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 Methods

Consider a statistical learning problem where we would like to model the relationship between a feature $\boldsymbol{X} \in \mathcal{X}$ and a response $Y \in \mathcal{Y}$. Suppose that we observe a sample of of n data points, $(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)$, identically and independently distributed (i.i.d.) according to a unknown distribution $\mathbb{P}(\boldsymbol{X}, Y)$ over $\mathcal{X} \times \mathcal{Y}$. We are interested in predicting a new response Y_{n+1} from a new feature value $\boldsymbol{X}_{n+1}$. The observations $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$ are called the training data and $(\boldsymbol{X}_{n+1}, Y_{n+1})$ the test point. When no confusion arises, we often omit the subscript $(n+1)$ and simply write $(\boldsymbol{X}, Y)$ for the prototypical test point. The test point is assumed independent of the training data and is drawn from the same unknown distribution $\mathbb{P}$. Our goal is to make accurate prediction under a wide range of distributions. In particular, we consider a non-parametric, distribution-free setting with no strong assumptions on the data generative distribution other than i.i.d.

We focus on the scenario with matrix-valued predictors and binary label response; that is, $\mathcal{X} = \mathbb{R}^{d_1 \times d_2}$ and $\mathcal{Y} = \{-1, 1\}$. 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 (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-valued feature and a binary label response indicating whether the image is pedestrian (Shashua et al., 2004).

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 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-valued feature, $d_1 d_2$, is often comparable to, or even larger than the number of sample, n. Our method exploits the structural information in the data matrix to overcome these challenges.

2.0 Three main problems

Before we proceed with our proposal for matrix-valued features, we present the concrete learning problems of our interest. We consider three major supervised learning problems: classification, level set estimation, and regression estimation.

2.2 *The problem of classification*: Classification is the problem of identifying to which of a set of categories a new observation belongs, based on training samples. We aim to find a decision rule $g(\boldsymbol{X})$ that has small error

$$\mathbb{P}_{\boldsymbol{X},y}\left(y \neq g(\boldsymbol{X})\right),$$

where $g(\boldsymbol{X}) = \text{sign}(f(\boldsymbol{X}))$ and $f : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}$ is a decision function. 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. We want to tackle matrix valued classification preserving the matrix structure.

2.3 *The problem of 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. One example can be found in medical decision making. In Osteosarcoma treatment, the degree of tumor necrosis is used to guide the choice of postoperative chemotherapy (Man et al., 2005). 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 estimate the π-level set of the regression function based on classification problem.

2.4 *The problem of regression 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. 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 $p(\boldsymbol{X})$ based on level set estimation.

The three problems represent common 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.

2.1 Choice of decision function space

Here, we consider a set of linear predictors as decision function class and extend to nonlinear case.

add description. see below as an example.

2.1.1 Linear predictors

Transition from earlier section?

We impose low-rankness on a linear predictor of the form $f(\boldsymbol{X}) = \langle \boldsymbol{B}, \boldsymbol{X} \rangle$, where $\boldsymbol{B}, \boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$ and $\langle \boldsymbol{X}, \boldsymbol{X'} \rangle = \mathrm{Tr}(\boldsymbol{X}^T \boldsymbol{X'})$. Specifically, the coefficient matrix $\boldsymbol{B}$ has low-rank r usually much smaller than the matrix size $\min(d_1, d_2)$,

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

The aforementioned three problems are formulated as empirical structure minimization over functions with matrices as inputs.
In this section, we propose a family of matrix kernels, which are building blocks for defining functions in matrix space.
Kernels 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 using proper notion of similarity.
Unlike vectors, matrix-valued inputs encode two-way relationship across rows and columns at a time.
Taking into account of the two-way relationship is essential in the kernel development.
Our proposed kernel uses concept from latent factor models and incorporates the two-way similarities in kernels via low-rank regularization.
.....

The condition determines trade-off between model complexity and flexibility. This low-rankness makes distinction from classical classification problem for feature vectors and preserves structural information of feature matrices. This kind of linear predictors with low rank coefficient has been proposed in Support Vector Machine (SVM) classification problem (Pirsiavash et al., 2009; Luo et al., 2015). We apply this linear function based on matrix-valued features not only to classification but also level set estimation and regression estimations. Furthermore, we extend linear case to nonlinear case in the next section. ~~To the best of our knowledge, we are among the first who propose matrix version of feature mapping and kernels.~~ We suggest nonlinear learning methods based on the new concepts.

There are some prior work on matrix/tensor kernels. (See for example, tensor SVM in the reading folder.)
Most of these methods are inadaptive and do not use label information to guide the kernel construction.

2.1.2 Nonlinear predictors

We generalize classical kernel method for vector case to matrix case. Before proposing a new matrix feature mapping and kernel, we introduce notations and operations used later. Let $\phi_i \colon \mathbb{R}^{d_i} \to \mathcal{H}_i$ be feature mappings with a classical kernel $K_i \colon \mathbb{R}^{d_i} \times \mathbb{R}^{d_i} \to \mathbb{R}$ for $i = 1, 2$. $\mathcal{H}_i$ denotes image space of ϕ_i and a possibly infinite dimensional Hilbert space. Let $\mathcal{H}^d = \mathcal{H}^{1\times d} = \{(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) \colon x_i \in \mathcal{H}, \text{ for } i = 1, \ldots, d.\}$ denote the collection of row vectors with each entry taking value in a Hilbert space $\mathcal{H}$. Matrix algebraic operations are carried over as follows,

Proposition 1. *Let $\boldsymbol{A} = [\![a_i]\!]$ and $\boldsymbol{B} = [\![b_i]\!]$ be two vectors in $\mathcal{H}_1^{d_1}$ and $\boldsymbol{A}' = [\![a_i']\!]$ be a vector in $\mathcal{H}_2^{d_2}$. Let $\boldsymbol{P} \in \mathbb{R}^{d_1\times r}$ and $\boldsymbol{P}' \in \mathbb{R}^{d_2\times r}$ be real valued matrices. Then, we have well defined operations.*

- *Inner product:* $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_i \langle a_i, b_i \rangle \in \mathbb{R}$.
- *Linear combination:* $\boldsymbol{AP} = [\![c_i]\!] \in \mathcal{H}_1^r$ *where* $c_i = \sum_{k\in[d_1]} a_k p_{ki}$ *for all* $i \in [r]$.

- *Summation:* $\boldsymbol{A} + \boldsymbol{B} = [\![a_i + b_i]\!] \in \mathcal{H}_1^{d_1}$.
- *Matrix product:* $\boldsymbol{A}^T\boldsymbol{B} = [\![c_{ij}]\!] \in \mathbb{R}^{r\times r}$, *where* $c_{ij} = \langle a_i, b_j\rangle$, *for all* $i, j \in [r]$.
- *Tuple operation:* $(\boldsymbol{A}, \boldsymbol{A}')(\boldsymbol{P}, \boldsymbol{P}') = (\boldsymbol{AP}, \boldsymbol{A}'\boldsymbol{P}') \in \mathcal{H}_1^r \times \mathcal{H}_2^r$.

Now we present a feature mapping and matrix kernel. We define matrix valued feature mapping first.

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* d_1*-by-*d_2 *matrices:*

$$\begin{aligned} \Phi \colon \mathbb{R}^{d_1\times d_2} &\to \mathcal{H}_1^{d_1} \times \mathcal{H}_2^{d_2} \\ \boldsymbol{X} &\mapsto (\Phi_1(\boldsymbol{X}), \Phi_2(\boldsymbol{X})) \overset{def}{=} \left((\phi_1(\boldsymbol{X}_{1:}), \ldots, \phi_1(\boldsymbol{X}_{d_1:})), (\phi_2(\boldsymbol{X}_{:1}) \ldots, \phi(\boldsymbol{X}_{:d_2}))\right). \end{aligned} \tag{1}$$

Notice that the matrix feature mapping considers both row-wise and column-wise extended feature. We can define a linear function $f \colon \mathbb{R}^{d_1\times d_2} \to \mathbb{R}$ with respect to $\Phi(\boldsymbol{X}) \in \mathcal{H}_1^{d_1} \times \mathcal{H}_2^{d_2}$,

$$\begin{aligned} f(\boldsymbol{X}) &\overset{\text{def}}{=} \langle \boldsymbol{B}, \Phi(\boldsymbol{X})\rangle, \text{ where } \boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2) \in \mathcal{H}_1^{d_1} \times \mathcal{H}_2^{d_2} \\ &= \langle \boldsymbol{B}_1, \Phi_1(\boldsymbol{X})\rangle + \langle \boldsymbol{B}_2, \Phi_2(\boldsymbol{X})\rangle. \end{aligned} \tag{2}$$

Matrix valued feature map (1) and corresponding linear function (2) can be reduced down to existing vector-based methods and generalized to tensor case (see Supplement for the details). We assume that $\boldsymbol{B}$ in (2) admits low rank decomposition,

$$\boldsymbol{B} = \boldsymbol{C}\boldsymbol{P}^T, \text{ where } \boldsymbol{C} = (\boldsymbol{C}_1, \boldsymbol{C}_2) \in \mathcal{H}_1^r \times \mathcal{H}_2^r \text{ and } \boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2) \in \mathbb{R}^{d_1\times r} \times \mathbb{R}^{d_2\times r}. \tag{3}$$

When classical kernels K_i are linear kernels whose corresponding feature maps ϕ_i are identify $i = 1, 2$, One can check that considered linear functions with low-rank r in (3) is equivalent to linear functions with low-rank $2r$ in Section 2.1.1.

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*

$$\boldsymbol{K} \colon \mathbb{R}^{d_1 \times d_2} \times \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}$$

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

Notice that this kernel definition is generalization from classical kernel based on vector valued features. We can associate the feature mapping in Definition 1 with the matrix kernel like classical case which based on vectors. Given $\{\boldsymbol{W}_i, K_i\}_{i=1,2}$, we have

$$\begin{aligned}
\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}') &= \sum_{j,k \in [d_1]} w_{jk}^{(1)} K_1(\boldsymbol{X}_{j:}, \boldsymbol{X}'_{k:}) + \sum_{j,k \in [d_2]} w_{jk}^{(2)} K_2(\boldsymbol{X}_{:j}, \boldsymbol{X}'_{:k}) \\
&= \langle \boldsymbol{W}_1, \Phi_1(\boldsymbol{X})^T \Phi_1(\boldsymbol{X}') \rangle + \langle \boldsymbol{W}_2, \Phi_2(\boldsymbol{X})^T \Phi_2(\boldsymbol{X}') \rangle \\
&= \langle \boldsymbol{W}, \Phi(\boldsymbol{X})^T \Phi(\boldsymbol{X}') \rangle, \text{ where } \boldsymbol{W} = (\boldsymbol{W}_1, \boldsymbol{W}_2).
\end{aligned}$$

By Definition 2, we can successfully learn weight matrices thereby nonlinear function given pre-specified row-wise and column-wise kernels $\{K_i\}_{i=1,2}$ avoiding feature mapping in Definition 1. In Section 2.2, we specify learning problem and show how we learn nonlinear function using the matrix kernel.

2.2 Classification

We consider a large margin classifier that minimizes a cost function in f over a decision function class $\mathcal{F}$

$$\min_{f \in \mathcal{F}} n^{-1} \sum_{i=1}^{n} L\left(y_i f(\boldsymbol{X}_i)\right) + \lambda J(f), \tag{4}$$

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 (4). However, our estimation schemes and theorems are applicable to general large-margin classifiers.

Here we present the solution to (4) with nonlinear kernels; linear case is a special case of nonlinear case. Based on the decision function class in Section 2.1.2, we solve the following optimization problem.

$$(\hat{\boldsymbol{C}}, \hat{\boldsymbol{P}}) = \underset{\{\boldsymbol{C} \in \mathcal{H}_1^r \times \mathcal{H}_1^r, \boldsymbol{P} \in \mathbb{R}^{d_1 \times r} \times \mathbb{R}^{d_2 \times r}\}}{\arg\min} n^{-1} \sum_{i=1}^{n} \left(1 - y_i \langle \boldsymbol{C}\boldsymbol{P}^T, \Phi(\boldsymbol{X}_i) \rangle\right)_+ + \lambda \|\boldsymbol{C}\boldsymbol{P}^T\|_F^2. \quad (5)$$

Notice that the optimization problem (9) degenerates to the conventional SVM with vectorized feature matrices when the coefficient is full rank. From the solution to (9), our estimated decision rule is

$$\hat{g}(\boldsymbol{X}) = \text{sign}(\hat{f}(\boldsymbol{X})) = \text{sign}\left(\langle \hat{\boldsymbol{C}}\hat{\boldsymbol{P}}^T, \Phi(\boldsymbol{X}) \rangle\right).$$

We make a remark on the implication of our formulation (9). The formulation (9) 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 the a dual representation of the solution to (9).

$$\begin{aligned} f(\boldsymbol{X}) &= \sum_{i=1}^{n} \alpha_i y_i \langle \Phi(\boldsymbol{X}_i)\boldsymbol{H}_{\boldsymbol{P}}, \Phi(\boldsymbol{X})\boldsymbol{H}_{\boldsymbol{P}} \rangle, \text{ where } \boldsymbol{H}_{\boldsymbol{P}} = \boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T \quad (6) \\ &= \sum_{i=1}^{n} \alpha_i y_i \left(\langle \Phi_1(\boldsymbol{X}_i)\boldsymbol{H}_{\boldsymbol{P}_1}, \Phi_1(\boldsymbol{X})\boldsymbol{H}_{\boldsymbol{P}_1} \rangle + \langle \Phi_2(\boldsymbol{X}_i)\boldsymbol{H}_{\boldsymbol{P}_2}, \Phi_2(\boldsymbol{X})\boldsymbol{H}_{\boldsymbol{P}_2} \rangle \right) \\ &= \sum_{i=1}^{n} \alpha_i y_i \left(\sum_{j,k \in [d_1]} [\boldsymbol{H}_{\boldsymbol{P}_1}]_{jk}, K_1(\boldsymbol{X}_{j:}^{(i)}, \boldsymbol{X}_{k:}) + \sum_{j,k \in [d_1]} [\boldsymbol{H}_{\boldsymbol{P}_2}]_{jk}, K_2(\boldsymbol{X}_{:j}^{(i)}, \boldsymbol{X}_{:k}) \right). \end{aligned}$$

where $\{\alpha_i\}_{i=1}^n$ are (sparse) dual solution of (9) and $\boldsymbol{X}_{jk}^{(i)}$ denotes (j,k)-entry of $\boldsymbol{X}_i$. Notice that the last representation of (6) is a linear combination of matrix kernel induced by weight matrices $\{\boldsymbol{H}_{\boldsymbol{P}_i}\}_{i=1,2}$ and row and column-wise kernels $\{K_i\}_{i=1,2}$. Therefore, our considered function space can be written as

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

The projection matrices $\{\boldsymbol{H}_{\boldsymbol{P}_i}\}_{i=1,2}$ play role in reducing the feature dimension and at the same time, with coefficients $\boldsymbol{\alpha}$ find the best function that minimizes the classification risk. The procedure is summarized as the optimization over the union or RKHS induced by low rank weight matrices $\{\boldsymbol{W}_i\}_{i=1,2,}$

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

2.3 Level set estimation

We propose weighted loss function from (4) to estimate the level set.

$$\min_{f\in\mathcal{F}} n^{-1} \sum_{i=1}^n \omega_\pi(y_i) L\left(y_i f(\boldsymbol{X}_i)\right) + \lambda J(f), \tag{7}$$

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 (7) reduce to (4). The optimizer to Equation (7) with respect to all measurable function class yields an consistent estimate of the Bayes rule $g_\pi(\boldsymbol{X}) = \text{sign}\left(f_\pi(\boldsymbol{X})\right)$ with $f_\pi(\boldsymbol{X}) = p(\boldsymbol{X}) - \pi$ (Lin et al., 2002; Wang et al., 2008). Therefore, under the considered

decision function class as in Section 2.1, we obtain a minimizer $\hat{f}_\pi$ to (7) and estimate the level set as

$$\hat{S}(\pi) = \{\boldsymbol{X} : \mathbb{R}^{d_1 \times d_2} : \text{sign}(\hat{f}_\pi(\boldsymbol{X})) = 1\}. \tag{8}$$

2.4 Regression function estimation

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

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

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 (8) in Section 2.3 given $\pi = h/H$. Here we use consistency of level set estimation. Therefore, we estimate the regression function,

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

by repeatedly estimating the level sets in (8) with different π values, say $\pi = \frac{h}{H}$ for $h = 1, \ldots, H$.

3 Algorithm

In this Section, we describe the algorithm to seek the optimizer of Equation (4) in the case of hinge loss function $L(z) = (1-z)_+$ and consider 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 $\boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2) \in \mathbb{R}^{d_1 \times r} \times \mathbb{R}^{d_2 \times r}\}$ with pre-specified row and columnwise kernels K_1, K_2. Equation (9) is written as

$$\begin{aligned} &\min_{\substack{\boldsymbol{C}\in\mathcal{H}_1^{d_1}\times\mathcal{H}_2^{d_2},\\ \boldsymbol{P}\in\mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}}} \frac{1}{2}\|\boldsymbol{C}\boldsymbol{P}^T\|_F^2 + C\sum_{i=1}^n \xi_i, \\ &\text{subject to } y_i\langle \boldsymbol{C}\boldsymbol{P}^T, \Phi(\boldsymbol{X}_i)\rangle \le 1-\xi_i \text{ and } \xi_i \ge 0, i = 1,\ldots,n. \end{aligned} \tag{9}$$

We optimize Equation (9) with a coordinate descent algorithm that solves one block holding the other block fixed. Each step is a convex optimization and can be solved with quadratic programming. To be specific, first we update $\boldsymbol{C}$ holding $\boldsymbol{P}$ fixed. The dual problem of Equation (9) is

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

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

$$\boldsymbol{C} = \sum_{i=1}^n \alpha_i y_i \Phi(\boldsymbol{X}_i)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1} \in \mathcal{H}_r^r \times \mathcal{H}_c^r. \tag{10}$$

We only use formual (10) without information about feature mapping $\Phi(\cdot)$. Second, we update $\boldsymbol{P}$ holding $\boldsymbol{C}$ fixed. The dual problem of Equation (9) is

$$\max_{\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)} -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_i)\right), \boldsymbol{C}\left((\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T\Phi(\boldsymbol{X}_j)\right)\rangle, \tag{11}$$

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

We can find an optimizer of (11) without the feature mapping. To show this, notice that by plugging (10) into (11), we have

$$\begin{aligned}
\boldsymbol{C}^T\boldsymbol{C} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{K}(i,j)\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1} \in \mathbb{R}^{r\times r}\times\mathbb{R}^{r\times r}, \qquad (12)\\
\boldsymbol{C}^T\Phi(\boldsymbol{X}_i) &= \sum_{j=1}^{n}\alpha_i y_i (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{K}(i,j) \in \mathbb{R}^{r\times d_1}\times\mathbb{R}^{r\times d_2},
\end{aligned}$$

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 $[\Phi_1(\boldsymbol{X}_i)^T\Phi_1(\boldsymbol{X}_j)]_{ss'} = K_1(\boldsymbol{X}^{(i)}_{s:}, \boldsymbol{X}^{(j)}_{s':})$ and vice versa for $\Phi_2(\cdot)$. Equations in (12) gives us inner product in (11) expressed only in terms of $\boldsymbol{P}$ and $\{\boldsymbol{K}(i,j)\colon\ i,j\in[n]\}$. Therefore, we can update $\boldsymbol{P}$ from an optimal coefficient $\boldsymbol{\alpha}$ of (11) and the formulas in (12) without feature mapping specification.

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

Finally, we have the nonlinear function output of the form,

$$\begin{aligned}
\hat{f}(\boldsymbol{X}) = \sum_{k=1}^{n}\hat{\alpha}_k y_k \Bigg(&\sum_{i=1}^{d_1}\sum_{j=1}^{d_1}[\hat{\boldsymbol{P}}_1(\hat{\boldsymbol{P}}_1^T\hat{\boldsymbol{P}}_1)^{-1}\hat{\boldsymbol{P}}_1^T]_{ij}K_r\left([\boldsymbol{X}_k]_{i:}, [\boldsymbol{X}]_{j:}\right) \qquad (13)\\
&+ \sum_{i=1}^{d_2}\sum_{j=1}^{d_2}[\hat{\boldsymbol{P}}_2(\hat{\boldsymbol{P}}_2^T\hat{\boldsymbol{P}}_2)^{-1}\hat{\boldsymbol{P}}_2^T]_{ij}K_c\left([\boldsymbol{X}_k]_{:i}, [\boldsymbol{X}]_{:j}\right)\Bigg).
\end{aligned}$$

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 (13)

Algorithm gives the full description. **[FIXME (Miaoyan): Add Algorithm 2 for Regression Estimation. Put it in an algorithm environment]**

4 Theory

5 Conclusion

SUPPLEMENTARY MATERIAL

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.