbb{I}[\boldsymbol{y}_\alpha = \boldsymbol{z}_\alpha]|\boldsymbol{x}_\alpha], \forall \alpha, \boldsymbol{z}_\alpha$ given an incomplete game $\boldsymbol{x}$. We can use these estimates as the input feature moments, $\psi_{\alpha,\boldsymbol{z}}(\boldsymbol{y}) = \mathbb{I}[\boldsymbol{y}_\alpha = \boldsymbol{z}_\alpha]$, and run Herding in (1.23,1.24) to produce pseudo-samples of the final territory of the whole board, $\{\boldsymbol{y}^t\}$. The average of these pseudo-samples are taken as our predicted final territory. Since the conditional probabilities are estimated independently at each location from a finite training set, we again have the problem of inconsistent moments. From the analysis in Section 1.2.3, the joint distribution of $\boldsymbol{y}$ from Herding will minimize its L_2 distance from the input local conditionals.

When the state of a patch in an incomplete game $\boldsymbol{x}_\alpha$ is all *Empty*, a local classifier cannot tell which player is more likely to win the patch. As an example, consider the empty patch "A-B" in Figure 1.4. The possession of this patch depends solely on the stones in its neighborhood. As a result, the local classifier p_{AB} will not provide any preference for y_A and y_B to be *Black* or *White* and the 4 possible states of patch "A-B" will have equal probability: $p_{AB}(y_{AB} = BB) = p_{AB}(y_{AB} = WW)$ and $p_{AB}(y_{AB} = BW) = p_{AB}(y_{AB} = WB)$. Herding treats these probabilities as moments that the average over its pseudo-samples must match. These vague moments must be matched along with more informative moments from neighboring classifiers (e.g. from the black stones above patch "A-B"). Simultaneously satisfying all of these constraints will lead to the undesirable result that about half of the Herding pseudo-samples will be $y_A = Black$ and half $y_A = White$. We avoid this problem by reducing the 4 local moments into two input moments - namely, $p_{AB}(y_{AB} = BB \text{ or } WW)$ or $p_{AB}(y_{AB} = BW \text{ or } BW)$. This allows Herding to only check if y_A and y_B have the same color and ignore the fact that y_A is more likely to be one color than the other. As

a result, Herding is free to give high probability to either $y_A = Black$ or $y_A = White$ when neighboring patches have a strong preference for either player. In general, when the conditioning patch is all $Empty$, we reduce the inputs moments in this manner by combining states with opposing colors: $\mathbb{E}_{p_\alpha}[\mathbb{I}[\boldsymbol{y}_\alpha = \boldsymbol{z}_\alpha \text{ or } \neg\boldsymbol{z}_\alpha]|\boldsymbol{x}_\alpha \text{ is all } Empty]$.

We compare the performance of Herding using a 5-stone cross-shape patch with the *Boltzmann5* CRF model proposed by Stern et al. (2004). That CRF includes unary features for each single stone, $\psi_i(y_i) = y_i$, and pairwise features for neighboring stone pairs, $\psi_{ij}(y_i, y_j) = y_i y_j$. The MLE of the parameters is learned with two inference methods: a generalization of Swendsen-Wang sampler and loopy belief propagation (BP). As the difference of the predictive performance w.r.t. cross entropy is not significant between those two methods (see Figure 4 of Stern et al. (2004)) and Swendsen-Wang is much slower than BP, we only compare Herding with a *Boltzmann5* model trained and tested with loopy BP.

We train and test our predictive models on a data set of 25089 games by professional players[3]. We prune the games that are not complete, and use the GNU GO program[4] 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) and train Herding and CRF in three stages of 20, 80, and 150 moves. The predictive performance is evaluated at each of these stages using the metric of cross entropy between actual territory outcomes, $\boldsymbol{y} \in \{0,1\}^N$, and the prediction, $\hat{\boldsymbol{y}} \in [0,1]^N$:

$$\mathcal{H} = \frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]. \tag{1.29}$$

Training each stage of the CRF model takes roughly 3 hours, while only 34 seconds were needed to compute the local conditional probabilities used by Herding. Prediction times are similar for the two methods: an average of 0.37 seconds per test game in the CRF and 0.5 seconds for Herding.

Figure 1.5 shows the cross entropy on the test set with the CRF model and Herding. The mean of the cross entropy for both the CRF model and Herding as well as the variance for the CRF decrease at later stages of games. This is reasonable since when a game is close to complete, it becomes easier to predict the final territory. Compared to the CRF, Herding has smaller variance across different games throughout all the stages. It also achieves

3. Provided by `http://gokifu.com`.
4. Downloaded from `http://www.gnu.org/software/gnugo`.

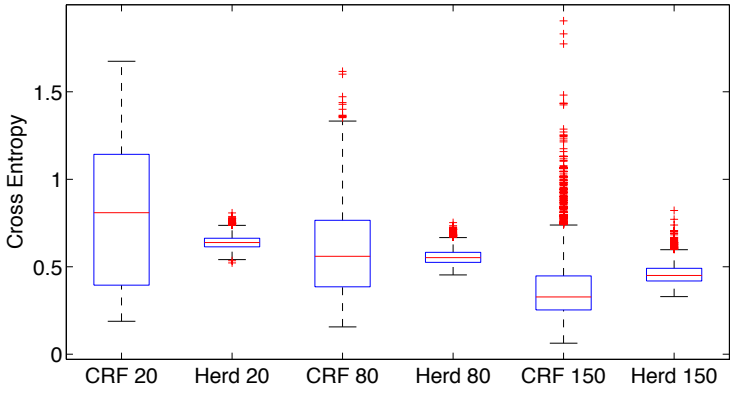
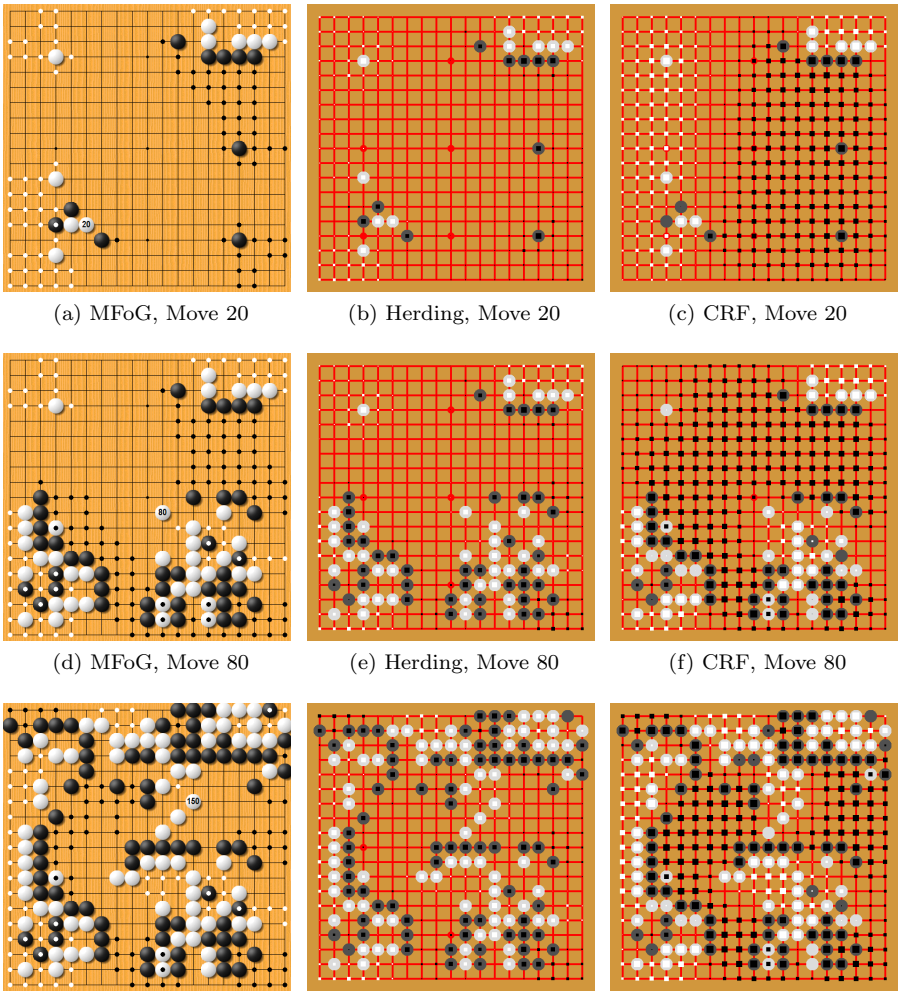


Figure 1.5: Cross entropy of final territory prediction on the test set by the CRF model and Herding at Move 20, 80, and 150. Lower is better.

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 150).

We compare the predictions of these two methods with a rule based commercial software, "Many Faces of Go," using an example game in Figure 1.6. We can see the apparent difference between the predictions of Herding and the 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, the 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 gives more conservative 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 observed at Move 150. Also, we notice that the CRF is capable of detecting captured stones such as the two white stones in the lower middle part at Move 80 and 150 but it often makes false positive mistakes at early stages of a game. In contrast, Herding usually has conservative predictions for captured stones.



(a) MFoG, Move 20 (b) Herding, Move 20 (c) CRF, Move 20

(d) MFoG, Move 80 (e) Herding, Move 80 (f) CRF, Move 80

(g) MFoG, Move 150 (h) Herding, Move 150 (i) CRF, Move 150

Figure 1.6: Territory predictions of "Many faces of go" (MFoG), Herding and the CRF model at three stages: Move 20, 80, and 150. The large circles represent current stones. MFoG outputs deterministic predictions (small dots). For the latter 2 methods, small squares represent final territory prediction from 0 (maximum white square) to 1 (maximum black square).

1.5 Conclusion

In this chapter we introduced the Herding approach to structured prediction problems and discussed its relationship to other prevailing methods including Conditional Random Fields, Structured Support Vector Machines, Max-Margin Markov Networks, and Structured Perceptrons. In particular, we present Herding as a new technique for combining local, discriminatively trained classifiers over subsets of labels into a harmonious joint model. The method is an alternative to piecewise trained CRFs and follows a markedly different philosophy in that it never learns a joint model, but rather generates representative points of some (unknown) joint distribution $p(\boldsymbol{y}|\boldsymbol{x})$.

An important theoretical contribution of this chapter relative to previous work (Gelfand et al., 2010) is that we prove that inconsistent marginals will be orthogonally projected onto the marginal polytope. This makes Herding a unique way to combine inconsistent local classifiers. Remarkably, the fast convergence rate of $\mathcal{O}(1/T)$ is preserved in this situation.

We demonstrated our algorithm on image segmentation and Go game prediction tasks, showing it is competitive in both tasks. Herding differs greatly from approaches that combine local predictions using CRFs. Predictions based on piecewise CRFs output the most likely assignment that maximizes the *product* of local beliefs (implicitly assuming independence), while Herding obtains a joint distribution over all labels that compromises among inconsistent local beliefs in terms of the L_2 distance. Herding's predictions often appear more coherent as a result (see Figure 1.1 and Figure 1.6).

As different local predictions may have various confidence in their prediction, it is important to assign different weights during combination. In the image segmentation task, we optimize the relative step size λ on an evaluation set in order to balance the evidence from the local appearance cue and boundary information. In the Go prediction task, we ignore uninformative marginals from classifiers on empty patches. A more principled approach to weighting local classifiers is a direction of future research.

1.6 References

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5), 2011. ISSN 0162-8828.

H. Block and S. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. In *Proceedings of the American Mathematical Society*, volume 26(2, page 229235, 1970.

Y. Boykov and V. Kolmogorov. 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.

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

Y. Chen, A. Gelfand, C. Fowlkes, and M. Welling. Integrating local classifiers through nonlinear dynamics on label graphs with an application to image segmentation. In *ICCV*, pages 2635–2642, 2011.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

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

A. Gelfand, L. van der Maaten, Y. Chen, and M. Welling. On herding and the cycling perceptron theorem. In *Advances in Neural Information Processing Systems 23*, pages 694–702, 2010.

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

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

M. Minsky and S. Papert. *Perceptrons - An introduction to computational geometry.* MIT Press, 1969.

P. Pletscher, C. S. Ong, and J. M. Buhmann. Entropy and margin maximization for structured output learning. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 83–98, Berlin, Heidelberg, 2010. Springer-Verlag.

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

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

C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *Proceedings of the 24th international conference on Machine learning*, pages 863–870. ACM, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Neural Information Processing Systems (NIPS-03)*, Vancouver, CA, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, New York, NY, USA, 2004.

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

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