A nonlinear kernel support matrix machine for matrix learning

Yunfei Ye

Abstract
In many problems of supervised tensor learning, real world data such as face images or MRI scans are naturally represented as matrices, which are also called as second order tensors. Most existing classifiers based on tensor representation, such as support tensor machine and kernelized support tensor machine need to solve iteratively which occupy much time and may suffer from local minima. In this paper, we present a kernel support matrix machine which performs a matrix-form inner product with maximum margin classifier. Specifically, the matrix inner product is introduced to leverage the inherent structural information within matrix data. Further, matrix kernel functions are applied to detect the nonlinear relationships. We analyze a unifying optimization problem for which we propose an asymptotically convergent algorithm. Theoretical analysis for the generalization bounds is derived based on Rademacher complexity with respect to a probability distribution. We demonstrate the merits of the proposed method by exhaustive experiments on both simulation study and a number of real-word datasets from a variety of application domains.

Keywords Kernel support matrix machine · Supervised tensor learning · Reproducing kernel matrix · Hilbert space · Kernel functions

1 Introduction
The supervised learning tasks are often encountered in many fields including pattern recognition, image processing and data mining. Data are represented as feature vectors to handle such tasks. Among all the algorithms based on the vector framework, support vector machine (SVM) \[30\] is the most representative one due to numerous theoretical and computational developments. Later, the support vector method was extended to improve its performance in many applications. Radial basis function classifiers were introduced in SVM to solve nonlinear separable problems \[22\]. The use of SVM for density estimation \[31\] and ANOVA decomposition \[27\] has also been studied. Least squares SVM (LS-SVM) \[28\] modifies the equality constraints in the optimization problem to solve a set of linear equations instead of quadratic ones. Transductive SVM (TSVM) \[12\] tries to minimize misclassification error of a particular test set. \(\nu\)-SVM \[23\] includes a new parameter \(\nu\) to effectively control the number of support vectors for both regression and classification. One-Class SVM (OCSVM) \[24\] aims to identify one available class, while characterizing other classes is either expensive or difficult. Twin SVM (TWSVM) \[14\] is a fast algorithm solving two quadratic programming problems of smaller sizes instead of a large one in classical SVM.

However it is more natural to represent real-world data as matrices or higher-order tensors. Within the last decade, advanced researches have been exploited on retaining the structure of tensor data and extending SVM to tensor patterns. Support tensor machine (STM) was studied to separate multilinear hyperplanes by applying alternating projection methods \[1\]. Tao et al. \[29\] proposed a supervised tensor learning (STL) and extended the classical linear C-SVM \[4\], \(\nu\)-SVM and least squares SVM \[28\] to general tensor patterns. One-Class STM (OCSTM) was generalized to obtain most interesting tensor class with maximal margin \[2, 5\]. Joint tensor feature analysis (JTFA) \[32\] was proposed for tensor feature extraction and recognition. Support higher-order tensor machine (SHTM) \[8\] integrates the merits of linear C-SVM and rank-one decomposition. Kernel methods for tensors were also introduced to solve nonlinear problems. Factor kernel \[26\] calculates the similarity between tensors using techniques of tensor unfolding and singular value decomposition.
decomposition (SVD). Dual structure-preserving kernels (DuSK) [9] is a generalization of SHTM with dual-tensorial mapping functions to detect dependencies of nonlinear tensors. Support matrix machines (SMM) [16] aims to solve a convex optimization problem considering a hinge loss plus a so-called spectral elastic net penalty. A novel Kernelized support tensor machine (KSTM) integrates kernelized tensor factorization to approximate the tensor data in kernel space [10]. These methods essentially take advantage of the low-rank assumption, which can be used for describing the correlation within a tensor.

In this paper we are concerned with classification problems on a set of matrix data. We present a kernel support matrix machine (KSMM) and it is motivated by the use of matrix Hilbert space [34]. It’s cornerstones is the introduction of a matrix as the inner product to explore the complicated correlation among samples. KSMM is a general framework for constructing a matrix-based hyperplane through calculating the weighted average distance. The hyperplane is determined by two elements: the regression matrix to obtain multiple distances and the weight matrix to determine the relative importance. We show that the regression matrix in our model is indeed a combination of a set of support matrices while the weight matrix is a combination of the matrix inner products of support matrices. Further, matrix kernel functions are applied to detect the nonlinear relationships. We analyze a unifying optimization problem for which we propose an asymptotically convergent algorithm built on the sequential minimal optimization (SMO) [19] algorithm. Generalization bounds of SVM were discussed based on Rademacher complexity with respect to a probability distribution [25]; here we extend to a more general and flexible framework. The contribution of this paper is listed as follows. One main contribution is to develop a new classifier for matrix learning where the optimization problem is solved directly without adopting the technique of alternating projection method. Important special case of the framework includes the classifier of SVM. Another contribution lies within a matrix-based hyperplane that we propose in the algorithm to separate objects instead of determining multiple hyperplanes as in STM.

The rest of this paper is organized as follows. We review our concerned problem in Sect. 2.1. In Sect. 2.2, we discuss the framework of KSMM in linear case. We show its dual problem and present a template algorithm to solve this problem. In Sect. 2.3 we extend to the nonlinear task by adopting the methodology of reproducing kernels. Section 2.4 deals with the generalization bounds based on Rademacher complexity with respect to a probability distribution. Differences among several classifiers are discussed in Sect. 2.5. In Sect. 3 we study our model’s performance in both simulation study and benchmark datasets. Finally, concluding remarks are drawn in Sect. 4.

## 2 Kernel support matrix machine

In this section, we first formulate the binary classification problem for matrix data. Then, we put forward the KSMM which makes a closed connection between matrix Hilbert Space [34] and the STL. We construct a hyperplane in the matrix Hilbert space to separate two communities of examples. Next, the SMO algorithm is introduced to handle with the new optimization problem. We further derive the generalization bounds for KSMM based on Rademacher complexity with respect to a probability distribution. Finally, we analyze and compare the differences of KSMM with other state-of-the-art methodologies.

### 2.1 Problem description

We first introduce some basic notations and definitions. In this study, scales are denoted by lowercase letters, e.g., $s$, vectors by boldface lowercase letters, e.g., $v$, matrices by boldface capital letters, e.g., $M$ and general sets or spaces by gothic letters, e.g., $S$.

The Frobenius norm of a matrix $X \in \mathbb{R}^{m \times n}$ is defined by,

$$
\|X\| = \sqrt{\sum_{i,j=1}^{m,n} x_{ij}^2},
$$

which is a generalization of the normal $\ell_2$ norm for vectors.

The inner product of two same-sized matrices $X, Y \in \mathbb{R}^{m \times n}$ is defined as the sum of products of their entries, i.e.,

$$
\langle X, Y \rangle = \sum_{i,j=1}^{m,n} x_{ij} y_{ij}.
$$

In order to facilitate description, we first formulate the matrix classification problem as follows. Given a set of samples $\{(y_i, X_i)\}_{i=1}^{N}$ for binary classification problem, where $X_i \in \mathbb{R}^{m \times n}$ are the input matrix data and $y_i \in \{-1, +1\}$ are the corresponding class labels. As we have seen, $X_i$ is represented in matrix form. To fit a vector-based classifier, one general approach is to reshape $X_i$ into a vector. Then, the soft margin SVM is defined as,

$$
\min_{w,b} \frac{1}{2} w^t w + C \sum_{i=1}^{N} \left[ 1 - y_i \left( w^t x_i + b \right) \right]_+,
$$

where $x_i = \text{vec}(X_i)$, $[1-u]_+ = \max\{0, 1-u\}$ is called the hinge loss function, $w \in \mathbb{R}^m$ is the regression parameter, $b \in \mathbb{R}$ is the offset term and $C \in \mathbb{R}$ denotes a penalty parameter.

To perform matrix data directly, Hao et al. [8] consider an equivalent formulation as follows:
\[
\min_{\mathbf{W},b}\frac{1}{2}\text{tr}((\mathbf{W}^\top\mathbf{W}) + C \sum_{i=1}^{N} [1 - y_i (\text{tr}(\mathbf{W}^\top\mathbf{X}_i) + b)]_+.
\]

where \( \mathbf{W} \in \mathbb{R}^{m \times n} \). Moreover, \( \text{tr}(\mathbf{W}^\top\mathbf{W}) = \text{vec}(\mathbf{W})^\top\text{vec}(\mathbf{W}) \) and \( \text{tr}(\mathbf{W}^\top\mathbf{X}_i) = \text{vec}(\mathbf{W})^\top\text{vec}(\mathbf{X}_i) \) which implies that the reformulation cannot capture the correlation among columns or rows in the initial matrix.

To take the structural information into consideration, one natural approach is to consider the dependency of the regression matrix \( \mathbf{W} \). Intuitively, one can consider the following formulation,

\[
\min_{\mathbf{W},b}\mathcal{L}(\mathbf{W}) + P(\mathbf{W}),
\]

where \( \mathcal{L}(\mathbf{W}) \) is a loss function and \( P(\mathbf{W}) \) is a penalty function defined on \( \mathbf{W} \).

Since \( \mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^\top \), factorization techniques have been introduced to explore nonlinear relationships of matrix data in kernel space [7–9]. Another intuitive way to leverage the structural information of matrix data is by imposing the low-rank constraint. However, determining the rank of a matrix can be NP-hard while the nuclear norm \( \|\mathbf{W}\|_* \) is best convex approximation of \( \text{rank}(\mathbf{W}) \) [35]. Typically, Luo et al. [16] suggested \( P(\mathbf{W}) = \frac{1}{2}\text{tr}(\mathbf{W}^\top\mathbf{W}) + \tau \|\mathbf{W}\|_* \) which is a spectral extension of the conventional elastic net to formulate the optimization problem.

### 2.2 Kernel support matrix machine in linear case

Inspired by the previous work, we introduce the matrix inner product to the framework of STM in matrix space. The matrix inner product is defined as follows:

**Definition 1 (Matrix inner product)** Let \( \mathcal{H} = \mathbb{R}^{m \times n} \) be a real linear space, the matrix inner product is a mapping \( \langle \cdot , \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}^{m \times n} \) satisfying the following properties, for all \( \mathbf{X}, \mathbf{X}_1, \mathbf{X}_2, \mathbf{Y} \in \mathcal{H} \).

1. \( \langle \mathbf{X}, \mathbf{Y} \rangle_{\mathcal{H}} = \langle \mathbf{Y}, \mathbf{X} \rangle^\top_{\mathcal{H}} \).
2. \( \langle \lambda \mathbf{X}_1 + \mu \mathbf{X}_2, \mathbf{Y} \rangle_{\mathcal{H}} = \lambda \langle \mathbf{X}_1, \mathbf{Y} \rangle_{\mathcal{H}} + \mu \langle \mathbf{X}_2, \mathbf{Y} \rangle_{\mathcal{H}} \).
3. \( \langle \mathbf{X}, \mathbf{X} \rangle_{\mathcal{H}} = 0 \) if and only if \( \mathbf{X} \) is a null matrix.
4. \( \langle \mathbf{X}, \mathbf{X} \rangle_{\mathcal{H}} \) is positive semidefinite.

This thus motivates us to reformulate the optimization problem in STM. Considering a set of samples \( \{(y_i, \mathbf{X}_i)\}_{i=1}^{N} \) for binary classification problem, where \( \mathbf{X}_i \in \mathbb{R}^{m \times n} \) are input matrix data and \( y_i \in \{-1, +1\} \) are corresponding class labels. We assume that \( \{\mathbf{X}_i\}_{i=1}^{N} \) and \( \mathbf{W} \in \mathbb{R}^{m \times n} \) are in the matrix Hilbert space \( \mathcal{H} \). \( \mathbf{V} \in \mathbb{R}^{m \times n} \) is a symmetric matrix satisfying:

\[
\langle \mathbf{X}, \mathbf{X} \rangle_{\mathcal{H}} \langle \mathbf{V}, \mathbf{V} \rangle \geq 0
\]

for all \( \mathbf{X} \in \mathcal{H} \). In particular, the optimization problem of KSMM can be described in the following way:

\[
\min_{\mathbf{W},b,\xi,\nu} \frac{1}{2}\|\mathbf{W}\|_{\mathcal{H}}^2(\mathbf{V}) + C \sum_{i=1}^{N} \xi_i
\]

s.t. \( y_i \left( \langle \mathbf{W}, \mathbf{X}_i \rangle_{\mathcal{H}}, \frac{\mathbf{V}}{\|\mathbf{V}\|} + b \right) \geq 1 - \xi_i, 1 \leq i \leq N \)

\( \xi \geq 0 \),

where the inner product \( \langle \cdot , \cdot \rangle_{\mathcal{H}} \) is specified as \( \langle \mathbf{A}, \mathbf{B} \rangle_{\mathcal{H}} = \mathbf{A}^\top \mathbf{B} \) for \( \mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n} \) and the norm is defined as \( \|\cdot\|_{\mathcal{H}}(\mathbf{V}) = (\langle \cdot, \cdot \rangle_{\mathcal{H}}(\mathbf{V}, \mathbf{V})^{1/2} \). \( \xi = [\xi_1, \ldots, \xi_N]^\top \) is the vector of all slack variables of training examples and \( C \) is the trade-off between the classification margin and misclassification error.

**Remark 1** The proposed problem (7) degenerates into the classical SVM when \( n = 1 \).

**Remark 2** Two classes of labels are separated by a hyperplane \( \langle \mathbf{W}, \mathbf{X}_i \rangle_{\mathcal{H}}, \frac{\mathbf{V}}{\|\mathbf{V}\|} + b = 0 \). The expression can be decomposed into two parts: one is controlled by regression matrix \( \mathbf{W} \) while the other is constrained by weight matrix \( \mathbf{V} \). Each entry of the matrix inner product \( \langle \mathbf{W}, \mathbf{X}_i \rangle_{\mathcal{H}} \) measures a relative “distance” from \( \mathbf{X} \) to a certain hyperplane. To make explicit those values underlying their own behavior, we introduce a weight matrix \( \mathbf{V} \) which determines the relative importance of each hyperplane on the average.

Once the model has been solved, the class label of a testing example \( \mathbf{X} \) can be predicted as follows:

\[
y(\mathbf{X}) = \text{sgn} \left( \langle \mathbf{W}, \mathbf{X} \rangle_{\mathcal{H}}, \frac{\mathbf{V}}{\|\mathbf{V}\|} \right) + b \right).
\]

The Lagrangian function of the optimization problem (7) is,

\[
\mathcal{L}(\mathbf{W}, b, \xi, \nu) = \frac{1}{2}\|\mathbf{W}\|_{\mathcal{H}}^2(\mathbf{V}) + C \sum_{i=1}^{N} \xi_i
\]

\[-\sum_{i=1}^{N} \alpha_i \left( y_i \left( \langle \mathbf{W}, \mathbf{X}_i \rangle_{\mathcal{H}}, \frac{\mathbf{V}}{\|\mathbf{V}\|} \right), b \right) - 1 + \xi_i \]

\[-\sum_{i=1}^{N} \beta_i \xi_i.
\]

Let the partial derivatives of \( \mathcal{L}(\mathbf{W}, b, \xi, \nu) \) with respect to \( \mathbf{W}, b, \xi \) and \( \nu \) be zeros respectively, we have,
\[
\mathbf{W} = \sum_{i=1}^{N} a_i y_i \mathbf{x}_i,
\]
\[
\sum_{i=1}^{N} a_i y_i = 0,
\]
\[
a_i + \beta_i = C, \quad i = 1, \ldots, N,
\]
\[
\mathbf{V} = a \sum_{i,j=1}^{N} a_i a_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_H,
\]
where \(a\) is a positive real number. Substituting (10) into (9) yields the dual of the optimization problem (7) as follows:
\[
\begin{align*}
\max_{a} & \sum_{i=1}^{N} a_i - \frac{1}{2} \left( \sum_{i,j=1}^{N} a_i a_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_H \sum_{i,j=1}^{N} a_i a_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_H \right) \\
\text{s.t.} \quad & \sum_{i=1}^{N} a_i y_i = 0, \\
& 0 \leq a_i \leq C, \quad 1 \leq i \leq N
\end{align*}
\]
(11)

where \(a_i\) are the Lagrange multipliers.

Notice that for all \(\mathbf{x} \in H\), we have,
\[
\left\langle \mathbf{x}^T \mathbf{V} \mathbf{x}, \frac{\mathbf{V}}{\|\mathbf{V}\|} \right\rangle = \left\langle \mathbf{x}^T \frac{\mathbf{V}}{\|\mathbf{V}\|} \mathbf{x} \right\rangle = \frac{1}{\|\mathbf{V}\|} \sum_{j=1}^{m} \left( \sum_{i=1}^{n} x_{i,j} w_{i,j} \right) \left( \sum_{j=1}^{m} x_{i,j} w_{i,j} \right)
\]
\[
= \frac{1}{\|\mathbf{V}\|} \sum_{j=1}^{m} \left( \sum_{i=1}^{n} x_{i,j} w_{i,j} \right) \left( \sum_{j=1}^{m} x_{i,j} w_{i,j} \right)
\]
\[
= \frac{1}{\|\mathbf{V}\|} \sum_{j=1}^{m} \left( \sum_{i=1}^{n} x_{i,j} w_{i,j} \right)^2 \geq 0,
\]
which indicates that the matrix \(\mathbf{V}\) we derive from Lagrange multiplier method satisfies condition (6).

Furthermore, the Karush–Kuhn–Tucker (KKT) conditions are fulfilled when the optimization problem (7) is solved, that is for all \(i\):
\[
a_i = 0 \Rightarrow y_i f(\mathbf{x}_i) \geq 1,
\]
\[
0 < a_i < C \Rightarrow y_i f(\mathbf{x}_i) = 1,
\]
\[
a_i = C \Rightarrow y_i f(\mathbf{x}_i) \leq 1,
\]
where \(f(\mathbf{x}_i) = \langle \mathbf{w}, \mathbf{x}_i \rangle_H, \frac{\mathbf{v}}{\|\mathbf{v}\|} \rangle + b\). Next, we summarize and improve the SMO algorithm to solve the optimization problem (11). At each step, SMO chooses two Lagrange multipliers to jointly optimize the objective function \(J(\mathbf{a})\) while other multipliers are fixed, which can be computed as follows:
\[
J(\mathbf{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \left( \sum_{i,j=1}^{N} a_i a_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_H \right)^{1/2}
\]
(14)

For convenience, all quantities that refer to the first multiplier will have a subscript 1, while all quantities that refer to the second multiplier will have a subscript 2. Without loss of generality, the algorithm computes the second Lagrange multiplier \(a_2\) and then updates the first Lagrange multiplier \(a_1\) at each step. Notice that \(a_1 y_1 + a_2 y_2 = \text{constant} \Leftrightarrow a_1 = \text{constant} - y_1 y_2 a_2\), we can rewrite (14) in terms of \(a_2\) as:
\[
J(a_2) = \left( 1 - y_1 y_2 \right) a_2 - \frac{1}{2} \left( \left( \mathbf{w}^T \mathbf{W} \mathbf{W}^T \mathbf{w} \right) \right)^{1/2} + \text{constant},
\]
(15)

where \(\mathbf{W} = \sum_{i=1}^{N} a_i y_i \mathbf{x}_i\), and \(\frac{\partial J}{\partial a_2} = y_2 (\mathbf{x}_2 - \mathbf{x}_1) = \mathbf{A}\).

Compute the partial derivative and second partial derivative of the objective function, we can obtain that,
\[
\frac{\partial J}{\partial a_2} = 1 - y_1 y_2 - \left\langle \mathbf{A}^T \mathbf{W} \mathbf{W}^T \mathbf{W} \mathbf{w} \right\rangle \frac{1}{\left( \mathbf{w}^T \mathbf{W} \mathbf{W}^T \mathbf{w} \right)^{1/2}},
\]
\[
\frac{\partial^2 J}{\partial a_2^2} = -\frac{\left( \left\langle \mathbf{A}^T \mathbf{A} \mathbf{W}^T \mathbf{W} \right\rangle + \left\langle \mathbf{A}^T \mathbf{W} \mathbf{A}^T \mathbf{W} + \mathbf{W} \mathbf{A}^T \mathbf{W} \right\rangle \right) \left\langle \mathbf{W}^T \mathbf{W} \mathbf{W} \mathbf{w} \right\rangle - 2 \left\langle \mathbf{A}^T \mathbf{W} \mathbf{W} \mathbf{w} \right\rangle^2}{\left( \mathbf{w}^T \mathbf{W} \mathbf{W} \mathbf{w} \right)^{3/2}},
\]
(16)

We can easily derive that,
\[
\left\langle \mathbf{A}^T \mathbf{A} \mathbf{W}^T \mathbf{W} \right\rangle = \left\langle \mathbf{A}^T \mathbf{A} \right\rangle_{H^*} \mathbf{w} = \frac{2}{\alpha} \mathbf{w} \mathbf{w} \geq 0,
\]
\[
\left\langle \mathbf{A}^T \mathbf{W} \mathbf{A}^T \mathbf{W} + \mathbf{W} \mathbf{A}^T \mathbf{W} \right\rangle \left\langle \mathbf{W}^T \mathbf{W} \mathbf{W} \mathbf{w} \right\rangle - 2 \left\langle \mathbf{A}^T \mathbf{W} \mathbf{W} \mathbf{w} \right\rangle^2
\]
\[
= \frac{1}{2} \left( \left\langle \mathbf{A}^T \mathbf{W} + \mathbf{W} \mathbf{A}^T \mathbf{W} + \mathbf{W} \mathbf{A}^T \mathbf{W} \right\rangle \left\langle \mathbf{W}^T \mathbf{W} \mathbf{W} \mathbf{w} \right\rangle - \frac{1}{2} \left\langle \mathbf{A}^T \mathbf{W} + \mathbf{W} \mathbf{A}^T \mathbf{W} \mathbf{W} \mathbf{w} \right\rangle^2 \right) \geq 0.
\]
(17)
The second inequality holds according to the Cauchy-Schwarz inequality. The second partial derivative of the objective function is no more than zero. Therefore, the location of the constrained maximum of the objective function is either at the bounds or at the extreme point.

On the other hand, let \( \frac{df}{da_2} = 0 \) be zero we obtain a function of the sixth degree which does not have a closed-form. Therefore, the Newton’s method is applied to iteratively find the optimal value of \( a_2 \). At each step, we update the \( a_2 \) as:

\[
\begin{align*}
a_2^{n+1} &= a_2^n - J''(a_2^n) \cdot a_2^n, \\
a_2^n &= a_2^{old},
\end{align*}
\]

until it converges to \( a_2^{new} \).

Remember that the two Lagrange multipliers must fulfill all of the constraints of problem (9) that the lower bound \( L \) and the upper bound \( H \) of \( a_2 \) can be concluded as for labels \( y_1 \neq y_2 \):

\[
L = \max\{0, a_2^{old} - a_1^{old}\}, \quad H = \min\{C, C + a_2^{old} - a_1^{old}\}.
\]

If labels \( y_1 = y_2 \), then the following bounds apply to \( a_2 \):

\[
L = \max\{0, a_2^{old} + a_2^{old} - C\}, \quad H = \min\{C, a_1^{old} + a_2^{old}\}.
\]

Next, the constrained maximum is found by clipping the unconstrained maximum to the bounds of the domain:

\[
a_2^{new, clipped} = \begin{cases} H, & \text{if } a_2^{new} \geq H \\ a_2^{new}, & \text{if } L \leq a_2^{new} \leq H \\ L, & \text{if } a_2^{new} \leq L. \end{cases}
\]

Then the value of \( a_1 \) is calculated from the new, clipped \( a_2 \):

\[
a_1^{new} = a_1^{old} + y_1 y_2 (a_2^{new} - a_2^{new, clipped}).
\]

This process is repeated iteratively until the maximum number of outer loops \( M \) is reached or all of the Lagrange multipliers hold the KTT conditions. Typically, we terminate the inner loop of Newton’s method if \( \frac{1}{\|J'(a_2^n)\|^2} < \varepsilon \), where \( \varepsilon \) is a threshold parameter.

Then we present the strategy on the choices of two Lagrange multipliers. When iterates over the entire training set, the first one which violates the KTT condition (13) is determined as the first multiplier.

Once a violated example is found, the second multiplier is chosen randomly.

Furthermore, we would like to update the parameter \( b \) using following strategy: the parameter \( b_2 \) updates when the new \( a_2 \) is not at the bounds which forces the output \( f(X_2) \) to be \( y_2 \):

\[
b_2 = y_2 - \left( \frac{\sum_{j=1}^{N} \hat{a}_j y_j (X_j, X_2) - \sum_{j=1}^{N} \hat{a}_j y_j (X_j, X_2) H}{\|\sum_{j=1}^{N} \hat{a}_j y_j (X_j, X_2) H\|} \right).
\]

The parameter \( b_1 \) updates when the new \( a_1 \) is not at the bounds which forces the output \( f(X_1) \) to be \( y_1 \):

\[
b_1 = y_1 - \left( \frac{\sum_{j=1}^{N} \hat{a}_j y_j (X_1, X_j H) - \sum_{j=1}^{N} \hat{a}_j y_j (X_1, X_j H)}{\|\sum_{j=1}^{N} \hat{a}_j y_j (X_1, X_j H\|} \right).
\]

When both \( b_1 \) and \( b_2 \) are updated, they are equal. When both Lagrange multipliers are at the bounds, any number in the interval between \( b_1 \) and \( b_2 \) is consistent with the KKT conditions. We choose the threshold to be the average of \( b_1 \) and \( b_2 \). The Pseudo-code of the overall algorithm is listed above.

The objective function increases at every step and the algorithm will converge asymptotically. Even though the extra Newton’s method is applied in each iteration, the overall algorithm does work efficiently.

### 2.3 Kernel support matrix machine in nonlinear case

Kernel methods, which refer to as “kernel trick” were brought to the field of machine learning in the twentieth century to overcome the difficulty in detecting certain dependencies of nonlinear problems. A Reproducing Kernel Matrix Hilbert Space (RKMHS) [34] was introduced to develop
kernel theories in the matrix Hilbert space. In this section, we define a nonlinear mapping and apply these algorithms in our KSMM.

We start by defining the following mapping on a matrix $X \in \mathbb{R}^{m \times n}$ column by column.

$$\Phi : X \mapsto \Phi(X) = [\Phi(x_1), \ldots, \Phi(x_n)] \in \mathbb{R}^{m \times n},$$

where $\Phi(X)$ is in a matrix Hilbert space $H'$. Naturally, the kernel function is defined as inner products of elements in the feature space:

$$K(X_i, X_j) = \langle \Phi(X_i), \Phi(X_j) \rangle_{H'} \in \mathbb{R}^{m \times n}. \quad (26)$$

Further details of the structure of a RKHS can be found in [34].

Substituting (26) into (11) with mapping $\Phi$ yields the nonlinear problem as follows:

$$\begin{align*}
\max_{a} & \sum_{i = 1}^{N} a_i - \frac{1}{2} \left( \sum_{i,j=1}^{N} a_i a_j y_i y_j K(X_i, X_j) \right) \\
\text{s.t.} & \sum_{i=1}^{N} a_i y_i = 0, \\
& 0 \leq a_i \leq C, \quad 1 \leq i \leq N.
\end{align*} \quad (27)$$

The revised SMO algorithm is still applied under such circumstance. We emphasize that $W = \sum_{i=1}^{N} a_i y_i \Phi(x_i)$ and $\frac{dW}{da_i} = y_i \Phi(x_i) - \Phi(x_i) = A$. The following abbreviations are derived to compute the partial derivative and second partial derivative of the object function $J(\alpha)$ in (16).

$$\begin{align*}
A^T A & = K(X_2, X_2) + K(X_1, X_1) - K(X_1, X_2) - K(X_2, X_1), \\
A^T W & = \sum_{i=1}^{N} a_i y_i y_i \Phi(x_i), \\
W^T W & = \sum_{i,j=1}^{N} a_i a_j y_i y_j K(X_i, X_j).
\end{align*} \quad (28)$$

Some possible choices of $K$ include,

- Linear kernel: $K(X_i, Y) = X_i^T Y + a I_{m \times m}$,
- Polynomial kernel: $K(X_i, Y) = (X_i^T Y + a I_{m \times m})^\beta$,
- Gaussian kernel: $K(X_i, Y) = \exp(\gamma \|X_i - Y\|)$, where $a \geq 0, \beta \in \mathbb{N}, \gamma > 0, X, Y \in H = C^{m \times n}$ the $i$-th column of $X$ and $\circ$ is the Hadamard product [11].

Additionally, if $\Phi$ is an identical mapping with $K(X_i, X_j) = X_i^T X_j$, the optimization problem will degenerate into a linear one.

Hilbert spaces of vector-valued functions [17], matrix-valued reproducing kernels [6, 20] and learning of function-valued functions [13] have been used as a theoretical framework to develop nonlinear methods. The important difference is that the kernel is represented as an operator-valued function in the literature, while in our framework it is an inner product. Furthermore, these operator-valued kernels are defined upon two scalar Hilbert spaces instead of one matrix Hilbert space in our research.

### 2.4 Generalization bounds for KSMM

In this section, we use Rademacher complexity to obtain generalization bounds for soft-SVM and STM with Frobenius norm constraint. We will show how this leads to generalization bounds for KSMM.

Throughout this section, we assume that $b = 0$ in the optimization problem to simplify the derivations. To simplify the notation, we denote,

$$\mathcal{F} = \ell \circ H_p = \{ z \mapsto \ell(h, z) : z \in \mathcal{Z}, h \in H_p \},$$

where $\mathcal{Z}$ is a domain, $H_p$ is a hypothesis class and $\ell$ is a loss function. Given $f \in \mathcal{F}$, we define,

$$L_D(f) = \mathbb{E}_{z \sim D}[f(z)], \quad L_S(f) = \frac{1}{N} \sum_{i=1}^{N} f(z_i),$$

where $D$ is the distribution of elements in $\mathcal{Z}$, $S$ is the training set and $N$ is the number of examples in $S$.

We repeat the symbols and assumptions in Sect. 2.2 for further study. A STM problem in matrix space can be reformulated as:

$$\begin{align*}
\min_{W, \xi} & \frac{1}{2} \|W\|^2 + C \sum_{i=1}^{N} \xi_i \\
\text{s.t.} & y_i((W, X_i) + b) \geq 1 - \xi_i, \quad 1 \leq i \leq N \\
& \xi \geq 0,
\end{align*} \quad (31)$$

where $W, \{X_i\}_{i=1}^{N} \in \mathbb{R}^{m \times n}$ and $\|\cdot\|$ is the Frobenius norm.

Consider the vector as a specialization of matrix that the number of its row or column is equal to one, we rewrite the theorem from Shalev-Shwartz and Ben-David [25] in the following way. It bounds the generalization error for SVM and STM (for matrix data) of all predictors in $H_p$ using their empirical error.

**Theorem 1** Suppose that $D$ is a distribution over $\mathcal{X} \times \mathcal{Y}$ such that with probability $1$ we have that $\|X\| \leq R$. Let $H_p = \{ W : \|W\| \leq B \}$ and let $\ell' : H_p \times \mathcal{Z} \rightarrow \mathbb{R}$ be a loss function of the form,

$$\ell'(W, (X, y)) = \Phi(\langle W, X \rangle, y), \quad (32)$$

such that for all $y \in \mathcal{Y}$, $a \mapsto \Phi(a, y)$ is a $r$-Lipschitz function and $\max_{a \in [-BR, BR]} \Phi(a, y) \leq c$. Then, for any $\delta \in (0, 1)$, with probability of at least $1 - \delta$ over the choice of an i.i.d. sample of size $N$,
\[ \forall W \in H_p, L_D(W) \leq L_S(W) + \frac{2 \rho BR}{\sqrt{N}} + c\sqrt{\frac{2 \ln(2/\delta)}{N}}. \] (33)

**Remark 3** When \( m = 1 \) or \( n = 1 \), the matrix \( X \in \mathbb{R}^{\infty \times \infty} \) transforms into a vector and its Frobenius norm is consistent with the corresponding Euclidean norm. The optimization problems in both classifiers are identical.

In the case of KSMM, we have the following result where we denote by \( \| \cdot \|_H(V) \).

**Theorem 2** Suppose that \( D \) is a distribution over \( H \times \mathcal{Y} \) where \( H \) is a matrix Hilbert space such that with probability 1 we have that \( \|X\|_H(V) \leq R' \). Let \( H'_p = \{ W' : \|W'\|_H(V) \leq B' \} \) and let \( \varepsilon : H'_p \times \mathcal{Z} \to \mathbb{R} \) be a loss function of the form,

\[ \varepsilon(W', (X, y)) = \Phi \left( \left\langle \langle W', X \rangle_H, \frac{V}{\|V\|} \right\rangle, y \right). \] (34)

such that for all \( y \in \mathcal{Y} \), \( a \mapsto \Phi(a, y) \) is a \( \rho \)-Lipschitz function and \( \max_{a \in [-B'R', B'R']} \| \Phi(a, y) \| \leq c' \). Then, for any \( \delta \in (0, 1) \), with probability of at least \( 1 - \delta \) over the choice of an i.i.d. sample of size \( N \),

\[ \forall W' \in H'_p, L_D(W') \leq L_S(W') + \frac{2 \rho B'R'}{\sqrt{N}} + c'\sqrt{\frac{2 \ln(2/\delta)}{N}}. \] (35)

**Proof** See Appendix 1.

The following corollary compares the generalization bounds with the same hinge-loss function \( \Phi(a, y) = \max \{0, 1 - ay\} \).

**Corollary 1** In the same domain of \( X \in H \) and \( W \in H_p \), we have \( R' \leq R, B' \leq B \) and \( c' \leq c \).

**Proof** For any \( X \in \mathbb{R}^{\infty \times \infty} \),

\[ \|X\|_H^2 = \left\langle \langle X, X \rangle_H, \frac{V}{\|V\|} \right\rangle = \left\langle X^T X, \frac{V}{\|V\|} \right\rangle = \frac{1}{\|V\|} \sum_{p,q=1}^m \sum_{i=1}^n x_{ip} x_{iq} v_{pq} \]

\[ \leq \frac{1}{\|V\|} \left( \sum_{p,q=1}^m \left( \sum_{i=1}^n x_{ip}^2 \right) \right)^{1/2} \left( \sum_{p,q=1}^m \left( \sum_{i=1}^n x_{ip}^2 \right) \right)^{1/2} \]

\[ \leq \left( \sum_{p,q=1}^m \left( \sum_{i=1}^n x_{ip}^2 \right) \right)^{1/2} \left( \sum_{p,q=1}^m \left( \sum_{i=1}^n x_{ip}^2 \right) \right)^{1/2} \]

\[ = \|X\|^2. \] (36)

Thus, \( R' \leq R \) and so does \( B' \leq B \). With \( R' \leq R, B' \leq B \), we have \([-B'R', B'R'] \subseteq [-BR, BR] \) and \( c' \leq c \).

**Theorem 3** Suppose that \( D \) is a distribution over \( H \times \mathcal{Y} \) where \( H \) is a matrix Hilbert space such that with probability 1 we have that \( \|X\|_1 = \max \sum_{i=1}^n |x_i| \) and \( \|X\|_\text{max} = \max \sum_{i=1}^n |x_i| \) for \( X \in \mathbb{R}^{\infty \times \infty} \). The following theorem bounds the generalization error of all predictors in \( H_p \), using their empirical error.

\[ \forall W \in H_p, L_D(W) \leq L_S(W) + \frac{2 \rho BRn}{\sqrt{N}} + c\sqrt{\frac{2 \ln(2/\delta)}{N}}. \] (38)

**Proof** See Appendix 2.

Therefore, we have two bounds given in Theorem 2 and Theorem 3 of KSMM. Apart from the extra \( n \ln(n) \) factor, they look in a similar way. These two theorems are constrained to different prior knowledge, one captures low \( H \)-norm assumption while the latter is limited to low max norm on \( W \) and low 1-norm on \( X \). Note that there is no limitation
on the dimension of \( W \) to derive the bounds in which kernel methods can be naturally applied.

### 2.5 Analysis of KSMM versus other methods

We discuss the differences of MRMLKSVM [7], SVM, STM, SHTM [8], DuSK [9] and our new method as follows:

DuSK, which uses CP decomposition and a dual-tensorial mapping to derive a tensor kernel, is a generalization of SHTM. KSMM constructs a matrix-based hyperplane with Newton’s method applied in the process of seeking appropriate parameters. All the optimization problems mentioned above only need to be solved once. Based on the alternating projection method, STM, MRMLKSVM need to be solved iteratively, which consume much more time. For a set of matrix samples \( \{X_i, y_i\}_{i=1}^N \), the memory space occupied by SVM is \( O(N+1)mn + 1 \), STM requires \( O(Nmn + m + n + 1) \), DuSK requires \( O(N+1)(r(m+n)+1) \), MRMLKSVM requires \( O(Nmn + r(m+n) + 1) \) and KSMM requires \( O((N+2)mn + 1) \), where \( r \) is the rank of matrix. KSMM calculates weight matrix \( V \) to determine the relative importance of each hyperplane on the average.

Naturally STM is a multilinear SVM using different hyperplanes to separate the projections of data points. KSMM is a nonlinear STL and construct a single hyperplane in the matrix Hilbert space.

From the previous work [3], we know that the computational complexity of SVM is \( O(N^2mn) \), thus STM is \( O(2N^2T_i mn) \), DuSK is \( O(N^2r^2(m+n)) \), MRMLKSVM is \( O(2N^2T_2m^2) \), while the complexity of KSMM is \( O(N^2Pmn^2) \), where \( \{T_i\}_{i=1}^2 \) is the corresponding number of iterations and \( P \) is the average number of iterations of Newton’s method, which is usually small in practice. Moreover, its complexity can be narrowed for the optimal time complexity of multiplication of square matrices has been \( O(n^2.3728639) \) up to now [15].

### 3 Experiments

In this section, we conduct one simulation study on synthetic data and four experiments on benchmark datasets. We validate the effectiveness of KSMM with other methodologies (DuSK [9], SVM, MRMLKSVM [7], STM, SMM [16]), since they have been proven successful in various applications.

We introduce two performance metrics to verify our claims about the improvement of the proposed approach. We report the accuracy which counts on the proportion of correct predictions, \( F_1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \) is the harmonic mean of precision and recall. Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. In multiple classification problems, macro-averaged F-measure [33] is adopted as the average of \( F_1 \) score for each category.

All experiments were conducted on a computer with Intel(R) Core(TM) i7 (2.50 GHZ) processor with 8.0 GB RAM memory. The algorithms were implemented in Matlab.

#### 3.1 Simulation study

In order to get better insight of the proposed approach, we now focus on the behavior of proposed methods in binary classification problems. We apply the Wishart distribution defined over symmetric, positive-definite matrices to build the synthetic data set. Its probability density function is given by,

\[
    f(X) = \frac{1}{2^{np/2} |\Lambda|^{n/2} |\Gamma|^{n/2}} |X|^{(n-p-1)/2} e^{-tr(\Lambda^{-1}X)/2},
\]

where \( X \) and \( \Lambda \) are \( p \times p \) symmetric, positive-definite matrices, \( n \) is the number of degrees of freedom greater than \( p - 1 \) and \( 1_{n} \) is the multivariate gamma function. The problem is verified with the following set-ups:

We generate \( 2N \) samples independently which can be decomposed into a latent value drawn from the Wishart distribution plus a Gaussian white noise. We randomly select half of examples as a training set while other examples are organized as a test set. We set \( N = 50 \) or \( 100 \), \( \Lambda = uu^T, u \sim \mathcal{N}(0, I_p) \), \( n = p \) for the first class, \( n = 2p \) for the second class, and \( p = 10, \ldots, 30 \). For each setting we average results over 10 trials. The input matrices are converted into vectors when it comes to SVM problems. All the kernels select the optimal trade-off parameter from \( C \in \{10^{-2}, 10^{-1}, \ldots, 10^{2}\} \), kernel width parameter from \( \sigma \in \{10^{-4}, 10^{-3}, \ldots, 10^{4}\} \) and rank from \( r \in \{1, 2, \ldots, 10\} \). All the learning machines use the same training and test set. We use cross validation to determine the optimal tradeoff parameters. Gaussian RBF kernels are used on MRMLKSVM, DuSK and SVM which denoted as KSMMRBF, DuSKRBF and SVMRBF respectively while we set \( K(X, Y) = X^TY + \sigma \) in KSMM.

The results are presented in Table 1, where best results are highlighted in bold type. We can observe that KSMM performs well in general. We are interested in accuracy in comparison and one way to understand this is to realize that our kernels are represented as matrices in calculation and Newton’s method is included which occupy much space and time. In addition, the observations demonstrate the size of training set has positive effect on the performance. KSMM has a significant performance even the sample size is small. When the training set is large enough, the accuracy is increasing along with the growing number of attributes. That
is reasonable for the expectation values of examples in two classes are equal to $pA$ and $2pA$ respectively which make it easier to identify as $p$ increases.

### 3.2 Data sets and discussions

Next, we evaluate the performance of our classifier on real data sets coming from variety of domains. We consider the following benchmark datasets to perform a series of comparative experiments on multiple classification problems. We use the ORL\textsuperscript{1} \cite{21}, the Sheffield Face dataset,\textsuperscript{2} the Columbia Object Image Library (COIL-20)\textsuperscript{3} \cite{18} and the Binary Alphadigits.\textsuperscript{4} To better visualize the experimental data, we randomly choose a small subset for each database, as shown in Fig. 1. Table 2 summarizes the properties of all datasets used in our experiments.

\begin{table}
\centering
\caption{Prediction performance in simulation study in terms of accuracy (mean and standard deviation)}
\begin{tabular}{c|c|c|c|c|c|c|c}
\hline
& & & & & & & \\
N & P & STM & SVM\textsubscript{RBF} & DuSK\textsubscript{RBF} & MRMLKSVM\textsubscript{RBF} & SMM & KSMM\textsubscript{Linear} \\
\hline
50 & 10 & 76.4 (6.7) & 80.8 (6.3) & 81.2 (5.2) & 80.4 (5.0) & 81.2 (5.1) & 83.2 (6.1) \\
15 & 82.8 (8.9) & 89.2 (4.1) & 90.0 (1.8) & 90.4 (3.4) & 87.9 (3.5) & 92.4 (2.7) \\
20 & 85.6 (5.9) & 89.6 (3.4) & 88.8 (6.4) & 88.8 (4.3) & 88.6 (2.5) & 91.2 (2.4) \\
25 & 88.0 (4.2) & 92.8 (1.0) & 91.6 (2.0) & 93.6 (0.8) & 92.2 (3.2) & 94.0 (1.3) \\
30 & 84.8 (2.0) & 89.6 (3.4) & 92.4 (2.9) & 93.2 (3.2) & 92.8 (2.2) & 95.2 (2.0) \\
100 & 10 & 81.8 (4.9) & 86.0 (3.2) & 85.0 (4.1) & 79.4 (7.3) & 81.4 (3.2) & 86.6 (2.9) \\
15 & 85.2 (3.2) & 89.0 (3.2) & 87.8 (2.1) & 87.6 (3.0) & 86.7 (1.2) & 89.2 (2.1) \\
20 & 84.8 (6.7) & 90.2 (2.1) & 90.0 (2.1) & 91.0 (2.0) & 89.5 (1.4) & 91.2 (2.5) \\
25 & 89.4 (3.2) & 92.0 (1.1) & 91.6 (2.3) & 93.2 (1.5) & 93.1 (2.1) & 93.4 (2.2) \\
30 & 86.8 (7.4) & 93.6 (3.4) & 92.2 (2.1) & \textbf{94.8 (2.9)} & 93.2 (2.9) & \textbf{94.8 (2.8)} \\
\hline
\end{tabular}
\end{table}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig1}
\caption{Example images for classification problems. a ORL 32 $\times$ 32, b Sheffield face dataset, c Binary Alphadigits, d COIL-20}
\end{figure}

\textsuperscript{1} http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
\textsuperscript{2} https://www.sheffield.ac.uk/eee/research/el/research/face.
\textsuperscript{3} http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
\textsuperscript{4} http://www.cs.toronto.edu/~roweis/data.html.
The ORL 32×32 contains 40 distinct subjects of each of ten different images with 32×32 pixels. For some subjects, the images were taken at different times, varying the lighting, facial expressions and facial details. The Sheffield (previously UMIST) Face Database consists of 564 images of 20 individuals. Each individual is shown in a range of poses from profile to frontal views at the 112×92 field and images are numbered consecutively as they were taken. The COIL-20 is a database of two sets of images in which 20 objects were placed on a motorized turntable against background. We use the second one of 1440 images with backgrounds discarded and sizes normalized, each of which has 128×128 pixels. We crop all images into 32×32 pixels to efficiently apply above algorithms. The Binary Alphadigits is composed of digits of “0” to “9” and capital “A” to “Z” with 20×16 pixels, each of which has 39 examples. In experiments, we randomly choose 50% of images of each individual together as the training set and other images retained as test set for multiple classification.

Table 2

| Dataset       | #Instances | #Class | Size     |
|---------------|------------|--------|----------|
| ORL 32×32     | 400        | 40     | 32×32    |
| UMIST         | 564        | 20     | 112×92   |
| COIL-20       | 1440       | 20     | 128×128  |
| Binary Alphadigits | 1404   | 36     | 20×16    |

Note that parameters of different algorithms are set as in the simulation study. For each setting we average results over 10 trials. We use cross validation to determine the optimal tradeoff parameters. For multiple classification task, we use the strategy of one-against-one (1-vs-1) method. For KSMM, we set \( K(X, Y) = \exp(-\sigma \|X(:, i) - Y(:, j)\|^2)_{noa} \) as the matrix kernel function. Due to the effectiveness of SMM in dealing with data matrices, we examine the convergence behavior in terms of the number of iterations of SMM and KSMM. Figures 2 and 3 summarize experimental results for above datasets. Similar patterns of learning curves are observed in macro-averaged F-measure and accuracy, which shows that KSMM outperforms the baseline methods. We can see that KSMM obtains a better result though SMM exhibits a faster convergence than KSMM, which means that KSMM occupies more time. The SVM approach gives slightly worse result on UMIST, for structural information is broken by straightly convert matrices into vectors. It is worth noting that on Binary Alphadigits dataset it is very hard for classification algorithms to achieve satisfying accuracy since the dimension is low and some labels are rather difficult to
identify, e.g. digit “0” and letter “O”, digit “1” and letter “I”. These results clearly show that KSMM can successfully deal with classification problems. One explanation of the outstanding performance of our method is due to the construction of matrix-based hyperplane which calculates multiple relative distances. The final strategy focuses on the weighted average of these values with weight matrix $V$.

However, most methods in the literature try to separate two classes upon one single hyperplane, even applying the magic of kernels to transform a nonlinear separable problem into a linear separable one in rather high dimension.

Overall, the results indicate that KSMM is a significantly effective and competitive alternative for both binary and multiple classification. Note that any reasonable matrix kernel function can be applied in this study.

### 4 Concluding remarks

In this paper, we have proposed a KSMM for matrix classification problems. Different from conventional kernel methods, KSMM is based on the integration of weighed average distance with maximum margin classifier. Typically this is done by introducing a matrix as the inner product in matrix Hilbert space, which provides a different perspective on leveraging underlying structure. The theoretical analysis of its generalization bounds highlights the reliability and robustness of KSMM in practice. Intuitively, the optimization problem arising in KSMM only needs to be solved once while other tensor-based classifiers, such as STM, MRM-LKSVM need to be solved iteratively.

As our experimental results demonstrate, KSMM is competitive in terms of accuracy with state-of-the-art classifiers on several classification benchmark datasets. As previous work focuses on constraining the properties of regression matrix, this paper provides a new insight into exploiting the structural information of matrix data.

In future work, we will seek technical solutions of (11) to improve efficiency or figure out other approach to the use of matrix Hilbert space since the problem we analyze here is non-convex. We could only obtain a local optimal solution other than a global one which might deteriorate the performance of KSMM in experiments. Another interesting topic would be to design specialized method to learn the matrix kernel and address parameters. Figuring out that

---

**Fig. 3** Comparing the accuracy versus the number of iterations of SMM and KSMM for solving different tasks
matrix kernel functions and STL are closely related, hence, a natural extension to this work is the derivation of a unifying matrix kernel-based framework for regression, clustering, among other tasks.

Acknowledgements The work is supported by National Natural Science Foundations of China under Grant 11531001 and National Program on Key Basic Research Project under Grant 2015CB856004. We are grateful to Dong Han for our discussions.

Appendix 1: Proof of Theorem 2

First, we recall some basic notations that are useful to our analysis.

The Rademacher complexity of $F$ with respect to $S$ is defined as follows:

$$ R(F \circ S) = \frac{1}{N} \mathbb{E}_{\sigma \sim \{\pm 1\}^N} \left[ \sup_{f \in F} N \sum_{i=1}^{N} \sigma_i f(z_i) \right]. \quad (40) $$

More generally, given a set of vectors, $A \subset \mathbb{R}^N$, we define,

$$ R(A) = \frac{1}{N} \mathbb{E}_{\sigma} \left[ \sup_{a \in A} \sum_{i=1}^{N} \sigma_i a_i \right]. \quad (41) $$

To prove the theorem about generalization bounds for KSMM, we rely on the following lemmas:

**Lemma 1** Assume that for all $z$ and $h \in H$, we have that $||h(z)|| \leq c$, then with probability at least $1 - \delta$, for all $h \in H_p$,

$$ L_D(h) - L_S(h) \leq 2 \mathbb{E}_{S' \sim D'} R(\ell \circ H_p \circ S') + c \sqrt{\frac{2 \ln(2/\delta)}{N}}. \quad (42) $$

**Lemma 2** For each $i = 1, \ldots, N$, let $\Phi_i : \mathbb{R} \to \mathbb{R}$ be a $\rho$-Lipschitz function, namely for all $a, \beta \in \mathbb{R}$ we have $|\Phi_i(a) - \Phi_i(\beta)| \leq \rho |a - \beta|$. For $a \in \mathbb{R}^N$, let $\Phi(a)$ denote the vector $[\Phi_1(a_1), \ldots, \Phi_N(a_N)]$ and $\Phi(A) = \{\Phi(a) : a \in A\}$. Then,

$$ R(\Phi(A)) \leq \rho R(A). \quad (43) $$

The proof of Lemma 1 and 2 can be discovered in Shalev-Shwartz and Ben-David [25]. Additionally, we present the next lemma.

**Lemma 3** Let $S = (X_1, \ldots, X_N)$ be a finite set of matrices in a matrix Hilbert space $H$. Define $H \circ S = \{(\langle W, X_1 \rangle_H, V) \leq \langle W, X_2 \rangle_H, V) \leq \cdots, \langle W, X_N \rangle_H, V) \} : \|W\|_H (V) \leq 1\}$. Then,

$$ R(H \circ S) \leq \frac{\max_i ||X_i||_H(V)}{\sqrt{N}}. \quad (44) $$

**Proof** Using Cauchy-Schwartz inequality, we have,

$$ NR(H \circ S) = \mathbb{E}_{\sigma} \left[ \sup_{a \in H \circ S} N \sum_{i=1}^{N} \sigma_i a_i \right] $$

$$ = \mathbb{E}_{\sigma} \left[ \sup_{W : ||W||_H(V) \leq 1} \sum_{i=1}^{N} \sigma_i \langle W, X_i \rangle_H, V \rangle \right] $$

$$ = \mathbb{E}_{\sigma} \left[ \sup_{W : ||W||_H(V) \leq 1} \left\langle \sum_{i=1}^{N} \sigma_i X_i \right\rangle_H, V \rangle \right] $$

$$ \leq \mathbb{E}_{\sigma} \left[ \left\| \sum_{i=1}^{N} \sigma_i X_i \right\|_H(V) \right]. \quad (45) $$

Next, using Jensen’s inequality we have that,

$$ \mathbb{E}_{\sigma} \left[ \left( \sum_{i=1}^{N} \sigma_i X_i \right)^2 \right] \leq \left( \mathbb{E}_{\sigma} \left[ \left( \sum_{i=1}^{N} \sigma_i X_i \right)^2 \right] \right)^{1/2} \leq \left( \mathbb{E}_{\sigma} \left[ \left( \sum_{i=1}^{N} \sigma_i X_i \right)^2 \right] \right)^{1/2}. \quad (46) $$

Since the variables $\sigma_1, \ldots, \sigma_N$ are independent we have,
\[
E_{\sigma} \left[ \left\| \sum_{i=1}^{N} \sigma_i X_i \right\|_{H}^2 (V) \right] = E_{\sigma} \left[ \sum_{i,j=1}^{N} \sigma_i \sigma_j \langle \langle X_i, X_j \rangle_{H}, \frac{V}{\|V\|} \rangle \right] = \sum_{i,j=1}^{N} \left( \langle \langle X_i, X_j \rangle_{H}, \frac{V}{\|V\|} \rangle \right) \sum_{i,j=1}^{N} \left( \langle \langle X_i, X_j \rangle_{H}, \frac{V}{\|V\|} \rangle \right) E_{\sigma} [\sigma_i \sigma_j] = \sum_{i=1}^{N} \left\| X_i \right\|_{H}^2 (V) \leq N max \left\| X_i \right\|_{H}^2 (V).
\]

Combining these inequalities we conclude our proof. Finally, we complete our proof as follows. Let \( \mathcal{F} = \{ (X, y) \mapsto \Phi(\langle \langle W', X \rangle_{H'}, \frac{V}{\|V\|}, y) : W' \in \mathcal{H}' \} \). Indeed, the set \( \mathcal{F} \circ \mathcal{S} \) can be written as,

\[
\mathcal{F} \circ \mathcal{S} = \left\{ \Phi\left( \langle \langle W', X \rangle_{H'}, \frac{V}{\|V\|}, y \rangle \right) : \mathcal{F} \right\}, \quad \Phi\left( \langle \langle W', X \rangle_{H'}, \frac{V}{\|V\|}, y \rangle \right) : W' \in \mathcal{H}' \}.
\]

And \( R(\mathcal{F} \circ \mathcal{S}) \leq \frac{d B'}{\sqrt{N}} \) with probability 1 follows directly by combining Lemma 2 and 3. Then the claim of Theorem 2 follows from Lemma 1.

**Appendix 2: Proof of Theorem 3**

First, we summarize the following lemma [25], due to Massart, which states that the Rademacher complexity of a finite set grows logarithmically with the size of the set.

**Lemma 4** (Massart lemma) Let \( A = \{a_1, \ldots, a_N \} \) be a finite set of vectors in \( \mathbb{R}^m \). Define \( \bar{a} = \frac{1}{N} \sum_{i=1}^{N} a_i \), Then,

\[
R(A) \leq \max_{a \in A} \left\| a - \bar{a} \right\|_2 \frac{\sqrt{2 \ln N}}{m}.
\]

Define \( \mathcal{H}_{\text{max}} \circ \mathcal{S} = \{ \langle \langle W, X \rangle_{H}, \frac{V}{\|V\|}, \rangle : \|W\|_{\text{max}} \leq 1 \} \). Next we bound the Rademacher complexity of \( \mathcal{H}_{\text{max}} \circ \mathcal{S} \).

**Lemma 5** Let \( S = \{X_1, \ldots, X_N \} \) be a finite set of matrices in \( \mathbb{R}^{m \times m} \). Then,

\[
R(\mathcal{H}_{\text{max}} \circ \mathcal{S}) \leq \max_{1 \leq i \leq N} \left\| X_i \right\|_1 \sqrt{\frac{2(m \ln 2 + \ln n)}{N}}.
\]

**Proof** Using inequality (36), we have,

\[
N R(\mathcal{H}_{\text{max}} \circ \mathcal{S}) = E_{\sigma} \left[ \sup_{a \in \mathcal{H}_{\text{max}} \circ \mathcal{S}} \sum_{i=1}^{N} \sigma_i a_i \right] = E_{\sigma} \left[ \sup_{a \in \mathcal{H}_{\text{max}} \circ \mathcal{S}} \left\| W \right\|_{\text{max}} \sum_{i=1}^{N} \sigma_i \langle \langle W, X_i \rangle_{H}, \frac{V}{\|V\|} \rangle \right] \leq E_{\sigma} \left[ \sup_{a \in \mathcal{H}_{\text{max}} \circ \mathcal{S}} \left\| W \right\| \sum_{i=1}^{N} \sigma_i X_i \right] \leq n \sup_{a \in \mathcal{H}_{\text{max}} \circ \mathcal{S}} \|W\|_{\text{max}} \left\| \sum_{i=1}^{N} \sigma_i X_i \right\|_1.
\]

For each \( j = 1, \ldots, n \), we define \( u_j^{\gamma} = \left( \sum_{i=1}^{m} \gamma_i X_{ij}, \ldots, \sum_{i=1}^{m} \gamma_i X_{Nij} \right) \in \mathbb{R}^N \). Note that \( \|u_j^{\gamma}\|_2 \leq \sqrt{N max_{1 \leq i \leq N} \|X_{ij}\|_1} \). Let \( U = \{u_j^{\gamma} : j = 1, \ldots, n, \gamma \in \{+1\}^m \} \). The right-hand side of Eq. [51] is \( NnR(U) \). Using Massart lemma (Lemma 4) we have that,

\[
R(U) \leq \max_{1 \leq i \leq N} \left\| X_i \right\|_1 \sqrt{\frac{2(m \ln 2 + \ln n)}{N}},
\]

which concludes our proof.

The rest of the proof is identical to the proof of Theorem 2, while relying on Lemma 5 instead of relying on Lemma 3.

**References**

1. Cai D, He X, Han J (2006) Learning with tensor representation. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2006-2716. http://hdl.handle.net/2142/11195
2. Chen Y, Wang K, Zhong P (2016) One-class support tensor machine. Knowl Based Syst 96:14–28
3. Chu C, Kim SK, Lin YA, Yu Y, Bradski G, Ng AY, Olukotun K (2007) Map-reduce for machine learning on multicore. In: Proceedings of the 2006 conference on advances in neural information processing systems, vol 19. MIT Press, Cambridge, pp 281–288
4. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
5. Erfani SM, Baktashmotlagh M, Rajasegarar S, Nguyen V, Leckie C, Bailey J, Ramamohanarao K (2016) R1stm: One-class support tensor machine with randomised kernel. In: Proceedings of the 2016 SIAM international conference on data mining. SIAM, pp 198–206. https://doi.org/10.1137/1.9781611974348.23
6. Evgeniou T, Micchelli CA, Pontil M (2005) Learning multiple tasks with kernel methods. J Mach Learn Res 6:615–637
7. Gao X, Fan L, Xu H (2018) Multiple rank multi-linear kernel support vector machine for matrix data classification. Int J Mach Learn Cybernet 9(2):251–261
8. Hao Z, He L, Chen B, Yang X (2013) A linear space higher-order tensor machine for classification. IEEE Trans Image Process 22(7):2911–2920
9. He L, Kong X, Yu PS, Yang X, Ratin AB, Hao Z (2014) Dusk: a dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In: Proceedings of the 2014 SIAM international conference on data mining. SIAM, pp 127–135. https://doi.org/10.1137/1.9781611973440.15
10. He L, Lu CT, Ma G, Wang S, Shen L, Philip SY, Ratin AB (2017) Kernelized support tensor machines. In: Proceedings of the 34th international conference on machine learning, vol 70, pp 1442–1451
11. Horn RA (1990) The hadamard product. In: Proceedings of symposia in applied mathematics, vol 40. American Mathematical Society, Providence, pp 87–169
12. Joachims T (1999) Transductive inference for text classification using support vector machines. In: Proceedings of the 16th international conference on machine learning, vol 99, pp 200–209
13. Kadri H, Duflos E, Preux P, Rakotomamonjy A, Audiffren J (2016) Operator-valued kernels for learning from functional response data. J Mach Learn Res 17(1):613–666
14. Khemchandani R, Chandra S et al (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
15. Le Gall F (2014) Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th international symposium on symbolic and algebraic computation. ACM, pp 296–303. https://doi.org/10.1145/2608628.2608664
16. Luo L, Xie Y, Zhang Z, Li WJ (2015) Support matrix machines. In: Proceedings of the 32nd international conference on machine learning, vol 37, pp 938–947
17. Micchelli CA, Pontil MA (2005) On learning vector-valued functions. Neural Comput 17(1):177–204
18. Nene SA, Nayar SK, Murase H (1996) Columbia object image library (coil-20). Technical report, Columbia University, CUCS-005-96. http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
19. Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods—support vector learning. MIT Press, Cambridge, pp 185–208
20. Reisert M, Burkhardt H (2007) Learning equivariant functions with matrix valued kernels. J Mach Learn Res 8:385–408
21. Samaria FS, Harter AC (1994) Parameterization of a stochastic model for human face identification. In: Proceedings of 1994 IEEE workshop on applications of computer vision. IEEE, pp 138–142. https://doi.org/10.1109/ACV.1994.341300
22. Schölkopf B, Sung KK, Burges CJ, Girosi F, Niyogi P, Poggio T, Vapnik V (1997) Comparing support vector machines with gaussian kernels to radial basis function classifiers. IEEE Trans Signal Process 45(11):2758–2765
23. Schölkopf B, Smola AJ, Williamson NC, Bartlett PL (2000) New support vector algorithms. Neural Comput 12(5):1207–1245
24. Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
25. Shalev-Shwartz S, Ben-David S (2014) Understanding machine learning: from theory to algorithms. Cambridge University Press, New York
26. Signoretti M, De Lathauwer L, Suykens JA (2011) A kernel-based framework to tensorial data analysis. Neural Netw 24(8):861–874
27. Stitson MO, Gammerman A, Vapnik V, Vovk V, Watkins C, Weston J (1997) Support vector regression with anova decomposition kernels. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods—support vector learning. MIT Press, Cambridge, pp 285–292
28. Suykens JA, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 13(1):1–42
29. Vapnik V (1995) The nature of statistical learning theory. Springer, New York
30. Weston J, Gammerman A, Stitson M, Vapnik V, Vovk V, Watkins C (1997) Density estimation using support vector machines. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods—support vector learning. MIT Press, Cambridge, pp 293–306
31. Wong WK, Lai Z, Xu Y, Wen J, Ho CP (2015) Joint tensor feature analysis for visual object recognition. IEEE Trans Cybern 45(11):2425–2436
32. Yang Y, Liu X (1999) A re-examination of text categorization methods. In: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 42–49. https://doi.org/10.1145/312624.312647
33. Ye Y (2017) The Matrix Hilbert space and its application to matrix learning. arXiv:1706.08110
34. Zhou H, Li L (2014) Regularized matrix regression. J R Stat Soc B 76(2):463–483

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.