Journal of Machine Learning Research 11 (2010) 1-48 Submitted 10/09; Published ??/??

Cumulative distribution networks and the derivative-sum-product algorithm: Models and inference for cumulative distribution functions on graphs

Jim C. Huang JIMHUA@MICROSOFT.COM
Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA

Brendan J. Frey FREY@PSI.TORONTO.EDU
Probabilistic and Statistical Inference Group
University of Toronto
Toronto, ON M5S 3G4, Canada

Editor: Tommi Jaakkola

Abstract

We present a class of graphical models for directly representing the joint cumulative distribution function (CDF) of many random variables, called *cumulative distribution networks* (CDNs). Unlike graphs for probability density and mass functions, in a CDN, the marginal probabilities for any subset of variables are obtained by computing limits of functions in the model. We will show that the conditional independence properties in a CDN are distinct from the conditional independence properties of directed, undirected and factor graphs, but include the conditional independence properties of bidirected graphs. In order to perform inference in such models, we describe the 'derivative-sum-product' (DSP) message-passing algorithm in which messages correspond to derivatives of the joint CDF. We will then apply CDNs to the problem of learning to rank players in multiplayer team-based games and suggest several future directions for research.

Keywords: Graphical models, cumulative distribution function, message-passing algorithm, inference

1. Introduction

Probabilistic graphical models provide a pictorial means of specifying a joint probability density function (PDF) defined over many continuous random variables, the joint probability mass function (PMF) of many discrete random variables, or a joint probability distribution defined over a mixture of continuous and discrete variables. Each variable in the model corresponds to a node in a graph and edges between nodes in the graph convey statistical dependence relationships between the variables in the model. The graphical formalism allows one to obtain the independence relationships between random variables in a model by inspecting the corresponding graph, where the separation of nodes in the graph implies a particular conditional independence relationship between the corresponding variables.

A consequence of representing independence constraints between subsets of variables using a graph is that the joint probability often factors into a product of functions defined over subsets of neighboring nodes in the graph. Typically, this allows us to decompose a large

multivariate distribution into a product of simpler functions, so that the task of inference and estimation of such models can also be simplified and efficient algorithms for performing these tasks can be implemented. Often, a complex distribution over observed variables can be constructed using a graphical model with latent variables introduced, where the joint probability over the observed variables is obtained by marginalization over the latent variables. The model with additional latent variables has the advantage of having a more compact factorized form as compared to that for the joint probability over the observed variables. However, this often comes at the cost of a significantly higher computational cost for estimation and inference, as additional latent variables often require one to either approximate intractable marginalization operations (Minka, 2001) or to sample from the model using Markov Chain Monte Carlo (MCMC) methods (Neal, 1993). Furthermore, there is also the problem that there are possibly an infinite number of latent variable models associated with any given model defined over observable variables, so that adding latent variables for any given application can often present difficulties in terms of model identifiability, which is desirable in the case in which one wishes to interpret the parameters in a graphical model. These issues may hamper the applicability of graphical models for many real-world problems in the presence of latent variables.

Another possible limitation of many graphical models is that the joint PDF/PMF itself might not be appropriate as a probability model for certain applications. For example, in learning to rank, the *cumulative distribution function* (CDF) is a probabilistic representation that arises naturally as a probability measure over inequality events of the type $\{X \leq x\}$. The joint CDF lends itself to such problems that are easily described in terms of inequality events in which statistical dependence relationships also exist among events. An example of this type of problem is that of predicting multiplayer game outcomes with a team structure (Herbrich, Minka and Graepel, 2007). In contrast to the canonical problems of classification or regression, in learning to rank we are required to learn some mapping from inputs to inter-dependent output variables so that we may wish to model both stochastic orderings between variable states and statistical dependence relationships between variables.

Given the above, here we present a class of graphical models called *cumulative distribution networks* (CDN) in which we represent the joint CDF of a set of observed variables. As we will show, CDNs can be viewed as graphical models for a subset of joint probability distributions that could be obtained by exact marginalization of latent variables in a directed graphical model. Thus, CDNs can be viewed as providing a means to construct complex distributions over observed variables without the need to explicitly introduce latent variables and then marginalize. The resulting model consists of a factorized form for the joint CDF, where the principal operations required for answering probabilistic queries and for marginalization consist of differentiation and computing limits respectively, in contrast to summation/integration in graphical models for PDFs with latent variables. Furthermore, the parameterization of the model as a joint CDF has the advantage that the global normalization constraint can be enforced locally for each function in the CDN, unlike the case of undirected graphs. We will present the basic properties of CDNs and show that the rules for ascertaining conditional independence relationships among variables in a CDN are distinct from the rules in directed, undirected and factor graphs (Pearl, 1988; Lauritzen, 1996; Kschischang, Frey and Loeliger, 2001). We will show that the conditional independence properties in a CDN include, but are not limited to, the conditional independence prop-

erties for bidirected graphs (Drton and Richardson, 2008; Richardson and Spirtes, 2002; Richardson, 2003).

We will then discuss the problem of performing inference under CDNs in which the principal challenge is to compute the derivatives of the joint CDF. To this end we will describe a message-passing algorithm for inference in CDNs called the *derivative-sum-product algorithm* based on previous work (Huang and Frey, 2008; Huang, 2009). To demonstrate the applicability of CDNs, we will use the message-passing algorithm for inference in order to apply CDNs to the problem of learning to rank, where we will show that CDFs arise naturally as a probability models in which it is easy to specify stochastic ordering constraints amongst variables in the model.

1.1 Notation

Before we proceed, we will establish some notation to be used throughout the paper. We will denote bipartite graphs as $\mathcal{G} = (V, S, E)$ where V, S are two disjoint sets of nodes and $E \subseteq \{V \times S, S \times V\}$ is a set of edges that correspond to ordered pairs (α, s) or (s, α) for $\alpha \in V$ and $s \in S$. We will denote neighboring sets $\mathcal{N}(\alpha)$ and $\mathcal{N}(s)$ as

$$\mathcal{N}(\alpha) = \{s \in S : (\alpha, s) \in E\}$$
$$\mathcal{N}(s) = \{\alpha \in V : (\alpha, s) \in E\}.$$

Furthermore, let $\mathcal{N}(A) = \cup_{\alpha \in A} \mathcal{N}(\alpha)$.

Throughout the paper we will use boldface notation to denote vectors and/or matrices. Scalar and vector random variables will be denoted as X_α and $\mathbf{X}_A$ respectively where α is a node in a graph $\mathcal{G}$ and A denotes a set of nodes in $\mathcal{G}$. The notation $|A|, |\mathbf{x}|, |\mathbf{X}|$ will denote the cardinality, or number of elements, in set A and vectors $\mathbf{x}, \mathbf{X}$ respectively. We will also denote the mixed partial derivative/finite difference as $\partial_{\mathbf{x}_A}\Big[\cdot\Big]$, where the mixed derivative here is taken with respect to arguments $x_\alpha \; \forall \; \alpha \in A$.

1.2 Cumulative distribution functions

Here we provide a brief definition for the joint CDF $F(\mathbf{x})$ defined over random variables $\mathbf{X}$, denoted individually as X_α. The joint *cumulative distribution function* $F(\mathbf{x})$ is then defined as the function $F : \mathbb{R}^{|\mathbf{X}|} \mapsto [0, 1]$ such that

$$F(\mathbf{x}) = \mathbb{P}\left[\bigcap_{X_\alpha \in \mathbf{X}} \big\{X_\alpha \leq x_\alpha\big\}\right] \equiv \mathbb{P}\big[\mathbf{X} \leq \mathbf{x}\big].$$

Thus the CDF is a probability defined over events $\{X_\alpha \leq x_\alpha\}$. Alternately, the CDF can be defined in terms of the joint probability density function (PDF) or probability mass function (PMF) $P(\mathbf{x})$ via

$$F(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} P(\mathbf{u}) \; d\mathbf{u},$$

where $P(\mathbf{x})$, if it exists, satisfies $P(\mathbf{x}) \geq 0$, $\int_{\mathbf{x}} P(\mathbf{x}) \; d\mathbf{x} = 1$ and $P(\mathbf{x}) = \partial_{\mathbf{x}}\Big[F(\mathbf{x})\Big]$ where $\partial_{\mathbf{x}}\Big[\cdot\Big]$ denotes the higher-order mixed derivative operator $\partial_{x_1,\cdots,x_K}\Big[\cdot\Big] \equiv \dfrac{\partial^K}{\partial x_1 \cdots \partial x_K}$ for $\mathbf{x} = [x_1 \; \cdots \; x_K] \in \mathbb{R}^K$.

A function F is a CDF for some probability $\mathbb{P}$ if and only if F satisfies the following conditions:

1. The CDF $F(\mathbf{x})$ converges to unity as all of its arguments tend to ∞, or
$$F(\infty) \equiv \lim_{\mathbf{x}\to\infty} F(\mathbf{x}) = 1.$$
2. The CDF $F(\mathbf{x})$ converges to 0 as any of its arguments tends to $-\infty$, or
$$F(-\infty, \mathbf{x} \setminus x_\alpha) \equiv \lim_{x_\alpha \to -\infty} F(x_\alpha, \mathbf{x} \setminus x_\alpha) = 0 \quad \forall\, X_\alpha \in \mathbf{X}.$$
3. The CDF $F(\mathbf{x})$ is monotonically non-decreasing, so that
$$F(\mathbf{x}) \leq F(\mathbf{y}) \;\forall\, \mathbf{x} \leq \mathbf{y},\; \mathbf{x}, \mathbf{y} \in \mathbb{R}^{|\mathbf{X}|}.$$
where $\mathbf{x} \leq \mathbf{y}$ denotes elementwise inequality of all the elements in vectors $\mathbf{x}, \mathbf{y}$.
4. The CDF $F(\mathbf{x})$ is right-continuous, so that
$$\lim_{\boldsymbol{\epsilon}\to\mathbf{0}^+} F(\mathbf{x} + \boldsymbol{\epsilon}) \equiv F(\mathbf{x}) \;\forall\, \mathbf{x} \in \mathbb{R}^{|\mathbf{X}|}.$$

A proof of forward implication in the above can be found in (Wasserman, 2004; Papoulis and Pillai, 2001).

Proposition 1.1. Let $F(\mathbf{x}_A, \mathbf{x}_B)$ be the joint CDF for variables $\mathbf{X}$ where $\mathbf{X}_A, \mathbf{X}_B$ for a partition of the set of variables $\mathbf{X}$. The joint probability of the event $\{\mathbf{X}_A \leq \mathbf{x}_A\}$ is then given in terms of $F(\mathbf{x}_A, \mathbf{x}_B)$ as

$$F(\mathbf{x}_A) \equiv \mathbb{P}\Big[\mathbf{X}_A \leq \mathbf{x}_A\Big] = \lim_{\mathbf{x}_B\to\infty} F(\mathbf{x}_A, \mathbf{x}_B). \tag{1}$$

□

The above proposition follows directly from the definition of a CDF in which

$$\lim_{\mathbf{x}_B\to\infty} \bigcap_{\alpha\in A\cup B} \{X_\alpha \leq x_\alpha\} = \bigcap_{\alpha\in A} \{X_\alpha \leq x_\alpha\}. \tag{2}$$

Thus, marginal CDFs of the form $F(\mathbf{x}_A)$ can be computed from the joint CDF by computing limits.

1.3 Conditional cumulative distribution functions

In the sequel we will be making use of the concept of a conditional CDF for some subset of variables $\mathbf{X}_A$ conditioned on event M. We formally define the conditional CDF below.

Definition 1.1. Let M be an event with $\mathbb{P}[M] > 0$. The conditional CDF $F(\mathbf{x}_A \mid M)$ conditioned on event M is defined as

$$F(\mathbf{x}_A \mid M) \equiv \mathbb{P}\Big[\mathbf{X}_A \leq \mathbf{x}_A \mid M\Big] = \frac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \cap M\Big]}{\mathbb{P}\big[M\big]}. \tag{3}$$

□

We will now find the above conditional CDF for different types of events M. .

Lemma 1.2. Let $F(\mathbf{x}_C)$ be a marginal CDF obtained from the joint CDF $F(\mathbf{x})$ as given by Proposition 1.1 for some $\mathbf{X}_C \subseteq \mathbf{X}$. Consider some variable set $\mathbf{X}_A \subseteq \mathbf{X}$. Let $M = \omega(\mathbf{x}_C) \equiv \{\mathbf{X}_C \leq \mathbf{x}_C\}$ for $\mathbf{X}_C \subset \mathbf{X}$. If $F(\mathbf{x}_C) > 0$, then $F(\mathbf{x}_A|\omega(\mathbf{x}_C)) \equiv F(\mathbf{x}_A|\mathbf{X}_C \leq \mathbf{x}_C) = \dfrac{F(\mathbf{x}_A, \mathbf{x}_C)}{F(\mathbf{x}_C)}$. □

Thus a conditional CDF of the form $F(\mathbf{x}_A|\omega(\mathbf{x}_C))$ can be obtained by taking ratios of joint CDFs, which consists of computing limits to obtain the required marginal CDFs. It follows from Lemma 1.2 that marginalization over variables $\mathbf{X}_C$ can be viewed as a special case of conditioning on $\mathbf{X}_C < \infty$.

To compute conditional CDFs of the form $F(\mathbf{x}_A|x_\beta)$ where we instead condition on observing x_β, we need to *differentiate* the joint CDF, as we now show.

Lemma 1.3. Consider some variable set $\mathbf{X}_A \subseteq \mathbf{X}$. Let $M = \{x_\beta < X_\beta \leq x_\beta + \epsilon\}$ with $\epsilon > 0$ for some scalar random variable $X_\beta \notin \mathbf{X}_A$. If $F(x_\beta)$ and $F(\mathbf{x}_A, x_\beta)$ are differentiable with respect to x_β so that $\partial_{x_\beta}\Big[F(x_\beta)\Big]$ and $\partial_{x_\beta}\Big[F(\mathbf{x}_A, x_\beta)\Big]$ exist with $\partial_{x_\beta}\Big[F(x_\beta)\Big] > 0$, then the conditional CDF $F(\mathbf{x}_A|x_\beta) \equiv \lim_{\epsilon\to 0^+} F(\mathbf{x}_A|x_\beta < X_\beta < x_\beta + \epsilon) = \lim_{\epsilon\to 0^+} \dfrac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \cap \{x_\beta < X_\beta \leq x_\beta + \epsilon\}\Big]}{\mathbb{P}\Big[x_\beta < X_\beta \leq x_\beta + \epsilon\Big]}$ is given by

$$F(\mathbf{x}_A|x_\beta) = \frac{\partial_{x_\beta}\Big[F(\mathbf{x}_A, x_\beta)\Big]}{\partial_{x_\beta}\Big[F(x_\beta)\Big]} \propto \partial_{x_\beta}\Big[F(\mathbf{x}_A, x_\beta)\Big]. \tag{4}$$

Proof. We can write

$$\begin{aligned} F(\mathbf{x}_A|x_\beta < X_\beta \leq x_\beta + \epsilon) &= \frac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \cap \{x_\beta < X_\beta \leq x_\beta + \epsilon\}\Big]}{\mathbb{P}\Big[x_\beta < X_\beta \leq x_\beta + \epsilon\Big]} \\ &= \frac{\frac{1}{\epsilon}\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \cap \{x_\beta < X_\beta \leq x_\beta + \epsilon\}\Big]}{\frac{1}{\epsilon}\mathbb{P}\Big[x_\beta < X_\beta \leq x_\beta + \epsilon\Big]} = \frac{\frac{F(\mathbf{x}_A, x_\beta+\epsilon) - F(\mathbf{x}_A, x_\beta)}{\epsilon}}{\frac{F(x_\beta+\epsilon) - F(x_\beta)}{\epsilon}}. \end{aligned} \tag{5}$$

Taking limits, and given differentiability of both $F(x_\beta)$ and $F(\mathbf{x}_A, x_\beta)$ with respect to x_β, the conditional CDF $F(\mathbf{x}_A|x_\beta)$ is given by

$$F(\mathbf{x}_A|x_\beta) \equiv \frac{\lim_{\epsilon\to 0^+} \dfrac{F(\mathbf{x}_A, x_\beta+\epsilon) - F(\mathbf{x}_A, x_\beta)}{\epsilon}}{\lim_{\epsilon\to 0^+} \dfrac{F(x_\beta+\epsilon) - F(x_\beta)}{\epsilon}} = \frac{\partial_{x_\beta}\Big[F(\mathbf{x}_A, x_\beta)\Big]}{\partial_{x_\beta}\Big[F(x_\beta)\Big]} \propto \partial_{x_\beta}\Big[F(\mathbf{x}_A, x_\beta)\Big], \tag{6}$$

where the proportionality constant does not depend on $\mathbf{x}_A$. □

Lemma 1.4. Let $M = \{\mathbf{x}_C < \mathbf{X}_C \leq \mathbf{x}_C + \boldsymbol{\epsilon}\} \equiv \cap_{\gamma \in C}\{x_\gamma < X_\gamma \leq x_\gamma + \epsilon\}$ with $\epsilon > 0$ for $\mathbf{X}_C \subset \mathbf{X}$ and $\boldsymbol{\epsilon} = [\epsilon \ \cdots \ \epsilon]^T \in \mathbb{R}^{|\mathbf{X}_C|}$. Consider the set of random variables $\mathbf{X}_A \subset \mathbf{X}$ with $\mathbf{X}_C \cap \mathbf{X}_A = \emptyset$. If both $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_C)\Big]$ and $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_A, \mathbf{x}_C)\Big]$ exist for all $\mathbf{x}_C$ with $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_C)\Big] > 0$, then the conditional CDF $F(\mathbf{x}_A|\mathbf{x}_C) \equiv \lim_{\boldsymbol{\epsilon} \to \mathbf{0}^+} F(\mathbf{x}_A|\mathbf{x}_C < \mathbf{X}_C \leq \mathbf{x}_C + \boldsymbol{\epsilon}) = \lim_{\boldsymbol{\epsilon} \to \mathbf{0}^+} \frac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \cap \{\mathbf{x}_C < \mathbf{X}_C \leq \mathbf{x}_C + \boldsymbol{\epsilon}\}\Big]}{\mathbb{P}\Big[\mathbf{x}_C < \mathbf{X}_C \leq \mathbf{x}_C + \boldsymbol{\epsilon}\Big]}$ is given by

$$F(\mathbf{x}_A|\mathbf{x}_C) = \frac{\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_A, \mathbf{x}_C)\Big]}{\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_C)\Big]} \propto \partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_A, \mathbf{x}_C)\Big], \tag{7}$$

where $\partial_{\mathbf{x}_C}\Big[\cdot\Big]$ is a mixed derivative operator with respect to $\{x_\gamma, \gamma \in C\}$.

Proof. We can proceed by induction on variable set $\mathbf{X}_C$ with the base case given by Lemma 1.2. Let $\mathbf{X}_C = \mathbf{X}_{C'} \cup X_\beta$ with $X_\beta \notin \mathbf{X}_{C'} \cup \mathbf{X}_A$. Let $M' \equiv M'(\boldsymbol{\xi}) = \{\mathbf{x}_{C'} \leq \mathbf{X}_{C'} \leq \mathbf{x}_{C'} + \boldsymbol{\xi}\} = \cap_{\gamma \in C'}\{x_\gamma < X_\gamma \leq x_\gamma + \xi\}$ and $M \equiv M(\boldsymbol{\xi}, \epsilon) = M' \cap \{x_\beta < X_\beta \leq x_\beta + \epsilon\}$ with $\boldsymbol{\epsilon} = [\boldsymbol{\xi}^T \ \epsilon]^{\mathrm{T}}$. Suppose that $\partial_{\mathbf{x}_{C'}}\Big[F(\mathbf{x}_{C'})\Big] > 0$ and we have computed

$$F(\mathbf{x}_A, x_\beta|\mathbf{x}_{C'}) \equiv \lim_{\boldsymbol{\xi} \to 0^+} F\big(\mathbf{x}_A, x_\beta|M'(\boldsymbol{\xi})\big) = \frac{\partial_{\mathbf{x}_{C'}}\Big[F(\mathbf{x}_A, x_\beta, \mathbf{x}_{C'})\Big]}{\partial_{\mathbf{x}_{C'}}\Big[F(\mathbf{x}_{C'})\Big]} \tag{8}$$

and

$$F(x_\beta|\mathbf{x}_{C'}) \equiv \lim_{\boldsymbol{\xi} \to 0^+} F\big(x_\beta|M'(\boldsymbol{\xi})\big) = \frac{\partial_{\mathbf{x}_{C'}}\Big[F(x_\beta, \mathbf{x}_{C'})\Big]}{\partial_{\mathbf{x}_{C'}}\Big[F(\mathbf{x}_{C'})\Big]}. \tag{9}$$

Then we can write

$$F(\mathbf{x}_A \mid M) = \frac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \cap \{x_\beta < X_\beta \leq x_\beta + \epsilon\} \mid M'\Big]}{\mathbb{P}\Big[x_\beta < X_\beta \leq x_\beta + \epsilon \mid M'\Big]} = \frac{\frac{F(\mathbf{x}_A, x_\beta + \epsilon|M') - F(\mathbf{x}_A, x_\beta|M')}{\epsilon}}{\frac{F(x_\beta + \epsilon|M') - F(x_\beta|M')}{\epsilon}}. \tag{10}$$

Thus, since $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_C)\Big] > 0$ by hypothesis, we obtain

$$\begin{aligned} F(\mathbf{x}_A|\mathbf{x}_C) &= \lim_{\epsilon \to 0^+, \boldsymbol{\xi} \to 0^+} \frac{\frac{F(\mathbf{x}_A, x_\beta + \epsilon|M') - F(\mathbf{x}_A, x_\beta|M')}{\epsilon}}{\frac{F(x_\beta + \epsilon|M') - F(x_\beta|M')}{\epsilon}} = \frac{\lim_{\epsilon \to 0^+} \frac{F(\mathbf{x}_A, x_\beta + \epsilon|\mathbf{x}_{C'}) - F(\mathbf{x}_A, x_\beta|\mathbf{x}_{C'})}{\epsilon}}{\lim_{\epsilon \to 0^+} \frac{F(x_\beta + \epsilon|\mathbf{x}_{C'}) - F(x_\beta|\mathbf{x}_{C'})}{\epsilon}} \\ &= \frac{\partial_{x_\beta, \mathbf{x}_{C'}}\Big[F(\mathbf{x}_A, x_\beta, \mathbf{x}_{C'})\Big]}{\partial_{x_\beta, \mathbf{x}_{C'}}\Big[F(x_\beta, \mathbf{x}_{C'})\Big]} = \frac{\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_A, \mathbf{x}_C)\Big]}{\partial_{\mathbf{x}_C}\Big[F(\mathbf{x}_C)\Big]}. \end{aligned} \tag{11}$$

□

Thus a conditional CDF of the form $F(\mathbf{x}_A|\mathbf{x}_C)$ can be obtained by differentiation of the joint CDF. By Schwarz's Theorem this differentiation is invariant to the order in which variables are processed provided that the derivatives required to compute $F(\mathbf{x}_A|\mathbf{x}_C)$ exist and are continuous.

2. Cumulative distribution networks

Graphical models allow us to simplify the computations required for obtaining conditional probabilities of the form $P(\mathbf{x}_A|\mathbf{x}_B)$ or $P(\mathbf{x}_A)$ by allowing us to model conditional independence constraints in terms of graph separation constraints. However, for many applications it may be desirable to compute other conditional and marginal probabilities such as probabilities of events of the type $\{\mathbf{X} \leq \mathbf{x}\}$. Here we will present the cumulative distribution network (CDN), which is a graphical framework for directly modelling the joint cumulative distribution function, or CDF. With the CDN, we can thus expand the set of possible probability queries so that in addition to formulating queries as conditional/marginal probabilities of the form $P(\mathbf{x}_A)$ and $P(\mathbf{x}_A|\mathbf{x}_B)$, we can also compute probabilities of the form $F(\mathbf{x}_A|\omega(\mathbf{x}_B)), F(\mathbf{x}_A|\mathbf{x}_B), P(\mathbf{x}_A|\omega(\mathbf{x}_B))$ and $F(\mathbf{x}_A)$, where $F(\mathbf{u}) \equiv \mathbb{P}\Big[\mathbf{U} \leq \mathbf{u}\Big]$ is a CDF and we denote the inequality event $\{\mathbf{U} \leq \mathbf{u}\}$ using $\omega(\mathbf{x}_U)$. Examples of this new type of query could be "Given that the drug dose was less than 1 mg, what is the probability of the patient living at least another year?", or "Given that a person prefers one brand of soda over another, what is the probability of that person preferring one type of chocolate over another?". A significant advantage with CDNs is that the graphical representation of the joint CDF may naturally allow for queries which would otherwise be difficult, if not intractable, to compute under directed, undirected and factor graphical models for PDFs/PMFs.

Here we will define the CDN and we will show that the conditional independence property in such graphical models are distinct from the properties for directed, undirected and factor graphs. We will then show that the conditional independence properties in CDNs include the properties of bidirected graphs (Drton and Richardson, 2008; Richardson, 2003). Finally, we will show that CDNs provide a tractable means of parameterizing models for learning to rank in which we can construct multivariate CDFs from a product of CDFs defined over subsets of variables.

Definition 2.1. *The cumulative distribution network (CDN) is an undirected bipartite graphical model consisting of a bipartite graph $\mathcal{G} = (V, S, E)$, where V denotes variable nodes and S denotes factor nodes, with edges in E connecting factor nodes to variable nodes. The CDN also includes a specification of functions $\phi_s(\mathbf{x}_s)$ for each function node $s \in S$, where $\mathbf{x}_s \equiv \mathbf{x}_{\mathcal{N}(s)}$, $\cup_{s \in S} \mathcal{N}(s) = V$ and each function $\phi_s : \mathbb{R}^{|\mathcal{N}(s)|} \mapsto [0,\ 1]$ satisfies the properties of a CDF. The joint CDF over the variables in the CDN is then given by the product over CDFs $\phi_s : \mathbb{R}^{|\mathcal{N}(s)|} \mapsto [0,\ 1]$, or*

$$F(\mathbf{x}) = \prod_{s \in S} \phi_s(\mathbf{x}_s),$$

where each CDF ϕ_s is defined over neighboring variable nodes $\mathcal{N}(s)$. □

An example of a CDN defined over three variable nodes with four CDN function nodes is shown in Figure 1, where the joint CDF over three variables X, Y, Z is given by

$$F(x, y, z) = \phi_a(x, y)\phi_b(x, y, z)\phi_c(y, z)\phi_d(z).$$

In the CDN, each function node (depicted as a diamond) corresponds to one of the functions $\phi_s(\mathbf{x}_s)$ in the model for the joint CDF $F(\mathbf{x})$. Thus, one can think of the CDN as a factor graph for modelling the joint CDF instead of the joint PDF. However, as we will see shortly, this leads to a different set of conditional independence properties as compared to the conditional independence properties of directed, undirected and factor graphs.

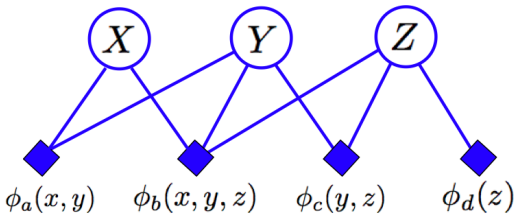


Figure 1: A cumulative distribution network (CDN) defined over three variables and four functions.

Since the CDN is a graphical model for the joint CDF, the functions in the CDN must be such that $F(\mathbf{x})$ is a CDF for some probability $\mathbb{P}$. The following lemma establishes a sufficient condition that the CDN functions ϕ_s be themselves CDFs in order for F to be a CDF.

Lemma 2.1. *If all functions $\phi_s(\mathbf{x}_s)$ satisfy the properties of a CDF, then the product $\prod_{s \in S} \phi_s(\mathbf{x}_s)$ also satisfies the properties of a CDF.* □

Proof. If for all $s \in S$, we have $\lim_{\mathbf{x}_s \to \infty} \phi_s(\mathbf{x}_s) = 1$, then $\lim_{\mathbf{x} \to \infty} \prod_{s \in S} \phi_s(\mathbf{x}_s) = 1$. Furthermore, if for any given $\alpha \in V$ and for $s \in \mathcal{N}(\alpha)$, we have $\lim_{x_\alpha \to -\infty} \phi_s(\mathbf{x}_s) = 0$, then $\lim_{x_\alpha \to -\infty} \prod_{s \in S} \phi_s(\mathbf{x}_s) = 0$.

To show that the product of monotonically non-decreasing functions is monotonically non-decreasing, we note that $\mathbf{x}_s < \mathbf{y}_s$ for all $s \in S$ iff $\mathbf{x} < \mathbf{y}$, since $\cup_{s \in S} \mathcal{N}(s) = V$. Thus if we have $\phi_s(\mathbf{x}_s) \leq \phi_s(\mathbf{y}_s)\ \forall\ \mathbf{x}_s < \mathbf{y}_s$ for all $s \in S$, we can then write

$$F(\mathbf{x}) = \prod_{s \in S} \phi_s(\mathbf{x}_s) \leq \prod_{s \in S} \phi_s(\mathbf{y}_s) = F(\mathbf{y}).$$

Finally, a product of right-continuous functions is also right-continuous. Thus if all of the functions $\phi_s(\mathbf{x}_s)$ satisfy the properties of a CDF, then the product of such functions also satisfies the properties of a CDF. □

Although the condition that each of the ϕ_s functions be a CDF is sufficient for the overall product to satisfy the properties of a CDF, we emphasize that it is not a necessary condition, as one could construct a function that satisfies the properties of a CDF from a product of functions that are not CDFs. The sufficient condition above ensures, however, that we can

construct CDNs by multiplying together CDFs to obtain another CDF. Furthermore, the above definition and theorem do not assume differentiability of the CDF or of the CDN functions: the following proposition shows that differentiability and non-negativity of the derivatives of functions ϕ_s with respect to all neighboring variables in $\mathcal{N}(s)$ imply both differentiability and monotonicity of the joint CDF $F(\mathbf{x})$. In the sequel we will assume that whenever CDN functions are differentiable, the order in which their derivatives are computed does not matter (Schwarz' Theorem).

Proposition 2.2. *If the mixed derivatives* $\partial_{\mathbf{x}_A}\Big[\phi_s(\mathbf{x}_s)\Big]$ *satisfy* $\partial_{\mathbf{x}_A}\Big[\phi_s(\mathbf{x}_s)\Big] \geq 0$ *for all* $s \in S$ *and* $A \subseteq \mathcal{N}(s)$*, then*

- $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x})\Big] \geq 0$ *for all* $C \subseteq V$
- $F(\mathbf{x}) \leq F(\mathbf{y})$ *for all* $\mathbf{x} < \mathbf{y}$.
- $F(\mathbf{x})$ *is differentiable.*

□

Proof. A product of differentiable functions is differentiable and so $F(\mathbf{x})$ is differentiable. To show that $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x})\Big] \geq 0 \;\forall\; C \subseteq V$, we can group the functions $\phi_s(\mathbf{x}_s)$ arbitrarily into two functions $g(\mathbf{x})$ and $h(\mathbf{x})$ so that $F(\mathbf{x}) = g(\mathbf{x})h(\mathbf{x})$. The goal here will be to show that if all derivatives $\partial_{\mathbf{x}_A}\Big[g(\mathbf{x})\Big]$ and $\partial_{\mathbf{x}_A}\Big[h(\mathbf{x})\Big]$ are non-negative, then $\partial_{\mathbf{x}_A}\Big[F(\mathbf{x})\Big]$ must also be non-negative. For all $C \subseteq V$, applying the product rule to $F(\mathbf{x}) = g(\mathbf{x})h(\mathbf{x})$ yields

$$\partial_{\mathbf{x}_C}\Big[F(\mathbf{x})\Big] = \sum_{A \subseteq C} \partial_{\mathbf{x}_A}\Big[g(\mathbf{x})\Big]\partial_{\mathbf{x}_{C\setminus A}}\Big[h(\mathbf{x})\Big],$$

so if $\partial_{\mathbf{x}_A}\Big[g(\mathbf{x})\Big], \partial_{\mathbf{x}_{C\setminus A}}\Big[h(\mathbf{x})\Big] \geq 0$ for all $A \subseteq C$ then $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x})\Big] \geq 0$. By recursively applying this rule to each of the functions $g(\mathbf{x}), h(\mathbf{x})$ until we obtain sums over terms involving $\partial_{\mathbf{x}_A}\Big[\phi_s(\mathbf{x}_s)\Big] \;\forall\; A \subseteq \mathcal{N}(s)$, we see that if $\partial_{\mathbf{x}_A}\Big[\phi_s(\mathbf{x}_s)\Big] \geq 0$, then $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x})\Big] \geq 0 \;\forall\; C \subseteq V$.

Now, $\partial_{\mathbf{x}_C}\Big[F(\mathbf{x})\Big] \geq 0$ for all $C \subseteq V$ implies that $\partial_{x_\alpha}\Big[F(\mathbf{x})\Big] \geq 0$ for all $\alpha \in V$. By the Mean Value Theorem for functions of several variables, it then follows that if $\mathbf{x} < \mathbf{y}$, then

$$F(\mathbf{y}) - F(\mathbf{x}) = \sum_{\alpha \in V} \partial_{z_\alpha}\Big[F(\mathbf{z})\Big](y_\alpha - x_\alpha) \geq 0,$$

and so $F(\mathbf{x})$ is monotonic. □

The above ensures differentiability and monotonicity of the joint CDF through constraining the derivatives of each of the CDN functions. We note that although it is merely sufficient for the first order derivatives to be non-negative in order for $F(\mathbf{x})$ to be monotonic, the condition that the higher order mixed derivatives of the functions $\phi_s(\mathbf{x}_s)$ be non-negative also implies non-negativity of the first order derivatives. Thus in the sequel, whenever we assume differentiability of CDN functions, we will assume that for all $s \in S$,

all mixed derivatives of $\phi_s(\mathbf{x}_s)$ with respect to any and all subsets of argument variables are non-negative.

Having described the above conditions on CDN functions, we will now provide some examples of CDNs constructed from a product of CDFs.

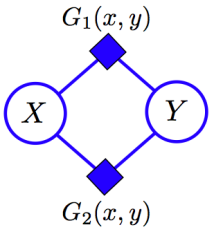
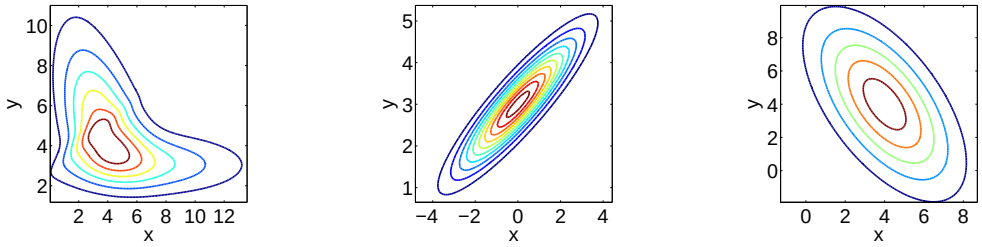


Figure 2: A CDN defined over two variables X and Y with functions $G_1(x,y), G_2(x,y)$.



(a) (b) (c)

Figure 3: a) Joint probability density function $P(x,y)$ corresponding to the distribution function $F(x,y)$ using bivariate Gaussian CDFs as CDN functions; b),c) The PDFs corresponding to $\partial_{x,y}\Big[G_1(x,y)\Big]$ and $\partial_{x,y}\Big[G_2(x,y)\Big]$.

Example 2.1 (Product of bivariate Gaussian CDFs)**.** As a simple example of a CDN, consider two random variables X and Y with joint CDF modeled by the CDN in Figure 2, so that $F(x,y) = G_1(x,y)G_2(x,y)$ with

$$G_1(x,y) = \Phi\left(\begin{bmatrix}x\\y\end{bmatrix}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right), \quad \boldsymbol{\mu}_1 = \begin{bmatrix}\mu_{x,1}\\ \mu_{y,1}\end{bmatrix}, \qquad \boldsymbol{\Sigma}_1 = \begin{bmatrix}\sigma_{x,1}^2 & \rho_1\sigma_{x,1}\sigma_{y,1}\\ \rho_1\sigma_{x,1}\sigma_{y,1} & \sigma_{y,1}^2\end{bmatrix},$$

$$G_2(x,y) = \Phi\left(\begin{bmatrix}x\\y\end{bmatrix}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right), \quad \boldsymbol{\mu}_2 = \begin{bmatrix}\mu_{x,2}\\ \mu_{y,2}\end{bmatrix}, \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix}\sigma_{x,2}^2 & \rho_2\sigma_{x,2}\sigma_{y,2}\\ \rho_2\sigma_{x,2}\sigma_{y,2} & \sigma_{y,2}^2\end{bmatrix},$$

where $\Phi(\cdot; \mathbf{m}, \mathbf{S})$ is the multivariate Gaussian CDF with mean vector $\mathbf{m}$ and covariance $\mathbf{S}$. Taking derivatives, the density $P(x,y)$ is given by

$$\begin{aligned}P(x,y) &= \partial_{x,y}\Big[F(x,y)\Big] = \partial_{x,y}\Big[G_1(x,y)G_2(x,y)\Big]\\ &= G_1(x,y)\partial_{x,y}\Big[G_2(x,y)\Big] + \partial_x\Big[G_1(x,y)\Big]\partial_y\Big[G_2(x,y)\Big]\\ &+ \partial_y\Big[G_1(x,y)\Big]\partial_x\Big[G_2(x,y)\Big] + \partial_{x,y}\Big[G_1(x,y)\Big]G_2(x,y).\end{aligned}$$

As functions G_1, G_2 are Gaussian CDFs, the above derivatives can be expressed in terms of Gaussian CDF and PDFs. For example,

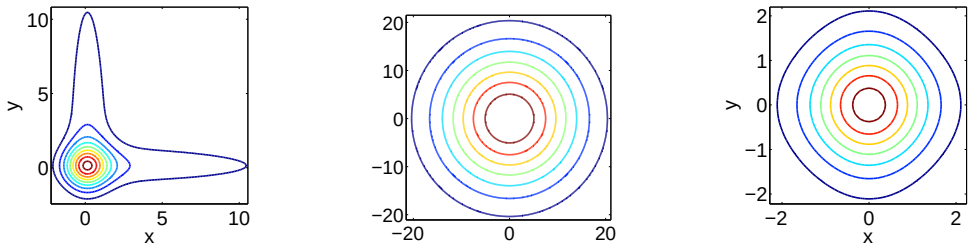
$$\begin{aligned}\partial_x\Big[G_1(x,y)\Big] &= \int_{-\infty}^{y} Gaussian\left(\begin{bmatrix}x\\t\end{bmatrix}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) dt \\ &= Gaussian(x; \mu_{x,1}, \sigma^2_{x,1}) \int_{-\infty}^{y} Gaussian(t; \mu_{y|x,1}, \sigma^2_{y|x,1})\, dt \\ &= Gaussian(x; \mu_{x,1}, \sigma^2_{x,1}) \Phi(y; \mu_{y|x,1}, \sigma^2_{y|x,1}) \qquad (12)\end{aligned}$$

where

$$\begin{aligned}\mu_{y|x,1} &= \mu_{y,1} + \rho_1 \frac{\sigma_{y,1}}{\sigma_{x,1}}(x - \mu_{x,1}) \\ \sigma^2_{y|x,1} &= (1 - \rho_1^2)\sigma^2_{y,1} \qquad (13)\end{aligned}$$

Other derivatives can be obtained similarly. The resulting joint PDF $P(x, y)$ obtained by differentiating the CDF is shown in Figure 3(a), where the CDN function parameters are given by $\mu_{x,1} = 0, \mu_{x,2} = 4, \mu_{y,1} = 3, \mu_{y,2} = 4, \sigma_{x,1} = \sqrt{3}, \sigma_{x,2} = \sqrt{5}, \sigma_{y,1} = 1, \sigma_{y,2} = \sqrt{10}, \rho_1 = 0.9, \rho_2 = -0.6$. The PDFs corresponding to $\partial_{x,y}\Big[G_1(x,y)\Big]$ and $\partial_{x,y}\Big[G_2(x,y)\Big]$ are shown in Figures 3(b) and 3(c). □

The next example provides an illustration of the use of copula functions for constructing multivariate CDFs under the framework of CDNs.



(a) (b) (c)

Figure 4: a) Joint probability density function $P(x, y)$ corresponding to the distribution function $F(x, y)$ using bivariate Gumbel copulas as CDN functions, with Student's-t and Gaussian marginal input CDFs; b),c) The PDFs corresponding to $\partial_{x,y}\Big[G_1(x,y)\Big]$ and $\partial_{x,y}\Big[G_2(x,y)\Big]$.

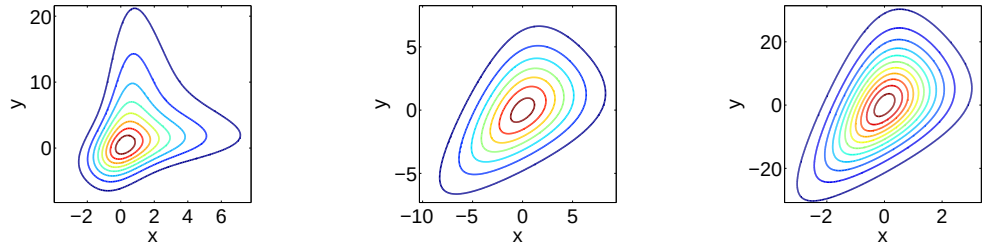
Example 2.2 (Product of copulas)**.** We can repeat the above for the case where each CDN function consists of a copula function (Nelsen, 1999). Copula functions provide a flexible means to construct CDN functions ϕ_s whose product yields a joint CDF under Lemma 2.1. Copula functions allow one to construct a multivariate CDF ϕ_s from marginal CDFs $\{F(x_\alpha)\}_{\alpha\in\mathcal{N}(s)}$ so that

$$\phi_s(\mathbf{x}_s) = \zeta_s\Big(\{F(x_\alpha)\}_{\alpha\in\mathcal{N}(s)}\Big),$$

where ζ_s is a copula defined over variables $X_\alpha, \alpha \in \mathcal{N}(s)$. For the CDN shown in Figure 2, we can set the CDN functions G_1, G_2 to Gumbel copulas so that

$$G_1(x,y) = \zeta_1(H_{1,x}(x), H_{1,y}(y)) = \exp\left(-\left(-\frac{1}{\theta_1}\left(\log H_{1,x}(x) + \log H_{1,y}(y)\right)\right)^{\theta_1}\right),$$
$$G_2(x,y) = \zeta_2(H_{2,x}(x), H_{2,y}(y)) = \exp\left(-\left(-\frac{1}{\theta_2}\left(\log H_{2,x}(x) + \log H_{2,y}(y)\right)\right)^{\theta_2}\right),$$

with $H_{1,x}, H_{2,x}$ set to univariate Gaussian CDFs with parameters $\mu_{1,x}, \mu_{2,x}, \sigma_{1,x}, \sigma_{2,x}$ and $H_{1,y}, H_{2,y}$ set to univariate Student's-t CDFs with parameters $\sigma_{1,y}, \sigma_{2,y}$. One can then verify that the functions G_1, G_2 satisfy the properties of a copula function (Nelsen, 1999) and so the product of G_1, G_2 yields the CDF $F(x,y)$. An example of the resulting joint probability density $P(x,y)$ obtained by differentiation of $F(x,y)$ for parameters $\mu_{1,x} = \mu_{2,x} = 0, \sigma_{1,x} = \sigma_{2,x} = \sigma_{1,y} = \sigma_{2,y} = 10, \theta_1 = \theta_2 = 1$ is shown in Figure 4(a), with the PDFs corresponding to $\partial_{x,y}\Big[G_1(x,y)\Big]$ and $\partial_{x,y}\Big[G_2(x,y)\Big]$ shown in Figures 4(b) and 4(c). □



(a) (b) (c)

Figure 5: a) Joint probability density function $P(x,y)$ corresponding to the distribution function $F(x,y)$ using bivariate sigmoidal functions as CDN functions; b),c) The PDFs corresponding to $\partial_{x,y}\Big[G_1(x,y)\Big]$ and $\partial_{x,y}\Big[G_2(x,y)\Big]$.

Example 2.3 (Product of bivariate sigmoids)**.** As another example of a probability density function constructed using a CDN, consider the case in which functions $G_1(x,y)$ and $G_1(x,y)$ in the CDN of Figure 2 are set to be multivariate sigmoids of the form

$$G_1(x,y) = \frac{1}{1 + \exp(-w_x^1 x) + \exp(-w_y^1 y)}$$
$$G_2(x,y) = \frac{1}{1 + \exp(-w_x^2 x) + \exp(-w_y^2 y)},$$

with $w_x^1, w_y^1, w_x^2, w_y^2$ non-negative. An example of the resulting joint probability density $P(x,y)$ obtained by differentiation of $F(x,y) = G_1(x,y)G_2(x,y)$ for parameters $w_x^1 = 12.5, w_y^1 = 0.125, w_x^2 = 0.4, w_y^2 = 0.5$ is shown in Figure 5(a), with the PDFs corresponding to $\partial_{x,y}\Big[G_1(x,y)\Big]$ and $\partial_{x,y}\Big[G_2(x,y)\Big]$ shown in Figures 5(b) and 5(c). □

The above examples demonstrate that one can construct multivariate CDFs by taking a product of CDFs defined over subsets of variables in the graph.

2.1 Marginal and conditional independence properties

In this section, we will derive the marginal and conditional independence properties for a CDN. We will see that the conditional independence properties for a CDN are distinct from those of Bayesian networks, Markov random fields or factor graphs. To begin, we consider a toy example of the marginal independence property for a three-variable CDN in Figure 6, where variables X and Y are separated by variable Z with respect to graph $\mathcal{G}$, and so are marginally independent. In a CDN, variables that share no neighbors in the CDN graph are marginally independent: we formalize this with the following theorem.

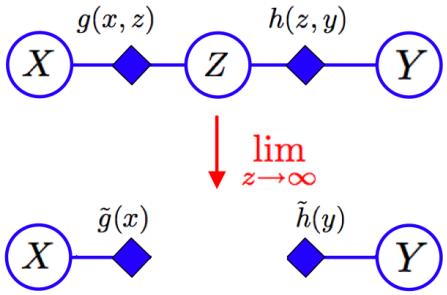


Figure 6: Marginal independence property of CDNs: if two variables X and Y share no common function nodes, they are marginally independent.

Theorem 2.3 (Marginal Independence)**.** *Let $\mathcal{G} = (V, S, E)$ be a CDN and let $A, B \subseteq V$ be disjoint sets of variables. Then $A \perp\!\!\!\perp B$ if $\mathcal{N}(A) \bigcap \mathcal{N}(B) = \emptyset$.* □

Proof. Since $\mathcal{N}(A) \bigcap \mathcal{N}(B) = \emptyset$, we have

$$F(\mathbf{x}) = \prod_{s \in \mathcal{N}(A)} \phi_s(\mathbf{x}_s) \prod_{s \in \mathcal{N}(B)} \phi_s(\mathbf{x}_s) \prod_{s \notin \mathcal{N}(A) \cup \mathcal{N}(B)} \phi_s(\mathbf{x}_s).$$

Marginalizing over all other variables $\mathbf{X}_{V \setminus \{A,B\}}$, we obtain

$$\begin{aligned} F(\mathbf{x}_A, \mathbf{x}_B) &= \lim_{\mathbf{x}_{V \setminus \{A,B\}} \to \infty} F(\mathbf{x}) = \lim_{\mathbf{x}_{V \setminus \{A,B\}} \to \infty} \prod_{s \in S} \phi_s(\mathbf{x}_s) \\ &= \lim_{\mathbf{x}_{V \setminus \{A,B\}} \to \infty} \prod_{s \in \mathcal{N}(A)} \phi_s(\mathbf{x}_s) \prod_{s \in \mathcal{N}(B)} \phi_s(\mathbf{x}_s) \prod_{s \notin \mathcal{N}(A) \cup \mathcal{N}(B)} \phi_s(\mathbf{x}_s) \\ &= \prod_{s \in \mathcal{N}(A)} \lim_{\mathbf{x}_{\mathcal{N}(s) \setminus A} \to \infty} \phi_s(\mathbf{x}_s) \prod_{s \in \mathcal{N}(B)} \lim_{\mathbf{x}_{\mathcal{N}(s) \setminus B} \to \infty} \phi_s(\mathbf{x}_s) \prod_{s \in S \setminus \{\mathcal{N}(A) \cup \mathcal{N}(B)\}} \lim_{\mathbf{x}_{\mathcal{N}(s)} \to \infty} \phi_s(\mathbf{x}_s), \end{aligned}$$

where in the last line we have the used the fact that the limit of a product is equal to the product of limits. Let

$$\begin{aligned} g(\mathbf{x}_A) &= \prod_{s \in \mathcal{N}(A)} \lim_{\mathbf{x}_{\mathcal{N}(s) \setminus A} \to \infty} \phi_s(\mathbf{x}_s) \\ h(\mathbf{x}_B) &= \prod_{s \in \mathcal{N}(B)} \lim_{\mathbf{x}_{\mathcal{N}(s) \setminus B} \to \infty} \phi_s(\mathbf{x}_s). \end{aligned}$$

Since g, h are products of CDFs, they satisfy the properties of a CDF and so by Lemma 2.1, we have $\prod_{s \in S \setminus \{\mathcal{N}(A) \bigcup \mathcal{N}(B)\}} \lim_{\mathbf{x}_{\mathcal{N}(s)} \to \infty} \phi_s(\mathbf{x}_s) = 1$ and $\lim_{\mathbf{x}_A \to \infty} g(\mathbf{x}_A) = \lim_{\mathbf{x}_B \to \infty} h(\mathbf{x}_B) = 1$. Furthermore, it follows that $F(\mathbf{x}_A) = \lim_{\mathbf{x}_B \to \infty} F(\mathbf{x}_A, \mathbf{x}_B) = g(\mathbf{x}_A)$ and $F(\mathbf{x}_B) = \lim_{\mathbf{x}_A \to \infty} F(\mathbf{x}_A, \mathbf{x}_B) = h(\mathbf{x}_B)$ by marginalizing away the appropriate sets of variables. Thus, we have $F(\mathbf{x}_A, \mathbf{x}_B) = F(\mathbf{x}_A)F(\mathbf{x}_B)$ and so $A \perp\!\!\!\perp B$. □

Note that the converse to the above does not generally hold: if disjoint sets A and B do share functions in S, they can still be marginally independent, as one can easily construct a bipartite graph in which variable nodes are not separated in the graph but the function nodes connecting A to B correspond to factorized functions so that $A \perp\!\!\!\perp B$. Having derived the marginal independence property in a CDN, we now consider the conditional independence property of a CDN. To motivate this, we first present a toy example in Figure 7 in which we are given CDNs for variables X, Y, Z, W and we condition on variable Z. Here the separation of X and Y by unobserved variable W implies $X \perp\!\!\!\perp Y|Z$, but separation of X and Y by observed variable Z only implies the marginal independence relationship $X \perp\!\!\!\perp Y$. In general, variable sets that are separated in a CDN by unobserved variables will be conditionally independent given all other variables: thus, as long as two variables are separated by some unobserved variables then they are independent, irrespective of the fact that there may be other variables observed as well. We formalize this conditional independence property with the following theorem.

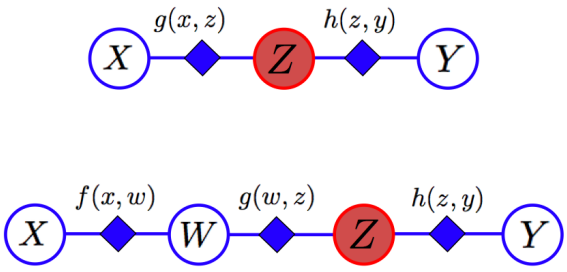


Figure 7: Conditional independence in CDNs. Two variables X and Y are marginally independent given that the variable Z separates X from Y with respect to the graph (*top*). When an unobserved variable W separates X from Y, X, Y are conditionally independent given Z (*bottom*). The bottom graph thus implies $X \perp\!\!\!\perp Y$,$X \perp\!\!\!\perp Z$, $W \perp\!\!\!\perp Y$, $X \perp\!\!\!\perp Y|W$ and $X \perp\!\!\!\perp Y|Z$.

Theorem 2.4 (Conditional independence in CDNs)**.** *Let $\mathcal{G} = (V, S, E)$ be a CDN. For all disjoint sets of $A, B, C \subseteq V$, if C separates A from B relative to graph $\mathcal{G}$ then*

$$A \perp\!\!\!\perp B|V \setminus (A \cup B \cup C).$$

□.

Proof. If C separates A from B, then marginalizing out variables in C yields two disjoint subgraphs with variable sets A', B', with $A \subseteq A'$,$B \subseteq B'$, $A' \cup B' = V \setminus C$ and $\mathcal{N}(A') \bigcap \mathcal{N}(B') = \emptyset$. From Theorem 2.3, we therefore have $A' \perp\!\!\!\perp B'$. Now consider the set

$V \setminus \{A, B, C\}$ and let $\tilde{A}, \tilde{B}$ denote a partition of the set so that

$$\tilde{A} \cup \tilde{B} = V \setminus \{A, B, C\}, \quad \tilde{A} \bigcap \tilde{B} = \emptyset$$
$$\tilde{A} \bigcap B' = \emptyset, \quad \tilde{B} \bigcap A' = \emptyset.$$

From the semi-graphoid axioms (Lauritzen, 1996; Pearl, 1988), $A' \perp\!\!\!\perp B'$ implies $A \perp\!\!\!\perp B | V \setminus \{A, B, C\}$ since $\tilde{A} \subset A'$ and $\tilde{B} \subset B'$. □

An illustration of the proof is provided in Figures 8(a),8(b). As a corollary of the conditional independence property, variables that share no common function nodes in the CDN are also marginally independent, as we now show.

Corollary 2.5. *Let $A, B, C \subseteq V$ be three disjoint node sets so that C separates A from B with respect to $\mathcal{G}$. Then $A \perp\!\!\!\perp B$.* □

The corollary is readily proven by noting that $V \setminus (A \cup B)$ separates sets A and B with respect to $\mathcal{G}$.

Theorem 2.6 (Conditional inequality independence in CDNs)**.** Let $\mathcal{G} = (V, S, E)$ be a CDN and let $A, B \subseteq V$ be disjoint sets of variable nodes. If A and B are separated with respect to $\mathcal{G}$, then for any $W \subseteq V \setminus (A, B)$ $A \perp\!\!\!\perp B | \omega(\mathbf{x}_W)$ where $\omega(\mathbf{x}_W) \equiv \{\mathbf{X}_W \leq \mathbf{x}_W\}$.

Proof. If A and B are separated with respect to $\mathcal{G}$, then we can write

$$F(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_{V \setminus (A,B)}) = g(\mathbf{x}_A, \mathbf{x}_{V \setminus (A,B)}) h(\mathbf{x}_B, \mathbf{x}_{V \setminus (A,B)}) \tag{14}$$

for some functions g, h that satisfy the conditions of Lemma 3.1.1. This then means that $F(\mathbf{x}_A, \mathbf{x}_B | \omega(\mathbf{x}_W))$ is given by

$$\begin{aligned} F(\mathbf{x}_A, \mathbf{x}_B | \omega(\mathbf{x}_W)) &= \frac{\lim\limits_{\mathbf{x}_{V \setminus (A,B,W)} \to \infty} F(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_{V \setminus (A,B)})}{\lim\limits_{\mathbf{x}_{V \setminus W} \to \infty} F(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_{V \setminus (A,B)})} \\ &\propto F(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_W) = g(\mathbf{x}_A, \mathbf{x}_W) h(\mathbf{x}_B, \mathbf{x}_W), \end{aligned} \tag{15}$$

which implies $A \perp\!\!\!\perp B | \omega(\mathbf{x}_W)$. □

The above conditional inequality independence property, in which we condition on *inequality* events of the type $\omega(\mathbf{x}_W)$, is distinct from the conditional independence property described in Theorem 2.4. We show that if a CDF $F(\mathbf{x})$ satisfies the conditional independence property of Proposition 2.6 for a given CDN, then F can be written as a product over functions defined over connected sets.

Theorem 2.7 (Factorization property of a CDN)**.** Let $\mathcal{G} = (V, S, E)$ be a bipartite graph and let the CDF $F(\mathbf{x})$ satisfy the conditional independence property implied by the CDN described by $\mathcal{G}$, so that graph separation of A and B by $V \setminus (A, B)$ with respect to $\mathcal{G}$ implies $A \perp\!\!\!\perp B | \omega(\mathbf{x}_W)$ for any $W \subseteq V \setminus (A, B)$ and for any $\mathbf{x}_W \in \mathbb{R}^{|W|}$. Then there exist functions $\phi_s(\mathbf{x}_s), s \in S$ that satisfy the properties of a CDF such that the joint CDF $F(\mathbf{x})$ factors as $\prod\limits_{s \in S} \phi_s(\mathbf{x}_s)$.

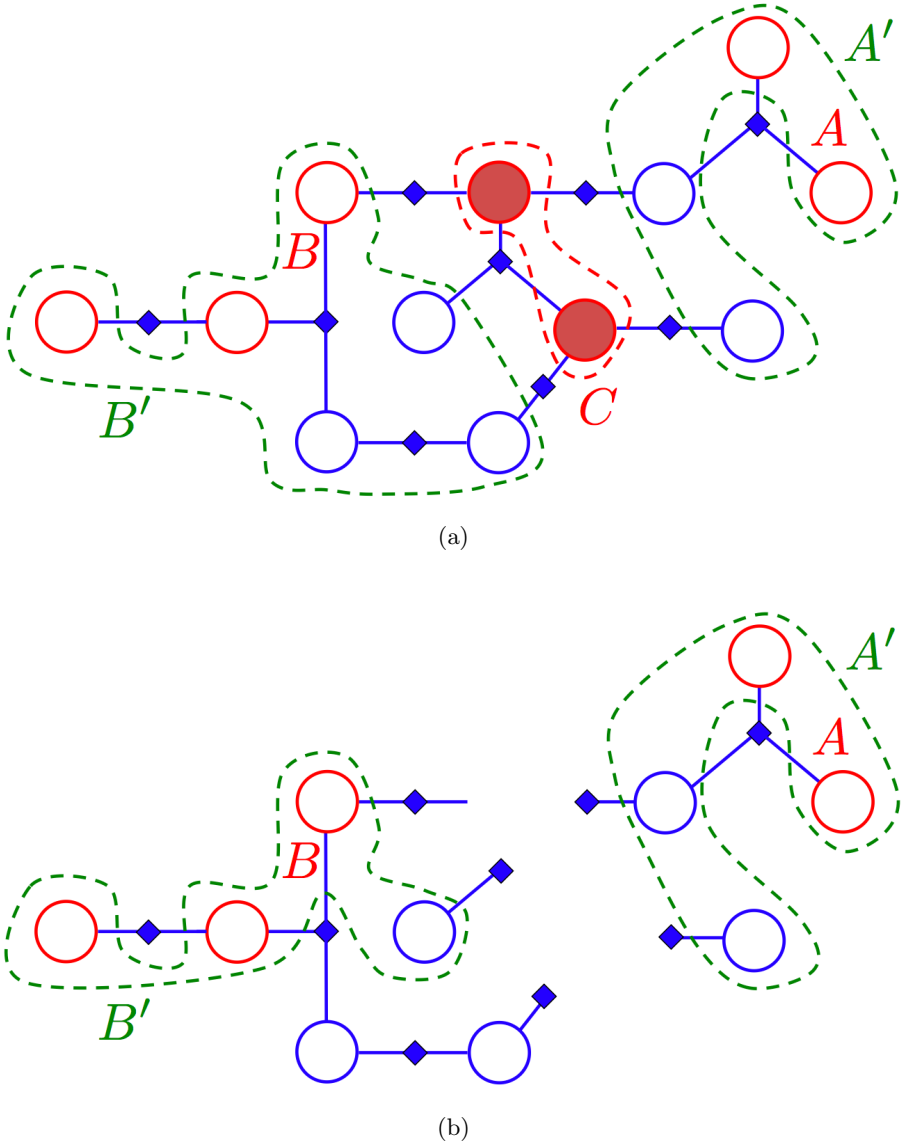


Figure 8: Example of conditional independence due to graph separation in a CDN. a) Given bipartite graph $\mathcal{G} = (V, S, E)$, node set C separates set A from B (nodes in red) with respect to $\mathcal{G}$. Furthermore, we have for A', B' (nodes in green dotted line) $A \subseteq A', B \subseteq B'$, $A' \cup B' = V \setminus C$ and $\mathcal{N}(A') \bigcap \mathcal{N}(B') = \emptyset$ as shown. b) Marginalizing out variables corresponding to nodes in C yields two disjoint subgraphs of $\mathcal{G}$ and so $A \perp\!\!\!\perp B | V \setminus \{A, B, C\}$.

Proof. The proof here parallels that for the Hammersley-Clifford theorem for undirected graphical models Lauritzen (1996). We begin our proof by defining $\psi_U(\mathbf{x}), \zeta_U(\mathbf{x})$ as functions that depend only on variable nodes in some set $U \subseteq V$ and that form a Möbius transform

pair

$$\psi_U(\mathbf{x}) = \sum_{W \subseteq U} \zeta_W(\mathbf{x}) \tag{16}$$

$$\zeta_U(\mathbf{x}) = \sum_{W \subseteq U} (-1)^{|U \setminus W|} \psi_W(\mathbf{x}), \tag{17}$$

where we take $\psi_U(\mathbf{x}) \equiv \log F(\mathbf{x}_U)$. Now, we note that $F(\mathbf{x})$ can always be written as a product of functions $\prod_{U \subseteq V} \phi_U(\mathbf{x})$ where each function ϕ_U satisfies the properties of a CDF: a trivial example of this is to set $\phi_V(\mathbf{x}) = F(\mathbf{x})$ and $\phi_U(\mathbf{x}) = 1$ for all $U \subset V$. Since by hypothesis F satisfies all of the conditional independence properties implied by the CDN described by $\mathcal{G}$, if we take $\phi_U(\mathbf{x}) = \exp\big(\zeta_U(\mathbf{x})\big)$, then it suffices to show that $\zeta_U(\mathbf{x}) \equiv 0$ for subsets of variable nodes U for which any two non-neighboring variable nodes $\alpha, \beta \in U$ are separated such that $\alpha \perp\!\!\!\perp \beta | \omega(\mathbf{x}_W)$ for any $W \subseteq U \setminus (\alpha, \beta)$. Observe that we can write $\zeta_U(\mathbf{x})$ as

$$\begin{aligned}\zeta_U(\mathbf{x}) &= \sum_{W \subseteq U} (-1)^{|U \setminus W|} \psi_W(\mathbf{x}) \\ &= \sum_{W \subseteq U \setminus (\alpha \cup \beta)} (-1)^{|U \setminus W|} \Big(\psi_W(\mathbf{x}) - \psi_{W \cup \alpha}(\mathbf{x}) - \psi_{W \cup \beta}(\mathbf{x}) + \psi_{W \cup \alpha \cup \beta}(\mathbf{x}) \Big). \end{aligned} \tag{18}$$

If $\alpha, \beta \in U$ are separated and $W \subseteq U \setminus \{\alpha, \beta\}$, then $\alpha \perp\!\!\!\perp \beta | \omega(\mathbf{x}_W)$ and

$$\begin{aligned}\psi_{W \cup \alpha \cup \beta}(\mathbf{x}) - \psi_{W \cup \alpha}(\mathbf{x}) &= \log \frac{F(x_\alpha, x_\beta, \mathbf{x}_W)}{F(x_\alpha, \mathbf{x}_W)} = \log \frac{F(x_\alpha | \omega(\mathbf{x}_W)) F(x_\beta | \omega(\mathbf{x}_W)) F(\mathbf{x}_W)}{F(x_\alpha | \omega(\mathbf{x}_W)) F(\mathbf{x}_W)} \\ &= \log \frac{F(x_\beta | \omega(\mathbf{x}_W)) F(\mathbf{x}_W)}{F(\mathbf{x}_W)} \\ &= \log F(x_\beta, \mathbf{x}_W) - \log F(\mathbf{x}_W) \\ &= \psi_{W \cup \beta}(\mathbf{x}) - \psi_W(\mathbf{x}). \end{aligned} \tag{19}$$

Thus if U is any set where nodes $\alpha, \beta \in U$ are separated, then for all $W \subseteq U \setminus (\alpha \cup \beta)$ we must have $\psi_W(\mathbf{x}) - \psi_{W \cup \alpha}(\mathbf{x}) - \psi_{W \cup \beta}(\mathbf{x}) + \psi_{W \cup \alpha \cup \beta}(\mathbf{x}) \equiv 0$ and so $\zeta_U(\mathbf{x}) = 0$. Since $F(\mathbf{x}) = \exp(\psi_V(\mathbf{x})) = \exp\Big(\sum_U \zeta_U(\mathbf{x})\Big) = \prod_U \phi_U(\mathbf{x})$ where the product is taken over subsets of variable nodes U that are not separated, and noting that for each s, variable nodes in $\mathcal{N}(s)$ are not separated, we can then substitute $\phi_U(\mathbf{x}) = \phi_s(\mathbf{x}_s)$ into the product with $U = \mathcal{N}(s)$. Thus we can write $F(\mathbf{x}) = \prod_{s \in S} \phi_s(\mathbf{x}_s)$, where each function ϕ_s is defined over the set of variable nodes $\mathcal{N}(s)$. □

Thus, if $F(\mathbf{x})$ satisfies the conditional independence property where graph separation of A and B with respect to $\mathcal{G}$ implies $A \perp\!\!\!\perp B | \omega(\mathbf{x}_W)$ for any $W \subseteq V \setminus (A, B)$, then F can be written as a product of functions of the form $\prod_{s \in S} \phi_s(\mathbf{x}_s)$. The above theorem then demonstrates equivalence between the conditional independence property $A \perp\!\!\!\perp B | \omega(\mathbf{x}_W)$

and the factored form for $F(\mathbf{x})$. This means that $A \perp\!\!\!\perp B|\omega(\mathbf{x}_W)$ in turn implies both $A \perp\!\!\!\perp B$ and $A \perp\!\!\!\perp B|V \setminus (A, B, C)$ for disjoint sets $A, B, C \subseteq V$ where C separates A from B with respect to $\mathcal{G}$. As the latter two properties are those of bi-directed graphical models, the independence properties of CDNs then include those of bi-directed graphical models Drton and Richardson (2008).

In addition to the above, both the conditional independence properties of Theorem 2.4 and Proposition 2.6 are closed under marginalization, which consists of computing limits of CDN functions. Thus if $\mathcal{G}$ is a CDN model for $F(\mathbf{x})$, then the graph for CDN for CDF $F(\mathbf{x}_A) = \lim_{\mathbf{x}_{V \setminus A} \to \infty} F(\mathbf{x}_A, \mathbf{x}_{V \setminus A})$ is given by a subgraph of $\mathcal{G}$ which then implies only a subset of the independence properties of $\mathcal{G}$. The next proposition formalizes this.

Proposition 2.8. *Let $\mathcal{G} = (V, S, E)$ be a CDN and let $A, B, C \subset V$ be disjoint sets of nodes with $C \subseteq V \setminus \{A, B\}$ separating A from B with respect to $\mathcal{G}$. Let $\mathcal{G}' = (V', S', E')$ be a subgraph of $\mathcal{G}$ with $V' \subseteq V, S' \subseteq S, E' \subseteq E$. Similarly, let $A' = A \cap V', B' = B \cap V', C' = C \cap V'$ be disjoint sets of nodes. Then C' separates A' from B' with respect to $\mathcal{G}'$.* □

As a result, the conditional independence relation $A' \perp\!\!\!\perp B'|V' \setminus (A' \cup B' \cup C')$ must also hold in the subgraph $\mathcal{G}'$, such that $\mathcal{G}'$ implies a subset of the independence constraints implied by $\mathcal{G}$. The above closure property under marginalization is a property that also holds for Markov random fields, but not for Bayesian networks (see Richardson and Spirtes (2002) for an example). The above closure and conditional independence properties for CDNs have also been previously shown to hold for bidirected graphs as well, which we will now describe.

2.2 The relationship between cumulative distribution networks and bidirected graphs

Graphical models with some of the independence properties of CDNs have in fact been studied previously in the statistics literature. The connected set property for CDNs presented in Theorem 2.7 is in fact identical to the connected set property of (Richardson and Spirtes, 2002), which was also derived in the context of bidirected graphical models (Drton and Richardson, 2008; Richardson and Spirtes, 2002; Richardson, 2003), which are graphical models where the lack of an edge between two nodes implies a marginal independence constraint. We provide a formal definition for a bi-directed graphical model below.

Definition 2.2. Let $G = (V, E)$ be a graph consisting of nodes $\alpha \in V$ and *bi-directed* edges $e \in E$ consisting of unordered pairs of nodes α, β, denoted by (α, β). For arbitrary joint probability P defined on random variables X_α, G is a bi-directed graphical model for P if $(\alpha, \beta) \notin E \Leftrightarrow \alpha \perp\!\!\!\perp \beta$.

Alternately, we denote edges in a bi-directed graph as $(\alpha, \beta) \equiv \alpha \leftrightarrow \beta$. Note that $\alpha \leftrightarrow \beta$ is *not* equivalent to having both directed edges $\alpha \rightarrow \beta$ and $\alpha \leftarrow \beta$. It can be shown (Richardson and Spirtes, 2002) that any bi-directed graphical model corresponds to a directed graphical model with latent variables marginalized out. In particular, we define the *canonical* directed acyclic graph (DAG) for the bi-directed graph G as a directed graph $\tilde{G}$ with additional latent variables such that if $\alpha \leftrightarrow \beta$ in G, then $\alpha \leftarrow u_{\alpha,\beta} \rightarrow \beta$ in $\tilde{G}$ for some latent variable $u_{\alpha,\beta}$. Thus bi-directed graphical models can be viewed as models obtained

from a corresponding canonical DAG with latent variables marginalized out, such that independence constraints between neighboring variable nodes in G can be viewed as arising from the absence of any shared latent variables in the canonical DAG $\tilde{G}$. This suggests the usefulness of bi-directed graphical models for problems where we cannot discount the presence of unobserved variables but we either A) do not have sufficient domain knowledge to specify distributions containing latent variables, and/or B) we wish to avoid marginalizing over these latent variables. In such cases, one can instead attempt to parameterize a probability define on observed variables using a bi-directed graphical model in which independence constraints among variables are implied by both the corresponding canonical DAG and bi-directed graphs. Examples of a canonical DAG and corresponding bi-directed graph that imply the same set of independence constraints among observed variables are shown in Figures 9(a),9(b). Several parameterizations had been previously proposed for bidirected graphical models. Covariance graphs (Kauermann, 1996) were proposed in which variables are jointly Gaussian with zero pairwise covariance if there is no edge connecting the two variables in the bidirected graph. In addition, (Silva and Ghahramani, 2009a) proposed a mixture model with latent variables in which dependent variables in the bidirected graph can be explained by the causal influence of common components in the mixture model. For bidirected graphical models defined over binary variables, a parameterization was proposed based on joint probabilities over connected components of the bidirected graph so that the joint probability of any subset of variables could be obtained by Möbius inversion (Drton and Richardson, 2008).

Suppose now that we are given a bi-directed graph G and a CDN $\mathcal{G}$ defined over the same variables nodes V. Let G and $\mathcal{G}$ have the same connectivity, such that for any pair of variable nodes $\alpha, \beta \in V$, a path between α, β exists both in G and $\mathcal{G}$. Then both G and $\mathcal{G}$ imply the same set of marginal independence constraints, as we have shown above that in a CDN, two nodes that do not share any function nodes in common are marginally independent (Theorem 2.3). An example of a bidirected graph and CDN that imply the same set of marginal independence constraints is shown in Figures 9(b), 9(c). In addition to implying the same marginal independence constraints as a bi-directed graphical model, the conditional independence property given in Theorem 2.4 for CDNs corresponds to the dual global Markov property of (Kauermann, 1996) for bidirected graphical models, which we now present.

Theorem 2.9. *Let $G = (V, E)$ be a bi-directed graphical model and let $A, B, C \subseteq V$ be three disjoint node sets so that $V \setminus (A \cup B \cup C)$ separates A from B with respect to G. Then $A \perp\!\!\!\perp B | C$.* □

While the conditional and marginal independence constraints implied by both a bi-directed graph and a CDN of the same connectivity are identical, Proposition 2.6 shows that an additional set of conditional independence constraints of the form $A \perp\!\!\!\perp B | \omega(\mathbf{x}_W)$ holds for CDNs but not for bi-directed graphs. As a result, CDNs model a subset of the distributions that satisfy the independence constraints of a corresponding bi-directed graph with the same connectivity. In general, CDNs do not model the full set of the probability distributions that can be modelled by bi-directed graphical models with the same connectivity. However, for probabilities that can be modelled by any of CDN, bi-directed graph or corresponding canonical DAG models, working with such models using

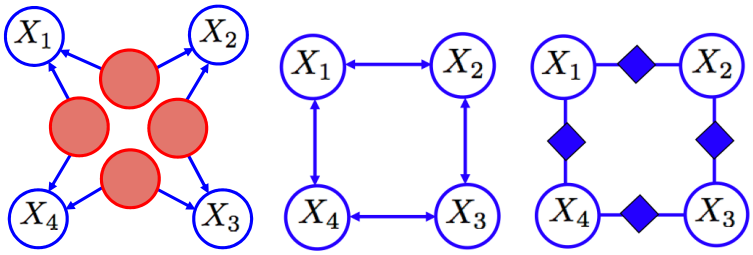


Figure 9: Graphical models over four variables X_1, X_2, X_3, X_4 in which graph separation of variable nodes imply the marginal independence relations $X_1 \perp\!\!\!\perp X_3, X_2 \perp\!\!\!\perp X_4$. a) A canonical directed acyclic graphical model with additional latent variables, shown as shaded nodes; b)A bidirected graph; b) A corresponding CDN.

bi-directed graphs or canonical DAGs will often be difficult, as in the case of the former there currently exist only parameterizations for Gaussian and multinomial models and in the latter case, we must deal with intractable marginalization operations. In such cases CDNs will offer a class of models that will be relatively easier to parameterize and for which inference and learning will be easier compared to having to perform marginalization of latent variables.

In the case of CDNs defined over discrete variables, there is an additional conditional independence property that is also not implied in bidirected graphical models. For a CDN defined over discrete variables taking values in an ordered set $\mathcal{X} = \{r_1, \cdots, r_K\}$, conditioning on the event $\mathbf{X}_C = r_1\mathbf{1}$ yields conditional independence between disjoint sets $A, B, C \subseteq V$ in which C separates A, B with respect to $\mathcal{G}$. We define the corresponding *min-independence* property below.

Definition 2.3 (Min-independence)**.** *Let $\mathbf{X}_A, \mathbf{X}_B, \mathbf{X}_C$ be sets of ordinal discrete variables that take on values in the totally ordered alphabet $\mathcal{X}$ with minimum element $r_1 \in \mathcal{X}$ defined as $r_1 \prec \alpha \; \forall \, \alpha \neq r_1, \alpha \in \mathcal{X}$. $\mathbf{X}_A$ and $\mathbf{X}_B$ are said to be min-independent given $\mathbf{X}_C$ if*

$$\mathbf{X}_A \perp\!\!\!\perp \mathbf{X}_B | \mathbf{X}_C = r_1\mathbf{1},$$

where $r_1\mathbf{1} = [r_1 \;\cdot\; r_1]^T$. □

Theorem 2.10 (Min-independence property of CDNs)**.** *Let $\mathcal{G} = (V, S, E)$ be a CDN defined over ordinal discrete variables that take on values in the totally ordered alphabet $\mathcal{X}$ with minimum element $r_1 \in \mathcal{X}$ defined as $r_1 \prec \alpha \; \forall \, \alpha \neq r_1, \alpha \in \mathcal{X}$. Let $A, B, C \subseteq V$ be arbitrary disjoint subsets of V, with C separating A, B with respect to $\mathcal{G}$. Then $\mathbf{X}_A$ and $\mathbf{X}_B$ are min-independent given $\mathbf{X}_C$.* □

Proof. Since C separates A from B with respect to $\mathcal{G}$, we can write

$$F(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = \phi(\mathbf{x}_A, \mathbf{x}_C)\psi(\mathbf{x}_B, \mathbf{x}_C).$$

The conditional CDF $F(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C = r_1\mathbf{1})$ is then given by

$$\begin{aligned} F(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C = r_1\mathbf{1}) &= \frac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \bigcap \{\mathbf{X}_B \leq \mathbf{x}_B\} \bigcap \{\mathbf{X}_C = r_1\mathbf{1}\}\Big]}{\mathbb{P}\Big[\mathbf{X}_C = r_1\mathbf{1}\Big]} \\ &= \frac{\mathbb{P}\Big[\{\mathbf{X}_A \leq \mathbf{x}_A\} \bigcap \{\mathbf{X}_B \leq \mathbf{x}_B\} \bigcap \{\mathbf{X}_C \leq r_1\mathbf{1}\}\Big]}{\mathbb{P}\Big[\mathbf{X}_C \leq r_1\mathbf{1}\Big]} \\ &\propto \phi(\mathbf{x}_A, r_1\mathbf{1})\psi(\mathbf{x}_B, r_1\mathbf{1}). \end{aligned}$$

and so $\mathbf{X}_A \perp\!\!\!\perp \mathbf{X}_B|\mathbf{X}_C = r_1\mathbf{1}$. □

Thus, in the case of a CDN defined over discrete variables where each variable can have values in the totally ordered alphabet $\mathcal{X}$, a finite difference with respect to variables $\mathbf{X}_C$, when evaluated at the vector of minimum elements $\mathbf{X}_C = r_1\mathbf{1}$ is equivalent to directly evaluating the CDF at $\mathbf{X}_C = r_1\mathbf{1}$. Thus CDNs defined over discrete variables admit an additional set of rules for assessing conditional independence between sets of variables than those for bidirected graphs defined over discrete variables. This means that in the case of models defined over ordinal discrete variables, the particular set of conditional independence relationships amongst variables in the model is determined as a function of the ordering over possible labels for each variable in the model, so that one must exercise care in how such variables are labelled and what ordering is satisfied by such labels.

2.3 Stochastic orderings in a cumulative distribution network

The CDN, in providing a graphical model for the joint CDF over many random variables, also allows one to easily specify *stochastic ordering* constraints between subsets of variables in the model. Informally, a stochastic ordering relationship $X \preceq Y$ holds between two random variables X, Y if samples of Y tend to be larger than samples of X. We will focus here on first-order stochastic ordering constraints (Lehmann, 1955; Shaked and Shanthikumar, 1994) of the form $X \preceq Y$ and how one can specify such constraints in terms of the CDN functions in the model. We note that such constraints are not a necessary part of the definition for a CDN or for a multivariate CDF, so that the graph for the CDN alone does not allow one to inspect stochastic ordering constraints based on graph separation of variables. However, the introduction of stochastic ordering constraints, in combination with separation of variables with respect to the graph, do impose constraints on the products of CDN functions, as we will now show. We will define below the concept of first-order stochastic orderings among random variables, as this is the primary definition for a stochastic ordering that we will make use of. We refer the reader to (Lehmann, 1955; Shaked and Shanthikumar, 1994) for additional definitions of stochastic orderings.

Definition 2.4. *Consider two scalar random variables X and Y with marginal CDFs $F_X(x)$ and $F_Y(y)$. Then X and Y are said to satisfy the first-order stochastic ordering constraint $X \preceq Y$ if $F_X(t) \geq F_Y(t)$ for all $t \in \mathbb{R}$.* □

The above definition of stochastic ordering is stronger than the constraint $\mathbb{E}[X] \leq \mathbb{E}[Y]$ which is often used and one can show that $X \preceq Y$ implies the former constraint. Note

that the converse is not true: $\mathbb{E}[X] \leq \mathbb{E}[Y]$ does not necessarily imply $X \preceq Y$. For example, consider two Gaussian random variables X and Y for which $\mathbb{E}[X] \leq \mathbb{E}[Y]$ but $Var[X] \gg Var[Y]$. The definition of a stochastic ordering can also be extended to disjoint sets of variables $\mathbf{X}_A$ and $\mathbf{X}_B$.

Definition 2.5. *Let $\mathbf{X}_A$ and $\mathbf{X}_B$ be disjoint sets of variables so that $\mathbf{X}_A = \{X_{\alpha_1}, \cdots, X_{\alpha_K}\}$ and $\mathbf{X}_B = \{X_{\beta_1}, \cdots, X_{\beta_K}\}$ for some strictly positive integer K. Let $F_{\mathbf{X}_A}(\mathbf{t})$ and $F_{\mathbf{X}_B}(\mathbf{t})$ be the CDFs of $\mathbf{X}_A$ and $\mathbf{X}_B$. Then $\mathbf{X}_A, \mathbf{X}_B$ are said to satisfy the stochastic ordering relationship $\mathbf{X}_A \preceq \mathbf{X}_B$ if*

$$F_{\mathbf{X}_A}(\mathbf{t}) \geq F_{\mathbf{X}_B}(\mathbf{t})$$

for all $\mathbf{t} \in \mathbb{R}^K$. □

Having defined stochastic orderings, we will now present the corresponding constraints on CDN functions which are implied by the above definitions.

Proposition 2.11. *Let $\mathcal{G} = (V, S, E)$ be a CDN, with $A, B \subset V$ so that $A = \{\alpha_1, \cdots, \alpha_K\}$ and $B = \{\beta_1, \cdots, \beta_K\}$ for some strictly positive integer K. Let $\mathbf{t} \in \mathbb{R}^K$. Then A, B satisfy the stochastic ordering relationship $\mathbf{X}_A \preceq \mathbf{X}_B$ if and only if*

$$\prod_{s \in \mathcal{N}(A)} \lim_{\mathbf{u}_{\mathcal{N}(s) \setminus A} \to \infty} \phi_s(\mathbf{u}_{\mathcal{N}(s) \setminus A}, \mathbf{t}_{\mathcal{N}(s) \bigcap A}) \geq \prod_{s \in \mathcal{N}(B)} \lim_{\mathbf{u}_{\mathcal{N}(s) \setminus B} \to \infty} \phi_s(\mathbf{u}_{\mathcal{N}(s) \setminus B}, \mathbf{t}_{\mathcal{N}(s) \bigcap B})$$

for all $\mathbf{t} \in \mathbb{R}^K$. □

The above can be readily obtained by marginalizing over variables in $V \setminus A, V \setminus B$ respectively to obtain expressions for $F(\mathbf{x}_A), F(\mathbf{x}_B)$ as products of CDN functions. The corresponding ordering then holds from Definition 2.5 if and only if $F_{\mathbf{X}_A}(\mathbf{t}) \geq F_{\mathbf{X}_B}(\mathbf{t})$ for all $\mathbf{t} \in \mathbb{R}^K$.

2.4 Discussion

We have presented the CDN and sufficient conditions on the functions in the CDN in order for the CDN to model to a CDF. We have shown that the conditional independence relationships that follow from graph separation in CDNs are different from the relationships implied by graph separation in Bayesian networks, Markov random fields and factor graph models. We have shown that the conditional independence properties of CDNs include, but are not limited to, the marginal independence properties of bidirected graphs, such that CDNs model a subset of all probability distributions that could be modelled by bi-directed graphs. For ordinal discrete random variables, an additional independence property called min-independence holds in which two disjoint sets of variables A and B are conditionally independent of one another if one observes all variables in set C being equal to the minimum element of the poset.

Although it was shown that CDNs can model a subset of the distributions that could be modelled by bi-directed graphical models, CDNs will in general be easier to work with as compared to the corresponding canonical DAG model where one is required to marginalize over many latent variables. In the next section, we will describe why it is that CDNs are easier to work with than corresponding graphical models with latent variables. The

fundamental reason for this is that A) in a CDN, marginalization consists of computing limits, unlike marginalization in models for probability densities and B) conditioning on observations in a CDN consists of computing derivatives, which are significantly easier to compute compared to the integrals/sums required for computing conditionals in models for probability densities.

3. The derivative-sum-product algorithm

In the previous section, we showed that for a joint CDF, we could compute conditional probabilities of the forms $F(\mathbf{x}_A|\omega(\mathbf{x}_B)), F(\mathbf{x}_A|\mathbf{x}_B), P(\mathbf{x}_A|\omega(\mathbf{x}_B))$ and $P(\mathbf{x}_A|\mathbf{x}_B)$, in addition to probabilities of the type $P(\mathbf{x}_A), F(\mathbf{x}_A)$. In directed, undirected or factor graphs, computing and evaluating such conditional CDFs/PDFs would generally require us to integrate over several variables, which may be an intractable operation requiring sampling methods or approximation schemes. However in a CDN, computing and evaluating such conditionals is comparatively easier, as we can compute the relevant quantities by *differentiating* the joint CDF and then evaluating the total mixed derivative for any given vector of observations $\mathbf{x}$. In this section we will show that if we model the joint CDF using a CDN with a tree-structured graph, then we can derive a class of message-passing algorithms called called *derivative-sum-product* (DSP) for efficiently computing and evaluating derivatives in CDNs. Since that the CDF factorizes for a CDN, the global mixed derivative can then be decomposed into a series of local mixed derivative computations, where each function $s \in S$ and its derivatives can be readily evaluated for observations $\mathbf{x}_s$ for any functions ϕ_s that satisfy the properties of a CDF. Throughout this section, we will assume that the sufficient conditions for the CDN functions $\phi_s(\mathbf{x}_s)$ hold in order for the CDN to model a valid joint CDF (Lemma 2.1). We will further assume that the derivatives/finite differences of CDN functions $\phi_s(\mathbf{x}_s)$ with respect to all subsets of argument variables all exist and that the order of differentiation does not affect the computation of any mixed derivatives. In the case where we are differentiating with respect to a set of variables $\mathbf{X}_C$ which are observed with values $\mathbf{x}_C$, we assume that the resulting derivative/finite difference is evaluated at the observed values $\mathbf{x}_C$. In the case where we are given a function $G(x)$ defined over a single ordinal discrete variable $x \in \mathcal{X}$ where $\mathcal{X} = \{r_0, r_1, \cdots, r_{N-1}\}$ and $r_0 < r_1 \cdots < r_{N-1}, r_i \in \mathbb{R}$ are N real-valued scalars, we define the finite difference of G with respect to x, evaluated at x as

$$\partial_x\Big[G(x)\Big] = \begin{cases} G(r_0) & \text{if } x = r_0 \\ G(r_i) - G(r_{i-1}) & \text{if } x = r_i,\ i = 1, \cdots, N-1 \end{cases}$$

3.1 Differentiation in cumulative distribution networks

We first consider the problem of computing the marginal CDF $F(x_\alpha)$ for particular variable X_α. We note that in the CDN, marginalization is a simple procedure that involves taking limits with respect to the variables in the network, so that if we let

$$F(\mathbf{x}) = F(x_\alpha, \mathbf{x}_{V\setminus\alpha}) = \prod_{s\in\mathcal{N}(\alpha)} \phi_s(x_\alpha, \mathbf{x}_{\mathcal{N}(s)\setminus\alpha}) \prod_{s\notin\mathcal{N}(\alpha)} \phi_s(\mathbf{x}_s),$$

then the marginal CDF for X_α is given by

$$F(x_\alpha) = \lim_{\mathbf{x}_{V\setminus\alpha}\to\infty} F(x_\alpha, \mathbf{x}_{V\setminus\alpha}) = \prod_{s\in\mathcal{N}(\alpha)} \phi_s(x_\alpha, \infty) \prod_{s\notin\mathcal{N}(\alpha)} \phi_s(\infty) = \prod_{s\in\mathcal{N}(\alpha)} \phi_s(x_\alpha, \infty).$$

Thus for any x_α, we can obtain any distribution of the type $F(\mathbf{x}_A)$ in time $O(|S||V|)$ by taking the product of limits of functions $\lim_{\mathbf{x}_{\mathcal{N}(s)\setminus\alpha}\to\infty} \phi_s(x_\alpha, \mathbf{x}_{\mathcal{N}(s)\setminus\alpha}) = \phi_s(x_\alpha, \infty)$. Furthermore, we can compute any conditional cumulative distribution of the type $F(\mathbf{x}_A|\omega(\mathbf{x}_B))$ in the same fashion by marginalizing the joint CDF over variables in $V \setminus \{A, B\}$ and computing

$$F(\mathbf{x}_A|\omega(\mathbf{x}_B)) = \frac{F(\mathbf{x}_A, \mathbf{x}_B)}{F(\mathbf{x}_B)} = \frac{\lim_{\mathbf{x}_{V\setminus\{A,B\}}\to\infty} F(\mathbf{x})}{\lim_{\mathbf{x}_{V\setminus B}} F(\mathbf{x})}.$$

Note that the above marginalization contrasts with the problem of exact inference in density models, where potentially intractable marginalization operations must be performed locally for each variable node in order to obtain the desired marginals.

Although obtaining marginals in the CDN is relatively simple, computing and evaluating probability distributions of the form $F(\mathbf{x}_A|\mathbf{x}_B), P(\mathbf{x}_A|\omega(\mathbf{x}_B)), P(\mathbf{x}_A|\mathbf{x}_B)$ and $P(\mathbf{x}_A)$ is more involved. We have seen previously that in order to compute conditional CDFs, we must compute corresponding higher-order derivatives with respect to these observed variables. Fortunately, computing and evaluating derivatives is generally tractable compared to the marginalization operation in probability densities. In particular, given observed data we may wish to numerically evaluate probabilities under the model, such that computing derivatives for each function ϕ_s requires that we store only the numerical value for the derivatives. Provided that the CDN functions are chosen to be differentiable, computing derivatives of these functions will consist simply of function evaluations and will be tractable, in contrast with the generally intractable problem of evaluating exact probabilities for models of probability density functions with latent variables.

Since the factorization of the joint CDF modelled by a CDN consists of a product of functions $\phi_s(\mathbf{x}_s)$, the intuition here is that we can distribute the differentiation operation such that at each function node in the CDN, we compute the derivatives with respect to local variables and passes the result to its neighbors. The resulting algorithm consists of passing messages $\mu_{\alpha\to s}(\mathbf{x}), \mu_{s\to\alpha}(\mathbf{x})$ from variable nodes to function nodes and from function nodes to variable nodes, analogous to the operation of the sum-product algorithm in factor graphs. In the Appendix, we present the derivation of the algorithm in the setting where we wish to compute the mixed derivative of the CDF $F(\mathbf{x})$ modelled by a tree-structured CDN: the derivation is analogous to the derivation for the sum-product algorithm, but with the summation operator replaced by the differentiation operator. To illustrate the corresponding message-passing algorithm, consider the following toy example.

Example 3.1. Consider the toy example of a CDN over four random variables U, X, Y, Z from Figure 10. The joint CDF is given by $F(u, x, y, z) = g(u, x, y)h(y, z)$. Let Z be the root node so that X and U are leaf nodes. Then the messages from leaves to root are given

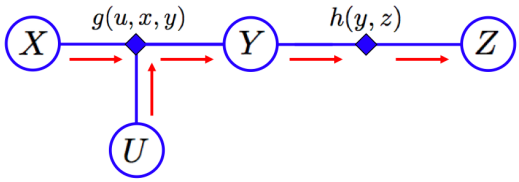


Figure 10: Flow of messages in the toy example of CDN defined over variables X, Y, Z, U.

by

$$\begin{aligned}
&\mu_{X\to g}(x) = 1 \\
&\mu_{U\to g}(u) = 1 \\
&\mu_{g\to Y}(y;u,x) = \partial_{u,x}\Big[g(u,x,y)\mu_{X\to g}(x)\mu_{U\to g}(u)\Big] \\
&\mu_{Y\to h}(y;u,x) = \mu_{g\to Y}(y;u,x) \\
&\mu_{h\to Z}(z;u,x,y) = \partial_{y}\Big[h(y,z)\mu_{Y\to h}(y;u,x)\Big].
\end{aligned}$$

Figure 10 shows the flow of the above messages.

Once we have propagated messages from the leaf nodes to the root node, we can evaluate the joint probability $P(u,x,y,z) = \partial_z\Big[\mu_{h\to Z}(z;u,x,y)\Big]$ at the root node so that

$$\begin{aligned}
P(u,x,y,z) = \partial_z\Big[\mu_{h\to Z}(z;u,x,y)\Big] &= \partial_z\Big[\partial_y\Big[h(y,z)\mu_{Y\to h}(y;u,x)\Big]\Big] \\
&= \partial_z\Big[\partial_y\Big[h(y,z)\mu_{g\to Y}(y;u,x)\Big]\Big] \\
&= \partial_z\Big[\partial_y\Big[h(y,z)\partial_{u,x}\Big[g(u,x,y)\mu_{X\to g}(x)\mu_{U\to g}(u)\Big]\Big]\Big] \\
&= \partial_{x,y,z,u}\Big[g(u,x,y)h(y,z)\Big] \\
&= \partial_{x,y,z,u}\Big[F(u,x,y,z)\Big].
\end{aligned}$$

The above example illustrates the fact that if the graph topology is a tree, then the message-passing algorithm yields the correct mixed derivatives with respect to each variable in the CDN so that we obtain the joint probability $P(\mathbf{x}) = \partial_{\mathbf{x}}\Big[F(\mathbf{x})\Big]$ at the root node of the tree by multiplying all incoming messages at that node.

The above example also illustrates a potential source for complexity: each message consists of a symbolic expression that is a sum of products of derivatives of CDN functions. For larger graphs, it is easy to see that such a message-passing scheme would grow in complexity as the symbolic expression for each message would grow in size as we pass from leaf nodes to the root. However, for practical purposes in which we wish to obtain numerical values for probabilities at the observed data, we are interested in *evaluating* derivatives corresponding to marginal/conditional probabilities for observed values of data $\mathbf{x}$, with unobserved variables marginalized out by taking limits. As the message-passing algorithm allows us to decompose the total mixed derivative computation into a series of local computations, each term in this decomposition consists of a derivative that can

be "clamped" to the observed values for its arguments. Moreover, this "clamping" need only be performed locally for each CDN function as we evaluate each outgoing message. In the above example, given observed values u^*, x^* the message $\mu_{g\to Y}(y; u, x)$ consists of computing a derivative with respect to u, x, *followed by evaluation* of the derivative at u^*, x^*. Thus by "clamping" to observed values, messages in the above scheme will not increase in size, regardless of the functional forms chosen for the CDN functions. By evaluating each derivative in the example for u^*, x^*, y^*, z^*, we can obtain a numerical value for the probability $P(u^*, x^*, y^*, z^*)$ by multiplying messages at the root node.

3.2 Inference in cumulative distribution networks

Thus far we have presented a message-passing scheme for computing derivatives of the joint CDF in order to obtain the joint PDF/PMF $P(\mathbf{x})$. Here we will demonstrate the correspondence between computing higher order derivatives and the problem of inference in a CDN. The relation between differentiation and inference in CDNs is analogous to the relation between marginalization and inference in factor graphs. Thus, just as the sum-product algorithm allows one to compute distributions of the type $P(\mathbf{x}_A|\mathbf{x}_B)$, message-passing in a CDN allows us to compute conditional distributions of the form $F(\mathbf{x}_A|\mathbf{x}_B)$ and $P(\mathbf{x}_A|\mathbf{x}_B)$ for disjoint sets $A, B \subset V$.

In order to compute conditional distributions of the above types, we will assume that when computing a conditional distribution such as $F(\mathbf{x}_A|\mathbf{x}_B)$ or $P(\mathbf{x}_A|\mathbf{x}_B)$, we have $P(\mathbf{x}_B) = \partial_{\mathbf{x}_B}\Big[F(\mathbf{x}_B)\Big] > 0$. Now consider the problem of computing the quantity $F(\mathbf{x}_A|\mathbf{x}_B)$. We can write this as

$$F(\mathbf{x}_A|\mathbf{x}_B) = \frac{\partial_{\mathbf{x}_B}\Big[F(\mathbf{x}_A, \mathbf{x}_B)\Big]}{\partial_{\mathbf{x}_B}\Big[F(\mathbf{x}_B)\Big]} = \frac{\lim\limits_{\mathbf{x}_{V\setminus\{A,B\}}\to\infty} \partial_{\mathbf{x}_B}\Big[F(\mathbf{x})\Big]}{\lim\limits_{\mathbf{x}_{V\setminus B}\to\infty} \partial_{\mathbf{x}_B}\Big[F(\mathbf{x})\Big]} = \frac{\partial_{\mathbf{x}_B}\left[\lim\limits_{\mathbf{x}_{V\setminus\{A,B\}}\to\infty} F(\mathbf{x})\right]}{\partial_{\mathbf{x}_B}\left[\lim\limits_{\mathbf{x}_{V\setminus B}\to\infty} F(\mathbf{x})\right]}$$

$$\propto \partial_{\mathbf{x}_B}\left[\lim_{\mathbf{x}_{V\setminus\{A,B\}}\to\infty} F(\mathbf{x})\right],$$

so that by combining the operations of taking limits and computing derivatives/finite differences, we can compute any conditional probability of the form $F(\mathbf{x}_A|\mathbf{x}_B)$. To compute the conditional CDF for any variable node in the network, we can pass messages from leaf nodes to root and then from the root node back to the leaves. For any given variable node, we can then multiply all incoming messages to obtain the conditional CDF for that variable, up to a scaling factor. We will now demonstrate this principle using the previous toy example CDN.

Example 3.2. Consider the toy example of a CDN over four random variables U, X, Y, Z from Figure 10. Suppose we wish to compute $F(y|x, z) = \lim\limits_{u\to\infty} F(u, y|x, z)$. This is equivalent to message-passing in a CDN defined over variables X, Y, Z with U marginalized out (Figure 11) so that $\tilde{g}(x, y) = \lim\limits_{u\to\infty} g(u, x, y)$. Thus the message updates are given by

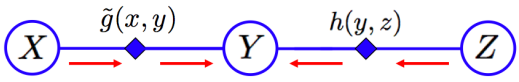


Figure 11: Flow of messages in the toy example CDN of Figure 10 with variable U marginalized out in order to compute the conditional CDF $F(y|x,z)$.

$$\mu_{X\to\tilde{g}}(x) = 1,\ \mu_{\tilde{g}\to Y}(y;x) = \partial_x\Big[\tilde{g}(x,y)\mu_{X\to\tilde{g}}(x)\Big] = \partial_x\Big[\tilde{g}(x,y)\Big]$$
$$\mu_{Z\to h}(z) = 1,\ \mu_{h\to Y}(y;z) = \partial_z\Big[h(y,z)\mu_{Z\to h}(z)\Big] = \partial_z\Big[h(y,z)\Big].$$

Once we have computed the above messages, we can evaluate the conditional CDF $F(y|x,z)$ at node Y as

$$F(y|x,z) = \frac{\mu_{\tilde{g}\to Y}(y;x)\mu_{h\to Y}(y;z)}{\mathcal{Z}} = \frac{\partial_z\Big[h(y,z)\Big]\partial_x\Big[\tilde{g}(x,y)\Big]}{\mathcal{Z}}.$$

Note that the normalizing constant $\mathcal{Z}$ can be readily obtained by computing

$$\mathcal{Z} = \lim_{y\to\infty}\partial_z\Big[h(y,z)\Big]\partial_x\Big[\tilde{g}(x,y)\Big] = \partial_{x,z}\Big[\lim_{y\to\infty} h(y,z)\tilde{g}(x,y)\Big],$$

so that

$$F(y|x,z) = \frac{\mu_{\tilde{g}\to Y}(y;x)\mu_{h\to Y}(y;z)}{\mathcal{Z}} = \frac{\partial_z\Big[h(y,z)\Big]\partial_x\Big[\tilde{g}(x,y)\Big]}{\partial_{x,z}\Big[\lim_{y\to\infty} h(y,z)\tilde{g}(x,y)\Big]} = \frac{\lim_{u\to\infty}\partial_z\Big[h(y,z)\Big]\partial_x\Big[\tilde{g}(u,x,y)\Big]}{\partial_{x,z}\Big[\lim_{u,y\to\infty} h(y,z)\tilde{g}(u,x,y)\Big]}$$
$$= \frac{\partial_{x,z}\Big[\lim_{u\to\infty} F(u,x,y,z)\Big]}{\partial_{x,z}\Big[\lim_{u,y\to\infty} F(u,x,y,z)\Big]}.$$

Note that in the above, if we were to observe $X = x^*, Z = z^*$, we could then evaluate $F(y|x^*,z^*)$ given any candidate value y for variable Y.

The above example shows that the message-passing algorithm can be used to compute conditional CDFs of the form $F(\mathbf{x}_A|\mathbf{x}_B)$, up to a normalizing constant $\mathcal{Z}$. We can readily obtain distributions of the type $P(\mathbf{x}_A|\mathbf{x}_B)$ from $F(\mathbf{x}_A|\mathbf{x}_B)$ by computing $\partial_{\mathbf{x}_A}\Big[F(\mathbf{x}_A|\mathbf{x}_B)\Big]$ using the above message-passing scheme and then multiplying messages together to obtain conditional PDFs. We note that computing the normalizing constant $\mathcal{Z}$ can be viewed as the result of message-passing in a CDN in which the variables $\mathbf{X}_A$ have been marginalized out in addition to variables $\mathbf{X}_{V\setminus\{A,B\}}$ and then evaluating the resulting messages at the observed values $\mathbf{x}_B$. Equivalently, one can compute $\mathcal{Z} = \lim_{\mathbf{x}_A\to\infty}\partial_{\mathbf{x}_B}\Big[F(\mathbf{x}_A,\mathbf{x}_B)\Big]$ after message-passing with only variables in $V\setminus\{A,B\}$ marginalized out.

- For each leaf variable node α' and for all function nodes $s \in \mathcal{N}(\alpha')$, propagate $\mu_{\alpha' \to s}(\mathbf{x}) = 1$. For each leaf function node with function $\phi_s(x_{\alpha'})$, send the messages $\mu_{s \to \alpha'}(\mathbf{x}) = \phi_s(x_{\alpha'})$.
- For each non-leaf variable node α and neighboring function nodes $s \in \mathcal{N}(\alpha)$,

$$\mu_{\alpha \to s}(\mathbf{x}) = \prod_{s' \in \mathcal{N}(\alpha) \setminus s} \mu_{s' \to \alpha}(\mathbf{x}).$$

- For each non-leaf function node s and neighboring variable nodes $\alpha \in \mathcal{N}(s)$,

$$\mu_{s \to \alpha}(\mathbf{x}) = \partial_{\mathbf{x}_{\mathcal{N}(s) \setminus \alpha}} \left[\phi_s(\mathbf{x}_s) \prod_{\beta \in \mathcal{N}(s) \setminus \alpha} \mu_{\beta \to s}(\mathbf{x}) \right].$$

- For root node $\alpha \in V$, repeat the 2^{nd} and 3^{rd} steps above from α to leaf nodes α'.

Table 1: The derivative-sum-product (DSP) algorithm for inference in a CDN defined over discrete variables.

3.3 Derivative-sum-product: A message-passing algorithm for inference in cumulative distribution networks

Because the fundamental operations required for message-passing consist of differentiation/finite differences, sums and products, we will refer to the above class of message-passing algorithms as the derivative-sum-product (DSP) algorithm. For CDNs defined over discrete ordinal variables, the DSP algorithm is shown in Table 1. As can be seen, for graphs defined over discrete variables, the DSP algorithm is analogous to the sum-product algorithm with the summation operation replaced by a finite difference operation. For graphs defined over discrete ordinal variables that take on one of K values, for an observed $\mathbf{x}$, each message $\mu_{\alpha \to s}, \mu_{s \to \alpha}$ consists of a K-vector, analogous to messages in the sum-product algorithm. To see this, we note that each time we compute a finite difference with respect to variables in $\mathcal{N}(s) \setminus \alpha$, we also evaluate the result at $\mathbf{x}_{\mathcal{N}(s) \setminus \alpha}$, ensuring that each message is a K-vector.

While the DSP algorithm for discrete variable networks for computing and evaluating $P(\mathbf{x})$ has the same order of complexity as the sum-product algorithm, the required complexity increases for CDNs defined over continuous variables. For such models, we are required to invoke the product rule of differential calculus in order to express these messages in terms of the derivatives of CDN functions and combinations thereof. To this end, we need to define two additional sets of messages $\lambda_{\alpha \to s}(\mathbf{x})$ and $\lambda_{s \to \alpha}(\mathbf{x})$ which correspond to $\partial_{x_\alpha} \left[\mu_{\alpha \to s}(\mathbf{x}) \right]$ and $\partial_{x_\alpha} \left[\mu_{s \to \alpha}(\mathbf{x}) \right]$ respectively. We first derive the expression for $\lambda_{\alpha \to s}(\mathbf{x})$ by applying the product rule of differential calculus to the message $\mu_{\alpha \to s}(\mathbf{x})$, bearing in mind that each of

the messages $\mu_{s\to\alpha}(\mathbf{x})$ depends on variable X_α. This yields

$$\lambda_{\alpha\to s}(\mathbf{x}) = \partial_{x_\alpha}\Big[\mu_{\alpha\to s}(\mathbf{x})\Big] = \partial_{x_\alpha}\left[\prod_{s'\in\mathcal{N}(\alpha)\setminus s}\mu_{s'\to\alpha}(\mathbf{x})\right] = \mu_{\alpha\to s}(\mathbf{x})\sum_{s'\in\mathcal{N}(\alpha)\setminus s}\frac{\lambda_{s'\to\alpha}(\mathbf{x})}{\mu_{s'\to\alpha}(\mathbf{x})}.$$

In order to derive the general expressions for $\mu_{s\to\alpha}(\mathbf{x})$, $\lambda_{s\to\alpha}(\mathbf{x})$, we first note that for any two differentiable multivariate functions $f(\mathbf{x}), g(\mathbf{x})$, the product rule for computing the higher order derivative of a product of functions is given by

$$\partial_{\mathbf{y}}\Big[f(\mathbf{y})g(\mathbf{y})\Big] = \sum_{\mathbf{y}_A\subseteq\mathbf{y}}\partial_{\mathbf{y}_A}\Big[f(\mathbf{y})\Big]\partial_{\mathbf{y}\setminus\mathbf{y}_A}\Big[g(\mathbf{y})\Big].$$

The key observation we make here is that to evaluate the above derivative for observed $\mathbf{y}$, we can evaluate each term in the summation for the observed $\mathbf{y}$ such that the above is merely a sum of products of scalars. Thus, given a vector of observed variable values $\mathbf{x}$, the messages in the DSP algorithm for continuous variables will all consist of scalars, allowing us to obtain numerical values for probabilities under the model.

To compute messages $\mu_{s\to\alpha}(\mathbf{x}), \lambda_{s\to\alpha}(\mathbf{x})$ from $\mu_{s\to\alpha}(\mathbf{x})$, applying the above product rule yields

$$\begin{aligned}
\mu_{s\to\alpha}(\mathbf{x}) &= \partial_{\mathbf{x}_{\mathcal{N}(s)\setminus\alpha}}\left[\phi_s(x_\alpha,\mathbf{x}_{\mathcal{N}(s)\setminus\alpha})\prod_{\beta\in\mathcal{N}(s)\setminus\alpha}\mu_{\beta\to s}(\mathbf{x})\right] \\
&= \sum_{B\subseteq\mathcal{N}(s)\setminus\alpha}\partial_{\mathbf{x}_B}\Big[\phi_s(\mathbf{x}_s)\Big]\prod_{\beta\in B}\mu_{\beta\to s}(\mathbf{x})\prod_{\beta\in\mathcal{N}(s)\setminus\{\alpha\cup B\}}\lambda_{\beta\to s}(\mathbf{x}) \qquad (20)\\
\lambda_{s\to\alpha}(\mathbf{x}) &= \partial_{x_\alpha}\Big[\mu_{s\to\alpha}(\mathbf{x})\Big] \\
&= \sum_{B\subseteq\mathcal{N}(s)\setminus\alpha}\partial_{\mathbf{x}_B,x_\alpha}\Big[\phi_s(\mathbf{x}_s)\Big]\prod_{\beta\in B}\mu_{\beta\to s}(\mathbf{x})\prod_{\beta\in\mathcal{N}(s)\setminus\{\alpha\cup B\}}\lambda_{\beta\to s}(\mathbf{x}),
\end{aligned}$$

where we have made use of the tree-structure of the CDN to write the derivative of a product of messages as a product of derivatives of the messages. The above updates then define the DSP algorithm for CDNs defined over continuous variables, with a total of four sets of messages defined solely in terms of the CDN functions, their derivatives and linear combinations thereof. The message-passing algorithm for continuous CDNs is summarized in Table 2 and is illustrated in Figure 12.

We see from Table 2 that the DSP algorithm grows exponentially in complexity as the number of neighboring variable nodes for any given function increases, as the updates at function nodes require one to perform a sum over all subsets of neighboring variables. However, in many cases the computational complexity will be tractable for sparser graphs, as demonstrated by the following example.

Example 3.3 (Derivative-sum-product on a linear first-order chain CDN)**.** Consider the CDN defined over K variables such that the joint CDF over these variables is given by

$$F(\mathbf{x}) = \prod_{k=1}^{K-1}\phi_k(x_k, x_{k+1}), \qquad (21)$$

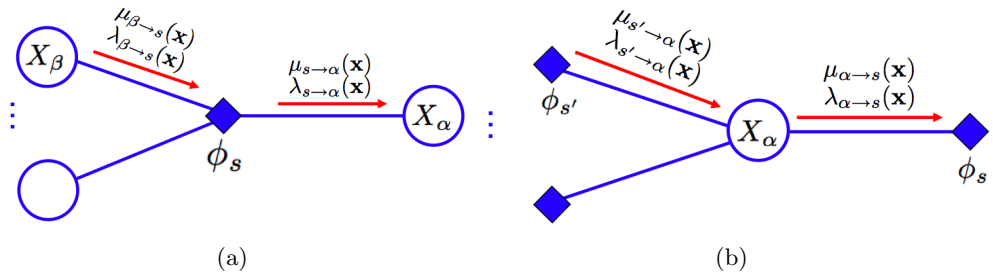


Figure 12: a) Computation of the message from a function node s to a variable node α; b) Computation of the message from a variable node α to a function node s.

- For each leaf variable node α' and for all function nodes $s \in \mathcal{N}(\alpha')$, propagate $\mu_{\alpha'\to s}(\mathbf{x}) = 1, \lambda_{\alpha'\to s}(\mathbf{x}) = 0$. For each leaf function node with function $\phi_s(x_{\alpha'})$, send the messages $\mu_{s\to\alpha'}(\mathbf{x}) = \phi_s(x_{\alpha'}), \lambda_{s\to\alpha'}(\mathbf{x}) = \partial_{x_{\alpha'}}\Big[\phi_s(x_{\alpha'})\Big]$.

- For each non-leaf variable node α and neighboring function nodes $s \in \mathcal{N}(\alpha)$,

$$\mu_{\alpha\to s}(\mathbf{x}) = \prod_{s'\in\mathcal{N}(\alpha)\setminus s} \mu_{s'\to\alpha}(\mathbf{x}),$$
$$\lambda_{\alpha\to s}(\mathbf{x}) = \partial_{x_\alpha}\Big[\mu_{\alpha\to s}(\mathbf{x})\Big] = \mu_{\alpha\to s}(\mathbf{x}) \sum_{s'\in\mathcal{N}(\alpha)\setminus s} \frac{\lambda_{s'\to\alpha}(\mathbf{x})}{\mu_{s'\to\alpha}(\mathbf{x})}.$$

- For each non-leaf function node s and neighboring variable nodes $\alpha \in \mathcal{N}(s)$,

$$\mu_{s\to\alpha}(\mathbf{x}) = \sum_{B\subseteq\mathcal{N}(s)\setminus\alpha} \partial_{\mathbf{x}_B}\Big[\phi_s(\mathbf{x}_s)\Big] \prod_{\beta\in B} \mu_{\beta\to s}(\mathbf{x}) \prod_{\beta\in\mathcal{N}(s)\setminus\{\alpha\cup B\}} \lambda_{\beta\to s}(\mathbf{x}),$$
$$\lambda_{s\to\alpha}(\mathbf{x}) = \partial_{x_\alpha}\Big[\mu_{s\to\alpha}(\mathbf{x})\Big]$$
$$= \sum_{B\subseteq\mathcal{N}(s)\setminus\alpha} \partial_{\mathbf{x}_B, x_\alpha}\Big[\phi_s(\mathbf{x}_s)\Big] \prod_{\beta\in B} \mu_{\beta\to s}(\mathbf{x}) \prod_{\beta\in\mathcal{N}(s)\setminus\{\alpha\cup B\}} \lambda_{\beta\to s}(\mathbf{x}).$$

- For root node $\alpha \in V$, repeat the 2^{nd} and 3^{rd} steps above from α to leaf nodes α'.

Table 2: The derivative-sum-product (DSP) algorithm for inference in a CDN defined over continuous variables.

so that the variable nodes are connected in the chain-structured graph shown in Figure 13. In this case, the DSP messages can be written as

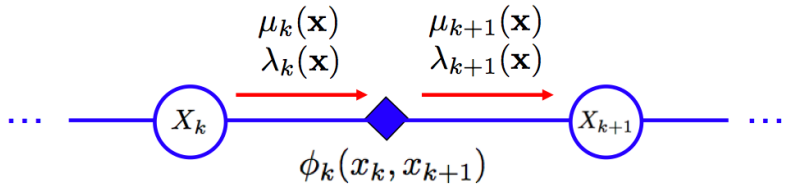


Figure 13: The DSP algorithm for a chain-structured CDN.

$$\begin{aligned}
\mu_{k+1}(\mathbf{x}) &\equiv \mu_{\phi_k \to X_{k+1}}(\mathbf{x}) \\
&= \partial_{x_k}\Big[\phi_k(x_k, x_{k+1})\Big]\mu_k(\mathbf{x}) + \phi_k(x_k, x_{k+1})\lambda_k(\mathbf{x}), \quad k = 1, \cdots, K-1 \\
\lambda_{k+1}(\mathbf{x}) &\equiv \lambda_{\phi_k \to X_{k+1}}(\mathbf{x}) \\
&= \partial_{x_k, x_{k+1}}\Big[\phi_k(x_k, x_{k+1})\Big]\mu_k(\mathbf{x}) + \partial_{x_{k+1}}\Big[\phi_k(x_k, x_{k+1})\Big]\lambda_k(\mathbf{x}), \quad k = 1, \cdots, K-1.
\end{aligned}$$

Example 3.4 (Sampling from a cumulative distribution network)**.** We can further take advantage of the derivative-sum-product algorithm for generating samples from the CDF modelled by a CDN. We can proceed as follows: arbitrarily select a variable in the model, say X_1. Then, generate a sample x_1^* from its marginal CDF $F(x_1)$ (which we obtain by marginalizing out all other variables). Given x_1^*, we can then proceed to generate samples for its children by marginalizing out all other unobserved variables and then sampling from the conditional distribution $F(x_2|x_1^*)$. We can continue this way until we have sampled a complete configuration $\mathbf{x}^* = [x_1^*, \cdots, x_K^*]$. The algorithm for sampling from the joint CDF modelled by a CDN is then given by

- Pick a sampling ordering $X_1, X_2, \cdots, X_K$,
- For variable $X_k, k = 1, \cdots, K$, compute

$$F(x_1, \cdots, x_k) = \lim_{x_{k+1}, \cdots, x_K \to \infty} F(x_1, \cdots, x_k, x_{k+1} \cdots, x_K).$$

- Sample x_i^* from

$$F(x_k|x_1, \cdots, x_{k-1}) = \frac{\partial_{x_1, \cdots, x_{k-1}}\Big[F(x_1, \cdots, x_k)\Big]}{\lim_{x_k \to \infty} \partial_{x_1, \cdots, x_{k-1}}\Big[F(x_1, \cdots, x_k)\Big]}.$$

From the above we see that if the CDN has a tree structure, then we can compute the conditional CDFs $F(x_k|x_1, \cdots, x_{k-1})$ exactly via DSP. In the case of a CDN with cycles, we can always convert it to one with a tree structure by clustering variables and corresponding function nodes, as can be done in the case of factor graphs (Kschischang, Frey and Loeliger, 2001). This generally incurs an increase in function node complexity, but with the benefit of being able to sample from the joint CDF defined by the CDN and not from an approximation thereof.

3.4 Discussion

We have presented the derivative-sum-product algorithm for computing derivatives in a CDN. We have shown that the algorithm is an analog of the sum-product algorithm in factor graphs, so that for tree-structured graphs the algorithm yields exact derivatives of the joint CDF. For graphs defined over continuous variables, the DSP algorithm can be implemented through two sets of messages in order to compute the higher order derivatives of the joint CDF. While we have presented the DSP algorithm for computing derivatives given a set of CDN functions, we have not addressed here the issue of how to learn these CDN functions from data. A possible method would be to run DSP to obtain the joint PDF and then maximize this with respect to model parameters for a particular $\mathbf{x}$. Another issue we have not addressed is how to perform inference in graphs with cycles: an interesting future direction would be to investigate exact or approximate methods for doing so and connections to methods in the literature (Minka, 2001; Neal, 1993) for doing this in traditional graphical models. We will further discuss these issues in the concluding section.

Having defined the CDN and having described the DSP algorithm, we will now proceed to apply both of these to the general problem of learning to rank from examples. As we will see, the ability to model a joint CDF using a graphical framework will yield advantages in both representation and computation for this class of problems.

4. Learning to rank in multiplayer team-based games with cumulative distribution networks

In this section, we will apply CDNs and the DSP algorithm to the problem of structured ranking learning in which the goal is to learn a model for ranking players in a multiplayer game. For this problem, we observe the scores achieved by several players over many games $t = 1 \cdots, T$ in which players interactively compete in groups, or teams, which change with each game. For any given game, players compete in teams so that at the end of each game, each player will have achieved a score as a result of actions taken by all players during the game. For example, these player scores could correspond to the number of targets destroyed or the number of flags stolen, so that a higher player score reflects a better performance for that player. Here we will define a game Γ_t as a triplet $(\mathcal{P}_t, \mathcal{T}_t, \mathcal{O}_t)$, where $\mathcal{P}_t \subset \mathcal{P}$ is a subset of the set $\mathcal{P}$ of all players and $\mathcal{T}_t$ is a partition of $\mathcal{P}_t$ into sets corresponding to teams for game Γ_t, so that if $\mathcal{T}_t = \{\mathcal{T}_t^1, \cdots, \mathcal{T}_t^N\}$ then there are N teams for game Γ_t and player $k \in \mathcal{P}_t$ is assigned to team n for game Γ_t if and only if $k \in \mathcal{T}_t^n$. For example, a game involving six players labelled $1, 2, 3, 4, 5, 6$ organized into three teams of two players each could correspond to $\mathcal{P}_t = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{T}_t = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$. Without loss of generality we will label the teams in a game by $n = 1, \cdots, N$ where each team corresponds to a set in the partition $\mathcal{T}_t$.

In addition to the above, we will denote by $\mathcal{O}_t$ the *outcome* of a game that consists of the pair $(\mathbf{x}_{\mathcal{P}_t}, \mathbf{r}_{\mathcal{T}_t})$, where $\mathbf{x}_{\mathcal{P}_t} \in \mathbb{R}^{|\mathcal{P}_t|}$ is a vector of player scores for game Γ_t and the set $\mathbf{r}_{\mathcal{T}_t}$ is defined as a partially ordered set of team performances, or set of ranks for each team. Such ranks are obtained by first computing the sum of the player scores for each team $n = 1, \cdots, N$, and then ranking the teams by sorting the resulting sums. We will refer to these sums in the sequel as the team scores t_n. An example of this for the previous example of a game with six players assigned to three teams is $\mathbf{x}_{\mathcal{P}_t} = [30\ 12\ 15\ 25\ 100\ 23]^T$,

so that $\mathbf{r}_{\mathcal{T}_t} = \{2,\ 1,\ 3\}$ is the corresponding partially ordered set of team rankings. We will also denote by $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{T}_t^n|}$ the vector of player scores for team n in game Γ_t. Games can also be classified into various types, such that the sizes and/or number of teams are constrained in different ways for different game types. For example, a "SmallTeam" game type would consist of two teams with at most two players per team, whereas a "FreeForAll" game type would constrain the number of teams to be at most eight, with one player per team. Furthermore, the team rankings are a function of unweighted sums of player scores: although there is no reason *a priori* to weigh the scores of players differently for determining the rank of a team, one could extend the above scheme for determining team rankings to weigh player scores according to player type or player-specific features.

Given the above, the goal is to construct a model that will allow us to predict the outcome $\mathcal{O}_t$ of the new game before it begins, given $\mathcal{P}_t$ and previous game outcomes $\mathcal{O}_1, \cdots, \mathcal{O}_{t-1}$. In particular, we wish to construct a model that will minimize the number of mis-ordered teams based on the set of team performances $\mathbf{r}_{\mathcal{T}_t}$ for game Γ_t. Here, the probability model for the given game should account for the team-based structure of games, such that team performances are determined by individual player scores and a game outcome is determined by the ordering of team scores. We will demonstrate here that the graphical framework of CDNs makes it straightforward to model both the notion of ordering of variables in the model as well as statistical independence relationships among these variables. In particular, the model we will construct here will be amenable to exact inference via the DSP algorithm.

Our model will be similar in design to the TrueSkill$^{\text{TM}}$ model of (Herbrich, Minka and Graepel, 2007) for skill rating in Halo 2$^{\text{TM}}$, whereby each player $k \in \mathcal{P}_t$ is assigned a probability distribution over latent skill variables S_k, which is then inferred from individual player scores over multiple games using the expectation propagation algorithm for approximate inference (Minka, 2001). Inference in the TrueSkill$^{\text{TM}}$ model thus consists of applying expectation propagation to a factor graph for a given game in order to update probabilities over player skills. An example of such a factor graph is shown in Figure 14. In TrueSkill$^{\text{TM}}$, the factors connecting team-specific nodes to one another dictate a constraint on relative differences in the total player scores between teams, while factors connecting player nodes to their team-specific nodes enforce the constraint that the team score is determined by the sum of player scores. Finally, for teams $n, n+1$, there is a difference variable $H_{n,n+1}$ and a corresponding factor which declares a tied rank between two teams if the difference between the two team scores is below some threshold parameter. Having described the TrueSkill model, we will now proceed to describe an alternate model formulated using the framework of CDNs.

4.1 A cumulative distribution network for modelling multiplayer game outcomes

Here we will examine a model for multiplayer game outcomes that will be modeled using a CDN. The model will be designed on a game-by-game basis in which the team assignments of players for a given game determines the connectivity of the graph $\mathcal{G}$ for the CDN. In our model the team variables will correspond to the ranks of teams: we will call such variables *team performances* and denote these as R_n for team n in order to contrast these with the

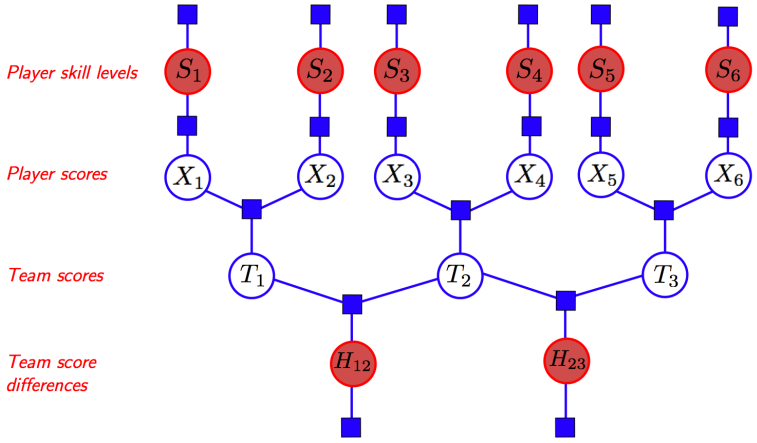


Figure 14: The TrueSkill$^{\text{TM}}$ factor graph for a particular Halo 2$^{\text{TM}}$ game involving three teams with two players each with the team scores $T_1 = t_1, T_2 = t_2, T_3 = t_3$ with $t_1 < t_2 < t_3$ so that team 3 here achieved the highest total of player scores. The variables H_{12}, H_{23} correspond to differences in team scores which determine the ranking of teams, so that teams n and $n+1$ are tied in their rankings if the difference in their team scores is below a threshold parameter. Here, $\mathcal{P}_t = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{T}_t = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$. Latent variables correspond to nodes in red and observed variables correspond to nodes in blue. Each player $k = 1, 2, 3, 4, 5, 6$ is assigned a skill function that reflects the distribution of that player's skill level S_k given past game outcomes. Each player then achieves score X_k in any given game and team scores $T_n, n = 1, 2, 3$ are then determined as the sum of player scores for each team.

team score variables T_n in the TrueSkill model. Our model will account for player scores X_k for each player $k \in \mathcal{P}_t$ in the game, the team performances R_n for each team $n = 1, \cdots, N$ in the game and each player's skill function $s_k(x_k)$, which is a CDF specific to each player. For any given game, R_n will be determined as the sum of the player scores for team n, and then sorting the resulting sums so that R_n corresponds to the rank of team n. The set of observed team performances $\mathbf{r}_{\mathcal{T}_t}$ will be given by the joint configuration of the R_n variables for that game. The goal will then be to adapt player skill functions $s_k(x_k)$ given previous game outcomes. We will design our model according to two principles. First, the relationship between player scores and team performances is modeled as being stochastic, as both player scores and team assignments vary from one game to the next, so that given knowledge of the players in that game and their team assignments, there is some uncertainty in how a team will rank once the game is over. Second, team performance variables depend on those of other teams in the game, so that each team's performance should be linked to that of other teams in a game.

The CDN framework allows us to satisfy both desiderata in the form of modelling constraints on the marginal CDFs for variables in the model. To address the first point, we will require a set of CDN functions that connect player scores to team performances. Here we will make use of the cumulative model for ordinal regression (see Appendix) that relates a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ on inputs $\mathbf{x}$ to a single output ordinal variable $y \in \{r_1, \cdots, r_L\}$ so that $\mathbb{P}[y = r_l] = \mathbb{P}[\theta(r_{l-1}) < f(\mathbf{x}) + \epsilon \leq \theta(r_l)] = F_\epsilon(\theta(r_l) - f(\mathbf{x})) - F_\epsilon(\theta(r_{l-1}) - f(\mathbf{x}))$, where ϵ is an additive noise variable and $\theta(r_0), \cdots, \theta(r_L)$ are the cutpoint parameters of the model

with $\theta(r_0) = -\infty, \theta(r_L) = \infty$. Equivalently, we can write $\mathbb{P}[y \leq r_l] = \mathbb{P}[\epsilon \leq \theta(r_l) - f(\mathbf{x})]$. In the context of multiplayer games, we perform separate ordinal regressions for different game types, as the cutpoints that are learned for a given game type may vary between different game types due to differing team sizes between game types. For a given game type, we treat the set of all games as a bag of pairs of player score vectors $\mathbf{x}_n$ and team performances r_n from which cutpoints in an ordinal regression model can be learned. Thus, we learn a set of cutpoints $\theta(r_0) < \cdots < \theta(r_L)$ once using all of the games in the training data set for a given game type. Team performances are treated as being independent: thus, we can use the CDN framework to augment the above parametric model in order to account for statistical dependencies between multiple team performances in any given game.

We will model multiplayer games using a CDN in which players are grouped into teams and teams compete with one another. If there are N teams for any given game, then we can assign a CDN function g_n for each team such that

$$g_n(\mathbf{x}_n, r_n) = \int_{-\infty}^{\mathbf{x}_n} F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \sigma^2\big) P\big(\mathbf{u}\big)\, d\mathbf{u}$$

where $F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \sigma^2\big)$ is a cumulative model relating input player scores to output team performance and $\mathbf{x}_n, r_n$ are the player scores and team performance for team n. The regression function in the cumulative model is given by $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with $\mathbf{w}$ set to the vector of ones $\mathbf{1}$, as we weigh the contributions of players on a team equally. Furthermore, $\theta(r_n)$ are the cutpoints that define contiguous intervals in which r_n is the ranking for team n based on that team's performance and $P(\mathbf{u})$ is a probability density over the vector of player scores $\mathbf{u}$. Once the cutpoints have been estimated by ordinal regression, we will model the distributions $F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \sigma^2\big), P\big(\mathbf{u}\big)$ in Equation (22) as

$$\begin{aligned} F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \sigma^2\big) &= \Phi\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \sigma^2\big), \\ P(\mathbf{u}) &= Gaussian(\mathbf{u}; \mu\mathbf{1}, \sigma^2\mathbf{I}). \end{aligned} \tag{22}$$

To address the fact that teams compete in any given game, we model ordinal relationships between team performance using the notion of stochastic orderings (Section 2.3), so that for two teams with team performances R_X, R_Y, $R_X \preceq R_Y$ if $F_{R_X}(t) \geq F_{R_Y}(t) \,\forall\, t \in \mathbb{R}$, where $F_{R_X}(\cdot), F_{R_Y}(\cdot)$ are the marginal CDFs of R_X, R_Y. This then allows us to design models in which we can express differences in team performances in the form of pairwise constraints on their marginal CDFs. We note at this juncture that while it is possible to model such stochastic ordering constraints between variables using directed, undirected or factor graphs, doing so introduces additional constraints that are likely to increase the difficulty of performing inference under such models. In contrast, the CDN framework here allows us to explicitly specify such stochastic ordering constraints, in addition to allowing for tractable computations in the resulting model. Thus, although each of the R_n variables are a deterministic function of the sum of player scores, we can nevertheless model them as being stochastic using the framework of CDNs to specify orderings amongst the R_n variables . By contrast, it will generally be more difficult in terms of computation and representation to enforce constraints of the type $\big[R_n \preceq R_{n+1}\big]$ in a directed/undirected/factor graph model.

For the proposed CDN model, given N ranked teams, we can thus define $N-1$ functions $h_{n,n+1}$ so that

$$h_{n,n+1}(r_n, r_{n+1}) = \Phi\left(\begin{bmatrix} r_n \\ r_{n+1} \end{bmatrix}; \begin{bmatrix} \tilde{r}_n \\ \tilde{r}_{n+1} \end{bmatrix}, \mathbf{\Sigma} \right)$$

where

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{bmatrix}$$

and $\tilde{r}_n \leq \tilde{r}_{n+1}$ are chosen without loss of generality such that $\tilde{r}_n = n$ so as to enforce $R_n \preceq R_{n+1}$ in the overall model. Finally, we will use a *skill function* $s_k(x_k)$ for each player k to model that player's distribution over game scores given previous game outcomes. The player performance nodes in the CDN will then be connected to the team performance nodes via the above CDN functions g_n and team performance variable nodes R_n are linked to one another via the above CDN functions $h_{n,n+1}$. The joint CDF for a given game Γ_t with N teams is then given by

$$F(\mathbf{x}_{\mathcal{P}_t}, \mathbf{r}_{\mathcal{T}_t}) = \prod_{n=1}^{N} g(\mathbf{x}_n, r_n) \prod_{n=1}^{N-1} h_{n,n+1}(r_n, r_{n+1}) \prod_{k \in \mathcal{P}_t} s_k(x_k). \tag{23}$$

The above functions and model variables jointly define the CDN for modelling multiplayer games. An example is given in Figure 15 for a game with three teams and six players. One can readily verify from the CDN of Figure 15 using Proposition 2.11 that for the above model and for any given game, the stochastic ordering relationship $R_1 \preceq R_2 \preceq \cdots \preceq R_N$ as defined above can be enforced by marginalizing over all player scores in the CDN and having selected appropriate cutpoints that satisfy $\theta(r_1) < \theta(r_2) < \theta(r_3)$ and parameters $\tilde{r}_1 < \tilde{r}_2 < \tilde{r}_3$, so that we have $F(r_1) \geq F(r_2) \geq F(r_3)$.

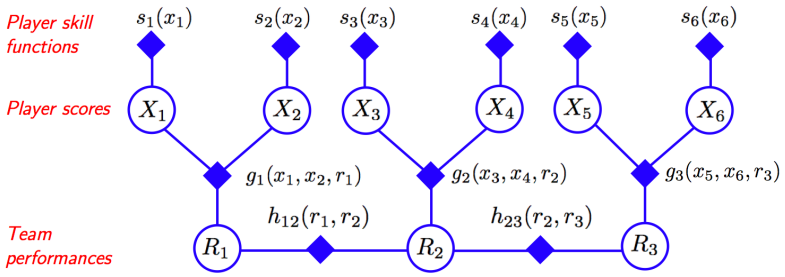


Figure 15: CDN for the player and team performances in a game of Halo 2™ for three teams with two players each. Each player $k = 1, 2, 3, 4, 5, 6$ achieves score X_k in a match and team performances $R_n, n = 1, 2, 3$ are determined based on the sum of player performances for each team.

Having presented the CDN for modelling multiplayer games, we will now proceed to describe a method for predicting game outcomes in which we update player skill functions after each game using message-passing.

4.2 Ranking players in multiplayer games using the derivative-sum-product algorithm

Here we will apply the DSP algorithm in the context of ranking players in multiplayer games with a team structure, where the problem consists of jointly predicting multiple ordinal

output variables. It should be noted that while it may be possible to construct similar models using a directed, undirected or factor graph, the CDN allows us to simultaneously specify both ordinal and statistical independence relationships among model variables while allowing for a tractable inference algorithm.

In order to compute the DSP messages using the above CDN functions, we must compute the derivatives of all CDN functions. Since all of our functions are themselves Gaussian CDFs, the derivatives $\partial_{\mathbf{x}_A}\Big[\phi_s(\mathbf{x}_s)\Big]$ can be easily evaluated with respect to variables $\mathbf{X}_A$ as

$$\partial_{\mathbf{x}_A}\Big[\Phi\big(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}\big)\Big] = Gaussian\Big(\mathbf{x}_A;\boldsymbol{\mu}_A,\boldsymbol{\Sigma}_A\Big)\Phi\Big(\mathbf{x}_B;\tilde{\boldsymbol{\mu}}_B,\tilde{\boldsymbol{\Sigma}}_B\Big)$$

where

$$\mathbf{x} = \left[\begin{array}{c}\mathbf{x}_A\\ \mathbf{x}_B\end{array}\right], \quad \boldsymbol{\mu} = \left[\begin{array}{c}\boldsymbol{\mu}_A\\ \boldsymbol{\mu}_B\end{array}\right], \quad \boldsymbol{\Sigma} = \left[\begin{array}{cc}\boldsymbol{\Sigma}_A & \boldsymbol{\Sigma}_{A,B}\\ \boldsymbol{\Sigma}_{A,B}^T & \boldsymbol{\Sigma}_B\end{array}\right],$$

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_B &= \boldsymbol{\mu}_B + \boldsymbol{\Sigma}_{A,B}^T\boldsymbol{\Sigma}_A^{-1}(\mathbf{x}_A - \boldsymbol{\mu}_A),\\ \tilde{\boldsymbol{\Sigma}}_B &= \boldsymbol{\Sigma}_B - \boldsymbol{\Sigma}_{A,B}^T\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\Sigma}_{A,B}.\end{aligned}$$

The message computations in the CDN are given in the Appendix. We ensure that each message is properly normalized by locally computing the constant $\mathcal{Z} = \lim_{z\to\infty}\mu(z)$ for each message and multiplying each message pair μ, λ by $\mathcal{Z}^{-1}$.

Given the above CDN model for multiplayer games, we would like to then estimate the player skill functions $s_k(x_k)$ for each player k from previous games played by that player. Denote by the set $T_k \subseteq \{1,\cdots,T\}$ the set of games in which player k participated. We then seek to estimate $s_k(x_k)$ for player k given previous team performances $\mathbf{r}_{\mathcal{T}_t}, t \in T_k$ and player scores for *all other* players $\mathbf{x}_{\mathcal{P}_t\setminus k}$ for all games $t \in T_k$ in which player k participated. Denote by $\mathcal{O}_t^{-k}$ the outcome of a game with the player score for player k removed from $\mathbf{x}_{\mathcal{P}_t}$. We will define the skill function $s_k(x_k)$ for a player to be given by

$$s_k(x_k) = F\Big(x_k|\{\mathcal{O}_t^{-k}\}_{t\in T_k}\Big) = \prod_{t\in T_k} F(x_k|\mathcal{O}_t^{-k}).$$

The above expression for the skill function $s_k(x_k)$ for player k corresponds to the conditional distribution $F\Big(x_k|\{\mathcal{O}_t^{-k}\}_{t\in T_k}\Big)$ given all past games played by player k with the assumption that team performances and player scores are independently drawn from CDFs $F(\mathbf{r}_{\mathcal{T}_t}, \mathbf{x}_{\mathcal{P}_t})$ for $t = 1,\cdots,T$. The skill function s_k can then be readily estimated by the DSP algorithm, since each game outcome is modeled by a tree-structured CDN. More precisely, we first initialize $s_k(x_k) = \Phi(x_k;\mu,\beta^2)$. For each game Γ_t we can perform message-passing to obtain the conditional CDF $F(x_k|\mathcal{O}_t^{-k}) = \mu_{g_n\to X_k}(\mathbf{r}_{\mathcal{T}_t}, \mathbf{x}_{\mathcal{P}_t\setminus k})$ for player k (assuming the message $\mu_{g_n\to X_k}$ has been properly normalized as described above) and then perform a multiplicative update $s_k(x_k) \leftarrow s_k(x_k)\mu_{g_n\to X_k}$. The skill function $s_k(x_k)$ can then be used to make predictions for player k's scores in future games. We will proceed in the next section to apply the model and the above inference procedure to the problem of modelling Halo 2$^{\text{TM}}$ games.

4.3 The Halo 2™ Beta dataset

The Halo 2™ Beta dataset (v1.1)[1] consists of player scores for four game types ("HeadToHead", "FreeForAll", "SmallTeams" and "LargeTeams") over a total of 6,465 players. The descriptions for each of the four game modes are given below.

- HeadToHead: 6227 games/1672 players, one player competing against another player
- FreeForAll: 60022 games/5943 players, up to eight players playing against each other
- SmallTeams: 27539 games/4992 players, up to four players per team, two competing teams
- LargeTeams: 1199 games/2576 players, up to eight players per team, two competing teams

To construct the above CDN model, we set the cutpoints $\theta(r_n)$ in the above cumulative model using ordinal regression of team ranks on team performances for all games in the training set. We initialized all player skill functions to $s_k(x_k) = \Phi(x_k; \mu, \beta^2)$. The set of parameters $\{\mu, \rho, \beta, \sigma\}$ in the CDN model was set to $\{25, -0.95, 20, 0.25\}$ for "HeadToHead", $\{50, -0.2, 10, 0.2\}$ for "FreeForAll", $\{20, -0.1, 10, 0.027\}$ for "SmallTeams" and $\{1, -0.9, 1, 0.01\}$ for "LargeTeams" game modes. For each of these game modes, we applied the DSP algorithm as described above in order to obtain updates for the player skill functions $s_k(x_k)$. An example of such an update at the end of a game with four competing players is shown in Figure 16.

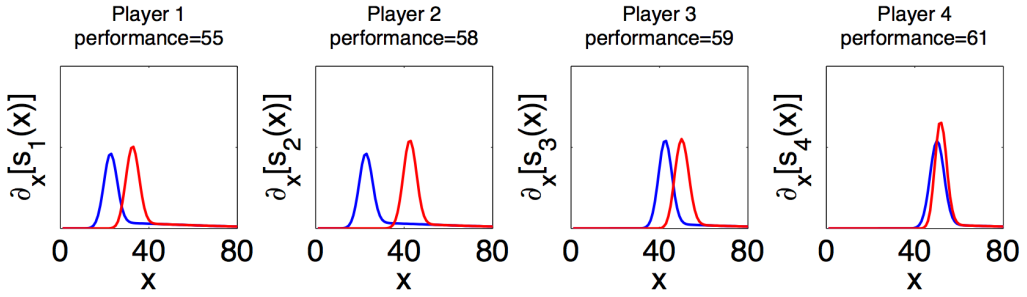


Figure 16: An example of derivative-sum-product updates for a four-player free-for-all game, with the derivative of the skill functions before the updates (blue) and afterwards (red).

Before each game, we can predict the team performances using the player skills learned thus far via the rule $x_k^* = \arg\max_{x_k} \partial_{x_k} \left[s_k(x_k) \right]$. For each game, the set of team performances is then defined by the ordering of teams once the game is over, where we add the predicted player scores x_k^* together for each team and sorting the resulting sums in ascending order. For any predicted set of team performances, an error is incurred for that game if two teams for that game were misranked such that the number of errors for a given game is $\sum_{n=1}^{N-1} \sum_{m>n}^{N} [R_n \leq R_m] \wedge [R_n^{true} > R_m^{true}]$. One can then compute an error rate over the entire set of games for which we make predictions about team performances.

1. Credits for the use of the Halo 2™ Beta Dataset are given to Microsoft Research Ltd. and Bungie.

A plot showing the average prediction error rate obtained for the above CDN models over five runs of DSP is shown in Figure 17. It is worth noting that our choice of multivariate Gaussian CDFs as CDN functions in the above model requires that we use a sampling method in order to evaluate the CDN functions, so that the error bars over the five runs are shown. In addition, Figure 17 also shows the error rates reported by (Herbrich, Minka and Graepel, 2007) for TrueSkill$^{\text{TM}}$ and ELO (Elo, 1978), which is a statistical rating system used in chess. Here, we see that the ability to specify both ordinal relationships and statistical dependence relationships between model variables using a CDN allows us to achieve higher predictive accuracy than either TrueSkill$^{\text{TM}}$or the ELO method.

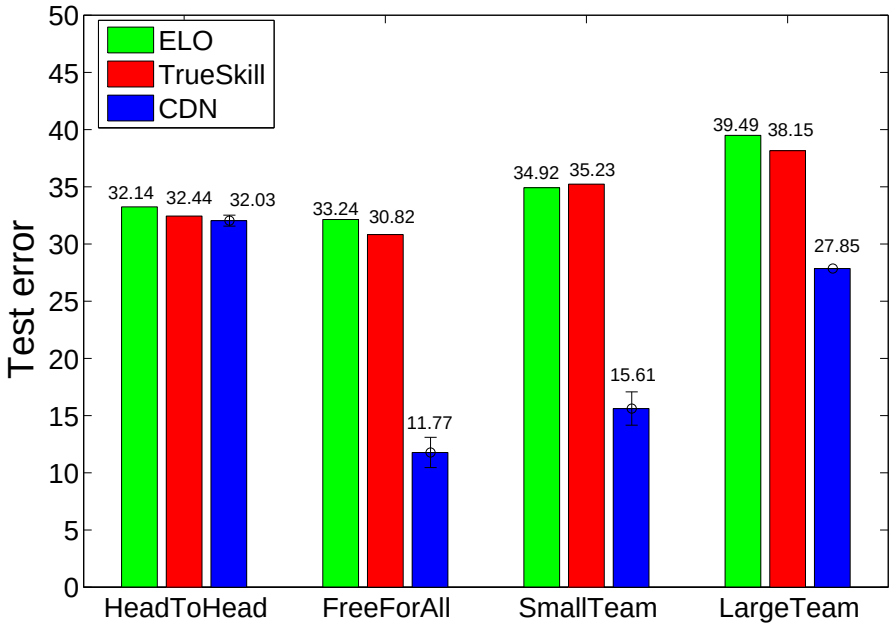


Figure 17: Prediction error on the Halo 2$^{\text{TM}}$ Beta dataset (computed as the fraction of team predicted incorrectly before each game) for DSP, ELO (Elo, 1978) and TrueSkill$^{\text{TM}}$ (Herbrich, Minka and Graepel, 2007) methods. Error bars over five runs of DSP are shown.

4.4 Discussion

In this section we presented a model and method for learning to rank in the context of multiplayer team-based games such as Halo 2$^{\text{TM}}$. Our model represent both statistical dependence relationships and notions of orderings of variables in the model such as team performances and individual player scores. We then used the DSP algorithm to compute conditional CDFs for each player's score. Comparisons to the TrueSkill$^{\text{TM}}$ and ELO methods for factor graph models show that our model and method allows both for fast estimation and improved test error on the Halo 2$^{\text{TM}}$ Beta dataset.

While the above method has the advantage of providing a flexible probabilistic model and allowing for tractable inference, the choice of multivariate Gaussian CDFs for CDN

functions requires the use of sampling methods in order to evaluate DSP messages. Future work could focus on finding faster parameterizations of the CDN functions that do not require sampling.

5. Conclusion

We have proposed the CDN as a graphical model for joint CDFs over many variables. We have shown that the conditional independence properties of a CDN are distinct from the independence properties of directed, undirected and factor graphs. However, these properties include, but are not limited to, those for bidirected graphs. We have then demonstrated that inference in a CDN corresponds to computing derivatives/finite differences. We described the DSP algorithm for computing such derivatives/finite differences by passing messages in the CDN where each message corresponds to local derivatives of the joint CDF.

We used the graphical framework provided by CDNs to formulate models and methods for learning to rank in a structured setting in which we must account for statistical dependence relationships between model variables. We first applied the DSP algorithm to the problem of ranking in multiplayer gaming where players compete in teams. The DSP algorithm allowed us to compute distributions over player scores given previous game outcomes while accounting for the team-based structure of the games, whereby we were able to show improved results over previous methods. The CDN framework was then used to construct loss functionals for structured ranking learning where we wish to account for statistical dependence relationships which arise in ranking a set of objects. We showed that many probability models for rank data can be viewed as particular CDNs with different connectivities between pairwise object preferences. Based on the work and results presented, we can recommend future directions of research pertaining to the methods presented in this manuscript.

5.1 Future work

While we presented a framework for constructing a graphical model for a joint CDF, there may be applications in which one may wish to instead optimize the log-probability density $\log P(\mathbf{x}|\boldsymbol{\theta})$. We presented the DSP algorithm for both discrete and continuous-variable networks and we showed how DSP could be used to compute the probability density $P(\mathbf{x}|\boldsymbol{\theta})$ from the joint CDF $F(\mathbf{x}|\boldsymbol{\theta})$ modelled by the CDN. In order to perform maximum likelihood learning in which we wish to maximize the log-likelihood $\mathcal{L}(\mathbf{x};\boldsymbol{\theta}) = \log P(\mathbf{x}|\boldsymbol{\theta})$ with respect to a parameter vector $\boldsymbol{\theta}$ for a given set of observed variables $\mathbf{x}$, one can use modified versions of DSP messages in order to compute the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x};\boldsymbol{\theta})$ of the log-likelihood. The guiding principle here is that the gradient operator can be distributed amongst local functions in the CDF, much like the differentiation operation in DSP, so that by modifying DSP messages appropriately we can obtain the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x};\boldsymbol{\theta})$. Once computed, the gradient vector can then be used in a gradient-descent algorithm to optimize the log-likelihood. Future research in this direction could be directed at establishing what class of graphs can yield tractable gradient computations, as well as the complexity/accuracy tradeoffs involved in computing gradients in graphs with cycles.

We have shown that our message-passing algorithm leads to the correct set of derivatives of the joint CDF provided that the underlying graph is a tree. As with the sum-product

algorithm for factor graphs, if the graph contains cycles, then the derivative-sum-product is no longer guaranteed to yield the correct mixed derivatives of the joint CDF, so that messages may begin to 'oscillate' as they propagate around cycles in the graph. One important direction to pursue is to establish conditions under which the presence of cycles will not lead to oscillations in messages: one could resort to a similar methodology as that employed by (Weiss and Freeman, 2001), where a graph with cycles is "unwrapped" and the resulting messages are analyzed.

We showed that for graphs defined over continuous variables, the complexity of computing DSP message updates at a given function node increased exponentially with the number of neighboring variable nodes, as one has to sum over products of messages incoming from all subsets of variables connected to the function node. However, it may be possible to approximate messages using simpler, tractable forms such as conditional univariate Gaussian CDFs. Future work here would be to establish tractable methods for performing such approximations and gauge the performance of such an approximate scheme for inference in CDNs on both synthetic and real-world data.

As we have demonstrated, the graph separation criterion for assessing conditional independence in CDNs includes those of bidirected graphs (Richardson and Spirtes, 2002). As such graphs are a special case of mixed graphs containing undirected, directed and bidirected edges, a future avenue of research would be to investigate whether one can tractably approximate such mixed graphical models using a hybrid graphical formulation combining the CDN model with that of factor graphs for joint probability density/mass functions. The Bayesian learning approach adopted by (Silva and Ghahramani, 2009b) could provide a framework with which to qualitatively and quantitatively compare the use of CDNs for constructing such mixed graphical models.

Acknowledgements

Part of this work was done while JCH was a graduate student at the University of Toronto. We wish to thank Zoubin Ghahramani, Nebojsa Jojic, Frank Kschischang, Christopher Meek, Tom Minka, Radford Neal and Thomas Richardson for valuable comments and feedback that have contributed significantly to the content of the manuscript.

6. Appendix

6.1 Derivation of the derivative-sum-product algorithm

To begin, let $\mathcal{G} = (V, S, E)$ be a tree-structured CDN and suppose we wish to compute the joint probability $P(\mathbf{x})$ and evaluate it at observation $\mathbf{x}$. We note that we can root the graph at some node α and we can write the joint CDF as

$$F(\mathbf{x}) = \prod_{s \in \mathcal{N}(\alpha)} T_s\big(\mathbf{x}_{\tau_s^\alpha}\big),$$

where $\mathbf{x}_{\tau_s^\alpha}$ denotes the vector of configurations for all variables in the subtree τ_s^α rooted at variable node α and containing function node s (Figure 18), and $T_s\big(\mathbf{x}_{\tau_s^\alpha}\big)$ corresponds to the product of all functions located in the subtree τ_s^α.

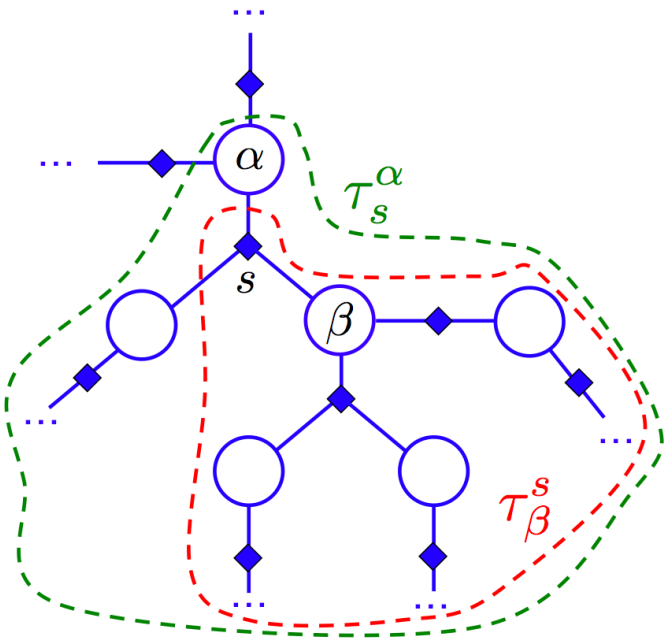


Figure 18: Example of the subtrees $\tau_s^\alpha, \tau_\beta^s$ for a tree-structured CDN given by the graph $\mathcal{G}$.

Now suppose we are interested in computing the probability

$$P(\mathbf{x}) = \partial_{\mathbf{x}}\Big[F(\mathbf{x})\Big] = \partial_{\mathbf{x}}\Bigg[\prod_{s\in\mathcal{N}(\alpha)} T_s\big(\mathbf{x}_{\tau_s^\alpha}\big)\Bigg].$$

Here, we take advantage of the fact that the graph has a tree structure, so that

$$\partial_{\mathbf{x}}\Bigg[\prod_{s\in\mathcal{N}(\alpha)} T_s\big(\mathbf{x}_{\tau_s^\alpha}\big)\Bigg] = \partial_{x_\alpha}\Bigg[\prod_{s\in\mathcal{N}(\alpha)} \partial_{\mathbf{x}_{\tau_s^\alpha\setminus\alpha}}\Big[T_s\big(\mathbf{x}_{\tau_s^\alpha}\big)\Big]\Bigg] = \partial_{x_\alpha}\Bigg[\prod_{s\in\mathcal{N}(\alpha)} \mu_{s\to\alpha}\big(\mathbf{x}_{\tau_s^\alpha}\big)\Bigg].$$

We have introduced the set of functions $\mu_{s\to\alpha}(\mathbf{x}) \equiv \mu_{s\to\alpha}\big(\mathbf{x}_{\tau_s^\alpha}\big)$ defined by

$$\mu_{s\to\alpha}(\mathbf{x}) \equiv \mu_{s\to\alpha}\big(\mathbf{x}_{\tau_s^\alpha}\big) = \partial_{\mathbf{x}_{\tau_s^\alpha\setminus\alpha}}\Big[T_s\big(\mathbf{x}_{\tau_s^\alpha}\big)\Big], \tag{24}$$

where we have assumed that each of the derivatives/finite differences have been evaluated at the desired values $\mathbf{x}_{\tau_s^\alpha\setminus\alpha}$. By its definition, $\mu_{s\to\alpha}(\mathbf{x})$ only depends on variables in the subtree τ_s^α and corresponds to the higher order derivative of the joint CDF with respect to variables in the subtree $\tau_s^\alpha \setminus \alpha$. We can thus view the functions $\mu_{s\to\alpha}(\mathbf{x})$ as messages being passed from each function node $s \in \mathcal{N}(\alpha)$ in the CDN to a neighboring variable node α.

We can now write $T_s\big(\mathbf{x}_{\tau_s^\alpha}\big)$ as a product of functions owing to the tree structure of the graph $\mathcal{G}$, so that

$$T_s\big(\mathbf{x}_{\tau_s^\alpha}\big) = \phi_s(x_\alpha, \mathbf{x}_{\mathcal{N}(s)\setminus\alpha}) \prod_{\beta\in\mathcal{N}(s)\setminus\alpha} T_\beta\Big(\mathbf{x}_{\tau_\beta^s}\Big), \tag{25}$$

where $\mathbf{x}_{\tau_\beta^s}$ denotes the vector of configurations for all variables in the subtree τ_β^s which is rooted at function node s and contains node β (Figure 18), and T_β is the product of all functions in the subtree τ_β^s. Substituting Equation (25) into Equation (24), we obtain

$$\mu_{s\to\alpha}(\mathbf{x}) \equiv \mu_{s\to\alpha}\big(\mathbf{x}_{\tau_s^\alpha}\big) = \partial_{\mathbf{x}_{\tau_s^\alpha}\setminus\alpha}\left[\phi_s(x_\alpha, \mathbf{x}_{\mathcal{N}(s)\setminus\alpha}) \prod_{\beta\in\mathcal{N}(s)\setminus\alpha} T_\beta\Big(\mathbf{x}_{\tau_\beta^s}\Big)\right] \tag{26}$$

$$= \partial_{\mathbf{x}_{\mathcal{N}(s)\setminus\alpha}}\left[\phi_s(x_\alpha, \mathbf{x}_{\mathcal{N}(s)\setminus\alpha}) \prod_{\beta\in\mathcal{N}(s)\setminus\alpha} \partial_{\mathbf{x}_{\tau_\beta^s}\setminus\beta}\Big[T_\beta\Big(\mathbf{x}_{\tau_\beta^s}\Big)\Big]\right] \tag{27}$$

$$= \partial_{\mathbf{x}_{\mathcal{N}(s)\setminus\alpha}}\left[\phi_s(x_\alpha, \mathbf{x}_{\mathcal{N}(s)\setminus\alpha}) \prod_{\beta\in\mathcal{N}(s)\setminus\alpha} \mu_{\beta\to s}\Big(\mathbf{x}_{\tau_\beta^s}\Big)\right]. \tag{28}$$

Here we have defined messages $\mu_{\beta\to s}(\mathbf{x}) \equiv \mu_{\beta\to s}\Big(\mathbf{x}_{\tau_\beta^s}\Big)$ from variable nodes to function nodes. Similar to the definition for $\mu_{s\to\alpha}(\mathbf{x})$, the message $\mu_{\beta\to s}(\mathbf{x})$ only depends on variables in the subtree τ_β^s and corresponds to the higher order derivative of the joint CDF with respect to variables in the subtree $\tau_\beta^s \setminus \beta$.

Finally, to compute the messages $\mu_{\beta\to s}(\mathbf{x})$ from variables to functions, we can write each of the functions $T_\beta\big(\mathbf{x}_{\tau_\beta^s}\big)$ as a product such that

$$T_\beta\big(\mathbf{x}_{\tau_\beta^s}\big) = \prod_{s'\in\mathcal{N}(\beta)\setminus s} T_{s'}\big(\mathbf{x}_{\tau_{s'}^\beta}\big), \tag{29}$$

where $T_{s'}$ is defined identically to T_s above but for function node s'. Substituting this into the expression for $\mu_{\beta\to s}(\mathbf{x})$ in Equation (28) yields

$$\mu_{\beta\to s}(\mathbf{x}) = \partial_{\mathbf{x}_{\tau_\beta^s}\setminus\beta}\Big[T_\beta\big(\mathbf{x}_{\tau_\beta^s}\big)\Big] = \prod_{s'\in\mathcal{N}(\beta)\setminus s} \partial_{\mathbf{x}_{\tau_{s'}^\beta}\setminus\beta}\Big[T_{s'}\big(\mathbf{x}_{\tau_{s'}^\beta}\big)\Big] \tag{30}$$

$$= \prod_{s'\in\mathcal{N}(\beta)\setminus s} \mu_{s'\to\beta}(\mathbf{x}). \tag{31}$$

Thus, to compute messages from variables to functions, we simply take the product of all incoming messages except for the message coming from the destination function node. As in the sum-product algorithm, variables with only two neighboring functions simply pass messages through unchanged. We see here that the process of differentiation in a CDN can be implemented as an algorithm in which we pass messages $\mu_{\alpha\to s}$ from variables to neighboring function nodes and messages $\mu_{s\to\alpha}$ from functions to neighboring variable nodes. Messages can be computed recursively from one another as described above: we start from an arbitrary root variable node α and propagate messages up from leaf nodes to the root node. As in the sum-product algorithm, leaf variable nodes α' send the message $\mu_{\alpha'\to s}(\mathbf{x}) = 1$ while leaf function nodes $\phi_s(x_{\alpha'})$ send the message $\mu_{s\to\alpha'}(\mathbf{x}) = \phi_s(x_{\alpha'})$.

The message-passing algorithm proceeds until messages have been propagated along every edge in the network and the root variable node has received all incoming messages

from the remainder of the network. Once all messages have been sent, we can obtain the probability density of the variables in the graph from differentiating the product of incoming messages at the root node α, so that

$$P(\mathbf{x}) = \partial_{x_\alpha}\left[\prod_{s\in\mathcal{N}(\alpha)} \mu_{s\to\alpha}(\mathbf{x})\right].$$

6.2 Ordinal regression

In many domains, one is faced with the problem of predicting multinomial variables that can each take one of a finite number of values in some discrete set $\mathcal{X} = \{r_1, \cdots, r_K\}$ for some integer K. Such multinomial variables can then be distinguished as being of the type

- *Nominal*, or *categorical*, so that the set $\mathcal{X}$ does not admit an ordering of variable values.

- *Ordinal*, so that the set $\mathcal{X}$ admits a total ordering over variable values of the type $r_1 \prec \cdots \prec r_K$.

An example of a nominal variable is gender, such as $\mathcal{X} = \{Male, Female\}$ and an example of an ordinal variable is a grading scheme $\mathcal{X} = \{A, B, C, D\}$ so that the possible variable values satisfy the total ordering $D \prec C \prec B \prec A$.

In ordinal regression, the goal is to predict a discrete variable $y \in \{r_1, \cdots, r_K\}$ given a set of features $\mathbf{x}$, where $r_1 \prec \cdots \prec r_K$ are an ordered set of labels. Unlike the general problem of multiclass classification in which variables to be predicted are nominal, output labels in the setting of ordinal regression are not permutation-invariant and so any model for the problem should account for the orderings of the output variable values.

One model for performing ordinal regression is the *cumulative model* (McCullagh, 1980), which relates an input vector $\mathbf{x}$ to an ordinal output y via a function f and a set of *cutpoints* $\theta(r_1) < \cdots < \theta(r_K)$ along the real line $\mathbb{R}$ so that $y = r_k$ if $\theta(r_{k-1}) < f(\mathbf{x}) + \epsilon \leq \theta(r_k)$, where ϵ is additive noise and we define $\theta(r_0) = -\infty, \theta(r_K) = \infty$ (Figure 19). If $P(\epsilon)$ is the probability density function from which the noise variable ϵ is drawn, then we can write

$$\begin{aligned}
\mathbb{P}\big[y = r_k\big] &= \mathbb{P}\big[\theta(r_{k-1}) < f(\mathbf{x}) + \epsilon \leq \theta(r_k)\big] \\
&= \mathbb{P}\big[\{\theta(r_{k-1}) - f(\mathbf{x}) < \epsilon\} \bigcap \{\epsilon \leq \theta(r_k) - f(\mathbf{x})\}\big] \\
&= F_\epsilon(\theta(r_{k-1}) - f(\mathbf{x})) - F_\epsilon(\theta(r_k) - f(\mathbf{x})),
\end{aligned}$$

where $F_\epsilon \equiv F(\epsilon)$ is the corresponding cumulative distribution function for $P(\epsilon)$. The above equation defines a likelihood function for a given observed pair $(\mathbf{x}, y)$, so that the cutpoints $\theta(r_k)$ and the regression function $f(\mathbf{x})$ can subsequently be estimated from training data by maximizing the likelihood function with respect to the cutpoints $\theta(r_k)$ and the regression function $f(\mathbf{x})$.

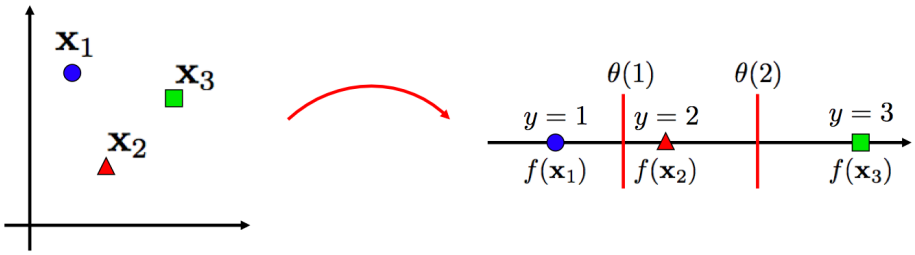


Figure 19: An illustration of the ordinal regression model. A given point has label $y = r_k$ if $\theta(r_{k-1}) < f(\mathbf{x}) + \epsilon \leq \theta(r_k)$, where ϵ is a noise variable.

6.3 Derivative-sum-product message updates for learning to rank in multiplayer games

Here we present the DSP algorithm for updating player ranks. Messages are ensured to be properly normalized locally by computing the constant $\mathcal{Z} = \lim_{z\to\infty} \mu(z)$ for each message and multiplying the message pair μ, λ by $\mathcal{Z}^{-1}$.

- Initialize for each player score node X_k:

$$\mu_{X_k \to g_n}(x_k) = s_k(x_k),$$
$$\lambda_{X_k \to g_n}(x_k) = \partial_{x_k}\Big[s_k(x_k)\Big].$$

- Pass messages from function node g_n to team performance node R_n for neighboring player nodes $\mathbf{X}_n$, $n = 1, \cdots, N$:

$$\mu_{g_n \to R_n}(\mathbf{r}, \mathbf{x}) = \sum_{\substack{s,t|\mathbf{X}_s \cup \mathbf{X}_t = \mathbf{X}_n \\ \mathbf{X}_s \bigcap \mathbf{X}_t = \emptyset}} \partial_{\mathbf{x}_s}\Big[g_n(\mathbf{x}_n, r_n)\Big] \prod_{j|X_j \in \mathbf{X}_s} \mu_{X_j \to g_n}(x_j) \prod_{j|X_j \in \mathbf{X}_t} \lambda_{X_j \to g_n}(x_j),$$
$$\lambda_{g_n \to R_n}(\mathbf{r}, \mathbf{x}) = \sum_{\substack{s,t|\mathbf{X}_s \cup \mathbf{X}_t = \mathbf{X}_n \\ \mathbf{X}_s \bigcap \mathbf{X}_t = \emptyset}} \partial_{\mathbf{x}_s, r_n}\Big[g_n(\mathbf{x}_n, r_n)\Big] \prod_{j|X_j \in \mathbf{X}_s} \mu_{X_j \to g_n}(x_j) \prod_{j|X_j \in \mathbf{X}_t} \lambda_{X_j \to g_n}(x_j).$$

- Set $\mu_{h_{n-1,n} \to R_n}(\mathbf{r}, \mathbf{x}) = \lambda_{h_{n-1,n} \to R_n}(\mathbf{r}, \mathbf{x}) = 1$ for $n = 1$. Pass messages from team performance node R_n to neighboring team performance nodes R_{n+1} and function nodes $h_{n,n+1}$ for $n = 1, \cdots, N$:

$$\begin{aligned}
\mu_{R_n \to h_{n,n+1}}(\mathbf{r}, \mathbf{x}) &= \mu_{h_{n-1,n} \to R_n}(\mathbf{r}, \mathbf{x}) \mu_{g_n \to R_n}(\mathbf{r}, \mathbf{x}), \\
\lambda_{R_n \to h_{n,n+1}}(\mathbf{r}, \mathbf{x}) &= \lambda_{h_{n-1,n} \to R_n}(\mathbf{r}, \mathbf{x}) \mu_{g_n \to R_n}(\mathbf{r}, \mathbf{x}) \\
&\quad + \mu_{h_{n-1,n} \to R_n}(\mathbf{r}, \mathbf{x}) \lambda_{g_n \to R_n}(\mathbf{r}, \mathbf{x}), \\
\mu_{h_{n,n+1} \to R_{n+1}}(\mathbf{r}, \mathbf{x}) &= \mu_{R_n \to h_{n,n+1}}(\mathbf{r}, \mathbf{x}) \partial_{r_n}\Big[h_{n,n+1}(r_n, r_{n+1})\Big] \\
&\quad + \lambda_{R_n \to h_{n,n+1}}(\mathbf{r}, \mathbf{x}) h_{n,n+1}(r_n, r_{n+1}), \\
\lambda_{h_{n,n+1} \to R_{n+1}}(\mathbf{r}, \mathbf{x}) &= \mu_{R_n \to h_{n,n+1}}(\mathbf{r}, \mathbf{x}) \partial_{r_n, r_{n+1}}\Big[h_{n,n+1}(r_n, r_{n+1})\Big] \\
&\quad + \lambda_{R_n \to h_{n,n+1}}(\mathbf{r}, \mathbf{x}) \partial_{r_{n+1}}\Big[h_{n,n+1}(r_n, r_{n+1})\Big].
\end{aligned}$$

- Set $\mu_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x}) = \lambda_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x}) = 1$ for $n = N$. Pass messages from team performance node R_n to neighboring team performance nodes R_{n-1} and function nodes $h_{n-1,n}$ for $n = 1, \cdots, N$:

$$\begin{aligned}
\mu_{R_n\to h_{n-1,n}}(\mathbf{r},\mathbf{x}) &= \mu_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x})\mu_{g_n\to R_n}(\mathbf{r},\mathbf{x}),\\
\lambda_{R_n\to h_{n-1,n}}(\mathbf{r},\mathbf{x}) &= \lambda_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x})\mu_{g_n\to R_n}(\mathbf{r},\mathbf{x})\\
&\quad + \mu_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x})\lambda_{g_n\to R_n}(\mathbf{r},\mathbf{x}),\\
\mu_{h_{n-1,n}\to R_{n-1}}(\mathbf{r},\mathbf{x}) &= \mu_{R_n\to h_{n-1,n}}(\mathbf{r},\mathbf{x})\partial_{r_n}\Big[h_{n-1,n}(r_{n-1},r_n)\Big]\\
&\quad + \lambda_{R_n\to h_{n-1,n}}(\mathbf{r},\mathbf{x})h_{n-1,n}(r_{n-1},r_n),\\
\lambda_{h_{n-1,n}\to R_{n-1}}(\mathbf{r},\mathbf{x}) &= \mu_{R_n\to h_{n-1,n}}(\mathbf{r},\mathbf{x})\partial_{r_{n-1},r_n}\Big[h_{n-1,n}(r_{n-1},r_n)\Big]\\
&\quad + \lambda_{R_n\to h_{n-1,n}}(\mathbf{r},\mathbf{x})\partial_{r_{n-1}}\Big[h_{n-1,n}(r_{n-1},r_n)\Big].
\end{aligned}$$

- Pass messages from each team performance node R_n to neighboring function nodes g_n:

$$\begin{aligned}
\mu_{R_n\to g_n}(\mathbf{r},\mathbf{x}) &= \mu_{h_{n-1,n}\to R_n}(\mathbf{r},\mathbf{x})\mu_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x}),\\
\lambda_{R_n\to g_n}(\mathbf{r},\mathbf{x}) &= \lambda_{h_{n-1,n}\to R_n}(\mathbf{r},\mathbf{x})\mu_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x})\\
&\quad + \mu_{h_{n-1,n}\to R_n}(\mathbf{r},\mathbf{x})\lambda_{h_{n,n+1}\to R_n}(\mathbf{r},\mathbf{x}).
\end{aligned}$$

- Pass messages from function nodes g_n to neighboring player score nodes X_k:

$$\begin{aligned}
\mu_{g_n\to X_k}(\mathbf{r},\mathbf{x}) = &\sum_{\substack{s,t|\mathbf{X}_s\cup\mathbf{X}_t=\mathbf{X}_n\setminus X_k\\ \mathbf{X}_s\bigcap\mathbf{X}_t=\emptyset}}\prod_{j|X_j\in\mathbf{X}_s}\mu_{X_j\to g_n}(x_j)\prod_{j|X_j\in\mathbf{X}_t}\lambda_{X_j\to g_n}(x_j)\\
&\cdot\left(\partial_{\mathbf{x}_s}\Big[g_n(\mathbf{x}_n,r_n)\Big]\lambda_{R_n\to g_n}(\mathbf{r},\mathbf{x}) + \partial_{\mathbf{x}_s,r_n}\Big[g_n(\mathbf{x}_n,r_n)\Big]\mu_{R_n\to g_n}(\mathbf{r},\mathbf{x})\right),\\
\lambda_{g_n\to X_k}(\mathbf{r},\mathbf{x}) = &\sum_{\substack{s,t|\mathbf{X}_s\cup\mathbf{X}_t=\mathbf{X}_n\setminus X_k\\ \mathbf{X}_s\bigcap\mathbf{X}_t=\emptyset}}\prod_{j|X_j\in\mathbf{X}_s}\mu_{X_j\to g_n}(x_j)\prod_{j|X_j\in\mathbf{X}_t}\lambda_{X_j\to g_n}(x_j)\\
&\cdot\left(\partial_{\mathbf{x}_s,x_k}\Big[g_n(\mathbf{x}_n,r_n)\Big]\lambda_{R_n\to g_n}(\mathbf{r},\mathbf{x}) + \partial_{\mathbf{x}_s,x_k,r_n}\Big[g_n(\mathbf{x}_n,r_n)\Big]\mu_{R_n\to g_n}(\mathbf{r},\mathbf{x})\right)
\end{aligned}$$

- For each player score node X_k,

$$\begin{aligned}
\mu_{X_k\to s_k}(\mathbf{r},\mathbf{x}) &= \mu_{g_n\to X_k}(\mathbf{r},\mathbf{x}),\\
\lambda_{X_k\to s_k}(\mathbf{r},\mathbf{x}) &= \lambda_{g_n\to X_k}(\mathbf{r},\mathbf{x}).
\end{aligned}$$

- Update player skill functions $s_k(x_k)$ using the multiplicative rule

$$s_k(x_k) \leftarrow s_k(x_k)\mu_{g_n\to X_k}(\mathbf{x},\mathbf{r}).$$

References

Drton, M. and Richardson, T.S. (2008) Binary models for marginal independence. *Journal of the Royal Statistical Society, Series B*, **70**, 287-309.

Elo, A.E. (1978) *The rating of chess players: Past and present.* Arco Publishing.

Gumbel, E. J. and Mustafi, C. K. (1967) Some analytical properties of bivariate extreme distributions. *Journal of the American Statistical Association*, **62**, 569-588.

Herbrich, R., Minka, T.P. and Graepel, T. (2007) TrueSkill$^{\text{TM}}$: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems (NIPS)*, **19**, 569-576.

Huang, J.C. and Frey, B.J. (2008) Cumulative distribution networks and the derivative-sum-product algorithm. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI)*, 290-297.

Huang, J.C. (2009) *Cumulative distribution networks: Inference, estimation and applications of graphical models for cumulative distribution functions.* University of Toronto Ph.D. thesis.

Kauermann, G. (1996) On a dualization of graphical Gaussian models. *Scandinavian Journal of Statistics*, **23**, 105-116.

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

Lauritzen, S. (1996) *Graphical models.* Oxford Science Publications.

Lehmann, E. L. (1955) Ordered families of distributions. *The Annals of Mathematical Statistics*, **26**, 399-419.

McCullagh, P. (1980) Regression models for ordinal data. *Journal of the Royal Statistical Society, Series B (Methodological)*, **42(2)**, 109-142.

Minka, T.P. (2001) Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 362-369.

Neal, R.M. (1993) Probabilistic inference using Markov chain Monte Carlo methods, Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.

Nelsen, R.B. (1999) *An introduction to copulas.* Springer.

Papoulis, A. and Pillai, S.U. (2001) Probability, random variables and stochastic processes. *McGraw Hill.*

Pearl, J. (1988) *Probabilistic reasoning in intelligent systems.* Morgan Kaufmann.

Richardson, T.S. and P. Spirtes (2002) Ancestral graph Markov models. *Annals of Statistics*, **30**, 962-1030.

Richardson, T.S. (2003) Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, **30**, 145-157.

Shaked, M. and Shanthikumar, J.G. (1994). *Stochastic orders and their applications.* Academic Press.

Silva, R. and Ghahramani, Z. (2009) Factorial mixture of Gaussians and the marginal independence model. In *Proceedings of the Twelfth Annual Conference on Artificial Intelligence and Statistics (AISTATS)*, *JMLR: W & CP*, **5**, 520-527.

Silva, R. and Ghahramani, Z. (2009). The hidden life of latent variables: Bayesian learning with mixed graph models. *Journal of Machine Learning Research.*

Wasserman, L. (2004) All of statistics: A concise course in statistical inference. *Springer.*

Weiss, Y. and Freeman, W.T. (2001) Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, **13**, 2173-2200.