中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-07-013

Global and Componentwise Extrapolations for Accelerating Training of Bayesian Networks and Conditional Random Fields

Chun-Nan Hsu, Han-Shen Huang, Bo-Hou Yang, and Yu-Ming Chang

Oct. 8, 2007 || Technical Report No. TR-IIS-07-013

http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2007/tr07.html

Global and Componentwise Extrapolations for Accelerating Training of Bayesian Networks and Conditional Random Fields

Chun-Nan Hsu CHUNNAN@IIS.SINICA.EDU.TW
Institute of Information Science, Academia Sinica, Taipei, Taiwan

Han-Shen Huang HANSHEN@IIS.SINICA.EDU.TW
Institute of Information Science, Academia Sinica, Taipei, Taiwan

Bo-Hou Yang ERICYANG@IIS.SINICA.EDU.TW
Department of Electrical Engineering, Chang Gung University, Taoyuan, Taiwan
Institute of Information Science, Academia Sinica, Taipei, Taiwan

Yu-Ming Chang PORTER@IIS.SINICA.EDU.TW
Institute of Information Science, Academia Sinica, Taipei, Taiwan

Editor:

Abstract

The triple jump extrapolation method is an effective approximation of Aitken's acceleration that can accelerate the convergence of many parameter learning algorithms, including EM and generalized iterative scaling. It has two options — global and componentwise extrapolation. Empirical studies showed that neither can dominate the other in all cases and it is not known which one is better under what condition. In this paper, we investigate this problem and conclude that, when the Jacobian is (block) diagonal, componentwise extrapolation will be more effective. We derive two hints that allow us to determine the block diagonality. The first hint is that when we have a highly sparse data set, the Jacobian of the EM mapping for training a Bayesian network will be block diagonal. The second hint is that the block diagonality of the Jacobian of the GIS mapping for training CRF is negatively correlated with the strength of feature dependencies. We empirically verify these hints with controlled and real-world data sets. The results show that our hints can accurately predict which method will be superior. The results also show that both global and componentwise extrapolation can provide substantial acceleration. In particular, when applied to train large-scale CRF models, the GIS variant accelerated by componentwise extrapolation not only outperforms its global extrapolation counterpart, as our hint predicts, but can also compete with limited-memory BFGS (L-BFGS), the de facto standard for CRF training, in terms of both computational efficiency and F-scores.

Keywords: Bayesian Networks, Conditional Random Fields, Expectation-Maximization (EM) Algorithm, Generalized Iterative Scaling, Triple-Jump Extrapolation

1. Introduction

Aitken's acceleration is one of the most commonly used method to speed up fixed-point iteration methods (Burden and Faires, 1988). Since many machine learning and pattern recognition algorithms can be considered as a fixed-point iteration method, we can apply Aitken's acceleration to accelerate these algorithms. For example, the expectation-maximization (EM) algorithm (Dempster et al., 1977) can be considered as a fixed-point iteration method. We can apply Aitken's acceleration to speed up its convergence (Louis, 1982; McLachlan and Krishnan, 1997). Another example

is the generalized iterative scaling (GIS) algorithm (Darroch and Ratcliff, 1972), which is a classical method to train exponential probabilistic models. However, GIS usually converges slowly, especially when applied to train a conditional random field model (CRF) (Lafferty et al., 2001) for large-scale entity recognition tasks, where a CRF model may contain millions of parameters to be estimated from tens thousands of sentences. Again, since GIS can also be considered as fixed-point iteration, we can apply Aitken's acceleration to speed up its convergence. In fact, all *bound optimization methods* (Salakhutdinov et al., 2003; Salakhutdinov and Roweis, 2003), including EM, GIS, non-negative matrix factorization (NMF) and concave-convex procedure (CCCP), can be accelerated by Aitken's acceleration.

Basically, the idea of Aitken's acceleration is to extrapolate according to previous and current estimation of the parameters to be learned. However, the multivariate version of Aitken's acceleration requires to compute or approximate the Jacobian of the mapping matrix of the fixed-point iteration, which may not have a closed form and can be intractable even for a very simple model. Many variants of Aitken's acceleration have been proposed to approximate the Jacobian. One of the methods is the triple jump extrapolation method for the EM algorithm (Huang et al., 2005; Hesterberg, 2005; Schafer, 1997). The idea is to estimate the extrapolation rate by considering the previous two consecutive estimates of the parameter vectors. The triple jump extrapolation method can effectively accelerate the EM algorithm by substantially reducing the number of iterations required for the EM algorithm to converge. Another benefit of the triple jump method is that it can be easily integrated with existing EM packages for any probabilistic model. Though the triple jump method, as all variants of Aitken's acceleration, may not monotonically increase the likelihood, we can apply the idea proposed by Salakhutdinov and Roweis (2003) to resolve the issue. The idea is to discard the extrapolation if it fails to improve the likelihood and use the estimate obtained without the extrapolation. In this way, convergence can be guaranteed (Huang et al., 2005).

The triple jump method can extrapolate the parameter vector with a fixed scalar extrapolation rate for all dimensions or a vector of extrapolation rates for different dimensions. We refer to the former approach as *global extrapolation* and the latter as *componentwise extrapolation*. We tried them for a variety of probabilistic models and found that in general, global extrapolation yields a better performance for the EM algorithm, but there are cases where componentwise extrapolation yields very high speedup (Huang et al., 2005). When applied to accelerate CRF training, both method can be many orders of magnitude faster than GIS, with componentwise extrapolation further outperforming global extrapolation by many folds. But for real-world applications that involve millions of parameters, componentwise extrapolation requires millions of bytes more of memory space to store extrapolation rates. Therefore, though it is possible to empirically determine which method to apply, it is important to understand when and why componentwise extrapolation is superior and necessary in advance.

In this paper, we investigate when componentwise extrapolation should be preferred and when should not. We conclude that, when the Jacobian of the fixed-point iteration mapping is (block) diagonal, componentwise extrapolation should be preferred. Otherwise, global extrapolation should be applied. Previously, Schafer (1997, chap. 3) suggested that for the EM algorithm, when the global and componentwise rates of convergence are different, componentwise extrapolation should be preferred. However, he did not formally justify this claim and how to determine when the rates may be different. In fact, we can show that when the mapping is (block) diagonal, the rates will be different and our conclusion follows.

However, it is difficult to determine the block diagonality of the Jacobian of a given model. In this paper, we identify analytical conditions when a Jacobian will be (block) diagonal. The analytical conditions imply hints for us to easily determine which extrapolation method should be applied without actually computing their Jacobians. We identify two such hints and verify them empirically with controlled and real-world data sets. The first hint is that when we have a highly sparse data set, the Jacobian of the EM mapping for training a Bayesian network will be more likely to be block diagonal and therefore, componentwise extrapolation will be more effective. Otherwise, global extrapolation should be used. The second hint is that the block diagonality of the Jacobian of the GIS mapping for training CRF is negatively correlated with the strength of feature dependencies. To verify the second hint and demonstrate the usefulness of our findings, we investigate the entity recognition problem in natural language processing, an application of CRF which usually involves millions of parameters with tens thousands of training examples. Applying GIS is impractical here because it is prohibitively slow for large data sets (Malouf, 2002; Sha and Pereira, 2003). Our hint predicts that componentwise extrapolation will be more effective than global extrapolation. Experimental results over three large-scale real-world data sets confirm our prediction, showing that componentwise extrapolation can converge about four times as fast as global extrapolation. The results also show that both extrapolation methods can accelerate GIS drastically, reducing the convergence time to a practical level. Moreover, the results also show that componentwise extrapolation can compete with limited-memory BFGS (L-BFGS) (Nocedal and Wright, 1999), the de facto standard algorithm for CRF training, in terms of both computational efficiency and quality of trained models.

In the remainder of this paper, we first provide a review of a formal derivation of the triple jump extrapolation method for global extrapolation. Next, Section 3 compares the convergence properties of global and componentwise extrapolations and presents the main result of this paper. Then, we verify our result with a variety of probabilistic models. In Section 4, we consider the models trained by the EM algorithm, including mixtures of Gaussians, Bayesian networks and Semi-supervised Bayesian classifiers. Then in Section 5, we consider the problem of accelerating the GIS algorithm for training CRFs. Finally, we summarize our conclusions in the last section.

2. Triple Jump Extrapolation

In this section, we review Aitken's acceleration for fixed-point iteration algorithms and present the derivation of the triple jump extrapolation method.

2.1 Aitken's Acceleration

We consider the parameter learning problem as solving an equation by fixed-point iteration. Let θ be a l-dimensional parameter vector of a probabilistic model in the space Ω. A parameter estimation problem is usually solved by a *bound optimization method* (Salakhutdinov and Roweis, 2003), which in turn solves the equation $\theta^{(t+1)} = M(\theta^{(t)})$, where M is a mapping $M : \Omega \to \Omega$ defined by the bound optimization method and $\theta^{(t)} \in \Omega$ is the result of the t-th iteration, $t = 0, 1, 2, \ldots$. Starting from $\theta^{(t)}$ at iteration t, the fixed-point iteration method solves the equation $\theta = M(\theta)$ by iteratively substituting the input of a function M with the output of M in the previous iteration:

$$\theta^{(t+1)} = M(\theta^{(t)}), \;\; \theta^{(t+2)} = M(\theta^{(t+1)}), \;\; \ldots.$$

If M is continuous and $\theta^{(t)}$ converges to a local optimum θ^*, then $\theta^* = M(\theta^*)$.

For example, in the EM algorithm, M is a mapping that combines the E-step and the M-step such that after each iteration, the data likelihood given $\theta^{(t+1)}$ will be improved from the data likelihood given $\theta^{(t)}$. The EM algorithm looks for a parameter vector that satisfies $\theta = M(\theta)$ by iteratively substituting θ on the RHS with that on the LHS until convergence. Therefore, a bound optimization method such as the EM algorithm is equivalent to solving θ^* by the *fixed-point iteration method* (Burden and Faires, 1988).

Suppose that we apply the fixed-point iteration method from $\theta^{(t)}$ in the neighborhood of θ^* and the iteration converges at θ^*. Assuming that the mapping M is differentiable. Then we can apply a linear Taylor expansion of M around θ^* so that

$$\theta^{(t+1)} = M(\theta^{(t)}) \approx \theta^* + M'(\theta^*)(\theta^{(t)} - \theta^*) = \theta^* + J(\theta^{(t)} - \theta^*), \tag{1}$$

where J abbreviates $M'(\theta^*)$, the Jacobian of the mapping M at θ^*. We can apply M to $\theta^{(t)}$ consecutively for h times to obtain $\theta^{(t+h)}$. From Eq. (1), we have

$$\theta^{(t+h)} \approx \theta^* + J^h(\theta^{(t)} - \theta^*). \tag{2}$$

The eigen decomposition of the Jacobian J at θ^* is

$$J = Q \begin{pmatrix} \lambda_1 & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \lambda_n \end{pmatrix} Q^{-1} = Q\Lambda Q^{-1}, \tag{3}$$

where $Q = [v_1, \dots, v_n]$ contains the eigenvectors corresponding to eigenvalues $\lambda_1, \dots, \lambda_n$, respectively. Then, J^h in Eq. (2) becomes:

$$J^h = Q \begin{pmatrix} \lambda_1^h & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \lambda_n^h \end{pmatrix} Q^{-1} = Q\Lambda^h Q^{-1}.$$

Since $\theta^{(t+h)} \to \theta^*$ when $h \to \infty$ in bound optimization methods, it is required that $\lim_{h\to\infty} J^h = 0$ to ensure convergence. It follows that $\lim_{h\to\infty} \lambda_i^h = 0$, and thus, $-1 < \lambda_i < 1$ for all i.

Within the neighborhood of θ^*, the rate of convergence of M is determined by the largest eigenvalue of J, which is the slowest one among all eigenvalues to converge to zero. More generally, the rate is determined by the spectral radius ρ of J when the eigenvalues can be negative. The spectral radius ρ is defined by $\max\{|\lambda_{max}|, |\lambda_{min}|\}$, where λ_{max} and λ_{min} are the greatest and least eigenvalues of J. Since monotonicity and convergence can be proved for bound optimization methods, in previous works, the following assumption on the eigenvalues of J is usually expected to be true for bound optimization methods, including both EM and GIS algorithms. This assumption implies that $\rho = \lambda_{max}$.

Assumption 1 *The eigenvalues of the Jacobian of the mapping M lie in $[0, 1)$ (Dempster et al., 1977; McLachlan and Krishnan, 1997; Salakhutdinov et al., 2003).*

Aitken's acceleration is a classical method for accelerating fixed-point iteration. The multivariate version of Aitken's acceleration can be derived as follows (McLachlan and Krishnan, 1997). Suppose that when $t \to \infty$, $\theta^t \to \theta^*$. Then we can express θ^* as

$$\theta^* = \theta^{(t)} + \sum_{h=1}^{\infty} (\theta^{(t+h)} - \theta^{(t+h-1)}). \tag{4}$$

Suppose that $\theta^{(t)}$ is in the neighborhood of θ^*. Based on Eq. (2), $\theta^{(t+h+1)} - \theta^{(t+h)}$ can be written as:

$$\theta^{(t+h+1)} - \theta^{(t+h)} \approx \left[\theta^* + J^h(\theta^{(t+1)} - \theta^*)\right] - \left[\theta^* + J^h(\theta^{(t)} - \theta^*)\right] = J^h(\theta^{(t+1)} - \theta^{(t)}). \quad (5)$$

Applying Eq. (5) in Eq. (4) gives the multivariate Aitken's acceleration:

$$\begin{aligned} \theta^* &\approx \theta^{(t)} + \sum_{h=0}^{\infty} J^h(\theta^{(t+1)} - \theta^{(t)}) \\ &= \theta^{(t)} + (I - J)^{-1}(\theta_M^{(t)} - \theta^{(t)}), \end{aligned} \quad (6)$$

since the eigenvalues of J are between 0 and 1, from Assumption 1. In Eq. (6), we replace $\theta^{(t+1)}$ with $\theta_M^{(t)}$ to emphasize that $\theta^{(t+1)}$ is obtained by applying the mapping $M(\theta^{(t)})$ here.

In fact, Aitken's acceleration can be considered as performing extrapolations to accelerate fixed-point iteration. Aitken's acceleration extrapolates with a different rate along the direction of each eigenvalue for each dimension of the parameter vector.

One of the drawbacks of Aitken's acceleration is that it requires to compute the Jacobian of the fixed-point iteration mapping, which may not have a closed form and can be intractable even for a very simple model with a low dimensional parameter space. Other drawbacks include that it may not always converge and that it may be numerically unstable (Jamshidian and Jennrich, 1997).

2.2 The Triple Jump Extrapolation Method

The triple jump extrapolation method is an approximation of Aitken's acceleration. Instead of using the Jacobian matrix, the triple jump extrapolation method estimates the largest eigenvalue of the Jacobian matrix according to previous estimates of the parameter vectors in the extrapolation to alleviate the drawbacks of Aitken's acceleration.

We start from Eq. (6) of Aitken's acceleration. We substitute J with Eq. (3), and then $(I - J)^{-1}$ in Eq. (6) becomes:

$$\begin{aligned} (I - J)^{-1} &= \left[Q\left[I - \Lambda\right]Q^{-1}\right]^{-1} \\ &= Q[I - \Lambda]^{-1}Q^{-1} \\ &= Q \begin{pmatrix} \frac{1}{1-\lambda_1} & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \frac{1}{1-\lambda_n} \end{pmatrix} Q^{-1}. \end{aligned} \quad (7)$$

With the eigen decomposition of J, we can map θ^* in Eq. (6) from the original parameter space to the eigenspace spanned by Q:

$$\begin{aligned} \psi^* = Q^{-1}\theta^* &\approx Q^{-1}\theta^{(t)} + Q^{-1}(I - J)^{-1}(\theta_M^{(t)} - \theta^{(t)}) \\ &= Q^{-1}\theta^{(t)} + \left[I - \Lambda\right]^{-1} Q^{-1}(\theta_M^{(t)} - \theta^{(t)}) \\ &= \psi^{(t)} + \left[I - \Lambda\right]^{-1}(\psi_M^{(t)} - \psi^{(t)}). \end{aligned}$$

The relation between ψ^* and θ^* can also be written as:

$$\theta^* = \psi_1^* v_1 + \cdots + \psi_n^* v_n,$$

where ψ_i^* with $i = 1, \ldots, n$ denotes the i-th transformed parameter vector ψ^* in the eigenspace. Along with Eq. (7), we can observe that the multivariate Aitken's acceleration is in fact a series of univariate Aitken's acceleration along the direction of u_i:

$$\psi_i^* \approx \psi_i^{(t)} + \frac{1}{1-\lambda_i}(\psi_{Mi}^{(t)} - \psi_i^{(t)}). \tag{8}$$

Let $\varphi^{(t)} = \theta^{(t)} - \theta^*$ denote the difference between current estimated parameter vector to the local maximum. The global rate of convergence of a fixed-point iteration method is defined as the ratio:

$$R = \lim_{t\to\infty} R^{(t)} \equiv \lim_{t\to\infty} \frac{\|\varphi^{(t+1)}\|}{\|\varphi^{(t)}\|} \tag{9}$$

Dempster et al. (1977) have shown that $R = \lambda_{max}$, the largest eigenvalue of J for the EM algorithm. Their result can be generalized to bound optimization methods. Therefore, instead of computing the Jacobian, we can simplify Aitken's acceleration by replacing every eigenvalue λ_i with a single value $\gamma^{(t)}$ such that $\gamma^{(t)}$ is an approximation of λ_{max} at the t-th iteration:

$$\theta^{(t+1)} \quad = \quad \theta^{(t)} + (1-\gamma^{(t)})^{-1}(\theta_M^{(t)} - \theta^{(t)}). \tag{10}$$

Note that $(1-\gamma^{(t)})^{-1}$ in Eq. (10) can be written as $Q\,\mathrm{diag}(1-\gamma^{(t)})^{-1}Q^{-1}$. Compared with Eq. (7), we can observe that the extrapolation assumes $\lambda_i = \gamma^{(t)}$ for all i and performs Aitken's acceleration accordingly.

We can estimate $\gamma^{(t)}$ as follows. Let $\theta^{(t)} = M(\theta^{(t-1)})$ and $\theta^{(t+1)} = M(\theta^{(t)})$. We have, by Eq. (5):

$$J(\theta^{(t)} - \theta^{(t-1)}) \approx \theta^{(t+1)} - \theta^{(t)} = \theta_M^{(t)} - \theta^{(t)}.$$

Then, we substitute J with $\gamma^{(t)}$. Let $\gamma^{(t)}(\theta^{(t)} - \theta^{(t-1)}) = \theta_M^{(t)} - \theta^{(t)}$, we have

$$|\gamma^{(t)}| \quad = \quad \frac{\|\theta_M^{(t)} - \theta^{(t)}\|}{\|\theta^{(t)} - \theta^{(t-1)}\|}. \tag{11}$$

Since λ_i's are non-negative in both EM and GIS algorithms by Assumption 1, our estimation of $\gamma^{(t)}$ is defined by:

$$\gamma^{(t)} \equiv \frac{\|\theta_M^{(t)} - \theta^{(t)}\|}{\|\theta^{(t)} - \theta^{(t-1)}\|}. \tag{12}$$

Therefore, we have $\gamma^{(t)}$ as an estimate of λ_{max}. It can be shown that as $t \to \infty$, $\gamma^{(t)} \leq \lambda_{max}$ asymptotically in the neighborhood of θ^* for the EM algorithm (Hesterberg, 2005). This estimation applies two previous estimates in an attempt to extrapolate to the local optimum. Because this is similar to the hop, step and jump phases in triple jump, Huang et al. (2005) named this method the *Triple Jump Acceleration.*

Aitken's acceleration does not guarantee to reach θ^* directly from $\theta^{(t)}$ because it is based on the assumption that $\theta^{(t)}$ is within the neighborhood of θ^*. When $\theta^{(t)}$ is not close enough to θ^*, the extrapolation jumps to a $\theta^{(t+1)}$ that might fail to improve the likelihood. By applying the idea of *adaptive overrelaxed bound optimization methods* (Salakhutdinov and Roweis, 2003), we can combine a bound optimization method with the triple jump extrapolation that is guaranteed to converge, as described in (Huang et al., 2005).

Since estimating $\gamma^{(t)}$ applies two previous estimates of the parameter vector, the triple jump method invokes Eq. (12) at every other iteration, if all extrapolations successfully improve the likelihood. Starting from $\theta^{(0)}$ in the neighborhood of θ^*, we need to apply the underlying fixed-point iteration mapping M to obtain $\theta_M^{(0)} = \theta^{(1)}$, again to obtain $\theta_M^{(1)}$, and then we can apply Eq. (10), the triple jump extrapolation, to obtain $\theta^{(2)}$. To apply the extrapolation again, we cannot simply use $\theta^{(1)}$, $\theta^{(2)}$ and $\theta_M^{(2)}$ in Eq. (12) to obtain $\gamma^{(2)}$, because in Eq. (12), $\theta^{(t)}$ must be obtained by the mapping M too so that the ratio is a reasonable estimate of the eigenvalue. Therefore, to apply the extrapolation again, we need to apply M to obtain $\theta_M^{(2)} = \theta^{(3)}$, again to obtain $\theta_M^{(3)}$, and then we can apply the extrapolation to obtain $\theta^{(4)}$, and so on. Therefore, the triple jump algorithm applies the extrapolation at the $2i$-th iteration, $i = 1, 2, \ldots$, assuming that all extrapolations successfully improve the likelihood.

In the case that the parameter space has only one dimension, Eq. (12) provides an exact approximation of $J(\theta^*) = M'(\theta^*)$. When the convergence is slow, we will have $M' \approx 1$ and $\gamma^{(t)} \approx 1$, too. Then, $1/(1-\gamma^{(t)})$ will be very large and provide a large acceleration. In a multi-dimensional case, the convergence rate is determined by the largest $\text{eig}(J(\theta^*))$. When the eigenvalue is close to one, the convergence will be slow. Aitken's acceleration can provide a large acceleration when we have a good approximation of λ_{max} but may also cause numerical insatiability when $\lambda_{max} \approx 1$. Since $\gamma^{(t)} \leq \lambda_{max}$, the triple jump extrapolation is numerically more stable than directly using the eigenvalues.

3. Global and Componentwise Extrapolations

It is also possible to approximate λ_i in each dimension, or divide the parameter space into subspaces and use Eq. (12) to obtain an approximation for each subspace, as reported in (Huang et al., 2005). In this section, we investigate general conditions of when componentwise extrapolation should be preferred.

3.1 Componentwise Extrapolation

Huang et al. (2005) proposed a different estimate. In that work, the parameter vector is divided into sub-vectors. A sub-vector may be a single component or consist of a subset of components in the parameter vector. Each sub-vector has its estimate of eigenvalue:

$$\gamma_p^{(t)} \equiv \frac{\|\theta_{Mp}^{(t)} - \theta_p^{(t)}\|}{\|\theta_p^{(t)} - \theta_p^{(t-1)}\|} , \tag{13}$$

where $p \leq l$ is the index of sub-vectors. Then we can perform the extrapolation separately for each sub-vector:

$$\theta_p^{(t+1)} \quad = \quad \theta_p^{(t)} + (1-\gamma_p^{(t)})^{-1}(\theta_{Mp}^{(t)} - \theta_p^{(t)}) . \tag{14}$$

The final estimate of the parameter vector is the concatenation of all resulting sub-vectors. Extrapolation using Eq. (13) (i.e., componentwise extrapolation) may accelerate convergence and outperform extrapolation using Eq. (12) (i.e., global extrapolation) in certain cases, though in general using Eq. (12) performs better for the EM algorithm. A rule of thumb is that when componentwise convergence rates are different, using Eq. (13) may perform better (Schafer, 1997, chap. 3).

3.2 Convergence Rates and Extrapolation Methods

Let us recall that the global rate of convergence R of the EM algorithm is defined in Eq. (9). Now let $\varphi_i^{(t)} = \theta_i^{(t)} - \theta_i^*$ denote the componentwise difference. The i-th componentwise rate of convergence is defined as

$$R_i = \lim_{t\to\infty} R_i^{(t)} \equiv \lim_{t\to\infty} \frac{\varphi_i^{(t+1)}}{\varphi_i^{(t)}} \,. \tag{15}$$

We note that $\varphi^{(t)}$ in Eq. (9) is a vector while $\varphi_i^{(t)}$ is a scalar.

When $R_i = R$ for all component i, global extrapolation is more appropriate than componentwise extrapolation, and vice versa. The global rate of convergence R is known to be the largest eigenvalue of the Jacobian (Dempster et al., 1977). R_i is also one of the eigenvalues but due to eigen transformation, R_i is not necessarily the i-th eigenvalue. The following Lemma is helpful for us to understand why R and R_i are eigenvalues and which eigenvalue corresponds to R_i.

Lemma 2 *The $l \times l$ Jacobian matrix J can be decomposed into a linear combination of its eigenvalues*

$$J = \sum_{j=1}^{k} \lambda_j u_j v_j^T = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + \cdots + \lambda_k u_k v_k^T \,,$$

where $1 > \lambda_1 > \lambda_2 > \cdots > \lambda_k > 0$ are $k(\leq l)$ distinct eigenvalues of J, u_j, v_j $(j = 1, \cdots, k)$ form the bases of the j-th eigenvector spaces for J and J^T, respectively. Moreover,

$$J^t = \sum_{j=1}^{k} \lambda_j^t u_j v_j^T. \tag{16}$$

Proof Since J is a real-valued square matrix and can be decomposed as $J = Q\Lambda Q^{-1}$. Let $Q = [\, u_1 \, \cdots \, u_l \,]$ and

$$Q^{-1} = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_l^T \end{bmatrix}.$$

Eq. (16) follows immediately from (Searle, 1982, page 293). ■

With this Lemma, Meng and Rubin (1994) showed that the global rate of convergence R for EM is the largest eigenvalue and gave the sufficient and necessary condition of when the componentwise rate $R_i = R$. We restate the proof of their findings less formally here.

A Taylor expansion yields

$$\begin{aligned} \varphi^{(t)} &= \theta^{(t)} - \theta^* \\ &\approx J(\theta^{(t-1)} - \theta^*) \\ &= J\varphi^{(t-1)} \\ &= J \cdot J \cdot \varphi^{(t-2)} \\ &= \vdots \\ &= J^t \varphi^{(0)} \,. \end{aligned}$$

From Lemma 2, we have

$$\varphi^{(t)} = \sum_{j=1}^{k} \lambda_j^t u_j v_j^T \varphi^{(0)}. \tag{17}$$

That is, the difference between the t-th estimate $\theta^{(t)}$ to the local maximum θ^* is a linear combination of the eigenvalues of J. Now, consider the i-th component θ_i of the parameter vector and the j-th largest eigenvalue λ_j of J. The contribution of λ_j to θ_i is

$$\lambda_j^t \cdot [u_j v_j^T] \cdot \varphi^{(0)} = \lambda_j^t \cdot [u_j v_j^T] \cdot \begin{bmatrix} \vdots \\ \varphi_i^{(0)} \\ \vdots \end{bmatrix} = \lambda_j^t \begin{bmatrix} \vdots \\ w_{ij} \\ \vdots \end{bmatrix}. \tag{18}$$

Note that $u_j v_j^T$ is a matrix defining the eigen transformation of the j-th eigenvalue. $u_j v_j^T$ maps the difference of the i-th component $\varphi_i^{(0)}$ to w_{ij}. If $w_{ij} \neq 0$, then λ_j contributes to the convergence of θ_i and the convergence rate for the i-th component is at least as slow as λ_j.

If for any component i, we have $w_{i1} \neq 0$, that is, the mapping result of the largest eigenvalue λ_1 is nonzero, then the global rate of convergence is at least as slow as the largest eigenvalue λ_1. That is, $R = \lambda_1$. If for a given component i, we have $w_{i1} \neq 0$, then the componentwise rate of convergence for the i-th component is as slow as the global rate of convergence. That is, $R_i = R$. Otherwise, the componentwise rate of convergence is different from the global rate. Meng and Rubin (1994) proved this by following the definitions of the componentwise rate and global rate of convergence with Lemma 2.

Corollary 3 *$R_i \neq R$ if $w_{i1} = e_i^T u_1 v_1^T \varphi^{(0)} = 0$, where e_i is the i-th column of the identity matrix I_d.*

An obvious case that makes $w_{i1} = 0$ is when $\varphi^{(0)}$ is a zero vector. That is, our initial value is exactly the local maximum, which is unlikely to happen. Since w_{i1} is the inner product of the i-th row of the matrix $u_1 v_1^T$ and $\varphi^{(0)}$, if they are orthogonal, then $w_{i1} = 0$. This is unlikely, too. A more possible case is that the i-th row of $u_1 v_1^T$ is a zero vector.

When $J = Q\Lambda Q^{-1}$ is a diagonal matrix, Q and Q^{-1} will be diagonal, too. As a result, $u_j v_j^T$ will be singular. That is, some of their rows will be zero vectors and thus make $R_i \neq R$. Therefore, we can conclude that when J is a diagonal matrix, we have $R_i \neq R$, and we should apply componentwise triple jump extrapolation. More precisely, we should also require that J is not only diagonal but also not proportional to the identity matrix so that all eigenvalues are distinct. Similarly, if J is block diagonal, $u_j v_j^T$ will also be singular and lead to the same consequence. We summarize our conclusion with the following claim:

Claim 4 *If the Jacobian J of the mapping of a bound optimization method is (block) diagonal, then applying componentwise triple jump extrapolation to the mapping will require less iterations to converge than applying global triple jump extrapolation.*

This claim is useful as a guideline for selecting which extrapolation method to apply. But to establish it as a guarantee of the rate of convergence, we need to make the following assumptions.

Assumption 5 *We assume that the estimations of λ_{max} and θ are perfect:*

A1 (Perfect estimation of λ_{max}) *Assuming that our estimation of the largest eigenvalues of J is perfectly accurate, then $\gamma^{(t)}$ in Eq. (10) will be λ_{max} of J in global extrapolation, while in componentwise extrapolation, $\gamma_p^{(t)}$ in Eq. (14) will be $\lambda_{p\,max}$, the largest eigenvalue of J in the dimensions corresponding to sub-vector θ_p.*

A2 (Perfect partitioning of θ) *We also assume that when applying componentwise extrapolation, we divide the parameter vector θ into sub-vectors corresponding to the diagonal blocks in J, if J is block diagonal. That is, we have a sub-vector θ_p in θ from dimension p to q, $1 \leq p \leq q \leq l$, if and only if $J(p:q, p:q)$ is one of the diagonal blocks in J. Here we follow the colon notation as defined in (Golub and Loan, 1996, Sec. 1.2.5). Clearly, when J is diagonal, our sub-vectors will contain each of the components of θ.*

Under these assumptions, we can show that starting at the same parameter vector, if J is block diagonal, applying a step of componentwise extrapolation will move the parameter vector closer to the local optimum than applying a step of global extrapolation.

Theorem 6 *Given Assumption 5, and a starting parameter vector, $\theta^{(t)}$, let $\theta_{TJ}^{(t+1)}$ be the new parameter vector after applying a step of global triple jump extrapolation and $\theta_{CTJ}^{(t+1)}$ be the new parameter vector after applying a step of componentwise triple jump extrapolation. If the Jacobian J is block diagonal, then $\|\theta_{TJ}^{(t+1)} - \theta^*\| \geq \|\theta_{CTJ}^{(t+1)} - \theta^*\|$.*

Proof From (A2), since J is block diagonal, for all blocks $J(p:q, p:q)$ in J,

$$\theta_{Mp}^{(t)} - \theta_p^* = J(p:q, p:q)(\theta_p^{(t)} - \theta_p^*) \; . \tag{19}$$

Consider a sub-vector $\theta_p^{(t)}$. Applying a triple jump extrapolation, we have

$$\begin{aligned} \theta_p^{(t+1)} &= \theta_p^{(t)} + (1-\gamma)^{-1}(\theta_{Mp}^{(t)} - \theta_p^{(t)}) \\ &= \theta_p^{(t)} + (1-\gamma)^{-1}(J(p:q, p:q)(\theta_p^* - \theta_p^{(t)}) + \theta_p^* - \theta_p^{(t)}) \; , \end{aligned}$$

for both global and componentwise extrapolation. Whether it is global or componentwise depends on which eigenvalue that we use for γ. For global extrapolation, from (A1), we have $\gamma = \lambda_{max}$, the largest eigenvalue of J, while for componentwise, $\gamma = \lambda_{p\,max}$, the largest eigenvalue of $J(p:q, p:q)$. This is because we assume that we can estimate these eigenvalues perfectly.

The above equation can be considered as a new mapping of the parameter vector. Rearranging the equation, we have

$$\theta_p^{(t+1)} - \theta_p^* = \left[I - \frac{I - J(p:q, p:q)}{1-\gamma} \right] (\theta_p^{(t)} - \theta_p^*) \; , \tag{20}$$

where $I - \frac{I-J(p:q,p:q)}{1-\gamma}$ is the Jacobian of the new mapping, with eigenvalues $1 - \frac{1-eig(J(p:q,p:q))}{1-\gamma}$. Let λ be one of the eigenvalues of $J(p:q, p:q)$. Then

$$\lambda_{ctj} = 1 - \frac{1-\lambda}{1-\lambda_{p\,max}}$$

is one of the eigenvalues of the new mapping when we apply componentwise extrapolation. Since $0 \leq \lambda \leq \lambda_{p\,max} \leq 1$, λ_{ctj} is negative. Similarly, when we apply global extrapolation, in the same dimension, the eigenvalue will be

$$\lambda_{tj} = 1 - \frac{1-\lambda}{1-\lambda_{max}} \leq \lambda_{ctj} \ .$$

The inequality holds because $\lambda_{max} \geq \lambda_{p\,max}$. Since λ_{ctj} is negative, we have $|\lambda_{tj}| \geq |\lambda_{ctj}|$ for each dimension of the new mapping. From Eq. (20) and Lemma 2, the difference $\|\theta_p^{(t+1)} - \theta_p^*\|$ obtained by applying global optimization will therefore be greater than or equal to the difference obtained by applying componentwise extrapolation. Also, since the new parameter vectors $\theta_{TJ}^{(t+1)}$ and $\theta_{CTJ}^{(t+1)}$ are simply the concatenation of their respective $\theta_p^{(t+1)}$, for all p, we obtain that $\|\theta_{TJ}^{(t+1)} - \theta^*\| \geq \|\theta_{CTJ}^{(t+1)} - \theta^*\|$. ■

Note that Eq. (19) may not hold unless J is block diagonal. In the case when J is not diagonal, it is likely that using λ_{max} as γ may move the parameter vector closer to the optimum, and therefore applying global extrapolation will be more effective. A caveat of this theorem is that it does not guarantee anything if the extrapolation is applied from different parameter vectors. Another caveat is that in the real world, Assumption 5 may not hold. That is, our partitioning of parameter vectors and our estimation of eigenvalues may not be accurate. Nevertheless, in the remainder of this paper, we will empirically show that our claim is still valid for a variety of probabilistic models even if our estimation is not perfectly accurate.

4. Case Study: EM

In this section, we start by deriving the general form of Jacobian of the EM mapping. Then we review two simple mixture-of-Gaussian models. They were selected because they are so simple that the closed-form of their Jacobians have been derived in previous works, and one of them is diagonal and the other is not, perfectly fitting our need of initial verification.

Then we consider the Bayesian Network models because training their parameters is one of the most widely used applications of the EM algorithm. We investigate when their Jacobians are (block) diagonal. We show that the rate of missing data and diagonality are correlated and conclude that componentwise extrapolation should be preferred when the missing rate is high, otherwise global extrapolation should be used. We present results of numerical experiments to verify our claim.

4.1 Jacobian of the EM Mapping

Suppose we want to use the EM algorithm to train a probabilistic model with a l-dimensional parameter vector θ from an incomplete data set $\mathcal{D} = (\mathcal{D}_{obs}, \mathcal{D}_{mis})$, where $\mathcal{D}_{obs}$ denotes the observed values and $\mathcal{D}_{mis}$ denotes the missing values. Now let $d = \{\ldots, y_{im}, \ldots\}$ be the data set with all missing values in $\mathcal{D}$ imputed (i.e., filled in by some estimation method) and $f(d|\theta)$ be the probability density of d given θ, then

$$L(\theta) = \log f_{\mathcal{D}_{obs}}(\mathcal{D}_{obs}|\theta) = \log \int f(d|\theta) d\mathcal{D}_{mis}$$

is the log-likelihood of the observed data $\mathcal{D}_{obs}$. The maximum likelihood principle states that the best parameter vector is the one that maximizes the log-likelihood of the observed data. However, it

is usually difficult to derive a closed-form solution for the integral for the observed data likelihood in a complex probabilistic model. The EM algorithm solves this problem by iteratively imputing the missing data and searching for θ^* that maximizes the expected complete data likelihood.

To determine which extrapolation may be more effective, we review the general form of Jacobian of the EM mapping matrix, derived by (Dempster et al., 1977). The Jacobian of the EM algorithm is given by

$$J = I - \mathcal{I}_{obs}\mathcal{I}_c^{-1} , \tag{21}$$

where I is the $l \times l$ identity matrix,

$$\mathcal{I}_c = E\left[-\frac{\partial^2[\log f(\mathcal{D}|\theta)]}{\partial\theta\partial\theta^T} \middle| \mathcal{D}_{obs}, \theta \right] \Big|_{\theta=\theta^*}$$

is the Fisher's information of the expected complete data, and

$$\mathcal{I}_{obs} = -\frac{\partial^2 L(\theta)}{\partial\theta\partial\theta^T}\Big|_{\theta=\theta^*}$$

is the Fisher's information of the observed data. Fisher's information measures how flat the likelihood surface is. Computing Fisher's information can be intractable for complex models with a high dimensional parameter space. However, for our purpose, we only need to see if the Jacobian is (block) diagonal.

4.2 Mixtures of Gaussians

Our first example is from Meng and Rubin (1994). Suppose we have a set of one-dimension data $\mathcal{D}_{obs} = X = \{x_i | i = 1, 2, \ldots\}$ from the following distribution:

$$f_{ex1}(X|\mu, \sigma^2) = (1 - \pi)N(\mu, \sigma^2) + \pi N(\mu, \sigma^2/\lambda). \tag{22}$$

That is, the data set comes from a mixture of two Gaussians with the same mean but different variances. Assuming that we know the mixture ratio π and constant λ, then our parameter vector is $\theta = (\mu, \sigma^2)$. We can estimate the parameter vector from data by the EM algorithm by creating a missing, unobservable variable $Q \in \{1, \lambda\}$ that assigns membership of an observed variable X. Therefore, our complete, augmented data set is $\mathcal{D} = \{(x_i, q_i) | i = 1, 2, \ldots\}$.

We can use Eq. (21) to compute the Fisher's information of the observed and missing data by differentiating the log-likelihood of the data twice to determine whether the Jacobian of this model is diagonal. If both information matrices, $\mathcal{I}_c$ and $\mathcal{I}_{obs}$, are diagonal, then the Jacobian will be diagonal, too. Though this model is simple, its Jacobian is still quite complex. Nevertheless, Meng and Rubin (1994) showed that in this case, the Jacobian is a 2×2 diagonal matrix and empirically show that the componentwise rate of convergence is different:

$$\begin{pmatrix} \frac{E\left(\text{Var}(q|x,\theta)\left(\frac{x-\mu}{\sigma}\right)^2\right)}{(\lambda\pi+(1-\pi))} & 0 \\ 0 & E\left(\text{Var}(q|x,\theta)\left(\frac{x-\mu}{\sigma}\right)^4\right)/2 \end{pmatrix}$$

Interestingly, with a different parameter vector, another one-dimensional Mixtures of Gaussian model from (Louis, 1982) has a Jacobian that is not diagonal. In this case, the parameter vector is $\theta = (\mu_0, \mu_1, \pi)$ with the variance known to be $\sigma^2 = 1$. The distribution for the observed data is

$$f_{ex2}(X|\mu 0, \mu_1, \pi) = (1 - \pi)N(\mu_0, 1) + \pi N(\mu_1, 1) . \tag{23}$$

We introduce an additional unobserved membership assignment variable $Q \in \{0, 1\}$ for the augmented complete data set. In this case, Louis (1982) showed that $\mathcal{I}_c$ is diagonal, while $\mathcal{I}_{obs}$ is not diagonal, though it is symmetric. Thus J is not diagonal.

4.3 Bayesian Networks

We now consider a more practical model, the Bayesian network, to determine when its Jacobian is diagonal. The EM algorithm is applied to train a Bayesian network model when we have latent variables whose values are not observable or when some of the values of variables in the training data are missing. In theory, the Jacobian of the EM algorithm for the Bayesian network can be obtained from Eq. (21). However, unlike the toy examples in the previous sub-section, it is difficult to obtain the closed-form of the Jacobian for Bayesian networks. Fortunately, since our purpose is only to determine if the Jacobian is diagonal, there is no need to obtain the Jacobian. In fact, if we can show that the Fisher's information $\mathcal{I}_{obs}$ and $\mathcal{I}_c$ are (block) diagonal, then the Jacobian must be (block) diagonal as well. Therefore, it suffices to just determine if the off-diagonal elements of $\mathcal{I}_{obs}$ and $\mathcal{I}_c$ are zero.

A Bayesian network consists of a set of variables $X = \{X_i | i = 1, 2, \ldots\}$, the graph structure of the variables, and their conditional probability tables. Suppose we have a variable X_i whose parent nodes include a set of variables denoted by U_i. The conditional probability table for a variable X_i consists of entries of the form

$$w_{ijk} \equiv \Pr(X_i = x_{ik} | U_i = u_{ij})$$

to denote the probability that X_i has its k-th possible value x_{ik} under the condition that its parent U_i has the j-th combination of values, u_{ij}. Since w_{ijk} denotes the probability, to ensure that w_{ijk} is in $[0, 1]$ during the training process, a common technique used in practice is applying softmax reparameterization:

$$w_{ijk} = \frac{e^{\theta_{ijk}}}{\sum_{k'} e^{\theta_{ijk'}}}$$

Therefore, the parameters that we want to estimate from data using the EM algorithm are the set $\theta = \{\theta_{ijk} | i, j, k = 1, 2, \ldots\}$.

A Bayesian network models the joint event $y = \{\ldots, X_i = x_{ik}, \ldots\}$, which is a set of variable-value pairs representing that variable X_i is instantiated by the value x_{ik}. Some of the variable's value in an event may be missing either because in that particular case, its value is not available, or because the variable is a latent variable. Many algorithms are available for efficiently computing $P_\theta(y_{mis}|y_{obs})$, the conditional probability of missing values given observed values and the parameters θ. The Bayesian network allows us to factorize any conditional probability into an expression of w_{ijk}, the entries in the conditional probability table (Russell et al., 1995). The training data for the Bayesian network is a set of I.I.D. event cases $\mathcal{D}_{obs} = \{\ldots, y_{obs}, \ldots\}$. The EM algorithm is applied to learn θ that maximizes the log-likelihood of the observed data.

We start by giving Lemma 7 which summarizes possible values of $\frac{\partial}{\partial \theta_{ijk}} w_{ijk}$ that will be frequently used:

Lemma 7 *For Bayesian networks with softmax reparameterization, the derivative of $w_{i'j'k'}$ with respect to θ_{ijk} is:*

$$\frac{\partial w_{i'j'k'}}{\partial \theta_{ijk}} = \begin{cases} 0 & : i \neq i' \text{ or } j \neq j' \\ -w_{ijk}w_{ijk'} & : (i,j) = (i',j') \text{ and } k \neq k' \\ w_{ijk}(1-w_{ijk}) & : (i,j,k) = (i',j',k') \end{cases} \tag{24}$$

Proof See Appendix. ■

Now let us consider the problem of determining the diagonality of $\mathcal{I}_{obs}$. Its off-diagonal elements are the second partial derivatives of the log-likelihood of observed data $L(\theta)$ with respect to two different parameters. When they are all zero, $\mathcal{I}_{obs}$ will be diagonal.

Since we assume that the training data $\mathcal{D}_{obs}$ is I.I.D., $\frac{\partial}{\partial \theta_{ijk}} L(\theta) = \sum_{\mathcal{D}_{obs}} \frac{\partial}{\partial \theta_{ijk}} \log P_\theta(y)$. Lemma 8 describes the first order derivative of $P(y)$ and when the derivative is zero.

Lemma 8 *For Bayesian networks with softmax reparameterization, the derivative of $P(y)$ with respect to θ_{ijk} is:*

$$\frac{\partial}{\partial \theta_{ijk}} P(y) = P(x_{ik}, u_{ij}, y) - w_{ijk} P(u_{ij}, y), \tag{25}$$

and the derivative must be 0 if u_{ij} d-separates (Pearl, 1988) the observations in $y - \{U_i = u_{ij}\}$ and X_i, that is, if $P(X_i|u_{ij}, y) = P(X_i|u_{ij})$.

Proof See Appendix. ■

Lemma 8 implies that "$X_i = x_{ik}$" must not appear in y to satisfy the d-separation condition. Intuitively, if "$X_i = x_{ik}$" appears in y, y and X_i will not be conditionally independent given u_{ij} because $P(X_i|u_{ij}, y)$ will be equal to 1 if $X_i = x_{ik}$ and 0 otherwise. Based on Lemma 8, Theorem 9 shows the conditions in which $\frac{\partial^2 \log P(y)}{\partial \theta_{i'j'k'} \partial \theta_{ijk}} = 0$, which implies that the second derivatives of $L(\theta)$ and thus the off-diagonal elements of $\mathcal{I}_{obs}$ are zero.

Theorem 9 $\frac{\partial^2 \log P(y)}{\partial \theta_{i'j'k'} \partial \theta_{ijk}}$ *is 0 if one of the following conditions holds:*

1. *The condition stated in Lemma 8 holds, or*
2. *$u_{i'j'}$ d-separates the observations in $y \cup \{u_{ij}, x_{ik}\}$ and $X_{i'}$.*

Proof See Appendix. ■

The first condition is straightforward because the derivative of zero is still zero. We can obtain the second condition by expanding y to $y \cup \{u_{ij}, x_{ik}\}$ and applying Lemma 8 to $\theta_{i'j'k'}$ again. Though both conditions in Theorem 9 are difficult to check directly in real applications, in the next subsection, we will relate them to the proportion of missing values in training data, which is easy to check to determine which extrapolation method should be used.

Next, we consider the diagonality of $\mathcal{I}_c$. The elements in $\mathcal{I}_c$ are the expectation of the second partial derivatives of the complete data log-likelihood. Since if the second partial derivatives are zero, their expected values must be zero, too, we will consider the second partial derivatives. We will

show that $\mathcal{I}_c$ is always a block diagonal matrix for any Bayesian network. Each block corresponds to parameters θ_{ijk} that share the same i and j but different k. In other words, they are the parameters of the k possible values of X_i given its parents $U_i = u_{ij}$. Note that since here we are considering the complete data log-likelihood, missing values will be imputed and $\forall i \exists k, j \{u_{ij}, x_{ik}\} \subset y$. That is, each X_i and its parents U_i are instantiated with no missing value in $y \in \mathcal{D}$, the complete data set.

Lemma 10 *If u_{ij} and x_{ik} are observed in y, $\frac{\partial^2 \log P(y)}{\partial \theta_{ijk'} \partial \theta_{ijk}}$, the second order derivative of two parameters with the same i, j, will be:*

$$\frac{\partial^2 \log P(y)}{\partial \theta_{ijk'} \partial \theta_{ijk}} = \begin{cases} -w_{ijk}(1 - w_{ijk}) & \text{:if } k' = k \\ w_{ijk} w_{ijk'} & \text{:if } k' \neq k. \end{cases}$$

Moreover, the derivatives with other (i', j', k')'s will be zero in this case.

Proof See Appendix. ■

From Lemma 10, we can order θ_{ijk} by i, j and k so that $\frac{\partial^2 \log P(\mathcal{D})}{\partial \theta^2}$ is a block diagonal matrix, implying that $\mathcal{I}_c = E(\frac{\partial^2 \log P(\mathcal{D})}{\partial \theta^2})$ is also a block diagonal matrix.

Theorem 11 *$\mathcal{I}_c$ is a block diagonal matrix:*

$$\begin{pmatrix} B_{11} & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & & \ddots & \vdots \\ \vdots & & B_{ij} & & 0 \\ \vdots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & B_{IJ} \end{pmatrix},$$

in which each block B_{ij} is a $k \times k$ matrix:

$$\begin{pmatrix} E\left(\frac{\partial^2 \log P(\mathcal{D})}{\partial \theta_{ij1} \partial \theta_{ij1}}\right) & \cdots & E\left(\frac{\partial^2 \log P(\mathcal{D})}{\partial \theta_{ij1} \partial \theta_{ijk}}\right) \\ \vdots & \ddots & \vdots \\ E\left(\frac{\partial^2 \log P(\mathcal{D})}{\partial \theta_{ijk} \partial \theta_{ij1}}\right) & \cdots & E\left(\frac{\partial^2 \log P(\mathcal{D})}{\partial \theta_{ijk} \partial \theta_{ijk}}\right) \end{pmatrix}.$$

Proof See Appendix. ■

Since $\mathcal{I}_c$ is a block diagonal matrix, J will be a block diagonal matrix if $\mathcal{I}_{obs}$ is also block diagonal with the same block layout, and if that is the case, we should apply componentwise extrapolation.

4.4 Missing Data and Diagonality

Theorem 9 implies the following corollary that allows us to check whether $\mathcal{I}_{obs}$ and thus J is block diagonal.

Corollary 12 *Let V_i and $V_{i'}$ be the descendents of X_i and $X_{i'}$. $\frac{\partial^2 \log P(y)}{\partial\theta_{i'j'k'}\partial\theta_{ijk}}$ is 0 if V_i or $V_{i'}$ contains none of the observed random variables in y.*

From Corollary 12, we can show that the more missing data we have, the Jacobian is closer to diagonal or block diagonal, and the superiority of the performance of componentwise extrapolation to global extrapolation is more obvious. This is because the higher the missing rate or the less the values observed in y, the less likely they may appear in descendent nodes V_i and $V_{i'}$. Therefore, $\frac{\partial^2 \log P(y)}{\partial\theta_{i'j'k'}\partial\theta_{ijk}}$ will be more likely to be zero as $\mathcal{I}_{obs}$ is close to block diagonal.

The effect of high missing rates to the whole data set $\mathcal{D}_{obs}$ is that $\frac{\partial^2 \log P(\mathcal{D}_{obs})}{\partial\theta_{i'j'k'}\partial\theta_{ijk}}$ will be close or equal to zero because $\frac{\partial^2 \log P(\mathcal{D}_{obs})}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} = \sum_y \frac{\partial^2 \log P(y)}{\partial\theta_{i'j'k'}\partial\theta_{ijk}}$. Hence, most off-diagonal elements will be close or equal to zero and the Jacobian will appear like a diagonal or block diagonal. It follows that when training a Bayesian network with the EM algorithm, if the missing rate of the training data is high, componentwise extrapolation will be more effective than global extrapolation.

4.5 Semi-Supervised Bayesian Classifier

We provide theoretical analysis with experiments on the semi-supervised Bayesian classifier, which consists of a cluster random variable C and a set of feature random variables $F_1, \ldots, F_N$. There are N links from C to each F_n. The model assumes that the feature random variables are conditionally independent given C.

From Theorem 11, we know that $\mathcal{I}_c$ is block diagonal. Therefore, we only need to discuss $\mathcal{I}_{obs}$ of the model, which is simpler than general Bayesian networks in that every feature node shares the same parent node. Note that C and F_n might contain missing values. Theorem 13 describes conditions when off-diagonal elements of $\mathcal{I}_{obs}$ are zero.

Theorem 13 $\frac{\partial^2}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} \log P(y)$ *of a Bayesian classifier is zero if one of the following conditions is satisfied:*

1. *X_i is a feature variable and is not observed, or*
2. *$X_{i'}$ is a feature variable and is not observed.*

Proof See Appendix. ■

Corollary 14 $\frac{\partial^2}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} \log P(y)$ *of a Bayesian classifier is nonzero if one of the following conditions is satisfied:*

1. *$i = i'$ and X_i is a class variable,*
2. *X_i and $X_{i'}$ are feature and class variables and the feature variable is observed, or*
3. *X_i and $X_{i'}$ are feature variables or the same feature variable and are observed.*

Proof See Appendix. ■

Corollary 14 implies that whether $\mathcal{I}_{obs}$ is block diagonal or not depends on the missing rates. First, we consider the case when the missing rate is low. Suppose that the features are all observed. From Theorem 13, no element is guaranteed to be zero in $\frac{\partial^2 \log P(y)}{\partial \theta^2}$ so that $\sum_{y'} \frac{\partial^2 \log P(y')}{\partial \theta^2}$ is unlikely to be a block diagonal matrix. However, if the missing rate is high, $\mathcal{I}_{obs}$ will be much closer to a block diagonal matrix. An extreme case is that only one feature is observed in every training example. From Theorem 13, most values outside the block diagonal area will be zero. Therefore, CTJEM will be more likely to outperform global TJEM under such circumstances.

4.6 Experimental Results

We report convergence rate comparison of the EM algorithm, the TJEM algorithm (i.e., the triple jump EM algorithm applying global extrapolation), and the CTJEM algorithm (i.e., the triple jump EM algorithm applying componentwise extrapolation) for the probabilistic models discussed in this section. We will show that the experimental results are consistent with our prediction. We review these models as follows.

- The Gaussian mixture model defined by Meng and Rubin (1994) in Eq. (22) and the model by Louis (1982) in Eq. (23).

Meng's model has a diagonal Jacobian while Louis' model has a non-diagonal one.Therefore, we predict that CTJEM will outperform TJEM for Meng's model while TJEM will outperform CTJEM for Louis'. For each model, we synthesized a data set with 10,000 data points. To ensure the reliability of our results, we compare the rate of convergence of three EM variants 100 times. Each time these EM variants were initialized with a randomly generated parameter vector. For Meng's model, the termination condition for all EM variants was when the improvement of the likelihood between two consecutive iterations was less than 10^{-10}, and for Louis' model, 10^{-5}. The experiment was run on a Windows XP machine with Pentium 4 3.2GHz CPU and 2GB RAM.

- The ALARM Bayesian network (Cooper and Herskovits, 1992) with different proportions of random missing values.

The ALARM network is a real world Bayesian network with 37 multinomial nodes. We randomly assigned conditional probabilities as the true distributions and synthesized 2,000 examples as our experimental data set. Then, we randomly removed 50% and 90% values from the data set to produced two data sets, respectively. Our analysis in Section 4.4 predicts that CTJEM will be faster for the one with 90% missing because in that case J will be block diagonal, and TJEM will have an advantage over CTJEM for the data set with 50% of missing values. For both models, we tested three EM variants with 100 randomly generated initial parameter vectors. The termination condition for all trials was when the likelihood improvement was less than 10^{-4}. The experiment was run on a Windows Server machine with Xeon 3.4GHz CPU and 3.5GB RAM.

- A Bayesian classifier with different proportions of random missing values.

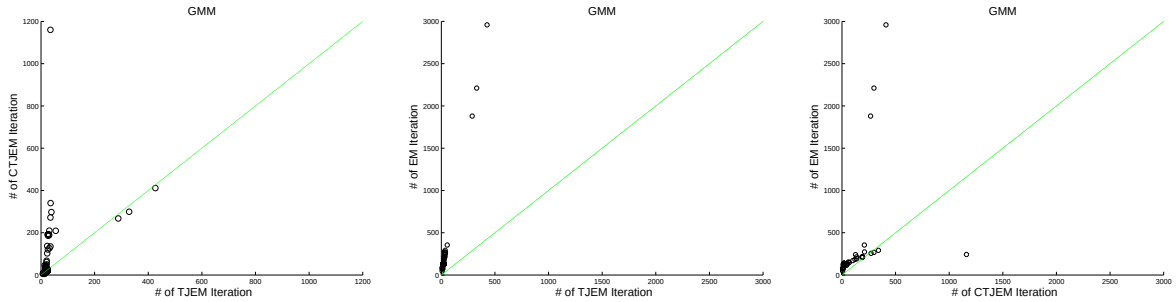
We designed a Bayesian classifier with 20 feature variables. All the features and class variables have five possible values. We randomly synthesized a data set with 10,000 examples and then randomly removed 50% and 90% of variable values to produce two data sets, respectively. According to our analysis in Section 4.5, it is expected that CTJEM will perform better for the data set with 90%

missing but not as well compared to TJEM for the data set with 50% missing. We compared the EM variants 100 times with different initialization and applied the same termination condition as that for the ALARM network for all trials. The experiment was run on a Windows XP machine with Athlon Dual 2GHz CPU and 2.37GB RAM.

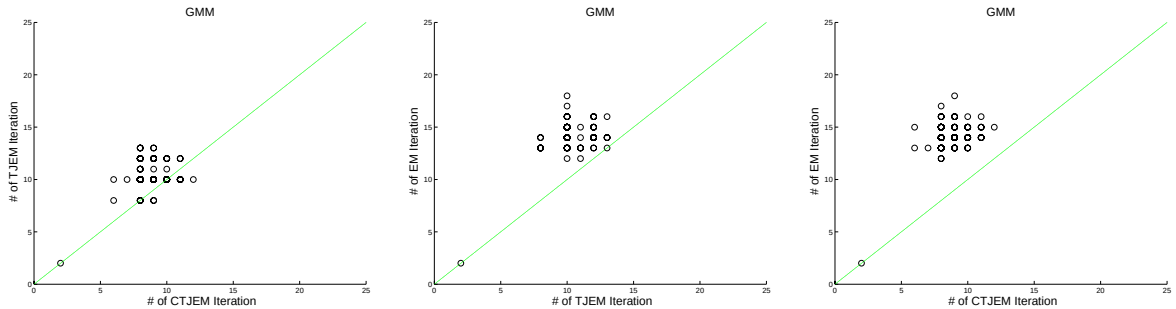
We use the scattered plots to compare the number of iterations required for convergence. More specifically, the number of iterations here is the number of times that E-step is executed. In each scattered plot, the coordination of each data point is the iterations of the X-axis method and the Y-axis method for the same trial. There are 100 data points in each plot, representing the results of 100 trials. A data point appear in the upper triangle if the X-axis method converges faster and in the lower triangle if the Y-axis method is faster. We arranged the axes of the plots such that if data points appear in the upper triangle, then the results are consistent with our predictions. Note that in EM, every iteration contains the E-step and M-step, but in TJEM and CTJEM, an iteration could be

1. a regular EM iteration;
2. a regular EM iteration plus extrapolation, after parameter vectors for extrapolation are ready;
3. an E-step only, if the parameter vector fails to improve the likelihood.

While the second situation may induce additional overheads, the third takes less time than a regular EM iteration. On average, EM, TJEM, and CTJEM take almost the same time to finish an iteration.



(a) Louis' Model, not block diagonal: TJEM vs CTJEM (left), TJEM vs EM (center), CTJEM vs EM (right).

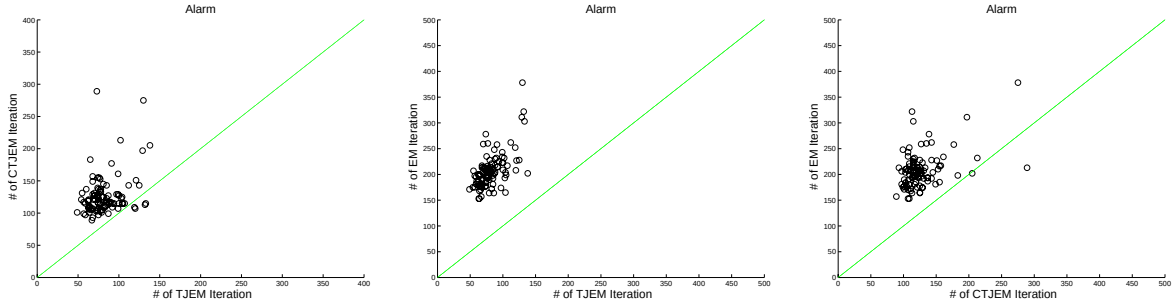


(b) Meng's model, block diagonal: CTJEM vs TJEM (left), TJEM vs EM (center), CTJEM vs EM (right).

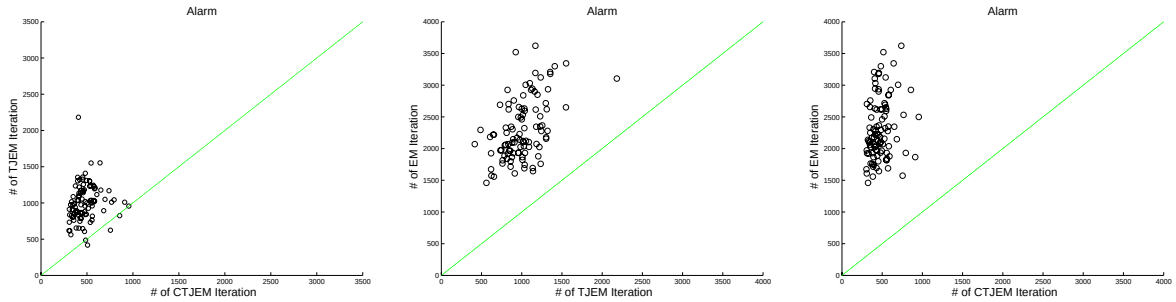
Figure 1: Convergence rate comparison of EM,TJEM, and CTJEM for two models of mixtures of Gaussians.

Figures 1, 2, and 3 show the results for the three pairs of the models in our case study. The plots show that both TJEM and CTJEM outperform EM, confirming that they are effective, as has been

established in previous work. More importantly, they clearly show that the results are consistent with our prediction. That is, for models with a (block) diagonal Jacobian, CTJEM converges faster, while for the other models, TJEM converges faster. Few exceptions either appear near the origin in the plots or near the diagonal line, meaning that those are either easy cases requiring very few iterations or ties. Ties occurred mostly for the two Bayesian network models with 50% of missing rates because in those cases, though the Jacobian may not be block diagonal but will be sparse and affect componentwise rates of convergence. As a result, the advantage of TJEM may not be as obvious as the advantage that CTJEM has for the data set with a 90% missing.



(a) ALARM model with 50% data missing, not block diagonal: TJEM vs CTJEM (left), TJEM vs EM (center), CTJEM vs EM (right).



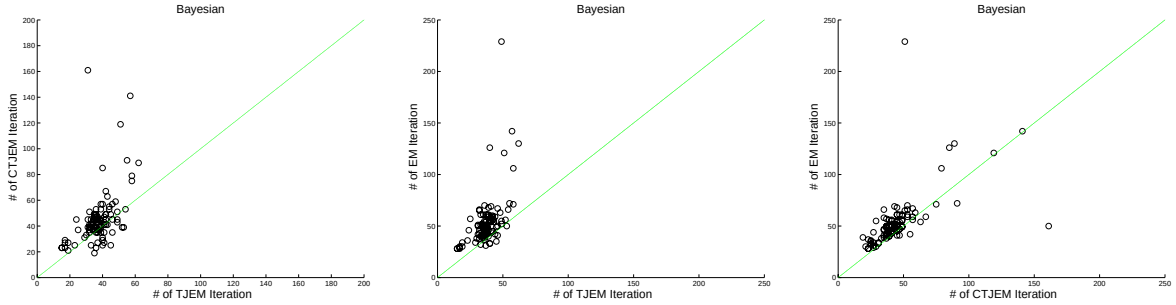
(b) ALARM model with 90% data missing, block diagonal: CTJEM vs TJEM (left), TJEM vs EM (center), CTJEM vs EM (right).

Figure 2: Convergence rate comparison of EM,TJEM, and CTJEM for ALARM Bayesian network models with different proportions of missing data.

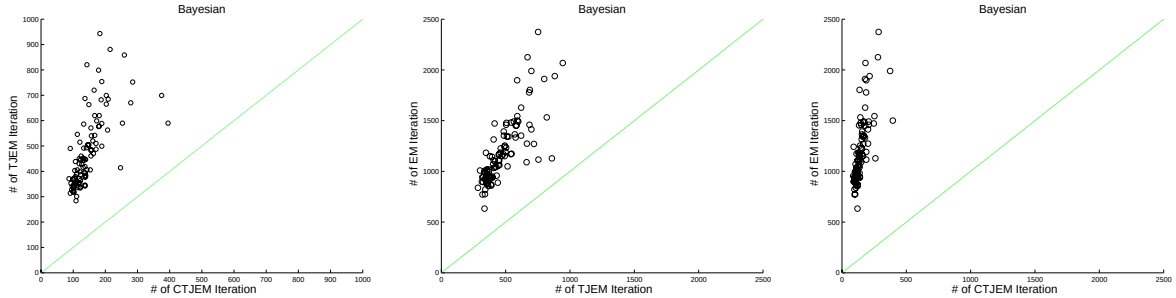
We also compare the CPU time required for different methods. Table 1 gives the average CPU time results over 100 trials. The speedup is defined as the CPU time required for EM divided by the CPU time required for TJEM or CTJEM. The results show that very high speedup can be achieved by extrapolation, especially when the right method is applied to the right problem according to our predictions.

5. Case Study: GIS

The generalized iterative scaling (GIS) algorithm is one of the bound optimization methods and becomes well-known because of the conditional random field (CRF) model. GIS can also be accelerated by the triple jump extrapolation method. This section presents our results which show



(a) Bayesian classifier model with 50% data missing, not block diagonal: TJEM vs CTJEM (left), TJEM vs EM (center), CTJEM vs EM (right).



(b) Bayesian classifier model with 90% data missing, block diagonal: CTJEM vs TJEM (left), TJEM vs EM (center), CTJEM vs EM (right).

Figure 3: Convergence rate comparison of EM,TJEM, and CTJEM for semi-supervised Bayesian classifier models with different proportions of missing data.

that when applying GIS to train a CRF, componentwise extrapolation is more effective than global extrapolation when the features are not correlated.

5.1 Conditional Random Fields

The CRF (Lafferty et al., 2001) is one of the most prevailing solutions to sequential data classification. In a CRF, sequences and their labels are transformed into features. The probability of a labeling result is a function of weighted sum of features. Training of CRFs is to assign proper weights for all features, trying to maximize the log-likelihood or penalized log-likelihood with the training data.

Let $\{x_1, \ldots, x_K\}$ denote a set of K data sequences and $\{y_1, \ldots, y_K\}$ the corresponding labels. A CRF defines l features to be transformed from a given instance (x, y):

$$F(x, y) = (f_1(x, y), \ldots, f_l(x, y))^T,$$

where $f_i(x, y)$ is the number of times that feature i occurs in (x, y). A CRF is parameterized by the weights for all features:

$$\theta = (\theta_1, \ldots, \theta_l)^T.$$

Type	Data set		CTJEM	TJEM	EM
Not Diagonal	GMM Louis	CPU Time (sec)	1.431	**0.674**	4.609
		Speedup	3.219	**6.834**	1
	ALARM 50%	CPU Time(min)	4.714	**2.701**	7.747
		Speedup	1.643	**2.868**	1
	Bayes 50%	CPU Time(min)	2.926	**2.361**	3.477
		Speedup	1.188	**1.472**	1
Diagonal or Block Diagonal	GMM Meng	CPU Time(sec)	**0.249**	0.286	0.391
		Speedup	**1.570**	1.367	1
	ALARM 90%	CPU Time(min)	**16.962**	33.976	75.748
		Speedup	**4.466**	2.229	1
	Bayes 90%	CPU Time(min)	**8.473**	27.327	68.862
		Speedup	**8.127**	2.520	1

Table 1: Average CPU time and speedup comparisons of EM, TJEM and CTJEM for various models and data missing rates.

Then, the conditional probability of y given x is:

$$p_\theta(y|x) = \frac{\exp(\theta^T F(x,y))}{Z_\theta(x)},$$

where $Z_\theta(x)$ is a normalization term:

$$Z_\theta(x) = \sum_y \exp(\theta^T F(x,y)).$$

Training of CRFs is to search for the weight vector that maximizes the log-likelihood function as the objective function. The log-likelihood function, denoted by $L(\theta)$, is:

$$\begin{aligned} L(\theta) &= \sum_k \log p_\theta(y_k|x_k) \\ &= \sum_k \left[\theta^T F(x_k,y_k) - \log Z_\theta(x_k)\right], \end{aligned} \tag{26}$$

The gradient of $L(\theta)$ along the axis of θ_i is $\nabla_i L(\theta)$:

$$\nabla_i L(\theta) = \widetilde{E} f_i - E f_i,$$

where $\widetilde{E} f_i$ is:

$$\widetilde{E} f_i = \sum_k f_i(y_k, x_k),$$

and $E f_i$ is the expected number of occurrence of f_i:

$$\sum_k \sum_y p_\theta(y|x_k) f_i(y, x_k).$$

5.2 Accelerating GIS by Triple Jump Extrapolation

This sub-section presents how to apply the triple jump extrapolation method to accelerate GIS. GIS can be formulated as a fixed point iteration method. At iteration t, GIS computes $\Delta(\theta^{(t)})$, which will be abbreviated as $\Delta^{(t)} = (\Delta_1^{(t)}, \ldots, \Delta_l^{(t)})$ for all $i = 1 \ldots l$ with this formula:

$$\Delta_i^{(t)} = \frac{1}{S} \log \frac{\widetilde{E} f_i}{E f_i},$$

where $S = \max_k \sum_i f_i(y_k, x_k)$ is the maximum number of feature occurrences in a training sequence (Lafferty et al., 2001). Then, we update the weights by $\theta^{(t+1)} = \theta^{(t)} + \Delta^{(t)}$, which is a fixed point iteration method with the definition of M as:

$$\theta^{(t+1)} = M(\theta^{(t)}) = \theta^{(t)} + \Delta^{(t)}.$$

When the global optimum θ^* is reached, $\Delta(\theta^*)$ should be a zero vector, which implies that we obtain the fixed-point solution $\theta^* = M(\theta^*)$. Since GIS can be considered as a fixed-point iteration method and Assumption 1 holds for GIS, we can apply the triple jump extrapolation method to accelerate GIS.

However, when applied to train CRFs, GIS suffers from overfitting. Recent works usually use Gaussian priors with the same mean and variance for each θ_i as a penalty term to train CRFs to avoid overfitting. The penalized log-likelihood function $\mathcal{L}(\theta)$ is:

$$\mathcal{L}(\theta) = L(\theta) - \sum_i \frac{(\theta_i - \mu)^2}{2\sigma^2} + \text{const.}, \tag{27}$$

and the gradient along the direction of θ_i is:

$$\nabla_i \mathcal{L}(\theta) = \widetilde{E} f_i - E f_i - \frac{\theta_i - \mu}{\sigma^2}. \tag{28}$$

A Gaussian prior with $\mu = 0.0$ is widely adopted as a penalty term to avoid overfitting, which is based on the idea that the number of important features with high weights should be less than the number of unimportant features. Based on Eq. (28), the weights of unseen features in the training data set should be zero at θ^*. In addition, the weights of observed features should be greater than zero, showing that they are more important than unseen ones. Sha and Pereira (2003) reported that they use negative weights to discourage some undesired features. More generally, we can also assign negative weights to unseen features by setting $\mu < 0$. In this case, the weights of unseen features should be μ, and those of observed features should be greater than μ.

To accelerate CRF training subject to $\mathcal{L}(\theta)$ with triple jump, we need to formulate penalized GIS (PGIS) as fixed-point iteration. Our goal is to find the global optimum, where the gradients are zero. In other words, PGIS is supposed to solve the equation $\nabla_i \mathcal{L}(\theta) = 0$:

$$\widetilde{E} f_i = E f_i + \frac{\theta_i - \mu}{\sigma^2}.$$

We multiply the RHS with $\exp(\frac{1}{\beta}(\theta_i - \theta_i))$ and obtain:

$$\widetilde{E} f_i = (E f_i + \frac{\theta_i - \mu}{\sigma^2}) \exp(\frac{1}{\beta}(\theta_i - \theta_i)), \tag{29}$$

where β is an arbitrary constant learning rate in $(0, 1)$. Then, we can rearrange Eq. (29) to obtain the following equation:

$$\theta_i = \theta_i + \beta \log \frac{\widetilde{E} f_i}{E f_i + \frac{\theta_i - \mu}{\sigma^2}}. \tag{30}$$

Now we have the PGIS algorithm, which updates each θ_i with Eq. (30). Let $M(\theta)$ be the RHS, the PGIS algorithm also solves $\theta = M(\theta)$ as the GIS algorithm. Assigning $\beta = 1/S$, we obtain a new update rule quite similar to GIS. The only difference is the additional term $\frac{\theta - \mu}{\sigma^2}$ in the RHS. In fact, GIS can be considered as a special case of PGIS when $\sigma^2 \to \infty$. In this way, we have formulated PGIS as fixed-point iteration so that we can apply the triple jump method to PGIS and accelerate its convergence. We therefore have many combinations of GIS variants to train a CRF model. We will consider the following combinations in this paper:

- GIS: generalized iterative scaling
- PGIS: penalized GIS
- TJPGIS: PGIS with global triple jump extrapolation
- CTJGIS: GIS with componentwise triple jump extrapolation
- CTJPGIS: PGIS with componentwise triple jump extrapolation

5.3 Jacobian of the PGIS Mapping

Previously, Salakhutdinov et al. (2003) provided the general form of Jacobian of the GIS mapping as follows:

$$I - \frac{1}{S} \mathrm{Cov}(\theta^*) D(\theta^*)^{-1},$$

where $\mathrm{Cov}(\theta^*)$ is covariance of the feature vectors under model distribution $p_{\theta^*}(x, y)$ and $\mathrm{D}(\theta^*) \equiv \mathrm{diag}(\bar{F}(\theta^*))$ is a diagonal matrix of $\bar{F}(\theta^*) \equiv \sum_{x,y} p_{\theta^*}(x, y) F(x, y)$, the mean of the feature vectors. We can derive the general form of Jacobian of the PGIS mapping by a similar technique.

Theorem 15 *Let $\Lambda \equiv diag(\frac{1}{\sigma^2})$. The Jacobian of the PGIS Mapping is*

$$I - \frac{1}{S}[Cov(\theta^*) + \Lambda][D(\theta^*) + \frac{1}{S}\Lambda]^{-1}$$

Proof See Appendix. ■

Since I, D and Λ are all diagonal, whether Jacobian of PGIS is diagonal or block diagonal depends only on the covariance matrix $\mathrm{Cov}(\theta^*)$, which is (block) diagonal when (subsets of) the features are independent. It follows that if subsets of the features are independent, componentwise extrapolation CTJPGIS should be preferred. We will empirically verify this corollary next.

5.4 Experimental Result: Synthesized Data

We performed a controlled experiment with two synthesized data sets. One was designed to have independent features and the other dependent features so that the Jacobian of PGIS for the former data set will be diagonal while for the latter will not. We synthesized both data sets using the hidden Markov model (HMM). To generate independent data, we used five state labels and 26 possible input values in HMM, with all state transition probabilities and emission probabilities assigned uniformly. The HMM model for dependent data also contains five state labels but only five possible input values. Its state transition probabilities were given by

$$p(y_i|y_{i-1}) = \begin{cases} 0.92, & \text{if } y_i = (y_{i-1}+1) \\ 0.02, & \text{otherwise} \end{cases}$$

and emission probabilities given by

$$p(x_i|y_i) = \begin{cases} 0.92, & \text{if } x_i = y_i \\ 0.02, & \text{otherwise} \end{cases}$$

we extracted two types of features as defined in (Lafferty et al., 2001) to train CRFs:

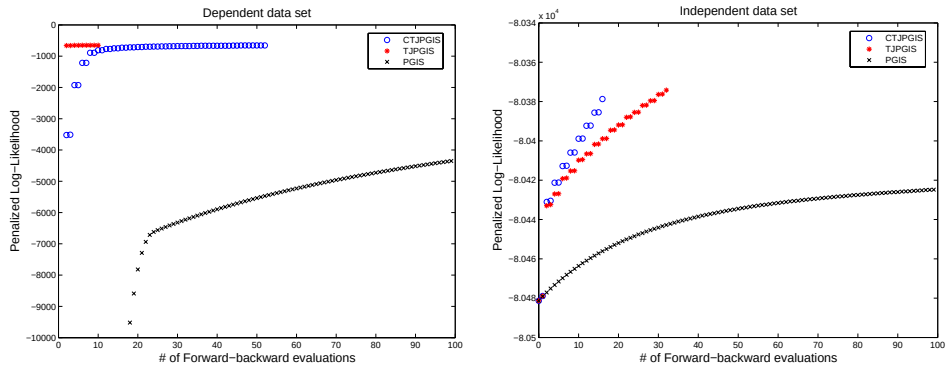
$$\begin{aligned} f_{y',y,<i-1,i>}(X,Y) &= \delta(Y_{i-1}=y')\delta(Y_i=y) \\ g_{x,y,j}(X,Y) &= \delta(Y_j=y)\delta(X_j=x), \end{aligned}$$

where $\delta(.)$ is the indicator function. These features state that, given X and Y, $f_{y',y,<i-1,i>}=1$ if the state labels at positions $i-1$ and i of the sequence are y' and y, respectively. Otherwise its value is 0. If at position j the state label is y and the observed value is x then $g_{x,y,j}=1$; otherwise it is 0. They correspond to transition and emission probabilities in HMM. Clearly, if the probabilities are distributed uniformly, the feature values will be independent of each other and the Jacobian will be diagonal. On the contrary, if the distributions are far from uniform, their values will correlate with each other and the Jacobian will not be diagonal.

We implemented the GIS variants by replacing the L-BFGS optimization part in CRF++ (Kudo, 2006) [1] with PGIS, i.e., Eq. (30), and our extrapolation methods. We then used the HMM models described above to synthesize a sample of 1,000 sequences of length 25 as the independent data set and a sample of 1,000 sequences of length 50 as the dependent data set. Then we applied our implementation of PGIS, TJPGIS and CTJPGIS to train CRF models for each data set. The convergence rate is measured by the required number of forward-backward evaluations for CRF training to converge. This measure will be abbreviated as *iterations*. The termination condition for all trials was $\frac{|\mathcal{L}(\theta^{(t-1)})-\mathcal{L}(\theta^{(t)})|}{\mathcal{L}(\theta^{(t)})} < 10^{-6}$. The experiment was run on a Fedora 7 x86-64 machine with AMD Athlon 64 X2 3800+ CPU and 4GB RAM.

The convergence rate performance results are shown in Figure 4. The CPU time results are given in Table 2. The results show that, for both data sets, both TJPGIS and CTJPGIS converged much faster than PGIS, which was still far away from convergence after more than 100 iterations. More importantly, as predicted, CTJPGIS actually converged fastest for the independent data set while TJPGIS performed the best for the dependent one.

1. Available under LGPL from the following URL: `http://crfpp.sourceforge.net/`.



(a) Dependent features: not block diagonal (b) Independent features: block diagonal

Figure 4: Convergence rate comparison of PGIS,TJPGIS, and CTJPGIS for synthesized data sets with independent (a) and dependent features (b), respectively.

Data Set & Type		CTJPGIS	TJPGIS	PGIS
Dependent Features	CPU Time(sec)	32.01	**7.35**	> 295.14
(Not Block Diagonal)	Iteration	53	**10**	> 500
Independent Features	CPU Time(sec)	**5.02**	9.88	391.61
(Block Diagonal)	Iteration	**16**	32	1324

Table 2: Comparison of CPU time of PGIS, TJPGIS, and CTJPGIS for synthesized data sets with independent and dependent features.

5.5 Experimental Result: Entity Recognition

We evaluate the performance of our GIS variants with three large real-world data sets for entity recognition to further verify our claim: CoNLL-2000 chunking task Sang and Buchholz (2000), BioNLP/NLPBA-2004 bio-entity recognition task Kim et al. (2004), and BioCreative II gene mention tagging task Wilbur et al. (2007). CoNLL-2000 and BioCreative II are single type entity recognition tasks. The problem is to label each token in an input sentence as one of `B` (beginning of entity), `I` (in an entity), and `O` (outside an entity). BioNLP/NLPBA-2004 is a multiple type task, where each type `X` has its corresponding `B-X`, `I-X` and `O-X`. These data sets have been used in competitions. The performance was measured by the F-scores for the hold-out sets. The F-score is defined as $F = \frac{2PR}{P+R}$, where P is the precision and R the recall. Table 3 gives the details of these data sets. GIS is known to be prohibitively slow for large data sets (Malouf, 2002; Sha and Pereira, 2003). In this experiment, we will show that the triple jump extrapolation can accelerate GIS to a practical level, and that as our analysis predicts, componentwise extrapolation always outperforms global extrapolation and is the method of choice for these tasks.

To deal with these large data sets, we modified the base GIS algorithm into adaptive overrelaxed variant of GIS as described in (Salakhutdinov and Roweis, 2003). Since adaptive overrelaxed GIS

	CoNLL 2000	BioNLP/NLPBA 2004	BioCreative II
# of training sentences	8,936	20,546	15,000
# of tokens in training	211K	472K	$\sim 350K^*$
# of test sentences	2,012	4,260	5,000
# of tokens in test	49K	97K	$\sim 100K^*$
# of entity types	1	5	1
# of features/parameters†	1M	6M	7M
F-score†	$\sim 93\%$	$\sim 70\%$	$\sim 87\%$
Reference	Sha and Pereira (2003)	Settles (2004)	Kuo et al. (2007)

*BioCreative II did not provide standardized tokenization of the sentences.

†Data about # of features/parameters and F-scores are obtained from the given references.

Table 3: Three data sets used in the experiment of our GIS variants.

is also a fixed-point iteration method, the triple jump extrapolation is applicable. All GIS variants in this experiment were built on top of adaptive overrelaxed GIS, with the learning rate initialized from 1.8 and increased by ten percent at each iteration until the likelihood fails to improve. Also, to maintain numerical stability, we assigned $\theta_i = \mu \neq 0$ if the i-th feature does not appear in the training data, which implies that $\widetilde{Ef}_i = 0$ and we will have $\log 0$ in Eq. (30). We also used μ as the lower bound of the weight for the least informative features and assigned $\theta_i = \mu$ any time when $\lambda_i < \mu$. We set $\mu = -0.1$ for all of our GIS variants in this experiment. We also ran CRF++ with default settings to obtain the performance results of L-BFGS. We used a tight termination condition $\frac{|\mathcal{L}(\theta^{(t-1)}) - \mathcal{L}(\theta^{(t)})|}{\mathcal{L}(\theta^{(t)})} < 10^{-7}$ for all methods compared in this experiment, including L-BFGS, to ensure a fair comparison. The experiment was run on a Fedora 7 x86-64 Linux machine with AMD Athlon 64 X2 3800+ CPU and 4GB RAM.

The features used in these tasks concern word occurrences, word pairing, part-of-speech tags and their pairing, etc. They were adopted from the references given in Table 3. These references were selected because they described the best performing CRF systems in their corresponding competitions. Since the number of features is huge and only a small proportion of the features may be correlated with each other. Intuitively, we expect that the covariance matrix will be quite sparse and should be block diagonal. Therefore, we predict that componentwise extrapolation will converge faster than global extrapolation. We may apply a statistical test for diagonality of large dimensional covariance matrices (see Kapetanios, 2004) to rigorously justify our prediction. However, we did not perform such a test because existing tests are prohibitively expensive given the size of our matrix and they are not applicable to test block diagonality. Instead, we randomly sampled subsets of training sentences and examined the resulting small covariance matrices. We found that the matrices were block diagonal. Therefore, we maintain our prediction that componentwise extrapolation will be more effective in this task.

Figure 5 to 7 give the comparison of the rates of convergence of PGIS, TJPGIS, CTJPGIS and L-BFGS and Table 4 provides their performance data. We draw the following conclusions from the results.

- Both TJPGIS and CTJPGIS accelerate PGIS drastically for all three tasks by many orders of magnitude. The trials for PGIS were cut off because they were still too far away from convergence for all data sets.
- As predicted, CTJPGIS is more than three times as fast as TJPGIS for all three tasks.
- CTJPGIS can compete with L-BFGS, winning in two out of three tasks, in terms of both rate of convergence and CPU time.
- Both TJPGIS and CTJPGIS can achieve F-scores as good as L-BFGS and those reported in the literature (see Table 3).

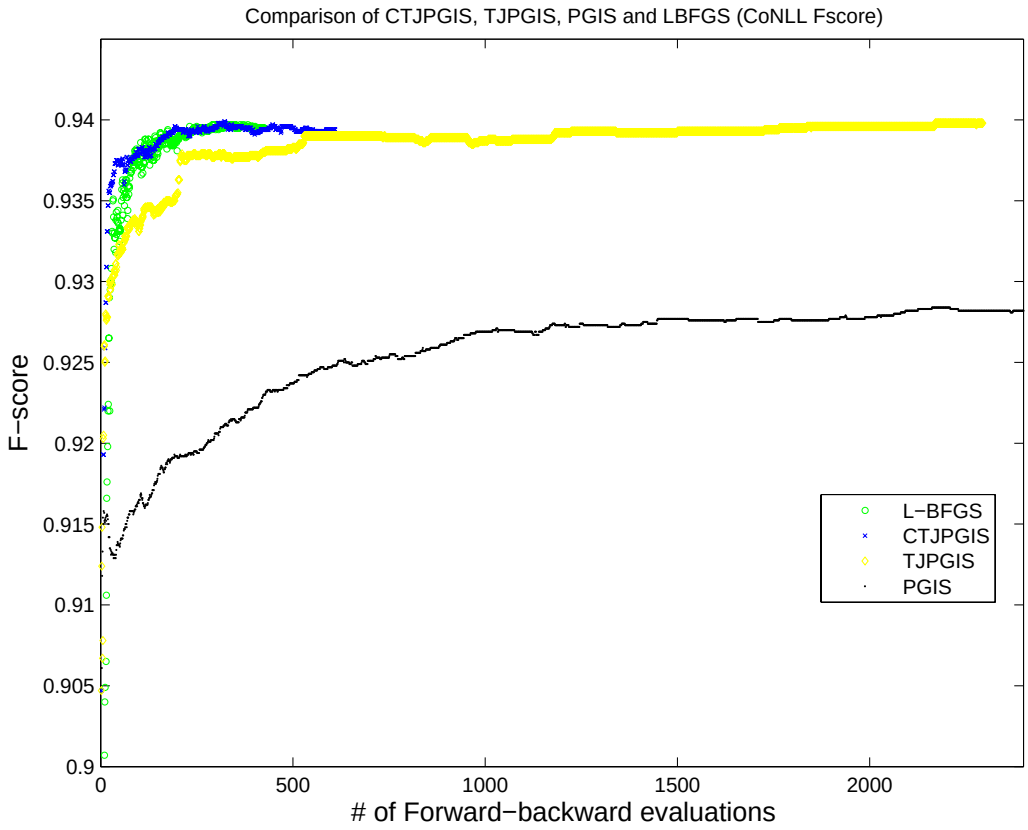


Figure 5: Comparison of F-scores of PGIS,TJPGIS, CTJPGIS, and L-BFGS as a function of the number of iterations for CoNLL-2000 data set.

6. Conclusions

In this paper, we verify that, when the componentwise rate of convergence is different from the global rate of convergence, componentwise extrapolation should be preferred. We show that the

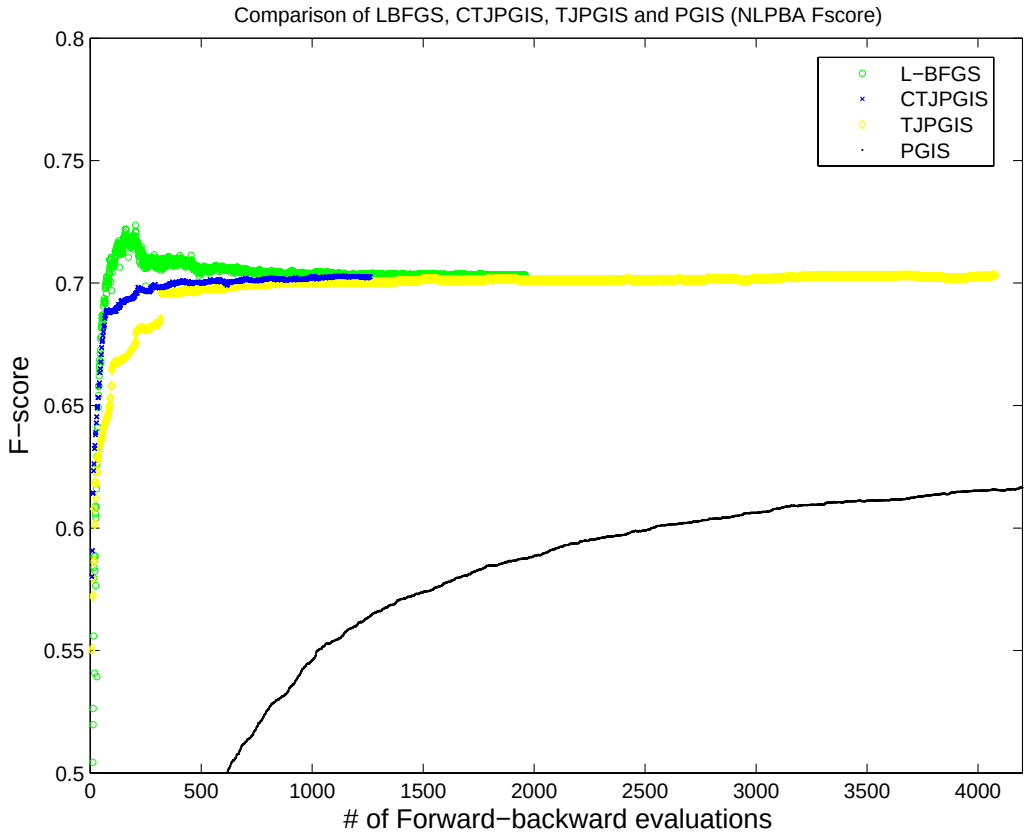


Figure 6: Comparison of F-scores of PGIS,TJPGIS, CTJPGIS, and L-BFGS as a function of the number of iterations for BioNLP/NLPBA-2004 data set.

componentwise rate and the global rate of convergence are different if the Jacobian of the fixed-point iteration mapping is diagonal or block diagonal.

We proceed to compare two simple mixtures of Gaussians models. One has a diagonal Jacobian and the other's is not. Experimental results with these models are consistent with our claim. We then analyze the Bayesian networks and identify analytical conditions for a general Bayesian network to have a block diagonal Jacobian. Based on the analysis results, we predict that for a particularly sparse data set with a large proportion of missing data, componentwise extrapolation should be preferred. The results are applied to the Alarm network and the semi-supervised Bayesian classifier model. Again, experimental results show that the componentwise triple jump method can quickly reach the optimum in a small number of iterations, consistent with our prediction.

Then we consider the GIS algorithm for training CRFs. We derive a penalized variant of GIS, PGIS, so that we can apply the triple jump extrapolation method. Our experimental results on synthesized data are consistent with our claim. Then we showed that the Jacobian of PGIS for large-scale entity recognition tasks is quite likely to be block diagonal and thus we predicted that CTJPGIS will perform the best in terms of convergence rate. Again, the results match our prediction.

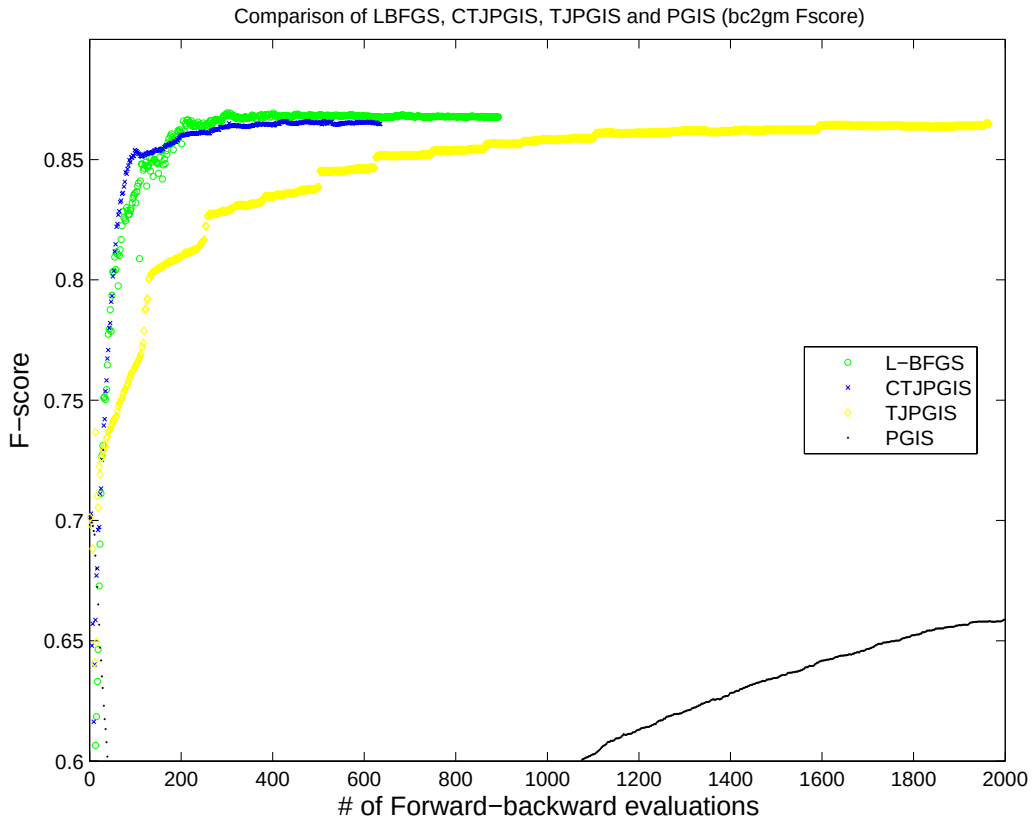


Figure 7: Comparison of F-scores of PGIS,TJPGIS, CTJPGIS, and L-BFGS as a function of the number of iterations for BioCreative II data set.

Data set		L-BFGS	CTJPGIS	TJPGIS	PGIS
CoNLL-2000	CPU Time (sec)	583	816	3001	>41463
	Iteration	427	619	2316	>30906
	Final F-score (%)	93.95	93.94	93.98	>93.35
BioNLP/NLPBA-2004	CPU time (sec)	63158	38800	132286	>162462
	Iteration	1961	1279	4117	>5161
	Final F-score (%)	70.33	70.26	70.32	>62.13
BioCreative II	CPU time (sec)	4615	3011	9114	>17656
	Iteration	895	639	1974	>3926
	Final F-score (%)	86.77	86.49	86.49	>69.30

Table 4: Performance comparison of PGIS,TJPGIS, CTJPGIS, and L-BFGS for CoNLL-2000, BioNLP/NLPBA-2004 and BioCreative II data sets.

Moreover, we also show that CTJPGIS can compete with L-BFGS, the de facto standard training algorithm for CRF training.

Our results suggest that when considering accelerating a bound optimization method with the triple jump extrapolation method, in the following cases, componentwise extrapolation should be applied.

- for Bayesian networks, check if the data set is sparse;
- for CRFs, check if the features are not correlated.

For other bound optimization methods, we can check the diagonality of the Jacobian to determine which extrapolation strategy will produce a better speedup.

Acknowledgements

This research was supported in part by the National Research Program in Genomic Medicine (NRPGM), National Science Council, Taiwan, under Grant No. NSC95-3112-B-001-017 (Bioinformatics Core Service), and in part by National Science Council, Taiwan under Grant No. NSC95-2221-E-001-038.

References

Richard L. Burden and Douglas Faires. *Numerical Analysis*. PWS-KENT Pub Co., 1988.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

Tim Hesterberg. Staggered aitken acceleration for EM. In *Proceedings of the Statistical Computing Section of the American Statistical Association*, Minneapolis, Minnesota, USA, August 2005.

Han-Shen Huang, Bo-Hou Yang, and Chun-Nan Hsu. Triple-jump acceleration for the EM algorithm. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 649–652, November 2005.

Mortaza Jamshidian and Robert I. Jennrich. Acceleration of the EM algorithm by using quasi-newton methods. *Journal of the Royal Statistical Society, , Series B*, 59(3):569–587, 1997.

George Kapetanios. On testing for diagonality of large dimensional covariance matrices, 2004. URL `http://ideas.repec.org/p/qmw/qmwecw/wp526.html`.

Jin-Dong Kim, Tomoko Ohta, Yashimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, pages 70–75, 2004.

Taku Kudo. CRF++: Yet another CRF toolkit, 2006. URL `http://chasen.org/~taku/software/CRF++/`.

Cheng-Ju Kuo, Yu-Ming Chang, Han-Shen Huang, Kuan-Ting Lin, Bo-Hou Yang, Yu-Shi Lin, Chun-Nan Hsu, and I-Fang Chung. Rich feature set, unification of bidirectional parsing and dictionary filtering for high f-score gene mention tagging. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pages 105–107, 2007.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference on Machine Learning (ICML'03)*, pages 282–289, 2001.

Thomas A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 44:226–233, 1982.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, 2002.

Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics. Wiley-Interscience, 1997.

Xiao-Li Meng and Donald B. Rubin. On the global and componentwise rates of convergence of the EM algorithm. *Linear Algebra and Its Applications*, 199:413–425, 1994.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Stuart Russell, John Binder, Daphne Koller, and Keiji Kanazawa. Local learning in probabilistic networks with hidden variables. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1146–1152, 1995.

Ruslan Salakhutdinov and Sam Roweis. Adaptive overrelaxed bound optimization methods. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, pages 664–671, 2003.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. On the convergence of bound optimization algorithms. In *Conference on Uncertainty in Artificial Intelligence (UAI'03)*, pages 509–516, 2003.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL-2000)*, pages 127–132, 2000.

Joseph L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman and Hall, New York, 1997.

Shayle R. Searle. *Matrix Algebra Useful For Statistics*. Wiley, New York, 1982.

Burr Settles. Biomedical named entity recognition using conditional random fields and novel feature sets. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, pages 104–107, 2004.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, the North American Chapter of the Association for Computational Linguistics (NAACL'03)*, pages 213–220, 2003.

John Wilbur, Larry Smith, and Lorrie Tanabe. Biocreative 2. gene mention task. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pages 7–16, 2007.

Appendix A. Proofs

Proof of Lemma 7:

$$\begin{aligned}\frac{\partial w_{ijk}}{\partial \theta_{ijk'}} &= -\frac{e^{\theta_{ijk}} e^{\theta_{ijk'}}}{(\sum_{k''} e^{\theta_{ijk''}})^2} = -w_{ijk} w_{ijk'} \\ \frac{\partial w_{ijk}}{\partial \theta_{ijk}} &= \frac{e^{\theta_{ijk}}}{\sum_{k''} e^{\theta_{ijk''}}} - \left(\frac{e^{\theta_{ijk'}}}{\sum_{k''} e^{\theta_{ijk''}}}\right)^2 \\ &= w_{ijk}(1 - w_{ijk}).\end{aligned}$$

■

Proof of Lemma 8:

We rewrite $P(y)$ for θ_{ijk} as:

$$\begin{aligned}P(y) &= \sum_{j',k'} P(u_{ij'}, x_{ik'}, y) \\ &= \sum_{j',k'} P(u_{ij'}) P(x_{ik'}|u_{ij'}) P(y|u_{ij'}, x_{ik'}) \\ &= \sum_{j',k'} P(u_{ij'}) w_{ij'k'} P(y|u_{ij'}, x_{ik'}).\end{aligned}$$

Note that $P(u_{ij'})$, the prior of $u_{ij'}$, is not a function of w_{ijk} based on the parameter independence assumption of Bayesian networks. Besides, $P(y|u_{ij'}, x_{ik'})$ is not a function of w_{ijk}. If there exists conflict between $(u_{ij'}, x_{ik'})$ and the corresponding values in y, $P(y|u_{ij'}, x_{ik'}) = 0$. Otherwise, $P(y|u_{ij'}, x_{ik'}) = \frac{P(u_{ij'}, x_{ik'}, y)}{P(u_{ij'}, x_{ik'})}$ in which both $P(u_{ij'}, x_{ik'}, y)$ and $P(u_{ij'}, x_{ik'})$ contain $w_{ij'k'}$ and thus $w_{ij'k'}$ can be eliminated.

Therefore, $\frac{\partial}{\partial \theta_{ijk}} P(y)$ is:

$$\begin{aligned}\frac{\partial}{\partial \theta_{ijk}} P(y) &= \sum_{j',k'} P(u_{ij'}) P(y|u_{ij'}, x_{ik'}) \frac{\partial}{\partial \theta_{ijk}} w_{ij'k'} \\ &= \sum_{k'} P(u_{ij}) P(y|u_{ij}, x_{ik'}) \frac{\partial}{\partial \theta_{ijk}} w_{ijk'}.\end{aligned}$$

Based on Lemma 7, the above equation can be further simplified:

$$\begin{aligned}
\frac{\partial}{\partial\theta_{ijk}}P(y) &= P(u_{ij})P(y|u_{ij},x_{ik})\frac{\partial}{\partial\theta_{ijk}}w_{ijk} + \\
&\quad \sum_{k'\neq k} P(u_{ij})P(y|u_{ij},x_{ik'})\frac{\partial}{\partial\theta_{ijk}}w_{ijk'} \\
&= P(u_{ij})P(y|u_{ij},x_{ik})w_{ijk}(1-w_{ijk}) - \\
&\quad \sum_{k'\neq k} P(u_{ij})P(y|u_{ij},x_{ik'})w_{ijk}w_{ijk'} \\
&= P(u_{ij})P(y|u_{ij},x_{ik})w_{ijk} - \\
&\quad w_{ijk}\sum_{k'} P(u_{ij})P(y|u_{ij},x_{ik'})w_{ijk'} \\
&= P(u_{ij},x_{ik},y) - w_{ijk}\sum_{k'} P(u_{ij},x_{ik'},y) \\
&= P(u_{ij},x_{ik},y) - w_{ijk}P(u_{ij},y). \qquad (31)
\end{aligned}$$

When u_{ij} d-separates the observations in $y-\{u_{ij}\}$ and X_i, Eq. (31) can be rewritten as:

$$\begin{aligned}
& P(u_{ij},x_{ik},y) - w_{ijk}P(u_{ij},y) \\
= \;& P(x_{ik}|u_{ij},y)P(u_{ij},y) - w_{ijk}P(u_{ij},y) \\
= \;& w_{ijk}P(u_{ij},y) - w_{ijk}P(u_{ij},y) = 0.
\end{aligned}$$

■

Proof of Theorem 9:

The first condition is straightforward. The second condition can also be proved by Lemma 8. From Eq. (25), we have

$$\begin{aligned}
& \frac{\partial^2}{\partial\theta_{i'j'k'}\partial\theta_{ijk}}\log P(y) \\
= \;& \frac{1}{P(y)}\frac{\partial}{\partial\theta_{i'j'k'}}\left(P(u_{ij},x_{ik},y) - w_{ijk}P(u_{ij},y)\right) \\
= \;& \frac{1}{P(y)}\frac{\partial}{\partial\theta_{i'j'k'}}P(u_{ij},x_{ik},y) - P(u_{ij},y)\frac{\partial}{\partial\theta_{i'j'k'}}w_{ijk} \\
& -w_{ijk}\frac{\partial}{\partial\theta_{i'j'k'}}P(u_{ij},y). \qquad (32)
\end{aligned}$$

We can consider $y' = y\cup\{u_{ij},x_{ik}\}$ as another observed training data, and $u_{i'j'}$ d-separates y' and $x_{i'k'}$. By Lemma 8, we obtain that $\frac{\partial}{\partial\theta_{i'j'k'}}P(u_{ij},x_{ik},y)$, the first term of the above equation, is 0. Similarly, $\frac{\partial}{\partial\theta_{i'j'k'}}P(u_{ij},y)=0$. Besides, Lemma 7 describes that $\frac{\partial}{\partial\theta_{i'j'k'}}w_{ijk}=0$ here. Therefore, Eq. (32) is also 0 under the second condition.

■

Proof of Lemma 10:

We start from

$$\frac{\partial^2 \log P(y)}{\partial\theta_{ijk'}\partial\theta_{ijk}} = \frac{\partial}{\partial\theta_{ijk'}}\left(P(u_{ij}, x_{ik}|y) - w_{ijk}P(u_{ij}|y)\right).$$

If u_{ij} and x_{ik} are exactly the observed values in y, $P(u_{ij}, x_{ik}|y)$ and $P(u_{ij}|y)$ are 1 and the above equation becomes:

$$\frac{\partial^2 \log P(y)}{\partial\theta_{ijk'}\partial\theta_{ijk}} = \frac{\partial}{\partial\theta_{ijk'}}\left(1 - w_{ijk}\right).$$

From Lemma 7, the second order partial derivative is $-w_{ijk}(1 - w_{ijk})$ if $k' = k$, is $w_{ijk}w_{ijk'}$ if $k' \neq k$, and is 0 otherwise.

■

Proof of Theorem 11:

Let $\tilde{p}(w)$ denote the number of times that the predicate in w occurs in the data set $\mathcal{D}$. For example, suppose $w_{ijk} = \Pr(X_i = x_{ik}|U_i = u_{ij})$, then $\tilde{p}(w_{ijk})$ is the number of times that $X_i = x_{ik}|U_i = u_{ij}$ occurs in $\mathcal{D}$.

$$\begin{aligned}
\log f(\mathcal{D}|\theta) &= \log \prod_y \prod_i \frac{e^{\theta_{ijk}}}{\sum_{k'} e^{\theta_{ijk'}}} \\
&= \sum_{ijk} \tilde{p}(w_{ijk})(\log\ e^{\theta_{ijk}} - \log \sum_{k'} e^{\theta_{ijk'}}) \\
&= \sum_{ijk} \tilde{p}(w_{ijk})(\theta_{ijk} - \log \sum_{k'} e^{\theta_{ijk'}}) \\
&= \sum_{ijk} \tilde{p}(w_{ijk})\theta_{ijk} - \sum_{ij} \tilde{p}(w_{ij}) \log \sum_{k'} e^{\theta_{ijk'}}\ .
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial \log\ f(\mathcal{D}|\theta)}{\partial\theta_{ijk}} &= \tilde{p}(w_{ijk}) - \frac{\partial \sum_{ij} \tilde{P}(w_{ij}) \log \sum_{k'} e^{\theta_{ijk}}}{\partial\theta_{ijk}} \\
&= \tilde{p}(w_{ijk}) - \sum_{ij} \tilde{p}(w_{ij}) \cdot \frac{e^{\theta_{ijk}}}{\sum_{k'} e^{\theta_{ijk'}}}\ .
\end{aligned}$$

For $\theta_{i'j'k'}$, $i' = i, j' = j, k' = k$ or $k' \neq k$, we have $\frac{\partial^2\ \log\ f(\mathcal{D}|\theta)}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} = e^{\theta_{ijk}} \cdot \frac{\theta_{ijk}}{\sum_{k'} e^{\theta_{ijk'}}}$; for $\theta_{i'j'k'}$, $i'j'k' \neq ijk$, $\frac{\partial^2\ \log\ f(\mathcal{D}|\theta)}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} = 0$. ■

Proof of Theorem 13:

In the first condition, if X_i is not observed, X_i is d-separated with y by the cluster node. Based on the first condition in Theorem 9, $\frac{\partial^2}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} \log P(y) = 0$.

In the second condition, if $X_{i'}$ is not observed, $X_{i'}$ is d-separated with $\{y, x_{ik}\}$ by the cluster node. Based on the second condition in Theorem 9, $\frac{\partial^2}{\partial\theta_{i'j'k'}\partial\theta_{ijk}} \log P(y) = 0$.

■

Proof of Corollary 14:

The three conditions are the complement of Theorem 13. The class variable and feature variables are probabilistically dependent because there are direct links between the class and feature variables. Therefore, we know from Lemma 8 that the derivative of $\log P(y)$ with respect to the parameters of the class variable is not guaranteed to be zero because y cannot be d-separated from the class variable. Accordingly, the first is true and the second condition can be easily verified. The third condition is true because X_i and $X_{i'}$ are not independent of y. ■

Proof of Theorem 15: From Eq. (26) and Eq. (27), the penalized log-likelihood function is:

$$\mathcal{L}(\theta) = \sum_k [\theta^T F(x_k, y_k) - \ln Z_\theta(x_k)] - \sum_i \frac{(\theta_i - \mu)^2}{2\sigma^2}$$

Note that $\ln Z(\theta) \leq Z(\theta)/Z(\psi) + \ln Z(\psi) - 1$ for any ψ, and $\exp \sum_i \theta_i g_i(x) \leq \sum_i g_i(x) \exp \theta_i + [1 - \sum_i g_i(x)]$, with $\sum_i g_i(x) \leq 1$. We have a lower bound:

$$\begin{aligned}
\mathcal{L}(\theta) &\geq \sum_{x,y} \tilde{p}(x,y) \sum_i \theta_i f_i(x,y) - \ln Z(\psi) + \sum_{x,y} p(x,y|\psi) \sum_i \frac{f_i(x,y)}{s} - \\
&\quad \sum_{x,y} p(x,y|\psi) \sum \frac{f_i(x,y)}{s} \exp[s(\theta_i - \psi_i)] - \sum_i \frac{(\theta_i - \mu)^2}{2\sigma^2} \\
&\equiv G(\theta, \psi)
\end{aligned}$$

Let M be the PGIS mapping. Applying the Taylor expansion of M around θ^*, we have:

$$\theta^{t+1} \approx \theta^* + M'(\theta^*)(\theta^t - \theta^*)$$

Near a local optimum, this matrix is related to the curvature of the function $G(\theta, \psi)$:

$$\lim_{\theta^t \to \theta^*} M'(\theta^t) = -[\nabla^2_G(\theta^*, \psi^*)][\nabla^2_G(\theta^*)]^{-1} \tag{33}$$

where the mixed partial derivatives and Hessian are defined as:

$$\begin{aligned}
\nabla^2_G(\theta^*, \psi^*) &\equiv \left[\frac{\partial^2 G(\theta, \psi)}{\partial\theta \partial\psi^T} \,\middle|\, \begin{array}{l} \theta = \theta^* \\ \psi = \theta^* \end{array} \right] \\
\nabla^2_G(\theta^*) &\equiv \left[\frac{\partial^2 G(\theta, \psi)}{\partial\theta \partial\theta^T} \,\middle|\, \begin{array}{l} \theta = \theta^* \\ \psi = \theta^* \end{array} \right]
\end{aligned}$$

Using these to compute second order statistics, we have:

$$\begin{aligned}
\nabla^2_G(\theta^*) &= -s \,\text{diag}[\bar{F}(\theta)] - \Lambda = -s\text{D}(\theta^*) - \Lambda \\
\nabla^2_G(\theta^*, \psi^*) &= s \,\text{diag}[\bar{F}(\theta)] - \\
&\quad \left[\sum_{x,y} p(x,y|\theta^*) F(x,y) F(x,y)^T - [\bar{F}(\theta^*)][\bar{F}(\theta^*)]^T \right] \\
&= s\text{D}(\theta^*) - \text{Cov}(\theta^*)
\end{aligned}$$

where Λ is

$$\frac{\partial^2 \sum_i \frac{(\theta_i-\mu)^2}{2\sigma^2}}{\partial\theta\partial\theta^T} = \begin{pmatrix} \frac{1}{\sigma^2} & & & 0 \\ & \frac{1}{\sigma^2} & & \\ & & \ddots & \\ 0 & & & \frac{1}{\sigma^2} \end{pmatrix} = \text{diag}(\frac{1}{\sigma^2}).$$

According to Eq. (33), Jacobian $M'(\theta^*)$ is of the form:

$$\begin{aligned} \left.\frac{\partial M(\theta)}{\partial\theta}\right|_{\theta=\theta^*} &= [s\text{D}(\theta^*) - \text{Cov}(\theta^*)][-s\text{D}(\theta^*) - \Lambda]^{-1} \\ &= [s\text{D}(\theta^*) - \Lambda][-s\text{D}(\theta^*) - \Lambda]^{-1} + \\ &\quad [\text{Cov}(\theta^*) + \Lambda][-s\text{D}(\theta^*) - \Lambda]^{-1} \\ &= I - \frac{1}{s}[\text{Cov}(\theta^*) + \Lambda][\text{D}(\theta^*) + \frac{1}{s}\Lambda]^{-1} \end{aligned}$$

■