Feature Selection - Feature Extraction

Preprocessing

Why ?

Given some data, we often wish to perform preprocessing in order to:

1. transform it to a format that our algorithms can take as input
2. reduce time complexity: Less computation
3. reduce space complexity: Less parameters
4. decouple predictors
5. make it have properties (0-mean, unit variance, sparseness,...)
6. More interpretable: simpler explanation
7. Data visualization (structure, groups, outliers, etc)

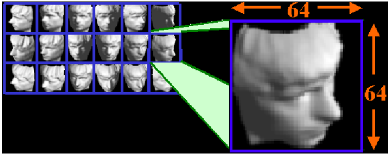
Examples

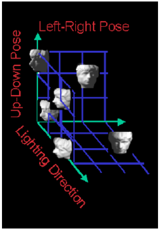
The curse of dimensionality

- Image data: each pixel of an image
- Genomic data: expression levels of genes (Several thousand features)
- Text categorization: frequencies of phrases in a document or in a web page (More than ten thousand features)

Intuition...

EXAMPLES





- Every pixel?
- Perceptually meaningful structure? (Up-down pose, Left-right pose, Lighting direction)

$\Rightarrow$ Reduction of the high-dimensional inputs to an intrinsically 3D manifold.

Feature Selection vs Extraction

Feature selection

Choosing $k < d$ important features, ignoring the remaining $d - k$
$\Rightarrow$ Subset selection algorithms

Feature extraction

Project the original $x_i, 1 \leq i \leq d$ dimensions to new $k < d$ dimensions, $z_i, 1 \leq i \leq k$
$\Rightarrow$ Discover low dimensional representations (smooth manifold) for data in high dimension.
$\Rightarrow$ Manifold Learning

But before...

Data has to be prepared

- Missing values → Imputation
- Outliers
- Data format
- Numerical / categorical → One hot-encoding
- ...

INTRODUCTION
- Preprocessing

LINEAR FEATURE EXTRACTION
- Principal Component Analysis
- Multidimensional Scaling
- Linar Discriminant Analysis

NONLINEAR FEATURE EXTRACTION
- Introduction
- ISOMAP
- Local Linear Embedding
- Laplacian Eigenmaps
- Locality Preseving Projection

PCA - Definition

One standard method for decoupling and dimensionality reduction of continuous data is Principal Component Analysis (PCA)

- Find a low-dimensional space such that when x is projected there, information loss is minimized.
- The projection of x on the direction of w is: $z = w^T x$
- Find w such that $\mathbb{V}(z)$ is maximized

$$
\begin{aligned}
\mathbb{V}(z) &= \mathbb{V}(w^T x) = \mathbb{E}\left((w^T x - w^T \mu)^2\right) \\
&= \mathbb{E}\left((w^T x - w^T \mu)(w^T x - w^T \mu)\right) \\
&= \mathbb{E}\left(w^T (x-\mu)(x-\mu)^T w\right) \\
&= w^T \mathbb{E}\left((x-\mu)(x-\mu)^T\right) w \\
&= w^T \Sigma w
\end{aligned}
$$

Principal components

First PC

Maximize $\mathbb{V}(z)$ subject to $\|w\| = 1 \Rightarrow \max_u u^T \Sigma u - \alpha(u^T u - 1)$ $\Sigma u = \alpha u \Rightarrow u$ eigenvector of Σ
Choose u with the largest eigenvalue for $\mathbb{V}(z)$ to be maximized

Second PC

Deflation or $\mathbb{V}(z_2)$ subject to $\|w\| = 1$ and orthogonal to u

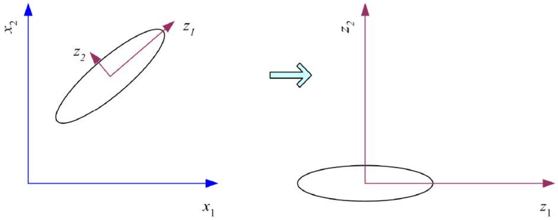
$$\max_w w^T \Sigma w - \alpha(w^T w - 1) - \beta(w^T u)$$

$\Sigma w = \alpha w \Rightarrow w$ eigenvector of Σ
continue k times

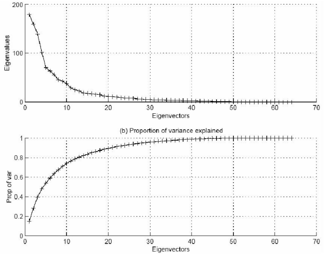
What PCA does

$$z = W^T(x - m)$$

where $W_{.,j}$ is the j^{th} eigenvector of Σ, and m is the sample mean.



Choice of k



$Sp(\Sigma) = \lambda_1 \geq \lambda_2 \cdots$

POV: proportion of variance.

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

- Typically, k/ POV>threshold
- Scree graph plot of POV, stop at elbow

```python
from sklearn.decomposition import PCA
X = ...
pca = PCA(n_components=2)
pca.fit(X)
PCA(n_components=2)
print(pca.explained_variance_ratio_)
```

Multidimensional Scaling - Definition

Definition

Given pairwise distances between N points, $d_{ij}, i, j \in \{1 \cdots N\}$, place on a low dimensional map such as distances are preserved.

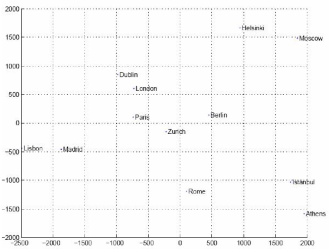
Sammon stress

$$z = g(x|\theta)$$

Find θ minimizing

$$\mathbb{E}(\theta|X) = \sum_{r,s} \frac{(\|z^r - z^s\| - \|x^r - x^s\|)^2}{\|x^r - x^s\|^2}$$

EXAMPLE

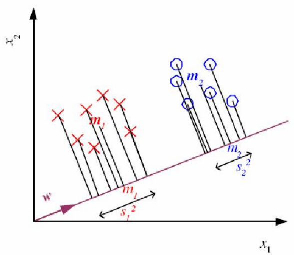



```python
from sklearn.manifold import MDS
X = ..
embedding = MDS(n_components=2)
X2 = embedding.fit_transform(X)
```

LDA - Definition

Definition - LDA or Fisher Discriminant Analysis

Find a low-dimensional space such that when x is projected, classes are well-separated.



$$\max_w \frac{m_1 - m_2}{s_1^2 + s_2^2}$$

- $$m_i = \frac{\sum_t w^T x_t r_t}{\sum_t r_t}$$
- $$s_i = \sum_t r_t \left(w^T x_t - m_i\right)^2$$

Data Scattering

$$\max_{w} \frac{m_1 - m_2}{s_1^2 + s_2^2}$$

Inter class

$$\begin{aligned}(m_1 - m_2)^2 &= \left(w^T\mu_1 - w^T\mu_2\right)^2 \\ &= w^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = w^T S_B w\end{aligned}$$

where $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$

$$\max_w \frac{m_1 - m_2}{s_1^2 + s_2^2}$$

Intra class

$$\begin{aligned} s_1^2 &= \sum_t r_t \left(w^T x_t - m_1\right)^2 \\ &= \sum_t r_t w^T (x_t - m_1)(x_t - m_1)^T w = w^T S_1 w \end{aligned}$$

where $S_1 = \sum_t r_t (x_t - m_1)(x_t - m_1)^T$

$s_1^2 + s_2^2 = w^T S_W w$, $S_W = S_1 + S_2$

Fisher's discriminant

Find w maximizing

$$\frac{w^T S_B w}{w^T S_W w}$$

Solutions

- LDA: $w = c.S_W^{-1}(m_1 - m_2)$
- Parametric: $w = \Sigma^{-1}(\mu_1 - \mu_2)$, when $p(x|C_i) \approx \mathcal{N}(\mu_i, \Sigma)$

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
X,y = ..
lda = LinearDiscriminantAnalysis(n_components=2)
X2 = lda.fit(X, y).transform(X)
```

What about the multiple class case ($C > 2$) ?

scattering

Inter class: $S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T \quad m = \frac{1}{C}\sum_{i=1}^{C}\mu_i$

Intra class: $S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} r_{t,i}\left(x_t - m_i\right)\left(x_t - m_i\right)^T$

Definition

$$\max_w \frac{W^T S_B W}{W^T S_W W}$$

- The largest eigenvectors of $S_W^{-1} S_B$
- Maximum rank of $C - 1$

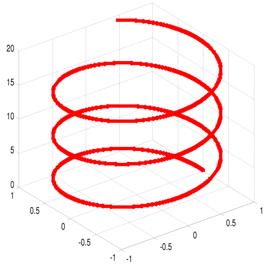
INTRODUCTION
- Preprocessing

LINEAR FEATURE EXTRACTION
- Principal Component Analysis
- Multidimensional Scaling
- Linar Discriminant Analysis

NONLINEAR FEATURE EXTRACTION
- Introduction
- ISOMAP
- Local Linear Embedding
- Laplacian Eigenmaps
- Locality Preseving Projection

Deficiencies of Linear Methods

Data may not be best summarized by linear combination of features.
Example: PCA cannot discover 1D structure of a helix



ISOMAP - Algorithm

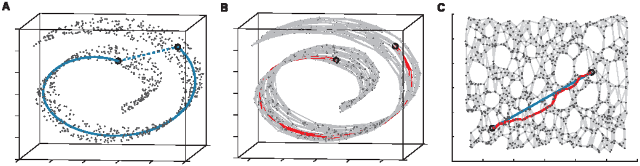
Seminal Paper

J. B. Tenenbaum, V. de Silva and J. C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science 290 (5500): 2319-2323, 22 December 2000

1. Constructing neighbourhood graph G
2. $\forall$ pair of points in G: shortest path distances $\approx$ geodesic distances.
3. Use MDS with geodesic distances.

ISOMAP - Example

- Construction of the neighbourhood graph G (K- nearest neighborhood (K=7)). D_G: 1000$\times$1000 (Euclidean) distance matrix (fig A)
- shortest paths in G: D_G:1000$\times$1000 geodesic distance matrix of two arbitrary points along the manifold (fig B)
- Embedding G in $\mathbb{R}^d$ using MDS: Find a d-D Euclidean space preserving pairwise distances (fig C)



ISOMAP - PROS AND CONS

Advantages

- Nonlinear
- Globally optimall low-dimensional Euclidean representation even though input space is highly folded, twisted, or curved.
- Guarantee asymptotically to recover the true dimensionality.

Disadvantages

- May not be stable, depends on the topology of the manifold
- asymptotically recover geometric structure of nonlinear manifolds
  - N high: pairwise distances $\approx$ geodesics, but costly
  - N small: geodesic distances very inaccurate
- Distance matrix is dense $\Rightarrow$ does not scale to large datasets

$\rightarrow$ Landmark Isomap proposed to overcome this problem

```python
from sklearn.manifold import Isomap
X = ...
iso = Isomap(n_components=2)
X2= iso.fit_transform(X)
```

LLE - Algorithm

Seminal Paper

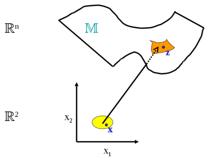
Sam T. Roweis and Lawrence K. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science 22:Vol. 290 no. 5500 pp. 2323-2326, 2000

ISOMAP vs. LLE

Local Linear Embedding $\Rightarrow$ local approach $\Rightarrow$ The resulting matrix is sparse...Apply efficient sparse matrix solvers

Characterictics of a Manifold

- Locally M is a linear patch
- how to combine all local patches together?



LLE - Algorithm

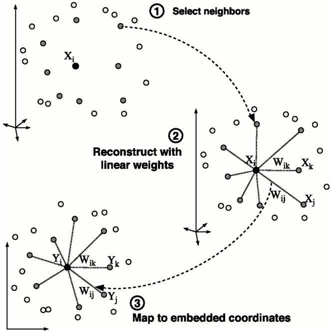
Assumption: manifold is roughly linear when viewed locally
Approximation error can be made small:

$$Min_W \|x_i - \sum_{j=1}^{k} w_{ij} x_j\|^2$$

1. W: a linear representation of every data point by its neighbors. This is an intrinsic geometrical property of the manifold
2. A good projection should preserve this local geometric property as much as possible

LLE - Algorithm

- We expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold.
- Each point can be written as a linear combination of its neighbors. The weights chosen to minimize the reconstruction error.



LLE - Algorithm

Optimal weights

The weights that minimize the reconstruction errors are invariant to rotation, rescaling and translation of the data points.

- Invariance to translation is enforced by adding the constraint that the weights sum to one.
- The weights characterize the intrinsic geometric properties of each neighborhood.

Local geometry is preserved

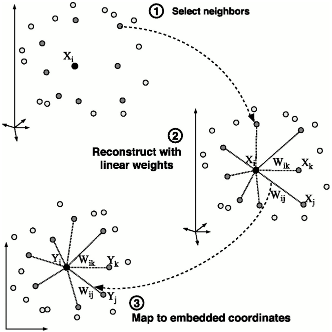
The same weights that reconstruct the data points in D dimensions should reconstruct it in the manifold in d dimensions.

LLE - Algorithm

Low-dimensional embedding
$Y \in \mathcal{M}_{d,N}(\mathbb{R})$

$$\min_{Y} \sum_{i=1}^{N} \|Y_{.,i} - YW_{i,.}\|^2$$

Use the same weights from the original space



LLE - Constrained LS Problem

Optimization

Compute the optimal weight for each point individually:

$$\|x_i - \sum_{j=1}^{k} w_{ij} x_j\|^2 = \|\sum_{j=1}^{k} w_{ij}(x_i - x_j)\|^2 = \sum_{j=1}^{k}\sum_{k} w_{ij} w_{ik} C_{jk}$$

where $C_{jk} = (x_i - x_j)^T (x_i - x_k)$
Can be minimized using a Lagrange multiplier for $\sum_j w_{ij} = 1$

Solution

$$w_{ij} = \frac{\sum_k C_{jk}^{-1}}{\sum_{lm} C_{lm}^{-1}}$$

Local Linear Embedding

LLE - Space

- $Y_{.,i} \in \mathbb{R}^k$: projected vector for X_i
- The geometrical property is best preserved if

$$E(Y) = \sum_i \|Y_{.,i} - \sum_j w_{ij} Y_{.,j}\|^2 \quad \text{is small}$$

- Y: eigenvectors of the lowest d non-zero eigenvalues of

$$M = (I - W)^T (I - W)$$

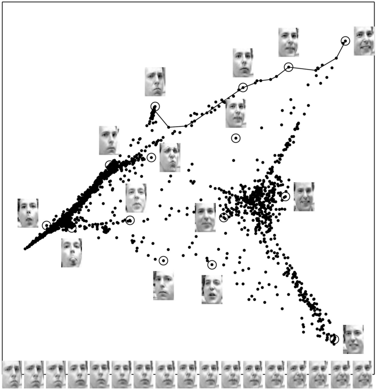
Eigenvalue problem: $E(Y) = Tr(YMY^T)$
$U = \left(U_1 \cdots U_d\right)$: bottom eigenvectors of M. Then

$$Y = U^T \text{ and } M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki} w_{kj}$$

```
from sklearn.manifold import LocallyLinearEmbedding
X = ...
lle = LocallyLinearEmbedding(n_components=2)
X2 = lle.fit_transform(X)
```

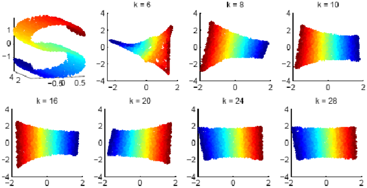
LLE - Example

Images of faces mapped into the embedding space described by the first two coordinates of LLE. Representative faces are shown next to circled points. The bottom images correspond to points along the top-right path (linked by solid line) illustrating one particular mode of variability in pose and expression.



LLE - Limitations

- Require dense data points on the manifold for good estimation
- Need for a good neighborhood $\Rightarrow$ How to choose k?
    - small $\rightarrow$ rank deficient tangent space and lead to over-fitting
    - large $\rightarrow$ Tangent space will not match local geometry well



Laplacian Eigenmaps

Seminal Paper

M. Belkin, P. Niyogi, Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, Neural Computation; 15 (6):1373-1396,June 2003.

The essentials

1. Build the adjacency graph
2. Choose the weights for edges in the graph
3. Eigen-decomposition of the graph laplacian
4. Form the low-dimensional embedding

LAPLACIAN EIGENMAPS - ALGORITHM

Adjacency graph

Connect nodes i and j if x_i and x_j are closed:

1. ϵ-neighborhood: $\|x_i - x_j\|^2 < \epsilon$
   - geometrically motivated, transitive relation
   - graphs with several connected components, choice of ϵ
2. n-nearest neighbors : i among the n nearest neighbors of j. Leads to connected graphs but less intuitive, choice of n ?

Weights

1. heat kernel: if i, j connected: $w_{ij} = -e^{-\frac{\|x_i - x_j\|^2}{\sigma}}$
2. simple-minded: $w_{ij} = 1$ iff i, j connected

Laplacian Eigenmaps - Algorithm

Eigen decomposition

For each connected component of the graph G, compute eigenelements for the generalized eigen problem $Lf = \lambda Df$

- D: diagonal weight matrix $D_{ii} = \sum_j w_{ji}$
- $L = D - W$: Laplacian Matrix

Embedding

f_i: solutions, ordered w.r.t. their eigenvalues $0 = \lambda_0 \leq \cdots \leq \lambda_{k-1}$:
$\forall i \in \{0 \cdots k-1\}) Lf_i = \lambda_i D f_i$
Leaving out f_0, embedding achieved in $Span(f_1 \cdots f_m)$, $m > 0$:

$$x_i \mapsto (f_1(i) \cdots f_m(i))$$

Laplacian Eigenmaps & spectral clustering

1. Involve the same computations.
2. Laplacian Eigenmaps only compute the embedding, i.e., dimension reduction.
3. Spectral clustering not only computes the embedding, but also computes the clustering in the embedded space.

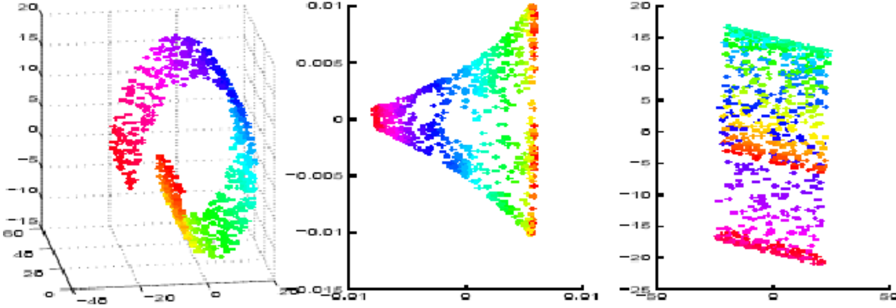
LAPLACIAN EIGENMAPS & LLE

1. LLE computes the eigenvectors of $E = (I - W)^T (I - W)$
2. Under certain conditions (...) $Ef \approx \frac{1}{2}\mathcal{L}^2 f$

$\mathcal{L}$: Laplace Beltrami Operator

$$\int \|\nabla f\|^2 = \int \mathcal{L}(f) f$$

Example



Swiss Roll dataset, 2D Laplacian Representation and PCA.

Definition

LPP

Preserve local structure of the data. Apply laplacian eigenmaps

Algorithm

$Min \frac{1}{2} \sum_{i,j} (y_i - y_j)^2 S_{ij}$ subject to $\quad (\forall i) \quad y_i = w^T x_i$

$$\begin{aligned}
\frac{1}{2} \sum_{i,j} (y_i - y_j)^2 S_{ij} &= \frac{1}{2} \sum_{i,j} (w^T x_i - w^T x_j)^2 S_{ij} \\
&= \sum_{i,j} w^T x_i D_{ii} x_i^T w - w^T X S X^T w \\
&= w^T X (D - S) X^T w = w^T X L X^T w
\end{aligned}$$

LPP - Algorithm

Algorithm

$$\underset{W}{Arg\min}\, w^T X L X^T w$$

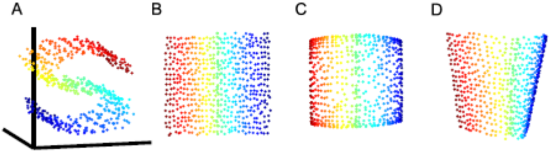
subject to $w^T X D X^T w = 1$

w that minimizes the objective function is given by the minimum eigenvalue solution to

$$X L X^T w = \lambda X D X^T w$$

Properties

LPP $\neq$ Laplacian eigenmaps because computes explicit linear transformation



A: Dataset (600 points sampled from the S curve); B: Isomap ; C: Laplacian eigenmaps /LPP ; D: LLE.
$K = 6$ nearest neighborhoods were used for computing the embeddings