Focused Belief Propagation for Query-Specific Inference

Anton Chechetka
Carlos Guestrin

14 May 2010
Query-Specific inference problem

Using information about **the query** to speed up convergence of **belief propagation**

for the query marginals

\[ P(X) \propto \prod_{ij \in E} f_{ij}(X_i X_j) \]
Graphical models

- **Talk** focus: *pairwise Markov random fields*
- **Paper:** *arbitrary factor graphs* (more general)
- Probability is a product of *local* potentials
  \[ P(X) \propto \prod_{ij \in E} f_{ij}(X_i X_j) \]
- **Graphical structure**

- ☺ Compact representation
- ☹ Intractable inference
  - ☻ Approximate inference often works well in practice
(loopy) Belief Propagation

- Passing messages along edges

\[ m_{i \rightarrow k} \]

- Variable belief:

\[ \tilde{P}^{(t)}(x_i) \propto \prod_{ij \in E} m_{j \rightarrow i}^{(t)}(x_i) \]

- Update rule:

\[ m_{j \rightarrow i}^{(t+1)}(x_i) = \sum_{x_j} f_{ij}(x_i x_j) \prod_{kj \in E, k \neq i} m_{k \rightarrow j}^{(t)}(x_j) \]

- Result: all single-variable beliefs
(loopy) Belief Propagation

Update rule:

\[ m_{j \rightarrow i}^{(t+1)} (x_i) = \sum_{x_j} f_{ij} (x_i x_j) \prod_{kj \in E, k \neq i} m_{k \rightarrow j}^{(t)} (x_j) \]

Message dependencies are local:

Round–robin schedule

- Fix message order
- Apply updates in that order until convergence
Dynamic update prioritization

- **Fixed** update sequence is not the best option.
- Dynamic **update scheduling** can speed up convergence:
  - Tree-Reweighted BP [Wainwright et al., AISTATS 2003]
  - Residual BP [Elidan et al. UAI 2006]
- Residual BP $\rightarrow$ apply the **largest change** first.
Residual BP [Elidan et al., UAI 2006]

- Update rule:
  \[ m_{j\rightarrow i}^{(NEW)}(x_i) = \sum_{x_j} f_{ij}(x_i, x_j) \prod_{k \in E, k \neq i} m_{k\rightarrow j}^{(OLD)}(x_j) \]

- Pick edge with \textit{largest residual}:
  \[ \max \left| m_{j\rightarrow i}^{(NEW)} - m_{j\rightarrow i}^{(OLD)} \right| \]

- Update:
  \[ m_{j\rightarrow i}^{(OLD)} \leftarrow m_{j\rightarrow i}^{(NEW)} \]

More effort on the difficult parts of the model 😊

But no query 😞
Our contributions

- Using *weighted* residuals to prioritize updates
- Define message weights reflecting the *importance* of the message *to the query*
- Computing importance weights efficiently
- Experiments: faster convergence on large relational models
Why edge importance weights?

- Residual BP updates →
- *no influence* on the query
- wasted computation

- want to update ←
- *influence* on the query
  *in the future*

Our work → max approx. *eventual effect on* \( P(query) \)

Residual BP → max *immediate residual reduction*
Query-Specific BP

- Update rule:

\[ m_{j \rightarrow i}^{(NEW)}(x_i) = \sum_{x_j} f_{ij}(x_i x_j) \prod_{kj \in E, k \neq i} m_{k \rightarrow j}^{(OLD)}(x_j) \]

- Pick edge with

\[ \text{max} \left\| m_{j \rightarrow i}^{(NEW)} - m_{j \rightarrow i}^{(OLD)} \right\| \]

- Update

\[ m_{j \rightarrow i}^{(OLD)} \leftarrow m_{j \rightarrow i}^{(NEW)} \]

Rest of the talk:

*defining and computing edge importance*
Edge importance base case

- Pick edge with \( \max \| \Delta m_{j \to i} \| \times A_{j \to i} \)
  - approximate eventual update effect on \( P(Q) \)

Base case: edge *directly connected* to the query

\( A_{j \to i} = ?? \)

**change in query belief**

\[
\| \Delta P(Q) \| \leq \| P^{(NEW)}(Q) - P^{(OLD)}(Q) \| \times 1
\]

**change in message**

\[
\| \Delta m_{j \to i} \| \leq \| m^{(NEW)}_{j \to i} - m^{(OLD)}_{j \to i} \|
\]

tight bound
Edge importance one step away from the query: $A_{r\rightarrow j} = \text{??}$

\[
\| \Delta P(Q) \| \leq \| \Delta m_{j\rightarrow i} \| \leq \| \Delta m_{r\rightarrow j} \| \times \sup \left| \frac{\partial m_{i\rightarrow i}}{\partial m_{r\rightarrow j}} \right|
\]

| Change in query belief | \leq | Change in message | Message importance |
|------------------------|---------------------------------|------------------|
| $\|\Delta P(Q)\|$      | $\|\Delta m_{j\rightarrow i}\|$  | $\|\Delta m_{r\rightarrow j}\|$ | \sup | $\left| \frac{\partial m_{i\rightarrow i}}{\partial m_{r\rightarrow j}} \right|$ |

Can compute **in closed form** looking at only $f_{ji}$ [Mooij, Kappen; 2007]
Edge importance general case

Base case: $A_{j \rightarrow i} = 1$

One step away:

$$A_{r \rightarrow j} = \sup \left| \frac{\partial m_{j \rightarrow i}}{\partial m_{r \rightarrow j}} \right|$$

Generalization? 🙁 expensive to compute 🙁 bound may be infinite

$$\| \Delta P(Q) \| \leq \| \Delta m_{s \rightarrow h} \| \times \sup \left| \frac{\partial P(Q)}{\partial m_{s \rightarrow h}} \right|$$

sensitivity$(\pi)$: max impact along the path $\pi$
Edge importance general case

There are a lot of paths in a graph, trying out every one is intractable 😞
Efficient edge importance computation

\[ A = \max_{\text{all paths } \pi} \text{ from } \leftarrow \text{ to query } \text{sensitivity}(\pi) \]

There are a lot of paths in a graph, trying out every one is intractable 😞

**always decreases** as the path grows

\[ \text{sensitivity}(h \rightarrow r \rightarrow j \rightarrow i) = \]

Dijkstra’s (shortest paths) alg. will **efficiently** find max-sensitivity paths for every edge 😊

**decomposes** into individual edge contributions
Query-Specific BP

- Run Dijkstra’s alg starting at query to get edge weights

\[ A_{j \rightarrow i} = \max \text{ all paths } \pi \text{ from } i \text{ to } \text{query sensitivity}(\pi) \]

More effort on the difficult and relevant parts of the model

\[ \max |m_{j \rightarrow i}| = |m_{j \rightarrow i}| \times A_{j \rightarrow i} \]

- Takes into account not only graphical structure, but also strength of dependencies
Outline

- Using *weighted* residuals to prioritize updates
  - Define message weights reflecting the importance of the message *to the query*
  - Computing importance weights efficiently
    - As an initialization step before residual BP
- Restore *anytime behavior* of residual BP
- Experiments: faster convergence on large relational models
Big picture

Before

anytime property

all marginals

all marginals

After

long initialization

Dijkstra

BP

query marginals

This is broken!
Interleaving BP and Dijkstra’s

- Dijkstra’s expands the **highest weight** edges **first**
  - Can pause it at any time and get the **most relevant submodel**

When to stop BP and resume Dijkstra’s?
Interleaving

- Dijkstra’s expands the **highest weight edges first**

expanded on previous iteration

query

not yet expanded

just expanded

\[
\min_{\in \text{expanded edges}} A_{ij} \geq A
\]

suppose \( M \geq \max_{j \rightarrow i \in \text{ALL EDGES}} \left| m^{(NEW)}_{j \rightarrow i} - m^{(OLD)}_{j \rightarrow i} \right| \)

\[
\text{actual priority of } \leftarrow \quad \text{upper bound on } \leftarrow \text{priority}
\]

\[
\max_{j \rightarrow i \in \text{EXPANDED}} \left| m^{(NEW)}_{j \rightarrow i} - m^{(OLD)}_{j \rightarrow i} \right| \times A_{j \rightarrow i} = M \times \min_{\in \text{expanded}} A
\]

no need to expand further at this point
Any-Time query-specific BP

Query-specific BP:

Dijkstra’s alg.

BP updates

same BP update sequence!

Anytime QSBP:
Experiments – the setup

Relational model: semi-supervised people recognition in collection of images [with Denver Dash and Matthai Philipose @ Intel]
- One variable per person
- Agreement potentials for people who look alike
- Disagreement potentials for people in the same image
- 2K variables, 900K factors

Error measure: KL from the fixed point of BP
Experiments – convergence speed

FACES

Anytime Query-Specific BP (with Dijkstra’s interleaving)

Query-Specific BP

Residual BP

better

KL from FP

Time, sec

0 2000 4000 6000 8000

0 0.2 0.4 0.6 0.8 1
Conclusions

- Prioritize updates by **weighted** residuals
  - Takes *query info* into account

- Importance weights depend on both the **graphical structure** and **strength of local dependencies**

- **Efficient computation** of importance weights

- Much **faster convergence** on large relational models

Thank you!