G-KMM: A flexible kernel mean matching optimization method for density ratio estimation involving multiple train & test datasets

Cristian Daniel Alecsa

Technical University of Cluj-Napoca, Cluj-Napoca, Romania
Romanian Institute of Science and Technology, Cluj-Napoca, Romania

Abstract. In the present paper we introduce new optimization algorithms for the task of density ratio estimation. More precisely, we consider extending the well-known KMM (kernel mean matching) method using the construction of a suitable loss function, in order to encompass more general situations involving the estimation of density ratio with respect to subsets of the training data and test data, respectively. The codes associated to our `Python` implementation can be found at `https://github.com/CDAlecsa/Generalized-KMM`.

Keywords: Kernel mean matching, quadratic optimization, density ratio estimation, loss function.

1 Introduction

In statistical data processing, the comparison of two distributions is of paramount importance. In general, the problem of assessing whether two probability distributions are equivalent or not is addressed through the so-called two-sample tests. There exists classical methods that tackle this issue, such as the *t-test* which compares the means of two Gaussian distributions with common variance, and the well-known non-parametric *Kolmogorov-Smirnov test*. Recently, in [1], Gretton et. al. introduced the *maximum mean discrepancy* (MMD) statistic which compares the similarities through a positive-defined kernel across two samples in an universal reproducing kernel Hilbert space (universal RKHS), and is commonly used for multivariate two-sample testing. In [2], Kirchler et. al. extended the MMD statistic by using a neural network trained on an auxiliary dataset which defines the so-called *deep maximum mean discrepancy* (DMMD) statistic, where the mapping from the input domain to the network's last hidden layer is utilized as the kernel used in the MMD statistic.
A different approach to the problem of two-sample testing is based upon the evaluation of a divergence between two distributions, such as the *f-divergence* which includes the case of the *Kullback-Leibler divergence* and the *Pearson divergence*, respectively. Due to the fact that the density estimation is a hard task, a practical approach to the divergence estimation is to directly approximate the density ratio function. One of the most well known method is the *kernel mean matching* (KMM) algorithm from [3] based on infinite-order moment matching, and which represents the MMD statistic in the particular case when a distribution is weighted accordingly to the density ratio model. There are several extensions of the KMM method such as the *Ensemble KMM* introduced in [4] where, instead of a single train-test split, one uses multiple non-overlapping test datasets and a single train dataset. A more generalized version was introduced in [5] where the main idea is to divide the training data due to the fact that the *Ensemble KMM* is not suitable with large training datasets. Consequently, this method which we will call it *Efficient Sampling KMM*, takes a bootstrap approach for the training data and then merges the results with

an aggregation process. A combination between the aforementioned two methodologies was done in [6] where Haque and his coauthors considered a KMM-type density ratio estimation called SKMM based on using bootstrap generation method for the training data and a partitioning of the test data. An extension of the classical *KMM* method is the neural network technique introduced [7] where the loss function is the actual MMD objective and the bandwidth of the underlying kernel is considered as a hyper-parameter. Due to the inherent flexibility of neural architectures and that the training is done on randomized batches this Deep Learning approach outperforms the classical KMM algorithms.

Different alternatives to MMD-based methods mostly rely on solving different *optimization problems*. Such examples are the *unconstrained least squares importance fitting* (uLSIF) from [8] and the *relative uLSIF* (RuLSIF) method introduced in [9], where in the latter one the density ratio model involves a mixture of densities. The uLSIF method was successfully employed in [10] for the comparison of distributions using a permutation test approach. In more detail, by utilizing a weighted Gaussian kernel model at test samples, the weights of the density ratio model are learned through a *quadratic optimization problem*, after which the *Pearson divergence* is employed in order to compare the train and test distributions. A general method encompassing the constrained variant of the uLSIF method, namely LSIF, is the Bregman formulation given in [11]. The unified method which utilizes the Bregman divergence contains as a particular case the well-known *KL importance estimation procedure* (KLIEP) from [12] for which the *objective function* is given with respect to the test samples, while the constraints depend on the training samples. When the true density ratio is approximated through a linear or kernel model, then one obtains a *convex optimization problem with constraints*. On the other hand, for the situation when the choice of the density ratio model is the *log-linear model* as in [13] then the underlying density fitting framework reduces to a *unconstrained convex optimization problem* and therefore the global solution can be obtained by iterative methods such as *gradient descent*.

As previously mentioned, the KMM method introduced in [7] uses the MMD statistic as the *loss function* for the density ratio approximation which is modeled through a neural network. A similar approach was developed recently in [14] in which a neural-type approach was developed for the RuLSIF method in the setting of change point detection. In both these works, the true density ratio is represented as a neural network and the learning of such a network depends on a loss function suitable for density ratio estimation. A different approach for the RuLSIF method is the one from [15] where the density ratio is not directly represented as a neural network but is considered as a weighted feed-forward neural model (instead of a weighted kernel model). By employing the RuLSIF approach one finds at each iteration the weights as the global optimum solution for the quadratic optimization problem. After finding the weights, the classical backpropagation algorithm is applied to the density ratio model with respect to the parameters of the feed-forward neural model. In addition to this, the aim of the method introduced in [15] is to estimate the density ratio from a few training samples, by using instances in different but related datasets (also called source datasets).

As we formerly emphasized, the framework regarding the density ratio models *Ensemble KMM* from [4] and *Efficient Sampling KMM* from [5] depends on multiple train or test datasets. On the other hand, the density fitting methodology proposed in [16] (which we shall briefly call it *MultiDistribution DRE*) is related to the idea that one has access to i.i.d. samples from multiple distributions. As a particular case, this can be perceived as having multiple test datasets and a single reference train dataset. The purpose is to estimate the density ratios between all pairs of distributions. This can be efficiently done by employing the Bregman divergence with respect to the reference density function and thus optimizing

a vector density ratio. Consequently, the optimization function can be written as a sum of multiple objective mappings, where each of them depends on a particular density ratio component. Accordingly, this approach generalizes the LSIF and KLIEP optimization algorithms to the so-called Multi-LSIF and Multi-KLIEP methods, respectively.
There are various alternatives to the aforementioned density ratio methods. The most well known approach to the previously mentioned techniques is the probabilistic density fitting method where, as described in [17], one learns a probabilistic classifier that separates the train and test samples. The methodology behind these classification algorithms is that the density ratio is approximated by the ratio of the sample sizes multiplied by the class posterior probabilities, the latter ones being obtained from the classifier's output. Furthermore, the main advantage of the probabilistic classification technique is that it is easy to implement it in a real world situation.
We end this section with the table (1), in which we describe the investigation that was done in different papers which are in connection to our theoretical and empirical results.

Articles	Methodology
Sugiyama et al. (2011) [10]	- comparison of distributions using permutation tests based on the uLSIF method
Miao et al. (2015) [4]	- *Ensemble KMM* method which utilizes multiple non-overlapping test datasets and a single train dataset
Chandra et al. (2016) [5]	- *Efficient Sampling KMM* method based upon a bootstrap approach for the training data which merges the results with an aggregation proces
Haque et al. (2016) [6]	- SKMM method which uses a using bootstrap generation method for the training data and a partitioning of the test data
Hushchyn & Ustyuzhanin (2021) [14]	- a RuLSIF type neural network model for change point detection tasks
Yu et al. (2021) [16]	- the aim is to estimate the density ratios between all pairs of distributions.
de Mathelin et al. (2022) [7]	- an extension of the classical KMM method to neural networks

Table 1: Recent research contributions

2 Motivation

Let's consider two sample sets $\mathcal{X} = \{x_i \mid x_i \in \mathbb{R}^d\}_{i=1}^n$ and $\mathcal{X}' = \{x'_j \mid x'_j \in \mathbb{R}^d\}_{j=1}^{n'}$ such that $\mathcal{X} \overset{i.i.d.}{\sim} P$ and $\mathcal{X}' \overset{i.i.d.}{\sim} P'$, where P and P' are probability distributions with densities p, p', respectively.
The MMD (maximum mean discrepancy) statistic between $\mathcal{X}$ and $\mathcal{X}'$ is defined as

$$MMD_{\phi,\psi}(\mathcal{X}, \mathcal{X}') = \|\mathbb{E}_{p'(x)}[\psi(x)] - \mathbb{E}_{p(x)}[\phi(x)]\|^2,$$

where $\phi, \psi : \mathbb{R}^d \to \mathbb{R}^p$ are two given feature maps. By defining the density ratio function $r(x) = \dfrac{p(x)}{p'(x)}$, where we assume that $p'(x) > 0$ for all x, and choosing $\psi(x) = r(x)\phi(x)$, then we obtain the loss function used in the KMM (kernel mean matching) approach from [3] with respect to an approximation $\hat{r}$ of r:

$$MMD_{\phi,\hat{r}\phi}(\mathcal{X}, \mathcal{X}') = \|\mathbb{E}_{p'(x)}[\hat{r}(x)\phi(x)] - \mathbb{E}_{p(x)}[\phi(x)]\|^2$$

By using that

$$1 = \int p(x)dx = \int r(x)p'(x)dx = \mathbb{E}_{p'(x)}[r(x)]$$

and taking into account the above objective function, one obtains the following *optimization problem with constraints* (that needs to be solved by considering an approximation of the density ratio model $\hat{r}$):

$$\begin{cases} \min_{\hat{r}} \|\mathbb{E}_{p'(x)}[\hat{r}(x)\phi(x)] - \mathbb{E}_{p(x)}[\phi(x)]\|^2 \\ \\ \text{subject to} \begin{cases} \hat{r}(x) \geq 0 \text{ for all } x \\ \mathbb{E}_{p'(x)}[\hat{r}(x)] = 1 \end{cases} \end{cases} \tag{OptPb-KMM}$$

For simplifying the formulation of (OptPb-KMM), we introduce the kernel map $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle$. Furthermore, let's consider $h \in \mathbb{R}^{n'}$ such that $h_j = \frac{n'}{n} \sum_{i=1}^{n} K(x'_j, x_i)$ for $j = \{1, \ldots, n'\}$. At the same time, define $H \in \mathbb{R}^{n' \times n'}$ such that $H_{jk} = K(x'_j, x'_k)$ for $j, k \in \{1, \ldots, n'\}$, along with $r_{\mathcal{X}'} \in \mathbb{R}^{n' \times 1}$, where $r_{\mathcal{X}'} = (r(x'_1), \ldots, r(x'_{n'}))^T$. If we define $\hat{r}(x)$ as a model approximating the true density ratio $r(x)$, and ignoring irrelevant constants with respect to $\hat{r}(x)$ then (OptPb-KMM) becomes the following *quadratic optimization problem with constraints*:

$$\begin{cases} \min_{\hat{r}_{\mathcal{X}'}} \left(\frac{1}{2} \hat{r}_{\mathcal{X}'}^T H \hat{r}_{\mathcal{X}'} - h^T \hat{r}_{\mathcal{X}'} \right) \\ \text{subject to} \begin{cases} (\hat{r}_{\mathcal{X}'})_j \geq 0 \text{ for } j \in \{1, \ldots, n'\} \\ \\ \frac{1}{n'} \sum_{j=1}^{n'} (\hat{r}_{\mathcal{X}'})_j = 1 \end{cases} \end{cases}$$

A different kind of density ratio technique is the RuLSIF method from [9] which uses a generalization of the density ratio i.e., the α-relative density ratio $r_\alpha(x) = \frac{p(x)}{q_\alpha(x)}$ for $\alpha \in [0, 1)$, where $q_\alpha(x)$ represents the α-mixture density of $p(x)$ and $p'(x)$ i.e., $q_\alpha(x) = \alpha p(x) + (1 - \alpha)p'(x)$. In the RuLSIF method, one models the true density ratio as a linear kernel method with weights $w \in \mathbb{R}^{n \times 1}$, namely $\hat{r}(x) = w^T k(x)$, where $k(x) = (K(x, x_1), \ldots, K(x, x_n))^T \in \mathbb{R}^{n \times 1}$ and $K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\tilde{\sigma}^2} \right)$ is the corresponding Gaussian kernel with the variance $\tilde{\sigma}^2$, respectively. Let's define $H \in \mathbb{R}^{n \times n}$ and $h \in \mathbb{R}^{n \times 1}$, namely $H_{l,k} = \frac{\alpha}{n} \sum_{i=1}^{n} K(x_i, x_l)K(x_i, x_k) + \frac{1 - \alpha}{n'} \sum_{j=1}^{n'} K(x'_j, x_l)K(x'_j, x_k)$ and $h_l = \frac{1}{n} \sum_{i=1}^{n} K(x_i, x_l)$, where $l, k \in \{1, \ldots, n\}$. By ignoring constants irrelevant to the weights w and using a regularization parameter λ, we obtain the following *unconstrained regularized quadratic optimization problem* corresponding to the RuLSIF loss:

$$\min_w \left(\frac{1}{2} w^T H w - h^T w + \frac{\lambda}{2} w^T w \right) \tag{OptPb-RuLSIF}$$

In order to generalize the KMM approach to multiple sample sets, we observe that the objective function belonging to (OptPb-KMM) leads to

$$\mathbb{E}_{p(x)}[\phi(x)] = \int \phi(x)p(x)dx = \int r(x)\phi(x)p'(x)dx = \mathbb{E}_{p'(x)}[r(x)\phi(x)], \tag{1}$$

therefore, under the true density ratio model $r(x)$, the loss function is minimized. We will use this simple technique for extending in a precise manner the KMM algorithm to multiple sample sets.

- The first case we investigate is when we have, for $i \in \{1, \ldots, N\}$, the sets $\mathcal{X}_i = \{x_{k,(i)} \,|\, x_{k,(i)} \in \mathbb{R}^d\}_{k=1}^{n_i}$ and $\mathcal{X}' = \{x'_j \,|\, x'_j \in \mathbb{R}^d\}_{j=1}^{n'}$ such that $\mathcal{X}_i \overset{i.i.d.}{\sim} P_i$ and $\mathcal{X}' \overset{i.i.d.}{\sim} P'$, where P_i and P' are probability distributions with densities p_i, p', respectively. If we consider $\mathcal{X}'$ to be the training dataset and $\mathcal{X}_i$ to represent the non-overlapping partitions of a given test dataset $\mathcal{X}$, then the technique proposed in *Ensemble KMM* uses the fact that $p(x \in \mathcal{X}) = \sum_{i=1}^{N} \frac{n_i}{n} p(x \,|\, x \in \mathcal{X}_i)$, where $n = \sum_{j=1}^{N} n_j$. Therefore, the probability associated to the test sample set $\mathcal{X}$ can be written as a mixture between non-overlapping partitions with the weights given by the ratio between the size of the partition and the total size of the test dataset. This is in accordance with the definition from the formulation of *Ensemble KMM* of the density ratio corresponding to $\mathcal{X}$ and $\mathcal{X}'$ which is given by a mixture of density ratios between $\mathcal{X}_i$ and $\mathcal{X}'$, respectively. By dropping the notation of conditional probability density concerning the partitions of $\mathcal{X}$, let's consider the general case when $r(x) = \sum_{i=1}^{N} \omega_i r_i(x)$ where $r_i(x) = \frac{p_i(x)}{p'(x)}$ for $i \in \{1, \ldots, N\}$ and where the weights satisfy $\sum_{i=1}^{N} \omega_i = 1$ with $\omega_i \in [0, 1]$ for $i \in \{1, \ldots, N\}$. Therefore

$$r(x) = \sum_{i=1}^{N} \omega_i r_i(x) = \sum_{i=1}^{N} \omega_i \frac{p_i(x)}{p'(x)} = \frac{\sum_{i=1}^{N} \omega_i p_i(x)}{p'(x)} = \frac{p(x)}{p'(x)},$$

where $p(x)$ represents the mixture density defined as $p(x) = \sum_{i=1}^{N} \omega_i p_i(x)$. Inspired by the identity (1) from the case of KMM, we infer the loss function which is minimized under the true density ratio $r(x)$:

$$\begin{aligned} \mathbb{E}_{p'(x)}[r(x)\phi(x)] &= \int r(x)\phi(x)p'(x)dx = \int \left(\sum_{i=1}^{N} \omega_i p_i(x) \right) \phi(x)dx \\ &= \sum_{i=1}^{N} \omega_i \int p_i(x)\phi(x)dx = \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)], \end{aligned}$$

thus we consider the following loss function which needs to be solved with respect to the approximate model $\hat{r}$ of the density ratio r:

$$\mathcal{L} = \left\| \mathbb{E}_{p'(x)}[\hat{r}(x)\phi(x)] - \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\|^2 \tag{2}$$

An alternative of the above computations is to consider the approach of *MultiDistribution DRE* where we define as before, for $i \in \{1, \ldots, N\}$, $r_i(x) = \frac{p_i(x)}{p'(x)}$. In this case we compute for every $i \in \{1, \ldots, N\}$ the following:

$$\mathbb{E}_{p'(x)}[r_i(x)\phi(x)] = \int r_i(x)\phi(x)p'(x)dx = \int \phi(x)p_i(x)dx = \mathbb{E}_{p_i(x)}[\phi(x)],$$

hence we can define the i^{th} loss mapping with respect to the approximation $\hat{r}_i$ of r_i:

$$\mathcal{L}_i = \|\mathbb{E}_{p'(x)}[\hat{r}_i(x)\phi(x)] - \mathbb{E}_{p_i(x)}[\phi(x)]\|^2,$$

therefore one can propose the mixture loss function $\mathcal{L}$ where the weights w_i represent the contribution of each particular loss function $\mathcal{L}_i$, namely

$$\mathcal{L} = \sum_{i=1}^{N} \omega_i \mathcal{L}_i = \sum_{i=1}^{N} \omega_i \|\mathbb{E}_{p'(x)}[\hat{r}_i(x)\phi(x)] - \mathbb{E}_{p_i(x)}[\phi(x)]\|^2. \tag{3}$$

It is worth pointing out that the loss function (3) resembles the approach of *Ensemble KMM* structure, where one solves simultaneously N optimization problems, with the condition that the objective function of the i^{th} problem is related to the approximation $\hat{r}_i(x)$ of its associated density ratio model r_i.

• Now we turn our attention to our second case which we shall analyze it, where from a practical point of view one has multiple non-overlapping training datasets and a single test dataset. In order to do this we consider, for $i \in \{1, \ldots, N'\}$, the sets $\mathcal{X} = \{x_k \,|\, x_k \in \mathbb{R}^d\}_{k=1}^{n}$ and $\mathcal{X}'_i = \{x'_{j,(i)} \,|\, x'_{j,(i)} \in \mathbb{R}^d\}_{j=1}^{n'_i}$ such that $\mathcal{X} \overset{i.i.d.}{\sim} P$ and $\mathcal{X}'_i \overset{i.i.d.}{\sim} P'_i$, where P and P'_i are probability distributions with densities p, p'_i, respectively. In a similar fashion with the previous case, let's consider $r_i(x) = \dfrac{p(x)}{p'_i(x)}$ and the weights $\tilde{\omega}_i \in [0, 1]$, for each $i \in \{1, \ldots, N'\}$ such that $\sum\limits_{i=1}^{N'} \tilde{\omega}_i = 1$. Then, it follows that

$$\begin{aligned}
\mathbb{E}_{p(x)}[\phi(x)] &= \int \phi(x)p(x)dx = \int \phi(x)p(x)\left(\sum_{i=1}^{N'} \tilde{\omega}_i\right) dx \\
&= \int \phi(x)\left(\tilde{\omega}_1 p(x) + \ldots + \tilde{\omega}_{N'} p(x)\right) dx \\
&= \int \phi(x)\left(\tilde{\omega}_1 r_1(x)p'_1(x) + \ldots + \tilde{\omega}_{N'} r_{N'}(x)p'_{N'}(x)\right) dx \\
&= \int \phi(x)\left(\sum_{i=1}^{N'} \tilde{\omega}_i r_i(x)p'_i(x)\right) dx = \sum_{i=1}^{N'} \tilde{\omega}_i \int \phi(x)r_i(x)p'_i(x)dx \\
&= \sum_{i=1}^{N'} \tilde{\omega}_i \mathbb{E}_{p'_i(x)}[r_i(x)\phi(x)],
\end{aligned}$$

hence it is natural to propose the following loss function:

$$\mathcal{L} = \left\| \sum_{i=1}^{N'} \tilde{\omega}_i \mathbb{E}_{p'_i(x)}[\hat{r}_i(x)\phi(x)] - \mathbb{E}_{p(x)}[\phi(x)] \right\|^2 \tag{4}$$

For the previous case we shall present an alternative method in order to infer a suitable loss function. We proceed by considering, for each $i \in \{1, \ldots, N'\}$ the weights $\omega_i \in [0, 1]$ that satisfy $\sum\limits_{i=1}^{N'} \omega_i = 1$. Along with these we define the mixture probability density $p'(x) =$

$\sum\limits_{i=1}^{N'} \omega_i p'_i(x)$ and the corresponding true density ratio $r(x) = \dfrac{p(x)}{p'(x)}$. Then, it follows that

$$\begin{aligned}\mathbb{E}_{p(x)}[\phi(x)] &= \int \phi(x)p(x)dx = \int \phi(x)r(x)p'(x)dx = \int \phi(x)r(x)\left(\sum_{i=1}^{N'} \omega_i p'_i(x)\right)dx \\ &= \sum_{i=1}^{N'} \omega_i \int \phi(x)r(x)p'_i(x)dx = \sum_{i=1}^{N'} \omega_i \mathbb{E}_{p'_i(x)}[r(x)\phi(x)],\end{aligned}$$

which defines the following loss function:

$$\mathcal{L} = \left\| \sum_{i=1}^{N'} \omega_i \mathbb{E}_{p'_i(x)}[\hat{r}(x)\phi(x)] - \mathbb{E}_{p(x)}[\phi(x)] \right\|^2 \tag{5}$$

In what follows we will show an equivalence between (4) and (5) under certain assumptions on the weights with respect to the true density ratio. By using that $p(x) = r_i(x)p'_i(x)$, it follows for each $i \in \{1, \ldots, N'\}$ that

$$r(x) = \frac{r_i(x)p'_i(x)}{\sum\limits_{j=1}^{N'} \omega_j p'_j(x)}.$$

So, for every $k \in \{1, \ldots, N'\}$ we have that

$$\begin{aligned}\sum_{i=1}^{N'} \omega_i \mathbb{E}_{p'_i(x)}[r(x)\phi(x)] &= \sum_{i=1}^{N'} \omega_i \int r(x)\phi(x)p'_i(x)dx = \int \phi(x)r(x)\left(\sum_{i=1}^{N'} \omega_i p'_i(x)\right)dx \\ &= \int \phi(x)r_k(x)p'_k(x)dx = \mathbb{E}_{p'_k(x)}[r_k(x)\phi(x)].\end{aligned}$$

Therefore, by summing over k, it follows that

$$\sum_{i=1}^{N'} \omega_i \mathbb{E}_{p'_i(x)}[r(x)\phi(x)] = \sum_{k=1}^{N'} \left(\frac{1}{N'}\right) \mathbb{E}_{p'_k(x)}[r_k(x)\phi(x)],$$

for which we can select the uniformly distributed weights $\tilde{\omega}_i = \dfrac{1}{N'}$ for every $i \in \{1, \ldots, N'\}$.

• We end the present section with a brief remark about a particular case regarding the inference of (5). By using the form of the true density ratio, we obtain the following computations:

$$r(x) = \frac{p(x)}{p'(x)} = \frac{p(x)}{\sum\limits_{i=1}^{N'} \omega_i p'_i(x)} = \frac{p(x)}{\sum\limits_{i=1}^{N'-1} \omega_i p'_i(x) + \omega_{N'} p'_{N'}(x)}.$$

Let's consider the situation when $p'_{N'}(x) = p(x)$ and $\omega_{N'} = \alpha \in [0, 1)$. Then, it follows that

$$r(x) = \frac{p(x)}{\sum\limits_{i=1}^{N'-1} \omega_i p'_i(x) + \alpha p(x)},$$

where $\sum_{i=1}^{N'-1} \omega_i + \alpha = \sum_{i=1}^{N'} \omega_i = 1$, so $\sum_{i=1}^{N'-1} \omega_i = 1 - \alpha$. This can be considered as a generalization of the relative density ratio due to the fact that, when $N' = 2$, one has access to the test dataset $\mathcal{X}$ and the train dataset $\mathcal{X}'_1$, respectively. Therefore, one obtains the α-relative density ratio $r_\alpha(x) = \frac{p(x)}{\alpha p(x) + (1-\alpha)p'(x)}$. Finally, we highlight that, for $i \in \{1, \ldots, N'\}$, the sample sets $\mathcal{X}'_i = \{x'_{j,(i)} \,|\, x'_{j,(i)} \in \mathbb{R}^d\}_{j=1}^{n'_i}$ corresponding to $p'_i(x)$ form a partition of non-overlapping sets. Hence, the situation when $p'_{N'}(x) = p(x)$ is equivalent to the fact that $\mathcal{X}$ is non-overlapping with any other training subsets $\mathcal{X}'_i$ for $i \in \{1, \ldots, N'-1\}$. In order to avoid this limitation, when dealing with the particular case of the generalized relative density ratio, we propose to formally use the same formulas as above despite the fact that the non-overlapping condition does not hold in general between train and test datasets, respectively.

3 Structure of the paper

The aim of the previous section (2) was to present in a step-by-step manner the main stimulus behind our `Generalized KMM` method. For this, we investigated an approach for devising a *quadratic optimization problem with constraints* based on the situation of multiple non-overlapping training datasets, along with the case of multiple non-overlapping test datasets. Our algorithmic framework is different than the methodologies from *Ensemble KMM* introduced in [4] and *Efficient Sampling KMM* from [5] since our technique is constructed using a loss-function approach. In section (4) we will actually construct our optimizer. By introducing an extended density ratio function using mixtures of probability densities, our technique is based upon the construction of a non-negative loss mapping which attains its minimum value of 0 under the true density ratio. The empirical version is obtained when one uses an approximate linear kernel model using the points of the whole test dataset, by utilizing some constraints suitable to numerical implementations. In section (5) we present some numerical simulations developed with a custom implementation made in `Python` regarding the comparison of probability densities under the learned density ratio weights, along with importance reweighting examples. Finally, in the last section (6), we discuss about the advantages of our generalized method along with the underlying limitations.

4 Proposed optimization method

In what follows we consider our general KMM-type optimization technique based upon the computations made in the previous section. In the usual case of (OptPb-KMM) and its extensions, one considers optimizing the MMD-based loss function involving the density ratio model only at the training points. Inspired by the techniques utilized in (OptPb-RuLSIF) we propose to use a linear kernel model for the density ratio in order to obtain an optimization problem with respect to the underlying parameters of the model. It is worth pointing out that this methodology is similar to the one proposed in [7] where a neural network was used for the density ratio model. Furthermore, the training of the neural network model is made at each epoch with respect to non-overlapping shuffled data batches thus the setting from [7] is similar to ours (see also the alternative of the bootstrap aggregation technique from [5]). But, the main difference is that, in our case, the train and test partitions are given at the beginning of the algorithm and are not randomly created at each iteration.

Let's consider N' training sets $\mathcal{X}'_l = \{x'_{j,(l)} \,|\, x'_{j,(l)} \in \mathbb{R}^d\}_{j=1}^{n'_l}$ such that $\mathcal{X}'_l \overset{i.i.d.}{\sim} P'_l$, where P'_l is the probability distribution with the underlying density p'_l for every $l \in \{1, \ldots, N'\}$. At the same time, let's suppose that we have N test sets $\mathcal{X}_i = \{x_{k,(i)} \,|\, x_{k,(i)} \in \mathbb{R}^d\}_{k=1}^{n_i}$ such that $\mathcal{X}_i \overset{i.i.d.}{\sim} P_i$ where P_i is the probability distribution corresponding to the density p_i for each $i \in \{1, \ldots, N\}$. In what follows we will consider the test and train mixtures of probability densities $p(x) = \sum\limits_{i=1}^{N} \omega_i p_i(x)$ and $p'(x) = \sum\limits_{j=1}^{N'} \gamma_j p'_j(x)$, where the weights satisfy $\sum\limits_{i=1}^{N} \omega_i = \sum\limits_{j=1}^{N'} \gamma_j = 1$, with $\omega_i \in [0, 1]$ for every $i \in \{1, \ldots, N\}$ and $\gamma_j \in [0, 1]$ for each $j \in \{1, \ldots, N'\}$. The general density ratio between the train and test samples is defined as

$$r(x) = \frac{p(x)}{p'(x)} = \frac{\sum\limits_{i=1}^{N} \omega_i p_i(x)}{\sum\limits_{j=1}^{N'} \gamma_j p'_j(x)}.$$

One observes that

$$\begin{aligned} \mathbb{E}_{p(x)}[\phi(x)] &= \int \phi(x) p(x) dx = \int \phi(x) \left(\sum_{i=1}^{N} \omega_i p_i(x) \right) dx \\ &= \sum_{i=1}^{N} \int \phi(x) \omega_i p_i(x) dx = \sum_{i=1}^{N} \omega_i \int \phi(x) p_i(x) dx = \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)]. \end{aligned} \tag{6}$$

At the same time we have that

$$\begin{aligned} \mathbb{E}_{p'(x)}[r(x)\phi(x)] &= \int r(x)\phi(x)p'(x)dx = \int r(x)\phi(x) \left(\sum_{j=1}^{N'} \gamma_j p'_j(x) \right) dx \\ &= \sum_{j=1}^{N'} \gamma_j \int r(x)\phi(x)p'_j(x)dx = \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[r(x)\phi(x)]. \end{aligned} \tag{7}$$

Also

$$\mathbb{E}_{p'(x)}[r(x)\phi(x)] = \int r(x)\phi(x)p'(x)dx = \int \phi(x)p(x)dx = \mathbb{E}_{p(x)}[\phi(x)] \tag{8}$$

By combining (6), (7) and (8) we arrive at

$$\sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[r(x)\phi(x)] = \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)],$$

and taking into account that

$$\mathbb{E}_{p'(x)}[r(x)] = \int r(x)p'(x)dx = \int p(x)dx = 1,$$

along with

$$\mathbb{E}_{p'(x)}[r(x)] = \int r(x)p'(x)dx = \int r(x) \left(\sum_{j=1}^{N'} \gamma_j p'_j(x) \right) dx = \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)} \left[r(x) \right],$$

we therefore consider the following optimization problem:

$$\begin{cases} \min_{\hat{r}} \left\| \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)] - \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\|^2 \\ \text{subject to} \begin{cases} \hat{r}(x) \geq 0 \text{ for all } x \\ \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}\left[\hat{r}(x)\right] = 1 \end{cases} \end{cases} \quad \text{(OptPb-G-KMM)}$$

where $\hat{r}(x)$ represents a density ratio model which approximates the true density ratio $r(x)$. In the following we shall show that the optimization problem presented in (OptPb-G-KMM) can be written as a *quadratic optimization problem with constraints.* Then, the underlying loss function can be written as

$$\begin{aligned} \mathcal{L} := \left\| \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)] - \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\|^2 &= \left\| \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\|^2 \\ &+ \left\langle \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)], \sum_{k=1}^{N'} \gamma_k \mathbb{E}_{p'_k(x)}[\hat{r}(x)\phi(x)] \right\rangle \\ &- 2 \left\langle \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)], \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\rangle. \end{aligned}$$

Ignoring constants irrelevant with respect to $\hat{r}(x)$, the objective function defined above can be taken as

$$\mathcal{L} = \left\langle \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)], \sum_{k=1}^{N'} \gamma_k \mathbb{E}_{p'_k(x)}[\hat{r}(x)\phi(x)] \right\rangle - 2 \left\langle \sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)], \sum_{i=1}^{N} \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\rangle,$$

hence

$$\mathcal{L} = \sum_{j=1}^{N'} \sum_{k=1}^{N'} \left\langle \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)], \gamma_k \mathbb{E}_{p'_k(x)}[\hat{r}(x)\phi(x)] \right\rangle - 2 \sum_{j=1}^{N'} \sum_{i=1}^{N} \left\langle \gamma_j \mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)], \omega_i \mathbb{E}_{p_i(x)}[\phi(x)] \right\rangle.$$

By consider employing empirical averages, we therefore obtain that $\mathbb{E}_{p_i(x)}[\phi(x)] \approx \frac{1}{n_i} \sum_{l=1}^{n_i} \phi(x_{l,(i)})$ and $\mathbb{E}_{p'_j(x)}[\hat{r}(x)\phi(x)] \approx \frac{1}{n'_j} \sum_{t=1}^{n'_j} \hat{r}\left(x'_{t,(j)}\right) \phi\left(x'_{t,(j)}\right)$, which implies that the empirical loss function $\widehat{\mathcal{L}}$ which approximates $\mathcal{L}$ takes the form

$$\begin{aligned} \widehat{\mathcal{L}} = &\sum_{j=1}^{N'} \sum_{k=1}^{N'} \left\langle \gamma_j \left(\frac{1}{n'_j} \sum_{t=1}^{n'_j} \hat{r}\left(x'_{t,(j)}\right) \phi\left(x'_{t,(j)}\right) \right), \gamma_k \left(\frac{1}{n'_k} \sum_{s=1}^{n'_k} \hat{r}\left(x'_{s,(k)}\right) \phi\left(x'_{s,(k)}\right) \right) \right\rangle \\ &- 2 \sum_{j=1}^{N'} \sum_{i=1}^{N} \left\langle \gamma_j \left(\frac{1}{n'_j} \sum_{t=1}^{n'_j} \hat{r}\left(x'_{t,(j)}\right) \phi\left(x'_{t,(j)}\right) \right), \omega_i \left(\frac{1}{n_i} \sum_{l=1}^{n_i} \phi(x_{l,(i)}) \right) \right\rangle. \end{aligned}$$

Taking $n'_{max} := \max_{j\in\{1,\dots,N'\}}\{n'_j\}$ and multiplying $\widehat{\mathcal{L}}$ with $\frac{1}{2}\left(n'_{max}\right)^2$, we simplify the previous identity as follows:

$$\widehat{\mathcal{L}} = \sum_{j=1}^{N'}\sum_{k=1}^{N'}\left(\frac{\left(n'_{max}\right)^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}\left\langle\sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)\phi\left(x'_{t,(j)}\right),\sum_{s=1}^{n'_k}\hat{r}\left(x'_{s,(k)}\right)\phi\left(x'_{s,(k)}\right)\right\rangle\right)$$
$$-\sum_{j=1}^{N'}\sum_{i=1}^{N}\left(\frac{\left(n'_{max}\right)^2}{n_i n'_j}\gamma_j\omega_i\left\langle\sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)\phi\left(x'_{t,(j)}\right),\sum_{l=1}^{n_i}\phi\left(x_{l,(i)}\right)\right\rangle\right).$$

By utilizing the linearity of the inner product, we obtain

$$\widehat{\mathcal{L}} = \sum_{j=1}^{N'}\sum_{k=1}^{N'}\left(\frac{\left(n'_{max}\right)^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}\sum_{t=1}^{n'_j}\sum_{s=1}^{n'_k}\hat{r}\left(x'_{t,(j)}\right)\left\langle\phi\left(x'_{t,(j)}\right),\phi\left(x'_{s,(k)}\right)\right\rangle\hat{r}\left(x'_{s,(k)}\right)\right)$$
$$-\sum_{j=1}^{N'}\sum_{i=1}^{N}\left(\frac{\left(n'_{max}\right)^2}{n_i n'_j}\gamma_j\omega_i\sum_{t=1}^{n'_j}\sum_{l=1}^{n_i}\hat{r}\left(x'_{t,(j)}\right)\left\langle\phi\left(x'_{t,(j)}\right),\phi\left(x_{l,(i)}\right)\right\rangle\right).$$

From the definition of the kernel mapping as an inner product of the feature maps i.e., $K(x,y) = \langle\phi(x),\phi(y)\rangle$, it follows that

$$\widehat{\mathcal{L}} = \sum_{j=1}^{N'}\sum_{k=1}^{N'}\left(\frac{\left(n'_{max}\right)^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}\sum_{t=1}^{n'_j}\sum_{s=1}^{n'_k}\hat{r}\left(x'_{t,(j)}\right)K\left(x'_{t,(j)},x'_{s,(k)}\right)\hat{r}\left(x'_{s,(k)}\right)\right)$$
$$-\sum_{j=1}^{N'}\sum_{i=1}^{N}\left(\frac{\left(n'_{max}\right)^2}{n_i n'_j}\gamma_j\omega_i\sum_{t=1}^{n'_j}\sum_{l=1}^{n_i}\hat{r}\left(x'_{t,(j)}\right)K\left(x'_{t,(j)},x_{l,(i)}\right)\right). \quad (9)$$

In order to simplify the previous computations we consider the following notations:

$$\hat{r}_{\mathcal{X}'_j} := \left(\hat{r}\left(x'_{1,(j)}\right),\dots,\hat{r}\left(x'_{n'_j,(j)}\right)\right)^T \in \mathbb{R}^{n'_j\times 1},\ j\in\{1,\dots,N'\}$$
$$\hat{r}_{\mathcal{X}_i} := \left(\hat{r}\left(x_{1,(i)}\right),\dots,\hat{r}\left(x_{n_i,(i)}\right)\right)^T \in \mathbb{R}^{n_i\times 1},\ i\in\{1,\dots,N\}.$$

At the same time we define for $i\in\{1,\dots,N\}$ and $j\in\{1,\dots,N'\}$ the vector $h^{[i,j]}\in\mathbb{R}^{n'_j\times 1}$, such that

$$h_t^{[i,j]} = \frac{\left(n'_{max}\right)^2}{n_i n'_j}\gamma_j\omega_i\sum_{l=1}^{n_i}K\left(x'_{t,(j)},x_{l,(i)}\right) \text{ for each } t\in\{1,\dots,n'_j\}.$$

Also, for every $j,k\in\{1,\dots,N'\}$ we define the matrix $H^{[j,k]}\in\mathbb{R}^{n'_j\times n'_k}$, such that

$$H_{t,s}^{[j,k]} = \frac{\left(n'_{max}\right)^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}K\left(x'_{t,(j)},x'_{s,(k)}\right) \text{ for each } t\in\{1,\dots,n'_j\} \text{ and } s\in\{1,\dots,n'_k\}.$$

By using the above notations we obtain the following computations:

$$\frac{\left(n'_{max}\right)^2}{n_i n'_j}\gamma_j\omega_i\sum_{t=1}^{n'_j}\sum_{l=1}^{n_i}\hat{r}\left(x'_{t,(j)}\right)K\left(x'_{t,(j)},x_{l,(i)}\right) = \sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)\left(\frac{\left(n'_{max}\right)^2}{n_i n'_j}\gamma_j\omega_i\sum_{l=1}^{n_i}K\left(x'_{t,(j)},x_{l,(i)}\right)\right)$$
$$= \sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)h_t^{[i,j]}$$
$$= \left(h^{[i,j]}\right)^T\hat{r}_{\mathcal{X}'_j}. \quad (10)$$

On the other hand, by denoting

$$C^{[j,k]} := \frac{(n'_{max})^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}\sum_{t=1}^{n'_j}\sum_{s=1}^{n'_k}\hat{r}\left(x'_{t,(j)}\right)K\left(x'_{t,(j)},x'_{s,(k)}\right)\hat{r}\left(x'_{s,(k)}\right), \tag{11}$$

we find that

$$\begin{aligned}
C^{[j,k]} &= \sum_{t=1}^{n'_j}\sum_{s=1}^{n'_k}\hat{r}\left(x'_{t,(j)}\right)\left(\frac{(n'_{max})^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}K\left(x'_{t,(j)},x'_{s,(k)}\right)\right)\hat{r}\left(x'_{s,(k)}\right)\\
&= \sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)\sum_{s=1}^{n'_k}\left(\frac{(n'_{max})^2}{n'_j n'_k}\frac{\gamma_j\gamma_k}{2}K\left(x'_{t,(j)},x'_{s,(k)}\right)\hat{r}\left(x'_{s,(k)}\right)\right)\\
&= \sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)\sum_{s=1}^{n'_k}H^{[j,k]}_{t,s}\hat{r}\left(x'_{s,(k)}\right)\\
&= \sum_{t=1}^{n'_j}\hat{r}\left(x'_{t,(j)}\right)\left(H^{[j,k]}\hat{r}_{\mathcal{X}'_k}\right)_t\\
&= \left(\hat{r}_{\mathcal{X}'_j}\right)^T H^{[j,k]}\left(\hat{r}_{\mathcal{X}'_k}\right).
\end{aligned} \tag{12}$$

By merging (9), (10), (11) and (12), we find a simpler formulation of the empirical loss, namely

$$\widehat{\mathcal{L}} = \sum_{j=1}^{N'}\sum_{k=1}^{N'}\left(\left(\hat{r}_{\mathcal{X}'_j}\right)^T H^{[j,k]}\left(\hat{r}_{\mathcal{X}'_k}\right)\right) - \sum_{j=1}^{N'}\sum_{i=1}^{N}\left(\left(h^{[i,j]}\right)^T\left(\hat{r}_{\mathcal{X}'_j}\right)\right). \tag{13}$$

In order to make our method more similar to RuLSIF we consider modeling the true density ratio as a linear model i.e., $\hat{r}(x) = \langle\theta,\xi(x)\rangle$ where $\theta = (\theta_1,\ldots,\theta_b) \in \mathbb{R}^b$ and $\xi : \mathbb{R}^d \to \mathbb{R}^b$ such that $\xi(x) = (\xi_1(x),\ldots,\xi_b(x))$ for each $x \in \mathbb{R}^d$. For $j \in \{1,\ldots,N'\}$ we define the matrix $A^{[j]} \in \mathbb{R}^{n'_j\times b}$ such that

$$A^{[j]} = \begin{pmatrix}\xi_1(x'_{1,(j)}) & \ldots & \xi_b(x'_{1,(j)})\\ \vdots & \vdots & \vdots\\ \xi_1(x'_{n'_j,(j)}) & \ldots & \xi_b(x'_{n'_j,(j)})\end{pmatrix} \tag{14}$$

Some easy computations reveal that

$$A^{[j]}\theta = \begin{pmatrix}\xi^T(x'_{1,(j)})\theta\\ \cdots\\ \xi^T(x'_{n'_j,(j)})\theta\end{pmatrix} = \hat{r}_{\mathcal{X}'_j} \in \mathbb{R}^{n'_j\times 1} \tag{15}$$

Therefore, (15) implies that

$$\left(\hat{r}_{\mathcal{X}'_j}\right)^T H^{[j,k]}\left(\hat{r}_{\mathcal{X}'_k}\right) = (A^{[j]}\theta)^T H^{[j,k]}(A^{[k]}\theta) = \theta^T\left((A^{[j]})^T H^{[j,k]}A^{[k]}\right)\theta. \tag{16}$$

Using again (15), it follows that

$$\left(h^{[i,j]}\right)^T\left(\hat{r}_{\mathcal{X}'_j}\right) = \left(h^{[i,j]}\right)^T A^{[j]}\theta = \left(\left(h^{[i,j]}\right)^T A^{[j]}\right)\theta. \tag{17}$$

Consequently, (13), (16) and (17) imply that

$$\widehat{\mathcal{L}} = \theta^T \left(\sum_{j=1}^{N'} \sum_{k=1}^{N'} (A^{[j]})^T H^{[j,k]} A^{[k]} \right) \theta - \left(\sum_{j=1}^{N'} \sum_{i=1}^{N} (h^{[i,j]})^T A^{[j]} \right) \theta. \tag{18}$$

For the particular case of kernel methods we can choose $\xi(x)$ with the same technique as in the case of RuLSIF, namely we will use the test dataset defined as the reunion of the non-overlapping test sample datasets: $\mathcal{X} = \bigcup_{i=1}^{N} \mathcal{X}_i$. Therefore, we will select $\xi_k(x) = K(x, x_k)$ where $x_k \in \mathcal{X}$ for each $k \in \{1, \dots, b\}$, thus $b = \sum_{i=1}^{N} n_i$.
In what follows we will select K as a kernel endowed with non-negative values, such as the *RBF kernel* or the *Laplacian kernel.* In [3] the value of $\hat{r}(x)$ was bounded (only with respect to the training samples) in the interval $[0, B]$, where $B > 0$. In our case, in order to simplify this condition, we consider $\hat{r}(x) = \langle \theta, \xi(x) \rangle \geq 0$ and using that K takes non-negative values, we shall impose that $\theta_k \in [0, B]$ for every $k \in \{1, \dots, b\}$, where $B > 0$ is a constant chosen up to our choice. On the other hand, we define $\varXi = (\varXi_1, \dots, \varXi_b) \in \mathbb{R}^b$ as

$$\varXi := \sum_{j=1}^{N'} \left(\frac{\gamma_j}{n'_j} \right) \sum_{k=1}^{n'_j} \xi(x_{k,(j)}).$$

The constraint $\sum_{j=1}^{N'} \gamma_j \mathbb{E}_{p'_j(x)} [\hat{r}(x)] = 1$ from (OptPb-G-KMM) leads to its empirical counterpart, namely

$$1 \approx \sum_{j=1}^{N'} \gamma_j \left(\frac{1}{n'_j} \sum_{k=1}^{n'_j} \hat{r}(x_{k,(j)}) \right) = \sum_{j=1}^{N'} \left(\frac{\gamma_j}{n'_j} \right) \sum_{k=1}^{n'_j} \hat{r}(x_{k,(j)}) = \sum_{j=1}^{N'} \left(\frac{\gamma_j}{n'_j} \right) \sum_{k=1}^{n'_j} \langle \theta, \xi(x_{k,(j)}) \rangle = \langle \theta, \varXi \rangle.$$

Using the above identity $\langle \theta, \varXi \rangle = 1$, similar to the numerical description of KMM from [3], we consider $\varepsilon > 0$ such that $|\langle \theta, \varXi \rangle - 1| \leq \varepsilon$, hence $\sum_{k=1}^{b} \theta_k \varXi_k \leq \varepsilon + 1$ and $- \sum_{k=1}^{b} \theta_k \varXi_k \leq \varepsilon - 1$, respectively.
By combining (18) with the constraints presented above, we finally obtain our `Generalized KMM` method represented by the following *empirical generalized KMM optimization problem*:

$$\begin{cases} \min_{\theta} \left[\theta^T \left(\sum_{j=1}^{N'} \sum_{k=1}^{N'} (A^{[j]})^T H^{[j,k]} A^{[k]} \right) \theta - \left(\sum_{j=1}^{N'} \sum_{i=1}^{N} (h^{[i,j]})^T A^{[j]} \right) \theta \right] \\ \\ \text{subject to} \begin{cases} \theta_k \in [0, B] \text{ for } k \in \{1, \dots, b\} \\ + \sum_{k=1}^{b} \theta_k \varXi_k \leq \varepsilon + 1 \\ - \sum_{k=1}^{b} \theta_k \varXi_k \leq \varepsilon - 1. \end{cases} \end{cases} \tag{OptPb-Empirical-G-KMM}$$

5 Results & experiments

In this section we present some numerical simulations based on our implementation of the `Generalized KMM` optimizer concerning certain experiments made on some synthetic datasets. We highlight that our codes hinge on `SKLearn` [18] and the `CVXPY` package [19] and [20], respectively. At the same time, all the details about our implementation and the corresponding experiments can be found in our `GitHub` link presented in the *Abstract* of the present paper. It is of utmost importance to mention that our parameter σ which will appear in the experiments from the following sequel and in the underlying implementation, is denoted as γ in the `SKLearn` implementation (and when $\gamma \approx \tilde{\sigma}^{-2}$ then the aforementioned parameter is associated with the variance $\tilde{\sigma}^2$ of the kernel K).

Our first experiment is related to an application of the `Generalized KMM` described through the optimization problem (OptPb-Empirical-G-KMM), along with the classical KMM method, respectively. The implementation for the underlying KMM algorithm is inspired by the codes belonging to [21] and [22], respectively. For this experiment we have considered 4 clusters (two of them belonging to the training dataset and the other two representing the test samples) consisting of different number of samples, i.e. 200, 1000, 1000 and 300, respectively. The clusters were generated using the function `make_blobs` from `SKLearn` with the following standard deviation values: 0.6, 0.6, 0.9 and 0.6, respectively. For both numerical methods, the parameter B was set to 1000 while the values of the parameter σ were chosen as 1.0, 3.5, 2.0 and $None$. We mention that $None$ is equivalent to the default value used in the definition of the kernels from `SKLearn`. In the case of the `Generalized KMM` algorithm each vector and matrix that is defined through a kernel has the same default value defined as `1/n_features`, but we have chosen to write it as $None$ since this value is already shown in the plots related to the KMM method (due to the fact that we use the same datasets for both methods hence we have the same number of features). From the results depicted in figure (1) we observe that the classical KMM method has weight values smaller than those of the `Generalized KMM` algorithm. Our optimization method gives weights a higher value near the boundary of the two training clusters, while KMM emphasize also the samples belonging near the center of the training data. One also observers that by increasing the σ parameter the values of the weights also increase. On the other hand, the particular case when σ is attributed the value of $None$ (depicted in the bottom right plot) shows that KMM leads to fewer weights with high values in contrast with the `Generalized KMM` method.

Our next experiments are related to the comparison of various distributions by employing the cases of multiple train and test datasets. In figures (2), (3) and (4), for each simulation that we have made, a custom selection of the train and test distributions is represented in the corresponding left plot while in the associated right plot we have considered visualizing the predictions of a basic `SGDRegressor` with and without the sample weights generated by the `Generalized KMM` algorithm. In order to inspect more closely the comparison of the effect of the density ratio weights, the title of each right plot shows the MAE with and without the sample weights. For all the plots containing the regression results, the target is generated using a *sinc* function at which we added a noise term following a normal distribution. Also, the B term involved in (OptPb-Empirical-G-KMM) was set to 1000.

The first simulation we have done is related to the case of multiple train datasets and it is shown in figure (2). For this, we have generated 3 random normal train datasets with sizes 200, 150 and 100, with the means -0.5, 0.5 and 1.5, and with the standard deviation equal to 0.1. At the same time, the test dataset is composed of only 30 samples generated using a normal distribution with mean 1.0 and standard deviation 0.4. In the left plots of the first row we have chosen a lower of value of σ, namely 0.1 which implies that the weighted train distribution is uniformly distributed with respect to the train partitions. This can be easily visualized in the corresponding right plot where the weighted and un-

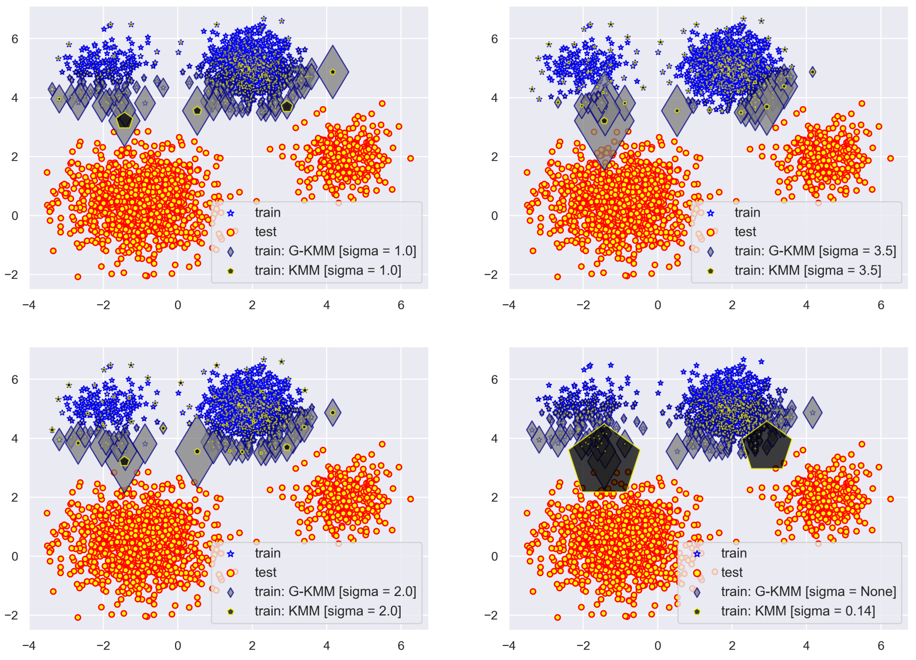

Fig. 1: KMM vs. `Generalized KMM`

weighted predictions behave similarly. In the right pair of plots from the first row of figure (2) we took σ equal to 1.0 but we have chosen the case of the α-relative density ratio with $\alpha = 0.25$, and where the γ_j weights of the training subsets were considered as 0.5, 0.2 and 0.05, respectively. The effect of the α-mixture density can be seen through the visualization of the distorsion of the weighted train distribution towards the skewed test dataset. For the last case, which is represented in the second row, we have set σ to 100, α to 0.5 and the γ_j values as 0.05, 0.2 and 0.25, respectively. These choices shows a similar effect as in the previous case regarding the γ_j mixture values.

Our next simulations presented in figure (3) correspond to the case of multiple test datasets. We have generated a single train dataset of size 300 from a normal distribution with mean 1.0 and standard deviation 0.25 for the results depicted in the first row, while for the second row the train was generated using a normal distribution with mean 0.5 and standard deviation 0.25, respectively. On the other hand, the test datasets, both of size 100, were generated from normal distributions with means -0.5 and 1.5, with the corresponding standard deviations equal to 0.15. Furthermore, for all our simulations depicted in figure (3) the parameter σ was chosen as 100. The left plots belonging to the first row shows that the weighted train distribution becomes closer to the test subset which overlaps the train dataset. On the other hand, the right plots from the first row shows the effect of the α mixture coefficient which was set to 0.75 along with the γ_1 coefficient of the single train dataset which was eventually chosen as 0.25. Here, we see that the mixture coefficient emphasize much more the test dataset which is closer to the training dataset, and it eventually leads to a worse approximation of the `SGDRegressor`. Finally, the simulation made in the plots from the second rows are based upon the same choice of

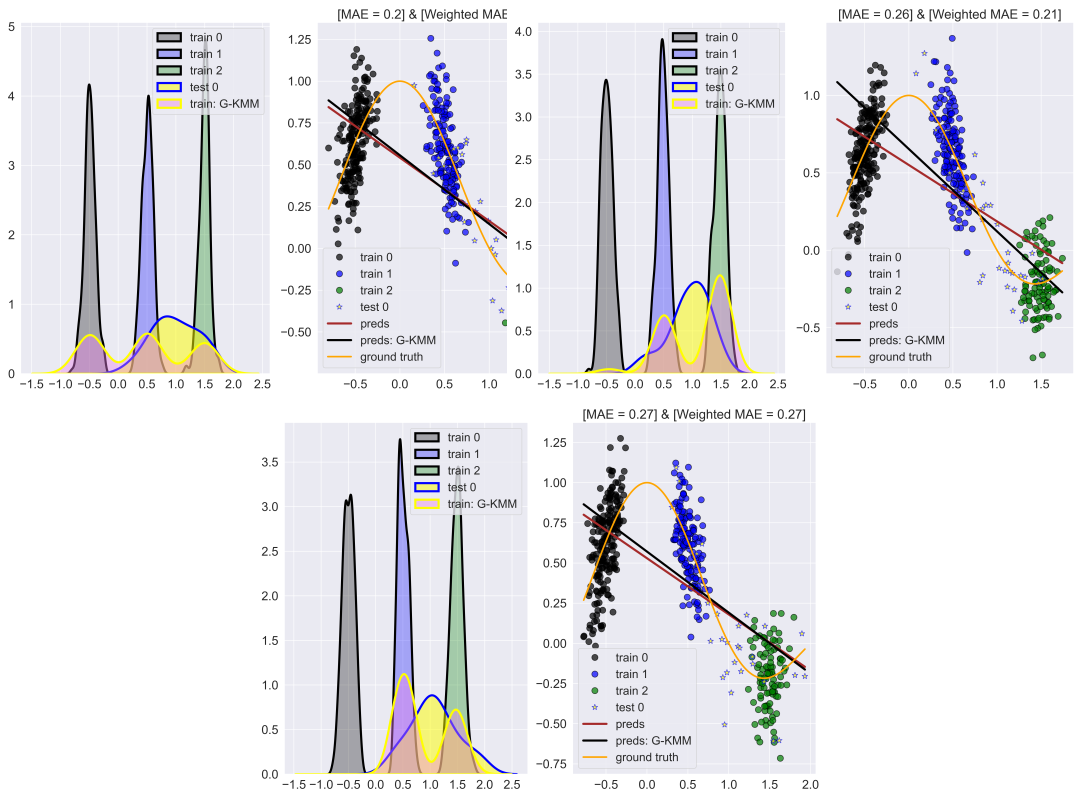

Fig. 2: Multiple train datasets

the coefficents as in the previously described simulation, namely α is 0.75, γ_1 was set to 0.25 and σ to 100, respectively. The main difference is that the training dataset is shifted to the left hence it is located between the two test datasets. One can observe that the α-mixture density ratio approach is suitable for this regression problem setting, by leading to a uniform-like distribution of the weighted train dataset.

The last experiment that we will present involves multiple train and also multiple test datasets and it is shown in figure (4). As before, we generate the train and test datasets using a random normal distribution. In the corresponding simulations we created 3 train datasets of sizes 200, 150 and 100, along with 2 test datasets of sizes equal to 100. The training subsets were generated from random normal distributions with means -0.5, 0.5 and 1.5, and standard deviation 0.1. On the other hand, the two test datasets were generated using random normal distribution with means -0.5 and 1.5, with a standard deviation equal to 0.15. In the plots further to the left from the first row of (4), we have chosen the value 0.1 for σ. Similar to the left-most plots from the first row of figure (2), the low value for σ implies a weighted train distribution with 3 peaks uniformly distributed, along with a weighted MAE equal to the MAE obtained from the unweighted predictions. On the other hand, in the right-most plots from the first row of (4) the value for σ was increased to 10. This leads to only 2 peaks, uniformly distributed and centered at the test distributions. Consequently, the MAE metric decreases if one uses the weighted predictions of the `SGDRegressor`.

Now, let's turn our attention to the simulations made in the second row of figure (4). For the results shown in the further to the left plots we have chosen σ equal to 100, and the ω_i test weights 0.85 and 0.15, respectively. Along with these we have considered an α-mixture density approach, where α was not defined directly, i.e. at first the γ_j weights of the train

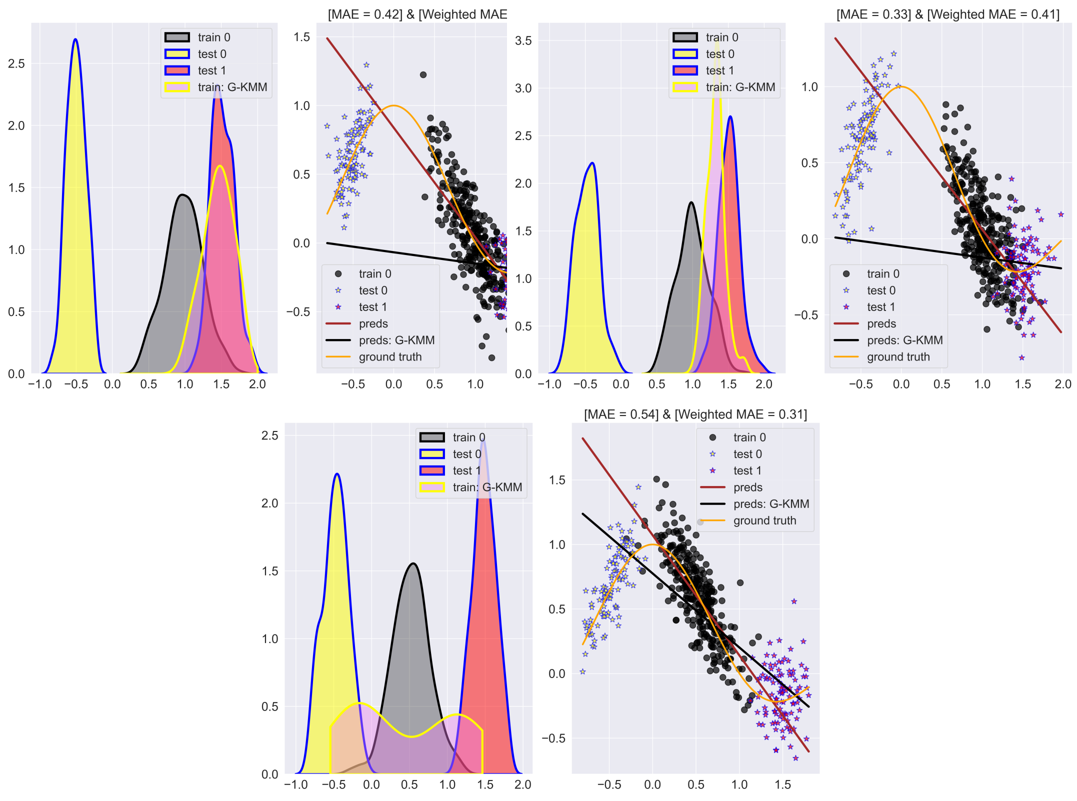

Fig. 3: Multiple test datasets

subsets were constructed using the ratio of each train subset size and the size of the total training dataset, and then α was determined such that α and the sum of γ_j add to the value of 1 (for this see the basic example belonging to the ending part of section (2)). In this case one observes that the mixture density technique modifies the distribution of the peaks of the weighted training data. Furthermore, the value of the weights ω_i related to the test datasets shows that the higher the ω_i weight is then the higher is the peak pointing to the corresponding test dataset. Similar to the case of multiple test datasets which were depicted in the last row of figure (3), the α-mixture density approach is crucial in the learning process of the optimal density ratio weights.
For the right-most plots shown in the second row of figure (4) we considered also σ equal to 100. But, the weights corresponding to the training subsets, namely γ_j were chosen this time as 0.25, 0.2 and 0.05 while the weights ω_i for the test subsets were selected with the values 0.15 and 0.85, respectively. As explained before, since α and the sum of all the γ_j coefficients must sum up to 1, we set α to the value of 0.5. Due to the fact that the weight of the second test subset is higher than the coefficient corresponding to the first test subset, the peak of the weighted train dataset is higher in the location of the second test subset.

Finally, we conclude the present section by highlighting that the experiment made in figure (1) reveals a qualitative comparison between the classical KMM algorithm and our `Generalized KMM` density ratio optimization method. At the same time, the simulations presented in figures (2), (3) and (4) shows the versatility of our method through the choices of the coefficients, especially for the case of the α-mixture density ratio.

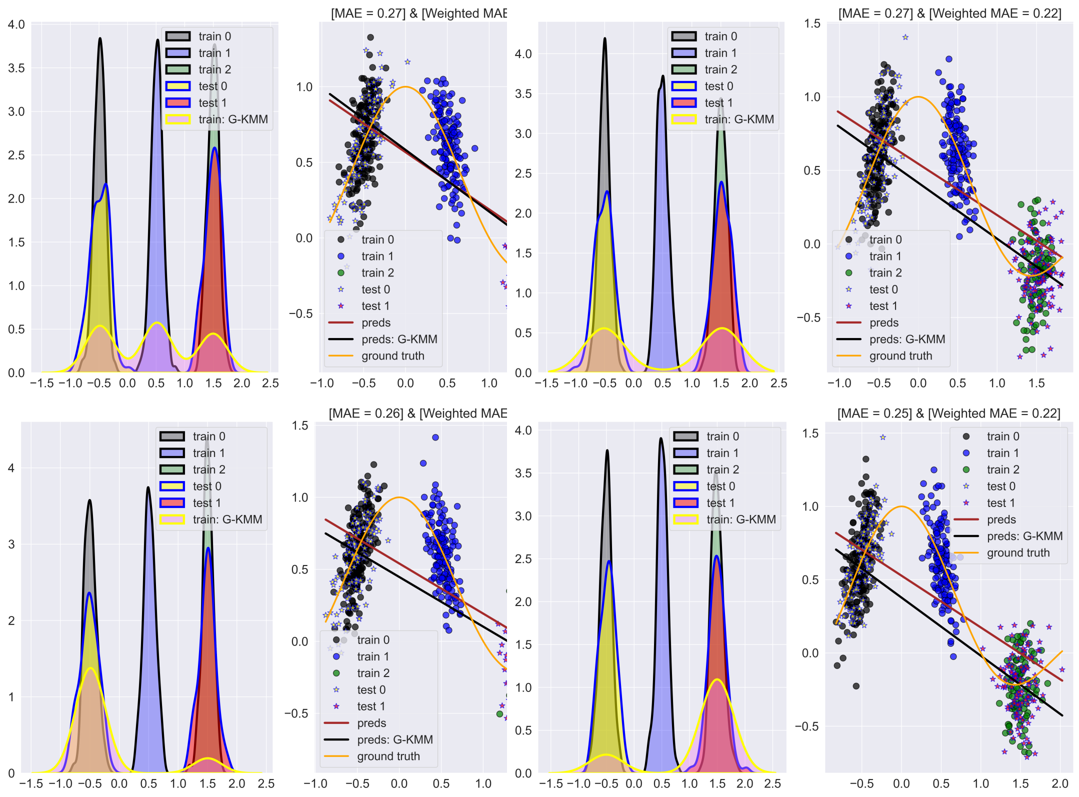

Fig. 4: Multiple train & test datasets

6 Conclusions & perspectives

In this final section we present a brief overview of our `Generalized KMM` algorithm given through the *quadratic optimization problem with constraints* (OptPb-Empirical-G-KMM) along with the underlying limitations and the possible extensions for future research.

6.1 Novelty

In the present study, our main contribution is the introduction of a new type of density ratio estimation technique entitled `Generalized KMM`, which is an extension of the classical KMM algorithm. From both a theoretical and a practical point of view our proposed method is completely novel from the following perspectives:

– The *Ensemble KMM* method from [4] is based on the idea of dividing the test dataset into multiple non-overlapping test sets, while *Efficient Sampling KMM* introduced in [5] is associated with the idea of a bootstrap aggregation approach for the training data. In contrast, our method is not developed using heuristic arguments, but it relies on the construction of a suitable loss function which attains its minimum value in the theoretical situation when one uses the true density ratio.
– In [3], the classical KMM algorithm uses directly the density ratio model with respect to the training samples. But, in our work we employed the approach used in RuLSIF where the density ratio is approximated with a linear kernel model, where the underlying kernel depends on the test points. Hence we minimize a loss function with respect to some weights belonging to a lower-dimensional space, where the dimension is given by the total number of test samples.

- Although we have constructed our minimization problem in connection to the cases consisting of non-overlapping train/test datasets, our approach contains as a particular case also a generalized version of the α-relative density ratio, which is unique from the point of view of KMM-type methods. On the other hand, it is worth emphasizing that the theoretical construction of our method was done using the idea of non-overlapping sets, while the case of the α-relative density ratio is devised only through a formal and mimetic approach.

6.2 Research limitations

Our method has not only advantages but it is also constrained by our inherent methodology as shown below:

- Despite the fact that the `Generalized KMM` is rigorously developed, one loses the parallelization property of the *Ensemble KMM* and *Efficient Sampling KMM*, respectively.
- Similar to the classical KMM algorithm, our method has the same dependence on the hyper-parameters ε, B and σ, respectively.

6.3 Recommendations for future research

For future research, we propose the following methods to enlarge our KMM-type framework:

- In a similar manner with [7] we can extend our algorithm to the case of neural networks. More precisely, one can utilize the objective function given in (OptPb-Empirical-G-KMM) along with the constraints which can be applied directly into the forward propagation process. Consequently, we can make our method faster using randomized batch learning, hence we may utilize the KMM-type algorithm for adjusting the probability densities associated to data augmentation sample sets.
- Similar to the classical KMM algorithm, our method is suitable for the estimation of the density ratio weights. In general, only a few training samples contribute to the reweighting process, due to the fact that the density ratio estimation and the regression/classification learning are separated. In order to alleviate this, we can proceed as in [23] by simultaneously training our `Generalized KMM` density ratio model and the underlying weighted loss function, in the framework of supervised learning.

References

1. A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, The Journal of Machine Learning Research, vol. 13, no. 1, 2012, pp. 723-773.
2. M. Kirchler, S. Khorasani, M. Kloft, C. Lippert, Two-sample testing using deep learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 1387-1398.
3. A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, B. Schölkopf, Covariate shift by kernel mean matching, Dataset shift in machine learning, vol. 3, no. 4, 2009, pp. 5.
4. Y-Q. Miao, A.K. Farahat, M.S. Kamel, Ensemble kernel mean matching, in: 2015 IEEE International Conference on Data Mining, IEEE, 2015, pp. 330-338.
5. S. Chandra, A. Haque, L. Khan, C. Aggarwal, Efficient sampling-based kernel mean matching, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 811-816.
6. A. Haque, Z. Wang, S. Chandra, Y. Gao, L. Khan, C. Aggarwal, Sampling-based distributed kernel mean matching using Spark, in: 2016 IEEE International Conference on Big Data (Big Data), IEEE, 2016, pp. 462-471.
7. A. de Mathelin, F. Deheeger, M. Mougeot, N. Vayatis, Fast and Accurate Importance Weighting for Correcting Sample Bias, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Cham: Springer International Publishing, 2022, pp. 659-674.

8. T. Kanamori, S. Hido, M. Sugiyama, A least-squares approach to direct importance estimation, The Journal of Machine Learning Research, vol. 10, 2009, pp. 1391-1445.
9. M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, M. Sugiyama, Relative density-ratio estimation for robust distribution comparison, Neural computation, vol. 25, no. 5, 2013, pp. 1324-1370.
10. M. Sugiyama, T. Suzuki, Y. Itoh, T. Kanamori, M. Kimura, Least-squares two-sample test, Neural networks, vol. 24, no. 7, 2011 pp. 735-751.
11. M. Sugiyama, T. Suzuki, T. Kanamori, Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation, Annals of the Institute of Statistical Mathematics, vol. 64, 2012, pp. 1009-1044.
12. M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. Von Bünau, M. Kawanabe, Direct importance estimation for covariate shift adaptation, Annals of the Institute of Statistical Mathematics, vol. 60, 2008, pp. 699-746.
13. Y. Tsuboi, H. Kashima, S. Hido, S. Bickel, M. Sugiyama, Direct density ratio estimation for large-scale covariate shift adaptation, Journal of Information Processing, vol. 17, 2009, pp. 138-155.
14. M. Hushchyn, A. Ustyuzhanin, Generalization of change-point detection in time series data based on direct density ratio estimation, Journal of Computational Science, vol. 53, 2021, pp. 101385.
15. A. Kumagai, T. Iwata, Y. Fujiwara, Meta-learning for relative density-ratio estimation, Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 30426-30438.
16. L. Yu, Y. Jin, S. Ermon, A unified framework for multi-distribution density ratio estimation, arXiv preprint arXiv:2112.03440, 2021.
17. M. Sugiyama, T. Suzuki, T. Kanamori, Density ratio estimation in machine learning, Cambridge University Press, 2012.
18. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, the Journal of machine Learning research, vol. 12, 2011, pp. 2825-2830.
19. A. Agrawal, R. Verschueren, S. Diamond, S. Boyd, A rewriting system for convex optimization problems, Journal of Control and Decision, vol. 5, no. 1, 2018, pp. 42-60.
20. S. Diamond, S. Boyd, CVXPY: A Python-embedded modeling language for convex optimization, The Journal of Machine Learning Research, vol. 17, no. 1, 2016, pp. 2909-2913.
21. A. de Mathelin, F. Deheeger, G. Richard, M. Mougeot, N. Vayatis, ADAPT: Awesome domain adaptation python toolbox, arXiv preprint arXiv:2107.03049, 2021.
22. T. Fang, N. Lu, G. Niu, M. Sugiyama, Rethinking importance weighting for deep learning under distribution shift, Advances in neural information processing systems, vol. 33, 2020, pp. 11996-12007.
23. S. Chen, X. Yang, Tailoring density ratio weight for covariate shift adaptation, Neurocomputing, vol. 333, 2019, pp. 135-144.

Authors

C.D. Alecsa received his PhD in Mathematics from Babeş-Bolyai University and has a broad experience in both academia and industry. Currently, he is a researcher on two national grants: at the Technical University of Cluj-Napoca (on Optimization) and at the Romanian Institute of Science and Technology (on Machine Learning), respectively. His research interests include Pattern Recognition, Machine Learning, Statistics, and Applied Mathematics.