Machine learning

Dimensionality reduction

Hamid Beigy

Sharif University of Technology

December 25, 2021

Table of contents

Introduction

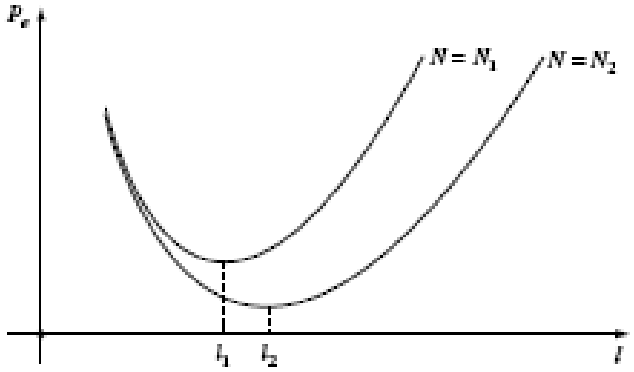
The complexity of any classifier or regressors depends on the number of input variables or features. These complexities include

- Time complexity: In most learning algorithms, the time complexity depends on the number of input dimensions(D) as well as on the size of training set (N). Decreasing D decreases the time complexity of algorithm for both training and testing phases.
- Space complexity: Decreasing D also decreases the memory amount needed for training and testing phases.
- Samples complexity: Usually the number of training examples (N) is a function of length of feature vectors (D). Hence, decreasing the number of features also decreases the number of training examples.
  Usually the number of training pattern must be 10 to 20 times of the number of features.

There are several reasons why we are interested in reducing dimensionality as a separate preprocessing step.

- Decreasing the time complexity of classifiers or regressors.
- Decreasing the cost of extracting/producing unnecessary features.
- Simpler models are more robust on small data sets. Simpler models have less variance and thus are less depending on noise and outliers.
- Description of classifier or regressors is simpler / shorter.
- Visualization of data is simpler.

- In practice, for a finite N, by increasing the number of features we obtain an initial improvement in performance, but after a critical value further increase of the number of features results in an increase of the probability of error. This phenomenon is also known as the peaking phenomenon.



- If the number of samples increases ($N_2 \gg N_1$), the peaking phenomenon occures for larger number of features ($l_2 > l_1$).

High-dimensional space

- In most applications of data mining/ machine learning, typically the data is very high dimensional (the number of features can easily be in the hundreds or thousands).
- Understanding the nature of high-dimensional space (hyperspace) is very important, because hyperspace does not behave like the more familiar geometry in two or three dimensions.
- Consider the $N \times D$ data matrix

$$S = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{pmatrix}.$$

- Let the minimum and maximum values for each feature x_j be given as

$$min(x_j) = \min_i \{x_{ij}\}$$

$$max(x_j) = \max_i \{x_{ij}\}$$

- The data hyperspace can be considered as a D-dimensional hyper-rectangle, defined as

$$R_D = \prod_{j=1}^{D} [min(x_j), max(x_j)] .$$

High-dimensional space (cont.)

- Hypercube
  - Assume the data is centered to have mean : $\mu = 0$.
  - Let m denote the largest absolute value in S.

$$m = \max_{j=1}^{D} \max_{i=1}^{N} \{|x_{ij}|\} .$$

  - The data hyperspace can be represented as a hypercube $H_D(l)$, centered at 0, with all sides of length $l = 2m$.

$$H_D(l) = \left\{ x = (x_1, \ldots, x_D)^T \mid \quad \forall i \quad x_i \in \left[-\frac{l}{2}, \frac{l}{2} \right] \right\} .$$

- Hypersphere
  - Assume the data is centered to have mean : $\mu = 0$.
  - Let r denote the largest magnitude among all points in S.

$$r = \max_{i} \{\|x_i\|\} .$$

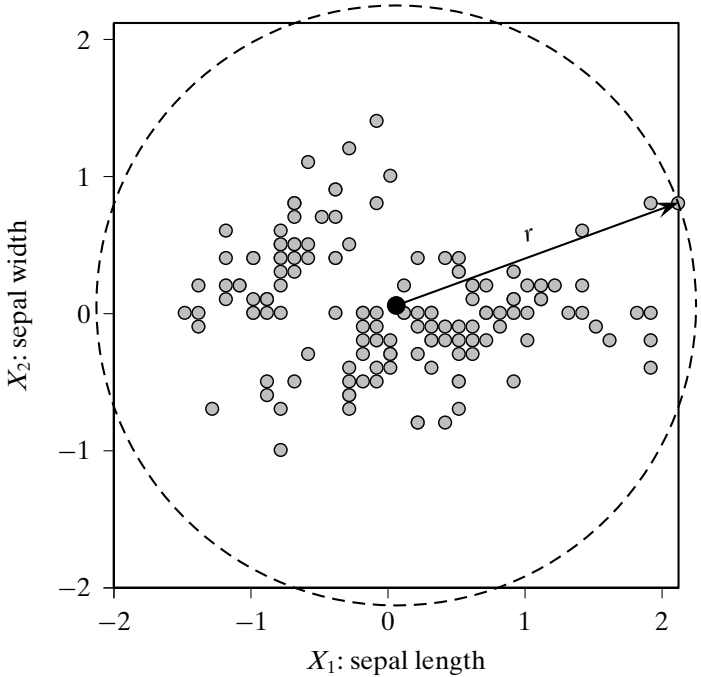
  - The data hyperspace can also be represented as a D-dimensional hyperball centered at 0 with radius r

$$B_D(r) = \{x \mid \|x_i\| \leq r\}$$

  - The surface of the hyperball is called a hypersphere, and it consists of all the points exactly at distance r from the center of the hyperball

$$S_D(r) = \{x \mid \|x_i\| = r\}$$

- Consider two features of Irish data set



- The volume of a hypercube with edge length l equals to

$$vol(H_D(l)) = l^D.$$

- The volume of a hyperball and its corresponding hypersphere equals to

$$vol(S_D(r)) = \left(\frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \right) r^D.$$

where gamma function for $\alpha > 0$ is defined as

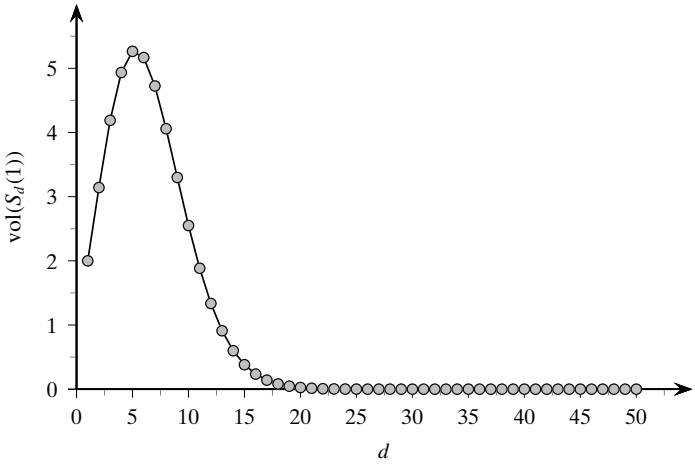
$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$$

- The surface area of the hypersphere can be obtained by differentiating its volume with respect to r

$$area(S_D(r)) = \frac{d}{dr} vol(S_D(r)) = \left(\frac{2\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}\right)} \right) r^{D-1}.$$

- An interesting observation about the hypersphere volume is that as dimensionality increases, the volume first increases up to a point, and then starts to decrease, and ultimately vanishes.
- For the unit hypersphere ($r = 1$),

$$\lim_{D\to\infty} vol(S_D(r)) = \lim_{D\to\infty} \left(\frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \right) r^D \to 0.$$



Hypersphere Inscribed Within Hypercube

- Consider the space enclosed within the largest hypersphere that can be accommodated within a hypercube.
- Consider a hypersphere of radius r inscribed in a hypercube with sides of length $2r$.
- The ratio of the volume of the hypersphere of radius r to the hypercube with side length $l = 2r$ equals to

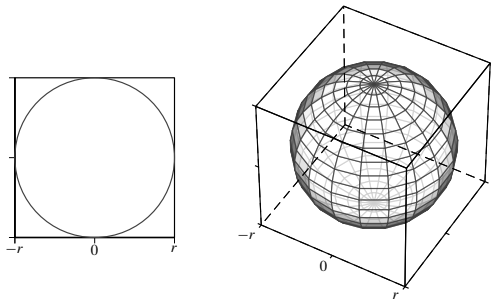
$$\frac{vol(S_2(r))}{vol(H_2(2r))} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} = 0.785$$

$$\frac{vol(S_3(r))}{vol(H_3(2r))} = \frac{\frac{4}{3}\pi r^3}{8r^3} = \frac{\pi}{6} = 0.524$$

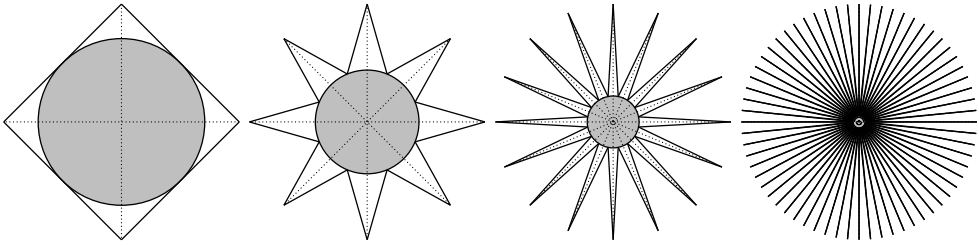
$$\lim_{D\to\infty} \frac{vol(S_D(r))}{vol(H_D(2r))} = \lim_{D\to\infty} \left(\frac{\pi^{\frac{D}{2}}}{2^D \Gamma\left(\frac{D}{2}+1\right)} \right) \to 0.$$

Hypersphere Inscribed within Hypercube

- Hypersphere inscribed inside a hypercube for two and three dimensions.



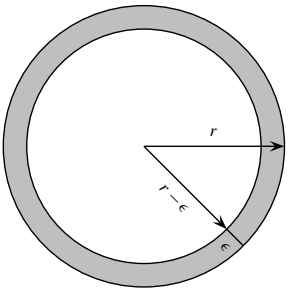
- Conceptual view of high-dimensional space for two, three, four, and higher dimensions.



In d dimensions there are 2^d **corners** and 2^{d-1} **diagonals**.

Volume of Thin Hypersphere Shell

- Consider the volume of a thin hypersphere shell of width ϵ bounded by an outer hypersphere of radius r, and an inner hypersphere of radius $r - \epsilon$.
- Volume of the thin shell equals to the difference between the volumes of the two bounding hyperspheres.



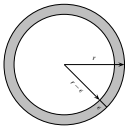
- Let $S_D(r, \epsilon)$ denote the thin hypershell of width ϵ. Its volume equals

$$vol(S_D(r, \epsilon)) = vol(S_D(r)) - vol(S_D(r - \epsilon)) = K_D r^D - K_D (r - \epsilon)^D$$

$$K_D = \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2} + 1\right)}$$

- Ratio of **the volume of the thin shell** to **the volume of the outer sphere** equals to

$$\frac{vol(S_D(r,\epsilon))}{vol(S_D(r))} = \frac{K_D r^D - K_D(r-\epsilon)^D}{K_D r^D} = 1 - \left(1 - \frac{\epsilon}{r}\right)^D$$



- For $r = 1$ and $\epsilon = 0.01$

$$\frac{vol(S_2(1,0.01)}{vol(S_2(1))} = 1 - \left(1 - \frac{0.01}{1}\right)^2 \simeq 0.02$$
$$\frac{vol(S_3(1,0.01)}{vol(S_3(1))} = 1 - \left(1 - \frac{0.01}{1}\right)^3 \simeq 0.03$$
$$\frac{vol(S_4(1,0.01)}{vol(S_4(1))} = 1 - \left(1 - \frac{0.01}{1}\right)^4 \simeq 0.04$$
$$\frac{vol(S_5(1,0.01)}{vol(S_5(1))} = 1 - \left(1 - \frac{0.01}{1}\right)^5 \simeq 0.05$$

- As D increases, in the limit we obtain

$$\lim_{D\to\infty} \frac{vol(S_D(r,\epsilon))}{vol(S_D(r))} = \lim_{D\to\infty} 1 - \left(1 - \frac{\epsilon}{r}\right)^D \to 1.$$

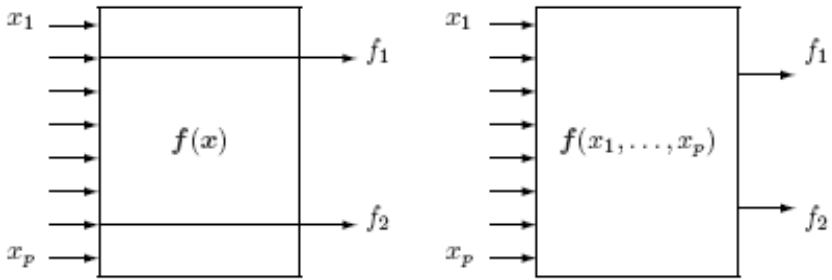
- Almost all of the volume of the hypersphere is contained in the thin shell as $D \to \infty$.

Volume of Thin Hypersphere Shell (cont.)

- **Almost all of the volume of the hypersphere** is contained in the **thin shell** as $D \to \infty$.
- This means that in high-dimensional spaces, **unlike in lower dimensions**, most of the **volume is concentrated around the surface (within ϵ) of the hypersphere**, and the **center is essentially void**.
- In other words, **if the data is distributed uniformly in the D-dimensional space**, then **all of the points essentially lie on the boundary of the space (which is a $D-1$ dimensional object)**.
- Combined with **the fact that most of the hypercube volume is in the corners**, we can observe that in **high dimensions, data tends to get scattered on the boundary and corners of the space**.
- As a consequence, **high-dimensional data can cause problems** for data mining and analysis, although in some cases high-dimensionality can help, for example, for nonlinear classification.
- It is important **to check whether the dimensionality can be reduced while preserving the essential properties of the full data matrix**. This can aid **data visualization** as well as **data mining**.

Dimensionality reduction methods

- There are two main methods for reducing the dimensionality of inputs
  - Feature selection: These methods select d ($d < D$) dimensions out of D dimensions and $D - d$ other dimensions are discarded.
  - Feature extraction: Find a new set of d ($d < D$) dimensions that are combinations of the original dimensions.



Feature selection methods

Feature selection methods

- Feature selection methods select d ($d < D$) dimensions out of D dimensions and $D - d$ other dimensions are discarded.
- Reasons for performing feature selection
  - Increasing the predictive accuracy of classifiers or regressors.
  - Removing irrelevant features.
  - Enhancing learning efficiency (reducing computational and storage requirements).
  - Reducing the cost of future data collection (making measurements on only those variables relevant for discrimination/prediction).
  - Reducing complexity of the resulting classifiers/regressors description (providing an improved understanding of the data and the model).
- Feature selection is not necessarily required as a pre-processing step for classification/regression algorithms to perform well.
- Several algorithms employ regularization techniques to handle over-fitting or averaging such as ensemble methods.

Feature selection methods

- Feature selection methods can be categorized into three categories.
  - Filter methods: These methods use the statistical properties of features to filter out poorly informative features.
  - Wrapper methods: These methods evaluate the feature subset within classifier/regressor algorithms. These methods are classifier/regressors dependent and have better performance than filter methods.
  - Embedded methods:These methods use the search for the optimal subset into classifier/regression design. These methods are classifier/regressors dependent.
- Two key steps in feature selection process.
  - Evaluation: An evaluation measure is a means of assessing a candidate feature subset.
  - Subset generation: A subset generation method is a means of generating a subset for evaluation.

Feature selection methods (Evaluation measures)

- Large number of features are not informative (irrelevant or redundant).
- Irrelevant features are features that don't contribute to a classification or regression rule.
- Redundant features are features that are strongly correlated.
- In order to choose a good feature set, we require a means of a measure to contribute to the separation of classes, either individually or in the context of already selected features. We need to measure relevancy and redundancy.
- There are two types of measures
  - Measures that relay on the general properties of the data.
    These assess the relevancy of individual features and are used to eliminate redundancy.
    All these measures are independent of the final classifier.
    These measures are inexpensive to implement but may not well detect the redundancy.
  - Measures that use a classification rule as a part of their evaluation.
    In this approach, a classifier is designed using the reduced feature set and a measure of classifier performance is employed to assess the selected features.
    A widely used measure is the error rate.

- The following measures relay on the general properties of the data.
  - Feature ranking: Features are ranked by a metric and those that fail to achieve a prescribed score are eliminated.
    Examples of these metrics are: Pearson correlation, mutual information, and information gain.
  - Interclass distance: A measure of distance between classes is defined based on distances between members of each class.
    Example of these metrics is: Euclidean distance.
  - Probabilistic distance: This is the computation of a probabilistic distance between class-conditional probability density functions, i.e. the distance between $p(x|C_1)$ and $p(x|C_2)$ (two-classes).
    Example of these metrics is: Chhernoff dissimilarity measure.
  - Probabilistic dependency: These measures are multi-class criteria that measure the distance between class-conditional probability density functions and the mixture probability density function for the data irrespective of the class, i.e. the distance between $p(x|C_i)$ and $p(x)$.
    Example of these metrics is: Joshi dissimilarity measure.

- Complete search: These methods guarantee to find the optimal subset of features according to some specified evaluation criteria.
  For example exhaustive search and branch and bound methods are complete.
- Best individual N: The simplest method is to assign a score to each feature and then select N top ranks features.
- Sequential search: In these methods, features are added or removed sequentially. These methods are not optimal, but are simple to implement and fast to produce results.
  1. Sequential forward selection: It is a bottom-up search procedure that adds new features to a feature set one at a time until the final feature set is reached.
  2. Generalized sequential forward selection: In this approach, at each time $r > 1$, features are added instead of a single feature.
  3. Sequential backward elimination: It is a top-down procedure that deletes a single feature at a time until the final feature set is reached.
  4. Generalized sequential backward elimination : In this approach, at each time $r > 1$ features are deleted instead of a single feature.

Feature extraction

- Let S consist of N points over D feature, i.e. it is an $N \times D$ matrix

$$S = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{pmatrix}.$$

- Each point $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})^T$ is a vector in D-dimensional space spanned by the D basis vectors $e_1, e_2, \dots, e_D$, e_i corresponds to i^{th} feature.
- The standard basis is an orthonormal basis for the data space: the basis vectors are pairwise orthogonal $e_i^T e_j = 0$, and have unit length $\|e_i\| = 1$.
- Given any other set of D orthonormal vectors $u_1, u_2, \dots, u_D$,with $u_i^T u_j = 0$ and $\|u_i\| = 1$ (or $u_i^T u_i = 1$), we can re-express each point x as the linear combination

$$x = a_1 u_1 + a_2 u_2 + \dots + a_D u_D.$$

- Let $a = (a_1, a_2, \dots, a_D)^T$, then we have $x = Ua$.
- U is the $D \times D$ matrix, whose i^{th} column comprises u_i.
- Matrix U is an orthogonal matrix, whose columns, the basis vectors, are orthonormal, that is, they are pairwise orthogonal and have unit length. This means that $U^{-1} = U^T$.

- Multiplying both sides of $x = Ua$ by U^T, results in

$$U^T x = U^T U a$$
$$a = U^T x$$

Example

- Let $e_1 = (1, 0, 0)^T$, $e_2 = (0, 1, 0)^T$, and $e_3 = (0, 0, 1)^T$ be the standard basis vectors
- Let $u_1 = (-0.39, 0.089, -0.916)^T$, $u_2 = (-0.639, -0.742, 0.200)^T$, and $u_3 = (-0.663, 0.664, 0.346)^T$ be the new basis vectors.
- The new coordinate of the centered point $x = (-0.343, -0.754, 0.241)^T$ can be computed as

$$a = U^T x = \begin{pmatrix} -0.390 & 0.089 & -0.916 \\ 0.639 & -0.742 & 0.200 \\ -0.663 & 0.664 & 0.346 \end{pmatrix} \begin{pmatrix} -0.343 \\ -0.754 \\ 0.241 \end{pmatrix} = \begin{pmatrix} -0.154 \\ 0.828 \\ -0.190 \end{pmatrix}.$$

- There are infinite choices for the set of orthonormal basis vectors, one natural question is whether there exists an optimal basis, for a suitable notion of optimality.
- We are interested in finding the optimal d-dimensional representation of S, with $d \ll D$.
- In other words, given a point x, and assuming that the basis vectors have been sorted in decreasing order of importance, we can truncate its linear expansion to just d terms, to obtain

$$x' = a_1 u_1 + a_2 u_2 + \ldots + a_d u_d = U_d a_d.$$

- Since we have $a_d = U_d^T x$, restricting it to the first d terms, we get $a_d = U_d^T x$.
- Hence, we obtain $x' = U_d U_d^T x = P_d x$.
- Projection error equals to $\epsilon = x - x'$.
- By substituting, we conclude that x' and ϵ are orthogonal vectors: $x'^T \epsilon = 0$.

Example

- Let $u_1 = (-0.39, 0.089, -0.916)^T$. The new coordinate of the centered point $x = (-0.343, -0.754, 0.241)^T$ using the first basis vector can be computed as

$$x' = a_1 u_1 = -0.154 u_1 = \begin{pmatrix} 0.060 & -0.014 & 0.141 \end{pmatrix}^T$$

- Projection of x on u_1 can be obtained directly from

$$P_1 = U_1 U_1^T = \begin{pmatrix} -0.390 \\ 0.089 \\ -0.916 \end{pmatrix} \begin{pmatrix} -0.390 & 0.089 & -0.916 \end{pmatrix} = \begin{pmatrix} 0.152 & -0.035 & 0.357 \\ -0.035 & 0.008 & -0.082 \\ 0.357 & -0.082 & 0.839 \end{pmatrix}$$

- The new coordinate equals to $x' = P_1 x = \begin{pmatrix} 0.060 & -0.014 & 0.141 \end{pmatrix}^T$
- Projection error equals to $\epsilon = a_2 u_2 + a_3 u_3 = x - x' = P_1 x = \begin{pmatrix} -0.40 & -0.74 & 0.10 \end{pmatrix}^T$
- Vectors ϵ and x' are orthogonal

$$x'\epsilon = \begin{pmatrix} 0.060 & -0.014 & 0.141 \end{pmatrix} \begin{pmatrix} -0.40 \\ -0.74 \\ 0.10 \end{pmatrix} = 0$$

- In feature extraction, we are interested to find a new set of k ($k \ll D$) dimensions that are combinations of the original D dimensions.
- Feature extraction methods may be supervised or unsupervised.
- Examples of feature extraction methods
  - Principal component analysis (PCA)
  - Factor analysis (FA)s
  - Multi-dimensional scaling (MDS)
  - ISOMap
  - Locally linear embedding
  - Linear discriminant analysis (LDA)

Feature extraction methods

Feature extraction methods

Principal component analysis

- PCA project $D-$dimensional input vectors to $k-$dimensional input vectors via a linear mapping with minimum loss of information. Dimensions are combinations of the original D dimensions.
- The problem is to find a matrix W such that the following mapping results in the minimum loss of information.

$$Z = W^T X$$

- PCA is unsupervised and tries to maximize the variance.
- The principle component is w_1 such that the sample after projection onto w_1 is most spread out so that the difference between the sample points becomes most apparent.
- For uniqueness of the solution, we require $\|w_1\| = 1$,
- Let $\Sigma = Cov(X)$ and consider the principle component w_1, we have

$$\begin{aligned} z_1 &= w_1^T x \\ Var(z_1) &= E[(w_1^T x - w_1^T \mu)^2] = E[(w_1^T x - w_1^T \mu)(w_1^T x - w_1^T \mu)^T] \\ &= E[w_1^T (x-\mu)(x-\mu)^T w_1] = w_1^T E[(x-\mu)(x-\mu)^T] w_1 = w_1^T \Sigma w_1 \end{aligned}$$

- The mapping problem becomes

$$w_1 = \underset{w}{argmax} \quad w^T \Sigma w \qquad \text{subject to } w_1^T w_1 = 1.$$

- Writing this as Lagrange problem, we have

$$\underset{w_1}{maximize} \quad w_1^T \Sigma w_1 - \alpha(w_1^T w_1 - 1)$$

- Taking derivative with respect to w_1 and setting it equal to 0, we obtain

$$2\Sigma w_1 = 2\alpha w_1 \quad \Rightarrow \quad \Sigma w_1 = \alpha w_1$$

- Hence w_1 is eigenvector of Σ and α is the corresponding eigenvalue.
- Since we want to maximize $Var(z_1)$, we have

$$Var(z_1) = w_1^T \Sigma w_1 = \alpha w_1^T w_1 = \alpha$$

- Hence, we choose the eigenvector with the largest eigenvalue, i.e. $\lambda_1 = \alpha$.

- Let $\epsilon_i = x_i - x_i'$ denote the error vector. The MSE equals to

$$\begin{aligned} MSE(W) &= \frac{1}{N}\sum_{i=1}^{N}\|\epsilon_i\|^2 \\ &= \sum_{i=1}^{N}\frac{\|x_i\|^2}{N} - W^T\Sigma W \\ &= Var(S) - W^T\Sigma W. \end{aligned}$$

- Since $var(S)$, is a constant for a given dataset S, the vector W that minimizes $MSE(W)$ is thus the same one that maximizes the second term,

$$\begin{aligned} MSE(W) &= Var(S) - W^T\Sigma W \\ &= Var(S) - \lambda_1 \end{aligned}$$

- Example: Let

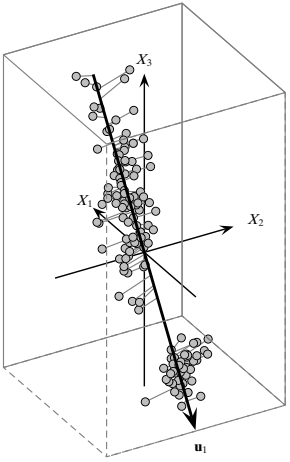
$$\Sigma = \begin{pmatrix} 0.681 & -0.039 & 1.265 \\ -0.039 & 0.187 & -0.320 \\ 1.265 & -0.320 & 3.092 \end{pmatrix}$$

The largest eigenvalue of Σ equals to $\lambda = 3.662$ and the corresponding eigenvector equals to $w_1 = (-0.390, 0.089, -0.916)^T$

- The variance of S equals $var(S) = 0.681 + 0.187 + 3.092 = 3.96$.
- MSE equals to

$$\begin{aligned} MSE(W_1) &= var(S) - \lambda_1 \\ &= 3.96 - 3.662 = 0.298 \end{aligned}$$

- Principle component



- The second principal component, w_2, should also
  - maximize variance
  - be unit length
  - orthogonal to w_1 (z_1 and z_2 must be uncorrelated)
- The mapping problem for the second principal component becomes

$$w_2 = \underset{w}{argmax} \quad w^T \Sigma w \qquad \text{subject to } w_2^T w_2 = 1 \text{ and } w_2^T w_1 = 0.$$

- Writing this as Lagrange problem, we have

$$\underset{w_2}{maximize} \quad w_2^T \Sigma w_2 - \alpha(w_2^T w_2 - 1) - \beta(w_2^T w_1 - 0)$$

- Taking derivative with respect to w_2 and setting it equal to 0, we obtain

$$2\Sigma w_2 - 2\alpha w_2 - \beta w_1 = 0$$

- Pre-multiply by w_1^T, we obtain

$$2w_1^T \Sigma w_2 - 2\alpha w_1^T w_2 - \beta w_1^T w_1 = 0$$

- Note that $w_1^T w_2 = 0$ and $w_1^T \Sigma w_2 = (w_2^T \Sigma w_1)^T = w_2^T \Sigma w_1$ is a scaler.

- Since $\Sigma w_1 = \lambda_1 w_1$, therefore we have

$$w_1^T \Sigma w_2 = w_2^T \Sigma w_1 = \lambda_1 w_2^T w_1 = 0$$

- Then $\beta = 0$ and the problem reduces to

$$\Sigma w_2 = \alpha w_2$$

- This implies that w_2 should be the eigenvector of Σ with the second largest eigenvalue $\lambda_2 = \alpha$.
- Let the projected dataset be denoted by A.
- The total variance for A is given as

$$var(A) = \lambda_1 + \lambda_2$$

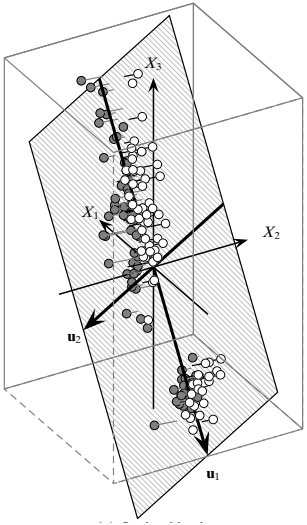
- Let $\epsilon_i = x_i - x_i'$ denote the error vector. The MSE equals to

$$MSE(W) = Var(S) - var(A).$$

- The MSE objective is minimized when total projected variance $var(A)$ is maximized

$$MSE(W) = Var(S) - \lambda_1 - \lambda_2$$

- Example: Two first Principle components



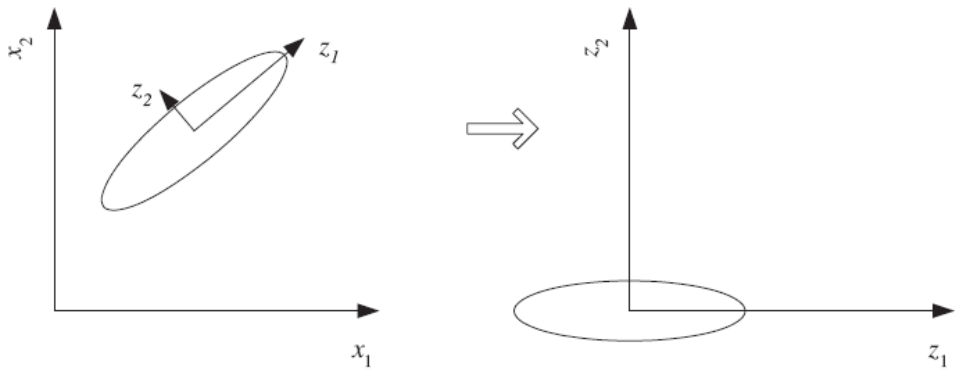
- We are now interested in the best k-dimensional ($k \ll D$) approximation to S.
- Assume that we have already computed the first $j-1$ principal components or eigenvectors, $w_1, w_2, \ldots, w_{j-1}$, corresponding to the $j-1$ largest eigenvalues of Σ
- To compute the j^{th} new basis vector w_j, we have to ensure that it is normalized to unit length, that is, $w_j^T w_j = 1$, and is orthogonal to all previous components w_i (for $i \in [1, j)$).
- The projected variance along w_j is given as $w_j^T \Sigma w_j$
- Combined with the constraints on w_j, this leads to the following maximization problem with Lagrange multipliers:

$$\underset{w_j}{\textit{maximize}} \quad w_j^T \Sigma w_j - \alpha(w_j^T w_j - 1) - \sum_{i=1}^{j-1} \beta_i (w_i^T w_j - 0)$$

- Solving this, results in $\beta_i = 0$ for all $i < j$.
- To maximize the variance along w_j, we use the j^{th} largest eigenvalue of Σ.

- In summary, to find the best k-dimensional approximation to Σ, we compute the eigenvalues of Σ.
- Because Σ is positive semidefinite, its eigenvalues must all be non-negative, and we can thus sort them in decreasing order
  $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{j-1} \geq \lambda_j \geq \ldots \geq \lambda_D \geq 0$
- We then select the k largest eigenvalues, and their corresponding eigenvectors to form the best k-dimensional approximation.
- Since Σ is symmetric, for two different eigenvalues, their corresponding eigenvectors are orthogonal. (Show it)
- If Σ is positive definite ($x^T \Sigma x > 0$ for all non-null vector x), then all its eigenvalues are positive.
- If Σ is singular, its rank is k ($k < D$) and $\lambda_i = 0$ for $i = k+1, \ldots, D$.

- Define
$$Z = W^T(X - \mathbf{m})$$
- Then k columns of W are the k leading eigenvectors of S (the estimator of Σ).
- $\mathbf{m}$ is the sample mean of X.
- Subtracting $\mathbf{m}$ from X before projection centers the data on the origin.



- How to normalize variances?

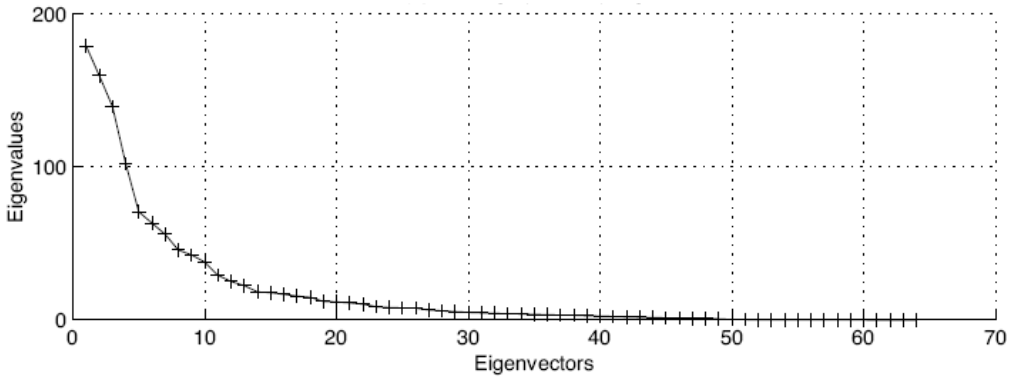
Principal component analysis (example)

25 randomly chosen 64×64 pixel images

Mean and the first three principal components

Principal component analysis (selecting k)

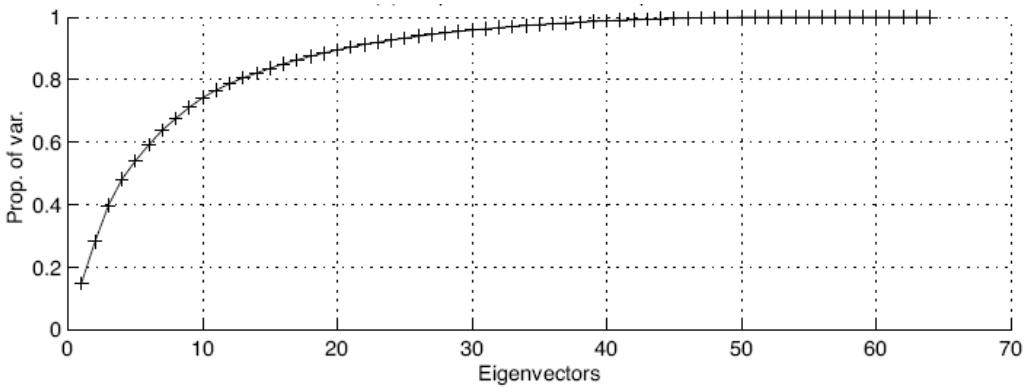
- How to select k?
- Since all eigenvalues are positive and $|S| = \prod_{i=1}^{D} \lambda_i$ is small, then some eigenvalues have little contribution to the variance and may be discarded.
- Scree graph is the plot of variance as a function of the number of eigenvectors.



- How to select k?
- We select the leading k components that explain more than for example 95% of the variance.
- The proportion of variance (POV) is

$$POV = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{D} \lambda_i}$$

- By visually analyzing it, we can choose k.



Principal component analysis (selecting k)

- How to select k?
- Another possibility is to ignore the eigenvectors whose corresponding eigenvalues are less than the average input variance (why?).
- In the pre-processing phase, it is better to pre-process data such that each dimension has mean $\mathbf{0}$ and unit variance(why and when?).
- Question: Can we use the correlation matrix instead of covariance matrix? Drive solution for PCA.

Principal component analysis (conclusions)

- PCA is sensitive to outliers. A few points distant from the center have large effect on the variances and thus eigenvectors.
- Question: How can use the robust estimation methods for calculating parameters in the presence of outliers?
  - A simple method is discarding the isolated data points that are far away.
- Question: When D is large, calculating, sorting, and processing of S may be tedious. Is it possible to calculate eigenvectors and eigenvalues directly from data without explicitly calculating the covariance matrix?

Feature extraction methods

Kernel principal component analysis

- PCA can be extended to find nonlinear directions in the data using kernel methods.
- Kernel PCA finds the directions of most variance in the feature space instead of the input space.
- Linear principal components in the feature space correspond to nonlinear directions in the input space.
- Using kernel trick, all operations can be carried out in terms of the kernel function in input space without having to transform the data into feature space.
- Let ϕ correspond to a mapping from the input space to the feature space.
- Each point in feature space is given as the image of $\phi(x)$ of the point x in the input space.
- In feature space, we can find the first kernel principal component W_1 ($W_1^T W_1 = 1$) by solving

$$\Sigma_\phi W_1 = \lambda_1 W_1$$

- Covariance matrix Σ_ϕ in feature space is equal to

$$\Sigma_\phi = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i)\phi(x_i)^T$$

- We assume that the points are centered.

- Plugging Σ_ϕ into $\Sigma_\phi W_1 = \lambda_1 W_1$, we obtain

$$\begin{aligned}
\left(\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T\right)W_1 &= \lambda_1 W_1 \\
\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\left(\phi(x_i)^T W_1\right) &= \lambda_1 W_1 \\
\sum_{i=1}^{N}\left(\frac{\phi(x_i)^T W_1}{N\lambda_1}\right)\phi(x_i) &= W_1 \\
\sum_{i=1}^{N} c_i \phi(x_i) &= W_1
\end{aligned}$$

- $c_i = \frac{\phi(x_i)^T W_1}{N\lambda_1}$ is a scalar value

- Now substitute $\sum_{i=1}^{N} c_i \phi(x_i) = W_1$ in $\Sigma_\phi W_1 = \lambda_1 W_1$, we obtain

$$\begin{aligned}
\left(\frac{1}{N} \sum_{i=1}^{N} \phi(x_i)\phi(x_i)^T \right) \left(\sum_{i=1}^{N} c_i \phi(x_i) \right) &= \lambda_1 \sum_{i=1}^{N} c_i \phi(x_i) \\
\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} c_j \phi(x_i)\phi(x_i)^T \phi(x_j) &= \lambda_1 \sum_{i=1}^{N} c_i \phi(x_i) \\
\sum_{i=1}^{N} \left(\phi(x_i) \sum_{j=1}^{N} c_j \phi(x_i)^T \phi(x_j) \right) &= N\lambda_1 \sum_{i=1}^{N} c_i \phi(x_i)
\end{aligned}$$

- Replacing $\phi(x_i)^T \phi(x_j)$ by $K(x_i, x_j)$

$$\sum_{i=1}^{N} \left(\phi(x_i) \sum_{j=1}^{N} c_j K(x_i, x_j) \right) = N\lambda_1 \sum_{i=1}^{N} c_i \phi(x_i)$$

- Multiplying with $\phi(x_k)^T$, we obtain

$$\begin{aligned}\sum_{i=1}^{N}\left(\phi(x_k)^T\phi(x_i)\sum_{j=1}^{N}c_jK(x_i,x_j)\right) &= N\lambda_1\sum_{i=1}^{N}c_i\phi(x_k)^T\phi(x_i)\\ \sum_{i=1}^{N}\left(K(x_k,x_i)\sum_{j=1}^{N}c_jK(x_i,x_j)\right) &= N\lambda_1\sum_{i=1}^{N}c_iK(x_k,x_j)\end{aligned}$$

- By some algebraic simplificatin, we obtain (do it)

$$K^2C = N\lambda_1 KC$$

- Multiplying by K^{-1}, we obtain

$$\begin{aligned}KC &= N\lambda_1 C\\ KC &= \eta_1 C\end{aligned}$$

- Weight vector C is the eigenvector corresponding to the largest eigenvalue η_1 of the kernel matrix K.

- Replacing $\sum_{i=1}^{N} c_i \phi(x_i) = W_1$ in constraint $W_1^T W_1 = 1$, we obtain

$$\begin{aligned} \sum_{i=1}^{N} \sum_{j=1}^{N} c_j c_j \phi(x_i)^T \phi(x_j) &= 1 \\ C^T K C &= 1 \end{aligned}$$

- Using $KC = \eta_1 C$, we obtain

$$\begin{aligned} C^T(\eta_1 C) &= 1 \\ \eta_1 C^T C &= 1 \\ ||C||^2 &= \frac{1}{\eta_1} \end{aligned}$$

- Since C is an eigenvector of K, it will have unit norm.
- To ensure that W_1 is a unit vector, multiply C by $\sqrt{\frac{1}{\eta_1}}$

- In general, we do not map input space to the feature space via ϕ, hence we cannot compute W_1 using

$$\sum_{i=1}^{N} c_i \phi(x_i) \quad = \quad W_1$$

- We can project any point $\phi(x)$ on to principal direction W_1

$$W_1^T \phi(x) = \sum_{i=1}^{N} c_i \phi(x_i)^T \phi(x) \quad = \quad \sum_{i=1}^{N} c_i K(x_i, x)$$

- When $x = x_i$ is one of the input points, we have

$$a_i = W_1^T \phi(x_i) = K_i^T C$$

where K_i is the column vector corresponding to the I^{th} row of K and a_i is the vector in the reduced dimension.

- If we sort the eigenvalues of K in decreasing order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$, we can obtain the j^{th} principal component.
- This shows that all computation are carried out using only kernel operations.

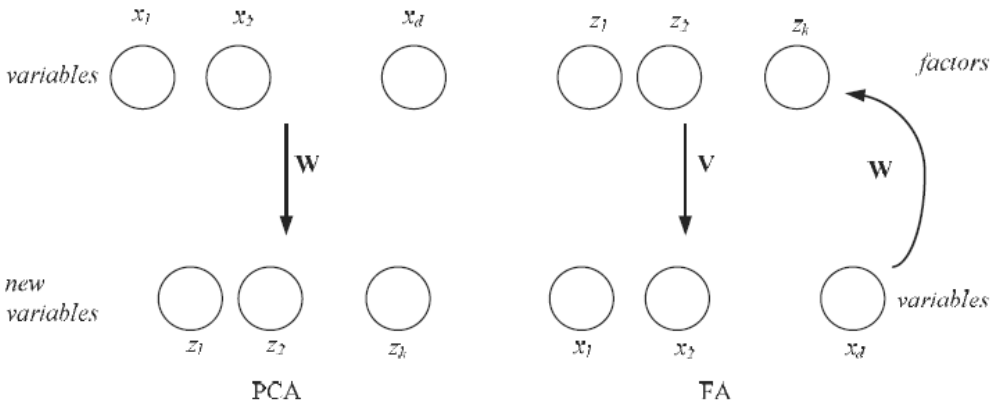
Feature extraction methods

Factor analysis

- In PCA, from the original dimensions x_i (for $i = 1, \ldots, D$), we form a new set of variables z_i that are linear combinations of x_i

$$Z = W^T(X - \mu)$$

- In factor analysis (FA), we assume that there is a set of unobservable, latent factors z_j (for $j = 1, \ldots, k$), which when acting in combination generate x.
- Thus the direction is opposite that of PCA.
- The goal is to characterize the dependency among the observed variables by means of a smaller number of factors.
- Suppose there is a group of variables that have high correlation among themselves and low correlation with all the other variables. Then there may be a single underlying factor that gave rise to these variables.
- FA, like PCA, is a one-group procedure and is unsupervised. The aim is to model the data in a smaller dimensional space without loss of information.
- In FA, this is measured as the correlation between variables.



There are two uses of factor analysis:

- It can be used for knowledge extraction when we find the loadings and try to express the variables using fewer factors.
- It can also be used for dimensionality reduction when $k < D$.

1. Sample x drawn from some unknown probability density $p(x)$ with $\mathbb{E}[x] = \mu$ and $Cov(x) = \Sigma$.
2. We assume that $\mu = 0$ and we can always add μ after projection.
3. In FA, each input dimension, x_i can be written as a weighted sum of $k < D$ factors, z_j plus the residual term.

$$x_i = v_{i1}z_1 + v_{i2}z_2 + \ldots + v_{ik}z_k + \epsilon_i$$

4. This can be written in vector-matrix form as

$$X = VZ + \epsilon_i$$

   V is the $D \times k$ matrix of weights, called factor loadings.
5. Factors are unit normals ($\mathbb{E}[z_j] = 0$, $Var(z_j) = 1$) and uncorrelated ($Cov(z_i, z_j) = 0$, $i \neq j$).
6. To explain what is not explained by factors, there is an added source (ϵ_i) for each input.
7. It is assumed that
   - Noise are zero-mean ($\mathbb{E}[\epsilon_i] = 0$) with unknown variance $Var(\epsilon_i) = \psi_i$.
   - Noise are uncorrelated among themselves ($Cov(\epsilon_i, \epsilon_i) = 0$, $i \neq j$).
   - Thus, $\Sigma_\epsilon = \mathbb{E}\left[\epsilon\epsilon^T\right] = diag[\psi_1, \psi_2, \ldots, \psi_D]$.
   - Noise are also uncorrelated with the factors, ($Cov(\epsilon_i, z_j) = 0$, $\forall i, j$).

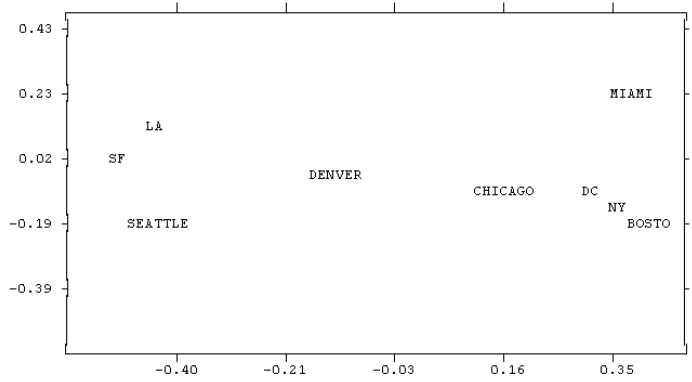
1. We have
$$\Sigma_x = \mathbb{E}\left[XX^T\right] = V\,\mathbb{E}\left[ZZ^T\right]V^T + \Sigma_\epsilon$$
2. Since factors are uncorrelated unit normals, hence $\mathbb{E}\left[ZZ^T\right] = I$
$$\boxed{\Sigma_x = VV^T + \Sigma_\epsilon}$$
3. If we have V, then
$$Z = WX$$
4. Post multiplying by X^T and taking expectations and using $\mathbb{E}\left[ZZ^T\right] = I$, we get
$$\begin{aligned}\mathbb{E}\left[ZX^T\right] &= \mathbb{E}\left[Z\left[(VZ)^T + \epsilon^T\right]\right] \\ &= \mathbb{E}\left[ZZ^TV^T\right] + \mathbb{E}\left[Z\epsilon^T\right] = V^T\end{aligned}$$
5. Also
$$\mathbb{E}\left[ZX^T\right] = W\,\mathbb{E}\left[XX^T\right] = W\Sigma_x$$
6. Hence, $V^T = W\Sigma_x$
$$W = V^T\Sigma_x^{-1}$$
7. By combining the above equations, we obtain
$$z = V^T\Sigma_x^{-1}x$$

Feature extraction methods

Multidimensional Scaling

- MDS is an approach mapping the original high dimensional space to a lower dimensional space preserving pairwise distances.
- Goal of Multidimensional scaling (MDS): Given pairwise dissimilarities, reconstruct a map that preserves distances.

		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0



- MDS is an approach mapping the original high dimensional space to a lower dimensional space preserving pairwise distances.
- MDS addresses the problem of constructing a configuration of N points in Euclidean space by using information about the distances between the N patterns.
- Given a collection of not necessarily Euclidean distances d_{ij} between pairs of points $\{x_1, \ldots, x_N\}$.
- Let D be an $N \times N$ distance matrix for the input space.
- Given a matrix D, MDS attempts to find N points $z_1, \ldots, z_N$ in k dimensions, such that if $\hat{d}_{ij}$ denotes the Euclidean distance between z_i and z_j , then $\hat{D}$ is similar to D.
- MDS minimizes

$$\min_z \sum_{i=1}^{N} \sum_{j=1}^{N} \left(d_{ij} - \hat{d}_{ij} \right)^2$$

- Now, the objective function of MDS can be reduced to

$$\min_{z} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(x_i^T x_j - z_i^T z_j \right)^2$$

- MDS algorithm
  1. Build a Gram matrix of inner products $G = XX^T$
  2. Find the top k eigenvectors of G : $\psi_1, \ldots, \psi_k$ with the top k eigenvalues $\lambda_1 \ldots, \lambda_k$. Let $\Lambda = diag(\lambda_1 \ldots, \lambda_k)$.
  3. Calculate

$$\boxed{Z = \Lambda^{\frac{1}{2}} diag(\lambda_1 \ldots, \lambda_k)^T}$$

- When Euclidean distance is used, MDS and PCA produce the same results.
- But, the distances need not be based on Euclidean distances and can represent many types of dissimilarities between objects.

Feature extraction methods

Locally Linear Embedding

- Locally linear embedding (LLE) recovers global nonlinear structure from locally linear fits.
- The idea is that each point can be approximated as a weighted sum of its neighbors.
- The neighbors either defined using a given number of neighbors (n) or distance threshold (ϵ).
- Let x^r be an example in the input space and its neighbors be $x^s_{(r)}$. We find weights in such a way that minimize the following objective function.

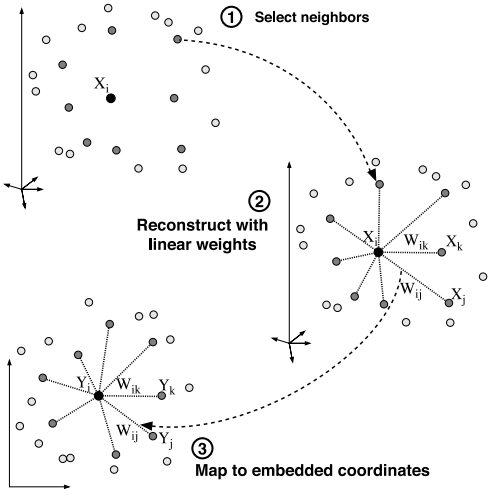
$$E[W|x] = \sum_{r=1}^{N} \left\| x^r - \sum_s w_{rs} x^s_{(r)} \right\|^2$$

- The idea in LLE is that the reconstruction weights w_{rs} reflect the intrinsic geometric properties of the data is also valid for the new space.
- The first step of LLE is to find w_{rs} in such a way that minimize the above objective function subject to $\sum_s w_{rs} = 1$.

- The second step of LLE is to keep w_{rs} fixed and construct the new coordinates Y in such a way that minimize the following objective function.

$$\mathbb{E}[Y|W] = \sum_{r=1}^{N} \left\| y^r - \sum_s w_{rs} y^s_{(r)} \right\|^2$$

in such a way that $Cov(Y) = I$ and $\mathbb{E}[Y] = 0$.



S. T. Roweis and L. K. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science, Vol. 290, No. 22, pp. 2323-2326, Dec. 2000.

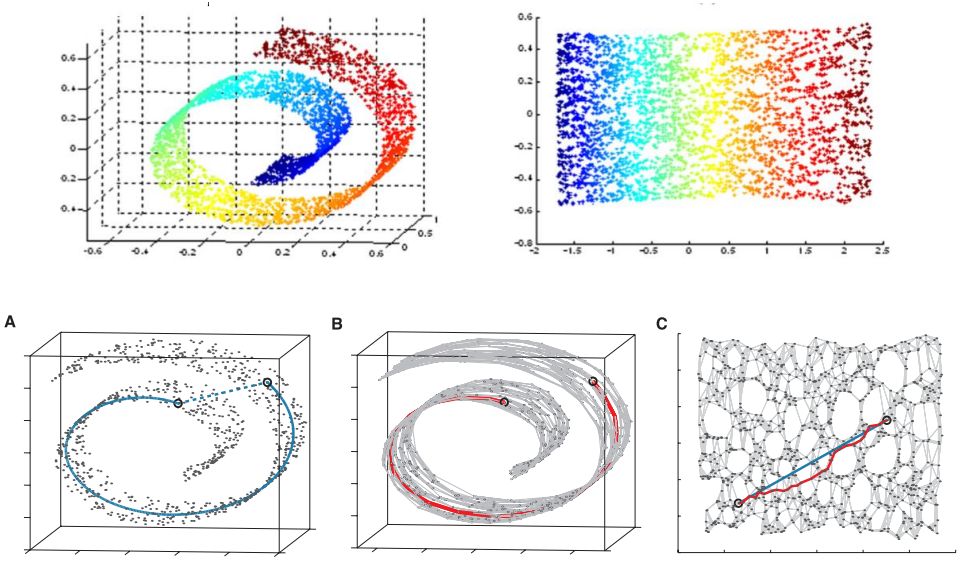
Feature extraction methods

Isomap

- Isomap is a technique similar to LLE for providing a low dimensional representation of a high dimensional data set.
- Isomap differs in how it assesses similarity between objects and in how the low dimensional mapping is constructed.
- Isomap is a nonlinear generalization of classical MDS.
- The main idea is to perform MDS, not in the input space, but in the geodesic space of the nonlinear data manifold.
- The geodesic distances represent the shortest paths along the curved surface of the manifold measured as if the surface were flat.
- The geodesic distances can be computed with e.g. the Floyd Warshall algorithm.
- Isomap then applies MDS to the geodesic distances.

- We start with data points in high dimensional space, lying near some manifold
- For each data point i we find the points j on manifold within some Euclidean distance $d(i,j) \leq \epsilon$.
- We construct a graph on the manifold with an edge between i and j if $d(i,j) \leq \epsilon$.
- We find the shortest path $d_G(i,j)$ between points i and j on the graph.
- Finally, we apply classical MDS to the distances $d_G(i,j)$.

Isomap (geodesic distance)



Reference:

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science, Vol. 290, No. 22, pp. 2319-2323, Dec. 2000

Feature extraction methods

Linear discriminant analysis

Linear discriminant analysis

- Linear discriminant analysis (LDA) is a supervised method for dimensionality reduction for classification problems.
- This method has been discussed in classification.

Reading

1. Section 12.1 of Pattern Recognition and Machine Learning Book (**bishop2006**).
2. Chapter 12 & 14.4 of Machine Learning: A probabilistic perspective (**mur2012**).
3. Chapter 20 of Probabilistic Machine Learning: An introduction (**mur2022**).

References i

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill.

Questions?