Exact inference and learning for cumulative distribution functions on loopy graphs

Jim C. Huang, Nebojsa Jojic and Christopher Meek

NIPS 2010

Presented by Jenny Lam

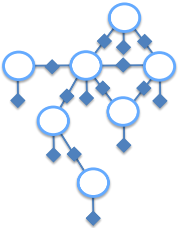
Previous work

- Cumulative distribution networks and the derivative-sum-product algorithm. Huang and Frey, 2008. UAI.
- Cumulative distribution networks: Inference, estimation and applications of graphical models for cumulative distribution functions. Huang, 2009. Ph.D. Thesis.
- Maximum-likelihood learning of cumulative distribution functions on graphs. Huang and Jojic, 2010. Journal of ML research.

Cumulative Distribution Network: definition

A CDN $\mathcal{G}$ is a bipartite graph (V, S, E) where

- V is the set of variable nodes,
- S is the set of function nodes,
  with $\phi : \mathbf{R}^{|N(\phi)|} \to [0, 1]$ is a CDF,
- E is the set of edges, connecting functions to their variables.



The joint CDF of this CDN is $F(x) = \prod_{\phi \in S} \phi$.

CDNs: what are they for?

- PDF models must enforce a normalization constraint.
- PDFs are made more tractable by restricting to, e.g., Gaussians.
- Many non-Gaussian distributions are conveniently parametrized as CDFs.
- CDNs can be used to model heavy-tailed distributions, which are important in climatology and epidemiology.

Inference from joint CDF

Conditional CDF

$$F(\mathbf{x}_B|\mathbf{x}_A) = \frac{\partial_{\mathbf{x}_A} F(\mathbf{x}_A, \mathbf{x}_B)}{\partial_{\mathbf{x}_A} F(\mathbf{x}_A)}$$

Likelihood

$$P(\mathbf{x}|\theta) = \partial_{\mathbf{x}} F(\mathbf{x}|\theta)$$

For MLE, need gradient of log likelihood

$$\nabla_\theta \log P(\mathbf{x}|\theta) = \frac{1}{P(\mathbf{x}|\theta)} \nabla_\theta P(\mathbf{x}|\theta)$$

Mixed derivative of a product

$$\partial_{\mathbf{x}} [f \cdot g] = \sum_{U \subseteq \mathbf{x}} \partial_U f \cdot \partial_{\overline{U}} g$$

which has $2^{|x|}$ terms. More generally,

$$\partial_{\mathbf{x}} \prod_{i=1}^{k} f_i = \sum_{U_1,\ldots U_k} \prod_{i=1}^{k} \partial_{U_i} f_i$$

where we sum over all partitions $U_1, \ldots U_k$ of $\mathbf{x}$ into k subsets. There are $k^{|x|}$ terms in this sum.

Mixed derivative over a separation

Partition the functions of a CDN into M_1 and M_2

- with variable sets C_1 and C_2 and $S_{1,2} = C_1 \cap C_2$
- and G_1 and G_2 the products of functions in M_1 and M_2.

Then

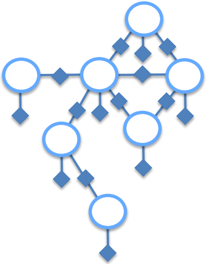
$$\partial_{\mathbf{x}} [G_1 G_2] = \sum_{A \subseteq S_{1,2}} \left[\partial_{\mathbf{x}_{C_1 \setminus S_{1,2}}} \partial_{\mathbf{x}_A} G_1\right] \left[\partial_{\mathbf{x}_{C_2 \setminus S_{1,2}}} \partial_{\mathbf{x}_{S_{1,2} \setminus A}} G_2\right]$$

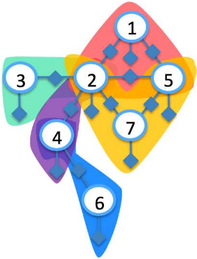
Junction Tree: definition

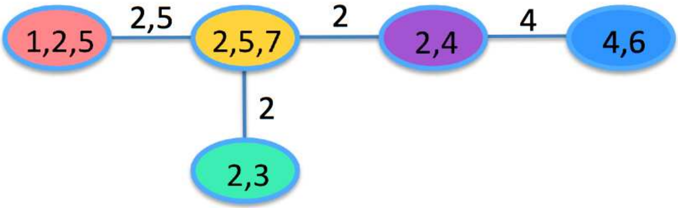
Let $\mathcal{G} = (V, S, E)$ be a CDN.
A tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ is a *junction tree* for $\mathcal{G}$ if

1. $\mathcal{C}$ is a cover for V:
   each $C_j \in \mathcal{C}$ is a subset of V and $\bigcup_j C_j = V$
2. family preservation holds:
   for each $\phi \in S$, there is a $C_j \in \mathcal{C}$ such that $scope(\phi) \subseteq C_j$
3. running intersection property holds:
   if $C_i \in \mathcal{C}$ is on the path between C_j and C_k, then $C_j \cap C_k \subseteq C_i$

Junction Tree: example







Construction of the junction tree

In implementation

- greedily eliminate the variables with the minimal fill-in algorithm
- construct elimination subsets for nodes in the junction tree using the MATLAB Bayes Net Toolbox (Murphy, 2001)

Decomposition of the joint CDF

Partitioning function of S into M_j, the joint CDF is

$$F(\mathbf{x}) = \prod_{C_j \in \mathcal{C}} \psi_j(\mathbf{x}_{C_j}), \quad \text{where } \psi_j \equiv \prod_{\phi \in M_j} \phi$$

Let r be a chosen root of the joint tree. Then

$$F(\mathbf{x}) = \psi_r(\mathbf{x}_{C_r}) \prod_{k \in \mathcal{E}_r} T_k^r(\mathbf{x})$$

where

$$T_k^r(\mathbf{x}) = \prod_{j \in \tau_k^r} \psi_j(\mathbf{x}_{C_j})$$

and τ_k^r is the subtree rooted at k.

Derivative of the joint CDF

$$
\begin{aligned}
\partial_{\mathbf{x}} F(\mathbf{x}) &= \partial_{\mathbf{x}} \left[\psi_r(\mathbf{x}_{C_r}) \prod_{k \in \mathcal{E}_r} T_k^r(\mathbf{x}) \right] \\
&= \partial_{\mathbf{x_{C_r}}} \partial_{\mathbf{x}_{\overline{\mathbf{C_r}}}} \left[\psi_r(\mathbf{x}_{C_r}) \prod_{k \in \mathcal{E}_r} T_k^r(\mathbf{x}) \right] \\
&= \partial_{\mathbf{x}_{C_r}} \left[\psi_r(\mathbf{x}_{C_r}) \, \partial_{\mathbf{x}_{\overline{\mathbf{C_r}}}} \prod_{k \in \mathcal{E}_r} T_k^r(\mathbf{x}) \right] \\
&= \partial_{\mathbf{x}_{C_r}} \left[\psi_r(\mathbf{x}_{C_r}) \prod_{k \in \mathcal{E}_r} \color{red}{\partial_{\mathbf{x}_{\tau_k^r \setminus C_r}} T_k^r(\mathbf{x})} \right]
\end{aligned}
$$

the last equality follows from the running intersection property

Messages to the root of the junction tree

Message from children k to root r, where $A \subseteq C_r$

$$m_{k\to r}(A) \equiv \partial_{\mathbf{x}_A}\left[\partial_{\mathbf{x}_{\tau_k^r \setminus C_r}} T_k^r(\mathbf{x})\right]$$

In particular

$$m_{k\to r}(\varnothing) = \partial_{\mathbf{x}_{\tau_k^r \setminus C_r}} T_k^r(\mathbf{x})$$

At the root, if $U_r \subseteq \mathcal{E}_r$, and $A \subseteq C_r$

$$m_r(A, U_r) \equiv \partial_{\mathbf{x}_A}\left[\psi_r(\mathbf{x}_{C_r}) \prod_{k\in\mathcal{E}_r} m_{k\to r}(\varnothing)\right]$$

Messages in the rest of the junction tree

$$m_i(A, U_i) \equiv \partial_{\mathbf{x}_A} \left[\psi_i(\mathbf{x}_{C_i}) \prod_{j \in U_i} m_{j \to i}(\varnothing) \right]$$

where $A \subseteq C_i$ and $U_i \subseteq \mathcal{E}_i$.

$$m_{j \to i}(A) \equiv \partial_{\mathbf{x}_A} \left[\partial_{\mathbf{x}_{\tau_j^i \setminus S_{i,j}}} T_j^i(\mathbf{x}) \right]$$

where $A \subseteq S_{i,j}$.

Messages in the rest of the junction tree

In terms of messages

$$m_i(A, U_i) = \partial_{\mathbf{x}_A}\left[\psi_i(\mathbf{x}_{C_i}) m_{k\to i}(\varnothing) \prod_{j\in U_i\setminus\{k\}} m_{j\to i}(\varnothing)\right]$$
$$= \sum_{B\subseteq A\cap S_{i,k}} m_{k\to i}(B) m_i(A\setminus B, U_i\setminus\{k\})$$

$$m_{j\to i}(A) = \partial_{\mathbf{x}_{A, C_j\setminus S_{i,j}}}\left[\psi_j(\mathbf{x}_{C_j}) \prod_{l\in\mathcal{E}_j\setminus\{i\}} T_l^j(\mathbf{x})\right]$$
$$= m_j\left(A\cup(C_j\setminus S_{i,j}),\ \mathcal{E}_j\setminus\{i\}\right)$$

Gradient of the likelihood

Likelihood

$$P(\mathbf{x}|\theta) = \partial_{\mathbf{x}} \left[F(\mathbf{x}|\theta)\right] = m_r \left(C_r, \mathcal{E}_r\right)$$

Gradient likelihood

$$\nabla_\theta m_r \left(C_r, \mathcal{E}_r\right)$$

decomposed similarly to $m_r \left(C_r, \mathcal{E}_r\right)$ in the junction tree:

- $\mathbf{g}_i \equiv \nabla_\theta m_i$
- $\mathbf{g}_{j\to i} \equiv \nabla_\theta m_{j\to i}$

JDiff algorithm: outline

for each cluster (from leaf to root):

1. compute derivative within cluster
2. compute messages from children
3. send messages to parent

foreach *Node* $j \in \mathcal{C}$ **do**

  $U_j \leftarrow \emptyset$; $\psi_j \leftarrow \prod_{s \in M_j} \phi_s$;

1 **foreach** *Subset* $A \subseteq C_j$ **do**

    $m_j(A, \emptyset) \leftarrow \partial_{\mathbf{x}_A}[\psi_j]$;

    $\mathbf{g}_j(A, \emptyset) \leftarrow \nabla_{\boldsymbol{\theta}} \partial_{\mathbf{x}_A}[\psi_j]$;

  end

2 **foreach** *Neighbor* $k \in \mathcal{E}_j \bigcap \tau_k^j$ **do**

    $S_{j,k} \leftarrow C_j \bigcap C_k$;

    foreach *Subset* $A \subseteq C_j$ **do**

      $m_j(A, U_j \bigcup k) \leftarrow \sum_{B \subseteq A \bigcap S_{j,k}} m_{k \to j}(B) m_j(A \setminus B, U_j)$;

      $\mathbf{g}_j(A, U_j \bigcup k) \leftarrow \sum_{B \subseteq A \bigcap S_{j,k}} m_{k \to j}(B) \mathbf{g}_j(A \setminus B, U_j) + \mathbf{g}_{k \to j}(B) m_j(A \setminus B, U_j)$;

    end

    $U_j \leftarrow U_j \bigcup k$;

  end

  if $j \neq r$ **then**

    $k \leftarrow \{l | \mathcal{E}_j \bigcap \tau_j^l \neq \emptyset\}$; $S_{j,k} \leftarrow C_j \bigcap C_k$;

3   **foreach** *Subset* $A \subseteq S_{j,k}$ **do**

      $m_{j \to k}(A) \leftarrow m_j\left(A \bigcup C_j \setminus S_{j,k}, \mathcal{E}_j \setminus k\right)$;

      $\mathbf{g}_{j \to k}(A) \leftarrow \mathbf{g}_j\left(A \bigcup C_j \setminus S_{j,k}, \mathcal{E}_j \setminus k\right)$;

    end

  else

    return $\left(m_r(C_r, \mathcal{E}_r), \mathbf{g}_r(C_r, \mathcal{E}_r)\right)$

  end

end

Complexity of JDiff

O-notation of number of steps/terms in each inner loop for fixed j:

1. $\displaystyle\sum_{k=1}^{|C_j|} \binom{|C_j|}{k} |M_j|^k = (|M_j| + 1)^{|C_j|}$
2. $\displaystyle(|\mathcal{E}_j| - 1) \max_{k \in \mathcal{E}_j} \sum_{l=0}^{|S_{j,k}|} \binom{|S_{j,k}|}{l} 2^{|C_j \setminus S_{j,k}|} 2^l$
3. $2^{|S_{j,k}|}$

Total. Exponential in tree-width of graph

$$O\left(\max_j (|M_j| + 1)^{|C_j|} + \max_{(j,k) \in \mathcal{E}} (|\mathcal{E}_j| - 1) 2^{|C_j \setminus S_{j,k}|} 3^{|S_{j,k}|} \right)$$

Application: symbolic differentiation on graphs

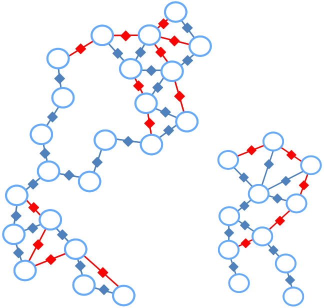
Computation of $\partial_{\mathbf{x}} F(\mathbf{x})$ on CDNs

- Grids: 3×3 to 9×9
- Cycles: 10 to 20 nodes

	JDiff	Mathematica	D*
Grids	1 s. – 20 min.	6.2 s. - ∞	9.2 s. - ∞
Cycles	0.81 s. – 2.83 s.	1.2 s. – 580 s.	6.7 s. – 12.7 s.

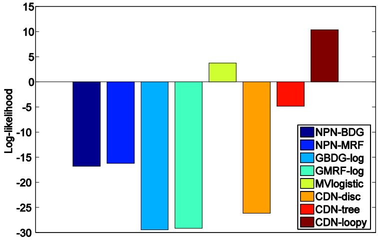
Application: modeling heavy-tailed data

- Rainfall: 61 daily measurements of rainfall at 22 sites in China
- H1N1: 29 weekly mortality rates in 11 cities in the Northeastern US during the 2008-2009 epidemic

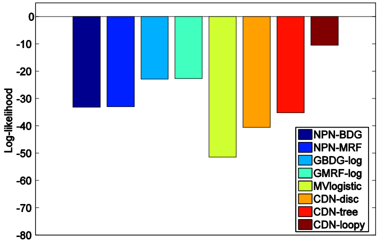


Application: modeling heavy-tailed data

Average test log-likelihoods on leave-one-out cross-validation



Rainfall data



H1N1 mortality

Future work

- Develop compact models (bounded treewidth) for applications in other areas (seismology)
- Study connection between CDNs and other copula-based algorithms
- Develop faster approximate algorithms