Nonparametric learning with matrix-valued predictors in high dimensions

Abstract

We consider the problem of learning the relationship between a binary label response and a high-dimensional matrix-valued predictor. Such data problems arise commonly in brain imaging studies, sensor network localization, and personalized medicine. Existing regression analysis often takes a parametric procedure by imposing a pre-specified relationship between variables. However, parametric models are insufficient in capturing complex regression surfaces defined over high-dimensional matrix space. Here, we propose a flexible nonparametric framework for various learning tasks, including classification, level set estimation, and regression, that specifically accounts for the matrix structure in the predictors. Unlike classical approaches, our method adapts to the possibly non-smooth, non-linear pattern in the regression function of interest. The proposal achieves prediction and interpretability simultaneously via a joint optimization of prediction rules and dimension reduction in the matrix 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 XXX data analyses.

Keywords: Nonparametric learning, matrix-valued predictors, high dimension, classification, level-set estimation, regression.

1 Introduction

2 Introduction (skipped)

Consider a statistical 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 $Y \in \{-1, 1\}$. 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 (**?**). 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 (**?**).

Our motivating problem for learning with matrix-valued predictors comes from brain network analysis. Label prediction based on networks is an important problem in neuroscience. In these studies, each individual in the sample is represented by their own brain network, and the nodes (brain regions of interest) shared across all networks are mapped onto a common atlas. In our analysis of human connectom project, 68 brain nodes are extracted from six functional brain regions. A connectivity is computed for every pair of nodes, resulting in an adjacency matrix of size 68×68 for each individual (see Figure 1). The main scientific goal is to predict the individual's disease status based on the brain connectivity network. Two main approaches have been used in the existing literature. One approach is to reduce the network to its global summary measures such as the average degree, clustering coefficient, or average path length, and use those measures as features

for disease predicting. An alternative approach to label predicting is to treat connectivity network as a "bag of non-ordered edges," which essentially vectorized the adjacency matrix and ignoring the network nature of the predictor. This approach suffers from ....The number of individual ..., much smaller than the predictor dimension...

In the above 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, $d_1 d_2$, 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. We use structured sparsity penalties to incorporate the network information by penalizing both the number of simple modules and the number of nodes selected....

The classification problem has long been interested. Many attempts have been developed and performed well for example, decision tree, nearest neighbor, neural network and support vector machine to name a few. However, most of methods have focused on vector valued features. In many classification problems, the input features are naturally

represented as matrices or tensors rather than vectors.

Knowledge about the class probability itself is of significant interest and can tell us the confidence of the outcome of classification. Traditionally, the regression problem is addressed through distribution assumption like logistic regression or linear discriminant analysis (LDA). In many applications, however, it is often difficult to justify the assumptions made in logistic regression or satisfy the Gaussian assumption in LDA. These issues become more challenging for matrix features because of high dimensionality. We establish distribution free method for estimating the regression function based on level set estimation.

Other examples can be found in medical decision making. In Osteosarcoma treatment, the degree of tumor necrosis is used to guide the choice of postoperative chemotherapy (**?**). Patients with $\geq 90\%$ necrosis is labeled as 1, which is response variable y. Suppose that $\boldsymbol{X}$ is a feature matrix collected from the patient such as gene expression levels on each tissue. Knowledge of the regression level set is needed to allow effective postoperative chemotherapy without a biopsy. We consider a nonparametric way to estimator the π-level set of the regression function based on classification problem.

Notation: For a given set $A \subset \mathbb{R}^{d_1 \times d_2}$, we use $\mathbb{1}\{A\} \in \{0, 1\}$ to denote the indicator function, and $I(A) = \{-1, 1\}$ to denote the (shifted) indicator function. A set A uniquely defines an indicator function $I(\boldsymbol{X} \in A)$ from $\mathbb{R}^{d_1 \times d_2} \to \{-1, 1\}$. We also use $(f) \in \{-1, 1\}$ to denote the sign of function f; that is, $f = 1$ if $f > 0$ and $f = -1$ if $f \leq 0$. We restrict to Borel-measurable functions and L2? Do we require $\{\boldsymbol{X} : p(\boldsymbol{X}) = 1/2\}$ to have measure zero?? We use $\mathcal{MN}(\boldsymbol{B}, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ to denote the matrix normal distribution, meaning $\text{vec}(\boldsymbol{X}) \sim \mathcal{N}(\text{vec}(\boldsymbol{B}), \boldsymbol{\Sigma}_1 \otimes \boldsymbol{\Sigma}_1)$. Given a d_1-by-d_2 matrix $\boldsymbol{B}$, we use $\boldsymbol{B}_i$ to denote the i-th row of $\boldsymbol{B}$, where $i \in [d_1]$. We use $\cdot_p$ to denote the p-norm for vectors for $p \geq 0$. The (p, q)-norm of a matrix $\boldsymbol{B}$ is defined as $\boldsymbol{B}_{p,q} = \boldsymbol{b}_q$, where $\boldsymbol{b} = (\boldsymbol{B}_{1p}, \ldots, \boldsymbol{B}_{d_1 p}) \in \mathbb{R}^{d_1}$ consists

of the p-norms for each of the rows in $\boldsymbol{B}$. In particular, $\boldsymbol{B}_{1,0} = \#\{i \in [d_1] \colon \boldsymbol{B}_i \neq 0\}$ denotes the number of non-zero rows in $\boldsymbol{B}$, and ${\boldsymbol{B}^T}_{1,0}$ denotes the number of non-zero columns in $\boldsymbol{B}$.

Need consistent notation $\mathbb{P}$, $\mathbb{P}_{\boldsymbol{X},y}$, $\mathbb{P}_{\boldsymbol{X}}$.

3 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, $y \in \{0, 1\}$ 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.

We consider three main supervised learning problems: classification, level set estimation, and regression.

2.1 *Classification*: Classification is the problem of predicting the label $y \in \{-1, 1\}$ to which a new matrix observation $\boldsymbol{X}$ belongs. A prediction rule (also called a classifier) decides that $y = 1$ if $\boldsymbol{X} \in S$ and $y = 0$ if $\boldsymbol{X} \notin S$, where S is a Borel subset of $\mathbb{R}^{d_1 \times d_2}$. The set S uniquely defines an indicator function, $I(\boldsymbol{X} \in S)$, in the matrix space. With a little abuse of notations, we will use the term "classifier" to refer to as both the set S and its associated indicator function $I(\boldsymbol{X} \in S)$. We formulate the classification problem as

choosing a classifier that minimizes the expected classification error,

$$R(S) = \mathbb{P}\left[y \neq I(\boldsymbol{X} \in S)\right], \quad \text{for all } S \in \mathcal{S}, \tag{1}$$

where $\mathcal{S}$ is a given set of candidate classifiers, and the probability is taken over the pair $(\boldsymbol{X}, y)$ according to the joint distribution $\mathbb{P}_{\boldsymbol{X},y}$. The expected classification error $R(S)$ is also referred to as the classification risk. When the candidate set $\mathcal{S}$ consists of all Borel subsets of $\mathbb{R}^{d_1 \times d_2}$, the minimizer of (1) is called the Bayes classifier. It is known that (one of) the Bayes classifier can be written as

$$\{\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2} \colon p(\boldsymbol{X}) \geq 1/2\}, \tag{2}$$

where $p(\boldsymbol{X}) = \mathbb{P}(y = 1|\boldsymbol{X})$ is the conditional probability function defined in matrix space. In principle, Bayes classifier is non-unique because of the arbitrary prediction rules allowed on the boundaries $\partial S_{\text{bayes}} = \{\boldsymbol{X} \colon p(\boldsymbol{X}) = 1/2\}$. Without loss of generality, we will use (2) as the canonical form of Bayes classifier.

In practice the population joint distribution $\mathbb{P}_{\boldsymbol{X},y}$ is unknown, so the objective function (1) and the minimizer (2) need to be estimated through the data $(\boldsymbol{X}_i, y_i)_{i=1}^n$. The first goal of our paper is to estimate the Bayes classifier for matrix classification:

Question 1. When matrix dimension $d_1 d_2$ far exceeds the sample size n, how to efficiently perform matrix classification without much assumptions on $\mathbb{P}_{\boldsymbol{X},y}$?

2.2 *Level set estimation*: The problem of level set estimation generalizes the classification task. For a given $\pi \in [0, 1]$, the π-level set of the conditional probability function $p(\boldsymbol{X})$ is defined by

$$(\pi) = \{\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2} \colon p(\boldsymbol{X}) \geq \pi\}. \tag{3}$$

An important characterization of π-level set is that it is the optimal weighted classifier (**???**). Specifically, among all Borel subsets of $\mathbb{R}^{d_1 \times d_2}$, $S_{\text{bayes}}(\pi)$ minimizes the expected π-weighted classification error,

$$R_\pi(S) = \mathbb{E}\left[w_\pi(y)\mathbb{1}(y \neq I(\boldsymbol{X} \in S))\right], \tag{4}$$

where the expectation is taken jointly over $(\boldsymbol{X}, y)$, and we define $w_\pi(y) = 1 - \pi$ if $y = 1$ and $w_\pi(y) = \pi$ if $y = -1$; that is, $w_\pi(\cdot)$ assigns possibly unequal weights for the two labels. In light of (1) and (3), the level set estimation problem is an extension to π-weighted classification of the usual classification with $\pi = 1/2$. In particular, the π-level set $S_{\text{bayes}}(\pi)$ serves the role of Bayes classifier in the risk minimization. Accurate level set estimation plays an important role in applications of geographical elevation maps, imaging contour detection, and motion tracking. We consider the following question in our context:

Question 2. How to simultaneously estimate the level set and identify important variables in the matrix-valued predictors $\boldsymbol{X}$, for the goal of interpretable prediction?

2.3 *Nonparametric regression*: The problem of nonparametric regression is to estimate the conditional mean $\mathbb{E}(y|\boldsymbol{X})$ as a continuous-valued function defined in the matrix space. In the context of binary response with $\{-1, 1\}$ labels, estimating regression is equivalent to estimating conditional probability $p(\boldsymbol{X}) = \mathbb{P}(y = 1|\boldsymbol{X}) = 2\mathbb{E}(y|\boldsymbol{X}) - 1$. Throughout the paper we will work with $p(\boldsymbol{X})$ and refer to it as the regression function. The function $p(\boldsymbol{X})$, among all measurable functions $f\colon \mathbb{R}^{d_1 \times d_2} \to [0, 1]$, minimizes the expected squared error (also known as regression risk),

$$R_{\text{reg}}(f) = \mathbb{E}\left[y + 1 - 2f(\boldsymbol{X})\right]^2, \tag{5}$$

where the expectation is taken jointly over $(\boldsymbol{X}, y)$. Our final goal is the function estimation:

Question 3. How to achieve accurate and interpretable learning of the regression $p(\boldsymbol{X})$ as a function in the high-dimensional matrix space $\mathbb{R}^{d_1 \times d_2}$?

Matrix-valued predictors impose unique challenges to the aforementioned three problems. We consider nonparametric learning without imposing particular function forms of $\mathbb{P}_{\boldsymbol{X},y}$, and we allow matrix dimension $d_1 d_2$ to grow sub-exponentially with the sample size n. This scenario is important yet notably hard for three reasons. First, matrix-valued predictors represent various aspects of data features including global structure (e.g. clustering patterns, community hubs) and local structure (e.g. node degrees, edge connection). While classical learning algorithms have been successful with multivariate predictors invariant to indexing, challenge arises with structured predictors represented by matrices. Second, nonparametric methods typically rely on some notion of local smoothness in the predictor domain. Extracting useful neighborhood structure in the matrix-valued predictor domain has yet to explore. Third, matrix-based prediction often lead to high dimension low sample size problems. This setting brings challenges to interpretable prediction. Unlike parametric models, non-parametric learning makes little assumptions on the variable relationships, which adds complexity to interpretation. Achieving accurate prediction while maintaining descriptive simplicity is therefore our main goal.

4 From classification to regression: a new deal

The three problems of our interest represent a range of learning tasks with increasing difficulties. Classification is a special case of level set estimation with $\pi = 1/2$, whereas the level set is a discrete approximation of the regression function. A common approach is to address regression first, and then solve the earlier two using plug-in estimators (Figure 1a).

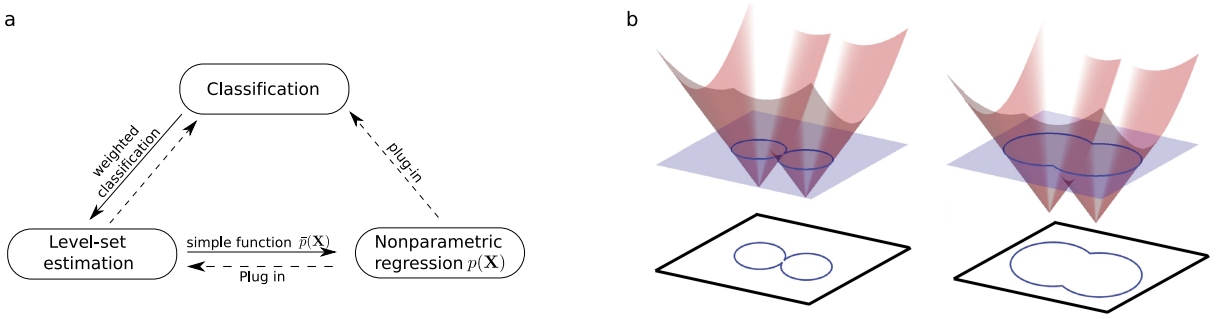

Figure 1: (a) Our learning reduction approach (solid line) to three learning problems and classical plug-in approaches (dashed line). (b) Schematic diagram for nonparametric function estimation via level set estimation (Figure modified from **?**).

This procedure, however, undermines the fact that regression is generally harder than the other two. Indeed, as we show in Section 5, regression has a slower convergence rate $\mathcal{O}(n^{-1/2})$ compared to the fast rate $\mathcal{O}(n^{-1})$ of classification. Ignorance of the increased complexity violates Vapnik's maxim: *When solving a given problem, one should try to avoid solving a more general problem as an intermediate step.*

Following Vapnik's principle, we develop a "learning reduction" approach by relating the regression problems to classifications, which are more fundamental and easier to address. In this section we restrict our attention to the population properties of regression function $p(\boldsymbol{X})$ when the true distribution $\mathbb{P}_{\boldsymbol{X},y}$ is known. This ideal situation leads to a cleaner characterization with known (deterministic) objective functions in (1), (3), and (2.3). The finite sample estimation will be presented in Section 5, in which we address the general case with unknown distribution $\mathbb{P}_{\boldsymbol{X},y}$, and the only information is through empirical (stochastic) risk estimated from the training set $(\boldsymbol{X}_i, y_i)_{i=1}^n$.

4.1 Level set approaches to nonparametric matrix regression

Our building block is to use level sets to estimate regression function $p(\boldsymbol{X})$ through classifications. The level set approach bridges the two sides of a same coin – characteristic (set indicator) functions in functional analysis and weighted classifications in statistical learning – in order to take the best of the two worlds. Specifically, let $p(\boldsymbol{X})\colon \mathbb{R}^{d_1\times d_2}\to[0,1]$ be the target regression function of interest. Let $\Pi=\{1/H,2/H,\ldots,(H-1)/H\}$ be a sequence of evenly spaced points in $[0,1]$, where $H\in\mathbb{N}_+$ is the smoothing parameter. We introduce an H-step function from $\mathbb{R}^{d_1\times d_2}$ to $[0,1]$,

$$\bar{p}(\boldsymbol{X})=\frac{1}{2H}\sum_{\pi\in\Pi}I(\boldsymbol{X}\in\bar{S}(\pi))+\frac{1}{2}, \tag{6}$$

where, for every $\pi\in\Pi$, the set $\bar{S}(\pi)\subset\mathbb{R}^{d_1\times d_2}$ is the classifier that minimizes the π-weighted classification risk (3) $R_\pi(S)$,

$$\bar{S}(\pi)\stackrel{\text{def}}{=}\arg\min_{S\in\mathcal{S}}\mathbb{E}\left[w_\pi(y)\mathbb{1}(y\neq I(\boldsymbol{X}\in S))\right], \tag{7}$$

subject to the constraint $S\in\mathcal{S}$, with $\mathcal{S}$ being a given candidate set of classifiers. When the set $\mathcal{S}$ is rich enough, e.g., $\mathcal{S}$ consists of all Borel sets, then $\bar{S}(\pi)$ reduces to $S_{\text{bayes}}(\pi)$ in (2.2). We leave the $\mathcal{S}$ in general here; the specific choice of $\mathcal{S}$ will be described in Section 4.2.

The following assumption quantifies the identification of level sets from weighted classification.

Assumption 1 (Identifiability)**.** *For a given $\pi\in(0,1)$, we call a level set (π) is identifiable from weighted classification, if there exist constants $\alpha\in(0,1]$ and $C>0$, such that*

$$\mathbb{P}_{\boldsymbol{X}}\left[S\Delta(\pi)\right]\leq C\left[R_\pi(S)-R_\pi((\pi))\right]^\alpha,\quad \textit{for all } S\subset\mathbb{R}^{d_1\times d_2} \tag{8}$$

where $S\Delta S'=\{\boldsymbol{X}\colon \boldsymbol{X}\in S/S' \text{ or } S'/S\}$ denotes the set difference between S and S'. The largest possible value of α in (1) *is called the noise level of weighted classification.*

The condition (1) controls the perturbation of classifier in matrix probability space $\mathbb{P}_{\boldsymbol{X}}$ with respect to the weighted classification risk. The identifiability implies that the set (π) is the unique (up to a measure-zero set in $\mathbb{P}_{\boldsymbol{X}}$) global optimizer of the weighted classification risk. The noise level α generally depends on the probability mass of $\boldsymbol{X}$ and the changing behavior of $p(\boldsymbol{X})$ near the boundary $\partial S_{\text{bayes}}(\pi) = \{\boldsymbol{X} \colon p(\boldsymbol{X}) = \pi\}$. Note that α is upper bounded by 1, because we always have $R_\pi(S) - R_\pi((\pi)) \leq \mathbb{P}_{\boldsymbol{X}}[S\Delta(\pi)]$. In particular, $\alpha = 0$ corresponds to no identifiability.

Our first result is to bound the regression excess risk using classification excess risk.

Theorem 4.1 (Nonparametric regression via weighted classifications)**.** *Let $p(\boldsymbol{X})$ be a regression function, and $\bar{p}(\boldsymbol{X})$ be the linear combination of weighted classifiers in* (4.1)*. Suppose the level sets $\{S_{bayes}(\pi)\}_{\pi\in\Pi}$ are all identifiable with constants $\alpha \in (0, 1]$ and $C \in (0, \infty)$. Then,*

$$R_{\text{reg}}(\bar{p}) - R_{\text{reg}}(p) \leq 8\mathbb{E}_{\boldsymbol{X}} |\bar{p}(\boldsymbol{X}) - p(\boldsymbol{X})| \leq \frac{8}{H} + C \sum_{\pi\in\Pi} \left[R_\pi(\bar{S}(\pi)) - R_\pi((\pi))\right]^\alpha. \quad (9)$$

Theorem 4.1 shows the key role of $\bar{p}(\boldsymbol{X})$ in bridging regression and classification. The results suggest that the estimation of $p(\boldsymbol{X})$ can be reduced to estimation of $\bar{p}(\boldsymbol{X})$, or equivalently, to a sequence of weighted classifications $\{\bar{S}(\pi)\}_{\pi\in\Pi}$. The approximation error (4.1) consists of two terms. The first term is the discretization error due to the step function approximation to the regression function, which goes to 0 as $H \to \infty$. The second term is the excess risk in classification optimized over $\mathcal{S}$ compared to that over all Borel sets, representing the capability of the candidate classifiers $\mathcal{S}$.

Estimating $\bar{p}(\boldsymbol{X})$ as a surrogate of $p(\boldsymbol{X})$ provides several benefits. From a statistical perspective, $\bar{p}(\boldsymbol{X})$ is a finite combination of weighted classifiers, which are easier to solve than regression. From the perspective of functional analysis, the function $\bar{p}(\boldsymbol{X})$ provides

a valid approximation to $p(\boldsymbol{X})$ even when $p(\boldsymbol{X})$ is non-smooth and oscillating. In particular, the estimation of $\bar{p}(\boldsymbol{X})$ relies less crucially on the local neighborhood of $\boldsymbol{X}$. This feature is especially appealing for matrix-valued predictors, since the predictor space is high dimensional and often barely explored by small sample size data.

We conclude this section with a remark on the Assumption 1. We show in Appendix **??** that the Assumption 1 is equivalent to

$$\mathbb{P}_{\boldsymbol{X}}\left(|p(\boldsymbol{X}) - \pi| \leq t\right) \leq C' t^{\alpha/(1-\alpha)}, \quad \text{for all } t \in (0,\ t_*], \tag{10}$$

for some constants $t_* > 0$ and $C' = C'(t_*) > 0$. The condition (4.1) controls the probability mass of $\boldsymbol{X}$ located near the level set boundary $\partial S_{\text{bayes}}(\pi) = \{\boldsymbol{X} : p(\boldsymbol{X}) = \pi\}$. The value of α typically depends on uncertainty near the optimal decision boundary but less crucially on the predictor dimensions. Consider a simple case when the entries of matrix $\boldsymbol{X}$ are i.i.d. drawn from Uniform$[-1, 1]$; that is, $\mathbb{P}_{\boldsymbol{X}}$ is the Lebesque measure on $[-1, 1]^{d_1 d_2}$. Then, the best rate $\alpha \to 1$ corresponds to a sharp change (e.g. a jump) of $p(\boldsymbol{X})$ at the boundary, whereas the worst rate $\alpha \to 0$ corresponds to a sticky change (e.g., a plateau) of $p(\boldsymbol{X})$ near the boundary. An intermediate case $\alpha = 1/2$ happens when $p(\boldsymbol{X}) = \langle \boldsymbol{B}, \boldsymbol{X} \rangle$ is a linear function near the boundary, with $\boldsymbol{B} \in \mathbb{R}^{d_1 \times d_2}$ being a non-zero coefficient matrix (verify?).

4.2 Sparse and low-rank function boundaries

We now describe the choice of candidate classifiers $\mathcal{S}$ in (4.1). We rewrite the optimization (4.1) as the minimization over continuous-valued functions,

$$\bar{S}(\pi) = \{\boldsymbol{X} : \bar{f}(\boldsymbol{X}) \geq 0\}, \quad \text{with} \quad \bar{f}(\boldsymbol{X}) = \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[w_\pi(y) \mathbb{1}(y \neq f(\boldsymbol{X}))\right], \tag{11}$$

where $\bar{f} \in \mathcal{F}$ is a continuous-valued function from $\mathbb{R}^{d_1 \times d_2}$, and its sign function $\bar{f}$ induces the classifier $\bar{S}(\pi) \in \mathcal{S}$. The choice of $\mathcal{S}$ thus reduces to the choice of function family $\mathcal{F}$. A desirable $\mathcal{F}$ should balance the prediction and interpretability; i.e., $\mathcal{F}$ should be flexible enough for accurate prediction while being simple enough for high interpretability.

We propose the linear function family $\mathcal{F}$ with low-rank two-way sparse matrix coefficients,

$$\mathcal{F}(r, s_1, s_2) = \{f \colon \boldsymbol{X} \mapsto \langle \boldsymbol{X}, \boldsymbol{B} \rangle + b \;\big|\; \mathrm{rank}(\boldsymbol{B}) \le r,\ \mathrm{supp}(\boldsymbol{B}) \le (s_1, s_2),\ \boldsymbol{B} \in \mathbb{R}^{d_1 \times d_2},\ b \in \mathbb{R}\}, \tag{12}$$

where $\mathrm{rank}(\boldsymbol{B})$ denotes the rank of the coefficient matrix, and $\mathrm{supp}(\boldsymbol{B})$ denotes the two-way sparsity, with $s_1 = \|\boldsymbol{B}\|_{1,0}$ and $s_2 = \|\boldsymbol{B}^T\|_{1,0}$ being the numbers of non-zero rows and columns of $\boldsymbol{B}$, respectively. For the theory, we assume that (r, s_1, s_2) are known; the adaptation to unknown (r, s_1, s_2) in practice is described in Section 8. Combining formulations (4.1), (4.2) and (4.2) yields our (population version) "learning deduction" approach to nonparametric matrix regression,

$$\begin{aligned} \bar{p}(\boldsymbol{X}) &= \frac{1}{2H} \sum_{\pi \in \Pi} I(\boldsymbol{X} \in \bar{S}(\pi)) + \frac{1}{2}, \quad \text{where} \\ \bar{S}(\pi) &= \{\boldsymbol{X} \colon \bar{f}(\boldsymbol{X}) \ge 0\}, \quad \text{with} \quad \bar{f}(\boldsymbol{X}) = \underset{f \in \mathcal{F}(r, s_1, s_2)}{\arg\min}\ \mathbb{E}\left[w_\pi(y) \mathbb{1}(y \ne f(\boldsymbol{X}))\right], \end{aligned} \tag{13}$$

where the expectation is taken with respect to the unknown joint probability $\mathbb{P}_{\boldsymbol{X},y}$. The empirical estimator with unknown $\mathbb{P}_{\boldsymbol{X},y}$ will be addressed in Section 5.

The low-rank two-way sparse classifiers (4.2) enable efficient variable selection in the high-dimensional matrix learning, thereby achieving high interpretability in prediction. In the brain network analysis, for example, scientists are interested in identifying important nodes attached to at least one active edge with non-zero effects. Classical entrywise sparsity

essentially treats $\boldsymbol{X}$ as "a bag of edges", and loses the two-way indexing information due to vectorizing. In contrast, our two-way sparsity efficiently identifies the underlying active nodes by making use of matrix structure in the predictors.

It is worthy noting that the linearity in the classifiers $\mathcal{F}$ does not preclude the global nonlinearity in the regression function $p(\boldsymbol{X})$ or its variant $\bar{p}(\boldsymbol{X})$. As shown in the following examples, many nonlinear regression functions in existing literature are special cases of our representation (4.2) with (4.2), in the sense that the second term in the approximation (4.1) becomes precisely zero.

Example 1 (Monotonic function)**.** *Suppose the regression function can be expressed as* $p = g \circ f$*, where* $\circ$ *denotes the function composition,* $g \colon \mathbb{R} \to [0,1]$ *is an arbitrary monotonic (link) function, and* $f(\boldsymbol{X}) = \langle \boldsymbol{X}, \boldsymbol{B} \rangle$ *for some low-rank two-way sparse matrix* $\boldsymbol{B}$*. Then, the level sets satisfy* $S(\pi) = \bar{S}(\pi)$ *for all* $\pi \in [0,1]$*.*

Common link functions, such as logistic function $g(t) = (1 + \exp(-t))^{-1}$, arctangent function $g(t) = \frac{1}{\pi}\arctan(t) + \frac{1}{2}$, truncated rectified linear unit (ReLU) function $g(t) = t\mathbb{1}(t \in [0,1]) + \mathbb{1}(t > 1)$, and all inverse cumulative distribution functions are included in our functional class. In particular, our model incorporate the parametric matrix regression models in the earlier literature (**???**).

Our function family also incorporates certain parametric models from matrix linear discriminant analysis (LDA) (**?**).

Example 2 (Multivariate normal mixtures)**.** *Suppose the matrix-valued predictor* $\boldsymbol{X}$ *follows Gaussian mixture distribution,* $\boldsymbol{X}|y = -1 \sim \mathcal{MN}(\boldsymbol{B}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ *and* $\boldsymbol{X}|y = 1 \sim \mathcal{MN}(\boldsymbol{B}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$*, where* $\mathcal{MN}$ *denotes matrix normal distribution, and* $\boldsymbol{B}_1$*,* $\boldsymbol{B}_2$ *are low-rank two-way sparse mean matrices. Furthermore, assume* $\mathbb{P}(y = 1) = \mathbb{P}(y = -1) = 1/2$*. Then, the level sets satisfy* $S(\pi) = \bar{S}(\pi)$ *for all* $\pi \in [0,1]$*.* *(*$\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ *psd?)*

The following proposition gives a sufficient condition for exact recovery of $S(\pi)$ through $\bar{S}(\pi)$.

Proposition 1 (Low-rank and sparse boundaries)**.** *Let $p(\boldsymbol{X})$ be a regression function. For a given $\pi \in \Pi$, let $S(\pi)$ be the level set defined in* (2.2)*, and $\bar{S}(\pi)$ be the classifier defined in* (4.2) *optimized over $\mathcal{F}(r, s_1, s_2)$. Suppose that there exists a low-rank two-way sparse matrix $\boldsymbol{B}_\pi$ and a real value $b_\pi \in \mathbb{R}$, such that every element in the boundary set $\partial S(\pi) = \{\boldsymbol{X} : p(\boldsymbol{X}) = \pi\}$ is the solution of $\langle \boldsymbol{X}, \boldsymbol{B}_\pi \rangle = b_\pi$. Then $S(\pi) = \bar{S}(\pi)$ or $S(\pi) = \bar{S}^c(\pi)$. Note that $(\boldsymbol{B}_\pi, b_\pi)$ is allowed to vary depending on π. (conditions on p?)*

In principle, more complicated classifiers, such as neural network, decision trees, and boosting, can also be brought to bear on the level set construction (4.2). The ability to import and adapt existing classification methods is one advantage of the proposed learning reduction framework. We find that, in our motivating brain network analysis, the low-rank two-way sparse classifiers (4.2) provide the benefit of interpretable predictions. We focus on linear classifiers here because interpretation is at least as important as prediction in the neuroimaging application, and variable selection through $\boldsymbol{B}_\pi$ is a powerful way to achieve this. The nonlinear extension of $\mathcal{F}(r, s_1, s_2)$ will be discussed in Section 7.

5 Large-margin learning with high dimensional matrices

In previous sections we have established the population properties from classification to regression. In this section we address the empirical learning problems when the true distribution $\mathbb{P}_{\boldsymbol{X},y}$ is unknown. The objective function in the optimization (4.2) now becomes

empirical (stochastic) risks estimated from high dimensional low sample size training data $(X_i, y_i)_{i=1}^n$. We first develop an efficient large-margin classification with $\pi = 1/2$, and then generalize the accuracy guarantees to level set estimator $\hat{S}(\pi)$. The sample complexity for the proposed nonparametric matrix regression is established, leading to a high dimensional consistency result that allows the matrix dimension $d_1 d_2$ to grow sub-exponentially with sample size n.

5.1 Classification of high dimensional matrices

We consider the matrix classification problem with $\pi = 1/2$ in the formulation (4.2). We propose the estimated matrix classifier based on penalized empirical surrogate risk minimization,

$$\hat{S}_{\text{bayes}} = \{\boldsymbol{X} \colon \hat{f}(\boldsymbol{X}) \geq 0\}, \quad \text{where} \quad \hat{f} = \underset{f \in \mathcal{F}(r, s_1, s_2)}{\arg\min} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i f(\boldsymbol{X}_i)) + \lambda \|f\|_F^2 \right\}, \quad (14)$$

where $\mathcal{F}(r, s_1, s_2)$ is the function family specified in (4.2), the surrogate loss $\ell(z) \colon \mathbb{R} \mapsto \mathbb{R}_{\geq 0}$ is a non-increasing function of the margin $z = yf(\boldsymbol{X})$, $\lambda > 0$ is the penalty parameter, and we define the penalization term $\|f\|_F^2 = \|\boldsymbol{B}\|_F^2$, with $\boldsymbol{B}$ being the coefficient matrix associated with $f \in \mathcal{F}(r, s_1, s_2)$. Examples of large-margin loss functions are hinge loss $\ell(z) = (1 - z)_+$ for support vector machines, logistic loss $\ell(z) = \log(1 + e^{-z})$ for important vector machines, exponential loss $\ell(z) = \exp(-z)$ for boosting, and ψ-loss $\ell(z) = 2\min(1, (1 - z)_+)$, where $z_+ = \max(z, 0)$ denotes the non-negative truncation of $z \in (-\infty, \infty)$. Our algorithm implements the hinge loss for illustration, although our framework and theory are applicable to general large-margin losses (**?**).

The estimation (4) generalizes the population formulation (4.2) from three aspects. First, the population expectation in (4.2) is replaced by the empirical sample average,

which is common in statistical learning problems with i.i.d. assumption. Second, we add the ridge penalization $\lambda\|f\|_F^2$ to control the magnitude of the classifiers. The oracle tuning parameter λ depends on the sample size and the problem dimension as we will describe in the next paragraph. In practice, we choose λ in a data-adaptive fashion via cross validation. The resulting sieve estimator enjoys numerical stability and statistical accuracy. Third, we replace the binary loss in (4.2) by a more manageable large-margin loss. This relaxation allows us to leverage efficient large-margin algorithms while maintaining desirable statistical performance under mild assumptions.

We now provide the estimation accuracy guarantees for the matrix classifier (4). Let $d_1 = d_2 = d$ for ease of presentation. We consider the high dimensional regime as both the sample size n and matrix dimension d diverge, while treating (r, s_1, s_2) as fixed constants. This scenario is particularly relevant in modern neuroimaging applications, for example, when the number of nodes in the brain connectivity matrix grow as the resolution of atlas template increases.

Assumption 2 (Conditions for matrix classification)**.** *Let* $(\boldsymbol{X}) = I(\boldsymbol{X} \in S_{bayes}) = I(\boldsymbol{X} : p(\boldsymbol{X}) \geq 1/2)$ *denote the set indicator function corresponding to the Bayes classifier* (2)*, and let* $R_\ell(f) = \mathbb{E}[\ell(yf(\boldsymbol{X}))]$ *denote the surrogate risk of a function* $f \colon \mathbb{R}^{d_1 \times d_2} \mapsto \mathbb{R}$*. Consider the following assumptions:*

(i) (Approximation error). There exists a sequence of bounded function $f_n^* \in \mathcal{F}(r, s_1, s_2)$ *whose excess surrogate risk vanishes; i.e.,* $R_\ell(f_n^*) - R_\ell() \leq a_n \to 0$ *and* $\|f_n^*\|_F \leq C$*, for some constant* $C > 0$ *and some vanishing sequence* $a_n \to 0$ *as* $n, d \to \infty$*,*

(ii) (Local variance-mean relationship). There exist constants $\rho \in [0, 1], C > 0, \delta > 0$*, such that, for all* $f \in \mathcal{F}(r, s_1, s_2)$ *with* $R_\ell(f_n^*) - R_\ell() \leq \delta$*, the variance of the excess surrogate*

loss is bounded in terms of its expectation; i.e.,

$$Var[\ell(yf(\boldsymbol{X})) - \ell(y(\boldsymbol{X}))] \leq C[R_\ell(f) - R_\ell()]^\rho. \tag{15}$$

Assumption 2(i) ensures the vanishing difference in the surrogate risk attained by $\mathcal{F}(r, s_1, s_2)$ and that by . Note that this assumption is weaker than the convergence between the candidate set $\mathcal{F}(r, s_1, s_2)$ and Bayes classifier . The former concerns the convergence in 1-dimensional risk value whereas the later concerns the convergence in high dimensional matrix space. Assumption 2(ii) quantifies the local smoothness of loss variance within a neighborhood of . The exponent ρ depends on the joint distribution of $(\boldsymbol{X}, y)$ and the specification of surrogate loss ℓ. In particular, $\rho = 1$ is satisfied (for all distributions)? by strictly convex loss such as ψ-loss function. The variance-mean relationship (2) is generally weaker than the identifiability condition (1), in the sense that Assumption 1 implies Assumption 2(ii) with $\rho = \alpha \wedge 1$ when the surrogate loss reduces to usual binary loss.

We show that the proposed classifier (4) achieves statistical consistency even when the matrix dimension d far exceeds the sample size n.

Theorem 5.1 (Accuracy for matrix classification)**.** *Let S_{bayes} be the Bayes classifier* (2) *defined in the matrix space $\mathbb{R}^{d\times d}$. Suppose Assumption 2 holds with $\rho \in [0, 1]$. Consider the penalized empirical surrogate risk minimizer $\hat{S}_{bayes}$ in* (4)*, where $\mathcal{F}(r, s_1, s_2)$ is the low-rank two-way sparse function family* (4.2)*, and the penalty parameter is selected as $\lambda \asymp \left(\frac{r(s_1+s_2)\log d}{n}\right)^{1/(2-\rho)}$. Then, with very high probability over the training data $(\boldsymbol{X}_i, y_i)_{i=1}^n$, we have*

$$R(\hat{S}_{bayes}) - R() \leq C\left(\frac{r(s_1 + s_2)\log d}{n} + a_n\right)^{1/(2-\rho)}. \tag{16}$$

Here $R(\cdot)$ denotes the expected classification error defined in (4.2) with respect to a

prototypical test point, whereas the high probability conclusion for the event (5.1) is with respect to the training data.

To gain insight from the results (5.1), consider the case when the statistical error dominates the approximation error in that $a_n \lesssim \frac{r(s_1+s_2)\log d}{n}$. Then, the bound (5.1) immediately implies the classification consistency in the high dimensional regime $d, n \to \infty$, as long as the matrix dimension d grows sub-exponentially in sample size n; i.e., $d = o(e^n)$. This remarkable sample complexity highlights the benefit of low-rank two-way sparse model and structural risk minimization approaches to matrix classification. Furthermore, we find that, for fixed d, the estimated classifier (5.1) reaches a fast rate $O(1/n)$ when $\rho = 1$. This observation generalizes the asymptotics for usual classification with fixed feature dimension (**???**). As we discussed in Section 4.1, the fast rate may happen when noise is low near the classifier boundary ∂.

5.2 Level set estimation in high dimensional matrix space

Our classification results naturally generalize to π-level set estimation for arbitrary $\pi \in [0, 1]$. We omit the derivation here but only summarize the difference. The proposed π-level set estimator $\hat{S}(\pi)$ is

$$\hat{S}(\pi) = \{\boldsymbol{X} \colon \hat{f}_\pi(\boldsymbol{X}) \geq 0\}, \quad \text{where} \quad \hat{f}_\pi = \min_{f \in \mathcal{F}(r, s_1, s_2)} \left\{ \frac{1}{n} \sum_{i=1}^{n} w_\pi(y_i) \ell\left(y_i f(\boldsymbol{X}_i)\right) + \lambda \|f\|_F^2 \right\}, \tag{17}$$

where $w_\pi(y) = 1 - \pi$ if $y = 1$ and $w_\pi(y) = \pi$ if $y = -1$. We impose unequal cost for the two labels $\{-1, 1\}$ in the objective function. Many common large-margin losses can be accommodated into the weighted classification (5), e.g., hinge loss $\ell(z) = (1 - z)_+$, logistic loss $\ell(z) = \log(1 + e^{-z})$, and exponential loss $\ell(z) = e^{-z}$. The properties and

choices of surrogate loss in weighted classification have been studied earlier (**??**). We choose hinge loss for parsimony; our theory generalizes to other large-margin losses with certain conditions (**?**).

We need some additional notation to establish the level set estimation accuracy. Let $f_{\text{bayes},\pi}(\boldsymbol{X}) = I(\boldsymbol{X} \in (\pi)) = I(\boldsymbol{X} \colon p(\boldsymbol{X}) \geq \pi)$ denote the set indicator function corresponding to the π-level set, we will refer it to as the level set function. Let $R_\pi(f) = \mathbb{E}\left[w_\pi(y)\mathbb{1}\{y \neq f(\boldsymbol{X})\}\right]$ denote the weighted classification risk, and $R_{\ell,\pi}(f) = \mathbb{E}\left[w_\pi(y)\ell(yf(\boldsymbol{X}))\right]$ denote the surrogated weighted classification risk. An important fact of hinge loss is that, for all $\pi \in [0,1]$, the weighted excess satisfies the bound,

$$R_\pi(f) - R_\pi() \leq R_{\ell,\pi}(f) - R_{\ell,\pi}(), \quad \text{for all measurable functions } f, \tag{18}$$

Therefore, convergence respect to the surrogate risk $R_{\ell,\pi}$ implies convergence with respect to the classification risk R_π. (For other aforementioned surrogate losses, the excess bound holds up to a constant term and an additional exponent in $(0,1)$ on the right hand side.) The bound (5.2) also implies that, among all measurable functions $f \colon \mathbb{R}^{d\times d} \mapsto [0,1]$, the level set function minimizes the surrogate loss $R_{\ell,\pi}(f)$. The following assumption asserts the essential uniqueness of the minimizer.

Assumption 3 (Identifiability under surrogate loss)**.** *There exist constants $C > 0$, $\alpha \in (0,1]$, and $\delta > 0$, such that, for all functions with $R_{\ell,\pi}(f) - R_{\ell,\pi}(f_{bayes,\pi}) \leq \delta$,*

$$\mathbb{E}\big|I(\boldsymbol{X} \colon f(\boldsymbol{X}) \geq 0) - f_{bayes,\pi}(\boldsymbol{X})\big| \leq C\left[R_{\ell,\pi}(f) - R_{\ell,\pi}(f_{bayes,\pi})\right]^\alpha. \tag{19}$$

Assumption 3 is analogous to the identifiability Assumption 1 in the current context of surrogate loss. On one hand, Assumption 3 strengthens the earlier variance-mean assumption (2), in the sense that α in (3) implies $\rho = \alpha$ in (2). On the other hand, Assumption 3

relaxes the usual identifiability condition (1) because of the excess bound (5.2) for hinge loss. With Assumption 3, our estimator $\hat{S}_{\text{bayes}}(\pi)$ accurately recovers the level set in the high-dimensional matrix space.

Theorem 5.2 (Accuracy for level set estimation)**.** *Consider the π-level set estimation with any given $\pi \in (0,1)$. Suppose Assumption 2(i) and Assumption 3 hold with $\alpha \in (0,1]$. Let $\hat{S}(\pi)$ be the level set estimator in* (5) *with penalty parameter $\lambda \asymp \left(\frac{r(s_1+s_2)\log d}{n}\right)^{1/(2-\alpha)}$. Then, with very high probability over the training data,*

$$\mathbb{P}\left(\hat{S}(\pi)\ \Delta\ (\pi)\right) \leq C\left(\frac{r(s_1+s_2)\log d}{n} + a_n\right)^{\alpha/(2-\alpha)},$$

where the probability on the left hand side is taken with respect to a prototypical test point $\boldsymbol{X}$ i.i.d. from the training set $(\boldsymbol{X}_i)_{i=1}^n$.

Theorem 5.2 reveals the weak dependence on matrix dimension in the level set estimation error. This result again highlights the benefit of proposed low-rank two-way sparsity function boundary models. Furthermore, the convergence rate depends on the exponent α in (1). As we discuss in Section 4, the value of α is related to the changing behavior of $p(\boldsymbol{X})$ around the level set boundary ∂. Our characterization with α requires that the function $p(\boldsymbol{X})$ changes at least polynomially fast in distance from the level set boundary. Accurate set estimation is more difficult at levels where the function is relatively flat (small α), as intuition would suggest. The case $\alpha = 1$ generally corresponds to discontinuity in the function $p(\boldsymbol{X})$ (**?**), and the case $\pi = 1/2$ reduces to the usual classification. In practice, we suggest to choose tuning parameter λ in (5) via data adaptive procedures such as cross validation, which requires no prior knowledge of α.

5.3 Nonparametric matrix regression

In this section, we present the empirical estimator and accuracy guarantee for our nonparametric matrix regression. We propose the following function estimator $\hat{p}(\cdot)\colon \mathbb{R}^{d\times d} \mapsto [0,1]$ based on empirical surrogate risk minimization,

$$\hat{p}(\boldsymbol{X}) = \frac{1}{2H}\sum_{\pi\in\Pi} I(\boldsymbol{X}\in\hat{S}(\pi)) + \frac{1}{2}, \quad \text{where}$$

$$\hat{S}(\pi) = \{\boldsymbol{X}\colon \hat{f}(\boldsymbol{X})\geq 0\}\subset\mathbb{R}^{d\times d}, \quad \text{with} \quad \hat{f} = \underset{f\in\mathcal{F}(r,s_1,s_2)}{\arg\min}\left\{\frac{1}{n}\sum_{i=1}^{n} w_\pi(y_i)\ell(y_i f(\boldsymbol{X}_i)) + \lambda\|f\|_F^2\right\}. \tag{20}$$

In contrast to formulation (4.2), we have chosen to use a large-margin loss and an additional ridge penalty. The benefit of these two modifications have been shown in earlier two sections.

We now provide the accuracy guarantee between the estimator $\hat{p}(\boldsymbol{X})$ and target regression function $p(\boldsymbol{X})$. There are three sources of error to consider in our learning framework: the statistical error in classification due to finite sample size, the approximation error due to the size of the function space $\mathcal{F}(r,s_1,s_2)$, and an additional source of discretization error due to the approximation from classification to regression. Combining results in Theorem 4.1 and earlier two sections, we obtain the main result of this section as follows.

Theorem 5.3 (Accuracy for nonparametric matrix regression)**.** *Suppose Assumption 2(i) and Assumption 3 hold with a constant $\alpha\in(0,1]$ for all $\pi\in\Pi$. Under the choice $\lambda \asymp \left(\frac{r(s_1+s_2)\log d}{n}\right)^{1/(2-\alpha)}$ in* (5.3)*, we have, with probability at least $1-C\exp(-a_n\lambda^{2-\alpha})$,*

$$\mathbb{E}|\hat{p}(\boldsymbol{X}) - p(\boldsymbol{X})| \leq \underbrace{\frac{1}{H}}_{\textit{discretization error}} + CH\left(\underbrace{\frac{r(s_1+s_2)\log d}{n}}_{\textit{statistical error}} + \underbrace{a_n}_{\textit{approximation error}}\right)^{\alpha/(2-\alpha)},$$

where $C > 0$ is a constant, and the expectation on the left hand side is taken respect to a prototypical test point $\boldsymbol{X}$ i.i.d. from the training set $(\boldsymbol{X}_i)_{i=1}^n$. Where does the H in the second term arise?? variance-mean tradeoff?

Theorem 5.3 implies the high dimensional consistence of our nonparametric matrix regression, provided that the true level set boundaries are well approximated by the sparse representation $\mathcal{F}(r, s_1, s_2)$

[High-dimensional consistency] Consider the same set-up as in Theorem 5.3. Assume $a_n \lesssim \frac{r(s_1+s_2)\log d}{n}$, and choose $H \asymp \left[\frac{n}{r(s_1+s_2)\log d}\right]^{(4-2\alpha)/\alpha}$. Then, the regression function estimator $\hat{p}(\boldsymbol{X})\colon \mathbb{R}^{d\times d} \to [0,1]$ achieves high dimensional consistency; that is,

$$\mathbb{E}|\hat{p}(\boldsymbol{X}) - p(\boldsymbol{X})| \leq \mathcal{O}_p\left(\frac{r(s_1+s_2)\log d}{n}\right)^{\alpha/(4-2\alpha)}, \quad \text{as } d, n \to \infty \text{ while } d = o(e^n), \tag{21}$$

where $\mathcal{O}_p$ denotes the stochastic dominance in probability.

In practice, data adaptive procedures are useful for choosing the smoothing parameter H. A large value of H yields better precision at a higher computational cost. We set the default value $H = \lfloor n^{1/2} \rfloor$ in our algorithm, where $\lfloor \cdot \rfloor$ denotes the greatest integer bounded by $n^{1/2}$. This choice seems satisfactory in our simulations and data analysis considered.

We conclude this section by comparing the errors in regression (5.3) and classification (5.1). We find that the regression error $(n^{-1}\log d)^{\alpha/(4-2\alpha)}$ is slower than the corresponding classification rate $(n^{-1}\log d)^{-\alpha/(2-\alpha)}$. Furthermore, the regression problem requires more assumptions than classification, as we describe in Sections 5.1 and 5.2. Our learning reduction approach bridges these two tasks using level set estimation, a problem lies somewhere in between. The connection allows us to disentangle complexity and leverage existing algorithms. This principle may extend to other learning framework that relates one supervised problem to another which is more fundamental.

6 Alternating optimization for structural risk minimization

revised till here In this Section, we describe an algorithm to seek the optimizer of Equation (4) 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 (4.2) is written as

$$\min_{\substack{\boldsymbol{C}\in\mathcal{H}_1^{d_1}\times\mathcal{H}_2^{d_2},\\ \boldsymbol{P}\in\mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}}} \frac{1}{2}\|\boldsymbol{C}\boldsymbol{P}^T\|_F^2 + C\sum_{i=1}^n \xi_i, \tag{22}$$
$$\text{subject to } y_i\langle \boldsymbol{C}\boldsymbol{P}^T, \Phi(\boldsymbol{X}_i)\rangle \le 1-\xi_i \text{ and } \xi_i \ge 0, i = 1,\dots,n.$$

Optimization problem (7) 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 (7) 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

$$\boldsymbol{C} = \sum_{i=1}^n \alpha_i y_i \Phi(\boldsymbol{X}_i)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1} \in \mathcal{H}_r^r \times \mathcal{H}_c^r. \tag{23}$$

We use the formula (8) without information about feature mapping $\Phi(\cdot)$. Second, we assume that $\boldsymbol{C}$ is fixed and update $\boldsymbol{P}$. The dual problem of Equation (7) with fixed $\boldsymbol{C}$ is

$$\max_{\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)} -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_i)\right), \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_j)\right)\rangle, \tag{24}$$

$$\text{subject to } 0 \leq \alpha_i \leq C, i = 1, \ldots, n,$$

We can find an optimizer of (9) based on kernel information only. We obtain the following formula by plugging (8) into components of (9).

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

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 (9) 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,

$$\hat{f}(\boldsymbol{X}) = \sum_{k=1}^{n}\hat{\alpha}_k y_k \Bigg(\sum_{i=1}^{d_1}\sum_{j=1}^{d_1}[\hat{\boldsymbol{P}}_1(\hat{\boldsymbol{P}}_1^T\hat{\boldsymbol{P}}_1)^{-1}\hat{\boldsymbol{P}}_1^T]_{ij} K_r\left([\boldsymbol{X}_k]_{i:}, [\boldsymbol{X}]_{j:}\right) + \sum_{i=1}^{d_2}\sum_{j=1}^{d_2}[\hat{\boldsymbol{P}}_2(\hat{\boldsymbol{P}}_2^T\hat{\boldsymbol{P}}_2)^{-1}\hat{\boldsymbol{P}}_2^T]_{ij} K_c\left([\boldsymbol{X}_k]_{:i}, [\boldsymbol{X}]_{:j}\right) \Bigg). \tag{25}$$

Algorithm 1 gives the full description for classification.

By the similar way with little modification, we can obtain an algorithm for weighted margin classifier (5). From the explanation in Section 5.2 and 5.3, we summarize level set and regression estimation procedure in Algorithm 2.

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 (10)

Sketch of ADMM

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

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

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

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

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 (5) 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:**

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

Output: Level sets $\hat{S}(\pi_h)$ for $h = 1, \ldots H$ and regression function $\hat{p}(\boldsymbol{X})$.

1. Update $\boldsymbol{B}$:

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

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

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

2. Update $\boldsymbol{c}$:

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

3. Update $\boldsymbol{S}$:

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

4. Update $\boldsymbol{\Lambda}$:

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

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 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*

$$\begin{aligned} \Phi(\boldsymbol{X}) \colon \mathbb{R}^{d_1 \times d_2} &\to (\mathcal{H}_1 \times \mathcal{H}_2)^{d_1 \times d_2} \\ \boldsymbol{X} \mapsto \Phi(\boldsymbol{X}) = [\![\Phi(\boldsymbol{X})_{ij}]\!] &\text{ where } \Phi(\boldsymbol{X})_{ij} \stackrel{def}{=} \left(\phi_2(\boldsymbol{X}_{i:}), \phi_1(\boldsymbol{X}_{:j})\right). \end{aligned} \tag{26}$$

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 by,

$$\begin{aligned} f(\boldsymbol{X}) \stackrel{def}{=} \langle \boldsymbol{B}, \Phi(\boldsymbol{X}) \rangle, \text{ where } \boldsymbol{B} = [\![(\boldsymbol{b}_i^{row}, \boldsymbol{b}_j^{col})]\!] \in (\mathcal{H}_1 \times \mathcal{H}_2)^{d_1 \times d_2}, \\ \text{with } \boldsymbol{b}_i^{row} \in \mathcal{H}_1 \text{ and } \boldsymbol{b}_j^{col} \in \mathcal{H}_2 \text{ for all } (i, j) \in [d_1] \times [d_2]. \end{aligned} \tag{27}$$

Notice we impose this structure on $\boldsymbol{B}$ for identifiability issue. These matrix valued feature mapping (11) and corresponding linear function (12) 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 (12) admits low rank

decomposition as in Section 4.2,

$$\boldsymbol{B} = \boldsymbol{P}_1 \boldsymbol{C} \boldsymbol{P}_2^T, \text{ where } \boldsymbol{P}_1 \in \mathbb{R}^{d_1 \times r}, \boldsymbol{P}_2 \in \mathbb{R}^{d_2 \times r} \text{ and } \boldsymbol{C} = [\![(\boldsymbol{c}_i^{\text{row}}, \boldsymbol{c}_j^{\text{col}})]\!] \in (\mathcal{H}_1 \times \mathcal{H}_2)^{r \times r} \quad (28)$$
$$\text{with } \boldsymbol{c}_i^{\text{row}} \in \mathcal{H}_1 \text{ and } \boldsymbol{c}_j^{\text{col}} \in \mathcal{H}_2 \text{ for all } i, j \in [r].$$

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 4.2, we show that considered linear functions (12) with low-rank r defined by (13) are equivalent to the linear functions in Section 4.2 with the low-rank r constraint (4.2). 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*

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

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 results

9 Discussion

1. Semi-parametric:

   Parametric components (low-rankness + sparsity) for interpretation.

   Nonparametric (infinity smoothing parameter, level sets) for prediction

   2. Allow extension to bounded regression estimation for continuous response
   3. Allow other covariates.
   4. Adaptive resolution of level sets. e.g. detecting maxima and minima location.
   5. Convex relaxation