Dimensionality Reduction

- Many high dimensional datasets:
  - Gene expression microarrays
  - Text documents
  - digital images
  - SNP data
  - Clinical data
- Bad news: Learning is very hard in high dimensional data, especially when n (data point) < d (dimensions).
- Good news: No way any real-world data can be distributed uniformly in a high dimensional space. There should be an intrinsic dimensionality that is much smaller than the embedding dimensionality!

Dimensionality Reduction

Problems of learning in high dimensional spaces:

- Curse of dimensionality (all points become equidistant) => distance functions are not useful => problem for clustering, KNN,…

- Classification overfitting (to many parameter to set!).

- High computational costs.

- Bad learning behavior in high dimensional spaces.

    - Example: The optimal convergence rate for non-parametric regression is $n^{-p/(2p+d)}$.

    - Assume p=2, d=10 and n=10,000, if we increase d from 10 to 20, we have to increase n to 10,000,000 to achieve the same rate!

Dimensionality Reduction

- Feature selection:
  - Filter
  - Wrapper
  - Embedded
  - Markov Blanket
- Feature extraction/construction:
  - Clustering
  - PCA
  - MDS
  - Kernel PCA
  - ISO maps

Filter

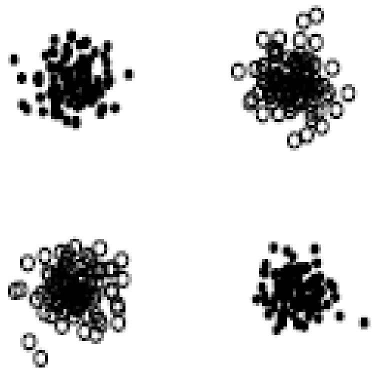
- Method: Rank each feature according to some univariate metric and select the highest ranking features.
- The scoring should reflect the discriminative power of each feature.
- Metric examples:
  - Fisher score: $V(i) = \frac{(\mu_{(+)}(i) - \mu_{(-)}(i))^2}{\sigma^2_{(+)}(i) + \sigma^2_{(-)}(i)}$
  - T-test: calculate p-value of the t-statistic assuming that the means are identical.
  - Information Gain: $I(i) = \sum_{x_i} \sum_{y} P(X = x_i, Y = y) \log \frac{P(X = x_i, Y = y)}{P(X = x_i) P(Y = y)}$
  - AUC of the ROC curve

Filter

- Correlation Filtering:
  - *Why?*: Diversify the features: highly correlated features tend to favor the same data.
  - Simple algorithm:
    - select features incrementally (according to some metric).
    - check the correlation of the new features with the already selected features.
    - If exceeds a threshold, do not add it!

Filter

- Advantages: Very efficient and fast to compute.
- Disadvantages: A feature that is not useful by itself can be very useful when combined with others. Filter methods can miss it!
  - Example1: "data mining" can be very predictive in document classification, while each individual term is not!
  - Example2: famous XOR example:



Wrapper

Objective: Search for the “best” subset of features

- Feature subset assessment:
  - Assess the quality of a set of features using a specific classification algorithm by internal cross-validation.
- Feature subset search:
  - We cannot do exhaustive search!
  - Apply some heuristic:
    - Forward selection
    - Backward elimination
    - Beam search
    - Simulated annealing

Embedded

Objective: Search for the "best" subset of features

- Feature selection is part of the model building, e.g. decision tree.
- Regularization:
  - Very important especially when we have large number of features but small sample size.
  - Automatically shuts down unnecessary features.
  - Incorporated into the objective function:

$$Error_{Reg}(\mathrm{w}, \mathrm{D}) = Error(\mathrm{w}, \mathrm{D}) + \lambda||\mathrm{w}||$$

Embedded Regularization

- Example: Lasso for linear regression (use L1 norm)

$$L = \sum_i (y_i - \sum_p \beta_p x_{ip})^2 + \lambda \sum_p ||\beta_p||_1$$

  - Lead to sparse solution

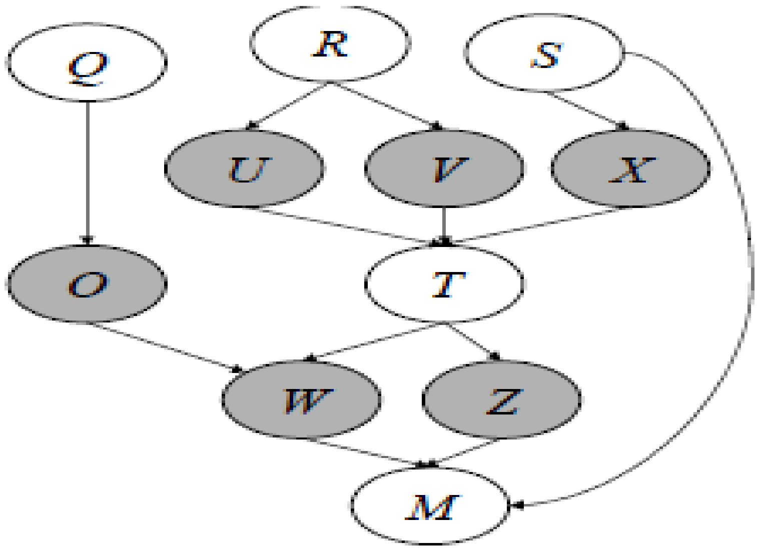
- Regularization= goodness of fit + complexity penalty.
- Perform features are selected in parallel with model learning.
- Regularization is incorporated in many scores (AIC, BIC,…).
- SVM also employs some sort of regularization by maximizing the margin. This is why SVM is less prone to overfitting.

Markov Blanket

- Markov Blanket of variable T (MB(T)) is the minimal set of variables, conditioned on which all other variables are probabilistically independent of T:

  P(T|MB(T))=P(T|V): V denote all variables.

- In Bayesian Network, MB is the set of parents, children and spouses.



Markov Blanket

- MB can be used for:
  - Variable selection for classification
  - Causal discovery: reduce the number of variables an experimentalist has to consider to discover direct causes of T.

- MB can be discovered by:
  - Applying a BN learning algorithm (e.g. PC, K2) to learn the whole network.
  - Apply a specific MB learning algorithm: usually faster than learning the whole structure.

Markov Blanket

The Incremental Association Markov Blanket (IAMB) algorithm [Tsamardinos, 2003]

- Forward phase:
  - *Objective:* Add all variables that belong in MB(T) and possibly more (false positives) the candidate MB (CMB) set.
  - *How:* start with CMB=ϕ, then add to CMB the variable X that maximizes mutual information: MI(X, T|CMB)
- Backward phase:
  - *Objective:* Remove the false positives from CMB so that CMB=MB(T) at the end.
  - *How:* Remove features one-by-one by testing whether a feature X from CMB is independent of T given the remaining CMB.

Dimensionality Reduction

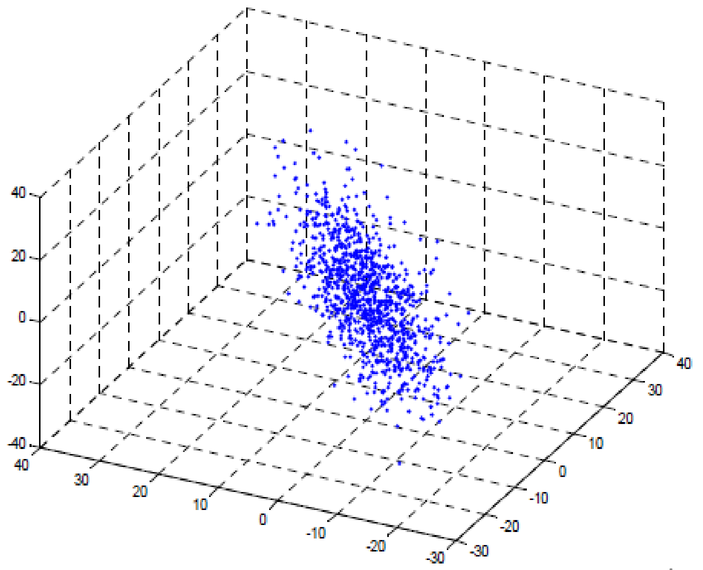
- Feature selection:
  - Filter
  - Wrapper
  - Embedded
  - Markov Blanket
- Feature extraction/construction:
  - Clustering
  - PCA
  - MDS
  - Kernel PCA
  - ISO maps

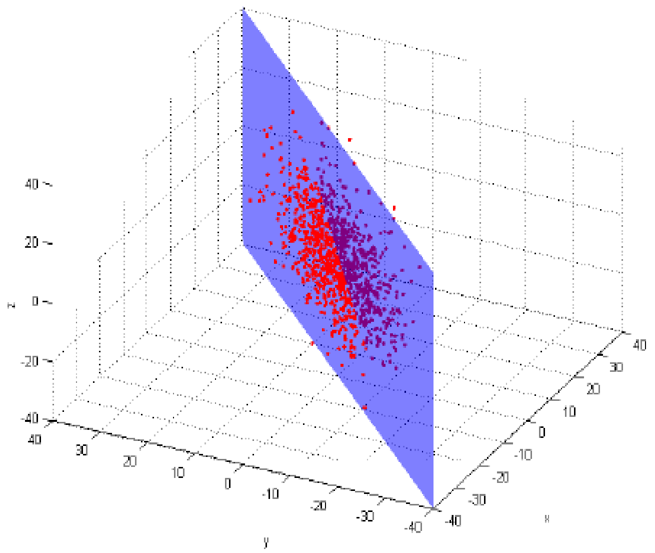
Clustering

- Clustering relies on a similarity measure: Euclidean distance, Mahalanobis distance, Cosine distance…
- Deterministic clustering methods (like k-means or hierarchical clustering) is not very useful.
- It is better to use soft (probabilistic) clustering:
    - Example: Mixture of Gaussian.
    - Replace each data point with the set of cluster posteriors.
    - $x \rightarrow P(c=i|x)$: number of features = number of clusters.

PCA

- PCA: Principle Component Analysis (closely related to SVD).
- PCA finds a ***linear*** projection of high dimensional data into a lower dimensional subspace such as:
  - The variance retained is maximized.
  - The least square reconstruction error is minimized.





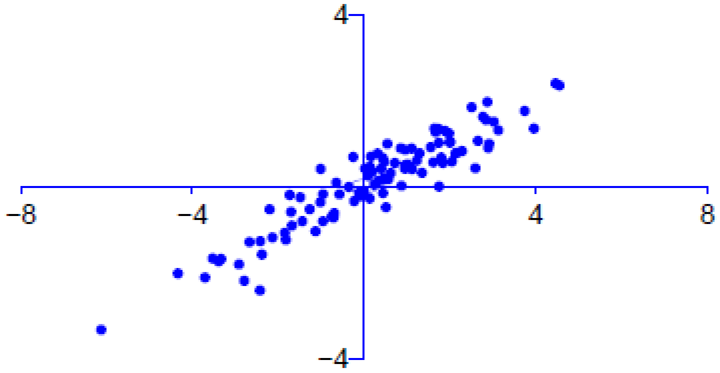
PCA

PCA steps (to reduce dimensionality from *d* to *m*):

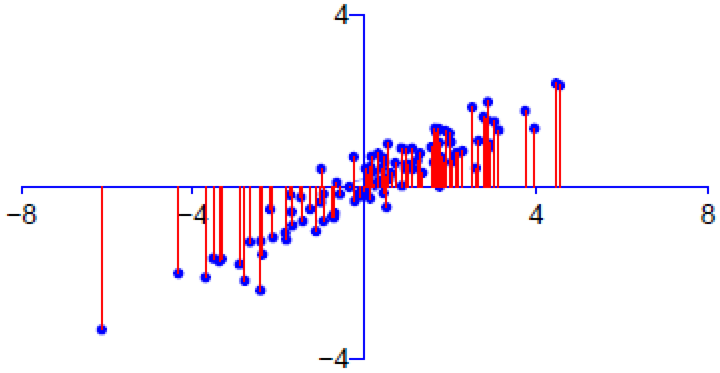
- Center the data (subtract the mean).
- Calculate the *dxd* covariance matrix: C= $\frac{A^T A}{N}$
- Calculate the eigenvectors of the covariance matrix (orthogonal).
- Select the *m* eigenvectors that correspond to the heights *m* eigenvalues to be the new space dimensions.
  - The variance in each new dimension is given by the eigenvalues.
  - Note that if we use all eigenvectors, we do loose any information (space rotation).
  - How to select *m*? Look for prominent gap in the eigenvalue spectrum

PCA

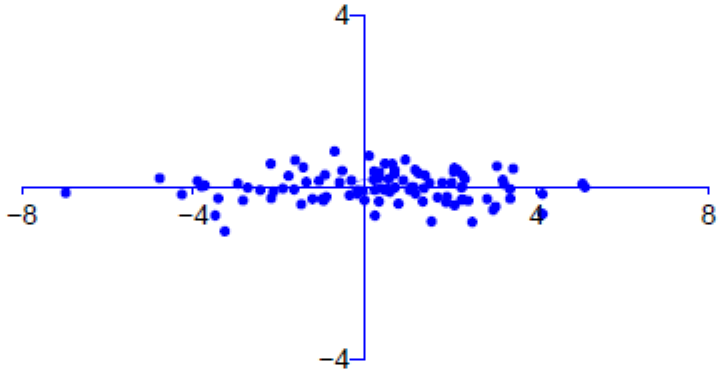
Feature selection vs. Feature extraction



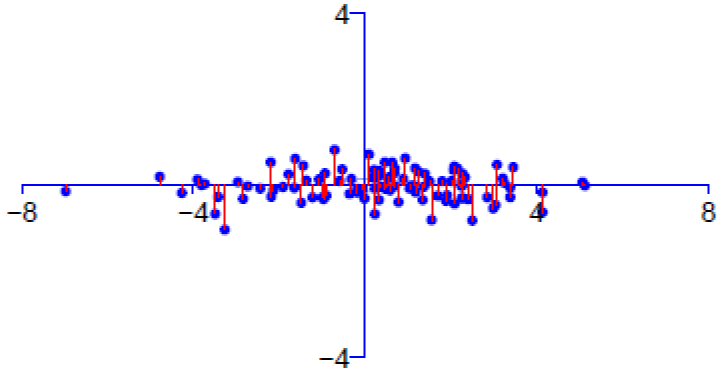
Original data



Project on the axis with highest variance



De-correlate the data with PCA



Residuals are much reduced

PCA (derivation)

- Find the direction for which the variance is maximized:

$$\begin{aligned} \mathbf{r}_1 &= \operatorname{argmax}_{\mathbf{r}_1} \operatorname{var}\left(\hat{\mathbf{Y}}\mathbf{r}_1\right) \\ \text{subject to :} \quad & \mathbf{r}_1^{\mathrm{T}}\mathbf{r}_1 = 1 \end{aligned}$$

- Rewrite in terms of the covariance matrix:

$$\text{var}= N^{-1}\left(\hat{\mathbf{Y}}\mathbf{r}_1\right)^{\mathrm{T}}\hat{\mathbf{Y}}\mathbf{r}_1 = \mathbf{r}_1^{\mathrm{T}}\underbrace{\left(N^{-1}\hat{\mathbf{Y}}^{\mathrm{T}}\hat{\mathbf{Y}}\right)}_{\text{sample covariance}}\mathbf{r}_1 = \mathbf{r}_1^{\mathrm{T}}\mathbf{S}\mathbf{r}_1$$

- Solve via constrained optimization:

$$L\left(\mathbf{r}_1, \lambda_1\right) = \mathbf{r}_1^{\mathrm{T}}\mathbf{S}\mathbf{r}_1 + \lambda_1\left(1 - \mathbf{r}_1^{\mathrm{T}}\mathbf{r}_1\right)$$

PCA (derivation)

$$L(\mathbf{r}_1, \lambda_1) = \mathbf{r}_1^{\mathrm{T}} \mathbf{S} \mathbf{r}_1 + \lambda_1 \left(1 - \mathbf{r}_1^{\mathrm{T}} \mathbf{r}_1\right)$$

- Gradient with respect to r1

$$\frac{\mathrm{d}L(\mathbf{r}_1, \lambda_1)}{\mathrm{d}\mathbf{r}_1} = 2\mathbf{S}\mathbf{r}_1 - 2\lambda_1 \mathbf{r}_1 \Longrightarrow \mathbf{S}\mathbf{r}_1 = \lambda_1 \mathbf{r}_1 .$$

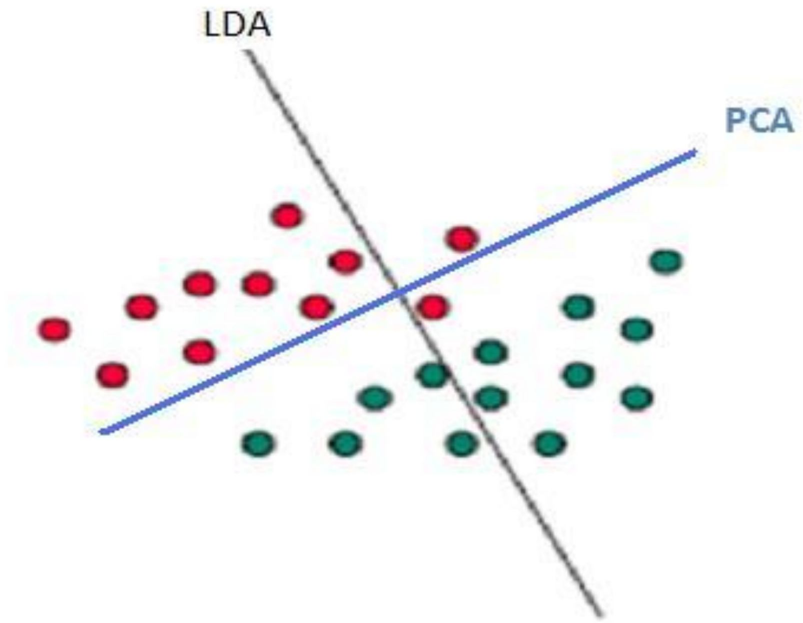
This is the eigenvalue problem!

- Multiply by r_1^T : $\lambda_1 = \mathbf{r}_1^{\mathrm{T}} \mathbf{S} \mathbf{r}_1 .$

The projection variance of each principal component is given by its eigenvalue

PCA

- Unsupervised: maybe bad for classification!



Some PCA/SVD applications

- LSI: Latent Semantic Indexing.
- Google/PageRank algorithm (random walk with restart).
- Kleinberg/Hits algorithm (compute hubs and authority scores for nodes).
- Image compression (*other methods: DCT used in JPEG, and wavelet compression which we will discuss later!*)
- Data visualization (by projecting the data on 2D).

PPCA

- [Tipping and Bishop 1999] showed that PCA can be expressed as the maximum likelihood solution of a probabilistic latent variable model.
- Advantages:
  - We can use an EM algorithm that avoids evaluating the covariance matrix.
  - EM allows us to incorporate missing values in the data.
  - Can perform a mixture of PCA.
  - The dimensionality of the principal subspace can be automatically found from data with a Bayesian treatment.
  - PPCA can run generatively to provide samples from the distribution.

PPCA

- Let *z* be a latent variable that represent the Principal-component subspace, then the distribution of data given z is:

$$p(x|z) = N(x|Wz + \mu, \sigma^2 I)$$

- Model parameters are W, μ and σ^2 : estimated using maximum likelihood.

- There is a closed-form solution.

- However, it is faster to apply EM for high dimensions.

- PPCA is naturally expressed as a mapping from the latent space to the data space. To reverse the mapping, we apply Bayes' theorem.

MDS

- MDS: Multidimensional scaling [Cox and Cox, 1994] is often used in visualization.
- MDS give points in a low dimensional space such that the Euclidean distances between them reproduce the original distance matrix.

  Given distance matrix

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \cdots & \delta_{I,I} \end{pmatrix}.$$

  Map the input points x_i to z_i such as $\|z_i - z_j\| \approx \delta_{ij}$

- In classical MDS, this norm is the Euclidean distance (principal coordinate analysis)
- Distances → inner products (Gram matrix) → embedding
- There is a formula to obtain the Gram matrix G from the distance matrix Δ.

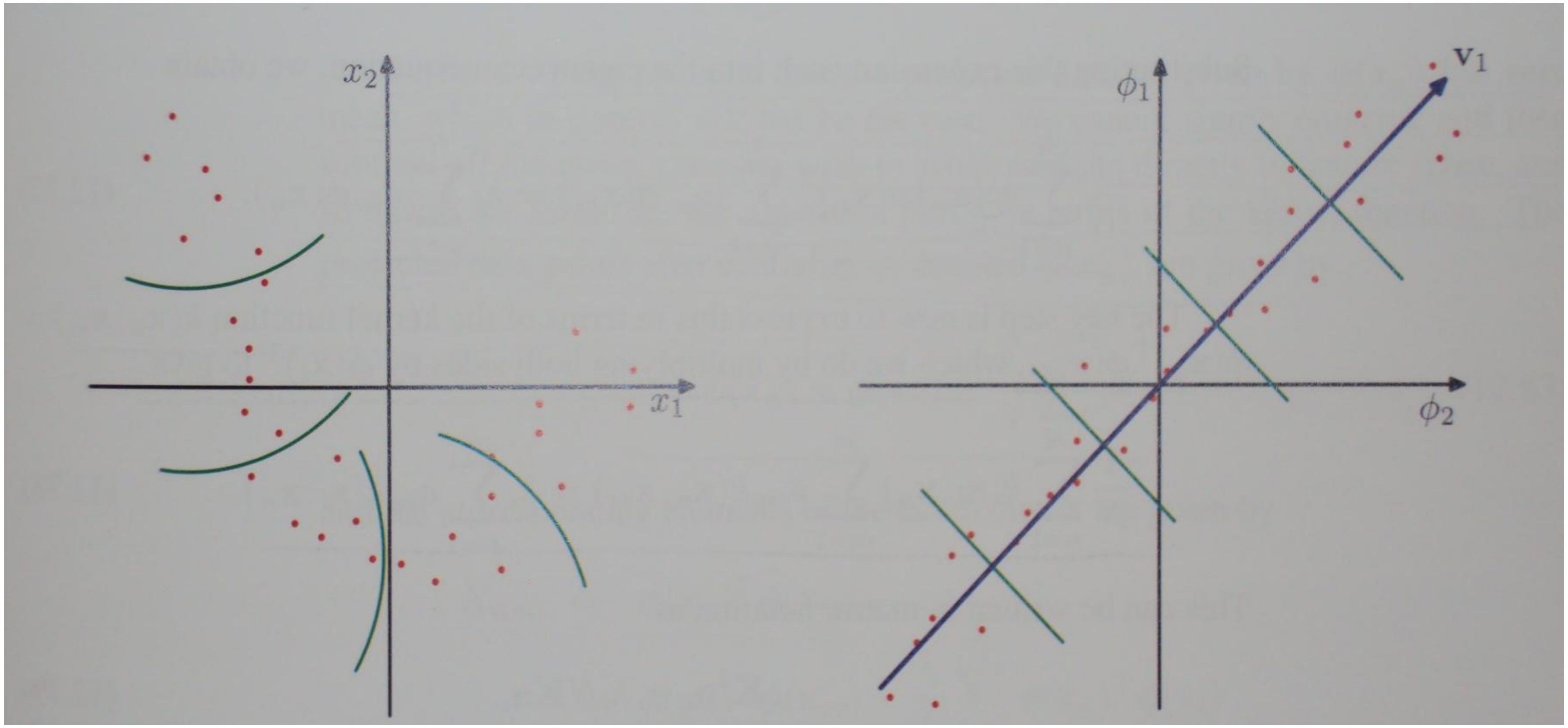
PCA and MDS duality

- Preserve Euclidean distances = retain features with largest variance
- PCA uses the covariance matrix (*dxd*): $C=n^{-1}X^TX$
- MDS uses the Gram (inner product) matrix (*NxN*): $G=XX^T$
- G has the same rank and eigenvalues (up to a constant) as C.
- *Classical MDS is equivalent to PCA when the distances in the input space are the Euclidean distance.*
- If d>N, do MDS with cost $O(N^3)$
- If n>d, do PCA with cost $O(d^3)$
- If we do not have the points in the original space, and we have only a distance matrix, we cannot perform PCA! (we don't know *d*).
- Note that both PCA and MDS are invariant to space rotation!

Kernel PCA

- Kernel PCA [Scholkopf et al. 1998] performs nonlinear projection.
- Given input (x_1, … x_n), kernel PCA computes the principal components in the feature space ($\varphi(x_1)$, … $\varphi(x_n)$).
- Avoid explicitly constructing the covariance matrix in feature space.
- Use the kernel trick: formulate the problem in terms of the kernel function $k(x,x')= \varphi(x). \varphi(x')$ without explicitly doing the mapping.
- Popular kernels: polynomial or Gaussian.
- Kernel PCA is non-linear version of MDS (use Gram matrix=Kernel matrix) in the feature space instead of Gram matrix in the input space.

Kernel PCA



Original space

A non-linear feature space

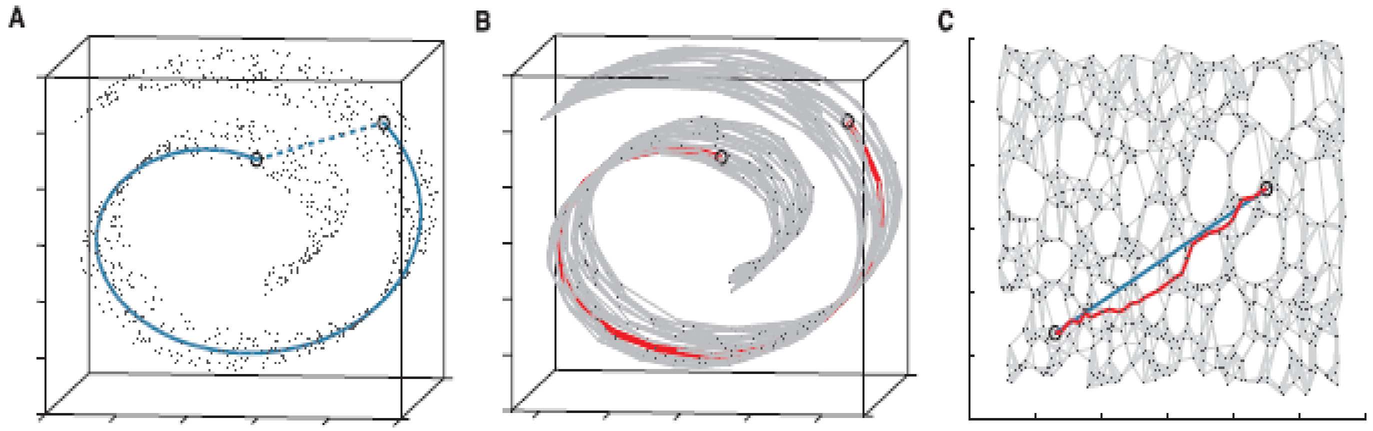
Kernel PCA

- The number of principal components in the feature space can be higher than the original dimensionality!
- However, the number of principal components cannot be bigger than *N* because kernel PCA uses the *NxN* kernel matrix (remember duality between PCA and MDS).
- The generic kernels do not usually perform well, therefore we should define more data oriented kernels!
- We should try to model the data manifold!

Isomap

- Isomap [Tenenbaum et al. 2000] tries to preserve the distances along the data Manifold (Geodesic distance).

- Cannot compute Geodesic distances without knowing the Manifold!

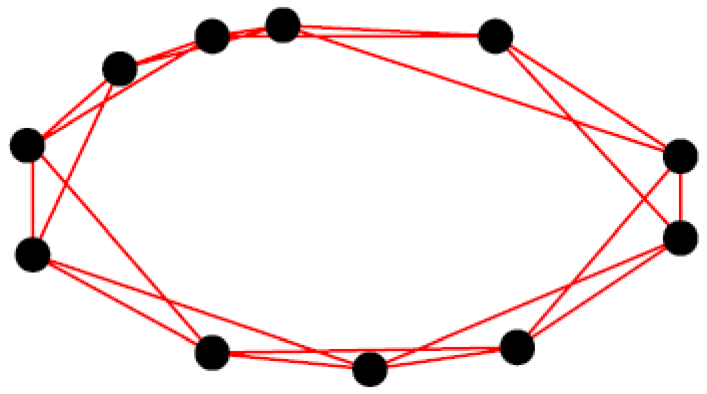


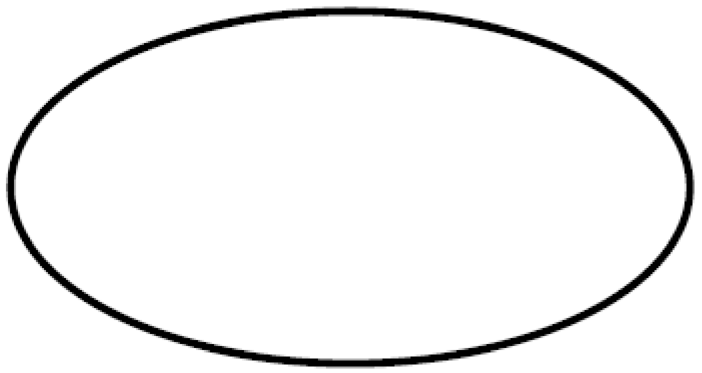
Blue: true manifold distance, red: approximated shortest path distance

- Approximate the Geodesic distance by the shortest path in the adjacency graph

Isomap

- Construct the neighborhood graph (connect only k-nearest neighbors): the edge weight is the Euclidean distance.



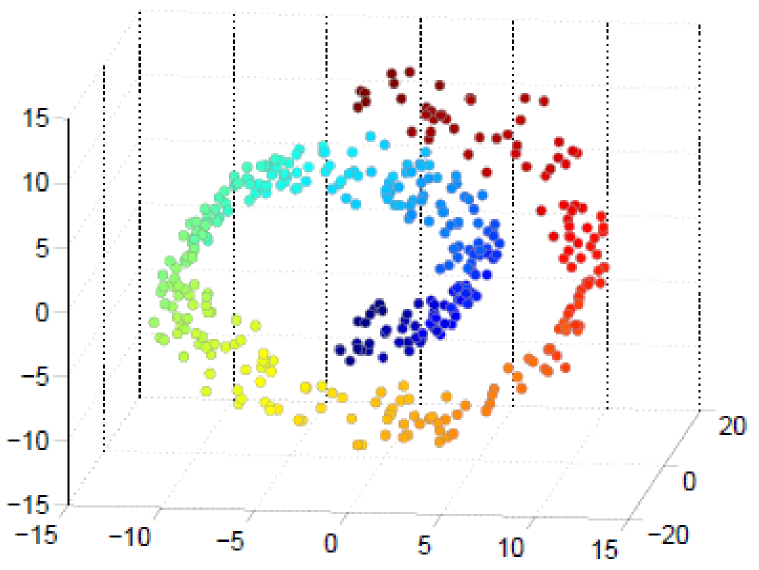


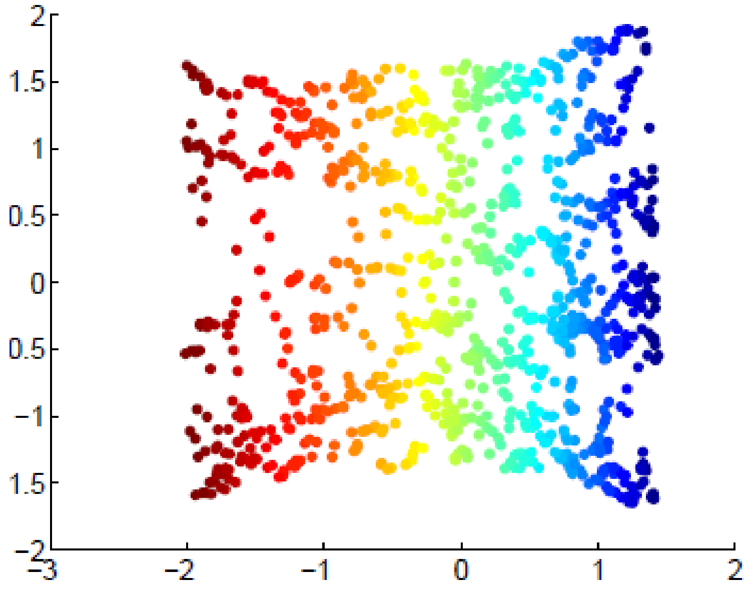
- Estimate the pairwise Geodesic distances by the shortest path (use Dijkstra algorithm).

- Feed the distance matrix to MDS.

Isomap

- Euclidean distances between outputs match the geodesic distances between inputs on the Manifold from which they are sampled.





Related Feature Extraction Techniques

Linear projections:

- Probabilistic PCA [Tipping and Bishop 1999]
- Independent Component Analysis (ICA) [Comon , 1994]
- Random Projections

Nonlinear projection (manifold learning):

- Locally Linear Embedding (LLE) [Roweis and Saul, 2000]
- Laplacian Eigenmaps [Belkin and Niyogi, 2003]
- Hessian Eigenmaps [Donoho and Grimes, 2003]
- Maximum Variance Unfolding [Weinberger and Saul, 2005]