

# LECTURE 5: CLASSICAL ML: DIMENSIONALITY REDUCTION / LINEAR AND NON-LINEAR METHODS

EKATERINA MURAVLEVA

---

# OVERVIEW OF THE COURSE

Lecture 1: General course information, CRISP-DM methodology

Lecture 2: Supervised learning/unsupervised learning. Classification/regression problems. Accuracy metrics (precision, recall, ROC-AUC scores). Concept of loss functions, overfitting / underfitting.

Lecture 3: Classical ML: Linear regression, logistic regression, support vector machine

Lecture 4: Classical ML: Decision trees, random forests, boosting.

Lecture 5: Classical ML: Dimensionality reduction: linear, non-linear methods.

Lecture 6: Classical ML: Clustering methods

Lecture 7: Basic neural networks

Lecture 8: Scalable algorithms



# RECAP OF LECTURE 4

- Decision trees
- Random forests
- Gradient boosting



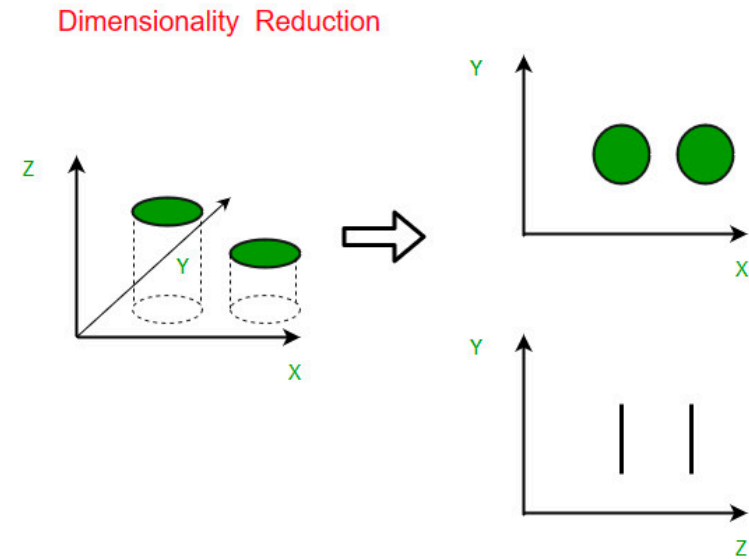
# PLAN OF LECTURE 5

- Dimensionality reduction
- Linear methods
- Non-linear methods

# WHAT IS DIMENSIONALITY REDUCTION

**Dimensionality reduction** is the process of reducing the number of variables (features) in the dataset, while retaining as much information as possible.

It involves transforming a high-dimensional dataset into a lower-dimensional space, making it easier to analyze and visualize.



---

# TYPES OF DIMENSIONALITY REDUCTION

The simplest dimensionality reduction would be **feature selection**;

we only select features that are relevant for the **downstream task**.

Another type is **feature extraction**, which involves certain transformation of the features.

The simplest approach is linear transformation of the features.

---

# LINEAR DIMENSIONALITY REDUCTION

In linear dimensionality reduction, given features  $x_i$  we process them using linear transformation:

$$z_i = Wx_i, \text{ where } x_i \in \mathbb{R}^N, \quad z_i \in \mathbb{R}^d \text{ and } d \ll N$$

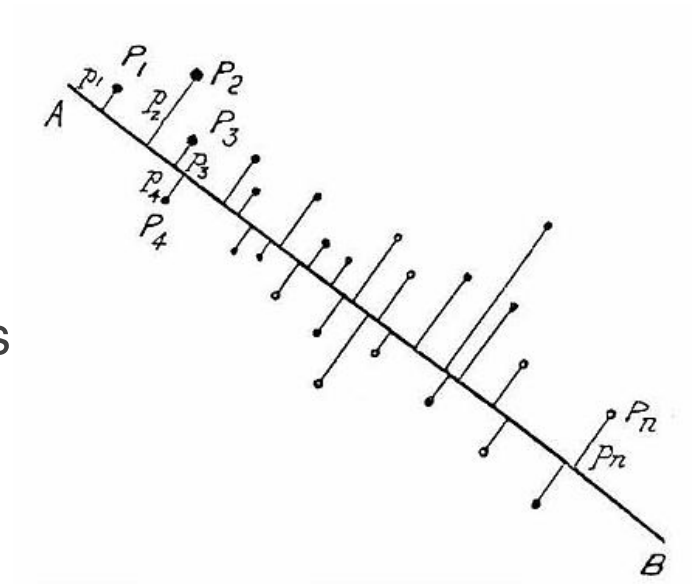
How to choose the projection matrix  $W$ ?

# PRINCIPAL COMPONENT ANALYSIS (1)

The classical approach for linear dimensionality reduction is **principal component analysis (PCA)**.

As proposed by Pearson in 1901, the idea is to find a  $k$ -dimensional subspace  $L_k$  such that the sum of squares of distances from all datapoints to  $L_k$  is minimal:

$$\sum_{i=1}^m \text{dist}^2(x_i, L_k) \rightarrow \min$$

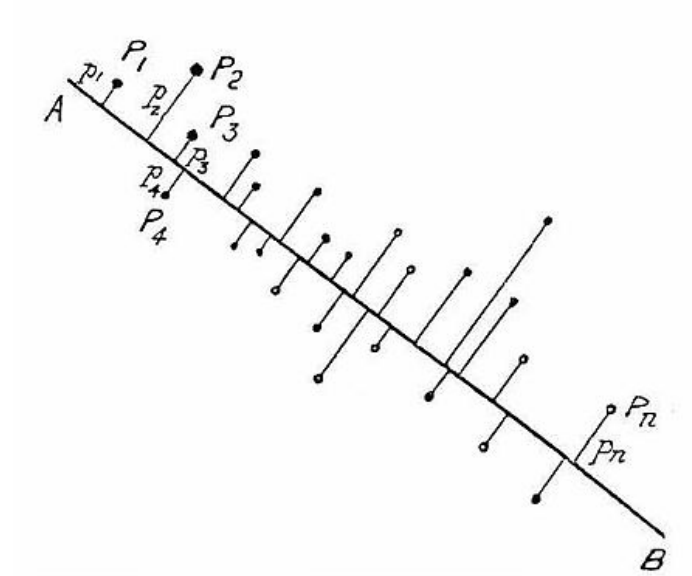


# PRINCIPAL COMPONENT ANALYSIS (2)

$$\sum_{i=1}^m \text{dist}^2(x_i, L_k) \rightarrow \min$$

Each vector in  $L_k$  can be written as a linear combination

$$L_k = \{a_0 + \beta_1 a_1 + \cdots + \beta_k a_k \mid \beta_i \in \mathbb{R}\}, \text{ where } a_s \text{ are basis vectors}$$



# PCA: HOW IT WORKS

The classical algorithm for Principal Component Analysis has the following form:

1. Compute mean and covariance matrices

$$\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i, \quad C = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x}_i)(x_i - \bar{x}_i)^\top,$$

2. Compute  $d$  largest eigenvalues of the covariance matrix  $C$  and eigenvectors

The corresponding matrix performs the dimensionality reduction.

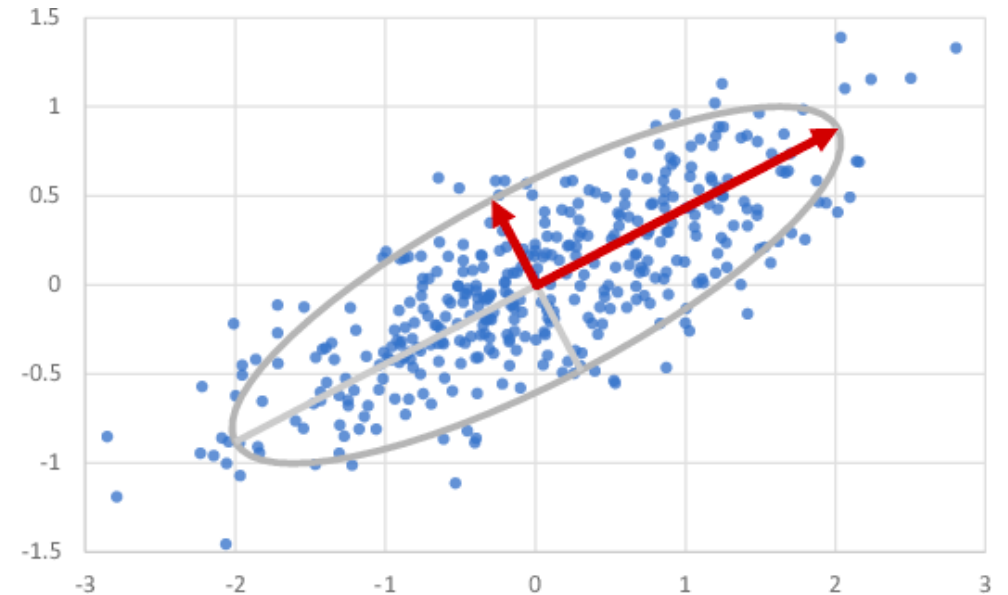
# EXAMPLE OF PCA COMPONENTS

The PCA builds the basis of ordered components, such that features

$z = Q^T x$  capture most of the variance of the dataset.

In the example on the right, the first component (long arc of the ellipsoid) covers most of the variance.

Now lets discuss how PCA can be derived.



# DERIVATION OF THE PCA

Let  $x$  be a datapoints and  $Q = [q_1, \dots, q_N]$  be the orthonormal basis (we assume that the data has zero mean)

The first  $d$  components will be **principal components**

The vector  $x$  can be approximated by the vector  $\hat{x}$  with  $d$  components as

$$\hat{x} = \sum_{j=1}^d (x, q_j) q_j$$

Now we want to find  $q_j$  that minimizes the reconstruction error of such projection

$E\|x - \hat{x}\|^2$  where  $E$  denotes the mathematical expectation, and we get

$$E\|x - \hat{x}\|^2 = \sum_{j=d+1}^N (Cq_j, q_j) \text{ where}$$

$C = Exx^\top$  is the **covariance matrix** of the dataset

It can be shown that the optimal values are obtained if  $q_1, \dots, q_d$  are the largest eigenvalues of the covariance matrix  $C$

# DERIVATION OF THE PCA (2)

$$\hat{x} = \sum_{j=1}^d (x, q_j) q_j$$

Now we want to find  $q_j$  that minimizes the reconstruction error of such projection

$E\|x - \hat{x}\|^2$  where  $E$  denotes the mathematical expectation.

It can be shown that

$$E\|x - \hat{x}\|^2 = \sum_{j=d+1}^N (Cq_j, q_j) \text{ where}$$

$C = Exx^T$  is the **covariance matrix** of the dataset

It can be shown that the optimal values are obtained if  $q_1, \dots, q_d$  are the largest eigenvalues of the covariance matrix  $C$

# EXAMPLE: FIRST COMPONENT

Suppose we want to find a linear projection  $z = w^\top x$  that captures most of the variance of the dataset, where  $\|w\| = 1$

$$E\|z\|^2 \rightarrow \max$$

Then,

$$E\|z\|^2 = E\|w^\top x\|^2 = (Cw, w), \quad C = Exx^\top$$

The maximum of this quadratic form with norm constrain is attained at the eigenvector, corresponding to the largest eigenvalue of  $C$

# SVD AND PCA

A more numerically stable way to compute the PCA is not by computing the eigenvectors of the covariance matrix, but computing left singular vectors of the data matrix

$$X = [x_1, \dots, x_m]$$

$X = U\Sigma V^T$ , where  $U$  and  $V$  are orthogonal matrices, and  $\Sigma$  is the diagonal matrix with singular values on the diagonal  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N$

More details in numerical linear algebra course!

---

# ADVANTAGES OF PCA

- (+) PCA gives a robust way to reduce dimensionality of features.
- (+) Can be used to visualize datasets as two-dimensional projections
- (+) Reduces computational time
- (+) May remove redundant features
- (+) May prevent overfitting: if we use PCA features, overfitting can be less

---

# DISADVANTAGES OF PCA

- (-) Can lead to loss of information, if the number of components is small for a particular dataset
- (-) Does not work if the data do not lie on a  $d$ -dimensional hyperplane.

---

# LINEAR DISCRIMINANT ANALYSIS

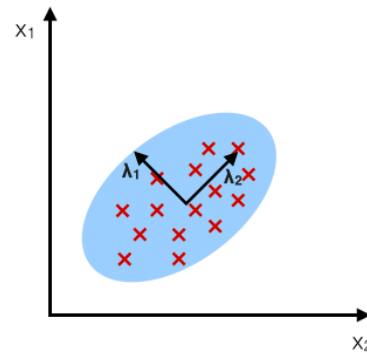
- PCA is unsupervised; when we have information about classes we can try to find linear combination of variables that maximize class information, rather than feature variance.
- This is at the core of **linear discriminant analysis** (LDA)
- It is a variant of **supervised dimensionality reduction**

# LINEAR DISCRIMINANT ANALYSIS

For binary classification, we just need to find a linear combination, i.e. reducing the dimension to just one.

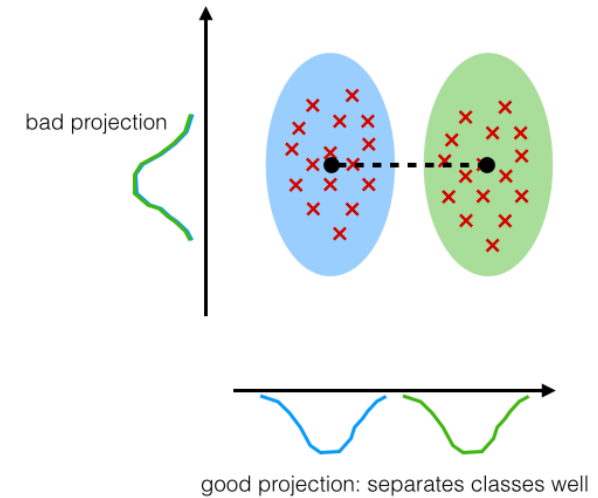
## PCA:

component axes that maximize the variance



## LDA:

maximizing the component axes for class-separation



# COMPUTING LDA

Suppose we have two classes.

For each class, we compute the mean value  $\mu_1, \mu_2$  for each class, and intra-class variance (variance of samples for each class)  $\sigma_1, \sigma_2$

**Fisher linear discriminant is defined as**

$$J(w) = \frac{\|\mu_1 - \mu_2\|^2}{\sigma_1^2 + \sigma_2^2}$$

We want the mean values to be different, but scale by the variance

# OPTIMIZATION PROBLEM FOR LDA

The Fisher discriminant can be written as

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \rightarrow \max$$

Where  $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$  is called **between-class scatter** (for two classes - at most rank 1)

And  $S_W = S_1 + S_2$  is **within-class scatter**, some of two covariance matrices for both classes.

The maximization problem leads to generalized eigenvalue problem

$$S_W^{-1} S_B w = \lambda w$$

But since it is rank 1, it admits a very simple solution

$$w_* = S_W^{-1} (\mu_1 - \mu_2)$$

# OPTIMIZATION PROBLEM FOR LDA

The Fisher discriminant can be written as

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \rightarrow \max$$

Where  $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$  is called **between-class scatter** (for two classes - at most rank 1)

And  $S_W = S_1 + S_2$  is **within-class scatter**, some of two covariance matrices for both classes.

The maximization problem leads to generalized eigenvalue problem

$$S_W^{-1} S_B w = \lambda w$$

But since it is rank 1, it admits a very simple solution

$$w_* = S_W^{-1} (\mu_1 - \mu_2)$$

# SOLUTION OF OPTIMIZATION PROBLEM FOR LDA

The Fisher discriminant can be written as

$$J(w) = \frac{w^T S_B w}{w^T S_w w} \rightarrow \max$$

The maximization problem leads to generalized eigenvalue problem

$$S_W^{-1} S_b w = \lambda w$$

But since it is rank 1, it admits a very simple solution

$$w_* = S_W^{-1} (\mu_1 - \mu_2)$$

---

# LDA ALGORITHM

1. Compute the  $d$ -dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Solve generalized eigenvalue problem and choose top  $k$  eigenvalues.



# LDA AND SVM

LDA and SVM look similar (construct separating hyperplane) but they are different.

SVM focuses on support vectors.

LDA focuses on the whole dataset.

LDA can be used for dimensionality reduction.

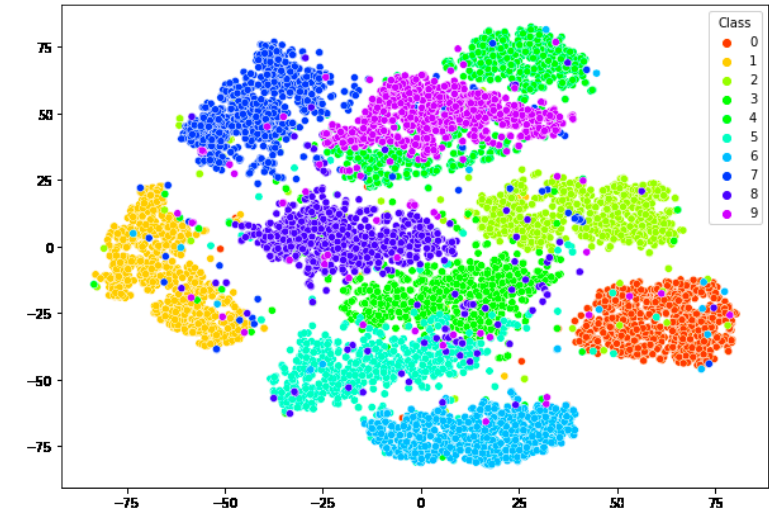
# NONLINEAR DIMENSIONALITY REDUCTION

Lets start again with the basic problem:

we want to replace high-dimensional features with low-dimensional features.

For example, we are given MNIST dataset (28x28 images)

and we want to plot 2D projections of it to visualize it.



---

# EMBEDDINGS

Such visualizations can be done by non-linear dimensionality reduction.

Key idea: try to assign low-dimensional vector  $y_i$  to each sample  $x_i$  in the dataset.

What could be the loss function for such embedding?

Try to keep similarities between points!

# T-SNE: PROBABILITY

In t-SNE (**t-distributed Stochastic Nearest Neighbour**)

We assign probabilities proportional to the similarity of objects:

$$\text{For } j \neq i \text{ set } p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2\right)}$$

Note they sum up to 1 over  $j$ .

# T-SNE:

In high dimensions, the Euclidean distance does not work well (i.e. probabilities become closer to a constant),

Thus the embeddings  $y_i$  are assumed to model different type of the distribution, namely

Student t-distribution:

$$q_{ij} = \frac{\left(1 + \left\| \mathbf{y}_i - \mathbf{y}_j \right\|^2\right)^{-1}}{\sum_k \sum_{l \neq k} \left(1 + \left\| \mathbf{y}_k - \mathbf{y}_l \right\|^2\right)^{-1}}$$

Typically,  $y_i$  are 2D or 3D for t-SNE.

Then, we optimize over  $y_i$  the KL-divergence between P and Q:

$$KL(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

---

# T-SNE: SUMMARY

T-SNE is a very popular visualization tool. Due to optimization, it scales badly with the size of the dataset.

Another popular tool is UMAP (Uniform Manifold Approximation and Projection)

The theory and algorithms of UMAP is advanced, see <https://pair-code.github.io/understanding-umap/>

It is approx 10x time faster!

The idea uses the graph constructed from the data. The graph idea is quite important for spectral methods

---

# GRAPH METHODS FOR DIMENSIONALITY REDUCTION

We will briefly cover the classical graph-based methods for dimensionality reduction:

Laplacian eigenmaps and IsoMap.

# LAPLACIAN EIGENMAPS

The **first step** is to construct the **graph** from the data.

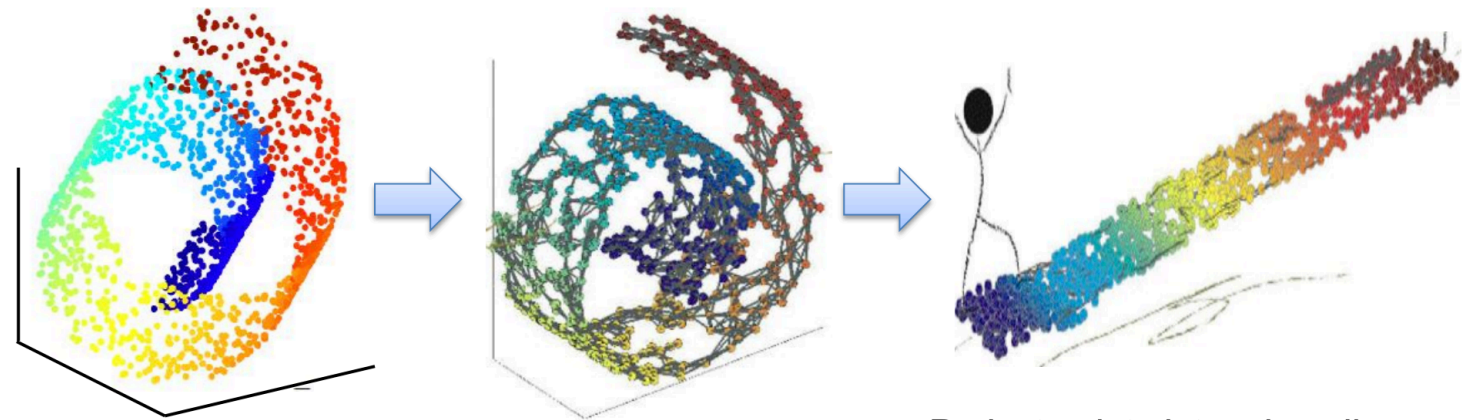
Two options:

- 1) Eps-neighborhood. All points within distance  $\varepsilon$  are connected
- 2) n nearest neighbors.

The **second step** is to select the weights:

- 1) Heat kernel  $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}}$ ;

- 2) Just constant.



Construct graph from data points  
(capture local information)

Project points into a low-dim  
space using “eigenvectors of  
the graph”

---

# LAPLACIAN EIGENMAPS

We have the weighted adjacency matrix  $W$

We build the **graph Laplacian**

$$L = D - W,$$

where diagonal matrix has degrees of the vertices on the diagonal:

$$d_i = \sum_j w_{ij} \text{ degree of a vertex}$$

---

# LAPLACIAN EIGENMAPS: EMBEDDINGS

Given the graph Laplacian, we compute the lowest eigenvalues/eigenvectors of it.

Suppose we compute  $d$  eigenvectors.

Then, we get the matrix of size  $m \times d$  where  $m$  is the size of the dataset and  $d$  is the reduced dimension.

Each row of this matrix **represents the embedding of a datapoint!**



# PCA VS LAPLACIAN EIGENMAPS

## PCA

Linear embedding

Based on the largest eigenvalues of the covariance matrix

Eigenvectors give latent features

To get embedding of the points, project onto latent features:

## Eigenmaps

Nonlinear embedding

Based on the smallest eigenvectors of the  $m \times m$  Laplacian matrix between datapoints

Eigenvectors give embeddings

---

# WHY GRAPH LAPLACIAN

The motivation behind eigenvectors of the graph Laplacian can be understood from the quadratic form associated with it:

$$(Lx, x) = \sum_{ij} w_{ij} \|x_i - x_j\|^2$$

Thus minimization of this quadratic form leads to embeddings that try to preserve the structure of the graph (advanced topic, related to manifolds).

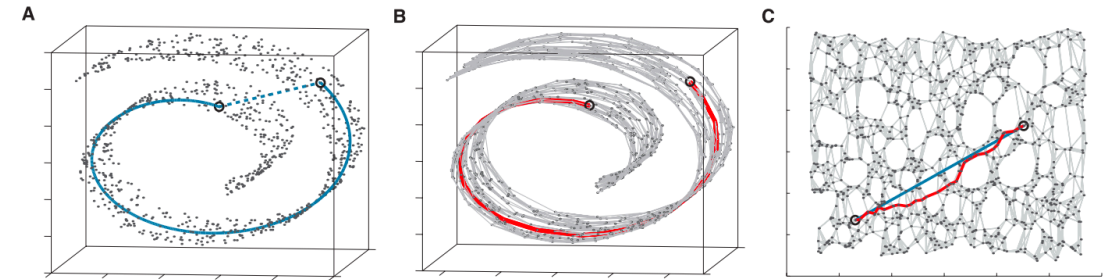
# ISOMAP

High-level idea of ISOMAP is as follows:

Instead of Euclidean Distance between points use graph distance

On Swiss roll dataset the points may be close to each other in the euclidean space, but far in the 'geodesic distance'.

Once the metric is fixed, we can learn embeddings to approximate this metric.



**Fig. 3.** The "Swiss roll" data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (A) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (B) The neighborhood graph  $G$  constructed in step one of Isomap (with  $K = 7$  and  $N =$

1000 data points) allows an approximation (red segments) to the true geodesic path to be computed efficiently in step two, as the shortest path in  $G$ . (C) The two-dimensional embedding recovered by Isomap in step three, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding (blue) now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).



# RECAP OF LECTURE 5

- Dimensionality reduction
- Linear methods
- Non-linear methods



# NEXT LECTURE

Classical ML:

Clustering methods (unsupervised learning)