Machine Learning
For Natural Language Processing

Abdelhak Mahmoudi
abdelhak.mahmoudi@um5.ac.ma

2020

Content

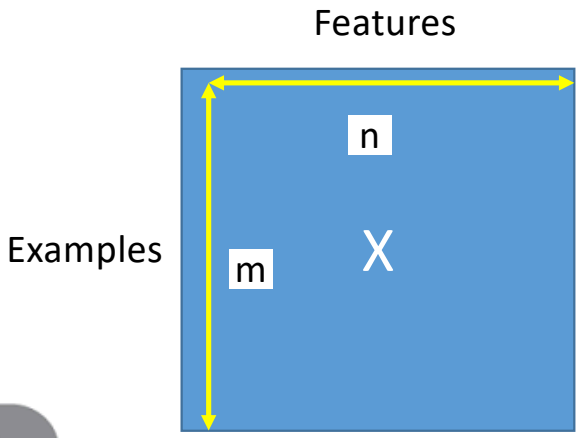
1. Introduction
2. Machine Learning

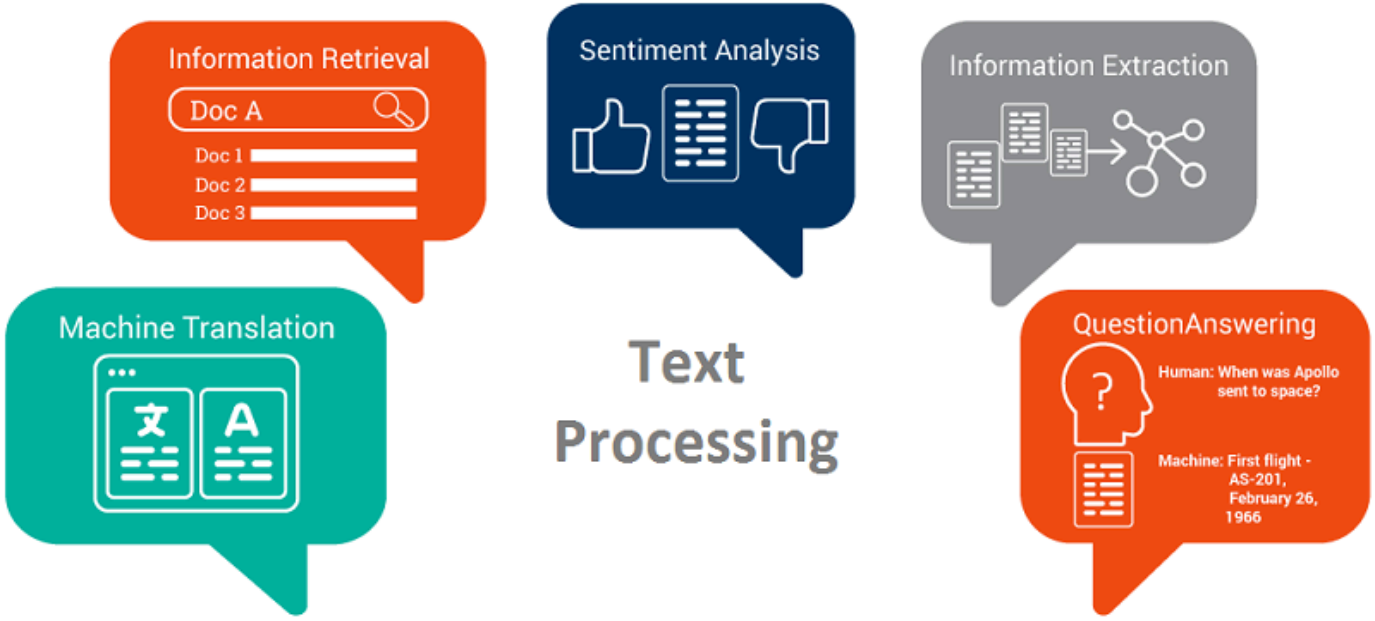
   1. Supervised Learning, 2. Unsupervised Learning
3. Natural Language Processing

   1. Regular Expressions, 2. Tokenization, 3. Character Encoding, 4. Part-of-Speech Tagging, 5. Chunking, 6. Stemming and Lemmatization, 7. Parsing, 8. Named Entity Recognition, 9. Topic Segmentation
4. Introduction to Deep Learning for NLP

   1. Sequence models, 2. Embeddings, 3. BERT models

Unsupervised Learning



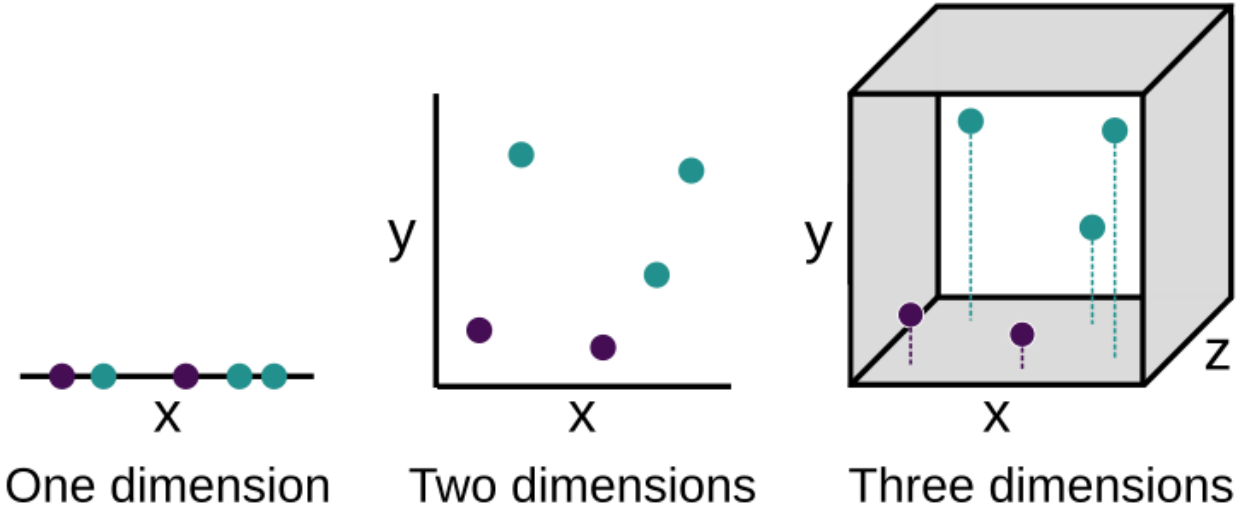


Unsupervised Learning

- **Dimensionality Reduction**
  - **Principal Component Analysis (PCA)**
- Clustering
  - K-Means
  - Mean-Shift

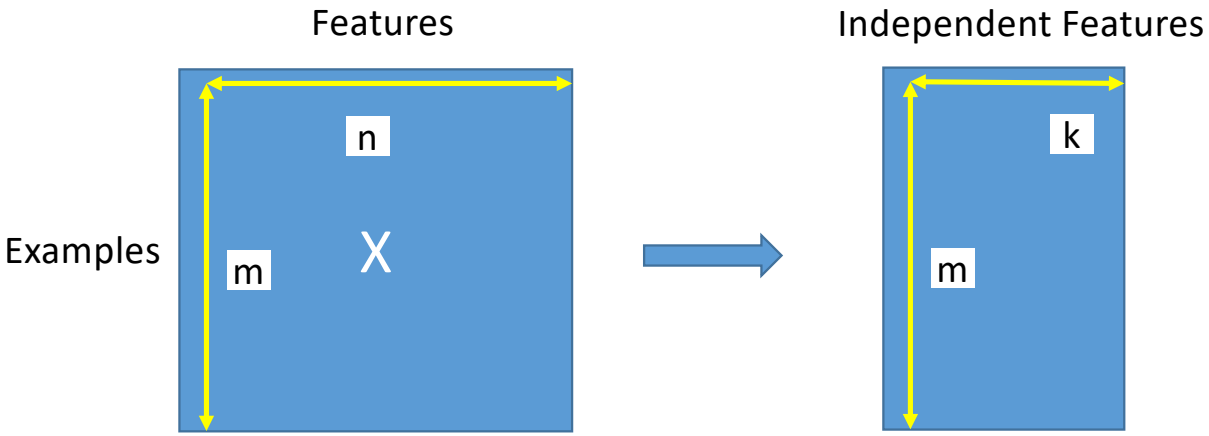
Dimensionality Reduction

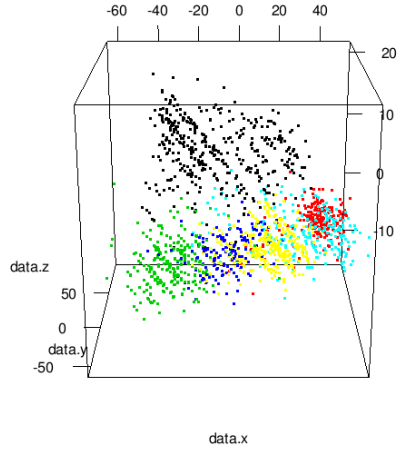
- Curse of dimensionality (n >> m)
  - Data are at risk of being very sparse in high dimensional space
  - High risk of overfitting



Dimensionality Reduction

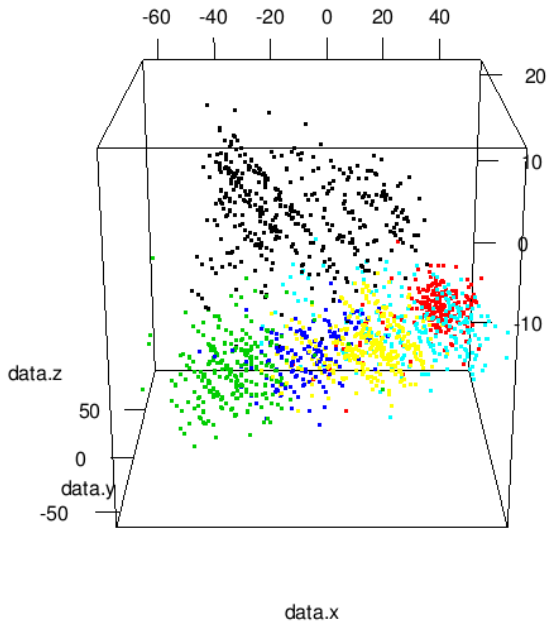
- Transforms feature space from n to k (k<n)
  - Some features are probably corelated (dependent)
  - Some features are almost constant
  - Transform but preserve the maximum of variance



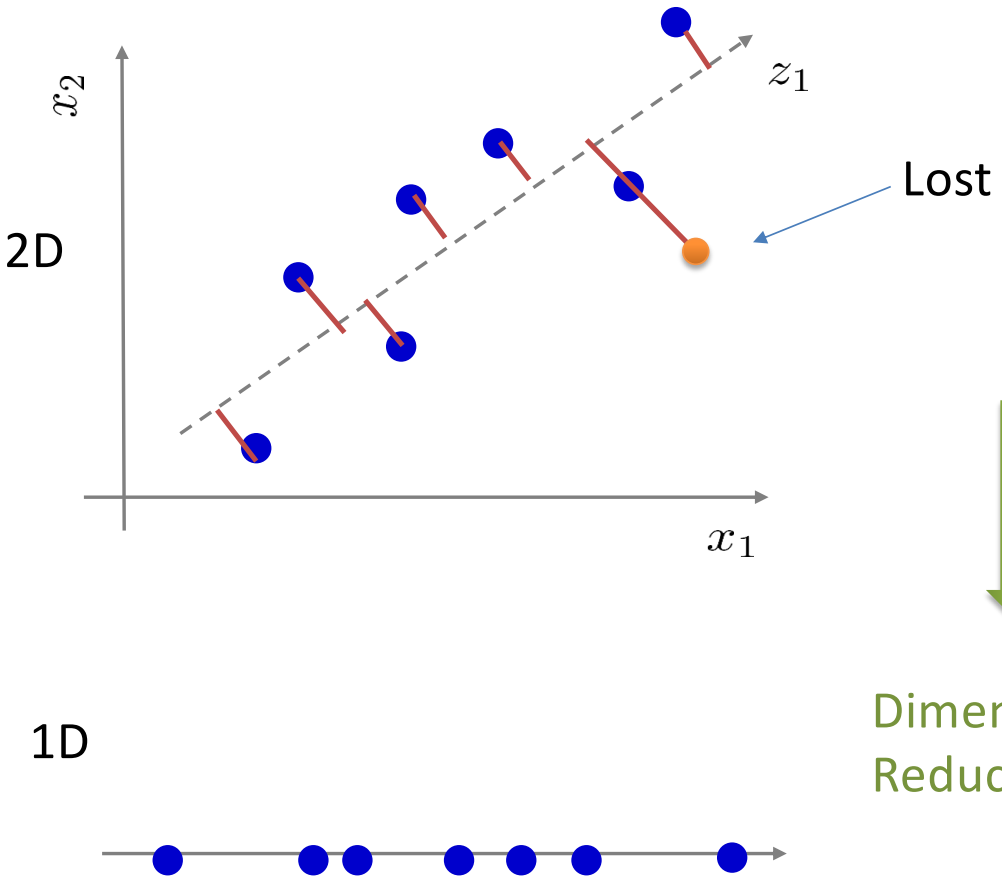


Dimensionality Reduction

- Often
  - Not necessarily lead to better performance
  - Not the better way to address overfitting !
- Always
  - Speed up training
  - Allow data compression
  - Allow data exploration
  - Allow data visualization (DataViz)

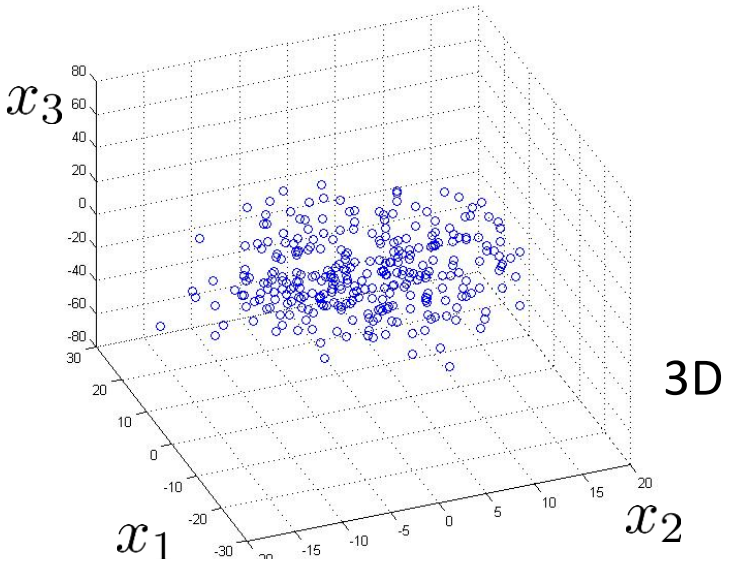


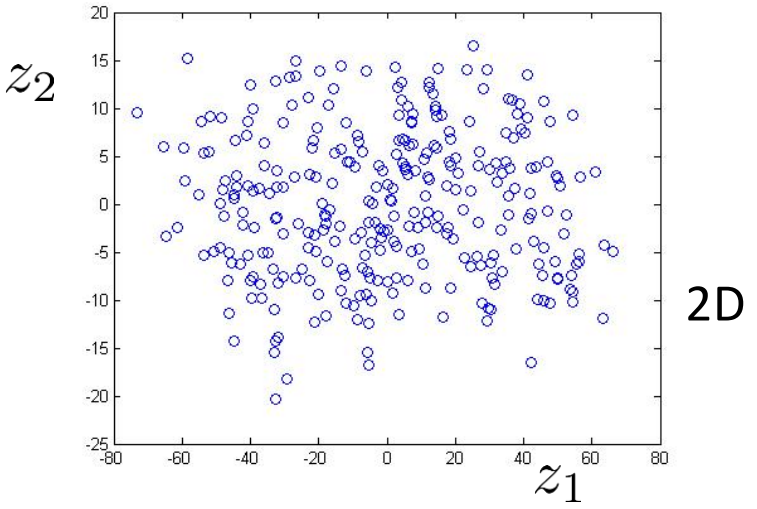
Principal Component Analysis



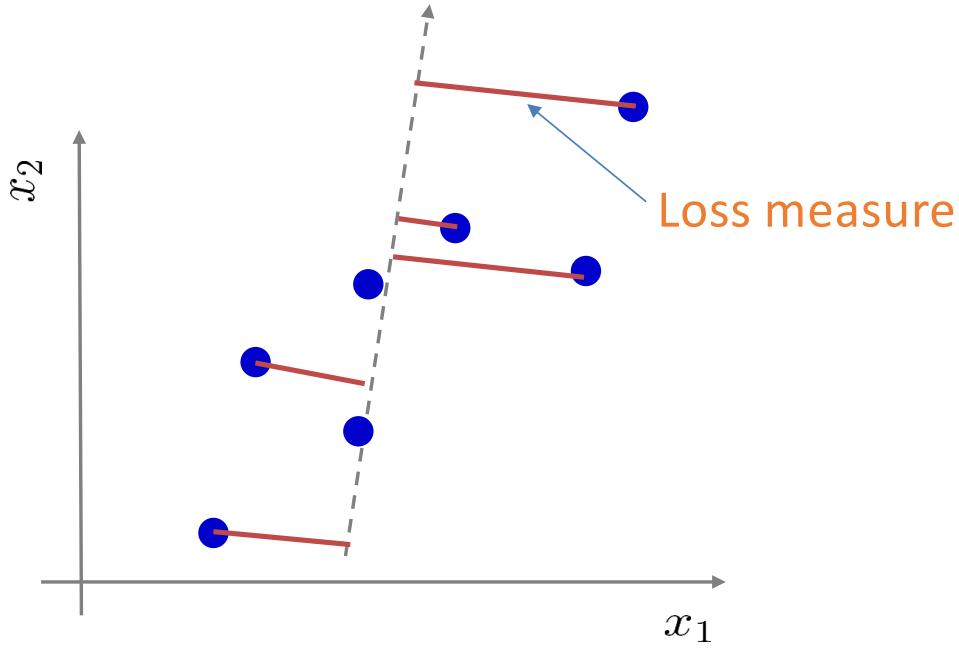
Reconstruction
With Loss

Dimensionality
Reduction

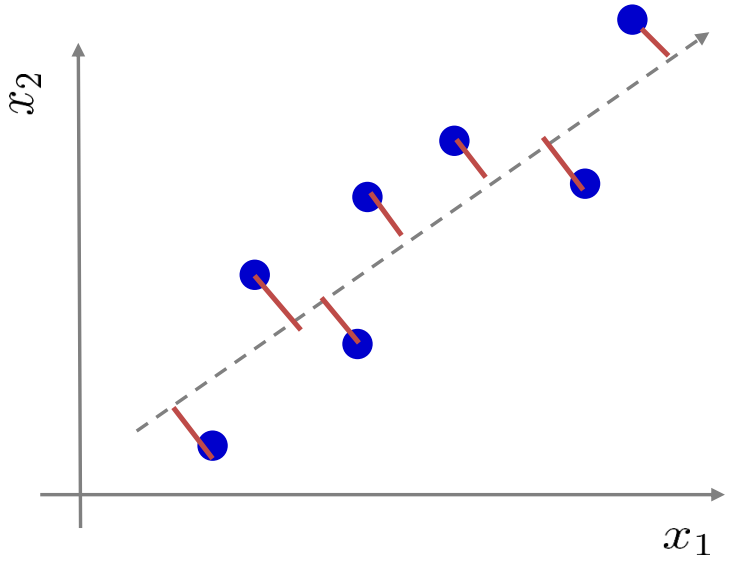




Principal Component Analysis



Maximum loss
Less variance



Minimum loss
More variance

Principal Component Analysis

- Singular Value Decomposition (SVD) (very costly)
  - Parallelization: Incremental PCA (fast), Randomized PCA (faster)
- PCA assumes that the dataset is centered around the origin
- How many dimensions to preserve?
  - Reduce dimensions that add up to a sufficiently large portion of the explained variance (e.g., 99%)
- Kernel PCA (kPCA): use the kernel trick like SVM
- In practice, use kPCA to transform the feature space, then perform classification or regression or clustering.

Principal Component Analysis

- Hyper-Parameters Tuning
  - d: polynomial Kernel
  - γ: RBF kernel
  - k: Number of retained principal components
  - Etc.

Unsupervised Learning

- Principal Component Analysis (PCA)
- **Clustering**
  - **K-Means**
  - **Mean-Shift**

K-Means

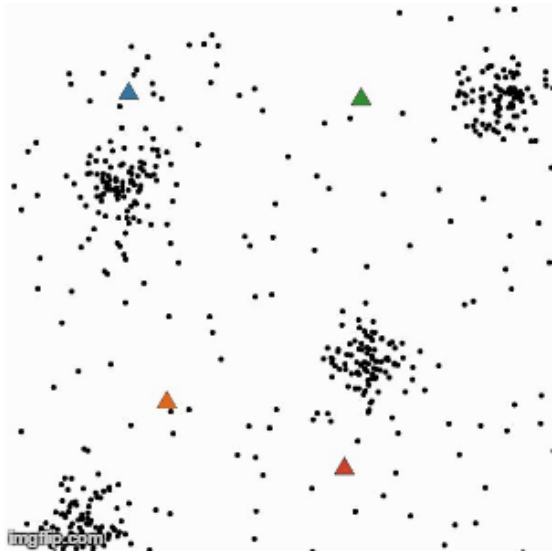
- Pick a number of clusters k
- Initialize centroids randomly
- Problem of local optima
  - Run K-means a lot of times
- Sensible to initial conditions
- Have to specify k !

Repeat until convergence:

Assign each example to the cluster of the nearest centroid

Compute the mean in each cluster

Put the mean as the new centroid



Mean Shift

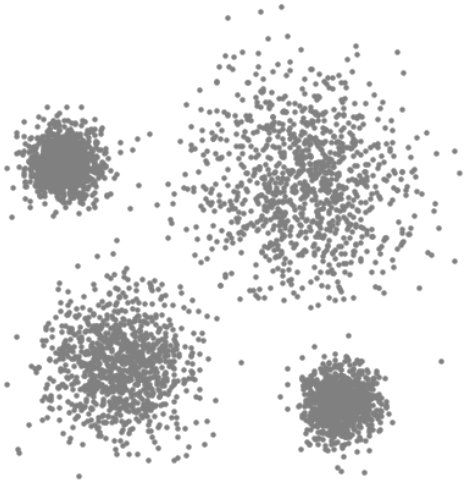
- Chose a radius r of the clusters
- Initialize centroids at each example
- No need to specify the number of clusters

For each example:

Repeat until convergence:

Compute the mean in its cluster with radius r

Shift the cluster to the new mean centroid



Other Clustering methods

- Expectation Maximization (EM)
- Hierarchical Clustering
- Affinity Propagation (AP)
- Etc.