Factorization, Principal Component Analysis and Singular Value Decomposition

Florian Buettner/Volker Tresp

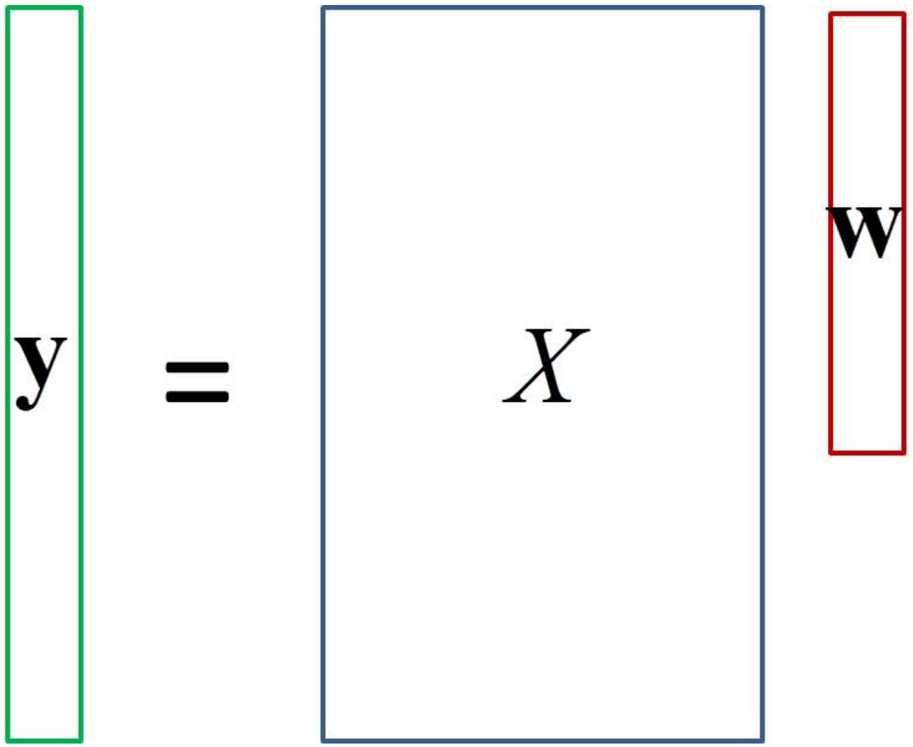
Summer 2018

buettner.florian@siemens.com

Recall: Linear Regression

- Predict single output $\mathbf{y}$ based on weighted sum of input $\mathbf{X}$ (design matrix)
- Many inputs and one output

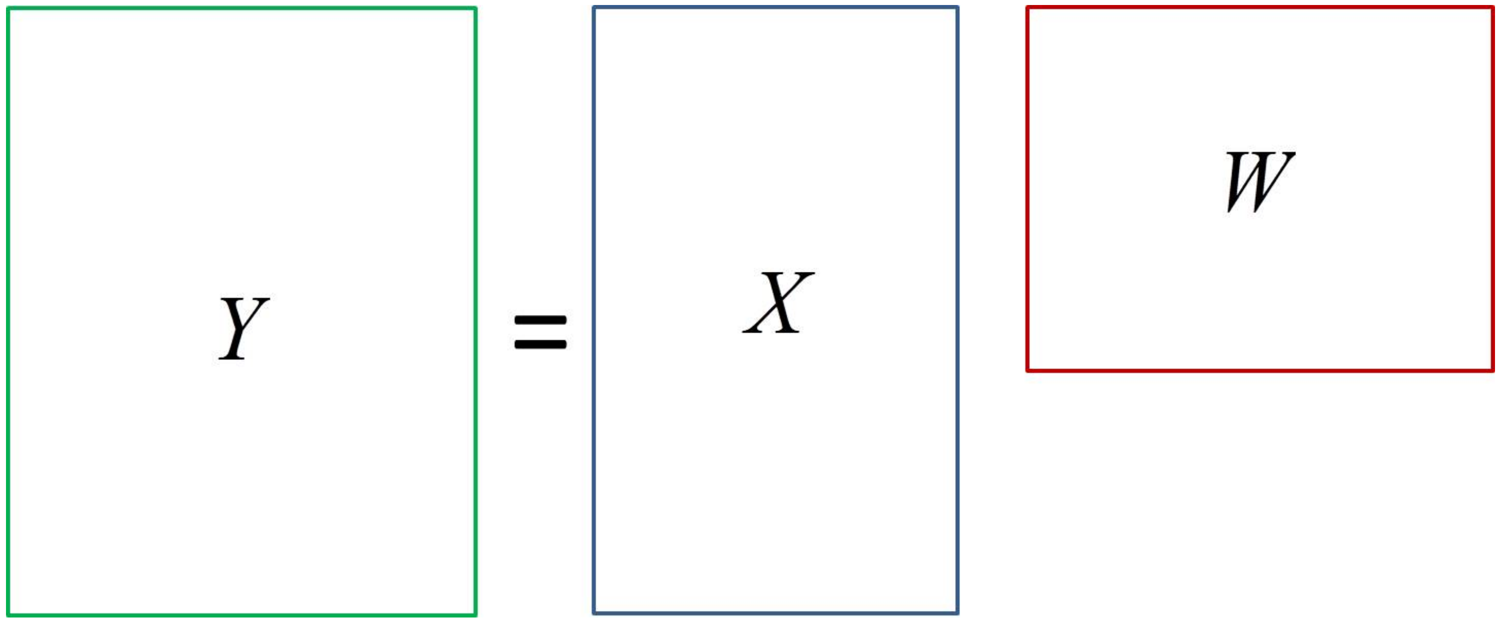
$$\mathbf{y} \approx X\mathbf{w}$$



Multivariate Linear Regression: Linear Regression with Multiple Outputs

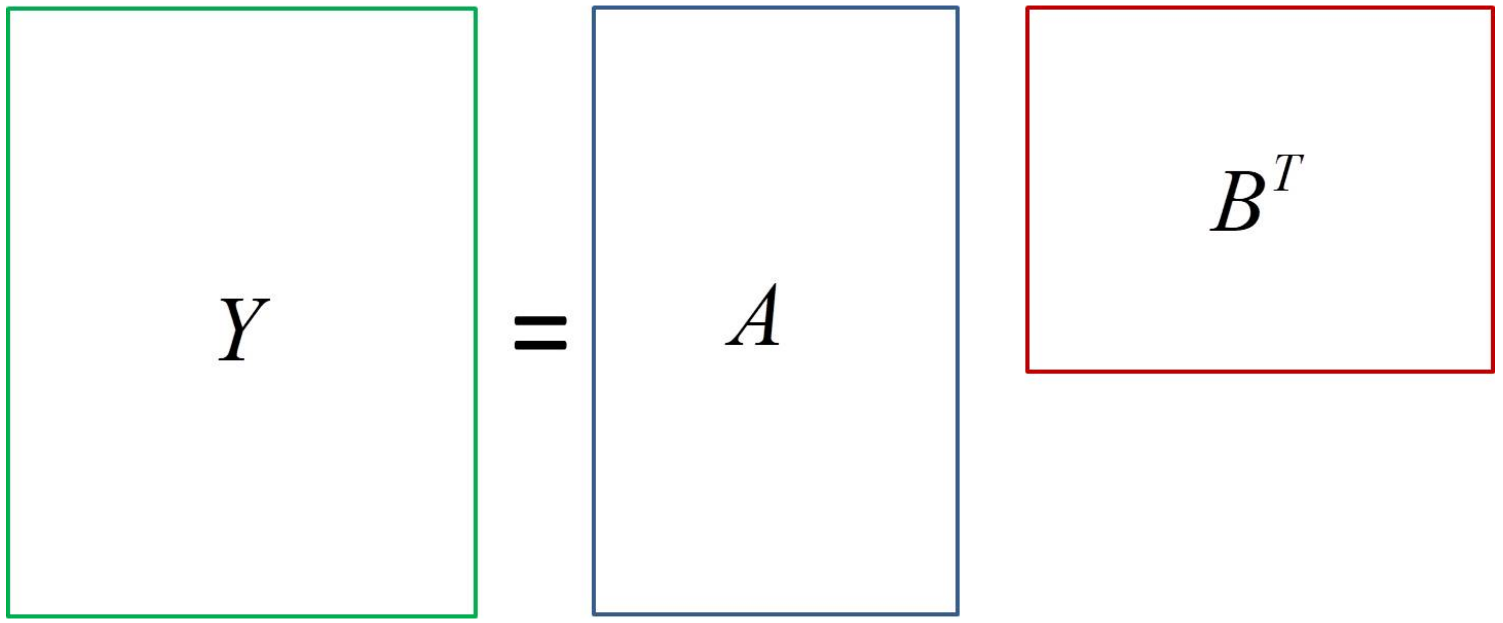
- Predict multiple outputs $\mathbf{Y}$ as on weighted sums of input $\mathbf{X}$ (design matrix)
- Many inputs and many outputs

$$Y \approx XW$$



Unknown Inputs

- *Now we assume that the inputs X are also unknown*
- We change the notation and write $A = X$ and $B = W^T$ and get



Example: Reconstruct missing values in Y

- Each row of Y corresponds to a user, each column of Y corresponds to a movie and $y_{i,j}$ is the rating of user i for movie j
- Thus the i-th row of A describes the latent attributes or **latent factors of the ith user** and the j-th row of B describes the latent attributes or **latent factors of the movie** associated with the j-th movie

Cost Function

- To find $\mathbf{A}$ and $\mathbf{B}$ we can define a least-squares cost function with regularisation terms

$$\sum_{(i,j)\in\mathcal{R}} \left(y_{i,j} - \sum_{k=1}^{r} a_{i,k} b_{j,k} \right)^2 + \lambda \sum_{i=1}^{N} \sum_{k=1}^{r} a_{i,k}^2 + \lambda \sum_{j=1}^{M} \sum_{k=1}^{r} b_{j,k}^2$$

  Here $\mathcal{R}$ is the set of existing ratings, r (rank) is the number of latent factors, and λ is a regularization parameter

- Note, that the cost function ignores movies which have not been rated yet and treats them as missing

- A and B are found via stochastic gradient descent

- After convergence, we can predict for any user and any movie

$$\widehat{y}_{i,j} = \sum_{k=1}^{r} a_{i,k} b_{j,k}$$

Symmetry of the Factorization

- Matrix factorization was the most important component in the winning entries in the Netflix competition
- Note that the i-th row of A contains the latent factor of user i (capturing coordinated variation across movies) and the j-th row of B contains the latent factors of movie j (variation across users)

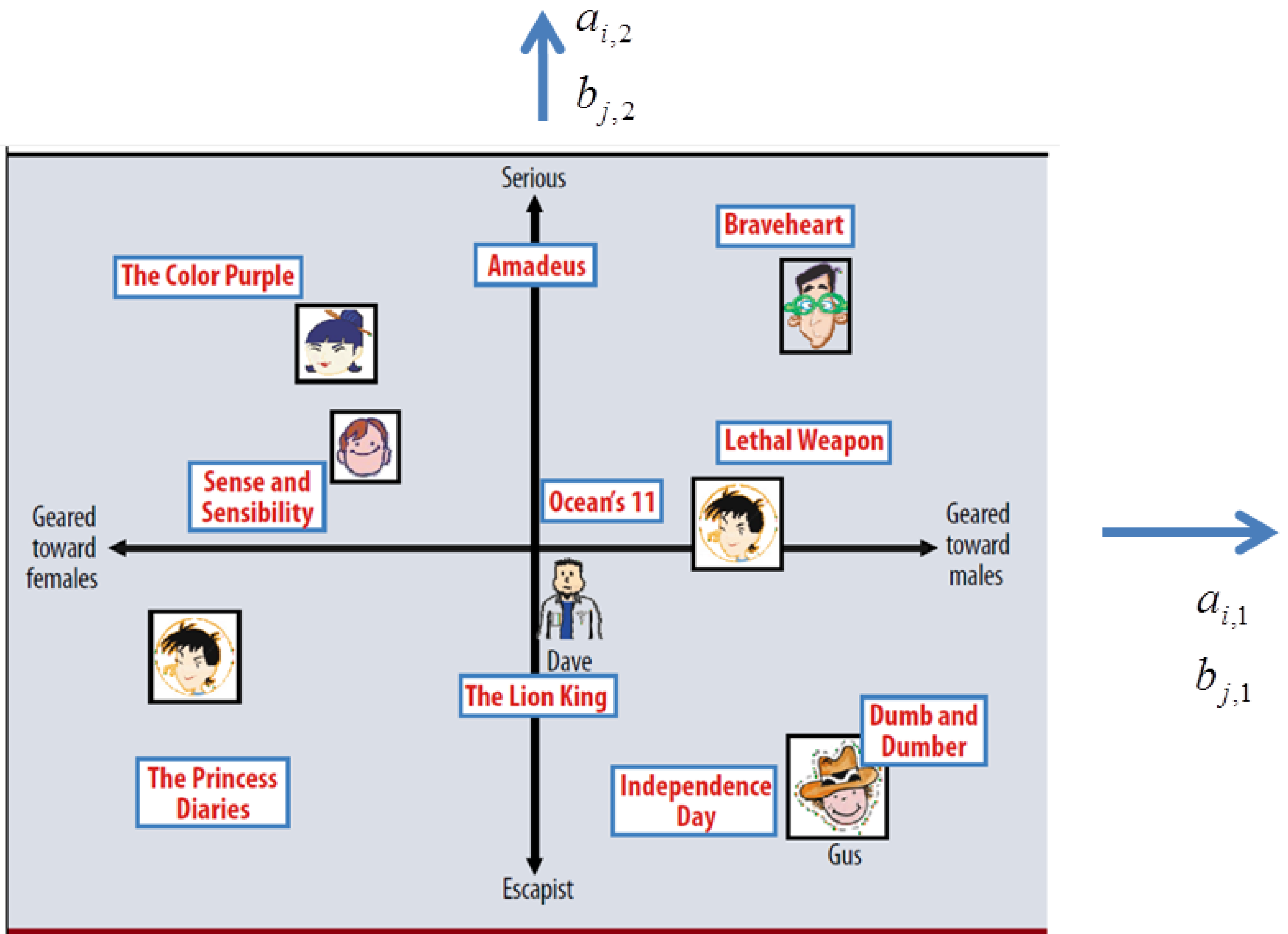


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

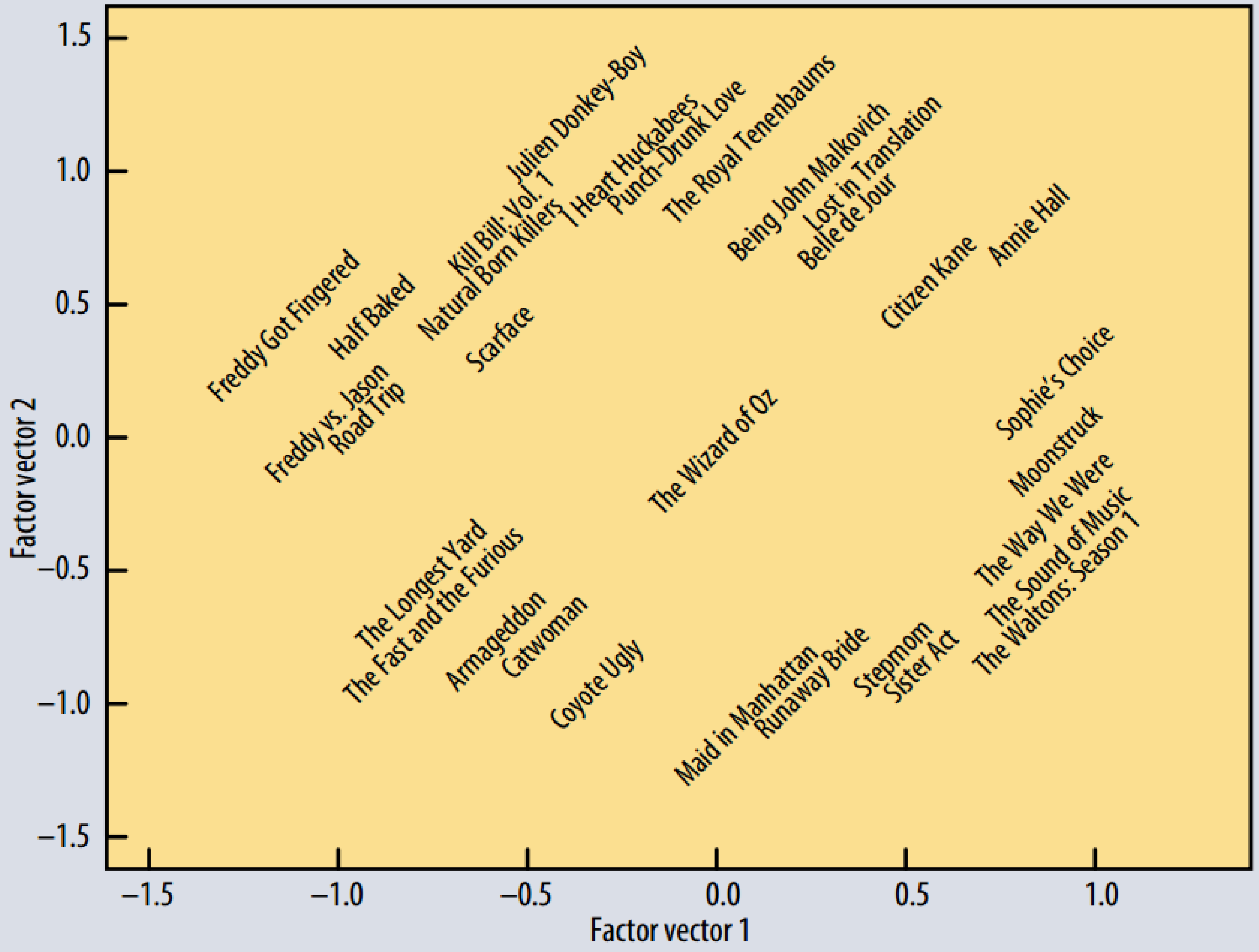


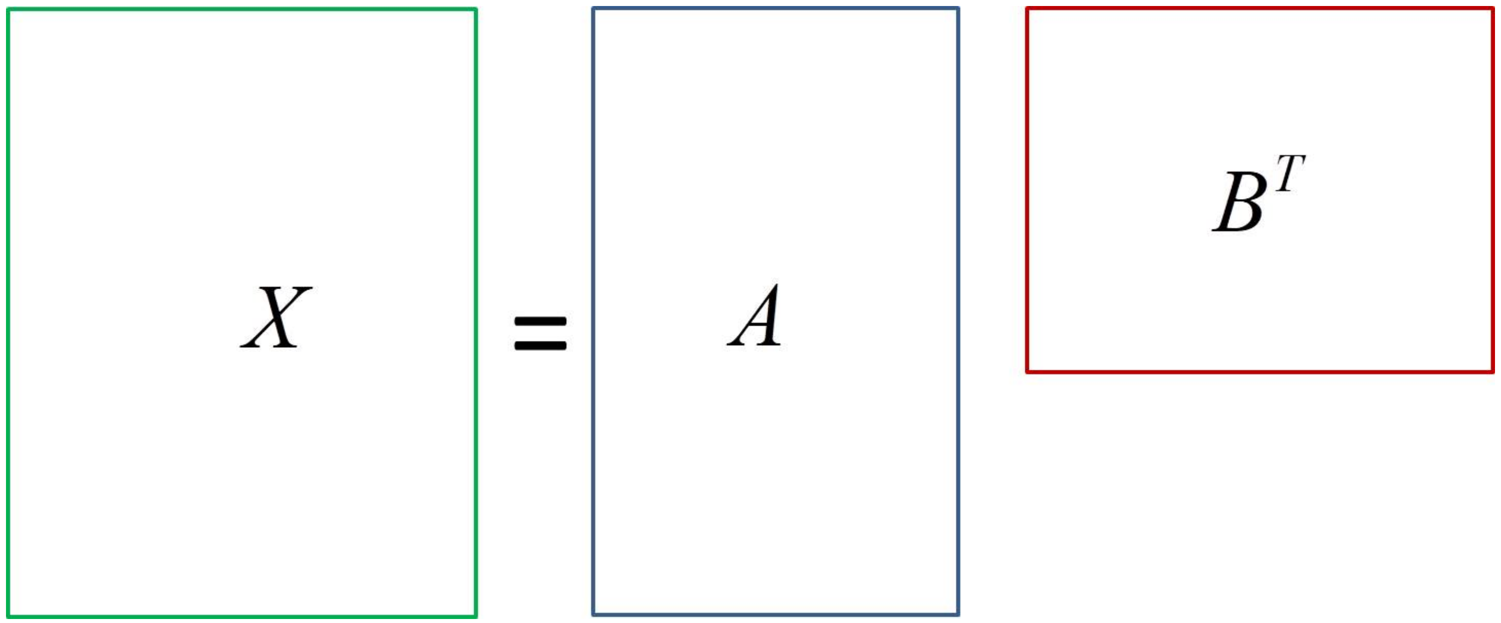
Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Factorization of the Design Matrix for classification

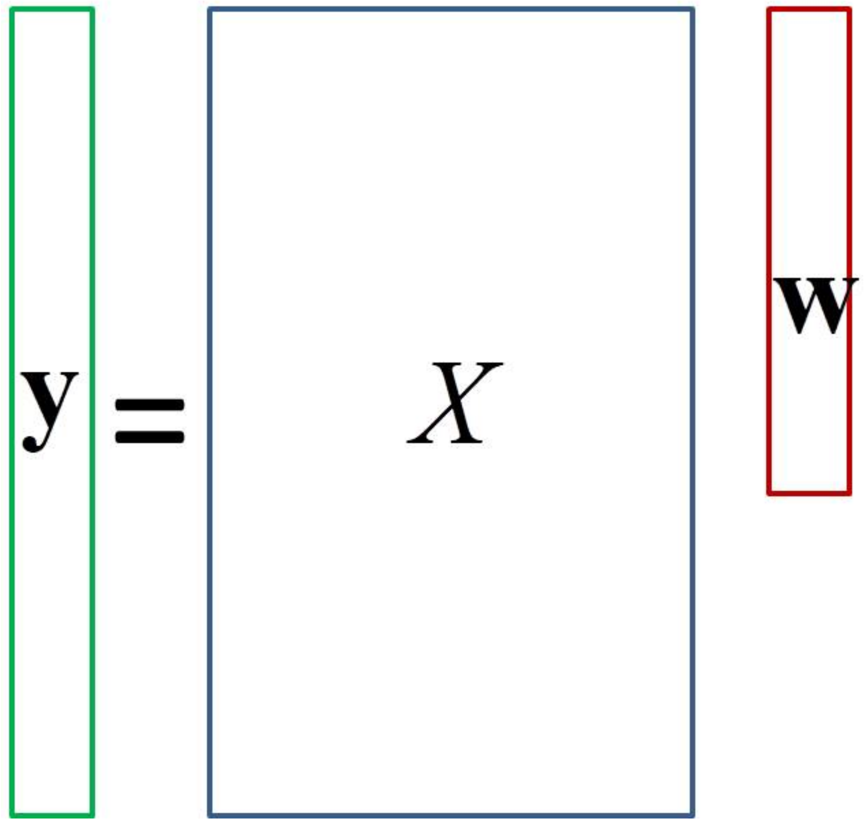
- So far we started with $Y \approx AB^T$, i.e., we factorized the output matrix and were interested in the latent factors of rows and columns to reconstruct missing values in Y
- In other regression/classification tasks it makes sense to factorize the design matrix X as

$$X \approx AB^T$$

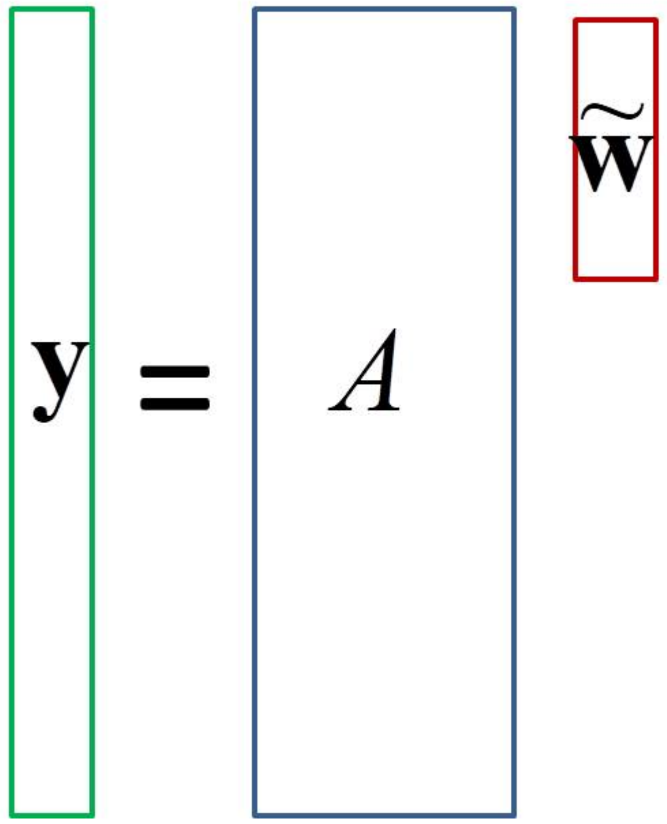
- This is a form of dimensional reduction, if $r < M$
- As we will see later, a classifier with A as design matrix can give better results than a classifier with X as design matrix. Example: X has many columns and is extremely sparse, A might have a small number of columns and is non-sparse


X
$=$
A
B^T

Predictions with original attributes



Predictions with latent factors



Matrix Factorization as an Autoencoder

- Compute latent factors A as weighted sum of inputs X

$$A = XV$$

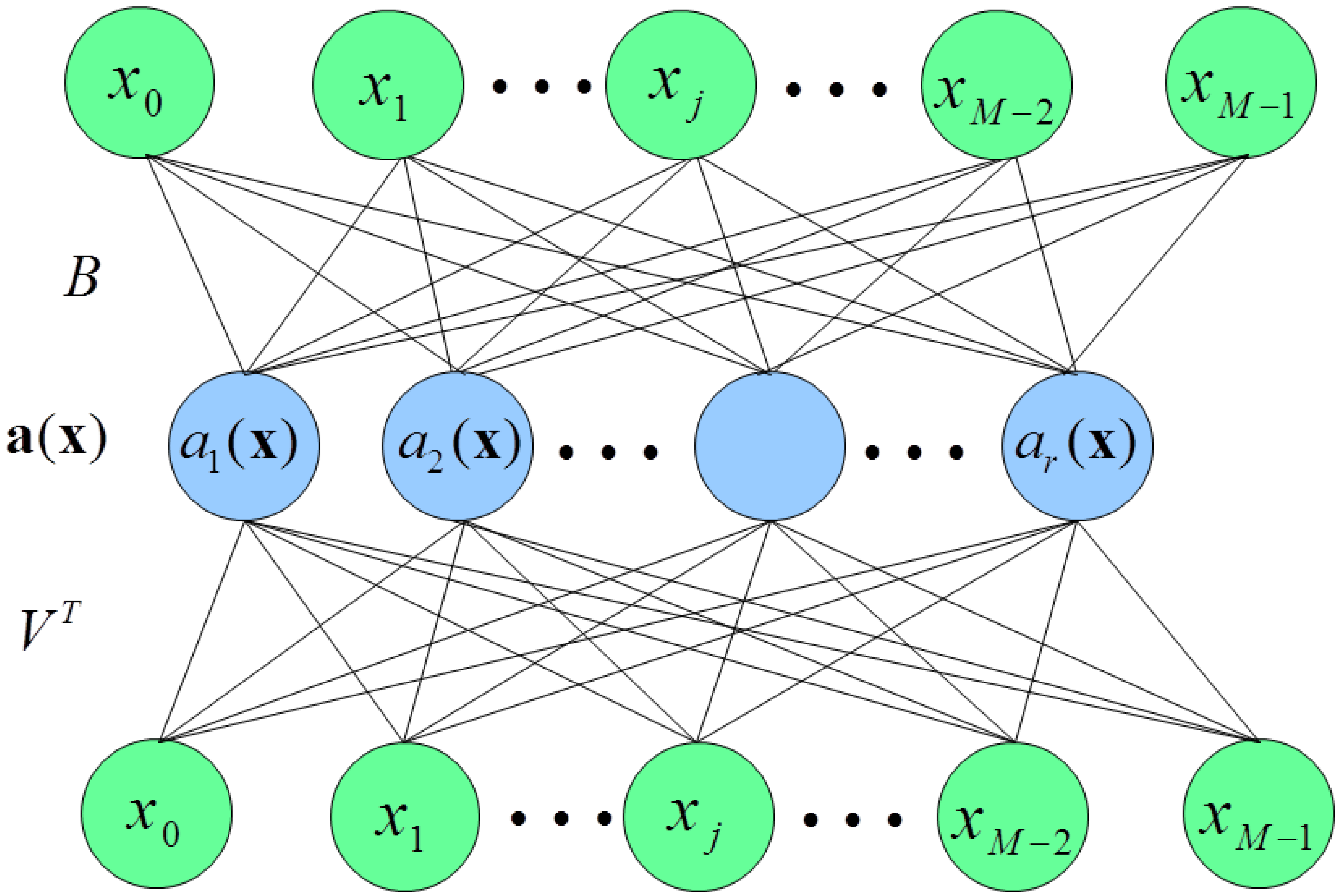
- Then reconstruct X from latent factors A

$$X \approx XVB^T$$

or

$$\mathbf{x}_i \approx BV^T\mathbf{x}_i$$

- Thus if X is complete, we can learn the factorization via an autoencoder


x_0
x_1
x_j
x_{M-2}
x_{M-1}
B
$\mathbf{a}(\mathbf{x})$
$a_1(\mathbf{x})$
$a_2(\mathbf{x})$
$a_r(\mathbf{x})$
V^T
x_0
x_1
x_j
x_{M-2}
x_{M-1}

PCA

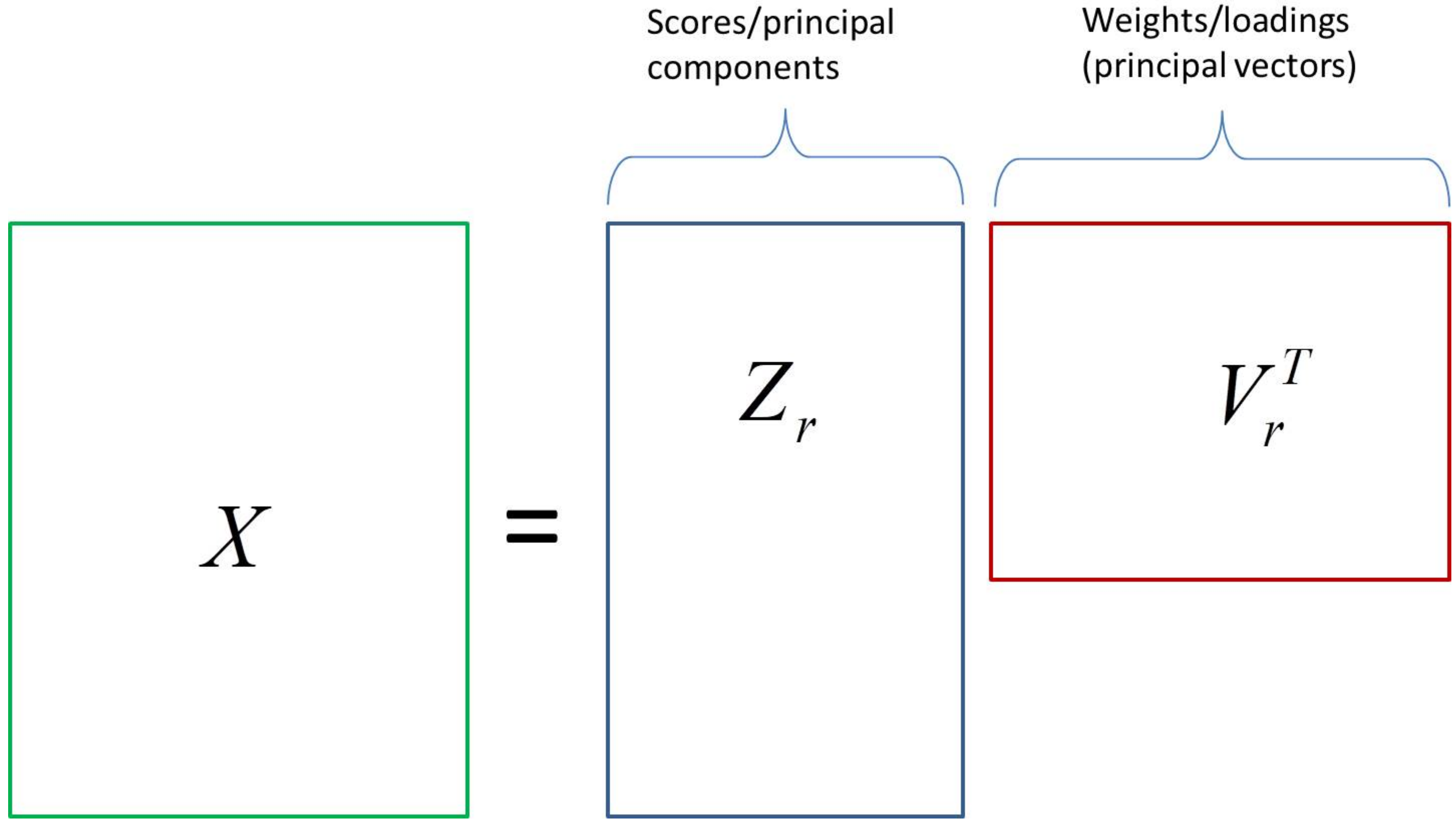
- The factorization approach as described is not unique and it has only been recently used in machine learning
- More traditional is the factorization via a principal component analysis (PCA)
- Transform correlated observations X into set of orthogonal latent variables that (iteratively) maximize the variance of the latent factors
- With $A \rightarrow Z$ and $B \rightarrow V$ we get

$$X \approx Z_r V_r^T$$

- The i-th row of A contains the r principal *principal components* of i (new name for the latent factors). With $r = min(M, N)$ the factorization is without error. With $r < min(M, N)$ this is an approximation
- The decomposition is unique and is optimal for any r with respect to the cost function

$$\sum_{i,j} \left(x_{i,j} - \sum_{k=1}^{r} z_{i,k} v_{j,k} \right)^2$$

- We will now derive the solution


Scores/principal components
Weights/loadings (principal vectors)
X
=
Z_r
V_r^T

Dimensionality Reduction

- We want to compress the M-dimensional $\mathbf{x}$ to an $r-$dimensional $\mathbf{z}$ using a linear transformation

- We want that $\mathbf{x}$ can be reconstructed from $\mathbf{z}$ as well as possible in the mean squared error sense for all data points $\mathbf{x}_i$

  $$\sum_i (\mathbf{x}_i - V_r \mathbf{z}_i)^T (\mathbf{x}_i - V_r \mathbf{z}_i)$$

  where V_r is an $M \times r$ matrix.

- We want the columns of V to be orthonormal

First Component

- Let's first look at $r = 1$ and we want to find the vector $\mathbf{v}$
- We require that $\|\mathbf{v}\| = 1$
- The reconstruction error for a particular $\mathbf{x}_i$ is given by

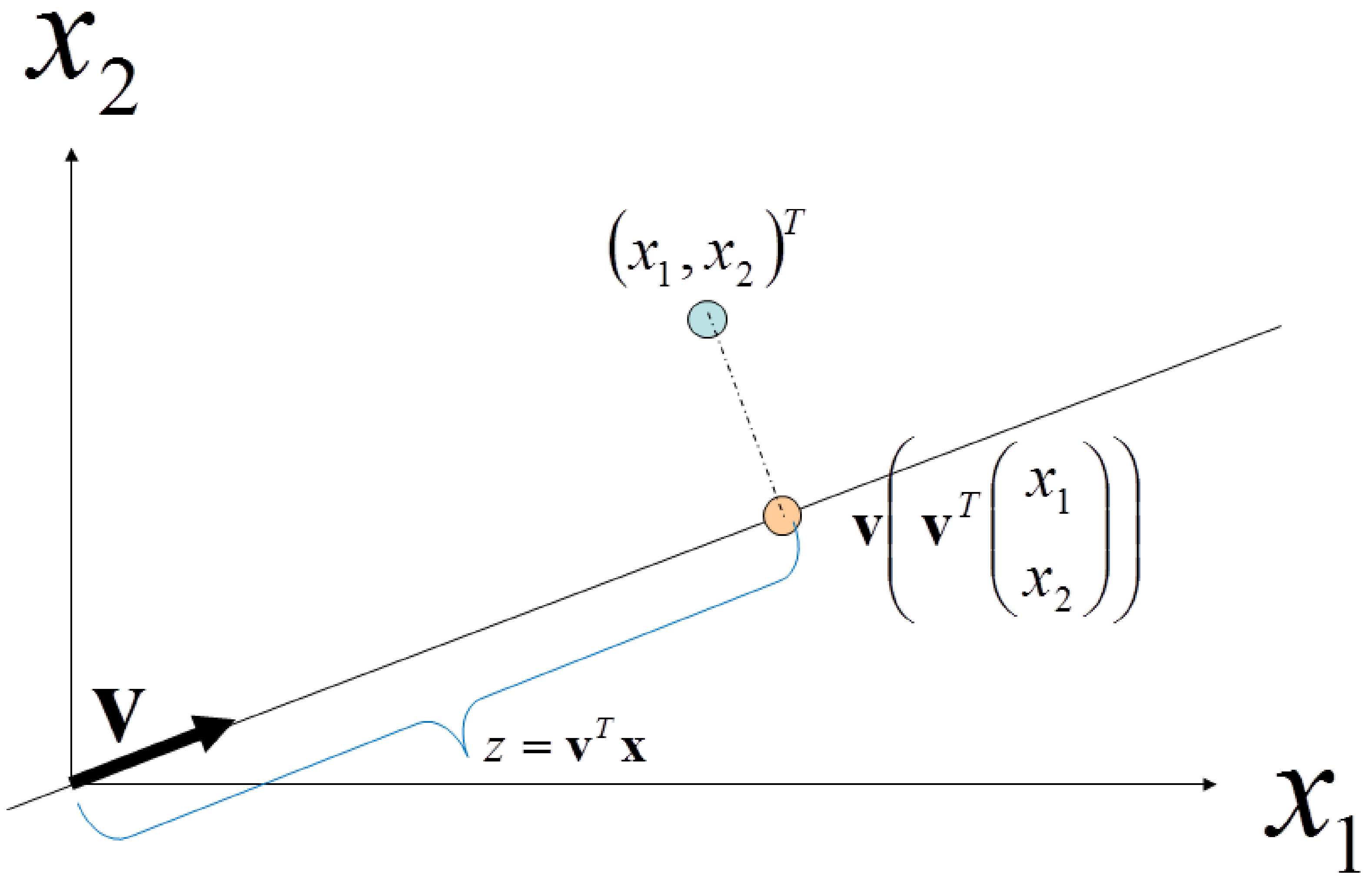
$$(\mathbf{x}_i - \hat{\mathbf{x}}_i)^T(\mathbf{x}_i - \hat{\mathbf{x}}_i) = (\mathbf{x}_i - \mathbf{v}z_i)^T(\mathbf{x}_i - \mathbf{v}z_i).$$

  The optimal z_i is then (see figure)

$$z_i = \mathbf{v}^T\mathbf{x}_i$$

  Thus we get

$$\hat{\mathbf{x}}_i = \mathbf{v}\mathbf{v}^T\mathbf{x}_i$$


x_2
$(x_1, x_2)^T$
$\mathbf{v}\left(\mathbf{v}^T\begin{pmatrix}x_1\\x_2\end{pmatrix}\right)$
$\mathbf{v}$
$z = \mathbf{v}^T\mathbf{x}$
x_1

Computing the First Principal Vector

- So what is $\mathbf{v}$? We are looking for a $\mathbf{v}$ that minimized the reconstruction error over all data points. We use the Lagrange parameter λ to guarantee length 1

$$L = \sum_{i=1}^{N} (\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i)^T(\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i) + \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

$$= \sum_{i=1}^{N} \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i + \mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i + \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

$$= \sum_{i=1}^{N} \mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i + \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

Computing the First Principal Vector

- The first term does not depend on $\mathbf{v}$. We take the derivative with respect to $\mathbf{v}$ and obtain for the second term

$$\frac{\partial}{\partial \mathbf{v}} \mathbf{x}_i^T \mathbf{v} \mathbf{v}^T \mathbf{x}_i$$

$$= \frac{\partial}{\partial \mathbf{v}} (\mathbf{v}^T \mathbf{x}_i)^T (\mathbf{v}^T \mathbf{x}_i) = 2 \left(\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T \mathbf{x}_i \right) (\mathbf{v}^T \mathbf{x}_i)$$

$$= 2\mathbf{x}_i (\mathbf{v}^T \mathbf{x}_i) = 2\mathbf{x}_i (\mathbf{x}_i^T \mathbf{v}) = 2(\mathbf{x}_i \mathbf{x}_i^T)\mathbf{v}$$

and for the last term

$$\lambda \frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T \mathbf{v} = 2\lambda \mathbf{v}$$

- We set the derivative to zero and get

$$\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \mathbf{v} = \lambda \mathbf{v}$$

or in matrix form

$$\Sigma \mathbf{v} = \lambda \mathbf{v}$$

where $\Sigma = X^T X$

- Recall that the Lagrangian is maximized with respect to λ
- Thus the first principal vector $\mathbf{v}$ is the first eigenvector of Σ (with the largest eigen-value)
- $z_i = \mathbf{v}^T \mathbf{x}_i$ is called the first principal component of $\mathbf{x}_i$

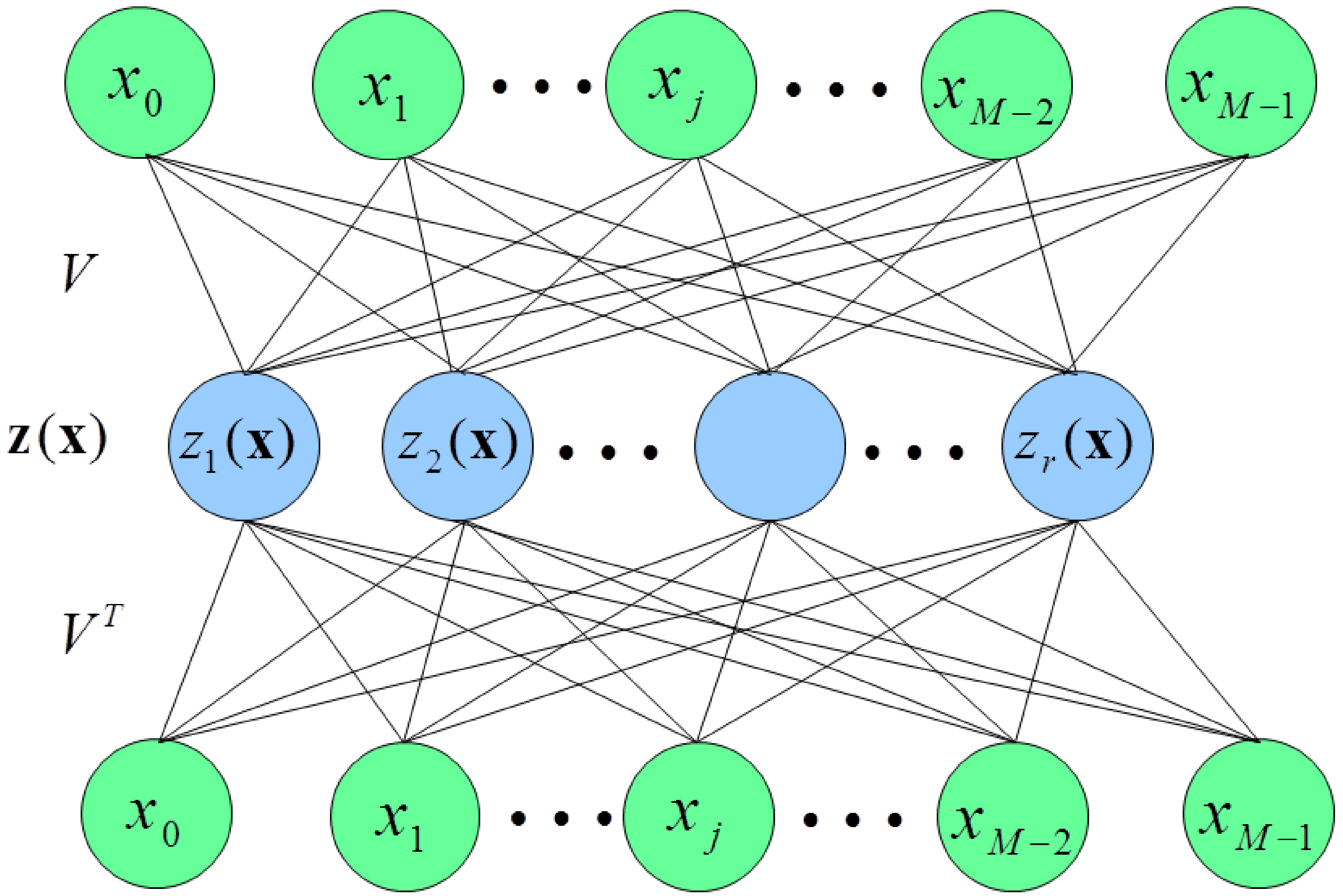
Computing all Principal Vectors

- The second principal vector is given by the second eigenvector of Σ and so on
- For a rank r approximation we get

$$\mathbf{z}_i = V_r^T \mathbf{x}_i$$

- Here, the columns of V_r are all orthonormal and correspond to the r eigenvectors of Σ with the largest eigenvalues
- The optimal reconstruction is

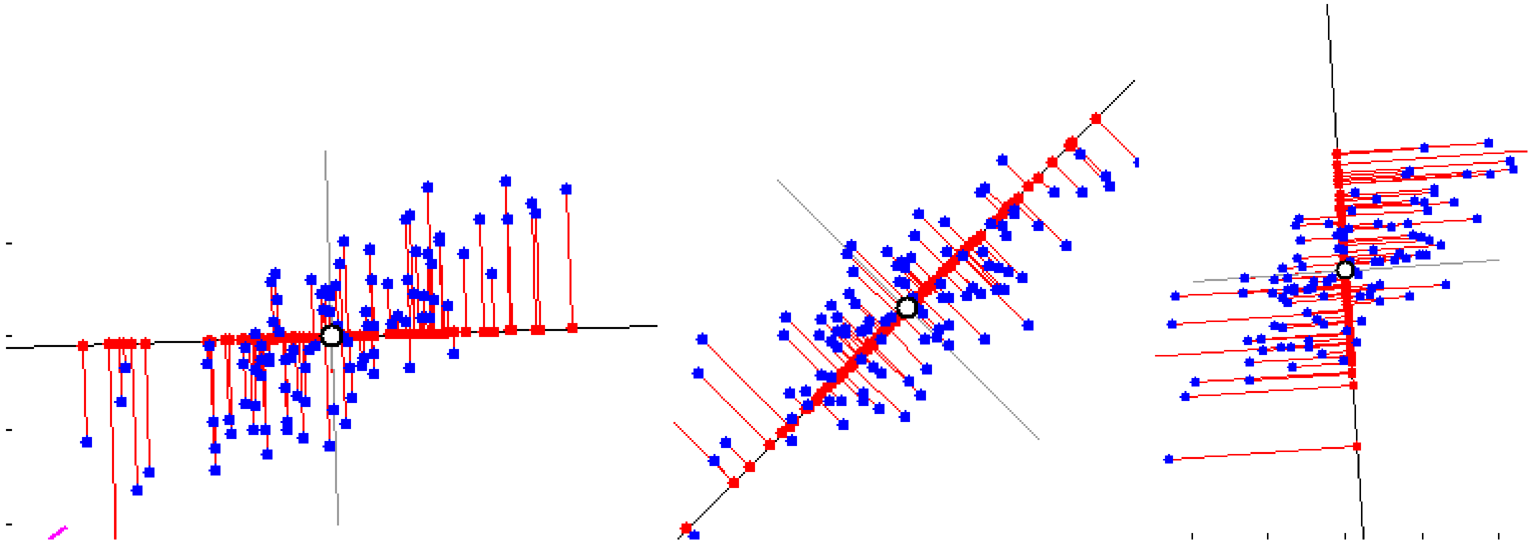
$$\hat{\mathbf{x}}_i = V_r \mathbf{z}_i$$



x_0
x_1
x_j
x_{M-2}
x_{M-1}
V
$\mathbf{z}(\mathbf{x})$
$z_1(\mathbf{x})$
$z_2(\mathbf{x})$
$z_r(\mathbf{x})$
V^T
x_0
x_1
x_j
x_{M-2}
x_{M-1}

Reconstruction error and Variance

- Minimizing the reconstruction error (as in our derivation) turns out to be equivalent to maximizing the variance
- First PC has largest possible variance, each succeeding PC has the highest variance possible under the constraint that it is orthogonal to all preceding components.



Variance explained

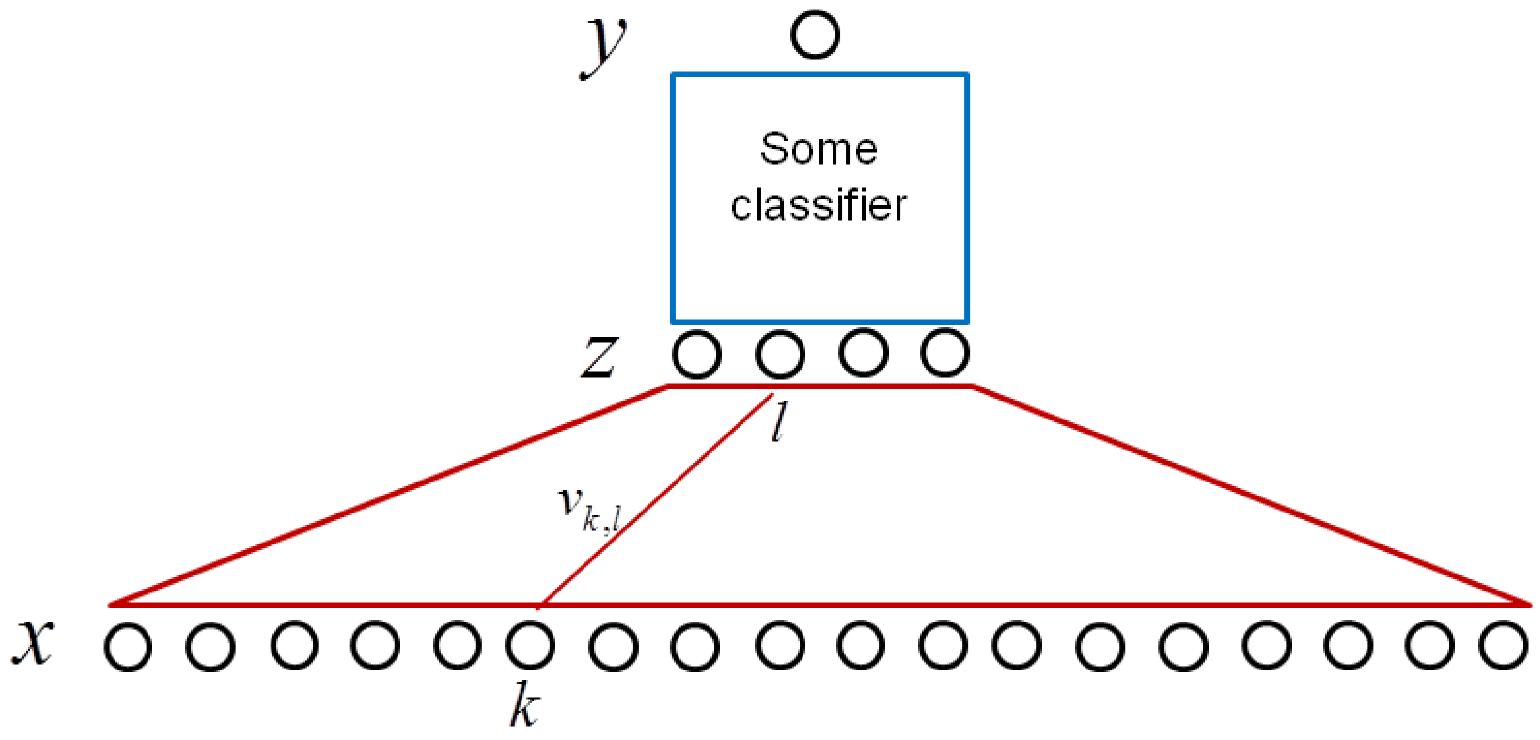
- Σ is (proportional to) the sample covariance matrix
- PCs are eigenvectors of Σ
- We diagonalize Σ via eigenvalue decomposition, with eigenvalues being the values on the diagonal
- Sum of diagonal remains the same as for sample covariance matrix
- Relative magnitude of eigenvalue corresponds to relative variance explained by specific PC

PCA Applications

Classification and Regression

- First perform an PCA of X and then use as input to the classifier $\mathbf{z}_i$ instead of $\mathbf{x}_i$, where

$$\mathbf{z}_i = V_r^T \mathbf{x}_i$$



y
Some classifier
z
l
$v_{k,l}$
x
k

Similarity and Novelty

- A distance measure (Euclidian distance) based on the principal components is often more meaningful than a distance measure calculated in the original space
- Novelty detection / outlier detection: We calculate the reconstruction of a new vector $\mathbf{x}$ and calculate

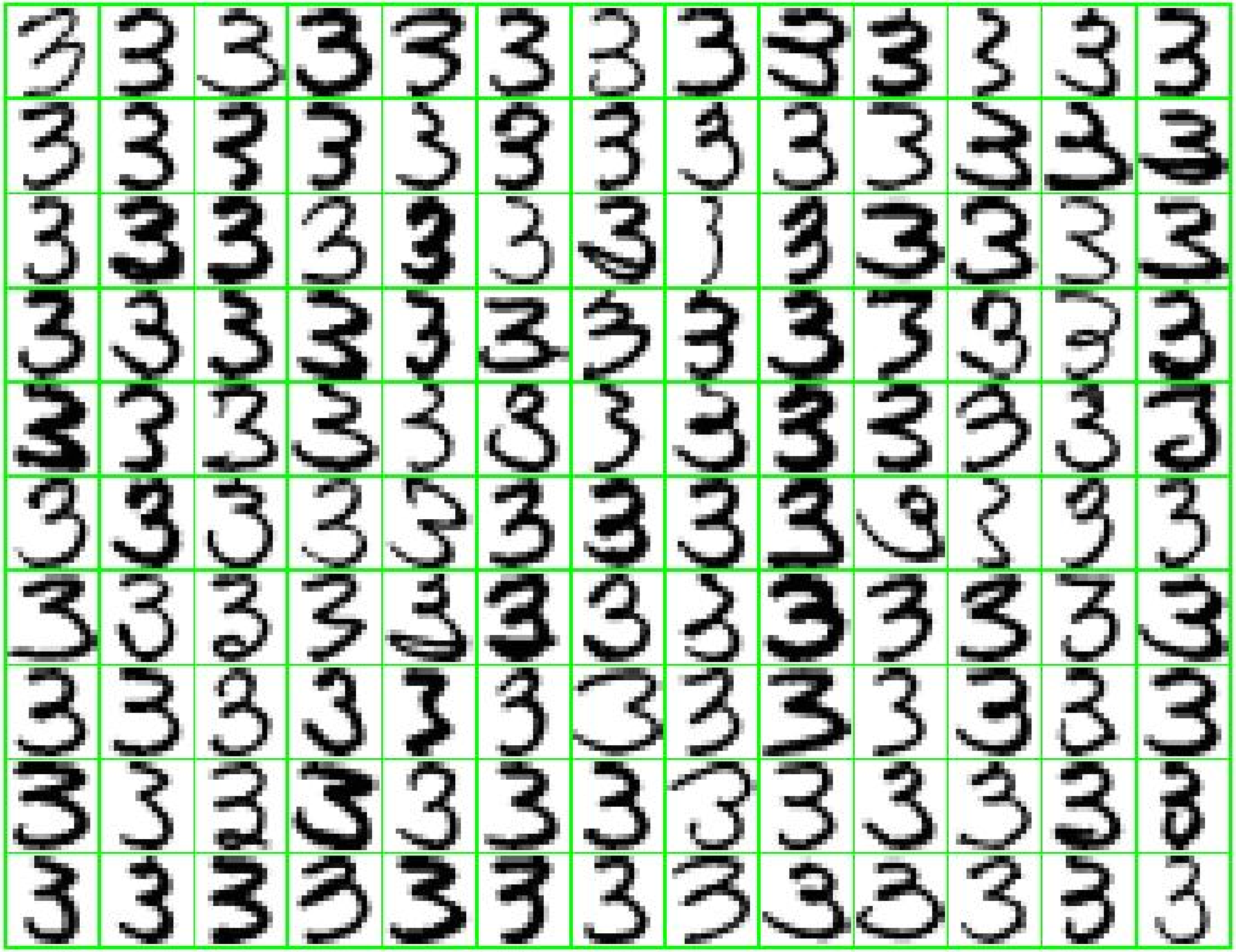
$$\|\mathbf{x} - V_r V_r^T \mathbf{x}\| = \|V_{-r}^T \mathbf{x}\|$$

  If this distance is large, then the new input is unusual, i.e. might be an outlier
- Here V_{-r} contains the $M - r$ eigenvectors $\mathbf{v}_{r+1}, ..., \mathbf{v}_M$ of Σ

PCA Example: Handwritten Digits

Data Set

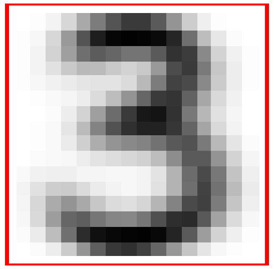
- 130 handwritten digits “ 3 ” (in total: 658): significant difference in style
- The images have 16×16 grey valued pixels. Each input vector $\mathbf{x}$ consists of 256 grey values of the pixels: applying a linear classifier to the original pixels gives bad results



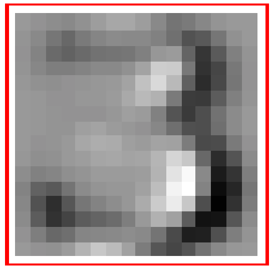
Visualisation

- We see the first two principal vectors $\mathbf{v}_1$, $\mathbf{v}_2$
- $\mathbf{v}_1$ prolongs the lower portion of the "3"
- $\mathbf{v}_2$ modulates thickness

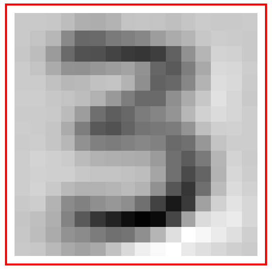
$$\hat{\mathbf{x}}_i =$$



$$+ \; z_{i,1} \cdot$$



$$+ \; z_{i,2} \cdot$$



$.$

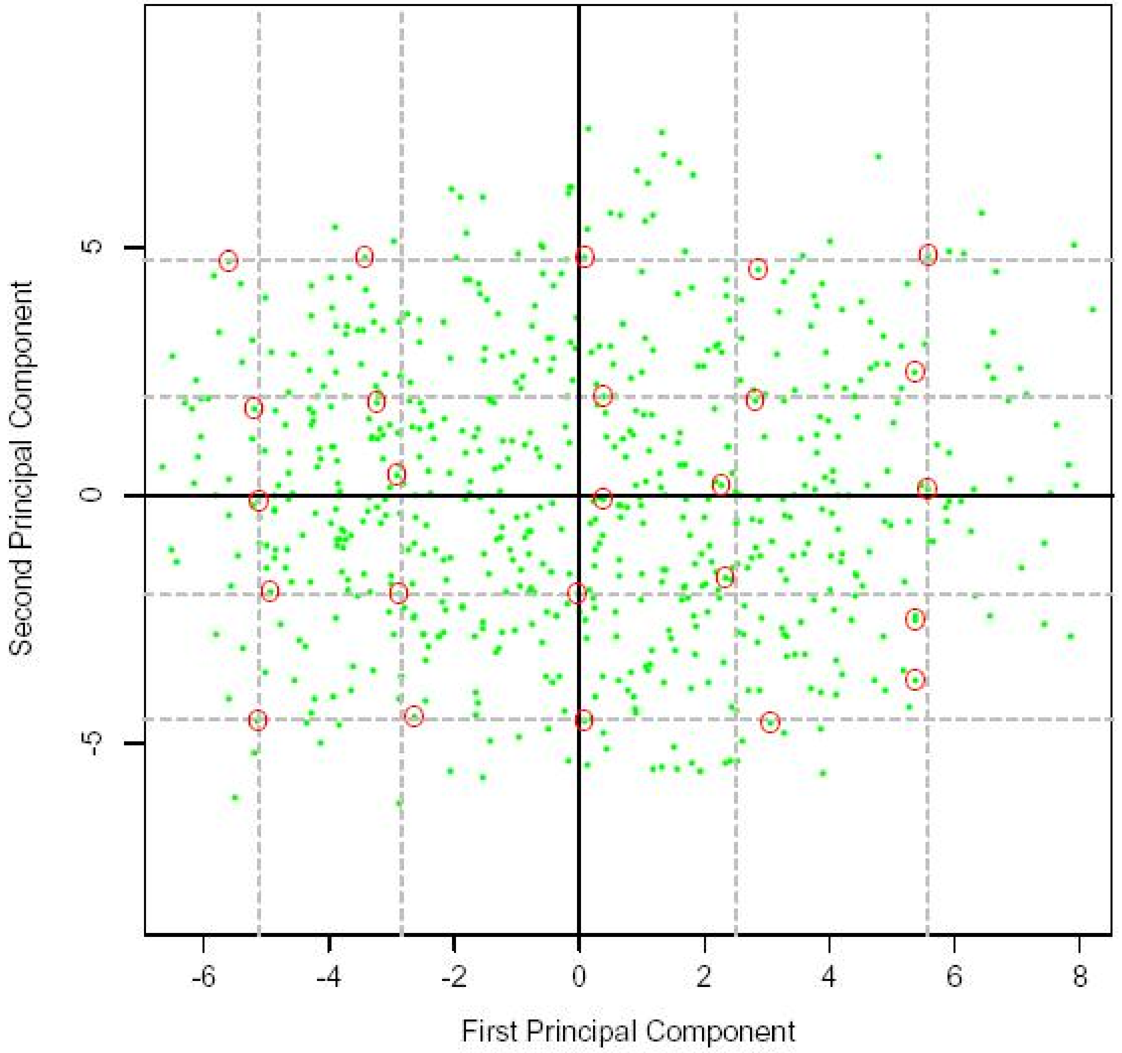
$\mathbf{m}$ $\mathbf{v}_1$ $\mathbf{v}_2$

Visualisation: Reconstruction

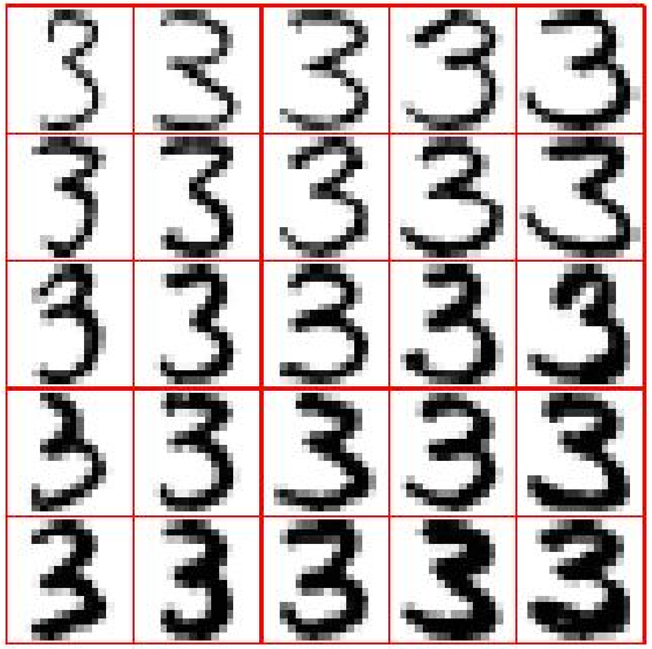
- For different values of the principal components z_1 and z_2 the reconstructed image is shown

$$\hat{\mathbf{x}} = \mathbf{m} + z_1\mathbf{v}_1 + z_2\mathbf{v}_2$$

- $\mathbf{m}$ is a mean vector that was subtracted before the PCA was performed and is now added again. $\mathbf{m}$ represents 256 mean pixel values averaged over all samples



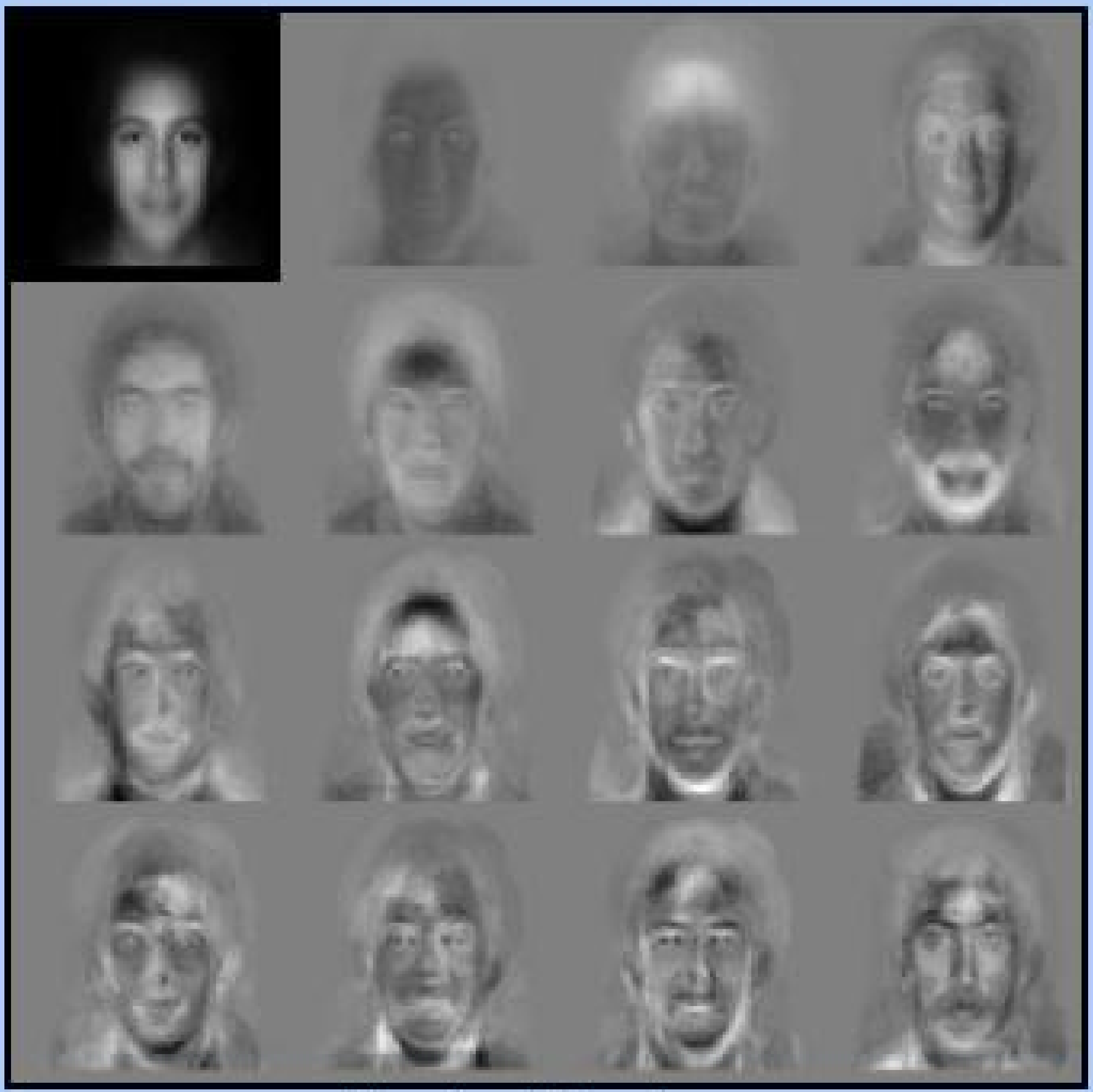
First Principal Component
Second Principal Component
-6
-4
-2
0
2
4
6
8
-5
0
5



Eigenfaces: similarity search of images

Data Set

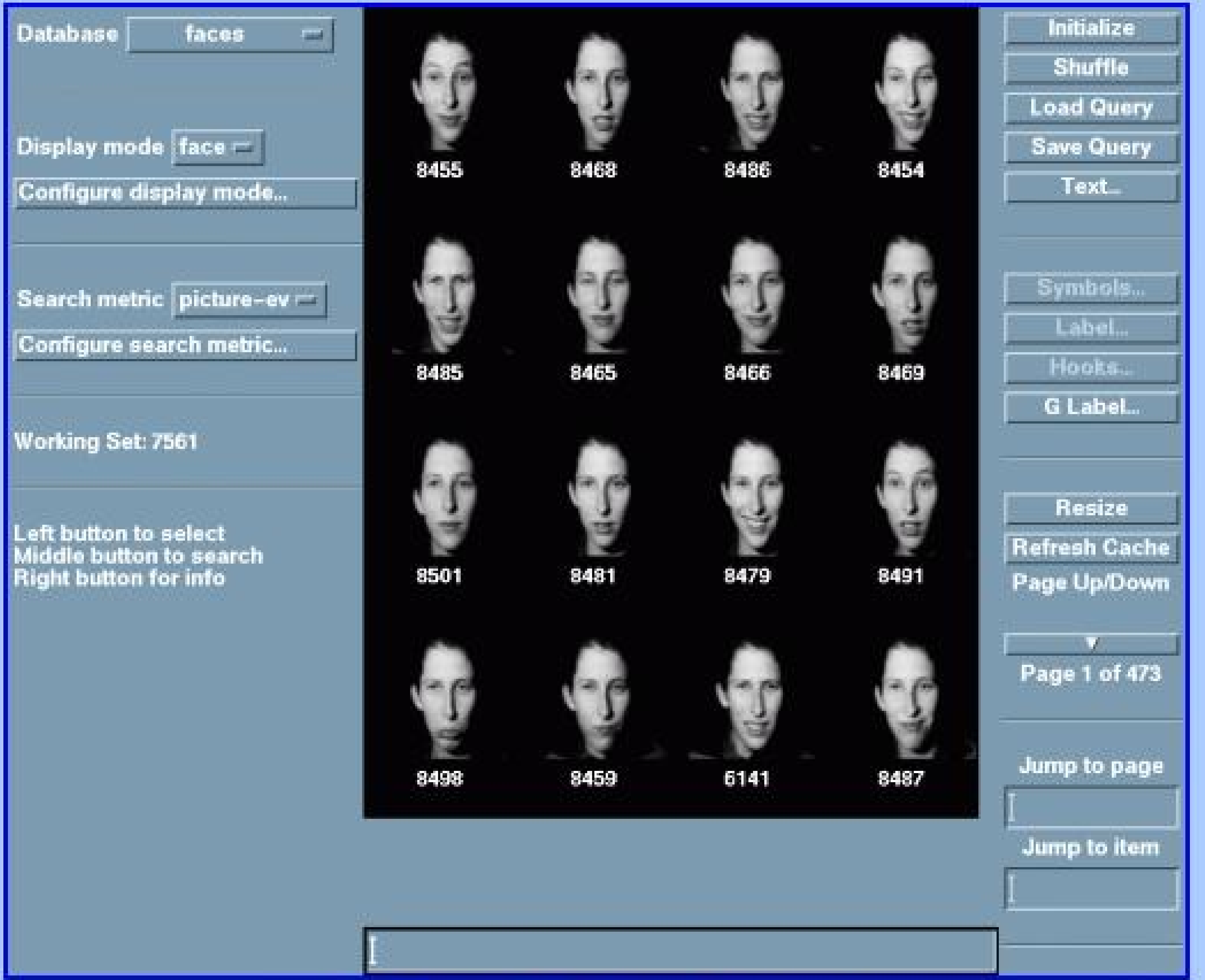
- PCA for face recognition
- http://vismod.media.mit.edu/vismod/demos/facerec/basic.html
- 7562 images from 3000 persons
- $\mathbf{x}_i$ contains the pixel values of the i-th image. Obviously it does not make sense to build a classifier directly on the $256 \times 256 = 65536$ pixel values
- PCs (Eigenfaces) were calculated based on 128 images (training set)
- For recognition on test images, the first $r = 20$ principal components are used
- Almost each person had at least 2 images; many persons had images with varying facial expression, different hair style, different beards, ...


Standard Eigenfaces

Similarity Search based on Principal Components

- The upper left image is the test image. Based on the Euclidian distance in PCA-space the other 15 images were classified as nearest neighbors. All 15 images came form the correct person.
- Thus, distance is evaluated following

$$||\mathbf{z} - \mathbf{z}_i||$$



MIT Media Lab Database Photobook

Recognition Rate

- 200 pictures were selected randomly from the test set. In 96% of all cases the nearest neighbor was the correct person

PCA with Centered Data

- Sometimes the mean is subtracted first

$$\tilde{x}_{i,j} = x_{i,j} - m_j$$

  where

$$m_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$$

- $\tilde{X}$ now contains the centered data
- Centering is recommended when data are approximately normally distributed

PCA with Centered Data (cont'd)

- Let

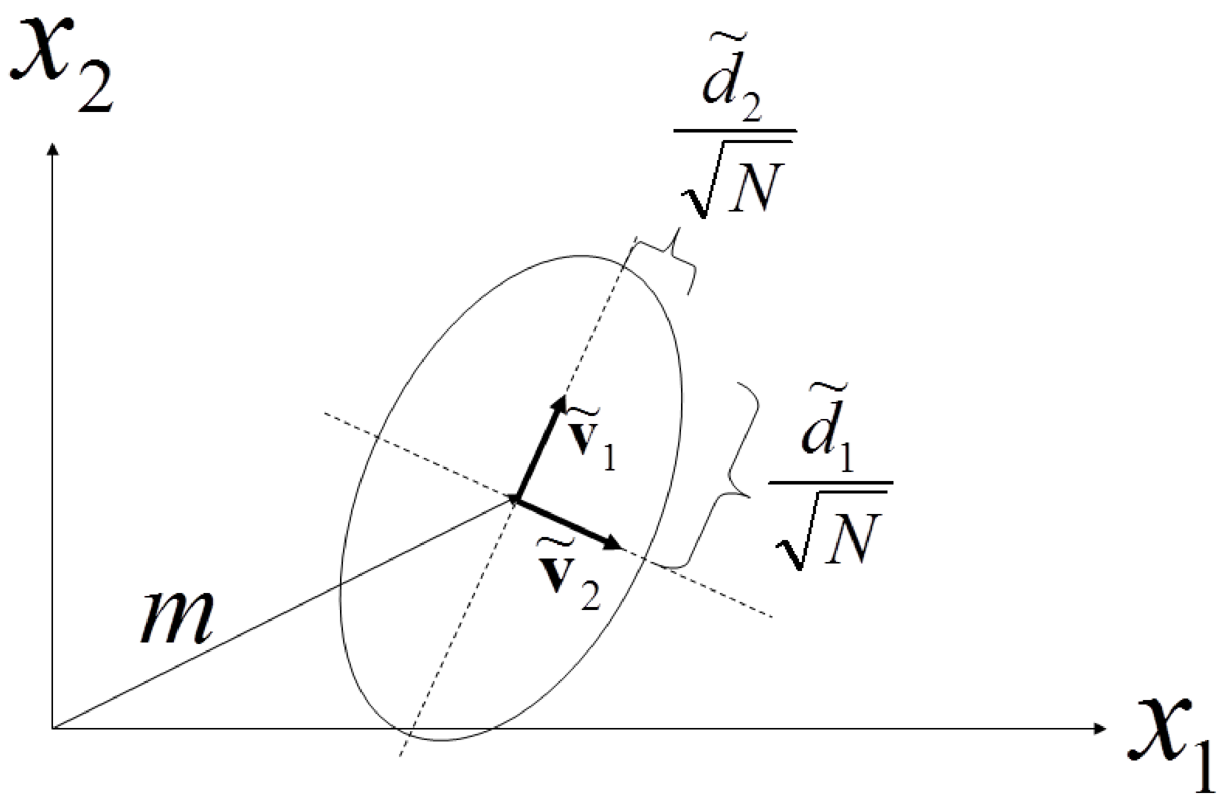
$$\hat{\tilde{X}} = \tilde{V}_r \tilde{V}_r^T \tilde{X}$$

then

$$\boxed{\hat{\mathbf{x}}_i = \mathbf{m} + \sum_{l=1}^{r} \tilde{\mathbf{v}}_l \tilde{z}_{i,l}}$$

with $m = (m_1, \ldots, m_M)^T$

$$\tilde{z}_{i,l} = \tilde{\mathbf{v}}_l^T \tilde{\mathbf{x}}_i$$


x_2
$\frac{\tilde{d}_2}{\sqrt{N}}$
$\tilde{\mathbf{v}}_1$
$\frac{\tilde{d}_1}{\sqrt{N}}$
$\tilde{\mathbf{v}}_2$
m
x_1

PCA and Singular Value Decomposition

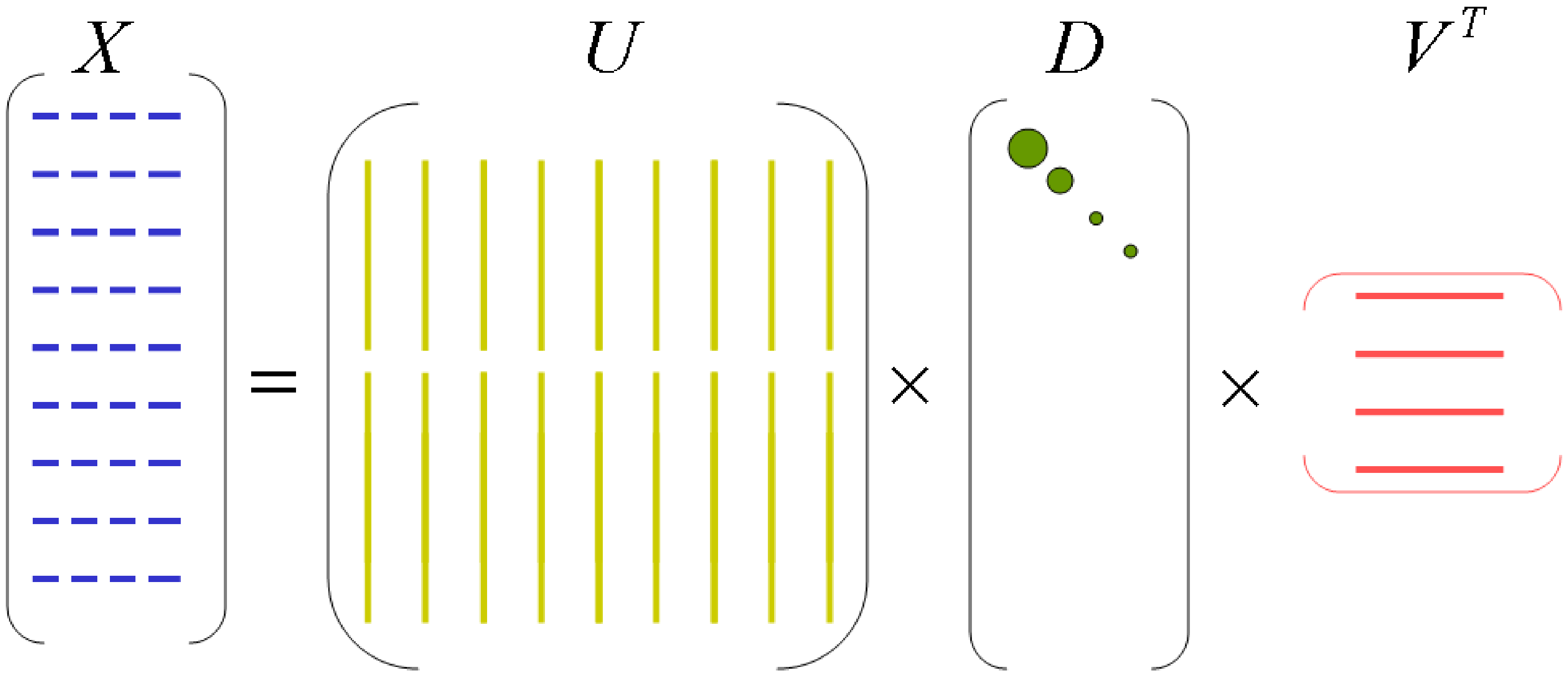
Singular Value Decomposition (SVD)

- Any $N \times M$ matrix X can be factored as

$$X = UDV^T$$

  where U and V are both **orthonormal** matrices. U is an $N \times N$ Matrix and V is an $M \times M$ Matrix.

- D is an $N \times M$ **diagonal matrix** with diagonal entries (singular values) $d_i \geq 0, i = 1, \ldots, \tilde{r}$, with $\tilde{r} = \min(M, N)$
- The $\mathbf{u}_j$ (columns of U) are the left singular vectors
- The $\mathbf{v}_j$ are the right singular vectors
- The d_j are the singular values


X
$=$
U
$\times$
D
$\times$
V^T

Covariance Matrix and Kernel Matrix

- We get for the empirical covariance matrix

$$\Sigma = \frac{1}{N} X^T X = \frac{1}{N} V D^T U^T U D V^T = \frac{1}{N} V D^T D V^T = \frac{1}{N} V D_V V^T$$

- And for the empirical kernel matrix

$$K = \frac{1}{M} X X^T = \frac{1}{M} U D V^T V D^T U^T = \frac{1}{M} U D D^T U^T = \frac{1}{M} U D_U U^T$$

- With

$$\Sigma V = \frac{1}{N} V D_V \quad K U = \frac{1}{M} U D_K$$

one sees that the columns of V are the eigenvectors of Σ and the columns of U are the eigenvectors of K: The eigenvalues are the diagonal entries of D_V, respectively D_U.

- The square root of the eigenvalues of $X^T X$ are the singular values of X
- The columns of V are both the principal vectors and the eigenvectors

More Expressions

- The SVD is

$$X = UDV^T$$

from which we get

$$X = UU^T X$$

$$X = XVV^T$$

Reduced Rank

- In the SVD, the d_i are ordered: $d_1 \geq d_2 \geq d_3 ... \geq d_{\tilde{r}}$. In many cases one can neglect $d_i, i > r$ and one obtains a rank-r Approximation. Let D_r be a diagonal matrix with the corresponding entries. Then we get the approximation

$$\hat{X} = U_r D_r V_r^T$$

$$\hat{X} = U_r U_r^T X$$

$$\hat{X} = X V_r V_r^T$$

where U_r contains the first r columns of U. Correspondingly, V_r

Best Approximation

- The approximation above is the best rank-r approximation with respect to the squared error (Frobenius Norm). The approximation error is

$$\sum_{i=1}^{N} \sum_{j=1}^{M} (x_{j,i} - \widehat{x}_{j,i})^2 = \sum_{j=r+1}^{\tilde{r}} d_j^2$$

Factors for Rows and Columns

- Recall that in the netflix example on matrix factorization with $X \approx AB^T$, the rows of A contained the factors for the users and the rows of B contained the factors for the movies
- In the PCA, factors for entities associated with the columns are the rows of
$$T_r = X^T U_r$$
- With this definition,
$$\hat{X} = Z_r D_r^{-1} T_r^T$$
since
$$Z_r D_r^{-1} T_r^T = X V_r D_r^{-1} U_r^T X = U_r D_r V_r^T V_r D_r^{-1} U_r^T U_r D_r V_r^T = U_r D_r V_r^T$$

PCA in practice

- Using the SVD on X to perform PCA makes much better sense numerically than forming the covariance matrix X^TX
- Formation of X^TX can cause loss of precision
- ..is mo

Application: Similarities Between Documents

Feature Vectors for Documents

- Given a collection of N documents and M keywords
- X is the *term-frequency* (tf) matrix; $x_{i,j}$ indicates how often word j occurred in document i.
- Some classifiers use this representation as inputs
- On the other hand, two documents might discuss similar topics (are "semantically similar") without using the same key words
- By doing a PCA we can find document representations as rows of $Z_r = XV$ and term representations as rows of $T = X^T U$
- This is also known as *Latent Semantic Analysis* (LSA)

Simple Example

- In total 9 sentences (documents):
  - 5 documents on human-computer interaction) (c1 - c5)
  - 4 texts on mathematical graph theory (m1 - m4)
- The 12 key words are in italic letters

Example of text data: Titles of Some Technical Memos

c1:	*Human* machine *interface* for ABC *computer* applications
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 measurement
m1:	The generation of random, binary, ordered *trees*
m2:	The intersection *graph* of paths in *trees*
m3:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4:	*Graph minors*: A *survey*

tf-Matrix and Word Correlations

- The tf-Matrix X
- Based on the original data, the Pearson correlation between *human* and *user* is negative, although one would assume a large semantic correlation

X^T

	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

$\underline{r}$ (human.user) = -.38 Pearson correlation between the words *human* and *user*

$\underline{r}$ (human.minors) = -.29 Pearson correlation between the words *human* and *minor*

Singular Value Decomposition

- Decomposition $X = UDV^T$

$$V =$$

$$\begin{array}{rrrrrrrrr}
0.22 & -0.11 & 0.29 & -0.41 & -0.11 & -0.34 & 0.52 & -0.06 & -0.41 \\
0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11 \\
0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49 \\
0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01 \\
0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27 \\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\
0.30 & -0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17 \\
0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58 \\
0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23 \\
0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.23 \\
0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18
\end{array}$$

$$D =$$

$$\begin{array}{ccccccccc}
3.34 & & & & & & & & \\
 & 2.54 & & & & & & & \\
 & & 2.35 & & & & & & \\
 & & & 1.64 & & & & & \\
 & & & & 1.50 & & & & \\
 & & & & & 1.31 & & & \\
 & & & & & & 0.85 & & \\
 & & & & & & & 0.56 & \\
 & & & & & & & & 0.36
\end{array}$$

$$X = UDV^T$$

$$U =$$

$$\begin{array}{rrrrrrrrr}
0.20 & 0.61 & 0.46 & 0.54 & 0.28 & 0.00 & 0.01 & 0.02 & 0.08 \\
-0.06 & 0.17 & -0.13 & -0.23 & 0.11 & 0.19 & 0.44 & 0.62 & 0.53 \\
0.11 & -0.50 & 0.21 & 0.57 & -0.51 & 0.10 & 0.19 & 0.25 & 0.08 \\
-0.95 & -0.03 & 0.04 & 0.27 & 0.15 & 0.02 & 0.02 & 0.01 & -0.03 \\
0.05 & -0.21 & 0.38 & -0.21 & 0.33 & 0.39 & 0.35 & 0.15 & -0.60 \\
-0.08 & -0.26 & 0.72 & -0.37 & 0.03 & -0.30 & -0.21 & 0.00 & 0.36 \\
0.18 & -0.43 & -0.24 & 0.26 & 0.67 & -0.34 & -0.15 & 0.25 & 0.04 \\
-0.01 & 0.05 & 0.01 & -0.02 & -0.06 & 0.45 & -0.76 & 0.45 & -0.07 \\
-0.06 & 0.24 & 0.02 & -0.08 & -0.26 & -0.62 & 0.02 & 0.52 & -0.45
\end{array}$$

Approximation with $r = 2$ and Word Correlations

- Reconstruction $\hat{X}$ with $r = 2$
- Shown is $\hat{X}^T$
- Based on $\hat{X}$ the correlation between *human* and *user* is almost one! The similarity between *human* and *minors* is strongly negative (as it should be)
- In document $m4$: "*Graph minors: a survey*" the word *survey* which is in the original document gets a smaller value than the term *trees*, which was not in the document originally

$\hat{X}^T$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-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	0.70	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	-0.20	-0.11
survey	0.10	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	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

r (human.user) = .94 Pearson correlation between the words *human* and *user*

r (human.minors) = -.83 Pearson correlation between the words *human* and *minor*

Document Correlations in the Original and the Reconstructed Data

- Top: document correlation in the original data X: The average correlation between documents in the c-class is almost zero
- Bottom: in $\widehat{X}$ there is a strong correlation between documents in the same class and strong negative correlation across document classes

Correlations between titles in raw data:

	c1	c2	c3	c4	c5	m1	m2	m3
c2	-0.19							
c3	0.00	0.00						
c4	0.00	0.00	0.47					
c5	-0.33	0.58	0.00	-0.31				
m1	-0.17	-0.30	-0.21	-0.16	-0.17			
m2	-0.26	-0.45	-0.32	-0.24	-0.26	0.67		
m3	-0.33	-0.58	-0.41	-0.31	-0.33	0.52	0.77	
m4	-0.33	-0.19	-0.41	-0.31	-0.33	-0.17	0.26	0.56

0.02	
-0.30	0.44

Average Pearson correlation in the three document blocks in the raw data

Correlations in two dimensional space:

c2	0.91							
c3	1.00	0.91						
c4	1.00	0.88	1.00					
c5	0.85	0.99	0.85	0.81				
m1	-0.85	-0.56	-0.85	-0.88	-0.45			
m2	-0.85	-0.56	-0.85	-0.88	-0.44	1.00		
m3	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1.00	
m4	-0.81	-0.50	-0.81	-0.84	-0.37	1.00	1.00	1.00

0.92	
-0.72	1.00

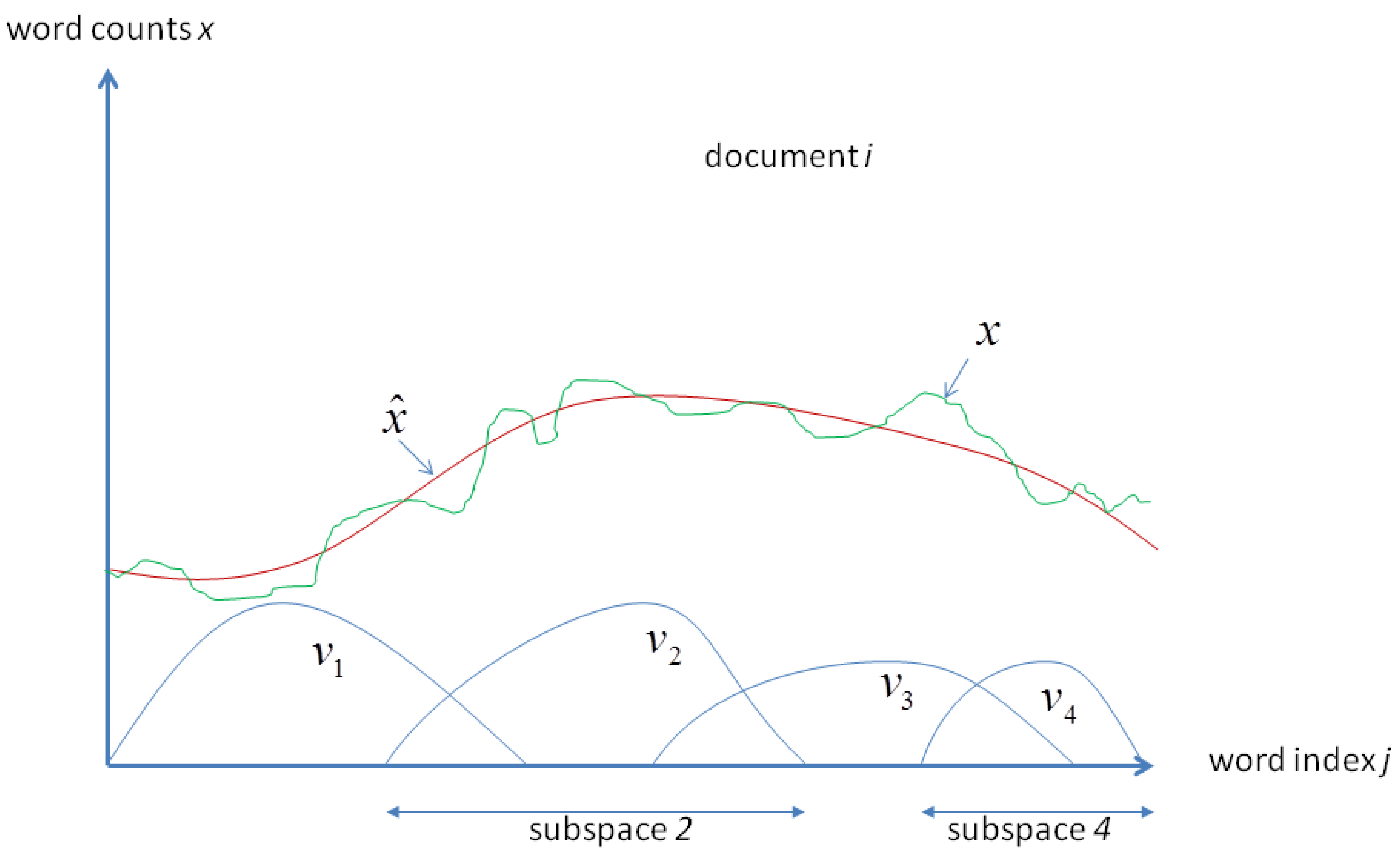
Average Pearson correlation in the three document blocks in the reconstructed data

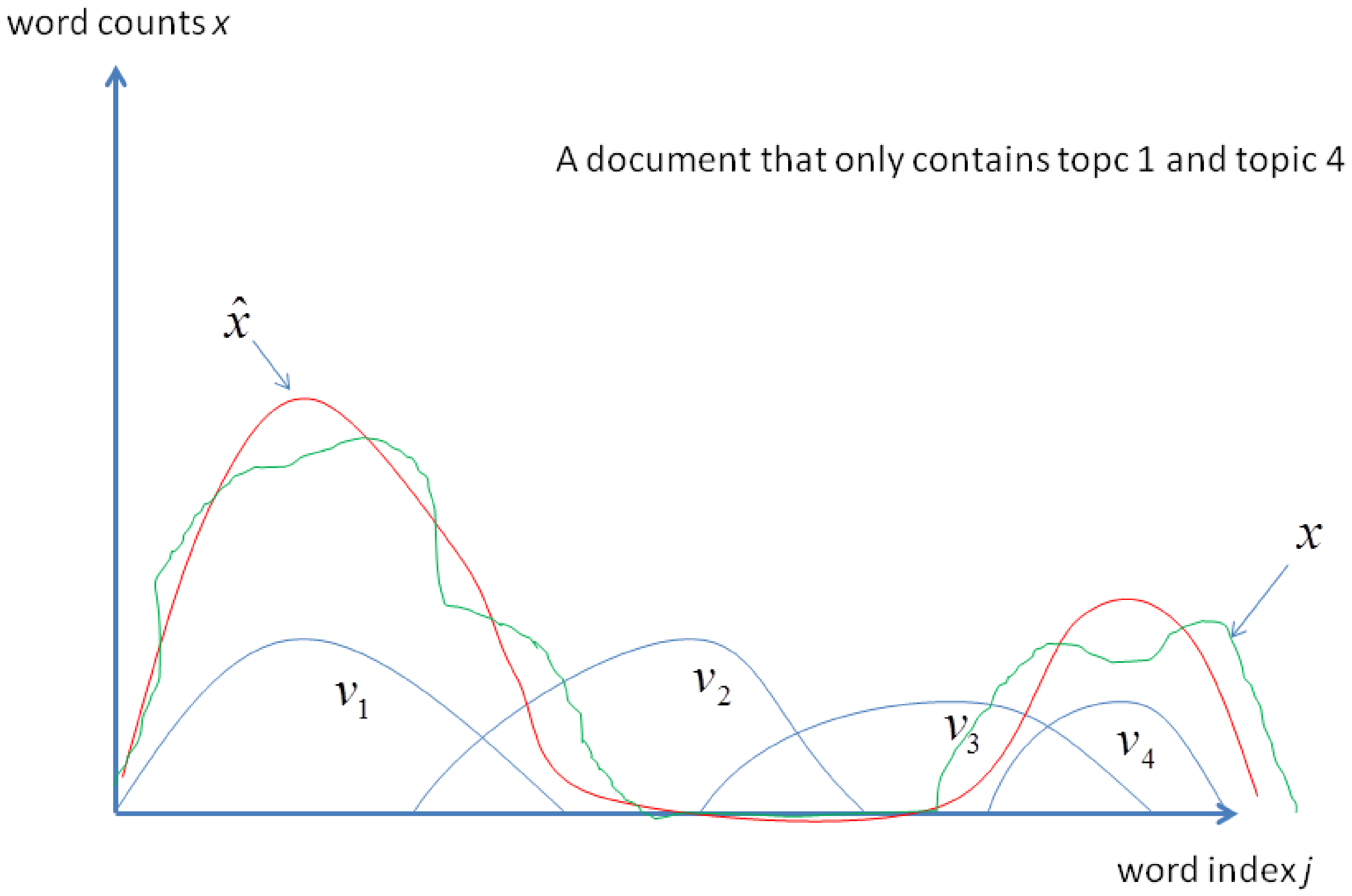
Applications of LSA

- LSA-similarity often corresponds to the human perception of document or word similarity
- There are commercial applications in the evaluation of term papers
- There are indications that search engine providers like Google and Yahoo, use LSA for the ranking of pages and to filter our Spam (spam is unusual, novel)

Illustration

- The next slides illustrate LSA, where the horizonal axis stands for the word index and the vertical axis stands for the word count
- If we consider word counts as functions of the index (functions as infinite-dimensions vectors) then the LSA (and the PCA) does function smoothing
- The columns of V would then define the basis functions (note that in the LSI the clumns would be ortonormal, in contrast to the situation displayed in the plot)
- The columns of V define patterns
- If, as shown, the columns of V have limited support, they define different subspaces


word counts x
document i
x
$\hat{x}$
v_1
v_2
v_3
v_4
word index j
subspace 2
subspace 4


word counts x
A document that only contains topc 1 and topic 4
$\hat{x}$
x
v_1
v_2
v_3
v_4
word index j

Limitations of PCA

- PCA assumes approximate normality of the input space distribution
  - PCA may still be able to produce a good low dimensional projection of the data even if the data isn't normally distributed
- PCA may not perform well if the data lies on a complicated manifold
  - Consider non-linear alternatives (ICA, LLE, diffusion map, t-SNE, ...)
- PCA assumes that the input data is real and continuous and can't deal with missing data
  - Consider probabilistic generalizations of PCA with alternative, non-Gaussian noise models

Probabilistic PCA

- $Y \sim WX + \epsilon$
  - Latent variables (PCs) $X \sim N(0, I)$
  - Noise $\epsilon \sim N(0, \Psi)$
- Chose noise model (Gaussian, Bernoulli, Poisson,...)
- Write down marginal likelihood
- Solve with maximum likelihood or EM
- Use dual version for non-linear generalizations (Gaussian Process Latent Variable Models)

Extensions

- Factorization approaches are part of many machine learning solutions
- An autoencoder, as used in deep learning, is closely related
- Factorization can be generalized to more dimensions:
- For example a 3-way array (tensor) $\underline{X}$ with dimensions subject, predicate, object. The tensor has a one where the triple is known to exist and zero otherwise. Then we can approximate (PARAFAC)

  $$\hat{x}_{i,j,l} = \sum_{k=1}^{r} a_{i,k} b_{j,k} c_{l,k}$$

  Here A contains the latent factors of the subject, B of the object, and C of the predicate
- If the entries of X are nonnegative (for example represent counts) it sometimes improves interpretability of the latent factors by enforcing that the factor matrices are nonnegative as well (nonnegative matrix factorization (NMF), probabilistic LSA (pLSA), latent Dirichlet allocation (LDA))

Extensions to observed input and output space

- Consider that we have several input dimensions and several output dimensions
- It makes sense to maintain that the latent factors should be calculated from just the input representation but that this mapping itself should be derived by also including the training outputs
- This can be done via partial least squares (pLS) or via a canonical correlation analysis (CCA)
  - PLS reconstructs the direction in the input space that explains the maximum variance in the output space
  - CCA iteratively identifies linear combinations of input and output space, that have maximum correlation with each other

Relationships to Clustering and Classification

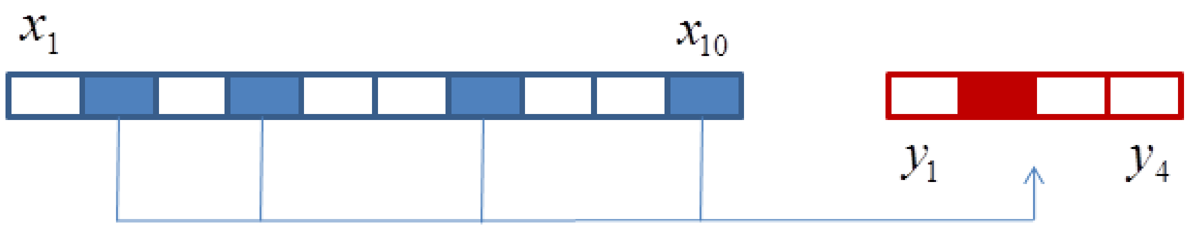
- In **multiclass classification** an object is assigned to one out of several classes. In the ground truth the object belongs to exactly one class. The classifier might only look at a subset of the inputs
- Clustering is identical to classification, only that the class labels are unknown in the training data
- In **multi-label classification** an object can be assigned to several classes. This means that also in the ground truth the object can belongs to more than one class. Each class might only look at an individual subset of the inputs. Let $\mathcal{I}_k$ be the set of inputs affiliated with class k
- Factor analysis and topic models are related to multi-label classification where the class labels (latent factors) are unknown in the training set (this interpretation works best with non-negative approaches like NMF, pLSI, LDA)
- This also leads to interpretable similarity. Consider a term-document matrix

  - Two documents i and i' are semantically similar if their topic profiles are similar, i.e. $\mathbf{z}_i \approx \mathbf{z}_{i'}$
  - Two terms j and j' are semantically similar if they appear in the same sets $\mathcal{I}_k$, i.e., if their input set profiles are similar (since $T_r = X^T U_r = V_r D_r$) (again, this interpretation works best with non-negative approaches like NMF, pLSI, LDA)

- Related in data mining: subspace clustering and frequent item set mining (the input set profiles $\mathcal{I}_k$ are the frequent item sets)

Multiclass classification / Clustering

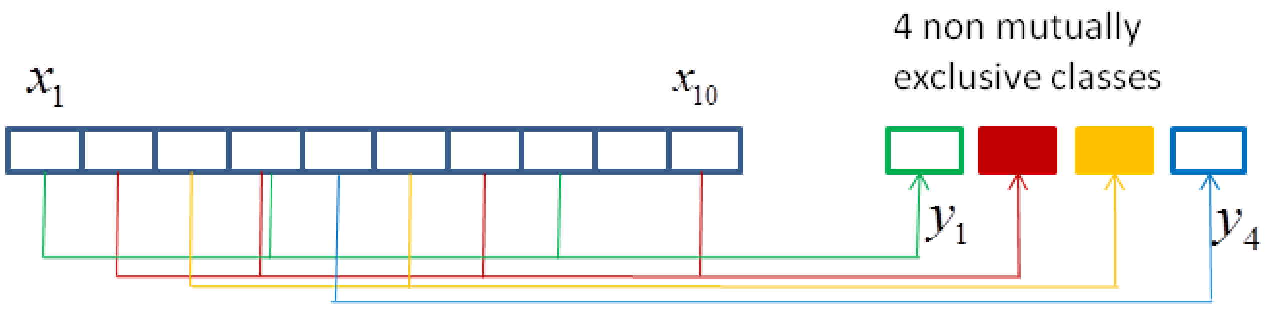
4 mutually exclusive classes



Only the four dark blue inputs are found to be important for the classification decision

- Classification: targets (y) are known during training
- Clustering: targets (y) are unknown during training

Multi-label (multi-output) Classification / Factor Analysis



- In a factor analysis the *y* are the factors and the relevant inputs are the one where the principal vectors are not close to zero
- Thus, each class *k* might be sensitive to other input sets I_k

- Examples:
  - Each data point is a document, each factor represents a topic (sports, politics, …), a document might cover several topics and the topic-specific inputs I_k are keywords relevant for classifying a topic
  - Each data point is a customer, each factor represents a buying pattern (party, baby at home, single, …), a customer might cover several buying patterns and the buying-pattern specific inputs I_k are items relevant for classifying a buying pattern (beer, pretzels) , (diapers, baby food)

- Muli-label classification: targets (y) are known during training
- Factor analysis: targets (y) are unknown during training