# Large margin classification and probability estimation for matrix feature

**Abstract**

The text of your abstract. 200 or fewer words.

*Keywords:* 3 to 6 keywords, that do not appear in the title

General Principle:
1. Free-write. Your goal initially should be to "write" rather than to "write well".
2. draft an outline first, but don't be afraid to deviate from your plan if you suddenly get a new idea.
Outlines should be good enough to use but cheap enough to throw away.
3. Write in chunks. e.g. one day for one section / one central piece.

# 1 Introduction

# 2 Methods

We derive our methodology for classification and probability estimation on a set of data matrices. In particular, we propose a large-margin classifier considering matrix features. Based on the new classification method, we suggest training a series of weighed classifiers and using them to construct the probability estimation.

## 2.1 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 4. 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 5. 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:

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

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 3. We obtain our classification rule as

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

## 2.2 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 $\pi$ if $y = -1$. Wang et al. (2008) 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.

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

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.

**Shceme**

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 3.

# 3 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 $\mathcal{F} = \{f : f(\cdot) = \langle \boldsymbol{U}\boldsymbol{V}^T, \cdot \rangle,$ 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.

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

# 4 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 $\{\boldsymbol{h}(\boldsymbol{X}_i), y_i\}_{i=1}^n$. 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.

# 5 Theory

# 6 Conclusion

## SUPPLEMENTARY MATERIAL

# References

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