Kernel Approaches to Supervised/Unsupervised Machine Learning*

S.Y. Kung

Princeton University

Topics:

1. Kernel-Based Unsupervised/supervised Machine Learning: Overview

2. Kernel Based Representation Spaces

3. Kernel Spectral Analysis (Kernel PCA)

4. Kernel Based Supervised Classifiers


6. Extension to Nonvectorial Data and NPD (None–Positive–definite) Kernels
1. Overview

Machine learning techniques are useful for data mining and pattern recognition when only pairwise relationships of the training objects are known – as opposed to the training objects themselves. Such pairwise learning approach has a special appeal to many practical applications such as multimedia and bioinformatics, where a variety of heterogeneous sources of information are available. The primary information used in the kernel approach is either (1) the kernel matrix (K) associated with either vectorial data or (2) the similarity matrix (S) associated with nonvectorial objects.
From Linear to Kernel SVMs

Wolfe optimization:

Subject to the constraints:

\[ \sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \ i = 1, \ldots, N \]

Linear inner-product objective function

\[
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \right\}
\]

Kernel inner-product objective function

\[
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right\}
\]

Decision Function:

\[
f(x) = \sum_{i=1}^{N} y_i \alpha_i K(x, x_i) + b.
\]
Why the Kernel Approach:
- A New Distance Metric

Traditionally, we have linear dot-product for two *vectorial data* \( x, \) and \( y. \)

Traditionally, \( \langle x, y \rangle = x^T y \)

\[
D^2 = \| x \|^2 + \| y \|^2 - 2x^T y = \langle x, x \rangle + \langle y, y \rangle - 2\langle x, y \rangle
\]

In the Kernel Approach, a new distance:

\[
\tilde{D}^2 = K(x, x) + K(y, y) - 2K(x, y)
\]

New Dot-Product means different distance (norm) metric in the new Hibert space.
Kernelization ⇒ Cornerization

(The figures below were downloaded from Wiki.)
Application to XOR Problem

Not linearly separable in $\mathbf{X}$....

...but linearly separable in kernel spaces: e.g. $F$, $E$, & $K$.
PCA vs. Kernel PCA
Why the Kernel Approach: nonvectorial data

Such pairwise learning approach has a special appeal to many practical applications such as multimedia and bioinformatics, where a variety of heterogeneous sources of information are available.

The learning techniques can be extended to the situations where only pairwise relationships of the training objects are known as opposed to the training objects themselves.
Two kinds of Kernel (or Similarity) Matrices

The primary information used in the kernel approach:

• **Vectorial Data:** PD Kernel Matrix for polynomial kernel:
  \[ K(x_i, x_j) = (1 + \frac{x_i \cdot x_j}{\sigma^2})^{2p} \]
  or Gaussian kernel:
  \[ K(x_i, x_j) = \exp \left\{ -\frac{||x_i-x_j||^2}{2\sigma^2} \right\} \]

  Sigmoid Kernel (rarely used) can be NPD.

• **Nonvectorial Data:** NPD Kernel Matrix well-defined centroid/distance.
  PD

  Similarity Matrix
(1) Kernel matrix \((K)\) for vectorial data

\[
Z = \begin{bmatrix}
z_{11} & z_{12} & \cdots & z_{1N} \\
z_{21} & z_{22} & \cdots & z_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
z_{M1} & z_{M2} & \cdots & z_{MN}
\end{bmatrix}
= \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_N
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_M
\end{bmatrix}
\]

Microarray gene expression data:

Kernel matrix for tissue samples:

\[
S_{xx} = \begin{bmatrix}
S(x_1,x_1) & S(x_1,x_2) & \cdots & S(x_1,x_N) \\
S(x_2,x_1) & S(x_2,x_2) & \cdots & S(x_2,x_N) \\
\vdots & \vdots & \ddots & \vdots \\
S(x_N,x_1) & S(x_N,x_2) & \cdots & S(x_N,x_N)
\end{bmatrix}
\]

Kernel matrix for genes:

\[
S_{yy} = \begin{bmatrix}
S(y_1,y_1) & S(y_1,y_2) & \cdots & S(y_1,y_M) \\
S(y_2,y_1) & S(y_2,y_2) & \cdots & S(y_2,y_M) \\
\vdots & \vdots & \ddots & \vdots \\
S(y_M,y_1) & S(y_M,y_2) & \cdots & S(y_M,y_M)
\end{bmatrix}
\]
Hierarchical clustering of gene expression data the grouping of genes (rows) and normal and malignant lymphocytes (columns). Depicted are the 1.8 million measurements of gene expression from 128 microarray analyses of 96 samples of normal and malignant lymphocytes. 

(2) Similarity matrix (S) for nonvectorial objects.

(a) Signal 1  
Signal 2  
Signal 3  

Similarity Matrix

(b) Sequence 1  CCATCTATTGCTTAC
Sequence 2  CCAACTATAC
Sequence 3  GCTTACATTTTGCG

Similarity Matrix

(c) A=
\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]
Topics:

1. Kernel-Based Unsupervised/supervised Machine Learning: Overview

2. Kernel Based Representation Spaces
   - 2.1 Basic Kernel Hilbert Space
   - 2.2 Spectral Space
   - 2.3 Empirical Space
   - 2.4 Linear Mapping Betw. Spaces

3. Kernel Spectral Analysis (Kernel PCA)

4. Kernel Based Supervised Classifiers

5. Fast Clustering Methods for Positive-Definite Kernels (Kernel Component Analysis and Kernel Trick)

6. Extension to Nonvectorial Data and NPD Kernels
Kernel Based Representation Spaces

\[ K(x, x') = \sum_{j=1}^{J} \phi^{(j)}(x) \phi^{(j)}(x') \]

New vector space:

\[ x = [x^{(1)}, x^{(2)}, \ldots, x^{(N)}]^T \rightarrow \phi(x) = [\phi^{(1)}(x), \phi^{(2)}(x), \ldots, \phi^{(J)}(x)]^T \]

“Linear” discriminantant function:

\[ f(x) = w^T \phi(x) + b = \sum_{j} w^{(j)} \phi^{(j)}(x) + b. \]
Polynomial Kernel Function

\[ K(x, x') = (1 + x \cdot x')^2 = \vec{\phi}(x)^T \vec{\phi}(x') \]
\[ \vec{\phi}(x) = [1 \ \sqrt{2u} \ \sqrt{2v} \ \sqrt{2uv} \ u^2 \ v^2]^T \]

“Linear” discriminant function:

\[ w \cdot \vec{\phi}(x) + b \]

Output space

Feature space

New vector space:

\[ x = [u \ v]^T \]
XOR Training Dataset

\[
\begin{bmatrix}
+1 \\
+1
\end{bmatrix}, \begin{bmatrix}
-1 \\
-1
\end{bmatrix}, \begin{bmatrix}
+1 \\
-1
\end{bmatrix}, \begin{bmatrix}
-1 \\
+1
\end{bmatrix}.
\]

teacher = +1 teacher = +1 teacher = -1 teacher = -1

\[
\begin{align*}
\vec{\phi}(x_1) &= \begin{bmatrix}
+1 \\
+\sqrt{2} \\
+\sqrt{2} \\
+1
\end{bmatrix}, \\
\vec{\phi}(x_2) &= \begin{bmatrix}
1+1 \\
-\sqrt{2} \\
-\sqrt{2} \\
+1
\end{bmatrix}, \\
\vec{\phi}(x_3) &= \begin{bmatrix}
+1 \\
+\sqrt{2} \\
-\sqrt{2} \\
+1
\end{bmatrix}, \\
\vec{\phi}(x_4) &= \begin{bmatrix}
+1 \\
-\sqrt{2} \\
-\sqrt{2} \\
+1
\end{bmatrix}
\end{align*}
\]

\[
\mathbf{w}^T = \begin{bmatrix}
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0
\end{bmatrix} \quad b = 0.
\]
The representation vector space has the following properties:

• *It is a Hilbert space endowed with dot-product.*

• *It has non-orthogonal bases with a finite dimension, J.*

• *It is independent of the training dataset.*
Kernel and Basis Function

Orthogonal Basis Function

Assume $K(x, y)$ is positive-definite (PD)[21], i.e. $\lambda_k \geq 0$

$$K(x, y) = \sum_{k=1}^{\infty} \lambda'_k \psi^{(k)}(x)\psi^{(k)}(y)$$

Assume $K(x, y)$ is positive-definite (PD)[21], i.e. $\lambda_k \geq 0$

$$\phi^{(k)}(x) = \sqrt{\lambda'_k} \psi^{(k)}(x)$$

$$K(x, y) = \sum_{k} \phi^{(k)}(x)\phi^{(k)}(y)$$
2.1 Basic Kernel Hilbert Space

These basis functions form the coordinate bases of the representation vector space with the dot-product:

\[ \phi(x) = [\phi^{(1)}(x), \phi^{(2)}(x), \ldots]^T \]

\[ K(x, y) = \phi(x)^T \phi(y) \]

\[ f(x) = w^T \phi(x) + b. \]
Basic Kernel Hilbert Space: $\mathcal{F}$

It has the following properties:

• It is a Hilbert space endowed with a dot-product.

\[
K(x, y) = \sum_{k=1}^{\infty} \phi^{(k)}(x) \phi^{(k)}(y) = \overrightarrow{\phi}(x)^T \overrightarrow{\phi}(y)
\]

• It has ordered orthogonal bases with indefinite dimension, and

• It is independent of the training dataset.
Kernel Matrix

\[ K = \begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_N) \\
K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_N) \\
\vdots & \vdots & \ddots & \vdots \\
K(x_N, x_1) & K(x_N, x_2) & \cdots & K(x_N, x_N)
\end{bmatrix} \]

Two Factorizations of Kernel Matrix:

(1) \( F \): Basic Kernel Hilbert Space

(2) \( E \): Kernel Spectral Space
Factorization: Basic Kernel Hilbert Space

\[ K = \Phi^T \Phi, \]

\[ \Phi = \begin{bmatrix} \phi(x_1) & \phi(x_2) & \cdots & \phi(x_N) \end{bmatrix} \]

\[ = \begin{bmatrix} \phi^{(1)}(x_1) & \phi^{(1)}(x_2) & \cdots & \phi^{(2)}(x_N) \\ \phi^{(2)}(x_1) & \phi^{(2)}(x_2) & \cdots & \phi^{(2)}(x_N) \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \]

\[ x \rightarrow \Phi(x). \]

\[ \int \phi^{(k)}(x)\phi^{(j)}(x) dx = \begin{cases} 0 & \text{if } k \neq j, \\ \lambda_k & \text{if } k = j, \end{cases} \]
Kernel Spectral Factorization

Apply the eigenvalue decomposition on the kernel-matrix $K$:

$$K = U^T \Lambda U = U^T \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} U = E^T E$$

$$E = \Lambda^{\frac{1}{2}} U \equiv \begin{bmatrix}
  e^{(1)}(x_1) & e^{(1)}(x_2) & \cdots & e^{(1)}(x_N) \\
  e^{(2)}(x_1) & e^{(2)}(x_2) & \cdots & e^{(2)}(x_N) \\
  \vdots & \vdots & \ddots & \vdots \\
  e^{(N)}(x_1) & e^{(N)}(x_2) & \cdots & e^{(N)}(x_N)
\end{bmatrix}$$

$$E \equiv \begin{bmatrix}
  \overrightarrow{e}^{(1)} \\
  \overrightarrow{e}^{(2)} \\
  \vdots \\
  \overrightarrow{e}^{(N)}
\end{bmatrix}$$

Spectral Component 1
Spectral Component 2

$$\overrightarrow{e}^{(k)^T} \overrightarrow{e}^{(j)} = \begin{cases}
  0 & \text{if } k \neq j, \\
  \lambda_k & \text{if } k = j,
\end{cases}$$
2.2 Spectral Vector Space: $E$

$$E \equiv \begin{bmatrix} \vec{e}(x_1) & \vec{e}(x_2) & \cdots & \vec{e}(x_N) \end{bmatrix}$$

$$x_i \rightarrow \vec{e}(x_i)$$
Spectral Space: $E$

It has the following properties:

• It is a Hilbert space endowed with the conventional (linear) dot-product.

• It has a finite dimension, $N$, with ordered orthogonal bases.

• It is dependent of the training dataset. More precisely, $E$ should be referred to as the kernel spectral space with respect to the training dataset.
The Twin Spaces

\( \mathcal{F}: \) Basic Kernel Hilbert Space

\[
\sum_{k=1}^{K} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \phi(x_i) - \phi(x_j) \|^2
\]

\( \mathcal{E}: \) Kernel Spectral Space

\[
\sum_{k=1}^{K} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \mathbf{e}(x_i) - \mathbf{e}(x_j) \|^2
\]
\[ E: \text{Kernel Spectral Space} \]

The limiting case of KCA (i.e. with indefinite number of training patterns covering the entire vector space \( \mathbb{R}^N \)):

\[ F: \text{Basic Kernel Hilbert Space} \]

When \( N \) is extremely large with the training dataset uniformly distributed in the entire \( \mathbb{R}^N \), then

\[
\frac{\lambda'_1}{\lambda_1} = \frac{\lambda'_2}{\lambda_2} = \ldots = \frac{\lambda'_N}{\lambda_N}
\]
Preservation of Dot-Products

\[ \phi(x_i)^T \phi(x_j) = e(x_i)^T e(x_j) \]

K-Means on \( E \)
The spectral space $E$ is effective for dimensional reduction.

Note that, although both kernel Hilbert have ordered orthogonal bases, the latter is more effective for dimensional reduction since it takes into account of the distribution of the training dataset.

Given the equivalence between the two spaces, the (same) supervised/unsupervised learning results can be (more easily) computed via the finite-dimensional spectral space, as opposed to the indefinite-dimensional space $F$.

\textbf{e.g.} Spectral K-Means on $E$
XOR Dataset

\[
x_1 = \begin{bmatrix} +1 \\ +1 \end{bmatrix}, x_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, x_3 = \begin{bmatrix} +1 \\ -1 \end{bmatrix}, \text{ and } x_4 = \begin{bmatrix} -1 \\ +1 \end{bmatrix}
\]

\[
K(x, y) = (1 + x^T y)^2
\]

Empirical Vector

\[
k(x) = \begin{bmatrix} K(x_1, x) \\ K(x_2, x) \\ K(x_3, x) \\ K(x_4, x) \end{bmatrix} = \begin{bmatrix} (1 + u + v)^2 \\ (1 - u - v)^2 \\ (1 + u - v)^2 \\ (1 - u + v)^2 \end{bmatrix}
\]

\[
The\quad matrix\quad \Phi = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}
\]

\[
K = \Phi^T \Phi
\]
**XOR Dataset**

Kernel Matrix

\[ K = \Phi^T \Phi = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix} \]

**diagonalization**

\[ U^T \Lambda U = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \]

**spectral factorization**

\[ K = \begin{bmatrix} \sqrt{3} & \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \sqrt{3} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ \sqrt{3} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} \\ \sqrt{3} & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \end{bmatrix} \]
XOR Dataset

Spectral Space: $\mathbf{E}$

\[
\mathbf{e}(x) \equiv \Lambda^{-\frac{1}{2}} \mathbf{U} \mathbf{K}(x) = \begin{bmatrix}
\frac{1+u^2+v^2}{\sqrt{3}} \\
\sqrt{2}uv \\
\sqrt{2}u \\
\sqrt{2}v
\end{bmatrix}
\]

Ordered orthogonal components

\[
\mathbf{e}(x_1) = \begin{bmatrix} +\sqrt{3} \\
+\sqrt{2} \\
+\sqrt{2} \\
+\sqrt{2} \end{bmatrix},
\mathbf{e}(x_2) = \begin{bmatrix} +\sqrt{3} \\
+\sqrt{2} \\
-\sqrt{2} \\
-\sqrt{2} \end{bmatrix},
\mathbf{e}(x_3) = \begin{bmatrix} +\sqrt{3} \\
-\sqrt{2} \\
+\sqrt{2} \\
-\sqrt{2} \end{bmatrix},
\mathbf{e}(x_4) = \begin{bmatrix} +\sqrt{3} \\
-\sqrt{2} \\
-\sqrt{2} \\
+\sqrt{2} \end{bmatrix}.
\]
Now verify:

\[ \| \vec{\phi}(x_i) - \vec{\phi}(x_j) \| = \| \vec{e}(x_i) - \vec{e}(x_j) \| \]

\[ \| \vec{\phi}(x_1) - \vec{\phi}(x_2) \|^2 = K_{11} + K_{22} - 2K_{12} = 9 + 9 - 2 = 16. \]

\[
S = \begin{bmatrix}
1.73 & 1.41 & 1.41 & 1.41 \\
1.73 & 1.41 & -1.41 & -1.41 \\
1.73 & -1.41 & 1.41 & -1.41 \\
1.73 & -1.41 & -1.41 & 1.41 \\
\end{bmatrix} \begin{bmatrix}
1.73 & 1.73 & 1.73 & 1.73 \\
1.41 & 1.41 & -1.41 & -1.41 \\
1.41 & -1.41 & 1.41 & -1.41 \\
1.41 & -1.41 & -1.41 & 1.41 \\
\end{bmatrix}
\]

\[ \| \vec{e}(x_1) - \vec{e}(x_2) \| = 4. \]
XOR Dataset
Comparison of Clustering Analyses

Solution by K-means
\[
\{(x_1, x_3)(x_2, x_4)\}
\]
\[
\{(x_1, x_4)(x_2, x_3)\}
\]

Solution by Kernel K-means
\[
\{(x_1, x_3)(x_2, x_4)\}
\]
\[
\{(x_1, x_4)(x_2, x_3)\}
\]
\[
\{(x_1, x_2)(x_3, x_4)\}
\]
The empirical vector is defined as

\[ \vec{k}(x) \equiv \begin{bmatrix} K(x_1, x) \\ K(x_2, x) \\ \vdots \\ K(x_N, x) \end{bmatrix} \]

\[ K = \begin{bmatrix} \vec{k}(x_1) & \vec{k}(x_2) & \cdots & \vec{k}(x_N) \end{bmatrix} \]
XOR Dataset

4 original vectors
\[ x_1 = \begin{bmatrix} +1 \\ +1 \end{bmatrix}, x_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, x_3 = \begin{bmatrix} +1 \\ -1 \end{bmatrix}, \text{ and } x_4 = \begin{bmatrix} -1 \\ +1 \end{bmatrix} \]

empirical vector:
\[ \vec{k}(x) = \begin{bmatrix} K(x_1, x) \\ K(x_2, x) \\ K(x_3, x) \\ K(x_4, x) \end{bmatrix} = \begin{bmatrix} (1 + u + v)^2 \\ (1 - u - v)^2 \\ (1 + u - v)^2 \\ (1 - u + v)^2 \end{bmatrix} \]

4 empirical vectors
\[ \vec{k}(x_1) = \begin{bmatrix} 9 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{k}(x_2) = \begin{bmatrix} 1 \\ 9 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{k}(x_3) = \begin{bmatrix} 1 \\ 1 \\ 9 \\ 1 \end{bmatrix}, \quad \vec{k}(x_4) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 9 \end{bmatrix} \]
Empirical Space: $\mathcal{K}$

It has the following properties:

• It is a Hilbert space endowed with the conventional (linear) dot-product.

• It has a finite dimension, $N$, with non-orthogonal bases. (Thus, unlike the spectral space, it cannot enjoy the advantage of having ordered orthogonal bases.)

• It is dependent of the training dataset.
2.4 Linear Mapping:

“The Three Spaces”

\[
\vec{k}(x) = \begin{bmatrix}
K(x_1, x) \\
K(x_2, x) \\
\vdots \\
K(x_N, x)
\end{bmatrix} = \begin{bmatrix}
\vec{\phi}(x_1)^T \vec{\phi}(x) \\
\vec{\phi}(x_2)^T \vec{\phi}(x) \\
\vdots \\
\vec{\phi}(x_N)^T \vec{\phi}(x)
\end{bmatrix} = \Phi^T \vec{\phi}(x)
\]

\[
a^T \vec{k}(x) = a^T \Phi^T \vec{\phi}(x) = w^T \vec{\phi}(x)
\]

\[
w = \Phi a
\]

\[
\vec{e}(x) \equiv \Lambda^{-\frac{1}{2}} U \vec{k}(x) \\
\vec{k}(x) = U^T \Lambda^{\frac{1}{2}} \vec{e}(x) = \mathbf{E}^T \vec{e}(x)
\]

\[
a^T \vec{k}(x) = a^T \mathbf{E}^T \vec{e}(x) = v^T \vec{e}(x)
\]

\[
v = \mathbf{E} a
\]
Linear Mapping:
Introduce a unifying Vector “z”

\[ w^T \vec{\phi}(x) = v^T \vec{e}(x) = a^T \vec{k}(x) \]

\[ \Phi^T w = \Phi^T \Phi a = K a \]

\[ E^T v = E^T E a = K a \]

\[ z = \Phi^T w = E^T v = K a \]
Linear Mapping:

Kernel PCA

\[
\max_w \frac{w^T \Phi \Phi^T w}{w^T w} = \max_v \frac{v^T E E^T v}{v^T v} = \max_a \frac{a^T K^2 a}{a^T K a}
\]

\[
w^T \Phi \Phi^T w = a^T K^2 a
\]

\[
w^T w = a^T K a
\]

\[
Ka = \lambda_1 a
\]

\[
\frac{a^T K^2 a}{a^T K a} = \frac{\lambda_1^2 a^T a}{\lambda_1 a^T a} = \lambda_1
\]
Linear Mapping:

Representation Theorem

A kernel solution $w$ can always be represented by

\[ w = \Phi a \]

The discriminant function (for SVM or KFD) can be expressed as

\[ f(x) = w^T \phi(x) + b = a^T \Phi^T \phi(x) + b = a^T k(x) + b \]
None-Preservation of Dot-Products

\[ \vec{k}(x_i)^T \vec{k}(x_j) = \vec{e}(x_i)^T \Lambda^{1/2} U^T U \Lambda^{1/2} \vec{e}(x_j) \]
\[ = \vec{e}(x_i)^T \Lambda \vec{e}(x_j) \neq K_{ij} \]

Consequently, it opens up a new and promising clustering strategy, termed "empirical K-means".
Topics:

1. Kernel-Based Unsupervised/supervised Machine Learning: Overview

2. Kernel Based Representation Spaces

3. Kernel Spectral Analysis
   - 3.1 Conventional PCA
   - 3.2 Kernel PCA

4. Kernel Fisher Discriminant (KFD) analysis

5. Kernel-based Support Vector Machine and Robustification of Training

6. Fast Clustering Methods for Positive-Definite Kernels (Kernel Component Analysis and Kernel Trick)

7. Extension to Nonvectorial Data and NPD (None-Positive-definite) Kernels
3.1 Conventional PCA

\[ C_{\text{conventional}} = \sum_{n=1}^{N} x_n x_n^T = XX^T \]

Therefore, the PCA may be solved by applying eigenvalue decomposition (i.e. diagonalization) to

as opposed to

\[ S = X^T X \]
For the four XOR training data,

\[
XX^T = \begin{bmatrix}
+1 & -1 & +1 & -1 \\
+1 & -1 & -1 & +1
\end{bmatrix}
\begin{bmatrix}
+1 & +1 \\
-1 & -1 \\
+1 & -1 \\
-1 & +1
\end{bmatrix}
= \begin{bmatrix}
4 & 0 \\
0 & 4
\end{bmatrix}
\]

which has two eigenvalues 4.0 and 4.0.

The corresponding pairwise score matrix is

\[
X^T X = \\
\begin{bmatrix}
+1 & +1 \\
-1 & -1 \\
+1 & -1 \\
-1 & +1
\end{bmatrix}
\begin{bmatrix}
+1 & -1 & +1 & -1 \\
+1 & -1 & -1 & +1
\end{bmatrix}
= \begin{bmatrix}
+2 & -2 & 0 & 0 \\
-2 & +2 & 0 & 0 \\
0 & 0 & +2 & -2 \\
0 & 0 & -2 & +2
\end{bmatrix}
\]

The two nonzero eigenvalues are again 4.0 and 4.0, just the same as before.
3.2 Kernel Spectral Analysis[32]

The kernel PCA basically amounts to the spectral decomposition of $K$. 
The kernel PCA basically amounts to the spectral decomposition of $K$.

$$S_{\text{kernel}} = \sum_{n=1}^{N} \vec{\phi}(x_n) \vec{\phi}(x_n)^T = \Phi \Phi^T$$

$$K a^{(m)} = \lambda^{(m)} a^{(m)}$$

$$\lambda^{(m)} \Phi a^{(m)} = \Phi K a^{(m)}$$

$$\Phi^T \Phi = K \quad \Rightarrow \quad \lambda^{(m)} \Phi a^{(m)} = \Phi \Phi^T \Phi a^{(m)} = S_{\text{kernel}} \Phi a^{(m)}$$

Reconfirms the Representer Theorem:

$$w^{(m)} = \Phi a^{(m)}$$
Finding the $m$-th (kernel) components:

\[ w^T \phi(x) = v^T \epsilon(x) = a^T k(x); \]

set of training data:

\[ Ka^{(m)} \]
Kernel PCA

\[ \mathbf{w}^{(m)T} \mathbf{w}^{(j)} = \mathbf{w}^{(m)T} \Phi^T \Phi \mathbf{a}^{(j)} = \mathbf{a}^{(m)T} \mathbf{K} \mathbf{a}^{(j)} = 0 \]

\[ \mathbf{w}^{(m)T} \mathbf{w}^{(m)} = \mathbf{a}^{(m)T} \mathbf{K} \mathbf{a}^{(m)} = 1 \quad \Rightarrow \quad \| \mathbf{a}^{(m)} \|^2 = \frac{1}{\lambda^{(m)}} \]

The eigenvector property can be confirmed as follows:

\[
\begin{bmatrix}
\mathbf{a}^{(1)T} \\
\mathbf{a}^{(2)T} \\
\vdots \\
\mathbf{a}^{(N)T}
\end{bmatrix}
K = \Lambda^{-\frac{1}{2}} \mathbf{U} \mathbf{K} = \Lambda^{-\frac{1}{2}} \mathbf{U} \mathbf{U}^T \Lambda \mathbf{U} = \Lambda [\Lambda^{-\frac{1}{2}}] \mathbf{U} = \Lambda
\]

KCA

\[
\begin{bmatrix}
\mathbf{a}^{(1)T} \\
\mathbf{a}^{(2)T} \\
\vdots \\
\mathbf{a}^{(N)T}
\end{bmatrix}
K = [\Lambda^{\frac{1}{2}}] \mathbf{U} = \mathbf{E}.
\]
Kernel PCA

From the spectral factorization perspective

\[
K = [\Lambda^{\frac{1}{2}}]U = E.
\]

\[
EE^T = \Lambda^{\frac{1}{2}}UU^T \Lambda^{\frac{1}{2}} = \Lambda
\]

\[
v_m = [0 \cdots 0 \ 1 \ 0 \cdots 0]
\]

\[
w^{(m)T} \phi(x) = v^{(m)T} e(x) = e^{(m)}(x)
\]
Kernel PCA: XOR Dataset

\[ S = \Phi^T \Phi = \begin{bmatrix}
9 & 1 & 1 & 1 \\
1 & 9 & 1 & 1 \\
1 & 1 & 9 & 1 \\
1 & 1 & 1 & 0
\end{bmatrix} \]

\[ K = U^T \Lambda U = \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2}
\end{bmatrix} \begin{bmatrix}
12 & 0 & 0 & 0 \\
0 & 8 & 0 & 0 \\
0 & 0 & 8 & 0 \\
0 & 0 & 0 & 8
\end{bmatrix} \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2}
\end{bmatrix} \]

\[ K = \sqrt{3} \begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix} \begin{bmatrix}
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}}
\end{bmatrix} \]
Kernel PCA

The two principal eigenvectors

\[
\begin{bmatrix}
  a^{(1)T} \\
  a^{(2)T}
\end{bmatrix} = \begin{bmatrix}
  \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} \\
  \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}}
\end{bmatrix}
\]

\[\|a^{(1)}\|^2 = \frac{1}{\lambda^{(1)}} = \frac{1}{12}, \quad \|a^{(2)}\|^2 = \frac{1}{\lambda^{(2)}} = \frac{1}{8};\]

The two principal components

\[
\begin{bmatrix}
  a^{(1)T} \\
  a^{(2)T}
\end{bmatrix} [K] = \begin{bmatrix}
  \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} \\
  \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}}
\end{bmatrix} \begin{bmatrix}
  9 & 1 & 1 & 1 \\
  1 & 9 & 1 & 1 \\
  1 & 1 & 9 & 1 \\
  1 & 1 & 1 & 9
\end{bmatrix}
\]

\[= \begin{bmatrix}
  \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} \\
  \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2}
\end{bmatrix}\]
PCA vs. Kernel PCA

○: positive
×: negative

(-1, +1)
(+1, +1)
(-1, -1)
(+1, -1)

○: positive
×: negative

(+ \sqrt{3}, +\sqrt{2})
(+ \sqrt{3}, +\sqrt{2})
(+ \sqrt{3}, -\sqrt{2})
(+ \sqrt{3}, -\sqrt{2})

First PCA Component
Second PCA Component
First Kernel PCA Component
Second Kernel PCA Component
PCA vs. Kernel PCA

(The figures below were downloaded from Wiki.)

Cornering Effect of Kernel Hilbert Space
<table>
<thead>
<tr>
<th></th>
<th>PCA</th>
<th>Kernel-PCA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel</strong></td>
<td>linear: $x^T y$</td>
<td>nonlinear: $K(x, y)$</td>
</tr>
<tr>
<td><strong>covariance matrix</strong></td>
<td>$S_{\text{conventional}}$</td>
<td>$S_{\text{kernel}}$</td>
</tr>
<tr>
<td><strong>pairwise matrix</strong></td>
<td>${K_{ij} = x_i^T x_j}$</td>
<td>${K_{ij} = K(x_i, x_j)}$</td>
</tr>
<tr>
<td><strong>1st Principal Component</strong></td>
<td>$e^{(1)^T} x$</td>
<td>$a^{(1)^T} \bar{k}(x)$</td>
</tr>
</tbody>
</table>
Topics:
1. Kernel-Based Unsupervised/supervised Machine Learning: Overview
2. Kernel Based Representation Spaces
3. Principal Component analysis (Kernel PCA)

4. Supervised Kernel Classifiers
   4.1 Fisher Discriminant Analysis (FDA)
   4.2 Kernel Fisher Discriminant (KFD)
   4.3 Perfect KFD
   4.4 Perturbed Fisher Discriminant (PFD)
      - Stochastic Modeling of Training Dataset
   4.5 Hybrid Fisher-SVM Approach

5. Fast Clustering Methods for Positive-Definite Kernels

6. Extension to Nonvectorial Data and NPD (None-Positive-definite) Kernels.
4.1 Fisher Discriminant Analysis (FDA) [8] = Linear Discriminant Analysis (LDA)

In the traditional LDA,

**Total Scatter Matrix**

\[ S = \sum_{i=1}^{N} [x_i - \mu][x_i - \mu]^T \]

**Between-Class Scatter Matrix**

\[ S_B = N_+ [\mu_+ - \mu][\mu_+ - \mu]^T + N_- [\mu_- - \mu][\mu_- - \mu]^T \]

**Within-Class Scatter Matrix**

\[ S_W = \sum_{i: y_i = +1} [x_i - \mu_+][x_i - \mu_+]^T + \sum_{i: y_i = -1} [x_i - \mu_-][x_i - \mu_-]^T \]

\[ S = S_W + S_B \]
Traditional FDA, • Data-Distribution-Dependent

\[ J'(w) = \frac{\text{signal}}{\text{noise}} = \frac{w^T S_B w}{w^T S_W w} \]

• Range: \([0, \infty]\)

A useful variant,

\[ J(w) = \frac{\text{signal}}{\text{signal} + \text{noise}} = \frac{w^T S_B w}{w^T (S_B + S_W) w} = \frac{w^T S_B w}{w^T S w} \]

• Range: \([0,1]\)

\[ J(w) = \frac{1}{1 + \frac{1}{J'(w)}} \]

both share the same optimal \(w\).

Assume that \(N > M\), so that \(S\) is invertible:

\[ \mathbf{w}_{\text{opt}} = S^{-1} [\mu_+ - \mu_-] \]
By definition:

\[ S_B = N_+ [\mu_+ - \mu][\mu_+ - \mu]^T + N_- [\mu_- - \mu][\mu_- - \mu]^T \]

Note that

\((\mu, \mu_+, \text{and } \mu_-)\) are collinear

More precisely,

\[
\begin{align*}
[\mu_+ - \mu] &= \frac{N_-}{N} [\mu_+ - \mu_-] \\
[\mu_- - \mu] &= -\frac{N_+}{N} [\mu_+ - \mu_-]
\end{align*}
\]

\[
S_B = \left\{(\frac{N_-}{N} \sqrt{N_+})^2 + (\frac{N_+}{N} \sqrt{N_-})\right\}[\mu_+ - \mu_-][\mu_+ - \mu_-]^T \\
&= \frac{N_-N_+}{N} [\mu_+ - \mu_-][\mu_+ - \mu_-]^T,
\]
Denote that
\[ \mathbf{d} = d_+ \mathbf{e}_+ - d_- \mathbf{e}_- \]

It follows that
\[ \mathbf{X} \mathbf{d} = \frac{\sqrt{N_+ N_-}}{\sqrt{N}} \mathbf{X} \left[ \frac{\mathbf{e}_+}{N_+} - \frac{\mathbf{e}_-}{N_-} \right] = \frac{\sqrt{N_+ N_-}}{\sqrt{N}} [\mu_+ - \mu_-]. \]

Recall that
\[ S_B = \frac{N_- N_+}{N} [\mu_+ - \mu_-][\mu_+ - \mu_-]^T \]

\[ \Rightarrow S_B = \mathbf{X} \mathbf{d} \mathbf{d}^T \mathbf{X}^T \]

Recall that \( \mathbf{S} = \mathbf{X} \mathbf{X}^T \), it follows that
\[ J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{X} \mathbf{Q} \mathbf{X}^T \mathbf{w}}{\mathbf{w}^T \mathbf{X} \mathbf{X}^T \mathbf{w}} \]

where \( \mathbf{Q} = \mathbf{d} \mathbf{d}^T \)
Closed-Form Solution of FDA (M≥N):

Fisher Score

\[ J(w) = \frac{w^T \bar{X}dd^T\bar{X}^T w}{w^T \bar{X}\bar{X}^T w} \]

Generically,

\[ z_{opt} = d \]

where

\[ d = \frac{\sqrt{N_-}}{\sqrt{N \times N_+}} e_+ - \frac{\sqrt{N_+}}{\sqrt{N \times N_-}} e_- \]

\[ J_{opt} = \frac{d^T dd^T d}{d^T d} = \|d\|^2 = 1 \]
The optimal solution can be expressed as

\[
\bar{X}^T = \begin{bmatrix} R & 0 \end{bmatrix} Q
\]

\[
w_{\text{opt}} = Q^T \begin{bmatrix} R^{-1} & d \\ 0 & 0 \end{bmatrix}
\]

\[
\bar{X}^T w_{\text{opt}} = \begin{bmatrix} R & 0 \end{bmatrix} QQ^T \begin{bmatrix} R^{-1} & d \\ 0 & 0 \end{bmatrix} = \mathbf{d}
\]

\[
w_{\text{opt}}^T x = \begin{cases} 
\frac{\sqrt{N_-}}{\sqrt{N \times N_+}} & \text{for the positive hyperplane} \\
-\frac{\sqrt{N_+}}{\sqrt{N \times N_-}} & \text{for the negative hyperplane}
\end{cases}
\]

\[
J(w_{\text{opt}}) = \frac{\text{signal}}{\text{signal} + \text{noise}} = \frac{\text{signal}}{\text{signal}} = 1
\]
4.2 KFD Analysis

In the KFD Analysis, we shall assume the generic situations that training vectors are distinctive and other pathological distributions are excluded.

References [22–25]
“universal” vector: $z$

$$f(x) = w^T \phi(x) + b = v^T e(x) + b = a^T k(x) + b.$$

$$z = \Phi^T w = E^T v = K a$$

Most common mappings:

$$z = E^T v \quad v = E a$$
“Data” Matrix in Spectral Space: \( E \)

\[
K = U^T \Lambda U = U^T \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} U = E^T E
\]

Pre-centered Matrix:

\[
\bar{E} = E \left( I - \frac{e e^T}{N} \right)
\]

\[
J = \frac{v^T \bar{E} d d^T \bar{E}^T v}{v^T \bar{E} \bar{E}^T v}
\]
\[ d = d_+e_+ - d_-e_- \]

\[ d_+ = \frac{\sqrt{N_-}}{\sqrt{N \times N_+}} \text{ and } d_- = \frac{\sqrt{N_+}}{\sqrt{N \times N_-}}. \]

\[ e^T d = 0 \quad \Rightarrow \quad \left( I - \frac{ee^T}{N} \right) d = d \]

\[ J = \frac{v^T \bar{E}dd^T\bar{E}^Tv}{v^T \bar{E}\bar{E}^Tv} = \frac{v^T \bar{E}dd^T\bar{E}^Tv}{v^T \bar{E}(I - \frac{ee^T}{N})\bar{E}^Tv} \]

\[ z = E^Tv \quad \Rightarrow \quad J = \frac{z^Tdd^Tz}{z^T(I - \frac{ee^T}{N})z} \]
4.3 Perfect KFD
A perfect KFD solution expressed in the “universal” vector space is:

\[ J = \frac{z^Tdd^Tz}{z^T(I - \frac{ee^T}{N})z} \]

\[ z_{\text{opt}} = d. \]

\[ e^T d = 0 \]

\[ J_{\text{opt}} = \frac{d^Tdd^Td}{d^Td} = \|d\|^2 = 1 \]
Note that $J = 1$ implies that

Within-Class Scatter Matrix $\rightarrow 0$, after the projection.

Namely, the positive and negative training vectors fall on two separate but parallel hyperplanes.
Positive & Negative Hyperplanes.

\[ \mathbf{v}_{\text{opt}} = \mathbf{E}^{-T} \mathbf{d} \quad \Rightarrow \quad \mathbf{v}_{\text{opt}}^T \mathbf{E} = \mathbf{d}^T \]

The entries of \( \mathbf{d} \) have two different values:

- \( d_+ \) for all the positive training patterns.
- \( d_- \) for all the negative training patterns.

\[ \mathbf{v}_{\text{opt}}^T \overrightarrow{\mathbf{e}}(\mathbf{x}) = \begin{cases} d_+ & \text{for positive hyperplane} \\ d_- & \text{for negative hyperplane} \end{cases} \]
The optima are Data-Distribution-Dependent.

\[ \tilde{J}(w) = \frac{w^T S_B w}{w^T S w} = 0 \]
KFD: XOR Example

\[ E = \begin{bmatrix} \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \end{bmatrix} \quad \rightarrow \quad \tilde{E} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \end{bmatrix} \]

Because \( E \) is Data-Distribution-Dependent.

\[ z_{opt} = \left[ I - \frac{1}{N} ee^T \right] d = \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix} \]

Data-Distribution-Independent.

Because \( E \) is Data-Distribution-Dependent.

\[ v_{opt} = E^{-T} \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \sqrt{2} \\ 0 \\ 0 \end{bmatrix} \]

Data-Distribution-Dependent.
KFD: XOR Example

\[ J_\varepsilon(v) = \frac{v^T \overline{E} d d^T \overline{E}^T v}{v^T \overline{E} \overline{E}^T v} = \frac{\left[ \begin{array}{cccc} 1/2 & 1/2 & -1/2 & -1/2 \\ -1/2 & 1/2 & -1/2 & -1/2 \end{array} \right] \left[ \begin{array}{c} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{array} \right]^2}{\left[ \begin{array}{cccc} 1/2 & 1/2 & -1/2 & -1/2 \\ -1/2 & 1/2 & -1/2 & -1/2 \end{array} \right]} = 1. \]

Confirmed for this example:

The optima are Data-Distribution-Independent.

Data-Distribution-Independence

First example:

- $x_1$: (1.1, 0.8)
- $x_2$: (1.1, 0.8)
- $x_3$: (-1, -1)
- $x_4$: (-1.1, -0.8)
\[ K = \]
\[
\begin{bmatrix}
9.0000 & 8.4100 & 1.0000 & 0.8100 \\
8.4100 & 8.1225 & 0.8100 & 0.7225 \\
1.0000 & 0.8100 & 9.0000 & 8.4100 \\
0.8100 & 0.7225 & 8.4100 & 8.1225 \\
\end{bmatrix}
\]

\[ [U^T, \Lambda] = \text{eigs}(K) \]
\[ U^T = \]
\[
\begin{bmatrix}
0.5154 & -0.5098 & -0.4841 & 0.4900 \\
0.4841 & -0.4900 & 0.5154 & -0.5098 \\
0.5154 & 0.5098 & -0.4841 & -0.4900 \\
0.4841 & 0.4900 & 0.5154 & 0.5098 \\
\end{bmatrix}
\]
\[ \Lambda = \]
\[
\begin{bmatrix}
18.6606 & 0 & 0 & 0 \\
0 & 15.3059 & 0 & 0 \\
0 & 0 & 0.1844 & 0 \\
0 & 0 & 0 & 0.0941 \\
\end{bmatrix}
\]

\[ \text{LambdaSqrt} = \]
\[
\begin{bmatrix}
4.3198 & 0 & 0 & 0 \\
0 & 3.9123 & 0 & 0 \\
0 & 0 & 0.4294 & 0 \\
0 & 0 & 0 & 0.3067 \\
\end{bmatrix}
\]

\[ E = \text{LambdaSqrt} \times U = \]
\[
\begin{bmatrix}
2.2264 & 2.0913 & 2.2264 & 2.0913 \\
-1.9943 & -1.9172 & 1.9943 & 1.9172 \\
-0.2079 & 0.2213 & -0.2079 & 0.2213 \\
0.1503 & -0.1564 & -0.1503 & 0.1564 \\
\end{bmatrix}
\]

\[ \text{data-centered} E = ee = \]
\[
\begin{bmatrix}
0.0675 & -0.0675 & 0.0675 & -0.0675 \\
-1.9943 & -1.9172 & 1.9943 & 1.9172 \\
-0.2146 & 0.2146 & -0.2146 & 0.2146 \\
0.1503 & -0.1564 & -0.1503 & 0.1564 \\
\end{bmatrix}
\]

\[ \text{denominator_matrix} = ee \times ee' = \]
\[
\begin{bmatrix}
0.0182 & 0.0000 & -0.0580 & 0.0000 \\
0.0000 & 15.3059 & 0.0000 & 0.0000 \\
-0.0580 & 0.0000 & 0.1842 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0941 \\
\end{bmatrix}
\]
```plaintext
>> data-centered E=

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0675</td>
<td>-0.0675</td>
<td>0.0675</td>
<td>-0.0675</td>
</tr>
<tr>
<td>-1.9943</td>
<td>-1.9172</td>
<td>1.9943</td>
<td>1.9172</td>
</tr>
<tr>
<td>-0.2146</td>
<td>0.2146</td>
<td>-0.2146</td>
<td>0.2146</td>
</tr>
<tr>
<td>0.1503</td>
<td>-0.1564</td>
<td>-0.1503</td>
<td>0.156</td>
</tr>
</tbody>
</table>
```
KFD: Numerical Analysis

In general,

\[ \begin{bmatrix} x & x & \ldots & x & x \end{bmatrix}^T \]

each element has its own sensitivity.

Two different decision and two different scenarios

\[ \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T \text{ is numerically stable, as it is robust w.r.t. perturbation} \]

\[ \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T \text{ is numerically risky, as small perturbation could cause big effect.} \]
Easy Case

\[
\begin{bmatrix}
\rho & \eta_{opt} \\
1.0000 & 0.0000
\end{bmatrix}
\]

\[
\begin{bmatrix}
score_{ve} & score_{hybrid\_AR} & score_{hybrid\_dR} & score_{vopt}
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.9351 & 0.9383 & 0.9383 & 0.9383
\end{bmatrix}
\]

\[v = [\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}]^{T}\]

\[
\text{Fisher Score} = 0.9996
\]

\[
\begin{bmatrix}
\rho & \eta_{opt} \\
1.0000 & 0.0000
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{score}_{ve} & \text{score}_{hybrid\_AR} & \text{score}_{hybrid\_dR} & \text{score}_{vopt}
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.9351 & 0.9383 & 0.9383 & 0.9383
\end{bmatrix}
\]

\[
\text{Imperfect FDA score} = 0.9997
\]
Hard Case

\[
\text{>> numerator_matrix} = ee*d*d'*ee'
\]

\[
\begin{bmatrix}
0.0182 & -0.0000 & -0.0580 & 0.0000 \\
-0.0000 & 0.0000 & 0.0000 & -0.0000 \\
-0.0580 & 0.0000 & 0.1843 & -0.0000 \\
0.0000 & -0.0000 & -0.0000 & 0.0000 \\
\end{bmatrix}
\]

\[
\text{>> } v = [0 \ 0 \ 1 \ 0]^T
\]

\[
\text{>> } v = [0 \ 0 \ 1 \ 0]^T
\]

\[
\text{>> Fisher Score} = 1.
\]

\[
\text{>> Fisher Score} = 1.
\]

\[
\begin{bmatrix}
\rho \ \eta_{\text{opt}}
\end{bmatrix}
\begin{bmatrix}
1.0000 \\ -0.2401
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{score}_\text{ve} \ \text{score}_\text{hybrid_AR} \ \text{score}_\text{hybrid_dR} \ \text{score}_\text{vopt}
\end{bmatrix}
\begin{bmatrix}
0.1684 \\ 0.0919 \\ 0.1571 \\ 0.1684
\end{bmatrix}
\]

“Imperfect” FDA score = 1.0000
Not as Hard Case
- poorest result

\[
\begin{bmatrix}
\rho & \eta_{\text{opt}} \\
1.0000 & 0.0000
\end{bmatrix}
\]

\[
\begin{bmatrix}
score_{\text{ve}} & score_{\text{hybrid AR}} & score_{\text{hybrid dR}} & score_{\text{vopt}} \\
0.0860 & 0.0863 & 0.0863 & 0.0863
\end{bmatrix}
\]

\[
> \text{numerator\_matrix} = \ee \ast d \ast d' \ast \ee'
\]

\[
\begin{bmatrix}
0.0000 & -0.0000 & 0.0000 & 0.0000 \\
-0.0000 & 0.0060 & -0.0000 & -0.0237 \\
0.0000 & -0.0000 & 0.0000 & 0.0000 \\
0.0000 & -0.0237 & 0.0000 & 0.0940
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 1
\end{bmatrix}^T
\]

\[
> \text{v} = [0 \ 0 \ 0 \ 1]^T
\]

\[
> \text{Fisher Score} = 1.
\]

\[
\begin{bmatrix}
\rho & \eta_{\text{opt}} \\
1.0000 & 0.0000
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{score\_ve} & \text{score\_hybrid\_AR} & \text{score\_hybrid\_dR} & \text{score\_vopt} \\
0.0860 & 0.0863 & 0.0863 & 0.0863
\end{bmatrix}
\]

\[
\vopt_{\text{dR}} =
\]

\[
\begin{bmatrix}
-0.0000 \\
0.0047 \\
-0.0000 \\
0.2803
\end{bmatrix}
\]

Imperfect FDA score = 0.9634
Solution Space of Perfect KFDs

\[ z = d - \xi e. \]

It may be further verified that a vector is a perfect KFD if and only if it falls on the two-dimensional solution space prescribed above.
Solution Space of Perfect KFDs

Illustration of why both the numerator and denominator are pictorially shown to be equal to 1.

\[ z = d - \xi e \]

\[ (I - \frac{ee^T}{N}) z \]
Invariant Properties

The numerator remains constant:

\[(d^T - \xi e^T)dd^T (d - \xi e) = 1\]

The denominator remains constant:

\[(d^T - \xi e^T)(I - \frac{ee^T}{N}) (d - \xi e) = 1\]
Selection of decision threshold: \( b \)

\[
f(x) = v^T \overrightarrow{e}(x) + b = 0
\]

To create a maximum margin between the training vectors to the decision hyperplane, the threshold is set as the average of the two projected centroids (instead of the projected mass center).

\[
b = \xi - \xi_0
\]

\[
\xi_0 = \frac{d_+ + d_-}{2}
\]
Two Case Studies:

Case I:

\[ \xi = 0: \ z = d \]

\[ b = \xi - \xi_0 \]

Case II:

\[ \xi = \xi_0 \]

\[ z = d - \xi e = \frac{d_+ - d_-}{2} y \]

\[ b = 0 \]

This self-centered solution matches the SVM approach.
Margin of Separation

However, different perfect KFDs have their own margins of separation which have a simple formula:

\[
\frac{d_+ - d_-}{\|v_{opt}\|} = \sqrt{\frac{N}{N_+ \times N_- \|v_{opt}\|}}
\]

This plays a vital role in the perturbation analysis, since the wider is the separation the more robust is the classifier.
Margin of Separation

○: positive

×: negative

(+ $\sqrt{3}$, $\sqrt{2}$)
(+ $\sqrt{3}$, $\sqrt{2}$)

(+ $\sqrt{3}$, $-\sqrt{2}$)
(+ $\sqrt{3}$, $-\sqrt{2}$)

First Kernel PCA Component

Second Kernel PCA Component
Maximum Separation by Perfect KFD

The solution can be obtained by minimization of

\[ \| \mathbf{v}_{\text{opt}} \| \]

\[ z_{\text{max-margin}} = d - \xi e = d - \frac{d^T K^{-1} e}{e^T K^{-1} e} e \]

This perfect KFD has a maximal PFD among its own kind.
4.4 Perturbed Fisher Discriminant (PFD)

perfect is no optimal!

optimal is no perfect!
Stochastic Modeling of Training Dataset

Now each training pattern is viewed as an instantiation of a noisy measurement of the (noisy-free) signal, i.e. the data matrix associated with the robustified (noise-added) training patterns can be expressed as:

\[ X + N \]

\[
E[(X + N)(X + N)^T] = E[(X)(X)^T] + \rho^2 I
\]

With some extensive analysis, the following can be established for the robustification in the spectral space

\[
EE^T + \rho^2 I = \Lambda + \rho^2 I
\]

\[
K' = [K + \rho I]
\]
Say No to Conventional KFD or Perfect KFD.

Say Yes to PFD!

Application Motivation:
Stochastic Model of Training Dataset
Perturbed Fisher Discriminant

Simulation Supports
Perturbed Fisher Discriminant

\[ J_{\text{PFD}} = \frac{v^T E d d^T E^T v}{v^T [E (I - \frac{ee^T}{N}) E^T + \rho I] v} \]
Maximal PFD Among Perfect KFDs

PFD for Perfect KFD

\[ J_{PF D} = \frac{1}{1 + \rho \| \mathbf{v} \|^2} \]

Best Perfect KFD from PFD perspective:

\[ z_{\text{min-amp}} = z_{\text{max-margin}} = d - \frac{d^T K^{-1} e}{e^T K^{-1} e} e \]
Perfect KFD vs. Imperfect KFD

\[ \text{total} = \text{signal} + \text{noise} \]

- Perfect KFD
- Imperfect KFD

Diagram showing signal, noise, and mass center with vectors and angles.
Globally Optimal PFD Solution

\[ v_{\text{opt}} = [\mathbf{E}(I - \frac{\mathbf{e}\mathbf{e}^T}{N})\mathbf{E}^T + \rho\mathbf{I}]^{-1}\mathbf{E}\mathbf{d} \]

\[ v_{\text{opt}} = [\Lambda + \rho\mathbf{I}]^{-1}\mathbf{E}[\mathbf{d} - \eta\mathbf{e}] \]

where

\[ \eta = \frac{\mathbf{e}'^T\mathbf{d}}{\mathbf{e}'^T\mathbf{e} - 1} \]

and

\[ \mathbf{e}'^T = N^{-1}\mathbf{e}^T\mathbf{E}^T[\Lambda + \rho\mathbf{I}]^{-1}\mathbf{E} \]
Dataset2: (easy case):

score(best perfect) score(AR) score(vopt)

\[
\begin{array}{cccc}
\text{rho=.04:} & 0.9825 & 0.9880 & 0.9891 \\
\text{rho=.4:} & 0.8486 & 0.9081 & 0.9292 \\
\end{array}
\]
Dataset5 (hard case):

score(best perfect) score(AR) score(vopt)

rho=0.04:  0.5448   0.6283   0.6910
rho=0.4:   0.1069   0.1983   0.5190
Simulations: compare $z = d$, and optimal $z$ 5 points,

(1) for $\rho = 1$: $PFD = 0.8017$ versus $0.8464$, with $KFD = 0.9933$;

(2) for $\rho = 0.1$: $PFD = 0.9675$ versus $0.9778$, with $KFD = 0.9959$; and

(3) again for $\rho = 0.1$, $PFDs$ are $0.9573$ and $0.9734$ for perfect optimal $KFD$ and regularized perfect $KFD$ respectively.
4.5 Hybrid Fisher-SVM
Computationally speaking, when all the three spaces are basically equivalent, e.g. KFD and SVM, the empirical space is the most attractive, since the spectral space incurs the extra cost for spectral decomposition.
Positive & Negative Hyperplanes: in $\mathbb{K}$

$$J = \frac{a^T \bar{K} d d^T \bar{K}^T a}{a^T \bar{K} \bar{K}^T a} = \frac{a^T K d d^T (K^T a)}{a^T K (I - \frac{e e^T}{N}) K^T a}$$

$$z_{\text{perfect}} = d$$

$$a_{\text{perfect}} = K^{-T} d$$

$$k_{\text{perfect}}^T \tilde{k}(x) = \begin{cases} 
  d_+ & \text{for the positive hyperplane} \\
  d_- & \text{for the negative hyperplane.}
\end{cases}$$

It is perfect, but not optimal!
The Optimal PFD Solution

\[ a_{\text{opt}} = E^{-1}v_{\text{opt}} = K'^{-1} (d - \eta d) \]

\[ K' = [K + \rho I] \]

Regulation by stochastic model of data.

\[ \text{Derivation:} \quad a_{\text{opt}} = E^{-1}v_{\text{opt}} \]
\[ = E^{-1}[\Lambda + \rho I]^{-1}E[d - \eta e] \]
\[ = U^T \Lambda^{-\frac{1}{2}}[\Lambda + \rho I]^{-1} \Lambda^{\frac{1}{2}} U[d - \eta e] \]
\[ = K^T [U^T \Lambda U + U^T \rho I U]^{-1} [d - \eta e] \]
\[ = [K + \rho I]^{-1} [d - \eta e]. \]
An alternative and effective regularization approach is via imposition of constraints in the Wolf optimization, a technique adopted by kernel-SVM classifiers to alleviate over-training problems.
Linear SVM

$$w = \Phi a$$

$$a \equiv \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} \equiv \begin{bmatrix} y_1 \alpha_1 \\ \vdots \\ y_N \alpha_N \end{bmatrix}$$

Lagrange multipliers can be solved by the Wolfe optimization:

$$\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \right\}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \ i = 1, \ldots, N$$
Kernel SVM

Since the dot-product is changed to:

\[
\vec{\phi}(x_i)^T \vec{\phi}(x_j) = e(x_i)^T e(x_j) = K_{ij}
\]

Solution by the Wolfe optimization:

\[
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right\}
\]

subject to

\[
\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \ i = 1, \ldots, N
\]
Ignoring for a moment the constraints on $a_i$, the optimal solution can be obtained by taking the derivative of the above with respect to $a_i$. This leads to:

$$
\max_{a_i} \left\{ \sum_{i=1}^{N} a_i y_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K_{ij} \right\} = \max_{a} \left\{ a^T y - \frac{1}{2} a^T Ka \right\}
$$

$$Ka = y
$$

$$a = \begin{bmatrix} a_1 \\
\vdots \\
a_N \end{bmatrix} \equiv \begin{bmatrix} y_1 \alpha_1 \\
\vdots \\
y_N \alpha_N \end{bmatrix}$$
XOR Example

Linear System Solution

Nonsingular $K$ and exact solution:

$$\begin{bmatrix} \frac{1}{8} & \frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} \end{bmatrix}$$

KFD Solution

$$a_{\text{perfect}} = K^{-T}d$$

$$z_{\text{opt}} = [I - \frac{1}{N}ee^T]d = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$a_{\text{opt}} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{8\sqrt{2}} \\ \frac{1}{8\sqrt{2}} \\ \frac{1}{8\sqrt{2}} \\ \frac{1}{8\sqrt{2}} \end{bmatrix}$$

$$f(x) = a^T k(x) = uv = 0$$
Hybrid Fisher-SVM Classifier

A Hybrid Classifier Combining KFD and SVM

(1) On one hand, the cost function in the optimization formulation follows that of KFD.

(2) On the other hand, a second regularization - based on the Wolfe optimization adopted by SVM - is incorporated.
Hybrid Fisher-SVM Classifier

$$\max_{a_i} \left\{ \sum_{i=1}^{N} a_i (d_i - \eta) - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K'_{i,j} \right\} = \max_{a} \left\{ a^T d - \frac{1}{2} a^T K'a \right\}$$

subject to $\sum_{i=1}^{N} a_i = 0$, $0 \leq a_i \leq d_+ C$, for all positive samples,

and $0 \leq -a_i \leq -d_- C$, for all negative samples.
Hybrid Fisher-SVM Classifier

Constraint-Relaxed Solution

\[ a_{opt} = K'^{-1} z \]

where \( z_i = d_i - \eta \), and

\[ \eta = \frac{e'^T d}{e'^T e - 1} \]

This is exactly the same as the optimal PFD solution. Recall that

\[
\begin{align*}
    a_{opt} &= E^{-1} v_{opt} \\
    &= E^{-1} [\Lambda + \rho I]^{-1} E [d - \eta e] \\
    &= U^T \Lambda^{-\frac{1}{2}} [\Lambda + \rho I]^{-1} \Lambda^{\frac{1}{2}} U [d - \eta e] \\
    &= K^T [U^T \Lambda U + U^T \rho I U]^{-1} [d - \eta e] \\
    &= [K + \rho I]^{-1} [d - \eta e].
\end{align*}
\]
Class-Balanced Wolfe Optimization with Perturbation-Adjusted Kernel Matrix

The approach is based on class-balanced Wolfe optimization with perturbation-adjusted kernel matrix.

1. Just like KFD, the class-balanced constraints will be adopted.

2. Under the perturbation analysis, the kernel matrix should be replaced by its perturbation-adjusted more realistic representation:

In summary, by imposing a stochastic model on the training data, we derive a hybrid Fisher-SVM classifier which contains Fisher solution and SVM solution as special cases. On one hand, it gives the same solution as the optimal Fisher when the constraints are lifted. On the other hand, it approaches the (class-balanced) SVM when the perturbation is small.
Kernel Shift

\[ K \rightarrow K + s \ I \]

Does such a shift improve the predictability of the supervised classifiers?
the SVM classifier has a very poor rate of 46% in accuracy since it fails the critical Mercer condition. Applying nonlinear kernel function to the similarity matrix can improve the performance to 77%. However, there remains a fraction of negative eigenvalues (0.3% or 6 out of 2200). It was also reported in [?] that the performance can be as high as 99% when the Mercer condition is met. It is worth mentioning that the kernel function associated with the best accuracy (i.e. to 99%) happens to be the one corresponding to inner-product kernels w.r.t. the empirical vectors. Therefore, the Mercer condition is assured.

four classes of 2200 yeast genes (cytoplasm, mitochondria, extracellular, and nuclear). The original similarity matrix has a large percentage of negative eigenvalues (189 out of 2200, or 8.5%). Therefore, it is of no surprise that
Subcellular Dataset: hy50

<table>
<thead>
<tr>
<th>Kernel</th>
<th>PNE</th>
<th>min_evalue</th>
<th>Accuracy (w/o shift) --&gt; w/ offset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>8.37%</td>
<td>-27.79</td>
<td>41.71% --&gt; 52.66%</td>
</tr>
<tr>
<td>K2</td>
<td>20.10%</td>
<td>-23.27</td>
<td>49.47% --&gt; 54.87%</td>
</tr>
<tr>
<td>K3</td>
<td>0%</td>
<td>4.0922e+004</td>
<td>54.73%</td>
</tr>
<tr>
<td>K4</td>
<td>10.16%</td>
<td>-5.9002e+005</td>
<td>54.82% --&gt; 51.96%</td>
</tr>
<tr>
<td>K5</td>
<td>0%</td>
<td>1.6442e-010</td>
<td>75.08%</td>
</tr>
</tbody>
</table>

PNE: Percentage of negative eigenvalues in the kernel before applying diagonal shift.
Topics:

1. Overview
2. Kernel Based Representation Spaces
3. Principal Component analysis (Kernel PCA)
4. Supervised Classifier
5. Clustering Methods for PD Kernels
   • 5.1 K-means and Convergence
   • 5.2 Kernel K-means
   • 5.3 Spectral Space: Dimension-Reduction
   • 5.4 Kernel NJ: Hierarchical Clustering
   • 5.5 Fast Kernel Trick
6. Extension to Nonvectorial Data and NPD Kernels
Gene expression

Unsupervised Clustering Analysis

Three major types of clustering approaches:

• Unsupervised
  – K-mean (EM)
  – SOM
  – HC

Clustering
5.1 Conventional K-means and Convergence

References: [9][17][18]
Clustering Criteria:

- Intra-Cluster Compactness
- Inter-Cluster Divergence
Exclusive Membership in K-means

\[ \mu_1, \mu_2, \mu_3 \]

exclusive membership
Likelihood-Based Criteria

The best parameters can be trained by maximizing the log-likelihood function:

\[
\log p(\mathcal{X} | \Theta)
\]

Let the K classes be denoted by \( C_k \) for and \( k = 1, \ldots, K \).

and parameterized by

\[ \Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_K\} \]
Under the i.i.d. (independent identical distribution) assumption,

\[ p(\mathcal{X} | \Theta) = \prod_{t} p(x_t | \Theta) \]

exclusive membership

\[ k = \arg\max_j p(x_t | \Theta_j) \]

\[ p(x_t | \Theta) = p(x_t | \Theta_k) \]

\[ p(\mathcal{X} | \Theta) = \prod_{k=1}^{K} \prod_{x_t \in C_k} p(x_t | \Theta_k) \quad \text{**} \]

\[ p(x_t | \Theta_k) = \frac{1}{\sqrt{(2\pi \sigma)^M}} \exp \left\{ -\frac{\|x_t - \mu_k\|^2}{2\sigma^2} \right\} \quad \text{**} \]

Substituting Eq. ** into Eq.*, we obtain

\[ - \log p(\mathcal{X} | \Theta) = - \sum_{k=1}^{K} \sum_{x_t \in C_k} \log p(x_t | \Theta_k) = \sum_{k=1}^{K} \sum_{x_t \in C_k} \|x_t - \mu_k\|^2 + \text{Constant} \]
K-means

The objective function to minimize is

$$E(\mathcal{X}) = \sum_{k=1}^{K} \sum_{x_t \in C_k} \|x_t - \mu_k\|^2$$
K-means Flowchart

Initiation

Winner Identification

Centroid Update

\[ k = \arg \max_j p(x_t | \Theta_j) \]

\[ \mu_k = \frac{1}{N_k} \sum_{x_t \in C_k} x_t \]

Convergence

Yes

Stop

No
Monotonic Convergence: The convergence is assured if (and only if) the pairwise matrix is positive definite. Thus, the convergence is assured for all the vector clustering problems, since all the commonly adopted kernel functions are positive definite.
Monotonic Convergence: The convergence is assured if (and only if) the pairwise matrix is positive definite. Thus, the convergence is assured for all the vector clustering problems, since most commonly adopted kernel functions are positive definite.
Sufficiency

M-Step: update the new centroids (two arrows)

..so as to reduce $E(x)$:

$$E(X) = \sum_{x_t \in X} \left( x_t - \mu_{r(x_t)} \right)^2$$
Sufficiency

E-Step: winner identification i.e. the new membership for some data. (3rd and 5th data will have a new owner (winner).

Does the new ownership reduce E(x)?

\[ E(X) = \sum_{x_t \in X} (x_t - \mu_{r(x_t)})^2 \]
Convergence: Sufficiency

E-Step: new membership (color coded) are assigned

\[
\sum_{x_t \in X} (x_t - \mu_{r(x_t)})^2 \rightarrow \sum_{x_t \in X} (x_t - \mu'_{r'(x_t)})^2
\]

Reduction of Energy
Sufficiency

The new membership help reduces $E(x)$!

\[
\sum_{x_t \in X} \left( x_t - \mu_{r'(x_t)} \right)^2
= \sum_{x_t \in X} \left( x_t - \mu_r(x_t) \right)^2 - \left( x_t - \mu_B \right)^2 + \left( x_t - \mu_G \right)^2
\leq \sum_{x_t \in X} \left( x_t - \mu_r(x_t) \right)^2
\]
Likewise, a function is called \textit{monotonically decreasing} (non-increasing) if, whenever $x \leq y$, then $f(x) \geq f(y)$, so it reverses the order.

\textit{M-Step: Maximization} $\rightarrow$ \textit{Minimization}

\[ ||x - \mu'||^2 + ||y - \mu'||^2 \leq ||x - \mu||^2 + ||y - \mu||^2 \]

\textbf{Reduction of Energy}

\[ \sum_{x_t \in X} (x_t - \mu'_{r'(x_t)})^2 \rightarrow \sum_{x_t \in X} (x_t - \mu'_{r''(x_t)})^2 \]

\[ \sum_{x_t \in X} (x_t - \mu'_{r'(x_t)})^2 \leq \sum_{x_t \in X} (x_t - \mu_{r'(x_t)})^2 \]
Necessity: Proof by counter-example.

If the kernel matrix is NPD, then ..

.. not necessary Reduction of Energy after centroid update!

\[ \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]
5.2 Kernel K-Means
Kernel K-Means

Conventional K-means, the objective function is

\[
\sum_{k=1}^{K} \sum_{x_t \in C_k} \| x_t - \mu^{(k)} \|^2
\]

K-means has very limited capability to separate clusters that are non-linearly separable in input space.

The objective of the kernel K-means aims to minimize:

\[
\sum_{k=1}^{K} \sum_{x_t \in C_k} \| \phi(x_t) - \mu^{(k)}_F \|^2 = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| e(x_t) - \mu^{(k)}_E \|^2
\]

Criteria A: centroid-based objective function
For PD Kernel, centroids are defined.

Algorithm: Kernel $K$-means

(i) Winner Identification: Given any training vector $x$, compute its distances to the $K$ centroids. Identify the winning cluster as the one with the shortest distance

$$\|c - \Theta(x)\|^2$$

(ii) Membership Update: If the vector $x$ is added to a cluster $C_k$, then update the centroid of $C_k$ (whether explicitly or not).
Spectral K-Means

• Kernel PCA provides a suitable Hilbert space which can adequately manifest the similarity of the objects, vectorial or non-vectorial.

Kernel trick: Implicit formula

• circumvent the explicit derivation of the centroids
• obviates the need of performing eigenvalue decomposition.
Three (Equivalent) Objective Functions

Assuming that the pairwise (similarity) matrix $K$ is PD, then the following clustering criteria are equivalent.

**Equivalence under Mercer (PD) Condition:**

$$A = B = C$$
Kernel K-means

**centroid-based objective function**

\[
A = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \phi(x_t) - \mu_{F}^{(k)} \|^2 = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \epsilon(x_t) - \mu_{\epsilon}^{(k)} \|^2
\]

**centroid-free but norm-based objective function** [Duda73]

\[
B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \phi(x_i) - \phi(x_j) \|^2
\]

\[
= \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \epsilon(x_i) - \epsilon(x_j) \|^2
\]

**centroid-free and norm-free objective function**

\[
C = \sum_{i=1}^{N} K_{ii} - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij} \right\}
\]
Kernel K-means

**Centroid-Based Objective Function**

\[
A = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \phi(x_t) - \mu^{(k)}_F \|^2 = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \overline{e}(x_t) - \mu^{(k)}_E \|^2
\]

**Centroid-Free but Norm-Based Objective Function [Duda73]**

\[
B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \phi(x_i) - \phi(x_j) \|^2
\]

\[
= \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \overline{e}(x_i) - \overline{e}(x_j) \|^2
\]

**Centroid-Free and Norm-Free Objective Function**

\[
C = \sum_{i=1}^{N} K_{ii} - \sum_{k=1}^{K} \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij}
\]
Compare A and B

\[ A = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \overrightarrow{e}(x_t) - \mu_{C_k}^{(k)} \|^2 \]

\[ B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \overrightarrow{e}(x_i) - \overrightarrow{e}(x_j) \|^2 \]
Compare B and C

\[ B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \overrightarrow{e}(x_i) - \overrightarrow{e}(x_j) \|^2 \]

\[ C = \sum_{i=1}^{N} K_{ii} - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij} \right\} \]

\textit{C is centroid-free and norm-free!}
5.3 Spectral K-Means

• Kernel Spectral Analysis is an effective approach.
Kernel K-means and Spectral clustering

In [7],
“…kernel k-means and spectral clustering….are related. Specifically, we can rewrite the weighted kernel k-means objective function as a trace maximization problem whose relaxation can be solved with eigenvectors. The result shows how a particular kernel and weight scheme is connected to the spectral algorithm of Ng, Jordan, and Weiss [27].

….. by choosing the weights in particular ways, the weighted kernel k-means objective function is identical to the normalized cut.”
Spectral Approximation

In the full-dimension, the full column of $E$ is used:

\[ E = \Lambda^{\frac{1}{2}}U \rightarrow \overrightarrow{e}(x_i) \rightarrow K = E^TE = U^T\Lambda U \]

In the reduced-dimension

\[ \Lambda' = \text{Diag}\{\lambda_1, \ldots, \lambda_m, 0, \ldots, 0\} \]

\[ E' = \Lambda'^{\frac{1}{2}}U. \rightarrow \overrightarrow{e}'(x_i) \]

The Spectral “Approximate” Kernel matrix:

\[ K' = E'^TE' = U^T\Lambda'^{\frac{1}{2}}\Lambda'^{\frac{1}{2}}U = U^T\Lambda'U. \]
Spectral Error Analysis

The Spectral “Error” Kernel matrix is defined as

\[ \tilde{K} = K - K' = U^T \tilde{\Lambda} U \]

where

\[ \tilde{\Lambda} = \Lambda - \Lambda' \]

centroid-free and norm-free objective function

\[ C = \sum_{i=1}^{N} K_{ii} - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij} \right\} \]
Spectral Error Analysis

\[
C = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \overline{e}(x_i) - \overline{e}(x_j) \|^2 = \sum_{i=1}^{N} K_{ii} - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij} \right\}
\]

\[
C' = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \overline{e}'(x_i) - \overline{e}'(x_j) \|^2 = \sum_{i=1}^{N} K_{ii}' - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij}' \right\}
\]

error = \[C - C' = \sum_{i=1}^{N} K_{ii}' - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij}' \right\}
\]

Since the first summation is independent of the clustering,

effective-deviation = \[\| C - C' - \sum_{i=1}^{N} K_{ii}' \| = \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij} \right\} \leq K \lambda_{m+1}
\]
Perturbation Analysis

\[ \sum_{x_i \in C_k} \sum_{x_j \in C_k} \tilde{K}_{ij} \leq \lambda_{m+1} \]

\[ \left[ \frac{1}{\sqrt{N_k}} \quad \frac{1}{\sqrt{N_k}} \quad \ldots \quad \frac{1}{\sqrt{N_k}} \right] \left[ \begin{array}{c} \frac{1}{\sqrt{N_k}} \\ \frac{1}{\sqrt{N_k}} \\ \vdots \\ \frac{1}{\sqrt{N_k}} \end{array} \right] \leq \lambda_{m+1} \]

effective-deviation = \[ \| C - C' - \sum_{i=1}^{N} \tilde{K}_{ii} \| = \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \tilde{K}_{ij} \right\} \leq K \lambda_{m+1} \]
If the kernel matrix is NPD, it is necessary to convert it to PD. One way to achieve this objective is by adopting a PD kernel matrix in the empirical space:

\[ K^2 = U^T \Lambda^2 U = [U^T \Lambda][\Lambda U] \]

In the full-dimension, the full-size columns of \( \Lambda U \) are used as the training vectors.

The dimension-reduced vectors are formed from the columns of \( \Lambda' U \).
Perturbation Analysis

By using the same perturbation analysis and, in addition, noting that the perturbation is upper bounded by the largest eigenvalue terms of magnitude) of \( \tilde{K} \)

\[
effective-deviation \leq \max\{K \lambda_{m+1}^2, K \lambda_N^2\}
\]

Here \( K \) denote the number of clusters (instead of the kernel matrix).

\[
\begin{bmatrix}
\frac{1}{\sqrt{N_k}} & \frac{1}{\sqrt{N_k}} & \cdots & \frac{1}{\sqrt{N_k}} \\
\end{bmatrix}
\begin{bmatrix}
1 \\
\sqrt{N_k} \\
\sqrt{N_k} \\
\vdots \\
\sqrt{N_k} \\
\end{bmatrix}
\leq \max\{K \lambda_{m+1}^2, -K \lambda_N^2\}
\]
Comparison with Spectral Clustering

The popular spectral clustering for graphic partitioning [27, 34] resorts to a uniform (as opposed to weighted) truncation strategy.

\[ E'' = \text{Diag}\{1, \ldots, 1, 0 \ldots 0\} \ U \]

where the number of 1's are usually set to be equal to K or greater.

In the Spectral Clustering, the training patterns are now represented by the columns of \( E'' \). Unfortunately, the KCA-type error analysis is difficult to derive for the spectral clustering.
### Comparison Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal Weightings</td>
<td>1 or 0</td>
<td>$\lambda_i$ or 0</td>
<td>$\lambda_i^2$ or 0</td>
</tr>
<tr>
<td>Perturbation Analysis</td>
<td>N/A</td>
<td>$K\lambda_{m+1}$</td>
<td>$\max{K\lambda_{m+1}^2, K\lambda_N^2}$</td>
</tr>
</tbody>
</table>

**Table 1.1.** Comparisons of Spectral Clustering, Spectral K-Means, and Empirical K-Means in terms of their (1) diagonal weightings and (2) perturbation analysis.
The perturbation analysis has several practical implications

• *As long as $K\lambda_{m+1}$ is within the tolerance level, then $m$-dimensional spectral $K$-means will be sufficient to yield a satisfactory clustering result.*

• *If the global optimum holds a margin over any local optimum by an amount greater than $2K\lambda_{m+1}$ (considering the worst possible situation), then the full-dimensional spectral $K$-means and $m$-dimensional spectral $K$-means will have the same optimal solution.*

• *When a NPD kernel matrix is encountered, it is common practice to discard the component corresponding to the negative spectral values. In this case, the same error analysis can be applied again to assess the level of deviation.*
Kernel Trick needs No Space at All.

Many applications may be solved without any specific space.

Kernel Trick for Unsupervised Learning

Kernel K-Means (SOM/NJ) w/o explicitly computing centroids
Kernel Trick is effective for:

Kernel NJ Algorithm

Neighbor-Joining (NJ) Algorithm

Kernel K-Means for Highly Sparse Data
5.4 Hierarchical NJ Algorithm
Hierarchical clustering analysis plays an important role in biological taxonomic studies, as a phylogenetic tree or evolutionary tree is often adopted to show the evolutionary lineages among various biological entities.
Phylogenetic tree of SARS viral genes

The phylogenetic tree can help reveal the epidemic links.

The severe acute respiratory syndrome (SARS) virus started its outbreak during the winter of 2002. All of the SARS viruses can be linked via a phylogenetic tree, obtained from genomic data by measuring the DNA similarities between various SARS viruses.
Five vectorial or nonvectorial objects – (A, B, C, D, E) – have the following (positive definite) kernel matrix:

\[
K = \begin{bmatrix}
S_{AA} & S_{AB} & S_{AC} & S_{AD} & S_{AE} \\
S_{BA} & S_{BB} & S_{BC} & S_{BD} & S_{BE} \\
S_{CA} & S_{CB} & S_{CC} & S_{CD} & S_{CE} \\
S_{DA} & S_{DB} & S_{DC} & S_{DD} & S_{DE} \\
S_{EA} & S_{EB} & S_{EC} & S_{ED} & S_{EE}
\end{bmatrix} = \begin{bmatrix}
1.0 & 0.6 & 0.2 & 0.2 & 0.4 \\
0.6 & 1.0 & 0.3 & 0.3 & 0.4 \\
0.2 & 0.3 & 1.0 & 0.5 & 0.6 \\
0.2 & 0.3 & 0.5 & 1.0 & 0.8 \\
0.4 & 0.4 & 0.6 & 0.8 & 1.0
\end{bmatrix}.
\]
Neighbor-Joining (NJ) Algorithm

It starts with \( N \) nodes, i.e. one object per node:

\[
\{ x_1, x_2, \ldots, x_N \}
\]

Step 1: Compute \( N(N-1)/2 \) similarity/distance measures for the \( N \) objects.

These values correspond to the off-diagonal entries of a symmetric \( N \times N \) similarity matrix.

Step 2: Merger of two most similar nodes: Update the merger node, remove the two old nodes.

Step 3: After \( N-1 \) mergers, all the objects are merged into one common ancestor, i.e. the root of all objects.
The conventional NJ Score Update is approximation-based.

Distance is defined for a Hilbert Space endowed with an inner-product.

Exact method based centroids is natural.
Kernel Trick

The winner can be identified to be the cluster, say $C_k$, that yields the shortest distance.

$$\| \vec{\phi}(x) - c \| \leq \| \vec{\phi}(x) - \frac{\sum_{i \in C_k} \vec{\phi}(x_i)}{\rho} \|^2$$

The computation can be carried out exclusively based on $K_{ij}$. 
Kernel Trick

\[
\| \vec{\phi}(x) - c \|^2 = \| \vec{\phi}(x) - \frac{\sum_{i \in C_k} \vec{\phi}(x_i)}{p} \|^2 \\
= \begin{bmatrix}
-\frac{e_p^T}{p} & 1
\end{bmatrix}
[S_{p+1}]
\begin{bmatrix}
-\frac{e_p}{p} \\
1
\end{bmatrix}
\]

\[
[S_{p+1}] = \begin{bmatrix}
S_p & \vec{v} \\
\vec{v}^T & \| \vec{\phi}(x) \|^2
\end{bmatrix}
\]

\[
S_p \text{ is the } p \times p \text{ pairwise matrix}
\]

\[
\vec{\phi}(x_i) \cdot \vec{\phi}(x_j), x_i \in C_k \text{ and } x_j \in C_k
\]
Kernel Trick for NJ Algorithm

(i) Winner Identification: Given any training vector $x$, compute its distances to the $K$ centroids. Identify the winning cluster as the one with the shortest distance

$$
\left\| c - \phi(x) \right\|^2
$$

(ii) Membership Update: If a node is added to a cluster $C_k$, then merge that node into the cluster $C_k$ to form $C'_k$. 


Kernel-NJ Algorithm

Because $S_{DE} = 0.8$ has the highest pairwise similarity, (DE) will first merge into a new node: F.

\[ s_{AF} = \frac{\vec{e}(A)^T \vec{e}(F) + \vec{e}(A)^T \vec{e}(E)}{2} = \frac{0.2 + 0.4}{2} = 0.3. \]

\[ s_{BF} = \frac{\vec{e}(B)^T \vec{e}(F) + \vec{e}(B)^T \vec{e}(E)}{2} = \frac{0.3 + 0.4}{2} = 0.35. \]

\[ s_{CF} = \frac{\vec{e}(C)^T \vec{e}(F) + \vec{e}(C)^T \vec{e}(E)}{2} = \frac{0.1 + 0.6}{2} = 0.35. \]

\[ s_{FF} = \frac{\vec{e}(F)^T \vec{e}(F)}{2} = \frac{0}{0} = 0.9. \]

This leads to a reduced (4 x 4) similarity matrix:

\[
\begin{bmatrix}
S_{AA} & S_{AB} & S_{AC} & S_{AF} \\
S_{BA} & S_{BB} & S_{BC} & S_{BF} \\
S_{CA} & S_{CB} & S_{CC} & S_{CF} \\
S_{FA} & S_{FB} & S_{FC} & S_{FF}
\end{bmatrix}
= \begin{bmatrix}
1.0 & 0.6 & 0.2 & 0.3 \\
0.6 & 1.0 & 0.3 & 0.35 \\
0.2 & 0.3 & 1.0 & 0.35 \\
0.3 & 0.35 & 0.35 & 0.9
\end{bmatrix}
\]

\[ d_{AF} = S_{AA} + S_{FF} - 2S_{AF} \]

\[ d_{AF} = 1 + 0.9 - 2 \times 0.3 = 0.6 \]

\[ d_{CF} = 1 + 0.9 - 2 \times 0.35 = 1.2 \]
Kernel-NJ Algorithm

The overall NJ algorithm runs as follows:

\[
\{A, B, C, D, E\} \Rightarrow \{A, B, C, DE\} \Rightarrow \{AB, C, DE\} \\
\Rightarrow \{AB, CDE\} \Rightarrow \{ABCDE\}.
\]
5.5 Fast Kernel Trick
For Kernel K-Means
Winner Identification

\[
\arg \min_k \| \mu_k \|^2 - 2R_{rk}
\]

\[
\| \phi(x_r) - \mu_k \|^2 = \| \phi(x_r) - \frac{\sum_{i \in c_k} \phi(x_i)}{p_k} \|^2
\]

\[
= \| \phi(x_r) \|^2 + \| \mu_k \|^2 - \frac{2\sum_{i \in c_k} \phi(x_i)^T \phi(x)}{p_k}
\]

\[
= \| \phi(x_r) \|^2 + \| \mu_k \|^2 - \frac{2\sum_{i \in c_k} K(x_i, x)}{p_k}
\]

\[
= \| \phi(x_r) \|^2 + \| \mu_k \|^2 - 2R_{rk}
\]

\[
R_{rk} = \frac{\sum_{i \in c_k} K(x_r, x_i)}{p_k}
\]
Similarity Table for Fast Kernel Trick

$R_{rk}$:

N Objects \quad K Clusters
In the actual coding of fast kernel trick, we must take into account the fact whenever a cluster is winning a new member there is always another cluster losing an old member. As a result, two clusters (i.e. two columns of the similarity measures) will have to be updated. For illustration simplicity, we focus only on the winning cluster.
Update of Similarity Table:

Slow Update

$$R'_{jk} = K(x_j, \mu'_k) = \frac{1}{p_k + 1} \sum_{x_i \in C'_k} K(x_j, x_i)$$

Fast Update

$$R'_{jk} = \frac{p_k}{p_k + 1} R_{jk} + \frac{1}{p_k + 1} K_{jr}$$

Update of Centroids' Auto-Correlations:

$$\|\mu'_k\|^2 = \left\| \frac{\sum_{i \in C'_k} \phi(x_i)}{p_k + 1} \right\|^2$$

$$= \frac{p_k^2}{(p_k + 1)^2} \|\mu_k\|^2 + \frac{2}{(p_k + 1)^2} \sum_{i \in C_k} K(x_r, x_i) + \frac{1}{(p_k + 1)^2} K(x_r, x_r)$$

$$= \frac{p_k^2}{(p_k + 1)^2} \|\mu_k\|^2 + \frac{2}{(p_k + 1)^2} \sum_{i \in C_k} K_{ri} + \frac{1}{(p_k + 1)^2} K_{rr}$$  (1.65)
Fast Kernel Trick

• Winner Identification: negligible

• Winner Update:
  • Winner Column Update
  • Winner's Auto-Correlation

• Loser Update:
  • Loser Column Update
  • Loser's Auto-Correlation

$N_u O(N)$
Fast Kernel Trick

Winner Column Update:

\[ R'_{jk} = \frac{p_k}{p_k + 1} R_{jk} + \frac{1}{p_k + 1} K_{jr} \]

Winner's Auto-Correlation:

\[ \|\mu'_k\|^2 = \frac{p_k^2}{(p_k + 1)^2} \|\mu_k\|^2 + \frac{2}{(p_k + 1)^2} \sum_{i \in C_k} K_{ri} + \frac{1}{(p_k + 1)^2} K_{rr} \]

Loser Column Update:

\[ R'_{jk} = \frac{p_q}{p_q - 1} R_{jq} - \frac{1}{p_q - 1} K_{jr} \]

Loser's Auto-Correlation:

\[ \|\mu'_q\|^2 = \frac{p_q^2}{(p_q - 1)^2} \|\mu_q\|^2 - \frac{2}{(p_q)^2} \sum_{i \in C'_q} K_{ri} + \frac{1}{(p_q)^2} K_{rr} \]
Comparison of Computational Cost

When to Use What?

Note that, for K-means via kernel component, there incurs a (one-time) cost of $O(N^3) \sim O(N^4)$ for factorization.

$$O(N^3) + N_r N_e O(KMN)$$

In contrast, the kernel trick obviates the need of performing spectral decomposition.

$$N_r N_u O(N)$$
When to use (Fast) Kernel Trick?

For Sparse Kernel Matrix, (Fast) Kernel Trick is relatively more attractive.

As stated in [7],

"Thus far, only eigenvector based algorithms have been employed to minimize normalized cuts in spectral clustering and image segmentation. However, software to compute eigenvectors of large sparse matrices (often based on the Lanczos algorithm) can have substantial computational overheads, especially when a large number of eigenvectors are to be computed. In such situations, our equivalence has an important implication: we can use k-means-like iterative algorithms for directly minimizing the normalized-cut of a graph."
When not to use Fast Kernel Trick?

Fast Kernel Trick has less advantage:

1. When the original vector dimension is small or when the dimension can be substantially reduced with good approximation. (This is because the processing time of Kernel Trick is independent of vector dimension.)

2. Another occasion is for Kernel NJ is itself a finite algorithm, so kernel trick will already be fast enough and there is no need of fast kernel trick or spectral dimension reduction.
Regular None-Sparse Kernel Matrix

Spectral K-Means is relatively more attractive.

Kernel Trick is less effective if the required number of K-means, $N_r$, is large.

Multiple Trial

**MATLAB:** `Kmeans(A,B,C,..)`

\[ A = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \phi(x_t) - \mu_{\mathcal{H}}^{(k)} \|^2 \]
Topics:

1. Overview
2. Kernel Based Representation Spaces
3. Principal Component analysis (Kernel PCA)
4. Supervised Classifier
5. Fast Clustering Methods for Positive-Definite Kernels

5. Extension to Nonvectorial Data and NPD (Non-Positive-definite) Kernels

6.1 The Shift Approach
   • a. Shift-Invariance
   • b. Concern on The Shift Approach
6. 2 The Truncation Approach
6. 3 Empirical Vectors Space
Extension of Vectorial data to nonvectorial data.

\[ c = \frac{A + B}{2} \]

Distance is defined for a Hilbert Space endowed with an inner-product.

A Vital Assumption: \[ K(x_i, x_j) = S^*(x_i, x_j) = \text{Positive Definite} \]
Application of K-means to Graph Partitions:

By adjusting kernels and weights, the clustering algorithm can be applied to a broad spectrum of Graph Partitions applications. Some examples are:

- Clustering of Gene expression profiles
- Network Analysis of Protein-protein-interaction
- Interconnection Networks
- Parallel processing algorithm: partitioning
- Document data mining: word-by-document matrix
- Market basket data: transaction-by-item matrix
For all the vector clustering problems:

The problem is well defined and its convergence is assured if positive definite (PD) kernel functions are adopted.

For nonvectorial data clustering problems:

The problem definition may requires some attention.
The convergence of K-means can be assured only after the similarity matrix is converted to a PD matrix.
The ESSENCE of having a PD Kernel
‘Forcing’ a PD Kernel:

Three Approaches

• Shift Approach
• Truncation Approach
• Squared Kernel: $K^2$
6.1 The Shift Approach

Kernel Shift

If $K$ is NPD, it must be converted to a PD Matrix.

\[ K + sI \]
6.1a Shift-Invariance

Does such a Shift Change the Nature of the Problem?

Kernel NJ is Not Shift-Invariant
Kernel K-means is Shift-Invariant
Kernel SOM can be Shift-Invariant
Kernel NJ is Not Shift- Invariant

For Kernel-NJ, the positive shift tends to favor identification of a larger cluster.

\[
\| \vec{\phi}(x) - c \|^2 = \| \vec{\phi}(x) - \frac{\sum_{i \in C_k} \vec{\phi}(x_i)}{p} \|^2
\]

\[
= \begin{bmatrix} \frac{-e_p^T}{p} & 1 \end{bmatrix} \begin{bmatrix} e_p \\ p \\ 1 \end{bmatrix}
\]

Apply Shift, then the squared-distance is adjusted by

\[ s + \frac{s}{(p+1)} \]

This penalizes smaller cluster and favors larger clusters.
Kernel-NJ

\[
\begin{bmatrix}
S_{AA} & S_{AB} & S_{AC} & S_{AF} \\
S_{BA} & S_{BB} & S_{BC} & S_{BF} \\
S_{CA} & S_{CB} & S_{CC} & S_{CF} \\
S_{FA} & S_{FB} & S_{FC} & S_{FF}
\end{bmatrix}
= \begin{bmatrix}
1.0 & 0.6 & 0.2 & 0.3 \\
0.6 & 1.0 & 0.3 & 0.35 \\
0.2 & 0.3 & 1.0 & 0.35 \\
0.3 & 0.35 & 0.35 & 0.9
\end{bmatrix}.
\]

\[
d_{AB} = (1) + (1) - 2 \times 0.6 = 0.8
\]

\[
\begin{align*}
S'_{AA} &= S'_{BB} = S'_{CC} = 1 + s \\
d'_{AB} &= (1 + s) + (1 + s) - 2 \times 0.6 = 0.8 + 2 \times s
\end{align*}
\]

\[
S'_{FF} = 0.9 + \frac{s}{2}
\]

\[
d'_{CF} = (1 + s) + (0.9 + \frac{s}{2}) - 2 \times 0.35 = 1.2 + \frac{3s}{2}
\]

\[
s = 2
\]

\[
d'_{AB} = 0.8 + 2 \times 2 = 4.8
\]

\[
d'_{CF} = 1.2 + \frac{3 \times s}{2} = 1.2 + \frac{3 \times 2}{2} = 4.2
\]
Kernel K-means is Shift-Invariant

Equivalence under Mercer (PD) Condition: \( A = B = C \)

**centroid-based objective function**

\[
A = \sum_{k=1}^{K} \sum_{x_t \in C_k} \| \overrightarrow{\phi}(x_t) - \mu^{(k)}_H \|^2
\]

**centroid-free objective function**

\[
B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \overrightarrow{\phi}(x_i) - \overrightarrow{\phi}(x_j) \|^2
\]

**centroid-free and norm-free objective function**

\[
C = \sum_{i=1}^{N} K_{ii} - \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} K_{ij} \right\}
\]
Example: centroid-free and norm-free objective function

Theorem 2 (Equivalence of Ratio Association and Norm-free Clustering Criterion)

Suppose that the affinity matrix $A_{ij}$ of a network is used as the similarity matrix, i.e. $S_{ii} = A_{ii}$, then the norm-free clustering criterion and ratio association are equivalent and consequently Kernel K-means (and other variants) can be applied for optimal graphic partitioning. † In other words, the optimal solution minimizing the norm-free clustering criterion is exactly the same as that maximizing the ratio association.

\[
\max \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} S_{ij} \right\}
\]

\[
\leftrightarrow \max \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} A_{ij} \right\}
\]
Kernel K-means admits a centroid-free criterion

*Equivalence under Mercer (PD) Condition:* \[ A = B = C \]

centroid-based objective function

\[
A = \sum_{k=1}^{K} \sum_{x_t \in C_k} \left\| \overrightarrow{\phi}(x_t) - \mu_\mathcal{H}^{(k)} \right\|^2
\]

centroid-free objective function

\[
B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \left\| \overrightarrow{\phi}(x_i) - \overrightarrow{\phi}(x_j) \right\|^2
\]

\[\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\]

NPD \quad \leftrightarrow \quad \text{Pseudo-Norm}
Proof of Shift-Invariance

\[
B = \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \| \vec{\phi}(x_i) - \vec{\phi}(x_j) \|^2
\]
Kernel K-means is Shift-Invariant

Theorem 4 (Shift Invariance of Norm-Free Objective Function)

Let \( S' = S + s \mathbf{I} \), then the optimal solution for norm-free objective function based on \( S' \) will be exactly the same as that based on \( S \).

\[
\sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} S'_{ij} \right\} = \sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} S_{ij} \right\} + (N-K)s
\]

One way to assure convergence is to force PD by adding a positive shift to the diagonal of the similarity matrix, i.e. \( S' = S + s \mathbf{I} \) where \( s \geq -\lambda_{\text{min}} \).

Conversion of K into a PD Matrix.
If $K$ is NPD, it must be converted to a PD Matrix.

$$K + s \mathbf{I}$$
6.1b Concern on The Shift

Is the formulation rational?

Test Case:

The Tale of Twins.
Necessity:
Proof by counter-example.

\[
\begin{align*}
\|\mathbf{x}_2 - \mu\|_\Lambda^2 + \|\mathbf{x}_3 - \mu\|_\Lambda^2 + \|\mathbf{x}_4 - \mu\|_\Lambda^2 &\approx -\frac{3}{4}, \\
\|\mathbf{x}_2 - \mu'\|_\Lambda^2 + \|\mathbf{x}_3 - \mu'\|_\Lambda^2 + \|\mathbf{x}_4 - \mu'\|_\Lambda^2 &\approx -\frac{2}{3},
\end{align*}
\]

\[
\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}
\]
Example for Pseudo-Norm in Type-B

\[ A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

Note: Empirical vectors
The concern lies in the twins are separated far from each other.

\[
K' = K_{NPD} - \lambda_4 K_{NPD} - (-1 - \epsilon^2)I = \begin{bmatrix}
2 & 0 & 0 & 0 \\
0 & \epsilon^2 & 0 & -\epsilon \\
0 & 0 & 1 + \epsilon^2 & 0 \\
0 & -\epsilon & 0 & 1 \\
\end{bmatrix}
\]

\[
K' = \begin{bmatrix}
\sqrt{2} & 0 & 0 & 0 \\
0 & \epsilon & 0 & 0 \\
0 & 0 & \sqrt{1 + \epsilon^2} & 0 \\
0 & -1 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
\sqrt{2} & 0 & 0 & 0 \\
0 & \epsilon & 0 & -1 \\
0 & 0 & \sqrt{1 + \epsilon^2} & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
6.2 The Truncation Approach

A popular and simple means to make an NPD matrix PD is by truncation of negative spectral components. [1,30]. Same perturbation analysis applies.

\[
\Lambda' = \text{Diag}\{\lambda_1, \cdots, \lambda_m, 0, \cdots, 0\}
\]

\[
\tilde{\Lambda} = \Lambda - \Lambda'
\]

\[
\lambda_m \geq 0 > \lambda_{m+1}
\]

effective-deviation = \[\sum_{k=1}^{K} \left\{ \frac{1}{N_k} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \tilde{K}_{ij} \right\} \leq -K\lambda_N\]
6.3 Empirical Vectors Space

The $K^2$ Approach [36]

- Supervised Learning
- Unsupervised Learning: Empiric K-Means
Empirical K–Means: Empirical Vectors

Example for Pseudo-Norm in Type-B

\[ \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

\[ K_{NPD} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \epsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -\epsilon \\ 0 & 0 & 0 & 0 \\ 0 & -\epsilon & 0 & -\epsilon^2 \end{bmatrix} \]

The proximity of the twins are preserved in the empirical space.
\[ \sum_{i=1}^{N} a_i K(x_i, x) + b \]

\[ \sum_{i=1}^{N} a_i K'(x_i, x) + b \]

vectorization

x = sequence
Empirical Eigen-Space

- It is a Hilbert space endowed with the conventional (linear) dot-product.
- It has a finite dimension, N, with non-orthogonal bases. Its energy is more concentrated to the higher components.
- It is dependent of the training dataset.

Empirical Eigen-Space vs. spectral space

Empirical Eigen-Space has
1. Positive eigenvalues for all kind of data (vectorial/nonvectorial).
2. It has a stronger denoising capability than the spectral space. (Its energy is more concentrated to the higher components).
If the kernel matrix is NPD, it is necessary to convert it to PD. One way to achieve this objective is by adopting a PD kernel matrix in the empirical space:

\[ K^2 = U^T \Lambda'^2 U = [U^T \Lambda][\Lambda U] \]

In the full-dimension, the full-size columns of \( \Lambda U \) are used as the training vectors.

The dimension-reduced vectors are formed from the columns of \( \Lambda' U \).
Perturbation Analysis

By using the same perturbation analysis and, in addition, noting that the perturbation is upper bounded by the largest eigenvalue terms of magnitude) of $\tilde{K}$

\[
\text{effective-deviation} \leq \max \{ K\lambda_{m+1}^2, K\lambda_N^2 \}
\]

Here $K$ denote the number of clusters (instead of the kernel matrix).

\[
\begin{bmatrix}
\frac{1}{\sqrt{N_k}} & \frac{1}{\sqrt{N_k}} & \ldots & \frac{1}{\sqrt{N_k}} \\
\end{bmatrix}
\begin{bmatrix}
\tilde{K}
\end{bmatrix}
\begin{bmatrix}
\frac{1}{\sqrt{N_k}} \\
\frac{1}{\sqrt{N_k}} \\
\vdots \\
\frac{1}{\sqrt{N_k}} \\
\end{bmatrix}
\leq \max \{ \lambda_{m+1}, -\lambda_{N} \}
\]
Future Research

Kernel Approach to Integration of heterogeneous data. [33].
References


References


References


For background reading on Linear Algebra:
