Learning to Segment Any Random Vector

Aapo Hyvärinen and Jukka Perkiö

***Abstract*— We propose a method that takes observations of a random vector as input, and learns to segment each observation into two disjoint parts. We show how to use the internal coherence of segments to learn to segment almost any random variable. Coherence is formalized using the principle of autoprediction, i.e. two elements are similar if the observed values are similar to the predictions given by the elements for each other. To obtain a principled model and method, we formulate a generative model and show how it can be estimated in the limit of zero noise. The ensuing method is an abstract, adaptive (learning) generalization of well-known methods for image segmentation. It enables segmentation of random vectors in cases where intuitive prior information necessary for conventional segmentation methods is not available.**

I. Introduction

Assume we observe a very high-dimensional random vector $\mathbf{x} = (x_1, \ldots, x_n)$, and we want to segment each observation $\mathbf{x}(t)$ into a small number of disjoint segments. That is, we want to associate a discrete label to each element $x_i(t)$ that tells which segment the i-th element belongs to in the current observation. While many methods exist for performing such a segmentation, they all require that the user defines a *similarity* measure between the different elements $x_i(t)$ based on some intuitive prior information. Here, our purpose is to show how the similarity relationships of the elements can be learned from the statistics of the random vector.

The classic application of segmentation methods is image segmentation (see e.g. [1], [2]), where the elements of $\mathbf{x}$ typically give the grey-scale values of the image at different spatial points. A large number of methods have been proposed for image segmentation in computer vision. Image segmentation is also effortlessly performed by our visual system. However, conventional image segmentation methods and neural models use a large amount of prior knowledge on the structure of images and the sampling structure of the image. Our purpose is to develop a system that can *learn* to perform such segmentation on almost any kind of random vectors.

The basic idea is illustrated in Figure 1. For humans, segmenting the image in a) is fast and "effortless". This is presumably because the visual system has learned the statistical regularities in natural images. Segmentation of data that has different statistical properties then becomes difficult or impossible for the visual system. This is illustrated in b), where the image in a) has been transformed so that the pixels have been randomly permuted (reordered). Finding the segment is now impossible for humans. However, we claim it is possible to learn to perform segmentation even in this latter case, because the statistical dependencies in a) and b) are similar except for a permutation. If the segmentation system is based on the statistical structure of the input, we can learn to segment the image in b) provided that we can observe many images that have the same statistical structure as the image in b).

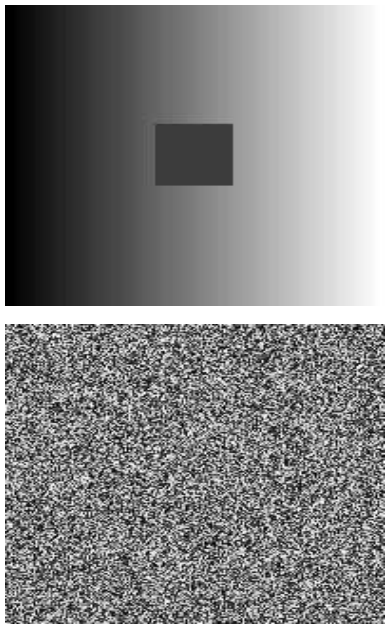

Fig. 1. Illustration of the easy segmentation of natural images and the necessity for learning. The image in a) is effortlessly segmented by our visual system. However, if we just permute the pixels, obtaining b), the segmentation is impossible for the human visual system. Our method is based on learning the statistical structure of the input, and therefore, it can segment a) and b) equally well, assuming that we can learn the statistical structure of the data by observing many typical images of the image class in question. For example, if we can observe many images that have been permuted in the same way as the image in b), we can segment them even though any segmentation structure seems to be completely absent.

We propose here a statistical model that generalizes the concept of segmentation to arbitrary random vectors. The basic assumption is that is is possible to define and learn a segmentation for any random vector by learning to use its statistical structure. At the same time, we hope to elucidate the probabilistic underpinnings of classic image segmentation, which can be useful for improving current image segmentation methods. However, our purpose is to propose a general-purpose data analysis method, so it is not likely to be competitive with image segmentation methods that exploit the special structure of images.

Our approach is quite different from methods that decompose a random vector into a small number of components,

The authors are with the Helsinki Institute for Information Technology (Basic Research Unit) and the Dept of Computer Science, University of Helsinki, Finland. Web page: http://www.cs.helsinki.fi/aapo.hyvarinen/

such as principal component analysis, factor analysis, and independent component analysis. These methods represent each observed data vector as a linear *combination* of a number of components, whose number is typically of the order of n, and the patterns (basis vectors or weight vectors) corresponding to each component are fixed. In contrast, segmentation represents each observed data vector as a single pattern of $+1$'s and -1's (in the basic case of two segments), but this pattern is not at all fixed: instead, it can take any of the 2^n different values possible for a binary vector.

How do we use the statistical structure of the data for segmentation? The basic principle is that the data inside each segment should be "coherent" in some intuitive sense of the word. To formalize this, we propose the notion of *autoprediction*, which means that the observed values inside the same segment predict each other well, whereas values cannot be predicted across segment borders. In the following, we first define the problem formally and discuss the definition of coherence in Section II. We propose a generative model in Section III and show how to estimate it in Section IV. Simulations are reported in Section V and experiments with real data in Section VI. Section VII discusses extensions and related methods, and Section VIII concludes the paper.

II. Problem definition and scope

A. Notation and basic approach

Let us first define the segmentation problem formally. The observed random vector is denoted by $\mathbf{x} = (x_1, \ldots, x_n)$. It is assumed that we have access to a large number of observations from which we can learn the statistical structure of the random vector. Let us denote by $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))^T$ an observation of the random vector $\mathbf{x}$. It is important to emphasize that we want to segment each $\mathbf{x}(t)$ separately, i.e. the segmentation will be different for each sample index t.

We shall here concentrate on the basic case of segmenting the vector into two parts. Then, the problem is to find a function

$$\mathbf{s} : \mathbf{x}(t) \rightarrow \{-1, +1\}^n \tag{1}$$

that has binary vector values. In the following, we often use the short-cut notation $\mathbf{s}(t)$ for $\mathbf{s}(\mathbf{x}(t))$, and $\mathbf{s}$ for $\mathbf{s}(\mathbf{x})$. The elements of $\mathbf{s}$ are denoted by s_i.

The function $\mathbf{s}$ should be such that the variables x_i for which $s_i(\mathbf{x}(t)) = +1$ form a "coherent whole", and likewise for those x_i for which $s_i(\mathbf{x}(t)) = -1$. Coherence should be a measure of how the segmentation follows the statistical structure of the random vector $\mathbf{x}$.

B. Coherence and segmentation

Coherence can be defined in many ways and different methods can be obtained. Intuitively, the central part in Figure 1 a) is coherent. Here, we discuss what kind of a definition of coherence we should find to fulfill two slightly conflicting purposes: 1) to be in line with the way human observers segment natural images 2) to be generalizable to other kinds of data sets.

In the simplest case, one could think that if some of the observed x_i are close to a given value, and others are close to another value, we can define two coherent segments using this closeness criterion, i.e. by clustering. However, this approach is not satisfactory. First, we do not know if this division is in line with the statistics of the input. For example, if, in a natural image, two pixels that are far away from each other have similar values, this does not mean that they are likely to belong to the same segment as defined by a human observer. Statistically, this is reflected in the fact that the pixels are *not* correlated: Similar gray-scale values for uncorrelated variables are not evidence for belonging to the same segment. Second, in natural images, two segments can be very clearly visible even if the values are quite different inside the segments. This is already seen in Fig. 1 where the background pixels have values in a large range, including those in the central "figure" segment. Thus, any kind of thresholding or clustering of pixel values is clearly insufficient as an account of segmentation by the human visual system.

Another point that we must consider is that while image segmentation algorithms consider the proximity relations between pixels, in the general case we do not have prior information on which variables are "close" to each other — the whole notion is meaningless in some cases. Any such proximity relations must be *learned* by a statistical analysis of the data.

A final point is that while in natural images correlations are typically positive, and coherence is related to pixels having similar values, this need not be the case for other kinds of data sets. We want our coherence criterion to be such that if x_i and x_j are strongly negatively correlated, i.e. typically have values of opposing signs, our coherence measure will consider it coherent for these two variables to have values of opposing signs. This means an invariance to the definition of the signs of the variables: in many kinds of data sets it may not be clear whether we should use x_i or $-x_i$ as input for some i.

To move towards a computational definition of coherence, let us define coherence of a segmentation by the following principle of *autoprediction*. The prediction of the observed value of any variable using observed values of other variables in the *same* segment (cluster, partition) should be as *good* as possible (small prediction error). In contrast, prediction of the observed value of the variable using observed values of variables that belong to a *different* segment should be as *poor* as possible (large prediction error). "Prediction" here means the best possible prediction using knowledge of the statistics of the random vector.

The principle of autoprediction brings us closer to a computational definition. In the following sections, we propose a generative model as well as a computationally simpler definition of coherence that take into account the discussion above.

III. Generative model for learning segmentation

A. Basic idea

Now, we define a simple generative model for $\mathbf{s}$ and $\mathbf{x}$ that incorporates the structure explained in the preceding section. The main point is that the data in the two segments are generated independently from each other. The same model is used for both segments, but two independent observations are sampled from (generated by) the model, one for each segment. We use a model that gives "autopredictable" data, i.e. data in which variables can be predicted from each other. This implies that inside a given segment, the variables can be predicted from each other. In contrast, the values of variables in the other segment cannot be predicted because the two segments were "filled in" by two independent samples.

B. Conditionally gaussian segments

Next, we describe the generative model in detail. In the first step of the generative model, the binary segmentation labels s_i are created. (The segmentation labels are considered latent random variables in this generative model, and the segmentation function is based on probabilistic inference of these random variables.) This gives a binary label vector for each observation (sample point). A large number of models is possible. For example, one could use different versions of the Boltzmann machine [3]. However, as will be seen below, the exact specification of this part is immaterial to the estimation, so we do not propose a detailed model here.

The fundamental question is, How should the data be generated, given the segmentation labels s_i? We propose that each segment is generated using a multivariate gaussian distribution, independently from each other. Let us denote a global covariance matrix of a gaussian distribution by $\mathbf{\Sigma}$. Now, given a realization of $\mathbf{s}$, consider the set of indices i for which $s_i = 1$. For those indices, we extract the corresponding submatrix from $\mathbf{\Sigma}$ by taking elements whose both row and column indices belong to that index set. We call that submatrix $\mathbf{\Sigma}_+$, and we generate a gaussian sample vector with the covariance matrix Σ_+. These values are then given to the $x_i(t)$ whose indices i belong to that same set where $s_i = 1$. Then we look at the remaining set of indices (for which $s_i = -1$), and generate a new gaussian vector using the corresponding submatrix of $\mathbf{\Sigma}$, independently from the previous gaussian vector. This gives the values for the remaining $x_i(t)$.

C. Simplified low-rank model

A great simplification of the algebraic developments below is obtained if we model the covariance matrix by a *subspace structure*, i.e.

$$\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^T \tag{2}$$

where the matrix $\mathbf{A}$ has only a small number k of columns. In other words, the generated gaussian data inside one segment is concentrated in a subspace of k dimensions. It must be emphasized that this does not imply that the observed data $\mathbf{x}$ would belong to a subspace of the data space because the total data generating process is nonlinear. We assume that $\mathbf{A}$ has orthogonal columns of unit norm. Some gaussian i.i.d. noise is further added to the data.

In other words, the data is generated as a linear combinations of some *prototypes* given by the columns of $\mathbf{A}$, and the coefficients for each prototype are different in the two segments (and in each observation of $\mathbf{x}$).

Thus, in this simplified low-rank model, we generate one observed data vector $\mathbf{x}(t)$, where t is the sample index, as follows. Generate the segment labels, i.e. a binary vector $\mathbf{s}(t)$. Generate a zero-mean gaussian vector $\mathbf{y}_+(t) = (y_+(1,t), ..., y_+(k,t))^T$ independently from each other, where k is the subspace dimension, i.e. the number of columns in $\mathbf{A}$. These are the coefficients of the columns of $\mathbf{A}$ inside the segment where $s_i = 1$. Likewise, generate independent zero-mean gaussian variables $\mathbf{y}_-(t) = (y_-(1,t), ..., y_-(k,t))^T$ (for the segment where $s_i = -1$). The variance of all these gaussian coefficients is chosen as σ_y^2. Create also independent gaussian noise variables n_i of variance σ_n. Then one observed data vector is generated as

$$x_i(t) = \begin{cases} \sum_j a_{ij} y_-(j,t) + n_i, & \text{if } s_i(t) = -1 \\ \sum_j a_{ij} y_+(j,t) + n_i, & \text{if } s_i(t) = +1 \end{cases} \tag{3}$$

The procedure is repeated for each observed data vector. All generated gaussian variables are independent from each other, and they are independent from the s_i which are also independent for all observed data vectors.

IV. Estimation of the model

Now we show how to estimate the model by maximum likelihood estimation. The complete likelihood of the data (for one data point), which depends on the latent variables $\mathbf{y}_-$, $\mathbf{y}_+$, and $\mathbf{s}$ as well, is given by

$$\begin{aligned} &\log p(\mathbf{x}, \mathbf{y}_-, \mathbf{y}_+, \mathbf{s}|\mathbf{A}) \\ &= \frac{1}{2\sigma_n^2}\|\mathbf{x} - \frac{1}{2}(\mathbf{1} - \mathbf{s}) \otimes (\mathbf{A}\mathbf{y}_-) - \frac{1}{2}(\mathbf{1} + \mathbf{s}) \otimes (\mathbf{A}\mathbf{y}_+)\|^2 \\ &\qquad + \log p(\mathbf{s}) - \frac{1}{2\sigma_y^2}(\|\mathbf{y}_-\|^2 + \|\mathbf{y}_+\|^2) + \text{const.} \end{aligned} \tag{4}$$

where $\mathbf{1}$ is a vectors of all ones, and $\otimes$ denotes component-wise multiplication of two vectors. The likelihood for the whole data set is, of course, obtained by summing this over all the observation of $\mathbf{x}$, and the corresponding inferred values of the latent variables. (For notational simplicity, we omitted the sample index t from $\mathbf{x}, \mathbf{y}_-, \mathbf{y}_+$ and $\mathbf{s}$.)

Now, we can estimate the parameter matrix $\mathbf{A}$ as well as the latent variables by maximization of the likelihood with respect to all the latent variables and $\mathbf{A}$, given observations of the $\mathbf{x}$.

To simplify the estimation, we assume that the noise level is very low, i.e. σ_n^2 is very small. Then, for any real data set, the weight on the reconstruction error, i.e. the first term in (4) dominates the others. This implies that the prior model for $\mathbf{s}$ is immaterial in the estimation of the model. The estimation is then basically a problem of optimal reconstruction of the data: for each observed variable in each observed data point, the latent variables are chosen to minimize distance between

the observed data point $\mathbf{x}(t)$ and its reconstruction either by $\mathbf{A}\mathbf{y}_+(t)$ or $\mathbf{A}\mathbf{y}_-(t)$.

The likelihood is simple to maximize after this additional simplification. The optimal value of $\mathbf{s}$, given $\mathbf{A}$ and other latent variables, can be computed in closed form:

$$\hat{\mathbf{s}}(t, \hat{\mathbf{A}}, \hat{\mathbf{y}}_-(t), \hat{\mathbf{y}}_+(t)) = \text{sign}(\mathbf{x}(t) \otimes (\hat{\mathbf{A}}(\hat{\mathbf{y}}_+(t) - \hat{\mathbf{y}}_-(t)) - \frac{1}{2}(\hat{\mathbf{A}}\hat{\mathbf{y}}_+(t))^2 + \frac{1}{2}(\hat{\mathbf{A}}\hat{\mathbf{y}}_-(t))^2) \quad (5)$$

The squares in this formula mean taking squares of all elements of the vector.

The optimal values for $\mathbf{A}$ as well as for $\mathbf{y}_-$ and $\mathbf{y}_+$ can be computed by simple pseudoinverse computations. First, let us compute the optimal $\mathbf{y}$, given $\mathbf{A}$. Optimization of the likelihood is then a simple regression problem that can be solved with classic pseudoinverses. Denote by $I(t)$ the set of indices (for a single sample point) for which $\hat{s}_i(t) = 1$. Denote by $\mathbf{A}_I$ a matrix which contains the rows of $\mathbf{A}$ whose index is in $I(t)$, and likewise by $\mathbf{x}_{I(t)}$ a vector which contains the elements of $\mathbf{y}_+(t)$ whose indices are in $I(t)$. Then, we simply compute

$$\hat{\mathbf{y}}_+(t) = \hat{\mathbf{A}}^{pinv}_{I(t)} \mathbf{x}_{I(t)} \quad (6)$$

where $\mathbf{M}^{pinv}$ denotes the Moore-Penrose pseudoinverse of the matrix $\mathbf{M}$. The computation of the optimal $\mathbf{y}_-$ is completely analogue.

$$\hat{\mathbf{y}}_-(t) = \hat{\mathbf{A}}^{pinv}_{\bar{I}(t)} \mathbf{x}_{\bar{I}(t)} \quad (7)$$

where $\bar{I}(t)$ denotes the complement set of the index set $I(t)$.

The computation of the optimal $\mathbf{A}$ is straightforward as follows. Denote one row of $\mathbf{A}$ by $\mathbf{A}_i$. For a given index $i \in \{1 \dots n\}$, compute a matrix $\tilde{\mathbf{Y}}$ of size $n \times t$ which is such that each element is equal to $\hat{y}_+(i,t)$ if $\hat{s}_i(t) = 1$ and otherwise equal to $\hat{y}_-(i,t)$. Thus, it collects the estimates of the coefficients really used in the data generation process. Denote by $\mathbf{X}_i$ an t-dimensional vector $(x_i(1), \dots, x_i(T)$. Then, the estimate of $\mathbf{A}_i$ is obtained as

$$\hat{\mathbf{A}}_i = \mathbf{X}_i \tilde{\mathbf{Y}}^{pinv} \quad (8)$$

After computing this for all i, the matrix has to be orthogonalized using the classic method:

$$\hat{\mathbf{A}} = \hat{\mathbf{A}}(\hat{\mathbf{A}}^T \hat{\mathbf{A}})^{-1/2} \quad (9)$$

See, e.g. [4] on the computation of the matrix square root.

Thus, the estimation of the parameters and the latent variables is done by cycling though the updates in Equations (5,6,7,8,9). Note that these updates need no stepsizes or other parameters.

For fast and reliable convergence, a good initial point is important. We propose that the columns of $\mathbf{A}$ are initially set to the k first principal components, which we use in the simulations and experiments below. The $\hat{\mathbf{s}}$ could be initialized randomly, or perhaps by setting $\hat{s}_i(t) = 1$ for all i and t, which we used below. Using (6,7), one can then compute the initial $\mathbf{y}_+(t)$ and $\mathbf{y}_-(t)$ and start the iteration.

V. Simulations

To test our estimation method, we created artificial data according to the generative model and then estimated it using the method proposed above.

The dimension of the data was $n = 50$, and the subspace dimension k was taken to be three. We had two different conditions. In the first, "random" one, we used a completely random prior in which all $\mathbf{s} \in \{-1, +1\}^n$ have equal probabilities. In the second, we used a Boltzmann machine prior in which we created by intuitively appealing dependencies by arranging the variables on a ring-like structure, and defining the connection b_{ij} between a variable and its 6 nearest neighbours to equal 1, and all other connections to equal 0. The columns of $\mathbf{A}$ were randomly chosen in both conditions. The variances were chosen as $\sigma_y = 1$ and $\sigma_n = .01$.

An illustration of a vector created in the Boltzmann condition is shown in Figure 2 a), which shows $\mathbf{s}$ and b), which shows the $\mathbf{x}$. In total, 1000 such random vectors were created in both conditions.

The task was now to learn to find the original segments in each vector. The resulting segmentation for the vector in Figure 2 b) is shown in Figure 2 c). We can see that the segmentation coincides with the true segmentation that is visually evident in Figure 2 a) because we chose a visually intuitive Boltzmann machine structure.

To assess the quality of the segmentation for the whole ensemble of 1000 vectors, the obtained segmentations $\hat{\mathbf{s}}(t)$ were compared to the true segmentations $\mathbf{s}(t)$ used in generating the data. This was computed by simply counting what proportion of elements in the two vectors was equal. Thus, we obtained 1000 different numbers that should be close to 1. Actually, since the segmentation is symmetric with respect to the "figure" and the "ground", we have to compute the maximum between the similarity of $\hat{\mathbf{s}}(t)$ and $-\hat{\mathbf{s}}(t)$ to the true segmentation.

Histograms of the matches are shown in Figure 3 for the random condition, and in Figure 4 for the Boltzmann condition. approximately 90% of the vectors had a match of more than 80%. Thus, our method found the original segments with reasonable accuracy.

VI. Experiments on real images

We also tested our method in segmentation of grey-scale random images. One should note, however, that this is merely an illustration of our method on real data and the method is not proposed as a serious competitor for existing image segmentation methods which use a lot of prior information on the structure of images and do not try to learn it from scratch.

We used images from Hans van Hateren's natural stimuli collection[1] as well as some images from the CalTech human face and background datasets. [2] All the images were converted to greyscale and rescaled so that they contain approximately 65,000 pixels.

[1] http://hlab.phys.rug.nl/archive.html, see [5]

[2] http://www.vision.caltech.edu/html-files/archive.html

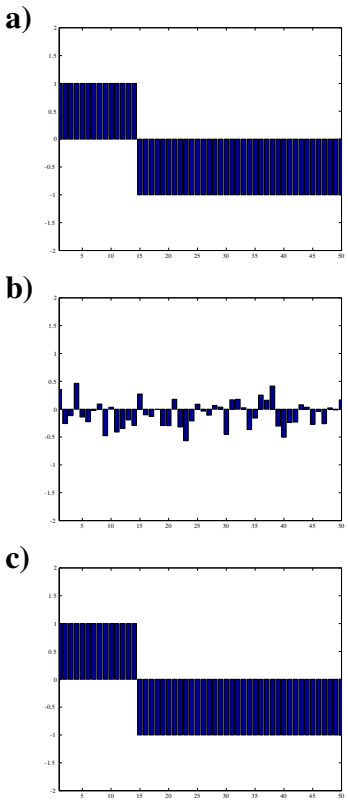

Fig. 2. Illustration of segmentation of one observation. Horizontal axis is index i in all plots. Vertical axis is value of the i-th element of the vector plotted. **a)** The original segmentation label vector $\mathbf{s}$ for one observation in the Boltzmann condition. For visualization purposes, the covariance structure follows the vector indices. **b)** The observed data vector $\mathbf{x}$ **c)** The estimated segmentation $\hat{\mathbf{s}}$ found by our method (the sign of s_i is immaterial).

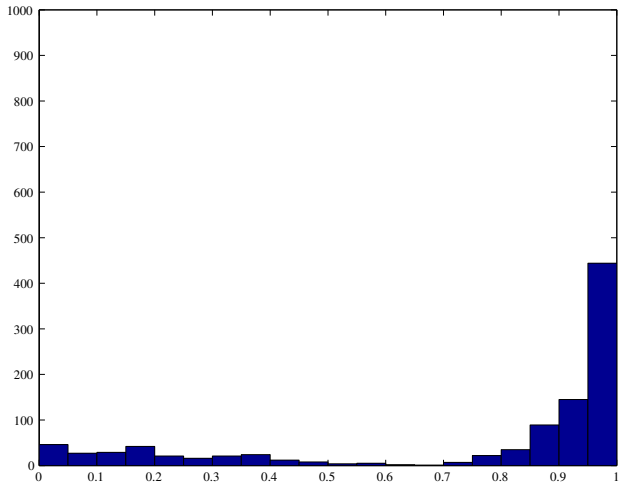

Fig. 3. Accuracy of segmentation in the random condition. Histogram of matches between estimated segmentations and the true underlying segmentations with artificial data. Horizontal axis: correlation coefficient between underlying true segmentation and its estimate.

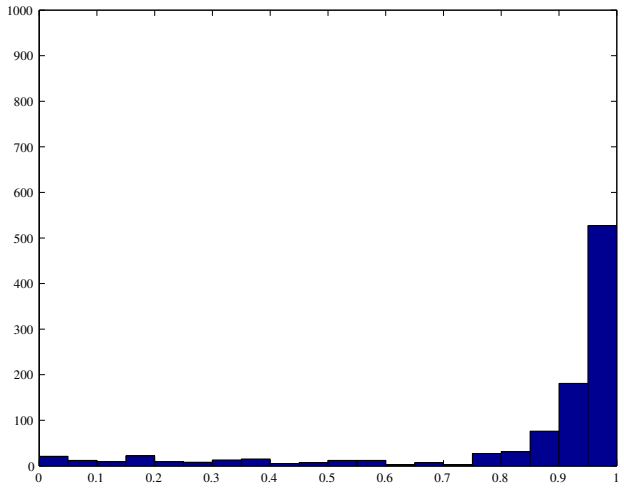

Fig. 4. Accuracy of segmentation in the Boltzmann condition. Histogram of matches between estimated segmentations and the true underlying segmentations with artificial data. Horizontal axis: correlation coefficient between underlying true segmentation and its estimate.

Due to the overwhelming computational complexity of learning to segment full images, we considered segmentation of 32×32 pixel patches only. The subspace dimension was set to three.

We considered each image separately, treating patches sampled from each image as a separate random variable. The model parameters were separately learned for each full-size image using 10,000 randomly sampled patches from that image. We then took a smaller number of patches from the images to be segmented.

The estimated basis vectors for three images are shown in Fig. 5. It turns our that the basis vectors are not very different from the PCA basis vectors. This does not seem to be an algorithmic artefact, because it is intuitively appealing: The basis vectors typically describe a DC component, and vertical and horizontal edges, which seems to be reasonable in image segmentation. The edges are actually much sharper than the PCA basis vectors.

Some results for segmenting image patches are shown in Figure 6. The results are reasonably well in line with human segmentation. However, the basis vector similar to the DC component seems to be quite dominant which is seen in the fact that the segmentations are not very different from a simple thresholding. Thus, the method may not be very useful for natural grey-scale images, which merely serve as an illustration of and inspiration for our method.

VII. Discussion

a) Multidimensional locations: An important extension of the present method is to the cases where the elements to be segmented are multidimensional. For example, in color images every spatial location has typically three RGB values. It would be meaningless to consider each value separately, in which case each pixel would get three segmentation results. Likewise, texture detectors can be used to associate texture information to each spatial location. One of the advantages of our method is that it can be readily extended to this multidimensional case. We only need to consider assign the

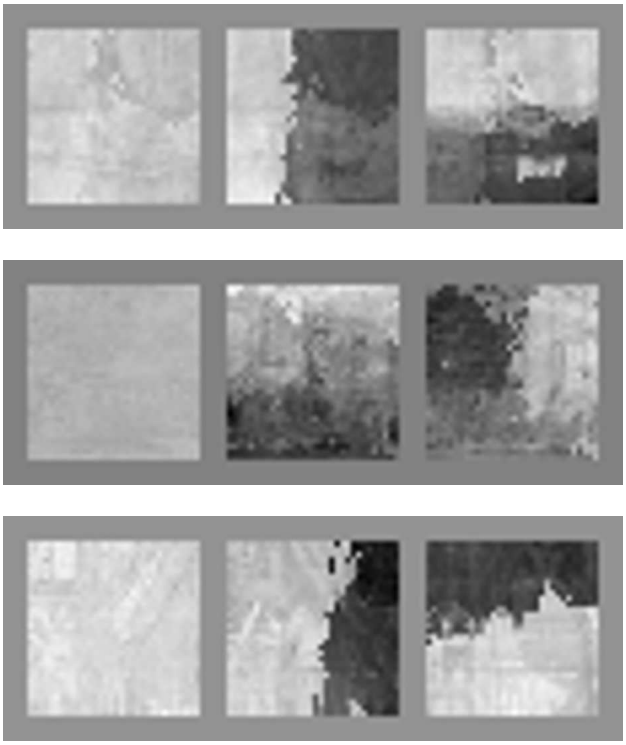

Fig. 5. Estimated basis vectors for three natural images, with subspace dimension equal to three. Each row shows the basis vectors for one image.

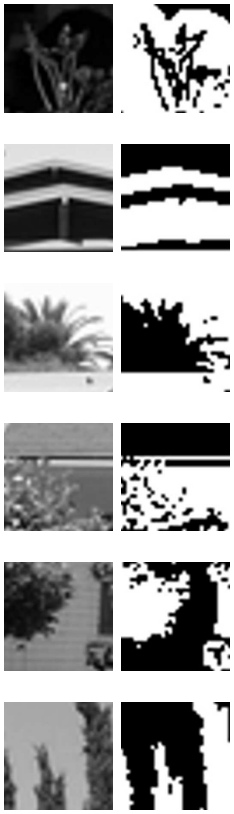

Fig. 6. Segmentation results for natural images are shown for six patches of 32×32 pixels taken from relatively natural images. The left-hand displays show the original patches, and the right-hand displays show the resulting segmentation, black and white giving the two segments.

same s_i for each of the, say q, variables that all belong to the same spatial location. In practice, this leads to a very simple modification of the algorithm, where each $\hat{s}_i$ is computed by first computing q signs as in (5), and finally assigning the sign that get more "votes" to $\hat{s}_i$.

b) Related work: A related method was proposed in [6] where the purpose was to segment multivariate time-series so that the segmentation need not be the same for all the series. Implicitly, this then introduces a segmentation of the multivariate time series for each time step. This method is quite different from ours in that it is based on the temporal structure of the time series. A complementary approach to ours is provided by [7], in which the similarity matrix is learned using examples of segmented vectors. This is in contrast to our approach which is completely unsupervised.

c) Future work: An important question for future work is what kind of data sets might be easily segmented using our method. We believe such applications can probably be found in natural language processing (segmentation of text documents), bioinformatics (segmentation of micro-arrays), and multivariate econometric data sets. Another important question is how to estimate more than two segments. Also, the method is very noise-sensitive because it is based on approximation of low noise, which also leads to the total neglect of a prior model for the segments, $p(\mathbf{s})$. Future work should consider interesting priors for $\mathbf{s}$ and incorporate their estimation to the method. This should a be straightforward addition to the likelihood; the problems are mainly computational because then computation of the optimal $\hat{\mathbf{s}}$ is much more complicated than in Eq. (5).

VIII. Conclusion

We proposed a new method of multivariate data analysis that is based on learning to segment observations of any random vector, based on its statistical structure. This can also be seen as an improvement of image segmentation methods by introducing the aspect of adaptation to the method. We formulated a generative model to accomplish the task, and showed how it can be simply estimated in a low-noise approximation. Simulations and experiments on real data confirmed the validity of the method.

References

[1] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1101–1113, 1993.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[3] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[4] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley Interscience, 2001.

[5] J. H. van Hateren and A. van der Schaaf, "Independent component filters of natural images compared with simple cells in primary visual cortex," *Proc. Royal Society, Ser. B*, vol. 265, pp. 359–366, 1998.

[6] A. Gionis, H. Mannila, and E. Terzi, "Clustered segmentations," in *3rd Workshop on Mining Temporal and Sequential Data*, 2004.

[7] F. Bach and M. Jordan, "Learning spectral clustering," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.