# The Implementation and Application of Multiclass Belief Propagation on Hadoop

*Guanyu Wang*
Institute for Software Research
CMU
guanyuw@andrew.cmu.edu

*Yuchen Tian*
Institute for Software Research
CMU
yuchent@andrew.cmu.edu

*Yu Su*
Institute for Software Research
CMU
ysu1@andrew.cmu.edu

*Huanchen Zhang*
Institute for Software Research
CMU
huanchez@andrew.cmu.edu

December 3, 2012

## 1 Introduction

### 1.1 Belief Propagation

Based on the pairwise MRF graph, there is a message passing algorithm, called Belief Propagation[12] to compute the marginal possibility of unknown nodes.

The messages are defined as $m_{ij}(x_j)$, which is the chance that node j is in state $x_j$ in node i's opinion.

Given node i is in state $x_i$, we can simply get $m_{ij}(x_j) \equiv \psi(x_i, x_j)$, where $\psi(x_i, x_j)$ is the constrain between unknown node i and unknown node j.

Then, expand $x_i$ with the information of what other unknown nodes think node i:

$$m_{ij}(x_j) \equiv \sum_{x_i} \psi(x_i, x_j) \prod_{k \in N(x_i) \setminus j} m_{ki}(x_i)$$

Gather the internal constrains between the observation and unknown node i:

$$m_{ij}(x_j) \equiv \sum_{x_i} \phi(x_i)\psi(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i)$$

This describes how we update the messages. Then the final probability of node i is in state $x_i$ would be(k is used for normalization):

$$b_i(x_i) \equiv k\phi(x_i) \prod_{j \in N(i)} m_{ji}(x_i)$$

### 1.2 Fast Algorithm for Belief Background

Fast algorithm of BP approximates the BP algorithm with a matrix inversion problem[5]. It uses a linear system to approach the final solution to the Belief Propagation, call it Fast BP. Under the assumption that all probabilities are around 0.5, the approximation accuracy is quite good. The main equation for Fast BP is the following one:

$$W\mathbf{b_h} = \phi_\mathbf{h}$$

where $W$ can be viewed as a function of the adjacency matrix of the graph, degree matrix as well as the homophily factor Then the Belief Propagation problem actually has been converted into a matrix inversion problem, computing $W^{-1}\phi_h$. One main limitation now for Fast BP is that it support only two classes.

## 2 Survey

Next we list the papers that each member read, along with their summary and critique.

### 2.1 Papers read by Yu Su

The first paper was the tutorial paper on Belief Propagation by Yedidia [12]

- *Main idea*: A lot of inference models such as Bayesian Networks, Pair-wise Markov Random Field Graph, Potts models and Factor Graphs were introduced in this paper. The author also showed how to convert different models into the Pair-wise Markov Random Field Graph. Then the Standard Belief Propagation algorithm was developed on Pair-wise MRF graph and it's intuitively exact on loop-free graphs. The author applied Bethe Approximation in free energy theory to prove that BP is exact on loop-free graphs. The observation from the Kikuchi Approximation in free energy theory led to the Generalized Belief Propagation algorithm, which is constructed on clusters of the original graph.
- *Use for our project*: This paper is a good start for those who do not know Belief Propagation before. It also introduced many models and showed how to convert them, which is helpful when we are constructing models on real data sets.
- *Shortcomings*: The regional graph method introduced to solve the accuracy problem suffers exponential computing complexity.

The second paper was a deeper discussion on BP by Yedidia [13]

- *Main idea*:
  This paper introduced Factor Graph and re-expressed Belief Propagation in this model. Then the author proved that BP is exact on loop-free graph by showing the equivalence of BP and Bethe Approximation in free energy theory. A more important contribution of this paper is that it pointed out what condition must be kept to get a valid approximation on general graphs. Several methods are introduced to get better accuracy

and more chance to converge on loopy graphs, such as region graph method, junction graph method, etc.
- *Use for our project*: This paper would be very useful if we want to do some extra work such as prove the accuracy of our new method.
- *Shortcomings*: There is still no systematic method to choose regions, which influence much on accuracy and complexity. People still have to tune the algorithm for different problems to get better answers.

The third paper was sum-product algorithm by Kschischang [6]

- *Main idea*: This paper introduced sum-product algorithm on factor graph to compute marginal functions. The prevalence of the application of factor graph and sum-product algorithm is demonstrated by generalizing forward-backward algorithm, Viterbi algorithm and Kalman Filtering into sum-product algorithm. This paper summarized many methods in applying sum-product algorithm on loopy graphs. Some of them utilized the intrinsic properties of the problem and get good result, such as by carefully scheduling message passing. Some methods translated loopy graphs into loop-free graphs, such as clustering or stretching nodes.
- *Use for our project*: Many background and examples are introduced in this paper. And since sum-product algorithm is similar to BP, this paper helps understand BP.
- *Shortcomings*: The graph translation methods might be too complex to compute that the accuracy improvements might be not reachable.

## 2.2 Papers read by Yuchen Tian

The first paper was the Pioneer Error Correcting Code paper by Thomas[1]

- *Main idea*: The main contribution in Thomas's paper is brought up the idea of using Error Correcting Code(ECOC) in multiclass classification. The basic idea behind this is to convert a multiclass problem into several binary classification problem by designing a N bit code for each class, then run a classifier N times to compute how likely for a given example the probability on the $N_{th}$ bit is 1. The author then introduced a simple loss-based decoding to make decisions. In experiment, the author compared ECOC approach with One-Vs-All scheme and multiclass decision tree approach, with decision tree and neural network chosen as the underlying binary classifiers. The experiment showed that ECOC approach has better accuracy than the other two.
- *Use for our project*: It offers us a general way to solving a multiclass learning problem when we only have binary classifier(like BP in this case). It is also useful to help us get some idea how to extend the Fast BP into multiclass presentation. **More details about how this is related to BP can be found in the Methods section.**
- *Shortcomings*: Bad coding design may lead to excessive number of classifiers, thus make the computation very expensive.

The second paper was by Erin. It offers a general framework to reduce multiclass problem to binary classification problem.[7]

- *Main idea*: The main contribution in this paper is that the author presented a unifying approach of reducing multiclass classification problem to binary classification problems. By extending ECOC approach from binary to triple values and using loss-based decoding instead of hamming distance decoding, they demonstrated that One-Vs-All and All-Vs-All scheme are just special cases of ECOC approaches. This provides us a better way to understand ECOC, that it is a generalized way to solve multiclass labeling using binary classifiers.
- *Use for our project*: Unify some general and intuitive model under framework of Error Correcting Code, may simplify our assumption and design of algorithm.
- *Shortcomings*: Suffers the same problem with Thomas's paper, and loss-based decoding scheme may be expensive in Hadoop environment.

The third paper was by Ryan concerning mainly the efficiencies of various algorithms in comparing to general OVA scheme.[11]

- *Main idea*: In this paper, author argues that One-Vs-All scheme is just as accurate as other approaches and criticizes that the existing literature suffers from two major weakness: improper controlled or not well reported. If the right and well-tune classifier is chosen, then the difference between those approaches and One-Vs-All is very small. In order to hold a fair game, they chose the well tuned binary classifier such as SVM and using some statistical methods to measure whether the difference between the performance of OVA and ECOC approaches are statistically significant.
  Their conclusion comes that in performance OVA scheme has nearly identical performance as other approaches.
- *Use for our project*: A very thorough survey on comparing the efficiencies of various approaches. Very useful in helping choosing appropriate algorithm that is well-balanced on both efficiency and accuracy.
- *Shortcomings*: OVA scheme converges much slower than AVA in some datasets, while AVA scheme leads to many more classifiers to train.

## 2.3 Papers read by Huanchen Zhang

The first idea is brought up by Malewicz, introducing Pregel[8].

- *Main idea*: Pregel is in essence a message passing model developed by Google and inspired by Valiants Bulk Synchronous Parallel model, where vertices send messages to each other in a series of iterations called supersteps. The input to the Pregel framework is a directed graph. Within the graph, each vertex is associated with a modifiable, user defined value. The directed edges are associated with their source vertices, and each edge consists of a modifiable, user defined value and a target vertex identifier. The computations consist of a sequence of iteration called supersteps. In each superstep, the framework invokes a user defined function for each vertex. The function can read messages sent to V in superstep S-1, and send messages to other vertices that will be received at superstep S+1 and modify the state value of V and its

out going edges. Edges do not have associated computation. Vertices can deactivate themselves by voting to halt, which mean the vertex has no further work to do unless triggered externally, by a message from other vertex. The whole algorithm terminates when all vertices are inactive and there is no message in transit.
Out of Google, there is a similar open source project Apache Giraph.

- *Use for our project*: Pregel is a BSP framework on top of Hadoop for large-scale graph mining. Belief propagation is also a message passing algorithm on graph, so Pregel can be used to implement BP.
- *Shortcomings*:Bulk synchronous computation can be inefficient, because it needs to keep old and new messages, which means 2x overhead, and send redundant messages to vertices in the same node.

The second paper is written by U Kang's about PEGASUS[3].

- *Main idea*: PEGASUS is an open source Peta Graph Mining library implemented on top of Hadoop. Unlike the message passing model of Pregel, it views graph mining problems as a repeated matrix-vector multiplication and expresses a graph mining problem as a chained MapReduce. It provides a primitive called GIM-V (generalized iterated matric-vector multiplication), which is a generalization of normal matrix-vector multiplication. The usual matrix-vector multiplication is $M \times v = v'$ where $v'_i = \sum_{j=1}^{n} m_{i,j} v_j$. More specifically, GIM-V separates the usual matrix-vector multiplication with three functions combine2, combineAll, assign, which are implemented in MapReduce to achieve good parallelism.
  - combine2: multiply $m_{i,j}$ and $v_j$.
  - combineAll: sum n multiplication results for node i.
  - assign: overwrite previous value of $v_i$ with new result to make $v'_i$.
- *Use for our project*: As discussed in the paper, by customizing these three operations, we can obtain different, useful algorithms including PageRank, Random Walk with Restart, connected components, and diameter estimation. Belief propagation algorithm can also be implemented using GIM-V as discussed in next paper.
- *Shortcomings*: With PEGASUS, graph mining algorithms are written as a series of MapReduce jobs, which requires passing the entire state of the graph from one state to the next. This can cost much communication and serialization overhead.

The third paper is U Kang's Paper about the implementation of BP on Hadoop[4].

- *Main idea*: In this paper, belief propagation is also formulated as a variant of GIM-V. First, the original undirected graph G is converted to a directed line graph $L(G)$. The directed line graph is a graph such that each node in $L(G)$ represents an edge in $G$, and there is an edge from $v_i$ to $v_j$ of $L(G)$ if the corresponding edges $e_i$ and $e_j$ form a length-two directed path from $e_i$ to $e_j$ in $G$. Then the belief propagation is formulated in GIM-V as

$$m(s)^{next} = A' \times_G m^{cur} \tag{1}$$

where $A'$ is the adjacency matrix of $L(G)$.
Experiment shows when analyzing large scale graph which cannot fit in memory, belief propagation on Hadoop is the only solution. For medium-to-large graph whose nodes fit in memory but edges do not fit in memory, belief propagation on hadoop can also run faster than single machine BP. So, this method can be very useful if the graph is very large.

- *Use for our project*: This Hadoop-BP implementation can be used to analyze some interesting problems on our data sets.
- *Shortcomings*: Same potential problem as previous one. In the MapReduce computation framework, only the file systems (HDFS) can be used for communication and serialization. So, this chained MapReduce jobs can cost much communication and serialization overhead, because it needs multiple iterations and each iteration needs disk writes and reads.

## 2.4 Papers read by Guanyu Wang

The first paper was Fast approximation algorithm for BP by by Koutra, et al. [5]

- *Main idea*: They use a linear system to approximate the final solution to the Belief Propagation. There are mainly two key ideas for the correctness of the approximation: (1) Using the *odds ratio* instead of the original probabilities in all computations. (2) Using the Maclaurin expansion to linearize the consecutive product and taking the first-order approximation. Under the assumption that all probabilities are not far from a half, the approximation accuracy is quite good.

  Mainly, the **FaBP** solves the linear system

$$\mathbf{b_h} = [\mathbf{I} + \alpha\mathbf{D} - c'\mathbf{A}]^{-1}\phi_\mathbf{h} \tag{2}$$

  to approximate the Belief Propagation with $n$ nodes, where $\mathbf{b_h}$ (what we want to achieve) is the "about-half" approximated odds ratio of final probabilities for two classes, $\mathbf{I}$ is the $n \times n$ identity matrix, $\mathbf{D}$ is the $n \times n$ diagonal matrix of degrees, $\mathbf{A}$ is the $n \times n$ symmetric adjacency matrix, and $\phi_\mathbf{h}$ is the "about-half" approximated odds ratio of prior probabilities.

  The most important approximation technique used in the approximation is that the original sum-product message updating rule and belief updating rule can be written as continued product with just odd-ratio variables.

$$m_r(i,j) \leftarrow B[h_r, b_r(i)/m_r(j,i)] \tag{3}$$

$$b_r(i) \leftarrow \phi_r(i)\Pi_{j\in N(i)} m_r(j,i) \tag{4}$$

  where the $< v >$ are odd-ratio for different variables, i.e. $b, m, h, \phi$. $B(a,b)$ is the blending function $B(a,b) = \frac{ab+1}{a+b}$. Then using the "about-half" odds ratio to replace the common one:

$$v_r = \frac{v}{1-v} = \frac{1/2 + v_h}{1/2 - v_h} \approx 1 + 4v_h \tag{5}$$

Also, since all the updating equations are continued product, then using the Taylor expansion for logarithm after taking the logarithm on both side of Eq. (3) and Eq. (4). This finally provides us the linear update rules, which leads to the final linear system (2).

- *Why useful*: **FaBP** provides a brand-new idea that "compress" the information for any node into one variable, and using the first order approximation to simplify the computation while keeping accuracy. To know more about this point clear, refer [5] for details. The approximation technique and analysis for deriving this equation can become the fundament of our project.
- *Shortcomings*: There is an inherent property inside the **FaBP** ruins its direct extension to multi-classes problems: all the computations depends on the *odds ratio*. Obviously, $\mathbf{b_h}$ and $\phi_\mathbf{h}$ can maintain their definitions when there are only two types of probabilities. Another difficulty for the extension arises from the $\alpha$ and $c$ in Eq.(2). They all depend on a homophily factor, which depicts the similarity between two connected nodes. If there are more than two classes, how to describe the homophily (or heterophily) is not clear.

The second paper was Decision Directed Acyclic Graph (**DDAG**) algorithm by by Platt et al. [10]

- *Main idea*: Based on the idea to convert the two-class classifiers into a multi-class classifier, they provides a framework called Decision Directed Acyclic Graph (**DDAG**), whose nodes contains just binary classifier, to do multiple classification. By combing many two-class classifiers with a directed acyclic structure, these classifiers can work with order as multiple-classifier.

  More precisely, the main result in this paper is the algorithm called Directed Acyclic Graph SVM (DAGSVM), which combines the results of many 1-v-1 SVMs. So constructing a Rooted Binary DAG first, in which all the nodes evaluate a binary function, which is actually one SVM. The node is then exited via the left edge, if the binary function is zero; or the right edge, if the output SVM result is one. This process is repeated on all the following children. Any time one SVM is activated, the algorithm can get a negative conclusion about one class, i.e. this input is not in this class. It is easy to see that the algorithm can reject more classes along it moves into deep layer of the constructed DAG and finally achieve the real class. According to their analysis and experiment. The DAGSVM algorithm is superior to other multiclass SVM algorithms in both training and evaluation time.

  This work brings great connections among one-versus-one classification, one-versus-rest classification and the multiple classification problems.
- *Why useful* **FaBP** can be viewed as a good two-classifier, this paper provides a clever way to use **FaBP** as elementary unit to construct more powerful classifiers: in the multiple classes case (assume there are $n$ different classes), for any node, we can always choose any two classes and test which one this node prefer. After comparing all $\binom{n}{2}$ pairs of classes, we will have a complete partial order.

- *Shortcoming* There are mainly two weak points: (1) The **DDAG** algorithm needs a acyclic graph, however if we want to use similar ideas on Belief Propagation, there is a chance that we finally get a circle, then the original idea fails. (2) The number of binary classifiers needed to construct the multiple classes one is $N(N-1)/2$ for $N$ classes. This quadratically increasing number may hurt the real implementation (e.g. on Hadoop).

The third paper was walk-sum interpretation of Gaussian **BP** by Malioutov et al. [9]

- *Main idea*: By decomposing the correlation between each pair of variables as a sum over all walks between those variables in the graph, they provide a walk-sum interpretation for the Gaussian Belief Propagation as well as loopy Belief Propagation.

  This work develops a "walk-sum" formulation for computation of means, variances and correlations as sums over certain sets of weighted walks in a graph. It first provides a description of the walk summability: a walk of $l > 0$ in a graph $G$ is a sequence of nodes $\{w_0, w_1, \cdots, w_l\}$, $w_i \in V$ and any two consecutive nodes $(w_k, w_{k+1})$ represents an edge in this graph, i.e. $(w_k, w_{k+1}) \in E$. Define the weight of a walk to be the product of edge weights along the walk. There is a connection between the Gaussian inference and the walk. The covariance can be decomposed as a sum of infinite number of new matrices, and their new matrices' elements can be computed as a sum of walks' weight. If the sum over all walks define on a Gaussian distribution is well defined (converge to the same value for all possible summation order, etc.), then this distribution is walk-summable. The important part is that in walk-summable models, means and variances correspond to walk-sums over certain sets of walks, and the exact walk-sums over infinite sets of walks for means and variances can be computed efficiently in a recursive fashion. Finally they show that these walk-sum computations map exactly to belief propagation updates.
- *Why useful* It provides a new direction: using other expressions/understanding for Belief Propagation. We already know that **BP** can be presented as an energy minimizing problem (refer [2]). The energy function can be written as a format like $\min f(x) = g(x) + h(x)$ with differentiable part $g(x)$ and non-differentiable part $h(x)$. Usually the generalized descent gradient method can be used to solve it. At the same time, this function can hopefully be approximated by the second-order methods. Also, the walk-sum interpretation may also be written as an optimization problem with adjacency matrix and degree matrix, etc.
- *Shortcomings* This paper just discusses the Gaussian and Loopy Belief Propagation. Will similar analysis work for general **BP** is still obscure.

# 3 Proposed Method

## 3.1 Convergence Detection

We found that there is no convergence detection in U Kang's implementation of BP. The algorithm terminates after fixed number of iterations, without knowing whether it converging

or not. One more map-reduce job might be added after each iteration to perform the convergence detection, which is a useful enhancement.

## 3.2 Methods for Extending Fast BP to Multiclass Problems

### 3.2.1 Error Correcting Code Methods

#### Motivation

Fast BP is now limited to solve problems with only 2 labels. Formally extending Fast BP to multiclass can be difficult. Instead of formally formulating the multiclass Fast BP, we emphasize BP as a tool to give the correct label rather than compute the probability. Thus we can divide a single multiclass problem into several binary problems so that we can use Fast BP to give the label.

#### General Definition

To solve a multiclass problem, which has $k$ classes and $l$ training example $(x_1, y_1)...(x_l, y_l)$, we can use error correcting code approach, which usually involves the following steps:

#### Code Design

In this step, for each class in the $k$ classes, we design a unique $N$ bits code. For example, for a problem recognizing hand writings of 10 digits, we can define the following 6 bits error codes, shown in Figure 1.

| | Code Word | | | | | |
|---|---|---|---|---|---|---|
| Class | vl | hl | dl | cc | ol | or |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 1 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 0 | 1 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 1 | 0 | 0 |
| 9 | 0 | 0 | 1 | 1 | 0 | 0 |

Figure 1: Coding scheme for 10 digits recognizing problem

The meaning of those codes follows the description in the figure 2 from [1].
There are basically two principles for design a code[1]:

1. **Row Distance**: The hamming distance between each two classes' codes should be as large as possible to make classes more distinguishable from each other.

| Column position | Abbreviation | Meaning |
|---|---|---|
| 1 | vl | contains vertical line |
| 2 | hl | contains horizontal line |
| 3 | dl | contains diagonal line |
| 4 | cc | contains closed curve |
| 5 | ol | contains curve open to left |
| 6 | or | contains curve open to right |

Figure 2: Meaning of each bit

2. **Column Distance**: Each bit classifier should be independent from each other.

It is obvious that there is a trade-off between those two principles, which leads to different code design methods. We will adapt the one described in Thomas's Paper[1], which is every effective to generate compact yet distinguish code.

### Running Fast BP for Bits

As we already described in the previous sections, after designing of code, we run Fast BP for each bits, note as $f_1...f_n$. We can interpret $f_i(x)$ as the probability that the number appear on the $i^{th}$ bit of the code for a given example $x$ is 1.

### Deciding Label

For a node $r$, after computation, we get a vector of predictions of those functions:

$$f(r) = (f_1(r)...f_n(r))$$

In order to make final prediction we should find the class which is *closest* to $f(r)$.
In general, there are different approaches to achieve this:

1. **Hamming Decoding**: Transform $f(r)$ from real-value to binary value, and calculate the Hamming distance with code of each class. The label that has the shortest hamming distance is chosen.

2. **Loss-based Decoding**: This method is introduced in [7], which defines a loss function between probability made by $f$ and the target bit of class, without transform $f(r)$ into binaries.

### Practical Variations of ECOC

Error correcting code suffers from one major disadvantage, that is, if the number of classes is large, the number of bits will grow excessively, even exponentially. Sometimes, if the classification algorithm itself is time-consuming, such as Fast BP, even the number of bits is reasonable, ECOC becomes very expensive.

So we introduce a practical variations of ECOC, called One-VS-All schema. In [7], One-VS-All is proved to be a special case of ECOC.

It follows a very simple strategy: for every class, we train a binary classifier to distinguish whether a node is a class or not. The decision of final label is simple too, we just pick the class that has the highest $YES$ probability. One-VS-All will significantly reduce the number of bits involved, especially when $k$ is large. And as [11] suggests, when the underlying binary classifier performs well, One-VS-All works as well as ECOC approach.

Actually, the two schemes are just special case of ECOC, but are much more straight forward.

### 3.2.2 Quadratic approximation for FaBP extension

A significant fact of the original **FaBP** algorithm is the assumption that all parameters are "about half", i.e. close to 1/2. This assumption provides a foundation for the correctness of Maclaurin series expansion used in the analysis. However, when there are more than two classes, without the concept of odd-ratio, we cannot achieve the a similar assumption for any parameter. But we can use the quadratic approximation, i.e. second order Taylor approximation for the logarithms, i.e. instead of using $\log(x) \approx \log(1) + \log'(1)(x-1)$ with the assumption that $x$ (the odd-ratio) is close to $\frac{1}{2}$, we use

$$\log(x) \approx \log(1) + \log'(x-1) + \frac{\log''(1)}{2}(x-1)^2$$

without the "about half" assumption.

Take the original belief update rule $b_i(x_i) = \eta \cdot \phi_i(x_i) \cdot \Pi_{j \in N(i)} m_{ij}(x_i)$ for example, we can derive

$$\log(b_i(x_i)) = \log \eta + \log(\phi_i(x_i)) + \sum_{j \in N(i)} \log(m_{ij}(x_i)) \tag{6}$$

$$2b_i(x_i) - \frac{1}{2}b_i^2(x_i) - \frac{1}{2} = C_{ii} + \sum_{j \in N(i)} (2m_{ij}(x_i) - \frac{1}{2}m_{ij}^2(x_i) - \frac{1}{2}) \tag{7}$$

where $C_{ij} = \log \eta + \log(\phi_i(x_j))$. For simplicity, we assume there are three states for the node, i.e. $x_i \in \{x_1, x_2, x_3\}$. Then the left hand side of Eq. 7 is

$$\begin{bmatrix} b_i(x_1) \\ b_i(x_2) \\ b_i(x_3) \end{bmatrix} \circ \begin{bmatrix} 2 - \frac{1}{2}b_i(x_1) \\ 2 - \frac{1}{2}b_i(x_2) \\ 2 - \frac{1}{2}b_i(x_3) \end{bmatrix} - \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = diag(\mathbf{b_i})( \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} - \frac{1}{2}\mathbf{b_i}) - \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \tag{8}$$

where "$\circ$" is the hadamard product (element-wise product), $\mathbf{b_i} = \begin{bmatrix} b_i(x_1) \\ b_i(x_2) \\ b_i(x_3) \end{bmatrix}$ and note that

$$diag(b_i)\tilde{\mathbf{1}} = \begin{bmatrix} b_i(x_1) & 0 & 0 \\ 0 & b_i(x_2) & 0 \\ 0 & 0 & b_i(x_3) \end{bmatrix} \tilde{\mathbf{1}} = \mathbf{b_i} \tag{9}$$

For the right hand side of Eq. 7, since there is a neighbor sum, we have to use the adjacency matrix $\mathbf{A}$ and $\mathbf{A_i}$ is the $i$th column of the adjacency matrix. If we denote $2m_{ij}(x_i) - \frac{1}{2}m_{ij}^2(x_i) - \frac{1}{2}$ as $M_{ij}(x_i)$, the right hand side can be written as

$$C_{ii} + \begin{pmatrix} M_{i1}(x_1) & M_{i2}(x_1) & \cdots & M_{in}(x_1) \end{pmatrix} \mathbf{A_i} \tag{10}$$

Let $\mathbf{M_{ij}} = \begin{bmatrix} M_{ij}(x_1) \\ M_{ij}(x_2) \\ M_{ij}(x_3) \end{bmatrix}$, then we can re-write Eq. 7 as

$$diag(\mathbf{b_i})(2 \cdot \tilde{\mathbf{1}} - \frac{1}{2}\mathbf{b_i}) - \frac{1}{2}\tilde{\mathbf{1}} = \begin{pmatrix} \mathbf{M_{i1}} & \mathbf{M_{i2}} & \cdots & \mathbf{M_{in}} \end{pmatrix} \mathbf{A_i} + \mathbf{C_i} \tag{11}$$

also note that $2m_{ij}(x_i) - \frac{1}{2}m_{ij}^2(x_i) - \frac{1}{2}$ has the same formation of left hand side, then $M_{ij}(x_i)$ can also be written as $diag(\mathbf{m_{ij}})(2 \cdot \tilde{\mathbf{1}} - \frac{1}{2}\mathbf{m_{ij}}) - \frac{1}{2}\tilde{\mathbf{1}}$, where $\mathbf{m_{ij}} = \begin{bmatrix} m_{ij}(x_1) \\ m_{ij}(x_2) \\ m_{ij}(x_3) \end{bmatrix}$.

One problem that still need to be solved is that actually Eq. 11 is not a linear system since $\mathbf{M_{ij}}$ is a matrix with matrix entries. But we believe that there is a chance to write the general Belief Propagation as a matrix system using quadratic approximation by appropriate handling the adjacency matrix and using some approximation techniques to compress the message matrices $\mathbf{M_{ij}}$. The message update equation can also been approximated using the same method. After converting the original message passing into a matrix equation (maybe not linear, but still solvable), we can solve them to approximate the real **BP**.

# 4 Experiments

## 4.1 Datasets

### 4.1.1 DBLP dataset and other citation networks

#### Description of Dataset

DBLP is a online publication database, focused on Computer Science.

A labelled subset of DBLP is used in our experiment. It consists of 14,376 papers, 14,475 authors, 8920 terms, 20 conferences in total. A fraction of nodes are labelled with which of the following four fields it belongs to, AI, DM,IR and DB. Among the labelled nodes, 4057 authors, 100 paper, 20 conference are included.

#### Graph Construction and Information

The graph is constructed as follows: we simply put all the nodes together. So it will be heterogeneous. The basic information about this graph is shown in the Table 1 and 2.

The network is illustrated in Figure 3.

| #nodes | #edges |
|---|---|
| 4177 | 170794 |

Table 1: DBLP Network

| #DB | #DM | #AI | #IR |
|---|---|---|---|
| 1230 | 763 | 1029 | 1155 |

Table 2: Distribution of labels in DBLP Network

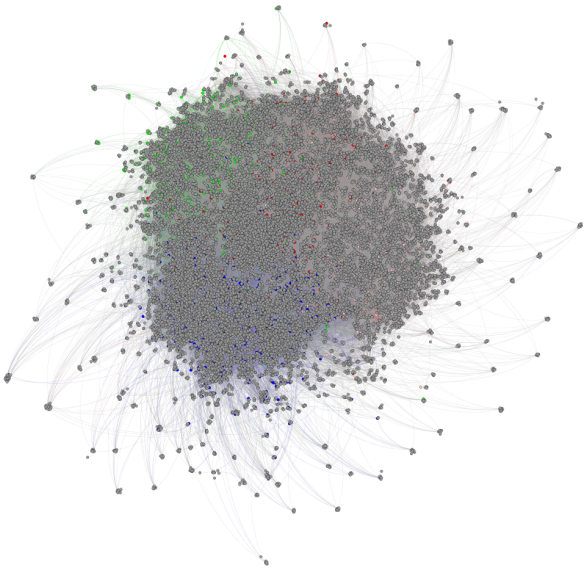

Figure 3: Visualization of DBLP network

#### 4.1.2 Social Networking(KDD Cup 2012)

**Description of Dataset**

This dataset comes from Tencent Weibo, one of the largest micro-blogging website in China. It contains many different types of information and was chosen to be the material for KDD Cup 2012.

Despite the original purpose of the competition of KDD Cup, we found something interesting from the dataset: We can get the label of some users. There is a category system in which users are labeled with hierarchy categories. And there is a following history for each user.

#### Methods of Experiments on Dataset

As the BP algorithm is defined on undirected graphs, we need to fit the dataset into a more applicable format. The following relationship is naturally uni-directional, however, most people would follow back to his/her follower if he/she found that the particular follower is similar to himself/herself. Thus, we can keep the users and their bi-directional relationship to get a undirected graph in which edges indicates the similarity of two nodes.

After filtering the dataset, we got 2,087,070 edges in between 463,605 nodes. 6,095 of the nodes are labeled as one of 6 main categories.

### 4.1.3 Amazon Product co-purchasing networks

#### Description of Dataset

This network represents co-purchasing relationship between products on Amazon website. Most nodes fell into four major categories: Book, Music, DVD and Videos. The whole net work consists of 542684 nodes and 3387388 edges in all.

#### Graph Construction and Information

Although the relationship of this network is defined as co-purchasing, actually it is not symmetry. So we first filter a subgraph, in which each edge represents mutual co-purchasing relationship. The basic information of this mutual co-purchasing network is illustrated in Table 3 and 4.

Because this a huge graph, the full scale picture of this graph mixed everything together and it is hard to tell any useful pattern from this graph, so we randomly sample a 5000 node subgraph, which is shown Figure 5.

| #nodes | #edges |
|---|---|
| 393361 | 922867 |

Table 3: Amazon mutual co-purchasing Network

| #Books | #Music | #DVD | #Video |
|---|---|---|---|
| 287678 | 74867 | 18923 | 11893 |

Table 4: Distribution of labels in Amazon Network

### 4.1.4 Flickr photo network

#### Description of Dataset

This data set is from MIR (Multimedia Information Retrieval) FLICKR Retrieval Evaluation.In this dataset, photos are associated with one or more tags from Flickr user. Photos

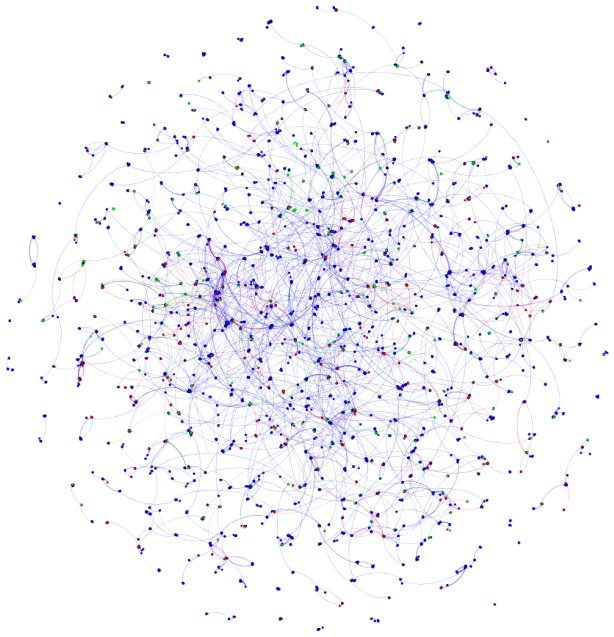

Figure 4: Visualization of Amazon network

are also annotated manually by annotator with a category. We build this network of photos based on whether two photos share same tags.

There are 22,872 photos that have at least one tag in this dataset. Tags are from Flickr users, like dog, doors, ocean etc. For photos with at least one tag, each of them is associated with 9.8 tags on average. The photo with largest number of tags has 75 tags. The photo with smallest number of tags has only 1 tag. The number of photos only with 1 tag is 813. The number of photos with 2 tags is 987.

The photos are manually annotated with one or more of the 24 categories like animals, baby, indoor etc. There are 24,581 photos with at least one category. For photos with at least one category, each of them is associated with 3.8 categories on average. The most widely used category is people (10,373 times) followed by structures (9992 times).

### Methods of Experiments on Dataset

We can build a network of photos based on whether two photos share tags. If we build a network of photos based on whether two photos share 1 tag, the average degree is 781. If we build a network of photos based on whether 2 photos share at least two tags, the average degree is 92. If we build a network of photos based on whether two photos share at least 3 tags, the average degree is 16. In later experiment, we can try different network building criterion.

With this network of photos, we can run BP and **FaBP**, and predict the category of photos.

## 4.2 Experiment Settings

### Overview

In order to validate the performance of our proposed methods, on each graph we carried out the following experiments:

1. **MBP**: Multiclass BP
2. **OVA**: One-VS-ALL schema using **FaBP** as underlying classifier
3. **ECOC**: Error correcting code schema using **FaBP** as underlying classifier

### Multiclass BP

In order to run Multiclass BP, we should first speicfy the following parameters:

- $\phi$ : Priors for all the nodes.
- $\psi$ : The correlation matrix which specifies the *closeness* between two of labels.

To estimate $\pi$, before each experiment, we randomly select $p\%$ of the set of nodes whose labels are already known to us. The $p\%$ for each graph is listed in Table ?.

| **Datasets** | **p** |
|---|---|
| DBLP | 20% |
| Flickr | 8% |
| Tencent | 5% |
| Amazon | 30% |

Table 5: Percentage of labelled data in prior $\phi$

As to correlation matrix $\phi$, we estimate this matrix using the following strategy: for a pair of given labels, say A and B, we calculate all the edges between the two labels, denoted as $\#(AB)$. For each label, we also compute the number of edges associated with that label, which means all the edges that have at least one end being the given label. In our case, we denote them as $\#(A)$ and $\#(B)$. Then $\psi$ is defined as follows:

$$\psi_{A\to B} = \frac{\#(AB)}{\#(A)}$$
$$\psi_{B\to A} = \frac{\#(AB)}{\#(B)}$$

Repeat the above procedure for each pair of labels, then we have the estimation of the correlation matrix.

### One-VS-All with FaBP

**FaBP** algorithm is parameterized by one single important factor: the "about-half" homophily factor, denoted as $h_h$. The bigger $h_h$ means the stronger the homophily phenomenon in the graph.

| **Datasets** | $h_h$ |
|---|---|
| DBLP | 0.002 |
| Flickr | 0.2 |
| Tencent | 0.002 |
| Amazon | 0.01 |

Table 6: The homophily factor for each network

As to the prior $\phi$, we follows the same settings with **MBP**.

When the algorithm finished, for each node, say $N$, we combine all the k probabilities $P(N = k_i)$ in a vector, which specifies how likely the given node is within that class. Then we pick the class that has the greatest $P(N = k_i)$ as the label of that node.

### ECOC with FaBP

The settings of prior $\phi$ and homophily factor $h_h$ is the same as those in **OVS**.

When coming to the code design, we adopts the exhaustive algorithm described in [1]. When given $k$ labels, each label is encoded using $2^{k-1} - 1$ bits. For the first class, we encode it using all ones. For the second class, we set the first $2^{k-2}$ bits of its coding as zeros, then the next $2^{k-2} - 1$ bits as ones. For the third class, we set the first $2^{k-3}$ bits as zeros, the following $2^{k-3}$ as ones, which are followed by $2^{k-3} - 1$ zeros. Repeating the this procedure until all the classes are defined.

The code matrix when $k = 4$ is illustrated in Table 7.

| **class** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

Table 7: Code design when $k = 4$

As described in **Methods** section, we run **FaBP** for each bit. For each node, after we get the probability vector, we transform it into a binary vector, using the following strategy: for a given bit $b_i$, if $P(b_i = 1) > 0.5$, then the corresponding position of the binary vector is set to 1, else to zero. Then we calculated the Hamming distance between this vector and coding vector of each class, the one has the smallest distance is assigned to the node.

### 4.3 Experiment Results

All the results are illustrated in the following table.

| Datasets | MBP | OVS | ECOC |
|---|---|---|---|
| DBLP | 0.17 | **0.776** | 0.726 |
| Flickr | **0.653** | 0.421 | 0.13 |
| Tencent | 0.002 | | |
| Amazon | **0.736** | 0.636 | 0.626 |

Table 8: The homophily factor for each network

# References


[1] Thomas G. Dietterich.et.al. Solving multiclass learning problem via error-correcting output codes. *Journal of Artifical Intelligence Research*, 1995.

[2] Pedro Felzenszwalb, , Pedro F. Felzenszwalb, and Daniel P. Huttenlocher. Efficient belief propagation for early vision. In *In CVPR*, pages 261–268, 2004.

[3] U. Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pages 229–238, Washington, DC, USA, 2009. IEEE Computer Society.

[4] U Kang.et.al. Inference of beliefs on billion-scale graphs. *Large-scale Data Mining: Theory and Applications 2010, in conjunction with KDD 2010*, 2010.

[5] Danai Koutra, Tai-You Ke, U. Kang, Duen Horng Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *ECML/PKDD (2)*, pages 245–260, 2011.

[6] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 47:498–519, 1998.

[7] Erin L.Allwein.et.al. Reducing multiclass to binary : A unifying approach for margin classiers. *Journal of Machine Learning Research*, 2000.

[8] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[9] Dmitry M. Malioutov, Jason K. Johnson, and Alan S. Willsky. Walk-sums and belief propagation in gaussian graphical models. *J. Mach. Learn. Res.*, 7:2031–2064, December 2006.

[10] John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[11] Ryan Rifkin.et.al. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 2004.

[12] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Exploring artificial intelligence in the new millennium. chapter Understanding belief propagation and its generalizations, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[13] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2005.

# A Appendix

## A.1 List of Innovations

- Evaluate BP and Fast BP on many real world datasets.
- Improve BP on Hadoop with convergence detection.
- Use ECOC to extend Fast BP to multiclass.
- Use quadratic approximation to extend Fast BP to multiclass.

## A.2 Labor Division

The team will perform the following tasks

- Implementation of BP on Hadoop with convergence detection [Yu Su,Huanchen Zhang,1 weeks]
- Parse data and build network from datasets mentioned above [All, almost done]
- Evaluate BP and Fast BP on large graphs [All, already have preliminary results, 1 week]
- Research on extending Fast BP algorithm to Multiclass Problem [Guanyu Wang,Yuchen Tian, 2 weeks]
- Implement and evaluate multiclass Fast BP algorithm [Yu Su,Huanchen Zhang,1 weeks]

## A.3 Activities

- Guanyu Wang: Performed dataset analysis and network building. Read papers and tried to deduct quadratic approximation for FaBP extension.
- Yuchen Tian: Performed dataset analysis and network building. Proposed error correcting code methods for FaBP extension.
- Yu Su: Performed dataset analysis and network building. Did some preliminary experiments.
- Huanchen Zhang: Performed dataset analysis and network building. Did some preliminary experiments.

## A.4 Previous Labor Division

The team will perform the following tasks

- Implementation of BP on Hadoop [Yu Su,Huanchen Zhang,2 weeks]
- Apply BP on Large Graph [All,2 weeks]
- Research on extending Fast BP algorithm to Multiclass Problem [Guanyu Wang,Yuchen Tian,2 weeks]

# Contents