Support Vector Machine Framework for Multi-View Metric Learning

Riikka Huusari, Hachem Kadri, Cécile Capponi

Support Vector Machine Framework for Multi-View Metric Learning

Riikka Huusari [1] & Hachem Kadri [1] & Cécile Capponi [1]

[1] *LIS, CNRS, Aix-Marseille Univ., 39 rue Joliot-Curie F-13453 Marseille*
email: {firstname.lastname}@lis-lab.fr

Résumé. Cet article s'intéresse à l'apprentissage multivue par des méthodes à noyaux et d'apprentissage de métriques. Dans ce cadre, nous considérons MVML (multi-view metric learning), une méthode récemment développée, et nous proposons un algorithme basé sur les séparateurs à vaste marge (SVM) qui apprend conjointement un classifieur et des métriques entre les vues permettant ainsi de tenir compte des caractéristiques multi-vues du problème d'apprentissage. Des expérimentations sur données réelles ont été réalisées afin d'évaluer les performances de l'algorithme proposé.

Mots-clés. apprentissage de métriques multi-vues, SVM, méthodes à noyaux

Abstract. In this article we tackle the supervised multi-view learning problem with kernel methods and metric learning. In this context we consider a recently developed multi-view metric learning (MVML) framework, and propose a SVM-based algorithm that jointly learns the classifier and metrics between views. These metrics permit taking into account the multi-view characteristics of the learning problem. Experiments on real data were performed to evaluate the performance of the proposed algorithm.

Keywords. multi-view learning, metric learning, SVM, kernel methods

1 Introduction

Multi-view learning refers to the learning framework where each data example is observed under several views, e.g. a bird might be represented by both image and sound. In this paper we consider a supervised multi-view learning problem in context of classification, and thus denote our data to be $\{\mathbf{x}_i, y_i\}_{i=1}^n$ on $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y}$ is $\{-1, 1\}$ and $\mathbf{x}_i = (\mathbf{x}_i^1, \ldots, \mathbf{x}_i^v)$. The views in a multi-view problem are often correlated, complementary or redundant, or even contradictory. Thus, using all the views in a learning problem is expected to be more beneficial than if individual views were used independently. One of the first approaches in multi-view learning was co-regularization (Blum and Mitchell, 1998), where view agreement was used in context of semi-supervised learning. There are many kernel-based approaches in multi-view learning, of these the simplest and most known is the multiple kernel learning (MKL) framework (Xu et al., 2013), where a simple linear combination of the kernel matrices from each view is learned.

More recently, vector-valued (or operator-valued) reproducing kernel Hilbert spaces (RKHSs) have been introduced in the context of multi-view learning (Minh et al., 2016;

Kadri et al., 2013). Unlike MKL, using vector-valued kernels allows modeling both within-view and between-view interactions. Even though vector-valued kernels offer a good way to model the view interactions, a central question that is already asked with scalar-valued kernels is even more prominent in this context: how to construct the kernels? There exist many schemes for learning separable vector-valued kernels (Dinuzzo et al., 2011), however this class of kernels is restrictive. In the multi-view metric learning (MVML) introduced by Huusari et al. (2018), a general vector-valued kernel matrix is learned jointly with the classification/regression problem by introducing a metric matrix operating between the views to the vector-valued kernel. The MVML was introduced with squared loss function. It is possible to perform classification tasks with this loss, but a more suitable choice is to use large margin classifiers or SVMs (Cortes and Vapnik, 1995).

2 Multi-view learning with matrix-valued kernels

Vector-valued RKHS were introduced by Micchelli and Pontil (2005) as a way to extend kernel machines from scalar to vector outputs. As the name states, the functions in vector-valued RKHS $\mathcal{H}$ output vectors, that is, the function $\mathbf{z} \mapsto K(\mathbf{x}, \mathbf{z})\mathbf{q}$ belongs to $\mathcal{H}$ when $\mathbf{z}, \mathbf{x} \in \mathcal{X}$, $\mathbf{q} \in \mathbb{R}^v$ and the kernel function is positive definite. Thus using the vector-valued RKHS theory is especially useful in multi-task setting, where the outputs are the multiple labels (Evgeniou et al., 2005). This framework is also suitable for multi-view learning, where the outputs are combined after the learning step to give the final prediction (Kadri et al., 2013), or then in a setting where a combination operator for combining the vector into one label is either present in optimization problem, or learned with it (Minh et al., 2016; Huusari et al., 2018).

As we are dealing with vector-valued functions, the kernel function between two data samples outputs a matrix; in our case kernel is a function $K(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{v \times v}$, where v is the number of views. Intuitively, instead of one similarity value associated with the two samples, our kernel outputs one value for every two-view combination.

A major theorem in kernel framework is the representer theorem. Similarly to the scalar-valued case, the solution to a general learning problem with matrix-valued kernels can be stated as $f(\mathbf{x}) = \sum_i K(\mathbf{x}_i, \mathbf{x})\mathbf{c}_i$, where $\mathbf{c}_i \in \mathbb{R}^v$ (see for example Micchelli and Pontil (2005) for details). Thus, by adding a combination operator $\mathcal{W}$ for combining the outputs of the decision function f, we can write our general multi-view vector-valued optimization problem as

$$\arg\min_f \sum_{i=1}^{n} V\left(y_i, \mathcal{W}\left(f(\mathbf{x}_i)\right)\right) + \lambda \|f\|_{\mathcal{H}}^2. \tag{1}$$

$$= \arg\min_{\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{R}^v} \sum_{i=1}^{n} V\left(y_i, \mathcal{W}\left(\sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j)\mathbf{c}_j\right)\right) + \lambda \sum_{i,j=1}^{n} \langle \mathbf{c}_i, K(\mathbf{x}_i, \mathbf{x}_j)\mathbf{c}_j \rangle. \tag{2}$$

There are many classes of matrix-valued kernels, most common are separable and transformable (Alvarez et al., 2012). Another class of vector-valued kernels considered in Huusari et al. (2018) is written as

$$K(\mathbf{x}_i, \mathbf{x}_j)_{lm} = \left\langle \Phi_l(\mathbf{x}_i^l), C_{\mathcal{X}_l\mathcal{X}_m} \Phi_m(\mathbf{x}_j^m) \right\rangle, \tag{3}$$

where Φ_l (resp. Φ_m) is the feature map associated with the scalar-valued kernel k_l (resp. k_m) defined on the view l (resp. m). MVML framework assumes that the operator $C_{\mathcal{X}_l\mathcal{X}_m} : \mathcal{H}_m \to \mathcal{H}_l$ can be written as $C_{\mathcal{X}_l\mathcal{X}_m} = \mathbf{\Phi}_l \mathbf{A}_{lm} \mathbf{\Phi}_m^T$ where $\mathbf{\Phi}_s = (\Phi_s(\mathbf{x}_1), ..., \Phi_s(\mathbf{x}_n))$ with $s = l, m$. Thus the kernel matrix is $\mathbf{K} = \mathbf{HAH}$, where $\mathbf{H} = \text{blockdiag}(\mathbf{K}_1, \cdots, \mathbf{K}_v)$,[1] and the matrix $\mathbf{A} = (\mathbf{A}_{lm})_{l,m=1}^v \in \mathbb{R}^{nv \times nv}$ encodes pairwise similarities between the views.

Substituting the MVML kernel to the optimization problem (2) and adding regularization for metric matrix, we get a general MVML optimization problem

$$\min_{\mathbf{A},\mathbf{c}} \sum_{i=1}^{n} V(y_i, \mathbf{w}^T \mathbf{H}_{\mathbf{x}_i} \mathbf{AHc}) + \lambda \langle \mathbf{c}, \mathbf{HAHc} \rangle + \eta \|\mathbf{A}\|_F^2, \tag{4}$$

where $\mathbf{H}_{\mathbf{x}_i} \in \mathbb{R}^{v \times vn}$ consists of the rows in $\mathbf{H}$ that correspond to $\mathbf{x}_i$, and $\mathbf{w}$ gives a linear combination over the views. With the change of variables $\mathbf{g} = \mathbf{AHc}$ we can write

$$\min_{\mathbf{A},\mathbf{g}} \sum_{i=1}^{n} V(y_i, \mathbf{w}^T \mathbf{H}_{\mathbf{x}_i} \mathbf{g}) + \lambda \langle \mathbf{g}, \mathbf{A}^\dagger \mathbf{g} \rangle + \eta \|\mathbf{A}\|_F^2. \tag{5}$$

Figure 1 illustrates the idea behind MVML framework. There MVML is compared to MKL that considers only within-view dependencies, and to output kernel learning (OKL) (Dinuzzo et al., 2011) where separable kernels are learnt. MVML is the only method giving linear separation of the two classes. This means that it groups the data points into groups based on their class, not view, and thus is able to construct a good approximation of the initial data transformations by which we generated the second view.

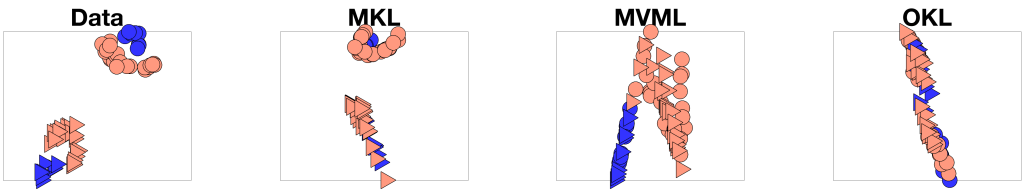

Figure 1: Simple two-view dataset and its transformations - left: original data, left middle: MKL, right middle: MVML, and right: OKL transformation. Only MVML shows a linear separation of classes (blue/pale red) of the views (circles/triangles).

[1] Given a set of $n \times n$ matrices $\mathbf{K}_1, \cdots, \mathbf{K}_v$, $\mathbf{H} = \text{blockdiag}(\mathbf{K}_1, \cdots, \mathbf{K}_v)$ is the block diagonal matrix satisfying $\mathbf{H}_{l,l} = \mathbf{K}_l, \forall l = 1, \ldots, v$.

Algorithm 1 Multi-View Metric Learning with SVM loss

Initialize $\mathbf{A} \succ 0$ and $\mathbf{w}$
while not converged **do**
  Solve for Lagrangian multipliers α_i in Equation 7; update $\mathbf{g}$ via Equation 8
  Solve for $\mathbf{A}$ as in original MVML
return $\mathbf{A}$, $\mathbf{g}$, $\mathbf{w}$

3 Algorithm: MVML-SVM

In the previous work the MVML optimization problem (6) was solved with squared loss function. As it is a general loss, the MVML could be applied in both regression and classification problems. However it is not optimal for many classification problems, and thus here we introduce MVML with hinge loss. This gives us the optimization problem

$$\min_{A,g} \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^T \mathbf{H}_{x_i} \mathbf{g}) + \lambda \langle \mathbf{g}, \mathbf{A}^\dagger \mathbf{g} \rangle + \eta \|\mathbf{A}\|_F^2. \tag{6}$$

We see straight away that the solution for metric matrix $\mathbf{A}$ is exactly the same than for the squared loss MVML, as the matrix is only present (directly) in the two regularizers. Indeed, with any loss function in our framework, and with this change of variables, we can solve for $\mathbf{A}$ always the same way. This means that we can also use the group-sparse formulation of the regularization term as in Huusari et al. (2018).

To solve $\mathbf{g}$ we firstly introduce the slack variables ξ_i to the optimization problem. Solving this requires then writing the dual problem with Lagrangian multipliers, and then solving for the multipliers α_i from

$$\max_{\alpha_i} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2\lambda} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{w}^T \mathbf{H}_{\mathbf{x}_i} \mathbf{A} \mathbf{H}_{\mathbf{x}_j}^T \mathbf{w}, \qquad s.t.\ 0 \leq \alpha_i \leq \frac{1}{n}, \tag{7}$$

where we have set parameter $C = \frac{1}{n}$. Having the α_i we can calculate

$$\mathbf{g} = \frac{1}{2\lambda} \mathbf{A} \sum_{i=1}^{l} \alpha_i y_i \mathbf{H}_{\mathbf{x}_i}^T \mathbf{w}. \tag{8}$$

In the end we have an iterative algorithm calculating alternately $\mathbf{g}$ and $\mathbf{A}$ (Algorithm 1). For computational efficiency, a Nyström approximation (Williams and Seeger, 2001) on all one-view kernels is applied.

The Rademacher bound proved in Huusari et al. (2018) can be used with known results (Mohri et al., 2012, chapter 8) to obtain a generalization bound for MVML-SVM.

Table 1: Classification accuracies ± standard deviation. The number after the dataset indicates the level of Nyström approximation on the kernels. The results for efSVM classification for Flower17-dataset are missing as only similarity matrices for each view were provided. Last column reports the best result obtained when using only one view.

METHOD	**MVMLSVM**	**MVMLSVMsp**	**MVMLSVMCov**	**MVMLSVM_I**	**OKL**	**MLKR**
Flower17 (6%)	74.66 ± 2.04	74.93 ± 2.15	75.64 ± 2.38	**76.96 ± 2.35**	68.73 ± 1.95	63.82 ± 2.51
Flower17 (24%)	77.43 ± 1.39	74.75 ± 2.14	77.79 ± 1.70	78.41 ± 1.75	76.76 ± 1.62	65.44 ± 1.36
uWaveG. (6%)	93.13 ± 0.18	**93.29 ± 0.15**	92.37 ± 0.22	91.63 ± 0.33	70.09 ± 1.07	71.09 ± 0.94
uWaveG. (24%)	**93.32 ± 0.04**	92.03 ± 0.04	91.15 ± 0.14	92.30 ± 0.02	76.65 ± 0.33	86.38 ± 0.31

METHOD	**lpMKL**	**MUMBO**	**efSVM**	**lfSVM**	**1 view SVM**
Flower17 (6%)	75.54 ± 2.61	75.27 ± 2.32	-	15.32 ± 1.94	11.59 ± 1.54
Flower17 (24%)	**78.75 ± 1.58**	76.47 ± 1.42	-	38.24 ± 2.31	22.79 ± 0.79
uWaveG. (6%)	92.34 ± 0.18	90.08 ± 0.42	80.00 ± 0.74	71.24 ± 0.41	56.54 ± 0.38
uWaveG. (24%)	92.85 ± 0.13	90.68 ± 0.33	84.07 ± 0.23	72.99 ± 0.06	58.01 ± 0.05

4 Experiments

We performed experiments with two multi-view classification datasets: Flower17[2] (7 views, 17 classes, 80 samples per class) and uWaveGesture[3] (3 views, 8 classes, 896/3582 data samples for training/testing). Hyperparameters are cross-validated, and kernels are Gaussian, with parameter σ as mean of the distances. Multi-class classification is done with one-vs-all scheme (except for MUMBO). The results are displayed in Table 1, where MVML-SVM is compared with standard SVMs on individual views and early- and late fusion (meaning data concatenation and combining results from individual classifiers). OKL (Dinuzzo et al., 2011) is a kernel learning method for separable kernels, and MLKR (Weinberger and Tesauro, 2007) is an algorithm for metric learnig in kernel setting. lpMKL is the MKL algorithm introduced in Kloft et al. (2011). MUMBO is a multi-class boosting based multi-view algorithm which is intended to reinforce the cooperation among views (Koço and Capponi, 2011). For our MVML we have two ways of learning the metric matrix, full and block-sparse, and two fixed choices (see Huusari et al. (2018) for details). Our MVML-SVM obtains better accuracies than most of the other methods.

5 Conclusion

We have updated the multi-view metric learning framework that learns a regression solution and a metric between views, to a classification setting within SVM framework. Our experiments on real datasets validate the approach.

[2]http://www.robots.ox.ac.uk/~vgg/data/flowers/17.

[3]http://www.cs.ucr.edu/~eamonn/time_series_data.

References

[1] Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. "Kernels for vector-valued functions: A review". In: *Foundations and Trends® in Machine Learning* 4.3 (2012), pp. 195–266.

[2] Avrim Blum and Tom Mitchell. "Combining labeled and unlabeled data with co-training". In: *Proceedings of COLT 1998*. ACM, 1998, pp. 92–100.

[3] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[4] Francesco Dinuzzo, Cheng S. Ong, Gianluigi Pillonetto, and Peter V. Gehler. "Learning output kernels with block coordinate descent". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 49–56.

[5] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. "Learning multiple tasks with kernel methods". In: *JMLR* 6.Apr (2005), pp. 615–637.

[6] Riikka Huusari, Hachem Kadri, and Cécile Capponi. "Multi-view Metric Learning in Vector-valued Kernel Spaces". In: *AISTATS*. 2018.

[7] Hachem Kadri, Stéphane Ayache, Cécile Capponi, Sokol Koço, François-Xavier Dupé, and Emilie Morvant. "The multi-task learning view of multimodal data". In: *Asian Conference on Machine Learning*. 2013, pp. 261–276.

[8] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. "Lp-norm multiple kernel learning". In: *JMLR* 12.Mar (2011), pp. 953–997.

[9] Sokol Koço and Cécile Capponi. "A Boosting Approach to Multiview Classification with Cooperation". In: *Proceedings of ECML PKDD'11*. 2011, pp. 209–228.

[10] Charles A. Micchelli and Massimiliano Pontil. "On learning vector-valued functions". In: *Neural computation* 17.1 (2005), pp. 177–204.

[11] Ha Quang Minh, Loris Bazzani, and Vittorio Murino. "A unifying framework in vector-valued reproducing kernel Hilbert spaces for manifold regularization and co-regularized multi-view learning". In: *JMLR* 17.25 (2016), pp. 1–72.

[12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[13] Kilian Q. Weinberger and Gerald Tesauro. "Metric learning for kernel regression". In: *Artificial Intelligence and Statistics*. 2007, pp. 612–619.

[14] Christopher KI Williams and Matthias Seeger. "Using the Nyström method to speed up kernel machines". In: *NIPS*. 2001, pp. 682–688.

[15] Chang Xu, Dacheng Tao, and Chao Xu. "A survey on multi-view learning". In: *arXiv preprint arXiv:1304.5634* (2013).