Parallel Algorithm of Local Support Vector Regression for Large Datasets

Le-Diem Bui[1,3], Minh-Thu Tran-Nguyen[1], Yong-Gi Kim[3], and Thanh-Nghi Do[1,2(✉)]

[1] College of Information Technology, Can Tho University, Cantho, 92100, Vietnam
[2] UMI UMMISCO 209 (IRD/UPMC), UPMC, Sorbonne University, Pierre and Marie Curie University, Paris 6, France
[3] AI Lab, Computer Science Department, Gyeongsang National University, Jinju, Korea
dtnghi@cit.ctu.edu.vn

Abstract. We propose the new parallel algorithm of local support vector regression (local SVR), called kSVR for effectively dealing with large datasets. The learning strategy of kSVR performs the regression task with two main steps. The first one is to partition the training data into k clusters, followed which the second one is to learn the SVR model from each cluster to predict the data locally in the parallel way on multi-core computers. The kSVR algorithm is faster than the standard SVR for the non-linear regression of large datasets while maintaining the high correctness in the prediction. The numerical test results on datasets from UCI repository showed that our proposed kSVR is efficient compared to the standard SVR.

Keywords: Support vector regression (SVR) · Local support vector regression (local SVR) · Large datasets

1 Introduction

Support vector machines (SVM) proposed by [1] and kernel-based methods have shown practical relevance for classification, regression and novelty detection. Successful applications are reported for face identification, text categorization and bioinformatics [2]. Nevertheless, the SVM learning is accomplished through a quadratic programming (QP), so that the computational cost of a SVM approach is at least square of the number of training datapoints making SVM impractical to handle large datasets. There is a need to scale-up SVM learning algorithms to deal with massive datasets.

In this paper, we propose the new parallel algorithm of local support vector regression (local SVR), called kSVR for effectively dealing with the non-linear regression of large datasets. Instead of building a global SVR model, as done by the classical algorithm is very difficult to deal with large datasets, the kSVR algorithm is to learn in the parallel way an ensemble of local ones that are easily

T.K. Dang et al. (Eds.): FDSE 2017, LNCS 10646, pp. 139–153, 2017.
https://doi.org/10.1007/978-3-319-70004-5_10

trained by the standard SVR algorithms. The kSVR algorithm performs the training task with two main steps. The first one is to use $k$means algorithm [3] to partition the training dataset into k clusters. The idea is to reduce the data size for training local non-linear SVR models at the second step. The algorithm learns k non-linear SVR models in the parallel way on multi-core computers in which a SVR model is trained in each cluster to predict the data locally. The numerical test results on datasets from UCI repository [4] showed that our proposal is efficient compared to the standard SVR in terms of training time and prediction correctness. The kSVR algorithm is faster than the standard SVR in the non-linear regression of large datasets while maintaining the high prediction correctness.

The paper is organized as follows. Section 2 briefly introduces the SVR algorithm. Section 3 presents our proposed parallel algorithm of local SVR for the non-linear regression of large datasets. Section 4 shows the experimental results. Section 5 discusses about related works. We then conclude in Sect. 6.

2 Support Vector Regression

Given a training dataset with m datapoints x_i $(i = 1, \ldots, m)$ in the n-dimensional input space R^n, having corresponding targets $y_i \in R$, support vector regression (SVR) proposed by [1] tries to find the best hyperplane (denoted by the normal vector $w \in R^n$ and the scalar $b \in R$) that has at most ϵ deviation from the target value y_i. The SVR pursues this goal with the quadratic programming (1).

$$\begin{aligned} &\min (1/2)\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i - \alpha_i{}^*)(\alpha_j - \alpha_j{}^*)K\langle x_i, x_j\rangle - \sum_{i=1}^{m}(\alpha_i - \alpha_i{}^*)y_i + \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i{}^*) \\ &s.t.\begin{cases} \sum_{i=1}^{m}(\alpha_i - \alpha_i{}^*) = 0 \\ 0 \le \alpha_i, \alpha_i{}^* \le C \quad \forall i = 1, 2, ..., m \end{cases} \end{aligned} \tag{1}$$

where C is a positive constant used to tune the margin and the error and a linear kernel function $K\langle x_i, x_j\rangle = \langle x_i \centerdot x_j\rangle$.

The support vectors (for which $\alpha_i, \alpha_i{}^* > 0$) are given by the solution of the quadratic programming (1), and then, the predictive hyperplane and the scalar b are determined by these support vectors. The prediction of a new datapoint x based on the SVR model is as follows:

$$predict(x, SVR\ model) = \sum_{i=1}^{\sharp SV}(\alpha_i - \alpha_i{}^*)K\langle x, x_i\rangle - b \tag{2}$$

Variations on SVR algorithms use different prediction functions [5]. It only needs replacing the usual linear kernel function $K\langle x_i, x_j\rangle = \langle x_i \centerdot x_j\rangle$ with other popular non-linear kernel functions, including:

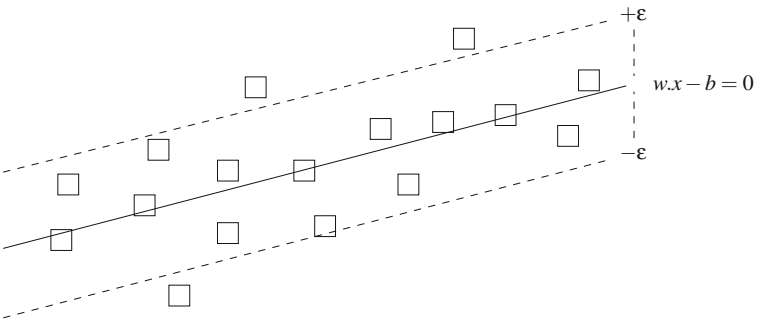

Fig. 1. Linear support vector regression

– a polynomial function of degree d : $K\langle x_i, x_j\rangle = (\langle x_i \centerdot x_j\rangle + 1)^d$
– a RBF (Radial Basis Function): $K\langle x_i, x_j\rangle = e^{-\gamma\|x_i - x_j\|^2}$

The SVR models are most accurate and practical relevance for many successful applications reported in [2].

3 Parallel Algorithm of Local Support Vector Regression

The study in [6] illustrated that the computational cost requirements of the SVR solutions in (1) are at least $O(m^2)$ (where m is the number of training datapoints), making standard SVM intractable for large datasets. Learning a global SVR model on the full massive dataset is challenge due to the very high computational cost.

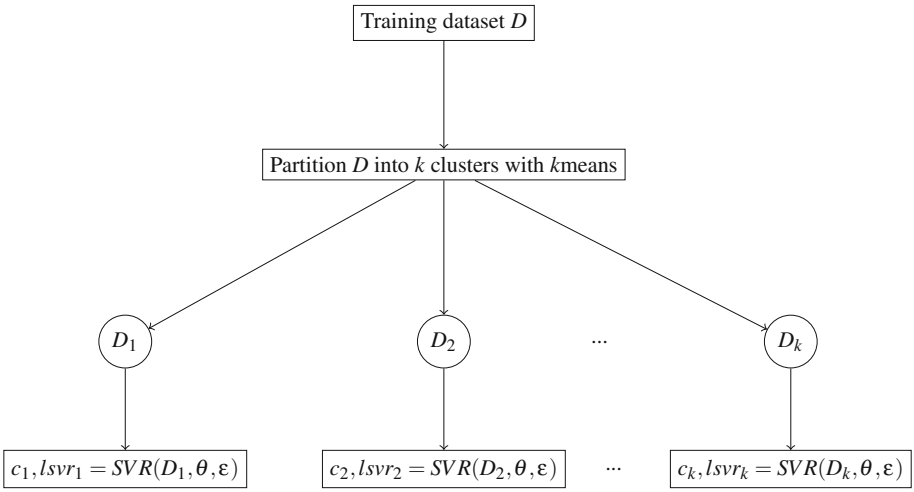

Fig. 2. Training k local SVR models (kSVR)

Learning k Local SVR Models

Our proposed kSVR algorithm learns an ensemble of local SVR models that are easily trained by the standard SVR algorithm. As illustrated in Fig. 2, the kSVR handles the regression task with two main steps. The first one uses $k$means algorithm [3] to partition the full training set into k clusters, and then the second one trains an ensemble of local SVR models in which a non-linear SVR is learnt from each cluster to predict the data locally. We consider a simplest regression task given a target variable y and a predictor (variable) x. Figure 3 shows the comparison between a global SVR model (left part) and 3 local SVR models (right part) for this regression task, using a non-linear RBF kernel function with $\gamma = 10$, a positive constant $C = 10^5$ (i.e. the hyper-parameters $\theta = \{\gamma, C\}$) and a tolerance $\epsilon = 0.05$.

Since the cluster size is smaller than the full training data size, the standard SVR can easily perform the training task on the data cluster. Furthermore, the kSVR learns independently k local models from k clusters. This is easily parallelized to take into account the benefits of high performance computing, e.g. multi-core computers or grids. The simplest development of the parallel kSVR algorithm is based on the shared memory multiprocessing programming model OpenMP [7] on multi-core computers. The parallel training of kSVR is described in Algorithm 1.

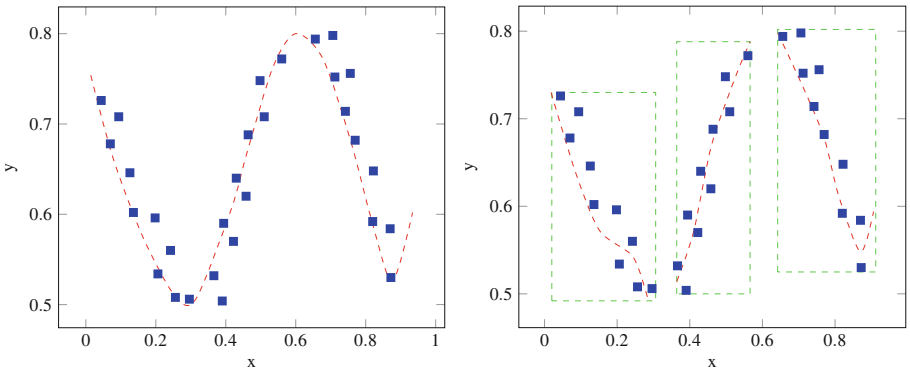

Fig. 3. Global SVR model (left part) versus 3 local SVR models (right part)

Performance Analysis

The performance analysis starts with the algorithmic complexity of building k local SVR models with the parallel kSVR algorithm. The full dataset with m datapoints is partitioned into k balanced clusters (the cluster size is about $\frac{m}{k}$). The training complexity of a local SVR is $O((\frac{m}{k})^2)$. Therefore, the algorithmic complexity of parallel training k local SVR models on a P-core processor is $O(\frac{k}{P}(\frac{m}{k})^2) = O(\frac{m^2}{kP})$. This complexity analysis illustrates that parallel learning

k local SVR models in the kSVR algorithm [1] is kP times faster than building a global SVR model (the complexity is at least $O(m^2)$).

The performance analysis in terms of the generalization capacity of such kSVR models is illustrated in [8–11]. The parameter k is used in the kSVR to give a trade-off between the generalization capacity and the computational cost. This point can be understood as follows:

- If k is large then the kSVR algorithm reduces significant training time. And then, the size of a cluster is small; The locality is extremely with a very low capacity.
- If k is small then the kSVR algorithm reduces insignificant training time. However, the size of a cluster is large; It improves the capacity.

It leads to set k so that the cluster size is a large enough (e.g. 200 proposed by [9]).

Algorithm 1. Parallel training algorithm of k local support vector regression

input :
training dataset D
number of local models k
tolerance ϵ
hyper-parameter γ of RBF kernel function
positive constant C for tuning margin and errors

output:
k local SVR models

```
1  begin
2      /*kmeans performs in the parallel way the data clustering on dataset D;*/
3      creating k clusters denoted by D_1, D_2, ..., D_k and
4      their corresponding centers c_1, c_2, ..., c_k
5      #pragma omp parallel for schedule(dynamic)
6      for i ← 1 to k do
7          /*learning local support vector regression model from D_i;*/
8          lsvr_i = svr(D_i, γ, C, ε)
9      end
10     return kSVR-model = {(c_1, lsvr_1), (c_2, lsvr_2), ..., (c_k, lsvr_k)}
11 end
```

[1] It must be noted that the complexity of the kSVR approach does not include the $k$means clustering used to partition the full dataset. But this step requires insignificant time compared with the quadratic programming solution.

Prediction of a New Datapoint x Using k Local SVR Models

The kSVR-model $= \{(c_1, lsvr_1), (c_2, lsvr_2), \ldots, (c_k, lsvr_k)\}$ is used to predict the target value of a new datapoint x as follows. The first step is to find the closest cluster based on the distance between x and the cluster centers:

$$c_{NN} = \arg\min_{c} \ distance(x, c) \tag{3}$$

And then, the target value of x is predicted by the local SVR model $lsvr_{NN}$ (corresponding to c_{NN}):

$$predict(x, kSVR\ model) = predict(x, lsvr_{NN}) \tag{4}$$

4 Evaluation

We are interested in the performance of the new parallel algorithm of local SVR (denoted by kSVR) for data regression. We have implemented the kSVR algorithm in C/C++, OpenMP [7], using the highly efficient standard library SVM, LibSVM [12]. Our evaluation of the performance is reported in terms of training time and prediction correctness. We are interested in the comparison the regression results obtained by our proposed kSVR with LibSVM.

All experiments are run on machine Linux Fedora 20, Intel(R) Core i7-4790 CPU, 3.6 GHz, 4 cores and 32 GB main memory.

Datasets

Experiments are conducted with the 5 datasets from UCI repository [4]. Table 1 presents the description of datasets. The evaluation protocols are illustrated in the last column of Table 1. Datasets are already divided in training set (Trn) and test set (Tst). We used the training data to build the SVR models. Then, we predicted the test set using the resulting models.

Tuning Parameters

We propose to use RBF kernel type in kSVR and SVR models because it is general and efficient [13]. The cross-validation protocol (2-fold) is used to tune the regression tolerance ϵ, the hyper-parameter γ of RBF kernel (RBF kernel of two individuals x_i, x_j, $K[i, j] = exp(-\gamma \|x_i - x_j\|^2)$) and the cost C (a trade-off between the margin size and the errors) to obtain a good correctness. For two largest datasets (Buzz in social media Twitter, YearPredictionMSD), we used a subset randomly sampling about 5% training dataset for tuning hyper-parameters due to the expensive computational cost. Furthermore, our kSVR uses the parameter k local models (number of clusters). We propose to set k so that each cluster consists of about 200 individuals. The idea gives a trade-off between the generalization capacity [10] and the computational cost. Table 2 presents the hyper-parameters of kSVR and SVR in the regression.

Table 1. Description of datasets

ID	Datasets	#Datapoints	#Dimensions	Target domain	Evaluation protocol
1	Appliances energy prediction	19 735	27	[10.0, 1080.0]	13 500 Trn - 6 235 Tst
2	Facebook comment volume	40 949	53	[0.0, 1 305.0]	27 500 Trn - 13 449 Tst
3	BlogFeedback	60 021	280	[0.0, 1424.0]	52 397 Trn - 7 624 Tst
4	Buzz in social media (Twitter)	583 250	77	[0.0, 75 724.5]	400 000 Trn - 183 250 Tst
5	YearPredictionMSD	515 345	90	[1 922, 2 011]	400 000 Trn - 115 345 Tst

Table 2. Hyper-parameters of kSVR and SVR

ID	Datasets	γ	C	ϵ	k
1	Appliances energy prediction	0.02	100 000	0.1	30
2	Facebook comment volume	0.001	100 000	0.1	300
3	BlogFeedback	0.4	100 000	0.05	500
4	Buzz in social media (Twitter)	0.1	100 000	0.1	4 000
5	YearPredictionMSD	0.01	100 000	0.1	1 500

Regression Results

The regression results of LibSVM and kSVR on the datasets are given in Table 3, Figs. 4, 5 and 6.

As it was expected, our kSVR algorithm outperforms LibSVM in terms of training time. The average of kSVR and LibSVM training time are 8.45 min and

Table 3. Regression results in terms of mean absolute error (MAE) and training time (minutes)

ID	Datasets	Mean absolute error (MAE)		Training time (min)	
		LibSVM	kSVR	LibSVM	kSVR
1	Appliances energy prediction	47.81	47.94	2.55	0.05
2	Facebook comment volume	8.97	8.59	27.91	0.1
3	BlogFeedback	9.85	6.40	53.78	3.86
4	Buzz in social media (Twitter)	235.25	46.73	5 193.59	31.94
5	YearPredictionMSD	8.18	7.86	2 477.91	6.33
6	Average	62.01	23.50	1 551.15	8.45

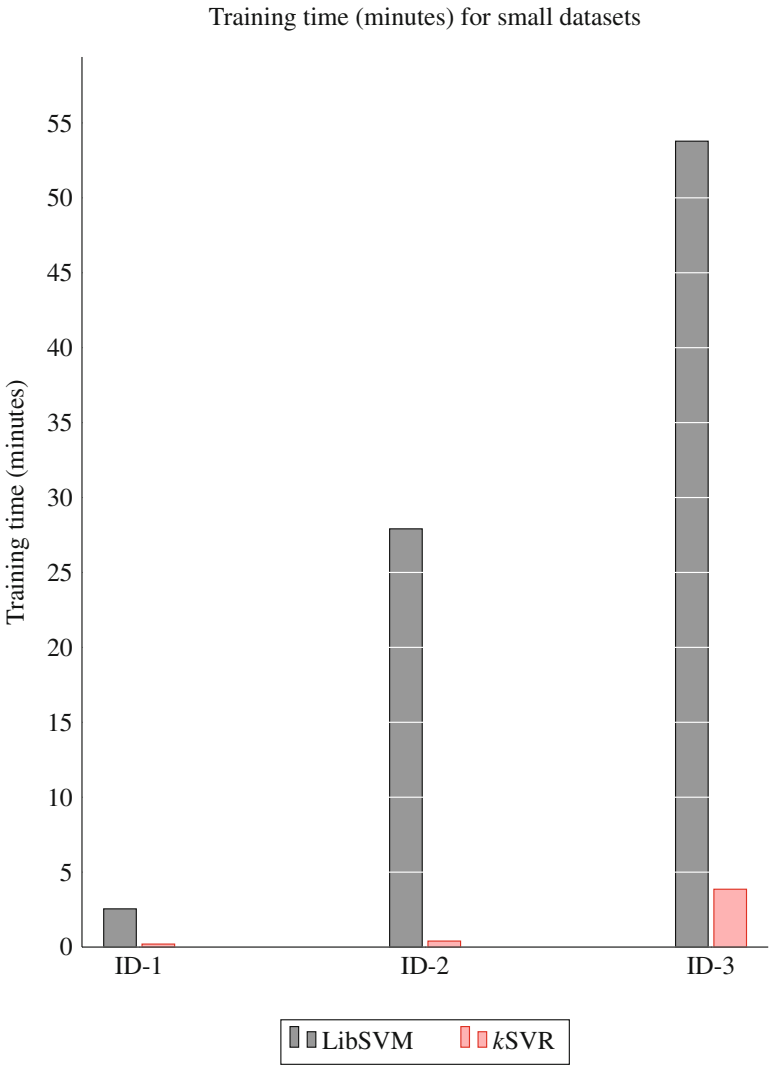

Fig. 4. Comparison of training time (minutes) for small datasets

1551.15 min, respectively. It means that our kSVR is 183.5 times faster than LibSVM.

For 3 first small datasets, the training speed improvements of kSVR versus LibSVM is 21.10 times. With two large datasets (Buzz in social media - Twitter and YearPredictionMSD), the learning time improvements of kSVR against LibSVM is more significant (about 200.44 times).

In terms of prediction correctness (mean absolute error - MAE), the error average made by kSVR and LibSVM are 23.50 and 62.01, respectively. The

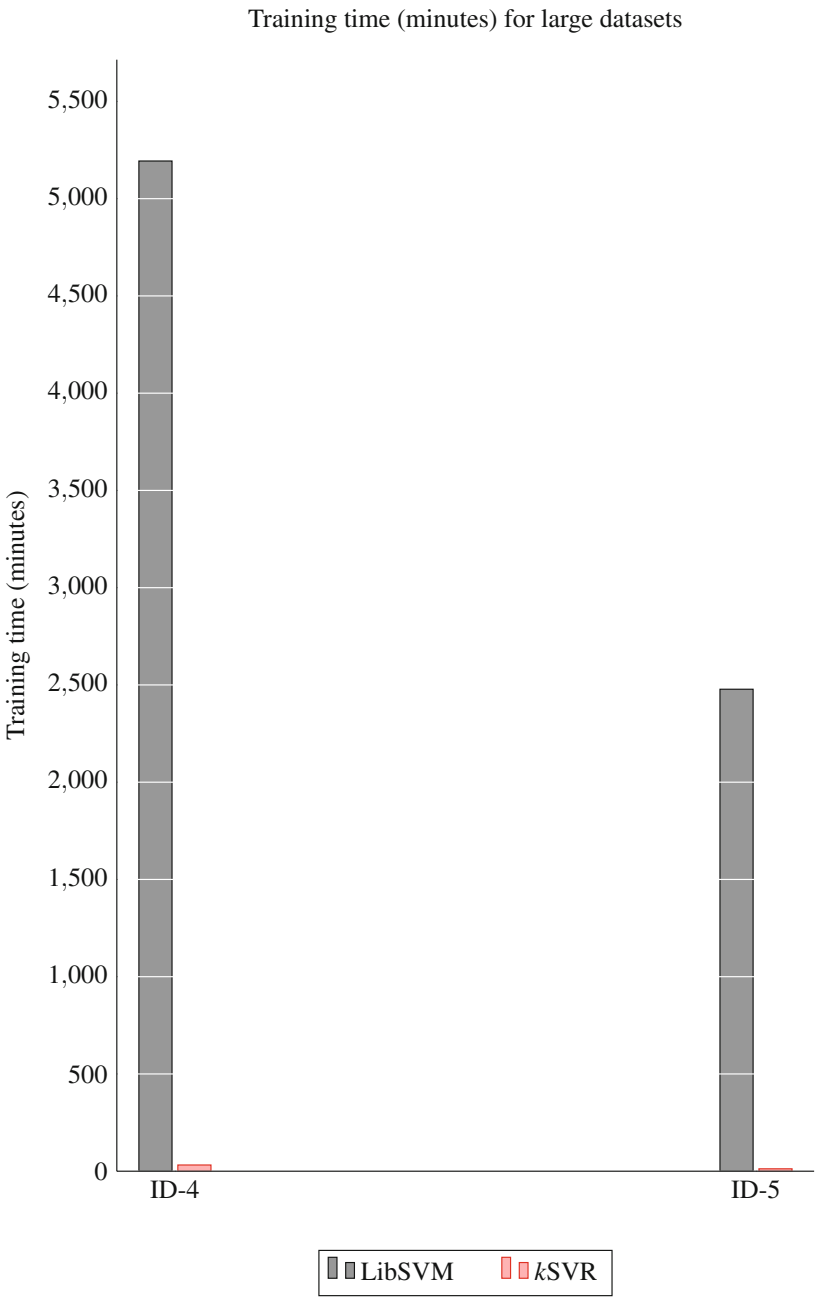

Fig. 5. Comparison of training time (minutes) for large datasets

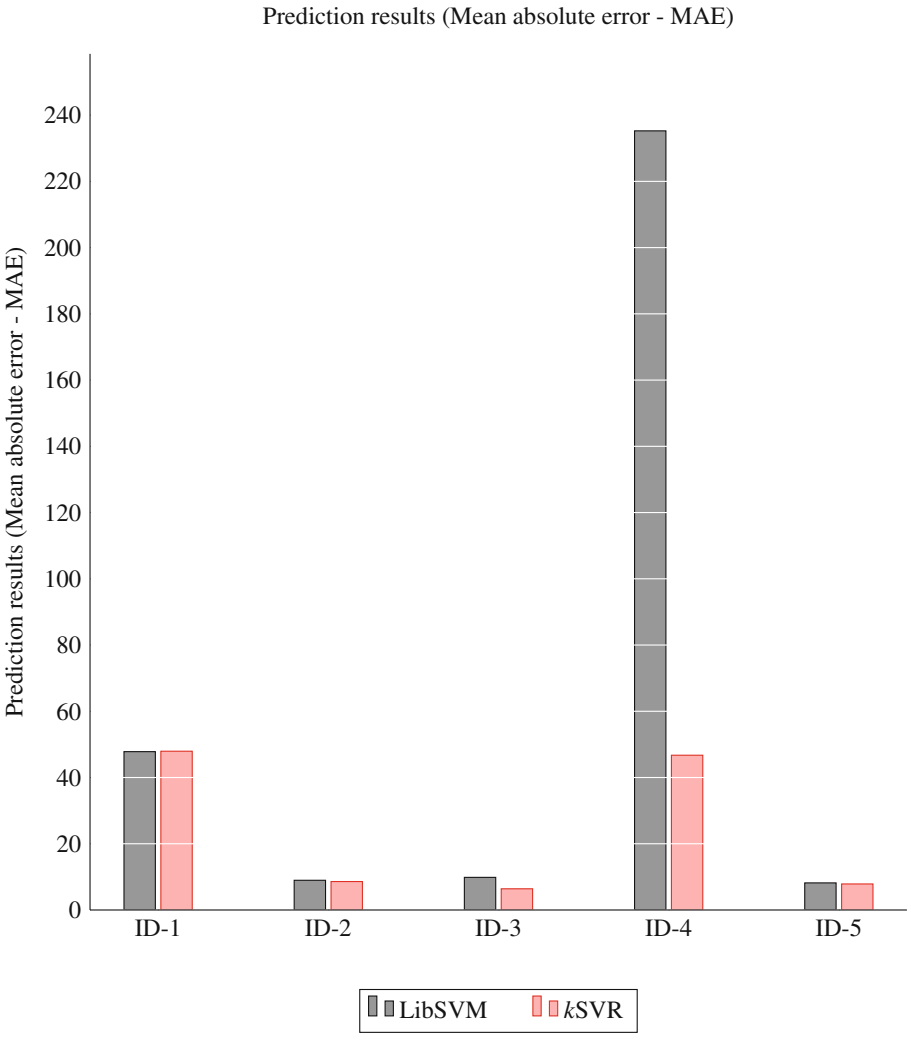

Fig. 6. Comparison of prediction results

comparison of prediction correctness, dataset by dataset, shows that kSVR is beaten only once (with Appliances energy prediction dataset) by LibSVM (4 wins, 1 defeat). It illustrates that our kSVR is more accurate than LibSVM for the prediction.

kSVR improves not only the training time, but also the prediction correctness when dealing with large datasets. The regression results allow to believe that our proposed kSVR is efficient for handling these data volumes.

5 Discussion on Related Works

Our proposal is related to large-scale SVM learning algorithms. The improvements of SVM training on very large datasets include effective heuristic methods in the decomposition of the original quadratic programming into series of small problems [6,12,14,15]. Recent works [16,17] proposed the stochastic gradient descent methods for dealing with large scale linear SVM solvers. The parallel and distributed algorithms [18–20] for the linear classification improve learning performance for large datasets by dividing the problem into sub-problems that execute on large numbers of networked PCs, grid computing, multi-core computers.

The review paper [21] provides a comprehensive survey on large-scale linear support vector classification. LIBLINEAR [22] and its extension [23] uses the Newton method for the primal-form of SVM and the coordinate descent approach for the dual-form SVM to deal with very large linear classification and regression. The parallel algorithms of logistic regression and linear SVM using Spark [24] are proposed in [25]. The distributed Newton algorithm of logistic regression [26] is implemented in the Message Passing Interface (MPI). The parallel dual coordinate descent method for linear classification [27] is implemented in multi-core environments using OpenMP. The incremental and decremental algorithms [28] aim at training linear classification of large data that cannot fit in memory. These algorithms are proposed to efficiently deal large-scale linear classification tasks in a very-high-dimensional input space. But the computational cost of a non-linear SVM approach is still impractical. The work in [29] tries to automatically determine which kernel classifiers perform strictly better than linear for a given data set.

Our proposal is in some aspects related to local SVM learning algorithms. The first approach is to classify data in hierarchical strategy. This kind of training algorithm performs the classification task with two main steps. The first one is to cluster the full dataset into homogeneous groups (clusters) and then the second one is to learn the local supervised classification models from clusters. The paper [30] proposed to use the expectation-maximization (EM) clustering algorithm [31] for partitioning the training set into k joint clusters (the EM clustering algorithm makes a soft assignment based on the posterior probabilities [32]); for each cluster, a neural network (NN) is learnt to classify the individuals in the cluster. The parallel mixture of SVMs algorithm proposed by [33] constructs local SVM models instead of NN ones in [30]. CSVM [34] uses $k$means algorithm [3] to partition the full dataset into k disjoint clusters; then the algorithm learns weighted local linear SVMs from clusters. More recent kSVM [35], krSVM [36] (random ensemble of kSVM), tSVM [11] propose to parallely train the local non-linear SVMs instead of weighting linear ones of CSVM. DTSVM [37,38] uses the decision tree algorithm [39,40] to split the full dataset into disjoint regions (tree leaves) and then the algorithm builds the local SVMs for classifying the individuals in tree leaves. These algorithms aim at speeding up the learning time.

The second approach is to learn local supervised models from k nearest neighbors (kNN) of a new testing individual. First local learning algorithm of Bottou

and Vapnik [9] find kNN of a test individual; train a neural network with only these k neighborhoods and apply the resulting network to the test individual. k-local hyperplane and convex distance nearest neighbor algorithms are also proposed in [41]. More recent local SVM algorithms aim to use the different methods for kNN retrieval; including SVM-kNN [42] trying different metrics, ALH [43] using the weighted distance and features, FaLK-SVM [44] speeding up the kNN retrieval with the cover tree [45].

A theoretical analysis for such local algorithms discussed in [8] introduces the trade-off between the capacity of learning system and the number of available individuals. The size of the neighborhoods is used as an additional free parameters to control generalisation capacity against locality of local learning algorithms.

6 Conclusion and Future Works

We presented the new parallel algorithm of local SVR that achieves high performances for the non-linear regression of large datasets. The training task of kSVR is to partition the full training dataset into k clusters. This step is to reduce data size in training local SVR. And then it easily learns k non-linear SVR models in the parallel way on multi-core computers in which a SVR model is trained in each cluster to predict the data locally. The numerical test results on datasets from UCI repository showed that our proposed kSVR is efficient in terms of training time and prediction correctness compared to the standard LibSVM. An example of its effectiveness is given with the non-linear regression of YearPredictionMSD dataset (having 400000 datapoints, 90 dimensions) in 6.33 min and 7.86 mean absolute error obtained on the prediction of the testset.

In the near future, we intend to provide more empirical test on large benchmarks and comparisons with other algorithms. A promising avenue for future research aims at improving the prediction correctness and automatically tuning hyperparameters of kSVR.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995). doi:10.1007/978-1-4757-3264-1
2. Guyon, I.: Web page on svm applications (1999). http://www.clopinet.com/isabelle/Projects/-SVM/app-list.html
3. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, vol. 1, pp. 281–297. University of California Press, January 1967
4. Lichman, M.: UCI machine learning repository (2013)
5. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods. Cambridge University Press, New York (2000)

6. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning, pp. 185–208 (1999)
7. OpenMP Architecture Review Board: OpenMP application program interface version 3.0 (2008)
8. Vapnik, V.: Principles of risk minimization for learning theory. In: Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, 2–5 December 1991], pp. 831–838 (1991)
9. Bottou, L., Vapnik, V.: Local learning algorithms. Neural Comput. **4**(6), 888–900 (1992)
10. Vapnik, V., Bottou, L.: Local algorithms for pattern recognition and dependencies estimation. Neural Comput. **5**(6), 893–909 (1993)
11. Do, T.-N., Poulet, F.: Parallel learning of local SVM algorithms for classifying large datasets. In: Hameurlain, A., Küng, J., Wagner, R., Dang, T.K., Thoai, N. (eds.) Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXI. LNCS, vol. 10140, pp. 67–93. Springer, Heidelberg (2017). doi:10.1007/978-3-662-54173-9_4
12. Chang, C.C., Lin, C.J.: LIBSVM : a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(27), 1–27 (2011)
13. Lin, C.: A practical guide to support vector classification (2003)
14. Boser, B., Guyon, I., Vapnik, V.: An training algorithm for optimal margin classifiers. In: Proceedings of 5th ACM Annual Workshop on Computational Learning Theory, pp. 144–152. ACM (1992)
15. Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In: Principe, J., Gile, L., Morgan, N., Wilson, E. (eds.) Neural Networks for Signal Processing VII, pp. 276–285 (1997)
16. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: primal estimated sub-gradient solver for SVM. In: Proceedings of the Twenty-Fourth International Conference Machine Learning, pp. 807–814. ACM (2007)
17. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 161–168. NIPS Foundation (2008). http://books.nips.cc
18. Do, T.N.: Parallel multiclass stochastic gradient descent algorithms for classifying million images with very-high-dimensional signatures into thousands classes. Vietnam J. Comput. Sci. **1**(2), 107–115 (2014)
19. Do, T.-N., Poulet, F.: Parallel multiclass logistic regression for classifying large scale image datasets. In: Le Thi, H.A., Nguyen, N.T., Do, T.V. (eds.) Advanced Computational Methods for Knowledge Engineering. AISC, vol. 358, pp. 255–266. Springer, Cham (2015). doi:10.1007/978-3-319-17996-4_23
20. Do, T.-N., Tran-Nguyen, M.-T.: Incremental parallel support vector machines for classifying large-scale multi-class image datasets. In: Dang, T.K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., Neuhold, E. (eds.) FDSE 2016. LNCS, vol. 10018, pp. 20–39. Springer, Cham (2016). doi:10.1007/978-3-319-48057-2_2
21. Yuan, G., Ho, C., Lin, C.: Recent advances of large-scale linear classification. Proc. IEEE **100**(9), 2584–2603 (2012)
22. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**(4), 1871–1874 (2008)
23. Ho, C., Lin, C.: Large-scale linear support vector regression. J. Mach. Learn. Res. **13**, 3323–3348 (2012)

24. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. HotCloud 2010, Berkeley, p. 10. USENIX Association (2010)
25. Lin, C., Tsai, C., Lee, C., Lin, C.: Large-scale logistic regression and linear support vector machines using spark. In: IEEE International Conference on Big Data, Big Data 2014, Washington, DC, 27–30 October 2014, pp. 519–528 (2014)
26. Zhuang, Y., Chin, W.-S., Juan, Y.-C., Lin, C.-J.: Distributed Newton methods for regularized logistic regression. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS, vol. 9078, pp. 690–703. Springer, Cham (2015). doi:10.1007/978-3-319-18032-8_54
27. Chiang, W., Lee, M., Lin, C.: Parallel dual coordinate descent method for large-scale linear classification in multi-core environments. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 13–17 August 2016, pp. 1485–1494 (2016)
28. Tsai, C., Lin, C., Lin, C.: Incremental and decremental training for linear classification. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, 24–27 August 2014, pp. 343–352 (2014)
29. Huang, H., Lin, C.: Linear and kernel classification: when to use which? In: Proceedings of the SIAM International Conference on Data Mining (2016)
30. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Comput. **3**(1), 79–87 (1991)
31. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. R. Stat. Soc. Ser. B **39**(1), 1–38 (1977)
32. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
33. Collobert, R., Bengio, S., Bengio, Y.: A parallel mixture of SVMs for very large scale problems. Neural Comput. **14**(5), 1105–1114 (2002)
34. Gu, Q., Han, J.: Clustered support vector machines. In: Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, 29 April–1 May 2013, vol. 31, pp. 307–315. JMLR Proceedings (2013)
35. Do, T.-N.: Non-linear classification of massive datasets with a parallel algorithm of local support vector machines. In: Le Thi, H.A., Nguyen, N.T., Do, T.V. (eds.) Advanced Computational Methods for Knowledge Engineering. AISC, vol. 358, pp. 231–241. Springer, Cham (2015). doi:10.1007/978-3-319-17996-4_21
36. Do, T.-N., Poulet, F.: Random local SVMs for classifying large datasets. In: Dang, T.K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., Neuhold, E. (eds.) FDSE 2015. LNCS, vol. 9446, pp. 3–15. Springer, Cham (2015). doi:10.1007/978-3-319-26135-5_1
37. Chang, F., Guo, C.Y., Lin, X.R., Lu, C.J.: Tree decomposition for large-scale SVM problems. J. Mach. Learn. Res. **11**, 2935–2972 (2010)
38. Chang, F., Liu, C.C.: Decision tree as an accelerator for support vector machines. In: Ding, X. (ed.) Advances in Character Recognition. InTech (2012)
39. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
40. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.: Classification and Regression Trees. Wadsworth International, Belmont (1984)

41. Vincent, P., Bengio, Y.: K-local hyperplane and convex distance nearest neighbor algorithms. In: Advances in Neural Information Processing Systems, pp. 985–992. The MIT Press (2001)
42. Zhang, H., Berg, A., Maire, M., Malik, J.: SVM-KNN: discriminative nearest neighbor classification for visual category recognition. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2126–2136 (2006)
43. Yang, T., Kecman, V.: Adaptive local hyperplane classification. Neurocomputing **71**(13–15), 3001–3004 (2008)
44. Segata, N., Blanzieri, E.: Fast and scalable local kernel machines. J. Mach. Learn. Res. **11**, 1883–1926 (2010)
45. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 97–104. ACM (2006)