Topology preserving mappings using cross entropy adaptation

Ying Wu  
University of Paisley  
School of Computing  
Scotland

Colin Fyfe  
University of Paisley  
School of Computing  
Scotland

Abstract: In this paper, we review the technique of Cross Entropy and apply it in a novel manner to the creation of topology preserving mappings. We create an underlying latent space which is mapped into data space by mapping the latent points through a number of basis functions and then linearly combining the output of these basis functions to create centres or prototypes in data space. In this paper, we use the cross entropy method for adapting the parameters of this linear combination matrix. We further show that the method can be extended to deep architectures, architectures with multiple layers of adaptive units which can be used to try to create more interesting features than can be found with one adaptive layer.

Key–Words: Cross entropy, topology preserving mappings, deep architectures.

1 Introduction

The cross entropy method has been well introduced in [6] and was motivated as an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks [14]. 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 [14]. Generally speaking, the basic mechanism involves an iterative procedure of two phases.

Latent variable modeling [4] is a powerful approach where a set of observed data variables are associated with additional latent variables in some lower dimensional embedded space. The most common approach for Latent Variable Models is the maximization of the likelihood function using gradient-based algorithms or the EM algorithm. We discuss two methods for creating nonlinear mappings which capture the data manifold in such a way as to perform a topology preserving mapping; each of these methods is such that nearby data points are projected into latent space to nearby positions.

There has been much discussion lately on the deficiencies of existing adaptation algorithms such as backpropagation for supervised learning or Hebbian learning for unsupervised learning. In this paper, we investigate a general method for adapting parameters for unsupervised exploratory data analysis by applying the cross entropy method to create topology preserving mappings. The dissatisfaction with existing adaptation algorithms has led to questions about the architectures on which these algorithms are used and has led to a discussion of the merits of deep architectures in which each individual layer is relatively narrow but where the cumulative layers have a power beyond that which is possible with simpler architectures. Thus we extend the above algorithm to develop a multilayer topology preserving mapping for clustering. We demonstrate that different clusters can be identified more clearly in higher layer.

2 The Cross Entropy Method

The cross-entropy (CE) method [6] is a general Monte Carlo approach to combinatorial and continuous multi-extremal optimization [14]. The method originated from the field of rare event simulation, where very small probabilities need to be accurately estimated.

Consider a Monte Carlo simulation which draws instances from the true distribution of events. Such a problem usually requires an inordinate number of draws before enough of the rare events are seen to make a reliable estimate of their probability of occurring. A better way is to use Importance Sampling (IS) [1] which is a general technique for estimating the properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest. However, such
a method has the drawback that the optimal reference parameters to be used in IS are usually difficult to obtain. The cross entropy method thus has the advantage that it provides a simple adaptive procedure for estimating the optimal reference parameters with asymptotic convergence properties. Not only being applied to the estimates of rare-event problems in dynamic models [7, 12], a simple modification of the CE method [14] has been also applied to solve difficult combinatorial optimization problems (COPs), see ([13]).

2.1 Rare-Event Simulation via Cross Entropy

Generally speaking, the basic mechanism of cross entropy 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 cross entropy method generally uses importance sampling rather than simple Monte Carlo methods: if the original pdf of the data is \( f(x) \), then we require to find a pdf, \( g(x) \), 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 \( l = (S(x) > \gamma) \) be the event in which we are interested. Then

\[
    l = \int I_{S(x) > \gamma} \frac{f(x)}{g(x)} g(x)dx
    = E_g \left[ I_{S(x) > \gamma} \frac{f(X)}{g(X)} \right]
\]

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

\[
    \hat{l} = \frac{1}{N} \sum_{i=1}^{N} I_{S(X_i) > \gamma} W(X_i)
\]

where \( W(x) = \frac{f(x)}{g(x)} \) is the likelihood ratio and \( X_i \) are the samples drawn from \( g() \). Then the simplest algorithm [6] depends on working within a family of pdfs whose parameters we update i.e. let \( f(x) = f(x, u) \), \( u \) being a parameter of the family to which \( f() \) belongs; then the basic algorithm is

1. Define \( \hat{v}_0 = u \). Set \( t = 1 \).
2. Generate random samples, \( X_1, ..., X_N \) from \( f(x, \hat{v}_{t-1}) \).
3. Calculate \( S(X_1), ..., S(X_N) \) and order them. Let \( \gamma_t \) be the \( 1 - \rho \) sample quantile, above which we identify the “elite” samples.
4. Use the same samples to calculate

\[
    \hat{u}_{t,j} = \frac{\sum_{i=1}^{N} I_{S(X_i) > \gamma_t} W(X_i, u, \hat{v}_{t-1}) X_{ij}}{\sum_{i=1}^{N} I_{S(X_i) > \gamma_t} W(X_i, u, \hat{v}_{t-1})}
\]

5. If \( \gamma_t = \gamma \), continue; else \( t = t + 1 \) and return to 2
6. Generate a sample \( X_1, ..., X_{N_1} \) from \( f(x, \hat{v}_t) \) and estimate

\[
    l = \frac{1}{N_1} \sum_{i=1}^{N_1} I_{S(X_i) > \gamma_t} W(X_i, u, \hat{v}_{t-1})
\]

Note that, although step 4 looks formidable, it is actually only counting the fraction of samples which satisfy the current criterion. For fixed \( \gamma_t \) and \( \hat{v}_{t-1} \), we derive \( \hat{v}_t \) in step 4 with the target,

\[
    \max_v \hat{D}(v) = \max_v \frac{1}{N} \sum_{i=1}^{N} I_{S(X_i) \geq \gamma_t} W(X_i, u, \hat{v}_{t-1}) \ln f(X_i; v).
\]

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

\[
    CE_{f(u,v)}(f(,u)) = H(f(,v)) + KL(f(,v), f(,u))
\]

where \( CE \) is the cross entropy, \( KL \) is the Kullback-Leibler divergence and \( H(.) \) is the Shannon entropy. Since \( H(.) \) is constant, [12] equates the cross entropy with the Kullback-Leibler divergence.

2.2 Combinatorial Optimization via Cross Entropy

We may also use the CE method for Combinatorial Optimization Problems (COPs). The main idea is to first turn each COP into a rare-event estimation problem, the so-called associated stochastic problem(ASP) and then perform the cross entropy method.
We usually wish to maximize some performance function $S(\mathbf{x})$ over all states $\mathbf{x}$ in data set $N$. Denoting the maximum by $\gamma^*$, we have

$$\gamma^* = \max_{\mathbf{x} \in N} S(\mathbf{x})$$  \hspace{1cm} (6)

Thus, by defining a family of pdfs $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ on the data set $N$, we follow [12] to associate with (6) the following estimation problem

$$l(\gamma) = \mathbf{P}_u(S(\mathbf{X}) \geq \gamma) = \mathbf{E}_u I_{\{S(\mathbf{x}) \geq \gamma\}}$$ \hspace{1cm} (7)

where $\mathbf{X}$ is a random vector with pdf $f(\cdot; \mathbf{u}), \mathbf{u} \in \mathcal{V}$. To estimate $l$ for a certain $\gamma$ close to $\gamma^*$, we can make adaptive changes to the probability density function according to the Kullback-Leibler cross-entropy. Thus we create a sequence $f(\cdot; \mathbf{u}), f(\cdot; \mathbf{v}_1), f(\cdot; \mathbf{v}_2), \ldots$ of pdfs that are optimized in the direction of the optimal density. By associating the underlying combinatorial optimization problem, that is to optimize $S$, with a rare-event probability of $\mathbf{P}_u(S(\mathbf{X}) \geq \gamma)$, we can obtain an estimate of the “reference parameter vector” $\mathbf{v}^*$ via the CE algorithm.

Similar to the algorithm described in the previous subsection, the process is as follows:

- For a fixed $\mathbf{v}_{t-1}$, we choose the $(1-\varrho)$-quantile of $S(\mathbf{X})$, $\gamma_t$, which satisfies

  $$\mathbf{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geq \gamma_t) \geq \varrho,$$ \hspace{1cm} (8)

  The estimator $\hat{\gamma}_t$ can be obtained by drawing a set of random samples $\mathbf{X}_1, \ldots, \mathbf{X}_N$ and evaluating the sample $(1-\varrho)$-quantile of performances

  $$\hat{\gamma}_t = S_{[(1-\varrho)N]}$$ \hspace{1cm} (9)

- For fixed $\hat{\gamma}_t$ and $\mathbf{v}_{t-1}$, we optimize $\mathbf{v}_t$ by

  $$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \hat{\gamma}_t\}} \ln f(\mathbf{x}_i; \mathbf{v}).$$ \hspace{1cm} (10)

  It is important to observe that in the ASP, given fixed $\hat{\gamma}_t$ and $\mathbf{v}_{t-1}$, (10) is different from (4) in that (10) does not contain the likelihood ratio term $W$. This is because the initial reference parameter vector $\mathbf{u}$ is arbitrary, however using this parameter is essential when solving a rare event problem.

### 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 mapping is Kohonen’s self-organizing map (SOM) [11]. The Generative Topographic Mapping (GTM) [5] is a mixture of experts model which treats the data as having been generated by a set of latent points where the mapping is non-linear. One of us [8] has derived an alternative topology preserving model, called the Topographic Products of Experts (ToPoE), based on the analysis of products of experts [9], which is closely related to the generative topographic mapping.

We follow [5, 8] to create a latent space of points $\mathbf{x}_1, \ldots, \mathbf{x}_K$ which lie equidistantly on a line or at the corners of a grid. To allow non-linear modeling, we define a set of $M$ basis functions, $\phi_1(), \ldots, \phi_M()$, with centres $\mu_j$ in latent space. Thus we have a matrix $\Phi$ where $\phi_{kj} = \phi_j(\mathbf{x}_k)$, each row of which is the response of the basis functions to one latent point, or, alternatively each column of which is the response of one of the basis functions to the set of latent points. Typically, the basis function is a squared exponential. These latent points are then mapped to a set of points $\mathbf{m}_1, \ldots, \mathbf{m}_K$ in data space where $\mathbf{m}_j = (\Phi_j \mathbf{W})^T$, through a set of weights, $\mathbf{W}$. The matrix $\mathbf{W}$ is $M \times D$ and is the sole parameter which we change during training. We have

$$\mathbf{m}_k = \sum_{j=1}^{M} \mathbf{w}_j \phi_j(\mathbf{x}_k)$$ \hspace{1cm} (11)

$$= \sum_{j=1}^{M} \mathbf{w}_j \exp(-\beta ||\mu_j - \mathbf{x}_k||^2), \forall k \in \{1, \ldots, K\},$$

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.

In the GTM [5], each latent point is defined as an individual expert and the whole mapping is then defined as a mixture of experts. The parameters of this model are updated by the Expectation-Maximization algorithm. In the ToPoE [8], the experts are combined as a product of experts each with a specific responsibility for every data point and the mapping is updated using gradient ascent on the negative log likelihood function. In the next section, we retain the underlying structure of both these algorithms (latent points being mapped through basis functions and then linearly combined) but adjust the parameters to fit the data using the cross entropy method.

#### 3.1 Applying Cross Entropy to Topology Preserving Mappings

Let us base the objective function on the minimisation of the error between the data points, $\mathbf{t}_i$ and the closest
projected latent point, $m_{bs}$. The algorithm is summarized as:

1. Randomly select a data point, $t$.

2. Find the closest prototype, say $m_{bs}$, to $t$.

3. Generate $T$ samples from the Gaussian distribution, $\mathcal{N}(m_{bs}, \beta_{bs}^2 I)$. Call the samples, $y_{ks,1}, \ldots, y_{ks,T}$. We note that we are using $m_1, m_2, \ldots, m_K$ to perform two conceptually separate functions, as prototypes or means to which the data will be quantized and as centres of Gaussian distributions from which samples will be drawn.

4. Evaluate the samples using $S(y) = \exp(-\gamma ||y - t||^2)$ 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

\[
\Delta_w = \left( \frac{1}{r} \sum_{j=T-r}^T y_{ks,p(j)} - m_{bs} \right) \phi(x_{ks}) \\
\Delta_\beta = \frac{1}{r} \sum_{j=T-r}^T (y_{ks,p(j)} - m_{bs})(y_{ks,p(j)} - m_{bs})^T \\
\beta_{bs}^2 = \beta_{bs}^2 + \eta_0 \Delta_\beta
\]

where $\eta, \eta_0$ are the learning rates with typically $\eta = 10\eta_0$.

7. Update the prototypes’ positions using (11).

Figure 1 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, $(x_1, x_2)$, from the function $x_2 = x_1 + 1.25 \sin(x_1) + \rho$, where $\rho$ 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 and we have linked them in the order in which they appear in latent space, $1 \rightarrow 2 \rightarrow 3 \rightarrow \ldots \rightarrow 20$. We clearly see that the one dimensional nature of the data has been identified and that some measure of topology preservation has been achieved: nearby points are closest to nearby latent points’ projections.

Figure 1: The data are shown by ‘+’s and the latent points’ projections are shown by ‘*’s. The latent points’ projections have been linked in the order in which they appear in latent space.

### 4 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 [3] 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 [2, 10] 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 receives 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, $X = (x_1, \ldots, x_K)$ that have the structure of lying equidistantly on a line or on a grid. These latent points are nonlinearly mapped to points, $(m_1, \ldots, m_K)$ in the input space through a set of basis function, which forms a set of reference vectors, $m_k = W \phi(x_i)$. 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 $t_n^{latent}$, we have

$$
t_n^{latent} = \sum_{i=1}^{K} r_{ni} x_i \quad (12)
$$

$$
r_{ni} = \frac{\exp(-\gamma d_{ni}^2)}{\sum_k \exp(-\gamma d_{nk}^2)} \quad (13)
$$

where $r_{ni}$ is responsibility of the $i^{th}$ latent point for the $n^{th}$ data point and $d_{pq} = ||t_p - m_q||$, 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 $21 \times 21$ grid of latent points being passed through a $5 \times 5$ 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, $R = \sum_{n=1}^{N} r_n$, and then form a responsibility grid using $R$ for each latent point. Since the latent points are fixed in (12), 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 2. 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 cannot identify different clusters directly. We also find that the higher-level the layer is, the hotter the areas corresponding to different clusters 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.

5 Conclusion

In this paper, we have investigated topology preserving mapping with the cross-entropy method. We use stochastic units drawn from a Gaussian distribution to sample the parameters of the performance function. With the CE algorithm, we make adaptive changes to the probability density function of the parameters according to the Kullback-Leibler cross-entropy. Topology preserving mapping can be solved by gradient-based algorithms or the EM algorithm, however, due to the constraints on the parameters, and the complicated nature of the likelihood function, direct optimization of the likelihood function is not a simple task. We consider the cross entropy method converges to the global optimum with high accuracy. We developed a multilayer topology preserving mapping with the cross entropy method, which allowed us to demonstrate deep architecture in unsupervised data exploratation. The results have been expressed in a particular way which allows us to easily evaluate how well the algorithm works. We have shown that the deep architecture algorithm has identified the clusters more accurately.

References:


Figure 2: 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 1st layer, the 2nd layer, the 3rd layer and the 4th layer.


