Dimensionality Reduction

Prof. Dr. Renato Pajarola
Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

Copyrights

- Most figures of these slides are copyright protected and come from the various indicated sources

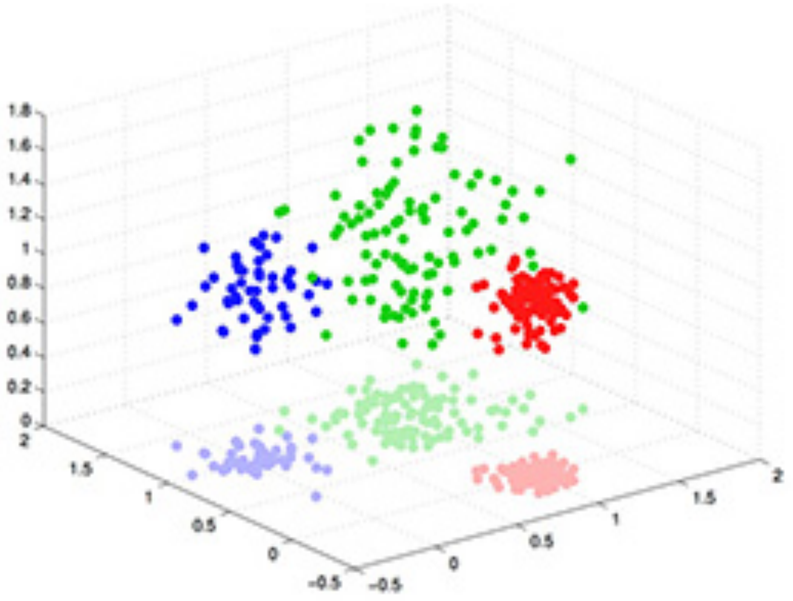
- You understand that the slides contain copyright protected material and therefore the following conditions of use apply:
  - The slides may be used for personal teaching purposes only
  - Publishing the slides to any public web site is not allowed
  - Sharing the slides with other persons or institutions is prohibited

Overview

1. Dimensionality reduction
2. Low-rank SVD approximation
3. Principal component analysis
4. Multidimensional scaling
5. Self-organizing maps

Dimensionality Reduction

- Given a set of high-dimensional data points $\boldsymbol{x}_i \in \mathbb{R}^K$ (for $i=1 \ldots N$), dimensionality reduction methods compute a projection $\boldsymbol{y}_i \in \mathbb{R}^{1,2 \text{ or } 3}$ into a lower dimensional Euclidean space
  - For 1D charts and curve plots, 2D scatter or height-plots and maps, or visualizations in 3D
- Key goal: preserve the relative relations between projected data points $\boldsymbol{y}_i$ in the low-dimensional space as given by the data records $\boldsymbol{x}_i$ in their original high-dimensional input space
  - E.g. preserve relative distances, outliers, clusters, patterns

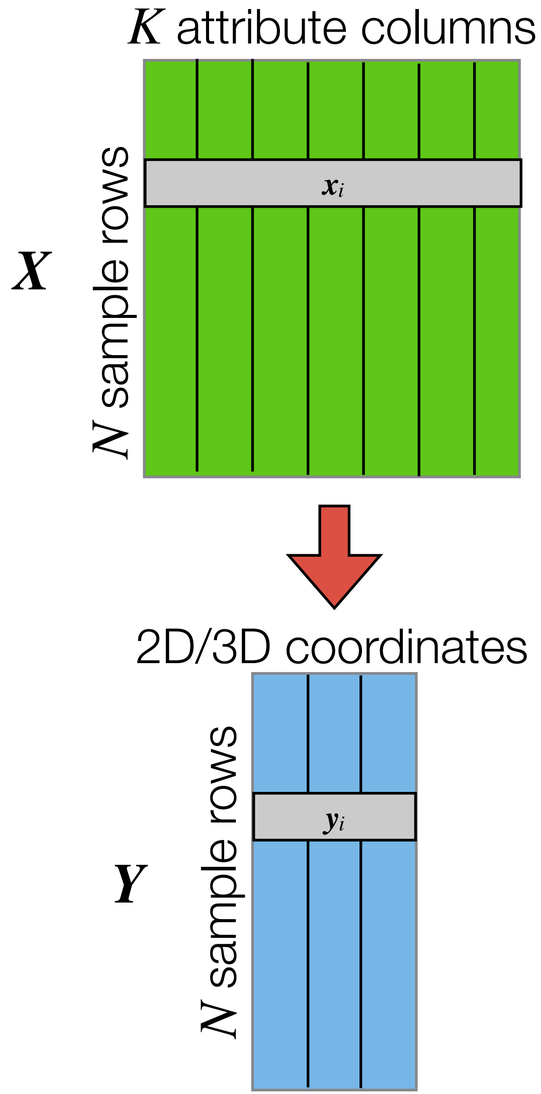


Source http://bigdata.csail.mit.edu/node/277

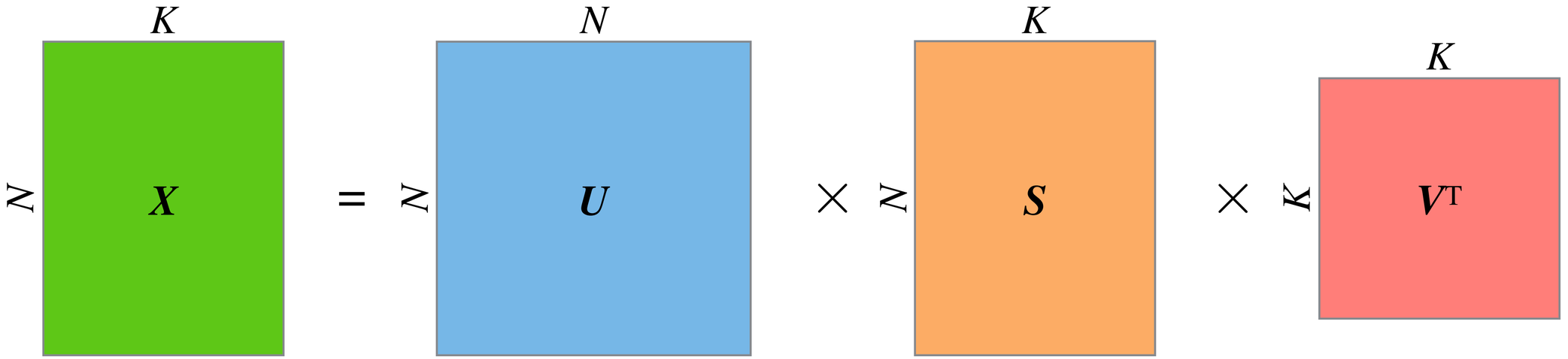
- Mapping high-dimensional data points into 2D or 3D is a very important step for data visualization

Dimensionality Reduction

- Two main strategies: *feature selection* and *feature extraction*
- Feature selection: find a subset of the original variables/dimensions
  - E.g. using information gain or guided search methods
- Feature extraction: transform data from original space to a space of fewer dimensions
- Express input data vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots \boldsymbol{x}_N\}$ as a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ with N rows and K columns and each $\boldsymbol{x}_i \in \mathbb{R}^K$
  - Data items = rows
  - Data attributes = columns
- The sample mean of each column of $\boldsymbol{X}$ is typically shifted to zero
  - Column-wise zero mean extracted and subtracted beforehand
- Reduced dimensionality output data vectors $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots \boldsymbol{y}_i\}$ as a matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times \{1,2,3\}}$ with $\boldsymbol{y}_i \in \mathbb{R}^{\{1,2,3\}}$

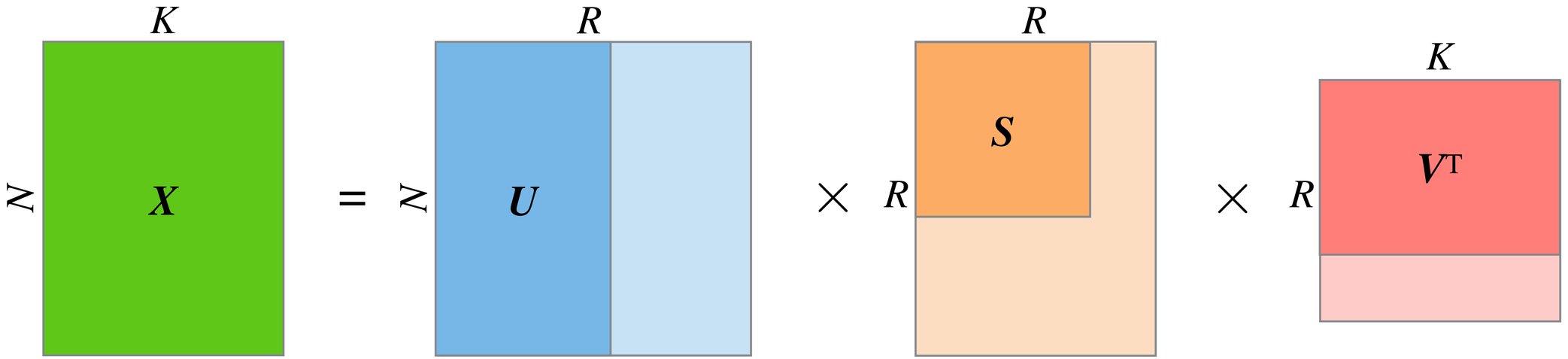


Singular Value Decomposition (SVD)



- Singular Value Decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}$ represents $\boldsymbol{X}$ in terms of *singular vectors* and *singular values*
  - Matrices $\boldsymbol{U}$ and $\boldsymbol{V}$: left and right singular vectors
    - both column-orthonormal
  - (Generalized) diagonal matrix $\boldsymbol{S}$: the K ordered, non-zero singular values $\sigma_{i=1\ldots K}$
- SVD represents high-dimensional data points (matrix) $\boldsymbol{X}$ *exactly*
  - Plain decomposition, transformation into different data representation space

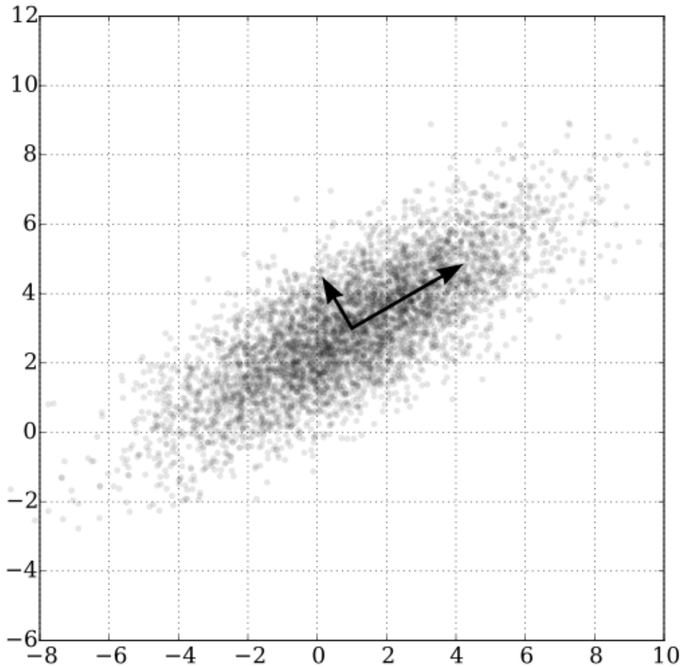
Low-rank Approximation using SVD



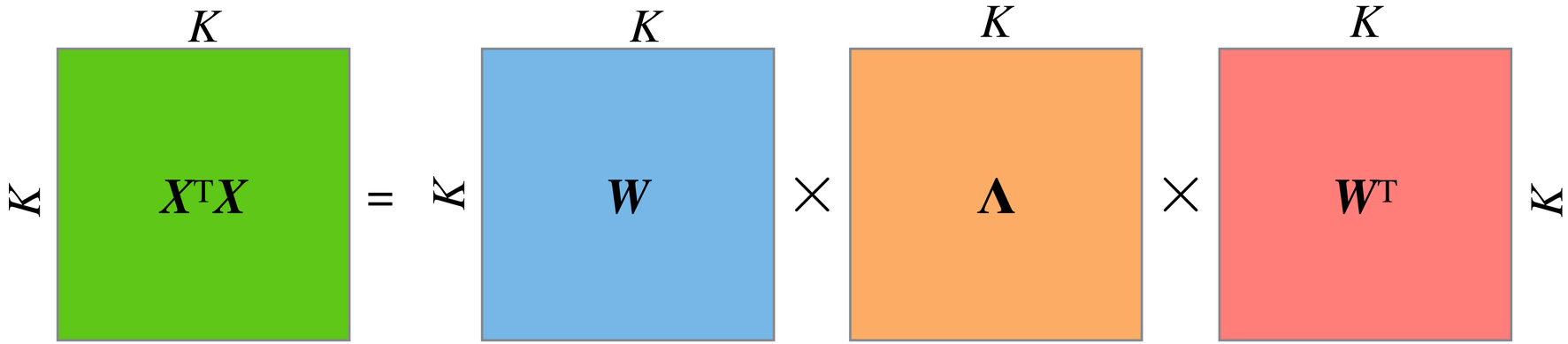
- We can *approximate* matrix $\boldsymbol{X}$ by using fewer singular values and vectors of the SVD decomposition
  - Use only $R < K$ singular values/vectors, i.e. take sub-matrices of $\boldsymbol{U}$, $\boldsymbol{S}$, $\boldsymbol{V}$
  - *Low-rank* approximation
- Original high-dimensional data points $\boldsymbol{X}$ can be represented in low-dimensional display space as points $\boldsymbol{Y} = \boldsymbol{US}$ (or $\boldsymbol{Y} = \boldsymbol{U}$)
  - E.g. keeping only the $R \leq 3$ most significant singular values of $\boldsymbol{S}$ and columns of $\boldsymbol{U}$

Principal Component Analysis (PCA)

- *Principal Component Analysis* is the main linear technique for dimensionality reduction
- Performs a linear mapping of the high-dimensional input data to a lower-dimensional output space
  - Transforms the data into a new coordinate system
  - New coordinates known as *principal components*
  - First principal component has highest possible variance
  - Each succeeding component has highest variance possible while being orthogonal to the preceding components
- Corresponds to fitting a *K*-dimensional ellipsoid to the data
  - Each axis corresponds to a principal component
- Can be done by constructing the *covariance matrix* of the data and computing its eigenvectors
  - Eigenvectors of the largest 2 or 3 eigenvalues define subspace of maximal variance of the original data
- Also known as *Karhunen–Loève transform* (KLT)

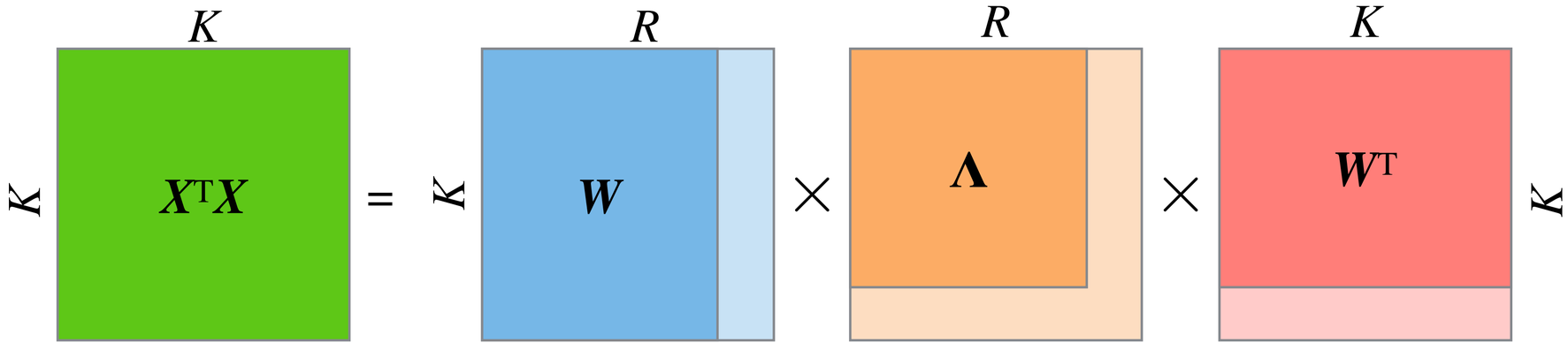


Eigendecomposition-based PCA



- The covariance matrix $M = X^TX$ of the data can be analyzed to find the principal components
  - Requires the data X to be column-wise mean-centered
- Eigendecomposition $X^TX = W\Lambda W^T$
  - Diagonal matrix Λ may only have $j \leq K$ non-zero eigenvalues $\lambda_{i=1..j}$
- Data points projected in new coordinate system: $X_T = XW$

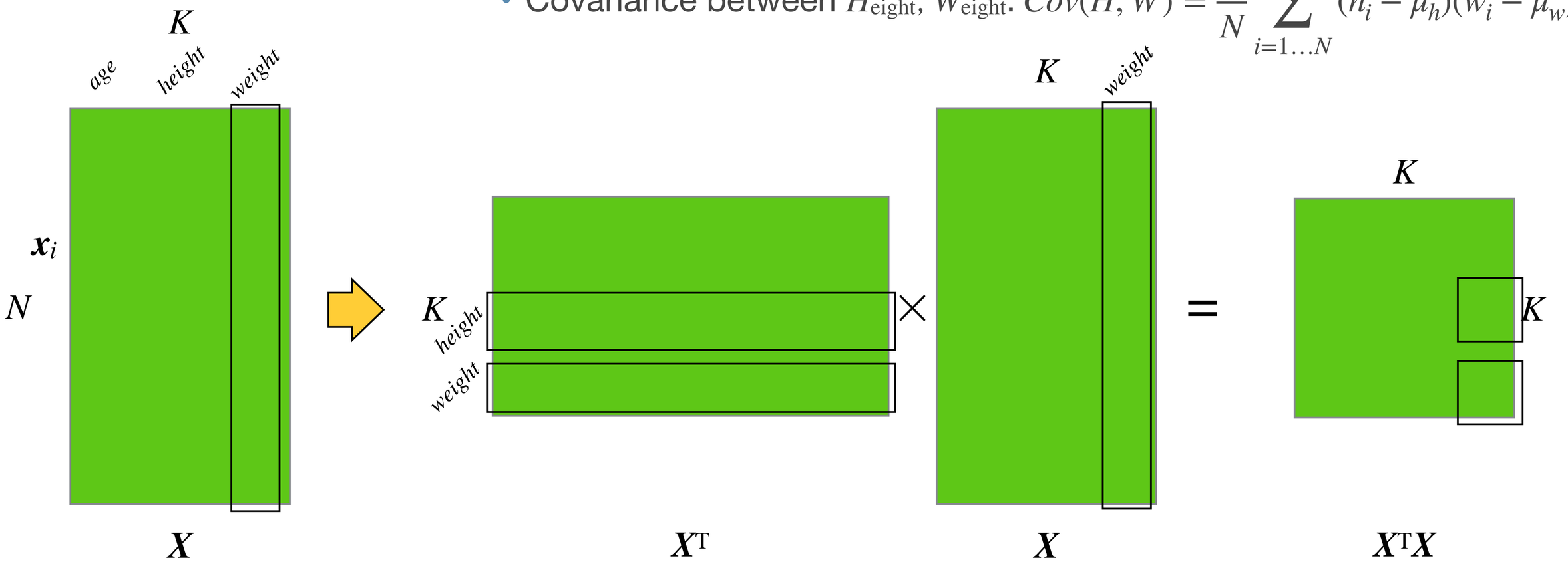
Eigendecomposition-based PCA



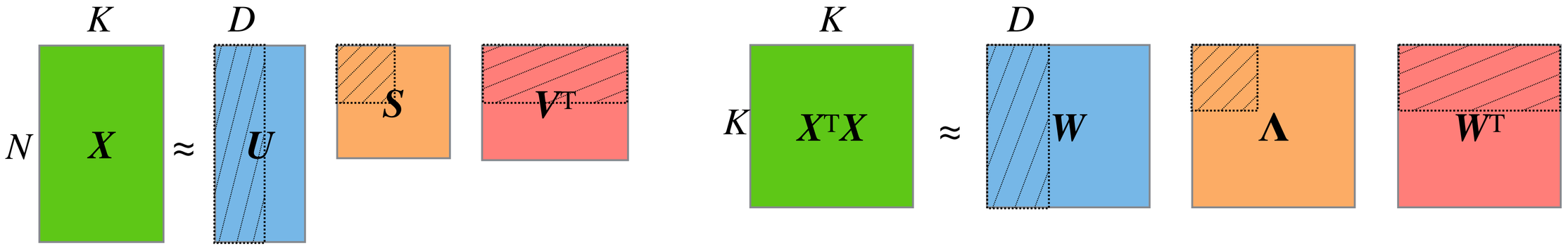
- The covariance matrix $\boldsymbol{M} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ of the data can be analyzed to find the principal components
  - Requires the data $\boldsymbol{X}$ to be column-wise mean-centered
- Eigendecomposition $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^{\mathrm{T}}$
  - Diagonal matrix $\boldsymbol{\Lambda}$ may only have $j \leq K$ non-zero eigenvalues $\lambda_{i=1..j}$
- Data points projected in new coordinate system: $\boldsymbol{X}_{\mathbf{T}} = \boldsymbol{X}\boldsymbol{W}$
- Reduced dimensionality data points given by $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}_R$
  - E.g. keeping only the $R \leq 3 \leq K$ most significant eigenvalues of $\boldsymbol{\Lambda}$ and columns of $\boldsymbol{W}$

Covariance Matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$

- $\boldsymbol{X}$ is mean-centered
- Variance of variable W_{eight}: $Var(W) = \frac{1}{N}\sum_{i=1\ldots N}(w_i = \mu_w)^2$
- Covariance between H_{eight}, W_{eight}: $Cov(H, W) = \frac{1}{N}\sum_{i=1\ldots N}(h_i - \mu_h)(w_i - \mu_w)$



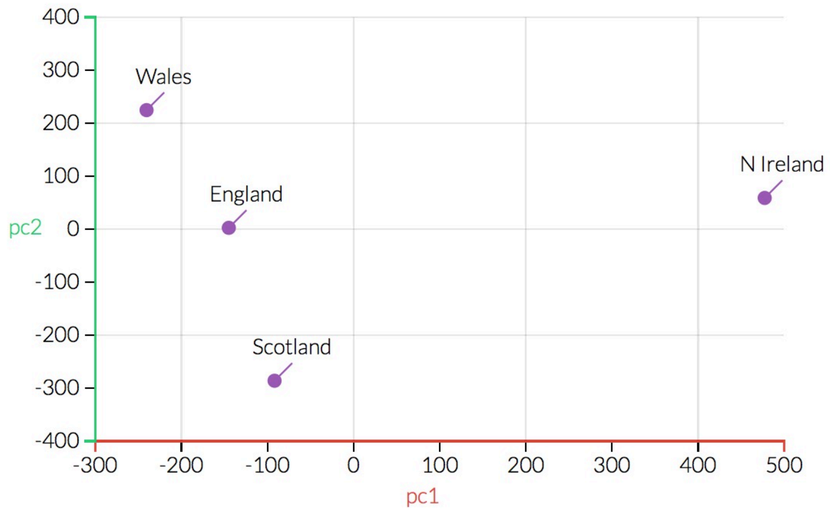
Eigendecomposition SVD Equivalence



- Eigendecomposition equivalence to SVD: $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}_D \Leftrightarrow \boldsymbol{Y} = \boldsymbol{U}_D\boldsymbol{S}_D$
  - Keeping only the $D \leq K$ most significant dimensions, columns of $\boldsymbol{W}$, or $\boldsymbol{U}$ and $\boldsymbol{S}$
  - Use $\boldsymbol{W}$ if other (new) data $\boldsymbol{X}_{new}$ has to be projected into the same space
- $\boldsymbol{U}$ has unit columns, $\boldsymbol{Y} = \boldsymbol{U}_D$ thus represents the data $\boldsymbol{X}$ in a 1.0 x… x 1.0 D-dimensional scatterplot
  - Projects data $\boldsymbol{X}$ into a unit D-dimensional box
- Keeping e.g. 95% of the variance in the data = keeping all eigenvalues λ_i for which $\sum_{i=1}^{D} \lambda_i \Big/ \sum_{i=1}^{K} \lambda_i \geq 95\,\%$
  - Or the corresponding to singular values $\sigma_i = \sqrt{\lambda_i}$
    - σ_i from SVD of $\boldsymbol{X}$ and λ_i from eigendecomposition fo $\boldsymbol{X}^T\boldsymbol{X}$

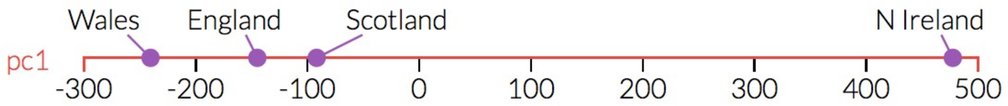
17D Example: Eating in the UK

- Average consumption of 17 types of food in grams per person per week for every country in the UK
- Clear difference of Northern Ireland in 1D projection
- Visible outlier or special case in 2D projection
  - From the table, this makes sense: Northern Irish eat way more grams of fresh potatoes and way fewer of fresh fruits, cheese, fish and alcoholic drinks

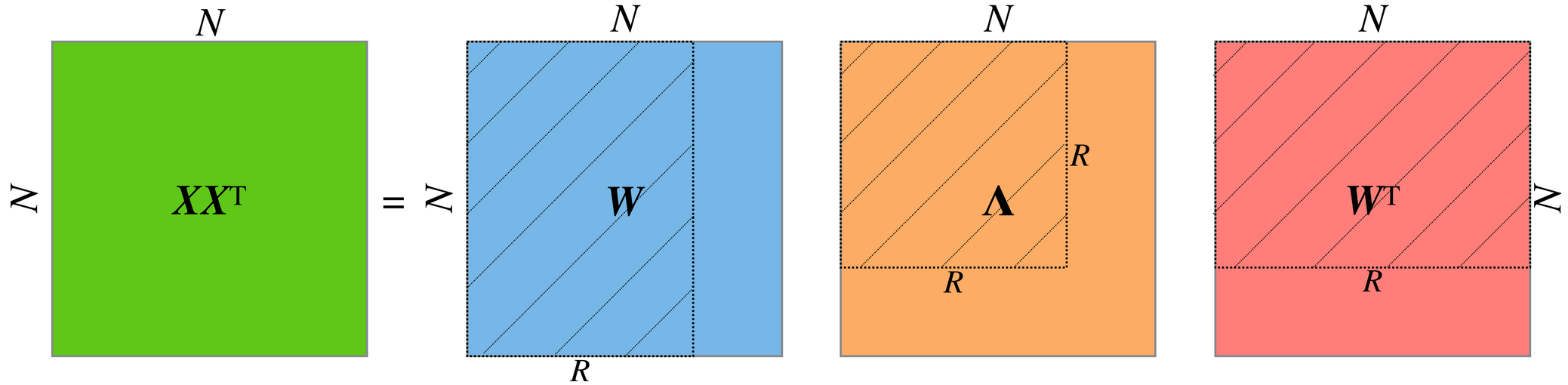


	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472	1494	1462	1582
Cheese	105	66	103	103
Confectionery	54	41	62	64
Fats and oils	193	209	184	235
Fish	147	93	122	160
Fresh fruit	1102	674	957	1137
Fresh potatoes	720	1033	566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

http://setosa.io/ev/principal-component-analysis/



Using Gram, Hermitian and Self-adjoint Matrix



- The eigenvalues of matrix $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{X}^T$ are the eigenvalues of $\boldsymbol{M} = \boldsymbol{X}^T\boldsymbol{X}$ plus additional 0s
  - Or opposite more dimensions than elements, i.e. if $K > N$
  - Matrix of all pairwise inner products
- Given an eigenvector $\boldsymbol{e}$ of $\boldsymbol{W}$ (from $\boldsymbol{X}^T\boldsymbol{X}$), then $\boldsymbol{Me}$ is an eigenvector of $\boldsymbol{X}\boldsymbol{X}^T$and the corresponding eigenvalue from both $\boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{X}\boldsymbol{X}^T$
  - If not zero
    - reverse holds too
- Thus all eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ are also eigenvalues of $\boldsymbol{X}\boldsymbol{X}^T$

MultiDimensional Scaling (MDS)

- MDS computes low dimensional representation based on *pairwise distances* between data points
  - Particularly useful to visualize information contained in a distance matrix
- Key goal: preserve original distances as well as possible
  - Map positions of original data points based on distances
  - In fact, positions are computed solely from pairwise distances
- Problem: given matrix $\mathbf{D} = (d_{ij})\ 1 \leq i, j \leq N$ compute $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \in \mathbb{R}^R$ s.t. $d_{ij} \approx \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2$
  - e.g. $R = 2$
- Different versions of MDS set different requirements on the approximation of the given distances
  - Classical MDS
  - Metric MDS
  - Non-metric MDS

Classical MDS (cMDS)

- *Classical* MDS (cMDS) aims to compute positions $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \in \mathbb{R}^R$ such that $\tilde{d}_{ij} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2 \approx d_{ij}$
- cMDS formulation based on the assumption that the distances d_{ij} are Euclidean distances
- The values d_{ij} can be *dissimilarities*, not necessarily *distances*
  - For large target dimensions R, there often exists $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ s.t. $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2 = d_{ij}$
    - d is an actual Euclidean distance
  - In other cases, e.g. $R = 2$, MDS finds an approximation
- Solution not unique, consider centered configuration
- Define centering matrix $\mathbf{C}$ as follows:

$$\mathbf{C} = \mathbf{I}_N - \frac{1}{N} \cdot \mathbf{1}_N \mathbf{1}_N^T$$

$\mathbf{C}$ is NxN identity matrix minus an all-ones-over-N

  - Can be used to make matrix of observations mean-centered
    - i.e. given $\mathbf{X} \in \mathbb{R}^{N \times K}$, rows of $\mathbf{CX}$ are mean-centered

Classical MDS (cMDS)

- Compute the coordinates of the points from the distance matrix $\mathbf{D} = (d_{ij})$
- It can be shown that:

$$-\frac{1}{2}\mathbf{C}\,\mathbf{D}^2\,\mathbf{C} = \mathbf{X}\,\mathbf{X}^T = \mathbf{B}$$

  - Applied *double-centering* on squared input distance matrix

- By eigenvector decomposition of $\mathbf{B}$, we can obtain:

$$\mathbf{B} = \mathbf{W}\,\mathbf{\Lambda}\,\mathbf{W}^T = \mathbf{W}\,\mathbf{\Lambda}^{1/2}\,\mathbf{\Lambda}^{1/2}\,\mathbf{W}^T = \mathbf{X}\,\mathbf{X}^T$$

- The exact positions of the data points are therefore: $\mathbf{X} = \mathbf{W}\,\mathbf{\Lambda}^{1/2}$
- Keep only the first R eigenpairs to obtain the desired positions in the low-dimensional space:

$$\mathbf{Y} = \mathbf{W}_R\,\mathbf{\Lambda}^{1/2}$$

Classical MDS (cMDS)

- Main steps of cMDS:
  1. Compute squared distance matrix $\mathbf{D}^2$ (i.e. square the input distance matrix)
  2. Compute matrix $\mathbf{B} = -\frac{1}{2}\mathbf{C}\,\mathbf{D}^2\,\mathbf{C}$, with $\mathbf{C} = \mathbf{I}_N - \frac{1}{N} \cdot \mathbf{1}_N\mathbf{1}_N^T$
  3. Compute eigendecomposition $\mathbf{B} = \mathbf{W}\,\mathbf{\Lambda}\,\mathbf{W}^T$
  4. Compute low-dimensional positions $\mathbf{Y} = \mathbf{W}_R\,\mathbf{\Lambda}^{1/2}$

Classical vs Metric MDS

- Classical MDS corresponds to minimizing a specific loss function termed *strain*, given $\mathbf{B} = (b_{ij})$

$$\text{Strain}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) = \sqrt{\frac{\sum_{i,j} (b_{ij} - \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle)^2}{\sum_{i,j} b_{ij}^2}}$$

- Metric MDS generalizes the loss function used in the optimization
  - E.g. using *stress* function based on sum of squared residuals between original and "new" distances

$$\text{Stress}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) = \sqrt{\frac{\sum_{i,j} (d_{ij} - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|)^2}{\sum_{i,j} d_{ij}^2}}$$

Non-Metric MDS

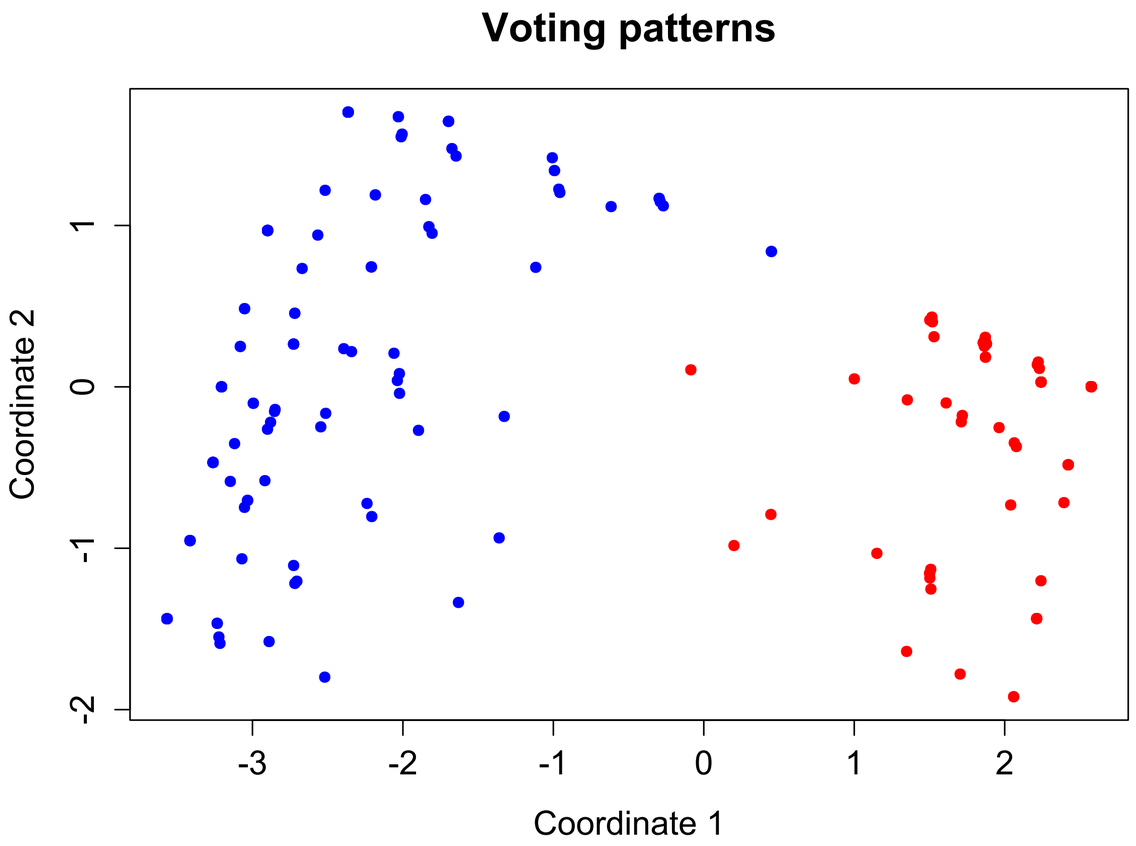
- Works with input dissimilarities δ_{ij} that only correspond to a ranking between original points
  - Spacing between successive dissimilarities is not interesting/not available
- Further generalization of stress function
  - Based on a monotonically increasing function θ of input dissimilarities

$$\text{Stress}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) = \sqrt{\frac{\sum_{i,j} \left(\theta(\delta_{ij}) - \|\boldsymbol{y}_i - \boldsymbol{y}_j\| \right)^2}{\sum_{i,j} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2}}$$

- Starting from initial coordinates, iteratively adjust to minimize stress function
  - Different numerical optimization strategies possible
    - gradient descent, conjugate gradient, simulated annealing etc.
  - Initialization random or based on cMDS

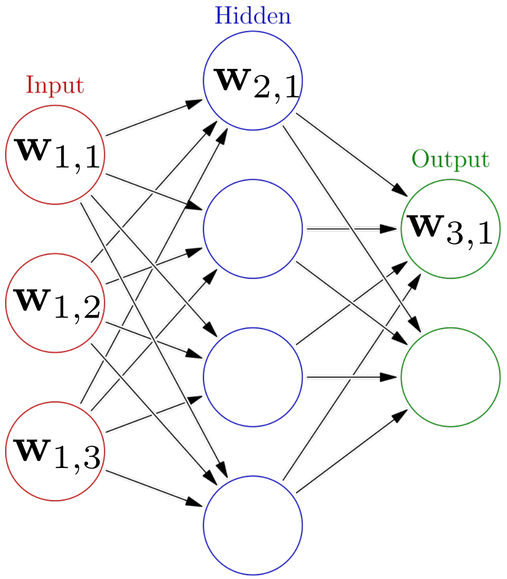
MDS Example

- Classical MDS of vote data from the *United States House of Representatives*
  - Each data record represents one Member of Congress
    - color represents political party (red for Republican, blue for Democrat).
- Used votes (K=10 binary data attributes):
  - H.R. 2642: Federal Agriculture Reform and Risk Management Act of 2013
  - H.R. 2609: Energy and Water Development and Related Agencies Appropriations Act
  - H.R. 1564: Audit Integrity and Job Protection Act
  - H.R. 1797: Pain-Capable Unborn Child Protection Act
  - H.R. 3: Northern Route Approval Act
  - H.R. 1911: Improving Postsecondary Education Data for Students Act
  - H.R. 1256: Swap Jurisdiction Certainty Act
  - H.R. 1960: National Defense Authorization Act for Fiscal Year 2014
  - H.R. 45: To repeal the Patient Protection and Affordable Care Act
  - H.R. 1062: SEC Regulatory Accountability Act

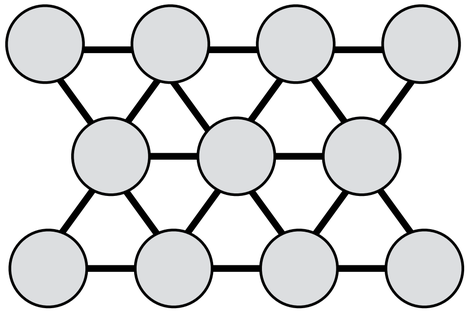


Self-Organizing Maps (Kohonen Maps)

- Dimensionality reduction technique based on Artificial Neural Networks (ANNs)
- Artificial Neural Network:
  - Collection of neurons (or nodes), interconnected to form a network
  - Each neuron produces a response as function of input signal and a vector of weights
    - each neuron holds a weight vector
  - Network corresponds to a function on the input
  - Function can be learned by adjusting weights during a training phase
    - weights modified so that output produced for a certain input matches known output for that input



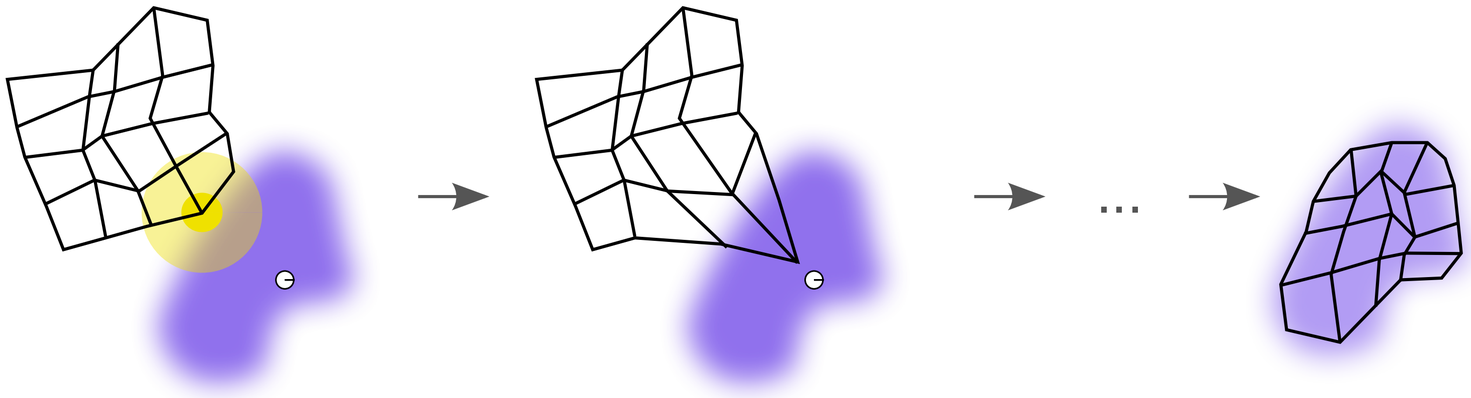
- A Self-Organizing Map (SOM) is a special type of ANN
  - Based on unsupervised learning
  - Uses N_{neurons} neurons arranged on a low-dimensional map space with certain topology
    - e.g. 1D line, 2D rectangular or hexagonal grid
  - Each neuron has a weight vector $\boldsymbol{w} \in \mathbb{R}^K$ of same size of input data vector $\boldsymbol{x} \in \mathbb{R}^K$

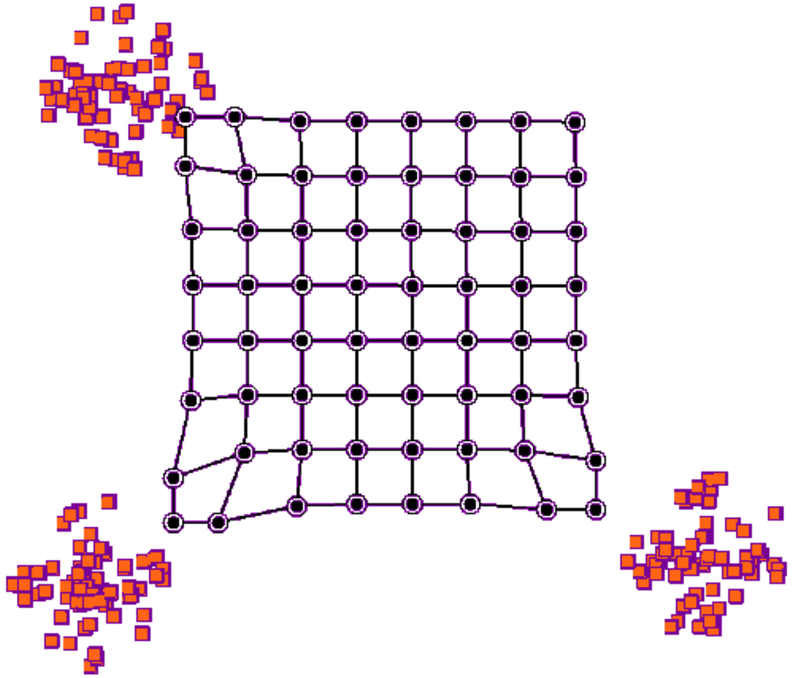


Self-Organizing Maps (Kohonen Maps)

- Given a collection of input samples $X = \{x_i\}$, SOMs are trained so that different parts of the network respond similarly to similar patterns in input data
  - Iteratively modify weights w_j of neurons so that they match input data vectors x_i
  - Neighboring vectors in input space, with similar x, should be mapped to neighboring neurons
- If network has lower dimensionality than input data dimensions, only the most important similarity relationships are preserved and mapped onto network neighborhood
- Training performed by iterating over individual samples x_i in multiple cycles ($N_{\text{timesteps}}$ time steps)
  - Start with random weight vectors w_j for all neurons
    - other initialization possibles
  - For each sample x_i at timestep t, find and update neurons with weights w_j closest to x_i
    - requires defining similarity weights/samples, neighborhood system, update rules
- Given trained SOM, an input point i gets mapped to the neuron j with most similar attribute vector
  - Thus with w_j most similar to x_i

SOM Training





SOM Training

Initialize neuron weight vectors $\boldsymbol{w}_i \in \mathbb{R}^K$

for $t = 1$ to $N_{\text{timesteps}}$

  for each $\boldsymbol{x}_i$ in set of input points $\boldsymbol{X}$

    for $k = 1$ to N_{neurons}

      compute distance d_k between $\boldsymbol{x}_i$ and $\boldsymbol{w}_k$

    end for

    find id k_{BMU} of Best Matching Unit (BMU)

    for $k = 1$ to N_{neurons}

      update weight vector $\boldsymbol{w}_k$

    end for

  end for

end for

e.g. Euclidean distance: $d_k = \|\boldsymbol{x}_i - \boldsymbol{w}_k\|$

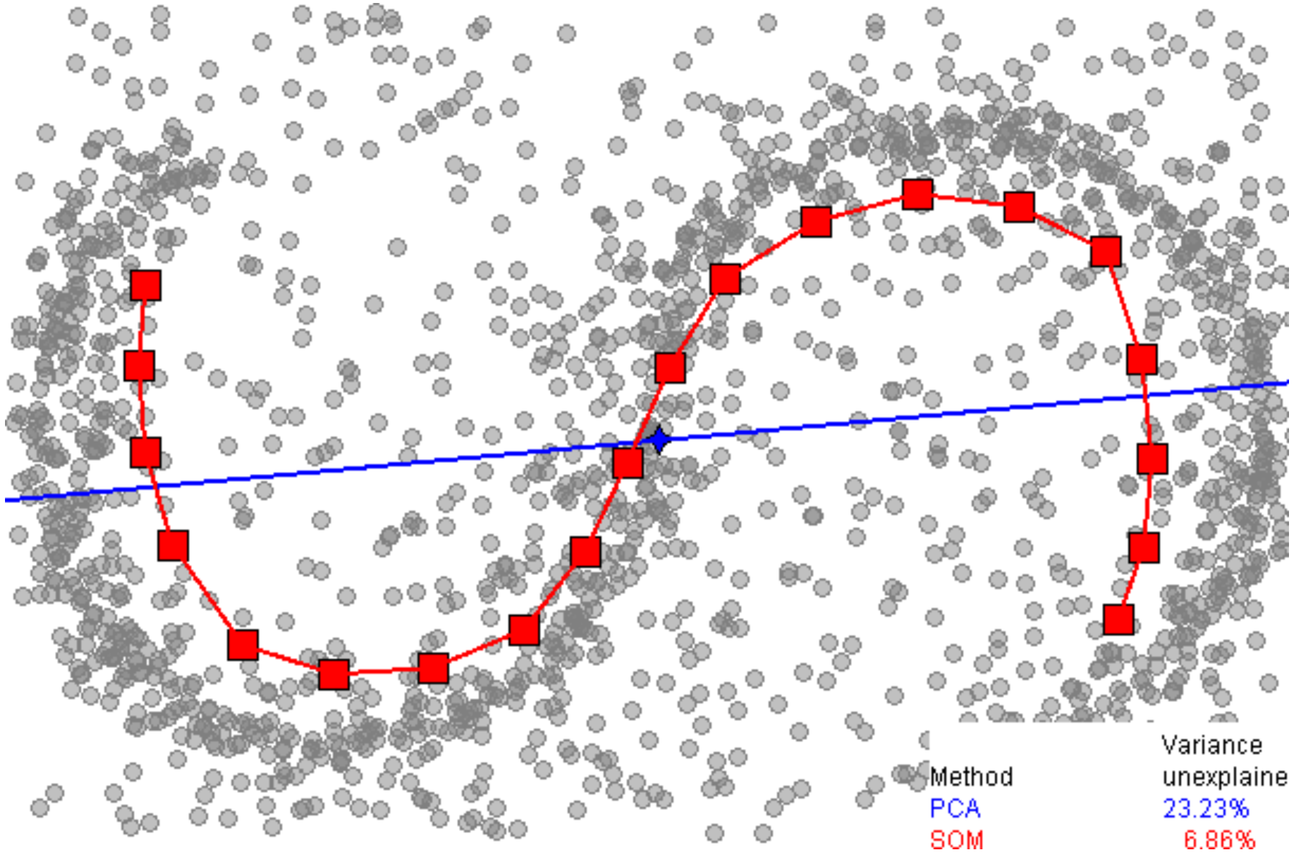
$$k_{BMU} = \arg\min_k \{d_k\}$$

$$\boldsymbol{w}_k = \boldsymbol{w}_k + \Theta(k, k_{BMU}, t) \cdot \alpha(t) \cdot (\boldsymbol{x}_i - \boldsymbol{w}_k)$$

decreasing function of distance between neurons k and k_{BMU} on neighborhood system of the SOM

size of neighborhood decreases with time

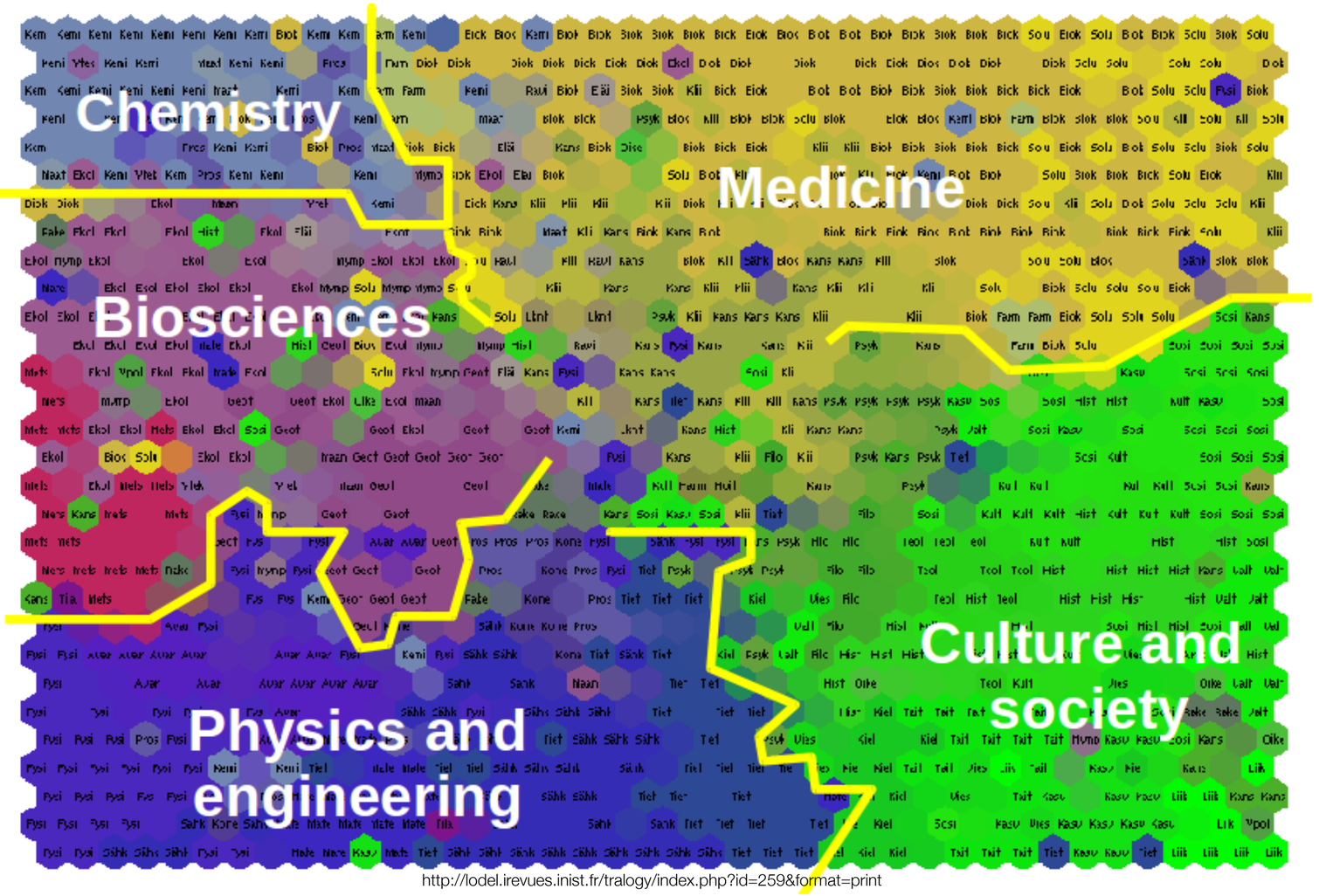
learning rate factor; decreases with time



- One-dimensional SOM versus principal component analysis (PCA)
  - The SOM is the red line with squares, 20 nodes
  - The first principal component is presented by a blue line
  - Data points are the small grey circles
- The fraction of variance unexplained in this example is 23.23% for PCA and 6.86% for SOM

SOM Example

- Map of documents in which two documents are close to each other in the two-dimensional space if their contents are similar
  - Represent documents as the occurrences of words, words that occur in same kinds of contexts tend to have similar meanings
- Collection of 3224 applications sent to Academy of Finland
  - Likey (Language Independent Keyphrase Extraction) extracts relevant terminology
    - finds terms by comparing word frequency with a large reference corpus



http://lodel.irevues.inist.fr/tralogy/index.php?id=259&format=print

- The Likey method is able to find automatically terms (words or phrases) using an approach in which the frequency or rank of a word or phrase in the corpus at hand is compared with that in a large reference corpus

Recap

- **Review of matrix SVD:** singular value decomposition, low-rank approximation using SVD
- **Principal component analysis:** eigendecomposition based PCA, covariance matrix, eigendecomposition and SVD equivalence
- **Multidimensional scaling:** classical, metric and non-metric MDS
- **Self-organizing maps:** artificial neural network based SOMs (aka Kohonen maps), iterative training of SOM's network weight vectors

- Required textbook Chapter(s): 6