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

2.1 Models and Motivation

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 [3]. 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 [1].

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 method has been ..., a lack. A key challenge with matrix data is the complex structure in the feature space. Matrix space is a tensor product space in which both rows and columns. The notions of features are not well defined for matrix space. For example, one may be interested in entrywise significance, or important vectors, or the spaces that corresponds to the matrix. Find the right representations. interpretable, adaptive, informative, ..., More relevant in non-linear learning. Bad representation leads to complicated data..., sensitive to noise. A model that is complex in the original space may projected into simpler surface in manifold. (complex function surface). Another challenge is ...Notably, the ambient dimension with matrix-valued feature is $d_1 d_2$, while the number of sample is n. Modern applications are often in the high dimensional regime where $d_1 d_2$ is comparable to, or even larger, than the sample size n. (do we allow d to increase with n in the asymptotic result?). The challenge of high-dimensionality is notably hard especially for matrix-valued features. A second challenge is the notation of informative features. While vectors ... Matrices encodes more information. such as columns space, row space, local structure, global structure, entrywise, spacewise, vs. volumne, etc. One of our contribution is to have a general framework that ...

Before we propose methods to address the challanges. Our framework is general in that it . We focus on three concrete three learning problems and the connection between them. As we will see, combinations

2.2 Three Main Learning Problems

The distribution $y|\boldsymbol{X}$ is binary, so its distribution is completely determined by the class probability $\mathbb{P}(Y = 1|\boldsymbol{X})$. Based on the properties $\mathbb{E}(y|\boldsymbol{X}) = 2\mathbb{P}(Y = 1|\boldsymbol{X}) - 1$, it is equivalent to estimate $\mathbb{E}(Y|\boldsymbol{X})$. We denote the function $\eta(\boldsymbol{X}) = \mathbb{E}(y|\boldsymbol{X})$. Three questions in increasing difficulty.

Classification. Find the level set $S_0 = \{\boldsymbol{X} : \eta(\boldsymbol{X}) \geq 0\}$.

Level-set estimation. Given $\pi \in [-1, 1]$, find the level set $S_\pi = \{\boldsymbol{X} : \eta(\boldsymbol{X}) \geq \pi\}$.

Regression: Estimate the function $\eta(\boldsymbol{X})$: $\mathbb{R}^{d_1 \times d_2} \mapsto [-1, 1]$.

Each of them does not depend on the unknown function η, and furthermore, the minimizer of decision rule is the desired properties. Problem 1 is a special case of Problem 2. Problem 3 can be addressed by.

Because log-likelihood for η is unknown, we consider the loss function that does not depend on the ground truth and distribution. The following loss can be treated as non-likelihood loss.

Example 1. *Define classification error* $\mathbb{P}[y \neq g(\boldsymbol{X})]$. *Find a decision rule* g *such that minimizes classification error in the space* $g \in \mathcal{F}$.

Example 2. *Define weighted classification error:* $\mathbb{E}[|y - \pi|\mathbb{1}\{y \neq g(\boldsymbol{X})\}]$. *Find a decision rule* g *such that minimizes classification error in the space* $g \in \mathcal{F}$.

Example 3. *Define least-squared loss* $\mathbb{E}[f(\boldsymbol{X}) - y]^2$. *Find a decision rule* g *such that minimizes classification error in the space* $f \in \mathcal{F}$.

3 Example: Trace regression model

3.1 Example: Non-linear matrix completion

$$y_{ij} = \text{sign}(f(x_{ij})), \text{ where } \quad \boldsymbol{X} = [\![x_{ij}]\!].$$

Estimate $\boldsymbol{B}$ itself recovers the binary matrix completion problem.

3.2 Example: Graph classification

Both node level and edge level. 1. projection of nodes. 2. of edges (edges are generated from eigen model.. angle??? different homophily effect to the response label)

Difference in topology. absence of presence of small hubs. Output is the projection matrices.

3.2.1 Choice of loss function

Large-margin loss which is convex of the 0-1 loss. The sample version is

$$L(g) = \sum_{i=1}^{n} (1 - y_i g(\boldsymbol{X}_i))_+.$$

Other fisher-consistent loss is also possible. Such as ..... Similarly, the weighted version is

$$L(g) = \sum_{y_i=1} (1-\pi)(1 - g(\boldsymbol{X}_i))_+ + \sum_{y_i=-1} (1+\pi)(1 - g(\boldsymbol{X}_i))_+$$

Consider the optimization problem

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

where depending on the application, the cost function $L(\cdot, \cdot)$ can be selected to be, e.g., the least-squares (LS), the logistic or the hinge loss; $J(\cdot)$ is an increasing function; and, $\lambda > 0$ is a regularization parameter that controls overfitting.

4 Matrix kernels for subspace embedding

The aforementioned three problems are formulated as empirical risk minimization over functions defined over matrices. In this section, we will propose a family of matrix kernels as building blocks for constructing functions in matrix space. Informally, a matrix kernel is a distance measure between matrices based on their similarity. Kernels defined over non-vector objects have recently evolved into a rapidly developing area of structured learning. Unlike vectors, matrix-value feature encodes a two-way relationship across rows and columns. Taking into account the two-way relationship is essential in the matrix kernel formulation.

We now propose a matrix kernel inspired by low-rank latent factor models. The use of matrix kernel in learning will be described in Section XX.

Definition 1 (Matrix Kernel)**.** *Given two matrices $\boldsymbol{M}$ and $\boldsymbol{M'}$ from the space of matrix $\mathcal{M}$. The problem of matrix comparison is to find a mapping s*

$$s\colon \mathcal{M} \times \mathcal{M} \to \mathbb{R}$$

such that $s(\boldsymbol{M}, \boldsymbol{M'})$ quantifies the similarity (or dissimilarity) of $\boldsymbol{M}$ and $\boldsymbol{M'}$.

(Add a figure)

Goal: topological description. Map each matrix to a feature vector, and use distance and metrics on vectors for learning on graphs.

Let $\boldsymbol{X} \in \mathrm{Sym}_d(\mathbb{R})$ be the feature matrix. We define a feature mapping Φ that sends a matrix in $\mathrm{Sym}_d(\mathbb{R})$ to $\mathrm{Sym}_d(\mathcal{H}^2)$. Here $\mathrm{Sym}_d(\mathcal{H}^2)$ denote the collection of d-by-d symmetric matrices with each matrix entry takes value in $\mathcal{H}^2$. Let $\boldsymbol{B} \in \mathcal{H}^{d\times d}$ be a matrix defined in $\mathcal{H}$. Replace product in $\mathbb{R}$ to inner product in $\mathcal{H}$.

5 Mahalanobis metric

1. Sum. $\boldsymbol{B} + \boldsymbol{B}' = [\![b_{ij} + b'_{ij}]\!] \in \mathcal{H}^{d\times d}$
2. Liner combination. Let $\boldsymbol{P} \in \mathbb{R}^{d\times r}$ be a real-valued matrix. Define a $\boldsymbol{BP} = [\![c_{ij}]\!] \in \mathcal{H}^{d\times r}$, where $c_{ij} = \sum_s p_{sj} b_{is} \in \mathcal{H}^2$.
3. Inner product. $\langle \boldsymbol{B}, \boldsymbol{B}' \rangle = \sum_{ij} \langle b_{ij}, b'_{ij} \rangle$.
4. Matrix product $\boldsymbol{BB}' = [\![c_{ij}]\!] \in \mathbb{R}^{d\times d}$, where $c_{ij} = \sum_s \langle b_{is},\ b'_{sj} \rangle_H$.

Here $\mathcal{H}^2 = \mathcal{H} \times \mathcal{H}$ denotes the Cartesian product of two Hilbert space. Specifically, the mapping $\boldsymbol{X} \mapsto \Phi(\boldsymbol{X}) = [\![f_{ij}]\!]$, where each element is a pair of possibly infinite dimensional features defined as

$$f_{ij} = [\Phi(\boldsymbol{X})]_{ij} \overset{\text{def}}{=} (\phi(\boldsymbol{X}_{i:}), \phi(\boldsymbol{X}_{:j})) \in \mathcal{H}^2, \quad \text{for all } i \geq j,\ (i,j) \in [d] \times [d],$$

and $f_{ji} = f_{ij}$. Note that the entry f_{ij} takes value in $\mathcal{H}^2$ for all $(i,j) \in [d] \times [d]$. Furthermore, $\Phi(\boldsymbol{X})$ is a symmetric matrix in the sense that $[\Phi(\boldsymbol{X})]_{ij} = [\Phi(\boldsymbol{X})]_{ji}$ for all (i,j). Then the decision function is defined in the high dimensional features,

$$f(\boldsymbol{X}) = \langle \mathcal{B},\ \Phi(\boldsymbol{X}) \rangle, \text{ where } \mathcal{B} \in (\mathcal{H}^2)^{d\times d}.$$

The parameter $\mathcal{B} = [\![\boldsymbol{b}_{ij}]\!]$ is a d-by-d matrix defined over $\mathcal{H}^2$; that is, each entry $\boldsymbol{b}_{ij} = (\boldsymbol{b}_{ij1}, \boldsymbol{b}_{ij2}) \in \mathcal{H}^2$ for all $(i,j) \in [d]^2$. We consider low-rank structure in $\mathcal{B} = \boldsymbol{P}^T \mathcal{C} \boldsymbol{P}$. Then low rank function space

$$\begin{aligned}
\mathcal{F} &= \left\{ f \colon \boldsymbol{X} \mapsto \langle \boldsymbol{C}, \boldsymbol{P}^T \Phi(\boldsymbol{X}) \boldsymbol{P} \rangle \,\middle|\, \boldsymbol{PP}^T = \boldsymbol{I},\ \boldsymbol{P} \in \mathbb{R}^{d\times r},\ \mathcal{C} \in (\mathcal{H}^2)^{r\times r} \right\} \\
&= \left\{ f \in \text{RKHS generated by } \mathcal{K}(\boldsymbol{P}) \,\middle|\, \boldsymbol{PP}^T = \boldsymbol{I},\ \boldsymbol{P} \in \mathbb{R}^{d\times r} \right\} \\
&= \left\{ f \in \text{RKHS generated by } \mathcal{K}(\boldsymbol{W}) \,\middle|\, \boldsymbol{W} \succeq 0,\ \text{rank}(\boldsymbol{W}) \leq r \right\}
\end{aligned}$$

The orthogonality condition is based on the relationship $\boldsymbol{W} = \boldsymbol{P}^T \Lambda^2 \boldsymbol{P}$ and scale invariance between the function $\langle \mathcal{C}, \boldsymbol{P}^T \Phi(\boldsymbol{X}) \boldsymbol{P} \rangle = \langle \Lambda \boldsymbol{P} \mathcal{C}, \Phi(\boldsymbol{X}) \boldsymbol{P} \Lambda \rangle$. The kernel

$$\begin{aligned} \mathcal{K}(\boldsymbol{W}) &= \langle \boldsymbol{P}^T \Phi(\boldsymbol{X}) \boldsymbol{P}, \boldsymbol{P}^T \Phi(\boldsymbol{X}') \boldsymbol{P} \rangle \\ &= c_{\text{col}} \mathcal{K}_{\text{row}}(\boldsymbol{P}\boldsymbol{P}^T) + c_{\text{row}} \mathcal{K}_{\text{col}}(\boldsymbol{P}\boldsymbol{P}^T) \end{aligned}$$

Properties 1. *Let $\boldsymbol{P} \in \mathbb{R}^{d \times r}$ be a projection matrix. The mapping $\boldsymbol{P} \circ \Phi = \boldsymbol{P}\Phi(\boldsymbol{X})\boldsymbol{P} \in (\mathcal{H}^2)^{r \times r}$ defines a kernel $\mathcal{K}(\boldsymbol{X}, \boldsymbol{X}') =$*

Linear function family: Represented by features. Let $\boldsymbol{X} \in \text{Sym}_d(\mathbb{R})$.

$$\begin{aligned} \mathcal{F}(r) &= \{ f \colon \boldsymbol{X} \mapsto \langle \boldsymbol{W},\ \boldsymbol{X} \rangle \ \big|\ \text{rank}(\boldsymbol{W}) \le r,\ \boldsymbol{W} \in \text{Sym}_d(\mathbb{R}) \} \\ &= \{ f \colon \boldsymbol{X} \mapsto \langle \boldsymbol{C},\ \boldsymbol{P}^T \boldsymbol{X} \boldsymbol{P} \rangle \ \big|\ \boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I},\ \boldsymbol{P} \in \mathbb{R}^{d \times r},\ \boldsymbol{C} \in \text{Sym}_r(\mathbb{R}) \} \end{aligned}$$

Definition 2 (Kernel defined in matrix space)**.** *Let $d(\cdot,\ \cdot)$ be a kernel defined in vector space $\mathbb{R}^d$, and $\boldsymbol{W} = [\![w_{ij}]\!] \in \mathbb{R}^{d \times d}$ be a rank-r semi-positive definite matrix. Then $\boldsymbol{K}$ and $\boldsymbol{W}$ induce a low-rank projection kernel $\mathcal{K}$ in matrix space:*

$$\begin{aligned} \mathcal{K} \colon \mathbb{R}^{d \times d} \times \mathbb{R}^{d \times d} &\mapsto \mathbb{R} \\ (\boldsymbol{X}, \boldsymbol{X}') &\mapsto \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{i,j \in [d]} w_{ij} d(\boldsymbol{x}_i, \boldsymbol{x}'_j), \end{aligned}$$

where $\boldsymbol{x}_i, \boldsymbol{x}'_j$ denote the i-th and j-th columns of $\boldsymbol{X}$, $\boldsymbol{X}'$, respectively.

Often, the kernel $\boldsymbol{K}$ is specified by users, whereas the projection kernel $\boldsymbol{W}$ is learned by our algorithm. In particular $\boldsymbol{W} \propto \boldsymbol{I}_{d \times d}$ corresponds to the classical vectorization case. We denote the low-rank projection kernel $\mathcal{K} = \mathcal{K}(\boldsymbol{W})$.

Properties 2 (Properties of projection kernel)**.** *The kernel family defines a set of kernels that indexed by $\boldsymbol{W}$. In particular,*

- $\mathcal{K}(\boldsymbol{W}) + \mathcal{K}(\boldsymbol{W}') = \mathcal{K}(\boldsymbol{W} + \boldsymbol{W}')$ *for all* $\boldsymbol{W}, \boldsymbol{W}' \in \mathbb{R}^{d\times d}$.
- *In the special rank-1 case* $\boldsymbol{W} = [\![w_i w_j]\!]$, *our kernel corresponds to the bandwidth selection.*
- *The kernel* $\mathcal{K}(\boldsymbol{W})$ *corresponds to a valid feature mapping* $\Phi\colon \mathbb{R}^{d\times d} \to \mathcal{H}^r$.
- *Rank-k kernel is a convex hull of rank-1 kernels.*

in total $d(d-1)/2$ terms involving interaction between row and column $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. $C = \mathcal{H}_1 + \cdots + \mathcal{H}_{d(d-1)/2}$. Then rank-$r$ projection over the left space. ....? $\boldsymbol{X}$

Consider the family of kernels

$$\mathcal{F}(r) = \left\{ f \in \text{RKHS generated by } \mathcal{K}(\boldsymbol{W}) \;\middle|\; \boldsymbol{W} \succeq 0,\ \text{rank}(\boldsymbol{W}) = r \right\}.$$

We use $\mathcal{F}(r)$ to denote the set of kernels generated by rank-r projection.

Each kernel $\mathcal{K}$ generates a RKHS. We consider the optimization over the union of RKHS. The union of RKHS is not RKHS.

$$\max_{f\in\mathcal{F}(r)} L(f) = \max_{\substack{\text{rank}(\boldsymbol{W})=r,\\ \boldsymbol{W}\succeq 0}} \max_{f\in\text{RKHS}(\mathcal{K}(\boldsymbol{W}))} L(f)$$

Connection to adaptive kernel learning The above can be viewed as an optimization of kernels

$$\begin{aligned}\max_{f\in\mathcal{F}(r)} L(f) &= \max_{f=\alpha_1 f_1+\cdots\alpha_r f_r\colon\, f_i\in\mathcal{K}(1),\alpha\in\mathbb{R}^n} L(f)\\ &= \max_{f\in\mathcal{K}(1)\oplus\cdots\oplus\mathcal{K}(1)} L(f)\end{aligned}$$

Connection to Metric learning

$$\max_{f\in\mathcal{F}(r)} L(f) = \max_{\alpha\in\Delta, f\in\text{RKHS}(\mathcal{K}(1))} L(f)$$

Ensemble methods.

Example 4 (Multiple kernel learning)**.** *Let* $\boldsymbol{X} = diag(x_i), \boldsymbol{X}' = diag(x_i')$ *are diagonal matrices and inner-product kernel* $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}') = g(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)$*. Then our rank-1 projection kernel reduces to*

$$\mathcal{K}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{i \in [d]} w_i^2 d(x_i, x_i')$$

Example 5 (Feature selection with lasso penalty)**.** *Let* $\boldsymbol{X}, \boldsymbol{X}'$ *are diagonal matrices.*

$$\mathcal{K}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{\boldsymbol{\alpha} \in \Delta(r)} \alpha_i d(x_i, x_i')$$

weighted features. Let $\mathcal{F}(1)$ *denote the RKHS induced by rank-1 projection. Then rank* $\mathcal{F}(r) = \mathcal{F}(1) \oplus \cdots \oplus \mathcal{F}(1)$

Example 6. *Group lasso on the features. Weight* $\sim$ *Multinormial*(d, r)*. Then rank-kernel is* r*.*

Example 7 (Convolution kernel)**.**

Orthogonality between rank-1 kernels. If and only if $\mathcal{F}(1) \cap \mathcal{F}(1) = 0$

Each of the kernel induces the functions defined in matrix space.

$$\begin{aligned}
\mathcal{F}(r) &= \left\{ f \in \mathcal{F}(\boldsymbol{A}) \;\middle|\; \boldsymbol{A} \succeq 0,\ \operatorname{rank}(\boldsymbol{A}) \leq r \right\} \\
&= \left\{ \boldsymbol{X} \mapsto \sum_{i,j} a_{ij} \boldsymbol{K}(\boldsymbol{x}_i,\ \cdot) \;\middle|\; \boldsymbol{A} = [\![a_{ij}]\!] \text{ is a rank-}r \text{ PSD matrix} \right\}.
\end{aligned}$$

Example 8 (Rank-1 kernel)**.**

$$\mathcal{K}(\boldsymbol{W}) = \sum_{ij} w_i w_j K(\boldsymbol{x}_i, \boldsymbol{x}_j') \quad \textit{for some } \boldsymbol{a} \in \mathbb{R}^d.$$

$$\mathcal{F} = \{ \boldsymbol{X} \mapsto \sum_{ij} a_i a_j \boldsymbol{K}(\boldsymbol{x}_i, \cdot) \;\big|\; \boldsymbol{a} \in \mathbb{R}^d \}$$

$$f \in \operatorname{Span}(\mathcal{F}) \text{ and } \operatorname{rank}(f) \leq r.$$

5.0.1 Non-Asymmetric case

$$\mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) = \sum_{(i_1,i_2,j_1,j_2)} a_{i_1 j_1} b_{j_1 j_2} d(\boldsymbol{x}_{i_1 i_2}, \boldsymbol{x}'_{j_1 j_2}) = \langle D(\boldsymbol{X}, \boldsymbol{X}'),\ \boldsymbol{P}_1 \otimes \boldsymbol{P}_2 \rangle$$

5.1 Classification

We consider linear predictors on matrix feature of the form

$$f_{\boldsymbol{B}}(\boldsymbol{X}) = \langle \boldsymbol{B}, \boldsymbol{X} \rangle,$$

where $\boldsymbol{B}, \boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$ and $\langle \boldsymbol{X}, \boldsymbol{X}' \rangle = \mathrm{Tr}(\boldsymbol{X}^T \boldsymbol{X}')$. We extend the linear model to non-linear case in Section 7. This matrix representation has advantage of regularizing the coefficient matrix $\boldsymbol{B}$ and restricting the number of parameters. Specifically, we assume that $\boldsymbol{B}$ has low-rank structure such that

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

We show how low rankness let us consider column and row wise structure of feature matrices and prevents overfitting in Section 8. Assume we are given a set of training data and label pairs $\{\boldsymbol{X}_i, y_i\}_{i=1}^n$ where $\boldsymbol{X}_i \in \mathbb{R}^{d_1 \times d_2}$ and $y_i \in \{-1, +1\}$. Our goal is to learn a model with a low error on the training data. One successful approach is large-margin classifiers. A large-margin classifier minimizes a cost function in f over a decision function class $\mathcal{F}$:

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

where $J(f)$ is a penalty 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 functions. However, our estimation schemes and theorems are applicable to general large-margin classifiers. Consider linear decision function class with low rank coefficient $\mathcal{F} = \{f : f(\cdot) = \langle \boldsymbol{U}\boldsymbol{V}, \cdot \rangle \text{ where } \boldsymbol{U} \in \mathbb{R}^{d_1 \times r}, \boldsymbol{V} \in \mathbb{R}^{d_2 \times r}\}$. The solution $f(\boldsymbol{X})$ of Equation (1) is shown to have the form (check Supplement)

$$f(\boldsymbol{X}) = \sum_{i=1}^{n} \alpha_i y_i \langle \boldsymbol{P}_r \boldsymbol{X}_i, \boldsymbol{P}_r \boldsymbol{X} \rangle,$$

where $\{\alpha_i\}_{i=1}^n$ are solution (spare) in the dual problem of Equation (1) and $\boldsymbol{P}_r \in \mathbb{R}^{r \times d_1}$ is the projection matrix induced by low rank coefficient $\boldsymbol{U}, \boldsymbol{V}$. Let $\langle \cdot, \cdot \rangle_{\boldsymbol{P}_r}$ denote the low rank linear kernel for a pair of matrices:

$$\langle \boldsymbol{X}, \boldsymbol{X}' \rangle_{\boldsymbol{P}_r} \stackrel{\text{def}}{=} \langle \boldsymbol{P}_r \boldsymbol{X}, \boldsymbol{P}_r \boldsymbol{X}' \rangle, \quad \text{for all } \boldsymbol{X}, \boldsymbol{X}' \in \mathbb{R}^{d_1 \times d_2}, .$$

Therefore, we consider the decision function of the form:

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i y_i \langle \boldsymbol{X}_i, \cdot \rangle_{\boldsymbol{P}_r}.$$

We can think of our considered classification function class as the reproducing kernel Hilbert space induced by rank-r linear kernels $\{\langle \cdot, \cdot \rangle_{\boldsymbol{P}} : \boldsymbol{P} \in \mathbb{R}^{r \times d_1} \text{ is a projection matrix}\}$. We estimate coefficient $\{\hat{\alpha}_i\}_{i=1}^n$ and the projection matrix $\hat{\boldsymbol{P}}_r$ from a given dataset. Detailed estimation algorithm appears in Section 6. We obtain our classification rule as

$$G(\boldsymbol{X}) = \text{sign}\left(\sum_{i=1}^{n} \hat{\alpha}_i y_i \langle \boldsymbol{X}_i, \boldsymbol{X} \rangle_{\hat{\boldsymbol{P}}_r} \right).$$

5.2 Probability function estimation

Our proposed method is designed to estimate $p(\boldsymbol{X}) \stackrel{\text{def}}{=} \mathbb{P}(y = 1 | \boldsymbol{X})$ at any $\boldsymbol{X}$ which does not necessarily belong to the observed training data set. We consider the weighed version

of (1),

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

where $\omega_\pi(y)=1-\pi$ if $y=1$ and π if $y=-1$. **(author?)** [2] showed that The minimizer $\hat{f}_\pi$ to Equation (2) is a consistent estimate of $\text{sign}(p(\boldsymbol{X})-\pi)$. Therefore, for a given smoothing parameter $H\in\mathbb{N}_+$, We estimate the target probability through two main steps.

$$\begin{aligned} p(\boldsymbol{X}) &\overset{\text{step1}}{\approx} \sum_{h=1}^{H}\frac{h-1}{H}\mathbb{1}\left\{\boldsymbol{X}:\frac{h-1}{H}\le p(\boldsymbol{X})<\frac{h}{H}\right\} \\ &\overset{\text{step2}}{\approx} \sum_{h=1}^{H}\frac{h-1}{H}\mathbb{1}\left\{\boldsymbol{X}:\text{sign}(\hat{f}_{\frac{h-1}{H}})=1,\text{sign}(\hat{f}_{\frac{h}{H}})=-1\right\}, \end{aligned}$$

where Step 1 approximates the target probability by linear combination of step functions and Step 2 uses the fact that the solution of (2) is consistent to Bayes rule. The probability estimation scheme can be summarized as follows.

Scheme

S.1 Choose a sequence of weight $\pi_h=\frac{h}{H}$, for $h=1,\ldots,H$.

S.2 For each weight $\pi_h\in[0,1]$, solve Equation (2) with $\omega_{\pi_h}(y)$

S.3 Denote the sequence of solutions and decision regions

$$\{\hat{f}_h\}_{h=1}^{H} \text{ and } \{\hat{\mathcal{D}}_h\}_{h=1}^{H}=\left\{\left\{\boldsymbol{X}:\text{sign}(\hat{f}_{\frac{h-1}{H}})=1,\text{sign}(\hat{f}_{\frac{h}{H}})=-1\right\}\right\}_{h=1}^{H}$$

S.4 Estimate the target probability function by

$$\hat{p}(mX)=\sum_{h=1}^{H}\frac{h-1}{H}\mathbb{1}\left\{\boldsymbol{X}\in\hat{\mathcal{D}}_h\right\}.$$

Notice that this scheme can be applied to any large-margin classifiers though we focus on the hinge loss function in this paper. Solution of weighted hinge loss in Equation (2) is solved with simple modification from classification algorithm in Section 6.

6 Algorithm

In this Section, we describe the algorithm to seek the optimizer of Equation (1) in the case of hinge loss function $L(z) = (1-z)_+$ and linear function class $\mathcal{F} = \{f : f(\cdot) = \langle \boldsymbol{U}\boldsymbol{V}^T, \cdot \rangle, \text{ where } \boldsymbol{U} \in \mathbb{R}^{d_1 \times r} \boldsymbol{V} \in \mathbb{R}^{d_2 \times r}\}$. Equation (1) is written as

$$\min_{\{(\boldsymbol{U},\boldsymbol{V}) \in \mathbb{R}^{d_1 \times r} \times \mathbb{R}^{d_2 \times r}\}} n^{-1} \sum_{i=1}^{n} \left(1 - y_i \langle \boldsymbol{U}\boldsymbol{V}^T, \boldsymbol{X}_i \rangle\right)_+ + \lambda \|\boldsymbol{U}\boldsymbol{V}^T\|_F^2 \tag{3}$$

We optimize Equation (3) 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, when we fix $\boldsymbol{V}$ and update $\boldsymbol{U}$ we have the following equivalent dual problem

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n : \boldsymbol{\alpha} \geq 0} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{X}_i, \boldsymbol{X}_j \boldsymbol{V} (\boldsymbol{V}^T \boldsymbol{V})^{-1} \boldsymbol{V}^T \rangle \right)$$
$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq \frac{1}{2\lambda n}, \quad i = 1. \cdots, n,$$

We use quadratic programming to solve this dual problem and update $\boldsymbol{U} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{X}_i \boldsymbol{V} (\boldsymbol{V}^T \boldsymbol{V})^{-1}$. Similar approach is applied to update $\boldsymbol{V}$ fixing $\boldsymbol{U}$. The Algorithm 1 gives the full description.

7 Extension to nonlinear case

We extend linear function class to non-linear class with kernel trick. We enlarge feature space through feature mapping $\boldsymbol{h} : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^{d_1 \times d_2'}$. Once this mapping fixed, the procedure is the same as before. We fit the linear classifier using pair of input feature and label

Algorithm 1: Linear classification algorithm

Input: $(\boldsymbol{X}_1, y_1), \cdots, (\boldsymbol{X}_n, y_m)$, rank r

Parameter: U,V

Initizlize: $\boldsymbol{U}^{(0)}, \boldsymbol{V}^{(0)}$

Do until converges

Update $\boldsymbol{U}$ fixing $\boldsymbol{V}$:

Solve $\max_{\boldsymbol{\alpha}} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j y_i y_j \langle \boldsymbol{X}_i, \boldsymbol{X}_j \boldsymbol{V}(\boldsymbol{V}^T\boldsymbol{V})^{-1}\boldsymbol{V}^T\rangle$.

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

Update $\boldsymbol{V}$ fixing $\boldsymbol{U}$:

Solve $\max_{\boldsymbol{\alpha}} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j y_i y_j \langle \boldsymbol{X}_i, \boldsymbol{U}(\boldsymbol{U}^T\boldsymbol{U})^{-1}\boldsymbol{U}^T \boldsymbol{X}_j\rangle$.

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

Output: $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{V}^T$

$\{\boldsymbol{h}(\boldsymbol{X}_i), y_i\}_{i=1}^n$. Define a nonlinear low rank kernel in similar way to linear case.

$$\langle \boldsymbol{X}, \boldsymbol{X}'\rangle_{\boldsymbol{P}_r,h} \stackrel{\text{def}}{=} \langle \boldsymbol{P}_r h(\boldsymbol{X}), \boldsymbol{P}_r h(\boldsymbol{X}')\rangle = \text{trace}\left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}')\boldsymbol{P}_r^T\boldsymbol{P}_r\right] \quad \text{for all } \boldsymbol{X}, \boldsymbol{X}' \in \mathbb{R}^{d_1\times d_2},$$

where $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}') \stackrel{\text{def}}{=} h(\boldsymbol{X})h^T(\boldsymbol{X}') \in \mathbb{R}^{d_1\times d_1}$ denotes the matrix product of mapped features. The solution function $f(\cdot)$ of (1) on enlarged feature can be written

$$f(\cdot) = \sum_{i=1}^n \alpha_i y_i \langle \boldsymbol{P}_r h(\boldsymbol{X}_i), \boldsymbol{P}_r h(\cdot)\rangle = \sum_{i=1}^n \alpha_i y_i \langle \boldsymbol{X}_i, \cdot\rangle_{\boldsymbol{P}_r,h} = \sum_{i=1}^n \alpha_i y_i \text{trace}\left[\boldsymbol{K}(\boldsymbol{X}_i, \cdot)\ \boldsymbol{P}_r^T\boldsymbol{P}_r\right],$$

which involves feature mapping $h(\boldsymbol{X})$ only thorough inner products. In fact, we need not specify the the transformation $h(\boldsymbol{X})$ at all but only requires knowledge of the $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}')$. A sufficient condition and a necessary condition for $\boldsymbol{K}$ being reasonable appear in Supplement. Three popular choices for $\boldsymbol{K}$ are

- Linear kernel: $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}') = \boldsymbol{X}\boldsymbol{X}'^T$.
- Polynomial kernel with degree m: $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}') = (\boldsymbol{X}\boldsymbol{X}'^T + \lambda \boldsymbol{I})^{\circ m}$.
- Gaussian kernel: the (i, j)-th entry of $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}')$ is

$$[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}')]_{(i,j)} = \exp\left\{-\frac{1}{2\sigma^2}\|\boldsymbol{X}[i, :] - \boldsymbol{X}'[j, :]\|_2^2\right\}$$

  for all $(i, j) \in [d_1] \times [d_1]$.

One can check detailed description for non-linear case algorithm in Supplement.

8 Theory

9 Conclusion

SUPPLEMENTARY MATERIAL

References

[1] Amnon Shashua, Yoram Gdalyahu, and Gaby Hayun. Pedestrian detection for driving assistance systems: single-frame classification and system level performance. *IEEE Intelligent Vehicles Symposium, 2004*, pages 1–6, 2004.

[2] Junhui Wang, Xiaotong Shen, and Yufeng Liu. Probability estimation for large-margin classifiers. *Biometrika*, 95(1):149–167, 2008.

[3] Hua Zhou and Lexin Li. Regularized matrix regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(2):463–483, 2014.