Primal Sparse Max-Margin Markov Networks

Jun Zhu[⋆‡] Eric P. Xing[‡] Bo Zhang[⋆]
[⋆]Dept. Comp. Sci. & Tech.
TNList Lab, Tsinghua University
Beijing, 100084 China
[‡]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jun-zhu@mails.thu.edu.cn; epxing@cs.cmu.edu; dcszb@tsinghua.edu.cn

ABSTRACT

Max-margin Markov networks (M^{3}N) have shown great promise in structured prediction and relational learning. Due to the KKT conditions, the M^{3}N enjoys dual sparsity. However, the existing M^{3}N formulation does not enjoy primal sparsity, which is a desirable property for selecting significant features and reducing the risk of over-fitting. In this paper, we present an ℓ_1-norm regularized max-margin Markov network (ℓ_1-M^{3}N), which enjoys dual and primal sparsity simultaneously. To learn an ℓ_1-M^{3}N, we present three methods including projected sub-gradient, cutting-plane, and a novel EM-style algorithm, which is based on an equivalence between ℓ_1-M^{3}N and an adaptive M^{3}N. We perform extensive empirical studies on both synthetic and real data sets. Our experimental results show that: (1) ℓ_1-M^{3}N can effectively select significant features; (2) ℓ_1-M^{3}N can perform as well as the pseudo-primal sparse Laplace M^{3}N in prediction accuracy, while consistently outperforms other competing methods that enjoy either primal or dual sparsity; and (3) the EM-algorithm is more robust than the other two in prediction accuracy and time efficiency.

Categories and Subject Descriptors

I.5.1 [**Pattern Recognition**]: Models - Statistical

General Terms

Algorithms, Experimentation

Keywords

ℓ_1-norm Max-margin Markov networks, Primal sparsity, Dual sparsity

1. INTRODUCTION

In recent years, discriminative learning has expanded its scope to model structured data based on composite features that explicitly exploit the structural dependencies among elements in high-dimensional inputs (e.g., text sequences, image lattices) and structured interpretational outputs (e.g., part-of-speech tagging, image segmentation). Three major approaches have been successfully explored to perform discriminative learning by using graphical models to capture sequential, spatial or relational structure: (1) maximum conditional likelihood [16], (2) max-margin [20], and (3) maximum entropy discrimination [27].

Discriminative models, such as conditional random fields (CRFs) [16] and max-margin Markov networks (M^{3}N) [20], usually have a complex and high-dimensional feature space, because in principle they can use arbitrary and overlapping features of inputs. Thus it is desirable to pursue a sparse representation of such models that leaves out irrelevant features. We say a model enjoys the *primal sparsity* if only a few input features in the original model have non-zero weights (see Section 3.1 for precise definitions). Primal sparsity is an important property for selecting significant features and reducing the risk of over-fitting [12]. Feature selection is also useful to interpret complex data and to reduce memory cost and running time. To achieve primal sparsity, in likelihood-based estimation, a commonly used strategy is to add an ℓ_1-penalty to the likelihood function, which can be viewed as a maximum a posteriori (MAP) estimation under a Laplace prior. The sparsity of the ℓ_1-norm regularized maximum (conditional) likelihood estimation is due to a hard threshold introduced by the Laplace prior, and weights less than the threshold will be set to exact zeros [14].

Another type of sparsity is the *dual sparsity* as enjoyed by large margin models, like the unstructured (i.e., the output is univariate) SVM and the structured M^{3}N [20] or structural SVM [23, 13]. Dual sparsity refers to a phenomenon that only a few lagrange multipliers in the dual form of the original model are non-zero. A dual sparse model has a robust decision boundary which depends only on a few support vectors. The dual sparsity also provides a theoretical motivation of the cutting-plane algorithms [23, 13] and the bundle methods [21], which generally explore the fact that in max-margin Markov models only a few (e.g., polynomial) number of constraints are necessary to achieve a sufficiently accurate solution. Although both primal and dual sparsity can benefit structured prediction models, unfortunately, they do not co-exist in a single existing structured prediction model. For example, the M^{3}N is only dual sparse, while the ℓ_1-norm regularized CRF [2] is only primal sparse.

In this paper, we introduce the ℓ_1-norm regularized max-margin Markov networks (ℓ_1-M^{3}N), which enjoy the primal and dual sparsity simultaneously. The ℓ_1-M^{3}N is a generalization of the unstructured ℓ_1-norm SVM [4, 26] to the much broader structured prediction. The primal sparsity of

KDD'09, June 28–July 1, 2009, Paris, France.

the ℓ_1-M^{3}N makes it able to select significant features. To learn an ℓ_1-M^{3}N, we present three methods: (1) projected sub-gradient; (2) cutting-plane; and (3) a rather novel EM-style algorithm based on an equivalence between ℓ_1-M^{3}N and a novel *adaptive* M^{3}N. The EM-algorithm is similar to the variational learning method of the Laplace max-margin Markov network (LapM^{3}N) [27] but essentially differs in updating scaling coefficients. Because of a smooth shrinkage effect, LapM^{3}N is *pseudo-primal sparse* (i.e., only a few input features have *large* weights) and does not select significant features. Finally, we perform extensive studies on both synthetic and real data sets. Results show that ℓ_1-M^{3}N can effectively select significant features and can perform as well as the closely related pseudo-primal sparse LapM^{3}N, while consistently outperforms competing models that enjoy either dual or primal sparsity; and the EM-algorithm is more robust than the other two methods.

This paper is structured as follows. The next section presents the preliminaries of max-margin Markov networks. Section 3 presents the primal sparse ℓ_1-M^{3}N. Section 4 presents the three learning algorithms for the ℓ_1-M^{3}N. Section 5 presents our empirical studies, and Section 6 concludes this paper.

2. PRELIMINARIES

In structured prediction, our goal is to learn a predictive function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from a structured input $\mathbf{x} \in \mathcal{X}$ to a structured output $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_l$ represents a space of multivariate and structured outputs. For example, in part-of-speech (POS) tagging, each input $\mathbf{x}$ is a word sequence, $\mathcal{Y}_i$ consists of all the POS tags and each output (label) $\mathbf{y} = (y_1, \cdots, y_l)$ is a sequence of POS tags. We assume a finite feasible set of labels $\mathcal{Y}(\mathbf{x})$ for any $\mathbf{x}$.

Let $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ be a parametric discriminant function. In this paper, we concern ourselves with the special case of a linear model, where F is defined by a set of K feature functions $f_k : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ and their weights w_k: $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$. The generalization to the non-linear case can be done similarly as the feature selection in an SVM [11].

By using different loss functions, the parameters $\mathbf{w}$ can be estimated by maximizing the conditional likelihood [16] or by maximizing the margin [1, 20, 23]. We focus on the max-margin models and will provide empirical comparison with likelihood-based methods.

2.1 Max-Margin Markov networks

The max-margin Markov networks (M^{3}N) [20] approach the structured prediction problem by defining a predictive rule as an optimization problem:

$$h_0(\mathbf{x}; \mathbf{w}) = \arg \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} F(\mathbf{x}, \mathbf{y}; \mathbf{w}), \quad (1)$$

where the model parameter $\mathbf{w}$ is estimated by solving a constrained optimization problem, with a set of fully labeled training data $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$:

$$\begin{aligned} \text{P0 (M}^3\text{N)}: &\quad \min_{\mathbf{w},\xi} \ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \xi_i \\ \text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i): &\quad \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i, \ \xi_i \geq 0 \ , \end{aligned}$$

where ξ_i represents a slack variable absorbing errors in training data, C is a positive constant, and $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \mathbf{f}(\mathbf{x}^i, \mathbf{y})$. $\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$ is the "margin" favored by the true label $\mathbf{y}^i$ over a prediction $\mathbf{y}$, and $\Delta\ell_i(\mathbf{y})$ is a loss penalized on $\mathbf{y}$ compared with $\mathbf{y}^i$, e.g., *hamming loss* [20]: $\Delta\ell_i(\mathbf{y}) = \sum_{j=1}^l \mathbb{I}(y_j \neq y_j^i)$, where $\mathbb{I}(\cdot)$ is an indicator function that equals to one if the argument is true and zero otherwise.

The problem P0 can be efficiently solved or approximately solved with a cutting-plane [23], message-passing [20], or gradient-decent [3, 19] method, which generally explores the sparse dependencies among individual labels in $\mathbf{y}$, as reflected in the specific design of the feature functions (e.g., based on pair-wise labeling potentials). As described shortly, these algorithms can be directly employed as subroutines in solving our proposed model. The prediction problem (1) can be efficiently solved too by exploring these sparse dependencies in $\mathbf{y}$, e.g., using the Viterbi algorithm when the graphical model is a linear chain [16].

3. PRIMAL SPARSE MAX-MARGIN MARKOV NETWORKS

In this section, we introduce a primal sparse max-margin Markov network. We begin with a brief overview of three types of sparsity.

3.1 Sparsity

Primal Sparsity: We say a model enjoys the primal sparsity, if only a few features in the original model have *non-zero* weights. The term "primal" stems from a convention in the optimization literature, which generally refers to (constrained) problems pertaining to the original model. For example, P0 is the primal form of the max-margin Markov networks. By primal sparsity, we mean that only a few elements of $\mathbf{w}$ are non-zero.

As we have stated, primal sparsity is important for selecting significant features and reducing the risk of overfitting. To achieve the primal sparsity, ℓ_1-regularization has been extensively studied in different learning paradigms. For likelihood-based estimation, recent work includes the structure learning of graphical models [17, 24]. For max-margin learning, the unstructured 1-norm SVM [26, 29] has been proposed. Our work represents a generalization of the 1-norm SVM to structured learning, as we shall see.

Dual Sparsity: Dual sparsity refers to a phenomenon that only a few lagrange multipliers in the dual form of the original model turn out to be *non-zero*. Dual sparsity is an intrinsic property as enjoyed by max-margin models. For instance, for the max-margin Markov networks, the Lagrangian of P0 is as follows:

$$\begin{aligned} L(\mathbf{w}, \xi, \alpha, v) = &\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \xi_i \\ &- \sum_{i,\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} \alpha_i(\mathbf{y})[\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}) + \xi_i] - v^\top \xi, \end{aligned}$$

where $\alpha_i(\mathbf{y})$, $\forall i$, $\forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)$ are nonnegative lagrange multipliers, one for each of the margin constraints in P0, and v_i are the lagrange multipliers for the constraints $\xi_i \geq 0$, $\forall i$.

Since P0 is a convex program and satisfies the Slater's condition [6], the saddle point of the Lagrangian L is the KKT point of P0. From the stationary condition, we get the optimum solution of P0:

$$\mathbf{w} = \sum_{i,\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} \alpha_i(\mathbf{y}) \Delta\mathbf{f}_i(\mathbf{y}).$$

From the Complementary Slackness condition, we get:

$$\forall i, \ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i): \quad \alpha_i(\mathbf{y})[\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}) + \xi_i] = 0,$$

Thus, for those constraints that are inactive, i.e., the equality does not hold, the corresponding lagrange multipliers

$\alpha_i(\mathbf{y})$ will be zero, and the optimum solution $\mathbf{w}$ does not depend on these inactive constraints.

When a model is dual sparse, its decision boundary depends on a few number of support vectors, which in principle leads to a robust decision boundary. Moreover, as we have stated, the dual sparsity provides a theoretical motivation of the cutting-plane [23, 13] and bundle [21] methods.

Pseudo-primal Sparsity: Besides the primal and dual sparsity, there is another type of regularization which yields *pseudo*-primal sparse estimates. We say a model is pseudo-primal sparse, if only a few elements of $\mathbf{w}$ have *large* values. In other words, $\mathbf{w}$ can have many non-zero small elements. Thus, a pseudo-primal sparse model does not explicitly discard features by setting their weights to exactly zeros. The recently proposed Laplace max-margin Markov network (LapM^{3}N) [27] is pseudo-primal sparse because of a smooth shrinkage effect.

Unfortunately, although both the primal and dual sparsity can benefit structured prediction models, they usually do not co-exist. For example, the powerful M^{3}N is not primal sparse, because it employs an ℓ_2-norm penalty that cannot automatically select significant features. Below, we present a primal sparse max-margin Markov network to achieve both primal and dual sparsity in one single model.

3.2 Primal Sparse M^{3}N

To introduce the primal sparsity in max-margin Markov networks, we propose to use the ℓ_1-norm instead of the ℓ_2-norm of model parameters. Therefore, we formulate the ℓ_1-M^{3}N as follows, which is a generalization of the unstructured 1-norm SVM [26, 4] to the structured learning setting:

$$\text{P1 } (\ell_1\text{-M}^3\text{N}): \quad \min_{\mathbf{w},\xi} \ \frac{1}{2}\|\mathbf{w}\|_1 + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t. } \forall i,\ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i): \quad \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,$$

where $\|.\|_1$ is the ℓ_1-norm.

Like P0, the problem P1 is a convex program and satisfies the Slater's condition. Therefore, due to KKT conditions, the ℓ_1-M^{3}N enjoys the dual sparsity. The difference between the ℓ_1-M^{3}N and the standard M^{3}N lies in the regularizer they use. Compared to the ℓ_2-norm in M^{3}N, which is differentiable everywhere, the ℓ_1-norm in the ℓ_1-M^{3}N is not differentiable at the origin. This singularity property will ensure that the ℓ_1-M^{3}N is able to remove many noise features by estimating their weights to exactly zeros. However, the differentiability of ℓ_2-norm makes the M^{3}N have all the input features for prediction. When the feature space is high dimensional and has many noise features, the M^{3}N will suffer a poor generalization ability caused by these noise features. Thus, the ℓ_1-M^{3}N would be a better choice when the underlying true model is sparse. More insights about the primal sparsity of the ℓ_1-M^{3}N will be presented in the next section, along with the algorithm development.

4. THREE LEARNING ALGORITHMS

In this section, we introduce three algorithms for learning an ℓ_1-M^{3}N and will empirically compare them.

4.1 Cutting-plane Method

The first method we propose is a cutting-plane method [15], which has been used to learn an M^{3}N [23, 13].

Basically, the cutting-plane method aims to find a small sub-set of the constraints in the problem P1 to get a sufficiently accurate solution. To construct such a sub-set, the algorithm takes an iterative procedure to select informative (e.g., the mostly violated) constraints under the current model. This procedure will terminate when no constraints are selected or the solution is accurate enough. As we have stated, due to the dual sparsity of the max-margin models, only a few constraints are active for the optimum solution. Therefore, the cutting-plane method is a valid strategy. For example, for the M^{3}N, as shown in [23], a polynomial number of constraints are sufficient to get a good solution.

Algorithm 1 Cutting-plane Method

```
Input: data $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$, constants $C$ and $\epsilon$
Output: $\mathbf{w}$
Initialize $S_i \leftarrow \emptyset, \forall 1 \leq i \leq N$.
repeat
  for $i = 1$ to $N$ do
    Compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}^i)} H_i(\mathbf{y};\mathbf{w})$.
    Compute $\xi_i = \max\{0, \max_{\mathbf{y}\in S_i} H_i(\mathbf{y};\mathbf{w})\}$.
    if $H_i(\hat{\mathbf{y}};\mathbf{w}) > \xi_i + \epsilon$ then
      $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$.
      $\mathbf{w} \leftarrow$ optimize the LP problem over $S = \cup_i S_i$.
    end if
  end for
until no change in $S$.
```

Let $H_i(\mathbf{y};\mathbf{w}) = \Delta\ell_i(\mathbf{y}) - \mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y})$. The cutting-plane algorithm is outlined in Algorithm 1, where ϵ is a precision parameter. Specifically, the algorithm maintains working sets S_i for each training data to record the selected constraints. The constraints in the sub-set $S = \cup_{i=1}^N S_i$ defines a relaxation of the problem P1. The algorithm proceeds to find the most violated constraint, i.e., the constraint associated with $\hat{\mathbf{y}}$, for each data $\mathbf{x}^i$. If the margin violation of this constraint exceeds the current ξ_i by more than ϵ, the constraint is added to the working set S_i. Once a new constraint has been added, the algorithm re-computes the solution by solving an LP (linear programming) problem over the working set S. The LP formulation of the problem P1 over the working set $S = \cup_{i=1}^N S_i$ is as follows:

Let $\mathbf{w} = \mu - v$, where $\mu_k, v_k \geq 0,\ \forall k$. For each component k, we can assume that at least one of μ_k and v_k is zero; otherwise, we can minus each of them by the smaller one, without changing w_k. Therefore, $\|\mathbf{w}\|_1 = \mu + v$, and the LP formulation of the problem P1 is as follows:

$$\min_{\mu,v,\xi} \ \frac{1}{2}(\mu + v) + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t. } \forall i,\ \forall \mathbf{y} \in S_i: \quad (\mu - v)^\top\Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,$$
$$\forall k: \quad \mu_k, v_k \geq 0,$$

This LP problem can be solved with a standard solver, such as MOSEK. In Alg. 1, finding $\hat{\mathbf{y}}$ is a loss-augmented prediction problem: $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}^i)} \mathbf{w}^\top\mathbf{f}(\mathbf{x}^i, \mathbf{y}) + \Delta\ell_i(\mathbf{y})$, which can be efficiently done as the prediction problem (1).

Since the working set S is increased during the iteration, the relaxation of the problem P1 gets tighter and the solution gets more accurate as the algorithm proceeds.

4.2 Projected Sub-gradient

The second learning algorithm we present is a projected sub-gradient method based on an equivalent reformulation of the problem P1.

4.2.1 Reformulation

In the problem P1, each ξ_i is associated with a set of constraints: $\forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i) : \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i$. This set of constraints can be equivalently written as one constraint:

$$\xi_i \geq \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} (\Delta \ell_i(\mathbf{y}) - \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y})).$$

Since the above constraint ensures that $\xi_i \geq 0$, the problem P1 can be equivalently written as:

$$\min_{\mathbf{w}, \xi} \ \frac{1}{2} \|\mathbf{w}\|_1 + C \sum_{i=1}^N \xi_i$$

$$\text{s.t. } \forall i : \quad \xi_i \geq \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} (\Delta \ell_i(\mathbf{y}) - \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y})),$$

where the equality of the constraints holds when the convex program gets the optimum; otherwise, we can find a new ξ that yields a smaller objective value while satisfying all the constraints. Putting all the above arguments together, we get the following equivalent formulation of the problem P1:

$$\min_{\mathbf{w}} \ \frac{1}{2} \|\mathbf{w}\|_1 + NC \mathcal{R}_{hinge}(\mathbf{w}) \qquad (2)$$

where $\mathcal{R}_{hinge}(\mathbf{w}) \triangleq \frac{1}{N} \sum_i \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} [\Delta \ell_i(\mathbf{y}) - \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y})]$ is the structured hinge loss. It is piecewise linear and convex.

The last step is to show that the formulation (2) is equivalent to the following constrained optimization problem:

$$\text{P1}' : \quad \min_{\mathbf{w}} \ \mathcal{R}_{hinge}(\mathbf{w}), \ \text{ s.t. : } \ \|\mathbf{w}\|_1 \leq \lambda$$

This is because for the optimum solution $\mathbf{w}^\star$ of the problem (2) with a specific C, we can set $\lambda = \|\mathbf{w}^\star\|_1$ in P1$'$. Then, $\mathbf{w}^\star$ is the optimum solution of P1$'$, otherwise, we can find a new $\mathbf{w}'$ that achieves a smaller objective value in the problem (2). Conversely, we can inverse the mapping to find those values of C in (2) that give rise to the same solution as P1$'$. The problem P0 has a similar formulation as the P1$'$, but with the constraint replaced by $\|\mathbf{w}\|_2^2 \leq \lambda$.

4.2.2 Projected Sub-gradient Algorithm

To solve the constrained convex optimization problem P1$'$, probably the most widely used algorithm is the projected sub-gradient method for convex optimization [5].

Let $\mathcal{W}$ denote the convex set of $\mathbf{w}$ defined by the constraint $\|\mathbf{w}\|_1 \leq \lambda$, and let $\bigtriangledown$ denote a sub-gradient of the structured hinge loss $\mathcal{R}_{hinge}(\mathbf{w})$, then projected sub-gradient methods minimize the hinge loss $\mathcal{R}_{hinge}(\mathbf{w})$ by generating the sequence $\{\mathbf{w}^{(t)}\}$ via:

$$\mathbf{w}^{(t+1)} = \Pi_{\mathcal{W}}\big(\mathbf{w}^{(t)} - \eta_t \bigtriangledown^{(t)}\big), \qquad (3)$$

where $\bigtriangledown^{(t)}$ is *any* sub-gradient evaluated at $\mathbf{w}^{(t)}$, η_t is a learning rate, and $\Pi_{\mathcal{W}}(\mathbf{w}) = \arg\min_{\mu \in \mathcal{W}} \|\mu - \mathbf{w}\|_1$.

Here, by sub-gradient, we mean a vector defined as follows:

Definition 1 (Sub-gradient). *Let $h : \mathcal{W} \rightarrow \mathbb{R}$ be a convex function. A sub-gradient at $w \in \mathcal{W}$ is a vector $g \in \mathbb{R}^K$ such that $\forall w' \in \mathcal{W}$, $h(w') \geq h(w) + g^\top (w' - w)$.*

Algorithm 2 Projected Sub-gradient Method

```
Input: data $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$, constants $\lambda$, iteration number $T$
Output: $\mathbf{w}^T$
for $t = 1$ to $T - 1$ do
  Compute $g(\mathbf{w}^{(t)}) = \frac{1}{N} \sum_{i=1}^N g_i(\mathbf{w}^{(t)})$.
  Update $\mathbf{w}^{(t+1)} = \Pi_{\mathcal{W}}\big(\mathbf{w}^{(t)} - \eta_t g(\mathbf{w}^{(t)})\big)$.
end for
```

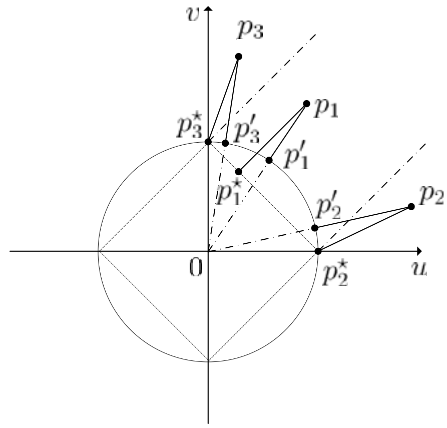

Figure 1: Projection of a point to the ℓ_2-ball (solid); and ℓ_1-ball (dashed) in the two-dimensional space.

Finding one sub-gradient of the hinge loss $\mathcal{R}_{hinge}(\mathbf{w})$ is easy. Specifically, for each component: $\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} [\Delta \ell_i(\mathbf{y}) - \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y})]$, one sub-gradient[1] is: $g_i(\mathbf{w}) = \mathbf{f}(\mathbf{x}^i, \mathbf{y}^\star) - \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i)$, where $\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^i, \mathbf{y}) + \Delta \ell_i(\mathbf{y})$, which is the loss-augmented prediction under the current model and can be efficiently done as we stated in Section 4.1. Therefore, one sub-gradient of $\mathcal{R}_{hinge}(\mathbf{w})$ is: $g(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N g_i(\mathbf{w})$, and can be efficiently computed.

With the sub-gradients computed, we can develop a batch or an online algorithm to learn the ℓ_1-M^{3}N, like the sub-gradient methods [19] for M^{3}N. Algorithm 2 outlines the batch version, where the projection to an ℓ_1-ball can be performed with the efficient projection algorithms in [8].

From the perspective of projection, we can also see the difference between the M^{3}N and ℓ_1-M^{3}N. As illustrated in Figure 1, the first orthant (the other orthants are similar) in a two-dimensional space is partitioned into three regions by the dashed lines. For the points inside the balls (either ℓ_1-ball or ℓ_2-ball), no projection is needed. Thus, we only consider the points outside of the balls. For the point p_1, which is far from both axes, the projection to the ℓ_2-ball is the point p_1', and the projection to the ℓ_1-ball is $p_1^\star$. Both p_1' and $p_1^\star$ do not have a zero coordinate. For the point p_2, whose second coordinate is small (i.e., close to the axis u), the projection to the ℓ_2-ball is p_2', while the projection to the ℓ_1-ball is $p_2^\star$ whose second coordinate is zero. Similarly, the projection of the point p_3 to the ℓ_1-ball is $p_3^\star$, whose first coordinate is zero, while the projection to the ℓ_2-ball (i.e., the point p_3') does not have a zero coordinate. In general, for the ℓ_1-M^{3}N, the projection of a small weight (i.e., close to one axis) to an ℓ_1-ball tends to be zero. In contrast, the projection of a non-zero weight to an ℓ_2-ball is always non-zero. Thus, the existing ℓ_2-norm M^{3}N does not explicitly set small weights to zeros and cannot select significant features, while the ℓ_1-M^{3}N can select significant features.

4.3 EM-Style Algorithm

The third rather novel method we present to learn the ℓ_1-M^{3}N is an EM-style algorithm, which is based on an equivalence between the ℓ_1-M^{3}N and an *adaptive* M^{3}N.

[1] This is because $\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} [\Delta \ell_i(\mathbf{y}) - \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y})] = \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^i)} [\Delta \ell_i(\mathbf{y}) + \mathbf{w}^\top \mathbf{f}(\mathbf{x}^i, \mathbf{y})] - \mathbf{w}^\top \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i)$, where the last term is a constant. This is a piecewise linear maximization problem, whose sub-gradient is determined by the component that achieves the maximum value [7].

4.3.1 Equivalence Theorem

In [10], an equivalence between the adaptive regression and LASSO is presented. Here, we extend the result to the max-margin Markov networks. Basically, we show that the ℓ_1-M^{3}N is equivalent to an *adaptive* Max-Margin Markov Network (AdapM^{3}N), defined as follows:

$$\begin{aligned} \text{P2 (AdapM}^3\text{N)}: \quad & \min_{\mathbf{w},\tau,\xi} \ \mathbf{w}^\top \Sigma^{-1} \mathbf{w} + C \sum_{i=1}^{N} \xi_i, \\ \text{s.t. } \forall i, \ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i): \quad & \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i; \xi_i \geq 0 \\ \forall k: \quad & \frac{1}{K} \sum_{k=1}^{K} \tau_k = \frac{1}{\lambda}; \tau_k \geq 0. \end{aligned}$$

where $\Sigma = diag(\tau)$. Here, by adaptivity, we mean that the values of the scaling factors τ are automatically adapted during the estimation process.

The rationale behind the adaptive M^{3}N is that: by adaptively penalizing different components, the coefficients of irrelevant features can be shrunk to zero, i.e., the corresponding τ_k go to zero. Because of the constraint $\frac{1}{K}\sum_{k=1}^{K}\tau_k = \frac{1}{\lambda}$, each estimate is a balance among τ_k. The idea of using adaptive regularization to achieve sparse estimates has been extensively studied in Automatic Relevance Determination (ARD) [18] and sparse Bayesian learning [22].

Now, we show that the adaptive M^{3}N produces the same estimate as the ℓ_1-M^{3}N and thus gives sparse estimates.

Theorem 2. *The AdapM^{3}N yields the same estimate as the ℓ_1-M^{3}N.*
Proof: See Appendix A. □

4.3.2 An EM-style Algorithm

Based on Theorem 2, we can develop an EM-style algorithm to approximately learn an ℓ_1-M^{3}N by solving the problem P2. Specifically, the algorithm iteratively solves the following two steps until a fixed point (i.e., a local optimum) is obtained:

Step 1: keep τ fixed, optimize P2 over $(\mathbf{w}, \xi)$:

$$\begin{aligned} & \min_{\mathbf{w},\xi} \ \mathbf{w}^\top \Sigma^{-1} \mathbf{w} + C \sum_{i=1}^{N} \xi_i \\ \text{s.t. } \forall i, \ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i): \quad & \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i; \ \xi_i \geq 0, \end{aligned}$$

This is an M^{3}N problem and can be efficiently solved with the sub-gradient [19] or exponentiated gradient [3] methods.

Step 2: keep $(\mathbf{w}, \xi)$ fixed, optimize P2 over τ. The problem reduces to:

$$\min_{\tau} \ \mathbf{w}^\top \Sigma^{-1} \mathbf{w}, \ \text{ s.t.}: \frac{1}{K} \sum_{k=1}^{K} \tau_k = \frac{1}{\lambda}, \ \tau_k \geq 0, \ \forall k.$$

By forming a Lagrangian and doing some algebra, it is easy to show that the solution is:

$$\forall k: \quad \tau_k = \frac{K|w_k|}{\lambda \sum_k |w_k|} \qquad (4)$$

In practice, to avoid divergent results (i.e., zero diagonal entries in Σ), re-parametrization of the problem P2 can be performed. Specifically, we define new variables:

$$\forall k: \quad \gamma_k = \sqrt{\frac{1}{\lambda \tau_k}} w_k, \ \text{ and } \ \beta_k = \sqrt{\lambda \tau_k}.$$

Then, the EM-style algorithm alternatively updates (γ, ξ) and β, as outlined in the Algorithm 3 and detailed below:

Update (γ, ξ): solve the problem:

$$\begin{aligned} & \min_{\gamma,\xi} \ \lambda \gamma^\top \gamma + C \sum_{i=1}^{N} \xi_i, \qquad (5) \\ \text{s.t. } \forall i, \ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i): \quad & \gamma^\top \text{diag}(\beta) \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i; \ \xi_i \geq 0 \end{aligned}$$

Algorithm 3 EM-Style Algorithm

```
Input: data $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$, constants $\lambda$ and $C$, iteration number $T$
Output: $\mathbf{w}^T$
Initialize $\beta_k \leftarrow \sqrt{\lambda}$ (i.e., $\tau_k \leftarrow 1$), $\forall 1 \leq k \leq K$.
for $t = 1$ to $T - 1$ do
    Compute $(\gamma, \xi)$ by solving the QP problem (5).
    Update $\beta$ using Eq. (6).
end for
```

Again, this is an M^{3}N problem and can be efficiently solved with the methods [19, 3, 13].

Update β:

$$\forall k: \quad \beta_k = \frac{\sqrt{K}|\gamma_k|}{\sqrt{\sum_{j=1}^{K} \gamma_j^2}}. \qquad (6)$$

More details about the derivation are in the proof of Theorem 2. Note that the EM-algorithm is similar to the variational learning method of the LapM^{3}N [27], which also iterates between solving an M^{3}N problem and updating adaptive coefficients. The essential difference lies in the second step. When $\tau_k = 0$ in Eq. (4), or $\beta_k = 0$ in Eq. (6), the feature k will be discarded in the final sparse estimate. But, the update rule in the LapM^{3}N ensures that adaptive coefficients are always positive and finite. Therefore, the LapM^{3}N does not explicitly discard features. This observation (approximately) explains why the ℓ_1-M^{3}N is primal sparse, while the LapM^{3}N is pseudo-primal sparse. See [28] for more theoretical analysis. In this algorithm, we usually keep C fixed, while tuning the parameter λ.

5. EXPERIMENTS

This section presents some empirical evaluation of the ℓ_1-M^{3}N on both synthetic and real data sets. We compare it with the un-regularized conditional random fields (CRFs) [16], the ℓ_2-norm regularized CRFs (ℓ_2-CRF), the primal sparse ℓ_1-norm regularized CRFs (ℓ_1-CRF), the dual sparse M^{3}N, and the pseudo-primal sparse LapM^{3}N. We use the method [2] to learn an ℓ_1-CRF and [19] to learn an M^{3}N.

5.1 Evaluation on Synthetic Data Sets

We follow the method as described in [27] to do the experiments. We generate sequence data sets, i.e., each input $\mathbf{x}$ is a sequence $(x_1, \cdots, x_L)$, and each component x_l is a d-dimensional vector of input features. The synthetic data are generated from pre-specified CRF models with either i.i.d. instantiations of the input features or correlated instantiations of the input features, from which samples of the structured output $\mathbf{y}$, i.e., a sequence $(y_1, \cdots, y_L)$, can be drawn from the conditional distribution $p(\mathbf{y}|\mathbf{x})$ defined by the CRF based on a Gibbs sampler.

Due to space limitation, we only report the results on the data sets with correlated input features. We get the same conclusions in the i.i.d case. Specifically, we set $d = 100$ and 30 input features are relevant to the output. The 30 relevant features are partitioned into 10 groups. For the features in each group, we first draw a real-value from a standard normal distribution and then corrupt the feature with a random Gaussian noise (zero mean and standard variance 0.05) to get 3 correlated features. Then, we generate 10 linear-chain CRFs with 8 binary states (i.e., $L = 8$ and $\mathcal{Y}_l = \{0, 1\}$). The feature functions include: 200 real valued state-feature functions, of which each is over a one-dimensional input feature

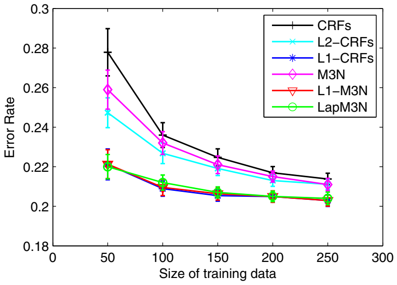

Figure 2: Error rates of different models.

and a class label; and 4 (2×2) transition feature functions capturing pairwise label dependencies. Each CRF generates a data set that contains 1000 instances.

We do K-fold cross-validation on each data set and take the average (both accuracy and the variance) over the 10 data sets as the final results. In each run we choose one part to do training and test on the rest $K-1$ parts. K is changed from 20, 10, 7, 5, to 4. In other words, we use 50, 100, about 150, 200, and 250 samples during the training. Figure 2 shows the average performance. We can see that the primal sparse models (i.e., ℓ_1-M^{3}N and ℓ_1-CRFs) outperform the M^{3}N, which is only dual sparse. Because of a smooth shrinkage effect [27], the LapM^{3}N can shrink the weights of irrelevant features to be extremely small by choosing an appropriate regularization constant. In fact, the ℓ_1-M^{3}N is an extreme case of the LapM^{3}N when the LapM^{3}N's regularization constant goes to infinity [28]. Thus, the LapM^{3}N performs similarly to the primal-sparse models. Obviously, the un-regularized CRFs perform much worse than the other models which use regularization on these sparse data sets. The ℓ_2-CRFs perform comparably with the M^{3}N, which also uses the ℓ_2-norm regularizer. This is reasonable because as noted in [9], the ℓ_2-norm regularized maximum likelihood estimation of CRFs has a similar convex dual as that of the M^{3}N, and the only difference is the loss they try to optimize, namely, ℓ_2-CRFs minimize the log-loss while M^{3}N minimizes the structured hinge loss.

Figure 3 shows the average weights of different models doing 10-fold CV on the first data set and the weights of the CRF model (first plot) that generates this data set. For CRFs, ℓ_2-CRFs, LapM^{3}N and M^{3}N, all the weights are nonzero, although the weights of LapM^{3}N are generally much smaller because of the shrinkage effect [27]. For ℓ_1-M^{3}N and ℓ_1-CRFs, the estimates are sparse. Both of them can discard all the noise features when choosing an appropriate regularization constant. As shown in [27], ℓ_1-CRFs are sensitive to the regularization constant. As we shall see the ℓ_1-M^{3}N with the EM-style algorithm is very robust. Note that all the models have quite different average weights from the model that generates the data. This is because we use a stochastic procedure (i.e., Gibbs sampler) to assign labels to the generated data samples. In fact, if we use the CRF model to predict on its generated data, the error rate is about 0.5. Thus, the learned models, which get higher accuracy, are different from the model that generates the data.

5.2 OCR Data Sets

The OCR data set is partitioned into 10 subsets for 10-fold CV as in [20, 19]. We randomly select N samples from each fold and put them together to do 10-fold CV. We vary N from 100, 150, 200, to 250, and denote the selected data sets by OCR100, OCR150, OCR200, and OCR250, respectively.

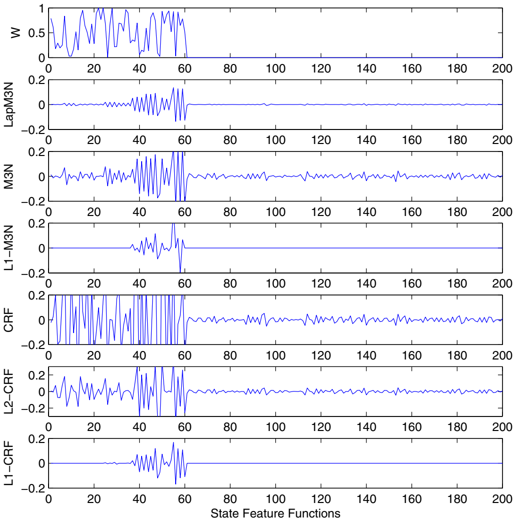

Figure 3: The average weights of different models.

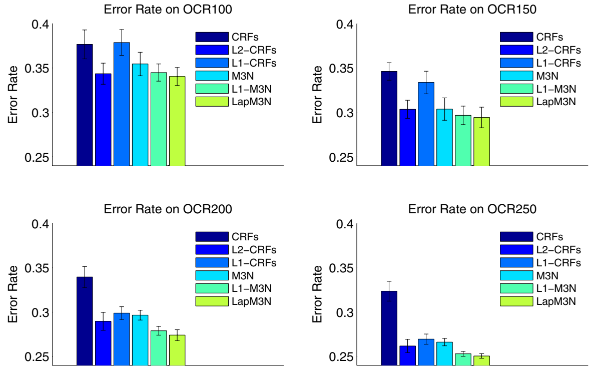

Figure 4: Evaluation results on OCR data sets with different numbers of selected data.

On these data sets and the web data as in Section 5.4, our implementation of the cutting-plane method for ℓ_1-M^{3}N is extremely slow. The warm-start simplex method of MOSEK does not help either. For example, if we stop the algorithm with 600 iterations on OCR100, then it will take about 20 hours to finish the 10 fold CV. Even with more than 5 thousands of constraints in each training, the performance is still bad (the error rate is about 0.45). The projected sub-gradient and the EM-style algorithm both are efficient. The EM-algorithm yields slightly better performance than the projected sub-gradient, as we shall see.

The results are shown in Figure 4, which are achieved by the EM-style algorithm. We can see that as the number of training data increases, all the models get lower error rates and smaller variances. Generally, the ℓ_1-M^{3}N performs comparably with the LapM^{3}N, while consistently outperforms all the other models. M^{3}N outperforms the standard, un-regularized, CRFs and the ℓ_1-CRFs. Again, ℓ_2-CRFs perform comparably with M^{3}N. This is reasonable due to the understanding of their only difference on the loss functions [9] as we have stated.

5.3 Comparison of the Algorithms

We have presented three algorithms to learn an ℓ_1-M^{3}N, including the projected sub-gradient, cutting-plane, and the EM-style algorithm. We empirically compare them on the synthetic and OCR data sets. We focus on: (1) effectiveness

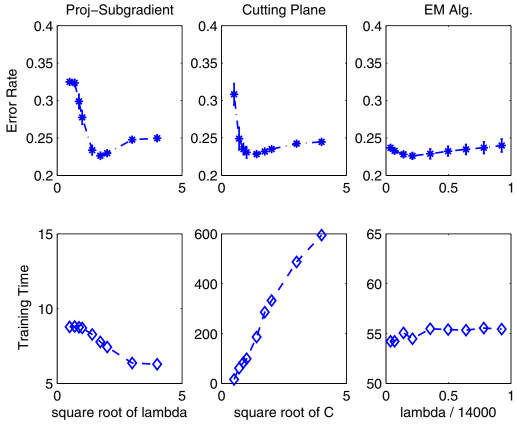

Figure 5: The error rates and training time (CPU-seconds) of different algorithms on the first synthetic data set against the regularization constants.

in recovering sparse patterns; (2) prediction accuracy; and (3) time efficiency. For the EM-algorithm, we keep C fixed.

5.3.1 Recovering Sparse Patterns

Table 1 shows the numbers of non-zero average weights learned by different algorithms with different regularization constants doing 10-fold CV on the first synthetic data set. The rows indicated by "Total" show the numbers of non-zero weights for all the 200 state-feature functions and the rows with "Irrelevant" show the numbers of non-zero weights for the 140 state-feature functions based on the 70 irrelevant input features. In the EM-algorithm, we set τ (or β) to be zero if it is less than 10^{-4}. We can see that the EM-algorithm has similar numbers (in reverse orders because of different effects of their regularization constants) of non-zero weights as the cutting-plane method. However, the sub-gradient method keeps many features, whose weights are small but not exactly zero, and truncating the feature weights with the same threshold as in the EM-algorithm doesn't change the sparse pattern much. Perhaps tuning the learning rate could make this tail of very small features disappear. Note that for different algorithms, the regularization parameters are generally incomparable. In these experiments, we select a set of values around the best one we have tried for each algorithm.

5.3.2 Accuracy and Efficiency

Figure 5 shows the error rates and training time of the three algorithms for the ℓ_1-M^{3}N doing 10-fold CV on the first synthetic data set. We can see that both the sub-gradient and cutting-plane methods are sensitive to their regularization constants (see exact values in Table 1). For the sub-gradient method, the time depends largely on the ℓ_1-ball's radius. For larger balls, the projection will be easier. Consider the extreme case that the ℓ_1-ball is big enough to contain the model weights (a point in the space $\mathbb{R}^K$), then the projection is not needed. So, the training time decreases when λ increases (i.e., the radius of ℓ_1-ball increases). For the cutting-plane method, the time is mainly dependent on the LP solver, e.g., MOSEK as we use. As C gets bigger, the training time increases very fast because more features have non-zero weights (see Table 1). For the EM-algorithm, both the error rate and training time are stable. We use 15 EM-iterations in these experiments and each iteration takes about 3.7 cpu-seconds, smaller than the time of the sub-gradient method. Since the number of features (204 in total) is small, the projection to an ℓ_1-ball can be efficiently done by using the algorithm [8]. Therefore, the overall training time of the projected sub-gradient algorithm is smaller than those of the other two methods. The best performance of all the three algorithms on this data set is comparable.

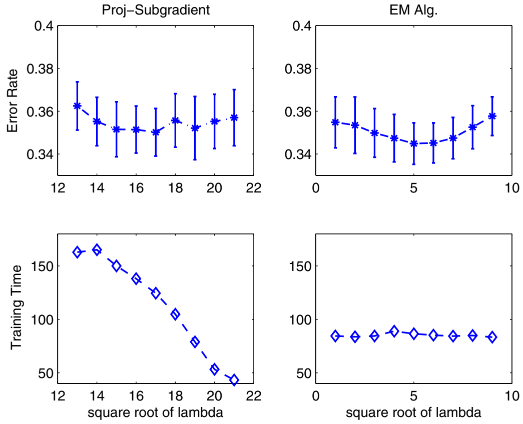

Figure 6: The error rates and training time (CPU-seconds) of different algorithms on the OCR100 data set against the regularization constants.

Figure 6 shows the training time and error rates of the sub-gradient and EM-style algorithms on the OCR100. The cutting-plane method is not scalable on this data set, as we stated in Section 5.2. We can see that both algorithms are robust in the performance on this data set. The best performance of the EM-algorithm is slightly better than that of the sub-gradient method. This may be due to an improper learning rate in the sub-gradient method. For the training time, the sub-gradient method is again sensitive to its regularization constant, while the EM-algorithm is very stable. As we have stated, the decrease in the training time of the sub-gradient method is due to the fact that projection to a large ℓ_1-ball is easier than projection to a smaller ball.

5.4 Web Data Extraction

Web data extraction is a task to identify interested information from web pages, as extensively studied in [25]. Each sample is a data record or an entire web page which is represented as a set of HTML elements. One striking characteristic of web data extraction is that various types of structural dependencies between HTML elements exist, e.g. the HTML tag tree or the Document Object Model (DOM) structure is itself hierarchical. In [25], hierarchical CRFs are shown to have great promise and achieve better performance than flat models like linear-chain CRFs [16]. One method to construct a hierarchical model is to first use a parser to construct a so called vision tree. Then, based on the vision tree, a hierarchical model can be constructed accordingly to extract the interested attributes. See [25] for an example of the vision tree and the corresponding hierarchical model.

In these experiments, we identify four attributes—*Name*, *Image*, *Price*, and *Description* for each product item. We use the data set that is built with web pages generated by 37 different templates [25]. For each template, there are 5 pages for training and 10 for testing. We assume that data records are given, and compare different models on the accuracy of extracting attributes in the given records. There are 1585 data records in the 185 training pages and

Table 1: The number of non-zero average weights by different algorithms on the first synthetic data set.

Proj-Subgradient	$\sqrt{\lambda}$	0.5	0.7	0.87	1	1.41	1.73	2	3	4
	Irrelevant	138	136	140	140	140	140	140	140	140
	Total	198	196	200	200	200	200	200	200	200
Cutting-plane	$\sqrt{C}$	0.5	0.7	0.87	1	1.41	1.73	2	3	4
	Irrelevant	0	0	2	16	103	122	134	139	140
	Total	13	21	26	44	145	169	184	186	189
EM Alg.	$\frac{\lambda}{14000}$	0.036	0.069	0.14	0.21	0.35	0.5	0.64	0.78	0.93
	Irrelevant	140	140	128	108	48	18	2	0	0
	Total	200	198	182	158	90	54	34	32	28

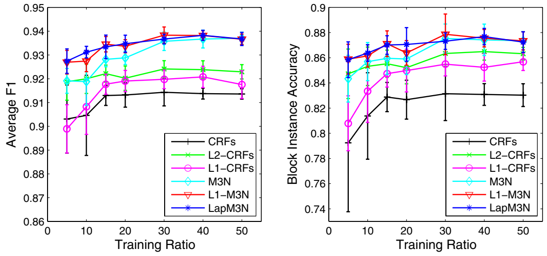

Figure 7: Average F1 and block instance accuracy with different numbers of training data.

3391 data records in the 370 testing pages. We use the two comprehensive evaluation measures, i.e. average F1 and block instance accuracy [25]. Average F1 is the average value of the F1 scores of the four attributes, and block instance accuracy is the percent of data records whose *Name*, *Image*, and *Price* are all correctly identified.

We randomly select $m = 5, 10, 15, 20, 30, 40,$ or, 50 percent of the training records as training data, and test on all the testing records. For each m, 10 independent experiments are conducted and the average performance is summarized in Figure 7. We can see that: first, the models (especially the max-margin models, i.e., M^{3}N, ℓ_1-M^{3}N, and LapM^{3}N) with regularization (i.e., ℓ_1-norm, ℓ_2-norm, or the entropic regularization of LapM^{3}N) can significantly outperform the un-regularized CRFs. Second, the max-margin models generally outperform the conditional likelihood-based models. Third, the primal sparse ℓ_1-M^{3}N performs comparably with the pseudo-primal sparse LapM^{3}N, and outperforms all other models, especially when the number of training data is small. Finally, as in the previous experiments on OCR data, the ℓ_1-M^{3}N generally outperforms the ℓ_1-CRFs, which suggests the potential promise of the max-margin based ℓ_1-M^{3}N.

6. CONCLUSIONS

We have presented the ℓ_1-norm max-margin Markov network (ℓ_1-M^{3}N), which enjoys both the primal and dual sparsity, and introduced three methods to learn an ℓ_1-M^{3}N, including a projected sub-gradient, a cutting-plane and a novel EM-style algorithm that is based on an equivalence between the ℓ_1-M^{3}N and an adaptive M^{3}N. We conducted extensive empirical studies on both synthetic and real world OCR and web data. Our results show that: (1) the ℓ_1-M^{3}N can effectively select significant features; (2) the ℓ_1-M^{3}N can perform as well as the closely related pseudo-sparse LapM^{3}N in prediction accuracy, while consistently outperforms other competing models that enjoy either primal or dual sparsity; and (3) the EM-style algorithm is more robust than the other two in both prediction accuracy and time efficiency.

7. ACKNOWLEDGMENTS

This work was done while J.Z. was a visiting researcher at CMU under a support from NSF DBI-0546594 and DBI-0640543 awarded to E.X.; J.Z. and B.Z. are also supported by Chinese NSF Grant 60621062 and 60605003; National Key Foundation R&D Projects 2003CB317007, 2004CB318108 and 2007CB311003; and Basic Research Foundation of Tsinghua National Lab for Info Sci & Tech.

8. REFERENCES

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *ICML*, 2003.

[2] G. Andrew and J. Gao. Scalable training of ℓ_1-regularized log-linear models. In *ICML*, 2007.

[3] P. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for larg-margin structured classification. In *NIPS*, 2004.

[4] K. Bennett and O. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optim. Methods Softw*, (1):23–34, 1992.

[5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. In *Lecture Notes of EE392o*, Stanford University, Autumn Quarter, 2003-2004.

[8] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projection onto the l_1-ball for learning in high dimensions. In *ICML*, 2008.

[9] A. Globerson, T. Y. Koo, X. Carreras, and M. Collins. Exponentiated gradient algorithms for log-linear structured prediction. In *ICML*, 2007.

[10] Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In *ICANN*, 1998.

[11] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *NIPS*, 2002.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, NY, 2001.

[13] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[14] A. Kaban. On Bayesian classification with laplace priors. *Pattern Recognition Letters*, 28(10):1271–1282, 2007.

[15] J. E. Kelley. The cutting-plane method for solving convex programs. *J. Sco. Indust. Appl. Math.*, (4):703–712.

[16] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[17] S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using ℓ_1-regularization. In *NIPS*, 2006.

[18] R. M. Neal. Bayesian learning of neural networks. In *Leture Notes in Statistics*, 1996.

[19] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. (Online) Subgradient methods for structured prediction. In *AISTATS*, 2007.

[20] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.

[21] C. H. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *SIGKDD*, 2007.

[22] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *JMLR*, (1):211–244, 2001.

[23] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[24] M. J. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *NIPS*, 2006.

[25] J. Zhu, Z. Nie, B. Zhang, and J.-R. Wen. Dynamic hierarchical Markov random fields for integrated web data extraction. *JMLR*, (9):1583–1614, 2008.

[26] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *NIPS*, 2004.

[27] J. Zhu, E. Xing, and B. Zhang. Lapalce maximum margin Markov networks. In *ICML*, 2008.

[28] J. Zhu and E. P. Xing. On primal and dual sparsity of Markov networks. In *ICML*, 2009.

[29] H. Zou. An improved 1-norm svm for simultaneous classification and variable selection. In *AISTATS*, 2007.

APPENDIX

A. PROOF OF THEOREM 2

Proof: Our proof uses some techniques of the proof in [10]. We do re-parametrization by defining $\gamma_k = \sqrt{\frac{1}{\lambda \tau_k}} w_k$, and $\beta_k = \sqrt{\lambda \tau_k}$, $\forall k$. Then, $\mathbf{w} = \mathrm{diag}(\gamma)\beta = \mathrm{diag}(\beta)\gamma \triangleq g(\gamma, \beta)$, and the problem P2 changes to:

$$\begin{aligned} \mathrm{P3}\ :\quad & \min_{\gamma,\beta,\xi}\ \lambda\gamma^\top\gamma + C\sum_{i=1}^N \xi_i, \\ \text{s.t. } \forall i,\ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i) :\quad & g(\gamma,\beta)^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0 \\ \forall k :\quad & \sum_{k=1}^K \beta_k^2 = K; \beta_k \geq 0. \end{aligned}$$

The Lagrangian for P3 is as follows:

$$\begin{aligned} L(\gamma, \beta, \xi, \alpha, \mu, v, \eta) = \lambda\gamma^\top\gamma + C\sum_{i=1}^N \xi_i + \mu^\top\xi + v(\sum_{i=1}^K \beta_k^2 - K) \\ - \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})(g(\gamma,\beta)^\top \Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}) + \xi_i) + \eta^\top\beta. \end{aligned}$$

Taking the derivation of L w.r.t γ and β, we get the normal equations:

$$\begin{cases} \frac{\partial L}{\partial \gamma} = 2\lambda\gamma - \frac{\partial g(\gamma,\beta)}{\partial \gamma}(\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})) \\ \frac{\partial L}{\partial \beta} = 2v\beta - \frac{\partial g(\gamma,\beta)}{\partial \beta}(\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})) + \eta \end{cases} \tag{7}$$

With the definition of $g(\gamma, \beta)$, we can get $\frac{\partial g(\gamma,\beta)}{\partial \gamma} = \mathrm{diag}(\beta)$ and $\frac{\partial g(\gamma,\beta)}{\partial \beta} = \mathrm{diag}(\gamma)$. Let $(\hat\gamma, \hat\beta, \hat\xi)$ be the optimum solution of P3, and let $(\hat\alpha, \hat\mu, \hat v, \hat\eta)$ be the dual optimum. Using the optimality condition that $\frac{\partial L}{\partial \gamma} = 0$ and $\frac{\partial L}{\partial \beta} = 0$ and doing same algebra, we can get:

$$2\lambda\mathrm{diag}(\hat\gamma)\hat\gamma = 2v\mathrm{diag}(\hat\beta)\hat\beta + \mathrm{diag}(\hat\beta)\hat\eta. \tag{8}$$

By the Complementary slackness theorem, $\mathrm{diag}(\hat\beta)\hat\eta = 0$. Thus, we have:

$$\forall k,\ \hat\beta_k^2 = \frac{\lambda}{v}\hat\gamma_k^2. \tag{9}$$

The equality constraint $\sum_k \beta_k^2 = K$ implies that:

$$\forall k,\ \hat\beta_k = \frac{\sqrt{K}|\hat\gamma_k|}{\sqrt{\sum_{j=1}^K \hat\gamma_j^2}}. \tag{10}$$

To get the optimality conditions for the original problem, we let $\hat{\mathbf{w}}$ be the optimum solution of P2. From the definition of γ and β, we get $\forall k,\ |\hat w_k| = \hat\beta_k|\hat\gamma_k|$. Thus,

$$\forall k,\ |\hat w_k| = \frac{\sqrt{K}\hat\gamma_k^2}{\sqrt{\sum_j \hat\gamma_j^2}} \ \Rightarrow\ \frac{|\hat w_k|}{\sum_{j=1}^K |\hat w_j|} = \frac{\hat\gamma_k^2}{\sum_{j=1}^K \hat\gamma_j^2} \tag{11}$$

From the first equation in (7) and using the optimality condition of $\frac{\partial L}{\partial \gamma} = 0$, we can get:

$$\forall k,\ 2\lambda\hat\gamma_k - \hat\beta_k(\sum_{i,\mathbf{y}} \hat\alpha_i(\mathbf{y})\Delta f_i^k(\mathbf{y})) = 0. \tag{12}$$

We consider two cases. First, if $\hat\beta_k = 0$, then we have $\hat w_k = \hat\gamma_k = 0$. Second, if $\hat\beta_k \neq 0$, then we get:

$$\forall k,\ 2\lambda\frac{\hat\gamma_k}{\hat\beta_k} - \sum_{i,\mathbf{y}} \hat\alpha_i(\mathbf{y})\Delta f_i^k(\mathbf{y}) = 0. \tag{13}$$

From Eq. (10) and (11), we get: $\hat\beta_k^2 = \frac{K|\hat w_k|}{\sum_{j=1}^K |\hat w_j|}$. Thus,

$$\forall k,\ \frac{\hat\gamma_k}{\hat\beta_k} = \hat\gamma_k\hat\beta_k\frac{1}{\hat\beta_k^2} = \hat w_k\frac{\sum_{j=1}^K |\hat w_j|}{K|\hat w_k|} = \frac{1}{K}\mathrm{sign}(\hat w_k)\sum_{j=1}^K |\hat w_j|. \tag{14}$$

Therefore, the optimality conditions are:

$$\forall k,\ \begin{cases} -\sum_{i,\mathbf{y}} \hat\alpha_i(\mathbf{y})\Delta f_i^k(\mathbf{y}) + 2\frac{\lambda}{K}\mathrm{sign}(\hat w_k)\sum_{j=1}^K |\hat w_j| = 0 \\ \text{or } \hat w_k = 0, \end{cases}$$

which can easily be shown to have the same form as the optimality conditions of the following problem:

$$\begin{aligned} \mathrm{P4}:\quad & \min_{\mathbf{w},\xi}\ \frac{\lambda}{K}(\sum_{k=1}^K |w_k|)^2 + C\sum_{i=1}^N \xi_i, \\ \text{s.t. } \forall i,\ \forall \mathbf{y} \in \mathcal{Y}(\mathbf{x}^i) :\quad & \mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i; \xi_i \geq 0 \end{aligned}$$

The last step is to show that the optimum lagrange multipliers in P3 and P4 are the same. Eq. (11) implies that: $\sum_{k=1}^K |\hat w_k| = \sqrt{K}\sqrt{\sum_{j=1}^K \hat\gamma_j^2}$. Thus, $\hat\gamma^\top\hat\gamma = \frac{1}{K}(\sum_{k=1}^K |\hat w_k|)^2$. Substituting this result into the Lagrangian L and using the equality constraint $\sum_{i=1}^K \beta_k^2 = K$ and Complementary slackness theorem that $\hat\eta^\top\hat\beta = 0$, we can get:

$$\begin{aligned} L(\hat\gamma, \hat\beta, \hat\xi, \hat\alpha, \hat\mu, \hat v, \hat\eta) &= \frac{\lambda}{K}(\sum_{k=1}^K |\hat w_k|)^2 + C\sum_{i=1}^N \hat\xi_i + \hat\mu^\top\hat\xi \\ &\quad - \sum_{i,\mathbf{y}} \hat\alpha_i(\mathbf{y})(\hat{\mathbf{w}}^\top\Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}) + \hat\xi_i), \end{aligned}$$

which is the Lagrangian of the problem P4 evaluated at the optimal solution. Therefore, the optimum dual variables $\hat\alpha$ are the same for both P3 and P4, and the above optimality conditions are the optimality conditions of P4.

Similar to the re-formulation of the ℓ_1-M^{3}N, the problem P4 can be formulated as the problem:

$$\min_{\mathbf{w}}\ \mathcal{R}_{hinge}(\mathbf{w}),\ \ \text{s.t.}:\ \ \sum_{k=1}^K |w_k| \leq \lambda,$$

which is an ℓ_1-M^{3}N problem as in P1$'$. □