Probabilistic Sequential Independent Components Analysis

Max Welling
Richard S. Zemel
Geoffrey E. Hinton

Dept. of Computer Science
University of Toronto
6 King's College Road
Toronto, M5S 3G4

September 14, 2004

Abstract

Under-complete models, which derive lower-dimensional representations of input data, are valuable in domains in which the number of input dimensions is very large, such as data consisting of a temporal sequence of images. We present the "under-complete product of experts" (UPoE), where each expert models a one dimensional projection of the data. Maximum-likelihood learning rules for this model constitute a tractable and exact algorithm for learning under-complete independent components. The learning rules for this model coincide with approximate learning rules proposed earlier for under-complete ICA models. We also derive an efficient sequential learning algorithm from this model, and discuss its relationship to sequential ICA, projection pursuit density estimation, and feature induction algorithms for additive random field models. We demonstrate the efficacy of these novel algorithms on high-dimensional continuous datasets.

1 Introduction

The fundamental aim of unsupervised learning is to find useful representations or transformations of data. Considerable progress has been achieved using the simplest such form of representation, a linear transformation. Methods such as principal components analysis, factor analysis, and projection pursuit search for a linear transformation of the observed variables that has some suitable properties, typically formulated in terms of information-theoretic criteria, such as entropy maximization, or minimization of mutual information between the components of the representation.

Perhaps the most significant development in unsupervised learning over the last decade has been the independent components analysis (ICA) algorithm, or family of algorithms. These algorithms also search for linear transformations of data, where here the transformed components are as statistically independent as possible. In this paper, we propose a new learning algorithm that can be seen as a fast, efficient, and principled method to perform a particular form of ICA: *under-complete* ICA. Before describing our new method, we motivate the emphasis on this form.

Three forms of ICA exist, based on the relationship between the number of input dimensions (D) and hidden dimensions (J): square ($J = D$), over-complete ($J > D$), and under-complete ($J < D$). The standard ICA model applies to the square case [Bell and Sejnowski, 1995]; however, the assumption that $J = D$ is restrictive. Over-complete ICA methods [Olshausen and Field, 1997, Lewicki and Sejnowski, 1998, Teh et al., 2003] are motivated by signal and image processing approaches, which involve multi-scale, redundant basis sets. However, domains in which the number of input dimensions is very large, such as data consisting of a temporal sequence of images, call for under-complete methods, which reduce the input dimensionality.

In this paper we propose a novel algorithm for learning under-complete independent components, based on optimizing a probabilistic density model of the data. Our components are independent under a different stochastic model that is more tractable than earlier approaches. We show that the learning rules for this model coincide with *approximate* learning rules proposed before for under-complete ICA models. Moreover, we derive an efficient and principled sequential learning algorithm that searches for the maximum decrease in Kullback-Leibler divergence between the data and model distribution, given what the model has learned so far. When this algorithm is run until the number of experts is equal to the number of input dimensions, it constitutes a sequential learning algorithm for square ICA.

We derive both a parallel and a sequential algorithm for learning the components, but focus on the latter. Sequential methods for learning components are attractive for a number of reasons. First, they contain the potential for a form of model selection. If some metric exists for evaluating if adding a new component is worthwhile, i.e., the advantage in data representation ability outweighs the increase in model complexity, then the method contains an automatic stopping criterion. Second, the learning algorithms are generally much simpler, involving the learning of only a few parameters rather than all parameters of all components simultaneously. Finally, this simplicity often translates into computational efficiency.

The rest of this paper is organized as follows. We first describe the under-complete ICA model in more detail, and present our new objective and learning algorithms, both parallel and sequential versions. We then present empirical results of our model, and relate it to a number of other methods for sequentially learning components, such as projection pursuit and maximum entropy models. We conclude by suggesting applications and extensions of this approach.

2 Under-complete ICA

Independent components analysis refers to a family of algorithms whose aim is to find directions in input space that are statistically independent. Let $\mathbf{x}$ be a multivariate random vector distributed according to $p(\mathbf{x})$, then, we wish to find a linear transformation, $\mathbf{s} = \mathbf{W}\mathbf{x}$, for which $\mathbf{s}$ is approximately factorized: $p(\mathbf{s}) \approx \prod_i p_i(s_i)$.

Algorithms for standard (i.e., square, noiseless) independent components analysis have been developed from both a *causal* probability density model approach, utilizing stochastic hidden variables, and from an information-theoretic approach, which does not posit any probabilistic model for the data. Recently, a third formulation of ICA has been proposed, utilizing a probability density model without stochastic hidden variables. This "product of experts" model generalizes differently to over-complete representations [Hinton et al., 2001] than earlier models.

Under-complete ICA algorithms, including our new algorithm presented here, fall into the same three categories. We briefly review previous approaches to UICA, and then present our method, based on a probability density model without stochastic hidden variables.

2.1 UICA by dimensionality reduction

A simple way to deal with fewer source variables $\{s_i\}$ than input dimensions is to reduce the dimensionality of the input space before applying one of the ICA methods designed for the square case (equal number of source variables and input dimensions). If one believes that the interesting structure of the data is captured by the high variance subspace, then principal components analysis can be used to reduce the dimensionality. However, this is a strong assumption: this approach will automatically discard low variance directions, which may turn out to be interesting. One would like to be able to search through all dimensions for independent components.

In this paper we therefore take the point of view that *a priori* all dimensions are equally important. A convenient way to factor out the influence of variance is to "sphere" or "whiten" the data, which is indeed common practice for almost all ICA algorithms. Technically, this is performed by computing an eigenvalue decomposition of the sample covariance matrix:

$$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T \tag{1}$$

and transforming to a new basis by,

$$\mathbf{x}'_n = \mathbf{R}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{x}_n \tag{2}$$

where $\mathbf{\Lambda}^{-\frac{1}{2}}$ is a diagonal matrix with one over the square-root of the eigenvalues on the diagonal. Although $\mathbf{R}$ is an *arbitrary* orthonormal matrix (i.e. a rotation), typical choices are given by $\mathbf{R} = \mathbf{I}$ or $\mathbf{R} = \mathbf{U}$.

2.2 Causal density models for UICA

Standard probabilistic models for ICA take the form of causal generative models where a number of sources $\{s_j\}$, distributed according to the prior distributions $p_j(s_j)$, are linearly combined to produce the observed random variables $\mathbf{x} = \mathbf{A}\mathbf{s}$. The model is given by:

$$p(\mathbf{x}|\mathbf{A}) = \int_{\mathbf{s}} p(\mathbf{x}|\mathbf{s},\mathbf{A}) \prod_{j=1}^{J} p_j(s_j) d\mathbf{s} \tag{3}$$

where $p(\mathbf{x}|\mathbf{s},\mathbf{A})$ is the likelihood term, which models the noise of the observed variables. In the square noiseless case, the sources are estimated using a linear model of the data, $\mathbf{s} = \mathbf{W}\mathbf{x} = \mathbf{A}^{-1}\mathbf{x}$. Under these assumptions, the model simplifies to:

$$p(\mathbf{x}|\mathbf{W}) = \int_{\mathbf{s}} \delta(\mathbf{x} - \mathbf{A}\mathbf{s}) \prod_{j=1}^{J} p_j(s_j) d\mathbf{s} = |\mathbf{W}| \prod_{j=1}^{J} p_j(\mathbf{w}_j^T\mathbf{x}) \tag{4}$$

where $|\cdot|$ is the absolute value of the determinant, and $|\mathbf{W}|$ normalizes for the change in volume induced by the transformation. The vector $\mathbf{w}_j$ is the j'th row of the matrix $\mathbf{W}$.

A gradient ascent learning rule for the transformations $\mathbf{W}$ can be derived from a maximum-likelihood approach [Pearlmutter and Parra, 1996]:

$$\frac{\partial L}{\partial \mathbf{W}} = \left\langle \frac{\partial \log p(\mathbf{x}|\mathbf{W})}{\partial \mathbf{W}} \right\rangle_{\tilde{p}} = \mathbf{W}^{-T} + \left\langle \sum_{j=1}^{J} \frac{\partial \log p_j(s_j)}{\partial s_j} \mathbf{x}^T \right\rangle_{\tilde{p}} \tag{5}$$

where $\langle \cdot \rangle_{\tilde{p}}$ is an average with respect to the empirical data distribution $\tilde{p}$. Assuming particular forms of sub- or super-Gaussian $p_j(s_j)$ leads to specific types of components.

For under-complete models, where the number of sources is smaller than the number of observed variables, observation noise is necessary to make the probability distribution proper, i.e., normalizable over input space (Eq.3). Formally, it is easy to write down an EM algorithm for learning the parameters of this model, i.e. the mixing matrix $\mathbf{A}$ and the parameters determining the shape of the prior distributions $p_i(s_i)$. However, in practice the required computations are intractable due to the need to compute the posterior distribution $p(\mathbf{s}|\mathbf{x})$ and integrate over it. An approximate maximum likelihood learning rule may be derived by replacing the stochastic relation between the source variables and input variables by a deterministic one of the form $s^* = \mathbf{W}\mathbf{x} = \mathbf{A}^{\#}\mathbf{x}$ where $\mathbf{A}^{\#} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the pseudo-inverse [Ridder et al., 2002]. Including an extra volume factor $\sqrt{|\mathbf{A}^T\mathbf{A}|} = \sqrt{|\mathbf{W}\mathbf{W}^T|}$ due to the change in variables and assuming isotropic Gaussian noise of unit variance[1], the approximate maximum-likelihood learning rule simplifies to a form similar to the square causal model:

$$\frac{\partial L}{\partial \mathbf{W}} \approx \mathbf{W}^{\#T} + \left\langle \frac{\partial \log p_j(s_j^*)}{\partial s_j^*} \mathbf{x}^T \right\rangle_{\tilde{p}} \tag{6}$$

In the language of graphical models, the noisy causal generative model can be described by a two-layer directed acyclic graph as depicted on the left in figure 1. From this we confirm that the model assumes marginal independence between the sources: $p(\mathbf{s}) = \prod_i p_i(s_i)$. We also see that given an observed data-vector, the sources exhibit dependencies due to the phenomenon of explaining away: $p(\mathbf{s}|\mathbf{x}) \neq \prod_i p_i(s_i|\mathbf{x})$.

2.3 Mutual information minimization for UICA

Algorithms for under-complete ICA have also been derived from an information-theoretic formulation. Under this formulation, the components are no longer stochastic random variables s_j for a given input $\mathbf{x}$, but instead are deterministic linear functions of the inputs, $z_j = \mathbf{w}_j^T\mathbf{x}$. One approach in this framework minimizes a natural measure of the information, the mutual information between the components:

$$I(z_1, z_2, ...z_J) = \sum_{j=1}^{J} H(z_j) - H(\mathbf{z}) \tag{7}$$

Where the joint entropy $H(\mathbf{z}) = -\int f(\mathbf{z}) \log f(\mathbf{z}) d\mathbf{z}$, and $H(z_j) = -\int f_j(z_j) \log f_j(z_j) dz_j$ is the marginal entropy of component j. I is non-negative and equals zero when the components are completely independent.

In the noiseless square case, the output entropy $H(\mathbf{z}) = H(\mathbf{x}) + \left\langle \log(|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}|) \right\rangle$, where $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}|$ is the Jacobian of the transformation $\mathbf{x} \to \mathbf{z}$. Because $H(\mathbf{x})$ is independent of the parameters $\{\mathbf{w}_j\}$ and if we again assume

[1] We assume that the data has been sphered in a pre-processing step.

that the data is sphered in a preprocessing step, the objective can be simplified to:

$$I(\mathbf{z}|\mathbf{W}) = C - \left\langle \log(|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}|) - \sum_j \log f_j(z_j) \right\rangle_{\tilde{p}} = C - \log|\mathbf{W}| - \left\langle \sum_j \log f_j(z_j) \right\rangle_{\tilde{p}} \tag{8}$$

When $f_j(z_j)$ matches $p_j(s_j)$, then minimizing $I(\mathbf{z}|\mathbf{W})$ with respect to $\mathbf{W}$ yields the same learning rule for $\mathbf{W}$ as in the square causal density model above (Equation 5).

In the under-complete case, the simple linear relationship between the densities of $\mathbf{z}$ and $\mathbf{x}$ is not valid, so deriving a learning rule using mutual information minimization requires calculating the entropy $H(\mathbf{z})$, which is difficult. A simple approximation such as a Gaussian may be used [Lu and Rajapakse, 2000], in which case the entropy $H(\mathbf{z})$ is proportional to the covariance of $\mathbf{z}$: $C = \mathbf{W}Cov[\mathbf{x}]\mathbf{W}^T = \mathbf{W}\mathbf{W}^T$ (assuming again that the data is sphered). An objective function may be defined that approximates $I(\mathbf{z}|\mathbf{W})$:

$$I(\mathbf{z}|\mathbf{W}) \approx -\frac{1}{2}\log(|\mathbf{W}\mathbf{W}^T|) - \left\langle \sum_{j=1}^{J} \log f_j(z_j) \right\rangle_{\tilde{p}} \tag{9}$$

Differentiating this objective with respect to $\mathbf{W}$ yields the gradient ascent learning rule

$$-\frac{\partial I}{\partial \mathbf{W}} \approx \mathbf{W}^{\#T} + \left\langle \sum_j \frac{\partial \log f_j(z_j)}{\partial z_j} \mathbf{x}^T \right\rangle_{\tilde{p}} \tag{10}$$

If $f_j(z_j)$ matches $p_j(s_j)$, then this learning rule is equivalent to the undercomplete ICA rule for the causal stochastic model (Equation 6).

A very similar derivation of Eqn. 10, based on the approach of maximizing the entropy of non-linear transformations of the input variables (see [Bell and Sejnowski, 1995] for the square case), was proposed in [Stone and Porrill, 1998].

2.4 A sequential information-theoretic algorithm for UICA

FastICA [Hyvarinen, 1999] is a fast sequential method for finding independent components, derived from the same objective of minimizing the mutual information of the components. Finding an invertible transformation $\mathbf{W}$ that minimizes $I(\mathbf{z}|\mathbf{W})$, the mutual information between the components, is equivalent to finding orthogonal directions in which $J(z_j)$, a normalized form of entropy known as *negentropy*, is minimized [Comon, 1994]:

$$J(z_j) = H(\nu_j) - H(z_j) \tag{11}$$

where ν_j is a Gaussian random vector with the same covariance matrix as z_j.

FastICA is a fixed-point algorithm that minimizes mutual information by iteratively finding projections $\mathbf{w}_j$ that maximize $J(z_j)$, and forcing them to be orthogonal. In order to optimize this objective, $H(z_j)$ must be approximated. Hyvarinen [Hyvärinen, 1998] suggests an approximation that is more robust than conventional cumulant-based approximations, which leads to an objective for each component $\mathbf{w}_j$ (row of the transformation matrix $\mathbf{W}$):

$$J(\mathbf{w}_j) = [\langle F(\mathbf{w}_j^T\mathbf{x})\rangle - \langle F(\nu)\rangle]^2 \tag{12}$$

where $F()$ is a non-quadratic function. The components are estimated one-by-one using a deflation scheme, which ensures that the components are decorrelated. Choosing $F()$ as the log of a sub- or super-Gaussian density $f()$ leads to the formation of the corresponding form of components.

The resulting fixed-point algorithm is fast and efficient. However, approximations are required to compute the negentropy and there is no probabilistic model associated with the discovered components.

3 Under-complete Product of Experts (UPoE)

The "Product of Experts" (PoE) models data as products of one-dimensional projections [Hinton, 1999]. The model has recently been extended to develop over-complete representations [Hinton and Teh, 2001, Teh et al., 2003]. While the results of this approach are impressive, the model parameters are hard to learn, and approximate methods are required. This paper introduces the Under-complete Product of Experts (UPoE) model, an efficient and exact algorithm for learning under-complete representations under the same model structure.

The UPoE model provides an alternative form of probability density model for under-complete ICA. Rather than a causal generative model, as in Section 2.2, it defines a data model without stochastic hidden variables. We show below that the learning rules for the parallel version of this model directly match those derived as approximations to the under-complete ICA models presented above. In addition, the sequential version of our model is also closely analogous to the FastICA method. However, the objective at each stage of the UPoE model is based on the maximal decrease in the log-likelihood of a density model.

3.1 Probability model for UPoE

The UPoE model consists of a number (J) of "experts" modelling certain projections of the input space. We will denote a 1-dimensional expert with $\mathcal{T}(z_j|\alpha_j)$, where $z_j = \mathbf{w}_j^T\mathbf{x}$ is the projection of $\mathbf{x}$ onto the vector $\mathbf{w}_j$ and α_j represent additional parameters used to model the shape of the expert. These parametrized experts play the role of the density functions $f_j(z_j)$ in the earlier UICA formulations.

As before, we assume that the data has been preprocessed such that the mean is zero and the covariance matrix is equal to the identity (i.e. the data has been "sphered"). This has the effect that "interesting" dimensions (dimensions that we like to model with an expert) with small variance are more easily detected.

In the UPoE these experts, or components, are combined by taking their product. When $J < D$ this does however not constitute a proper, normalizable probability distribution in D dimensions. To repair this we fill the *remaining* dimensions (indicated by $\mathbf{y}$) with uncorrelated Gaussian "noise" of unit variance. Note that this is a slightly different procedure than what is usually done for causal generative ICA models where noise is added to *all* directions. We want to stress that the term noise should not be taken too literally, as we are not trying to model a physical noise process. In the present context it simply means that we do not wish to model that direction explicitly with an expert and consequently ignore all its statistical properties higher than second order.

According to the above comments the UPoE model thus becomes,

$$p(\mathbf{y}, \mathbf{z}) = \prod_{i=1}^{D-J} \mathcal{N}(y_j|0,1) \prod_{j=1}^{J} \mathcal{T}(z_j|\alpha_j) \tag{13}$$

where $y_i = \mathbf{v}_i^T\mathbf{x}$ is a projection of $\mathbf{x}$ onto the vector $\mathbf{v}_i$. The vectors $\{\mathbf{v}_i\}$ form an orthonormal basis in the orthogonal complement of the space spanned by the vectors $\{\mathbf{w}_j\}$. While the vectors $\{\mathbf{v}_i\}$ are orthonormal, the vectors $\{\mathbf{w}_j\}$ associated with the experts can have arbitrary length and direction. If we collect the vectors $\{\mathbf{v}_i\}$ as rows in a matrix $\mathbf{V}$ and similarly for $\{\mathbf{w}_j\}$ in a matrix $\mathbf{W}$ we have the following relation,

$$\mathbf{V}^T\mathbf{V} = \mathbf{I} - \mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}\mathbf{W} \tag{14}$$

where $\mathcal{P} = \mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}\mathbf{W}$ is the projection operator on the space spanned by the vectors $\{\mathbf{w}_j\}$ and $\mathcal{P}^\perp = \mathbf{I} - \mathcal{P}$ is the projection operator onto the orthogonal complement of that space, i.e. the space spanned by the basis vectors $\{\mathbf{v}_i\}$, which is indeed given by $\mathbf{V}^T\mathbf{V}$.

Since the data are provided in $\mathbf{x}$ space we like to express the probability density in those variables. Transforming variables in densities involves a volume factor (Jacobian) for the transformation $\mathbf{x}^T \rightarrow [\mathbf{y}^T, \mathbf{z}^T]$ as follows,

$$p(\mathbf{x}) = p(\mathbf{y}, \mathbf{z}) \left| \frac{\partial(\mathbf{y}, \mathbf{z})}{\partial \mathbf{x}} \right| \tag{15}$$

Using that $|\mathbf{A}| = \sqrt{|\mathbf{A}\mathbf{A}^T|}$, with $\mathbf{A}^T = [\mathbf{V}^T|\mathbf{W}^T]$, $\mathbf{V}\mathbf{W}^T = \mathbf{W}\mathbf{V}^T = 0$ and[2] $\mathbf{V}\mathbf{V}^T = \mathbf{I}$ we arrive at,

$$p(\mathbf{x}) = \prod_{i=1}^{D-J} \mathcal{N}(\mathbf{v}_i^T\mathbf{x}|0,1) \prod_{j=1}^{J} \mathcal{T}(\mathbf{w}_j^T\mathbf{x}|\alpha_j) \sqrt{|\mathbf{W}\mathbf{W}^T|} \tag{16}$$

We note that the model has no hidden variables and has a simple normalization factor $Z = 1/\sqrt{|\mathbf{W}\mathbf{W}^T|}$.

In the limit that $J = D$ we note that the UPoE model (Eqn.16) becomes identical to the square noiseless ICA model (Eqn.4) if we identify the priors p_i with the experts $\mathcal{T}_i$ and use the fact that $\sqrt{|\mathbf{W}\mathbf{W}^T|} = |\mathbf{W}|$.

In contrast to the causal generative model for UICA which follows the semantics of a directed acyclic graph, the UPoE can be described as the limit of an undirected graphical model (see figure 1 on the right), where the values of the nodes in the top layer are deterministic (rather than stochastic) functions of the values of the nodes in bottom layer. This limit can also be viewed as a *factor graph* [Kschischang et al., 2001] where the factor nodes in the top-layer determine the interaction between the variables nodes in the bottom-layer. The top-layer nodes are marginally independent due to Eqn.13, but since they are deterministically related to the bottom-layer nodes there is no explaining away effect.

In summary, the UPoE consists of a number ($J < D$) of parameterized experts $\mathcal{T}(\mathbf{w}_j^T\mathbf{x}|\alpha_j)$ to model one dimensional projections of the data. These marginal distributions are combined in a multiplicative fashion, while the remaining dimensions are modelled only through their first and second order statistics (which were standardized through pre-sphering). The free parameters of the model are the weight-matrix $\mathbf{W}$ and the parameters of the experts $\{\alpha_j\}$. In the following section we will show how they can be inferred from the data.

3.2 Parallel learning of components in UPoE

To learn the parameters of the UPoE model we use the log-likelihood as our objective:

$$\begin{aligned} L &= \frac{1}{N}\sum_{n=1}^{N} \log\ p(\mathbf{x}_n) = \langle \log\ p(\mathbf{x}) \rangle_{\tilde{p}} \\ &= \frac{1}{2}\log\left|\mathbf{W}\mathbf{W}^T\right| \quad + \left\langle \sum_{i=1}^{D-J} \log \mathcal{N}(\mathbf{v}_i^T\mathbf{x}_n|0,1) + \sum_{j=1}^{J} \log \mathcal{T}(\mathbf{w}_j^T\mathbf{x}_n|\alpha_j) \right\rangle_{\tilde{p}} \end{aligned} \tag{17}$$

[2]Note that $\mathbf{V}^T\mathbf{V}$ is a projection operator which has $J - D$ eigenvalues 1, and the rest zeros. Assume that $\mathbf{u}$ is an eigenvector of $\mathbf{V}^T\mathbf{V}$ with eigenvalue 1, then $\mathbf{V}\mathbf{u}$ is an eigenvector of $\mathbf{V}\mathbf{V}^T$ with eigenvalue 1. Hence, all eigenvalues of $\mathbf{V}\mathbf{V}^T$ are 1 which implies that it must be equal to $\mathbf{I}$.

where $\langle\cdot\rangle_{\tilde{p}}$ is an average with respect to the empirical data distribution $\tilde{p}$. To perform gradient ascent on this cost function, we need its derivatives with respect to the matrix $\mathbf{W}$ and the expert parameters $\{\alpha_j\}$. We first define[3] the "energy" $E(z_j)$ through the relation:

$$\mathcal{T}(z_j|\alpha_j) = \frac{1}{Z_j(\alpha_j)} e^{-E(z_j;\alpha_j)} \tag{18}$$

where Z_j is the normalizing constant for expert j, which depends on α_j but not on $\mathbf{w}_j$.

After some algebra one then finds that the gradients are given by:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{W}} &= \mathbf{W}^{\#T} - \left\langle E'(\mathbf{z})\, \mathbf{x}^T \right\rangle_{\tilde{p}} &(19)\\ \frac{\partial L}{\partial \alpha_j} &= -\frac{\partial \log Z_j(\alpha_j)}{\partial \alpha_j} - \left\langle \frac{\partial E(z_j;\alpha_j)}{\partial \alpha_j} \right\rangle_{\tilde{p}} &(20) \end{aligned}$$

where the pseudo-inverse is defined as: $\mathbf{W}^{\#} = \mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}$, and $E'(\cdot)$ denotes the derivative of E with respect to z_j. In the derivation we have used the fact that the quadratic noise term averaged over the (sphered) data is independent of $\mathbf{W}$,

$$\left\langle \mathbf{x}^T P^{\perp} \mathbf{x} \right\rangle_{\tilde{p}} = \mathbf{tr}\left(P^{\perp} \left\langle \mathbf{x}\mathbf{x}^T \right\rangle_{\tilde{p}} \right) = \mathbf{tr}\left(P^{\perp} \right) = D - J \tag{21}$$

with $\mathbf{tr}$ denoting the trace. We also used the following fact:

$$\frac{1}{2} \frac{\partial \log \left| \mathbf{W}\mathbf{W}^T \right|}{\partial \mathbf{W}} = \mathbf{W}^{\#T} \tag{22}$$

Learning now proceeds through the updates $\mathbf{W} \rightarrow \mathbf{W} + \eta \partial L / \partial \mathbf{W}$ and $\alpha_j \rightarrow \alpha_j + \gamma \partial L / \partial \alpha_j$ for appropriate step-sizes η and γ.

The learning rule for the matrix $\mathbf{W}$ in Equation 19 matches that defined for the mutual information minimization approach to under-complete ICA (Equation 10), and the learning rule for the causal density model (Equation 6). In those cases, the rule is an *approximate* gradient based on a mutual information or maximum likelihood objective, while here it is derived as an *exact* learning rule for maximum likelihood under the under-complete PoE model.

Assume for a moment that we are changing the j^{th} component, while all other components are kept fixed. It is important to observe that the first term in Eqn.19 depends on *all* components through the pseudo-inverse, while the second term only depends on the $\mathbf{w}_j$ parameters of the component that we are currently training. This implies that the first term represents what the model already knows about the data and therefore causes the components to diversify. Without this term all components would become the same. Another way of seeing this is by rewriting the first term into a form very similar to the second term:

$$\begin{aligned} \mathbf{W}^{\#T} &= \frac{1}{2} \frac{\partial \log \left| \mathbf{W}\mathbf{W}^T \right|}{\partial \mathbf{W}} \\ &= -\frac{\partial}{\partial \mathbf{W}} \log \int d\mathbf{x} \prod_{i=1}^{D-J} \mathcal{N}(\mathbf{v}_i^T \mathbf{x}|0,1) \prod_{j=1}^{J} \mathcal{T}(\mathbf{w}_j^T \mathbf{x}|\alpha_j) \\ &= \left\langle E'(\mathbf{z})\, \mathbf{x}^T \right\rangle_p &(23) \end{aligned}$$

[3]We have found it convenient here to define the energy as the z-dependent part of the log-probability. However, log-probabilities including the log-partition function are more often found in the ICA literature.

where $\langle\cdot\rangle_p$ means an average with respect to the model distribution p (Eqn.16). The second lines follows from the fact that Eqn.16 is normalized while in the third line we used that $\partial/\partial\mathbf{W}\left\langle\mathbf{x}^T P^{\perp}\mathbf{x}\right\rangle_p = 0$ because the noise model has unit variance. One could choose to compute the above average through sampling from the model p (see Appendix A). Since Eqn.23 has precisely the same form as the second term in Eqn.19 muliplied by a minus sign, these samples take on the opposite role as the data. Their role is to shield off or "nullify" the data that are well represented by the model, causing learning to focus on poorly represented data. In fact, learning only stops when the average of $E'(\mathbf{z})\mathbf{x}^T$ over the empirical data distribution and the model distribution (represented by samples) match.

3.3 Sequential learning of components in UPoE

Learning the components can occur either in parallel or sequentially. In practise, the $\mathcal{O}(J^2D)$ computation of the pseudo-inverse $\mathbf{W}^{\#}$ in Eqn.19 *for every gradient update* is the computational bottleneck of the learning algorithm[4]. In standard, square ICA this problem was elegantly solved by defining a *natural gradient* [Amari, 1998] which converted the $\mathbf{W}^{-1}$ into $\mathbf{W}$. However, in the under-complete case, the natural gradient still depends on the pseudo-inverse, therefore not resulting in improved efficiency per iteration[5]. What is needed is a way to avoid the recomputation of $\mathbf{W}^{\#}$ for every step of gradient ascent learning.

We propose a more efficient sequential learning algorithm, which draws inspiration from projection pursuit density estimation (PPDE) [Friedman et al., 1984, Huber, 1985]. In PPDE the learning procedure is split into two parts: in one phase we search for a direction $\hat{\mathbf{w}}$ in which the projected data look non-normally distributed, i.e. we look for "interesting" directions in the data. This is usually implemented by defining a projection index and minimizing it. In the other phase, we fit a model to the marginal distribution in that direction and use it to replace the current estimate of that marginal distribution. In fact, one can show [Huber, 1985, Ripley, 1996] that for a given direction $\hat{\mathbf{w}}_j$, the optimal multiplicative update for the model at round $j-1$ is given by,

$$p_j(\mathbf{x}) = p_{j-1}(\mathbf{x})\,\frac{p^{\hat{\mathbf{w}}_j}_{\mathbf{data}}(\hat{\mathbf{w}}_j^T\mathbf{x})}{p^{\hat{\mathbf{w}}_j}_{j-1}(\hat{\mathbf{w}}_j^T\mathbf{x})} \tag{24}$$

where $p^{\hat{\mathbf{w}}_j}_{\mathbf{data}}$ is the marginal data distribution in the direction $\hat{\mathbf{w}}_j$ and $p^{\hat{\mathbf{w}}_j}_{j-1}$ is the marginal model distribution at round $j-1$ in direction $\hat{\mathbf{w}}_j$. The new model distribution $p_j(\mathbf{x})$ is still normalized after this update. This procedure is initiated with a "prior" noise model $p_0(\mathbf{x})$ which is typically a multivariate standard normal distribution.

It is not difficult to compute the change in the Kullback-Leibler distance between the data and the model distribution due to this update [Huber, 1985],

$$\mathcal{Q} = \mathcal{D}\left(p_{\mathbf{data}}||p_j\right) - \mathcal{D}\left(p_{\mathbf{data}}||p_{j-1}\right) = -\mathcal{D}\left(p^{\hat{\mathbf{w}}_j}_{\mathbf{data}}||p^{\hat{\mathbf{w}}_j}_{j-1}\right)$$

PPDE *minimizes* this projection index $\mathcal{Q}$ over directions $\hat{\mathbf{w}}_j$. The algorithm thus searches for directions for which the improvement of the model is largest. These are the directions where the model is most different from the data distribution. Although theoretically appealing, the computational load of this algorithm is large. This is due to the fact that the marginal distributions are typically modelled by splines or histograms, which makes the computation of the KL distance $\mathcal{D}\left(p^{\hat{\mathbf{w}}_j}_{\mathbf{data}}||p^{\hat{\mathbf{w}}_j}_{j-1}\right)$ and its derivatives (needed for gradient descent) cumbersome.

[4]Note that the second term of Eqn.19 scales as $\mathcal{O}(JDN)$. The linear dependence on the number of data points is however easily circumvented using batch methods.

[5]Because we work with sphered data the covariant form of the updates will also not lead to faster convergence.

We now describe a procedure, based on the above ideas, that trains the UPoE model sequentially. Due to the parametric form of the experts, this learning algorithm will turn out to be very efficient albeit less flexible than the plain vanilla PPDE procedure. First, we observe that if we choose the components orthogonal to each other and of length 1, i.e. we absorb the length of $\mathbf{w}_j$ into the parameters of the expert α_j, then $|\mathbf{W}\mathbf{W}^T| = 1$ in Eqn.16. To exploit this simplification, we will assume that all experts are chosen orthogonal in the following. However, since the data was preprocessed to have unit variance and no covariance (i.e. the data is decorrelated) this condition turns out *not* to be very restrictive. To see that, recall that in $(\mathbf{y}, \mathbf{z})$-coordinates all components are independent (Eqn.13). To ensure that the UPoE is *decorrelated* in $\mathbf{x}$ coordinates, the components must be orthogonal: $\mathbf{W}\mathbf{W}^T \propto \mathbf{I}$ which is precisely what we assume[6].

Now consider adding a new expert to the UPoE model, or more accurately, replacing a Gaussian noise direction $\mathcal{N}_j$ with a new parameterized expert $\mathcal{T}_j$. The change in the UPoE model can thus be written as,

$$p_j(\mathbf{x}) = p_{j-1}(\mathbf{x}) \frac{\mathcal{T}_j(\hat{\mathbf{w}}_j^T \mathbf{x}; \alpha_j)}{\mathcal{N}_j(\hat{\mathbf{w}}_j^T \mathbf{x})} \tag{25}$$

The fact that the model $p_{j-1}(\mathbf{x})$ is Gaussian in any direction orthogonal to the component directions $\{\mathbf{w}_1, ..., \mathbf{w}_{j-1}\}$ guarantees that the new model p_j is again normalized. Note that Eqn.25 is precisely in the form of Eqn.24 if the marginal data distribution in the direction $\hat{\mathbf{w}}_j$ is perfectly described by the expert $\mathcal{T}_j$.

To compute the projection index we determine the change in KL divergence between the data distribution and the model distribution, assuming update Eqn.25,

$$\mathcal{Q} = \mathcal{D}\left(p_{\mathbf{data}}||p_j\right) - \mathcal{D}\left(p_{\mathbf{data}}||p_{j-1}\right) = \mathcal{D}\left(p^{\hat{\mathbf{w}}_j}_{\mathbf{data}}||\mathcal{T}_j\right) - \mathcal{D}\left(p^{\hat{\mathbf{w}}_j}_{\mathbf{data}}||\mathcal{N}_j\right) \tag{26}$$

In contrast to the PPDE objective in Eqn.25 this projection index has *two* terms. The first term searches for directions in which the data can be well described by the expert while the second term prefers directions that are "interesting", i.e. non-normally distributed. Compared to Eqn.25, the first term is new and appears because we didn't assume that the data distribution in the direction $\hat{\mathbf{w}}_j$ can be modelled with infinite precision. However, the most important difference between the two projection indices is that the latter is far more tractable than the one used for PPDE. This can be seen by inserting the empirical data distribution $\tilde{p}$ for $p_{\mathbf{data}}$ and rewriting Eqn.26 as,

$$\mathcal{Q} = \left\langle E_j(\hat{\mathbf{w}}_j^T \mathbf{x}; \alpha_j) - \frac{1}{2}(\hat{\mathbf{w}}_j^T \mathbf{x})^2 \right\rangle_{\tilde{p}} + \log Z_j(\alpha_j) - \frac{1}{2}\log(2\pi)$$

which is trivially evaluated. It is also easy to compute the derivatives which are needed for learning,

$$\frac{\partial \mathcal{Q}}{\partial \hat{\mathbf{w}}_j} = \left\langle \left(E'_j(\hat{\mathbf{w}}_j^T \mathbf{x}; \alpha_j) - \hat{\mathbf{w}}_j^T \mathbf{x}\right) \mathbf{x}^T \right\rangle_{\tilde{p}} \tag{27}$$

There are two reasons for the simplification. Firstly, the entropy of the marginal data distribution $p^{\hat{\mathbf{w}}}$ drops out of the projection index. Secondly, the marginal distribution of the model in the new direction, $p^{\hat{\mathbf{w}}_j}_{j-1}$ does not need to be estimated because it is exactly Gaussian in all directions orthogonal to the the subspace spanned by the directions $\{\hat{\mathbf{w}}_1, ..., \hat{\mathbf{w}}_{j-1}\}$. To maintain orthogonality between the components we need to re-orthogonalize $\hat{\mathbf{w}}_j$ with respect to $\{\hat{\mathbf{w}}_1, ..., \hat{\mathbf{w}}_{j-1}\}$ after every gradient update,

$$\mathbf{w}_j \rightarrow \mathbf{w}_j - \mathbf{W}_{j-1}^T \mathbf{W}_{j-1} \mathbf{w}_j \qquad \mathbf{w}_j \rightarrow \mathbf{w}_j / \parallel \mathbf{w}_j \parallel \tag{28}$$

where $\mathbf{W}_{j-1}$ is the matrix with the component directions $\{\mathbf{w}_1, ..., \mathbf{w}_{j-1}\}$ as its rows.

[6]Of course, the maximum likelihood solution may sustain a small error in the covariance estimate in order to model other features of the density better. However, even in that case we predict that the matrix $\mathbf{W}$ should be close to orthogonal.

In figure 2 we plot some examples of projection indices for Student-t experts with various settings of the parameters $\alpha = (\theta, \beta)$ (inverse scale and inverse temperature). The Student-t density is a symmetric distribution with a strong peak in the middle and heavy tails (see appendix B for details). Projections of the data that have excess concentrations of data points in the center and in the tails will act as attractors in minimizing the projection index. Conversely, Gaussian projections, with excess concentrations of data points in the intermediate regimes, will act as repellers in minimizing the projection index.

The expert parameters α_j can be learned *simultaneously* with the directions $\hat{\mathbf{w}}_j$, by minimizing $\mathcal{Q}$. The gradients are given by,

$$\frac{\partial \mathcal{Q}}{\partial \alpha_j} = \left\langle \frac{\partial E_j(\hat{\mathbf{w}}_j^T \mathbf{x}; \alpha_j)}{\partial \alpha_j} \right\rangle_{\tilde{p}} + \frac{\partial \log Z_j(\alpha_j)}{\partial \alpha_j} \tag{29}$$

which is precisely equivalent to Eqn.20. This flexibility to adapt the expert at the same time as we learn the direction $\hat{\mathbf{w}}_j$ may become important if different directions in the data have qualitatively different marginals, e.g. some directions could be highly kurtotic while others could be bimodal.

Since $\mathcal{Q}$ represents the change in the negative log-likelihood (if we replace the data distribution with the empirical distribution) we should stop learning when all remaining directions satisfy $\mathcal{Q} \geq 0$.

Finally, we summarize the resulting algorithm below:

Sequential Learning Algorithm for UPoEs

Repeat for j=1 to J or until $\mathcal{Q} \geq 0$:

1. Initialize $\hat{\mathbf{w}}_j$ to a random unit length D-vector, orthogonal to the previous directions $\{\hat{\mathbf{w}}_1, ..., \hat{\mathbf{w}}_{j-1}\}$.
2. *Repeat until convergence:*
   2i. Take a gradient step to decrease projection index $\mathcal{Q}$ over directions $\hat{\mathbf{w}}_j$ (Eqn.27) and parameters α_j (Eqn.29).
   2ii. Apply Gram-Schmidt orthonormalization to $\hat{\mathbf{w}}_j$ (Eqn.28).
3. Update model (Eqn.25).

---

4 Relation to Other Models

In this section we will relate the UPoE to some other models, including the UICA models that were reviewed in section 2, projection pursuit (PP) and additive random field models.

4.1 Under-complete ICA

First, let's return to the UICA models discussed in section 2. The causal generative model for under-complete ICA defined in Section 2.2 is different from the under-complete PoE model since it is defined in terms of stochastic hidden source variables $\{s_i\}$ which are difficult to integrate out (Eqn.3). Learning the causal generative model requires computing averages over the posterior distributions $p(\mathbf{s}_n|\mathbf{x}_n)$, which is intractable. In contrast, the UPoE model circumvents this problem by avoiding stochastic hidden units altogether.

The information-minimization approach to under-complete ICA defined in Section 2.3 is also different from the UPoE model, in that it does not define a probabilistic density model for the data. Instead of maximizing maximum likelihood for a density model, it minimizes mutual information between components. Similarly to the UPoE, the minimal mutual information method may be thought of as a bottom-up approach. They both consider a cost function in terms of the components $\mathbf{z} = \mathbf{W}\mathbf{x}$. In contrast, the causal generative approach has a top down philosophy: $\mathbf{x} = \mathbf{A}\mathbf{s}$.

Despite the differences, all of these models are in fact closely related, as is evidenced by the fact that the exact learning rule for the UPoE has been used as an approximation to the intractable gradient ascent learning rules for the earlier UICA models [Stone and Porrill, 1998, Lu and Rajapakse, 2000, Ridder et al., 2002].

In the complete noiseless case, i.e. when $J = D$ and $p(\mathbf{x}|\mathbf{s}) = \delta(\mathbf{x} - \mathbf{W}^{-1}\mathbf{s})$, the PoE and square ICA models become in fact equivalent. This implies that the presented sequential learning algorithm may also be considered as a sequential learning algorithm for square, noiseless ICA.

The formulation of ICA that is closest in spirit to the sequential learning method presented in this paper is the one underlying the FastICA algorithm [Hyvarinen, 1999] (see Section 2.4). There, new components are added sequentially by minimizing the negentropy as a projection index. The main difference with the sequential procedure described here is that our (different) projection index is based on the maximal decrease in log-likelihood of a probabilistic model.

4.2 Projection pursuit

In section 3.3 we described the forward or synthetic PPDE procedure. The resulting model is in fact very similar to a UPoE model, with the subtle difference that the background model in PPDE is a full dimensional standard normal while the background model for a UPoE is normal only in the orthogonal complement of the space spanned by the components. The PPDE procedure is more flexible than the sequential learning algorithm for UPoEs because the marginals are fit with histograms or splines and the components are not necessarily orthogonal. The price to be paid is its computational inefficiency.

There is also a backward or analytic approach to PPDE [Friedman, 1987]. The idea is that one starts with the data distribution and sequentially strips away its non-Gaussian structure. A practical method to inform the algorithm about the structure that has already been detected in previous rounds is to "Gaussianize" those directions. This amounts to transforming the old data set into a new one where previously detected directions are now normally distributed. This technique only works when the directions are mutually orthogonal. In [Zhu, 2002] it is argued that the resulting density model is in fact very cumbersome, due to the need to "unwrap" these transformations.

4.3 Additive random field models

In the discrete domain product models are known under various names; additive random field models [Pietra et al., 1997], log-linear models [Darroch and Ratcliff, 1972] and maximum entropy models [Zhu et al., 1997]. These models have an elegant dual interpretation as the distribution that minimizes the KL divergence $\mathcal{D}(p||p^0)$ with a "prior model" p^0, subject to a number of constraints $\langle \Phi_i(\mathbf{x}) \rangle_{p_{\mathbf{model}}} = \langle \Phi_i(\mathbf{x}) \rangle_{p_{\mathbf{data}}}$. These constraints are enforced using Lagrange multipliers $\{\lambda_i\}$ and combined with $\mathcal{D}(p||p^0)$ to form a Lagrangian,

$$\mathcal{L} = \mathcal{D}(p||p^0) + \sum_i \lambda_i \left(\langle \Phi_i(\mathbf{x}) \rangle_{p_{\mathbf{model}}} - \langle \Phi_i(\mathbf{x}) \rangle_{p_{\mathbf{data}}} \right) \tag{30}$$

Setting to zero the functional derivatives of $\mathcal{L}$ with respect to the model distribution $p(\mathbf{x})$, we arrive at the following solution,

$$p(\mathbf{x}) \propto p^0(\mathbf{x}) e^{\sum_{i=1}^{J} \lambda_i \Phi_i(\mathbf{x})} \tag{31}$$

The features $\Phi_i(\mathbf{x})$ are selected from a large "library" while the Lagrange multipliers λ_i (or weights), which multiply the features are learned. This procedure is typically sequential, minimizing a similar objective as the projection index proposed in this paper, $\mathcal{D}\left(p_{\mathbf{data}}||p_j\right) - \mathcal{D}\left(p_{\mathbf{data}}||p_{j-1}\right)$. Identifying features with $\Phi(y_i, \mathbf{x}) = \delta(y_i - \hat{\mathbf{w}}_i^T \mathbf{x})$ we can show that the optimal choice for the weights is given by $\lambda(y_i) = \log \mathcal{T}(y_i; \alpha_i) - \log \mathcal{N}(y_i)$, which precisely corresponds to the UPoE model.

The discrete search over features is therefore replaced in our case with a continuous optimization over directions $\hat{\mathbf{w}}$ while the estimation of the weights λ could be identified with the optimization of expert parameters α. Thus, in many respects the UPoE model and its sequential learning algorithm are the analogue in the continuous domain of the additive random field model and its feature pursuit algorithm.

5 Experiments

In this section we describe three experiments that test the ability of the proposed algorithms to find meaningful components in the data. The first experiment was designed to compare the parallel and the sequential learning procedures. The second experiment is a standard test for projection pursuit algorithms to find multi-modal directions in the data that can be used for visualization or to facilitate classification. And finally, we apply the sequential algorithm to a large image data-set in order to show its efficiency in extracting components in high dimensional spaces.

5.1 Parallel vs. sequential learning

To compare the parallel learning algorithm of section 3.2 with the sequential algorithm described in section 3.3 we trained models with varying numbers of components. The data-set[7] consisted of 1965 face images of 128 pixels each. This data-set was centered, sphered and reduced to 50 dimensions using PCA, keeping only the high variance directions. The data cases were split in 1000 randomly chosen training cases and 965 test cases. We used Student-t experts (see appendix B) to describe the marginal distributions which can gracefully interpolate between a normal distribution and a super-Gaussian distribution (highly peaked distribution with heavy tails).

We used three different training procedures for this experiment. Results labelled with "PAR" indicate that the parallel training procedure was used (Eqn.19 & 20). Each time a component is added to the model, the other components are initialized at the ones previously learned, but are allowed to change during learning. Results labelled with "PP" indicate that the sequential, projection pursuit inspired training procedure from section 3.3 was used. We anticipate (at least) two reasons why the PP procedure could produce inferior results to the parallel procedure: a) the orthogonality constraint of the components is too restrictive and b) the fact that we are not "back-fitting" the parameters of the previous components (i.e. we keep the directions and expert parameters of the previous experts fixed) gives learning less flexibility. To separate these two sources of diminished performance, we use a third sequential learning method that does not constrain the components to be orthogonal, but does *not* back-fit the parameters of the earlier found components. This procedure, labelled with "SEQ", uses gradient descent with the derivative Eqn.19, but only updates one component at a time, keeping all the earlier ones fixed[8].

[7] Obtained from *www.cs.toronto.edu/~roweis/data*

[8] Note that we do not advertise this method as a practical learning algorithm because it is as inefficient as the parallel learning rule.

In figure 3 we show the log-likelihood for these three different training procedures. Curves labelled with "TRN" indicate results for training data while curves labelled with "TST" indicate results for test data. Although the parallel procedure outperforms both sequential methods on training data, there is no significant difference on the test data. We also infer from this that the orthogonality constraint (constraining the covariance of the model to be identical to that of the data, i.e. $C_p = C_{\tilde{p}} = \mathbf{I}$) does seem to have a small effect on the training data but not on the test data.

5.2 Leptograpsus crabs

The Leptograpsus crab data set[9] contains 5 morphological measurements of 200 crabs. There are 4 distinct classes in the data set: male and female crabs each in two different color forms. This data set was used in [Ripley, 1996] to test the ability of projection pursuit algorithms to find multi-modal projections for the purpose of visualization or classification.

The Crabs data were first centered and sphered before presentation to the sequential learning algorithm. The experts were parameterized as a mixture of 2 Student-t distributions (see appendix B) with fixed settings of the inverse temperature at $\beta_{1,2} = 20$ and means at $\mu_1 = -1, \mu_2 = +1$. Figure 4 and 5 show that the algorithm is indeed able to extract multi-modal projections of the data. It was found however that there were many local minima present and each time the results looked slightly different.

5.3 Independent components of natural images

To show that the sequential learning algorithm is able to learn efficiently from large amounts of high dimensional data we collected $100,000$ patches of natural images[10] of size 30×30 pixels. This data set was centered, sphered and reduced to 400 dimensions using PCA. In figure 6 we plotted 25 out of 100 components $\hat{\mathbf{w}}$ trained on this data set. Training was done in batches of 100 cases and involved an adaptive step-size. Learning took a few hours on a 1-Gz PC. The results are qualitatively similar to the Gabor-like components found using ICA in [Bell and Sejnowski, 1997] providing evidence that the algorithm has converged to a sensible solution.

6 Conclusion

The UPoE model and its learning algorithms provide a link between under-complete ICA, projection pursuit and additive random field models. The parallel learning rules have been proposed in the literature as approximate learning rules for under-complete ICA. This paper provides insight into what those learning rules really accomplish. The sequential learning rules can be interpreted as a a parametric variant of PPDE, but are also similar in spirit to the FastICA fitting method. Finally, the UPoE and its sequential learning rules may be interpreted as the continuous analogue of additive random field models and their feature induction techniques.

Apart from shedding light on and unifying previous approaches to UICA, the UPoE provides fast learning in a fully probabilistic setting. We obtain many advantages from the fully probabilistic approach: it allows one to reason about queries in a statistically sound manner, one can sample new data points, one can get a

[9]Obtained from *www.stats.ox.ac.uk/pub/PRNN/*
[10]Obtained from www.cis.hut.fi/projects/ica/data/images

handle on how many significant sources were present that generated the data, it can be used to compress the data or restore the data if part of it got lost, and many more.

Important features of the UPoE are its tractability and its efficient learning rules. The most important disadvantage is that the they are limited to learning under-complete, or square models. In some settings over-complete models are preferred, particularly when the fidelity of the representation outweighs its efficiency. Over-complete PoE models [Teh et al., 2003] can be applied here, but these are much harder to learn than UPoEs.

Another limitation is its restriction to the continuous domain. Many applications, such as document retrieval and language processing, require models to work in the (positive) discrete data domains. Many existing models, such as the aspect model, suffer from intractable inference which is needed for learning. Extending PoE models into this domain is a topic of future research.

In [Welling et al., 2002] a product model was described that has the ability to topographically order its components. This idea readily extends to the UPoE model. Unfortunately, learning has to proceed using approximate methods such as contrastive divergence. Extending these ideas to the discrete domain may have interesting applications in latent semantic analysis where the topics can be ordered topographically according to their interdependencies.

"Extreme components analysis" or XCA [Welling et al., 2003] is a special case of a UPoE where all experts are Gaussian. In that case the *only* relevant statistics are second order in contrast to the UPoE model described in this paper where *only* higher order statistics are modelled (i.e. the second order statistics are factored out). The XCA model is a probabilistic model that can be understood as modelling the main directions of variation in the data by its principal components and in addition modelling the soft constraints in the data using its minor components (directions of low variance). The optimal number of principal and minor components is determined automatically using a single eigenvalue decomposition of the sample covariance matrix.

On the algorithmic level we think there is still some progress to be made. For instance, to minimize the projection index we have currently implemented a very simple adaptive gradient descent procedure. Clearly, second order methods like Newton steps are desired here to further improve efficiency. Although we have not experienced any problems with the Gram-Schmidt procedure described in this paper (Eqn.28) more stable methods to orthogonalize vectors may become valuable in future applications.

There are many ways in which the UPoE model can be extended. One possibility is to consider mixtures of UPoEs, UPoEs through time and so on. But possibly the most interesting and relevant problem to be dealt with is to find an automatic way to determine the number of components from the data. Bayesian methods suggest itself for this task, but the need to compute the posterior distribution $p(\mathbf{W}|\mathbf{x})$ and average over it will render this problem intractable. A number of approximate methods such as "variational Bayesian" inference or MCMC sampling are applicable, but it is currently unknown to us how effective they will be.

Acknowledgements

We would like to thank Yee Whye Teh, Simon Osindero, Sam Roweis, Matthias Seeger and Miguel Carreira-Perpinan for discussions and suggestions.

A Sampling

Sampling from the UPoE is relatively straightforward. We sample $z_j \sim \mathcal{T}(z_j|\alpha_j)$ and $y_i \sim \mathcal{N}(y_i|0,1)$ and combine them into a sample in $\mathbf{x}$-space using,

$$\mathbf{x} = \mathcal{P}\mathbf{x} + \mathcal{P}^{\perp}\mathbf{x} = \mathbf{W}^{\#}\mathbf{z} + \mathbf{V}^T\mathbf{y} \tag{32}$$

To explicitly compute an orthonormal basis $\mathbf{V}^T$ we can compute the following SVD decomposition $\mathbf{ABC}^T = SVD([\mathbf{W}^T|\mathbf{0}])$. The last $J-D$ columns of $\mathbf{A}$ then form and orthonormal basis in the complement subspace, $\mathbf{A} = [\mathbf{A}_{DJ}|\mathbf{V}^T]$. Alternatively, one can sample $\mathbf{x}' \sim \mathcal{N}(\mathbf{x}'|0,\mathbf{I})$ and subsequently project the samples on the orthogonal subspace: $\mathbf{V}^T\mathbf{y} = \mathcal{P}^{\perp}\mathbf{x}'$. The pseudo-inverse of $\mathbf{W}$ can be computed as $\mathbf{W}^{\#} = A_{DJ}\mathbf{B}_{JJ}^{-1}\mathbf{C}_{JJ}^T$ or simply as $\mathbf{W}^{\#} = \mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}$.

B Student-T Experts

The probability distribution of a (generalized) Student-t distribution is given by,

$$\mathcal{T}(z) = \frac{\Gamma(\beta)\,\theta}{\Gamma(\beta-\frac{1}{2})\sqrt{2\pi}}\left(1+\frac{1}{2}\left(\theta(z-\mu)\right)^2\right)^{-\beta} \tag{33}$$

where μ is its mean, $\theta > 0$ is an inverse scale parameter and $\beta > \frac{1}{2}$ an inverse "temperature" which controls the sharpness of the distribution. We can easily sample from it for arbitrary β and θ: first compute $a = \beta - \frac{1}{2}$ and $b = \theta^2$. Next, sample precision parameters from a gamma distribution, $y \sim c\,y^{a-1}e^{-y/b}$. Finally, sample z from a normal distribution with that precision $z \sim c\,e^{-\frac{1}{2}y\,z^2}$. The derivatives of the log-likelihood for the parameters μ, θ, β are given by,

$$\frac{\partial L}{\partial \theta} = \frac{1}{\theta} - \left\langle \frac{\beta\theta(z-\mu)^2}{1+\frac{1}{2}\left(\theta(z-\mu)\right)^2}\right\rangle_{\tilde{p}} \tag{34}$$

$$\frac{\partial L}{\partial \beta} = \Psi(\beta) - \Psi(\beta-\frac{1}{2}) - \left\langle \log\left(1+\frac{1}{2}\left(\theta(z-\mu)\right)^2\right)\right\rangle_{\tilde{p}} \tag{35}$$

$$\frac{\partial L}{\partial \mu} = \left\langle \frac{-\beta\theta^2(z-\mu)}{1+\frac{1}{2}\left(\theta(z-\mu)\right)^2}\right\rangle_{\tilde{p}} \tag{36}$$

where $\Psi(x) = \partial/\partial x \ln\Gamma(x)$ is the "digamma" function. It is not hard to compute the variance and kurtosis of a central Student-T distribution,

$$\left\langle (z-\mu)^2\right\rangle_{\mathcal{T}} = \frac{1}{\theta^2\,(\beta-\frac{3}{2})} \qquad \beta > \frac{3}{2} \tag{37}$$

$$\frac{\left\langle (z-\mu)^4\right\rangle_{\mathcal{T}}}{\left\langle (z-\mu)^2\right\rangle_{\mathcal{T}}} - 3 = \frac{3}{(\beta-\frac{5}{2})} \qquad \beta > \frac{5}{2} \tag{38}$$

Thus, small values for β represent peaked distributions with large kurtosis.

Using centralized Student-t distributions, the derivative of the projection index Eqn.27 with respect to the component $\hat{\mathbf{w}}_j$ is now given by,

$$\frac{\partial \mathcal{Q}}{\partial \hat{\mathbf{w}}_j} = \left\langle \left(\frac{\beta_j\theta_j^2\hat{\mathbf{w}}_j^T\mathbf{x}}{1+\frac{1}{2}\left(\theta_j\hat{\mathbf{w}}_j^T\mathbf{x}\right)^2} - \hat{\mathbf{w}}_j^T\mathbf{x}\right)\;\mathbf{x}^T\right\rangle_{\tilde{p}} \tag{39}$$

A more general family of experts is given by *mixtures* of Student-t distributions.

$$\mathcal{T}(\mathbf{x}) = \sum_a \pi_a \mathcal{T}_a(\mathbf{x};\alpha_a) \tag{40}$$

where π_a are the mixture coefficients. Learning mixture models is straightforward using the EM algorithm,

$$\begin{aligned}
r_{an} &= \frac{\pi_a \ \mathcal{T}_a(z_n;\alpha_a)}{\sum_a \pi_a \ \mathcal{T}_a(z_n;\alpha_a)} && (41)\\
\pi_a^{\mathbf{new}} &= \frac{1}{N}\sum_n r_{an} && (42)\\
\alpha_a^{\mathbf{new}} &= \alpha_a + \frac{\varepsilon}{N}\sum_n r_{an}\frac{\partial \log \mathcal{T}_a(z_n;\alpha_a)}{\partial \alpha_a} && (43)\\
\hat{\mathbf{w}}^{\mathbf{new}} &= \hat{\mathbf{w}} + \frac{\varepsilon}{N}\sum_a\sum_n r_{an}\frac{\partial \log \mathcal{T}_a(z_n;\alpha_a)}{\partial \hat{\mathbf{w}}_a} && (44)
\end{aligned}$$

where the projections also need to be orthonormalized at every step. For μ and θ there are faster IRLS updates available [Titterington et al., 1985],

$$\begin{aligned}
w_{an} &= \frac{r_{an}}{\left(1+\frac{1}{2}\left(\theta_a(z_n-\mu_a)\right)^2\right)} && (45)\\
\mu_a^{\mathbf{new}} &= \frac{\sum_n w_{an} z_n}{\sum_n w_{an}} && (46)\\
\left(\theta^2\right)_a^{\mathbf{new}} &= \frac{N\pi_a^{\mathbf{new}}}{\sum_n \alpha_a w_{an}(z_n-\mu_a)^2} && (47)
\end{aligned}$$

The new weights w_{an} downweight outliers which makes this a robust alternative to the mixture of Gaussians model [Titterington et al., 1985].

References

[Amari, 1998] Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276.

[Bell and Sejnowski, 1995] Bell, A. and Sejnowski, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159.

[Bell and Sejnowski, 1997] Bell, A. and Sejnowski, T. (1997). The independent components of natural scenes are edge filters. *Vision Research*, 37:3327–3338.

[Comon, 1994] Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, 36:287–314.

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

[Friedman, 1987] Friedman, J. (1987). Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266.

[Friedman et al., 1984] Friedman, J., Stuetzle, W., and Schroeder, A. (1984). Projection pursuit density estimation. *Journal of the American Statistical Association*, 79:599–608.

[Hinton, 1999] Hinton, G. (1999). Products of experts. In *Proceedings of the International Conference on Artificial Neural Networks*, volume 1, pages 1–6.

[Hinton and Teh, 2001] Hinton, G. and Teh, Y. (2001). Discovering multiple constraints that are frequently approximately satisfied. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 227–234.

[Hinton et al., 2001] Hinton, G., Welling, M., Teh, Y., and Osindero, K. (2001). A new view of ICA. In *Int. Conf. on Independent Component Analysis and Blind Source Separation*.

[Huber, 1985] Huber, P. (1985). Projection pursuit. *Annals of Statistics*, 13(2):435–475.

[Hyvärinen, 1998] Hyvärinen, A. (1998). New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing Systems*, volume 10, pages 273–279.

[Hyvarinen, 1999] Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634.

[Kschischang et al., 2001] Kschischang, F., Frey, B., and Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

[Lewicki and Sejnowski, 1998] Lewicki, M. and Sejnowski, T. (1998). Learning overcomplete representations. Technical report, Howard Hughes Medical Institute, Computational Neurobiology Lab, The Salk Institute. Accepted for publication in Neural Computation.

[Lu and Rajapakse, 2000] Lu, W. and Rajapakse, J. (2000). A neural network for undercomplete independent component analysis. In *8th European Symposium on Artificial Neural Networks*, Bruges, Belgium.

[Olshausen and Field, 1997] Olshausen, A. and Field, D. (1997). Sparse coding with over-complete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325.

[Pearlmutter and Parra, 1996] Pearlmutter, B. and Parra, L. (1996). A context sensitive generalization of ICA. *Proceedings of the International Conference on Neural Information Processing*, pages 151–157.

[Pietra et al., 1997] Pietra, S. D., Pietra, V. D., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

[Ridder et al., 2002] Ridder, D. D., Duin, D., and Kittler, R. (2002). Texture description by independent components. In *Proceedings of the joint IAPR International Workshops on Syntactical and Structural Pattern Recognition*, Windsor, Canada.

[Ripley, 1996] Ripley, B. (1996). *Pattern recognition and neural networks*. Cambridge University Press.

[Stone and Porrill, 1998] Stone, J. and Porrill, J. (1998). Undercomplete independent component analysis for signal separation and dimesnion reduction. Technical report, University of Sheffield, Departement of Psychology.

[Teh et al., 2003] Teh, Y., Welling, M., Osindero, S., and Hinton, G. (2003). Energy-based models for sparse overcomplete representations. Technical report, University of Toronto, Department of Computer Science. submitted.

[Titterington et al., 1985] Titterington, D., Smith, A., and Makov, U. (1985). *Statistical analysis of finite mixture distributions*. John & Wiley & Sons.

[Welling et al., 2003] Welling, M., Agakov, F., and Williams, C. (2003). Extreme components analysis. In *Advances in Neural Information Processing Systems*, volume 16, Vancouver, Canada.

[Welling et al., 2002] Welling, M., Hinton, G., and Osindero, S. (2002). Learning sparse topographic representations with products of student-t distributions. In *Advances in Neural Information Processing Systems*, volume 15, Vancouver, Canada.

[Zhu, 2002] Zhu, M. (2002). A note on projection pursuit. Technical report, University of Waterloo, Department of Statistics. Technical Report.

[Zhu et al., 1997] Zhu, S., Wu, Z., and Mumford, D. (1997). Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660.

Figure 1: Left: directed acyclic graph corresponding to the causal generative UICA model. Right: undirected graphical model with deterministic "hidden" units corresponding to the UPoE. This graph can also be viewed as a factor graph.

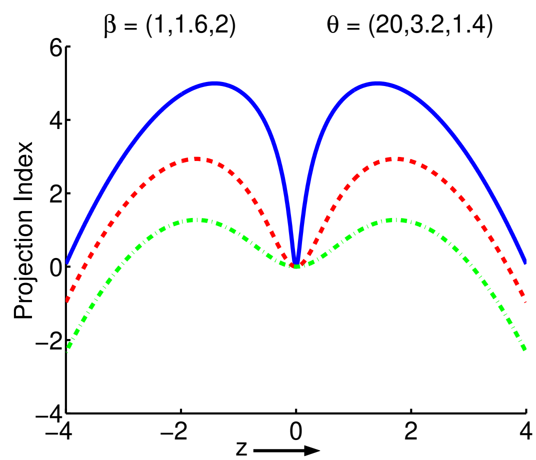

Figure 2: The projection index $f(z) = \beta \log(1 + \frac{1}{2}(\theta z)^2) - \frac{1}{2}z^2$ for the Student-t experts. Top (solid blue) line has values $\beta = 1, \theta = 20$, middle (red dashed) line has values $\beta = 1.6, \theta = 3.2$ and bottom (green dash-dotted) line has values $\beta = 2, \theta = 1.4$

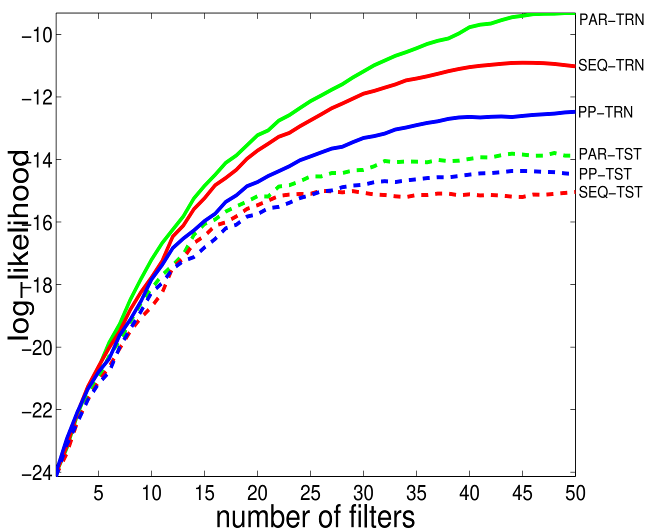

Figure 3: Models trained on the "Frey-faces" data set with varying numbers of components. Solid curves represent log-likelihood on training data while dashed lines represent log-likelihood on test data. "PAR" indicates that the components were trained in parallel using Eqn.19. "SEQ" indicates that the model was trained using Eqn.19 sequentially (see main text for details) and "PP" indicates that the model was trained using the algorithm described in section 3.3.

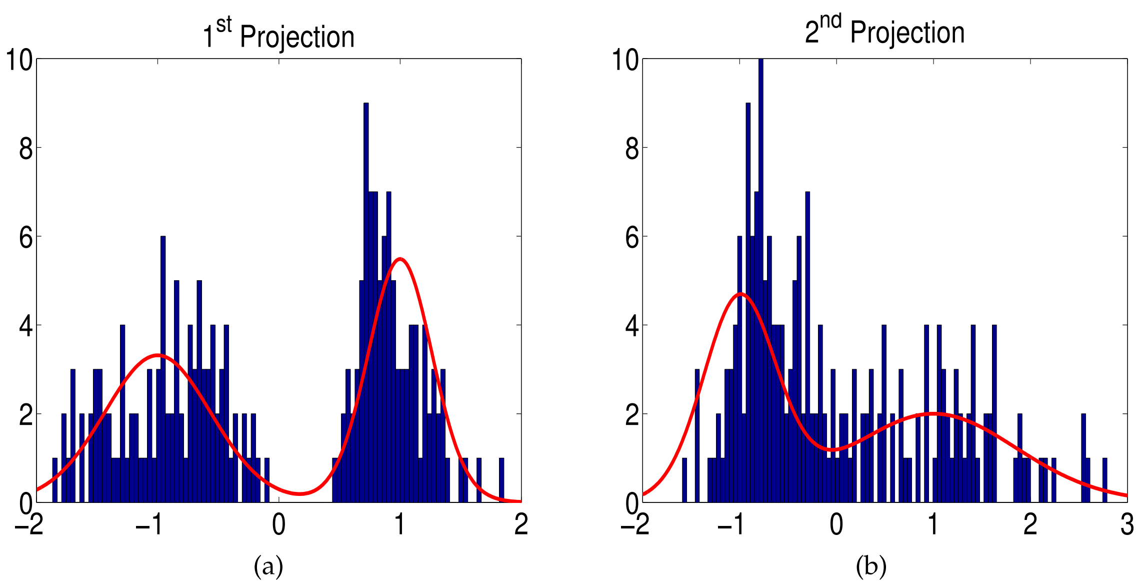

Figure 4: First projection (a) and second projection (b) of the Crabs data set found by the sequential learning algorithm. Overlayed are the fitted mixture of Student-t distributions with means fixated at $\mu_1 = -1$ and $\mu_2 = +1$ (solid red curve).

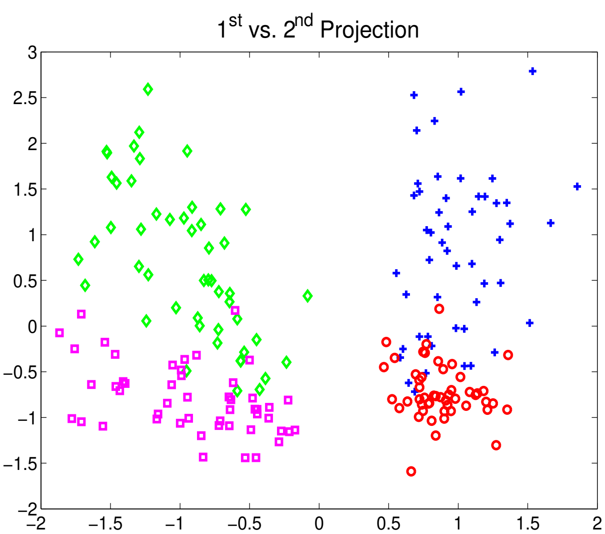

Figure 5: First versus second projection of the Crabs data set corresponding to the histograms shown in figure 4. The 4 different symbols ($\circ, \Box, \Diamond, +$) correspond to the 4 different classes of crab species and sex.

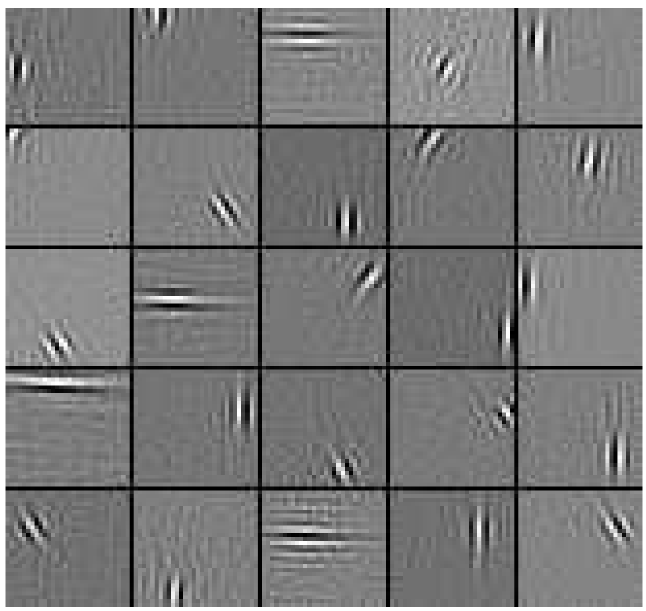

Figure 6: 25 randomly chosen components out of a 100 learned components (using the sequential algorithm) on the natural image data set.