IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Transformation invariant component analysis for binary images

Zoran Zivkovic
ISLA Lab
University of Amsterdam, The Netherlands
zivkovic@science.uva.nl

Jakob Verbeek
GRAVIR-INRIA
655 av. de l'Europe, 38330 Montbonnot, France
verbeek@inrialpes.fr

Abstract

There are various situations where image data is binary: character recognition, result of image segmentation etc. As a first contribution, we compare Gaussian based principal component analysis (PCA), which is often used to model images, and "binary PCA" which models the binary data more naturally using Bernoulli distributions. Furthermore, we address the problem of data alignment. Image data is often perturbed by some global transformations such as shifting, rotation, scaling etc. In such cases the data needs to be transformed to some canonical aligned form. As a second contribution, we extend the binary PCA to the "transformation invariant mixture of binary PCAs" which simultaneously corrects the data for a set of global transformations and learns the binary PCA model on the aligned data. [1]

1. Introduction

Principal component analysis (PCA) is a popular technique for processing, compressing and visualizing data. PCA finds a low dimensional representation of the data by linear projection, this linearity is a limiting factor for complex data. While nonlinear variants have been proposed, an alternative paradigm is to capture data complexity by a combination of local linear projections. This leads to the mixture of probabilistic PCAs (MPPCA) [10] model. In computer vision PCA has been used to model faces [11], handwritten digits [4], for tracking objects [1], etc.

The usual probabilistic PCA model is based on a Gaussian density over the data [10]. There are various situations, however, where the processed image data is binary or can be considered binary. Character recognition is an example where the data is usually binary by nature. Furthermore, various image segmentation algorithms, used to preprocess data, typically lead to binary data, e.g. skin-color segmentation, and foreground/background segmentation [14]. Finally, the "layered image model" [12] represents an image as a composition of layers where each layer corresponds to a different object. The layers are combined using binary masks. In fact, the skin-color segmentation and the foreground/background segmentation can be seen as simple two-layer models. In all these cases binary data naturally occurs, but is often modelled using Gaussian-based models for simplicity. In this paper we consider Bernoulli distributions instead, which are more natural for binary data.

In this paper we build upon the "binary PCA" model of [8], which is based on linear compression of the log-odds of the Bernoulli distributions modeling the pixel values. We apply binary PCA on binary image data and compare it to Gaussian-based PCA. As mentioned above, mixtures of PCA models can be used to handle more complex data. However, it still might be difficult to model variation due to global transformations, such as shifts, rotations, scaling, etc. Take for example the binary images of a silhouette of a human walking in Figure 1. If we use the PCA directly on the images then it will focus on describing the translation of the person since it is the dominant image deformation. It is more natural to handle the unknown translation separately, and to use PCA to describe the complex non-rigid deformations of the silhouette. In spirit of [3, 6] we present a "transformation invariant mixture of binary PCAs" which is invariant to a set of global transformations.

The rest of this paper is organized as follows. In Section 2 we briefly describe the binary PCA model, and in Sections 3 and 4 we extend it to the transformation invariant mixture of binary PCAs. In Section 5 we present experimental results, and in Section 6 we list our conclusions and some topics for further research.

2. Binary PCA

A natural way to model binary data is using Bernoulli distributions. For an univariate variable $x \in \{0, 1\}$ we have:

$$p(x|\alpha) = \alpha^x (1-\alpha)^{1-x} \quad (1)$$

where α is the probability that $x = 1$. Using the log-odds parameter $\theta = \log(\alpha/(1-\alpha))$ and the logistic function

[1] The work described in this paper was partially supported by the EU FP6-002020 COGNIRON ("The Cognitive Companion") project.

$\sigma(\theta) = (1 + e^{-\theta})^{-1}$ this can be equivalently written as:

$$p(x|\theta) = \sigma(\theta)^x \sigma(-\theta)^{1-x}. \quad (2)$$

The log-likelihood from above can then be written as:

$$\mathcal{L}(\theta) = x\theta + \log \sigma(-\theta). \quad (3)$$

Suppose we have N images of equal size, the n-th image will be denoted by X_n and the pixel value at position $\mathbf{d}$ in 2D image coordinates is denoted by $X_n(\mathbf{d})$. Another notation that we will use considers the image as a long vector obtained by concatenating the rows of the image. If the image has D pixels we will get a D-dimensional vector. The d-th element of the vector will be denoted by X_{nd} and it will correspond to the value denoted by $X_n(\mathbf{d})$ of the image. The data set of N images can then be represented by a large $N \times D$ dimensional matrix X where the rows of the matrix contain the images and X_{nd} denotes the d-th pixel of the n-th image. Given the log-odds, the pixels within an image are considered independent and we can write:

$$p(X_n|\Theta_n) = \prod_d \sigma(\Theta_{nd})^{X_{nd}} \sigma(-\Theta_{nd})^{1-X_{nd}}. \quad (4)$$

The log-odds parameter corresponding to the pixel value X_{nd} is denoted by Θ_{nd}. The log-likelihood of the data set of N images equals:

$$\mathcal{L}(\Theta) = \sum_{n,d} X_{nd}\Theta_{nd} + \log \sigma(-\Theta_{nd}). \quad (5)$$

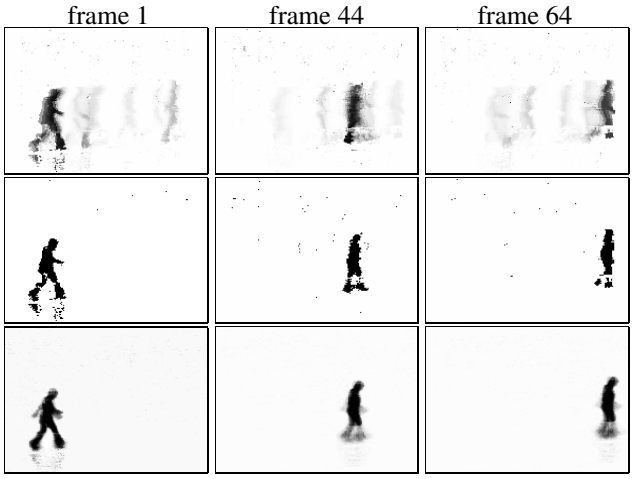
Binary principal component analysis [8] starts by assuming that each row of the log-odds matrix Θ is given by the row vector μ plus a linear combination of $L \ll D$ basis vectors (images) contained in the rows of the $L \times D$ matrix W. The linear combination is obtained through the coefficients contained in the $N \times L$ matrix U:

$$\Theta_{nd} = \mu_d + \sum_l U_{nl} W_{ld}. \quad (6)$$

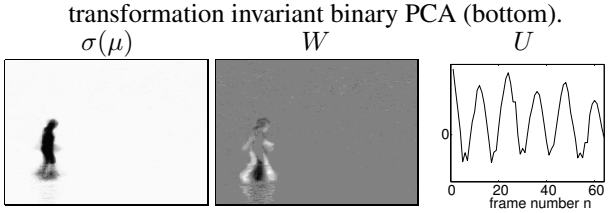
We will denote the parameters of the log-odds matrix by $\Omega = (\mu, W, U)$. The low dimensional structure in the data can then be discovered by finding the parameters Ω that maximize the log-likelihood (5). The maximum can not be found in closed form but there exist an efficient iterative procedure [8]. For completeness of the text the iterative update equations are given in Appendix A.

3. Transformation invariant binary PCA

Before applying PCA to the data it makes sense to transform the data to some canonical aligned form as we discussed in the introduction. See also Figure 1. Let $\mathcal{T}$ denote the unknown global transformation (typically translation, rotation or scaling) that transforms the image X_n into some canonical form $Z_n = \mathcal{T}(X_n)$. We will consider the $\mathcal{T}$ as a random variable with a prior distribution $p(\mathcal{T})$. The probabilistic model can be written as:



a) the original images (middle),reconstruction using $L = 5$ component binary PCA (top), reconstruction using $L = 1$ transformation invariant binary PCA (bottom).



b) the model learned using the transformation invariant binary PCA.

Figure 1. Illustration of transformation invariant binary PCA. A human walking sequence of 64 frames was used to learn the model. The image size is 140×190. The sequence presents the result of a simple foreground/background segmentation algorithm. Note that U nicely captures the cyclical walking motion while W models the corresponding deformations.

$$p(X_n, \mathcal{T}|\Theta_n) = p(\mathcal{T})p(X_n|\mathcal{T}, \Theta_n), \quad (7)$$

$$p(X_n|\mathcal{T}, \Theta_n) = \prod_d \sigma(\Theta_{nd})^{Z_{nd}} \sigma(-\Theta_{nd})^{(1-Z_{nd})}. \quad (8)$$

The transformation $\mathcal{T}$ variable is unobserved. The EM algorithm [2] can be used to perform the transformation invariant PCA but in general case this might be intractable. In spirit of [3, 6] we consider only a discrete set of T transformations $\mathcal{T}_t \in \{\mathcal{T}_1, ..., \mathcal{T}_T\}$. We denote the prior on transformations by $p_t = p(\mathcal{T}_t)$. The likelihood becomes:

$$p(X_n|\Theta_n) = \sum_t p_t p(X_n|\mathcal{T}_t, \Theta_n). \quad (9)$$

and the log-likelihood for a set of N images is:

$$\mathcal{L}(\Omega) = \sum_n \ln p(X_n|\Theta_n). \quad (10)$$

The transformation invariant PCA is performed by finding the parameters $\Omega = (p_t, \mu, W, U)$ that maximize (10).

3.1. EM algorithm

The log-likelihood is a complex function. The EM algorithm presents a simple iterative solution and it is derived as follows. First from the Jensen's inequality we have:

$$\mathcal{L}(\Omega) \geq \sum_{n,t} q_{nt} \ln p_t p(X_n|\mathcal{T}_t, \Theta_n) - \sum_{n,t} q_{nt} \ln q_{nt} \quad (11)$$

where q_{nt} is a $N \times T$ matrix, and each row is a discrete distribution: $\sum_t q_{nt} = 1$. Only the first term from the equation depends on the parameters Ω and it is denoted as:

$$Q(\Omega) = \sum_{n,t} q_{nt} \ln p_t p(X_n|\mathcal{T}_t, \Theta_n). \quad (12)$$

Let $\widehat{\Omega}$ denote the current parameter estimates in the EM itarative procedure and $\widehat{\Theta}$ the corresponding log-odds values. The exact EM sets the q_{nt} to the posterior distribution over the unknown transformation

$$q_{nt} = p(\mathcal{T}_t|X_n, \widehat{\Theta}_n) = p(X_n, \mathcal{T}_t|\widehat{\Theta}_n)/p(X_n|\widehat{\Theta}_n). \quad (13)$$

It can be shown that the inequality sign in (11) becomes equality for $\Omega = \widehat{\Omega}$. Then the new paramaters $\widehat{\Omega}$ are found that maximize the function Q while keeping q_{nt} fixed. These two steps are repeated iteratively until convergence:

$$\text{E step:} \quad \text{calculate } Q(\Omega) \quad (14)$$

$$\text{M step:} \quad \widehat{\Omega} = \arg\max_{\Omega}(Q(\Omega)). \quad (15)$$

3.2. E step

By substituting (8) into (12), taking into account that $\sum_t q_{nt} = 1$, we obtain:

$$Q(\Omega) = \sum_{n,t} q_{nt} \ln p_t + \sum_{n,d} \langle Z_{nd} \rangle \Theta_{nd} + \log \sigma(-\Theta_{nd}) \quad (16)$$

where

$$\langle Z_{nd} \rangle = \sum_t q_{nt} Z_{nd}. \quad (17)$$

The $\langle Z_{nd} \rangle$ can be seen as "expected aligned data" since q_{nt} are set to the posterior distribution for the unknown transformation. Next, we discuss how to efficiently calculate Q for a large number of possible discrete transformations.

3.3. Fast E step for discrete shifts

It is particulary interesting to consider the discrete 2D image shift transformation since it is a common transformation to align images. Furthermore, an efficient solution for the E step is available as we describe here. Note that by transforming an image to log-polar coordinates, shifts correspond to rotations and scalings [13].

Let $\mathcal{T}_t$ correspond to a 2D discrete shift described by the 2D vector $\mathbf{t}$. A pixel value of the aligned image is then $Z_{nd} = Z_n(\mathbf{d}) = X_n(\mathbf{d} + \mathbf{t})$. We consider all possible discrete shifts. The total number of shifts is equal to the number of pixels D. We will show, following [3], that it is possible to calculate $Q(\Omega)$ in $D \log D$ time.

First, we will need to calculate the posterior distribution q_{nt} for all D possible shifts. The likelihood (7) equals:

$$p(X_n, \mathcal{T}_t|\widehat{\Theta}_n) = p_t \exp\left[\sum_d Z_{nd}\widehat{\Theta}_{dn} + \log\sigma(-\widehat{\Theta}_{dn})\right] \quad (18)$$

By using the alternative notation where we consider the Z_n and $\widehat{\Theta}_n$ as 2D arrays and not just as long vectors, the above equation can be written as:

$$p_t \exp\left[\sum_{\mathbf{d}} X_n(\mathbf{d} + \mathbf{t})\widehat{\Theta}_n(\mathbf{d}) + \log\sigma(-\widehat{\Theta}_n(\mathbf{d}))\right]. \quad (19)$$

The summation goes now over the 2D image positions $\mathbf{d}$. The second term in the exponent is computed in linear time. The first term in the exponent has the form of a 2D convolution of image X_n with the $\widehat{\Theta}_n$. It is possible to compute the first term for all possible discrete shifts $\mathbf{t}$ efficiently using the fast fourier transform (FFT). The computation time will be proportional to $D \log D$. The final $q_n t$-s are computed by normalization (13).

Note that q_{nt}, for a fixed n, has $T = D$ values and can be also considered as a 2D array. We denote q_{nt} corresponding to the discrete shift $\mathbf{t}$ as $q_n(\mathbf{t})$. Using this notation we can write $\langle Z_{nd} \rangle$ as:

$$\langle Z_{dn} \rangle = \sum_{\mathbf{t}} q_n(\mathbf{t}) X_n(\mathbf{d} + \mathbf{t}). \quad (20)$$

Again the equation has form of a 2D image convolution and can be computed efficiently. In conclusion $Q(\Omega)$ can be computed efficiently by performing for each image 2 image convolutions and a few simple additional operations computed in time linear with respect to the number of pixels D.

3.4. M step

The first term in $Q(\Omega)$ depends only on p_t. It is easy to verify that the p_t maximizing $Q(\Omega)$ is given by $p_t = (1/N)\Sigma_n q_{nt}$. If there is not enough data to estimate p_t reliably, we may use a uniform prior distribution over the transformations: $p_t = 1/D$.

The second term in $Q(\Omega)$ has the same form as the log likelihood for the binary PCA from the previous section (5). The only difference is that the data X_{nd} is replaced by the expected aligned data $\langle Z_{nd} \rangle$ computed as described above.

It is not possible to find the maximum of $Q(\Omega)$ with respect to μ, U and W directly but we can use the same iterative procedure as in the regular binary PCA to find $\widehat{\Omega}$. Alternatively we could use just one step of the iterative procedure that will improve $Q(\Omega)$ and continue to E-step again. This presents a generalized EM algorithm that also has a guaranteed convergence.

3.5. The algorithm summary

For the sake of clarity we summarize the practical algorithm for the discrete shifts:

Initialization: The parameter μ can be initialized by the mean value for the data taking care that it represents the log-odds. However in case where there are large shifts in the data it might be useful to take a single image and set the μ_{nd} to some small positive value for white pixels and small negative value for the black pixels. For the first few iterations we keep the basis vectors V and the coefficients U to zero and then initialize them by some small random values, for example sampled from a zero mean Gaussian distribution with the standard deviation 0.001.

1: For each image calculate the posterior distribution for all possible shifts q_{nt} using (13) and (19).

2: Calculate the expected aligned images $\langle Z_n \rangle$ (20).

3: Update the parameter estimates $\widehat{\Omega}$ using the update equations from Appendix A and, if required that for p_t.

4: Stop if increase of the data (log)likelihood is below some threshold, otherwise go to 1.

4. Transformation invariant clustering

A more flexible model can be obtained if a mixture of PCAs is used. This is particulary appropriate if the data is naturally divided into a number of clusters. The extension of the transformation invariant PCA from the previous section to transformation invariant mixture of PCAs is straightforward, similar to [10], and we describe it here only briefly.

We intricude an additional discrete unobserved variable $c \in \{1, \ldots, C\}$ which denotes the unknown class (cluster) label. For each of the C clusters we have a separate set of log-odds parameters Θ_c and corresponding μ_c, W_c, U_c. By Θ_{cn} we denote the log-odds parameters for the c-th class and n-th image. The model (7) becomes:

$$p(X_n, \mathcal{T}_t, c|\Theta) = p_{ct} p(X_n|\mathcal{T}, \Theta_{cn}) \quad (21)$$

where $p_{ct} = p(\mathcal{T}_t, c)$ is the joint prior distribution on class labels and the transformations. The joint posterior equals:

$$q_{cnt} = p(\mathcal{T}_t, c|X_n, \widehat{\Theta}) = p(X_n, \mathcal{T}_t|\widehat{\Theta})/p(X_n|\widehat{\Theta}_n). \quad (22)$$

The posterior distribution on the class label c is $q_{cn} = \sum_t q_{cnt}$. The EM steps are performed in similar way as before. In case of discrete shifts (19) and (20) are performed for every class c.

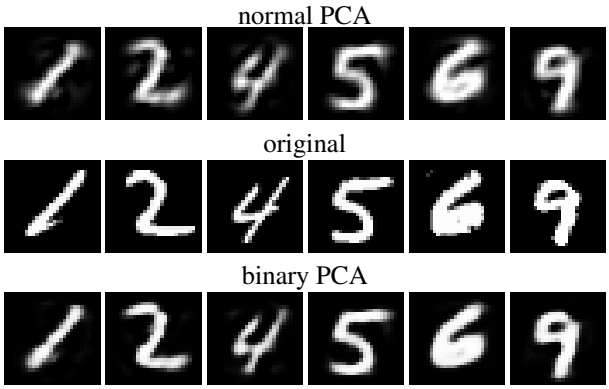


Figure 2. A few 28×28 images from the MNIST dataset are shown in the middle row. The reconstructed images using the normal PCA are presented in the top row and the images reconstructed using the binary PCA are in the last row.

	e_2	e_{log}	e_{01}
Norm.PCA	0.015 (0.006)	7.6 (0.6)	0.039 (0.012)
Bin.PCA	0.006 (0.003)	6.4 (0.3)	0.029 (0.009)

Table 1. Reconstruction error results for the MNIST dataset. Normal and binary PCA with $L = 40$ components are compared. The mean error per pixel over all images is reported. The standard deviation over images is reported within the brackets.

5. Experiments

In this section we analyze the performance of the transformation invariant binary component analysis on a number of examples. We start with comparing the Bernoulli to the Gaussian based component analysis.

5.1. Reconstruction/compression

In order to compare the quality of the Gaussian and Bernoulli based models we conducted several experiments. In each experiment we learn a model on a collection of images X_n. We used 5000 images from the 60000 training images in the MNIST dataset [7]. The remaining images are used for testing. The dataset contains 28×28 images of handwritten digits. Using the model, we compress the images to the PCA scores. Finally, we use the model to project the PCA scores back to images $\hat{X}_n$. See some examples in Figure 2. We then measure the difference between the original image and its reconstruction after compression/decompression. We measured the error in three ways. (i) Quadratic loss: the sum of the squared differences per pixel value, $e_2 = (1/D) \sum_d (X_{nd} - \hat{X}_{nd})^2$. (ii)Sigmoidal loss: the sum of the log-likelihood of the original images given the reconstructions, $e_{log} = 1/D \sum_d X_{nd} \ln \hat{X}_{nd} + (1 - X_{nd}) \ln(1 - \hat{X}_{nd})$. As the reconstruction from the Gaussian model can be outside $(0, 1)$, we first map values outside this interval to $\epsilon = 10^{-6}$ and $1 - epsilon$ respectively.(iii) Zero-one loss: first we threshold the reconstruc-

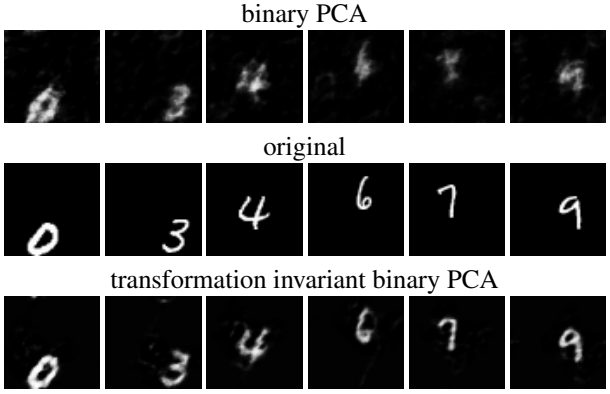


Figure 3. Example resconstructions of 56×56 images constructed by randomly positioning the images from the MNIST dataset. Reconstructed images using the binary PCA with $L = 40$ (top), the original images (middle), and reconstructions using transformation invariant binary PCA with $L = 40$ (bottom).

	e_2	e_{log}	e_{01}
Bin.PCA	0.010 (0.003)	5.47 (0.3)	0.021 (0.007)
Trans. inv. Bin.PCA	0.006 (0.003)	5.12 (0.6)	0.012 (0.006)

Table 2. Reconstruction error results for the MNIST dataset with random global 2D shifts. The original 28×28 images are set at a random position in a larger 56×56 image. Binary PCA and transformation invariant binary PCA with $L = 40$ components are compared. The mean error per pixel over all images is reported. The standard deviation over images is reported within the brackets.

tions at $\sigma(\Theta_{nd}) > 1/2$ to get a binary reconstruction $\hat{X}_n^{01}$, then we measure the number of pixels that differ from the original, $e_{01} = (1/D)\sum_d |X_{nd} - \hat{X}_{nd}^{01}|$. The results for $L = 40$ PCA are reported in Table 5.1. Clearly, binary PCA leads to big improvements, also visible in Figure 2.

5.2. Reconstruction/compression - non-aligned data

We performed a similar experiment to the experiment from the previous section but now the data was not aligned. We used the MNIST data and generated a new non-aligned dataset by placing each MNIST 28×28 image into a larger 56×56 image at a random position. See Figure 3. The binary PCA and the transformation invariant binary PCA with $L = 40$ components are compared. For the transformation invariant binary PCA we used for each image the inverse of the most likely global transformation $max_t(q_{nt})$ to reconstruct the original image. The same was done previously in Figure 1 for the walking sequence. Table 5.2 shows the advantage of transformation invariant binary PCA. In Figure 3 we can see that regular binary PCA with $L = 40$ could not deal with the complexity of the data and the reconstructions look poor. On the other hand, the transformation invariant version still gives visually quite good reconstruction.

5.3. Clustering

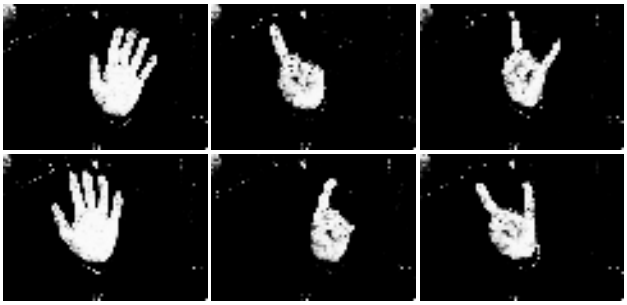
We demonstrate here an application of the transformation invariant mixture of binary PCAs for automatical video anlysis. A simple 447-frame sequence is recorded of a hand moving in front of the camera and changing between 3 different configurations. The images are segmented using a skin-color model. A few frames are shown in Figure 4. We used these images to train the transformation invariant mixture of binary PCAs with $C = 3$ classes and $L = 1$ principal component. The algorithm automatically detects the three different hand configurations while the small deformations of each configuration are modelled by the principal component. This is shown in Figure 4b. In the experiments here we did not include the knowledge that the transitions through the sequence are smooth as for example in [5]. Including the smoothness probalby improves the results.

Furthermore, we hand labelled the different hand configurations for the whole sequence. This is used as the ground-truth. By looking at the means, see Figure 4b, we decided which class c corresponds to which hand labelled configuration. Then we compared for each image if the class with the highest posterior probability q_{cn} (see Section 4) is the same as the ground-truth label. As the result 435 from the 447 were correctly labelled. We also tried a Gaussian-based transformation invariant mixture of probabilistic PCA version similar to [6] and the results, from multiple trials using similar initial conditions, were never higher than 420 correctly classified frames. Binary PCA models the binary data more closely than the Gaussian version. This helps the mixture of binary PCAs to better separate the data clusters.

6. Conclusions and further work

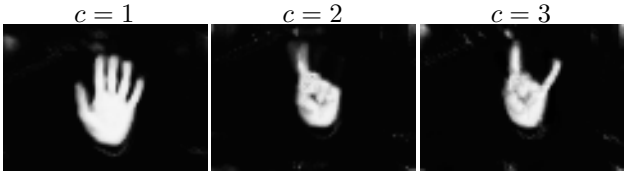
There are various situations where binary or close-to-binary images need to be analyzed. We presented the transformation invariant mixture of binary PCAs that model the binary images in a natural way using Bernoulli distributions. In our experiments we show that the binary PCA can reconstruct binary data much better than the normal PCA. Furthermore, we describe how the binary PCA can be simply extended to a potentially very useful transformation invariant version which simultaneously corrects the data for a set of global transformations and learns the binary PCA model on the aligned data.

A disadvantage of the binary PCA is that it must be performed using an iterative procedure (see Appendix A) while there is a closed form solution for the Gaussian-based PCA. This is less relevant in the transformation invariant version which is iterative in both cases. Still, the iterations require solving two $L \times L$ linear systems for each data point (see Appendix A) which might be prohibitive if the number of components L is large. Furthermore, just projecting data to the PCA low-dimensional space must be done iteratively



a) a few skin-color segmented images from the sequence $\sigma(\mu)$:

$c = 1$ $c = 2$ $c = 3$



W:

b) the model learned using the transformation invariant mixture of binary PCAs.

Figure 4. Illustration of transformation invariant mixture of binary PCAs. The model is learned from a sequence of a moving hand in three diferent configurations.

while for the Gaussian version this is computed directly.

Another disadvantage is that the binary PCA does not define a proper generative model that can be used to define conditional distribution on low dimensional coefficients given the data. Generative versions, e.g. [9], are computationally more expensive.

The MATLAB code for the transformation invariant mixture of binary PCAs will be available at: `http://staff.science.uva.nl/~zivkovic/`

Appendix I: Binary PCA update equations

U-update: First intermediate quantities are computed:

$$H_{nd} = \Theta_{nd}^{-1} \tanh(\Theta_{nd}/2) \quad (23)$$

$$A_{nll'} = \Sigma_d H_{nd} W_{ld} W_{l'd} \quad (24)$$

$$B_{nl} = \Sigma_d (2X_{nd} - 1 - H_{nd}\mu_d) W_{ld} \quad (25)$$

Row n of U is computed by solving linear system:

$$\Sigma_{l'} A_{nll'} U_{nl'} = B_{nl} \quad (26)$$

W-update: First intermediate quantities are computed:

$$A_{dll'} = \Sigma_n H_{nd} U_{nl} U_{nl'} \quad (27)$$

$$B_{dl} = \Sigma_n (2X_{nd} - 1 - H_{nd}\mu_d) U_{nl} \quad (28)$$

Column d of W is computed by solving the linear system:

$$\Sigma_{l'} A_{dll'} W_{l'd} = B_{dl} \quad (29)$$

μ-update:

$$\mu = (\Sigma_n H_{nd})^{-1} \Sigma_n (2X_{nd} - 1 - H_{nd}(UV)_{nd}) \quad (30)$$

References

[1] M. Black and A. Jepson. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 1(39):1–38, 1977.

[3] B. Frey and N. Jojic. Transformation-invariant clustering using the EM algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25:1–17, 2003.

[4] G. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

[5] N. Jojic, N. Petrovic, B. Frey, and T. Huang. Transformed hidden markov models: Estimating mixture models and inferring spatial transformations in video sequences. *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

[6] A. Kannan, N. Jojic, and B. Frey. Fast transformation-invariant component analysis (or factor analysis). *In Advances in Neural Information Processing Systems (NIPS)*, 2003.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] A. Schein, L. Saul, and L. Ungar. A generalized linear model for principal component analysis of binary data. *In Proc. Int. Workshop on Artificial Intelligence and Statistics*, pages 14–21, 2003.

[9] M. Tipping. Probabilistic visualisation of high-dimensional binary data. *In Advances in Neural Information Processing Systems (NIPS)*, 1999.

[10] M. Tipping and C. Bishop. Mixture of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–483, 1999.

[11] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[12] J. Wang and E. Adelson. Layered representation for motion analysis. *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 361–366, 1993.

[13] G. Wolberg and S. Zokai. Robust image registration using log-polar transform. *In Proc. IEEE Int. Conf. image processing, vol. 1*, pages 493–496, 2000.

[14] Z.Zivkovic and F. der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.