Efficient Sampling-based Kernel Mean Matching

Swarup Chandra, Ahsanul Haque and Latifur Khan
Department of Computer Science
The University of Texas at Dallas, Richardson, TX, USA.
Email: (src093020, axh129430, lkhan)@utdallas.edu

Charu Aggarwal
IBM T. J. Watson Research Center
Yorktown, NY, USA.
Email: charu@us.ibm.com

***Abstract*—Many real-world applications exhibit scenarios where distributions represented by training and test data are not similar, but related by a covariate shift, i.e., having equal class conditional distribution with unequal covariate distribution. Traditional data mining techniques suffer to learn a good predictive model in the presence of covariate shift. Recent studies have proposed approaches to address this challenge by weighing training instances based on density ratio between test and training data distributions. Kernel Mean Matching (KMM) is a well known method for estimating density ratio, but has time complexity cubic in the size of training data. Therefore, KMM is not suitable in real-world applications, especially in cases where the predictive model needs to be updated periodically with large training data. We address this challenge by taking fixed-size samples from training and test data, performing independent computations on these samples, and combining the results to obtain overall density ratio estimates. Our empirical evaluation demonstrates a large gain in execution time, while also achieving competitive accuracy on numerous benchmark datasets.**

***Index Terms*—Covariate Shift; Density Ratio; Bootstrap Aggregation**

I. Introduction

Probabilistic models used in data mining applications, such as regression and classification [1], traditionally assume that the training and test data exhibit similar data distribution. However, in situations where this assumption is not valid, naive application of these models may result in dramatic failures. For example, consider a sentiment classification problem of online product reviews. If a new product (called A) is launched, classification of product reviews may help understand customer sentiments. While the true sentiments of these reviews are unknown, reviews and sentiments of another (similar) product (called B) may be readily available. Clearly, the two sets of reviews represent two different data distributions. Applying a sentiment classification model naively may result in an undesirably low accuracy. Therefore, an adaptation mechanism is needed to leverage available information to perform prediction, i.e., use reviews of product B to learn a classifier for classifying reviews of product A. In general, this problem is known as *transfer learning* [2].

When both training and test datasets occur from the same domain, difference in data distribution between the two datasets may be an outcome of a biased training data generation method. Reasons for such bias could be limited availability of labeled data, high cost of manual data annotation, etc. This transfer learning scenario is called *sample selection bias* [3], commonly occurring in surveys and sociology studies [4]. Typically, a correction mechanism is employed where the training data distribution is made similar to that of the test data distribution, either via instance weighting or feature weighting.

Particularly in the case of data classification, it is assumed that the training and test probability distributions are related such that $p_{tr}(y|\mathbf{x}) = p_{te}(y|\mathbf{x})$ and $p_{tr}(\mathbf{x}) \neq p_{te}(\mathbf{x})$. Here, $p_{tr}(\cdot)$ and $p_{te}(\cdot)$ represents the probability distribution of training and test data respectively. This scenario is called *covariate shift* [3]. A density ratio estimate, denoted by $\beta(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{tr}(\mathbf{x})}$, is computed for each training instance $\langle \mathbf{x} \rangle$. This is used to adapt the given training data while learning an appropriate model suitable for label prediction on the test data. Each estimate represents the so-called *importance weight* [5] (or instance weight) for the corresponding training data instance.

Since both $p_{tr}(\cdot)$ and $p_{te}(\cdot)$ are unknown, recent studies have proposed methods to estimate $\beta(\cdot)$ directly, i.e., without explicitly evaluating the two distributions. *Kernel Mean Matching* (KMM) [3] is one such method that computes training instance weights by minimizing mean discrepancy between the training and test data distributions in a Reproducing Kernel Hilbert Space (RKHS) [6]. This method, however, suffers from scalability when large datasets are used. Particularly, its time complexity is cubic in the training dataset size, whereas linear in the test dataset size. In practice, this often becomes a bottleneck when employed in data mining operations such as data stream classification [7], where a classifier needs to be trained periodically. Moreover, computations in KMM require the whole training and test dataset to be in the memory. In situations where the dataset is distributed across multiple systems, one cannot directly employ KMM to perform density ratio estimations for the complete dataset.

In this paper, we address the challenge of scalability in the Kernel Mean Matching procedure by appealing to the method of *m-out-of-n bootstrap* aggregating (a.k.a. m/n bagging) over the training data only. The main idea is to generate multiple fixed-size bootstrap samples of size m from the training data of size n (where $m < n$), and estimate corresponding instance weights for each sample independently using KMM. These sample-based instance weights are aggregated for each training instance to form its final density ratio estimate. We denote this method by VFKMM. Furthermore, while VFKMM requires the complete test dataset to be in memory, we propose a method that partitions the test dataset into smaller subsets, and applies VFKMM on each of them. We denote this method by

EVFKMM. The results of these are finally combined to obtain the complete list of density ratio estimates. Importantly, our method decentralizes the KMM method, and can be readily applied to large datasets in a distributed (or parallel) fashion. We empirically evaluate both the proposed approaches on benchmark datasets to illustrate their effectiveness.

II. Preliminaries

A. Notations

In general, we use a bold letter to indicate an array, and a capital-bold letter to indicate a set of arrays. An element belonging to a training set is indicated by a subscript tr, while that of a test set is indicated by a subscript te. Since a set may contain multiple elements, each element is indexed by a superscript integer. Also, an estimated value is denoted with a hat.

B. Covariate Shift

In the case of data classification - the focus of this paper - inequality in probability distribution of training and test datasets can be represented in the form of joint probability distribution, given by $p_{tr}(\mathbf{x}, y) \neq p_{te}(\mathbf{x}, y)$. If the class conditional distributions are assumed to be equal, i.e., $p_{tr}(y|\mathbf{x}) = p_{te}(y|\mathbf{x})$, inequality in joint distribution can be attributed to covariate distributions only, i.e., $p_{tr}(\mathbf{x}) \neq p_{te}(\mathbf{x})$. This situation is known as *covariate shift* [8]. In general, the relation between $p_{tr}(\mathbf{x})$ and $p_{te}(\mathbf{x})$ is accounted by computing an importance weight $\beta(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{tr}(\mathbf{x})}$ for each training instance $\mathbf{x}$. Recent studies have focused on computing $\beta(\mathbf{x})$ without explicitly estimating $p_{te}(\mathbf{x})$ and $p_{tr}(\mathbf{x})$. These include Kernel Mean Matching (KMM) [3], Kullback-Leibler Importance Estimation Procedure (KLIEP) [9], and unconstrained Least Square Importance Fitting (uLSIF) [10]. While providing elegant solutions to estimate importance weights, these methods do not scale well on large datasets.

C. Kernel Mean Matching

The idea in Kernel Mean Matching is to minimize the mean distance between weighted training data distribution $\beta(\mathbf{x})p_{tr}(\mathbf{x})$ and corresponding test data distribution $p_{te}(\mathbf{x})$ in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{F}$ with feature map $\phi : \mathcal{D} \rightarrow \mathcal{F}$. Mean distance is measured by computing the *Maximum Mean Discrepancy* (MMD), given by $\left\| E_{\mathbf{x}\sim p_{tr}(\mathbf{x})}[\beta(\mathbf{x})\phi(\mathbf{x})] - E_{\mathbf{x}\sim p_{te}(\mathbf{x})}[\phi(\mathbf{x})] \right\|$, where $\|\cdot\|$ is the l_2 norm, and $\mathbf{x} \in \mathbf{X} \subseteq \mathcal{D}$ is a data instance in a dataset $\mathbf{X}$. Here, it is assumed that p_{te} is absolutely continuous with respect to p_{tr}, i.e. $p_{te}(\mathbf{x}) = 0$ whenever $p_{tr}(\mathbf{x}) = 0$. Additionally, the RKHS kernel h is assumed to be universal in $\mathcal{D}$. It has been shown that under these conditions, minimizing MMD converges to $p_{te}(\mathbf{x}) = \beta(\mathbf{x})p_{tr}(\mathbf{x})$ [11].

*1) Centralized KMM (*CENKMM*):* In particular, minimizing MMD to obtain optimal importance weights is equivalent to minimizing the corresponding quadratic program that approximates the population expectation with an empirical expectation. The empirical approximation of MMD to obtain the desired $\hat{\beta}(\mathbf{x})$ is given by the quadratic program,

$$\hat{\boldsymbol{\beta}} \approx \underset{\beta}{\text{minimize}} \frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} - \boldsymbol{\kappa}^T\boldsymbol{\beta} \tag{1}$$

subject to $\beta(\mathbf{x}^{(i)}) \in [0, B], \forall i \in \{1 \dots n_{tr}\}$, and $\left|\sum_{i=1}^{n_{tr}} \beta(\mathbf{x}^{(i)}) - n_{tr}\right| \leq n_{tr}\epsilon$. Here n_{tr} and n_{te} are sizes of the training and test datasets respectively, and $\hat{\beta}(\mathbf{x}) \in \hat{\boldsymbol{\beta}}$. $\mathbf{K}$ and $\boldsymbol{\kappa}$ are matrices of a RKHS kernel $h(\cdot)$ with $K^{(ij)} = h(\mathbf{x}_{tr}^{(i)}, \mathbf{x}_{tr}^{(j)}) \in \mathbf{K}$, and $\kappa^{(i)} = \frac{n_{tr}}{n_{te}} \sum_{j=1}^{n_{te}} h(\mathbf{x}_{tr}^{(i)}, \mathbf{x}_{te}^{(j)}) \in \boldsymbol{\kappa}$. $B > 0$ is an upper bound on the solution search space, and ϵ is the normalization error.

*2) Ensemble KMM (*ENSKMM*):* Recently, Miao et al. [12] proposed a technique called *ensemble KMM* (denoted by ENSKMM) to address the problem of scalability in *cenKMM* when the complete test dataset cannot be stored in the memory. They construct an ensemble of estimators by dividing only the test data instances into k components, while assuming that the complete training dataset fit in the memory. For each test component $\mathbf{C}$, the weight estimates $\hat{\boldsymbol{\beta}}_c$ is computed by solving Equation 1, using the complete set of n_{tr} training instances. These component-wise instance weights are combined to form $\hat{\boldsymbol{\beta}} = \frac{1}{k}\sum_{c=1}^{k} \hat{\boldsymbol{\beta}}_c$. While the study demonstrates improvement in accuracy and execution time, computational efficiently is still limited by requiring the complete training dataset in the memory. As mentioned in [12], the time complexity of KMM is $\mathcal{O}(n_{tr}^3 + n_{tr}^2 d + n_{tr}n_{te}d)$. Clearly, this method becomes computationally expensive with increase in n_{tr}, even when it is reasonably smaller than n_{te}.

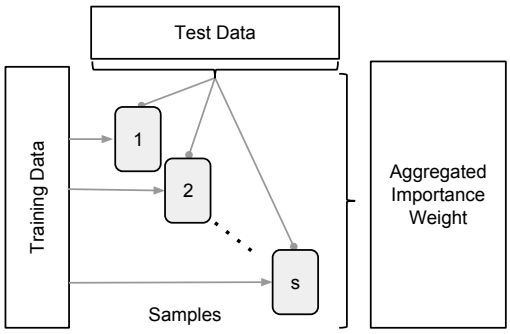


Fig. 1: Illustration of the *VFKMM* process with s samples from training data. KMM is applied over each training sample with the complete test dataset.

III. Sampling-based KMM

The challenge of scalability while estimating density ratio can be addressed naively by splitting the training data into smaller subsets and applying KMM over each subset independently. We refer to this naive method by Ensemble of TRaining data KMM or ENSTRKMM. Here, sampling over training data is performed without replacement, where each sample represents an independent subset (or partition). KMM is applied over each sample, using the complete test dataset. Union of density ratio estimates from all the samples provide

Algorithm 1: VFKMM : KMM by Bootstrap Aggregation

Data: Covariates $\mathbf{X}_{tr}$ and $\mathbf{X}_{te}$
Input: Sample Size: m, Tolerance: η, Parameters: θ
Result: $\hat{\boldsymbol{\beta}}$
begin
  $s \leftarrow \frac{\ln \eta}{m \ln\left(1-\frac{1}{n_{tr}}\right)}$ /* Compute # samples */
  $\hat{\boldsymbol{\beta}} \leftarrow zeros$
  for $i \in \{1, s\}$ **do**
    $\mathbf{X}_{tr}^{*(i)} \leftarrow generateSample(\mathbf{X}_{tr}, m)$
    $\hat{\boldsymbol{\beta}}^{*(i)} \leftarrow KMM(\mathbf{X}_{tr}^{*(i)}, \mathbf{X}_{te}, \theta)$ /*Equation 1*/
    $\hat{\boldsymbol{\beta}} \leftarrow aggregate(\hat{\boldsymbol{\beta}}^{*(i)})$
  return $normalize(\hat{\boldsymbol{\beta}})$

weight for each training data instance. However, such a method may not perform well since a small subset of training data instances (chosen uniformly at random) may exhibit glaringly different distribution compared to the original training data distribution. This can adversely affect KMM output [11].

In the following sections, we present and analyze a more efficient approach to address the above scalability problem.

A. Very Fast KMM

Given an i.i.d. set of training covariates, denoted by $\mathbf{X}_{tr}$, and an i.i.d. set of test covariates, denoted by $\mathbf{X}_{te}$, such that the size of $\mathbf{X}_{tr}$ is sufficiently large, the problem is to efficiently estimate instance weight $\hat{\beta}(\mathbf{x})$ for each $\mathbf{x} \in \mathbf{X}_{tr}$ using the Kernel Mean Matching method. Estimation of $\beta(\mathbf{x}) \in \boldsymbol{\beta}$ is sensitive to the training data distribution, i.e., the estimates may vary depending on the size and choice of instances used as the training data. Therefore, we appeal to the *m-out-of-n* bootstrap sampling (or m/n bootstrap) method, where $m < n_{tr}$. We fix the value of m to achieve scalability in KMM's computational time, and consider $n = n_{tr}$.

An overview of our proposed approach to achieve scalability in Kernel Mean Matching is illustrated in Figure 1. We refer to it by Very Fast KMM or VFKMM. A number of fixed-size bootstrap samples are generated from the training dataset $\mathbf{X}_{tr}$. Data instances in each sample are chosen with replacement from $\mathbf{X}_{tr}$, where each sample is denoted by $\mathbf{X}_{tr}^{*}$. By considering the complete set of test data instances, $\hat{\beta}^{*}(\mathbf{x})$ for each instance $\mathbf{x} \in \mathbf{X}_{tr}^{*}$ is computed using Equation 1. Since each instance can be associated with multiple samples, it would have an equivalent number of estimated instance weights. Final weight for each instance is calculated by taking the average of all weights calculated for this instance.

Algorithm 1 details the process of using m/n bootstrap method over training data to compute instance weights using KMM. Ideally, we desire each training data instance to be selected at least once across all samples, i.e., each data instance is associated with at least one sample. Due to randomness in the selection process, this desired property cannot be guaranteed. With large number of samples, however, one can be highly confident that each data instance belong to at least one sample. Therefore, we define η as the sampling error tolerance level such that each instance in the training dataset is associated with at least one of the s samples with probability $1-\eta$. Since m data instances are to be selected per sample, the minimum number of samples needed, given η, is $\frac{\ln \eta}{m \ln\left(1-\frac{1}{n_{tr}}\right)}$, where $\frac{1}{n_{tr}}$ is the probability of selecting a training instance from $\mathbf{X}_{tr}$ uniformly at random with replacement. We repeat the process of sample generation and density-ratio estimation s times, thereby generating s independent samples. Density ratios estimated for each element, which can be associated with multiple samples, are aggregated. The $normalize$ procedure averages them across those associated samples.

B. Complexity Analysis

We now compute the time and space complexity of VFKMM. As mentioned in [12], the original KMM approach (CENKMM) has a time complexity of $\mathcal{O}(n_{tr}^3+n_{tr}^2 d+n_{tr}n_{te}d)$. However, in the sampling process explained above, each sample contains m training data instances instead of n_{tr}. Hence, the time complexity of estimating $\hat{\boldsymbol{\beta}}^{*}$ in each sample is $\mathcal{O}(m^3 + m^2 d + mn_{te}d)$. The aggregation requires $\mathcal{O}(m)$. Together, the time complexity per sample is $\mathcal{O}(m^3 + m^2 d + mn_{te}d + m)$. Since computation on samples can be done in parallel, the overall time complexity remains the same. Clearly, scale-up in execution time is achieved when $m \ll n_{tr}$. Similarly, the space complexity of CENKMM is $\mathcal{O}(n_{tr}^2 + n_{tr}n_{te})$, whereas that of VFKMM is $\mathcal{O}(m^2 + mn_{te} + n_{tr})$.

C. Extension

As mentioned in Section II, ENSKMM requires the complete training data to be in memory. Likewise, VFKMM requires the complete test data to be in memory. When both training and test datasets are large, in-memory sequential computation of either methods becomes a bottleneck in the application employing it. Applying a similar sampling process as VFKMM over test data only approximates the test data distribution. Moreover, a method that successfully applies KMM on independent partitions of test data has been demonstrated in an earlier study [12].

Instead of sampling from test dataset, we propose a method that augments VFKMM with ENSKMM. Here, we first split the test data into k components, where each component is a subset with fixed number of data instances, called *test-component*. Next, for each test-component, we employ VFKMM wherein fixed-size samples are obtained from the training data. Finally, we aggregate the density ratio resulting from KMM, similar to the $normalize$ method of VFKMM in Algorithm 1. We refer to this method by Ensemble-based Very Fast KMM or EVFKMM.

IV. Empirical Evaluation

A. Dataset

Table I lists the datasets used in our empirical evaluation. All datasets are publicly available [13], [14]. Particularly, Syn002 is synthetically generated using MOA [15]. In order to

Dataset	# Features	Total Size
ForestCover	54	50,000
KDD	34	50,000
Syn002	70	50,000
MNIST	780	50,000

TABLE I: List of datasets.

		Training Data		Test Data	
		SR	SWR	SR	SWR
Method	CENKMM	✗	✗	✗	✗
	ENSKMM	✗	✗	✗	✔
	ENSTRKMM	✗	✔	✗	✗
	VFKMM*	✔	✗	✗	✗
	EVFKMM*	✔	✗	✗	✔

TABLE II: List of methods used for evaluation. SR denotes sampling with replacement, and SWR denotes sampling without replacement. The proposed approaches are indicated with *, while others are baseline methods.

simulate sampling bias between the training and test data, we follow a procedure similar to a previous study [3]. For each dataset, we first compute the covariate mean $\bar{\mathbf{x}}$ of all data instances, and select n_{tr} data instances with probability of $p(\xi = 1|\mathbf{x}^{(i)}) = \exp \frac{-\left\|\mathbf{x}^{(i)} - \bar{\mathbf{x}}\right\|}{\sigma}$, where ξ is a binary indicator variable with 1 indicating training data, and σ is the standard deviation of $\left\|\mathbf{x}^{(i)} - \bar{\mathbf{x}}\right\|$, $\forall \mathbf{x}^{(i)}$. Remaining part of the dataset is considered for testing.

B. Methods

Table II lists competing methods considered for evaluation. It also shows the type of sampling methods used in each approach, where *SR* indicates sampling with replacement and *SWR* indicates sampling without replacement. We use the original KMM (CENKMM) and ensemble KMM (ENSKMM) approaches as baseline methods for comparison (mentioned in Section II). We also consider the naive ENSTRKMM method, mentioned in Section III, as another baseline approach. It partitions the training data by sampling without replacement. Note that in Table II, there are no methods that perform sampling with replacement (SR) on the test data.

C. Experiments

We first evaluate sensitivity of each competing method to input parameters, i.e., m and η (or s). Density ratio estimates depend on the training data sample size m for VFKMM and EVFKMM, and the number of test data partitions k for ENSKMM and EVFKMM. Note that k is inversely proportional to the test data partition size. For the sake of comparison, we unify the notations m and k with the relation $m = \frac{n_{tr}}{k}$ for each training dataset, and vary the number of partitions with $k = \{5, 10, 15, 20\}$. In the case of VFKMM and EVFKMM, the effect of larger m on training data samples can be observed with smaller k value. In this set of experiments, we choose $\eta = 0.001$.

In the next set of experiments, we vary η with $k = 10$. Particularly, we vary η such that the resulting number of samples belong to the set $s = \{50, 100, 150, 200\}$. Here, instead of referring the results of these experiments with respect to η, we refer it with respect to s for clarity of understanding.

Importantly, the primary purpose of the paper is to demonstrate the scalability of proposed approaches. Therefore, in the final set of experiments, we vary the size of training dataset with $n_{tr} = \{100, 500, 1000, 1500, 2000\}$, and estimate density ratio using each competing method. Here, we use $k = 10$, $\eta = 0.001$.

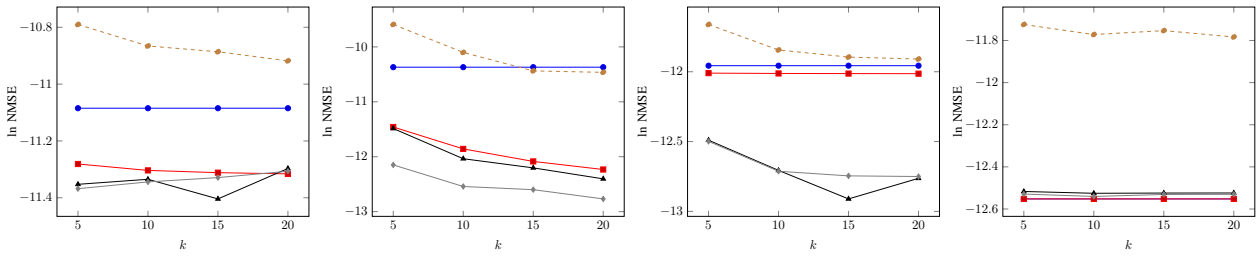
Note that each of these sets of experiments were performed over all datasets listed in Table I. Here, we only consider the first $50,000$ instances in each dataset for simplicity. Moreover, a large amount of data merely increases the test dataset size.

We measure the goodness of estimated importance weights (denoted as $\hat{\beta}(\mathbf{x}) \in \hat{\boldsymbol{\beta}}$) by computing the Normalized Mean Square Error (NMSE). This is given by $\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{\beta}(\mathbf{x}^{(i)})}{\sum_{j=1}^{n}\hat{\beta}(\mathbf{x}^{(j)})} - \frac{\beta(\mathbf{x}^{(i)})}{\sum_{j=1}^{n}\beta(\mathbf{x}^{(j)})}\right)$, where $\beta(\mathbf{x}^{(i)}) = \frac{1}{p(\xi=1|\mathbf{x}^{(i)})}$, following [12]. Here, n indicates dataset size, which depends on the method used. For example, in CENKMM, $n = n_{tr}$ since it estimates density ratio for all training data, whereas in VFKMM and EVFKMM, $n \leq n_{tr}$ since density ratio is estimated for only those data instances which are selected during random sampling. We use the well-known QP solver in CVXOPT python library [16] to execute the KMM quadratic program, with $B = 1000$ and $\epsilon = \frac{\sqrt{n_{tr}}-1}{\sqrt{n_{tr}}}$. Following [3], we use a Gaussian kernel with width γ equal to the median of pairwise distances.

D. Results

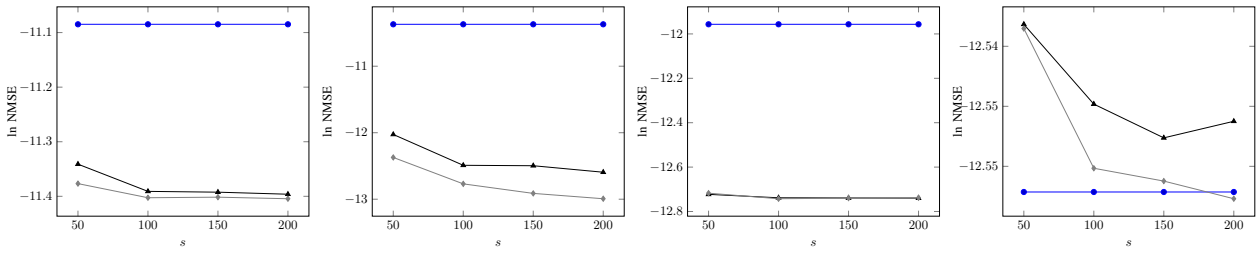
Each experiment is performed over 5 different sets of training and test data, generated via the random training data selection process, with 5 iterations in each to account for randomness in the sampling process. Moreover, we present our results on the natural logarithmic scale for clarity in comparison with competing methods.

We first present results illustrating the behavior of VFKMM and EVFKMM when k (i.e. $m = \frac{n_{tr}}{k}$) is varied, and compare it with that of CENKMM, ENSKMM and ENSTRKMM. In all these experiments, we choose $n_{tr} = 500$. Figure 2 shows the ln NMSE obtained on various datasets. Here, lower value of k indicates larger m, thereby lower s for VFKMM and EVFKMM. Also, a lower value of ln NMSE is desired. As seen in these plots, VFKMM and EVFKMM resulted in better (smaller) ln NMSE than baseline methods, on most datasets. However, ln NMSE of baseline methods in MNIST is marginally better than the proposed methods. This mixed result is expected since the sampling-based methods on the training data approximates CENKMM. Additionally, VFKMM and EVFKMM performs equivalently, with EVFKMM resulting in a better NMSE than VFKMM on multiple datasets including KDD and MNIST. Importantly, ENSTRKMM performs worst on all datasets. This supports our hypothesis that splitting (sampling without replacement) training data produces largely different data distributions compared to the original training data distribution.



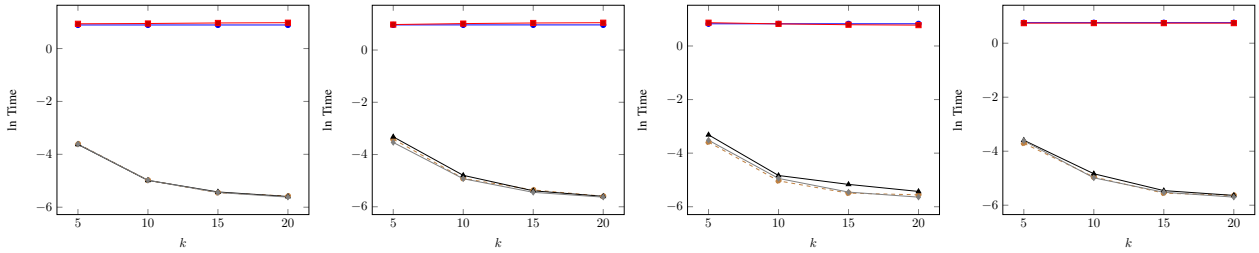
(a) ForestCover (b) KDD (c) Syn002 (d) MNIST

Fig. 2: Logarithm of NMSE with different k ($\propto \frac{1}{m}$) on VFKMM and EVFKMM, compared to CENKMM, ENSKMM and ENSTRKMM.



(a) ForestCover (b) KDD (c) Syn002 (d) MNIST

Fig. 3: Logarithm of NMSE with different s on VFKMM and EVFKMM, compared to CENKMM.



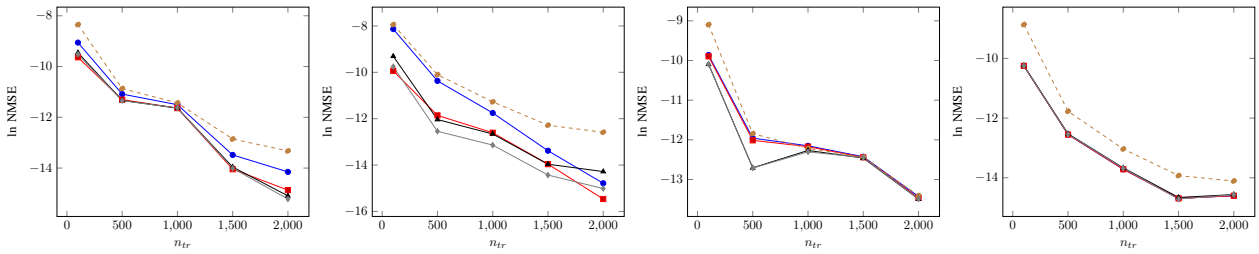
(a) ForestCover (b) KDD (c) Syn002 (d) MNIST

Fig. 4: Average execution time (in logarithm scale) with different k on VFKMM and EVFKMM, compared to CENKMM, ENSKMM and ENSTRKMM.

Figure 3 shows the effect of varying η (equivalently varying s) over all datasets using VFKMM and EVFKMM. Clearly, larger number of samples results in lower error for the two methods. Particularly, the proposed approach results in a larger error than baseline on MNIST dataset in Figure 2. With more number of samples, a decrease in error can be observed more distinctly in this dataset than others, in Figure 3. Note that performance of VFKMM and EVFKMM are very similar in Figure 3c.

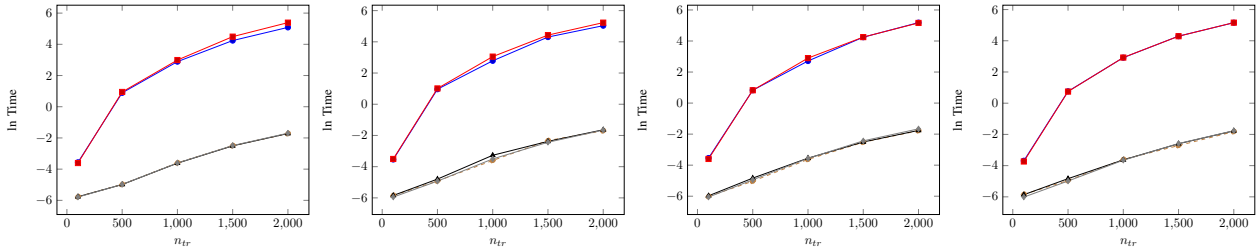
Nevertheless, the major advantage of performing sampling with replacement over training data can be observed in the lower execution time obtained on all datasets compared to the baseline methods. Figure 4 illustrates this with the average run-time (in natural logarithm of seconds) on each sample (or component) while computing density ratio. The figure shows that with smaller m (equivalently larger k), execution time per sample drastically decreases. Here, time used is the average execution for QP, where it represents the time per component in ENSKMM and the time per sample in VFKMM and EVFKMM. We assume that each component or sample can be executed in parallel.

The next set of experiments showcases the effect of NMSE and execution time on different training dataset size n_{tr}. Here, we set $k = 10$ (fixing m). Figure 5 shows the ln NMSE for all datasets considered on each competing method. From the plots, it can be observed that marginally equivalent results were obtained for each value of n_{tr} on all methods, following Figure 2. Moreover, increase in n_{tr} reduces error due to larger training sample size. As before, the benefit of VFKMM and EVFKMM is better observed in the execution time as illustrated in Figure 6. With increasing training set size, the execution time of all methods increases as expected due to the $\mathcal{O}(n_{tr}^3)$ time complexity of KMM. However, the execution time of VFKMM is very minimal as compared to other methods due to smaller size of training data in each sample. The dramatic gain in performance of VFKMM and EVFKMM



(a) ForestCover (b) KDD (c) Syn002 (d) MNIST

Fig. 5: Logarithm of NMSE with different n_{tr} on VFKMM and EVFKMM, compared to CENKMM, ENSKMM and ENSTRKMM.



(a) ForestCover (b) KDD (c) Syn002 (d) MNIST

Fig. 6: Average execution time (in logarithm scale) with different n_{tr} on VFKMM and EVFKMM, compared to CENKMM, ENSKMM and ENSTRKMM.

is traded with more number of samples. Nevertheless, each of these samples can be utilized independently to compute density estimates in a parallel and distributed manner, thereby taking advantage of a distributed system. We leave this for future work.

V. Conclusion

We present a scalable method to perform covariate shift correction using the Kernel Mean Matching method. Employing the idea of approximating the training dataset with bootstrap samples, we show a technique (VFKMM) to estimate overall importance weights for a given training dataset. Our empirical evaluation shows a dramatic improvement in execution time, while also resulting with improved performance in density ratio estimates on various large datasets. We also propose a fully scalable method called EVFKMM, where VFKMM is employed over partitions of test data.

Acknowledgments

This material is based upon work supported by NSF award no. DMS-1322353, and The Air Force Office of Scientific Research under award no. FA9550-14-1-0173, FA9550-12-1-0077.

References

[1] P.-N. Tan, M. Steinbach, V. Kumar *et al.*, *Introduction to data mining*. Pearson Addison Wesley Boston, 2006, vol. 1.

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, 2006, pp. 601–608.

[4] R. A. Berk, "An introduction to sample selection bias in sociological data," *American Sociological Review*, pp. 386–398, 1983.

[5] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *The Journal of Machine Learning Research*, vol. 8, pp. 985–1005, 2007.

[6] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," *Dataset shift in machine learning*, vol. 3, no. 4, p. 5, 2009.

[7] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *Computational Intelligence Magazine, IEEE*, vol. 10, no. 4, pp. 12–25, 2015.

[8] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[9] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in neural information processing systems*, 2008, pp. 1433–1440.

[10] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *The Journal of Machine Learning Research*, vol. 10, pp. 1391–1445, 2009.

[11] Y.-l. Yu and C. Szepesvári, "Analysis of kernel mean matching under covariate shift," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 607–614.

[12] Y.-Q. Miao, A. K. Farahat, and M. S. Kamel, "Ensemble kernel mean matching," in *Data Mining (ICDM), IEEE International Conference on*. IEEE, 2015, pp. 330–338.

[13] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[14] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[15] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[16] J. Dahl and L. Vandenberghe, "Cvxopt: A python package for convex optimization," in *Proc. eur. conf. op. res*, 2006.