Distance Metrics & Dimensionality reduction

M. Vazirgiannis

DaSciM, LIX, École Polytechnique

September 2024

Outline

- **Distance Measures**
- Data Exploration and Preprocessing
- Dimensionality Reduction

Distance Measures

- Machine Learning algorithms capitalize on similarity or distance measures between objects.
- Similarity or distance between data points can be expressed as:
  - Explicit similarity for each pair of objects
  - Similarity obtained indirectly based on data vector attributes
- A distance $d(i,j)$ is a metric iff
  1. $d(i,j) \geq 0$ for all i,j and $d(i,j) = 0$ iff $i = j$
  2. $d(i,j) = d(j,i)$ for all i and j
  3. $d(i,j) \leq d(i,k) + d(k,j)$ for all i,j and k
- It has to have the shuffling invariant property

Distance

- Notation: n objects with p attributes

$$x(i) = (x_1(i), x_2(i), \ldots, x_p(i))$$

- Most common distance metric is *Euclidean* distance:

$$d_E(i,j) = \left(\sum (x_k(i) - x_k(j))^2\right)^{1/2}$$

- Makes sense in the case where the different measurements are are proportional; each variable measured in the same units.
- If the measurements are different, length and weight, it is not clear – need for standardization

Weighted Euclidean distance

- Finally, if we have some idea of the relative importance of each variable, we can weight them:

$$d_E(i,j) = \left(\sum w_k \left(x_k(i) - x_k(j) \right)^2 \right)^{1/2}$$

Other Distance Metrics

- Minkowski or L_p metric:

$$d_E(i,j) = \left(\sum_{k=1}^{p} (x_k(i) - x_k(j))^{\lambda} \right)^{1/\lambda}$$

- Manhattan, city block or L_1 metric:

$$d_E(i,j) = \sum_{k=1}^{p} |x_k(i) - x_k(j)|$$

- Chebyshev L_∞

$$d_E(i,j) = \max_k |x_k(i) - x_k(j)|$$

Variants of the L_1 family

- Sorensen

$$d_{\text{sor}}(i,j) = \frac{\sum_{k=1}^{p} |x_k(i) - x_k(j)|}{\sum_{i=1} |x_k(i) + x_k(j)|}$$

- Gowers

$$d_{gow}(i,j) = 1/p \sum_{k=1}^{p} |x_k(i) - x_k(j)|$$

- Lorentzian

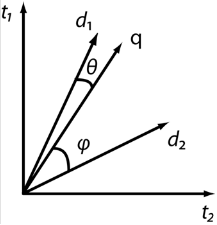
$$d_{lor}(i,j) = \sum_{k=1}^{p} \ln\left(1 + |x_k(i) - x_k(j)|\right)$$

- Inner product

$$s_{IP}(i,j) = \sum_{k=1}^{p} x_k(i)x_k(j)$$

- Harmonic Mean

$$s_{HM}(i,j) = 2\sum_{k=1}^{p} \frac{x_k(i)x_k(j)}{x_k(i)+x_k(j)}$$



- Cosine based similarity

$$\mathrm{sim}(q,d) = \frac{q \cdot d}{|q||d|} = \frac{\sum_{k=1}^{p} w_{k,q} \cdot w_{k,d}}{\sqrt{\sum_{k=1}^{p} w_{k,q}^2} \cdot \sqrt{\sum_{k=1}^{p} w_{k,d}^2}}$$

Intersection family [1]

- Intersection

$$s_{IS}(i,j) = \sum_{k=1}^{p} \min\left((x_k(i), x_k(j)\right)$$

- Czekanowski

$$s_{Cze}(i,j) = \frac{2\sum_{k=1}^{p} \min\left(x_k(i), x_k(j)\right)}{\sum_{k=1}^{p} \left(x_k(i) + x_k(j)\right)}$$

- Jaccard

$$s_{Jac}(i,j) = \frac{\sum_{k=1}^{p} x_k(i)x_k(j)}{\sum_{k=1}^{p} x_k(i)^2 + \sum_{k=1}^{p} x_k(j)^2 - \sum_{k=1}^{p} x_k(i)x_k(j)}$$

- Dice

$$s_{Dice}(i,j) = \frac{2\sum_{k=1}^{p} x_k(i)x_k(j)}{\sum_{k=1}^{p} x_k(i)^2 + \sum_{k=1}^{p} x_k(j)^2}$$

[1] Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions Sung-Hyuk Cha, INT. J. OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES

Squared L2 family

- Squared Euclidean

$$d_{sqe}(i,j) = \sum_{k=1}^{p} (x_k(i) - x_k(j))^2$$

- Pearson x^2

$$d_{pre}(i,j) = \frac{\sum_{k=1}^{p} (x_k(i) - x_k(j))^2}{x_k(j)}$$

- Divergence

$$d_{DIV}(i,j) = 2\sum_{k=1}^{p} \frac{(x_k(i) - x_k(j))^2}{(x_k(i) + x_k(j))^2}$$

Shannon's entropy family

- Kullback Leibler

$$d_{KL}(i,j) = \sum_{k=1}^{p} x_k(i) \ln \frac{x_k(i)}{x_k(j)}$$

- Jeffreys

$$d_{JF}(i,j) = \sum_{k=1}^{p} (x_k(i) - x_k(j)) \ln \frac{x_k(i)}{x_k(j)}$$

- K-divergence

$$d_{kids}(i,j) = \sum_{k=1}^{p} x_k(i) \ln \frac{2x_k(i)}{x_k(i) + x_k(j)}$$

- Jensen Shannon

$$d_{JS}(i,j) = 1/2 \left[\sum_{k=1}^{p} x_k(i) \ln \frac{2x_k(i)}{x_k(i) + x_k(j)} + \sum_{k=1}^{p} x_k(j) \ln \frac{2x_k(j)}{x_k(i) + x_k(j)} \right]$$

Distance metrics – Nominal values / text

- Nominal variables
  - Number of matches divided by number of dimensions

A	A	B	B	C	B	B	C	C	A
A	B	B	A	C	B	B	C	C	C

7/10

- Edit (Levenshtein) distance
  - **k**itten → **s**itten (substitution of "s" for "k")
  - sitt**e**n → sitt**i**n (substitution of "i" for "e")
  - sittin → sittin**g** (insertion of "g" at the end)

Exploratory Data Analysis

- Methods not including formal statistical modeling and inference
  - Detection of mistakes
  - Checking of assumptions
  - Preliminary selection of appropriate models
  - Determining relationships among the explanatory variables, and
  - Assessing the direction and rough size of relationships between explanatory and outcome variables (i.e. demographics – purchase)
- Useful information about the data
  - Min and Max values
  - Mean Value
  - Standard Deviation
  - Number of instances per value (for nominal data)
  - Percentage of missing values
  - Data distribution

Standardization

- 0-1 scaling:
  - each variable V is recomputed as

$$V = (V - \min V)/(\max V - \min V)$$

  - allows variables to have differing means and standard deviations but equal ranges.
  - at least one value at the 0 and 1 endpoints.
- Dividing each value by the range:
  - each variable V is recomputed as

$$V = V/(\max V - \min V)$$

  - means, variances, and ranges of the variables are still different
  - ranges are likely to be more similar.

Standardization

- Z-score scaling:
  - each variable V is recomputed as
    $(V - \text{mean} \;\; \text{of} \;\; V)/s$,
    s standard deviation.
  - all variables have equal means (0) and standard deviations (1) but different ranges.
- Dividing each value by the standard deviation.
  - transformed variables with variances of 1
  - different means and ranges.

Dependence among Variables

- Covariance and correlation measure linear dependence
- Assume variables X and Y and n objects taking on values $x(1), \dots, x(n)$ and $y(1), \dots, y(n)$.
- Sample covariance of X and Y is:

$$\mathrm{Cov}(X, Y) = \frac{1}{N} \sum_{1}^{n} (x(i) - \overline{x})(y(i) - \overline{y})$$

- Covariance measures how X and Y vary together.
    - large and positive if large values of $X \Rightarrow$ large values of Y, and small $X \Rightarrow$ small Y

Sample correlation coefficient

- Covariance depends on ranges of X and Y
- Standardize dividing with standard deviation
- Sample correlation coefficient

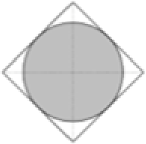
$$\rho(X, Y) = \frac{\sum_{i=1}^{n}(x(i) - \overline{x})(y(i) - \overline{y})}{\left(\sum_{i=1}^{n}(x(i) - \overline{x})^2 \sum_{i=1}^{n}(y(i) - \overline{y})^2\right)^{\frac{1}{2}}}$$

Dimensionality Reduction

Curse of Dimensionality

- Some coordinates do not contribute to the data representation.
- Subsets of the dimensions may be highly correlated.
- Nearest neighbor is distorted in a high dimensional space
- Low dimension intuitions do not apply to high dimensions
- Empty space phenomenon

Empty space phenomenon



- Hyper-sphere (**S**) within a hyper-rectangle (**R**)
- Respective volumes:

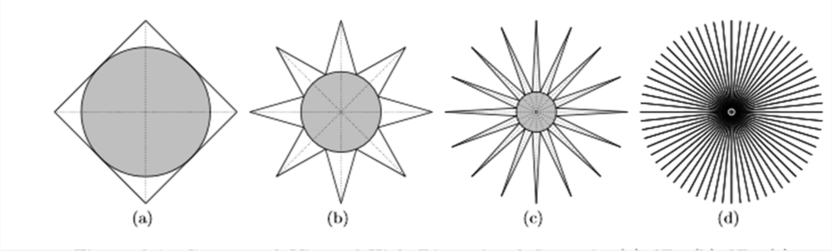
$$\mathrm{V}(\boldsymbol{S}) = 2r^d \pi^{\frac{d}{2}} / d\Gamma(d/2), \boldsymbol{V}(\boldsymbol{R}) = (2\boldsymbol{r})^{\boldsymbol{d}}$$

- Fraction of sphere within the rectangle becomes insignificant with d increasing:

$$\lim_{d \to \infty} \left(\pi^{\frac{d}{2}} / d2^{d-1} \Gamma \left({}^{d}/2 \right) \right) = 0$$

  - the normal distribution in high dimensions
  - longest/shortest distances converge
  - clustering becomes infeasible

Inscription of hyper sphere in a hypercube[2]



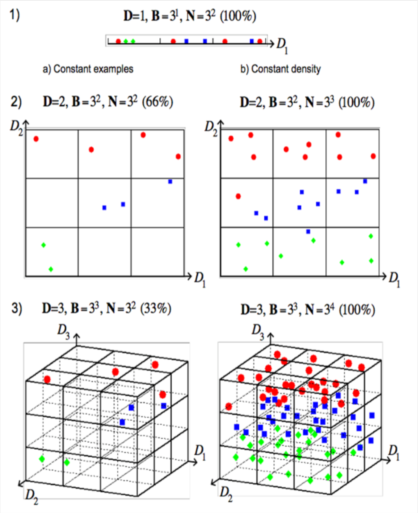
- The radius of the inscribed circle accurately reflects the difference between the volume of the hypercube and the inscribed hypersphere in d-dimensions.

[2]http://www.cs.rpi.edu/ zaki/www-new/uploads/Dmcourse/Main/chap6.pdf

Curse of Dimensionality [Belmann 1961]

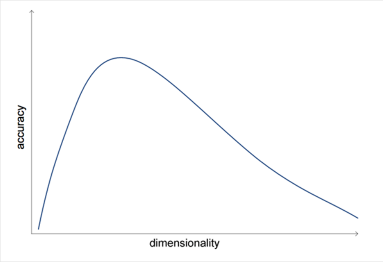
- Some coordinates do not contribute to the data representation.
- Subsets of the dimensions may be highly correlated.
- Nearest neighbor is distorted in a high dimensional space
- Low dimension intuitions do not apply to high dimensions

- Assuming 3 classes (colors)
  - same number of points embedded in higher dimensions (sparsity)
  - need exponentially more points to maintain density in higher dimensions (curse of dimensionality)
  - Data tend to gather in extremely of small areas of the multidimensional space (empty space phenomenon)



Curse of Dimesionality

- Point queries
  - "as dimensionality increases, distance to the nearest data point approaches the distance to the farthest data point" - "When Is "Nearest Neighbor" Meaningful? " - Beyer et al., [1999]
  - Increasing dimensionality may decrease of overall accuracy of system according to statistical learning theory approach [Vapnik, 1998].
  - for a given dataset, there is a maximum number of dimensions above which the quality of data analysis degrades when the number of training samples is small relative to dimensionality



Deterministic dimensionality reduction

- methods optimise an objective function
- does not contain any local optima - the solution space is convex [Boyd and Vandenberghe, 2004].
- has usually the form of solving an eigenvalue problem.
- final embedding space formed by eigenvectors which correspond to smallest or largest eigenvalues.

Deterministic methods classification

- Global methods: eigen-decomposition of a dense cost matrix
  - Methods: Principal Component Analysis, Multidimensional Scaling, Kernel Principal Component Analysis, Isomap, Maximum Variance Unfolding
- Local methods: eigen-decomposition of a sparse cost matrix
  - Methods: Locally Linear Embedding, Laplacian Eigenmaps etc.

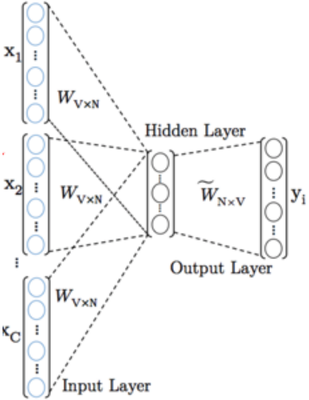
Dim. Reduction – Linear Algorithms

- *Matrix Factorization* methods
  - Principal Components Analysis (PCA)
  - Singular Value Decomposition (SVD)
  - Multidimensional Scaling (MDS)
  - Non negative Matrix Factorization (NMF)
  - Latent Semantic Indexing (LSI)

CBOW architecture

- Input C words
- Task: predict middle word
- Large dimensionality input CxV
- Hidden layer dim: N
- Output NxV

Data mapped to a N<<V dimensional space



Low Rank Approximation

- Data: $\boldsymbol{X} = \{\boldsymbol{x}_i \in \boldsymbol{R}^{mxn} | \boldsymbol{x}_i$ columns of $\boldsymbol{X}\}$
- Goal: approximate $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^{\boldsymbol{T}}$, $\boldsymbol{U} \in \boldsymbol{R}^{mxr}, \boldsymbol{V} \in \boldsymbol{R}^{nxr}, r << n$
  - each data vector $\boldsymbol{x}_i : \boldsymbol{x}_i \sim \mathrm{U}\boldsymbol{v}_i^T$, v_i is the i-th column of $\boldsymbol{V}$
- Geometric interpretation:
  - each data vector $\boldsymbol{x}_i \in R^m,_i \sim Uv_i{}^T$, is approximated by its projection to an r-dimensional space spanned by the column vectors of U
  - $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{V}^T$ the approximation matrix, max rank r

Evaluating the approximation

- Assuming a matrix X_{mxn} we need to define their similarity/distance.
- A popular matrix norm is the Frobenius (L_2 norm treated as a vector)

$$|X|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2} = Tr(XX^T)$$

- So assuming: $X = UV^T$
- the error approximation will be: $|X|_F - \left|UV^T\right|_F$

Dim. Reduction–Eigenvectors

- A *nxn* matrix
  - eigenvalues $\lambda : |A - \lambda I| = 0$
  - Eigenvectors $x : Ax = \lambda x$
  - Matrix rank: $\#$ linearly independent rows or columns
  - A real symmetric table A *nxn* can be expressed as: $A = U\Lambda U^T$
  - U's columns are A's eigenvectors
  - Λ's diagonal contains A's eigenvalues
  - $A = UAU^T = \lambda_I x_I x_1^T + \lambda_2 x_2 x_2^T + \ldots + \lambda_n x_i x_n^T$
  - $x_1 x_1^T$ represents projection via x_1 (λ_i eigenvalue, x_i eigenvector)
  - Interpretations XX^T vs. $X^T X$

Singular Value Decomposition (SVD)

Eigen decomposition applied to square matrices. For non square matrices we apply Singular Value Decomposition.
SVD insight: treat the rows of X_{nxm} matrix as n points in a m-dimensional space

- Consider the problem of finding the best k-dimensional subspace with respect to the set of points ($k << m$).
- Best - best least squares fit: minimize the sum of the squares of the perpendicular distances of the points to the subspace.

Singular Value Decomposition (SVD) - I

- Let X a *nxm* table, $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$
  - $\boldsymbol{U}$: orthogonal *mxm*, its columns are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$
  - $\boldsymbol{U}$, $\boldsymbol{V}$ define orthogonal basis $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}\boldsymbol{V}^T = 1$
  - Σ: *mxn* contains A's sigular values (square roots of $\boldsymbol{X}\boldsymbol{X}^T$ eigenvalues)
  - $\boldsymbol{V}$: *nxn*, its columns are eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$
- *k*-dimensional matrix approximation $X_k = U_k \Sigma_k V_k^T$

Multidimensional Scaling (MDS)

- Application of SVD on the data distance matrix XX^T
- Aim to minimize the stress:

$$\text{stress (X, X')} = \frac{\sum_{ij} \left(d(i,j) - d'(i,j)\right)^2}{\sum_{ij}(d(i,j))^2}$$

  Complexity $O(N^3)$ (N: number of vectors)
- Result:
  - A new representation of the data in a lower dimensional space.
- Implement usually by:
  - Eigen decomposition of the inner product matrix
  - projection on the k eigenvectors corresponding to the k largest eigenvalues.

Multidimensional Scaling

- Data is given as rows in X
  - $C = XX^T$ (inner product of x_i with x_j)
  - Eigen decomposition of $C' = ULU^{-1}$
  - Eventually $X' = U_k L_k^{1/2}$, where k is the projection dimension

$$X = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 6 & 4 & 2 & 6 \\ 1 & 5 & 6 & 8 \\ 1 & 4 & 4 & 6 \\ 3 & 4 & 9 & 5 \end{bmatrix} \xrightarrow{XX^T} C = \begin{bmatrix} 54. & 62. & 81. & 60. & 79. \\ 62. & 92. & 86. & 66. & 82. \\ 81. & 86. & 126. & 93. & 117. \\ 60. & 66. & 93. & 69. & 85. \\ 79. & 82. & 117. & 85. & 131. \end{bmatrix}$$

$$\xrightarrow{EVD} U = \begin{bmatrix} \boxed{\begin{matrix} -0.3540571 & -0.0266618 \\ -0.4041785 & -0.8612673 \\ -0.5309769 & 0.1813750 \\ -0.3931327 & 0.0342107 \\ -0.5242075 & 0.4726950 \end{matrix}} & \begin{matrix} -0.0427173 & 0.7674171 & -0.5321456 \\ 0.2512931 & -0.1004458 & 0.1470402 \\ -0.5293206 & 0.1591309 & 0.6161685 \\ -0.4240133 & -0.5922139 & -0.5601532 \\ 0.6892456 & -0.1579292 & 0.0420115 \end{matrix} \end{bmatrix}$$

$$L = \begin{bmatrix} \boxed{429.83919 \quad 28.182284} & 13.857017 & 0.1215106 & 1.380D-14 \end{bmatrix}$$

$$X' = U_2 L_2^{1/2} = \begin{bmatrix} -152.18764 & -0.7513907 \\ -173.73175 & -24.27248 \\ -228.23469 & 5.1115622 \\ -168.98385 & 0.9641354 \\ -225.32491 & 13.321624 \end{bmatrix}$$

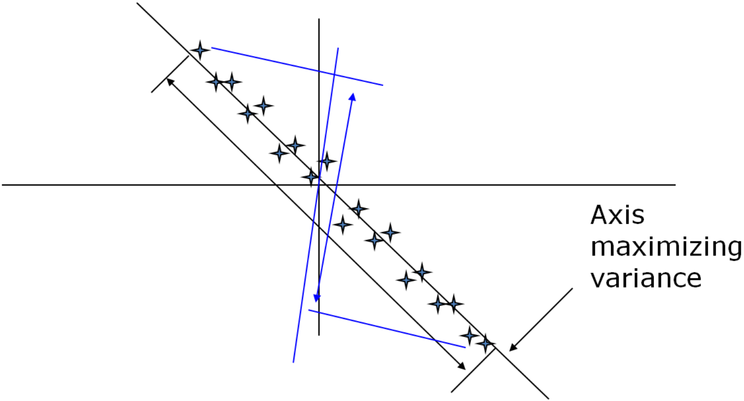
Principal Components Analysis (PCA)

- Main concept of Principal Components Analysis: dimensionality reduction, maintaining as much as possible data's variance.
- SVD on the data covariance matrix
- Data variance: $V(X) = \sigma^2 = E\left[(X-\mu)^2\right]$
- Let N objects, with mean value, m, it is approximated as:

$$\frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2$$

Sample of N objects with unknown mean value: $\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\overline{x})^2$

Dimensionality reduction based on variance maintenance



Covariance Matrix

- Let Matrix $X = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$ where X_i vectors
- covariance matrix Σ is the matrix whose (i, j) entry is the covariance

$$\Sigma = \begin{bmatrix} \mathrm{E}\left[(X_1-\mu_1)(X_1-\mu_1)\right] & \mathrm{E}\left[(X_1-\mu_1)(X_2-\mu_2)\right] & \cdots & \mathrm{E}\left[(X_1-\mu_1)(X_n-\mu_n)\right] \\ \mathrm{E}\left[(X_2-\mu_2)(X_1-\mu_1)\right] & \mathrm{E}\left[(X_2-\mu_2)(X_2-\mu_2)\right] & \cdots & \mathrm{E}\left[(X_2-\mu_2)(X_n-\mu_n)\right] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}\left[(X_n-\mu_n)(X_1-\mu_1)\right] & \mathrm{E}\left[(X_n-\mu_n)(X_2-\mu_2)\right] & \cdots & \mathrm{E}\left[(X_n-\mu_n)(X_n-\mu_n)\right] \end{bmatrix}$$

- Also: $\mathrm{cov}(X) = [X']^T X'$, where $X' = X - M$

Principal Components Analysis (PCA)

- PCA intuition: maximization of the covariance.
  - Variance: Depicts the maximum deviation of a random variable from the mean.

$$\sigma^2 = \sum_{i=1}^{n} (x_i - \mu_i)^2 / n$$

- Method:
  - Data feature p variables and contained as rows in matrix X_{pxn}
  - Covariance matrix $W = [X']^T X'$, $X' = X - M$
- Calculate eigenvalues and eigenvectors of

$$W = U \Lambda U^T$$

- Retain k largest eigenvalues and eigenvectors
  - k is estimated by $\sum_{j=k+1}^{p} \lambda_j / \sum_{j=1}^{p} \lambda_j > 85\%$
- Projection: $X'_k = X' U_k$

Principal Components Analysis

X=

x_1	2	3	4	5
x_2	6	4	2	6
x_3	1	5	6	8
x_4	1	4	4	6
x_5	3	4	9	5

15 20 25 30

$\Sigma_{j=1}^{5} x_{ij}$

$\text{sum}_j/5$

$x_{ij} - m_j$

X′=

x_1	-1	-1	-1	-1
x_2	3	0	-3	0
x_3	1	1	1	2
x_4	-2	-1	-1	1
x_5	0	0	0	-1

3 4 5 6

$X'^T X'$

Cov=

3.70	1.05	-1.05	0.2
1.05	0.7	0.55	0.55
-1.05	0.55	2.20	0.70
0.2	0.55	0.70	1.70

$(\Sigma_{f=1}^{5}(x_{if}-m_i)(x_{jf}-m_j))/4$

$Cov = ULU^T$

U=

-0.90	-0.18	-0.19	0.34
-0.20	-0.40	-0.20	-0.86
0.38	-0.66	-0.54	0.34
-0.01	-0.60	0.78	0.09

L=

4.38	2.89	1.02	0.004

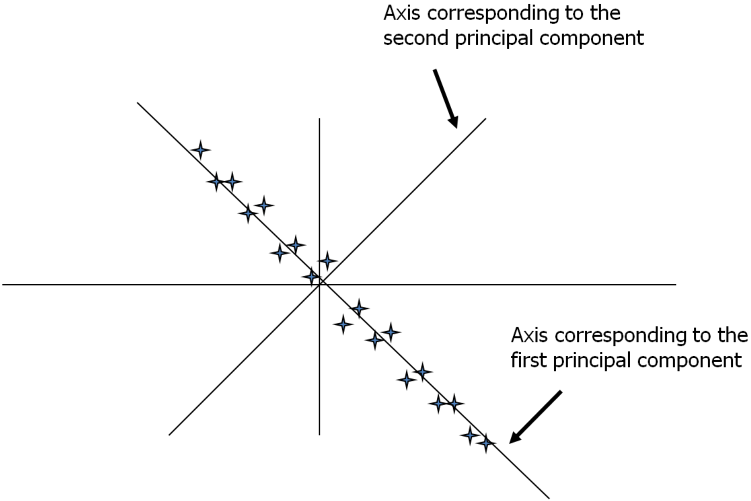
k=2

-0.90	**-0.18**
-0.20	**-0.40**
-0.38	**-0.66**
-0.01	**-0.60**

$X'U_k$

2.2	1.16
0	0
-2.4	-1.56
2.7	0.54
0.2	0.4

PCA, example


Axis corresponding to the
second principal component
Axis corresponding to the
first principal component

PCA Synopsis & Applications

- Preprocessing step preceding the application of data mining algorithms (such as clustering).
- Data Visualization & Noise reduction.
- It is a dimensionality reduction method
- Nominal complexity $O\left(np^2 + p^3\right)$
    - n: number of data points
    - p: number of initial space dimensions
- The new space maintains sufficiently the data variance.

Explaining data by factorization

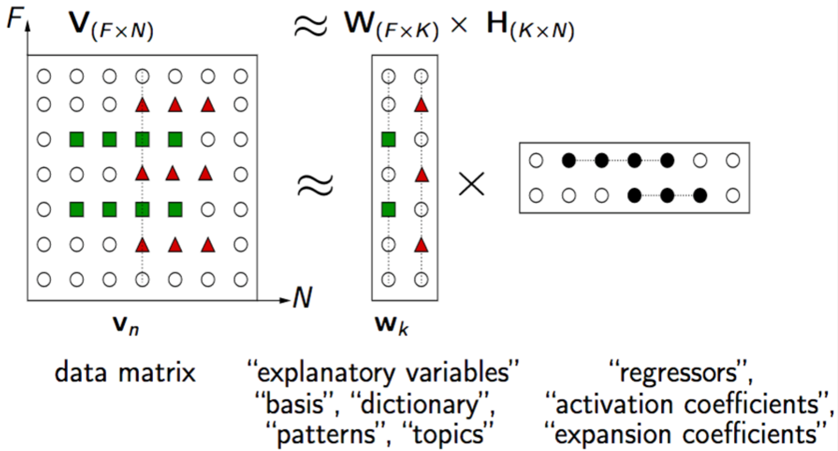


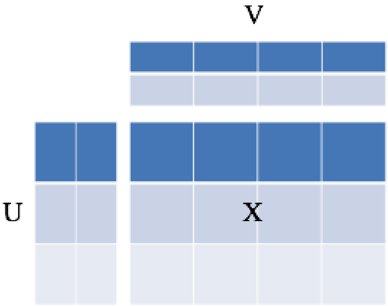
Illustration by C. Févotte

Non Negative Matrix factorization (NMF)

- Data is often nonnegative by nature
  - pixel intensities; occurrence counts; food or energy consumption; user scores; stock market values;
- Interpretability of the results, optimal processing of nonnegative data may call for processing under Nonnegativity constraints
- Applying SVD results in factorized matrices with positive and negative elements may contradict the physical meaning of the result.
  - Nonnegative matrix factorization (NMF)
  - find the reduced rank nonnegative factors to approximate a given nonnegative data matrix.

NMF model

- $X \simeq UV^T$
  - $U = [u_{fk}], w_{fk} >= 0$
  - $V = [v_{kn}], h_{kn} >= 0$
  - $k << f, n$



NMF

- Assume X (mxn) data matrix and $r << m, n$
- NMF aims to find non negative matrices

$$U \in R^{mxr}, V \in R^{rxn} : X \approx UV^T$$

- To find U, V, optimization problem:

$$\min_{(U,V)} \left\| X - UV^T \right\|_2$$

- Alternative error function:

$$\min_{U,V} f(U,V) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(X_{ij} \log \frac{X_{ij}}{(UV^\top)_{ij}} - X_{ij} + \left(UV^\top\right)_{ij} \right)$$
$$\text{s.t. } U_{ia} \geq 0, V_{jb} \geq 0, \forall i, a, b, j$$

Alternating Least squares

1. Suppose we know U, with V unknown.
   for each j minimize $\left\|X_{.j} - UV_{.j}^T\right\|_2$
   - find $V_{.j}$ that minimizes with $X_{.j}$ and U known.
   - Frobenius norm: sum of squares,
     - minimization is a least-squares problem, i.e. linear regression
     - "predicting" $X_{.j}$ from W.

$$V_{.j} = \left(U^T U\right)^{-1} U^T X.j$$

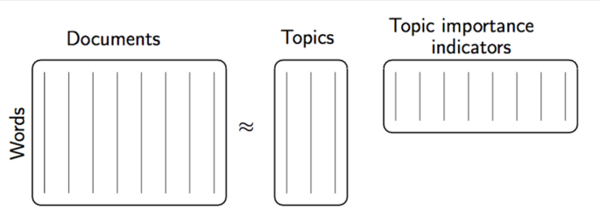
     - repeat for all columns $V_{.j}$
2. assume V, with U unknown: $X^T = VU^T$
   - Interchange roles of U, V in the above optimization
   - Compute a row of U, repeat for all rows

Alternating Least squares

- Putting all this together
    - random initialization of U and V
    - alternate:
        - Compute U assuming V known
        - Compute V based on that new U
        - ...
    - may generate negative values: truncate to 0

NMF issues, applications

- choice of NMF dimensionality
- U_{mxr}, r (rank) choice: via SVD ...
- Applications
  - Topic detection
  - Source separation (music, speech)
  - Clustering
  - Recommendations



t SNE

- based on the "Stochastic Neighbor Embedding" - Hinton, 2002.
- Stochastic Neighbor Embedding (SNE): map high-dimensional Euclidean point distances to conditional probabilities representing similarities.
- similarity between x_j, x_i - conditional probability, $p_{j|i}$ x_j, pick x_i as its neighbor , using a probability density under a Gaussian centered at x_i
- For nearby datapoints, $p_{j|i}$ is high vs. for widely separated datapoints, $p_{j|i}$ will be very low

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

For low-dimensional representations y_i, y_j resp. conditional probability $q_{j|i}$:

$$q_{j|i} = \frac{exp(-|y_i - y_j|^2/2\sigma_i^2)}{\sum_{k \neq i}(exp(-|y_i - y_k|^2/2\sigma_i^2)}$$

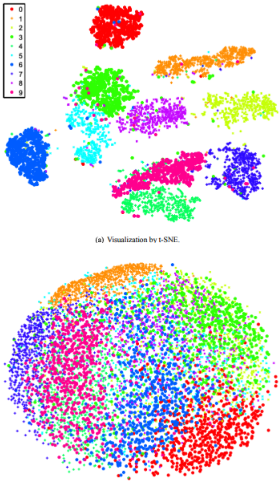
Stochastic Neighbor Embeeding, (NIPS 2002), Geoffrey E. Hinton, Sam Roweis

- For correct mapping: $p_{j|i} = q_{j|i}$, thus SNE finds a low-dimensional data representation minimising cost as Kullback Leibler divergence with gradient descent
- The cost function C:

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

The gradient of symmetric SNE: $\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - qji)(y_i - yj)$

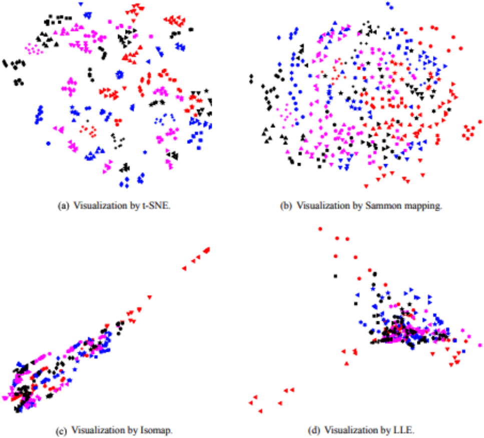
- Data Sets: **MNIST** data set,Olivetti faces, COIL-20 data set, the word-features data set, Netflix data set
- Baselines: ISOMAP, t-SNE, Sammon mapping, Isomap, LLE



(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

- Data Sets: MNIST data set,**Olivetti faces**, COIL-20 data set, the word-features data set, Netflix data set
- Baselines: ISOMAP, t-SNE, Sammon mapping, Isomap, LLE



SVD application - Latent Structure in documents[3]

- Documents are represented based on the Vector Space Model
- Vector space model consists of the keywords contained in a document.
- In many cases baseline keyword based performs poorly – not able to detect synonyms.
- Therefore document clustering is problematic
- Example where of keyword matching with the query: "IDF in computer-based information look-up"

	access	document	retrieval	information	theory	database	indexing	computer
Doc1	x	x	x			x	x	
Doc2				x	x			x
Doc3			x	x				x

[3] Indexing by Latent Semantic Analysis (1990) Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Journal of the American Society of Information Science

Latent Semantic Indexing (LSI) -I

- Finding similarity with exact keyword matching is problematic.
- Using SVD we process the initial document-term document.
- Then we choose the k larger singular values. The resulting matrix is of order k and is the most similar to the original one based on the Frobenius norm than any other k-order matrix.

Latent Semantic Indexing (LSI) - II

- The initial matrix is SVD decomposed as: $A = ULV^T$
- Choosing the top-k singular values from L we have:

$$A_k = U_k L_k V_k^\top$$

- L_k is square kxk containing the top-k singular values of the diagonal in matrix L,
- U_k, the mxk matrix containing the first k columns in U (left singular vectors)
- V_k^T, the kxn matrix containing the first k lines of V^T (right singular vectors) Typical values for k $\sim$ 200-300 (empirically chosen based on experiments appearing in the bibliography)

LSI capabilities

- Term to term similarity: $A_k A_k^T = U_k L_k^2 U_k^T$, $A_k = U_k L_k V_t$
- Document-document similarity: $A_k^T A_k = V_k L_k^2 V_k^T$
- Term document similarity (as an element of the transformed – document matrix)
- Extended query capabilities transforming initial query q to q_n: $q_n = q^T U_k L_k^{-1}$
- Thus q_n can be regarded a line in matrix V_k

LSI – an example

- LSI application on a term – document matrix
  - C1: Human machine Interface for Lab ABC computer application
  - C2: A survey of user opinion of computer system response time
  - C3: The EPS user interface management system
  - C4: System and human system engineering testing of EPS
  - C5: Relation of user-perceived response time to error measurements
  - M1: The generation of random, binary unordered trees
  - M2: The intersection graph of path in trees
  - M3: Graph minors IV: Widths of trees and well-quasi-ordering
  - M4: Graph minors: A survey
- The dataset consists of 2 classes, 1st: "human – computer interaction" (c1-c5) 2nd: related to graph (m1-m4). After feature extraction the titles are represented as follows.

LSI – an example

	C1	C2	C3	C4	C5	M1	M2	M3	M4
human	1	0	0	1	0	0	0	0	0
Interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	0	1	1	2	0	0	0	0	0
Response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	0	1	1	1
Minors	0	0	0	0	0	0	0	1	1

LSI – an example

- $A = ULV^T$
- $A =$

1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

LSI – an example

- $A = ULV^T$
- $U =$

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41	0	0	0
0.2	-0.07	0.14	-0.55	0.28	0.5	-0.07	-0.01	-0.11	0	0	0
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.3	0.06	0.49	0	0	0
0.4	0.06	-0.34	0.1	0.33	0.38	0	0	0.01	0	0	0
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27	0	0	0
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05	0	0	0
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05	0	0	0
0.3	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17	0	0	0
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58	0	0	0
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23	0	0	0
0.04	0.62	0.22	0	-0.07	0.11	0.16	-0.68	0.23	0	0	0
0.03	0.45	0.14	-0.01	-0.3	0.28	0.34	0.68	0.18	0	0	0

LSI – an example

- $A = ULV^T$
- $L =$

3.34	0	0	0	0	0	0	0	0
0	2.54	0	0	0	0	0	0	0
0	0	2.35	0	0	0	0	0	0
0	0	0	1.64	0	0	0	0	0
0	0	0	0	1.5	0	0	0	0
0	0	0	0	0	1.31	0	0	0
0	0	0	0	0	0	0.85	0	0
0	0	0	0	0	0	0	0.56	0
0	0	0	0	0	0	0	0	0.36
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

LSI – an example

- $A = ULV^T$
- $V =$

0.2	-0.06	0.11	-0.95	0.05	-0.08	0.18	-0.01	-0.06
0.61	0.17	-0.5	-0.03	-0.21	-0.26	-0.43	0.05	0.24
0.46	-0.13	0.21	0.04	0.38	0.72	-0.24	0.01	0.02
0.54	-0.23	0.57	0.27	-0.21	-0.37	0.26	-0.02	-0.08
0.28	0.11	-0.51	0.15	0.33	0.03	0.67	-0.06	-0.26
0	0.19	0.1	0.02	0.39	-0.3	-0.34	0.45	-0.62
0.01	0.44	0.19	0.02	0.35	-0.21	-0.15	-0.76	0.02
0.02	0.62	0.25	0.01	0.15	0	0.25	0.45	0.52
0.08	0.53	0.08	-0.03	-0.6	0.36	0.04	-0.07	-0.45

LSI – an example

- Choosing the 2 largest singular values we have

$U_k =$

0.22	-0.11
0.2	-0.07
0.24	0.04
0.4	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.3	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

$L_k =$

3.34	0
0	2.54

$V_k^T =$

0.2	0.61	0.46	0.54	0.28	0	0.02	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53

LSI (2 singular values)

- $A_k =$

	C1	C2	C3	C4	C5	M1	M2	M3	M4
human	0.16	0.4	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
Interface	0.14	0.37	0.33	0.4	0.16	-0.03	-0.07	-0.1	-0.04
Computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
User	0.26	0.84	0.61	0.7	0.39	0.03	0.08	0.12	0.19
System	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
Response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
Time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.2	-0.11
Survey	0.1	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
Trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
Graph	-0.06	0.34	-0.15	-0.3	0.2	0.31	0.69	0.98	0.85
Minors	-0.04	0.25	-0.1	-0.21	0.15	0.22	0.5	0.71	0.62

LSI Example

- Query: "human computer interaction" retrieves documents: c1, c2, c4 but *not* c3 and c5.
- If we submit the same query (based on the transformation shown before) to the transformed matrix we retrieve (using cosine similarity) all c1-c5 even if c3 and c5 have no common keyword to the query.
- According to the transformation for the queries we have:

Query transformation

	query
human	1
Interface	0
computer	1
User	0
System	0
Response	0
Time	0
EPS	0
Survey	0
Trees	0
Graph	0
Minors	0

$q =$

1
0
1
0
0
0
0
0
0
0
0
0

Query transformation

$q^T =$

1	0	1	0	0	0	0	0	0	0	0	0

$U_k =$

0.22	-0.11
0.2	-0.07
0.24	0.04
0.4	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.3	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

$L_k =$

0.3	0
0	0.39

$q_n = q^T U_k L_k =$

0.138	-0.0273

Map docs to the 2 dim space $V_k L_k =$

0.2	-0.06
0.61	0.17
0.46	-0.13
0.54	-0.23
0.28	0.11
0	0.19
0.01	0.44
0.02	0.62
0.08	0.53

3.34	0
0	2.54

0.67	-0.15
2.04	0.43
1.54	-0.33
1.8	-0.58
0.94	0.28
0	0.48
0.03	1.12
0.07	1.57
0.27	1.35

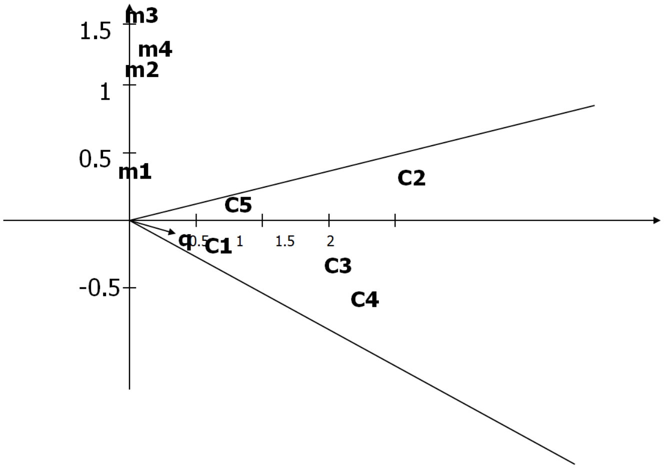
$q_n L_k =$

0.138	-0.0273

3.34	0
0	2.54

0.46	-0.069

Query transformation



Query transformation

- Comparison of the transformed query to the new document vectors based on cosine similarity, where the similarity is computed as:

$$\text{Cos}\,(x, y) = < x, y > /|x|| \cdot ||y||$$

Where $x = (x_1, \ldots, x_n)\,, y = (y_1, \ldots, y_n)$

$$< x, y >= x_1 * y_1 + \ldots + x_n * y_n$$

Query transformation

- The cosine similarity matrix of query vector to the documents is:

	query
C1	0.99
C2	0.94
C3	0.99
C4	0.99
C5	0.9
M1	-0.14
M2	-0.13
M3	-0.11
M4	0.05