Supervised and Localized Dimensionality Reduction from Multiple Feature Representations or Kernels

Mehmet Gönen[1,2]
gonen@cis.hut.fi

Ethem Alpaydın[2]
alpaydin@boun.edu.tr

[1]Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University, School of Science and Technology

[2]Department of Computer Engineering
Boğaziçi University

Abstract

We propose a supervised and localized dimensionality reduction method that combines multiple feature representations or kernels. Each feature representation or kernel is used where it is suitable through a parametric gating model in a supervised manner for efficient dimensionality reduction and classification, and local projection matrices are learned for each feature representation or kernel. The kernel machine parameters, the local projection matrices, and the gating model parameters are optimized using an alternating optimization procedure composed of kernel machine training and gradient-descent updates. Empirical results on benchmark data sets validate the method in terms of classification accuracy, smoothness of the solution, and ease of visualization.

1 Introduction

In pattern recognition tasks, data instances can be described using different feature representations (possibly coming from different input sources) and different similarity (kernel or distance) measures can be defined over pairs of data instances. Different feature representations or similarity measures may be suitable for different cases and combining them gives us the possibility to construct a learner with higher accuracy or a more informative visualization of the data.

Suppose we are given a training data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where N is the number of training instances, $\boldsymbol{x}_i = \{\boldsymbol{x}_i^m\}_{m=1}^{P}$, P is the number of feature representations, $\boldsymbol{x}_i^m \in \mathbb{R}^{D_m}$, D_m is the dimensionality of the corresponding feature representation, and $y_i \in \{-1, +1\}$. In multiple kernel learning (MKL), the discriminant function is written as the unweighted sum of discriminant functions calculated in each feature representation (Bach et al., 2004):

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}^m) \rangle + b$$

where we can use different mapping functions, $\Phi_m(\cdot)$, on each feature representation, $\boldsymbol{w}_m$ is its corresponding weight vector, and b is the bias parameter. After eliminating $\boldsymbol{w}_m$ from the model using duality conditions, the discriminant function becomes

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \sum_{m=1}^{P} \alpha_i y_i \eta_m \underbrace{\langle \Phi_m(\boldsymbol{x}_i^m), \Phi_m(\boldsymbol{x}^m) \rangle}_{k_m(\boldsymbol{x}_i^m, \boldsymbol{x}^m)} + b$$

where the weights satisfy $\eta_m \geq 0$ and $\sum_{m=1}^{P} \eta_m = 1$.

Using a fixed combination rule (e.g., convex combination) has the disadvantage of assigning the same weight to a kernel over the whole input space. Localized multiple kernel learning (LMKL) overcomes this limitation and learns data-dependent kernel weights using a parametric gating model (Gönen and Alpaydın, 2008). LMKL rewrites the MKL discriminant function as

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}^{\mathcal{G}}|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}^m)\rangle + b$$

where $\eta_m(\cdot|\cdot)$ is a parametric gating model which assigns a weight to feature representation $\Phi_m(\cdot)$ and $\boldsymbol{x}^{\mathcal{G}}$ is the feature representation in which the gating model is learned. The resulting discriminant function becomes

$$f(\boldsymbol{x}) = \sum_{i=1}^{N}\sum_{m=1}^{P} \alpha_i y_i \eta_m(\boldsymbol{x}_i^{\mathcal{G}}|\mathbf{V}) k_m(\boldsymbol{x}_i^m, \boldsymbol{x}^m)\eta_m(\boldsymbol{x}^{\mathcal{G}}|\mathbf{V}) + b.$$

The gating model in LMKL uses a softmax gating that divides the gating space into regions:

$$\eta_m(\boldsymbol{x}^{\mathcal{G}}|\mathbf{V}) = \frac{\exp(\langle \boldsymbol{v}_m, \boldsymbol{x}^{\mathcal{G}}\rangle + v_{m0})}{\sum\limits_{h=1}^{P}\exp(\langle \boldsymbol{v}_h, \boldsymbol{x}^{\mathcal{G}}\rangle + v_{h0})} \qquad \forall m \tag{1}$$

where $\mathbf{V} = \{\boldsymbol{v}_1, v_{10}, \boldsymbol{v}_2, v_{20}, \dots, \boldsymbol{v}_P, v_{P0}\}$ is the vector of gating parameters. The feature representation used in gating, $\boldsymbol{x}^{\mathcal{G}}$, can be one of the feature representations or a concatenation of them.

Since each feature is used in a localized part of the input space, one can reduce the dimensionality in there leading to a smoother overall discriminant. Furthermore, the gating model which is fed by the concatenation of all features is very high-dimensional and can be simplified when the feature representations are redundant and/or correlated.

In this paper, we propose a supervised and localized dimensionality reduction method, named as SLDR, from multiple feature representations or kernels coupled with a kernel machine. In Section 2, we describe the method in more detail and derive the learning algorithm step by step in Appendix A. We then demonstrate its performance on benchmark data sets in Appendix B and conclude in Section 3.

2 Supervised and localized dimensionality reduction

The idea is to learn a projection matrix for each feature representation and decide the projection to use for each data instance using a gating model. So, we have P different projections for each data instance:

$$\boldsymbol{z}^m = \mathbf{W}_m^\top \boldsymbol{x}^m$$

where $\mathbf{W}_m \in \mathbb{R}^{D_m \times R_m}$ and R_m is the dimensionality of the corresponding projection space. Using these local projection matrices, we can rewrite the discriminant function as

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}^{\mathcal{G}}|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\mathbf{W}_m^\top \boldsymbol{x}^m)\rangle + b \tag{2}$$

where $\eta_m(\cdot|\cdot)$ assigns a weight to the corresponding projection space. If we use the gating model in (1), we have $P(D_{\mathcal{G}} + 1)$ parameters to learn, where $D_{\mathcal{G}}$ is the dimensionality of the gating space. For example, if we use the concatenation of all feature representations as the gating space, we may need to optimize a very large number of parameters. Instead, we also reduce the dimensionality in the gating model as follows:

$$\boldsymbol{z}^{\mathcal{G}} = \mathbf{T}^\top \boldsymbol{x}^{\mathcal{G}}$$

where $\mathbf{T} \in \mathbb{R}^{D_{\mathcal{G}} \times R_{\mathcal{G}}}$ and $R_{\mathcal{G}}$ is the dimensionality of the new projected gating space. Rewriting the gating model, we get

$$\eta_m(\boldsymbol{x}^{\mathcal{G}}|\mathbf{V}, \mathbf{T}) = \frac{\exp(\langle \boldsymbol{v}_m, \mathbf{T}^\top \boldsymbol{x}^{\mathcal{G}}\rangle + v_{m0})}{\sum\limits_{h=1}^{P}\exp(\langle \boldsymbol{v}_h, \mathbf{T}^\top \boldsymbol{x}^{\mathcal{G}}\rangle + v_{h0})} \qquad \forall m. \tag{3}$$

Integrating the new discriminant function in (2) into the canonical support vector machine (SVM) framework and minimizing the sum of squared norms of each feature representation gives us the following optimization problem:

$$\begin{aligned}
&\text{minimize} \quad \frac{1}{2}\sum_{m=1}^{P}\|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i \\
&\text{with respect to} \quad \boldsymbol{w}_m \in \mathbb{R}^{S_m}, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}_+^N, \mathbf{V} \in \mathbb{R}^{P(R_{\mathcal{G}}+1)}, \mathbf{T} \in \mathbb{R}^{D_{\mathcal{G}} \times R_{\mathcal{G}}}, \mathbf{W}_m \in \mathbb{R}^{D_m \times R_m} \\
&\text{subject to} \quad y_i\left(\sum_{m=1}^{P}\eta_m(\boldsymbol{x}_i^{\mathcal{G}}|\mathbf{V})\langle\boldsymbol{w}_m, \Phi_m(\mathbf{W}_m^\top\boldsymbol{x}_i^m)\rangle + b\right) \geq 1 - \xi_i \qquad \forall i
\end{aligned} \tag{4}$$

where S_m is the dimensionality of the mapped feature space constructed by $\Phi_m(\cdot)$. We now need to optimize the gating model parameters and the projection matrices in addition to the original SVM parameters. Unfortunately, this problem is not convex and we propose to use an alternating optimization approach (see Appendix A). In this iterative approach, a canonical SVM solver is called in the inner loop and the extra parameters can be updated using gradient-descent, projected gradient-descent, or by solving another optimization problem. Similar alternating optimization approaches have been proposed to optimize the disjoint parameter subsets of learning algorithms in a coupled manner (Chapelle et al., 2002; Pereira and Gordon, 2006; Gönen and Alpaydın, 2008; Rakotomamonjy et al., 2008; Lin et al., 2009).

Our work builds upon those of Chapelle et al. (2002) and Pereira and Gordon (2006) who combine dimensionality reduction and classifier training for a single feature representation. Lin et al. (2009) propose a dimensionality reduction method that uses multiple kernels to embed data instances from different feature spaces to a unified feature space using a graph embedding framework. Our previous work (Gönen and Alpaydın, 2010) uses this same idea of supervised and localized dimensionality reduction except that the same, single feature representation is used in all local models and gating; combining multiple feature representations is novel in this present paper.

We perform experiments on the Multiple Features (MultiFeat) digit recognition data set from the UCI Repository (details are given in Appendix B). Figure 1 shows the two-dimensional projected gating space and the two-dimensional local projection spaces obtained by SLDR without a decrease in accuracy and using fewer support vectors. The gating model effectively eliminates two of the six feature representations and divides the projected gating space into four regions.

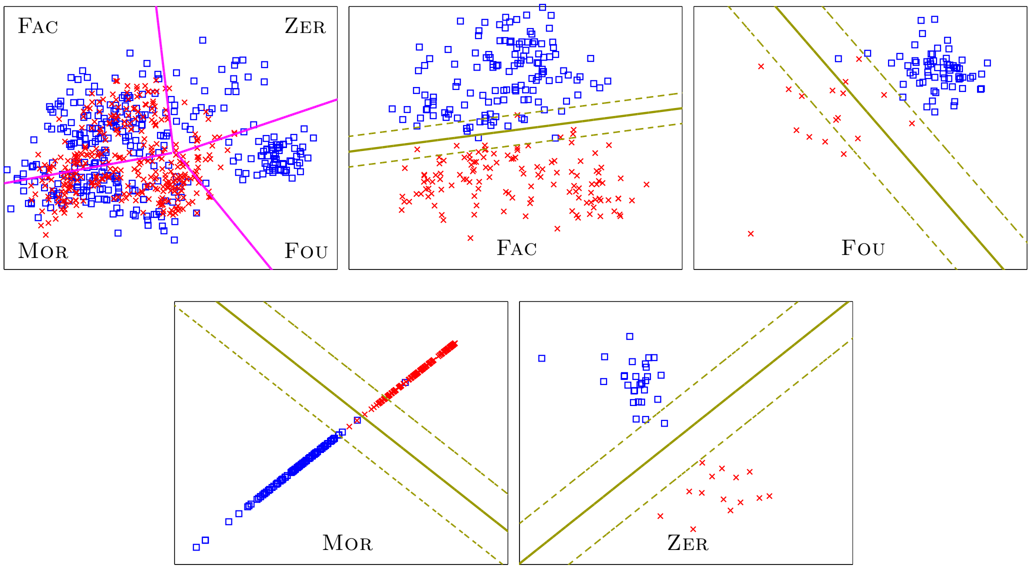

Figure 1: The projected gating space and the local projection spaces obtained by SLDR with $R_m = 2$ and $R_{\mathcal{G}} = 2$ on the MultiFeat data set

3 Conclusions

We introduce a supervised and localized dimensionality reduction method that uses multiple feature representations or kernels. The proposed method has three basic components: (a) the gating model that assigns weights to each feature representation or kernel, (b) the local projection matrices that perform dimensionality reduction separately for each feature representation or kernel, and (c) the kernel machine that performs classification using these locally obtained projections.

The training of these three components are performed in a coupled and supervised manner using an alternating optimization procedure. The gating model parameters and the local projection matrices are updated at each iteration using gradient-descent steps calculated from the objective function of the kernel machine.

The proposed algorithm is tested and compared with SVM, MKL, and LMKL on benchmark data sets for classification tasks. We achieve comparable accuracy results using significantly fewer support vectors. The reduction in support vector count is mainly due to low dimensionality of the projection spaces, which allows us to find smooth discriminants.

We use KPCA to map from kernels to features. Another possibility is to use an *empirical kernel map* (Schölkopf et al., 2004) by using the pairwise kernel values of a data instance with all the training instances as the feature vector. We also perform simulations with this approach and see that KPCA works better. The empirical kernel map is very high-dimensional and does not lend itself well to dimensionality reduction.

Like other MKL methods, SLDR also provides knowledge extraction through the weights assigned to feature representations or kernels. A figure like Figure 1 is very informative: We can understand the most informative feature representation or kernel by looking at the gating values and we can conclude that the feature representations or kernels not used by the gating model do not carry useful information at all. Feature representations or kernels never used by the gating can be completely eliminated in order to reduce the cost of data acquisition and processing.

Acknowledgments

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

References

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

M. Gönen and E. Alpaydın. Localized multiple kernel learning. In *ICML*, 2008.

M. Gönen and E. Alpaydın. Supervised learning of local projection kernels. *Neurocomputing*, 73(10–12): 1694–1703, 2010.

G. R. G. Lanckriet, T. de Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh. Dimensionality reduction for data in multiple feature representations. In *Advances in Neural Processing Systems 21*, 2009.

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

F. Pereira and G. Gordon. The support vector decomposition machine. In *ICML*, 2006.

A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *JMLR*, 9:2491–2521, 2008.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Cambridge, MA, 2002.

B. Schölkopf, K. Tsuda, and J.-P. Vert. A primer on kernel methods. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*, chapter 2. The MIT Press, 2004.

A Learning algorithm

A.1 Learning the classifier

The optimization problem (4) becomes convex for given gating model parameters and projection matrices. In this case, we can obtain the dual problem as

$$\begin{aligned}
\text{maximize} \quad & J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^{P}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_{+}^{N} \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned} \tag{5}$$

where the kernel function is defined as

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) k_m(\mathbf{W}_m^\top \boldsymbol{x}_i^m, \mathbf{W}_m^\top \boldsymbol{x}_j^m) \eta_m(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T})$$

and the decision function becomes

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

This dual optimization problem is exactly equivalent to the canonical SVM dual problem and we can use any SVM solver to find the support vector coefficients.

A.2 Learning the gating model

If we fix the support vector coefficients and the projection matrices, we can update the gating model parameters using gradient-descent. The gradients of the objective function in (5) with respect to the gating model parameters are given as

$$\begin{aligned}
\frac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^{P})}{\partial \boldsymbol{v}_m} &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{h=1}^{P} \Upsilon_{ij} \eta_h(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) k_m(\mathbf{W}_h^\top \boldsymbol{x}_i^h, \mathbf{W}_h^\top \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) \\
&\quad \mathbf{T}^\top (\boldsymbol{x}_i^{\mathcal{G}} (\delta_m^h - \eta_m(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T})) + \boldsymbol{x}_j^{\mathcal{G}} (\delta_m^h - \eta_m(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T}))) \\
\frac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^{P})}{\partial v_{m0}} &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{h=1}^{P} \Upsilon_{ij} \eta_h(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) k_m(\mathbf{W}_h^\top \boldsymbol{x}_i^h, \mathbf{W}_h^\top \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) \\
&\quad (\delta_m^h - \eta_m(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) + \delta_m^h - \eta_m(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T}))
\end{aligned}$$

where $\Upsilon_{ij} = \alpha_i \alpha_j y_i y_j$, and δ_m^h is 1 if $m = h$ and 0 otherwise.

A.3 Learning the projection matrices

We can also update the projection matrices using gradient-descent. For given support vector coefficients and gating model parameters, the gradient of the objective function in (5) with respect to the entries of $\mathbf{W}_m$ matrices is given as:

$$\frac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^{P})}{\partial \mathbf{W}_m[k, l]} = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \Upsilon_{ij} \eta_m(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) \frac{\partial k_m(\mathbf{W}_m^\top \boldsymbol{x}_i^m, \mathbf{W}_m^\top \boldsymbol{x}_j^m)}{\partial \mathbf{W}_m[k, l]} \eta_m(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T})$$

where $[\cdot, \cdot]$ indexes the entries of a matrix. Three commonly used kernel functions, linear kernel, polynomial kernel, and Gaussian kernel, can be represented in terms of projection matrices as follows:

$$\begin{aligned}
k_m^L(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m) &= \langle \mathbf{W}_m^\top \boldsymbol{x}_i^m, \mathbf{W}_m^\top \boldsymbol{x}_j^m \rangle \\
k_m^P(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m) &= (\langle \mathbf{W}_m^\top \boldsymbol{x}_i^m, \mathbf{W}_m^\top \boldsymbol{x}_j^m \rangle + 1)^q \\
k_m^G(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m) &= \exp(-\|\mathbf{W}_m^\top \boldsymbol{x}_i^m - \mathbf{W}_m^\top \boldsymbol{x}_i^m\|_2^2 / s^2).
\end{aligned}$$

The derivatives of the kernels with respect to the entries of the projection matrices are given as:

$$\begin{aligned}
\frac{\partial k_m^L(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m)}{\partial \mathbf{W}_m[k,l]} &= \boldsymbol{x}_i^m[k]\boldsymbol{z}_j^m[l] + \boldsymbol{z}_i^m[l]\boldsymbol{x}_j^m[k] \\
\frac{\partial k_m^P(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m)}{\partial \mathbf{W}_m[k,l]} &= (\boldsymbol{x}_i^m[k]\boldsymbol{z}_j^m[l] + \boldsymbol{z}_i^m[l]\boldsymbol{x}_j^m[k]) q (\langle \boldsymbol{z}_i^m, \boldsymbol{z}_j^m \rangle + 1)^{q-1} \\
\frac{\partial k_m^G(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m)}{\partial \mathbf{W}_m[k,l]} &= -2(\boldsymbol{x}_i^m[k] - \boldsymbol{x}_j^m[k])(\boldsymbol{z}_i^m[l] - \boldsymbol{z}_j^m[l]) k_m^G(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m)/s^2
\end{aligned}$$

where $[\cdot]$ indexes the entries of a vector. We also need to calculate the gradient of the objective function in (5) with respect to the entries of $\mathbf{T}$:

$$\begin{aligned}
\frac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{T}[e,f]} = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \sum_{h=1}^P \sum_{m=1}^P & \Upsilon_{ij} \eta_h(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) k_h(\mathbf{W}_h^\top \boldsymbol{x}_i^h, \mathbf{W}_h^\top \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T}) \\
& \boldsymbol{v}_m[f](\boldsymbol{x}_i^{\mathcal{G}}[e](\delta_m^h - \eta_m(\boldsymbol{x}_i^{\mathcal{G}} | \mathbf{V}, \mathbf{T})) + \boldsymbol{x}_j^{\mathcal{G}}[e](\delta_m^h - \eta_m(\boldsymbol{x}_j^{\mathcal{G}} | \mathbf{V}, \mathbf{T}))).
\end{aligned}$$

Both $\mathbf{T}$ and $\{\mathbf{W}_m\}_{m=1}^P$ matrices are performing projections and they would be members of the Stiefel manifold (i.e., $\mathbf{T}^\top \mathbf{T} = \mathbf{I}$ and $\mathbf{W}_m^\top \mathbf{W}_m = \mathbf{I}$) in order to obtain meaningful projections. We can achieve this by projecting these matrices to the Stiefel manifold after initialization and gradient update. For example, the singular value decomposition can be used to make the columns of these matrices mutually orthogonal and have unit norm.

A.4 Complete algorithm

The complete algorithm of our proposed method called supervised and localized dimensionality reduction (SLDR) is summarized in Algorithm 1. The gating model parameters and the projection matrices are initialized to random numbers at the first iteration. $\Delta^{(t)}$, $\nu^{(t)}$, and $\mu^{(t)}$ are the step sizes of the corresponding gradient-descent updates. These step sizes can be taken as constants or can be optimized using a line search method like Armijo's rule. Line search increases the time complexity of each iteration due to additional calls to the canonical SVM solver but the algorithm converges in fewer iterations. The three gradient formulations depend only on the current support vectors (i.e., if $\alpha_i = 0$, $\boldsymbol{x}_i$ does not contribute to the gradients) and have ignorable time complexity compared to solving the canonical SVM problems with locally combined kernel matrix $\mathbf{K}_\eta = \{k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1}^N$. We can determine the convergence of the algorithm by monitoring the change in the objective function value.

B Experiments

We evaluate the proposed method on benchmark data sets in terms of visualization and classification performances. We implement the main body of our algorithm in MATLAB and solve the optimization problems with MOSEK[1] optimization software. We stop the algorithm when the objective value of the current iteration is not less than 0.999 times the objective function value of the previous iteration (i.e., when we achieve less than one-thousandth improvement in the objective function value).

B.1 Dimensionality Reduction from Multiple Feature Representations

We perform experiments on the Multiple Features (MULTIFEAT) digit recognition data set from the UCI Machine Learning Repository[2], composed of six different feature representations for 2000

[1] Available from `http://www.mosek.com`.

[2] Available from `http://archive.ics.uci.edu/ml`.

Algorithm 1 Supervised and Localized Dimensionality Reduction (SLDR)

```
Initialize $\{\mathbf{V}^{(0)}, \mathbf{T}^{(0)}, \mathbf{W}_{1:P}^{(0)}\}$ to small random numbers
repeat
    Calculate $\mathbf{K}_\eta$ using $\{\mathbf{V}^{(t)}, \mathbf{T}^{(t)}, \mathbf{W}_{1:P}^{(t)}\}$
    Solve canonical SVM with $\mathbf{K}_\eta$
    $\mathbf{W}_m^{(t+1)} \Leftarrow \mathbf{W}_m^{(t)} - \Delta^{(t)} \dfrac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{W}_m} \qquad \forall m$
    Calculate $\mathbf{K}_\eta$ using $\{\mathbf{V}^{(t)}, \mathbf{T}^{(t)}, \mathbf{W}_{1:P}^{(t+1)}\}$
    Solve canonical SVM with $\mathbf{K}_\eta$
    $\mathbf{T}^{(t+1)} \Leftarrow \mathbf{T}^{(t)} - \nu^{(t)} \dfrac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{T}}$
    Calculate $\mathbf{K}_\eta$ using $\{\mathbf{V}^{(t)}, \mathbf{T}^{(t+1)}, \mathbf{W}_{1:P}^{(t+1)}\}$
    Solve canonical SVM with $\mathbf{K}_\eta$
    $\mathbf{V}^{(t+1)} \Leftarrow \mathbf{V}^{(t)} - \mu^{(t)} \dfrac{\partial J(\mathbf{V}, \mathbf{T}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{V}}$
until convergence
```

handwritten numerals. The properties of these feature representations are summarized in Table 1. We generate a binary classification problem from the MultiFeat data set by separating small ('0' - '4') digits from large ('5' - '9') digits.

Table 1: Multiple feature representations in the MultiFeat data set.

Name	Dim.	Data Source
Fac	216	Profile correlations
Fou	76	Fourier coefficients of the shapes
Kar	64	Karhunen-Loève coefficients
Mor	6	Morphological features
Pix	240	Pixel averages in 2×3 windows
Zer	47	Zernike moments

A random one-third of the dataset is reserved as the test set and the remaining two-thirds is resampled using 5×2 cross-validation to generate ten training and validation sets, with stratification. The validation sets of all folds are used to optimize C by trying values 1, 10, and 100. The best configuration (the one that has the highest average accuracy on the validation folds) is used to train the final classifiers on the training folds and their performance is measured over the test set. We have ten test set results, and we report their averages and standard deviations.

We compare SVM, MKL, LMKL, and SLDR in terms of classification performance and model complexity (i.e., stored support vector percentage). We train SVMs with linear kernels calculated on each feature representation singly and report the results of the one with the highest average validation accuracy. We also train an SVM with linear kernel calculated on the concatenation of all feature representations, which will be referred as All. MKL is the original formulation of Bach et al. (2004). LMKL combines linear kernels calculated on each feature representation and uses the concatenation of all feature representations in the gating model without any dimensionality reduction. Our proposed SLDR combines linear kernels calculated on each projection space ($R_m \in \{1, 2, 3, 4, 5\}$) and uses the projected gating space ($R_\mathcal{G} \in \{2, 4, 6, 8, 10\}$) obtained from the concatenation of all feature representations.

Table 2 summarizes the classification results on the MultiFeat data set. SVM (Fac) is the most accurate classifier with a single feature representation. We see that integrating multiple feature representations obtains higher average test accuracy than SVM (Fac). LMKL is accurate as MKL storing half as many support vectors. SLDR is also accurate and stores much fewer support vectors. SLDR achieves statistically comparable accuracy result compared with other integration methods and stores significantly fewer support vectors.

Table 2: Classification results on the MULTIFEAT data set.

Method	Test Accuracy	Support Vector
SVM (FAC)	94.97±0.87	17.93±0.91
SVM (ALL)	97.69±0.44	23.34±1.13
MKL	97.40±0.37	32.59±0.82
LMKL	97.73±0.57	14.35±1.43
SLDR⋆	97.09±0.52	2.89±0.67

⋆: $R_m = 2$ and $R_{\mathcal{G}} = 10$

The average test accuracies and support vector percentages obtained with changing R_m and $R_{\mathcal{G}}$ are shown in Figure 2. We see that SLDR consistently finds accurate solutions storing very few support vectors.

Figure 1 shows the projected gating space and the local projection spaces obtained by SLDR with $R_m = 2$ and $R_{\mathcal{G}} = 2$. The projected gating space shows all of the training instances and how instances are divided among feature spaces. Each training instance is drawn in the projection space where it has the maximum gating value. The gating model effectively eliminates two of the feature representations (KAR and PIX) and divides the projected gating space into four regions. In each region, a specific feature representation is used and a local projection matrix is learned. The separating hyperplanes and the margin boundaries obtained in the local projection spaces for each region are also given in Figure 1. Even when we use one-dimensional projections in each region, we get an accurate solution as shown in Figure 3; SLDR again uses only four out of the six feature representations and eliminates FOU and PIX.

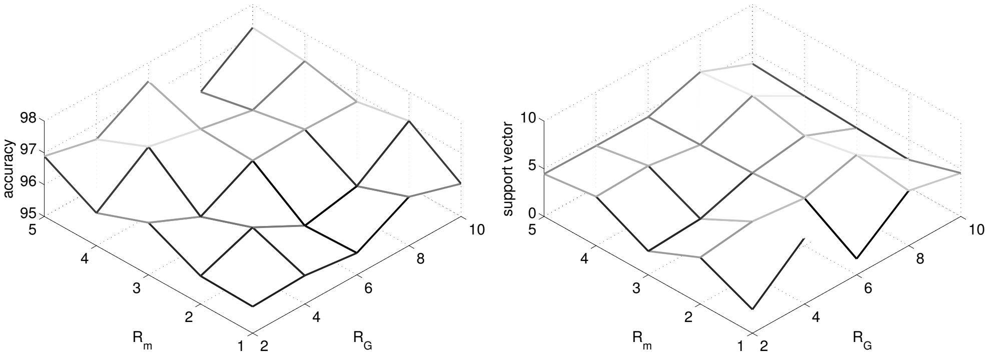

Figure 2: The average test accuracies and support vector percentages obtained by SLDR with changing R_m and $R_{\mathcal{G}}$ on the MULTIFEAT data set

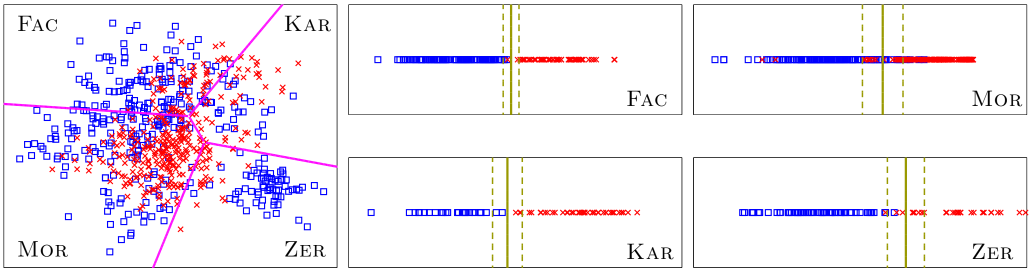

Figure 3: The projected gating space and the local projection spaces obtained by SLDR with $R_m = 1$ and $R_{\mathcal{G}} = 2$ on the MULTIFEAT data set

B.2 Dimensionality Reduction from Multiple Kernels

In some applications, we may not have the feature representations but the kernel values (similarity measures) defined on pairs of data instances. In order to learn the gating model and the projection matrices, we need data instances to be in vector format. For this, we apply kernel principle component analysis (KPCA) (Schölkopf and Smola, 2002) to each kernel function and the resulting projections are used as feature representations to fed into SLDR.

We perform protein location prediction experiment on the MIPS Comprehensive Yeast Genome Database (CYGD) (Mewes et al., 2000). CYGD assigns subcellular locations for 1150 proteins according to whether they participate in the ribosome or not. We use the seven kernels given in Lanckriet et al. (2004) and apply KPCA to each of them by following the first approach. The eigenvectors that corresponds to the nonzero eigenvalues are preserved and the resulting projections are taken as the feature representations. We use the same experimental procedure that we use for the MULTIFEAT data set.

Table 3: Multiple kernels in the YEAST data set.

Name	Similarity	Data source
$\mathbf{K}_{\text{B}}$	BLAST	Protein sequences
$\mathbf{K}_{\text{D}}$	Diffusion kernel	Protein interactions
$\mathbf{K}_{\text{E}}$	Gaussian kernel	Gene expression
$\mathbf{K}_{\text{FFT}}$	FFT	Hydropathy profile
$\mathbf{K}_{\text{LI}}$	Linear kernel	Protein interactions
$\mathbf{K}_{\text{PFAM}}$	PFAM HMM	Protein sequences
$\mathbf{K}_{\text{SW}}$	Smith-Waterman	Protein sequences

Table 4 lists classification results. We see that integrating multiple feature representations obtained from different kernels does not improve the average test accuracy compared with the best SVM result using the kernel $\mathbf{K}_{\text{E}}$. We see that unlike other methods (MKL and LMKL), SLDR obtains lower (but not significantly) average test accuracy compared with the best SVM result and stores significantly fewer support vectors.

Table 4: Classification results on the YEAST data set.

Method	Test Accuracy	Support Vector
SVM ($\mathbf{K}_{\text{E}}$)	98.80±0.25	100.00± 0.00
SVM (ALL)	92.08±0.59	100.00± 0.00
MKL	89.92±3.95	100.00± 0.00
LMKL	76.64±1.77	95.30± 9.51
SLDR⋆	95.31±2.50	2.04± 1.71

⋆: $R_m = 5$ and $R_{\mathcal{G}} = 2$

We also perform protein fold recognition experiments using multiple feature representations and kernels. We use the protein fold recognition data set, which we call PROTEIN, from the UCSD Multiple Kernel Learning Repository[3]. We construct a binary classification problem by combining the major structural classes $\{\alpha, bta\}$ into one class and $\{\alpha/bta, \alpha+bta\}$ into another class. The data set consists of 10 different feature representations (COMPOSITION, HYDROPHOBICITY, L1, L14, L30, L4, POLARITY, POLARIZABILITY, SECONDARY, VOLUME) and two kernels (SWBLOSUM, SW-PAM). Due to small size of this dataset, we use 10-fold cross-validation instead of 5×2 cross-validation and use 90 per cent of the learning set for training at each fold. We calculate linear kernel on the feature representations to obtain 12 kernels and apply KPCA to each of them in order to obtain feature representations to fed into SLDR.

[3] Available from `http://mkl.ucsd.edu`.

Table 5 summarizes the classification results on the PROTEIN data set. We see that integrating multiple feature representations obtains higher average test accuracy than the best SVM result with the single feature representation L14. SLDR achieves statistically comparable accuracy result compared to the best integration method MKL and stores significantly fewer support vectors. Similar to what we see on the YEAST data set, LMKL, unlike SLDR, is affected by the high dimensionality of the gating space.

Table 5: Classification results on the PROTEIN data set.

Method	Test Accuracy	Support Vector
SVM (L14)	71.44±1.12	51.27±1.82
SVM (ALL)	81.57±0.85	97.93±0.56
MKL	84.05±0.87	83.96±2.13
LMKL	75.77±2.23	40.41±1.73
SLDR$\star$	82.17±1.93	4.18±2.05

$\star$: $R_m = 5$ and $R_{\mathcal{G}} = 6$

If we learn the gating model in a two-dimensional projected gating space and use one-dimensional projections in each region, we obtain the projection spaces shown in Figure 4. SLDR uses only three out of the 12 feature representations, namely L30, SWBLOSUM, and VOLUME, and there, one-dimensional discriminants suffice.

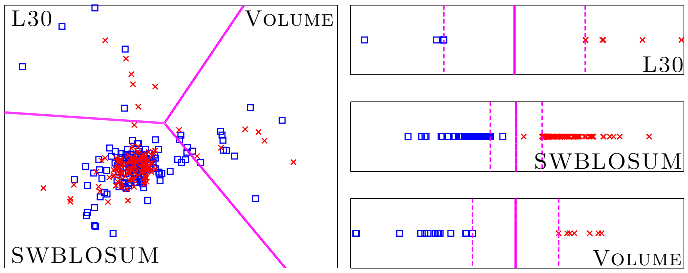

Figure 4: The projected gating space and the local projection spaces obtained by SLDR with $R_m = 1$ and $R_{\mathcal{G}} = 2$ on the PROTEIN data set