Gaussian Belief Propagation for Solving Systems of Linear Equations: Theory and Application

Danny Bickson

School of Computer Science and Engineering
The Hebrew University of Jerusalem

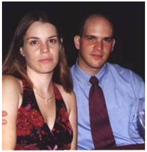
15/5/08

Joint work

UCSD:

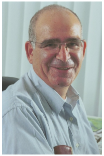
Jack K. Wolf

Paul H. Siegel



Ori Shental

HUJI:

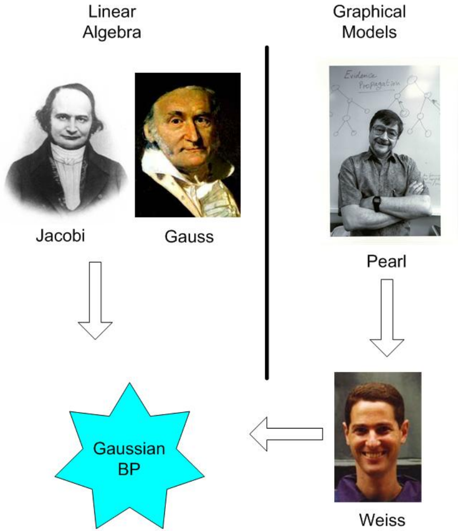


Danny Dolev

Danny Bickson

- NSF Grant No. CCR-0514859
- EVERGROW, IP 1935 of the EU Sixth Framework

Talk outline



Take-home message

- New approach: solving a linear system of algebraic equations as a probabilistic inference problem.
- Gaussian belief propagation (GaBP) solver:
  - Iterative
  - Convergent
  - Exact
  - Efficient
  - Distributed message-passing implementation for very large systems
  - Superior to classical iterative methods
  - Countless applications in the mathematical sciences and engineering

Outline

1. Theory
   - Introduction
   - Derivation
   - The GaBP solver algorithm
   - Properties
2. Application
   - Linear Detection

Problem formulation

Definitions

- $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n \in \mathbb{N}^*$, is a given data matrix.
- $\mathbf{b} \in \mathbb{R}^m$ is a given observation vector.
- $\mathbf{x} \in \mathbb{R}^n$ is a vector of unknown variables.

System of linear equations

$$\mathbf{Ax} = \mathbf{b}$$

Solution

- A unique solution, $\mathbf{x}^*$, exists iff $\mathbf{A}$ has full column rank.
- $\mathbf{x}^* = \mathbf{A}^\dagger \mathbf{b}$, where $\mathbf{A}^\dagger \triangleq (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the Moore-Penrose pseudoinverse matrix.

Problem formulation (cont.)

Assumption

The data matrix $\mathbf{A}$ is square (*i.e.* , $m = n$) and symmetric.

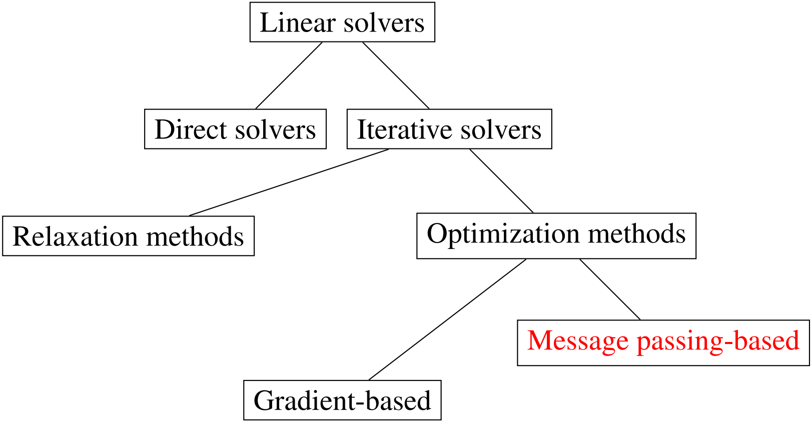
Solution

$$\mathbf{x}^* = \mathbf{A}^{\dagger}\mathbf{b} = \mathbf{A}^{-1}\mathbf{b}$$

Related problems

- Efficient distributed (large) matrix inversion or
- Determinant computation.

GaBP solver and classical solution methods



From linear algebra to probabilistic inference

Proposition [Bickson *et al.* ,'07]

The computation of the solution vector, $\mathbf{x}^*$, is equivalent to the inference of the vector of marginal means, $\mu \in \mathbb{R}^n$, over the graph $\mathcal{G}$ with the associated joint Gaussian probability density function $p(\mathbf{x}) \sim \mathcal{N}(\mu \triangleq \mathbf{A}^{-1}\mathbf{b}, \mathbf{A}^{-1})$.

From linear algebra to probabilistic inference (cont.)

Proof.

- Define a quadratic form: $q(\mathbf{x}) \triangleq \mathbf{x}^T\mathbf{A}\mathbf{x}/2 - \mathbf{b}^T\mathbf{x}$.
- $\mathbf{A}$ is symmetric $\Rightarrow \partial q(\mathbf{x})/\partial\mathbf{x}|_{\mathbf{x}^*} = \mathbf{A}\mathbf{x}^* - \mathbf{b} = \mathbf{0}$.
- Define a joint Gaussian probability density function using the quadratic form

$$
\begin{aligned}
p(\mathbf{x}) &\propto \exp\big(-q(\mathbf{x})\big) = \exp\left(-\mathbf{x}^T\mathbf{A}\mathbf{x}/2 + \mathbf{b}^T\mathbf{x}\right) \\
&\propto \exp\big(-(\mathbf{x}-\mu)^T\mathbf{A}(\mathbf{x}-\mu)/2\big) = \mathcal{N}(\mu, \mathbf{A}^{-1}),
\end{aligned}
$$

where the mean $\mu = \mathbf{A}^{-1}\mathbf{b} = \mathbf{x}^*$.

□

From linear algebra to probabilistic inference (cont.)

- Shift the solution problem from an algebraic to a probabilistic domain.
- A deterministic vector-matrix linear equation translates to an inference problem in the corresponding graph.
- Calls for the utilization of belief propagation (BP) as an efficient inference engine.

From linear algebra to probabilistic inference (cont.)

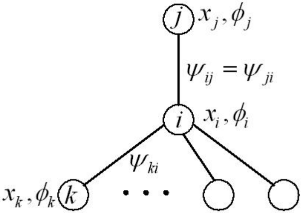
- Shift the solution problem from an algebraic to a probabilistic domain.
- A deterministic vector-matrix linear equation translates to an inference problem in the corresponding graph.
- Calls for the utilization of belief propagation (BP) as an efficient inference engine.

Remark

Data matrix $\mathbf{A}$ does **not** have to be positive semi-definite.

Graphical model

- Consider the graph $\mathcal{G}$ corresponding to the joint Gaussian $p(\mathbf{x})$, with edge potentials ψ_{ij} and self-potentials ϕ_i.
- Determined according to the pairwise factorization $p(\mathbf{x}) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{\{i,j\}} \psi_{ij}(x_i, x_j)$.



- where

$$\begin{aligned}\psi_{ij}(x_i, x_j) &\triangleq \exp(-x_i A_{ij} x_j),\\ \phi_i(x_i) &\triangleq \exp\left(b_i x_i - A_{ii} x_i^2/2\right) \propto \mathcal{N}(\mu_{ii} = b_i/A_{ii}, P_{ii}^{-1} = A_{ii}^{-1}).\end{aligned}$$

Inference

- We would like to infer the marginal densities, which must also be Gaussian

$$p(x_i) \sim \mathcal{N}(\mu_i = \{\mathbf{A}^{-1}\mathbf{b}\}_i = x_i^*, P_i^{-1} = \{\mathbf{A}^{-1}\}_{ii}).$$

- Now, (Gaussian) BP can come into action...

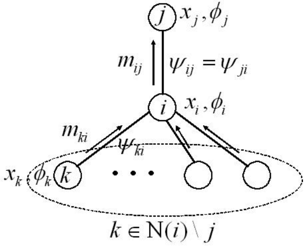
Discrete belief propagation (BP)

Sum-product rule

$$m_{ij}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k \in \mathrm{N}(i)\setminus j} m_{ki}(x_i)$$

Product rule

$$\Pr(x_i) \propto \phi_i(x_i) \prod_{k \in \mathrm{N}(i)} m_{ki}(x_i)$$



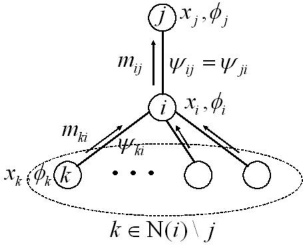
Continuous BP

Integral-product rule

$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k \in \mathrm{N}(i) \setminus j} m_{ki}(x_i) dx_i$$

Product rule

$$p(x_i) \propto \phi_i(x_i) \prod_{k \in \mathrm{N}(i)} m_{ki}(x_i)$$



Gaussian BP

- Gaussian BP (GaBP) is a special case of continuous BP, where the underlying distribution is Gaussian [Weiss and Freeman,'01].

Lemma: product of Gaussian densities

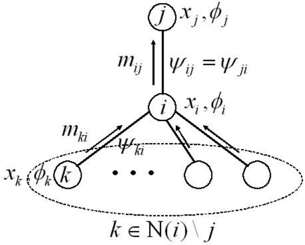
Let $f_1(x) = \mathcal{N}(\mu_1, P_1^{-1})$ and $f_2(x) = \mathcal{N}(\mu_2, P_2^{-1})$.Then their product $f(x) = f_1(x)f_2(x) \propto \mathcal{N}(\mu, P^{-1})$ where

$$\begin{aligned} \mu &\triangleq P^{-1}(P_1\mu_1 + P_2\mu_2), \\ P^{-1} &\triangleq (P_1 + P_2)^{-1}. \end{aligned}$$

Gaussian BP (cont.)

Integral-product rule

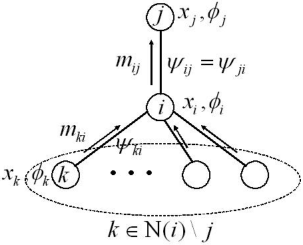
$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k \in \mathrm{N}(i)\setminus j} m_{ki}(x_i) dx_i$$



Gaussian BP (cont.)

Integral-product rule

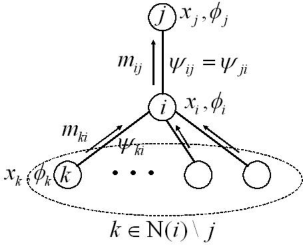
$$\phi_i(x_i) \prod_{k \in \mathrm{N}(i) \setminus j} m_{ki}(x_i)$$



Gaussian BP (cont.)

Integral-product rule

$$\phi_i(x_i) \prod_{k \in \mathrm{N}(i) \setminus j} m_{ki}(x_i)$$

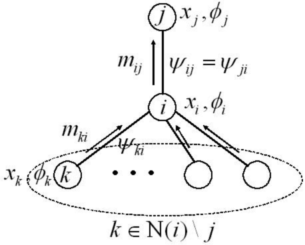


- $p(\mathbf{x})$ is jointly Gaussian⇒

Gaussian BP (cont.)

Integral-product rule

$$\phi_i(x_i) \prod_{k \in \mathrm{N}(i)\backslash j} m_{ki}(x_i)$$

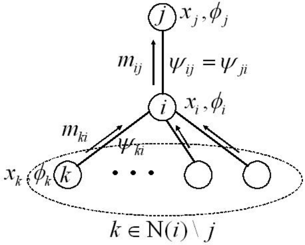


- $p(\mathbf{x})$ is jointly Gaussian$\Rightarrow$
  - Gaussian self-potentials $\phi_i(x_i) \propto \mathcal{N}(\mu_{ii} = b_i/A_{ii}, P_{ii}^{-1} = A_{ii}^{-1})$

Gaussian BP (cont.)

Integral-product rule

$$\phi_i(x_i) \prod_{k \in \mathrm{N}(i) \setminus j} m_{ki}(x_i)$$

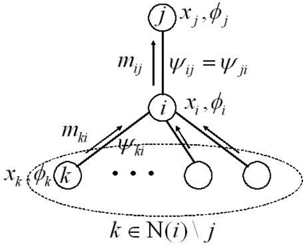


- $p(\mathbf{x})$ is jointly Gaussian$\Rightarrow$
  - Gaussian self-potentials $\phi_i(x_i) \propto \mathcal{N}(\mu_{ii} = b_i/A_{ii}, P_{ii}^{-1} = A_{ii}^{-1})$
  - Gaussian messages $m_{ki}(x_i) \propto \mathcal{N}(\mu_{ki}, P_{ki}^{-1})$

Gaussian BP (cont.)

Integral-product rule

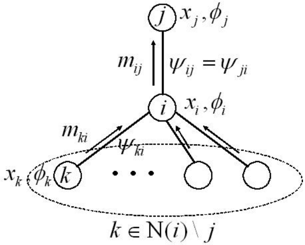
- Applying the multivariate version of the Gaussian densities product lemma:



Gaussian BP (cont.)

Integral-product rule

$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j) \mathcal{N}(\mu_{i\backslash j}, P_{i\backslash j}^{-1}) dx_i$$

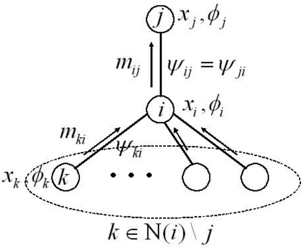


- Applying the multivariate version of the Gaussian densities product lemma:
  - Precision $P_{i\backslash j} = \overbrace{P_{ii}}^{\phi_i(x_i)} + \sum_{k\in \mathrm{N}(i)\backslash j} \overbrace{P_{ki}}^{m_{ki}(x_i)}$
  - Mean $\mu_{i\backslash j} = P_{i\backslash j}^{-1}\left(\overbrace{P_{ii}\mu_{ii}}^{\phi_i(x_i)} + \sum_{k\in \mathrm{N}(i)\backslash j} \overbrace{P_{ki}\mu_{ki}}^{m_{ki}(x_i)}\right)$

Gaussian BP (cont.)

Integral-product rule

$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j) \mathcal{N}(\mu_{i\backslash j}, P_{i\backslash j}^{-1}) dx_i$$

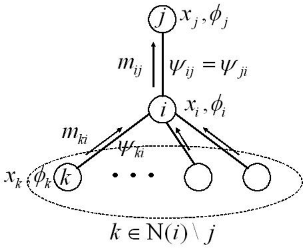


Gaussian BP (cont.)

Integral-product rule

$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j) \mathcal{N}(\mu_{i\backslash j}, P^{-1}_{i\backslash j}) dx_i$$

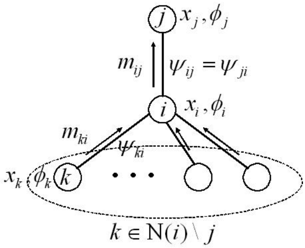
- Using the Gaussian integral $\int_{-\infty}^{\infty} \exp{(-ax^2 + bx)} dx = \sqrt{\pi/a} \exp{(b^2/4a)}$:



Gaussian BP (cont.)

Integral-product rule

$$m_{ij}(x_j) \propto \mathcal{N}(\mu_{ij}, P_{ij}^{-1})$$



- Using the Gaussian integral $\int_{-\infty}^{\infty} \exp(-ax^2 + bx)dx = \sqrt{\pi/a}\exp(b^2/4a)$:
  - Message precision $P_{ij} = -A_{ij}^2 P_{i\backslash j}^{-1}$
  - Message mean $\mu_{ij} = -P_{ij}^{-1} A_{ij} \mu_{i\backslash j}$

Gaussian BP (cont.)

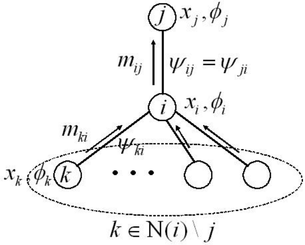
Product rule

$$p(x_i) \sim \mathcal{N}(\mu_i, P_i^{-1})$$

- Marginal precision $P_i = \overbrace{P_{ii}}^{\phi_i(x_i)} + \sum_{k \in \mathrm{N}(i)} \overbrace{P_{ki}}^{m_{ki}(x_i)}$
- Marginal mean

$$\mu_i = P_{i \setminus j}^{-1} \left(\overbrace{P_{ii}\mu_{ii}}^{\phi_i(x_i)} + \sum_{k \in \mathrm{N}(i)} \overbrace{P_{ki}\mu_{ki}}^{m_{ki}(x_i)} \right)$$

- Mean and precision like in the product term of the integral-product rule, but summing over all incoming messages.



The GaBP solver algorithm

Initialize

- ✓ Set the neighborhood $\mathrm{N}(i)$ to include $\forall k \neq i$ such that $A_{ki} \neq 0$.
- ✓ Fix the scalars $P_{ii} = A_{ii}$ and $\mu_{ii} = b_i / A_{ii}, \forall i$.
- ✓ Set the initial $k \rightarrow i, k \in \mathrm{N}(i)$ scalar messages $P_{ki} = 0$ and $\mu_{ki} = 0$.
- ✓ Set a convergence threshold ϵ.

The GaBP solver algorithm

Iterate & check

- ✓ Compute the $i \to j, i \in \mathrm{N}(j)$ scalar messages
  $P_{ij} = -A_{ij}^2 / \left(P_{ii} + \sum_{k \in \mathrm{N}(i) \setminus j} P_{ki}\right)$,
  $\mu_{ij} = \left(P_{ii}\mu_{ii} + \sum_{k \in \mathrm{N}(i) \setminus j} P_{ki}\mu_{ki}\right) / A_{ij}$.
- ✓ Propagate the $\mathrm{N}(i) \ni k \to i$ messages P_{ki} and μ_{ki}, $\forall i$ (under chosen scheduling).
- ✓ If the messages P_{ij} and μ_{ij} did not converge (w.r.t. ϵ), iterate again.
- ✓ Else, continue next step.

The GaBP solver algorithm

Infer & solve

✓ Compute the marginal means
$\mu_i = \left(P_{ii}\mu_{ii} + \sum_{k\in \mathrm{N}(i)} P_{ki}\mu_{ki}\right) / \left(P_{ii} + \sum_{k\in \mathrm{N}(i)} P_{ki}\right), \forall i.$

(✓ Optionally compute the marginal precisions
$P_i = P_{ii} + \sum_{k\in \mathrm{N}(i)} P_{ki}$)

✓ Find the solution
$x_i^* = \mu_i, \forall i.$

Convergence and Exactness

- We can use results from the literature on probabilistic inference in graphical models:

Theorem [based on Weiss and Freeman,'01,Claim 4]

If the matrix $\mathbf{A}$ is strictly diagonally dominant (*i.e.* , $|A_{ii}| > \sum_{j \neq i} |A_{ij}|, \forall i$), then the GaBP solver converges and the marginal means converge to the true solution.

- This sufficient condition can be relaxed:

Theorem [based on Johnson *et al.* ,'06,Proposition 2]

If the spectral radius (maximum of the absolute values of the eigenvalues) ρ of the matrix $|\mathbf{I}_n - \mathbf{A}|$ satisfies $\rho(|\mathbf{I}_n - \mathbf{A}|) < 1$, then the GaBP solver converges and the marginal means converge to the true solution.

Convergence and Exactness (cont.)

- Only sufficient (but not necessary) conditions are known.
- Examples for convergence when sufficient conditions do not hold:
  - Tree graphs;
  - Graph representing Gaussian-signaling randomly-spread CDMA system.
- Either converging to the exact solution or diverging.
  - Can not converge to a wrong solution.
- Exact region of convergence and convergence rate are open problems.

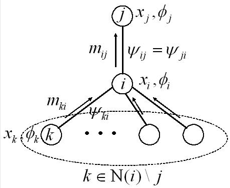
In contrast to ordinary BP:

- Convergence guarantees exactness of the inferred probabilities.
- Convergence is not limited to tree or sparse graphs, and can occur even for dense (fully-connected) graphs.

Message-passing efficiency

For a dense data matrix $\mathbf{A}$

- $\mathcal{O}(n^2)$ unique messages per iteration.
- Naive approach, because...
- Messages transmitted from a node are very much correlated:
  - Differ only in one summation term.
- Broadcast the aggregated sum messages:
  - Reduces the number of unique messages to $\mathcal{O}(n)$ per iteration.

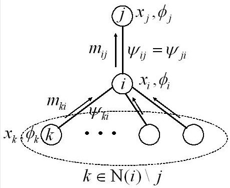


Message-passing efficiency

Iterate

✓ Broadcast the aggregated sum messages
$\tilde{P}_i = P_{ii} + \sum_{k \in \mathrm{N}(i)} P_{ki}$,
$\tilde{\mu}_i = \tilde{P}_i^{-1}(P_{ii}\mu_{ii} + \sum_{k \in \mathrm{N}(i)} P_{ki}\mu_{ki}), \forall i$
(under chosen scheduling).

✓ Compute the $\mathrm{N}(j) \ni i \rightarrow j$ internal scalars
$P_{ij} = -A_{ij}^2/(\tilde{P}_i - P_{ji})$,
$\mu_{ij} = (\tilde{P}_i\tilde{\mu}_i - P_{ji}\mu_{ji})/A_{ij}$.



Computational complexity

Well-conditioned dense data matrix ($\kappa(\mathbf{A}) \triangleq ||\mathbf{A}||_p||\mathbf{A}^{-1}||_p = \mathcal{O}(1)$)

Algorithm	Operations per message	Unique messages	Operations per iteration	Iterations	Operations
Broadcast GaBP	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$
Gaussian elimination	"	"	"	"	$\mathcal{O}(n^3)$
Jacobi method	"	"	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$

Sparse (2-D Poisson) data matrix ($\kappa(\mathbf{A}) = \mathcal{O}(n)$)

Algorithm	Operations per message	Unique messages	Operations per iteration	Iterations	Operations
Broadcast GaBP	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$< \mathcal{O}(\sqrt{n})$	$< \mathcal{O}(n\sqrt{n})$
Gaussian elimination	"	"	"		$\mathcal{O}(n^3)$
Jacobi method	"	"	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$

Linear channels

$$\mathbf{y} = \mathbf{R}\mathbf{x} + \mathbf{n}$$

- $\mathbf{x}$, input vector
- $\mathbf{n}$, additive noise vector
- $\mathbf{y}$, output of a bank of filters matched to the physical channel $\mathbf{S}$
- $\mathbf{R} = \mathbf{S}^T\mathbf{S}$, correlation matrix

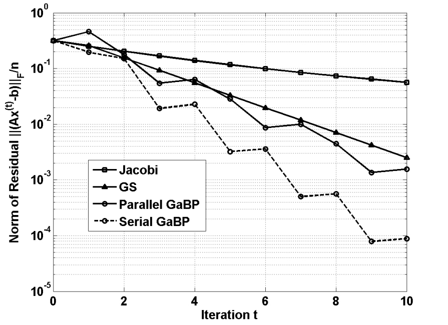
Linear detection

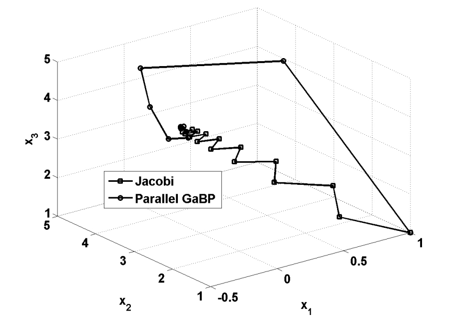
$$\hat{\mathbf{x}} = \Delta\{\mathbf{x}^*\} = \Delta\{\mathbf{A}^{-1}\mathbf{b}\}$$

- $\mathbf{x} = \{x_1, \dots, x_K\}^T$, hidden input vector
- $\mathbf{b} = \mathbf{y} = \{y_1, \dots, y_K\}^T$, observed noisy output vector
- $\mathbf{A}$, $K \times K$ positive-definite symmetric matrix approximating the channel transformation
- $\mathbf{x}^*$, solution to $\mathbf{Ax} = \mathbf{b}$
- $\Delta\{\cdot\}$, clipping to input alphabet
- $\hat{\mathbf{x}}$, decision

Application examples (cont.)

Linear detection (decorrelation) in CDMA:





Linear detection (cont.)

Setup

- CDMA
- Gold spreading sequences of length $N = 7$.
- $K = 3$ and $K = 4$ users$\Rightarrow$Correlation matrices $\mathbf{R}_3$ and $\mathbf{R}_4$, which are not diagonally dominant, but $\rho(|\mathbf{I}_3 - \mathbf{R}_3|) = 0.9008 < 1$ and $\rho(|\mathbf{I}_4 - \mathbf{R}_4|) = 0.8747 < 1$.
- Decorrelator ($\mathbf{A} = \mathbf{R}$) detector.
- $\mathbf{b}(= \mathbf{y} = \mathbf{Rx} + \mathbf{n})$ is all-1's.
- Comparison to MUD based on classical iterative methods [Grant & Schlegel,'99],[Tan & Rasmussen,'00],[Yener *et al.* ,'02].

Linear detection (cont.)

Algorithm	Iterations t ($\mathbf{R}_3$)	Iterations t ($\mathbf{R}_4$)
Jacobi	111	24
GS	26	26
Parallel GaBP	**23**	**24**
Optimal SOR	17	14
Serial GaBP	**16**	**13**
Jacobi+Steffensen	59	—
Parallel GaBP+Steffensen	**13**	**13**
Serial GaBP+Steffensen	**9**	**7**

Future directions

- Finding the exact region of convergence and convergence rate.
  - Parallel vs. serial scheduling
- Variety of applications.

Take-home message

- New approach: solving a linear system of algebraic equations as a probabilistic inference problem.
- Gaussian belief propagation (GaBP) solver:
  - Iterative
  - Convergent
  - Exact
  - Efficient
  - Distributed message-passing implementation for very large systems
  - Superior to classical iterative methods
  - Countless applications in the mathematical sciences and engineering

References

[Bickson *et al.*,'07]

- "Gaussian belief propagation for solving systems of linear equations: Theory and application" (Trans. IT submission).
- "Gaussian belief propagation solver for systems of linear equations" (ISIT '08).
- "Gaussian belief propagation based multiuser detection" (ISIT '08).
- "Linear detection via belief propagation" (proc. of Allerton '07).
- "A message-passing solver for linear systems" (proc. of ITA '08).
- "Peer-to-Peer rating" (proc. of P2P computing '07)
- "A unifying framework for rating users and data items in Peer-to-Peer and social networks" (PPNA Journal '08)
- "Large scale Gaussian BP solver for kernel ridge regression" (NIPS workshop '07)

References

[Weiss and Freeman,'01] "Correctness of belief propagation in Gaussian graphical models of arbitrary topology".

[Johnson *et al.* ,'06]

- "Walk-sum interpretation and analysis of Gaussian belief propagation".
- "Walk-sums and belief propagation in Gaussian graphical models".

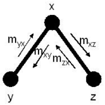
[Plarre and Kumar,'04] "Extended message passing algorithm for inference in loopy Gaussian graphical models".

THANK YOU!

Toy linear system

3×3 equations

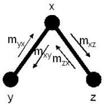
$$\underbrace{\begin{pmatrix} A_{xx}=1 & A_{xy}=-2 & A_{xz}=3 \\ A_{yx}=-2 & A_{yy}=1 & A_{yz}=0 \\ A_{zx}=3 & A_{zy}=0 & A_{zz}=1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} -6 \\ 0 \\ 2 \end{pmatrix}}_{\mathbf{b}}$$



Toy linear system

3×3 equations

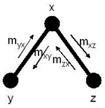
$$\underbrace{\begin{pmatrix} x^* \\ y^* \\ z^* \end{pmatrix}}_{\mathbf{x}^*} = \underbrace{\begin{pmatrix} -1/12 & -1/6 & 1/4 \\ -1/6 & 2/3 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{pmatrix}}_{\mathbf{A}^{-1}} \underbrace{\begin{pmatrix} -6 \\ 0 \\ 2 \end{pmatrix}}_{\mathbf{b}} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$



Toy linear system

3×3 equations

Message	Computation	t=0	t=1	t=2	t=3
P_{xy}	$-A_{xy}^2/(P_{xx}+P_{zx})$	0	−4	1/2	**1/2**
P_{yx}	$-A_{yx}^2/(P_{yy})$	0	−4	**−4**	**−4**
P_{xz}	$-A_{xz}^2/(P_{zz})$	0	−9	3	**3**
P_{zx}	$-A_{zx}^2/(P_{xx}+P_{yx})$	0	−9	**−9**	**−9**
μ_{xy}	$(P_{xx}\mu_{xx}+P_{zx}\mu_{zx})/A_{xy}$	0	3	6	**6**
μ_{yx}	$P_{yy}\mu_{yy}/A_{yx}$	0	**0**	**0**	**0**
μ_{xz}	$(P_{xx}\mu_{xx}+P_{yx}\mu_{yx})/A_{xz}$	0	−2	**−2**	**−2**
μ_{zx}	$P_{zz}\mu_{zz}/A_{zx}$	0	2/3	**2/3**	**2/3**



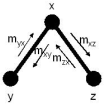
Toy linear system

3×3 equations

Solution	Computation
$\mu_x = x^*$	$\left(P_{xx}\mu_{xx} + P_{zx}\mu_{zx} + P_{yx}\mu_{yx}\right) / \left(P_{xx} + P_{zx} + P_{yx}\right) = \mathbf{1}$
$\mu_y = y^*$	$\left(P_{yy}\mu_{yy} + P_{xy}\mu_{xy}\right) / \left(P_{yy} + P_{xy}\right) = \mathbf{2}$
$\mu_z = z^*$	$\left(P_{zz}\mu_{zz} + P_{xz}\mu_{xz}\right) / \left(P_{zz} + P_{xz}\right) = -\mathbf{1}$

- Tree$\Rightarrow$

$$P_x^{-1} = (P_{xx} + P_{yx} + P_{zx})^{-1} = -1/12 = \{\mathbf{A}^{-1}\}_{xx}$$
$$P_y^{-1} = (P_{yy} + P_{xy})^{-1} = 2/3 = \{\mathbf{A}^{-1}\}_{yy}$$
$$P_z^{-1} = (P_{zz} + P_{xz})^{-1} = 1/4 = \{\mathbf{A}^{-1}\}_{zz}$$

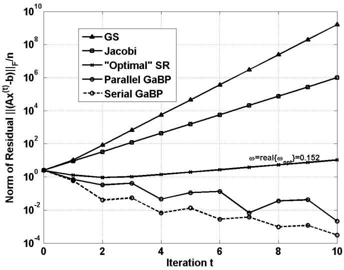


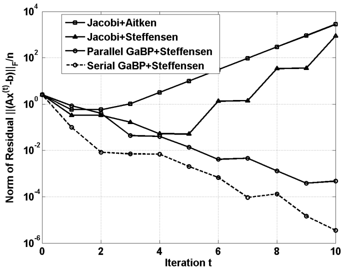
Symmetric, but not positive semi-definite, data matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

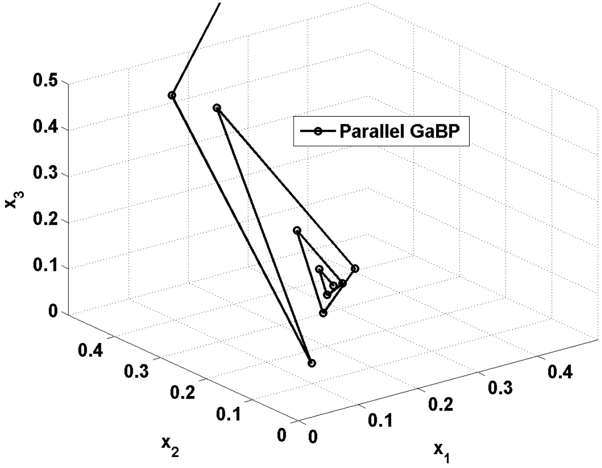
Algorithm	**Iterations** t
Jacobi,GS,SR,CG,Jacobi+Aitken,Jacobi+Steffensen	–
Parallel GaBP	**38**
Serial GaBP	**25**
Parallel GaBP+Steffensen	**21**
Serial GaBP+Steffensen	**14**

Symmetric, but not positive semi-definite, data matrix





Symmetric, but not positive semi-definite, data matrix



Jacobi method

Vector-wise

$$\mathbf{x}^{(t+1)} = \mathbf{D}^{-1}\big(\mathbf{b} - (\mathbf{L}+\mathbf{U})\mathbf{x}^{(t)}\big)$$

Element-wise

$$x_i^{(t+1)} = A_{ii}^{-1}\Big(b_i - \sum_{j\neq i} A_{ij}x_j^{(t)}\Big) \quad \forall i$$

Jacobi method

Vector-wise

$$\mathbf{x}^{(t+1)} = \mathbf{D}^{-1}\big(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}^{(t)}\big)$$

Convergence

- Sufficient condition: $\rho\big(\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\big) < 1$
  - holds, *e.g.* , if $\mathbf{A}$ is diagonally dominant, or
  - if $\mathbf{A}$, $\mathbf{D}$ and $\mathbf{D} - \mathbf{L} - \mathbf{U}$ are all positive definite.
- Necessary condition: diagonal terms in the matrix are greater (in magnitude) than other terms.

Jacobi Algorithm

- Given a system of linear equations of the form $Ax = b$, where A is invertible, we have a unique solution $x = A^{-1}b$.
- Looking at the i equation:

$$\sum_j a_{ij}x_j = b_i \tag{1}$$

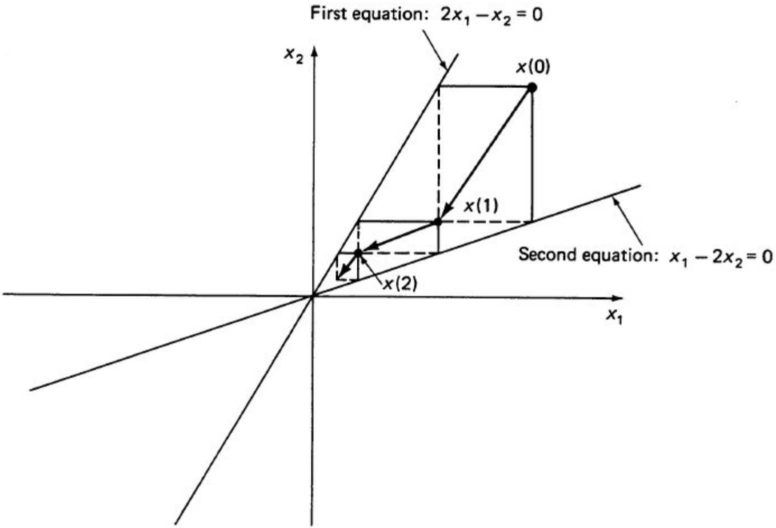
- Assuming $a_{ii} \neq 0$ we get:

$$x_i = \frac{(b_i - \sum_{j\neq i} a_{ij}x_j)}{a_{ii}} \tag{2}$$

- **The algorithm** Starting for an initial guess $x(0)$, compute for $i = 1, 2, \cdots$

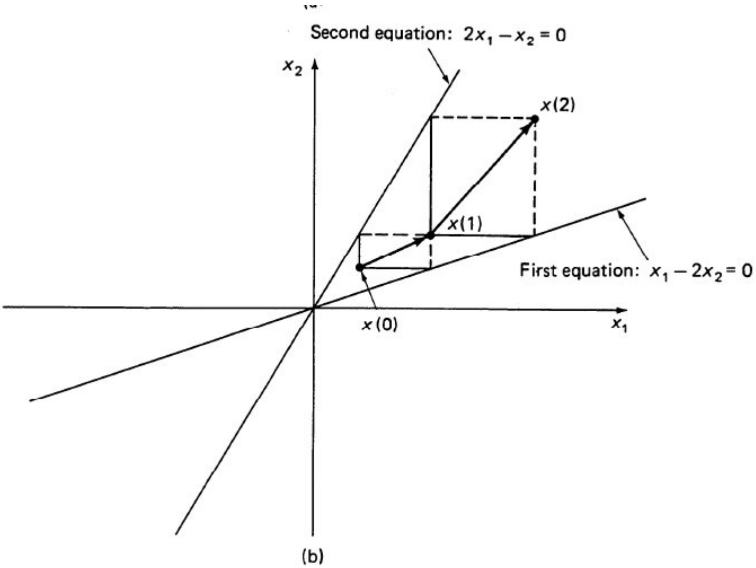
$$x_i^{(t)} = \frac{(b_i - \sum_{j\neq i} a_{ij}x_j^{(t-1)})}{a_{ii}} \tag{3}$$

Jacobi Convergence



(a)

Jacobi Divergence



Gauss-Seidel (GS) method

Vector-wise

$$\mathbf{x}^{(t+1)} = (\mathbf{D} + \mathbf{L})^{-1}(\mathbf{b} - \mathbf{U}\mathbf{x}^{(t)})$$

Element-wise

$$x_i^{(t+1)} = A_{ii}^{-1}\Big(b_i - \sum_{j<i} A_{ij}x_j^{(t+1)} - \sum_{j>i} A_{ij}x_j^{(t)}\Big) \quad \forall i$$

GS method as an instance of the GaBP solver

A ‘serial scheduling’ version of Jacobi method⇒Instance of the serial GaBP solver.

Gauss-Seidel (GS) method

Vector-wise

$$\mathbf{x}^{(t+1)} = (\mathbf{D}+\mathbf{L})^{-1}(\mathbf{b}-\mathbf{U}\mathbf{x}^{(t)})$$

Convergence

- Sufficient condition: $\rho\big((\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\big) < 1$
  - Holds, *e.g.*, if $\mathbf{A}$ is diagonally dominant, or
  - positive definite.
- Necessary condition: diagonal terms in the matrix are greater (in magnitude) than other terms.

Successive over-relaxation method

Vector-wise

$$\mathbf{x}^{(t+1)} = (\mathbf{D} + \omega\mathbf{L})^{-1}\Big(\omega\mathbf{b} - \big((1-\omega)\mathbf{D} - \omega\mathbf{U}\big)\mathbf{x}^{(t)}\Big)$$

Element-wise

$$x_i^{(t+1)} = (1-\omega)x_i^{(t)} + \omega A_{ii}^{-1}(b_i - \sum_{j<i} A_{ij}x_j^{(t+1)} - \sum_{j>i} A_{ij}x_j^{(t)}) \quad \forall i$$

SOR method as an instance of the GaBP solver

Gauss-Seidel method averaged over two consecutive iterations⇒Instance of the serial GaBP solver with damping.

Successive over-relaxation method

Vector-wise

$$\mathbf{x}^{(t+1)} = (\mathbf{D} + \omega\mathbf{L})^{-1}\Big(\omega\mathbf{b} - \big((1-\omega)\mathbf{D} - \omega\mathbf{U}\big)\mathbf{x}^{(t)}\Big)$$

Convergence

- Necessary condition: $\omega \in (0, 2)$
  - Successive relaxation (SR) for $\omega \in (0, 1)$
  - Successive over-relaxation (SOR) for $\omega \in (1, 2)$
- and sufficient for symmetric positive definite matrices.
- If $\rho\big((\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\big) < 1$, optimal convergence rate is given for

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho\big((\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\big)}}$$

Convergence acceleration: Aitken's method

- Consider a sequence $\{x_n\}$, obtained by using GaBP iterations, converging to the limit $\hat{x}$.
- According to Aitken's method, if there exists a real number a such that $|a| < 1$ and $\lim_{n\to\infty}(x_n - \hat{x})/(x_{n-1} - \hat{x}) = a$, then the sequence $\{y_n\}$ defined by

$$y_n = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}$$

  converges to $\hat{x}$ faster than $\{x_n\}$ in the sense that $\lim_{n\to\infty} |(\hat{x} - y_n)/(\hat{x} - x_n)| = 0$.
- A generalization of over-relaxation (3 consecutive iterations used rather than 2).

Convergence acceleration: Steffensen's iterations

- Encapsulate Aitken's method
- Starting with x_n, two iterations are run to get x_{n+1} and x_{n+2}. Next, Aitken's method is used to compute y_n, this value is replaced with the original x_n, and GaBP is executed again to get a new value of x_{n+1}. This process is repeated iteratively until convergence.