Chapter 7

Optimising Topographic Transformations

7.1 Introduction

It has been previously noted that the storage and computational requirements of the Sammon mapping grow in the order of the square of the number of data points to be mapped. Any neural network implementation will also suffer from this undesirable $O(N^2)$ scaling behaviour.

For an efficient implementation of Sammon's algorithm, all the inter-point distances in the data space are calculated *a priori* and stored in a $(N \times N)$ matrix. For a thousand data points, and only single precision arithmetic, this would require 4 megabytes (MB) of storage. Even a machine with 64MB of dedicated memory could only store a 4000-pattern distance matrix.[1] For such large numbers of patterns, the inter-point distances would need to be calculated, repeatedly, on-the-fly.

This in turn will impact on the already considerable computational demands. Each calculation of the STRESS measure necessitates a loop of $N(N-1)/2$ distance calculations. Calculation of the gradients (for minimisation routines) requires a similar number of cycles, with additional vector operations.

There have been two characteristic approaches to alleviate the difficulties of training STRESS-based mappings. Firstly, various optimisation schemes have been considered for standard mapping procedures which will be covered in the next section. In Section 7.3, a new, and highly effective, algorithm is proposed for the training of topographic models whose outputs are linear in their train-able parameters, such as radial basis function networks.

A popular second approach has been the development of alternative heuristic strategies for structure preservation, based on Sammon's algorithm or MDS. These schemes are reviewed, with comment, in Section 7.4.

[1] Of course, virtual memory systems permit much larger matrices but the time spent retrieving the data from disc is prohibitive and defeats the purpose of storing it in the first place.

7.2 Optimisation Schemes for the Sammon Mapping

Sammon's original algorithm adopted a Newton-Raphson-based steepest-gradient method for minimising the STRESS, which required the setting of a "magic factor" parameter, "determined empirically to be 0.3 or 0.4". This was a 'batch' algorithm which calculated the derivatives over the entire pattern set before adjusting any weights.

Mao and Jain [1995] used gradient-descent for their MLP approach, but updated the weights on a pattern-by-pattern basis. This is an attempt to alleviate the computational demands associated with large data sets. See section 3.5.1 for earlier comment on this particular approach.

In the NEUROSCALE model, in addition to the above two schemes, both a conjugate-gradient and the quasi-Newton BFGS technique [Press, Teukolsky, Vetterling, and Flannery 1992] have been evaluated. The relative efficiency of these optimisation strategies is illustrated in the figures below. Comparative STRESS against time plots are given for an MLP trained according to Mao and Jain [1995] and by conjugate-gradient and BFGS techniques. Also plotted are the latter two optimisation schemes applied to the RBF NEUROSCALE model.[2] In figure 7.1, the data set used was the 150-point SPHERES_3 data, and the initial point configuration was the principal component projection. In figure 7.2, equivalent plots are given for 300 data points from the same spherical distribution. The MLP network contained 12 hidden units (tanh) with linear outputs while the RBF comprised 40 Gaussian basis functions centred at random. These values were chosen to provide similar asymptotic STRESS minima. For the on-line training of the MLP, some considerable time was required in hand-optimising the learning rate and momentum term in order to obtain even adequate performance.

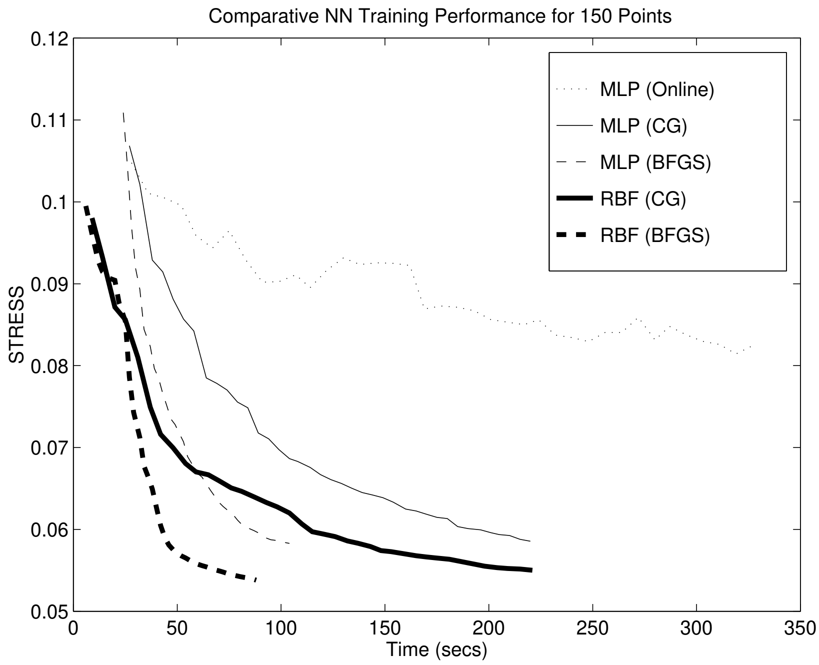

Figure 7.1: Comparison of training times for two models and three optimisation schemes for neural network implementations of the Sammon Mapping. The dataset is the 150-point SPHERES_3 set.

An example for some real, high-dimensional data is given in 7.3. The data comprise 217 points from the 1992 RAE database (see Chapter 4), and each is composed of 80 input variables. This precludes application of the BFGS technique in the case of the MLP, due to the large number of weight parameters resulting from this high input dimension. However, in the case of an RBF with fixed centres, this method may still be employed.

[2]The timings given were obtained on a 25-MFLOPS SUN Sparc-10/51 workstation.

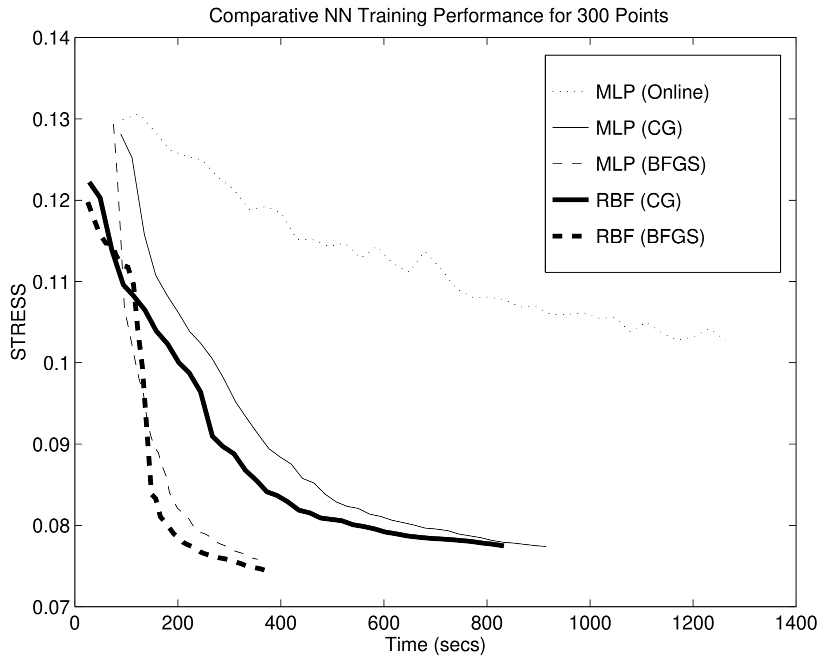

Figure 7.2: Comparison of training times for two models and three optimisation schemes for neural network implementations of the Sammon Mapping. The data is the 300-point `SPHERES_3` set.

Webb [1995] utilised the *iterative majorisation* method for minimisation of the STRESS [Heiser 1991]. This technique exploits the Cauchy-Schwarz inequality in order to minimise an upper bound on the loss function. This still entails a double loop over the data points, but does not require any gradient calculations. However, previous studies by de Leeuw [1988], of convergence in MDS applications, concluded that the method "is reliable and very simple, but that it is generally slow, and sometimes intolerably slow."

Klein and Dubes [1989] applied the stochastic *simulated annealing* optimisation technique [Aarts and Korst 1989] to the Sammon mapping. The authors assessed the performance of the scheme on three data sets (real and synthetic) and their results corroborated earlier work by de Soete, Hubert, and Arabie [1987] — simulated annealing procedures generate good (low-STRESS) configurations but run times are very extended. Klein and Dubes thus concluded that the "computational cost [of simulated annealing] makes it impractical, especially for small problems."

This feature of exaggerated computational cost is often observed with the application of simulated annealing to many problem domains [Aarts and Korst 1989]. Simulated annealing is particularly appropriate for problems either characterised by many sub-optimal local minima or where the computation of the gradient of the cost function is either impossible or computationally prohibitive. In the case of a simplified structure-preserving mapping with cost function $\sum_i \sum_j (d^*_{ij} - d_{ij})^2$, the gradient may be calculated with little additional cost to that of calculating the STRESS. The number of floating-point operations required to calculate the gradient of the STRESS is approximately 22% greater than that of the STRESS alone, and some of the requisite distance calculations may be shared by both routines.

A statistical mechanics approach was also adopted by Hofmann and Buhmann [1995] for generating MDS configurations. They exploited mean-field and saddle-point techniques in order to derive an algorithm for approximation of the MDS problem based on the Expectation-Maximisation (EM) scheme. The authors indicated an interest in bench-marking this method against standard approaches, but have not yet done this [Hofmann 1995].

Again with respect to MDS, Tarazaga and Trosset [1993] considered the reformulation of the optimi-

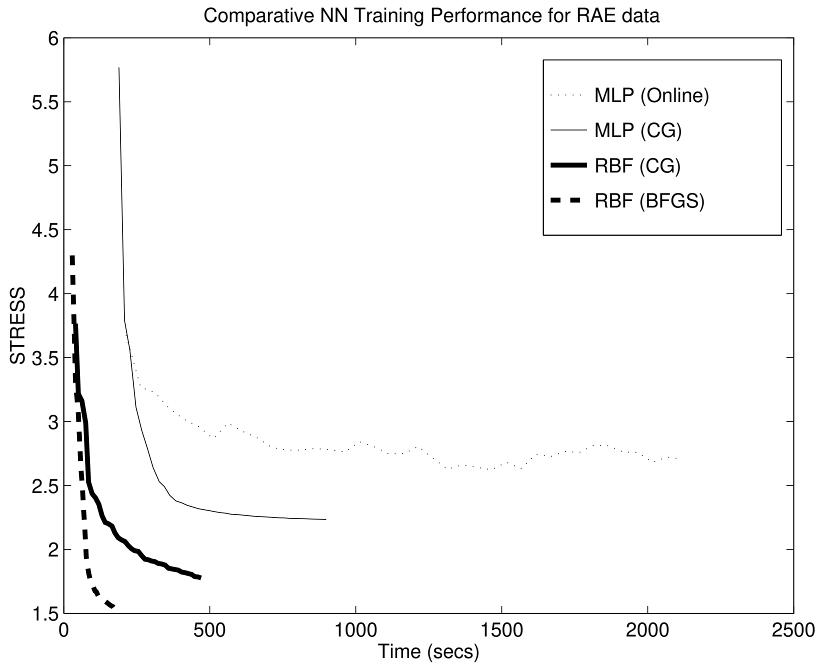

Figure 7.3: Comparison of training times for two models and three optimisation schemes for neural network implementations of the Sammon Mapping. The data comprises 217 points from the 1992 RAE database and has input dimension of 80.

sation problem in terms of functions of the matrix $\mathbf{B} = \mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ (see Chapter 5 for the context of this). This is a non-trivial task, as it requires managing constraints on the rank of **B**, and the authors do not propose any specific solutions.

7.3 An Improved Optimisation Algorithm for NEUROSCALE

Previous approaches to training neural networks to minimise STRESS-based measures have generally treated the task as one of standard nonlinear optimisation. Derivatives of the error with respect to the network weights were calculated and supplied to standard routines — gradient-descent, conjugate-gradient and BFGS for example.

This section details a new algorithm for the training of models that are linear in their weights, such as radial basis function neural networks. The advantage gained by using such networks in supervised problems with a sum-of-squares error function — that they could be trained via a single pass pseudo-inverse approach — is lost in the relative supervision case where the STRESS measure introduces quartic terms. The following algorithm effectively decomposes the training problem into a two-step procedure, one step of which is linear and can be computed efficiently.

7.3.1 The Algorithm

The equations for calculating the derivatives in the NEUROSCALE model are:

$$\frac{\partial E}{\partial w_k} = \sum_i^N \frac{\partial E}{\partial \mathbf{y}_i} \cdot \frac{\partial \mathbf{y}_i}{\partial w_k}, \quad \text{where} \tag{7.1}$$

$$\frac{\partial E}{\partial \mathbf{y}_i} = -2 \sum_{j \neq i} \left(\frac{d^*_{ij} - d_{ij}}{d_{ij}} \right) (\mathbf{y}_i - \mathbf{y}_j). \tag{7.2}$$

Equation (7.1) may be expressed in matrix form (for each separate output dimension) as:

$$\boldsymbol{\nabla} E = \mathbf{J}^{\mathrm{T}} \boldsymbol{\delta}, \tag{7.3}$$

as demonstrated in Section 6.2.2.

The set of linear equations implicit in equation (7.3) above, for the gradient of the STRESS measure with respect to the weights, is equivalent to the set of *normal equations* of a linear least-squares problem [Strang 1988]. In such a *supervised* problem, the error E is given by $E = \frac{1}{2} \sum_i \parallel \mathbf{y}_i - \mathbf{t}_i \parallel^2$, where the vectors $\mathbf{t}_i$ are the *targets*, such that

$$\frac{\partial E}{\partial \mathbf{y}_i} = (\mathbf{y}_i - \mathbf{t}_i). \tag{7.4}$$

In the relative supervision algorithm, there is an expression for $\partial E / \partial \mathbf{y}_i$ given by equation (7.2). This equation may be combined with equation (7.4) above to give a set of vectors that can be considered to represent *estimated targets* $\hat{\mathbf{t}}_i$:

$$\hat{\mathbf{t}}_i = \mathbf{y}_i - \frac{\partial E}{\partial \mathbf{y}_i}, \tag{7.5}$$

$$= \mathbf{y}_i + 2 \sum_{j \neq i} \left(\frac{d^*_{ij} - d_{ij}}{d_{ij}} \right) (\mathbf{y}_i - \mathbf{y}_j). \tag{7.6}$$

The vectors $\hat{\mathbf{t}}_i$ represent those *exact* targets for the network that would lead to an identical expression for the weight derivatives in the RBF, in the least-squares supervised case, as those obtained from the relative supervision approach. Of course, the problem cannot simply be solved in the one step as these targets $\hat{\mathbf{t}}_i$ are not fixed, but are dependent upon the current outputs of the network, $\mathbf{y}_i$, and thus the weights also.

For a fixed set of estimated targets $\hat{\mathbf{t}}_i$, the normal equations can be solved directly by

$$\mathbf{W} = \mathbf{J}^{+}\hat{\mathbf{T}}, \tag{7.7}$$

where $\hat{\mathbf{T}} = (\hat{\mathbf{t}}_1, \hat{\mathbf{t}}_2, \ldots, \hat{\mathbf{t}}_N)^{\mathrm{T}}$ and $\mathbf{J}^{+}$ denotes the *pseudo-inverse* [Golub and Kahan 1965] of the Jacobian.

An approach, therefore, to minimising E would be to repetitively estimate a set of targets and then for each successive set, solve the above least-squares problem directly. However, in the early stages of training when STRESS is high, the targets given by equation (7.5) are poor and often lead to an increase in STRESS. A more effective approach is to introduce a parameter η, which is initially small, and estimate the targets as

$$\hat{\mathbf{t}}_i = \mathbf{y}_i - \eta \frac{\partial E}{\partial \mathbf{y}_i}, \tag{7.8}$$

and increase η as STRESS decreases during the training procedure.

The training algorithm as implemented in NEUROSCALE then becomes:

❶ Initialise weights $\mathbf{W}$ to small random values.

❷ Initialise η to some small value.

❸ Calculate $\mathbf{J}^{+}$, where $\mathbf{J}_{ik} = \phi_k(\mathbf{x}_i)$.

❹ Calculate estimated targets $\hat{\mathbf{t}}_i$ from equation 7.8.

❺ Solve for weights using $\mathbf{W} = \mathbf{J}^{+}\hat{\mathbf{T}}$.

❻ Calculate STRESS.

❼
- If STRESS has increased, $\eta = \eta \times k_{down}$ where $0 < k_{down} < 1$ is a constant.
- If STRESS has decreased, $\eta = \eta \times k_{up}$ where $k_{up} > 1$ is also a constant.

❽ If not converged, return to Step ❹.

Note that, for fixed basis functions, the potentially computationally expensive pseudo-inverse calculation need only be performed once and the $(h \times N)$ matrix $\mathbf{J}^{+}$ stored. Appropriate values for k_{down} and k_{up} are, for example, 0.1 and 1.2 respectively. These values appeared acceptably robust for various data sets.

7.3.2 Convergence Behaviour

It can be shown that this new scheme converges on a minimum of E.

At time step t let

$$\mathbf{J}\mathbf{W}^t = \mathbf{Y}^t, \quad \text{and} \tag{7.9}$$

$$\hat{\mathbf{T}}^t = \mathbf{Y}^t - \eta \left.\frac{\partial E}{\partial \mathbf{Y}}\right|_{\mathbf{Y}=\mathbf{Y}^t}, \tag{7.10}$$

$$= \mathbf{Y}^t - \eta \boldsymbol{\Delta}^t, \quad \text{letting} \quad \boldsymbol{\Delta}^t = \left.\frac{\partial E}{\partial \mathbf{Y}}\right|_{\mathbf{Y}=\mathbf{Y}^t}. \tag{7.11}$$

Then,

$$\mathbf{Y}^{t+1} = \mathbf{J}\mathbf{W}^{t+1}, \tag{7.12}$$
$$= \mathbf{J}(\mathbf{J}^{+}\hat{\mathbf{T}}^{t}), \tag{7.13}$$
$$= \mathbf{J}\mathbf{J}^{+}(\mathbf{Y}^{t} - \eta\boldsymbol{\Delta}^{t}), \tag{7.14}$$
$$= \mathbf{Y}^{t} - \eta\mathbf{J}\mathbf{J}^{+}\boldsymbol{\Delta}^{t}, \tag{7.15}$$

since $\mathbf{J}\mathbf{J}^{+}\mathbf{Y}^{t} = \mathbf{Y}^{t}$ from equation (7.9). (Note that in general, $\mathbf{J}\mathbf{J}^{+} \neq \mathbf{I}$.)

Now, $\boldsymbol{\Delta}^{t}$ is the direction of steepest descent, in $\mathbf{Y}$-space, on the STRESS surface. For the proposed scheme to minimise STRESS, it is required that the change in weights, and thus in $\mathbf{Y}$, implied by equation (7.15) represents a *descent direction* upon the cost surface. For this to be the case, the inner-product of the term $\mathbf{J}\mathbf{J}^{+}\boldsymbol{\Delta}^{t}$ and the direction of steepest descent $\boldsymbol{\Delta}^{t}$ must be *positive*. This inner-product is given by

$$\mathrm{vec}[\boldsymbol{\Delta}^{t}]^{\mathrm{T}}\mathrm{vec}[\mathbf{J}\mathbf{J}^{+}\boldsymbol{\Delta}^{t}] = \mathrm{tr}\left[(\boldsymbol{\Delta}^{t})^{\mathrm{T}}(\mathbf{J}\mathbf{J}^{+}\boldsymbol{\Delta}^{t})\right], \tag{7.16}$$
$$= \sum_{l=1}^{q}(\boldsymbol{\delta}_{l}^{t})^{\mathrm{T}}(\mathbf{J}\mathbf{J}^{+})\boldsymbol{\delta}_{l}^{t}, \tag{7.17}$$

where $\boldsymbol{\delta}_{l}^{t}$ are the q component column vectors of the matrix $\boldsymbol{\Delta}^{t}$.

Since by considering its singular value decomposition as $\mathbf{J} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$, $(\mathbf{J}\mathbf{J}^{+})$ is equal to $\mathbf{U}\mathbf{U}^{\mathrm{T}}$ and is thus positive semi-definite. The inner-product will then be non-negative for all $\boldsymbol{\Delta}^{t}$. Indeed, it will be positive excepting when the subspace of $\boldsymbol{\Delta}^{t}$ lies in the null-space of $(\mathbf{J}\mathbf{J}^{+})$, which is identical to the null-space of $\mathbf{J}\mathbf{J}^{\mathrm{T}}$. Thus the local minima of the cost surface descended by the shadow-targets algorithm are identical to those of E.

7.3.3 Interpretation

Firstly, consider equation (7.8). The estimated target $\hat{\mathbf{t}}_{i}$ is identical to the new point obtained after one step of a gradient-descent optimisation of the standard Sammon Mapping. In this sense, the proposed algorithm is effectively *shadowing*, step by step, the standard Sammon Mapping generation procedure. For this reason, the algorithm will be subsequently referred to as the *shadow-targets* algorithm.

However, in contrast to the Sammon mapping, by reference to equation (7.15), the combination of reduced basis functions and smoothing parameter(s) in the RBF network serves to transform this standard optimisation step $\boldsymbol{\Delta}$ to an alternative step, $(\mathbf{J}\mathbf{J}^{+})\boldsymbol{\Delta}$. The matrix $\mathbf{J}\mathbf{J}^{+}$ is effectively a projection matrix (it is clearly idempotent) that projects the columns of the error matrix $\boldsymbol{\Delta}$ onto the subspace spanned by the columns of $\mathbf{J}$. If there is a full complement of basis functions of sufficiently small σ (such that $\mathbf{J}$ is not ill-conditioned), then $\mathbf{J}\mathbf{J}^{+} = \mathbf{I}$, and the algorithm effectively defaults to a repetitive *a posteriori* fitting of an evolving Sammon map. Typically, however, either there will be fewer basis functions than patterns or, as demonstrated in Section 6.6.3, the width parameter σ of the basis functions will be chosen such that $\mathbf{J}\mathbf{J}^{+}$ is singular.

7.3.4 Performance Comparison

Speed of Training

Figure 7.4 illustrates typical plots of the evolution of STRESS during training for an RBF model trained on the 150-point SPHERES_3 data set. The RBF comprised 40 Gaussian basis functions ($\sigma = 2.0$), and training performance is shown for the best standard non-linear optimisation technique, BFGS, and for shadow-targets.

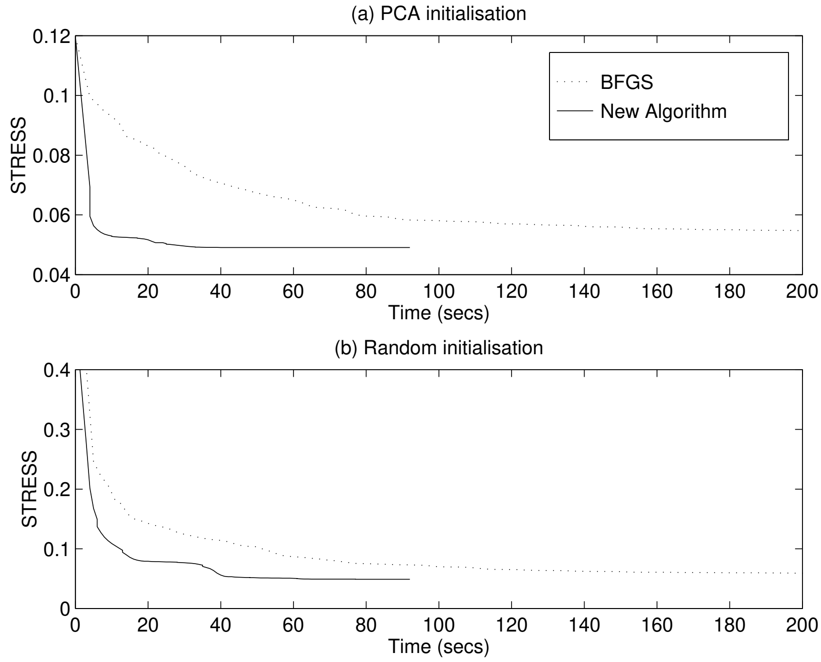

Figure 7.4: Evolution of STRESS during training for both BFGS optimisation, and shadow-targets, on the `SPHERES_3` dataset. The upper plot (a) is for PCA initialisation, its partner (b) is for random initialisation.

For the high input dimension data set `RAE_PCB`, the illustration of STRESS minimisation is given in figure 7.5. In this example, a plot is only given for PCA initialisation. There were 70 basis functions in the network of type $r^2 \log(r)$.

Local Minima

In terms of the algorithm's susceptibility to entrapment in sub-optimal local minima, an identical experiment to that given in figure 6.3 is given in figure 7.6. For the `IRIS_45` dataset, 1000 mappings were generated with the shadow-targets algorithm, and a histogram of the final minimum STRESS values plotted.

Because the shadow-targets algorithm follows the Sammon mapping, it is unsurprising to observe that in figure 7.6, the minima in the plot correspond to those of the Sammon Mapping, the level-1 minima. In comparison to the standard BFGS optimisation technique, the algorithm exhibits a significant number of sub-optimal minima, although because of the apparent absence of level-2 minima, the lowest STRESS obtained is now 0.0028, and on nearly 50% of the runs. This represents a very considerable improvement when compared directly to the Sammon Mapping histogram of figure 6.1, and is as a result of the perturbing of the trajectory in $\mathbf{Y}$ space by the $(\mathbf{JJ}^{+})$ matrix, with its associated smoothing effect.

It was shown in Section 7.3.2 that the shadow-targets algorithm was susceptible to the same level-2 minima as other minimisation methods. However, as noted in the above paragraph, figure 7.6 indicates that the algorithm is not trapped by these minima, and the explanation given in Section 6.2.2 concerning invariance of STRESS under rotation and translation once again applies. The major distinction between the shadow-targets algorithm and those employed previously is that the effect of perturbing solutions within the principal eigenspace of the network Jacobian $\mathbf{J}$ is no longer evident. More direct evidence for this is given in figure 7.7, which plots the evolution of STRESS for two identical networks from identical starting configurations. One of the networks is trained by simple gradient-descent, and one by shadow-targets. In both cases, the learning rate η was identical and constant during training.

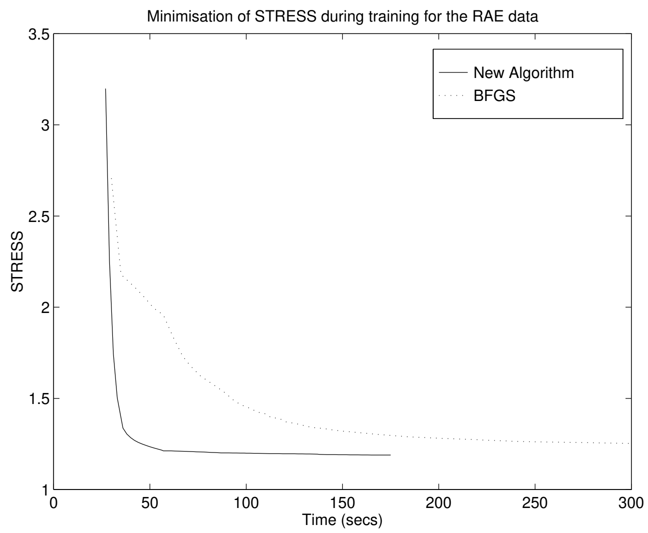

Figure 7.5: Evolution of STRESS during training for both BFGS optimisation, and shadow-targets, on the RAE_PCB dataset.

The gradient-descent trained network exhibits the behaviour illustrated in figure 6.4 earlier, where training is effectively terminated as the error vectors $\boldsymbol{\delta}$ are in the direction of the minor eigenvectors of $\mathbf{J}$. That the gradient-descent scheme initially exhibits faster convergence than its shadow-targets counterpart is also consistent with the behaviour illustrated in that earlier figure. The trajectory of a single output dimension of the configuration for gradient-descent is defined by

$$\Delta \mathbf{z}_{gd} = -\eta \mathbf{J}\mathbf{J}^{\mathrm{T}} \boldsymbol{\delta} = -\eta \sum_{k} \lambda_k^2 \beta_k \mathbf{u}_k, \tag{7.18}$$

as shown in Section 6.2.2. For shadow-targets, this is

$$\Delta \mathbf{z}_{st} = -\eta \mathbf{J}\mathbf{J}^{+} \boldsymbol{\delta} = -\eta \sum_{k:\lambda_k \neq 0} \beta_k \mathbf{u}_k, \tag{7.19}$$

and the trajectory is no longer perturbed towards the direction of the principal eigenvectors of $\mathbf{J}\mathbf{J}^{\mathrm{T}}$.

Curvature and Generalisation

As the shadow-targets algorithm effectively repetitively fits a Sammon mapping, and does not seek a solution that implicitly reduces $\| \mathbf{W} \|^2$, it might be expected that values of curvature during training should be significantly greater than for the BFGS training method. By contrast with figure 6.8, figure 7.8 illustrates the curvature of the RBF transformation during shadow-targets training on the IRIS_45 data set. For comparison, the equivalent plot for BFGS training, in figure 6.8, is also plotted. The RBFs in both experiments had 45 Gaussian basis functions with width parameter $\sigma = 1.0$, and the results confirm the hypothesis that curvature is not significantly reduced. (Some reduction in curvature may often be observed if the initial weights tend to be too large.)

The plot of the evolution of STRESS and curvature for the new algorithm is superimposed on that for BFGS and *a posteriori* fitted RBFs and shown in figure 7.9. This illustrates that the curvature of a network, trained by shadow-targets to a given value of STRESS, is in most cases lower than that of an identical network, supervisorially trained to produce a standard Sammon configuration, although not to the same extent as the BFGS-trained version.

An indication of the generalisation performance that may be obtained from networks trained by the shadow-targets algorithm is given in figure 7.10. This illustrates, in a similar form to figure 6.13 on

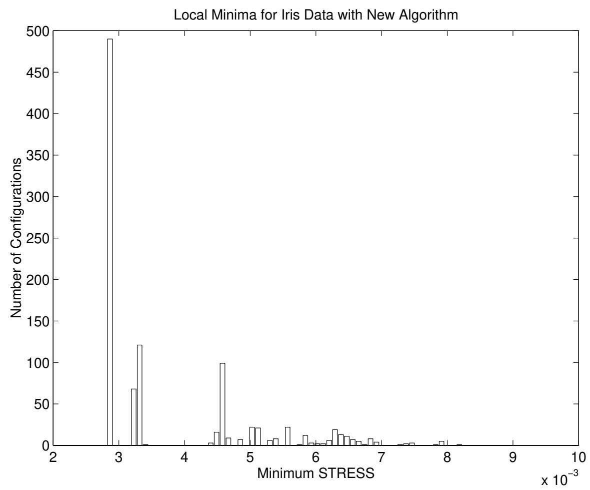

Figure 7.6: Histogram of number of final configurations with corresponding STRESS's for 1000 runs of NEUROSCALE, with $\alpha = 0$, on the IRIS_45 dataset. The width of the 45 Gaussian basis functions was 3.0. The NEUROSCALE model was trained with shadow-targets.

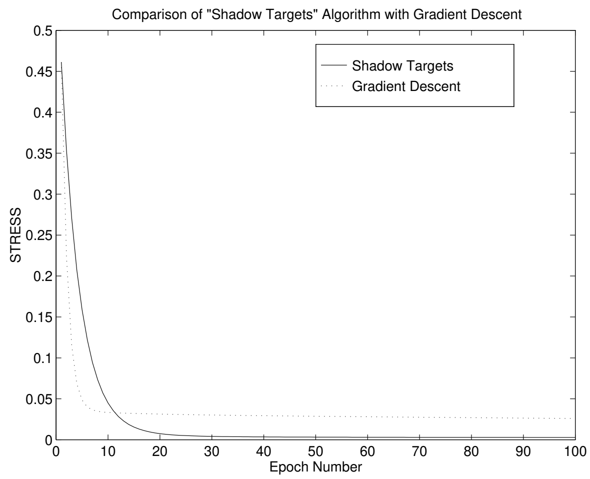

Figure 7.7: Evolution of STRESS for identically initialised networks for gradient descent and shadow-targets training.

page 99 of the previous chapter, the test STRESS that would be obtained by the network at each step of the training algorithm. This is shown for three separate networks with values of σ of 1,10 and 30, and the equivalent plots for BFGS (where $\sigma = 1$) and *a posteriori* networks are superimposed.

From figure 7.10, it is evident that the minimum training STRESS with shadow-targets is superior to that of BFGS in all cases. However, for equivalent values of basis function width parameter ($\sigma = 1$), test STRESS is lower for the relatively supervised version. This is a result of the implicit weight decay in the latter algorithm, and is consistent with the curvature plot of the previous figure. However, it can be seen that for a higher value of width parameter, 30, both training and test STRESS are much lower with shadow-targets. A network trained by BFGS with $\sigma = 30$ would exhibit very high STRESS due to the exceedingly poor conditioning of the matrix $\mathbf{J}$. The plot for shadow-targets with $\sigma = 10$ reflects the unusual behaviour demonstrated in figure 6.22 in the previous chapter, where it was observed that generalisation performance was relatively poor for certain intermediate values of basis function width.

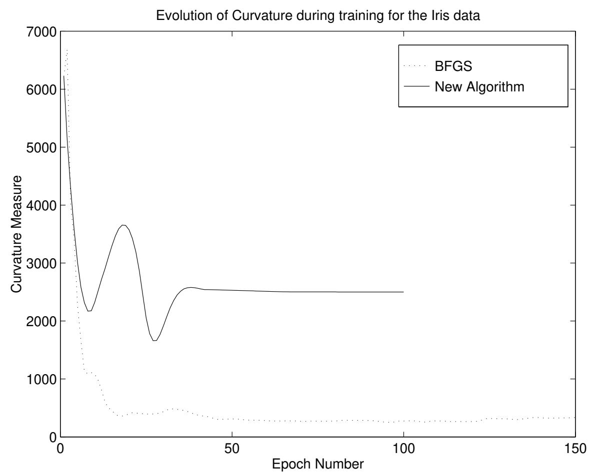

Figure 7.8: Curvature against time during the shadow-targets training of a NEUROSCALE mapping on the IRIS_45 data.

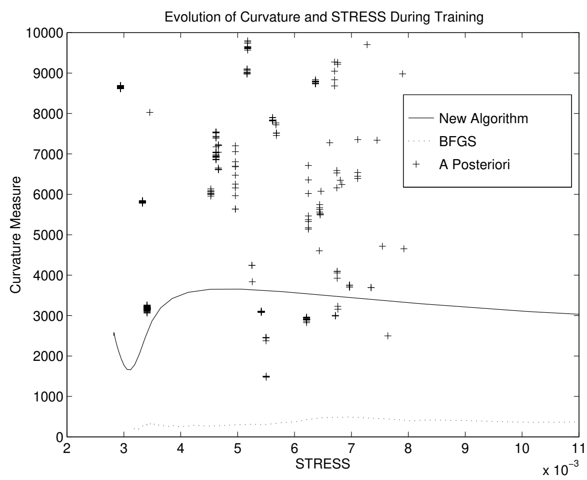

Figure 7.9: Curvature of a posteriori fitted RBF networks compared to that during training of a NEUROSCALE model by BFGS and shadow-targets methods.

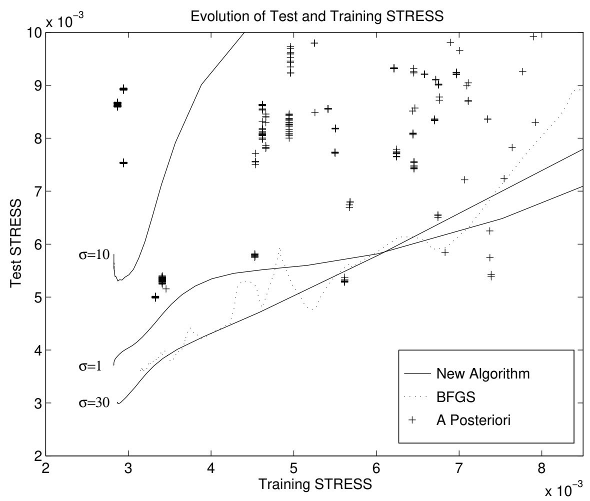

Figure 7.10: Test STRESS of a posteriori fitted RBF networks compared to that during training of a NEUROSCALE model by BFGS and shadow-targets methods.

7.4 Alternative Mapping Strategies

7.4.1 Review

Sammon [1969] himself remarked upon the undesirable scaling properties of his projection method and suggested that some sort of data compression pre-processing might be appropriate. Specifically, he utilised a clustering algorithm to determine a set (maximum ≈ 250) of data prototypes which may then be mapped using the standard algorithm. This approach was further refined by Pykett [1978], who mapped class centroids and superimposed circles on the map as an approximate indicator of class dispersion.

Chang and Lee [1973] proposed two alternative strategies for mapping large numbers of pattern vectors. The first, the "relaxation method" is really an optimisation technique and is designed to minimise the contribution to the overall STRESS of individual pairs of points. It is possible to directly perturb a pair of points $\mathbf{y}_i$ and $\mathbf{y}_j$ such that $\| \mathbf{y}_i - \mathbf{y}_j \| = d^*_{ij}$. This will naturally disrupt all other optimised distances, and Chang and Lee proposed reducing this disruption by introducing a term into the algorithm that attenuates the perturbation for increasing distance, and so making the mapping more local. Iterated application of this algorithm should then produce a stable solution. In more extensive analysis, however, Siedlecki, Siedlecka, and Sklansky [1988] observed the algorithm to be often unstable and to possess poor convergence properties.

Chang and Lee's second strategy was the "frame method". In this mapping, a subset of M of the original N points are selected and mapped according to Sammon's algorithm to generate a fixed *frame*. All the remaining $N - M$ points are then mapped, but are only constrained by the inter-point distances to the M points in the frame, as distinct from the entire set of N data points. This is equivalent to the mapping of a $N \times M$ distance matrix, and so clearly saves on storage and computational requirements. (However, there is no reason given for the fixing of the M points first, in preference to simply scaling the entire $N \times M$ distance matrix in one go.) The quality of this approximated mapping obviously depends on both the characteristics of the data and on the M points in the frame. An example is given of the frame method on a simple data set, but Siedlecki et al. [1988] question the evidence for the value of this approximate mapping technique. However, in a paper on the representation of rigid structures with minimal 'connected-ness' information, Levine and Kreifeldt [1994] discuss the potential of extending their work to the generation of MDS configurations from a partial set of distances and indicate future research in this area.

Another approximation approach was the *triangulation* mapping of Lee, Slagle, and Blum [1977]. This is based upon the fact that any three points may be placed in a plane such that all three inter-point distances are preserved exactly. Thus in a single pass of the dataset, $2N - 3$ distances (of the total $N(N-1)/2$) may be retained by locating each point sequentially such that the distances to two previously projected points are retained. There remains the choice of which of these distances to select for preservation — the authors propose that a good criterion for selection is that the *minimal spanning tree (MST)* of the data be maintained. (The minimal spanning tree [Gower and Ross 1969] is the tree connecting all points, with no loops, that has the smallest span, defined by the sum of the inter-point distances along the tree.) This entails the mapping of each point from the MST (in a *breadth-first* manner) so that the distance to its nearest neighbour is fixed, plus one additional distance which may either be selected as the second nearest-neighbour or some alternative reference point. This choice thus allows the emphasis of the mapping to be either local or global, or indeed, offers the user a choice of "perspective". Siedlecki et al. [1988] illustrate how this selection can affect the clarity of clusters in the final projection. Note that the construction of this mapping will be partially ambiguous as there are two alternative solutions for the placing of any point in the plane with respect to any other two points.

A modified form of the Sammon STRESS was adopted by Niemann and Weiss [1979]. The criterion to be minimised was of the form $\sum_{ij}(s^*_{ij})^r(s^*_{ij} - s_{ij})^2$, where s^*_{ij} and s_{ij} were the *squared* Euclidean distances in the data and map space respectively. Choice of parameter r allowed greater emphasis to be placed on local or global structure. The authors also utilised a co-ordinate descent technique, which involved adjusting a single parameter in the map at each step, by a fixed amount which was determined by the solution of a cubic equation. Siedlecki et al. [1988] observed this method to have very good convergence properties, but that it generated considerable distortion.

The frame method and the triangulation mapping were combined by Biswas, Jain, and Dubes [1981], to effect the *Sammon-Triangulation* technique. They proposed the generation of an M-point frame first, followed by the mapping of all other points, via the triangulation method, to retain the distances to the two nearest-neighbours from amongst the points within the frame. An additional step in the procedure is required to ensure that the triangle equality is maintained for the projected points. The authors claim that the frame method "leads to projections that greatly distort the data" while the Sammon-Triangulation scheme "produces better results than the triangulation method because more global information about the data set is retained". For a large number of patterns, Biswas et al. reiterate the suggestion of mapping only a prototypical subset of the patterns, or of performing some *a priori* clustering.

The approach to reducing the storage and computational cost of Kakusho and Mizoguchi [1983] is to only constrain the position of each point by the distances to the k nearest-neighbours (in data space). This is another form of a localised Sammon mapping, and exploits a basic idea from NMDS dating back to Shepard and Carroll [1966]. A similar concept is entertained by Schwartz, Shaw, and Wolfson [1989] for the mapping of the convoluted 3-dimensional primate visual cortex. The mapping of a given point is again restricted to a local "patch", with the remark that the distance matrix then becomes sparse and can be maintained as a collection of binary search trees. The patches were empirically chosen to represent around 10% of the total data area.

7.4.2 Comment

Apart from both the relaxation method of Chang and Lee [1973] and the co-ordinate descent technique of Niemann and Weiss [1979], all the other approaches discussed above are effectively *ad hoc* methods for *subset* mapping. Furthermore, the majority of these schemes are based on retention of *local* distances [Chang and Lee 1973; Lee, Slagle, and Blum 1977; Biswas, Jain, and Dubes 1981; Kakusho and Mizoguchi 1983; Schwartz, Shaw, and Wolfson 1989].

It has already been emphasised that in general visualisation applications, global structure can be highly informative — for example, the ordering of research ratings in the RAE data from Chapter 4. While local mappings seek to maintain clusterings under the dimension-reduction process, the inter-cluster structure is often equally, and sometimes more, important. Indeed, too much emphasis on local spatial information can lead to confusion of clusters, a phenomenon observed by Siedlecki et al. [1988]. In such cases, the original approach suggested by Sammon [1969], to select cluster prototypes for mapping, is actually the most sensible.

The requisite data pre-processing stage that this implies only transfers the scaling problem to one of selecting either the best set of prototypes or the best subset of the distance matrix, and this is a non-trivial question which is also likely to be computationally demanding. Of these two alternative strategies, selecting M_1 prototype points for individual mapping requires scaling a $(M_1 \times M_1)$ matrix, while selecting M_2 points as a basis for scaling a subset of the distance matrix requires scaling a $(N \times M_2)$ matrix. Clearly, for similar computation, more prototypes can be selected in the first method than in the second — $M_1 \gg M_2$ for large N. However, the second approach does permit the mapping of the entire set of data points, albeit constrained by only a partial set of distances.

The use of a parameterised transformation, as exploited in NEUROSCALE, is an advantage when scaling a subset of data points. Because it is now possible to project new data, all the remaining data points may still be mapped by the transformation once it is defined. The network is effectively training on the subset alone, and then generalising to the remaining data. Clearly, the subset of points must be representative of the distribution of the data as a whole if a good projection is to be expected.

One particular example approach might be to determine a set of K cluster centres, via the K-means procedure [MacQueen 1967], and then scale these with the standard NEUROSCALE algorithm. Subsequently, the entire dataset may be projected. Strictly, the STRESS calculation for the K means should be adjusted to account for the number of data points that each mean represents.

The above algorithm is only one of many potential approaches for coping with the demands of large datasets. Overall, this suggests, as intimated by Levine and Kreifeldt [1994], that the key issue is that of choosing the subset of distances that, when scaled to generate the mapping, best approximates the mapping obtained by scaling the complete distance matrix. This is clearly a very complex question, and one that demands considerably more study than there is opportunity to include in the remainder of this thesis. As such, it must remain an important area for future research.

7.5 Conclusions

The comparison of standard nonlinear optimisation procedures for both MLPs and RBFs revealed that the quasi-Newton BFGS technique exhibited best performance in terms of minimising training time. Furthermore, RBFs were better in this respect than MLPs, particularly where the input dimension was high and the number of weights in the MLP precluded the use of BFGS. On-line training of an MLP was significantly worse than all other optimisation approaches.

However, a new 'shadow-targets' training algorithm proposed in Section 7.3 for networks with outputs linear in the weights, such as RBFs, proved to be significantly better, in terms of speed of convergence, than even BFGS, attaining the region of the local minimum an order of magnitude more quickly. This algorithm was also shown to converge to the same classes of minima as the standard methods, and still exhibit generally lower curvature and test STRESS than similar *a posteriori* mappings, due to its inherent smoothing effect. However, for equivalent basis function smoothing parameter σ, networks trained with the shadow-targets algorithm exhibited greater curvature than those employing standard nonlinear optimisation techniques, as the former does not incorporate the implicit weight-decay of the latter methods in the relative supervision context.

Importantly, however, when employing the shadow-targets algorithm it is possible to use larger values of σ without suffering from the increased local minima effects discussed in Section 6.2.2 in the previous chapter. This enables the efficient generation of low-STRESS training *and* test transformations, which are superior to the standard nonlinear-optimised versions. In this respect, the method of choice for the training of a topographic transformation would be to utilise the shadow-targets algorithm with an appropriately large σ. This does necessitate, however, the determination of a suitable value for the basis function width parameter, although from the results of Section 6.6.3, this choice may be guided by the heuristic that "bigger is better".

It should be noted that the shadow-targets training algorithm is equally appropriate to mappings which incorporate some subjective element (i.e. $\alpha > 0$) and can be expected to offer significant reductions in the required training time for such applications. However, the argument that the transformation should be maximally smooth no longer applies and this can be expected to complicate the choice of σ, which is now likely to be significantly data-dependent. Further investigation would be appropriate in this direction.

Finally, in Section 7.4, some of the alternative approaches designed to ameliorate the computational problems associated with the mapping of large datasets were reviewed. These may be seen as generally *ad hoc* approaches to clustering or subset selection, and as such pose equally non-trivial problems which, for the purposes of topographic mappings, is still an open research issue.