Multi-view Kernel Completion

Sahely Bhadra · Samuel Kaski · Juho Rousu

Abstract In this paper, we introduce the first method that (1) can complete kernel matrices with completely missing rows and columns as opposed to individual missing kernel values, with help of information from other incomplete kernel matrices. Moreover, (2) the method does not require any of the kernels to be complete *a priori*, and (3) can tackle non-linear kernels. The kernel completion is done by finding, from the set of available incomplete kernels, an appropriate set of related kernels for each missing entry. These aspects are necessary in practical applications such as integrating legacy data sets, learning under sensor failures and learning when measurements are costly for some of the views. The proposed approach predicts missing rows by modelling both within-view and between-view relationships among kernel values. For within-view learning, we propose a new kernel approximation that generalizes and improves Nyström approximation. We show, both on simulated data and real case studies, that the proposed method outperforms existing techniques in the settings where they are available, and extends applicability to new settings.

Sahely Bhadra
Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Finland.
Tel.: +358-46-9235652
E-mail: firstname.lastname@aalto.fi

Samuel Kaski
Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Finland.
Tel.: +358-50-3058694
E-mail: firstname.lastname@aalto.fi

Juho Rousu
Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Finland.
Tel.: +358-50-4151702
E-mail: firstname.lastname@aalto.fi

1 Introduction

In recent years, many methods have been proposed for multi-view learning, i.e, learning with data collected from multiple sources or "views" to utilize the complementary information in them. Kernelized methods capture the similarities among data points in a kernel matrix. The multiple kernel learning (MKL) framework (c.f. Gönen and Alpaydin, 2011) is a popular way to accumulate information from multiple data sources, where kernel matrices built on features from individual views are combined for better learning. In MKL methods, it is commonly assumed that full kernel matrices for each view are available. However, in partial data analytics, it is common that information from some sources is not available for some data points.

The incomplete data problem exists in a wide range of fields, including social sciences, computer vision, biological systems, and remote sensing. For example, in remote sensing, some sensors can go off for periods of time, leaving gaps to data. A second example is that when integrating legacy data sets, some views may not available for some data points, because integration needs were not considered when originally collecting and storing the data. For instance, gene expression may have been measured for some of the biological samples, but not for others, and as biological sample material has been exhausted, the missing measurements cannot be made any more. On the other hand, some measurements may be too expensive to repeat for all samples; for example, patient's genotype may be measured only if a particular condition holds. All these examples introduce missing views, i.e, all features of a view for a data point can be missing simultaneously.

Novelty in problem definition: Previous methods for kernel completion have addressed completion of the aggregated Gaussian kernel matrix by integrating multiple incomplete kernels (Williams and Carin, 2005) or single-view kernel completion assuming individual missing values (Graepel, 2002; Paisley and Carin, 2010), or required at least one complete kernel with a full eigen-system to be used as an auxiliary data source (Tsuda et al, 2003; Trivedi et al, 2005), or assume the eigen-system of two kernels to be exactly the same (Shao et al, 2013), or assumed a linear kernel approximation (Lian et al, 2015). Williams and Carin (2005) do not complete the individual incomplete kernel matrix but complete only aggregated kernels when all kernels are Gaussian. Due to absence of full rows/columns in the incomplete kernel matrices, no existing or non-existing single-view kernel completion method (Graepel, 2002; Paisley and Carin, 2010) can be applied to complete kernel matrices of individual views independently. In the multi-view setting, Tsuda et al (2003) have proposed an expectation maximization based method to complete an incomplete kernel matrix for a view, with the help of a complete kernel matrix from another view. As it requires a full eigensystem of the auxiliary full kernel matrix, that method cannot be used to complete a kernel matrix with missing rows/columns when no other auxiliary complete kernel matrix is available. Both Trivedi et al (2005) and Shao et al (2013) match kernels through their Graph Laplacians, which may not work optimally if the kernels have different eigen-structures arising from different types of measurements.The method by Shao et al (2013) completes multiple kernels sequentially, making an implicit assumption that the adjacent kernels in the sequence are related. This can be a

hard constraint and in general may not match the reality. On the other hand, Lian et al (2015) proposed a generative model based method which approximates the similarity matrix for each view as a linear kernel in some low-dimensional space. Therefore, it is unable to model highly non-linear kernels such as RBFs.

Contribution: In this paper, we propose a novel method to complete all incomplete kernel matrices collaboratively, by learning both between-view and within-view relationships among the kernel values (Figure 1). We model between-view relationships in the following two ways: (1) Initially, adapting the strategies from multiple kernel learning (Argyriou et al, 2005; Cortes et al, 2012), we complete kernel matrices by expressing individual normalized kernel matrices corresponding to each view as a convex combination of normalized kernel matrices of other views. (2) Second, to model relationships between kernels having different eigensystems we propose a novel approach of restricting the local embedding of one view in the convex hull of local embeddings of other views. We relate theoretically the kernel approximation quality of the different approaches to the properties of the underlying eigen-spaces of the kernels, pointing out settings where different approaches are optimal.

For within-view learning, we begin from the concept of local linear embedding (Roweis and Saul, 2000), reconstructing each feature representation for a kernel as a sparse linear combination of other available feature representations or "basis" vectors in the same view. We assume the local embeddings, i.e., the reconstruction weights and the basis vectors for reconstructing each samples, are similar across views. In this approach, the non-linearity of kernel functions of individual views is also preserved in the basis vectors. We show that the proposed within-view kernel reconstruction can be seen to generalizing and improving the Nyström method (Williams and Seeger, 2001) which have been successfully applied to efficient kernel learning. Most importantly, for a general single kernel matrix the proposed scheme results into optimal low rank approximation which is not reached by the Nyström method.

For between-view learning, we recognize that the similarity of the eigen-spaces of the views plays a crucial role. When the different kernels have similar optimal low-rank approximations, we show that predicting *kernel values* across views is a potent approach. For this case, we propose a method relying on a technique previously used in multiple kernel learning literature, namely, restricting a kernel matrix into the convex hull of other kernel matrices (Argyriou et al, 2005; Cortes et al, 2012). Here, we use this technique for simultaneously completing multiple incomplete kernel matrices.

For the case when the eigen-systems of the different views differ, we propose methods that, instead of kernel values, translate the *reconstruction weights* across views. For the cases where the leading eigen-vectors are similar but eigen-value spectra are different, we show that it is sufficient to maintain one global set of reconstruction weights, used in the within-view reconstructions of all views. In the case of heterogeneous leading eigen-vector sets across views, we propose to learn the reconstruction weights for each view using the reconstruction weights of the other views.
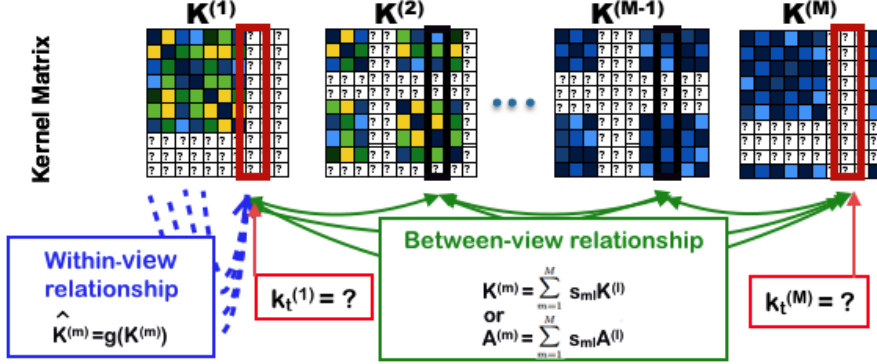
2 Multi-view kernel completion

Fig. 1 We assume N data samples with M views, with a few samples missing from each individual view, and consequently corresponding rows and columns are missing (denoted by '?') in kernel matrices ($\mathbf{K}^{(m)}$). The proposed method predicts the missing kernel rows/columns (e.g., the tth column in views 1 and m) with the help of other samples of the same view (within-view relationship, blue arrows) and the corresponding sample in other views (between-view relationship, green arrows).

We assume N data samples $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ from a multi-view input space $\mathscr{X} = \mathscr{X}^1 \times \cdots \times \mathscr{X}^{(M)}$, where $\mathscr{X}^{(m)}$ is the input space generating the mth view. We denote by $\mathbf{X}^{(m)} = \{\mathbf{x}_1^{(m)}, \ldots, \mathbf{x}_N^{(m)}\}, \forall\, m = 1, \ldots, M$, the set of samples for the mth view, where $\mathbf{x}_i^{(m)} \in \mathscr{X}^{(m)}$ is the ith observation in the mth view and $\mathscr{X}^{(m)}$ is the input space. For simplicity of notation we sometimes omit the superscript (m) denoting the different views when there is no need to refer to several views at a time.

Considering an implicit mapping of the observations of the mth view to an inner product space $\mathscr{F}^{(m)}$ via a mapping $\phi^{(m)} : \mathscr{X}^{(m)} \to \mathscr{F}^{(m)}$, and following the usual recipe for kernel methods (Bach et al, 2004), we specify the kernel as the inner product in $\mathscr{F}^{(m)}$. The kernel value between the ith and j^{th} data points is defined as $k_{ij}^{(m)} = \langle \phi_i^{(m)}, \phi_j^{(m)} \rangle$, where $\phi_i^{(m)} = \phi^{(m)}(\mathbf{x}_i^{(m)})$ and $k_{ij}^{(m)}$ is an element of $\mathbf{K}^{(m)}$, the kernel Gram matrix for the set $\mathbf{X}^{(m)}$.

In this paper we make the assumption that a subset of samples is observed in each view, and correspondingly, a subset of views is observed for each sample. Let $I = [1, \ldots, N]$ be the set of indices of all data points and $I^{(m)}$ be the set of indices of all available data points in the mth view. Hence for each view, only a kernel sub-matrix ($\mathbf{K}_{I^{(m)}I^{(m)}}^{(m)}$) corresponding to the rows and columns indexed by $I^{(m)}$ is observed. Our aim is to predict a complete positive semi-definite (PSD) kernel matrix ($\hat{\mathbf{K}}^{(m)} \in \mathbb{R}^{N \times N}$) corresponding to each view. The crucial task is to predict the missing (t^{th}) rows and columns of $\hat{\mathbf{K}}^{(m)}$, for all $t \in \{I/I^{(m)}\}$. Our approach for predicting $\hat{\mathbf{K}}^{(m)}$ is based on learning both between-view and within-view relationships among the kernel values (Figure 1). The sub-matrix $\hat{\mathbf{K}}_{I^{(m)}I^{(m)}}^{(m)}$ should be approximately equal to

the observed matrix $\mathbf{K}^{(m)}_{I^{(m)}I^{(m)}}$, however, in our approach, approximation quality of the two parts of the kernel matrix can be traded.

2.1 Within-view kernel relationships

For within-view learning, relying on the concept of local linear embedding (Roweis and Saul, 2000), we reconstruct the feature map of tth data point ϕ_t by a sparse linear combination of observed data samples

$$\hat{\phi}_t = \sum_{i \in I} a_{it} \phi_i$$

where $a_{it} \in \mathbb{R}$ is the reconstruction weight of the ith feature representation for representing the tth sample. Hence, approximated kernel values can be expressed as

$$\hat{k}_{tt'} = \langle \hat{\phi}_t, \hat{\phi}_{t'} \rangle = \sum_{i,j \in I} a_{it} a_{jt'} \langle \phi_i, \phi_j \rangle.$$

We note that the above formulation retains the non-linearity of the feature map ϕ and the corresponding kernel. We collect all reconstruction weights of a view into the matrix $\mathbf{A} = (a_{ij})_{i,j=1}^{N}$. Further, by $\mathbf{A}_I$ we denote the sub-matrix of $\mathbf{A}$ containing the rows indexed by I, the known data samples in the view. Thus the reconstructed kernel matrix $\hat{\mathbf{K}}$ can be written as

$$\hat{\mathbf{K}} = \mathbf{A}_I^T \mathbf{K}_{II} \mathbf{A}_I = g(\mathbf{K}). \tag{1}$$

Note that $\hat{\mathbf{K}}$ is positive semi-definite when $\mathbf{K}$ is positive semi-definite. Thus, a by-product of this approximation is that in optimization, PSD property is automatically guaranteed without inserting explicit positive semi-definiteness constraints.

Intuitively, the reconstruction weights are used to extend the known part of the kernel to the unknown part, in other words, the unknown part is assumed to reside within the span of the known part.

We further assume that in each view there exists a sparse embedding in $\mathscr{F}$, given by a small set of samples $B \subset I$, called a basis set, that is able to represent all possible feature representations in that particular view. Thus the non-zero reconstruction weights are confined to the basis set: $a_{ij} \neq 0$ only if $i \in B$. To select such a sparse set of reconstruction weights, we regularize the reconstruction weights by the $\ell_{2,1}$ norm (Argyriou et al, 2006) of the reconstruction weight matrix,

$$\|\mathbf{A}_I\|_{2,1} = \sum_{i \in I} \sqrt{\sum_{j \in I} (a_{ij})^2}. \tag{2}$$

Finally, for the observed part of the kernel, we add the additional objective that the reconstructed kernel values closely approximate the known or observed values. To this end, we define a loss function measuring the within-view approximation error for each view as

$$Loss_{within} = \|\hat{\mathbf{K}}_{II} - \mathbf{K}_{II}\|_2^2. \tag{3}$$

Hence, for individual views the observed part of a kernel is approximated by

$$\hat{\mathbf{K}}_{II} = \mathbf{A}_{II}^{*T} \mathbf{K}_{II} \mathbf{A}_{II}^{*} \tag{4}$$

where the reconstruction weights $\mathbf{A}_{II}^{*}$ (here the superscript $*$ indicates the optimum values) are optimized using (2) and (3) by

$$\mathbf{A}_{II}^{*} = \arg\min_{\mathbf{A}_{II}} \|\mathbf{A}_{II}^{T} \mathbf{K}_{II} \mathbf{A}_{II} - \mathbf{K}_{II}\|_{2}^{2} + \lambda \|\mathbf{A}_{II}\|_{2,1} \tag{5}$$

Where λ is user defined hyper-parameter.

   Without the $\ell_{2,1}$ regularization, the above approximation loss could be trivially optimized by choosing $\mathbf{A}_{II}$ as the identity matrix. The $\ell_{2,1}$ regularization will have the effect of zeroing out some of the diagonal values and introducing non-zeros to the sub-matrix $\mathbf{A}_{BI}$, corresponding to the rows and columns indexed by B and I respectively, where $B = \{i|a_{ii} \neq 0\}$

   In section 3 we show by Theorem 1 that (5) corresponds to a generalized form of the Nyström method (Williams and Seeger, 2001) which is a sparse kernel approximation method that has been successfully applied to efficient kernel learning. Nyström method finds a small set of vectors (not necessarily linearly independent) spanning the kernel, whereas our method searches for linearly independent basis vectors (c.f. Section 3, Lemma 1) and optimizes the reconstruction weights for the data samples. In particular, we show that (5) achieves the best rank-r approximation of a kernel, when the original kernel has rank higher than r, which is not achieved by Nyström method (c.f. Theorem 2).

2.2 Between-view kernel relationships

For a completely missing row or column of a kernel matrix, there is not enough information available for completing it within the same view, and hence the completion needs to be based on other information sources, in our case the other views where the corresponding kernel parts are known. In the following, we introduce two approaches for relaying information of the other views for completing the unknown rows/columns of a particular view. The first technique is based on learning a convex combination of the kernels, extending the multiple kernel learning (Argyriou et al, 2005; Cortes et al, 2012) techniques to kernel completion. The second technique is based on learning reconstruction weights so that they share information between the views.

Between-view learning of kernel values. In multi-view kernel completion the perhaps simplest situation arises when the kernels of the different views are similar, i.e.

$$\hat{\mathbf{K}}^{(m)} \approx \hat{\mathbf{K}}^{(l)}, \ \forall \ l \text{ and } m. \tag{6}$$

In in this case predicting kernel values across views may lead to good kernel approximation. One way to model the similarity is to require the kernels of the different

views to have the similar low-rank approximations. In Theorem 3 we show that optimal rank-r approximation can be achieved if the kernels have the same 'true' rank-r approximation and the kernels themselves have rank at least r.

However, this is probably an overly restrictive assumption in most applications. Thus, in our approach we allow the views to have different approximate kernel matrices with a parametrized relationship learned from data. To learn between-view relationships we express the individual normalized kernel matrix ($\frac{k_{tt'}}{\sqrt{k_{tt},k_{t't'}}}$) corresponding to each view as a convex combination of normalized kernel matrices of the other views. Hence the proposed model learns kernel weights $\mathbf{S} = (s_{ml})_{m,l=1}^M$ between all pairs of kernels (m,l) such that

$$\hat{\mathbf{K}}^{(m)} \approx \sum_{l=1,l\neq m}^M s_{ml}\hat{\mathbf{K}}^{(l)}, \tag{7}$$

where the kernel weights are confined to a convex combination

$$\mathscr{S} = \{\mathbf{S}|s_{ml} \geq 0, \sum_{l=1,l\neq m}^M s_{ml} = 1\}.$$

The kernel weights then can flexibly pick up a subset of relevant views to the current view m. This gives us between-view loss as

$$Loss_{between}^{(m)}(\hat{\mathbf{K}},\mathbf{S}) = \|\hat{\mathbf{K}}^{(m)} - \sum_{l=1,l\neq m}^M s_{ml}\hat{\mathbf{K}}^{(l)}\|_2^2. \tag{8}$$

Previously, Argyriou et al (2005) have proposed a method for learning kernels by restricting the search in the convex hull of a set of given kernels to learn parameters of individual kernel matrices. Here, we apply the idea to kernel completion, which has not been previously considered. We further note that kernel approximation as a convex combination has the interpretation of avoiding extrapolation in the space of kernels, and can be interpreted as a type of regularization to constrain the otherwise flexible set of PSD kernel matrices.

Between-view learning of reconstruction weights. In practical applications, the kernels arising in a multi-view setup might be very heterogeneous in their distributions. In such cases, it might not be realistic to find a convex combination of other kernels that are closely similar to the kernel of a given view. In particular, when the eigenspectra of the kernels are very different, we expect a low between-view loss (8) to be hard to achieve.

Here we assume the approximated kernel matrices have related eigen-spaces in that the eigen-vectors of the related kernels can be written as linear combinations of eigen vectors of the others, but each of them have their own set of eigen-values. In other words,

$$\hat{\mathbf{K}}^{(m)} = \mathbf{U}^{(m)}\Sigma^{(m)}\mathbf{U}^{(m)^T} = \mathbf{U}^{(1)}\mathbf{T}^{(m)}\Sigma^{(m)}\mathbf{T}^{(m)^T}\mathbf{U}^{(1)^T} \tag{9}$$

where $\mathbf{U}^{(m)}$ contains eigen-vectors of $\mathbf{K}^{(m)}$ in its column and $\Sigma^{(m)}$ contains corresponding eigen-values in its diagonal. $\mathbf{T}^{(m)}$ is a linear operator such that $\mathbf{U}^{(m)} = \mathbf{U}^{(1)}\mathbf{T}^{(m)}$. Above the matrices $\mathbf{T}^{(m)}$ allow rotations of the eigen-vectors while the scaling is of them is governed by the matrices $\Sigma^{(m)}$.

For this situation, we propose and alternative approach, where instead of the kernel values, we assume that the basis sets and the reconstruction weights have between-view dependencies that we can learn. Theorem 4 claims when kernels of all views satisfy (9) then learning a set of reconstruction weights, used in in all views, i.e.

$$\mathbf{A}^{(1)} = \ldots = \mathbf{A}^{(M)} \tag{10}$$

gives better approximation than learning a convex combination of kernels as in (7).

However, assuming that kernel functions in *all* the views have similar eigen-vectors is also unrealistic for many real world data-sets with heterogeneous sources and kernels applied to them. On the contrary, it is quite possible that only for a subset of views the eigen-vectors of approximated kernel are linearly related. Thus, in our approach we allow the views to have different reconstruction weights, but assume a parametrized relationship learned from data. This also allows the model to find an appropriate set of related kernels from the set of available incomplete kernels, for each missing entry.

To capture the relationship, we assume the reconstruction weights in a view can be approximated by a convex combination of the reconstruction weights of the other views,

$$\mathbf{A}_{I^{(m)}}^{(m)} \approx \sum_{l=1,l \neq m}^{M} s_{ml}\mathbf{A}_{I^{(m)}}^{(l)}, \tag{11}$$

where the coefficients s_{ml} are defined as in (7). This gives us between-view loss for reconstruction weights as

$$Loss_{between}^{(m)}(\mathbf{A},\mathbf{S}) = \|\mathbf{A}_{I^{(m)}}^{(m)} - \sum_{l=1,l \neq m}^{M} s_{ml}\mathbf{A}_{I^{(m)}}^{(l)}\|_2^2. \tag{12}$$

The reconstructed kernel is thus given by

$$\hat{\mathbf{K}}^{(m)} = \left(\sum_{l=1,l \neq m}^{M} s_{ml}\mathbf{A}_{I^{(m)}}^{(l)^T} \right) \mathbf{K}_{I^{(m)}I^{(m)}}^{(m)} \left(\sum_{l=1,l \neq m}^{M} s_{ml}\mathbf{A}_{I^{(m)}}^{(l)} \right). \tag{13}$$

3 Theoretical Analysis

In this section, we present the theoretical results underlying our methods. We begin by showing the relationship and advantages of our within-kernel approximation to the Nyström method, and follow with theorems establishing the approximation quality of different kernel completion models.

3.1 Rank of the within-kernel approximation

We begin analysis of the rank of the proposed within-kernel approximation method, given in (4) and (5). It approximates the individual kernels as $\hat{\mathbf{K}} = \mathbf{A}^{*T}\mathbf{K}\mathbf{A}^*$ where

$$\mathbf{A}^* = \underset{\mathbf{A}}{\arg\min}\|\mathbf{A}^T\mathbf{K}\mathbf{A} - \mathbf{K}\|_2^2 + \lambda\|\mathbf{A}\|_{2,1}. \qquad (14)$$

For the purposis of the analysis, we derive an equivalent form that reveals the rank behaviour of the method more easily. Above, the matrix $\mathbf{A}$ simultaneously indicates the position of feature maps in the underlying subspace and also selects the basis vectors for defining these subspaces. Hence $\mathbf{A}$ can be written as convolution of two operators $\mathbf{A} = \mathbf{P}\hat{\mathbf{A}}$ where $\mathbf{P} = diag(\mathbf{p})$ and $\mathbf{p} \in \{0,1\}^N$. Here $\mathbf{p}$ acts as a selector operator such that $\mathbf{p}_i = 1$ if $i \in B$ or the ith feature-map is selected as a basis vectors and all other elements of $\mathbf{p}$ is assigned zeros values. $\hat{\mathbf{A}}$ is a matrix of size $\mathbf{A}$, such that $\hat{\mathbf{A}}_{i\in B} = \mathbf{A}_{i\in B}$ and others elements are zeros.

The $\ell_{2,1}$ norm on $\mathbf{A}$ in (14) assigns zeros to few rows of the matrix $\mathbf{A}$; equivalently ℓ_1 norm on selection operator ($\mathbf{p}$) fulfils the same purpose. Again, after rows selection are done both $\hat{\mathbf{A}}_{i\in B}$ and $\mathbf{A}_{i\in B}$ denotes reconstruction weight for kernel by using selected rows (B) and would behave similarly. Therefore the (14) is equivalent to

$$[\hat{\mathbf{A}}^*, \mathbf{P}^*] = \underset{\hat{\mathbf{A}},\mathbf{P}=diag(\mathbf{p})}{\arg\min}\|\hat{\mathbf{A}}^T\mathbf{P}^T\mathbf{K}\mathbf{P}\hat{\mathbf{A}} - \mathbf{K}\|_2^2 + \lambda\|\mathbf{p}\|_1 + \lambda_A\|\hat{\mathbf{A}}\|_2^2. \qquad (15)$$

To see the equivalence, note that at optimum, the rows of $\hat{A}$ that are not selected by P will be identically zero, since the value of the first term of the objective only depends on the selected rows.

Now, the approximated kernel can be written as

$$\begin{aligned}
\hat{\mathbf{K}} &= \hat{\mathbf{A}}^{*T}\mathbf{P}^{*T}\mathbf{K}\mathbf{P}^*\hat{\mathbf{A}}^* \\
&= \hat{\mathbf{A}}_B^{*T}\mathbf{W}^*\hat{\mathbf{A}}_B^*,
\end{aligned} \qquad (16)$$

where, $\mathbf{W}^*$ and $\hat{\mathbf{A}}_B^*$ are non-zero sub-matrices of $\mathbf{P}^{*T}\mathbf{K}\mathbf{P}^*$ (corresponding to B rows and B columns) and $\hat{\mathbf{A}}^*$ (corresponding to B rows) respectively.

Lemma 1 *For $rank(\mathbf{K}) \geq r$, $\exists\ \lambda^r$ and λ_A such that the solution of (15) selects a $\mathbf{W}^* \in \mathbb{R}^{r\times r}$ with $rank(\mathbf{W}^*) = r$.*

Proof When $rank(\mathbf{K}) \geq r$ then there must exist a rank-r submatrix of $\mathbf{K}$ of size $r \times r$. Again, in case of ℓ_1 regularization on binary vector $\mathbf{p}$, one can tuned λ to λ^r to have required sparsity on $\mathbf{p}$, i.e., $\|\mathbf{p}^*\|_1 = r$. Moreover, ℓ_1 regularization on binary vector ensures the linearly independence of selected columns and rows when λ_A is significantly small. Because if a solution of (15) selects a column which is linearly dependent on other selected columns, then the solution, in compare to some other solution which selects the same columns except this linearly dependent column, will increase the value of ℓ_1 norm regularizer while keeping the first part of the objective function same and lowering the third part of (15) insignificantly. Hence that can not be a optimum solution. This completes the proof.

3.2 Relation to Nyström approximation

Nyström method (Williams and Seeger, 2001) approximates the kernel matrix $\mathbf{K}$ as

$$\hat{\mathbf{K}}_{nys} = \mathbf{CW}^{-1}\mathbf{C}^T \tag{17}$$

where the matrix $\mathbf{C} \in \mathbb{R}^{N \times c}$ consists of c randomly chosen columns of kernel $\mathbf{K}$ and $\mathbf{W} \in \mathbb{R}^{c \times c}$ is a matrix consisting of the intersection of those c columns with the corresponding c rows. Due to the random selection, Nyström method, unlike ours (as established above), does not in general produce linearly independent set of vectors. One can re-write

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{W} \\ \mathbf{C}_2 \end{bmatrix}$$

and then the Nyström approximation as

$$\hat{\mathbf{K}}_{nys} = \begin{bmatrix} \mathbf{W} \\ \mathbf{C}_2 \end{bmatrix} \mathbf{W}^{-1} \begin{bmatrix} \mathbf{W} \ \mathbf{C}_2^T \end{bmatrix} = \begin{bmatrix} \mathbf{W} & \mathbf{C}_2^T \\ \mathbf{C}_2 & \mathbf{C}_2\mathbf{W}^{-1}\mathbf{C}_2^T \end{bmatrix} \tag{18}$$

For non-invertible $\mathbf{W}$, its pseudo inverse can be used instead of $\mathbf{W}^{-1}$.

Theorem 1 *The Nyström approximation of $\mathbf{K}$ is a feasible solution of (15), i.e., for invertible $\mathbf{W}$, $\exists \hat{\mathbf{A}} \in \mathbb{R}^{N \times c}$ such that $\hat{\mathbf{K}}_{nys} = \hat{\mathbf{A}}^T\mathbf{W}\hat{\mathbf{A}}$.*

Proof Equate

$$\hat{\mathbf{A}}^T = \begin{bmatrix} I_c \\ \mathbf{C}_2\mathbf{W}^{-1} \end{bmatrix}$$

in (18) where I_c denotes the identity matrix of size c.

The above theorem shows that the approach of (15), by finding the optimal feasible solution, will always produce better kernel approximation with the same level of sparsity as the Nyström method.

3.3 Low-rank approximation quality

Nyström approximation satisfies following low-rank approximation properties (Kumar et al, 2009):

– If $r = rank(\mathbf{K}^{(m)}) \leq c$ and $rank(\mathbf{W}^{(m)}) = r$, then the Nyström approximation is exact, i.e., $\|\mathbf{K} - \hat{\mathbf{K}}_{nys}^{(m)}\|_2^2 = 0$.
– For general $\mathbf{K}^{(m)}$ when $rank(\mathbf{K}^{(m)}) \geq r$ and $rank(\mathbf{W}^{(m)}) = r$, then the Nyström approximation is not the best rank-r approximation of $\mathbf{K}^{(m)}$.

Below, we will establish that our approach will result in the best rank-k approximation also in the general case where the original kernel has rank higher than k.

Lemma 2 *If $\mathbf{K}_r$ be the best rank-r approximation of a kernel $\mathbf{K}$ with $rank(\mathbf{K}) \geq r$ and $\mathbf{W}$ be a full rank sub-matrix of $\mathbf{K}$ of size $r \times r$, i.e, $rank(\mathbf{W}) = r$. Then $\exists \hat{\mathbf{A}} \in \mathbb{R}^{N \times r}$ such that for the proposed approximation $\hat{\mathbf{K}} = \hat{\mathbf{A}}^T\mathbf{W}\hat{\mathbf{A}}$ is equivalent to $\mathbf{K}_r$, i.e., $\|\mathbf{K} - \hat{\mathbf{K}}\|_2^2 = \|\mathbf{K} - \mathbf{K}_r\|_2^2$.*

Proof Using eigenvalue decomposition one can write

$$\mathbf{K}_r = \mathbf{U}\mathbf{U}^T \quad \text{and} \quad \mathbf{W} = \mathbf{U}_W\mathbf{U}_W^T$$

where $\mathbf{U} \in \mathbb{R}^{N \times r}$ and $rank(\mathbf{U}) = r$ and $\mathbf{U}_W \in \mathbb{R}^{r \times r}$ and $rank(\mathbf{U}_W) = r$.

Using invertible property of $\mathbf{U}_W$, one can express $\mathbf{U}$ as $\mathbf{U} = (\mathbf{U}\mathbf{U}_W^{-1})\mathbf{U}_W$.

$$\begin{aligned}
\mathbf{K}_r &= \mathbf{U}\mathbf{U}^T \\
&= (\mathbf{U}\mathbf{U}_W^{-1})(\mathbf{U}_W\mathbf{U}_W^T)(\mathbf{U}\mathbf{U}_W^{-1})^T \\
&= \hat{\mathbf{A}}^T\mathbf{W}\hat{\mathbf{A}}
\end{aligned}$$

where $\hat{\mathbf{A}}^T = \mathbf{U}\mathbf{U}_W^{-1}$.

Theorem 2 *If $rank(\mathbf{K}) \geq r$, then $\exists \lambda^r$ such that the proposed approximation $\hat{\mathbf{K}}$ in (16) is equivalent to the best rank-r approximation of $\mathbf{K}$, i.e., $\|\mathbf{K} - \hat{\mathbf{K}}\|_2^2 = \|\mathbf{K} - \mathbf{K}_r\|_2^2$, where $\mathbf{K}_r$ is the best rank-r approximation of $\mathbf{K}$.*

Proof Lemma 1 proves that there exist a λ_r for which the optimum solution of (15) results into a $\mathbf{W}^* \in \mathbb{R}^{r \times r}$ such that $rank(\mathbf{W}^*) = r$. According to Lemma 2 there exist also a feasible $\hat{\mathbf{A}}$ which reconstructs $\mathbf{K}_r$. Let us assume $\hat{\mathbf{A}}^*$ is the optimum solution of (15) with $\lambda_A = 0$, then

$$\begin{aligned}
\|\mathbf{K} - \hat{\mathbf{A}}^{*T}\mathbf{W}^*\hat{\mathbf{A}}^*\|_2^2 &\leq \|\mathbf{K} - \hat{\mathbf{A}}^T\mathbf{W}^*\hat{\mathbf{A}}\|_2^2 \\
\|\mathbf{K}^-\hat{\mathbf{K}}\|_2^2 &\leq \|\mathbf{K} - \mathbf{K}_r\|_2^2.
\end{aligned}$$

This completes the proof.

3.4 Low-rank approximation quality of multiple kernel matrices

In this section, we establish the approximation quality achieved in the multi-view setup, when the different kernels are similar either in the sense of having the same underlying 'true' low-rank approximations (Theorem 3) or more generally similar sets of eigen-vectors (Theorem 4).

Theorem 3 *Assume $\mathbf{K}^{(1)}, \ldots, \mathbf{K}^M$ are M kernel matrices such that $\forall m$, $rank(\mathbf{K}^m) \geq r$ and that all of them have the same rank-r approximation, i.e., $\mathbf{K}_r^{(1)} = \ldots = \mathbf{K}_r^{(m)}$ (assumption in (6)). Then $\exists \lambda^r$ and λ_A such that the following optimization problem:*

$$\begin{aligned}
\min_{\hat{\mathbf{A}}^{(m)}, \mathbf{P}} \quad & \sum_{m=1}^{M} \|\mathbf{K}^{(m)} - \hat{\mathbf{A}}^{(m)T}\mathbf{P}^T\mathbf{K}^{(m)}\mathbf{P}\hat{\mathbf{A}}^{(m)}\|_2^2 + \lambda^r \sum_{m=1}^{M} \|\mathbf{p}\|_1 + \lambda_A \sum_{m=1}^{M} \|\mathbf{A}^{(m)}\|_2^2 \\
s.t. \quad & \hat{\mathbf{A}}^{(m)T}\mathbf{P}^T\mathbf{K}^{(m)}\mathbf{P}\hat{\mathbf{A}}^{(m)} = \hat{\mathbf{A}}^{(l)T}\mathbf{P}^T\mathbf{K}^{(l)}\mathbf{P}\hat{\mathbf{A}}^{(l)} \quad \forall m \text{ and } l \\
& \mathbf{P} = diag(\mathbf{p}) \text{ and } \mathbf{p}_i \in \{0, 1\}
\end{aligned} \tag{19}$$

produces the exact rank-r approximation for individual matrices, i.e., $\sum_{m=1}^{M} \|\mathbf{K}^{(m)} - \hat{\mathbf{A}}^{(m)T}\mathbf{P}^T\mathbf{K}^{(m)}\mathbf{P}\hat{\mathbf{A}}^{(m)}\|_2^2 = \sum_{m=1}^{M} \|\mathbf{K}^{(m)} - \mathbf{K}_r^{(m)}\|_2^2$.

Proof Each symmetric positive semi-definite kernel matrix can be written as

$$\mathbf{K}^{(m)} = \mathbf{X}^{(m)}\mathbf{X}^{(m)^T},$$

where $\mathbf{X}^{(m)} \in \mathbb{R}^{N \times rank(\mathbf{K}^{(m)})}$ and columns of $\mathbf{X}^{(m)}$ are orthogonal to each other.

When all $\mathbf{K}^{(m)}$s have the same rank-r approximation then the first r columns of $\mathbf{X}^{(m)}$ are same for all m. Hence $\mathbf{X}^{(m)}$ can be expressed as

$$\mathbf{X}^{(m)} = \left[\mathbf{X}_r \mathbf{X}_{r^c}^{(m)} \right]$$

where r^c denotes the complement of set r. Here $\mathbf{X}_r \in \mathbb{R}^{N \times r}$ is a rank-r matrix and hence it is possible to find a set of r rows from $\mathbf{X}_r$ which together produce a rank-r sub-matrix of size $r \times r$. let $\mathbf{P}^{*^T}$ be such a selector operator which select r linearly independent rows from $\mathbf{X}_r$, i.e. . Moreover, according to Lemma 1 there exist a λ_r for which the optimization problem in (19) gives the required sparsity in $\mathbf{P}^*$.

Hence ,

$$\mathbf{W}^{(m)} = \mathbf{P}^{*^T}\mathbf{K}^{(m)}\mathbf{P}^* = \mathbf{X}_W^{(m)}\mathbf{X}_W^{(m)^T}$$

where $\mathbf{X}_W^{(m)} = \mathbf{P}^{*^T}\mathbf{X}^{(m)} = \mathbf{P}^{*^T}\left[\mathbf{X}_r\mathbf{X}_{r'}^{(m)}\right]$ and hence $\mathbf{X}_W^{(m)}$ contains r linearly independent rows of $\mathbf{X}^{(m)}$ and hence for all m, $rank(\mathbf{W}^{(m)}) = r$.

When the parameter λ_A is significantly small then, using Theorem 2, we can prove that for a $\mathbf{W}^{(m)}$ with $rank(\mathbf{W}^{(m)}) = r$, there exist a $\hat{\mathbf{A}}^{(m)}$ which is able to generate exact rank-r approximation for individual kernel matrix i.e,

$$\|\mathbf{K}^{(m)} - \hat{\mathbf{A}}^{(m)^T}\mathbf{P}^{*^T}\mathbf{K}^{(m)}\mathbf{P}^*\hat{\mathbf{A}}^{(m)}\|_2^2 = \|\mathbf{K}^{(m)} - \mathbf{K}_r^{(m)}\|_2^2, \ \forall m.$$

This completes the proof.

Theorem 4 *Assume $\mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(M)}$ are M kernel matrices such that $\forall m$, $rank(\mathbf{K}^{(m)}) = r$ and all of them have same eigen-space, i.e, eigen-vectors are linearly transferable and eigen values are different (assumption in (9)), i.e., $\mathbf{K}^{(m)} = \mathbf{U}^{(m)}\Sigma^{(m)}\mathbf{U}^{(m)^T}$ such that $\mathbf{U}^{(m)} = \mathbf{U}^{(1)}\mathbf{T}^{(m)}$. Then $\exists \lambda^r$ and λ_A such that the following optimization problem (by the assumption in (10))*

$$\min_{\hat{\mathbf{A}},\mathbf{P}} \sum_{m=1}^{M} \|\mathbf{K}^{(m)} - \hat{\mathbf{A}}^T\mathbf{P}^T\mathbf{K}^{(m)}\mathbf{P}\hat{\mathbf{A}}\|_2^2 + \lambda^r \sum_{m=1}^{M} \|\mathbf{p}\|_1 + \lambda_A \sum_{m=1}^{M} \|\hat{\mathbf{A}}^{(m)}\|_2^2$$
$$\mathbf{P} = diag(\mathbf{p}) \ and \ \mathbf{p}_i \in \{0,1\} \tag{20}$$

selects a rank-r sub-matrix $\mathbf{W}^{*(m)} \in \mathbb{R}^{r \times r}$ with $rank(\mathbf{W}^{*(m)}) = r$ of each kernel $\mathbf{K}^{(m)}$ which can produce the exact reconstruction for individual matrices, i.e.,

$$\sum_{m=1}^{M} \|\mathbf{K}^{(m)} - \hat{\mathbf{A}}^{*T}\mathbf{W}^{(m)}\hat{\mathbf{A}}^*\|_2^2 = 0$$

Proof According to assumption in (9) each kernel matrix can be written as $\mathbf{K}^{(m)} = \mathbf{U}^{(m)} \Sigma^{(m)} \mathbf{U}^{(m)T} = \mathbf{U}^{(1)} \mathbf{T}^{(m)} \Sigma^{(m)} \mathbf{T}^{(m)T} \mathbf{U}^{(1)T}$ where $\mathbf{U}^{(1)} \in \mathbb{R}^{N \times r}$ is orthonormal. Hence it is possible to find a set of r rows from $\mathbf{U}^{(1)}$ which together produce a rank-r sub-matrix of size $r \times r$. let $\mathbf{P}^{*T}$ be such selector operator which selects r linearly independent rows from $\mathbf{U}^{(1)}$. Let r^* denote the set of indices of such linearly independence rows of $\mathbf{U}^{(1)}$. Hence $\mathbf{U}^{(1)} = \begin{bmatrix} \mathbf{U}_{r*}^{(1)} \\ \mathbf{U}_{r^c}^{(1)} \end{bmatrix}$ and $\mathbf{U}_{r*}^{(1)}$ is invertible.

According to Lemma 1 there exist a λ_r for which the optimization problem (20) gives the required sparsity in $\mathbf{P}^{*T}$. Hence using the assumption $\mathbf{U}^{(m)} = \mathbf{U}^{(1)} \mathbf{T}^{(m)}$ we get,

$$\mathbf{W}^{*(m)} = \mathbf{P}^{*T} \mathbf{K}^{(m)} \mathbf{P}^* = \mathbf{P}^{*T} \mathbf{U}^{(1)} \left(\mathbf{T}^{(m)} \Sigma^{(m)} \mathbf{T}^{(m)T} \right) \mathbf{U}^{(1)T} \mathbf{P}^* = \mathbf{U}_{r*}^{(1)} \left(\mathbf{T}^{(m)} \Sigma^{(m)} \mathbf{T}^{(m)T} \right) \mathbf{U}_{r*}^{(1)T}$$

Hence the optimization problem (20) selects a sub-matrix $\mathbf{W}^{*(m)}$ such that $\mathbf{W}^{*(m)} \in \mathbb{R}^{r \times r}$ and $rank(\mathbf{W}^{*(m)}) = r$. Then each kernel matrix is expressed in terms of $\mathbf{W}^{*(m)}$ as

$$\begin{aligned} \mathbf{K}^{(m)} = \mathbf{U}^{(m)} \Sigma^{(m)} \mathbf{U}^{(m)T} &= \mathbf{U}^{(1)} \left(\mathbf{T}^{(m)} \Sigma^{(m)} \mathbf{T}^{(m)T} \right) \mathbf{U}^{(1)T} \\ &= \left(\mathbf{U}^{(1)} \mathbf{U}_{r*}^{(1)-1} \right) \mathbf{W}^{*(m)} \left(\mathbf{U}^{(1)} \mathbf{U}_{r*}^{(1)-1} \right)^T . \end{aligned}$$

Defining $\hat{\mathbf{A}}^* = \left(\mathbf{U}^{(1)} \mathbf{U}_{r*}^{(1)-1} \right)^T$ we get, $\mathbf{K}^{(m)} = \hat{\mathbf{A}}^{*T} \mathbf{W}^{*(m)} \hat{\mathbf{A}}^*$. This completes the proof.

4 Optimization problems

Here we present the optimization problems for **Multi-view Kernel Completion** (**MKC**), arising from the within-view and between-view kernel approximations described above.

MKC using semi-definite programming (MKC$_{sdp}$): This is the most general case where we do not put any other restrictions on kernels of individual views, other than restricting them to be positive semi-definite kernels. In this general case we propagate information from other views by learning between-view relationships depending on kernel values in (7). Hence, using (3) and (8) we get

$$\min_{\substack{\mathbf{S}, \hat{\mathbf{K}}^{(m)}, \\ m=1,\ldots,M}} \sum_{m=1}^{M} \left(\|\hat{\mathbf{K}}_{I^{(m)}I^{(m)}}^{(m)} - \mathbf{K}_{I^{(m)}I^{(m)}}^{(m)}\|_2^2 + \lambda_1 \|\hat{\mathbf{K}}^{(m)} - \sum_{l=1,l\neq m}^{M} s_{ml} \hat{\mathbf{K}}^{(l)}\|_2^2 \right)$$

$$s.t. \quad \hat{\mathbf{K}}^{(m)} \succeq 0 \ \forall m = 1,\ldots,M$$

$$\mathbf{S} \in \mathscr{S}. \tag{21}$$

We solve this non-convex optimization problem by iteratively solving it for $\mathbf{S}$ and $\hat{\mathbf{K}}^{(m)}$ using block-coordinate descent. For a fixed $\mathbf{S}$, to update the $\hat{\mathbf{K}}^{(m)}$'s we need to solve a semi-definite program with M positive constraints.

MKC using homogeneous embeddings (MKC$_{embd(hm)}$): An optimization problem with M positive semi-definite constraints is inefficient for even a data set of size 100. To avoid solving the SDP in each iteration we assume a kernel approximation (1). When kernel functions in different views are not the same and kernel matrices in different views have different eigen-spectra, we learn relationships among underlying embeddings of different views (10), instead of the actual kernel values. Hence, using (3), (1) and (10) along with $\ell_{2,1}$ regularization on $\mathbf{A}$, we get

$$\min_{\substack{\mathbf{A},\hat{\mathbf{K}}^{(m)}, \\ \forall m=1,\dots,M}} \sum_{m=1}^{M} \|\hat{\mathbf{K}}^{(m)}_{I^{(m)}I^{(m)}} - \mathbf{K}^{(m)}_{I^{(m)}I^{(m)}}\|_2^2 + \lambda \|\mathbf{A}\|_{2,1}$$
$$s.t \quad \hat{\mathbf{K}}^{(m)} = \mathbf{A}_{I^{(m)}}^{T} \mathbf{K}^{(m)}_{I^{(m)}I^{(m)}} \mathbf{A}_{I^{(m)}}$$
$$\mathbf{S} \in \mathscr{S} \tag{22}$$

Theorem 4 shows that the above formulation is appropriate when the first few eigenvectors for all kernels are same while corresponding eigen values may be different.

MKC using heterogeneous embeddings (MKC$_{embd(ht)}$): When kernel matrices in different views have different eigen-spectra both in eigenvalues and eigenvectors, we learn relationships among underlying embeddings of different views (11), instead of the actual kernel values. Hence, using (3), (1) and (12) along with $l_{2,1}$ regularization on $\mathbf{A}^{(m)}$, we get

$$\min_{\substack{\mathbf{S},\mathbf{A}^{(m)},\hat{\mathbf{K}}^{(m)}, \\ \forall m=1,\dots,M}} \sum_{m=1}^{M} \left(\|\hat{\mathbf{K}}^{(m)}_{I^{(m)}I^{(m)}} - \mathbf{K}^{(m)}_{I^{(m)}I^{(m)}}\|_2^2 + \lambda_1 \|\mathbf{A}^{(m)}_{I^{(m)}} - \sum_{l=1,l\neq m}^{M} s_{ml}\mathbf{A}^{(l)}_{I^{(m)}}\|_2^2 \right)$$
$$+\lambda_2 \sum_{m=1}^{M} \|\mathbf{A}^{(m)}\|_{2,1}$$
$$s.t \quad \hat{\mathbf{K}}^{(m)} = \mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}^{(m)}_{I^{(m)}I^{(m)}} \mathbf{A}_{I^{(m)}}^{(m)}$$
$$\mathbf{S} \in \mathscr{S} \tag{23}$$

MKC using kernel approximation (MKC$_{app}$): On the other hand when the low rank approximation of related kernels are same (6) then between-view relationships are learnt on kernel values using (8). In this case the kernel is approximate to avoid solving the SDP :

$$\min_{\substack{\mathbf{S},\mathbf{A}^{(m)},\hat{\mathbf{K}}^{(m)}, \\ \forall m=1,\dots,M}} \sum_{m=1}^{M} \left(\|\hat{\mathbf{K}}^{(m)}_{I^{(m)}I^{(m)}} - \mathbf{K}^{(m)}_{I^{(m)}I^{(m)}}\|_2^2 + \lambda_1 \|\hat{\mathbf{K}}^{(m)} - \sum_{l=1,l\neq m}^{M} s_{ml}\hat{\mathbf{K}}^{(l)}\|_2^2 \right)$$
$$+\lambda_2 \sum_{m=1}^{M} \|\mathbf{A}^{(m)}\|_{2,1}$$
$$s.t \quad \hat{\mathbf{K}}^{(m)} = \mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}^{(m)}_{I^{(m)}I^{(m)}} \mathbf{A}_{I^{(m)}}^{(m)}$$
$$\mathbf{S} \in \mathscr{S} \tag{24}$$

Theorem 3 shows that this method results into the exact rank-r approximation when rank-r approximation kernels for related views are same. We solve all the above-mentioned non-convex optimization problems with $l_{2,1}$ regularization by sequentially updating $\mathbf{S}$ and $\mathbf{A}^{(m)}$. In each iteration $\mathbf{S}$ is updated by solving a quadratic program and for each m, $\mathbf{A}^{(m)}$ is updated using proximal gradient descent.

5 Algorithms

Here we present the algorithms for solving various optimization problems, described in previous section.

5.1 Algorithm to solve $\mathbf{MKC}_{embd(ht)}$

In this section the Algorithm 1 describes the algorithm to solve $\mathbf{MKC}_{embd(ht)}$ (23).

Substituting $\hat{\mathbf{K}}^{(m)} = \mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}_{I^{(m)}I^{(m)}}^{(m)} \mathbf{A}_{I^{(m)}}^{(m)}$, the optimization problem (23) contains two sets of unknowns, $\mathbf{S}$ and the $\mathbf{A}^{(m)}$'s. We update $\mathbf{A}^{(m)}$ and $\mathbf{S}$ in an iterative manner. In the kth iteration for a fixed $\mathbf{S}^{k-1}$ from the previous iteration, to update $\mathbf{A}^{(m)}$'s we need to solve following for each m:

$$\mathbf{A}^{(m)^k} = \arg\min_{\mathbf{A}^{(m)}} Aobj_{\mathbf{S}}^k(\mathbf{A}^{(m)}) + \lambda_2 \Omega(\mathbf{A}^{(m)})$$

where $\Omega(\mathbf{A}^{(m)}) = \|\mathbf{A}^{(m)}\|_{2,1}$ and $Aobj_{\mathbf{S}}^k(\mathbf{A}^{(m)}) = \|\mathbf{K}_{I^{(m)}I^{(m)}}^{(m)} - \left[\mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}_{I^{(m)}I^{(m)}}^{(m)} \mathbf{A}_{I^{(m)}}^{(m)}\right]_{I^{(m)}I^{(m)}} \|_2^2 +$ $\lambda_1 \sum_{m=1}^M \|\mathbf{A}^{(m)} - \sum_{l=1,l\neq m}^M s_{ml}^{k-1} \mathbf{A}^{(l)}\|_2^2$.

Instead of solving this problem in each iteration we update $\mathbf{A}^{(m)}$ using proximal gradient descent. Hence, in each iteration,

$$\mathbf{A}^{(m)^k} = Prox_{\gamma\lambda_2\Omega}\left(\mathbf{A}^{(m)^{k-1}} - \gamma\partial Aobj_{\mathbf{S}}^k(\mathbf{A}^{(m)^{k-1}})\right) \tag{25}$$

where $\partial Aobj_{\mathbf{S}}^k(\mathbf{A}^{(m)})$ is the differential of $Aobj_{\mathbf{S}}^k(\mathbf{A}^{(m)})$ at $\mathbf{A}^{(m)^{k-1}}$ and γ is the step size which is decided by a line search. In (25) each row of $\mathbf{A}^{(m)}$ (i.e., $\mathbf{a}_t^{(m)}$) can be solved independently and we apply a proximal operator on each row. Following Bach et al (2011), the solution of (25) is

$$\mathbf{a}_t^{(m)^k} = max\left(0, (1 - \frac{\gamma\lambda_2}{\|\Delta\mathbf{a}_t^{(m)^{k-1}}\|_2})\Delta\mathbf{a}_t^{(m)^{k-1}}\right), \tag{26}$$

where $\Delta\mathbf{a}_t^{(m)^{k-1}}$ is the tth row of $\left(\mathbf{A}^{(m)^{k-1}} - \gamma\partial Aobj_{\mathbf{S}}^k(\mathbf{A}^{(m)^{k-1}})\right)$.

Again, in the kth iteration, for fixed $\mathbf{A}^{(m)^k}$'s, the $\mathbf{S}$ is updated by independently updating each row ($\mathbf{s}_m$) through solving the following Quadratic Program:

$$\mathbf{s}_m^k = \arg\min_{\mathbf{s}_m} Sobj_{\mathbf{A}^{(m)},m=1,\ldots,M}^k(\mathbf{s}_m)$$
$$s.t \quad \sum_{l\neq m} s_{ml} = 1,$$
$$s_{ml} \geq 0 \,\forall l \tag{27}$$

Algorithm 1 . $\text{MKC}_{embd(ht)}\Big(\mathbf{K}^{(m)}, I^{(m)}, \forall m \in [1,\ldots,M]\Big)$

Initialization:
$s_{mm}^0 = 0$, $s_{ml}^0 = \frac{1}{M-1}$,
$\mathbf{A}_{I^{(m)}I^{(m)}}^{(m)0}$ = Identity matrix and
$a_{tt'}^{(m)0} \sim uniform(-1,1)$ for all $t,t' \notin I^{(m)}$

repeat
  for m=1 to M **do**
    for t=1 to N **do**
      $\mathbf{a}_t^{(m)k} = max\left(0, (1 - \frac{\gamma\lambda_2}{\|\Delta\mathbf{a}_t^{(m)k-1}\|_2})\Delta\mathbf{a}_t^{(m)k-1}\right)$

                         [according to (25) and (26)]
                         [γ is fixed by line search]
    end for
  end for

  for m=1 to M **do**
    $\mathbf{s}_m^k = \arg\min_{\mathbf{s}_m} Sobj_{\mathbf{A}^{(m)},m=1,\ldots,M}^k(\mathbf{s}_m)$
                         [according to (27)]
  end for
until convergence

---

where $Sobj_{\mathbf{A}^{(m)},m=1,\ldots,M}^k(\mathbf{s}_m) = \|\mathbf{A}^{(m)k} - \sum_{l=1,l\neq m}^M s_{ml}\mathbf{A}^{(l)k}\|_2^2$.

5.2 Algorithm to solve MKC_{sdp}

In this section the Algorithm 2 describes the algorithm to solve MKC_{sdp}(21). The optimization problem (21) has two sets of unknowns, $\mathbf{S}$ and the $\hat{\mathbf{K}}^{(m)}$'s. We update $\hat{\mathbf{K}}^{(m)}$ and $\mathbf{S}$ in an iterative manner. In the kth iteration, for fixed $\mathbf{S}^{k-1}$ and $\hat{\mathbf{K}}^{(m)k-1}$, the $\hat{\mathbf{K}}^{(m)}$ is updated independently by solving following Semi-definite Programming:

$$\hat{\mathbf{K}}^{(m)k} = \arg\min_{\mathbf{K}} Kobj_{\mathbf{S}}^k(\hat{\mathbf{K}}^{(m)})$$

$$s.t \quad \hat{\mathbf{K}}^{(m)} \succeq 0, \tag{28}$$

where $Kobj_{\mathbf{S}}^k(\hat{\mathbf{K}}^{(m)}) = \|\hat{\mathbf{K}}_{I^{(m)}I^{(m)}}^{(m)} - \mathbf{K}_{I^{(m)}I^{(m)}}^{(m)}\|_2^2 + \lambda\|\hat{\mathbf{K}}^{(m)} - \sum_{l=1,l\neq m}^M s_{ml}^{k-1}\hat{\mathbf{K}}^{(l)k-1}\|_2^2 + \lambda\sum_{l=1,l\neq m}^M \|\left(\hat{\mathbf{K}}^{(l)k-1} - \sum_{l'=1,l'\neq l,m}^M s_{ll'}^{k-1}\hat{\mathbf{K}}^{(l')k-1}\right) - s_{lm}^{k-1}\hat{\mathbf{K}}^{(m)}\|_2^2$

Again, in the kth iteration, for fixed $\hat{\mathbf{K}}^{(m)k}, \forall m = [1,\ldots,M]$, $\mathbf{S}$ is updated by independently updating each row ($\mathbf{s}_m$) through solving the following Quadratic Program:

$$\mathbf{s}_m^k = \arg\min_{\mathbf{s}_m} Sobj_{[psd]\hat{\mathbf{K}}^{(m)},m=1,\ldots,M}^k(\mathbf{s}_m)$$

$$s.t \quad \sum_{l\neq m} s_{ml} = 1,$$

$$s_{ml} \geq 0 \,\forall l \tag{29}$$

Here $Sobj_{[psd]\hat{\mathbf{K}}^{(m)},m=1,\ldots,M}^k(\mathbf{s}_m) = \|\hat{\mathbf{K}}^{(m)k} - \sum_{l=1,l\neq m}^M s_{ml}\hat{\mathbf{K}}^{(l)k}\|_2^2$.

Algorithm 2 MKC$_{sdp}$ $\left(\mathbf{K}^{(m)}, I^{(m)}, \forall m \in [1, \ldots, M] \right)$

Initiaization:
$s_{mm}^0 = 0,\ s_{ml}^0 = \frac{1}{M-1}$,

$\hat{\mathbf{K}}_{I^{(m)} I^{(m)}}^{(m)^0} = \mathbf{K}_{I^{(m)} I^{(m)}}^{(m)}$ and
$\hat{k}_{tt'}^{(m)^0} \sim uniform(-1,1)$ for all $t, t' \notin I^{(m)}$

repeat
    for m=1 to M **do**
      $\hat{\mathbf{K}}^{(m)^k} = \arg\min_{\hat{\mathbf{K}}^{(m)}} Kobj_{\mathbf{S}}^k(\hat{\mathbf{K}}^{(m)})$
                            [according to (28)]
    end for

    for m=1 to M **do**
      $\mathbf{s}_m^k = \arg\min_{\mathbf{s}_m} Sobj_{[psd]\hat{\mathbf{K}}^{(m)}, m=1,\ldots,M}^k (\mathbf{s}_m)$
                            [according to (29)]
    end for

until convergence

5.3 Algorithm to solve **MKC$_{app}$**

In this section the Algorithm 3 describes the algorithm to solve **MKC$_{app}$**(24) which is similar to Algorithm 1. Substituting $\hat{\mathbf{K}}^{(m)} = \mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}_{I^{(m)}, I^{(m)}}^{(m)} \mathbf{A}_{I^{(m)}}^{(m)}$, the optimization problem (24) also has two sets of unknowns, $\mathbf{S}$ and the $\mathbf{A}^{(m)}$'s and again we update $\mathbf{A}^{(m)}$ and $\mathbf{S}$ in an iterative manner. In the kth iteration for a fixed $\mathbf{S}^{k-1}$ from previous iteration, to update $\mathbf{A}^{(m)}$'s, unlike , we need to solve following for each m:

$$\mathbf{A}^{(m)^k} = \arg\min_{\mathbf{A}^{(m)}} Aobj_{[app]\mathbf{S}}^k(\mathbf{A}^{(m)}) + \lambda_2 \Omega(\mathbf{A}^{(m)^{k-1}})$$

where $\Omega(\mathbf{A}^{(m)}) = \|\mathbf{A}^{(m)}\|_{2,1}$ and

$$Aobj_{[app]\mathbf{S}}^k(\mathbf{A}^{(m)}) = \|\mathbf{K}_{I^{(m)} I^{(m)}}^{(m)} - \left[\mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}_{I^{(m)} I^{(m)}}^{(m)} \mathbf{A}_{I^{(m)}}^{(m)} \right]_{I^{(m)} I^{(m)}} \|_2^2$$
$$+ \lambda_1 \sum_{m=1}^{M} \|\mathbf{A}_{I^{(m)}}^{(m)^T} \mathbf{K}_{I^{(m)} I^{(m)}}^{(m)} \mathbf{A}_{I^{(m)}}^{(m)} - \sum_{l=1, l \neq m}^{M} s_{ml}^{k-1} \mathbf{A}_{I^{(l)}}^{(l)^T} \mathbf{K}_{I^{(l)} I^{(l)}}^{(l)} \mathbf{A}_{I^{(l)}}^{(l)} \|_2^2.$$

For this case too, instead of solving this problem in each iteration we update $\mathbf{A}^{(m)}$ using proximal gradient descent. Hence, in each iteration,

$$\mathbf{A}^{(m)^k} = Prox_{\gamma \lambda_2 \Omega} \left(\mathbf{A}^{(m)^{k-1}} - \gamma \partial Aobj_{[app]\mathbf{S}}^k(\mathbf{A}^{(m)^{k-1}}) \right) \tag{30}$$

where $\partial Aobj_{[app]\mathbf{S}}^k(\mathbf{A}^{(m)^{k-1}})$ is the differential of $Aobj_{[app]\mathbf{S}}^k(\mathbf{A}^{(m)})$ at $\mathbf{A}^{(m)^{k-1}}$ and γ is the step size which is decided by a line search. By applying proximal operator on each row of $\mathbf{A}$ (i.e., $\mathbf{a}_t$) in (30)

Algorithm 3 $\text{MKC}_{app}\left(\mathbf{K}^{(m)}, I^{(m)}, \forall m \in [1, \ldots, M]\right)$

Initialization:

$s_{mm}^0 = 0$, $s_{ml}^0 = \frac{1}{M-1}$,

$\mathbf{A}_{I^{(m)}I^{(m)}}^{(m)^0}$ = Identity matrix and

$a_{tt'}^{(m)^0} \sim uniform(-1,1)$ for all $t, t' \notin I^{(m)}$

repeat
   for m=1 to M **do**
      for t=1 to N **do**
$$\mathbf{a}_t^{(m)^k} = max\left(0, (1 - \frac{\gamma\lambda_2}{\|\Delta\mathbf{a}_t^{(m)^{k-1}}\|_2})\Delta\mathbf{a}_t^{(m)^{k-1}}\right)$$
                    [according to s (30, 31)]
                    [λ is fixed by line search]
      end for
   end for
   for m=1 to M **do**
      $\mathbf{s}_m^k = \arg\min_{\mathbf{s}_m} Sobj_{[app]\mathbf{A}}^k(\mathbf{s}_m)$
                    [according to (32)]
   end for
until convergence

$$\mathbf{a}_t^{(m)^k} = max\left(0, (1 - \frac{\gamma\lambda_2}{\|\Delta\mathbf{a}_t^{(m)^{k-1}}\|_2})\Delta\mathbf{a}_t^{(m)^{k-1}}\right), \tag{31}$$

where $\Delta\mathbf{a}_t^{(m)^{k-1}}$ is the tth row of $\left(\mathbf{A}^{(m)^{k-1}} - \gamma\partial Aobj_{[app]\mathbf{S}}^k(\mathbf{A}^{(m)})\right)$.

Again, in the kth iteration, for fixed $\mathbf{A}^{(m)^k}, \forall m = [1, \ldots, M]$, $\mathbf{S}$ is updated by independently updating each row ($\mathbf{s}_m$) through solving the following Quadratic Program:

$$\mathbf{s}_m^k = \arg\min_{\mathbf{s}_m} Sobj_{[app]\mathbf{A}}^k(\mathbf{s}_m)$$
$$s.t \quad \sum_{l \neq m} s_{ml} = 1,$$
$$s_{ml} \geq 0 \; \forall l \tag{32}$$

where $Sobj_{[app]\mathbf{A}}^k(\mathbf{s}_m) = \|\mathbf{A}_{I^{(m)}}^{(m)^T}\mathbf{K}_{I^{(m)}I^{(m)}}^{(m)}\mathbf{A}_{I^{(m)}}^{(m)} - \sum_{l=1, l \neq m}^M s_{ml}\mathbf{A}_{I^{(l)}}^{(l)^T}\mathbf{K}_{I^{(l)}I^{(l)}}^{(l)}\mathbf{A}_{I^{(l)}}^{(l)}\|_2^2$.

6 Experiments

We apply the proposed **MKC** method on a variety of data sets, with different types of kernel functions in different views, along with different amounts of missing data points. The objectives of our experiments are: (1) to compare the performance of **MKC** against other existing methods in terms of the ability to predict the missing kernel rows, (2) to empirically show that the proposed kernel approximation with the help of the reconstruction weights also improves running-time over the MKC_{sdp} method.

6.1 Experimental setup

6.1.1 Data sets:

To evaluate the performance of our method, we used 4 simulated data sets with 100 data points and 5 views, as well as two real-world multi-view data sets: (1) Dream Challenge 7 data set (DREAM) (Daemen et al, 2013; Heiser et al, 2012) and (2) Reuters RCV1/RCV2 multilingual data (Amini et al, 2009).

   Synthetic data sets: We followed the following steps to simulate our synthetic data sets:

1   We generated the first 10 points $(\mathbf{X}_{B^{(m)}}^{(m)})$ for each view, where $\mathbf{X}_{B^{(m)}}^{(1)}$ and $\mathbf{X}_{B^{(m)}}^{(2)}$ are uniformly distributed in $[-1,1]^5$ and $\mathbf{X}_{B^{(m)}}^{(3)}$, $\mathbf{X}_{B^{(m)}}^{(4)}$ and, $\mathbf{X}_{B^{(m)}}^{(5)}$ are uniformly distributed in $[-1,1]^{10}$.

2   These 10 data points were used as basis sets for each view, and further 90 data points in each view were generated by $\mathbf{X}^{(m)} = \mathbf{A}^{(m)}\mathbf{X}_{B^{(m)}}^{(m)}$, where the $\mathbf{A}^{(m)}$ are uniformly distributed random matrices $\in \mathbb{R}^{90 \times 10}$. We chose $\mathbf{A}^{(1)} = \mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)} = \mathbf{A}^{(4)} = \mathbf{A}^{(5)}$.

3   Finally, $\mathbf{K}^{(m)}$ was generated from $\mathbf{X}^{(m)}$ by using different kernel functions for different data sets as follows:
    – TOYL : Linear kernel for all views
    – TOYG1 and TOYG0.1 : Gaussian kernel for all views where the kernel with of the Gaussian kernel are 1 and 0.1 respectively.
    – TOYLG1 : Linear kernel for the first 3 views and Gaussian kernel for the last two views with the kernel width 1. Note that with this selection view 3 shares reconstruction weights with view 4 and 5, but has the same kernel as views 1 and 2.

    Figure 2 shows the the eigen-spectra of kernel matrices are very much different for TOYLG1 where we have used both Linear and Gaussing kernels in views.

   The Dream Challenge 7 data set (DREAM): For Dream Challenge 7, genomic characterizations of multiple types on 53 breast cancer cell lines are provided. They consist of DNA copy number variation, transcript expression values, whole exome sequencing, RNA sequencing data, DNA methylation data and RPPA protein quantification measurements. In addition, some of the views are missing for some cell lines. For 25 data points all 6 views are available. For all the 6 views, we calculated Gaussian kernels after normalizing the data sets. We generated two other kernels by using Jaccard's kernel function over binarized exome data and RNA sequencing data. Hence, the final data set has 8 kernel matrices. Figure 2 shows the eigen-specta of the kernel matrices of all views, which is quite different for different views.

   RCV1/RCV2: Reuters RCV1/RCV2 multilingual data set contains aligned documents for 5 languages (English, French, Germany, Italian and Spanish). Originally the documents are in any one of these languages and then corresponding documents for other views have been generated by machine translations of the original document. For our experiment, we randomly selected 1500 documents which were originally in English. The latent semantic kernel (Cristianini et al, 2002) is used for all languages.
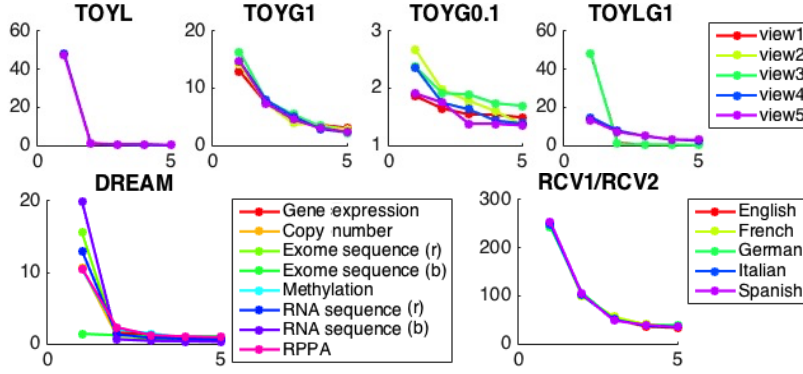
Fig. 2 Eigen-spectra of kernel matrices of the different views in the data sets. It shows that eigen-spectra of TOYLG1, the data with two different kernels and the real world DREAM data-sets are very much different in different views. Each coloured line in a plot shows the eigen-spectrum in one view. Here, (r) indicates use of Gaussian kernel on real values whereas (b) indicates use of Jaccard's kernel on binarized values.

6.1.2 Evaluation setup

Each of the data sets was partitioned into tuning and test sets. The missing views were introduced in these partitions independently. To induce missing views, we randomly selected data points from each partition, a few views for each of them, and deleted the corresponding rows and columns from the kernel matrices. The tuning set was used for parameter tuning. All the results have been reported on the test set which was independent of the tuning set.

For all 4 synthetic data sets as well as RCV1/RCV2 we chose 40% of the data samples as the tuning set, and the rest 60% were used for testing. For the DREAM data set these partitions were 60% for tuning and 40% for testing.

We generated versions of the data with different amounts of missing values. For the first test case, we deleted 1 view from each selected data point in each data set. In the second test case, we removed 2 views for TOY and RCV1/RCV2 data sets and 3 views for DREAM. For the third one we deleted 3 views among 5 views per selected data point in TOY and RCV1/RCV2, and 5 views among 8 views per selected data point in DREAM.

We repeated all our experiments for 5 random tuning and test partitions with different missing entries and report the average performance on them.

6.1.3 Compared methods

We compared performance of the proposed methods, $\mathbf{MKC}_{embd(hm)}$ $\mathbf{MKC}_{embd(ht)}$, $\mathbf{MKC}_{app}$, $\mathbf{MKC}_{sdp}$, with *k nearest neighbour* (KNN) imputation as a baseline KNN has previously been shown to be a competitive imputation method (Brock et al, 2008). For KNN imputation we first concatenated underlying feature representations from all views to get a joint feature representation. We then sought *k* nearest data points by using their available parts, and the missing part was imputed as either average (**Knn**) or the weighted average (**wKnn**) of the selected neighbours. We also compare our

result with generative model based approach of Lian et al (2015) (**MLFS**) and with an Em-based kernel completion method (**EM**$_{based}$) proposed by Tsuda et al (2003). We thank authors for sharing their software for **MLFS**. Tsuda et al (2003) cannot solve our problem when no view is complete, hence we study the relative performance only in the cases which it can solve. For Tsuda et al (2003)'s method we assume the first view is complete.

We also compared **MKC**$_{embd(ht)}$, with **MKC**$_{rnd}$ where we assumed the basis vectors are selected randomly with uniform distribution with out replacement and after that reconstruction weights for all views are optimizied.

The hyper-parameters λ_1 and λ_2 of **MKC** and k of **Knn** and **wKnn** were selected with the help of tuning set, from the range of 10^{-3} to 10^3 and $[1, 2, 3, 5, 7, 10]$ respectively. All reported results indicate performance in the test sets..

6.2 Prediction error comparisons

6.2.1 Average Relative Error (ARE)

We evaluated the performance of all methods using the *average relative error* (ARE) (Xu et al, 2013). Let $\hat{\mathbf{k}}_t^{(m)}$ be the predicted t^{th} row for the mth view and the corresponding true values of kernel row be $\mathbf{k}_t^{(m)}$, then the *relative error* is the relative root mean square deviation. The average relative error (in percentage) is then computed over all missing data points for a view, that is,

$$\text{ARE} = \frac{100}{n_t^{(m)}} \left(\sum_{t \notin I^{(m)}} \frac{\|\hat{\mathbf{k}}_t^{(m)} - \mathbf{k}_t^{(m)}\|_2}{\|\mathbf{k}_t^{(m)}\|_2} \right). \tag{33}$$

Here $n_t^{(m)}$ is the number of missing samples in the mth view.

6.2.2 Results

Table 1 shows the Average Relative Error (33) for the compared methods. It shows that the proposed **MKC** methods generally predict missing values more accurately than **Knn**, **wKnn**, **EM**$_{based}$ and **MLFS**. In particular, the differences in favor to the **MKC** methods increase when the number of missing views is increased.

The most recent method **MLFS** performs comparatively for the DREAM data-set and the data-set with linear kernels (TOYL). But it deteriorates very badly with the increased of non-linearity of kernels, i.e., TOYG1 and TOYLG1. For highly non-linear sparse kernels (TOYG0.1) and for RCV1/RCV2 data-set with large amount of missing views the **MLFS** fails to predict. The **EM**$_{based}$ sometimes has more than 200% error and higher (more than 200%) variance. The most accurate method in each setup is one of the proposed **MKC**'s. **MKC**$_{embd(hm)}$ is generally the least accurate of them, but still competitive against the other compared methods. We further note that:

Number of missing views = 1 (TOY and RCV1/RCV2) and 1 (DREAM)						
Algorithm	TOYL	TOYG1	TOYG0.1	TOYLG1	DREAM	RCV1/RCV2
$MKC_{embd(ht)}$	**0.07 (± 0.09)**	7.40 (± 9.20)	84.91 (± 5.18)	4.50 (± 6.72)	13.36 (± 26.53)	1.79 (± 0.89)
MKC_{app}	0.09 (± 0.10)	**5.02 (± 3.60)**	76.24 (± 10.59)	**2.11 (± 3.40)**	14.46 (± 28.39)	**1.15 (± 0.48)**
MKC_{sdp}	0.22 (± 0.32)	11.29 (± 6.29)	**7.83 (± 5.46)**	6.06 (± 7.84)	20.19 (± 41.28)	-
$MKC_{embd(hm)}$	0.19 (± 0.18)	27.54 (± 14.38)	86.08 (± 6.34)	8.93 (± 11.86)	16.12 (± 30.27)	3.27 (± 1.26)
MKC_{rnd}	0.08 (± 0.09)	16.79 (± 13.14)	93.36 (± 5.18)	4.50 (± 6.72)	16.00 (± 28.83)	1.40 (± 0.28)
MLFS	0.28 (± 0.20)	79.20 (± 22.95)	100.00 (± 0.00)	25.34 (± 31.42)	**11.69 (± 26.47)**	15.13 (± 3.25)
EM_{based}	20.65 (± 41.08)	554.08 (± 90.00)	31.23 (± 37.02)	759.74 (± 90.00)	14.78 (± 32.93)	23.38 (± 29.00)
Knn	0.34 (± 0.53)	42.89 (± 27.93)	62.69 (± 8.77)	11.27 (± 15.53)	14.94 (± 25.29)	5.79 (± 2.65)
wKnn	0.34 (± 0.53)	45.47 (± 29.50)	62.80 (± 8.86)	15.30 (± 20.15)	15.00 (± 25.35)	5.91 (± 2.71)
Number of missing views = 2 (TOY and RCV1/RCV2) and 3 (DREAM)						
Algorithm	TOYL	TOYG1	TOYG0.1	TOYLG1	DREAM	RCV1/RCV2
$MKC_{embd(ht)}$	0.08 (± 0.07)	9.43 (± 6.72)	86.72 (± 3.34)	**3.26 (± 5.07)**	16.13 (± 28.29)	2.74 (± 0.85)
MKC_{app}	**0.07 (± 0.05)**	**6.89 (± 3.44)**	84.40 (± 9.04)	4.01 (± 6.03)	17.51 (± 27.65)	**1.61 (± 0.65)**
MKC_{sdp}	0.34 (± 0.39)	19.87 (± 13.88)	**18.30 (± 12.94)**	37.88 (± 49.58)	32.86 (± 51.73)	-
$MKC_{embd(hm)}$	0.14 (± 0.09)	29.69 (± 9.85)	96.19 (± 1.60)	13.78 (± 21.33)	18.33 (± 29.56)	2.71 (± 0.71)
MKC_{rnd}	0.08 (± 0.07)	17.94 (± 6.91)	93.24 (± 3.34)	6.96 (± 9.39)	17.70 (± 28.29)	2.29 (± 0.11)
MLFS	0.58 (± 0.26)	79.54 (± 11.88)	100.00 (± 0.00)	30.86 (± 38.79)	16.59 (± 31.28)	25.23 (± 36.29)
EM_{based}	28.66 (± 42.28)	202.58 (± 339.02)	61.87 (± 50.48)	298.57 (± 281.79)	25.98 (± 63.70)	27.83 (± 13.70)
Knn	0.26 (± 0.26)	52.18 (± 16.34)	97.65 (± 11.62)	19.64 (± 25.62)	22.04 (± 30.07)	7.47 (± 2.34)
wKnn	0.26 (± 0.26)	54.94 (± 16.02)	98.24 (± 11.17)	20.90 (± 27.16)	22.20 (± 30.26)	7.61 (± 2.40)
Number of missing views = 3 (TOY and RCV1/RCV2) and 5 (DREAM)						
Algorithm	TOYL	TOYG1	TOYG0.1	TOYLG1	DREAM	RCV1/RCV2
$MKC_{embd(ht)}$	**0.05 (± 0.04)**	12.87 (± 3.40)	89.88 (± 3.26)	**5.13 (± 7.17)**	**20.04 (± 30.58)**	1.69 (± 0.85)
MKC_{app}	0.10 (± 0.05)	**12.04 (± 3.71)**	89.69 (± 5.54)	5.72 (± 7.88)	20.43 (± 30.39)	2.91 (± 3.15)
MKC_{sdp}	0.41 (± 0.35)	86.21 (± 55.84)	**17.59 (± 9.37)**	438.92 (± 624.21)	97.79 (± 89.51)	-
$MKC_{embd(hm)}$	0.16 (± 0.10)	32.70 (± 10.63)	95.43 (± 1.75)	15.91 (± 23.31)	22.13 (± 33.29)	2.45 (± 1.54)
MKC_{rnd}	0.08 (± 0.06)	20.56 (± 4.88)	94.54 (± 3.26)	7.95 (± 10.76)	21.04 (± 30.58)	**1.38 (± 0.29)**
MLFS	8.55 (± 3.08)	99.92 (± 0.23)	100.00 (± 0.00)	32.17 (± 32.28)	21.54 (± 30.94)	100.00 (± 0.00)
EM_{based}	21.46 (± 40.73)	101.87 (± 63.31)	554.08 (± 90.00)	231.98 (± 416.30)	60.47 (± 245.88)	29.76 (± 12.28)
Knn	0.39 (± 0.33)	62.32 (± 14.94)	112.79 (± 13.22)	24.93 (± 33.57)	27.54 (± 36.88)	8.66 (± 1.99)
wKnn	0.38 (± 0.33)	66.94 (± 14.74)	97.96 (± 4.52)	27.48 (± 36.90)	27.57 (± 36.70)	8.85 (± 1.99)

Table 1 Average relative error percentage (33). The smallest ARE for each setup are boldfaced. The figures are ARE averaged over all views and 5 random validation and test partitions with different missing entries (standard deviation in parentheses).

– $MKC_{embd(ht)}$ is consistently the best when different views have different kernel functions or eigen-spectra, e.g., TOYLG1 and DREAM (Figure 2). Better performance of $MKC_{embd(ht)}$ than $MKC_{embd(hm)}$ in DREAM data gives evidence of applicability of $MKC_{embd(ht)}$ in real-world data set.

– MKC_{app} performs best or very close to $MKC_{embd(ht)}$ when kernel functions and eigen-spectra of all views are the same (for instance TOYL, TOYG1 and RCV1/RCV2). As MKC_{app} learns between-view relationships on kernel values it is not able to perform well for TOYLG1 and DREAM where very different kernel functions are used in different views.

– MKC_{sdp} outperforms all other methods when kernel functions are highly non-linear (such as in TOYG0.1). On less non-linear cases, MKC_{sdp} on the other hand trails in accuracy to the other MKC variants. MKC_{sdp} is computationally more demanding than the others, to the extent that on RCV1/RCV2 data we had to skip it.

Figure 3 depicts the performance as the number of missing samples per view is increased. Here, $MKC_{embd(ht)}$, MKC_{app} and $MKC_{embd(hm)}$ prove to be the most robust methods over all data sets. The performance of MKC_{sdp} seems to be the most sensitive to amount of missing samples. Overall, EM_{based}, Knn, and wKnn have worse error rates than the MKC methods.
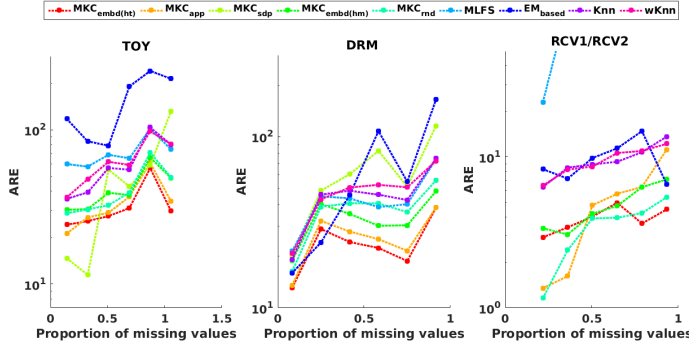
Fig. 3 ARE (33) for different proportions of missing samples. Values are averages over views and random validation and test partitions. The value for TOY are additionally averaged over all 4 TOY data sets.
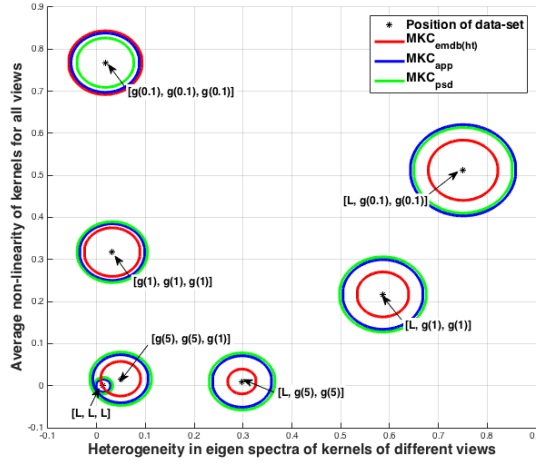

Fig. 4 Figure shows how the prediction error (bigger size of circle indicates the more ARE) $\mathbf{MKC}_{embd(ht)}$(red), $\mathbf{MKC}_{app}$(blue) and $\mathbf{MKC}_{sdp}$(green) changes with increase of the non-linearity and the heterogeneity of kernel functions used in different views. It show $\mathbf{EM}_{based}$(red) performed best for almost all cases while for highly non-linear kernel $\mathbf{MKC}_{sdp}$(green) performed better. Text at center of each circle indicates kernel functions used in 3 views for that data-set: "L"=linear and "$g(w)$"= Gaussian with width w.

6.3 Comparison of performance of different versions of the proposed approach

Figure 4 shows how relative prediction error (ARE) of $\mathbf{MKC}_{embd(ht)}$, $\mathbf{MKC}_{app}$ and $\mathbf{MKC}_{sdp}$ vary with two properties of given data-sets. Namely, (1) difference among eigen-spectra of kernel of different views(x axis) and (2) non-linearity of kernels for all views(y axis). For this experiment, we consider 3rd,4th and 5th views of TOY data. Here all views have been generated from same embedding. Non-linearity of kernel

function varies with combination of linear and Gaussian kernel where the kernel with of the Gaussian kernel varies among 5, 1 and 0.1.

The heterogeneity of eigen-spectra of all kernels are calculated as average mean square difference of kernel spectra for each pair of kernels. The non-linearity of kernel is indicated by the average of the 20th eigen-values of all views. Each circle indicates amount of performance error in prediction by $\mathbf{MKC}_{embd(ht)}$, $\mathbf{MKC}_{app}$ and $\mathbf{MKC}_{sdp}$ where radius of each circle is proportional to "$log(ARE) + Thr$". A constant "Thr" was required to have positive radii for all circles for better visualization. We further note that

- The performance of $\mathbf{MKC}_{embd(ht)}$ is the best among these three methods for all most all cases.
- Only when all views have similarly high non-linear kernel (top-left corner), $\mathbf{MKC}_{sdp}$ performs best among all. It also shows that the performance of $\mathbf{MKC}_{sdp}$ improves with increase of non-linearity.
- We can also see that with the increase of heterogeneity in kernels (increase of x-axis) the performance of $\mathbf{MKC}_{app}$ deteriorates and is getting worse than that of $\mathbf{MKC}_{embd(ht)}$.

6.4 Running time comparison

Number of missing views = 1 (TOY and RCV1/RCV2) and 1 (DREAM)			
Algorithm	TOY(mins)	DREAM(mins)	RCV1/RCV2(hrs)
$\mathbf{MKC}_{embd(ht)}$	5.00($\pm$ 2.04)	0.86($\pm$ 0.29)	45.93($\pm$ 2.27)
$\mathbf{MKC}_{app}$	2.91($\pm$ 0.39)	1.89($\pm$ 0.62)	16.59($\pm$ 0.28)
$\mathbf{MKC}_{sdp}$	14.82($\pm$ 4.39)	1.13($\pm$ 0.11)	-
$\mathbf{MKC}_{embd(hm)}$	0.15($\pm$ 0.07)	0.05($\pm$ 0.03)	0.28($\pm$ 0.02)
$\mathbf{MKC}_{rnd}$	2.71($\pm$ 0.53)	0.85($\pm$ 0.18)	4.21($\pm$ 0.56)
$\mathbf{MLFS}$	0.15($\pm$ 0.01)	0.01($\pm$ 0.01)	1.72($\pm$ 0.11)
$\mathbf{EM}_{based}$	0.50($\pm$ 0.19)	0.03($\pm$ 0.05)	0.03($\pm$ 0.00)
Number of missing view = 2 (TOY and RCV1/RCV2) and 3 (DREAM)			
Algorithm	TOY(mins)	DREAM(mins)	RCV1/RCV2(hrs)
$\mathbf{MKC}_{embd(ht)}$	7.58($\pm$ 2.18)	1.13($\pm$ 0.12)	25.86($\pm$ 0.36)
$\mathbf{MKC}_{app}$	2.78($\pm$ 0.68)	1.29($\pm$ 0.25)	34.42($\pm$ 1.28)
$\mathbf{MKC}_{sdp}$	25.65($\pm$ 5.43)	1.97($\pm$ 0.34)	-
$\mathbf{MKC}_{embd(hm)}$	0.11($\pm$ 0.05)	0.03($\pm$ 0.01)	0.47($\pm$ 0.02)
$\mathbf{MKC}_{rnd}$	1.33($\pm$ 0.59)	1.34($\pm$ 0.19)	3.63($\pm$ 0.50)
$\mathbf{MLFS}$	0.14($\pm$ 0.01)	0.01($\pm$ 0.02)	2.61($\pm$ 1.01)
$\mathbf{EM}_{based}$	0.45($\pm$ 0.08)	0.06($\pm$ 0.06)	0.03($\pm$ 0.00)
Number of missing views = 3 (TOY and RCV1/RCV2) and 5 (DREAM)			
Algorithm	TOY(mins)	DREAM(mins)	RCV1/RCV2(hrs)
$\mathbf{MKC}_{embd(ht)}$	6.83($\pm$ 2.14)	3.39($\pm$ 1.11)	24.39($\pm$ 2.13)
$\mathbf{MKC}_{app}$	2.20($\pm$ 0.66)	3.64($\pm$ 1.79)	20.26($\pm$ 1.72)
$\mathbf{MKC}_{sdp}$	178.1($\pm$ 162.9)	4.94($\pm$ 2.48)	-
$\mathbf{MKC}_{embd(hm)}$	0.12($\pm$ 0.08)	0.03($\pm$ 0.02)	0.57($\pm$ 0.00)
$\mathbf{MKC}_{rnd}$	1.83($\pm$ 0.73)	1.31($\pm$ 0.20)	2.81($\pm$ 0.17)
$\mathbf{MLFS}$	0.07($\pm$ 0.05)	0.05($\pm$ 0.01)	1.04($\pm$ 0.40)
$\mathbf{EM}_{based}$	0.45($\pm$ 0.05)	0.10($\pm$ 0.05)	0.03($\pm$ 0.00)

Table 2 Average running time over all views and for TOY, over all 4 data sets. The running times for **Knn** and **wKnn** are around 10^{-3} minutes for all data sets.

Table 2 depicts the running times for the compared methods. $\mathbf{MKC}_{app}$, $\mathbf{MKC}_{embd(ht)}$ and $\mathbf{MKC}_{embd(hm)}$ are many times faster than $\mathbf{MKC}_{sdp}$. In particular, $\mathbf{MKC}_{embd(hm)}$ is competitive in running time with the significantly less accurate $\mathbf{EM}_{based}$ and $\mathbf{MLFS}$ methods, except on the RCV1/RCV2 data. As expected, $\mathbf{Knn}$ and $\mathbf{wKnn}$ are orders of magnitude faster but fall far from the reconstruction quality of the $\mathbf{MKC}$ methods.

7 Conclusions

In this paper, we have introduced new methods for kernel completion in the multi-view setting. The methods are able to propagate relevant information across views to predict missing rows/columns of kernel matrices in multi-view data. In particular, we are able to predict missing rows/columns of kernel matrices for non-linear kernels, and do not need any complete kernel matrices *a priori*.

Our method of within-view learning approximates the full kernel by a sparse basis set of examples with local reconstruction weights, picked up by $\ell_{2,1}$ regularization. This approach has the added benefit of circumventing the need of an explicit PSD constraint in optimization. We showed that the method generalizes and improves Nyström approximation. For learning between views, we proposed two alternative approaches, one based on learning convex kernel combinations and another based on learning a convex set of reconstruction weights. The heterogeneity of the kernels in different views affects which of the approaches is favourable. We related theoretically the kernel approximation quality of these methods to the similarity of eigen-spaces of the individual kernels.

Our experiments show that the proposed multi-view completion methods are in general more accurate than previously available methods. In terms of running time, due to the inherent non-convexity of the optimization problems, the new proposals have still room to improve. However, the methods are amenable for efficient parallelization, which we leave for further work.

References

Amini M, Usunier N, Goutte C (2009) Learning from multiple partially observed views - an application to multilingual text categorization. In: Advances in Neural Information Processing Systems 22, pp 28–36

Argyriou A, Micchelli CA, Pontil M (2005) Learning convex combinations of continuously parameterized basic kernels. In: Proceedings of the 18th Annual Conference on Learning Theory, pp 338–352

Argyriou A, Evgeniou T, Pontil M (2006) Multi-task feature learning. In: Advances in Neural Information Processing Systems, pp 41–48

Bach F, Lanckriet G, Jordan M (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st International Conference on Machine Learning, ACM, pp 6–13

Bach F, Jenatton R, Mairal J, Obozinski G (2011) Convex optimization with sparsity-inducing norms. Optimization for Machine Learning pp 19–53

Brock G, Shaffer J, Blakesley R, Lotz M, Tseng G (2008) Which missing value imputation method to use in expression profiles: a comparative study and two selection schemes. BMC Bioinformatics 9:1–12

Cortes C, Mohri M, Rostamizadeh A (2012) Algorithms for learning kernels based on centered alignment. Journal of Machine Learning Research 13:795–828

Cristianini N, Shawe-Taylor J, Lodhi H (2002) Latent semantic kernels. Journal of Intelligent Information Systems 18(2-3):127–152

Daemen A, Griffith O, Heiser L, et al (2013) Modeling precision treatment of breast cancer. Genome Biology 14(10)

Gönen M, Alpaydin E (2011) Multiple kernel learning algorithms. Journal of Machine Learning Research 12:2211–2268

Graepel T (2002) Kernel matrix completion by semidefinite programming. In: Proceedings of the 12th International Conference on Artificial Neural Networks, Springer, pp 694–699

Heiser LM, Sadanandam A, et al (2012) Subtype and pathway specific responses to anticancer compounds in breast cancer. Proceedings of the National Academy of Sciences 109(8):2724–2729

Kumar S, Mohri M, Talwalkar A (2009) On sampling-based approximate spectral decomposition. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, pp 553–560

Lian W, Rai P, Salazar E, Carin L (2015) Integrating features and similarities: Flexible models for heterogeneous multiview data. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, pp 2757–2763

Paisley J, Carin L (2010) A nonparametric Bayesian model for kernel matrix completion. In: The 35th International Conference on Acoustics, Speech, and Signal Processing, IEEE, pp 2090–2093

Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326

Shao W, Shi X, Yu PS (2013) Clustering on multiple incomplete datasets via collective kernel learning. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, pp 1181–1186

Trivedi A, Rai P, Daumé III H, DuVall SL (2005) Multiview clustering with incomplete views. In: Proceedings of the NIPS Workshop

Tsuda K, Akaho S, Asai K (2003) The em algorithm for kernel matrix completion with auxiliary data. The Journal of Machine Learning Research 4:67–81

Williams C, Seeger M (2001) Using the nyström method to speed up kernel machines. In: Proceedings of the 14th Annual Conference on Neural Information Processing Systems, EPFL-CONF-161322, pp 682–688

Williams D, Carin L (2005) Analytical kernel matrix completion with incomplete multi-view data. In: Proceedings of the ICML Workshop on Learning With Multiple Views

Xu M, Jin R, Zhou ZH (2013) Speedup matrix completion with side information: Application to multi-label learning. In: Advances in Neural Information Processing Systems, pp 2301–2309