Manifold Matching for High-Dimensional Pattern Recognition

Seiji Hotta
Tokyo University of Agriculture and Technology
Japan

1. Introduction

In pattern recognition, a kind of classical classifier called *k-nearest neighbor rule* (kNN) has been applied to many real-life problems because of its good performance and simple algorithm. In kNN, a test sample is classified by a majority vote of its *k*-closest training samples. This approach has the following advantages: (1) It was proved that the error rate of kNN approaches the Bayes error when both the number of training samples and the value of *k* are infinite (Duda et al., 2001). (2) kNN performs well even if different classes overlap each other. (3) It is easy to implement kNN due to its simple algorithm. However, kNN does not perform well when the dimensionality of feature vectors is large. As an example, Fig. 1 shows a test sample (belonging to class 5) of the MNIST dataset (LeCun et al., 1998) and its five closest training samples selected by using Euclidean distance. Because the selected five training samples include the three samples belonging to class 8, the test sample is misclassified into class 8. Such misclassification is often yielded by kNN in high-dimensional pattern classification such as character and face recognition. Moreover, kNN requires a large number of training samples for high accuracy because kNN is a kind of memory-based classifiers. Consequently, the classification cost and memory requirement of kNN tend to be high.

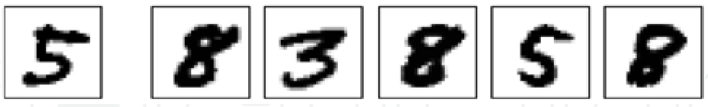

Fig. 1. An example of a test sample (leftmost). The others are five training samples closest to the test sample.

For overcoming these difficulties, classifiers using subspaces or linear manifolds (affine subspace) are used for real-life problems such as face recognition. Linear manifold-based classifiers can represent various artificial patterns by linear combinations of the small number of bases. As an example, a two-dimensional linear manifold spanned by three handwritten digit images '4' is shown in Fig. 2. Each of the corners of the triangle represents pure training samples, whereas the images in between are linear combinations of them. These intermediate images can be used as artificial training samples for classification. Due to this property, manifold-based classifiers tend to outperform kNN in high-dimensional pattern classification. In addition, we can reduce the classification cost and memory requirement of manifold-based classifiers easily compared to kNN. However, bases of linear

Source: Pattern Recognition Techniques, Technology and Applications, Book edited by: Peng-Yeng Yin,
ISBN 978-953-7619-24-4, pp. 626, November 2008, I-Tech, Vienna, Austria

manifolds have an effect on classification accuracy significantly, so we have to select them carefully. Generally, orthonormal bases obtained with *principal component analysis* (PCA) are used for forming linear manifolds, but there is no guarantee that they are the best ones for achieving high accuracy.

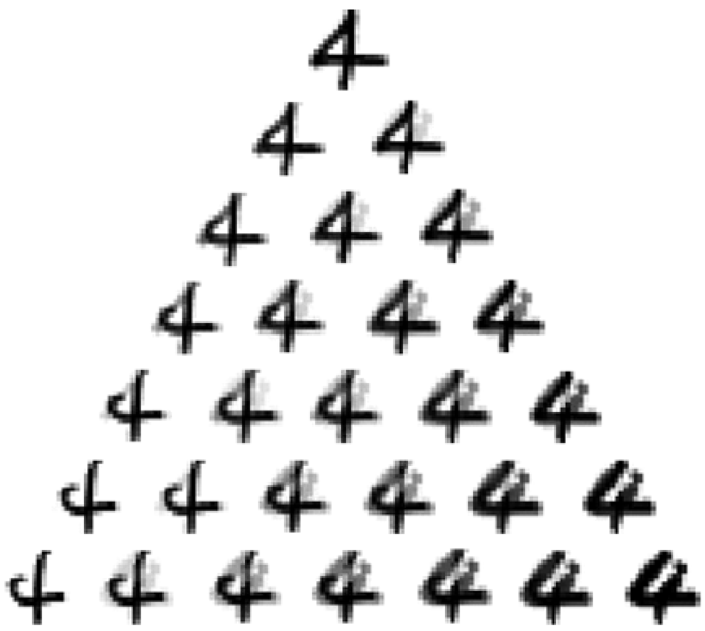

Fig. 2. A two-dimensional linear manifold spanned by three handwritten digit images '4' in the corners.

In this chapter, we consider about achieving high accuracy in high-dimensional pattern classification using linear manifolds. Henceforth, classification using linear manifolds is called *manifold matching* for short. In manifold matching, a test sample is classified into the class that minimizes the residual length from a test sample to a manifold spanned by training samples. This classification rule can be derived from optimization for reconstructing a test sample from training samples of each class. Hence, we start with describing square error minimization between a test sample and a linear combination of training samples. Using the solutions of this minimization, we can define the classification rule for manifold matching easily. Next, this idea is extended to the distance between two linear manifolds. This distance is useful for incorporating transform-invariance into image classification. After that, accuracy improvement through kernel mapping and transform-invariance is adopted to manifold matching. Finally, learning rules for manifold matching are proposed for reducing classification cost and memory requirement without accuracy deterioration. In this chapter, we deal with handwritten digit images as an example of high-dimensional patterns. Experimental results on handwritten digit datasets show that manifold-based classification performs as well or better than state-of-the-art such as a support vector machine.

2. Manifold matching

In general, linear manifold-based classifiers are derived with *principal component analysis* (PCA). However, in this section, we start with square error minimization between a test sample and a linear combination of training samples. In pattern recognition, we should not compute the distance between two patterns until we had transformed them to be as similar to one another as possible (Duda et al., 2001). From this point of view, measuring of a distance between a test point and each class is formalized as a square error minimization problem in this section.

Let us consider a classifier that classifies a test sample into the class to which the most similar linear combination of training samples belongs. Suppose that a d-dimensional training sample $\boldsymbol{x}_i^j = (x_{i1}^j \cdots x_{id}^j)^\top \in \mathbb{R}^d$ $(i = 1, ..., n_j)$ belonging to class j (j = 1, ...,C), where n_j and C are the numbers of classes and training samples in class j, respectively. The notation $\top$ denotes the transpose of a matrix or vector. Let $\mathbf{X}_j = (\boldsymbol{x}_1^j|\boldsymbol{x}_2^j|\cdots|\boldsymbol{x}_{n_j}^j) \in \mathbb{R}^{d\times n_j}$ be the matrix of training samples in class j. If these training samples are linear independent, they are not necessary to be orthogonal each other.

Given a test sample $\boldsymbol{q} = (q_1 \dots q_d)^\top \in \mathbb{R}^d$, we first construct linear combinations of training samples from individual classes by minimizing the cost for reconstructing a test sample from $\mathbf{X}_j$ before classification. For this purpose, the reconstruction error is measured by the following square error:

$$\begin{aligned} \min_{\boldsymbol{b}_j} \; & \|\boldsymbol{q} - \mathbf{X}_j\boldsymbol{b}_j\|^2 = \left\|\boldsymbol{q} - \sum_{i=1}^{n_j} b_i^j \boldsymbol{x}_i^j\right\|^2 \\ \text{s.t.} \; & \boldsymbol{b}_j^\top \mathbf{1}_{n_j} = \sum_{i=1}^{n_j} b_i^j = 1, \end{aligned} \tag{1}$$

where $\boldsymbol{b}_j = (b_1^j \cdots b_{n_j}^j)^\top \in \mathbb{R}^{n_j}$ is a weight vector for the linear combination of training samples from class j, and $\mathbf{1}_{n_j} = (1 \cdots 1)^\top \in \mathbb{R}^{n_j}$ is a vector of which all elements are 1. The same cost function can be found in the first step of locally linear embedding (Roweis & Saul, 2000). The optimal weights subject to sum-to-one are found by solving a least-squares problem. Note that the above cost function is equivalent to $\|(\mathbf{Q}-\mathbf{X}_j)\boldsymbol{b}_j\|^2$ with $\mathbf{Q} = (\boldsymbol{q}|\boldsymbol{q}|\cdots|\boldsymbol{q}) \in \mathbb{R}^{d\times n_j}$ due to the constraint $\boldsymbol{b}_j^\top\mathbf{1}_{n_j} = 1$. Let us define $\mathbf{C}_j = (\mathbf{Q} - \mathbf{X}_j)^\top(\mathbf{Q} - \mathbf{X}_j)$, and by using it, Eq. (1) becomes

$$\begin{aligned} \min_{\boldsymbol{b}_j} \; & \boldsymbol{b}_j^\top \mathbf{C}_j \boldsymbol{b}_j \\ \text{s.t.} \; & \boldsymbol{b}_j^\top \mathbf{1}_{n_j} = 1. \end{aligned} \tag{2}$$

The solution of the above constrained minimization problem can be given in closed form by using Lagrange multipliers. The corresponding Lagrangian function is given as

$$J_L(\boldsymbol{b}_j) = \frac{1}{2}\boldsymbol{b}_j^\top \mathbf{C}_j \boldsymbol{b}_j + \lambda(\boldsymbol{b}_j^\top \mathbf{1}_{n_j} - 1), \tag{3}$$

where λ is the Lagrange multiplier. Setting the derivative of Eq. (3) to zero and substituting the constraint $\boldsymbol{b}_j^\top \mathbf{1}_{n_j} = 1$ into the derivative, the following optimal weight is given:

$$\boldsymbol{b}_j = \frac{\mathbf{C}_j^{-1}\mathbf{1}_{n_j}}{\mathbf{1}_{n_j}^\top \mathbf{C}_j^{-1}\mathbf{1}_{n_j}}. \tag{4}$$

Regularization is applied to $\mathbf{C}_j$ before inversion for avoiding over fitting or if $n_j > d$ using a regularization parameter $\alpha > 0$ and an identity matrix $\mathbf{I}_{n_j} \in \mathbb{R}^{n_j \times n_j}$ such as $\mathbf{C}_j + \alpha \mathbf{I}_{n_j}$.

In the above optimization problem, we can get rid of the constraint $\boldsymbol{b}_j^\top \mathbf{1}_{n_j} = 1$ by transforming the cost function from $\|\boldsymbol{q} - \mathbf{X}_j \boldsymbol{b}_j\|^2$ to $\|\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)\|^2$, where $\boldsymbol{m}_j$ is the centroid of class j, i.e., $\boldsymbol{m}_j = \sum_{i=1}^{n_j} \boldsymbol{x}_i^j / n_j$, and $\bar{\mathbf{X}}_j = (\boldsymbol{x}_1^j - \boldsymbol{m}_j | \cdots | \boldsymbol{x}_{n_j}^j - \boldsymbol{m}_j) \in \mathbb{R}^{d \times n_j}$, respectively. By this transformation, Eq. (1) becomes

$$\min_{\boldsymbol{b}_j} \|\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)\|^2. \tag{5}$$

By setting the derivative of Eq. (5) to zero, the optimal weight is given as follows:

$$\boldsymbol{b}_j = (\bar{\mathbf{X}}_j^\top \bar{\mathbf{X}}_j)^{-1} \bar{\mathbf{X}}_j^\top (\boldsymbol{q} - \boldsymbol{m}_j). \tag{6}$$

Consequently, the distance between $\boldsymbol{q}$ and the linear combination of class j is measured by

$$\begin{aligned} d_j &= \|\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)\|^2 \\ &= \|\boldsymbol{q} - \boldsymbol{m}_j\|^2 - 2(\boldsymbol{q} - \boldsymbol{m}_j)^\top \bar{\mathbf{X}}_j \boldsymbol{b}_j + \boldsymbol{b}_j^\top \bar{\mathbf{X}}_j^\top \bar{\mathbf{X}}_j \boldsymbol{b}_j \\ &= \|\boldsymbol{q} - \boldsymbol{m}_j\|^2 - (\boldsymbol{q} - \boldsymbol{m}_j)^\top \bar{\mathbf{X}}_j (\bar{\mathbf{X}}_j^\top \bar{\mathbf{X}}_j)^{-1} \bar{\mathbf{X}}_j (\boldsymbol{q} - \boldsymbol{m}_j) \\ &= \|\boldsymbol{q} - \boldsymbol{m}_j\|^2 - \|\mathbf{V}_j^\top (\boldsymbol{q} - \boldsymbol{m}_j)\|^2, \end{aligned} \tag{7}$$

where $\mathbf{V}_j \in \mathbb{R}^{d \times r}$ is the eigenvectors of $\bar{\mathbf{X}}_j \bar{\mathbf{X}}_j^\top \in \mathbb{R}^{d \times d}$, where r is the rank of $\bar{\mathbf{X}}_j \bar{\mathbf{X}}_j^\top$. This equality means that the distance d_j is given as a residual length from $\boldsymbol{q}$ to a r-dimensional linear manifold (affine subspace) of which origin is $\boldsymbol{m}_j$ (cf. Fig. 3). In this chapter, a manifold spanned by training samples is called *training manifold*.

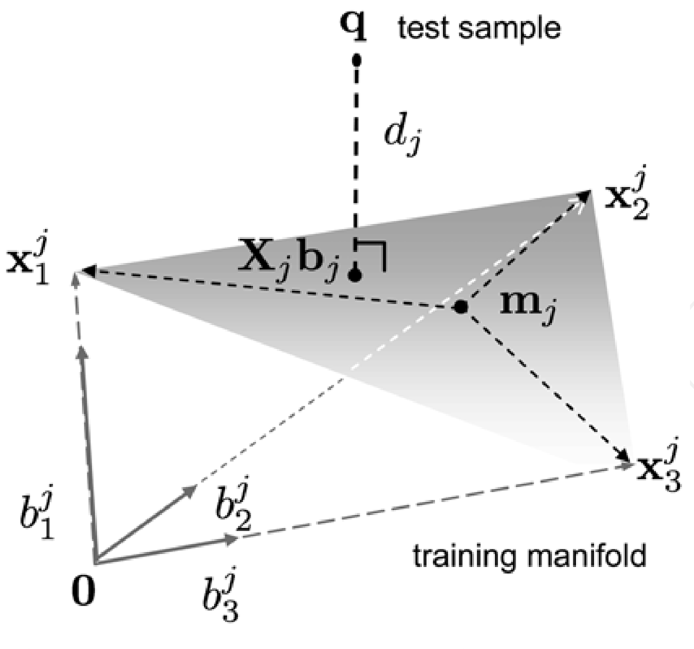

Fig. 3. Concept of the shortest distance between $\boldsymbol{q}$ and the linear combination of training samples that exists on a training manifold.

In a classification phase, the test sample $\boldsymbol{q}$ is classified into the class that has the shortes distance from $\boldsymbol{q}$ to the linear combination existing on the linear manifold. That is we define the distance between $\boldsymbol{q}$ and class j as $d_j = \|\boldsymbol{q} - \mathbf{X}_j\boldsymbol{b}_j\|^2$ or $d_j = \|\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j\boldsymbol{b}_j)\|^2$, test sample's class (denoted by ω) is determined by the following classification rule:

$$\omega = \arg\min_j d_j. \tag{8}$$

The above classification rule is called with different names according to the way of selection the set of training samples $\mathbf{X}_j$. When we select the k-closest training samples of $\boldsymbol{q}$ from each class, and use them as $\mathbf{X}_j$, the classification rule is called *local subspace classifier* (LSC) (Laaksonen, 1997; Vincent & Bengio, 2002). When all elements of $\boldsymbol{b}_j$ in LSC are equal to $1/k$, LSC is called local-mean based classifier (Mitani & Hamamoto, 2006). In addition, if we use an image and its tangent vector as $\boldsymbol{m}_j$ and $\bar{\mathbf{X}}_j$ respectively in Eq. (7), the distance is called *one-sided tangent distance* (1S-TD) (Simard et al., 1993). These classifier and distance are described again in the next section. Finally, when we use the $r' \ll r$ eigenvectors corresponding to the r' largest eigenvalues of $\bar{\mathbf{X}}_j\bar{\mathbf{X}}_j^\top$ as $\mathbf{V}_j$, the rule is called *projection distance method* (PDM) (Ikeda et al., 1983) that is a kind of subspace classifiers. In this chapter, classification using the distance between a test sample and a training manifold is called *one-sided manifold matching* (1S-MM).

2.1 Distance between two linear manifolds

In this section, we assume that a test sample is given by the set of vector. In this case the dissimilarity between test and training data is measured by the distance between two linear manifolds. Let $\mathbf{Q} = (\boldsymbol{q}_1 | \boldsymbol{q}_2 | \dots | \boldsymbol{q}_m) \in \mathbb{R}^{d\times m}$ be the set of m test vectors, where $\boldsymbol{q}_i = (q_{i1} \cdots q_{id})^\top \in \mathbb{R}^d$ $(i = 1, ..., m)$ is the ith test vector. If these test vectors are linear independent, they are not necessary to be orthogonal each other. Let $\boldsymbol{a} = (a_1 \dots a_m)^\top \in \mathbb{R}^m$ is a weight vector for a linear combination of test vectors.

By developing Eq. (1) to the reconstruction error between two linear combinations, the following optimization problem can be formalized:

$$\begin{aligned} \min_{\boldsymbol{a},\boldsymbol{b}} \quad & \|\mathbf{Q}\boldsymbol{a} - \mathbf{X}\boldsymbol{b}\|^2 \\ \text{s.t.} \quad & \boldsymbol{a}^\top \mathbf{1}_m = 1,\ \boldsymbol{b}^\top \mathbf{1}_n = 1, \end{aligned} \tag{9}$$

The solutions of the above optimization problem can be given in closed form by using Lagrange multipliers. However, they have complex structures, so we get rid of the two constraints $\boldsymbol{a}^\top \mathbf{1}_m = 1$ and $\boldsymbol{b}^\top \mathbf{1}_n = 1$ by transformating the cost function from $\|\mathbf{Q}\boldsymbol{a} - \mathbf{X}\boldsymbol{b}\|^2$ to $\|(\boldsymbol{m}_q + \bar{\mathbf{Q}}\boldsymbol{a}) - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j\boldsymbol{b}_j)\|^2$, where $\boldsymbol{m}_q$ and $\bar{\mathbf{Q}}$ are the centroid of test vectors (i.e., $\boldsymbol{m}_q = \Sigma_{i=1}^m \boldsymbol{q}_i/m$) and $\bar{\mathbf{Q}} = (\boldsymbol{q}_1 - \boldsymbol{m}_q | \dots | \boldsymbol{q}_m - \boldsymbol{m}_q) \in \mathbb{R}^{d\times m}$, respectively. By this transformation, Eq. (9) becomes

$$\min_{\boldsymbol{a},\boldsymbol{b}} \|(\boldsymbol{m}_q + \bar{\mathbf{Q}}\boldsymbol{a}) - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j\boldsymbol{b})\|^2. \tag{10}$$

The above minimization problem can be regarded as the distance between two manifolds (cf. Fig. 4). In this chapter, a linear manifold spanned by test samples is called *test manifold*.

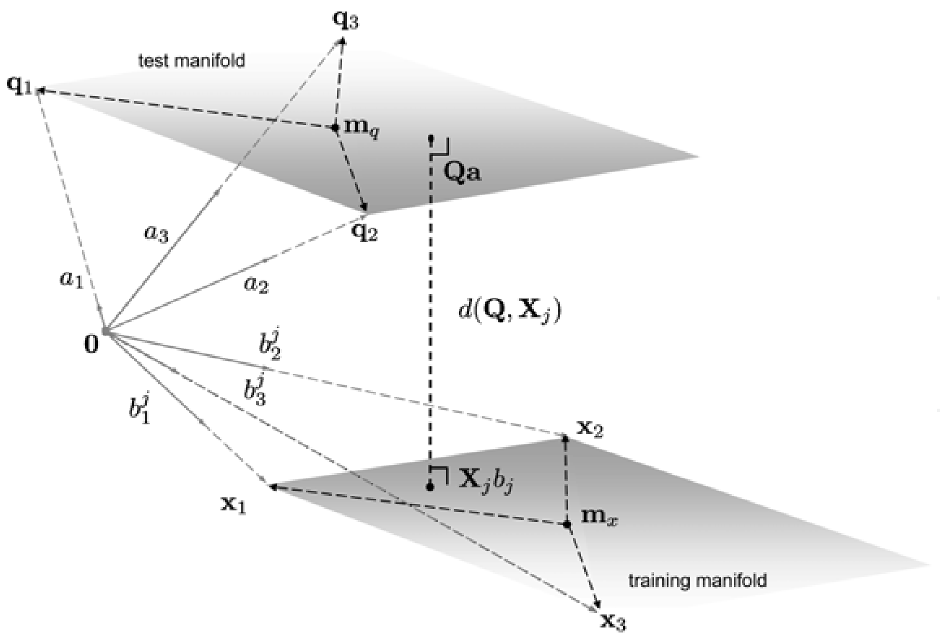

Fig. 4. Concept of the shortest distance between a test manifold and a training manifold.

The solutions of Eq. (10) are given by setting the derivative of Eq. (10) to zero. Consequently, the optimal weights are given as follows:

$$\boldsymbol{a} = (\mathbf{Q}_1 - \mathbf{Q}_2\mathbf{X}_1^{-1}\mathbf{X}_2)^{-1}(\mathbf{Q}_2\mathbf{X}_1^{-1}\bar{\mathbf{X}}_j^\top - \bar{\mathbf{Q}}^\top)(\boldsymbol{m}_q - \boldsymbol{m}_j), \tag{11}$$

$$\boldsymbol{b} = (\mathbf{X}_1 - \mathbf{X}_2\mathbf{Q}_1^{-1}\mathbf{Q}_2)^{-1}(\bar{\mathbf{X}}_j^\top - \mathbf{X}_2\mathbf{Q}_1^{-1}\bar{\mathbf{Q}}^\top)(\boldsymbol{m}_q - \boldsymbol{m}_j), \tag{12}$$

where

$$\mathbf{Q}_1 = \bar{\mathbf{Q}}^\top\bar{\mathbf{Q}},\ \mathbf{X}_1 = \bar{\mathbf{X}}_j^\top\bar{\mathbf{X}}_j, \tag{13}$$

$$\mathbf{Q}_2 = \bar{\mathbf{Q}}^\top\bar{\mathbf{X}}_j,\ \mathbf{X}_2 = \bar{\mathbf{X}}_j^\top\bar{\mathbf{Q}}. \tag{14}$$

If necessary, regularization is applied to $\mathbf{Q}_1$ and $\mathbf{X}_1$ before inversion using regularization parameters α_1, $\alpha_2 > 0$ and identity matrices $\mathbf{I}_m \in \mathbb{R}^{m\times m}$ and $\mathbf{I}_{n_j} \in \mathbb{R}^{n_j\times n_j}$ such as $\mathbf{Q}_1 + \alpha_1\mathbf{I}_m$ and $\mathbf{X}_1 + \alpha_2\mathbf{I}_{n_j}$.

In a classification phase, the test vectors $\mathbf{Q}$ is classified into the class that has the shortest distance from $\mathbf{Q}\boldsymbol{a}$ to the $\mathbf{X}_j\boldsymbol{b}_j$. That is we define the distance between a test manifold and a training manifold as $d(\mathbf{Q}, \mathbf{X}_j) = \|(\boldsymbol{m}_q + \bar{\mathbf{Q}}\boldsymbol{a}) - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j\boldsymbol{b}_j)\|^2$, and the class of the test manifold (denoted by ω) is determined by the following classification rule:

$$\omega = \arg\min_j d(\mathbf{Q}, \mathbf{X}_j). \tag{15}$$

The above classification rule is also called by different names according to the way of selecting the sets of test and training, i.e., $\mathbf{Q}$ and $\mathbf{X}_j$. When two linear manifolds are represented by orthonormal bases obtained with PCA, the classification rule of Eq. (15) is called inter-subspace distance (Chen et al., 2004). When $\boldsymbol{m}_q$, $\boldsymbol{m}_j$ are bitmap images and $\bar{\mathbf{Q}}$, $\bar{\mathbf{X}}_j$ are their tangent vectors, the distance $d(\mathbf{Q},\mathbf{X}_j)$ is called *two-sided tangent distance* (2S-TD) (Simard et al., 1993). In this chapter, classification using the distance between two linear manifolds is called *two-sided manifold matching* (2S-MM).

3. Accuracy improvement

We encounter different types of geometric transformations in image classification. Hence, it is important to incorporate transform-invariance into classification rules for achieving high accuracy. Distance-based classifiers such as kNN often rely on simple distances such as Euclidean distance, thus they suffer a high sensitivity to geometric transformations of images such as shifts, scaling and others. Distances in manifold-matching are measured based on a square error, so they are also not robust against geometric transformations. In this section, two approaches of incorporating transform-invariance into manifold matching are introduced. The first is to adopt kernel mapping (Schölkopf & Smola, 2002) to manifold matching. The second is combining *tangent distance* (TD) (Simard et al., 1993) and manifold matching.

3.1 Kernel manifold matching

First, let us consider adopting kernel mapping to 1S-MM. The extension from a linear classifier to nonlinear one can be achieved by a kernel trick $\Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{y}) = K(\boldsymbol{x}, \boldsymbol{y})$ for mapping samples from an input space to a feature space $\mathbb{R}^d \mapsto \mathcal{F}$ (Schölkopf & Smola, 2002). By applying kernel mapping to Eq. (1), the optimization problem becomes

$$\begin{aligned} &\min_{\boldsymbol{b}_j} \|(\mathbf{Q}^\Phi - \mathbf{X}_j^\Phi)\boldsymbol{b}_j\|^2 \\ &\text{s.t.} \quad \boldsymbol{b}_j^\top \mathbf{1}_{n_j} = 1, \end{aligned} \tag{16}$$

where $\mathbf{Q}^\Phi$ and $\mathbf{X}_j^\Phi$ are defined as $\mathbf{Q}^\Phi = (\Phi(\boldsymbol{q})|\cdots|\Phi(\boldsymbol{q}))$ and $\mathbf{X}_j^\Phi = (\Phi(\boldsymbol{x}_1^j)|\cdots|\Phi(\boldsymbol{x}_{n_j}^j))$, respectively. By using the kernel trick and Lagrange multipliers, the optimal weight is given by the following:

$$\boldsymbol{b}_j = \frac{\mathbf{K}_j^{-1}\mathbf{1}_{n_j}}{\mathbf{1}_{n_j}^\top \mathbf{K}_j^{-1}\mathbf{1}_{n_j}}, \tag{17}$$

where $\mathbf{K}_j \in \mathbb{R}^{n_j \times n_j}$ is a kernel matrix of which the (*k*, *l*)-element is given as

$$(\mathbf{K}_j)_{kl} = K(\boldsymbol{q}, \boldsymbol{q}) - K(\boldsymbol{q}, \boldsymbol{x}_l^j) - K(\boldsymbol{x}_k^j, \boldsymbol{q}) + K(\boldsymbol{x}_k^j, \boldsymbol{x}_l^j). \tag{18}$$

When applying kernel mapping to Eq. (5), kernel PCA (Schölkopf et al., 1998) is needed for obtaining orthonormal bases in $\mathcal{F}$. Refer to (Maeda & Murase, 2002) or (Hotta, 2008a) for more details.

Next, let us consider adopting kernel mapping to 2S-MM. By applying kernel mapping to Eq. (10), the optimization problem becomes

$$\min_{\boldsymbol{a},\boldsymbol{b}} \left\| \Phi(\boldsymbol{m}_q) + \bar{\mathbf{Q}}^\Phi \boldsymbol{a} - (\Phi(\boldsymbol{m}_x) + \bar{\mathbf{X}}_j^\Phi \boldsymbol{b}) \right\|^2, \tag{19}$$

where $\Phi(\boldsymbol{m}_q)$, $\Phi(\boldsymbol{m}_x)$, $\bar{\mathbf{Q}}^\Phi$, and $\bar{\mathbf{X}}_j^\Phi$ are given as follows:

$$\Phi(\boldsymbol{m}_q) = \frac{1}{m}\sum_{k=1}^{m}\Phi(\boldsymbol{q}_k),\ \Phi(\boldsymbol{m}_x) = \frac{1}{n_j}\sum_{l=1}^{n_j}\Phi(\boldsymbol{x}_l^j), \tag{20}$$

$$\bar{\mathbf{Q}}^{\Phi} = (\Phi(\boldsymbol{q}_1) - \Phi(\boldsymbol{m}_q)|\Phi(\boldsymbol{q}_2) - \Phi(\boldsymbol{m}_q)|\cdots|\Phi(\boldsymbol{q}_m) - \Phi(\boldsymbol{m}_q))\,, \tag{21}$$

$$\bar{\mathbf{X}}_j^{\Phi} = \left(\Phi(\boldsymbol{x}_1^j) - \Phi(\boldsymbol{m}_x)|\Phi(\boldsymbol{x}_2^j) - \Phi(\boldsymbol{m}_x)|\cdots|\Phi(\boldsymbol{x}_{n_j}^j) - \Phi(\boldsymbol{m}_x)\right). \tag{22}$$

By setting the derivative of Eq. (19) to zero and using the kernel trick, the optimal weights are given as follows:

$$\boldsymbol{a} = (\mathbf{K}_{QQ} - \mathbf{K}_{QX}\mathbf{K}_{XX}^{-1}\mathbf{K}_{XQ})^{-1}(\mathbf{K}_{QX}\mathbf{K}_{XX}^{-1}\boldsymbol{k}_X - \boldsymbol{k}_Q), \tag{23}$$

$$\boldsymbol{b} = (\mathbf{K}_{XQ}\mathbf{K}_{QQ}^{-1}\mathbf{K}_{QX} - \mathbf{K}_{XX})^{-1}(\mathbf{K}_{XQ}\mathbf{K}_{QQ}^{-1}\boldsymbol{k}_Q - \boldsymbol{k}_X), \tag{24}$$

where $\mathbf{K}_{QQ} \in \mathbb{R}^{m\times m}$, $\mathbf{K}_{XX} \in \mathbb{R}^{n_j\times n_j}$, $\mathbf{K}_{QX} \in \mathbb{R}^{m\times n_j}$, $\mathbf{K}_{XQ} \in \mathbb{R}^{n_j\times m}$, $\boldsymbol{k}_Q \in \mathbb{R}^{m}$, and $\boldsymbol{k}_X \in \mathbb{R}^{n_j}$ of which the (*k*, *l*)-elements of matrices and the *l*th element of vectors are given by

$$\begin{aligned}(\mathbf{K}_{QQ})_{kl} &= K(\boldsymbol{q}_k, \boldsymbol{q}_l) - \frac{1}{m}\sum_{t=1}^{m}K(\boldsymbol{q}_k, \boldsymbol{q}_t)\\ &\quad - \frac{1}{m}\sum_{s=1}^{m}K(\boldsymbol{q}_s, \boldsymbol{q}_l) + \frac{1}{m^2}\sum_{s=1}^{m}\sum_{t=1}^{m}K(\boldsymbol{q}_s, \boldsymbol{q}_t),\end{aligned} \tag{25}$$

$$\begin{aligned}(\mathbf{K}_{XX})_{kl} &= K(\boldsymbol{x}_k^j, \boldsymbol{x}_l^j) - \frac{1}{n_j}\sum_{t=1}^{n_j}K(\boldsymbol{x}_k^j, \boldsymbol{x}_t^j)\\ &\quad - \frac{1}{n_j}\sum_{s=1}^{n_j}K(\boldsymbol{x}_s^j, \boldsymbol{x}_l^j) + \frac{1}{n_j^2}\sum_{s=1}^{n_j}\sum_{t=1}^{n_j}K(\boldsymbol{x}_s^j, \boldsymbol{x}_t^j),\end{aligned} \tag{26}$$

$$\begin{aligned}(\mathbf{K}_{QX})_{kl} &= K(\boldsymbol{q}_k, \boldsymbol{x}_l^j) - \frac{1}{n_j}\sum_{t=1}^{n_j}K(\boldsymbol{q}_k, \boldsymbol{x}_t^j)\\ &\quad - \frac{1}{m}\sum_{s=1}^{m}K(\boldsymbol{q}_s, \boldsymbol{x}_l^j) + \frac{1}{mn_j}\sum_{s=1}^{m}\sum_{t=1}^{n_j}K(\boldsymbol{q}_s, \boldsymbol{x}_t^j),\end{aligned} \tag{27}$$

$$\begin{aligned}(\mathbf{K}_{XQ})_{kl} &= K(\boldsymbol{x}_k^j, \boldsymbol{q}_l) - \frac{1}{m}\sum_{s=1}^{m}K(\boldsymbol{x}_k^j, \boldsymbol{q}_s)\\ &\quad - \frac{1}{n_j}\sum_{t=1}^{n_j}K(\boldsymbol{x}_t^j, \boldsymbol{q}_l) + \frac{1}{mn_j}\sum_{t=1}^{n_j}\sum_{s=1}^{m}K(\boldsymbol{x}_t^j, \boldsymbol{q}_s),\end{aligned} \tag{28}$$

$$\begin{aligned}(\boldsymbol{k}_Q)_l &= \frac{1}{m}\sum_{s=1}^{m}K(\boldsymbol{q}_l, \boldsymbol{q}_s) - \frac{1}{n_j}\sum_{t=1}^{n_j}K(\boldsymbol{q}_l, \boldsymbol{x}_t^j)\\ &\quad - \frac{1}{m^2}\sum_{s=1}^{m}\sum_{t=1}^{m}K(\boldsymbol{q}_s, \boldsymbol{q}_t) + \frac{1}{mn_j}\sum_{s=1}^{m}\sum_{t=1}^{n_j}K(\boldsymbol{q}_s, \boldsymbol{x}_t^j),\end{aligned} \tag{29}$$

$$(\boldsymbol{k}_X)_l = \frac{1}{m}\sum_{s=1}^{m} K(\boldsymbol{x}_l^j, \boldsymbol{q}_s) - \frac{1}{n_j}\sum_{t=1}^{n_j} K(\boldsymbol{x}_l^j, \boldsymbol{x}_t^j) - \frac{1}{mn_j}\sum_{t=1}^{n_j}\sum_{s=1}^{m} K(\boldsymbol{x}_t^j, \boldsymbol{q}_s) + \frac{1}{n_j^2}\sum_{s=1}^{n_j}\sum_{t=1}^{n_j} K(\boldsymbol{x}_s^j, \boldsymbol{x}_t^j). \tag{30}$$

In addition, Euclidean distance between $\Phi(\boldsymbol{m}_q)$ and $\Phi(\boldsymbol{m}_x)$ in $\mathcal{F}$ is given by

$$\begin{aligned} d_{qx}^{\Phi} &= \|\Phi(\boldsymbol{m}_q) - \Phi(\boldsymbol{m}_x)\|^2 \\ &= \frac{1}{m^2}\sum_{k=1}^{m}\sum_{s=1}^{m} K(\boldsymbol{q}_k, \boldsymbol{q}_s) - \frac{1}{mn_j}\sum_{k=1}^{m}\sum_{l=1}^{n_j} K(\boldsymbol{q}_k, \boldsymbol{x}_l^j) \\ &- \frac{1}{mn_j}\sum_{l=1}^{n_j}\sum_{k=1}^{m} K(\boldsymbol{x}_l^j, \boldsymbol{q}_k) + \frac{1}{n_j^2}\sum_{l=1}^{n_j}\sum_{t=1}^{n_j} K(\boldsymbol{x}_l^j, \boldsymbol{x}_t^j). \end{aligned} \tag{31}$$

Hence, the distance between a test manifold and a training manifold of class j in $\mathcal{F}$ is measured by

$$\begin{aligned} &d(\mathbf{Q}, \mathbf{X}_j)^{\Phi} = \\ &d_{qx}^{\Phi} + 2\boldsymbol{k}_Q^\top \boldsymbol{a} - 2\boldsymbol{k}_X^\top \boldsymbol{b} + \boldsymbol{a}^\top \mathbf{K}_{QQ}\boldsymbol{a} - \boldsymbol{a}^\top \mathbf{K}_{QX}\boldsymbol{b} - \boldsymbol{b}^\top \mathbf{K}_{XQ}\boldsymbol{a} + \boldsymbol{b}^\top \mathbf{K}_{XX}\boldsymbol{b}. \end{aligned} \tag{32}$$

If necessary, regularization is applied to $\mathbf{K}_{QQ}$ and $\mathbf{K}_{XX}$ such as $\mathbf{K}_{QQ} + \alpha_1 \mathbf{I}_m$, $\mathbf{K}_{XX} + \alpha_2 \mathbf{I}_{n_j}$.

For incorporating transform-invariance into kernel classifiers for digit classification, some kernels have been proposed in the past (Decoste & Sch¨olkopf, 2002; Haasdonk & Keysers, 2002). Here, we focus on a *tangent distance kernel* (TDK) because of its simplicity. TDK is defined by replacing Euclidean distance with a tangent distance in arbitrary distance-based kernels. For example, if we modify the following radial basis function (RBF) kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\beta\|\boldsymbol{x} - \boldsymbol{y}\|^2) \tag{33}$$

by replacing Euclidean distance with 2S-TD, we then obtain the kernel called *two sided TD kernel* (cf. Eq.(36)):

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\beta \times d_{2S}(\boldsymbol{x}, \boldsymbol{y})). \tag{34}$$

We can achieve higher accuracy by this simple modification than the use of the original RBF kernel (Haasdonk & Keysers, 2002). In addition, the above modification is adequate for kernel setting because of its natural definition and symmetric property.

3.2 Combination of manifold matching and tangent distance

Let us start with a brief review of tangent distance before introducing the way of combining manifold matching and tangent distance.

When an image $\boldsymbol{q}$ is transformed with small rotations that depend on one parameter α, and so the set of all the transformed images is given as a one-dimensional curve S_q (i.e., a nonlinear manifold) in a pixel space (see from top to middle in Fig. 5). Similarly, assume that the set of all the transformed images of another image $\boldsymbol{x}$ is given as a one-dimensional curve S_x. In this situation, we can regard the distance between manifolds S_q and S_x as an adequate dissimilarity for two images $\boldsymbol{q}$ and $\boldsymbol{x}$. For computational issue, we measure the distance between the corresponding tangent planes instead of measuring the strict distance between their nonlinear manifolds (cf. Fig. 6). The manifold S_q is approximated linearly by its tangent hyperplane at a point $\boldsymbol{q}$:

$$S_q \simeq \boldsymbol{q} + \sum_{i=1}^{r} \alpha_i^q \boldsymbol{t}_i^q = \boldsymbol{q} + \mathbf{T}_q \boldsymbol{\alpha}_q, \tag{35}$$

where $\boldsymbol{t}_i^q$ is the ith d-dimensional *tangent vector* (TV) that spans the r-dimensional tangent hyperplane (i.e., the number of considered geometric transformations is r) at a point $\boldsymbol{q}$ and the $\boldsymbol{\alpha}_i^q$ is its corresponding parameter. The notations $\mathbf{T}_q$ and $\boldsymbol{\alpha}_q$ denote $\mathbf{T}_q = (\boldsymbol{t}_1^q \dots \boldsymbol{t}_r^q)$ and $\boldsymbol{\alpha}_q = (\alpha_1^q \dots \alpha_r^q)^\top$, respectively.

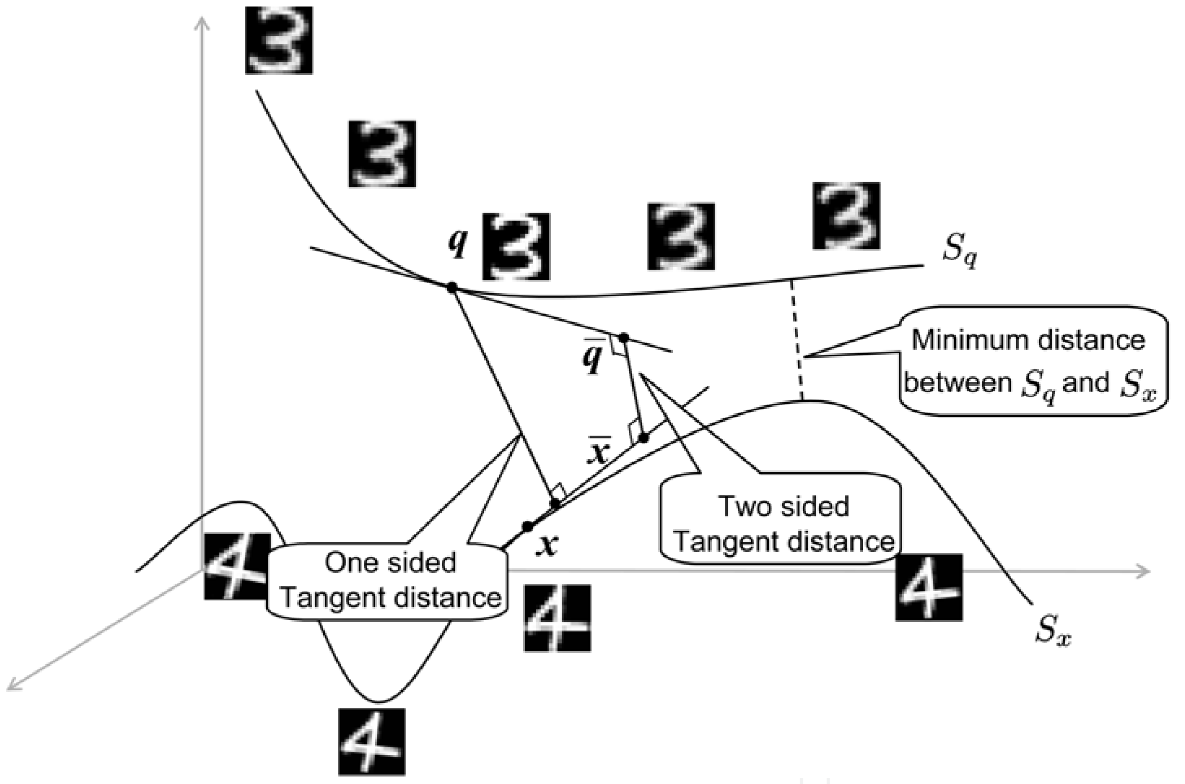
$\boldsymbol{q}$
S_q
$\bar{\boldsymbol{q}}$
$\bar{\boldsymbol{x}}$
$\boldsymbol{x}$
Minimum distance between S_q and S_x
Two sided Tangent distance
One sided Tangent distance
S_x

Fig. 6. Illustration of Euclidean distance and tangent distance between $\boldsymbol{q}$ and $\boldsymbol{x}$. Black dots denote the transformed-images on tangent hyperplanes that minimize 2S-TD.

For approximating S_q, we need to calculate TVs in advance by using finite difference. For instance, the seven TVs for the image depicted in Fig. 5 are shown in Fig. 7. These TVs are derived from the Lie group theory (thickness deformation is an exceptional case), so we can deal with seven geometric transformations (cf. Simard et al., 2001 for more details). By using these TVs, geometric transformations of $\boldsymbol{q}$ can be approximated by a linear combination of the original image $\boldsymbol{q}$ and its TVs. For example, the linear combinations with different amounts of α of the TV for rotation are shown in the bottom in Fig. 5.

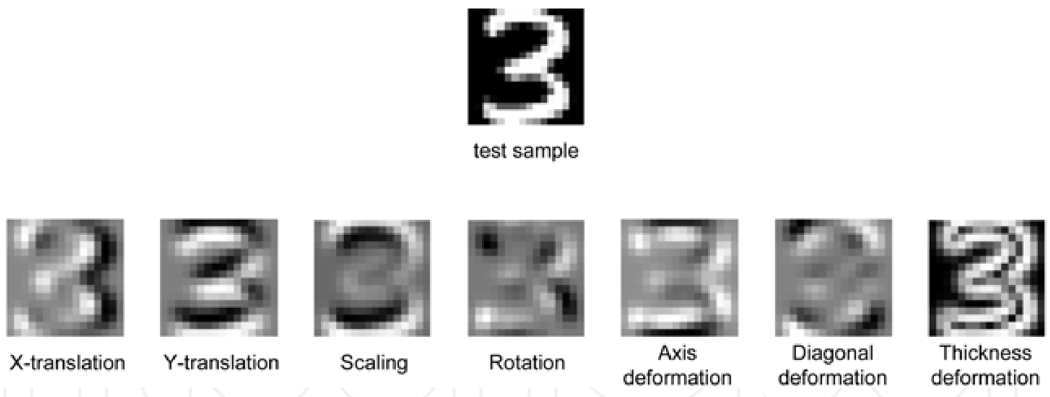

Fig. 7. Tangent vectors $\boldsymbol{t}_i$ for the image depicted in Fig. 3. Fromleft to right, they correspond to x-translation, y-translation, scaling, rotation, axis deformation, diagonal deformation and thickness deformation, respectively.

When measuring the distance between two points on tangent planes, we can use the following distance called *two sided TD* (2S-TD):

$$d_{2S}(\boldsymbol{q}, \boldsymbol{x}) = \|\boldsymbol{q} + \mathbf{T}_q\boldsymbol{\alpha}_q - (\boldsymbol{x} + \mathbf{T}_x\boldsymbol{\alpha}_x)\|^2 . \tag{36}$$

The above distance is the same as 2S-MM, so the solutions of $\boldsymbol{\alpha}_q$ and $\boldsymbol{\alpha}_x$ can be given by using Eq. (11) and Eq. (12). Experimental results on handwritten digit recognition showed that kNN with TD achieves higher accuracy than the use of Euclidean distance (Simard et al., 1993).
Next, a combination of manifold matching and TD for handwritten digit classification is introduced. In manifold matching, we uncritically use a square error between a test sample and training manifolds, so there is a possibility that manifold matching classifies a test sample by using the training samples that are not similar to the test sample. On the other hand, Simard *et al.* investigated the performance of TD using kNN, but the recognition rate of kNN deteriorates when the dimensionality of feature vectors is large. Hence, manifold matching and TD are combined to overcome each of the difficulty. Here, we use the k-closest neighbors to a test sample for manifold matching for achieving high accuracy, thus the algorithm of the combination method is described as follows:

Step1: Find k-closest training samples $\boldsymbol{x}_1^j$, ..., $\boldsymbol{x}_k^j$ to a test sample from class j according to d_{2S}.
Step2: Store the geometric transformed images of the k-closest neighbors existing on their tangent planes as $\mathbf{X}_j = (\bar{\boldsymbol{x}}_1^j|\cdots|\bar{\boldsymbol{x}}_k^j)$, where $\bar{\boldsymbol{x}}_i^j$ is calculated using the optimal weight $\boldsymbol{\alpha}_{x_i}^j$ as follows:

$$\bar{\boldsymbol{x}}_i^j = \boldsymbol{x}_i^j + \mathbf{T}_{x_i}^j\boldsymbol{\alpha}_{x_i}^j \ (i = 1, ..., k). \tag{37}$$

Step3: Also store the k geometric transformed images of the test sample used for selecting the k-closest neighbors $\boldsymbol{x}_i^j$ using 2S-TD as $\mathbf{Q}$ = ($\bar{q}_1 | \dots | \bar{q}_k$), where $\bar{\boldsymbol{q}}_i$ is calculated using the optimal weight $\boldsymbol{\alpha}_i^j$ as follows:

$$\bar{\boldsymbol{q}}_i^j = \boldsymbol{q} + \mathbf{T}_q\boldsymbol{\alpha}_i^j \ (i = 1, ..., k). \tag{38}$$

Step4: Classify $\mathbf{Q}$ with 2S-MM using $\mathbf{X}_j$.

The two approaches described in this section can improve accuracy of manifold matching easily. However, classification cost and memory requirement of them tend to be large. This fact is showed by experiments.

4. Learning rules for manifold matching

For reducing memory requirement and classification cost without deterioration of accuracy, several schemes such as learning vector quantization (Kohonen, 1995; Sato & Yamada, 1995) were proposed in the past. In those schemes, vectors called codebooks are trained by a steepest descent method that minimizes a cost function defined with a training error criterion. However, they were not designed for manifold-based matching. In this section, we adopt *generalized learning vector quantization* (GLVQ) (Sato & Yamada, 1995) to manifold matching for reducing memory requirement and classification cost as small as possible.

Let us consider that we apply GLVQ to 1S-MM. Given a labelled sample $\boldsymbol{q} \in \mathbb{R}^d$ for training (not a test sample), then measure a distance between $\boldsymbol{q}$ and a training manifold of class j by $d_j = \|\boldsymbol{q} - \mathbf{X}_j\boldsymbol{b}_j\|^2$ using the optimal weights obtained with Eq. (4). Let $\mathbf{X}_1 \in \mathbb{R}^{d\times n_1}$ be the set of codebooks belonging to the same class as $\boldsymbol{q}$. In contrast, let $\mathbf{X}_2 \in \mathbb{R}^{d\times n_2}$ be the set of codebooks belonging to the nearest different class from $\boldsymbol{q}$. Let us consider the relative distance difference $\mu(\boldsymbol{q})$ defined as follows:

$$\mu(\boldsymbol{q}) = \frac{d_1 - d_2}{d_1 + d_2}, \tag{39}$$

where d_1 and d_2 represent distances from $\boldsymbol{q}$ to $\mathbf{X}_1\boldsymbol{b}_1$ and $\mathbf{X}_2\boldsymbol{b}_2$, respectively. The above $\mu(\boldsymbol{q})$ satisfies $-1 < \mu(\boldsymbol{q}) < 1$. If $\mu(\boldsymbol{q})$ is negative, $\boldsymbol{q}$ is classified correctly; otherwise, $\boldsymbol{q}$ is misclassified. For improving accuracy, we should minimize the following cost function:

$$S = \sum_{i=1}^{N} f(\mu(\boldsymbol{q}_i)), \tag{40}$$

where N is the total number of labelled samples for training, and $f(\mu)$ is a monotonically increasing function. To minimize S, a steepest descent method with a small positive constant ϵ $(0 < \epsilon < 1)$ is adopted to each $\mathbf{X}_j$:

$$\mathbf{X}_j \leftarrow \mathbf{X}_j - \epsilon \frac{\partial S}{\partial \mathbf{X}_j}, \ (j = 1, 2). \tag{41}$$

Now $\partial S / \partial \mathbf{X}_j$ is derived as

$$\begin{aligned} \frac{\partial S}{\partial \mathbf{X}_j} &= \frac{\partial S}{\partial \mu}\frac{\partial \mu}{\partial d_j}\frac{\partial d_j}{\partial \mathbf{X}_j} \\ &= (-1)^j \frac{\partial f}{\partial \mu}\frac{4d_{3-j}}{(d_1+d_2)^2}(\boldsymbol{q} - \mathbf{X}_j\boldsymbol{b}_j)\boldsymbol{b}_j^\top \ (j = 1, 2). \end{aligned} \tag{42}$$

Consequently, the learning rule can be written as follows:

$$\mathbf{X}_j \leftarrow \mathbf{X}_j + \delta_j(\boldsymbol{q} - \mathbf{X}_j\boldsymbol{b}_j)\boldsymbol{b}_j^\top \ (j = 1, 2), \tag{43}$$

where $\delta_j = (-1)^j \epsilon \frac{\partial f}{\partial \mu} \frac{d_{3-j}}{(d_1+d_2)}$ $(j = 1, 2)$. If we use $d_j = \|\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)\|^2$ as the distance, the learning rule becomes

$$\begin{aligned} \boldsymbol{m}_j &\leftarrow \boldsymbol{m}_j - \delta_j(\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)), \\ \bar{\mathbf{X}}_j &\leftarrow \bar{\mathbf{X}}_j - \delta_j(\boldsymbol{q} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j))\boldsymbol{b}_j^\top . \end{aligned} \tag{44}$$

Similarly, we can apply a learning rule to 2S-MM. Suppose that a labelled manifold for training is given by the set of m vectors $\mathbf{Q} = (\boldsymbol{q}_1 | \boldsymbol{q}_2 | \dots | \boldsymbol{q}_m)$ (not a test manifold). Given this $\mathbf{Q}$, a distance between $\mathbf{Q}$ and $\mathbf{X}_j$ is measured as $d(\mathbf{Q}, \mathbf{X}_j) = \|\boldsymbol{m}_q + \bar{\mathbf{Q}}\boldsymbol{a} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)\|^2$ using the optimal weights obtained with Eq. (11) and Eq. (12). Let $\mathbf{X}_1$ be the set of codebooks belonging to the same class as $\mathbf{Q}$. In contrast, let $\mathbf{X}_2$ be the set of codebooks belonging to the nearest different class from $\mathbf{Q}$. By applying the same manner mentioned above to 2S-MM, the learning rule can be derive as follows:

$$\begin{aligned} \boldsymbol{m}_j &\leftarrow \boldsymbol{m}_j - \delta_j(\boldsymbol{m}_q + \bar{\mathbf{Q}}\boldsymbol{a} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j)), \\ \bar{\mathbf{X}}_j &\leftarrow \bar{\mathbf{X}}_j - \delta_j(\boldsymbol{m}_q + \bar{\mathbf{Q}}\boldsymbol{a} - (\boldsymbol{m}_j + \bar{\mathbf{X}}_j \boldsymbol{b}_j))\boldsymbol{b}_j^\top , \end{aligned} \tag{45}$$

where $\delta_j = (-1)^j \epsilon \frac{\partial f}{\partial \mu} \frac{d(\mathbf{Q}, \mathbf{X}_{3-j})}{(d(\mathbf{Q}, \mathbf{X}_1) + d(\mathbf{Q}, \mathbf{X}_2))}$ $(j = 1, 2)$.

In the above learning rules, we change $d_j/(d_1 + d_2)^2$ into $d_j/(d_1 + d_2)$ for setting ϵ easily. However, this change dose not affect the convergence condition (Sato & Yamada, 1995). As the monotonically increasing function, a sigmoid function $f(\mu, t) = 1/(1 - e^{-\mu t})$ is often used in experiments, where t is learning time. Hence, we use $f(\mu, t)\{1 - f(\mu, t)\}$ as $\partial f / \partial \mu$ in practice.

classifier	parameter (s)
one sided manifold matching (1S-MM)	# dimension of manifolds r'
two sided manifold matching (2S-MM)	# dimension of training manifolds r'
Local Subspace Classifier (LSC)	# neighbor training samples k
Kernelized 1S-MM (K1S-MM, cf. Section 3)	# dimension of training manifolds r' in $\mathcal{F}$ parameters for kernel function
2S-MM with 2S-TD (2S-MM with 2S-TD, cf. Section 3)	# neighbors k
kNN with Euclidean Distance (kNN)	# neighbor training samples k
kNN with 2S-TD (kNN+2S-TD)	# neighbor training samples k
Support Vector Machine (SVM)	Soft margin constant and parameters for kernel function

Table 1. Summary of classifiers used in experiments

In this case, $\partial f / \partial \mu$ has a single peak at $\mu = 0$, and the peak width becomes narrower as t increases. After the above training, $\boldsymbol{q}$ and $\mathbf{Q}$ are classified by the classification rules Eq. (8) and Eq. (15) respectively using trained codebooks. In the learning rule of Eq. (43), if the all elements of $\boldsymbol{b}_j$ are equal to $1/\sqrt{n_j}$, this rule is equivalent to GLVQ. Hence, Eq. (43) can be regarded as a natural extension of GLVQ. In addition, if $\mathbf{X}_j$ is defined by k-closest training samples to $\boldsymbol{q}$, the rule can be regarded as a learning rule for LSC (Hotta, 2008b).

5. Experiments

For comparison, experimental results on handwritten digit datasets MNIST (LeCun et al., 1998) and USPS (LeCun et al., 1989) are shown in this section. The MNIST dataset consists of

60,000 training and 10,000 test images. In experiments, the intensity of each 28 $\times$ 28 pixels image was reversed to represent the background of images with black. The USPS dataset consists of 7,291 training and 2,007 test images. The size of images of USPS is 16 $\times$ 16 pixels. The number of training samples of USPS is fewer than that of MNIST, so this dataset is more difficult to recognize than MNIST. In experiments, intensities of images were directly used for classification.

The classifiers used in experiments and their parameters are summarized in Table 1. In 1SMM, a training manifold of each class was formed by its centroid and r' eigenvectors corresponding to the r' largest eigenvalues obtained with PCA. In LSC, k-closest training samples to a test sample were selected from each class, and they were used as $\mathbf{X}_j$. In 2S-MM, a test manifold was spanned by an original test image ($\boldsymbol{m}_q$) and its seven tangent vectors ($\bar{\mathbf{X}}_j$) such as shown in Fig. 7. In contrast, a training manifold of each class was formed by using PCA. In K1S-MM, kernel PCA with TDK (cf. Eq. 34) was used for representing training manifolds in $\mathcal{F}$. All methods were implemented with MATLAB on a standard PC that has Pentium 1.86GHz CPU and 2GB RAM. In implementation, program performance optimization techniques such as mex files were not used. For SVM, the SVM package called LIBSVM (Chang & Lin, 2001) was used for experiments.

5.1 Test error rate, classification time, and memory size

In the first experiment, test error rates, classification time per test sample, and a memory size of each classifier were evaluated. Here, a memory size means the size of a matrix for storing training samples (manifolds) for classification. The parameters of individual classifiers were tuned on a separate validation set (50000 training samples and 10000 validation samples for MNIST; meanwhile, 5000 training samples and 2000 validation samples for USPS).

Table 2 and Table 3 show results on MNIST and USPS, respectively. Due to out of memory, the results of SVM and K1S-MM in MNIST were not obtained with my PC. Hence, the result of SVM was referred to (Decoste & Schölkopf, 2002). As shown in Table 2, 2S-MM outperformed 1S-MM but the error rate of it was higher than those of other manifold matching such as LSC. However, classification cost of the classifiers other than 1S-MM and 2S-MM was very high. Similar results can be found in the results of USPS. However, the error rate of 2S-MM was lower than that of SVM in USPS. In addition, manifold matching using accuracy improvement described in section 3 outperformed other classifiers. However, classification cost and memory requirement of them were very high.

classifier	test error [%]	time [sec.]	memory size for classification
1S-MM ($r' = 30$)	4.3	0.003	784 $\times$ 310
2S-MM ($r' = 20$)	3.1	0.03	784 $\times$ 210
LSC ($k = 26$)	1.4	1	784 $\times$ 60000
K1S-MM	–	–	out of memory
2S-MM+2S-TD ($k = 23$)	0.7	1.9	784 $\times$ 60000
kNN ($k = 3$)	2.9	0.3	784 $\times$ 60000
kNN+2S-TD ($k = 1$)	1.4	0.6	784 $\times$ 60000
SVM (Polynomial kernel) with 1-pixel translation	1.2	–	784$\times$ ca. 30000 (# support vectors)

Table 2. Test error rates, classification time per test sample, and memory size on MNIST.

classifier	test error [%]	time [sec.]	memory size for classification
1S-MM ($r' = 30$)	5.1	0.001	256×310
2S-MM ($r' = 20$)	4.2	0.01	256×210
LSC (k=11)	3.9	0.05	256×7291
K1S-MM ($r' = 10$)	3.3	1.6	$256 \times 7291 + 729 \times 10 \times 10$
2S-MM+2S-TD ($k = 7$)	2.2	1.2	256×7291
kNN ($k = 1$)	5.3	0.2	256×7291
kNN+2S-TD	2.4	0.13	256×7291
SVM (RBF kernel)	4.6	0.005	256×3220 (# support vectors)

Table 3. Test error rates, classification time per test sample, and memory size on USPS.

5.2 Effectiveness of learning

Next, the effectiveness of learning for manifold matching was evaluated by experiments. In general, handwritten patterns include various geometric transformations such as rotation, so it is difficult to reduce memory sizes without accuracy deterioration. In this section, learning for 1S-MM using Eq. (44) is called *learning 1S-MM* (L1S-MM). The initial training manifolds were formed by PCA as shown in the left side of Fig. 8. Similarly, learning for 2S-MM using Eq. (45) is called *learning 2S-MM* (L2S-MM). The initial training manifolds were also determined by PCA. In contrast, a manifold for training and a test manifold were spanned by an original image and its seven tangent vectors. The numbers of dimension for training manifolds of L1S-MM and L2S-MM were the same as those of 1S-MM and 2S-MM in the previous experiments, respectively. Hence, their classification time and memory size did not change. Learning rate ϵ was set to $\epsilon = 10^{-7}$ empirically. Batch type learning was applied to L1S-MM and L2S-MM to remove the effect of the order which training vectors or manifolds were presented to them. The right side of Fig. 8 shows the trained bases of each class using MNIST. As shown in this, learning enhanced the difference of patterns between similar classes.

method	test error [%]	training time [s]	memory size for training
L1S-MM ($r' = 30$)	2.3	109680	784×60000
L2S-MM ($r' = 20$)	1.8	71909	784×60000
GLVQ	10.4	536	784×60000
SVM	1.2	–	60000×60000

Table 4. Test error rates, training time, and memory size for training on MNIST.

method	test error [%]	training time [s]	memory size for training
L1S-MM ($r' = 30$)	4.9	2789	256×7291
L2S-MM ($r' = 20$)	3.7	3437	256×7291
GLVQ	8.7	37	256×7291
SVM	4.6	34.9	7291×7291

Table 5. Test error rate and training time on USPS.

Figure 9 shows training error rates of L1S-MM and L2S-MM in MNIST with respect to the number of iteration. As shown in this figure, the training error rates decreased with time. This means that the learning rules described in this chapter converge stably based on the convergence property of GLVQ. Also 50 iteration was enough for learning, so the maximum number of iteration was fixed to 50 for experiments. Table 4 and Table 5 show test error rates, training time, and memory size for training on MNIST and USPS, respectively. For comparison, the results obtained with GLVQ were also shown. As shown in these tables, accuracy of 1S-MM and 2S-MM was improved satisfactorily by learning without increasing of classification time and memory sizes. The right side of Fig. 8 shows the bases obtained with L2S-MM on MNIST. As shown in this, the learning rule enhanced the difference of patterns between similar classes. It can be considered that this phenomenon helped to improve accuracy. However, training cost for manifold matching was very high by comparison to those of GLVQ and SVM.

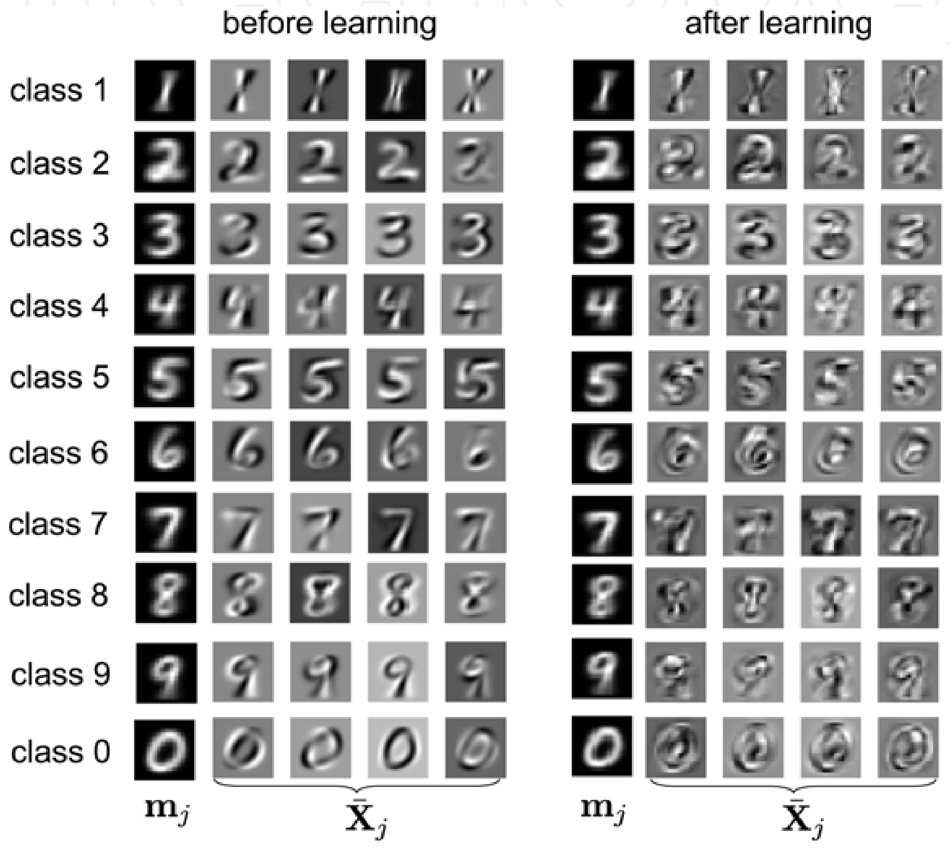

Fig. 8. Left: Origins ($\boldsymbol{m}_j$) and orthonormal bases $\mathbf{X}_j$ of individual classes obtained with PCA (initial components for training manifolds). Right: Origins and bases obtained with L2S-MM (components for training manifolds obtained with learning).

6. Conclusion

In this chapter manifold matching for high-dimensional pattern classification was described. The topics described in this chapter were summarized as follows:
- The meaning and effectiveness of manifold matching
- The similarity between various classifiers from the point of view of manifold matching
- Accuracy improvement for manifold matching
- Learning rules for manifold matching

Experimental results on handwritten digit datasets showed that manifold matching achieved lower error rates than other classifiers such as SVM. In addition, learning improved accuracy and reduced memory requirement of manifold-based classifiers.

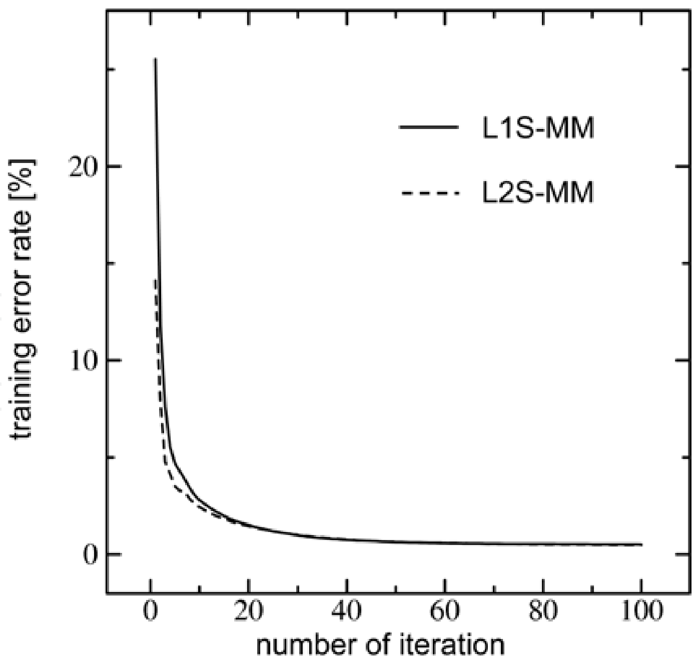

Fig. 9. Training error rates with respect to the number of iteration.

The advantages of manifold matching are summarized as follows:
- Wide range of application (e.g., movie classification)
- Small memory requirement
- We can adjust memory size easily (impossible for SVM)
- Suitable for multi-class classification (not a binary classifier)

However, training cost for manifold matching is high. Future work will be dedicated to speed up a training phase and improve accuracy using prior knowledge.

7. References

Chang, C.C. and Lin, C. J. (2001), LIBSVM: A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Chen, J.H., Yeh, S.L., and Chen, C.S. (2004), Inter-subspace distance: A new method for face recognition withmultiple samples," *The 17th Int'l Conf. on Pattern Recognition ICPR (2004)*, Vol. 3, pp. 140–143

Duda, R.O., Hart, P.E., & Stork, D.G. (2001). Pattern classification. 2nd edition, John Wiley & Sons.

Decoste, D. and Sch¨olkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, Vol. 46, pp. 161–190

Haasdonk, B. & Keysers, D. (2002), Tangent distance kernels for support vector machines. *The 16th Int'l Conf. on Pattern Recognition ICPR (2002)*, Vol. 2, pp. 864–868

Hotta, S. (2008a). Local subspace classifier with transform-invariance for image classification. *IEICE Trans. on Info. & Sys.*, Vol. E91-D, No. 6, pp. 1756–1763

Hotta, S. (2008b). Learning vector quantization with local subspace classifier. *The 19th Int'l Conf. on Pattern Recognition ICPR (2008)*, to appear

Ikeda, K., Tanaka, H., and Motooka, T. (1983). Projection distance method for recognition of hand-written characters. J. IPS. Japan, Vol. 24, No. 1, pp. 106–112

Kohonen., T. (1995). Self-Organizingmaps. 2nd Ed., Springer-Verlag, Heidelberg

Laaksonen, J. (1997). Subspace classifiers in recognition of handwritten digits. *PhD thesis, Helsinki University of Technology*

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., & Jackel, L.D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, Vol. 1, No. 4, pp. 541–551

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. of the IEEE*, Vol. 86, No. 11, pp. 2278-2324

Maeda, E. and Murase, H. (1999). Multi-category classification by kernel based nonlinear subspace method. *Proc. of ICASSP*, Vol. 2, pp. 1025–1028

Mitani, Y. & Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Patt. Recog. Lett.*, Vol. 27, No. 10, pp. 1151–1159

Roweis, S.T. & Saul, L.K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, Vol. 290–5500, pp. 2323–2326

Sato,A. and Yamada, K. (1995). Generalized learning vector quantization. *Prop. of NIPS*,Vol. 7, pp. 423–429

Schölkopf, B., Smola, A.J., and M¨uller, K.R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, Vol. 10, pp. 1299–1319

Schölkopf, B. and Smola, A.J. (2002). Learning with kernels. *MIT press*

Simard, P.Y., LeCun, Y., & Denker, J.S. (1993). Efficient pattern recognition using a new transformation distance. *Neural Information Processing Systems*, No. 5, pp. 50–58

Simard, P.Y., LeCun, Y., Denker, J.S., & Victorri, B. (2001). Transformation invariance in pattern recognition – tangent distance and tangent propagation. *Int'l J. of Imaging Systems and Technology*, Vol. 11, No 3

Vincent, P. and Bengio, Y. (2002). K-local hyperplane and convex distance nearest neighbor algorithms. *Neural Information Processing Systems*

Pattern Recognition Techniques, Technology and Applications
Edited by Peng-Yeng Yin

ISBN 978-953-7619-24-4
Hard cover, 626 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

A wealth of advanced pattern recognition algorithms are emerging from the interdiscipline between technologies of effective visual features and the human-brain cognition process. Effective visual features are made possible through the rapid developments in appropriate sensor equipments, novel filter designs, and viable information processing architectures. While the understanding of human-brain cognition process broadens the way in which the computer can perform pattern recognition tasks. The present book is intended to collect representative researches around the globe focusing on low-level vision, filter design, features and image descriptors, data mining and analysis, and biologically inspired algorithms. The 27 chapters coved in this book disclose recent advances and new ideas in promoting the techniques, technology and applications of pattern recognition.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Seiji Hotta (2008). Manifold Matching for High-Dimensional Pattern Recognition, Pattern Recognition Techniques, Technology and Applications, Peng-Yeng Yin (Ed.), ISBN: 978-953-7619-24-4, InTech, Available from: http://www.intechopen.com/books/pattern_recognition_techniques_technology_and_applications/manifold_matching_for_high-dimensional_pattern_recognition