Dimensionality Reduction

Sophie Giffard-Roisin

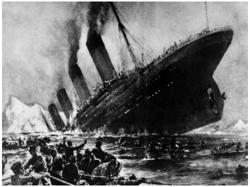
sophie.giffard@colorado.edu

Clone the repo `github.com/sophiegif/Workshop_dimensionality_reduction`

April 18, 2019

Outline

Dimensionality Reduction: Goal



- ML example: Predicting Titanic survivors (...!!)
- Data: list of passengers with their information
- Feature dimension?

	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	3	Palsson, Mrs. Nils (Alma Cornelia Berglund)	female	29.0	0	4	349909	21.075	NaN	S
1	2	Beane, Mr. Edward	male	32.0	1	0	2908	26.000	NaN	S
2	3	Palsson, Miss. Stina Viola	female	3.0	3	1	349909	21.075	NaN	S
3	3	Torber, Mr. Ernst William	male	44.0	0	0	364511	8.050	NaN	S
4	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.000	NaN	S

Problem Setting

- Better explain the data by reducing the number of features (dimensions)
- Can be seen as 'pre-processing step' ... but not always!
- Also : for visualizing/interpreting the data, for clustering...

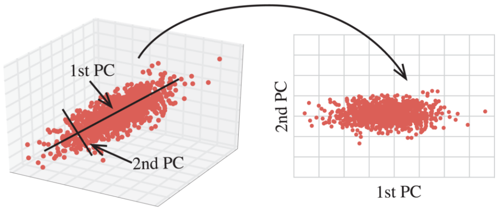
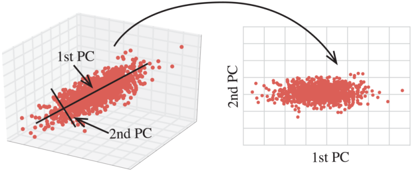


Figure: Most known example of dimensionality reduction: PCA (Principal Component Analysis)

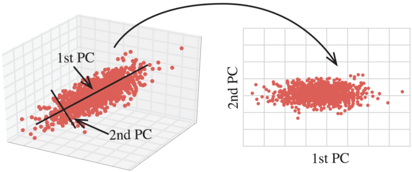
Outline

Principal Component Analysis



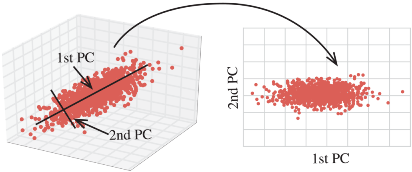
- **Linear** projection in a low dimensionality space
- ... such that the reduction error is minimized

Principal Component Analysis



- **Linear** projection in a low dimensionality space
- ... such that the reduction error is minimized
- 1) Find linear transfo. where 1sts axes maximize the variance:
  - covariance matrix $C = X^T X$ (X = *centered* feature matrix)
  - diagonalize $C = VLV^T$, V = eigenvectors; L = diagonal mat. of eigenvalues λ_i in $<$ order

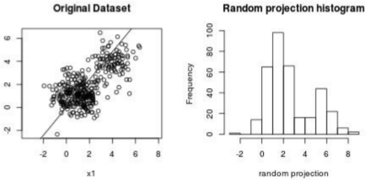
Principal Component Analysis

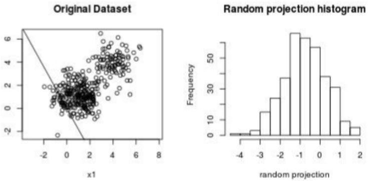


- **Linear** projection in a low dimensionality space
- ... such that the reduction error is minimized
- 1) Find linear transfo. where 1sts axes maximize the variance:
  - covariance matrix $C = X^T X$ (X = *centered* feature matrix)
  - diagonalize $C = VLV^T$, V = eigenvectors; L = diagonal mat. of eigenvalues λ_i in $<$ order
- 2) Select the Nb of eigenvectors (= PC) to keep
- Limitation: x_i should be iid Gaussian random variables (...!)

Random Projection

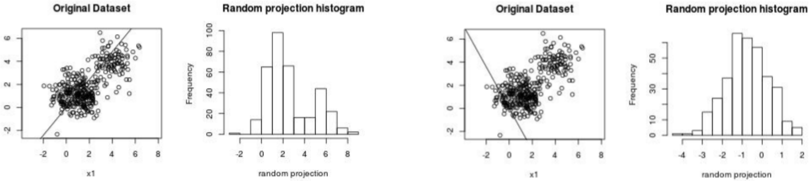
- **Linear random** projections in low dim. spaces
- Johnson-Lindenstrauss lemma: *a small set of points in a high-dim. space can be embedded in a space of $<<$ dim. s.t. distances between points are nearly preserved.*
- Nb of dim. in $O\left(\frac{\log(m)}{\varepsilon^2}\right)$ with $m =$ nb of **samples**





Random Projection

- **Linear random** projections in low dim. spaces
- Johnson-Lindenstrauss lemma: *a small set of points in a high-dim. space can be embedded in a space of $<<$ dim. s.t. distances between points are nearly preserved.*
- Nb of dim. in $O\left(\frac{\log(m)}{\varepsilon^2}\right)$ with $m =$ nb of **samples**
- Computationally efficient
- Limitation: useful if your feature dimension $>>$ nb of samples



Non-linear methods, ex. Isomap

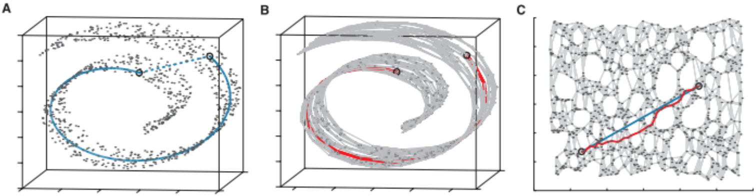


Figure: 2 points that are close together in Euclidean space may not reflect their intrinsic similarity. (Tenenbaum et al.)

- Lower-dim. embedding **maintaining geodesic distances**.

Non-linear methods, ex. Isomap

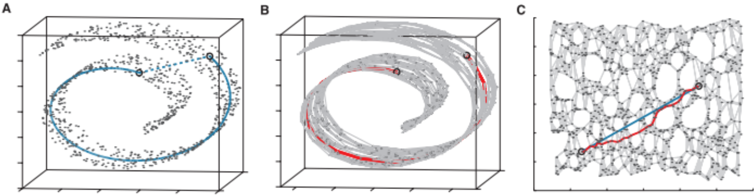


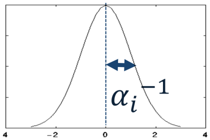
Figure: 2 points that are close together in Euclidean space may not reflect their intrinsic similarity. (Tenenbaum et al.)

- Lower-dim. embedding **maintaining geodesic distances**.
- 3 steps:
  - Nearest neighbor search (define k)
  - Shortest-path graph search (Dijkstra's Algorithm)
  - Compute lower-dimensional embedding
- Limitation: Can be sensitive to k

Dim. reduction incorporated in learning methods

Ex. Automatic Relevance Determination Regression or Classif.

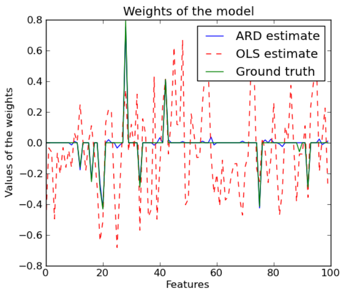
- Reduce feature dimension **according to the output**
- Automatic dim. reduction of input
- Idea:
  - Look for weights w s.t. $y = w^T \Phi(x)$
  - Prior on the w_i s : $P(w_i) = N(0, \alpha_i^{-1})$



Dim. reduction incorporated in learning methods

Ex. Automatic Relevance Determination

- Reduce feature dimension **according to the output**
- Automatic dim. reduction of input
- Idea:
  - Look for weights w s.t. $y = w^T \Phi(x)$
  - Prior on the w_i s : $P(w_i) = N(0, \alpha_i^{-1})$



And also...other methods!

- *Linear:* ICA (independent component analysis)
- *Non-linear:* Autoencoders (deep learning)
- *Built-in ML:* Random Forest

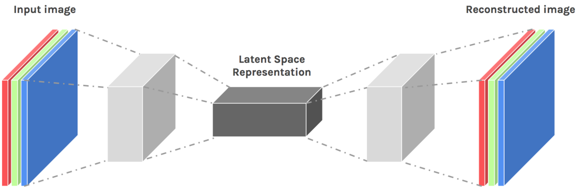


Figure: Autoencoder.

Things not to forget when doing dim. reduction

- Always find hyper-parameters on **training set**! (mean, std, eigenvectors, ...)
- **Visualize** your data first
- Linear methods are often not that bad
- **Understanding your problem/data** will help you know what are the relevant features!
- **Information $\neq$ Variability**

Your time to work!

- Play with different methods (scikitlearn = best friend)
- Python notebook: `github.com/sophiegif/Workshop_dimensionality_reduction`