Metrics and visualisation for crime analysis and genomics

Laros, J.F.J.

Chapter 4

Error Visualisation in the Particle Model

In this chapter, we introduce a new method of analysing the errors introduced by multidimensional scaling techniques. We associate an error of an item in the output of these methods with a charge, and then interpolate this charge to define a field.

We give a general method on how to do this interpolation and we provide several methods and fine tuning techniques to calculate the charge of the items.

4.1 Introduction

The *push and pull* model [43], also known as the particle or spring model [19] is a way of accomplishing a dimension reduction. However, as is the case in each dimension reduction algorithm, an error is introduced in the output in almost all cases. This error can be uniformly distributed throughout the whole picture (this would perhaps be preferable), or it can be accumulated in one or more parts of the picture. In the latter case, parts of the constructed dimension reduction can be useless and thus visualisation of the error would gain insight in the quality of the picture.

If an error map could be constructed for a dimension reduction, a potential user could use this map to assess the quality of the original picture and decide which parts are usable or not. By making a continuous map of the error, various standard techniques, like looking at the gradient, can be used for further analysis.

Error visualisation is important for dimension reduction techniques, because (apart from some trivial cases) we know that the produced image is only a projection of the original data in some way, an approximation so to speak. Because of this, the resulting image has errors and these errors might or might not be uniformly distributed throughout the image. In the first case, there is no further analysis to be done, in the latter case however, it can very well be that a part of the outcome of a dimension reduction is completely unusable, while an other part is still of value. To find out which parts of such an image are valuable we need to gain insight into the error distribution throughout the image.

Populair methods to visualise the error usually give each item in the output picture a colour, or a grey value, denoting its error, but with many points this can give a rather chaotic picture which is hardly usable.

A general way of constructing an error map is given in Section 4.2. Furthermore, two natural implementations are given, along with a general threshold function to filter out small deviations in the error maps. An artificial example dataset and various experiments on it are given in Section 4.3. We conclude in Section 4.4, we speculate upon the application of our method in other dimension reduction techniques and give suggestions for further research.

4.2 Constructing the error map

In the push and pull model, the distances between any two particles p and q from a given set is given by an input matrix. We call these distances the *desired distances*, denoted by $d_{\text{desired}}(p,q)$. The actual distances in the picture we call the *realised distances*, denoted by $d_{\text{realised}}(p,q)$. Clearly, the goal of the push and pull model is to minimise the difference between d_{desired} and d_{realised} for all pairs of points.

The output of the push and pull technique is a statical model of particles, where in general there is a difference between the desired and realised distances. The difference between these distances contributes to the error of a particle, in some way. There are several choices of functions to calculate the total error connected with a particle. This will be discussed in the next two subsections.

Once the total error connected with a particle is calculated, we can make a plot of the error propagating through space. We let the error decay as the distance to the particle grows. The error of a particle can be seen as analogous to a charge, an *error charge* so to speak. We let this error propagate through the space, analogous to the way an electrical charge propagates. We use a variant of the well-known *law of Coulomb* to calculate the field strength at any distance from the particle. When there are multiple particles, we use the sum of the field strength values to calculate the strength of the *error field* at any point in the space.

The scalar form of Coulomb's law, which describes the magnitude of the electrostatic force F on two charged particles q_1 and q_2, is defined as:

$$F = k_e \frac{q_1 q_2}{r^2}$$

where r is the distance between the particles and k_e is Coulomb's constant.

Of course, we are not bound to the law of Coulomb to indicate the decline of the field strength. The space that is the result of the dimension reduction technique needs not to be flat, so using a different function to calculate the error field is quite acceptable. Furthermore, we have no particular reason to use Coulomb's law, except for the analogue with nature and therefore mans familiarity with it.

In a model with n particles $\{p_1, p_2, \ldots, p_n\}$, we use the following formula to calculate the field strength at position (x, y) in the error map:

$$\mathit{field}(x,y) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathit{error}(p_i)}{(1 + d((x,y), \mathit{pos}(p_i)))^{\gamma}}$$

Where γ is the speed at which the field drops off. Choosing $\gamma = 1$ will result in a linear drop off, $\gamma = 2$ will result in a drop off similar to the one in the law of Coulomb. In this paper, we shall use $\gamma = 0.5$. The function d calculates the Euclidean distance between the point (x, y) and the location of the particle p_i, which is given by the $\mathit{pos}(p_i)$. The function *error* is a generalised function returning an error value. In the following subsections we discuss a number of possibilities for this function and their consequences for the error map.

A first and natural choice for the error function follows from the assumption that each particle has an equal amount of influence over each other particle. Thus we use the average of all pairwise errors to calculate the total error of a particle. We refer to this average als the *global error*.

So in a model consisting of n particles, the global error of a particle $q \in \{p_1, p_2, \ldots, p_n\}$ is calculated with the following formula:

$$\mathit{error}_{\text{global}}(q) = \frac{1}{n}\sum_{i=1}^{n} |d_{\text{desired}}(p_i, q) - d_{\text{realised}}(p_i, q)|$$

After applying this function to each particle in the model, we can make the error map with the *field* function, as described above.

If one wants to gain more insight in the (possibly dense) clusters that can arise, a natural approach is to use a decline function for the error itself. This cancels the errors of remote particles (and possible overshadowing effects of those remote particles) and focuses on the particles in the area. To calculate this error, we use a weighed average of all pairwise errors, where the weight is a (declining) function of the distance between the particles. We refer to this weighted average as the *local error*.

Again, we have a model consisting of n particles. To calculate the local error of a particle $q \in \{p_1, p_2, \ldots, p_n\}$, we use the following formula:

$$\mathit{error}_{\text{local}}(q) = \frac{1}{n}\sum_{i=1}^{n} \frac{|d_{\text{desired}}(p_i, q) - d_{\text{realised}}(p_i, q)|}{(1 + d_{\text{realised}}(p_i, q))^{\delta}}$$

Here δ is the speed at which the error of a particle loses its influence over other particles (comparable to the function of γ in the *field* function). In this chapter we use $\delta = 0.5$.

The local error field for each point in the error map can be calculated with the *field* function again.

4.2.1 Minimum correction

A technique that can be applied before we calculate either the local or the global error field is to define a threshold for the error and subtracting this threshold from the particles that have a higher error. Using such a threshold will filter out small errors that might not be of interest.

A natural choice for the threshold is the minimum error that occurs. We shall call this the *minimum correction* for the remainder of this chapter. To do the minimum correction, we first calculate the minimum of the total errors of all particles

$$err_{\min} = \min(error(p_1), error(p_2), \ldots, error(p_n))$$

and then subtract $err_{\min}$ from the error of each particle:

$$error'(p_i) = error(p_i) - err_{\min} \quad (1 \leq i \leq n)$$

For pictures with a high number of particles, of which the errors have a broad distribution, one can choose a higher threshold to see which points are responsible for most of the error.

4.3 Experiments

To give a good impression of the effect of the different choices that can be made, we have constructed a small, artificial dataset.

$$\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\sqrt{2} & \frac{3}{20} & \frac{9}{10} \\ \frac{1}{2} & 0 & \frac{1}{2}\sqrt{2} & \frac{1}{2} & \frac{3}{20} & \frac{9}{10} \\ \frac{1}{2} & \frac{1}{2}\sqrt{2} & 0 & \frac{1}{2} & \frac{3}{20} & \frac{9}{10} \\ \frac{1}{2}\sqrt{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{3}{20} & \frac{9}{10} \\ \frac{3}{20} & \frac{3}{20} & \frac{3}{20} & \frac{3}{20} & 0 & \frac{9}{10} \\ \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & 0 \end{pmatrix}$$

Table 4.1: Sample input

In Table 4.1, we see the pairwise distances of six defining points from an object that resembles the one shown in Figure 4.1 (leaving out one point). The points a, b, c and d form a square, point f has a large distance to the corner points, making it the top of a pyramid. Point e is only defined as being very close to the corners of this square. So close in fact, that the triangle inequality does not hold because of this point e. If we take out e, the remaining points would have been embeddable in $\mathbb{R}^3$ though, as can be seen in Figure 4.1.

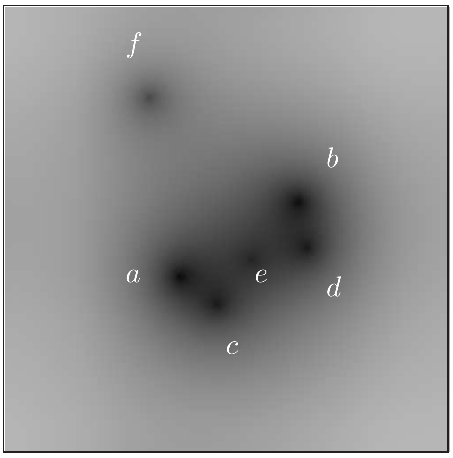

Figure 4.2: Global error map

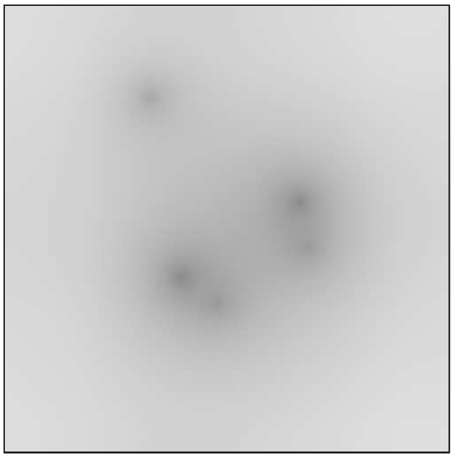

Figure 4.3: Global error map with minimum correction

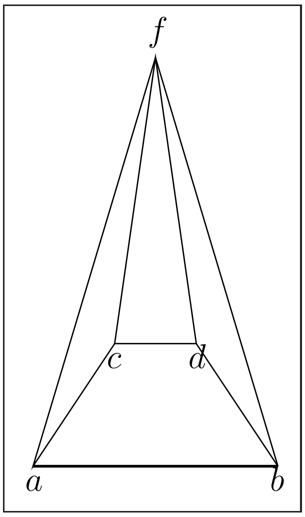

Figure 4.1: Approximate sample input (apart from one point)

As mentioned before, apart from point e, the object is a pyramid; the submatrix consisting of the first four rows and columns give the distances of the square ground plane and the last row and column give the distances to the top. The total figure can certainly not be embedded in a 2-dimensional plane (even a pyramid would be a problem), so we must introduce a number of approximations to make an embedding possible.

In Figure 4.2, we see the global error map of a dimension reduction done on the data in Table 4.1, where, as mentioned before, we use $\gamma = 0.5$. The shape is what we would expect. The ground plane of the pyramid, consisting of points a, b, c and d are the four points in the centre of the image, now in the form of a trapezoid. Point e is in the centre of this trapezoid and point f can be found in the upper left of the picture.

From this map, it is clear that the two points a and b, located on the broad side of the trapezoid have the largest error.

If we apply the minimum correction on the global error map, we get a picture as seen in Figure 4.3. Point e has vanished from this map and it is even more clear which points have the largest error.

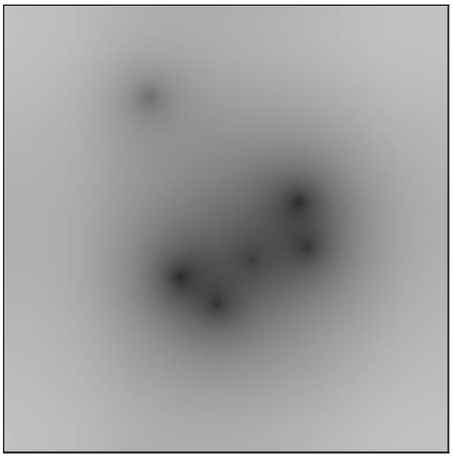
Figure 4.4: Local error map

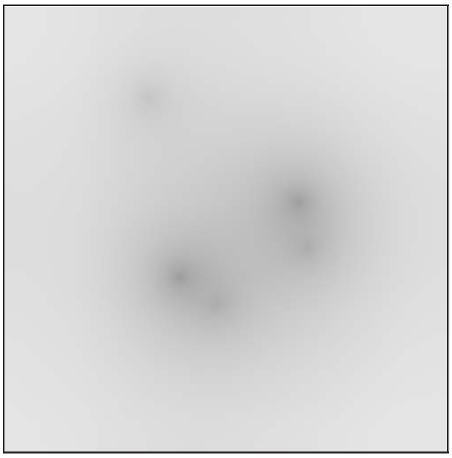
Figure 4.5: Local error map with minimum correction

In Figure 4.4 we see the local error map of the dimension reduction done on the data given in Table 4.1, where, as mentioned before, we use $\gamma = 0.5$ and $\delta = 0.5$. When compared to Figure 4.2, we see that the top of the pyramid, point f, is relatively less prominent in the local error map. The difference between the points in the ground plane, points a, b, c and d and the other points, e and f, however, has gotten larger. This can be explained by the disturbing influence of point e, that apparently is responsible for most of the errors of the points in the ground plane.

If we apply the minimum correction on the local error map, we get a picture as seen in Figure 4.5. Again, point e has vanished and this time point f is almost invisible as well. The only points that are still clearly visible are the ones that have a significant error because of a local disturbance.

Because the error map is a "continuous" map, we can apply some standard computer graphics techniques on it for further analysis. We can for example compute the derivative of the error map,

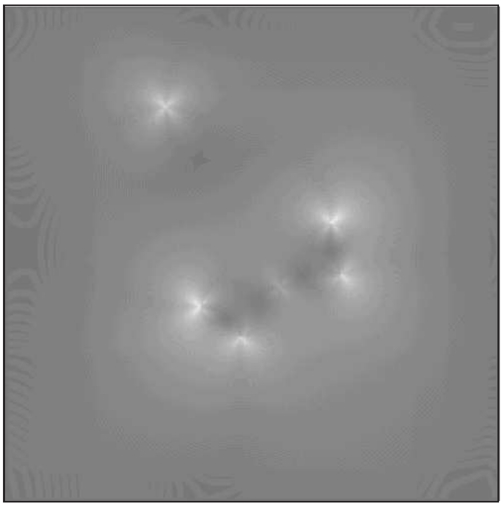

Figure 4.6: Gradient of the global error map

In Figure 4.6 we see such a derivative. We made this image by taking the maximum gradient in a 5×5 square, iterated over all points in the original image.

What we can see from this picture is the “stability” of the error field. In the dark areas, the error does not change very much, so they can be called stable. The white areas are the ones that have large fluctuations in the error. For more complicated input data, this can be of use to assess the quality of the error map.

4.4 Conclusions and further research

In this chapter, we have shown that the construction of an error map can be useful for the analysis of the quality of (parts of) the output of a dimension reduction technique. We have given several natural approaches to construct different kinds of error maps, each emphasising different aspects of the error under consideration.

The method described in this chapter can be applied to other dimension reduction techniques as well. For example, it can be applied for Principle Component Analysis [35,57], Principle Curves [29] and Multidimensional Scaling [5,16] and their (metric) variants without alteration. The non-metric variants of Multidimensional Scaling require a radically different error function, because these techniques don’t use a metric distance in their visualisation.

On the other hand, Self Organising Maps [39] have their own way of constructing an error map: by calculating a gradient from the original map.

Apart from the examples we have given, other kinds of visualisations are possible. We can, for example, use different values for γ to alter the speed at which the error field drops off. This can be useful if we want to analyse the visualisation of a large number of input points. In that case we could increase the drop off speed to analyse the errors of particles that are very close together.

Automatically determining the value of γ would be an interesting follow up for this research.

We can also use different decline functions. This might be preferable if we have some expert knowledge of the input data for example.