

A New Approach to Cluster-Weighted Modeling

Danil V. Prokhorov, Lee A. Feldkamp, and Timothy M. Feldkamp
Ford Research Laboratory
Ford Motor Company
Dearborn, MI, USA

Abstract

We discuss a novel approach to joint density estimation called cluster-weighted modeling (CWM). The base approach was originally proposed by Neil Gershenfeld¹. We describe two innovations to the base CWM. Among these, the first enables the CWM to work with continuous streams of data. The second addresses the commonplace problem of local minima which may be encountered during CWM parameter adjustment process. Our approach to mitigate this problem is quite elaborate, but it represents a principled way of improving efficacy of the parameter adjustment process. We illustrate CWM and our performance enhancements with an example.

1 Introduction

Cluster-weighted modeling (CWM) was introduced by Neil Gershenfeld [1] as an elegant approach to approximating joint density functions of general mappings of the form $y = f(\vec{x})$, where \vec{x} is a vector of arguments. Mappings are assumed to be given as a set of pairs (y, \vec{x}) .

The CWM approach is particularly appealing because of its simplicity, generality, and flexibility. Even in cases in which a feedforward layered network (multilayer perceptron) might be preferred, it is useful to have CWM to provide an independent “second opinion” on the nature of the training problem. In such cases, one does not necessarily make full use of the density estimation carried out by CWM, but only the expected value of the output for a given input vector. However, the density information can always be used, for example, to detect cases in which a mere regression fit to the data is not adequate, such as when the output can have two or more values for the same input vector. It is also possible to assess whether the model is based on sufficient training data in a particular portion of input space to justify estimating an output.

Section 2 provides two formulations of CWM, the second one having been developed in this work. We also discuss

¹It is a pleasure to acknowledge Neil Gershenfeld for introducing one of the authors (LAF) to a pre-publication description of CWM and Gintaras Puskorius for early collaboration on the method.

the problem of local minima associated with optimization process required for parameter adjustment in either of the formulations. Section 3 summarizes our improved implementation of CWM followed by an example in Section 4. We conclude with some final remarks and our recommendations on the applicability of CWM (Section 5).

2 Description of CWM

2.1 Batch formulation using EM and SVD

In this section we describe the base CWM formulation. We follow closely [1], pp. 178-183, for the reader’s convenience. Our goal is to approximate the density in the joint input-output space. Let us have a set of N observations $\{y_n, \vec{x}_n\}_{n=1}^N$, where \vec{x}_n is a vector of known inputs, and y_n is the measured output (scalar here for simplicity; generalization to a vector of outputs is straightforward). If we were given the joint density $p(y, \vec{x})$, then we could find any quantity of interest derived from it, e.g., a conditional forecast $\langle y | \vec{x} \rangle$. Assuming that the joint density $p(y, \vec{x})$ can be approximated by clusters arbitrarily well, we can write

$$\begin{aligned} p(y, \vec{x}) &= \sum_m p(y, \vec{x}, c_m) \\ &= \sum_m p(y | \vec{x}, c_m) p(\vec{x} | c_m) p(c_m), \end{aligned} \quad (1)$$

where $p(c_m)$ is the weight (also called *prior*) of the cluster m , $p(\vec{x} | c_m)$ is a probability measure reflecting the influence of the cluster m in the input space, and $p(y | \vec{x}, c_m)$ is a similar measure for the output space. The input probability measure $p(\vec{x} | c_m)$ in (1) can be expressed via D -dimensional separable Gaussians:

$$p(\vec{x} | c_m) = \prod_{d=1}^D \frac{\exp\left(-\frac{(x_d - \mu_{m,d})^2}{2\sigma_{m,d}^2}\right)}{\sqrt{2\pi\sigma_{m,d}^2}} \quad (2)$$

A straightforward alternative is to use a single Gaussian with the full covariance matrix, but it is not used here due to increased storage and computational requirements.

The output probability measure $p(y | \vec{x}, c_m)$ is also

a Gaussian

$$p(y|\vec{x}, c_m) = \frac{\exp\left(-\frac{(y-f(\vec{x}, \vec{w}_m))^2}{2\sigma_{m,y}^2}\right)}{\sqrt{2\pi\sigma_{m,y}^2}} \quad (3)$$

but it has a different set of parameters. Its variance is $\sigma_{m,y}^2$, while the mean is expressed as a function that depends on both \vec{x} and coefficients \vec{w}_m .

Consider the conditional forecast

$$\langle y|\vec{x} \rangle = \frac{\sum_m f(\vec{x}, \vec{w}_m)p(\vec{x}|c_m)p(c_m)}{\sum_m p(\vec{x}|c_m)p(c_m)} \quad (4)$$

Here Gaussians are used to control the interpolation among the functions $f(\vec{x}, \vec{w}_m)$, rather than to serve directly as a basis for function approximation (as, e.g., in radial basis function (RBF) networks). This means that f can be chosen to reflect prior knowledge of the local relationship between \vec{x} and y . We can also trade off the complexity of the whole system with that of $f(\vec{x}, \vec{w}_m)$. Using even simple local models (e.g., linear) does not preclude us from handling arbitrary combinations of smoothness and discontinuity globally, because there are no constraints on the positions of clusters relative to each other. In addition, the use of Gaussian clusters does not mean that we assume a Gaussian error model. Multiple output clusters (3) can be associated with one input value to capture multi-modal or other types of distributions.

To estimate the parameters, we adopt the *expectation-maximization* (EM) formalism [2]. First, we calculate the posteriors

$$\begin{aligned} p(c_m|y, \vec{x}) &= \frac{p(y, \vec{x}, c_m)}{p(y, \vec{x})} \\ &= \frac{p(\vec{x}|c_m)p(y|\vec{x}, c_m)p(c_m)}{\sum_m p(\vec{x}|c_m)p(c_m)p(y|\vec{x}, c_m)} \end{aligned} \quad (5)$$

which are then used to update the cluster weights (priors)

$$p(c_m) = \frac{1}{N} \sum_n p(c_m|y_n, \vec{x}_n) \quad (6)$$

Similarly, utilizing the notation

$$\langle z \rangle_m \equiv \frac{\sum_n p(c_m|y_n, \vec{x}_n) z_n}{\sum_n p(c_m|y_n, \vec{x}_n)} \quad (7)$$

the new input means $\vec{\mu}^{new}$ and input variances $\sigma_{m,d}^{2,new}$ are computed:

$$\vec{\mu}_m^{new} = \langle \vec{x} \rangle_m \quad (8)$$

$$\sigma_{m,d}^{2,new} = \langle (x_d - \mu_{m,d})^2 \rangle_m \quad (9)$$

The output model parameters are obtained by choosing values that maximize the cluster-weighted log-likelihood:

$$\begin{aligned} 0 &= \frac{\partial}{\partial \vec{w}_m} \log \prod_n p(y_n, \vec{x}_n) \\ &= \left\langle [y - f(\vec{x}, \vec{w}_m)] \frac{\partial f(\vec{x}, \vec{w}_m)}{\partial \vec{w}_m} \right\rangle_m \end{aligned} \quad (10)$$

For linear output models, we propose to parameterize the output function relative to the input center:

$$f(\vec{x}, \vec{w}_m) = \mu_{m,y} + \sum_{d=1}^D w_{m,d} (x_d - \mu_{m,d}) \quad (11)$$

This enables easy comparisons to be made to the case of a simpler output model in which the linear variation is not included. We can solve for the optimal parameters \vec{w}_m^{new} of the cluster:

$$\begin{aligned} \underbrace{\langle y(x_j - \mu_{m,j}) \rangle_m}_{a_j} &= \sum_{i=1}^D w_{m,i} \underbrace{\langle (x_j - \mu_{m,j})(x_i - \mu_{m,i}) \rangle_m}_{B_{ji}} \\ &\Rightarrow \vec{w}_m^{new} = \mathbf{B}^{-1} \cdot \vec{a} \end{aligned} \quad (12)$$

The mean of the output clusters may then be updated using

$$\mu_{m,y}^{new} = \langle y_n \rangle_m \quad (13)$$

and the output variance vector by

$$\sigma_{m,y}^{2,new} = \langle (f(\vec{x}_n, \vec{w}_m) - \mu_{m,y}^{new})^2 \rangle_m \quad (14)$$

We begin with an initial configuration of M clusters with the same (or different) structures as the output model. (The value of M and the output model structure are problem dependent.) Starting from the initial configuration and iterating the sequence of evaluating the probabilities (2),(3), computing the expectation (5), then finding the most likely parameters for the given distribution via updates (6),(8),(9),(12),(13), and (14), results in the maximum likelihood distribution with parameters closely approximating the original joint density.

Replacing the potentially costly matrix inversion in (12) with recursive least squares (RLS) was proposed in [3]. Here we use the RLS to compute all cluster parameters, not just those of the output models. As a result, we obtain a new capability missing from previous implementations of CWM: the ability to process continuous streams of data.

2.2 Pattern-by-Pattern Formulation

Our innovations to the basic (batch) algorithm for CWM described in the previous section amount to recursive computation of not only model parameters \vec{w} but also cluster

weights (priors), means and variances. We utilize the RLS algorithm assuming that, in the limit of a small enough learning rate and finite data, we would closely approximate a solution of the batch CWM algorithm.

We summarize the weighted RLS recursion below:

$$\begin{aligned}\vec{G} &= \mathbf{P} \vec{h} (R + \vec{h}^T \mathbf{P} \vec{h})^{-1} \\ \vec{z}^{new} &= \vec{z}^{old} + \vec{G} \xi \\ \mathbf{P}^{new} &= \mathbf{P}^{old} - \vec{G} \vec{h}^T \mathbf{P} + \mathbf{Q}\end{aligned}\quad (15)$$

Here \vec{z} is a vector assembled from parameters chosen for updates, R is the inverse of the weight applied to error ξ , \mathbf{P} is an approximate error covariance matrix (just a number in the scalar case), and a positive definite (often diagonal) matrix \mathbf{Q} reflects additive deweighting of data.

We begin with updating the model parameters assuming all $m = 1, 2, \dots, M$ values of $p(c_m|y_n, \vec{x}_n)$ from the previous iteration of the algorithm. For the n -th data point, the m -th cluster and the model structure of (11), the RLS error is

$$\xi_m = \sqrt{p(c_m|y_n, \vec{x}_n)}(y_n - f(\vec{x}_n, \vec{w}_m)) \quad (16)$$

The derivative of f with respect to the i -th component of \vec{w}_m is

$$h_i = \sqrt{p(c_m|y_n, \vec{x}_n)}(x_{n,i} - \mu_{m,i}) \quad (17)$$

for $i = 1, 2, \dots, D$ and

$$h_0 = \sqrt{p(c_m|y_n, \vec{x}_n)} \quad (18)$$

for the bias $\mu_{m,y}$. The expressions above can be understood by examining (10). The RLS algorithm is a zero mean estimator; hence one can use the term $y_n - f(\vec{x}_n, \vec{w}_m)$ as the instantaneous RLS error. For the chosen (linear) output model the term $\frac{\partial f(\vec{x}_n, \vec{w}_m)}{\partial \vec{w}_m}$ is equal to $x_{n,i} - \mu_{m,i}$ for the i -th component of the parameter vector \vec{w}_m . Finally, we must multiply all components of ξ and h by $\sqrt{p(c_m|y_n, \vec{x}_n)}$ in order to ensure appropriate weighting of each data point and maintain internal consistency of the RLS.

The update for the prior $p(c_m)$ is based on $\xi_m = p(c_m|y_n, \vec{x}_n) - p(c_m)$ and $h = 1$. Next, the input means and variances are updated based on their respective errors and derivatives $h = h_0$ of (18):

$$\xi_m = \sqrt{p(c_m|y_n, \vec{x}_n)}(x_{n,i} - \mu_{m,i}) \quad (19)$$

for input means and

$$\xi_m = \sqrt{p(c_m|y_n, \vec{x}_n)}((x_{n,i} - \mu_{m,i})^2 + \sigma_{min}^2 - \sigma_{m,i}^2) \quad (20)$$

for input variances using new values of $\mu_{m,i}$ updated via (19). Finally, the output variance $\sigma_{m,y}^2$ is recomputed with $h = h_0$ of (18) based on

$$\xi_m = \sqrt{p(c_m|y_n, \vec{x}_n)}((y_n - f(\vec{x}_n, \vec{w}_m))^2 + \sigma_{y,min}^2 - \sigma_{m,y}^2) \quad (21)$$

where $f(\vec{x}_n, \vec{w}_m)$ takes into account values of \vec{w}_m just updated via (16) as well as the following bias weight correction: $\mu_{m,y}^{new} = \mu_{m,y} + \sum_{d=1}^D w_{m,d}(\mu_{m,i}^{new} - \mu_{m,i})$. We introduce the parameters σ_{min}^2 and $\sigma_{y,min}^2$ in order to prevent a cluster from collapsing to zero width in any input or output dimension. (Their choice should be conditioned upon the data resolution.) It should be noted that *separate* RLS routines are required for proper handling of the updates above.

2.3 Local minima

CWM is a numerical method of maximizing the joint density. Its efficacy depends critically on its ability not to fall prey to the well known (and fundamental) problem of local minima. (Though dealing with *maximization* of the joint density (log-likelihood), we will use the much more popular term “local minimum”.) We find that both formulations, the batch and pattern-by-pattern, are prone to get stuck in undesirable local minima *regardless* of the initial values of the cluster parameters and approaches taken to update them. To illustrate this, we consider a simple “four-clump” problem, with two pairs of unequally weighted data clumps (Figure 1). The total number of data points is 2000. Each clump is generated from a Gaussian distribution with the standard deviation of 0.2 around its center. The true probability of data in the two dense clumps is 0.475 per clump, whereas the probability of data in two sparse clumps is 0.025 per clump (i.e., the dense clumps contain many more points).

A correct positioning of four clusters is shown in Figure 1. The cluster weights are $p(c_0) = 0.033$, $p(c_1) = 0.454$, $p(c_2) = 0.489$, $p(c_3) = 0.024$. We observed three types of falling into a local minimum of the joint density, or *failure modes*. Figures 2 and 3 illustrate two of these failure modes. In each case one cluster is trying to do the job of two clusters, with another cluster being either practically irrelevant (its weight $p(c) \approx 0$) or forced to share some density with its neighbor (two clusters located near each other on the same clump), as shown in Figure 3.

The stretched cluster #0 (Figure 2) captures two clumps (its weight $p(c_0)$ becomes 0.055), which results in a very elongated one-standard-deviation bar for the x dimension. Cluster #3 gets extinguished ($p(c_3) \approx 0$).

Cluster #1 (Figure 3) describes its clump well ($p(c_1) = 0.492$), whereas clusters #0 and #3 are positioned near each other thereby sharing density for the same clump

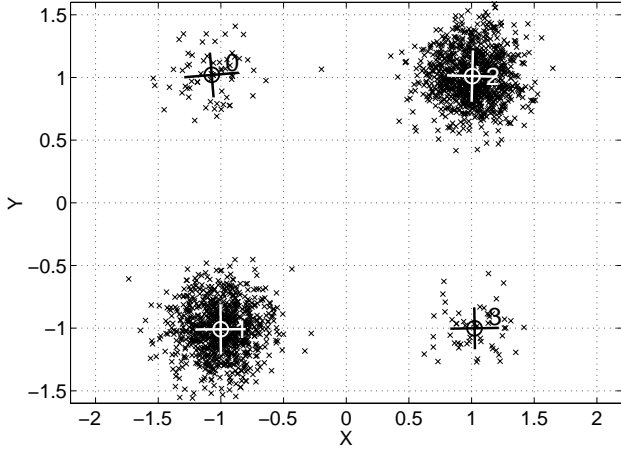


Figure 1: Illustration of a correct positioning of clusters to maximize the likelihood of a model of the joint density of the four clumps of data points (2000 points total, shown as crosses). The four clusters are positioned correctly (shown as circles), with their standard deviations $\sigma_{m,x}$ and σ_y close to 0.2 (shown for each cluster as two perpendicular bars).

with center at $(-1, -1)$ ($p(c_0) = 0.230$ and $p(c_3) = 0.223$).

We can make the following conclusions about the CWM algorithm: 1) clusters whose values of $p(c)$ become very small have little chance for further participation in the density modeling process, since they are capable of modeling only a tiny fraction of the overall density (e.g., a single point which happens to coincide with the cluster's location); 2) clusters located in the vicinity of each other and sharing parts of the density which could have been better approximated by a single cluster do not have "incentive" to separate, even if other parts of the density are not approximated well. In the next section we describe our improved implementation of the CWM algorithm.

3 Synopsis of our implementation

We provide here a summary of our improved implementation. We begin by initializing all M clusters at initial locations (chosen randomly or based on problem specifics) within the region of the (y, \vec{x}) space of interest; other parameters of the algorithm, minimum variances for all coordinates (inputs and the output), the matrices P , R and Q , are also initialized ($P(0) = 10^5 I$, $R = 1000 I$ and $Q = 0.001 I$ are typical values).

For each data point (y_n, \vec{x}_n) we compute $p(\vec{x}_n | c_m)$ of (2) and $p(y_n | \vec{x}_n, c_m)$ of (3) for all clusters, $m = 1, 2, \dots, M$. We then compute the posteriors $p(c_m | y_n, \vec{x}_n)$ of (5) for every cluster m , followed by updates of \vec{w}_m , $p(c_m)$, $\mu_{m,i}$, $\sigma_{m,i}^2$ and $\sigma_{m,y}^2$ (as described in Section 2.2).

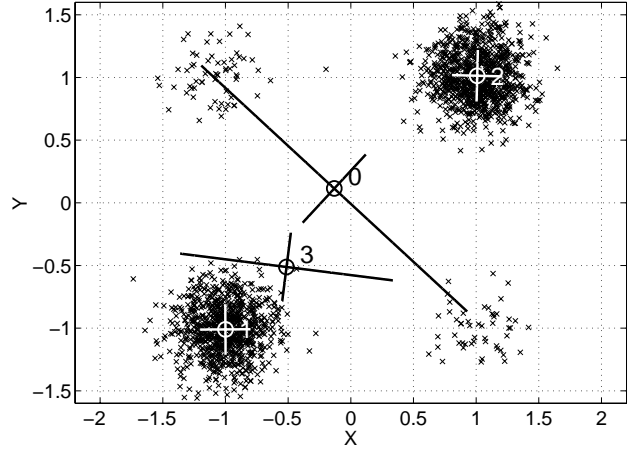


Figure 2: Illustration of a second failure mode in an attempt to maximize the likelihood of a model of the joint density of the four clumps of data points (see Figure 1 for details on data characteristics).

We can detect nearly redundant clusters via the following scheme. For each data point (y_n, \vec{x}_n) and cluster m we compute

$$\rho_m = p(y_n, \vec{x}_n) \sum_{k \neq m} \frac{p(c_k | y_n, \vec{x}_n) p'_m(c_k)}{p(c_k)} \quad (22)$$

where $p'_m(c_k)$ is recursively updated by a separate RLS structure with

$$\xi_m = \frac{p(c_k | y_n, \vec{x}_n)}{1 - p(c_m | y_n, \vec{x}_n)} - p'_m(c_k) \quad (23)$$

and $h = 1$ for all $k \neq m$, as described in Section 2.2 (see equations for recursive updates of $p(c_m)$). The quantity $p'_m(c_k)$ may be interpreted as an estimate of the fraction of the density represented by cluster m that cluster k would represent if its weight were increased to reflect the absence of cluster m . For each cluster we can compare ρ_m and $p(y_n, \vec{x}_n)$ (or some averages thereof). Clusters with $\rho_m \approx p(y_n, \vec{x}_n)$ are deemed suitable for reinitialization.

Applying the approach based on (22)-(23), we determine whether there are poorly utilized (underutilized) clusters. If so, then we identify non-negligible overutilized clusters, i.e., those for which one (or both) of the following ratios is larger than a chosen threshold (no less than the unity):

$$|\lambda_m| / \sqrt{\frac{6}{N_1 p(c_m)}}, \quad \kappa_m / \sqrt{\frac{24}{N_1 p(c_m)}} \quad (24)$$

Here a window of length N_1 is multiplied by $p(c_m)$ instead of N as we judge the significance of estimates of the skewness λ_m and the kurtosis κ_m for each cluster m (both λ_m

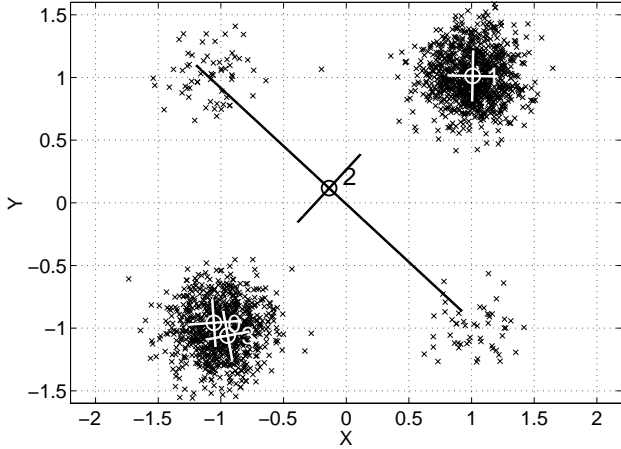


Figure 3: Illustration of a third failure mode in an attempt to maximize the likelihood of a model of the joint density of the four clumps of data points (see Figure 1 for details on data characteristics).

and κ_m are computed separately for each input/output dimension and recursively, in a way similar to that for the variance in Section 2.2)². These are computed for all clusters deemed essential to the modeling process (i.e., unavailable for reinitialization). If no poorly utilized cluster is present, then we advance to the next data point. We monitor a running sum of the values $\log(p(y_n, \vec{x}_n))$ (or in a window) and stop the training when it becomes large enough.

For all overutilized clusters, we suggest splits into cluster pairs based on fitting statistical quantities (moments) of the original distribution. In order to compute parameters for the cluster pair, we need to obtain a system of equations that ties all the relevant cluster parameters together. Moments of the original cluster (indexed as 0) can be approximated through a function of parameters of the cluster pair (indexed as 1 and 2) as follows:

$$\hat{m}_0^i = \frac{p(c_1)}{\sqrt{(2\pi\sigma_1^2)}} \int_{-\infty}^{\infty} z^i \exp\left(-\frac{(z-d_1)^2}{2\sigma_1^2}\right) dz + \frac{p(c_2)}{\sqrt{(2\pi\sigma_2^2)}} \int_{-\infty}^{\infty} z^i \exp\left(-\frac{(z-d_2)^2}{2\sigma_2^2}\right) dz \quad (25)$$

where σ_1^2 and σ_2^2 are the corresponding cluster variances for a given coordinate (its index is omitted to avoid cluttering the notation), d_1 and d_2 are (one-dimensional) distances between the cluster's center and the center of the cluster 0 for the same coordinate, and $i = 1, 2, 3, 4$. The mean (center)

²We note that in [4], p. 612, the expressions for the standard deviations of the skewness and kurtosis estimates differ from those of the first edition (the denominators in (24)). A numerical experiment suggests that those of the first edition are correct.

of cluster 0 is denoted as m_0^1 , and its variance is denoted as m_0^2 . The two other quantities, m_0^3 and m_0^4 , are related to the skewness and kurtosis, respectively; $m_0^3 = \sigma^3\lambda_0$ and $m_0^4 = \sigma^4(\kappa_0 + 3)$. All quantities \hat{m}_0^i denote estimates of the corresponding quantities m_0^i . By substituting $z' = z - d$ and $\hat{m}_0^i = m_0^i$, we can solve analytically the integrals in (25) for all i , obtaining an underdetermined system of non-linear equations. This system should be solved with respect to $p(c_1)$, d_1 , $p(c_2)$, d_2 , σ_1 and σ_2 . By assuming $p(c_1) = p$, $p(c_2) = 1 - p$, $p \in (0, 1)$, we sweep through the values of d_2 from a reasonable range so as to match $m_0^1 = 0$, m_0^2 and m_0^3 exactly while closely approximating m_0^4 . For all coordinates for which non-Gaussianity is statistically significant, there may be several overlapping solutions which differ by the errors of fitting m_0^4 . We evaluate the sum of absolute values of errors for each solution and choose one with the smallest cumulative error.

The approach just described emphasizes the need to fit the skewness of a local distribution perfectly. For distributions for which the kurtosis is a dominant factor (e.g., see cluster #0 in Figure 2: $\lambda_0 \approx 0$, $\kappa_{0,x} \approx -2$), no sweep through d_2 is necessary, and the solution can be obtained in closed form.

The efficacy of each proposed split or recombination is estimated based on the Kullback-Leibler (KL) distance [5]. Assuming a grid fine enough, KL can be approximated as

$$KL \approx \sum_n h(y_n, \vec{x}_n) \log(h(y_n, \vec{x}_n)/g(y_n, \vec{x}_n)) \Delta_n \quad (26)$$

where $h(y_n, \vec{x}_n)$ is the probability density associated with the cluster pair, $g(y_n, \vec{x}_n)$ is the probability density associated with the original cluster, n is the index of the grid point, and Δ_n is the volume at (\vec{x}_n, y_n) (i.e., a product of step sizes for all coordinates). Out of all overutilized clusters, we choose one (or a few) which has the maximum efficacy and split them, reinitializing all poorly utilized clusters accordingly. Then we proceed to the next data point.

Testing CWM is accomplished after the CWM parameters have converged. For each data point from the testing set (y_n, \vec{x}_n) we compute $p(\vec{x}_n|c_m)$ of (2) and $p(y_n|\vec{x}_n, c_m)$ of (3) for all clusters, $m = 1, 2, \dots, M$, then compute $p(y, \vec{x})$ of (1) using known values of $p(c_m)$. For a block of points one can also evaluate the average output $\langle y|\vec{x} \rangle$ of (4).

4 Example

We describe here the simplest and most illustrative among examples of problems studied thus far with CWM. We compare our pattern-by-pattern algorithm with reinitialization on the four-clump problem with the batch CWM with and without reinitialization of clusters done with the following simple approach. The reinitialization is done at random for

all clusters whose $p(c_m)$ become too small, so each of those clusters would disappear from one place of the (x, y) space and reappear in another, with its weight set to a reasonable value.

For each run, we begin with four clusters initialized at random in the square with side of 2 centered at zero. We obtain the global minimum of the negative log-likelihood (1080 or ≈ 0.54 per data point) in less than 50 epochs of training in each of 100 independent runs with our approach. This is much better than results of the base (batch) CWM (68 successful runs of 100 with the simple reinitialization and 40/100 without it).

The simple reinitialization strategy is universal and usually effective *assuming* that we can apply it long enough. In practice this reinitialization strategy can be very inefficient, exponentially degrading with increase of the dimensionality of the data. Furthermore, this strategy is worthless in situations when two (or more) clusters share the same portion(s) of the overall density (see Figure 3).

4.1 Summary of Experimental Observations

We would like to discuss a few behavioral differences between the base CWM algorithm and the pattern-by-pattern algorithm with or without reinitializations of clusters.

1. We observed that aggressive training frequently results in many clusters being extinguished. To mitigate this, the learning rate of statistical quantities (particularly, that of prior $p(c_m)$) should be smaller than that of parameters \vec{w} .
2. For all fixed-data-set problems studied so far, we were able to obtain good solutions with the base CWM algorithm, sometimes requiring multiple training attempts. We suspect, however, that training problems characterized by noticeably disjoint distributions of data points (such as in the four-clump problem) will show our approach to have a significant advantage. Furthermore, our reinitialization method can easily be adapted for use with the base CWM algorithm to enable acceptable solutions without overall restarts.

5 Conclusions

In various limits, CWM can be reduced to familiar forms. 1) If the output density and model is eliminated, CWM becomes standard (input) density modeling (a mixture model [2]). 2) If the output model is reduced from a linear model to merely a constant, the cluster-weighted model is functionally equivalent to a form of RBF network; the EM training mechanics used by CWM are a useful alternative to the gradient-based methods commonly employed when basis function centers and/or variances are to be learned in addition to weights for the links connecting basis functions to the outputs.

The CWM is an efficient way of modeling static (lacking temporal ordering) linear or nonlinear mappings, preferably low-dimensional. It can be most useful as a replacement for look-up tables still pervasive in many automotive applications. (It is expected that the CWM could be an increasingly more appealing alternative to conventional look-up tables when the dimensionality of problems exceeds two.) The final representation (after training) in CWM is usually quite transparent, much more so than, e.g., representations of multilayer perceptrons, with parameters which are well understood statistically. In contrast to RBF networks, the CWM may achieve superior interpolation with many fewer clusters because each cluster is equipped with its own output model. It is particularly powerful in verifying whether a given mapping possesses multivaluedness. This may be very useful in checking experimental data for consistency.

All versions of CWM are prone to the problem of local minima. We have described a successful attempt to mitigate this by employing our sophisticated cluster reinitialization algorithm. Though generally efficacious, the algorithm may be time-consuming for multidimensional problems. Its efficacy is also vulnerable to inaccurate estimates of the skewness and kurtosis. That is why using too many clusters for too few data points (i.e., less than 50 points per cluster) may cause reinitialization of the “wrong” clusters, thereby delaying the convergence process. Overall, the choice of all CWM parameters necessary to initiate training does not appear to be an exquisite process. The CWM approach is sufficiently flexible to be quickly adapted for many data modeling applications utilizing an acceptable amount of adjustment and problem specific intuition.

References

- [1] N. A. Gershenfeld. *Nature of Mathematical Modeling*, MIT Press, 1998.
- [2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995. The discussion on mixture models is of particular relevance here.
- [3] G. V. Puskorius and L. A. Feldkamp, “Local model updates via recursive least squares for cluster-weighted modeling.” Presentation at *the International Joint Conference on Neural Networks (IJCNN)*, Washington DC, July 1999.
- [4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge: Cambridge University Press, 2nd Edition, 1994.
- [5] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice-Hall, 1999.