Learning from Data 1
Data Visualisation

David Barber
dbarber@anc.ed.ac.uk
course page : http://anc.ed.ac.uk/∼dbarber/lfd1/lfd1.html

In data visualisation we attempt to gain intuition about the structure of a dataset. Typically, this method is unsupervised (does not make use of any target values), although supervised visualisation is also possible. We have seen how to use PCA to reduce the dimensionality of data in a linear fashion to such a degree that we can plot the reduced dimension dataset in two or three dimensions. Canonical Variates (see elsewhere) also performs linear dimension reduction exploiting class information – if we reduce the dimension to only two or three, we can also visualise the data. Non-linear dimension reduction, such as autoencoders can also be used in the same way for visualisation. In autoencoders, we constructed the error function to be the squared error loss between the input and the output. However, there was no explicit requirement that the low dimensional representation of the data should, in some sense, be a good visualisation of the high dimensional data. This issue is addressed here by considering methods that try to preserve (at least locally) the topology of the high dimensional data.

Multidimensional Scaling

In multidimensional scaling (MDS) we are given distances d_{rs} between every pair of observations (that is, we may not have direct access to any high-dimensional data, but we do have access to a measure of the "distances" between every two points). The idea is to try to reconstruct what the original data was, based solely on these distances. For example, given only the distances between towns, can we construct a map for the coordinates of the towns themselves? A practical area for such methods is in the visualisation of proteins and other macromolecules based on measures of similarity between the molecules.

1 Classical Scaling

Consider a set of datapoints $\{\mathbf{x}^\mu, \mu = 1, \ldots, P\}$, where the dimension of each datapoint, $dim(\mathbf{x}^\mu) = n$. From this data we can form the distances between all the datapoints:

$$T^x_{ab} = (\mathbf{x}^a - \mathbf{x}^b)^2 \tag{1.1}$$

to form a $P \times P$ distance matrix T^x between the $\mathbf{x}$ datapoints. The idea in Classical Scaling is to find a set of vectors $\mathbf{y}^\mu, \mu = 1, \ldots, P$ such that the distance matrix T^y between the $\mathbf{y}$ points matches as closely as possible the distance matrix T^x. The dimension of the datapoints, $dim(\mathbf{y})$ is typically chosen to be small, either two or three so that we can visualise the data.

In other words : Given a distance matrix T only, how can we find a set of points $\mathbf{y}^\mu$ that has this distance matrix?

The interesting thing about Classical Scaling is that the solution to this problem, for the case of using Euclidean squared distance, is analytic! Those interested readers may consult the mathematical details at the end of this chapter. However, the resulting algorithm is straightforward.

1.1 The Algorithm

Given a $P \times P$ distance matrix T :

1. Calculate the $P \times P$ matrix M with elements

$$M_{ab} = -\frac{1}{2}\left(T_{ab} - \frac{1}{P}\sum_a T_{ab} + \frac{1}{P^2}\sum_{ab} T_{ab} - \frac{1}{P}\sum_b T_{ab}\right) \tag{1.2}$$

2. Calculate the m largest eigenvalues $\lambda^i, i = 1, \ldots, m$ of $\mathbf{M}$, and their corresponding eigenvectors $\mathbf{e}^i$.

Figure 1: Classical scaling solution to representing 28 world cities on a two dimensional map, given only their intercity distances.

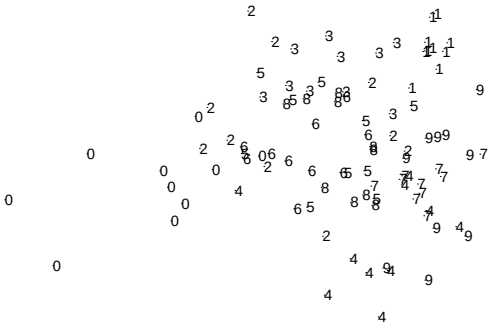

Figure 2: Classical scaling solution to representing digits in two dimensions. Note how some digits are more closely clustered together than others.

3. The points $\mathbf{y}^j, j = 1, \ldots P$ in the m dimensional space are then gives by the positions $y_i^j = \sqrt{\lambda^i} e_j^i$.

Example : intercity

We are given the intercity distances of 28 major cities in the world. This is therefore a 28×28 dimensional matrix T. The above algorithm is coded below in Matlab to form a 3 dimensional representation of the cities.

```
% Classical Scaling demo : assume that T is given
P = size(T,1);

S=-0.5*(T-repmat(sum(T,1),P,1)./P-repmat(sum(T,2),1,P)./P+repmat(sum(sum(T)),P,P)./P^2);

[V,Lambda] = eigs(S,3);

y = V*sqrt(Lambda);

plot3(y(:,1),y(:,2),y(:,3),'.'); rotate3d
```

The result is given in fig(1) where we have plotted only two of the three dimensions. Note how the representation is roughly correct from our experience of where cities are in the world.

Example : Digit data

We can also use classical scaling to reduce the dimension. I took 10 examples for each of the ten classes of digit – 100 datapoints therefore in total. Each digit is represented as a 784 dimensional vector. I then formed the 100×100 dimensional distance matrix T, and used classical scaling to plot the resulting 3 dimensional reconstructions. The results are given in fig(2).

2 Sammon Mapping

The Sammon mapping is a technique more general (and more widely used) than classical scaling. The idea is very simple. Given a $P \times P$ dissimilarity matrix d_{ij}, and a function $d(\mathbf{y}^i, \mathbf{y}^j)$ that measures the dissimilarlity of two vectors $\mathbf{y}^i$ and $\mathbf{y}^j$, we look to place objects in a space such that their dissimilarities are close to the given dissimilarities d_{ij}. An objective function to achieve this is

$$E = \frac{1}{\sum_{ij} d_{ij}} \sum_{i<j} \frac{\left(d_{ij} - d(\mathbf{y}^i, \mathbf{y}^j)\right)^2}{d_{ij}} \tag{2.1}$$

We minimise E with respect to the positions $\mathbf{y}^i, i = 1, \ldots p$. The division by d_{ij} is included in order to encourage the solution to deal with small dissimilarities accurately. (We do not divide by d_{ij}^2 since then small and large dissimilarities would be treated roughly equally). In order to train such a model, standard (non-linear) optimisation techniques can be employed.

For example, we could define dissimilarities as

$$d(\mathbf{y}^i, \mathbf{y}^j) = \left(\mathbf{y}^i - \mathbf{y}^j\right)^4. \tag{2.2}$$

Then, given a set of target dissimilarities d_{ij} we then need to arrange the vectors $\mathbf{y}^i$ to minimize the (weighted) difference between the given dissimilarities and those measured above. The parameters of the optimization are therefore the vectors $\mathbf{y}^i$ themselves.

Strictly speaking, the Sammon "Mapping" is not a mapping, since it does not yield a function that describes how general points in one space are mapped to another (it only describes how a limited set of points is related).

Making a Mapping

Given points $\mathbf{x}^i$ in a n-dimensional space (possibly very high dimensional) to represent them by points $\mathbf{y}^i$ in a m-dimensional space (possibly very low dimensional, say 2) in such a way that the separation between the points in the two spaces is roughly similar. One way to obtain this mapping is to parameterize the positions of the objects in the lower dimensional space

$$\mathbf{y} = f(\mathbf{x}; \mathrm{W}) \tag{2.3}$$

The distance then between two mapped points is a function of the parameters of the mapping W. The optimal parameters can then be found by optimization. The method Neuroscale is one such procedure.

3 A word of warning

It can be that very high dimensional datasets appear to lie on ring when plotted using visualisation methods. It may well be that the data really does have this kind of structure. However, in high dimensions, the distance matrix (between every pair of points) will be dominated by those points that are furthest apart. This will give the impression that most points are a long way from each other, and a ring or circular type two dimensional representation will likely be the visualisation solution. One should therefore bear in mind that global topological constraints on the data are unlikely to be accurately represented by these visualisation procedures, and one should be wary of reading too much into the precise structure of the visualisation.

4 Just for interest ...

Here we briefly describe the mathematics behind the solution of the Classical Scaling. If we consider a single element of the distance matrix, we have

$$T_{ab} = \mathbf{x}^a \cdot \mathbf{x}^a - 2\mathbf{x}^a \cdot \mathbf{x}^b + \mathbf{x}^b \cdot \mathbf{x}^b \tag{4.1}$$

For convenience, let us define a matrix

$$X_{ij} = x_j^i \tag{4.2}$$

Furthermore, define the matrix

$$E = XX^T \tag{4.3}$$

Then we can express one element of T as

$$T_{ab} = E_{aa} - 2E_{ab} + E_{bb} \tag{4.4}$$

If it were not for the terms E_{aa} and E_{bb}, life would be easy since, in that case, we would have a known matrix, T expressed as the outer product of an unknown matrix, X which would be easy to solve. What we need to do therefore is to express the unknown matrix elements E_{aa} and E_{bb} in terms of the known matrix T. In order to do this, we make the following extra assumption – the data has zero mean, $\sum_a x_i^a = 0$. Clearly, this does not affect the solution since it is only defined up to an arbitrary shift. In that case, $\sum_a E_{ab} = \sum_{ai} x_i^a x_i^b = 0$. Hence,

$$\sum_a T_{ab} = \sum_a E_{aa} - 2\sum_a E_{ab} + PE_{bb} \tag{4.5}$$

$$= \sum_a E_{aa} + PE_{bb} \tag{4.6}$$

This means that we could express E_{bb} in terms of T, if only we knew what $\sum_a E_{aa}$ is. But this can also be obtained by now summing over b:

$$\sum_{ab} T_{ab} = P\sum_a E_{aa} + P\sum_b E_{bb} \tag{4.7}$$

$$= 2P\sum_a E_{aa} \tag{4.8}$$

This means

$$PE_{bb} = \sum_a T_{ab} - \sum_a E_{aa} \tag{4.9}$$

$$= \sum_a T_{ab} - \frac{1}{2P}\sum_{ab} T_{ab} \tag{4.10}$$

so that

$$T_{ab} = \frac{1}{P}\sum_a T_{ab} - \frac{1}{P^2}\sum_{ab} T_{ab} + \frac{1}{P}\sum_b T_{ab} - 2E_{ab} \tag{4.11}$$

In other words, we can write

$$\left[XX^T\right]_{ab} = -\frac{1}{2}\left(T_{ab} - \frac{1}{P}\sum_a T_{ab} + \frac{1}{P^2}\sum_{ab} T_{ab} - \frac{1}{P}\sum_b T_{ab}\right) \tag{4.12}$$

The right hand side are elements of a now known matrix, T', for which we can find an eigen-decomposition

$$T' = V\Lambda V^T \tag{4.13}$$

where V is an orthogonal matrix and Λ is diagonal matrix containing the eigenvalues. Since each column of T' sums to zero, this matrix has at most rank $P-1$. A solution for the data position is given by taking the first r columns of $V\Lambda^{\frac{1}{2}}$, where r is the rank of T'. This means that if we have P vectors $\mathbf{x}^\mu, \mu = 1, \ldots, P$ based only on the Euclidean square distances between them, we can reconstruct a set of P objects in a $P-1$ dimensional space that has exactly the same distance structure.

If we wish to look for lower dimensional approximate reconstructions (in the sense that the distance matrix in the lower dimensional space will not exactly match the given distance matrix T) , we can simply take those columns of $V\Lambda^{\frac{1}{2}}$ corresponding to the largest eigenvalues of T'.