Representative Subsets For Big Data Learning using k-NN Graphs

Raghvendra Mall, Vilen Jumutc, Rocco Langone, Johan A.K. Suykens
KU Leuven, ESAT/STADIUS
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
{raghvendra.mall,vilen.jumutc,rocco.langone,johan.suykens}@esat.kuleuven.be

***Abstract*—In this paper we propose a deterministic method to obtain subsets from big data which are a good representative of the inherent structure in the data. We first convert the large scale dataset into a sparse undirected k-NN graph using a distributed network generation framework that we propose in this paper. After obtaining the k-NN graph we exploit the fast and unique representative subset (FURS) selection method [1], [2] to deterministically obtain a subset for this big data network. The FURS selection technique selects nodes from different dense regions in the graph retaining the natural community structure. We then locate the points in the original big data corresponding to the selected nodes and compare the obtained subset with subsets acquired from state-of-the-art subset selection techniques. We evaluate the quality of the selected subset on several synthetic and real-life datasets for different learning tasks including big data classification and big data clustering.**

I. Introduction

In the modern era with the advent of new technologies and its widespread usage there is a huge proliferation of data. This immense wealth of data has resulted in massive datasets and has led to the emergence of the concept of Big Data. However, the choices for selecting a predictive model for Big Data learning is limited as only a few tools scale to large scale datasets. One way is to develop efficient learning algorithms which are fast, scalable and might use parallelization or distributed computing. Recently, a tool named Mahout [3] (http://www.manning.com/owen/) was built which implemented several machine learning techniques for big data using a distributed Hadoop [5] framework. The other direction is sampling [6] and [7]. There are several machine learning algorithms which build predictive models on a small representative subset of the data [2], [8]–[13] with out-of-sample extensions properties. This property allows inference for previously unseen part of the large scale data. The methods which belong to this class include kernel based methods, similarity based methods, prototype learning methods, instance based methods, manifold learning etc.

Sampling [14] is concerned with selection of points as a subset which can be used to estimate characteristics of the whole dataset. The main disadvantage of probabilistic sampling techniques is that every time the algorithm runs different subsets are obtained. It often results in large variations in the performance. Another disadvantage is that most probabilistic sampling techniques cannot capture some characteristic of the data like the inherent cluster structure unless the cluster information is available in advance. However, in case of real-life datasets this information is not known beforehand and is learnt by unsupervised learning techniques. In this paper we propose a framework to overcome these problems and select representative subsets that retain the natural cluster structure present in the data.

We first convert the big data into an undirected and weighted k-Nearest Neighbor (k-NN) [15], [16] graph where each node represents a data point and each edge represents the similarity between the data points. In this paper we propose a distributed environment to convert big data into this k-NN graph. After obtaining the k-NN graph we use the fast and unique representative subset (FURS) selection technique proposed in [1] and [2]. We propose a simple extension of FURS method to handle the case of weighted graphs. FURS selects nodes from different dense regions in the graph while retaining the inherent community structure. Finally, we map these selected nodes to the points in the original data. These points capture the intrinsic cluster structure present in the data.

We compare and evaluate the resulting subset with other sampling techniques like simple random sampling [6], stratified-random sampling [7] and a subset selection technique based on maximizing the Rènyi entropy criterion [17] and [8]. For classification, we use the subset to build a subsampled-dual least squares support vector machine (SD-LSSVM) model as proposed in [10] and use the out-of-sample extensions property to determine the class labels for points in the big data. For clustering we utilize the kernel spectral clustering (KSC) method proposed in [11]. We build the training model on the subset and again use the out-of-sample extension property of the model to infer cluster affiliation for the entire dataset. Figure 1 represents the flow chart of the steps undertaken.

II. Distributed k-NN graph generation framework

In this section we describe a parallel approach for network generation from the kernel matrix. The kernel matrix is in general a full matrix and a full graph can be generated corresponding to the kernel matrix. However, most real-life datasets have underlying sparsity i.e. each point in the dataset is similar to only a few other points in the big data. Hence, we propose to use the k-NN graph [15] and [16] for representing the big data. Now, we will present a resilient way of handling big and massive datasets just by sequencing and distributing computations in a smart way.

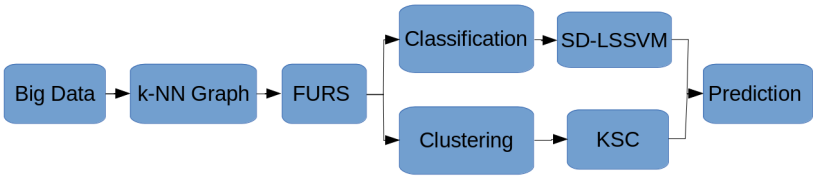


Fig. 1: Steps involved in obtaining the subset from big data and its evaluation w.r.t. learning performance.

A. Initial Setup

Our approach is based on an emerging Julia language (http://julialang.org/) and the model of the asynchronous management of co-routines [18]. To generate an undirected network where each point has connection with its top k most similar points we need a k-NN graph. This k-NN graph is obtained by sorting columns of the corresponding kernel matrix. The kernel matrix consists of similarity values between every pair of points in the data. Calculation of the entire kernel matrix for big data is not feasible or might be prohibitively expensive on a single machine or a supercomputer with enough storage and RAM. To resolve this problem we address the computation via a cluster-based approach.

Before proceeding with the distributed computational model one has to select a proper bandwidth for the Radial-Basis function (RBF) kernel which we are using to precompute the entries of the kernel matrix. There are several methods which have been proposed to tune the bandwidth [19]–[21] most of which are computationally expensive. Since the structure of the original data is not known in advance, we choose the Silverman's Rule of Thumb [22] which results in an approximate k-NN graph for the data. The bandwidth $\sigma \in \mathbb{R}^d$ used in the RBF kernel is computed as follows:

$$\sigma = \hat{\sigma} N^{-1/(d+4)}, \tag{1}$$

where $\hat{\sigma}$ is a mean standard deviation across all dimensions d and N is the total number of observations in dataset $\mathcal{D}$.

B. Kernel Matrix Evaluations

The computation of the kernel matrix is quite straightforward after obtaining σ and is given by:

$$\Omega = \begin{pmatrix} K(x_1, x_1) & \dots & K(x_1, x_N) \\ \vdots & \ddots & \vdots \\ K(x_N, x_1) & \dots & K(x_N, x_N) \end{pmatrix}, \tag{2}$$

where $K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$ is the Radial-Basis function. To compute Ω efficiently without loading the entire dataset one has to consider a batch cluster-based approach where for every node we load a batch subset $\mathcal{D}_p \subset \mathcal{D}$, $p \in \{1, \dots, P\}$ of data such that $\cup_{p=1}^{P} \mathcal{D}_p = \mathcal{D}$. The corresponding matrix slice is X_p. $\mu_p \in \mathbb{R}^d$ and $Var_p \in \mathbb{R}^d$ are the mean vector and variance vector of the data in set $\mathcal{D}_p$. We obtain an average $\mu_X = \frac{1}{P}\sum_{p=1}^{P} \mu_p$ and an average $Var_X = \frac{1}{P}\sum_{p=1}^{P} Var_p$, where X is the matrix representation for the dataset $\mathcal{D}$.

Finally, we obtain the dimension-wise standard deviation $\sigma_X = \sqrt{Var_X} \in \mathbb{R}^d$. To obtain $\hat{\sigma}$ we simply take an average across all dimensions d of σ_X as $\hat{\sigma} = \frac{1}{d}\sum_{i=1}^{d} \sigma_X^{(i)}$. The overall setup is a Map-Reduce and All-Reduce settings [4] and can be implemented using platforms like Hadoop [5] or Spark [23].

After obtaining σ we proceed with the batch-computation of the kernel matrix where for every node in a cluster we assign a matrix slice X_p over which we are performing calculations. For each batch p, we estimate $\Omega^{(p)}$ which consists of $\Omega_{ij}^{(p)}$, where $i \in \{1, \dots, N\}$, $p \in \{1, \dots, P\}$, $j \in \{m \times (p-1)+1, \dots, m \times p\}$ and m is the batch size. To compute $\Omega^{(p)}$ we load the subset X_p first and then feed chunks of the entire dataset to construct rows of $\Omega^{(p)}$ which span index $i \in \{1, \dots, N\}$. After calculating the corresponding slice $\Omega^{(p)}$ of the kernel matrix we sort in ascending order the columns of $\Omega^{(p)}$ and pick indices corresponding to the top k values. The reduction step is performed by joining indices $j \in \{m \times (p-1)+1, \dots, m \times p\}$ with the latter picked indices of k nearest neighbors for each j. By aggregating the tuples (j, k) and (k, j) one can obtain the edge list for a k-NN graph. This concept is explained in Figure 2.

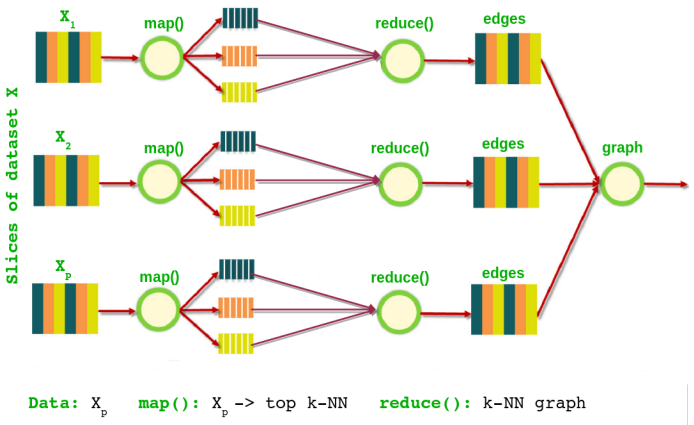


Fig. 2: Map-Reduce setting for the distributed generation of an undirected and weighted k-NN graph.

C. Sparsity & Analysis of k-NN graphs

We generate k-NN graphs for different numbers of neighbors k as the amount of sparsity in data is not known in advance. In our experiments, we generate graphs $\forall k \in \{10, 100, 500\}$. For smaller values of k we obtain graphs that are sparse representations of the data. For small k fewer edges are obtained whereas for large values of k we obtain dense graphs. Since the graph is undirected the total number of edges in a k-NN graph is equal to $2k \times N$ where N is the total number of nodes in the network. The total number of connections in the densest graph would be $N \times (N-1)$ excluding

self edges. Hence, the amount of sparsity in the graph can be mathematically be represented as: $Sparsity = 1 - \frac{2k \times N}{N \times (N-1)}$.

We need to only create a k-NN graph for the largest value of k. From this k-NN graph we can further obtain k-NN graphs for smaller values of k. We use a notion based on median degree to determine whether a sparse or a dense k-NN graph is a better representative of the original data. We calculate the median degree (m_k) for each k-NN graph and determine the number of nodes with degree $\geq m_k$. We choose m_k as it is central to the degree distribution and not influenced by outliers. The larger the number of nodes whose degree is greater than m_k for a k-NN graph the better is its representation for the big dataset. The rationale is that if a dataset has an underlying sparse representation then each point has fewer points in its vicinity. However, with large values of k in the k-NN graph we are enforcing additional connections between less similar points. As a result there will be more nodes with degrees smaller than m_k. Hence, a k-NN graph with large values of k is not a good representative for a sparse dataset. We also illustrate this in section IV.

D. Complexity analysis

In the computational complexity of the distributed k-NN graph generation, the most expensive part is the construction of the kernel matrix. The latter inherits the complexity of $\mathcal{O}(N^2)$. After distributing the calculations we can achieve an almost linear speedup in terms of the number p of corresponding nodes/workers in a cluster as $\mathcal{O}(N^2/p)$. The second important computationally expensive part is sorting of columns for $\Omega^{(p)}$ slices of the kernel matrix. Its complexity is $\mathcal{O}(N \log N)$ which corresponds to the complexity of the merge-sort [24] method as a default choice for `sortperm` operation in Julia (http://julia.readthedocs.org/en/latest/stdlib/sort/#sorting-algorithms). After taking into account the total number of nodes p in a cluster and the total number of columns N we have to process the computational complexity grows as $\mathcal{O}(\frac{N^2 \log N}{p})$. The final computational complexity is given as $\mathcal{O}(N^2(1 + \log N)/p)$.

E. Machine Configuration for Experiments

In our experiments for generation of the k-NN graph we use a computer with 64-bit, 132 Gb RAM, 2.2GHz processor and 40 cores. We utilize all these 40 cores for parallel computations. Once we have obtained the subset, the rest of the learning procedures like classification and clustering are performed on a PC machine with Intel Core i7 CPU and 8 GB RAM under Matlab 2013b. This is to showcase the power of model based techniques [2], [10], [11] for big data learning.

III. FURS for Weighted Graphs

Once we generate the k-NN graph $G(V, E)$, where V represents the set of vertices and E represents the set of edges, we use it as a network where every connection is weighted according to the corresponding weight of the edge in the graph. The weights are the obtained by selecting the top k values from $\Omega^{(p)}$. The degree of each node is calculated as the sum of the weight of all the edges to and from that node. The FURS selection technique can handle large scale networks as shown in [1], [2] for unweighted networks as long as the network can be stored on a single computer. All the operations involved in FURS selection method are performed on a single computer.

The problem of subset selection including nodes from different dense regions in the graph can be stated as:

$$\begin{aligned} \max_{B} \quad & J(B) = \sum_{j=1}^{M} D(v_j) \\ \text{s.t.} \quad & v_j \in c_i, \\ & c_i \in \{c_1, \ldots, c_{noc}\} \end{aligned} \tag{3}$$

where $D(v_j)$ represents the weighted degree centrality of the node v_j, M is the size of the subset, c_i represents the i^{th} community and noc represents the number of communities in the network which cannot be determined explicitly beforehand.

The FURS selection technique is a greedy solution to the aforementioned problem and is given as:

$$\begin{aligned} & J(B) = 0 \\ & \text{While } |B| < M \\ & \max_{B} \quad J(B) := J(B) + \sum_{j=1}^{M^t} D(v_j) \\ & \text{s.t.} \quad \text{Nbr}(v_j) \rightarrow \text{deactivated, iteration} t, \\ & \quad\quad\; \text{Nbr}(v_j) \rightarrow \text{activated, iteration} t+1, \end{aligned} \tag{4}$$

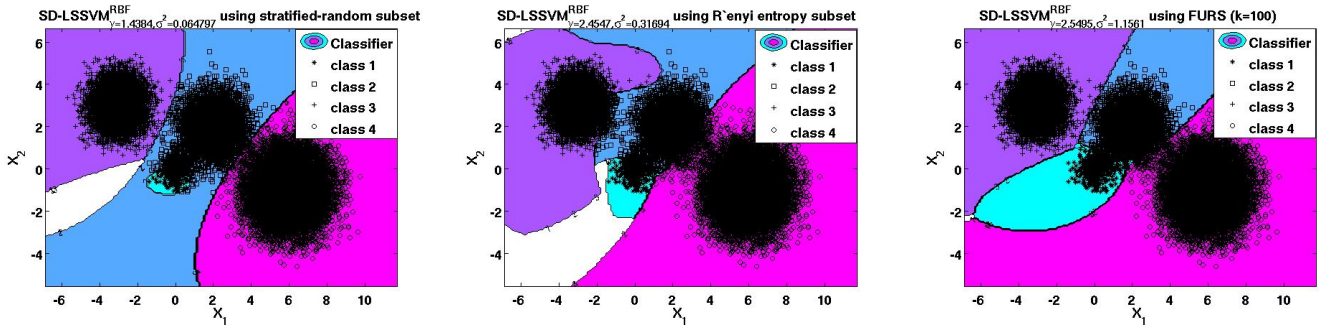
where M^t is the size of the set of nodes selected by FURS during iteration t and $\text{Nbr}(v_j)$ represents the neighbors or the set of nodes to which node v_j is connected. A detailed description of the steps involved in FURS selection technique are mentioned in [1]. After obtaining the representative subset by FURS algorithm, we map the nodes in the selected subset back to the points of the original data.

IV. Classification Experiments

There are several supervised learning techniques [8]–[10] which build a predictive model on a small representative subset of the data and use its out-of-sample extension property to obtain the required class information about the unseen part of the data. In this paper we showcase the effectiveness of representative subset selection for the subsampled-dual LSSVM (SD-LSSVM) model proposed in [10].

A. Experimental setup

We conduct experiments on 2 synthetic datasets. The first synthetic dataset (4G) comprises 4 Gaussians with $20,000$, $5,000$, $3,000$ and 500 2-dimensional points respectively. The Gaussian with the smallest density is considered negative ($-$) class while the other 3 Gaussians form the positive ($+$) class. There is a small overlap between the positive and negative class. The second synthetic dataset (5G) consists of 3 Gaussians with $50,000$, $20,000$, $10,000$ points. They form the $+$ class. There are 2 separate Gaussian of 500 points which



(a) (b) (c)

Fig. 3: Generalization results using different subset selection techniques for the synthetic 4G dataset. We can observe from Figure 3a and 3b that the stratified-random and stratified-Rènyi entropy based selection technique cannot result in good generalization for the classification problem. However, the FURS selection technique (on k-NN graph with $k = 10$) performs much better as depicted in Figure 3c. This can also be observed from the results in Table II.

Dataset	N	d	N_+	N_-	M	M_+	M_-	N_{test}
4G	28,500	2	28,000	500	114	112	2	28,386
5G	81,000	2	80,000	1,000	324	320	4	80,676
Magic	19,020	11	12,363	6657	100	65	35	18,920
Shuttle	58,000	9	45,820	12,180	100	79	21	57,900
Skin	245,057	3	193,595	51,462	500	395	105	244,557

TABLE I: Dataset description and usage. N_+ and N_- represents the number of points in the $(+)$ and $(-)$ class respectively. We select as few points (M) as possible while still retaining good generalization. M_+ and M_- represents the number of points in the selected subset from the $(+)$ and $(-)$ class respectively. We maintain the ratio of $(+)$ to $(-)$ class in the selected subset.

form the $-$ class. We also performed experiments on 3 real-life datasets obtained from the UCI repository [27]. Table I gives a summary of the datasets used in the experiments.

For each dataset we run the Map-Reduce procedure once and from a big k-NN graph obtain graphs for different values of k, *i.e.* $k \in \{10, 100, 500\}$. Since FURS is a deterministic algorithm, we run the subset selection only once on the k-NN graph emerged from Map-Reduce process. To leverage effects of the unbalanced datasets we perform all subset selections separately w.r.t. each class. We generate separate k-NN graphs for each class and select points from each of these classes. The total number of data points selected from each class by the different subset selection techniques is given in Table I. We compare the performance of the subset selected by FURS algorithm with stratified-random [7] and stratified-Rènyi entropy-based [8] subset selection methodologies.

In all our experiments regarding classification and supervised learning we use a 2-step procedure for tuning the hyperparameters of the SD-LSSVM model [10], such as γ and the bandwidth σ of RBF kernel. This procedure consists of Coupled Simulated Annealing (CSA) [25] initialized with 5 random sets of parameters for the first step and the simplex method [26] for the second step. After CSA converges to some local minima we select the tuple of parameters that attains the lowest error and start the simplex procedure to refine our selection. On every iteration step for CSA and simplex method we perform a 10-fold cross-validation. We run SD-LSSVM model 50 times to leverage effects of randomizations and test the model on nearly test set for predictions. We average the error results and report it along with the standard deviations. Figure 3 depicts the results of different subset selection technique on the 4G synthetic dataset.

B. Numerical results

In this subsection we present the generalization errors obtained for several public and artificial datasets described in Table I. It can be observed from Table II that for most datasets FURS algorithm results in the improved classification rates and lower standard deviations. We can observe that for some difficult artificial datasets, like 4G and 5G stratified-random (SR) sampling and stratified-Rènyi entropy (SRE) based subset selection techniques might fail because classes are highly unbalanced and selected subset may contain outliers or boundary cases. On the other hand for some datasets, like Magic, stratified-random sampling is performing best.

Dataset	SR	SRE	FURS$_{k=10}$	FURS$_{k=100}$	FURS$_{k=500}$
	Generalization or Test Error $\pm$ Standard Deviation				
4G	0.395±0.235	0.898±0.375	**0.252±0.023**	**0.331±0.063**	0.414±0.070
5G	0.264±0.225	0.298±0.142	0.358±0.296	**0.082±0.016**	**0.086±0.009**
Magic	**25.32±3.913**	**28.44±4.446**	33.07±2.076	31.05±4.143	36.13±3.062
Shuttle	2.437±1.104	2.330±0.958	4.223±0.998	**1.482±0.600**	**1.980±0.715**
Skin	0.578±0.387	**0.254±0.078**	3.277±1.133	**0.494±0.078**	0.772±0.080

TABLE II: Averaged generalization errors along with their standard deviations for SD-LSSVM model. Generalization errors are expressed as percentage. The best and second-best results are highlighted.

In general, we can observe improved generalization errors for FURS$_{k=100}$ and FURS$_{k=500}$. The SRE based selection technique performs better in some cases like Magic and Skin dataset. This might indicate that we need to increase size of subset for FURS selection technique for these datasets. The performance of SRE can be explained by the characteristics

of the Rènyi entropy criterion which selects points uniformly from the dataset. Figure 4 shows the variations in the error estimations of the different subset selection methods for the Shuttle dataset. From Figure 4 we can observe that the FURS$_{k=10}$ performs worse indicating that a dense k-NN graph is better representative of the original dataset. We can also observe that the FURS selection method has lower standard deviations in the error estimations.

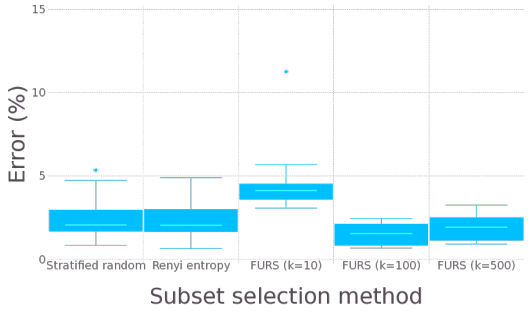


Fig. 4: Error estimations for classification by different subset selection techniques on Shuttle dataset. FURS$_{k=100}$ results in the best mean performance with the lowest variance.

V. Clustering Experiments

A powerful model based clustering technique with out-of-sample extensions property is the kernel spectral clustering (KSC) [11] method. We use it in our clustering experiments.

A. Experimental Setup

We conducted clustering experiments on the 2 synthetic datasets (4G and 5G) which we also used for supervised learning. However, we generate the k-NN graph for the whole data in the unsupervised case followed by applying the FURS algorithm on the whole network. We also experimented on 5 real-life datasets, 3 (House, Mopsi Finland and KDDCupBio) of which are obtained from http://cs.joensuu.fi/sipu/datasets/ and others (Gas Sensory Array Drift (Batch) and Power Consumption (Power)) is obtained from UCI repository [27]. In our experiments the training set and validation set are obtained by the subset selection technique. We use the same size for the training and validation sets (M) and the same kernel parameter σ for different subset selection technique. We generate k-NN graphs with different k ($k \in \{10, 100, 500\}$).

We use the median degree (m_k) to estimate the nodes from which the representative set can be selected by FURS algorithm (section II-C). The idea of calculating the number of nodes whose degree is greater than m_k helps to determine whether a sparse or dense representation is better for the k-NN graph as mentioned in section II-C and depicted in Table III. In Table III $\mathcal{S}_k$ represents the set containing nodes whose degree is greater than m_k. From Table III we observe that $|\mathcal{S}_{k=10}|, |\mathcal{S}_{k=100}|$ and $|\mathcal{S}_{k=500}|$ represents the cardinality or the number of points whose degree is above median degree m_k for the respective k-NN graph. Table III also provides a summary of the datasets.

Dataset	N	d	M	N_{test}	$\lvert\mathcal{S}_{k=10}\rvert$	$\lvert\mathcal{S}_{k=100}\rvert$	$\lvert\mathcal{S}_{k=500}\rvert$
4G	28,500	2	100	80,500	15,852	**19,760**	*19,384*
5G	81,000	2	810	81,000	45,328	**51,304**	*47,156*
House	34,112	3	342	34,112	**20,556**	*19,400*	18,864
Mopsi Finland (MF)	13,467	2	135	13,467	*7,202*	**7,388**	6,231
Batch	13,910	128	139	13,910	**7,171**	*5,534*	4,649
KDDCupBio	145,751	74	1,458	145,751	**64,034**	*60,307*	55,700
Power	2,049,280	7	2,050	2,049,280	*1,096,668*	**1,292,203**	1,083,454

TABLE III: Dataset description and usage. We use the entire dataset as test set for clustering. We select 1% of the total points as training set and validation set (M). For each dataset we bold and emphasize the sets from which we obtain the training set and validation set respectively. The Power dataset results 20Gb size k-NN graph for $k = 500$.

We compare the FURS selection technique with simple random sampling and the Rènyi entropy based selection method. We run FURS only once whereas we perform other subset selection methods 10 times. We compare clustering results based on several internal (Silhouette (SIL), Davies-Bouldin (DB)) [28] and external quality (ARI, VI) [28] metrics. Figure 5 reflects the clustering results for different subset selection technique on 4G synthetic dataset.

B. Results & Analysis

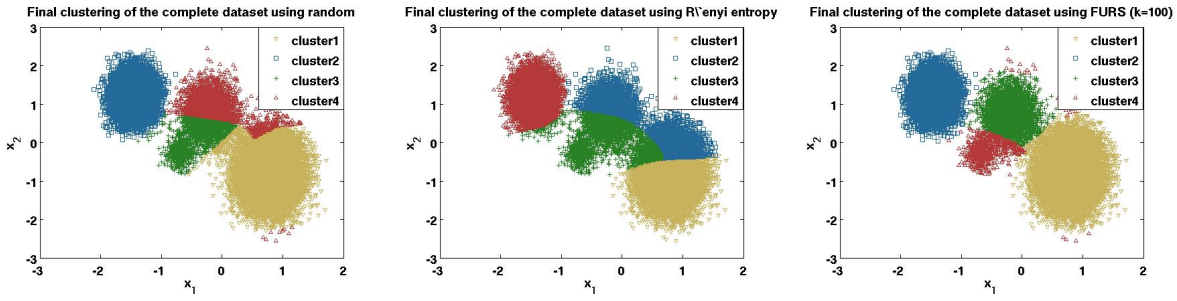
Table IV provides a detailed comparison of the FURS selection technique with other subset selection methods on various internal and external quality metrics. From Table IV we observe that the FURS algorithm completely outperforms other techniques on the two synthetic datasets. For the Batch dataset, FURS performs best w.r.t. internal quality metrics like SIL and DB while Rènyi entropy based subset selection technique gives best results w.r.t. VI. Higher values of SIL criterion and lower values of DB index represent better quality clusters. From Table IV we observe that for the large scale real-world datasets (House and Mopsi Finland) where the ground truth is unknown, the FURS selection technique performs better than random sampling and Rènyi entropy based subset. For big datasets (KDDCupBio and Power) we use the DB quality measure as the silhouette measure is computationally very expensive. Random sampling technique gives better results on KDDCupBio but it also shows large variations in the results.

VI. Conclusion

We proposed a method to obtain representative subsets of the big data for model based learning techniques. We proposed to convert the big data into a k-NN graph using a distributed framework and then selected the required representative subset using the FURS algorithm. The selected subset retained the natural cluster structure of the data. We illustrated the effectiveness of this selected subset for big data learning.

Acknowledgments

EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This chapter reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants. Flemish Government: FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants. IWT: projects: SBO POM (100031); PhD/Postdoc grants. iMinds Medical Information Technologies SBO 2014. Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017.)



(a) (b) (c)

Fig. 5: Comparison of clustering performance of KSC model using the same kernel parameter ($\sigma = 0.188$) but different subsets for building the model. We can observe poor generalizations for random and Rènyi entropy based selection techniques in 5a and 5b. However, the FURS algorithm results in good generalization performance as depicted in 5c and Table IV.

Dataset	**Random**				**Rènyi entropy**				**FURS**			
	ARI	VI	DB	SIL	ARI	VI	DB	SIL	ARI	VI	DB	SIL
4G	0.743 ± 0.157	0.485 ± 0.213	0.804 ± 0.167	0.667 ± 0.059	0.402 ± 0.236	1.223 ± 0.412	2.909 ± 2.820	0.594 ± 0.148	**0.985**	**0.097**	**0.616**	**0.765**
5G	0.664 ± 0.233	0.638 ± 0.402	2.011 ± 0.998	0.649 ± 0.049	0.471 ± 0.143	0.911 ± 0.263	0.885 ± 0.276	0.601 ± 0.054	**0.988**	**0.081**	**0.808**	**0.764**
Batch	**0.064 ± 0.029**	2.633 ± 0.099	4.287 ± 0.764	0.503 ± 0.066	0.006 ± 0.008	**1.978 ± 0.115**	3.969 ± 0.783	0.460 ± 0.134	0.039	2.452	**1.868**	**0.669**
	DB		**SIL**		**DB**		**SIL**		**DB**		**SIL**	
House	0.612 ± 0.154		0.679 ± 0.073		0.507 ± 0.028		0.579 ± 0.006		**0.407**		**0.751**	
Mopsi Finland	0.897 ± 0.935		0.824 ± 0.085		**0.526 ± 0.223**		0.886 ± 0.010		0.568		**0.920**	
	DB				**DB**				**DB**			
KDDCupBio	**2.932 ± 1.205**				4.233 ± 1.82				3.883			
Power	2.558 ± 1.374				2.0619 ± 0.760				**1.921**			

TABLE IV: Comparison of FURS algorithm with other subset selection technique w.r.t. various quality metrics.

REFERENCES

[1] R. Mall, R. Langone and J.A.K. Suykens; FURS: Fast and Unique Representative Subset selection retaining large scale community structure. *Social Network Analysis and Mining*, 3(4):1-21, 2013.

[2] R. Mall, R. Langone and J.A.K. Suykens; Kernel Spectral Clustering for Big Data Networks. *Entropy (SI: Big Data)*, 15(5):1567-1586, 2013.

[3] S. Owen, R. Anil, T. Dunning and E. Friedman; Mahout in Action, *Manning Publications Co.*, 1^{st} Edition, 2011.

[4] A. Agarwal, O. Chapelle and J. Langford; A Reliable Effective Terascale Linear Learning System. *JMLR*, 15(1):1111-1133, 2014.

[5] T. White; Hadoop: The Definitive Guide. 1^{st} *Edition, O'Reilly Media, Inc.*, (ISBN: 0596521979, 9780596521974), 2009.

[6] D.S. Yates, D.S. Moore and D.S. Starnes; The Practice of Statistics. 3^{rd} *Edition, Freeman*, (ISBN 978-0-7167-7309-2), 2008.

[7] C.E. Särdnal, B.Swenson and J. Wretman; Model assisted survey sampling. *Springer-Verlag*, (ISBN 0-387-40620-4), 1992.

[8] K. De Brabanter, J. De Brabanter, J.A.K. Suykens and B.De Moor; Optimized Fixed-Size Kernel Models for Large Data Sets. *Computational Statistics & Data Analysis*, 54(6):1484-1504, 2010.

[9] R. Mall and J.A.K. Suykens; Sparse Variations to Fixed-Size Least Squares Support Vector Machines for Large Scale Data. *In Proc. of PAKDD 2013*, pp. 161-173, GoldCoast, Australia, 2013.

[10] R. Mall and J.A.K. Suykens; Very Sparse LSSVM Reductions for Large Scale Data. *IEEE Trans. on Neural Net. and Learning Sys*, in press.

[11] C. Alzate and J.A.K. Suykens; Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):335-347, 2010.

[12] R. Mall, S. Mehrkanoon, R. Langone and J.A.K. Suykens; Optimal Reduced Sets for Sparse Kernel Spectral Clustering. *IJCNN*, 2014.

[13] R. Mall, R. Langone and J.A.K. Suykens; Self-Tuned Kernel Spectral Clustering for Large Scale Networks, *In Proc. of the IEEE BigData 2013)*, Santa Clara, USA, Oct. 2013.

[14] D.S. Moore and G.P. McCabe; Introduction to the practice of statistics. *W.H. Freeman & Co.*, 5^{th} Edition, (ISBN 0-7167-6282-X), 2005.

[15] D. Eppstein, M.S. Paterson and F. Yao; On-nearest neighbor graphs. *Discrete and Computational Geometry*, 17(3): 263-282, 1997.

[16] G.L. Miller, S. Teng, W. Thurston and S.A. Vavasis; Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1): 1-29, 1997.

[17] C.K.I. Williams and M. Seeger; Using the Nyström method to speed up kernel machines. *Advances in NIPS*, 13:682-688, 2001.

[18] J. Liu, A. Kimbell and A.C. Myers; Interruptible Iterators. *SIGPLAN Not.*, 41(1):283-294, 2006.

[19] M. Rudemo; Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9:65-78, 1982.

[20] A.W. Bowman; An alternative method of cross-validation for smoothing of density estimators. *Biometrika*, 71:353-360, 1984.

[21] A. Kleiner, A. Talwalker, P. Sarkar and M.I. Jordan; The Big Data Bootstrap. *ICML*, 2012.

[22] B.W. Silverman; Density Estimation for Statistics and Data Analysis. *London: Chapman & Hall/CRC*, (ISBN 0-412-24620-1) 1998.

[23] M. Zaharia, M.Chowdhury, M.J. Franklin, S.Shenker and I. Stoica; Spark: Cluster computing with Working Sets. *In Proc. of the* 2^{nd} *USENIX Conference on Hot Topics in Cloud Computing*, Boston, 2010.

[24] T.H. Cormen, C. Stein, R.L. Rivest and C.E. Leiserson; Introduction to Algorithms. *McGraw-Hill Higher Education*, 2^{nd} Edition, (ISBN: 0070131511), 2001.

[25] S. Xavier de Souza, J.A.K. Suykens, J.Vandewalle and D.Bolle; Coupled Simulated Annealing for Continuous Global Optimization. *IEEE Tran. Systems, Man, and Cybertics - Part B*, 40(2): 320-335, 2010.

[26] J.A. Nelder and R. Mead; A simplex method for function minimization. *Computer Journal*, 7:308-313, 1965.

[27] C.L. Blake and C.J. Merz; UCI repository of machine learning databases. *http://archive.ics.uci.edu/ml/datasets.html* , Irvine, CA, 2007.

[28] R. Rabbany, M. Takaffoli, J. Fagnan, O.R. Zaiane, and R.J.G.B. Campello; Relative Validity Criteria for Community Mining Algorithms. *In Proc. of ASONAM*, pp. 258-265, 2012.