Visualisation

David Barber

What is Data Visualisation?

- Data is often very high dimensional meaning that we can't directly 'see' the data.
- It's difficult to get intuitions about the data since we can't 'see it'.
- In Data Visualisation we try to find a low dimensional representation (2 or 3 dimensions) so that we 'see something'.
- There is no 'correct' or 'perfect' visualisation. Every low dimensional representation will lose some information contained in the original high dimensional data.
- Such visualisations can be useful to see whether there might be clusters of datapoints, or which datapoints are in some sense 'similar' to another.
- Historically, methods such as PCA or Sammon 'mappings' were popular, but they are now less preferred.
- We tend to heuristically prefer representations that better preserve neighbourhood structure.
- Some visualisation methods (autoencoders) can be good but they are very expensive to train.

Setup

Each data vector $\mathbf{x}_n$ is in a high dimensional space. Given the set of datapoints

$$\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$$

we want to find a corresponding low dimensional (2 or 3) vector representation $\mathbf{y}_n$ for each $\mathbf{x}_n$ to give

$$\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$$

- We would like $\mathcal{Y}$ to preserve both the local and global structure in $\mathcal{X}$.
- Unfortunately, many ‘classical’ visualisation methods (Sammon mapping, Isomap, Locally Linear Embedding) don’t work that well on real-world data sets.
- We will focus on Stochastic Neighbour Embedding (SNE) and its ‘robust’ variant t-SNE which is one of the most popular current approaches.

Stochastic Neighbour Embedding (SNE)

We define an $N \times N$ Markov transition matrix:

$$p_{j|i} = \frac{\exp\left(-\left(\mathbf{x}_i - \mathbf{x}_j\right)^2 / (2\sigma_i^2)\right)}{\sum_{j \neq i} \exp\left(-\left(\mathbf{x}_i - \mathbf{x}_j\right)^2 / (2\sigma_i^2)\right)}, \qquad p_{i|i} = 0$$

Similarly we define

$$q_{j|i} = \frac{\exp\left(-\left(\mathbf{y}_i - \mathbf{y}_j\right)^2\right)}{\sum_{j \neq i} \exp\left(-\left(\mathbf{y}_i - \mathbf{y}_j\right)^2\right)}, \qquad q_{i|i} = 0$$

- The transition p describes the neighbourbood structure – how easily can one jump to other points from a given point.
- If we want to preserve this structure, we need to find $\mathcal{Y}$ such that q is approximately the same as p.
- Note that the $\mathbf{y}_n$ scale is arbitrary, so we have ‘fixed’ the variance to $1/\sqrt{2}$ in the y space.

SNE

- SNE minimises the KL divergence between each conditional distribution $p_i \equiv p_{\cdot|i})$:

$$E(\mathcal{Y}) = \sum_i \mathrm{KL}(p_i|q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- The optimisation is performed using gradient descent with parameters $\mathcal{Y}$.
- One can show (exercise) that

$$\frac{\partial E}{\partial \mathbf{y}_i} = 2 \sum_j \left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)\left(\mathbf{y}_i - \mathbf{y}_j\right)$$

- Criticisms about this approach are that the KL divergence is not symmetric.
- For example there is a large cost for using widely separated y points (small $q_{j|i}$) to represent nearby x points (large $p_{j|i}$).
- SNE therefore focuses on making sure that the local structure is correct, but loses fidelity in retaining the global structure.
- Another problem is that the Gaussian form of $p_{j|i}$ means that points which are far away will have negligible impact on the objective. We therefore need to make such points have an influence on the objective.

t-SNE

There are many potential ways to 'robustify' the SNE procedure. The t-SNE approach, which seems to work reasonably well, is:

- t-SNE uses a 'symmetric' loss

  $$E = \mathrm{KL}(p|q) = \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$$

  where the 'joint' distribution is defined

  $$p_{i,j} = \frac{p_{j|i} + p_{i|j}}{2N}, \qquad p_{i,i} = 0$$

- Note that this definition ensures that $p_i = \sum_j p_{i,j} > 1/(2N)$ which encourages each datapoint to have a significant effect on the cost function.
- The gradient is (exercise)

  $$\frac{\partial E}{\partial \mathbf{y}_i} = 4 \sum_j \left(p_{i,j} - q_{i,j}\right)\left(\mathbf{y}_i - \mathbf{y}_j\right)$$

- This works quite well, but there is an additional step that is also useful.

t-SNE

- If we use a student t-distribution (rather than a Gaussian) this has heavier tails and can therefore assign non-negligible mass to y points that are quite far apart. Defining a student t-distribution with a single degree of freedom,

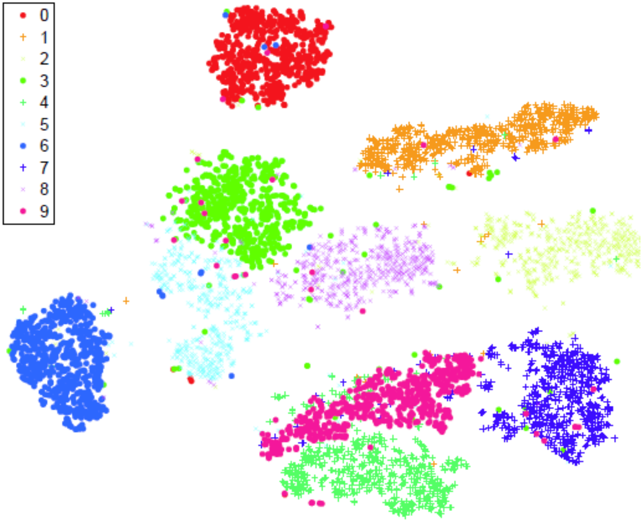
$$q_{i,j} = \frac{\left(1 + (\mathbf{y}_i - \mathbf{y}_j)^2\right)^{-1}}{\sum_{i \neq j} \left(1 + (\mathbf{y}_i - \mathbf{y}_j)^2\right)^{-1}}, \qquad q_{i,i} = 0$$

  Note that the sum in the denominator is over all pairs of distinct points.
- When $(\mathbf{y}_i - \mathbf{y}_j)^2$ is large, the '1' term becomes irrelevant and the q distribution will be essentially invariant with respect to the overall length scale.
- Hence, for all but the finest length scales, pairs of points that are very far apart will have a similar contribution to points that are reasonably far apart.
- The gradient is (exercise)

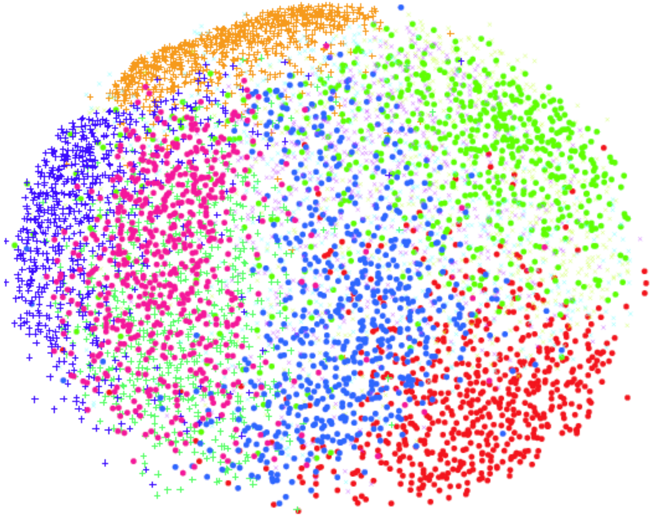
$$\frac{\partial E}{\partial \mathbf{y}_i} = 4 \sum_j \frac{(p_{i,j} - q_{i,j})}{1 + (\mathbf{y}_i - \mathbf{y}_j)^2} (\mathbf{y}_i - \mathbf{y}_j)$$

- Note that is unclear why the t-SNE authors do not define a student t-distribution on the original x datapoints as well.

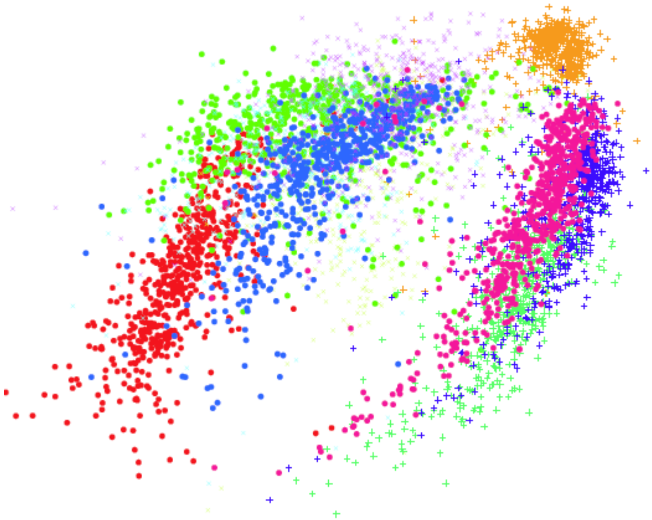
MNIST 2D visualisation: t-SNE



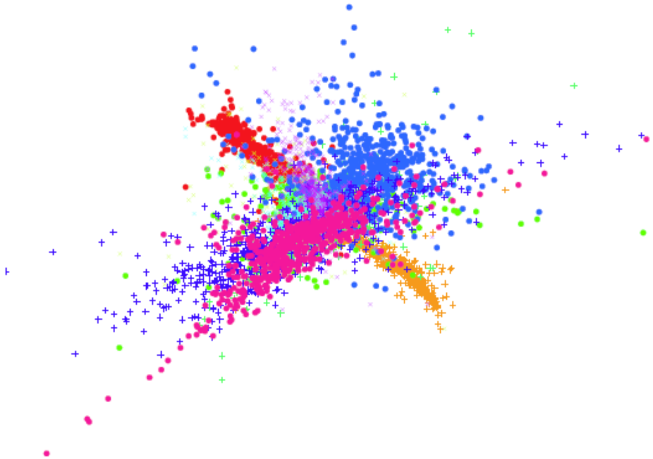
MNIST 2D visualisation: Sammon Mapping



MNIST 2D visualisation: Isomap



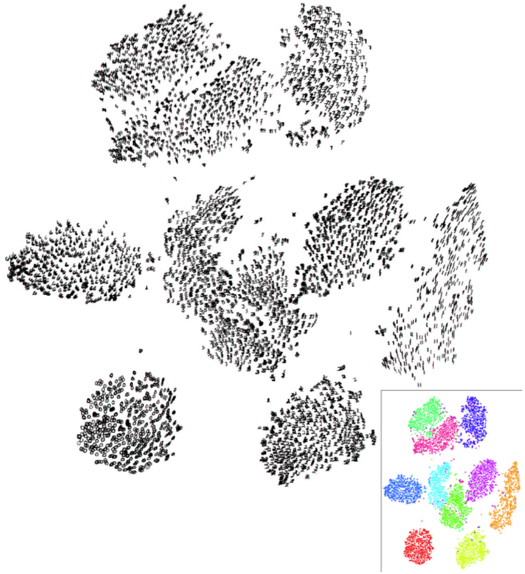
MNIST 2D visualisation: LLE



Large Datasets

- Like most visualisation methods, t-SNE has an $O(N^2)$ cost just to calculate the objective function since the distance between all pairs of points is calculated.
- For large datasets, this means that is very expensive to train t-SNE and related methods since each iteration of gradient descent requires an $O(N^2)$ calculation.
- A cheaper alternative is to first define a desired number of neighbours and calculate a graph of which are the nearest neighbours of each node (x datapoint).
- We then select (randomly) a small set of 'landmark' datapoints in x, indexed by i'. We can then calculate a new transition matrix for these datapoints as follows. Starting from i' we randomly sample j according to $p(j|i = i')$. We repeat sampling from this Markov chain until we land on another landmark $j' \neq i'$. We then repeat this procedure many times for landmark i' and then normalise to obtain the transition $p(j'|i)$. We then repeat this for each landmark i'.
- We then use $p(j'|i')$ in place of the original full $p(j|i)$ transition to find a visualisation for the chosen landmark points.
- Whilst it is expensive to calculate $p(j'|i')$, this only needs to be done once.

MNIST 2D visualisation: t-SNE



Visualisation of 6000 landmark points (using the random walk transition).

Using Autoencoders to visualise MNIST

Autoencoders (with a 2D bottleneck) can also be used to visualise data).

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

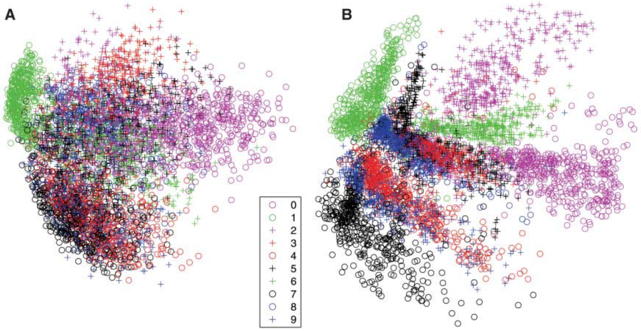


Figure from Hinton and Salakhutdinov Science 2006. The results can also be very good but training is quite slow and not practical for a ‘quick’ visualisation.