University of Oxford

High Dimensional Covariate Shift in Disease Risk Prediction

by

Shuhan Liu

Wolfson College

A dissertation submitted in partial fulfilment of the degree of Master of Science in Statistical Science.

Department of Statistics, 24–29 St Giles,
Oxford, OX1 3LB

September 2020

Abstract

Most statistical learning models are constructed on the assumption that the distribution of the training set is the same as the test set. Hence a model's performance on the training samples is regarded as an effective indicator for its performance in predicting test samples. However, this assumption is not held perfectly in practice. For example, in medical experiments, certain laboratory conditions and group control may be applied throughout the data collection, but may not be expected after being deployed outside lab. Also, samples may be drawn over a long period of time such that the old data is no longer representative to the more recent ones. This issue is known as covariate shift and can result in poor predictive performance. Covariate shift is common in genetics research where the majority of samples are of White ethnicity, but the results need to be employed to a wider range of populations. In this report, we investigate covariate shift adaption for high-dimensional data by reweighting training samples. We explore a range of supervised dimension reduction techniques, together with various density ratio estimation algorithms. Using an artificial dataset and the UK Biobank genotyping dataset, we compare the effectiveness of our proposed models to the ordinary unweighted model.

Acknowledgements

I would like to express my most sincere gratitude toward my supervisor, Dr. Brieuc Lehmann, who has always be reliable and responsive. He has continuously guided me throughout this project, replied to my queries with great patience and discussed in details about potential solutions. This project would not been done without his help and kindness.

I would also like to thank Yuqing Long, who has always been supportive and inspiring, I genuinely appreciate his help throughout the years. Also many thanks to Alex, Natalia and Weixi, the year in Oxford would have a lot less joy without them.

Contents

1 Introduction

1.1 Background

The lack of ethnicity diversity in genome-wide association studies (GWASs) is well-documented. A study in 2009 showed that, 96% of participants in GWASs are of European descendants [Need and Goldstein (2009)]. Although this proportion dropped to 81% in 2016, the increasing proportion of non-European participants are mostly Asian descendants. Remaining ancestries such as African, Hispanic & Latin American and Arab & Middle Eastern are still under-represented [Popejoy and Fullerton (2016)]. Conducting GWASs in European ancestry was a practical starting point given the availability of samples and the limitations in funding, genotyping technologies, and analytic methods [Peterson et al. (2019)]. However, the predictive power of GWAS findings and genetic diagnostic accuracy are weakened when transferring to other ancestries. A recent work examined the transferability of single-ancestry GWASs by calculating polygenic risk scores for eight well-studied phenotypes, finding that results from large-scale GWASs may have limited portability to other populations using standard approaches, and highlighting the need for generalised risk prediction methods [Martin et al. (2017)].

Recent methodological advances have leveraged patterns of global and local ancestry for improved variant association power [Pasaniuc et al. (2011)] and enhanced identification of causal variants [Zaitlen et al. (2010)]. However, there is limited success in identifying genome-wide significant genotypes associated with disease [Martin et al. (2017)]. There is an ongoing effort to disentangle the role of demography in heritability, linear mixed models and polygenic risk prediction [Vilhjálmsson et al. (2015), Carlson et al. (2013), Martin et al. (2019)]. With a similar objective, this report aims to explore the transfer learning methods which allow improved generalisation in ancestrally diverse populations, so that algorithms constructed on data of White individuals (mostly White British) can be deployed to other ethnic backgrounds.

Genotyping datasets exhibit some common characteristics. For example, each individual consists of a large number of variants, although only a small fraction of genotypes have a causal relationship to a specific phenotype. Genotypes from different populations also differ in distributions. Moreover, contiguous variants are likely to be correlated. A detailed illustration of these features is provided in Chapter 5 using the UK Biobank dataset.

1.2 Covariate Shift Correction

Given that in GWAS, an analysis may be carried out using samples that are mainly Europeans, then in practice are deployed to individuals from other populations that have very different features from the training samples. One possible remedy to improve the transferability is adopting a covariate shift correction [Shimodaira (2000)], introduced as follows:

Suppose we have n_{tr} labelled training samples, denoted as $(x_1^{tr}, y_1^{tr}), \cdots, (x_{n_{tr}}^{tr}, y_{n_{tr}}^{tr}) \subseteq \mathcal{X} \times \mathcal{Y}$, with joint distribution P_{XY}^{tr}. The n_{te} test samples are $(x_1^{te}, y_1^{te}), \cdots, (x_{n_{te}}^{te}, y_{n_{te}}^{te}) \subseteq$

$\mathcal{X} \times \mathcal{Y}$, which are generated from P_{XY}^{te}. The labels y_{te} are unobserved and we aim to predict them from features x_{te}. Covariate shift refers to the two following assumptions: (1) $P_{XY}^{tr} \neq P_{XY}^{te}$, (2) the distributions of labels conditional on features remain unchanged in both training and test domains, i.e. $P_{Y|X}^{tr} = P_{Y|X}^{te}$, and therefore the joint distributions differ only via the marginal distributions P_X^{tr} and P_X^{te}.

For a model $f \in \mathcal{F}$, let $L(f(x), y)$ be a point-wise loss function that measures the discrepancy between the true label y and the predicted label $f(x)$. The goal is to minimise the risk under the test data distribution:

$$R^{te} = \mathbb{E}_{P_{XY}^{te}}[L(f(x), y)].$$

In practice, the joint distribution P_{XY}^{te} is unknown, so the above risk is approximated by its empirical risk:

$$R_{emp}^{te} = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} L(f(x_i^{tr}), y_i^{tr})$$

Under covariate shift, standard learning methods such as maximal likelihood estimation are no longer consistent, the same is true for the above empirical risk minimisation (in the sense that empirical risk does not converge to its optimal solution when the size of training samples tends to infinity) [Shimodaira (2000)]. Nonetheless, this inconsistency can be eliminated by expressing the empirical risk as an importance-weighted equation:

$$R_{emp}^{te}(\beta) = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta(x_i^{tr}) L(f(x_i^{tr}), y_i^{tr})$$

where $\beta(x) = \frac{P_X^{te}(x)}{P_X^{tr}(x)}$. This is demonstrated as follows:

$$\begin{aligned}
R^{te} &= \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{XY}^{te} \cdot L(f(x), y) dx dy \\
&= \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{XY}^{tr} \cdot \frac{P_{XY}^{te}}{P_{XY}^{tr}} \cdot L(f(x), y) dx dy \\
&= \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{XY}^{tr} \cdot \frac{P_X^{te}}{P_X^{tr}} \cdot L(f(x), y) dx dy \qquad \text{from assumption (2)} \\
&= \mathbb{E}_{P_{XY}^{tr}}[\beta(x) \cdot L(f(x), y)] \\
&= R^{tr}(\beta) \\
&\approx \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta(x_i^{tr}) L(f(x_i^{tr}), y_i^{tr}).
\end{aligned}$$

Thus we can account for the the distribution difference between source and target domains by assigning each training sample a weight β, such that the re-weighted training data have the same distribution as the target domain samples. However, to carry out importance weighting, it is required to compute the density ratio $\frac{P_X^{te}(x)}{P_X^{tr}(x)}$ which neither the numerator nor the denominator is given explicitly. A naive approach would be to estimate the training and test domain densities separately using flexible methods such as kernel density estimation [Sheather and Jones (1991)], then derive the importance by taking ratio of the estimated densities. However, this approach is ineffective due to difficulty in estimating density accurately, especially in high dimensional domains [Fishman (2013)]. Moreover, division by an estimated quantity can lead to magnified estimation error [Sugiyama et al. (2010a)].

Hence, it is imperative to avoid explicit estimation of distribution densities when learning the density ratio. Estimating the densities is more general than estimating the density ratios since once the former is computed accurately, the latter is easily derived, but not the vice versa [Sugiyama et al. (2010a)]. In this report, we explore some algorithms to estimate this density ratio accurately by rewriting the ratio as a linear or non-linear model, without the need to know the distribution of features in either the training or test domains.

The outline of this report is as follows:

- Chapter 2 discusses the models for estimating the density ratio without explicitly computing the probability densities separately, then illustrates their performance in high dimensional feature space.
- Chapter 3 explains why supervised dimension reduction methods are more effective than unsupervised ones in our context, then explores a number supervised methods.
- Chapter 4 presents the results derived from experimenting on an artificial genotyping dataset and discusses the implications.
- Chapter 5 introduces the UK Biobank dataset, then provides the experiment results.
- Chapter 6 outlines some of future work directions.
- Chapter 7 gives a summary of methods we used in this report and their performance.

2 Importance Estimation

As mentioned in section 1.2, to perform covariate shift correction, it is crucial to obtain accurate estimates of the density-ratio. Here, we focus on three density-ratio estimation techniques: Kernel Mean Matching (KMM), Unconstrained Least-squares Importance Fitting (uLSIF) and Kullback-Leibler Importance Estimation Procedure (KLIEP). Each of these methods formulates the density ratio as some linear or non-linear models and computes the ratio explicitly. Further details are shown in the following.

As before, assume we have n_{tr} training samples $(x_1^{tr}, y_1^{tr}), \cdots, (x_{n_{tr}}^{tr}, y_{n_{tr}}^{tr}) \subseteq \mathcal{X} \times \mathcal{Y}$ and n_{te} test samples $(x_1^{te}, y_1^{te}), \cdots, (x_{n_{te}}^{te}, y_{n_{te}}^{te}) \subseteq \mathcal{X} \times \mathcal{Y}$, drawn from distributions P_{XY}^{tr} and P_{XY}^{te} respectively. Also suppose the conditional distributions for the two domains are the same, i.e. $P_{Y|X}^{tr} = P_{Y|X}^{te}$. We aim to find the true density ratio, expressed as $\beta(x) = \frac{P_X^{te}(x)}{P_X^{tr}(x)}$.

2.1 Kernel Mean Matching (KMM)

The moment matching method exploits fact that two distributions are equivalent if and only if all their moments are the same. For the true density ratio, we have $\beta(x)P_X^{tr}(x) = P_X^{te}(x)$, i.e. all moments of $\beta(x)P_X^{tr}(x)$ and $P_X^{te}(x)$ match each other. Therefore, one simple implementation is to find the estimate of $\beta(x)$ such that first-order moment (i.e. the mean) of $\beta(x)P_X^{tr}(x)$, is compatible with $P_X^{te}(x)$ [Gretton et al. (2009), Sugiyama et al. (2010b)].

2.1.1 Finite-order Approach

The first moment implementation in finite order can be expressed as

$$\underset{\beta}{\operatorname{argmin}} \left\| \int \boldsymbol{x}\beta(\boldsymbol{x})p_{\text{tr}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int \boldsymbol{x}p_{\text{te}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \right\|^2$$

where $\|\cdot\|$ represents the Euclidean norm. Given some non-linear function $\phi(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}^m$, the non-linear variant of the above equation is expressed as

$$\begin{aligned}
&\underset{\beta}{\operatorname{argmin}} \left\| \int \phi(\boldsymbol{x})\beta(\boldsymbol{x})p_{\text{tr}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int \phi(\boldsymbol{x})p_{\text{te}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \right\|^2 \\
&= \underset{\beta}{\operatorname{argmin}} \left\| \int \phi(\boldsymbol{x})\beta(\boldsymbol{x})p_{\text{tr}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \right\|^2 - 2\left\langle \int \boldsymbol{\phi}(\boldsymbol{x})\beta(\boldsymbol{x})p_{\text{tr}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \int \boldsymbol{\phi}(\boldsymbol{x})p_{\text{te}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \right\rangle + \text{constant}
\end{aligned}$$

In practice, we only have a finite number of samples, thus the theoretical average is approximated with its sample average. Using the vector representation, the above equation can be shown as:

$$\hat{\boldsymbol{\beta}}_{\text{tr}} := \underset{\boldsymbol{\beta}\in\mathbb{R}^{n_{tr}}}{\operatorname{argmin}} \widehat{MM}(\boldsymbol{\beta}), \quad \text{where} \quad \widehat{MM}(\boldsymbol{\beta}) := \frac{1}{n_{\text{tr}}^2}\boldsymbol{\beta}^\top \Phi_{\text{tr}}^\top \Phi_{\text{tr}}\boldsymbol{\beta} - \frac{2}{n_{\text{tr}}n_{\text{te}}}\boldsymbol{\beta}^\top \Phi_{\text{tr}}^\top \Phi_{\text{te}} 1_{n_{\text{te}}} \tag{1}$$

where $\boldsymbol{\Phi}_{\text{te}} = \left(\boldsymbol{\phi}\left(\boldsymbol{x}_1^{\text{te}}\right), \cdots, \boldsymbol{\phi}\left(\boldsymbol{x}_{n_{\text{te}}}^{\text{te}}\right)\right)$ and $\boldsymbol{\Phi}_{\text{tr}} = \left(\boldsymbol{\phi}\left(\boldsymbol{x}_1^{\text{tr}}\right), \ldots, \boldsymbol{\phi}\left(\boldsymbol{x}_{n_{\text{tr}}}^{\text{tr}}\right)\right)$, MM stands for moment matching and $\widehat{MM}$ denotes its empirical estimate, $\mathbf{1}_n$ denotes the n-dimensional vector with all ones.

The optimal solution of $\boldsymbol{\beta}$ can be found by solving $\widehat{MM}'(\boldsymbol{\beta}) = 0$.

$$\frac{2}{n_{\text{tr}}^2}\Phi_{\text{tr}}^\top\Phi_{\text{tr}}\beta - \frac{2}{n_{\text{tr}}n_{\text{te}}}\Phi_{\text{tr}}^\top\Phi_{\text{te}}1_{n_{\text{te}}} = 0 \tag{2}$$

We must add a non-negative constraint to this optimisation problem: $\boldsymbol{\beta} \geq 0$ since the ratios cannot be negative. We also impose upper bound B such that $\boldsymbol{\beta} \leq B$, B works as a regulariser that prevents the algorithm assigning very heavy weights to a few samples and causing a small effective sample size. This quadratic optimisation problem is linearly constrained and can be numerically solved by standard optimisation algorithms.

By using the algorithm we described, we can estimate the sample importance $\boldsymbol{\beta}$ at every data point in the training set. However, in order to make predictions on new samples out of our current dataset, we need to estimate the entire density ratio function, which can be done by assuming the following linear density-ratio model [Kanamori et al. (2009)]

$$\hat{\beta}(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{x})^\top\boldsymbol{\theta}$$

where $\boldsymbol{\psi}(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}^b$ is some basis function that is assumed to be non-negative and $\boldsymbol{\theta} \in \mathcal{R}^b$ is some non-negative model parameter. Then we can express the density ratio as $\left(\beta\left(x_1^{\text{tr}}\right), \ldots, \beta\left(x_{n_{\text{tr}}}^{\text{tr}}\right)\right)^\top = \boldsymbol{\Psi}_{\text{tr}}^\top\boldsymbol{\theta}$ where $\boldsymbol{\Psi}_{\text{te}} := \left(\boldsymbol{\psi}\left(x_1^{\text{te}}\right), \ldots, \boldsymbol{\psi}\left(x_{n_{\text{te}}}^{\text{te}}\right)\right)$. The optimisation problem is computed in terms of $\boldsymbol{\theta}$ and can be expressed as

$$\widehat{\boldsymbol{\theta}} := \underset{\boldsymbol{\theta}\in\mathbb{R}^b}{\operatorname{argmin}}\left[\frac{1}{n_{\text{tr}}^2}\boldsymbol{\theta}^\top\boldsymbol{\Psi}_{\text{tr}}\boldsymbol{\Phi}_{\text{tr}}^\top\boldsymbol{\Phi}_{\text{tr}}\boldsymbol{\Psi}_{\text{tr}}^\top\boldsymbol{\theta} - \frac{2}{n_{\text{tr}}n_{\text{te}}}\boldsymbol{\theta}^\top\boldsymbol{\Psi}_{\text{tr}}\boldsymbol{\Phi}_{\text{tr}}^\top\boldsymbol{\Phi}_{\text{te}}\mathbf{1}_{n_{\text{te}}}\right]$$

and the analytical solution for $\widehat{\boldsymbol{\theta}}$ is

$$\begin{gathered}\widehat{\boldsymbol{\theta}} = \frac{n_{tr}}{n_{te}}\left(\boldsymbol{\Psi}_{\text{tr}}\boldsymbol{\Phi}_{\text{tr}}^\top\boldsymbol{\Phi}_{\text{tr}}\boldsymbol{\Psi}_{\text{tr}}^\top\right)^{-1}\boldsymbol{\Psi}_{\text{tr}}\boldsymbol{\Phi}_{\text{tr}}^\top\boldsymbol{\Phi}_{\text{te}}\mathbf{1}_{n_{\text{te}}}\\ \text{subject to} \qquad \boldsymbol{\theta} \geq \mathbf{0}, \qquad \text{and} \qquad \boldsymbol{\theta} \leq B\end{gathered}$$

The upper bound B can be found via cross-validation. The finite-order approach is simple and computationally efficient. However, it is not always consistent and does not guarantee to give the true density ratio. To ensure the consistency, we need to compare all the moments up to the infinite order and this motivates us to incorporate kernel methods into our current algorithm.

2.1.2 Infinite-order Approach

The infinite-order version, namely, the Kernel Mean Matching (KMM) method, uses a universal reproducing kernel $K\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ as a non-linear transformation. [Gretton et al.

(2009)] shows that the solution of the optimal density ratio estimate can be derived by solving

$$\min_{r\in\mathcal{H}} \left\| \int K(x,\cdot)p_{\rm te}(x)\mathrm{d}x - \int K(x,\cdot)\beta(x)p_{\rm tr}(x)\mathrm{d}x \right\|_{\mathcal{H}}^2$$

Again, in practice with finite number of samples, this optimisation problem is reduced to

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_{tr}}} \left[\frac{1}{n_{\rm tr}^2}\boldsymbol{\beta}^\top \boldsymbol{K}_{\rm tr,tr}\boldsymbol{\beta} - \frac{2}{n_{\rm tr}n_{\rm te}}\boldsymbol{\beta}^\top \boldsymbol{K}_{\rm tr,te}\mathbf{1}_{n_{\rm te}} \right]$$

where $\boldsymbol{K}_{\rm tr,tr}$ and $\boldsymbol{K}_{\rm tr,te}$ are the two Gram matrices defined as $(\boldsymbol{K}_{\rm tr,tr})_{ij} = K(x_i^{tr}, x_j^{tr})$ and $(\boldsymbol{K}_{\rm tr,te})_{ij} = K(x_i^{tr}, x_j^{te})$

By taking the first order derivative and setting it to zero, we obtain

$$\widehat{\boldsymbol{\beta}}_{\rm tr} = \frac{n_{\rm tr}}{n_{\rm te}}\boldsymbol{K}_{\rm tr,tr}^{-1}\boldsymbol{K}_{\rm tr,te}\mathbf{1}_{n_{\rm te}}$$

subject to the upper bound B and lower bound 0 on $\boldsymbol{\beta}$.

For new samples, we use the linear density ratio model, together with the kernel functions:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^b} \left[\frac{1}{n_{\rm tr}^2}\boldsymbol{\theta}^\top \boldsymbol{\Psi}_{\rm tr}\boldsymbol{K}_{\rm tr,tr}\boldsymbol{\Psi}_{\rm tr}^\top\boldsymbol{\theta} - \frac{2}{n_{\rm tr}n_{\rm te}}\boldsymbol{\theta}^\top \boldsymbol{\Psi}_{\rm tr}\boldsymbol{K}_{\rm tr,te}\mathbf{1}_{n_{\rm te}} \right]$$

subject to $\boldsymbol{\theta} \geq 0$ and $\boldsymbol{\theta} \leq B$.

The KMM method is always consistent as long as the reproducing kernels we choose are universal. Also, it is formulated as a convex quadratic programming problem, which always leads to a unique global solution. However, its performance heavily depends on the kernel we choose and there is no reliable method to tune the kernel parameters, for example, the kernel width in a Gaussian kernel. This is because we cannot in practice find a Gaussian width that fits all moments well and hence standard methods such as cross-validation are ineffective. One practical strategy uses the median distance between the samples as the Gaussian bandwidth [Smola and Schölkopf (1998)], although there is lack of justification for this heuristic.

2.2 Kullback-Leibler Importance Estimation Procedure (KLIEP)

The Kullback-Leibler Importance Estimation Procedure (KLIEP) is based on the minimisation of the Kullback-Leibler divergence between the distribution of reweighed training features and the distribution of test features. Compared to KMM, one noticeable advantage of KLIEP is that its model selection (such as choosing kernel parameters) can be carried out via cross-validation procedure, thus leaving no untuned parameter. This technique is introduced as follows [Tsuboi et al. (2009), Sugiyama et al. (2010b)].

Similar to KMM, we first rewrite the density ratio as a linear model

$$\hat{\beta}(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{x})^\top\boldsymbol{\theta} \tag{3}$$

where $\boldsymbol{\psi}(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}^b$ is some basis function and $\boldsymbol{\theta} \in \mathcal{R}^b$ is the model parameter, assuming both $\boldsymbol{\theta}$ and $\boldsymbol{\psi}(\boldsymbol{x})$ to be non-negative. We learn the parameter $\boldsymbol{\theta}$ such that the KL divergence between $\hat{\beta}(x)P_X^{tr}(x)$ and $P_X^{te}(x)$ is minimised.

$$\begin{aligned}\mathrm{KL}\left[p_{\mathrm{te}}(\boldsymbol{x}) \| \hat{\beta}(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x})\right] &= \int_{\mathcal{X}} p_{\mathrm{te}}(\boldsymbol{x}) \log \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})\hat{\beta}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \int_{\mathcal{X}} p_{\mathrm{te}}(\boldsymbol{x}) \log \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} d\boldsymbol{x} - \int_{\mathcal{X}} p_{\mathrm{te}}(\boldsymbol{x}) \log \hat{\beta}(\boldsymbol{x}) d\boldsymbol{x}\end{aligned}$$

The first term is independent of $\hat{\beta}(\boldsymbol{x})$ and thus can be ignored from our optimisation problem. For the second term, we denote it as J_{KLIEP}

$$J_{\mathrm{KLIEP}} = \int_{\mathcal{X}} p_{\mathrm{te}}(\boldsymbol{x}) \log \hat{\beta}(\boldsymbol{x}) d\boldsymbol{x}$$

Since the population distribution is usually unknown, in practice we approximate this quantity using its empirical value:

$$\begin{aligned}\widehat{J_{\mathrm{KLIEP}}} &= \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{te}}} \log \hat{\beta}(\boldsymbol{x}) && (4) \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{te}}} \log \left(\boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\theta}\right) \qquad \text{from equation (3)} && (5)\end{aligned}$$

We aim to maximise equation 4 in order to minimise KL divergence. Therefore, the optimisation problem is formulated as

$$\begin{aligned}&\underset{\boldsymbol{\theta}}{\text{maximize}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{te}}} \log \left(\boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\theta}\right) \\ &\text{subject to} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\theta} = N_{\mathrm{tr}} \quad \text{and} \quad \boldsymbol{\theta} \geq \mathbf{0}\end{aligned}$$

$\boldsymbol{\theta} \geq \mathbf{0}$ ensures the density-ratio estimates are non-negative. The first constraint is from the fact that the probability density function $p_{\mathrm{te}}(\boldsymbol{x})$ integrates to 1, i.e.

$$\begin{aligned}1 &= \int_{\mathcal{X}} p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int_{\mathcal{X}} p_{\mathrm{tr}}(\boldsymbol{x}) \hat{\beta}(\boldsymbol{x}) d\boldsymbol{x} \\ &\approx \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{tr}}} \hat{\beta}(\boldsymbol{x}) \qquad \text{approximated by its empirical value} \\ &= \boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\theta} \qquad \text{from equation (3)}\end{aligned}$$

This is a convex optimisation problem and hence a global optimum is guaranteed. The solution can be obtained via iteratively applying gradient ascent and checking whether the constraints are satisfied at each iteration. The algorithm is illustrated using the following pseudo-code:

Algorithm 1: KL Importance Estimation Procedure

Input: Data samples $D^{\text{te}} = \{\boldsymbol{x}_i^{\text{te}}\}_{i=1}^{n_{\text{te}}}$ and $D^{\text{tr}} = \left\{\boldsymbol{x}_j^{\text{tr}}\right\}_{j=1}^{n_{\text{tr}}}$ and basis function $\psi(\boldsymbol{x})$.

Output: The density ratio estimator $\hat{\beta}(\boldsymbol{x})$.

```
1  $\boldsymbol{\Psi}_{te} \longleftarrow \left(\boldsymbol{\psi}\left(\boldsymbol{x}_1^{te}\right), \ldots, \boldsymbol{\psi}\left(\boldsymbol{x}_{n_{te}}^{\text{te}}\right)\right)^\top$ ;
2  $\bar{\boldsymbol{\psi}}_{\text{tr}} \longleftarrow \frac{1}{n_{\text{tr}}} \sum_{j=1}^{n_{\text{tr}}} \psi\left(\boldsymbol{x}_j^{\text{tr}}\right)$ ;
3  Initialize $\boldsymbol{\theta}\,(> \mathbf{0})$ and $\varepsilon(0 < \varepsilon \ll 1)$ ;
4  repeat
5      $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \varepsilon \boldsymbol{\Psi}_{\text{te}}^\top (\mathbf{1}_{n_{\text{te}}} / \boldsymbol{\Psi}_{\text{te}} \boldsymbol{\theta})$ ;                // gradient ascent
6      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \left(1 - \overline{\boldsymbol{\psi}}_{\text{tr}}^\top \boldsymbol{\theta}\right) \overline{\boldsymbol{\psi}}_{\text{tr}} / \left(\overline{\boldsymbol{\psi}}_{\text{tr}}^\top \overline{\boldsymbol{\psi}}_{\text{tr}}\right)$ ;    // Constraint satisfaction
7      $\boldsymbol{\theta} \leftarrow \max(\mathbf{0}, \boldsymbol{\theta})$ ;                      // Constraint satisfaction
8      $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} / \left(\bar{\boldsymbol{\psi}}_{\text{tr}}^\top \boldsymbol{\theta}\right)$ ;                   // Constraint satisfaction
9  until convergence;
10 $\hat{\beta}(\boldsymbol{x}) \longleftarrow \boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\theta}$
```

2.2.1 Model Selection by Cross-validation

The performance of KLIEP depends on the choice of the basis functions $\psi(\boldsymbol{x})$ as well as the parameters of the basis functions. Since the only constraint on the basis function is its non-negativeness, one reasonable choice is to use a Gaussian kernel

$$\beta(\boldsymbol{x}) = \sum_{\ell=1}^{n_{\text{te}}} \theta_\ell K\left(\boldsymbol{x}, \boldsymbol{x}_\ell^{\text{te}}\right)$$

Notice that we only use the test set data as centres of the Gaussian kernel. This is because $\beta(\boldsymbol{x})$ tends to have large values if $P_X^{te}(x)$ is large and $P_X^{te}(x)$ is small, while $\beta(\boldsymbol{x})$ tends to be zero if the opposite. Thus a small number of kernels is needed in the target region that is small even close to 0 and a larger number of kernels required when the values at the target region is large. This choice may therefore achieve a more reasonable allocation of kernels. Furthermore, for the sake of computational efficiency when the test set sample size is large, it is possible to draw a random subset from$\{x_i^{te}\}_{i=1}^{n_{te}}$ as the Gaussian centres rather than using the entire dataset.

We can hence choose the model that maximises J_{KLIEP}. For different models (e.g. models with different Gaussian kernel width), the values of J_{KLIEP} can be approximated via cross-validation. To be specific, we split the test set samples $\boldsymbol{X}_{te}$ into T folds $\{\boldsymbol{X}_{te}^t\}_{t=1}^T$, then compute the estimated density ratios $\hat{\beta}^t(\boldsymbol{x})$ from $\left\{\boldsymbol{X}_{\text{te}}^k\right\}_{k\neq t}^T$ and estimate the J_{KLIEP} score using

$$\hat{J}_{\text{KLIEP}}^t = \frac{1}{|\boldsymbol{X}_{\text{te}}^t|} \sum_{x \in \boldsymbol{X}_{\text{te}}^{\text{t}}} \hat{\beta}^t(\boldsymbol{x})$$

We repeat this procedure for each fold and each model, then select the one which has the maximum average $\widehat{J_{\text{KLIEP}}}$ across all the folds. The pseudo-code for this cross-validation is given as follows:

Algorithm 2: cross-validation for KLIEP

```
Input: Data samples $D^{\mathrm{te}} = \{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}}$ and $D^{\mathrm{tr}} = \left\{\boldsymbol{x}_j^{\mathrm{tr}}\right\}_{j=1}^{n_{\mathrm{tr}}}$ and basis function $\psi(\boldsymbol{x})$.
Output: The density ratio estimator $\widehat{\beta}(\boldsymbol{x})$.
1  Split $D^{\mathrm{te}} = \{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}}$ into $T$ disjoint subsets $\{\boldsymbol{X}_{te}^t\}_{t=1}^{T}$;
2  for Model $m = 1, \cdots, M$ do
3      for each fold $t = 1, \cdots, T$ do
4          $\hat{\beta}_t(\boldsymbol{x}) \leftarrow$ KLIEP $\left(\left\{\boldsymbol{X}_{\mathrm{te}}^k\right\}_{k \neq t}^{T}, \boldsymbol{X}^{\mathrm{tr}}, \boldsymbol{\psi}(\boldsymbol{x})\right)$;
5          $\widehat{\mathrm{J}_{\mathrm{KLIEP}}}^{t}(m) \longleftarrow \frac{1}{|\mathcal{X}_t^{\mathrm{te}}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_t^{\mathrm{te}}} \log \widehat{\beta}_t(\boldsymbol{x})$
6      end
7      $\widehat{\mathrm{J}_{\mathrm{KLIEP}}}(m) \longleftarrow \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathrm{J}_{\mathrm{KLIEP}}}^{t}(m)$
8  end
9  $\hat{m} \longleftarrow \mathrm{argmax}_m \widehat{\mathrm{J}_{\mathrm{KLIEP}}}(m)$;
10 $\widehat{\beta}(\boldsymbol{x}) \longleftrightarrow$ KLIEP $(D^{\mathrm{te}}, D^{\mathrm{tr}}, \boldsymbol{\psi}_{\widehat{m}}(\boldsymbol{x}))$
```

2.3 Unconstrained Least-square Importance Fitting (uLSIF)

Unconstrained Least-square Importance Fitting (uLSIF) [Yamada et al. (2013)] is similar to KLIEP except the former aims to minimise the squared error rather than the KL divergence. Again, we assume the following linear model for the density ratio

$$\hat{\beta}(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{x})^{\top}\boldsymbol{\theta}$$

where $\boldsymbol{\psi}(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}^b$ is some non-negative basis function and $\boldsymbol{\theta} \in \mathcal{R}^b$ is the non-negative model parameter.

We learn the parameters $\boldsymbol{\theta} \in \mathcal{R}^b$ in this linear model such that the squared error J_{uLSIF} is minimised:

$$\begin{aligned} J_{uLSIF}(\boldsymbol{\theta}) &:= \frac{1}{2}\int_{\mathcal{X}} (\widehat{\beta}(\boldsymbol{x}) - \beta(\boldsymbol{x}))^2 p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} \\ &= \frac{1}{2}\int_{\mathcal{X}_{tr}} \widehat{\beta}(\boldsymbol{x})^2 p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} - \int_{\mathcal{X}_{te}} \widehat{\beta}(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} + \frac{1}{2}\int_{\mathcal{X}_{te}} \beta(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} \end{aligned}$$

The last component this equation does not contain $\widehat{\beta}$ and hence can be ignored in our optimisation problem. We aim to minimise

$$J(\boldsymbol{\theta}) := \frac{1}{2}\int_{\mathcal{X}_{tr}} \widehat{\beta}(\boldsymbol{x})^2 p_{\mathrm{tr}}(\boldsymbol{\beta}) d\boldsymbol{x} - \int_{\mathcal{X}_{te}} \widehat{\beta}(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x}$$

In practice, we replace the expectations in the above equation by their empirical averages. Ten the objective equation can then be expressed as

$$
\begin{aligned}
\widehat{J}_{uLSIF}(\boldsymbol{\theta}) &:= \frac{1}{2n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}}\widehat{\beta}\left(\boldsymbol{x}_i^{\text{tr}}\right)^2 - \frac{1}{n_{\text{te}}}\sum_{j=1}^{n_{\text{te}}}\widehat{\beta}\left(\boldsymbol{x}_j^{\text{te}}\right) \\
&= \frac{1}{2}\sum_{\ell,\ell'=1}^{b}\theta_\ell\theta_{\ell'}\left(\frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}}\psi_\ell\left(\boldsymbol{x}_i^{\text{tr}}\right)\psi_{\ell'}\left(\boldsymbol{x}_i^{\text{tr}}\right)\right) - \sum_{\ell=1}^{b}\theta_\ell\left(\frac{1}{n_{\text{te}}}\sum_{j=1}^{n_{\text{te}}}\psi_\ell\left(\boldsymbol{x}_j^{\text{te}}\right)\right) \\
&= \frac{1}{2}\boldsymbol{\theta}^\top\widehat{\boldsymbol{H}}\boldsymbol{\theta} - \widehat{\boldsymbol{h}}^\top\boldsymbol{\theta}
\end{aligned}
$$

where $\widehat{\boldsymbol{H}}$ is the $b \times b$ matrix with the (ℓ, ℓ')-th element equivalent to

$$\widehat{H}_{\ell,\ell'} := \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}}\psi_\ell\left(\widehat{\boldsymbol{u}}_i^{\text{tr}}\right)\psi_{\ell'}\left(\widehat{\boldsymbol{u}}_i^{\text{tr}}\right)$$

and $\widehat{\boldsymbol{h}}$ is the b-dimensional vector with the ℓ-th element being

$$\widehat{h}_\ell := \frac{1}{n_{\text{te}}}\sum_{j=1}^{n_{\text{te}}}\psi_\ell\left(\boldsymbol{x}_j^{\text{te}}\right)$$

Now we can formulate the optimisation problem as follows:

$$\widetilde{\boldsymbol{\theta}} := \underset{\boldsymbol{\theta}\in\mathbb{R}^b}{\operatorname{argmin}}\left[\frac{1}{2}\boldsymbol{\theta}^\top\widehat{\boldsymbol{H}}\boldsymbol{\theta} - \widehat{\boldsymbol{h}}^\top\boldsymbol{\theta} + \frac{\lambda}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}\right] \tag{6}$$

subject to $\boldsymbol{\theta}$ being non-negative. Here we add a regularisation term $\frac{\lambda}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}$ to prevent the algorithm assigning very large weights to a small number of samples, with λ controlling the strength of the regularisation. By computing the first derivative of this equation and setting it to 0, we obtain the optimal solution as

$$\widetilde{\boldsymbol{\theta}} = \left(\widehat{\boldsymbol{H}} + \lambda\boldsymbol{I}_b\right)^{-1}\widehat{\boldsymbol{h}}$$

To satisfy the non-negativity constraint on $\boldsymbol{\theta}$, we express the solution of $\widehat{\boldsymbol{\theta}}$ as

$$\widehat{\boldsymbol{\theta}} = \max\left(0, \widetilde{\boldsymbol{\theta}}\right)$$

2.3.1 Leave-one-out Cross-validation (LOOCV)

As with KMM and KLIEP, the performance of uLSIF is affected by the choice of the basis functions and in practice, it is reasonable to use a Gaussian kernel as the basis function. To select the Gaussian width of the kernel as well as the regularisation parameter λ in equation 6, we employ leave-one-out cross-validation (LOOCV).

Denote $n := \min\left(n_{\text{tr}}, n_{\text{te}}\right)$. We hold out $\widehat{\boldsymbol{u}}_k^{\text{tr}}$ and $\widehat{\boldsymbol{u}}_k^{\text{te}}(k = 1, 2, \ldots, n)$ simultaneously in each iteration of the LOOCV procedure. Suppose that $\widehat{\beta}^{(-k)}(\boldsymbol{x})$ is the estimate of the density ratio computed from samples without $\widehat{\boldsymbol{u}}_k^{\text{tr}}$ and $\widehat{\boldsymbol{u}}_k^{\text{te}}$, then we can write the LOOCV score as

$$\widehat{J}_{\text{LOOCV}} = \frac{1}{n}\sum_{k=1}^{n}\left[\frac{1}{2}\left(\widehat{\beta}^{(-k)}\left(\boldsymbol{x}_k^{\text{tr}}\right)\right)^2 - \widehat{\beta}^{(-k)}\left(\boldsymbol{x}_k^{\text{te}}\right)\right]$$

The LOOCV procedure is demonstrated using the pseudo code shown in Algorithm 3.

Algorithm 3: uLSIF with LOOCV (in below $*$ denotes the element-wise multiplication.)

```
Input: Data samples $D^{\mathrm{te}} = \{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}}$ and $D^{\mathrm{tr}} = \left\{\boldsymbol{x}_j^{\mathrm{tr}}\right\}_{j=1}^{n_{\mathrm{tr}}}$ and basis function $\psi(\boldsymbol{x})$.
Output: The density ratio estimator $\hat{\beta}(\boldsymbol{x})$.
1  $n \longleftarrow \min(n_{\mathrm{tr}}, n_{\mathrm{te}}), b \longleftarrow \min(100, n_{\mathrm{nu}})$ ;    // b is the number of Gaussian
     centres, 100 is just an illustration and it can be replaced
     according to the size of test samples
2  Randomly choose $b$ centers $\{c_\ell\}_{\ell=1}^{b}$ from $\{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}}$ without replacement;
3  for each choice of Gaussian kernel width $\sigma$ do
4      $\hat{H}_{\ell,\ell'} \longleftarrow \frac{1}{n_{\mathrm{tr}}} \sum_{j=1}^{n_{\mathrm{tr}}} \exp\left(-\frac{\left\|\boldsymbol{x}_j^{\mathrm{tr}} - \boldsymbol{c}_\ell\right\|^2 + \left\|\boldsymbol{x}_j^{\mathrm{tr}} - \boldsymbol{c}_{\ell'}\right\|^2}{2\sigma^2}\right)$ for $\ell, \ell' = 1, \dots, b$ ;
5      $\widehat{h}_\ell \longleftarrow \frac{1}{n_{\mathrm{te}}} \sum_{i=1}^{n_{\mathrm{te}}} \exp\left(-\frac{\left\|x_i^{\mathrm{te}} - c_\ell\right\|^2}{2\sigma^2}\right)$ for $\ell = 1, \dots, b$ ;
6      $X_{\ell,i}^{\mathrm{te}} \longleftarrow \exp\left(-\frac{\left\|\boldsymbol{x}_i^{\mathrm{te}} - \boldsymbol{c}_\ell\right\|^2}{2\sigma^2}\right)$ for $i = 1, \dots, n$ and $\ell = 1, \dots, b$;
7      $X_{\ell,i}^{\mathrm{tr}} \longleftarrow \exp\left(-\frac{\left\|x_i^{\mathrm{tr}} - c_\ell\right\|^2}{2\sigma^2}\right)$ for $i = 1, \dots, n$ and $\ell = 1, \dots, b$;
8      for each value of the regularisation parameter $\lambda$ do
9          $\widehat{\boldsymbol{\theta}} \longleftarrow \widehat{H} + \frac{\lambda(n_{\mathrm{tr}}-1)}{n_{\mathrm{tr}}} I_b$ ;
10         $\boldsymbol{\theta}_0 \longleftarrow \widehat{B}^{-1}\widehat{h}1_n^\top + \widehat{B}^{-1}X^{\mathrm{tr}} \operatorname{diag}\left(\frac{\widehat{h}^\top \widehat{B}^{-1} X^{\mathrm{tr}}}{n_{\mathrm{tr}} 1_n^\top - 1_b^\top \left(X^{\mathrm{tr}} * \widehat{B}^{-1} X^{\mathrm{tr}}\right)}\right)$;
11         $\boldsymbol{B}_1 \longleftarrow \widehat{B}^{-1}X^{\mathrm{te}} + \widehat{B}^{-1}X^{\mathrm{tr}} \operatorname{diag}\left(\frac{1_b^\top \left(X^{\mathrm{te}} * \widehat{\theta}^{-1} X^{\mathrm{tr}}\right)}{n_{\mathrm{tr}} 1_n^\top - 1_b^\top \left(X^{\mathrm{tr}} * \widehat{B}^{-1} X^{\mathrm{tr}}\right)}\right)$;
12         $\boldsymbol{B}_2 \longleftarrow \max\left(\boldsymbol{O}_{b\times n}, \frac{n_{\mathrm{tr}}-1}{n_{\mathrm{tr}}\left(n_{\mathrm{n}_{te}}-1\right)} \left(n_{\mathrm{te}}\boldsymbol{B}_0 - \boldsymbol{B}_1\right)\right)$ ;
13         $\boldsymbol{\beta}_{\mathrm{tr}} \longleftarrow \left(\mathbf{1}_b^\top \left(\boldsymbol{X}^{\mathrm{tr}} * \boldsymbol{B}_2\right)\right)^\top$;   $\boldsymbol{\beta}_{\mathrm{te}} \longleftarrow \left(\mathbf{1}_b^\top \left(\boldsymbol{X}^{\mathrm{te}} * \boldsymbol{B}_2\right)\right)^\top$ ;
14         $\mathrm{LOOCV}(\sigma, \lambda) \longleftarrow \frac{\boldsymbol{\beta}_{\mathrm{tr}}^\top \boldsymbol{\beta}_{\mathrm{tr}}}{2n} - \frac{\mathbf{1}_n^\top \boldsymbol{\beta}_{\mathrm{te}}}{n}$;
15     end
16 end
17 $(\hat{\sigma}, \hat{\lambda}) \longleftarrow \operatorname{argmin}_{(\sigma,\lambda)} \mathrm{LOOCV}(\sigma, \lambda)$
18 $\tilde{H}_{\ell,\ell'} \longleftarrow \frac{1}{n_{\mathrm{tr}}} \sum_{j=1}^{n_{\mathrm{tr}}} \exp\left(-\frac{\left\|\boldsymbol{x}_j^{\mathrm{tr}} - \boldsymbol{c}_\ell\right\|^2 + \left\|\boldsymbol{x}_j^{\mathrm{tr}} - \boldsymbol{c}_{\ell'}\right\|^2}{2\widehat{\sigma}^2}\right)$ for $\ell, \ell' = 1, \dots, b$
19 $\tilde{h}_\ell \longleftarrow \frac{1}{n_{\mathrm{te}}} \sum_{i=1}^{n_{\mathrm{te}}} \exp\left(-\frac{\left\|\boldsymbol{x}_i^{\mathrm{te}} - \boldsymbol{c}_\ell\right\|^2}{2\widehat{\sigma}^2}\right)$ for $\ell = 1, \dots, b$
20 $\hat{\theta} \longleftarrow \max\left(\mathbf{0}_b, \left(\widetilde{\boldsymbol{H}} + \widehat{\lambda}\boldsymbol{I}_b\right)^{-1} \widetilde{\boldsymbol{h}}\right)$
21 $\widehat{\beta}(x) \longleftarrow \sum_{\ell=1}^{b} \widehat{\theta}_\ell \exp\left(-\frac{\|x - c_\ell\|^2}{2\widehat{\sigma}^2}\right)$
```

2.4 Challenges

To summarise, all three density-ratio estimation methods - KMM, KLIEP and uLSIF - avoid explicit density estimation and they all form convex optimisation problems and hence have unique global solutions. KMM is based on infinite order moment matching and is guaranteed to be consistent, although the lack of suitable tuning methods makes model selection difficult. KLIEP and uLSIF both are ratio matching approaches and use cross-validation methods to tune parameters. [Kanamori et al. (2009)] states that the ratio matching approach have advantage over the moment matching approach in term of numerically stability. Also KLIEP tends to give sparse solutions, which may lead to better computational efficiency [Tsuboi et al. (2009)].

Nonetheless, the three methods tend to perform less effectively when the dimensionality of the features grows. For instance, for both the moment matching approach [Stojanov et al. (2019), Yamada et al. (2013)] and the ratio matching approach [Yamada et al. (2013)], the effects of estimation error of $\hat{\beta}$ in the test domain prediction risks have been mathematically analysed and some simulation results have demonstrated that the prediction accuracy deteriorates as the feature dimensionality increases. Here we illustrate this issue using an artificial dataset:

Let

$$X_{tr} \sim 0.75N\left([0,\cdots,0]^d,\mathbf{\Sigma}\right) + 0.25N\left([0.5,\cdots,0.5]^d,\mathbf{\Sigma}\right)$$
$$X_{te} \sim 0.5N\left([1.5,\cdots,1.5]^d,\mathbf{\Sigma}\right) + 0.5N\left([2,\cdots,2]^d,\mathbf{\Sigma}\right)$$

where d is the dimensionality of variables. Also, we assume the covariance matrix $\mathbf{\Sigma}$ is an identity matrix (i.e. variables are independent) for simplicity. We further define the the distribution of the response variable as

$$P(Y=1|X) = sigmoid(3X_1 + sin(X_2))$$

then we label Y as positive if $P(Y=1|X) \geq 0.5$ and negative if otherwise. Notice that the response Y only depends on X_1 and X_2, which is similar to our dataset where a phenotype is only determined by a small number of genotypes. Here we vary the value of d and apply all three methods to assess their performance when the dimensionality of features change.

We draw 250 samples each from the training and test distributions, then over a range of dimensions (from 2 to 20), we apply the three density ratio estimation methods and compute the squared estimation error $||\beta - \hat{\beta}||^2$. This trial is repeated 50 times and we evaluate the mean of the squared estimation errors. Although the squared estimation errors show some fluctuation when the dimensionality increases, the overall trend is clear (Figure 1). For all three methods, the performance deteriorates in high dimensional regime.

Additionally, for each dimension, we fit a weighted logistic regression on the training samples, then use it to predict the labels of the test samples. We use the area under the ROC curve (AUC) as the evaluation metric.

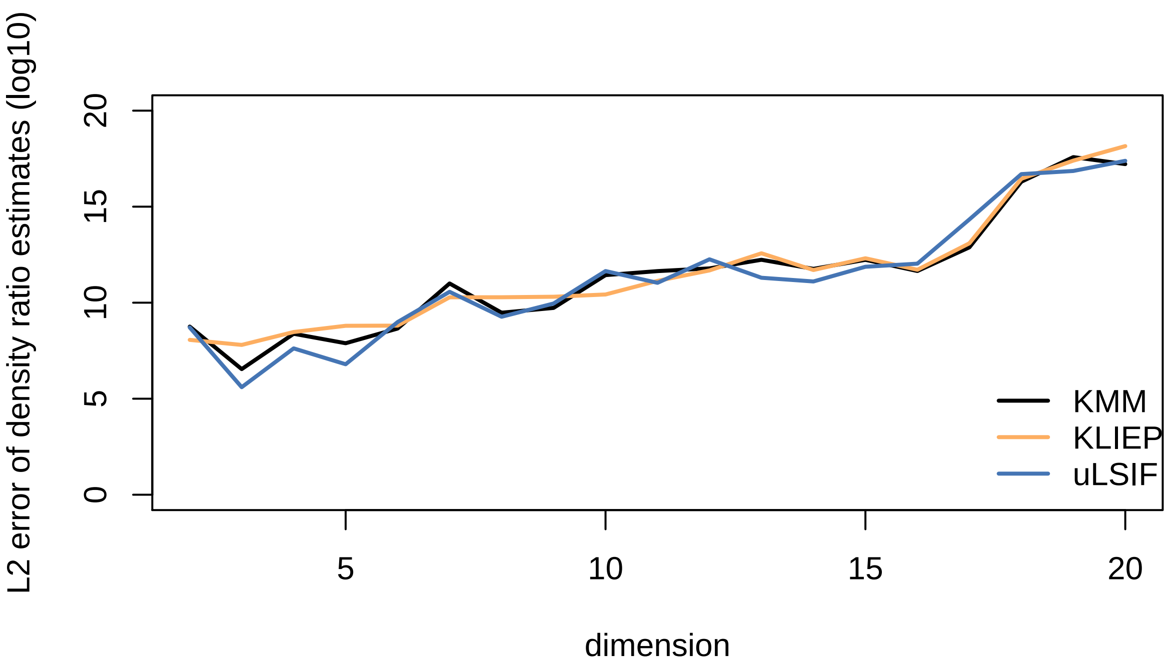


Figure 1: $||\beta - \hat{\beta}||^2$ for KMM, KLIEP and uLSIF against dimensionality of variables. The squared errors of density ratio are shown on log10 scale.

dimensionality	KMM	KLIEP	uLSIF
2	0.514 (0.081)	0.513 (0.093)	0.511 (0.090)
5	0.501 (0.103)	0.502 (0.105)	0.510 (0.112)
8	0.471 (0.100)	0.477 (0.106)	0.478 (0.102)
11	0.499 (0.112)	0.507 (0.105)	0.504 (0.109)
14	0.479 (0.136)	0.496 (0.110)	0.500 (0.113)
17	0.492 (0.107)	0.472 (0.121)	0.477 (0.119)
20	0.485 (0.108)	0.494 (0.108)	0.489 (0.101)

Table 1: Average AUC for different density ratio estimation methods over a range of feature dimensionality, values in bracket indicates the standard deviation.

Table 1 shows that the AUC decreases and its standard deviation increases as the dimensionality of the variables grows. This illustrates that when using these density ratio estimation algorithms on datasets with high dimensions, the predictive performance worsens and the prediction risk in the test domain increases. This motivates the next chapter: by first reducing the feature dimensionality, the performance of these algorithms may be enhanced.

3 Supervised Dimension Reduction

In genotype data modelling, dimension reduction is a standard processing step for a number of reasons. As mentioned above, a lower dimensional feature representation may lead to improved accuracy on density ratio estimates and lower test domain prediction risk. Genotype data are characterised by an enormous number of variants with a complex correlation structure. Such a large number of variables can cause issues such as singularity and can result some algorithms being unable to perform. [Wang and van der Laan (2011)]. Performance of predictive models depends on the interrelationship between model complexity, sample size and feature dimension, sophisticated models with high dimensional feature space but moderate sample size can severely overfit. In statistical learning, a rule of thumb is to have at least 10 samples for each feature dimension [Jain et al. (2000)]. However, in gene expression data, this ratio is very small. For example, for the UK Biobank dataset studied in this report (see chapter 5.1), the ratio is around 0.51. This weakens the validity of predictive results and can easily generate false positives [Wang et al. (2008)]. Further, the amount of computation required increases with the dimensionality of features. Hence, finding a lower dimensional feature representation can reduce both time complexity and space complexity when implement algorithms. It is worth noting that in genotype data, unsupervised dimension reduction methods are often ineffective because they do not account for the relevance of the features to the target labels. For example, in unsupervised PCA, a smaller dimensional features are constructed as a linear combination of genotypes in the original dataset, this approach assumes the direction with majority of variation contains the majority of information. However, this is always not true in genotype datasets where a phenotype is only affected by a small subset of genotypes, using such unsupervised methods may cause informative genes with weaker signals to be left out. Thus, investigating useful supervised dimension reduction (SDR) techniques will be our main focus.

There are multiple ways of to perform SDR in the context of genotype data. One approach is to select a small subset of genotypes based on a specific univariate measurement. This method is easy to understand, perform and interpret. It is often easily scalable to the dimensionality of the data. However, relies on the unrealistic assumption that the genotypes are independent to each other. Therefore, the results can carry a lot of noise and the selected genotypes are often highly correlated, which can create problems in subsequent analysis [Wang and van der Laan (2011)]. Alternatively, we can construct features as a linear or non-linear combination of genotypes, which better takes the correlation structure into consideration. In this chapter, we will introduce a few of such SDR methods, followed by a discussion about their strengths and limitations.

3.1 Supervised PCA

Unlike traditional PCA methods that create new uncorrelated variables by successively maximising the variance, the supervised PCA (SPCA) [Barshan et al. (2011)], estimates a sequence of principal components that have maximal dependence on the labels, thus giving more effective information fr the subsequent prediction tasks. The detailed mathematical framework for this algorithms is described as follows:

3.1.1 Hilbert-Schmidt Independence Criterion (HSIC)

To quantify the dependence between two random variables, say, $\boldsymbol{X}$ and $\boldsymbol{Y}$, we use the Hilbert-Schmidt Independence Criterion (HSIC) [Gretton et al. (2005)]. The HSIC is based on the fact that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent if and only if any continuous bounded function of these two variables are uncorrelated.

Specifically, given observations $\{(x_1, y_1), \ldots, (x_n, y_n)\} \sim P_{XY}$, we test their dependence against the null hypothesis $P_{XY} = P_X \times P_Y$. Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ defined on a non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$ if it satisfies

(1) $\forall x \in \mathcal{X}, \quad k_x = k(\cdot, x) \in \mathcal{H}$
(2) $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f, k(\cdot, x)\rangle_{\mathcal{H}} = f(x)$ (the reproducing property).

If $\mathcal{H}$ has a reproducing kernel, it is called a reproducing kernel Hilbert space (RKHS). Define $\mathcal{F}$ as a separable RKHS that contains all continuous bounded real value functions of $x \in \mathcal{X}$ to $\mathbb{R}$. Let the linear product in the kernel space be given by $k(\cdot, \cdot)$. Similarly define $\mathcal{G}$ as the RKHS of $y \in \mathcal{Y}$ to $\mathbb{R}$ with kernel function $l(\cdot, \cdot)$. Denote the covariance operator that maps elements of $\mathcal{G}$ to the elements of $\mathcal{F}$ as $C_{x,y} : \mathcal{G} \to \mathcal{F}$. Then the cross-covariance between elements of $\mathcal{F}$ and $\mathcal{G}$ is

$$\langle f, C_{xy} g\rangle = \mathrm{Cov}(f(x), g(y)) = \mathbb{E}_{x,y}[f(x), g(y)] - \mathbb{E}_x[f(x)]\mathbb{E}_y[g(y)] \quad \forall f \in \mathcal{F}, \forall g \in \mathcal{G}.$$

From the reproducing property of RKHS, we know that $f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{F}}$ and $g(y) = \langle f, k(\cdot, x)\rangle_{\mathcal{G}}$. Then this covariate operator can be expressed as

$$C_{xy} = \mathbb{E}_{x,y}[k(x, \cdot)l(y, \cdot)] - \mathrm{E}_x[k(x, \cdot)]\mathbb{E}_y[l(y, \cdot)] \quad .$$

The dependence between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is measured by the HSIC which is defined as the square of the Hilbert-Schmidt (HS) norm of the cross-covariance operator C_{xy}. The HS norm is defined as follows:

$$\|C_{xy}\|_{HS}^2 := \sum_{i,j} \langle C v_i, u_j\rangle_{\mathcal{F}}^2$$

where u_j and v_i are orthogonal bases of $\mathcal{F}$ and $\mathcal{G}$, respectively. The HSIC can be further expressed in terms of kernel functions:

$$\begin{aligned} HSIC\left(P_{XY}, \mathcal{F}, \mathcal{G}\right) := \|C_{xy}\|_{HS}^2 \\ &= \mathrm{E}_{x,x',y,y'}\left[k\left(x, x'\right) l\left(y, y'\right)\right] + \mathrm{E}_{x,x'}\left[k\left(x, x'\right)\right] \mathrm{E}_{y,y'}\left[l\left(y, y'\right)\right] \\ &\quad - 2\mathrm{E}_{x,y}\left[\mathrm{E}_{x'}\left[k\left(x, x'\right)\right] \mathrm{E}_{y'}\left[l\left(y, y'\right)\right]\right] \end{aligned}$$

where $\mathbf{E}_{x,x',y,y'}$ denotes the expectation over independent pairs of (x, y) and (x', y') drawn from $P_{\boldsymbol{X},\boldsymbol{Y}}$. As long as k, l are universal kernels, $\|C_{xy}\|_{HS}^2 = 0$ if and only if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent.

In practice, the empirical HSIC used as an approximation to the test statistic:

$$\widehat{HSIC} = \frac{1}{(n-1)^2} \operatorname{tr}(KHLH) \tag{7}$$

where $H, K, L \in \mathbb{R}^{n\times n}$, $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$, $L_{ij} := l(\mathbf{y}_i, \mathbf{y}_j)$, and H is the centering matrix defined as $H_{ij} := I - n^{-1}\mathbf{e}\mathbf{e}^T$. From equation 7 we conclude that to maximise the dependence between two kernels, we need to maximise the trace of $(KHLH)$

3.1.2 Supervised PCA Algorithm

Suppose our dataset consists of n samples and each sample has p features, then the feature matrix X has dimension $p \times n$. Further denote $Y_{\ell\times n}$ a matrix of outcome measures, where ℓ indicates the number of classes of labels. The goal is to map X to a lower dimensional subspace $U^T X$ such that the dependence between the projected data points $U^T X$ and Y is maximised. Substitute K with the kernel of $U^T X$ and L with the kernel of Y into equation 7 (here we assume both kernels are linear for simplicity, but in practice it can be chosen depending on the characteristics of the dataset), we obtain:

$$\begin{aligned} \mathrm{tr}(HKHL) &= \mathrm{tr}\left(HX^TUU^TXHL\right) \\ &= \mathrm{tr}\left(U^TXHLHX^TU\right) \end{aligned}$$

We wish to find an orthogonal transformation matrix U such that the features in the subspace uncorrelated. To achieve this, we add the following constraint:

$$\begin{array}{ll} \arg\max_U & \mathrm{tr}\left(U^TXHLHX^TU\right) \\ \text{subject to} & U^TU = I \end{array} \tag{8}$$

This formulation is similar to the optimisation problem for traditional PCA and it is clear that the the solution can be obtained via eigendecomposition of the real and symmetric matrix $Q = XHLHX^T$. Suppose Q has the following p eigenvalues $\lambda_1 \geq \ldots \geq \lambda_p$, corresponding to eigenvectors $v_1, \ldots, v_p$. To obtain a subspace in $\mathbb{R}^d$ where $d \ll p$, the solution is $U = [v_1, v_2, \ldots, v_d]$. Note that when the kernel matrix L is equivalent to an identity matrix I, the supervised PCA reduces to the unsupervised case.

3.1.3 The Dual Problem

The above solution of SPCA can also be derived by solving its dual optimisation problem, meaning that the eigendecomposition is to perform on an $n \times n$ matrix, rather than a $p \times p$ matrix. The dual problem is more computationally efficient for our dataset since the number of genotypes far exceed the sample size.

The matrix Q and the kernel matrix L in the primal problem are positive semi-definite and hence can be decomposed as follows:

$$\begin{aligned} &Q = XHLHX^T = \Psi\Psi^T \qquad \Rightarrow \Psi = XH\Delta^T \\ &L = \Delta^T\Delta \end{aligned}$$

We can then apply the singular value decomposition (SVD) on Ψ and rewrite Q using the results of the SVD

$$\begin{aligned} \Psi &= U\Sigma V^T \\ Q &= U\Sigma\Sigma^T U^T \end{aligned}$$

It is evident that U in the SVD consist of the eigenvectors of Q and therefore, $U = \Psi V \Sigma^{-1}$ is the solution to the optimisation problem. To map X to a d dimensional subspace, we select first d columns of matrix U, denoted as $\hat{U}_{p\times d}$, then the projected matrix is $\hat{U}^T X$. The pseudo-code for this algorithm is given as follows:

Algorithm 4: Dual Supervised PCA

Input: training feature matrix X, test data example x, kernel matrix of labels L, training data size n.

Output: Dimension reduced training data **Z** and test data **z**.

```
1 Decompose $L$ such that $L = \Delta^T \Delta$.
2 $H \leftarrow I - n^{-1}\mathbf{e}\mathbf{e}^T$
3 $\Psi \leftarrow XH\Delta^T$
4 Compute basis:
    $V \leftarrow$ eigenvectors of $\Psi^T\Psi = \Delta H \left[X^T X\right] H\Delta^T$ corresponding to the top $d$ eigenvalues.
        $\Sigma \leftarrow$ diagonal matrix of square roots of the top $d$ eigenvalues of $\Psi^T\Psi$.
        $U \leftarrow \Psi V \Sigma^{-1}$
5 Encode training data: $Z \leftarrow U^T X = \Sigma^{-1} V^T \Delta H \left[X^T X\right]$
6 Encode test example: $\mathbf{z} \leftarrow U^T\mathbf{x} = \Sigma^{-1} V^T \Delta H \left[X^T \mathbf{x}\right]$
```

Note that solving this optimisation problem does not involve expensive iterative optimisation procedures, making it suitable for high dimensional datasets.

3.1.4 Kernel Supervised PCA

Supervised PCA only allows linear combination of features, which may not always be effective. Here we extend the supervised PCA to allow nonlinear combinations by mapping the features onto a kernel space [Ghojogh and Crowley (2019)]. Define a feature map Φ which maps x to a feature space $\mathcal{H}$ such that $\Phi : x \to \mathcal{H}$ and $x \mapsto \Phi(x)$. Also the kernel matrix of our dataset is $K_x := K(X, X) = \Phi(X)$. Then the Kernel Supervised PCA (KSPCA) can be formulated as:

$$\arg\max_U \operatorname{tr}\left(U^T \Phi(X) HLH \Phi(X)^T U\right) \text{ subject to } \quad U^T U = I$$

As above, we let $Q = \Phi(X)HLH\Phi(X)^T$ and apply matrix decomposition on Q and L such that $Q = \Psi\Psi^T$ and $L = \Delta^T\Delta$. Then we perform SVD on Ψ and obtain $\Psi = \Phi(X)H\Delta^T = U\Sigma V^T$, where U consists of the eigenvectors of $\Psi(X)\Psi(X)^T$. In practice, it is often difficult and unnecessary to compute the value of $\Phi(X)$ explicitly, especially it can be up to an infinite order. Instead, we use the the kernel matrix $K_x = K(X, X) = \Phi(X)^\top \Phi(X)$. Also, note that K_x is an $n \times n$ matrix, so kSPCA is computationally feasible for datasets which have a relatively small number of samples but large number of features. By the representation theorem [Fulton and Harris (2013)], any solution $\boldsymbol{u} \in \mathcal{H}$ must lie in the span of the of all the training vectors mapped to the Hilbert space $\mathcal{H}$. Thus we can rewrite the transformation matrix U as a linear projection of the projected data points, i.e. $U = \Phi(X)\beta$, where $\beta := [\beta_1, \ldots, \beta_p] \in \mathbb{R}^{n\times p}$. Then the optimisation problem can be expressed as:

$$
\begin{aligned}
&\arg\max_U \operatorname{tr}\left(U^T\Phi(X)HLH\Phi(X)^TU\right)\\
&= \arg\max_U \operatorname{tr}\left(\beta^T\Phi(X)^T\Phi(X)HLH\Phi(X)^T\Phi(X)\beta\right)\\
&= \arg\max_U \operatorname{tr}\left(\beta^T K_x HLHK_x\beta\right)\\
&\text{subject to} \quad U^TU = \beta^T\Phi(X)^\top\Phi(X)\beta = \beta^T K_x\beta
\end{aligned}
$$

This is a generalised eigenvalue problem of (K_xHLHK_x, K_x) and has a closed-form solution [Ghojogh et al. (2019)]. Further, [Ghojogh and Crowley (2019)] shows that this optimisation problem can be alternatively solved by performing the eigendecomposition on $HLHK_x$.

The algorithm is illustrated using the following pseudo-code:

Algorithm 5: Kernel Supervised PCA

```
Input: Kernel matrix of training data $K_x$, kernel matrix of testing data $K_{\text{test}}$, kernel
        matrix of target variable $L$, testing data example $x$, training data size n
Output: Dimension reduced training data $\mathbf{Z}$ and testing data $\mathbf{z}$.
1 $H \leftarrow I - n^{-1}\,\mathrm{ee}^T$.
2 $Q \leftarrow K_xHLHK_x$
3 Compute basis: $\beta \leftarrow$ generalized eigenvectors of $(\mathrm{Q},\mathrm{K})$ corresponding to the top $d$
   eigenvalues.
4 Encode training data: $\mathbf{Z} \leftarrow \beta^T\left[\Phi(X)^T\Phi(X)\right] = \beta^TK_x$
5 Encode test example: $\mathbf{z} \leftarrow \beta^T\left[\Phi(X)^T\Phi(\mathbf{x})\right] = \beta^TK_{\text{test}}$
```

3.2 Local Fisher Discriminant Analysis

Supervised PCA produces a projection with maximal dependence on the response variable but does not consider local structure of the data. In a genotype dataset, there is often the presence of multimodality. For example, the genotypes exhibits a clustering structure since the contiguous genotypes are likely to be correlated. Since numerous clusters can cause participants to have a particular disease, these structure may reveal valuable information. As an alternative to supervised PCA, here we introduce Local Fisher Discriminant Analysis (LFDA) which integrates the idea of both Fisher Linear Discriminant Analysis (FDA) and Local Preserving Projection (LPP) such that the local structure of the data points in the original feature space is preserved in the embedding space.

3.2.1 Fisher Linear Discriminant Analysis

FDA is one of commonly applied supervised dimension reduction methods [Fisher (1936)]. It finds a linear combination of features that maximises the separation between classes while minimising the variance of variance within each class. Suppose we have p dimensional samples $x_i \in \mathbb{R}^p (i = 1, 2, \ldots, n)$ and the associated class labels $y_i \in \{1, 2, \ldots, c\}$ where c is the total number of classes. Also suppose n_j to be the number of samples in class ℓ such that $\sum_{\ell=1}^{c} n_\ell = n$.

Denote $S^{(w)}$ and $S^{(b)}$ be the within-class scatter matrix and between-class scatter matrix respectively, which are defined as:

$$S^{(w)} = \sum_{\ell=1}^{c} \sum_{i:y_i=\ell} (x_i - \mu_\ell)(x_i - \mu_\ell)^\top \tag{9}$$

$$S^{(b)} = \sum_{\ell=1}^{c} n_\ell (\mu_\ell - \mu)(\mu_\ell - \mu)^\top \tag{10}$$

where μ_ℓ is the mean of samples in class ℓ, computed as $\mu_\ell = \frac{1}{n_\ell} \sum_{i:y_i=\ell} x_i$ and μ is the mean of all samples $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n} \sum_{\ell=1}^{c} n_\ell \mu_\ell$.

Assume that $S^{(w)}$ has full rank, then the FDA transformation matrix T_{FDA} can be derived from solving the following optimisation problem:

$$T_{FDA} = \underset{T \in \mathbb{R}^{p \times d}}{\operatorname{argmax}} \left[\frac{\det\left(T^\top S^{(b)} T\right)}{\det\left(T^\top S^{(w)} T\right)} \right]$$

where d is the dimension of the projected feature space and $1 \leq d \leq p$. For this problem to be solvable, there is an implicit constraint that $\operatorname{rank}(T) = d$, otherwise the quantity in the denominator $T^\top S^{(w)} T$ is not invertible. The solution of the transformation matrix can be derived via solving the following generalised eigenvalue problem:

$$S^{(b)} \varphi = \lambda S^{(w)} \varphi$$

where $\{\varphi_k\}_{k=1}^{p}$ are the generalised eigenvectors corresponding to the generalised eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, then the transformation matrix T_{FDA} is given as:

$$T_{FDA} = (\varphi_1 | \varphi_2 | \cdots | \varphi_d)$$

3.2.2 Local Preserving Projection (LPP)

The Local Preserving Projection (LPP) is an unsupervised dimension reduction technique that preserves the local structure of the features after mapping [He and Niyogi (2004)]. A brief introduction is given here.

Let A be the $n \times n$ affinity matrix where $A_{i,j} \in [0, 1]$ indicates the affinity between x_i and x_j. $A_{i,j}$ is close to 1 if the distance between x_i and x_j is small, by contrast $A_{i,j}$ is close to 0 if the two samples are far apart. Some popular choice of the affinity matrix A are (1) the heat kernel, which is expressed as $A_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$ and $\sigma (> 0)$ is a tunable decay parameter [Belkin and Niyogi (2003)]; (2) the nearest neighbour affinity[Roweis and Saul (2000)] where we define $A_{i,j} = 1$ if x_j is the k-nearest neighbour of x_i or vice verse and $A_{i,j} = 0$ if otherwise. In this paper, we illustrate the algorithm using local scaling, which takes into account that the density of samples may differ depending on their regions [Zelnik-Manor and Perona (2005)]. The local scaling affinity matrix is defined as

$$A_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_i \sigma_j}\right)$$

σ_i indicates the local scaling of samples around x_i computed as $\sigma_i = \left\| x_i - x_i^{(K)} \right\|$, where $x_i^{(K)}$ is the K-th nearest neighbour of x_i.

Then the LPP transformation matrix is defined as the following:

$$T_{LPP} = \underset{T \in \mathbb{R}^{p \times d}}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i,j=1}^{n} A_{i,j} \left\| T^\top x_i - T^\top x_j \right\|^2 \right) \text{ subject to } T^\top X D X^\top T = I_d$$

where D is an $n \times n$ diagonal matrix with the i-th element on diagonal being $D_{i,i} = \sum_{j=1}^{n} A_{i,j}$. D represents the local structure of the samples in the original space. The role of the constraint is to prevent degeneracy. The transformation matrix T_{LPP} aims to keep data points which are close in the original space still close in the embedding space. This objective function can be further expressed as:

$$\begin{aligned} T_{LPP} &= \underset{T \in \mathbb{R}^{p \times d}}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i,j=1}^{n} A_{i,j} \left\| T^\top x_i - T^\top x_j \right\|^2 \right) \\ &= \underset{T \in \mathbb{R}^{p \times d}}{\operatorname{argmin}} \left(T^T X (D - A) X^T T \right) \\ &= \underset{T \in \mathbb{R}^{p \times d}}{\operatorname{argmin}} \left(T^T X L X^T T \right) \end{aligned}$$

where $L = D - A$ is named the Laplacian matrix. We further denote $\{\psi_k\}_{k=1}^{p}$ as the generalised eigenvectors corresponding to the generalised eigenvalues $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_p$ of the following generalised eigenvalue problem:

$$X L X^\top \Psi = \gamma X D X^\top \Psi$$

As shown in [He and Niyogi (2004)], the optimal solution is given by

$$T_{LPP} = (\Psi_p \, | \Psi_{p-1} | \cdots | \, \Psi_{p-d+1})$$

3.2.3 Local Fisher Discriminant Analysis (LFDA)

LLP is an unsupervised method which ignores the connection between the features and the labels while LDA is supervised but ignores the local structure of the features. Here, we combine the two methods to construct the Local Fisher Discriminant Analysis (LFDA) algorithm. LFDA is formulated as follows [Sugiyama (2007)].

First, we rewrite $S^{(w)}$ and $S^{(b)}$ in equation 9 and 10 as

$$\begin{aligned} S^{(w)} &= \tfrac{1}{2} \textstyle\sum_{i,j=1}^{n} W_{i,j}^{(w)} \left(x_i - x_j \right) \left(x_i - x_j \right)^\top \\ S^{(b)} &= \tfrac{1}{2} \textstyle\sum_{i,j=1}^{n} W_{i,j}^{(b)} \left(x_i - x_j \right) \left(x_i - x_j \right)^\top \end{aligned} \tag{11}$$

where

$$\begin{aligned} W_{i,j}^{(w)} &= \begin{cases} A_{i,j}/n_\ell & \text{if } y_i = y_j = \ell \\ 0 & \text{if } y_i \neq y_j \end{cases} \\ W_{i,j}^{(b)} &= \begin{cases} A_{i,j}(1/n - 1/n_\ell) & \text{if } y_i = y_j = \ell \\ 1/n & \text{if } y_i \neq y_j \end{cases} \end{aligned} \tag{12}$$

The derivation is given below:

$$
\begin{aligned}
S^{(w)} &= \sum_{\ell=1}^{c} \sum_{i:y_i=\ell} \left(x_i - \frac{1}{n_\ell} \sum_{j:y_j=\ell} x_j \right) \left(x_i - \frac{1}{n_\ell} \sum_{j:y_j=\ell} x_j \right)^\top \\
&= \sum_{i=1}^{n} x_i x_i^\top - \sum_{\ell=1}^{c} \frac{1}{n_\ell} \sum_{i,j:y_i=y_j=\ell} x_i x_j^\top \\
&= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} W_{i,j}^{(w)} \right) x_i x_i^\top - \sum_{i,j=1}^{n} W_{i,j}^{(w)} x_i x_j^\top \\
&= \frac{1}{2} \sum_{i,j=1}^{n} W_{i,j}^{(w)} \left(x_i x_i^\top + x_j x_j^\top - x_i x_j^\top - x_j x_i^\top \right)
\end{aligned}
$$

Additionally, define $S^{(m)}$ as the mixture scatter matrix [Fukunaga (2013)] where $S^{(m)} = S^{(w)} + S^{(b)} = \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^\top$, then we obtain

$$
\begin{aligned}
S^{(b)} &= \sum_{i=1}^{n} x_i x_i^\top - \frac{1}{n} \sum_{i,j=1}^{n} x_i x_j^\top - S^{(w)} \\
&= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{1}{n} \right) x_i x_i^\top - \sum_{i,j=1}^{n} \frac{1}{n} x_i x_j^\top - S^{(w)} \\
&= \frac{1}{2} \sum_{i}^{n} \left(\frac{1}{n} - W_{i,j}^{(w)} \right) \left(x_i x_i^\top + x_j x_j^\top - x_i x_j^\top - x_j x_i^\top \right)
\end{aligned}
$$

as required.

Notice that in equation 12, the affinity matrix A is defined such that $A_{i,j}$ is large only for data points x_i and x_j that are not far apart. Hence, when x_i and x_j are in the same class and they are close to each other, $W_{i,j}^{(w)}$ is positive while $W_{i,j}^{(b)}$ is negative, implying that the within-class scatter becomes small and the between-class scatter becomes large. By contrast, when x_i and x_j are in different classes, their within-class scatter shrinks and the between-class scatter increases.

We further define the LFDA transformation matrix T_{LFDA} as

$$
T_{LFDA} = \text{argmax}_{T \in \mathbb{R}^{p \times d}} \left[\text{tr} \left(\left(T^T S^{(w)} t \right)^{-1} T^T S^{(b)} T \right) \right]
$$

This can be solved by finding the generalised eigenvalues and generalised eigenvectors $\{\lambda_k, \varphi_k\}_{k=1}^{r}$ of the following generalised eigenvalues problem:

$$
S^{(b)} \varphi = \lambda S^{(w)} \varphi \tag{13}
$$

The optimal LDFA transformation matrix is given by

$$
T_{LFDA} = \left(\sqrt{\lambda_1} \varphi_1 \mid \sqrt{\lambda_2} \varphi_2 \mid \cdots \mid \sqrt{\lambda_d} \varphi_d \right)
$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$.

The algorithm is demonstrated in terms of pseudo-code, shown in the below:

Algorithm 6: Local Linear Fisher Discriminant Analysis

Input: feature matrix of training data, $X_{p\times n}$, training data labels Y, dimensionality of reduced space, d , number of neighbours to account for in KNN, k

Output: $d \times r$ transformation matrix T_{LFDA}.

```
1  $S^{(b)} \longleftarrow 0_{p\times p}$ ;              // $0_{p\times p}$ denotes the $p \times p$ matrix of zeroes
2  $S^{(w)} \longleftarrow 0_{p\times p}$ ;
3  for $\ell = 1, 2, \ldots, c$ ;              // compute the scatter matrix for each class
4  do
5      $\{\underline{x}_i\}_{i=1}^{n_\ell} \longleftarrow \{x_j\}_{j:y_j=\ell}$ ;
6      for $i = 1, 2, \ldots, n_\ell$ ;                        // determine local scaling
7      do
8          $\underline{x}_i^{(k)} \longleftarrow k$th nearest neighbor of $\underline{x}_i$ among $\left\{\underline{x}_j\right\}_{j=1}^{n_\ell}$ ;
9          $\underline{\sigma}_i \longleftarrow \left\|x_i - \underline{x}_i^{(k)}\right\|$
10     end
11     for $i, j = 1, 2, \ldots, n_\ell$ ;                    // define the affinity matrix
12     do
13         $\underline{A_{i,j}} \longleftarrow \exp\left(-\left\|\underline{x}_i - \underline{x}_j\right\|^2 / \left(\underline{\sigma}_i\underline{\sigma}_j\right)\right)$
14     end
15     $\underline{X} \longleftarrow \left(\underline{x}_1 \,|\underline{x}_2| \cdots |\, \underline{x}_{n_\ell}\right)$ ;
16     $\underline{G} \longleftarrow \underline{X}\,\mathrm{diag}\left(\underline{A}1_{n_\ell}\right)\underline{X}^\top - \underline{XAX}^\top$ ;
17     $S^{(b)} \longleftarrow S^{(b)} + \underline{G}/n + (1 - n_\ell/n)\,\underline{XX}^\top + \underline{X}1_{n_\ell}\left(\underline{X}1_{n_\ell}\right)^\top/n$ ;
18     $S^{(w)} \longleftarrow S^{(w)} + \underline{G}/n_\ell$
19 end
20 $S^{(b)} \longleftarrow S^{(b)} - X1_n\left(X1_n\right)^\top/n - S^{(w)}$ ;
21 $\{\lambda_k, \varphi_k\}_{k=1}^{r} \longleftarrow$ generalized eigenvalues and normalized eigenvectors of
       $\varphi = \lambda S^{(w)}\varphi$;   $\%\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ ;
22 $T_{LFDA} = \left(\sqrt{\lambda_1}\varphi_1 \left|\sqrt{\lambda_2}\varphi_2\right| \cdots |\, \sqrt{\lambda_r}\varphi_r\right)$
```

Notice that based on this formulation, two data points that are in the same class but far from each other do not come closer in the embedding space. One major advantage of LFDA over traditional FDA is that in the latter, the between-class scatter matrix $S^{(b)}$ has at most rank $c - 1$, implying that it can find at most $c - 1$ dimensions of meaningful features while the remaining dimensions resulted from FDA are merely arbitrary [Fukunaga (2013)]. This limitation makes FDA ineffective for the UK Biobank dataset we used on this report where the genotypes have only two classes and reducing a $\sim 780,000$ dimensional feature space to a one dimensional space is clearly not sensible. In contrast, the local between-class scatter matrix from LFDA in general has much higher rank because we incorporat the affinity matrix A into the weights. Thus, the LFDA can be practically implemented for dimensionality reduction into any dimensional space.

3.2.4 Kernel Local Linear Fisher Discriminant Analysis (KLFDA)

Similar as SPCA, LFDA can be kernelised to allow non-linear dimension reduction. Also, by using the kernel trick, the computational burden is greatly eased since the dimension of the kernel matrix is only decided by the sample size, not the dimensionality of features. To illustrate this algorithm, we will start with the LFDA optimisation problem we described previously, then show that it is possible to incorporate kernel methods into its formulation.

Recall that in LFDA (equation 13), the transformation matrix is solved by:

$$S^{(b)}\varphi = \lambda S^{(w)}\varphi$$

Also previously we have defined $S^{(m)}$ as the local mixture scatter matrix which is the sum of between-class and within-class scatters, i.e. $S^{(m)} = S^{(b)} + S^{(w)}$. From equation 11 and 12, $S^{(m)}$ can be expressed as [Sugiyama (2007)]:

$$\begin{aligned} S^{(m)} &= \frac{1}{2}\sum_{i,j=1}^{n} W_{i,j}^{(m)} \left(x_i - x_j\right)\left(x_i - x_j\right)^\top && (14)\\ &= \frac{1}{2}\sum_{i,j=1}^{n} W_{i,j}^{(m)} \left(x_i x_i^\top + x_j x_j^\top - x_i x_j^\top - x_j x_i^\top\right) && (15)\\ &= \sum_{i=1}^{n}\left(\sum_{j=1}^{n} W_{i,j}^{(m)}\right) x_i x_i^\top - \sum_{i,j=1}^{n} W_{i,j}^{(m)} x_i x_j^\top && (16)\end{aligned}$$

where $W^{(m)} = W^{(s)} + W(b)$ and is expressed as

$$W_{i,j}^{(m)} = \begin{cases} A_{i,j}/n & \text{if } y_i = y_j \\ 1/n & \text{if } y_i \neq y_j \end{cases}$$

We can further write equation 14 using its matrix form:

$$S^{(m)} = XL^{(m)}X^\top$$

where

$$L^{(m)} = D^{(m)} - W^{(m)}$$

and $D^{(m)}$ is an $n \times n$ diagonal matrix with its ith diagonal element being

$$D_{i,i}^{(m)} = \sum_{j=1}^{n} W_{i,j}^{(m)} \qquad .$$

Similarly, $S^{(w)}$ can be written as $S^{(w)} = XL^{(w)}X^\top$ where $L^{(w)} = D^{(w)} - W^{(w)}$ and $D_{i,i}^{(w)} = \sum_{j=1}^{n} W_{i,j}^{(w)}$.

Then the LFDA problem (equation 13) can be written as:

$$XL^{(b)}X^\top\varphi = \lambda XL^{(w)}X^\top\varphi \qquad (17)$$

where $L^{(b)} = L^{(m)} - L^{(w)}$.

We can further express $X^\top \varphi$ in terms of a vector $\alpha \in \mathbb{R}^n$ such that

$$X^\top \varphi = X^\top X \alpha = K\tilde{\alpha}$$

where K is an $n \times n$ kernel matrix with its element being $K_{i,j} \equiv x_i^\top x_j$. Then multiply both sides of equation 17 by X^T, we have

$$KL^{(b)}K\alpha = \lambda KL^{(w)}K\alpha \tag{18}$$

This shows that $\{x_i\}_{i=1}^n$ only appears in the form of its linear product, thus we can consider any non-linear mapping $\phi(\cdot)$ on x from the original feature space to a RKHS $\mathcal{H}$, then the matrix K in equation 18 represents the corresponding reproducing kernel matrix. Note that $KL^{(w)}K$ is always degenerate, leading the generalised eigenvalues problem to be unsolvable [Sugiyama (2007)], so it is necessary to regularise $KL^{(w)}K$. One possible means of regularisation adds a small constant ε and solves the following problem [Friedman (1989)]:

$$KL^{(b)}K\alpha = \lambda \left(KL^{(w)}K + \varepsilon I_n\right)\alpha$$

This gives the optimal transformation matrix of KLFDA as:

$$\left(\sqrt{\lambda_1}\alpha_1 \mid \sqrt{\lambda_2}\alpha_2 \ \cdots \mid \sqrt{\lambda_r}\alpha_r\right)^\top \begin{pmatrix} K(x_1, x') \\ K(x_2, x') \\ \vdots \\ K(x_n, x') \end{pmatrix}$$

3.3 Supervised Non-negative Matrix Factorisation

In our data, an element X_{ij} of the feature matrix is, in fact, the number of copies of the most frequent minor allele of the ith SNP for the jth person (note that the dimension of X is $p \times n$), which only takes values 0,1 or 2. In other words, each element in the feature matrix is either zero or positive, making X a non-negative matrix. In this case, Non-negative Matrix Factorisation (NMF) can be employed for dimension reduction and we can further make this algorithm supervised such that the features in the reduced dimension are more useful in discriminating different labels.

3.3.1 The NMF Model

The main idea of NMF is to approximate a non-negative matrix X by the product of two non-negative matrices T and W [Lee and Seung (1999)], i.e.

$$X_{p\times n} \approx T_{p\times k} W_{k\times n}$$

where T is referred to the type matrix and W is the weight matrix. The weight matrix consists of the coefficients by which the features of each sample can be constructed as a linear combination of columns (subcommunities) of the type matrix. The rank k of this approximation is chosen in a way that $(p + n)k \ll np$, and thus the dimension is reduced significantly. Also, k is a parameter that determines the complexity of the model and needs to be tuned carefully.

The implementation of NMF uses an iterative approach:

$$\begin{aligned}
&T_{ia} \leftarrow T_{ia} \textstyle\sum_j \frac{X_{ij}}{(TW)_{ij}} W_{aj} \\
&T_{ia} \leftarrow \frac{T_{ia}}{\sum_j T_{ja}} \\
&\quad\text{and} \\
&W_{aj} \leftarrow W_{aj} \textstyle\sum_i T_{ia} \frac{X_{ij}}{(TW)_{ij}}
\end{aligned}$$

These iterations converges to a local maximum of the following objective function [Lee and Seung (1999)]:

$$L(T, W) = \sum_{i=1}^{p} \sum_{j=1}^{n} [X_{ij} \log(TW)_{ij} - (TW)_{ij}] \tag{19}$$

subject to all elements in X, T and W are non-negative. We can interpret this objective function in terms of the likelihood where element of the feature matrix X_{ij} is modelled as an independent Poisson observation given its mean $(TW)_{ij}$, then we are trying to maximise the likelihood of generating the features in X from the type matrix T and the weights W. In our dataset, notice that the consecutive genotypes may have high correlation and this covariance structure is implicitly given in the type matrix T. We further add constraints to T such that each column sums to 1. Then each column can be interpreted as the composition of genotypes for each subcommunity. The general structure of NMF is illustrated using the pseudo code as the following:

Algorithm 7: Non-negative matrix factorisation

```
Input: feature matrix of training data $X_{p\times n}$, number of features in the reduced space
       $k \ll min[p, n]$
Output: a $p \times k$ type matrix $T$ and a $k \times n$ weight matrix $W$.
1 $T \longleftarrow 0_{p\times k}$ ;
2 $W \longleftarrow 0_{k\times n}$ ;                    // Initialise matrix $T$ and matrix $W$.
3 for $i = 1, 2, \ldots, maxIter$ do
4     $T_{ia} \leftarrow T_{ia} \sum_j \frac{X_{ij}}{(TW)_{ij}} W_{aj}$ ;
5     $T_{ia} \leftarrow \frac{T_{ia}}{\sum_j T_{ja}}$ ;
6     $W_{aj} \leftarrow W_{aj} \sum_i T_{ia} \frac{X_{ij}}{(TW)_{ij}}$ ;
7     check the stopping criterion ;   // the stopping criterion can be the KL
       divergence $D(X\|TW)$ is within some threshold
8 end
9 return $T, W$
```

3.3.2 Supervised NMF (SNMF)

The supervised version of NMF [Cai et al. (2017)] performs NMF on each class separately at first then combine the results into a single type matrix, such that each sample can be expressed as a mixture of all subcommunities. Suppose feature X are from c classes:

$$X = \left(X^{(1)}, X^{(2)}, \cdots, X^{(c)}\right)$$

where $X^{(i)}$ represents observations from the ith class. Then for each class, we obtain its corresponding type matrix $T^{(i)}$ via unsupervised NMF, then we combine these type matrices together to obtain a type matrix for all classes.

$$T = \left(T^{(1)}, T^{(2)}, \cdots, T^{(c)}\right)$$

T is still non-negative since each component $T^{(i)}$ is non-negative. Under the assumption that the rows of W are independent for different samples, for a fixed T, maximising the total Poisson log likelihood is equivalent to maximising the Poisson log likelihood of each sample. Hence, we can consider W as the non-negative coefficients for Poisson regression of X on T which maximise the Poisson log likelihood. For a sample $X_j = (X_{1j}, X_{2j}, \cdots, X_{pj})$, we fit a regression on T and make the resulting coefficients $W_j = (w_{1j}, \cdots, w_{kj})$ either zero (meaning the the corresponding variable in T is removed from the regression) or positive. The elements of W are computed via the following forward-backward procedure:

1. Fit a Poisson regression on T, assuming no intercept and identity link function. Also denote the coefficients of linear least square regression of X_j on T as initial values of W_j. If any coefficients in W_j are negative, remove the corresponding variables from the regression.

2. If any variables are removed, repeat step 1 until all the coefficients of the regression are positive. So far, the non-negativity constraints on X, T and W are satisfied. We denote this non-negative type matrix which has all coefficients non-negative as T_j^+, this is non-empty unless X is a zero matrix.

3. Compute the log likelihood by using equation 19, i.e.

$$L\left(T_j^+\right) = \sum_{i=1}^{p} \left(X_{ij} \log \left(T_j^+ W_j\right)_i - \left(T_j^+ W_j\right)_i\right)$$

4. For variables that are removed in Step 1 and Step 2, choose one and denote as $T_{j(new)}^+$, assign it to a very small positive coefficient that is denoted as ϵ, then compute the log likelihood for the new set of variables:

$$L\left(T_{j(new)}^+\right) = \sum_{i=1}^{p} \left(X_{ij} \log \left(T_{j(new)}^+ W_{j(new)}\right)_i - \left(T_{j(new)}^+ W_{j(new)}\right)_i\right)$$

   where $W_{j(new)} = (W_j(1-\varepsilon), \varepsilon)$. Here, we need to rescale W_j because we assume the data follow a Poisson distribution, so the sum of elements in X_j should equal the sum of TW_j. We have add a constraint on the column sum of T to be 1 previously, so we need $W_j^T 1 = W_j^T T^T 1 = X_j^T 1 = W_{j(new)}^T 1$, by defining $W_{j(new)}$ as above, we can ensure the sum of elements in $W_{j(new)}$ is the same as in X_j.

5. Compare $L\left(T_j^+\right)$ in step 3 and $L\left(T_{j(new)}^+\right)$ in Step 4 and if the latter is larger, i.e. by adding one more variable with non-negative coefficient ϵ, the log likelihood is increased, we use the new type matrix $T_{j(new)}^+$ then repeat Step 1-5. Otherwise set $W_{j(new)}$ to zero again.

6. Repeat Step 4 and Step 5 for all the removed variables.

Mote that the algorithm increases the log likelihood at each iteration, thus as long as the number of positive variables is finite, the convergence to maximum log likelihood is guaranteed.

3.3.3 Selecting the Number of Types

For the supervised NMF algorithm, we need to select the number of types $k_1 \cdots k_c$ for each class. These should be chosen in a way such that when we combine the individual type matrices together, the resulting matrix can most effectively discriminate these classes. A naive approach would be to attempt all combination of $k_1 \cdots k_c$ (i.e. for class i, k_i is chosen from $k_i = 1, \cdots, p$). Firstly, compute the combined type matrix T and combined weight matrix W on training samples for each combination. Then compare the model mis-classification errors on the validation set. However, this method is computationally expensive especially given the large number of genotypes p in our dataset. To perform this approach, p^2 combinations need to be attempted.

Here, we use another algorithm, proposed in [Cai et al. (2017)], which chooses the number of types for each class separately; then compares the deviances for each observation in the test set; followed by a Wilcoxon Rank-Sum test on these deviances; then standardise the test statistic for each fold and aggregate them into a single test statistic; finally k is chosen based on the standard deviation of this overall test statistic. For our dataset, there are only two classes of phenotypes. Suppose for each class we perform an r-fold cross-validation, such that in each iteration, we have $r - 1$ folds on training samples and 1 fold of test samples. To choose the number of types k_1 for class 1, we apply the following five steps for each value of $k_1 = 1, \cdots, p$:

1. For a fixed k_1, fit k_1 types on the training folds from class 1 to obtain the type matrix T by using the standard NMF algorithm .

2. Fit the remaining one test fold of class 1 samples, also one fold from the class 2 samples on T.

3. Compute one deviance for each data point in the samples specified in step 2. Note that the deviances for class 1 samples are required in order to make comparisons.

4. Use a Wilcoxon Rank-Sum test on these deviances to get one Z score for each fold. Notice that the deviances are not normally distributed, thus we use Wilcoxon Rank-Sum test which examines whether deviances from the two classes are systematically different from one another by using the ranks of these deviances [Wilcoxon (1992)]. If there is no statistical difference between the classes, the test statistic $W = \sum_{i=n+1}^{n+m} R_i$, where n is the number of samples from class 1 test fold and m is the number of samples in a single fold from class 2, R_i is the rank of the ith deviance in the two-class combined deviances, should follow a normal distribution with mean $\frac{m(n+m+1)}{2}$ and variance $\frac{nm(n+m+1)}{12}$. Then we can normalise the test statistic W to make it follow a standard normal distribution, namely, the Z score. We can obtain one Z-value for each fold of the cross-validation.

5. Sum the Z scores obtained from step 4 (r number of scores in total) then divide by $\sqrt{r}$, denote this statistic as Z_{all}. Dividing by $\sqrt{r}$ ensures that Z_{all} still follows

a standard normal distribution under the assumption that deviances from the two classes are from the same population.

6. Choose the smallest k for which Z_{all} is within one standard deviation of the largest Z_{all} score (p number of Z_{all} values in total).

Note that the power of the test is greatly improved by using the r-fold cross-validation and the combined Z scores, especially when the number of samples is small. Also, it is possible to assess whether the two classes can be easily separated or not using the Z scores. If the Z scores of deviances from class 1 are almost equal to those from class 2, this indicates that the current choice of k and T and W is sufficient for classifying samples from one class over the other; therefore, the overall standardised test statistic Z_{all} would have small variance. In contrast, when there is no clear separation, the Z scores from different folds tend to have large variance, and so does Z_{all}. For classes that are easily separable, the number of types is usually small while it is normally large when the separation is not clear. Thus, in a more general case when the number of classes is greater than 2, we first implement the above procedures for all classes, then identify the class which has easy separation from others, fix its number of types, then find the best-matching for other classes such that the mis-classification error is minimised.

3.4 Summary

To summarise, there are three types of supervised dimension reduction methods we introduced in this Chapter: PCA based, LFDA based and NMF based. These methods focus on different aspects of the features. Supervised PCA aims to maximise the dependence between transformed features and the labels. LFDA / KLFDA combines LPP and FDA such that the local structure in the original feature space is remained in the reduced space. Supervised NMF is designed based on the fact that all elements in our feature matrix is non-negative. PCA and LFDA related SDR algorithms allow us to incorporate kernel functions, thus can be used for both linear and non-linear feature transformation, where as supervised NMF can only reduce the dimensionality via linear approach.

In kSPCA, one may decide to use a Gaussian kernel, then necessary to tune the hyperparameter - the Gaussian width - appropriately. One possible solution is to maximise the average kernel values of samples within the same class while minimise this average for samples in different classes, details are provided in section 5.2. Alternatively, one could select the Gaussian width that maximises the HSIC or Kernel Target Alignment (KTA) between the Gaussian kernel matrix and a second matrix constructed from the labels [Damodaran (2018)].

Compared to PCA, NMF may be more interpretable in our scenario. Specifically, all elements in the weight matrix W of NMF are non-negative, while in PCA the sign of elements in the transformation matrix U is not deterministic. In the context of genotype data, it is not very interpretable if we construct a new feature via subtracting some genotypes, as in PCA. Also as demonstrated in [Lee and Seung (2001)], NMF tends to have good performance when the features are sparse. Given that in our dataset, most features have value 0, which makes NMF-based dimension reduction methods possibly more promising. However, in supervised NMF, the amount of computations required for iteratively refitting the Poisson regression in order to obtain the weight matrix W and for carrying the CV procedure to decide the number of types for different classes is enormous, this may make supervised NMF computationally prohibitive.

As for LFDA / KLFDA, when calculating the affinity matrix A, a CV procedure may be required for choosing the number of nearest neighbours. For some datasets, K in KNN can be chosen relatively easily if prior knowledge about the data structure is available. Also [Zelnik-Manor and Perona (2005)] experimented with various types of datasets (artificial data, image and high-dimensional data), then states that for local scaling affinity matrix, $K = 7$ works well in general. LFDA / KLFDA is invariant under linear transformations, implying that range of the transformation matrix can be uniquely determined, although the distance metric in the embedding space cannot be determined [Sugiyama (2007)].

4 Experiment on the Artificial Dataset

4.1 Data

In this chapter, we apply the supervised dimension reduction and density ratio estimation methods introduced above to an artificial genotype dataset that is generated using the *msprime v.7.1* simulator [Kelleher et al. (2016)]. This simulator efficiently samples coalescent trees under a range of evolutionary scenarios, then produces genotypes based on a coalescent model for chromosome 20. The model assumes a mutation rate of 2×10^8 mutations per base pair per generation, also it incorporates a recombination map of GRCh37. To generate samples from European and African ancestries, we used a demographic model previously inferred using the 1000 Genomes dataset [Gravel et al. (2011)], taking into account migration events and growth rates. We simulated 5,000 individuals from each population and each sample consists of 10,000 genotypes.

We then simulated the phenotypes $\boldsymbol{Y}$ from a normal distribution such that $\boldsymbol{Y} \sim N(\boldsymbol{X\beta} + \boldsymbol{\epsilon})$. $\boldsymbol{\beta}$ is the coefficient vector and $\boldsymbol{\epsilon}$ represents the random noise which follows $N(0, var(\boldsymbol{X\beta}))$. Note that most elements in $\boldsymbol{\beta}$ are set to zero since only a small proportion of genotypes are causal to a phenotype. The non-zero coefficients are drawn from a standard normal distribution. Additionally, in both populations, it is important to use the same $\boldsymbol{\beta}$ as well as to generate $\boldsymbol{\epsilon}$ from the same distribution. This is because in covariate shift, the assumption $P\left(\mathbf{y}^{\text{Eur}} \mid \mathbf{X}^{Eur}\right) = P\left(\mathbf{y}^{Afr} \mid \mathbf{X}^{Afr}\right)$ needs to be satisfied. The next step is to convert the continuous phenotype values into discrete labels. To do this, we pool all simulated African and European phenotypes together and label ones that are above the 80th quantile as positive and negative if otherwise. In practice, although the prevalence to a certain disease may differ among populations, the divergence should be within a reasonable range. Hence, we limit the proportion of positive phenotypes in both populations to be between 10% and 25%, and resample the phenotypes if this criterion is violated.

4.2 Implementation

For dimension reduction methods that incorporate a kernel matrix, namely KSPCA and KLFDA, the kernel parameter σ must be specified. For two points $\mathbf{x}, \mathbf{z}$, the Gaussian kernel function is defined as

$$\kappa(\mathbf{x}, \mathbf{z}, \sigma) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right).$$

If we consider quantities in a Gaussian kernel matrix as a distance measure, we wish to choose the Gaussian width σ appropriately such that in a mapped feature space, two samples with the same phenotype value are close (i.e. $\kappa(\mathbf{x}, \mathbf{z}, \sigma) \approx 1$) and they are far apart otherwise (i.e. $\kappa(\mathbf{x}, \mathbf{z}, \sigma) \approx 0$) [Li et al. (2010)]. Denote ω_i as a set of observations that are in class $i, i = 1, \cdots, L$ (in our dataset, $L = 2$), the mean "distance" of samples in the same class is

$$w(\sigma) = \frac{1}{\sum_{i=1}^{L} |\omega_i|^2} \sum_{i=1}^{L} \sum_{\mathrm{x} \in \omega_i} \sum_{z \in \omega_i} \kappa(\mathrm{x}, z, \sigma),$$

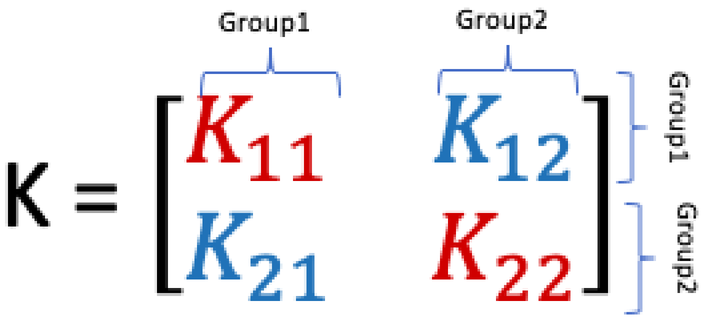
where $|\omega_i|$ is the number of samples within class i. The between class mean "distance" is

$$b(\sigma) = \frac{1}{\sum_{i=1}^{L}\sum_{\substack{j=1\\j\neq i}}^{L} |\omega_i|\,|\omega_j|} \sum_{i=1}^{L}\sum_{\substack{j=1\\j\neq i}}^{L}\sum_{x\in\omega_i}\sum_{z\in\omega_j} \kappa(\mathbf{x},\mathbf{z},\sigma).$$

We can scan a range of values for the parameter σ, then choose the one that simultaneously makes $w(\sigma)$ close to 1 and $b(\sigma)$ close to 0, i.e. we try to find σ which minimises

$$J(\sigma) = 1 - w(\sigma) + b(\sigma).$$

We further illustrate this method using a binary example. For a more intuitive explanation, suppose the samples are re-ordered by their labels such that observations from the same class are placed together. Then the kernel matrix is



where K_{ij} is a collection of kernel matrix elements between samples from group i and group j. Then $w(\sigma)$ corresponds to the mean of quantities in the diagonal clusters K_{11} and K_{22}. $b(\sigma)$ is the mean of elements in the off-diagonal clusters (i.e. K_{12} and K_{21}). Then we choose σ that maximises $J(\sigma)$.

For LFDA and KLFDA, one needs to select the dimensionality of the reduced space before implementation. We use a cross-validation procedure to choose from a range of dimensions. To save computational costs, a small subset of samples are split into, say, 5 folds. For each dimension, we train the LFDA/KLFDA with the 4 training folds then use its results to transform the features in the test fold. In the reduced space, a one-nearest-neighbour classifier is applied to the test samples, then the misclassification rate is computed. We repeat these steps on each of the CV test folds and select the dimension that minimises the average misclassification rate.

Theoretically, to choose the optimal number of types for each class of labels in SNMF (section 3.3.3), it is required to perform cross-validation on all possible dimensions. However, in this dataset the feature dimension is large and makes the algorithm computationally expensive. To ease the computational burden, we use the results of other dimension reduction methods to set an upper limit on the number of types that each class is allowed to keep in the reduced space. Hence, we perform CV up to a small number of dimensions, rather than all dimensions of the genotypes.

4.3 Results

With each set of estimated weights, we fit a LASSO logistic regression using the European samples. The LASSO tends to give sparse solutions, which is compatible to the prior knowledge that a phenotype is only associated to a small subset of genotypes. A logistic regression is preferred since the phenotypes are binary. Then we use the model to predict the phenotypes for the African samples. The results are shown in table 2. Furthermore, the weight estimates may have a large range, causing a large variance on predictive performance. Thus, one may consider using the exponentially-flattened importance weights, defined as $\hat{\beta}(x)^\gamma$, where $\gamma \in [0, 1]$ is the exponential flattening parameter. $\gamma = 0$ reduces the model to a standard LASSO whereas $\gamma = 1$ refers to the unflattened weights. Choosing $0 < \gamma < 1$ will give an intermediate estimator that balances the trade-off between statistical efficiency and consistency. Results with $\gamma = 0.5$ are given in table 3.

In this example, LFDA along with KLIEP yields similar results as the default setting where the weights all equal 1. This is because this model predicts the most sample wights to be close to 1 (see Figure 2). Methods with KSPCA and SNMF assign zero weights to the majority of samples, which significantly reduce the size of effective training samples and lead to poor predictive performance. Some models (such as KSPCA + KLEIP and KSPCA + uLSIF) failed to provide predictions due to possibly numerical instability and small dimensionality of the dimension reduced space, further details will be discussed in the end of Chapter 5.3. The model KSPCA + KMM and model KLFDA + KLIEP are essentially intercept-only models. This is due to the number of samples the model trained upon is too small. In the exponentially-flattened case, the LASSO also reduces all regression coefficients in KLFDA + KLIEP to zero, although this model has fewer training samples with zero weights. Methods using LFDA tends to be more effective on this dataset, possibly due to a relatively large effective sample size. In this experiment, the exponentially-flattened weights indeed have a smaller variance comparing to the raw wights, however, this does not lead to an improved model performance. One explanation is that when the density-ratios are estimated poorly, all flattened weights are also unreliable and then covariate shift adaptation fails work.

method	AUC	sensitivity	specificity	accuracy
default	0.651	0.393	0.801	0.700
KLFDA + KLIEP	0.5	1	0	0.245
KLFDA + KMM	0.511	0.344	0.665	0.586
KLFDA + ULSIF	0.531	0.635	0.419	0.472
KSPCA + KMM	0.5	1	0	0.245
LFDA + KLIEP	0.655	0.295	0.802	0.702
LFDA + KMM	0.621	0.362	0.791	0.685
LFDA + ULSIF	0.652	0.388	0.800	0.698
SNMF + KMM	0.546	0.400	0.678	0.609
SPCA + KLIEP	0.587	0.337	0.782	0.673
SPCA + KMM	0.592	0.338	0.783	0.673
SPCA + ULSIF	0.593	0.342	0.784	0.675

Table 2: Model performance on the simulated dataset using the raw density-ratio estimates. Default corresponds to weights all equal to 1.

method	AUC	sensitivity	specificity	accuracy
default	0.651	0.393	0.801	0.700
KLFDA + KLIEP	0.5	1	0	0.245
KLFDA + KMM	0.497	0.247	0.752	0.628
KLFDA + ULSIF	0.592	0.316	0.776	0.662
KSPCA + KMM	0.502	0.268	0.731	0.617
LFDA + KLIEP	0.651	0.392	0.801	0.700
LFDA + KMM	0.640	0.382	0.797	0.695
LFDA + ULSIF	0.651	0.390	0.800	0.699
SNMF + KMM	0.540	0.276	0.763	0.643
SPCA + KLIEP	0.611	0.346	0.786	0.677
SPCA + KMM	0.615	0.350	0.787	0.679
SPCA + ULSIF	0.615	0.362	0.791	0.685

Table 3: Model performance on the simulated dataset using the flattened weights ($\gamma = 0.5$).

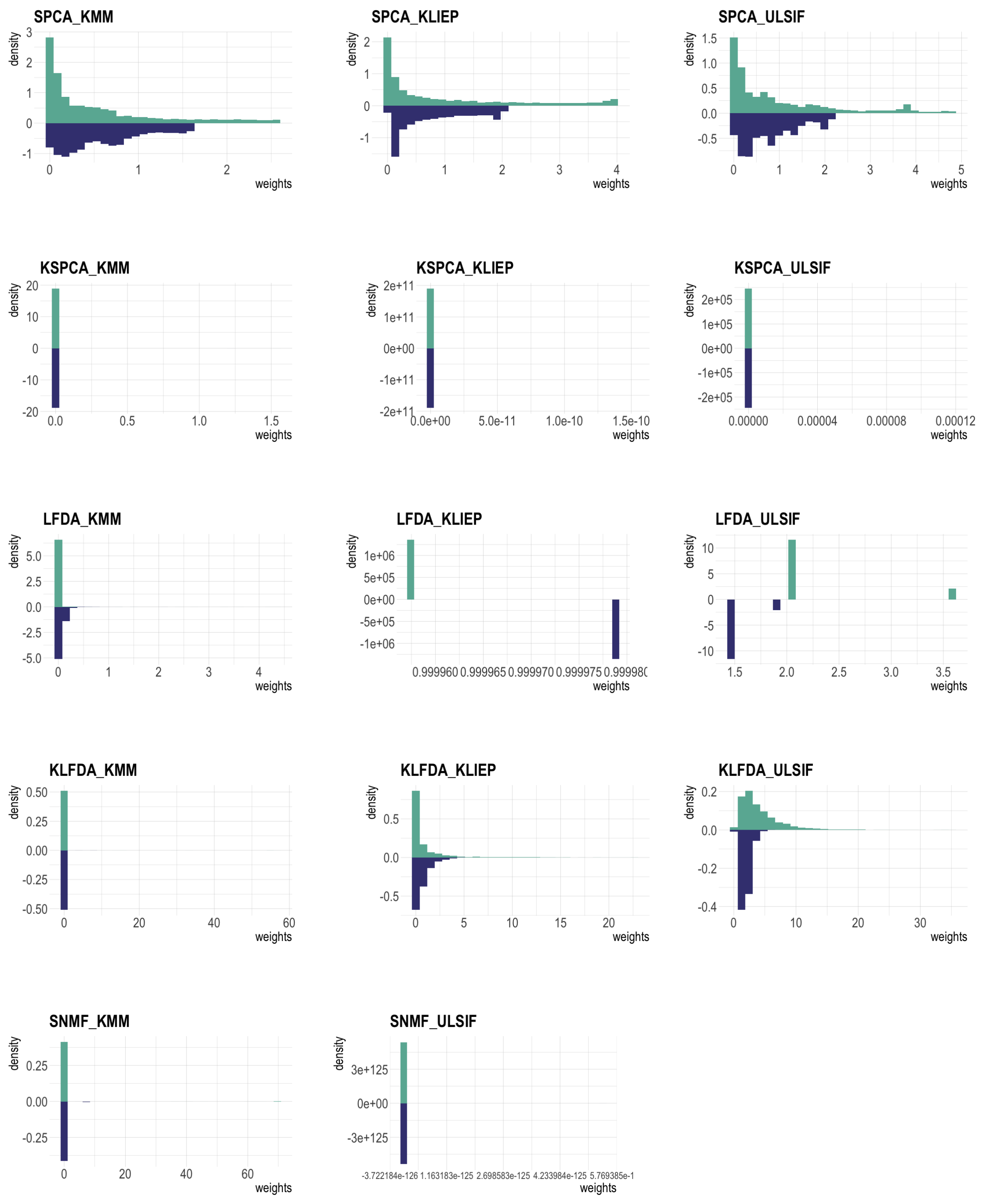


Figure 2: Each mirrored histogram represents a set of weight estimates. The upper half in green indicates the raw weights, the lower half in blue indicates the exponentially-flattened weights with $\gamma = 0.5$.

5 Experiment on the UK Biobank Dataset

5.1 Data

The Quality Controlled UK Biobank genomic data consists of genome-wide genotype data from 406,263 individuals across the United Kingdom [Bycroft et al. (2018)]. Among all participants, 219,135 are female (54%) and 187,128 are male (46%). At the time they registered the project, the youngest participant aged 49 and the eldest aged 83, half of the samples have age between 61 and 75. The blood, urine and saliva samples from each participant were provided, physical measurements were recorded (such as height, BMI, waist circumference etc.) along with a questionnaire focusing on their health and lifestyle (including whether each individual is diagnosed to asthma, breast cancer, prostate cancer, type II diabetes, high cholesterol, high blood pressure etc.). The genotypes of individuals were assayed using two Affymetrix platforms: 364,666 samples (89.8%) in the Applied Biosystems UK Biobank Axiom Array and the rest 41,597 samples (10.2%) in the UK BiLEVE Axiom. A series of Quality Control measures have been carried out, for instance (1) the overall missing rate of each remaining genotype cannot exceed 5% (2) minor allele frequencies (MAF) is at least 0.0001 (3) the missing call rate for each individual cannot exceed 10% [Biobank (2015)], which leaves us 784,256 genotypes that meet the criterion. Here are 4 important features related to this dataset that may affect our design of methods and experiment, which are illustrated as the following:

First, the size of this dataset is unprecedented in biomedical science field, and hence it provides the opportunity for a better understanding of human biology and diseases. However, it has also raised analytical challenges: although genotypes only take value from 0,1 and 2 and a large proportion are 0, meaning that this giant genotype array can be stored as a sparse matrix and hence ease the memory needed for storing it, the size of this matrix is still too significant and takes more than 1200 Gb of storage. Additionally, computation required for analysis grows with the number of samples and the size of variables (and depending on the complexity of algorithms, it can sometimes grow exponentially). Given the size of the matrix, this limits the range of algorithms can be implemented. Hence, one needs to carefully design the sampling method such that the validity of analysis is not weakened, and ensure that any algorithms employed are computationally feasible.

Second, this dataset is highly imbalanced in the ethnic background of participants: 94% participants reported their ethnicity as group "White" and remarkably, 93% of them are British. A list containing the detailed information about the samples' ethnicity is shown in table 4. This disproportionality in populations means that estimating using this data may be appropriate for White populations but cannot be transferred to other ethnic groups.

Third, the genotypes vary noticeably among different ethnic backgrounds. Principal component analysis (PCA) applied to the genotype matrix can be used to capture the population structure. The 10 major PCs are plotted in consecutive pairs (see figure 3) illustrating that populations from the similar ancestries also have the similar principal component scores. In particular, the PC1 and PC2 separates out the Black and Black British samples, whereas PC3 and PC4 indicates Asian and Asian British. These results demonstrate the importance of accounting for ethnic background in related genetic analysis. Otherwise the minority groups may be treated as random noise or outliers,

self-reported ethnicity		proportion(%)	
White		93.81	
	British		87.45
	Irish		2.62
	Any other White background		3.74
Asian		2.11	
	Indian		1.26
	Pakistani		0.39
	Bangladeshi		0.05
	Any other Asian background		0.41
Chinese		0.35	
Black		1.72	
	African		0.76
	Caribbean		0.93
	Any other Black background		0.03
Mixed		0.63	
	White and Asian		0.18
	White and Black African		0.09
	White and Black Caribbean		0.13
	Any other mixed background		0.23
Other/Unknown		1.38	

Table 4: Self-reported ethnicity categories in the 406,263 UK Biobank samples.

reducing the model's performance.

Typically, only a small fraction of genetic variants have a causal relationship to a specific phenotype. Using White British participants from UK Biobank, [Canela-Xandri et al. (2018)] analysed the genetic associations for 118 non-binary and 599 binary traits by fitting linear mixture models. It shows that about one in 7,500 genotypes reached the conventional genome-wide significant thresholds ($P < 10^{-8}$) for binary traits and one in about 300 genotypes was significant for non-binary traits. Thus, one may consider models that encourage sparse solutions or other screening methods that filters out non-significant variants.

5.2 Experiment Design

Ordinary machine learning algorithms split a dataset into training, validation and test samples, where the training data are used to learn the parameter, the validation samples are for comparing the performance of various models, then the test set is for computing an unbiased risk associated to the best performing model. However, in our dataset, considering the number of Black individuals is significantly smaller than the White individuals, the predictive risks may have a relatively large variance. To alleviate this, we use cross-validation, splitting the Black samples into 5 portions. Then we treat one portion as the test set and take the Black samples in the other four portions as training samples. To reflect the imbalance of the ethnicity in the whole dataset, we further scan

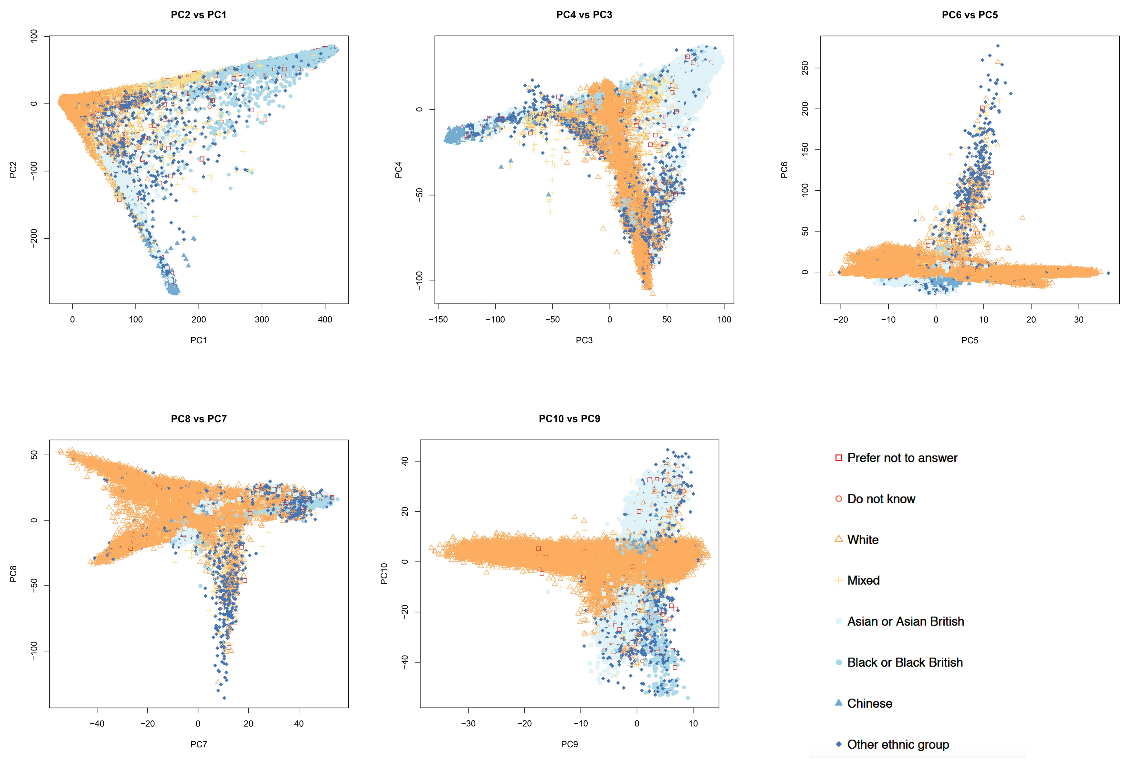


Figure 3: First 10 PCs of all genotypes in the UK Biobank dataset, colours and shapes of points indicate the corresponding ethnic group.

the White samples and select ones that best match the current training set Black samples in gender and age, such that in the overall training set, Black samples count for a small proportion of total all participants, say, 20%. Finally we repeat this procedure for each of the Black portions to obtain 5 train-test sets, for each test set only Black samples are included and each training set is a mix of Black and White samples where the latter holds the majority. Based on this train-test split, we can generate 5 separate predictive risk estimates, taking their mean for a more accurate assessment on effectiveness of the models.

Each individual has 784,256 genotypes recorded. Combined with the large sample size, this makes the computation highly costly. Additionally, only a small number of variants are in fact causal to a phenotype. Hence we implement a pre-screening process to provide us a preliminary understanding about the importance of each variant. For a genotype g, we fit a linear regression denoted as

$$Y = \beta_x \mathbf{X} + \beta_g g + \epsilon,$$

where Y is the phenotype, ϵ is the noise that follows $N(0, \text{var}[Y\mathbf{X}+g])$ and assuming this variance is constant across all samples. X is a covariate matrix that contains variables: age and the first 10 PCs of all genotypes. Note that the PCs in the regression can be considered as an indicator of the population structure. We loop this over all genotypes and for each regression, collect the p-value of each genotype (which correspond to the null hypothesis that $\beta_g = 0$) and rank these p-values, then only select ones which have small p-values. Theoretically, a logistic regression would be more appropriate since the disease phenotypes are binary. However, in practice, estimating coefficients through iteratively re-weighted least squares (IWLS) is computationally expensive especially given that we need to fit 780k+ regressions, hence linear regressions that are more simple to solve are used instead. We can reduce the computations by swapping the position of phenotype and genotype [Voorman et al. (2012)], i.e. we fit the following model instead

$$g = \gamma_x \mathbf{X} + \gamma_y Y + \xi.$$

This is valid because $\gamma_y = 0$ is equivalent to $\beta_g = 0$ in previous model and yields the same p-value. Whereas in the swapped model, the covariates for all regressions are unchanged so that the hat matrix, also other quantities that are solely dependent of the feature matrix only need to be computed once. To prevent data leakage, this pre-screening is applied to each train-test split separately. Then observations used to fit the regression are restricted to ones in each training set, then the results are deployed directly to the test set. In this experiment, we select 5% of lowest p-value variants, which is about 40,000 in size.

The original dataset contains missing values due to a number of reasons. For example, some variants are particularly complicated such that they are currently not supported by the genotyping analysis pipeline. Therefore, for each train-test split, genotypes that all values are missing in either training or test folds are removed. Also since we only keep a relatively small number of variants after screening, there would be cases, although rare, that two samples have exactly the same post-screening features. It is necessary to removed the duplicated samples, otherwise in later procedure, the kernel matrices constructed with the duplicated observations would be singular. For the rest of missing quantities, they are imputed with the mean value of its corresponding genotype.

Then for each train-test split, we apply the supervised dimension reduction methods on the training samples and employ the results onto the test samples. On the lower dimensional feature space, we then apply the density ratio estimation algorithms. We use the same algorithms to choose the Gaussian kernel width and the dimension of LFDA reduced space as in the artificial dataset (Chapter 4.2). Once the weights for training samples are obtained, we fit the training samples into a LASSO logistic regression. To tune the LASSO regularisation parameter, we further split each training set into 5 inner folds and use cross-validation to tune the LASSO regularisation parameter, the average AUC across inner folds is used as our evaluation metric and the regularisation parameter with largest AUC is chosen. Once we have both the optimal regularisation parameter and training sample weights, we finally apply the LASSO logistic regression to make predictions on the test set, then evaluate the performance across different test folds. This experiment design is illustrated in Figure 4.

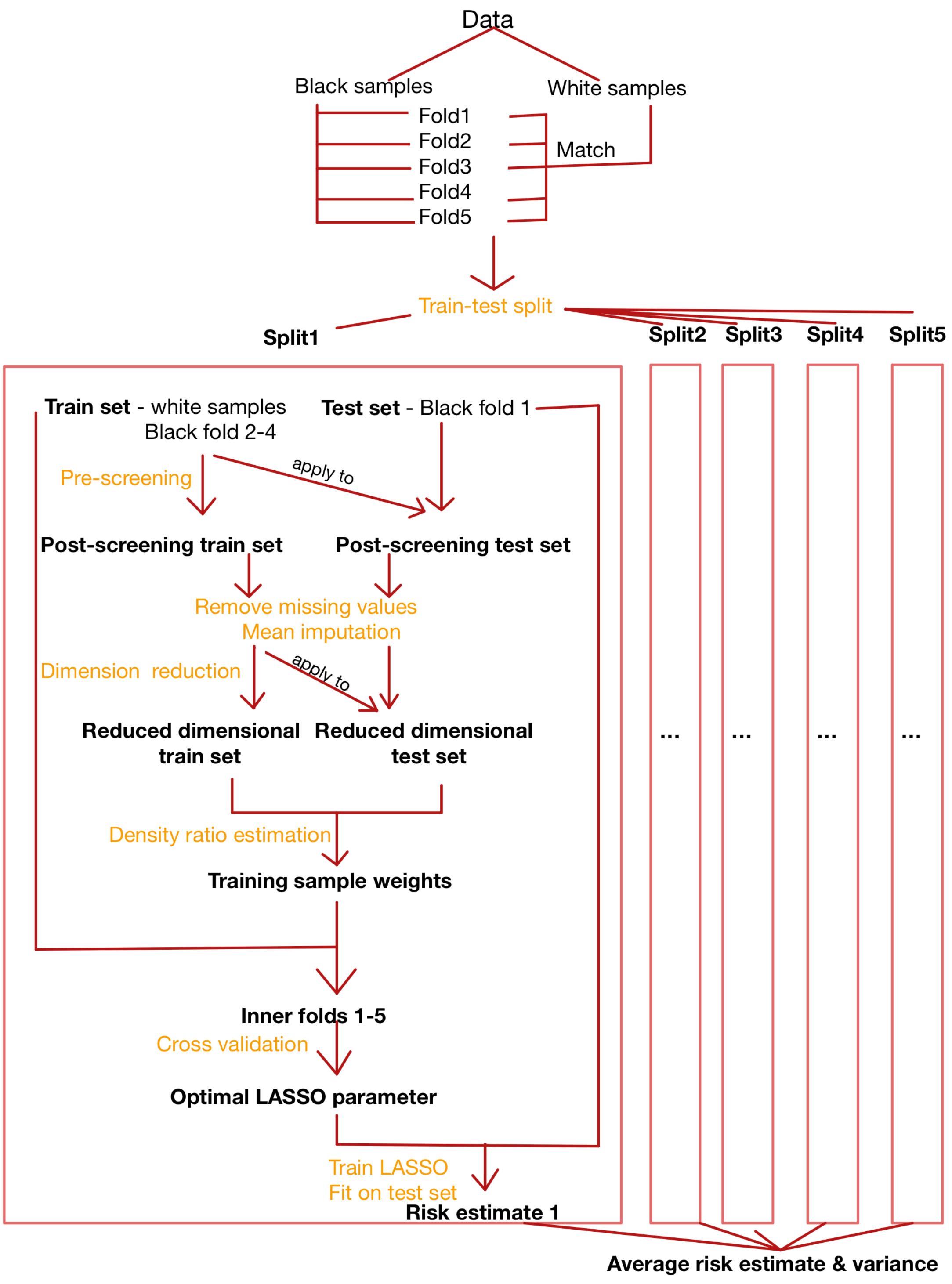


Figure 4: The design of experiment.

5.3 Results

We use the AUC as our main evaluation metric since it considers all possible decision thresholds. In medical diagnosis, one scientific interest is to find an optimal threshold that balances the sensitivity (probability of being test positive when disease is present) and specificity (probability of being test negative when disease is absent). A possible solution is to select one that maximises the Youden's J statistic, defined as

$$J = sensitivity + specificity - 1.$$

This cutoff point maximises the difference between the true positive rate and the false positive rate. With this threshold, we further report each model's sensitivity, specificity and accuracy. The results are summarised in Table 5 and Figure 5 shows density plots for each set of weight estimates.

method	AUC	sensitivity	specificity	accuracy
default (5)	0.568 (0.018)	0.554 (0.133)	0.642 (0.133)	0.639 (0.128)
KSPCA + KMM (2)	0.546 (0.018)	0.529 (0.195)	0.699 (0.151)	0.694 (0.141)
KSPCA + KLIEP (4)	0.546 (0.020)	0.593 (0.176)	0.617 (0.163)	0.579 (0.183)
LFDA + KLIEP (4)	0.564 (0.017)	0.514 (0.114)	0.672 (0.126)	0.668 (0.120)
SPCA + KLIEP (5)	0.535 (0.056)	0.633 (0.118)	0.510 (0.144)	0.480 (0.138)
SPCA + KMM (4)	0.525 (0.035)	0.708 (0.269)	0.431 (0.244)	0.586 (0.223)
SPCA + ULSIF (3)	0.568 (0.023)	0.405 (0.086)	0.800 (0.068)	0.791 (0.066)

Table 5: Models' performance on the UK Biobank dataset. For each evaluation metric, the numbers in brackets indicate the standard deviation. The numbers in the first column after name of each algorithm correspond to the number of folds a model has prediction results. Due to numerical instability, for some models the LASSO did not converge on all 5 train-test splits.

We did not perform SNMF here since it requires to iterate over all genotypes to find the optimal weight matrix W and type matrix T. Given the dimensionality of the Biobank dataset, it is computationally infeasible to implement it. The experiment results show that similar as in the simulated dataset, a number of methods, especially ones using KSPCA, tend to assign zero weights to a large proportion of training samples, which reduces the effective sample size and further leads to poor predictive performance. For this dataset, the only model that is comparable to the standard LASSO model applies SPCA along with ULSIF. Although our model has a marginally larger standard deviation in AUC, it significantly outperforms the default setting in predictive accuracy, given that the decision threshold is chosen by the Youden's score. This is possibly because our model gives large weights (>5) to Black samples in the each training set. From the figures, the model LFDA + KLIEP tends to filter out unimportant training samples rather than strengthening the Black samples in the training data, hence it has a narrow range of weights estimates without either large or small values.

The genotypes are likely to be correlated and the features post-screening can be similar to each other, hence numerical instability are observed during the implementation of these methods. For example, the within-sample covariance in LFDA and the kernel matrix in both KSPCA and KMM are occasionally nearly singular. The idea is to add a small constant, for example 10^{-6}, which works as a regulariser to the diagonal elements of these matrices if their determinants are almost 0. However, during the implementation, we have spotted that the smallest constant needed to make these matrices positive definite can be up to the magnitude of 10^{-1}, which greatly affects the results and sometimes leads to unreliable weight estimates. Another possible approach is to find the nearest positive definite matrix via the alternating projection method [Higham (2002)], but the computational requirement for working on such giant matrices makes it less effective in this application.

Secondly, the density plots demonstrate that the weights for Black samples do not appear to be significantly larger than the White samples, although the former exhibits more similar characteristics to the test set samples. This is possibly because the dimensionality of the feature reduced space is too small. For example, in SPCA and KSPCA, we keep the number of transformed covariates up to the threshold that 95% of dependency of the phenotypes against genotypes is explained. This maps the original features onto a one-dimensional space. Also in other methods, the reduced dimension is normally less than 5. Thus, the feature mapping might have reduced the dissimilarity between samples from different populations and subsequently causes the density-ratio estimation methods to perform ineffectively.

Also during the implementation, the LASSO regressions are fitted with the R package GLMNET [Friedman et al. (2009)] which applies a second-order approximation at the beginning of each iteration and then apply a cyclic coordinate descent method to minimise a quadratic function. However, there is no guarantee that GLMNET converges to an optimum and this issue is encountered during our implementation, causing numerous models fail to provide results. Our initial guess was the large variance of the weight estimates causes numerical instability. Thus, we have carried out separate trials where the weights are exponentially flattened, then we imposed an upper limit such that the maximal weights do not exceed 5, we further centred the weights such that their means equal one. Even with these measures, the LASSO was still unable to converge. [Yuan et al. (2010)] compares various softwares for solving L1-regularised classifiers, then states that due to the approximation procedure employed throughout the GLMNET implementation, it is less stable than coordinate descent methods in training large-scaled datasets. To tackle the convergence problem, one needs to modify the Fortran code which the GLMNET is implemented upon, however, this is beyond the scope of our research interest.

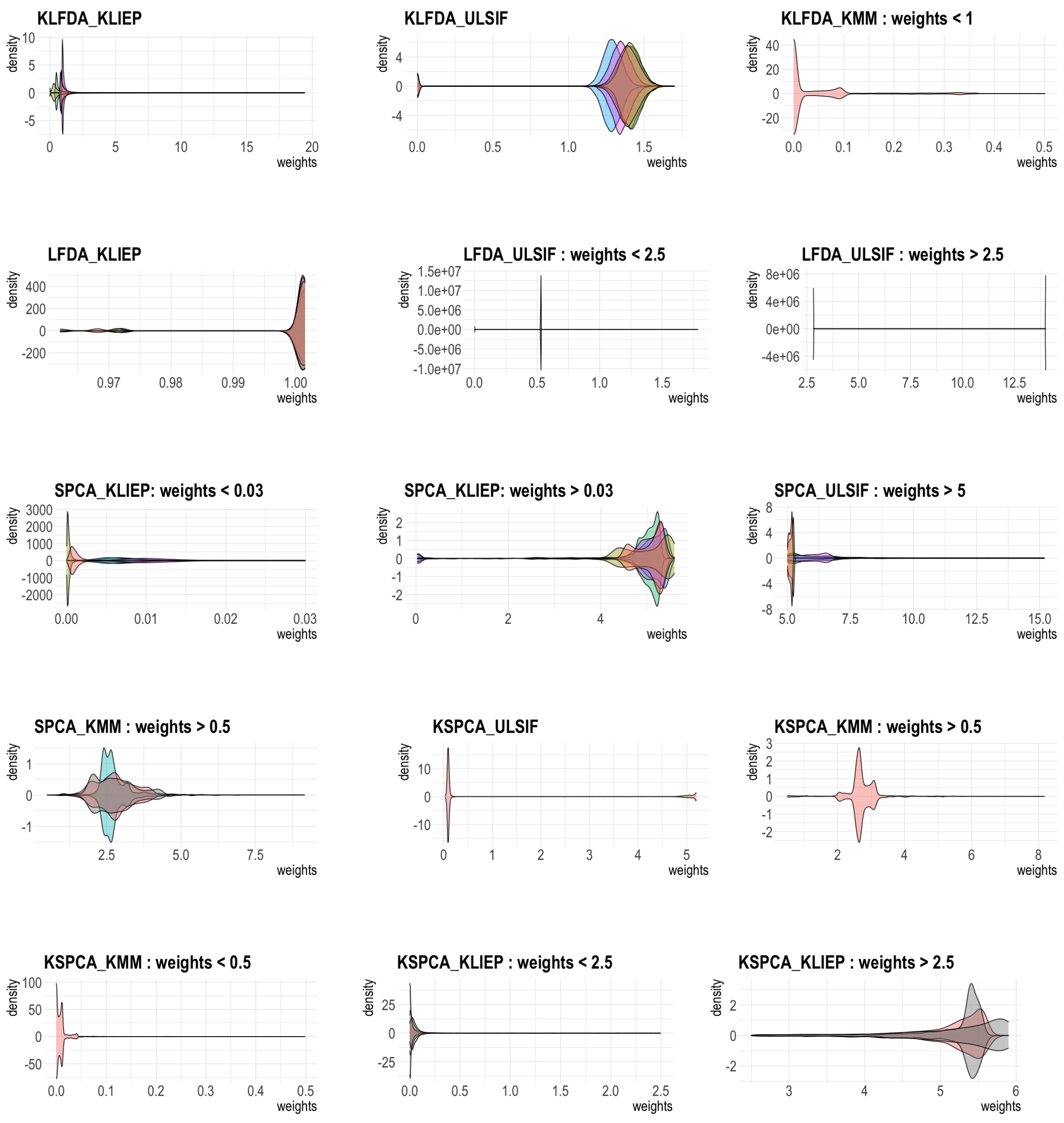


Figure 5: Density plots for weight estimates. Each colour represents a train-test split, the upper half of the plots shows the weights for the White individuals (account for approximately 80% of training samples), the lower half shows weights for the Black samples. For a clear visualisation, models SPCA + KLIEP, KSPCA + KMM, KSPCA + KLIEP and LFDA + ULSIF are each split into two subfigures with different scale on the y-axis. For SPCA + ULSIF, the plot here only shows weights that are greater than 5, this counts for about 10% of total training samples, the rest 90% are close to 0. Similarly for SPCA + KMM, this plot only shows the top 20% of weight estimates, the remaining are zeroes.

6 Further Work

In the pre-screening stage, we use univariate linear regressions asa from of supervised dimension reduction. This assumes that the variants are independent from each other, which is in general not true in a genotype dataset. In an extreme case, if only 1 genotype is statistically significant while all others are highly correlated to it, the current pre-screening approach would identify all genotypes as important ones. Hence, it may worth attempting more sophisticated algorithm that accounts for the correlation structure between genotypes, such as Guided Regularised Random Forest [Deng and Runger (2013)].

For SDR, some machine learning/deep reduction methods such as autoencoder and variational autoencoder may be able to capture useful information about the structure of the genotypes. Supervised variants may also enhance predictive performace [Ghosh and Kirby (2020), Le et al. (2018)].

The existing GLMNET implementation lacks theoretical convergence properties and may be ineffective in solving high dimensional problems. [Yuan et al. (2012)] proposed an improved GLMNET which finds an optimal solution by firstly incorporating the coordinate descent approach, followed by the Newton method. The asymptotic convergence is guaranteed, making it potentially more useful for large-scaled genomics datasets.

In this report, the proportion participants from the minority group in each training set was fixed at 20%. If the LASSO logistic regression can converge properly, further work could examine how the performance of our density ratio estimation techniques change with this proportion. Similarly, one could vary the number of variants selected in the pre-screening stage, then apply use the same models to assess whether our algorithms can still work effectively across various dimensionality.

In this work, we estimate the sample importance in the reduced dimension space, then fit a LASSO logistic regression on the original feature space. An alternatively worth consideration would be to both estimate the weights and fit the LASSO in the lower dimensional space. Given that we have abundant samples and features, some deep learning algorithms can possibly be applied to predict disease risks. For example, we can use Conv1D to reduce the dimensionality of features, followed by a LSTM to learn the sequential pattern of the dimension-reduced genotypes, then build classification models upon these results.

7 Conclusion

This work aims solve the problem that the training samples and the test samples differ in distributions by sample importance reweighting. We firstly introduced three density-ratio estimation methods - KMM, KLIEP and uLSIF - then illustrate that the performance of these models worsens when the dimensionality of features grows. Hence, we discussed a range of supervised dimension reduction methods, including SPCA, KSPCA, LFDA, KLFDA and SNMF. Then we implemented the above algorithms to a simulated genotyping dataset as well as the UK Biobank dataset. The results show that when applying to high dimensional datasets with complicated correlation structures, the algorithms may become numerically unstable, followed by a discussion about the possible reasons for this instability. Among those models that are computable, they marginally outperformed the standard LASSO logistic regression. Finally, we outlined the potential directions for improving the current designs.

References

Barshan, E., Ghodsi, A., Azimifar, Z., and Jahromi, M. Z. (2011). Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.

Biobank, U. (2015). Genotyping and quality control of uk biobank, a large-scale, extensively phenotyped prospective resource. *Available at biobank. ctsu. ox. ac. uk/crystal/docs/genotyping_qc. pdf. Accessed April*, 1:2016.

Bycroft, C., Freeman, C., Petkova, D., Band, G., Elliott, L. T., Sharp, K., Motyer, A., Vukcevic, D., Delaneau, O., O'Connell, J., et al. (2018). The uk biobank resource with deep phenotyping and genomic data. *Nature*, 562(7726):203–209.

Cai, Y., Gu, H., and Kenney, T. (2017). Learning microbial community structures with supervised and unsupervised non-negative matrix factorization. *Microbiome*, 5(1):110.

Canela-Xandri, O., Rawlik, K., and Tenesa, A. (2018). An atlas of genetic associations in uk biobank. *Nature genetics*, 50(11):1593–1599.

Carlson, C. S., Matise, T. C., North, K. E., Haiman, C. A., Fesinmeyer, M. D., Buyske, S., Schumacher, F. R., Peters, U., Franceschini, N., Ritchie, M. D., et al. (2013). Generalization and dilution of association results from european gwas in populations of non-european ancestry: the page study. *PLoS Biol*, 11(9):e1001661.

Damodaran, B. B. (2018). Fast optimal bandwidth selection for rbf kernel using reproducing kernel hilbert space operators for kernel based classifiers. *arXiv preprint arXiv:1804.05214*.

Deng, H. and Runger, G. (2013). Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Fishman, G. (2013). *Monte Carlo: concepts, algorithms, and applications*. Springer Science & Business Media.

Friedman, J., Hastie, T., and Tibshirani, R. (2009). glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, 1(4).

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175.

Fukunaga, K. (2013). *Introduction to statistical pattern recognition*. Elsevier.

Fulton, W. and Harris, J. (2013). *Representation theory: a first course*, volume 129. Springer Science & Business Media.

Ghojogh, B. and Crowley, M. (2019). Unsupervised and supervised principal component analysis: Tutorial. *arXiv preprint arXiv:1906.03148*.

Ghojogh, B., Karray, F., and Crowley, M. (2019). Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*.

Ghosh, T. and Kirby, M. (2020). Supervised dimensionality reduction and visualization using centroid-encoder. *arXiv preprint arXiv:2002.11934*.

Gravel, S., Henn, B. M., Gutenkunst, R. N., Indap, A. R., Marth, G. T., Clark, A. G., Yu, F., Gibbs, R. A., Bustamante, C. D., Project, . G., et al. (2011). Demographic history and rare allele sharing among human populations. *Proceedings of the National Academy of Sciences*, 108(29):11983–11988.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

He, X. and Niyogi, P. (2004). Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160.

Higham, N. J. (2002). Computing the nearest correlation matrix—a problem from finance. *IMA journal of Numerical Analysis*, 22(3):329–343.

Jain, A. K., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37.

Kanamori, T., Suzuki, T., and Sugiyama, M. (2009). Condition number analysis of kernel-based density ratio estimation. *arXiv preprint arXiv:0912.2800*.

Kelleher, J., Etheridge, A. M., and McVean, G. (2016). Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS computational biology*, 12(5):e1004842.

Le, L., Patterson, A., and White, M. (2018). Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *Advances in Neural Information Processing Systems*, pages 107–117.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562.

Li, C.-H., Lin, C.-T., Kuo, B.-C., and Chu, H.-S. (2010). An automatic method for selecting the parameter of the rbf kernel function to support vector machines. In *2010 IEEE International Geoscience and Remote Sensing Symposium*, pages 836–839. IEEE.

Martin, A. R., Gignoux, C. R., Walters, R. K., Wojcik, G. L., Neale, B. M., Gravel, S., Daly, M. J., Bustamante, C. D., and Kenny, E. E. (2017). Human demographic history impacts genetic risk prediction across diverse populations. *The American Journal of Human Genetics*, 100(4):635–649.

Martin, A. R., Kanai, M., Kamatani, Y., Okada, Y., Neale, B. M., and Daly, M. J. (2019). Clinical use of current polygenic risk scores may exacerbate health disparities. *Nature genetics*, 51(4):584–591.

Need, A. C. and Goldstein, D. B. (2009). Next generation disparities in human genomics: concerns and remedies. *Trends in Genetics*, 25(11):489–494.

Pasaniuc, B., Zaitlen, N., Lettre, G., Chen, G. K., Tandon, A., Kao, W. L., Ruczinski, I., Fornage, M., Siscovick, D. S., Zhu, X., et al. (2011). Enhanced statistical tests for gwas in admixed populations: assessment using african americans from care and a breast cancer consortium. *PLoS Genet*, 7(4):e1001371.

Peterson, R. E., Kuchenbaecker, K., Walters, R. K., Chen, C.-Y., Popejoy, A. B., Periyasamy, S., Lam, M., Iyegbe, C., Strawbridge, R. J., Brick, L., et al. (2019). Genome-wide association studies in ancestrally diverse populations: opportunities, methods, pitfalls, and recommendations. *Cell*, 179(3):589–603.

Popejoy, A. B. and Fullerton, S. M. (2016). Genomics is failing on diversity. *Nature News*, 538(7624):161.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Smola, A. J. and Schölkopf, B. (1998). *Learning with kernels*, volume 4. Citeseer.

Stojanov, P., Gong, M., Carbonell, J. G., and Zhang, K. (2019). Low-dimensional density ratio estimation for covariate shift correction. *Proceedings of machine learning research*, 89:3449.

Sugiyama, M. (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of machine learning research*, 8(May):1027–1061.

Sugiyama, M., Kawanabe, M., and Chui, P. L. (2010a). Dimensionality reduction for density ratio estimation in high-dimensional spaces. *Neural Networks*, 23(1):44–59.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2010b). Density ratio estimation: A comprehensive review (statistical experiment and its related topics).

Tsuboi, Y., Kashima, H., Hido, S., Bickel, S., and Sugiyama, M. (2009). Direct density ratio estimation for large-scale covariate shift adaptation. *Journal of Information Processing*, 17:138–155.

Vilhjálmsson, B. J., Yang, J., Finucane, H. K., Gusev, A., Lindström, S., Ripke, S., Genovese, G., Loh, P.-R., Bhatia, G., Do, R., et al. (2015). Modeling linkage disequilibrium increases accuracy of polygenic risk scores. *The american journal of human genetics*, 97(4):576–592.

Voorman, A., Rice, K., and Lumley, T. (2012). Fast computation for genome-wide association studies using boosted one-step statistics. *Bioinformatics*, 28(14):1818–1822.

Wang, H. and van der Laan, M. J. (2011). Dimension reduction with gene expression data using targeted variable importance measurement. *BMC bioinformatics*, 12(1):312.

Wang, Y., Miller, D. J., and Clarke, R. (2008). Approaches to working in high-dimensional data spaces: gene expression microarrays. *British journal of cancer*, 98(6):1023–1028.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. (2013). Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370.

Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2010). A comparison of optimization methods and software for large-scale l1-regularized linear classification. *The Journal of Machine Learning Research*, 11:3183–3234.

Yuan, G.-X., Ho, C.-H., and Lin, C.-J. (2012). An improved glmnet for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13:1999–2030.

Zaitlen, N., Paşaniuc, B., Gur, T., Ziv, E., and Halperin, E. (2010). Leveraging genetic variability across populations for the identification of causal variants. *The American Journal of Human Genetics*, 86(1):23–33.

Zelnik-Manor, L. and Perona, P. (2005). Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608.

The following provides important R functions for implementing methods introduced in this report.

The full code can be found in: https://github.com/shliu99/MSc-project-in-covariate-shift

Appendix A R code: Pre-screening

```r
# Estimate the p-value for each genotype in each train-test split
# load libraries
library(devtools)
library(snpnet)
library(readr)
library(dplyr)
library(parallel)

# set up parameters
plink2_path  <- "/apps/eb/skylake/software/PLINK/2.00a2.3_x86_64/plink2
   "
zstdcat_path <- "/apps/eb/skylake/software/zstd/1.4.4-GCCcore-8.3.0/bin
   /zstdcat"
pheno  <- "breast"
prop   <- 0.2
nfolds <- 5

# use parallel computing
cl = parallel::makeCluster(detectCores())

# loop over all 5 train-test splits
for (fold in 1:nfolds) {
  kfile  <- file.path("data",
                      pheno,
                      paste0("fold", fold),
                      paste0("prop", prop, ".txt"))
  pfile  <- "data/all_vars.tsv"
  gfile  <- "data/ukb_cal_pgen/ukb_cal_all"
  covars <-
    c("Sex", "Age", paste0("PC", 1:10), paste0("PC", 1:10, "_Sex"))
  configs <- list(
    results.dir = file.path("temp", pheno,
                            paste0("fold", fold),
                            paste0("prop", prop)),
    plink2.path = plink2_path,
    zstdcat.path = zstdcat_path,
    keep = kfile
  )

  train_ids <- read.table(kfile)[, 1]
  ntrain <- length(train_ids)

  ids <- readIDsFromPsam(paste0(gfile, ".psam"))
  id_df <- data.frame(ID = ids, stringsAsFactors = F) %>%
    mutate(sort_order = 1:n())

```

```r
### pre-screening

phe <- readPheMaster(
  phenotype.file = pfile,
  psam.ids = ids,
  family = NULL,
  covariates = covars,
  phenotype = pheno,
  status = NULL,
  split.col = NULL,
  configs = configs
)
# for shell command
# vars <- dplyr::mutate(dplyr::rename(
#   data.table::fread(cmd=paste0(configs[['zstdcat.path']], ' ',
#                                   paste0(pfile, '.pvar.zst'))), 'CHROM
  '='#CHROM'),
#   VAR_ID=paste(ID, ALT, sep='_'))$VAR_ID
# n_genotypes <- length(vars)

# for an R interactive session
vars <-
  dplyr::mutate(dplyr::rename(data.table::fread(cmd = paste0(
    zstdcat_path, ' ', paste0(gfile, '.pvar.zst')
  )), 'CHROM' = '#CHROM'),
  VAR_ID = paste(ID, ALT, sep = '_'))$VAR_ID
n_genotypes <- length(vars)



pvar <- pgenlibr::NewPvar(paste0(gfile, '.pvar.zst'))
pgen <- pgenlibr::NewPgen(paste0(gfile, '.pgen'),
                          pvar = pvar,
                          sample_subset = match(phe$ID, ids))
data.y <- phe[[pheno]]
# genotype data
if (pheno == "breast" | pheno == "prostate") {
  data.X.gen2 <- as.matrix(cbind(data.y, Age = phe$Age,
                                 phe[, paste0("PC", 1:10)]))
} else{
  data.X.gen2 <-
    as.matrix(cbind(
      data.y,
      Age = phe$Age,
      Sex1 = ifelse(phe$Sex == 1, 1, 0),
      Sex0 = ifelse(phe$Sex == 0, 1, 0),
      phe[, paste0("PC", 1:10)],
      phe[, paste0("PC", 1:10, "_Sex")]
    ))
}


# fit lm on variant ~ phe + sex + age + pc + pc*sex
p = ncol(data.X.gen2)
n = nrow(data.X.gen2)
XtX.inv = solve(t(data.X.gen2) %*% data.X.gen2)
XtX.inv.Xt =  XtX.inv %*% t(data.X.gen2)

prescreening <-
  function(col,
           data.X.gen2,
           p,
           n,
           XtX.inv,
           XtX.inv.Xt) {
    col = matrix(col, ncol = 1)
    beta.hat = matrix(XtX.inv.Xt %*% col, ncol = 1)
    RSS = t(col - data.X.gen2 %*% beta.hat) %*% (col - data.X.gen2 %*
  % beta.hat)
    sigma2.hat = as.numeric(RSS / (n - p))
    beta.var = XtX.inv[1, 1] * sigma2.hat
    p.val = 2 * pt(abs(beta.hat[1] / sqrt(beta.var)),
                   df = (n - p),
                   lower.tail = F)
    return(p.val)
  }

p.vals = vector("numeric")

### parallel computing
count = 0
n_each_time = 5000

while (count +  n_each_time <= n_genotypes) {
  data.X.train <- pgenlibr::ReadList(pgen, 1:5000, meanimpute = T)
  p.vals = append(
    p.vals,
    parallel::parApply(
      cl,
      data.X.train,
      MARGIN = 2,
      prescreening,
      data.X.gen2 = data.X.gen2,
      p = p,
      n = n,
      XtX.inv = XtX.inv,
      XtX.inv.Xt = XtX.inv.Xt
    )
  )
  count <- count +  n_each_time
  cat(paste("current sample: ", count))
}

if (count +  n_each_time > n_genotypes) {
  data.X.train <-
    pgenlibr::ReadList(pgen, (count + 1):n_genotypes, meanimpute = T)
  p.vals = append(
    p.vals,
    parallel::parApply(
      cl,
      data.X.train,
      MARGIN = 2,
          prescreening,
          data.X.gen2 = data.X.gen2,
          p = p,
          n = n,
          XtX.inv = XtX.inv,
          XtX.inv.Xt = XtX.inv.Xt
        )
      )
  }

  features_selected = matrix(NA, ncol = 2, nrow = n_genotypes)
  colnames(features_selected) = c("inx", "p.val")
  features_selected[, 1] = 1:n_genotypes
  features_selected[, 2] = p.vals
  features_selected <-
    features_selected[order(features_selected[, 2]), ]
  outfile = file.path("data",
                      pheno,
                      paste0("fold", fold),
                      paste0("prop", prop, "_pvals.txt"))
  write(t(features_selected), outfile, ncolumns  = 2)
  rm(p.vals, features_selected)
}

parallel::stopCluster(cl)
```

Appendix B R code: kernel functions and kernel parameter tuning

```r
# function(1) and function(2) are stored in a script named "fun_kernel.
    R"
# Users need to source this file when implement SDR methods

###############################################################
# (1) rbfkernel function - default sigma = median distance
rbfkernel <- function(X, sigma = NULL, Y = NULL) {
  if (!is.matrix(X))
  {
    print("X must be a matrix containing samples in its rows")
    return()
  }
  if (!is.null(Y)) {
    if (!is.matrix(Y))
    {
      print("Y must be a matrix containing samples in its rows or
          NULL if it should not be used")
      return()
    }
    if (ncol(X) != ncol(Y)) {
      print("The samples in the rows of X and Y must be of same
   dimension")
      return()
    }
  }
  n <- nrow(X)
  if (is.null(Y)) {
    XtX <- tcrossprod(X)
    XX <- matrix(1, n) %*% diag(XtX)
    D <- XX - 2 * XtX + t(XX)
    if (is.null(sigma)) {
      sigma = var(D)
    }
  }
  else{
    m <- nrow(Y)
    XX <- matrix(apply(X ^ 2, 1, sum), n, m)
    YY <- matrix(apply(Y ^ 2, 1, sum), n, m, byrow = TRUE)
    XY <- tcrossprod(X, Y)
    D <- XX - 2 * XY + YY
    if (is.null(sigma)) {
      sigma = var(D)
    }
  }
  # calculate rbf-kernel matrix
  K <- exp(-D / (2 * sigma))
  return(K)
}

#(2) function for linear kernel
linearkernel <- function(X, Y = NULL){
  X = as.matrix(X)
  if(is.null(Y)){
    out <- tcrossprod(X,X)
  } else{
    out <- tcrossprod(X, Y)
  }
  return(out)
}
```

Appendix C R code: tuning Gaussian kernel parameters

```r
rbfkernel_cv <- function(X, sigma_lst, Y = NULL, lab) {
  if (!is.matrix(X))
  {
    print("X must be a matrix containing samples in its rows")
    return()
  }
  if (!is.null(Y)) {
    if (!is.matrix(Y))
    {
      print("Y must be a matrix containing samples in its rows or
          NULL if it should not be used")
      return()
    }
    if (ncol(X) != ncol(Y)) {
      print("The samples in the rows of X and Y must be of same
  dimension")
      return()
    }
  }
  lab = as.numeric(lab) + 1
  n <- nrow(X)
  if (is.null(Y)) {
    XtX <- tcrossprod(X)
    XX <- matrix(1, n) %*% diag(XtX)
    D <- XX - 2 * XtX + t(XX)
  }
  else{
    m <- nrow(Y)
    XX <- matrix(apply(X ^ 2, 1, sum), n, m)
    YY <- matrix(apply(Y ^ 2, 1, sum), n, m, byrow = TRUE)
    XY <- tcrossprod(X, Y)
    D <- XX - 2 * XY + YY
  }
  # calculate rbf-kernel matrix
  cv.res = matrix(NA, nrow = 2, ncol = length(sigma_lst))
  cv.res[1, ] = sigma_lst
  for (sigma in sigma_lst) {
    K <-  exp(-D / (2 * sigma))
    withinSum <-  0
    num = matrix(0, 1, max(lab))
    for (i in 0:max(lab)) {
      index <- lab == i
      withinSum <-  withinSum + sum(K[index, index])
      num[i] <-  sum(index)
    }
    betweenSum <-  sum(K) - withinSum
    withinNum <-  sum(num ^ 2)
    betweenNum <-  sum(num) ^ 2 - withinNum
    w <-  withinSum / withinNum
    b <-  betweenSum / betweenNum
    j <-  1 - w + b
    cv.res[2, which(sigma_lst %in% sigmalibrary("class")
```

Appendix D R code: tuning dimension reduction methods parameters

```r
library("lfda")
source("scripts/fun_lfda.R")

N = 1000
set.seed(10)
data.cv = sample(1:ntrain, 1000)
data.X.train = data.X.train[data.cv, ]
train.inx = list()
test.inx = list()
inx = split(seq_len(N), cut(seq_len(N), breaks = 5))
for (i in 1:5) {
  train.inx[[i]] = seq_len(N)[!(seq_len(N) %in% inx[[i]])]
  test.inx[[i]] = inx[[i]]
}

dim.lst = c(10, 100, 300, 500)
mis.class.error = matrix(NA, ncol = 5, nrow = length(dim.lst))
rownames(mis.class.error) = dim.lst
colnames(mis.class.error) = paste0("foldi", 1:5)

for (foldi in 1:5) {
  data.X.train.cv = data.X.train[train.inx[[foldi]], ]
  data.X.test.cv = data.X.train[test.inx[[foldi]], ]
  data.y.cv = data.y[train.inx[[foldi]]]
  data.y.test = data.y[test.inx[[foldi]]]
  for (dim in dim.lst) {
    lfda.res <- lfda(data.X.train.cv,
                     data.y.cv,
                     dim,
                     knn = 7 ,
                     metric = "orthonormalized")
    X.train.lfda <- predict(lfda.res, data.X.train.cv)
    X.test.lfda <- predict(lfda.res, data.X.test.cv)
    knn.res = knn1(X.train.lfda, X.test.lfda, data.y.cv)
    mis.rate = sum(knn.res != data.y.test) / length(knn.res)
    mis.class.error[which(dim.lst %in% dim), foldi] = mis.rate
    cat(paste(
        "the misclassification rate for cv fold ", foldi,
        "dimension ",dim,
        "is", mis.rate,"\n"))
}}
```

Appendix E R code: supervised dimension reduction

```r
###################################################################
# (1) function for supervised PCA - primal
library(rARPACK)
source("scripts/fun_kernel.R")

spca <- function(X,Y, prop = 0.95){
  if(!is.matrix(X)){
    X = as.matrix(X)
  }
  X = t(X)
  n = ncol(X)
  L = linearkernel(Y)
  H = diag(n) - 1/n * outer(rep(1,n), rep(1,n))
  Q = X %*% H %*% L %*% H %*% t(X)
  if(det(Q) == 0){
    Q = Q + diag(1e-6,nrow(X))
  }
  eig.vals <- eigen(Q, only.values = TRUE)$values
  pc_n <- length(which(cumsum(eig.vals^2)/sum(eig.vals^2) <= 0.95)) + 1
  beta <- eigs(Q, k = pc_n, which = "LM")$vectors
  return(beta)
}

###################################################################
# (2) function for supervised PCA - improved efficiency on svd
library(rARPACK)
source("scripts/fun_kernel.R")

spca <- function(X,Y, prop = 0.95){
  if(!is.matrix(X)){
    X = as.matrix(X)
  }
  X = t(X)
  n = ncol(X)
  L = linearkernel(Y)
  eig = eigen(L)
  rm(L)
  delta = t(eig$vectors %*% diag(sqrt(ifelse(eig$values >= 0, eig$
   values, 0))))
  H = diag(n) - 1/n * outer(rep(1,n), rep(1,n))
  psi = X %*% H %*% t(delta)
  svd_sing <- svd(psi, nu = 0, nv = 0)$d
  pc_n <- length(which(cumsum(svd_sing^2)/sum(svd_sing^2) <= prop)) + 1
  v = svds(psi, k = pc_n, nu = 0, nv = pc_n)$v
  if(pc_n >= 2){
    u <- psi %*% v %*% diag(1 / svd_sing[1:pc_n])
  } else{
    u <- psi %*% v %*% (1 / svd_sing[1:pc_n])
  }
  return(u)
}

# both function(1) and function(2) are supervised PCA, function(1)
# uses the primal optimisation formulation, which does not require
# matrix decomposition, but need to store an n*n matrix; function(2)
# uses a p*p kernel matrix, but extra computations required for LU
# decomposition. The users need to find a balance between space
# complexity and time complexity, then choose the one that is more
# appropriate for their dataset.

################################################################
# (3) function for kernel SPCA
source("scripts/fun_kernel.R")
library(rARPACK)

kSPCA <- function(X, Y, sigma){
  X = as.matrix(X); Y = as.matrix(Y)
  K = rbfkernel(X, sigma = sigma)
  L = linearkernel(Y)
  n = nrow(K)
  if(det(Q) == 0){
    K = K + diag(1e-6, n)
  }
  H = diag(n) - 1/n * outer(rep(1,n), rep(1,n))
  Q = K %*% L %*% H %*% K
  if(det(Q) == 0){
    Q = Q + diag(1e-6, n)
  }
  eig.vals <- eigen(Q, only.values = TRUE)$values
  pc_n <- length(which(cumsum(eig.vals)/sum(eig.vals) <= 0.95)) + 1
  beta <- eigs(Q, k = pc_n, which = "LM")$vectors
  return(beta)
}

################################################################
# (4) function for lfda
library("lfda")

lfda <- function (x,
                  y,
                  r,
                  metric = c("orthonormalized", "plain",
                             "weighted"),
                  knn = 7) {
  metric <- match.arg(metric)
  x <- t(as.matrix(x))
  y <- t(as.matrix(y))
  d <- nrow(x)
  n <- ncol(x)
  if (is.null(r))
    r <- d
  tSb <- mat.or.vec(d, d)
  tSw <- mat.or.vec(d, d)
  for (i in unique(as.vector(t(y)))) {
    Xc <- x[, y == i]
    nc <- ncol(Xc)
    Xc2 <- t(as.matrix(colSums(Xc ^ 2)))
    distance2 <- repmat(Xc2, nc, 1) + repmat(t(Xc2), 1, nc) -
      2 * t(Xc) %*% Xc
    A <- getAffinityMatrix(distance2, knn, nc)
    Xc1 <- as.matrix(rowSums(Xc))
    G <- Xc %*% (repmat(as.matrix(colSums(A)), 1, d) * t(Xc)) -
      Xc %*% A %*% t(Xc)
    tSb <- tSb + (G / n) + Xc %*% t(Xc) * (1 - nc / n) + Xc1 %*%
      (t(Xc1) / n)
    tSw <- tSw + G / nc
  }
  X1 <- as.matrix(rowSums(x))
  tSb <- tSb - X1 %*% t(X1) / n - tSw
  tSb <- (tSb + t(tSb)) / 2
  tSw <- (tSw + t(tSw)) / 2
  if (det(tSw) == 0) {
    tSw = tSw + diag(1e-6, d)
  }
  if (r == d) {
    eigTmp <- eigen(solve(tSw) %*% tSb)
  }
  else {
    eigTmp <- suppressWarnings(rARPACK::eigs(
      A = solve(tSw) %*%
        tSb,
      k = r,
      which = "LM"
    ))
  }
  eigVec <- Re(eigTmp$vectors)
  eigVal <- as.matrix(Re(eigTmp$values))
  Tr <- getMetricOfType(metric, eigVec, eigVal, d)
  Z <- t(t(Tr) %*% x)
  out <- list(T = Tr, Z = Z)
  class(out) <- "lfda"
  return(out)
}

################################################################
# (5) function for kernel lfda

library("lfda")
klfda <-
  function (k,
            y,
            r,
            metric = c("weighted", "orthonormalized",
                       "plain"),
            knn = 7,
            reg = 0.001) {
    metric <- match.arg(metric)
    y <- t(as.matrix(y))
    n <- nrow(k)
    if (is.null(r))
      r <- n
    tSb <- mat.or.vec(n, n)
    tSw <- mat.or.vec(n, n)
    for (i in unique(as.vector(t(y)))) {
      Kcc <- k[y == i, y == i]
      Kc <- k[, y == i]
      nc <- nrow(Kcc)
      Kccdiag <- diag(Kcc)
      distance2 <- repmat(Kccdiag, 1, nc) + repmat(t(Kccdiag),
                                                   nc, 1) - 2 * Kcc
      A <- getAffinityMatrix(distance2, knn, nc)
      Kc1 <- as.matrix(rowSums(Kc))
      Z <- Kc %*% (repmat(as.matrix(colSums(A)), 1, n) * t(Kc)) -
        Kc %*% A %*% t(Kc)
      tSb <- tSb + (Z / n) + Kc %*% t(Kc) * (1 - nc / n) + Kc1 %*%
        (t(Kc1) / n)
      tSw <- tSw + Z / nc
    }
    K1 <- as.matrix(rowSums(k))
    tSb <- tSb - K1 %*% t(K1) / n - tSw
    tSb <- (tSb + t(tSb)) / 2
    tSw <- (tSw + t(tSw)) / 2
    if (det(tSw) == 0) {
      tSw = tSw + diag(1e-6, n)
    }
    if (det(tSb) == 0) {
      tSb = tSb + diag(1e-6, n)
    }
    eigTmp <- suppressWarnings(rARPACK::eigs(
      A = solve(tSw +
                  reg * diag(1, nrow(tSw), ncol(tSw))) %*% tSb,
      k = r,
      which = "LM"
    ))
    eigVec <- Re(eigTmp$vectors)
    eigVal <- as.matrix(Re(eigTmp$values))
    Tr <- getMetricOfType(metric, eigVec, eigVal, n)
    Z <- t(t(Tr) %*% k)
    out <- list(T = Tr, Z = Z)
    class(out) <- "lfda"
    return(out)
  }

# (6) Supervised NMF
data.X.train <- data.X.train[,colSums(data.X.train)!=0]
data.X.test <- data.X.test[,colSums(data.X.test)!=0]
t_types <-  chty(data.X.train, data.y, 2, 3)
nmb1 <-  t_types$r1
nmb2 <-  t_types$r2
t_total <- getT(data.X.train,data.y,nmb1,nmb2)
X.train.out <- spnmf(data.X.train, t_total)$W
X.test.out <- spnmf(data.X.test, t_total)$W
```

Appendix F   R code: density ratio estimation

```r
###############################################################
# (1) function for KMM
library("quadprog")
source("scripts/fun_kernel.R")
source("scripts/fun_gaussian_cv.R")

kmm <- function(x_tr, x_te) {
  x_tr = as.matrix(x_tr)
  x_te = as.matrix(x_te)
  n_tr = nrow(x_tr)
  n_te = nrow(x_te)
  sigma = kernelWidth_median(x_tr)
  # calculate kernel
  cat("computing kernel for training data... \n")
  Ktrtr = rbfkernel(x_tr, sigma = sigma)
  Xtrtr =
    if (det(Ktrtr) == 0) {
      Ktrtr = Ktrtr + diag(1e-6, n_tr)
    }
  # compute kappa
  cat("computing kappa... \n")
  Ktrte = rbfkernel(X = x_tr, Y = x_te, sigma = sigma)
  kappa = rowSums(Ktrte) * n_tr / n_te
  # compute epsilon
  cat("computing epsilon... \n")
  eps = (sqrt(n_tr) - 1) / sqrt(n_tr)
  # constraints
  Amat <-
    cbind(rep(-1, n_tr), rep(1, n_tr), diag(1, n_tr), diag(-1, n_tr))
  bvec <- c(-n_tr * (eps + 1),
            max(-n_tr * (eps - 1), 0),
            rep(0, n_tr),
            rep(-sqrt(n_tr) + 1, n_tr))
  # computing beta
  cat("computing beta... \n")
  beta = solve.QP(Ktrtr, kappa, Amat, bvec)$solution
  beta[beta < 0] = 0
  return(beta)
}

###############################################################
# (2) functions for KLIEP
library("densratio")
compute_kernel_Gaussian <- function(x, centers, sigma) {
  apply(centers, 1, function(center) {
    apply(x, 1, function(row) {
      kernel_Gaussian(row, center, sigma)
    })
  })
}

kernel_Gaussian <- function(x, y, sigma) {
  exp(- squared_euclid_distance(x, y) / (2 * sigma * sigma))
}
```

```r

squared_euclid_distance <- function(x, y) {
  sum((x - y) ^ 2)
}

num <- nrow(X.test.out) * 0.5
kliep.obj <-
  densratio(X.test.out,
            X.train.out,
            method = "KLIEP",
            kernel_num = num)
weights <-
  compute_kernel_Gaussian(X.train.out,
                          kliep.obj$kernel_info$centers,
                          kliep.obj$kernel_info$sigma) %*% kliep.obj$
    kernel_weights

###############################################################
# (3) functions for uLSIF
num <- nrow(X.test.out) * 0.5
weights <-
  densratio(X.test.out,
            X.train.out,
            method = "uLSIF",
            kernel_num = num)$compute_density_ratio(X.train.out)
```

Appendix G R code: evaluate the model performance

```r
## evaluate UK biobank results
library(readr)
library(pROC)

sdr = "spca"
dre = "ulsif"
weights_algo = paste(sdr, dre, sep = "_")

results <- matrix(NA, ncol = 7)
results = as.data.frame(results)
colnames(results) = c("algorithm", "fold", "auc", "threshold",
                      "sensitivity", "specificity", "accuracy")
for (fold in 1:5){
  res.path = file.path(paste0("fold", fold),
                       paste0(weights_algo,"_screen0.05_res.tsv"))
  preds = read_tsv(res.path)
  auc <- auc(preds$y, preds$pred_bin)
  # partial_auc <- auc(preds$y, preds$pred_bin, partial.auc = c(0,0.1),
  #                    partial.auc.correct = TRUE)
  roc.info <- roc(preds$y, preds$pred_bin)
  roc.df <- data.frame(sensitivity= roc.info$sensitivities,
                       specifity = roc.info$specificities,
                       threshold = roc.info$thresholds)
  roc.df$youden<- roc.df$sensitivity + roc.df$specifity - 1
  threshold.vals = roc.df[which(roc.df$youden == max(roc.df$youden)),]
  threshold <- threshold.vals$threshold
  labs <- ifelse(preds$pred_bin >= threshold, TRUE, FALSE)
  accurary <- sum(labs == preds$y)/nrow(preds)

  results <- rbind(results, c(weights_algo,
                              fold,
                              round(c(auc,
                              threshold,
                              threshold.vals$sensitivity,
                              threshold.vals$specifity,
                              accurary),3)))
}
```