An Efficient Method for Probabilistic Knowledge Integration*

Shenyong Zhang[1,2], Yun Peng[2], and Xiaopu Wang[1]
[1]Department of Astronomy and Applied Physics
University of Science and Technology of China, Hefei, Anhui 230026
[2]Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
{syzhang, ypeng}@umbc.edu, wxp@ustc.edu.cn

Abstract

This paper presents an efficient method, SMOOTH, for modifying a joint probability distribution to satisfy a set of inconsistent constraints. It extends the well-known "iterative proportional fitting procedure" (IPFP), which only works with consistent constraints. Comparing with existing methods, SMOOTH is computationally more efficient and insensitive to data. Moreover, SMOOTH can be easily integrated with Bayesian networks for Bayes reasoning with inconsistent constraints.

1. Introduction

In this paper we consider the problem of modifying a joint probability distribution (JPD) to satisfy a set of low-dimensional distributions called *constraints*. We discuss this problem in the context of probabilistic knowledge integration [7, 8] where the JPD is considered as a probabilistic knowledge base (PKB) and the constraints represent knowledge from other sources. Similar problems also exist in probabilistic reasoning where constraints are taken as evidences to alter our beliefs (or posteriors) of other variables [5].

Let $V = \{1, 2, \ldots, n\}$, X^V a set of discrete random variables for a chosen domain, and $Q_{(0)}$ the initial JPD over X^V. Also let $\mathsf{P} = \{P_1, P_2, \ldots, P_k\}$ be a set of constraints where each P_j is a low-dimensional PD over variables $X^{E_j}, E_j \subseteq V$. The goal is to integrate P into $Q_{(0)}$.

The *property set*, $S_j = \{Q \mid Q^{E_j} = P_j\}$, is the set of all JPDs over X^V that satisfy constraint $P_j \in \mathsf{P}$, and the *solution set*, $S = \bigcap_{j=1}^{k} S_j$, is the set of all JPDs that satisfy all constraints in P. Then P is consistent if $S \neq \varnothing$, and inconsistent otherwise.

* This work was supported in part by NIST award 60NANB6D6206 and the China Scholarship Council (CSC).

Integrating consistent constraints into $Q_{(0)}$ can be accomplished by the well-known algorithm *iterative proportional fitting procedure* (IPFP) [4, 7]. The procedure converges to a JPD in S that minimizes the distance to $Q_{(0)}$ (measured by Kullback-Leibler distance, also known as I-divergence). However, when P is inconsistent, we cannot use IPFP to compute the integrated JPD because in this situation IPFP does not converge but rather goes into cycles over points in S_j [2, 3, 8].

Vomlel has proposed two methods that extend IPFP to deal with inconsistent constraints. The first one, GEMA [8], generalizes the concept of expectation maximization, and the second one, CC-IPFP [3], is based on convex mixing of distributions generated by the standard IPFP. The time performance of GEMA is extremely sensitive to both the initial JPD and the constraints, and CC-IPFP is often very slow to converge. Also, these two algorithms directly manipulate full joint distributions, and thus can not take advantages of other probabilistic models such as the Bayesian networks (BNs).

To address these limitations, we propose a new algorithm called SMOOTH which works with both consistent and inconsistent constraints. Before presenting this new method in Section 3, we provide a brief background in Section 2. Section 4 reports the experiment results and compares SMOOTH with other methods. Section 5 concludes with directions for future research.

2. IPFP and Methods for Knowledge Integration with Inconsistent Input

Iterative fitting proportional procedure (IPFP) was first published in [4] in 1937. Its convergence proof was given in [1] based on I-divergence geometry. The main idea of IPFP is as follows.

Given a JPD $Q_{(0)}$ and a set of consistent constraints $\mathsf{P} = \{P_1, P_2, \ldots, P_k\}$, after iteratively modi-

fying the JPD using Eq. (1), IPFP converges to the limit JPD Q^* which is an I-projection of $Q_{(0)}$ on the solution set S, i.e. among all $Q \in S$, Q^* has the smallest I-divergence (or KL distance) to $Q_{(0)}$.

$$Q_{i\cdot k+j}(x) = Q_{i\cdot k+j-1}(x)\frac{P_j(x^{E_j})}{Q^{E_j}_{i\cdot k+j-1}(x^{E_j})} \quad (1)$$

where $i = 1, 2, \ldots$ is the iteration index.

For clarity, in the rest of this paper we use $\pi(Q, S_j)$ (borrowed from [7]) to denote the I-projection of Q on S_j. It can be shown that $Q_{i\cdot k+j} = \pi(Q_{i\cdot k+j-1}, S_j)$ in Eq. (1). In other words, IPFP iteratively projects the current JPD on the property set of the next constraint. We call each of these projections a *fitting step* since it modifies the current JPD to fit the next constraint.

To deal with inconsistent constraints, GEMA takes two steps in each iteration. Take the i^{th} iteration that starts with JPD $Q_{(i-1)\cdot k}$ as an example. In Step 1, it first uses Eq. (1) to compute k I-projections $\pi(Q_{(i-1)\cdot k}, S_j)$ for each of the k constraints P_j in P, and then takes a weighted sum of these k I-projections to obtain a distribution

$$\tilde{Q}_{i\cdot k} = \sum_{j=1}^{k} w_j \pi(Q_{(i-1)\cdot k}, S_j),$$

where $w_j \in (0,1), \Sigma_{j=1}^{k} w_j = 1$. In Step 2, GEMA first computes k marginals $\tilde{P}_j$ (with domain X^{E_j}) from $\tilde{Q}_{i\cdot k}$, then performs the standard IPFP on $Q_{(i-1)\cdot k}$ using all of these k $\tilde{P}_j$ as constraints to obtain $Q_{i\cdot k}$. It has been shown that GEMA converges to a distribution which has a minimum I-aggregate Ψ, the weighted sum of I-divergence to all of the constraints in P:

$$\Psi(Q \parallel P_1, \ldots, P_k) = \sum_{j=1}^{k} w_j I(P_j \parallel Q^{E_j}).$$

CC-IPFP, another algorithm proposed by Vomlel in [10], takes a different approach. At each iteration, it first computes the I-projection on S_j, then mixes it with the current JPD using Eq. (2)

$$Q_i = (1-\lambda_i)Q_i + \lambda_i Q_i \frac{P_j}{Q_i^{E_j}}. \quad (2)$$

where $j = (i-1) \bmod k + 1$, and λ_i monotonically decreases towards 0. It was conjectured that this method would converge to a distribution that minimizes the sum of total variations to all constraints.

Experiments (see Tables 2 - 4 in Section 4) show that the performance of GEMA is very sensitive to the data. For some combinations of $Q_{(0)}$ and P it converges within a few hundreds of iterations, but for other combinations, millions of iterations are needed. On the other hand, CC-IPFP converges uniformly but very slowly.

3. SMOOTH

One thing in common for both GEMA and CC-IPFP is that they only modify the JPDs while keeping the constraints unchanged through the iterations. Alternatively, one can make the modification *bi-directional*: at each iteration, not only the JPDs are pulled closer to the constraints but also the constraints are pulled towards the JPDs. By doing so, the inconsistency among the constraints is gradually reduced, which may lead to a faster and more stable convergence. Based on this idea we developed our new method SMOOTH.

3.1. The Algorithm SMOOTH

SMOOTH consists of two phases. Phase 1 performs the standard IPFP using all of the original constraints in P. It stops when the process converges (for consistent constraints) or starts to cycle (for inconsistent constraints). Phase 2 follows Eqs. (3) and (4) below. It differs from Phase 1 in that, before doing a standard IPFP fitting step, SMOOTH first modifies constraint $P_{(j,i-1)}$ to pull it closer to the current JPD.

$$P_{(j,i)} = (1-\alpha)P_{(j,i-1)} + \alpha Q^{E_j}_{(i-1)\cdot k+j-1} \quad (3)$$

where $\alpha \in (0,1)$. Then the current JPD is modified by the new constraint generated by Eq. (4).

$$Q_{(i-1)\cdot k+j} = Q_{(i-1)\cdot k+j-1}\frac{P_{(j,i)}}{Q^{E_j}_{(i-1)\cdot k+j-1}}. \quad (4)$$

From Eq. (3) we can see that the modified constraint $P_{(j,i)}$ is a mixture of the previous constraint $P_{(j,i-1)}$ and the marginal of the current JPD $Q^{E_j}_{(i-1)\cdot k+j-1}$.

Note that the distance between $Q^{E_j}_{(i-1)\cdot k+j-1}$ and P_j is gradually reduced to 0 by IPFP when the constraints are consistent. However, for inconsistent constraints, this distance cannot be further reduced by IPFP at the end of Phase 1. Since $Q^{E_j}_{(i-1)\cdot k+j-1}$ is the result of applying all other constraints, we can use its distance to P_j as a measure of inconsistency in the constraints. Since Eq. (3) pulls $P_{(j,i-1)}$ closer to $Q^{E_j}_{(i-1)\cdot k+j-1}$ it reduces or smoothes the degree of inconsistency among constraints. This is the reason we call our algorithm SMOOTH.

We call α in Eq. (3) the *smooth rate* because it controls the speed of smoothing. Note that Eq. (3) pulls $P_{(j,i)}$ away from $P_{(j,i-1)}$, causing some information loss of the original constraints. To minimize the information loss before it gets a chance to be integrated, α is set to a very small value.

The algorithm SMOOTH is shown below.

Algorithm SMOOTH. Consider an initial JPD $Q_{(0)}$ over domain X^V and a set of low-dimensional distributions $\mathsf{P} = \{P_1, P_2, \ldots, P_k\}$ where each $P_j \in \mathsf{P}$ has domain X^{E_j}, $E_j \subset V$. SMOOTH consists of the following two phases:

Phase 1: do the standard IPFP using all constraints in P until process converges or goes into cycles;

if convergence is reached **then exit**;

Phase 2:

for $j = 1, 2, \ldots, k$ $P_{(j,0)} = P_j$;
for $i = 1, 2, \ldots$ {
  if convergence is reached **then exit**;
  for $j = 1, 2, \ldots, k$ {

$$P_{(j,i)} = (1-\alpha)P_{(j,i-1)} + \alpha Q^{E_j}_{(i-1)\cdot k+j-1};$$

$$Q_{(i-1)\cdot k+j} = Q_{(i-1)\cdot k+j-1} \frac{P_{(j,i)}}{Q^{E_j}_{(i-1)\cdot k+j-1}};$$

}} □

Note that in Eq. (3), the update of the constraint uses the marginal of the current JPD, which is a low dimensional distribution. This is in contrast to the process of generating consistent constraints in Step 2 of GEMA which involves expensive operations of summing up full JPDs. In addition, note that SMOOTH is exactly the same as the standard IPFP except it uses a different constraint each time, and that the computation of Eq. (3) for updating constraints is local and can be done efficiently by BN procedures. This makes SMOOTH directly applicable to BNs using the IPFP based algorithms we developed earlier for BNs [2, 5].

3.2. Accelerating the Convergence with Incremental Smooth Rate

Experiments show that the iterative process of SMOOTH moves towards the convergence point fairly fast at the beginning, even with a small α. However, the process slows down drastically at the end, forming a long and flat tail. As discussed in Section 3.1, keeping α small at the beginning ensures information in the original constraints not to be lost too soon. When the process gets closer to the convergence point, we can use larger α since information of the original constraint has largely been absorbed. So we suggest a sigmoid function for increasing α:

$$\alpha = \frac{1}{1 + \exp(A - i/B)} \qquad (5)$$

where i is the iteration steps of Phase 2, A and B are constants. It can be seen that $\alpha \approx 0$ at the beginning and $\alpha \to 0$ when $i \to \infty$. Parameter A controls how long α is to remain small and B controls how fast α increases in the middle.

4. Experiments and Results

We have conducted computer experiments with different initial JPDs and constraints. All experiments were run on an Intel® Core™2 CPU of 2.40G Hz and 2.0G maximum memory for the JVM (Java Virtual Machine).

The following algorithms are compared experimentally: 1) GEMA, 2) CC-IPFP, 3) SMOOTH, 4) Accelerated SMOOTH (A-SMOOTH for short). For CC-IPFP, we use $\lambda_i = 1/(1+i)$ in Eq. (2), which is suggested by the authors [3]. For SMOOTH we set $\alpha = 0.01$ in Phase 2, and for A-SMOOTH, we set $A = 4.595$ ($\alpha_0 \approx 0.01$) and $B = 150$.

For all experiments, convergence is reached when the total variation $\Sigma \mid Q_{i\cdot k+j} - Q_{i\cdot k+j-1} \mid$ for all j are within the given error bound 10^{-12}. The number of fitting steps is used to measure the time performance of an algorithm because it is the core operation for all algorithms we experimented.

Experiment 1 uses the data taken from [3]. The initial JPD1 is a uniform distribution of three variables X_1, X_2, X_3. Three constraints, each a distribution of two variables, are generated according to the scheme in Table 1. They are consistent with each other when $\varepsilon = 4/20$ (called CONS0), inconsistent when $\varepsilon = 3/20$ (called CONS1).

Table 1. Input constraints

$P_j, j = 1,2$	$X_{j+1} = 0$	$X_{j+1} = 1$
$X_j = 0$	$1/2 - \varepsilon$	ε
$X_j = 1$	ε	$1/2 - \varepsilon$
P_3	$X_3 = 0$	$X_3 = 1$
$X_1 = 0$	ε	$1/2 - \varepsilon$
$X_1 = 1$	$1/2 - \varepsilon$	ε

The experiment results for the consistent constraints are given in Table 2 below. All three algorithms have converged to the same JPD. SMOOTH is significantly faster than the other two (84 fitting steps compared to 1164 for GEMA and 3507 for CC-IPFP). This is because SMOOTH is reduced to the standard IPFP (only executed Phase 1) for consistent constraints.

Table 2. Results for CONS0 ($\varepsilon = 4/20$)

Algorithm	**GEMA**	**CC-IPFP**	**SMOOTH**
Fitting steps	1164	3507	**84**
I-divergence	0.10453816	0. 10453816	0. 10453816

Experiment 2 compares performance with inconsistent constraints CONS1. Besides JPD1, another initial joint distribution JPD2 is also used. The initial I-aggregates for JPD1 and JPD2 are 0.11870910 and 0.34439004, respectively. The experiment results are given in Table 3. Fitting steps for SMOOTH and A-SMOOTH are given as the sum of Phase1 and Phase2.

Table 3. Results for inconsistent CONS1 ($\varepsilon = 3/20$)

	Fitting steps	I-divergence	I-aggregate
GEMA			
JPD-1	7,744,446	0.41502431	0.00367169
JPD-2	9,064,080	0.71979040	0.05727919
CC- IPFP			
JPD-1	>10,000,000	0.37048603	0.00461839
JPD-2	>10,000,000	0.70127029	0.05742945
SMOOTH			
JPD-1	177+3825	0.41503774	0.00367172
JPD-2	129+4899	0.71306584	0.05729201
A-SMOOTH			
JPD-1	177+375	0.41503891	0.00367227
JPD-2	129+402	0.71439294	0.05729532

Both SMOOTH and A-SMOOTH converge at a rate several orders of magnitude faster than the other two, and the resulted limit JPDs all have I-aggregates very close to that of GEMA. On the other hand, CC-IPFP does not converge within the max fitting step limit (10 million). The final JPD obtained using CC-IPFP has larger I-aggregates but smaller I-divergences than results from the others, indicating that CC-IPFP does not integrate the features of the initial constraints as much as the others.

To see that GEMA is data sensitive, we generated another set of 3 constraints (CONS2) which, unlike CONS1, is pair-wise inconsistent. The results using CONS2 with JPD1 and JPD2 are given in Table 4. It can be seen from Tables 3 and 4 that GEMA is very slow in all cases except one (JPD1 + CONS2). Similar phenomena have also been observed in some of our other experiments. On the other hand, both versions of SMOOTH have uniform performance for all combinations.

We have also experimented with larger JPD with 8 and 15 variables and varying number of constraints. The results are consistent with those reported above for smaller JPD.

Table 4. Fitting steps with CONS2

Algorithm	GEMA	CC-IPFP	SMOOTH
JPD1	780	>10,000,000	54+3405
JPD2	12,400,542	>10,000,000	216+3933

5. Conclusions

We propose a new method SMOOTH for modifying a given JPD with a set of constraints of low dimensional distributions. The main innovation of SMOOTH lies in the *bi-directional updates* of both JPD and constraints in dealing with inconsistent constraints. Experiments with different JPDs and constraints have confirmed that our new algorithms always converge, and the limit JPDs they reach always minimize the I-aggregate.

Comparing to existing methods such as GEMA and CC-IPFP, SMOOTH has the following advantages. 1) It works for both consistent and inconsistent constraint sets. 2) It enjoys a superior time performance. 3) It is not data-sensitive. 4) It can be directly applied to belief update in BN.

Several issues require further investigation. The first is to formally establish the convergence of SMOOTH, which we only verified empirically through experiments. The second is to incorporate the degree of confidence one has on individual constraints (consistent or inconsistent) in knowledge integration. The third is to extend this work to BN learning and structure update using low dimensional distributions.

6. References

[1] I. Csiszar, "I-divergence Geometry of Probability Distributions and Minimization Problems", *The Annuals of Probability*, Feb. 1975, 3(1), pp. 146-158.

[2] Z. Ding, Y. Peng, and R. Pan, "A Bayesian Approach to Uncertainty Modeling in OWL Ontology", in *Proceedings of the Int.l Conf. on Advances in Intelligent Systems – Theory and Applications*, Luxembourg, Nov. 15-18, 2004,.

[3] Jirousek R., Vomlel J., "Inconsistent Knowledge Integration in a Probabilistic Model", Proc. of workshop "*Mathematical Models for handling partial knowledge in A.I.*", Plenum Publ. Corp, New York, 1995.

[4] R. Kruithof, *Telefoonverkeersrekening*, De Ingeni-eur 52, E15-E25, 1937.

[5] R. Pan, Y. Peng, and Z. Ding, "Belief Update in Bayesian Networks Using Uncertain Evidence", *18th IEEE International Conference on Tools with Artificial Intelligence*, Nov. 2006, pp. 441-444.

[6] Y. Peng, Z. Ding, "Modifying Bayesian Networks by Probability Constraints", in *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, July 26-29, 2005.

[7] Vomlel J., "Methods of Probabilistic Knowledge Integration", PhD Thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, December 1999.

[8] Vomlel J., "Integrating Inconsistent Data in a Probabilistic Model", *Journal of Applied Non-Classical Logics*, 2003, pp. 1 – 20.