The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Automatic Parameter Tying: A New Approach for Regularized Parameter Learning in Markov Networks

Li Chou,[1] **Pracheta Sahoo,**[1] **Somdeb Sarkhel,**[1,2] **Nicholas Ruozzi,**[1] **Vibhav Gogate**[1]
[1]Department of Computer Science, The University of Texas at Dallas
[2]Adobe Research, San Jose, CA
{lkc130030,pxs167230}@utdallas.edu, sarkhel@adobe.com,
nicholas.ruozzi@utdallas.edu, vgogate@hlt.utdallas.edu

Abstract

Parameter tying is a regularization method in which parameters (weights) of a machine learning model are partitioned into groups by leveraging prior knowledge and all parameters in each group are constrained to take the same value. In this paper, we consider the problem of parameter learning in Markov networks and propose a novel approach called automatic parameter tying (APT) that uses *automatic* instead of *a priori* and *soft* instead of *hard* parameter tying as a regularization method to alleviate overfitting. The key idea behind APT is to set up the learning problem as the task of finding parameters and groupings of parameters such that the likelihood plus a regularization term is maximized. The regularization term penalizes models where parameter values deviate from their group mean parameter value. We propose and use a block coordinate ascent algorithm to solve the optimization task. We analyze the sample complexity of our new learning algorithm and show that it yields optimal parameters with high probability when the groups are *well separated*. Experimentally, we show that our method improves upon L_2 regularization and suggest several pragmatic techniques for good practical performance.

Introduction

Markov networks (MNs) compactly represent a joint probability distribution over a large number of random variables and are widely used in a variety of application domains such as natural language processing and computer vision for representing and reasoning about uncertainty. They are often described using an undirected graph which has one vertex for each variable and each potential function, the latter describe relationships between various random variables. The two key tasks over MNs are (1) learning the structure of the graph and parameters of the potential functions from data; and (2) answering probabilistic queries posed over the learned model (e.g., finding the probability of a variable given an assignment to a subset of variables). Because of the generality, flexibility, and wide applicability of MNs, efficiently solving the two aforementioned tasks is of both practical and theoretical interest in machine learning.

In this paper, we focus on the fundamental problem of learning the parameters of a MN, given its structure, from a fully observed dataset. This problem is typically solved via maximum log-likelihood parameter estimation (MLE). Since calculating the MLE is computationally intractable in many real-world domains, alternative tractable approximations such as maximum pseudo-log-likelihood estimation (MPLE) are generally used. However, both MLE and MPLE are susceptible to overfitting. To combat this issue, L_2 regularization is often employed in practice. L_2 regularization uses an uninformed prior distribution to penalize large parameter values, which helps smooth out the variations in the data. The penalty grows quadratically and thus larger parameter values are penalized more than smaller parameter values. Since the L_2 regularization term is concave, combining it with either MLE or MPLE preserves concavity and the resulting objective can be efficiently optimized using gradient-based methods.

An alternative regularization method is parameter tying, namely partitioning the parameters into groups and forcing all parameters in each group to take the same value. Parameter tying is typically performed *a priori*. For example, in convolutional neural networks (CNNs) (LeCun et al. 1998), parameters are shared (tied) between various neurons to take advantage of symmetries in images and to control the number of parameters. Similarly, in statistical relational learning (SRL) models (Getoor and Taskar 2007) such as Markov logic networks (Domingos and Lowd 2009) and probabilistic soft logic (Bach et al. 2015), weights are tied in order to exploit symmetries in relational domains.

A number of automatic tying schemes have also been investigated. (Nowlan and Hinton 1992) proposed utilizing a Gaussian mixture prior for parameter sharing to simplify neural networks. While Gaussian mixtures are a general model, the trade-offs with this approach are significant, e.g., added complexity resulting from an increase in the number of parameters that need to be selected with validation data, the creation of a large number of local minima, and slow convergence. (Han, Mao, and Dally 2016) investigated compression and pruning in neural networks, utilizing a form of k-means quantization and parameter tying. They incorporate these elements into their pipelined algorithm as a post-processing step and empirically validate the performance. (Liu and Page 2013) utilized k-means to initialize a nonparametric Bayesian (hard) tying approach. Recently, there has been growing interest in *locality-sensitive hashing* (LSH)

where, at a high-level, the objective is to develop hash functions such that the probability of collision for similar items is maximized to solve the *approximate nearest neighbor* (ANN) search problem. Quantization is generally used as a subroutine to partition a lower-dimensional feature space (Wang et al. 2016). Deep learning neural networks have been proposed to learn such hash functions (Zhu et al. 2016).

To the best of our knowledge, there has not been prior work on parameter tying for regularization in MNs. However, (Chou et al. 2016) recently proposed an automatic approach for tying (parameters) in Bayesian networks (BNs) (Pearl 1988; Darwiche 2009). Their parameter learning algorithm has three steps: (1) learn the parameters of the given BN using the MLE objective (this can be done in closed-form in BNs); (2) given a positive integer k, use the 1-dimensional k-means algorithm to group the conditional probabilities into k clusters; and (3) relearn the probabilities by forcing all parameters in each cluster to take the same value (again this can be done in closed form). Through experimental evaluations on a few benchmark datasets, Chou et al. showed that their (hard) parameter tying approach often yields models that not only have higher test set log-likelihood scores but also admit faster and more accurate inference compared to models trained using the MLE objective.

It is not clear how to apply Chou et al.'s method to solve the harder parameter learning task for MNs. Unlike BNs, the choices of (clique) potential functions for MNs are not unique, i.e., MNs are not identifiable. Specifically, there is an (infinte) continuum of parameter value settings that all represent the same probability distribution. For instance, multiplying all parameters in a potential function by a real constant $c > 0$ does not change the underlying distribution. Another example is a MN with two pairwise potentials that share one common variable. The information on the common variable can be split (shifted) in arbitrary ways that result in the same distribution. A consequence of this invariance is that applying the aforementioned k-means clustering technique to achieve parameter tying may not produce useful results.

We address this limitation by proposing a *soft*, instead of hard, parameter tying scheme dubbed APT. Given that a MN has high degree of freedom in terms of parameter settings, soft tying allows for greater flexibility for parameters to shift among cluster assignments. This type of soft tying can be viewed as a generalization of L_2 regularization that allows for k Gaussian priors (instead of one) and k different cluster center means (instead of only zero means). To automate parameter tying, we set up the learning problem as jointly selecting the parameters, group membership, and means such that either the MLE or MPLE plus the penalty term, informally described above, is maximized. We then propose a block coordinate ascent algorithm for this optimization problem and show that it converges to a local maximum.

The second contribution of this paper is a detailed theoretical analysis of our proposed algorithm. In particular, building on the work of (Bradley and Guestrin 2012) and (Ravikumar, Wainwright, and Lafferty 2010), we prove sample complexity bounds for our algorithm within the probably approximately correct framework. We show that we can learn the optimal group memberships of the parameters with high probability when the groups are well separated (i.e., sufficiently far from each other). Moreover, when the data is generated from a model having tied parameters, we show that the sample complexity of our algorithm can be significantly smaller than the MLE/MPLE learning task with L_2 regularization. These results provide the first rigorous theoretical justification of quantization that we are aware of.

We end the paper with a detailed empirical evaluation of our proposed algorithm. We compare with L_2 regularization on binary pairwise Markov network structures generated using the L_1 regularized logistic regression algorithm of (Ravikumar, Wainwright, and Lafferty 2010). Our results clearly show that APT outperforms L_2 regularization in terms of pseudo-log-likelihood (PLL) score, especially on dense networks. We also evaluated the impact of changing the number of groups on the PLL score and found that small to medium values for k often achieve the best score. These results demonstrate that APT is a promising, practical approach for controlling model complexity and improving generalization performance.

Preliminaries and Notation

Let $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ denote a set of random variables and $\boldsymbol{\Phi} = \{(F_1, \theta_1), \ldots, (F_m, \theta_m)\}$ be a set of weighted features where F_i is a feature function defined over a subset of variables, $S(F_i)$, called its scope and θ_i is a real number. Given an assignment of values, denoted by $\bar{\mathbf{x}}$ to all variables in $\boldsymbol{X}$, let $\bar{\mathbf{x}}_{S(F_i)}$ denote the projection of $\bar{\mathbf{x}}$ on $S(F_i)$. We assume that given $\bar{\mathbf{x}}_{S(F_i)}$, F_i takes a value from the set $\{0, 1\}$ where 0 indicates that the feature evaluates to false and 1 indicates that the feature is true.

A log-linear model or a Markov network (MN), denoted by $\mathcal{M}$ is a pair $\langle \boldsymbol{X}, \boldsymbol{\theta} \rangle$ and represents the following probability distribution

$$P_{\boldsymbol{\Phi}}(\bar{\mathbf{x}}) = \frac{1}{Z(\boldsymbol{\Phi})} \exp\left(\sum_i \theta_i \cdot F_i(\bar{\mathbf{x}}_{\mathbf{S}(\mathbf{F_i})}) \right),$$

where $Z(\boldsymbol{\Phi}) = \sum_{\bar{\mathbf{x}}} \exp\left(\sum_i \theta_i \cdot F_i(\bar{\mathbf{x}}_{\mathbf{S}(\mathbf{F_i})}) \right)$ is the normalization constant or the partition function. Note that computing the partition function is #P-hard in the worst case (Roth 1996). Often MNs are described using an undirected graph called the interaction graph or primal graph. This graph has one vertex for each variable $X_i \in \boldsymbol{X}$ and an edge between two vertices if the corresponding variables are included in the scope of a feature F_j. The primal graph also helps us analyze conditional independence properties of the underlying distribution; specifically each variable is conditionally independent of all other variables given its Markov blanket where the Markov blanket of a variable X_i is the subset of variables of $\boldsymbol{X}$ that are its neighbors in the primal graph.

Parameter (Weight) Learning

We assume that the MN structure, namely the features (and their respective scopes) are known while the parameter vec-

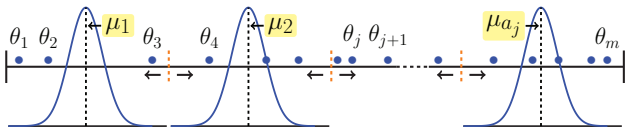

Figure 1: Quantization intervals denoting tied parameters. Each k-partition (interval) contains a set of quantized parameters θ_i (dots) and is associated with a local Gaussian distribution parameterized by $(\mu = \mu_{a_j}, \sigma)$. Short dashed lines on the intervals denote the quantization boundaries which shift according to an optimum (local) penalized parameter setting.

tor, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$, is unknown and needs to be estimated from data. In addition, we assume a fully observed dataset, $\mathcal{D} = \{\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(D)}\}$, that consists of D independent samples drawn from an unknown distribution such that the latter can be represented using the given MN structure. In parameter learning, we seek a choice of parameters that maximizes the log-likelihood of the data.

$$\ell(\boldsymbol{\theta}) = \sum_{d=1}^{D} \log P_{\boldsymbol{\theta}}(\bar{\mathbf{x}}^{(d)}).$$

Formally, this method of parameter learning is referred to as maximum likelihood estimation (MLE). The log-likelihood is a concave function of its parameters, which means that the MLE objective function can be maximized by a standard gradient-based procedure. However, evaluating the log-likelihood as well as its gradient requires computing the log-partition function. As computing the latter is #P-hard in general, tractable alternatives to the log-likelihood such as the pseudo-log-likelihood (Besag 1986) are often used in practice. Formally, maximum pseudo-log-likelihood estimation (MPLE) seeks to find parameters that maximizes the following objective.

$$p\ell(\boldsymbol{\theta}) = \sum_{d=1}^{D} \sum_{i=1}^{n} \log P_{\boldsymbol{\theta}}(\bar{x}_i^{(d)} | \bar{\mathbf{x}}_{\mathrm{MB}(X_i)}^{(d)}), \qquad (1)$$

where $\bar{x}_i^{(d)}$ is the value assigned to X_i in $\bar{\mathbf{x}}^{(d)}$ and $\mathrm{MB}(X_i)$ is the Markov blanket of X_i. Both the pseudo-log-likelihood as well as its gradient can be computed in linear time in the size of the MN and dataset. This yields a scalable approach for MN parameter learning (and we use this scheme in our experiments).

L_2 Regularization

Both MLE and MPLE are prone to overfitting when the size of the training dataset is small compared to the number of parameters. One way to combat overfitting, the Bayesian approach, is to introduce a prior distribution over the parameters. A zero-mean Gaussian prior distribution is the standard choice. After taking logs, the regularizer is of the form $-\frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 = -\frac{\lambda}{2}\sum_{i=1}^{m}(\theta_i)^2$. This penalty term, when added to the learning objective, is generally referred to as L_2 regularization. Two observations: (1) the penalty is concave in the parameters and thus preserves the concavity of both the MLE and MPLE learning objectives; and (2) the *hyperparameter* $\lambda \propto 1/\sigma^2$ controls the variance of the Gaussian distribution. Due to the inverse relation, high lambda values result in low variance and vice versa.

Quantization and Clustering

Quantization is the process of mapping a set of real numbers to a smaller set. Formally, a quantization function $\mathcal{Q}$, is a many-to-one mapping from a set of real numbers $\boldsymbol{A}$ to a set of real numbers $\boldsymbol{B}$, such that $|\boldsymbol{A}| \geq |\boldsymbol{B}|$. Our aim is to find a quantization that minimizes the average quantization error (i.e., $\min \frac{1}{|\boldsymbol{A}|}\sum_{a \in \boldsymbol{A}} |a - \mathcal{Q}(a)|$). A *k-level* quantizer fixes the size of $\boldsymbol{B}$, namely $|\boldsymbol{B}| = k$.

Closely related to the optimal *k-level* quantization problem is the k-means clustering problem (Pollard 1982; Bottou and Bengio 1995). Note that in our work, we only need to solve the 1-dimensional k-means clustering problem, which admits a polynomial time algorithm, $O(m^2k)$, via dynamic programming (Wang and Song 2011). Given a set $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ of real numbers, the 1-dimensional k-means algorithm seeks to partition $\boldsymbol{\theta}$ into k clusters such that the following objective function is minimized

$$\sum_{j=1}^{m} (\theta_j - \mu_{a_j})^2,$$

where $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_k\}$ are the cluster centers or means and $a_j \in \{1, \ldots, k\}$ denotes the cluster assignments, namely the cluster to which θ_j is assigned to. The cluster mean, μ_i, is given by

$$\mu_i = \frac{1}{\sum_{j=1}^{m} I(a_j, i)} \sum_{j=1}^{m} \theta_j \cdot I(a_j, i),$$

where $I(a_j, i)$ is an indicator function which equals 1 if $a_j = i$ and 0 otherwise.

From any k-clustering $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ where $\boldsymbol{a} = \{a_1, \ldots, a_m\}$ and $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_k\}$ of $\boldsymbol{\theta}$, we can define an equivalent quantizer such that $\mathcal{Q}(\theta_j) = \mu_{a_j}$.

Learning MNs with Parameter Tying

We define a parameter tied graphical model $\mathcal{M}$ as a triple $\langle \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{C} \rangle$, where $\boldsymbol{X}$ is a set of random variables, $\boldsymbol{\theta}$ is a set of parameters, and $\mathcal{C}$ is a set of equality constraints such that $\theta_i = \theta_j$ for some $\theta_i, \theta_j \in \boldsymbol{\theta}$. We are interested in learning the optimal parameter tied model $\mathcal{M}_T^*$ from data. Formally, the optimization problem can be defined as follows. Given training data $\mathcal{D}$ on variable set $\boldsymbol{X}$, find the constraint set $\mathcal{C}$ and parameters $\boldsymbol{\theta}$ such that the parameters respect the constraints and the log-likelihood of data is maximized.

For a fixed set of constraints, the learning problem can be formulated as maximizing a concave objective (the log-likelihood) over a convex set (the set of constraints). This can be done via projected gradient descent. However, searching over all possible constraint sets to find the best partition is infeasible. Specifically, given m parameters, the number of partitions of size k is given by the *Stirling numbers of the second kind*, denoted as $\left\{ {m \atop k} \right\}$. The total number

of partitions of a set is given by the *Bell number*, $B_m = \sum_{k=1}^{m} \left\{ {m \atop k} \right\}$. However, the problem can be simplified if we relax the equality constraints. Our approach is to approximate the equality constraints utilizing a penalty function which enforces a *soft tying* of parameters. We reformulate the learning of a parameter tied graphical model by adding a penalty for poor clusterings to the log-likelihood objective.

$$\arg\max_{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{\mu}} \ \ell(\boldsymbol{\theta}) - \frac{\lambda}{2}\sum_{j=1}^{m}(\theta_j - \mu_{a_j})^2. \tag{2}$$

The objective in equation (2) represents a regularized log-likelihood similar to L_2 regularization. Here, the penalty function is the k-means objective with an additional tuning hyperparameter λ that controls the magnitude of the penalty. This corresponds to a collection of k Gaussian priors such that the i-th prior has mean μ_i and variance proportional to $1/\lambda$ (see Figure 1). However, while equation (2) is concave in $\boldsymbol{\theta}$ for a fixed clustering, the objective function is no longer a concave optimization problem when $\boldsymbol{a}$ is not given (note that, in general, the k-means objective is not convex). It is easy to show that L_2 regularization is a special case of the objective in equation (2); all we have to do is assume that there is only one cluster and $\mu_1 = 0$.

Block Coordinate Ascent Learning Algorithm

In this section, we derive a block coordinate ascent technique for equation (2), thus achieving our aim of automating parameter tying in MNs. We will refer to this general algorithm simply as APT going forward. Given training data $\mathcal{D}$, the MN structure $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{\Phi} \rangle$, k clusters and penalty term λ, the APT algorithm performs coordinate ascent on the objective in equation (2) by alternating between finding the optimal parameters for a fixed clustering and finding the optimal clustering for a fixed vector of parameters (see Algorithm 1). Both of these optimizations are straightforward: a regularized maximum likelihood optimization problem and a one-dimensional k-means clustering problem respectively. The former can be solved using standard gradient ascent while the latter can be solved in polynomial time using dynamic programming.

Next, we make several remarks about Algorithm 1, which illustrate the flexibility and utility of our proposed method.

First, Algorithm 1 returns a soft clustering of the parameters. However, we can easily turn the soft clustering into hard clustering and relearn the parameters using the MLE objective while enforcing equality constraints $\mathcal{C}$ on all parameters assigned to the same cluster. This can be done via projected gradient ascent. That is, after each gradient step, the parameter vector may step outside of the set of constraints. If this happens, we simply project the parameters back into the constraint set. As the cluster constraints insist that all parameters in cluster i must have the same value, the projection operation simply replaces all parameters in cluster i with the average of all parameters in cluster i.

Second, since the objective function in equation (2) is bounded from above, the coordinate ascent procedure is guaranteed to converge to a local maxima. The algorithm increases the objective function each iteration since each

Algorithm 1: Automatic Parameter Tying (APT)

Input: A Markov network structure $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{\Phi} \rangle$, Integer k, Integer T and penalty λ.
Output: Feature vector $\boldsymbol{\theta}$ and clustering $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$.
begin
  Initialize $\boldsymbol{\theta}^{(0)}$ and $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ to random values.
  for $t = 1$ *to* T *or until convergence* **do**
    1. Update θ given $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ using gradient ascent:

$$\boldsymbol{\theta}^{(t)} = \arg\max_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}) - \frac{\lambda}{2}\sum_{j=1}^{m}(\theta_j - \mu_{a_j^{(t-1)}}^{(t-1)})^2$$

    2. Update $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ using 1D k-means given $\boldsymbol{\theta}$:

$$\langle \boldsymbol{a}^{(t)}, \boldsymbol{\mu}^{(t)} \rangle = \arg\min_{\boldsymbol{a},\boldsymbol{\mu}} \sum_{j=1}^{m}(\theta_j^{(t)} - \mu_{a_j})^2$$

    where $\mu_i = \frac{1}{\sum_{j=1}^{m} I(a_j,i)} \sum_{j=1}^{m} \theta_j \cdot I(a_j, i)$ and $I(a_j, i)$ is an indicator function which equals 1 if $a_j = i$ and 0 otherwise.
  return $\langle \boldsymbol{\theta}^{(T)}, \boldsymbol{a}^{(T)}, \boldsymbol{\mu}^{(T)} \rangle$

of the two sub-optimization problems, finding the optimal parameters for a fixed clustering (increasing log-likelihood) and finding the optimal clustering for a fixed vector of parameters (reducing penalty), contributes to increasing the objective function. In practice, the rate of convergence can be improved by initializing the parameters and cluster means to small values or after running a few iterations of gradient ascent that optimizes MLE plus an L_2 regularization term.

Third, note that Algorithm 1 optimizes MLE plus a penalty term which requires inference over the MN. The latter is often infeasible in practice. Its practical performance and convergence can be improved by using two strategies: (a) stochastic or mini-batch gradient ascent and (b) using the MPLE objective instead of MLE. Stochastic and mini batch gradient ascent can be used in two ways. First, we can use it to optimize $\boldsymbol{\theta}$ in Step 1 of the for loop of Algorithm 1. Second, we can use it in the outer loop by not running Step 1 until convergence, namely we run gradient ascent only for a few iterations in Step 1. In our experiments, we employ both strategies to speed up our algorithm.

Theoretical Analysis

In this section, we analyze the sample complexity of the proposed method and provide conditions under which it provably recovers the correct clustering assignments $\boldsymbol{a}$ and approximate cluster means $\boldsymbol{\mu}$ with small L_1 error. We demonstrate polynomial sample complexity when the clusters are well-separated (defined formally below). We consider two cases: (1) *Hard Tying*: when the true MN from which the data is generated has exactly $k < m$ unique parameters; and (2) *Soft Tying*: when the MN has m parameters and k is the number of clusters.

We begin by establishing conditions for well-separation of clusters in Lemmas 1 and Corollary 1. Let ω denote the maximum width of a cluster where the width of a cluster is

the distance between its farthest points. Formally,

$$\omega = \max_i \max_{(r,s)|a_r=i,a_s=i} |\theta_r - \theta_s|.$$

Let α denote the distance between two closest points in different clusters, namely

$$\alpha = \min_{(i,j)|a_i \neq a_j} |\theta_i - \theta_j|.$$

We will refer to α as the minimum inter-cluster distance. Let $\hat{\theta}_i$ and $\hat{\mu}_j$ denote the estimates of the true parameters θ_i and μ_j respectively based on N samples drawn independently and identically from the true MN.

Lemma 1. *Let $\max_i |\theta_i - \hat{\theta}_i| \leq \epsilon$. Then the 1D k-means clustering algorithm is guaranteed to return optimal cluster assignments if $\epsilon < \frac{\alpha-\omega}{4}$.*

Proof. (Sketch.) Notice that since the maximum error is bounded by ϵ, the maximum cluster width derived from the estimated parameters is bounded from above by $\omega + 2\epsilon$. Similarly, the minimum inter-cluster distance derived from the estimated parameters is bounded from below by $\alpha - 2\epsilon$. Thus, in order to ensure that the estimates $\hat{\theta}_i$ and $\hat{\theta}_j$ of any two parameters θ_i and θ_j such that a_i equals a_j in true model also have equal cluster assignments in the estimated model, the following constraint should be satisfied: $\alpha - 2\epsilon > \omega + 2\epsilon$. Rearranging, we get $\epsilon < \frac{\alpha-\omega}{4}$. □

Corollary 1. *For the hard tying case (namely there are k unique parameters) the 1D k-means clustering algorithm is guaranteed to return optimal cluster assignments if $\epsilon < \frac{\alpha}{4}$.*

Proof. When there are k unique parameters that also correspond to the cluster centers, the maximum cluster width ω equals 0 and the proof follows from Lemma 1. □

Lemma 1 and Corollary 1 help us derive the following definition for well-separation. We say that the triple $\langle \boldsymbol{\theta}, \boldsymbol{a}, \boldsymbol{\mu} \rangle$ denoting the parameters as well as cluster assignments and centers is *well-separated* for a given error bound ϵ iff $\epsilon < \frac{\alpha-\omega}{4}$ for the soft tying case and $\epsilon < \frac{\alpha}{4}$ for the hard tying case.

Next we use Lemma 1 and Corollary 1 in conjunction with the PAC and sample complexity bounds for MLE derived in (Bradley and Guestrin 2012) and (Ravikumar, Wainwright, and Lafferty 2010) to yield our desired sample complexity bounds. Formally,

Theorem 1 (**MLE Sample Complexity**). *Let $C_{min} > 0$ be a lower bound on the minimum eigenvalue of the Hessian of the negative log likelihood. Let the regularization hyperparameter λ be chosen such that $\lambda = C_{min}^2 n^{-\xi/2}/(2^6 m^2)$ where n is the number of training samples, m is the number of feature weights and $\xi \in (0, 1)$. Then, to recover the optimal cluster assignments $\boldsymbol{a}$ and centers $\boldsymbol{\mu}$ with L_1 error smaller than $(\alpha - \omega)/4$ with probability at least $(1 - \delta)$, it suffices to have training set size*

$$n \geq \frac{2^9}{C_{min}^2} \frac{16m^2}{(\alpha-\omega)^2} \log \frac{2m(m+1)}{\delta}.$$

For hard tying in which we have k unique parameters, the sample complexity for finding optimal cluster assignments $\boldsymbol{a}$ and centers $\boldsymbol{\mu}$ with L_1 error smaller than $\alpha/4$ is given by

$$n \geq \frac{2^9}{C_{min}^2} \frac{16k^2}{\alpha^2} \log \frac{2m(m+1)}{\delta}.$$

Proof. (Sketch.) Note that the bounds in Bradley and Guestrin straightforwardly apply to Step 1 (gradient ascent given fixed cluster centers and assignments) of our algorithm. The bounds in the theorem are obtained by substituting the error term in Bradley and Guestrin's bounds with the bounds derived in Lemma 1 and Corollary 1.[1] □

The sample complexity bound implies that when the minimum eigenvalue is large and/or when the difference between the minimum inter-cluster distance and the maximum cluster width is large (namely, the clusters are well separated), our algorithm is statistically efficient. As expected, the hard tying case is statistically more efficient than the soft tying case since the former does not depend on the cluster width.

Experiments

Experimental Setup

We evaluated APT on 20 real-world datasets which have been widely used in recent years to evaluate learning algorithms for probabilistic graphical models (Rahman and Gogate 2016; Rooshenas and Lowd 2014; Davis and Domingos 2010) (see Table 1 for details on the binary datasets). We implemented APT in C++ and all experiments were conducted on Intel i7 Ubuntu machines with 16GB of RAM.

For each dataset, we learned a pairwise binary MN structure (not the parameters) using the L_1 regularization based structure learning algorithm of (Ravikumar, Wainwright, and Lafferty 2010). This algorithm constructs the MN structure as follows. It learns a L_1 regularized logistic regression classifier $L(X_i)$ for predicting the value of each variable X_i given all other variables. Then, it adds an edge between two variables X_i and X_j if the features corresponding to X_i or X_j have non-zero weights in $L(X_j)$ and $L(X_i)$ respectively. Unfortunately, on many datasets, this method yields dense models. Therefore, in order to achieve sparsity we constrained the regularization hyperparameter (λ) so that the degree (size of the Markov blanket) of each variable is bounded by d. In our experiments, we used the following values for $d = \{5, 10, 15, 30, 50\}$ where $d = 10, 30$ are shown in supplemental material due to space constraint. We learned the L_1 regularized logistic regression classifier using the Orthant-Wise Limited-memory Quasi Newton (OWL-QN) method (Andrew and Gao 2007).

We experimented with the following values for the number of clusters $k = \{1, 2, 5, 10, 20, 100, 500, 1000, 5000, 10000\}$ and the regularization hyperparameter $\lambda = \{0.01, 0.1, 0.5, 1, 10, 15, 30, 50, 100\}$. We ran stochastic gradient ascent until convergence (or maximum of 5M iterations).

[1] Similarly, we can use Bradley and Guestrin's MPLE bounds to derive MPLE sample complexity bounds for APT. The derivation is straightforward; we just have to substitute the expression for ϵ in their MPLE bound. We skip the details for brevity.

					$d=5$			$d=15$			$d=50$		
Dataset	#vars	#train	#valid	#test	L_2	LTR	APT	L_2	LTR	APT	L_2	LTR	APT
nltcs	16	16181	2157	3236	5.05	**5.02**	5.02	5.10	4.99	**4.98**	–	–	–
msnbc	17	291326	38843	58265	6.17	6.12	**6.11**	6.22	6.10	**6.08**	–	–	–
kdd	64	180092	19907	34955	2.12	2.11	**2.11**	2.09	**2.08**	2.09	2.14	2.08	**2.07**
plants	69	17412	2321	3482	10.68	10.63	**10.59**	10.46	10.23	**10.21**	11.11	10.26	**10.24**
audio	100	15000	2000	3000	38.88	38.46	**38.44**	38.34	37.31	**37.22**	40.73	37.47	**37.03**
jester	100	9000	1000	4116	51.94	51.41	**51.28**	51.21	50.00	**49.75**	54.97	50.53	**50.04**
netflix	100	15000	2000	3000	54.91	54.41	**54.40**	54.22	52.84	**52.68**	57.52	53.32	**52.67**
accidents	111	12758	1700	2551	14.71	14.50	**14.47**	13.21	12.78	**12.70**	13.85	12.90	**12.69**
retail	135	22041	2938	4408	10.53	10.46	**10.45**	10.57	10.41	**10.40**	10.93	10.40	**10.39**
pumsb*	163	12262	1635	2452	11.58	11.47	**11.46**	10.13	9.80	**9.79**	11.17	9.94	**9.79**
dna	180	1600	400	1186	59.16	58.54	**58.46**	61.92	59.73	**59.54**	69.26	63.13	**62.84**
kosarek	190	33375	4450	6675	10.41	10.35	**10.34**	10.32	10.17	**10.17**	10.59	10.27	**10.25**
msweb	294	29441	3270	5000	16.98	16.80	**16.79**	17.04	**16.60**	16.60	14.80	13.74	**13.71**
book	500	8700	1159	1739	35.90	36.68	**35.82**	35.49	37.70	**35.20**	36.48	42.14	**35.88**
tmovie	500	4524	1002	591	71.91	72.87	**71.49**	63.16	66.43	**62.94**	61.22	66.22	**58.50**
webkb	839	2803	558	838	158.31	163.21	**158.08**	157.30	169.17	**155.51**	180.85	203.54	**158.71**
reuters	889	6532	1028	1540	91.33	92.29	**91.26**	**88.55**	91.98	88.65	91.19	99.66	**88.83**
20ng	910	11293	3764	3764	163.90	164.36	**163.30**	**160.82**	162.38	162.29	170.94	167.38	**166.71**
bbc	1058	1670	225	330	259.96	275.06	**259.18**	267.44	292.66	**256.60**	331.67	343.90	**260.95**
ad	1556	2461	327	491	6.79	6.58	**6.55**	6.37	**6.11**	6.16	6.22	**6.01**	6.06

Table 1: For each dataset, cols. (2-5): dataset characteristics; remaining cols.: test set neg. PLL scores (k and λ selected using the validation set) on 20 benchmark datasets for L_2 regularization, LTR and APT algorithm under various values of d.

The cluster centers were updated every 50K iterations. We used average PLL score to evaluate the resulting models (see equation (1)).

APT versus L_2 regularization

Table 1 shows the test set negative PLL scores (using various values of d) for APT and L_2 regularization on the 20 datasets. For each dataset, we select the k and λ values using the validation set. Lower negative PLL values are better and bold signifies the higher value achieved by the respective regularization method. From the results, we can clearly see APT outperforms L_2 across the majority (with the exception of reuters and 20ng) of datasets and complexity of model structure. Moreover, as the structure becomes increasingly dense (more neighboring nodes), the performance gap widens. One key takeaway here is when the underlying model has a large number of parameters, it is prudent to utilize APT for better generalization performance. Parameter learning algorithms for complex models such as CNNs or SRL models can leverage our method since the model will contain parameters that take on similar values, which is evident from the results.

APT versus LTR

We also compared APT with our adaptation of (Chou et al. 2016) to MNs, which we refer to as the learn-tie-relearn algorithm (LTR). Their algorithm only operates on BNs and a straightforward extension of their method to parameter learning in MNs is the following: (1) learn the parameters using the MLE objective; (2) cluster the resulting parameters into k clusters; and (3) relearn the parameters by adding equality constraints over the parameters in each cluster. However, we found that this approach has high variance. This is likely due to the scale invariance property of MNs. To combat this, in step (1) of the algorithm, we learned the parameters using L_2 regularization, which greatly reduces the variance and thus improves the performance of LTR.

Table 1 also shows the test set negative PLL scores (using various values of d) for APT and LTR. For each dataset we select the k and λ values using the validation set. Lower negative PLL values are better and bold signifies the best value achieved by the respective regularization method. From the results, we clearly observe that APT outperforms LTR across the majority of datasets and complexity of model structure (measured by d). However, the noticeable deviation from the previous results is that wider differences occur in datasets with higher complexity (more variables and neighboring nodes) as in the case with bbc ($d = 50$). Comparatively, we see that LTR mostly outperforms L_2. Thus, hard tying the parameters output by L_2 is highly beneficial.

Impact of varying k

Figure 2 shows average negative test set PLL scores for APT and L_2 regularization as a function of the number of clusters k on four randomly chosen datasets. To better organize the results and to avoid clutter, the comparison was made by fixing the maximum number of d neighboring nodes to 5, 15 and 50 for the learned MN structures and across varying k clusters. Consistent with the previous results, more complex structures ($d = 50$) create a wider performance gap between APT and L_2. Conversely, the performance gap is closer for simpler models ($d = 5$). The plots also show that by having the ability to control the parameter k (number of clusters), there is an optimal setting where the lowest test average negative PLL score can be achieved. For example, in dna, the best test average negative PLL score requires approximately 20 clusters. Overall, for each of the datasets, there is a setting of k where APT outperforms L_2. This demonstrates the utility of our approach.

We found that our algorithm converges rapidly and re-

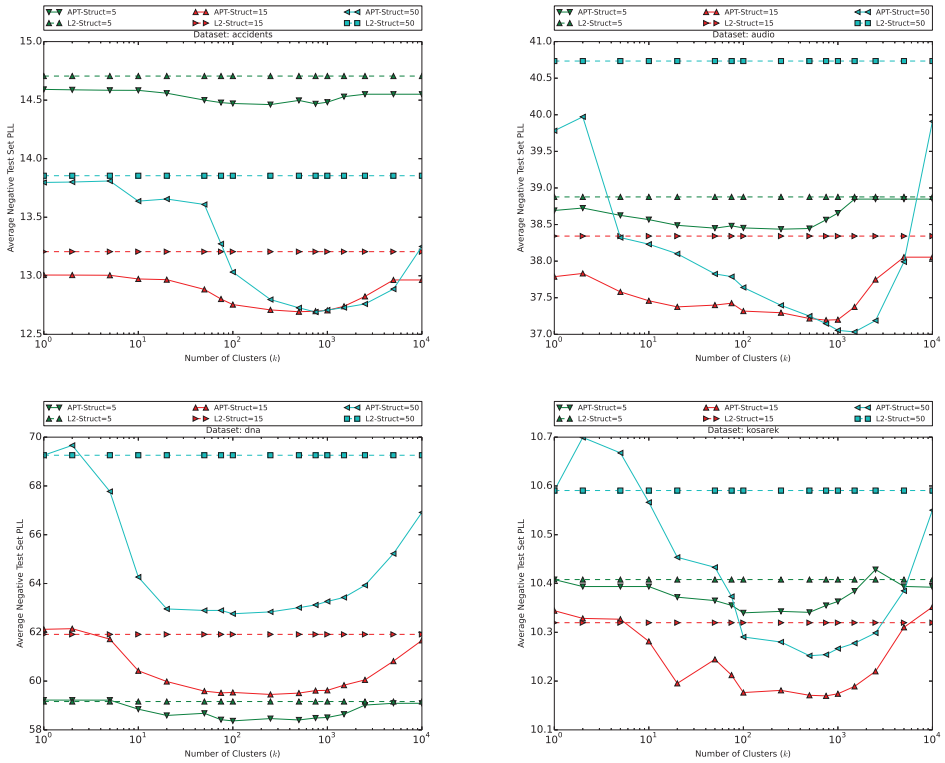

Figure 2: Avg. test set neg. PLL scores for L_2 (dotted) and APT (solid) for varying # of clusters (k) and model complexity (d). To minimize clutter, we show graphs for only three values of d and do not include results for LTR (whose variance is quite high).

quires roughly the same number of iterations as L_2 to converge in practice. Moreover, its variance is also low (and therefore not plotted in the graphs and tables), namely most local maxima reached achieve similar solution quality.

We end this section by mentioning several pragmatic techniques to achieve good practical performance. First, it is often beneficial to increase the regularization hyperparameter λ with the number of iterations. As the cluster centers and assignments converge, increasing λ increases the penalty and yields rapid convergence to good solutions. Second, hard tying is beneficial only for moderate to large values of k and in fact for small values of k, it may hurt the performance significantly. For small values of k, soft tying works much better. Finally, the rate of convergence can be improved in practice by updating the cluster centers but not the cluster assignments at each iteration of gradient ascent in Step 1 of Algorithm 1. Periodically updating the cluster assignments appears to yield much faster convergence than updating them at each iteration.

Conclusion

We investigated parameter tying, an alternative regularization method for MNs. Unlike other machine learning frameworks where parameter tying is specified *a priori*, we introduced an automatic approach to tying parameters (APT). Specifically, we incorporated a more informative and general penalty term that leverages clustering into the objective function for parameter learning in MNs. We showed that our approach generalizes L_2 regularization. Since our formulation of the penalized learning problem is no longer concave, we proposed a block coordinate ascent algorithm to solve the optimization problem efficiently. We provided sample complexity bounds for our proposed algorithm which show significant improvement over standard L_2 regularization with high probability when the clusters are well-separated. Empirically, we showed that our approach outperforms L_2 regularization on a variety of real-world datasets.

Acknowledgements

This work was supported in part by the DARPA Explainable Artificial Intelligence (XAI) Program under contract number N66001-17-2-4032, and by the National Science Foundation grants III-1527312, IIS-1652835 and IIS-1528037. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of DARPA, NSF or the US government.

References

Andrew, G., and Gao, J. 2007. Scalable training of l1-regularized log-linear models. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, 33–40.

Bach, S.; Broecheler, M.; Huang, B.; and Getoor, L. 2015.

Hinge-loss markov random fields and probabilistic soft logic. *CoRR* arXiv:1505.04406 [cs.LG].

Besag, J. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B* 48:259–302.

Bottou, L., and Bengio, Y. 1995. Convergence Properties of the K-Means Algorithms. In *Advances in Neural Information Processing Systems 7*, 585–592.

Bradley, J., and Guestrin, C. 2012. Sample complexity of composite likelihood. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, 136–160.

Chou, L.; Sarkhel, S.; Ruozzi, N.; and Gogate, V. 2016. On parameter tying by quantization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3241–3247.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Davis, J., and Domingos, P. 2010. Bottom-up learning of markove network structure. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, 271–278.

Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool.

Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Han, S.; Mao, H.; and Dally, W. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representation*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liu, J., and Page, D. 2013. Bayesian estimation of latently-grouped parameters in undirected graphical models. In *Proceedings of the Twenty-Sixth International Conference on Neural Information Processing Systems*, 1232 – 1240.

Nowlan, S., and Hinton, G. 1992. Adaptive soft weight tying using gaussian mixtures. In *Advances in Neural Information Processing Systems 4*, 993–1000.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pollard, D. 1982. Quantization and the Method of k-Means. *IEEE Transactions on Information Theory* IT-28(2).

Rahman, T., and Gogate, V. 2016. Merging strategies for sum-product networks. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 617–626.

Ravikumar, P.; Wainwright, M. J.; and Lafferty, J. 2010. High-dimensional ising model selection using L1-regularized logistic regression. *Annals of Statistics* 38(3):1287–1319.

Rooshenas, A., and Lowd, D. 2014. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the Thirty-First International Conference on Machine Learning*, 710–718.

Roth, D. 1996. On the Hardness of Approximate Reasoning. *Artificial Intelligence* 82:273–302.

Wang, H., and Song, M. 2011. Ckmeans.1d.dp: Optimal k-means Clustering in One Dimension by Dynamic Programming. *The R Journal* 3(2).

Wang, Z.; Duan, L.; Huang, T.; and Gao, W. 2016. Affinity preserving quantization for hashing: A vector quantization approach to learning compact binary codes. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1102–1108.

Zhu, H.; Long, M.; Wang, J.; and Cao, Y. 2016. Deep hashing network for efficient similarity retrieval. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2415–2421.