kNN - local weighting and efficiency in high-dimensional spaces and large datasets

David Fiedler
Artificial Intelligence Center, Department of Computer Science, Faculty of Electrical Engineering, CTU in Prague, Czech Republic

Outline

- Quick introduction to kNN
- Problem of the data structure locality
  - Problem
  - Princip and Analysis
  - Results
  - Conclusion and Further reading
- High-dimensional spaces and large datasets
  - Problem
  - Princip and Analysis
  - Results
  - Conclusion and Further reading

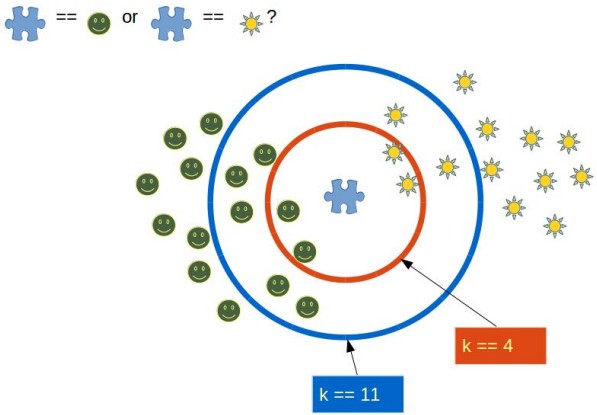
kNN - quick introduction

Algorithm:

1. Find **k** nearest neighbours of the sample in the training dataset
2. Then:
    a. Classify the sample according to weighted majority of the neighbours - *kNN classification*
    b. Compute the value of the sample as a weighted average of the of the neighbours - *kNN regression*

kNN - properties

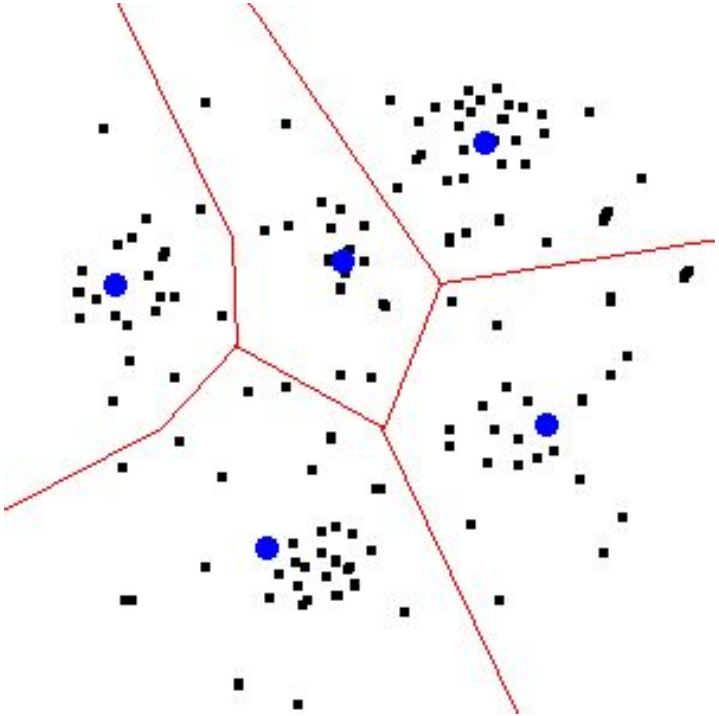
- Very simple :)
- *Instance-based learning* - sample is classified directly from the training data ie - there is no model.
- *Lazy learning algorithm* - all computation is in the classification step
- Very sensitive to local structure of the data - most samples are ignored in the testing phase



Locality in kNN

Motivation

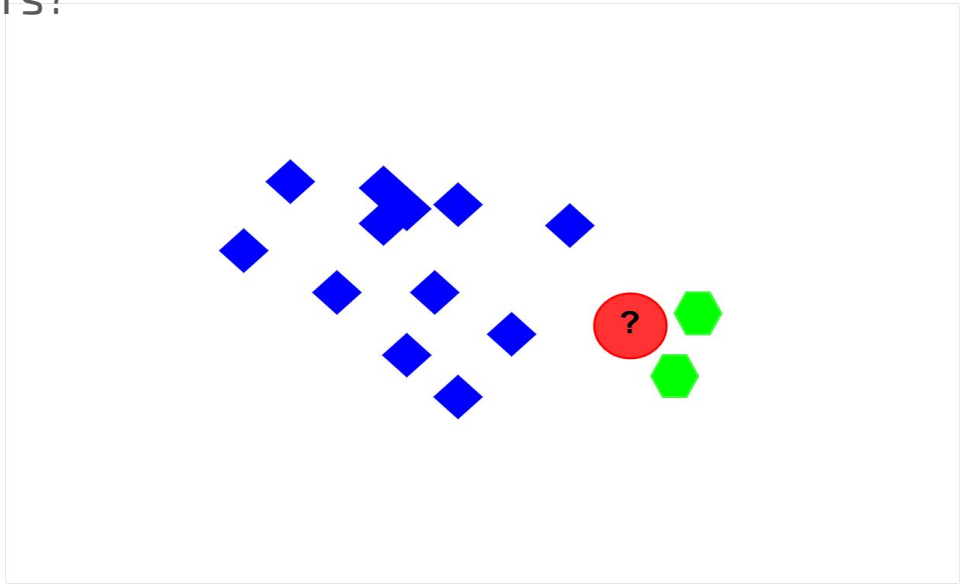
- Most kNN methods choses the same ***k*** for the whole dataset
- This can work well if the data are homogenous
  - Similar class sizes
  - Similar placement of the points inside classes



http://mnemstudio.org/clustering-k-means-introduction.htm

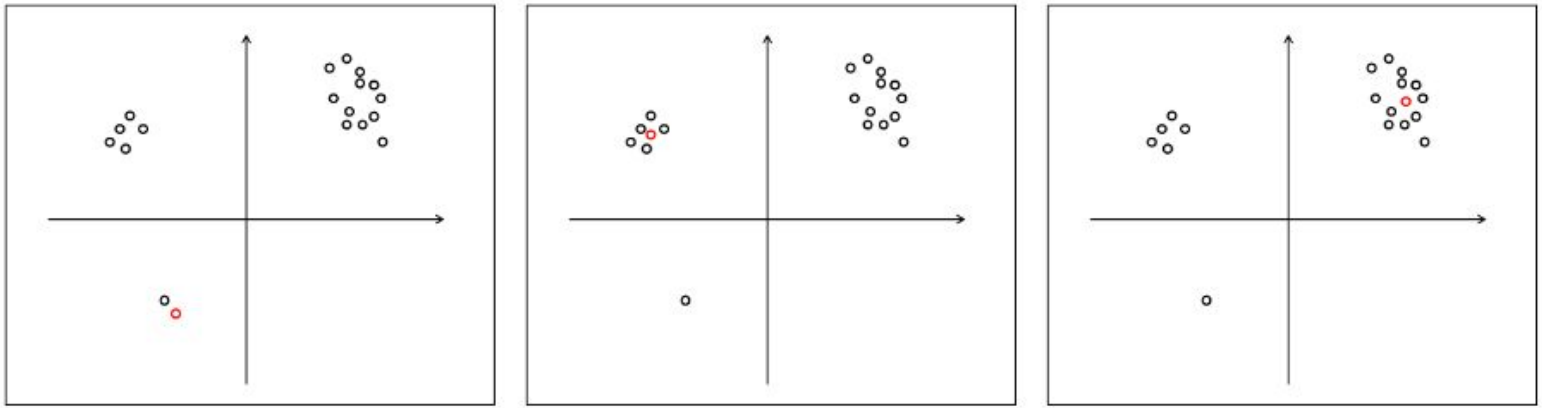
Motivation

- But what if the size of the classes differs?



Idea

- New idea - choose different ***k*** for each sample



(a) First scenario (b) Second scenario (c) Third scenario

[Anava and Levy, 2016]

Idea - Main problems

- Do not increase computation time significantly
- Yield confidence bounds

Minimal assumptions

- The real/underlying model is a Lipschitz function: for each x,y, there is constant ***K*** where $|f(x) - f(y)| \leq K|x - y|$
- Labels of the datapoints are independent

Idea

For each sample:

- Choose the **optimal number of nearest points** - Two extremes:

Big Lipschitzness (low noise) -> we use only nearest point

Lipschitzness goes to 0 (high noise) -> We use all points

How to choose *k* (the number of points) for a dataset with an arbitrary *K* (Lipschitzness)?

- Choose the **optimal weights**:

How?

Solution

A Greedy algorithm is presented as a solution:

- The decrease of the computed weight of the neighbours is slower than in classical kNN
- There is a stopping condition that triggers when the solution is optimal with high probability
  - Probability definition
  - Exploits Lipschitz to noise ratio to determine the optimal *k*

$$\min_{\boldsymbol{\alpha}\in\Delta_n} \left| \sum_{i=1}^{n} \alpha_i y_i - f(x_0) \right|$$

- Similar cost to kNN
  - Computing the costs: ***nd***
  - Sorting: ***n log n***
  - Algorithm itself: ***k****

Results

- New algorithm outperforms classic kNN in 7 datasets from 8
- The number of nearest neighbours was set optimally in all classic kNN scenarios, based on the best results from the learning phase

	Standard k-NN		Nadarays-Watson		Our algorithm (k^*-NN)	
Dataset (n, d)	**Error (STD)**	**Value of k**	**Error (STD)**	**Value of σ**	**Error (STD)**	**Range of k**
QSAR (1055,41)	0.2467 (0.3445)	2	0.2303 (0.3500)	0.1	**0.2105* (0.3935)**	1-4
Diabetes (1151,19)	0.3809 (0.2939)	4	0.3675 (0.3983)	0.1	**0.3666 (0.3897)**	1-9
PopFailures (360,18)	0.1333 (0.2924)	2	**0.1155 (0.2900)**	0.01	0.1218 (0.2302)	2-24
Sonar (208,60)	0.1731 (0.3801)	1	0.1711 (0.3747)	0.1	**0.1636 (0.3661)**	1-2
Ionosphere (351,34)	0.1257 (0.3055)	2	0.1191 (0.2937)	0.5	**0.1113* (0.3008)**	1-4
Fertility (100,9)	0.1900 (0.3881)	1	0.1884 (0.3787)	0.1	**0.1760 (0.3094)**	1-5
Slump (103,9)	3.4944 (3.3042)	4	2.9154 (2.8930)	0.05	**2.8057 (2.7886)**	1-4
Yacht (308,6)	6.4643 (10.2463)	2	5.2577 (8.7051)	0.05	**5.0418* (8.6502)**	1-3

[Anava and Levy, 2016]

Future work

Current kNN only consider distance between points. Could we utilize the geometric properties in the future?

- Something like capsule networks in neural network area?

High dimensional spaces and large datasets

Motivation

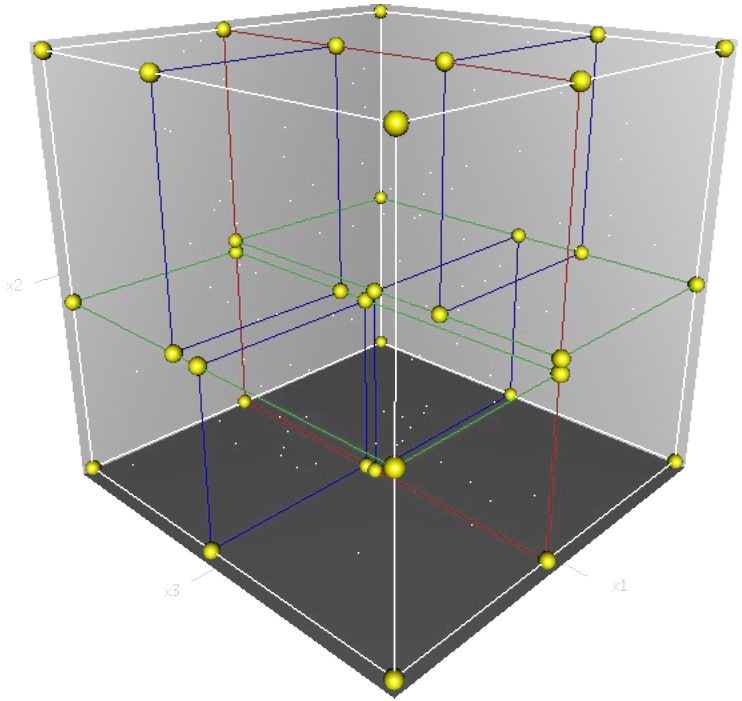
We determine the class of the sample from its ***k*** nearest neighbours.

However, to choose the nearest neighbours, we have to **compute the distance of the sample from all points** in the training dataset.

- Dataset can have a lot of samples!
- The dimension of the data can be high!

Intuitive speedup - k-d tree

- Only returns the k-nearest neighbours, no need for sorting - complexity *O(n)* -> *O(log n)*
- When the number of dimensions is high (close to n), the improvement over the linear search is very low



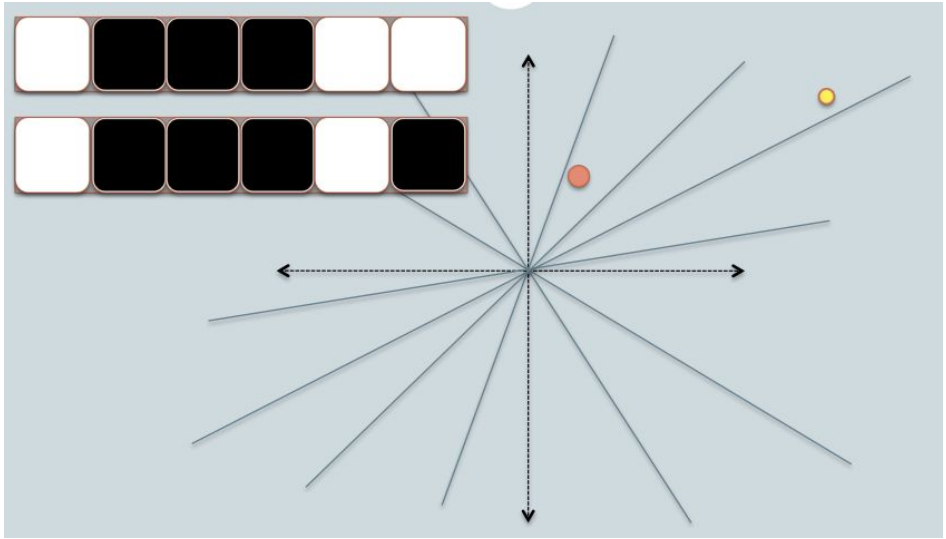
https://en.wikipedia.org/wiki/K-d_tree

Approximate nearest neighbour (ANN)

- Idea: find the k probably nearest neighbours
- Less time needed at the cost of less accuracy
- Solves the problem of k-d tree with high dimensionality
- Common methods:
  - Locality Sensitive Hashing
  - Best Bin First

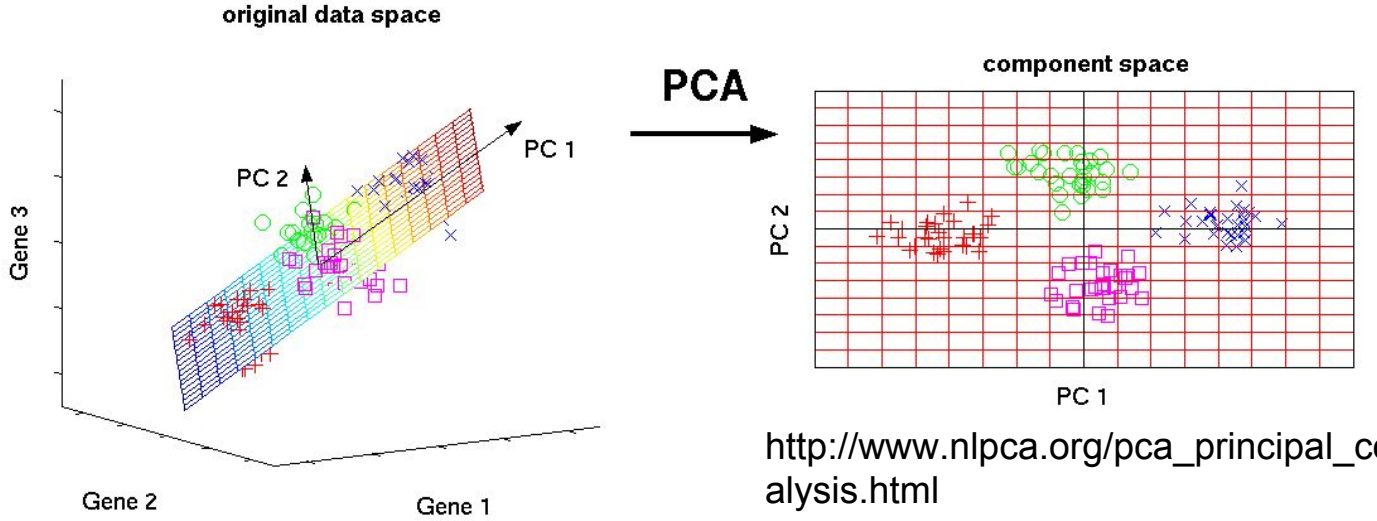
ANN - Locality Sensitive Hashing

- Save the datapoints to a hashtable
- Contrary to the classic hashing, here we try to maximize the probability that similar data are saved in the same row
- After hashing, we compute the distance only to the points with the same hash



ANN - Dimensionality reduction

- We can reduce the dimensionality of the data in a preprocessing step
- Principal Component Analysis (PCA) is a standard way to do that
- PCA finds the bases of the new space so that the error of the projection is minimal



http://www.nlpca.org/pca_principal_component_analysis.html

Another ANN Solution - [Deng et al.]

1. Divide the training dataset using the clustering
2. Find the appropriate cluster for the sample
3. Chose the ***k*** nearest neighbours from the cluster
4. Classify according to the weighted majority of the neighbours

Training

Testing

Solution - Training

First we choose the clustering method

Landmark-based spectral clustering [Cai and Chen]

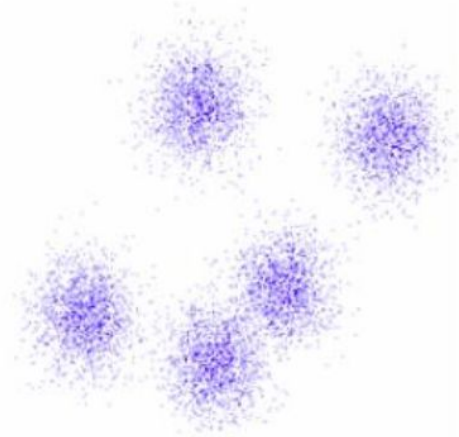
- Properties:
  - Low complexity compared to ordinary Spectral clustering -> linear scaling with respect to data
  - Keeps the properties of the spectral clustering

Solution - Training

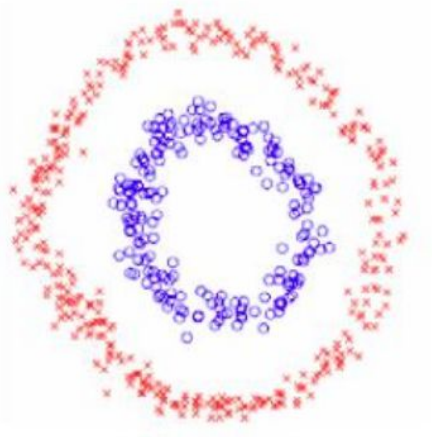
- Landmark-based spectral clustering principe:
  - Create new points to represent the data - *landmarks*
    - Landmarks are computed using k-means
  - Apply the *spectral clustering* [Luxburg]

Solution - Training - Spectral Clustering

- Focused on connectivity rather than on compactness
- It can create non-convex clusters



Compactness



Connectivity

https://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_2.pdf

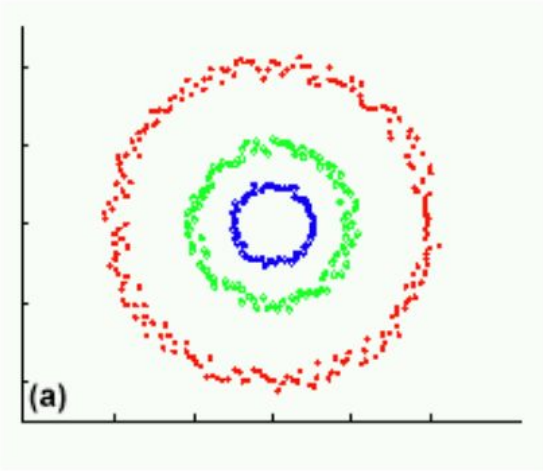
Solution - Training - Spectral Clustering

- Algorithm:
  - Start with a *similarity matrix* ***A*** where $A_{i,j}$ is similarity between x_i and x_j
  - Create a Laplacian Matrix ***L*** from *A*
  - Compute ***k*** eigenvectors ***V*** of *L*
  - Build matrix ***U*** from *V* as columns
  - Interpret the rows of *U* as the original data points and cluster them using k-means - **projecting into spectral dimension, dimensionality reduction**

Solution - Training - Spectral Clustering

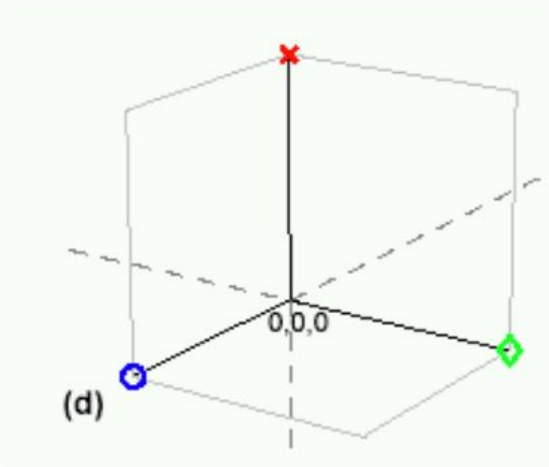
Dimension is future count

Original data



Dimension is the chosen number of eigenvectors

Projected data

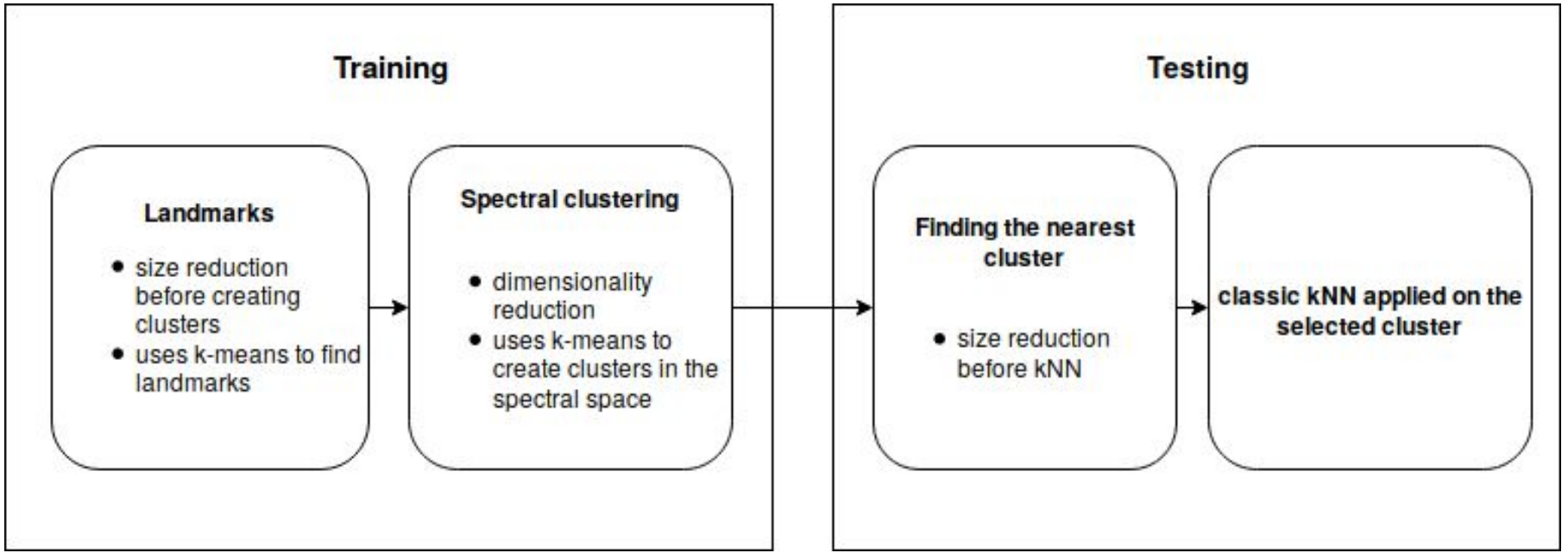


https://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_2.pdf

Solution - Testing

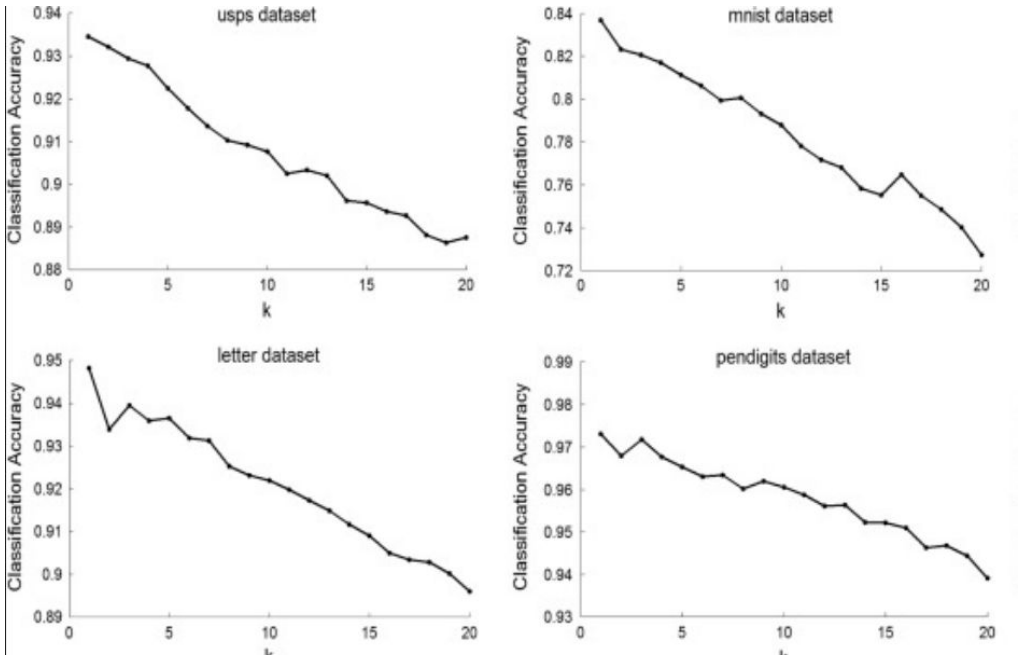
1. Find the appropriate cluster for the sample - the cluster with the nearest center to the sample
2. Chose the ***k*** nearest neighbours from the cluster
3. Classify according to the weighted majority of the neighbours

Solution - Schema



Results

- Here for **10 clusters** created using **1 eigenvector**
- **7-9 times faster** than kNN
- **1-2.6% less accuracy** than kNN



Remarks

- We can use different optimization for spectral clustering (instead of landmarks):
  - KASP - k-means-based approximate spectral clustering [Yan, Huang, and Jordan 2009]
  - CSC - Committees-based Spectral Clustering [Shinnou and Sasaki 2008]
  - Nyström [Chen et al. 2010]
- We can use different algorithm than k-means for creating the landmarks, for example *random sampling*

Thank you!

References

- [Anava and Levy, 2016] Oren Anava and Kfir Levy. “k*-nearest neighbors: From global to local”. *Advances in Neural Information Processing Systems*. pages 4916–4924. 2016.
- [Deng et al] Zhenyun Deng, Xiaoshu Zhu, Debo Cheng, Ming Zong, Shichao Zhang. “Efficient kNN classification algorithm for big data”. *Neurocomputing*. Volume 195, 2016. Pages 143-148.
- [Cai and Chen] Deng Cai, Xinlei Chen. "Large scale spectral clustering via landmark-based sparse representation". *IEEE Transactions on Cybernetics.* vol. 45, no. 8, pp. 1669-1680. 2015.
- [Luxburg] Ulrike Luxburg. “A tutorial on spectral clustering”. Statistics and Computing. 17 (4) (2007). pp. 395-416. 2007.

References

- [Chen et al. 2010] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin and E. Y. Chang. "Parallel spectral clustering in distributed systems". *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2010.
- [Shinnou and Sasaki 2008] H. Shinnou and M. Sasaki. "Spectral clustering for a large data set by reducing the similarity matrix size". *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*. 2008
- [Yan, Huang, and Jordan 2009] D. Yan, L. Huang and M. I. Jordan. "Fast approximate spectral clustering". *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'09).* 2009.