HOD process. Therefore, there is a tradeoff between memory and computing time. HOD provides a direct solution for the learning algorithm. In comparison, tuning the MC algorithm to provide lower error rates by increasing the number of samples will result in less iterations to reach convergence, but that will come at the expense of a higher computation time. In the end, the time needed to attain smaller error values makes the decimation method a better choice.

REFERENCES

[1] D. Ackley, G. Hinton, and T. Sejnowski, "A learning algorithm for Boltzmann machines," *Cogn. Sci.*, vol. 9, pp. 147–169, 1985.

[2] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, 2nd ed. New York: Wiley, 1989.

[4] L. Saul and M. Jordan, "Learning in Boltzmann trees," *Neural Comput.*, vol. 6, no. 6, pp. 1174–1184, 1994.

[5] S. Rüger, "Decimatable Boltzmann machines for diagnosis: Efficient learning and inference," in *Proc. World Congr. Sci. Comput. Model. Appl. Math.*, 1997, pp. 319–324.

[6] S. Rüger, A. Weinberger, and S. Wittchen, "Decimatable Boltzmann machines vs. Gibbs sampling," Informatik, Technische Universität, Berlin, Germany, Tech. Rep. 96-29, 1996.

[7] E. Farguell, E. Gómez-Ramírez, and F. Mazzanti, "Increasing Boltzmann machine learning speed and accuracy with the high order decimation method," *Res. Comput. Sci.*, vol. 21, pp. 3–12, Nov. 2006, CIC-IPN.

[8] E. Farguell, F. Mazzanti, and E. Gómez-Ramírez, *Boltzmann Machines Learning Using High Order Decimation*, ser. Studies in Fuzziness and Soft Computing. Berlin, Germany: Springer-Verlag, 2007, vol. 208, pp. 21–42.

[9] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van De Welde, W. Wenzel, J. Wnek, and J. Zhang, "The MONK's problems: A performance comparison of different learning algorithms," Comput. Sci. Dept., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-91-197, 1991 [Online]. Available: http://robots.stanford.edu/papers/thrun.MONK.html

[10] T. Sejnowski, "High-order Boltzmann machines," in *Proc. AIP Conf. 151 Neural Netw. Comput.*, Snowbird, Utah, E.U.A., 1987, pp. 398–403.

[11] H. Chen, P. Fleury, and A. Murray, "Continuous-valued probabilistic behavior in a VLSI generative model," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 755–770, May 2006.

[12] F. Albizuri, A. d'Anjou, M. Graña, F. Torrealdea, and M. Hernández, "The high-order Boltzmann machine: Learned distribution and topology," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 767–770, May 1995.

[13] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1991, Santa Fe Institute.

[14] S. Kullback, *Information Theory and Statistics*, 2nd ed. New York: Wiley, 1959.

[15] M. Graña, A. d'Anjou, F. Albizuri, M. Hernández, F. Torrealdea, A. de la Hera, and A. González, "Experiments of fast learning with high order Boltzmann machines," *Appl. Intell.*, vol. 7, no. 4, pp. 287–303, 1997.

[16] C. H. Park, H. Song, and K. Park, "Existence and classification of Hadamard matrices," in *Proc. 4th Int. Conf. Signal Process.*, 1998, pp. 117–121.

[17] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley-Interscience, 2001.

[18] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[19] L. Prechelt, "Proben1—A set of benchmarks and benchmarking rules for neural network training algorithms," Fakult fr Informatik, Universität Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, Sep. 1994.

[20] D. Klahr and R. Siegler, *The Representation of Children's Knowledge*. New York: Academic, 1978, pp. 61–116.

[21] C. Matheus and L. Rendell, "Constructive induction on decision trees," in *Proc. 11th Int. Joint Conf. Artif. Intell.*, Detroit, MI, 1989, pp. 645–650.

[22] M. Noordewier, G. Towell, and J. Shavlik, "Training knowledge-based neural Networks to recognize genes in DNA sequences," in *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufmann, 1991, vol. 3.

[23] M. Stone, "Asymptotics for and against cross-validation," *Biometrika*, vol. 64, pp. 29–35, 1977.

[24] X. Ma and K. Likharev, "Global reinforcement learning in neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 573–577, Mar. 2007.

[25] Enginyeria i Arquitectura La Salle, Universitat Ramon Llul, "A C implementation for a high order decimation BM," Barcelona, Spain [Online]. Available: http://www.salle.url.edu/~efarguell/HOD_BM.tar.gz

Matrix-Variate Factor Analysis and Its Applications

Xianchao Xie, Shuicheng Yan, James T. Kwok, and Thomas S. Huang

***Abstract*—Factor analysis (FA) seeks to reveal the relationship between an observed vector variable and a latent variable of reduced dimension. It has been widely used in many applications involving high-dimensional data, such as image representation and face recognition. An intrinsic limitation of FA lies in its potentially poor performance when the data dimension is high, a problem known as curse of dimensionality. Motivated by the fact that images are inherently matrices, we develop, in this brief, an FA model for matrix-variate variables and present an efficient parameter estimation algorithm. Experiments on both toy and real-world image data demonstrate that the proposed matrix-variant FA model is more efficient and accurate than the classical FA approach, especially when the observed variable is high-dimensional and the samples available are limited.**

***Index Terms*—Conditional expectation maximization (EM), face recognition, factor analysis (FA), matrix.**

I. INTRODUCTION

Factor analysis (FA) [14] is a canonical statistical method for modeling the covariance structure of high-dimensional data. It has been widely used for data representation and object recognition. For example, Hinton *et al.* [8] used a mixture of factor analyzers for the recognition of handwritten digits. Tipping and Bishop [4], [16] proposed a special FA model, called probabilistic principal component analysis (PPCA), and there are many other generalizations for data representation and visualization. In the vision literature, probabilistic visual learning and Bayesian face recognition [13] are also closely related to FA.

Several methods have been proposed to perform parameter estimation for the FA model [1], [5]. Though these methods often guarantee a numerically correct and stable solution, statistically speaking, the

Manuscript received August 19, 2007; revised February 12, 2008 and June 4, 2008; accepted August 5, 2008. First published September 26, 2008; current version published October 8, 2008.

X. Xie is with the Department of Statistics, Harvard University Science Center, Cambridge, MA 02138-2901 USA (e-mail: xxie@fas.harvard.edu).

S. Yan is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore (e-mail: eleyans@nus.edu.sg).

J. T. Kwok is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: jamesk@cse.ust.hk).

T. S. Huang is with the Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: huang@ifp.uiuc.edu).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2008.2004963

estimated covariance matrix usually performs poorly, especially when the ratio of sample size and the number of variables is not large. This problem is commonly known as the "curse of dimensionality," and various techniques have been proposed to alleviate this effect [7].

In this brief, we propose an extension of FA for 2-D data [matrix-variate factor analysis (MVFA)], together with an efficient parameter estimation procedure based on the expectation/conditional maximization (ECM) algorithm [12]. The key difference between classical FA and MVFA lies in the way the data are represented. While classical FA works with 1-D vectors, MVFA directly operates on the image data, namely, the matrices. Therefore, the observations do not need to be first transformed into vectors and the implicit structural information can be well preserved. MVFA also brings faster parameter estimation and higher classification performance.

II. Factor Analysis

Given a p-dimensional observation t, FA tries to explain the relationship among the variables in t by relating them to a q-dimensional latent variable x as

$$t = \mu + Wx + e \tag{1}$$

where W is a $p \times q$ matrix, μ is the mean, and e is the error. Both t and x here are vector variates.

In applications such as image processing and computer vision, the data (images or image patches) are often high dimensional (e.g., over tens of thousands), and the loading matrix W consists of an enormous number of parameters. The estimation of these parameters can be computationally infeasible even on modern computers. In addition, for the estimated parameter to be statistically reliable, an impractically large amount of data may also be needed. Therefore, with insufficient training samples, both the *curse of dimensionality* and *small sample size problem* set in and degrade the performance of conventional FA.

III. Matrix-Variate Factor Analysis

In this section, we propose an extension of FA, the MVFA, to deal with 2-D data. The motivation originates from observing that in image applications, the data (images) are matrices and thus the implicit structural information can be used to improve the convention model (1) and lead to higher accuracy. The advantage of matrix representation over vector representation comes from that the spatial relationships (column/row information, and the neighboring relationship) are well preserved. Another motivation comes from previous works on face hallucination [11], where a higher resolution image matrix can be directly inferred from a lower resolution image matrix with proper noise modeling. This leads us to employ a low-dimensional matrix as the latent variable of the observed (image) matrix. Though there have been previous works on exploring the 2-D property of image data, such as the work on generalized low-rank approximations of matrices (GLRAM) [19], MVFA is the first method that explicitly considers a probabilistic model.

Denote the observation T by an $n \times m$ random matrix. MVFA represents T by a low-rank decomposition

$$T = \Xi + UXV' + E \tag{2}$$

where the superscript $'$ denotes matrix (or vector) transpose, Ξ is the mean, X is the $\tilde{n} \times \tilde{m}$ latent matrix (with $\tilde{n} \le n$ and $\tilde{m} \le m$), U $(n \times \tilde{n})$ and V $(m \times \tilde{m})$ are factor loading matrices, and E is the error/noise. MVFA thus tries to explain the structure of the observed data T by using a mean effect and a common factor matrix of reduced size. Similar low-rank approximations have been successfully and popularly used in various areas, such as information retrieval [3] and machine learning [6]. The novelty here lies in expressing the relationship between the observed T and the underlying X directly in matrix form, which is more coherent with human perception.

In conventional FA, the data matrix T has to be converted to an nm-dimensional vector t, and the latent matrix X has to be converted to an $\tilde{n}\tilde{m}$-dimensional vector x as in (1). The number of parameters associated with W is thus $nm\tilde{n}\tilde{m}$. With MVFA, the total number of parameters in U and V is only $n\tilde{n} + m\tilde{m}$. As will be shown in Section IV-B3, the time complexity is also reduced. Thus, by controlling the size of the latent matrix X, the number of parameters can be greatly reduced and parameter estimation becomes more efficient.

As in FA, we may have two kinds of models. In the first kind, we regard X as a random matrix, while in the second, X is not random and varies from one sample to another. In this brief, we will focus on the first kind. In particular, we assume that X has $\tilde{n}\tilde{m}$ independent identically distributed (i.i.d.) normally distributed random variables that are independent of E. We also assume that each element e_{rs} of E is i.i.d. normal random variable with zero mean and variance σ_{rs}^2.

IV. Parameter Estimation Using Expectation/Conditional Maximization

In this section, we develop an expectation–maximization (EM)-based method [5] for parameter estimation. Without loss of generality, we assume that the data is centered, and the model in (2) reduces to

$$T = UXV' + E. \tag{3}$$

Here, we will focus on the isotropic noise model where

$$\sigma_{rs}^2 = \sigma^2. \tag{4}$$

Extension to the anisotropic model is straightforward and omitted here due to the lack of space. An intuitive explanation of this model is that a high-resolution image is bilinearly interpolated from a low-resolution image plus extra noise, which is often used in image scale normalization.

A. Density Function and Log-Likelihood

Assuming that X and E are independent matrices consisting of random normal variables, we then have

$$p(T,X) = p(X)p(T|X) \propto \exp\left\{-\frac{1}{2}\mathrm{tr}(X'X)\right\} \times (\sigma^2)^{-nm/2} \times \exp\left\{-\frac{1}{2\sigma^2}\mathrm{tr}((T-UXV')'(T-UXV'))\right\}.$$

For $i = 1, \ldots, N$, let T_i be the observed (i.i.d.) data matrix and X_i the corresponding latent matrix. The complete-data log-likelihood is

$$L_c = -\left[\sum_{i=1}^{N} \frac{nm}{2}\ln\sigma^2 + \frac{1}{2}\mathrm{tr}(X_i'X_i) + \frac{1}{2\sigma^2}\mathrm{tr}(T_i'T_i) - \frac{1}{\sigma^2}\mathrm{tr}(X_i'U'T_iV) + \frac{1}{2\sigma^2}\mathrm{tr}(VX_i'U'UX_iV')\right] \tag{5}$$

and the marginal density for the observed data T is

$$p(T) = \int p(T,X)\,dX = \prod_{i=1}^{N}\int p(T_i,X_i)\,dX_i. \tag{6}$$

Notice that there is an intrinsic indeterminacy in maximum-likelihood estimation, as any rotation of U or V will not change the observed-data likelihood in (6). To resolve this ambiguity, we assume that the singular value decompositions of U and V are of the form $U_{\mathrm{orth}}\Lambda_u$ and $V_{\mathrm{orth}}\Lambda_v$, respectively, where the orthogonal matrices on the right of the singular value decompositions are identity matrices and Λ_u and Λ_v are diagonal matrices. As will be shown later, this parametrization has two merits. First, it simplifies our computations because most steps now only involve diagonal matrices (Section IV-B). Second, it allows the use of parameter expansion (PX, Section IV-B3), a technique accelerating the convergence of the EM algorithm.

B. Using ECM for MVFA

As in [16], we view X as "missing" and adopt an EM-based approach as our computational tool. One difficulty here is that the complete-data problem does not have a close solution, imposing substantial difficulty in implementing the standard M-step in the EM algorithm. Therefore, we will use the ECM algorithm proposed in [12]. The ECM algorithm is a powerful generalization of the EM algorithm. It consists of two steps: the expectation (E-step) and the conditional maximization (CM-step).

1) E-Step: The E-step of the ECM algorithm is the same as that of the EM algorithm. We compute the expectation of the log-likelihood in (5) with respect to (w.r.t.) the distribution $p(X|T, U, V, \sigma^2)$. This reduces to computing the expectation of the complete-data sufficient statistics X_i and $X_i'X_i$ w.r.t. the conditional distribution $p(X_i|T_i, U, V, \sigma^2)$ for each X_i.

Let $\mathrm{vec}(\cdot)$ be the vectorization operator, which concatenates the rows of the matrix argument and then transposes the result to a column vector, and let $\otimes$ be the Kronecker product. Using the fact that $\mathrm{vec}(ABC) = (C' \otimes A)\mathrm{vec}(B)$ for matrices A, B, and C, the matrix model (3) can be rewritten as

$$T^v = WX^v + E^v \tag{7}$$

where $T^v = \mathrm{vec}(T)$ (and similarly, for X^v and E^v) and

$$W = V \otimes U \tag{8}$$

is the factor score. Equation (7) now involves only vector variates, and the E-step becomes straightforward. As shown in [16], we have

$$\langle X_i^v \rangle = M^{-1}W'T_i^v \tag{9}$$

$$\left\langle X_i^v X_i^{v\prime} \right\rangle = \sigma^2 M^{-1} + \langle X_i^v \rangle \langle X_i^v \rangle' \tag{10}$$

where $M = W'W + \sigma^2 I_{\tilde{n}\tilde{m}}$ and $I_{\tilde{n}\tilde{m}}$ is the $\tilde{n}\tilde{m} \times \tilde{n}\tilde{m}$ identity matrix. These statistics are computed by using the parameters at their current values.

Instead of a direct implementation, we can have a more efficient way of computing (9). Using the properties of the Kronecker product, it can be easily shown that

$$M = \Lambda_u^2 \otimes \Lambda_v^2 + \sigma^2 I_{\tilde{n}\tilde{m}} \tag{11}$$

$$W'T_i^v = (U' \otimes V')T_i^v = \mathrm{vec}(U'T_iV). \tag{12}$$

Thus, we first compute (12), which takes $O(nm\min(\tilde{n}, \tilde{m}))$ time by using an appropriate order for the multiplications. Because M^{-1} is diagonal, computing (9) also takes only $O(nm\min(\tilde{n}, \tilde{m}))$ time.

The computation of (10) appears to require $O((\tilde{n}\tilde{m})^2)$ time. However, as will be shown shortly, only the diagonal elements of the matrices in (10) are used in the CM-step. Thus, the computations here can also be simplified.

2) CM-Step: Unlike the M-step in the EM algorithm, each CM-step consists of several simple conditional maximization steps. The idea is to sequentially update subsets of the parameters, analogous to the Gauss–Seidel iterations or the cyclic coordinate ascent method. Here, the parameters are partitioned into three groups: $(U_{\mathrm{orth}}, \Lambda_u)$, $(V_{\mathrm{orth}}, \Lambda_v)$, and σ^2. The ECM algorithm then replaces the original M-step of EM with three CM-steps.

In the first CM-step, we maximize the expectation of L_c w.r.t. $(U_{\mathrm{orth}}, \Lambda_u)$, with $(V_{\mathrm{orth}}, \Lambda_v)$ and σ^2 being fixed at their current values. Based on (5), this is equivalent to minimizing

$$\widetilde{L_c^u} = -2\mathrm{tr}(\Lambda_u S U_{\mathrm{orth}}) + \mathrm{tr}(\Lambda_u^2 H) \tag{13}$$

where $S = \sum_{i=1}^N \langle X_i \rangle V'T_i'$ and $H = \sum_{i=1}^N \langle X_i \Lambda_v^2 X_i' \rangle$. Suppose that the QR decomposition of S' is $S' = QR$, where R is a matrix with elements $r_{ij} = 0$ for $i > j$ and $r_{ii} \le 0$ and Q is an orthogonal matrix. Substituting this into (13) and noticing that $Q'U_{\mathrm{orth}}$ is still a matrix with orthonormal column vectors, we have

$$\hat{U}_{\mathrm{orth}} = Q \begin{bmatrix} I_{n'} \\ 0 \end{bmatrix}$$

because $\Lambda_u S U_{\mathrm{orth}} = (\Lambda_u R')(Q'U_{\mathrm{orth}})$, Λ_u is a diagonal matrix with nonnegative elements and $r_{ii} \le 0$. After obtaining $\hat{U}_{\mathrm{orth}}$, we can easily obtain

$$\hat{\Lambda}_u = \mathrm{diag}(S\hat{U}_{\mathrm{orth}})\mathrm{diag}(H)^{-1}.$$

Note that only the diagonal elements of the matrices in (10) are used here; the implementation of the E-step can thus be simplified as mentioned in Section IV-B1.

The second CM-step maximizes the expectation of L_c w.r.t. $(V_{\mathrm{orth}}, \Lambda_v)$, with $U = U_{\mathrm{orth}}\Lambda_u$ and σ^2 fixed. The computations are essentially the same as those in the first step.

The third CM-step maximizes the expectation of L_c w.r.t. σ^2, with $U = U_{\mathrm{orth}}\Lambda_u$ and $V = V_{\mathrm{orth}}\Lambda_v$ fixed at the values derived from the above two steps. As in [16], we obtain

$$\tilde{\sigma}^2 = \frac{1}{nmN}\sum_{i=1}^N \{\mathrm{tr}(T_i'T_i) - 2\mathrm{tr}(\langle X_i \rangle' U'T_iV) + \mathrm{tr}(\langle VX_i'U'UX_iV' \rangle)\}. \tag{14}$$

3) Remarks: The merits of the ECM algorithm are its simplicity and stability, even when the complete-data problem is complicated. Moreover, it guarantees that the likelihood function is always increasing. Besides, it can be easily seen that the so-called "space-filling" condition[1] is satisfied here. Thus, the ECM algorithm also converges to a local optimum under the same conditions that guarantee the convergence of EM [12]. A diagrammatic comparison of the graph representation for the (vector-based) FA algorithm PPCA and the proposed MVFA is shown in Fig. 1.

Computational Complexity Analysis: We first consider the computational complexity of MVFA. For each sample, it takes $O(nm\min(\tilde{n}, \tilde{m}))$ time for evaluating (9) and $O(\tilde{n}\tilde{m})$ time for evaluating (10), assuming that only the diagonal elements of $\langle X_i^v X_i^{v\prime} \rangle$ (which are those actually used in the CM-steps) are computed. Therefore, the total per-iteration time complexity for the E-step is $O(Nnm\min(\tilde{n}, \tilde{m}))$. As for the CM-step, it takes $O(Nnm\min(\tilde{n}, \tilde{m}))$ time for the evaluation of S and $O(Nnm)$ time for (14), assuming proper orders are followed in the implementation. Hence, the total per-iteration computational complexity of MVFA is $O(Nnm\min(\tilde{n}, \tilde{m}))$.

On the other hand, the per-iteration complexity of conventional FA is $O(Nnm\tilde{n}\tilde{m})$ [16], and is thus much larger. On problems of huge scale, it cannot even be implemented within tolerable time and memory limitations.

Speedup Using Parameter Expansion: As is well known, the convergence of EM-type algorithms can sometimes be slow. In this section, we use PX [10] as a covariance adjustment technique for accelerating the ECM algorithm of Section IV-B. We introduce two auxiliary diagonal matrices Γ_u and Γ_v. The expanded model is then parameterized as

$$T^v | X^v, \Theta \sim N(\mu^v + WX^v, \sigma^2 I_{nm}) \tag{15}$$

$$X^v | \Theta \sim N(0, \Gamma_u \otimes \Gamma_v) \tag{16}$$

where $W = V \otimes U$.

It is known that the fraction of missing information is related to the variance of parameters given the complete data relative to that given the observed data [10]. By using auxiliary parameters, extra variance is introduced into the complete model while the observed model remains

[1]Loosely speaking, this means unconstrained maximization is allowed over the whole parameter space [12].

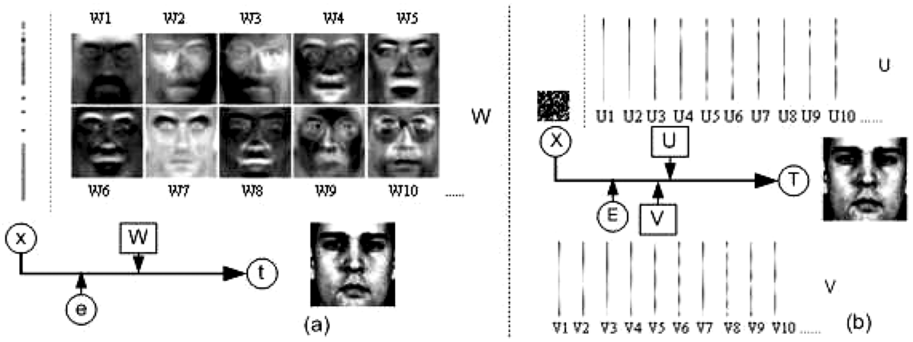
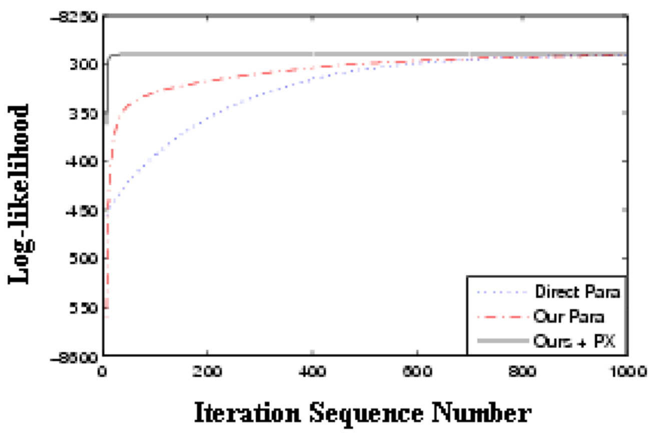
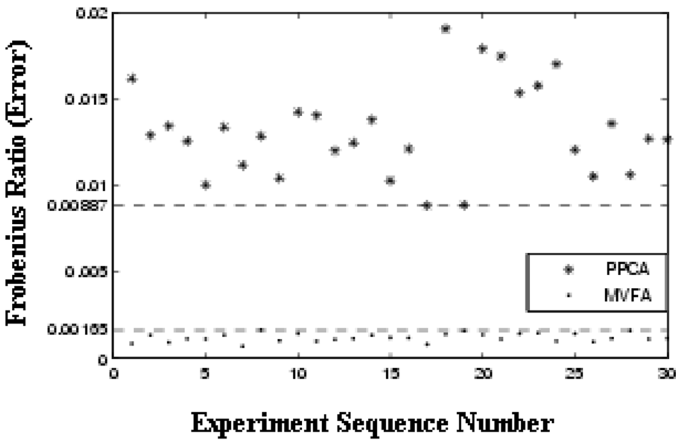

W1
W2
W3
W4
W5
W6
W7
W8
W9
W10 ......
W
x
W
t
e
(a)
U1 U2 U3 U4 U5 U6 U7 U8 U9 U10 ......
U
X
U
T
E
V
V
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 ......
(b)

Fig. 1. Graph representation of (a) PPCA and (b) MVFA. An example of the factor loading matrices and latent/observation variables (based on training samples from the XM2VTS face database) is shown next to the corresponding symbols. For simplicity, the data is assumed to have been centered. Moreover, the column vectors $\boldsymbol{W}_i$ of matrix $\boldsymbol{W}$ are reshaped into matrices for ease of display. Note that the $\boldsymbol{W}_i$ characterizes the global information of an image while $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ characterize the column and row information, respectively.



(a)



(b)

Fig. 2. (a) Log-likelihood values versus the number of iterations for the three methods on the first toy data set. (b) Normalized estimation errors for MVFA and PPCA obtained in the 30 repetitions.

the same, which in turn allows the algorithm based on the expanded model to enjoy a faster convergence.

The modified algorithm will be called the PX-ECM algorithm. Again, it has two steps: PX-E-step and PX-ECM-step. The major difference is that the parameters of the expanded model are now partitioned as $(U_{\rm orth}, \Lambda_u, \Gamma_u), (V_{\rm orth}, \Lambda_v, \Gamma_v)$, and σ^2, and we need to reduce the expanded parameters at the end of each iteration. It can be shown that the time complexity of a single iteration remains at $O(nm \min(\tilde{n}, \tilde{m}))$. Due to the lack of space, interested readers are referred to [10] for details.

V. Extensions of the Basic Model

In this section, we follow the approach in [8] and [15] and extend the model in (3) to a mixture of matrix-variate factor analyzers. Assume that we have a mixture of M factor analyzers, indexed by ω, taking values of $k = 1, \ldots, M$ for each observed data matrix. Here, $\omega = k$ means that the data point is generated from component k. We denote the mixing parameter by $\pi = [\pi_1, \ldots, \pi_M]$ with $\pi_k = P(\omega_k)$. Moreover, we denote the parameters for each distribution of the mixture components by U_k, V_k, and σ_k^2, respectively.

The log-likelihood of the observed data becomes $P(T) = \sum_{k=1}^{M} \int P(T|X, \omega = k) P(X|\omega = k) P(\omega = k)\, dX$. As in Section III, we again assume that the factor score matrix X consists of independent normally distributed random variables, and the noise is isotropic in each component.

Basically, the same ECM algorithm can be used for parameter estimation as in Section III. The difference is that we now have both X and ω as missing data in the E-step. Moreover, in the CM-step, we here have to update M groups of $(U_{\rm orth}, \Lambda_u, V_{\rm orth}, \Lambda_v, \sigma^2)$ parameters instead of just one. The complete algorithm is omitted here for space limitation.

The mixture of matrix-variate factor analyzers is, indeed, a reduced-dimension mixture of Gaussian distributions with parameters (U_k, V_k). For each component, its X_k is a matrix of lower dimensions, and so the proposed model achieves the goal of dimensionality reduction.

VI. Experiments

In this section, we perform a number of experiments on both toy and real-world data sets. For simplicity, isotropic noise is always assumed.

A. Toy Data Sets

In this section, experiments are performed on two toy data sets that are generated according to (2).

1) Convergence: The first data set consists of 2000 50×50 patterns, and the latent matrix X in (2) is of size 10×10. We compare the convergence properties of the following optimization methods for MVFA: 1) the ECM algorithm (Section IV-B); 2) the ECM algorithm with PX as discussed in Section IV-B3; and 3) an implementation based on direct parametrization of U and V. Denote all the parameters together by θ. In the experiment, θ is initialized randomly, and iterations stops when $\|\theta^{(t)} - \theta^{(t+1)}\| \leq 10^{-10}$ or the maximum number of iterations (set to 1000) is reached.

Fig. 2(a) shows how the log-likelihood improves with each iteration. It is evident that the ECM algorithm with PX converges most quickly in about 50 steps. On the other hand, the other two methods require nearly 1000 steps to achieve comparable likelihood values.

TABLE I
CLASSIFICATION ACCURACIES (IN PERCENT) OF MVFA AND PPCA ON THE MNIST DATABASE

	#training samples : # testing samples				
	70:430	60:440	50:450	40:460	30:470
MVFA	92.2	92.4	92.0	91.2	91.0
PPCA	91.8	86.8	77.8	55.6	45.7

2) Quality of Estimated Parameters: The second data set consists of 2000 20×20 patterns, and the latent matrix X in (2) is of size 5×5. We train both the MVFA model and the traditional (vector-based) PPCA model [16]. For faster convergence, PX is used in both models. We compare the estimated parameters with its ground truth. As mentioned in Section IV-B, there can be arbitrary rotations in the U and V estimates. Hence, we compute the normalized estimation error $(\|\hat{W}\hat{W}' - WW'\|_F)/(\|WW'\|_F)$, where $\|\cdot\|_F$ denotes the Frobenius norm and $W = V \otimes U$ (8) for MVFA. The experiment is repeated 30 times.

Fig. 2(b) shows the errors obtained from the 30 repetitions. Two dashed lines are also added to show the maximum of the MVFA values and the minimum of the PPCA values obtained. As can be seen, MVFA is superior in terms of accuracy. It also runs faster than PPCA (1745 s versus 11 620 s, on a 3.0-GHz Pentium-4 PC with 512-MB memory, running Matlab 7.0), which agrees with our time complexity analysis in Section IV-B3.

B. Real-World Data Sets

1) Statistical Classification: In this section, we perform classification experiments on the MNIST database [9] of handwritten digits. A total of 5000 28×28 images (500 per digit) are used. Instead of using the raw pixel intensities as features (dimensions are too small for further dimension reduction), we first extract 40 Gabor features, with five scales and eight directions, in the down-sampled positions. Each image is then re-encoded as a matrix of size $(14 \times 5 = 70) \times (14 \times 8 = 112)$.

One MVFA/PPCA model is trained for each class (digit). A test sample is classified to the class whose model produces the highest likelihood. We study the performance by varying the numbers of training and test samples (with the total fixed at 500 for each digit). For both MVFA and PPCA, all possible dimensionalities of the latent variables are tried and the best results reported.

As can be seen from Table I, MVFA significantly outperforms PPCA, especially when there are few training samples. Moreover, MVFA has a more stable performance as the number of training samples varies.

2) Dimensionality Reduction: In this section, experiments are performed on the XM2VTS and CMU PIE (pose, illumination, and expression) face databases.[2] There are 295 subjects (classes) in the XM2VTS database. We use a total of 875 images from the first three sessions for training and 295 images from the fourth session for testing. For the CMU PIE database, there are 68 classes. We use five near-frontal poses and four illuminations (indexed as 08, 10, 11, and 13), resulting in a total of 20 images per person. Twelve of them are for training, and the remaining eight are for testing. All the images are normalized to 64×64 by fixing the locations of the two eyes.

Here, we use MVFA/PPCA for dimensionality reduction. One MVFA/PPCA model is trained for the whole training set. For an observation T_0, we use the expectation of the latent variable $E(X|T = T_0)$ as the lower dimensional representation of T_0 for MVFA. Similarly, we use $E(x|t = t_0)$ for PPCA, where $t_0 = \text{vec}(T_0)$. The nearest neighbor classifier[3] is used in the low-dimensional space for classification. For comparison, we also report the results from PCA, GLRAM, and that directly based on original raw features, which is referred to as *w/o* in Table II. As in Section IV-B, all possible dimensionalities for the latent variables are tried and the best results are reported.

[2]Available at http://www.face-rec.org/databases/

[3]For MVFA, the obtained $\boldsymbol{E(X|T = T_0)}$ is still a matrix, and so the Frobenius norm is used to compute the distance for the nearest-neighbor classifier. On the other hand, the other methods only produce vectors in the latent space and so the usual Euclidean distance can be used.

TABLE II
CLASSIFICATION ACCURACIES (IN PERCENT) BY USING VARIOUS DIMENSIONALITY REDUCTION METHODS

	MVFA	PPCA	GLRAM	PCA	w/o
XM2VTS	**78.6**	76.6	71.2	69.5	68.5
CMU PIE	**87.3**	78.3	65.1	63.5	63.5

Results are shown in Table II. As can be seen, the low-dimensional representations obtained by MVFA are consistently superior than the others.

3) Remarks: Recall that our focus here is on the advantages of using a matrix representation (instead of a vector representation) in FA. Hence, for simplicity, we have only considered MVFA as an unsupervised learning method, and did not compare it with supervised face recognition algorithms such as Fisherfaces [2] and other linear discriminant analysis (LDA)-based methods. Moreover, discriminant analysis can certainly be added to both PPCA and MVFA as a postprocessing step for recognition purposes. However, this is not done here as it will blur the contributions of the FA and discriminant analysis steps and thus make the various FA algorithms difficult to compare. Besides, we did not compare with the Tensorface, because the Tensorface [17] is restricted to situations where the images for all poses, illumination conditions, and expressions for each person are available. MVFA, on the other hand, is more flexible and does not have such prerequisite on the training data.

VII. CONCLUSION AND FUTURE WORK

In this brief, we proposed a new covariance analysis technique called MVFA for the analysis of 2-D data (such as images). MVFA has several advantages over conventional vector-based FA. First, it utilizes the observation matrix directly, which is consistent with human perception. Second, MVFA is more accurate than FA in terms of parameter estimation and classification performance. Third, MVFA is computationally more efficient.

However, one disadvantage of MVFA is that the estimates may have bias if the model assumption is violated. To resolve this problem, one possibility is to extend this to an additive model, as $T = \Xi + \sum_{k=1}^{K} U_k X_k V_k' + E$. From the perspective of the dictionary method [7], each component then shares an MVFA formulation. This extension will be further studied in the future. Moreover, as demonstrated in [16], there exists a strong relationship between PPCA and PCA. An interesting research topic is to explore the underlying relationship between MVFA and the recently proposed GLRAM algorithm [19]. Another interesting future work is to extend this work to tensor-variate FA for handing general tensor data of arbitrary order [18], with MVFA as a special case for second-order tensor data, namely, matrices.

REFERENCES

[1] T. Anderson and H. Rubin, J. Neyman, Ed., "Statistical inference in factor analysis," in *Proc. 3rd Berkeley Symp. Math. Statist. Probab.*, Berkeley, CA, 1956, vol. 5, pp. 111–150.

[2] P. Belhumeur, J. Hespanda, and D. Kiregeman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[3] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Rev.*, vol. 37, no. 4, pp. 573–595, 1995.

[4] C. Bishop and M. Tipping, "A hierarchical latent variable model for data visualization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 281–293, Mar. 1998.

[5] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

[6] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 243–264, 2001.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.
[8] G. Hinton, M. Revow, and P. Dayan, "Recognizing handwritten digits using mixtures of linear models," in *Advances in Neural Information Processing Systems 7*. Cambridge, MA: MIT Press, 1995, pp. 1015–1022.
[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[10] C. Liu, D. Rubin, and Y. Wu, "Parameter expansion to accelerate EM: The PX-EM algorithm," *Biometrika*, vol. 85, no. 4, pp. 755–770, 1998.
[11] C. Liu, H. Shum, and C. Zhang, "A two-step approach to hallucinating faces global parametric model and local nonparametric model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2001, pp. 192–198.
[12] X. Meng and D. Rubin, "Maximum likelihood estimation via the ECM: A general framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.
[13] B. Moghaddam, T. Jebara, and A. Pentland, "Bayesian face recognition," *Pattern Recognit.*, vol. 33, no. 1, pp. 1771–1782, 2000.
[14] C. Spearman, "General intelligence, objectively determined and measured," *Amer. J. Psychol.*, vol. 15, pp. 201–293, 1904.
[15] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computat.*, vol. 11, no. 2, pp. 443–482, 1999.
[16] M. Tipping and C. Bishop, "Probabilistic principal component analysis," *J. Roy. Statist. Soc. B*, vol. 21, no. 3, pp. 611–622, 1999.
[17] M. Vasilescu and D. Terzopoulos, "Multilinear subspace analysis for image ensembles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2003, vol. 2, pp. 93–99.
[18] D. Xu, S. Yan, L. Zhang, H. Zhang, Z. Liu, and H. Shum, "Concurrent subspace analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 203–208.
[19] J. Ye, "Generalized low rank approximations of matrices," *Mach. Learn.*, vol. 61, no. 1–3, pp. 167–191, 2005.

Nonlinear Knowledge-Based Classification

Olvi L. Mangasarian and Edward W. Wild

***Abstract*—In this brief, prior knowledge over general nonlinear sets is incorporated into nonlinear kernel classification problems as linear constraints in a linear program. *These linear constraints are imposed at arbitrary points, not necessarily where the prior knowledge is given.* The key tool in this incorporation is a theorem of the alternative for convex functions that converts nonlinear prior knowledge implications into linear inequalities without the need to kernelize these implications. Effectiveness of the proposed formulation is demonstrated on publicly available classification data sets, including a cancer prognosis data set. Nonlinear kernel classifiers for these data sets exhibit marked improvements upon the introduction of nonlinear prior knowledge compared to nonlinear kernel classifiers that do not utilize such knowledge.**

***Index Terms*—Kernel classification, linear programming, prior knowledge, theorem of the alternative.**

I. Introduction

Prior knowledge has been used effectively in improving classification both for linear [8] and nonlinear [7] kernel classifiers as well as for nonlinear kernel approximation [19]. In all these applications, prior knowledge was converted to linear inequalities that were imposed on a linear program. The linear program generated a linear or nonlinear classifier, or a linear or nonlinear function approximation, all of which were more accurate than the corresponding results that did not utilize prior knowledge. However, whenever a nonlinear kernel was utilized in these applications, kernelization of the prior knowledge was not a transparent procedure that could be easily related to the original sets over which prior knowledge was given. In contrast, in [20], no kernelization of the prior knowledge sets was used in order to incorporate that knowledge into a nonlinear function approximation. We will use a similar approach here to incorporate prior knowledge into a nonlinear classifier without the need to kernelize the prior knowledge. Furthermore, the region in the input space on which the prior knowledge is given is completely arbitrary in this work, whereas in all previous classification work, prior knowledge had to be restricted to convex polyhedral sets. The present approach is possible through the use of a fundamental theorem of the alternative for convex functions that we describe in Section II of this brief, whereas previous work utilized such a theorem for linear inequalities *only*. An interesting, novel approach to knowledge-based support vector machines that modifies the hypothesis space rather than the optimization problem is given in [14]. In another recent approach, prior knowledge is incorporated by adding additional points labeled based on the prior knowledge to the data set [17]. A somewhat different approach for prior knowledge incorporation consists of the generation of additional points based on prior knowledge. This was employed in [17] and [22] where virtual examples were created as well as in [5] and [10].

In Section III, we describe our linear programming formulation that incorporates nonlinear prior knowledge into a nonlinear kernel, while Section IV gives numerical examples that show prior knowledge can improve a nonlinear kernel classification significantly. Section V concludes this brief.

We describe our notation now. All vectors will be column vectors unless transposed to a row vector by a prime $'$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. For $x \in R^n$, $\|x\|_1$ denotes the 1-norm: $(\sum_{i=1}^n |x_i|)$ while $\|x\|$ denotes the 2-norm: $(\sum_{i=1}^n (x_i)^2)^{1/2}$. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix, A' will denote the transpose of A, A_i will denote the ith row of A, and $A_{\cdot j}$ will denote the jth column of A. A vector of ones in a real space of arbitrary dimension will be denoted by e. Thus, for $e \in R^m$ and $y \in R^m$, the notation $e'y$ will denote the sum of the components of y. A vector of zeros in a real space of arbitrary dimension will be denoted by 0. For $A \in R^{m\times n}$ and $B \in R^{n\times k}$,, a *kernel* $K(A, B)$ maps $R^{m\times n} \times R^{n\times k}$ into $R^{m\times k}$. In particular, if x and y are column vectors in R^n, then $K(x', y)$ is a real number, $K(x', B')$ is a row vector in R^m, and $K(A, B')$ is an $m \times m$ matrix. We will make no assumptions whatsoever on our kernels other than symmetry, that is, $K(x', y)' = K(y', x)$, and in particular, we will not assume or make use of Mercer's positive-definiteness condition [23], [24], [4]. The base of the natural logarithm will be denoted by ε. A frequently used kernel in nonlinear classification is the Gaussian kernel [24], [2] whose ijth element, $i = 1, \dots, m,\ j = 1, \dots, k$, is given by $(K(A, B))_{ij} = \varepsilon^{-\mu\|A_i{}' - B_{\cdot j}\|^2}$, where $A \in R^{m\times n}$, $B \in R^{n\times k}$, and μ is a positive constant. We represent data as vectors for convenience, and the approach we describe below works with any data for which a suitable kernel can be found. The abbreviation "s.t." stands for "subject to."

II. Conversion of Nonlinear Prior Knowledge Into Linear Constraints

The problem that we wish to impart prior knowledge to consists of classifying a data set in R^n represented by the m rows of the matrix

Manuscript received July 31, 2007; revised February 6, 2008 and July 25, 2008; accepted August 24, 2008. Current version published October 8, 2008. This work was supported by National Science Foundation under Grants CCR-0138308 and IIS-0511905, by the Microsoft Corporation, and by ExxonMobil.

The authors are with the Computer Sciences Department, University of Wisconsin, Madison, WI 53706 USA (e-mail: olvi@cs.wisc.edu; wildt@cs.wisc.edu).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.
Digital Object Identifier 10.1109/TNN.2008.2005188