Greedy Unsupervised Multiple Kernel Learning

Grigorios Tzortzis and Aristidis Likas

Department of Computer Science, University of Ioannina,
GR 45110, Ioannina, Greece
{gtzortzi,arly}@cs.uoi.gr

Abstract. Multiple kernel learning (MKL) has emerged as a powerful tool for considering multiple kernels when the appropriate representation of the data is unknown. Some of these kernels may be complementary, while others irrelevant to the learning task. In this work we present an MKL method for clustering. The intra-cluster variance objective is extended by learning a linear combination of kernels, together with the cluster labels, through an iterative procedure. Closed-form updates for the combination weights are derived, that greatly simplify the optimization. Moreover, to allow for robust kernel mixtures, a parameter that regulates the sparsity of the weights is incorporated into our framework. Experiments conducted on a collection of images reveal the effectiveness of the proposed method.

1 Introduction

Despite the widespread interest in kernel methods in various fields, such as machine learning, computer vision and bioinformatics, their application is severely limited by the sensitivity to the choice of kernel. In recent years, multiple kernel learning (MKL) techniques [5], that determine the weights for combining a set of predefined (base) kernels as part of the learning process, have been developed to alleviate the kernel selection problem. These multiple kernels may describe different notions of similarity in the data, or, even, different modalities.

In this work, we focus on MKL clustering and employ the k-means *intra-cluster variance objective* together with a *linear combination for the base kernels*. Linearly mixing the kernels is the most common approach [2, 9, 6, 7, 11, 12], although lately effort has been put into considering nonlinear combinations [10, 3]. A simple iterative procedure is devised to recover the partitioning, using k-medoids [1], and the kernel mixing coefficients, which are estimated by *closed-form expressions*. Moreover, to avoid multiple restarts for k-medoids, a greedy initialization strategy is developed.

An important aspect of MKL is the sparsity of the kernel combination. Sparse approaches rely on the assumption that some kernels are irrelevant for the underlying problem and retain a few base kernels through an ℓ_1-norm regularizer on the weights [9, 12]. They are appealing since they greatly enhance interpretability and their models are distinguished by small capacity. However, *different kernels may capture different aspects of the data* and thus all kernels are important in

I. Maglogiannis, V. Plagianakos, and I. Vlahavas (Eds.): SETN 2012, LNAI 7297, pp. 73–80, 2012.

this case, *albeit to a different degree*. Therefore, to provide a more robust ranking of the kernels, based on their quality, the ℓ_2-norm regularizer was applied in [13, 2], while in [6, 11] the general ℓ_p-norm regularizer was introduced, that forces weights to become less sparse as p increases. Also, Lange and Buhmann [7] learned a linear mixture of similarity matrices using an entropy criterion to regulate sparsity. In our approach, a parameter that must be set beforehand controls the weights similarly to the ℓ_p-norm regularizer, allowing the *exploitation of the complementary information of the kernels*. As shown in the experiments, the results of using a single kernel or evenly all kernels can be vastly improved.

The rest of this paper is organized as follows. Next section contains the basics about kernel-based clustering and the greedy initialization scheme. Our MKL framework is detailed in Sect. 3. The empirical results are reported in Sect. 4, before the concluding remarks of Sect. 5.

2 Feature Space Clustering

To uncover the hidden structures of a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ it is common practice to map the instances to a higher dimensional reproducing kernel Hilbert space $\mathcal{H}$, *a.k.a.* feature space, via a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{H}$ and then perform clustering in space $\mathcal{H}$. Thus it is possible to get nonlinear separators in input space through linear separators in feature space.

Usually a kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ [4] is applied to directly provide the inner products in feature space without explicitly defining transformation ϕ (for certain kernel functions the transformations are intractable). This gives rise to the kernel matrix $K \in \Re^{N \times N}$, $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, which is the most common way of representing similarity in feature space. An important property is that the squared Euclidean distances in feature space can be computed using solely the kernel matrix entries:

$$\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = K_{ii} - 2K_{ij} + K_{jj} \ . \tag{1}$$

2.1 *k*-Medoids

To partition dataset $\mathcal{X}$ into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^M$, using a kernel matrix, k-medoids [1] can be utilized to minimize the intra-cluster variance in feature space (2) over $\mathbf{m}_k$ and δ_{ik}, where $\mathbf{m}_k \in \mathcal{X}$ is the k-th cluster medoid, δ_{ik} is an indicator variable with $\delta_{ik} = 1$ if $\mathbf{x}_i \in \mathcal{C}_k$ and 0 otherwise and l_k is the index of the data point corresponding to the k-th medoid, *i.e.* $\mathbf{m}_k = \mathbf{x}_{l_k}$.

$$\mathcal{E}_{\mathcal{H}} = \sum_{i=1}^{N} \sum_{k=1}^{M} \delta_{ik} \|\phi(\mathbf{x}_i) - \phi(\mathbf{m}_k)\|^2 = \sum_{i=1}^{N} \sum_{k=1}^{M} \delta_{ik} (K_{ii} - 2K_{il_k} + K_{l_k l_k}) \tag{2}$$

An analogous to typical k-means iterative procedure is employed to optimize the objective, where the medoids and the cluster assignments are updated in turn. This procedure converges to a local minimum that strongly depends on the initialization of the medoids. To circumvent this problem and avoid multiple restarts, we have adopted a greedy method for selecting initial medoids.

2.2 Greedy Medoid Initialization

The greedy medoid initialization algorithm is an incremental approach for deterministically finding a set of M medoids to initialize the k-medoids algorithm. The clusters are added to the solution one by one, such that the clustering objective (2) is minimized.

In detail, given a kernel K, we start by considering the whole dataset as one cluster, choosing the medoid of the dataset as the first medoid $\mathbf{m}_1$. Suppose that $k-1$ medoids have already been added and $d_i^{(k-1)}$ denotes the squared distance in feature space of instance $\mathbf{x}_i$ to its cluster medoid in the solution with $k-1$ clusters. In order to select the k-th medoid, a search is performed over all dataset points to find the one that provides the greatest reduction of the objective (2). If $\mathbf{x}_j$ is chosen as the k-th medoid then it will allocate all instances that are closer to it in feature space $\mathcal{H}$ than to their cluster medoid in the solution with $k-1$ clusters ($d_i^{(k-1)}$ distance). For each such reallocation the objective will decrease by $d_i^{(k-1)} - \|\phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)\|^2$. Therefore, the overall reduction caused by considering $\mathbf{x}_j$ as the k-th medoid can be quantified as:

$$b_j^{(k)} = \sum_{i=1}^{N} \max\left\{ d_i^{(k-1)} - \|\phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)\|^2, 0 \right\} \quad . \tag{3}$$

Obviously, the point $\mathbf{x}_j$ that yields the highest $b_j^{(k)}$ value (denoted by $\mathbf{x}_{j^*}$) is appointed as the k-th medoid, *i.e.* $\mathbf{m}_k = \mathbf{x}_{j^*}$, and all points $\mathbf{x}_i$ for which $d_i^{(k-1)} > \|\phi(\mathbf{x}_{j^*}) - \phi(\mathbf{x}_i)\|^2$ are assigned to the new cluster. *It must be stressed that the $k-1$ medoids remain the same.* The above is repeated until $k = M$. After initial medoids have been located, k-medoids can be executed to refine the result.

3 Learning a Linear Kernel Combination for Clustering

The integration of multiple high quality base kernels in the clustering process can enhance the potential of clustering algorithms, whereas the inclusion of uninformative or noisy kernels may lead to performance degradation. Therefore, it is extremely important to develop algorithms that *differentiate and rank the kernels according to the conveyed information.* To this end, we learn a weighted linear combination of base kernels, together with the cluster assignments.

3.1 Problem Formulation

Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ be a dataset with N instances and assume that V kernel matrices, $\{K^{(v)}\}_{v=1}^V$, with corresponding feature spaces $\{\mathcal{H}^{(v)}\}_{v=1}^V$, are available. Those kernels can be thought of as *providing different views*[1] of the original instances. Our target is to partition dataset $\mathcal{X}$ into M disjoint clusters,

[1] On the following we use the term view to refer to the different representations-perspectives of the original dataset, implied by the different kernels.

$\{\mathcal{C}_k\}_{k=1}^{M}$, by optimizing the intra-cluster variance objective in a feature space $\widetilde{\mathcal{H}}$, whose definition is based on all available views.

A convenient way is to mix the kernels by considering their linear combination (4), where w_v are the kernel weights and p is an exponent that controls the distribution of the weights across the views.

$$\widetilde{K} = \sum_{v=1}^{V} w_v^p K^{(v)} \ , w_v \geq 0 \ , \sum_{v=1}^{V} w_v = 1 \ , p \geq 1 \tag{4}$$

The composite matrix $\widetilde{K}$ *is also a kernel matrix*, to which a transformation $\widetilde{\phi}$ corresponds, *i.e.* $\widetilde{K}_{ij} = \widetilde{\phi}(\mathbf{x}_i)^\top \widetilde{\phi}(\mathbf{x}_j)$. The weights reflect the contribution of the individual kernels in $\widetilde{K}$ and an appropriate weight estimation should remove the irrelevant kernels (zero weight), while allowing less informative ones to contribute with a smaller degree (smaller weight). Weights are required to have unit sum to avoid overfitting. The clustering optimization task can now be posed as:

$$\min_{\{\mathcal{C}_k\}_{k=1}^{M} ,\{w_v\}_{v=1}^{V}} \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \|\widetilde{\phi}(\mathbf{x}_i) - \widetilde{\phi}(\mathbf{m}_k)\|^2}_{\mathcal{E}_{\widetilde{\mathcal{H}}}} \ , \ s.t. \ w_v \geq 0 \ , \sum_{v=1}^{V} w_v = 1 \ . \tag{5}$$

Note that the exponent p must be set a priori and is not part of the optimization. From (4) and (5) it easy to verify that that the intra-cluster variance in space $\widetilde{\mathcal{H}}$ is the weighted sum of the intra-cluster variances in the individual feature spaces $\mathcal{H}^{(v)}$, *under a common clustering* (6).

$$\mathcal{E}_{\widetilde{\mathcal{H}}} = \sum_{v=1}^{V} w_v^p \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik}(K_{ii}^{(v)} - 2K_{il_k}^{(v)} + K_{l_k l_k}^{(v)}) = \sum_{v=1}^{V} w_v^p \mathcal{E}_{\mathcal{H}^{(v)}} \ . \tag{6}$$

3.2 The Algorithm

A two step iterative scheme, called **G**reedy **U**nsupervised **MKL** (GUMKL), that alternates between computing the clusters for fixed weights and estimating the weights for fixed clusters, is proposed. Note that weights are uniformly initialized, as outlined in Algorithm 1.

Cluster Update. Given the weights, kernel $\widetilde{K}$ is calculated and the greedy method is applied to pick initial medoids for the current iteration. After that k-medoids is employed to get the cluster assignments and their representatives. Note that we elect to reinitialize the medoids at each iteration to avoid poor minima, since the feature space $\widetilde{\mathcal{H}}$ may change considerably after updating the weights, making the previous iteration medoids inappropriate for initialization.

Weight Estimation. For $p > 1$ the intra-cluster variance $\mathcal{E}_{\widetilde{\mathcal{H}}}$ is convex *w.r.t.* the (constrained) weights, as can be confirmed from (6). Hence, the *optimal values for the weights for the current clusters* can be determined by plugging into (6) the constraints from (5), through a Lagrange multiplier, and setting the

Algorithm 1. Greedy Unsupervised MKL

Input: Kernel matrices $\{K^{(v)}\}_{v=1}^{V}$, Number of clusters M, Exponent p ($p \geq 1$).
Output: Clustering solution $\{\mathcal{C}_k\}_{k=1}^{M}$, Weights $\{w_v\}_{v=1}^{V}$.

$t = 0$
$w_v^{(0)} = 1/V$, $v = 1, \ldots, V$ //Initial weights.
repeat
  $t = t + 1$
  $\widetilde{K}^{(t)} = \sum_{v=1}^{V} \left(w_v^{(t-1)}\right)^p K^{(v)}$
  $(\{l_k^{(t)}\}_{k=1}^{M})$=greedy-medoids-initialization$(\widetilde{K}^{(t)})$
  $(\{l_k^{(t)}\}_{k=1}^{M}, \{\mathcal{C}_k^{(t)}\}_{k=1}^{M}, \mathcal{E}_{\widetilde{\mathcal{H}}}^{(t)})$=$k$-medoids$(\widetilde{K}^{(t)}, \{l_k^{(t)}\}_{k=1}^{M})$
  Update the view weights $w_v^{(t)}$, $v = 1, \ldots, V$, through (7), (8).
until $|\mathcal{E}_{\widetilde{\mathcal{H}}}^{(t)} - \mathcal{E}_{\widetilde{\mathcal{H}}}^{(t-1)}| < \epsilon$

derivatives *w.r.t.* w_v to zero. After some manipulation the following closed-form solution is derived:

$$w_v = 1/\sum_{v'=1}^{V} \left(\frac{\mathcal{E}_{\mathcal{H}^{(v)}}}{\mathcal{E}_{\mathcal{H}^{(v')}}}\right)^{\frac{1}{p-1}} \quad , p > 1 \ . \tag{7}$$

For $p = 1$ the optimization (5) is actually a linear program and the solution lies on the corners of the simplex (8).

$$w_v = \begin{cases} 1 \ , \ v = \text{argmin}_{v'} \, \mathcal{E}_{\mathcal{H}^{(v')}} \\ 0 \ , \ \text{otherwise} \end{cases} \quad , p = 1 \tag{8}$$

Note that for $0 < p < 1$, $\mathcal{E}_{\widetilde{\mathcal{H}}}$ becomes concave (6), thus the updates (7) would increase $\mathcal{E}_{\widetilde{\mathcal{H}}}$, which, of course, is not desired.

From (7) it can be observed that views with lower intra-cluster variance $\mathcal{E}_{\mathcal{H}^{(v)}}$ are assigned higher weights, hence *kernels are ranked according to their quality as reflected through* $\mathcal{E}_{\mathcal{H}^{(v)}}$. The exponent p *acts as a regularizer on the ranking.* The greater (smaller) the p value is, the differences between the views are suppressed (amplified) and the weights become more uniform (sparser) (*i.e.* their distribution is more flat (more peaky)). Moreover, for $p = 1$ a completely sparse outcome emerges, where all views, except one, are discarded (8). In the limit $p \to \infty$, we get that $w_v = 1/V$ from (7) , *i.e.* all views are equally considered, irrespectively of their quality. Naturally, the most appropriate p value will depend on the underlying problem and the relative quality of kernels.

Convergence and Refinement. k-medoids monotonically decreases the intra-cluster variance and the subsequent weight updates yield a further reduction. However, the greedily selected medoids at the beginning of each iteration may increase the objective. Therefore, GUMKL cannot be guaranteed to monotonically converge, although this scenario rarely occurred during our experimentation.

It is possible to refine GUMKL results by running a kernel-based algorithm, using the composite kernel produced by GUMKL. In the empirical study we applied kernel k-means [4] and the clustering solution was further improved.

Table 1. Category composition of the tested corel datasets

Dataset	Categories			
D1	*owls*	*hippos*	*trains*	*animal paintings*
D2	*owls*	*hippos*	*trains*	*cargo ships*
D3	*buses*	*leopards*	*trains*	*cargo ships*
D4	*eagles*	*elephants*	*trains*	*passenger ships*
D5	*owls*	*mammals with horns*	*roses*	*cargo ships*

Fig. 1. Examples of images used in the experiments

4 Experimental Results

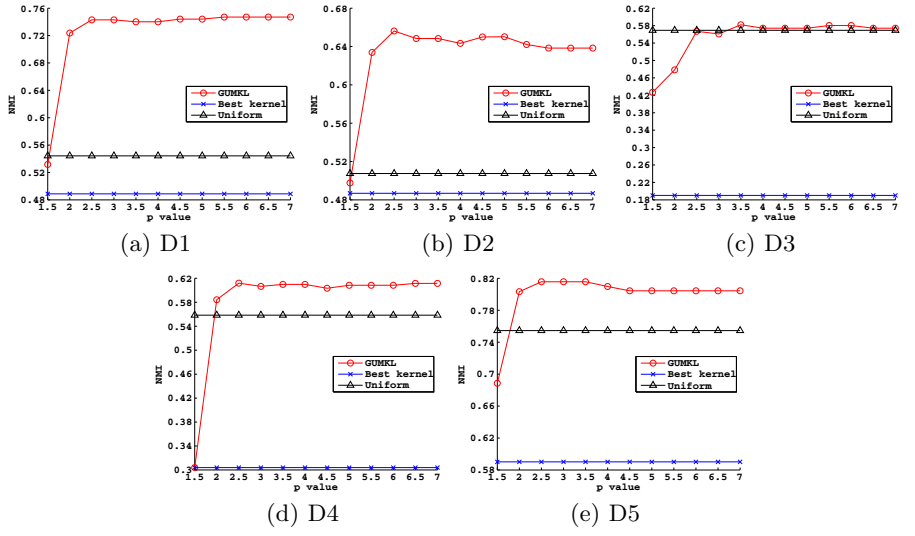
The effectiveness of GUMKL is evaluated on the task of unsupervised object category recognition on the corel images collection. There are 34 categories in total, each with 100 images that mainly consist of a salient foreground object (Fig. 1). Seven modalities, three color-related and four texture-related modalities, are available[2] in the form of attribute vectors, which naturally produce seven base kernels (each view corresponds to a modality).

To determine whether the proposed technique combines the kernels effectively and investigate the influence of the exponent p on the weight distribution and the returned clusters, GUMKL has been executed for various $p > 1$ values. It is compared to two baselines; i) selecting the best view in terms of intra-cluster variance and splitting the dataset using the corresponding kernel only (GUMKL for $p = 1$) and ii) evenly considering all views by taking a uniform sum of the kernels (GUMKL for $p \to \infty$). For the second baseline, a single iteration of Algorithm 1 is executed without updating the weights.

Regarding the setup of the experiments, five four class datasets were created from the collection, whose classes are described in Table 1. The linear kernel function was applied on the attribute vectors of each modality to compute the seven kernels, which makes the second baseline ($p \to \infty$) equivalent to merging the modalities. The number of clusters was always set equal to the true number of classes. To assess the quality of the returned clusters, the NMI criterion [8] is used. Higher NMI values indicate a better partitioning. Finally, the reported results were always refined by a run of kernel k-means as discussed in Sect. 3.2.

The obtained results are depicted in Figs. 2 and 3. It is evident that GUMKL systematically and considerably outperforms the baselines. Hence, exploiting multiple kernels, by adjusting the mixing coefficients according to the properties of each view, can improve clustering accuracy. A single kernel ($p = 1$) is always inferior to using multiple kernels, even to the simplest case of equal weights ($p \to \infty$), showing that the modalities contain complementary information. Therefore, an aggressive weighting strategy results in loss of useful information. To add to

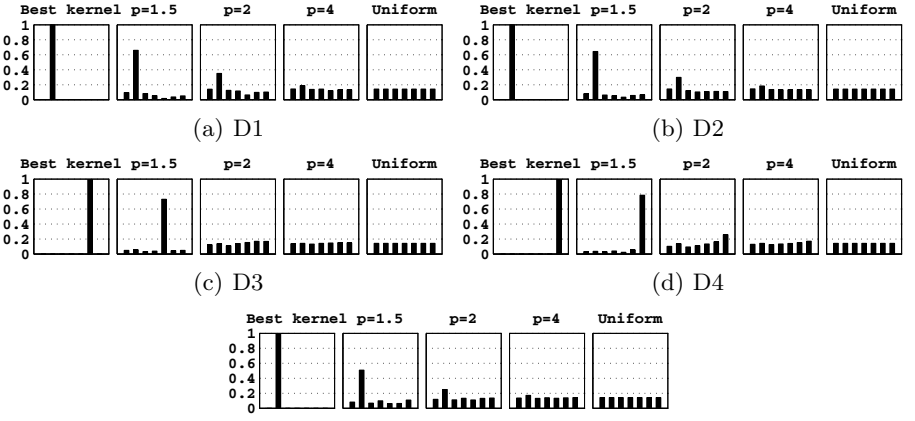
[2] `http://www.cs.virginia.edu/~xj3a/research/CBIR/Download.htm`



(a) D1 (b) D2 (c) D3

(d) D4 (e) D5

Fig. 2. Clustering performance comparison of GUMKL, for various p values, to the two baselines (best kernel, uniform)



(a) D1 (b) D2

(c) D3 (d) D4

(e) D5

Fig. 3. Weight distribution of GUMKL for indicative p values and of the two baselines (best kernel, uniform). The modalities, left to right, are color histogram, moment and coherence, coarseness and directionality of tamura texture, wavelet and mrsar texture.

that, GUMKL for $p = 1.5$ that produces a very sparse solution (Fig. 3), clearly underperforms. As p increases, coefficients exhibit a more uniform distribution, but the proposed method seems to be quite insensitive to p, if a reasonable value is chosen that avoids extremes ($2 < p < \infty$). The highest NMI is usually attained for $p \in [2.5, 3.5]$, which provides a nice balance between high sparsity and high uniformity. Finally, similar conclusions to the above can be drawn when the kernel k-means refinement is not applied, however the NMI values are lower.

5 Conclusions

We have proposed an unsupervised multiple kernel learning method for linearly combining a set of base kernels under the intra-cluster variance objective. Closed-form expressions are obtained for the combination weights, whose distribution is moderated by a parameter p that allows all kernels to contribute to the solution with distinct degrees. Stable performance is demonstrated in the experiments for a wide range of p, surpassing the compared baselines of single kernel clustering and uniform kernel summation.

In future work, a rigorous comparison to other MKL methods will be carried out. As the appropriate choice of p depends on the dataset, possible ways of automatically adjusting p will be explored. Moreover, implementing nonlinear MKL under the presented framework is in our plans.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc. (2006)
2. Cortes, C., Mohri, M., Rostamizadeh, A.: L2 regularization for learning kernels. In: Conf. on Uncertainty in Artificial Intelligence, pp. 109–116 (2009)
3. Cortes, C., Mohri, M., Rostamizadeh, A.: Learning non-linear combinations of kernels. In: Advances in Neural Information Processing Systems, pp. 396–404 (2009)
4. Filippone, M., Camastra, F., Masulli, F., Rovetta, S.: A survey of kernel and spectral methods for clustering. Pattern Recognition 41(1), 176–190 (2008)
5. Gönen, M., Alpaydin, E.: Multiple kernel learning algorithms. J. of Machine Learning Research 12, 2211–2268 (2011)
6. Kloft, M., Brefeld, U., Sonnenburg, S., Laskov, P., Müller, K.R., Zien, A.: Efficient and accurate lp-norm multiple kernel learning. In: Advances in Neural Information Processing Systems, pp. 997–1005 (2009)
7. Lange, T., Buhmann, J.M.: Fusion of similarity data in clustering. In: Advances in Neural Information Processing Systems, pp. 723–730 (2005)
8. Tzortzis, G., Likas, A.: The global kernel k-means algorithm for clustering in feature space. IEEE Trans. on Neural Networks 20(7), 1181–1194 (2009)
9. Valizadegan, H., Jin, R.: Generalized maximum margin clustering and unsupervised kernel learning. In: Advances in Neural Information Processing Systems, pp. 1417–1424 (2006)
10. Varma, M., Babu, B.R.: More generality in efficient multiple kernel learning. In: Int. Conf. on Machine Learning, pp. 1065–1072 (2009)
11. Xu, Z., Jin, R., Yang, H., King, I., Lyu, M.R.: Simple and efficient multiple kernel learning by group lasso. In: Int. Conf. on Machine Learning, pp. 1175–1182 (2010)
12. Zeng, H., Cheung, Y.-M.: Kernel Learning for Local Learning Based Clustering. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009, Part I. LNCS, vol. 5768, pp. 10–19. Springer, Heidelberg (2009)
13. Zhao, B., Kwok, J.T., Zhang, C.: Multiple kernel clustering. In: SIAM Int. Conf. on Data Mining, pp. 638–649 (2009)