CS369M: Algorithms for Modern Massive Data Set Analysis **Lecture 8 - 10/19/2009**

Spectral & Kernel Methods for Nonlinear Dimensionality Reduction (1 of 2)

Lecturer: Michael Mahoney *Scribe: Gourab Mukherjee and Deyan Simeonov*

**Unedited notes*

1 Linear Dimensionality Reduction

1.1 Definition

Let $K^{(x,y)}$ be a kernel. ψ is an *eigenfunction* of K if

$$\int K(x,x')\psi(x')dx' = \lambda\psi(x)$$

1.2 Theorem[Mercer]

Let K be a positive-definite kernel, $\int\int K(x,x')dxdx' < \infty$. Then we can write K in terms of a countable set of orthonormal eigenfunctions and eigenvalues, in other words there exist (ψ_i, λ_i), such that

$$K(x,x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

Our goal is to use this coordinate representation to construct feature maps.

Define $H = \{\sum_{j=1}^{\infty} \psi_j(x)c_j\}$ (note that we have $K(x,x') \in H$).
Define $f(x) = \sum_j c_j\psi_j(x)$ and $g(x) = \sum_j d_j\psi_j(x)$.
Then $< f,g >= \sum_j \frac{1}{\lambda_j} c_j d_j$ (the $\frac{1}{\lambda_j}$ factor smooths out the space.)

Claim. This is a reproducing kernel Hilbert space (RKHS):

$$< f(x), R(x,x') >= \sum_j \frac{1}{\lambda_j} c_j \lambda_j \psi_j(x) = \sum_j c_j \psi_j(x)$$

Let above is given a positive kernel K. Then we can construct a RKHS with K as the dot product. In addition to defining H, we can use this to find the feature map $\phi : X \to H$, such that $K(x,x') =< \phi(x), \phi(x') >$.

There are 2 representations:
1. Use H_K as the feature space:
Define $\phi(x) = K(\cdot, x)$. Then

$$< \phi(x), \phi(x') >=< K(\cdot,x), K(\cdot,x') >= K(x,x')$$

2. Use l_2 as the feature space:
Define $\phi(x) = (\lambda_1^{-1}\psi_1(x), \lambda_2^{-1}\psi_2(x), ...)$. Then

$$< \phi(x), \phi(x') >= \sum_{j=1}^{\infty} \lambda_j \psi_j(x)\psi_j(x') = K(x,x')$$

We want to use this to do optimizations.

1.3 Representative theorem

We have a kernel K on $X \times X$, $(\vec{x_i}, y_i) \in X \times \mathbb{R}$ $(1 \leq i \leq m$, a strictly increasing function $g : (0, \infty] \to \mathbb{R}$, a cost function $c : (X \times \mathbb{R}^2)^m \to \mathbb{R} \cup \infty$, and a class of functions $f : X \to \mathbb{R}, f(\cdot) = \sum_i \beta_i K(\cdot, z_i)$ with a RKHS norm (i.e. $(\sum_i \beta_i K(\cdot, z_i))^2 = \sum_{i,j} \beta_i \beta_j K(z_i, z_j)$).
If c is a regularized risk function, then $\min(c((x_1, y_1, (x_1)), ..., (x_m, y_m, (x_m))) + g(||f||))$ will admit a solution of the form $f(\cdot) = \sum_i \alpha_i K(\cdot, x)$. So, any algorithms, written in terms of dot products can be "kernelized".

There are "a priory" kernels:

- Gaussian kernel
- Polynomial kernel
- etc.

One can construct a "data-dependent" kernel:
1. Construct a similarity graph
2. Do eigenanalysis on the Laplacian or adjacency matrix. Then "embed" the data, using these eigenvectors.
In addition, we can view these things as kernels. They rely on a small number of algorithmic primitives. Think of these procedures as data analysis tools.

1.4 Linear dimensionality reduction methods

1.4.1 PCA

- maximize variance
- minimize construction error
- $C \sim XX^T$

You can "kernelize" PCA, i.e. write it in terms of dot products.

1.4.2 MDS (Multi-Dimensional Scaling)

- minimize dot product error
- $G \sim X^T X$

Let $\delta_{ij} = ||x_i - x_j||_2^2 = (x_i - x_j)^T (x_i - x_j)$.
Let $A_{ij} = -\frac{1}{2}\delta_{ij}$.
$B = HAH$, where H is a "centering" matrix ($H = I - \frac{1}{n} 1.1^T$).
B is SPSD.

Fact:
If $K_{ij} = f(||x_i - x_j||)$, then $K_{ij} = r(\delta_{ij})$ $(r(0) = 1)$.

$$\tilde{\delta_{ij}} = \text{Euclidean distance in feature space} = (\phi(x_i) - \phi(x_j))^T (\phi(x_i) - \phi(x_j)) = 2(1 - r(\delta_{ij}))$$

i.e. A is such that $A_{ij} = r(\delta_{ij}) - 1$, $A = K - 1.1^T$, so (fact) $HAH = HKH$.

In the linear case both PCA and MDS rely on SVD and can be constructed in $O(mn^2)$ time $(m > n)$.
Note: For isotropic kernels, i.e. $k_{ij} = f(||x_i - x_j))$, PCA is a form of MDS and vice-versa.

2 Non-LinearDimension Reduction

General Framework

- Derive some graph (often sparse) from the data.
- Derive Matrix from the graph (viz. adjacency matrix, Laplacian).
- Dense embedding into $\Re^d$ for eigen vectors.

2.1 ISOMAP

Algorithm

- Build the nearest neighbor graph.
- Look at the shortest path or geodesic distance between all pairs.
- Do Mutlidimensional scaling (MDS) based on A (the shortest path distance matrix).

Advantages

- Polynomial Time.
- No local minima.
- Non-iterative.

Disadvantages

- Non-linear Time.
- No immediate out of sample extension.

2.2 Local Linear Embedding (LLE)

Algorithm

Step1 : Construct the Adjacency Graph There aretwo variations:

1. ϵ neighborhood
2. K Nearest neighbor Graph

Step2 : Choosing weights Weights are choosen based on the projection of each datapoint on the linear subspace generated by its neighbors. $W_{ij} = \begin{cases} 0 & \text{if vertex j is not a neighbor of i} \\ \text{argmin} \sum_i ||x_i - \sum_j w_{ij} x_j||^2 & \text{otherwise} \end{cases}$

Step3 : Mapping to Embedded Co-ordinates Compute output $y \in \Re^d$ such that

$$\psi(y) = \sum_i ||y_i - \sum_j W_{ij} y_j||^2 \text{ is minimized .} \quad (1)$$

- The above minimization reduces to finding eigen vectors corresponding to the (k+1) lowest eigenvalues of the the positive definite matrix $(I - W)'(I - W)$
- Lowest eigen value is uninteresting so have to throw that eigen vetor out.

2.3 Laplacian Eigenmaps (LE)

Algorithm

Step1 : Construct the Adjacency Graph There aretwo variations:

1. ϵ neighborhood
2. K Nearest neighbor Graph

Step2 : Choosing weights

$$W_{ij} = \begin{cases} e^{-\frac{||x_i - x_j||^2}{4t}} & \text{if vertices i \& j are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

Step3 : Eigen maps We compute $y \in \Re^d$ such that

$$\psi(y) = \sum_{i,j} \frac{w_{ij}||y_i - y_j||^2}{\sqrt{D_{ii} \cdot D_{jj}}} \tag{2}$$

is minimized for each connected component of the graph where $D = diag\{\sum_i w_{ij} : j = 1(1)n\}$

3 References

1. Saul, Weinberger, Ham, Sha, and Lee. Spectral methods for dimensionality reduction .*Semisupervised Learning* MIT Press, Cambridge, MA, 2006
2. Belkin and Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 2003 - MIT Press