Canonical Dual Approach to Binary Factor Analysis

Ke Sun[1], Shikui Tu[1], David Yang Gao[2], and Lei Xu[1]

[1] Department of Computer Science and Engineering,
Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China
{ksun,sktu,lxu}@cse.cuhk.edu.hk
[2] Department of Mathematics, Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061, USA
gao@vt.edu

Abstract. Binary Factor Analysis (BFA) is a typical problem of Independent Component Analysis (ICA) where the signal sources are binary. Parameter learning and model selection in BFA are computationally intractable because of the combinatorial complexity. This paper aims at an efficient approach to BFA. For parameter learning, an unconstrained binary quadratic programming (BQP) is reduced to a canonical dual problem with low computational complexity; for model selection, we adopt the Bayesian Ying-Yang (BYY) framework to make model selection automatically during learning. In the experiments, the proposed approach `cdual` shows superior performance. Another BQP approximation `round` is also good in model selection and is more efficient. Two other methods, `greedy` and `enum`, are more accurate in BQP but fail to compete with `cdual` and `round` in BFA. We conclude that a good optimization is essential in a learning process, but the key task of learning is not simply optimization and an over-accurate optimization may not be preferred.

1 Introduction

Binary Factor Analysis (BFA) explores latent binary structures of data. Unlike in clustering analysis where the observations are scattered around several uncorrelated centers, in BFA the cluster locations are correlated and represented by a binary vector with independent dimensions. From an information theoretic perspective, the observables can be traced to several independent binary random variables as information sources. Research on BFA has been conducted with wide applications. One stream has been focused on analysis of binary data [1] with the aid of Boolean Algebra. The broad variety of binary data, such as social research questionnaires, market basket data and DNA microarray expression profiles, gives this research enormous practical value. Another stream tries to discover binary factors in continuous data [2] [3] [4], taking advantage of the representational capacity of the underlying binary structure. The present work falls in the second category.

A general and difficult problem in BFA is the combinatorial complexity in the inference of a m-bit binary code $\boldsymbol{y}(\boldsymbol{x})$ or a 2^m-point posterior distribution

T. Adali et al. (Eds.): ICA 2009, LNCS 5441, pp. 346–353, 2009.

$p(\boldsymbol{y} \mid \boldsymbol{x})$ for each training sample $\boldsymbol{x}$. Past efforts in tackling this problem include applying the Markov Chain Monte Carlo (MCMC) methods [3], or restricting the model so that the posterior distribution has independent dimensions [2].

Another difficulty in BFA is to determine an appropriate length of the internal binary code $\boldsymbol{y}$. The traditional approach for model selection has to enumerate a set $\mathcal{K}$ and perform maximum likelihood (ML) learning for each candidate length $\dim(\boldsymbol{y}) \in \mathcal{K}$, then the optimal length is selected via minimizing an information criterion [5] [6]. This two-phase approach suffers from excessive computation due to the computational complexity of BFA. In the past decade efforts have also been made to determine a proper model scale during parameter learning. As a general framework for parameter learning and model selection, BYY harmony learning [7] [8] is capable to discard redundant structures during training. The paper [8] investigates machine learning versus optimization from the BYY perspective, where BFA is discussed as a special case. The paper [4] studies BFA under the BYY framework, where $p(\boldsymbol{y} \mid \boldsymbol{x})$ is assumed to be free of structure.

This paper considers the same BFA model as in [4] and [2]. In help of a canonical duality of BQP [9][10], we can compute efficiently for each training sample $\boldsymbol{x}_t$ a binary code $\boldsymbol{y}(\boldsymbol{x}_t)$. As learning proceeds, redundant binary dimensions are identified and discarded with a BYY learning algorithm [8]. A comparison among four BQP methods is presented. The proposed approach `cdual` is not the best in BQP optimization but presents superior performance in BFA learning. A relax-and-round method `round`, which is rather rough from an optimization perspective, is also good in model selection and provides a performance even better than the accurate BQP techniques.

The rest of this paper is organized as follows. Section 2 introduces BFA and a BYY learning algorithm. Section 3 imports the canonical duality theory to overcome the BQP computational bottleneck. Section 4 includes an experimental comparison among four BQP methods in BFA learning. Section 5 concludes.

2 Binary Factor Analysis

This paper studies the BFA model

$$q(\boldsymbol{y}) = \prod_{i=1}^{m} \theta_i^{(1+y_i)/2}(1-\theta_i)^{(1-y_i)/2}, \quad q(\boldsymbol{x} \mid \boldsymbol{y}) = G(\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{y} + \boldsymbol{c},\, \boldsymbol{\Sigma}), \qquad (1)$$

where $\boldsymbol{y} \in \{-1, 1\}^m$ is an internal binary code, $0 < \theta_i < 1$, $i = 1, 2, \ldots, m$, $\boldsymbol{x}$ is a continuous observation, $G(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Psi})$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Psi}$, $\boldsymbol{\Sigma}$ is a positive definite diagonal matrix. This model has been studied previously from different perspectives [11] [4] [2].

Within the BYY framework [7], another joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ describes the observations with $p(\boldsymbol{x})$ and the inference of binary codes with $p(\boldsymbol{y} \mid \boldsymbol{x})$. In this paper, $p(\boldsymbol{x})$ is chosen as $p(\boldsymbol{x}) = \sum_{t=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_t)/N$, where $\delta(.)$ is the Dirac delta function; $p(\boldsymbol{y} \mid \boldsymbol{x}) = \delta(\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{x}))$ is assumed to be free of structure, where $\hat{\boldsymbol{y}}(\boldsymbol{x})$ is derived through maximizing the harmony function [7] [8] such that

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\{-1,1\}^m}{\arg\max}\ \log q(\boldsymbol{x},\boldsymbol{y}) = \underset{\boldsymbol{y}\in\{-1,1\}^m}{\arg\min}\left\{\frac{1}{2}\boldsymbol{y}^T\boldsymbol{Q}_y\boldsymbol{y} - \boldsymbol{f}_y^T(\boldsymbol{x})\boldsymbol{y}\right\}, \quad (2)$$

$\boldsymbol{Q}_y = \boldsymbol{A}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{A}$, $\boldsymbol{f}_y(\boldsymbol{x}) = [\log\boldsymbol{\theta} - \log(\boldsymbol{1}-\boldsymbol{\theta})]/2 + \boldsymbol{A}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{c})$, $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_m)^T$.

By definition, the harmony function is

$$H(p\,||\,q) = \int p(\boldsymbol{x})p(\boldsymbol{y}\,|\,\boldsymbol{x})\log q(\boldsymbol{y})q(\boldsymbol{x}\,|\,\boldsymbol{y})\,d\boldsymbol{y}\,d\boldsymbol{x} = \frac{1}{N}\sum_{t=1}^{N}\widetilde{H}(\boldsymbol{x}_t,\hat{\boldsymbol{y}}(\boldsymbol{x}_t),\boldsymbol{\Theta}),$$

$$\widetilde{H}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}) = \sum_{i=1}^{m}\left[\frac{1+y_i}{2}\log\theta_i + \frac{1-y_i}{2}\log(1-\theta_i)\right] + \log G\,(\boldsymbol{x}\,|\,\boldsymbol{A}\boldsymbol{y}+\boldsymbol{c},\boldsymbol{\Sigma})\,. \quad (3)$$

BYY harmony learning with automatic model selection (BYY-AUTO) is implemented by maximizing $H(p\,||\,q)$ on the training set. Starting from a large initial coding length, the gradient flow of $H(p\,||\,q)$ may either *push θ_i to 0 or 1 when y_i turns deterministic*, or *push the ratio $||\boldsymbol{A}_i||_2^2/\sqrt{\boldsymbol{A}_i^T\boldsymbol{\Sigma}\boldsymbol{A}_i}$ ($\boldsymbol{A}_i$ is the i'th column of $\boldsymbol{A}$) to 0 when $\boldsymbol{A}_i y_i$ is flooded by noise*. In both these cases the i'th bit of $\boldsymbol{y}$ is identified as redundant and is discarded while the learning proceeds. The learning process is sketched in Algorithm 1.

Algorithm 1. Free structure BYY-AUTO learning algorithm for BFA

Input : A set $\{\boldsymbol{x}_t\}_{t=1}^N \subset \Re^n$ of observations
Output: An estimated binary coding length $\dim(\boldsymbol{y})$; $\boldsymbol{\Theta} = \{\boldsymbol{\theta},\boldsymbol{A},\boldsymbol{c},\boldsymbol{\Sigma}\}$

1 Initialize $\dim(\boldsymbol{y})$ with a large integer m_0; $\boldsymbol{\Theta}_0 = \{\boldsymbol{\theta}_0,\boldsymbol{A}_0,\boldsymbol{c}_0,\boldsymbol{\Sigma}_0\}$;
2 **repeat**
3 Take $\mathcal{X}_e \subset \{\boldsymbol{x}_t\}_{t=1}^N$ sequentially or through a sampling algorithm;
4 Encode $\mathcal{X}_e$ into $\{\hat{\boldsymbol{y}}(\boldsymbol{x}) : \boldsymbol{x}\in\mathcal{X}_e\}$ with a binary encoder;
5 Update $\boldsymbol{\Theta}$ along the gradient flow of $\sum_{\boldsymbol{x}\in\mathcal{X}_e}\widetilde{H}(\boldsymbol{x},\hat{\boldsymbol{y}}(\boldsymbol{x}),\boldsymbol{\Theta})$;
6 **if** $\theta_i < \epsilon$ *or* $\theta_i > 1-\epsilon$ *or* $||\boldsymbol{A}_i||_2^2 < \delta\sqrt{\boldsymbol{A}_i^T\boldsymbol{\Sigma}\boldsymbol{A}_i}$ **then**
7 Discard the i'th dimension of $\boldsymbol{y}$; update $\boldsymbol{\Theta}$ accordingly;
8 **until** $H(p\,||\,q)$ *has reached convergence* ;

In the experiments we fix $\epsilon = 0.1$, $\delta = 2$, $|\mathcal{X}_e| = N$, $m_0 = 2m^\star - 1$, where $m^\star = \dim(\boldsymbol{y}^\star)$ is the "true" binary dimension in synthetic data generation.

3 Canonical Dual Approach to Binary Encoding

In BFA, a binary encoder $\hat{\boldsymbol{y}} : \{\boldsymbol{x}_t\}_{t=1}^N \to \{-1,1\}^m$ is usually employed so that a function can be computed numerically or a mapping, such as $p(\boldsymbol{y}\,|\,\boldsymbol{x},\boldsymbol{\Theta})$, can be regularized with the encoding results. In the context here, we need such an encoding, as in Eq. (2) or line 4 in Algorithm 1, to maximize $H(p\,||\,q)$ in Eq. (3) by a gradient-based optimization. Eq. (2) is a BQP that falls in NP-hard. The BFA-specific formula of $\boldsymbol{Q}_y$ and $\boldsymbol{f}_y(\boldsymbol{x})$ can not make the problem easier. An approximation is required to avoid the combinatorial complexity.

Both exact and heuristic approaches to BQP have been widely studied in the literature of optimization [12]. Recently Gao et al. have constructed a pair of *canonical dual problems* for BQP [9] [10] with zero duality gap. The solution $\bar{\boldsymbol{\zeta}}$ of the *canonical dual* problem

$$(\mathcal{P}^d): \quad \max_{\boldsymbol{\zeta}>\mathbf{0}} \left\{ P^d(\boldsymbol{\zeta}) = -\frac{1}{2}\boldsymbol{f}_y^T(\boldsymbol{x}) \left[\boldsymbol{Q}_y + diag(\boldsymbol{\zeta})\right]^{-1} \boldsymbol{f}_y(\boldsymbol{x}) - \frac{1}{2}\boldsymbol{e}^T\boldsymbol{\zeta} \right\}, \tag{4}$$

if exists, will lead to a solution $\hat{\boldsymbol{y}}(\boldsymbol{x}) = (\boldsymbol{Q}_y + diag(\bar{\boldsymbol{\zeta}}))^{-1}\boldsymbol{f}_y(\boldsymbol{x})$ of Eq. (2), where $\boldsymbol{e} = (1, 1, \ldots, 1)^T$. In contrast to the primal BQP, $\mathcal{P}^d$ is a constrained convex optimization problem that can be handled much more efficiently. Algorithm 2 employs a gradient descent to solve BQP through solving $\mathcal{P}^d$ in Eq. (4).

Algorithm 2. $\min_{y\in\{-1,1\}^m} \left\{ \frac{1}{2}\boldsymbol{y}^T\boldsymbol{Q}\boldsymbol{y} - \boldsymbol{f}^T\boldsymbol{y} \right\}$ via max its canonical dual

1 Normalize $\boldsymbol{Q}^{new} = \boldsymbol{Q}^{old}/tr(\boldsymbol{Q}^{old})$, $\boldsymbol{f}^{new}(\boldsymbol{x}) = \boldsymbol{f}^{old}(\boldsymbol{x})/tr(\boldsymbol{Q}^{old})$;
2 Pre-process $\boldsymbol{Q}^{new} = \boldsymbol{Q}^{old} + \boldsymbol{\Lambda}_Q$, $\boldsymbol{\Lambda}_Q$ is diagonal (optional);
3 Initialize $\boldsymbol{\zeta} = \max(-\boldsymbol{Q}\boldsymbol{e} - \boldsymbol{f}, \boldsymbol{Q}\boldsymbol{e} + \boldsymbol{f} - 2diag(\boldsymbol{Q}))$;
4 **for** *epoch* $\leftarrow 1$ **to** *max_epochs* **do**
5 $\quad \boldsymbol{y} = (\boldsymbol{Q} + diag(\boldsymbol{\zeta}))^{-1}\boldsymbol{f}$;
6 $\quad \bigtriangledown = (\boldsymbol{y} \circ \boldsymbol{y} - \boldsymbol{e})/2$;
7 $\quad$ **if** $||\bigtriangledown||_\infty <$ *Threshold* **then** break;
8 $\quad \boldsymbol{\zeta} = \boldsymbol{\zeta} + \gamma\bigtriangledown$ ($\gamma > 0$ is a small learning rate);
9 Round $\boldsymbol{y}$ to $\{-1, 1\}^m$; return $\boldsymbol{y}$.

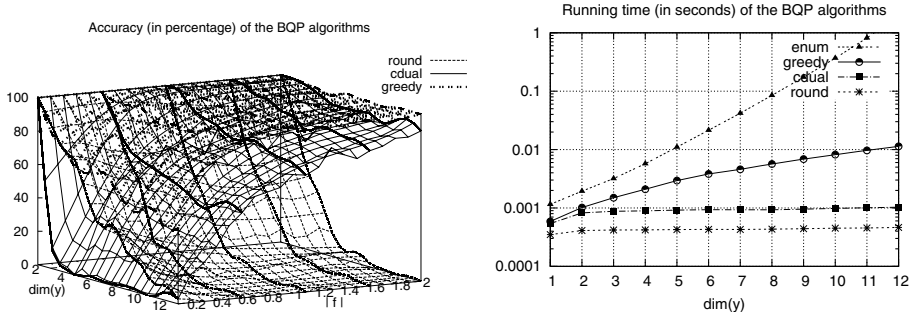
In the experiments, max_epochs $= 50$, *Threshold* $= 0.5$, $\gamma = 0.02$. *Step 2 is a classical trick [12] based on* $y_i^2 \equiv 1$ *but not adopted in the experiments.*

Table 1. Algorithms for solving the BQP in Eq. (2)

Name	Description
`enum`	exhaustively enumerate $\boldsymbol{y} \in \{-1, 1\}^m$, which was used in BFA [4]
`greedy`	the greedy BQP algorithm on page 203 [12]
`cdual`	the canonical dual approach to BQP (Algorithm 2 in this paper)
`round`	round $\tilde{\boldsymbol{y}} = \boldsymbol{Q}_y^{-1}\boldsymbol{f}_y(\boldsymbol{x})$ to the nearest binary vector in $\{-1, 1\}^m$, which was proposed (Table II, page 836 [11]) for BFA learning under the name "fixed posteriori approximation"

Figure 1 shows the accuracy and efficiency of the BQP algorithms listed in Table 1[1]. `round` is fastest but its performance degenerates greatly as $\dim(\boldsymbol{y})$ increases; `greedy` is most accurate among {`round`, `cdual`, `greedy`} but suffers from $O(\dim^3(\boldsymbol{y}))$ computation [12]; `cdual` is less accurate than `greedy` but is

[1] All experiments in this paper are implemented with GNU Octave 3.0.3 on a Intel Core 2 Duo 2.13GHz with 1GB RAM running FreeBSD 7.0.



(a) Percentage of correct solutions over a 10× 10 ($\dim(\boldsymbol{y})\times||\boldsymbol{f}_y(\boldsymbol{x})||_2$) grid of configurations (b) Computation cost in seconds (note the time axis is in log-scale)

Fig. 1. Performance (out of 100 runs) of the BQP algorithms with $tr(\boldsymbol{Q}) = 1.0$

much more efficient. As $||\boldsymbol{f}_y(\boldsymbol{x})||_2$ turns small, `round` becomes a little more accurate while the error of `cdual` and `greedy` raises up significantly.

If $||\boldsymbol{f}_y(\boldsymbol{x})||_2$ is small enough such that $||\boldsymbol{Q}_y^{-1}\boldsymbol{f}_y(\boldsymbol{x})||_\infty < 1$, then $\bigtriangledown P^d|_{\boldsymbol{\zeta}=\mathbf{0}} < \mathbf{0}$. From the convexity of $P^d(\zeta)$ on $\Re^{m+}$, the dual solution $\bar{\boldsymbol{\zeta}} > 0$ does not exist. This explains the failure of `cdual` on small $||\boldsymbol{f}_y(\boldsymbol{x})||_2$. We further consider its impact on BFA learning. In Algorithm 1, dimension i will be deducted if $|\theta_i - 0.5|$ is large enough, hence $\boldsymbol{\theta}$ is in a small neighbourhood around $0.5\boldsymbol{e}$ and $||\boldsymbol{f}_y(\boldsymbol{x})||_2^2 = ||\boldsymbol{A}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{c}) + [\log\boldsymbol{\theta} - \log(\mathbf{1}-\boldsymbol{\theta})]/2||_2^2$ is a convex function minimized around $\boldsymbol{x} = \boldsymbol{c}$. A small $||\boldsymbol{f}_y(\boldsymbol{x})||_2$ is due to samples lying between the 2^m representative clusters in BFA. `cdual` is not accurate on these samples.

4 Experiments

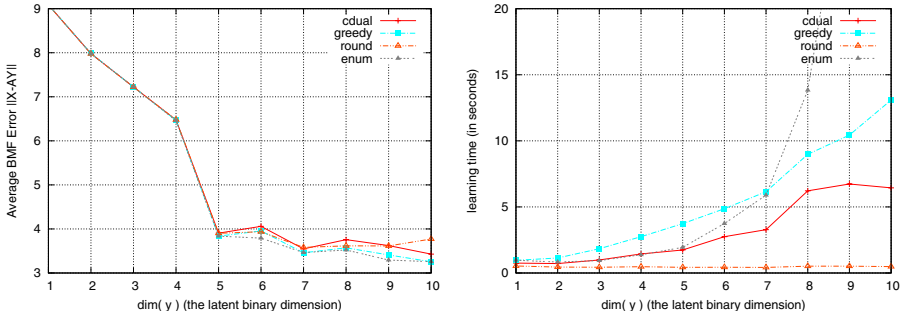
This section compares `enum`, `greedy`, `cdual` and `round` in BFA with synthetic data generated according to Eq. (1). Because of space limitation, we concentrate on the case where $\dim(\boldsymbol{x}) = 10$, $\boldsymbol{y}$ evenly taking values from the 2^{m-1} points $\{\mathbf{y} \in \{-1,1\}^m : \mod[\sum_{i=1}^m (y_i+1)/2,\, 2] = 0\}$[2], $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}$, $\boldsymbol{Q}$ is orthogonal, $\boldsymbol{\Lambda} = diag(\lambda_1, \lambda_2, \ldots, \lambda_m)$, λ_i uniformly distributed over the interval $(1,2)$ so that the scale $||\boldsymbol{A}_i||_2$ in each binary dimension does not vary too much, and $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$. Three aspects that may affect the learning performance are investigated: the true binary dimension($\dim(\boldsymbol{y}^\star)$), the sample size($N$) and the noise level($\sigma$).

4.1 Binary Matrix Factorization with Fixed Dimension

We fix the binary dimension by skipping line $6 \sim 7$ in Algorithm 1 and study its performance on the binary matrix factorization (BMF) $\boldsymbol{X}_{n\times N} = \boldsymbol{A}_{n\times m}\boldsymbol{Y}_{m\times N}$,

[2] This is a subset of $\{-1,1\}^m \subset \Re^m$ that can only be well separated with $\geq m$ hyperplanes. Hence the "true" binary dimension is m. It is chosen instead of $\{-1,1\}^m$ to simulate data in real world where not all 2^m binary encodings are valid or observed. A comparison between data generated with this 2^{m-1}-point subset and $\{-1,1\}^m$ is nevertheless interesting but omitted here for saving space.

where $\boldsymbol{A}$ is real and $\boldsymbol{Y} \in \{-1, 1\}^{m \times N}$. Such a factorization differs from the classical BMF in that the matrix to be factored is continuous rather than boolean and is different from [3] in the factorization form. It may be useful in recovering binary signals from continuous observations. For example, a noisy binary image Y can be reverted after rotation or scaling. Figure 2 presents the BMF error and learning time over $\dim(\boldsymbol{y})$. The training error gets an order `round` > `cdual` > `greedy` > `enum` while the running time is in a reverse order. When $\dim(\boldsymbol{y})$ is large, `round` and `cdual` is not as accurate as the others because of their deteriorated BQP accuracy. As a trade-off they are much faster. Training time of `round` appears to be constant over $\dim(\boldsymbol{y})$. `greedy` has avoided the exponential complexity of `enum` but still requires huge computation on a large $\dim(\boldsymbol{y})$. To sum up, `cdual` and `greedy` are recommended for the BMF task.



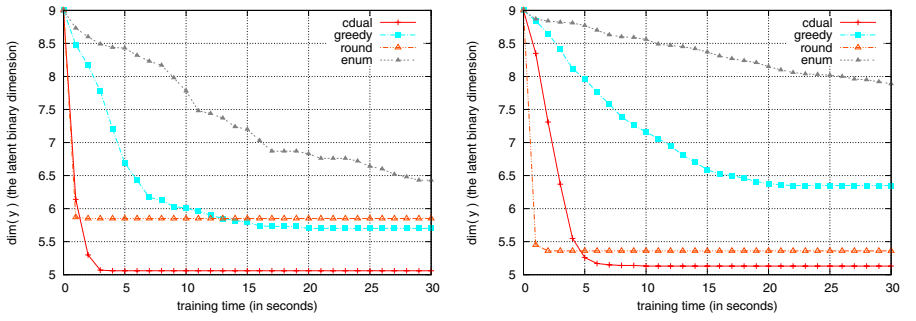
(a) BMF error $||X - AY||_2$ over $\dim(\boldsymbol{y})$ (b) Learning time(in seconds) over $\dim(\boldsymbol{y})$

Fig. 2. BMF when $\dim(\boldsymbol{x}) = 10$, $\dim(\boldsymbol{y}^\star) = 5$, $N = 100$, $\sigma = 0.3$, $\sum_{t=1}^{N} \boldsymbol{x}_t = \boldsymbol{0}$

4.2 Model Selection on Synthetic Data

In Algorithm 1, $\dim(\boldsymbol{y})$ is initialized large enough and deducted during learning. Since computational overhead arises when $\dim(\boldsymbol{y})$ is large, how soon this deduction stops determines the learning efficiency. With a true binary dimension $\dim(\boldsymbol{y}^\star) = 5$ and the learner's $\dim(\boldsymbol{y})$ initialized to 9, Figure 3 shows the average dimension deduction curve after 100 independent runs for $\sigma \in \{0.1, 0.3\}$. Two observations are made. (a) the convergence speed of $\dim(\boldsymbol{y})$ is in the same order as the BQP speed in Figure 1(b). Convergence slows down as the noise level increases. (b) `cdual` is robust to noise and yields the best accuracy; `enum` and `greedy` overestimates the model scale; `round` also shows a slight tendency of overestimation. The over/under estimation is controlled by the threshold δ and is related to $|\mathcal{X}_e|$ in Algorithm 1. They are both fixed in this paper for brevity.

Consider one binary dimension $\boldsymbol{y}_i$ with a small $||\boldsymbol{A}_i||_2$ and big noise. Maximizing $H(p \,||\, q)$ may further shrink $||\boldsymbol{A}_i||_2^2 / \sqrt{\boldsymbol{A}_i^T \boldsymbol{\Sigma} \boldsymbol{A}_i}$ to achieve model selection. The error of `cdual` on the samples lying among the 2^m representative clusters forms a natural regularization to dimension deduction. An over-accurate binary encoder does not have this type of regularization therefore tends to overestimation. Similar cases may arise in clustering. A carefully designed optimization



(a) $\dim(\boldsymbol{x}) = 10, \dim(\boldsymbol{y}^\star) = 5, \sigma = 0.1, N = 50$ (b) $\dim(\boldsymbol{x}) = 10, \dim(\boldsymbol{y}^\star) = 5, \sigma = 0.3, N = 50$

Fig. 3. Dimension deduction over learning time (average of 100 runs)

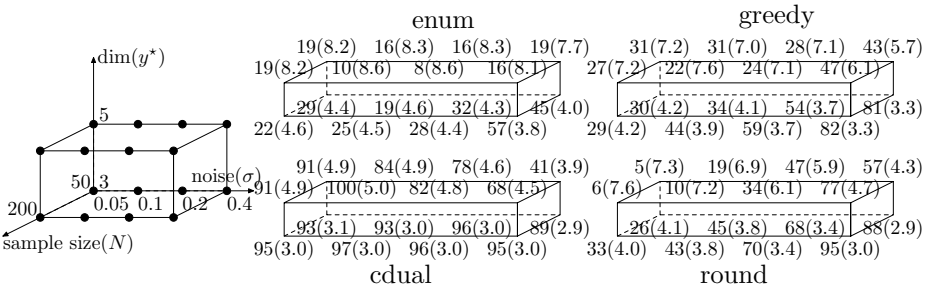


Fig. 4. "$\kappa(\overline{\dim(\boldsymbol{y})})$" out of 100 independent runs for each configuration tetrad $(\dim(\boldsymbol{x}), \dim(\boldsymbol{y}^\star), N, \sigma) \in \{10\} \times \{3, 5\} \times \{50, 200\} \times \{0.05, 0.1, 0.2, 0.4\}$, where κ is the number of correctly estimated $\dim(\boldsymbol{y})$ and $\overline{\dim(\boldsymbol{y})}$ is the average $\dim(\boldsymbol{y})$

algorithm which is not so accurate at the cluster boundaries may be good for model selection. As a sub-procedure, an optimization should be customized for learning instead of being isolated and implemented as accurately as possible.

Figure 4 shows the percentage κ of correctly estimated $\dim(\boldsymbol{y})$ and the average resulting $\dim(\boldsymbol{y})$ on a $2 \times 2 \times 4$ configuration grid. According to the experiments κ is sensitive to the threshold δ which is set to 2 here. Generally the performance degrades as N goes small or $\dim(\boldsymbol{y}^\star)$ goes large. In the batch algorithm where $|\mathcal{X}_e| = N$, the resulting model scale tends to be smaller as σ increases. Therefore κ may increase with σ during overestimation, as in the case of {`enum`, `greedy`, `round`}. `cdual` is the most robust and shows the best performance in nearly all configurations. `round` is also good especially when σ is large. Moreover, they outperform `greedy` and `enum` considerably in computational cost.

5 Concluding Remarks

The combinatorial complexity in BFA has been avoided through a canonical dual approach and the ML training enumeration in model selection has been avoided

by the BYY harmony learning. Among the algorithms investigated, `cdual` provides the best overall performance; `round` is comparably accurate on large noise and is much more efficient; `greedy` and `enum` are both inaccurate and time-consuming. A good optimization is crucial in learning. Learning, however, is not simply optimization. It includes an essential task to select a hierarchy of structures and a proper level in this hierarchy, which is difficult on small sample size and may get promoted with a customized optimization. A more detailed discussion on BFA learning and optimization will follow in subsequent works.

Acknowledgement

The work described in this paper was fully supported by a grant from the Research Grant Council of the Hong Kong SAR (Project No: CUHK4177/07E).

References

1. Keprt, A., Snásel, V.: Binary factor analysis with help of formal concepts. In: Snásel, V., Belohlávek, R. (eds.) CLA. CEUR Workshop Proceedings, CEUR-WS.org., vol. 110, pp. 90–101 (2004)
2. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) NIPS, pp. 1345–1352. MIT Press, Cambridge (2007)
3. Meeds, E., Ghahramani, Z., Neal, R.M., Roweis, S.T.: Modeling dyadic data with binary latent factors. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) NIPS, pp. 977–984. MIT Press, Cambridge (2007)
4. An, Y., Hu, X., Xu, L.: A comparative investigation on model selection in binary factor analysis. Journal of Mathematical Modelling and Algorithms 5 (4), 447–473 (2006)
5. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 19 (6), 716–723 (1974)
6. Schwarz, G.: Estimating the Dimension of a Model. The Annals of Statistics 6 (2), 461–464 (1978)
7. Xu, L.: A unified learning scheme: Bayesian-Kullback Ying-Yang machines. In: Touretzky, D.S., Mozer, M., Hasselmo, M.E. (eds.) NIPS, pp. 444–450. MIT Press, Cambridge (1995); A preliminary version in Proc. ICONIP 1995, Beijing, pp. 977–988 (1995)
8. Xu, L.: Machine learning problems from optimization perspective. A special issue for CDGO 2007, Journal of Global Optimization (in press, 2008)
9. Gao, D.Y.: Solutions and optimality criteria to box constrained nonconvex minimization problems. Journal of Industrial and Management Optimization 3 (2), 293–304 (2007)
10. Fang, S.C., Gao, D.Y., Shue, R.L., Wu, S.Y.: Canonical Dual Approach for Solving 0-1 Quadratic Programming Problems. Journal of Industrial and Management Optimization 4 (1), 125–142 (2008)
11. Xu, L.: BYY harmony learning, independent state space and generalized APT financial analysis. IEEE Transactions on Neural Networks 12 (4), 822–849 (2001)
12. Merz, P., Freisleben, B.: Greedy and Local Search Heuristics for Unconstrained Binary Quadratic Programming. Journal of Heuristics 8 (2), 197–213 (2002)