Learning Maximum Margin Markov Networks from examples with missing labels

Vojtech Franc xfrancv@fel.cvut.cz
Andrii Yermakov yermaand@fel.cvut.cz
Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Structured output classifiers based on the framework of Markov Networks provide a transparent way to model statistical dependencies between output labels. The Markov Network (MN) classifier can be efficiently learned by the maximum margin method, which however requires expensive completely annotated examples. We extend the maximum margin algorithm for learning of unrestricted MN classifiers from examples with partially missing annotation of labels. The proposed algorithm translates learning into minimization of a novel loss function which is convex, has a clear connection with the supervised margin-rescaling loss, and can be efficiently optimized by first-order methods. We demonstrate the efficacy of the proposed algorithm on a challenging structured output classification problem where it beats deep neural network models trained from a much higher number of completely annotated examples, while the proposed method used only partial annotations.
Keywords: Markov networks, missing labels, structured output classification, learning

1. Introduction

The structured output (SO) classification deals with the prediction of a set of statistically interdependent labels. A transparent way to model the dependencies provides the framework of Markov Networks. In this paper, we tackle the learning of a SO classifier, further referred to as the Markov Network (MN) classifier, capturing the dependencies between pairs of labels. Formally, the MN classifier is defined as follows. Let $(\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ is a finite set of objects and $\mathcal{E} \subseteq \binom{\mathcal{V}}{2}$ is a set of interacting object pairs. Let $\boldsymbol{x} \in \mathcal{X}$ be an observation and $\boldsymbol{y} = (y_v \in \mathcal{Y} \mid v \in \mathcal{V})$ a tuple of labels assigned to objects. The labels are from a finite set $\mathcal{Y}$. For example, in the case of image segmentation $\boldsymbol{x}$ is an image, $\mathcal{V}$ are pixels, $\boldsymbol{y}$ contains the pixel labels, and $\mathcal{E}$ defines the pixel neighborhood. Match between the observation $\boldsymbol{x}$ and a label $y_v \in \mathcal{Y}$ assigned to object $v \in \mathcal{V}$ is scored by a function $f_v \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Match between labels $(y_v, y_{v'})$ assigned to interacting objects $\{v, v'\} \in \mathcal{E}$ is scored by a function $f_{vv'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Given an observation $\boldsymbol{x} \in \mathcal{X}$, the MN classifier $\boldsymbol{h} \colon \mathcal{X} \to \mathcal{Y}^{\mathcal{V}}$ outputs labelling $\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}$ with the maximal total score

$$\boldsymbol{h}(\boldsymbol{x}) \in \operatorname*{Argmax}_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}} \Big[\sum_{v \in \mathcal{V}} f_v(\boldsymbol{x}, y_v) + \sum_{\{v,v'\} \in \mathcal{E}} f_{vv'}(y_v, y_{v'}) \Big] . \tag{1}$$

Evaluation of the MN classifier (1) requires solving a discrete optimization task, so-called max-sum problem. In general, the max-sum problem is an NP-hard but there are sub-

classes which can be solved efficiently. For example, when the graph $(\mathcal{V}, \mathcal{E})$ is a tree, the max-sum problem can be solved by dynamic programming, when the pair-wise functions $f_{vv'}$, $\{v, v\} \in \mathcal{E}$, are supermodular, it can be solved by max-flow algorithm, or one can resort to approximate solution by linear programming (LP) relaxation (Werner, 2007). The choice of the graph $(\mathcal{V}, \mathcal{E})$ and the class of score functions $(f_v, f_{vv'})$ offers a powerful way to encode the prior knowledge about the prediction problem. The particular score functions from the chosen class are then learned from examples.

The linearly parametrized score functions can be learned by a family of methods further denoted as Maximum Margin Markov Network (M3N) algorithms (Taskar et al., 2004a,b; Tsochantaridis et al., 2005; Franc and Savchynskyy, 2008). The M3N algorithms translate learning into a convex optimization problem. The M3N algorithms require a training set of completely annotated examples $\mathcal{D}_{xy} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i) \in \mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \mid i = 1, \ldots, m\}$. The requirement of the complete annotation constitutes a significant limitation because gathering the annotation manually is expensive.

This paper tackles the learning of MN classifiers from partially annotated examples $\mathcal{D}_{xa} = \{(\boldsymbol{x}^i, \boldsymbol{a}^i) \in \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \mid i = 1, \ldots, m\}$ where $\mathcal{A} = \{\mathcal{Y} \cup \{?\}\}$ denotes a set of admissible annotations. Given the annotation $\boldsymbol{a}^i = (a^i_v \in \mathcal{A} \mid v \in \mathcal{V})$ of the i-th training input $\boldsymbol{x}^i \in \mathcal{X}$, the label of the object $v \in \mathcal{V}$ is either known, $a^i_v \in \mathcal{Y}$, or missing, $a^i_v =?$, meaning that all labels are possible. It has been shown by Antoniuk and Franc (2015) that a statistically consistent algorithm, learning SO classifiers from partially annotated examples $\mathcal{D}_{xa}$, can be based on the minimization of a partial loss which counts errors only on the annotated objects. The minimization of the partial loss is at the core of many methods learning SO classifiers from the partial annotations (Chuong et al., 2008; Fernandes and Brefeld, 2011; Lou and Hamprecht, 2012; Yu et al., 2014). The existing methods replace the discrete partial loss by a proxy, e.g., by the ramp loss and its modifications, which is more amenable to optimization. To our knowledge, all existing proxies are however nonconvex and the existing solvers are local optimization methods prone to get stuck in local optima. Besides this, the minimization of nonconvex proxies is computationally demanding and not tractable for MN classifier unless its structure is restricted, e.g., to tree graphs $(\mathcal{V}, \mathcal{E})$. We resolve this problem by proposing a novel proxy of the partial loss which is i) convex, ii) theoretically grounded, iii) tractable for unrestricted MN classifier, and iv) it can be efficiently optimized by simple first-order methods like stochastic gradient algorithm.

The deep neural architectures have been successfully applied to a large number of prediction problems including SO classification. However, deep learning has its limits: it is data hungry and it produces hard to interpret black-box models. Besides, canonical deep architectures fail in some problems, e.g., learning simple arithmetic operations (Trask et al., 2018), learning functions on binary chains like even/odd parity (Shalev-Shwartz et al., 2017), or, most related to this paper, in SO classification with complex interactions between labels. An example of such challenging problem is prediction of the solution of the Sudoku puzzle (Wang et al., 2019). The weaknesses and strengths of deep learning and interpretable models, like the Markov Markov networks, are complementary. Therefore, many current works integrate the advantages of deep networks and interpretable models like Markov Networks (Chen et al., 2015; Schwing and Urtasun, 2015; Zheng et al., 2015), MAXSAT framework (Wang et al., 2019), relational reasoning models (Palm et al., 2018), or first-order logic rules (Yang et al., 2017). To our knowledge, all existing methods, how-

ever, require a training set of completely annotated examples. The method proposed in this paper allows the learning of complex structured models from cheaper partial annotations and it can be seamlessly combined with neural networks. Contributions of this paper:

1. We propose a convex loss for learning the MN classifiers (1) from partially annotated examples $\mathcal{D}_{xa}$. The loss can be efficiently optimized by simple first-order methods. The loss minimization is tractable even in case of an unrestricted MN classifiers with an arbitrary neighbourhood $(\mathcal{V}, \mathcal{E})$ and unrestricted class of score functions $(f_v, f_{vv'})$.

2. We provide a theoretical justification of the proposed loss function. We prove that asymptoticaly the value of proposed loss equals to the value of standard supervised margin-rescaling loss.

3. We demonstrate experimentally the efficacy of the proposed method on challenging problem of learning the Sudoku solver. Our approach significantly outperforms the deep neural networks on both Sudoku with symbolic and visual input.

2. Existing methods

In section 2.1, we outline the supervised M3N algorithm. In section 2.2, we describe a known formulation of learning from partially annotated examples and describe the issues of existing nonconvex loss functions.

2.1. Supervised Maximum Margin Markov Networks

Let us assume that the score functions defining the MN classifier (1) are linear in parameters, i.e. $f_v(\boldsymbol{x}, y) = \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, y)\rangle$, $v \in \mathcal{V}$, and $f_{vv'}(y_v, y_{v'}) = \langle \boldsymbol{w}, \boldsymbol{\psi}(y_v, y_{v'})\rangle$, $\{v, v'\} \in \mathcal{E}$, where $\boldsymbol{\psi}_v \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$, $v \in \mathcal{V}$, and $\boldsymbol{\psi}_{vv'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^d$, $\{v, v'\} \in \mathcal{E}$, are fixed feature maps and $\boldsymbol{w} \in \mathbb{R}^d$ is a parameter vector to learn. The MN classifier is then an instance of a linear classifier $\boldsymbol{h} \colon \mathcal{X} \times \mathbb{R}^d \to \mathcal{Y}^{\mathcal{V}}$ defined as

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w}) \in \operatorname*{Argmax}_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}} \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})\rangle \tag{2}$$

where $\boldsymbol{\psi} \colon \mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \to \mathbb{R}^d$ is the joint input-output feature map defined as

$$\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{v \in \mathcal{V}} \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) + \sum_{\{v,v'\} \in \mathcal{E}} \boldsymbol{\psi}_{vv'}(y_v, y_{v'})\,. \tag{3}$$

Let $p(\boldsymbol{x}, \boldsymbol{y})$ be a distribution over $\mathcal{X} \times \mathcal{Y}^{\mathcal{V}}$, let $\ell \colon \mathcal{Y}^{\mathcal{V}} \times \mathcal{Y}^{\mathcal{V}} \to \mathbb{R}_+$ be a target loss and $R^\ell(\boldsymbol{h}, p) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y} \sim p} \ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w}))$ the ℓ-risk measuring the performance of $\boldsymbol{h}$ on $p(\boldsymbol{x}, \boldsymbol{y})$. Let $\mathcal{D}_{xy} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i) \in (\mathcal{X} \times \mathcal{Y}^{\mathcal{V}}) \mid i = 1, \ldots, m\}$ be a training set of completely annotated examples drawn from i.i.d. random variables distributed according to $p(\boldsymbol{x}, \boldsymbol{y})$. The goal of supervised learning is to use $\mathcal{D}_{xy}$ to infer $\boldsymbol{w}$ such that the ℓ-risk $R^\ell(\boldsymbol{h}, p)$ is close to the Bayes ℓ-risk $R^\ell_*(p) = \inf_{\boldsymbol{h} \colon \mathcal{X} \to \mathcal{Y}^{\mathcal{V}}} R^\ell(h, p)$.

The M3N-based algorithm casts learning of $\boldsymbol{w}$ from $\mathcal{D}_{xy}$ as a convex problem

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^n} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \Delta(\boldsymbol{x}^i, \boldsymbol{y}^i, \boldsymbol{w})\right] \tag{4}$$

where $\Delta\colon \mathcal{Y}^{\mathcal{V}} \times \mathcal{Y}^{\mathcal{V}} \times \mathbb{R}^d \to \mathbb{R}_+$ is the margin-rescaling loss defined as

$$\Delta(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \max_{\boldsymbol{y}' \in \mathcal{Y}^{\mathcal{V}}} \left[\ell(\boldsymbol{y}, \boldsymbol{y}') + \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}')\rangle\right] - \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})\rangle \,, \tag{5}$$

where $\lambda \geq 0$ is a regularization constant. In the case of the additive loss (8), the loss-augmented classification (LAC) problem in (5), i.e. the maximization task, becomes an instance of the max-sum problem and it can be efficiently solved by dynamic programming when $(\mathcal{V}, \mathcal{E})$ is a tree, otherwise, the LAC can be replaced by its linear programming upper bound (Taskar et al., 2004a,b; Tsochantaridis et al., 2005; Franc and Savchynskyy, 2008). In both cases, the loss is convex in the parameters and it can be evaluated together with its subgradient in time polynomial in size of $(\mathcal{V}, \mathcal{E})$ and $\mathcal{Y}$.

2.2. Learning from partially annotated examples

Let $\mathcal{D}_{xa} = \{(\boldsymbol{x}^i, \boldsymbol{a}^i) \in \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \mid i = 1, \ldots, m\}$ be a set of partially annotated training examples drawn from i.i.d. random variables with the distribution

$$p''(\boldsymbol{x}, \boldsymbol{a}) = \sum_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}} p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) \tag{6}$$

where $\mathcal{A} = \{\mathcal{Y} \cup \{?\}\}$ denotes a set of admissible annotations of an object $v \in \mathcal{V}$ and $p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a})$ is a distribution over $\mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \times \mathcal{A}^{\mathcal{V}}$. The distribution $p'(\boldsymbol{x}, \boldsymbol{y})$ defined over $\mathcal{X} \times \mathcal{Y}^{\mathcal{V}}$ is obtained from $p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a})$ by marginalization w.r.t annotations $\mathcal{A}^{\mathcal{V}}$, i.e.,

$$p'(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{a} \in \mathcal{A}^{\mathcal{V}}} p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) \,. \tag{7}$$

The goal of learning is to use $\mathcal{D}_{xa}$ to infer $\boldsymbol{w}$ which has the ℓ-risk $R^{\ell}(\boldsymbol{h}, p')$ close to the Bayes ℓ-risk $R^{\ell}_*(p')$. Note that the goals of supervised learning and learning from partially annotated examples are the same, however, the training sets are different.

A statistically consistent approach to minimize the ℓ-risk by minimizing a partial loss $\ell^p\colon \mathcal{A}^{\mathcal{V}} \times \mathcal{Y}^{\mathcal{V}} \to \mathbb{R}_+$ on $\mathcal{D}_{xa}$ was described in Antoniuk and Franc (2015). The approach requires two assumptions. First, the target loss $\ell\colon \mathcal{Y}^{\mathcal{V}} \times \mathcal{Y}^{\mathcal{V}} \to \mathbb{R}$ is additive, i.e.,

$$\ell(\boldsymbol{y}, \boldsymbol{y}') = \sum_{v \in \mathcal{V}} \ell_v(y_v, y'_v) \,, \tag{8}$$

where $\ell_v\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, $v \in \mathcal{V}$, are arbitrary loss functions such that $\ell_v(y, y') = 0$ iff $y = y'$. An example of the additive loss is the Hamming loss $\ell(\boldsymbol{y}, \boldsymbol{y}') = \sum_{v \in \mathcal{V}} [\![y_v \neq y'_v]\!]$. Second, the labels are missing at random (MAR), meaning that the annotation process $p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y})$ defining the distribution $p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) = p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{x}, \boldsymbol{y})$ satisfies

$$p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{z} \in \{0,1\}^{\mathcal{V}}} p(\boldsymbol{z} \mid \boldsymbol{x}) \prod_{v \in \mathcal{V}} [\![a_v = \alpha(y_v, z_v)]\!] \,, \tag{9}$$

where $\alpha(y_v, z_v) = y_v$ if $z_v = 1$, $\alpha(y_v, z_v) =?$ if $z_v = 0$, and $p(\boldsymbol{z} \mid \boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$, are conditional distributions over $\boldsymbol{z} \in \{0, 1\}^{\mathcal{V}}$ such that marginals $p(z_v = 1 \mid \boldsymbol{x}) > 0$, $\forall v \in \mathcal{V}$[1]. The MAR

1. We distinguish the distributions by their arguments, i.e., $p(z_v \mid \boldsymbol{x})$ and $p(z_{v'} \mid \boldsymbol{x})$ are different marginal distributions provided $v \neq v'$.

process means that the examples $\mathcal{D}_{xa}$ are generated as follows. The nature generates $(\boldsymbol{x}, \boldsymbol{y})$ from $p(\boldsymbol{x}, \boldsymbol{y})$. The annotator decides objects to label based on observing the input $\boldsymbol{x}$. His decision is stochastic, represented by a binary vector $\boldsymbol{z} \in \{0,1\}^{\mathcal{V}}$ generated from $p(\boldsymbol{z} \mid \boldsymbol{x})$. The annotator reveals the labels of the objects $\mathcal{V}_{lab} = \{v \in \mathcal{V} \mid z_v = 1\}$, i.e. he sets $a_v = y_v$, $v \in \mathcal{V}_{lab}$, while labels of the remaining objects are not provided, i.e. $a_v = ?$, $v \in \mathcal{V} \setminus \mathcal{V}_{lab}$.

Under the two assumptions, the original task of minimizing the ℓ-risk can be replaced by an equivalent auxiliary task of minimizing ℓ^p-risk defined as follows. Given a target additive loss ℓ, an associated partial loss $\ell^p \colon \mathcal{A}^{\mathcal{V}} \times \mathcal{Y}^{\mathcal{V}} \to \mathbb{R}_+$ counts penalty only on the annotated objects, i.e.,

$$\ell^p(\boldsymbol{a}, \boldsymbol{y}) = \sum_{v \in \mathcal{V}} [\![a_y \neq ?]\!] \ell_v(a_v, y_v) \,, \tag{10}$$

The auxiliary task is then to find $\boldsymbol{h}$ with the ℓ^p-risk $R^{\ell^p}(\boldsymbol{h}, p'') = \mathbb{E}_{\boldsymbol{x},\boldsymbol{a} \sim p''} \ell^p(\boldsymbol{a}, \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w}))$ close to the Bayes ℓ^p-risk $R^{\ell^p}_*(p'') = \inf_{\boldsymbol{h} \colon \mathcal{X} \to \mathcal{Y}^{\mathcal{V}}} R^{\ell^p}(\boldsymbol{h}, p'')$. Equivalence between the original task, i.e., minimization of ℓ-risk, and the auxiliary task, i.e., minimization of ℓ^p-risk, was shown in Antoniuk and Franc (2015). Namely, for all sequences of classifiers $\boldsymbol{h}_m \colon \mathcal{X} \to \mathcal{Y}^{\mathcal{V}}$ it holds that $R^{\ell^p}(\boldsymbol{h}_m, p'') \xrightarrow{P} R^{\ell^p}_*(p'')$ implies $R^{\ell}(\boldsymbol{h}_m, p') \xrightarrow{P} R^{\ell}_*(p')$. Minimization of ℓ^p-loss was implemented in many algorithms learning from partially annotated examples $\mathcal{T}_{xa}$ like, e.g., Chuong et al. (2008); Sarawagi and Gupta (2008); Vedaldi and Zisserman (2009); Yu and Joachims (2009); Wang and Mori (2010); Girshick et al. (2011); Fernandes and Brefeld (2011); Lou and Hamprecht (2012); Yu et al. (2014). A direct minimization of ℓ^p-loss is however difficult. Therefore, the existing approaches minimize a proxy of the form

$$\Delta^p_{\text{NON}-\text{CVX}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{U}^P} \left[\ell^p(\boldsymbol{a}, \boldsymbol{y}) + \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle \right] - \max_{\boldsymbol{y} \in \mathcal{U}^R} \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle \,, \tag{11}$$

where the label set $\mathcal{U}^P \subseteq \mathcal{Y}^{\mathcal{V}}$ is so called penalty space and $\mathcal{U}^R \subseteq \mathcal{Y}^{\mathcal{V}}$ is a reward space. For example, for $\mathcal{U}^R = \mathcal{U}^P = \mathcal{Y}^{\mathcal{V}}$ the loss $\Delta^p_{\text{NON}-\text{CVX}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w})$ becomes the Ramp-loss most frequently used in literature. Application of this approach to learning the parameters of the MN classifier (2) leads to a nonconvex problem

$$\boldsymbol{w}^* \in \operatorname*{Argmin}_{\boldsymbol{w} \in \mathbb{R}^n} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \Delta^p_{\text{NON}-\text{CVX}}(\boldsymbol{x}^i, \boldsymbol{a}^i, \boldsymbol{w}) \right] . \tag{12}$$

A standard method to solve (12) is the convex-concave procedure (CCCP) (Yuille and Rangarajan, 2003) which, however, has three serious issues. First, the CCCP finds a local optimum, the quality of which depends on the initialization and is strongly variable. Second, the CCCP algorithm is computationally demanding as it alternates the prediction of all training examples and supervised learning until convergence. Third, the concave term in (11) cannot be approximated by LP relaxation, in turn, the approach is not tractable for unrestricted MN classifiers, and, for this reason, it has been used so far only for learning tree based MN classifiers (Lou and Hamprecht, 2012).

3. Proposed method

In section 3.1, we propose the partial margin-rescaling loss as a novel convex proxy of ℓ^p-loss. The partial margin-rescaling loss requires knowledge of the marginals of the missingness

process $p(\boldsymbol{z} \mid \boldsymbol{x})$, the estimation of which is discussed in section 3.2. Finally, in section 3.3, we propose a linear programming relaxation of the partial margin-rescaling loss whose minimization is tractable even for unrestricted MN classifiers.

3.1. Partial margin-rescaling loss

To sum up, the missing label scenario described in the previous section considers data $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a})$ generated from $p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a})$, however, we have access only to a sample of the partially annotated examples $\mathcal{D}_{xa} = \{(\boldsymbol{x}^i, \boldsymbol{a}^i) \in \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \mid i = 1, \ldots, m\}$. The goal is to learn parameters $\boldsymbol{w}$ of the MN classifier (2) by minimizing the partial loss ℓ^p-loss (10) on $\mathcal{D}_{xa}$. In this section, we derive a convex proxy of the ℓ^p-loss by extending the margin-rescaling loss (5) to the setting with missing labels generated by MAR process (9).

A straightforward plugging of ℓ^p-loss into the margin-rescaling loss (5) yields

$$\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w}) = \max_{\boldsymbol{y}' \in \mathcal{Y}^{\mathcal{V}}} \left[\ell^p(\boldsymbol{a}, \boldsymbol{y}') + \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}') \rangle - \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle \right] . \tag{13}$$

The value of $\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w})$ is zero when the score of the correct labelling $\boldsymbol{y}$ is greater than the score of any the incorrect labelling $\boldsymbol{y}'$ extended by the value of $\ell^p(\boldsymbol{a}, \boldsymbol{y}')$. It is easy to see that $\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w})$ upper bounds the partial loss $\ell^p(\boldsymbol{a}, h(\boldsymbol{x}, \boldsymbol{w}))$, that is, $\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w}) \geq \ell^p(\boldsymbol{a}, \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w}))$ holds for all $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w}) \in \mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \times \mathcal{A}^{\mathcal{V}} \times \mathbb{R}^d$. The proxy $\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w})$ has clear relation to the ℓ^p-loss, however, it requires the complete labeling $\boldsymbol{y}$, not available in the partially annotated examples $\mathcal{D}_{xa}$. We propose to replace $\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w})$ by a proxy $\Delta^p \colon \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \times \mathbb{R}^d \to \mathbb{R}$, named *partial margin-rescaling* loss, and defined as

$$\Delta^p(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) = \max_{\boldsymbol{y}' \in \mathcal{Y}^{\mathcal{V}}} \left[\ell^p(\boldsymbol{a}, \boldsymbol{y}') + \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}') \rangle - \langle \boldsymbol{w}, \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) \rangle \right] , \tag{14}$$

where $\boldsymbol{\psi}^p \colon \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \to \mathbb{R}^d$ is the input-annotation feature map defined as

$$\boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) = \sum_{v \in \mathcal{V}} \frac{[\![a_v \neq ?]\!]}{p(z_v = 1 \mid x)} \boldsymbol{\psi}_v(x, y_v) + \sum_{v,v' \in \mathcal{E}} \frac{[\![a_v \neq ? \wedge a_{v'} \neq ?]\!]}{p(z_v = 1, z_{v'} = 1 \mid x)} \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) . \tag{15}$$

The partial margin-rescaling loss $\Delta^p(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w})$ can be evaluated on the partially annotated examples $(\boldsymbol{x}, \boldsymbol{a}) \in \mathcal{D}_{xa}$ and it is convex in the parameters $\boldsymbol{w}$. The loss requires knowledge of marginals $p(z_v = 1 \mid \boldsymbol{x})$ and $p(z_v = 1, z_{v'} = 1 \mid \boldsymbol{x})$ of the missing label process $p(\boldsymbol{z} \mid \boldsymbol{x})$. The marginals can be easily estimated by maximum-likelihood method from the partially annotated examples $\mathcal{D}_{xa}$ as discussed in Section 3.2.

The partial margin-rescaling loss $\Delta^p(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w})$ is obtained from $\Delta^p_*(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w})$ by replacing the correct labelling score $\langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$, which cannot be computed, by the score $\langle \boldsymbol{w}, \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) \rangle$ which can be computed. This replacement is justified by the fact that the expectation of both scores equals as stated by the following theorem.

Theorem 1 *Let $\boldsymbol{\psi} \colon \mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \to \mathbb{R}^d$ be the input-output feature map defined by (3) and $\boldsymbol{\psi}^p \colon \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \to \mathbb{R}^d$ the input-annotation feature map defined by (15). Let both $\boldsymbol{\psi}$ and $\boldsymbol{\psi}^p$ be constructed from the same set of $\boldsymbol{\psi}_v \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$, $v \in \mathcal{V}$, and $\boldsymbol{\psi}_{vv'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^d$, $\{v, v'\} \in \mathcal{E}$. Let $p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) = p(\boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y})$ be a distribution over $\mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \times \mathcal{A}^{\mathcal{V}}$ such that the annotation process $p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y})$ satisfies (9). Then it holds that*

$$\mathbb{E}_{\boldsymbol{a} \sim p(\boldsymbol{a} \mid \boldsymbol{x})} \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{y}, \boldsymbol{a} \sim p(\boldsymbol{y}, \boldsymbol{a} \mid \boldsymbol{x})} \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) , \qquad \forall \boldsymbol{x} \in \mathcal{X} .$$

The proof is in Appendix A. An immediate consequence of Theorem 1 is the following:

Corollary 2 *Let* $\Delta^p_*\colon \mathcal{X} \times \mathcal{Y}^{\mathcal{V}} \times \mathcal{A}^{\mathcal{V}} \times \mathbb{R}^d \to \mathbb{R}_+$ *be the loss defined by (13) and* $\Delta^p\colon \mathcal{X} \times \mathcal{A}^{\mathcal{V}} \times \mathbb{R}^d \to \mathbb{R}^d$ *the partial margin-rescaling loss defined by (14). Under the assumptions of Theorem 1 it holds that*

$$\mathbb{E}_{\boldsymbol{a}\sim p(\boldsymbol{a}|\boldsymbol{x})}\Delta^p(\boldsymbol{x},\boldsymbol{a},\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{y},\boldsymbol{a}\sim p(\boldsymbol{y},\boldsymbol{a}|\boldsymbol{x})}\Delta^p_*(\boldsymbol{x},\boldsymbol{y},\boldsymbol{a},\boldsymbol{w})\,, \qquad \forall \boldsymbol{x}\in\mathcal{X}, \boldsymbol{w}\in\mathbb{R}^d\,. \tag{16}$$

Corollary 2 together with the law of large numbers guarantee that for the increasing number of examples m, the sample averages of $\Delta^p_*(\boldsymbol{x},\boldsymbol{y},\boldsymbol{a},\boldsymbol{w})$ and $\Delta^p(\boldsymbol{x},\boldsymbol{a},\boldsymbol{w})$ converge to the same value. This justifies using $\Delta^p(\boldsymbol{x},\boldsymbol{a},\boldsymbol{w})$, which can be evaluated on $\mathcal{D}_{xa}$, as a replacement for $\Delta^p_*(\boldsymbol{x},\boldsymbol{y},\boldsymbol{a},\boldsymbol{w})$, which cannot be evaluated but has a clear relation to ℓ^p-loss.

Having defined the Δ^p-loss, we pose learning of the parameters $\boldsymbol{w}$ of the MN classifier (2) from the partially annotated examples $\mathcal{D}_{xa}$ as unconstrained convex problem

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^n}\left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^m \Delta^p(\boldsymbol{x}^i,\boldsymbol{a}^i,\boldsymbol{w})\right]. \tag{17}$$

The approach is further denoted as MIssing Label Maximum Margin Markov Network (MIL-M3N) algorithm. The problem (17) can be solved by first-order methods, like stochastic gradient or cutting plane methods. These methods require an oracle evaluating $\Delta^p(\boldsymbol{x},\boldsymbol{a},\boldsymbol{w})$ and its subgradient w.r.t $\boldsymbol{w}$, which reads $\frac{\partial\Delta^p(\boldsymbol{x},\boldsymbol{a},\boldsymbol{w})}{\partial\boldsymbol{w}} = \boldsymbol{\psi}(\boldsymbol{x},\hat{\boldsymbol{y}}) - \boldsymbol{\psi}^p(\boldsymbol{x},\boldsymbol{a})$, where

$$\begin{aligned}\hat{\boldsymbol{y}} \;&\in\; \operatorname*{Argmax}_{\boldsymbol{y}\in\mathcal{Y}^{\mathcal{V}}}\Big[\ell^p(\boldsymbol{a},\boldsymbol{y}) + \langle\boldsymbol{w},\boldsymbol{\psi}(\boldsymbol{x},\boldsymbol{y})\rangle\Big]\\ &=\; \operatorname*{Argmax}_{\boldsymbol{y}\in\mathcal{Y}^{\mathcal{V}}}\Big[\sum_{v\in\mathcal{V}}\big([\![a_v \neq ?]\!]\ell(a_v,y_v) + \langle\boldsymbol{w},\boldsymbol{\psi}_v(\boldsymbol{x},y_v)\rangle\big) + \sum_{\{v,v'\}\in\mathcal{E}}\langle\boldsymbol{w},\boldsymbol{\psi}(y_v,y_{v'})\rangle\Big]\,.\end{aligned} \tag{18}$$

The bottle neck is solving the loss-augmented classification (LAC) problem (18) which is an instance of NP-hard max-sum problem. In section 3.3 we derive an algorithm completely avoiding solving the LAC problem which is applicable to an unrestricted MN classifier.

3.2. Estimating the marginals

Evaluation of the partial margin-rescaling loss (14) requires the unary marginals $p(z_v \mid \boldsymbol{x})$, $v\in\mathcal{V}$, and pair-wise marginals $p(z_v, z_{v'} \mid \boldsymbol{x})$, $\{v,v'\}\in\mathcal{E}$, of the distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$ describing the label missingness. The marginals can be estimated from the partially annotated examples $\mathcal{D}_{xa}$ in a principled way via the maximum-likelihood method. Note that the estimation of the marginals is considerably simpler compared to the estimation of $p(\boldsymbol{y} \mid \boldsymbol{x})$ governing the optimal classifier. As an example, consider the so-called homogeneous completely at random missingness, when

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{v\in\mathcal{V}} p(z_v) = \theta^t(1-\theta)^t \tag{19}$$

where $t = \sum_{v\in\mathcal{V}} z_v$ and $\theta\in[0,1]$ is the probability that a randomly chosen part $v\in\mathcal{V}$ is annotated. The ML estimate of the marginals based on $\mathcal{D}_{xa}$ is then

$$\begin{aligned}\hat{p}(z_v \mid \boldsymbol{x}) &= \theta, \quad v\in\mathcal{V},\\ \hat{p}(z_v, z_{v'} \mid \boldsymbol{x}) &= \theta^2, \quad \{v,v'\}\in\mathcal{E}\,,\end{aligned} \qquad \text{where} \quad \theta = \frac{1}{m|\mathcal{V}|}\sum_{i=1}^m\sum_{v\in\mathcal{V}}[\![a^i_v \neq ?]\!]\,. \tag{20}$$

The estimated marginals are used in the input-annotation features $\boldsymbol{\psi}^p(\boldsymbol{x},\boldsymbol{a})$ defined by (15).

3.3. LP relaxation of the partial margin-rescaling loss

As described in section 3.1, learning of the MN classifier by minimizing Δ^p-loss is applicable when the LAC problem (18) leads to a tractable max-sum problem. In this section we replace the max-sum problem emerging in (18) by Schlesinger's linear programming relaxation (Schlesinger, 1976; Werner, 2007). A similar idea was previously used in supervised learning of MN classifier (Taskar et al., 2004a; Franc and Laskov, 2011). The LP relaxed partial margin-rescaling loss remains convex and its minimization is tractable for an unrestricted class of MN classifiers. The following derivation borrows the basic results and terminology related to LP relaxation from Werner (2007).

The Δ^p-loss defined by (14) can be written as

$$\Delta^p(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}} f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w}) - \langle \boldsymbol{w}, \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) \rangle \,, \tag{21}$$

where

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w}) = \sum_{v \in \mathcal{V}} \Big([\![a_v \neq ?]\!] \ell_v(a_v, y_v) + \langle \boldsymbol{w}, \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) \rangle \Big) + \sum_{\{v,v'\} \in \mathcal{E}} \langle \boldsymbol{w}, \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \rangle \,.$$

The optimal value of the max-sum problem $\max_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}} f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{w})$ can be approximated by a linear program (LP). The optimal value of the LP always upper bounds the optimal value of the max-sum problem and in some cases the bound is tight. We exploit the Lagrange dual of the LP, because the inequality constraints of the dual have a simple form allowing to formulate the upper bound of the max-sum problem as an unconstrained minimization problem. In particular, the LP relaxation upper bound reads

$$\max_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{V}}} f(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}, \boldsymbol{w}) \leq \min_{\boldsymbol{\varphi} \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}} u(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{\varphi}, \boldsymbol{w}) \,, \tag{22}$$

where $\boldsymbol{\varphi} \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}$ is a vector representation of discrete functions (so called potentials or messages) $\varphi_{vv'} \colon \mathcal{Y} \to \mathbb{R}$, $\{v, v'\} \in \mathcal{E}$, $\varphi_{v'v} \colon \mathcal{Y} \to \mathbb{R}$, $\{v, v'\} \in \mathcal{E}$, and

$$\begin{aligned}
u(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{\varphi}, \boldsymbol{w}) &= \sum_{v \in \mathcal{V}} \max_{y \in \mathcal{Y}} u_v(\boldsymbol{x}, y, a_v, \boldsymbol{\varphi}, \boldsymbol{w}) + \sum_{v,v' \in \mathcal{E}} \max_{y,y' \in \mathcal{Y}^2} u_{vv'}(y, y', \boldsymbol{\varphi}, \boldsymbol{w}) \,, \\
u_v(\boldsymbol{x}, a_v, y, \boldsymbol{\varphi}, \boldsymbol{w}) &= [\![a \neq ?]\!] \ell_v(a_v, y) + \langle \boldsymbol{w}, \boldsymbol{\psi}_v(\boldsymbol{x}, y) \rangle - \sum_{v' \in \mathcal{N}(v)} \varphi_{vv'}(y) \,, \\
u_{vv'}(y, y', \boldsymbol{\varphi}, \boldsymbol{w}) &= \langle \boldsymbol{w}, \boldsymbol{\psi}_{vv'}(y, y') \rangle + \varphi_{vv'}(y) + \varphi_{v'v}(y') \,.
\end{aligned} \tag{23}$$

The LP-relaxation upper bound $\min_{\boldsymbol{\varphi} \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}} u(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{\varphi}, \boldsymbol{w})$ is an unconstrained problem which is convex both in the parameters to learn $\boldsymbol{w}$, and the auxiliary parameters $\boldsymbol{\varphi}$. Substituting the upper bound (22) to the formula for the partial margin-rescaling loss (21) yields the LP-relaxed partial margin-rescaling loss

$$\Delta^p_{LP}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) = \min_{\boldsymbol{\varphi} \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}} u(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{\varphi}, \boldsymbol{w}) - \langle \boldsymbol{w}, \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) \rangle \,.$$

By the inequality (22), it holds that $\Delta^p(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) \leq \Delta^p_{LP}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) \,, \forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{a} \in \mathcal{A}^{\mathcal{V}}, \boldsymbol{w} \in \mathbb{R}^d$ and the equality $\Delta^p(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) = \Delta^p_{LP}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w})$ occurs when the LP-relaxation is tight. The tight approximation is guaranteed, e.g., when the neighbourhood graph $(\mathcal{V}, \mathcal{E})$ is a tree or

when pair-wise scores $f_{vv'}(y,y') = \langle \boldsymbol{\psi}_{v,v'}(y,y'), \boldsymbol{w}\rangle$, $\{v,v'\} \in \mathcal{E}$, are super-modular. Besides, the LP-relaxation is tight for a broader class of max-sum problems which, however, do not have an explicit characterization.

Learning of the parameters $\boldsymbol{w}$ of the MN classifier from partially annotated examples $\mathcal{D}_{xa}$, further denoted as LP-MIL-M3N algorithm, leads to the following convex problem:

$$\begin{aligned} \boldsymbol{w}^* &= \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^m \Delta^p_{LP}(\boldsymbol{x}^i, \boldsymbol{a}^i, \boldsymbol{w})\right] \\ &= \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \min_{\substack{\boldsymbol{\varphi}^1\in\mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|} \\ \vdots \\ \boldsymbol{\varphi}^m\in\mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}}} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^m \left(u(\boldsymbol{x}^i, \boldsymbol{a}^i, \boldsymbol{\varphi}^i, \boldsymbol{w}) - \langle \boldsymbol{w}, \boldsymbol{\psi}^p(\boldsymbol{x}^i, \boldsymbol{a}^i)\rangle\right)\right]. \end{aligned} \tag{24}$$

The optimization problem (24) as the following properties:

- The problem (24) has $d + 2|\mathcal{E}||\mathcal{Y}|m$ variables, i.e., the number of variables grows linearly with the number of parameters d, the number of examples m, the number of labels $|\mathcal{Y}|$, and the number of edges $\mathcal{E}$.
- Besides the parameters $\boldsymbol{w} \in \mathbb{R}^d$ to be learned, each training example has a vector of auxiliary variables $\boldsymbol{\varphi}^i \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}$, used to make the prediction of that example by the MN classifier a tractable problem. The auxiliary variables are not used after training.
- When the LP-relaxation is tight, i.e., $\Delta^p(\boldsymbol{x}^i, \boldsymbol{a}^i, \boldsymbol{w}) = \Delta^p_{LP}(\boldsymbol{x}^i, \boldsymbol{a}^i, \boldsymbol{w})$, $i \in \{1,\dots,m\}$, the optimal solution of (24) coincides with the optimal solution of the problem (17) which minimizes the (unrelaxed) Δ^p-loss. As mentioned above, this happens for example when $(\mathcal{V},\mathcal{E})$ is a tree. In this sense, (24) subsumes (17) as a special case.
- The problem (24) can be reformulated as a convex quadratic program with $d + 2|\mathcal{E}||\mathcal{Y}|m + |\mathcal{E}||\mathcal{V}|m$ variables and $m(|\mathcal{V}||\mathcal{Y}| + |\mathcal{E}||\mathcal{Y}|^2)$ inequality constraints. That is, the number of constraints and variables is polynomial in the problem size. However, this approach is efficient only for small problem instances.

For large problem instances, the problem (24) can be solved efficiently by simple to implement stochastic gradient algorithms. To this end, we rewrite objective of (24) as

$$Q(\boldsymbol{w}, \boldsymbol{\varphi}^1, \dots, \boldsymbol{\varphi}^m) = \frac{1}{m}\sum_{i=1}^m Q_i(\boldsymbol{w}, \boldsymbol{\varphi}^1, \dots, \boldsymbol{\varphi}^m)$$

where $Q_i(\boldsymbol{w}, \boldsymbol{\varphi}^1, \dots, \boldsymbol{\varphi}^m) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + u(\boldsymbol{x}^i, \boldsymbol{a}^i, \boldsymbol{\varphi}^i, \boldsymbol{w}) - \langle \boldsymbol{w}, \boldsymbol{\psi}^p(\boldsymbol{x}^i, \boldsymbol{a}^i)\rangle$. The sub-gradient of Q_i w.r.t. $\boldsymbol{w}$ and $\boldsymbol{\varphi}^1, \dots, \boldsymbol{\varphi}^m$, then reads

$$\frac{\partial Q_i}{\partial \boldsymbol{w}} = \lambda \boldsymbol{w} + \sum_{v\in\mathcal{V}} \boldsymbol{\psi}_v(\boldsymbol{x}^i, \bar{y}_v) + \sum_{\{v,v'\}\in\mathcal{E}} \boldsymbol{\psi}_{vv'}(\bar{\bar{y}}_v, \bar{\bar{y}}_{v'}) - \boldsymbol{\psi}^p(\boldsymbol{x}^i, \boldsymbol{a}^i)$$

$$\left.\begin{aligned} \frac{\partial Q_i}{\partial \varphi^i_{vv'}(y)} &= [\![\bar{\bar{y}}_v = y]\!] - [\![\bar{y}_v = y]\!] \\ \frac{\partial Q_i}{\partial \varphi^i_{v'v}(y)} &= [\![\bar{\bar{y}}_{v'} = y]\!] - [\![\bar{y}_{v'} = y]\!] \end{aligned}\right\}, \quad \{v,v'\} \in \mathcal{E}, y \in \mathcal{Y}\,.$$

$$\frac{\partial Q_i}{\partial \varphi^j_{vv'}(y)} = \frac{\partial Q_i}{\partial \varphi^j_{v'v}(y)} = 0\,, \quad j \in \{1,\dots,m\} \setminus \{i\}, \quad \{v,v'\} \in \mathcal{E}, y \in \mathcal{Y}\,,$$

where

$$\begin{aligned} \bar{y}_v &\in \operatorname*{argmax}_{y\in\mathcal{Y}} u_v(\boldsymbol{x}^i, a^i_v, y, \boldsymbol{\varphi}^i)\,, \qquad v \in \mathcal{V}\,, \\ (\bar{\bar{y}}_v, \bar{\bar{y}}_{v'}) &\in \operatorname*{argmax}_{(y,y')\in\mathcal{Y}^2} u_{vv'}(y, y', \boldsymbol{\varphi}^i, \boldsymbol{w})\,, \qquad \{v,v'\} \in \mathcal{E}\,. \end{aligned}$$

Recall that $u_v(\boldsymbol{x}^i, a^i_v, y, \boldsymbol{\varphi}^i)$ and $u_{vv'}(y, y', \boldsymbol{\varphi}^i, \boldsymbol{w})$ are the reparametrized scores of the LAC problem defined by (23). The gradient computation requires $\mathcal{O}(|\mathcal{V}||\mathcal{Y}|+|\mathcal{E}||\mathcal{Y}|^2+d(|\mathcal{V}|+|\mathcal{E}|))$ operations in total, i.e., it is polynomial in the main dimensions of the model. The stochastic gradient method randomly samples $i \in \{1,\dots,m\}$, computes the gradient of Q_i w.r.t. $\boldsymbol{w}$ and $\boldsymbol{\varphi}^1,\dots,\boldsymbol{\varphi}^m$ and uses it to update the variables. The MN classifier can be used as the last layer of a NN architecture. In this case, the features $\boldsymbol{\psi}_v(\boldsymbol{x}, y_v)$, $v \in \mathcal{V}$, are not fixed, but they are the outputs of a NN whose parameters are learned simultaneously with $\boldsymbol{w}$.

4. Experiments

4.1. Synthetic data: Hidden Markov Chain

The input and output are sequences of symbols $\boldsymbol{x} = (x_1,\dots,x_{100}) \in \{1,\dots,30\}^{100}$ and $\boldsymbol{y} = (y_1,\dots,y_{100}) \in \{1,\dots,30\}^{100}$ generated from the Hidden Markov Chain

$$p(\boldsymbol{x},\boldsymbol{y}) = p(y_1)\prod_{i=2}^{100} p(y_i \mid y_{i-1})p(x_i \mid y_i)\,. \tag{25}$$

We used the uniform state prior $p(y_1) = 1/30$, the emission probability $p(x_i \mid y_i) = 1/2$ if $x_i = y_i$ and $p(x_i \mid y_i) = 1/58$ otherwise, and the transition probabilities $p(y_i \mid y_{i-1}) = 1/2$ if $y_i = y_{i-1}$ and $p(y_i \mid y_{i-1}) = 1/10$ otherwise. We generated the partial annotation $\boldsymbol{a} \in (\{1,\dots,30\} \cup \{?\})^{100}$ using $p(\boldsymbol{a} \mid \boldsymbol{x})$ given by (9) and missingness process $p(\boldsymbol{z} \mid \boldsymbol{x})$ given by (19). We used the probability of missing label $\theta \in \{0, 0.1, 0.2, 0.5\}$.

We learned the MN classifier (2) by the proposed MIL-M3N algorithm solving (17) with the BMRM (Teo et al., 2010). The graph $(\mathcal{V},\mathcal{E})$ is a chain and the LAC problem is solved by the dynamic programming. The feature maps $\boldsymbol{\psi}_v(\boldsymbol{x}, y) = \mathbf{1}_{x_v,y}$, and $\boldsymbol{\psi}_{vv'}(y, y') = \mathbf{1}_{y,y'}$, are one-hot-encodings of the symbols (x_v, y) and (y, y'), respectively. We use the normalized Hamming loss $\ell(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{n}\sum_{v\in\mathcal{V}} [\![y_v \neq y_{v'}]\!]$ as the target loss. We approximate the marginals by (20). Note that for data $\mathcal{T}_{xa}$ generated with $\theta = 0$, i.e. no missing labels, the MIL-M3N becomes equivalent to a standard supervised M3N algorithm. We used three baselines. First, the Bayes optimal rule minimizing the Hamming loss $h_{HAM}(\boldsymbol{x}) = (\hat{y}_1,\dots,\hat{y}_n)$ where $\hat{y}_v \in \operatorname{argmax}_{y_v\in\mathcal{Y}} p(y_v \mid \boldsymbol{x})$, $v \in \mathcal{V}$. Second, the Bayes optimal rule minimizing 0/1-loss $h_{0/1}(\boldsymbol{x}) \in \operatorname{argmax}_{\boldsymbol{y}\in\mathcal{Y}^n} p(\boldsymbol{y} \mid \boldsymbol{x})$. The rules are derived from the generating distribution (25). Third, the CCCP algorithm to learn the MN classifier by minimizing the Ramp-loss (12).

We trained the MN classifier on $\mathcal{D}_{xa}$ with the number of examples $m \in \{50, 100, \dots, 1000\}$. The optimal regularization constant λ was selected from $\{0.1, 1, 10, 100\}$ based on the Hamming loss evaluated on $5,000$ validation examples. For the best λ, we report the training partial error (i.e., the average of ℓ^p-loss) and the test error (i.e. average of ℓ-loss) computed on $10,000$ test examples. The experiment was repeated 5 times. Figure 1 shows the mean

and standard deviation of the errors. The test error of the proposed MIL-M3N algorithm approaches the test error of the fully supervised M3N algorithm as the number of training examples increases. The performance gap is proportional to the portion of missing labels, however, it always vanishes when the number of training examples is above 300. The profile of the training partial error of MIL-M3N copies the error profile of the supervised M3N algorithm decreased by a factor proportional to the number of missing labels not counted by the ℓ^p-loss. The error of the CCCP is higher and does not decrease monotonically with the number of training examples. The CCCP suffers from the non-convexity of the Ramp-loss.

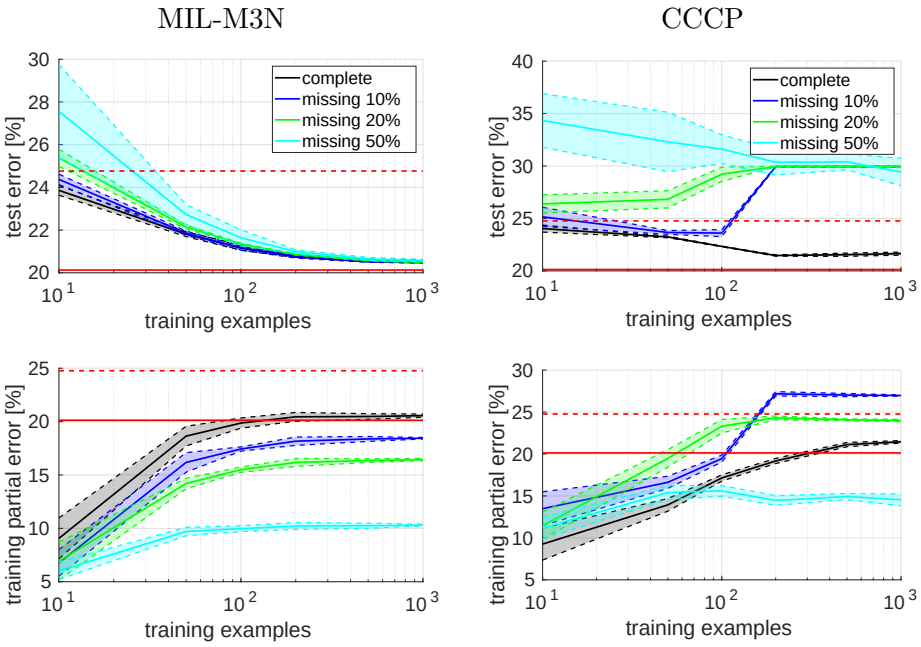

Figure 1: Results on the synthetic data generated from HMC (25). Test error and training partial error of MN classifier trained from examples with various amount of missing labels shown as a function of the number of training examples. The left and the right column show results of the proposed MIL-M3N algorithm and the CCCP algorithm, respectively. The horizontal solid red line shows error of the Bayes rule for Hamming loss and the dashed red line the Bayes rule for 0/1-loss.

4.2. Sudoku

The Sudoku puzzle is composed 9×9 cells $\mathcal{V} = \{(i,j) \in \mathcal{N} \mid 1 \leq i \leq 9\,, 1 \leq j \leq 9\}$. Each cell is either filled with a number from 1 to 9 or it is empty $\square$. We represent the puzzle assignment by $\boldsymbol{x} = (x_v \in \{\square, 1, \ldots, 9\} \mid v \in \mathcal{V})$. The task is to fill the empty cells with a number so that each row, each column, and each of non-verlapping 3×3 sub-grids contain all numbers from 1 to 9. We represent the puzzle solutions by $\boldsymbol{y} = (y_v \in \{1, \ldots, 9\} \mid v \in \mathcal{V})$. We used Sudoku assignments and their correct solutions to generate a training set $\mathcal{D}_{xa}$ of assignments and partial solutions. The partial solution was generated by MAR process (9)

using $p(z_v = 1 \mid \boldsymbol{x}) = 1$ if $x_v \in \{1, \dots, 9\}$ and $p(z_v = 1 \mid \boldsymbol{x}) = 1 - \tau$ if $x_v = \square$, where $\tau \in [0, 1]$ is the probability of missing label for empty cell. This process corresponds to instructing the annotator preparing $\mathcal{D}_{xa}$, to copy the numbers of filled cells to the corresponding label and to label $100(1 - \tau)\,\%$ of randomly chosen empty cells of the puzzle. Note that τ is the portion of unlabeled empty cells, not the portion of all unlabeled objects like in the previous experiment. The value of τ thus determines to what extent the puzzle is unsolved, e.g. $\tau = 0.2$ requires the annotator to label (and to solve) 80% of empty cells.

We used the proposed LP-MIL-M3N algorithm to train MN classifier (2) predicting the Sudoku solution $\boldsymbol{y}$ from assignment $\boldsymbol{x}$. The prior knowledge is encoded by revealing the algorithm that cells in rows, in columns, and in 3×3 sub-grids are related, i.e., by setting $\mathcal{E} = \{\{(v, v'), (u, u')\} \mid v = v' \vee v' = u' \vee (\lceil v/3 \rceil = \lceil u/3 \rceil \wedge \lceil v'/3 \rceil = \lceil u'/3 \rceil)\}$. The feature maps $\boldsymbol{\psi}_v(\boldsymbol{x}, y) = \mathbf{1}_{x_v, y}$, and $\boldsymbol{\psi}_{vv'}(y, y') = \mathbf{1}_{y,y'}$, are one-hot-encodings of the symbol pairs (x_v, y) and (y, y'), respectively. We use the normalized Hamming loss $\ell(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{n} \sum_{v \in \mathcal{V}} [\![y_v \neq y_{v'}]\!]$ as the target loss. We approximate the marginals by the same $p(z_v \mid \boldsymbol{x})$ which was used for annotation, assuming that the annotator followed the instructions. The parameters $\boldsymbol{w}$ of MN classifier were found by solving (24) with ADAM (Kingma and Ba, 2015). The regularization was not used, i.e., $\lambda = 0$. We evaluated the MN classifier by Augmented Directed Acyclic Graph solver (Schlesinger, 1976; Werner, 2007).

We trained from $\mathcal{D}_{xa}$ with varying number of examples m and the probability of missing label for empty cells $\tau \in \{0, 0.1, 0.2, 0.5\}$. We tested on 100 puzzles involving prediction of $81 \cdot 100 = 8,100$ labels. The test error is the portion of incorrectly predicted cells per puzzle. The experiment was repeated 5 times. Figure 2(a) shows the mean and standard deviation of the test error as a function of the number of training examples. It is seen that the test error decreases with the number of training examples and with the portion of missing labels. To obtain the perfect Sudoku solver with zero test error, it was enough to train from 100 examples regardless the portion of missing labels. Figure 2(c)(d) show the score functions of the MN classifier learned from 100 examples of half-solved Sudokus. The scores nicely capture the rules of the game. Namely, the unary scores enforce the solver to copy the input symbol to the label if the cell is filed, and the pair-wise scores enforce the related cells to have different labels.

4.3. Visual Sudoku

We replaced the input symbols $\{1, \dots, 9\}$ by 28×28 images of hand-written digits from MNIST dataset (LeCun and Cortes, 2010). The empty cells were replaced by all-black images. As a feature map of the unary scores, we use $\boldsymbol{\psi}_v(\boldsymbol{x}, y) = (\bar{\boldsymbol{\psi}}_1, \dots, \bar{\boldsymbol{\psi}}_9)$, where $\bar{\boldsymbol{\psi}}_{y'} \in \mathbb{R}^{2000}$, $y' \neq y$, are all-zero vectors, $\bar{\boldsymbol{\psi}}_y = (k(\boldsymbol{x}_v, \boldsymbol{\mu}_1), \dots, k(\boldsymbol{x}_v, \boldsymbol{\mu}_{2000})) \in \mathbb{R}^{2000}$ is a vector of RBF kernels $k(\boldsymbol{x}_v, \boldsymbol{\mu}_i) = \exp(-2\|\boldsymbol{x}_v - \boldsymbol{\mu}_i\|^2)$ evaluated for the image $\boldsymbol{x}_v$ of the v-th cell and 2,000 randomly sampled images of MNIST digits. There is no overlap between the digits used in the test Sudokus, the training Sudokus, and the 2,000 digits. All other settings are the same as for the symbolic Sudoku experiment. Figure 2 shows the mean and standard deviation of the test error as a function of the number of training examples.

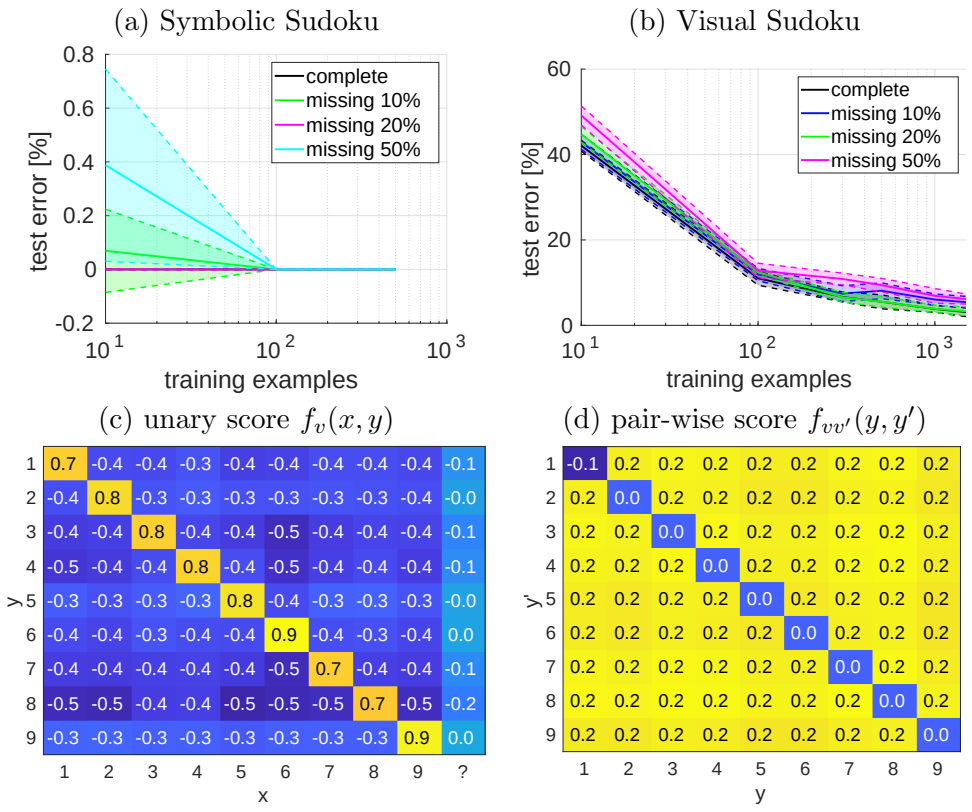

Figure 2: Figure (a) and (b) shows test error as function of the number of training examples for symbolic and visual Sudoku problem, respectively. The curves are computed for different portion of unsolved empty cells in the annotation. Figure (c) and (d) visualize the learned quality scores of the MN classifier for symbolic Sudoku.

4.4. Comparison with neural architectures

Solving the Sudoku by deep networks was considered in Wang et al. (2019). They used both the symbolic and the visual Sudoku composed of MNIST digits as we do. They trained SATNet architecture which is a CNN endowed with a layer implementing a maximum satisfiability (MAXSAT) solver. The SATNet can better learn hard interactions between output variables than canonical neural architectures. They also trained a common convolutional neural network (ConvNet) as a baseline. They used $9,000$ completely annotated training examples and $1,000$ test examples. The MN classifier was trained by the proposed LP-MIL-M3N algorithm from 500 examples in the case of symbolic Sudoku and 1,500 examples in the case of visual Sudoku, using the setting described in the previous section. Following Wang et al. (2019), we measure the performance in terms of accuracy, defined as the portion of perfectly solved puzzles, i.e., already a single misclassified label counts as an error.

Table 1 shows comparison between SATNet, ConvNet, and the MN classifier. The MN classifier significantly outperforms both neural models, ConvNet and SATNet, in the case of symbolic as well as visual Sudoku. The MN classifier is trained on a significantly smaller number of examples, and it performs better even when trained using partial annotations.

Method	Label annotation	Accuracy Symbolic Sudoku	Visual Sudoku
MN classifier trained by LP-MIL-M3N	complete	100.0±0.0%	90.0±2.9%
	10% missing	100.0±0.0%	80.4±3.8%
	20% missing	100.0±0.0%	89.8±3.0%
	50% missing	100.0±0.0%	76.4±3.3%
ConvNet	complete	15.1%	0.1%
SATNet	complete	98.3%	63.2%

Table 1: The accuracy of predicting solution of the symbolic and visual Sudoku puzzle by the MN classifier trained by the proposed LP-MIL-M3N algorithm compared to two neural models, the CovnNet and the SATNet proposed by Wang et al. (2019).

5. Conclusions

We proposed a novel loss function for learning MN classifier from partially annotated examples. The loss is convex, has a clear connection to the standard supervised margin-rescaling loss, and it can be efficiently optimized by first-order methods. We showed that the proposed loss can be used to learn MN classifiers with a complex neighbourhood structure, which on a challenging problem of predicting Sudoku solution significantly outperforms neural architectures trained from a higher number of completely annotated examples, while the proposed method used only partial annotations. To our knowledge, the proposed method is the first efficient algorithm for learning the unrestricted M3N classifiers from partial annotations.

Acknowledgments

The research was supported by the Czech Science Foundation project GACR GA19-21198S.

References

K. Antoniuk and V. Franc. Consistency of structured output learning with missing labels. In *Proc. of Asian Conference on Machine Learning (ACML)*, 2015.

L.C. Chen, A.G. Schwing, A.L. Yuille, and R. Urtasun. Learning deep structured models. In *Proc. of International Conference on Machine Learning (ICML)*, 2015.

B.D. Chuong, L. Quoc, C.H. Teo, O. Chapelle, and A. Smola. Tighter bounds for structured estimation. In *Proc. of Neural Information Processing Systems (NIPS)*, 2008.

E.R. Fernandes and U. Brefeld. Learning from partially annotated sequences. In *Proc. of European Conference on Machine Learning (ECML/PKDD)*, 2011.

V. Franc and P. Laskov. Learning maximal margin markov networks via tractable convex optimization. *Control Systems and Computers*, (2):25–34, April 2011.

V. Franc and B. Savchynskyy. Discriminative learning of max-sum classifiers. *Journal of Machine Learning Research*, 9(1):67–104, 2008.

R.B. Girshick, P.F. Felzenszwalb, and D.A. McAllester. Object detection with grammar models. In *Proc. of Neural Information Processing Systems (NIPS)*, 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of International Conference on Learning Representations (ICLR)*, 2015.

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

X. Lou and F.A. Hamprecht. Structured learning from partial annotations. In *Proc. of International Conference on Machine Learning (ICML)*, 2012.

R. Palm, U. Paquet, and O. Winther. Recurrent relational networks. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

S. Sarawagi and R. Gupta. Accurate max-margin training for structured output spaces. In *Proc. of International Conference on Machine Learning (ICML)*, 2008.

M.I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, (4):113–130, 1976. In Russian.

A. Schwing and R. Urtasun. Fully connected deep structured networks. *ArXiv*, abs/1503.02351, 2015.

S. Shalev-Shwartz, O. Shamir, and S. Shammah. Failures of gradient-based deep learning. In *Proc. of International Conference on Machine Learning (ICML)*, 2017.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. of International Conference on Machine Learning (ICML)*, 2004a.

B. Taskar, C. Guestrin, and D. Koller. Maximum-margin markov networks. In *Proc. of Neural Information Processing Systems (NIPS)*, 2004b.

C.H. Teo, S.V.N. Vishwanthan, A.J. Smola, and Q.V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11(10):311–365, 2010.

A. Trask, F. Hill, S.E. Reed, J. Rae, C. Dyer, and P. Blunsom. Neural arithmetic logic units. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, (6):1453–1484, 2005.

A. Vedaldi and A. Zisserman. Structured output regression for detection with partial truncation. In *Proc. of Neural Information Processing Systems (NIPS)*, 2009.

P.W. Wang, P.L. Donti, B. Wilder, and J.Z. Kolter. SATnet: Bridging deep learning and logical reasoning using a differential satisfiability solver. In *ICML*, 2019.

Y. Wang and G. Mori. A discriminative latent model of object classes and attributes. In *Proc. of European Conference on Computer Vision (ECCV)*, 2010.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

F. Yang, Z. Yang, and W.W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Proc. of Neural Information Processing Systems (NIPS)*, 2017.

C.N.J. Yu and T. Joachims. Learning structural svms with latent variables. In *Proce. of International Conference on Machine Learning (ICML)*, 2009.

H.F. Yu, P. Jain, P. Kar, and I.S. Dhillon. Large-scale multi-label learning with missing labels. In *Proc. of International Conference on Machine Learning (ICML)*, 2014.

A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15 (4):915–936, 2003.

S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *Proc. of ICCV*, 2015.

Appendix A. Proof of Theorem 1

Let us introduce shortcuts $\delta(a_v, y_v, z_v) = [\![a_v = \alpha(y_v, z_v)]\!]$ and $\delta(a_v, a_{v'}, y_v, y_{v'}, z_v, z_{v'}) = [\![a_v = \alpha(y_v, z_v) \wedge a_{v'} = \alpha(y_{v'}, z_{v'})]\!]$. By marginalizing $p(\boldsymbol{a} \mid \boldsymbol{x}) = \sum_{\boldsymbol{y}} p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{y} \mid \boldsymbol{x})$, where $p(\boldsymbol{a} \mid \boldsymbol{x}, \boldsymbol{y})$ is a MAR process given by (9), we derive that

$$\begin{aligned} p(a_v \mid \boldsymbol{x}) &= \sum_{y_v} \sum_{z_v} p(z_v \mid \boldsymbol{x}) p(y_v \mid \boldsymbol{x}) \delta(a_v, y_v, z_v) \,, \\ p(a_v, a_{v'} \mid \boldsymbol{x}) &= \sum_{y_v, y_{v'}} \sum_{z_v, z_{v'}} p(z_v, z_{v'} \mid \boldsymbol{x}) p(y_v, y_{v'} \mid \boldsymbol{x}) \delta(a_v, a_{v'}, y_v, y_{v'}, z_v, z_{v'}) \,. \end{aligned} \tag{26}$$

By using (26), we derive the equality $\mathbb{E}_{\boldsymbol{a} \sim p(\boldsymbol{a}|\boldsymbol{x})} \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x})} \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})$ as follows

$$\begin{aligned} \mathbb{E}_{\boldsymbol{a} \sim p(\boldsymbol{a}|\boldsymbol{x})} \boldsymbol{\psi}^p(\boldsymbol{x}, \boldsymbol{a}) &= \mathbb{E}_{\boldsymbol{a} \sim p(\boldsymbol{a}|\boldsymbol{x})} \left[\sum_v \frac{[\![a_v \neq ?]\!]}{p(z_v \mid \boldsymbol{x})} \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) + \sum_{v,v'} \frac{[\![a_v \neq ? \wedge a_{v'} \neq ?]\!]}{p(z_v = 1, z_{v'} = 1 \mid \boldsymbol{x})} \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \right] \\ &= \sum_v \sum_{a_v} p(a_v \mid \boldsymbol{x}) \frac{[\![a_v \neq ?]\!]}{p(z_v \mid \boldsymbol{x})} \boldsymbol{\psi}_v(\boldsymbol{x}, \boldsymbol{y}_v) + \sum_{v,v' \in \mathcal{E}} \sum_{a_v, a'_v} p(a_v, a_{v'} \mid \boldsymbol{x}) \frac{[\![a_v \neq ? \wedge a_{v'} \neq ?]\!]}{p(z_v = 1, z_{v'} = 1 \mid \boldsymbol{x})} \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \\ &= \sum_v \sum_{a_v} \sum_{y_v} \sum_{z_v} p(z_v \mid \boldsymbol{x}) p(y_v \mid \boldsymbol{x}) \delta(a_v, y_v, z_v) \frac{[\![a_v \neq ?]\!]}{p(z_v = 1 \mid \boldsymbol{x})} \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) \\ &+ \sum_{v,v'} \sum_{a_v, a_{v'}} \sum_{y_v, y_{v'}} \sum_{z_v, z_{v'}} p(z_v, z_{v'} | \boldsymbol{x}) p(y_v, y_{v'} | \boldsymbol{x}) \delta(a_v, a_{v'}, y_v, y_{v'}, z_v, z_{v'}) \frac{[\![a_v \neq ? \wedge a_{v'} \neq ?]\!]}{p(z_v = 1, z_{v'} = 1 | \boldsymbol{x})} \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \\ &= \sum_v \sum_{y_v} \frac{p(z_v = 1 \mid \boldsymbol{x}) p(y_v \mid \boldsymbol{x})}{p(z_v = 1 \mid \boldsymbol{x})} \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) + \sum_{v,v'} \sum_{y_v, y_{v'}} \frac{p(z_v = 1, z_{v'} = 1 \mid \boldsymbol{x}) p(y_v, y_{v'} \mid \boldsymbol{x})}{p(z_v = 1, z_{v'} = 1 \mid \boldsymbol{x})} \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \\ &= \sum_v \sum_{y_v} p(y_v \mid \boldsymbol{x}) \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) + \sum_{v,v'} \sum_{y_v, y_{v'}} p(y_v, y_{v'} \mid \boldsymbol{x}) \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \\ &= \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x})} \left[\sum_v \boldsymbol{\psi}_v(\boldsymbol{x}, y_v) + \sum_{v,v'} \boldsymbol{\psi}_{vv'}(y_v, y_{v'}) \right] = \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x})} \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) \,. \end{aligned}$$