MACHINE LEARNING AND APPLICATIONS GROUP

Non-linear Dimensionality Reduction and Embedding

Miloš Jovanović

Dimensionality reduction

- Unsupervised (?)
- Why reduce dimensions?
  - Curse of Dimensionality (who is hurt by CoD?)
  - Regularization
  - Feature Extraction - expressiveness   (linear?)
  - Increse efficiency (memory and speed)
  - Visualizations
  - Noise reduction
- Reduce so to preserve:
  - variance?   structure?   distances?   neighborhood?
- Preprocessing step (for prediction, information retrieval, vizualisation, …)   => evaluation!

Dim. reduction technique is ?

- PCA
- SVD
- Matrix Factorization

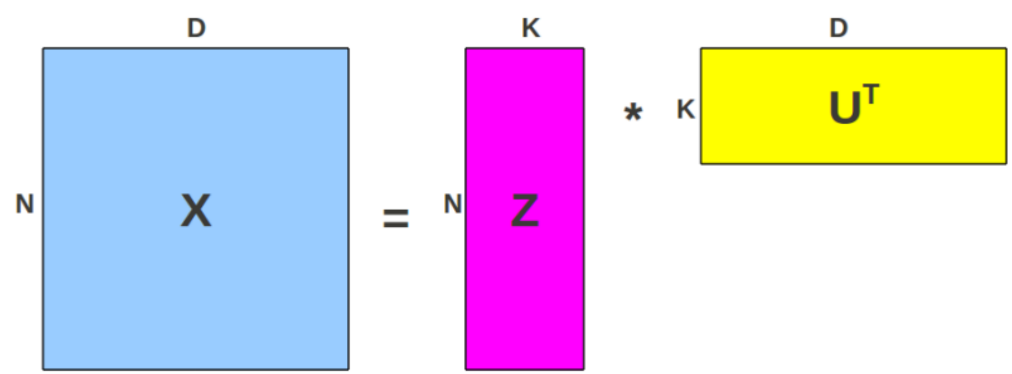
Linear

$$Z = XU$$

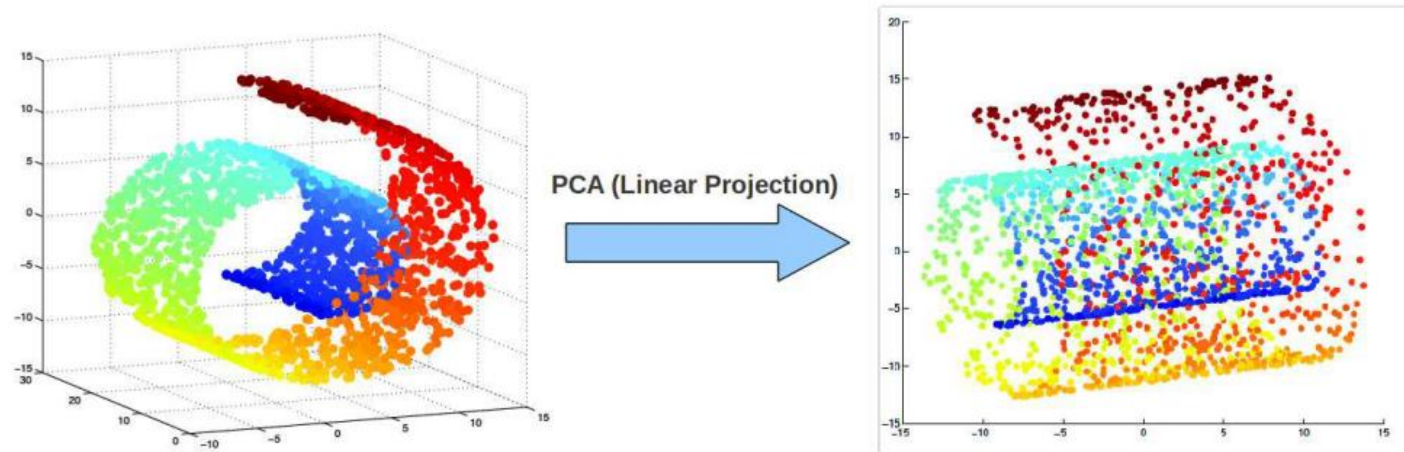
$$UU^\top = I$$

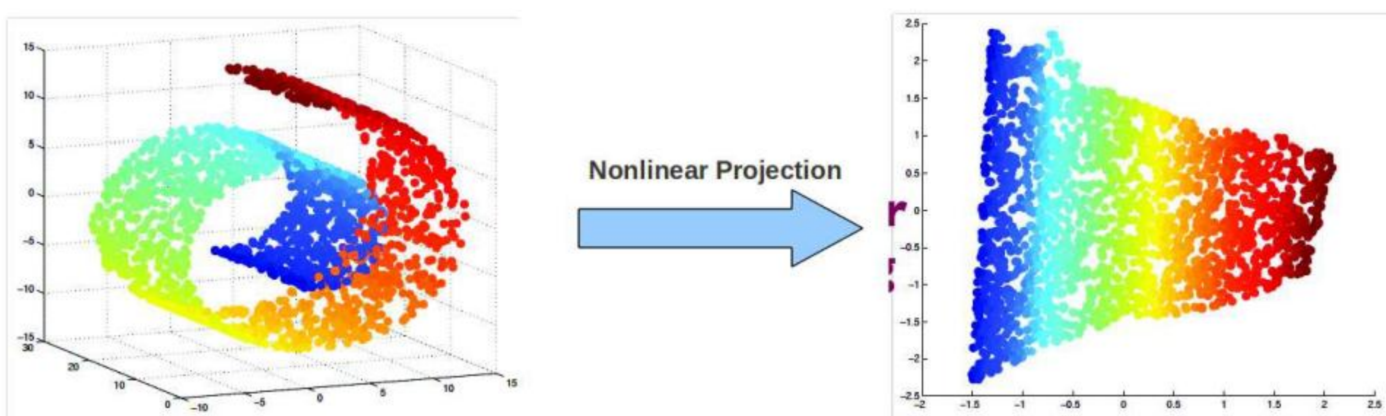
$$X = ZU^\top$$

- Very powerful!
  - … and fast!



Manifolds





Nonlinear PCA

- PCA, based on covariance (centered X):

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top \qquad \sum_n \mathbf{x}_n = \mathbf{0}$$

nonlinear transformation $\phi(\mathbf{x})$

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^\top$$

- Solve: $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$

$$\sum_n \phi(\mathbf{x}_n) = \mathbf{0} \qquad \tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N}\sum_{l=1}^{N} \phi(\mathbf{x}_l)$$

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$$

MultiDimensional Scaling (MDS)

- Usual distance calculation:
  - given points on a map (with coordinates), calculate distances
- MDS:
  - given distances, calculate cords
  - gradient descend, or eigen => dual PCA

$$\min_{Y} \sum_{i=1}^{t} \sum_{i=1}^{t} (\mathtt{d}_{ij}^{(X)} - \mathtt{d}_{ij}^{(Y)})^2$$

$$\text{where } \mathtt{d}_{ij}^{(X)} = ||x_i - x_j||^2 \text{ and } \mathtt{d}_{ij}^{(Y)} = ||y_i - y_j||^2$$

Sammon mapping

- more importance on small distances
- non-linear, non-convex
- circular embeddings with uniform density

$$Cost = \sum_{ij} \left(\frac{\| \mathbf{x}_i \overset{\text{high-D distance}}{-} \mathbf{x}_j \| - \| \mathbf{y}_i \overset{\text{low-D distance}}{-} \mathbf{y}_j \|}{\| \mathbf{x}_i - \mathbf{x}_j \|} \right)^2$$

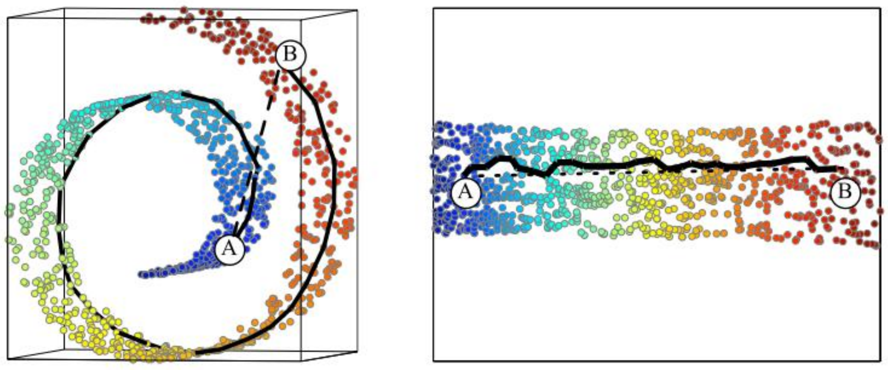
Isomap

- Graph-based
  - k-nearest neighbor graph, Euclidean weights $\mathbf{D}$
  - pairwise geodesic distances – Dijkstra, Floyd
- "local MDS without local optima"

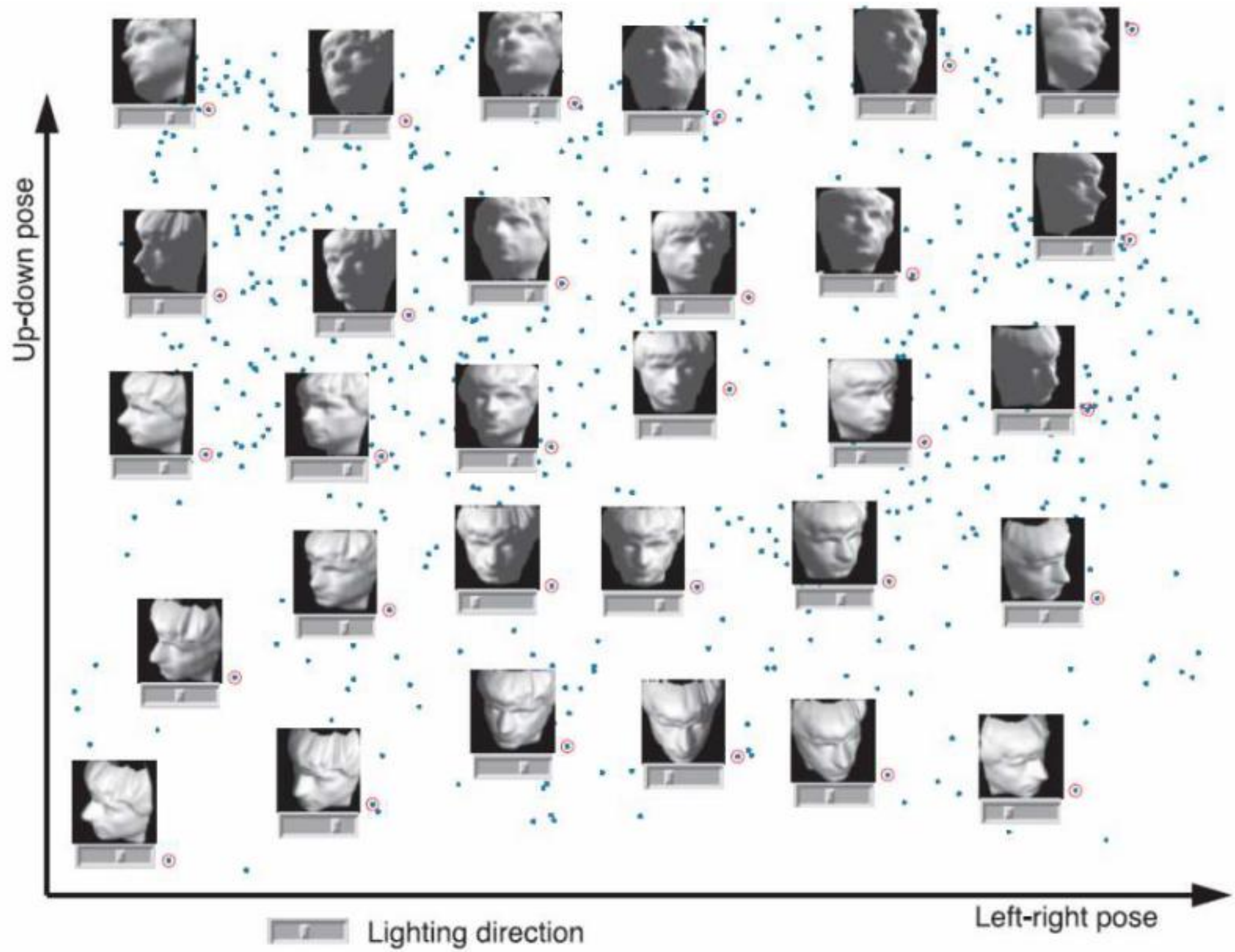
$$\mathbf{G} = -\frac{1}{2}\mathbf{HDH} \qquad \mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$$

- eigen: $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ $\{\lambda_1, \ldots, \lambda_N\}$

$$z_{ik} = \sqrt{\lambda_k} v_{ki}$$



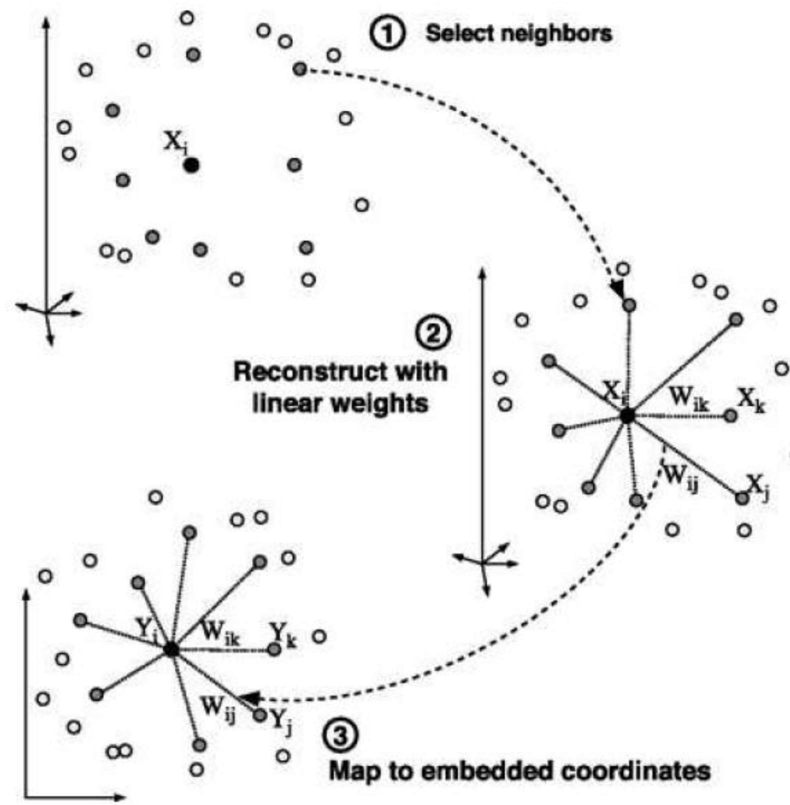
Isomap



Local Linear Embedding (LLE)

- Preserve neighborhood structure
- Assumption: manifold locally linear
  - locally linear, globally non-linear
  - local mapping efficient

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}} W_{ij} \mathbf{x}_j$$



Local Linear Embedding (LLE)

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}} W_{ij} \mathbf{x}_j$$

- Problem1:
  - For each i learn W independently
  - W quadratic programming – efficient

$$W = \arg\min_{W} \sum_{i=1}^{N} ||\mathbf{x}_i - \sum_{j \in \mathcal{N}_i} W_{ij} \mathbf{x}_j||^2$$

$$s.t. \forall i \quad \sum_{j} W_{ij} = 1$$

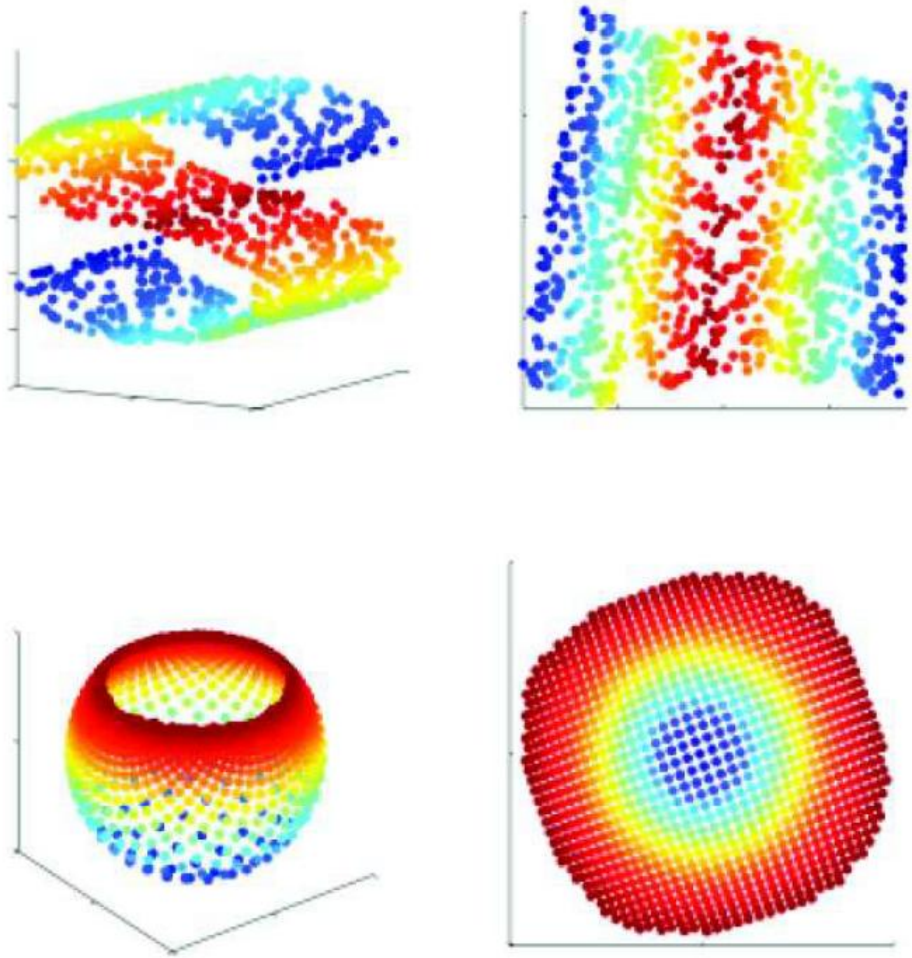
Local Linear Embedding (LLE)

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}} W_{ij} \mathbf{x}_j$$

- Problem2:
  - Z a sparse eigenvector problem
  - Solution invariant to global translation, rotation and reflection
  - choose bottom K non-zero eigenvectors
    - can be calculated iteratively without full matrix diagonalization

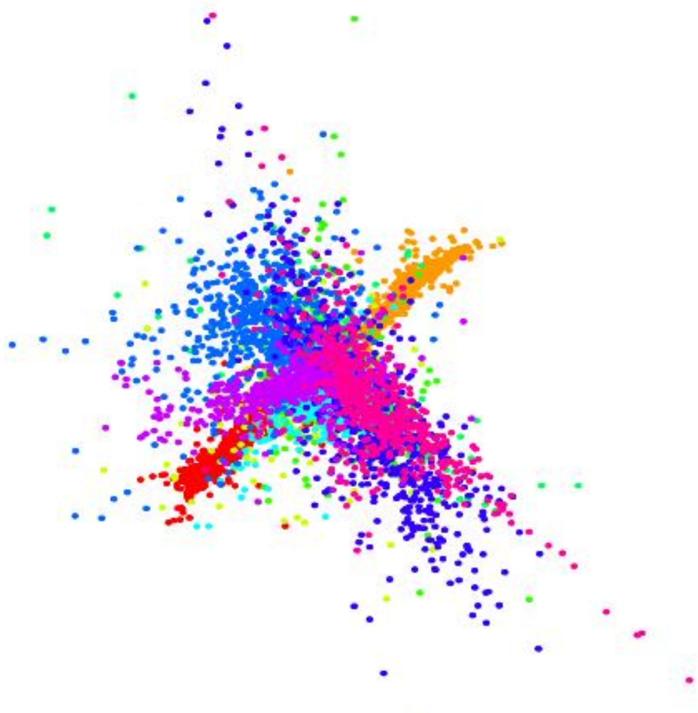
$$\mathbf{Z} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{N} ||\mathbf{z}_i - \sum_{j \in \mathcal{N}} W_{ij} \mathbf{z}_j||^2 \quad s.t. \forall i \quad \sum_{i=1}^{N} \mathbf{z}_i = 0, \quad \frac{1}{N} \mathbf{Z}\mathbf{Z}^\top = \mathbf{I}$$

Local Linear Embedding (LLE)



Local Linear Embedding (LLE)

- Problem:
  - no forcing to separate instances
  - only unit variance constraing



Laplacian Eigenmaps

- Very similar to LLE:
  - identify the nearest neighbor **graph**
  - define the edge weigths: $W_{ij} = e^{-(||\mathbf{x}_i - \mathbf{x}_j||)/s}$

$$\min_Y \sum_{i=1}^{t} \sum_{j=1}^{t} (\mathbf{y}_i - \mathbf{y}_j)^2 W_{ij}$$

$$\min_Y \mathrm{Tr}(YLY^T)$$

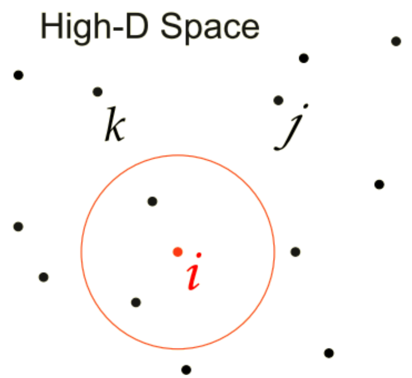
$$L = R - W \qquad R_{ii} = \sum_{j=1}^{t} W_{ij} \qquad YRY^T = I$$

  - compute the bottom eigenvectors of L

L – Graph Laplacian Y – Graph Spectra

SNE

- Probabilistic model (Stochastic Neighborhood Embedding)



probability of picking j
given that you start at i

$$p_{j|i} = \frac{e^{-d_{ij}^2 / 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 / 2\sigma_i^2}}$$

$$Cost = KL(P \| Q) | = \sum_{i<j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

extension stiffness

$$\frac{\partial KL(P \| Q)}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_i - \mathbf{y}_j)(p_{ij} - q_{ij})$$

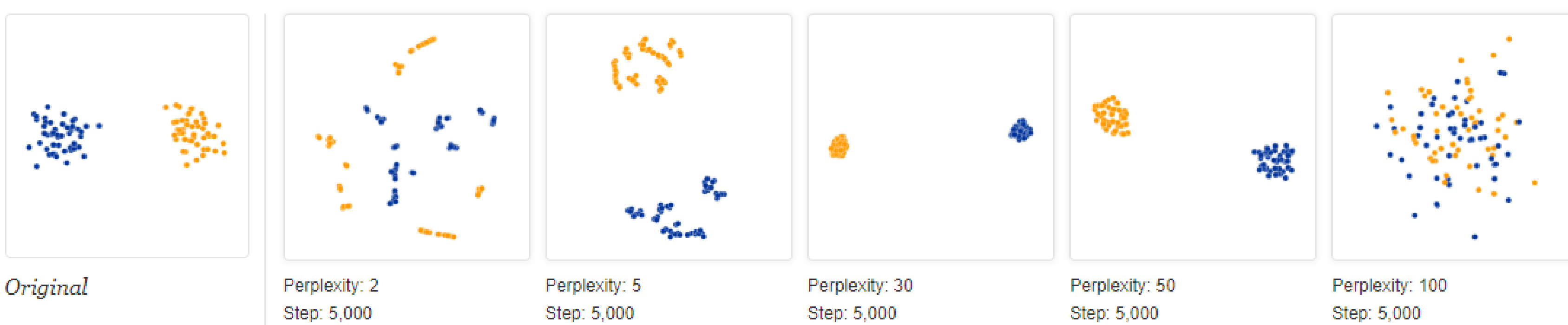
t-SNE

- Gaussians at many spatial scales
  - infinite gaussian mixture (same mean)

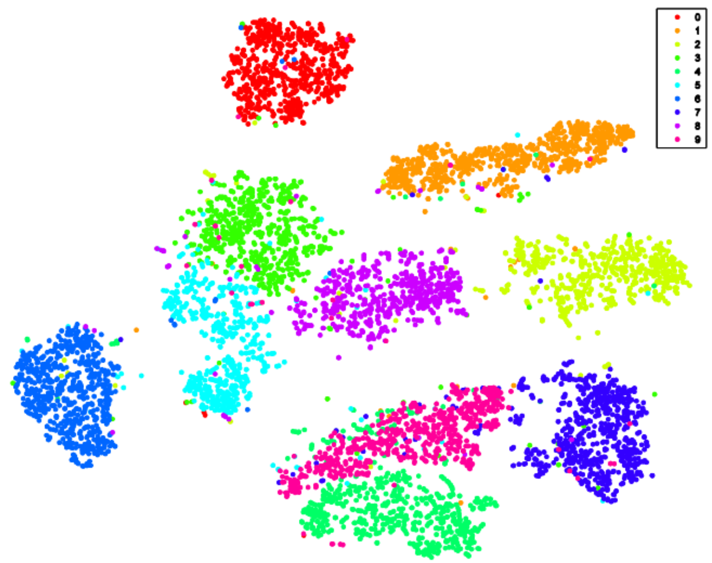
    => t-distribution
- Tricks for optimization:
  - add gaussian noise to y after update
  - annealing and momentum
  - adaptive global step-size
  - dimension decay

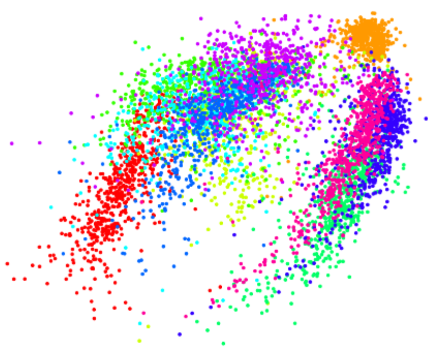
t-SNE

- Demo!   Toolbox!
- Hyperparameters really matter
- Cluster sizes in a t-SNE plot mean nothing
- Distances between clusters might not mean anything
- Random noise doesn't always look random

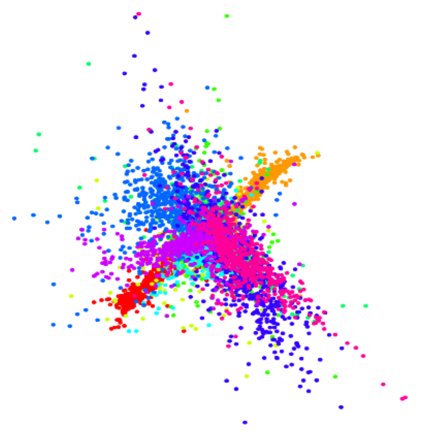


t-SNE on MNIST





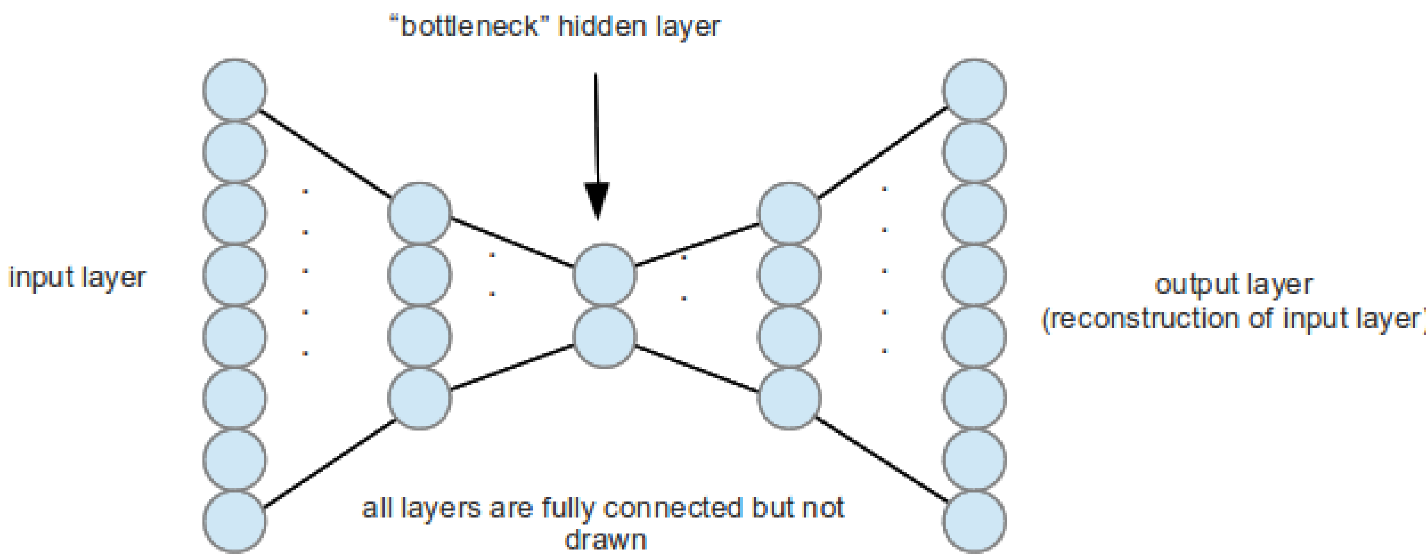
Isomap



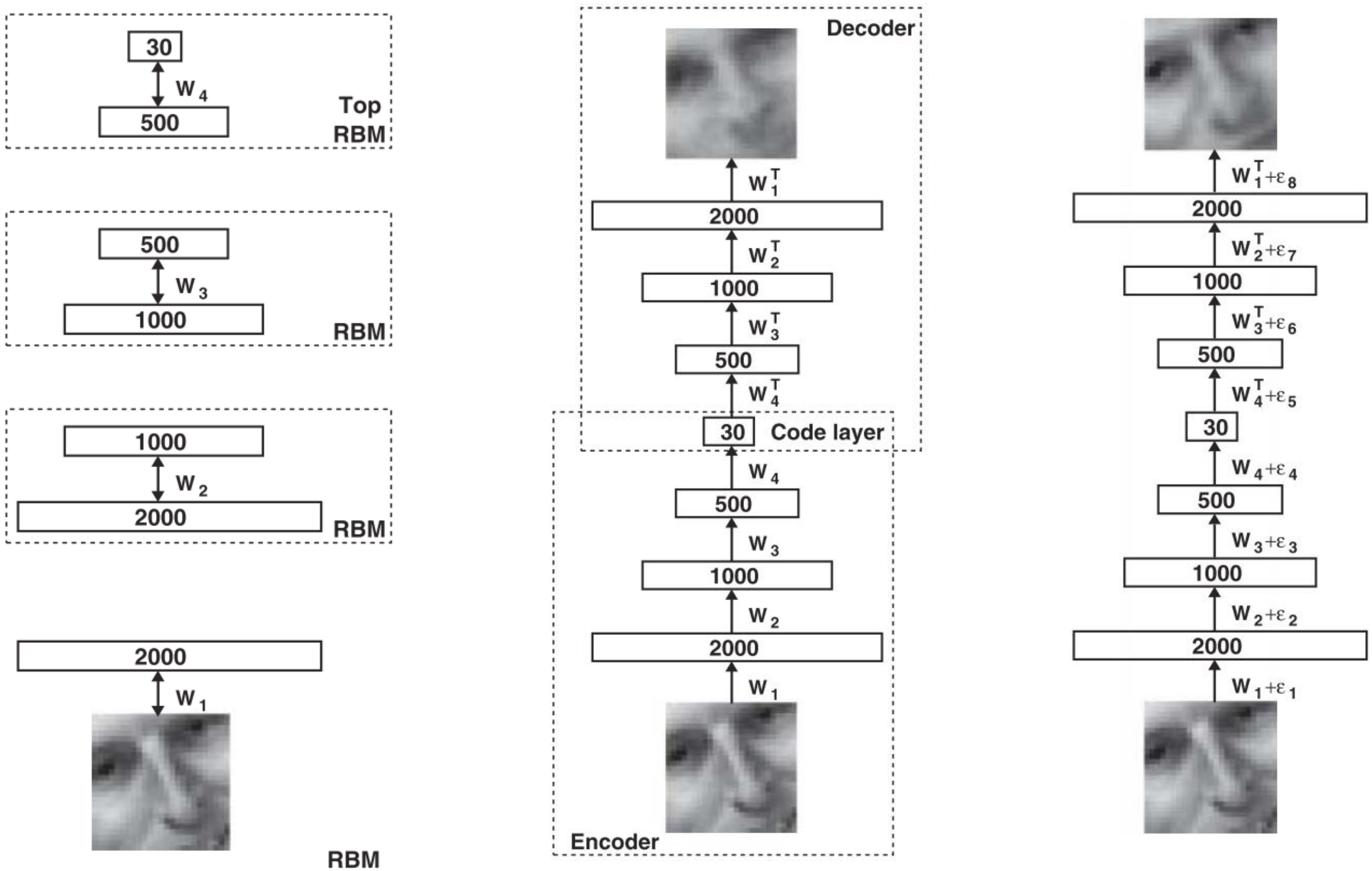
Locally Linear Embedding

Autoencoders

- Deep neural networks



Autoencoders



Autoencoders

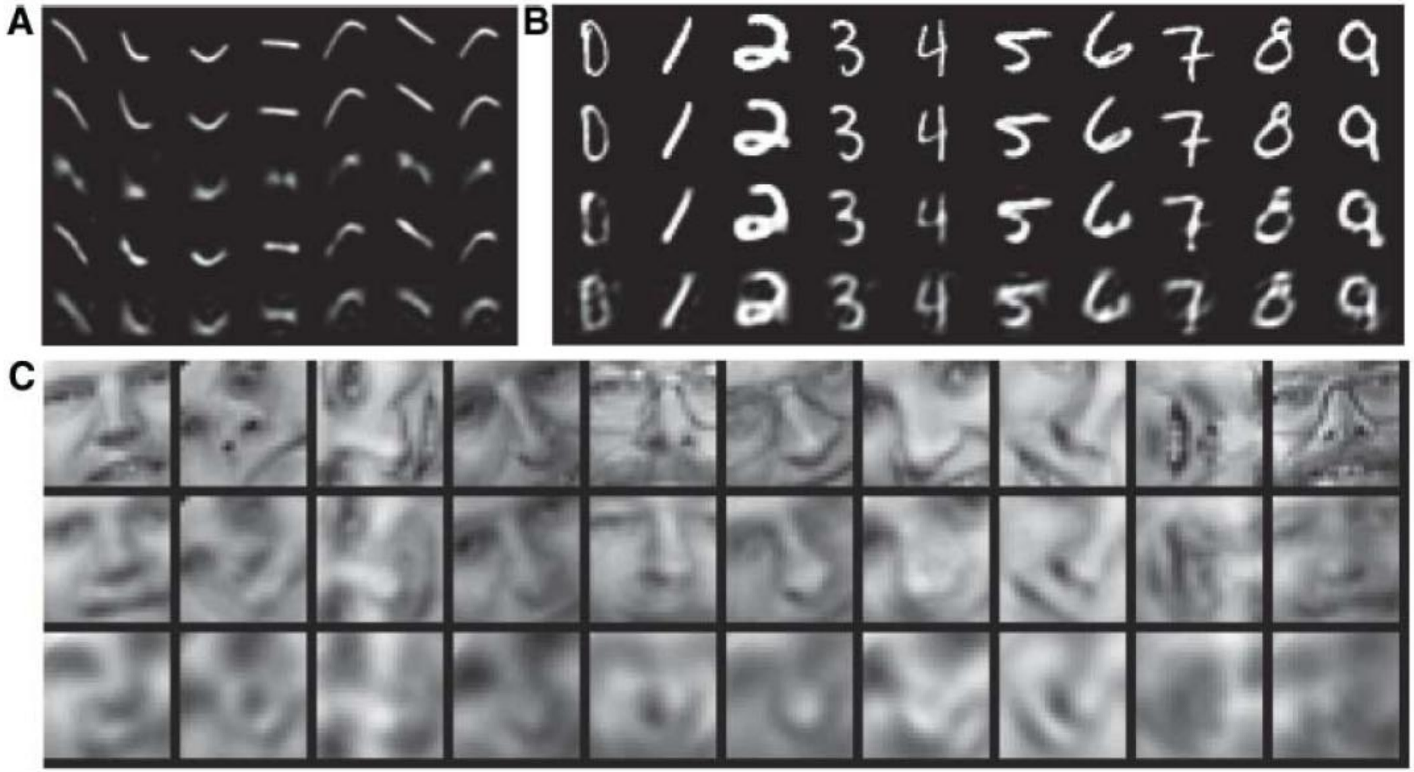
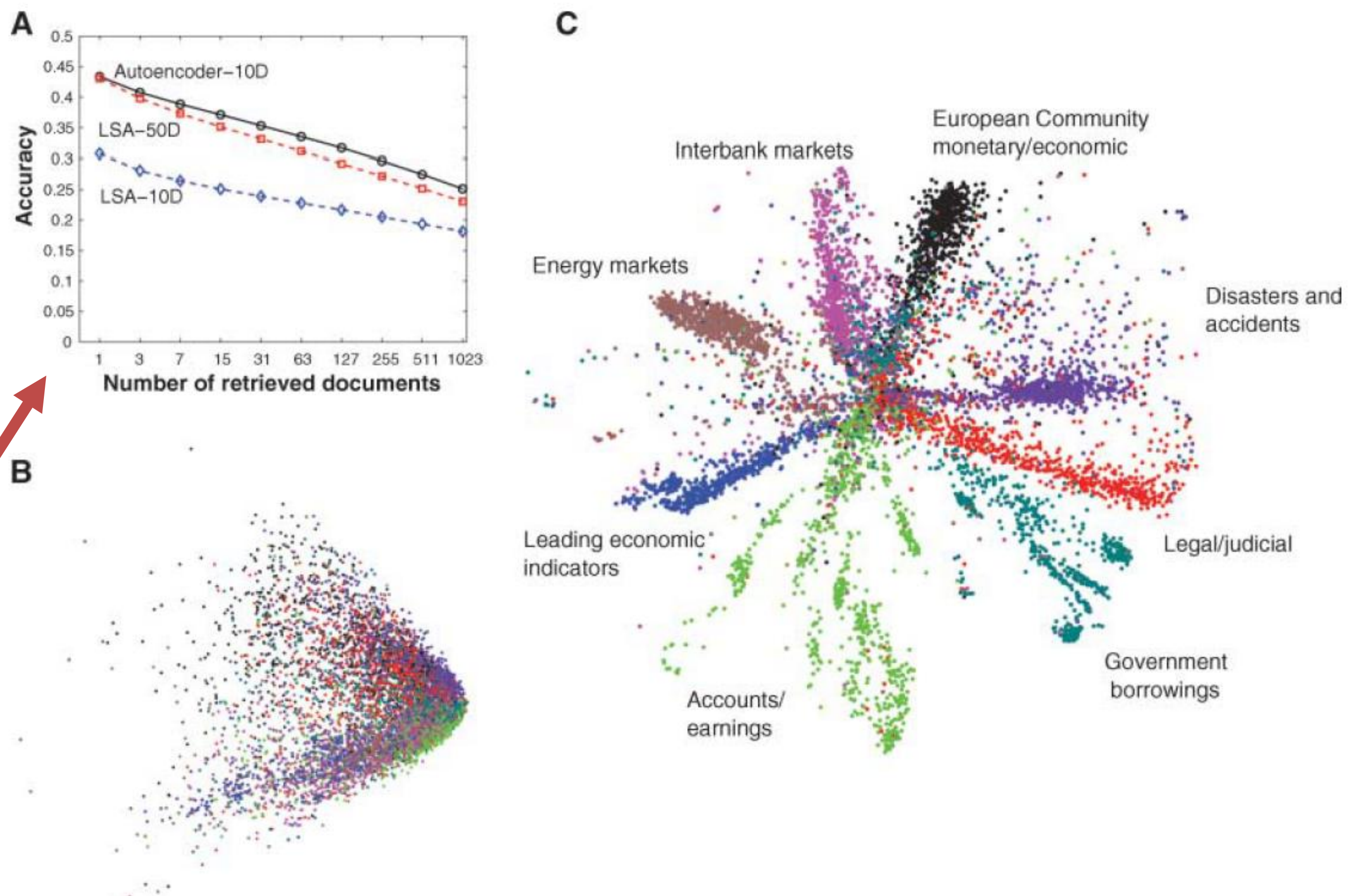


Fig. 2. (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error per image for the last four rows is 1.44, 7.64, 2.45, 5.90. (**B**) Top to bottom: A random test image from each class; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional logistic PCA and standard PCA. The average squared errors for the last three rows are 3.00, 8.01, and 13.87. (**C**) Top to bottom: Random samples from the test data set; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional PCA. The average squared errors are 126 and 135.

Autoencoders

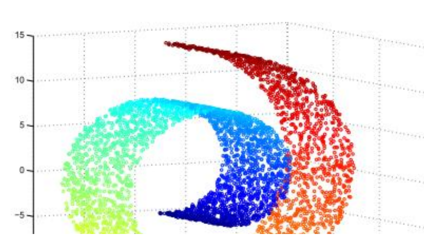
Fig. 4. (**A**) The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. (**B**) The codes produced by two-dimensional LSA. (**C**) The codes produced by a 2000-500-250-125-2 autoencoder.

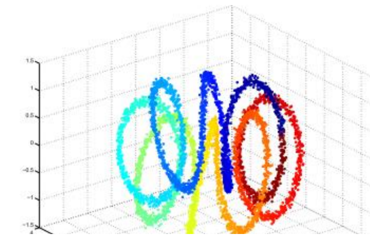


Evaluation!

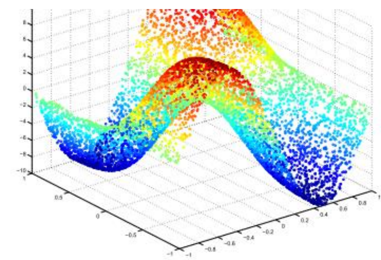
Comparisson (1-NN)

- Artificial datasets:

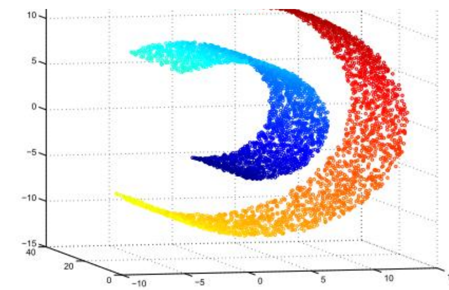




Dataset (d)	*None*	*PCA*	*Isomap*	*KPCA*	*MVU*	*DM*	*LLE*	*LEM*	*HLLE*	*LTSA*	*Sammon*	*Autoenc.*	*LLC*	*MC*
Swiss roll (2D)	3.68%	29.76%	**3.40%**	30.24%	4.12%	33.50%	3.74%	22.06%	3.56%	3.90%	22.34%	49.00%	26.72%	22.66%
Helix (1D)	1.24%	35.50%	13.18%	38.04%	7.48%	35.44%	32.32%	15.24%	52.22%	**0.92%**	52.22%	52.22%	27.44%	25.94%
Twin peaks (2D)	0.40%	0.26%	0.22%	**0.12%**	0.56%	0.26%	0.94%	0.88%	0.14%	0.18%	0.32%	49.06%	11.04%	0.30%
Broken Swiss (2D)	2.14%	25.96%	14.48%	32.06%	32.06%	58.26%	36.94%	10.66%	**6.48%**	15.86%	27.40%	87.86%	37.06%	32.24%
HD (5D)	24.19%	22.18%	23.26%	27.46%	25.38%	23.14%	**20.74%**	24.70%	50.02%	42.62%	20.70%	49.18%	34.14%	21.34%



(c) Twinpeaks dataset.

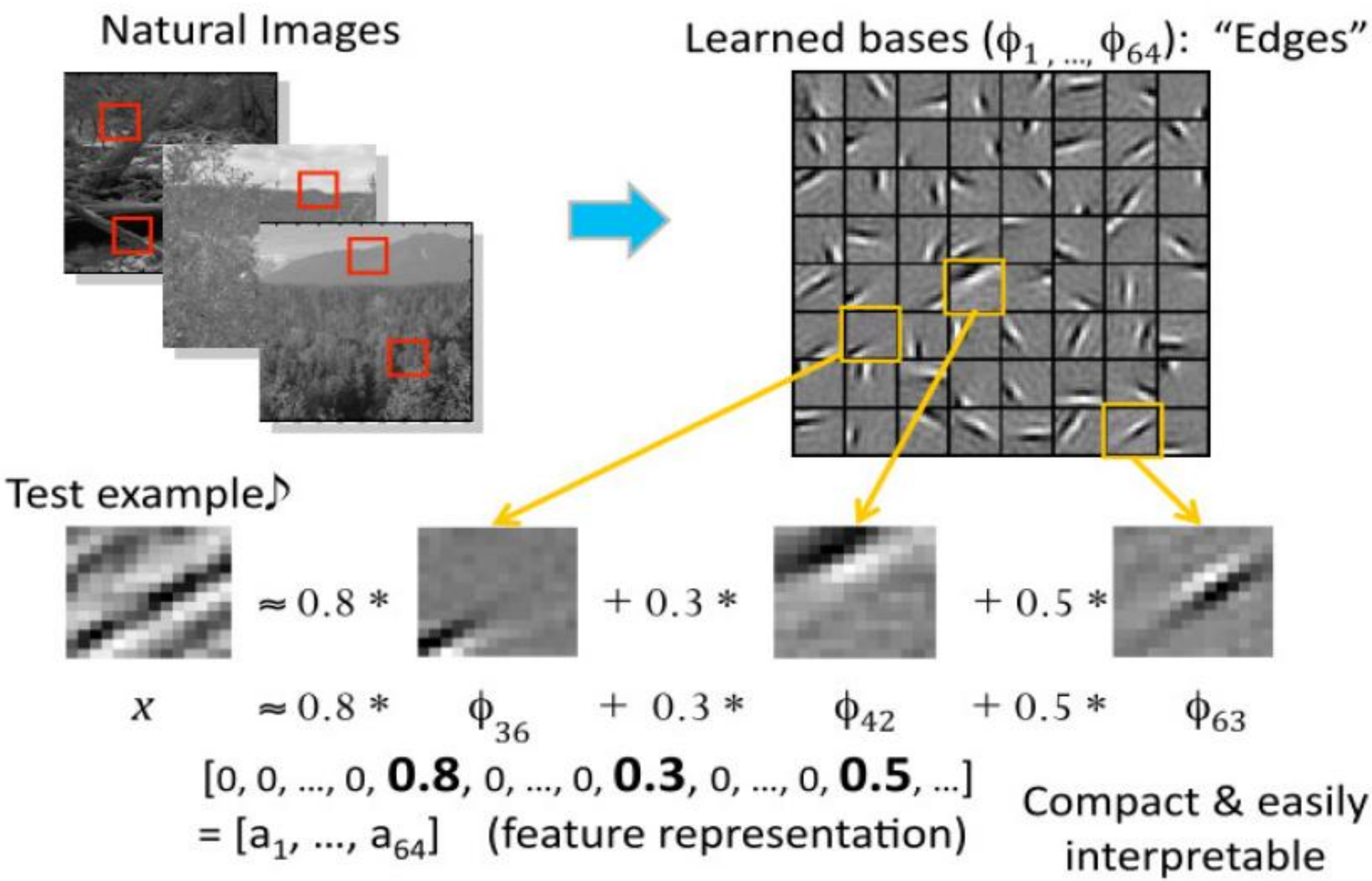


(d) Broken Swiss roll dataset.

- Natural datasets

Dataset (d)	*None*	*PCA*	*Isomap*	*KPCA*	*MVU*	*DM*	*LLE*	*LEM*	*HLLE*	*LTSA*	*Sammon*	*Autoenc.*	*LLC*	*MC*
MNIST (20D)	5.11%	**6.74%**	12.64%	13.86%	13.58%	25.00%	10.02%	11.30%	91.66%	90.32%	6.90%	7.18%	16.12%	14.84%
COIL20 (5D)	0.14%	3.82%	15.69%	7.78%	25.14%	11.18%	22.29%	95.00%	50.35%	4.17%	**0.83%**	51.11%	4.31%	27.36%
ORL (8D)	2.50%	4.75%	27.50%	6.25%	24.25%	90.00%	11.00%	97.50%	56.00%	12.75%	2.75%	**6.25%**	11.25%	22.50%
NiSIS (15D)	8.24%	**7.95%**	13.36%	9.55%	15.67%	48.98%	15.48%	47.59%	48.98%	24.68%	48.98%	9.22%	26.86%	18.91%
HIVA (15D)	4.63%	5.05%	4.92%	5.07%	4.94%	5.46%	4.97%	4.81%	**3.51%**	**3.51%**	**3.51%**	5.12%	**3.51%**	4.79%

Sparse Coding

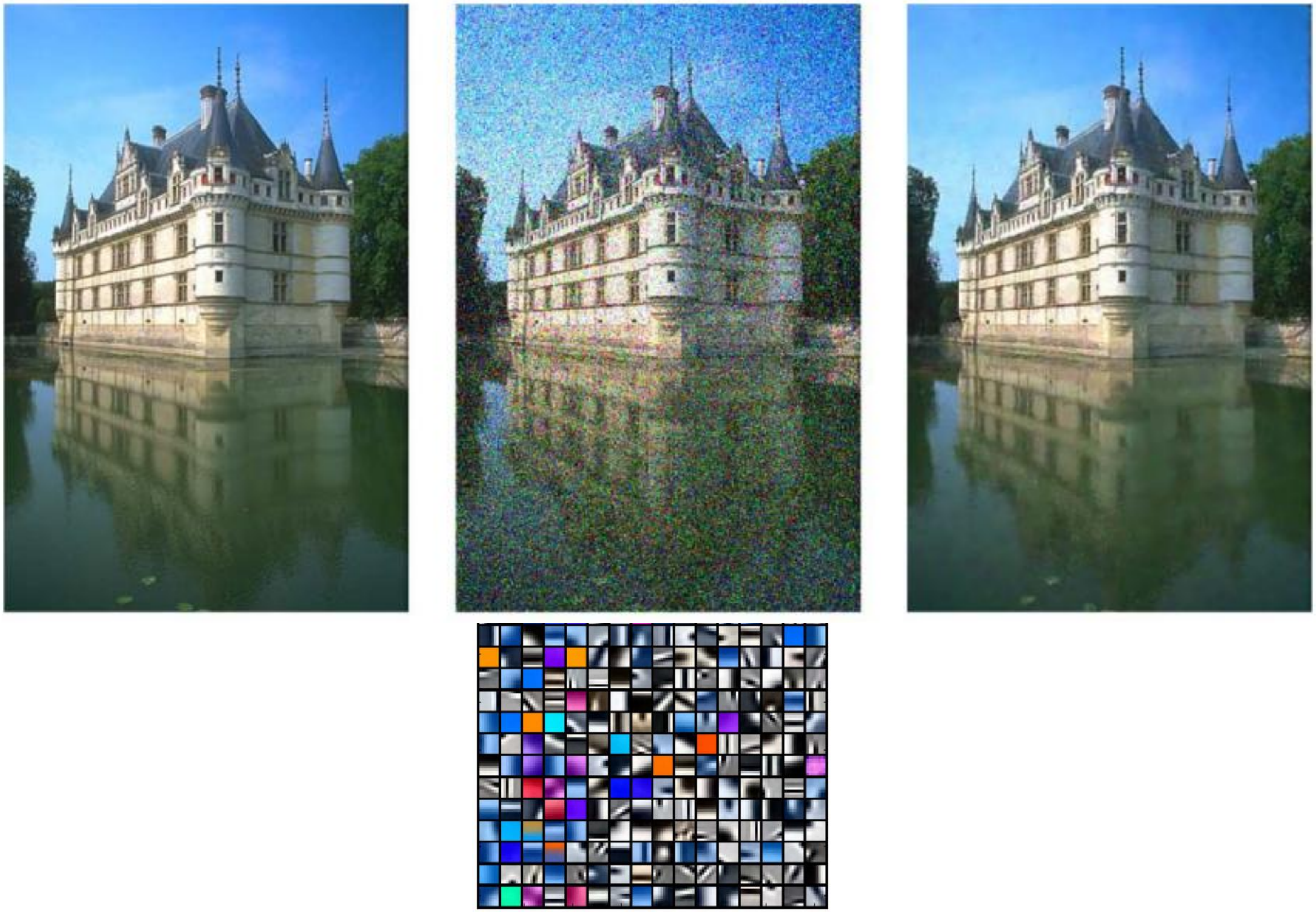


Sparse Coding

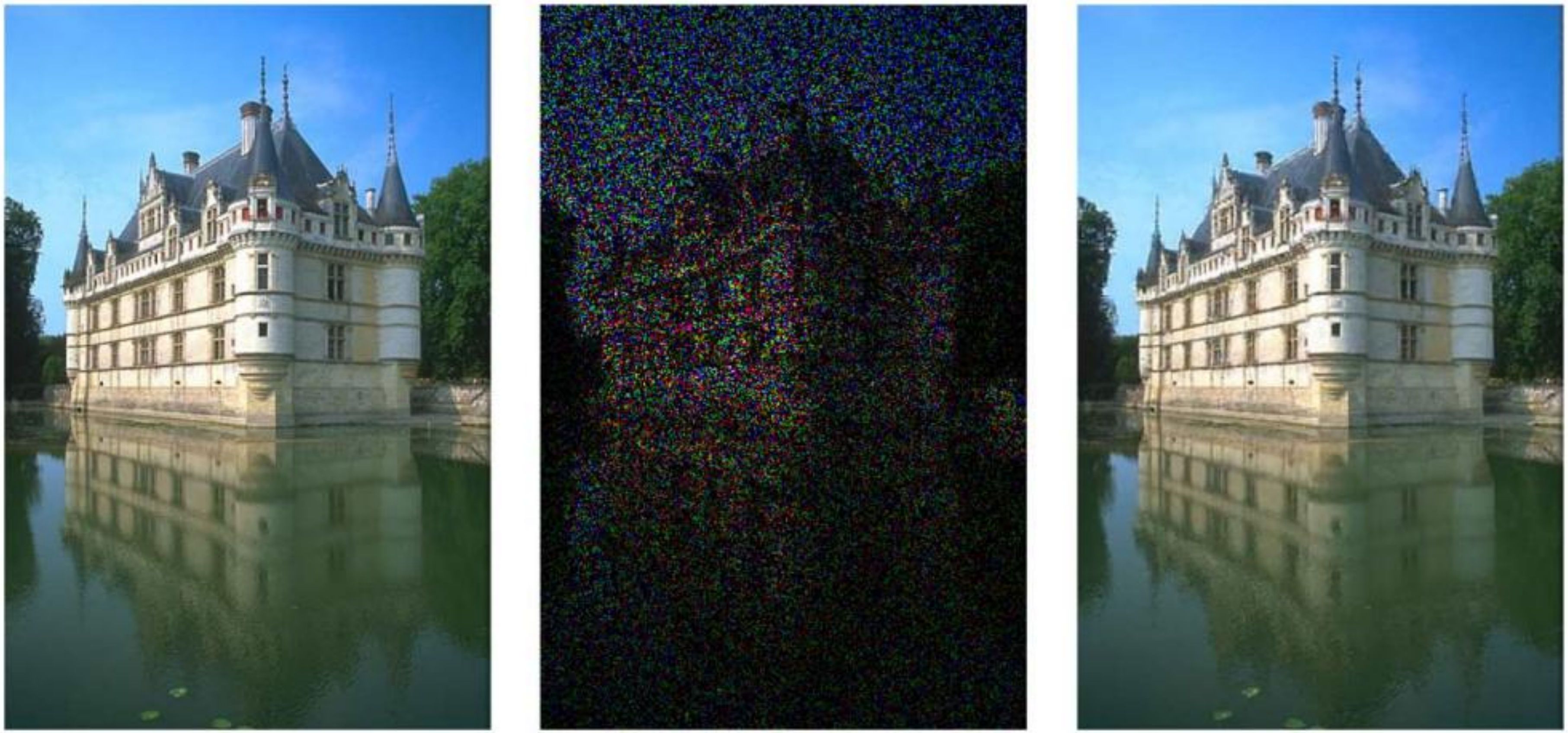
$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \underbrace{\frac{1}{2}||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2}_{\text{data fitting term}} + \underbrace{\lambda\psi(\boldsymbol{\alpha})}_{\text{sparsity-inducing regularization}}$$

- Alternate optimization over **D** and α
- Matching Pursuit, Orthogonal Matching Pursuit, …
- Dictionary learning

Color Image Denoising

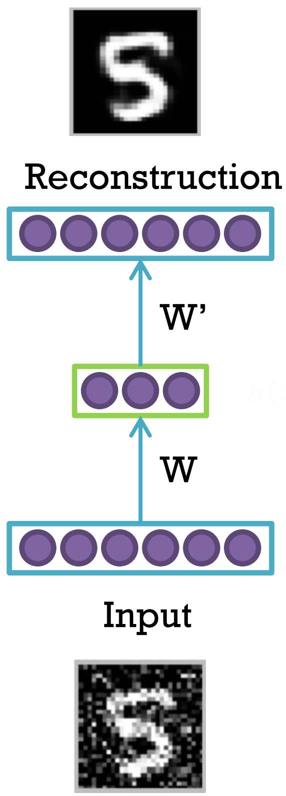


Color Image Denoising

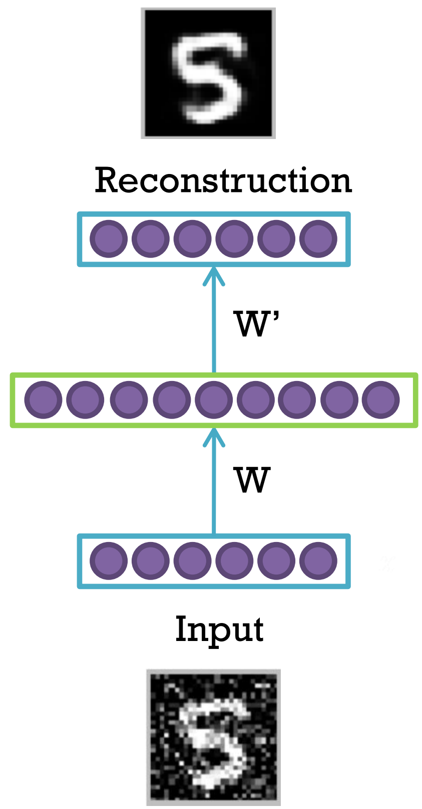


Color Image Denoising

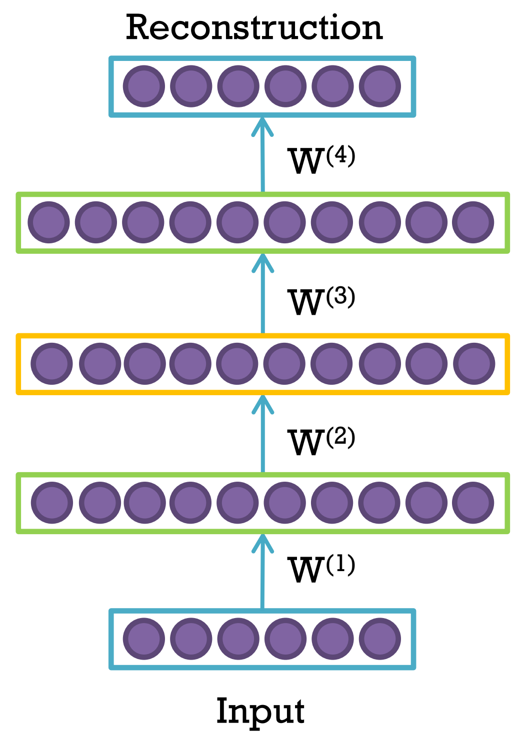
Stacked Sparse Auto-encoders



Denoising Autoencoder

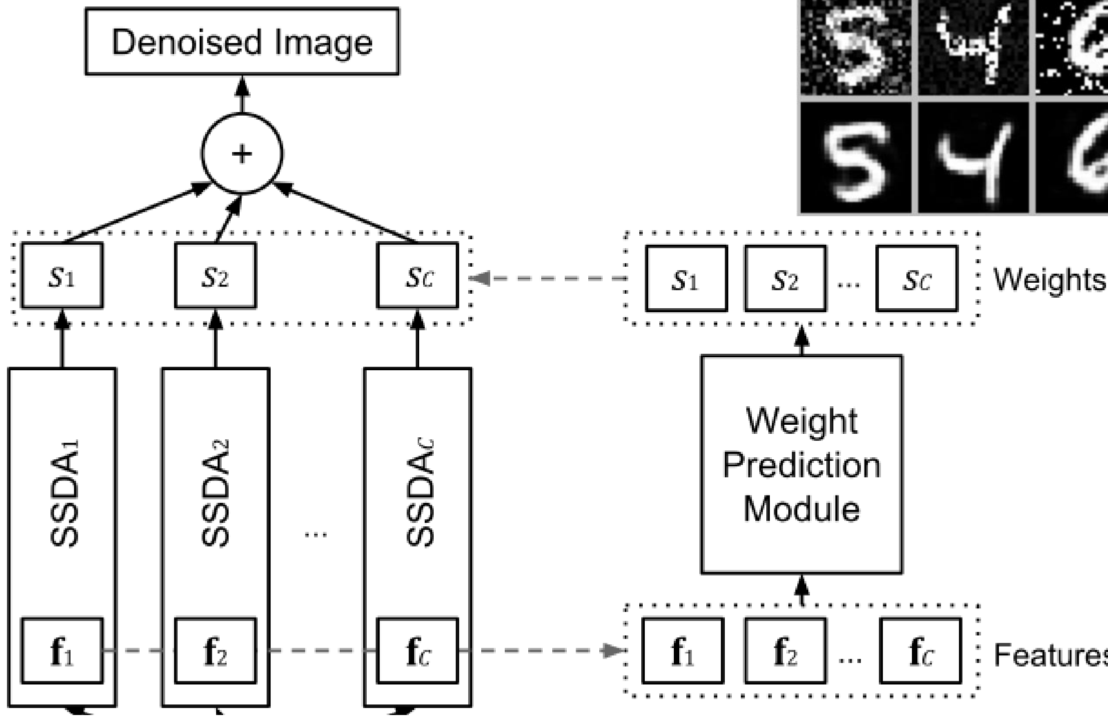


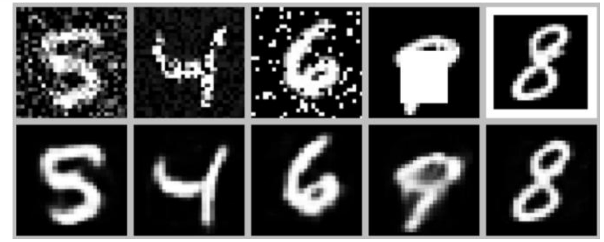
Sparse Denoising Autoencoder



Stacked Sparse Denoising Autoencoder

Adaptive Multi-Column SSDA





Method / Noise Type	Clean	Gaussian	S & P	Speckle	Block	Border	Average
No denoising	**1.09%**	29.17%	18.63%	8.11%	25.72%	90.05%	28.80%
Gaussian SSDA	2.13%	**1.52%**	2.44%	5.10%	20.03%	8.69%	6.65%
Salt & Pepper SSDA	1.94%	1.71%	2.38%	4.78%	19.71%	2.16%	5.45%
Speckle SSDA	1.58%	5.86%	6.80%	**2.03%**	19.95%	7.36%	7.26%
Block SSDA	1.67%	5.92%	15.29%	7.64%	**5.15%**	1.81%	6.25%
Border SSDA	8.42%	19.87%	19.45%	13.89%	31.38%	**1.12%**	15.69%
AMC-SSDA	1.50%	**1.47%**	**2.22%**	**2.09%**	**5.18%**	**1.15%**	**2.27%**
Tang et al. [26]*	1.24%	-	-	-	19.09%	1.29%	-

Embedding

- Structure-preserving mapping
- Euclidean embedding (Euclidean space)
  - images
  - words
  - graphs
  - bipartite-categories (co-occurance)
- Allows to apply computational learning on symbols (objects)
- Even allow arithmetic!?
- Allow visualization: embedding + t-sne

Summary

- Why reduce dimensions?
  - Curse of Dimensionality (who is hurt by CoD?)
  - Regularization
  - Feature Extraction - expressiveness (linear?)
  - Increse efficiency (memory and speed)
  - Visualizations
  - Noise reduction
- Reduce so to preserve:
  - variance? structure? distances? neighborhood?
- Parametric VS Non-parametric Encoding (new instances)
- With or without Decoding

References

Bengio Y. **Learning Deep Architectures for AI**. Foundations and Trends in Machine Learning, Vol 2 No 1. 2009.

Burges CJC. **Dimension Reduction: a guided tour**. Foundations and Trends in Machine Learning, Vol. 2, No. 4, 2009

Ghodsi A. **Dimensionality Reduction: a short tutorial**. Department of Statistics and Actuarial Science, TR. 2006

Globerson A et al. **Euclidean Embedding of Co-occurence Data**. Journal of Machine Learning Research 8. 2007.

Hinton G. **Non-linear dimensionality reduction**, Course CSC 2535 materials, 2013

Hinton GE, Salakhutdinov RR. **Reducing the Dimensionality of Data with Neural Networks**. Science, vol 313. 2006.

Khoshneshin M, Street WN. **Collaborative Filtering via Euclidean Embedding**. RecSys 2010, Barcelona, Spain.

Mairal J, Elad M, Sapiro G. **Sparse Representation for Color Image Restoration**. IEEE Transactions on Image Processing, vol 17, no 1. 2008

Piyush Rai, **Nonlinear Dimensionality Reduction**, Course CS5350/6350 materials, 2011

Saul LK, et al. **Spectral Methods for Dimensionality Reduction**, Chapter in Semi-Supervised Learning by Chapelle O et al. 2006

Saul LK, Roweis ST. **An Introduction to Locally Linear Embedding**. 2010.

van der Maaten L, Postma E, Herik J. **Dimensionality Reduction: a comparative review**. Tilburg centre for Creative Computing, TR 2009-005, 2009.

Yuhong Guo, Topics In Computer Science: **Data Representation Learning**, Course CIS 5590 materials, 2014

Q&A

Thank you!