The Local Inconsistency Resolution Algorithm

Oliver Richardson [1]

Abstract

We present a generic algorithm for learning and approximate inference across a broad class of statistical models, that unifies many approaches in the literature. Our algorithm, called local inconsistency resolution (LIR), has an intuitive epistemic interpretation. It is based on the theory of probabilistic dependency graphs (PDGs), an expressive class of graphical models rooted in information theory, which can capture inconsistent beliefs.

1. Introduction

What causes a person to change their mind? According to some, it is a response to internal conflict: the result of discovering new information that contradicts our beliefs, or becoming aware of discrepancies between beleifs we already hold (**?**). Inconsistencies can be difficult to detect, however (**?**), and indeed can only be resolved once we are aware of them. Some things are also beyond our control; for example, we might recieve conflicting information from two trusted sources and be unable to resolve their disagreement. So in practice, we resolve inconsistencies *locally*—little by little, and looking at only a small part of the picture at a time.

This can have externalities; fixing one inconsistency can easily create others out of view. Furthermore, some inconsistencies are not local in nature, and can only be seen when considering many components at once. Yet despite its imperfections, this process of locally resolving inconsistency can be quite useful. As we shall soon see, it is a powerful recipe for learning and approximate inference. We formalize the process in the language of probability and convex optimization, and show how that many popular techniques in the literature arise naturally as instances of it.

Our approach leans heavily on the theory of Probabilistic Dependency Graphs (PDGs), which are very flexible graphical models that allow for arbitrary—even inconsistent—probabilistic information, weighted by confidence (Richardson & Halpern, 2021). There is a natural way to measure how inconsistent a PDG is, and many standard loss functions can be viewed as measuring the inconsistency of a PDG that describes the appropriate situation (Richardson, 2022). We introduce an algorithm to operationalize the process of adjusting parameters to resolve this inconsistency.

In general, even just calculating a PDG's degree of inconsistency is intractable. Much of variational inference can be understood as adopting extra beliefs to minimize an overapproximation of it that is easier to calculate (Richardson, 2022). Our approach can capture this, but also enables the opposite: focusing on small parts of the graph at a time to address tractable underapproximations of the global inconsistency. This makes it more suitable for distributed settings, and more amenable to parallelization. The algorithm, which we call *local inconsistency resolution* (LIR), is quite expressive, and naturally reduces to a wide variety of learning and inference algorithms in the literature. This observation suggests a generic approach to learning and inference in models with arbitrary structure.

2. Mathematical Preliminaries

We write $\mathcal{V}X$ for the set of values that a varible X can take on, and $\Delta\mathcal{V}X$ for the set of distributions over $\mathcal{V}X$. A conditional probability distribution (cpd) is a map $p(Y|X) : \mathcal{V}X \to \Delta\mathcal{V}Y$. A *directed hypergraph* $(N, \mathcal{A})$ is a set of nodes N and a set of arcs $\mathcal{A}$, each $a \in \mathcal{A}$ of which is associated with a set $S_a \subseteq N$ of source nodes, and $T_a \subseteq N$ target nodes. We also write $S \xrightarrow{a} T \in \mathcal{A}$ to specify an arc a together with its sources $S = S_a$ and targets $T = T_a$.

Geometry. We will need various parameter spaces Θ. To simplify the presentation, assume that each Θ is a convex subset of $\mathbb{R}^n$ (not necessarily of the same dimension). A *vector field* over Θ is a differentiable map X assigning to each $\theta \in \Theta$ a vector $X_\theta \in \mathbb{R}^n$. The *gradient* of a twice differentiable map $f : \Theta \to \mathbb{R}$, which we write $\nabla_\Theta f(\Theta)$, is a vector field. Given a vector field X and an initial point $\theta_0 \in \Theta$, there is a unique trajectory $y(t)$ that solves the ODE $\{\frac{\mathrm{d}}{\mathrm{d}t}y(t) = X_{y(t)}, y(0) = \theta_0\}$, and we adopt the notation $\exp_{\theta_0}(X) := y(1)$ for a compact description of it. At first glance, exp only gives us access to $y(1)$, but it is easily verified that $\exp_{\theta_0}(tX) = y(t)$. So altogether, the map

*Equal contribution [1]Department of Computer Science, Cornell University, Ithaca NY, USA. Correspondence to: Oliver Richardson <oli@cs.cornell.edu>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023.

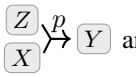
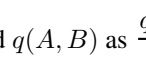
$t \mapsto \exp_\theta(t\nabla_\Theta f(\Theta))$ is the smooth path beginning at θ that follows the gradient of f. It is known as *gradient flow*.

Probabilistic Dependency Graphs. A PDG is a directed graph whose arcs carry probabilistic and causal information, weighted by confidence (Richardson & Halpern, 2021). We now introduce an equally expressive variant, whose explicit parametric nature will prove useful for our purposes.

Definition 2.1. A *Parametric Probabilistic Dependency Graph* (PPDG) $\mathcal{M}(\Theta) = (\mathcal{X}, \mathcal{A}, \Theta, \mathbb{P}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is a directed hypergraph $(\mathcal{X}, \mathcal{A})$ whose nodes correspond to variables, each arc $a \in \mathcal{A}$ of which is associated with:

- a parameter space $\Theta_a \subseteq \mathbb{R}^n$, with a default value θ_a^{init}.
- a map $\mathbb{P}_a : \Theta_a \times \mathcal{V}S_a \to \Delta\mathcal{V}T_a$ that gives a cpd $\mathbb{P}_a^\theta(T_a|S_a)$ over a's targets given its sources, for every $\theta \in \Theta_a$,
- confidences $\alpha_a \in \mathbb{R}$ in the functional dependence of T_a on S_a expressed by a, and $\beta_a \in [0, \infty]$ in the cpd $\mathbb{P}_a$.

A PDG is the object obtained by fixing the parameters; thus, a choice of $\theta \in \Theta := \prod_{a\in\mathcal{A}} \Theta_a$ yields a PDG $\mathcal{M} = \mathcal{M}(\theta)$.

Clearly, a PDG is the special case of a PPDG in which every $\Theta_a = \{\theta_a^{\text{init}}\}$ is a singleton. Conversely, a PPDG may be viewed as a PDG by adding each Θ_a as a variable, as illustrated in Figure 1. We often identify the label a with the cpd $\mathbb{P}_a$, and specify (P)PDGs in graphical notation, drawing a cpd $p(Y|X,Z)$ as  and $q(A,B)$ as  .

Unless otherwise specified, take $\beta, \alpha{=}1$ by default. We write $\mathcal{M}_1 + \mathcal{M}_2$ for the PDG that has the arcs of both $\mathcal{M}_1$ and $\mathcal{M}_2$, and represents their combined information.

PDG Semantics and Inconsistency. The power of PDGs comes from their semantics, which sew their (possibly inconsistent) cpds and confidences together into joint probabilistic information. A PDG contains two kinds of information: structural information about causal mechanisms, (the graph $\mathcal{A}$ and weights $\boldsymbol{\alpha}$), and observational data (the cpds $\mathbb{P}$ and confidences $\boldsymbol{\beta}$). With respect to a PDG $\mathcal{M}$, the *observational incompatibility* of a joint probability measure $\mu \in \Delta\mathcal{V}\mathcal{X}$ is given by a weighted sum of relative entropies

$$\mathit{OInc}_\mathcal{M}(\mu) := \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \boldsymbol{D}\Big(\mu(T,S) \,\Big\|\, \mathbb{P}_a(T|S)\mu(S)\Big), \quad (1)$$

and can be thought of as the excess cost of using codes optimized for each cpd, weighted by the confidence we have in them, if $\mathcal{X} \sim \mu$. If $\mathcal{M}$'s observational confidences are positive ($\boldsymbol{\beta} > \mathbf{0}$), then $\mathit{OInc}_\mathcal{M}(\mu) = 0$ if and only if μ has every conditional marginal described by $\mathbb{P}$.

We can also score μ by its incompatibility with the structural information $(\mathcal{A}, \boldsymbol{\alpha})$. This *structural deficiency* is given by:[1]

$$\mathit{SDef}_\mathcal{M}(\mu) := \mathop{\mathbb{E}}_{\mu}\left[\log \frac{\mu(\mathcal{X})}{\lambda(\mathcal{X})} \prod_{S \xrightarrow{a} T} \left(\frac{\lambda(T|S)}{\mu(T|S)}\right)^{\alpha_a}\right], \quad (2)$$

and, roughly, measures μ's failure to arise as a result of independent causal mechanisms along each edge. If $\mathcal{A}$ is a qualitative Bayesian Network, for instance, then $\mathit{SDef}_\mathcal{A}(\mu) \geq 0$ with equality iff μ has the independencies of $\mathcal{A}$. We encourage the reader to consult previous work for further details.

With confidence $\gamma \geq 0$ in the structural information overall, the γ-*inconsistency* of $\mathcal{M}$ is the smallest possible overall incompatibility of any distribution with $\mathcal{M}$, and denoted

$$\langle\!\langle \mathcal{M} \rangle\!\rangle_\gamma := \inf_\mu \Big(\mathit{OInc}_\mathcal{M}(\mu) + \gamma \mathit{SDef}_\mathcal{M}(\mu)\Big). \quad (3)$$

Richardson (2022) argues that this inconsistency measure (3) is a "universal" loss function, largely showing how it specializes to standard loss functions in a wide variety of situations. It follows that, at an abstract level, much of machine learning can be viewed as inconsistency resolution. We take this idea a few steps further, by operationalizing the resolution process, and allowing it to be done locally.

3. Local Inconsistency Resolution (LIR)

Attention and Control. There are two distinct senses in which inconsistency resolution can be *local*: we can restrict what we can see, or what we can do about it. Correspondingly, there are two "focus" knobs for our algorithm: one that restricts our attention to the inconsistency of a subset of arcs $A \subseteq \mathcal{A}$, and another that restricts our control to (only) the parameters of arcs $C \subseteq \mathcal{A}$ as we resolve that inconsistency. The former makes for an underestimate of the inconsistency that is easier to calculate, while the latter makes for an easier optimization problem. These restrictions are not just cheap approximations, though: they are also appropriate modeling assumptions for actors that cannot see and control everything at once.

Attention and control need not be black or white. A more general approach is to choose an *attention mask* $\varphi \in \mathbb{R}^\mathcal{A}$ and a *control mask* $\chi \in [0, \infty]^\mathcal{A}$. Large $\varphi(a)$ makes a salient, while $\varphi(a) = 0$ keeps it out of the picture. Similarly, large $\chi(a)$ gives significant freedom to change a's parameters, small $\chi(a)$ affords only minor adjustments, and $\chi(a) = 0$ prevents change altogether. Either mask can then be applied to a tensor that has an axis corresponding to $\mathcal{A}$, via pointwise multiplication ($\odot$).

The Algorithm. LIR modifies the parameters θ of a PPDG $\mathcal{M}(\Theta)$ so as to make it more consistent with its context.

[1] In (2), λ is base measure, a property of $\mathcal{X}$. The precise choice is not important, but think: uniform or an appropriate analogue.

It proceeds as follows. First, recieve context in the form of a PDG $\mathcal{Ctx}$, and initialize mutable memory $\mathcal{M}(\Theta)$. In each iteration, choose γ (which can be viewed as attention to structure), an attention mask φ over the arcs of $\mathcal{M}(\Theta)+\mathcal{Ctx}$, and a control mask χ over the arcs of $\mathcal{M}(\Theta)$. Calculate $\langle\!\langle \varphi \odot (\mathcal{M}(\theta)+\mathcal{Ctx})\rangle\!\rangle_\gamma$, the inconsistency of the combined context and memory, weighted by attention. (For discrete PDGs, this can be done with the methods of Richardson et al. (2023).) Then mitigate this local inconsistency by updating mutable memory θ via (an approximation to) gradient flow, changing a's parameters in proportion to control $\chi(a)$. The procedure is fully formalized in Algorithm 1.

Algorithm 1 Local Inconsistency Resolution (LIR)

Input: context $\mathcal{Ctx}$, mutable memory $\mathcal{M}(\Theta)$.
Initialze $\theta^{(0)} \leftarrow \theta^{\text{init}}$;
for $t = 0, 1, 2, \ldots$ **do**
  $\mathcal{Ctx} \leftarrow \textsc{Refresh}(\mathcal{Ctx})$; `//optional`
  $\varphi, \chi, \gamma \leftarrow \textsc{Refocus}()$;
  $\theta^{(t+1)} \leftarrow \exp_{\theta^{(t)}}\left\{-\chi \odot \nabla_\Theta \left\langle\!\!\left\langle \varphi \odot \left(\mathcal{Ctx} + \mathcal{M}(\Theta)\right)\right\rangle\!\!\right\rangle_\gamma\right\}$;

In order to execute this procedure, we must say something about how the choice of (φ, χ, γ) is made. Thus, we must supply an additional procedure REFOCUS to select attention and control masks. We focus mostly on the case where γ is fixed, and REFOCUS chooses non-deterministically from a fixed set of attention/control mask pairs $(\varphi, \chi) \in \mathbf{F}$, which we call *foci*. Algorithm 1 also allows us to a select a second procedure, REFRESH, which makes it easier to model recieving new information in online settings.

The ODE on the last line of Algorithm 1, which is an instance of gradient flow, may be approximated with an inner loop running an iterative gradient-based optimization algorithm. Alternatively, if REFOCUS produces small χ, then it is well-approximated by a single gradient descent step of size χ. At the other extreme: if χ is infinite in every component, then, so long as the parameterizations $\mathbb{P}$ are log-concave, the final line reduces to

$$\theta^{(t+1)} \leftarrow \arg\min_\theta \left\langle\!\!\left\langle \varphi \odot (\mathcal{Ctx} + \mathcal{M}(\theta))\right\rangle\!\!\right\rangle_\gamma, \quad \text{because of}$$

Theorem 3.1. *If $\mathbb{P}$ is log-concave, then for small enough γ, the map $\theta \mapsto \langle\!\langle \varphi \odot (\mathcal{Ctx} + \mathcal{M}(\theta))\rangle\!\rangle_\gamma$ is convex.*[2]

In the remaining sections, we give a sample of some historically important algorithms that are instances of LIR.

4. LIR in the Classification Setting

Consider a parametric classifier $p_\theta(Y|X)$, perhaps arising from a neural network whose final layer is a softmax. Suppose $\mathcal{V}Y$ is a finite set of classes. If $\mathcal{V}X$ is itself a manifold (such as the space of images), we can regard a value $x \in \mathcal{V}X$ as parameterizing a deterministic cpd, written $\overset{x}{\twoheadrightarrow} X$. Together with a labeled sample (x, y), we get a PPDG $\mathcal{M}(\theta) := \overset{x}{\twoheadrightarrow} X \xrightarrow{p_\theta} Y \xleftarrow{y}$ whose observational inconsistency is $\langle\!\langle \mathcal{M} \rangle\!\rangle_0 = -\log p_\theta(y|x)$, the standard training objective for such a classifier (Richardson, 2022). Each cpd plays major role in this inconsistency.

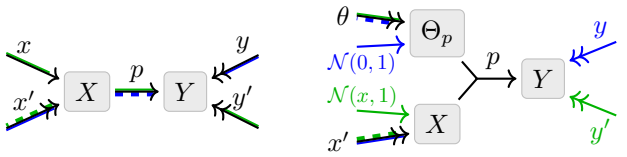


Figure 1. Two illustrations of adversarial training. Left: the PPDG obtained by including a perturbed input x' and target y' to the classification setting. Right: the PDG obtained by making the parameters for p explicit, together with a Gaussian prior $\Theta_p \sim \mathcal{N}(0, 1)$ over them. Both are colored with two foci: the blue focus trains the network, and the green one creates adversarial examples. Dashes indicate control.

What happens when we resolve this inconsistency by modifying the parameters associated to different arcs?

- Adjusting θ amounts to training the network in the standard way. In this case, the value χ of the control mask corresponds roughly to the product of the learning rate and the number of optimization iterations.
- Adjusting y is like a forward pass, in that it adjusts y to match distribution $p_\theta(Y|x)$.
- Adjusting x creates an adversarial example. That is, it makes incremental changes to the input x until the (fixed) network assigns it label y.

Stochastic Gradient Descent (SGD). Take the mutable state to be the classifier p as before. Define REFRESH so that it draws a batch of samples $\{(x_i, y_i)\}_{i=1}^m$, and returns a PDG with a single arc describing their emperical distribution $d(X, Y)$; let REFOCUS be such that $\varphi(d) = \infty$ (reflecting high confidence in the data). If $\eta := \chi(p)\varphi(p)$ is small, then LIR is SGD with batch size m and learning rate η.

Adversarial training. Suppose we want to slightly alter x to obtain x' that is classified as y' instead of y. By adding arcs corresponding to x' and y' to $\mathcal{M}$, and relaxing the cpd $\mathbb{P}_x$ associated with x to be a Gaussian centered x rather than a point mass, we get the PPDG on the left of Figure 1. An iteration of LIR whose focus is the edges marked in green (with control over the dashed green edge) is then an adversarial attack with Euclidean distance (Biggio et al., 2013). The blue focus, by contrast, "patches" the adversarial example by adjusting the model parameters to again classify it correctly. Thus, LIR that alternates between the two foci,

[2] All proofs can be found in the appendix.

in which REFRESH selects a fresh $(x, y, x' = x)$ from the dataset and target label y', is adversarial training, a standard defense to adversarial attacks (Goodfellow et al., 2014).

The ML community's focus on adversarial examples may appear to be a cultural phenomenon, but mathematically, it is no accident. At this level of abstraction, there is no difference between model parameters and inputs. Indeed, if we make the parameterization of p explicit and add L2 regularization (i.e., a Gaussian prior over Θ_p), the symmetry becomes striking (Figure 1, right). This may help explain why, even outside of adversarial contexts, it can be just as sensible to train an input, as a model (Kishore et al., 2021).

5. The EM Algorithm as LIR

Suppose we have a generative model $p(Z, X|\Theta)$ describing the probability over an observable variable X and a latent one Z. Given an observation $X{=}x$, the standard approach for trying to learn the parameters despite the missing data is called the EM algorithm. It iteratively computes

$$\theta_{\text{EM}}^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{z\sim p(Z|x,\theta_{\text{EM}}^{(t)})}[\log p(x, z|\theta)].$$

Proposition 5.1. *LIR* $\left(\overset{x}{\twoheadrightarrow} \boxed{X} \;,\; \begin{array}{c} p \\ \swarrow\;\searrow \\ \boxed{X} \qquad \boxed{Z} \end{array} \overset{q}{\underset{(\infty)}{\leftarrow}} \right)$ *in which* REFOCUS *fixes* $\varphi = \mathbf{1}$ *and alternates between full control of* p *and* q *implements EM, in that* $\theta_{EM}^{(t)} = \theta_{LIR}^{(2t)}$.

This result is closely related to one due to Neal & Hinton (1998), who view it as an intuitive explanation of why the EM algorithm works. Indeed, it is obvious in this form that every adjustement reduces the overall inconsistency. The result can also be readily adapted to an entire dataset by replacing x with a high confidence emperical distribution, or batched with the same technique in Section 4. It also captures fractional EM when $\chi < \infty$.

This form of the EM algorithm is closely related to variational inference. Indeed, analogous choices applied to the analysis of Richardson (2022) yields the usual training algorithm for variational autoencoders (VAEs).

6. Generative Adversarial Training as LIR

LIR also subsumes more complex training procedures such as the one used to train GANs (Goodfellow et al., 2020). The goal is to train a network G to generate images that cannot be distinguished from real ones. More precisely, define X to be either an image $X_{\text{fake}} \sim G$ or from a dataset $X_{\text{real}} \sim p_{\text{data}}$, based on a fair coin C. A discriminator D then predicts C from X. The generator also has a belief that, even given X, the coin is equally likely heads as tails (call this e). This state of affairs is summarized below.

$$\mathcal{m}(\Theta) := \quad \begin{array}{ccccc} & \overset{G}{\rightarrow} & \boxed{X_{\text{fake}}} & & \boxed{C} \overset{50/50!}{\leftarrow} \\ p_{\text{data}}! & \rightarrow & \boxed{X_{\text{real}}} & \twoheadrightarrow & \boxed{X} \quad e \circlearrowright D \end{array}$$

The GAN objective is typically written as a 2-player minimax game: $\min_G \min_D \mathcal{L}^{\text{GAN}}(G, D)$, where

$$\mathcal{L}^{\text{GAN}}(G, D) = \mathop{\mathbb{E}}_{x\sim p_{\text{data}}}[\log D(x)] + \mathop{\mathbb{E}}_{x'\sim G}[\log(1 - D(x'))].$$

The Discriminator's Focus. The discriminator has full control over D, and attends to everything but e. That inconsistency of this PDG is what might be called the discriminator's objective: the expected KL divergence from D to the optimal discriminator. If D also disbelieves that any image is equally likely to be fake as real (by chosing $\varphi(e) = -1$), then the inconsistency becomes $-\mathcal{L}^{\text{GAN}}$.

The Generator's Focus. The generator has control over G. If it ignores D attends only to e, the inconsistency is the Jenson-Shanon Divergence between G and p_{data}. If the generator also disbelieves the discriminator D (i.e., $\varphi(D) = -1$), then the inconsistency becomes $+\mathcal{L}^{\text{GAN}}$.

Standard practice is to use small $\chi(G)$ and large $\chi(D)$, so that the discriminator is well-adapted to the generator.

7. Message Passing Algorithms as LIR

Nearly every standard graphical model can be viewed as a factor graph, and correspondlingly admits an (approximate) inference procedure known variously as (loopy) belief propogation (Koller & Friedman, 2009), the generalized distributive law (**?**), and the sum-product algorithm (Kschischang et al., 2001). It also turns out to be the special case of LIR specialized to factor graphs.

A *factor graph* over a set of variables $\mathcal{X}$ is a set of factors $\Phi = \{\phi_a : \mathbf{X}_a \to \mathbb{R}_{\geq 0}\}_{a\in\mathcal{A}}$, where each $\mathbf{X}_a \subseteq \mathcal{X}$ is called the *scope* of a. Conversely, for $X \in \mathcal{X}$, let ∂X be the set of factors with X in scope. Φ specifies a distribution $\Pr_\Phi(\mathcal{X}) \propto \prod_a \phi_a(\mathbf{X}_a)$, and corresponds to a PDG

$$\mathcal{m}_\Phi = \left\{ \overset{\propto \phi_a}{\underset{(\alpha,\,\beta=1)}{\rightarrow}} \boxed{\mathbf{X}_a} \right\}_{a\in\mathcal{A}}$$

that specifies the same joint distirbution $\Pr_\Phi$, when observation and structure are weighted equally (i.e., $\gamma = 1$).

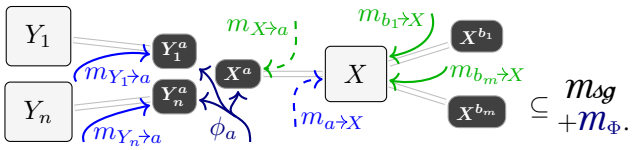
Sum-product belief propogation (Kschischang et al., 2001) aims to approximate marginals of $\Pr_\Theta$ with only local computations: messages sent between factors and the variables they have in scope. Its state consists of pairs of "messages" $\{m_{X\to a}, m_{a\to X}\}$, both (unnormalized) distributions over X, for each pair (a, X) with $a \in \partial X$, which together form a PDG $\mathcal{m}_{\mathcal{BP}}$ in the same way as the original factor graph. Af-

ter initialization, belief propogation repeatedly recomputes:

$$m_{X \to a}(x) :\propto \prod_{b \in \partial X \setminus a} m_{b \to X}(x) \tag{4}$$

$$m_{a \to X}(x) :\propto \sum_{\mathbf{y} \in \mathcal{V}(\mathbf{X}_a \setminus X)} \phi_a(x, \mathbf{y}) \prod_{Y \in \mathbf{X}_a \setminus X} m_{Y \to a}(Y(\mathbf{y})), \tag{5}$$

where $Y(\mathbf{y})$ is the value of Y in the joint setting $\mathbf{y}$. Finally, variable marginals $\{b_X\}_{X \in \mathcal{X}}$, which we regard as another PDG, $\mathcal{B}$, are computed from the messages according to $b_X(x) \propto \prod_{a \in \partial X} m_{a \to X}(x)$. Observe that every calculation is a (marginal of) a product of factors, and thus amounts to inference in some "local" factor graph. The traditional depiction of messages moving between variables and factors (see Appendix A) is not so different from the PDG



Indeed, it can be shown that (4,5) minimize inconsistency of the dotted components in their appropriate contexts (shown in green and blue above, and formalized in Appendix A).

Proposition 7.1. *If* REFOCUS *selects a focus non-deterministically from* $\{a \to X, X \to a, X\}_{X \in \mathcal{X}, a \in \partial X}$ *(details in Appendix A), then the possible runs of* LIR$(\mathcal{M}_\Phi, \mathcal{M}_{\mathcal{BG}} + \mathcal{B})$ *are precisely those of BP for different message schedules.*

There are many established variants of this algorithm. Some of them are generated different by clustering factors together—in the language of Koller & Friedman (2009), that is to say choosing something other than the Bethe cluster graph as the basis for message passing. Our analysis immediately applies to these other cluster graphs.

Minka (2005) offers a different perspective, in which a broader class of message passing algorithms can be viewed as iteratively adjusting some local context to minimize an α-divergence. We suspect that LIR generalizes thses procedure as well—not only because it is similar in spirit, but also because because these divergences can be viewed as the degree of inconsistency of a PDG containing two distributions (Richardson, 2022).

8. Discussion and Future Work

These examples are only the beginning. Our initial investigations suggest that opinion dynamics models, the training process for diffusion models, and much more, are all naturally captured by LIR. The surprising generality of LIR begs some theoretical questions. What assumptions are needed to prove that it reduces overall inconsistency, as is often the case? What are the simplest choices we could make to produce an efficient non-standard algorithm? How expressive is this mode of computation?

It also suggests a novel approach to structured generative modeling: haphazardly assemble a PDG with many variables, existing models, priors, constraints, and data of all shapes and sizes. Then, train new models to predict variables from one another, using LIR (with random refocusing, say). Is this effective? We are excited to find out!

References

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Š rndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Advanced Information Systems Engineering*, pp. 387–402. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-40994-3_25. URL https://doi.org/10.1007%2F978-3-642-40994-3_25.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2014.

Kishore, V., Chen, X., Wang, Y., Li, B., and Weinberger, K. Q. Fixed neural network steganography: Train the images, not the network. In *International Conference on Learning Representations*, 2021.

Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

Minka, T. Divergence measures and message passing. Technical Report MSR-TR-2005–173, Microsoft Research, Cambridge, U.K., 2005.

Neal, R. M. and Hinton, G. E. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pp. 355–368. Springer, 1998.

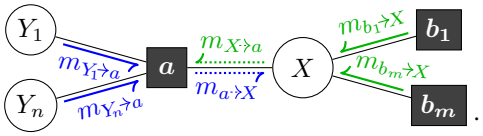
Richardson, O. E. Loss as the inconsistency of a probabilistic dependency graph: Choose your model, not your loss function. *AISTATS '22*, 151, 2022.

Richardson, O. E. and Halpern, J. Y. Probabilistic dependency graphs. *AAAI '21*, 2021.

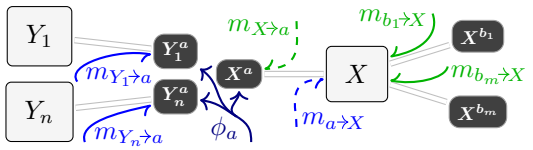
Richardson, O. E., Halpern, J. Y., and Sa, C. D. Inference in probabilistic dependency graphs. *UAI '23*, 2023.

A. Details on Belief Propogation

The usual schematic illustration of belief propogation (Kschischang et al., 2001) looks something like:



This is only a schematic, but the PDG $\mathcal{M}\mathcal{s}\mathcal{g}$ can be made to look similar to it. Adding a variable X^a for every pair (X, a) with $X \in \mathbf{X}_a$ along with edges asserting that $X^a = X$, we obtain the equivalent PDG in the main body of the paper:



We now define the views. Modulo a small subtlety, the following is essentially true: Equation (4) adjusts the parameters of $C_{X \to a} := \{m_{X \to a}\}$ so as to minimize 1-inconsistency in context $A_{X \to a} := \{m_{b \to X}\}_{b \in \partial X \setminus a} \cup \{m_{X \to a}\}$, while (5) adjusts $C_{a \to X} := \{m_{a \to X}\}$ so as to minimize the 1-inconsistency in context $A_{a \to X} := \{\phi_a, m_{a \to X}\} \cup \{m_{Y \to a}\}_{Y \in \mathbf{X}_a \setminus X}$.

The only wrinkle is that we do not want to attend to the structural aspect of a message e that we are updating—that is, we must select φ so as to ignore its causal weight α_e. Intuitively: when we are updating some message e, we are interested in summarizing information in the other messages (both observational and causal information), purely with an observation.

More precisely, the foci

$$\mathbf{F} := \left\{ (\varphi_j, \chi_j) \ : \ j \in \bigcup_{\substack{a \in A \\ X \in \mathbf{X}_a}} \left\{ a \to X,\ X \to a,\ X \right\}, \right\}$$

are indexed by messages and variables, and defined as follows. The attention mask φ_j is given by:

$$\varphi_j(a) := \begin{cases} \binom{1}{1} & \text{if } a \in A_j \setminus C_j \\ \binom{1}{0} & \text{if } a \in C_j \\ \binom{0}{0} & \text{otherwise} \end{cases},$$

where $\binom{\phi_1}{\phi_2}$ scales β_a by ϕ_1 and α_a by ϕ_2. Finally, full control over C_j means defining

$$\chi_j(a) := \begin{cases} \infty & \text{if } a \in C_j \\ 0 & \text{otherwise.} \end{cases}$$

With these definitions, Proposition 7.1 follows easily.

(Continue to Appendix B for proofs!)

B. Proofs

First, some extra details for Theorem 3.1. By parameteriations $\mathbb{P}$ log-concave, we mean that, for every $a \in \mathcal{A}$, and $(s,t) \in \mathcal{V}(S_a, T_a)$, the function

$$\theta \mapsto -\log \mathbb{P}_a^{\theta}(T_a = t \mid S_a = a) \quad : \Theta_a \to [0, \infty]$$

is convex. This is true for many families of distributions of interest. For example, if S_a, T_a is discrete, and the cpd is parameterized by stochastic matrices $\mathbf{P} = [p_{s,t}] \in [0,1]^{\mathcal{V}(S_a,T_a)}$, then

$$-\log \mathbb{P}_a^{\mathbf{P}}(T_a = t | S_a = s) = -\log(p_{s,t})$$

which is clearly convex in $\mathbf{P}$.

To take another example: if $\mathbb{P}_a$ is linear Gaussian, i.e., $\mathbb{P}_a(T|S) = \mathcal{N}(T|\mathbf{A}s + b, \sigma^2)$, parameterized by $(\mathbf{A}, b, 1/\sigma^2)$, then

$$-\log \mathbb{P}_a^{(\mathbf{A},b,\sigma^2)}(t|s) = -\frac{1}{2}\log\frac{2\pi}{\sigma^2} + \frac{1}{2}\left(\frac{t - \mathbf{A}s + b}{\sigma}\right)^2$$

which is convex in $(\mathbf{A}, b, \frac{1}{\sigma^2})$. Now, for the proof.

Theorem 3.1. *If $\mathbb{P}$ is log-concave, then the map $\theta \mapsto \langle\!\langle \varphi \odot (\mathcal{C}\boldsymbol{tx} + \mathcal{M}(\theta)) \rangle\!\rangle_\gamma$ is convex.*

Proof. By definition,

$$\langle\!\langle \varphi \odot (\mathcal{C}\boldsymbol{tx} + \mathcal{M}(\theta)) \rangle\!\rangle_\gamma = \inf_\mu \{ OInc_{\mathcal{C}\boldsymbol{tx}}(\mu) + OInc_{\mathcal{M}(\theta)} + SDef_{\mathcal{M}(\theta)}(\mu) + OInc_{\mathcal{M}(\theta)}(\mu) \}.$$

Only the final term actually depends on θ, though. Let $F(\mu)$ capture the first three terms. For all of our examples, and indeed, if γ is chosen small enough, it will be convex in μ (Richardson & Halpern, 2021). Then we have

$$\begin{aligned}
\langle\!\langle \varphi \odot (\mathcal{C}\boldsymbol{tx} + \mathcal{M}(\theta)) \rangle\!\rangle_\gamma &= \inf_\mu \left(F(\mu) + \mathop{\mathbb{E}}_\mu \left[\sum_{S \xrightarrow{a} T} \beta_a \log \frac{\mu(T|S)}{\mathbb{P}_a(T|S)} \right] \right) \\
&= \inf_\mu \left(F(\mu) + \mathop{\mathbb{E}}_\mu \left[\sum_{S \xrightarrow{a} T} \beta_a \log \frac{\mu(T|S)}{\lambda(T|S)} \right] + \mathop{\mathbb{E}}_\mu \left[\sum_{S \xrightarrow{a} T} \beta_a \log \frac{\lambda(T|S)}{\mathbb{P}_a(T|S)} \right] \right)
\end{aligned}$$

The second term is then entropy (relative to the base distribution), which is convex in μ. The first term, $F(\mu)$, is convex in μ as well, and neither depend on θ. The final term is linear in μ. Since $\mathbb{P}$ is log-convex in θ, the $\log \frac{\lambda(t|s)}{\mathbb{P}_a(t|s)}$ convex in θ, and so that third term is a conic combination of expectations that are all convex, and hence itself convex in θ. Thus, the sum of all three terms in the infemum is jointly convex in θ and in μ. Taking an infemum over μ pointwise, the result is still convex in θ. □

Proposition 7.1. *If* REFOCUS *selects a view non-deterministically from $\{a \to X, X \to a, X\}_{X \in \mathcal{X}, a \in \partial X}$ with φ, χ as above, and $\gamma = 1$, then the possible runs of* LIR$(\mathcal{M}_\Phi, \mathcal{M}\boldsymbol{g} + \mathcal{B})$ *are precisely those of BP for different message schedules.*

Proof. When $\gamma = 1$, and $\alpha, \beta = 1$ for all of the input factors, then the optimal distribution μ^* that realizes the infemum is just the product of factors. It follows that any distribution that has those marginals will minimize the observational inconsistency.

The different orders that the (4), and (5) can be ordered for different adjacent pairs (a, X) correspond to both the message passing schedules, and to the possible view selections of LIR. □