Fast b-Matching via Sufficient Selection Belief Propagation

Bert Huang and Tony Jebara
bert@cs.columbia.edu, jebara@cs.columbia.edu
Computer Science Department, Columbia University

Summary

Maximum weight perfect b-matching (b-matching) is useful for resource allocation, semi-supervised learning, spectral clustering, graph embedding, and manifold learning.

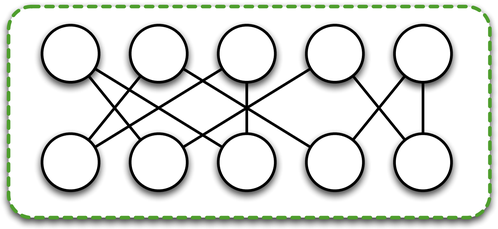
Posing b-matching as probabilistic inference yields a lightweight *belief propagation* (BP) solver: $O(N^3)$ time and $O(N^2)$ space for a dense graph of N nodes (Huang & Jebara 2007).

We provide two speedups that significantly improve the scalability of BP for b-matching.

Space: $O(N^2) \rightarrow O(N)$. We unroll recursion, and characterize beliefs with two $N \times 1$ vectors instead of $N \times N$ matrix.

Time: $O(N^3) \rightarrow O(N^{2.5})$. We compute message updates via *sufficient selection*, based on faster BP by McAuley & Caetano (2010).

Maximum Weight Perfect b-Matching



Given sets of node descriptors $\{x_1, \ldots, x_m\}$ and $\{x_{m+1}, \ldots, x_{m+n}\}$, target degrees $\{b_1, \ldots, b_{m+n}\}$, and weight function W, compute

$$\underset{\mathbf{A}}{\text{argmax}} \quad \sum_{i=1}^{m} \sum_{j=m+1}^{m+n} A_{ij} W(x_i, x_j)$$

$$\text{s.t.} \quad \sum_{j=1}^{m+n} A_{ij} = b_i, \forall i, \qquad A_{ij} = A_{ji}, \forall (i,j)$$

W is any function, e.g., linear kernel, or arbitrary weights, and $N = m + n$.

Belief Propagation for b-Matching

Writing b-matching objective as *factorized probability distribution*, standard loopy BP is guaranteed to converge to true optimal setting in $O(N)$ iterations when the *LP-relaxation* is integral (Huang & Jebara 2007; Sanghavi et al. 2008).

The simplified belief update rule uses the *selection* operation, denoted

$$\sigma_k(S) = s \in S \text{ where } |\{t \in S | t \geq s\}| = k.$$

The belief update rule is

$$B_{ij}^t = W(x_i, x_j) - \sigma_{b_j}(\{B_{jk}^{t-1} | k \neq i\}).$$

$$\begin{bmatrix} B_{11}^t & \ldots & B_{1N}^t \\ \vdots & \ddots & \vdots \\ B_{N1}^t & \ldots & B_{NN}^t \end{bmatrix}$$

BP stores $N \times N$ matrix of beliefs: $O(N^2)$ additional space.

Using classical selection algorithms, each row takes $O(N)$ to update, $O(N^2)$ per iteration, $O(N^3)$ total work until convergence.

Acknowledgements/Notes

This work was supported by DHS Contract N66001-09-C-0080–"Privacy Preserving Sharing of Network Trace Data (PPSNTD) Program"

Thanks to T. Caetano and B. Shaw for helpful discussions.

Code is available at `http://www.cs.columbia.edu/~bert/code/bmatching/`

Bert is graduating this summer. Contact him with employment and collaboration opportunities:
bert@cs.columbia.edu, http://berthuang.com

Saving Space by Unrolling Recursion

The selection operation for each update returns one of two values:

$$\sigma_{b_j}(\{B_{jk}^{t-1} | k \neq i\}) \in \{\sigma_{b_j}(\{B_{jk}^{t-1} | k\}), \sigma_{b_j+1}(\{B_{jk}^{t-1} | k\})\}.$$

We name these two values $\alpha_j^t = -\sigma_{b_j}(\{B_{jk}^{t-1} | k\})$, $\beta_j^t = -\sigma_{b_j+1}(\{B_{jk}^{t-1} | k\})$.

The belief lookup rule is then $B_{ij}^t = W(x_i, x_j) + \begin{cases} \alpha_j^t & \text{if } A_{ji}^t \neq 1 \\ \beta_j^t & \text{otherwise.} \end{cases}$

The $\vec{\alpha}$, $\vec{\beta}$ beliefs and the current estimate for **A** can be computed by looking at one row at a time, and the full **B** matrix need never be stored in memory.

Algorithm 1 Belief Propagation for b-Matching. Computes the adjacency matrix of the maximum weight b-matching.

$\alpha_j^0, \beta_j^0 \leftarrow 0, \forall j$
$\mathbf{A}^0 \leftarrow [0]$
$t \leftarrow 1$
while not converged **do**
  for all $j \in \{1, \ldots, m+n\}$ **do**
    $A_{jk}^t \leftarrow 0, \forall k$
    $\alpha_j^t \leftarrow \sigma_{b_j}(\{B_{jk}^{t-1} | k\})$ {Algorithm 2}
    $\beta_j^t \leftarrow \sigma_{b_j+1}(\{B_{jk}^{t-1} | k\})$
    for all $\{k | B_{jk}^{t-1} \geq \alpha_j^t\}$ **do**
      $A_{jk}^t \leftarrow 1$
    end for
  end for
  delete $\mathbf{A}^{t-1}$, α^{t-1} and β^{t-1} from memory
  $t \leftarrow t + 1$
end while

Algorithm 2 Sufficient Selection. Given sort-order of β^t values and partial sort-order of weights, selects the b_j'th and $b_j + 1$'th greatest beliefs of row j.

$k \leftarrow 1$
bound $\leftarrow \infty$
$S \leftarrow \emptyset$
$\tilde{\alpha}_j{}^t \leftarrow -\infty$
$\tilde{\beta}_j{}^t \leftarrow -\infty$
while $\tilde{\beta}_j{}^t <$ bound **do**
  if $k \leq c$ **then**
    $u \leftarrow I_{jk}$
    if (u is unvisited and $(B_{ju}^{t-1} > \min(S))$ **then**
      $S \leftarrow (S \setminus \min(S)) \cup B_{ju}^{t-1}$
    end if
  end if
  $v \leftarrow e_k$
  if (v is unvisited and $(B_{jv}^{t-1} > \min(S))$ **then**
    $S \leftarrow (S \setminus \min(S)) \cup B_{jv}^{t-1}$
  end if
  bound $\leftarrow W(x_j, x_u) + \beta_v^{t-1}$
  $\tilde{\alpha}_j^t \leftarrow \sigma_{b_j}(S)$
  $\tilde{\beta}_j^t \leftarrow \sigma_{b_j+1}(S)$
  $k \leftarrow k + 1$
end while
$\alpha_j^t \leftarrow \tilde{\alpha}_j^t$
$\beta_j^t \leftarrow \tilde{\beta}_j^t$

Saving Time with Sufficient Selection

Each belief row is an element-wise sum of two vectors $W(x_i, \cdot)$ and $\vec{\alpha}$ or $\vec{\beta}$

McAuley & Caetano (2010) exploited similar structure by pre-sorting each vector, computing *maximization* in expected $O(\sqrt{N})$ time

We compute *b-selection*, and keep a sorted cache of the top weights

Sort $\vec{\beta}$ messages at beginning of each iteration

Sufficient selection: examine entries corresponding to greatest W and β in order. Stop looking at values after seeing b greater than bound on remaining entries:

greatest unseen entry $\leq$ (least W so far) + (least β so far)

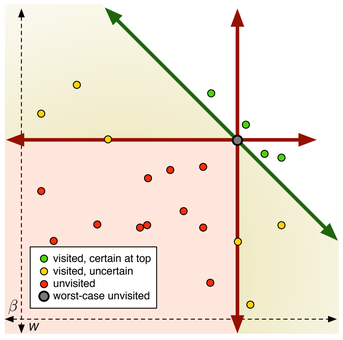


Figure 1. Visualization of sufficient selection. Selecting $W + \beta$, we can stop once b points are green. Red points are unvisited, and in the worst case are located at the gray dot.

Theorem 1. *Considering the element-wise sum of two real-valued vectors of length N with independently random sort orders, the expected number of elements that must be compared to compute the selection of the b'th greatest entry* $\sigma_b(\{w_i + \beta_i | i\})$ *is* $O(\sqrt{bN})$.

Experiments: Synthetic Data

Synthetic Gaussian data: 20 dimensional, zero-mean, negative Euclidean weights

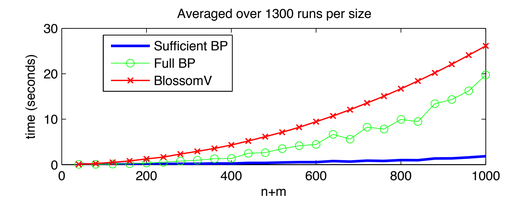


Figure 2. Running time for 1-matching compared to full BP and (unipartite) Blossom V (Kolmogorov 2009)

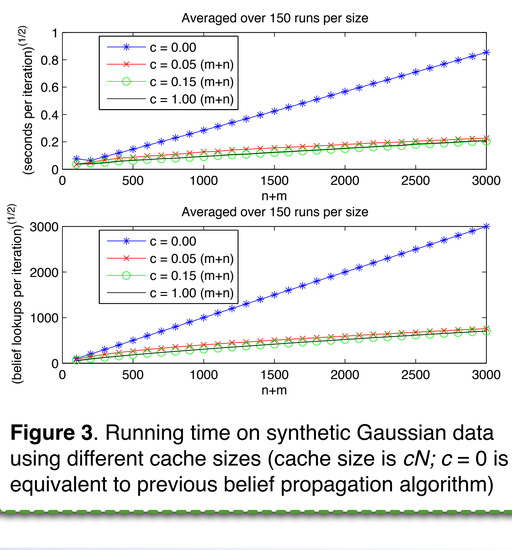


Figure 3. Running time on synthetic Gaussian data using different cache sizes (cache size is cN; $c = 0$ is equivalent to previous belief propagation algorithm)

Experiments: MNIST Digits

MNIST digit matching: Match each test digit to 6b training digits

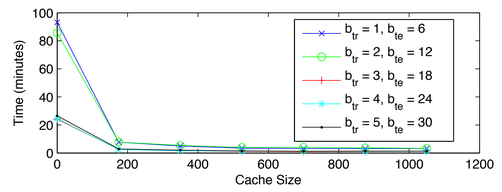


Figure 4. Running times for matching subsampled MNIST data using different cache sizes.

b_{tr}	b_{te}	Time (min.)	Belief Lookups	% Full
1	6	285.77	4.5992×10^{10}	0.94%
4	24	306.76	5.2208×10^{10}	1.11%

Table 1. Running time for full MNIST (60,000 x 10,000 candidate edges). Neither memory nor running time would be bearable on a PC without the improvements provided here.

References

B. Huang and T. Jebara. *Loopy belief propagation for bipartite maximum weight b-matching*. AISTATS 2007

V. Kolmogorov. *Blossom v: a new implementation of a minimum cost perfect matching algorithm*. Math. Programming Comp. 2009

J. McAuley and T. Caetano. *Exploiting data-independence for fast belief-propagation*. ICML 2010

S. Sanghavi, D. Maliotov, A. Willsky. *Linear programming analysis of loopy belief propagation for weighted matching*. NIPS 2007