Diagram Structure Recognition by Bayesian Conditional Random Fields

Yuan Qi
MIT CSAIL
32 Vassar Street
Cambridge, MA, 02139, USA
alanqi@csail.mit.edu

Martin Szummer
Microsoft Research
7 J J Thomson Avenue
Cambridge, CB3 0FB, UK
szummer@microsoft.com

Thomas P. Minka
Microsoft Research
7 J J Thomson Avenue
Cambridge, CB3 0FB, UK
minka@microsoft.com

Abstract

Hand-drawn diagrams present a complex recognition problem. Elements of the diagram are often individually ambiguous, and require context to be interpreted.

We present a recognition method based on Bayesian conditional random fields (BCRFs) that jointly analyzes all drawing elements in order to incorporate contextual cues. The classification of each object affects the classification of its neighbors. BCRFs allow flexible and correlated features, and take both spatial and temporal information into account. BCRFs estimate the posterior distribution of parameters during training, and average predictions over the posterior for testing. As a result of model averaging, BCRFs avoid the overfitting problems associated with maximum likelihood training. We also incorporate Automatic Relevance Determination (ARD), a Bayesian feature selection technique, into BCRFs. The result is significantly lower error rates compared to ML- and MAP-trained CRFs.

1. Introduction

Many computer vision tasks can be effectively formalized as joint classification of multiple pixels or elements. Joint classification enables modeling of dependence between elements, allowing structure and context to be taken into account. For example, image segmentation requires joint labeling of multiple pixels such that related pixels have the same label but pixels across edge or color boundaries have different labels. Object recognition is similarly facilitated by contextual information, such as knowledge about the scene, which consists of multiple interacting elements. In this paper, we tackle another instance of the same problem, namely the recognition of hand-drawn diagram structure (Figure 2). Specifically, we try to identify which stroke fragments are parts of containers versus connectors. This task is challenging because individual pen-strokes look alike and only acquire meaning relative to neighboring strokes. Rather than classify strokes entirely based on local features, we desire to find a globally-consistent labeling of the diagram—an approach which turns out to considerably improve performance.

Joint modeling of structured data can be performed by generative graphical models, such as Bayesian networks or Markov random fields. For example, Tu et al. (2003) propose a generative model for globally parsing images into faces, text, shadows, etc. This sort of approach could be used for diagrams, but in our experience generative models take substantial effort to design and implement for each individual problem, and the result can easily require a prohibitive amount of computation.

Conditional random fields (CRF) are a conditional approach for classifying structured data, proposed by Lafferty et al. (2001). CRFs model only the label distribution conditioned on the observations. Unlike generative models, they do not need to explain the observations or features. This also allows CRFs to use flexible features such as complex functions of multiple observations. The same CRF architecture can be used on many different problems, simply by changing the features. The modeling power of CRFs has shown great benefit in several applications, including region classification (Kumar & Hebert, 2004), diagram labeling (Szummer & Qi, 2004), and CRFs have also been extended to perform simultaneous segmentation and recognition (Cowans & Szummer, 2005).

To summarize, CRFs provide a compelling model for structured data. Consequently, we need effective training, inference, and feature selection algorithms. The standard maximum likelihood (ML) criterion is prone to overfitting the data, especially when CRFs are trained with large numbers of features. Previous approaches to reduce overfitting include maximum a posteriori (MAP) and large margins (Taskar et al., 2004). Both of these approaches involve free parameters which are hard to tune.

Instead we use the method of Bayesian conditional random fields (BCRFs), proposed by (Qi et al., 2005). BCRFs can be viewed as a generalization of Bayes Point Machines (BPMs) (Herbrich et al., 1999; Minka, 2001a), discriminative Bayesian classifiers for single elements. The advantages of BCRFs over conventionally-trained CRFs include:

1. Model averaging, which exploits the uncertainty of the estimated parameters. Given a test diagram, the predictions of all possible CRFs are averaged according to their posterior probability. Traditional approaches provide only a point estimate of model parameters, ignoring uncertainty and risking overfitting.

2. Automatic hyperparameter tuning. We illustrate this by performing feature selection via Automatic Relevance Determination (ARD). This method prunes features by maximizing the model evidence (MacKay, 1992), and has been used successfully in neural networks (MacKay, 1992) and logistic classifiers (Tipping, 2000).

In the following sections, we first review CRFs and expectation propagation, then present Bayesian conditional random fields, incorporate automatic relevance determination, and finally give experimental results.

2. Conditional Random Fields

A conditional random field (CRF) models label variables according to an undirected graphical model conditioned on observed data (Lafferty et al., 2001). Let $\mathbf{x}$ be an "input" vector describing the observed data instance, and $\mathbf{t}$ be an "output" random vector over labels of the data components. We assume that all labels for the components belong to a finite label alphabet $\mathcal{T} = \{1, \ldots, M\}$. For example, the input $\mathbf{x}$ could be image features based on small patches, and $\mathbf{t}$ be labels denoting 'person, 'car' or 'other' patches. Formally, we have the following definition of CRFs (Lafferty et al., 2001):

Definition 2.1 *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph such that $\mathbf{t}$ is indexed by the vertices of G. Then $(\mathbf{x}, \mathbf{t})$ is a conditional random field (CRF) if, when conditioned on $\mathbf{x}$, the random variables t_i obey the Markov property with respect to the graph: $p(t_i|\mathbf{x}, \mathbf{t}_{\mathcal{V}-i}) = p(t_i|\mathbf{x}, \mathbf{t}_{\mathcal{N}_i})$ where $\mathcal{V}-i$ is the set of all nodes in G except the node i, $\mathcal{N}_i$ is the set of neighbors of the node i in G, and $\mathbf{t}_\Omega$ represents the random variables of the vertices in the set Ω.*

Unlike traditional generative random fields, CRFs only model the conditional distribution $p(\mathbf{t}|\mathbf{x})$ and do not explicitly model the marginal $p(\mathbf{x})$. Note that the label vectors t_i are globally conditioned on the whole observation $\mathbf{x}$ in CRFs. Thus, we do not assume that the observed data $\mathbf{x}$ are conditionally independent as in a generative random field.

A CRF defines the conditional distribution of the labels $\mathbf{t}$ given the observations $\mathbf{x}$ to be proportional to a product of potential functions on cliques of the graph G. For simplicity, we consider only pairwise clique potentials such that

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \prod_{\{i,j\} \in \mathcal{E}} g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) \qquad (1)$$

where

$$Z(\mathbf{w}) = \sum_{\mathbf{t}} \prod_{\{i,j\} \in \mathcal{E}} g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) \qquad (2)$$

is a normalizing factor known as the partition function, $g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w})$ are pairwise potentials, and $\mathbf{w}$ are the model parameters. Note that the partition function is a complex function of the model parameter $\mathbf{w}$. This makes Bayesian training much harder for CRFs than for Bayesian linear classifiers, since the normalizer of a Bayesian linear classifier is a constant.

In standard conditional random fields, the pairwise potentials are defined as

$$g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) = \exp(\mathbf{w}_{t_i,t_j}^{\mathrm{T}} \boldsymbol{\phi}_{i,j}(t_i, t_j, \mathbf{x})) \qquad (3)$$

where $\boldsymbol{\phi}_{i,j}(t_i, t_j, \mathbf{x})$ are features extracted for the edge between vertices i and j of the conditional random field, and $\mathbf{w}_{t_i,t_j}$ are weights corresponding to labels $\{t_i, t_j\}$ in $\mathbf{w}$, where $\mathbf{w} = [\mathbf{w}_{1,1}^{\mathrm{T}}, \mathbf{w}_{1,2}^{\mathrm{T}}, \ldots, \mathbf{w}_{M,M}^{\mathrm{T}}]^{\mathrm{T}}$. There are no restrictions on the relation between features.

Though we could use the above log-linear potential functions, for convenience we replace the exponential by the probit function $\Psi(\cdot)$ (the cumulative distribution function of a Gaussian). In addition, to incorporate robustness against labeling errors, we add a small probability ϵ of a label being incorrect, which will serve to bound the potential value away from zero. The final potentials are

$$g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) = \epsilon + (1 - 2\epsilon)\Psi(\mathbf{w}_{t_i,t_j}^{\mathrm{T}} \boldsymbol{\phi}_{i,j}(t_i, t_j, \mathbf{x})). \qquad (4)$$

Given the data likelihood (1) and a Gaussian prior

$$p_0(\mathbf{w}) \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathrm{diag}(\boldsymbol{\alpha}^{-1})), \qquad (5)$$

the posterior distribution of the parameters is

$$p(\mathbf{w}|\mathbf{t}, \mathbf{x}) \propto p_0(\mathbf{w}) \frac{1}{Z(\mathbf{w})} \prod_{k \in \mathcal{E}} g_k(t_i, t_j, \mathbf{x}; \mathbf{w})$$

where $k = \{i, j\}$ indexes edges in a CRF. Since this posterior distribution cannot be analytically computed, it is approximated by EP and power EP approaches, which are described in the following sections. Moreover, as shown in Section 5, we can use the Bayesian machinery to estimate the hyperparameter $\boldsymbol{\alpha}$ in the prior (5).

3. Expectation Propagation

Expectation propagation exploits the fact that the posterior is a product of simple terms. If we approximate each of these terms well, we can get a good approximation of the posterior. We fit an approximation of the form $p(\mathbf{w}|\mathbf{t}, \mathbf{x}) \approx q(\mathbf{w})$ where

$$q(\mathbf{w}) = p_0(\mathbf{w}) \frac{1}{\tilde{Z}(\mathbf{w})} \prod_{k \in \mathcal{E}} \tilde{g}_k(\mathbf{w}) \tag{6}$$

The terms $\tilde{g}_k(\mathbf{w})$ and $1/\tilde{Z}(\mathbf{w})$ will have the form of a Gaussian, so that the approximate posterior $q(\mathbf{w})$ will be Gaussian. The parameters of the approximation are the Gaussian parameters for the individual terms $\tilde{g}_k(\mathbf{w})$ and $1/\tilde{Z}(\mathbf{w})$, from which the mean and variance of $q(\mathbf{w})$ can be derived by standard formulas. The parameters for the numerator terms are updated in the same way as for Bayesian linear classifiers (Minka, 2001b). Specifically, for $\tilde{g}_k$, the algorithm first removes its contribution to $q(\mathbf{w})$, giving a Gaussian $q^{\setminus \tilde{g}_k}(\mathbf{w})$. Then it finds a new choice of $\tilde{g}_k$ which minimizes KL-divergence to the true g_k, holding $q^{\setminus \tilde{g}_k}$ fixed. This process can be written succinctly as follows.

While not converged: Loop over k:

$$q^{\setminus \tilde{g}_k}(\mathbf{w}) = q(\mathbf{w}) / \tilde{g}_k(\mathbf{w}) \tag{7}$$

$$\tilde{g}_k^{\text{new}}(\mathbf{w}) = \underset{\acute{g}_k(\mathbf{w})}{\operatorname{argmin}} \, \mathrm{KL}(g_k(\mathbf{w}) q^{\setminus \tilde{g}_k}(\mathbf{w}) \parallel \acute{g}_k(\mathbf{w}) q^{\setminus \tilde{g}_k}(\mathbf{w})) \tag{8}$$

$$= \operatorname{moments}\Big[g_k(\mathbf{w}) q^{\setminus \tilde{g}_k}(\mathbf{w}) \Big] / q^{\setminus \tilde{g}_k}(\mathbf{w}) \tag{9}$$

Here $\operatorname{moments}$ is a "moment matching" operator: it finds the Gaussian having the same moments as its argument, thus minimizing KL-divergence. Algorithmically, eq. (7) divides the Gaussians to get a new Gaussian, and calls it $q^{\setminus \tilde{g}_k}(\mathbf{w})$. Similarly, eq. (9) constructs a Gaussian whose moments match $g_k(\mathbf{w}) q^{\setminus \tilde{g}_k}(\mathbf{w})$ and divides it by $q^{\setminus \tilde{g}_k}(\mathbf{w})$, to get a new Gaussian approximation term $\tilde{g}_k(\mathbf{w})^{\text{new}}$. This update provides a new $q(\mathbf{w})$ according to (6). The moment calculations involve integrals of products of Gaussians with probits and are computed as in Minka (2001b).

As for the denominator, if we were to apply this method directly to $\tilde{Z}(\mathbf{w})$, we would have to compute $\operatorname{moments}\Big[\frac{1}{Z(\mathbf{w})} q^{\setminus \tilde{Z}}(\mathbf{w})\Big]$, which is difficult. Our solution is to not minimize KL-divergence, but instead to use a different divergence measure which makes $\tilde{Z}(\mathbf{w})$ similar to $Z(\mathbf{w})$. This shortcut has been formalized as the "Power EP" method, and its implications are discussed by Minka (2004). Using this alternative divergence measure, we only have to compute $\operatorname{moments}\Big[Z(\mathbf{w}) q^{\setminus \tilde{Z}}(\mathbf{w})\Big]$, which is a problem of the same form as the numerator terms, but involving an average over $\mathbf{t}$. This problem is solved by a nested invocation of EP. By interleaving the updates in this nested EP with those of the outer EP, we obtain an elegant algorithm in which numerator and denominator terms are updated in parallel, with similar formulas. This algorithm is derived in (Qi et al., 2005) and the next section describes a concise form for it.

4. Power EP for Conditional Random Fields

Recall that the posterior distribution for the parameters is

$$p(\mathbf{w}|\mathbf{t}, \mathbf{x}) \propto p_0(\mathbf{w}) \frac{\prod_{k \in \mathcal{E}} g_k(t_i, t_j, \mathbf{x}; \mathbf{w})}{\sum_{\mathbf{t}} \prod_{k \in \mathcal{E}} g_k(t_i, t_j, \mathbf{x}; \mathbf{w})}. \tag{10}$$

We fit an approximation of the form

$$q(\mathbf{w}) = p_0(\mathbf{w}) \frac{\prod_{k \in \mathcal{E}} \tilde{g}_k(\mathbf{w})}{\sum_{\mathbf{t}} \prod_{k \in \mathcal{E}} \tilde{f}_k(\mathbf{w}) \tilde{f}_k(t_i) \tilde{f}_k(t_j)}. \tag{11}$$

Here $\tilde{f}_k(t_i)$ is a discrete distribution on the label t_i. The parameters of the approximation are the Gaussian parameters of $\tilde{g}_k(\mathbf{w})$ and $\tilde{f}_k(\mathbf{w})$, as well as the discrete parameters of $\tilde{f}_k(t_i)$. The terms involving only t_i sum away in the denominator of $q(\mathbf{w})$, but we use them to define a separate quantity

$$q(t_i) = \prod_{k \in \mathcal{E}} \tilde{f}_k(t_i). \tag{12}$$

Initialize $\tilde{f}_k(t_i) = 1$ (uniform) and $\tilde{g}_k(\mathbf{w}) = \tilde{f}_k(\mathbf{w}) = 1$ (infinite variance). By $g_k(\mathbf{w})$ denote the exact term $g_k(t_i, t_j, \mathbf{x}; \mathbf{w})$ with the observed values t_i and t_j in the training set. Then the algorithm proceeds as follows:

While not converged: Loop over k:

$$q^{\setminus \tilde{g}_k}(\mathbf{w}) = q(\mathbf{w}) / \tilde{g}_k(\mathbf{w}) \tag{13}$$

$$\tilde{g}_k^{\text{new}}(\mathbf{w}) = \operatorname{moments}\Big[g_k(\mathbf{w}) q^{\setminus \tilde{g}_k}(\mathbf{w}) \Big] / q^{\setminus \tilde{g}_k}(\mathbf{w}) \tag{14}$$

$$q^{\setminus \tilde{f}_k}(\mathbf{w}) = q(\mathbf{w}) / \tilde{f}_k(\mathbf{w}) \tag{15}$$

$$q^{\setminus \tilde{f}_k}(t_i) = q(t_i) / \tilde{f}_k(t_i) \tag{16}$$

$$f_k(\mathbf{w}) = \sum_{t_i, t_j} g_k(t_i, t_j, \mathbf{x}; \mathbf{w}) q^{\setminus \tilde{f}_k}(t_i) q^{\setminus \tilde{f}_k}(t_j) \tag{17}$$

$$\tilde{f}_k^{\text{new}}(\mathbf{w}) = \operatorname{moments}\Big[f_k(\mathbf{w}) q^{\setminus \tilde{f}_k}(\mathbf{w}) \Big] / q^{\setminus \tilde{f}_k}(\mathbf{w}) \tag{18}$$

$$\tilde{f}_k^{\text{new}}(t_i) = \sum_{t_j} \int_{\mathbf{w}} g_k(t_i, t_j, \mathbf{x}; \mathbf{w}) q^{\setminus \tilde{f}_k}(\mathbf{w}) q^{\setminus \tilde{f}_k}(t_j) \, d\mathbf{w} \tag{19}$$

Note that (13) and (15) use the same value of $q(\mathbf{w})$; the numerator and denominator terms are updated in parallel. After all three updates are complete we obtain a new $q(\mathbf{w})$ and

$q(\mathbf{t})$. The equations (17,18) look imposing but all they mean is that you compute the moments of $g_k(t_i, t_j, \mathbf{x}; \mathbf{w}) q^{\setminus \tilde{f}_k}(\mathbf{w})$ for each combination of t_i and t_j, then average according to the discrete probabilities $q^{\setminus \tilde{f}_k}(t_i) q^{\setminus \tilde{f}_k}(t_j)$. Belief propagation is a special case of EP where the approximations are discrete distributions, so unsurprisingly the update (19) is equivalent to loopy belief propagation on the labels t_i, with the twist that the edge potentials are averaged over the parameters $\mathbf{w}$. Due to our choice of edge potentials, this average is available analytically. The final result of training is the Gaussian $q(\mathbf{w})$.

A straightforward implementation of the updates costs $O(d^3)$ time, where d is the dimension of the parameter vector $\mathbf{w}$, since they involve inverting the covariance matrix of $\mathbf{w}$. However, as shown in Qi et al. (2005), we can compute them with low-rank matrix updates in $O(d^2)$ time.

5. Automatic Relevance Determination

We also include ARD in the BCRF training. By pruning away irrelevant features, we gain robustness to noise and faster testing times.

Recall that the prior has a vector of hyperparameters $\boldsymbol{\alpha}$, representing the amount we wish to shrink each weight. ARD tunes these hyperparameters by maximizing the model evidence, which is the posterior probability of the model marginalized over model parameters (MacKay, 1992). As shown by MacKay (1992), if we approximate the posterior distribution of the model parameters by a Gaussian with mean $\mathbf{m}_w$ and variance $\mathbf{V}_w$, then $\boldsymbol{\alpha}$ can be updated by an EM equation:

$$\alpha_j^{\text{new}} = \frac{1}{(\mathbf{V}_w)_{jj} + (\mathbf{m}_w)_j^2}. \tag{20}$$

In the case of EP, $\mathbf{m}_w$ and $\mathbf{V}_w$ are the moments of $q(\mathbf{w})$, which itself depends on $\boldsymbol{\alpha}$. Thus we have an EM algorithm where the E-step is EP and the M-step is (20). When a feature is irrelevant, the corresponding α_j goes to infinity which forces $w_j = 0$, effectively pruning the feature from the model.

6. Inference by Approximate Model Averaging

Unlike traditional classification problems, where we have a scalar output for each input, a BCRF jointly labels all the hidden vertices in an undirected graph. The trained BCRF infers the labels by model averaging, which makes full use of the training data by employing not only the estimated mean of the parameters, but also the estimated uncertainty (variance).

Given a new test graph $\mathbf{x}^\star$, a BCRF trained on $(\mathbf{x}, \mathbf{t})$ approximates the predictive distribution as follows:

$$p(\mathbf{t}^\star|\mathbf{x}^\star, \mathbf{t}, \mathbf{x}) = \int p(\mathbf{t}^\star|\mathbf{x}^\star, \mathbf{w}) p(\mathbf{w}|\mathbf{t}, \mathbf{x}) \, \mathrm{d}\mathbf{w} \tag{21}$$

$$\approx \int p(\mathbf{t}^\star|\mathbf{x}^\star, \mathbf{w}) q(\mathbf{w}) \, \mathrm{d}\mathbf{w} \tag{22}$$

$$= \int \frac{q(\mathbf{w})}{Z(\mathbf{w})} \prod_{\{i,j\} \in \mathcal{E}} g_{i,j}(t_i^\star, t_j^\star, \mathbf{x}^\star; \mathbf{w}) \, \mathrm{d}\mathbf{w} \tag{23}$$

where $q(\mathbf{w})$ is the fitted approximation to the true posterior $p(\mathbf{w}|\mathbf{t}, \mathbf{x})$. Even though we have approximated this posterior as a Gaussian, the integral (23) is still complicated if there are many edges in the graph.

Thus we will approximate the integral using EP. Since we are interested in the marginal distribution of $t_i^\star$, we use a fully-factorized approximation of the form

$$q(\mathbf{t}^\star) = \int \frac{q(\mathbf{w})}{\tilde{Z}(\mathbf{w})} \prod_{\{i,j\} \in \mathcal{E}} \tilde{f}_k(\mathbf{w}) \tilde{f}_k(t_i) \tilde{f}_k(t_j) \, \mathrm{d}\mathbf{w}, \tag{24}$$

with an analogous approximation of $\tilde{Z}(\mathbf{w})$. This leads to the same updates (16, 19) for $\tilde{f}_k(t_i)$. Interestingly, we do not need to compute $\tilde{f}_k(\mathbf{w})$ because it is always cancelled by the corresponding term of $Z(\mathbf{w})$. Thus the average over $\mathbf{w}$ in (19) is always with respect to $q(\mathbf{w})$ alone. In summary, testing with a BCRF reduces to loopy belief propagation over the labels, with each edge potential averaged over the fixed parameter distribution $q(\mathbf{w})$.

7. Application to Ink Classification

Here we apply BCRFs to ink classification, specifically to discrimination between containers and connectors in drawings of organization charts. BCRFs enable joint classification of all ink on a page. The classification of one ink fragment can influence the classification of others, so that context is exploited.

We break the task into three steps:

1. Subdivision of pen strokes into fragments,
2. Construction of a conditional random field on the fragments,
3. BCRF training and inference on the network.

The input is electronic ink recorded as sampled locations of the pen, and collected into *strokes* defined as pen-down to pen-up events. In the first step, the strokes are divided into simpler components called *fragments*. Fragments should be small enough to belong to a single container or connector. In contrast, strokes occasionally span more than one part, for example when a user draws a container and a connector

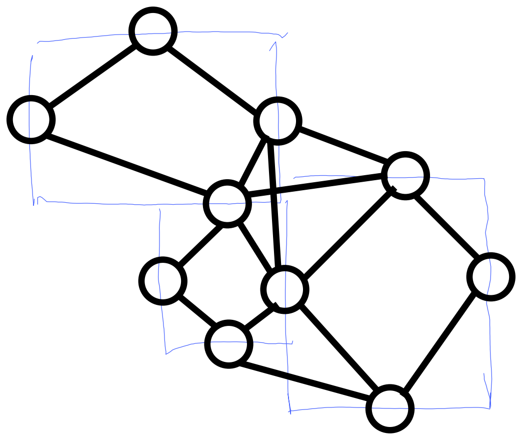


Figure 1. The conditional random field superimposed on part of the chart from Figure 2. There is one node (circled) per fragment, and edges indicate pairwise potentials between neighboring fragments.

without lifting the pen. One could choose the fragments to be individual sampled dots of ink, however, this would be computationally expensive. We choose fragments to be groups of ink dots within a stroke that form straight line segments (within some tolerance) (Figure 2).

In the second step, we construct a conditional random field on the fragments. Each ink fragment is represented by a node in the network (Figure 1). The node has an associated label variable t_i, which takes on the values -1 (container) or 1 (connector). The potential functions g quantify how compatible labels are with the underlying ink and with other labels. Weights in the potential functions characterize the exact dependence of labels with the ink, and these weights are trained from data.

We extract features both from single fragments and from pairs of fragments. Specifically, our approach is to compute many redundant low-level ink features, and represent them as potentials. BCRF-ARDs then learns which features or combinations of features that are discriminative for the task. The details about feature extraction from the organization charts can be found in (Szummer & Qi, 2004).

8. Experiments and Discussion

We asked 17 subjects to draw given organization charts on a TabletPC device capturing online handwriting. The given charts consisted of rectangular containers and connectors made from line segments, but the subjects' drawings were quick and rough. We focused on graphical elements, and any text was removed.

The pen strokes were subdivided yielding a database of 1000 fragments, which we split into training sets drawn by half of the subjects, and test sets drawn by the other half. We built a conditional random field, with pairwise potential between all pairs of fragments that were within 5mm of each other, resulting in 3000 pairwise potentials.

We compared BCRF-ARD with CRFs trained by maximum likelihood (ML) and maximum a posteriori (MAP) methods on the ink recognition task. We included CRFs with probit potential functions (4) and with exponential potential functions (3).

The results are shown in Table 1. BCRF-ARD outperforms the other approaches. For BCRF and MAP training, prior parameters for the potentials were $\boldsymbol{\alpha} = 5\mathbf{I}$ (when not using ARD), with the labeling noise rate $\epsilon = 0$. Here $\mathbf{I}$ represents the identity matrix. In other words, we used the independent Gaussian prior distribution over $\mathbf{w}$. For BCRF-ARD training, we optimized over the prior parameters $\boldsymbol{\alpha}$, which reduced the number of active features from 148 to 38. We could have used the model evidence to optimize ϵ as well, but for simplicity we set it to zero.

The time for BCRF training is only slightly longer than ML training, as discussed by (Qi et al., 2005). The time for testing is essentially the same, since both run belief propagation, just on different edge potentials.

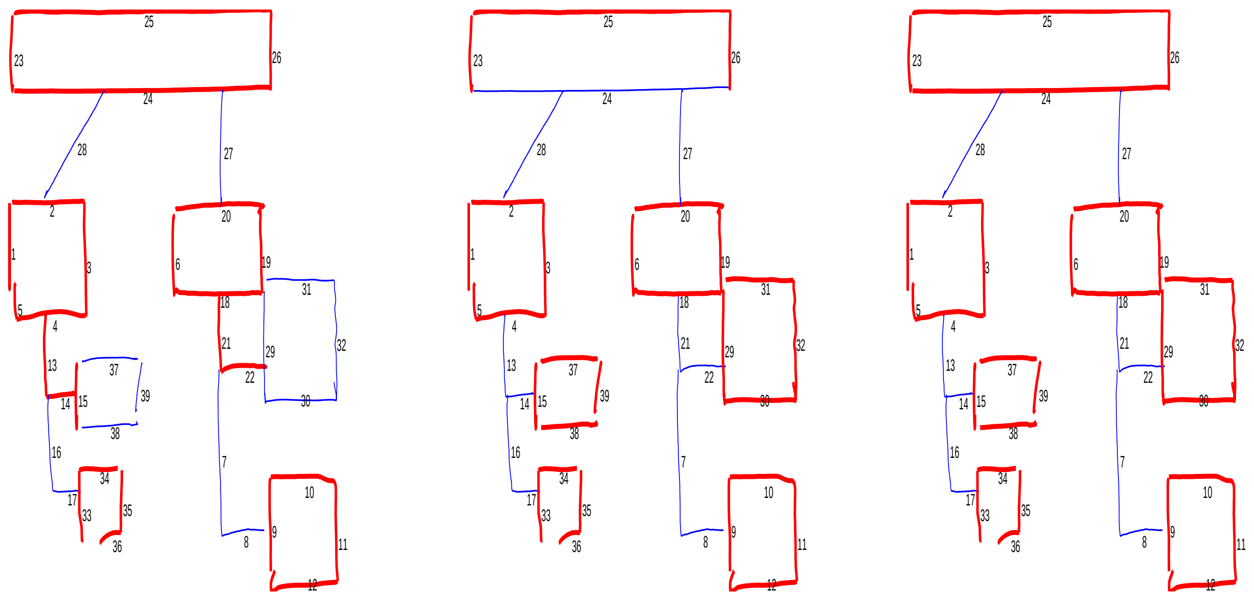
Table 1. Test error rates on organization charts. The results are averaged over 10 random train-test splits. The stars indicate errors with 98% significance of being worse than BCRF error. The diamond indicates 85% significance that BCRFs outperform MAP-trained CRFs with exponential potentials. Moreover, ARD further improves BCRF test accuracy.

Algorithm	Test error rates
ML-Probit-CRF	10.7 ± 1.21 $\star$
MAP-Probit-CRF	6.00 ± 0.64 $\star$
ML-Exp-CRF	10.1 ± 1.21 $\star$
MAP-Exp-CRF	5.20 ± 0.79 $\diamond$
BCRF	4.39 ± 0.62
BCRF-ARD	$\mathbf{3.98 \pm 0.48}$

Finally, Figure 2 illustrates the power of BCRFs using contextual information for joint classification. To the left we see the results of a Bayes Point Machine, an excellent Bayesian classifier that however does not exploit the contextual information as BCRFs. There are two ambiguous rectangles created from fragments 18-21-22-29 and 14-16-17. The independent classification approach misclassifies fragments 19, 29, 16 and 1. In the middle, we see the joint-classification results of a MAP-trained CRF. To the right, we see that a BCRF with pairwise potentials resolves the ambiguity and correctly classifies all fragments.

9. Summary

In this paper, we have incorporated ARD, a Bayesian feature selection technique, into BCRFs and applied BCRF-



(a) Individual classification by BPM (b) joint classification by MAP-trained CRF (c) joint classification by BCRF

Figure 2. Classification of a chart, with results of a BPM that does not exploit contextual informations (left), a MAP-trained CRF (middle) and a BCRF (right) that both propagate information using pairwise potentials. Containers are shown in bold and connectors as thin lines. The BCRF achieves a more accurate classification than the other two approaches.

ARD to the joint analysis of hand-drawn diagrams, achieving accurate test performance. BCRF-ARD can also be applied in many other computer vision problems, such as image segmentation, denoising, and scene recognition.

Acknowledgments

We wish to thank Christopher Bishop, Michel Gangnet, and Hannah Pepper for useful discussions, ink collection, and ink preprocessing.

References

Cowans, P., & Szummer, M. (2005). A Graphical Model for Simultaneous Partitioning and Labeling *Proc. AI & Statistics*.

Herbrich, R., Graepel, T., & Campbell, C. (1999). Bayes point machine: Estimating the Bayes point in kernel space. *IJCAI Workshop SVMs* (pp. 23–27).

Kumar, S., & Hebert, M. (2004). Discriminative fields for modeling spatial dependencies in natural images. *NIPS 16*.

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

MacKay, D. J. (1992). Bayesian interpolation. *Neural Computation*, *4*, 415–447.

Minka, T. (2001a). *A family of algorithms for approximate Bayesian inference*. Doctoral dissertation, MIT.

Minka, T. P. (2001b). Expectation propagation for approximate Bayesian inference. *Proc. Uncertainty in AI*.

Minka, T. P. (2004). Power EP. `http://www.research.microsoft.com/~minka/`

Minka, T. P., & Lafferty, J. (2002). Expectation propagation for the generative aspect model. *Proc. Uncertainty in AI (UAI)*.

Qi, Y., Szummer, M., & Minka, T. P. (2005). Bayesian conditional random fields. *Proc. AI & Statistics*.

Szummer, M., & Qi, Y. (2004). Contextual recognition of hand-drawn diagrams with Conditional Random Fields. *Proc. 9th Intl. Workshop on Frontiers in Handwriting Recognition*.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. *NIPS 16*.

Tipping, M. E. (2000). The relevance vector machine. *NIPS* (pp. 652–658). The MIT Press.

Tu, Z., Chen, X., Yuille, A., & Zhu, S.-C. (2003). Image parsing: Segmentation, detection, and object recognition. *ICCV*.

Wiegerinck, W., & Heskes, T. (2002). Fractional belief propagation. *NIPS 15*.