Pattern Recognition Letters 31 (2010) 959–965

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Cost-conscious multiple kernel learning

Mehmet Gönen *, Ethem Alpaydın

Department of Computer Engineering, Boğaziçi University, TR-34342 Bebek, İstanbul, Turkey

ARTICLE INFO

Article history:
Received 30 April 2009
Received in revised form 1 November 2009
Available online 4 January 2010

Communicated by Y. Ma

Keywords:
Support vector machines
Kernel combination
Multiple kernel learning

ABSTRACT

Recently, it has been proposed to combine multiple kernels using a weighted linear sum. In certain applications, different kernels may be using different input representations and these methods do not consider neither the cost of acquiring them nor the cost of evaluating the kernels. We generalize the framework of Multiple Kernel Learning (Mkl) for this cost-conscious methodology. On 12 benchmark data sets from the UCI repository, we compare Mkl and its cost-conscious variants in terms of accuracy, support vector count, and total cost. Cost-conscious Mkl achieves statistically similar accuracy results by using fewer support vectors/kernels by best trading off accuracy brought by each representation/kernel with the concomitant cost. We also test our approach on two popular bioinformatics data sets from MIPS comprehensive yeast genome database (CYGD) and see that integrating the cost factor into kernel combination allows us to obtain cheaper kernel combinations by using fewer active kernels and/or support vectors.

1. Introduction

Support Vector Machine (Svm) is a discriminative classifier proposed for binary classification problems and is based on the theory of structural risk minimization (Vapnik, 1995). Given a sample of N independent and identically distributed training instances:

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N),$$

where $\boldsymbol{x}_i$ is the D-dimensional input vector and $y_i \in \{-1, +1\}$ is its class label, $i = 1, \ldots, N$, Svm basically finds the linear discriminant with the maximum margin in the feature space induced by the mapping function Φ. The resulting decision function is:

$$f(\boldsymbol{x}) = \operatorname{sgn}(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b).$$

The classifier can be trained by solving the following quadratic optimization problem:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \\ &\text{with respect to } \boldsymbol{w} \in \mathbb{R}^D, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \\ &\quad \text{subject to } y_i(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i) \rangle + b) \geqslant 1 - \xi_i \quad \forall i, \end{aligned}$$

where $\boldsymbol{w}$ is the vector of weight coefficients, C is a predefined positive trade-off parameter between model simplicity and classification error, $\boldsymbol{\xi}$ is the vector of slack variables, and b is the bias term of the separating hyperplane. Instead of solving this optimization problem directly, the Lagrangian dual function enables us to obtain the following dual formulation:

$$\begin{aligned} &\text{maximize } \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \overbrace{\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle}^{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}, \\ &\text{with respect to } \boldsymbol{\alpha} \in \mathbb{R}_+^N, \\ &\quad \text{subject to } \sum_{i=1}^{N}\alpha_i y_i = 0, \\ &\qquad C \geqslant \alpha_i \geqslant 0 \qquad \forall i, \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha}$ is the vector of dual variables corresponding to each separation constraint and the obtained kernel matrix is positive semi-definite. Solving this, we get $\boldsymbol{w} = \sum_{i=1}^{N}\alpha_i y_i \Phi(\boldsymbol{x}_i)$ and the decision function can be written as:

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{N}\alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right).$$

There are several kernel functions successfully used in the literature such as linear kernel (k_L), polynomial kernel (k_P), and Gaussian kernel (k_G):

$$\begin{aligned} k_L(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle, \\ k_P(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^q \quad q \in \mathbb{N}, \\ k_G(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/s^2\right) \quad s \in \mathbb{R}_{++}. \end{aligned}$$

There are also kernel functions proposed for particular applications, such as natural language processing (Lodhi et al., 2002) and bioinformatics (Schölkopf et al., 2004).

* Corresponding author. Tel.: +90 212 359 7183; fax: +90 212 287 2461.
E-mail addresses: gonen@boun.edu.tr (M. Gönen), alpaydin@boun.edu.tr (E. Alpaydın).

0167-8655/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2009.12.027

Selecting the kernel function and its parameters (e.g., q and s) is an important issue in SVM training. Kernel selection is generally done by selecting the best kernel function after applying a cross-validation procedure. In recent years, kernel combination methods have been proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters. Each kernel function can capture a different type of similarity and using several instead of one will enable a better solution. The reasoning is similar to combining different classifiers: instead of choosing a single kernel function and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking. The cost-conscious approach can similarly be justified: We add a learner to an ensemble if its contribution to accuracy is worth the increase in cost (time/space complexity of the learner and/or the cost of acquiring the input representation) (Demir and Alpaydın, 2005). Similarly, we include a kernel if it is worth the extra cost of computation or input representation.

In this paper, we demonstrate the effect of cost-conscious methodology in kernel combination on several benchmark data sets. The paper is organized as follows: Different kernel combination methods are reviewed in Section 2. Section 3 explains how cost-conscious kernel combination can be performed by reinterpreting the multiple kernel formulation of Bach et al. (2004). Experiments and results obtained are given in Section 4 and we conclude in Section 5.

2. Kernel combination methods

The kernel matrix in the objective function must be positive semi-definite to solve the optimization problem in Eq. (1) efficiently. In this case, the mathematical model becomes a convex optimization problem and there are several methods that can be used to obtain the global optimum. Simple rules, such as scaling with a positive number, summation, and multiplication, as summarized in Eq. (2) allow us to obtain new kernel functions from existing ones; they ensure that the combined kernel has also a positive semi-definite kernel matrix (Cristianini and Shawe-Taylor, 2000):

$$\begin{aligned} k(\boldsymbol{x}_i,\boldsymbol{x}_j) &= a k_1(\boldsymbol{x}_i,\boldsymbol{x}_j) \quad a \in \mathbb{R}_+, \\ k(\boldsymbol{x}_i,\boldsymbol{x}_j) &= k_1(\boldsymbol{x}_i,\boldsymbol{x}_j) + k_2(\boldsymbol{x}_i,\boldsymbol{x}_j), \\ k(\boldsymbol{x}_i,\boldsymbol{x}_j) &= k_1(\boldsymbol{x}_i,\boldsymbol{x}_j) k_2(\boldsymbol{x}_i,\boldsymbol{x}_j). \end{aligned} \tag{2}$$

Summation rule is applied successfully to computational biology (Pavlidis et al., 2001) and optical digit recognition (Moguerza et al., 2004) where heterogeneous data sets exist by the nature of these problems. In both works, summation of two or more kernels obtained from different representations of the same data set is used.

Such simple rules make assumptions about the combination rule before training. For example, unweighted summation of kernel functions gives equal importance to each kernel. Lanckriet et al. (2002, 2004a) replace the kernel function in the objective and decision functions with a linear combination of kernels. The new objective function is:

$$\underset{\boldsymbol{\eta}}{\text{minimize}} \ \underset{\boldsymbol{\alpha}}{\text{maximize}} \ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i,\boldsymbol{x}_j),$$

where m indexes kernels, P is their number, each k_m corresponds to a different kernel function, and η_m is the combination weight of k_m. The new decision function becomes:

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i,\boldsymbol{x}) + b \right).$$

Lanckriet et al. (2002, 2004a) represent this problem using a semi-definite programming (SDP) formulation by constraining the combined kernel matrix to be positive semi-definite. Lanckriet et al. (2004a) simplify further the model into a quadratically constrained quadratic programming (QCQP) problem by considering only the nonnegative combination weights. The QCQP formulation can be solved for both the support vector coefficients ($\boldsymbol{\alpha}$) and the combination weights ($\boldsymbol{\eta}$), whereas the SDP formulation gives only the combination weights and requires solving a canonical SVM problem to find the support vector coefficients. The QCQP formulation has been tested on genomic data fusion with different kernel functions evaluated on different data representations (Lanckriet et al., 2004b).

The primal optimization problem can be modified without directly changing the dual optimization problem (Bach et al., 2004):

$$\begin{aligned} &\text{minimize } \frac{1}{2} \left(\sum_{m=1}^{P} d_m \|\boldsymbol{w}_m\| \right)^2 + C \sum_{i=1}^{N} \xi_i, \\ &\text{with respect to } \boldsymbol{w}_m \in \mathbb{R}^{D_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \\ &\text{subject to } y_i \left(\sum_{m=1}^{P} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b \right) \geqslant 1 - \xi_i \quad \forall i, \end{aligned}$$

where d_m is the nonnegative weight assigned to the feature space induced by Φ_m, $\boldsymbol{w}_m$ is the vector of weight coefficients, and D_m is the dimensionality of the corresponding feature space. This model can be treated as a second-order cone program and the resulting dual problem is:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i \\ &\text{with respect to } \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}_+^N, \\ &\text{subject to } d_m \gamma \geqslant \left\| \sum_{i=1}^{N} \alpha_i y_i \Phi_m(\boldsymbol{x}_i) \right\| \quad \forall m, \\ &\sum_{i=1}^{N} \alpha_i y_i = 0, \\ &C \geqslant \alpha_i \geqslant 0 \quad \forall i. \end{aligned} \tag{3}$$

This formulation has nonlinear constraints different from the original dual problem and is a case of conic programming. It is difficult to solve this model because of the high space and time complexity. Squaring each side of the first set of constraints in Eq. (3) and replacing γ^2 with 2γ reveal the following QCQP problem:

$$\begin{aligned} &\text{minimize } \gamma - \sum_{i=1}^{N} \alpha_i, \\ &\text{with respect to } \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}_+^N, \\ &\text{subject to } d_m^2 \gamma \geqslant \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \overbrace{\langle \Phi_m(\boldsymbol{x}_i), \Phi_m(\boldsymbol{x}_j) \rangle}^{k_m(\boldsymbol{x}_i,\boldsymbol{x}_j)} \quad \forall m, \\ &\sum_{i=1}^{N} \alpha_i y_i = 0, \\ &C \geqslant \alpha_i \geqslant 0 \quad \forall i. \end{aligned}$$

Sonnenburg et al. (2006) develop a semi-infinite linear programming equivalent to this model and a solution method to solve the model iteratively, instead of solving it as a QCQP problem.

3. Cost-conscious kernel combination

We use a cost-conscious variant of the mathematical model developed in Bach et al. (2004). In general, d_m are treated as scaling factors to balance the scale differences between kernel function outputs and all d_m are selected as 1, because kernel outputs are already normalized before training:

$$\bar{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_i) k(\boldsymbol{x}_j, \boldsymbol{x}_j)}}. \quad (4)$$

The solution obtained from the optimization problem gives us a linear combination of kernel functions that satisfies:

$$\sum_{m=1}^{P} d_m^2 \eta_m = 1,$$

obtained from Karush–Kuhn–Tucker optimality conditions (Bach et al., 2004). This equation defines upper bounds for η_m and we can control the feasible region for η_m by changing d_m.

We use the same mathematical model but our interpretation for d_m is different. We think d_m as the cost coefficient for using k_m. There are two possible cases:

(a) We can combine different kernel functions and d_m may be considered as the cost of evaluating a kernel. For example, evaluating the Gaussian kernel function is more costly than evaluating the linear kernel. Here, the kernel cost is generally expressed in terms of the required processor time.
(b) We can combine different representations or modalities and d_m may be considered as the cost of extracting/sensing the corresponding representation/signal. Each data representation has its own data acquisition cost and kernel function evaluation time due to its different dimensionality. Kernel combination should favor the cheaper and smaller data representations if they are sufficient for accurate classification. If a particular data representation is not selected (i.e., its η is 0) after the training phase, we are not required to collect and prepare data for this representation in the testing phase. So, by assigning higher costs, we can eliminate some of the data representations and therefore decrease the total cost and time for test examples, unless the costly kernels/representations are absolutely necessary for accuracy. For example, in speech recognition where additional to the usual acoustic input, if we also use visual lip image as another source, we need to make sure that its contribution to accuracy is worth the cost of acquiring and processing the image. Or in biometrics where we have multiple modalities (face, fingerprint, iris, signature), we only want to include those whose costs can be justified in terms of additional accuracy.

From this perspective, setting all weights equal to 1 corresponds to assuming equal costs for all kernels, which, in general, is not true. We need a measure to estimate the total cost for the testing phase based on the number of support vectors and the kernel functions selected in training, which we define as:

$$c = \underbrace{\sum_{i=1}^{N} 1(\alpha_i > 0)}_{\substack{\#\text{ of support} \\ \text{vectors}}} \underbrace{\sum_{m=1}^{P} 1(\eta_m > 0) \frac{d_m}{\sum_{l=1}^{P} d_l}}_{\substack{\text{the total normalized cost} \\ \text{for active kernels}}}$$

where we multiply the number of support vectors and the summation of the normalized costs for active kernels.

4. Experiments

In our experiments, we use the MOSEK optimization package (Mosek, 2009) to solve QCQP problems. MOSEK enables us to obtain the support vector coefficients ($\boldsymbol{\alpha}$) and the kernel combination weights ($\boldsymbol{\eta}$) from the primal–dual solution found.

Our experimental methodology is as follows: given a data set, a random one-third is reserved as the test set and the remaining two-thirds is resampled using 5×2 cross-validation to generate 10 training and validation sets, with stratification. The validation sets of all folds are used to optimize C (by trying all values between 10^{-4} and 10^{+4} in log scale). The best configuration (the one that has the highest average accuracy on the validation folds) is used to train the final learners on the training folds and their performance is measured on the test set. So, for each data set, we have ten test set results.

In the result tables, we report the average accuracy percentage, support vector percentage, total cost, and normalized combination weights. The average accuracy, support vector percentage, and total cost are made bold if the difference between MKL and cost-conscious variant is significant using the 5×2 cv paired F test (Alpaydın, 1999). Total cost calculation is performed with the cost coefficients of cost-conscious variant to get comparable results. We also report the count of (W)ins–(T)ies–(L)osses of kernel combination with the cost-conscious MKL from direct comparison and the 5×2 cv paired F test. The Wilcoxon signed rank test is used to compare the two variants over a number of data sets in terms of average accuracies, support vector counts, total cost, and the result is shown as (W)in, (T)ie, or (L)oss.

4.1. Kernel selection on UCI data sets

We perform the experiments on 12 benchmark data sets (BANANA, HEART, IONOSPHERE, LIVERDISORDER, OPTDIGITS, PENDIGITS, PIMA, RINGNORM, SONAR, SPAMBASE, TWONORM, and WDBC) from the UCI repository (Asuncion and Newman, 2007). OPTDIGITS and PENDIGITS data sets are optical and pen-based digit recognition problems, respectively. Two-class subsets of these data sets (1VS8 and 3VS9 for OPDIGITS, 0VS8, 1VS7, and 5VS9 for PENDIGITS) are taken to obtain binary classification problems.

Three different kernel functions are used in this part: linear kernel (k_L), polynomial kernel (k_P), and Gaussian kernel (k_G). We use the second degree ($q = 2$) polynomial kernel and estimate s in the Gaussian kernel as the average nearest neighbor distance between instances in the training set. Outputs obtained from each kernel function are in different scales. In order to avoid this scale problem, kernel outputs are normalized by using Eq. (4). We experiment two scenarios on (k_L-k_P-k_G): in the first one, all kernels have equal cost ($d_L = 1.00$, $d_P = 1.00$, $d_G = 1.00$) and in the second, their cost increase as we go from linear to polynomial to Gaussian ($d_L = 1.00$, $d_P = \sqrt{2} = 1.41$, $d_G = 2.00$). The polynomial kernel is slightly more costly than the linear kernel because of taking the second power and the Gaussian kernel is more complicated than that because of the exp function. These values we use are rough estimates; exact values depend on the particular hardware implementation.

On all 12 data sets, the comparison of results by MKL and cost-conscious MKL is given in Table 1. We can see that PENDIGITS (5VS9), PIMA, and WDBC use only the linear kernel when we increase d_P and d_G and achieve statistically similar accuracy results. Removing the polynomial and the Gaussian kernel from the ensemble by penalizing them decreases total cost significantly for PENDIGITS (5VS9) and WDBC. The cost-conscious MKL assigns zero weight to the Gaussian kernel for OPTDIGITS (1VS8 and 3VS9), SONAR, and SPAMBASE data sets whereas the equal cost variant uses the Gaussian kernel with weights larger than 0.30. Nonlinear data sets such as BANANA and RINGNORM continue using the Gaussian kernel even if its cost is increased to 2. However the average test accuracy in RINGNORM data set is reduced drastically after changing the combination weight of the Gaussian kernel from 1.00 to 0.25. By comparing two variants over all data sets, we see that the cost-conscious MKL achieves similar accuracy results according to both the 5×2 cv paired F test (12 ties out of 15 tasks) and the Wilcoxon signed rank test. The 5×2 cv paired F test reports that total cost is decreased significantly over eight out of 15 tasks. Reduction in total

Table 1
The average accuracy percentage, support vector percentage, and total cost for (k_L-k_P-k_G) combination on benchmark data sets. The first line is the case where all kernels have equal cost and the second line is where complex kernels are penalized.

Data set	Test accuracy	SV	Total cost	η_L-η_P-η_G
Banana	83.17 ± 0.52	90.68 ± 0.11	70.14 ± 0.08	0.00–0.03–0.97
	83.27 ± 1.05	83.27 ± 0.11	64.40 ± 0.08	0.00–0.04–0.96
Heart	79.67 ± 1.74	79.44 ± 3.28	18.00 ± 0.74	1.00–0.00–0.00
	79.67 ± 1.74	79.44 ± 3.28	18.00 ± 0.74	1.00–0.00–0.00
Ionosphere	93.85 ± 1.13	64.36 ± 2.73	54.25 ± 7.54	0.01–0.46–0.53
	90.94 ± 1.57	**46.67 ± 2.52**	44.65 ± 7.54	0.29–0.60–0.10
Liverdisorder	64.17 ± 4.14	93.30 ± 1.54	57.01 ± 20.32	0.02–0.00–0.98
	65.91 ± 3.17	**80.87 ± 3.60**	66.57 ± 20.32	0.67–0.30–0.03
Optdigits (1vs8)	98.01 ± 0.14	28.21 ± 2.68	28.21 ± 1.00	0.27–0.38–0.34
	98.01 ± 0.14	**19.12 ± 1.83**	**10.46 ± 1.00**	0.53–0.47–0.00
Optdigits (3vs9)	96.71 ± 1.53	25.46 ± 4.19	25.46 ± 1.15	0.44–0.20–0.36
	97.37 ± 1.39	**15.25 ± 2.10**	**8.34 ± 1.15**	0.73–0.27–0.00
Pendigits (0vs8)	99.30 ± 0.11	11.70 ± 0.53	11.70 ± 0.11	0.18–0.71–0.11
	99.80 ± 0.00	**9.10 ± 0.11**	**9.10 ± 0.11**	0.25–0.66–0.09
Pendigits (1vs7)	100.00 ± 0.00	14.00 ± 1.69	14.00 ± 1.58	0.39–0.24–0.37
	100.00 ± 0.00	**6.50 ± 1.58**	**6.50 ± 1.58**	0.68–0.23–0.09
Pendigits (5vs9)	99.20 ± 0.21	33.20 ± 0.21	18.16 ± 0.05	0.79–0.21–0.00
	99.20 ± 0.21	32.60 ± 0.21	**7.39 ± 0.05**	1.00–0.00–0.00
Pima	73.09 ± 0.75	68.91 ± 1.62	20.14 ± 0.37	0.97–0.03–0.00
	72.93 ± 1.01	68.79 ± 1.65	15.58 ± 0.37	1.00–0.00–0.00
Ringnorm	**98.01 ± 0.63**	**26.85 ± 0.00**	**12.17 ± 0.79**	0.00–0.00–1.00
	87.25 ± 0.42	82.46 ± 1.16	56.05 ± 0.79	0.75–0.00–0.25
Sonar	81.14 ± 3.14	86.96 ± 2.46	86.96 ± 3.10	0.14–0.33–0.54
	80.29 ± 3.61	**69.13 ± 5.68**	**37.81 ± 3.10**	0.48–0.52–0.00
Spambase	92.53 ± 0.10	42.99 ± 4.33	42.99 ± 6.23	0.39–0.18–0.43
	92.43 ± 0.00	**31.06 ± 2.53**	**12.40 ± 6.23**	0.96–0.04–0.00
Twonorm	97.20 ± 0.00	89.10 ± 0.11	20.18 ± 0.05	1.00–0.00–0.00
	97.20 ± 0.00	89.00 ± 0.21	20.16 ± 0.05	1.00–0.00–0.00
Wdbc	95.45 ± 0.91	22.26 ± 3.21	16.97 ± 0.36	0.70–0.06–0.24
	94.97 ± 0.97	17.00 ± 1.59	**3.85 ± 0.36**	1.00–0.00–0.00
(W-T-L)	2-12-1	8-6-1	8-6-1	5×2 cv paired F test
(W-T-L)	4-5-6	13-1-1	12-1-2	Direct comparison
(W/T/L)	T	W	W	Wilcoxon rank test

cost is also reported to be statistically significant using the Wilcoxon signed rank test over 15 tasks.

4.2. Representation selection on handwritten digit data

Kernel combination can also be used to combine different data representations or modalities. In this case, it can be the case that extracting different representations or modalities may have costs associated with them and we do not want to use a costly one unless it is deemed necessary for classification. For example, Pendigits data set has four different representations: Dyn, Sta4, Sta8, and Sta16 (Alimoğlu and Alpaydın, 1997). Dyn contains eight successive pen points on two-dimensional coordinate system and is used when combining kernel functions on Pendigits data set. Sta16 is 16×16 image bitmap representation of the corresponding training instance formed by connecting the points in the Dyn representation by line segments. Sta4 and Sta8 are 4×4 and 8×8 subsampled bitmap representations of Sta16, respectively. Fig. 1 illustrates Dyn, Sta16, Sta8, and Sta4 representations on a sample data instance.

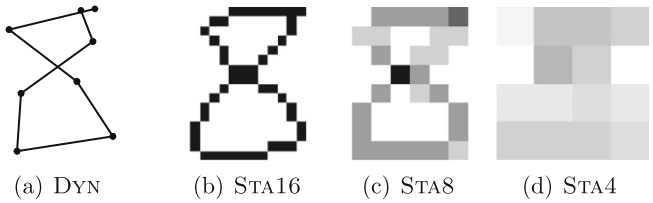

Fig. 1. Four different representations for digit eight.

In our experiments, we use linear kernels over four different representations of Pendigits data set. We form two sets of experiments as follows: (a) (k_{Dyn}-k_{Sta16}) with (d_{Dyn}-d_{Sta16}) taken as (1.00–1.00) and (1.00–4.00), increasing the cost of forming the image representation, (b) (k_{Dyn}-k_{Sta4}-k_{Sta8}-k_{Sta16}) with (d_{Dyn}-d_{Sta4}-d_{Sta8}-d_{Sta16}) taken as (1.00–1.00–1.00–1.00) and (1.00–1.00–2.00–4.00), increasing the cost of representation with higher dimensionalities.

As we see in Table 2, when both representations, Dyn and Sta16, have equal cost, both are chosen for 0vs8 and 1vs7. When we increase the cost of the Sta16 to 4, only Dyn representation is used

Table 2
The average accuracy percentage, support vector percentage, total cost, and kernel combination weights for (Dyn-Sta16) combination.

Data set	Test accuracy	SV	Total cost	η_{Dyn}-η_{Sta16}
Pendigits (0vs8)	99.40 ± 0.21	15.10 ± 0.95	15.10 ± 0.21	0.21–0.79
	99.00 ± 0.42	**8.80 ± 0.21**	**8.80 ± 0.21**	0.75–0.25
Pendigits (1vs7)	99.70 ± 0.32	11.00 ± 0.21	11.00 ± 0.02	0.35–0.65
	100.00 ± 0.00	**3.70 ± 0.11**	**0.74 ± 0.02**	1.00–0.00
Pendigits (5vs9)	99.20 ± 0.21	32.80 ± 0.21	6.56 ± 0.08	1.00–0.00
	99.20 ± 0.21	32.80 ± 0.42	6.56 ± 0.08	1.00–0.00

for 1vs7. The average accuracy percentage for all three tasks remains the same according to the 5×2 cv paired F test results with increasing d_{STA16}. Total cost in testing phase decreases because of two factors: (a) the average support vector percentages are decreased. (b) k_{STA16} is not evaluated when η_{STA16} is equal to 0. When we combine all four representations and penalize kernels proportional to their dimensionality (see Table 3), we obtain accuracy results that are statistically the same, but using only two representations (DYN and STA4) for 5vs9 and three representations (DYN, STA4, and STA8) for 0vs8 and 1vs7. The cost-conscious MKL also uses significantly fewer support vectors in all cases. Estimated total cost for both cases decrease drastically as a result of using fewer kernels and support vectors.

4.3. Kernel/representation selection on bioinformatics data

We perform protein location prediction and protein function prediction experiments on the MIPS comprehensive yeast genome database (Mewes et al., 2000).

4.3.1. Protein location prediction

CYGD assigns subcellular locations for 2318 and 1150 proteins according to whether they participate in the membrane and the ribosome, respectively. We combine the seven kernel functions used also in Lanckriet et al. (2004b) for comparing equal cost and cost-conscious variants of MKL.

k_{SW} and k_B are generated from protein sequences using Smith–Waterman (SW) and the BLAST pairwise sequence comparison algorithms, respectively. k_{Pfam} are also generated from protein sequences by replacing pairwise comparison scores with the expectation values obtained from hidden Markov models in the Pfam database. k_{FFT} is calculated by comparing the frequency content of the hydropathy profiles of the two proteins. k_{LI} is the inner product between interaction values for a pair of proteins. k_D is the diffusion kernel calculated over the graph constructed by using the same interaction data used in k_{LI}. k_E is the Gaussian kernel calculated over microarray gene expression measurements.

k_{SW}, k_B, and k_{Pfam} require pairwise comparison scores for protein sequences, so, they are computationally expensive. The same concern is also valid for k_{FFT}. k_{LI} obtained by inner product over protein interactions is a simple kernel. k_D requires constructing a graph from protein interactions and calculating a similarity measure based on a random walk on this graph. k_E is also a cheap kernel, which simply evaluates the Gaussian kernel function over 441 dimensional gene expressions.

Two different cost combinations are formed by considering all seven kernels despite the fact that k_E and k_{FFT} are not much relevant for membrane and ribosomal protein recognition tasks, respectively. We consider the following two (d_{SW}-d_B-d_{Pfam}-d_{FFT}-d_{LI}-d_D-d_E) combinations: (a) (1.00–1.00–1.00–1.00–1.00–1.00–1.00), all kernels are considered with equal cost coefficients as a base case, (b) (1.41–1.41–1.41–1.41–1.00–2.00–1.00), the diffusion kernel (k_D) is assigned the largest cost coefficient due to its computational complexity, k_{LI} and k_E are given cost coefficients smaller than those of sequence based kernels (k_{SW}-k_B-k_{Pfam}-k_{FFT}) due to their simplicity. The cost values we assign here are used as rough estimates to give us an ordering of the costs. Exact values depend on the implementation of these kernel functions and the time/space complexity of the data structures and the algorithms that are used.

Table 4 summarizes the results for ribosomal and membrane protein recognition problem. Cost-conscious variant uses statistically significantly fewer support vectors compared to the equal cost variant in membrane protein recognition. As an important result, it can be observed that discarding k_D (i.e., $d_D = 2$), which is computationally expensive, and using protein sequence based kernels with a larger cost coefficient than k_{LI} achieves similar classification results by improving total cost for both tasks.

4.3.2. Protein function prediction

CYGD categorizes 3588 proteins into 13 top-level categories, which can be interpreted as 13 binary classification tasks (YEAST1, YEAST2, ..., YEAST13), one for each category. The reason for decomposing into binary classification problems instead of using a multiclass formulation is that some proteins belong to more than one category. In this set of experiments, we use the eight kernel functions used also in Lanckriet et al. (2004a).

k_{Pfam} is the inner product between binary representations of protein sequences to Pfam domains. k_{PfamE} is an enriched variant of k_{Pfam} obtained by using additional domains. k_{TAP}, k_{Phys}, and k_{Gen} are three different diffusion kernels calculated over the graphs constructed from three different types of protein interactions: co-participation in a protein complex, genetic interactions, and protein–protein interactions, respectively. k_{Exp} and k_{ExpG} are calculated from cell cycle gene expression measurements. k_{Exp} is a binary kernel function that is determined by using Pearson correlation of a pair of expression profiles. k_{ExpG} is the Gaussian ker-

Table 3
The average accuracy percentage, support vector percentage, total cost, and kernel combination weights for (DYN-STA4-STA8-STA16) combination.

Data set	Test accuracy	SV	Total cost	η_{DYN}-η_{STA4}-η_{STA8}-η_{STA16}
PENDIGITS (0vs8)	99.50 ± 0.11	10.70 ± 0.11	10.70 ± 0.33	0.24–0.16–0.33–0.26
	99.30 ± 0.32	**5.80 ± 0.42**	**2.90 ± 0.21**	0.48–0.38–0.14–0.00
PENDIGITS (1vs7)	99.90 ± 0.11	9.70 ± 0.53	9.70 ± 1.33	0.32–0.31–0.31–0.06
	99.70 ± 0.32	**3.00 ± 0.42**	**1.50 ± 0.21**	0.45–0.51–0.04–0.00
PENDIGITS (5vs9)	99.30 ± 0.11	12.90 ± 0.95	12.90 ± 0.04	0.35–0.12–0.10–0.44
	99.50 ± 0.11	**10.30 ± 0.11**	**2.58 ± 0.03**	0.64–0.36–0.00–0.00

Table 4
The average accuracy percentage, support vector percentage, total cost, and kernel combinations weights for membrane and ribosomal protein recognition tasks.

Task	Test accuracy	SV	Total cost	η_{SW}-η_B-η_{Pfam}-η_{FFT}-η_{LI}-η_D-η_E
MEMBRANE	86.30 ± 0.61	84.42 ± 1.03	75.34 ± 5.09	0.28–0.31–0.11–0.05–0.00–0.15–0.10
	85.40 ± 0.55	**71.33 ± 1.67**	**37.81 ± 5.09**	0.00–0.26–0.06–0.00–0.07–0.00–0.60
RIBOSOMAL	98.99 ± 0.26	18.56 ± 1.56	6.86 ± 1.96	0.01–0.00–0.00–0.16–0.03–0.00–0.80
	99.07 ± 0.18	18.61 ± 1.46	6.08 ± 1.96	0.00–0.00–0.00–0.01–0.03–0.00–0.96

Table 5
The average accuracy percentage, support vector percentage, total cost, and kernel combination weights for (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) combination.

Task	Test accuracy	SV	Total cost	η_{Pfam}-η_{TAP}-η_{Phys}-η_{Gen}- η_{Exp}
Yeast1	**78.20 ± 0.84**	**91.08 ± 0.85**	91.08 ± 26.82	0.67–0.03–0.07–0.17–0.06
	72.70 ± 0.56	97.03 ± 0.59	50.96 ± 26.82	0.14–0.00–0.00–0.01–0.85
Yeast2	93.29 ± 0.10	**91.10 ± 2.33**	88.92 ± 0.08	0.23–0.01–0.14–0.18–0.44
	93.24 ± 0.06	96.54 ± 0.69	**11.47 ± 0.08**	0.00–0.00–0.00–0.00–1.00
Yeast3	**86.40 ± 0.46**	95.68 ± 0.86	95.68 ± 14.66	0.11–0.01–0.18–0.32–0.37
	83.96 ± 0.35	94.62 ± 1.43	**56.27 ± 14.66**	0.16–0.00–0.00–0.01–0.83
Yeast4	**84.81 ± 0.56**	93.17 ± 1.07	93.17 ± 18.68	0.22–0.03–0.23–0.21–0.31
	82.07 ± 0.49	94.48 ± 1.18	60.71 ± 18.68	0.16–0.01–0.01–0.00–0.82
Yeast5	91.51 ± 0.17	**75.42 ± 1.54**	75.42 ± 23.51	0.19–0.01–0.13–0.25–0.41
	91.50 ± 0.19	90.96 ± 1.77	60.76 ± 23.51	0.15–0.00–0.01–0.01–0.84
Yeast6	**86.75 ± 0.45**	91.74 ± 1.25	76.48 ± 29.43	0.63–0.00–0.13–0.23–0.01
	83.84 ± 0.66	95.46 ± 1.20	54.40 ± 29.43	0.16–0.00–0.00–0.01–0.82
Yeast7	**90.20 ± 0.41**	**83.09 ± 1.92**	75.09 ± 0.18	0.18–0.03–0.28–0.23–0.27
	87.01 ± 0.29	95.46 ± 0.63	**27.39 ± 0.18**	0.12–0.00–0.00–0.00–0.88
Yeast8	92.93 ± 0.16	91.67 ± 3.51	82.92 ± 0.23	0.34–0.01–0.10–0.14–0.42
	92.73 ± 0.16	95.91 ± 0.81	**27.52 ± 0.23**	0.10–0.00–0.00–0.00–0.90
Yeast9	94.59 ± 0.14	**84.71 ± 2.71**	84.71 ± 0.11	0.16–0.11–0.22–0.11–0.39
	94.57 ± 0.00	96.55 ± 0.94	**11.47 ± 0.11**	0.00–0.00–0.00–0.00–1.00
Yeast10	**89.97 ± 0.23**	94.88 ± 0.53	79.08 ± 17.56	0.09–0.00–0.24–0.23–0.44
	88.54 ± 0.48	92.54 ± 1.85	**44.18 ± 17.56**	0.15–0.00–0.02–0.00–0.83
Yeast11	94.66 ± 0.11	92.97 ± 1.92	75.30 ± 0.04	0.07–0.00–0.17–0.25–0.51
	94.65 ± 0.00	97.22 ± 0.38	**11.55 ± 0.04**	0.00–0.00–0.00–0.00–1.00
Yeast12	94.87 ± 0.35	75.80 ± 2.17	32.56 ± 0.87	0.99–0.01–0.00–0.00–0.00
	94.10 ± 0.62	83.02 ± 3.02	23.82 ± 0.87	0.59–0.00–0.00–0.00–0.41
Yeast13	97.74 ± 0.00	86.64 ± 8.66	85.43 ± 0.06	0.12–0.05–0.10–0.07–0.66
	97.74 ± 0.00	96.79 ± 0.54	**11.50 ± 0.06**	0.00–0.00–0.00–0.00–1.00
(W-T-L)	0-7-6	0-8-5	8-5-0	5×2 cv paired F test
(W-T-L)	0-1-12	2-0-11	13-0-0	Direct comparison
(W/T/L)	L	L	W	Wilcoxon signed rank test

Table 6
The average accuracy percentage, support vector percentage, total cost, and kernel combination weights for (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}) combination.

Task	Test accuracy	SV	Total cost	η_{PfamE}-η_{TAP}-η_{Phys}-η_{Gen}- η_{ExpG}-η_{SW}
Yeast1	79.15 ± 0.77	92.74 ± 0.84	92.74 ± 8.57	0.16–0.02–0.09–0.16–0.05–0.51
	78.15 ± 0.49	**86.26 ± 1.20**	**38.88 ± 8.57**	0.11–0.00–0.00–0.00–0.56–0.33
Yeast2	94.13 ± 0.29	70.44 ± 4.10	66.03 ± 3.99	0.01–0.01–0.16–0.21–0.09–0.52
	93.94 ± 0.49	**42.96 ± 3.49**	**13.78 ± 3.99**	0.00–0.00–0.00–0.00–0.63–0.36
Yeast3	86.40 ± 0.54	90.47 ± 1.07	86.85 ± 5.76	0.05–0.01–0.16–0.29–0.05–0.44
	85.03 ± 0.50	**66.71 ± 3.20**	**36.29 ± 5.76**	0.14–0.00–0.00–0.05–0.75–0.06
Yeast4	**85.64 ± 0.58**	84.27 ± 1.67	84.27 ± 0.37	0.17–0.03–0.20–0.18–0.06–0.37
	82.32 ± 0.76	**62.67 ± 0.82**	**28.15 ± 0.37**	0.34–0.01–0.00–0.00–0.65–0.00
Yeast5	93.12 ± 0.50	60.38 ± 2.86	60.38 ± 4.73	0.17–0.01–0.12–0.15–0.22–0.32
	94.54 ± 0.42	**49.11 ± 2.85**	**27.18 ± 4.73**	0.10–0.00–0.00–0.00–0.64–0.26
Yeast6	86.89 ± 0.37	84.23 ± 0.94	82.53 ± 0.62	0.14–0.01–0.12–0.24–0.04–0.47
	85.96 ± 0.85	**77.61 ± 1.59**	**30.23 ± 0.62**	0.10–0.00–0.00–0.00–0.50–0.40
Yeast7	**91.14 ± 0.46**	89.77 ± 1.19	87.96 ± 0.87	0.09–0.04–0.28–0.24–0.03–0.32
	88.07 ± 0.54	**75.24 ± 2.25**	**29.31 ± 0.87**	0.24–0.00–0.00–0.00–0.39–0.37
Yeast8	93.66 ± 0.19	83.07 ± 3.94	70.03 ± 3.43	0.00–0.04–0.18–0.22–0.03–0.52
	93.41 ± 0.31	**53.28 ± 1.69**	**15.33 ± 3.43**	0.00–0.00–0.00–0.00–0.65–0.34
Yeast9	94.67 ± 0.35	40.18 ± 5.08	39.40 ± 6.79	0.02–0.14–0.24–0.12–0.01–0.47
	94.72 ± 0.38	57.95 ± 3.32	**23.52 ± 6.79**	0.02–0.00–0.00–0.00–0.60–0.37
Yeast10	**90.09 ± 0.25**	70.98 ± 1.62	49.26 ± 6.70	0.00–0.00–0.23–0.19–0.02–0.57
	88.65 ± 0.09	71.62 ± 3.64	**25.36 ± 6.70**	0.00–0.00–0.01–0.00–0.61–0.38
Yeast11	94.65 ± 0.00	86.89 ± 2.20	58.53 ± 2.91	0.00–0.01–0.16–0.13–0.01–0.68
	94.65 ± 0.00	**58.95 ± 3.44**	**15.34 ± 2.91**	0.00–0.00–0.00–0.00–0.70–0.30
Yeast12	96.01 ± 0.24	54.92 ± 3.24	45.70 ± 2.61	0.39–0.11–0.07–0.02–0.02–0.37
	95.58 ± 0.59	**38.89 ± 1.58**	**15.94 ± 2.61**	0.34–0.00–0.00–0.00–0.46–0.20
Yeast13	97.68 ± 0.10	63.12 ± 5.14	54.04 ± 2.33	0.00–0.24–0.10–0.11–0.03–0.51
	97.74 ± 0.00	**30.47 ± 2.17**	**8.82 ± 2.33**	0.02–0.00–0.00–0.00–0.89–0.09
(W-T-L)	0-10-3	11-2-0	13-0-0	5×2 cv paired F test
(W-T-L)	3-1-9	11-0-2	13-0-0	Direct comparison
(W/T/L)	L	W	W	Wilcoxon signed rank test

nel defined on the expression profiles. k_{SW} is obtained by applying SW algorithm to yeast protein sequences.

Two different kernel subsets described in Lanckriet et al. (2004a) are also used for experiments: (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) and (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}). For the first subset, k_{Pfam} and diffusion kernels are assigned cost coefficients higher ($d_{Pfam} = 1.41$ and $d_{TAP} = d_{Phys} = d_{Gen} = 2.00$) than k_{Exp}($d_{Exp} = 1.00$). Likewise, k_{PfamE}, k_{SW}, and diffusion kernels are penalized ($d_{PfamE} = d_{SW} = 1.41$ and $d_{TAP} = d_{Phys} = d_{Gen} = 2.00$) in the second subset.

The results for each binary classification task in the first subset are given in Table 5. We see that avoiding the expensive kernels such as diffusion kernels leads to significant accuracy loss and increase in the number of support vectors stored. The cost-conscious MKL uses k_{Exp} only in four out of 13 tasks. In other tasks except YEAST12, k_{Exp} is used with a combination weight between 0.82 and 1. Support vector percentages are increased in nearly all cases but total kernel evaluation will be decreased for test instances due to the unused kernels. This result can also be seen from the estimated total cost results, which has an obvious decreasing trend for the cost-conscious variant. The 5×2 cv paired F test and the Wilcoxon signed rank test report significant win for total cost and significant loss for the average test accuracy and support vector percentages, indicating that in this case, the expensive kernels are worth their costs.

Table 6 lists the results for the second subset. Cost-conscious variant uses k_{ExpG} heavily with combination weights ranging from 0.39 to 0.89. Equal cost variant uses k_{SW} with a significant weight in all tasks. However cost-conscious variant prefers to use k_{SW} with a smaller coefficient. Different from the first subset, support vector percentages also decrease significantly in 11 out of 13 tasks, in addition to a decrease in total cost. The 5×2 cv paired F test reports a significant loss for the average test accuracy in three tasks and the difference between the average test accuracies of two variants is significant according to the Wilcoxon signed rank test. The Wilcoxon signed rank test finds significant wins for the number of support vectors stored and total cost.

5. Conclusions

This work introduces a cost-conscious kernel combination framework to include the cost of kernel computations and data acquisition/generation into the MKL mathematical model. In this work, we present results for two set of experiments on benchmark data sets: combining different kernels on the same data representation and combining different data representations with the same kernel. The results show that incorporating a cost factor into the model enables us to use only the necessary kernels and avoid costly kernel computations and input generation for some data representations in testing phase when possible. The cost of a kernel depends on the time/space complexity of the kernel implementation (in software or hardware) and the cost of sensing the input representation and manipulating it.

The cost-conscious MKL variant is also tested on two bioinformatics applications described from CYGD. Similar to the results obtained on the UCI benchmark data sets, the cost-conscious variant of MKL helps us trade-off the contribution of a kernel to accuracy with its complexity and can eliminate expensive data representations/kernels when possible. By using cost-conscious MKL in bioinformatics applications, we can select a kernel combination that enables us to get rid of obtaining costlier data representations (generally obtained through additional experimental processes) and evaluating expensive kernel functions.

To summarize, we see that integrating the cost factor into kernel combination allows us to identify and select necessary kernel functions for the classification task at hand. Expensive kernels or costlier representations, in terms of evaluation time and memory, can be disregarded if they do not convey important information.

Acknowledgments

This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, the Boğaziçi University Scientific Research Project 07HA101 and the Turkish Scientific Technical Research Council (TÜBİTAK) under Grant EEEAG 107E222. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK.

References

Alimoğlu, F., Alpaydın, E., 1997. Combining multiple representations and classifiers for pen-based handwritten digit recognition. In: Proc. 4th Internat. Conf. on Document Analysis and Recognition.

Alpaydın, E., 1999. Combined 5×2 cv F test for comparing supervised classification learning algorithms. Neural Comput. 11, 1885–1892.

Asuncion, A., Newman, D., 2007. UCI machine learning repository. Available from: <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Bach, F.R., Lanckriet, G.R.G., Jordan, M.I., 2004. Multiple kernel learning, conic duality, and the SMO algorithm. In: Proc. 21st Internat. Conf. on Machine Learning.

Cristianini, N., Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press.

Demir, Ç., Alpaydın, E., 2005. Cost-conscious classifier ensembles. Pattern Recognition Lett. 26, 2206–2214.

Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I., 2002. Learning the kernel matrix with semi-definite programming. In: Proc. 19th Internat. Conf. on Machine Learning.

Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I., 2004a. Learning the kernel matrix with semidefinite programming. J. Machine Learning Res. 5, 27–72.

Lanckriet, G.R.G., De Bie, T., Cristianini, N., Jordan, M.I., Noble, W.S., 2004b. A statistical framework for genomic data fusion. Bioinformatics 20, 2626–2635.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C., 2002. Text classification using string kernels. J. Machine Learning Res. 2, 419–444.

Mewes, H.W., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schüller, C., Stocker, S., Weil, B., 2000. MIPS: A database for genomes and protein sequences. Nucl. Acid Res. 28, 37–40.

Moguerza, J.M., Muñoz, A., De Diego, I.M., 2004. Improving support vector classification via the combination of multiple sources of information. In: Proc. Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops.

Mosek, 2009. The MOSEK Optimization Tools Manual Version 5.0 (Revision 137). MOSEK ApS, Denmark.

Pavlidis, P., Weston, J., Cai, J., Grundy, W.N., 2001. Gene functional classification from heterogeneous data. In: Proc. 5th Annual Internat. Conf. on Computational Molecular Biology.

Schölkopf, B., Tsuda, K., Vert, J.P. (Eds.), 2004. Kernel Methods in Computational Biology. The MIT Press.

Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B., 2006. Large scale multiple kernel learning. J. Machine Learning Res. 7, 1531–1565.

Vapnik, V.N., 1995. The Nature of Statistical Learning Theory. Springer-Verlag.