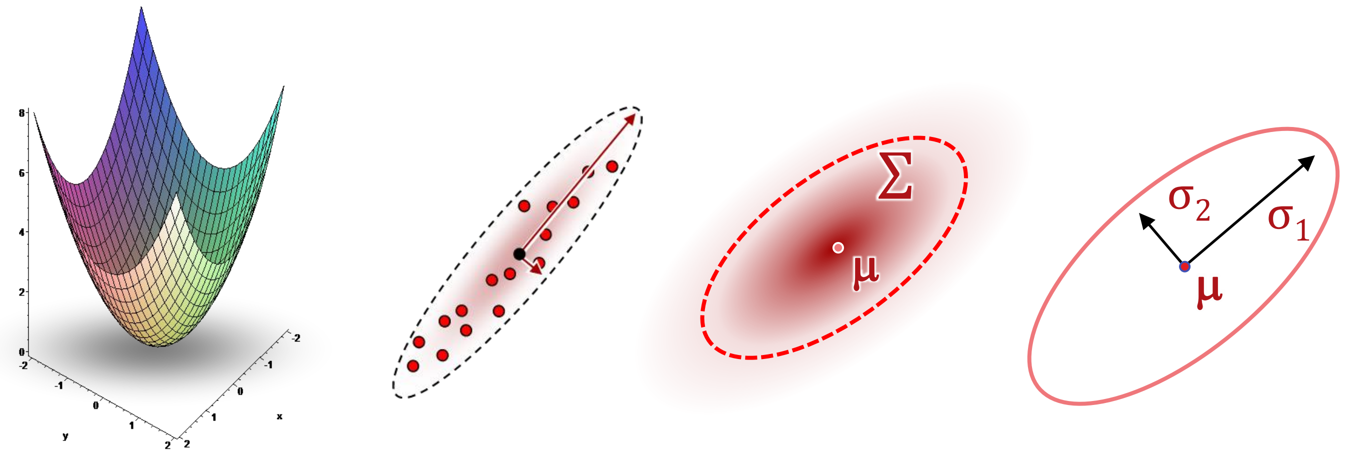
Modelling 1

SUMMER TERM 2020



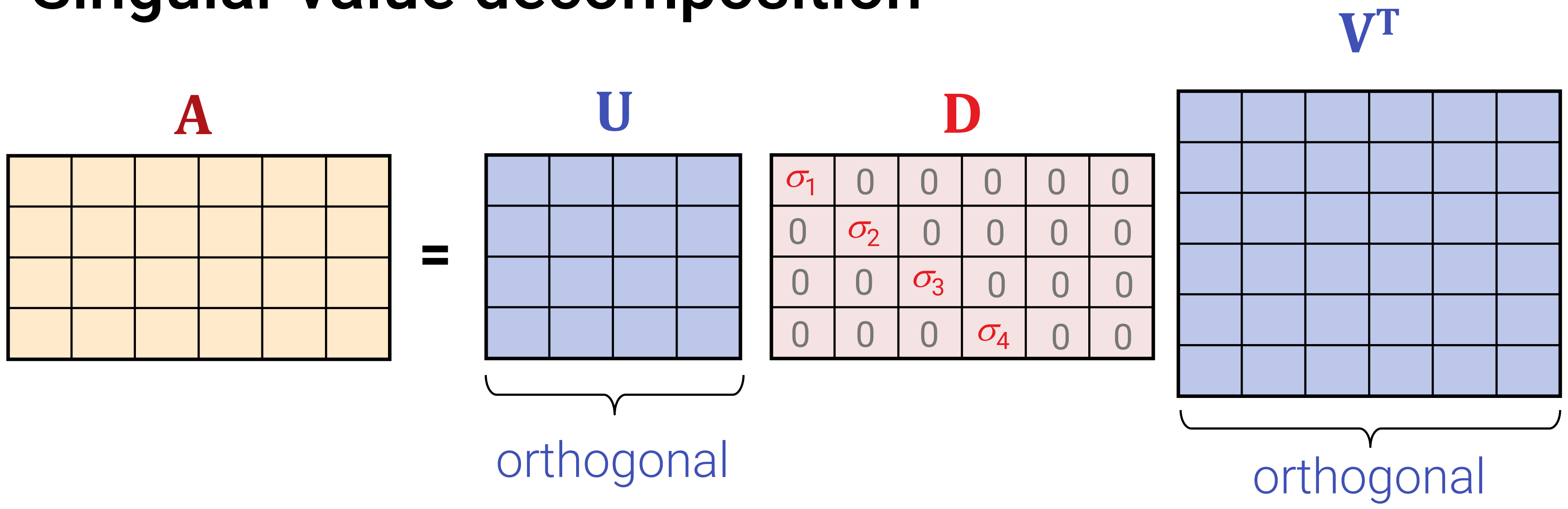
LECTURE 17

MDS and Dual PCA

Michael Wand · Institut für Informatik · Michael.Wand@uni-mainz.de

A Story about Dual Spaces

Singular value decomposition



PCA and MDS

Task: Reconstruct from Distances

Given:

- Pairwise distances between n points

$$\mathbf{D} = \begin{pmatrix} \ddots & & \iddots \\ & dist(\mathbf{x}_i, \mathbf{x}_j) & \\ \iddots & & \ddots \end{pmatrix}$$

- Points themselves $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ are not known!

Task:

- Compute $(\mathbf{x}_1, \dots, \mathbf{x}_n)$

Rough Steps

Classic Multidimensional Scaling (MDS)

- Convert distance matrix into matrix of pairwise scalar products:

$$\mathbf{G} = \begin{pmatrix} \ddots & & \iddots \\ & \langle \mathbf{x}_i, \mathbf{x}_j \rangle & \\ \iddots & & \ddots \end{pmatrix} = \mathbf{X}^{\mathrm{T}}\mathbf{X}, \qquad \mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{pmatrix}$$

- Take “square root” of $\mathbf{G}$

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} = \sqrt{\mathbf{G}}$$

(Recap from #2)

Square roots of SPD matrices

- Symmetric positive definite ("SPD") matrix $\mathbf{G}$
  - Symmetric
  - All eigenvalues positive
- $\mathbf{G}$ can be written as square of another matrix

$$\mathbf{G} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}} = \left(\mathbf{U}\sqrt{\boldsymbol{D}}\right) \cdot \left(\sqrt{\boldsymbol{D}}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\right)$$

$$\text{"}\sqrt{\mathbf{G}}\text{"} = \mathbf{U}\begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_1} \end{pmatrix}$$

More Details

Notation

Data matrix

$$\tilde{\mathbf{X}} = \begin{pmatrix} | & & | \\ \tilde{\mathbf{x}}_1 & \cdots & \tilde{\mathbf{x}}_n \\ | & & | \end{pmatrix}$$

d-dimensional input vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$

Centered data matrix

$$\mathbf{X} = \tilde{\mathbf{X}} - \bar{\mathbf{X}}$$

$$= \tilde{\mathbf{X}} - \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{x}}_{i,1} & \cdots & \frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{x}}_{\overset{\text{Zeilenindex}}{i},\overset{\text{Spaltenindex}}{1}} \\ \vdots & & \vdots \\ \frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{x}}_{i,n} & \cdots & \frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{x}}_{i,n} \end{pmatrix}$$

$$= \tilde{\mathbf{X}}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

MDS

Multi-Dimensional Scaling

- Input: $n \times n$ pairwise distances $d_{i,j}$ → matrix $\mathbf{D}$

$$\widetilde{\mathbf{D}} = -\frac{1}{2}\left[\left(d_{i,j}\right)^2\right]_{i,j}$$

- Compute “Gram matrix”
  - Pairwise scalar product matrix of centered vectors

$$\mathbf{G} = \left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\widetilde{\mathbf{D}}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

  - Explanation

$$\mathbf{G} = \mathbf{X}^T\mathbf{X} \xleftarrow{} (PCA: XX^T)$$

$$= \left(\widetilde{\mathbf{X}} - \overline{\mathbf{X}}\right)^T\left(\widetilde{\mathbf{X}} - \overline{\mathbf{X}}\right)$$

MDS (2)

Multi-Dimensional Scaling

- Next: Compute eigenstructure of $\mathbf{G} = \mathbf{X}^T\mathbf{X}$

$$\mathbf{G} = \mathbf{V_G}\mathbf{\Lambda_G}\mathbf{V_G^T} \longleftarrow \text{known!}$$

- Compare to

$$\mathbf{X} = \mathbf{U_X}\mathbf{\Lambda_X}\mathbf{V_X^T} \rightarrow \mathbf{G} = \mathbf{V_X}\mathbf{\Lambda_X^2}\mathbf{V_X^T} \longleftarrow \text{unknown!}$$

- This means, we get:

$$\mathbf{V_X} = \mathbf{V_G}, \qquad \mathbf{\Lambda_X} = \sqrt{\mathbf{\Lambda_G}}$$

- Hence: Reconstruction approach

$$\mathbf{X} \equiv \left(\sqrt{\mathbf{\Lambda_G}}\right)\mathbf{V_G^T} \longleftarrow \text{known!}$$

The Gram-matrix is invariant under orthogonal transformations of the x_i

equal up to an arbitrary rotation/reflection (U_x remains unknown)

MDS (3)

Multi-Dimensional Scaling

- Reconstruction

choosing "main axes" as coordinates
(see next slide; defined up to order/reflection)

$$\mathbf{X} \coloneqq \left(\sqrt{\boldsymbol{\Lambda}_{\mathbf{G}}}\right)\mathbf{V}_{\mathbf{G}}^{T}$$

- Distance-preserving embedding

$$\mathbf{x}_i = \left(\sqrt{\lambda_1}\mathbf{v}_1^{(i)}, \quad \dots \quad, \sqrt{\lambda_n}\mathbf{v}_{\mathrm{n}}^{(i)}\right)$$

$$\approx \left(\sqrt{\lambda_1}\mathbf{v}_1^{(i)}, \dots, \sqrt{\lambda_k}\mathbf{v}_k^{(i)}\right) \; (k \le n)$$

the rows of V_G,
diagonal entries of Λ_G

MDS (4)

Properties: MDS with Euclidian distances

- Recovers points
  - Up to global translation
  - Up to orthogonal mapping
- Reduced version (k-dim.)
  - Preserves distances in a least square sense
  - Dimensionality reduction)
- MDS is the dual of PCA
  - Result is the same:
    - MDS of distance matrix
    - PCA embedding centered point coordinates
  - Details: next slide

MDS is PCA

SVD of Centered Data Matrix

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{U}\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{\min(d,n)} \end{pmatrix}\mathbf{V}^T$$

Equivalence of MDS and PCA

PCA: $\mathbf{S} = \mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^T$

MDS: $\mathbf{G} = \mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^T$

If we know X, we can compute U, too!
Not possible from distances/scalar-prod. only.

$$\left.\begin{array}{ll} \textbf{PCA:} & emb_{PCA}(\mathbf{X}) = \mathbf{U}^T\mathbf{X} \\ \textbf{MDS:} & emb_{MDS}(\mathbf{X}) = \mathbf{\Lambda}\mathbf{V}^T \end{array}\right\} \longleftarrow \mathbf{X}^T = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^T \Rightarrow \mathbf{U}^T\mathbf{X} = \mathbf{\Lambda}\mathbf{V}^T$$

So Where is the Difference?

	PCA	**MDS**
input	points	pair-wise distances / scalar products
complexity	$d \times d$ eigenvalue problem (low dim., large data sets)	$n \times n$ eigenvalue problem (high dim., small data sets)
result	data embedding, principal variances, principal axes ("U")	data embedding, principal variances, no principal axes*)
subspace projection	can easily embed additional vectors	not obvious (yes: Nyström method)

*) Unless we know the original data X.

Nyström Projection

Embedding further Vectors

- Recompute everything
  - Expensive
  - Inconsistent for some applications (new coordinates)
- “Nyström Formula”
  - Compute embedding by linear combination of computed eigenvectors
  - Uses projections on input data set (scalar products only)
  - Assumes knowledge of point positions (later: measure distances only)

Nyström Projection

Nyström Projection

- **Reminder:** $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ $\quad \mathbf{G} = \mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^T$

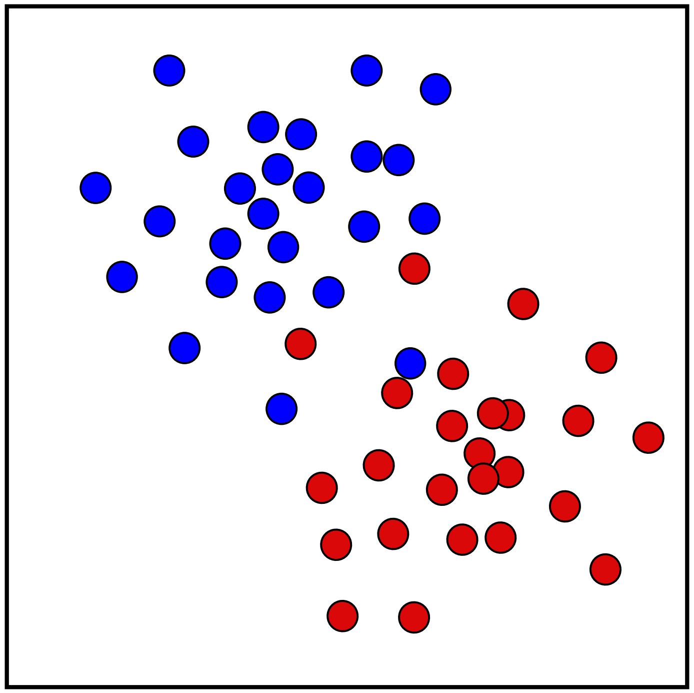
  $emb_{MDS}(\mathbf{X}) = \mathbf{\Lambda}\mathbf{V}^T \quad emb_{PCA}(\mathbf{X}) = \mathbf{U}^T\mathbf{X}$
- **Project vector $\mathbf{x}$ on principal axes $\mathbf{u}_1, \ldots, \mathbf{u}_d$:**

$$
\begin{aligned}
emb(\mathbf{x}) &= \mathbf{U}^T\mathbf{x} \\
&= (\mathbf{V}^T\mathbf{\Lambda}^{-1}\mathbf{X}^T)\mathbf{x} \\
&= \begin{pmatrix} \sum_{i=1}^{n} \frac{1}{\lambda_1} v_{i,1}\langle \mathbf{x}_i, \mathbf{x}\rangle \\ \vdots \\ \sum_{i=1}^{n} \frac{1}{\lambda_n} v_{i,n}\langle \mathbf{x}_i, \mathbf{x}\rangle \end{pmatrix}
\end{aligned}
$$

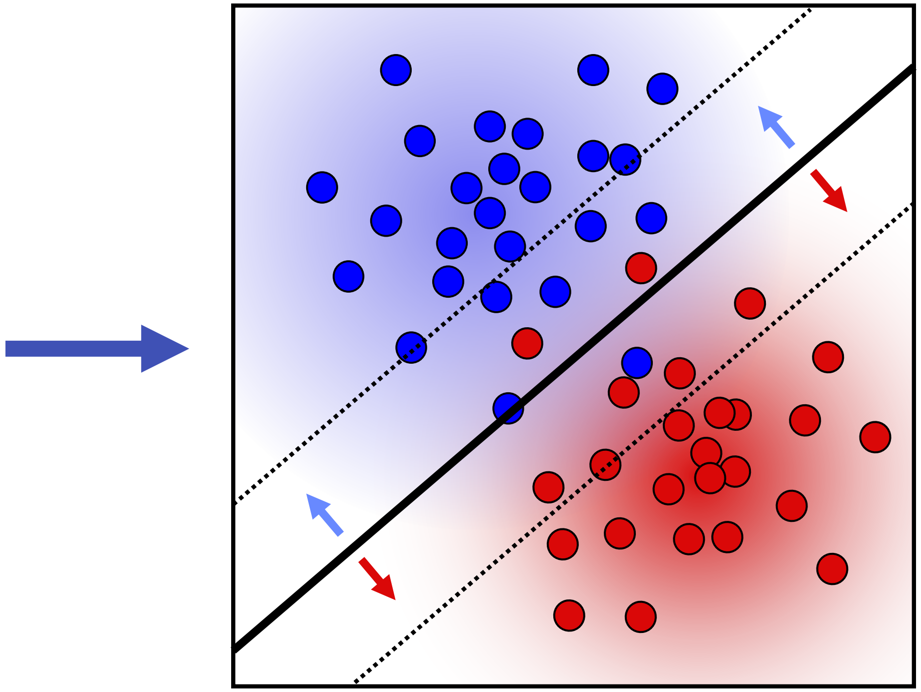
$$
\begin{aligned}
\mathbf{X} &= \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \\
\Rightarrow \mathbf{U} &= \mathbf{X}\mathbf{V}\mathbf{\Lambda}^{-1} \\
\Rightarrow \mathbf{U}^T &= \mathbf{V}^T\mathbf{\Lambda}^{-1}\mathbf{X}^T
\end{aligned}
$$

Kernel PCA

Support Vector Machines



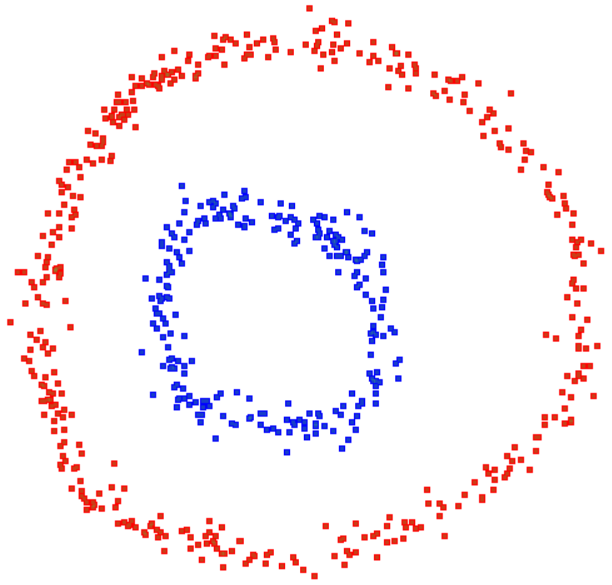
training set



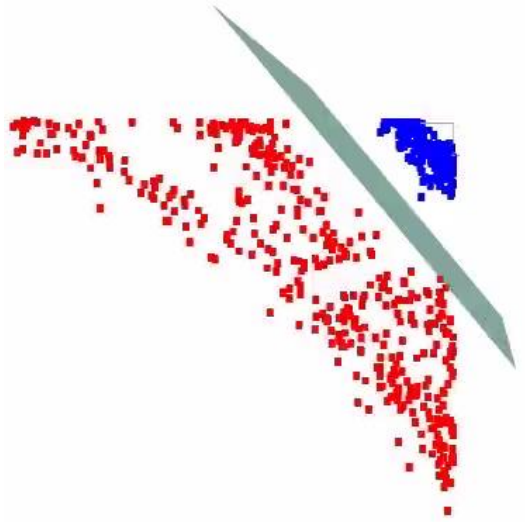
separating hyperplane, minimal penetration of margin (L_1)

Kernel Support Vector Machine

Example Mapping:



original space



"feature space"

$$\phi \colon \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x, y) \mapsto \left(x^2, xy, y^2\right)$$

Kernel PCA

“Kernel PCA is classical scaling in feature space” [Williams 2002]

Main Idea:

- MDS can be easily “kernelized” – just replace scalar product matrix $\mathbf{G}$ with kernel matrix
- No need to deal with feature space explicitly (which might be intractable)
- Will yield PCA anyway (but no eigenvectors)

Kernel PCA

Summary:

- Kernel PCA performs PCA/MDS in feature space using dot products (i.e. kernel evaluations) only
- It gives the same result as MDS

Kernel PCA

Remarks:

- Unlike “real” PCA, it does not output principal axes vectors
  - They are in feature space, i.e. usually inaccessible
  - Preimages in (low-dim.) input space do not need to exist (there are approximation techniques)
  - Even if so, they are difficult to compute...
    ...and the space is non-linear anyway
    (so they do not really help)

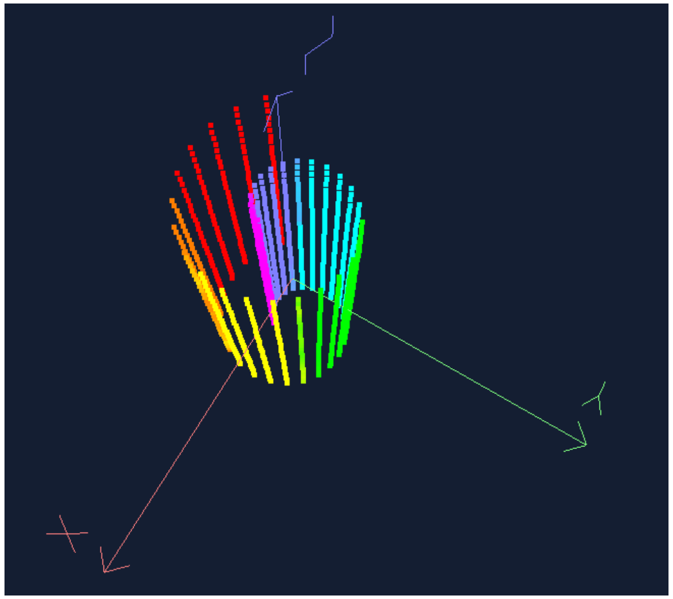
Kernel PCA

Complexity:

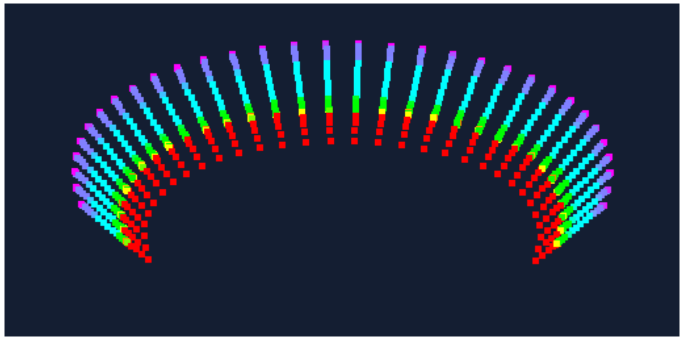
- Need to solve $n \times n$ eigenvalue problem
- Memory, Time: $\Omega(n^2)$
- Does not scale for large data sets
  - Can use approximation techniques
  - Idea: Landmark MDS
  - Compute embedding on small subset of landmark points (e.g. random subset)
  - Use Nyström formula to embed other points

Examples

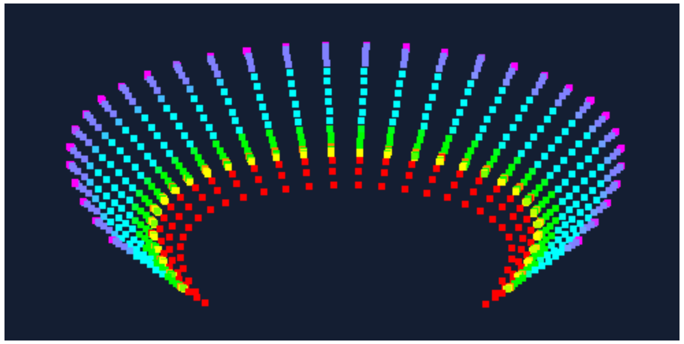
The (In)famous Swiss Roll?



the roll

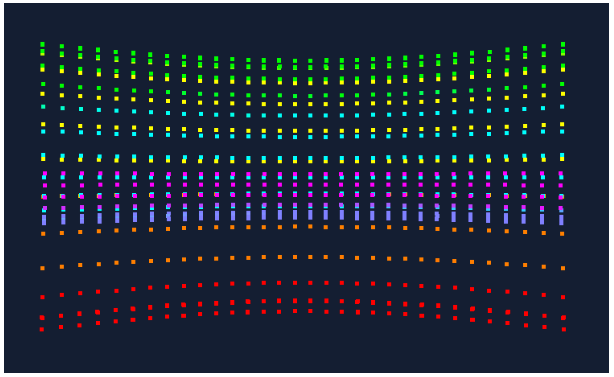


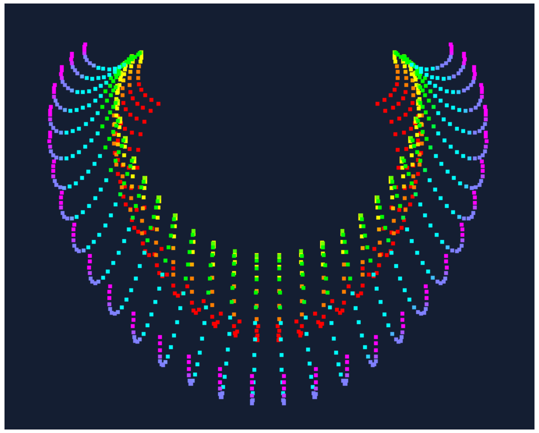
exp. kernel $\sigma = 0.30D$



exp. kernel $\sigma = 0.35D$

poly.-kernel (5th order) $\sigma = 0.35D$

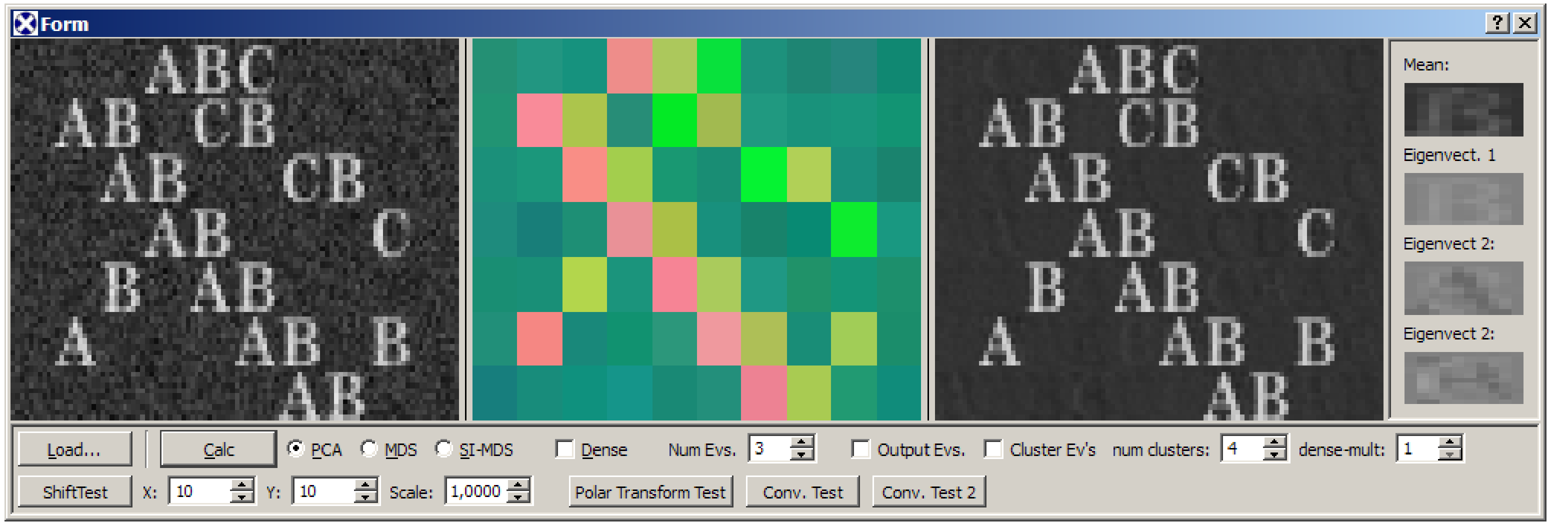




exp. kernel $\sigma = 0.35D$ (centered)

What Else Can You Do?

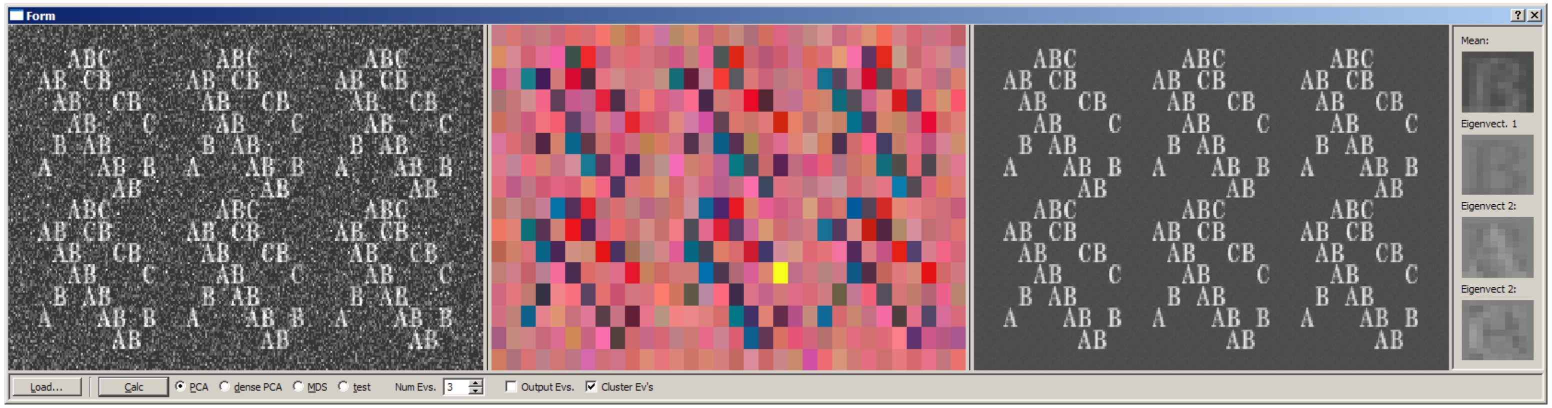
Image Denoising via PCA:



[✓ PCA] MDS SI-MDS

☐ Dense Num Evs.: 3 ☐ Output Evs. ☐ Cluster Ev's num clusters: 4 dense-mult: 1

X: 10 Y: 10 Scale: 1,0000

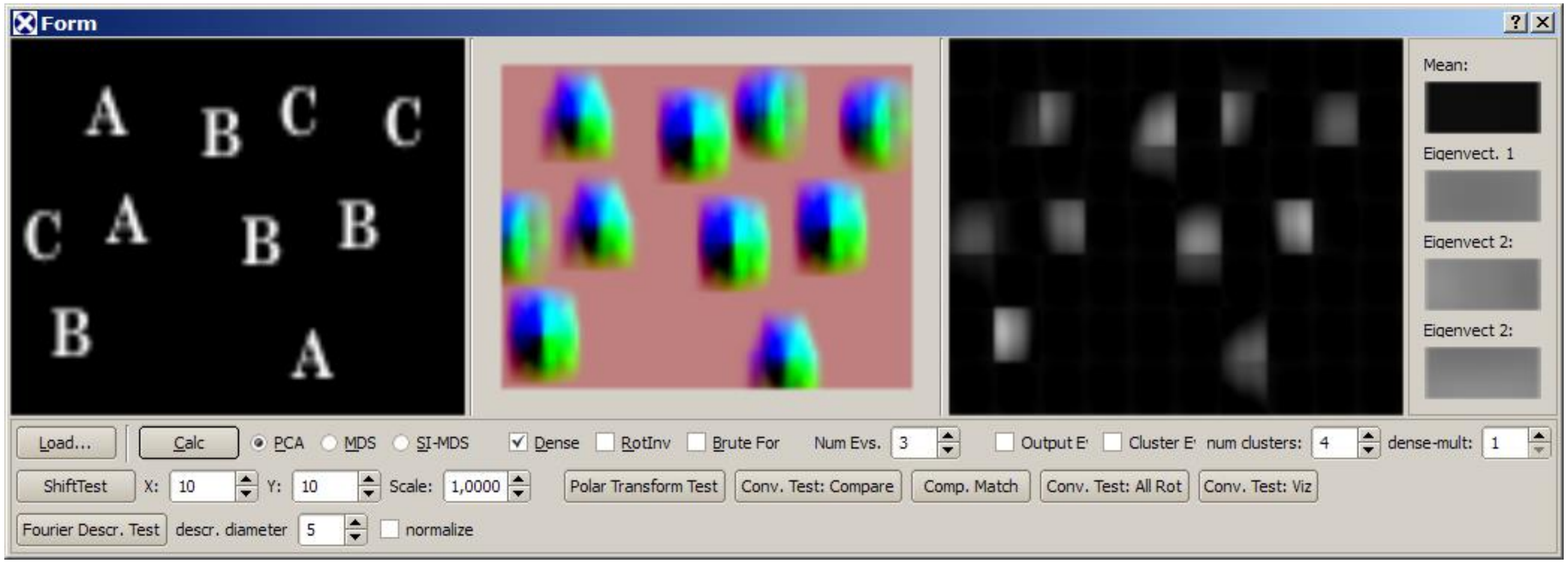


[✓ PCA] dense PCA MDS test

Num Evs.: 3 ☐ Output Evs. ☑ Cluster Ev's

What Else Can You Do?

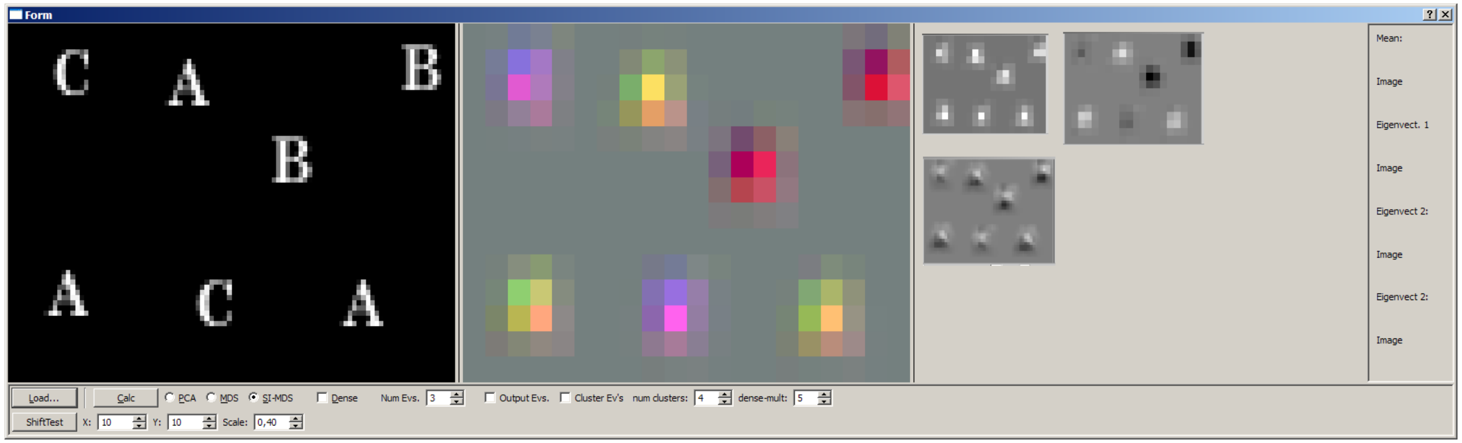
Does not work without correspondences:



[✓ PCA] MDS SI-MDS
[✓ Dense] RotInv Brute For
Num Evs. 3
Output E Cluster E num clusters: 4 dense-mult: 1
X: 10 Y: 10 Scale: 1,0000
descr. diameter 5 normalize

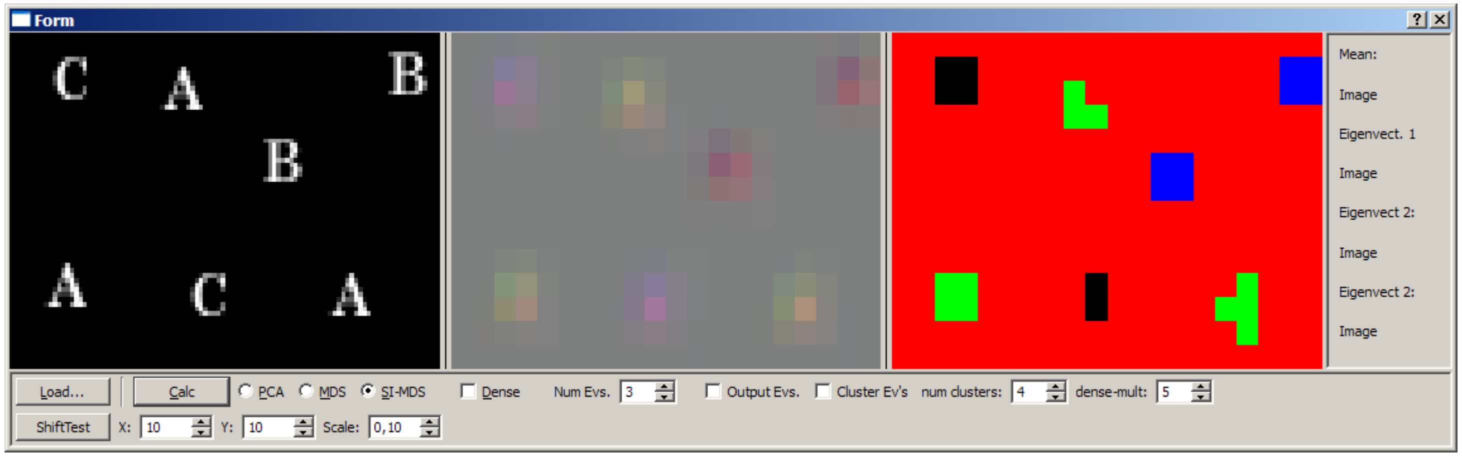
What Else Can You Do?

Shift invariant comparison kernel:



( ) PCA ( ) MDS [✓ SI-MDS] [ ] Dense Num Evs. 3 [ ] Output Evs. [ ] Cluster Ev's num clusters: 4 dense-mult: 5

X: 10 Y: 10 Scale: 0,40

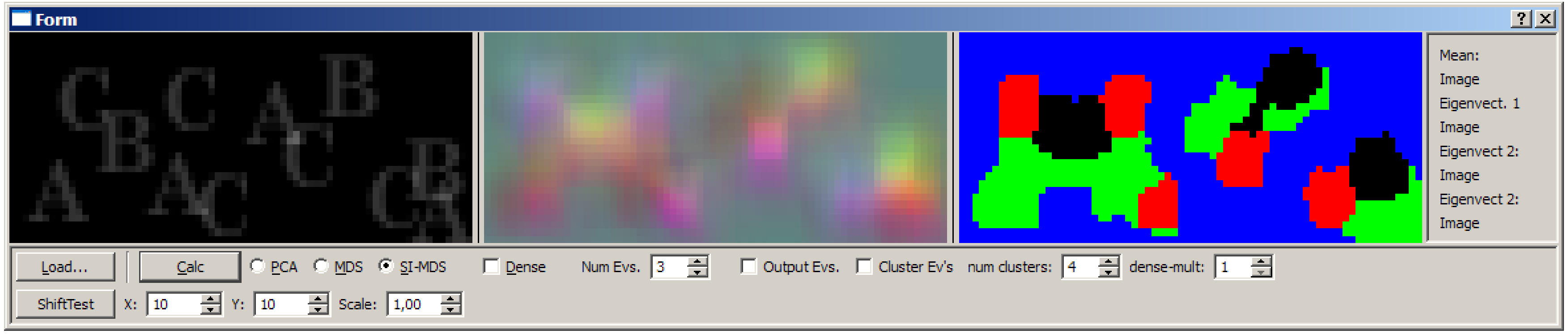


( ) PCA ( ) MDS [✓ SI-MDS] [ ] Dense Num Evs. 3 [ ] Output Evs. [ ] Cluster Ev's num clusters: 4 dense-mult: 5

X: 10 Y: 10 Scale: 0,10

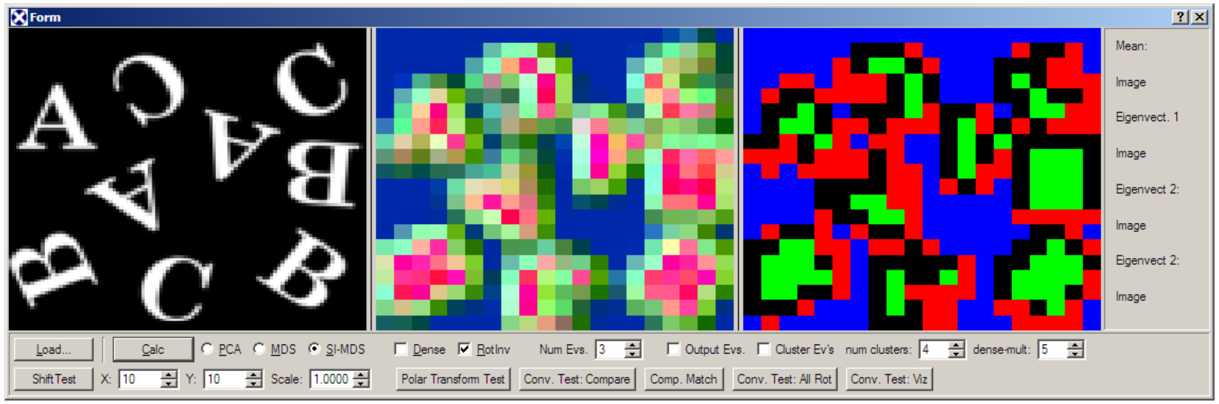
MDS (Kernel PCA)

Shift invariant comparison kernel:



PCA MDS [✓ SI-MDS] Dense Num Evs. 3 Output Evs. Cluster Ev's num clusters: 4 dense-mult: 1

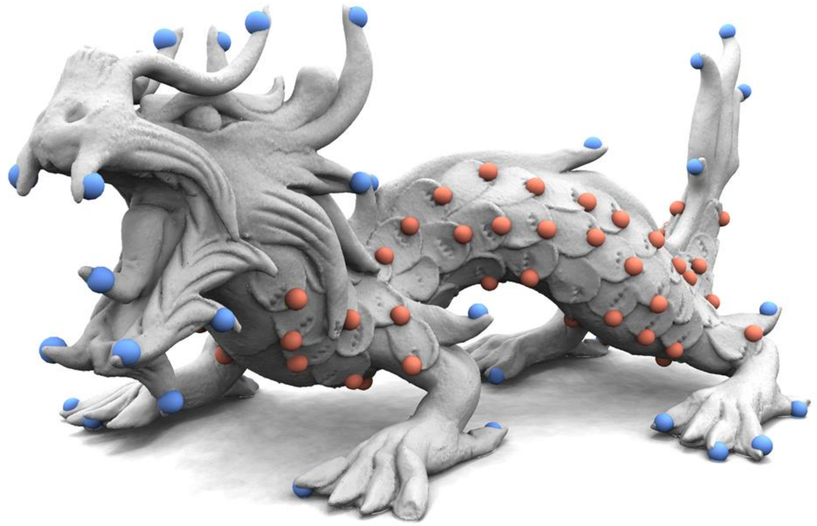
X: 10 Y: 10 Scale: 1,00



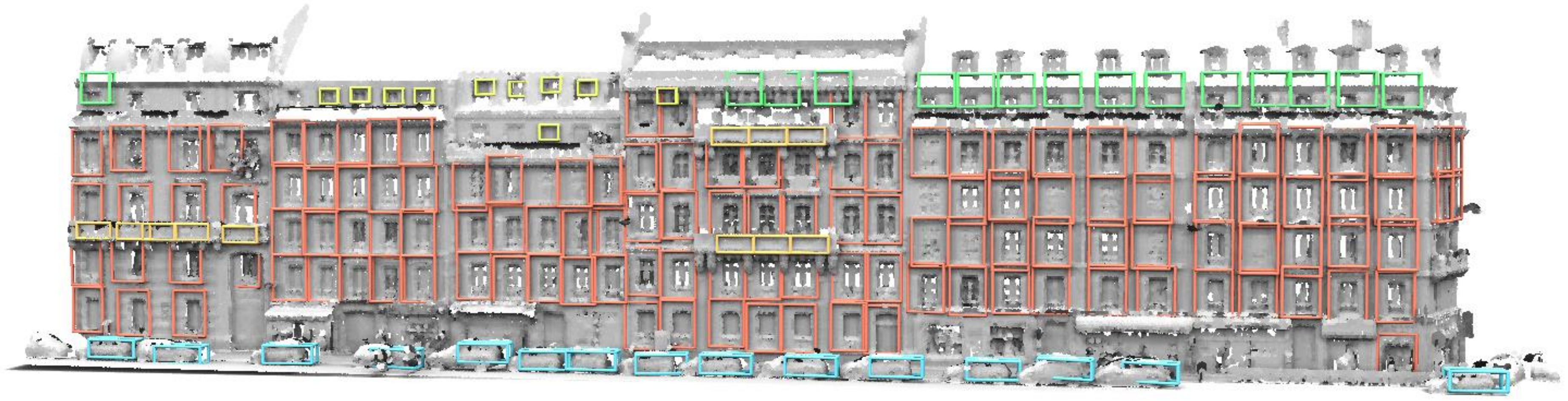
PCA MDS [✓ SI-MDS] Dense [✓ RotInv] Num Evs. 3 Output Evs. Cluster Ev's num clusters: 4 dense-mult: 5

X: 10 Y: 10 Scale: 1.0000

Co-Occurrence Clustering

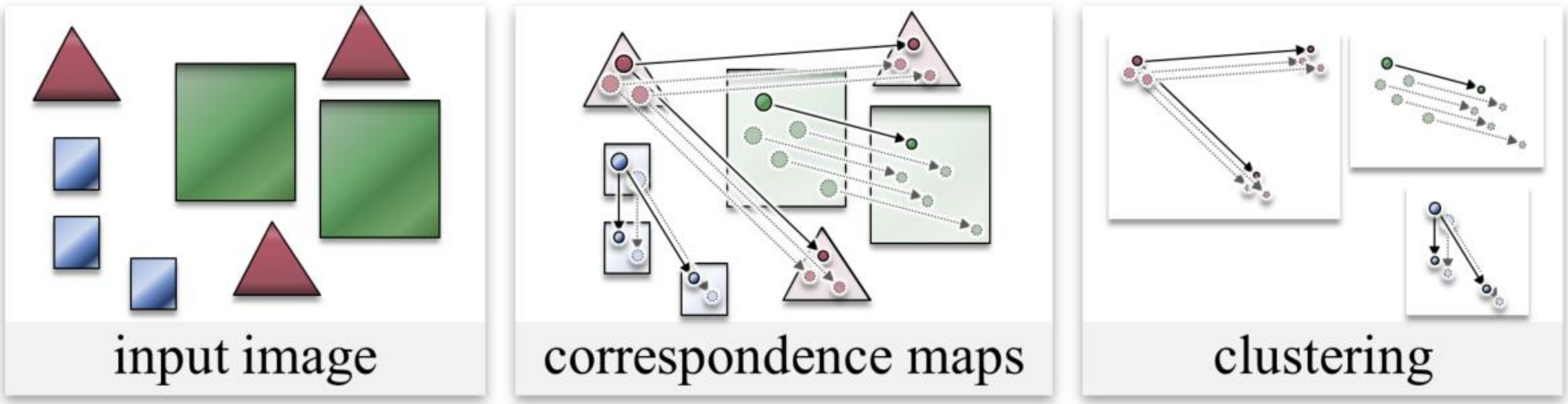


3D point clouds



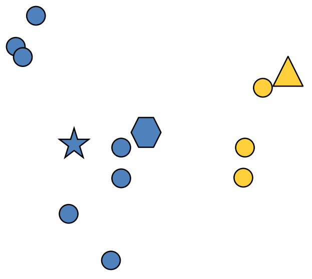
[Chuan Li et al., 3DV 2015]

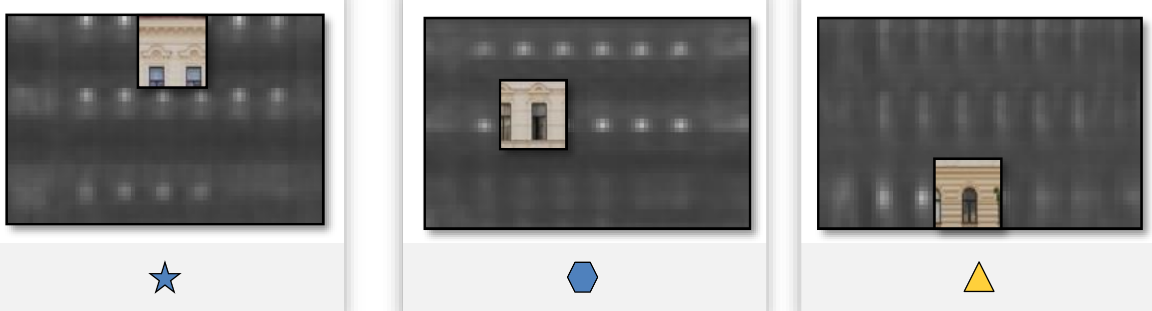
Co-Occurrence Clustering



Spectral Clustering

Input image


Embedding



Co-occurrence embedding

MDS / Kernel PCA References

B. Schölkopf, A. J. Smola, K.-R. Müller: Nonlinear Component Analysis as a Kernel Eigenvalue Problem. In: Neural Computation, 10:1299-1319, 1998.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Ratsch, A. J. Smola: Input Space versus Feature Space in Kernel-Based Methods. In: IEEE Trans. on Neural Networks, 10:1000-1017, 1999.

C. Williams: On a Connection between Kernel PCA and Metric Multidimensional Scaling. In: Machine Learning, 46:11-19, 2002.

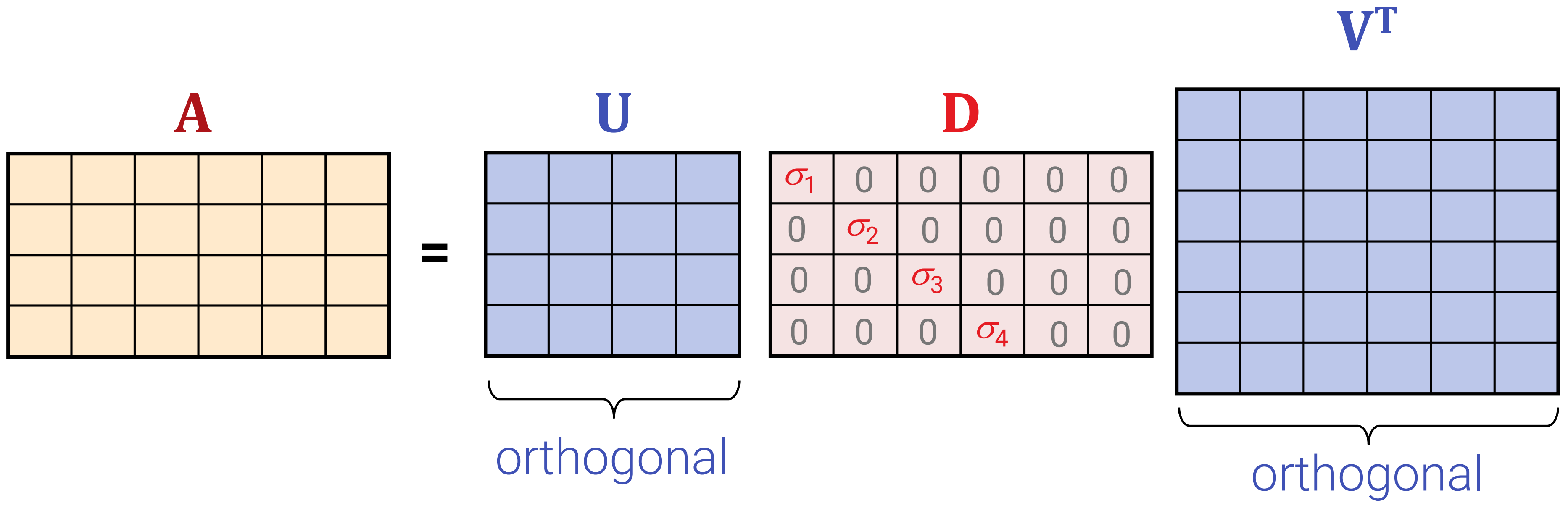
K.-R. Müller, S. Mika, G. Ratsch, K. Tsuda, B. Schölkopf: An Introduction to Kernel-Based Learning Algorithms. In: IEEE Trans. on Neural Networks, 12:181-201, 2001.

J. Shawe-Taylor, N. Cristianini: Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.

T. Cox, M. Cox: Multi-Dimensional Scaling. Chapman & Hall, 1994.

Matrix Factorization and Recommender Systems

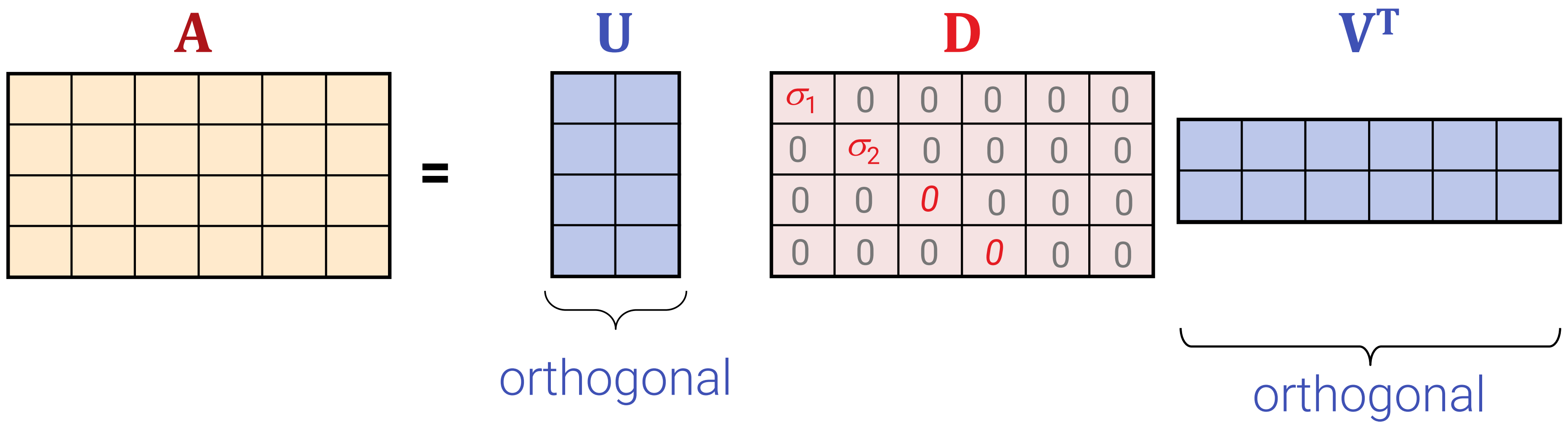
Matrix Factorization



Key ingredient: Spectral factorization

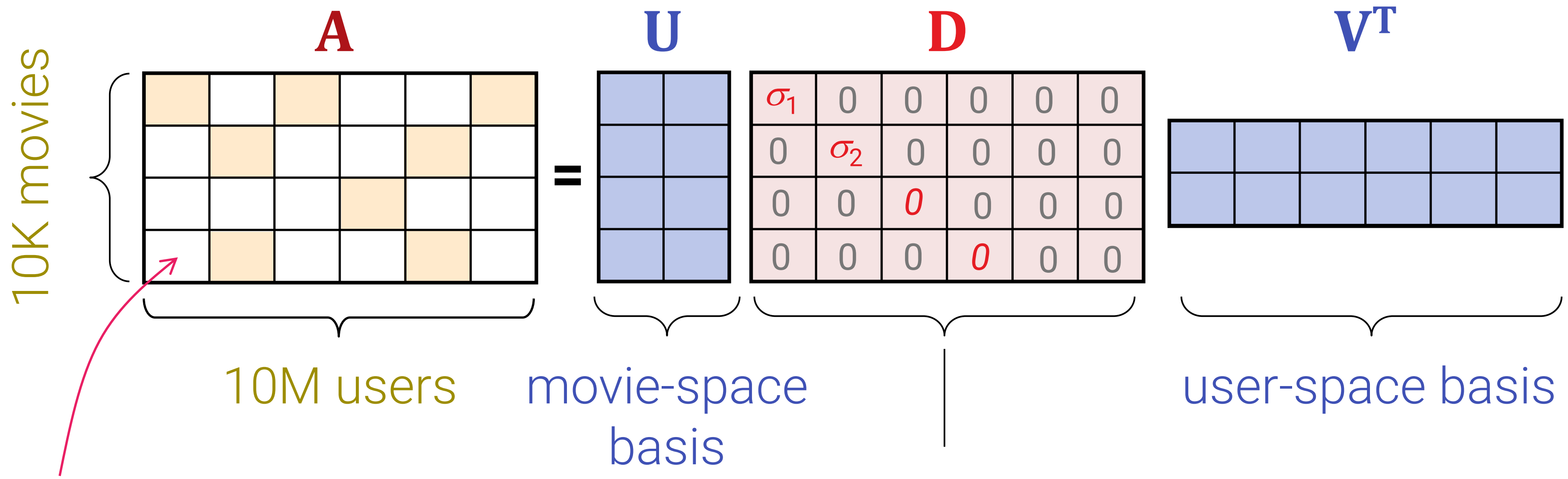
- Requires dense matrix $\mathbf{A}$
- Low-rank approximations: Largest σ_i only

Low-Rank Factorization



Recommender Systems

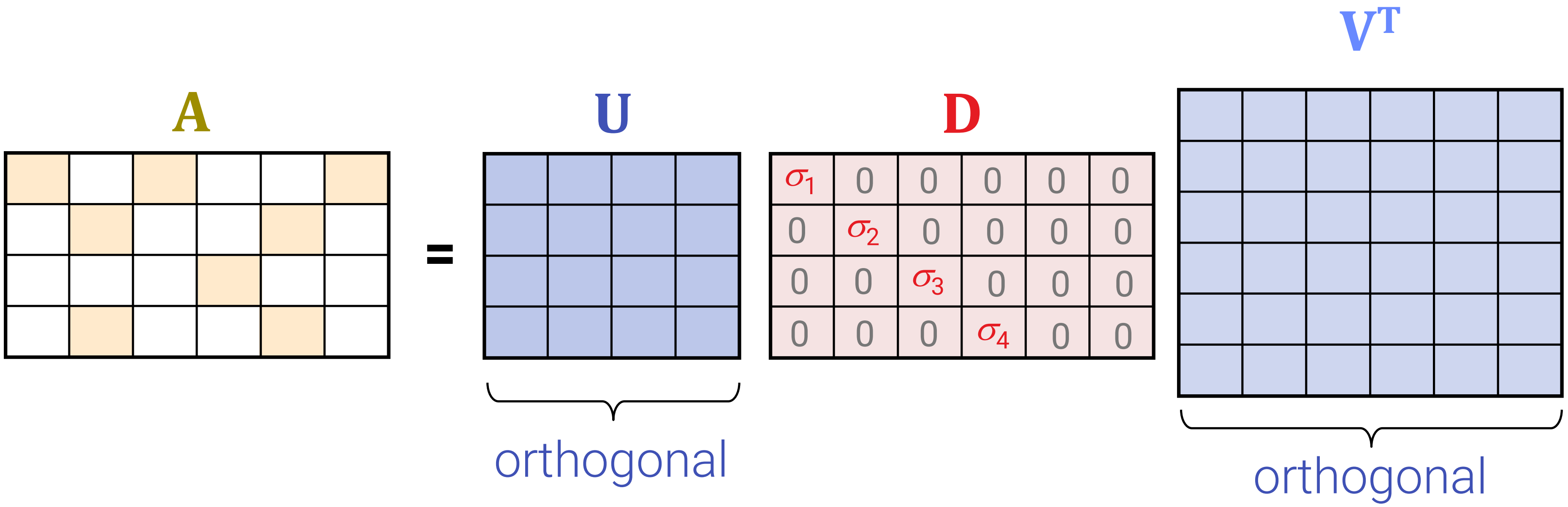
Example: Movie recommendations



on average 20
"likes" per user
score ∈ [0,1]

people's taste
is highly correlated
→ low-dimensional
subspace

Sparse Matrix Factorization

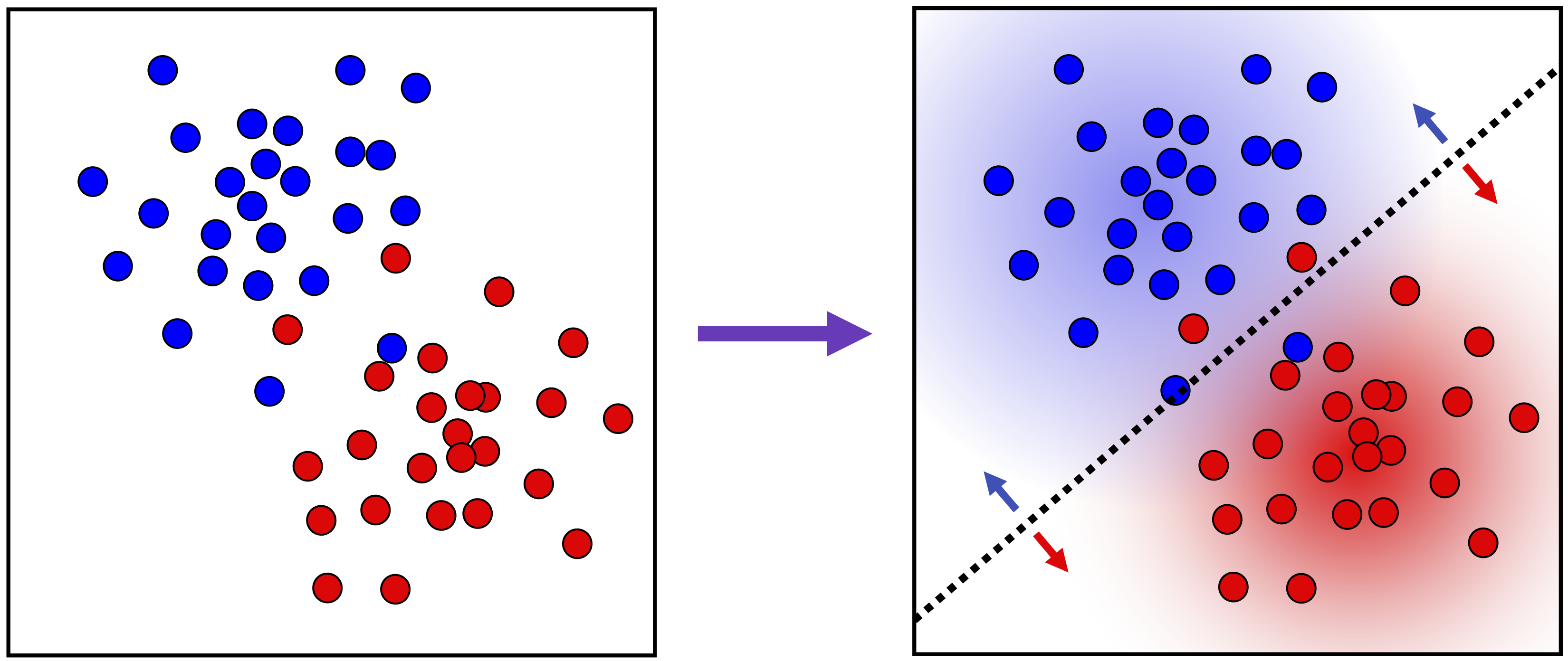


Key ingredient: Spectral factorization

- Objective $\left\|\mathbf{A} - \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}\right\|^2$
- Approximation only (optimum NP-hard)
- Popular: Alternating least-squares

Kernel PCA
(& Kernel Learning)

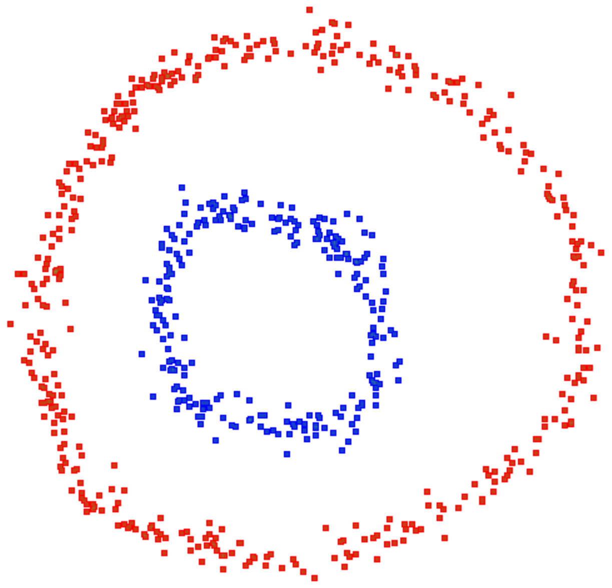
Example: Support Vector Machine

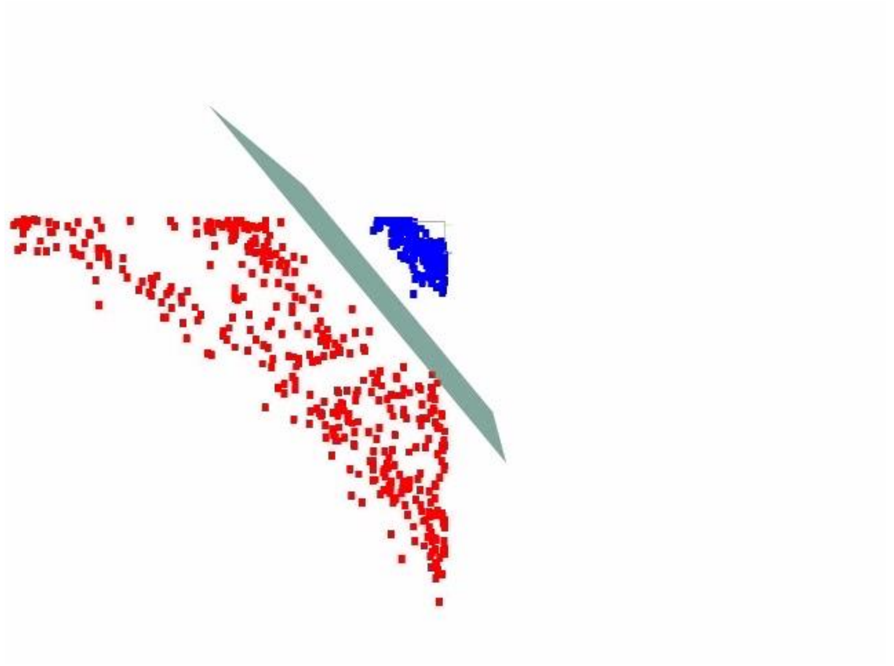


labeled sample

reconstructed density, decision rule

Example


original space


"feature space"

Example Mapping:

$$\mathbb{R}^2 \to \mathbb{R}^3$$
$$(x, y) \to (x^2, xy, y^2)$$

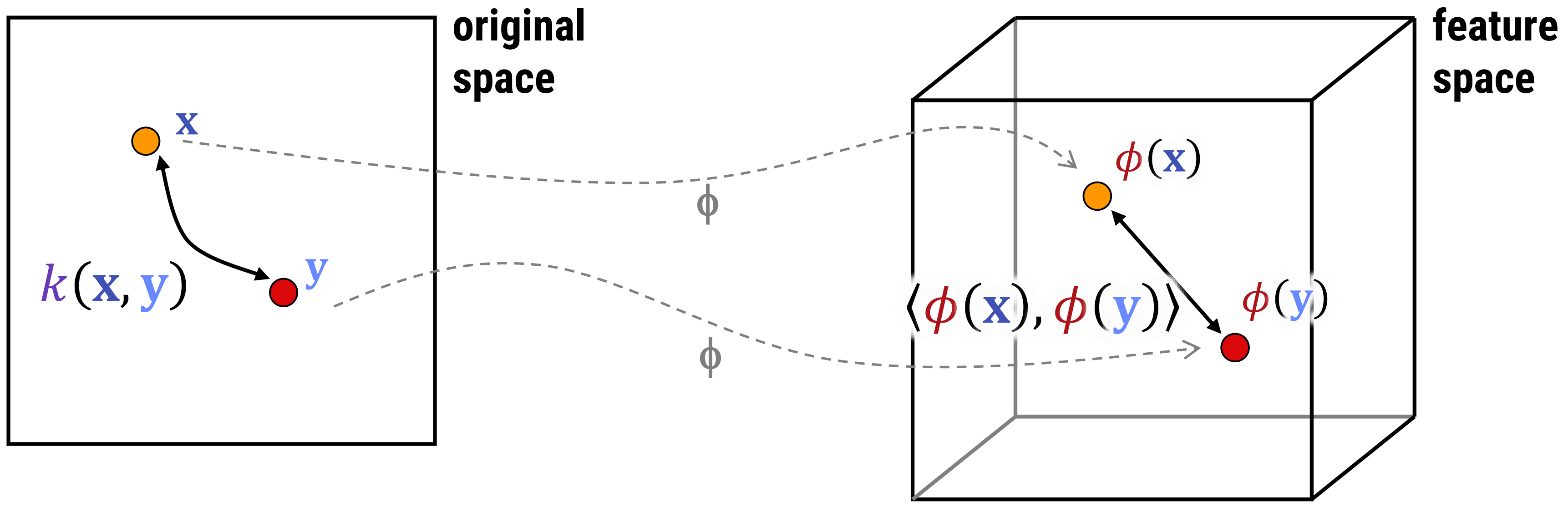
“The Kernel Trick”

Observation:

- Many data analysis algorithms can be expressed in terms of scalar products only
- Scalar products $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ can sometimes be computed efficiently, without explicit mapping
- **“Kernel trick”:** replace standard scalar product with kernel function:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y})$$

“The Kernel Trick”



tractable dimensionality (typ. 10D – 100D)

intractable dimensionality (typ. 10^{10}D – inf D)

Kernels Design

Kernel Design

- Converting $\phi \rightarrow k(\cdot, \cdot)$ is difficult
- Other way round:
  choose $k(\cdot, \cdot)$ that correspond to useful ϕ

Mercer kernels

- Conditions for valid kernels
  - Eigenfunctions of positive, symmetric kernels
- Sufficient:
  - Finite positivity property –
    Any matrix of pairwise scalar products of finite point sets is symmetric positive definite

Standard Kernels

Polynomial Kernel

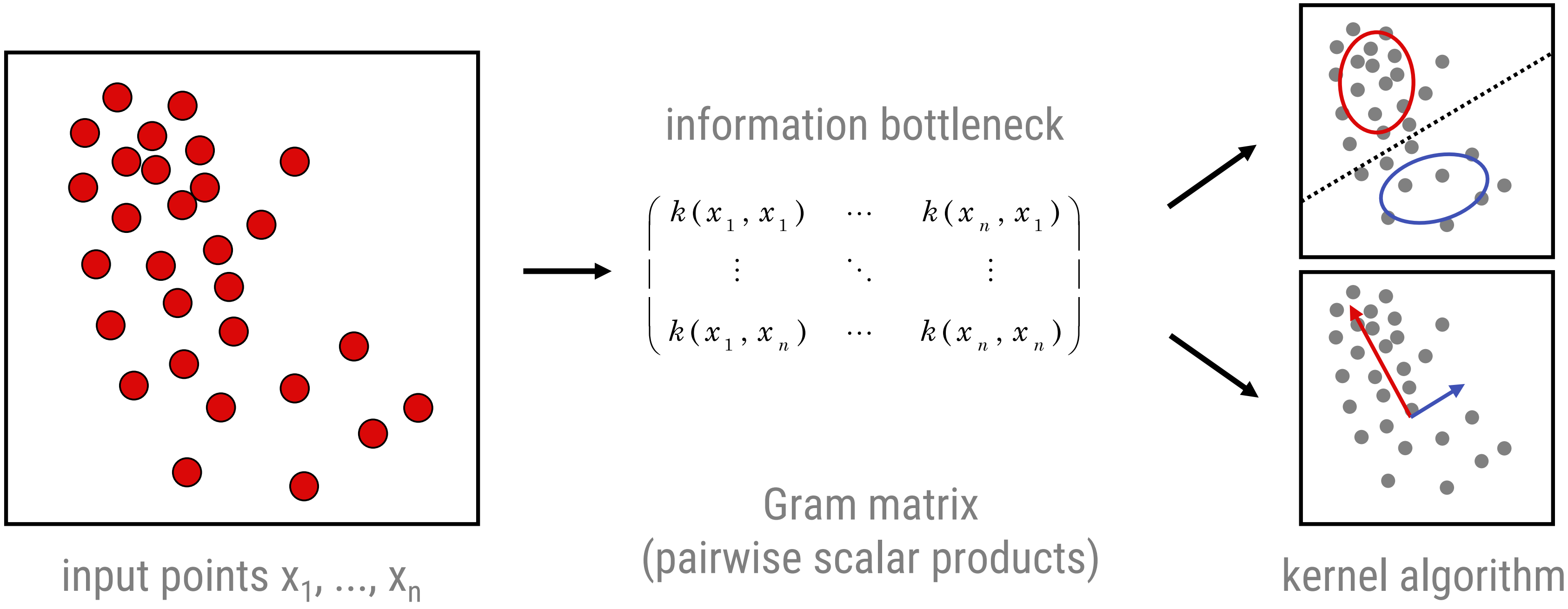
- $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d$
- Corresponds to multivariate monomials up to degree d

Exponential Kernel

- $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|/2\sigma^2)$
- Corresponds to infinite dimensional feature space

Kernel Algorithms

General Scheme for Kernel Algorithms



$$\begin{pmatrix} k(x_1,x_1) & \cdots & k(x_n,x_1) \\ \vdots & \ddots & \vdots \\ k(x_1,x_n) & \cdots & k(x_n,x_n) \end{pmatrix}$$

(c.f. Johnson-Lindenstrauss Lemma: pairwise distances provide less information than vectors themselves)

High-Dimensional Spaces are Weird

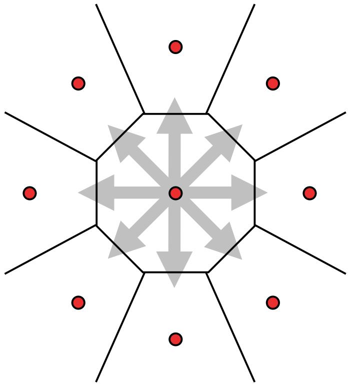
How Much Information is Contained in Pairwise Distances?

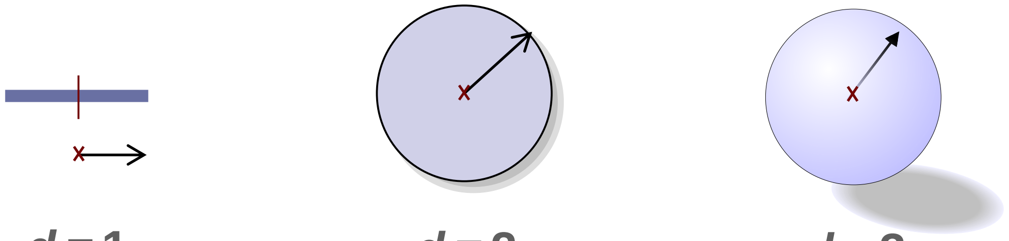
Higher Dimensions are Weird

Issues with High-Dimensional Spaces :

- *d*-dimensional space:
  d independent neighboring directions to each point

- Volume-distance ratio explodes





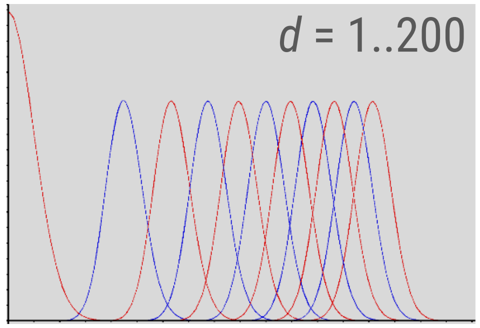
$\text{vol}(r) \in \Theta(r^d)$

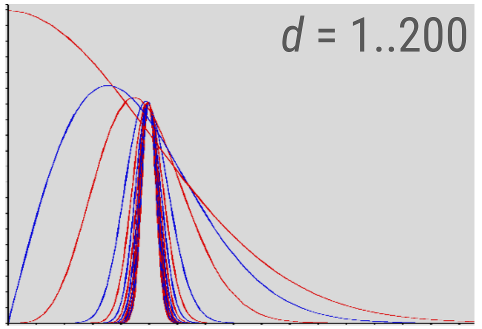
$d = 1$ $d = 2$ $d = 3$ $d \to \infty$

Higher Dimensions are Weird

More Weird Effects:

- Dart-throwing anomaly
  - Normal distributions gather prob.-mass in thin shells
  - [Bishop 95]





- Nearest neighbor ~ farthest neighbor
  - For unstructured points (e.g. iid-random)
  - Not true for certain classes of structured data
  - [Beyer et al. 99]

Johnson-Lindenstrauss Lemma

JL-Lemma: [Dasgupta et al. 99]

- Point set P in $\mathbb{R}^d$, $n := \#P$
- There is $f: \mathbb{R}^d \to \mathbb{R}^k$, $\quad k \in O(\varepsilon^{-2} \ln n)$
  $(k \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \ln n)$
- …that preserves all inter-point distances up to a factor of $(1 + \varepsilon)$

Random orthogonal linear projection

- Works with probability $\geq (1\text{-}1/n)$

This means…

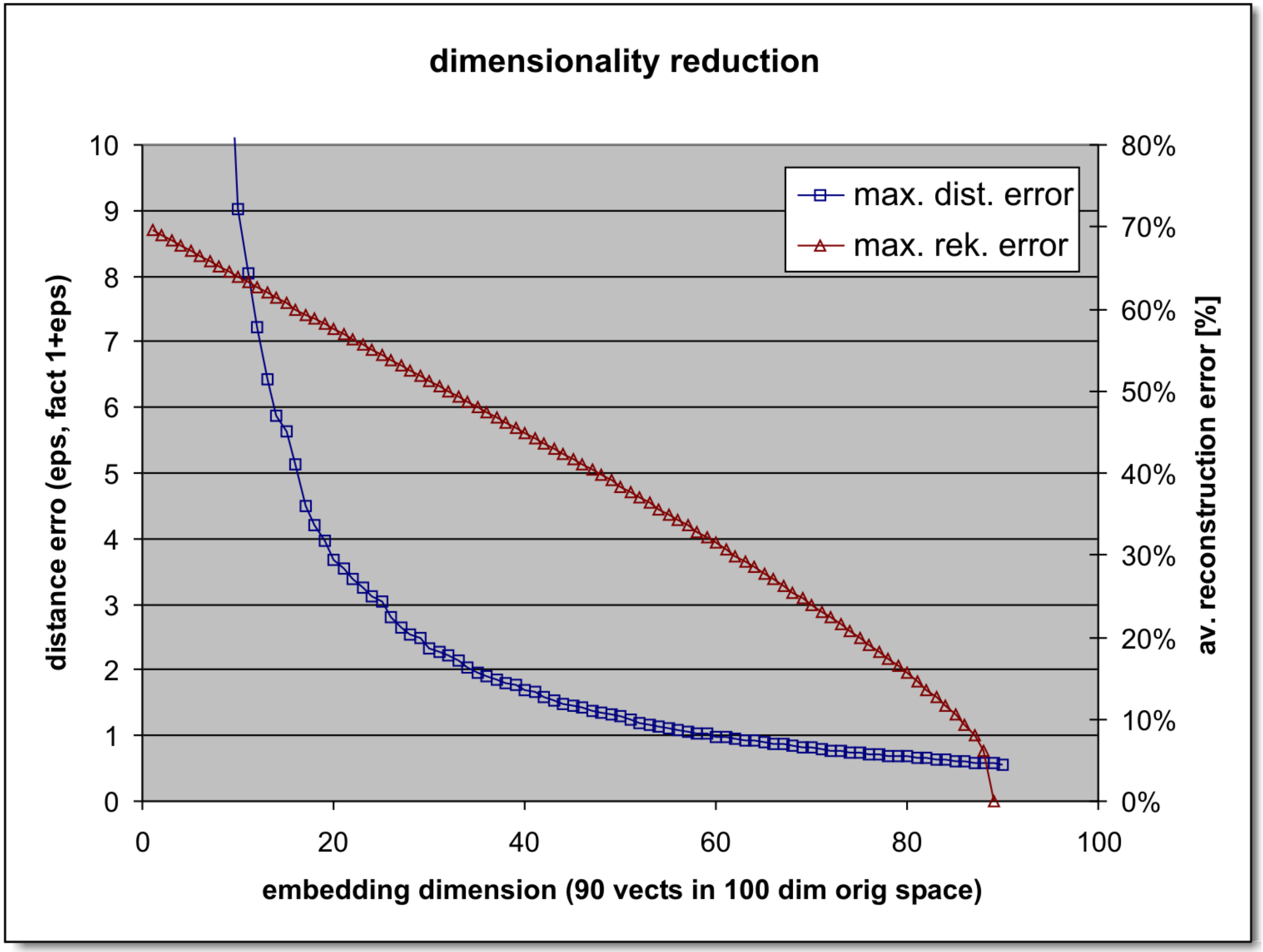
What Does the JL-Lemma Imply?

Pairwise distances in small point set P
(sub-exponential in d)
can be well-preserved in low-dimensional embedding

What does it not say?

Does not imply that the points *themselves* are well-represented (just the pairwise distances)

Experiment

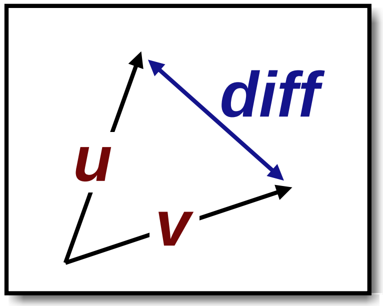


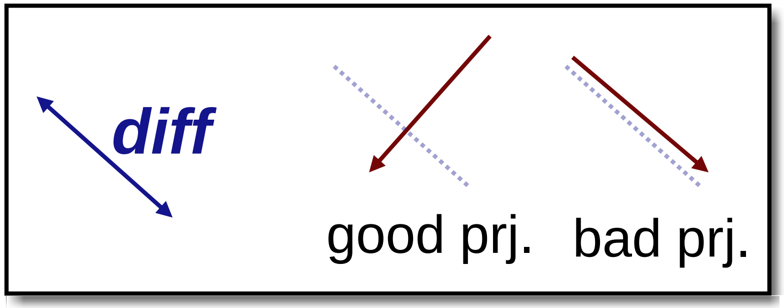
dimensionality reduction
10
9
8
7
6
5
4
3
2
1
0
80%
70%
60%
50%
40%
30%
20%
10%
0%
max. dist. error
max. rek. error
distance erro (eps, fact 1+eps)
av. reconstruction error [%]
0
20
40
60
80
100
embedding dimension (90 vects in 100 dim orig space)

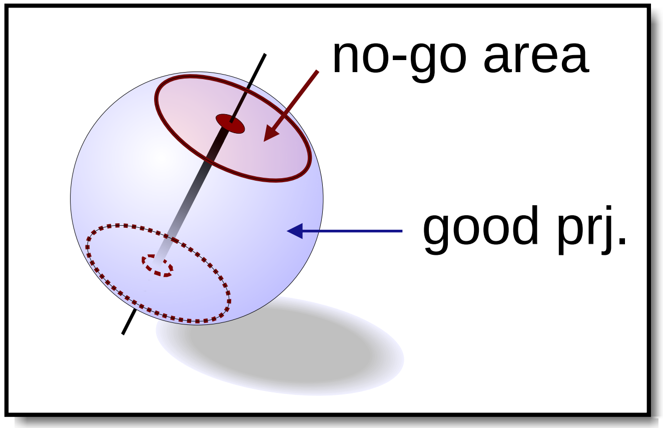
Intuition

Difference Vectors

- Normalize (relative error)
- Pole yields bad approximation
- Non-pole area much larger (high dimension)
- Need large number of poles (exponential in *d*)




diff
good prj.
bad prj.


no-go area
good prj.