Nonparametric learning with matrix-valued predictors in high dimensions

Chanwoo Lee[1], Lexin Li[2], Hao Helen Zhang[3], and Miaoyan Wang[1]

[1] *Department of Statistics, University of Wisconsin-Madison*

[2] *Department of Biostatistics and Epidemiology, University of California at Berkeley*

[3] *Department of Mathematics, University of Arizona*

Summary. We consider the problem of learning the relationship between a binary label response and a high-dimensional matrix-valued predictor. Such data problems are important in brain imaging studies, sensor network localization, and personalized medicine. Existing regression analysis typically takes a parametric procedure by imposing a pre-specified relationship between variables. Parametric models, however, often perform poorly for mis-specified models and for high-dimensional problems, especially in the high dimension, low sample size data. Here, we propose a flexible nonparametric framework for various learning tasks, including classification, level set estimation, and regression, that specifically accounts for the matrix-valued predictors. Unlike classical approaches, our method is distribution-free, allows exponential growth of feature dimension with sample size, and adapts to the possibly non-smooth, non-linear regression function of interest. The proposal achieves *predictive prediction* via a structural risk minimization within a low-rank, sparse model space. Generalization bounds, estimation consistency, and convergence rate are established. We demonstrate the advantage of our method over previous approaches through simulations and applications to human brain connectome data analyses.

Keywords: Nonparametric learning, high-dimensional matrix-valued predictors, sparse and low-rank models, classification, regression, feature selection

1. Introduction

Consider a predictive learning setting where we would like to model the relationship between a matrix-valued predictor $\boldsymbol{X} \in \mathbb{R}^{d\times d}$ and a binary label response $Y \in \{0,1\}$. Matrix-valued predictors ubiquitously arise in modern applications. One example is from electroencephalography studies of alcoholism. The data set records voltage value measured from 64 channels of electrodes on 256 subjects for 256 time points (Zhou and Li, 2014). Each feature is a 256×64 matrix and the response is a binary indicator of subject being alcoholic or control. Another example is pedestrian detection from image data. Each image is divided into 9 regions where local orientation statistics are generated with a total of 22 numbers per region. This yields a 22×9 matrix feature and a binary label response indicating whether the image is pedestrian (Shashua et al., 2004).

Label prediction based on networks is a particularly active problem in neuroscience. In this setting, each individual in the sample is represented by their own brain network, and the nodes (brain regions of interest) shared across all networks through a registration process that maps all individual brains onto a common atlas. In our analysis we use the parcellation of Power et al. (2011) (see Figure 1), which consists of 264 ROIs divided into 14 functional brain systems. A connectivity is then computed for every pair of nodes, resulting in an adjacency matrix of size 264×264 (see Figure 2). Two main approaches have been followed for the brain network analysis. One approach is to reduce the network to its global summary measures such as the average degree, clustering coefficient, or average path length (Bullmore and Sporns (2009)), and use those measures as features for training a classification method. An alternative approach to classification of large networks is to treat edge weights as a "bag of features," vectorizing the unique elements of the adjacency matrix and ignoring the network nature of the data. The number of individual 100....

In the above two examples and many other studies, researchers are interested in *interpretable prediction*, where the goal is to not only make accurate prediction but also identify features that are informative to the prediction. While classical learning algorithms have been successful in prediction with vector-valued predictors, the key challenge with matrix-valued predictors is the high-dimensional, complex structure in the feature space. A naive approach is to transform the feature matrices to vectors and apply classical methods based on vectors to solve the problem. However, this vectorization would destroy the structural information of the data matrices. Moreover, the reshaping matrices to vectors results in high dimensionality which leads to overfitting. Notably, the ambient dimension with matrix feature, d_1d_2, is often comparable to, or even exponentially larger than the number of sample, n. feature selection... Our method exploits the structural information in the data matrix to overcome these challenges.

Our goal in this paper is to develop a distribution-free prediction method that respects the matrix structure of the predictors and produces more interpretable results. To achieve this goal, we use structured sparsity penalties to incorporate the network information by penalizing both the number of submodules and the number of nodes selected.

2. Setting of the learning problem

2.1. Three learning problems

In this section we present the main learning goals of our interest. Let $\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$ denote the matrix-valued predictor, $y \in \{-1, 1\}$ denote the binary label response, and $\mathbb{P}_{\boldsymbol{X},Y}$ denote the unknown joint probability distribution over the pair $(\boldsymbol{X}, y)$. Suppose that we observe a sample of n training data points, $(\boldsymbol{X}_1, y_1), \ldots, (\boldsymbol{X}_n, y_n)$, identically and independently distributed (i.i.d.) according to $\mathbb{P}_{\boldsymbol{X},y}$. Let $(\boldsymbol{X}_{n+1}, y_{n+1})$ be a new test point drawn independently from the same distribution. Our goal is to make reliable prediction about y_{n+1} given the new feature value $\boldsymbol{X}_{n+1}$, with no strong distributional assumptions other than i.i.d. data. When no confusion arises, we often omit the subscript $(n+1)$ and simply write $(\boldsymbol{X}, y)$ for the prototypical test point.

The problem of learning is to find a decision function $f\colon \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}$ that minimizes certain expected risk $R(f)$ under the unknown distribution $\mathbb{P}_{\boldsymbol{X},y}$. The expected risk $R(f)$ is an functional over a given set of candidate functions $f \in \mathcal{F}$; the form of risk depends on the specific learning goals which are detailed in the next paragraph. We focus on the challenging setting where the feature matrix dimension d_1d_2 may grow sub-exponentially with the sample size n, and no assumptions are made about the distributional family of $\mathbb{P}_{\boldsymbol{X},y}$. The only available information is in the training data $\{(\boldsymbol{X}_i, y_i)\}_{i=1}^n$. We consider three main supervised learning problems: classification, level set estimation, and regression.

(a) *Classification*: Classification is the problem of predicting to which of the label $y \in \{-1,1\}$ a new observation $\boldsymbol{X}$ belongs. We formulate the learning problem as finding a binary-valued function (also called a classifier) from $\mathbb{R}^{d_1 \times d_2}$ to $\{-1,1\}$ that minimizes the expected risk. In classification, the expected risk $R(f)$ is defined as the (expected) classification error; i.e., the probability when the true label and the predicted label differs,

$$R(f) = \mathbb{P}_{\boldsymbol{X},y}\left(y \neq \text{sign} f(\boldsymbol{X})\right),$$

where the probability is taken with respect to both the predictor and response from the unknown joint distribution $\mathbb{P}_{\boldsymbol{X},y}$. Note that we have used sign$f$ to denote the binary-valued classifier, in order to distinguish from the real-valued decision function f. In general, it is common, but not necessary, to estimate sign$f$ through f, for the purpose of classification.

(b) *Level set estimation*: The π-level set of p given a fixed $\pi \in [0,1]$ is the set

$$S(\pi) = \{\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2} : p(\boldsymbol{X}) > \pi\}.$$

Accurate and efficient level set estimation plays an important role in many applications. We consider a nonparametric way to estimate the π-level set of the regression function based on classification problem.

(c) *Regression function estimation*: Regression function calculates expectation of y given a feature matrix $\boldsymbol{X}$ on the basis of a training set of data. In our setting, the regression $\mathbb{E}(y|\boldsymbol{X})$ is equivalent to the conditional probability $\mathbb{P}(y = 1|\boldsymbol{X})$ because the class label y is binary. Knowledge about the class probability itself is of significant interest and can tell us the confidence of the outcome of classification.

The three problems represent our learning tasks with increasing difficulties. Classification problem can be completed from level set $S(\frac{1}{2})$ utilizing Bayes rule. The level set estimation problem becomes trivial when we have all information about regression function. Accordingly, classical approach for the three problems is to find a solution for regression first, and address the other two based on the estimation. This is why the regression problem is also called soft classification. However, our approach finds classification rule first and address the level set estimation and regression problem in order. Through the sequence of solving the problems, we successfully solve the problems without assuming probability distribution.

A common challenge is *distribution-free*.

3. From classifications to regression: a level-set approach

3.1. Cost-weighted classification decision boundaries estimate the level sets

The main approach our method is to estimate regression function by estimating a set of level-set ...estimation from a penalized convex optimization problem. This equivalence allows us to use the rich set of methods developed in classification and derive conditional densities where the conditioning events are expressed as solutions to convex optimization problem... draw a picture here...

Geometric boundary?? Insert two figures...Connects two worlds. level-set is widely used in optimization communities,... Bayes boundary in statistical communities.

3.2. Sparse and low-rank decision boundaries

The aforementioned three problems are formulated as empirical structure minimization over functions with matrices as inputs. In this section, we introduce function classes we consider for the minimization problem. We start introducing linear function class in matrix space and generalize nonlinear one based on our proposed matrix kernel. One approach to finding a set of such nodes was proposed by Vogelstein et al. (2013), who called it a signal-subgraph, and proposed finding the minimal set of nodes (called signal vertices) which together are incident to all selected edges (but not every node connected to a selected edge is a signal vertex).

We propose the linear functions as a decision function class, $f(\boldsymbol{X}) = \langle \boldsymbol{B}, \boldsymbol{X} \rangle$, where $\boldsymbol{B}, \boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$ and $\langle \boldsymbol{X}, \boldsymbol{X}' \rangle = \text{Tr}(\boldsymbol{X}^T \boldsymbol{X}')$. The inner product is called Frobenius inner product in the space of matrices and is a generalization of the dot product from vector spaces. We impose low-rankness on the linear predictor to consider the degeneracy of the coefficient matrix $\boldsymbol{B}$ and leverage the structure information within the predictor $\boldsymbol{X}$. Specifically, the coefficient matrix has low-rank r usually much smaller than the matrix size $\min(d_1, d_2)$,

$$\boldsymbol{B} = \boldsymbol{C}\boldsymbol{P}^T \text{ where } \boldsymbol{C} \in \mathbb{R}^{d_1 \times r}, \boldsymbol{P} \in \mathbb{R}^{d_2 \times r} \text{ and } r \leq \min(d_1, d_2). \tag{1}$$

The low-rankness makes distinction from classical classification problem in vector spaces and preserves structural information of feature matrices. The low rank constraint on coefficient matrix has been proposed in Support Vector Machine (SVM) classification problem (Pirsiavash et al., 2009; Luo et al., 2015) in matrix spaces. However these papers only focus on application to the classification problem. Our paper is not confined in classification but further apply the linear function class to level set estimation and regression estimations. Furthermore, we extend linear case to nonlinear case based on matrix kernel concept in the next section. There are some prior work on matrix/tensor kernels. Most of these methods, however, are inadaptive and do not use label information to guide the kernel construction. We propose a new matrix kernel that solves the limitation.
Proposition 1 (Rademacher complexity)**.** .

4. Nonparametric learning with ultra-high dimensional matrices

4.1. Main Result I: Classification with high-dimensional matrices

We consider a single boundary...,

(why large-margin classification??)

three examples: large margin, logistic, neural network.

We estimate a decision function $f\colon \mathbb{R}^{d_1\times d_2}\to\mathbb{R}$ over function class introduced in Section 3.2. The decision function class denoted as $\mathcal{F}$ can be either linear or nonlinear. We propose a large margin classifier that minimizes a cost function in f over the function class $\mathcal{F}$.

$$\min_{f\in\mathcal{F}} n^{-1}\sum_{i=1}^{n} L\left(y_i f(\boldsymbol{X}_i)\right)+\lambda J(f), \tag{2}$$

where $J(f)$ is a regularization term for model complexity and $L(z)$ is a margin loss that is a function of the functional margin $yf(\boldsymbol{X})$. Examples of such loss functions are the hinge loss function $L(z)=(1-z)_+$ and the logistic loss function $L(z)=\log(1+e^{-z})$. For demonstration, we focus on the hinge loss case in Equation (2). However, our estimation schemes and theorems are applicable to general large-margin classifiers.

We present the solution to (2) with nonlinear kernels which incorporate linear case. Based on the considered decision function class, we solve the following optimization problem,

$$(\hat{\boldsymbol{P}}_1,\hat{\boldsymbol{C}},\hat{\boldsymbol{P}}_2)=\operatorname*{arg\,min}_{\{(\boldsymbol{P}_1,\boldsymbol{P}_2,\boldsymbol{C})\in\mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}\times(\mathcal{H}_1\times\mathcal{H}_2)^{r\times r}\}} n^{-1}\sum_{i=1}^{n}\left(1-y_i\langle \boldsymbol{P}_1\boldsymbol{C}\boldsymbol{P}_2^T,\boldsymbol{X}_i\rangle\right)_+ +\lambda\|\boldsymbol{P}_1\boldsymbol{C}\boldsymbol{P}_2^T\|_F^2. \tag{3}$$

Notice that the optimization problem (3) degenerates to the conventional SVM with vectorized feature matrices when feature mapping is identity and the coefficient is full rank. From the solution to (3), our estimated classifier is written

$$\hat{g}(\boldsymbol{X})=\operatorname{sign}(\hat{f}(\boldsymbol{X}))=\operatorname{sign}\left(\langle\hat{\boldsymbol{P}}_1\hat{\boldsymbol{C}}\hat{\boldsymbol{P}}_2^T,\boldsymbol{X}\rangle\right). \tag{4}$$

We make a remark on the implication of the formulation (4). The solution (4) implies a joint learning of dimension reduction and classification risk minimization. This is one of our contribution to combine two different processes into one. To check this, we see a dual representation of the solution to (3),

$$f(\boldsymbol{X})=\sum_{i=1}^{n}\alpha_i y_i\left(\sum_{j,k\in[d_1]}[\boldsymbol{H}_{\boldsymbol{P}_1}]_{jk},K_1(\boldsymbol{X}^{(i)}_{j:},\boldsymbol{X}_{k:})+\sum_{j,k\in[d_1]}[\boldsymbol{H}_{\boldsymbol{P}_2}]_{jk},K_2(\boldsymbol{X}^{(i)}_{:j},\boldsymbol{X}_{:k})\right), \tag{5}$$

where $\{\alpha_i\}_{i=1}^n$ are (sparse) dual solution to (3) and $\boldsymbol{X}^{(i)}_{jk}$ denotes (j,k)-entry of $\boldsymbol{X}_i$. Notice that the representation (5) can be viewed as an element of reproducing kernel Hilbert space (RKHS) induced by matrix kernel with weight matrices $\{\boldsymbol{H}_{\boldsymbol{P}_i}\}_{i=1,2}$ and row and column-wise kernels K_1,K_2. Therefore, our considered function space $\mathcal{F}$ can be written as

$$\begin{aligned}\mathcal{F}&=\{f\colon \boldsymbol{X}\mapsto\langle\boldsymbol{P}_1\boldsymbol{C}\boldsymbol{P}_2^T,\Phi(\boldsymbol{X})\rangle|\boldsymbol{C}\in(\mathcal{H}_1\times\mathcal{H}_2)^{d_1\times d_2}\text{ and }\boldsymbol{P}_i\in\mathbb{R}^{d_i\times r}\text{ for }i=1,2\}\\&=\{f\in\text{RKHS induced by matrix kernel }\boldsymbol{K}\text{ with }\{\boldsymbol{H}_{\boldsymbol{P}_i},K_i\}_{i=1,2}\}.\end{aligned}$$

Given row and column-wise kernels, we estimate weight matrices and an element of RKHS induced by the estimated weight matrices. The projection matrices $\{\boldsymbol{H}_{\boldsymbol{P}_i}\}_{i=1,2}$ play role in reducing the feature dimension and at the same time, we find the best element of RKHS that minimizes the classification risk by estimating coefficients $\boldsymbol{\alpha}$ in (5). The procedure is summarized as the following optimization

$$\max_{f\in\mathcal{F}}L(f)=\max_{\substack{\operatorname{rank}(\boldsymbol{W}_i)\le r,\\ \boldsymbol{W}_i\succeq 0,i=1,2}}\ \max_{\substack{f\in\text{RKHS}(\boldsymbol{K})\\ |\{\boldsymbol{W}_i,K_i\}_{i=1,2}}} L(f).$$

4.2. Main result II: Level set estimation in matrix space

We now consider multiple boundaries, for any $\pi \in [0,1]$. We propose weighted loss function from (2) to estimate the level set,

$$\min_{f\in\mathcal{F}} n^{-1}\sum_{i=1}^{n}\omega_\pi(y_i)L(y_i f(\boldsymbol{X}_i)) + \lambda J(f), \tag{6}$$

where $\omega_\pi(y) = 1-\pi$ if $y = 1$ and π if $y = -1$. The weighted loss accepts unequal costs for positive and negative misclassifications in margin classifier, where π is the known cost for the negative and $1-\pi$ is for the positive classes. Notice that equal cost $\pi = \frac{1}{2}$ make (6) reduce to (2). The optimizer to Equation (6) with respect to all measurable function class yields an consistent estimate of the Bayes rule $g_\pi(\boldsymbol{X}) = \text{sign}(f_\pi(\boldsymbol{X}))$ where $f_\pi(\boldsymbol{X}) = p(\boldsymbol{X}) - \pi$ (Lin et al., 2002; Wang et al., 2008). Therefore, under the considered decision function class, we obtain a minimizer $\hat{f}_\pi$ to (6) and estimate the level set as

$$\hat{S}(\pi) = \{\boldsymbol{X} \in \mathbb{R}^{d_1\times d_2} : \text{sign}(\hat{f}_\pi(\boldsymbol{X})) = 1\} \tag{7}$$

Assumption 1 (Identifiability)**.** *There exist constants $b > 0$ and $\alpha \geq 0$, such that, for any sufficient small $\delta > 0$,*

$$\mathbb{E}\left|\text{sign}f(\boldsymbol{X}) - \text{sign}f_{\text{bayes}}(\boldsymbol{X})\right| \leq b\left[R_\ell(f) - R_\ell(f_{\text{bayes}})\right]^\alpha$$

holds for all $f \in \{f \in \mathcal{F}(d,r,s) \colon R_\ell(f) - R_\ell(f_{\text{bayes}}) \leq \delta\}$ in a δ-neighborhood of f_{bayes}.
Theorem 4.1. *Assumption 1 holds with $\rho = \alpha\wedge 1$, and the estimation error for level sets satisfies*

$$\mathbb{P}(\hat{\boldsymbol{X}}\Delta\boldsymbol{X}_{bayes}) \leq C\left(\frac{rs\log d}{n} + a_n\right)^{\alpha/(2-\alpha\wedge 1)},$$

where $\hat{\boldsymbol{X}} = \{\boldsymbol{X}\colon \hat{f}(\boldsymbol{X}) \geq 0\}$ and $\boldsymbol{X}_{bayes} = \{\boldsymbol{X}\colon f_{\text{bayes}}(\boldsymbol{X}) \geq 0\}$ are estimated and true level sets, respectively.

More comments on the assumptions....

4.3. Main results III: Nonparametric regression function estimation

We propose a method to estimate the regression function $p(\boldsymbol{X}) \overset{\text{def}}{=} \mathbb{E}(y = 1|\boldsymbol{X})$ at any $\boldsymbol{X}$ which does not necessarily belong to the observed training data set. Non-smooth, non-continuous regression functions are allowed in our framework. Consider the following two steps of approximation to the target function.

$$\begin{aligned} p(\boldsymbol{X}) &\overset{\text{step1}}{\approx} \sum_{h=1}^{H}\frac{1}{H}\mathbb{1}\left\{\boldsymbol{X} : p(\boldsymbol{X}) \leq \frac{h}{H}\right\} \\ &= \sum_{h=1}^{H}\frac{1}{H}\mathbb{1}\left\{\boldsymbol{X} \notin S\left(\frac{h}{H}\right)\right\} \\ &\overset{\text{step2}}{\approx} \sum_{h=1}^{H}\frac{1}{H}\mathbb{1}\left\{\boldsymbol{X} \notin \hat{S}\left(\frac{h}{H}\right)\right\}. \end{aligned}$$

Step 1 approximates the target probability by linear combination of step functions where H is a smooth parameter. In step 2, we plug in the level set estimation defined in (7) given $\pi = h/H$. Here we use consistency of level set estimation. Therefore, we estimate the regression function as,

$$\hat{p}(\boldsymbol{X}) = \sum_{h=1}^{H} \frac{1}{H} \mathbb{1}\left\{\boldsymbol{X} \notin \hat{S}\left(\frac{h}{H}\right)\right\},$$

by repeatedly estimating the level sets as (7) with different π values, say $\pi = \frac{h}{H}$ for $h = 1, \ldots, H$.

Theorem 4.2 (Main result for probability estimation)**.** *Consider the same assumptions of Theorem* **??**. *For the estimator* $\hat{p}\colon \mathbb{R}^{d\times d} \to [0,1]$ *obtained from level-set estimation,*

$$\hat{p}(\boldsymbol{X}) = \frac{1}{H}\sum_{h=1}^{H} \mathbb{1}\{\boldsymbol{X}\colon \hat{f}_h(\boldsymbol{X}) \geq 0\}, \quad \textit{for all } \boldsymbol{X} \in \mathbb{R}^{d\times d}.$$

With probability at least $1 - C\exp(-an\lambda^{2-\alpha})$,

$$\mathbb{E}|\hat{p}(\boldsymbol{X}) - p(\boldsymbol{X})| \geq \underbrace{\frac{1}{2H}}_{\textit{discretization error}} + \frac{a(H+1)}{2} O\left(\underbrace{\frac{rs\log d}{n}}_{\textit{statistical error}} + \underbrace{a_n}_{\textit{approximation error}}\right)^{\alpha^2}.$$

Corollary 4.1. *Assume* $a_n \leq \frac{rs\log d}{n}$. *Choosing* $H \asymp (\frac{rs\log d}{n})^{-\alpha^2/2}$ *gives the estimation error,*

$$\mathbb{E}|\hat{p}(\boldsymbol{X}) - p(\boldsymbol{X})| \leq C\left(\frac{rs\log d}{n}\right)^{\rho/2}.$$

5. Two examples

The tensor regression provides a very general framework. By choosing a different distribution over predictor matrix, label ..., ..., we then obtain a different model. Several important examples are given below.

5.1. 1-bit matrix denoising

5.2. Multi-task learning

6. An ADMM algorithm for structural risk minimization

$$\min_{f\in\mathcal{F}} n^{-1}\sum_{i=1}^{n} L(y_i f(\boldsymbol{X}-i)) + \lambda J(f)$$

$$\min_{\boldsymbol{B}\in\mathcal{B}(r,s),\boldsymbol{c}\in\mathbb{R}^p} n^{-1}\sum_{i=1}^{n} L\left(y_i\left[\boldsymbol{w}_i^T\boldsymbol{c} + \langle \boldsymbol{X}_i, \boldsymbol{B}\rangle\right]\right) + \lambda\|\boldsymbol{B}\|_F^2$$

We introduce ADMM argument $\boldsymbol{B} = \boldsymbol{P}\boldsymbol{Q}^T$, where $\boldsymbol{P}, \boldsymbol{Q} \in \mathbb{R}^{d\times r}$ and penalization ρ. Given ρ and λ, we solve

$$\mathcal{L}(\boldsymbol{B},\boldsymbol{S},\boldsymbol{c},\boldsymbol{\Lambda};\rho,\lambda) = n^{-1}\sum_{i=1}^{n} L\left(y_i\left[\boldsymbol{w}_i^T\boldsymbol{c} + \langle \boldsymbol{X}_i,\boldsymbol{B}\rangle\right]\right) + \lambda\|\boldsymbol{B}\|_F^2 + \rho\|\boldsymbol{B}-\boldsymbol{S}\|_F^2 + \langle \boldsymbol{\Lambda}, \boldsymbol{B}-\boldsymbol{S}\rangle.$$

(a) Update $\boldsymbol{B}$:

$$\mathcal{L}(\boldsymbol{B},\boldsymbol{c};\boldsymbol{S},\Lambda,\rho,\lambda) = n^{-1}\sum_{i=1}^{n} L\left(y_i\left[\boldsymbol{w}_i^T\boldsymbol{c} + \langle \boldsymbol{X}_i,\boldsymbol{B}\rangle\right]\right) + (\lambda+\rho)\left\|\boldsymbol{B} - \frac{1}{2(\lambda+\rho)}(2\rho\boldsymbol{S}-\boldsymbol{\Lambda})\right\|_F^2.$$

Equivalently, define $\check{\boldsymbol{B}} = \boldsymbol{B} - \frac{1}{2(\lambda+\rho)}(2\rho\boldsymbol{S}-\boldsymbol{\Lambda})$, we have

$$\mathcal{L}(\check{\boldsymbol{B}},\boldsymbol{c};\boldsymbol{S},\Lambda,\rho,\lambda) = n^{-1}\sum_{i=1}^{n} L\left(y_i\left[\left\langle \boldsymbol{X}_i, \frac{2\rho\boldsymbol{S}-\boldsymbol{\Lambda}}{2(\lambda+\rho)}\right\rangle + \boldsymbol{w}_i^T\boldsymbol{c} + \langle \boldsymbol{X}_i,\check{\boldsymbol{B}}\rangle\right]\right) + (\lambda+\rho)\|\check{\boldsymbol{B}}\|_F^2.$$

(b) Update $\boldsymbol{c}$:

$$\mathcal{L}(\boldsymbol{c};\boldsymbol{B},\boldsymbol{S},\Lambda,\rho,\lambda) = n^{-1}\sum_{i=1}^{n} L\left(y_i\left[\boldsymbol{w}_i^T\boldsymbol{c} + \langle \boldsymbol{X}_i,\boldsymbol{B}\rangle\right]\right)$$

(c) Update $\boldsymbol{S}$:

$$\mathcal{L}(\boldsymbol{S};\boldsymbol{c},\boldsymbol{B},\Lambda,\rho,\lambda) = \left\|\boldsymbol{S} - \frac{2\rho\boldsymbol{B}+\boldsymbol{\Lambda}}{2\rho}\right\|_F^2, \quad \text{where} \quad \boldsymbol{S}\in\mathcal{B}(r,s).$$

(d) Update $\boldsymbol{\Lambda}$:

$$\boldsymbol{\Lambda}^{(t+1)} = \boldsymbol{\Lambda}^{(t)} + 2\rho(\boldsymbol{B}-\boldsymbol{S}).$$

Ideally, small ρ = ρ.

In this Section, we describe an algorithm to seek the optimizer of Equation (2) in the case of hinge loss function $L(z) = (1-z)_+$. We consider nonlinear decision function class $\mathcal{F} = \{f\colon \boldsymbol{X} \mapsto \langle \boldsymbol{C}\boldsymbol{P}^T, \Phi(\boldsymbol{X})\rangle | \boldsymbol{C} = (\boldsymbol{C}_1,\boldsymbol{C}_2) \in \mathcal{H}_1^{d_1}\times\mathcal{H}_2^{d_2}$ and $\boldsymbol{P} = (\boldsymbol{P}_1,\boldsymbol{P}_2) \in \mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}\}$ given row and columnwise kernels K_1, K_2. Notice Equation (3) is written as

$$\min_{\substack{\boldsymbol{C}\in\mathcal{H}_1^{d_1}\times\mathcal{H}_2^{d_2},\\ \boldsymbol{P}\in\mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}}} \frac{1}{2}\|\boldsymbol{C}\boldsymbol{P}^T\|_F^2 + C\sum_{i=1}^{n}\xi_i, \tag{8}$$
$$\text{subject to } y_i\langle \boldsymbol{C}\boldsymbol{P}^T, \Phi(\boldsymbol{X}_i)\rangle \le 1-\xi_i \text{ and } \xi_i \ge 0, i=1,\dots,n.$$

Optimization problem (8) is non-convex problem because low-rank constraint makes feasible set non-convex. We propose to utilize coordinate descent algorithm that solves one block holding the other block fixed. From this approach, we can solve a convex problem in each step. To be specific, first we update $\boldsymbol{C}$ holding $\boldsymbol{P}$ fixed. The dual problem of Equation (8) with fixed $\boldsymbol{P}$ is

$$\max_{\boldsymbol{\alpha}=(\alpha_1,\dots,\alpha_n)} -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \Phi(\boldsymbol{X}_i)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T, \Phi(\boldsymbol{X}_j)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\rangle$$
$$\text{subject to } 0 \le \alpha_i \le C, i=1,\dots,n.$$

We use quadratic programming to solve the dual problem and update $\boldsymbol{C}$ as

$$\boldsymbol{C} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\boldsymbol{X}_i)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1} \in \mathcal{H}_r^r \times \mathcal{H}_c^r. \tag{9}$$

We use the formula (9) without information about feature mapping $\Phi(\cdot)$. Second, we assume that $\boldsymbol{C}$ is fixed and update $\boldsymbol{P}$. The dual problem of Equation (8) with fixed $\boldsymbol{C}$ is

$$\max_{\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)} -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_i)\right), \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_j)\right)\rangle, \tag{10}$$
$$\text{subject to } 0 \le \alpha_i \le C, i = 1,\ldots,n,$$

We can find an optimizer of (10) based on kernel information only. We obtain the following formula by plugging (9) into components of (10).

$$\boldsymbol{C}^T\boldsymbol{C} = \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{K}(i,j)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1} \in \mathbb{R}^{r\times r}\times\mathbb{R}^{r\times r},$$
$$\boldsymbol{C}^T\Phi(\boldsymbol{X}_i) = \sum_{j=1}^{n}\alpha_i y_i (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{K}(i,j) \in \mathbb{R}^{r\times d_1}\times\mathbb{R}^{r\times d_2},$$

where $\boldsymbol{K}(i,j) \overset{\text{def}}{=} \left(\Phi_1(\boldsymbol{X}_i)^T\Phi_1(\boldsymbol{X}_j), \Phi_2(\boldsymbol{X}_i)^T\Phi_2(\boldsymbol{X}_j)\right) \in \mathbb{R}^{d_1\times d_1}\times\mathbb{R}^{d_2\times d_2}$. Notice that $[\Phi_1(\boldsymbol{X}_i)^T\Phi_1(\boldsymbol{X}_j)]_{ss'} = K_1(\boldsymbol{X}_{s:}^{(i)}, \boldsymbol{X}_{s':}^{(j)})$ and vice versa for $\Phi_2(\cdot)$. Therefore, we update $\boldsymbol{P}$ from an optimal coefficient $\boldsymbol{\alpha}$ to (10) without specifying feature mapping.

$$\boldsymbol{P} = \sum_{i=1}^{n}\alpha_i y_i (\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_i).$$

We end up obtaining nonlinear function output of the form,

$$\hat{f}(\boldsymbol{X}) = \sum_{k=1}^{n}\hat{\alpha}_k y_k \Bigg(\sum_{i=1}^{d_1}\sum_{j=1}^{d_1}[\hat{\boldsymbol{P}}_1(\hat{\boldsymbol{P}}_1^T\hat{\boldsymbol{P}}_1)^{-1}\hat{\boldsymbol{P}}_1^T]_{ij}K_r([\boldsymbol{X}_k]_{i:}, [\boldsymbol{X}]_{j:}) \tag{11}$$
$$+ \sum_{i=1}^{d_2}\sum_{j=1}^{d_2}[\hat{\boldsymbol{P}}_2(\hat{\boldsymbol{P}}_2^T\hat{\boldsymbol{P}}_2)^{-1}\hat{\boldsymbol{P}}_2^T]_{ij}K_c([\boldsymbol{X}_k]_{:i}, [\boldsymbol{X}]_{:j}) \Bigg).$$

Algorithm 1 gives the full description for classification.

By the similar way with little modification, we can obtain an algorithm for weighted margin classifier (6). From the explanation in Section 4.2 and 4.3, we summarize level set and regression estimation procedure in Algorithm 2.

7. Extension to nonlinear boundaries

We propose a family of matrix kernels, which are building blocks for defining functions in matrix space. Kernel methods defined on non-vector objects have recently evolved into a rapidly

Algorithm 1: Classification algorithm

```
Input: $(\boldsymbol{X}_1, y_1), \cdots, (\boldsymbol{X}_n, y_m)$, rank $r$, and pre-specified kernels $K_1, K_2$
Initizlize: $\boldsymbol{P}^{(0)} \in \mathbb{R}^{d_1 \times r} \times \mathbb{R}^{d_2 \times r}$
Do until converges
    Update $\boldsymbol{C}$ fixing $\boldsymbol{P}$ :
    Solve $\max_{\boldsymbol{\alpha}} -\sum_{i=1}^n \alpha_i + \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j y_i y_j \langle \Phi(\boldsymbol{X}_i), \Phi(\boldsymbol{X}_j)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T \rangle$
    $\boldsymbol{C} = \sum_{i=1}^n \alpha_i y_i \Phi(\boldsymbol{X}_i)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}$.
    Update $\boldsymbol{P}$ fixing $\boldsymbol{C}$ :
    Solve $\max_{\boldsymbol{\alpha}} -\sum_{i=1}^n \alpha_i + \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j y_i y_j \langle \Phi(\boldsymbol{X}_i), \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_j)\right)\rangle$.
    $\boldsymbol{P} = \sum_{i=1}^n \alpha_i y_i (\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_i)$.
Output: $\hat{f}$ of the form (11)
```

Algorithm 2: Level set & Regression Algorithm

```
Input: $(\boldsymbol{X}_1, y_1), \cdots, (\boldsymbol{X}_n, y_m)$, rank $r$,pre-specified kernels $K_1, K_2$, and smooth parameter
    $H$.
Initialize: $\pi_h = (h-1)/H$ for $h = 1, \ldots, H+1$
For $h = 1, \ldots, H+1$:
    Level set $\hat{S}(\pi_h)$ estimation:
        Train weighted margin classifier $\hat{f}_{\pi_h}$ from (6) based on Algorithm 1.
        $\hat{S}(\pi_h) = \{\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2} : \text{sign}(\hat{f}_{\pi_h}(\boldsymbol{X})) = 1\}$.
    Regression $\hat{p}(\boldsymbol{X})$ estimation:
        $\hat{p}(\boldsymbol{X}) = \sum_{h=1}^H \frac{1}{H}\mathbb{1}\left\{\boldsymbol{X} \notin \hat{S}(\pi_h)\right\}$.
Output: Level sets $\hat{S}(\pi_h)$ for $h = 1, \ldots H$ and regression function $\hat{p}(\boldsymbol{X})$.
```

developing branch of learning on structured data. Informally, a matrix kernel is a distance measure between two matrices with the same size using proper notion of similarity. Unlike vectors, matrix inputs represent two-way relationship across rows and columns at a time. Taking into account of this two-way relationship is essential in the kernel development. Our proposed kernel uses concept from latent factor models and incorporates the two-way similarities via low rank regularization. We generalize classical kernel method in vector spaces to matrix spaces. We briefly summarize kernel method for vector features and introduce new notations and operations which will be used later.

It has been popular and successful to extend linear classifiers in vector space to nonlinear classifiers using kernel method. Classical linear classifier finds linear boundaries in the input vector feature space. By introducing feature mapping which maps input feature space to enlarged dimension space, learning nonlinear classifier becomes possible. In fact, we need not specify the feature mapping at all to obtain an optimal function that minimizes pre-determined loss function. Instead, the learning process only requires knowledge of the kernel function that computes inner products in the transformed enlarged space, thereby avoiding heavy computation. We generalize this kernel approach to the case when input feature is matrix valued.

Before proposing a new matrix feature mapping and kernel, we introduce notations and operations needed later. Let $\phi_i\colon \mathbb{R}^{d_i} \to \mathcal{H}_i$ be feature mappings with a classical kernel defined on vectors $K_i\colon \mathbb{R}^{d_i} \times \mathbb{R}^{d_i} \to \mathbb{R}$ for $i = 1,2$. $\mathcal{H}_i$ denotes enlarged feature space by ϕ_i and a possibly infinite dimensional Hilbert space. Let $\mathcal{H}^{d_1\times d_2} = \{\boldsymbol{X}\colon \boldsymbol{X} = [\![x_{ij}]\!], x_{ij} \in \mathcal{H}\}$ denote the collection of d_1 by d_2 matrices with each entry taking value in a Hilbert space $\mathcal{H}$. Matrix algebraic operations are carried over from operations on real valued matrices. One can check exact definitions of operations in Supplement.

Now we present matrix the matrix kernel and associated feature mapping. We define matrix valued feature mapping over the d_1-by-d_2 matrix space.

Definition 1. *Let $\phi_1\colon \mathbb{R}^{d_1} \to \mathcal{H}_1$ and $\phi_2\colon \mathbb{R}^{d_2} \to \mathcal{H}_2$ be classical feature mappings defined on vector space. Then Φ is matrix feature mappings defined on*

$$\begin{aligned}\Phi(\boldsymbol{X})\colon \mathbb{R}^{d_1\times d_2} &\to (\mathcal{H}_1 \times \mathcal{H}_2)^{d_1\times d_2} \\ \boldsymbol{X} &\mapsto \Phi(\boldsymbol{X}) = [\![\Phi(\boldsymbol{X})_{ij}]\!] \textit{ where } \Phi(\boldsymbol{X})_{ij} \overset{def}{=} (\phi_2(\boldsymbol{X}_{i:}), \phi_1(\boldsymbol{X}_{:j}))\,.\end{aligned} \tag{12}$$

Notice that the matrix feature mapping considers both row-wise and column-wise enlarged features. From the feature mapping, the linear function $f\colon \mathbb{R}^{d_1\times d_2} \to \mathbb{R}$ with respect to enlarged space $\Phi(\boldsymbol{X}) \in (\mathcal{H}_1 \times \mathcal{H}_2)^{d_1\times d_2}$ is defined as,

$$\begin{aligned} f(\boldsymbol{X}) \overset{def}{=} \langle \boldsymbol{B}, \Phi(\boldsymbol{X})\rangle, \textit{ where } \boldsymbol{B} = [\![(\boldsymbol{b}_i^{row}, \boldsymbol{b}_j^{col})]\!] \in (\mathcal{H}_1 \times \mathcal{H}_2)^{d_1\times d_2}, \\ \textit{with } \boldsymbol{b}_i^{row} \in \mathcal{H}_1 \textit{ and } \boldsymbol{b}_j^{col} \in \mathcal{H}_2 \textit{ for all } (i,j) \in [d_1]\times[d_2].\end{aligned} \tag{13}$$

Notice we impose this structure on $\boldsymbol{B}$ for identifiability issue. These matrix valued feature mapping (12) and corresponding linear function (13) are generalization from existing classical kernel method in vector spaces and can be extended naturally to tensor case (see Supplement for the details). We assume that the coefficient $\boldsymbol{B}$ in (13) admits low rank decomposition as in Section **??**,

$$\begin{aligned}\boldsymbol{B} = \boldsymbol{P}_1\boldsymbol{C}\boldsymbol{P}_2^T, \text{ where } \boldsymbol{P}_1 \in \mathbb{R}^{d_1\times r}, \boldsymbol{P}_2 \in \mathbb{R}^{d_2\times r} \text{ and } \boldsymbol{C} = [\![(\boldsymbol{c}_i^{\text{row}}, \boldsymbol{c}_j^{\text{col}})]\!] \in (\mathcal{H}_1 \times \mathcal{H}_2)^{r\times r}, \\ \text{with } \boldsymbol{c}_i^{\text{row}} \in \mathcal{H}_1 \text{ and } \boldsymbol{c}_j^{\text{col}} \in \mathcal{H}_2 \text{ for all } i,j \in [r].\end{aligned} \tag{14}$$

Again, we have the structured $\boldsymbol{C}$ for identifiability. When feature mapping ϕ_i is identity for $i = 1,2$ implying the linear case in Section **??**, we show that considered linear functions (13) with low-rank r defined by (14) are equivalent to the linear functions in Section **??** with the low-rank r constraint (1). Therefore, our matrix feature mapping is generalization of classical feature mapping on vector spaces and extension to nonlinear case from linear functions on matrix features.

Now we define matrix kernel associated with the matrix feature mapping.

Definition 2. *Let $K_i(\cdot,\cdot)$ be classical kernels which can be represented as $K_i(\cdot,\cdot) = \langle \phi_i(\cdot), \phi_i(\cdot)\rangle$ for $i = 1,2$. Let weight matrices $\boldsymbol{W}_i = [\![w_{jk}^{(i)}]\!] \in \mathbb{R}^{d_i\times d_i}$ be rank-r semi-positive definite matrices for $i = 1,2$. Then $\{\boldsymbol{W}_i, K_i\}_{i=1,2}$ induce matrix kernel defined by*

$$\begin{aligned}\boldsymbol{K}\colon \mathbb{R}^{d_1\times d_2} \times \mathbb{R}^{d_1\times d_2} &\to \mathbb{R} \\ (\boldsymbol{X}, \boldsymbol{X}') &\mapsto \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{j,k\in[d_1]} w_{jk}^{(1)} K_1(\boldsymbol{X}_{j:}, \boldsymbol{X}'_{k:}) + \sum_{j,k\in[d_2]} w_{jk}^{(2)} K_2(\boldsymbol{X}_{:j}, \boldsymbol{X}'_{:k}).\end{aligned}$$

The matrix kernel incorporates classical kernel in vector spaces. Like classical kernel, we can associate the feature mapping in Definition 1 with the matrix kernel. Given $\{\boldsymbol{W}_i, K_i\}_{i=1,2}$, we have

$$\begin{aligned}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}') &= \sum_{j,k\in[d_1]} w_{jk}^{(1)} K_1(\boldsymbol{X}_{j:},\boldsymbol{X}'_{k:}) + \sum_{j,k\in[d_2]} w_{jk}^{(2)} K_2(\boldsymbol{X}_{:j},\boldsymbol{X}'_{:k}) \\ &= \langle(\boldsymbol{W}_1,\boldsymbol{W}_2)\Phi(\boldsymbol{X}),(\boldsymbol{W}_1,\boldsymbol{W}_2)\Phi(\boldsymbol{X}')\rangle.\end{aligned}$$

We can view the matrix kernel as weighted inner product of the feature mappings. From the kernel representation, we learn nonlinear function successfully avoiding specification of feature mapping $\Phi(\boldsymbol{X})$ as in classical vector case given pre-specified row and column-wise kernels K_1, K_2.

8. Numerical experiments

Acknowledgements

References

Lin, Y., Y. Lee, and G. Wahba (2002). Support vector machines for classification in nonstandard situations. *Machine learning 46*(1-3), 191–202.

Luo, L., Y. Xie, Z. Zhang, and W.-J. Li (2015). Support matrix machines. In *International conference on machine learning*, pp. 938–947.

Man, T.-K., M. Chintagumpala, J. Visvanathan, J. Shen, L. Perlaky, J. Hicks, M. Johnson, N. Davino, J. Murray, L. Helman, et al. (2005). Expression profiles of osteosarcoma that can predict response to chemotherapy. *Cancer research 65*(18), 8142–8150.

Pirsiavash, H., D. Ramanan, and C. C. Fowlkes (2009). Bilinear classifiers for visual recognition. In *Advances in neural information processing systems*, pp. 1482–1490.

Shashua, A., Y. Gdalyahu, and G. Hayun (2004). Pedestrian detection for driving assistance systems: single-frame classification and system level performance. *IEEE Intelligent Vehicles Symposium, 2004*, 1–6.

Wang, J., X. Shen, and Y. Liu (2008). Probability estimation for large-margin classifiers. *Biometrika 95*(1), 149–167.

Zhou, H. and L. Li (2014). Regularized matrix regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76*(2), 463–483.