

Dimensionality Reduction

Simplifying life

Yordan Darakchiev

Technical Trainer

iordan93@gmail.com





sli.do

#MachineLearning

Table of Contents

- Projections and dimensionality reduction
 - Problem definition
 - PCA
 - Kernel PCA
 - LinDA
- Manifold learning
 - Problem definition
 - Isometric mapping
 - t-SNE
- Real-world examples

Dimensionality Reduction

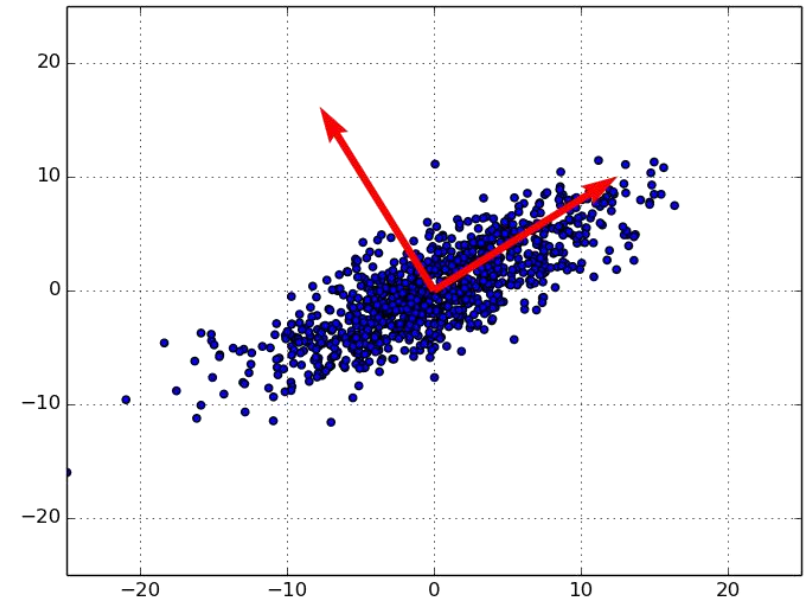
**Reducing the number of features
via projection**

Principal Component Analysis (PCA)

- A **non-parametric** technique for feature extraction
 - Transforms all variables so they are **linearly independent**
 - Note that this is a linear transformation
- The transformation is done to increase the explained variance in the data
- Number of principal components (PC): $p = \min(m, n)$
 - m – number of features, n – number of observations
 - Sorted from highest to lowest explained variance
- Most widely used practical application
 - Load a high-dimensional dataset with m features
 - Perform PCA, remove last k components
 - Result: dataset with $m - k$ features
 - Note that these **are not** $m - k$ of the original columns!

PCA Intuition

- There is a lot of math (and even physics) involved
- Intuitive explanation: casting a shadow
 - Projection into a lower-dimensional space
 - A nice [visualization](#) of PCA
- Example: 2D
 - The first principal component aligns with the axis of most variance in the data
 - The second is perpendicular to the first
 - All components are mutually orthogonal
 - I.e. linearly independent
 - Result: The red arrows represent a new coordinate system
 - Also: if we drop the second coordinate, we've achieved dimensionality reduction without too much loss of information



Example: PCA in scikit-learn

- Reproduce the 2D example
 - Generate a blob
 - Use a shear transformation to make it look like a rotated ellipse
 - You can look up how to do this, for example in Wikipedia
 - Plot the original data, after the transformation
 - **Rescale the data**
 - This is **really important** in PCA, otherwise columns with large values will dominate in the PCs
 - Fit a PCA and apply the learned transformation to the original data

```
from sklearn.decomposition import PCA  
pca = PCA().fit(data)  
transformed_data = pca.transform(data)
```

- Plot the transformed data
- Plot the explained variance ratios of the various components

Kernel PCA

- As we saw, PCA only works for linear transformations
 - We can obtain non-linear transformations using the "kernel trick"
 - Exactly the same way as we did in SVMs
- This allows us to separate non-convex and non-linear data
 - Such as nested circles
 - As before, we need to set the gamma parameter
 - Width of the kernel (in the case of rbf)
 - If we don't know the exact value, we need to perform grid search
- [Docs](#) (note how linear PCA doesn't separate the data)
- Usage

```
from sklearn.decomposition import KernelPCA
pca = KernelPCA(kernel = "rbf", gamma = 5).fit(data)
transformed_data = pca.transform(data)
```


Linear Discriminant Analysis (LinDA)

- LinDA is a supervised method which tries to identify the attributes that account for the most variance between classes
 - Used in classification cases, with known class labels
 - Returns a transformation of the input data, like PCA
- Comparison of 2-component PCA and LinDA on the Iris dataset

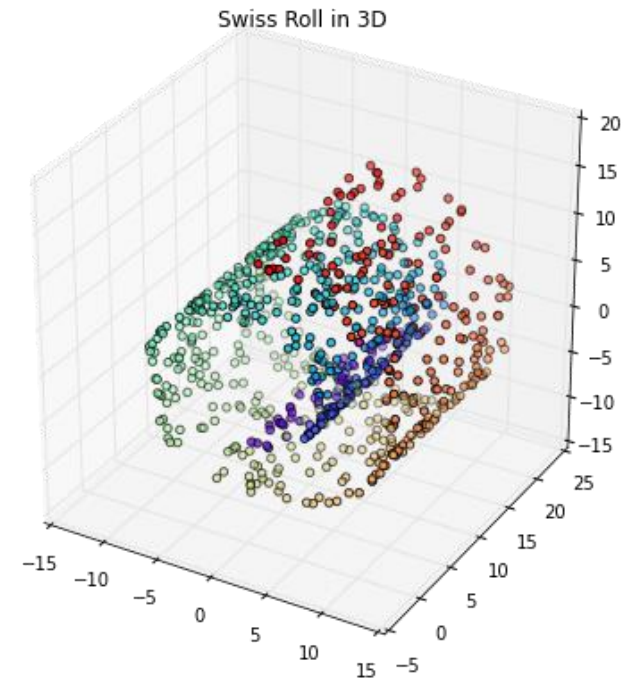
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components = 2)
transformed_attributes = lda.fit(attributes, labels).transform(attributes)
```
- Linear Discriminant Analysis and Latent Dirichlet Allocation **have the same acronyms**, and are both used for dimensionality reduction / transformation
 - They work in completely different ways

Manifold Learning

"Flattening out" surfaces
in fewer dimensions

Problem Definition

- Manifold = surface / shape (not necessarily a plane)
- Idea
 - The dimensions are only artificially high
 - Even though there are many features, they can be expressed with a few parameters
 - "Low-dimensional manifold in a high-dimensional space"
 - Non-linear methods
- Example: "swiss roll" dataset
 - 2D "curled" shape in 3D
- Used not only for dimensionality reduction, it's a common way to visualize high-dimensional datasets
 - Especially for classification cases



Isometric Mapping

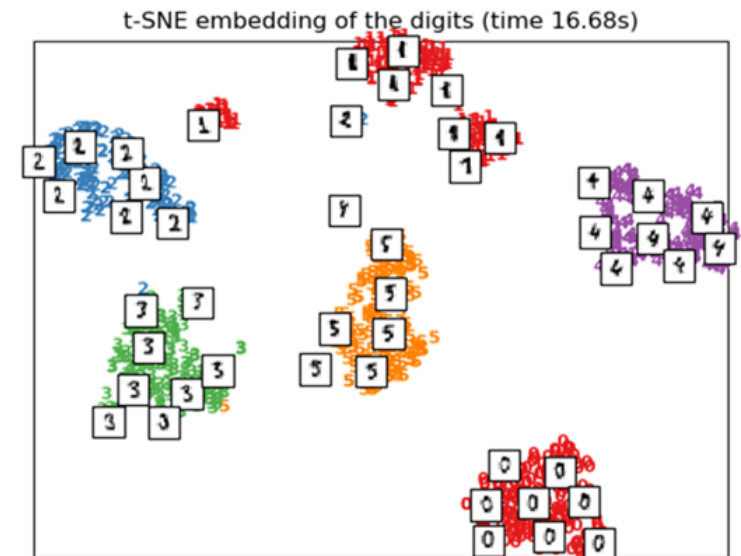
- Seeks a lower-dimensional projection (embedding) which preserves distances between points
- Algorithm overview
 - Find the k nearest neighbors of each point (kNN)
 - Construct a connectivity graph
 - Two points are connected if they're neighbors (from the first step)
 - Compute shortest paths (Dijkstra, Floyd – Warshall)
 - Perform projection on the graph
 - Similar to PCA
- High-level overview: project a manifold using nearest neighbors

```
from sklearn.manifold import Isomap
isomap = Isomap(n_neighbors = 5, n_components = 2)
transformed_data = isomap.fit_transform(data)
```

t-SNE

- t-distributed Stochastic Neighbor Embedding
 - Isomap tries to "unfold" a continuous structure
 - This is also true for other algorithms ([comparison](#))
 - t-SNE looks for local clusters in the data
 - Useful for revealing clusters and structure
- Usual implementation: Barnes – Hut (2D or 3D only)
 - To preserve high-dimensional structure, we need to initialize it with PCA (instead of randomly)
 - It can be slow
 - Mainly used for **visualization** in image and text processing

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, init = "pca")
transformed_data = tsne.fit_transform(data)
```



Real-World Examples

Seeing dimensionality reduction
in action

Example: Iris dataset

- Perform PCA on the Iris dataset
 - Use the results to perform feature selection
- Compare the results to isometric mapping and 2D t-SNE
 - Visualize the first 2 components in all three cases
- Choose a classifier and score it before / after PCA
 - E.g. random forest
 - In most cases, first components will give a lower score but the algorithm will perform much faster (fewer variables)
 - Don't forget to transform new incoming data!
- Compare results on the raw dataset, and on the isomap transformation

Example: Eigenfaces

- Dataset: Labeled Faces in the Wild
 - Included in `scikit-learn`
 - Example
 - Select only people with many images
 - In this case, ≥ 70
 - Use pixel values as raw inputs
 - Perform PCA to select the top 150 components (out of ~ 3000)
 - Use the PCA components to visualize "eigenfaces"
 - I.e. to see each principal component as an image
- ```
eigenfaces = pca.components_.reshape((n_components, h, w))
h, w are image height and width
```
- Train and optimize an SVM
  - Perform classification



# Example: Topics in Text

- Dataset: 20 Newsgroups
  - Included in `scikit-learn`
- Example (LatDA and NMF)
  - LatDA – **L**atent **D**irichlet **A**llocation
    - Most widely-used algorithm for topic extraction
  - NMF – **N**on-negative **M**atrix **F**actorization
    - Simpler than LatDA, also used for topic extraction
- Preprocessing: counting words
  - "Bag of words" model
    - Matrix of how often each word occurs in each document
- Visualization of topics: top  $n$  words for each topic
- LatDA and k-means clustering on the newsgroups

# Summary

- Projections and dimensionality reduction
  - Problem definition
  - PCA
  - Kernel PCA
  - LinDA
- Manifold learning
  - Problem definition
  - Isometric mapping
  - t-SNE
- Real-world examples

The image features a white background with two blue decorative bars. The top bar is a solid blue strip. The bottom bar is a gradient blue strip that transitions from a lighter blue on the left to a darker blue on the right. The word "Questions?" is centered in a blue, sans-serif font.

Questions?