Distributed Kalman Filter via Gaussian Belief Propagation

Danny Bickson  
IBM Haifa Research Lab  
Mount Carmel, Haifa 31905, Israel  
Email: dannybi@il.ibm.com

Ori Shental  
Center for Magnetic Recording Research  
UCSD  
9500 Gilman Drive  
La Jolla, CA 92093, USA  
Email: oshental@ucsd.edu

Danny Dolev  
School of Computer Science and Engineering  
Hebrew University of Jerusalem  
Jerusalem 91904, Israel  
Email: dolev@cs.huji.ac.il

Abstract—Recent result shows how to compute distributively and efficiently the linear MMSE for the multiuser detection problem, using the Gaussian BP algorithm. In the current work, we extend this construction, and show that operating this algorithm twice on the matching inputs, has several interesting interpretations. First, we show equivalence to computing one iteration of the Kalman filter. Second, we show that the Kalman filter is a special case of the Gaussian information bottleneck algorithm, when the weight parameter \( \beta = 1 \). Third, we discuss the relation to the Affine-scaling interior-point method and show it is a special case of Kalman filter.

Besides of the theoretical interest of this linking estimation, compression/clustering and optimization, we allow a single distributed implementation of those algorithms, which is a highly practical and important task in sensor and mobile ad-hoc networks. Application to numerous problem domains includes collaborative signal processing and distributed allocation of resources in a communication network.

I. INTRODUCTION

Recent work [1] shows how to compute efficiently and distributively the MMSE prediction for the multiuser detection problem, using the Gaussian Belief Propagation (GaBP) algorithm. The basic idea is to shift the problem from linear algebra domain into a probabilistic graphical model, solving an equivalent inference problem using the efficient belief propagation inference engine. [2] compares the empirical performance of the GaBP algorithm relative to other linear iterative algorithms, demonstrating faster convergence. [3] elaborates on the relation to solving systems of linear equations.

In the present work, we propose to extend the previous construction, and show that by performing the MMSE computation twice on the matching inputs we are able to compute several algorithms. First, we reduce the discrete Kalman filter computation [4] to a matrix inversion problem and show how to solve it using the GaBP algorithm. We show that Kalman filter iteration which is composed from prediction and measurement steps can be computed by two consecutive MMSE predictions. Second, we explore the relation to Gaussian information bottleneck (GIB) [5] and show that Kalman filter is a special instance of the GIB algorithm, when the weight parameter \( \beta = 1 \). To the best of the authors knowledge, this is the first algorithmic link between the information bottleneck framework and linear dynamical systems. Third, we discuss the connection to the Affine-scaling interior-point method and show it is an instance of the Kalman filter.

Besides of the theoretical interest of linking compression, estimation and optimization together, our work is highly practical since it proposes a general framework for computing all of the above tasks distributively in a computer network. This result can have many applications in the fields of estimation, collaborative signal processing, distributed resource allocation etc.

A closely related work is [6]. In this work, Frey et al. focus on the belief propagation algorithm (a.k.a. sum-product algorithm) using factor graph topologies. They show that the Kalman filter algorithm can be computed using belief propagation over a
factor graph. In this contribution we extend their work in several directions. First, we extend the computation to vector variables (relative to scalar variables in Frey’s work). Second, we use a different graphical model: an undirected graphical model which results in simpler update rules, where Frey uses factor-graph with two types of messages: factor to variables and variables to factors. Third and most important, we allow an efficient distributed calculation of the Kalman filter steps, where Frey’s algorithm is centralized.

Another related work is [7]. In this work the link between Kalman filter and linear programming is established. In this work we propose a new and different construction which ties the two algorithms.

The structure of this paper is as follows. In Section II we describe the discrete Kalman filter. In Section III we outline the GIB algorithm and discuss its relation to the Kalman filter. Section IV presents the Affine-scaling interior-point method and compares it to the Kalman filter algorithm. Section V presents our novel construction for performing an efficient distributed computation of the three methods.

II. KALMAN FILTER

A. An Overview

The Kalman filter is an efficient iterative algorithm to estimate the state of a discrete-time controlled process $x \in \mathbb{R}^n$ that is governed by the linear stochastic difference equation [1]:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \quad (1)$$

with a measurement $z \in \mathbb{R}^m$ that is $z_k = Hx_k + v_k$. The random variables $w_k$ and $v_k$ that represent the process and measurement AWGN noise (respectively). $p(w) \sim \mathcal{N}(0, Q), p(v) \sim \mathcal{N}(0, R)$. We further assume that the matrices $A, H, B, Q, R$ are given [1].

The discrete Kalman filter update equations are given by [4]:

The prediction step:

$$\hat{x}_k^- = Ax_{k-1} + Bu_{k-1}, \quad (2a)$$
$$P_k^- = AP_p^{-1} A^T + Q. \quad (2b)$$

The measurement step:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}, \quad (3a)$$
$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-), \quad (3b)$$
$$P_k = (I - K_k H) P_k^- . \quad (3c)$$

where $I$ is the identity matrix.

The algorithm operates in rounds. In round $k$ the estimates $\hat{x}_k, \hat{x}_k, P_k$ are computed, incorporating the (noisy) measurement $z_k$ obtained in this round. The output of the algorithm are the mean vector $\hat{x}_k$ and the covariance matrix $P_k$.

B. New construction

Our novel contribution is a new efficient distributed algorithm for computing the Kalman filter. We begin by showing that the Kalman filter can be computed by inverting the following covariance matrix:

$$E = \begin{pmatrix} -P_k & A \\ A^T & Q & H \\ 0 & H^T & R \end{pmatrix}, \quad (4)$$

and taking the lower right $1 \times 1$ block to be $P_k$.

The computation of $E^{-1}$ can be done efficiently using recent advances in the field of Gaussian belief propagation [1], [3]. The intuition for our approach, is that the Kalman filter is composed of two steps. In the prediction step, given $x_k$, we compute the MMSE prediction of $x_k$ [6]. In the measurement step, we compute the MMSE prediction of $x_{k+1}$ given $x_k$, the output of the prediction step. Each MMSE computation can be done using the GaBP algorithm [1]. The basic idea, is that given the joint Gaussian distribution $p(x,y)$ with the covariance matrix $C = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}$, we can compute the MMSE prediction

$$\hat{y} = \arg \max_y p(y|x) \propto \mathcal{N}(\mu_{y|x}, \Sigma_{y|x}^{-1}),$$

where

$$\mu_{y|x} = (\Sigma_{yy} - \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy})^{-1} \Sigma_{yx} \Sigma_{xx}^{-1} x,$$
$$\Sigma_{y|x} = (\Sigma_{yy} - \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy})^{-1}.$$
This in turn is equivalent to computing the Schur complement of the lower right block of the matrix $C$. In total, computing the MMSE prediction in Gaussian graphical model boils down to a computation of a matrix inverse. In [1] we have shown that for the specific case of linear detection we can compute the MMSE estimator using the GaBP algorithm. Next, we show that performing two consecutive computations of the MMSE are equivalent to one iteration of the Kalman filter.

**Theorem 1:** The lower right $1 \times 1$ block of the matrix inverse $E^{-1}$ (eq. 4), computed by two MMSE iterations, is equivalent to the computation of $P_k$ done by one iteration of the Kalman filter algorithm.

Proof of Theorem 1 is given in Appendix A.

In Section V we explain how to utilize the above observation to an efficient distributed iterative algorithm for computing the Kalman filter.

**III. GAUSSIAN INFORMATION BOTTLENECK**

Given the joint distribution of a source variable $X$ and another relevance variable $Y$, Information bottleneck (IB) operates to compress $X$, while preserving information about $Y$ [8], [9], using the following variational problem:

$$
\min_{p(t|x)} \mathcal{L} : \mathcal{L} \equiv I(X; T) - \beta I(T; Y)
$$

$T$ represents the compressed representation of $X$ via the conditional distributions $p(t|x)$, while the information that $T$ maintains on $Y$ is captured by the distribution $p(y|t)$. $\beta > 0$ is a Lagrange multiplier which weights the tradeoff between minimizing the compression information and maximizing the relevant information. As $\beta \to 0$ we are interested solely in compression, but all relevant information about $Y$ is lost $I(Y; T) = 0$. When $\beta \to \infty$ where are focused on preservation of relevant information, in this case $T$ is simply the distribution $X$ and we obtain $I(T; Y) = I(X; Y)$. The interesting cases are in between, when for finite values of $\beta$ we are able to extract rather compressed representation of $X$ while still maintaining a significant fraction of the original information about $Y$.

An iterative algorithm for solving the IB problem is given in [9]:

$$
P^{k+1}(t|x) = \frac{P^k(t)}{Z^{k+1}(x, \beta)} \cdot \exp(-\beta D_{KL}[p(y|x) \| p^k(y|t)]),
$$

$$
P^k(t) = \int_x p(x) P^k(t|x) dx,
$$

$$
P^k(y|t) = \frac{1}{P^k(t)} \int_x P^k(t|x)p(x, y) dx.
$$

where $Z^{k+1}$ is a normalization factor computed in round $k + 1$.

The Gaussian information bottleneck (GIB) [5] deals with the special case where the underlying distributions are Gaussian. In this case, the computed distribution $p(t)$ is Gaussian as well, represented by a linear transformation $T_k = A_k X + \xi_k$ where $A_k$ is a joint covariance matrix of $X$ and $T$, $\xi_k \sim N(0, \Sigma_{\xi_k})$ is a multivariate Gaussian independent of $X$. The outputs of the algorithm are the covariance matrices representing the linear transformation $T$: $A_k, \Sigma_{\xi_k}$.

An iterative algorithm is derived by substituting Gaussian distributions into (5), resulting in the following update rules:

$$
\Sigma_{\xi_{k+1}} = (\beta \Sigma_{t_k|y} - (\beta - 1) \Sigma_{t_k}^{-1}),
$$

$$
A_{k+1} = \beta \Sigma_{\xi_k + 1} \Sigma_{t_k|y}^{-1} A_k (I - \Sigma_{y|t} \Sigma_{x|y}^{-1}).
$$

**TABLE I**

**Summary of notations in the GIB [5] paper vs. Kalman filter [4]**

| GIB [5]         | Kalman [4] | Kalman meaning                                |
|-----------------|------------|-----------------------------------------------|
| $\Sigma_x$      | $P_0$      | a-priori estimate error covariance            |
| $\Sigma_y$      | $Q$        | Process AWGN noise                            |
| $\Sigma_{t_k}$  | $R$        | Measurement AWGN noise                        |
| $\Sigma_{y|t}$  | $A_t A_t^T$| process state transformation matrix            |
| $\Sigma_{x|y}$  | $H^T, H$   | measurement transformation matrix              |
| $\Sigma_{\xi_k}$| $P_k$      | posterior error covariance in round $k$       |
| $\Sigma_{x|y}$  | $P_k$      | a-priori error covariance in round $k$        |

Since the underlying graphical model of both algorithms (GIB and Kalman filter) is Markovian with Gaussian probabilities, it is interesting to ask what is the relation between them. In this work we show, that the Kalman filter posterior error covariance computation is a special case of the GIB algorithm when $\beta = 1$. Furthermore, we show
how to compute GIB using the Kalman filter when $\beta > 1$ (the case where $0 < \beta < 1$ is not interesting since it gives a degenerate solution where $A_k \equiv 0$ [5].) Table I outlines the different notations used by both algorithms.

Theorem 2: The GIB algorithm when $\beta = 1$ is equivalent to the Kalman filter algorithm. The proof is given in Appendix B.

Theorem 3: The GIB algorithm when $\beta > 1$ can be computed by a modified Kalman filter iteration. The proof is given in Appendix C.

There are some differences between the GIB algorithm and Kalman filter computation. First, the Kalman filter has input observations $z_k$ in each round. Note that the observations do not affect the posterior error covariance computation $P_k$ (eq. 3c), but affect the posterior mean $\hat{x}_k$ (eq. 3b). Second, Kalman filter computes both posterior mean $\hat{x}_k$ and error covariance $P_k$. The covariance $\Sigma_{\hat{x}_k}$ computed by the GIB algorithm was shown to be identical to $P_k$ when $\beta = 1$. The GIB algorithm does not compute the posterior mean, but computes an additional covariance $A_k$ (eq. 6b), which is assumed known in the Kalman filter.

From the information theoretic perspective, our work extends the ideas presented in [10]. Predictive information is defined to be the mutual information between the past and the future of a time series. In that sense, by using Theorem 2, Kalman filter can be thought of as a prediction of the future, which from the one hand compresses the information about past, and from the other hand maintains information about the present.
IV. RELATION TO THE AFFINE-SCALING ALGORITHM

One of the most efficient interior point methods used for linear programming is the Affine-scaling algorithm [11]. It is known that the Kalman filter is linked to the Affine-scaling algorithm [7]. In this work we give an alternate proof, based on different construction, which shows that Affine-scaling is an instance of Kalman filter, which is an instance of GIB. This link between estimation and optimization allows for numerous applications. Furthermore, by providing a single distribute efficient implementation of the GIB algorithm, we are able to solve numerous problems in communication networks.

The linear programming problem in its canonical form is given by:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0.
\end{align*}
\]

where \( A \in \mathbb{R}^{n \times p} \) with \( \text{rank}\{A\} = p < n \). We assume the problem is solvable with an optimal point that minimizes the cost function (7a):

\[ x_1 = x_0 - \frac{\alpha}{\gamma} D^2 r \]

where \( 0 < \alpha < 1 \) is the step size, \( r \) is the step direction.

\[
\begin{align*}
\gamma &= \max_i (\epsilon_i P D^2 c), \\
w &= (AD^2 A^T)^{-1} AD^2 c, \\
r &= (c - A^T w),
\end{align*}
\]

where \( \epsilon_i \) is the \( i \)th unit vector and \( P \) is a projection matrix given by:

\[ P = I - DA^T (AD^2 A^T)^{-1} AD. \]

The algorithm continues in rounds and is guaranteed to find an optimal solution in at most \( n \) rounds. In a nutshell, in each iteration, the Affine-scaling algorithm first performs an Affine-scaling with respect to the current solution point \( x_i \) and obtains the direction of descent by projecting the gradient of the transformed cost function on the null space of the constraints set. The new solution is obtained by translating the current solution along the direction found and then mapping the result back into the original space [7]. This has interesting analogy for the two phases of the Kalman filter.

Theorem 4: The Affine-scaling algorithm iteration is an instance of the Kalman filter algorithm iteration.

Proof is given in Appendix D.

V. EFFICIENT DISTRIBUTED COMPUTATION

We have shown how to express the Kalman filter, Gaussian information bottleneck and Affine-scaling algorithms as a two step MMSE computation. Each step involves inverting a \( 2 \times 2 \) block matrix. Recent result by Bickson and Shental et al. [1] show that the MMSE computation can be done efficiently and distributively using the Gaussian belief propagation algorithm. Because of space limitations the full algorithm is not reproduced here.

The interested reader is referred to [1], [3] for a complete derivation of the GaBP update rules and convergence analysis. The GaBP algorithm is summarized in Table [II]

Regarding convergence, if it converges, GaBP is known to result in exact inference [12]. Determining the exact region of convergence and convergence rate remain open research problems. All that is known is a sufficient (but not necessary) condition [13], [14] stating that GaBP converges when the spectral radius satisfies \( \rho(|I_K - A|) < 1 \), where \( A \) is first normalized s.t. the main diagonal contains ones. A stricter sufficient condition [12], determines that the matrix \( A \) must be diagonally dominant (i.e., \( |A_{ii}| > \sum_{j \neq i} |A_{ij}|, \forall i \)) in order for GaBP to converge.

Regarding convergence speed, [15] shows that when converging, the algorithm converges in \( O(\log(\epsilon)/\log(\gamma)) \) iterations, where \( \epsilon \) is the desired accuracy, and \( 1/2 < \gamma < 1 \) is a parameter related to the inverted matrix. The computation overhead in each iteration is determined by the number of non-zero elements of the inverted matrix \( A \). In practice, [16] demonstrates convergence of 5-10 rounds on sparse matrices with several millions of variables. [17] shows convergence of dense constraint matrices
\[ x = A^{-1} b \] via GABP [3].

| # | Stage       | Operation                                                                 |
|---|-------------|---------------------------------------------------------------------------|
| 1 | Initialize  | Compute \( P_{ii} = A_{ii} \) and \( \mu_{ii} = b_i / A_{ii} \).          |
|   |             | Set \( P_{ki} = 0 \) and \( \mu_{ki} = 0 \), \( \forall k \neq i \).      |
| 2 | Iterate     | Propagate \( P_{ki} \) and \( \mu_{ki} \), \( \forall k \neq i \) such that \( A_{ki} \neq 0 \). |
|   |             | Compute \( P_{i\backslash j} = P_{ii} + \sum_{k \in N(i) \backslash j} P_{ki} \) and \( \mu_{i\backslash j} = P_{i\backslash j}^{-1}(P_{ii}\mu_{ii} + \sum_{k \in N(i) \backslash j} P_{ki}\mu_{ki}) \). |
|   |             | Compute \( P_{ij} = -A_{ij} P_{i\backslash j} A_{ji} \) and \( \mu_{ij} = -P_{ij}^{-1} A_{ij} \mu_{i\backslash j} \). |
| 3 | Check       | If \( P_{ij} \) and \( \mu_{ij} \) did not converge, return to \#2. Else, continue to \#4. |
| 4 | Infer       | \( P_i = P_{ii} + \sum_{k \in N(i)} P_{ki} \), \( \mu_i = P_{i\backslash i}^{-1}(P_{ii}\mu_{ii} + \sum_{k \in N(i)} P_{ki}\mu_{ki}) \). |
| 5 | Output      | \( x_i = \mu_i \) of size up to \( 150,000 \times 150,000 \) in 6 rounds, where the algorithm is run in parallel using 1,024 CPUs. Empirical comparison with other iterative algorithms is given in [2]. |

### VI. Example Application

The TransFab software package is a distributed middleware developed in IBM Haifa Labs, which supports real-time forwarding of message streams, providing quality of service guarantees. We plan to use our distributed Kalman filter algorithm for online monitoring of software resources and performance. On each second, each node records a vector of performance parameters like memory usage, CPU usage, current bandwidth, queue sizes etc. The nodes execute the distributed Kalman filter algorithm on the background. Figure 3 plots a covariance matrix of running an experiment using two TransFab nodes propagating data. The covariance matrix is used as an input to the Kalman filter algorithm. Yellow sections show high correlation between measured parameters. Initial results are encouraging; we plan to report them using a future contribution.

### VII. Conclusion

In this work we have linked together several different algorithms from the fields of estimation (Kalman filter), clustering/compression (Gaussian information bottleneck) and optimization (Affine-scaling interior-point method). Besides of the theoretical interest in linking those different domains, we are motivated by practical problems in communication networks. To this end, we propose an efficient distributed iterative algorithm, the Gaussian belief propagation algorithm, to be used for efficiently solving these problems.

**Acknowledgment**

O. Shental acknowledges the partial support of the NSF (Grant CCF-0514859). The authors are grateful to Noam Slonim and Naftali Tishby from the Hebrew University of Jerusalem for useful discussions.

### References

[1] D. Bickson, O. Shental, P. H. Siegel, J. K. Wolf, and D. Dolev, “Gaussian belief propagation based multiuser detection,” in *IEEE Int. Symp. on Inform. Theory (ISIT)*, Toronto, Canada, July 2008.

[2] ——, “Linear detection via belief propagation,” in *Proc. 45th Allerton Conf. on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2007.

[3] O. Shental, D. Bickson, P. H. Siegel, J. K. Wolf, and D. Dolev, “Gaussian belief propagation solver for systems of linear equations,” in *IEEE Int. Symp. on Inform. Theory (ISIT)*, Toronto, Canada, July 2008.
applied to the $2 \times 2$ upper left submatrix of $E$, where $D \triangleq Q, C \triangleq A^T, B \triangleq A, A \triangleq P_{k-1}$ we get:

$$P_k^{-} = \frac{D}{Q} - \frac{C}{A^T} \frac{-A^{-1} B}{P_{k-1}} A.$$

Now we compute recursively the Schur complement of lower right $2 \times 2$ submatrix of the matrix $E$ using the matrix inversion lemma:

$$A^{-1} + A^{-1} B (D - CA^{-1} B)^{-1} CA^{-1}$$

where $A^{-1} \triangleq P_k^{-}, B \triangleq H^T, C \triangleq H, D \triangleq Q$. In total we get:

$$P_k^{-} = \frac{A^{-1} - P_k^{-} H^T (H P_k^{-} H^T + R)^{-1} H}{(I - K_k H) P_k^{-} = P_k}$$

**APPENDIX B**

**Proof:** Looking at [5, §39], when $\beta = 1$ we get

$$\Sigma_{\xi+1} = (\Sigma_{ty|x}^{-1})^{-1} = \Sigma_{ty|x} = \text{MMSE}$$

$$\Sigma_{ty|x} = \Sigma_{ty|x}^{-1} \Sigma_{ty|x} = \Sigma_{ty|x} = \text{MMSE}$$

$$\Sigma_{ty|x} = \Sigma_{ty|x}^{-1} \Sigma_{ty|x} = \Sigma_{ty|x} = \text{MMSE}$$

$$\Sigma_{ty|x} = \Sigma_{ty|x}^{-1} \Sigma_{ty|x} = \Sigma_{ty|x} = \text{MMSE}$$

$$\Sigma_{ty|x} = \Sigma_{ty|x}^{-1} \Sigma_{ty|x} = \Sigma_{ty|x} = \text{MMSE}$$

$$\Sigma_{ty|x} = \Sigma_{ty|x}^{-1} \Sigma_{ty|x} = \Sigma_{ty|x} = \text{MMSE}$$

**APPENDIX A**

**Proof:** We prove that inverting the matrix $E$ (eq. [4]) is equivalent to one iteration of the Kalman filter for computing $P_k$.

We start from the matrix $E$ and show that $P_k^{-}$ can be computed in recursion using the Schur complement formula:

$$D - CA^{-1} B$$  \hspace{1cm} (11)
Now we show this formulation is equivalent to the Kalman filter with the following notations:

\[ P_k^- \triangleq (A^T \Sigma_x A + \Sigma_\xi), \quad H \triangleq A^T \Sigma_y, \quad R \triangleq \Sigma_y, \]

\[ P_{k-1} \triangleq \Sigma_x, \quad Q \triangleq \Sigma_\xi. \]

Substituting we get:

\[
\begin{align*}
\frac{P_k^-}{(A^T \Sigma_x A + \Sigma_\xi)} + \frac{P_k^-}{(A^T \Sigma_x A + \Sigma_\xi)} A^T \Sigma_{xy}, \\
R \quad \frac{H^T}{H} \quad \frac{P_k^-}{P_k^-} \quad \frac{H^T}{H} \quad \frac{\Sigma_y}{\Sigma_y} + A^T \Sigma_{yx} (A^T \Sigma_x A + \Sigma_\xi) \Sigma_{xy} A, \\
\end{align*}
\]

Which is equivalent to (12). Now we can apply Theorem 1 and get the desired result.

APPENDIX C

Proof: In the case where \( \beta > 1 \), the MAP covariance matrix as computed by the GIB algorithm is:

\[ \Sigma_{\xi_{k+1}} = \beta \Sigma_{\xi_k} + (1 - \beta) \Sigma_{t_k} \quad (13) \]

This is a weighted average of two covariance matrices. \( \Sigma_{t_k} \) is computed at the first phase of the algorithm (equivalent to the prediction phase in Kalman literature), and \( \Sigma_{\xi_k} \) is computed in the second phase of the algorithm (measurement phase). At the end of the Kalman iteration, we simply compute the weighted average of the two matrices to get (13). Finally, we compute \( A_{k+1} \) using (eq. 6b) by substituting the modified \( \Sigma_{\xi_{k+1}} \).

APPENDIX D

Proof: We start by expanding the Affine-scaling update rule: