The On-line Cross Entropy Method for Unsupervised Data Exploration

Ying Wu
University of the West of Scotland
School of Computing
Paisley, PA1 2BE, UK
ying.wu@uws.ac.uk

Colin Fyfe
University of the West of Scotland
School of Computing
Paisley, PA1 2BE, UK
colin.fyfe@uws.ac.uk

Abstract: We investigate the use of the new Cross Entropy method as a tool for exploratory data analysis. We show how this method can be used to perform linear projections such as principal component analysis, exploratory projection pursuit and canonical correlation analysis. We further go on to show how topology preserving mappings can be created usin the cross entropy method. We also show how the cross entropy method can be used to train deep architecture nets which are one of the main current research directions for creating true artificial intelligence. Finally we show how the cross entropy method can be used to optimize parameters for latent variable models.

Key–Words: Cross entropy, Linear projections, Topographic mapping.

1 Introduction

The cross entropy method has been well introduced as [4] and was motivated by an adaptive algorithm for estimating probabilities of *rare events* in complex stochastic networks [12]. For example, a Monte Carlo simulation which draws instances from the true distribution of events would require an inordinate number of draws before enough of the rare events were seen to make a reliable estimate of their probability of occurring. It was soon realized that the cross entropy method can also be applied to solving difficult combinatorial optimization problems with a simple modification of the method of [12]. Generally speaking, the basic mechanism involves an iterative procedure of two phases:

1. draw random data samples from the currently specified distribution.
2. identify those samples which are, in some way, "closest" to the rare event of interest and update the parameters of the currently specified distribution to make these samples more representative in the next iteration.

The application of Exploratory Data Analysis (EDA) has received a great deal of investigation recently. The primary goal of EDA is to optimize the analyst's insight into a data set and into the underlying structure of a data set. Some unsupervised exploratory data analysis techniques such as principal component analysis (PCA), independent component analysis (ICA) and canonical correlation analysis (CCA) have been widely investigated. Recently, we applied the cross entropy method to independent component analysis with batch learning. In this paper, we derive a general method by applying the cross entropy method to several unsupervised exploratory data analysis techniques with on-line cross entropy learning.

2 The Cross Entropy Method

The Cross Entropy method is best approached from the perspective of its use in estimates of statistics due to rare events such as the probability measure associated with the rare event. We discuss this first before going on to apply the method to the field of optimization.

2.1 The Cross Entropy Method for Rare Event Simulations

The cross entropy method generally uses *importance sampling* rather than simple Monte Carlo methods: if the original pdf of the data is $f(\mathbf{x})$, then we require to find a pdf, $g(\mathbf{x})$, such that all of $g()$'s probability mass is allocated in regions in which the samples are close to the rare-event. More formally, let $l = (S(\mathbf{x}) > \gamma)$ be the event in which we are interested. Then

$$l = \int I_{\{S(\mathbf{x})>\gamma\}} \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = E_{g()} \left[I_{\{S(\mathbf{X})>\gamma\}} \frac{f(\mathbf{X})}{g(\mathbf{X})} \right] \quad (1)$$

where I_T is the indicator function describing when T in fact occurred. An unbiased estimator of this is

$$\hat{l} = \frac{1}{N}\sum_{i=1}^{N} I_{\{S(\mathbf{X_i})>\gamma\}} W(\mathbf{X}_i) \tag{2}$$

where $W(\mathbf{x}) = \frac{f(\mathbf{x})}{g(\mathbf{x})}$ is the likelihood ratio and $\mathbf{X}_i$ are the samples drawn from $g()$. Then the simplest algorithm [4] depends on working within a family of pdfs whose parameters we update i.e. let $f(\mathbf{x}) = f(\mathbf{x}, \mathbf{u})$, $\mathbf{u}$ being a parameter of the family to which $f()$ belongs; then the basic algorithm is

1. Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set t=1.
2. Generate random samples, $\mathbf{X}_1, ..., \mathbf{X}_N$ from $f(\mathbf{x}, \mathbf{v}_{t-1})$.
3. Calculate $S(\mathbf{X}_1), ..., S(\mathbf{X}_N)$ and order them. Let $\hat{\gamma}_t$ be the $1-\rho$ sample quantile, above which we identify the "elite" samples.
4. Use the same samples to calculate
$$\hat{v}_{t,j} = \frac{\sum_{i=1}^{N} I_{\{S(\mathbf{X}_i)>\hat{\gamma}_t\}} W(\mathbf{X}_i, \mathbf{u}, \hat{\mathbf{v}}_{t-1}) h(x_{ij})}{\sum_{i=1}^{N} I_{\{S(\mathbf{X}_i)>\hat{\gamma}_t\}} W(\mathbf{X}_i, \mathbf{u}, \hat{\mathbf{v}}_{\mathbf{t-1}})} \tag{3}$$
5. If $\hat{\gamma}_t = \gamma$, continue; else $t = t+1$ and return to 2
6. Generate a sample $\mathbf{X}_1, ..., \mathbf{X}_{N_1}$ from $f(\mathbf{x}, \hat{\mathbf{v}}_t)$ and estimate
$$l = \frac{1}{N_1}\sum_{i=1}^{N_1} I_{\{S(\mathbf{X}_i)>\hat{\gamma}_t\}} W(\mathbf{X}_i, \mathbf{u}, \hat{\mathbf{v}}_{t-1}) \tag{4}$$

Note that, although step 3 looks formidable, it is actually only counting the fraction of samples which satisfy the current criterion. The "cross entropy method" is so-called since we wish to minimise the Kullback-Leibler divergence between the data distribution and the importance sampling distribution which is related to the cross entropy via

$$CE_{f(.,v)}(f(.,u)) = H(f(.,v)) + KL(f(.,v), f(.,u)) \tag{5}$$

where CE is the cross entropy, KL is the Kullback-Leibler divergence and $H(.)$ is the Shannon entropy. Since $H(.)$ is constant, [10] equates the cross entropy with the Kullback-Leibler divergence.

2.2 Cross Entropy Method for Optimization

For unsupervised data exploration, we need to turn the data exploration methods into the so-called *associated stochastic problem*(ASP) firstly. The basic method is

- Generate random samples from the associated stochastic problem using some randomization method.
- Update the parameters (which will typically be parameters of the pdf generating the samples) to make the production of better samples more likely next time. For a Gaussian distribution, this results in updates only to the mean, μ, and covariance, Σ.

Note that, unlike the rare event simulations, we do not have a base parameterisation to work to and hence no need to have the $W(\mathbf{X}_i, \mathbf{u}, \hat{\mathbf{v}}_{t-1})$ term in the calculation. Of course we could have this term defined in terms of $\hat{\mathbf{v}}_t$ and $\hat{\mathbf{v}}_{t-1}$ but [10] show that this is not essential and indeed tends to introduce unnecessary noise to the convergence.

We usually wish to maximize some performance function $S(\mathbf{x})$ over all states $\mathbf{x}$ in data set $\aleph$. Denoting the maximum by γ^*, we have

$$\gamma^* = \max_{\mathbf{x}\in\aleph} S(\mathbf{x}) \tag{6}$$

Thus, by defining a family of pdfs $\{f(.;\mathbf{v}), \mathbf{v} \in \nu\}$ on the data set $\aleph$, we follow [10] to associate with (6) the following estimation problem

$$l(\gamma) = \mathbf{P}_v(S(\mathbf{X}) \geq \gamma)) = \mathbf{E}_v I_{\{S(\mathbf{x})>\gamma\}} \tag{7}$$

where $\mathbf{X}$ is a random vector with pdf $f(.;\mathbf{v}), \mathbf{v} \in \nu$. To estimate l for a certain γ close to γ^*,we can make adaptive changes to the probability density function according to the Kullback-Leibler cross-entropy. Thus we create a sequence $f(.;\mathbf{v}_0), f(.;\mathbf{v_1}), f(.;\mathbf{v_2}), \ldots$ of pdfs that are optimized in the direction of the optimal density and for the fixed $\hat{\gamma}_t$ and $\hat{\mathbf{v}}_{t-1}$, we derive the $\hat{\gamma}_t$ from the following program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) \quad = \quad \max_{\mathbf{v}} \frac{1}{\mathbf{N}}\sum_{i=1}^{N} I_{\{S(\mathbf{X}_i)>\hat{\gamma}_t\}} \ln f(\mathbf{X}_i; \mathbf{v})$$

One advantage that this representation has is that it is very simple to change the base learner. Typically, we use Gaussian distribution in this paper, which means we need to estimate the mean and variance of the elite samples as

$$\hat{\mu} \quad = \quad \frac{1}{N_{elite}}\sum_{i=1}^{N_{elite}} \mathbf{X}_i$$

$$\hat{\Sigma} = \frac{1}{N_{elite}} \sum_{i=1}^{N_{elite}} (\mathbf{X}_i - \hat{\mu})(\mathbf{X}_i - \hat{\mu})^T$$

3 ICA as Associated Stochastic Problem

We stated that the main idea of combinatorial optimization via cross entropy methods is to first turn each Combinatorial Optimization Problem into a rare-event estimation problem, the so-called *associated stochastic problem*(ASP). We wish to maximize some performance function $S(\mathbf{x})$ over all states $\mathbf{x}$ in data set $\aleph$. We define a family of pdfs $\{f(.\,;\mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ on the data set and optimize the parameters $\mathbf{v} \in \mathcal{V}$ of these pdfs in the direction of the optimal density so that we can then obtain an estimate of the "reference parameter vector" $\mathbf{v}^*$ via the CE algorithm.

In this section, we investigate a new ICA algorithm with the cross-entropy method for optimization. We use stochastic units drawn from a Gaussian distribution to sample the elements of the demixing matrix, $\mathbf{w}_i$. With the CE algorithm, we make adaptive changes to the probability density function of $\mathbf{w}_i$ according to the value of the performance function. Since we adopt the basic idea that all components should be as non-Gaussian as possible, the performance function can be the measurement of the non-gaussianity of a vector, for example by using kurtosis. The reference parameters of the pdfs are thus steered in the direction of the theoretically optimal density.

To be specific: we wish to transform a set of observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that are the result of a linear mixing of statistically independent sources $\mathbf{s}_1, \ldots, \mathbf{s}_n$ by $\mathbf{x} = \mathbf{A}\mathbf{s}$, into several components that are statistically independent by $\mathbf{y} = \mathbf{W}\mathbf{x}$. To maximize the absolute values of the kurtosis of each output component with the CE algorithm for optimization, we wish to maximize the performance function

$$S_i(\mathbf{w}_i) = |\mathtt{kurt}(\mathbf{w}_i^T \mathbf{x})| \qquad (8)$$

where i is the index of the independent component. We denote the maximum by γ^*, thus $\gamma^* = \max_{\mathbf{w}} S(\mathbf{w})$. We consider the associated stochastic problem (ASP) that

$$l(\gamma) = \mathbf{P}_{\mathbf{u}}(S(\mathbf{w}) \geq \gamma) = \mathbf{E}_{\mathbf{u}} I_{\{S(\mathbf{w}) > \gamma\}} \qquad (9)$$

which is our estimate of l for γ close to γ^* and typically $S(\mathbf{w}) > \gamma$ is a rare event. We assume the demixing weight, $\mathbf{w}_i$, is sampled from the Gaussian distribution with mean $\mathbf{m}$ and variance β^2. With the CE algorithm, we make adaptive changes to the probability density function, specifically the mean $\mathbf{m}$ and the variance β^2 and the update rule for the parameters is in the form of

$$\hat{\mathbf{m}} = \frac{\sum_{i=1}^{N} I_{\{S(\mathbf{w}_i) \geq \hat{\gamma}_t\}} \mathbf{w}_i}{\sum_{i=1}^{N} I_{\{S(\mathbf{w}_i) \geq \hat{\gamma}_t\}}} \qquad (10)$$

and

$$\hat{\beta}^2 = \frac{\sum_{i=1}^{N} I_{\{S(\mathbf{w}_i) \geq \hat{\gamma}_t\}} (\mathbf{w}_i - \hat{\mathbf{m}})^2}{\sum_{i=1}^{N} I_{\{S(\mathbf{w}_i) \geq \hat{\gamma}_t\}}}. \qquad (11)$$

Moreover, we wish to maintain the online update capabilities of cross entropy and so we use the batch-elite method described above rather than an instantaneous gradient ascent update. Actually [10] has already moved some way towards this in that a "smoothed version" of the parameter updating is used:

$$\mathbf{v}_t = \alpha \hat{\mathbf{v}}_t + (1 - \alpha) \hat{\mathbf{v}}_{t-1} \qquad (12)$$

where α is called the *smoothing parameter* with $0.7 \leq \alpha \leq 1$

Thus we summarize our ICA algorithm with CE method as follows:

1. Initialize $\mathbf{m}_0$ and β_0^2. Set t=1.
2. Generate random samples, $\mathbf{w}_1, ..., \mathbf{w}_N$ from the density $\mathcal{N}(\mathbf{m}_{t-1}, \beta_{t-1}^2)$. From the second independent component onwards, ensure $\mathbf{w}_1, \ldots, \mathbf{w}_N$ is orthogonal to the previous weights, for example simply by using the Gram-Schmidt deflation method.
3. Compute the sample $(1 - \varrho)$ quantile $\hat{\gamma}_t$ of the performance function according to (8).
4. Use the "elite" samples and update the reference parameters, $\mathbf{m}$ and β^2 according to (10) and (11). We denote the solution by $\hat{\mathbf{v}}_t$.
5. Apply (12) to smooth out the vector $\mathbf{v}_t$.
6. If not finished, return to step 2.

To demonstrate how well our ICA algorithm works, we use a 3-D real data set, in which the original signals are pieces of records from different humans speaking as shown in the left column of Figure 1. The number of observations is 40000 and we set $\varrho = 0.1, N = 50, \alpha = 0.7$. The number of iterations for each independent component is 20. The recovered independent components are shown in Figure 2. The batch learning is performed with the Gram-Schmidt method.

It is clear that all independent components have been separated as shown in Table 1 and the Amari error in this case is 0.0115. We find that the ICA algorithm converges so quickly that the simulation time

Figure 1: 3-D real data set used in simulation. Left: the original data set. Right: the mixed observations.

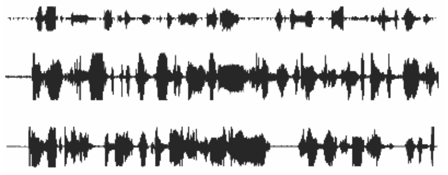

Figure 2: The ICs recovered by the ICA network with 3-D real data set.

is less than 60 seconds. Note that with a simulation using reinforcement learning, in which even when the number of observations is only 10000, the simulation time was around 15 minutes. Moreover, we can see that the CE algorithm has identified all the independent components with high accuracy from Table 2.

4 The On-line Cross Entropy Method in unsupervised data exploration

In this paper, we wish to maintain the on-line cross entropy method, considering that the performance function may be too complex to optimize. Therefore we generate samples or actions from the current estimated distributions but do not have a set of target answers for these samples/actions and then get an estimated performance function value from the generated samples or actions. We use stochastic units drawn from a Gaussian distribution to sample a variety of network weights, which means the weights $\mathbf{W}$ are all drawn from $N(\mu, \beta^2 I)$, the Gaussian distribution with mean μ and variance β^2.

	Kurt. 1	**Kurt. 2**	**Kurt. 3**
Originals	10.7588	9.8449	6.6837
Mixtures	6.5276	7.2470	8.1544
Recovered ICs	10.7590	9.8428	6.6834

Table 1: The kurtosis of the original signals, mixed observations and recovered independent components (ICs).

1.0000	0.0036	0.0008
0.0045	**1.0000**	0.0094
0.0031	0.0099	**0.9999**

Table 2: Correlation between the original sources and recovered signals.

We will also use an instantaneous gradient ascent update rather than the batch-elite method to optimize mean μ and variance β^2. As stated [10] has already developed a smoothed version of the parameter updating and uses:

$$\hat{\mathbf{v}}_t = \alpha \tilde{\mathbf{v}}_t + (1-\alpha)\hat{\mathbf{v}}_{t-1} \tag{13}$$

where $\tilde{\mathbf{v}}_t$ is the update which would have been found by the simple version of the algorithm described above. Thus, we create an initial M samples, $\mathbf{w}_{0,1}, ..., \mathbf{w}_{0,M}$ from $N(\mathbf{m}, \beta^2)$, an $n-$dimensional isotropic Gaussian distribution with centre, $\mathbf{m}$. We calculate the performance values for each of these samples and put the samples in order of increasing performances, $\mathbf{w}_{0,p(1)}, ..., \mathbf{w}_{0,p(M)}$ where $p(i) \in 1, ..., M$ and select the r highest performing samples, $\mathbf{w}_{0,p(M-r)}, ..., \mathbf{w}_{0,p(M)}$. We then move the distribution closer to these 'elite' samples, with the learning rules

$$\begin{aligned} \mu &\leftarrow (1-\eta)\mu \\ &+ \eta(\frac{1}{r}\sum_{j=M-r}^{M} \mathbf{w}_{0,p(j)} - \mu) \end{aligned} \tag{14}$$

$$\begin{aligned} \beta^2 &\leftarrow (1-\eta)\beta^2 \\ &+ \frac{\eta}{r}\sum_{j=M-r}^{M} (\mathbf{w}_{0,p(j)} - \mu)^T(\mathbf{w}_{0,p(j)} - \mu) \end{aligned} \tag{15}$$

where η is the learning rate. We then create another M samples, $\mathbf{w}_{1,1}, ..., \mathbf{w}_{1,M}$ from the new distribution, and iterate the process. We point out that η is also the parameter to reduce the possibility that $\mathbf{w}_{t,p(j)}$ will be zero or one, so that the mean and variance can be updated smoothly. This is important for the problem in continuous space such as principal component analysis. For $\eta = 1$, one might have either $\mathbf{w}_{t,p(j)} = 0$ or $\mathbf{w}_{t,p(j)} = 1$, which causes the algorithm converge to a wrong solution.

We summarize our algorithm as follows:

1. Select one data from the data set randomly

2. Generate a sample $\mathbf{w}_{t,1}, ..., \mathbf{w}_{t,M}$ from the currently estimated distribution $N(\mathbf{m}_t, \beta_t^2)$ and

PC1	0.0144	0.0086	-0.0061	-0.0282	**-0.9994**
PC2	0.0136	-0.0411	-0.0111	**0.9986**	-0.0283
PC3	-0.0127	-0.0076	**0.9998**	0.0108	-0.0067
PC4	0.0418	**0.9982**	0.0077	0.0409	0.0080
PC5	**0.9989**	-0.0414	0.0126	-0.0147	0.0144

Table 3: The weights from the artificial data experiment for five principal components by cross entropy method

compute the sample $1-\varrho$-quantile $\hat{\gamma}_t$ of the perfomance according to the performance function

3. Update the parameters of the distribution to make a higher reward more likely in future with (15) and (14).

4.1 Principal Component Analysis

Principal Component Analysis (PCA) finds the linear projection of a data set with maximal variance (or alternatively which gives a projection with minimal mean squared error between the projection and the original data set). Let the data be $\mathbf{x} \in X \subset R^n$. Then PCA finds the n-dimensional vector $\mathbf{w}$ such that the variance of $(\mathbf{w}^T\mathbf{x})$ is maximal (or again alternatively such that $(\mathbf{w}^T\mathbf{x})\mathbf{w}$ is closer (in Euclidean norm) to $\mathbf{x}$ over all the distribution than any other linear projection).

To perform PCA, we use the performance function $r = \frac{1}{1+\exp(-\gamma|\mathbf{w}^T\mathbf{x}|)}$. To identify multiple components, in this paper, we use Gram-Schmidt method as the deflation method. Thus, from the second component, right after generating a sample $\mathbf{w}_j$ from the currently estimated distribution $N(\mathbf{m}_t, \beta_t^2 I)$, we subtract $(\mathbf{w}_j^T\mathbf{m}_k)\mathbf{m}_k, k = 1, 2, \ldots, j-1$ from $\mathbf{w}_j$.

To illustrate our algorithm, we create a 5-dimensional artificial data set of 1000 samples, whose elements are drawn independently from Gaussian distributions with $x_i \sim N(0, i)$, so x_5 has the greatest variance and x_1 has the lowest variance. We can see that the five principal components have been identified clearly as shown in Table 3 and our algorithm converges smoothly as shown in Figure 3.

4.1.1 Deflationary orthogonalization in performance function

Another way in which to identifying multiple components is to integrate deflationary orthogonalization with the perfocmance function directly. The basic idea is to eliminate the parts that contain the information of the previous components having been estimated from the current component vector being estimated. In the PCA problem, assuming we have found

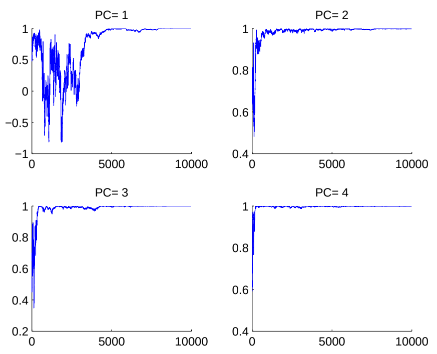

Figure 3: Convergence of the PCA weight vectors to the optimal directions

$j-1$ principal components, $\mathbf{w}_k, k = 1, 2, \ldots, j-1$, thus unless the j^{th} component is orthogonal to the components having been estimated, it contains the information of the previous components that can be work out by the sum of projection between the current weight vector $\mathbf{w}_j$ to be estimated and $\mathbf{w}_k, k = 1, 2, \ldots, j-1$ having been estimated and denoted by $\Sigma_j = \sum_{k=1}^{j-1}(\mathbf{w}_j^T\mathbf{m}_k)\mathbf{m}_k, k = 1, 2, \ldots, j-1$. Therefore, we re-define the performance function as

$$S(\mathbf{w}) = \begin{cases} \frac{1}{1+exp(-\gamma|\mathbf{w}_1^T\mathbf{x}|)} & \text{if j=1} \\ \frac{1}{1+exp(-\gamma|\mathbf{w}_j^T\mathbf{x}-\Sigma_j^T\mathbf{x}|)} & \text{if j>1} \end{cases} \quad (16)$$

Another way to explain the new perfomance function is PCA problem itself. PCA problem is to find the linear projection of a data set which contains maximal variance, $E(\mathbf{w}^T(x-\bar{x})(x-\bar{x})^T\mathbf{w})$. If the first principal component has been estimated, we have the largest variance, $\mathbf{w}_1^T\mathbf{x}$. Estimating the second principal component leads to the second largest variance. However, before the second principal component vector reaches the target, the variance will also contain part of the first largest variance, $\Sigma_j^T\mathbf{x}$, which must be eliminated.

We demonstrate deflationary orthogonalization in performance function with the same data set. We can see that the five principal components have been identified clearly as shown in Table 4.

4.2 Exploratory Projection Pursuit

Exploratory Projection Pursuit (EPP) [5, 6] is another projection technique which maximises some index of “interestingness”, instead of maximising variance. Here “interesting” means most projections of a high dimensional data set tend to be Gaussian and somewhat uninteresting, thus we can quantify how “interesting” the projection is as how different the distribution is from Gaussian distribution. The simplest way

PC1	-0.0027	0.0052	0.0222	0.0291	**0.9980**
PC2	0.0123	0.0710	0.0200	**0.9937**	0.0480
PC3	0.0608	-0.0531	**-0.9946**	0.0190	0.0376
PC4	0.0533	**0.9936**	-0.0365	-0.1061	-0.0042
PC5	**0.9966**	-0.0697	0.0928	0.0028	0.0130

Table 4: The weights from the artificial data experiment for five principal components by cross entropy method. Deflationary orthogonalization is integrated into the performance function

EC1	-0.0023	0.0028	-0.0019	0.0051	**1.0000**

Table 5: The weights from the artificial data experiment for EPP by cross entropy method

is to use the third moment or the fourth moment of the distribution.

To illustrate our method, we create 1000 samples of 5 dimensional data set in which 4 elements of each vector are drawn from $N(0,1)$, while the fifth contains data with negative kurtosis: we draw this also from $N(0,1)$, but randomly add or subtract 5. Before performing the algorithm, we sphere the data set by elimating the mean of data set and deviding by the inverse of the square root of its variance to give the data set in all directions that have zero mean and unit variance. We use $S(\mathbf{w}) = |\tanh(\mathbf{w}^\mathbf{T}\mathbf{x})|$ as the performance function. Table 5 shows the outcome of the simulation. We can see that the distribution with negative kurtosis has been identified with high accuracy. We can also see extremely stable and quick convergence, as shown in Figure 4, to the optimal direction.

4.3 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) is used when we believe there is some underlying relationship between two data sets. We can regard such a problem as that of maximizing the objective function $g(\mathbf{w}_1|\mathbf{w}_2) = E(y_1 y_2)$. Such problem is an unconstrained maximization problem which clearly has no finite solution, so we should add the constraint that $E(y_1^2 = 1)$ and similar with y_2. Instead of using multi- reward functions as we have done in reinforcement learning, we define the performance function for the first canonical correlation as

$$\begin{aligned} S(\mathbf{w}) &= |\mathbf{w}_{1j}^T\mathbf{x}_1 - \mathbf{w}_{2j}^T\mathbf{x}_2| + \frac{1}{2}|(\mathbf{w}_{1j}^T\mathbf{x}_1)^2 - 1| \\ &\quad + \frac{1}{2}|(\mathbf{w}_{2j}^T\mathbf{x}_2)^2 - 1| \quad j = 1 \end{aligned} \tag{17}$$

To identify multiple canonical correlations, we eliminate the parts that contain information of the previous components having been estimated from the current component vector being estimated. Thus for $j > 1$, we have

$$\begin{aligned} S(\mathbf{w}) &= |\mathbf{w}_{1j}^T\mathbf{x}_1 - \Sigma_{1j}^T\mathbf{x}_1 - \mathbf{w}_{2j}^T\mathbf{x}_2 + \Sigma_{2j}^T\mathbf{x}_2| \\ &\quad + \frac{1}{2}|(\mathbf{w}_{1j}^T\mathbf{x}_1)^2 - 1| + \frac{1}{2}|(\mathbf{w}_{2j}^T\mathbf{x}_2)^2 - 1| \\ &\geq |\mathbf{w}_{1j}^T\mathbf{x}_1 - \mathbf{w}_{2j}^T\mathbf{x}_2| + |\Sigma_{1j}^T\mathbf{x}_1 - \Sigma_{2j}^T\mathbf{x}_2| \\ &\quad + \frac{1}{2}|(\mathbf{w}_{1j}^T\mathbf{x}_1)^2 - 1| + \frac{1}{2}|(\mathbf{w}_{2j}^T\mathbf{x}_2)^2 - 1| \end{aligned}$$

To illustrate this method, we use an artificial data set similar to what had been used in reinforcement learnng, where there are two sets of artificial data, one is 4-dimensional vector and the other is 3-dimensional vector, each of whose elements is drawn from the zero-mean Gaussian distribution,$N(0,1)$ and we add an additional sample from $N(0,1)$ to the first elements of each vector and then divide by 2 to ensure that there is no more variance in the first elements than in the others. To generate the second correlation, we add an additional sample from $N(0,0.5)$ to the second elements of each vector, which so the second correlation is smaller than that of the first one. We illustrate the convergence of the CCA weight vectors in Figure 5: we can see consistent convergence towards the first two canonical correlation vectors smoothly. Table 6 shows that our algorithm works well in identifying different canonical correlations.

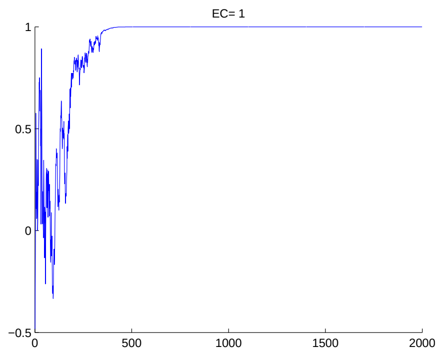

Figure 4: Convergence of EPP simulation by cross entropy method

5 Cross Entropy Latent Variable Models

The most common approach for Latent Variable Models is the maximization of the likelihood function using the EM algorithm. In this section we derive a new

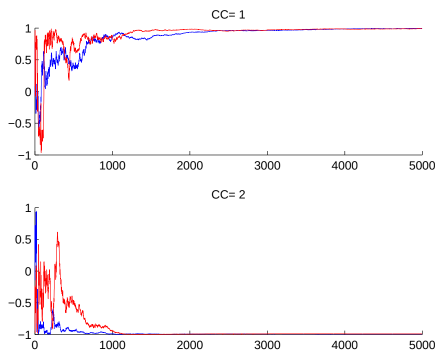

Figure 5: Convergence of the CCA weight vectors to the optimal directions

m1	m2	
0.9950	**0.9905**	CC1
0.0066	-0.1370	
0.0817	0.0113	
-0.0562		
-0.0573	-0.0240	CC2
-0.9914	**-0.9996**	
0.1136	0.0726	
0.0296		

Table 6: The first two Canonical Correlation (CC) weight vectors found for the artificial data

method for the optimization of the likelihood in latent variable models that is based on the *cross entropy* (CE) method. For latent variable models, we turn the optimization of the likelihood function into the associated stochastic problem (ASP) first. The basic method is

- Generate random samples from the associated stochastic problem using some randomization method.
- Update the parameters (which will typically be parameters of the pdf generating the samples) to make the production of better samples that correspond to the parameters to be optimized in the likelihood function next time. For a Gaussian distribution, this results in updates only to the mean, μ_{ce}, and covariance, Σ_{ce}.

5.1 Probabilistic Principal Component Analysis

Consider the $\mathbf{x}$-conditional probability distribution over the data space given by

$$p(\mathbf{t}|\mathbf{x}) = (2\pi\sigma^2)^{-d/2}\exp\left\{-\frac{1}{2\sigma^2}||\mathbf{t}-\mathbf{W}\mathbf{x}-\mu||^2\right\} \tag{18}$$

and $p(\mathbf{x})$ is assumed as a *prior* distribution over the latent variables, which is defined by

$$p(\mathbf{x}) = (2\pi)^{-q/2}\exp\left\{-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right\} \tag{19}$$

We then obtain the joint distribution of $\mathbf{t}$ and $\mathbf{x}$

$$\begin{aligned} p(\mathbf{t}_n,\mathbf{x}_n) &= (2\pi\sigma^2)^{-d/2}\exp\left\{-\frac{1}{2\sigma^2}||\mathbf{t}_n-\mathbf{W}\mathbf{x}_n-\mu||^2\right\} \\ &\quad\times(2\pi)^{-q/2}\exp\left\{-\frac{1}{2}||\mathbf{x}_n||^2\right\} \end{aligned} \tag{20}$$

Thus, the corresponding log-likelihood is given by

$$\begin{aligned} L_c(\theta) &= \sum_n[\ln p(\mathbf{t}_n|\mathbf{x},\mathbf{W},\sigma^2)+\ln p(\mathbf{x})] \\ &= \sum_n\Big[-\frac{q+d}{2}\ln(2\pi)-\frac{d}{2}\ln\sigma^2 \\ &\quad -\frac{1}{2\sigma^2}(\mathbf{t}_n-\mathbf{W}\mathbf{x}-\mu)^T(\mathbf{t}_n-\mathbf{W}\mathbf{x}-\mu)-\frac{1}{2}\mathbf{x}_n^T\mathbf{x}_n\Big] \\ &= -\frac{N(q+d)}{2}\ln(2\pi) \\ &\quad -\sum_{n=1}^{N}\Big\{\frac{d}{2}\ln\sigma^2+\frac{1}{2\sigma^2}(\mathbf{t}_n-\mu)^T(\mathbf{t}_n-\mu) \\ &\quad -\frac{1}{\sigma^2}\mathbf{x}_n^T\mathbf{W}^T(\mathbf{t}_n-\mu) \\ &\quad +\frac{1}{2\sigma^2}\mathtt{tr}(\mathbf{W}^T\mathbf{W}\mathbf{x}\mathbf{x}^T)+\frac{1}{2}\mathtt{tr}(\mathbf{x}\mathbf{x}^T)\Big\} \end{aligned} \tag{21}$$

We omit the first term, $-\frac{N(q+d)}{2}\ln(2\pi)$, and take the expectation of L_c with respect to the distribution $p(\mathbf{x}_n|\mathbf{t}_n, \mathbf{W}, \sigma^2)$, and we have

$$\begin{aligned} E\{L_c\} &= -\sum_{n=1}^{N}\left\{\frac{d}{2}\ln\sigma^2\right. \\ &+ \frac{1}{2\sigma^2}(\mathbf{t}_n-\mu)^T(\mathbf{t}_n-\mu) \\ &-\frac{1}{\sigma^2}E\{\mathbf{x}_n\}^T\mathbf{W}^T(\mathbf{t}_n-\mu) \\ &+ \frac{1}{2\sigma^2}\mathtt{tr}(\mathbf{W^TW}E\{\mathbf{xx}^T\}) \\ &+ \left.\frac{1}{2}\mathtt{tr}(E\{\mathbf{xx}^T\})\right\} \end{aligned} \qquad (22)$$

where $E\{\mathbf{x}\}$ is defined as the mean of the conditional distribution $p(\mathbf{x}_n|\mathbf{t}_n, \mathbf{W}, \sigma^2)$

$$E\{\mathbf{x}\} = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{t}-\mu) \qquad (23)$$

and $E\{\mathbf{xx^T}\}$ is defined as the variance of the conditional distribution $p(\mathbf{x}_n|\mathbf{t}_n, \mathbf{W}, \sigma^2)$

$$E\{\mathbf{xx}^T\} = \sigma^2\mathbf{M}^{-1} + E\{\mathbf{x}\}E\{\mathbf{x}\}^T \qquad (24)$$

where $\mathbf{M} = \mathbf{W^TW} + \sigma^2\mathbf{I}$. Thus the parameters to be estimated are $\mu, \mathbf{W}$ and σ^2. The estimation for μ is given by the mean of the data set and σ^2 can be determined by

$$\begin{aligned} \sigma^2 &= \frac{1}{Nd}\sum_{n=1}^{N}\left\{(\mathbf{t}_n-\mu)^T(\mathbf{t}_n-\mu)\right. \\ &\quad -2E\{\mathbf{x}_n\}^T\mathbf{W}^T(\mathbf{t}_n-\mu) \\ &\quad \left.+\mathtt{tr}(\mathbf{W}^T\mathbf{W}E\{\mathbf{xx}^T\})\right\}. \end{aligned} \qquad (25)$$

Therefore, the likelihood function of (22) is a function of the parameter $\mathbf{W}$. To optimize $\mathbf{W}$, we incoporate the cross entropy method into the optimization of the likelihood function by considering that (22) is the performance function $S(\mathbf{W})$ to be maximized over the parameter θ in the cross entropy method. Denoting the maximum by θ^*, we have

$$\gamma^* = \max_{\theta} S(\theta) \qquad (26)$$

Thus, by defining a family of pdfs $\{f(.;\mathbf{v}), \mathbf{v}\in\nu\}$ on the parameter θ, we follow [10] to associate with (6) the following estimation problem

$$l(\gamma) = \mathbf{P}_v(S(\theta)\geq\gamma) = \mathbf{E}_v I_{\{S(\theta)>\gamma\}} \qquad (27)$$

PC1					sigma
0.0028	-0.0160	0.0214	-0.0595	**0.9991**	7.2971

Table 7: The weights from the artificial data experiment for the 1st principal component by cross entropy method. The parameters of the performance function are $\mathbf{W}$.

where θ is the parameters in (22). We use stochastic units drawn from a Gaussian distribution to sample θ, which means the set of parameters θ is drawn from $\mathcal{N}(\mathbf{m}, \beta^2 I)$, the Gaussian distribution with mean $\mathbf{m}$ and variance β^2. To estimate l for a certain γ close to γ^*, we make adaptive changes to the probability density function $\mathcal{N}(\mathbf{m}, \beta^2 I)$ according to the Kullback-Leibler cross-entropy. Thus we create a sequence $f(.;\mathbf{v}_0), f(.;\mathbf{v}_1), f(.;\mathbf{v}_2), \ldots$ of pdfs that are optimized in the direction of the optimal density and for the fixed $\hat{\gamma}_t$ and $\hat{\mathbf{v}}_{t-1}$, we derive the $\hat{\gamma}_t$ from the following program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{\mathbf{N}}\sum_{i=1}^{N} I_{\{S(\theta_i)>\hat{\gamma}_t\}}\ln f(\theta_i;\mathbf{v}) \qquad (28)$$

Since we use a Gaussian distribution as the base learner, the mean and variance of the elite samples are estimated as

$$\mathbf{m} \leftarrow (1-\eta)\mathbf{m} + \eta\frac{1}{r}\sum_{j=M-r}^{M}\mathbf{w}_{p(j)} \qquad (29)$$

$$\begin{aligned} \beta^2 &\leftarrow (1-\eta)\beta^2 \\ &+ \eta(\frac{1}{r}\sum_{j=M-r}^{M}(\mathbf{w}_{p(j)}-\mathbf{m})^T(\mathbf{w}_{p(j)}-\mathbf{m})) \end{aligned} \qquad (30)$$

To illustrate our algorithm, we use the same artifical data set as before: the size of the data set is 1000 and we set the number of iteration as 50. We can see that the first principal component has been identified accurately as shown in Table 7 and our algorithm converges quickly as shown in Figure 6. We also find that the σ^2 in Table 7 is close to that calculated by the estimated lost variance, $\sigma^2_{\mathbf{ML}} = \frac{1}{D-q}\sum_{j=q+1}^{D}\lambda_j$, which equals the average variance over the lost dimensions.

Futhermore, given that μ is calculated by the mean of the data set, we can optimize $\mathbf{W}$ and σ^2 simultaneously by considering that (22) is the performance function $S(\mathbf{W}, \sigma^2)$ to be maximized over the parameters, $\mathbf{W}$ and σ^2. We can see in Table 8 that the first principal component has been identified accurately and Figure 7 shows that our algorithm converges quickly.

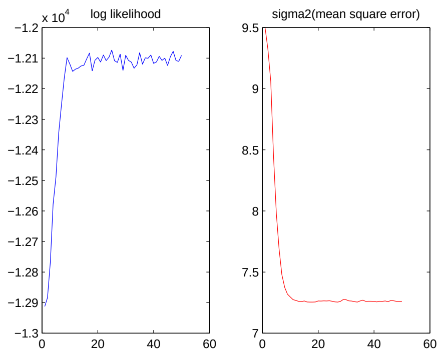

Figure 6: Convergence of the PCA weight vectors to the optimal directions.

PC1					sigma
-0.0007	-0.0081	-0.0369	-0.0830	**0.9958**	6.5497

Table 8: The weights from the artificial data experiment for the 1st principal component by the cross entropy method. The parameters in the performance function are $\mathbf{W}$ and σ^2.

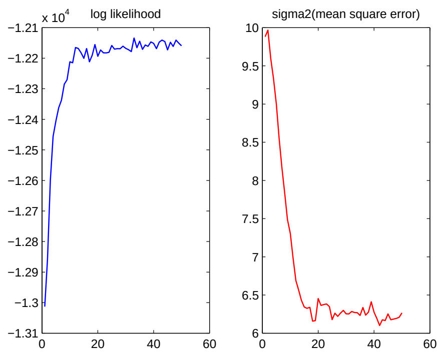

Figure 7: Convergence of the PCA weight vectors to the optimal directions. The parameters in the performance function are $\mathbf{W}$ and σ^2.

	Kurtosis 1	**Kurtosis 2**
Original signals	3.2958	3.9589
Mixed observations	3.8613	3.5871
Recovered ICs	3.2970	3.9547

Table 9: The kurtosis of the original signals, mixed observations and recovered independent components (ICs).

5.2 Independent Component Analysis

MacKay [11] has derived a maximum likelihood algorithm in which independent component analysis is performed based on a latent variable model. Defining $\mathbf{V}$ as the mixing matrix and $\mathbf{W}$ as the demixing matrix, the mixed observations $\mathbf{x}$ are generated from latent variables $\mathbf{s}$ that are independently distributed, with marginal distributions $p_i(s_i)$, through the linear mapping $\mathbf{x} = \mathbf{V}\mathbf{s}$. Then the likelihood is given by

$$\begin{aligned} P(\mathbf{x}|\mathbf{V}) &= \int P(\mathbf{x}|\mathbf{s})P(\mathbf{s})d\mathbf{s} \\ &= \int \prod_j \delta(\mathbf{x}_j - V_{ij}\mathbf{s}_i) \prod_i p_i(\mathbf{s}_i)d\mathbf{s} \\ &= \frac{1}{\det \mathbf{V}} \prod_i p_i(V_{ij}^{-1}\mathbf{x}_j) \end{aligned} \qquad (31)$$

where i is the index of the dimension of the sources and j is the index of the dimension of the observations. The log likelihood is then defined as follows

$$\begin{aligned} \log P(\mathbf{x}|\mathbf{V}) &= -\log|\det \mathbf{V}| + \textstyle\sum_i \log p_i(\mathbf{V}_{ij}^{-1}x_j) \\ &= \log|\det \mathbf{W}| + \textstyle\sum_i \log p_i(\mathbf{W}_{ij}x_j) \end{aligned} \qquad (32)$$

where $\mathbf{W} \equiv \mathbf{V^{-1}}$. [11] has stated that the distribution of the sources can be assumed as $p_i(s_i) \propto \frac{1}{\cosh(s_i)}$. Therefore, we define the performance function as

$$S(\mathbf{W}) = N \log|\det \mathbf{W}| + \sum_{j=1}^{D}\sum_{n=1}^{N} \frac{1}{\cosh(\mathbf{w}_j\mathbf{x}_n)} \qquad (33)$$

Thus (33) is to be maximized over the set of unmixed weight vectors $\mathbf{W}$ that are drawn from $\mathcal{N}(\mathbf{m}, \beta^2 I)$, the Gaussian distribution with mean $\mathbf{m}$ and variance β^2, which is estimated as (14) and (15). It is worth noting that if we assume that the number of sources is equal to the number of observations, instead of measuring different independent components sequentially by deflation methods, all the independent components can be evaluated simultaneously.

To illustrate our algorithm, we use the 2-dimensional real data set, 'chirp' and 'gong', provided

1.0000	0.0276
0.0026	**0.9996**

Table 10: Correlation between the original sources and recovered signals.

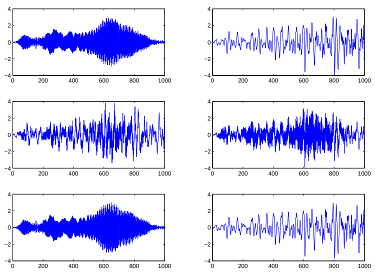

Figure 8: ICA as latent variable model with CE algorithm. Top: the original signals. Middle: mixed observations. Bottom: recovered ICs.

by Matlab and the number of signals is 1000. We can see all the independent components have been identified as shown in Figure 8 and our algorithm is highly accurate as shown in Table 10. The Amari error in this case is 0.0242.

5.3 Topology Preserving Manifolds

A topographic mapping captures some structure in the data set, in which points which are mapped close to one another have some common feature while points that are mapped far from one another do not share this feature. The most common topographic mappings are Kohonen's self-organizing map (SOM) [9]. The Generative Topographic Mapping (GTM) [3] is a mixture of experts model which treats the data as having been generated by a set of latent points. In this paper, we follow [3, 7] to create a latent space of points $t_1, t_2, \ldots, t_K$ which lies equidistantly on a line or at the corners of a grid. To allow non-linear modeling, we map these latent points through a set of M basis functions, typically squared exponentials centered in latent space, and then map the output of the basis functions to points, $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K$ through a set of weights. Thus, we have

$$\begin{aligned}\mathbf{m}_k &= \sum_{j=1}^{M} \mathbf{w}_j \Phi_j(\mathbf{t}_k), k = 1, \ldots, K \\ &= \sum_{j=1}^{M} \mathbf{w}_j \exp(-\lambda ||\mu_j - \mathbf{t}_k||^2) \qquad (34)\end{aligned}$$

where $\phi_j(), j = 1, \ldots, M$ are the M basis functions, and $\mathbf{w}_j$ is the weight from the j^{th} basis function to the data space. The algorithm is:

1. Randomly select a data point, $\mathbf{x}$.
2. Find the closest prototype, say $\mathbf{m}_{k*}$, to $\mathbf{x}$.
3. Generate T samples from the Gaussian distribution, $N(\mathbf{m}_{k*}, \beta^2_{k*})$. Call the samples, $\mathbf{y}_{k*,1}, ..., \mathbf{y}_{k*,T}$. We note that we are using $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K$ to perform two conceptually separate functions, as prototypes or means to which the data will be quantised and as centres of Gaussian distributions from which samples will be drawn.
4. Evaluate the samples using $S(\mathbf{y}) = \exp(- \parallel \mathbf{y} - \mathbf{x} \parallel^2)$ as the performance function.
5. Sort the samples using $p(1)$ as the worst to $p(T)$ as the best. i.e. we are identifying the r elite samples.
6. Update the parameters

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + \eta(\frac{1}{r}\sum_{j=T-r}^{T} \mathbf{y}_{k*,p(j)} - \mathbf{m}_{k*})\phi(\mathbf{t}_{k*}) \\ \beta^2_{k*} &= \beta^2_{k*} \\ &+ \frac{\eta_0}{r}\sum_{j=T-r}^{T}(\mathbf{y}_{k*,p(j)} - \mathbf{m}_{k*})(\mathbf{y}_{k*,p(j)} - \mathbf{m}_{k*})^T\end{aligned}$$

   where η, η_0 are the learning rates with typically $\eta = 10\eta_0$.
7. Update the prototypes positions using (34).

Figure 9 shows the result of a simulation in which there are 20 latent points lying equally spaced in a one dimensional latent space, passed through 5 basis functions and mapped to the data space by the linear mapping W. We generate 1000 2-dimensional data points,

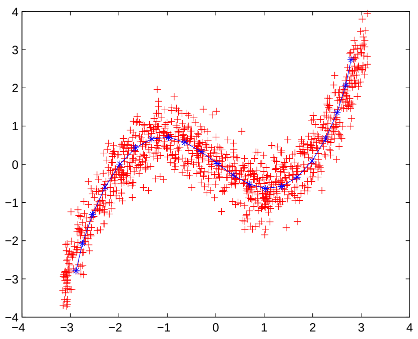

Figure 9: The data are shown by '+'s and the latent points' projections are shown by '*'s

(x_1, x_2), from the function $x_2 = x_1 + 1.25\sin(x_1) + \rho$, where ρ is the noise from a uniform distribution in $[0, 1]$. The number of iterations is 5000. the latent points' projections are shown in the figure. We clearly see that the one dimensional nature of the data has been identified.

6 Multilayer Topology Preserving Manifolds

Deep architectures are compositions of many layers of adaptive non-linear components, which allow representations of functions in a more compact form than shallow architectures. Bengio and LeCun [2] have demonstrated that deep architectures are often more efficient for solving complicated problems in terms of number of computational components and parameters. A greedy, layer-wise unsupervised learning algorithm [1, 8] has recently been introduced to provide an initial configuration of the parameters with which a gradient-based supervised (backpropagation) learning algorithm is initialized, which results in a very much more efficient learning machine. The idea behind the greedy algorithm is that simpler models are learned sequentially and each model in the sequence recieves a different representation of the data. Thus features produced by the lower layers represent lower-level abstractions, which are combined to form high-level features at the next layer, representing higher-level abstractions.

We extend the topology preserving mapping with cross entropy to a multilayer topology preserving mapping. In each layer, we create a q-dimensional latent space with a regular array of points, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_K)$ that have the structure of lying equidistantly on a line or on a grid. These latent points are nonlinearly mapped to points, $(\mathbf{m}_1, \ldots, \mathbf{m}_K)$ in the input space through a set of basis function, which forms a set of reference vectors, $\mathbf{m}_i = \mathbf{W}\phi(\mathbf{x}_i)$. Each of the reference vector then forms the centre of the Gaussian distribution in the input space and we can represent the distribution of the data points in input space in terms of a smaller q-dimensional nonlinear manifold. Thus we have a higher-level representation of the data points in the input space by the projection of the data points in the latent space. Denoting the latent space representation of each data point as $\mathbf{t}_n^{latent}$, we have

$$\mathbf{t}_n^{latent} = \sum_{i=1}^{K} r_{ni}\mathbf{x}_i \tag{35}$$

$$r_{ni} = \frac{\exp(-\gamma d_{ni}^2)}{\sum_k \exp(-\gamma d_{nk}^2)} \tag{36}$$

where r_{ni} is responsibility of the i^{th} latent point for the n^{th} data point and $d_{pq} = ||\mathbf{t}_p - \mathbf{m}_q||$, is the euclidean distance between the p^{th} data point and the projection of the q^{th} latent point. The projection of the data points in the latent space of this layer then becomes the data points in the input space of the next layer. Therefore, the topology preserving mapping in each layer performs a non-linear transformation on its input vectors and produces as output the vectors that will be used as input for the topology preserving mapping in the next layer and the projection of the data points in higher layers may represent more abstract features, whereas lower layers extract low-level features from the data set.

To demonstrate our multilayer topology preserving mapping algorithm, we create a deep architecture model with four layers, in each layer of which we use a 2 dimensional grid of latent points: we use a 21×21 grid of latent points being passed through a 5×5 set of basis vectors. We illustrate the algorithm on the well-known wine data set from the UCI Repository of machine learning databases. It has 178 samples, 13 features and 3 classes. Because some of the features are scaled up to 1500 and others lie between 0 and 1, we preprocessed the data by normalizing all features between -1 and 1. We develop a new way to represent the experimental results, which makes it easier for us to evaluate how well the algorithm works: we calculate the sum of the responsibility vectors over the data points, $R = \sum_{n=1}^{N} \mathbf{r}_n$, and then form a responsibility grid using R for each latent point. Since the latent points are fixed in (35), the data points belonging to the same cluster will have similar responsibility vectors, and thus the area where one cluster is located in the responsibility grid will become 'hot', which allows us to identify the clusters easily.

A plot of the responsibility grid and the projection of the data points in the latent space in each layer is shown in Figure 10. We can see that although the data points have been mapped into the latent space accurately, there are only several small hot areas in the responsibility grid, which means we can not identify different clusters directly. We also find that the higher-level the layer is, the hotter the areas corresponding to different cluters is and in the fourth layer, we can see that the three clusters have been identified clearly and the data points belonging to the same cluster is much closer than in the first layer in the latent space. Therefore, we consider the multilayer topology preserving mapping model has extracted more abstract features in the higher layer. In addition, we can also obviously see that in the fourth layer, the 'square' cluster and the 'cycle' cluster are closer to each other, which are far away from the 'cross' cluster.

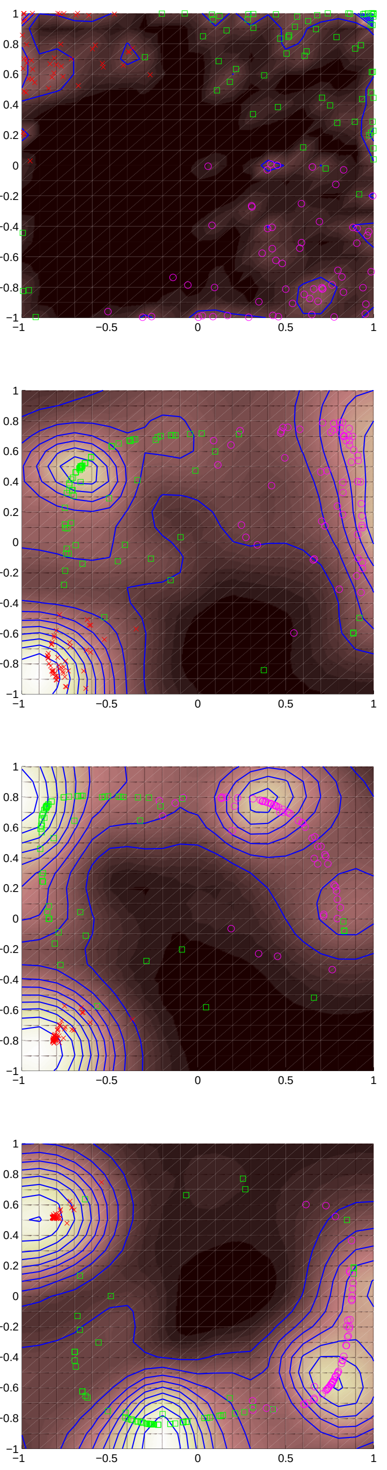

Figure 10: Plot of the responsibility grid and the projection of the data points in the latent space in each layer. Each contour line represents a constant value of R. The cross, square and cycle correspond to the data points for different clusters. From the top, the 1^{st} layer,the 2^{nd} layer, the 3^{rd} layer and the 4^{th} layer.

7 Conclusion

In this paper, we have investigated several unsupervised data exploration techniques with the on-line cross-entropy method. We regard such problems as an associated stochastic problem and also we consider our on-line cross-entropy method to be a close relative of reinforcement learning for perform data exploration. Thus, we use stochastic units drawn from a Gaussian distribution to sample the network weights. With the CE algorithm, we make adaptive changes to the probability density function of network weights according to the Kullback-Leibler cross-entropy. by using a variety of appropriate performance functions, we can see that all the results are accurate and stable.

Although Gram-schmit deflation method works well in identifying multiple components, in this paper a new way by integrating deflationary orthogonalization with the perfocmance function directly is derived. It is worth noting that different from what we have done in reinforcement learning where the immediate reward is weakened by eliminating sum of angle or projection between the current weight vector to be estimated and the wight vectors having been estimated, the basic idea is to eliminate the parts that contain the information of the previous components having been estimated from the current component vector being estimated. Comparing both of the two deflation methods, we think the method used in reinforcement learning is more general to different problems, but suffering being diffecult to set the parameters, and however, deflationary orthogonalization in perfocmance function shares the same idea, but performed in different way for different problems. We think deflationary orthogonalization in perfocmance function is more direct and more stable.

We have also shown how the cross entropy method can be used to optimize the parameters for latent variable models and have illustrated its use on probabilistic principal component analysis and independent component analysis.

Finally we have shown how the CE method may be used for deep architecture networks which are one of the main current hopes for true artificial intelligence. Future work will investigate other deep architectures with the cross entropy method.

References:

[1] Y. Bengio, P. Lamblin, D. Popovici, and Larochelle H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, volume 19, pages 153–160. MIT Press, 2007.

[2] Y. Bengio and Y. LeCun. *Scaling Learning Algorithms towards AI*, chapter Large-Scale Kernel Machines. MIT Press, 2007.

[3] C.M. Bishop, M. Svensen, and C.K.I. Williams. Gtm: A principle alternative to the self-organizing map. *In Advances in neural information processing systems*, 5:354–360, 1997.

[4] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubenstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2004.

[5] J.H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.

[6] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory daya analysis. *IEEE Transactions on Computers*, c-23(9):881–889, Sept 1974.

[7] C. Fyfe. Two topographic maps for data visualization. *Data Mining and Kownledge Discovery*, 14:207–224, 2007.

[8] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 16:1527–1554, 2006.

[9] T. Kohonen. *Self-organising maps*. Springer, 1995.

[10] D. P. Kroese and R. Y. Rubinstein. *The Cross Entropy: A Unified Approach to Combination Optimization, Monte-Calo Simulation and Mechine Learning*. Spinger, 2004.

[11] David J.C. MacKay. Maximum likelihood and covariant algorithms for independent component analysis. Technical report, Dept. of Physics, Cambridge University, Dec 1996.

[12] R.Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Reasearch*, 99:89–112, 1997.