PAPER

Protein Fold Classification Using Large Margin Combination of Distance Metrics

Chendra Hadi SURYANTO[†a)], *Nonmember*, **Kazuhiro FUKUI**[†b)], *Fellow*, *and* **Hideitsu HINO**[†c)], *Member*

SUMMARY Many methods have been proposed for measuring the structural similarity between two protein folds. However, it is difficult to select one best method from them for the classification task, as each method has its own strength and weakness. Intuitively, combining multiple methods is one solution to get the optimal classification results. In this paper, by generalizing the concept of the large margin nearest neighbor (LMNN), a method for combining multiple distance metrics from different types of protein structure comparison methods for protein fold classification task is proposed. While LMNN is limited to Mahalanobis-based distance metric learning from a set of feature vectors of training data, the proposed method learns an optimal combination of metrics from a set of distance metrics by minimizing the distances between intra-class data and enlarging the distances of different classes' data. The main advantage of the proposed method is the capability in finding an optimal weight coefficient for combination of many metrics, possibly including poor metrics, avoiding the difficulties in selecting which metrics to be included for the combination. The effectiveness of the proposed method is demonstrated on classification experiments using two public protein datasets, namely, Ding Dubchak dataset and ENZYMES dataset.

key words: *protein fold classification, distance metrics combination, large margin nearest neighbor, kernel learning, optimization*

1. Introduction

In structural biology, comparing the similarity between protein fold structures is an important task, because proteins with similar fold structure share the same biological functions. Furthermore, the evolutionary relationship between proteins can be inferred by structural comparison [1]. Since the protein structures contain complex sub-structures with large variation in the composition, comparing them is not straightforward. Many methods have been proposed for comparing the similarity or dissimilarity between protein fold structures [2]–[7]. Most of them require alignment according to the 3D protein backbone structures composed of the folded amino acids sequences, followed by computation of the root mean square deviation (RMSD) of the superimposed alpha-carbon atoms of the amino acid molecules. However, defining an optimal alignment between the protein structure is a difficult problem, especially when the structures are very different. Furthermore, RMSD, which is computed on the basis of the aligned and superposed structures, is not always proportional to the number of aligned parts. Consequently, different similarity measures have also been proposed to complement RMSD, such as Z-Score [2] and TM-Score [5]. The task of protein structural comparison can also be framed as an image set comparison to avoid the difficulties of alignment, by generating multiple views of protein visualization images by using 3D graphics molecular software. The similarity between two protein structures is then defined by the canonical angles between the corresponding subspaces that are generated from the image sets [7].

It is difficult to select one best method from a number of the protein structural comparison methods. For example, view-based methods are superior for classifying protein that have large intra-variation but have problems when classifying proteins with very similar structures. In contrast, alignment-based methods are superior for differentiating very similar protein structures but inferior for comparing very different structures [7]. This brought us to the idea of combining multiple methods to optimize the classification performance.

Many fusion and metric learning algorithms have been proposed for cases in which feature vector representation is available [8]–[14]. Unfortunately, for the case of protein structural comparison, feature vector representation is not always available. For example, alignment methods [2]–[6] only produce either distances or similarity values with the aligned sub-structures. Graph-based [15] and subspace-based representation [7] are another two examples of nontrivial features representation for protein structures, from which only similarity metrics can be obtained.

In this paper, we propose a combination of multiple distance metrics for protein fold classification, illustrated in Fig. 1. Given S distance measures $\{d_i\}_{i=1}^{S}$ (assumed to be obtained by using multiple techniques), our task is to learn an optimal weight coefficient $\boldsymbol{w}^* \in \mathbb{R}^S$ by minimizing the distance between samples that belong to the same class and enlarging the distance of samples that belong to other classes only using the sets of the distance matrices. This concept is similar to that of the large margin nearest neighbor (LMNN) [10] and closely related to the support vector machine (SVM), wherein the separation between classes is optimized according to convex optimization with a hinge loss function. However, unlike SVM, which is theoretically designed for two-class classification tasks, our method and the conventional LMNN are naturally applicable to multi-class problems. While the original LMNN learns a Mahalanobis-

Manuscript received July 24, 2015.
Manuscript revised October 28, 2015.
Manuscript publicized December 14, 2015.

[†]The authors are with the Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba-shi, 305–8577 Japan.
a) E-mail: chendra@cvlab.cs.tsukuba.ac.jp
b) E-mail: kfukui@cs.tsukuba.ac.jp
c) E-mail: hinohide@cs.tsukuba.ac.jp

DOI: 10.1587/transinf.2015EDP7294

Copyright © 2016 The Institute of Electronics, Information and Communication Engineers

based distance metric from a set of feature vectors of training data, our proposed method learns an optimal weight coefficients for combination of the distance metrics from training data. The final distance, termed as *overall distance*, is then defined as the linear combination of the distances that can be used for any distance-based classification such as k-nearest neighbor classification.

The main advantage of our proposed method is the capability in finding an optimal combination for many distance metrics, possibly including poor metrics. Accordingly, when there are a number of distance measures available, we can eliminate the difficulties in selecting the appropriate measures for the combination. In practice, this property is important because the distance measure that can perform the best on a certain data is commonly not known beforehand.

We demonstrate the effectiveness of the proposed method through classification experiments on 27 fold classes of proteins using Ding Dubchak dataset [16], [17] and six classes of protein enzymes using ENZYMES dataset [15]. Our proposed method is closely related to multiple kernel learning (MKL) [18]–[20], where MKL combines multiple kernel matrices instead of distances. Therefore, we compared the performance of our proposed method with generalized MKL (GMKL) [19] in addition to naïve averaging and voting.

We regard our current paper as an extension of our previous work [21] that introduce three loss functions for obtaining the optimal weights for the distance metrics combination. The contributions of the present work are as follows:

- generalization of the concept of the LMNN to learn optimal weight coefficients for a combination of distance metrics, as in [21];
- introduction of three loss functions to the problem formulation for combining distance metrics using hinge loss, smooth hinge loss, and logistic loss;
- comprehensive experiments on protein fold and enzymes classification, including performance comparison with naïve methods (averaging and voting) and GMKL [19] using the public protein dataset of Ding Dubchak [16], [17] and ENZYMES [15].

The rest of this paper is organized as follows. Section 2 provides an overview of LMNN. Section 3 describes how distance metrics are combined according to LMNN and presents the formulation of LMNN using three types of loss functions. Section 4 provides the experimental results and discussions. Finally, Sect. 5 is devoted to the conclusion.

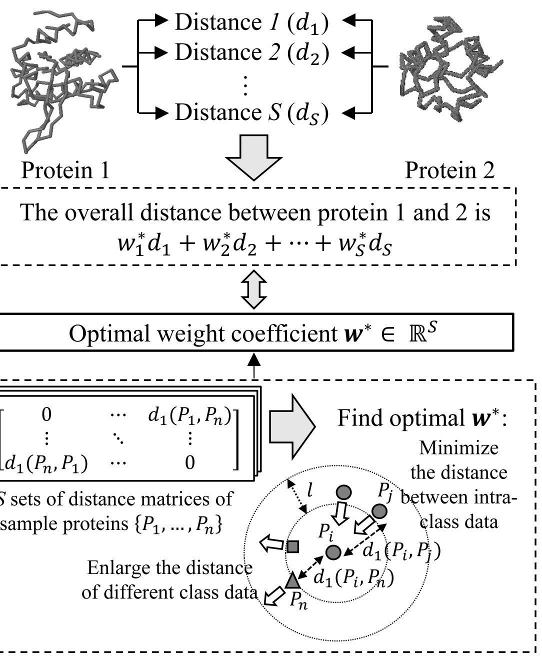
Distance 1 (d_1)
Distance 2 (d_2)
Distance S (d_S)
Protein 1
Protein 2
The overall distance between protein 1 and 2 is
$w_1^* d_1 + w_2^* d_2 + \cdots + w_S^* d_S$
Optimal weight coefficient $\boldsymbol{w}^* \in \mathbb{R}^S$
$d_1(P_1, P_n)$
$d_1(P_n, P_1)$
S sets of distance matrices of sample proteins $\{P_1, \ldots, P_n\}$
Find optimal $\boldsymbol{w}^*$:
Minimize the distance between intra-class data
Enlarge the distance of different class data
l
P_i
P_j
P_n
$d_1(P_i, P_j)$
$d_1(P_i, P_n)$

Fig. 1 Basic idea of the proposed method. Let $d_s(P_i, P_j)$ be the distance between two sample proteins P_i and P_j computed using method s. The objective is to optimize the weight combination of multiple distance measures $\{1, \ldots, S\}$, by minimizing the distances between each protein and its intra-class proteins and enlarging the distances of different class proteins under margin l, using only the sets of the distance matrices.

2. Large Margin Nearest Neighbor (LMNN)

Conventional metric learning algorithms search for a transformation of feature vectors of the sample data to obtain an optimal metric, used for task that depends on distances, such as classification using nearest-neighbor method [22]. Many metric learning methods have been proposed [8]–[14]. In the following, we review LMNN [10], which is the basis of our proposed method.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ be a training set, where $x_i \in \mathbb{R}^d$ is a data point (a feature vector) and $y_i \in \{1, 2, \ldots, C\}$ is the class label of x_i. Then, LMNN searches for a linear transformation $\mathbf{W} : \mathbb{R}^d \to \mathbb{R}^f$ ($f \leq d$), ensuring that the data points in the same class are brought closer to each other and that the margins between different classes are made larger, as shown in Fig. 2. The cost function to be minimized consists of two terms as follows:

$$F(\mathbf{W}) = \sum_{i=1}^{n} \sum_{j:y_j=y_i} \|\mathbf{W}(x_i - x_j)\|^2 + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \left[1 + \|\mathbf{W}(x_i - x_j)\|^2 - \|\mathbf{W}(x_i - x_h)\|^2\right]_+ , \tag{1}$$

where $\mu > 0$ is a balancing parameter between the two

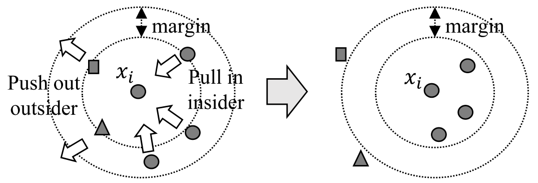

Fig. 2 Illustration of how LMNN works. The small circle in the center represents one sample data point x_i. LMNN learns a transformation matrix from a set of feature vectors of samples to pull in data points of the same class and push out data points of different classes with a margin of one unit distance.

terms. The first term imposes a cost when the distances between the data points in the same class are large and the second term imposes a cost when the distances between the data points in different classes are smaller than the distances to data points within the class. Here, $[\cdot]_+$ is the hinge loss defined as

$$[x]_+ = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

To optimize $F(\mathbf{W})$, the transformation matrix $\mathbf{W}$ is parameterized as the Mahalanobis-based distance metric with a covariance matrix $\mathbf{C} = \mathbf{W}^\top \mathbf{W}$, so that Eq. (1) becomes

$$F(\mathbf{C}) = \sum_{i=1}^{n} \sum_{j:y_j=y_i} \mathcal{M}(x_i, x_j) + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \left[1 + \mathcal{M}(x_i, x_j) - \mathcal{M}(x_i, x_h)\right]_+, \tag{3}$$

where $\mathcal{M}(x_i, x_j) = (x_i - x_j)^\top \mathbf{C}(x_i - x_j)$. The cost function $F(\mathbf{C})$ can be optimized using semidefinite programming or subgradient descent [10].

3. Extending LMNN for Combining Distance Metrics

While the conventional distance metric learning aims to learn a new distance metric through transformation of *feature vectors*, the goal of our proposed method is to learn an optimal *overall distance* through combination of multiple distance metrics. The motivation is to deal with the protein fold classification task where a number of protein structural dissimilarity measures are defined without feature vector representation in addition to dissimilarity measures defined in vector spaces with feature vectors representation. In the following, we reformulate Eq. (1) to work with distances instead of feature vectors.

Let the distance measure between two data points x_i and x_j computed using a concrete distance function with corresponding weight $\boldsymbol{w}$ be $d(x_i, x_j; \boldsymbol{w})$. The cost function in Eq. (1) is then rewritten as a cost function $J(\boldsymbol{w})$:

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \sum_{j:y_j=y_i} \left[d^2(x_i, x_j; \boldsymbol{w}) + \mu \sum_{h:y_h \neq y_i} [L(i, j, h; \boldsymbol{w})]_+ \right], \tag{4}$$

$$L(i, j, h; \boldsymbol{w}) = l + d^2(x_i, x_j; \boldsymbol{w}) - d^2(x_i, x_h; \boldsymbol{w}). \tag{5}$$

The same as in Eq. (1), $\mu > 0$ is a balancing parameter between the two terms. However, here the margin is set as a tuning parameter $l \geq 0$, while in Eq. (1) the margin is fixed to 1. In the original formulation of LMNN [10], the distance d was parameterized as a Mahalanobis-based distance. On the other hand, we parameterize d as a convex combination of multiple distance measures.

3.1 Combination of Distances

As mentioned in the introduction, methods for comparing protein fold structures use either distances (dissimilarities) or similarity scores to evaluate the similarity between two proteins. Before combination of the distances or similarity scores, these values need to be normalized into a distance measure with range $0 \leq d_{ij} \leq 1$, by redefining $d_{ij} = D_{ij} / \max(\mathbf{D})$, where d_{ij} is the distance between x_i and x_j, and $\mathbf{D}$ is the $n \times n$ distance matrix for n training samples. Likewise, we can convert a similarity measure to a distance measure by letting $d_{ij} = 1 - (u_{ij} / \max(\mathbf{U}))$, where u_{ij} denotes the similarity value between x_i and x_j, and $\mathbf{U}$ is the $n \times n$ similarity matrix for n training samples.

The linear combination of S distance measures is written as follows:

$$d(x_i, x_j; \boldsymbol{w}) = \sum_{s=1}^{S} w_s d_s(x_i, x_j), \tag{6}$$

where $d_s(x_i, x_j)$ are the normalized distance measures between x_i and x_j as obtained from the multiple protein structure comparison methods. To ensure that the convexity and S-simplex constraints are satisfied, the weight for each distance measure $\boldsymbol{w} = (w_1, \ldots, w_S)^\top \in \mathbb{R}^S$ must satisfy

$$\sum_{s=1}^{S} w_s = 1, \; w_s \geq 0. \tag{7}$$

Assuming that each distance $d_s(x_i, x_j)$ satisfies the axiom of distance metric, i.e., non-negativity, coincidence axiom, symmetry, and triangle inequality, Eq. (6) also satisfies the axiom of distance metric.

3.2 Optimization for LMNN

In the following, we formulate the optimization algorithm to obtain the optimal parameter $\boldsymbol{w}^*$. First, Eqs. (4) and (5) are rewritten as

$$J(\boldsymbol{w}) = \boldsymbol{w}^\top \mathbf{M} \boldsymbol{w} + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} [L(i, j, h; \boldsymbol{w})]_+ \tag{8}$$

$$L(i, j, h; \boldsymbol{w}) = l + \boldsymbol{w}^\top \boldsymbol{d}_{ij} \boldsymbol{d}_{ij}^\top \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{d}_{ih} \boldsymbol{d}_{ih}^\top \boldsymbol{w}, \tag{9}$$

where

$$\mathbf{M} = \sum_{i=1}^{n} \sum_{j:y_j=y_i} \boldsymbol{d}_{ij} \boldsymbol{d}_{ij}^\top, \tag{10}$$

$$\boldsymbol{d}_{ij} = (d_1(x_i, x_j), \ldots, d_S(x_i, x_j))^\top \in \mathbb{R}^S, \tag{11}$$

$$d(x_i, x_j; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{d}_{ij}. \tag{12}$$

The first term in Eq. (8) is a smooth quadratic function that can be directly optimized. In contrast, the second term contains hinge loss $[x]_+$, which is not straightforward to optimize since it is not a smooth function. In the following, we first provide an iterative algorithm to optimize Eq. (8) and then present three approaches for solving the second term.

Algorithm 1 Gradient based Algorithm for Minimizing $J(\boldsymbol{w})$

```
Initialize: $\boldsymbol{w}_0 = (1/S, \ldots, 1/S)$, $\mu > 0$, $l > 0$, $\epsilon$ are set to small values, such as $10^{-5}$ or $10^{-6}$, and $\mathbf{M}$ is as in Eq. (10).
repeat
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \left\{ \mathbf{M}\boldsymbol{w} + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \boldsymbol{g}_{ijh} \right\}$
    Simplex projection:
    for $s = 1, \ldots, S$ do
        if $w_s < 0$ then $w_s \leftarrow 0$
    end for
    for $s = 1, \ldots, S$ do
        $w_s \leftarrow \frac{w_s}{\sum_{s=1}^{S} w_s}$
    end for
until converges
Output: optimal coefficient $\boldsymbol{w}^* \leftarrow \boldsymbol{w}$
```

The first approach optimizes hinge loss by using the subgradient algorithm [21]. The second approach adds relaxation to the hinge loss through smoothing the hinge loss. Finally, the third approach replaces hinge loss with logistic loss.

3.2.1 Optimization Algorithm

Algorithm 1 provides an iterative algorithm based on the gradient method for optimizing Eq. (8). In Algorithm 1, $\boldsymbol{g}_{ijh}$ is replaced with either $\boldsymbol{g}_{ijh}^{HL}$, $\boldsymbol{g}_{ijh}^{SHL}$, or $\boldsymbol{g}_{ijh}^{LL}$, which corresponds to the gradient of the second terms when using hinge loss (Sect. 3.2.2), smooth hinge loss (Sect. 3.2.3), or logistic loss (Sect. 3.2.4), respectively.

3.2.2 Hinge Loss Optimization

Hinge loss $[\cdot]_+$ in Eq. (8) is not differentiable at the origin, but it is possible to optimize the hinge loss by using the subgradient method. The subgradient of $[L(i, j, h; \boldsymbol{w})]_+$ is as follows [21]:

$$\mathbb{R}^s \ni \boldsymbol{g}_{ijh}^{HL} = \begin{cases} 2(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}, & \text{if } z > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

where $z = L(i, j, h; \boldsymbol{w})$.

3.2.3 Smooth Hinge Loss Optimization

Smooth hinge loss [23] adds an intermediate criterion to the hinge loss so that it is differentiable everywhere. Smooth hinge loss is defined as follows:

$$h_{smooth}(x) = \begin{cases} x - \frac{1}{2}, & \text{if } x \geq 1, \\ \frac{1}{2}x^2, & \text{if } 0 < x < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

By replacing $[L(i, j, h; \boldsymbol{w})]_+$ with $h_{smooth}(L(i, j, h; \boldsymbol{w}))$, the gradient becomes

$$\mathbb{R}^s \ni \boldsymbol{g}_{ijh}^{SHL} = \begin{cases} 2(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}, & \text{if } z \geq 1, \\ (2(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w})z, & \text{if } 0 < z < 1, \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where $z = L(i, j, h; \boldsymbol{w})$.

3.2.4 Logistic Loss Optimization

Another alternative to using hinge loss is to use logistic loss $\log(1 + e^x)$, which is convex and differentiable everywhere. The cost function $J(\boldsymbol{w})$ in Eq. (8), approximated by logistic loss, is then written as follows:

$$\boldsymbol{w}^\top \mathbf{M}\boldsymbol{w} + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \log(1 + e^z), \tag{16}$$

where $z = L(i, j, h; \boldsymbol{w})$. The gradient of Eq. (16) is given by

$$\mathbb{R}^s \ni \boldsymbol{g}_{ijh}^{LL} = \frac{2e^{l+\boldsymbol{w}^\top(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}}(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}}{1 + e^{l+\boldsymbol{w}^\top(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}}}. \tag{17}$$

3.3 Flow of The Classification Framework

Figure 3 shows the flow of the classification framework.

(1) Training Phase:

Step 1: Given n training proteins and S methods that compute either similarities or distances, compute S $n \times n$ distance matrices in which each element is normalized ($0 \leq d_s(x_i, x_j) \leq 1$).
Step 2: Find the optimal $\boldsymbol{w}^* \in \mathbb{R}^S$ by using Algorithm 1.

(2) Recognition Phase:

Step 1: Given an input protein, compute S distances $(d_1, \ldots, d_S)$ between the input protein and each training protein and normalize the value similarly to that in the training phase.
Step 2: Compute the overall distance $d_{overall}$ by combining the S distances, using $\boldsymbol{w}^*$.
Step 3: Apply k-nearest neighbor to the overall distance $d_{overall}$ to predict the fold class of the input protein.

4. Experiments

To evaluate the validity of the proposed method, we conducted protein fold classification experiments using Ding Dubchak dataset [16], [17] and ENZYMES dataset [15], and compared the results with those from conventional methods based on averaging, voting, and GMKL [19]. The classification performance was evaluated in terms of precision (ratio of true positives to true positives and false positives) and recall (ratio of true positives to true positives and false negatives).

4.1 Experiments on Ding Dubchak Dataset

The Ding Dubchak dataset [16], [17] contains 27 fold classes of 693 proteins, as shown in Table 1. We conducted the classification experiments by combining distance metrics from the extracted features of the amino acid sequences

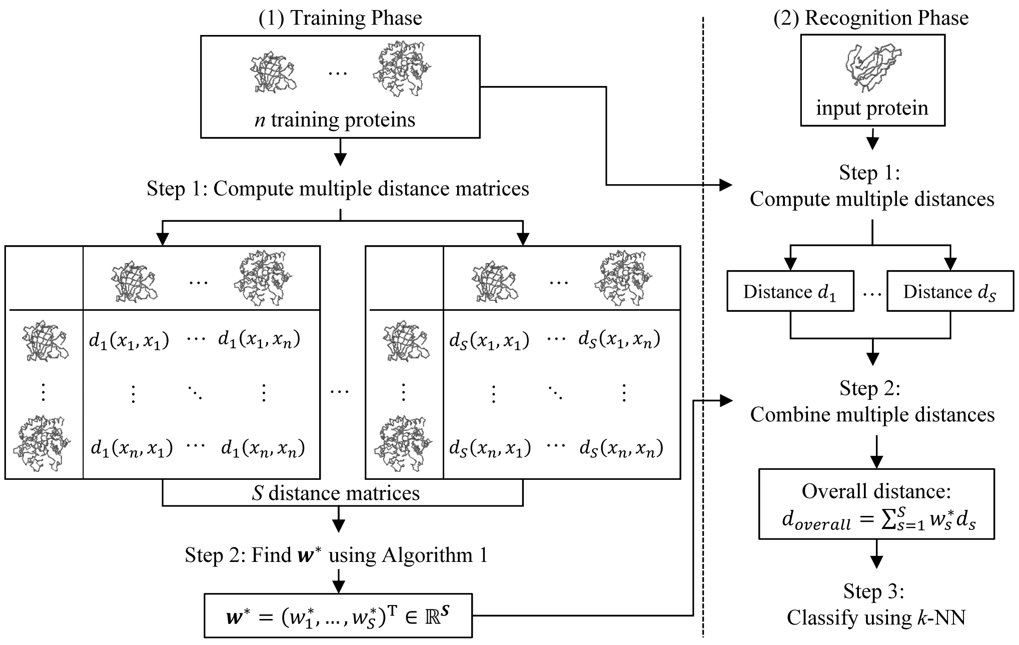
(1) Training Phase
n training proteins
Step 1: Compute multiple distance matrices
$d_1(x_1, x_1)$ ⋯ $d_1(x_1, x_n)$
$d_1(x_n, x_1)$ ⋯ $d_1(x_n, x_n)$
$d_S(x_1, x_1)$ ⋯ $d_S(x_1, x_n)$
$d_S(x_n, x_1)$ ⋯ $d_S(x_n, x_n)$
S distance matrices
Step 2: Find $\boldsymbol{w}^*$ using Algorithm 1
$\boldsymbol{w}^* = (w_1^*, \ldots, w_S^*)^{\mathrm{T}} \in \mathbb{R}^S$
(2) Recognition Phase
input protein
Step 1: Compute multiple distances
Distance d_1
Distance d_S
Step 2: Combine multiple distances
Overall distance: $d_{overall} = \sum_{s=1}^{S} w_s^* d_s$
Step 3: Classify using k-NN

Fig. 3 Flow of the classification framework.

Table 1 27-fold protein compositions used in the experiments.

No.	Fold class	# Proteins
1	Alpha; Globin-like	19
2	Alpha; Cytochrome c	16
3	DNA-binding 3-helical bundle	32
4	Alpha; Four-helical up-and-down bundle	15
5	Alpha; 4-helical cytokines	18
6	Alpha; EF-hand	15
7	Beta; Immunoglobulin-like beta-sandwich	74
8	Beta; Cupredoxins	21
9	Beta; Viral coat and capsid proteins	29
10	Beta; ConA-like lectins/glucanases	13
11	Beta; SH3-like barrel	16
12	Beta; OB-fold	32
13	Beta; beta-Trefoil	12
14	Beta; Trypsin-like serine proteases	13
15	Beta; Lipocalins	16
16	A/B; beta/alpha (TIM)-barrel	77
17	A/B; FAD (also NAD)-binding motif	23
18	A/B; Flavodoxin-like	24
19	A/B; NAD(P)-binding Rossmann-fold	40
20	A/B; P-loop nucleotide triphosphate hydrolases	22
21	A/B; Thioredoxin-like	17
22	A/B; Ribonuclease H-like motif	22
23	A/B; alpha/beta-Hydrolases	18
24	A/B; Periplasmic binding protein-like	15
25	A+B; beta-Grasp	15
26	A+B; Ferredoxin-like	40
27	Small; Small inhibitors, toxins, lectins	39
Total		693

and the 3D structural comparison methods. The original dataset does not contain the 3D structure of the proteins. Therefore, we downloaded the 3D structure for each protein from ASTRAL SCOP [24] for use by the 3D structural comparison methods. In the experiments, 18 types of similarity (or dissimilarity) metrics were used. Table 2 lists all distance metrics. We used 12 distances computed from 12 types of feature vectors extracted from the sequence of the amino acids [17] (D-1 to D-12), 2 dissimilarity measures from two 3D structural comparison methods (D-13 and D-17), and 4 similarity measures from four 3D structural comparison methods (D-14, D-15, D-16, and D-18). All the distance measures D1 to D-18 satisfy the axiom of a metric. Note that measures D-13 to D-18 lack of explicit feature vector representation. Since our main focus is to validate the effectiveness of the proposed method for metrics combination, we did not use any novel methods for computing the distances of the features from the sequences of the amino acids.

4.1.1 Experimental Setting

We conducted the experiments 10 times by repeatedly splitting the dataset into 50% training and 50% test data using stratified random sampling and used the same 10 set of training and test data for the conventional methods and the proposed methods. Since we compared the proposed method with GMKL [19], all metrics needed to be converted to kernel matrices, which are basically similarity matrices. All of the metrics are also needed to be converted to distance matrices for applying the proposed method. For GMKL, we converted all distances into kernel matrices by using a radial basis function $\exp(-d_{ij}^2/\sigma)$. The σ was tuned by simulated annealing, by using 5-fold cross validation of the training data. For the proposed method, to ensure the equivalence with those in GMKL, the obtained kernel matrices were converted back to distance matrices by using the method described in Sect. 3.1.

The parameters of the proposed method (margin l, balancing parameter μ, and k for k-NN) and GMKL were also tuned using 5-fold cross validation of the training data. For the proposed method, a number of balancing parameters and margins were then prepared, with values ranging from 0.01

to 1 in steps of 0.05. For GMKL, several step size limits for the backtracking line-search algorithm were set as follows: upper values: $0.1, 1.0, 2.1$; lower values: $0.1, 0.3, 0.9$. For the SVM used by GMKL, we also tuned the misclassification penalty, which was set to 0.5, 1, 1.5, and 2. We then applied the proposed method to all combinations of parameters with hinge loss, smooth hinge loss, and logistic loss optimization, and GMKL with L2 regularization. When using averaging, voting, and the proposed method, the k-nearest neighbor of the combined distances was used. When using GMKL, one-against-all SVM was used.

4.1.2 Experimental Results

Firstly, as the baseline performance, classification experiments were conducted for each feature and method listed in Table 2 with k-nearest neighbor. Table 3 summarizes the experimental results for all the features based on amino acid sequences and methods based on the 3D structures. In the study of protein analysis, protein fold prediction using the features from amino acid sequences has been known to be a very difficult task. Consequently, the classification results when using D-1 to D-12 were poor, just as expected. In contrast, although protein fold classification using the 3D structure is also a challenging task, since the fold categories are determined by the 3D geometrical structure the performances when using D-13 to D-18 were much better than when using D-1 to D-12. From Table 3, we can also confirm that the scoring used in the protein structure comparison, either similarity or dissimilarity, could significantly affect the classification results. For example, when using CE [2], [6], the recall when using RMSD (D-13) was 83.4%, but when using the Z-Score (D-14) the recall improved to 92.2%. When using TM-Align, the recall when using RMSD (D-17) was only 59.9%. In contrast, when using TM-Score (D-18), the performance significantly improved to 98.7%.

Secondly, we used several combinations of the distance matrices to demonstrate both the effectiveness and limitations of the proposed method, and compared the results with naïve methods such as averaging, voting, and GMKL [19] as the representative method from multiple kernel learning based algorithm. We did not compare our method with the conventional feature-based distance metric learning methods [8]–[14] because of the unavailability of the explicit feature vectors. Tables 4 and 5 show the combinations of distance matrices and the classification results for each method, respectively. In Table 5, HL, SHL, and LL are the proposed methods using hinge loss, smooth hinge loss, and logistic loss, respectively. In the following, we discuss the performance of each method shown in Table 5.

When using averaging, which is basically the same as using a uniform weight (each entry of $\boldsymbol{w}$ is $1/S$, where S is the number of the distances) to combine the distances, the performance was significantly affected by the number of good distance metrics. For example, when all the distance metrics were from the 3D structure-based method (combination C-4), the averaging method achieved high precision (97.0%) and recall (96.4%). However, when the metrics from both amino acid sequence and 3D structure based methods were combined, averaging performed poorly compared with the other methods.

The voting-based method uses the most votes among the distance metrics to determine the fold category of an input protein. The overall performance of the voting-based method was better than that of averaging, but voting also requires a majority of the distance metrics to have good accuracy. The performance of the voting method was better than averaging when multiple distance metrics from 3D structure-based methods were included (combinations C-3).

Multiple kernel learning is known to be the most widely used method for combination of metrics. When

Table 2 List of similarity and dissimilarity metrics used in the experiments. Metrics D-1 to D-12 are based on the feature extraction of the protein sequences [16], [17]; D-13 to D-18 are similarity (or dissimilarity) measurements based on the 3D protein structures.

Name	Type	Features or methods
D-1	Euclidean distance	Amino acid composition
D-2	Euclidean distance	Predicted 2nd structure
D-3	Euclidean distance	Hydrophobicity
D-4	Euclidean distance	van der Waals Volume
D-5	Euclidean distance	Polarity
D-6	Euclidean distance	Polarizability
D-7	Euclidean distance	Pse Amino Acid-1
D-8	Euclidean distance	Pse Amino Acid-4
D-9	Euclidean distance	Pse Amino Acid-14
D-10	Euclidean distance	Pse Amino Acid-30
D-11	Euclidean distance	SW BLOSUM62
D-12	Euclidean distance	SW PAM50
D-13	RMSD dissimilarity	CE [2], [6]
D-14	Z-Score similarity	CE [2], [6]
D-15	Canonical angles similarity	CMSM [21]
D-16	FATCAT similarity	FATCAT [4], [6]
D-17	RMSD dissimilarity	TM-Align [5]
D-18	TM-Score similarity	TM-Align [5]

Table 4 Combinations of distance matrices.

Comb.	D-1 to D-12	3D structure based					
		D-13	D-14	D-15	D-16	D-17	D-18
C-1	✓						
C-2	✓						✓
C-3	✓	✓	✓	✓	✓	✓	✓
C-4		✓	✓	✓	✓	✓	✓
C-5	✓	✓					
C-6	✓	✓	✓				
C-7	✓	✓	✓	✓			
C-8	✓	✓	✓	✓	✓		
C-9	✓	✓	✓	✓	✓	✓	

Table 3 Average precision (Prec.) and recall (Rec.) [%] for each baseline feature and method.

	D-1	D-2	D-3	D-4	D-5	D-6	D-7	D-8	D-9	D-10	D-11	D-12	D-13	D-14	D-15	D-16	D-17	D-18
Prec.	42.4	36.6	27.8	30.9	28.2	24.9	35.0	33.8	30.1	27.8	51.4	48.1	86.6	92.7	75.3	92.4	83.3	98.8
Rec.	38.2	37.4	24.9	25.5	25.1	21.8	32.1	29.0	23.9	20.4	44.8	43.5	83.4	92.2	74.2	91.2	59.9	98.7

Table 5 Average classification results in term of precision and recall [%] for combinations of distance matrices. HL, SHL, and LL stand for hinge loss, smooth hinge loss, and logistic loss, respectively. Bold text indicates the best precision or recall among the methods.

	Precision						Recall					
	Averaging	Voting	GMKL [19]	Ours			Averaging	Voting	GMKL [19]	Ours		
				HL	SHL	LL				HL	SHL	LL
C-1	52.3	58.0	**58.0**	51.0	51.5	50.6	41.3	50.3	**54.8**	45.6	45.9	42.2
C-2	63.8	67.7	86.6	**99.0**	98.7	98.1	54.0	59.5	84.4	**98.9**	98.6	98.0
C-3	89.4	91.4	97.6	**98.8**	98.5	97.8	87.0	88.3	97.4	**98.7**	98.3	97.5
C-4	97.0	97.0	92.5	**98.8**	98.4	97.3	96.4	96.5	92.3	**98.7**	98.2	96.8
C-5	65.8	66.1	86.0	90.1	90.2	**90.4**	55.2	58.3	84.4	89.2	89.2	**89.5**
C-6	75.5	75.4	93.1	95.4	**95.9**	95.7	67.1	68.3	92.3	95.0	**95.6**	95.4
C-7	76.9	78.5	92.5	96.0	**96.1**	95.9	69.5	71.5	91.7	95.7	**95.8**	95.6
C-8	79.4	86.1	93.4	**96.3**	**96.3**	96.0	72.5	80.6	92.8	**96.0**	**96.0**	95.7
C-9	85.8	87.9	96.5	**96.8**	96.5	96.4	81.6	82.9	96.2	**96.4**	95.9	95.9

Table 6 Average precision and recall [%] for combination C-1 using LMNN, SVM, and random forest (RF).

	Precision		Recall	
	Voting	Concatenation	Voting	Concatenation
LMNN	61.78	56.25	53.04	52.50
SVM	63.74	59.66	32.18	38.50
RF	63.86	63.48	46.18	52.41

GMKL [19] was used, the classification results improved significantly compared to results with the averaging and voting. For combination C-1, GMKL outperformed the other methods. However, the results for the other combinations were worse than with the proposed method. For comparison purpose, Table 6 shows the experimental results using original LMNN, SVM, and random forest (RF) for combination C-1 where feature representations were available. The combination was done by either voting or concatenation of the feature vectors. In voting-based method, the LMNN, SVM, or RF was first applied to each type of the features in the combination C-12 and then the final classification results were decided through voting among the 12 distances. In concatenation method, all feature vectors were first concatenated. Then, the LMNN, SVM, or RF was applied to the concatenated features. For SVM, we tested using linear, quadratic, and third degree polynomial kernel, where we reported the best results which were produced with quadratic kernel. For the RF, the number of trees was empirically set to 1000. When using voting-based method, all the methods obtained higher precisions than that of GMKL, where the highest precision of 63.86% was obtained by using random forest. With concatenation of the features before applying each method (LMNN, SVM, and RF), the precisions decreased. However, in term of recall, the performance of GMKL was still better than the performance of LMNN, SVM, and RF with either voting-based method or concatenation of the feature vectors.

The proposed method was used together with the three types of loss functions described in Sect. 3.2. The overall performances for the three loss functions were comparatively similar to one other. Although LMNN-based methods did not perform well for combination C-1, the precision and recall for the proposed method with smooth hinge loss (SHL) were slightly better than the best results from the single measure (D-11). In the case when the combination consists of distances from both amino acid sequence features and 3D structure-based methods, the proposed method outperformed averaging, voting, and GMKL. This suggests that when distance metrics are very different in terms of performance, our method is still able to find the optimal weights for combination. In combination C-2, where there were 12 poor distances with one very good distance measure, the proposed method with hinge loss could still achieve precision and recall comparable to the single measure of D-18.

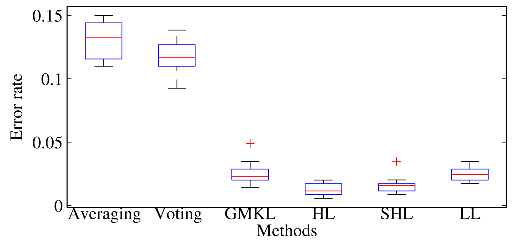

Fig. 4 Error rates for combination C-3.

Table 7 The p-value of the t-test for single measure D-18 and combination C-3.

	D-18	GMKL	HL	SHL	LL
D-18	-	0.0046	0.7351	0.2693	0.0007
GMKL	0.0046	-	0.0005	0.0265	0.6470
HL	0.7351	0.0005	-	0.0661	0.0002
SHL	0.2693	0.0265	0.0661	-	0.0081
LL	0.0007	0.6470	0.0002	0.0081	-

Figure 4 shows the boxplot of the error rate for combination C-3. Table 7 shows the p-value of the t-test for the 10 repeated experiments using the proposed method with combination C-3, GMKL, and the single measure D-18. From the Table 7, we can conclude that with more than 95% confidence ($p = 0.0005$) the proposed method using hinge loss performed better than that of the GMKL. The t-test results show that the combination of all 18 distances using the proposed method with hinge loss and smooth hinge loss performed the same as the single measure D-18. This points to the capability of the proposed method in finding an optimal combination when combining heterogeneous metrics to eliminate the necessity for *preselecting* the distance metrics.

Table 8 Average computational time and iterations for the proposed methods and GMKL for combination C-3.

	HL	SHL	LL	GMKL
Time (in secs)	11.05	148.88	133.59	628.63
Iteration	80	131	60	-

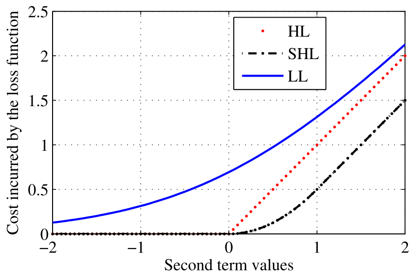

Fig. 5 Costs imposed by hinge loss (HL), smooth hinge loss (SHL), and logistic loss (LL).

Table 9 Average of optimal weight coefficients [%] for combination C-3, obtained by hinge loss (HL), smooth hinge loss (SHL), and logistic loss (LL) over the 10 repeated experiments.

Distances	HL	SHL	LL
D-1	3.06 ± 0.11	0.89 ± 0.03	4.20 ± 0.00
D-2	1.61 ± 0.04	0.77 ± 0.03	2.37 ± 0.00
D-3	0.41 ± 0.01	0.27 ± 0.01	2.56 ± 0.00
D-4	0.17 ± 0.00	0.24 ± 0.01	2.42 ± 0.00
D-5	0.00	0.24 ± 0.01	2.55 ± 0.00
D-6	0.00	0.26 ± 0.01	2.01 ± 0.00
D-7	0.10 ± 0.00	0.23 ± 0.00	2.86 ± 0.00
D-8	0.05 ± 0.00	0.21 ± 0.00	2.30 ± 0.00
D-9	0.00	0.18 ± 0.00	1.07 ± 0.00
D-10	0.00	0.00	2.20 ± 0.01
D-11	0.45 ± 0.01	0.37 ± 0.01	0.52 ± 0.00
D-12	0.00	0.28 ± 0.01	0.19 ± 0.00
D-13	3.03 ± 0.06	4.88 ± 0.25	17.63 ± 0.02
D-14	3.78 ± 0.22	4.63 ± 0.38	16.22 ± 0.01
D-15	18.92 ± 2.70	24.14 ± 4.75	8.16 ± 0.02
D-16	0.32 ± 0.01	0.48 ± 0.02	3.85 ± 0.00
D-17	4.39 ± 0.24	4.10 ± 0.28	13.06 ± 0.01
D-18	63.72 ± 5.45	57.83 ± 7.97	15.85 ± 0.01

4.1.3 Computational Time and Further Discussion

Table 8 compares the average computational time and iteration required for the proposed method and GMKL to complete the optimization for combination C-3. The experiments were conducted using Intel Xeon E5-2630 2.3 Ghz with 32 GB RAM. The proposed methods were implemented using Matlab, while the computation of the cost function and gradient for smooth hinge loss and logistic loss were implemented using C. When using the proposed method with hinge loss optimization, one experiment can be completed in a relatively short time, this is because the computation of the subgradient is very simple. With smooth hinge loss, more iterations were required before convergence, which points to the longer computational time. Logistic loss required the least iterations to converge. However, since it requires to compute exponential function in each iteration, the average computational time was longer than that of the hinge loss. For GMKL, we used the publicly available Matlab code from [19] that uses the quadratic programming function from Matlab optimization toolbox. Average computational time of GMKL was 628.63 seconds. GMKL was basically fast, except for the case when convergence solution could not be obtained. In that case, we used active-set method in quadratic programming, which can deal with non-convex problem with the trade-off in computational time.

Next, we conducted 10 repeated experiments for combination C-1, C-2, C-3, and C-4 by using hinge loss with random initial weights instead of equal weights. For combination C-3, the average precision and recall were 98.7% and 98.6%, respectively. For combination C-4, the average precision and recall were both 98.7%. For combination C-2, the average precision and recall were 99.0% and 98.9%, respectively. For combination C-1, the average precision and recall were 50.6% and 45.3%, respectively. By comparing these results with those in Table 5, we can see that the proposed method is not largely affected by the initial weights.

Finally, we discuss the characteristics of the hinge loss, smooth hinge loss, and logistic loss. Figure 5 shows how the costs imposed by the hinge loss were smoothened by smooth hinge loss and logistic loss. Table 9 shows the optimal weight coefficients when combining all distance metrics (combination C-3) using each loss function. Hinge loss tends to sparsely pick up the metrics, where the majority of the weights were given to D-18 (63.72%). Although D-15 performed worse than other 3D structure based method (see Table 3), the average weight for D-15 was 18.92%, which were higher than the other distances. This suggests that the proposed method can *selectively adjust the weight* for the most optimal combination. When using hinge loss, there were five distances with zero weights (D-5, D-6, D-9, D-10, and D-12). Smooth hinge loss also put most of the weights to D-18 (57.83%). However, smooth hinge loss considered more distance metrics than hinge loss whereas only D-10 had zero weight. When the cost function used logistic loss, all of the distance metrics were used. This suggests that using smooth loss functions, such as smooth hinge loss or logistic loss, the weights were more uniformly distributed over the available distance metrics.

4.2 Experiments on ENZYMES Dataset

In this experiment, we conducted classification experiments on ENZYMES dataset that contains six classes of protein enzymes (100 proteins in each class), where a graph-based representation is used to represent each protein [15]. Since it is not trivial to combine graph-based representation, conventional feature combinations and metric learning cannot be used. We combined three types of graph-based kernel matrices: the Weisfeiler-Lehman based subtree kernel [25], propagation based graph kernel [26], and GraphHopper kernel [27]. We conducted three repeated experiments by randomly selecting 30% of samples in each class as training and the rest as testing. When using the proposed method, the kernel matrices were converted into distance matrices as in the previous experiments with the Ding Dubchak dataset

Table 10 Summary of the classification results in term of precision and recall [%] for ENZYMES dataset. Bold text indicates the best results among the methods.

Method	Precision	Recall
Single metric (k-NN):		
- GraphHopper [25]	55.3	54.4
- Propagation [26]	19.6	18.3
- Weisfeiler-Lehman [27]	31.8	27.5
Combination:		
- Averaging (k-NN)	53.2	46.9
- Voting (k-NN)	52.7	31.7
- GMKL (SVM)	55.6	50.1
Ours (fixed step size, 1-NN):		
- Hinge loss	55.3	54.4
- Smooth hinge loss	55.3	54.4
- Logistic loss	53.7	52.7
Ours (with Armijo rule, 1-NN):		
- Hinge loss	54.2	53.4
- Smooth hinge loss	**55.8**	**54.9**
- Logistic loss	53.1	52.4

and fixed value of $k = 1$ for the k-NN was used. When using GMKL, the kernel matrices were directly used. The parameters for the proposed methods and the GMKL were tuned in the same manner as in the previous experiments. In addition to the use of the fixed value for step size in the proposed methods ($\epsilon = 10^{-5}$), we also used an adaptive step size by using Armijo rule [28]. We reported the experimental results based on the average of the three repeated experiments.

4.2.1 Experimental Results

Table 10 summarizes the experimental results. By using k-NN, the classification results using each kernel were relatively low. With averaging and voting, the performance were worse than those with the GraphHopper kernel which has 55.3% precision and 54.4% recall. When using GMKL, the precision was slightly better than that of GraphHopper, while the recall was not as bad as the averaging and voting. When using the proposed method with fixed step size, the hinge loss and smooth hinge loss put weight only to GraphHopper kernel, resulting the same performance as GraphHopper kernel. With Armijo rule, the proposed method with smooth hinge loss could slightly improve the performance of the GraphHopper kernel, where the precision and recall were improved to 55.8% and 54.9%, respectively. These results suggest that the proposed method can still effectively deal with distance metrics that have poor performance.

5. Concluding Remarks

In this paper, we proposed a method that finds optimal combinations of distance metrics for protein fold classification task by generalizing the concept of LMNN for combining multiple distance metrics. LMNN was originally proposed for learning Mahalanobis-based distance metric from a set of feature vectors of sample data. On the other hand, our objective is to find an optimal combination of distance metrics from multiple protein structural measurement methods, where natural feature vector representation is not always available. The final distance, termed as *overall distance*, is then defined as a linear combination of the multiple distance measures. The convex optimization problem is solved by using an iterative algorithm based on the subgradient and gradient methods; for this we adopted three types of loss functions: hinge loss, smooth hinge loss, and logistic loss. The effectiveness of the proposed method was demonstrated through classification experiments using the Ding Dubchak dataset and ENZYMES dataset, where the proposed method outperformed the naive conventional methods and GMKL. In particular, the proposed method could effectively find an optimal combination for distance metrics with very different performance. Thus, we can avoid the difficulties in *preselecting* distance metrics among a number of available distance metrics. However, an analyst should make an effort to make a list of possible distance measures before applying the proposed method. If there is no single good measure is included in the list of the metrics, the proposed method may not work effectively. In contrast, it is not a big problem if many poor metrics exist in the list because the proposed method can construct an optimal combination of poor metrics and good metrics.

To further improve the performance of the proposed method, imposing a locality neighborhood constraint by considering only the instances in a close region [10] is one direction of our future works. To impose explicit locality, another direction is by imposing different weights for different region ranges, as in the local metric learning [29], to our problem formulation. Since the proposed method is technically generic, we will also consider applying the proposed method to different classification tasks other than the protein fold classification problem.

Acknowledgments

We thank the reviewers for the constructive feedback. This work was supported by JSPS KAKENHI Grant Numbers 25870811 and 25540062.

References

[1] P. Koehl, "Protein structure similarities," Current Opinion in Structural Biology, vol.11, no.3, pp.348–353, 2001.

[2] I.N. Shindyalov and P.E. Bourne, "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path," Protein engineering, vol.11, no.9, pp.739–747, 1998.

[3] L. Holm and C. Sander, "Protein structure comparison by alignment of distance matrices," Journal of molecular biology, vol.233, no.1, pp.123–138, 1993.

[4] Y. Ye and A. Godzik, "Flexible structure alignment by chaining aligned fragment pairs allowing twists," Bioinformatics, vol.19, no.Suppl 2, pp.ii246–ii255, 2003.

[5] Y. Zhang and J. Skolnick, "TM-align: a protein structure alignment algorithm based on the TM-score," Nucleic acids research, vol.33, no.7, pp.2302–2309, 2005.

[6] A. Prlić, S. Bliven, P.W. Rose, W.F. Bluhm, C. Bizon, A. Godzik, and P.E. Bourne, "Precalculated protein structure alignments at the RCSB PDB website," Bioinformatics, vol.26, no.23, pp.2983–2985, 2010.

[7] C.H. Suryanto, S. Jiang, and K. Fukui, "Protein structure similarity based on multi-view images generated from 3D molecular visualiza-

tion," Proc. of the 21st International Conference on Pattern Recognition, pp.3447–3451, 2012.
[8] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in Computer Vision - ECCV 2002, ed. A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Lecture Notes in Computer Science, vol.2353, pp.776–790, Springer Berlin Heidelberg, 2002.
[9] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," Advances in Neural Information Processing Systems, pp.513–520, 2005.
[10] K.Q. Weinberger and L.K. Saul, "Distance metric learning for large margin nearest neighbor classification," Journal of Machine Learning Research, vol.10, pp.207–244, 2009.
[11] C. Xiong, D. Johnson, R. Xu, and J.J. Corso, "Random forests for metric learning with implicit pairwise position dependence," Proc. of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.958–966, 2012.
[12] F. Wang, J. Sun, and S. Ebadollahi, "Composite distance metric integration by leveraging multiple experts' inputs and its application in patient similarity assessment," Statistical Analysis and Data Mining, vol.5, no.1, pp.54–69, 2012.
[13] Z. Cui, W. Li, D. Xu, S. Shan, and X. Chen, "Fusing robust face region descriptors via multiple metric learning for face recognition in the wild," Proc. of the Computer Vision and Pattern Recognition, pp.3554–3561, 2013.
[14] X. Zhai, Y. Peng, and J. Xiao, "Heterogeneous metric learning with joint graph regularization for cross-media retrieval," Proc. of the 27th AAAI Conference on Artificial Intelligence Heterogeneous, pp.1198–1204, 2013.
[15] K.M. Borgwardt, C.S. Ong, S. Schönauer, S.V.N. Vishwanathan, A.J. Smola, and H.P. Kriegel, "Protein function prediction via graph kernels," Bioinformatics, vol.21, pp.47–56, 2005.
[16] C.H.Q. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," Bioinformatics, vol.17, no.4, pp.349–358, 2001.
[17] T. Damoulas and M.A. Girolami, "Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection," Bioinformatics, vol.24, no.10, pp.1264–1270, 2008.
[18] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, and M.I. Jordan, "Learning the kernel matrix with semidefinite programming," Journal of Machine Learning Research, vol.5, pp.27–72, 2004.
[19] M. Varma and B.R. Babu, "More generality in efficient multiple kernel learning," Proc. of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, pp.1065–1072, ACM, 2009.
[20] H. Hino, N. Reyhani, and N. Murata, "Multiple kernel learning with gaussianity measures," Neural Computation, vol.24, no.7, pp.1853–1881, 2012.
[21] C.H. Suryanto, H. Hino, and K. Fukui, "Combination of multiple distance measures for protein fold classification," Proc. of the 2nd Asian Conference on Pattern Recognition, pp.440–445, 2013.
[22] B. Kulis, "Metric learning: A survey," Foundations and Trends in Machine Learning, vol.5, no.4, pp.287–364, 2013.
[23] J.D.M. Rennie and N. Srebro, "Loss functions for preference levels: Regression with discrete ordered labels," Proc. of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling, pp.180–186, 2005.
[24] J.M.M. Chandonia, G. Hon, N.S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S.E. Brenner, "The ASTRAL Compendium in 2004," Nucleic Acids Research, vol.32, pp.D189–D192, 2004.
[25] N. Shervashidze and K. Borgwardt, "Fast subtree kernels on graphs," Proc. of the Neural Information Processing Systems, pp.1660–1668, Neural Information Processing Systems Foundation, 2009.
[26] M. Neumann, N. Patricia, R. Garnett, and K. Kersting, "Efficient graph kernels by randomization," Proc. of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I, pp.378–393, 2012.
[27] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K.M. Borgwardt, "Scalable kernels for graphs with continuous attributes," Proc. of the Neural Information Processing Systems, pp.216–224, 2013.
[28] C.T. Kelley, "Line search methods and the Armijo rule," in Iterative Methods for Optimization, pp.40–52, SIAM, 1999.
[29] J. Wang, A. Woznica, and A. Kalousis, "Parametric local metric learning for nearest neighbor classification," Advances in Neural Information Processing Systems, pp.1610–1618, 2012.

Chendra Hadi Suryanto received his Bachelor's degree in computer science from Institut Teknologi Sepuluh Nopember in 2004 and has worked as a software engineer for several years at various IT companies. He obtained his Master's degree in computer science from the University of Tsukuba in 2013. He is currently a Ph.D. candidate at the University of Tsukuba. His research interests include pattern recognition, computer vision, and human-machine interface systems.

Kazuhiro Fukui received his B.E. and M.E. (Mechanical Engineering) degrees from Kyushu University in 1986 and 1988, respectively. In 1988, he joined Toshiba Corporate Research and Development Center and served as a senior research scientist at Multimedia Laboratory in 2002. He received his Ph.D. from Tokyo Institute of Technology in 2003. He is currently a professor in the Department of Computer Science, Graduate School of Systems and Information Engineering at University of Tsukuba. His interests include the theory of computer vision, pattern recognition, and applications of these theories. He has been serving as a program committee member at many pattern recognition and computer vision conferences, including as an Area Chair of ICPR'12 and ICPR'14. He is a member of IEICE, IPSJ, SIAM and IEEE.

Hideitsu Hino received his Bachelor's degree in engineering in 2003, and Master's degree in Applied Mathematics and Physics in 2005 from Kyoto University, Japan. He joined Hitachi's Systems Development Laboratory and worked as a research staff from April 2005 to August 2007. He earned Doctor's degree in engineering in 2010 from Waseda University. From April 2013, he is an Assistant Professor at University of Tsukuba. His research interests include the analysis of learning algorithms from the view point of geometry. He is also interested in time series analysis, kernel methods, distance metric learning, ranking models and their applications. He is a member of IEICE, IPSJ, and IEEE.