Well-M^{3}N: a Maximum-Margin Approach to Unsupervised Structured Prediction

Shaukat Abidi, Massimo Piccardi, *Senior Member, IEEE*, Ivor W. Tsang, and Mary-Anne Williams

***Abstract*—Unsupervised structured prediction is of fundamental importance for the clustering and classification of unannotated structured data. To date, its most common approach still relies on the use of structural probabilistic models and the expectation-maximization (EM) algorithm. Conversely, structural maximum-margin approaches, despite their extensive success in supervised and semi-supervised classification, have not raised equivalent attention in the unsupervised case. For this reason, in this paper we propose a novel approach that extends the maximum-margin Markov networks (M^{3}N) to an unsupervised training framework. The main contributions of our extension are new formulations for the feature map and loss function of M^{3}N that decouple the labels from the measurements and support multiple ground-truth training. Experiments on two challenging segmentation datasets have achieved competitive accuracy and generalization compared to other unsupervised algorithms such as k-means, EM and unsupervised structural SVM, and comparable performance to a contemporary deep learning-based approach.**

***Index Terms*—Structured prediction, unsupervised training, convex relaxation, maximum-margin Markov networks, Well-SVM.**

I. Introduction

Unsupervised structured prediction refers to the use of classifiers that can explicitly model the inherent "structure" in the data and be trained without supervision. It is an increasingly useful framework that can be applied to tasks as diverse as the clustering of activities and trajectories, the segmentation of time series, the modelling of topics in text, the analysis of graphs in social networks and many more. With such structured data growing rapidly both in volume and spread, unsupervised structured prediction is becoming more urgent and in focus.

Conversely, unsupervised prediction of "ordinary" data (independent and identically distributed, or i.i.d.) has been covered by an extensive literature. Baseline approaches include k-means and mixture models learned in probabilistic frameworks [1], [2]. In recent years, maximum-margin approaches have also been used for unsupervised learning. An early example is unsupervised SVM that extends the conventional, supervised support vector machine to unsupervised prediction [3]. The challenge with this task is that the number of possible labelings is exponential in the number of samples and the problem is notoriously NP-hard. Therefore, solutions must rely on either local optima or relaxations of the original problem. The basic local approach consists of alternating optimizations where a step of SVM learning gives turns to a step of inference of the latent variables [3]; we refer to this approach as alternate-steps unsupervised SVM in the following. While local solutions are computationally efficient, they tend to show early convergence and have been reported as highly sensitive to the initialization [4]. A different support vector approach was proposed in [5], [6] in terms of minimum-enclosing hyperspheres. Determining the minimum-enclosing hypersphere of a set of points requires solving a quadratic objective with quadratic constraints and is a well-known convex problem. However, assigning the points to clusters remains a combinatorial problem and the overall solution is still only locally optimal. Another solution using conventional separating hyperplanes was proposed in [7], [8], [9], leveraging a Laplacian regression loss in place of the usual hinge loss. The use of the Laplacian loss helps mollify the issues of premature convergence. More recently, [10] has proposed a global solution of polynomial complexity; however, its soft-margin formulation differs from the conventional soft-margin SVM based on the hinge loss. Various heuristics based on incremental feature selection [11], recurrent local search [12], and classifier ensembles [13] have also been proposed. Given that combinatorial approaches are intrinsically local, exploring convex relaxations offers an appealing alternative. Xu *et al*. in [14] have proposed a convex relaxation of unsupervised SVM based on semidefinite programming (SDP), and have later extended it to the multi-class and structured cases [15], [16]. However, the computational complexity of SDP is much higher than that of quadratic programming and seems prohibitive even for datasets of limited size. More recently, Li *et al*. in [17], [18] have proposed a different relaxation based on the minimax theorem. This relaxation, called Weakly Labeled SVM or Well-SVM for short, has been proven to be tighter than the SDP relaxation of [14]. In addition, it can be implemented in terms of multi-kernel SVM and is efficient and scalable. Over the last few years, also a number of deep learning-based clustering approaches have been proposed [19]. All these approaches leverage deep autoencoders to generate a new, nonlinear feature space that can benefit the clustering step [20], [21], [22]. Other recently-proposed deep learning approaches have more restricted application since they either require knowledge of the output statistics [23] or are limited to categorical variables (i.e., words) as inputs [24].

In the case of structured data, the most established approach for unsupervised training is certainly expectation-maximization (EM) [25]. Paralleling the i.i.d. case, maximum-margin approaches have also been used to train structured predictors. For instance, structural SVM (SSVM) [26]

The authors are with University of Technology Sydney, Faculty of Engineering and IT, e-mail: {SyedShaukatRaza.Abidi, Massimo.Piccardi, Ivor.Tsang, Mary-Anne.Williams}@uts.edu.au.

Manuscript received November 15, 2017.

and maximum-margin Markov networks (M^3N) [27] solve maximum-margin objectives for supervised structured prediction. These approaches, too, can be easily extended to the unsupervised case using the alternate-steps approach. The main problems, again, rest with its premature convergence and sensitivity to initialization. For this reason, in this paper we propose a novel unsupervised training algorithm that extends the convex relaxation of [18] from the i.i.d. case to the case of structured prediction. The main contribution of the proposed extension is an original formulation of the feature map and loss function of M^3N that it is suitable for integration in the SVM relaxation. Experiments carried out on two challenging time-segmentation datasets show competitive performance compared to other unsupervised algorithms such as k-means, EM and alternate-steps unsupervised structural SVM, and comparable performance to a deep learning-based approach.

The rest of this paper is organized as follows: Section II summarizes Well-SVM and M^3N to set the ground for the proposed method. Its formulation - named Well-M^3N for analogy - is presented in Section III, while the experimental results are described in Section IV. The Conclusion summarizes the main findings.

II. Background

In this section, we introduce the main background materials, namely 1) Well-SVM, a convex relaxation for unsupervised SVM (Section II-A), and 2) the maximum-margin framework for structured prediction (Section II-B).

A. Well-SVM

Let us note the constrained objective of a conventional, soft-margin SVM as SVM(w, y), with w the parameter vector and y the labeling for the sample set. The problem entailed by unsupervised SVM is:

$$\min_{y} \min_{w} \text{SVM}(w, y) \tag{1}$$

that is, finding the labeling returning the minimum of all SVM primal problems, $\min_w SVM(w, y)$. This equates to finding the two clusters with maximum soft-margin between them. However, this formulation may degenerate in assigning all the samples to only one cluster with a hypothetically infinite margin. It is then necessary to limit the search over y to a set of feasible, "balanced" labelings which we will note as β hereafter, i.e., $y \in \beta$. An equivalent solution to (1) can be obtained by replacing the internal minimization with its dual:

$$\min_{y} \max_{\lambda} G(\lambda, y) \tag{2}$$

where λ is the vector of the dual variables. The idea behind the Well-SVM relaxation presented in [17], [18] is to exchange the order of $\max_\lambda$ and $\min_y$ to instead solve:

$$\max_{\lambda} \min_{y} G(\lambda, y) \tag{3}$$

Li *et al*. in [17] have proved that this relaxation is tighter than the earlier relaxation based on semidefinite programming proposed by Xu *et al*. [14]. For this reason, Well-SVM can be regarded as the tightest known relaxation of unsupervised SVM. The inner combinatorial minimization in (3) can be posed in terms of a continuous, constrained maximization:

$$\begin{aligned} &\max_{\theta} \ \theta \\ &s.t. \ \theta \leq G(\lambda, {}^l y) \quad \forall \, {}^l y \in \beta \end{aligned} \tag{4}$$

with corresponding Lagrangian:

$$\theta + \sum_{l : {}^l y \in \beta} \mu^l \left(G(\lambda, {}^l y) - \theta \right) \tag{5}$$

where $\mu^l \geq 0$ is the dual variable associated with the l-th constraint in (4) and ${}^l y$ is the l-th labeling. We have chosen to use a prescript apex for the labelings since their right apex and index will be used for other indexings. Differentiating (5) in θ, equating to zero and replacing the result in (3) leads to:

$$\max_{\lambda} \min_{\mu} \sum_{l : {}^l y \in \beta} \mu^l G(\lambda, {}^l y) \tag{6}$$

subject to constraints $\sum_l \mu^l = 1, \mu^l \geq 0 \ \forall l$. Since (6) meets the KKT conditions, the max-min order can be swapped to obtain:

$$\min_{\mu} \max_{\lambda} \sum_{l : {}^l y \in \beta} \mu^l G(\lambda, {}^l y) \tag{7}$$

Li *et al*. in [18] have shown that (7) can be solved as an instance of multi-kernel learning (MKL) and by using conventional SVM solvers. The solution is described in detail in Section III.

B. Maximum-Margin Structured Prediction

Structured prediction aims to predict multiple, dependent classes from an available measurement vector. Given a measurement, x, a prediction or labeling consists of a graph of class labels, y. The family of structured prediction we address is the generalized linear model where the score for an $\{x, y\}$ pair is given by $w^\top \psi(x, y)$, with w a parameter vector and $\psi(x, y)$ a feature function of x and y that embeds the structure in the data. With this model, prediction is formally expressed as:

$$\bar{y} = \operatorname*{argmax}_{y} w^\top \psi(x, y) \tag{8}$$

Model w can be learned, among others, by leveraging a maximum-margin approach. Given a training set, $\{x_i, y_i\}, i = 1 \dots N$, this learning objective can be written as:

$$\begin{aligned} &\min_{w, \xi} \ \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \xi_i \\ &s.t. \ w^\top \delta\psi_i(y) \geq \Delta(y_i, y) - \xi_i, \\ &\qquad i = 1 \dots N, \ \ \forall y \in \mathcal{Y} \end{aligned} \tag{9}$$

where the minimization is over the usual SVM trade-off between a regularization term, $\|w\|^2$, and the hinge loss over the training set, $\sum_{i=1}^{N} \xi_i$. In the constraints, function $\delta\psi_i(y) = \psi(x_i, y_i) - \psi(x_i, y)$ computes the difference of the feature functions for the ground-truth labeling, y_i, and another labeling, y. The learning objective is to ensure that this difference is bigger than an appropriate margin, being chosen as the loss $\Delta(y_i, y)$ for predicting y when the ground truth is y_i.

The immediate impediment in approaching (9) directly is that the number of constraints, $|\mathcal{Y}|$, grows exponentially with the number of nodes in the graphs. This makes direct solutions infeasible even for graphs of a relatively small size. Two main relaxations have been proposed in the literature to mollify this problem, namely *structural SVM* [26] and *maximum-margin Markov networks* (M^{3}N) [27]. Structural SVM ([26]) seeks an approximate solution that only satisfies a polynomial-size subset of the "most-violated" constraints. The solution is provenly ϵ-close to the full solution, with ϵ a constant that can be made arbitrarily small by an appropriate number of constraints. Conversely, M^{3}N restricts the feature and loss functions to be pairwise decomposable over the graph. In this way, the number of constraints drops to quadratic and the resulting minimization can be solved with the full set. For the integration proposed in this paper, we have carried out a preliminary set of experiments with both methods and noticed that M^{3}N had proven generally more accurate. For this reason, the remainder of this paper focuses on M^{3}N.

Through standard manipulation, the primal problem of (9) can be converted into its corresponding dual form:

$$\begin{aligned} &\max_{\alpha} \sum_{i,y} \alpha_{iy} \Delta(y_i, y) - \frac{1}{2} \sum_{i,y} \sum_{j,v} \alpha_{iy} \alpha_{jv} \delta\psi_i(y)^\top \delta\psi_j(v) \\ &s.t. \ \sum_{y} \alpha_{iy} = C, \forall i; \quad \alpha_{iy} \geq 0 \\ &\qquad i = 1 \ldots N, \ \ \forall y \in \mathcal{Y} \end{aligned} \tag{10}$$

where α_{iy} indicates the Lagrange multiplier for sample i and labeling y. The two conditions for the M^{3}N factorization of the dual (10) are: (i) the loss must be decomposable over the individual nodes of the labeling, i.e, $\Delta(y_i, y) = \sum_a I(y_i^a \neq y^a) = \sum_a \Delta t_i(y^a)$, where a is the index of an individual node; and (ii) the feature function for a labeling, y, must be pairwise-decomposable over the graph's edges i.e. $\delta\psi_i(y) = \sum_{a,b} \delta\psi_i(y^a, y^b)$. With such conditions, Taskar *et al*. in [27] have rewritten the dual problem using a set of dual variables over the nodes and edges defined as:

$$\begin{aligned} \lambda_i(y^a) \quad &= \sum_{y \sim [y^a]} \alpha_{iy} \qquad \forall a, \forall y^a, \ \ \forall i; \\ \lambda_i(y^a, y^b) \ &= \sum_{y \sim [y^a, y^b]} \alpha_{iy} \qquad \forall a, b, \forall y^a, y^b \ \ \forall i; \end{aligned} \tag{11}$$

where $y \sim [y^a]$ denotes a labeling with a fixed assignment of its a^{th} node, i.e., $[\ldots, y^a, \ldots]$. Similarly, $y \sim [y^a, y^b]$ is a labeling with a fixed assignment of its node pair (a, b), i.e., $[\ldots, y^a, \ldots, y^b, \ldots]$. From (11), it follows that $\lambda_i(y^a) = \sum_{y^b} \lambda_i(y^a, y^b)$. By using (11) to rewrite the linear and quadratic terms in (10), the dual becomes:

$$\begin{aligned} &\max_{\lambda} \ G(\lambda, y) = \\ &\max_{\lambda} \sum_{i} \sum_{a, y^a} \lambda_i(y^a) \Delta t_i(y^a) \\ &- \frac{1}{2} \sum_{i,j} \sum_{\substack{a,b \\ y^a, y^b}} \sum_{\substack{c,d \\ v^c, v^d}} \lambda_i(y^a, y^b) \lambda_j(v^c, v^d) \delta\psi_i(y^a, y^b)^\top \delta\psi_j(v^c, v^d) \\ &s.t. \ \sum_{y^a} \lambda_i(y^a, y^b) = \lambda_i(y^b); \ \ \sum_{y^a} \lambda_i(y^a) = C; \ \ \lambda_i(y^a, y^b) \geq 0. \end{aligned} \tag{12}$$

where, for consistency with (2-7), argument y in $G(\lambda, y)$ refers to the ground-truth labelings, not the predictions. By rewriting the dual in terms of the λ variables, the number of constraints becomes quadratic in the number of nodes in the graph, and thus manageable.

III. The Proposed Approach: Well-M^{3}N

In this section, we present the proposed integration of M^{3}N with Well-SVM. The integration is obtained by replacing the M^{3}N dual (12) in (7):

$$\min_{\mu} \max_{\lambda} \sum_{l: {}^l y \in \beta} \mu^l G(\lambda, {}^l y) \tag{13}$$

It is easy to show that, for any given set of labelings, (13) is an instance of weighted multi-kernel learning, with one kernel for each of the labelings, ${}^l y \in \beta$, weighed by weight μ^l. This is a convex problem in (μ, λ) that can be solved by algorithms such as SimpleMKL that alternate a step of solving for λ with fixed μ, with a step of solving for μ with fixed λ [28]. However, the solution also requires the generation of the labelings subject to the constraints of (4). Such a generation is not trivial and cannot be operated with the existing Well-SVM approach since $G(\lambda, {}^l y)$ only depends on ${}^l y$ implicitly. One of the main contributions of this paper, presented later in Section III-A, is an original remapping of the M^{3}N feature functions that makes such a dependence explicit and allows the generation of the labelings for (13).

Formally, the solution of (13) is obtained by iterating the following two main steps until convergence:

1) Find a labeling, ${}^l y$, violating constraint (4) for the current λ. Add ${}^l y$ to a working set of labelings, C_w. For the first iteration, an arbitrary labeling ${}^1 y$ can be used as the initial working set.
2) Solve multiple-kernel learning M^{3}N using all the labelings in the working set as the ground truth.

At its turn, an instance of multiple-kernel learning M^{3}N iterates through the following two sub-steps until convergence:

1) λ *step*: Using the current μ, solve for λ.
2) μ *step*: Using the current λ, solve for μ.

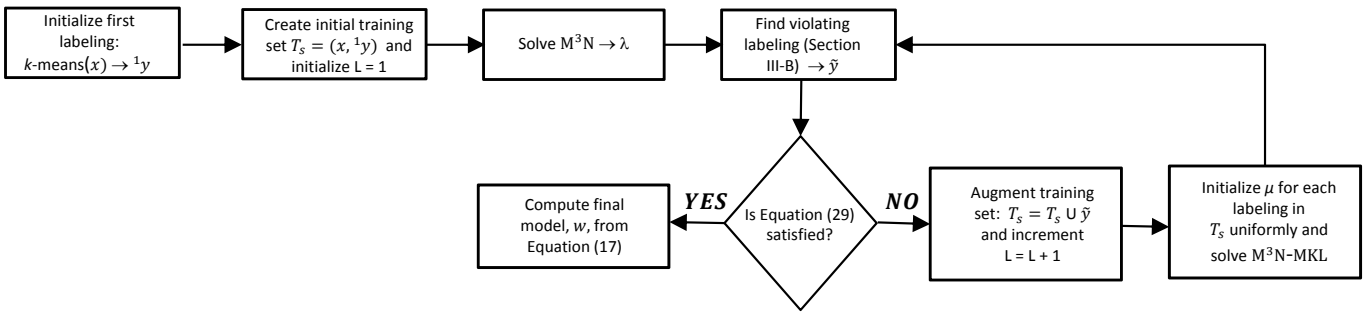

Fig. 1: Block diagram of the proposed Well-M^{3}N.

The overall procedure is called Well-M^{3}N and a block diagram of its main steps is shown in Fig. 1. Likewise, Algorithm 1 describes the algorithm using pseudo-code. We also provide an original, formal proof of convergence in Appendix A. In the following, we present the λ and μ steps in detail. Section III-A presents the feature mapping and loss function. Eventually, Section III-B presents the procedure for finding the violating labelings.

Algorithm 1 The main steps of Well-M^{3}N.

Initialization: ${}^1y; \mu^1 = 1.0$; $\gamma = 0.1$ ((27)) and $L = 1$
Solve Objective (12) and store λ
Find violating labeling (Section III-B)
repeat
  $L = L + 1$
  Initialize $\mu^{1:L} = \left[\mu^1, \ldots, \mu^L\right]^T$ uniformly
  Concatenate ground truths: ${}^{1:L}y = \left[{}^1y, \ldots, {}^Ly\right]$
  repeat
    Solve Objective (14) and store λ
    Compute $w^{1:L}$ ((16))
    Update $\mu^{1:L}$ ((17))
  until $\mu^{1:L}$ converges
  Find a violating labeling (Section (III-B))
until not (27)
Compute final w ((15)) for prediction

1) λ step: solving for λ with fixed μ: In the following, we show how to remap this step onto a conventional M^{3}N problem so as to solve it with any existing M^{3}N solver. We first introduce a fuller notation for the labelings: by ${}^ly_i^a$ we note the label of the a^{th} node in the l^{th} ground-truth labeling of graph i. We then make the following positions:

- $\tilde{\psi}_i(y^a, y^b) = [\sqrt{\mu^1}\psi_i(y^a, y^b), \ldots, \sqrt{\mu^L}\psi_i(y^a, y^b)]^\top$
  i.e., a concatenation of $\psi_i(y^a, y^b)$ L times, weighed by the square root of the μ coefficients;
- $\delta\tilde{\psi}_i(y^a, y^b) = \Big[\sqrt{\mu^1}\left(\psi_i({}^1y_i^a, {}^1y_i^b) - \psi_i(y^a, y^b)\right), \ldots, \sqrt{\mu^L}\left(\psi_i({}^ly_i^a, {}^ly_i^b) - \psi_i(y^a, y^b)\right)\Big]$
  i.e., a similar concatenation for the difference of feature functions with the L ground-truth labelings;
- $\tilde{\Delta}t_i(y^a) = \sum_{l=1}^{L} \mu^l \Delta t_i({}^ly_i^a, y^a)$
  i.e., the weighted sum of the loss of node y^a against ground-truth labeling ${}^ly_i^a$, for all labelings and all nodes. We assume that $\Delta t_i({}^ly_i^a, y^a)$ is the simple zero-one loss, $I({}^ly_i^a \neq y^a)$.

With the above positions, the inner maximization in (13) becomes:

$$\begin{aligned}
&\max_{\lambda} \sum_{l:y^l \in \beta} \mu^l G(\lambda, {}^ly) = \\
&\max \sum_i \sum_{a, y^a} \lambda_i(y^a) \tilde{\Delta} t_i(y^a) \\
&- \frac{1}{2} \sum_{i,j} \sum_{\substack{a,b \\ y^a, y^b}} \sum_{\substack{c,d \\ v^c, v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d)\delta\tilde{\psi}_i(y^a, y^b)^\top \delta\tilde{\psi}_j(v^c, v^d) \\
&s.t. \ \sum_{y^a} \lambda_i(y^a, y^b) = \lambda_i(y^b); \ \sum_{y^a} \lambda_i(y^a) = C; \ \lambda_i(y^a, y^b) \geq 0.
\end{aligned} \tag{14}$$

where the scalar products $\delta\tilde{\psi}^T \delta\tilde{\psi}$ and loss $\tilde{\Delta}t_i(y^a)$ incorporate the μ factor. This problem is formally identical to (12) and can therefore be solved with any existing M^{3}N solver. We have used CVX, a popular package for specifying and solving convex programs [29], [30].

2) μ step: solving for μ with fixed λ: Based on the representer theorem, the solution for the factorized dual (12) can be used to compute the primal model, w, as follows:

$$w = \sum_i \sum_{a,b} \sum_{y^a, y^b} \lambda_i(y^a, y^b)\delta\psi_i(y^a, y^b) \tag{15}$$

Therefore, with (14) the same approach can be used to compute a primal model, w^l, for each of the L ground-truth labelings:

$$w^l = \sum_i \sum_{a,b} \sum_{y^a, y^b} \lambda_i(y^a, y^b)\delta\psi_i^l(y^a, y^b) \quad l = 1 \ldots L \tag{16}$$

where $\delta\psi_i^l(y^a, y^b) = \sqrt{\mu^l}\left(\psi_i({}^ly_i^a, {}^ly_i^b) - \psi_i(y^a, y^b)\right)$.

Using the primal models from (16), the μ coefficients can eventually be updated in closed form ([18]) as follows:

$$\mu^l = \frac{\|w^l\|}{\sum_{l=1}^{L} \|w^l\|}, \;\; l = 1 \dots L \tag{17}$$

A. Feature Maps

The main challenge for finding violating labelings for (13) is to be able to rewrite objective (12) with an explicit dependence on y. To this aim, we start by encoding y by one-hot encoding, focussing on the case of binary nodes. For the binary variable of each node, we use encoding $[01]$ for the negative class and $[10]$ for the positive class. For an edge, we use encoding $[0001]$ if the edge connects a negative node with another negative node; encoding $[0010]$ for an edge connecting a negative and a positive node; and so forth. Such an encoding can also potentially accommodate multi-class nodes, yet at the cost of a significant increase in size of both the labelings and the required data structures.

For a graph with T binary nodes and E edges, there will be $2T$ binary variables to encode the nodes (*unary* variables) and $4E$ binary variables for the edges (*pairwise* variables), for a total of $2T+4E$ binary variables that we assume serialized into y. Index $t = 1 \dots T$ will be used to index the nodes and index $e = 1 \dots E$ for the edges. Without limitation of generality, we also assume one measurement per node, x^t, and that the feature function for a node-measurement pair is of the type $x^t y^t$.

Our first aim is to rewrite $\psi(x, y)$ as follows:

$$\psi(x, y) = h^\top y \tag{18}$$

in order to express the feature function as an explicit function of the labeling, y. Matrix h can be regarded as a *generalized measurement* that formats all the measurements in the graph such that the feature function can be expressed as $h^\top y$. In this way, the functional form is analogous to that of the unstructured case, xy, allowing using the same strategy for constraint generation. Moreover, this factorization can also be used in kernel matrices of structured predictors to separate the contribution of the labeling from that of the measurements, making unsupervised kernel learning significantly more efficient.

Assuming measurement x^t to be D-dimensional, feature function $\psi(x, y)$ is a $(2D + 4)$-dimensional vector organized as two D-dimensional parameter vectors, one per class, and four pairwise parameters, one per edge type. Hence, matrix h is $(2T + 4E) \times (2D + 4)$. Fig. 2 visualizes $\psi(x, y)$, h and y. In the figure, the blocks of '0' and '1' act as indicator functions that count the occurrences of edges of a given type. For instance, $\delta(y^a = P, y^b = P)$ will be equal to one if both nodes a and b are positive.

To map the M^{3}N dual (12) with this factorization, we need to express differences and products of feature vectors based on (18). To this aim, let us have two graphs with labelings y_i and y_j, of size $(2T_i + 4E_i)$ and $(2T_j + 4E_j)$, respectively. From (18), it follows that:

$$\psi(x_i, y_i)^\top \psi(x_j, y_j) = y_i^\top (h_i h_j^\top) y_j = y_i^\top \, H_{ij} \, y_j \tag{19}$$

Matrix H_{ij} is thus $(2T_i + 4E_i) \times (2T_j + 4E_j)$ (it is not square since the two graphs are generally not equal in size). Obviously, $H_{ij} = H_{ji}^\top$.

In the original M^{3}N formulation [27], the feature functions are decomposed as a sum of pairwise functions over the edges of nodes a and b:

$$\psi(x, y) = \sum_{a,b} \psi(x, y^a, y^b) \tag{20}$$

It is worth noting that such pairwise functions can also subsume unary terms by incorporating dummy nodes and edges as needed, and therefore they can be responsible for both pairwise and unary scores on the graph [27]. For this reason, we will ignore the unary terms in the following and decompose h over the edges of the graph. Notation h^{ab} refers to the portion of matrix h relative to nodes a and b, and y^{ab} to the binary variables for the edge. With this position, we have:

$$\psi(x, y) = \sum_{a,b} {h^{ab}}^\top y^{ab} \tag{21}$$

and, analogously:

$$\begin{aligned} \delta\psi_i(y) &= \psi_i(x_i, y_i) - \psi_i(x_i, y) \\ &= \sum_{a,b} \left[\psi_i(y_i^a, y_i^b) - \psi_i(y^a, y^b)\right] \\ &= \sum_{a,b} {h_i^{ab}}^\top \left[y_i^{ab} - y^{ab}\right] \end{aligned} \tag{22}$$

By means of these manipulations, the entire objective (14) can be expressed as a quadratic over y, as we show in the following subsection.

B. Finding a Violating Labeling

For ease of reference, we invert the sign of the M^{3}N objective (14) and decompose it into the following three terms:

$$-G(\lambda, y) = y^\top \mathcal{H} y + y^\top (\tau + \Delta) \tag{23}$$

where y is the concatenation of the labelings for all samples. The term in Δ maps the loss function, while the other two terms map, respectively, the quadratic and linear terms in the kernel product. Hereafter, we describe the values of H,τ and Δ, while their formal derivation is presented in Appendix B.

- By posing $H_{ij} = h_i h_j^\top$, $\mathcal{H}$ is formed as follows:

  $$\begin{aligned} \mathcal{H} = \tfrac{1}{2}[&H_{11}, H_{12} \dots H_{1N}; \\ &\dots; \\ &H_{N1}, H_{N2} \dots H_{NN}] \end{aligned}$$

  Please note that matrix $\mathcal{H}$ is positive semidefinite by construction. Its row dimension, D, is given by the sum of the labelings' length for the entire training set: $D = \sum_i 2T_i + 4E_i$.

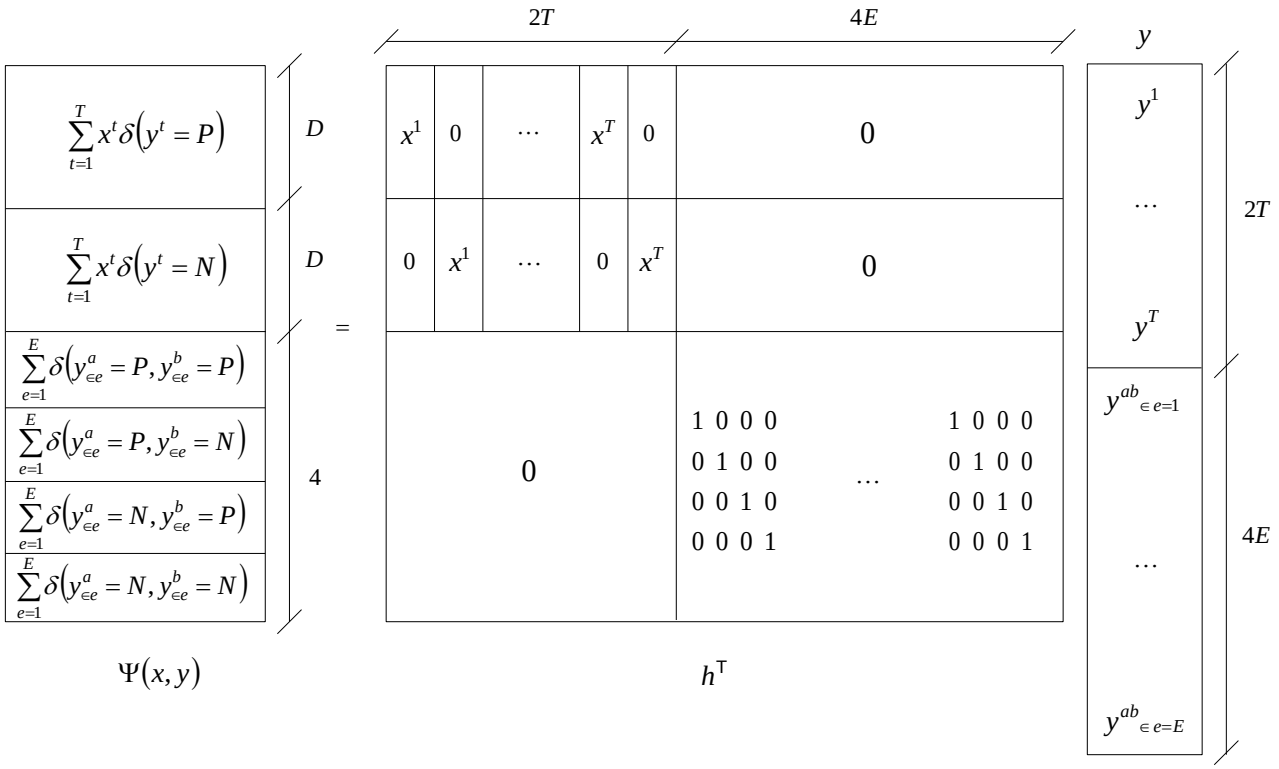

Fig. 2: Feature map h. The two binary variables for node t are noted jointly as y^t; the four binary variables for edge e are noted jointly with a double index as $y^{ab}_{\in e}$; and P and N denote positive and negative nodes, respectively.

- $\tau = \left[-h_1\left(\sum_j \sum_{c,d} h_j^{cd\top}\kappa_j^{cd}\right), -h_2^{\top}\left(\sum_j \sum_{c,d} h_j^{cd\top}\kappa_j^{cd}\right), \ldots, -h_N^{\top}\left(\sum_j \sum_{c,d} h_j^{cd\top}\kappa_j^{cd}\right)\right]$
  where $\kappa_j^{cd} = \sum_{v^c, v^d} \lambda_j(v^c, v^d) v_j^{cd}$
- $\Delta^{\top} = \left[\underbrace{\{\lambda_i(y^a = 0), \lambda_i(y^a = 1)\}}_{\lambda \text{ coefficients for the } a^{th} \text{ node}} \quad \underbrace{\mathbf{0} \in R^{4E}}_{\text{The loss has no pairwise terms}}\right]$
  where $\Delta^{\top}$ takes a value of $\lambda_i(y^a = 0)$ if the corresponding node in the ground truth is positive, and vice versa.

Having expressed $G(\lambda, y)$ as a quadratic form of y, we now seek to solve (4) which is the "crux" of the minimax relaxation. For the solution, we follow the approach of Li *et al*. [18]: let us assume that a working set C_w of labelings satisfying the constraints in (4) has already been determined. We now search to see if there exists another labeling $\tilde{y}$ for which function (23) takes a greater value than for every labeling in the working set. If such a labeling exists, it violates (4) and a re-computation of the optimum in μ and λ in (13) is required.

First, we find the argmax of the current working set, C_w:

$$\bar{y} = \underset{y \in C_w}{\operatorname{argmax}}\, y^{\top}\mathcal{H}y + y^{\top}(\tau + \Delta) \tag{24}$$

If C_w contains L ground-truth labelings, L evaluations of (23) over the training set are needed. Then, we pose $r = \mathcal{H}\bar{y} + \tau + \Delta$, and we search if a labeling $\tilde{y}$ exists such that:

$$\tilde{y}^{\top}\mathcal{H}\tilde{y} + \tilde{y}^{\top}(\tau + \Delta) > \bar{y}^{\top}r \tag{25}$$

Given the positive semidefiniteness of $\mathcal{H}$, the above search can be performed as a linear program:

$$\tilde{y} = \underset{y \in \beta}{\operatorname{argmax}}\, y^{\top}r \tag{26}$$

followed by a post-hoc verification of (25). If (25) holds, labeling $\tilde{y}$ is added to objective (13) and a re-computation triggered. Otherwise, training concludes.

To avoid adding labelings caused by numerical inaccuracies or over-fitting, we also impose a minimum threshold, γ, to validate a violating labeling. The violation criterion is therefore given as:

$$\tilde{y}^{\top}\mathcal{H}\tilde{y} + \tilde{y}^{\top}(\tau + \Delta) - \bar{y}^{\top}r > \gamma \tag{27}$$

The training algorithm terminates when the returned labeling, $\tilde{y}$, does not satisfy (27).

C. An Intuition of Well-M^{3}N

For the sake of providing an easier intuition of the proposed method, we depict a simplified, hypothetical scenario in Fig. 3. The goal is to detect the active phase of a signal which extends approximately between frames 45 and 120. The measurement could be an audio signal, an action feature from a video, or even a rise or fall in stock prices. By using a simple threshold-based detector such as k-means (the threshold is the continuous green line), the measurement is incorrectly decoded when it temporarily drops below the threshold. The output of the threshold-based detector is shown as the red dotted line. With Well-M^{3}N, this labeling only serves as the initial labeling (^1y) for the first round of M^{3}N training. The outcome is a first value for λ that is used to seek a violating labeling, ^2y (second inferred labeling in Fig. 3) that modifies some of the labels of ^1y. The ensuing step of MKL learning assigns two weights, μ_1 and μ_2, to these labelings, hopefully rewarding the labeling that better fits the data (^2y) and leading to a final, more correct classifier.

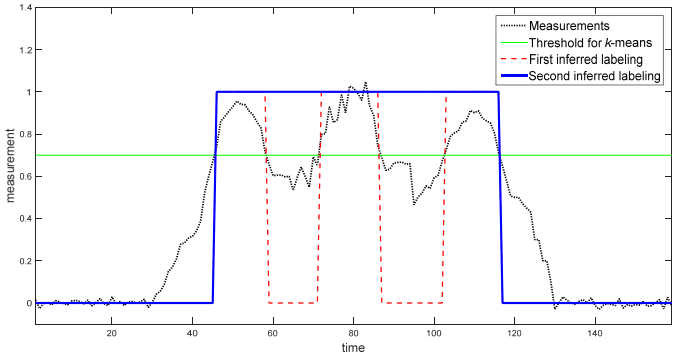

Fig. 3: An illustration of Well-M^{3}N.

IV. Experiments

In this section, we report experiments comparing the proposed Well-M^{3}N with three popular unsupervised algorithms: k-means, EM and unsupervised structural SVM (called unsupervised SSVM for short in the following) [25], [31]. This is a comprehensive comparison since k-means is a baseline algorithm that treats the data as unstructured, EM is a generative approach where latent variables are marginalized during training, and unsupervised SSVM is an example of local SVM algorithm. The comparison is carried out over two tasks of sequence segmentation, reporting the F_1 score of all predictors against the withheld ground-truth labels of the training set (Tables I and III) and of a separate test set (Tables II and III). For the second task, we also compare the proposed method with DCN (Deep Clustering Network), a very recent (2017) clustering approach based on deep learning [21]. As hyperparameters, we have used $C = 10^{-3}$ for both Well-M^{3}N and unsupervised SSVM, and $\epsilon = 0.1$ in the SSVM relaxation. For the EM model, we have used Gaussian emissions with full covariance matrices. For DCN, we have used a single hidden layer with 10 neurons[1].

A. Datasets Description

The experiments have been conducted over two time-segmentation datasets: 1) the Gesture Phase Segmentation dataset [32] from the UCI Machine Learning repository [33] and 2) a dataset of weather and climate data made available by the Australian Government Bureau of Meteorology (BoM). The Gesture Phase Segmentation dataset consists of Kinect skeletal information for six joints (hands, wrists, head and spine) of an actor narrating a story. As an example, Fig. 4 shows a visualization of the joints in a sub-sample of frames. Temporal annotation is provided in terms of gesture phases such as rest, preparation, stroke, hold and retraction. Given that we only assign binary labels, for our experiment we have equated 'rest' to the negative class and all the other active phases to the positive. Please note that these settings are probing as there will be many borderline measurements between the two classes. We have also scaled the measurements up by a factor of 100 to work in an approximately unitary range. In this dataset, a name such as 'Gesture-A1' refers to the video of actor 'A' narrating story '1'. The length of the videos varies from a minimum of $1,073$ to a maximum of $1,747$ frames. The BoM dataset consists of weather and climate time series made publicly available by the Australian Government Bureau of Meteorology. The time series we have used come from two locations near Hobart, Tasmania, and consist of daily observations of 16 features (rainfall, minimum and maximum temperatures, evaporation, hours of sunshine, speed of maximum wind gust; and temperature, relative humidity, cloud amount, wind speed, and MSL pressure at both 9am and 3pm) spanning from 1 July 2017 to 31 July 2018. For the experiments, we have used the rainfall feature as the binary response and the remaining features as predictors. Since the rainfall levels vary significantly, this task is expected to be probing.

B. Initialization

Initialization plays an important role in the performance of all tested algorithms. Despite its algebraic convexity, M^{3}N, too, appears to be sensitive to the starting ground-truth labeling. Therefore, we have decided to conduct the experiments by using k-means to initialize the three structured predictors. In turn, the initial centroids of k-means have been assigned randomly from the data for 10 iterations, and the resulting performance is reported in terms of mean and standard deviation. Overall, k-means has proved to be remarkably insensitive to its initialization, leading to very small variations in the results.

C. Experimental results

Table I reports the F_1 score (mean and standard deviation) over the Gesture Phase Segmentation dataset for the compared techniques from the initial labelings provided by k-means,

[1]All our code and data will be released publicly in the eventual acceptance of this paper.

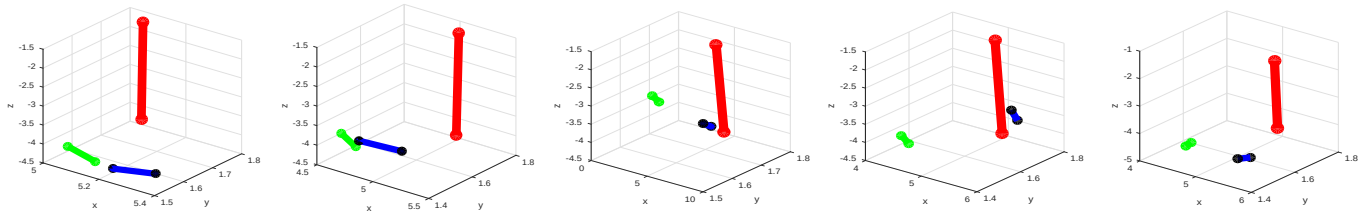

Fig. 4: Example frames from sequence A1 of the Gesture Phase Segmentation dataset (frames 44, 161, 197, 224, 322, in order). The first and last frames are examples of the rest phase while the intermediate frames are active phases. Each frame displays the position of the hands, wrists, head and spine of a narrating actor: head-spine: red bar; left hand-left wrist: blue bar; right hand-right wrist: green bar. This figure should be viewed in color.

showing the best results in bold face. Well-M^{3}N has achieved the best average F_1 score and the highest for 5 sequences out of 7, outperforming EM by 5 percentage points on average. In addition, it outperforms k-means and unsupervised SSVM by 6 and 9 percentage points, respectively. The most interesting result is the comparison with unsupervised SSVM: as we said previously, the main difference between these two techniques is that unsupervised SSVM replaces the predicted labeling at every learning iteration, whereas Well-M^{3}N accumulates them over iterations. This results in an averaging capability that makes the latter model more stable and effective.

To probe this comparison further, Fig. 5 shows a comparison of the labelings predicted by Well-M^{3}N and unsupervised SSVM for 96 frames of sequence Gesture-B3. This frame segment contains two events, one longer and one shorter, separated by a brief rest phase. In the figure, L_{gt} denotes the ground-truth labeling, L_1 is the initial labeling obtained by k-means, L_2 and L_3 are the intermediate labelings generated by Well-M^{3}N and unsupervised SSVM during learning, and L_p is their final prediction. The L_1 sequence shows that k-means produces an uncertain segmentation, with various bursts of false negatives. Using this sequence for initialization, Well-M^{3}N is able to produce an L_2 labeling that is much closer to the ground truth and an L_3 labeling that is all negative. By using all these labelings to train the final multi-kernel model, Well-M^{3}N is able to provide a final prediction (L_p) that correctly detects both events, with only a one frame glitch for the first and a slight delay for the second. Conversely, unsupervised SSVM only produces labelings that are very close to the initialization, with several bursts of false negatives that clearly affect the overall F_1 score for this sequence. This example illustrates better than anything the different behavior of the proposed Well-M^{3}N approach compared to an alternate-steps SVM approach.

In turn, Table II compares the generalization performance of Well-M^{3}N against k-means, EM and unsupervised structural SVM. To conduct this experiment, we have trained each model using only a single sequence, and tested it with the other six. Table II reports the F_1 score (mean $\pm$ standard deviation) over the six test sequences for every model and training sequence (for instance, the first row reports the average F_1 score over sequences A2-C3 for models trained with sequence A1). Well-M^{3}N clearly dominates the other techniques by a noticeable margin (at least 2 percentage points in the majority of cases and 3.5 on average). These results suggest that learning from multiple labelings can enjoy better generalization (or regularization) than learning from a single labeling such as with k-means or unsupervised structural SVM. The F_1 score is also higher than that of the generative probabilistic model that learns by marginalizing the labels (EM).

Eventually, Table III reports the F_1 scores (mean and standard deviation) of all the compared algorithms over the BoM dataset. Rows 1 and 2 show the accuracy over the undisclosed labels of the training sequences while rows 3 and 4 show the accuracy over the cross-validation sequences. On this dataset, the best performance has been achieved by the proposed Well-M^{3}N and the deep learning-based DCN. Well-M^{3}N has reported the highest F_1 score in two cases, DCN in one, and one tie. Given the deep architecture of DCN, it is possible that other combinations of numbers of layers and neurons could achieve even higher performance. However, the comparison is somehow biased because Well-M^{3}N is only assessed over the original feature space. The improvements over the best of the other techniques range between 0.2 percentage points (row 1) and 1.3 percentage points (row 2). In addition, the standard deviations are very low, suggesting that the differences should be statistically significant in all cases.

D. Comparison of the computational complexity of Well-M^{3}N and unsupervised SSVM

For an insight on the computational complexity of Well-M^{3}N, it can be useful to compare it with its closest method, unsupervised SSVM. A single run of both the baseline M^{3}N and SSVM solvers has quadratic complexity, $O(N^2)$, in the number of samples, N (see [27], [34] for details). Unsupervised SSVM iterates SSVM until the inferred latent variables stabilize: by referring to the number of iterations as I_{L}, its complexity can be expressed as $O(I_{\mathrm{L}}\, N^2)$. In our experiments, the value of I_{L} proved typically between 2 and 3. At its turn, Well-M^{3}N iterates M^{3}N over the number of violating labelings, L, which proved typically in the order of 2 to 4. In addition, at every iteration it solves a multi-kernel problem that is iterative at its turn. Referring to the number of MKL iterations as I_{MKL}, the overall complexity can then

TABLE I: Comparison of the F_1 score (mean $\pm$ standard deviation) for k-means, EM, unsupervised SSVM and Well-M^{3}N over the Gesture Phase Segmentation dataset.

DATASET	k-MEANS	EM	UNSUPERVISED SSVM	WELL-M^{3}N
GESTURE-A1 (32D)	0.60±0.00	**0.73±0.00**	0.59±0.01	0.66±0.00
GESTURE-A2 (32D)	0.58±0.00	**0.75±0.00**	0.59±0.00	0.66±0.00
GESTURE-A3 (32D)	0.72±0.03	0.68±0.00	0.72±0.04	**0.76±0.01**
GESTURE-B1 (32D)	0.88±0.01	0.86±0.01	0.88±0.00	**0.94±0.00**
GESTURE-B3 (32D)	0.88±0.01	0.75±0.02	0.81±0.00	**0.91±0.00**
GESTURE-C1 (32D)	0.74±0.02	**0.81±0.00**	0.63±0.00	**0.81±0.00**
GESTURE-C3 (32D)	0.78±0.03	0.70±0.02	0.78±0.00	**0.85±0.00**
Gesture average	0.74±0.00	0.75±0.00	0.71±0.00	**0.80±0.00**

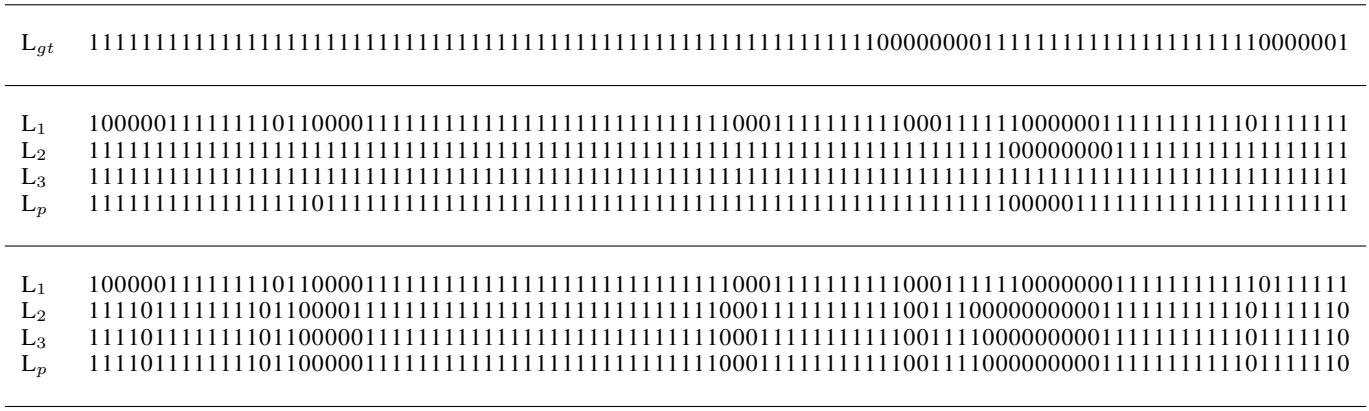

Fig. 5: Comparison of intermediate and final labelings from Well-M^{3}N (above) and unsupervised SSVM (below).

TABLE II: Comparison of the average cross-validation F_1 score (mean $\pm$ standard deviation) for k-means, EM, unsupervised SSVM and Well-M^{3}N over the Gesture Phase Segmentation dataset.

TRAINING SEQUENCE	k-MEANS	EM	UNSUPERVISED SSVM	WELL-M^{3}N
GESTURE-A1 (32D)	0.765±0.098	0.765±0.150	0.759±0.101	**0.861±0.080**
GESTURE-A2 (32D)	0.704±0.125	0.777±0.108	0.714±0.098	**0.793±0.096**
GESTURE-A3 (32D)	0.748±0.108	0.839±0.080	0.706±0.122	**0.840±0.087**
GESTURE-B1 (32D)	0.659±0.069	**0.775±0.083**	0.642±0.056	0.757±0.065
GESTURE-B3 (32D)	0.806±0.089	0.795±0.103	0.800±0.093	**0.827±0.083**
GESTURE-C1 (32D)	0.733±0.112	0.753±0.119	0.709±0.117	**0.807±0.091**
GESTURE-C3 (32D)	0.784±0.099	0.767±0.115	0.750±0.101	**0.833±0.089**
Gesture average	0.743±0.100	0.782±0.108	0.726±0.098	**0.817±0.084**

be expressed as $O(L\, I_{\text{MKL}}\, N^2)$. De facto, the training time of Well-M^{3}N has been approximately $5-6\times$ that of unsupervised SSVM in most cases. Figure 6 shows the rapid convergence of the Well-M^{3}N training algorithm as the number of labelings increases.

V. CONCLUSION

In this paper, we have presented a novel approach for performing unsupervised maximum-margin learning of structured predictors. The proposed approach, named Well-M^{3}N to credit its two main components (Well-SVM [18] and M^{3}N [27]), extends the minimax relaxation of [18] from the i.i.d. to the structured case. The extension is provided by leveraging maximum-margin Markov networks and reorganizing the structured feature vectors and loss function in a form that allows for finding violating labelings for the relaxation. Experimental results over two time-segmentation datasets have shown that the proposed approach has been able to achieve competitive F_1 scores, outperforming EM and unsupervised structural SVM in the majority of cases, both over the withheld labels and cross-validation test sets, and achieving comparable performance to a deep learning-based approach. In the light of this, we believe that the proposed technique can represent an interesting alternative to conventional unsupervised training approaches and find use for tasks as diverse as time-series segmentation, graph labeling, activity detection and other similarly structured tasks.

APPENDIX A
PROOF OF CONVERGENCE

In order to prove the convergence of Algorithm 1, we only need to prove that the training objective decreases or remains

TABLE III: Comparison of the F_1 score (mean $\pm$ standard deviation) for k-means, EM, unsupervised SSVM, Well-M^{3}N and DCN over the BoM dataset.

TRAINING	EVALUATION	k-MEANS	EM	UNSUPERVISED SSVM	WELL-M^{3}N	DCN
HOBART 1	HOBART 1	0.587 ± 0.001	0.594 ± 0.000	0.572 ± 0.009	**0.596 ± 0.001**	0.589 ± 0.000
HOBART 2	HOBART 2	0.560 ± 0.003	0.553 ± 0.010	0.546 ± 0.002	**0.573 ± 0.002**	**0.573 ± 0.000**
HOBART 1	HOBART 2	0.565 ± 0.002	0.551 ± 0.000	0.555 ± 0.001	0.569 ± 0.001	**0.573 ± 0.000**
HOBART 2	HOBART 1	0.580 ± 0.001	0.582 ± 0.004	0.583 ± 0.005	**0.590 ± 0.002**	0.585 ± 0.000
	Average	0.573 ± 0.003	0.570 ± 0.004	0.564 ± 0.004	**0.582 ± 0.001**	0.580 ± 0.000

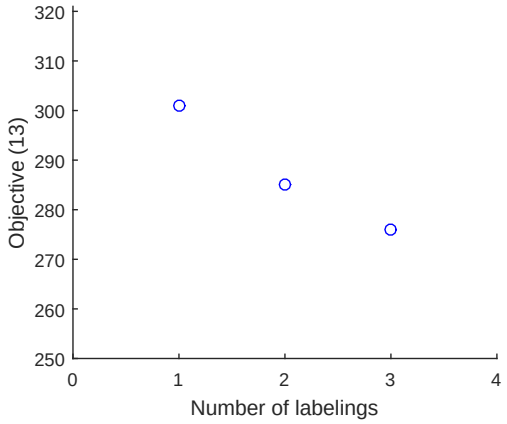

Fig. 6: Training objective of Well-M^{3}N (13) as a function of the number of labelings (BoM sequence Hobart 1).

equal every time a new ground-truth labeling is added, with some boundary conditions. To this aim, let us consider the objective as given in (7):

$$\min_{\mu} \max_{\lambda} \sum_{l=1}^{L} \mu^l G(\lambda, y^l) \tag{28}$$

Let us assume that the objective has been trained thus far using the current working set of L labelings. The resulting model is noted as $\bar{\lambda}_L$ and the weights as $\bar{\mu}^{1:L}$. We then find a new violating labeling, y^{L+1}, such that $G(\bar{\lambda}_L, y^{L+1}) < G(\bar{\lambda}_L, y^l), l = 1 \dots L$, and consider the following functional:

$$\max_{\lambda} \sum_{l=1}^{L+1} \mu^l G(\lambda, y^l) \tag{29}$$

computed at set point $\mu = \{\bar{\mu}^{1:L}, \mu^{L+1} = 0\}$. By construction, its value is identical to that of (28). By now granting the freedom to minimize over μ, we have that:

$$\min_{\mu} \max_{\lambda} \sum_{l=1}^{L+1} \mu^l G(\lambda, y^l) \leq \min_{\mu} \max_{\lambda} \sum_{l=1}^{L} \mu^l G(\lambda, y^l) \tag{30}$$

As for what the dependence on λ is concerned, $G(\lambda, y)$ is an SVM dual, a concave function over a convex domain. Its positive combination, $\sum_l \mu^l G(\lambda, y^l)$, with $\sum_l \mu^l = 1$, is therefore also concave over the same domain and its maximum is finite. In addition, the objective is bounded from below in y since the enumeration of labelings is (countably) finite. The fact that $G(\bar{\lambda}_L, y^{L+1}) < G(\bar{\lambda}_L, y^l), l = 1 \dots L$, guarantees that (30) is not trivially satisfied by the equality.

□

Let us also recall the constrained maximization in (4):

$$\begin{aligned} &\bar{\theta}_L(\lambda) = \max_{\theta} \ \theta \\ &s.t. \ \ \theta \leq G(\lambda, y^l) \quad l = 1 \dots L \end{aligned} \tag{31}$$

and prove that $\bar{\theta}_{L+1}(\lambda) \leq \bar{\theta}_L(\lambda)$. From the Lagrangian function, we have that:

$$\bar{\theta}_L(\lambda) = \min_{\mu} \sum_{l=1}^{L} \mu^l G(\lambda, y^l) \tag{32}$$

Let us note as $\bar{\mu}(\lambda)^{1:L}$ the argument of the minimum and consider the following function:

$$\sum_{l=1}^{L+1} \mu^l G(\lambda, y^l) \tag{33}$$

computed at set point $\mu = \{\bar{\mu}(\lambda)^{1:L}, \mu^{L+1} = 0\}$. Similarly to before, its value is identical to that of (32) by construction.

Therefore:

$$\bar{\theta}_{L+1}(\lambda) = \min_{\mu} \sum_{l=1}^{L+1} \mu^l G(\lambda, y^l) \ \leq \ \bar{\theta}_L(\lambda) \tag{34}$$

□

APPENDIX B
DERIVATION OF $\mathcal{H}$, τ AND Δ

In this Appendix, we derive the $\mathcal{H}$, τ and Δ terms used in Section (III-B). Let us start from the dual function of M^{3}N (12):

$$G(\lambda, y) = \underbrace{\sum_i \sum_{a,y^a} \lambda_i(y^a)\Delta t_i(y^a) -}_{\text{Linear in } \lambda} \underbrace{\frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d)\delta\psi_i(y^a, y^b)^\top \delta\psi_j(v^c, v^d)}_{\text{Quadratic in } \lambda} \tag{35}$$

First, let us consider the linear part of $G(\lambda, y)$:

$$\sum_i \sum_{a,y^a} \lambda_i(y^a)\Delta t_i(y^a) = \sum_i \sum_a \left[\lambda_i(y^a = 0)\Delta t_i(y^a = 0) + \lambda_i(y^a = 1)\Delta t_i(y^a = 1)\right] \tag{36}$$

The terms inconsistent with the ground-truth labeling contribute to $G(\lambda, y)$. Therefore, we can rewrite them as $y^\top \Delta$ where, for every graph i, Δ is formed as follows:

$$\Delta^\top = \left[\underbrace{\{\lambda_i(y^a = 0), \lambda_i(y^a = 1)\}}_{\text{for the } a^{th} \text{ node}} \quad \underbrace{\mathbf{0} \in R^{4E_i}}_{\text{All zeros for the edges}}\right]$$

For every graph i, the terms enclosed inside braces are concatenated, followed by a vector of $4E_i$ zeros. Such concatenation has to be repeated for all graphs, i.e., $i = 1 \dots N$.

Now, let us consider the quadratic part of $G(\lambda, y)$:

$$\begin{aligned}
&\frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d)\delta\psi_i(y^a, y^b)^\top \delta\psi_j(v^c, v^d) \\
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \\
&\left[\psi_i(x_i, y_i^a, y_i^b) - \psi_i(x_i, u_i^a, u_i^b)\right]^\top \left[\psi_j(x_j, y_j^c, y_j^d) - \psi_j(x_j, v_j^c, v_d^b)\right] \\
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \\
&\left[h_i^{ab\top} y_i^{ab} - h_i^{ab\top} u_i^{ab}\right]^\top \left[h_j^{cd\top} y_j^{cd} - h_j^{cd\top} v_j^{cd}\right] \\
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \left[\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right) - \right. \\
&\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} v_j^{cd}\right) - \left(h_i^{ab\top} u_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right) + \\
&\left.\left(h_i^{ab\top} u_i^{ab}\right)^\top \left(h_j^{cd\top} v_j^{cd}\right)\right]
\end{aligned} \tag{37}$$

Neglecting the last term since it is independent of y during the maximization of $G(\lambda, y)$ leads to:

$$\begin{aligned}
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \left[\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right) - \right. \\
&\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} v_j^{cd}\right) - \left(h_i^{ab\top} u_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right)\Big] \\
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \left[\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right) - \right. \\
&\left.2\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} v_j^{cd}\right)\right] \\
&= \underbrace{\frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \left[\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right)\right]}_{1^{st} term} \\
&\underbrace{-\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} v_j^{cd}\right)}_{2^{nd} term}
\end{aligned} \tag{38}$$

For the definition of $\mathcal{H}$, let us consider the first term of (38):

$$\begin{aligned}
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \lambda_i(y^a, y^b)\lambda_j(v^c, v^d) \\
&\left[\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right)\right] \\
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \\
&\left[\lambda_i(y^a, y^b)\left(h_i^{ab\top} y_i^{ab}\right)^\top \left(h_j^{cd\top} y_j^{cd}\right)\lambda_j(v^c, v^d)\right] \\
&= \frac{1}{2}\sum_{i,j} \sum_{\substack{a,b \\ y^a,y^b}} \sum_{\substack{c,d \\ v^c,v^d}} \left[\lambda_i(y^a, y^b) y_i^{ab\top} h_i^{ab} h_j^{cd\top} y_j^{cd} \lambda_j(v^c, v^d)\right] \\
&= \frac{1}{2}\sum_{i,j} \left[\left(\sum_{\substack{a,b \\ y^a,y^b}} \lambda_i(y^a, y^b) y_i^{ab\top} h_i^{ab}\right) \left(\sum_{\substack{c,d \\ v^c,v^d}} h_j^{cd\top} y_j^{cd} \lambda_j(v^c, v^d)\right)\right]
\end{aligned}$$

Given that $\sum_{y^a,y^b} \lambda(y^a, y^b) = C$, we obtain:

$$= \frac{1}{2}\sum_{i,j} \left[\left(\sum_{a,b} C y_i^{ab\top} h_i^{ab}\right) \left(\sum_{c,d} C h_j^{cd\top} y_j^{cd}\right)\right] \tag{39}$$

If we design h_i^{ab} as follows, where $h_i^{ab} \in R^{6\times(2D+4)}$ (Section (III-A)):

$$h_i^{ab} = \begin{bmatrix} Cx_i^b & 0^D & 0 & 0 & 0 & 0 \\ 0^D & Cx_i^b & 0 & 0 & 0 & 0 \\ 0^D & 0^D & C & 0 & 0 & 0 \\ 0^D & 0^D & 0 & C & 0 & 0 \\ 0^D & 0^D & 0 & 0 & C & 0 \\ 0^D & 0^D & 0 & 0 & 0 & C \end{bmatrix}$$

then we can easily rewrite the first term of (39) as:

$$\frac{1}{2}\sum_{i,j}\sum_{\substack{a,b \\ y^a,y^b}}\sum_{\substack{c,d \\ v^c,v^d}}\left[{y_i^{ab}}^\top h_i^{ab} {h_j^{cd}}^\top y_j^{cd}\right]$$

(C has been absorbed in h_i^{ab})

$$= \frac{1}{2}\sum_{i,j}\left[y_i^\top \underbrace{h_i h_j^\top}_{H_{ij}} y_j\right]$$

After the concatenation of label vector $y = [y_1, y_2, \ldots, y_N]$ and the formation of $\mathcal{H}$ as the matrix of blocks H_{ij}, we can write the above summation as:

$$= \frac{1}{2} y^\top \mathcal{H} y \text{ (comment: } 1/2 \text{ can be easily absorbed inside } \mathcal{H})$$
$$\rightarrow y^\top \mathcal{H} y$$

Similarly, for the definition of τ, let us consider the second term of (38) which is linear in y:

$$-\sum_{i,j}\sum_{\substack{a,b \\ y^a,y^b}}\sum_{\substack{c,d \\ v^c,v^d}}\lambda_i(y^a,y^b)\lambda_j(v^c,v^d)\left({h_i^{ab}}^\top y_i^{ab}\right)^\top\left({h_j^{cd}}^\top v_j^{cd}\right)$$

$$= -\sum_{i,j}\left(\sum_{\substack{a,b \\ y^a,y^b}}\lambda_i(y^a,y^b){h_i^{ab}}^\top y_i^{ab}\right)^\top\left(\sum_{\substack{c,d \\ v^c,v^d}}\lambda_j(v^c,v^d){h_j^{cd}}^\top v_j^{cd}\right)$$

After formulating h_i^{ab} as mentioned above, we have:

$$= -\sum_{i,j}\left(\sum_{a,b}{h_i^{ab}}^\top y_i^{ab}\right)^\top\left(\sum_{c,d}{h_j^{cd}}^\top \sum_{v^c,v^d}\lambda_j(v^c,v^d)v_j^{cd}\right)$$

Let $\kappa_j^{cd} = \sum_{v^c,v^d}\lambda_j(v^c,v^d)v_j^{cd}$

$$= -\sum_{i,j}\left(\sum_{a,b}{h_i^{ab}}^\top y_i^{ab}\right)^\top\left(\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)$$

$$= -\left(\sum_{i}\sum_{a,b}{h_i^{ab}}^\top y_i^{ab}\right)^\top\left(\sum_{j}\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)$$

$$= -\left(\sum_i {h_i}^\top y_i\right)^\top\left(\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)$$

$$= -\left(\sum_i y_i^\top h_i\right)\left(\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)$$

$$= -\sum_i y_i^\top\left(h_i\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)$$

After the concatenation of label vector $y = [y_1, y_2, \ldots, y_N]$ and the formation of τ as follows:

$$\tau = \left[-h_1\left(\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right), -h_2^\top\left(\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right), \ldots, -h_N^\top\left(\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)\right]$$

$$= -\sum_i y_i^\top\left(h_i\sum_j\sum_{c,d}{h_j^{cd}}^\top \kappa_j^{cd}\right)$$

$$= y^\top \tau$$

we have completed the rewriting of the quadratic part of (35) as $y^\top(\mathcal{H}y + \tau)$. Given that the linear part of (35) had already been rewritten as $y^\top\Delta$, we can finally rewrite the entire (35) as the following quadratic function of ground-truth labeling y:

$$-G(\lambda, y) = y^\top(\mathcal{H}y + \tau + \Delta) \qquad \square$$

REFERENCES

[1] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, 1967, pp. 281–297.

[2] G. J. McLachlan and D. Peel, *Finite mixture models*. Wiley Series in Probability and Statistics, 2000.

[3] V. N. Vapnik, *Statistical learning theory*. Wiley, 1998.

[4] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Efficient structured prediction with latent variables for general graphical models," in *ICML*, 2012.

[5] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "A support vector method for clustering," in *NIPS*, 2000, pp. 367–373.

[6] ——, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[7] K. Zhang, I. W. Tsang, and J. T. Kwok, "Maximum margin clustering made practical," in *ICML*. ACM, 2007, pp. 1119–1126.

[8] ——, "Maximum margin clustering made practical," *IEEE Trans. Neural Networks*, vol. 20, no. 4, pp. 583–596, 2009.

[9] X. L. Zhang and J. Wu, "Linearithmic time sparse and convex maximum margin clustering," *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 6, pp. 1669–1692, 2012.

[10] Z. S. Karnin, E. Liberty, S. Lovett, R. Schwartz, and O. Weinstein, "Unsupervised SVMs: On the complexity of the furthest hyperplane problem," in *COLT 2012 - The 25th Annual Conference on Learning Theory*, 2012, pp. 2.1–2.17.

[11] J. Peng, L. Mukherjee, V. Singh, D. Schuurmans, and L. Xu, "An efficient algorithm for maximal margin clustering," *J. Global Optimization*, vol. 52, no. 1, pp. 123–137, 2012.

[12] F. Gieseke, O. Kramer, A. Airola, and T. Pahikkala, "Efficient recurrent local search strategies for semi- and unsupervised regularized least-squares classification," *Evolutionary Intelligence*, vol. 5, no. 3, pp. 189–205, 2012.

[13] X. L. Zhang, "Heuristic ternary error-correcting output codes via weight optimization and layered clustering-based approach," *IEEE Trans. on Cybernetics*, vol. 45, no. 2, pp. 289–301, 2015.

[14] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, "Maximum margin clustering," in *NIPS*, 2005, pp. 1537–1544.

[15] L. Xu and D. Schuurmans, "Unsupervised and semi-supervised multi-class support vector machines," in *AAAI*, 2005, pp. 904–910.

[16] L. Xu, D. Wilkinson, and D. Schuurmans, "Discriminative unsupervised learning of structured predictors," in *ICML*, 2006, pp. 1057–1064.

[17] Y.-F. Li, I. W. Tsang, J. T. Kwok, and Z.-H. Zhou, "Tighter and convex maximum margin clustering," in *AISTATS 2009*, 2009, pp. 344–351.

[18] ——, "Convex and scalable weakly labeled SVMs," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2151–2188, 2013.

[19] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *arXiv*, no. 1801.07648, pp. 1–12, 2018.

[20] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, 2016, pp. 478–487.

[21] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *ICML*, 2017, pp. 3861–3870.

[22] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *CVPR*, 2016, pp. 5147–5156.

[23] Y. Liu, J. Chen, and L. Deng, "Unsupervised sequence classification using sequential output statistics," in *NIPS*, 2017, pp. 3533–3562.

[24] W. Ammar, C. Dyer, and N. A. Smith, "Conditional random field autoencoders for unsupervised structured prediction," in *NIPS*, 20147, pp. 3311–3319.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[26] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," in *Journal of Machine Learning Research*, 2005, pp. 1453–1484.

[27] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *NIPS 2003*, 2004, pp. 25–32.

[28] A. Rakotomamonjy, U. D. Rouen, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[29] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," 2014. [Online]. Available: http://cvxr.com/cvx

[30] ——, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, 2008, pp. 95–110.

[31] C.-N. J. Yu and T. Joachims, "Learning structural SVMs with latent variables," in *ICML*. ACM, 2009, pp. 1169–1176.

[32] R. C. Madeo, C. A. Lima, and S. M. Peres, "Gesture unit segmentation using support vector machines: segmenting gestures from rest positions," in *28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 46–52.

[33] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[34] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Mach. Learn.*, vol. 77, no. 1, pp. 27–59, Oct. 2009.