

Fast Kernel PCA and Kernel k -Means

Woosang Lim, Byungkon Kang, Kyomin Jung

Department of Computer Sciences, KAIST
Applied Algorithms Lab.

July 21, 2012

Outline

- 1 Principal Component Analysis
- 2 Clustering
- 3 Kernelized PCA and Clustering
- 4 Our Contribution

Principal Component Analysis (PCA)

Dimensionality reduction:

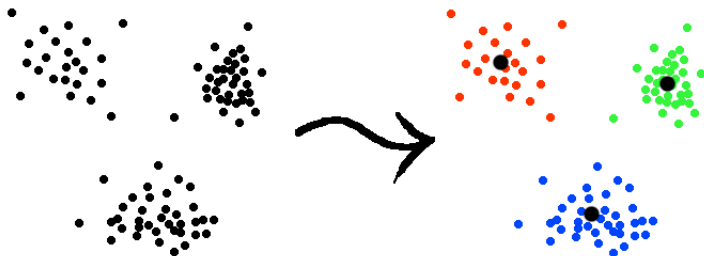
- Project each data point onto a lower-dimensional subspace spanned by k “principal directions”
- The projected points should “best” represent the original points
- Related to clustering [2]
 - The principal directions are related to the optimal clustering (see next slide)
 - If we can efficiently compute the k clusters, we can also efficiently compute the principal directions.



What is Clustering?

Group data points into *clusters* such that:

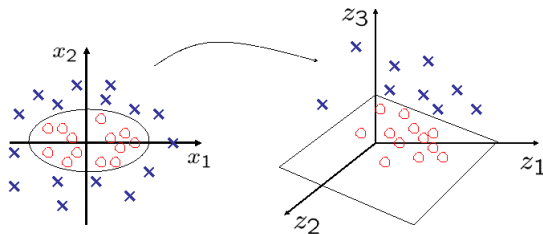
- Points in same clusters are more similar to each other, and
- Points in different clusters are more dissimilar.
- Can use iterative methods to cluster
 - Update the cluster centroids - center of mass of each cluster
 - Assign each point to the cluster with closest centroid, and repeat



Kernel Method

To address non-linearity:

- Use a custom similarity function called *kernel*.
- The kernel function maps points to some Euclidean space, where points might become linearly separable.



Drawback of Kernel Methods

For PCA:

- Requires eigen-decomposition of an $n \times n$ matrix
- Complexity is $\Omega(n^2k)$ to compute k principal directions

For clustering:

- Computing the centroids requires $O(n^2)$
- Centroid update happens every iteration

For large n , which is likely in real-world applications, naive kernel methods are extremely inefficient.

Drawback of Kernel Methods

For PCA:

- Requires eigen-decomposition of an $n \times n$ matrix
- Complexity is $\Omega(n^2k)$ to compute k principal directions

For clustering:

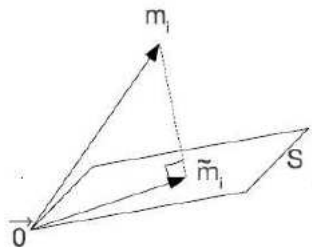
- Computing the centroids requires $O(n^2)$
- Centroid update happens every iteration

For large n , which is likely in real-world applications, naive kernel methods are extremely inefficient.

Subspace Approximation

Main idea: Instead of using all points, use only a sampled subset of the points.

- For PCA, we use this idea to approximately represent the subspace spanned by the principal directions.
- For clustering, we use this to approximate the centroids.



Results

Kernel PCA:

- Prove that given the optimal centroids of the data set, our algorithm computes the optimal principal directions efficiently.
- Up to 1000 times speed-up compared to traditional kernel PCA - from $\Omega(n^2k)$ to $O(k^3n)$
- Experimentally verify good accuracy





Kernel k -Means:

- Approximates the per-iteration cost function by factor $O(1 + \frac{1}{\ell})$ in $O(\ell^3 + n\ell)$ time.
- Experimentally verify speed and accuracy.

Applications

Our algorithms are useful in large-scale high-dimensional problems such as:

- Software engineering [6, 1] - mining call/dependency graphs
- Computer security [3, 5] - PCA for intrusion detection
- Bio-sociology(?) [4] - mining genetic-geospatial relationships

-  N. Anquetil and T.C. Lethbridge.
Experiments with clustering as a software modularization method.
In Conference on Reverse Engineering, 1999.
-  C. Ding and X. He.
k-means clustering via principal component analysis.
In Proceedings of ICML, 2004.
-  L. Mechtri, F. Djemili Tolba, and N. Ghoualmi.
Intrusion detection using principal component analysis.
In International Conference on Engineering Systems Management and Its Applications, 2010.
-  J Novembre, T Johnson, K Bryc, Z Kutalik, A R Boyko, A Auton, A Indap, K S King, S Bergmann, M R Nelson, M Stephens, and C D Bustamante.
Genes mirror geography within europe.
Nature, 456(7218):98–101, November 2008.
-  A. V. K. Prasad and S. R. Krishna.
Data mining for secure software engineering – source code management tool case study.
International Journal of Engineering Science and Technology, 2, 2010.
-  Tao Xie, S. Thummalapenta, D. Lo, and Chao Liu.

Data mining for software engineering.
IEEE Computer, 42(8):55–62, 2009.