Dimensionality reduction – prerequisites

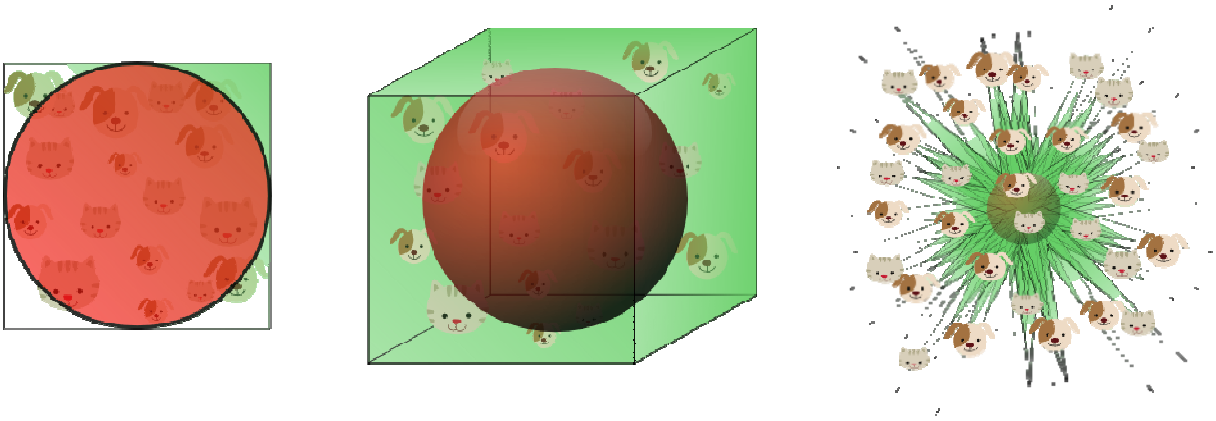
0. Motivation

1. Dimensionality reduction
    - Feature selection
    - Feature extraction
2. Dimensionality reduction
    - Linear
    - Non-linear
3. Linear algebra
    - Basics
    - Matrix factorizations/approximations
4. Probability and statistics
    - Random vectors
    - Random matrices
5. Other notions

0. Motivation (first 3 from A. Ng., Coursera)

- Data compression
    - Reduce the memory needed to store the data
    - Speed up the learning algorithm
    - Reduce the computational overhead of the algorithms
- Data visualisation
    - When the number of dimensions of the reduced features is 1, 2, or 3
- In the supervised case, as a preprocessing step, to prevent overfitting
    - ONLY WHEN the algortihm takes into consideration the labels
        - (e.g. PCA does not take into consideration the labels, so it must not be used in such a manner, but LDA takes into consideration the labels)
    - or has other theoretical properties that encourage you to use it
        - (e.g. JL transform can be used in conjunction with kNN with euclidian distance because it preserves euclidian distances between points)
    - Otherwise use regularization to prevent overfitting
    - Observation: it may happen for PCA to reduce overfitting, but there is no reason why it does this. You should run the learning algorithm without PCA, then if there are memory or time issues (or other pertinent reason to apply PCA – not prevent overfitting), then apply PCA
- Noise reduction
- Some algs. of dim. red. Also approximates the covariance matrix
    - PCA, PPCA, FA
- Avoid the curse of dimensionality (not necessarily regarding kNN)

- I have found many points of view:
  - A general one: The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience.
  - A general one: As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially (i.e, you overtfit)
    - Solution: regularization?
  - Euclidian Distance-based + uniformity of data, kNN can be found here: In high dimensional spaces, most of the training data resides in the corners of the hypercube defining the feature space. As mentioned before, instances in the corners of the feature space are much more difficult to classify than instances around the centroid of the hypersphere. This is illustrated by figure 11, which shows a 2D unit square, a 3D unit cube, and a creative visualization of an 8D hypercube which has 2^8 = 256 corners:



Highly dimensional feature spaces are sparse around their origin Figure 11. As the dimensionality increases, a larger percentage of the training data resides in the corners of the feature space.

Read here more, also about the blessing of non-uniformity: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Associated terms: Data sparsity – opposite for data density

If we would keep adding features, the dimensionality of the feature space grows, and becomes sparser and sparser.

  - When number of features > (or >>) number of instances
    - Solution: regularization?
  - When an algorithm's computational complexity increases with the number of dimensions. The term then simply refers to the increasing computational complexity.

- Kernel methods permit us to put our dataset in a high-dimensional space without having this problem (although, we might encounter other problems from above, right?)

1. Dimensionality reduction (classification after https://en.wikipedia.org/wiki/Dimensionality_reduction)

- Feature selection:
  i. = from a set of features, select (or remove) some of them
  ii. 2 types:
    1. filter (ro: filtru): example: 39/77 (or pb.3 in *FEATURE SELECTION EX ps6_sol.pdf*) for classification, but there are measures also for regression
    2. wrapper (ro: incapsulata):
       all discussed at *Special Chapters of Artificial Intelligence* course, Cristian Gatu in the context of regression, but the adaptation for classification is straightforward
       a. forward selection
       b. backward selection
       c. stepwise selection
- Feature extraction:
  i. = from a set of features, create new ones, usually fewer

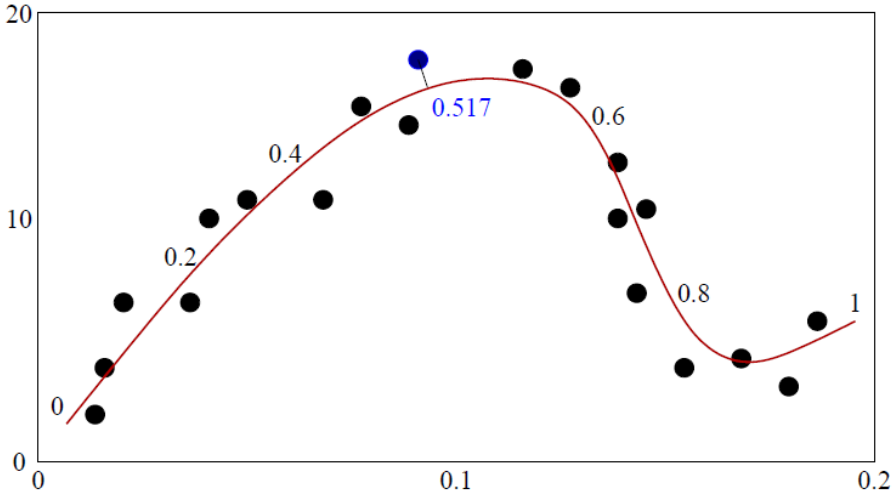
2. Dimensionality reduction

Manifold (or surface).

Examples: **1D manifold**

From „STA 414/2104, Statistical Methods for Machine Learning and Data Mining, Radford M. Neal, University of Toronto, 2014, Week 10”:

High dimensional data is often "really" lower-dimensional: For example:



These points all lie near a curve. Perhaps all that matters is where the points lie on this curve, with the small departures from the curve being unimportant.

If so, we can reduce this 2D data to one dimension, by just projecting each point to the nearest point on the curve. Specifying a point on the curve requires just one coordinate. For example, the blue point at $(0.9, 18)$ is replaced by 0.517.

2D manifold: a sphere in 3D space; 2D because we can indicate a point by (latitude, longitude)

In one simple form of dimensionality reduction, the manifold is just a hyperplane.

- Linear:

Definition 1 (Linear Dimensionality Reduction) *Given n d-dimensional data points $X = [x_1, ..., x_n] \in \mathbb{R}^{d \times n}$ and a choice of dimensionality $r < d$, optimize some objective $f_X(\cdot)$ to produce a linear transformation $P \in \mathbb{R}^{r \times d}$, and call $Y = PX \in \mathbb{R}^{r \times n}$ the low-dimensional transformed data.*

(from *cunningham15a.pdf* - Linear Dimensionality Reduction: Survey, Insights, and Generalizations, John P. Cunningham, Zoubin Ghahramani)

Algorithms:

- RCA / Random projections / JL transform -> first lin. Dim. Algo to study
  - Reduce dim. when only Euclidian distances matter in the algorithm
  - Preserves Euclidian distances: high dim. <-> low dim.s
- ZCA – zero-phase component analysis -> add observation in ex. 29/52 + create ex.
- PCA – principal component analysis
  - Introduce TLS, in contrast to OLS
  - Usually, the number of PCs is less than the number of columns. In this case, the data are orthogonally projected onto a subspace R^d of R^D.

        - Another interpretation: find a subspace such that when you project the points, the sample variance is maximized
    - When the number of PCs = the number of columns, we have a rotation+-reflexion... an orthogonal transf. Of the original data; another perspective: a change of basis (new features become uncorrelated)

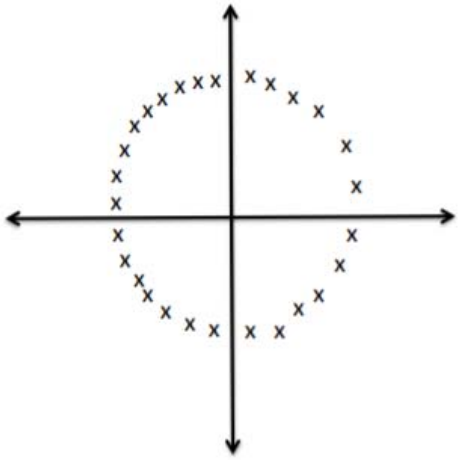


Figure 7: GPS data from a ferris wheel. PCA is not designed to discover nonlinear structure.

(from https://web.stanford.edu/class/cs168/l/l7.pdf)

- Linear autoencoder
- NMF – non-negative matrix factorization; suitable for DT (align parallel to the axes)
    - Sparse basis vectors
- FDA – Fisher's linear discriminant analysis (FDA = LDA – Linear discriminant analysis; Note that FDA is a manifold (subspace) learning method and LDA is a classification method; there also exists QDA – Quadratic discriminant analysis)
- CCA – canonical correlation analysis
- PPCA – probabilistic PCA
- FA – factor analysis
    - Connect it with soft clustering in low rank matrix factorization
    - rotations, for example varimax
        - in PCA
- ICA – independent component analysis; non-gaussian (from FA to ICA: *if we use non-gaussian distribution for p(zi) we arrive at ICA*)
    - Look at CMU slides (https://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/slides/ICA.pdf)

- Non-linear
    - Two ways of doing it: (idea from https://www.cs.utah.edu/~piyush/teaching/25-10-slides.pdf)
        - Nonlinearize a linear dim. red. method
            - Kernel PCA – the only non-linear dim. Reduction technique discussed in the book
        - Using manifold based methods

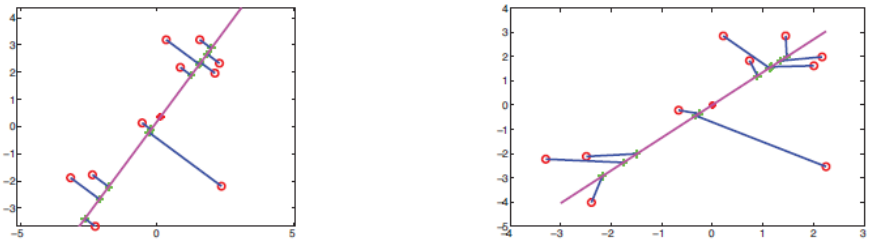
- SOM – self-organising maps
- GPLVM – gaussian process latent variable model
- Isomap
- Laplacian eigenmaps
- Principal curves and manifolds
- Non-linear autoencoder
- Others: see list at https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

Observation: other classifications:

- Probabilistic / Not
  - (Continuous) Latent variable model / Not
  - Gaussian / Not
- Supervised / Not

3. Linear Algebra

- Basics
  - Det
  - Rank
  - Linear independence of a set of vectors
  - Orthogonality, orthonormality of a set of vectors
  - Span
  - Basis
  - Orthogonal matrix
  - Orthogonal projection matrix
    - Mention also oblique projection matrix, see https://en.wikipedia.org/wiki/Projection_(linear_algebra)
      - Maybe from Kevin Murphy p.388



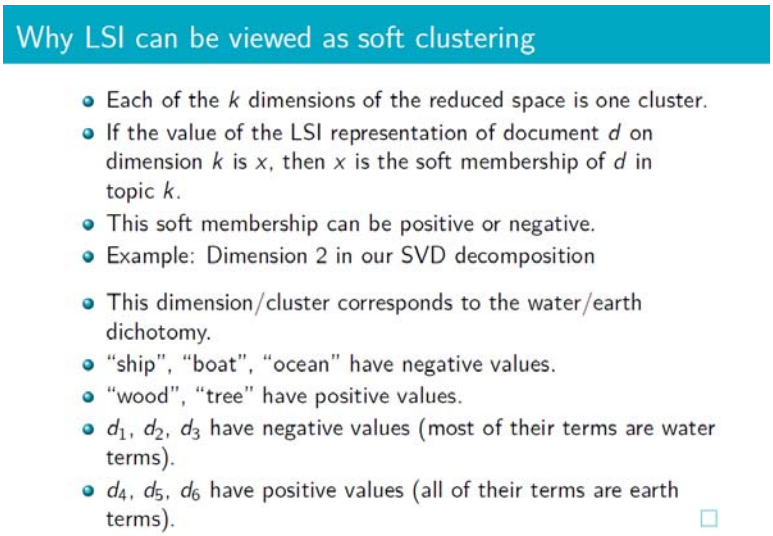
(a) (b)

Figure 12.5 An illustration of PCA and PPCA where $D = 2$ and $L = 1$. Circles are the original data points, crosses are the reconstructions. The red star is the data mean. (a) PCA. The points are orthogonally projected onto the line. Figure generated by `pcaDemo2d`. (b) PPCA. The projection is no longer orthogonal: the reconstructions are shrunk towards the data mean (red star). Based on Figure 7.6 of (Nabney 2001). Figure generated by `ppcaDemo2d`.

- Eigenvalues, eigenvectors, eigenspaces – maybe also discuss eigendecomposition
  - Definition
  - Interpretation
    - Geometric
    - Optimization
  - Computation
    - On paper
    - By the computer
      - Power method
      - Others
- Diagonalizable matrix, orthogonally diagonalizable matrix
- Singular values, left/right singular vectors – maybe also discuss SVD

- Matrix factorizations/approximations
    - In this context, rank = number of latent factors!
  - For each (if applicable):
    - Form
      - Matrix
      - Sum
    - Existence
    - Uniqueness
- Full-rank matrix factorization
  - LU
  - QR
  - Choleskly decomposition for real, symmetric matrices
  - Eigendecomposition for real, symmetric matrices
  - SVD
- Low-rank matrix factorization
  - Motivation (from https://web.stanford.edu/class/cs168/l/l9.pdf, page 4 – only the first 3):
    - Compression

- Denoising (e.g.: background removal in video)
- Matrix completion
- Soft clustering: + for both variables and points + interpretability
  From https://www.fi.muni.cz/~sojka/PV211/p18lsi.pdf slide 26/30:



Why LSI can be viewed as soft clustering

- Each of the k dimensions of the reduced space is one cluster.
- If the value of the LSI representation of document d on dimension k is x, then x is the soft membership of d in topic k.
- This soft membership can be positive or negative.
- Example: Dimension 2 in our SVD decomposition
- This dimension/cluster corresponds to the water/earth dichotomy.
- "ship", "boat", "ocean" have negative values.
- "wood", "tree" have positive values.
- d_1, d_2, d_3 have negative values (most of their terms are water terms).
- d_4, d_5, d_6 have positive values (all of their terms are earth terms).

- SVD
  - Best low-rank matrix factorization: Eckart-Young-Mirsky theorem

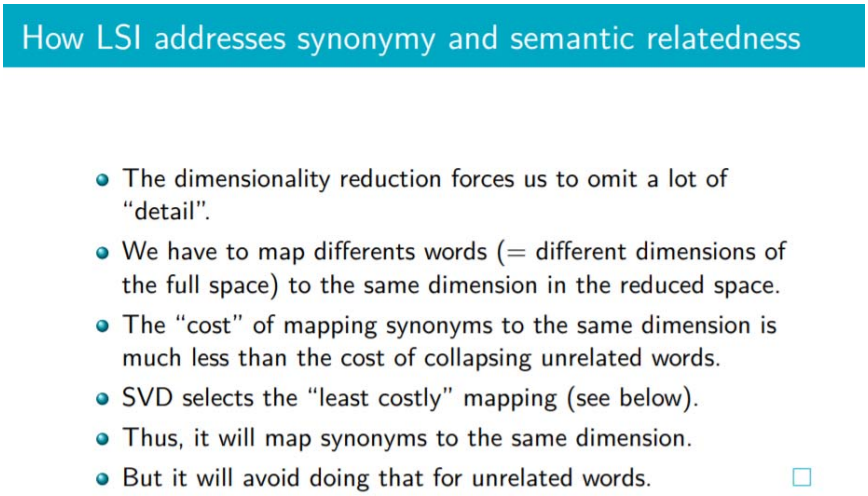
4. Probability and statistics

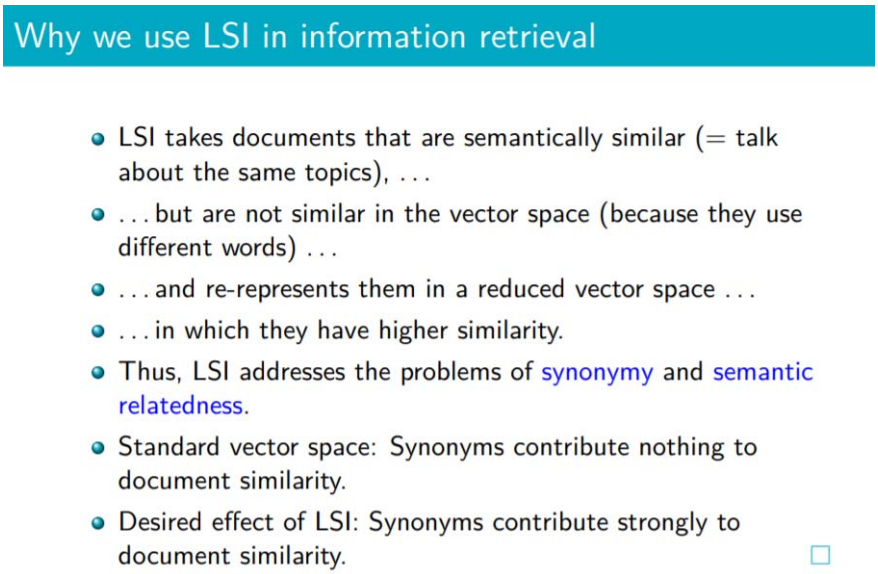
- Random vectors
  - Expectation
  - Covariance matrix
- Random matrices
  - Example: covariance matrix

5. Other notions

- Eigenfaces: PCA on face images
  - Main reason why applied: maany dimensions => de-noising
- Latent semantic analysis (LSA)/Latent semantic indexing (LSI): PCA on term-document matrix
  - Main reason why applied: maany dimensions => de-noising

- From https://www.fi.muni.cz/~sojka/PV211/p18lsi.pdf





- Why not centering: https://stats.stackexchange.com/questions/152879/latent-semantic-indexing-and-data-centering
- Why cos? From https://cmry.github.io/notes/euclidean-v-cosine

- # When to Use Cosine?

- Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter. This happens for example when working with text data represented by word counts. We could assume that when a word (e.g. `science`) occurs more frequent in document 1 than it does in document 2, that document 1 is more related to the topic of `science`. However, it could also be the case that we are working with documents of uneven lengths (Wikipedia articles for example). Then, `science` probably occurred more in document 1 just because it was way longer than document 2. Cosine similarity corrects for this.

- Text data is the most typical example for when to use this metric. However, you might also want to apply cosine similarity for other cases where some properties of the instances make so that the weights might be larger without meaning anything different. Sensor values that were captured in various lengths (in time) between instances could be such an example.

- FA-view of PCA:
    - PCA as approximating the covariance matrix
    - loadings, rescaled/standardized loadings;
    - scores, standardized scores
- Biplot (see: https://www.researchgate.net/profile/Ehab_Mahdy/post/Can_you_please_help_me_interpret_these_biplots/attachment/59d621b779197b8077980161/AS%3A297914397151239%401448039735565/download/Biplot.pdf and https://stats.stackexchange.com/questions/141085/positioning-the-arrows-on-a-pca-biplot)
    - In general: to mention the central property; in the context of low-rank matrix factorizations
    - In the context of „FA-view of PCA"
- Multicollinearity (see written draft)
    - Definition
        - from wikipedia
        - perfect multicollinearity = linear dependence => multicollinearity = „soft" linear dependence
    - 2 solutions:
        - PCA regression
            - Apply PCA on input
            - Retain only those PCs with eigenvalues not near zero
            - Apply OLS regression
            - For interpretability, write y = lin comb of PCs = lin comb of (lin comb of original variables) = lin comb of original variables
        - Ridge regression
- Other topics regarding eigenvalues discussed at *Big Data Analytics* course:
    - PageRank
    - Spectral Clustering
        - There are two types of spectral clustering
            - Classic/original/In graphs: Given a graph, take the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix of the graph and cluster according to the numbers in the eigenvector: if > 0 (or other threshold) then +, else –
            - In ML: see exercises