Journal of Universal Computer Science, vol. 21, no. 6 (2015), 842-855
submitted: 28/11/14, accepted: 20/2/15, appeared: 1/6/15 © J.UCS

Discrete Neighborhood Representations and Modified Stacked Generalization Methods for Distributed Regression

Héctor Allende-Cid
(Escuela de Ingeniería Informática,
Pontifícia Universidad Católica de Valparaíso, Chile
hector.allende@ucv.cl)

Héctor Allende, Raúl Monge
(Departamento de Informática,
Universidad Técnica Federico Santa María, Valparaíso, Chile
hallende@inf.utfsm.cl, rmonge@inf.utfsm.cl)

Claudio Moraga
(European Centre for Soft Computing, 33600, Mieres, Asturias, Spain
TU Dortmund University, 44220 Dortmund, Germany
mail@claudio-moraga.eu)

Abstract: When distributed data sources have different contexts the problem of Distributed Regression becomes severe. It is the underlying law of probability that constitutes the context of a source. A new Distributed Regression System is presented, which makes use of a discrete representation of the probability density functions (pdfs). Neighborhoods of similar datasets are detected by comparing their approximated pdfs. This information supports an ensemble-based approach, and the improvement of a second level unit, as it is the case in stacked generalization. Two synthetic and six real data sets are used to compare the proposed method with other state-of-the-art models. The obtained results are positive for most datasets.

Key Words: Distributed Machine Learning, Context-aware Regression, Similarity representation.
Category: I.2.11, G.3.

1 Introduction

Due to the rapid growth of the amount of data that is being stored, automatic learning has become increasingly important over the years. The total amount of information available on the Internet has had an exponential growth in the last years. By 2005, the total size was around 600 terabytes [D-lib]. Nowadays, the total amount of data stored is almost incalculable (a rough estimation indicates an amount of data in the order of zettabytes). The rapid growth of the data available, presents new oportunities for applications of Machine Learning and Automatic Data Analysis. Since human reasoning is not able to handle large data sets, the need of automatic data analyzers is a necessity. Due to this reason, challenges related with the scalability and efficiency of the learning algorithms have become of central importance. Classic machine learning algorithms, usually work with monolithic data sets, this means that the entire data set is loaded into main memory. When the amount of data is very large, this is unfeasible, because the algorithm will not be able to train on the whole data set, or it will be impractical due to computational or memory restrictions. Also, it should be noted, that if the amount of information that is distributed or the number of distributed sources is too large, this could lead to problems related to Big Data. To handle this type of problems, Parallel and Distributed approaches are having a lot of attention from the Machine Learning community. Distributed Machine Learning (DML) is often mentioned with Parallel Machine Learning (PML) in literature. While both attempt to improve the performance of traditional Data Mining systems, they assume different system architectures and take different approaches. In DML, computers and data are distributed and communicate through message passing. In PML, a parallel system is assumed with processors sharing memory and/or storage. Computers in a DML system may be viewed as processors sharing nothing. This difference in architecture greatly influences algorithm design, cost model, and performance measure in Distributed and Parallel Machine Learning.

Since it was proposed, the field of Distributed Machine Learning (DML) has been very active and is enjoying a growing amount of attention. There are many real world applications where the data is distributed naturally. In most cases, the amount of data distributed is so large, that is unfeasible to trasmit it to a centralized node (avoiding the creation of a Big Data problem), so there is no alternative other than to treat the problem with a Distributed Learning approach.

Most of the current DML techniques treat the distributed data sets as a single virtual table and assume that there is a global model which could be generated if the data were combined or centralized, completly neglecting the different semantic contexts that this distributed data sets may have [Wirth et al., 2001, Jung, 2012]. If we see this as a statistical learning problem, we deal with samples of data that follow different underlying laws of probability. Loosely speaking Machine Learning models try to find a function which relates a given output $\underline{y}$ with an input vector $\underline{x}$. The classic Machine Learning approach is related with the estimation of the joint probability distribution $H(\underline{X}, \underline{Y})$. The joint probability distribution can be decomposed in the conditional probability distribution and the marginal one ($H(\underline{X}, \underline{Y}) = F(\underline{Y}/\underline{X})G(\underline{X})$). In this paper, context is defined as the joint probability distribution that governs each data source.

The next section will present a brief view of the state of the art related to this work. In section 3 we disclose the proposed algorithm which will be able to address the problem addressed above. In [Allende-Cid et al., 2014b] a preliminary version was presented. In section 4 we show some experimental results with synthetic and real datasets. The last section is devoted to discuss the results and to make some conclusions.

2 State of the Art

In the field of Distributed Machine Learning (DML), in the last decade, a large amount of work has been reported. A large fraction of DML algorithms focus on combining predictive models. This approach has emerged from empirical experimentation due to a requirement for higher prediction accuracy. Recently, several researchers treat distributed learning systems as a centralized ensemble-based method [Chawla et al., 2004, Lattner et al., 2010, Lazarevic and Obradovic, 2001, Tsoumakas et al., 2002, Jung, 2013]. Several learning algorithms are applied at each local site, using separate training data to mine local knowledge. A new data point is then classified/predicted from the predictions of all local sites using ensemble methods such as stacking, boosting, majority voting, simple average, or winner-takes-all methods. In general, DML approaches apply ensemble methods to minimize the communication costs and to increase the performance of the system predictions.

There are well known approaches for distributed classification problems (see e.g. [Caragea et al., 2001, Moretti et al., 2008, Park and Kargupta, 2002, Rodriguez et al., 2011, Jung, 2014]. It is not straightforward to adapt these approaches to the regression task. Both problems are related, but the strategies to solve them are different. Another drawback of these methods, is that they assume that the underlying laws of probability of the distributed sources are the same. This is an assumption that is inherited from the ensemble-based approaches in classic machine learning. When a dataset is resampled, it is assumed that the resampled data follows the same underlying law of probability as the original set. In real-world distributed problems, it is impossible to assure that, because the real underlying law of probability is unknown.

Another challeging task is the one of Distributed Data Clustering. The authors of the work [Balcan et al., 2013] present a distributed K-means and K-median approach in topologies of general communication. In [Eyal et al., 2011] the authors present a generic distributed data clustering algorithm applied to sensor networks. Other distributed clustering works that can be found recently in the literature address the problem of large-scale data sets and evolving data streams [Forman and Zhang, 2000, Hefeeda et al., 2012, Ienco et al., 2013]. There are also distributed approaches to other tasks or problems, e.g. Genetic Algorithms [Lopez et al., 2011] and Mining Association Rules [Lin et al., 2013].

In this work, the task that will be addressed is the regression task. Loosely speaking the task of regression can be presented as the task of finding the relationship between input and output variables, where the outputs are real-valued. This can be used for predicting an expected value by presenting to the model an unknown input. Works related with the regression task in Distributed Machine Learning are scarce. The majority of algorithms deal with the classification problem. In this work the task of regression will be addressed.

In [Xing et al., 2003, Xing et al., 2005] the authors propose a series of algorithms based on, what the authors call, a meta-learning approach to deal with the regression problem addressing the context heterogeneity case. The authors propose a meta-learning-based hierarchical model that is able to be successfully used in distributed scenarios with context heterogeneity. The definition of context in this work is, however, the variance that the distributed sources have in their outputs, thus neglecting the context change in the input space. The authors claim that this change of context between distributed sites follows a Gaussian distribution.

In [Allende-Cid et al., 2013], an ensemble approach based on building neighborhoods of similar datasets is presented. To build the neighborhoods, it is assumed that the datasets follow a known underlying law of probability (which is possible when working with synthetic examples), and using the Hypothesis Test based on divergence measures, they form the corresponding neighborhoods. In this work it is necessary to assume a known underlying law of probability, i.e. multivariate normal distribution, in order to perform the Hypothesis tests [Pardo, 2005, Salicru et al., 1994]. The algorithm in [Allende-Cid et al., 2013] can be summarized as follows: At first, local models are trained with the available distributed data sets. In each distributed node, a local algorithm is trained with its corresponding data. After that, assuming that the underlying laws of probability of each distributed data sets are known (multivariate gaussian distributions), the mean and variance-covariance matrices (parameters of the underlying law of probability) are shared across all distributed nodes. With this information, in each distributed node i, an Hypothesis Test is performed with the parameters of node i and the parameters all other nodes $j = 1, \ldots, k$, where $j \neq i$ and k is the total number of distributed data sets. After performing all Hypothesis Tests, the neighborhood for node i is built if there was no evidence to reject H_0 (that the parameters of both underlying laws of probability were the same). Also all the local models are shared across the sites. Then, a second stage learner is trained, where the inputs of this learner are the outputs of *all* local models. The final output of the model is the ensemble of all second stage models, that belong to the same neighborhood, where the new data inputs are registered.

For a more complete review on the state of the art of Distributed Machine Learning Algorithms, please refer to [Peteiro-Barral and Guijarro-Berdinas, 2013].

3 A New Distributed Regression System

To avoid working under the assumption of a known underlying law of probability, we can use a discrete representation of the probability density functions (pdfs). For this we can use n-dimensional histograms. A histogram $H(x)$ of a set $[\underline{x}_1, \underline{x}_2]$ represents the frequency of each value. If the dataset is one-dimensional, the histogram is represented by a vector. The length of the vector depends in the number of bins used to represent the histogram. If the dataset is 2-dimensional, the histogram is represented by a matrix. For n-dimensional cases the data structure used to represent is a n-rank tensor. There is a computational cost related in the choice of the number of bins to construct the histogram. The length of the vector that represents the histogram is n^d, where n is the number of bins and d the dimension of the input vector x. There is a trade-off between the performance of the algorithm and the computational cost, as it will be seen in the Experiments section.

Before building histogram representations for all distributed datasets, the minimum and maximum per input dimension must be shared across all data sets. This is necessary, because we need to build histograms that are comparable with each other. To make them comparable we need bins with the same limits. This information shared across the system, does not contradict the restriction of sharing raw data, because it only shares the minimum and maximum global values of the examples of each distributed source. With this data, we obtain the global minimum and maximum values of all distributed sources, using this information to build histograms for all distributed sources, with the same bin limits. The idea is to use histograms to build a vector of size r, that represents the dissimilarity between 2 datasets, using r distance measures. We define a number of different distance/similarity measures, using preferable distances from different families [Cha, 2007]. If the underlying laws of probability of 2 data sets are identical the distance vector will be (0,0,..., 0). If there are k distributed data sources, distances vectors $\{d_{i1}, \ldots, d_{ik}\}$ will be generated for node i. Applying a clustering algorithm to all distance vectors, we can define a neighborhood for node i, by taking into account all distance vectors that belong to the same cluster of d_{ii}. E.g. if the distance vector d_{i1} and d_{i3} of a total of $k = 5$ distributed nodes belong to the same cluster as d_{ii}, then the nodes 1 and 3 belong to the neighborhood of i. In this work the clustering algorithm used is the Hierarchical Clustering algorithm. Preliminary results regarding histogram representation were presented in [Allende-Cid et al., 2014]. The main difference between this approach and the one presented in [Allende-Cid et al., 2014] is that the second level models, in this proposal are trained only with the outputs of the local models, that belong to the same neighborhood. In the previous work, the second level models were trained with all the outputs of the local models, whether they belonged to the neighborhood or not.

In the next subsection we present the proposed algorithm.

3.1 Learning

In this subsection we present the pseudo-code of the proposal and explain each phase in detail.

1. *Phase 1 - Local Learning*. Suppose that there are k distributed data sets. At each node N_i, where $i = 1, \ldots k$, we use an available learning algorithm to train a local predictive model L_i from data source D_i of that node. The choice of the learning algorithm is not restricted to any particular kind. The local training data consists in n dimensional input vectors $((x_{i1}, x_{i2}, \ldots, x_{in}))$ and a response variables y_i.

2. *Phase 2 - Model and information transmission*. Each node N_i, where $i = 1, \ldots, k$ receives the minimum and maximum of each attribute of the other nodes. With this information each distributed node N_i creates an n-dimensional histogram. The histograms are then shared across the distributed nodes. After that, in each distributed

Algorithm 1 Proposed Algorithm (Training batchwise)

```
 1: $k$ = number of sites
 2: Define the number of bins of the histogram and assign it to an auxiliary variable
    $nubs$.
 3: Data sets consist of $d$ dimensional vectors $(x_1, x_2, x_3, \ldots, x_d)$ and continuos re-
    sponse variable $y_i$.
 4: // PHASE 1
 5: for $i \in 1, \ldots, k$ do
 6:     Read (in parallel) the data of the prevailing batch.
 7:     Train each model $L_i$ with corresponding data set $D_i$. (It can be performed in
        parallel)
 8: // PHASE 2
 9: for $i \in 1, \ldots, k$ do
10:     Send (in parallel) from each node $D_i$ the model parameters to all other nodes
        $D_j$, where $j \neq i$. Each node $D_i$ receives the model parameters from the other
        models and the global minimum and maximum values of each variable.
11: Create the histogram for each data set $D_i$, where $i = 1, \ldots, k$ with $nubs$ bins and
    broadcast them to all other data sources. Each node $D_i$ receives the histograms
    of the other nodes.
12: With the histograms use a hierarchical clustering algorithm to build the neigh-
    borhood binary vectors.
13: // PHASE 3
14: In each distributed site $D_i$, with $i = 1, \ldots, k$ train a second level model $G_i$ based
    on Stacked Generalization, using only the output of the Local models trained in the
    sites that belong to the Neighborhood of $i$.
15: Using the neighborhoods calculate the weight vector to multiply each $G_i$ output by
    the corresponding weight.
16: // OUTPUT
17: Calculate the mean of all $G_i$ models that received the output of the local models
    that belong to the neighborhood $i$ multiplied by a corresponding weight.
```

node, the distance vector is calculated, with respect to all other nodes. Also the parameters of all the local models are shared across the distributed nodes, so each node has a copy of the rest of the local models. With a clustering algorithm, we generate the neighborhoods of distributed nodes. The result is a binary vector h_i with k binary variables, which indicates the nodes which follow the same underlying law of probability of D_i. In other terms we will refer to this vector as the Neighborhood vector. E.g. If there are 5 distributed nodes D_1,D_2,D_3, D_4 and D_5, and we are checking if node D_1 has the same distribution as the rest, and only the variables $h_1(1)$, $h_1(2)$ and $h_1(4)$ are equal to 1, then this means that the data contained in the nodes N_2 and N_4 follow the same distribution as in node N_1.

3. *Phase 3 - Generation of second level learners.* Since every node N_i has a copy of the other local models, each local model L_l, $l = 1, \ldots, k$, contained in node N_i is trained with the local data from that node (D_l). Each of the local models j that belong to the neighborhood of node N_i outputs a response variable $\hat{y}_{ij}$ with the local data D_i, where $j = 1, \ldots, k$. Each local node N_i applies then a stacked learning algorithm G_i (second level model) which is trained with the outputs of local models ($\hat{y}_{ij}$), that belong to the neighborhood of N_i and the real response variable y_i, obtained from the training data of the node D_i. This is inspired in the Stacking model of Wolpert [Wolpert, 1992].

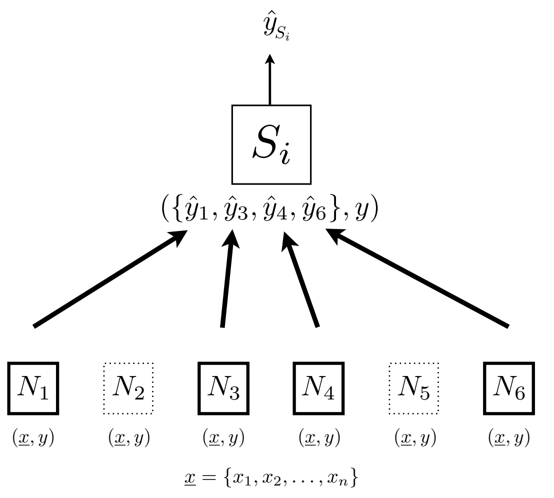
$\hat{y}_{S_i}$
S_i
$(\{\hat{y}_1, \hat{y}_3, \hat{y}_4, \hat{y}_6\}, y)$
N_1
N_2
N_3
N_4
N_5
N_6
$(\underline{x}, y)$
$\underline{x} = \{x_1, x_2, \ldots, x_n\}$

Figure 1: Example of a Stacking model S_i trained in Node i. The dotted nodes do not belong to the Neighborhood of Node i. The outputs of the local models in Node 1, 3, 4 and 6 (nodes that belong to the same neighborhood) are used as inputs for the Stacking model. Only these models are used to build the second level learner in Phase 3. $\hat{y}_{S_i}$ is the estimation of y, that the S_i stacking model predicts.

3.2 Predicting

1. *Final output of the proposed model.* The output of our model is the following: Whenever a new example arrives at a node N_i, we compute all the outputs of the local models that are stored in this node. We have an apriori information of which of the other nodes have data following a close underlying law of probability of the current node, which is reflected in the neighborhood vector mentioned above (h_j, where $j = 1, \ldots, k$). Then the output of the local models in this node are transmitted

to only the other nodes which have a non-zero label in this vector. The final output of the model is the weighted sum of all the G_i model outputs that received the output of the local models of their corresponding neighborhoods h_i. The weights are calculated with the following equation: $w'_j = \frac{w_j}{\sum w_j}$, where $w_j = 1 - |\hat{d}_{ij}|^2$. E.g. in the example presented in Fig. 1 the final output of the model is the weighted average of models G_1, G_3 and G_k, because only the variables $h_2(1)$, $h_3(3)$ and $h_3(k)$ are distinct from zero.

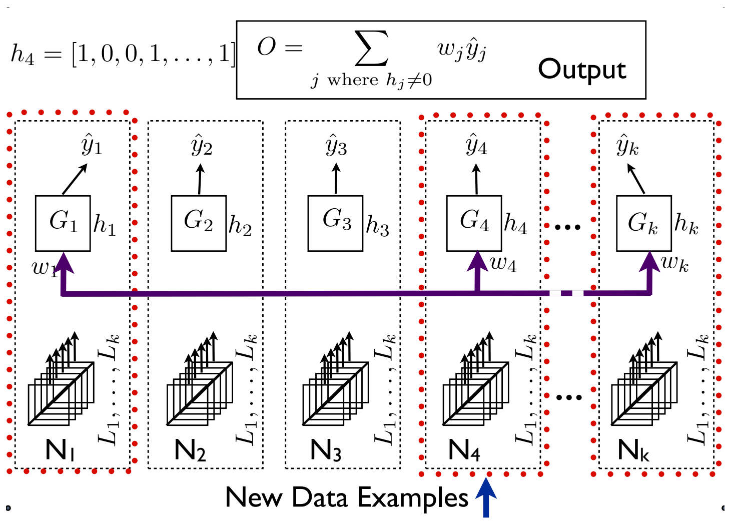

Figure 2: Architecture of the proposed system

4 Experimentation

In this section we present the results obtained by our proposal with 2 synthetic data sets and 6 real world data sets in terms of Mean-squared error (MSE). The functions used in the synthetic experiments were:

$$y = 0.01x_1 + 0.02{x_2}^2 + 0.9{x_3}^3 + \epsilon \tag{1}$$

and

$$y = 0.6x_1 + 0.3x_2 + \epsilon \tag{2}$$

where $\epsilon \sim N(0,1)$ and $\underline{x} \sim N(\underline{\mu}, \Sigma)$. For both synthetic experiments we tested a specific scenario: Data coming from multivariate normal distributions with different covariance matrix Σ and mean vectors $\underline{\mu}$. The number of distributed sources generated were 10, 12, 14, 16, 18 and 20. Half of the distributed sources were generated from a $x \sim N(\underline{\mu_1}, \Sigma_1)$ and the other half from $x \sim N(\underline{\mu_2}, \Sigma_2)$. In the first scenario, as stated previously half of the distributed sources were generated from a $N(\underline{\mu_1}, \Sigma_1)$ where

$\underline{\mu_1} = (0,0,\ldots,0)$ and $\Sigma_1 = [1,0,\ldots,0;0,1,\ldots,0;\ldots;0,0,\ldots,1]$ and the other with $\underline{\mu_2} = (i,i,\ldots,i)$ and $\Sigma_2 = [i^2,0,\ldots,0;0,i^2,\ldots,0;\ldots;0,0,\ldots,i^2]$ where $i = (1,2,3,4)$. Variable i represents the 4 different set of parameters with which the proposal was validated. With this different values of i we present 4 different cases. The total number of examples for each experiment were 10000. For further details on this synthetic experiments and preliminary results using histograms to build neighborhoods, please refer to [Allende-Cid et al., 2014].

The Communities and Crime, Parkinson and Bike Sharing and Wine Quality data sets can be obtained from [UCI]. The Communities and Crime dataset consists in 1994 examples and 128 attributes. The data was split in 32 datasets. The Parkinson dataset consists in 5875 examples and 26 attributes. It was split in 42 datasets, according to the patients ID. The Bike Sharing dataset consiste in 17389 examples and has 16 attributes. It was split in 12 datasets, according to the month of the year. The Wine dataset consists in 1599 examples of red wine and 4898 examples of white wine. The number of attributes is 11. The latter one was split in 10, 20 and 30 datasets, where half of the datasets are generated from the red wine examples and the other half from the white wine examples. The Wind dataset consists in 29050 examples and was distributed in 54 sources. For further details of the Wind dataset, please refer to [Allende-Cid et al., 2013b]. The Ailerons dataset consists in 13740 examples and 40 attributes. The data set was split in 4, 8 and 12 sources.

To compare our proposal we tested 4 different algorithms:

– Reference model that has access to all data. (*Global*)

– Local models (used only in the synthetic experiments)

– Model presented in [Xing et al., 2005]

– Model presented in [Allende-Cid et al., 2013]

– Proposal.

The performance measure is the mean value of the MSE obtained in each distributed source. This means that in each distributed source 80% of the data was set for training purposes and the other 20% for testing purposes. In each node the MSE was obtained with the testing data. The final performance measure is the mean of the MSE results obtained in each node. The local and the second level models were feedforward artificial neural networks. The optimum number of neurons of the hidden layer was searched from 2 to 10 neurons. The clustering algorithm used in this experimentation to create the neighborhoods was the Hierarchical Clustering. The distance metrics used were the Euclidean, Sorensen, Intersection and Kullback-Leibler distances.

In Table 1 we observe the results for Experiment 1. The last two columns show the results of the Proposal with different number of bins used to construct the histogram. We used 20 and 30 bins to show the influence of this parameter in the proposal, denoted

Case	#	Global	[Xing et al., 2005]	Local	Prop-20	Prop-30
1	10	24.05 ± 12.71	19.33 ± 3.879	7.177 ± 2.477	2.221 ± 0.809	**1.272** ± 0.041
1	12	37.22 ± 10.44	15.72 ± 1.247	5.644 ± 0.998	3.660 ± 1.258	**1.594** ± 0.116
1	14	28.34 ± 8.873	22.21 ± 5.491	9.060 ± 3.499	2.603 ± 1.267	**1.347** ± 0.082
1	16	25.36 ± 6.989	17.56 ± 3.275	6.774 ± 1.623	5.502 ± 1.568	**1.571** ± 0.347
1	18	20.80 ± 10.31	21.96 ± 2.769	7.469 ± 1.487	1.829 ± 0.567	**1.245** ± 0.119
1	20	23.26 ± 10.42	15.83 ± 0.354	4.593 ± 0.396	2.482 ± 1.117	**1.526** ± 0.103
2	10	5.981 ± 1.120	217.0 ± 10.85	8.634 ± 1.749	4.022 ± 1.116	**2.572** ± 0.325
2	12	17.25 ± 7.261	229.3 ± 8.127	12.96 ± 3.573	5.305 ± 1.771	**2.375** ± 0.505
2	14	16.72 ± 5.629	214.4 ± 14.64	51.40 ± 28.36	16.12 ± 9.816	**2.519** ± 0.348
2	16	14.36 ± 4.620	232.0 ± 10.94	13.81 ± 4.876	8.328 ± 3.561	**2.599** ± 0.272
2	18	10.21 ± 3.698	233.8 ± 16.72	13.19 ± 3.243	3.225 ± 1.269	**2.000** ± 0.235
2	20	8.914 ± 0.404	194.5 ± 6.934	7.508 ± 0.613	3.209 ± 1.310	**1.994** ± 0.203
3	10	30.16 ± 5.380	1634 ± 58.73	24.92 ± 5.107	6.551 ± 1.525	**5.32** ± 0.951
3	12	26.92 ± 6.849	1552 ± 50.36	21.47 ± 3.928	5.516 ± 0.545	**4.35** ± 4.321
3	14	35.33 ± 14.25	1551 ± 46.52	33.41 ± 6.082	**3.918** ± 0.349	4.915 ± 0.496
3	16	25.53 ± 5.473	1613 ± 75.13	42.03 ± 15.24	4.939 ± 0.439	**3.502** ± 4.152
3	18	24.83 ± 5.231	1599 ± 51.20	20.46 ± 4.113	4.963 ± 0.332	**3.383** ± 2.459
3	20	36.11 ± 9.980	1565 ± 84.05	32.86 ± 8.577	**3.745** ± 0.994	5.912 ± 2.492
4	10	41.08 ± 13.04	7274 ± 143.2	36.95 ± 4.930	**12.69** ± 4.559	18.68 ± 10.70
4	12	73.43 ± 21.18	7198 ± 219.0	72.67 ± 14.85	**12.72** ± 2.298	14.01 ± 10.34
4	14	36.10 ± 7.335	7214 ± 99.15	30.65 ± 5.767	11.19 ± 2.583	**10.43** ± 5.944
4	16	27.04 ± 11.56	7402 ± 231.6	46.24 ± 8.907	**9.091** ± 1.747	11.34 ± 8.563
4	18	69.57 ± 19.21	7244 ± 163.3	63.97 ± 20.57	13.88 ± 3.322	**10.37** ± 6.163
4	20	38.16 ± 9.801	7209 ± 145.5	58.61 ± 14.90	12.62 ± 3.384	**7.975** ± 3.557

Table 1: Mean value and standard deviation of 20 experimental runs. Synthetic Experiment 1.

by Prop-20 and Prop-30, respectively. As it can be seen in the results, in every case the proposal, whether it is with 20 or 30 number of bins, outperforms all the other models. The error from the model proposed in [Xing et al., 2005], as the difference of the parameters that were used to generate the data increases, has an exponential growth, while the performance of the proposal deteriorates in a polynomial way. It can be seen also, that as the number of bins used to build the histogram increases, the performance of the model also increases. Prop-30 outperforms almost in every case the other approaches, particularly when the degree of distortion of the underlying law of probability is low and minor differences should be noticed. Only in some experiments the algorithm Prop-20 has a better performance. Despite that, in those cases, the error of the Prop-30 algorithm has comparable results.

Case	#	Global	[Xing et al., 2005]	Local	Prop-20	Prop-30
1	10	1.000 ± 0.003	1.002 ± 0.003	1.006 ± 0.003	0.996 ± 0.009	**0.992** ± 0.007
1	12	1.003 ± 0.007	1.005 ± 0.007	1.010 ± 0.007	**1.003** ± 0.008	1.010 ± 0.012
1	14	0.997 ± 0.004	1.002 ± 0.003	1.010 ± 0.004	1.001 ± 0.003	**0.992** ± 0.006
1	16	1.003 ± 0.005	1.005 ± 0.006	1.013 ± 0.006	1.002 ± 0.012	**0.991** ± 0.007
1	18	1.007 ± 0.005	1.009 ± 0.006	1.017 ± 0.005	**0.995** ± 0.009	1.010 ± 0.004
1	20	1.000 ± 0.006	1.002 ± 0.006	1.014 ± 0.007	1.022 ± 0.009	**0.985** ± 0.006
2	10	0.987 ± 0.005	1.014 ± 0.006	0.992 ± 0.006	1.015 ± 0.011	**0.997** ± 0.005
2	12	0.982 ± 0.003	1.012 ± 0.004	0.993 ± 0.005	0.992 ± 0.008	**0.991** ± 0.008
2	14	0.991 ± 0.002	1.024 ± 0.009	0.998 ± 0.002	1.018 ± 0.009	**0.995** ± 0.006
2	16	0.978 ± 0.006	1.010 ± 0.008	0.988 ± 0.005	1.025 ± 0.009	**0.994** ± 0.007
2	18	0.987 ± 0.006	1.016 ± 0.008	0.999 ± 0.006	1.003 ± 0.007	**0.989** ± 0.009
2	20	0.984 ± 0.007	1.026 ± 0.012	1.003 ± 0.008	1.025 ± 0.006	**0.981** ± 0.010
3	10	0.993 ± 0.005	1.129 ± 0.018	1.000 ± 0.006	1.000 ± 0.011	**0.992** ± 0.006
3	12	1.008 ± 0.005	1.193 ± 0.022	1.016 ± 0.005	**0.995** ± 0.006	0.997 ± 0.008
3	14	0.989 ± 0.003	1.232 ± 0.030	0.999 ± 0.004	**0.996** ± 0.009	0.997 ± 0.011
3	16	0.993 ± 0.009	1.178 ± 0.028	1.001 ± 0.009	**0.977** ± 0.007	0.985 ± 0.007
3	18	0.990 ± 0.004	1.165 ± 0.024	1.002 ± 0.004	1.006 ± 0.009	**0.988** ± 0.005
3	20	0.994 ± 0.005	1.183 ± 0.017	1.010 ± 0.006	1.010 ± 0.005	**0.991** ± 0.007
4	10	1.017 ± 0.005	1.563 ± 0.039	1.022 ± 0.004	**0.983** ± 0.006	1.006 ± 0.008
4	12	1.010 ± 0.005	1.492 ± 0.040	1.015 ± 0.005	1.007 ± 0.012	**0.997** ± 0.007
4	14	1.018 ± 0.006	1.558 ± 0.043	1.024 ± 0.006	**0.980** ± 0.003	0.998 ± 0.008
4	16	1.011 ± 0.005	1.585 ± 0.044	1.021 ± 0.005	0.993 ± 0.009	**0.989** ± 0.004
4	18	1.019 ± 0.004	1.638 ± 0.041	1.033 ± 0.004	**0.983** ± 0.002	0.988 ± 0.009
4	20	1.020 ± 0.006	1.652 ± 0.055	1.033 ± 0.006	0.981 ± 0.007	**0.979** ± 0.007

Table 2: Mean value and standard deviation of 20 experimental runs. Synthetic Experiment 2.

In Table 2 we observe a similar behaviour than the one observed in the previous synthetic experiment. Both of the proposals, with different number of bins, outperform the other models. As was observed in the previous experiment, the best results were obtained with Prop-30. This proposal outperforms the other approaches in almost every experiment. Only in a few experiments the best results were obtained by Prop-20, but the results of both proposals are comparable.

The results of our proposal with the real data sets are presented in Table 3. The Mean and standard deviation of 20 experimental runs are presented. The number of bins used for the proposed algorithm was 10, for all real dataset experiments. The performance measure is the Mean-squared error. We compare the results with the models presented in [Xing et al., 2005] and [Allende-Cid et al., 2013]. The results were favorable for our

Dataset	#	Global	[Xing et al., 2005]	[A-Cid 2013]	Proposal
Parkinson	48	0.0022 ± 0.0004	**0.0025** ± **0.0002**	0.0034 ± 0.0006	0.0033 ± 0.0006
Crime	32	0.0080 ± 0.0031	**0.0335** ± 0.0057	0.0380 ± 0.0063	0.0363 ± **0.0056**
Bike	12	1.3759 ± 2.7726	13.660 ± 16.098	59.033 ± 15.812	**1.5803** ± **0.592**
Wind	54	30.661 ± 5.6572	42.936 ± 2.282	65.675 ± 9.318	**35.439** ± **2.574**
Wine	10	0.6014 ± 0.0253	0.8278 ± 0.1707	0.6268 ± 0.0254	**0.5406** ± **0.0131**
Wine	20	0.6025 ± 0.0238	0.8378 ± 0.1223	0.9258 ± 0.0617	**0.5293** ± **0.0165**
Wine	30	0.5998 ± 0.0132	0.8375 ± 0.077	1.1530 ± 0.0751	**0.5244** ± **0.0132**
Ailerons*	4	0.1299 ± 0.0065	11.3587 ± 16.235	0.1298 ± 0.0059	**0.1231** ± **0.0022**
Ailerons*	8	0.1303 ± 0.0005	1.0893 ± 0.388	**0.1311** ± **0.0005**	0.1328 ± 0.0031
Ailerons*	12	0.1311 ± 0.0003	0.3916 ± 0.009	0.1340 ± 0.0032	**0.1292** ± **0.0012**

Table 3: Mean value and standard deviation of 20 experimental runs. The Ailerons* results should be multiplied by 10^6.

proposal in the majority of the presented datasets. There were only 2 datasets (Parkinson and Crime), where the model proposed in [Xing et al., 2005] outperformed our model. Despite that, the difference between our proposal and [Xing et al., 2005] are in the range of 10^{-3}. In the rest of the presented datasets, the proposal outperformed the other approaches. As can be seen in the table, the difference was in some cases considerable. The Parkinson dataset, has been always been treated as a monolithic dataset in many works, so it is understandable that there is no considerable difference between the examples, thus affecting the proposal. It also should be pointed out, that the proposal outperforms the previous approach presented in [Allende-Cid et al., 2013] in every dataset. In the Wine dataset we observe that the proposed model even outperforms the model that has access to all the data. The results of the Global model, a model that uses all the data centralized, are only reported for comparison purposes. It should be pointed out that the Global model does not get the best results in every experiment as it should be expected. This could be due the fact that in these cases the data on which the model is trained, has a mixture of statistical distributions that affects the generalization of the model.

5 Conclusions and Future Work

In this proposal we present a distributed regression approach that is able to detect different contexts in the input space, thus improving the performance of local models in the task of regression from distributed sources. As the results show, it is very important, not to neglect the different contexts that are present in the distributed sources. Also it is sometimes impractical to assume that the underlying law of probability is known, so a discrete way of representing it is necessary. Also it is important to focus on the second level models, that are based on stacked generalization, in order to filter the inputs they receive (only receive the outputs of the local models, that are part of the neighborhood).

The clustering algorithm is also crucial, so in future studies, different algorithms like the found in [Bello-Orgaz et al., 2012] or [Menendez et al., 2014] could possibly improve the results. Further studies are necessary in order to establish error bounds and to formally prove the convergence of the algorithm.

Acknowledgments

This work was supported by the following research grants: Fondecyt 1110854 and DGIP-UTFSM. The work of C. Moraga was partially supported by the Foundation for the Advancement of Soft Computing, Mieres, Spain.

References

[Allende-Cid et al., 2013] Allende-Cid, H., Moraga, C., Allende, H and Monge, R. Context-Aware Regression from Distributed Sources. IDC 2013, Prague, Czech Republic. pp 17–22, 2013.

[Allende-Cid et al., 2013b] Allende-Cid, H., Moraga, C., Allende, H and Monge, R. Wind Speed Forecast under a Distributed Learning Approach. V Chilean Workshop of Pattern Recognition, Temuco, Chile, 2013.

[Allende-Cid et al., 2014] Allende-Cid, H., Allende, H., and Monge, R. Soft Computing applied to Distributed Regression with Context-Heterogeneity. Journal of Multivalued Logic and Soft Computing, (In press), 2014.

[Allende-Cid et al., 2014b] Allende-Cid, H., Moraga, C., Allende, H., and Monge, R. Regression from Distributed Data Sources using Discrete Neighborhood representations and modified Stacked Generalization Models. IDC 2014, Madrid, Spain, Studies in Computational Intelligence Volume 570, pp 249-258, 2014.

[Balcan et al., 2013] Balcan, M.-F., Ehrlich, S. and Liang, Y. Distributed k-means and k-median clustering on general communication topologies. Paper presented at the meeting of the NIPS, 2013.

[Bello-Orgaz et al., 2012] Bello-Orgaz, G., Menéndez, H., and Camacho, D. Adaptive K-Means Algorithm for overlapped graph clustering. International Journal of Neural Systems. Ed. By World Scientific. Vol. 22 (5), pp. 1–19, 2012.

[Caragea et al., 2001] Caragea, D., Silvescu, A., and Honavar, V.: Analysis and synthesis of agents that learn from distributed dynamic data sources. In: Wermter, S., Austin, J., Willshaw D.J. (eds.) Emergent Neural Computational Architectures Based on Neuroscience, pp. 547-559, 2001.

[Cha, 2007] Cha, S.-H. Comprehensive survey on distance/similarity measures between probability density functions. International Journal of Mathematical Models and Methods in Applied Sciences. 1(4): 300 – 307, 2007.

[Chawla et al., 2004] Chawla, N.V., Lawrence Hall, O., Kevin Bowyer, W., and Phillip Kegelmeyer, W. Learning ensembles from bites: A scalable and accurate approach. In Journal Machine Learning Res. 5: 421–445, 2004.

[D-lib] D-Lib Magazine. A research library based on historical collections of the Internet Archive. http://www.dlib.org/dlib/february06/arms/02arms.html (2000). Accesed 26 February 2014.

[Eyal et al., 2011] Eyal, I., Keidar, I., and Rom, R. Distributed data clustering in sensor networks. Distributed Computing, Volume 24, Issue 5, pp 207–222, 2011.

[Forman and Zhang, 2000] Forman, G., and Zhang, B. Distributed data clustering can be efficient and exact. SIGKDD Explor. Newsl. 2, 2, 34–38, 2000.

[Hefeeda et al., 2012] Hefeeda, M., Gao, F., and Abd-Almageed, W. Distributed approximate spectral clustering for large-scale datasets. In Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing (HPDC '12), 2012.

[Ienco et al., 2013] Ienco, D., Bifet, A., Zliobaite, I., and Pfahringer,B. Clustering Based Active Learning for Evolving Data Streams. Discovery Science, pp. 79–93, 2013.

[Jung, 2012] Jung, J.J.: "Evolutionary Approach for Semantic-based Query Sampling in Large-scale Information Sources," Information Sciences, Vol. 182, No. 1, pp. 30-39, 2012.

[Jung, 2013] Jung, J.J.: "Semantic Wiki-based Knowledge Management System by Interleaving Ontology Mapping Tool," International Journal on Software Engineering and Knowledge Engineering, Vol. 23, No. 1, pp. 51-63, 2013.

[Jung, 2014] Jung, J.J.: "Understanding information propagation on online social tagging systems: a case study on Flickr," Quality & Quantity, Vol. 48, No. 2, pp. 745-754, 2014.

[Lattner et al., 2010] Lattner, A., Grimme, A., and Timm, I. An evaluation of Meta Learning and Distributed Strategies in Distributed Machine Learning. European Conference on Data Mining 2010, pp. 67–74, 2010.

[Lazarevic and Obradovic, 2001] Lazarevic, A., and Obradovic, Z. The Distributed Boosting Algorithm. In Knowledge Discovery and Data mining, pp. 311–316, 2001.

[Lopez et al., 2011] López, L.I., Bardallo, J.M., De Vega, M.A., and Peregrin, A. Regaltc: A distributed genetic algorithm for concept learning based on regal and the treatment of counter examples. Soft Comput. 15(7), 1389-1403, 2011.

[Lin et al., 2013] Lin,C.-W., Hong, T.-P., Chen, Y.-F., Lin, T.-C., and Pan, S.-T. An Integrated MFFP-tree Algorithm for Mining Global Fuzzy Rules from Distributed Databases. Journal of Universal Computer Science, Volume 19, Issue 4, 2013.

[Menendez et al., 2014] Menéndez, H., Barrero, D., and Camacho, D. A Genetic Graph-based approach for Partitional Clustering. International Journal of Neural Systems. Ed. By World Scientific. Vol. 24 (1430008), pp. 1–19, 2014.

[Moretti et al., 2008] Moretti, C., Steinhaeuser, K., Thain, D., and Chawla, N.V.: Scaling up classifiers to cloud computers. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), pp. 472-481, 2008.

[Pardo, 2005] Pardo, L. Statistical Inference Based on Divergence Measures. Ed. Chapman and Hall, 2005.

[Park and Kargupta, 2002] Park, B. and Kargupta, H. Distributed Data Mining: Algorithms, Systems, and Applications. Data Mining Handbook. 2002.

[Peteiro-Barral and Guijarro-Berdinas, 2013] Peteiro-Barral, D. and Guijarro-Berdinas, B. A survey of methods for distributed machine learning. Journal of Progress in Artificial Intelligence, vol. 2, pp. 1–11, 2013.

[Rodriguez et al., 2011] Rodríguez, M., Escalante, D.M., and Peregrín, A.: Efficient distributed genetic algorithm for rule extraction. Appl. Soft Comput. 11(1), 733-743, 2011.

[Salicru et al., 1994] Salicrú,M., Morales,D., Menéndez,M. L., and Pardo,L. On the applications of divergence type measures in testing statistical hypotheses. J. Multivar. Anal. 51, Vol. 2, 372–391, 1994.

[Tsoumakas et al., 2002] Tsoumakas, G., Vlahavas, I.P. Effective Stacking of Distributed Classifiers. ECAI 2002: 340–344, 2002.

[UCI] Bache, K. and Lichman, M. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013

[Wirth et al., 2001] Wirth, R., Borth, M., and Hipp, J. When distribution is part of the semantics: A new problem class for distributed knowledge discovery. ECML 2001. 3–7, 2001.

[Wolpert, 1992] Wolpert, D., Stacked Generalization., Neural Networks, 5(2), pp. 241–259., 1992.

[Xing et al., 2003] Xing, Y., Madden, M., Duggan, J., and Lyons G. Context-based Distributed Regression in Virtual Organizations . In Parallel and Distributed computing for Machine Learning.7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 03), Cavtat-Dubrovnik, Croatia, 2003.

[Xing et al., 2005] Xing, Y., Madden, M., Duggan, J., and Lyons G. Context-Sensitive Regression Analysis for Distributed Data. ADMA 2005: 292–299, 2005.