EDML for Learning Parameters in Directed and Undirected Graphical Models

Khaled S. Refaat KREFAAT@CS.UCLA.EDU
Arthur Choi AYCHOI@CS.UCLA.EDU
Adnan Darwiche DARWICHE@CS.UCLA.EDU
Computer Science Department, University of California, Los Angeles, CA 90095 USA

Abstract

EDML is a recently proposed algorithm for learning parameters in Bayesian networks. It was originally derived in terms of approximate inference on a meta-network which underlies the Bayesian approach to parameter estimation. While this initial derivation helped discover EDML in the first place and provided a concrete context for identifying some of its properties (e.g., in contrast to EM), the formal setting was somewhat tedious in the number of concepts it drew on. In this paper, we propose a greatly simplified perspective on EDML which casts it as a general approach to continuous optimization. The new perspective has several advantages. First, it makes immediate some results that were non-trivial to prove initially. Second, it facilitates the design of EDML algorithms for new graphical models, leading to a new algorithm for learning parameters in Markov networks. We derive this algorithm in this paper, and provide an empirical comparison with a commonly used gradient method, showing that EDML can find better estimates several times faster.

1. Introduction

EDML is a recently proposed algorithm for learning MAP parameters of a Bayesian network from incomplete data (Choi et al., 2011; Refaat et al., 2012). EDML is procedurally very similar to Expectation Maximization (EM) (Dempster et al., 1977; Lauritzen, 1995), yet EDML was shown to have certain advantages, both theoretically and practically. Theoretically, EDML can in certain specialized cases provably converge in one iteration, whereas EM may require many iterations to solve the same learning problem. Some empirical evaluations further suggested that EDML and hybrid EDML/EM algorithms could find better parameter estimates than vanilla EM, in fewer iterations and less time.

EDML was originally derived in terms of approximate inference on a meta-network used for Bayesian approaches to parameter estimation; see, for example, Figure 2. This graphical representation of the estimation problem lent itself to the initial derivation of EDML, as well to the identification of certain key theoretical properties, such as the one we just described. The formal details, however, can be somewhat tedious as EDML draws on a number of different concepts. We review EDML in such terms in Appendix B.

In this paper, we propose a much simpler perspective on EDML, which views it more abstractly in terms of a simple method for continuous optimization, which we describe. This new perspective has a number of advantages. First, it makes immediate some results that were previously obtained for EDML, but through some effort. Second, it facilitates the design of new EDML algorithms for new classes of models, where graphical formulations of parameter estimation, such as meta-networks, are lacking. Here, we derive in particular a new parameter estimation algorithm for Markov networks, which is in many ways a more challenging task, compared to the case of say Bayesian networks. Empirically, we find that EDML is capable of finding better parameter estimates, under complete data, several times faster than more popular approaches like conjugate-gradient methods.

This paper is structured as follows. In Section 2, we highlight a simple iterative method for (approximately) solving continuous optimization problems. In Section 3, we formulate the EDML algorithm for parameter estimation in Bayesian network, as an instance of this optimization method. In Section 4, we derive a new EDML algorithm for Markov networks, based on

the same perspective. In Section 5, we contrast the two EDML algorithms for directed and undirected graphical models, in the complete data case. We empirically evaluate our new algorithm for parameter estimation in Markov networks, in Section 6, review related work in Section 7, and conclude in Section 8. Proofs of theorems appear in Appendix A.

2. An Approximate Optimization of Real-Valued Functions

Consider a real-valued objective function $f(x)$ whose input x is a vector of components:

$$x = (x_1, \ldots, x_i, \ldots, x_n),$$

where each component x_i is a vector in $\mathbb{R}^{k_i}$ for some k_i. Suppose further that we have a constraint on the domain of function $f(x)$ with a corresponding function g that maps an arbitrary point x to a feasible one $g(x)$.

Our goal here is to find a feasible input vector $x = (x_1, \ldots, x_i, \ldots, x_n)$ that optimizes the function $f(x)$. Given the difficulty of this optimization problem in general, we will settle for finding stationary points x in the constrained domain of function $f(x)$.

One approach for finding such stationary points is as follows. Let $x^\star = (x_1^\star, \ldots, x_i^\star, \ldots, x_n^\star)$ be a feasible point in the domain of function $f(x)$. For each component x_i, we define a sub-function

$$f_{x^\star}(x_i) = f(x_1^\star, \ldots, x_{i-1}^\star, x_i, x_{i+1}^\star, \ldots, x_n^\star).$$

That is, we use the n-ary function $f(x)$ to generate n sub-functions $f_{x^\star}(x_i)$. Each of these sub-functions is obtained by fixing all inputs x_j of $f(x)$, for $j \neq i$, to their values in $x^\star$, while keeping the input x_i free. We further assume that these sub-functions are subject to the same constraints that the function $f(x)$ is subject to.

We can now characterize all feasible points $x^\star$ that are stationary with respect to the function $f(x)$, in terms of local conditions on sub-functions $f_{x^\star}(x_i)$.

Claim 1 *A feasible point $x^\star = (x_1^\star, \ldots, x_i^\star, \ldots, x_n^\star)$ is stationary for function $f(x)$ iff for all i, component $x_i^\star$ is stationary for sub-function $f_{x^\star}(x_i)$.*

This is immediate from the definition of a stationary point. Assuming no constraints, a stationary point has gradient $\nabla f = 0$, i.e., $\nabla_{x_i} f(x^\star) = \nabla f_{x^\star}(x_i) = 0$ for all x_i, where $\nabla_{x_i} f$ denotes the sub-vector of gradient ∇f with respect to component x_i.[1]

[1] Under constraints, we consider points that are stationary with respect to the corresponding Lagrangian.

With these observations, we can now search for feasible stationary points $x^\star$ of the constrained function $f(x)$ using an iterative method that searches instead for stationary points of the sub-functions $f_{x^\star}(x_i)$. The method works as follows:

1. Start with some feasible point x^t of function $f(x)$ for $t = 0$
2. While some x_i^t is not a stationary point for constrained sub-function $f_{x^t}(x_i)$
   (a) Find a stationary point y_i^{t+1} for each constrained sub-function $f_{x^t}(x_i)$
   (b) $x^{t+1} = g(y^{t+1})$
   (c) Increment t

Note that the real computational work of this iterative procedure is in Steps 2(a) and 2(b), although we shall see later that such steps can sometime be performed efficiently. With an appropriate feasibility function $g(y)$,[2] one can guarantee that a fixed-point of this procedure yields a stationary point of the constrained function $f(x)$, by Claim 1. Further, any stationary point is trivially a fixed-point of this procedure (one can seed this procedure with such a point).

As we shall show in the next section, the EDML algorithm—which has been proposed for parameter estimation in Bayesian networks—is an instance of the above procedure with some notable observations: (1) the sub-functions $f_{x^t}(x_i)$ are convex and have a unique optimum; (2) these sub-functions have an interesting semantics, as they correspond to posterior distributions that are induced by Naive Bayes networks with soft evidence asserted on them; (3) defining these sub-functions requires inference in a Bayesian network parameterized by the current feasible point x^t; (4) there are already several convergent, fixed-point iterative methods for finding the unique optimum of these sub-functions; and (5) these convergent methods produce solutions that are always feasible and, hence, the feasibility function $g(y)$ corresponds to the identity function $g(y) = y$ in this case.

We next show this connection to EDML as proposed for parameter estimation in Bayesian networks. We follow by deriving an EDML algorithm (another instance of the above procedure), but for parameter estimation in undirected graphical models. We will also study the impact of having complete data on both versions of the EDML algorithm, and finally evaluate the new instance of EDML by comparing it to a conjugate gradient method when applied to complete datasets.

[2] See Theorem 5 in Appendix A

3. EDML for Bayesian Networks

From here on, we use upper case letters (X) to denote variables and lower case letters (x) to denote their values. Variable sets are denoted by bold-face upper case letters ($\mathbf{X}$) and their instantiations by bold-face lower case letters ($\mathbf{x}$). Generally, we will use X to denote a variable in a Bayesian network and $\mathbf{U}$ to denote its parents. A network parameter will therefore have the general form $\theta_{x|\mathbf{u}}$, representing the probability $Pr(X=x|\mathbf{U}=\mathbf{u})$.

Consider a (possibly incomplete) dataset $\mathcal{D}$ with examples $\mathbf{d}_1, \ldots, \mathbf{d}_N$. Consider also a Bayesian network with parameters θ. Our goal here is to find parameter estimates θ that minimize the negative log-likelihood:

$$f(\theta) = -\ell\ell(\theta|\mathcal{D}) = -\sum_{i=1}^{N} \log Pr_\theta(\mathbf{d}_i). \tag{1}$$

Here, $\theta = (\ldots, \theta_{X|\mathbf{u}}, \ldots)$ is a vector over the network parameters. Moreover, Pr_θ is the distribution induced by the Bayesian network structure under parameters θ. As such, $Pr_\theta(\mathbf{d}_i)$ is the probability of observing example $\mathbf{d}_i$ in dataset $\mathcal{D}$ under parameters θ.

Each component of θ is a parameter set $\theta_{X|\mathbf{u}}$, which defines a parameter $\theta_{x|\mathbf{u}}$ for each value x of variable X and instantiation $\mathbf{u}$ of its parents $\mathbf{U}$. The feasibility constraint here is that each component $\theta_{X|\mathbf{u}}$ satisfies the convex sum-to-one constraint: $\sum_x \theta_{x|\mathbf{u}} = 1$.

The above parameter estimation problem is clearly in the form of the constrained optimization problem that we phrased in the previous section and, hence, admits the same iterative procedure proposed in that section for finding stationary points. The relevant questions now are: What form do the sub-functions $f_{\theta^\star}(\theta_{X|\mathbf{u}})$ take in this context? What are their semantics? What properties do they have? How do we find their stationary points? Finally, what is the feasibility function $g(y)$ in this case? We address these questions next.

3.1. Form

We start by characterizing the sub-functions of the negative log-likelihood given in Equation 1.

Theorem 1 *For a given parameter set $\theta_{X|\mathbf{u}}$, the negative log-likelihood of Equation 1 has sub-functions:*

$$f_{\theta^\star}(\theta_{X|\mathbf{u}}) = -\sum_{i=1}^{N} \log\left(C^i_{\mathbf{u}} + \sum_x C^i_{x|\mathbf{u}} \cdot \theta_{x|\mathbf{u}}\right) \tag{2}$$

where $C^i_{\mathbf{u}}$ and $C^i_{x|\mathbf{u}}$ are constants that are independent of parameter set $\theta_{X|\mathbf{u}}$, given by

$$C^i_{\mathbf{u}} = Pr_{\theta^\star}(\mathbf{d}_i) - Pr_{\theta^\star}(\mathbf{u}, \mathbf{d}_i)$$
$$C^i_{x|\mathbf{u}} = Pr_{\theta^\star}(x, \mathbf{u}, \mathbf{d}_i)/\theta^\star_{x|\mathbf{u}}$$

Note that computing the constants C^i requires inference on the Bayesian network under parameters $\theta^\star$.[3]

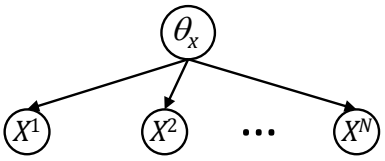

Figure 1. Estimation given independent observations.

3.2. Semantics

Equation 2 has an interesting semantics, as it corresponds to the negative log-likelihood of a root variable in a naive Bayes structure, on which soft, not necessarily hard, evidence is asserted (Choi et al., 2011).[4]

This model is illustrated in Figure 1. Let $\mathbb{P}$ denote the distribution of this model and let $\eta = (\eta_1, \ldots, \eta_N)$ denote soft observations on variable X, with strengths $\mathbb{P}(\eta_i|x)$. We then have

$$\log \mathbb{P}(\eta|\theta_X) = \sum_{i=1}^{N} \log \sum_x \mathbb{P}(\eta_i|x) \cdot \theta_x \tag{3}$$

The following result connects Equation 2 to the above likelihood of a soft dataset.

Theorem 2 *Consider Equations 2 and 3, and assume that each soft evidence η_i has the strength*

$$\mathbb{P}(\eta_i|x) \ = C^i_{\mathbf{u}} + C^i_{x|\mathbf{u}}$$

It then follows that

$$f_{\theta^\star}(\theta_{X|\mathbf{u}}) = -\log \mathbb{P}(\eta|\theta_{X|\mathbf{u}}) \tag{4}$$

This theorem yields the following interesting semantics for EDML sub-functions. Consider a parameter set

[3] Theorem 1 assumes tacitly that $\theta^\star_{x|\mathbf{u}} \neq 0$. More generally, however, $C^i_{x|\mathbf{u}} = \partial Pr_{\theta^\star}(\mathbf{d}_i)/\partial\theta_{x|\mathbf{u}}$, which can also be computed using some standard inference algorithms (Darwiche, 2003; Park & Darwiche, 2004).

[4] Soft evidence is an observation that increases or decreases ones belief in an event, but not to the point of certainty. The strength of soft evidence can be specified using weights on a set of mutually exclusive and exhaustive events. For more, see (Chan & Darwiche, 2005).

$\theta_{X|\mathbf{u}}$ and example $\mathbf{d}_i$ in our dataset. The example can then be viewed as providing "votes" on what this parameter set should be. In particular, the vote of example $\mathbf{d}_i$ for value x takes the form of a soft evidence η_i whose strength is given by

$$\mathbb{P}(\eta_i|x) = Pr_{\theta^\star}(\mathbf{d}_i) - Pr_{\theta^\star}(\mathbf{u}, \mathbf{d}_i) + Pr_{\theta^\star}(x, \mathbf{u}, \mathbf{d}_i)/\theta^\star_{x|\mathbf{u}}$$

The sub-function is then aggregating these votes from different examples and producing a corresponding objective function on parameter set $\theta_{X|\mathbf{u}}$. EDML optimizes this objective function to produce the next estimate for parameter set $\theta_{X|\mathbf{u}}$.

3.3. Properties

Equation 2 is a convex function, and thus has a unique optimum.[5] In particular, we have logs of a linear function, which are each concave. The sum of two concave functions is also concave, thus our sub-function $f_{\theta^\star}(\theta_{X|\mathbf{u}})$ is convex, and is subject to a convex sum-to-one constraint (Refaat et al., 2012).

Convex functions are relatively well understood, and there are a variety of methods and systems that can be used to optimize Equation 2 (Boyd & Vandenberghe, 2004). We describe one such approach, next.

3.4. Finding the Unique Optimum

In the estimation problem of Equation 3, we want to learn the parameters $\theta_{X|\mathbf{u}}$ given a soft dataset η. We have previously proposed a fixed-point algorithm that is convergent, and monotonically improves the objective (Refaat et al., 2012). Moreover, the solutions it produces already satisfy the convex sum-to-one constraint and, hence, the feasibility function g ends up being the identity function $g(\theta) = \theta$.

In particular, we start with some initial feasible estimates $\theta^t_{X|\mathbf{u}}$ at iteration $t = 0$, and then apply the following update equation until convergence:

$$\theta^{t+1}_{x|\mathbf{u}} = \frac{1}{N} \sum_{i=1}^{N} \frac{(C^i_{\mathbf{u}} + C^i_{x|\mathbf{u}}) \cdot \theta^t_{x|\mathbf{u}}}{C^i_{\mathbf{u}} + \sum_{x'} C^i_{x'|\mathbf{u}} \cdot \theta^t_{x'|\mathbf{u}}}$$

Note here that constants C^i are computed by inference on a Bayesian network structure under parameters θ^t (see Theorem 1 for the definitions of these constants).

4. EDML for Undirected Models

In this section, we show how parameter estimation for undirected graphical models, such as Markov networks and pairwise Markov random fields, can also be posed as an optimization problem, as described in Section 2.

For Markov networks, $\theta = (\ldots, \theta_{\mathbf{X}_a}, \ldots)$ is a vector over the network parameters. Each component $\theta_{\mathbf{X}_a}$ is a parameter set (or potential), assigning a number $\theta_{\mathbf{x}_a} \geq 0$ for each instantiation $\mathbf{x}_a$ of variables $\mathbf{X}_a$.[6]

The negative log-likelihood, for a Markov network is:

$$-\ell\ell(\theta|\mathcal{D}) = N \log Z_\theta - \sum_{i=1}^{N} \log Z_\theta(\mathbf{d}_i) \tag{5}$$

where Z_θ is the partition function, and where $Z_\theta(\mathbf{d}_i)$ is the partition function after conditioning on example $\mathbf{d}_i$, under parameterization θ.

Sub-functions with respect to Equation 5 may not be convex, as was the case in Bayesian networks. Consider instead the following objective function, which we shall subsequently relate to the negative log-likelihood:

$$f(\theta) = -\sum_{i=1}^{N} \log Z_\theta(\mathbf{d}_i), \tag{6}$$

with a feasibility constraint that the partition function Z_θ equals some constant α. The following theorem tells us that it suffices to optimize Equation 6 under the given constraint in order to optimize Equation 5.

Theorem 3 *Let α be a positive constant, and let $g(\theta)$ be a feasibility function defined such that $g(\theta_{\mathbf{x}_a}) \propto \theta_{\mathbf{x}_a}$ for all $\theta_{\mathbf{x}_a}$, and $Z_{g(\theta)} = \alpha$.[7] Point θ is stationary for the function of Equation 5 iff point $g(\theta)$ is feasible and stationary for the function of Equation 6.*

With Equation 6 as the new objective function for estimating the parameters of a Markov network, we can now cast its optimization in the terms of Section 2. We start by characterizing its sub-functions.

Theorem 4 *For a given parameter set $\theta_{\mathbf{X}_a}$, the objective function of Equation 6 has sub-functions:*

$$f_{\theta^\star}(\theta_{\mathbf{X}_a}) = -\sum_{i=1}^{N} \log \sum_{\mathbf{x}_a} C^i_{\mathbf{x}_a} \cdot \theta_{\mathbf{x}_a} \tag{7}$$

[5] More specifically, strict convexity implies a unique optimum, although under certain assumption, we can guarantee that Equation 2 is indeed strictly convex.

[6] One typically assumes an exponential representation $\theta_{\mathbf{x}_a} = \exp\{\tau_{\mathbf{x}_a}\}$ with meta-parameters $\tau_{\mathbf{x}_a}$. One can also use a feature-based representation, but for our purposes, a tabular representation suffices.

[7] Here, $g(\theta_{\mathbf{x}_a})$ denotes the component of $g(\theta)$ corresponding to $\theta_{\mathbf{x}_a}$. Moreover, the feasibility function $g(\theta)$ can be constructed, for example, by simply multiplying all entries of one parameter set by α/Z_θ. In our experiments, we normalize each parameter set to sum-to-one, but then update the constant $\alpha = Z_{\theta^t}$ for the subsequent iteration.

where $C^i_{\mathbf{x}_a}$ is a constant that is independent of the parameter set $\theta_{\mathbf{X}_a}$:

$$C^i_{\mathbf{x}_a} = Z_{\theta^\star}(\mathbf{x}_a, \mathbf{d}_i)/\theta^\star_{\mathbf{x}_a}$$

Note that computing this constant requires inference on the Markov network under parameters $\theta^\star$.[8]

Interestingly, this sub-function is convex and has a unique optimum, as in Bayesian networks. However, even when $\theta^\star$ is a feasible point, the unique optima of these sub-functions may not produce a feasible point when combined. Hence, the feasibility function $g(\theta)$ defined in Theorem 3 must be utilized in this case.

We now have another instance of the iterative algorithm proposed in Section 2, but for undirected graphical models. That is, we have just derived an EDML algorithm for such models.

5. EDML under Complete Data

We will now consider how EDML simplifies under complete data for both Bayesian and Markov networks. The key here is to identify the forms of corresponding sub-functions under complete data.

We start with Bayesian networks. Consider a variable X and a parent instantiation $\mathbf{u}$ in such a network. Let $\mathcal{D}\#(x\mathbf{u})$ represent the number of examples that contain $x\mathbf{u}$ in the complete dataset $\mathcal{D}$. Equation 2 of Theorem 1 then reduces to:

$$f_{\theta^\star}(\theta_{X|\mathbf{u}}) = -\sum_x \mathcal{D}\#(x\mathbf{u}) \log \theta_{x|\mathbf{u}} + C,$$

where C is a constant that is independent of parameter set $\theta_{X|\mathbf{u}}$. Assuming that $\theta^\star$ is feasible (i.e., satisfies the sum-to-one constraints), the unique optimum of this sub-function is $\theta_{x|\mathbf{u}} = \frac{\mathcal{D}\#(x\mathbf{u})}{\mathcal{D}\#(\mathbf{u})}$, which is guaranteed to yield a feasible point θ, globally. Hence, EDML produces the unique optimal estimates in its first iteration and terminates immediately thereafter.

The situation is different, however, for Markov Networks. In this case, and under a complete dataset $\mathcal{D}$, Equation 7 of Theorem 4 reduces to:

$$f_{\theta^\star}(\theta_{\mathbf{X}_a}) = -\sum_{\mathbf{x}_a} \mathcal{D}\#(\mathbf{x}_a) \log \theta_{\mathbf{x}_a} + C,$$

where C is a constant that is independent of parameter set $\theta_{\mathbf{X}_a}$. Assuming that $\theta^\star$ is feasible (i.e., satisfies $Z_{\theta^\star} = \alpha$), the unique optimum of this sub-function has the closed form[9]

$$\theta_{\mathbf{x}_a} = \frac{\alpha}{N} \frac{\mathcal{D}\#(\mathbf{x}_a)}{Z_{\theta^\star}(\mathbf{x}_a)/\theta^\star_{\mathbf{x}_a}}$$

which is equivalent to the unique optimum one would obtain in a sub-function for Equation 5 (Pietra et al., 1997; Murphy, 2012). Contrary to Bayesian networks, the collection of these optima for different parameter sets do not necessarily yield a feasible point θ. Hence, the feasibility function g of Theorem 3 must be applied here. The resulting feasible point, however, may no longer be a stationary point for the corresponding sub-functions, leading EDML to iterate further. Hence, under complete data, EDML for Bayesian networks converges immediately, while EDML for Markov networks may have to make multiple iterations.

Both results are consistent with what is already known in the literature on parameter estimation for Bayesian and Markov networks. The result on Bayesian networks is useful in confirming that EDML performs optimally in this case. The result for Markov networks, however, gives rise to a new algorithm for parameter estimation under complete data. We compare the performance of this new EDML algorithm to conjugate gradient descent in the next section.

6. Experimental Results

In this section, we illustrate the practical advantages of EDML, in comparison to popular general-purpose approaches such as the conjugate gradient method (CG), for the purposes of estimating parameters in Markov networks. We evaluated the EDML and CG algorithms by learning grid-structured pairwise MRFs from the CEDAR dataset of handwritten digits. This complete dataset consisted of 16x16 binary images, one for each digit from zero to nine. Experiments were run on a 3.6GHz Intel i5 CPU with access to 8GB RAM.

For CG, we made use of an open-source implementation provided by the Apache Commons Mathematics library,[10] which is a Java library. Our EDML implementation is also in Java, but more importantly, both CG and EDML rely on the same underlying engine for exact inference in Markov networks.[11]

For EDML, we damped our parameter estimates at each iteration, which is typical for algorithms like loopy belief propagation, which EDML was originally inspired by (Choi et al., 2011). In particular, we start

[8] Theorem 4 assumes tacitly that $\theta^\star_{\mathbf{x}_a} \neq 0$. More generally, however, $C^i_{\mathbf{x}_a} = \partial Z_{\theta^\star}(\mathbf{d}_i)/\partial\theta_{\mathbf{x}_a}$. See also Footnote 3.

[9] More generally, $\theta_{\mathbf{x}_a} = \frac{\alpha}{N} \frac{\mathcal{D}\#(\mathbf{x}_a)}{\partial Z_{\theta^\star}/\partial\theta_{\mathbf{x}_a}}$ (covers $\theta^\star_{\mathbf{x}_a} = 0$).

[10] Available at http://commons.apache.org/.

[11] We used the inference engine in the SamIam system, available at http://reasoning.cs.ucla.edu/samiam/.

Table 1. Speedup results Digits 8x8

digit	#iters CG	#iters EDML	iter S	time S
zero	897	1863	0.48	0.31
one	897	137	6.55	7.85
two	561	129	4.35	3.06
three	465	102	4.56	4.52
four	577	123	4.69	4.90
five	563	63	8.94	9.85
six	578	133	4.35	4.30
seven	568	118	4.81	5.30
eight	556	146	3.81	4.45
nine	567	122	4.65	5.01
average	**622.9**	**293.6**	**4.72**	**4.95**

Table 2. Speedup results Digits 16x16

digit	#iters CG	#iters EDML	iter S	time S
zero	45	165	0.27	3.87
one	104	99	1.05	15.22
two	46	177	0.26	3.63
three	43	115	0.37	5.20
four	56	179	0.31	4.38
five	43	150	0.29	4.18
six	48	101	0.48	6.76
seven	57	181	0.31	4.61
eight	48	193	0.25	3.77
nine	56	184	0.30	4.55
average	**54.6**	**154.4**	**0.39**	**5.62**

with an initial factor of 0.5, and adapt the factor dynamically, by halving it after each iteration when EDML does not improve its objective (adaptation is not started until after the first 5 iterations).

In our experiments, for each digit from zero to nine, we run CG until convergence to obtain parameter estimates of some quality q (i.e., in log likelihood), recording the number of iterations i_{cg} and time t_{cg} required by CG. EDML is then run, subsequently, until it obtains an estimate of the same quality q, or better, recording also the number of iterations i_{edml} and time t_{edml}. The iteration and time speed-ups S of EDML are computed as $\frac{i_{\text{cg}}}{i_{\text{edml}}}$, and $\frac{t_{\text{cg}}}{t_{\text{edml}}}$, respectively.

Tables 1 and 2 show the iteration and time speed-ups obtained for two datasets: 16×16 and 8×8 (downsampled) images of digits, respectively.[12] On average, we see that EDML was roughly $5\times$ faster in both sets, and was up to an order-of-magnitude faster in at least one case. In the 16×16 set, EDML required more iterations but was still $5.6\times$ faster in time. This is due in part by the number of times inference is invoked by CG (in line search), whereas EDML only needs to invoke inference once per iteration.

[12] For the 8×8 digits, we used a convergence threshold of 10^{-4} w.r.t. the absolute change in the parameter estimates. For the 16×16 digits, we used a more relaxed threshold based on relative change in the likelihood, also at 10^{-4}.

7. Related Work

As an iterative fixed-point algorithm, we can view EDML as a Jacobi-type method, where updates are performed in parallel (Bertsekas & Tsitsiklis, 1989). Correspondingly, a version of EDML, using Gauss-Seidel iterations, would update each parameter set in sequence using the most recently computed updates, which would lead to an algorithm that monotonically improves the log likelihood at each update. In this case, we obtain a coordinate descent algorithm, Iterative Proportional Fitting (IPF) (Jirousek & Preucil, 1995), as a special case of EDML, in the case where updates are performed in sequence.

The notion of fixing all parameters, except for one, has been exploited before for the purposes of optimizing the log likelihood of a Markov network, as a heuristic for structure learning (Pietra et al., 1997). This notion also underlies the IPF algorithm; see, e.g., (Murphy, 2012), Section 19.5.7. In the case of complete data, the resulting sub-function is convex, yet for incomplete data, it is not necessarily convex. In contrast, the sub-functions implied by Equation 6 are in fact convex, but yield equivalent stationary points in the case of both complete and incomplete data.

Optimization methods such as conjugate gradient, and L-BFGS (Liu & Nocedal, 1989), are more commonly used to optimize the parameters of a Markov network. For relational Markov networks or Markov networks that otherwise assume a feature-based representation (Domingos & Lowd, 2009), evaluating the likelihood is typically intractable, in which case one typically optimizes instead the pseudo-log-likelihood (Besag, 1975). For more on parameter estimation in Markov networks, see (Koller & Friedman, 2009; Murphy, 2012).

8. Conclusion

In this paper, we provided an abstract and simple view of the EDML algorithm, originally proposed for parameter estimation in Bayesian networks, as a particular method for continuous optimization. One consequence of this view is that it is immediate that fixed-points of EDML are stationary points of the log-likelihood, and vice-versa (Refaat et al., 2012). A more interesting consequence, is that it allows us to propose an EDML algorithm for a new class of models, Markov networks. Empirically, we find that EDML can find better parameter estimates for Markov networks under complete data, several times faster, compared to the conjugate gradient method. Empirical evaluation under incomplete data is left as future work.

References

Bertsekas, Dimitri P. and Tsitsiklis, John N. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

Besag, J. Statistical Analysis of Non-Lattice Data. *The Statistician*, 24:179–195, 1975.

Boyd, Stephen and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2004.

Chan, Hei and Darwiche, Adnan. On the revision of probabilistic beliefs using uncertain evidence. *AIJ*, 163:67–90, 2005.

Choi, Arthur and Darwiche, Adnan. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *AAAI*, pp. 1107–1114, 2006.

Choi, Arthur, Refaat, Khaled S., and Darwiche, Adnan. EDML: A method for learning parameters in Bayesian networks. In *UAI*, 2011.

Darwiche, Adnan. A differential approach to inference in Bayesian networks. *JACM*, 50(3):280–305, 2003.

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

Dempster, A.P., Laird, N.M., and Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Domingos, Pedro and Lowd, Daniel. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

Jirousek, Radim and Preucil, Stanislav. On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics & Data Analysis*, 19(2):177–189, 1995.

Koller, Daphne and Friedman, Nir. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Lauritzen, S. L. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

Liu, D. C. and Nocedal, J. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(3):503–528, 1989.

Murphy, Kevin Patrick. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Park, James and Darwiche, Adnan. A differential semantics for jointree algorithms. *AIJ*, 156:197–216, 2004.

Pietra, Stephen Della, Pietra, Vincent J. Della, and Lafferty, John D. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4): 380–393, 1997.

Refaat, Khaled S., Choi, Arthur, and Darwiche, Adnan. New advances and theoretical insights into EDML. In *UAI*, pp. 705–714, 2012.

A. Proofs

Proof of Theorem 1 First, the probability of an example $\mathbf{d}_i \in \mathcal{D}$ is: $Pr_\theta(\mathbf{d}_i) = \sum_{\mathbf{x}\sim\mathbf{d}_i} \prod_{x|\mathbf{u}\sim\mathbf{x}} \theta_{x|\mathbf{u}}$ where operator $\sim$ denotes compatibility between two instantiations (they set the same value to common variables). For a fixed parameter set $\theta_{X|\mathbf{u}}$, the probability $Pr_\theta(\mathbf{d}_i)$ is a linear function with respect to the parameters of $\theta_{X|\mathbf{u}}$:

$$\begin{aligned} Pr_\theta(\mathbf{d}_i) &= Pr_\theta(\neg\mathbf{u}, \mathbf{d}_i) + \sum_x Pr_\theta(x\mathbf{u}, \mathbf{d}_i) \\ &= Pr_\theta(\neg\mathbf{u}, \mathbf{d}_i) + \sum_x \frac{\partial Pr_\theta(\mathbf{d}_i)}{\partial \theta_{x|\mathbf{u}}} \theta_{x|\mathbf{u}} \\ &= C^i_{\mathbf{u}}[\theta] + \sum_x C^i_{x|\mathbf{u}}[\theta] \cdot \theta_{x|\mathbf{u}} \end{aligned}$$

where $C^i_{\mathbf{u}}[\theta]$ and $C^i_{x|\mathbf{u}}[\theta]$ are constants with respect to $\theta_{X|\mathbf{u}}$. Moreover $Pr_\theta(\neg\mathbf{u}, \mathbf{d}_i) = Pr_\theta(\mathbf{d}_i) - Pr_\theta(\mathbf{u}, \mathbf{d}_i)$. Thus our sub-function, the negative log-likelihood with respect to parameter set $\theta_{X|\mathbf{u}}$, has the form:

$f_{\theta^\star}(\theta_{X|\mathbf{u}}) = -\sum_{i=1}^N \log\Big(C^i_{\mathbf{u}}[\theta^\star] + \sum_x C^i_{x|\mathbf{u}}[\theta^\star] \cdot \theta_{x|\mathbf{u}}\Big)$. □

Proof of Theorem 2 The log-likelihood of soft evidence in this model is:

$$\begin{aligned} \log \mathbb{P}(\eta|\theta_{X|\mathbf{u}}) &= \sum_{i=1}^N \log \mathbb{P}(\eta_i|\theta_{X|\mathbf{u}}) \\ &= \sum_{i=1}^N \log \sum_x \mathbb{P}(\eta_i|x, \theta_{X|\mathbf{u}}) \mathbb{P}(x|\theta_{X|\mathbf{u}}) \\ &= \sum_{i=1}^N \log \sum_x \mathbb{P}(\eta_i|x) \cdot \theta_{x|\mathbf{u}}. \end{aligned}$$

If we substitute $\mathbb{P}(\eta_i|x) = C^i_{\mathbf{u}}[\theta^\star] + C^i_{x|\mathbf{u}}[\theta^\star]$, we have

$$\begin{aligned} \log \mathbb{P}(\eta|\theta_{X|\mathbf{u}}) &= \sum_{i=1}^N \log \sum_x \Big(C^i_{\mathbf{u}}[\theta^\star] + C^i_{x|\mathbf{u}}[\theta^\star]\Big) \cdot \theta_{x|\mathbf{u}} \\ &= \sum_{i=1}^N \log \Big(C^i_{\mathbf{u}}[\theta^\star] + \sum_x C^i_{x|\mathbf{u}}[\theta^\star] \theta_{x|\mathbf{u}}\Big) \end{aligned}$$

which is Equation 2, negated. □

Proof of Theorem 3 The partial derivative of the log likelihood $\ell\ell(\theta|\mathcal{D})$ w.r.t. parameter $\theta_{\mathbf{x}_a}$ is:

$$\frac{\partial \ell\ell}{\partial \theta_{\mathbf{x}_a}} = -\frac{N}{Z_\theta} \frac{\partial Z_\theta}{\partial \theta_{\mathbf{x}_a}} + \sum_{i=1}^N \frac{1}{Z_\theta(\mathbf{d}_i)} \frac{\partial Z_\theta(\mathbf{d}_i)}{\partial \theta_{\mathbf{x}_a}}.$$

First, note that:

$$\tfrac{1}{Z_\theta} \tfrac{\partial Z_\theta}{\partial \theta_{\mathbf{x}_a}} \theta_{\mathbf{x}_a} = Pr(\mathbf{x}_a),\ \tfrac{1}{Z_\theta(\mathbf{d}_i)} \tfrac{\partial Z_\theta(\mathbf{d}_i)}{\partial \theta_{\mathbf{x}_a}} \theta_{\mathbf{x}_a} = Pr_\theta(\mathbf{x}_a|\mathbf{d}_i)$$

Thus, with some re-arranging, we obtain:

$$Pr_\theta(\mathbf{x}_a) = \frac{1}{N} \sum_{i=1}^N Pr_\theta(\mathbf{x}_a|\mathbf{d}_i) \tag{8}$$

which is the "moment matching" condition for parameter estimation in Markov networks. Second, consider the simplified objective: $f(\theta) = -\sum_{i=1}^N \log Z_\theta(\mathbf{d}_i)$ which is subject to the constraint $Z = \alpha$. We construct the Lagrangian $L(\theta, \nu) = f(\theta) + \nu(Z - \alpha)$. Setting to zero the partial derivative w.r.t. ν, we obtain our constraint $Z = \alpha$. The partial derivative w.r.t. parameter $\theta_{\mathbf{x}_a}$ is:

$$-\sum_{i=1}^N \frac{1}{Z_\theta(\mathbf{d}_i)} \frac{\partial Z_\theta(\mathbf{d}_i)}{\partial \theta_{\mathbf{x}_a}} + \nu \frac{\partial Z_\theta}{\partial \theta_{\mathbf{x}_a}}.$$

We set the partial derivative to zero, multiply the second term by $\frac{\alpha}{Z} = 1$, and re-arrange, giving us:

$$\nu\alpha Pr_\theta(\mathbf{x}_a) = \sum_{i=1}^N Pr_\theta(\mathbf{x}_a \mid \mathbf{d}_i).$$

Summing each equation for all instantiations $\mathbf{x}_a$, we identify $\nu = \frac{N}{\alpha}$, which after substituting, gives us a condition equivalent to Equation 8.

Note that the stationary condition given by Equation 8 depends only on marginals, not the absolute value of the partition function. Moreover, applying a proper feasibility function $g(\theta)$ will not change the marginals implied by θ, as the multiplicative factors cancel out in each pair of terms, $\log Z_\theta - \log Z_\theta(\mathbf{d}_i)$. Thus if a point θ satisfies Equation 8, then $g(\theta)$ must also satisfy it. Similarly, if $g(\theta)$ satisfies Equation 8, the original point θ must also satisfy it. □

Proof of Theorem 4 First, the partition function conditioned on an example $\mathbf{d}_i \in \mathcal{D}$ is:

$$Z_\theta(\mathbf{d}_i) = \sum_{\mathbf{x}\sim\mathbf{d}_i} \prod_{\mathbf{x}_a\sim\mathbf{x}} \theta_{\mathbf{x}_a}$$

where operator $\sim$ denotes compatibility between two instantiations (they set the same value to common variables). For a fixed parameter set $\theta_{\mathbf{X}_a}$, the partition function $Z_\theta(\mathbf{d}_i)$ is a linear function with respect to the parameters $\theta_{\mathbf{X}_a}$:

$$\begin{aligned} Z_\theta(\mathbf{d}_i) &= \sum_{\mathbf{x}_a} Z_\theta(\mathbf{x}_a, \mathbf{d}_i) = \sum_{\mathbf{x}_a} \frac{\partial Z_\theta(\mathbf{d}_i)}{\partial \theta_{\mathbf{x}_a}} \theta_{\mathbf{x}_a} \\ &= \sum_{\mathbf{x}_a} C^i_{\mathbf{x}_a}[\theta] \cdot \theta_{\mathbf{x}_a} \end{aligned}$$

where $C^i_{\mathbf{x}_a}[\theta]$ is a constant with respect to $\theta_{\mathbf{X}_a}$. Thus, our sub-function, has the form:

$$f_{\theta^\star}(\theta_{\mathbf{X}_a}) = -\sum_{i=1}^N \log \sum_{\mathbf{x}_a} C^i_{\mathbf{x}_a}[\theta^\star] \cdot \theta_{\mathbf{x}_a}.$$ □

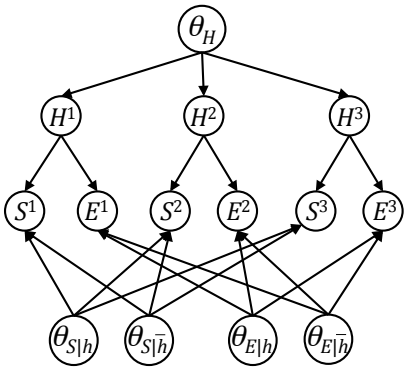

Figure 2. A meta network induced from a base network $S \longleftarrow H \longrightarrow E$. The CPTs here are based on standard semantics; see, e.g., (Darwiche, 2009), Ch. 18.

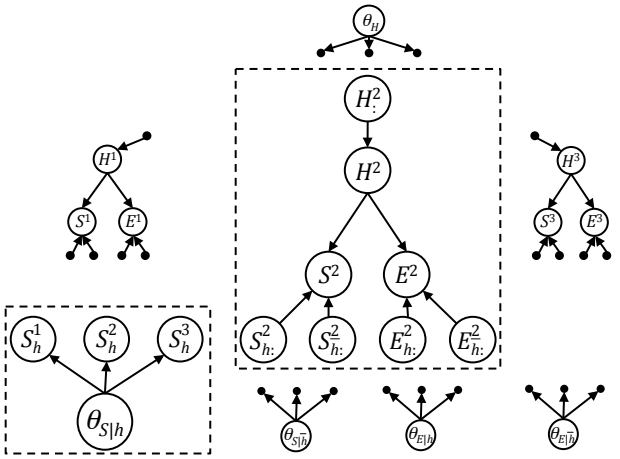

Figure 3. An edge-deleted network obtained from the meta network in Figure 2. Highlighted are the island for example $\mathbf{d}_2$ and the island for parameter set $\theta_{S|h}$.

Theorem 5 *Suppose we have a feasibility function*

$$g(y_1, \ldots, y_n) = (x_1, \ldots, x_n)$$

where $x_i \neq y_i$ implies that the point $(x_1, \ldots, y_i, \ldots, x_n)$ is infeasible (e.g., Euclidean projection satisfies this condition). Suppose now that the algorithm produces a sequence $x^t, y^{t+1}, x^{t+1} = x^t$. Then x^t must be a feasible and stationary point.

Proof By the statement of the iterative procedure, x^t is guaranteed to be feasible. Suppose that $g(y^{t+1}) = x^{t+1} = x^t$. First, it must be that $y^{t+1} = x^t$. Suppose instead that $y^{t+1} \neq x^t$, and thus for some component, $y_i^{t+1} \neq x_i^t$. By our feasibility function, $(x_1^t, \ldots, y_i^{t+1}, \ldots, x_n^t)$ must be infeasible. However, Step 2(a) ensures that $(x_1^t, \ldots, y_i^{t+1}, \ldots, x_n^t)$ is feasible. Hence, it must be that $y^{t+1} = x^t$. Further, by Step 2(a) and Claim 1, x^t must also be stationary. □

B. A Review of EDML

EDML is a recent method for learning Bayesian network parameters from incomplete data (Choi et al., 2011; Refaat et al., 2012). It is based on Bayesian learning in which one formulates estimation in terms of computing posterior distributions on network parameters. That is, given a Bayesian network, one constructs a corresponding *meta network* in which parameters are explicated as variables, and on which the given dataset $\mathcal{D}$ can be asserted as evidence; see Figure 2. One then estimates parameters by considering the posterior distribution obtained from conditioning the meta network on the given dataset $\mathcal{D}$. Suppose for example that the meta network induces distribution $\mathcal{P}$ and let θ denote an instantiation of variables that represent parameters in the meta network. One can then obtain MAP parameter estimates by computing $\text{argmax}_\theta \mathcal{P}(\theta|\mathcal{D})$ using inference on the meta network.

It is known that meta networks tend to be too complex for exact inference algorithms, especially when the dataset is large enough. The basic insight behind EDML was to adapt a specific approximate inference scheme to meta networks with the goal of computing MAP parameter estimates. In particular, the original derivation of EDML adapted the approximate inference algorithm proposed by (Choi & Darwiche, 2006), in which edges are deleted from a Bayesian network to make it sparse enough for exact inference, followed by a compensation scheme that attempts to improve the quality of the approximations obtained from the edge-deleted network. The adaptation of this inference method to meta networks is shown in Figure 3. The two specific techniques employed here were to augment each edge $\theta_{X|\mathbf{u}} \longrightarrow X^i$ by an auxiliary variable $X^i_{\mathbf{u}}$, leading to $\theta_{X|\mathbf{u}} \longrightarrow X^i_{\mathbf{u}} \longrightarrow X^i$, where $X^i_{\mathbf{u}} \longrightarrow X^i$ is an equivalence edge. This is followed by deleting the equivalence edge. This technique yielded a disconnected meta network with two classes of subnetworks, called *parameter islands* and *network islands*.

Deleting edges, as proposed by (Choi & Darwiche, 2006), leads to introducing two auxiliary nodes in the Bayesian network for each deleted edge. Moreover, approximate inference by edge deletion follows the deletion process by a compensation scheme that searches for appropriate CPTs of these auxiliary nodes. As it turns out, the search for these CPTs, which is done iteratively, was amenable to a very intuitive interpretation as shown in (Choi et al., 2011).

Algorithm 1 EDML

input:

G:	A Bayesian network structure
$\mathcal{D}$:	An incomplete dataset $\mathbf{d}_1, \ldots, \mathbf{d}_N$
θ:	An initial parameterization of structure G
$\alpha_{X\|\mathbf{u}}, \beta_{X\|\mathbf{u}}$:	Beta prior for each random variable $X\|\mathbf{u}$

1: **while** not converged **do**

2: $Pr \leftarrow$ distribution induced by θ and G

3: **Compute** Bayes factors:

$$\kappa^i_{x|\mathbf{u}} \leftarrow \frac{Pr(x\mathbf{u}|\mathbf{d}_i)/Pr(x|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1}{Pr(\bar{x}\mathbf{u}|\mathbf{d}_i)/Pr(\bar{x}|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1}$$

for each family instantiation $x\mathbf{u}$ and example $\mathbf{d}_i$

4: **Update** parameters:

$$\theta_{x|\mathbf{u}} \leftarrow \operatorname*{argmax}_p [p]^{\alpha_{X|\mathbf{u}}-1}[1-p]^{\beta_{X|\mathbf{u}}-1} \prod_{i=1}^{N} [\kappa^i_{x|\mathbf{u}} \cdot p - p + 1]$$

5: **return** parameterization θ

In particular, one set of CPTs corresponded to soft evidence on network parameters, where each network island contributes one piece of soft evidence for each network parameter. The second set of CPTs corresponded to updated parameter estimates, where each parameter island contributes an estimate of its underlying parameter set. This interpretation was the basis for the form of EDML shown in Algorithm 1.[13] EDML iterates just like EM does, producing new estimates after each iteration. However, EDML iterations can be viewed as having two phases. In the first phase, each example in the data set is used to compute a piece of soft evidence on each parameter set (Line 3 of Algorithm 1). In the second phase, the pieces of soft evidence pertaining to each parameter set are used to compute a new estimate of that set (by solving the convex optimization problem on Line 4 of Algorithm 1). The process repeats until some convergence criteria is met. Aside from this optimization task, EM and EDML have the same computational complexity.

[13] This form is specific to binary variables; a multi-valued generalization was provided in (Refaat et al., 2012).