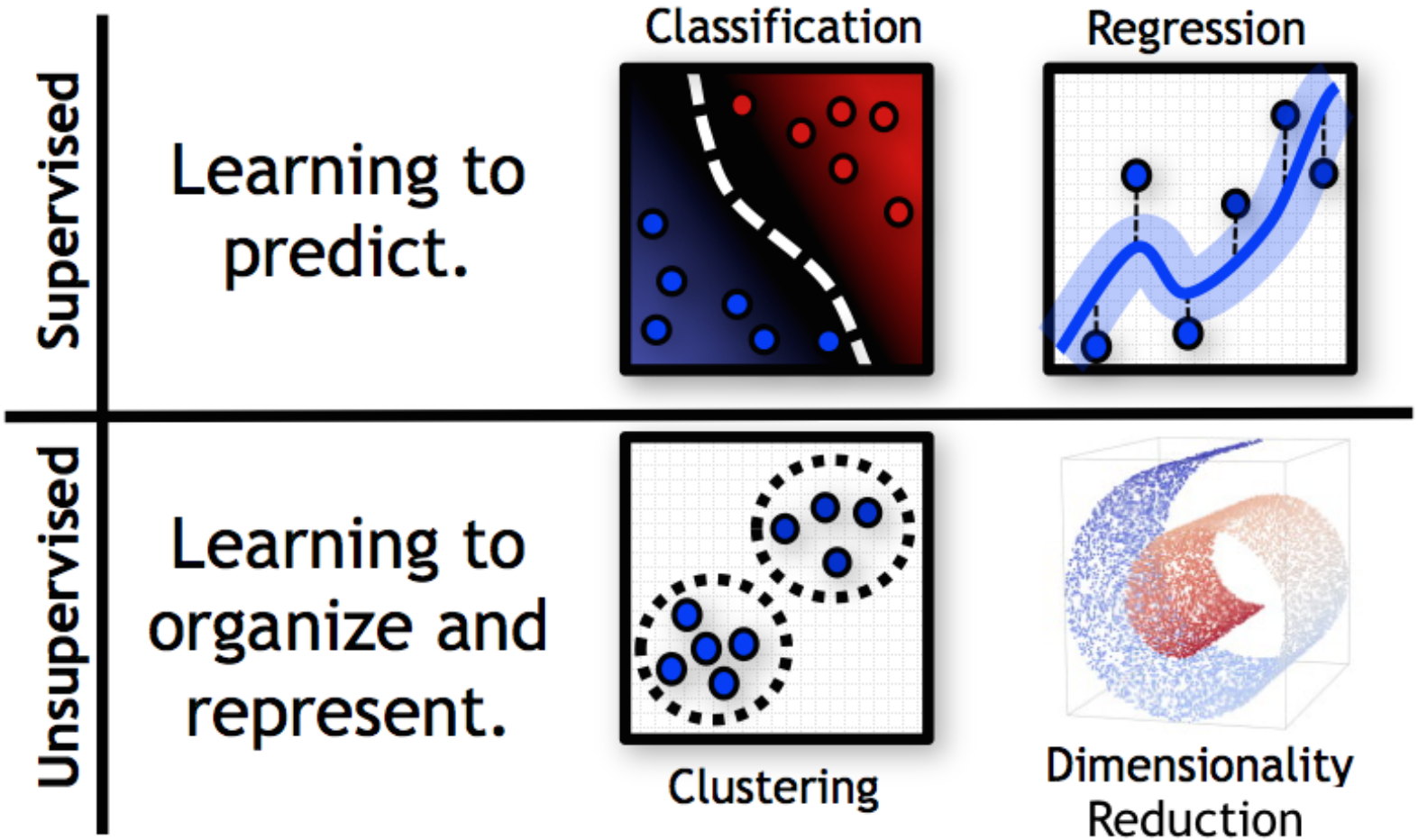
COMPSCI 589
Lecture 19: Graph-Based Dimensionality Reduction

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

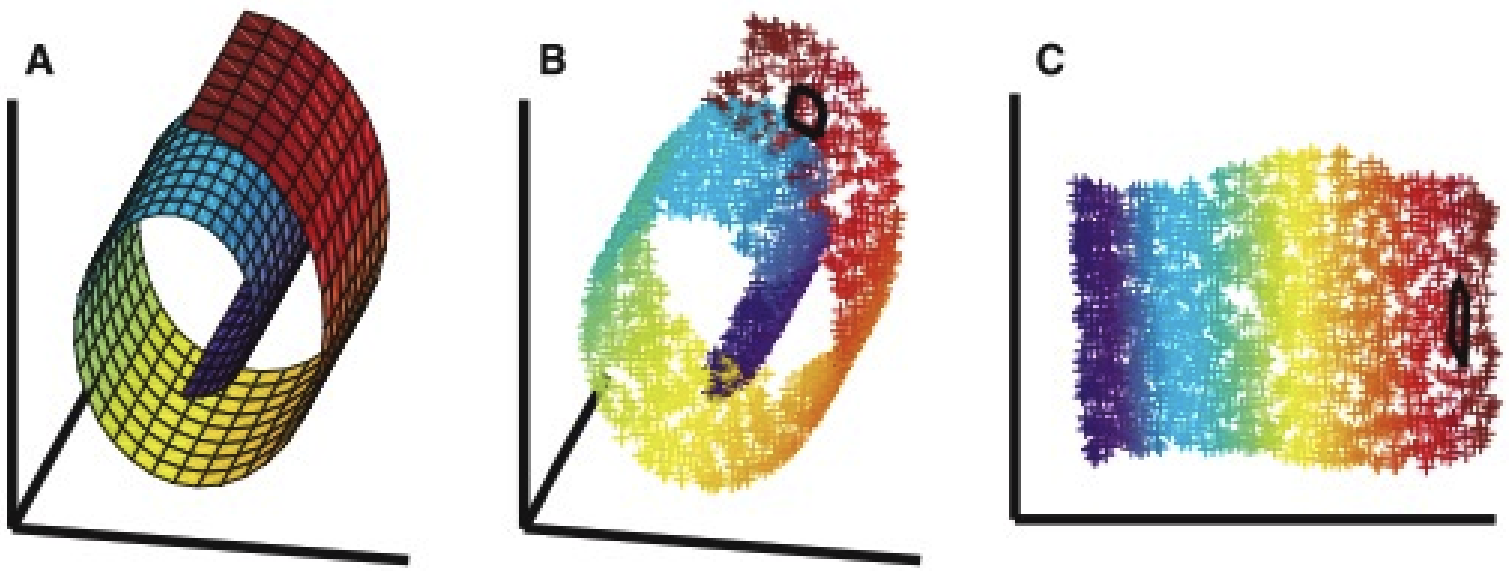
Machine Learning Tasks



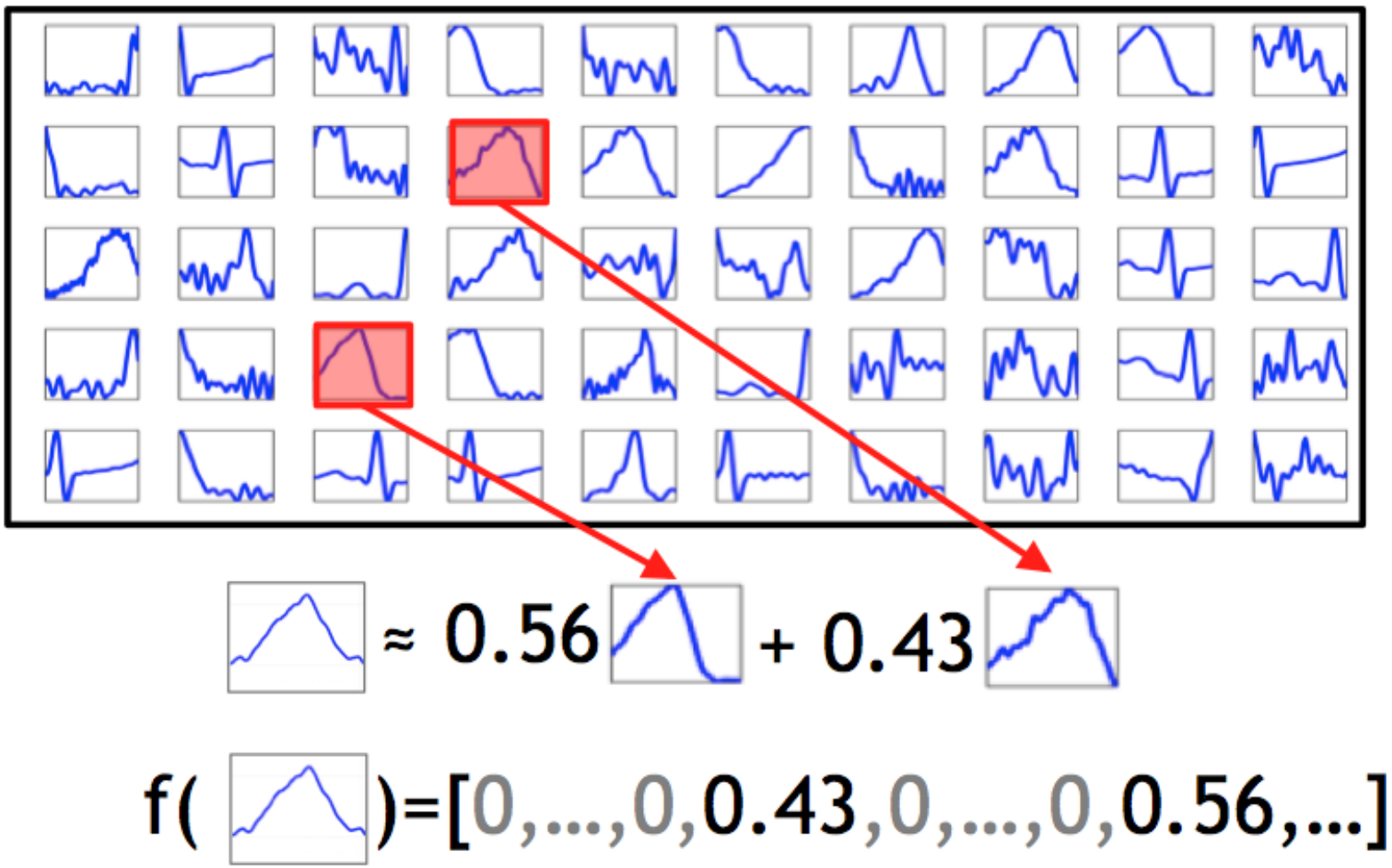
The Dimensionality Reduction Task

Definition: The Dimensionality Reduction Task

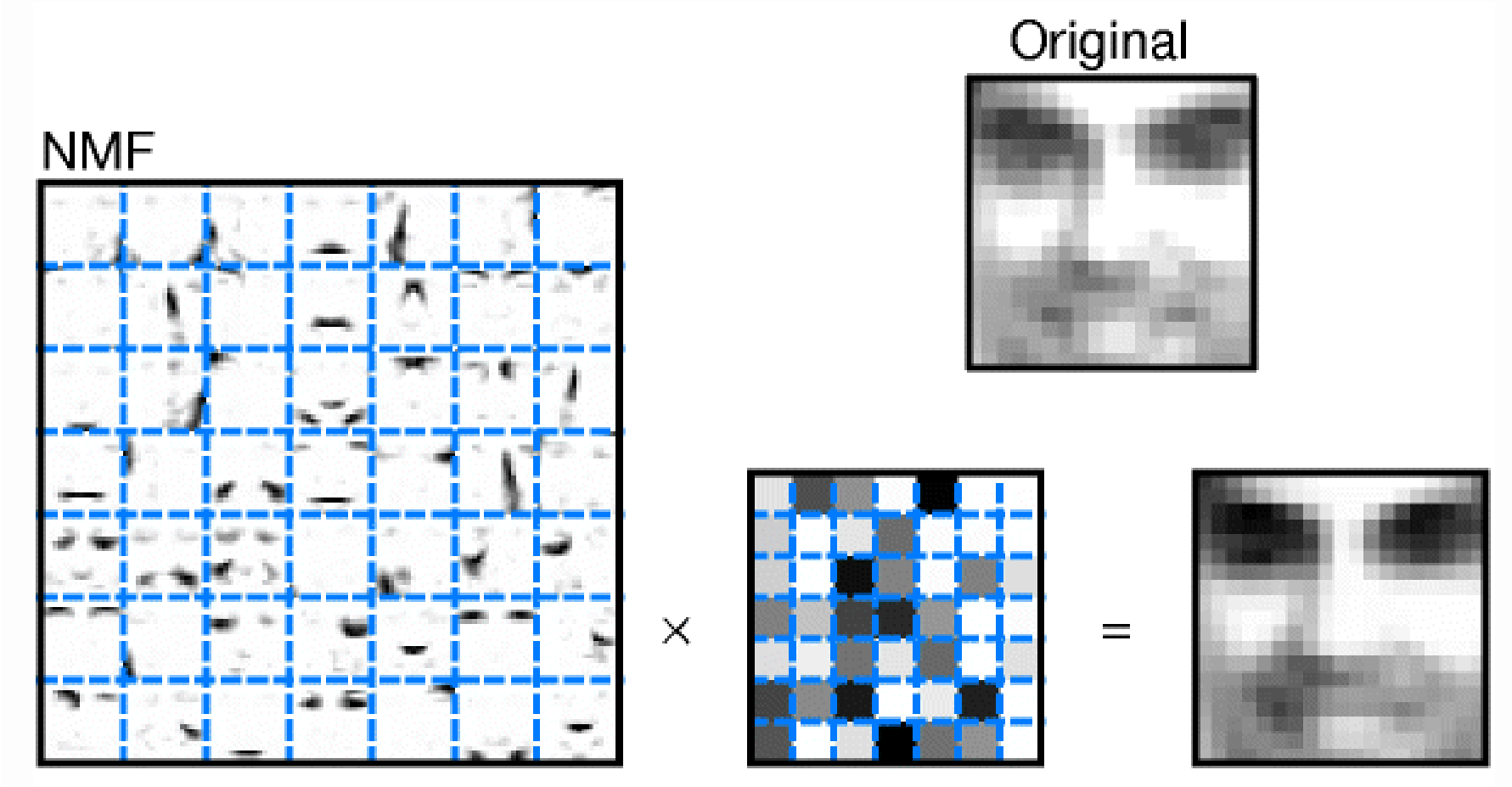
Given a collection of feature vectors $\mathbf{x}_i \in \mathbb{R}^D$, map the feature vectors into a lower dimensional space $\mathbf{z}_i \in \mathbb{R}^K$ where $K < D$ while preserving certain properties of the data.



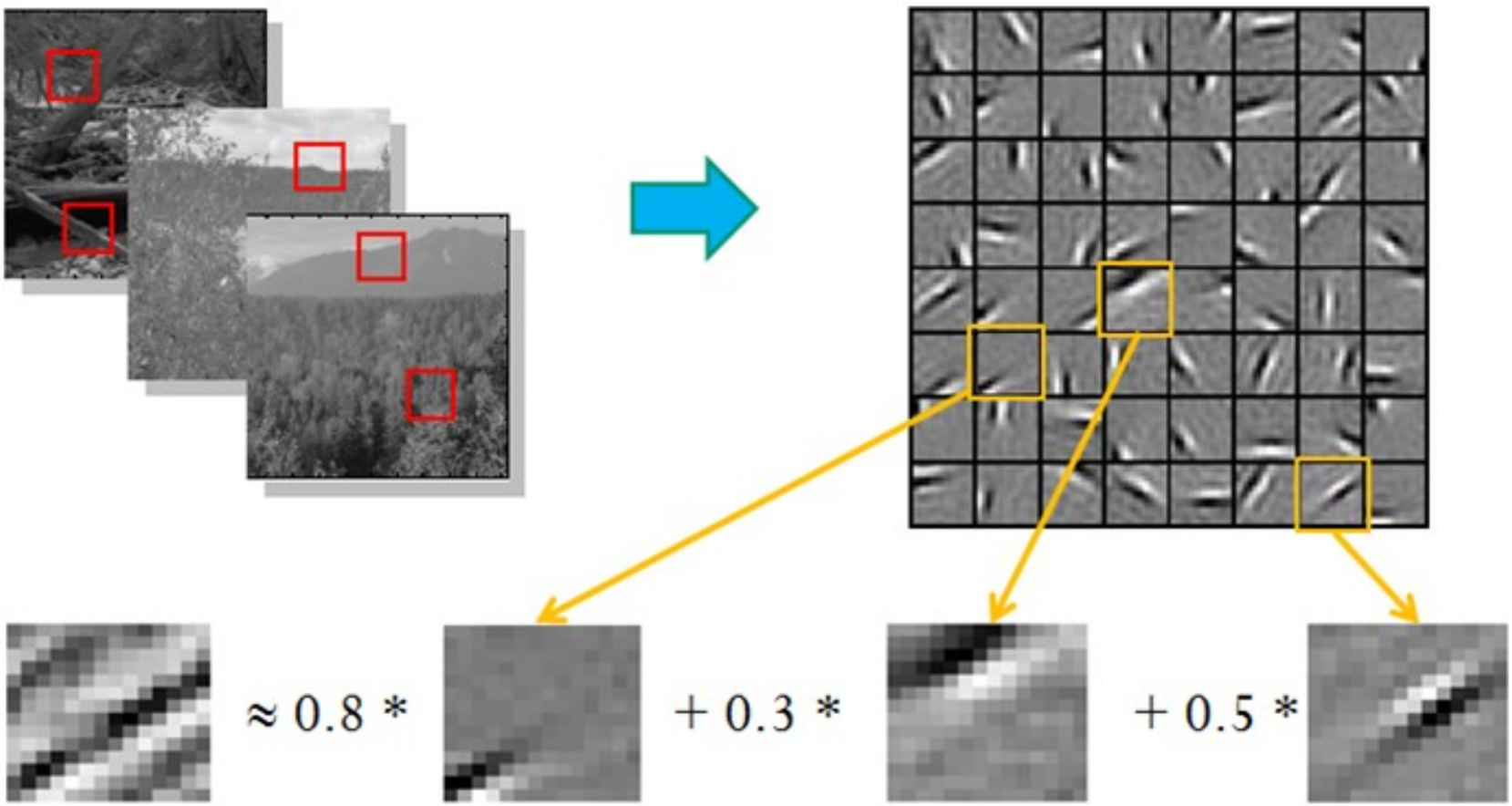
Example: Representing Time Series



Example: Learning Face Parts

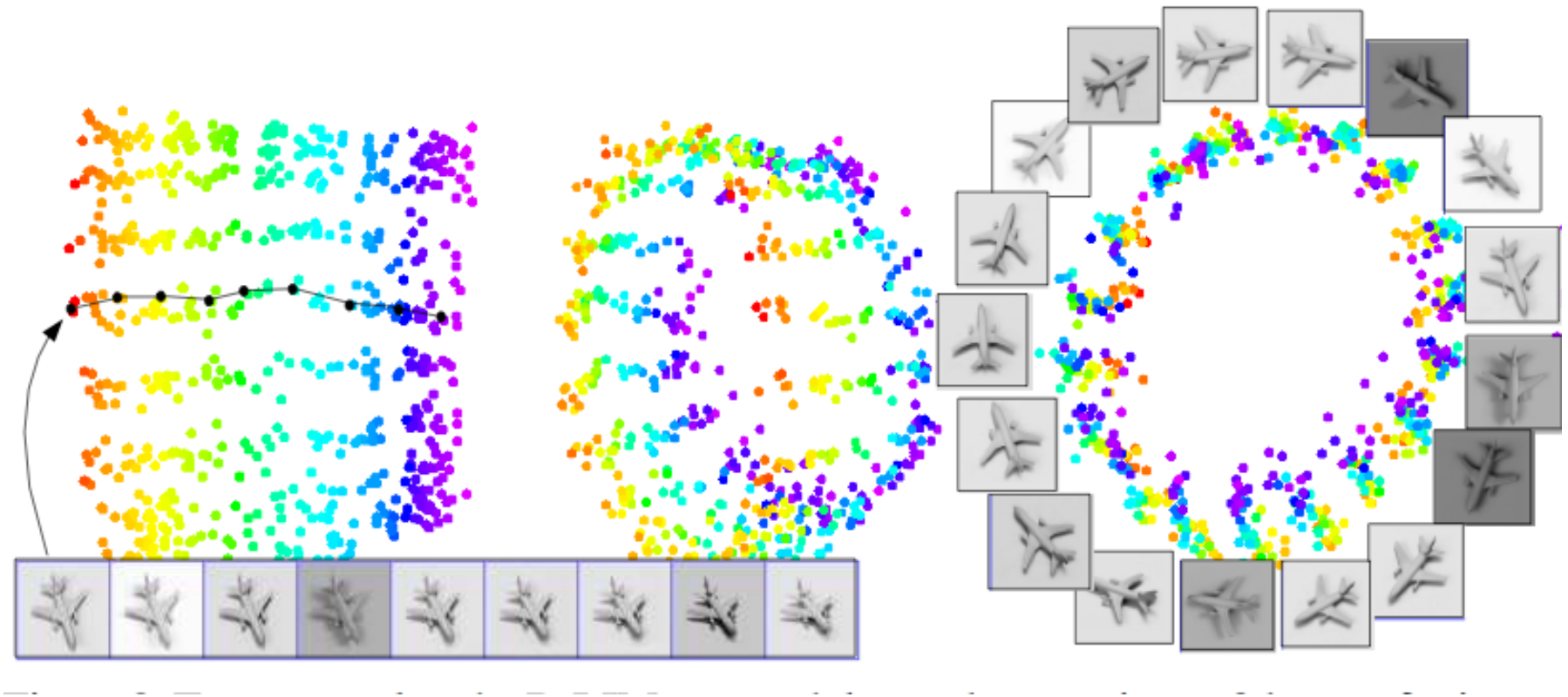


Example: Representing Natural Image Patches



$[a_1, \ldots, a_{64}] = [0, 0, \ldots, 0, \mathbf{0.8}, 0, \ldots, 0, \mathbf{0.3}, 0, \ldots, 0, \mathbf{0.5}, 0]$
(feature representation)

Example: Image Embedding



Applications

- Can be used as a pre-processing step to enable more accurate classification and regression on manifold data.
- Very low dimensional embeddings (ie: K=2,3) can be used to visualize complex manifold-structured data as in the previous example.
- Can be used to de-noise data by projecting to lower-dimensional space and then projecting back to the feature space.

Dimensionality Reduction vs Feature Selection

- The goal of feature selection is to remove features that are not informative with respect to the class label. This obviously reduces the dimensionality of the feature space.
- Dimensionality reduction can be used to compress the feature space for manifold structured data even when there is information in each of the feature dimensions so that none can be discarded.
- Another important property of dimensionality reduction is that it is unsupervised. It will attempt to preserve structure in the data that could be useful for a range of supervised problems, not a specific problem.
- Unlike feature selection, which is a supervised task, dimensionality reduction can sometimes compress out structure useful to a particular supervised task thereby increasing error.

Multidimensional Scaling

- MDS is a non-linear dimensionality reduction method that is explicitly designed to minimize the distortion in the pairwise distances between points when projecting them into a low dimensional embedding.
- Suppose we have a data set $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1:N}$.
- Let d_{ij} be the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Any distance metric can be used. Euclidean distance $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ is common.

Least Squares Multidimensional Scaling

- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Least-squares MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function, known as the *stress* function:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_N} \sum_{i<j} (d_{ij} - ||\mathbf{z}_i - \mathbf{z}_j||_2)^2$$

- Basically, this function attempts to have the regular Euclidean distances between points in $\mathbb{R}^K$ reflect the distances between the points in $\mathbb{R}^D$. This can be useful for data visualization when $K = 2$.

Classical Multidimensional Scaling Variants

- Let s_{ij} be the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$.
- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Classical MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function:

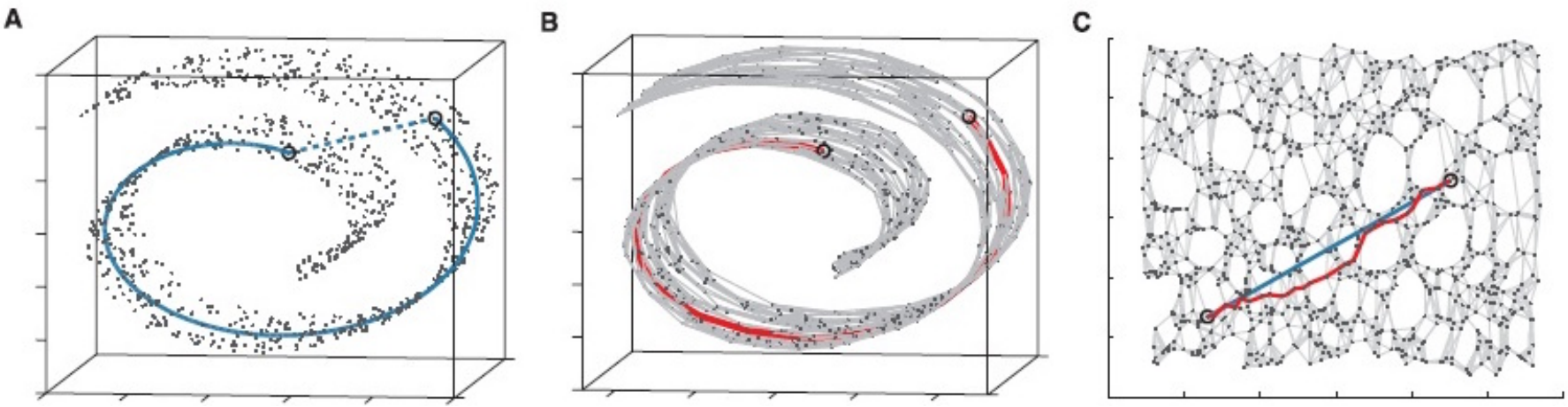
$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_N} \sum_{i<j} \left(s_{ij} - (\mathbf{z}_i - \bar{\mathbf{z}})^\top (\mathbf{z}_j - \bar{\mathbf{z}}) \right)^2$$

MDS Trade-offs

- Interestingly, to use MDS we actually don't need the raw feature vectors. It's enough to have the pairwise distance or similarity matrices.
- A significant issue with MDS is that we need to be able to specify a global similarity or distance matrix directly. This may not actually be easy to do if the data come from a complex manifold (regular Euclidean distance will fail).

Isometric feature mapping

- Isometric feature mapping (Isomap) is a non-linear dimensionality reduction method that is designed to minimize the distortion in geodesic distances on a manifold when projecting them into a low dimensional embedding.



Isometric feature mapping algorithm

- The isomap algorithm begins by computing the K-nearest neighbors of each data point and building the distance weighted K-nearest neighbor graph G over the data.
- For points i, j that are neighbors in the graph, isomap uses the straight line distance between them as d_{ij}.
- For points i, j that are not neighbors in the graph, isomap uses the length of the shortest path in G as d_{ij}. This is an approximation to the geodesic distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ on the manifold.
- Finally, isomap plugs the distances d_{ij} into MDS with classical scaling and computes the embedding.

Isomap Trade-offs

- Isomap fixes the problem where MDS needs a reasonable global notion of distance between points by assuming that Euclidean distances are reasonable over short length scales and using geodesic distance approximations over longer length scales.
- However, Isomap needs the input space to be sampled densely enough that the K-neareast neighbors of every data point are on the same part of the input manifold.
- Isomap can fail when different parts of the input manifold have different sampling density.

UMAP: High-Dimensional Relationships

- For each point $\mathbf{x}_i$, Uniform Manifold Approximation and Projection (UMAP) computes distances $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ to its K nearest neighbors. Any metric can be used.
- UMAP then computes a local "scale" parameters σ_i by solving the equation below using binary search. ρ_i is the distance to the closest neighbor.

$$\sum_{j \in \mathcal{N}_i} \exp\left(-\frac{d_{ij} - \rho_i}{\sigma_i}\right) = \log_2(K)$$

- It then defines a weight for each neighbor as shown below. Weights for non-neighbors are 0.

$$w_{ij} = \exp\left(-\frac{\max(0, d_{ij} - \rho_i)}{\sigma_i}\right)$$

- These weights represent local connectivity strengths in the original space and satisfy $0 \leq w_{ij} \leq 1$.

UMAP: Low-Dimensional Optimization

- UMAP defines a low-dimensional embedding vector $\mathbf{z}_i$ for each $\mathbf{x}_i$.
- It defines the similarity between low-dimensional embeddings of all pairs of data points as shown below were a, b are hyper-parameters.

$$s_{ij} = \frac{1}{1 + a \left\| \mathbf{z}_i - \mathbf{z}_j \right\|^{2b}}$$

- The low-dimensional embeddings are learned by minimizing the cross-entropy loss between the weights and the low-dimensional similarities.

$$L = -\sum_{i<j} \left[w_{ij} \log s_{ij} + (1 - w_{ij}) \log(1 - s_{ij}) \right]$$

- Pairs of data points with large weight are encouraged to be close in embedding space, while points with low weight are actually repulsed.