Combining Data Sources Nonlinearly for Cell Nucleus Classification of Renal Cell Carcinoma

Mehmet Gönen[1], Aydın Ulaş[2,⋆], Peter Schüffler[3], Umberto Castellani[2], and Vittorio Murino[2,4]

[1] Aalto University School of Science, Department of Information and Computer Science, Helsinki Institute for Information Technology (HIIT), Espoo, Finland
[2] University of Verona, Department of Computer Science, Verona, Italy
[3] ETH Zürich, Department of Computer Science, Zürich, Switzerland
[4] Istituto Italiano di Tecnologia (IIT), Genova, Italy

Abstract. In kernel-based machine learning algorithms, we can learn a combination of different kernel functions in order to obtain a similarity measure that better matches the underlying problem instead of using a single fixed kernel function. This approach is called *multiple kernel learning* (MKL). In this paper, we formulate a nonlinear MKL variant and apply it for nuclei classification in tissue microarray images of *renal cell carcinoma* (RCC). The proposed variant is tested on several feature representations extracted from the automatically segmented nuclei. We compare our results with single-kernel support vector machines trained on each feature representation separately and three linear MKL algorithms from the literature. We demonstrate that our variant obtains more accurate classifiers than competing algorithms for RCC detection by combining information from different feature representations nonlinearly.

Keywords: multiple kernel learning, renal cell carcinoma, support vector machines.

1 Introduction

Empirical success of kernel-based machine learning algorithms such as *support vector machines* (SVMs) is very much dependent on the kernel function used. Kernel selection is generally handled by choosing the best-performing kernel function among a set of kernel functions on a separate validation set. Instead of using a single fixed kernel function, *multiple kernel learning* (MKL) algorithms learn a combination of different kernel functions in order to obtain a similarity measure that better matches the underlying problem [8].

Most of the MKL algorithms proposed in the literature combine the kernels linearly (i.e., linear sum, convex sum, and conic sum) [1,12,14]. Similar to nonlinear classifier combination rules, we can also combine kernels nonlinearly to

⋆ Corresponding author.

obtain better kernels [5,7,13]. We formulate a nonlinear MKL variant derived from [5] and test it on cell nucleus classification of *renal cell carcinoma* (RCC) using *tissue microarray* (TMA) images by comparing it with single-kernel SVMs and linear MKL algorithms. Our experiments demonstrate that although it is more costly to use the proposed nonlinear MKL approach, the increase in accuracy is worth its computational complexity.

The paper is organized as follows: Section 2 introduces the data set used in this study. We explain the methods applied in Section 3 and give our experimental results in Section 4. We conclude the paper in Section 5.

2 Data Set

Cancer tissue analysis consists of several consecutive estimation and classification steps which require intensive laboratory practice. The TMA technology enables studies associating molecular changes with clinical endpoints [11]. In this technique, 0.6 mm tissue cylinders are extracted from primary tumor blocks of hundreds of different patients, and are subsequently embedded into a recipient paraffin block. Such array blocks can then be used for simultaneous analysis of primary tumors on DNA, RNA, and protein level.

In this work, we consider the computer based classification of tissue from RCC after such a workflow has been applied. The tissue has been transferred to an array and stained to make the morphology of cells and cell nuclei visible. Current image analysis software for TMAs requires extensive user interaction to properly identify cell populations on the TMA images, to select regions of interest for scoring, to optimize analysis parameters and to organize the resulting raw data. Because of these drawbacks, pathologists typically collect the TMA data by manually assigning a composite staining score for each spot. Such manual scoring can result in serious inconsistencies between data collected during different microscopy sessions. Manual scoring also introduces a significant bottleneck that limits the use of TMAs in high-throughput analysis.

The manual rating and assessment of TMAs under the microscope by pathologists is quite inconsistent due to the high variability of cancerous tissue and the subjective experience of humans, as shown in [6]. Therefore, decisions for grading and/or cancer therapy might be inconsistent among pathologists. With this work, we want to contribute to a more generalized and reproducible system that automatically processes the TMA images and thus helps pathologists in their daily work.

In a previous study, an automated pipeline of TMA processing was already proposed, concentrating on the investigation of various image features and associated kernels on the performance of an SVM classifier for cancerous cells [15]. In this work, we follow this workflow (see Fig. 1) and extend the nucleus classification using different MKL strategies to combine information from multiple sources (in our case different representations). By considering different types of features, in Section 4, we show that nonlinear MKL reaches significantly better accuracies than linear MKL algorithms and single-kernel SVMs.

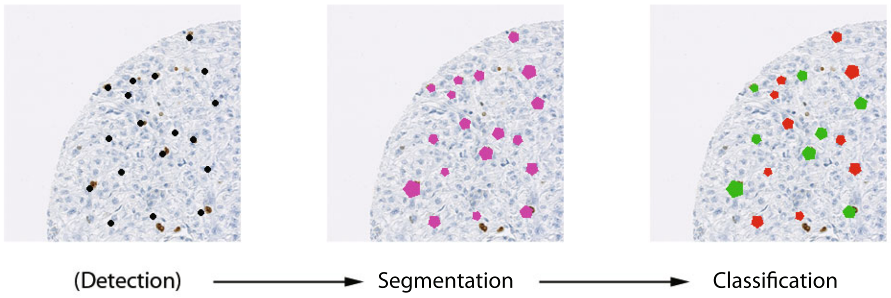

Fig. 1. One key point in the automatic TMA analysis for RCC is the nucleus classification. Nuclei are eosin stained and visible in the TMA image as dark blue spots. We want to do the classification of cell nuclei into cancerous or benign, which is recently done by trained pathologists with their eyes. The automatic approach comprises nucleus detection on the image, the segmentation of the nuclei and the classification, all based on training data labeled by two human experts.

2.1 Tissue Micro Arrays

Tissue Micro Arrays comprise several hundreds of roundish 1mm spots on one carrier plate. Each spot is a small piece of tissue, consisting of several hundreds cancerous and healthy cells. The morphological structure of the cells is made visible under light microscope by eosin staining. Further, proliferating cell nuclei expressing the protein MIB-1 (Ki-67 antigen) are immunohistochemically made visible by brown staining.

The TMA spots are scanned and stored for processing. The images are three channel color images of size 3000 px $\times$ 3000 px. The labeled dataset comprises eight tissue spots from eight patients, each showing 100–200 cells (see Fig. 2).

The TMA images are independently labeled by two pathologists [6]. Therefore, locations and disease states (cancer/non cancer) of each cell in the TMA image are known. From eight labeled TMA images, we extracted 1633 nuclei-patches of size 80 px $\times$ 80 px. Each patch shows a cell nucleus in the center (see Fig. 3). 1273 (78 per cent) from the nuclei form our data set, where the two pathologists agree on the label: 891 (70 per cent) benign and 382 (30 per cent) malignant nuclei.

2.2 Image Normalization and Patching

To minimize illumination variances among the scans, the TMA images were adjusted in contrast. For classification of the individual nuclei, we extracted patches from the whole image such that each 80 px $\times$ 80 px patch has one nucleus in the center (see Fig. 3).

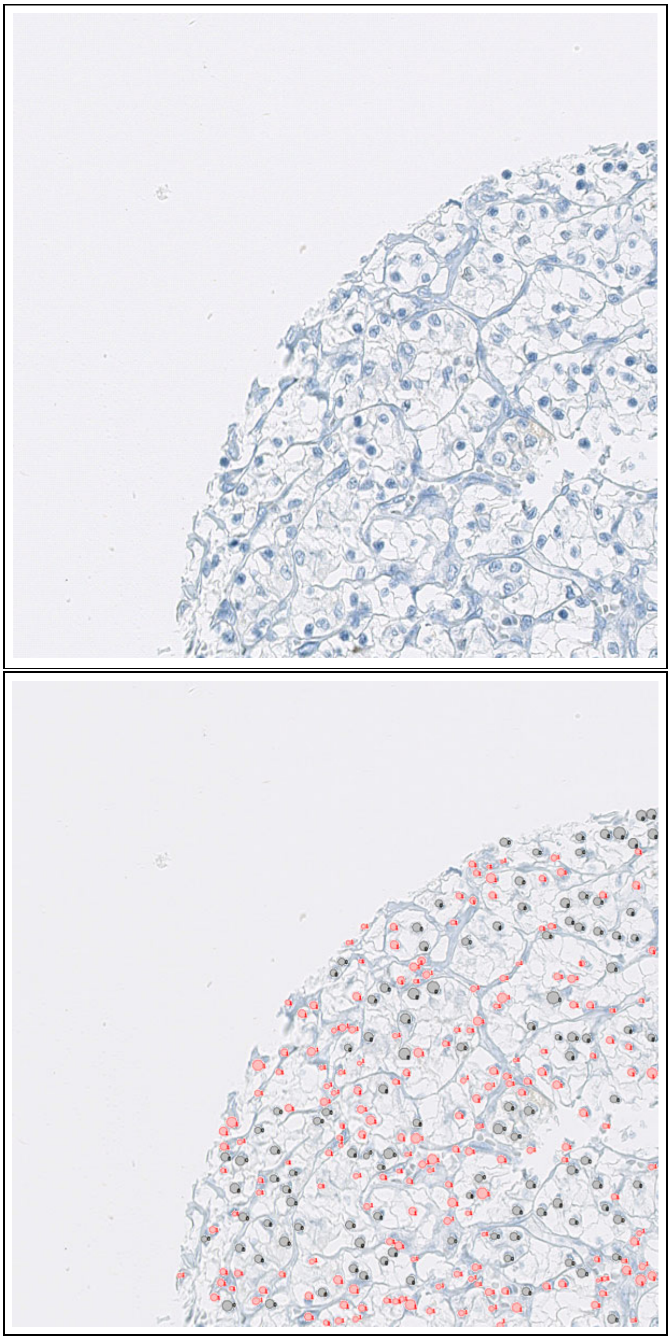

Fig. 2. Top: One 1500 px × 1500 px quadrant of a TMA spot from a RCC patient. **Bottom:** A pathologist exhaustively labeled all cell nuclei and classified them into malignant (black) and benign (red).

2.3 Segmentation

For graphcut segmentation [3], the gray intensities were used as unary potentials. As cell nuclei tend to be roundish, the binary potentials were linearly weighted based on their distance to the center to prefer roundish objects (see Fig. 3). The border of the segmented nuclei was used to calculate several shape features as described in the following section.

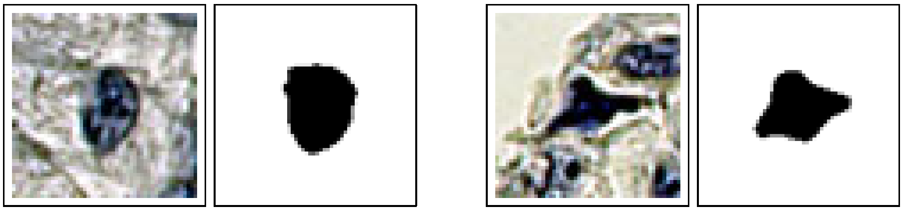

Fig. 3. Two examples of nucleus segmentation. The original 80 px × 80 px patches are shown, each with the corresponding nucleus shape found with graphcut.

2.4 Feature Extraction

For training and testing the various classifiers, we extracted several histogram-like features from the patches (see Table 1).

3 Methodology

The main idea behind SVMs [16] is to transform the input feature space to another space (possibly with a greater dimension) where the classes are linearly separable. After training, the discriminant function of SVM becomes $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b$, where $\boldsymbol{w}$ is the vector of weights, b is the threshold, and $\Phi(\,\cdot\,)$ is the mapping function. Using the dual formulation and the kernel trick, one does not have to define this mapping function explicitly and the discriminant function can be written as

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

where $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle$ is the kernel function that calculates a similarity measure between data instances. Selecting the kernel function is the most important issue in the training phase; it is generally handled by choosing the best-performing kernel function among a set of kernel functions on a separate validation set.

In recent years, MKL methods have been proposed [8], for learning a combination k_η of multiple kernels instead of selecting only one:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\eta}) = f_\eta(\{k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)_{m=1}^{P}\}; \boldsymbol{\eta}) \tag{1}$$

Table 1. Features extracted from patch images for training and testing. All features are histograms.

Name	Feature Description
`ALL`	**Patch Intensity**: A 16-bin histogram of gray scaled patch.
`FG`	**Foreground Intensity**: A 16-bin histogram of nucleus.
`BG`	**Background Intensity**: A 16-bin histogram of background.
`LBP`	**Local Binary Patterns**: This local feature has been shown to bring considerable performance in face recognition tasks. It benefits from the fact that it is illumination invariant.
`COL`	**Color Feature**: The only feature comprising color information. The colored patch (RGB) is rescaled to size 5×5. The 3×25 channel intensities are then concatenated to a feature vector of size 75.
`FCC`	**Freeman Chain Code**: The FCC describes the nucleus' boundary as a string of numbers from 1 to 8, representing the direction of the boundary line at that point [9]. The boundary is discretized by subsampling with grid size 2. For rotational invariance, the first difference of the FCC with minimum magnitude is used. The FCC is represented in a 8-bin histogram.
`SIG`	**1D-Signature**: Lines are considered from the object center to each boundary pixel. The angles between these lines form the signature of the shape [9]. As feature, a 16-bin histogram of the signature is generated.
`PHOG`	**Pyramid Histograms of Oriented Gradients**: PHOGs are calculated over a level 2 pyramid on the gray-scaled patches [2].

where the combination function $f_{\boldsymbol{\eta}}$ forms a single kernel from P base kernels using the parameters $\boldsymbol{\eta}$. Different kernels correspond to different notions of similarity and instead of searching which works best, the MKL method does the picking for us, or may use a combination of kernels. MKL also allows us to combine different representations possibly coming from different sources or modalities.

3.1 Linear Multiple Kernel Learning

There is significant work on the theory and application of MKL and most of the proposed algorithms use a linear combination function such as convex sum or conic sum. Fixed rules use the combination function in (1) as a fixed function of the kernels, without any training. Once we calculate the combined kernel, we train a single kernel machine using this kernel. For example, we can obtain a valid kernel by taking the mean of the combined kernels.

Instead of using a fixed combination function, we can also have a function parameterized by a set of parameters and then we have a learning procedure to optimize these parameters as well. The simplest case is to parameterize the sum rule as a weighted sum:

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\eta}) = \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)$$

with $\eta_m \in \mathbb{R}$. Different versions of this approach differ in the way they put restrictions on the kernel weights: [1,12,14]. For example, we can use arbitrary weights (i.e., linear combination), nonnegative kernel weights (i.e., conic combination), or weights on a simplex (i.e., convex combination).

3.2 Nonlinear Multiple Kernel Learning

A linear combination may be restrictive and nonlinear combinations are also possible [5,7,13]. [5] developed a nonlinear kernel combination method based on kernel ridge regression (KRR) and polynomial combination of kernels. The nonlinear combination can be formulated as

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\boldsymbol{q} \in \mathcal{Q}} \eta_{q_1 q_2 \dots q_P} k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1)^{q_1} k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)^{q_2} \dots k_P(\boldsymbol{x}_i^P, \boldsymbol{x}_j^P)^{q_P}$$

where $\mathcal{Q} = \{\boldsymbol{q} \colon \boldsymbol{q} \in \mathbb{Z}_+^P, \ \sum_{m=1}^{P} q_m \leq d\}$ and $\eta_{q_1 q_2 \dots q_P} \geq 0$. The number of parameters to be learned is too large and the combined kernel is simplified in order to reduce the learning complexity:

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\boldsymbol{q} \in \mathcal{R}} \eta_1^{q_1} \eta_2^{q_2} \dots \eta_P^{q_P} k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1)^{q_1} k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)^{q_2} \dots k_P(\boldsymbol{x}_i^P, \boldsymbol{x}_j^P)^{q_P}$$

where $\mathcal{R} = \{\boldsymbol{q} \colon \boldsymbol{q} \in \mathbb{Z}_+^P, \ \sum_{m=1}^{P} q_m = d\}$ and $\boldsymbol{\eta} \in \mathbb{R}^P$. For example, when $d = 2$, the combined kernel function becomes

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) k_h(\boldsymbol{x}_i^h, \boldsymbol{x}_j^h). \tag{2}$$

The combination weights are optimized by solving the following min-max optimization problem:

$$\underset{\boldsymbol{\eta} \in \mathcal{M}}{\text{mininimize}} \ \underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\text{maximize}} \ \boldsymbol{y}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top (\mathbf{K}_{\eta} + \lambda \mathbf{I}) \boldsymbol{\alpha}$$

where $\mathcal{M}$ is a positive, bounded, and convex set. Two possible choices for the set $\mathcal{M}$ are the ℓ_1-norm and ℓ_2-norm bounded sets defined as

$$\begin{aligned} \mathcal{M}_1 &= \{\boldsymbol{\eta} \colon \boldsymbol{\eta} \in \mathbb{R}_+^P, \ \|\boldsymbol{\eta} - \boldsymbol{\eta}_0\|_1 \leq \Lambda\} \\ \mathcal{M}_2 &= \{\boldsymbol{\eta} \colon \boldsymbol{\eta} \in \mathbb{R}_+^P, \ \|\boldsymbol{\eta} - \boldsymbol{\eta}_0\|_2 \leq \Lambda\} \end{aligned} \tag{3}$$

where $\boldsymbol{\eta}_0$ and Λ are two model parameters. A projection-based gradient-descent algorithm can be utilized to solve this min-max optimization problem. At each iteration, $\boldsymbol{\alpha}$ is obtained by solving a KRR problem with the current kernel matrix and $\boldsymbol{\eta}$ is updated with the gradients calculated using $\boldsymbol{\alpha}$ while considering the bound constraints on $\boldsymbol{\eta}$ due to $\mathcal{M}_1$ or $\mathcal{M}_2$.

We formulate a variant of this method by replacing KRR with SVM as the base learner. In that case, the optimization problem becomes

$$\underset{\boldsymbol{\eta} \in \mathcal{M}}{\text{mininimize}} \;\; J_{\boldsymbol{\eta}} = \underset{\boldsymbol{\alpha} \in \mathcal{A}}{\text{maximize}} \;\; \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top ((\boldsymbol{y}\boldsymbol{y}^\top) \odot \mathbf{K}_{\boldsymbol{\eta}}) \boldsymbol{\alpha}$$

where $\odot$ denotes the element-wise product between matrices and $\mathcal{A}$ is defined as

$$\mathcal{A} = \{\boldsymbol{\alpha} \colon \boldsymbol{\alpha} \in \mathbb{R}_+^P, \;\; \boldsymbol{y}^\top \boldsymbol{\alpha} = 0, \;\; \boldsymbol{\alpha} \leq C\}.$$

Note that the simultaneous optimization of $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ is not possible. Hence, we use a two-step optimization strategy to optimize them alternatively even though it is prone to sticking at local optima. We solve this optimization problem again using a projection-based gradient-descent algorithm. When updating the kernel parameters at each iteration, the gradients of $J_{\boldsymbol{\eta}}$ with respect to $\boldsymbol{\eta}$ are used. These gradients can be written as

$$\frac{\partial J_{\boldsymbol{\eta}}}{\partial \eta_m} = -\frac{1}{2} \sum_{h=1}^{P} \eta_h \boldsymbol{\alpha}^\top ((\boldsymbol{y}\boldsymbol{y}^\top) \odot \mathbf{K}_h \odot \mathbf{K}_m) \boldsymbol{\alpha}.$$

4 Experiments

4.1 Experimental Methodology

1273 nuclei samples were divided into ten folds with stratification. We then trained single-kernel SVMs with different kernels for each feature representation and combined the feature representations using four different MKL algorithms on these folds. In our experiments, we used three different kernel functions: the linear kernel (`LIN`), the second-degree polynomial kernel (`POL`), and the Gaussian kernel (`GAU`). Using a rule of thumb, the width parameter of the Gaussian kernel was chosen as $\sqrt{D}$ where D is the dimensionality of the corresponding feature representation.

We implemented single-kernel SVM and four MKL algorithms in MATLAB and solved the canonical SVM optimization problems with the LIBSVM software [4]. `SVM` denotes the single-kernel SVMs trained on each feature representation separately. `RBMKL` denotes the rule-based MKL algorithm that trains an SVM with the mean of the combined kernels. `SimpleMKL` is the iterative algorithm of [14] that uses projected gradient updates and trains single-kernel SVMs at each iteration. `GLMKL` denotes the group Lasso-based MKL algorithms proposed by [10,17]. In our implementation, we used ℓ_1-norm on the kernel weights and learned a convex combination of the kernels. `NLMKL` denotes the nonlinear MKL variant derived from [5], which uses the quadratic kernel given in (2) and selects the kernel weights from the set $\mathcal{M}_1$ in (3). In our implementation, $\boldsymbol{\eta}_0$ is taken as $\mathbf{0}$ and Λ is assigned to 1 arbitrarily.

As a summary, we have eight representations (ALL, FG, BG, LBP, COL, FCC, SIG, and PHOG), three kernels (LIN, POL, and GAU), and five algorithms (SVM, RBMKL, SimpleMKL, GLMKL, and NLMKL).

4.2 Results

Table 2 reports the single-kernel SVM accuracies for all feature representation and kernel function pairs. We see that the best performance was obtained as 76.9 per cent using (PHOG, GAU) pair. Independent of the kernel function used, feature representations BG and PHOG gave consistently higher accuracies than other representations.

Table 2. Single-kernel SVM accuracies

	LIN	POL	GAU
ALL	70.0±0.2	71.9±2.9	68.7±2.9
FG	70.0±0.2	71.2±3.7	65.9±4.3
BG	70.2±0.6	72.7±3.8	69.6±3.1
LBP	70.0±0.2	63.6±2.7	68.4±6.3
COL	70.2±3.0	62.9±3.5	67.2±3.4
FCC	70.0±0.2	69.8±0.7	62.9±5.5
SIG	70.0±0.2	69.6±3.4	66.0±3.0
PHOG	76.0±3.4	70.5±3.3	**76.9**±**2.7**

Next, using four different MKL algorithms, we combined eight kernels calculated on the feature representations with the same kernel function. Table 3 lists the results of best single-kernel SVMs and four MKL algorithms trained. We can achieve an accuracy of 83.3 per cent by combining eight GAU kernels with NLMKL. This result is better than all other MKL settings and single-kernel SVMs. In the last column of Table 3, the results of combining all possible feature representation and kernel function pairs (i.e., 24 kernels) in a single learner are shown. NLMKL is still the best MKL algorithm even though the average accuracy decreases to 83.1 per cent.

To give a feel of complexity, we also measured the time required to run each method. Table 4 gives the running times in seconds. We can see that NLMKL takes more time because of the second order dependency to the number of kernels in

Table 3. MKL accuracies

	LIN	POL	GAU	LIN+POL+GAU
SVM	76.0±3.4	72.7±3.8	76.9±2.7	NA
RBMKL	77.3±4.0	77.2±2.4	82.7±3.6	81.8±3.8
SimpleMKL	77.1±3.3	77.3±2.3	81.8±3.8	81.6±3.9
GLMKL	77.1±3.5	76.5±3.2	81.8±4.3	81.8±3.8
NLMKL	77.9±3.9	79.2±3.8	**83.3**±**3.6**	83.1±3.5

Table 4. Time required for each method (in seconds). Single kernel time measurements are summed over all representations.

	LIN	POL	GAU	LIN+POL+GAU
SVM	4.45	5.81	3.52	NA
RBMKL	1.56	0.87	1.35	2.57
SimpleMKL	35.55	11.07	11.71	32.81
GLMKL	11.11	4.61	5.20	14.27
NLMKL	45.25	39.21	44.28	323.83

the gradient computations. This difference becomes more apparent when we increase the number of combined kernels. The running time can be reduced by caching the element-wise products between the kernel matrices.

4.3 Discussion

In this paper, we formulated a nonlinear MKL algorithm derived from [5] and we have seen that proposed algorithm performs better than single-kernel SVMs and three linear MKL algorithms. When we were combining linear kernels on the feature representations, we observed that linear MKL algorithms achieved to outperform single-kernel SVMs, whereas the nonlinear MKL algorithm improved the average accuracy most thanks to the nonlinearity in kernel combination. Even though the kernels were nonlinear when we were combining polynomial and Gaussian kernels, the nonlinear MKL algorithm got better accuracies than single-kernel SVMs and linear MKL algorithms. We have seen that when we use the nonlinear MKL algorithm, we achieved 6.4 per cent improvement in accuracy compared to single-kernel SVMs.

5 Conclusion

In this paper, we formulate a nonlinear MKL algorithm variant and use it for the classification of nuclei in TMA images of RCC. We used SVMs extensively through different feature sets in our previous work [15]. This study extends our previous work using several feature sets in a nonlinear MKL setting and compares the results with single-kernel SVMs and several linear MKL algorithms.

We have seen that the nonlinear MKL algorithm performs better than single-kernel SVMs and linear MKL algorithms in all of the experiments. The proposed nonlinear MKL variant learns a better similarity measure than linear MKL algorithms by combining the input kernels nonlinearly. In this work, we used image-based feature sets for creating multiple feature representations. In a further application of this scenario, the use of other modalities or other features (e.g., SIFT) extracted from these images as well as the incorporation of complementary information of different modalities to achieve better classification accuracy is possible.

Acknowledgements. We acknowledge financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

References

1. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st International Conference on Machine Learning (2004)
2. Bosch, A., Zisserman, A., Munoz, X.: Representing shape with a spatial pyramid kernel. In: Proceedings of the 6th ACM International Conference on Image and Video Retrieval (2007)
3. Boykov, Y., Veksler, O., Zabih, R.: Efficient approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 1222–1239 (2001)
4. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines (2001)
5. Cortes, C., Mohri, M., Rostamizadeh, A.: Learning non-linear combinations of kernels. In: Advances in Neural Information Processing Systems, vol. 22 (2009)
6. Fuchs, T.J., Wild, P.J., Moch, H., Buhmann, J.M.: Computational pathology analysis of tissue microarrays predicts survival of renal clear cell carcinoma patients. In: Metaxas, D., Axel, L., Fichtinger, G., Székely, G. (eds.) MICCAI 2008, Part II. LNCS, vol. 5242, pp. 1–8. Springer, Heidelberg (2008)
7. Gönen, M., Alpaydın, E.: Localized multiple kernel learning. In: Proceedings of the 25th International Conference on Machine Learning (2008)
8. Gönen, M., Alpaydın, E.: Multiple kernel learning algorithms. Journal of Machine Learning Research 12, 2211–2268 (2011)
9. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital Image Processing Using MATLAB. Prentice-Hall, Inc., Englewood Cliffs (2003)
10. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: ℓ_p-norm multiple kernel learning. Journal of Machine Learning Research 12, 953–997 (2011)
11. Kononen, J., Bubendorf, L., Kallioniemi, A., Barlund, M., Schraml, P., Leighton, S., Torhorst, J., Mihatsch, M.J., Sauter, G., Kallioniemi, O.P.: Tissue microarrays for high-throughput molecular profiling of tumor specimens. Nature Medicine 4, 844–847 (1998)
12. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
13. Lewis, D.P., Jebara, T., Noble, W.S.: Nonstationary kernel combination. In: Proceedings of the 23rd International Conference on Machine Learning (2006)
14. Rakotomamonjy, A., Bach, F.R., Canu, S., Grandvalet, Y.: SimpleMKL. Journal of Machine Learning Research 9, 2491–2521 (2008)
15. Schüffler, P.J., Fuchs, T.J., Ong, C.S., Roth, V., Buhmann, J.M.: Computational TMA analysis and cell nucleus classification of renal cell carcinoma. In: Goesele, M., Roth, S., Kuijper, A., Schiele, B., Schindler, K. (eds.) Pattern Recognition. LNCS, vol. 6376, pp. 202–211. Springer, Heidelberg (2010)
16. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Sons, Chichester (1998)
17. Xu, Z., Jin, R., Yang, H., King, I., Lyu, M.R.: Simple and efficient multiple kernel learning by group Lasso. In: Proceedings of the 27th International Conference on Machine Learning (2010)