Flatten a Curved Space by Kernel: From Einstein to Euclid

Qiuyuan Huang, Dapeng Oliver Wu

Einstein's general theory of relativity fundamentally changed our view about the physical world. Different from Newton's theory, Einstein's space and time are not flat but can be warped by matter. For a curved space such as Einstein's space, Euclidean geometry is no longer suitable, and Riemannian geometry is usually used instead. In parallel with physics, due to an explosion of data from all fields of science, there is an increasing need for pattern analysis tools, which are capable of analyzing patterns of data in a non-Euclidean (curved) space. To handle data in a curved space, linear approaches are not directly applicable, and instead nonlinear approaches are the right weapon. However, early-day nonlinear approaches were usually based on gradient descent or greedy heuristics, and suffered from local minima and overfitting [1]. In contrast, kernel methods provide a powerful means for transforming data in a non-Euclidean curved space (such as Einstein space) into points in a high-dimensional Euclidean flat space, so that linear approaches can be applied to the transformed points in the high-dimensional Euclidean space. With this flattening capability, kernel methods combine the best features of linear approaches and nonlinear approaches, i.e., kernel methods are capable of dealing with nonlinear structures while enjoying a low computational complexity like linear approaches. In this column, we provide important insights into kernel methods and illustrate the power of kernel methods in two important pattern analysis problems: feature extraction and clustering.

INSIGHT INTO KERNEL METHODS: A TRANSDUCTIVE PARADIGM

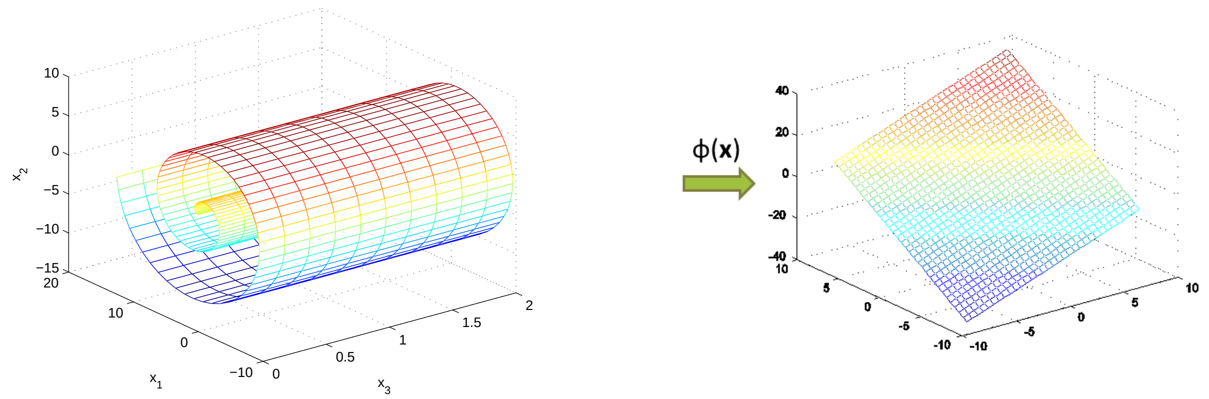
A linear pattern analysis method $\mathscr{A}$ can be extended to a kernel method via the following procedure:

1. Select a kernel suitable for a given nonlinear pattern analysis problem. A kernel is a function κ that for all $\mathbf{x}$ and $\mathbf{z}$ in the data space X, satisfies $\kappa(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$, where ϕ is a mapping from X to a Hilbert space, and $\langle \cdot, \cdot \rangle$ is an inner product.

2. Given a training data set $\{\mathbf{x}_i : i = 1, 2, \cdots, N\}$, calculate the kernel function $k_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for each pair of $\mathbf{x}_i$ and $\mathbf{x}_j$. The resulting $N \times N$ matrix $\mathbf{K}$ with entries $k_{i,j}$ is called the *kernel matrix*.

3. Train the given linear pattern analysis method $\mathscr{A}$ using the kernel matrix and the training data set, and obtain a pattern function $f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})$, where α_i ($i = 1, \cdots, N$) are obtained by training.

The term $\sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})$ is called the *dual representation* of $f(\mathbf{x})$ [1], and α_i ($i = 1, \cdots, N$) are called the *dual variables*. In essence, under dual representation, $f(\mathbf{x})$ is a linear combination of kernel functions evaluated at each training data point and a given $\mathbf{x}$. Hence, a kernel method actually conducts *transduction*, i.e., directly draws conclusions about new data from the training data, without constructing a model; in other words, transduction is a type of inference from observed, specific (training) cases



(a) Nonlinear surface in the input data space (b) Hyperplane in the feature space

Figure 1: Flatten a curved space by a nonlinear mapping $\phi(\mathbf{x})$.

to specific (test) cases (e.g., a given $\mathbf{x}$). This is different from *induction*, which is a type of inference from specific (training) cases to a general rule/model. Under an inductive paradigm, once the general rule/model is obtained through learning, the training data will be discarded and will not explicitly be part of the general rule/model.

Why is a kernel method capable of resolving nonlinear structures at a low computational cost? First, the capability of dealing with nonlinear structures is due to the use of $\phi(\mathbf{x})$, which flattens a curved space. Specifically, flattening is achieved by mapping $\mathbf{x} \in X$ to $\phi(\mathbf{x})$ in a high-dimensional feature space such that the nonlinear structure embedded in $\{\mathbf{x}_i\}$ becomes a linear structure in the feature space. For example, a nonlinear surface in X becomes a linear hyperplane in the feature space after applying map $\phi(\mathbf{x})$ (see Fig. 1). Second, low computational complexity is due to the dual representation of pattern function $f(\mathbf{x})$, also known as *kernel trick*, i.e., kernel $\kappa(\mathbf{x}_i,\mathbf{x}) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$ can be evaluated without computing $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x})$. This is because a kernel can be directly given as a function of $\mathbf{x}_i$ and $\mathbf{x}$ without explicitly defining $\phi(\cdot)$. Avoiding computing $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x})$ significantly reduces the computational complexity.

For feature extraction problems, $f(\mathbf{x})$ is a feature vector in a feature space. For clustering problems, $f(\mathbf{x})$ is a cluster index. For classification problems, $f(\mathbf{x})$ is a class index. For regression problems, $y = f(\mathbf{x})$ is a regression function. For nonlinear system identification problems, $\mathbf{x}(t+1) = f(\mathbf{x}(t))$ is a system state equation that governs the dynamics of a given nonlinear system. We will describe the first two pattern functions in the following sections.

KERNEL-BASED FEATURE EXTRACTION

Transforming the input data $\mathbf{x}$ into a feature vector $\phi(\mathbf{x})$ in a feature space is called feature extraction. The purpose of feature extraction is to extract relevant information from the input data. Dimensionality reduction (i.e., removing irrelevant feature dimensions) is usually involved in feature extraction. In this section, we describe two known techniques in the literature: *kernel principle component analysis* (KPCA) and a discriminant-learning-based kernel feature extraction method.

KPCA [1] utilizes a dual representation of an eigenvector $\mathbf{u}_j$ of the covariance matrix of $\mathbf{x}$, so that the projection P of $\phi(\mathbf{x})$ onto the direction $\mathbf{u}_j$ in the feature space is given by $P_{\mathbf{u}_j}(\phi(\mathbf{x})) =$

$\sum_{i=1}^{N} \alpha_{i,j} \kappa(\mathbf{x}_i, \mathbf{x})$. This dual representation enables us to avoid computing $\phi(\mathbf{x})$. KPCA is summarized below.

> Input: $\{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^L, i = 1, 2, \cdots, N\}$, kernel $\kappa(\cdot,\cdot)$, and k (the dimension of the output feature space).
>
> 1. Calculate $k_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, N$, and obtain the kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, whose i-th row and j-th column is $k_{i,j}$.
> 2. Find the k largest eigenvalues $\{\lambda_j : j = 1, \cdots, k\}$ and the corresponding eigenvectors $\{\mathbf{v}_j : \mathbf{v}_j \in \mathbb{R}^N, j = 1, \cdots, k\}$ of matrix $\mathbf{K}$.
> 3. Let $\alpha_{i,j} = v_{i,j} / \sqrt{\lambda_j}$ ($i = 1, \cdots, N$ and $j = 1, \cdots, k$), where $v_{i,j}$ is the i-th element of $\mathbf{v}_j$.
> 4. Compute $\tilde{x}_{j,m} = \sum_{i=1}^{N} \alpha_{i,j} \kappa(\mathbf{x}_i, \mathbf{x}_m)$ ($m = 1, \cdots, N$ and $j = 1, \cdots, k$), where $\tilde{x}_{j,m}$ is the j-th element of $\tilde{\mathbf{x}}_m$.
>
> Output: transformed data $\{\tilde{\mathbf{x}}_i : \tilde{\mathbf{x}}_i \in \mathbb{R}^k, i = 1, \cdots, N\}$.

kernel linear feature extraction (KLFE) [2] is a discriminant-learning-based kernel feature extraction method for supervised learning. Discriminant-learning-based feature extraction seeks a feature space that maximizes the difference between data of difference classes. Consider supervised learning for two classes and assume that the input data set $\mathscr{D}$ consists of $\{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, N\}$, where $\mathbf{x}_i \in \mathbb{R}^L$ and the class label $y_i \in \{-1, +1\}$. LFE [3, 4] seeks a linear transformation matrix $\mathbf{W}$ that maximizes the difference between transformed data points $\mathbf{Wx}$ of different classes. This difference is called a margin, similar to that in a support vector machine (SVM) [1]. The margin $\rho_i(\mathbf{W})$ of $\mathbf{x}_i$ under $\mathbf{W}$ is defined by $\rho_i(\mathbf{W}) = \mathbf{m}_i^T \mathbf{W} \mathbf{m}_i - \mathbf{h}_i^T \mathbf{W} \mathbf{h}_i$, where $\mathbf{m}_i^T \mathbf{W} \mathbf{m}_i$ is the Mahalanobis distance between $\mathbf{x}_i$ and its nearest neighbor in a different class, and $\mathbf{h}_i^T \mathbf{W} \mathbf{h}_i$ is the Mahalanobis distance between $\mathbf{x}_i$ and its nearest neighbor in the same class. Let $\mathbf{m}_i \triangleq \mathbf{x}_i - NM(\mathbf{x}_i, y_i)$, where the nearest miss function $NM(\cdot,\cdot)$ is given by

$$
\begin{aligned}
NM(\mathbf{x}, y) \triangleq \quad & \arg\min_{\mathbf{x}'} ||\mathbf{x}' - \mathbf{x}||_p, && (1)\\
& \text{s.t. } (\mathbf{x}', y') \in \mathscr{D}, && (2)\\
& y' \neq y, && (3)
\end{aligned}
$$

where $||\mathbf{x}||_p$ is l^p norm of $\mathbf{x}$. Let $\mathbf{h}_i \triangleq \mathbf{x}_i - NH(\mathbf{x}_i, y_i)$, where the nearest hit function $NH(\cdot,\cdot)$ is given by

$$
\begin{aligned}
NH(\mathbf{x}, y) \triangleq \quad & \arg\min_{\mathbf{x}'} ||\mathbf{x}' - \mathbf{x}||_p, && (4)\\
& \text{s.t. } (\mathbf{x}', y') \in \mathscr{D}, && (5)\\
& y' = y. && (6)
\end{aligned}
$$

The margin-maximizing $\mathbf{W}$ can be found by solving the following optimization problem:

$$
\begin{aligned}
\max_{\mathbf{W}} \quad & \textstyle\sum_{i=1}^{N} \rho_i(\mathbf{W}), && (7)\\
\text{s.t.} \quad & \|\mathbf{W}\|_F^2 = 1, \mathbf{W} \geq 0,
\end{aligned}
$$

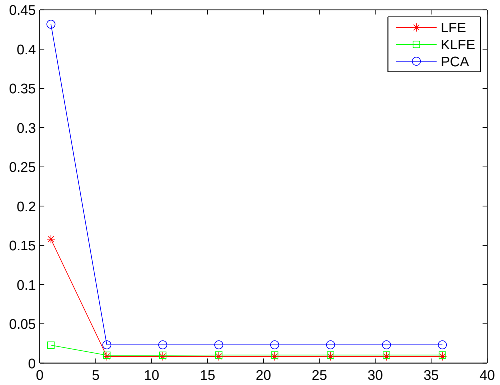


Figure 2: Classification error rate (y-axis) vs. dimension L (x-axis) of Swiss roll data

where $\|\mathbf{W}\|_F$ is the Frobenius norm of $\mathbf{W}$. $\mathbf{W} \geq 0$ means that matrix $\mathbf{W}$ has to be positive semi-definite. KLFE is a kernel extension of LFE. Using a nonlinear mapping $\phi(\mathbf{x})$ that maps $\mathbf{x} \in \mathbb{R}^L$ to $\phi(\mathbf{x}) \in \mathbb{R}^{\bar{L}}$ ($\bar{L} > L$), we can define $\bar{\mathbf{m}}_i \triangleq \phi(\mathbf{x}_i) - \phi(NM(\mathbf{x}_i, y_i))$ and $\bar{\mathbf{h}}_i \triangleq \phi(\mathbf{x}_i) - \phi(NH(\mathbf{x}_i, y_i))$. Under KLFE, the margin-maximizing $\bar{\mathbf{W}} \in \mathbb{R}^{\bar{L} \times \bar{L}}$ can be found by solving

$$\begin{aligned} \max_{\bar{\mathbf{W}}} \quad & \textstyle\sum_{i=1}^{N} (\bar{\mathbf{m}}_i^T \bar{\mathbf{W}} \bar{\mathbf{m}}_i - \bar{\mathbf{h}}_i^T \bar{\mathbf{W}} \bar{\mathbf{h}}_i), \\ \text{s.t.} \quad & \|\bar{\mathbf{W}}\|_F^2 = 1, \bar{\mathbf{W}} \geq 0. \end{aligned} \tag{8}$$

In [2], a KPCA-based method was proposed to efficiently compute nonlinearly transformed points $\tilde{\mathbf{x}}_i = \bar{\mathbf{W}} \phi(\mathbf{x}_i)$. KLFE can also achieve dimensionality reduction by choosing the dimensions with largest variance in the feature space.

To compare the performance of KLFE, LFE, and PCA, we use an experiment with synthetic data, a Swiss roll (Fig. 1(a)). To generate 3-dimensional sample points $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}, \mathbf{x}_i^{(3)}]^T$ $(i = 1, \cdots, N)$ on a Swiss roll, we let $\mathbf{x}_i^{(1)} = \theta \times \cos(\theta)$ and $\mathbf{x}_i^{(2)} = \theta \times \sin(\theta)$, where θ is a random variable uniformly distributed in $[0, 4\pi]$; $\mathbf{x}_i^{(3)}$ is a random variable uniformly distributed in $[0, 2]$; then the 3D sample points $\mathbf{x}_i$ $(i = 1, \cdots, N)$ are on a 3-D helix surface (Swiss roll). To evaluate pattern classification performance under various feature extraction schemes, we label sample points generated by $\theta \in [0, 2\pi]$ with $y = -1$ and label sample points generated by $\theta \in (2\pi, 4\pi]$ with $y = 1$. The 3-dimensional vector $\mathbf{x}_i$ is further mapped to a L-dimensional vector $\mathbf{z}_i$ by matrix $\mathbf{R}$, i.e., $\mathbf{z}_i = \mathbf{R}\mathbf{x}_i$, where matrix $\mathbf{R}$ is randomly generated and has dimension $L \times 3$. The purpose of mapping $\mathbf{x}_i$ to $\mathbf{z}_i$ is to add some irrelevant features and test whether a feature extraction scheme is able to perform well under irrelevant features. In this way, we obtain the input data set $\mathscr{D} = \{(\mathbf{z}_i, y_i) : i = 1, 2, \cdots, N\}$, where $\mathbf{z}_i \in \mathbb{R}^L$ and the class label $y_i \in \{-1, +1\}$. For each feature extraction scheme, we use K-Nearest-Neighbor (K=1) as the classifier so that we can evaluate the performance of feature extraction in terms of classification error rate. The classification error rates are averaged over 10 simulation runs, each with a different matrix $\mathbf{R}$. Fig. 2 shows the classification error rate vs. dimension L. We can see that KLFE and LFE achieve comparable performance, and both KLFE and LFE outperform PCA. When dimension $L = 1$, KLFE achieves better performance than LFE. In addition, KLFE is robust against the change of dimension L, because KLFE has an explicit mechanism to eliminate irrelevant features.

KERNEL-BASED CLUSTERING

Clustering partitions a set of objects into groups (clusters) so that objects in the same cluster are more similar (in some sense) to each other than to objects in other clusters. In this section, we describe three known clustering algorithms: kernel K-means, spectral clustering, and Self-Organizing-Queue (SOQ) based clustering [5].

The K-means algorithm is a widely used iterative clustering algorithm. In each iteration, the K-means algorithm computes a new centroid[1] μ_k for each cluster k and then updates the cluster members using the new centroids based on the nearest neighbor rule. Kernel K-means [6] is a kernel extension of K-means algorithm, which is summarized below.

Input: $\{\mathbf{x}_i : i = 1, \cdots, N\}$, kernel $\kappa(\cdot,\cdot)$, and the number of clusters K.

1. Initialize the K clusters and obtain $\{C_k^{(0)} : k = 1, \cdots, K\}$, where $C_k^{(t)}$ denotes the set containing all the members of Cluster k at Stage t.
2. Let $t = 0$.
3. For each $\mathbf{x}_i$ ($i = 1, \cdots, N$), update its new cluster index by $k^*(\mathbf{x}_i) = \arg\min_k ||\phi(\mathbf{x}_i) - \mu_k||_2^2$, where $||\phi(\mathbf{x}_i) - \mu_k||_2^2$ can be computed by

$$||\phi(\mathbf{x}_i) - \mu_k||_2^2 = ||\phi(\mathbf{x}_i) - \frac{1}{|C_k^{(t)}|} \sum_{\mathbf{x} \in C_k^{(t)}} \phi(\mathbf{x})||_2^2 \quad (9)$$

$$= \kappa(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{|C_k^{(t)}|} \sum_{\mathbf{x} \in C_k^{(t)}} \kappa(\mathbf{x}_i, \mathbf{x}) + \frac{1}{|C_k^{(t)}|^2} \sum_{\mathbf{x} \in C_k^{(t)}} \sum_{\mathbf{z} \in C_k^{(t)}} \kappa(\mathbf{x}, \mathbf{z}) \quad (10)$$

   Since we assume points $\{\phi(\mathbf{x}_i) : i = 1, \cdots, N\}$ form a linear geometric structure in the feature space due to flattening capability of $\phi(\cdot)$, we use the Euclidean distance in (9) instead of a geodesic distance used in a curved space. Again, in (10), the kernel trick bypasses direct computation of $\phi(\mathbf{x})$.

4. Update the membership of each cluster k ($k = 1, \cdots, K$) by $C_k^{(t+1)} = \{\mathbf{x}_i : k^*(\mathbf{x}_i) = k, i \in \{1, \cdots, N\}\}$.
5. If the termination criteria are not satisfied, let $t = t + 1$ and go to Step 3; otherwise, stop.

Output: $\{C_k^{(t+1)} : k = 1, \cdots, K\}$.

Spectral clustering techniques are widely used for graph clustering [7] or community detection [8], i.e., finding sets of "related" vertices (called communities) in a graph. Spectral clustering utilizes the spectrum of the Laplacian matrix $\mathbf{L}$ of a given graph for grouping the nodes, since the multiplicity K of the eigenvalue 0 of Laplacian $\mathbf{L}$ equals the number of connected components in the graph (denote these connected components by $A_1, \cdots, A_K$), and the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_K}$ of those components, where indicator vector $\mathbf{1}_{A_k} \in \mathbb{R}^N$, the i-th entry

[1] The centroid of a cluster is the arithmetic mean position of all the points/members in the cluster

of which is 1 if Node i belongs to A_k, and is 0 otherwise. Hence we can use the eigenvectors of eigenvalue 0 to obtain the indicator vectors $\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_K}$, which is exactly a partition of the graph into K connected components. Spectral clustering techniques can be categorized into unnormalized and normalized techniques. An unnormalized spectral clustering algorithm leverages the spectrum of the unnormalized Laplacian matrix of a given graph, while a normalized spectral clustering algorithm leverages the spectrum of the normalized Laplacian matrix. Spectral clustering can be regarded as a special type of weighted kernel K-means [6] since a weighted kernel K-means scheme can be reduced to an unnormalized/normalized spectral clustering scheme by choosing appropriate weight matrix and letting kernel matrix $\mathbf{K} = \mathbf{S}$, where $\mathbf{S}$ is an affinity matrix used in spectral clustering. The following shows an unnormalized spectral clustering algorithm.

Input: Affinity matrix $\mathbf{S}$ (where $\mathbf{S} \in \mathbb{R}^{N \times N}$), and the number of clusters K.

1. Compute the unnormalized Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{D}$ is a diagonal matrix whose diagonal entries are row-sum of $\mathbf{S}$.
2. Compute the K smallest eigenvalues and the corresponding eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_K$ of $\mathbf{L}$.
3. Let $\mathbf{U}$ ($\mathbf{U} \in \mathbb{R}^{N \times K}$) be a matrix containing vectors $\mathbf{u}_1, \cdots, \mathbf{u}_K$ as columns.
4. For $i = 1, \cdots, N$, let $\mathbf{y}_i$ ($\mathbf{y}_i \in \mathbb{R}^K$) be the vector corresponding to the i-th row of $\mathbf{U}$.
5. Use the K-means algorithm to partition $\{\mathbf{y}_i : i = 1, \cdots, N\}$ into clusters $C_1, \cdots, C_K$.
6. Let $A_k = \{j : \mathbf{y}_j \in C_k\}$ ($k = 1, \cdots, K$), where A_k contains the indices of nodes that belong to Cluster k.

Output: $\{A_k : k = 1, \cdots, K\}$.

The performance of existing spectral clustering techniques is not satisfactory for many applications. To improve the performance, a bio-inspired approach called Self-Organizing-Queue (SOQ) [5] was proposed for the graph clustering problem. The key idea of SOQ is to enable fictitious queues of intelligent nodes with self-organizing decision capability to choose a queue with most friends to join so that closely-related nodes are grouped into the same cluster/queue. The SOQ clustering algorithm is given as below.

Input: a set of N nodes, affinity matrix $\mathbf{S}$ (where $\mathbf{S} \in \mathbb{R}^{N \times N}$), and the number of clusters K.

1. Initialization: divide the set of N nodes into K queues; assign a queue to Current_Queue; Flag=1.
2. While (Flag)
3.   **WHO:** Choose *who* in Current_Queue as Current_Person.
4.   **HOW:** (*How* to) select a queue as Next_Queue for Current_Person to join.
5.   **WHERE:** (*Where* to) place Current_Person in Next_Queue.

6. Assign Next_Queue to Current_Queue.

7. **WHEN:** (*When* to) let Flag=0, i.e., stop the loop.

8. Endwhile

Output: the resulting K queues/clusters.

The key features of SOQ are: 1) self-organization, i.e., each node has the ability to decide where it wants to join; 2) the similarity matrix $\mathbf{S}$ can be asymmetric, and the entries in $\mathbf{S}$ can take any real value, including negative values. Note that none of the existing spectral clustering algorithms allows asymmetric similarity matrix and similarity matrix with negative entries.

There are many variations of SOQ, depending on how Step 1, 3, 4, 5, and 7 are implemented. For example, in Step 3 (WHO), we can choose the head of Current_Queue as Current_Person; in Step 4 (HOW), Current Person i can use the following criterion (called *Most Friends*) to choose Queue $\hat{k}$ as the Next Queue to join:

$$\hat{k} = \arg\max_{k} \frac{\sum_{j \in C_k} (s_{i,j} + s_{j,i})}{|C_k|} \tag{11}$$

where $s_{i,j}$ is the entry of $\mathbf{S}$ at row i and column j, and C_k is the set of indices of members in Queue k; in Step 5 (WHERE), we can place Current_Person at the tail of Next_Queue. Due to the self-organizing decision capability, SOQ clustering scheme achieves better clustering performance than existing spectral clustering techniques and K-means algorithm for many applications [5].

To compare the performance of representative kernel based clustering schemes, i.e., unnormalized spectral clustering (USC) [6], normalized cut (ncut) [6], and SOQ, as well as K-means, we conduct two experiments. Since unnormalized and normalized spectral clustering algorithms can be regarded as special types of weighted kernel K-means, we use spectral clustering to represent kernel K-means as well.

The first experiment uses synthetic data consisting of 2-D Gaussian-distributed sample points. To simulate four clusters, we use four 2-D Gaussian distributions with the same standard deviation of 0.05 and mean (-0.3, 0), (0, 0), (0.3, 0), and (0.6, 0), respectively, and each 2-D Gaussian distribution corresponds to one cluster; the two dimensions of the 2-D Gaussian are independent and identically distributed. The number of samples for the four clusters are 105, 15, 15, and 15, respectively, and the total number of points N is 150. Fig. 3 shows the 2-D positions of the 150 sample points. For spectral clustering algorithms, an affinity matrix is needed. Let the generated 2-D points be $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N$, with $\mathbf{p}_n = (x_n, y_n)$ for $1 \leq n \leq N$. We generate the affinity measure $s_{i,j}$ between any two points $\mathbf{p}_i$ and $\mathbf{p}_j$ by $s_{i,j} = \exp(-||\mathbf{p}_i - \mathbf{p}_j||_2^2/(2\sigma^2))$, where $\sigma = 1$ in this experiment. In this way, we obtain an affinity matrix $\mathbf{S}$.

We run the algorithms 20 times, each with a different randomly permuted input, to obtain 20 clustering results. By comparing to the ground truth in Fig. 3, we obtain the error rate for each experiment. For the 20 experiments, we calculate the mean clustering error rate and 95% confidence interval, which are listed in the second column of Table 1, where μ_{error} denotes the mean clustering error rate and $\mu_{error} \pm \nu$ denotes upper/lower bound of the confidence interval, respectively. Table 1 demonstrates that SOQ significantly outperforms K-means, USC, and ncut for this synthetic data set.

The second experiment uses real-world data, consisting of images of handwritten digits, which are described in [9] and are downloadable from [10]. The 10 digits data set is used in our experiment.

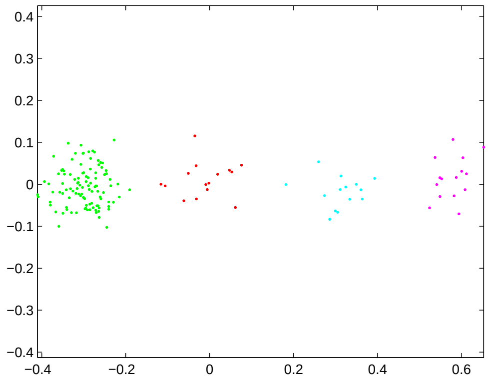


Figure 3: Positions of the two-dimensional sample points in a 2D plane; points with the same color belong to the same cluster (i.e., generated by the same distribution).

Table 1: Error rate

$\mu_{error} \pm \nu$	Synthetic Data	Handwritten Digits
K-means	0.3090 ± 0.0865	0.2661 ± 0.0349
USC	0.3483 ± 0.0620	0.3704 ± 0.0219
ncut	0.5020 ± 0.0474	0.3228 ± 0.0204
SOQ	0 ± 0	0.1603 ± 0.0192

There are 10 clusters in the data set, with 100 members in each cluster. Again, we run the algorithms 20 times, each with different randomly permuted input. For the 20 experiments, we calculate the mean clustering error rate and 95% confidence interval, which are listed in the third column of Table 1. Table 1 demonstrates that SOQ significantly outperforms K-means, USC, and ncut for this set of handwritten digits.

CONCLUSION

In this column, we have discussed kernel methods as pattern analysis tools, and provided insights in two important pattern analysis problems, namely, feature extraction and clustering.

Kernel methods have been widely applied to computer vision, image processing, information retrieval, text mining, handwriting recognition, geostatistics, kriging, bioinformatics, chemoinformatics, information extraction, among others. It is expected that kernel methods will provide valuable pattern analysis tools for emerging big data applications.

AUTHORS

Qiuyuan Huang (idfree@ufl.edu) is a Ph.D. candidate in Department of Electrical and Computer Engineering at University of Florida, Gainesville, FL. She received B.S. in Computer Science and

Engineering from University of Science and Technology of China, Hefei, China, in 2011. Her research interests are signal processing, machine learning, networking, social networks, and smart grid.

Dapeng Oliver Wu (wu@ece.ufl.edu) is a professor with Electrical and Computer Engineering Department at University of Florida, Gainesville, FL. His research interests are in the areas of networking, communications, signal processing, computer vision, and machine learning. He received University of Florida Research Foundation Professorship Award in 2009, AFOSR Young Investigator Program (YIP) Award in 2009, ONR Young Investigator Program (YIP) Award in 2008, NSF CAREER award in 2007, the IEEE Circuits and Systems for Video Technology (CSVT) Transactions Best Paper Award for Year 2001, and the Best Paper Awards in IEEE GLOBECOM 2011 and International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine) 2006.

Currently, he serves as an Associate Editor for IEEE Transactions on Circuits and Systems for Video Technology, Journal of Visual Communication and Image Representation, and International Journal of Ad Hoc and Ubiquitous Computing. He is the founder of IEEE Transactions on Network Science and Engineering. He was the founding Editor-in-Chief of Journal of Advances in Multimedia between 2006 and 2008, and an Associate Editor for IEEE Transactions on Wireless Communications and IEEE Transactions on Vehicular Technology between 2004 and 2007. He is also a guest-editor for IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Cross-layer Optimized Wireless Multimedia Communications. He has served as Technical Program Committee (TPC) Chair for IEEE INFOCOM 2012, and TPC chair for IEEE International Conference on Communications (ICC 2008), Signal Processing for Communications Symposium, and as a member of executive committee and/or technical program committee of over 80 conferences. He has served as Chair for the Award Committee, and Chair of Mobile and wireless multimedia Interest Group (MobIG), Technical Committee on Multimedia Communications, IEEE Communications Society. He was a member of Multimedia Signal Processing Technical Committee, IEEE Signal Processing Society from Jan. 1, 2009 to Dec. 31, 2012. He is an IEEE Fellow.

References

[1] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[2] J. Wang, J. Fan, H. Li, and D. Wu, "Kernel-based feature extraction under maximum margin criterion," *Journal of Visual Communication and Image Representation*, vol. 23, no. 1, pp. 53–62, 2012.

[3] Y. Sun and D. Wu, "A relief based feature extraction algorithm," in *Proceedings of SIAM International Conference on Data Mining*, 2008, pp. 188–195.

[4] ——, "Feature extraction through local learning," *Statistical Analysis and Data Mining*, vol. 2, no. 1, pp. 34–47, 2009.

[5] B. Sun and D. Wu, "Self-organizing-queue based clustering," *IEEE Signal Processing Letters*, vol. 19, no. 12, pp. 902–905, 2012.

[6] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 551–556.

[7] S. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[8] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[9] D. Verma and M. Meila, "A comparison of spectral clustering algorithms," *University of Washington, Tech. Rep. UW-CSE-03-05-01*, 2003.

[10] ——, "digit1000.mat," 2003. [Online]. Available: http://www.stat.washington.edu/spectral/datasets.html