Unsupervised Learning: Dimensionality Reduction

Dr Liew How Hui

May 2022

Outline

Unsupervised Learning

When the data we have does not have any labels, it is impossible for us to apply supervised learning.

E.g. The DNA or RNA data of virus is such an example.

Unsupervised learning is often performed as part of an *pattern recognition*.

In Week 1, the EDA (with numeric summary, visualisation and correlation relation as basic tools) is used to analyse each feature and each pair of features.

It is difficult to assess the results obtained from unsupervised learning methods $\rightarrow$ no "indisputable" performance metric!

Unsupervised Learning

For an unlabelled $n \times p$ **numeric** data X, it may have the following patterns:

- n is much larger than p and the data is distributed along some subspace in $\mathbb{R}^p \rightarrow$ dimension reduction
- n is small and p is large, the data are correlated and the features are linear combinations of fewer features $\rightarrow$ dimension reduction
- Data are in closely-linked nonlinear distributed regions in high-dimensional feature space $\rightarrow$ nonlinear dimension reduction
- Data are in "clusters" $\rightarrow$ clustering (next topic)

Dimensionality Reduction

Target: unlabelled data X with a set of **numeric features** $X_1, \cdots, X_p$ measured on n observations.

"Dimensionality reductions" are a class of unsupervised learning methods which try to **extract** essential "dimension" information of the data to "characterise" the data.

Some dimensionality reduction algorithms that are PCA (linear), Kernel-PCA (nonlinear), MDS (linear), SOM, SNE (nonlinear), t-SNE (non-parametric/ nonlinear), Sammon mapping (nonlinear), Isomap (nonlinear), LLE (nonlinear), CCA (Canonical-Correlation Analysis, nonlinear), MVU (Maximum Variance Unfolding, nonlinear), Laplacian Eigenmaps (nonlinear), etc.

Dimensionality Reduction (cont)

The good news is that PCA and t-SNE (t-Distributed Stochastic Neighbour Embedding) currently the simplest tools to effectively visualise data in lower dimensions.

The PCA tries to find a global structure and this can lead to local inconsistencies, i.e. far away point can become nearest neighbours.

The t-SNE tries to preserve local structure, i.e. the low dimensional neighbourhood "representation" should be the same as original neighbourhood.

Dimensionality Reduction in R

- prcomp: PCA
- cmdscale: Classical MDS
- MASS::isoMDS: Nonmetric MDS
- ica::icafast: ICA
- Rtsne::Rtsne: t-SNE
- uwot::umap: Uniform Manifold Approximation and Projection
- ruta::autoencode: Autoencoders are neural networks that are trained to reconstruct their original inputs.
- phateR::phate: Potential of Heat-diffusion for Affinity-based Trajectory Embedding
- Rdimtools::do.fa: Factor Analysis
- Rdimtools::do.isomap, etc.

Dimensionality Reduction in Python

Dimensionality reduction functions from `sklearn.decomposition`:

- `PCA([n_components, copy, ...])`: Principal component analysis
- `IncrementalPCA([n_components, ...])`: Incremental PCA for large datasets.
- `SparsePCA([n_components, ...])`: Extract the set of sparse components that best reconstruct X.
- `MiniBatchSparsePCA([...])`: A variant of SparsePCA that is faster but less accurate.

Dimensionality Reduction in Python (cont)

`sklearn.decomposition` (cont):

- `KernelPCA([n_components, ...])`: An extension of PCA which achieves non-linear dimensionality reduction through the use of kernels
- `TruncatedSVD([n_components, ...])`: Dimensionality reduction using truncated SVD (aka latent semantic analysis / LSA).
- `SparseCoder(dictionary, *[, ...])` or `sparse_encode(X, dictionary, *)`: It is an estimator that can be used to transform signals into sparse linear combination of atoms from a fixed, precomputed dictionary such as a discrete wavelet basis.

Dimensionality Reduction in Python (cont)

`sklearn.decomposition` (cont):

- `DictionaryLearning([...])` or `dict_learning(X, n_components, ...)`: It is a matrix factorisation problem that amounts to finding a (usually overcomplete) dictionary that will perform well at sparsely encoding the fitted data.
- `dict_learning_online(X[, ...])`: Solves a dictionary learning matrix factorisation problem online.
- `MiniBatchDictionaryLearning([...])`: A faster, but less accurate version of the dictionary learning algorithm that is better suited for large datasets.

Dimensionality Reduction in Python (cont)

`sklearn.decomposition` (cont):

- `FactorAnalysis([n_components, ...])`: It is similar to PCA but can model the variance in every direction of the input space independently (heteroscedastic noise)
- `FastICA([n_components, ...])`: a fast algorithm for Independent Component Analysis.
- `NMF([n_components, init, ...])` or `non_negative_factorization(X)`: Non-Negative Matrix Factorisation.
- `LatentDirichletAllocation([...])`: Latent Dirichlet Allocation with online variational Bayes algorithm.

Dimensionality Reduction in Python (cont)

Dimensionality reduction functions from `sklearn.manifold`:

- `MDS([n_components, metric, n_init, ...])`: Multidimensional scaling seeks a low-dimensional representation of the data in which the distances respect well the distances in the original high-dimensional space.
- `Isomap(*[, n_neighbors, ...])`: Isometric Mapping Embedding, an extension of MDS or Kernel PCA.

Dimensionality Reduction in Python (cont)

`sklearn.manifold` (cont):

- `LocallyLinearEmbedding(*[, ...])` or `locally_linear_embedding(X, *, ...)`: A lower-dimensional projection of the data which preserves distances within local neighbourhoods.
- `SpectralEmbedding([n_components, ...])` or `spectral_embedding(adjacency, *[, ...])`: Spectral embedding for non-linear dimensionality reduction using Laplacian Eigenmaps.
- `TSNE([n_components, perplexity, ...])`: T-distributed Stochastic Neighbor Embedding converts affinities of data points to probabilities.

Outline

PCA

Given unlabelled data X of shape $n \times p$.

Motivations:

- Some columns can be a constant (e.g. image data): Can we remove it?
- Some columns are "colinear".
  - **Multicollinearity** is a phenomenon in which two or more predictors in a multiple regression model are highly correlated;
  - Perfect multicolinearity: One column is a multiple of another column $\Rightarrow$ covariance matrix has a determinant of 0;
  - Hight multicolinearity: covariance matrix has a determinant **close to 0**.

PCA (cont)

A major goal of PCA is to identify new predictors from the original predictors called 'factors' or 'latent variables' that are uncorrelated to each other by using the 'covariance matrix' of X.

Step 1: Shift the data to "column mean" / centre $\bar{x}$,

$$x_{\cdot j} \to x_{\cdot j} - \bar{x}_j =: \widetilde{X}$$

Step 2 (necessary in real-world problem but not in test/exam): If scaling is necessary (due to large variations in the variances), transform the data:

$$s_{\cdot j} = \sqrt{\frac{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}{n-1}} \Rightarrow x_{\cdot j} \to \frac{x_{\cdot j} - \bar{x}_j}{s_{\cdot j}} =: \widetilde{X}$$

PCA (cont)

Step 3: Compute the **covariance matrix** of X:

$$\Sigma := \text{Cov}(X) = \frac{1}{n-1}\widetilde{X}^T\widetilde{X}$$

The command in R is `S = cov(X)`.

Step 4: Compute the eigenvalues and eigenvectors of Σ. According to year 1 linear algebra's "spectral theorem", Σ is nonnegative definite and can be diagonalised into $\Sigma = Q\Lambda Q^T$, where Q are formed from the eigenvectors and the diagonal matrix Λ are formed from the corresponding eigenvalues.

PCA (cont)

A matrix Σ is said to be *non-negative definite* if

$$\mathrm{x}\Sigma\mathrm{x}^T \geq 0, \quad \forall \mathrm{x} = (x_1, ..., x_p), \quad x_1^2 + ... + x_p^2 = 1.$$

Spectral Theorem: A non-negative definite matrix can be decomposed into

$$\Sigma = \lambda_1 \mathrm{v}_1 \mathrm{v}_1^T + \cdots + \lambda_q \mathrm{v}_q \mathrm{v}_q^T, \quad q \leq \min\{n, p\}. \qquad (1)$$

Here, λ_i are the "eigenvalues" and v_i are the "eigenvectors". The eigenvectors are orthogonal to each other and $\mathrm{v}_i^T \mathrm{v}_i = 1$. Note that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q.$$

PCA (cont)

Step 5 (optional): Construct the principal components:

$$
\begin{aligned}
PC_1 &= e_{11}\tilde{X}_1 + e_{12}\tilde{X}_2 + \cdots + e_{1q}\tilde{X}_q, \\
PC_2 &= e_{21}\tilde{X}_1 + e_{22}\tilde{X}_2 + \cdots + e_{2q}\tilde{X}_q, \\
&\vdots \\
PC_q &= e_{q1}\tilde{X}_1 + e_{q2}\tilde{X}_2 + \cdots + e_{qq}\tilde{X}_q.
\end{aligned}
$$

where $\mathrm{v}_i = (e_{i1}, e_{i2}, ..., e_{iq})$.

PCA (cont)

Example 1: Given an unlabelled data with 15 observations and feature columns (but tabulate in horizontal form to save space). Find all the principal components.

	1	2	3	4	5	6	7	8
X_1	8.095	6.91	4.119	4.4	4.65	2.329	8.272	4.595
X_2	4.104	5.272	4.063	5.366	5.238	4.711	2.46	0.581
X_3	2.351	-2.827	-3.786	-0.261	-1.096	-1.456	-1.727	-1.292

	9	10	11	12	13	14	15
X_1	8.071	6.403	4.136	7.283	2.744	4.939	4.924
X_2	5.883	3.624	3.514	-1.301	3.584	3.024	2.754
X_3	0.938	2.949	2.918	-0.738	3.866	-0.803	5.154

PCA (cont)

Example 1: (cont)

Solution: It is possible for us to find the principal components using linear algebra. First, let us, calculate the column means:

$$\overline{x}_{\cdot j} = (5.45800, 3.52513, 0.27933)$$

and shift X to the "centre", i.e. $\widetilde{X} = x_{\cdot j} - \overline{x}_{\cdot j}$:

	1	2	3	4	5	6	7	8
X_1	2.637	1.452	-1.339	-1.058	-0.808	-3.129	2.814	-0.863
X_2	0.578867	1.746867	0.537867	1.840867	1.712867	1.185867	-1.065133	-2.944133
X_3	2.071667	-3.106333	-4.065333	-0.540333	-1.375333	-1.735333	-2.006333	-1.571333

	9	10	11	12	13	14	15
X_1	2.613	0.945	-1.322	1.825	-2.714	-0.519	-0.534
X_2	2.357867	0.098867	-0.011133	-4.826133	0.058867	-0.501133	-0.771133
X_3	0.658667	2.669667	2.638667	-1.017333	3.586667	-1.082333	4.874667

PCA (cont)

Example 1: (cont)

The covariance matrix of X is

$$\text{Cov}(X) = \frac{1}{15-1}\widetilde{X}^T\widetilde{X} = \begin{bmatrix} 3.700046 & -0.441594 & -0.261248 \\ -0.441594 & 3.63697 & -0.097637 \\ -0.261248 & -0.097637 & 6.82242 \end{bmatrix}$$

We need to find the eigenvalues of $\text{Cov}(X)$. This can be done using university linear algebra (determinant, r.e.f)

$$\lambda_i = 6.845301,\ 4.105652,\ 3.208484$$

Using R: eigen(S).

PCA (cont)

Example 1: (cont)

The corresponding eigenvectors are the **weighted vectors**

$$v_1 = \begin{bmatrix} -0.080068 \\ -0.019308 \\ 0.996602 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0.722438 \\ -0.689991 \\ 0.044673 \end{bmatrix}, \quad v_3 = \begin{bmatrix} 0.686784 \\ 0.723560 \\ 0.069195 \end{bmatrix}$$

The first, second and third principal components of X are:

$$PC_1 = -0.080068X_1 - 0.019308X_2 + 0.996602X_3$$
$$PC_2 = 0.722438X_1 - 0.689991X_2 + 0.044673X_3$$
$$PC_3 = 0.686784X_1 + 0.723560X_2 + 0.069195X_3$$

PCA (cont)

PCA in Python

```
pca = sklearn.decomposition.PCA(); pca.fit(X)
```

PCA in R

```
pca = prcomp(x, retx=TRUE, center=TRUE,
scale=FALSE, tol=NULL, rank=NULL, ...)
```

pca$center: $\overline{x}_{\cdot j}$

pca$sdev: $\sqrt{\lambda_i}$

pca$rotation: $[v_1, v_2, \cdots]$

pca$x: $[\widetilde{X}v_1, \widetilde{X}v_2, \cdots]$

pca$scale: $\sqrt{\frac{\sum_i (x_{ij} - \overline{x}_{\cdot j})^2}{n-1}}$

PCA (cont)

Example 2: (Final Exam Jan 2019, Q1(c))

You are given the following information:

- The data set consists of 3000 observations and 2 predictors, X_1 and X_2.
- The covariance matrix, C of X_1 and X_2 is

$$C = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix}$$

PCA (cont)

Example 2: (cont)

(i) Compute the eigenvalues, λ_1 and λ_2. For each of the eigenvalues computed, find the eigenvectors, v_1 and v_2. (7 marks)

(ii) First principal component, PC1 is the linear combination of predictors that has the maximum variance among all principal components. Based on your answer in (i), write the equation of first principal component for this data set. (2 marks)

PCA (cont)

Solution: To solve **Example** 2, we need to use the eigenvalues and eigenvectors from Basic Maths / Linear Algebra.
(i) Eigenvalues:

$$
\begin{aligned}
|C - \lambda I| &= \begin{vmatrix} 2.0 - \lambda & 0.8 \\ 0.8 & 0.6 - \lambda \end{vmatrix} \\
&= \lambda^2 - 2.6\lambda + 0.56 = 0 \\
\Rightarrow \lambda &= 0.236985, 2.363015
\end{aligned}
$$

[2 marks]

PCA (cont)

Eigenvectors:
$\lambda = 0.236985$:

$$\begin{bmatrix} 2.0 - 0.236985 & 0.8 \\ 0.8 & 0.6 - 0.236985 \end{bmatrix} \mathbf{v}_1 = 0 \Rightarrow$$

$$\mathbf{v}_1 = \frac{1}{\sqrt{0.8^2 + (-1.763015)^2}} \begin{bmatrix} 0.8 \\ -1.763015 \end{bmatrix} = \begin{bmatrix} 0.413216 \\ -0.910633 \end{bmatrix}$$

[2.5 marks]

PCA (cont)

$\lambda = 2.363015$:

$$\begin{bmatrix} 2.0 - 2.363015 & 0.8 \\ 0.8 & 0.6 - 2.363015 \end{bmatrix} \mathbf{v}_2 = 0$$

$$\Rightarrow \mathbf{v}_2 = \frac{1}{\sqrt{0.8^2 + (-(-0.363015))^2}} \begin{bmatrix} 0.8 \\ -(-0.363015) \end{bmatrix}$$

$$= \begin{bmatrix} 0.910633 \\ 0.413217 \end{bmatrix}$$

[2.5 marks]

(ii) PC1 is the linear combination of predictors with highest eigenvalue $\lambda_1 = 2.363015$, i.e.

$$PC_1 = 0.910633X_1 + 0.413217X_2.$$

PCA (cont)

A little bit more linear algebra theory: Let $Y = \frac{1}{\sqrt{n-1}}\tilde{X}$ and $\boxed{\lambda_i = \sigma_i^2}$. From (1), we have

$$Y^T Y = \sigma_1^2 \mathbf{v}_1 \mathbf{v}_1^T + \cdots + \sigma_q^2 \mathbf{v}_q \mathbf{v}_q^T$$

Since $\mathbf{v}_i^T \mathbf{v}_j = 0$ for $i \neq j$, $\mathbf{v}_i^T \mathbf{v}_i = 1$, we have

$$Y^T Y \mathbf{v}_i = (\sigma_1^2 \mathbf{v}_1 \mathbf{v}_1^T + \cdots + \sigma_q^2 \mathbf{v}_q \mathbf{v}_q^T)\mathbf{v}_i = \sigma_i^2 \mathbf{v}_i$$
$$\Rightarrow Y^T \left(\frac{1}{\sigma_i} Y \mathbf{v}_i\right) = \sigma_i \mathbf{v}_i.$$

PCA (cont)

Let $\mathrm{u}_i = \frac{1}{\sigma_i} Y \mathrm{v}_i$ and $U = [\mathrm{u}_1, ..., \mathrm{u}_q]$. Putting left and right column vectors into matrix form, we have

$$Y^T U = VS, \quad V = [\mathrm{v}_1, ..., \mathrm{v}_q], S = diag(\sigma_1, \ldots, \sigma_q).$$

From which we obtain the singular value decomposition (SVD):

$$Y^T = VSU^T \Rightarrow \boxed{Y = USV^T}.$$

Note that $S^T = S$, U is related to `pca$x`, V is related to `pca$rotation` and σ_i is `pca$sdev` mentioned earlier in **Example** 1.

PCA (cont)

The proportion of variation explained by the kth principal component is defined to be the eigenvalue, λ_k for that component divided by the sum of the eigenvalues. In other words, the kth principal component explains the following proportion of the total variation, which denote as proportion of variance, PVE_k, explained by PC_k:

$$PVE_k = \frac{\lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}, \quad k = 1, \cdots, p. \qquad (2)$$

PCA (cont)

A related quantity is the proportion of variation explained by the first k principal components is the cumulative proportion of variance explained until PC_k:

$$CPVE_k = \frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}, \quad k = 1, 2, \cdots, p. \quad (3)$$

In another word, $CPVE_k$ is the cummulation of PVEs.

PCA (cont)

If the proportion of variation explained by the first k principal components is large, then not much information is lost by considering only the first k principal components. In general, we only retain the first k principal components and balance the two conflicting desires:

1. To obtain the simplest possible interpretation, **we want k to be as small as possible**.
2. To avoid loss of information, we want

$$CPVE_k \approx 1.$$

PCA (cont)

How many principal components should we use? One of the methods to determine the number of principal components is to look at a *scree plot* (`https://en.wikipedia.org/wiki/Scree_plot`). With the eigenvalues ordered from largest to the smallest, a scree plot is the plot of "principal component" λ_k against k. The number of components is determined at the point beyond which the remaining eigenvalues are all relatively small and of comparable size.

PCA (cont)

Example 3: The dataset (`https://www.openml.org/d/509`) from the guide "Places Rated Almanac", by Richard Boyer and David Savageau, copyrighted and published by Rand McNally, uses nine rating criteria:(1) Climate and Terrain, (2) Housing, (3) Health Care & Environment, (4) Crime, (5) Transportation, (6) Education, (7) The Arts, (8) Recreation, (9) Economics.

Note that within the dataset, except for housing and crime, the higher the score the better. For housing and crime, the lower the score the better. Where some communities might rate better in the arts, other communities might rate better in other areas such as having a lower crime rate and good educational opportunities.

PCA (cont)

Example 3: (cont)
Suppose that principal component analysis has been implemented and the eigenvalues of each principal component are shown in the table below.

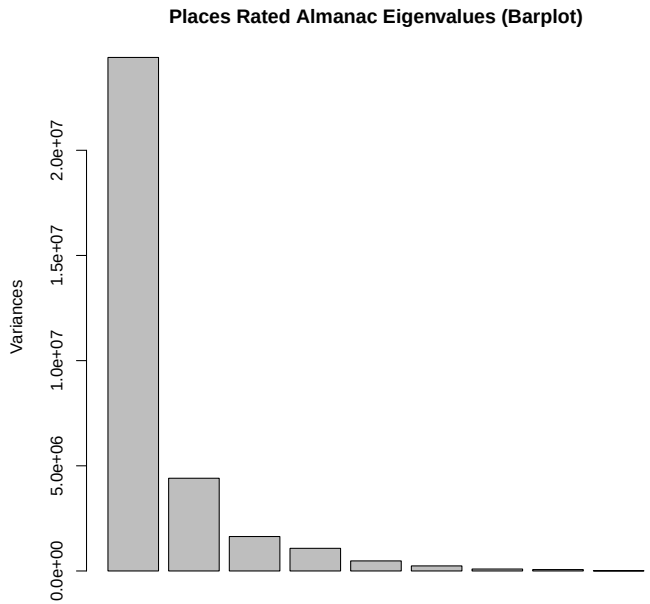
k	1	2	3	4	5	6
λ_k	24413669	4408005	1638040	1076356	478338.3	240851.8
k	7	8	9			
λ_k	92809.94	66995.9	10962.63			

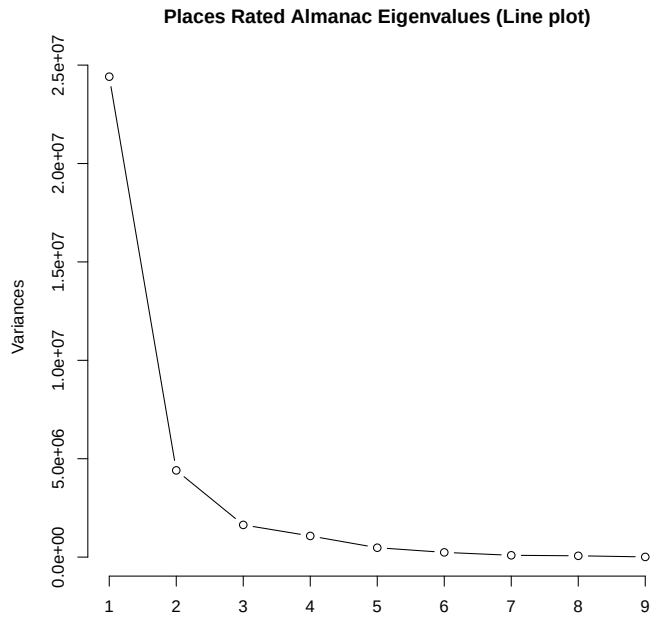
Compute the proportion of variance explained by each component and the cumulative proportion respectively. After that, construct a scree plot and determine the number of principal components to be considered.

PCA (cont)

Example 3: (cont)

There are two ways to represent "scree plot", one using "bar chart", the other using "line plot".





PCA (cont)

Example 4: The correlations between the principal components and the original variables for **Example** 3 are shown in the table below.

Variable	PC1	PC2	PC3
Climate	0.00642	-0.01546	0.00669
Housing	0.26914	-0.93721	0.08264
Health	0.17832	0.02054	-0.02776
Crime	0.02813	0.01090	-0.03761
Transportation	0.14930	-0.01876	-0.97153
Education	0.02519	0.00140	-0.04151
Arts	0.93086	0.28226	0.15103
Recreation	0.06982	-0.10385	-0.14957
Economy	0.02513	-0.17336	-0.01274

PCA (cont)

Example 4: (cont) Analysis of the components (+ve or −ve signs are related).

First Principal Component - PC1

The first principal component is **strongly correlated** with the original variable `Arts`. It increases with all features, in particular, `Arts`. Based on the correlation of 0.93086, the first principal component is primarily a measure of the `Arts` and it has $0.93086 \times PVE_1 = 70\%$ influence. It would follow that communities with high values tend to have a lot of arts available, in terms of museums, fine arts, public radio stations, public television stations, universities offering degrees in the arts, symphony orchestras, theatres, opera companies, dance companies, public libraries etc. Whereas communities with small values would have very few of these types of opportunities.

PCA (cont)

Example 4: (cont)

Second Principal Component - PC2

The second principal component decreases with the feature `Housing` and increases with the feature `Arts`. This component can be viewed as a measure of things utility bills, property taxes, mortgage payments related to `Housing` have 12.7% ($0.93721 \times PVE_2$) significance.

Third Principal Component - PC3

The third principal component decreases mainly with increasing `Transportation` and increases slightly with increasing `Arts`. This suggests that places with bad transportation has 4.9% ($0.97153 \times PVE_3$) significance.

PCA (cont)

Example 5: (Final Exam May 2019, Q1(c))
An investigation is carried out to examine the eating habit of citizens in each state in Malaysia. A total of 8 types of food had been investigated — the average consumption of each type of food (grams) per person per week in each state in Malaysia was recorded. The 8 types of food are listed below:

- x_1 = Fish
- x_2 = Meat
- x_3 = Grain
- x_4 = Dairy
- x_5 = Beans
- x_6 = Egg
- x_7 = Vegetables
- x_8 = Fruit

PCA (cont)

Example 5: (cont)
For advanced analysis on the eating habit, the analyst would like to reduce the dimension of the data. Therefore, principal component analysis has been performed. Eigenvalues computed from the principal component analysis are

$$\lambda = [0.0342, 0.6432, 2.3664, 0.5869, 1.1894, 0.0032, 5.6379, 0.0179]^T$$

PCA (cont)

Example 5: (cont)

(i) State the variance explained by each principal component. With a targeted cumulative proportion of variance explained (CPVE) of 85%, state the number of principal components to be considered. (4 marks)

(ii) Given that the eigenvector for PC1 is

$$e_1 = \begin{bmatrix} 0.174 & -0.626 & -0.146 & 0.667 & -0.249 & 0.743 & 0.598 & 0.484 \end{bmatrix}.$$

Interpret the principal component results with respect to the types of food. You should provide explanation on which types of food are the most and the least contributed to PC1, as well as the correlation between PC1 and that variable. (5 marks)

PCA (cont)

Solution: (i) Variance explained = eigenvalues in descending order.

PC	λ	PVE	CPVE
1	5.6379	0.5380	0.5380
2	2.3664	0.2258	0.7638
3	1.1894	0.1135	0.8773
4	0.6432	0.0614	0.9387
5	0.5869	0.0560	0.9947
6	0.0342	0.0033	0.9980
7	0.0179	0.0017	0.9997
8	0.0032	0.0003	1.0000
Total	10.4791		

With targeted CPVE of 85%, 3 principal components (PC1, PC2 and PC3), with CPVE of 87.73%, should be considered.

PCA (cont)

Solution: (ii)
The correlation between PC1 and variables are shown below:

Variable, x_i	Type of Food	e_{1i}
x_1	Fish	0.174
x_2	Meat	-0.626
x_3	Grain	-0.146
x_4	Dairy	0.667
x_5	Beans	-0.249
x_6	Egg	0.743
x_7	Vegetables	0.598
x_8	Fruit	0.484

PCA (cont)

Solution: (ii)

Egg is the most contributed to PC1 as it has the highest absolute in the eigenvalues which is 0.743. Egg is positively correlated with PC1 since it has a positive value. This means that PC1 will be higher if the egg consumption increases.

Grain is the least contributed to PC1 as it has the lowest absolute in the eigenvalues which is -0.146. Grain is negatively correlated with PC1 since it has a negative value. This means that PC1 will be lower if the grain consumption increases.

PCA (cont)

A *biplot* is a 2D "scatter plot" of PC_1 and PC_2
In the above, we required that $n > p$. However, for the case of $n \leq p$, we just need to transpose the data X and all the decomposition can be carried out.
For the case $n \leq p$, it corresponds to data that are collected on a **large number of variables** from a single population.
There would be too many pairwise correlations between the variables to consider, i.e. there are $\binom{p}{2} = \frac{p(p-1)}{2}$ correlations or scatter plots to be examined. E.g. when $p = 10$, there are 45 scatter plots!

PCA (cont)

A *biplot* is a 2D "scatter plot" of the first and the second principle components. In the above, we required that $n > p$. However, for the case of $n \leq p$, we just need to transpose the data X and all the decomposition can be carried out.

For the case $n \leq p$, it corresponds to data that are collected on a **large number of variables** from a single population. With a large number of variables, the dispersion matrix may be too large to study and interpret properly. There would be too many pairwise correlations between the variables to consider, i.e. there are $\binom{p}{2} = \frac{p(p-1)}{2}$ correlations or scatter plots to be examined. For example, when $p = 10$, there are 45 scatter plots! Hence, graphical displays may also not be particularly helpful when the data set is very large.

PCA (cont)

Example 6: Consider the "EATING IN THE UK" from `http://setosa.io/ev/principal-component-analysis/`. This is a **17-dimension** example. The table below is the average consumption of 17 types of food in grams per person per week for every country in the UK.

PCA (cont)

Example 6: (cont)

	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66
Carcass meat	245	227	242	267
Other meat	685	803	750	586
Fish	147	160	122	93
Fresh potatoes	720	874	566	1033
Processed potatoes	198	203	220	187
Fats and oils	193	235	184	209
Sugars	156	175	147	139
Fresh Veg	253	265	171	143
Other Veg	488	570	418	355
Processed Veg	360	365	337	334
Fresh fruit	1102	1137	957	674
Cereals	1472	1582	1462	1494
Beverages	57	73	53	47
Soft drinks	1374	1256	1572	1506
Alcoholic drinks	375	475	458	135
Confectionery	54	64	62	41

PCA (cont)

Example 6: (cont) The PVEs and CPVEs are given below.

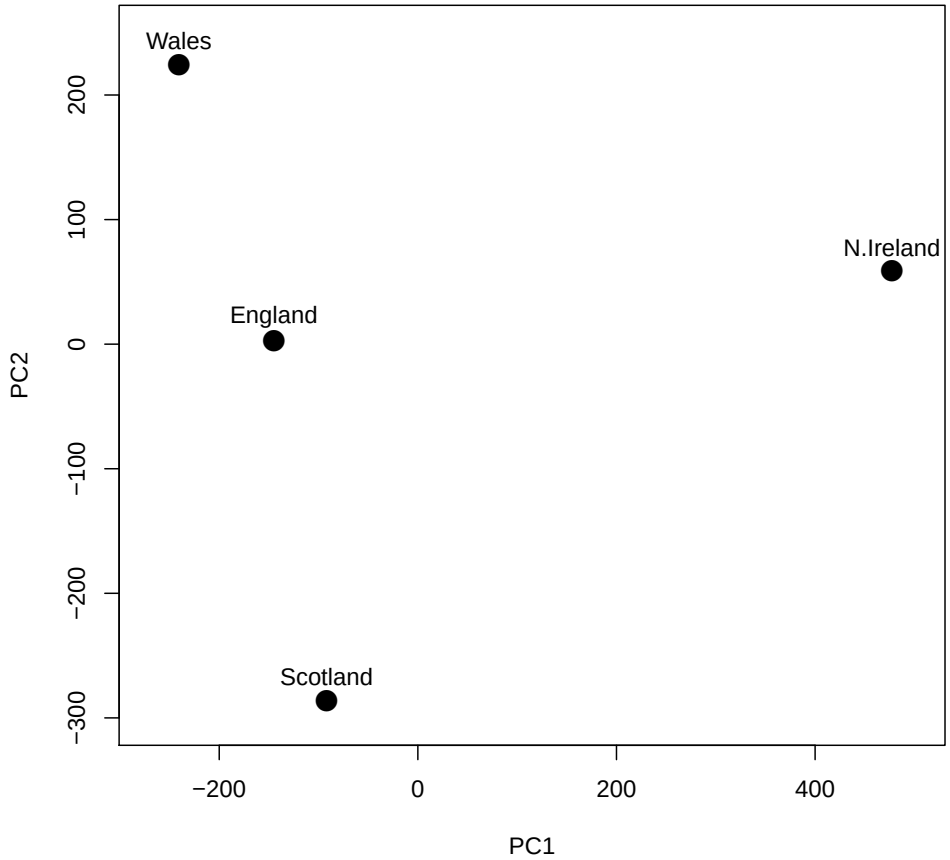
```
Importance of components:
                            PC1      PC2      PC3         PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

The script to obtain the PVEs & CPVEs as well as the biplot is listed below.

```
1  # From https://bioboot.github.io/bggn213_f17/class-material/UK_food_p
2  x = read.csv("UK_foods.csv")
3  rownames(x) = x[,1]
4  x = x[,-1]           # Remove first column
5  ### Print x as a nice table (an alternative is R's View() function)
6  #knitr::kable(x, caption="The full UK foods data table")
7  pca = prcomp(t(x))
8  print(summary(pca))  #print(pca$sdev^2)
9  plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500),
0    ylim=c(-300,250), pch=16, cex=2)
1  text(pca$x[,1], pca$x[,2]+19, colnames(x))
2  # biplot(pca)
```

PCA (cont)

Example 6: (cont)
The biplot:



PCA (cont)

Example 6: (cont)
From the biplot, we detect that Northern Ireland is a major outlier. Once we go back and look at the data in the table, this makes sense: the Northern Irish eat way more grams of fresh potatoes and way fewer of fresh fruits, cheese, fish and alcoholic drinks. It's a good sign that structure we've visualised reflects a major fact of real-world geography: Northern Ireland is the only of the four countries not on the island of Great Britain.

PCA (cont)

Example 7: (https://en.wikipedia.org/wiki/Eigenface)

- When viewed as vectors of pixel values, face images are extremely high-dimensional. E.g. 300x200 image = 60,000 dimensions (Slow and lots of storage)
- Eigenface idea: construct a low-dimensional linear subspace that contains most of the face images possible (possibly with small errors).
- Intuitively, the recognition process with the eigenface method is to project query images into the face-space spanned by eigenfaces calculated, and to find the closest match to a face class in that face-space.

PCA (cont)

Example 7: (cont) Recognition process pseudo-code:

- Given input image vector $\mathrm{x} \in \mathbb{R}^p$, the mean image vector from the database M, calculate the weight of the kth eigenface as:

$$w_k = V_k^T(\mathrm{x} - M)$$

Then form a weight vector
$W = [w_1, w_2, ..., w_k, ..., w_n]$.

- Compare W with weight vectors W_m of images in the database. Find the Euclidean distance.

$$d = \|W - W_m\|^2$$

PCA (cont)

Example 7: (cont)

Recognition process pseudo-code (cont):

- If $d < \epsilon_1$, then the mth entry in the database is a candidate of recognition.
- If $\epsilon_1 < d < \epsilon_2$, then x may be an unknown face and can be added to the database.
- If $d > \epsilon_2$, x is not a face image.

PCA (cont)

PCA has been very successful in a lot of dimensional reduction tasks. For example latent semantic analysis uses PCA, population structure in the genetic data from different geographical locations can be inferred using PCA etc.

Despite it's popularity, PCA has some obvious shortcomings, most notably is the assumption that data lie on a linear subspace. If the data lie along a curled plane, e.g. a swiss roll embedded in 3D Euclidean space, PCA wouldn't be able to find the 2-dimensional representation even though the data is obviously 2d.

PCA (cont)

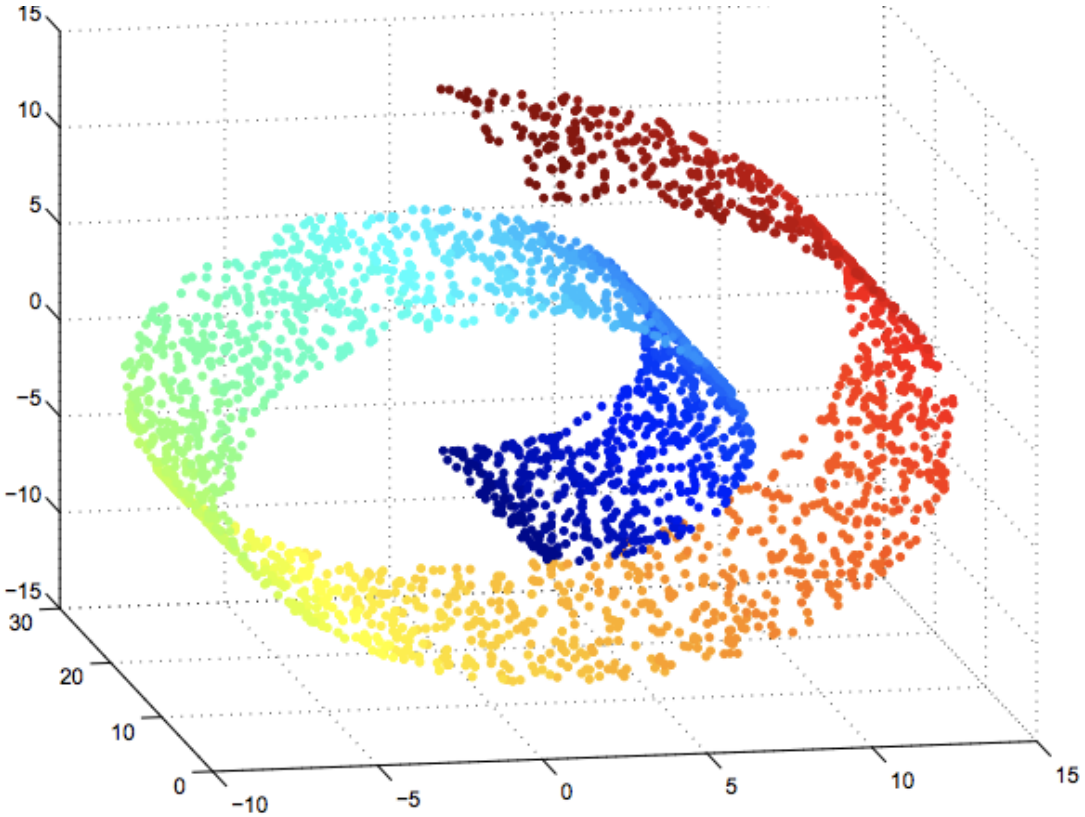


Figure 1: A curled plane: the swiss roll.

Generated using Rdimtools::aux.gensamples(500, dname="swiss")

Outline

Multidimensional Scaling (MDS)

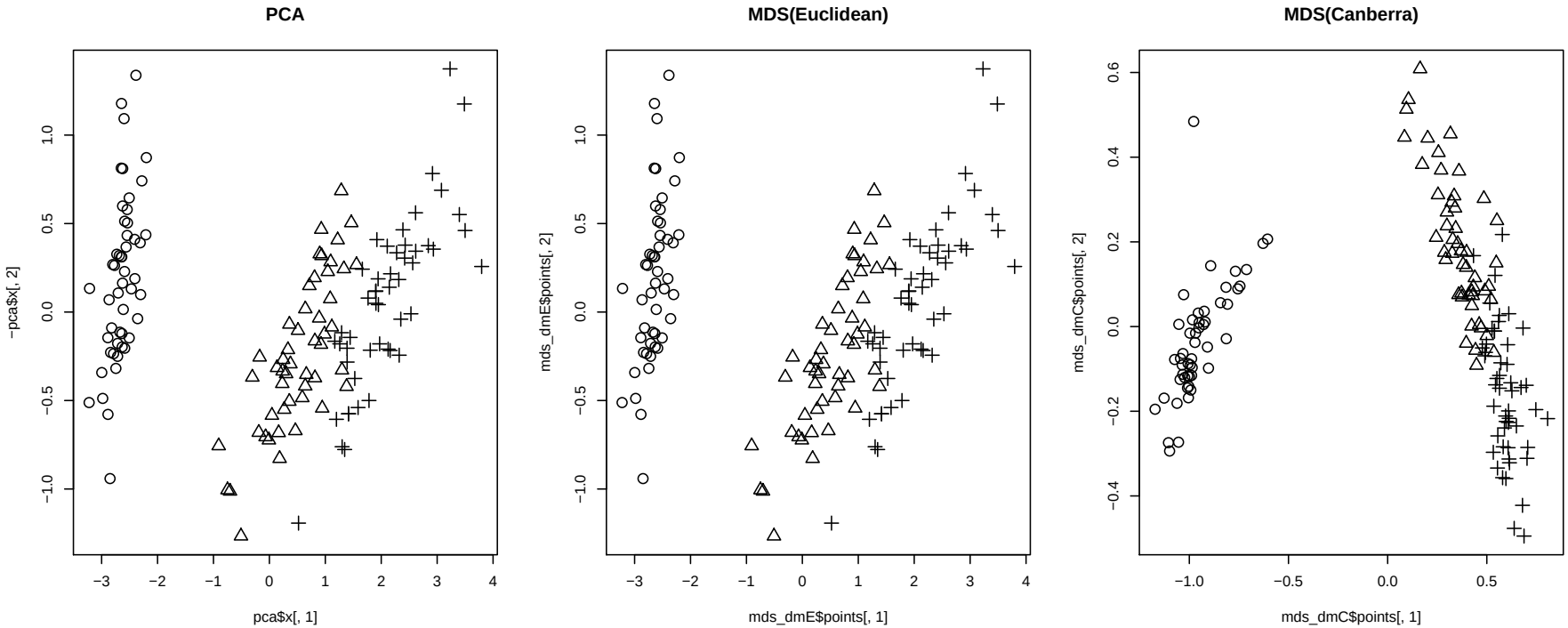
MDS (also known as Principal Coordinates Analysis, PCoA) is an algorithm that given a distance matrix with the distances between each pair of objects in a set, and a chosen number of dimensions, k, places each object into k-dimensional space such that the between-object distances are preserved as well as possible. MDS is most often used as a visualization tool. It is best suited to the problem that involve real distances, i.e. Manhattan distances or geometry. It yields the same result as PCA when Euclidean distances are used.

The R implementation in base R is listed below.

```
cmdscale(d, k = 2, eig = FALSE, add = FALSE,
   x.ret = FALSE, list. = eig || add || x.ret)
```

MDS (cont)

Example 8: The following are the biplots for PCA and MDS of iris flower dataset.



MDS (cont)

Advantages:

- flexible as any distance metric can be used
- preserves global structures

Disadvantages:

- computationally demanding and extremely inefficient for large numbers of observations
- suffers from crowding in the presence of large number of observations

Self-Organising Map (SOM)

SOM is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically 2D), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction.

SOM differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighbourhood function to preserve the topological properties of the input space.

Self-Organising Map (cont)

It is available in R but the diagram is difficult to use and interpret.

```
library(class)
SOM(data, grid = somgrid(), rlen = 10000,
    alpha, radii, init)
```

The algorithm depends on `somgrid` and returns an object with `code` (containing the column names) and `grid` which is the SOM 2D representation.

```
somgrid(xdim = 8, ydim = 6,
    topo = c("rectangular", "hexagonal"))
```

Self-Organising Map (cont)

A slightly easier to use SOM is provided by `kohonen`.

```
som(X, ...)
xyf(X, Y, ...)
supersom(data, grid=somgrid(), rlen = 100, alpha = c(0.05, 0.01),
         radius = quantile(nhbrdist, 2/3),
         whatmap = NULL, user.weights = 1, maxNA.fraction = 0L,
         keep.data = TRUE, dist.fcts = NULL,
         mode = c("online", "batch", "pbatch"), cores = -1, init,
         normalizeDataLayers = TRUE)
```

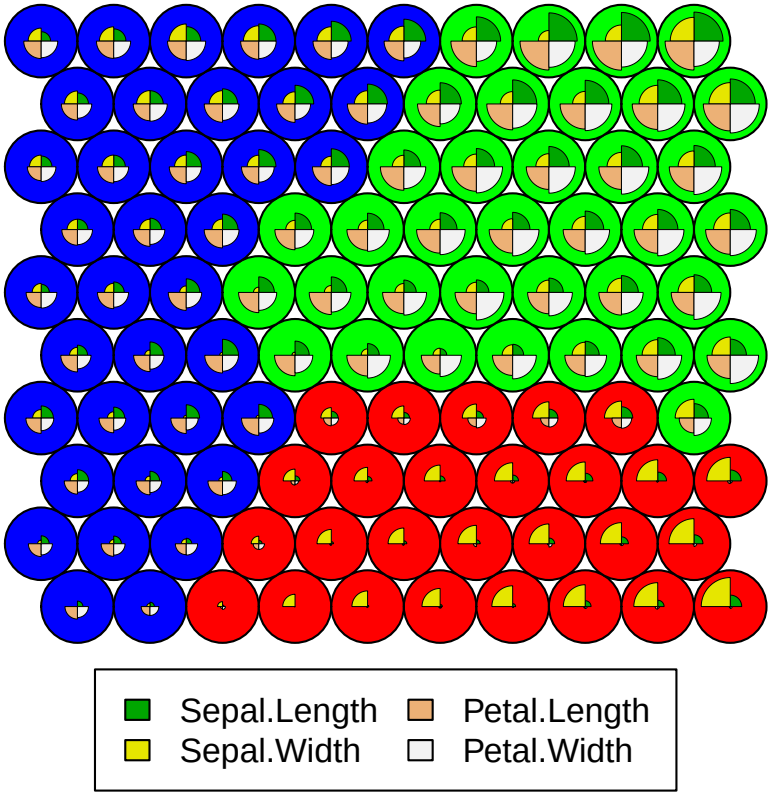
```
# https://rpubs.com/inayatus/som
# https://rstudio-pubs-static.s3.amazonaws.com/492825_4742a19ab5464d3
library(kohonen)
d.grid = somgrid(xdim = 10, ydim = 10, topo = "hexagonal")
set.seed(100)
model = som(as.matrix(iris[,1:4]), d.grid, rlen = 500,
            radius = 2.5, keep.data = TRUE,
            dist.fcts = "euclidean")
#plot(model, type="mapping", pchs=19, shape="round")
#plot(model, type="codes", main="Codes Plot", palette.name=rainbow)
#plot(model, type="changes")
```

Self-Organising Map (cont)

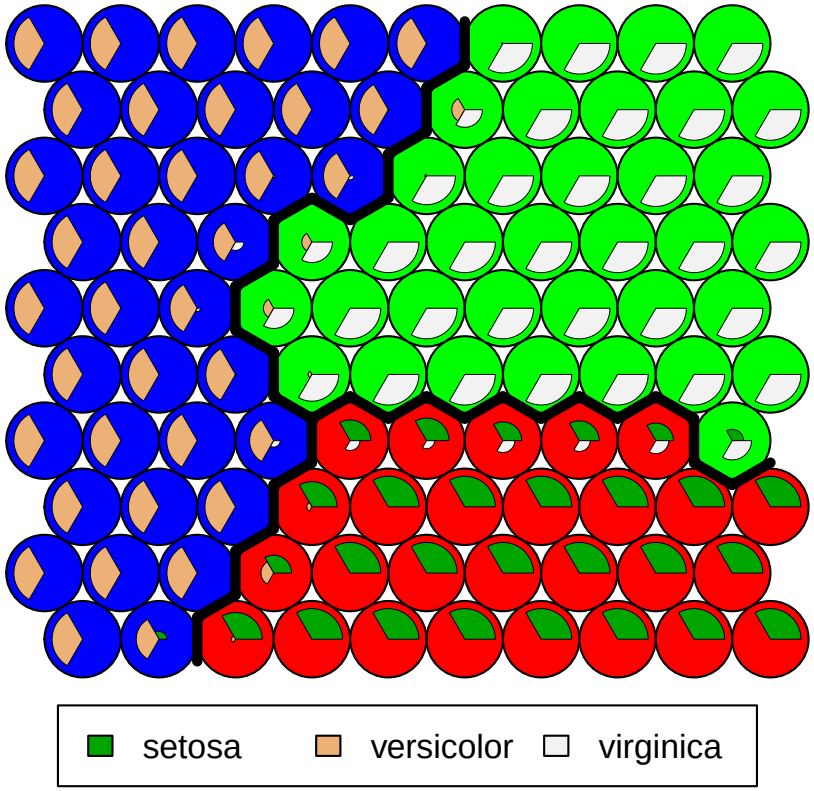
```
# split data
set.seed(100)
idx = sample(nrow(iris), nrow(iris)*0.8)
train = iris[ idx,]
test  = iris[-idx,]
# scaling data
trainX = scale(train[,-5])
testX  = scale(test[,-5], center = attr(trainX, "scaled:center"))
# make label
train.label = factor(train[,5])
test.label  = factor(test[,5])

cl = xyf(trainX, classvec2classmat(train.label), d.grid, rlen=500)
pred = predict(cl, newdata=list(independent=testX,
                                dependent=test.label))
table(Predict = pred$predictions[[2]], Actual = test.label)

pdf("s11_pca_som.pdf", width=10,height=5)
clust = kmeans(cl$codes[[2]], 3)
par(mfrow = c(1,2))
plot(cl, type = "codes", main = c("Unsupervised SOM\nCodes Plot",
     "Supervised SOM"), bgcol = rainbow(3)[clust$cluster])
add.cluster.boundaries(cl, clust$cluster)
```

Self-Organising Map (cont)

Unsupervised SOM
Codes Plot



Supervised SOM



Outline

SNE and t-SNE

PCA is a linear algorithm and it cannot "project" the nonlinear relationship between features to low dimensional space well. On the other hand, *SNE (Stochastic Neighbour Embedding)* and *t-SNE (t-Distributed SNE)* use probability distributions with random walk on neighbourhood graphs to identify and try to preserve the "nonlinear" structure of the data.

SNE and t-SNE (cont)

SNE converts the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities, i.e.

$$\mathbb{P}(\mathsf{X} = \mathsf{x}_j | \mathsf{X} = \mathsf{x}_i) = \frac{\exp(-\frac{\|\mathsf{x}_i - \mathsf{x}_j\|^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|\mathsf{x}_i - \mathsf{x}_k\|^2}{2\sigma_i^2})} =: p_{j|i}$$

where σ_i is the variance of the Gaussian that is centred on the data point x_i.

SNE and t-SNE (cont)

Assume that the "projected" high-dimensional points x_i and x_j to a low-dimension "space" is y_i and y_j respectively. The conditional probability of y_i and y_j is

$$\mathbb{P}(\mathrm{Y} = \mathrm{y}_j | \mathrm{Y} = \mathrm{y}_i) = \frac{\exp(-\|\mathrm{y}_i - \mathrm{y}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathrm{y}_i - \mathrm{y}_k\|^2)} =: q_{j|i}. \tag{4}$$

Note that, we define

$$\mathbb{P}(\mathrm{Y} = \mathrm{y}_i | \mathrm{Y} = \mathrm{y}_i) = 0, \quad \text{for all } i$$

since we only want to model pair-wise similarity.

SNE and t-SNE (cont)

To measure the minimisation of sum of difference of conditional probability, SNE minimises the sum of `https://en.wikipedia.org/wiki/Kullback-Leibler_divergence` over all data points using a gradient descent method:

$$C = \sum_i \sum_j p_{j|i} \ln \frac{p_{j|i}}{q_{j|i}}. \tag{5}$$

In other words, the SNE cost function (5) focuses on retaining the local structure of the data in the map (for reasonable values of the variance of the Gaussian in the high-dimensional space, σ_i).

SNE and t-SNE (cont)

t-SNE tries to minimise symmetric version of the SNE cost function:

$$C = \sum_i \sum_j p_{ij} \ln \frac{p_{ij}}{q_{ij}}, \; p_{ij} := \frac{p_{j|i} + p_{i|j}}{2n}, \; q_{ij} := \frac{q_{j|i} + q_{i|j}}{2n}$$

and employs a **heavy-tailed distribution in the low-dimensional space**:

$$\mathbb{P}(\mathrm{Y} = \mathrm{y}_j | \mathrm{Y} = \mathrm{y}_i) = \frac{(1 + \|\mathrm{y}_i - \mathrm{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathrm{y}_i - \mathrm{y}_k\|^2)^{-1}} =: q_{j|i} \tag{6}$$

to alleviate both the crowding problem & the optimisation problems of SNE.

SNE and t-SNE (cont)

In R, t-SNE is implemented in `tsne::tsne` and `Rtsne::Rtsne` (based on C++, much faster). In Python, t-SNE is implemented as `sklearn.manifold.TSNE`. Therefore many other manifold unsupervised learning methods in `sklearn.manifold` which are not explored, such as `Isomap`, `LocallyLinearEmbedding` (contain a lot of variance: `method='standard'`, `'hessian'`, `'modified'` or `'ltsa'`), `SpectralEmbedding`, `MDS` (multi-dimensional scaling), etc.

SNE and t-SNE (cont)

Example 9: An R script to compare PCA and t-SNE on the MNIST data is listed below.

```
# https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r
library(Rtsne)  # Uses Barnes-Hut-TSNE algorithm instead of the slowe
train = read.csv("mnist_train.csv")
X = train[,-1]
train$label = as.factor(train$label)
colours = rainbow(length(unique(train$label)))
names(colours) = unique(train$label)

# https://www.youtube.com/watch?v=xPBO-MMxIoQ[R Tutorial: PCA and t-S
pca = prcomp(train[,-1], rank=2) # project data to first two PCs only
plot(pca$x[,1:2], pch=as.character(train$label),
                  col=colours[train$label], main="Biplot")

# Takes a long time to calculate: dim(X) = 10000 x 784
tsne = Rtsne(X, dims=2, perplexity=30, verbose=TRUE, max_iter=500)
time.taken = system.time(Rtsne(X, dims=2, perplexity=30,
                               verbose=TRUE, max_iter=500))
plot(tsne$Y, t='n', main="tsne")
text(tsne$Y, labels=train$label, col=colours[train$label])
```

SNE and t-SNE (cont)

Example 9: (cont)