Dimensionality Reduction

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, UCL

Feb/May 2015

This practical session is an opportunity to explore data using the methods discussed in the morning. We have provided two data sets:

- `freyface.mat` – images of Brendan Frey's face in a variety of expressions: start with this one
- `kosstext.mat` – word counts from articles in a political discussion group

There is also MATLAB code to generate the artificial "swiss roll" data. Feel free to use other data sets if you have them.

We have also provided code to help with visualisation, as well as the authors' original implementations of lle and isomap.

All the data and code for this practical is available from: `http://www.gatsby.ucl.ac.uk/~maneesh/dimred/`

Matlab code for other dimensionality reduction techniques is available from:
`http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`

`freyface.mat`

Load the `mat` file. It contains a single object:

- `X` contains 1965 images of Brendan Frey's face. It is stored in integer format, although MATLAB does not support this very well. It is best to convert to double:

```
>> load freyface.mat
>> X = double(X);
```

- The function `showfreyface` renders the image(s) in its argument. Try `showfreyface(X(:,1:100))`.

Some things to try

PCA

- Find the eigenvectors of `X*X'/N`, both with and without first removing the mean:

```
>> N = size(X, 2);
>> [Vun, Dun] = eig(X*X'/N);
>> [lambda_un, order] = sort(diag(Dun));
>> Vun = Vun(:, order);
>> Xctr = X - repmat(mean(X, 2), 1, N);
>> [Vctr, Dctr] = eig(Xctr*Xctr'/N);
>> [lambda_ctr, order] = sort(diag(Dctr));
>> Vctr = Vctr(:, order);
```

  Which of these corresponds to PCA?

- Look at the eigenspectra (i.e. plot the `lambda`s). What might be a good choice for k? Is it easy to tell?

- Look at the top 16 eigenvectors in each case (the sorted output of `eig` above placed eigenvalues in increasing order, so this would be `showfreyface(V(:,end-15:end));` or you can use `eigs` to obtain just the top 16). Can you interpret them? How do they differ?
- Project the data onto the top two eigenvectors, and plot the resulting 2D points. You can use the function `explorefreymanifold` to explore this space. Does what you see make sense?

```
>> Y = V(:,end-1:end)' * X;
>> plot(Y(1,:), Y(2,:), '.');
>> explorefreymanifold(Y, X);
```

  (note: `V` here could be wither `Vun` or `Vctr`: how do the manifolds differ?)
- Try reconstructing a face from an arbitrary point in this space. That is, choose a point $\mathbf{y}$ within the space and compose the corresponding projected vector $\hat{\mathbf{x}}$ (see the lecture notes if you don't remember the relationship). Remember to add back in the mean when working with eigenvectors in the centred data. Does it look reasonable?
- Try adding noise to a face (`help randn` if you don't know how), projecting to the manifold (i.e. find the corresponding $\mathbf{y}$) and then reconstructing (i.e. find $\hat{\mathbf{x}}$).

MDS

- Construct the Gram matrix for the data after removing the mean.
- Use MDS to embed into a 2D space and plot the result (see PCA instructions above for help). Verify that you get the same result as PCA (there might well be sign differences).
- Look at the results of `svds(X, 2)`. How do they correspond to the results of PCA and MDA?
- Time (using `tic` and `toc` or `cputime`) the three equivalent algorithms: PCA, MDS and SVD. Which is fastest? Explore how this depends on the number of data dimensions and on the number of data.
- Can you think of different metrics to try with MDS?

Kernel PCA

- **[Advanced]** Try replacing the inner products with squared-exponential (i.e. Gaussian) kernel evaluations. What do you think of these results?

LLE

The function `lle` was provided by Roweis and Saul.

- This same face data was used in the LLE paper (see lecture notes). Can you reproduce the figure? (They used a 12-nearest-neighbour graph and 2 dimensions).
- Again, the function `explorefreymanifold` allows you to explore the projected space. Does it make more or less sense than the PCA one?
- Experiment with the neighbourhood size. How does it affect the results? Which neighbourhood gives the most interpretable embedding?
- Try reconstructing a face from an arbitrary point on the manifold. If you're a real MATLAB hacker, you can try to write an application that translates pointer-movement smoothly into appearance!

Isomap

The function `Isomap` was provided by Tenenbaum et al.

- The Isomap paper used a different face data set. Try running it on Brendan and compare to LLE. (Warning: it is not fast).
- Again, how does the neighbourhood affect the results?
- Can you reconstruct in this case?

More algorithms

Laurens van der Maaten's toolbox contains code for MVU, (t-)SNE and many others. You can download it from the link above and experiment.

More data

If you'd like, there's more data for you to experiment with. Or you can use your own if you'd like.

`swissroll.m`

This is code provided by Roweis and Saul to generate "swiss roll" data and run LLE. You may want to extract the data generation part of it and try running other algorithms on the same data set.

`kosstext.mat`

Load the `mat` file. There are two objects:

- `X` contains frequencies of is 6906 words in 3985 documents. It is stored as a sparse matrix.
- `words` is a cell array of 6906 words, which tells you which word is which. It will be useful only to interpret your results.

Some notes:

- `X` is large and sparse. It is best to run `svds` on it directly, rather than `eigs` on the covariance or Gram matrices.
- The data are not centred. If you centre them, the matrix will no longer be sparse. So it is probably best to run algorithms uncentered. Think about what effect this might have.
- You can use the word counts directly, but it is common practice to use a weighting function. Some choices:
  - normalised counts.

$$\widetilde{X}_{ij} = \frac{\text{counts of term } i \text{ in document } j}{\text{no. of terms in document } j} = \frac{X_{ij}}{\sum_k X_{kj}}$$

  - tf-idf(1) [term-frequency inverse-document-frequency]

$$\begin{aligned}\widetilde{X}_{ij} &= \frac{\text{counts of term } i \text{ in document } j}{\text{no. of terms in document } j} \frac{\text{no. of documents}}{\text{no. of documents containing term } i} \\ &= \frac{X_{ij}}{\sum_k X_{kj}} \frac{m}{\sum_j \mathbf{1}(X_{ik} > 0)}\end{aligned}$$

    where $\mathbf{1}()$ is the indicator function.
  - tf-idf(2) Apparently, some people add a 'log'.

$$\begin{aligned}\widetilde{X}_{ij} &= \frac{\text{counts of term } i \text{ in document } j}{\text{no. of terms in document } j} \log \frac{\text{no. of documents}}{\text{no. of documents containing term } i} \\ &= \frac{X_{ij}}{\sum_k X_{kj}} \log \frac{m}{\sum_j \mathbf{1}(X_{ik} > 0)}\end{aligned}$$