Mach Learn (2010) 79: 47–71
DOI 10.1007/s10994-009-5157-z

Multi-view kernel construction

**Virginia R. de Sa · Patrick W. Gallagher ·
Joshua M. Lewis · Vicente L. Malave**

Received: 28 February 2009 / Revised: 25 September 2009 / Accepted: 8 October 2009 /
Published online: 13 November 2009

Abstract In many problem domains data may come from multiple sources (or views), such as video and audio from a camera or text on and links to a web page. These multiple views of the data are often not directly comparable to one another, and thus a principled method for their integration is warranted. In this paper we develop a new algorithm to leverage information from multiple views for unsupervised clustering by constructing a custom kernel. We generate a multipartite graph (with the number of parts given by the number of views) that induces a kernel we then use for spectral clustering. Our algorithm can be seen as a generalization of co-clustering and spectral clustering and a relative of Kernel Canonical Correlation Analysis. We demonstrate the algorithm on four data sets: an illustrative artificial data set, synthetic fMRI data, voxels from an fMRI study, and a collection of web pages. Finally, we compare its performance to common alternatives.

Keywords Spectral clustering · Minimizing-disagreement · Multi-view · fMRI analysis · Kernel · Canonical correlation analysis · CCA · Co-clustering

1 Introduction

How should different sources of information be combined for unsupervised learning? Consider the case of experimental data from an fMRI experiment. The data include a blood-oxygen-level dependent (BOLD) response time course for each voxel and each pattern presentation as well as the spatial position of each voxel. What is the best way to combine these two sources of data to cluster voxels that are functionally related?

One common method for dealing with multiple sources of data is to ignore the distinctions and concatenate all the sources into one vector (e.g. Loeff et al. 2006; Cai et al. 2004).

Editors: Nicolo Cesa-Bianchi, David R. Hardoon, and Gayle Leen.

This work is supported by NSF CAREER grant IIS-0133996 and NSF IGERT grant DGE-0333451 and NSF CBET-0756828.

V.R. de Sa (✉)
Department of Cognitive Science, University of California, San Diego, CA 92093-0515, USA
e-mail: desa@ucsd.edu

Yet concatenating the sources represents an implicit assumption that all dimensions from all sources are directly comparable. This is certainly not the case in our example—the BOLD signal time course is clearly noncomparable to the spatial position of the voxels. Normalizing each dimension, e.g. by z-scoring, is also not appropriate in this case. After normalization previously comparable dimensions such as the x, y, and z spatial dimensions are now no longer so. Although one would not want to change the scaling on x versus y dependent on how big a spread in x data one has, this is precisely what normalization would do.

An alternate method is to deal with the different sources separately and then combine the results. One way to do this is to use each source to compute a measure of similarity, called a kernel, and then combine these. A potential combination approach is to simply add the kernel matrices with some weighting. This has been done for the unsupervised case (Zhou and Burges 2007), for the semi-supervised case (Joachims 2003), and for the supervised case, where the algorithm learns a convex combination of kernel matrices to satisfy an objective function (Lanckriet et al. 2004; Rakotomamonjy et al. 2008). For these approaches, the problem thus shifts to how one should weight the matrices from the different sources. Other methods of view combination are possible (Long et al. 2006, 2008).

We suggest an alternate approach based on motivation from the multi-view field (de Sa 1994; de Sa and Ballard 1998; Blum and Mitchell 1998) partially developed in a workshop paper (de Sa 2005).[1] The general idea is to construct a graph where each node represents one view of each data pattern (e.g. one frame of visual input from the camera) and we connect nodes of co-occuring patterns (e.g. video frame with temporally coincident auditory input). Given the incompatibility between views discussed above we should not create edges between two nodes within the same view, since the relative weighting of those within-view edges versus between-view edges would implicitly introduce unmotivated assumptions about the relationship between the two views. Instead we leverage within-view neighborhood information by smoothly reweighting between-view edges to reflect within-view similarity. In this way we are able to build a graph where all edges connect nodes from different views (maintaining their conceptual distinction) but those edges are weighted by within-view neighborhood relationships (maintaining each view's information). Our algorithm can be seen as a generalization of co-clustering (Dhillon 2001), a generalization of spectral clustering, and related to Kernel Canonical Correlation Analysis (Lai and Fyfe 2000; Hardoon et al. 2003, 2004; Blaschko and Lampert 2008).

2 Algorithm development

Our kernel combination algorithm has its origins in the field of graph cuts. It is based on ideas originally developed for the Minimizing-Disagreement (M–D) algorithm (de Sa 1994; de Sa and Ballard 1998). The idea behind the M–D algorithm is that when two (or more) networks receive data from different views with no explicit supervisory labels, a graph constructed from the data should be cut such that co-occurring patterns are placed in the same partition. Consider a data set with two classes of objects each with two one-dimensional views. Figure 1 shows a graph of this data set with edges between co-occuring samples. The points are ordered within each view by their value and colored according to class.

The M–D algorithm seeks a cut from top to bottom that crosses the fewest lines within the pattern space (subject to a balance constraint to prevent trivial solutions with empty or

[1] Sections 2, 3 and 4 have been reformulated and generalized. Section 7 appears mostly as it did before. Sections 1, 5, 6 and 8 are brand new as well as two of the Appendices.

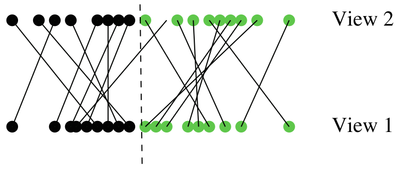


Fig. 1 A graph of co-occuring datapoints in two one-dimensional views. We draw edges between co-occuring points and points are organized based on within-view proximity. The M–D algorithm attempts to find a non-trivial cut for the graph that minimizes the number of edges cut

near empty clusters). In the present case, disagreement is minimized for the dashed line shown. In this paper we transform this intuitive idea for one-dimensional views to a general algorithm on a weighted multipartite graph.

The difficulty in transforming this intuitive idea into a general M–D graph cut algorithm is that in describing it as making a cut from top to bottom, we implicitly use a neighborhood relationship within each top set and bottom set, though this relationship is not explicitly represented. That is, we assume that points drawn in a line next to each other are similar points in the same view. Constructing a bipartite graph with the points as nodes and then applying a graph cut algorithm neglects these same-view neighborhood relationships.

One solution would be to simply connect co-occurring values **and** also join nearest neighbors (or join neighbors according to a similarity measure) in each view. However, this raises the issue we considered above: how to encode the relative strengths of the between-view pairing weights as compared to the within-view affinity weights since these quantities are in general non-comparable.

We approach this issue of comparison as follows: use the within-view neighborhood relationships to reweight the between-view edges. This implicitly encodes the neighborhood relationships in each view in the connections between nodes in different views. In particular, we begin by drawing reduced weight co-occurrence relationships between neighbors of an observed pair of patterns (weighted by a radial basis function such as a Gaussian kernel). Each input in each view is represented by a node in the graph. The strength of the weight between two nodes in different views depends on the number of multi-view patterns (which we can think of as co-occuring pairs of patterns) that are sufficiently close to both nodes (in both views). This representation has the semantics that we would like to smooth the noise in the actual patterns. This spectral Minimizing-Disagreement algorithm we refer to as **sMD**.

More specifically, let us define $\mathbf{x}_i^{(v)}$ as view v of the ith pattern. For each view of each pattern we will first construct a graph node and then define $n_i^{(v)}$ to represent the node for view v of the ith pattern.

Now consider two patterns from the same view, $\mathbf{x}_1^{(1)} = [1 \;\; 2 \;\; 1]^{\mathrm{T}}$ and $\mathbf{x}_2^{(1)} = [1 \;\; 2 \;\; 1]^{\mathrm{T}} + \boldsymbol{\epsilon}$. For small $\boldsymbol{\epsilon}$ it is sensible to consider these two patterns identical. This observation implies that $\mathbf{x}_1^{(2)}$, the co-occurring pattern for $\mathbf{x}_1^{(1)}$ from view 2 may reasonably be considered paired with $\mathbf{x}_2^{(1)}$. As we consider larger values of $\boldsymbol{\epsilon}$, the assumption of equivalence becomes less reasonable, and a Gaussian kernel weighting allows us to smoothly decrease the inferred edge weight of the pairing in question. With this initial principle of similarity in place, we see that to compute the total weight between nodes $n_i^{(1)}$ and $n_j^{(2)}$, we sum over all observed pattern co-occurrences the product of the Gaussian kernel weighted distance between $\mathbf{x}_i^{(1)}$ (the pattern represented by $n_i^{(1)}$) and $\mathbf{x}_k^{(1)}$ (with k ranging over all observed patterns) and the same term for the relationship between the $\mathbf{x}_j^{(2)}$ and $\mathbf{x}_k^{(2)}$. That is

$$\mathbf{W}^{(1\,2)}(i,j) = \sum_{k=1}^{p} \exp\left(-\frac{\|\mathbf{x}_i^{(1)} - \mathbf{x}_k^{(1)}\|^2}{2\sigma_1^2}\right)\exp\left(-\frac{\|\mathbf{x}_j^{(2)} - \mathbf{x}_k^{(2)}\|^2}{2\sigma_2^2}\right) \tag{1}$$

Intuitively, the term within the sum will be closer to one when pattern i is close to pattern k in view one and pattern j is close to the same pattern k in view two. Thus, if $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_j^{(2)}$ share many co-occurring neighbors $\mathbf{W}(i,j)^{(1\,2)}$ will be large. Given within-view affinity matrices defined as $\mathbf{W}^{(v)}(i,j) = \exp\left(\frac{-\|\mathbf{x}_i^{(v)} - \mathbf{x}_j^{(v)}\|^2}{2\sigma_v^2}\right)$ this sum can be more compactly written as

$$\mathbf{W}^{(1\,2)} = \mathbf{W}^{(1)} \times \mathbf{W}^{(2)} \tag{2}$$

To generate our full affinity matrix between all $2p$ nodes we take the $p \times p$ matrix $\mathbf{W}^{(1\,2)}$ and put it in a large $2p \times 2p$ matrix of the form

$$\mathbf{W}_{\mathrm{sMD}} = \begin{bmatrix} \mathbf{0} & \mathbf{W}^{(1\,2)} \\ (\mathbf{W}^{(1\,2)})^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \tag{3}$$

where $\mathbf{0}$ represents a $p \times p$ matrix of zeros. Appendix A gives the objective derivation of $\mathbf{W}_{\mathrm{sMD}}$.

For comparison, we consider two alternative algorithms for generating affinity matrices from multiple sources. The first alternative is to construct multi-view patterns via a simple concatenation of features (the default approach mentioned in the introduction). We call this algorithm **JOINT**, and define its affinity matrix element-wise as

$$\begin{aligned} \mathbf{W}_{\mathrm{JOINT}}(i,j) &= \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{\|\mathbf{x}_i^{(1)} - \mathbf{x}_j^{(1)}\|^2 + \|\mathbf{x}_i^{(2)} - \mathbf{x}_j^{(2)}\|^2}{2\sigma^2}\right) \\ &= \mathbf{W}^{(1)}(i,j)^{\frac{\sigma_1^2}{\sigma^2}} \cdot \mathbf{W}^{(2)}(i,j)^{\frac{\sigma_2^2}{\sigma^2}} \end{aligned}$$

Thus the affinity matrix for clustering in the joint space can be obtained by a per-entry (Hadamard) product of the affinity matrices for the individual modalities, where each affinity matrix might use a distinct σ^2.

$$\mathbf{W}_{\mathrm{JOINT}} = \mathbf{W}^{(1)} \circ \mathbf{W}^{(2)} \tag{4}$$

We also compare our algorithm to one where the affinity matrices of the two individual modalities are added. This corresponds to computing similarities separately for different views and then averaging the results. This idea is mentioned in Joachims (2003) for the semi-supervised case. We call this algorithm **SUM**, and define its affinity matrix as

$$\mathbf{W}_{\mathrm{SUM}} = \mathbf{W}^{(1)} + \alpha \mathbf{W}^{(2)} \tag{5}$$

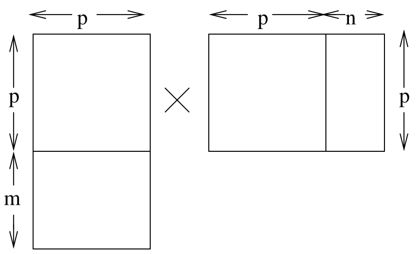


Fig. 2 A graphical view of the matrix multiplication required to compute $\mathbf{W}^{(1\,2)}$ when there are p patterns with both views, m patterns with only view 1 and n patterns with only view 2

2.1 Missing views

A significant benefit of our algorithm is that it can calculate affinities for samples with missing views. Consider a pattern, $\mathbf{x}_i^{(1)}$, that is missing a corresponding $\mathbf{x}_i^{(2)}$. This pattern can still be related to pattern $\mathbf{x}_j^{(2)}$ according to how similar $\mathbf{x}_i^{(1)}$ is to $\mathbf{x}_j^{(1)}$ and its neighbors. Thus we can construct a full bipartite affinity matrix between views using (1) where k sums over only the p paired patterns. This results in a matrix multiplication of the form $\mathbf{W}^{(1)} \times \mathbf{W}^{(2)}$ where this time $\mathbf{W}^{(1)}$ is $(p+m) \times p$ dimensional and $\mathbf{W}^{(2)}$ is $p \times (p+n)$ dimensional, given p co-occurring patterns, m patterns with only view 1 and n patterns with only view 2 (see Fig. 2). Note that the bottom right quadrant of the resulting $\mathbf{W}^{(1\,2)}$ matrix computes the affinity between an unpaired view one pattern and an unpaired view two pattern according to the sum of the affinities between this pair $(\mathbf{x}_{p+i}^{(1)}, \mathbf{x}_{p+j}^{(2)})$ and each of the set of observed pairs $\{(\mathbf{x}_1^{(1)}, \mathbf{x}_1^{(2)}), \ldots, (\mathbf{x}_p^{(1)}, \mathbf{x}_p^{(2)})\}$.

2.2 Extension to several views

Extension to more than two views is straightforward and results in the consideration of a multi-partite graph with each part corresponding to data from one view. Again connections between nodes in a view are not joined directly but only indirectly through nodes in other views. All connections between nodes in view i and nodes in view j are only based on connections derived from similarities to view i and view j patterns in the set of paired patterns. That is the connections between nodes in view i and view j are derived simply from the product of the affinity matrices for view i and view j.

The resulting affinity matrix has a multi-block structure as shown below (for an example with four views).

$$\mathbf{W}_{\text{sMD}} = \begin{bmatrix} \mathbf{0} & W^{(1\,2)} & W^{(1\,3)} & W^{(1\,4)} \\ (W^{(1\,2)})^{\mathrm{T}} & \mathbf{0} & W^{(2\,3)} & W^{(2\,4)} \\ (W^{(1\,3)})^{\mathrm{T}} & (W^{(2\,3)})^{\mathrm{T}} & \mathbf{0} & W^{(3\,4)} \\ (W^{(1\,4)})^{\mathrm{T}} & (W^{(2\,4)})^{\mathrm{T}} & (W^{(3\,4)})^{\mathrm{T}} & \mathbf{0} \end{bmatrix}$$

$\mathbf{W}^{(i\,j)}$ represents the product of the affinity matrix of view i with that of view j. Again different patterns may be missing various views with no trouble. Note in the multiple (more than 2) view case, the patterns must be lined up consistently across the different blocks so it will not necessarily be the case that all the paired patterns (for one block) will be the first rows (in the first matrix) and the first columns (in the second matrix) in the matrix products (as shown in Fig. 2). The example below shows explicitly how missing views are handled in the multiple view setting. Our algorithm greatly increases the number of views that may be used as it is not necessary to have all views present for all data vectors.

Views can be missing from one or more of the patterns. Consider for example the example below consisting of

pattern **a** with	view 1		view 3	view 4
pattern **b** with	view 1			view 4
pattern **c** with		view 2	view 3	
pattern **d** with		view 2	view 3	view 4
pattern **e** with				view 4

The constructed block affinity matrix would be of the form shown above.

Note that as there are no patterns with both a view 1 and view 2 input, there would be no direct connections between the nodes representing the view 1 patterns and those representing the view 2 patterns (they will however be connected through view 3 and view 4 patterns) and the entries ($w(\mathbf{a}^{(1)}, \mathbf{c}^{(2)})$, $w(\mathbf{a}^{(1)}, \mathbf{d}^{(2)})$, $w(\mathbf{b}^{(1)}, \mathbf{c}^{(2)})$, $w(\mathbf{b}^{(1)}, \mathbf{d}^{(2)})$) would all be 0 (the 2nd and 5th submatrices above numbered from the top using row major order). In the above, we use w as a shorthand notation indicating access into the appropriate entry of $\mathbf{W}_{\text{sMD}}$ (plain face ‘w’s below are values from the within-view affinity matrices). Thus, the above statement can be re-expressed as

$$W^{(1\,2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The $W^{(1\,3)}$ submatrix is computed as follows

$$\begin{aligned} W^{(1\,3)} &= \begin{bmatrix} w(\mathbf{a}^{(1)}, \mathbf{a}^{(3)}) & w(\mathbf{a}^{(1)}, \mathbf{c}^{(3)}) & w(\mathbf{a}^{(1)}, \mathbf{d}^{(3)}) \\ w(\mathbf{b}^{(1)}, \mathbf{a}^{(3)}) & w(\mathbf{b}^{(1)}, \mathbf{c}^{(3)}) & w(\mathbf{b}^{(1)}, \mathbf{d}^{(3)}) \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{w}(\mathbf{a}^{(1)}, \mathbf{a}^{(1)}) \\ \mathrm{w}(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}) \end{bmatrix} \times \begin{bmatrix} \mathrm{w}(\mathbf{a}^{(3)}, \mathbf{a}^{(3)}) & \mathrm{w}(\mathbf{a}^{(3)}, \mathbf{c}^{(3)}) & \mathrm{w}(\mathbf{a}^{(3)}, \mathbf{d}^{(3)}) \end{bmatrix} \end{aligned}$$

The computation of the other submatrices is given in Appendix B.

3 Application to spectral clustering

Spectral clustering is a successful method for clustering patterns that operates on the pairwise affinity matrix **W** between all pairs of patterns. In this paper, we consider spectral clustering (specifically the version given in Ng et al. 2001) as a way of evaluating the graphs that we construct. The algorithm takes the affinity matrix, normalizes it to get the normalized graph Laplacian **L**, and computes the eigenvectors of **L**. It can be shown that the second eigenvector of the normalized graph Laplacian is a relaxation of a binary vector solution that minimizes the normalized cut on a graph (Shi and Malik 1997). Spectral clustering performs well with non-Gaussian clusters and is easily implementable. It is also non-iterative with no local minima which makes it well suited for evaluating multiple different graph construction algorithms. The Ng et al. (2001) generalization to multiclass clustering (which we will build on) is summarized below for data patterns $\mathbf{x}_i$ to be clustered in to k clusters.

- Form the affinity matrix $\mathbf{W}(i, j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$
- Set the diagonal entries $\mathbf{W}(i, i) = 0$
- Compute the normalized graph Laplacian as $\mathbf{L} = \mathbf{D}^{-0.5}\mathbf{W}\mathbf{D}^{-0.5}$ where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}(i, i) = \sum_j \mathbf{W}(i, j)$

- Compute top k eigenvectors of $\mathbf{L}$ and place as columns in a matrix $\mathbf{X}$
- Form $\mathbf{Y}$ from $\mathbf{X}$ by normalizing the rows of $\mathbf{X}$
- Run k-means to cluster the row vectors of $\mathbf{Y}$
- Pattern $\mathbf{x}_i$ is assigned to cluster c iff row i of $\mathbf{Y}$ is assigned to cluster c

It is straightforward to use our multi-view derived affinity matrix as the affinity matrix for spectral clustering. See Appendix C for a particularly efficient way to compute the normalized affinity matrix in the two-view case. The final clustering/segmentation is obtained from the top eigenvectors. There are several slightly different ways to cluster the values of this eigenvector. We use the prescription of Ng, Jordan and Weiss where $\mathbf{Y}$ is obtained as follows (in MATLAB code)

```
Wv1 =exp(-distmatview1/(2*sigsq1));
Wv2 =exp(-distmatview2/(2*sigsq2));
Wv12=Wv1*Wv2;
Drow=(sum(Wv12'));
Dcol=(sum(Wv12));
Lw=diag(Drow.^(-.5))*Wv12*diag(Dcol.^(-.5));
[U,S,V]=svds(Lw)
X=[U(:,1:numclusts);V(:,1:numclusts)];
Xsq=X.*X;
divmat=repmat(sqrt(sum(Xsq')'),1,numclusts);
Y=X./divmat;
```

Note that computing the SVD of the matrix $\mathbf{L_W} = \mathbf{D}_{\text{row}}^{-0.5}\mathbf{W}\mathbf{D}_{\text{col}}^{-0.5}$, gives two sets of eigenvectors, those of $\mathbf{L_W}\mathbf{L}_\mathbf{W}^{\mathrm{T}}$ in the matrix $\mathbf{U}$ and those of $\mathbf{L}_\mathbf{W}^{\mathrm{T}}\mathbf{L_W}$ in the matrix $\mathbf{V}$, above. The algorithm above concatenates these to form the matrix $\mathbf{Y}$ (as one would get if performing spectral clustering on the large matrix $\mathbf{W}_{\text{sMD}}$). This thus provides clusters for each view of each pattern. To get a cluster for the multi-view pattern, when both views are approximately equally reliable, the top p rows of the $\mathbf{Y}$ matrix can be averaged with the bottom p rows before the k-means step. If one view is significantly more reliable than the other (e.g. auditory speech versus visual speech), one can just use the $\mathbf{Y}$ entries corresponding to the more reliable view (the eigenvectors of $\mathbf{L_W}\mathbf{L}_\mathbf{W}^{\mathrm{T}}$ reveal the clustering for the view 1 segments and the eigenvectors of $\mathbf{L}_\mathbf{W}^{\mathrm{T}}\mathbf{L_W}$ for the view 2 segments).

3.1 Relationship to co-clustering

It should be noted that our spectral clustering application solves the same (modulo some slight differences between spectral clustering implementations) generalized eigenproblem as the co-clustering algorithm of Dhillon (2001). There is however one key difference: in co-clustering the affinity matrix is simply given as a co-occurence matrix (between words and documents in their case) as opposed to being constructed as a product of affinity matrices on two different views. Dhillon shows how spectral clustering can simultaneously cluster words and documents. Our algorithm would reduce to co-clustering if the affinity matrices/kernels in the individual spaces were the identity matrix (for example no similarity information is given to indicate that cat and feline are similar words). Note this does not result in a trivial problem in the case of many to many mappings (as observed for words and documents (many documents contain the same word and many words appear in the same document)). Thus another way to view our algorithm is as a way of introducing within-view similarities to co-clustering.

3.2 Relationship to kernel CCA with Gaussian kernels

Canonical Correlation Analysis (Hotelling 1936) is perhaps the most classical unsupervised multi-view algorithm, and it has recently been used for multi-view clustering (Chaudhuri et al. 2009). Our algorithm is quite similar in form but different in details to Kernel Canonical Correlation Analysis (KCCA) (Hardoon et al. 2003, 2004) with Gaussian kernels followed by k-means segmentation (Blaschko and Lampert 2008).

The KCCA algorithm solves the generalized eigenproblem

$$\begin{bmatrix} \mathbf{0} & \mathbf{W}^{(1)}\mathbf{W}^{(2)} \\ \mathbf{W}^{(2)}\mathbf{W}^{(1)} & \mathbf{0} \end{bmatrix} \mathbf{y} = \lambda \begin{bmatrix} \mathbf{W}^{(1)}\mathbf{W}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{(2)}\mathbf{W}^{(2)} \end{bmatrix} \mathbf{y}$$

and our algorithm solves the generalized eigenproblem

$$\begin{bmatrix} \mathbf{0} & \mathbf{W}^{(1)}\mathbf{W}^{(2)} \\ \mathbf{W}^{(2)}\mathbf{W}^{(1)} & \mathbf{0} \end{bmatrix} \mathbf{y} = \lambda \begin{bmatrix} \mathbf{D}_{\text{row}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\text{col}} \end{bmatrix} \mathbf{y}$$

where $\mathbf{D}_{\text{row}}$ is the diagonal matrix of row sums of $\mathbf{W}^{(1)}\mathbf{W}^{(2)}$ and $\mathbf{D}_{\text{col}}$ is the diagonal matrix of column sums of the same matrix. We hypothesize that the differences in normalization given by the matrix on the right hand side reflect a pressure for the KCCA algorithm to remove within-view affinity information (and rely strictly on co-occurences) similar to the co-clustering algorithm mentioned in the subsection above. The normalization represented by the matrix on the right hand side of our eigenproblem aims to normalize for unequal vertex connectivity.

In addition, KCCA assumes centered kernels and our algorithm actually returns $\mathbf{D}^{0.5} * \mathbf{y}$ (following the algorithm of (Ng et al. 2001)). The specific algorithm of Blaschko and Lampert (2008) is also regularized and automatically searches for good regularization parameters. KCCA without regularization leads to degenerate solutions in the case of full rank kernel matrices (as with Gaussian kernels).

We believe a contribution of our work is to motivate the use of the Gaussian kernel in Kernel CCA and to give a formulation where missing views can be easily incorporated (see Sects. 2.1 and 2.2). This is not easily done for KCCA (see for example Kleine et al. 2008), but at least one attempt has been made (Blaschko et al. 2008).

We compare our algorithm to the Kernel CCA algorithm (using code from (Blaschko and Lampert 2008), specifically denoted as **KCCA**) when it is feasible. We found it too slow to run on the text/webpage dataset.

4 Comparison of algorithms on an illustrative artificial dataset

How do the **JOINT**, **SUM** and **sMD** algorithms differ? Figure 3 contains a simple example showing where clustering $\mathbf{W}_{\text{JOINT}}$ and $\mathbf{W}_{\text{sMD}}$ will lead to different results. The datapoints are numbered for the purposes of discussion. Consider in particular the membership of the circled datapoint (4). The **sMD** algorithm would cluster it with datapoints 1, 2 and 3. The **JOINT** algorithm is much more likely (over a wider range of parameters and noise levels) to cluster datapoint 4 with datapoints 5, 6, 7 and 8. In this purely artificial problem, there is no clear best answer. However in the next sections we will show a real-world motivated example and then a real-world example where the situation is a noisy version of what is shown here and the desired outcome is to group datapoint 4 with 1, 2, and 3.

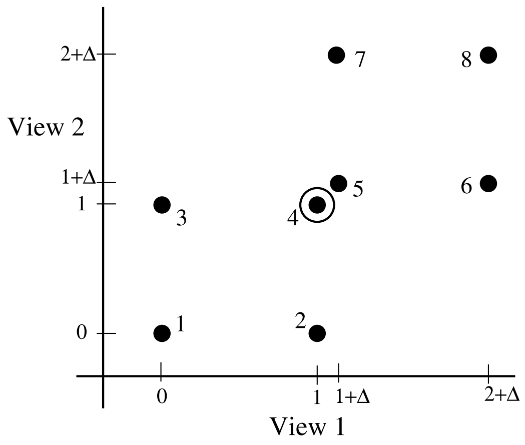


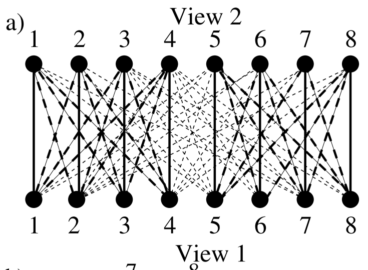
Fig. 3 A simple example that would give different solutions clustered in the joint space **JOINT**, than if the **sMD** algorithm was used

To quantify this effect, we construct an affinity matrix for each view ($\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$) from the example in Fig. 3 and run spectral clustering algorithms on noisy versions of these affinity matrices for varying levels of noise and varying cross-cluster strength. Using a scale parameter $\sigma^2 = 0.1$, we obtain the following affinity matrices (rounded to one decimal place). m is given by the relative spacing between the two clusters with respect to the σ^2 parameter in the spectral clustering algorithm $m = \exp^{-\Delta^2/(2\sigma^2)}$.

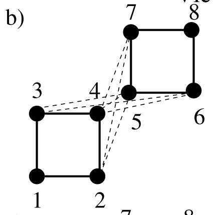
$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & m & 0 & m & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & m & 0 & m & 0 \\ 0 & m & 0 & m & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & m & 0 & m & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad \mathbf{W}^{(2)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & m & m & 0 & 0 \\ 0 & 0 & 1 & 1 & m & m & 0 & 0 \\ 0 & 0 & m & m & 1 & 1 & 0 & 0 \\ 0 & 0 & m & m & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

We performed simulations with placement noise for each simulation (Gaussian noise with mean 0 and standard deviation = 0.001), For $\Delta = 1$, all three algorithms correctly cluster nodes 1–4 and 5–8. However for $\Delta \leq 0.5$ the **JOINT** method breaks down and groups one of nodes 4 or 5 with the wrong cluster. The **SUM** algorithm breaks down for $\Delta \leq 0.2$ and the **sMD** algorithm continues to group appropriately until $\Delta \leq 0.13$. Figure 4 explains these results graphically as well as showing the actual (zero noise) matrices computed $\mathbf{W}_{\text{JOINT}}$, $\mathbf{W}_{\text{SUM}}$, and $\mathbf{W}_{\text{sMD}}$ (rounded to one decimal place except for $\mathbf{W}_{\text{JOINT}}$ which has been rounded to two decimal places due to the dearth of large entries ($e = 0.01$).

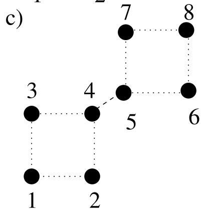
We also ran **KCCA** on this problem. With the scale parameter fixed at $\sigma^2 = 0.1$ and the default code from Blaschko and Lampert (2008) the algorithm didn't consistently group nodes 1–4 and 5–8 even for Δ larger than 2. By symmetricizing the output (similar to **sMD**), we got an algorithm that was very sensitive to the amount of noise added. With low placement noise (standard deviation = 0.001), it performed as desired for Δ as low as 0.08. however if the noise is increased to have standard deviation 0.03, it doesn't perform consistently correctly even for Δ as big as 2 (in this case it does not group 4 and 5 together but performs seemingly random segmentations). **SUM** and **sMD** perform well at this noise level for Δ larger than 0.4 and **JOINT** performs well for Δ larger than 0.7. Our hypothesis is that the



$$\mathbf{W}^{(1\,2)}_{sMD} = \begin{bmatrix} 1 & 1 & 1 & 1+m^2 & m & m & 0 & 0 \\ 1 & 1 & 1 & 1 & 2m & 2m & m & m \\ 1 & 1 & 1 & 1+m^2 & m & m & 0 & 0 \\ 1 & 1 & 1 & 1 & 2m & 2m & m & m \\ m & m & 2m & 2m & 1 & 1 & 1 & 1 \\ 0 & 0 & m & m & 1 & 1 & 1 & 1 \\ m & m & 2m & 2m & 1 & 1 & 1 & 1 \\ 0 & 0 & m & m & 1 & 1 & 1 & 1 \end{bmatrix}$$



$$\mathbf{W}_{SUM} = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & m & 0 & m & 0 \\ 1 & 0 & 2 & 1 & m & m & 0 & 0 \\ 0 & 1 & 1 & 2 & 2m & m & m & 0 \\ 0 & m & m & 2m & 2 & 1 & 1 & 0 \\ 0 & 0 & m & m & 1 & 2 & 0 & 1 \\ 0 & m & 0 & m & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \end{bmatrix}$$



$$\mathbf{W}_{JOINT} = \begin{bmatrix} 1 & e & e & 0 & 0 & 0 & 0 & 0 \\ e & 1 & 0 & e & 0 & 0 & 0 & 0 \\ e & 0 & 1 & e & 0 & 0 & 0 & 0 \\ 0 & e & e & 1 & m^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & m^2 & 1 & e & e & 0 \\ 0 & 0 & 0 & 0 & e & 1 & 0 & e \\ 0 & 0 & 0 & 0 & e & 0 & 1 & e \\ 0 & 0 & 0 & 0 & 0 & e & e & 1 \end{bmatrix}$$

Fig. 4 The resulting graphs (and matrices) resulting from the three algorithms (**a**) **sMD**, (**b**) **SUM**, (**c**) **JOINT** applied to the matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ above. The *light lines* correspond to weights of m and $2m$ and the *dark lines* correspond to weights of 1 and $1+m^2$. In (**a**) the *solid lines* correspond to *co-occurrence lines* and the *dashed lines*, inferred relationships. In (**c**) the *faint dotted lines* are very low weight connections (0.01). *Each algorithm tries to find the smallest normalized cut in its graph*

normalization that **KCCA** does removes information about spatial similarity in each of the individual views and leaves the algorithm more sensitive to noise. (On the other hand in the no-noise case the simple example here would be perfectly separated for even the smallest separations if only co-occurence information is used.)

5 An artificial fMRI example

In Sect. 6, we apply our algorithm to real fMRI data in order to cluster voxels into "functionally similar" volumes. As it is real experimental data, we lack a "ground truth" answer for the locations of the functional areas. To address this lack, we supplement the application to real data with an application to simulated data designed to mimic the real data. In the simulated data we specifically create two different groups of voxels with different functional responses.

Our example consists of a 4 by 4 patch of voxels. Each voxel is assigned to one of two "functional classes". Under the assumption that neighboring voxels are more likely to be performing similar computation, we make each functional class spatially contiguous. The particular example we used is shown in Fig. 5.

In our artificial example, we simulate an experiment recording the response from our 16 voxel patch to presentation of two types of stimuli as two Gaussian distributions, summarized in Table 1.

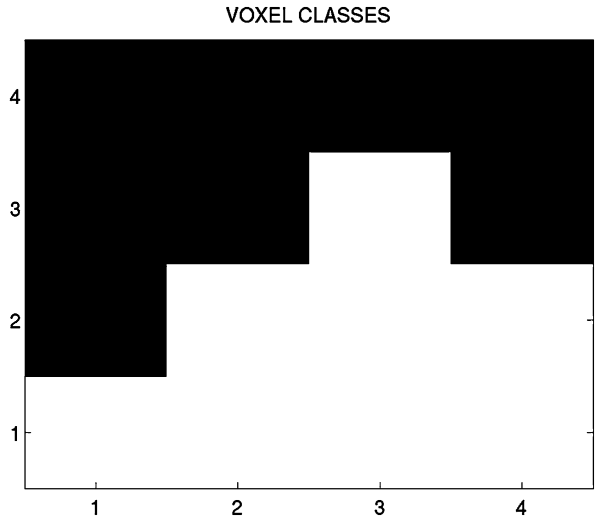


Fig. 5 The layout of our artificial fMRI example. *Black squares* represent voxels from functional class 1 and *white squares* represent voxels from functional class 2. All voxel figures for this section have the voxels represented in this consistent spatial configuration

Table 1 The Gaussian pdf parameters of voxel response to the different stimulus types

	Stimulus type 1		Stimulus type 2	
	Mean	std. dev.	Mean	std. dev.
Voxel class 1	0.25	1.0	−0.25	1.0
Voxel class 2	−0.25	1.0	0.25	1.0

Any drawn values that were greater than 2 or less than -2 were replaced by 2 and -2 respectively. The truncation range and parameter values were chosen to approximate the statistics seen in the real fMRI data that we use in Sect. 6. The responses to each stimulus and for each voxel are drawn independently. We simulate an experiment where 20 stimuli from each of the 2 stimulus types are presented. Thus the overall functional response for each voxel is a 40-Dimensional vector. These vectors were then normalized as is common in dealing with fMRI data.

In addition to the functional response view (which is 40-Dimensional) we have a 2-Dimensional (x, y) spatial view (which simply gives the spatial location of each voxel). For simplicity, each voxel is defined to be 1 unit long in the x and y values. The hope is that the spatial information will help to smooth out noisy responses (e.g. an "unsure" voxel surrounded by voxels from one cluster may be pulled to join that cluster).

Kernel matrices for each view are constructed using the Gaussian kernel. Spectral clustering results depend heavily on the spread or σ^2 parameter that reflects how close vectors need to be to be considered similar in each of the views. We performed a 2-Dimensional search for optimal values over the range $0.05, 0.1, 0.5, 1, 5, 10, 100, 1000$ (extended to 10000 for **KCCA**). We ran a suite of one hundred experiment simulations (each with 40 stimuli) at each pair of parameters to observe the performance of each method. This suite was repeated five times to establish an estimate of variability. (For **KCCA** less repetitions were done due to the long training time for this algorithm.) The resulting performance graphs are shown in Fig. 6. Each curve shows the performance as one parameter is varied for the optimal value of the other parameter (These optimal parameters were different for each algorithm.).

The **sMD** algorithm is striking in its excellent performance over a large range of parameters. Note that due to the ambiguity inherent in unsupervised cluster labeling the baseline

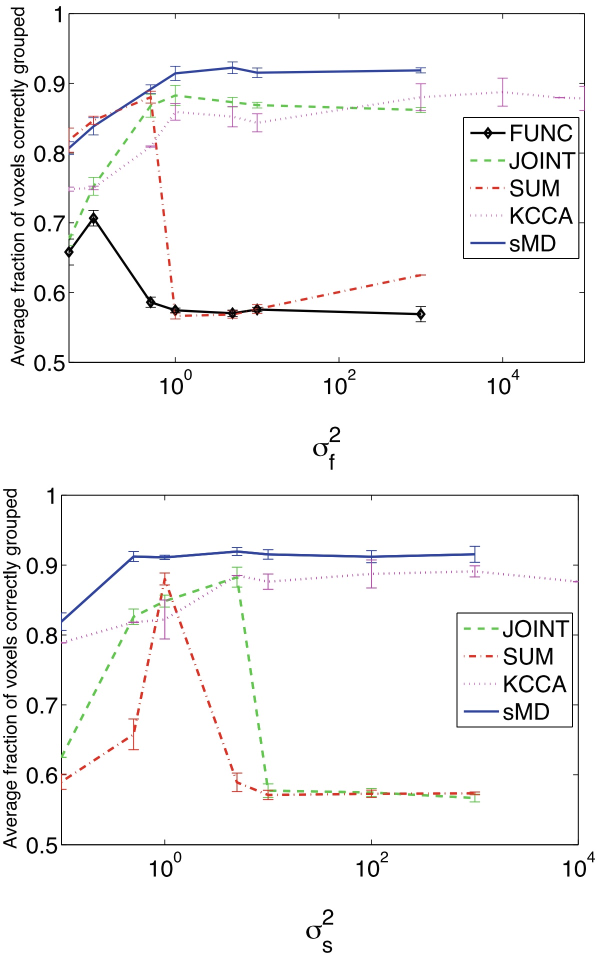


Fig. 6 Comparisons of performance and robustness. FUNC refers to the algorithm that uses only the functional responses and ignores spatial proximity information. **sMD** clustered points in the functional view (the eigenvectors of $\mathbf{L}_\mathbf{W}^T\mathbf{L}_\mathbf{W}$). **KCCA** used the functional view embedding and searched over the default 10 regularization values. The *figures* show the fraction of correct clusterings per voxel as a function of the spread parameter in the Gaussian kernel used for computing **functional** similarity (*top*) and **spatial** similarity (*bottom*). For each graph the non-varying parameter was set optimally. Each datapoint is shown as the mean of five runs of 100 simulations each (**KCCA** has a few less runs due to the long simulation times). The *error bars* show standard deviations across the five runs

performance is higher than 50% (since we choose the more charitable reading of assignment labels from two alternatives where performance at chance would be 50%). More insight can be gained by looking at examples of error fractions for each voxel across different algorithms and across different parameter values. In Fig. 7 we show the results of each algorithm at their optimal parameter settings.

In Fig. 8 we show how the combined kernel's dependence on the spatial information is lessened by increasing the spread parameter (σ^2) for the spatial kernel. For large values of the spread parameter in the Gaussian spatial kernel, all points are considered highly similar and every entry of the spatial kernel matrix approaches 1. As the **JOINT** algorithm uses the Hadamard (element-wise) product to combine the spatial kernel with the functional kernel, large values of the spatial spread parameter effectively downgrade the importance of the spatial information. The opposite is true for small values of the spread spatial spread parameter; in this case, the spatial information increases in importance. To see this, notice in Fig. 8 how small values of σ^2 in the spatial view lead to large errors on the voxels that differ in class from the majority of its neighbors.

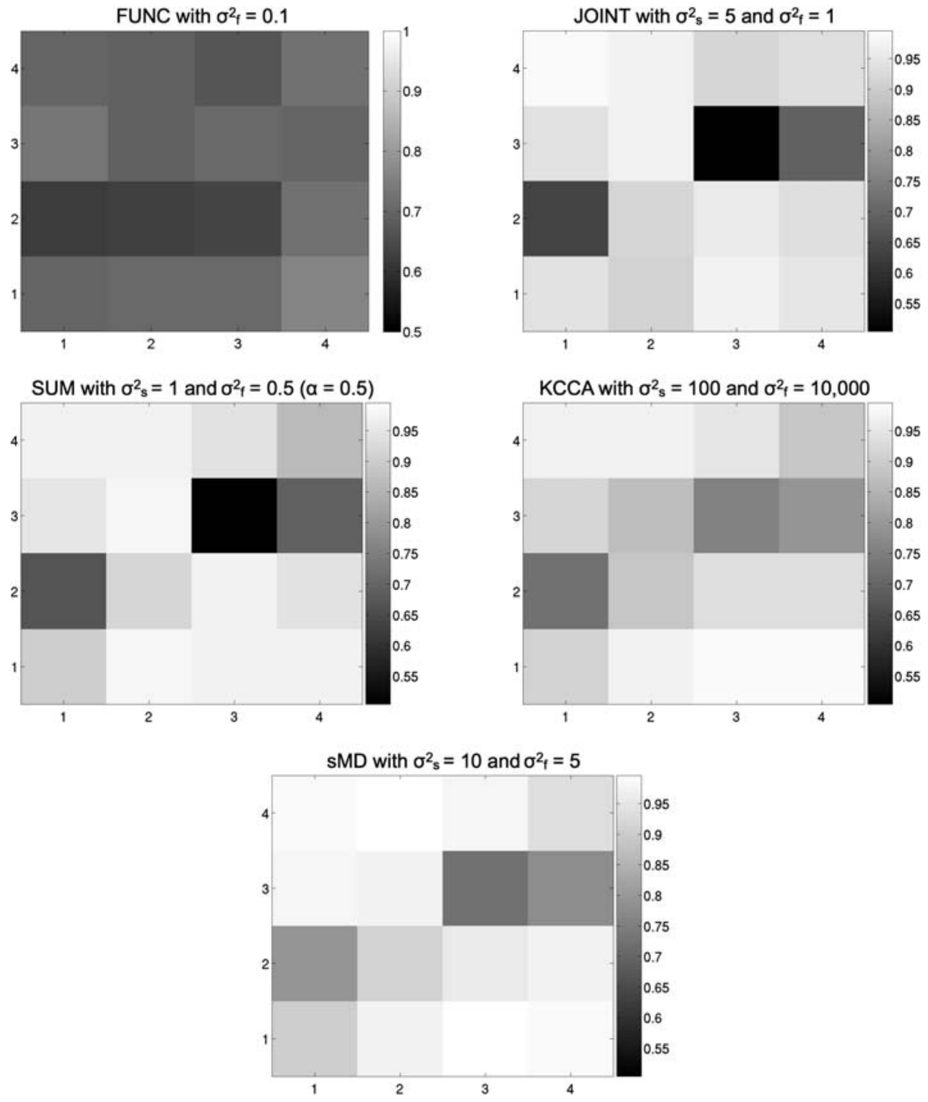


Fig. 7 Spatial layout of the performance each algorithm at its optimal parameter settings (within the values tried). FUNC refers to the algorithm that uses only the functional responses and ignores the spatial proximity information. The shade of *gray* for each voxel gives the fraction of trials when it was correctly grouped (scale given at right of each image)

The **JOINT** and **SUM** algorithms appear to be overly influenced by the spatial input even at their optimal parameter settings. In the **SUM** algorithm we can also easily adjust the weighting of the two views by adding another parameter, α, that weights the terms in the addition of the affinity matrices.

$$\mathbf{W}_{\mathrm{SUM}} = \mathbf{W}_{\mathrm{functional}} + \alpha \mathbf{W}_{\mathrm{spatial}} \tag{6}$$

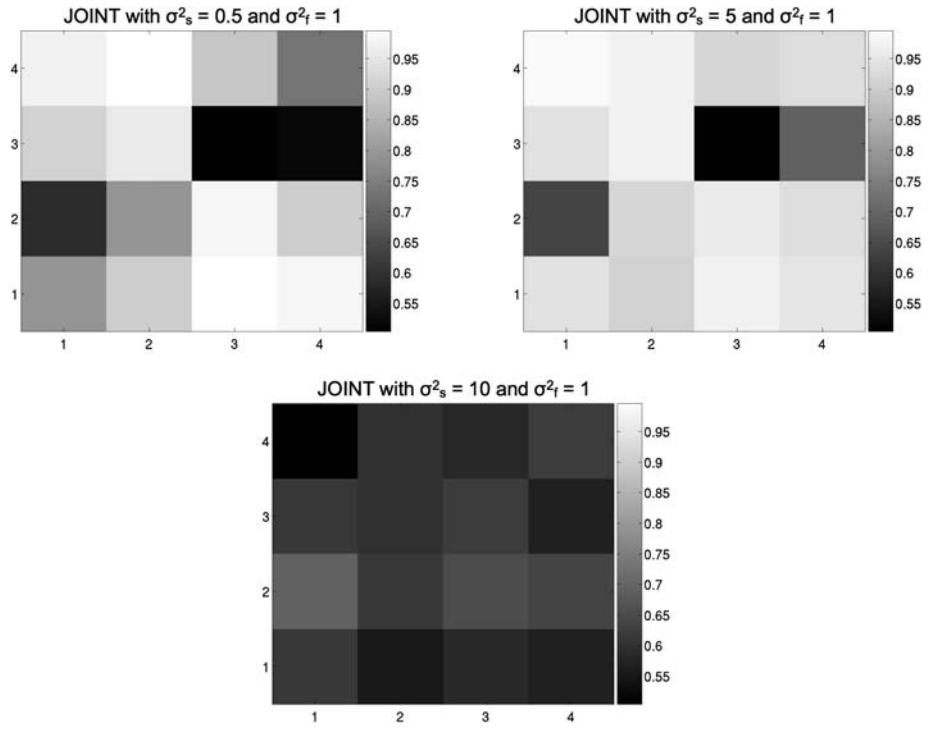


Fig. 8 The effect of varying the spread in the Gaussian spatial kernel for the **JOINT** algorithm. The shade of *gray* for each voxel gives the fraction of trials when it was correctly grouped (scale given at right of each image)

In fact, as preliminary experiments demonstrated that $\alpha < 1$ was preferable, all the experiments above were done with $\alpha = 0.5$. We show the dependence of the results on α in Fig. 9 and some examples in Fig. 10. Notice how varying α for the **SUM** algorithm has a similar effect to varying the spread parameter in the spatial kernel. In particular, both α and the spatial spread parameter can be changed to vary the amount of smoothing as desired. When parameter values are set to encourage smoothing, solutions where voxels have the same class as their neighbors will be favored. Similarly, parameter values that deprecate smoothing will tend to lead to solutions where voxels need not have the same class as their neighbors.

Overall, it is notable that the **sMD** algorithm maintains the best performance over a large range of parameters. This robustness is important in real applications where the correct answers are not known or where detailed parameter searches are infeasible.

We can provide insight to the **sMD** algorithm's robust performance on this application by referring back to the previous section. In this problem we have two views, a spatial one and a functional one. The spatial view is 2-dimensional (representing x position and y position) and the functional view is 40-Dimensional (representing the response to 20 stimuli of each of the 2 types). Consider, however, a reduction of this dataset to 1 dimension for each view. Each data point in this simplified situation thus consists of a single scalar spatial location value and a single scalar functional response to visual stimulus value). In order to make the graph a little clearer, assume that the 1-Dimensional spatial axis extends from 1 to 8 with a boundary between voxels 4 and 8. (We have also greatly reduced the amount of noise

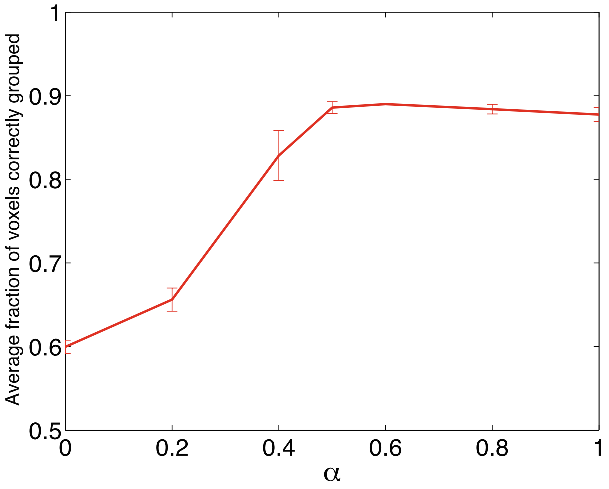


Fig. 9 Graph of performance of the **SUM** algorithm for optimal spread parameters as α (reflecting the relative weighting of the spatial and functional kernels) is varied. Each datapoint is shown as the mean of four runs of 100 simulations with error bars showing the standard deviation between the runs

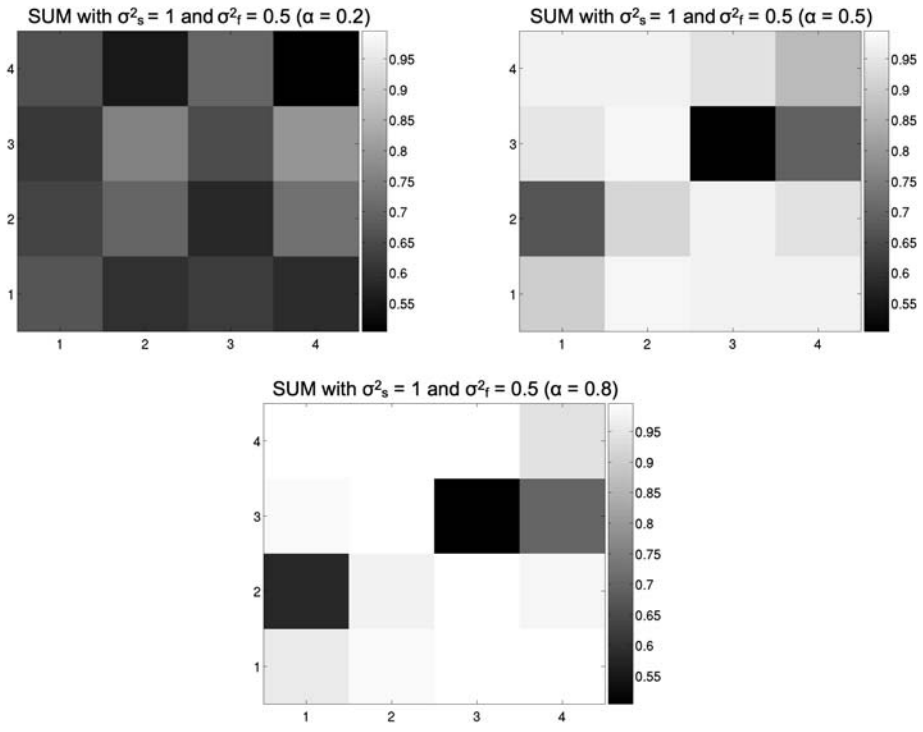


Fig. 10 Examples of multiple runs of the **SUM** algorithm for different values of α and the corresponding optimal spread parameters in each kernel. The shade of *gray* for each voxel gives the fraction of trials when it was correctly grouped (scale given at right of each image)

but remember in the actual simulation, there are 40 functional dimensions instead of 1.) Now, under the assumptions behind our data generation (based on what is known/assumed about real fMRI data), voxels 1–4 are generated from a truncated Gaussian with one mean

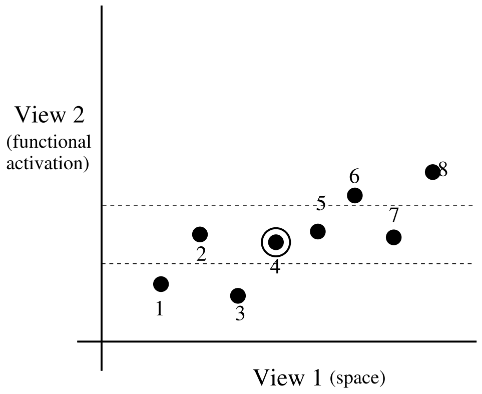


Fig. 11 An example with two 1-dimensional views similar to the simple example from the last section. The *dashed lines* represent the two means for the different voxel classes. Note that the correct answer is to cluster points 1–4 together. This is just a skewed and noisy version of the simple example in Fig. 3. Based on the arguments and analysis in that section, one can see that **JOINT** would be less likely than **sMD** to group 4 with 1, 2, and 3 in agreement with the empirical results for the higher dimensional problems shown above

and voxels 5–8 are generated from a truncated Gaussian with a different mean. A possible outcome is shown in Fig. 11 which is just a noisy version of Fig. 3. Note that as the data are stochastic, some possible data would allow **JOINT** to correctly separate the two voxel types but those will also be easy cases for **sMD**. In the next section, we show that with real fMRI data, the **sMD** algorithm gives a nice voxel partition that seems to reflect a similar process.

6 Application to fMRI data

Our goal in clustering fMRI data is functional segmentation: we wish to discover regions of voxels which have similar activity patterns across the stimuli (Golland et al. 2007; Golland et al. 2008). Such analysis can reveal intrinsic systems of voxels which behave similarly, which would not be noticed by a conventional linear regression/hypothesis testing approach. In contrast to other methods of grouping voxels such as principal component analysis or independent component analysis, clustering produces a decomposition of the data which is more easily interpretable. We would also prefer that our clusters be spatially smooth, reflecting prior beliefs about the spatial nature of brain organization. Smoothing the data is one way to introduce smoothness, but it would destroy fine-grained spatial data. Our algorithm allows us to weight to prefer spatial proximity, but doesn't force spatial proximity, unlike earlier work on constrained clustering approaches. Some of these algorithms (Wagstaff et al. 2001) introduce a version of k-means with a set of hard constraints, *must link* and *cannot link*, for samples which should or should not be clustered together. More recent approaches combine Gaussian mixture models with soft or probabilistic constraints (Law et al. 2004; Law et al. 2005; Lu and Leen 2005; Lu and Leen 2007). The conditional random fields (CRF) approach has also been used to spatially smooth fMRI responses in a data-dependent way (Woolrich et al. 2005; Wang and Rajapakse 2006). These approaches often involve a lot of human effort to craft the energy function and a lot of computer effort to estimate probability distributions. In this case, we use our general purpose algorithm for combining different sources to combine spatial and functional signals.

We tested our algorithm by clustering data from a single subject (Haxby et al. 2001).[2] Our analysis was restricted to a subset of 577 object-selective voxels, mostly located in the inferior temporal lobe. This dataset presented participants with a set of images from eight categories, with the result that each voxel responds to several categories, and from the distributed pattern of activity, object category can be decoded. Each voxel was z-scored, and then averaged within each block (mean of 20 seconds/10 images of activity). We clustered on either the functional activation (vector of mean activations), spatial proximity, or combinations computed using **JOINT**, **SUM**, **KCCA** or **sMD**, in order to find functionally similar groups of voxels.

Algorithm	σ^2 Spatial	σ^2 fMRI	Coherence	Symmetry
SPACE	0.1	N/A	1.000	0.0616
FUNC	N/A	0.10	0.6300	0.4658
JOINT	0.05	0.10	0.9968	0.8973
SUM	0.5	0.1	0.9344	0.9384
KCCA	100	10	0.9637	0.9452
sMD	5	0.50	0.9653	0.9589

To quantitatively examine the clustering results, we compared clusterings (see Fig. 12) for spatial coherence and symmetry, which one might expect given the known topography and gross structural symmetry between hemispheres. Coherence is given by a normalized sum over all points of how many of a voxel's nearest neighbors are in the same class. Symmetry gives a normalized count of the number of voxels that match their reflected voxel around the Y (mid-line) axis. Note that our goal is to find functionally similar groups of voxels. Without the "ground truth", there is no way to evaluate which method is performing the best. The symmetry and coherence measures are heuristic measures that one might hope a good solution would rate highly on. One should not expect them to be maximized by the "correct" clustering though.

The proximity(spatial) information does not encourage a symmetric partition. Clustering on the spatial information alone divides the brain into left and right hemispheres. Spectral clustering on functional data alone shows a "speckled" spatial pattern (results of clustering with straight k-means produced a very similar plot). The **SUM**, **sMD** and **KCCA** results are nicely smooth and symmetric.

7 Clustering results with the course web page dataset

We also applied our algorithm to a commonly used multi-view dataset of course web pages. This dataset consists of two views: the first view consists of text on the web page and the second view consists of text on the links to the web page (Blum and Mitchell 1998). We use the six class (course, department, faculty, project, staff, student) version in Bickel and Scheffer (2004) consisting of tfidf[3] vectors without stemming. Patterns were normalized

[2]This subject's data was made available as part of the MVPA toolbox, which we also used for data import and visualization functions. http://www.csbmb.princeton.edu/mvpa/.

[3]Term frequency inverse document frequency—where a document is stored as a vector of weighted words. tfidf weights words more if they occur more in a document and downweights words that occur often in the full dataset.

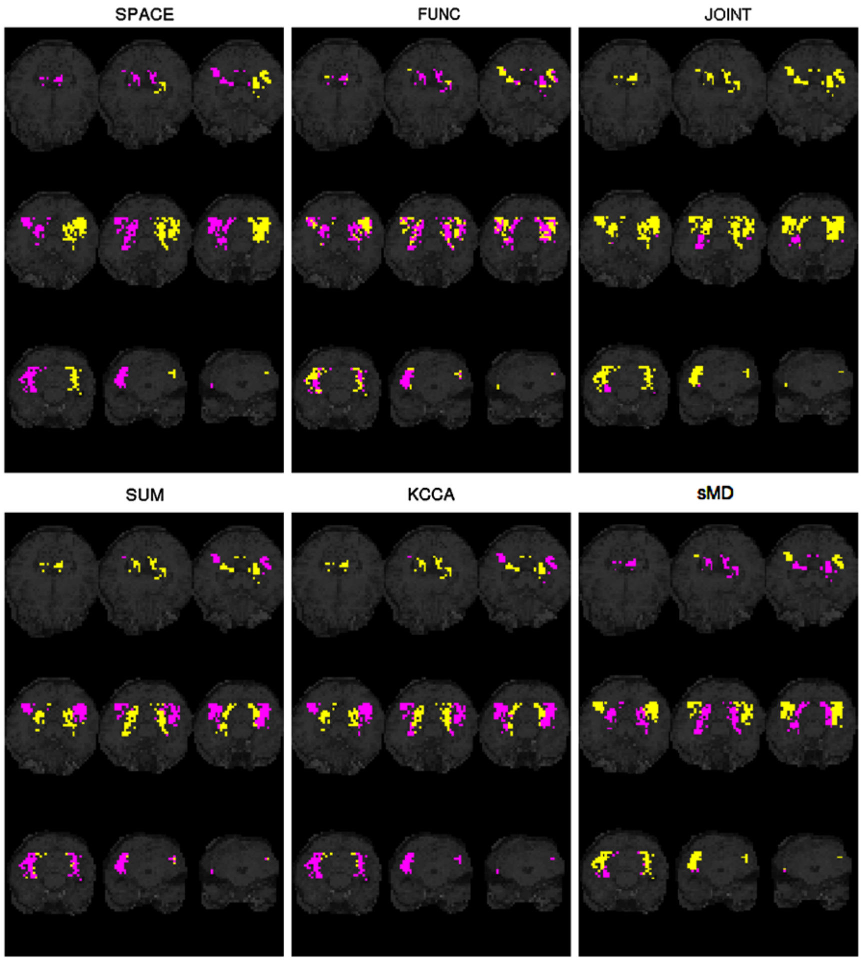


Fig. 12 Results of clustering fMRI data, for spatial location only, for functional data only, **JOINT**, **SUM**, **KCCA**, and **sMD**. **sMD** clustered points in the spatial view. **KCCA** used the functional view embedding and searched over 4 regularization values. These partitions were chosen because they score well on coherence and symmetry and had relatively balanced clusters. Coherence is based on how many neighbors are the same class, and symmetry looks for the same class patterns on each side of the center

within each view so that squared distances reflected the commonly used cosine similarity measure.

We use the average entropy error metric of Bickel and Scheffer (2004)

$$E = \sum_{i=1}^{k} \frac{m_i(-\sum_j p_{ij} \log_2(p_{ij}))}{m} \tag{7}$$

where p_{ij} is the proportion of cluster i that is from class j, m_i is the number of patterns in cluster i and m is the total number of patterns. On this dataset, with this error measure, perfect agreement would result in $E = 0$, everybody in the same cluster would give $E = 2.219$

Table 2 Average entropy where 2084 (90%, *top pane*) or 1158 (50%, *bottom pane*) of the patterns have both views. Alternatives A and B described in the text, plus **sMD** are considered. All values are given ± 1 standard error of the mean over 10 runs. All errors are using the average entropy error measure

	A	B	**sMD**
90% Paired			
Views 1 & 2	1.66 ± 0.003	1.68 ± 0.002	1.68 ± 0.003
View 1 only	1.83 ± 0.02	1.64 ± 0.02	1.63 ± 0.02
View 2 only	1.95 ± 0.02	2.04 ± 0.003	1.83 ± 0.02
50% Paired			
Views 1 & 2	1.67 ± 0.01	1.69 ± 0.002	1.64 ± 0.01
View 1 only	1.90 ± 0.02	1.68 ± 0.006	1.66 ± 0.006
View 2 only	2.04 ± 0.006	2.04 ± 0.003	1.95 ± 0.006

(and equal size clusters with probability measurement equal to the base class probabilities also gives $E = 2.2$).

We first compared the algorithms on the full dataset by searching for a good σ_1 and σ_2 from clustering in the individual views.

We found that (with the proper normalization), the **JOINT** method worked slightly better ($E = 1.64$) than the **SUM** ($E = 1.70$) and **sMD** version ($E = 1.66$) (standard error estimates are provided later when 90% of the data is used). These methods all performed better than Bickel and Scheffer's reported results using the same error measures (with 6 clusters) of approximately[4] 1.73 (multi-view) and 2.03 (single view) for their mixture-of-multinomials EM algorithm and approximately 1.97 (multi-view) and 2.07 (single view) for their spherical k-means algorithm (Bickel and Scheffer 2004).

As mentioned, when computing the SVD of the matrix $\mathbf{L_W} = \mathbf{D}_{\text{row}}^{-0.5}\mathbf{W}\mathbf{D}_{\text{col}}^{-0.5}$, one gets two sets of eigenvectors, those of $\mathbf{L_W}\mathbf{L_W^T}$ and those of $\mathbf{L_W^T}\mathbf{L_W}$ and for equally reliable views, the $\mathbf{Y}$ matrices can be averaged before the k-means step. For this dataset however, as in the fMRI dataset, view 1 is significantly more reliable than view 2 and we obtain improved performance by simply using the eigenvectors from view 1.

As also previously mentioned, **sMD** easily handles cases where patterns are missing co-occurence data. In this case we use the eigenvectors of $\mathbf{L_W}\mathbf{L_W^T}$ to find the clusters for both the paired and view 1 data and must use the eigenvectors of $\mathbf{L_W^T}\mathbf{L_W}$ to find the clusters for the data that only has view 2.

For comparison, we consider two other alternatives for clustering data that consists of some multi-view patterns and some single view patterns.

Alternative A (Using **JOINT**) cluster only the p patterns consisting of $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_j^{(2)}$ concatenated in the joint space. Spectral clustering will give clusters for these patterns. To report clusters for the $m + n$ unpaired patterns, report the cluster of the nearest same view paired pattern of the pattern.

Alternative B cluster the patterns from each view separately. In this case the pairing information is lost.

Results for different values of p are reported in Tables 2 and 3. Table 2 shows that there is a very slight but significant performance advantage for the multi-view patterns using Alternative A when 2084 (90%) of the patterns have both views, but that Alternatives B and

[4]Estimated from their graph.

Table 3 Average Entropy for **sMD** for varying amount of two-view data. (See Table 1 for an explanation of terms)

	2084 (90%)	1621 (70%)	1158 (50%)	694 (30%)	231 (10%)
Views 1 & 2	1.68 ± 0.003	1.66 ± 0.006	1.64 ± 0.01	1.68 ± 0.01	1.76 ± 0.03
View 1 Only	1.63 ± 0.02	1.66 ± 0.01	1.66 ± 0.006	1.67 ± 0.01	1.73 ± 0.02
View 2 Only	1.83 ± 0.02	1.91 ± 0.01	1.95 ± 0.006	1.97 ± 0.01	2.00 ± 0.01

our **sMD** method perform significantly better on the patterns that only have values for view 1 and our **sMD** method performs significantly better than both alternatives for patterns that only have values for view 2. When only 1158 (50%) of the patterns are provided with two views, the **sMD** algorithm performs significantly better in all categories. Table 3 shows how the **sMD** algorithm varies for different numbers of paired patterns.[5] Performance for the single view data is seen to decrease gradually with less paired training data. One value of an algorithm that can train with multi-view data and report data for single-view data would be when the single-view data arrive at a later time.

8 Conclusion

In this paper we developed a principled method for constructing a kernel over data containing multiple conceptually distinct views. As opposed to alternatives, our method naturally incorporates within-view similarity information and between-view co-occurences without artificially equating or relating them. We accomplish this by constructing a multipartite graph where edges are always between nodes from different views but are weighted by within-view similarities. The eigenvectors of the resulting affinity matrix can be efficiently computed and used to perform spectral clustering simultaneously across all views of the data. We also showed how this can be seen as a generalization of co-clustering and related to Kernel Canonical Correlation Analysis.

In order to evaluate our method we compared it across four datasets against two common alternative algorithms (and where feasible against a third). The two synthetic datasets act as intuition drivers and serve as a proof of concept that our algorithm can be expected to outperform competitors across a wide range of parameters and with noisy data. The results on real data, in both the fMRI and webpage domains, show that our algorithm performs well and is robust to large parameter changes and missing views.

As an extension to this work, our lab is currently working on using $\mathbf{W}_{\mathrm{sMD}}$ as a starting point for a manifold learning algorithm in the style of Laplacian eigenmaps. One could also consider using the Nystrom approximation (Charless Fowlkes Serge Belongie and Malik 2004) for out of sample estimates. This would allow one to train with paired data and provide cluster labels for later unpaired data.

Future work will also involve comparing our method on datasets used in other recent multi-view papers and further exploring the differences between our algorithm and Kernel CCA.

[5]The slight improvement in clustering performance (with increased variance) for the paired view data in the 50% paired case is likely due to an increased chance of not including inappropriate pairs in the paired dataset. Performance decreases with non independent sources of information have been observed with the non-spectral M–D algorithm. If leaving out some data vectors increases the independence between views, we would expect improved performance.

Acknowledgements Many thanks to Ulf Brefeld, Tobias Scheffer, Steffen Bickel, and Anjum Gupta for kindly sending their processed course web page datasets and to Marina Meila and Deepak Verma for providing a great library of spectral clustering code at http://www.cs.washington.edu/homes/deepak/spectral/library.tgz. Warm thanks to Jochen Triesch and Serge Belongie for helpful comments on a very early draft and many thanks to the anonymous reviewers for greatly improving the final version. This material is based upon work supported by the National Science Foundation under Grants CAREER IIS-0133996 and CBET-0756828 to V.R. de Sa and DGE-0333451 to V.R. de Sa and G.W. Cottrell.

Appendix A: Multi-view spectral clustering objective function and eigenproblem

Here we present the initial objective function and the eigenproblem that results. To give context to our derivation, we note that the standard single-view spectral clustering objective function is expressed:

$$J(\mathbf{y}) = \sum_{i,j=1}^{N} \left(y_i - y_j\right)^2 \mathbf{W}_{ij} \tag{8}$$

where $\mathbf{y}$ is the vector of $\{+1, -1\}$ class labels for the N data points and $\mathbf{W}_{ij}$ indicates the "similarity" of points i and j.

The sMD objective function is expressed as follows:

$$J_{\text{sMD}}\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}\right) = \sum_{i,j=1}^{N} \left(y_i^{(1)} - y_j^{(2)}\right)^2 \mathbf{W}_{ij}^{(1\,2)} \tag{9}$$

where $\mathbf{y}^{(v)}$ is the "view v" vector of class labels the N data points and $\mathbf{W}_{ij}^{(1\,2)}$ indicates the "similarity" of point i from view (1) and point j from view (2), calculated from the product of the within-view similarity matrices as described in the main body of the paper.

This objective is then used in the following optimization problem:

$$\min J\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}\right) \tag{10}$$

$$\text{s.t.} \quad \left(y_i^{(1)}\right)^2 = 1, \left(y_j^{(2)}\right)^2 = 1 \quad \text{for all } i, j \tag{11}$$

The integer $\{+1, -1\}$ class label constraints are then relaxed so that the $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$ vectors of labels can be real-valued, and the objective function is reformulated in matrix notation:

$$J_{\text{sMD}}\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}\right) = \sum_{i,j=1}^{N} \left(y_i^{(1)} - y_j^{(2)}\right)^2 \mathbf{W}_{ij}^{(1\,2)} \tag{12}$$

$$= \sum_{i,j=1}^{N} \left(y_i^{(1)}\right)^2 \mathbf{W}_{ij}^{(1\,2)} + \left(y_j^{(2)}\right)^2 \mathbf{W}_{ij}^{(1\,2)} - 2 y_i^{(1)} \mathbf{W}_{ij}^{(1\,2)} y_j^{(2)} \tag{13}$$

$$= \left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{D}_{\text{row}} \mathbf{y}^{(1)} + \left(\mathbf{y}^{(2)}\right)^{\mathrm{T}} \mathbf{D}_{\text{col}} \mathbf{y}^{(2)} - 2\left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{W}^{(1\,2)} \mathbf{y}^{(2)} \tag{14}$$

where $\mathbf{D}_{\text{row}}$ is a diagonal matrix with its (i,i) entry equal to the ith row sum of $\mathbf{W}^{(1\,2)}$ and $\mathbf{D}_{\text{col}}$ is a diagonal matrix with its (i,i) entry equal to the ith column sum of $\mathbf{W}^{(1\,2)}$.

In order to prevent the trivial solution of setting all the relaxed labels to 0, and in order to control for the scale of the entries in the $\mathbf{W}^{(1\,2)}$ matrix, we introduce new constraints, giving us the final constrained optimization problem:

$$\min_{\mathbf{y}^{(1)},\mathbf{y}^{(2)}} \left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{D}_{\text{row}}\,\mathbf{y}^{(1)} + \left(\mathbf{y}^{(2)}\right)^{\mathrm{T}} \mathbf{D}_{\text{col}}\,\mathbf{y}^{(2)} - 2\left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{W}^{(1\,2)}\mathbf{y}^{(2)} \tag{15}$$

$$\text{s.t.} \quad \left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{D}_{\text{row}}\,\mathbf{y}^{(1)} = 1 \tag{16}$$

$$\left(\mathbf{y}^{(2)}\right)^{\mathrm{T}} \mathbf{D}_{\text{col}}\,\mathbf{y}^{(2)} = 1 \tag{17}$$

The above constrained optimization problem gives rise to the following Lagrangian:

$$\begin{aligned}
&\mathcal{L}\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \lambda_1, \lambda_2\right) \\
&\quad = \left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{D}_{\text{row}}\,\mathbf{y}^{(1)} + \left(\mathbf{y}^{(2)}\right)^{\mathrm{T}} \mathbf{D}_{\text{col}}\,\mathbf{y}^{(2)} - 2\left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{W}^{(1\,2)}\mathbf{y}^{(2)} \\
&\qquad + \lambda_1\left(1 - \left(\mathbf{y}^{(1)}\right)^{\mathrm{T}} \mathbf{D}_{\text{row}}\,\mathbf{y}^{(1)}\right) + \lambda_2\left(1 - \left(\mathbf{y}^{(2)}\right)^{\mathrm{T}} \mathbf{D}_{\text{col}}\,\mathbf{y}^{(2)}\right)
\end{aligned} \tag{18}$$

Differentiating with respect to $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$ and setting equal to 0, we find

$$\mathbf{W}^{(1\,2)}\mathbf{y}^{(2)} = (1-\lambda_1)\,\mathbf{D}_{\text{row}}\,\mathbf{y}^{(1)} \tag{19}$$

$$\left(\mathbf{W}^{(1\,2)}\right)^{\mathrm{T}}\mathbf{y}^{(1)} = (1-\lambda_2)\,\mathbf{D}_{\text{col}}\,\mathbf{y}^{(2)} \tag{20}$$

Which we reexpress as the following generalized eigenproblem:[6]

$$\begin{pmatrix} \mathbf{0} & \mathbf{W}^{(1\,2)} \\ \left(\mathbf{W}^{(1\,2)}\right)^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \end{pmatrix} = (1-\lambda) \begin{pmatrix} \mathbf{D}_{\text{row}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\text{col}} \end{pmatrix} \begin{pmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \end{pmatrix} \tag{21}$$

The above eigenproblem relates via a simple matrix operation to the eigenproblem we solve, as indicated in the main body of the paper.

By way of comparison, we further note that SUM can be seen as using the following objective function in the framework of spectral clustering:

$$J_{\text{SUM}}(\mathbf{y}) = \sum_{i,j=1}^{N} (y_i - y_j)^2\,\mathbf{W}_{ij}^{(1)} + \alpha \sum_{i,j=1}^{N} (y_i - y_j)^2\,\mathbf{W}_{ij}^{(2)} \tag{22}$$

Appendix B: Computing submatrices for the missing view example

$$\begin{aligned}
W^{(1\,4)} &= \begin{bmatrix} w(\mathbf{a^{(1)}}, \mathbf{a^{(4)}}) & w(\mathbf{a^{(1)}}, \mathbf{b^{(4)}}) & w(\mathbf{a^{(1)}}, \mathbf{d^{(4)}}) & w(\mathbf{a^{(1)}}, \mathbf{e^{(4)}}) \\ w(\mathbf{b^{(1)}}, \mathbf{a^{(4)}}) & w(\mathbf{b^{(1)}}, \mathbf{b^{(4)}}) & w(\mathbf{b^{(1)}}, \mathbf{d^{(4)}}) & w(\mathbf{b^{(1)}}, \mathbf{e^{(4)}}) \end{bmatrix} \\
&= \begin{bmatrix} \mathrm{w}(\mathbf{a^{(1)}}, \mathbf{a^{(1)}}) & \mathrm{w}(\mathbf{a^{(1)}}, \mathbf{b^{(1)}}) \\ \mathrm{w}(\mathbf{a^{(1)}}, \mathbf{b^{(1)}}) & \mathrm{w}(\mathbf{b^{(1)}}, \mathbf{b^{(1)}}) \end{bmatrix} \\
&\quad \times \begin{bmatrix} \mathrm{w}(\mathbf{a^{(4)}}, \mathbf{a^{(4)}}) & \mathrm{w}(\mathbf{a^{(4)}}, \mathbf{b^{(4)}}) & \mathrm{w}(\mathbf{a^{(4)}}, \mathbf{d^{(4)}}) & \mathrm{w}(\mathbf{a^{(4)}}, \mathbf{e^{(4)}}) \\ \mathrm{w}(\mathbf{b^{(4)}}, \mathbf{a^{(4)}}) & \mathrm{w}(\mathbf{b^{(4)}}, \mathbf{b^{(4)}}) & \mathrm{w}(\mathbf{b^{(4)}}, \mathbf{d^{(4)}}) & \mathrm{w}(\mathbf{b^{(4)}}, \mathbf{e^{(4)}}) \end{bmatrix}
\end{aligned}$$

[6] Equality of the Lagrange multipliers follows because $(\mathbf{y}^{(1)})^T \mathbf{W}^{(1\,2)}\mathbf{y}^{(2)} = 1 - \lambda_1 = 1 - \lambda_2$.

$$W^{(23)} = \begin{bmatrix} w(\mathbf{c}^{(2)}, \mathbf{a}^{(3)}) & w(\mathbf{c}^{(2)}, \mathbf{c}^{(3)}) & w(\mathbf{c}^{(2)}, \mathbf{d}^{(3)}) \\ w(\mathbf{d}^{(2)}, \mathbf{a}^{(3)}) & w(\mathbf{d}^{(2)}, \mathbf{c}^{(3)}) & w(\mathbf{d}^{(2)}, \mathbf{d}^{(3)}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{w}(\mathbf{c}^{(2)}, \mathbf{c}^{(2)}) & \mathrm{w}(\mathbf{c}^{(2)}, \mathbf{d}^{(2)}) \\ \mathrm{w}(\mathbf{d}^{(2)}, \mathbf{c}^{(2)}) & \mathrm{w}(\mathbf{d}^{(2)}, \mathbf{d}^{(2)}) \end{bmatrix} \times \begin{bmatrix} \mathrm{w}(\mathbf{c}^{(3)}, \mathbf{a}^{(3)}) & \mathrm{w}(\mathbf{c}^{(3)}, \mathbf{c}^{(3)}) & \mathrm{w}(\mathbf{c}^{(3)}, \mathbf{d}^{(3)}) \\ \mathrm{w}(\mathbf{d}^{(3)}, \mathbf{a}^{(3)}) & \mathrm{w}(\mathbf{d}^{(3)}, \mathbf{c}^{(3)}) & \mathrm{w}(\mathbf{d}^{(3)}, \mathbf{d}^{(3)}) \end{bmatrix}$$

$$W^{(24)} = \begin{bmatrix} w(\mathbf{c}^{(2)}, \mathbf{a}^{(4)}) & w(\mathbf{c}^{(2)}, \mathbf{b}^{(4)}) & w(\mathbf{c}^{(2)}, \mathbf{d}^{(4)}) & w(\mathbf{c}^{(2)}, \mathbf{e}^{(4)}) \\ w(\mathbf{d}^{(2)}, \mathbf{a}^{(4)}) & w(\mathbf{d}^{(2)}, \mathbf{b}^{(4)}) & w(\mathbf{d}^{(2)}, \mathbf{d}^{(4)}) & w(\mathbf{d}^{(2)}, \mathbf{e}^{(4)}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{w}(\mathbf{c}^{(2)}, \mathbf{d}^{(2)}) \\ \mathrm{w}(\mathbf{d}^{(2)}, \mathbf{d}^{(2)}) \end{bmatrix} \times \begin{bmatrix} \mathrm{w}(\mathbf{a}^{(4)}, \mathbf{d}^{(4)}) & \mathrm{w}(\mathbf{b}^{(4)}, \mathbf{d}^{(4)}) & \mathrm{w}(\mathbf{d}^{(4)}, \mathbf{d}^{(4)}) & \mathrm{w}(\mathbf{e}^{(4)}, \mathbf{d}^{(4)}) \end{bmatrix}$$

and

$$W^{(34)} = \begin{bmatrix} w(\mathbf{a}^{(3)}, \mathbf{a}^{(4)}) & w(\mathbf{a}^{(3)}, \mathbf{b}^{(4)}) & w(\mathbf{a}^{(3)}, \mathbf{d}^{(4)}) & w(\mathbf{a}^{(3)}, \mathbf{e}^{(4)}) \\ w(\mathbf{c}^{(3)}, \mathbf{a}^{(4)}) & w(\mathbf{c}^{(3)}, \mathbf{b}^{(4)}) & w(\mathbf{c}^{(3)}, \mathbf{d}^{(4)}) & w(\mathbf{c}^{(3)}, \mathbf{e}^{(4)}) \\ w(\mathbf{d}^{(3)}, \mathbf{a}^{(4)}) & w(\mathbf{d}^{(3)}, \mathbf{b}^{(4)}) & w(\mathbf{d}^{(3)}, \mathbf{d}^{(4)}) & w(\mathbf{d}^{(3)}, \mathbf{e}^{(4)}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{w}(\mathbf{a}^{(3)}, \mathbf{a}^{(3)}) & \mathrm{w}(\mathbf{a}^{(3)}, \mathbf{d}^{(3)}) \\ \mathrm{w}(\mathbf{c}^{(3)}, \mathbf{a}^{(3)}) & \mathrm{w}(\mathbf{c}^{(3)}, \mathbf{d}^{(3)}) \\ \mathrm{w}(\mathbf{d}^{(3)}, \mathbf{a}^{(3)}) & \mathrm{w}(\mathbf{d}^{(3)}, \mathbf{d}^{(3)}) \end{bmatrix}$$

$$\times \begin{bmatrix} \mathrm{w}(\mathbf{a}^{(4)}, \mathbf{a}^{(4)}) & \mathrm{w}(\mathbf{a}^{(4)}, \mathbf{b}^{(4)}) & \mathrm{w}(\mathbf{a}^{(4)}, \mathbf{d}^{(4)}) & \mathrm{w}(\mathbf{a}^{(4)}, \mathbf{e}^{(4)}) \\ \mathrm{w}(\mathbf{d}^{(4)}, \mathbf{a}^{(4)}) & \mathrm{w}(\mathbf{d}^{(4)}, \mathbf{b}^{(4)}) & \mathrm{w}(\mathbf{d}^{(4)}, \mathbf{d}^{(4)}) & \mathrm{w}(\mathbf{d}^{(4)}, \mathbf{e}^{(4)}) \end{bmatrix}$$

Appendix C: An efficient eigendecomposition of $\mathbf{W}_{\mathrm{sMD}}$ with two views

Here we discuss how to compute eigenvectors of the normalized $\mathbf{W}_{\mathrm{sMD}}$ matrix, $\mathbf{D}^{-0.5}\mathbf{W}_{\mathrm{sMD}}\mathbf{D}^{-0.5}$, where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}(i,i) = \sum_j \mathbf{W}_{\mathrm{sMD}}(i,j)$ (row sums of $\mathbf{W}_{\mathrm{sMD}}$) which is equal to

$$\begin{bmatrix} \mathbf{D}_{\mathrm{row}}^{-0.5} & 0 \\ 0 & \mathbf{D}_{\mathrm{col}}^{-0.5} \end{bmatrix} \begin{bmatrix} 0 & \mathbf{W}^{(12)} \\ (\mathbf{W}^{(12)})^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{D}_{\mathrm{row}}^{-0.5} & 0 \\ 0 & \mathbf{D}_{\mathrm{col}}^{-0.5} \end{bmatrix}$$

(where $\mathbf{D}_{\mathrm{row}}$ ($\mathbf{D}_{\mathrm{col}}$) is the diagonal matrix with diagonal entries equal to the row (column) sums of $\mathbf{W}^{(12)}$) but that matrix has the same eigenvectors as the matrix

$$\begin{bmatrix} \mathbf{D}_{\mathrm{row}}^{-0.5}\mathbf{W}^{(12)}\mathbf{D}_{\mathrm{col}}^{-1}(\mathbf{W}^{(12)})^{\mathrm{T}}\mathbf{D}_{\mathrm{row}}^{-0.5} & 0 \\ 0 & \mathbf{D}_{\mathrm{col}}^{-0.5}(\mathbf{W}^{(12)})^{\mathrm{T}}\mathbf{D}_{\mathrm{row}}^{-1}\mathbf{W}^{(12)}\mathbf{D}_{\mathrm{col}}^{-0.5} \end{bmatrix}$$

which has conjoined eigenvectors of each of the blocks $\mathbf{D}_{\mathrm{row}}^{-0.5}\mathbf{W}^{(12)}\mathbf{D}_{\mathrm{col}}^{-1}(\mathbf{W}^{(12)})^{\mathrm{T}}\mathbf{D}_{\mathrm{row}}^{-0.5}$ and $\mathbf{D}_{\mathrm{col}}^{-0.5}(\mathbf{W}^{(12)})^{\mathrm{T}}\mathbf{D}_{\mathrm{row}}^{-1}\mathbf{W}^{(12)}\mathbf{D}_{\mathrm{col}}^{-0.5}$ and these parts can be found efficiently together by computing the SVD of the matrix $\mathbf{L}_{\mathbf{W}} = \mathbf{D}_{\mathrm{row}}^{-0.5}\mathbf{W}^{(12)}\mathbf{D}_{\mathrm{col}}^{-0.5}$.[7]

[7] This trick is used in the co-clustering literature (Dhillon 2001; Zha et al. 2001), but there the affinity submatrix $\mathbf{W}$ is derived simply from the term document matrix (or equivalent) not derived as a product of affinity matrices from different views. It is possible to combine these ideas and use multiple views, each (or one) of which is a co-clustering.

References

Bickel, S., & Scheffer, T. (2004). Multi-view clustering. In *Proceedings of the IEEE international conference on data mining* (pp. 19–26).

Blaschko, M., & Lampert, C. (2008). Correlational spectral clustering. *Computer Vision and Pattern Recognition*. DOI:10.1109/CVPR.2008.4587353. CVPR 2008. IEEE Conference on pp. 1–8 (2008).

Blaschko, M. B., Lampert, C. H., & Gretton, A. (2008). Semi-supervised Laplacian regularization of kernel canonical correlation analysis. In *ECML PKDD '08: Proceedings of the 2008 European conference on machine learning and knowledge discovery in databases—Part I* (pp. 133–145). Berlin/Heidelberg: Springer.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on computational learning theory (COLT-98)* (pp. 92–100).

Cai, D., He, X., Li, Z., Ma, W., & Wen, J. (2004). Hierarchical clustering of WWW image search results using visual, textual and link information. In *Proceedings of the 12th annual ACM international conference on Multimedia* (pp. 952–959).

Charless Fowlkes Serge Belongie, F. C., & Malik, J. (2004). Spectral grouping using the Nystrom method. *IEEE Transactions Pattern Analysis and Machine Intelligence*, *26*(2), 214–225.

Chaudhuri, K., Kakade, S., Livescu, K., & Sridharan, K. (2009). Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th annual international conference on machine learning*. New York: ACM.

de Sa, V. R. (1994). Learning classification with unlabeled data. In J. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems* (Vol. 6, pp. 112–119). San Mateo: Morgan Kaufmann.

de Sa, V. R. (2005). Spectral clustering with two views. In *ICML workshop on learning with multiple views* (20–27).

de Sa, V. R., & Ballard, D. H. (1998). Category learning through multimodality sensing. *Neural Computation*, *10*(5), 1097–1117.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD* 2001 (pp. 269–274).

Golland, Y., Golland, P., Bentin, S., & Malach, R. (2008). Data-driven clustering reveals a fundamental subdivision of the human cortex into two global systems. *Neuropsychologia*, *46*(2), 540–553.

Golland, P., Golland, Y., & Malach, R. (2007). Detection of spatial activation patterns as unsupervised segmentation of fMRI Data. In *LNCS: Vol. 4791. Proceedings of MICCAI: International Conference on Medical Image Computing and Computer Assisted Intervention* (pp. 110–118). Berlin: Springer.

Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2003). *Canonical correlation analysis; an overview with application to learning methods* (Technical Report CSD-TR-03-02). Department of Computer Science, Royal Holloway, University of London.

Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2004). Canonical correlation analysis: an overview with application to learning methods. *Neural Computation*, *16*, 2639–2664.

Haxby, J., Gobbini, M., Furey, M., Ishai, A., Schouten, J., & Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, *293*(5539), 2425–2430.

Hotelling, H. (1936). Relations between two sets of variables. *Biometrika*, *28*, 321–377.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th international conference on machine learning (ICML 2003)* (pp. 290–297).

Kleine, L. L., Monnet, V., Pechoux, C., & Trubuil, A. (2008). Role of bacterial peptidase f inferred by statistical analysis and further experimental validation. *HFSP Journal*, *2*(1), 29–41.

Lai, P., & Fyfe, C. (2000). Kernel and nonlinear canonical correlation analysis. In *IJCNN '00: Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks (IJCNN'00)* (Vol. 4, p. 4614). Washington: IEEE Computer Society.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Law, M., Topchy, A., & Jain, A. (2004). Clustering with soft and groupconstraints. In *Joint IAPR international workshop on syntactical and structural pattern recognition and statistical pattern recognition* (pp. 662–670).

Law, M., Topchy, A., & Jain, A. (2005). Model-based clustering with probabilistic constraints. In *Proceedings of SIAM data mining* (pp. 641–645).

Loeff, N., Alm, C., & Forsyth, D. (2006). Discriminating image senses by clustering with multimodal features. In *Proceedings of the COLING/ACL 2006 main conference poster sessions* (pp. 547–554).

Long, B., Wu, X., Zhang, Z. M., & Yu, P. S. (2006). Unsupervised learning on *k*-partite graphs. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 317–326). New York: ACM.

Long, B., Yu, P. S., & Zhang, Z. M. (2008). A general model for multiple view unsupervised learning. In *SDM* (pp. 822–833). Philadelphia: SIAM.

Lu, Z., & Leen, T. (2005). Semi-supervised learning with penalized probabilistic clustering. *Advances in Neural Information Processing Systems*, *17*, 849–856.

Lu, Z., & Leen, T. (2007). Penalized Probabilistic Clustering. *Neural Computation*, *19*(6), 1528–1567.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (Vol. 14).

Rakotomamonjy, A., Bach, F., Grandvalet, Y., & Canu, S. (2008). SimpleMKL. *Journal of Machine Learning Research*, *9*, 2491–2521.

Shi, J., & Malik, J. (1997). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 888–905.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the eighteenth international conference on machine learning* (pp. 577–584).

Wang, Y., & Rajapakse, J. C. (2006). Contextual modeling of functional mr images with conditional random fields. *IEEE Transactions on Medical Imaging*, *25*(6), 804–812.

Woolrich, M. W., Behrens, T. E., Beckmann, C. F., & Smith, S. M. (2005). Mixture models with adaptive spatial regularization for segmentation with an application to fmri data. *IEEE Transactions on Medical Imaging*, *24*(1), 1–11.

Zha, H., Ding, C., & Gu, M. (2001). Bipartite graph partitioning and data clustering. In *CIKM ’01* (pp. 25–32).

Zhou, D., & Burges, C. J. C. (2007). Spectral clustering and transductive learning with multiple views. In *ICML ’07: Proceedings of the 24th international conference on machine learning* (pp. 1159–1166). New York: ACM. DOI:http://doi.acm.org/10.1145/1273496.1273642.