1 Linear Dimensionality Reduction

1.1 Definition

Let $K(x,y)$ be a kernel. $\psi$ 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, $\int \int K(x,x')dx'dx' < \infty$. Then we can write $K$ in terms of a countable set of orthonormal eigenfunctions and eigenvalues, in other words there exist $(\psi_i, \lambda_i)$, 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):

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

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 $\phi : X \rightarrow H$, 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 $c : (X \times \mathbb{R})^m \to \mathbb{R}$, and a class of functions $f : X \to \mathbb{R}$, $f(\cdot) = \sum \beta_i K(\cdot, z_i)$ with a RKHS norm (i.e. $(\sum \beta_i K(\cdot, z_i))^2 = \sum_{i,j} \beta_i \beta_j K(z_i, z_j)$).

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 $f(\cdot) = \sum \alpha_i K(\cdot, x)$. So, any algorithms, written in terms of dot products can be "kernelized".

There are "a priori" 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
- $G \sim X^TX$

Let $\delta_{ij} = ||x_i - x_j||^2 = (x_i - x_j)^T(x_i - x_j)$.
Let $A_{ij} = -\frac{1}{2} \delta_{ij}$.
Let $B = HAH$, where $H$ is a "centering" matrix ($H = I - \frac{1}{n}11^T$).

$B$ is SPD.

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

$\tilde{\delta}_{ij} = $ 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. $A_{ij} = r(\delta_{ij}) - 1$, $A = K - 11^T$, so (fact) $HAH = HHK$.

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-Linear Dimension Reduction

General Framework

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

2.1 ISOMAP

Algorithm

- Build the nearest neighbor graph.
- Look at the shortest path or geodesic distance between all pairs.
- Do Multidimensional 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 are two variations:

1. \( \epsilon \) neighborhood
2. K Nearest neighbor Graph

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

**Step3 : Mapping to Embedded Co-ordinates** Compute output \( y \in \mathbb{R}^d \) such that

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

- 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 eigenvector out.
2.3 Laplacian Eigenmaps (LE)

Algorithm

Step 1: Construct the Adjacency Graph
There are two variations:
1. \( \epsilon \) neighborhood
2. K Nearest neighbor Graph

Step 2: Choosing weights

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

Step 3: Eigen maps

We compute \( y \in \mathbb{R}^d \) such that

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

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

3 References
