

AdaBoost.MRF: Boosted Markov random forests and application to multilevel activity recognition

AUTHOR(S)

Truyen Tran, Quoc-Dinh Phung, H Bui, Svetha Venkatesh

PUBLICATION DATE

01-01-2006

HANDLE

10536/DRO/DU:30044600

Downloaded from Deakin University's Figshare repository

Deakin University CRICOS Provider Code: 00113B

Deakin Research Online

This is the published version:

Truyen, Tran The, Phung, Dinh Q., Bui, Hung H. and Venkatesh, Svetha 2006, AdaBoost.MRF: boosted Markov random forests and application to multilevel activity recognition, *in CVPR 2006 : Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, Piscataway, N.J., pp. 1686-1693.

Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30044600

Reproduced with the kind permissions of the copyright owner.

Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright : 2006, IEEE

AdaBoost.MRF: Boosted Markov Random Forests and Application to Multilevel Activity Recognition

Tran The Truyen [†], Dinh Q. Phung [†], Hung H. Bui [‡*], Svetha Venkatesh [†]
[†]Department of Computing, Curtin University of Technology
GPO Box U 1987, Perth, WA, Australia
{trantt2,phungquo,svetha}@cs.curtin.edu.au
[‡]Artificial Intelligence Center, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025, USA
bui@ai.sri.com

Abstract

Activity recognition is an important issue in building intelligent monitoring systems. We address the recognition of multilevel *activities in this paper via a conditional Markov random field (MRF), known as the dynamic conditional random field (DCRF). Parameter estimation in general MRFs using maximum likelihood is known to be computationally challenging (except for extreme cases), and thus we propose an efficient boosting-based algorithm AdaBoost.MRF for this task. Distinct from most existing work, our algorithm can handle hidden variables (missing labels) and is particularly attractive for smarthouse domains where reliable labels are often sparsely observed. Furthermore, our method works exclusively on trees and thus is guaranteed to converge. We apply the AdaBoost.MRF algorithm to a home video surveillance application and demonstrate its efficacy.*

1. Introduction

Recently, there has been a growing research interest in developing probabilistic models to address the problem of automated recognition of activities of daily livings (ADLs) - an important issue in building intelligent monitoring systems. Most existing approaches, however, have taken a generative approach, in particular, the hidden Markov model and its variants. Notwithstanding initial success, generative models rely on joint probability models and thus suffer from several shortcomings, in particular failing to reflect the *discriminative* nature of data. In this paper, we advocate a discriminative approach for activity modeling and recognition. Observing that activities are naturally acted out in a hierarchical manner, correlated temporally and across *multilevel* of semantics, we propose the use of Dynamic Conditional Random Fields (DCRFs) [14], a conditional version of the Markov random fields (MRFs), to model the activities where temporal regularities at different levels of abstraction can be jointly represented. We then introduce AdaBoosted Markov Random Forests (AdaBoost.MRF), a novel boosting-based algorithm for parameter estimation of the general MRFs. Given training data, we train the DCRFs and later use them to annotate and segment unseen observation sequences. Distinct from most existing work on discriminative models including the work of [8, 14], our AdaBoost.MRF can also handle hidden variables (missing labels) - an important enhancement, especially in a smarthome environments when reliable labels are sparely observed.

Conditional Markov random fields [7, 8, 13] are powerful modeling tools in computer vision since they can incorporate arbitrary, overlapping and long-range features. Unfortunately, maximum likelihood estimation (MLE) of parameters in the general MRFs is known to be intractable, except for simple tree structures, and some approximations have been introduced. One of the earliest and most popular methods is pseudo-likelihood [2], which is very efficient. The method, nevertheless, cannot handle missing state variables, which often happen in real situations. Sampling-based methods such as MCMC are theoretically attractive, but they are often impractical for extremely slow convergence. The state-of-the-art includes message passing algorithms such as Pearl's belief propagation (BP) and the more recently introduced method by Wainwright, Jaakkola and Willsky (WJW) [16], which are efficient but their convergence is still an unsolved problem. In addition, the WJW

*Bui is supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

inference has not been applied for learning in conditional MRFs.

In this paper, we tackle the parameter estimation problem using a boosting-based algorithm. Our proposed AdaBoost.MRF is based on a ranking-based multiclass boosting algorithm called AdaBoost.MR [12]. At each round, the AdaBoost.MRF selects the best trained spanning tree of the network based on the performance on weighted error. The data is adaptively re-weighted to address more hard-to-classify instances. Each selected spanning tree is weighted and unioned together to recover the original network. Finally, the parameters of the network are the convex combination of all selected tree parameters. Since our method works exclusively on trees, inference is very efficient and surely converged. We show that under mild assumptions, the AdaBoost.MRF is guaranteed to reach the unique optimum. Furthermore, since the AdaBoost.MRF considers all the variables in the MRFs, the hidden variables problem can also be effectively handled.

We apply the AdaBoost.MRF to learn the parameters of the DCRFs on the activity data obtained in a home video monitoring scenario. We compare our AdaBoost.MRF with the maximum likelihood method, which used BP and WJW as inference engines. To evaluate the effectiveness of the discriminative DCRFs against generative methods, we implement a variant of the layered hidden Markov models (LHMMs) [11], which has previously been applied for activity recognition. Differing from the original LHMMs, our variant can handle partially observed state variables to make it compatible with the DCRFs considered in this paper. We also show that the multilevel DCRFs perform better than the flat-CRFs as more information is encoded.

The paper will continue with a related background in Section 2. Section 3 introduces the AdaBoost.MRF algorithm and discusses its convergence. Section 4 shows how we model and learn multilevel activities with MRFs, followed by Section 5 to present the experimental results. Finally, Section 6 concludes the paper with future direction.

2. Related Work

Our work is based on conditional MRFs, also known as conditional random fields (CRFs) [7, 8, 13]. A Markov random field (Figure 1) is an undirected graph $G = (V, E)$ which represents a random joint state variable $y = (y_1, y_2, ..., y_n)$, where each y_s corresponds to a node s in the network. The CRFs specify the conditional distribution of the state variable given the observation x

$$p(y|x) = \exp(\langle \lambda, \Psi(x, y) \rangle - \Phi(x)) \quad (1)$$

where $\lambda = \{\lambda_k\}$ is parameter vector, $\Psi(x, y) = \sum_c \psi(x, y_c)$ for $\psi(x, y_c)$ is the local potential or feature, and $\Phi(x)$ is the log-partition function.

Inference in general MRFs is known to be intractable except for trees with limited tree-widths. For other structures, approximate methods such as mean fields and belief propagation (BP) must be used. A more recently proposed method by Wainwright, Jaakkola and Willsky (WJW) [16] also offers an interesting alternative to compute the so-called pseudo-marginals based on minimising the upper bound of the log-partition function. Like BP, the WJW is an efficient message passing scheme. However, the BP is not guaranteed to converge and there has not been any formal proof nor extensive empirical evaluation of the WJW. Our AdaBoost.MRF, in contrast, works on tree inference and thus is guaranteed to converge with known analytical complexity. Besides, to the best of our knowledge, we are the first to perform learning using WJW in the conditional MRFs setting.

Work using MRFs to model activities is still limited (e.g. see [13]) and most applications employ directed models, such as hidden Markov models (HMMs). However, HMMs are flat models that are not capable of modeling multilevel activities directly. Recent extensions to HMMs to deal with such complex patterns include the abstract HMMs [3], the hierarchical HMMs [10], and the layered HMMs [11]. However, these attempts appear to ignore the case of partially observed states which are considered in this paper.

Learning in Markov random fields is mainly based on the maximum likelihood (ML) principle. Boosting [12] can provide an alternative for the task. It is originally proposed as a meta-learning method to combine weak classifiers into a strong one by iteratively addressing more hard-to-classify data instances. Recently, it has been interpreted in terms of functional gradient [9], a framework we adopt in this paper. Note that the AdaBoost.MR proposed in [12] addresses only simple classification, where the data of interest does not have any structure. For structured data such as CRFs, boosting has been applied in [15], but the algorithm relies on the BP for approximate inference and does not address the missing variables. Similarly, work in [4] is limited to tractable CRFs.

3. AdaBoost.MRF

In this section we describe AdaBoost.MRF, the boosting algorithm for parameter estimation of general Markov random fields. We consider the general case where the state label y may have a hidden component h and a visible component v.

3.1. Boosted Markov Random Forests

Given a data point x, the task of classification is often to find the best label y^* that maximises some function $F(x, y)$

$$y^* = \arg\max_y F(x, y) \quad (2)$$

IEEE COMPUTER SOCIETY

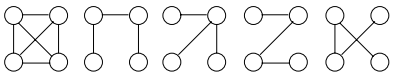


Figure 1. An example of Markov network (left-most) and some spanning trees (right).

where $F(x, y)$ is known as the strong learner in the boosting literature. As in the usual boosting setting, we formulate an objective function, which in this paper is based on the exponential loss of AdaBoost.MR [12]. Given training data pairs (x^i, y^i), $i = 1, .., D$, the loss is $L_{exp} = \sum_i \sum_y \exp(F(x^i, y) - F(x^i, y^i))$. Since for an instance i, we are given only the visible part v^i of y^i, we estimate the *incomplete* loss

$$L_{inco} = \sum_i \sum_v \exp(F(x^i, v) - F(x^i, v^i)) \tag{3}$$

The main difference from most of the previous boosting work is that the number of classes in our cases can be extremely huge, e.g. $|Y| = S^n$ where S is the state size for each node in the network.

In each round t of boosting, the strong learner $F_t(x, v)$ is updated by adding a 'weak-learner' $f_t(x, v)$ to the previous $F_{t-1}(x, v)$ as $F_t(x, v) = F_{t-1}(x, v) + \alpha_t f_t(x, v)$, where α_t is the weight of each weak learner in the ensemble. The weak learner and its weight are chosen to minimise the loss in (3), i.e.

$$(f_t, \alpha_t) = \arg\min_{f,\alpha} L_{inco} \tag{4}$$

Since we are interested in estimating the distribution $p(v|x)$, we may choose the weak learner as $f(x, v) = \log p(v|x)$. However, if we use the distribution defined over the general Markov networks, the computation of the weak learner itself is intractable. To address this issue, we propose the use of weak learners which in this case we choose to be spanning trees as weak approximations to the whole network. Thus, each learner is "weak" because it is a crude approximation of the true model with moderate complexity

$$f_t(x, v) = \log p_t(v|x) \tag{5}$$

This choice also allows incorporation of the hidden information since $f_t(x, v) = \log p_t(v|x) = \log \sum_h p_t(v, h|x)$. The strong learner F is a collection of trees, and we call our boosting method AdaBoost.MRF (AdaBoosted Markov Random Forests). Figure 1 shows a simple example of a four-node network and some spanning trees.

3.2. Loss bound using Hölder's inequality

In this subsection, we address further the intractability of the exponential loss in (3) by a tractable upper bound using tree likelihood. We have

$$\begin{aligned} L_{inco} &= \sum_{i,v} \exp\{\sum_{j=1}^{t} \alpha_j (\log p_j(v|x^i) - \log p_j(v^i|x^i))\} \\ &= \sum_i \frac{\sum_v \prod_j p_j(v|x^i)^{\alpha_j}}{\prod_j p_j(v^i|x^i)^{\alpha_j}} \end{aligned} \tag{6}$$

Although the evaluation of each weak learner is tractable, the sum over all visible variables in the numerator is unfortunately intractable, except for a special case that all selected spanning trees are the same.

Fortunately, there exists a technique that helps to remove the summation in the numerator. The idea is to apply the Hölder's inequality [5, Theorem 11] to the numerator $\sum_v \prod_j p_j(v|x^i)^{\alpha_j} \le \prod_j (\sum_v p_j(v|x^i)^{\alpha_j r_j})^{1/r_j}$, where $\sum_j 1/r_j = 1$ and $r_j > 0$. If we can ensure that $\alpha_j > 0$ and $\alpha_j r_j = 1 \; \forall j$, or $\sum_j \alpha_j = 1$, we obtain

$$\begin{aligned} L_{inco} &\le \sum_i \frac{\prod_j (\sum_v p_j(v|x^i))^{\alpha_j}}{\prod_j p_j(v^i|x^i)^{\alpha_j}} = \sum_i \frac{1}{\prod_j p_j(v^i|x^i)^{\alpha_j}} \\ &= \sum_i \exp(-\sum_j \alpha_j \log p_j(v^i|x^i)) \qquad (7) \\ &= \sum_i \exp(-F_t(x^i, v^i)) = L_H \qquad (8) \end{aligned}$$

since $\sum_v p_j(v|x^i) = 1$, $\forall i, j$. It can be seen that the new bound is tractable to evaluate, and is also convex so that a global minimum exists. We use the new loss L_H for learning. The domain of L_H is therefore a linear space of functions [9], which are $\{f_t(x^i, v) = \log p_t(v|x^i)\}$ in our case.

The requirement $\sum_j \alpha_j = 1$ can be met by defining the following ensemble

$$\begin{aligned} F_t(x, v) &= (1 - \alpha_t) F_{t-1}(x, v) + \alpha_t f_t(x, v) \qquad (9) \\ &= F_{t-1}(x, v) + \alpha_t h_t(x, v) \text{ where} \qquad (10) \\ h_t(x, v) &= f_t(x, v) - F_{t-1}(x, v) \qquad (11) \end{aligned}$$

Each previous weak learner's weight is scaled down by a factor of $1 - \alpha_t$ as $\alpha'_j \leftarrow \alpha_j(1 - \alpha_t)$, for $j = 1, ..., t - 1$, so that $\sum_{j=1}^{t-1} \alpha'_j + \alpha_t = \sum_{j=1}^{t-1} \alpha_j(1 - \alpha_t) + \alpha_t = 1$, since $\sum_{j=1}^{t-1} \alpha_j = 1$.

3.3. Weak learners, convergence and complexity

We now show how to carry out the stepwise optimisation in (4) with the incomplete loss replaced by the upper bound $L_H(F)$ in (8).

The loss $L_H(F)$ as a function of $F(x, v)$ can be minimised by moving in the gradient descent direction

$$\nabla L_H(x^i, v) = \begin{cases} -\exp(-F_{t-1}(x^i, v^i)) & \text{if } v = v^i \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

IEEE COMPUTER SOCIETY

However, as the functional gradient ∇L_H and and the functional direction h in (10) may not belong to the same function space, direct optimisation may not apply. In [9] the authors propose to find the best h_t pointing to the decreasing direction of L_H, i.e.

$$h_t = \arg\min_h \langle \nabla L_H, h \rangle < 0 \tag{13}$$

The step size α_t is determined using a line search or by setting it to a small constant $\in (0, 1)$.

Let $w_{i,t-1} \propto \exp(-F_{t-1}(x^i, v^i))$ be data weights, i.e. $\sum_i w_{i,t-1} = 1$. Substituting (12) into (13), we have

$$h_t \quad = \quad \arg\min_h \sum_i -w_{i,t-1} h(x^i, v^i) \tag{14}$$

As $h(x^i, v^i) = f(x^i, v^i) - F_{t-1}(x^i, v^i)$, minimising with respect to $h(x^i, v^i)$ and $f(x^i, v^i)$ is equivalent since $F_{t-1}(x^i, v^i)$ is a constant. Recall from (5) that $f(x^i, v^i) = \log p_\tau(v^i|x^i; \lambda_\tau)$, this minimisation translates to selecting the best tree t and its parameters λ_t as follows[1]

$$(t, \lambda_t) \quad = \quad \arg\max_{\tau, \lambda_\tau} \sum_i w_{i,t-1} \log p_\tau(v^i|x^i; \lambda_\tau) \tag{15}$$

Our final result has a satisfying interpretation: *the functional gradient descent step tries to solve the maximum re-weighted log-likelihood problem (15) for each tree, and select the best tree with the largest re-weighted log-likelihood*. As boosting proceeds, some trees may be more likely to be selected than others, so the accumulated weights of trees may be different.

As with the standard boosting [12], the data distribution is iteratively updated as

$$w_{i,t} \quad \propto \quad w_{i,t-1} \exp(-\alpha_t h_t(x^i, v^i)) \tag{16}$$

where h_t is the new learner added to the ensemble in (10).

Since $\alpha_t > 0$, the weight increases if $h_t = f_t - F_{t-1} < 0$. The new interpretation is that *for a given data instance i, if the new weak learner f_t is less likely than the average of previous weak learners F_{t-1}, the AdaBoost.MRF will increase the weight for that data instance*. This is different from the usual boosting behaviour, where the data weight increases if the strong learner fails to correctly classify the instance. The AdaBoost.MRF seems to maximise data likelihood rather than to minimise the training error, and this is particularly desirable for density estimation.

Since L_H is convex and twice differentiable, it is known that and if L_H is Lipschitz continuous and the search direction satisfies the condition in (13), our algorithm is guaranteed to converge to the global minimum [1, Proposition 1.2.3]. The algorithm terminates when we cannot find any weak learner h that satisfies the condition in (13).

[1] With slight abuse of notation, the same t is used for both the boosting step and the selected tree at that step.

The running time of AdaBoost.MRF scales linearly in number of trees R, each of which (Section 2) takes $\mathcal{O}(2|V|S^2)$ inference time. If we only consider limited spanning trees, just enough to cover the whole network, then R can be quite moderate. For example, for a fully connected network, we just need $R = |V|$, and in a grid-like network (Figure 3a), $R = 2$ is enough (Figure 4).

3.4. Combining the parameters

Up to this point, we have successfully estimated the parameters of individual trees, and thus the strong learner in the boosting sense, which may be enough for classification purposes. However, our ultimate goal is to (approximately) estimate the parameters of the original network, which is a superimposition of individual trees. This subsection argues for a sensible method for such an approximate estimation.

Recall that $F(x, y) = \sum_t \alpha_t f_t(x, y)$ and $f_t(x, y) = \log p_t(y|x)$, and $F(x, y) = \sum_t \alpha_t \log p_t(y|x)$. Assume that the tree distribution also belongs to the exponential family in (1) with different parameters λ_t and the same potential function $\Psi(x, y)$. We require that the parts of the parameters λ_t, which correspond to cliques outside the trees to be zero. Thus

$$F(x, y) = \langle \sum_t \alpha_t \lambda_t, \Psi(x, y) \rangle - \sum_t \alpha_t \Phi_t(x) \tag{17}$$

Thus, the label y^* returned by the strong learner in (2) becomes $y^* = \arg\max \langle \sum_t \alpha_t \lambda_t, \Psi(x, y) \rangle$. Obviously, y^* should also be the MAP assignment of the model defined by $y^{MAP} = \arg\max_y p(y|x) = \arg\max \langle \lambda, \Psi(x, y) \rangle$, i.e. $y^* = y^{MAP}$. One natural way is to set λ as the ensemble parameters $\lambda = \sum_t \alpha_t \lambda_t$ so that

$$p(y|x) \quad \propto \quad \exp \langle \sum_t \alpha_t \lambda_t, \Psi(x, y) \rangle \propto \prod_t p_t(y|x)^{\alpha_t} \tag{18}$$

$$p(y|x) \quad = \quad \frac{\prod_t p_t(y|x)^{\alpha_t}}{\sum_y \prod_t p_t(y|x)^{\alpha_t}} \tag{19}$$

Thus, the combined model is a Logarithmic Opinion Pool (LogOP) [6], a special case of the more general ensemble framework. Each model $p_t(y|x)$ is an expert to provide an estimate of the true distribution $q(y|x)$. The aggregator $p(y|x)$ is indeed a minimiser of the weighted sum of Kullback-Leibler divergences between the $q(y|x)$ and each $p_t(y|x)$ [6]

$$p(y|x) = \arg\min_{q(y|x)} \sum_t \alpha_t \sum_y q(y|x) \log \frac{q(y|x)}{p_t(y|x)} \tag{20}$$

The work of [6] shows that $p(y|x)$ is closer to the true distribution $q(y|x)$ than the average of all individual experts $p_t(y|x)$. Our boosting algorithm can be seen as an estimator of the weighting factors $\{\alpha_t\}$.

The AdaBoost.MRF is summarised in Figure 2.

IEEE COMPUTER SOCIETY

Input: $i = 1, ..., D$ data pairs, graphs $\{G_i = (V_i, E_i)\}$
Output: parameter vector λ
Begin
Select spanning trees for each data instance
Initialise $\{w_{i,0} = \frac{1}{D}\}$, and $\alpha_1 = 1$
For each boosting round $t = 1, 2, \ldots$
Train all trees given weighted data $\{w_i\}$
*/*Select the best tree distribution*/*
$f_t = \arg\max_{\tau,\lambda_\tau} \sum_i w_{i,t-1} \log p_\tau(v^i|x^i; \lambda_\tau)$
$h_t = f_t - F_{t-1}$
If $\sum_i w_{i,t-1} h_{i,t} \leq 0$ **Then** go to Output
If $t > 1$ **Then** select the step size $0 < \alpha_t < 1$
*/*Update the strong learner*/*
$F_t = (1 - \alpha_t)F_{t-1} + \alpha_t f_t$
*/*Scale down the previous learner weights*/*
$\alpha_j \leftarrow \alpha_j(1 - \alpha_t)$, for $j = 1, ..., t - 1$
*/*Update the data weight*/*
$w_{i,t} = \frac{1}{W_t} w_{i,t-1} \exp(-\alpha_t h_{i,t})$
End
Output $\lambda = \sum_t \alpha_t \lambda_t$
End

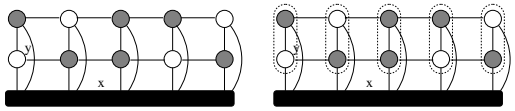
Figure 2. AdaBoost.MRF - AdaBoosted Markov Random Forests.

3.5. AdaBoost.MRF as guided search for MLE

As we rely on the boosting capacity to boost very weak learners to a strong one, we do not need to reach the maximum of the weighted log-likelihood in each round. We can simply run a few training iterations and take the partial results as long as the condition in (13) is met. To speedup the learning, we can initialise the parameters for each weak learner to the previously learned values. This procedure has an interesting interpretation for tree-structured networks. As we do not have to select the best spanning trees anymore, the algorithm is simply to optimise the re-weighted log-likelihood in a stage-wise manner. We argue that this approach can be attractive because more information from the data distribution can be used to guide the MLE, and it can create more diverse weak classifiers.

4. MRFs for Multilevel Activities

Our problem of interest is to model human behaviors in a smart environment at multiple levels of abstraction. Behaviours are naturally acted out in a hierarchical structure and we consider two-level of abstractions in this work using a two-layer dynamic conditional random field (DCRF) [14]. The bottom level presents primitive or atomic activities such as *go-to-cupboard* or *at-the-fridge*. Higher-order activities are captured at the higher level such as *having-snack* or *short-meal*. Differing from the original setting of the DCRF in [14], we allow some missing labels in our model, and thus we call the model the partially hidden DCRF (ph-DCRF) (Figure 3). We note that although the two-level DCRF is considered in this paper, the same construction can be generalized to model more complex semantics with richer levels of hierarchy and temporal interactions.



(a) ph-DCRF (b) Collapsed ph-DCRF

Figure 3. (a): The partially hidden DCRF (ph-DCRF), and (b): The equivalent chain CRF. Filled circles and bars are data observations, empty circles are hidden labels, shaded labels are the visible.

Given the training data, we first learn the parameters and then use it for annotating and segmenting unseen data. We now describe and compare some alternatives to the AdaBoost.MRF for parameter learning.

Part of our AdaBoost.MRF (Figure 2) requires parameter estimation using maximum likelihood (ML) of all trees of the ph-DCRFs. Another approach is to apply ML directly to the original network. Recall that $y = (v, h)$, the conditional incomplete log-likelihood is

$$\mathcal{L}(\lambda) = \log p(v|x) = \Phi(v, x) - \Phi(x) \tag{21}$$

The gradient $\frac{\partial \mathcal{L}(\lambda)}{\partial \lambda_k}$ follows as

$$\sum_{c,h_c} p(h_c|v_c, x)\psi_k(v_c, h_c, x) - \sum_{c,y_c} p(y_c|x)\psi_k(y_c, x) \tag{22}$$

Thus the computation of the gradient of log-likelihood reduces to computing the clique marginals. We assume only singleton and pairwise cliques, so the marginals can be estimated exactly and efficiently for our trees.

For the original ph-DCRF, exact estimation of marginals can be carried out by collapsing all the states at the current time into a mega-state (see Figure 3b) and performing a *forward-backward* procedure, which is infeasible for deep models. Approximate inference using the BP and WJW [16] methods has the complexity of $\mathcal{O}(2I|E|S^2)$, where I is the number of message passing rounds, $|E|$ is the number of edges in the network, and S is the state size per node. However, the number of rounds I until convergence if it does is not known analytically, and there has not been any theoretical estimate of it yet.

IEEE COMPUTER SOCIETY

In our AdaBoost.MRF, inference in the trees takes $\mathcal{O}(2|V|S^2)$ time, where $|V|$ is the number of nodes in the network. Thus, for D data instances, and R trees, the AdaBoost.MRF costs $\mathcal{O}(4DR|V|S^2)$ in total time for each gradient evaluation since we need to take both $\Phi(v,x)$ and $\Phi(x)$ into account. Similarly, the BP and WJW-based ML requires $\mathcal{O}(4DI|E|S^2)$ time. As for fully connected networks, $|E| = \frac{1}{2}|V|(|V|+1)$ while for the grid DCRFs, $|E| \approx 2|V|$, if we take only $R = |V|$ trees for the former case, and $R = 2$ for the latter case, the total complexity per gradient evaluation of the BP and WJW-based ML and the AdaBoost.MRF will be similar up to a constant I. We summarise the complexities in Table 1.

BP/WJW	AdaBoost.MRF
$\mathcal{O}(4DI\|E\|S^2)$	$\mathcal{O}(4DR\|V\|S^2)$

Table 1. Complexity per gradient evaluation.

5. Experimental Results

We consider a smart house environment for the elderly and apply the AdaBoost.MRF to the problem of learning and annotation of activities of daily livings (ADLs). The dataset was collected in our previous work [10], which captures some daily routines of actions performed in the kitchen and dining room. There are 45 video sequences for training and 45 sequences for testing. The observations are sequences of noisy coordinates of the actor walking in scene acquired using a background subtraction tracking algorithm. We consider 3 complex activities (states) at the top level: $\{1\}$ *short-meal,* $\{2\}$ *have-snack,* $\{3\}$ *normal-meal* and 12 primitive activities at the bottom level: $\{i\}$ *Door→Cupboard,* $\{ii\}$ *Cupboard→Fridge,* $\{iii\}$ *Fridge→Dining chair,* $\{iv\}$ *Dining chair→Door,* $\{v\}$ *Door→TV chair,* $\{vi\}$ *TV chair→Cupboard,* $\{vii\}$ *Fridge→TV chair,* $\{viii\}$ *TV chair→Door,* $\{ix\}$ *Fridge→Stove,* $\{x\}$ *Stove→Dining chair,* $\{xi\}$ *Fridge→Door,* $\{xii\}$ *Dining chair→Fridge*. Each complex activity is comprised of some primitive activities, and states at each level can freely transit to each other but generally we do not have this knowledge at hand for our experiment.

For evaluating the aspect of missing labels, we randomly provide half the labels for each level during training. For testing, the MAP assignments resulted from Pearl's loopy max-product algorithm are compared against the ground-truth.

5.1. Feature extraction

With the data described above, the input to the DCRFs is simply sequences of coordinates. At each time slice γ, we extract a vector of five elements from the observation sequence $g(x,\gamma) = (X, Y, u_X, u_Y, s = \sqrt{u_X^2 + u_Y^2})$, which correspond to the (X,Y) coordinates, the X & Y velocities, and the speed respectively. To fully specify the model, we consider three types of feature functions for the potentials of the network: (a) *data-association* corresponding to node potentials, (b) *temporal-relation* corresponding to state transition potentials at the same level, and (c) *cross-semantic-relation* corresponding to parent-child potentials across different levels.

For the first feature set, we define the data-association features at the bottom level as

$$\psi_{l,m,\epsilon}(x, y_\gamma^2) := \delta[y_\gamma^2 = l] g_m(x, \gamma + \epsilon) \tag{23}$$

where $m = 1, ..., 5$ is the index of components of $g(x,\gamma)$, $\epsilon = -s_1, ..., 0, ..., s_2$ is the amount of look-ahead or look-back, for some positive integers s_1, s_2, $y_\gamma^2 = 1, ..., 12$ is the state (at level 2). We choose $s_1 = s_2 = 2$ for reasonable computation, so that the current primitive activity y_γ^2 is correlated with five surrounding observation features $g(x,\gamma)$. At the top level, however, instant information such as velocities offer limited help since the complex activities often span long periods. Instead of using the real coordinates (X,Y) for data association, we quantize them into 24 squares in the room. We also use much larger windows with $s_1 = s_2 = 20$. To avoid computational overhead, we take $\epsilon = -s_1, -s_1 + 5, ..., s_2 - 5, s_2$.

The second and third feature sets consist of simple indicator functions

$$\psi_{l_1,l_2}(y_{\gamma-1}^d, y_\gamma^d) := \delta[y_{\gamma-1}^d = l_1]\delta[y_\gamma^d = l_2] \tag{24}$$

for the second set, and

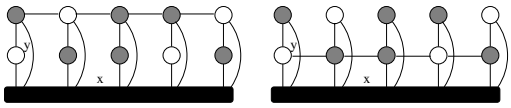
$$\psi_{l_{pa},l_{ch}}(y_\gamma^{d-1}, y_\gamma^d) := \delta[y_\gamma^{d-1} = l_{pa}]\delta[y_\gamma^d = l_{ch}] \tag{25}$$

for the third set, where $d = 1, 2$ is the depth level.

5.2. Spanning trees for AdaBoost.MRF

The AdaBoost.MRF algorithm described in Figure 2 requires the specification of a set of spanning trees which will be used as the weak classifiers. Given the grid structure considered in this experiment, there are many spanning trees that can be extracted. However, since the nature of our problem is about temporal regularities where the slice structure is repeated over time, it is natural to decompose the network into trees in a such a way that the structural repetition is maintained. With this hint, there are two most

IEEE COMPUTER SOCIETY



(a) Top process (b) Bottom process

Figure 4. (a,b) Two process view of the DCRF in activity modeling: (a) the complex activity, and (b) the primitive.

noticeable trees that stand out as shown in Figure 4, which roughly corresponds the top and bottom chains respectively.

With the same method, the number of trees for dynamic models which respect the Markov assumption is reduced drastically. If we impose further restrictions that each state can only interact with the level right above and right below it, then the number of trees can be manageable (e.g. see Figure 5 for another example).

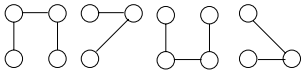


Figure 5. 2-slice structures of spanning trees for the DCRFs whose 2-slice structure is given in the left-most graph in Figure 1.

5.3. Segmentation and Annotation results

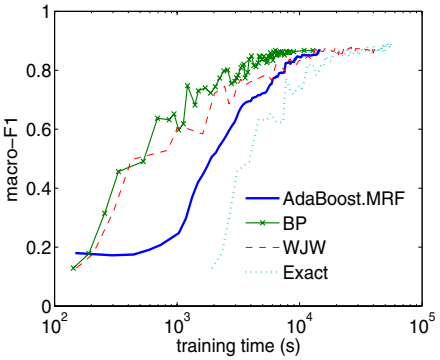


Figure 6. Macro-averaged F_1 scores at the bottom layer vs training time.

For comparing with the AdaBoost.MRF for the DCRFs, we implement ML learning methods based on BP, WJW and exact inference. We also evaluate the effectiveness of the DCRFs against the Layered HMMs (LHMMs) [11], where the output of the bottom HMM is used as the input for the top HMM. Since, it is difficult to encode rich feature information in the LHMMs without producing very large state space, we limit the LHMMs features to be the discretised positions and the differences between current position and the previous and next ones. Our new implementation of LHMMs differs from the original in [11] for each HMM has been extended to handle the partially observed states. To test whether adding more layers can improve the performance of the model, we run a simple *Flat-CRF* on the data at the lower level. All learning algorithms are initialised uniformly. For segmentation purposes, we report the macro-averaged F_1 scores on a per-label basis.

For parameter optimisation of the (re-weighted) log-likelihood, initially we used the limited memory quasi-Newton method (L-BFGS) as suggested in the CRF literature but it seems to be slower and it converges prematurely to poor solutions for the BP and the exact inference. The conjugate-gradient (CG) method works better in our experiments. For the Markov forests, we run for only two iterations of CG per boosting round with the initial parameters from the previously learned ones since we only need to meet the condition (13). The WJW inference loop is stopped if the messages have converged at the rate of 10^{-4} or after 100 rounds. It appears that the final performance of BP is sensitive to the choice of convergence rates, while it is fairly stable for the WJW. For example, the F_1 scores at the bottom level for BP are $0.84, 0.87$ and 0.82 corresponding to the rates of $10^{-3}, 10^{-4}$ and 10^{-5}, respectively. Below we report only the case of 10^{-4}, which appears to be the best both in terms of accuracy and speed. Learning algorithms for the DCRFs are stopped after 100 iterations if they have not converged at the rate of 10^{-5}.

The performance of the AdaBoost.MRF and its alternatives is reported in Figure 6 and Table 2, respectively. Overall, after enough training time, the AdaBoost.MRF performs comparably with the ML methods based on BP and WJW. The exact inference ML method gives slightly better result as expected but at the cost of much slower training time. However, it should be stressed that inference in our AdaBoost.MRF always converges, while it is not guaranteed in the BP and WJW and it is generally intractable in the exact method. The complexity per evaluation of the log-likelihood gradient is known and fixed for the AdaBoost.MRF, while for the BP and the WJW, it is generally dependent on the convergence criteria and how much the distribution is different from uniform. (see Table 1).

Table 2 also shows that the choice of discriminative model over the generative model in our activity recognition problem is justified. The LHMMs are worse than both the flat-CRFs at the bottom layer and the DCRFs at the top

Table 2. Macro-averaged F_1 scores for top and bottom layers.

Algorithm	Top-layer	Bottom-layer
AdaBoost.MRF	0.98	0.87
BP	0.99	0.87
WJW	0.98	0.87
Exact	0.98	0.88
LHMM	0.88	0.67
Flat-CRFs	-	0.78

layer. Furthermore, the DCRFs variants are more consistently accurate than the flat CRFs. The result is consistent with that in [14]. This can be explained by the fact that more information is encoded in the DCRFs.

Figure 7 shows the AdaBoost.MRF segmentation details of 22 randomly selected sequences which are concatenated together.

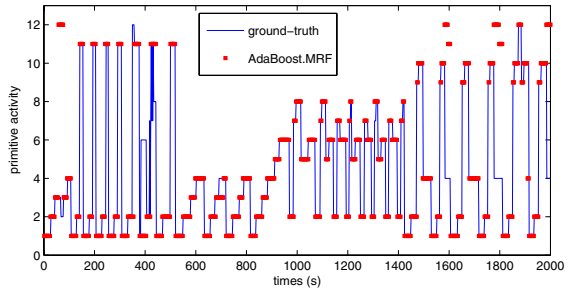


Figure 7. The segmentation compared with the ground-truth at the bottom level.

6. Conclusion

We have presented a novel method for using boosting in parameter estimation of the general Markov networks with hidden variables. The algorithm AdaBoost.MRF offers an efficient way to tackle the intractability of the maximum likelihood method by breaking the model into tractable trees and combining them to recover the original networks. We apply the algorithm to the new problem of multilevel activity recognition and segmentation using the recently proposed DCRFs.

We should stress that, however, the AdaBoost.MRF is not limited to the DCRFs but can be applied to arbitrary CRFs. In addition, not only it can discriminatively approximately estimate the conditional distributions $p(y|x)$, but also it can generatively learn the *joint* distributions $p(x, y)$.

Furthermore, in our experiments, it appears that the AdaBoost.MRF exhibits a structure learning behaviour since it may selectively pick some trees more frequently than others, giving higher weights to those trees. An important issue we have left unanswered is that how to automatically select the optimal tree at each round without knowing the set of trees in advance. We plan to investigate these aspects and the use of AdaBoost.MRF in wider range of applications.

References

[1] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachussets, 2 edition, 1999.

[2] J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussions). *Journal of the Royal Statistical Society Series B*, 36:192–236, 1974.

[3] H. H. Bui, S. Venkatesh, and G. West. Policy recognition in the Abstract Hidden Markov Model. *Journal of Artificial Intelligence Research*, 17:451–499, 2002.

[4] T. G. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *ICML*, Banff, Canada, 2004.

[5] G. Hardy, J. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, Cambridge, 2nd edition, 1952.

[6] T. Heskes. Selecting weighting factors in logarithmic opinion pools. In *Advances in NIPS*, volume 10, 1998.

[7] S. Kumar and M. Hebert. Discriminative Random Fields: A discriminative framework for contextual interaction in classification. In *ICCV*, 2003.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[9] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press, 2000.

[10] N. Nguyen, D. Phung, S. Venkatesh, and H. H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov models. In *Proc. CVPR*, San Diego, CA, Jun 2005.

[11] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *Fourth IEEE Int. Conf. on Multimodal Interfaces*, pages 3–8, 2002.

[12] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[13] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *ICCV*, 2005.

[14] C. A. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic Conditional Random Fields: factorized probabilistic models for labeling and segmenting sequence data. In *ICML*, 2004.

[15] A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *NIPS 17*, pages 1401–1408. 2005.

[16] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Trans. on Information Theory*, 51:2313–2335, Jul 2005.