BIOINFORMATICS

Vol. 00 no. 00 2013
Pages 1–10

Supplemental Data

Pooya Zakeri [1,2], Ben Jeuris [3], Raf Vandebril [3], Yves Moreau [1,2]

[1] Department of Electrical Engineering ESAT, SCD-SISTA KU Leuven, Leuven , Belgium.

[2] Future Health Department, iMind, Leuven , Belgium.

[3] Department of Computer Science, KU Leuven, Leuven, Belgium.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

1 METHODOLOGICAL APPROACH

Harmonic mean

The harmonic mean of positive numbers $a_1, \ldots, a_k$ is given by the expression

$$\mathcal{H}(a_1, \ldots, a_k) = \left(\frac{1}{k} \sum_{i=1}^{k} a_i^{-1} \right)^{-1}. \tag{1}$$

This mean is often used in situations where small numbers need to be emphasized and large outliers are less important.

Link between the Log-Euclidean and geometric mean

The Log-Euclidean mean can be considered as a generalization of the notion of geometric mean. Indeed, if $x_1, \ldots, x_n$ are n positive numbers, then their geometric mean is given by

$$\mathcal{G}(x_1, \ldots, x_n) = (x_1, \ldots, x_n)^{\frac{1}{n}} = \exp\left(\frac{1}{n} \sum_{i=1}^{n} \log(x_i) \right). \tag{2}$$

For SPD matrices, the log-Euclidean mean can be seen an approximation to the geometric mean by considering both as an adaptation of the arithmetic mean towards positive definite matrices.

First of all, the arithmetic mean can be stated as follows:

$$\mathcal{A}(A_1, \ldots, A_k) = \min_{X \in \mathbb{R}^{n \times n}} \sum_{i=1}^{k} \|X - A_i\|_F^2. \tag{3}$$

Here, each term in the cost function expresses the Euclidean distance between matrices X and A_i, and the minimizer will be exactly the arithmetic mean as we know it. Now, by switching to positive definite matrices, we restrict the search space of X to this set ($\mathcal{P}_n$) and the Euclidean distance is replaced with the Riemannian distance between positive definite matrices. The new optimization problem has the geometric matrix mean (the Karcher mean) as its minimizer:

$$\mathcal{G}(A_1,\ldots,A_k) = \min_{X\in\mathcal{P}_n} \sum_{i=1}^{k} \| \log(A_i^{-1/2} X A_i^{-1/2}) \|_F^2. \tag{4}$$

This shows the relation between the arithmetic and the geometric mean.

By contrast, there is an obvious one-to-one relation between the (Euclidean) vector space of symmetric matrices and the positive definite matrices using the matrix exponential and logarithm. This relation allows an interesting interpretation for the log-Euclidean mean: we start with the positive definite matrices $A_1,\ldots,A_k$, which we transform to symmetric matrices using the matrix logarithm. Of these transformed matrices the arithmetic mean is taken, and finally, the result is tranformed back to the positive definite matrices using the matrix exponential.

The Karcher mean optimization algorithm

Manifold optimization corresponds to a generalization of classical optimization algorithms where geometric concepts, such as gradient, Hessian, and others are redefined in the context of the manifold structure. In Algorithm 1, the general outline of such an optimization algorithm is shown. Another important operation which arises in manifold optimization is that of a retraction, which is the generalization of the action “taking a step in a certain direction”. The use of a retraction assures that each step in Algorithm 1 stays on the manifold, which is one of the advantages of this type of optimization.

Algorithm 1 The Karcher mean optimization algorithm

```
Let $A_1,\ldots,A_k$ be SPD matrices, $X_0$ an initial guess, and $R_X$ a retraction starting at a point $X$
• for $k = 0, 1, \ldots$
    • Determine the search direction $\xi_k$ using Steepest Descent, Conjugate Gradient, Newton, ...
    • $X_{k+1} = R_{X_k}(t^A \xi_k)$ where $t^A$ is an appropriate stepsize
• end
```

The computational cost of the steepest descent algorithm for the Karcher mean is of the order $O(n^3k)$ for one iteration. These iterations are stopped when the (absolute or relative) distance between two consecutive iterations is less than some specified tolerance, or when a maximum number of iterations is reached.

2 METHODS

2.1 Feature vectors

PsePSSM The PsePSSM was originally introduced in [Chou, 2001] to avoid complete loss of the sequence-order information.. In this representative model the score of PSSM profiles ($M_{i\to j}$) is first standardized ($S_{i\to j}$) as described in [Shen and Chou, 2009], and then a protein P can be represented as

$$\bar{\mathbf{P}}^{\xi}_{\mathrm{PsePSSM}} = [\bar{S}_1\ \bar{S}_2 \ldots \bar{S}_{20}\ G^{\xi}_1\ G^{\xi}_2 \ldots G^{\xi}_{20}]^T, \tag{5}$$

with

$$\bar{S}_j = \frac{1}{L}\sum_{i=1}^{L} S_{i\to j} \quad (j = 1, 2, \ldots, 20), \tag{6}$$

$$G^{\xi}_j = \frac{1}{L-\xi}\sum_{i=1}^{L-\xi} \left(S_{i\to j} - S_{(i+\xi)\to j}\right)^2 \quad (j = 1, 2, \ldots, 20;\ \ \xi < L), \tag{7}$$

where the first 20 coordinates are associated with the conventional sequence order-free representation of PSSM described earlier (PS3), and the other coordinates reflect the effect of sequence order. In our work the optimal range of ξ is considered from 0 to 10.

Functional domain composition using InterPro and CDD

We used InterPro Release 18 (13th February 2013), which consists of 24,356 entries stated with 27,358 indexes. The feature vector for this model is constructed by defining a vector domain type with 27,358 coordinates (FunD-InterPro) in which each coordinate is related to one FunD index from InterPro. Then, the instructions described by [Chou and Shen, 2007] are followed. First, the program IPRSCAN [Quevillon *et al.*, 2005] is used to compare each protein sequence in our data set with each of the domain sequences in the InterPro database. For each protein sequence in our data set, the i-th coordinate of its feature vector is then assigned 1 where the i-th domain sequence of the InterPro database is significantly similar to a sequence segment in the protein domain sequence; otherwise, it is assigned 0.

The CDD database is a comparatively complete and well-annotated FunD database that consists of the domain models imported from a series of well-known external protein FunD databases including 8,607 domains from NCBI CDD curation effort, 1,013 domains from SMART v6.0, 13,672 domains from PFAM v26.0, 4,873 domains from COGs v1.0, 10,885 domains from PRK v6.0, and 4,284 domains from TIGRFAM v13, organized into 3,295 multi-model superfamilies.

We utilized CDD v3.08 (01 NOV 2012), which contains 43,334 common domains and families from all source databases. The feature vector for this model is constructed by defining a vector domain type with 43,334 coordinates (FunD-cdd) for a protein sequence in which each coordinate is related to one FunD index from CDD. First, the program for reverse PSI-BLAST (RPS-BLAST) [Schaffer *et al.*, 2001] is used to compare each protein sequence in our data set with each of the domain sequences in the CDD database. For each protein sequence in our data set, the i-th coordinate of its feature vector is assigned 1 when the i-th profile of the CDD database is significantly similar to the profile of a protein domain sequence; otherwise, it is assigned .

Moreover, since the function of some proteins might be unknown, the InterPro and CDD database may fail to cover a benchmark data set completely. Hence, the feature vector for these proteins becomes a null vector. This problem can be solved by the hybridization approach, which uses another model representation for a protein sample corresponding to a null vector. Fortunately in the DD training data set, there are just 3 and 10 proteins that did not have any FunD in the InterPro and CDD databases respectively. The randomized process described by [Zakeri *et al.*, 2011] is followed to assign their FunD vector. In addition, the two types of functional domain composition (InterPro and CDD) can be integrated to form a single FunD feature vector with 70692 dimensions (FunD-Combined).

3 RESULTS

Supplemental Table 1 shows a summary of training and test data sets belonging to the 27 protein domain folds of SCOP corresponding to all major structural classes : $\alpha, \beta, \alpha/\beta, \alpha + \beta$.

Parameter selection details

For each protein feature except for SWr1 and SWr2, the kernel widths are chosen heuristically, through rules of thumb which often yield good results in practice [De Bie *et al.*, 2007]. This results in a Gaussian RBF kernel, with bandwidth equal to twice the average Euclidean distance of a data point to its nearest neighbour in the entire data set. For SWr1 and SWr2, we use a Gaussian RBF kernel with bandwidth equal to eight times the average euclidean distance of a data point to its nearest in the whole data set. This kernel width for SWr1 and SWr2, as well as the C-parameter, are chosen by maximizing the accuracy performance using 5-fold cross validation on the training set. For SW1 and SW2 data sources, this maximum is searched over a grid of multipliers of the average distance of the data point to its nearest neighbour in the complete data set. A one-against-other SVM classifier is constructed based on each representative model of the protein samples. To train SVMs, we used LIBSVM-3.1 implementation of the SVM algorithm [Chang and Lin, 2011]. The performance of the individual classifiers on the DD test data is listed in Supplemental Table 2.

The detailed performance

As listed in Supplemental Table 3 we report the detailed performance of proposed kernel fusion framework l in predicting the folding type of 383 protein domains of test set among 27 well-known protein domain folds with respect to a given accuracy measure such as correct classification rate, Mattew's correlation coefficient (MCC) and F-score of each fold.

Furthermore, 26 random kernels are generated in order to investigate the behaviour of the Arithmetic and log-Euclidean means after sequentially adding random kernels. In (Supplemental Fig. 1) we report the effect of sequentially incorporating random kernels to 26 fold informative kernel matrices. The performance of the combined kernel using LogE-KF and AM fluctuates slightly but the trend is downward for both of them. Regarding the results of combined kernel through LogE-KF, the success rate is almost decreasing over adding random kernels, falling from 81.72% to just under 73% after adding 26th random kernel. A similar trend is observed for performances of the fused kernel using AM, dropping from 60.57% to just under 53%. It is noticeable that the performance of fused kernel using LogE-KF is considerably higher than the result of combined kernel using AM even after adding 26 random kernels. In addition, it is observed that both approaches have not acute behavior in dealing with random kernels.

Heuristic and simple MKL method (AK-MKL)

We consider a heuristic and simple MKL method [Qiu and Lane, 2009], which chooses the kernel weights based on the relationship between the kernel matrix and the covariance matrix of the target labels (AK-MKL) (for more details, see the supplemental data). The basic idea behind their methods is that a kernel matrix K_i with more similarity to the target labels (the covariance matrix) should contribute more to the combined kernel. Obtaining the weight of each kernel is formulated as

$$w_i = \frac{A(K_i, y)}{\sum_{i=1}^{K} A(K_i, y)}, \tag{8}$$

Table 1. Summary of the protein domain fold and their secondary structure class used in DD data set

Fold names	Index	N-Train	N-Test
α			
Globin-like	1	13	6
Cytochrome c	3	7	9
DNA-binding 3-helical bundle	4	12	20
4-helical up-and-down bundle	7	7	8
4-helical cytokines	9	9	9
EF-hand	11	6	9
β			
Immunoglobulin-like	20	30	44
Cupredoxins	23	9	12
Viral coat and capsid proteins	26	16	13
ConA-like lectin/glucanases	30	7	6
SH3-like barrel	31	8	8
OB-fold	32	13	19
Beta-trefoil	33	8	4
Trypsin-like serine proteases	35	9	4
Lipocalins	39	9	7
α/β			
(TIM)-barrel	46	29	48
FAD (also NAD)-binding motif	47	11	12
Flavodoxin-like	48	11	13
NAD(P)-binding Rossmann-fold	51	13	27
P-loop	54	10	12
Thioredoxin-like	57	9	8
Ribonuclease H-like motif	59	10	14
Hydrolases	62	11	7
periplasmic binding protein-like	69	11	4
$\alpha+\beta$			
β-Grasp	72	7	8
Ferrdoxin-like	87	13	27
Small inhibitors, toxins, lectins	110	14	27

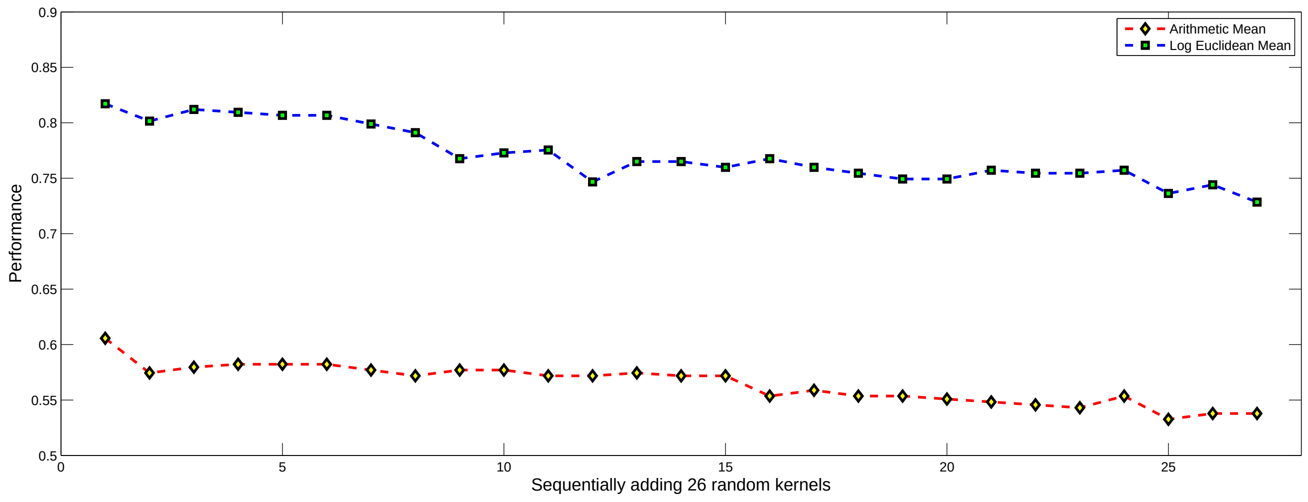

Fig. 1. The effect of sequentially adding 20 random kernels to 26 base kernels.

Table 2. The comparison between the performance of different protein features on the independent test set.

Protein feature	Dimension	Performance
Amino acid compos (C)	20	52.22
Hydroelectricity (H)	21	37.34
Polarity (P)	21	37.60
van der Walls volume (V)	21	34.73
Predicted secondary structure (S)	21	41.51
Polarizability (Z)	21	35.77
PseAA $\lambda = 1$ (L1)	22	44.39
PseAA $\lambda = 4$ (L4)	28	43.08
PseAA $\lambda = 14$ (L14)	48	43.86
PseAA $\lambda = 30$ (L30)	80	38.64
SW with BLOSUM62 (SWr1)	311	60.05
SW with PAM50 (SWr2)	311	59.01
PSSM profile 400D (PS1)	400	56.14
PSFM profile (PS2)	20	62.14
PSSM profile 20D (PS3)	20	62.66
PsePSSM$\xi = 0$ ((PSp0)	20	33.42
PsePSSM $\xi = 1$ (PSp1)	21	24.80
PsePSSM $\xi = 2$ (PSp2)	22	30.55
PsePSSM $\xi = 3$ (PSp3)	23	21.67
PsePSSM $\xi = 4$ (PSp4)	24	25.59
PsePSSM $\xi = 5$ (PP5)	25	31.07
PsePSSM $\xi = 6$ (PSp6)	26	26.11
PsePSSM $\xi = 7$ (PSp7)	27	27.94
PsePSSM$\xi = 8$ (PSp8)	28	28.46
PsePSSM $\xi = 9$ (PSp9)	29	24.28
PsePSSM $\xi = 10$ (PSp10)	30	27.15

where y represents the target labels and $A(K, y)$ is the (empirical) kernel-target alignment [Cristianini *et al.*, 2002].

The results of sequentially adding sequence-based features

In Figure 2, we consider the effect of sequentially incorporating sequence-based features according to the decreasing order of their kernel performances. The performance of the Hadamard product (elementwise) of base kernels (HPK) keeps increasing until PS2, PS3, SWr1, SWr2, Ps1 and C are included, achieving a total accuracy of 67%. However, the total accuracy of HPK declines dramatically if we continue to add less informative kernels. In fact, since the range of values of RBF kernel elements is between 0 and 1, multiplying more kernels causes the non-diagonal elements of combined kernels to approach zero and prediction fails. Moreover, the performance of uniformly weighted HM fluctuates by adding more features, but the trend is slightly upward, causing the best success rate of 65.80% to be achieved when adding the last, least informative kernel. In addition, the performance of uniformly weighted linear combinations of base kernels increases slowly by varying degrees until we include the sixteen most informative data sources, resulting in a best performance of 73.37%. Furthermore, comparable results with the best existing protein fold meta-predictors are achieved using the uniformly weighted linear kernel integration of 16 protein features. This performance is comparable with the result of the best existing fusion frameworks for classifying protein folds. By contrast, its performance decreases continuously if we continue to incorporate less informative protein features. However, there is a slight rise after adding PSp9 and then the performance decreases again when combining all kernels. This observation suggests that sequence based

Table 3. The correct classification rate, Mattew's correlation coefficient (MCC) and F-score results obtained using proposed kernel fusion frameworks on DD dataset classified into 27 well-known protein domain folds.

Folding Type	**GeoFold**			**LogEFold**			**AMKFold**			**HKFold**		
	Accuraccy	MCC	Fscore	Accuraccy	MCC	Fscore	Accuraccy	MCC	Fscore	Accuraccy	MCC	Fscore
Globin-like	100	1.00	1.00	100	1.00	1.00	100	1.00	1.00	100	1.00	1.00
Cytochrome c	100	1.00	1.00	100	1.00	1.00	100	1.00	1.00	100	0.95	0.95
DNA-binding 3-helical bundle	95	0.97	0.97	90	0.95	0.95	40	0.55	0.53	85	.0.63	0.63
4-helical up-and-down bundle	100	1.00	1.00	100	1.00	1.00	100	0.78	0.76	87.5	0.93	0.93
4-helical cytokines	100	0.95	0.95	100	0.90	.90	100	0.80	0.78	66.7	0.81	0.80
EF-hand	66.7	0.81	0.80	66.7	0.81	0.80	66.7	0.81	0.80	66.7	0.70	0.71
Immunoglobulin-like	97.7	0.99	0.99	90.9	0.95	0.95	79.55	0.78	0.80	77.3	0.63	0.67
Cupredoxins	83.3	0.87	0.87	83.3	0.87	0.87	66.7	0.81	0.80	66.7	0.81	0.80
Viral coat and capsid proteins	84.6	0.88	0.88	92.3	0.92	0.92	79.5	0.52	0.51	69.2	0.78	0.78
ConA-like lectin/glucanases	83.3	0.91	0.91	83.3	0.91	0.91	66.7	0.70	0.67	66.7	0.81	0.80
SH3-like barrel	100	0.94	0.94	87.50	0.93	0.93	76.9	0.61	0.55	50	0.53	0.53
OB-fold	73.7	0.82	0.82	84.2	0.86	0.86	50	0.40	0.33	42.1	0.45	0.47
Beta-trefoil	100	1.00	1.00	75.00	0.75	0.75	37.5	0.75	0.75	75	0.86	0.86
Trypsin-like serine proteases	50	0.57	0.57	50	0.71	0.67	21	0.28	0.29	50	0.49	0.50
Lipocalins	100	0.79	0.78	100	0.76	0.74	100	0.63	0.58	100	0.83	0.82
(TIM)-barrel	100	0.77	0.78	100	0.69	0.69	93.8	0.53	0.53	62.5	0.46	0.52
FAD (also NAD)-binding motif	91.7	0.91	0.92	91.7	0.91	0.92	66.7	0.72	0.73	91.7	0.91	0.92
Flavodoxin-like	61.6	0.78	0.76	30.8	0.49	0.44	30.8	0.49	0.44	30.8	0.44	0.42
NAD(P)-binding Rossmann-fold	92.6	0.90	0.91	85.2	0.84	0.85	55.6	0.55	0.58	59.3	0.71	0.71
P-loop	66.7	0.76	0.76	66.7	0.69	0.70	25	0.38	0.35	41.7	0.50	0.50
Thioredoxin-like	100	0.85	0.84	75	0.86	0.86	37.5	0.47	0.46	62.5	0.66	0.67
Ribonuclease H-like motif	75	0.86	0.85	75	0.86	0.86	0.08	0.28	0.15	50	0.70	0.67
Hydrolases	85.7	0.85	0.85	71.4	0.62	0.63	57.1	0.44	0.44	57.1	0.75	0.73
Periplasmic	50	0.71	0.67	25	0.50	0.40	0	0	0	25	0.50	0.40
β-Grasp	62.5	0.72	0.71	50	0.63	0.62	25	0.50	0.40	12.5	0.19	0.18
binding protein-like	55.6	0.71	0.70	37	0.56	0.53	11.1	0.27	0.19	51.8	0.55	0.57
Small inhibitors, toxins, lectins	96.3	0.92	0.93	96.3	0.98	0.98	88.9	0.94	0.94	92.6	0.75	0.76

PsePSSM features that reflect the effect of sequence order carry almost no complementary information with other protein features extracted from the PSSM profile (PS1, PS2, PS3, PSp0).

Similar trends are apparent for MKLdiv-dc, MKLdiv-conv and KA-MKL. The success rates fluctuate steadily but the trend is upward until the first 17 informative kernels are included, giving a best performance of over 72%. However, their performances begin to decline by adding less informative kernels and drop to 61.01%, 63.71%, and 61.88% respectively after combining all kernels. Although, it is worthwhile to notice that KA-MKL is a more cost-effective technique than MKLdiv-dc and MKLdiv-conv. The performance of SimpleMKL rises when incorporating PS2 and PS3, followed by a little drop after adding PS1, and then levels off until we fuse SWr1, SWr2, L1, L4. After adding S there is a sudden increase and the method peaks at a performance of 70.23% when including L30, P, and H. The trend experiences a slight decrease until we add V and Z and then there is a sudden drop by adding PSp0. If we continue to include less informative kernels, the performance steadily evolves downward and drops to 56.92% when fusing all kernels. It is easy to see that the results obtained by employing AM are comparable to the results of the MKL approaches for protein fold classification.

Contrary to the previous methods, the performance of AGH-KF increases gradually even when adding kernels considered to carry non-complementary information by AM. Its success rate is consistently outperforming other uniformly weighted kernel integration methods and almost always increases until the 26th kernel is included, resulting in the best performance of 86.68%. The same trend appears for LogE-KF, but the performance drops a little if we add the least informative kernel, where the total test accuracy of 81.72% is achieved. The experimental

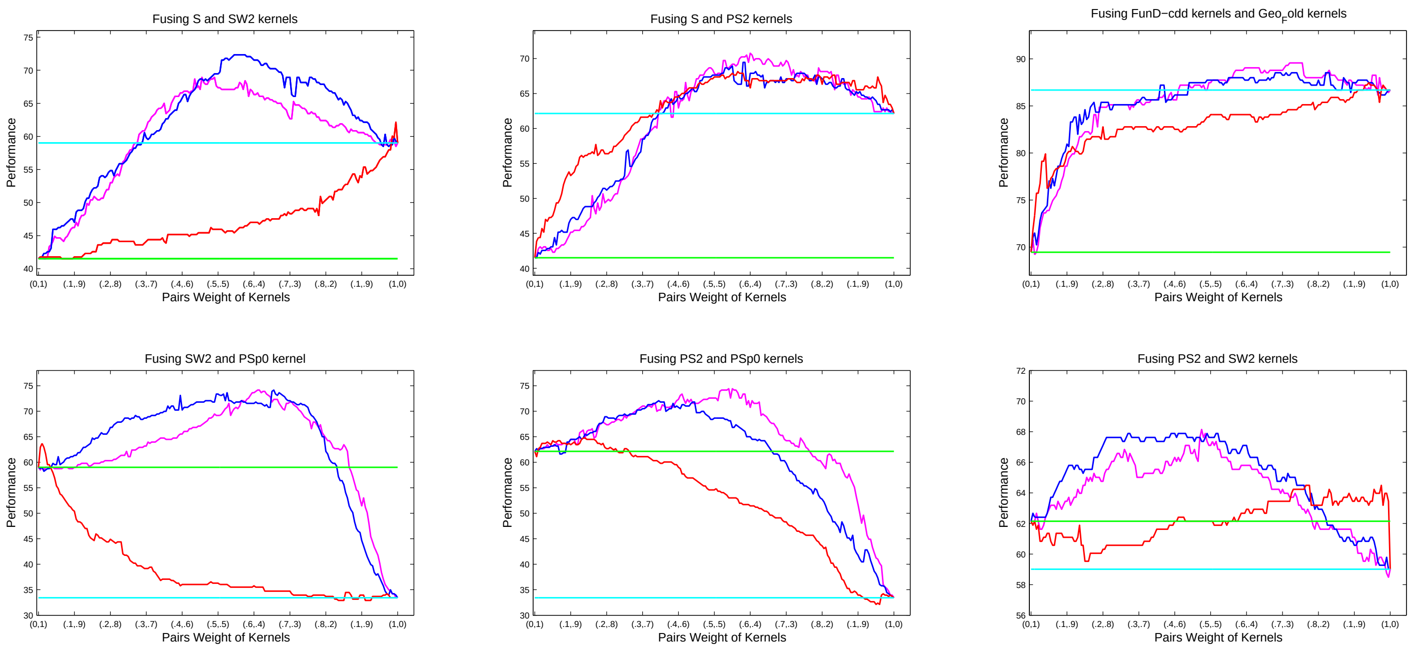

Fig. 2. The performance of convex linear combination of two different kernels using 201 different pairs weights of kernels (blue line). The relative performances of fused kernels through weighted Log-Euclidean mean (red line). The relative performances of fused kernels using weighted geometric mean (magenta line).

results on the SCOP PDB-40D benchmark dataset demonstrate that the geometric-based averaging of kernel matrices can effectively improve the accuracy of the state-of-the-art kernel fusion model.

Data fusion using kernel-based kNN

In contrast with the conventional k nearest-neighbor (kNN) approach, which uses a norm distance between training and test samples in the original space, the distance for kernel-based kNN is computed in the feature space. It has been shown that the Euclidean distance in the kernel feature space $\phi(x)$ can be defined by inner products between mapped samples as in the following formula:

$$\|\phi(x) - \phi(y)\|^2 = K(x,x) + K(y,y) - 2K(x,y). \tag{9}$$

Furthermore, for integrating kernel matrices using kernel-based kNN the combined kernel is used to determine the Euclidean distance between training and test samples in the feature space. Supplemental Table 4 lists the results (on the SCOP PDB-40D benchmark dataset)

Table 4. The performace of data fusion kernel-based kNN using different combined kernel matrices on the SCOP PDB-40D benchmark dataset

k	Arithmetic Mean	Harmonic Mean	LogE-KF	AGH-KF	Karcher-KF
k=1	49.68	46.48	53.26	81.46	82.77
k=2	49.87	46.47	53.26	81.46	82.77
k=3	44.65	44.91	48.56	81.46	81.2
k=4	42.82	42.82	43.60	80.94	81.2
k=5	40.47	39.43	38.38	81.98	80.42
k=6	38.90	39.67	36.55	79.11	79.37
k=7	37.34	39.95	35.25	78.33	78.59
k=8	36.29	39.67	34.20	78.33	78.51
k=9	35.77	40.73	32.38	77.81	78.51
k=10	34.73	39.43	33.68	78.07	76.76

Table 5. Performance with Individual and integrated string kernels. ROC, ROC50 and median RFP (mRFP) scores over 54 families

Method	ROC	ROC50	mRFP	description
LA-eig	0.908	0.591	0.0654	string kernel, beta=infinity [Saigo *et al.*, 2004]
LA-eig	0.908	0.597	0.0679	string kernel, beta=0.8 [Saigo *et al.*, 2004]
LA-eig	0.925	0.649	0.0541	string kernel, beta=0.5 [Saigo *et al.*, 2004]
LA-eig	0.923	0.661	0.0637	string kernel, beta=0.2 [Saigo *et al.*, 2004]
Mismatch(5:1)	0.872	0.400	0.0837	string kernel, [Saigo *et al.*, 2004]
AGH-KF	0.937	0.687	0.0441	combining all above string kernels except Mismatch kernel
LogE-KF	0.916	0.54	0.0661	combining all above string kernels except Mismatch kernel
AGH-KF	0.938	0.695	0.0442	combining all above string kernels
LogE-KF	0.933	0.691	0.0491	combining all above string kernels

from the kernel-based equivalent of kNN for different values of k using different combined kernel matrices. These matrices are obtained by taking various types of means of the twenty six kernel matrices based on various protein features listed in Supplemental Table 2. According to Table 2 and Supplemental Table 4, the performance of SVM exceeds the results of kernel-based kNN.

Remote homology detection using geometric kernel data fusion

We investigate the performance of our geometric kernel fusion approach on the protein remote homology detection problem as it was illustrated in [Liao and Noble, 2003, Saigo *et al.*, 2004] by fusing multiple kernels. Supplemental Table 5 reports the results of the combined kernels obtained by taking geometric and Log-Euclediean means of the various string kernel matrices. It also lists the performance of the SVM classification method based on different string kernels used in our proposed kernel fusion frameworks. Experimental results on the famous Nobel dataset[Liao and Noble, 2003] demonstrate that our geometric fusion approach can effectively improve the accuracy of the state-of-the-art string kernels for this problem.

REFERENCES

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, **2**(3), 27:1–27:27.

Chou, K.-C. (2001). Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, **43**(3), 246–255.

Chou, K.-C. and Shen, H.-B. (2007). Recent progress in protein subcellular location prediction. *Analytical Biochemistry*, **370**(1), 1 – 16.

Cristianini, N., Kandola, J., Elisseeff, A., and Shawe-Taylor, J. (2002). On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press.

De Bie, T., Tranchevent, L.-C., van Oeffelen, L. M. M., and Moreau, Y. (2007). Kernel-based data fusion for gene prioritization. *Bioinformatics*, **23**(13), i125–i132.

Liao, L. and Noble, W. S. (2003). Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computional Biology*, **10**(2), 857–868.

Qiu, S. and Lane, T. (2009). A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, **6**(2), 190–199.

Quevillon, E., Silventoinen, V., Pillai, S., Harte, N., Mulder, N., Apweiler, R., and Lopez, R. (2005). Interproscan: protein domains identifier. *Nucleic Acids Research*, **33**(suppl 2), W116–W120.

Saigo, H., Vert, J.-P., Ueda, N., and Akutsu, T. (2004). Protein homology detection using string alignment kernels. *Bioinformatics*, **20**(11), 1682–1689.

Schaffer, A. A., Aravind, L., Madden, T. L., Shavirin, S., Spouge, J. L., Wolf, Y. I., Koonin, E. V., and Altschul, S. F. (2001). Improving the accuracy of psi-blast protein database searches with composition-based statistics and other refinements. *Nucleic Acids Research*, **29**(14), 2994–3005.

Shen, H.-B. and Chou, K.-C. (2009). Predicting protein fold pattern with functional domain and sequential evolution information. *Journal of Theoretical Biology*, **256**(3), 441 – 446.

Zakeri, P., Moshiri, B., and Sadeghi, M. (2011). Prediction of protein submitochondria locations based on data fusion of various features of sequences. *Journal of Theoretical Biology*, **269**(1), 208 – 216.