Unsupervised Learning: Dimensionality Reduction

Héctor Corrada Bravo

University of Maryland, College Park, USA

CMSC 643: 2018-10-29

Unsupervised Learning

Unsupervised data: characterize patterns in predictor space where observation measurements are represented.

Mathematically, characterize $Pr(X)$ over p-dimensional predictor space.

Clustering methods assume that this space $Pr(X)$ can be partitioned into subspaces containing "similar" observations.

Unsupervised Learning: Dimensionality Reduction

Dimensionality reduction: assume observations can be represented in a space with dimension much lower than p.

We will see two general strategies for dimensionality reduction:

- data transformations into spaces of smaller dimension that capture global properties of a data set X,
- data embeddings into lower dimensional spaces that retain local properties of a data set X.

Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction method.

Goal: *embed data in high dimensional space (e.g., observations with a large number of variables), onto a small number of dimensions*.

Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction method.

Goal: *embed data in high dimensional space (e.g., observations with a large number of variables), onto a small number of dimensions*.

Most frequent use is in Exploratory Data Analysis and visualization

Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction method.

Goal: *embed data in high dimensional space (e.g., observations with a large number of variables), onto a small number of dimensions*.

Most frequent use is in Exploratory Data Analysis and visualization

Also be helpful in regression (linear or logistic) where we can transform input variables into a smaller number of predictors for modeling.

Principal Component Analysis

Mathematically, the PCA problem is:

Given:

- Data set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i$ is the vector of p variable values for the i-th observation.

Return:

- Matrix $[\phi_1, \phi_2, \ldots, \phi_p]$ of *linear transformations* that retain *maximal variance*.

Principal Component Analysis

Think of the first vector ϕ_1 as a linear transformation that embeds observations into 1 dimension:

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p$$

where ϕ_1 is selected so that the resulting dataset $\{z_1, \ldots, z_n\}$ has *maximum variance*.

Principal Component Analysis

In order for this to make sense mathematically:

- data has to be centered, i.e., each X_j has mean equal to zero
- transformation vector ϕ_1 has to be normalized, i.e., $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

Principal Component Analysis

Find ϕ_1 by solving optimization problem:

$$\max_{\phi 11, \phi_{21}, \ldots, \phi_{p1}} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2$$

$$\text{s. t.} \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Principal Component Analysis

Conceptually: *maximize variance* but *subject to normalization constraint*.

The second transformation ϕ_2 is obtained next solving a similar problem with the added constraint that ϕ_2 **is orthogonal** to ϕ_1.

Principal Component Analysis

Taken together $[\phi_1, \phi_2]$ define a pair of linear transformations of the data into 2 dimensional space.

$$Z_{n\times 2} = X_{n\times p}[\phi_1, \phi_2]_{p\times 2}$$

Principal Component Analysis

Each of the columns of the z matrix are called *Principal Components*.

The units of the PCs are *meaningless*.

In particular, comparing numbers *across* PCs doesn't make mathematical sense.

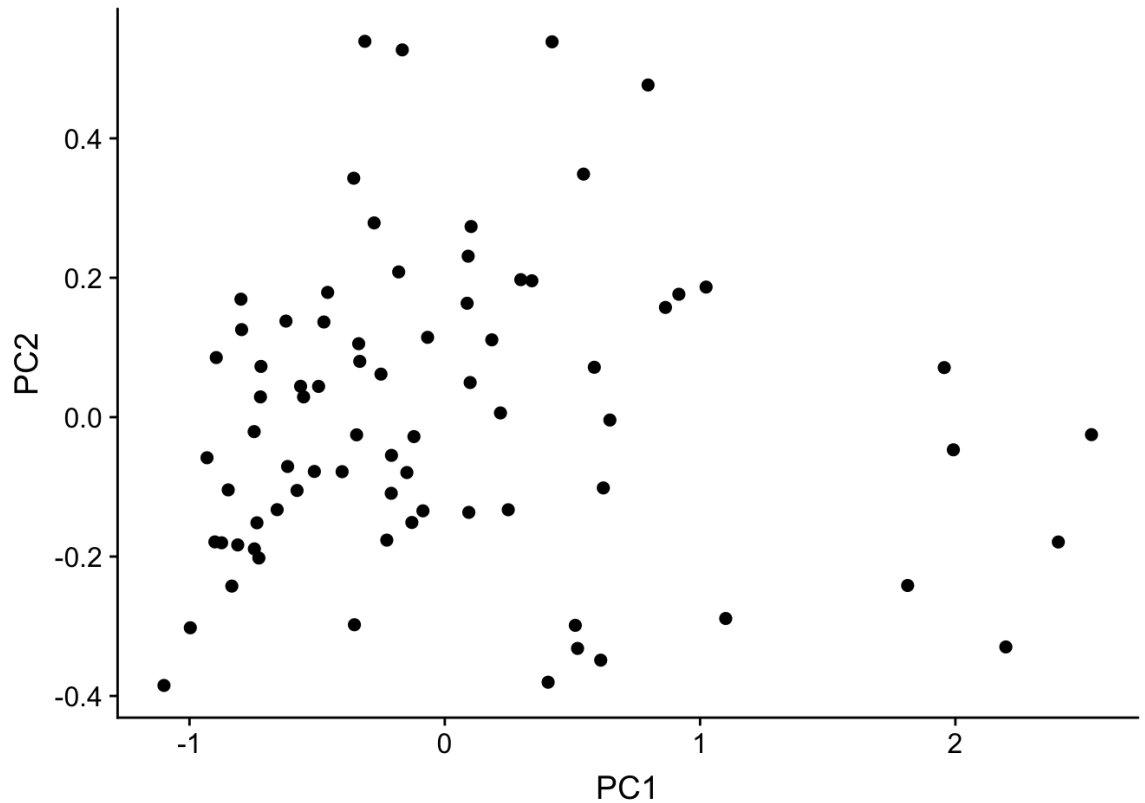
Principal Component Analysis

In practice, may also use a scaling transformation on the variables x_j to have unit variance.

In general, if variables x_j are measured in different units (e.g, miles vs. liters vs. dollars), variables should be scaled to have unit variance.

Conversely, if they are all measured in the same units, they should be scaled.

Principal Component Analysis



Mortgage affordability data embedded into the first two principal components.

Principal Component Analysis

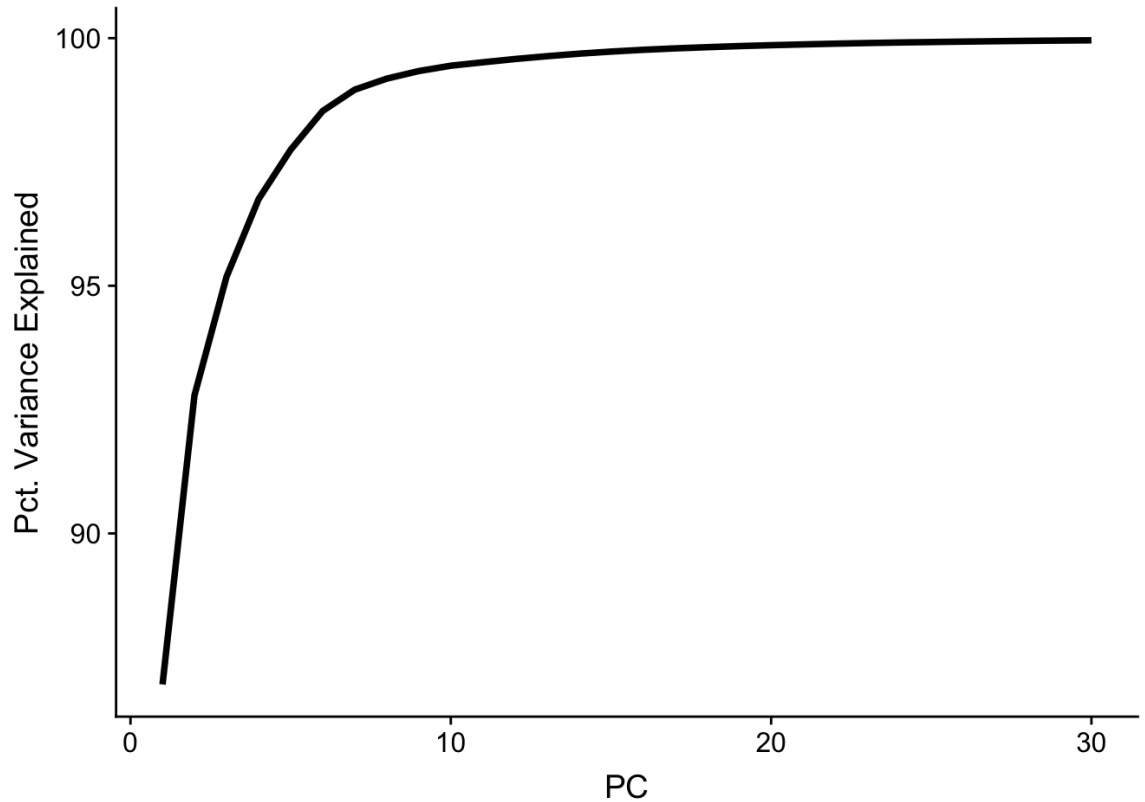
A natural question that arises: How many PCs should we consider in post-hoc analysis?

One result of PCA is a measure of the variance corresponding to each PC relative to the total variance of the dataset.

From that calculate the *percentage of variance explained* for the $_m$-th PC:

$$PVE_m = \frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$$

Principal Component Analysis



We can use this measure to choose number of PCs in an ad-hoc manner. In our case, using more than 10 or so PCs does not add information.

Principal Component Analysis

A useful *rule of thumb*:

- If no apparent patterns in first couple of PCs, stop!
- Otherwise, look at other PCs using PVE as guide.

Principal Component Analysis

A useful *rule of thumb*:

- If no apparent patterns in first couple of PCs, stop!
- Otherwise, look at other PCs using PVE as guide.

There are bootstrap based methods to perform a statistically guided selection of the number of PCs.

Principal Component Analysis

A useful *rule of thumb*:

- If no apparent patterns in first couple of PCs, stop!
- Otherwise, look at other PCs using PVE as guide.

There are bootstrap based methods to perform a statistically guided selection of the number of PCs.

However, there is no commonly agreed upon method for choosing number of PCs used in practice, and methods are somewhat ad-hoc.

Solving the PCA

The Principle Component solutions ϕ are obtained from the *singular value decomposition* of observation matrix $X_{n \times p} = UDV^T$

Solving the PCA

The Principle Component solutions ϕ are obtained from the *singular value decomposition* of observation matrix $X_{n \times p} = UDV^T$

Matrices U and V are orthogonal matrices, $U^T U = I$ and $V^T V = I$

Called the left and right *singular vectors* respectively.

Solving the PCA

The Principle Component solutions ϕ are obtained from the *singular value decomposition* of observation matrix $X_{n \times p} = UDV^T$

Matrices U and V are orthogonal matrices, $U^TU = I$ and $V^TV = I$

Called the left and right *singular vectors* respectively.

D is a diagonal matrix with $d_1 \geq d_2 \geq \ldots d_p \geq 0$. These are referred to as the *singular values*.

Solving the PCA

Using our previous notation V is the transformation matrix $V = [\phi_1, \phi_2, \cdots, \phi_p]$.

Principal components Z are given by the columns of UD. Since U is orthogonal, d_j^2 equals the variance of the jth PC.

Solving the PCA

From this observation we also see that we can write original observations x_i in terms of PCs z and transformations ϕ.

Specifically

$$x_i = z_{i1}\phi_1 + z_{i2}\phi_2 + \cdots + z_{ip}\phi_p$$

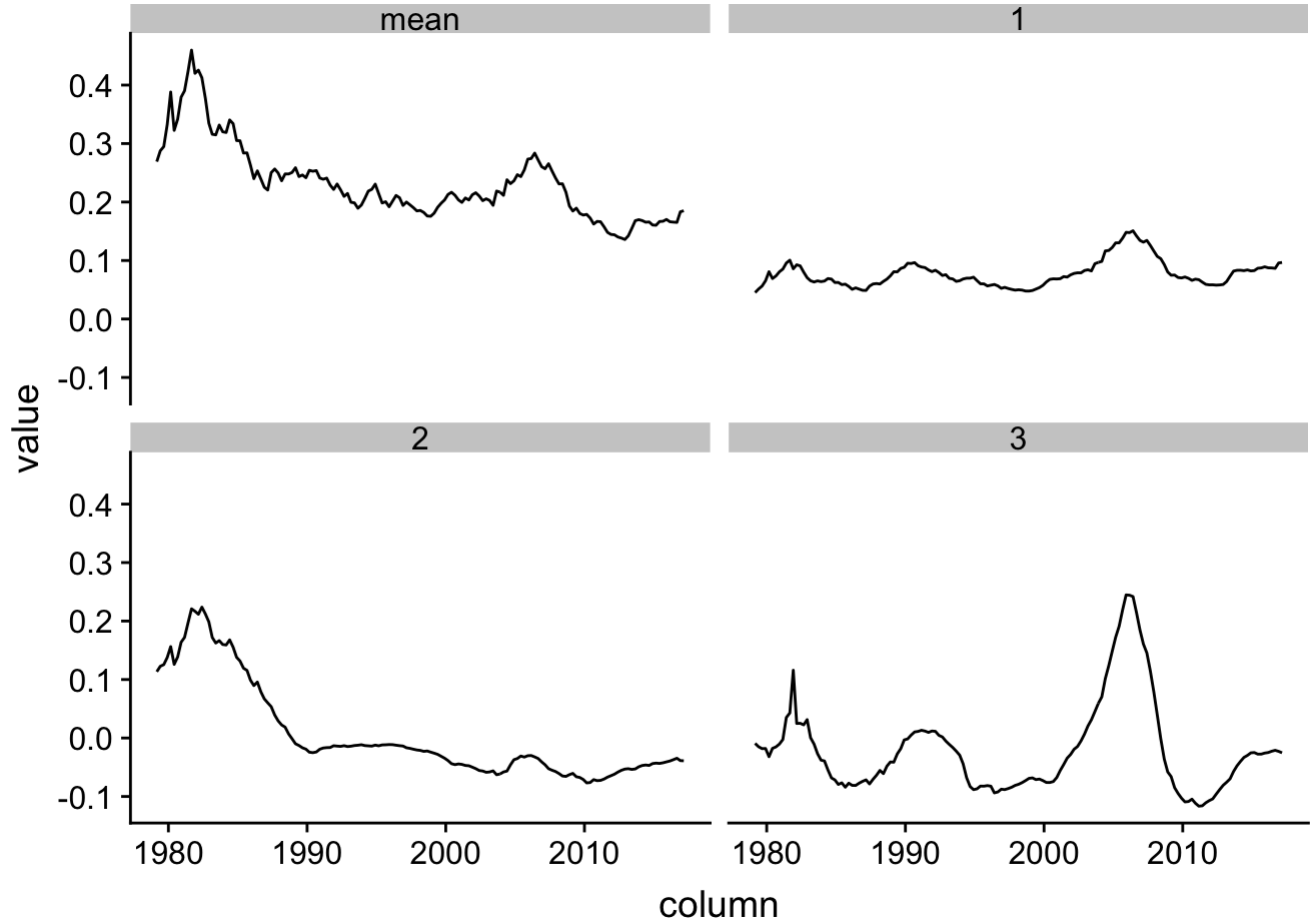
.

Solving the PCA

We can think of the ϕ_j vectors as a basis over which we can represent original observations i.

For this reason, another useful post-hoc analysis is to plot the transformation vectors $\phi_1, \phi_2, \ldots$.

Here we plot the mean time series (since we center observations X before performing the embedding) along with the first three ϕ_j vectors.



Kernel PCA

There is a connection between the singular value decomposition of X and the *eigenvalue* decomposition of XX^T such that

$$XX^T = UD^2U^T$$

.

Notice that the $_{ij}$th position of matrix XX^T is the inner product $x_i'x_j$.

Kernel PCA

As we saw before, we can use certain functions, kernel functions, in place of inner products

- induce non-linearities in learning methods,
- apply methods where we can obtain "similarity" functions directly.

Kernel PCA

Given a kernel matrix K, we obtain the eigenvalue decomposition of a "centered" kernel matrix

$$\tilde{K} = (I - M)K(I - M)^T = UD^2U$$

Where $M = \mathbf{1}\mathbf{1}^T/N$.

The principal components are obtained as before $Z = UD$.

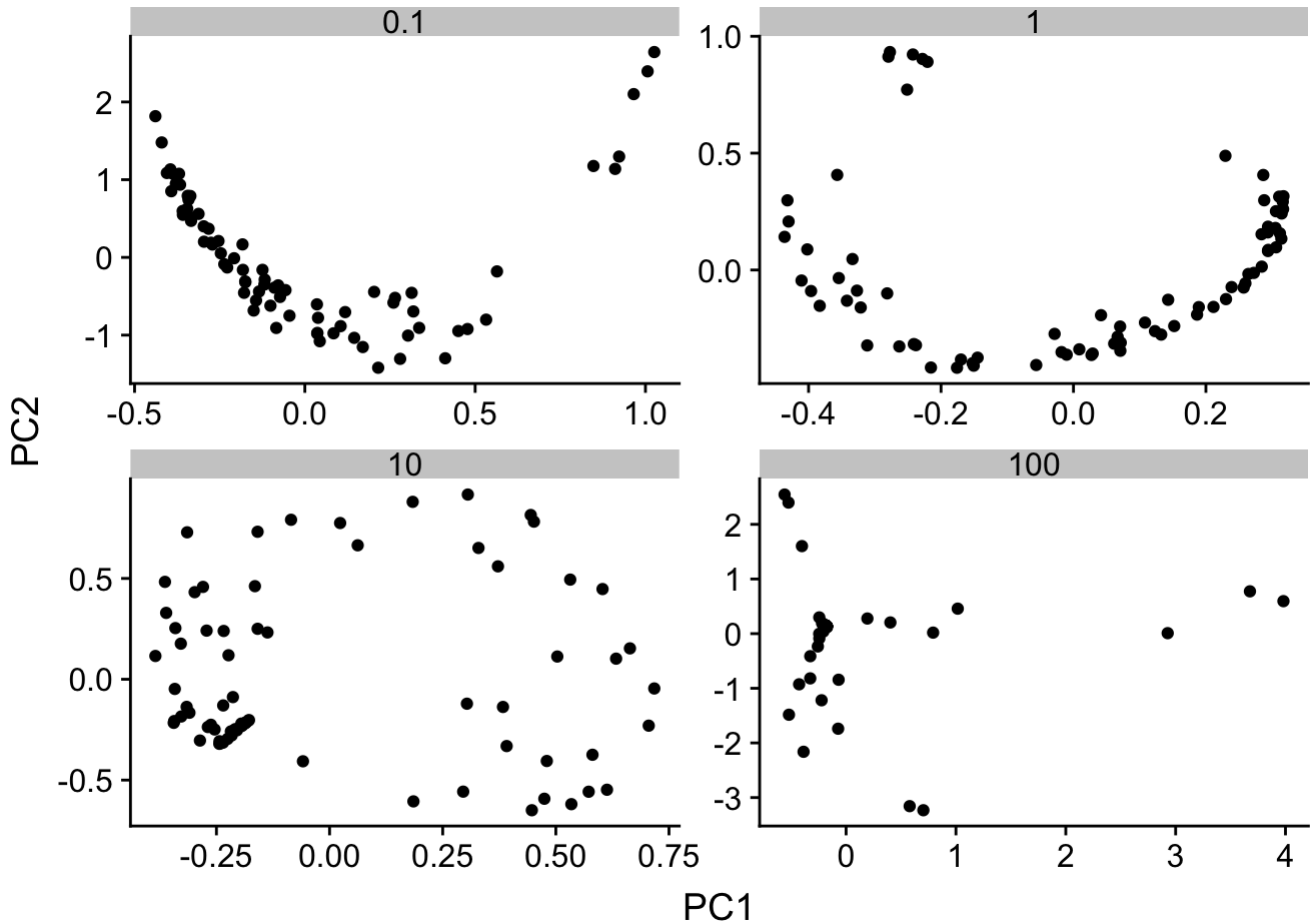
Kernel PCA

We apply kernel PCA using a radial basis function kernel

$$k(x, x_i) = \exp\{-\gamma \sum_{j=1}^{p} (x_j - x_{ij})^2\}$$

with various values of parameter γ.

Kernel PCA


0.1
1
10
100
PC2
PC1
2
1
0
-1
1.0
0.5
0.0
-0.5
-0.4
-0.2
0.2
0.75
0.50
0.25
0.00
-0.25
-2
-3
3
4

Kernel PCA

As usual in unsupervised learning, there is no principled way of choosing appropriate kernel functions or parameters other than ad-hoc observation of the resulting embeddings.

Multidimensional Scaling

Multidimensional scaling is a similar approach to PCA but looks at the task in a little different manner.

Given observations $x_1, \ldots, x_N$ in p dimensions, let d_{ij} be the distance between observations i and j. We may also use this algorithm given distances initially instead of p dimensional observations.

Multidimensional Scaling

Multidimensional scaling is a similar approach to PCA but looks at the task in a little different manner.

Given observations $x_1, \ldots, x_N$ in p dimensions, let d_{ij} be the distance between observations i and j. We may also use this algorithm given distances initially instead of p dimensional observations.

Multidimensional Scaling (MDS) seeks to find embeddings $z_1, \ldots, z_N$ of k dimensions for which Euclidean distance (in k dimensional space) is close to the input distances d_{ij}.

Multidimensional Scaling

In *least squares* MDS, we can do this by minimizing

$$S_M(z_1, \ldots, z_N) = \sum_{i \neq j} (d_{ij} - \|z_i - z_j\|)^2$$

A gradient descent algorithm is used to minimize this function.

Multidimensional Scaling

A related method that tends to better capture small distances is given by the *Sammon* mapping:

$$S_{S_m}(z_1, \ldots, z_N) = \sum_{i \neq j} \frac{(d_{ij} - \|z_i - z_j\|)^2}{d_{ij}}$$

Manifold learning and local embedding

PCA seeks to maximize variance of observations in the lower dimensional embedding.

Intuitively, that will result in faithfully capturing large distances between observations in predictor space.

Manifold learning and local embedding

PCA seeks to maximize variance of observations in the lower dimensional embedding.

Intuitively, that will result in faithfully capturing large distances between observations in predictor space.

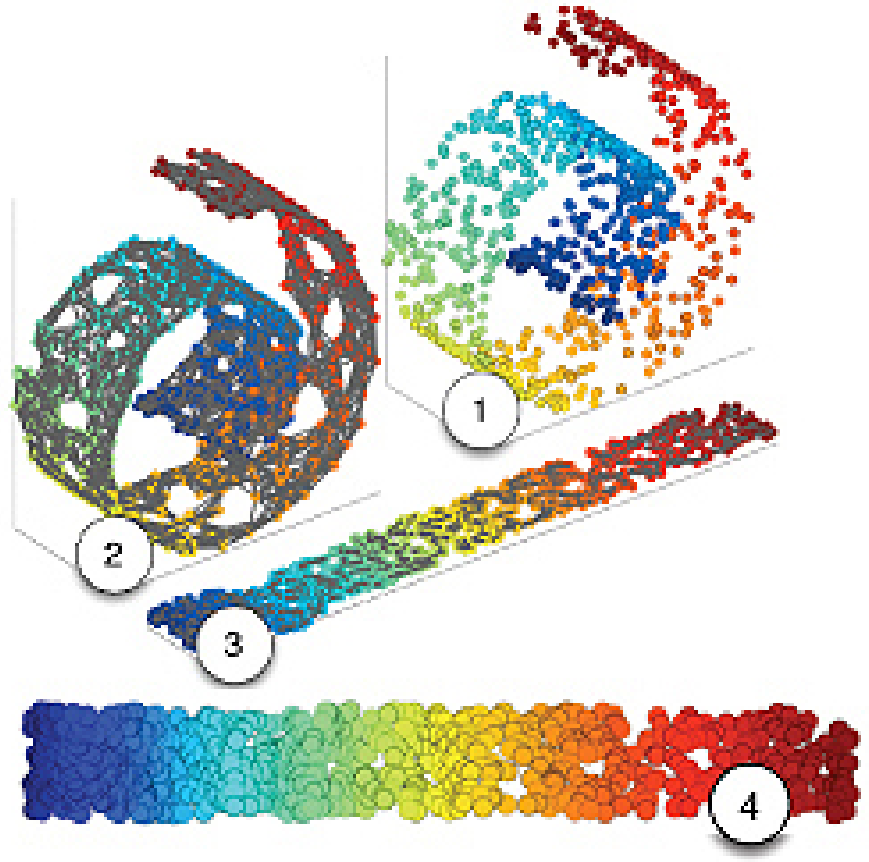
Similarly, MDS attempts to capture all pairwise distances between observations.

Manifold learning and local embedding

In some applications, this is not ideal, and methods that preserve *local* properties may be better suited.

A classical example is when observations lie in a smaller dimensional subspace of predictor space.

Methods that capture local structure are often able to better capture these subspaces.



Locally-Linear Embedding

A significant advance in manifold learning was achieved by the Locally-Linear Embedding method.

The intuition behind the method is that to capture local properties of the data in high-dimension, we should concentrate on preserving distances of neighbors in high-dimensional space,

Locally-Linear Embedding

In LLE, this is achieved by approximating each data point by linear combinations of neighboring points.

Then choose a lower dimensional embedding that best preserves these local approximations.

Locally-Linear Embedding

The algorithm is as follows:

- For each observation x_i in p dimensions, find its K nearest neighbors $\mathcal{N}(i)$.
- Approximate each observation as a linear combination of its neighbors by solving

$$\min_{w_{ik}} \|x_i - \sum_{k \in \mathcal{N}(i)} w_{ik} x_k\|^2$$

Add constraints $w_{ik} = 0$ if $k \notin \mathcal{N}(i)$ and $\sum_{k=1}^{N} w_{ik} = 1$.

Locally-Linear Embedding

These amounts to solving many small least squares problems.

Also note that $K < p$ for the least squares problems to have a solution.

Locally-Linear Embedding

- Given these local approximations, now find a low-dimensional embeddings z_i that approximates these well by minimizing

$$\sum_{i=1}^{N} ||z_i - \sum_{k=1}^{N} w_{ik} z_k||^2$$

Locally-Linear Embedding

The solution to this problem is given by the eigenvalue decomposition of matrix

$$(I - W)^T (I - W)$$

where W is the $N \times N$ approximation matrix.

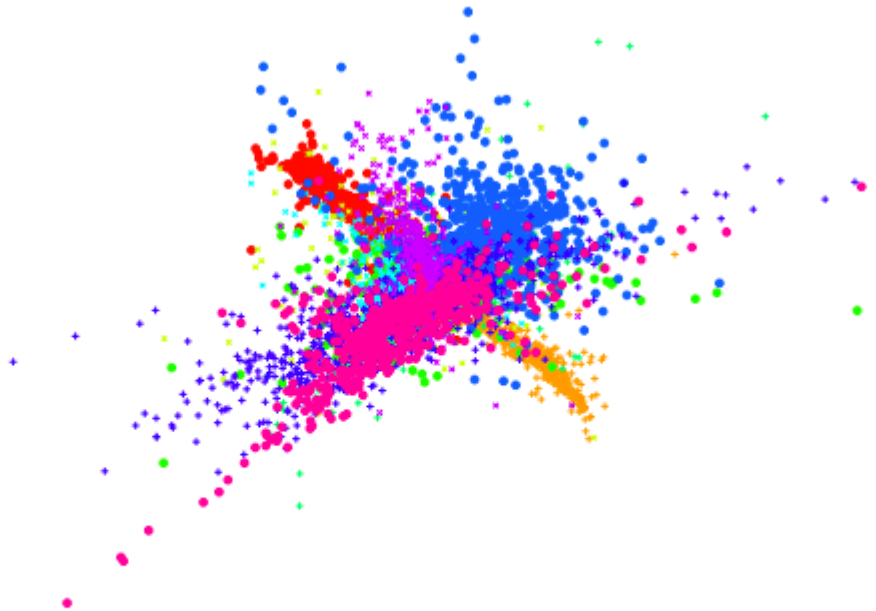
t-Distributed Stochastic Neighbor Embedding

Building on ideas from LLE, we arrive at a recent, very popular, embedding method.

t-SNE was designed to address a shortcoming of LLE where neighbors tended to crowd each other in the embedded space.

t-Distributed Stochastic Neighbor Embedding

For example, this is a LLE embedding of the MNIST digits dataset



(b) Visualization by LLE.

Figure 3: Visualizations of 6,000 handwritten digits from the MNIST data set.

t-Distributed Stochastic Neighbor Embedding

The two main ideas behind t-SNEs approach to solve the overcrowding problem are as follows:

- instead of linear approximations over neighbors use local density estimates based on a normal distribution;
- embed these density estimates to low dimension but use a heavier tailed distribution to overcome the crowding problem.

t-Distributed Stochastic Neighbor Embedding

Instead of operating directly over Euclidean distances in high-dimension in t-SNE we operate over conditional probability distributions based on Euclidean Distance.

Define

$$p_{j|i} = \frac{\exp\{-\|x_i - x_j\|^2 / 2\sigma_i^2\}}{\sum_{k \neq i} \exp\{-\|x_i - x_k\|^2 / 2\sigma_i^2\}}$$

t-Distributed Stochastic Neighbord Embedding

One way of performing an embedding is to define conditional probability distributions on the lower dimensional space

$$q_{j|i} = \frac{\exp\{-\|z_i - z_j\|^2\}}{\sum_{k \neq i} \exp\{-\|z_i - z_k\|^2\}}$$

t-Distributed Stochastic Neighbor Embedding

Minimize a divergence measure, e.g. Kullback-Liebler, between the two distributions

$$\sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

t-Distributed Stochastic Neighbor Embedding

Minimize a divergence measure, e.g. Kullback-Liebler, between the two distributions

$$\sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

A gradient method can be used to minimize this loss function.

t-Distributed Stochastic Neighbor Embedding

To overcome the crowding problem t-SNE uses two approaches.

First: notice that the conditional probabilities are not symmetric

$$p_{j|i} \neq p_{i|j}$$

.

t-Distributed Stochastic Neighbor Embedding

To address this, joint probability distributions are defined

$$p_{ij} = \frac{\exp\{-\|x_i - x_j\|^2 / 2\sigma_i^2\}}{\sum_{k \neq l} \exp\{-\|x_k - x_l\|^2 / 2\sigma_i^2\}}$$

and

$$q_{ij} = \frac{\exp\{-\|z_i - z_j\|^2\}}{\sum_{k \neq l} \exp\{-\|z_k - z_l\|^2\}}$$

t-Distributed Stochastic Neighbor Embedding

KL divergence between the two joint probability distributions is minimized

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-Distributed Stochastic Neighbor Embedding

KL divergence between the two joint probability distributions is minimized

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

The gradient of this divergence has a very simple form

$$\frac{\partial C}{\partial z_i} = 4 \sum_j (p_{ij} - q_{ij})(z_i - z_j)$$

t-Distributed Stochastic Neighbor Embedding

Second, instead of a normal distribution to define the joint probabilities q_{ij} in the embedding space, a longer tailed t distribution with one degree of freedom is used:

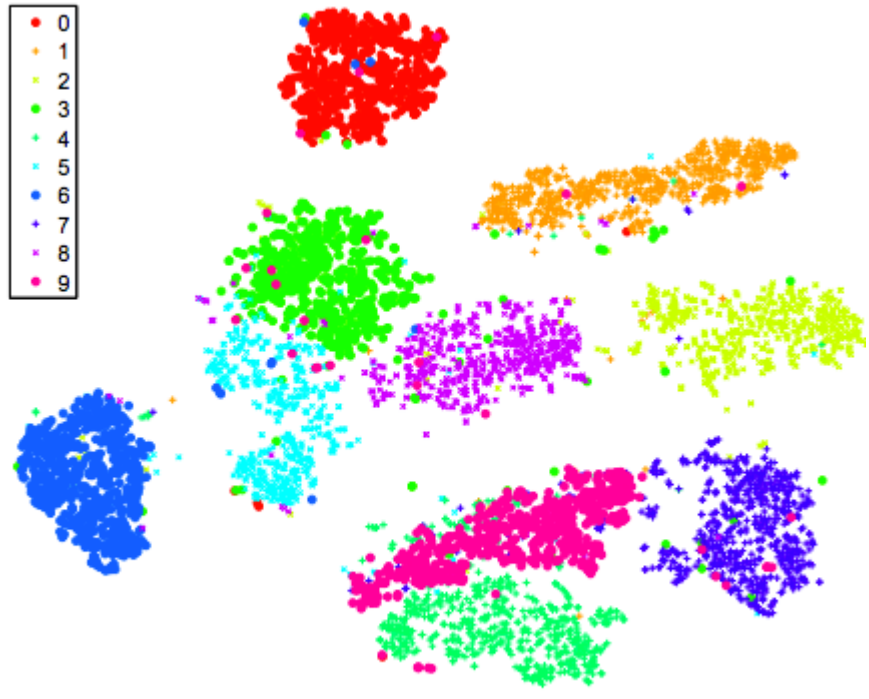
$$q_{ij} = \frac{(1 + \|z_i - z_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|z_k - z_l\|^2)^{-1}}$$

t-Distributed Stochastic Neighbor Embedding

KL divergence is again minimized, with slightly different gradient

$$\frac{\partial C}{\partial z_i} = 4 \sum_j (p_{ij} - q_{ij})(z_i - z_j)(1 + \|z_i - z_j\|^2)^{-1}$$

Here is the result of t-SNE on the MNIST digits data for which we showed the LLE result previously.



(a) Visualization by t-SNE.

t-Distributed Stochastic Neighbor Embedding

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

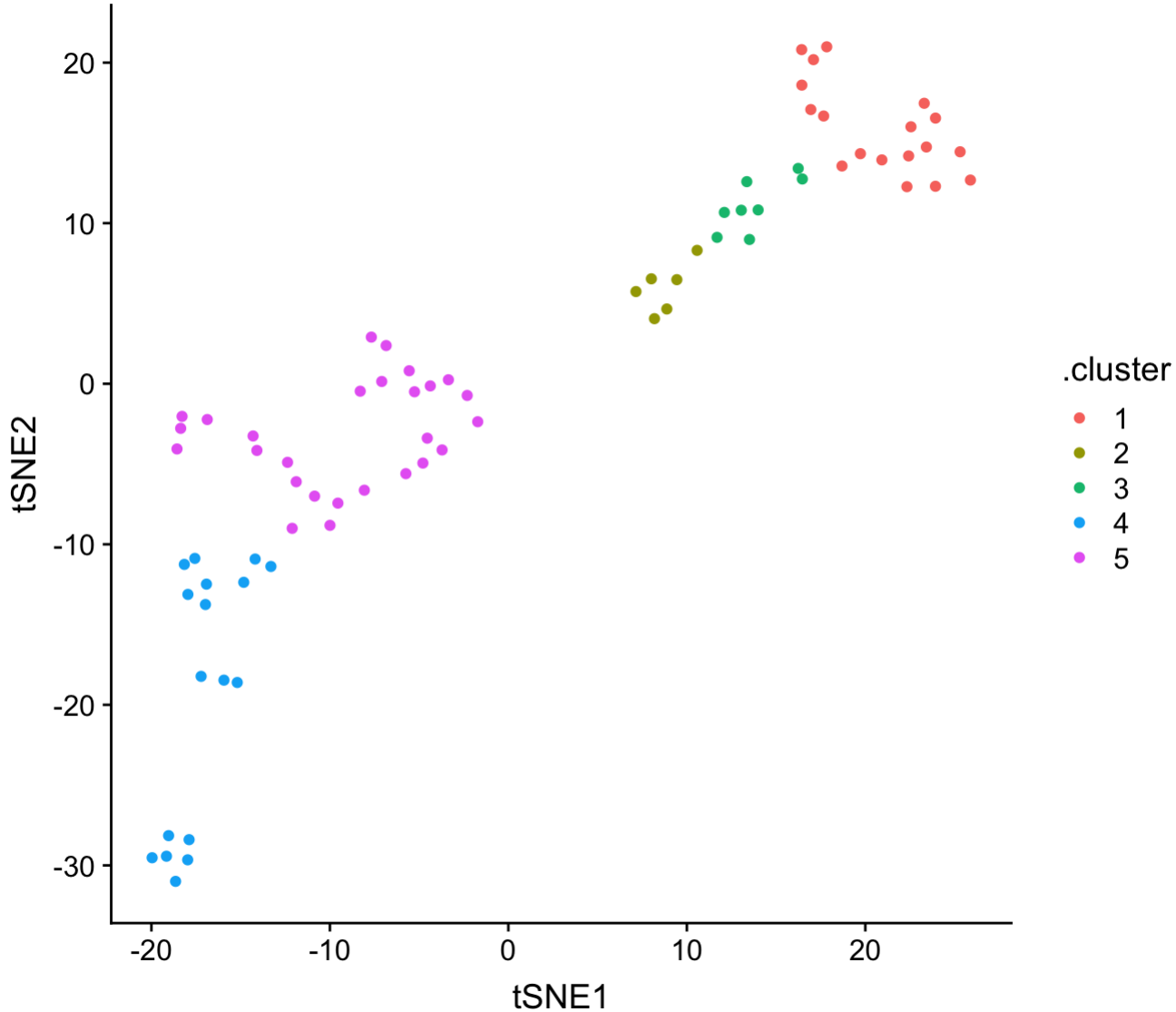
Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

t-Distributed Stochastic Neighbor Embedding

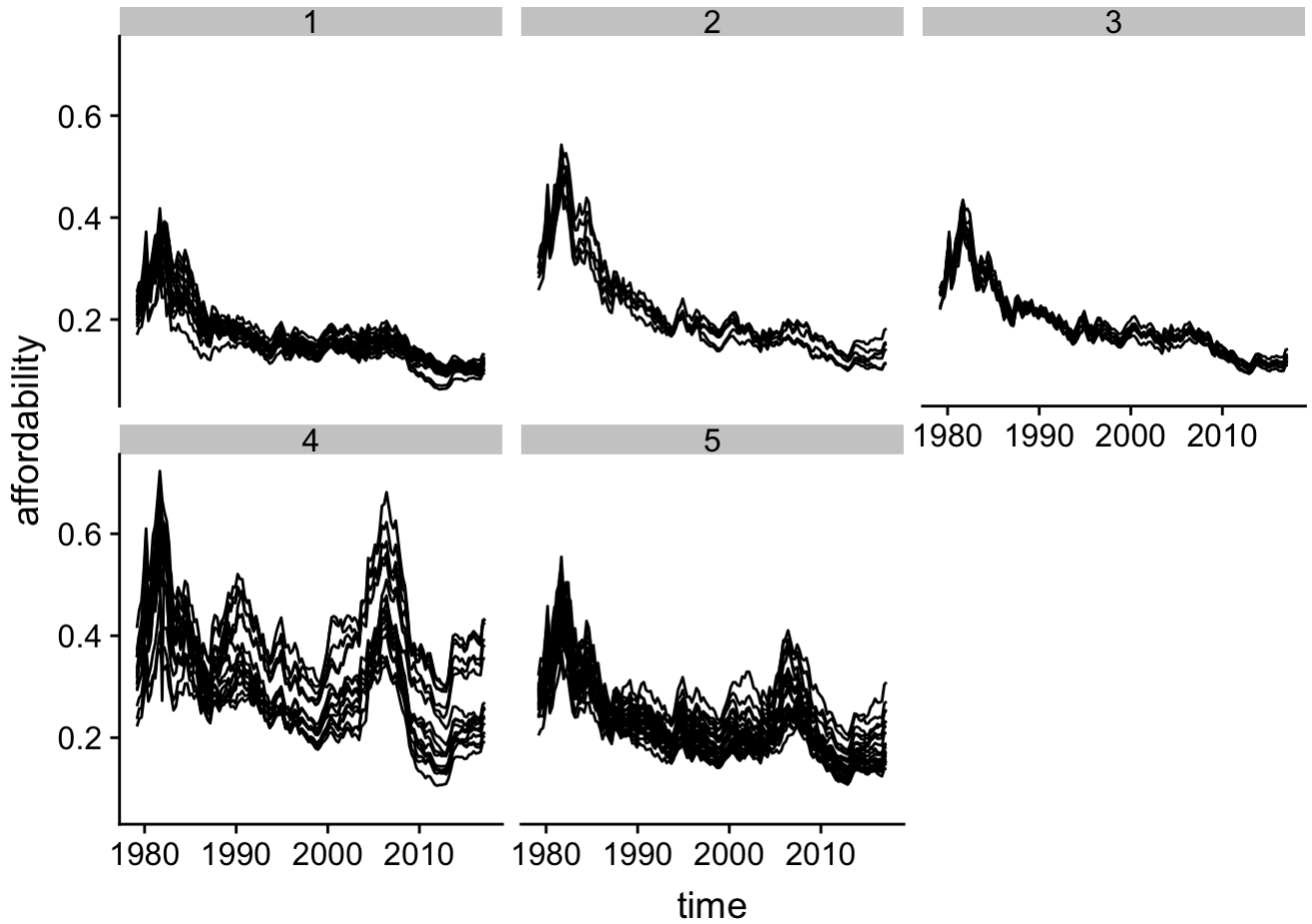
The parameter *Perplexity* is used to set parameter σ in the high-dimensional joint distribution, and is expressed roughly as the average number of neighbors contributing to the conditional probability distribution estimate.

t-SNE on affordability time series data with a perplexity parameter of 7.

Observations colored using the K-means algorithm on the two dimensional embedding.



Timeseries in the resulting clusters over the tSNE embedding.



Summary

Principal Component Analysis is a conceptually simple but powerful EDA tool. It is very useful at many stages of analyses.

PCA interpretation can be very ad-hoc, however. It is part of large set of unsupervised methods based on *matrix decompositions*, including Kernel PCA, Non-negative Matrix Factorization and others.

Embedding methods seek to capture local properties of observations. A popular recent method is the t-SNE method.