Decision trees using local support vector regression models for large datasets

Minh-Thu Tran-Nguyen, Le-Diem Bui, and Thanh-Nghi Do

College of Information Technology, Can Tho University, Vietnam; UMI UMMISCO 209 (IRD/UPMC), Sorbonne University Pierre and Marie Curie University, Paris 6, France; Al Lab, Computer Science Department, Gyeongsang National University, Korea

ABSTRACT

Our proposed decision trees using local support vector regression models (tSVR, rSVR) aim to efficiently handle the regression task for large datasets. The learning algorithm tSVR of regression models is done by two main steps. The first one is to construct a decision tree regressor for partitioning the full training dataset into $k$ terminal-nodes (subsets), followed which the second one is to learn the SVR model from each terminal-node to predict the data locally in a parallel way on multi-core computers. The algorithm rSVR learns the random forest of decision trees with local SVR models for improving the prediction correctness against the tSVR model alone. The performance analysis shows that our algorithms tSVR, rSVR are efficient in terms of the algorithmic complexity and the generalization ability compared to the classical SVR. The experimental results on five large datasets from UCI repository showed that proposed tSVR and rSVR algorithms are faster than the standard SVR in training the non-linear regression model from large datasets while achieving the high correctness in the prediction. Typically, the average training time of tSVR and rSVR are 1282.66 and 482.29 times faster than the standard SVR; Furthermore, tSVR and rSVR improve 59.43%, 63.70% of the relative prediction correctness compared to the standard SVR.

ARTICLE HISTORY
Received 15 June 2018
Accepted 27 October 2019

KEYWORDS
Support vector regression (SVR); decision tree; local support vector regression (local SVR); ensemble learning; large datasets

1. Introduction

In the last decades, there is an explosion in data due to a very fast progress in computer hardware, the rapid increasing number of internet users and mobile device access to the internet, the emergence of e-commerce and huge search engines. The report of researchers from the University of Berkeley illustrates that there is about 1 Exabyte ($10^9$ Gigabyte) of data generated every year (Lyman et al., 2003). Recent book (Committee on the Analysis of Massive Data; Committee on Applied and Theoretical Statistics; Board on Mathematical Sciences and Their Applications; Division on Engineering and Physical Sciences; National Research Council, 2013) shows that Google, Yahoo!, Microsoft, Facebook, Twitter, YouTube and other internet-based companies with hundreds of millions of users and
billion daily active users collect Exabytes of data. Such a huge amount of data brings out challenges in data analysis because current state-of-the-art techniques are not well suited for that data scale. Therefore, it is a high priority to scale-up learning algorithms for addressing massive datasets. Our research focuses on creating new algorithms of decision trees using local support vector regression models to efficiently handle the non-linear regression task for large datasets.

Support vector machines (SVM) proposed by Vapnik (1995) and kernel-based methods have shown the state-of-the-art relevant to many data mining problems, including the classification, the regression and the novelty detection (Guyon, 1999). The SVM algorithm produces the most accurate models in the practice. Nevertheless, the learning problem requires the solution of a quadratic programming (QP), so that the computational cost of an SVM approach is at least square of the number of training datapoints making SVM impractical to deal with large datasets. There is a need to scale-up SVM learning algorithms to handle massive datasets.

This paper presents the extension of our work in Tran-Nguyen, Bui, Kim, Do (2018). We propose new learning algorithms of decision tree using local support vector regression (called tSVR and rtSVR) to efficiently deal with the non-linear regression task of large datasets. Instead of training a global SVR model, as done by the classical SVR algorithm is very hard to handle large datasets, our tSVR algorithm is to learn in a parallel way an ensemble of local ones that are easily trained by the standard SVR algorithms. The tSVR trains regression models via two main steps. The first one is to use the decision tree algorithm (Breiman, Friedman, Olshen, & Stone, 1984) to partition the large training dataset into $k$ terminal-nodes (subsets). The idea is to reduce the data size for training local non-linear SVR models at the second step. And then, the tSVR learns $k$ non-linear SVR models in a parallel way on multi-core computers in which an SVR model is trained in each terminal-node to predict the data locally. The extension of tSVR is the algorithm rtSVR which trains the random forest of decision trees with local prediction SVR models for improving the prediction correctness versus the tSVR model alone.

The performance analysis shows that our algorithms tSVR, rtSVR are efficient in terms of the algorithmic complexity and the generalization ability compared to the classical SVR. The experimental results on five large datasets from UCI repository (Lichman, 2013) showed that our proposed tSVR, rtSVR are faster than the standard SVR in the non-linear regression of large datasets while achieving the high prediction correctness.

The remainder of our paper is organized as follows. In Section 2, we briefly present the standard SVR algorithm. In Section 3, we illustrate how the tSVR algorithm learns local regression models from large datasets. The algorithm rtSVR of random decision trees with local prediction SVR models is presented in Section 4. The numerical test results are presented in Section 5 before discusses about related works in Section 6. We then conclude in Section 7.

2. Support vector regression

Let us consider a regression task with $m$ datapoints $x_i$ ($i = 1, \ldots, m$) in the $n$-dimensional input space $R^n$, having corresponding response values $y_i \in R$. The support vector regression (SVR) proposed by Vapnik (1995) tries to find the best hyperplane (denoted
by the normal vector \( w \in \mathbb{R}^n \) and the scalar \( b \in \mathbb{R} \) that has at most \( \epsilon \) deviation from the response value \( y_i \). Figure 1 is a simple example of SVR. The training algorithm of SVR pursues this goal with the Lagrangian dual quadratic programming (1) using the Lagrangian multipliers \( \alpha_i \) and \( \alpha_i^* \):

\[
\begin{align*}
\min & \quad \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) - \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) y_i + \epsilon \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) \\
\text{s.t.} & \quad \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0 \\
& \quad 0 \leq \alpha_i, \alpha_i^* \leq C \quad \forall i = 1, 2, \ldots, m
\end{align*}
\]

where \( C \) is a positive constant used to tune the margin size and the error and a linear kernel function \( K(x_i, x_j) = \langle x_i \cdot x_j \rangle \) (the dot product of \( x_i, x_j \)).

The resolution of the quadratic programming (1) gives \( \#SV \) support vectors for which \( \alpha_i, \alpha_i^* > 0 \). The predictive hyperplane and the scalar \( b \) are determined by these support vectors. And then, the prediction of a new datapoint \( x \) is as follows:

\[
\text{predict}(x) = \sum_{i=1}^{\#SV} (\alpha_i - \alpha_i^*)K(x, x_i) - b
\]

Variations on training SVR models use different types of the kernel (Cristianini & Shawe-Taylor, 2000). It only needs replacing the linear kernel function \( K(x_i, x_j) = \langle x_i \cdot x_j \rangle \) with other non-linear ones:

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

The SVR models are the most accurate and practical pertinent for many successful applications reported in the classification, the regression, the novelty detection (Guyon, 1999).
3. \textit{tSVR} algorithm for dealing with large datasets

Platt studied the algorithmic complexity of the SVM approach in Platt (1999). His analysis illustrates that the training complexity of the SVR solution in (1) is at least square of the number of training datapoints (i.e. $O(m^2)$), making standard SVR intractable for large datasets. It means that learning the model from the full massive dataset using the standard SVR algorithm in the usual way is a challenge due to a very high computational cost.

3.1. \textit{tSVR} for learning local SVR models

Our proposed \textit{tSVR} algorithm learns an ensemble of local SVR models that are easily trained by the standard SVR algorithm. As illustrated in Figure 2, the \textit{tSVR} handles the regression task with two main steps. The first one of the algorithm \textit{tSVR} is to learn the decision tree regressor model (using the decision tree training algorithm Breiman et al., 1984) for partitioning the full training dataset into $k$ terminal-nodes (subsets). Thus, the second step of the algorithm \textit{tSVR} is to learn $k$ non-linear SVR models in a parallel...
way on multi-core computers in which the SVR model is trained in each terminal-node to predict the data locally.

**Algorithm 1**: Parallel training algorithm \(t\)SVR\((D, \text{minobj}, \gamma, C, \epsilon)\) for local support vector regression models

**input**:
- training dataset \(D\)
- minimum number of individuals \(\text{minobj}\) for a split to be tried
- tolerance \(\epsilon\)
- hyper-parameter \(\gamma\) of RBF kernel function
- positive constant \(C\) for tuning margin and errors

**output**:
- tree \(t\) with \(k\) local SVR models

1. begin
2. /*Decision tree algorithm partitions the full dataset \(D, \text{minobj}\)*/
3. learning a tree \(t\) for splitting the full dataset \(D\) into \(k\) terminal-nodes
4. denoted by \(D_1, D_2, \ldots, D_k\) (using the early stopped parameter \(\text{minobj}\))
5. #pragma omp parallel for schedule(dynamic)
6. for \(i \leftarrow 1\) to \(k\) do
7. /*Learning local support vector regression model from \(D_i, \text{minobj}\)*/
8. \(lsvr_i = \text{svr}(D_i, \gamma, C, \epsilon)\)
9. end
10. return \(t\)SVR-model = tree \(t\) and \(\{lsvr_1, lsvr_2, \ldots, lsvr_k\}\)
11. end

We consider a simple regression task given a response variable \(y\) and a predictor (variable) \(x\). **Figure 3** shows the comparison between a global SVR model (left part) and 4 local SVR models obtained by \(t\)SVR (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 terminal-node size is smaller than the size of the full training dataset, the standard SVR can easily perform the training task from the terminal-node, due to the square reduction of the complexity compared to the learning from the full dataset. In addition, \(k\) local models in the training task of \(t\)SVR are independently learnt from \(k\) terminal-nodes. This is a nice condition to train \(k\) local models in a parallel way to take into account the benefits of high-performance computing, e.g. multi-core computers or grids. We propose to develop the parallel \(t\)SVR algorithm in a simple way based on the

![Figure 3](image-url). Global SVR model (left part) versus 4 local SVR models trained by \(t\)SVR (right part).
shared memory multiprocessing programming model OpenMP (OpenMP Architecture
Review Board, 2008) for multi-core computers. The parallel training of tSVR is described
in Algorithm 1.

3.2. Prediction of a new individual \( x \) using tSVR model

A new individual \( x \) is predicted by the tSVR model as follows. Firstly, it pushes \( x \) down
the tree \( t \) of the tSVR model from the root to the terminal-node \( D_p \). And then the local
SVR model \( lsvr_p \) learnt from this terminal-node \( D_p \) is used to predict the response value
of \( x \).

3.3. Performance analysis

Let’s consider the performance of the learning algorithm tSVR in terms of the algorithmic
complexity and the generalization capacity.

We starts with the algorithmic complexity of tSVR for building \( k \) local SVR models in the
parallel way. Suppose that tSVR uses the decision tree regressor to partition the full dataset
(with \( m \) datapoints) into \( k \) balanced terminal-nodes and then the terminal-node size is
about \( m/k \); in other words the minimum number of datapoints for a split to be tried
\( \text{minob}j = m/k \). The training complexity of a local SVR model for a terminal-node is

\[
O(\text{minob}j^2) = O\left(\frac{m^2}{k^2}\right)
\]

Therefore, the algorithmic complexity of tSVR for parallel training \( k \) local SVR models on a
\( P \)-core processor is

\[
O\left(\frac{k m^2}{P k^2}\right) = O\left(\frac{m^2}{k P}\right) = O\left(\frac{\text{minob}j m}{P}\right)
\]

This complexity analysis illustrates that parallel learning \( k \) local SVR models in the tSVR
algorithm is \( kP \) times faster than building a global SVR model by the standard SVR algor-
ithm (the complexity at least \( O(m^2) \)).

It remarks that the complexity analysis of the tSVR excepts the decision tree regressor
learnt to split the full dataset. However this training the decision tree regressor has a very
low computational cost compared with the quadratic programming solution required by
the SVR learning algorithm.

We will state how to assess the generalization capacity of tSVR models learnt by the
tSVR algorithm via the margin size of the binary support vector classification (SVC).

The regression task of the SVR is considered as a classification problem of the SVC as
illustrated in an intuitive geometric approach (Bi & Bennett, 2003). Let us consider \( x, y \)
be the predictor variable and the response variable, respectively. In the left part of
Figure 4, the training SVR algorithm tries to find the optimal plane \( (w.x - b = 0) \) that has
at most \( \epsilon \) deviation from response values \( y_i \). This task is considered as the binary classifi-
cation one illustrated in the right part of Figure 4 where the response values \( y_i \) increased
by \( \epsilon \) forms the positive class \( D^+ \) and the response values \( y_i \) decreased by \( \epsilon \) forms the
negative class \( D^- \). The optimal plane for separating \( D^+ \) from \( D^- \) found by the learning
SVC algorithm is identical to the solution of the learning SVR one. It can intuitively
assess that the largest margin solution (biggest separation boundary of two classes) gives the safest prediction model. And then the generalization ability of SVR models trained by the SVR algorithm (e.g. tSVR) can be explained in terms of the margin size of the binary SVC.

The generalization capacity of such local models is presented in Vapnik (1991), Bottou and Vapnik (1992), and Vapnik and Bottou (1993). Recently Do and his colleague (Do & Poulet, 2016b, 2017) illustrate that the learning algorithms of local SVC models give a guarantee of the generalization capacity compared to the standard SVM algorithm for the global SVC models.

To give the justification for the generalization capacity of local SVR models, we turn back Theorem 5.2 (Vapnik, 2000, p. 139). This theorem states that the large margin hyperplane has the high generalization ability. With a training set containing \( m \) datapoints being separated by the maximal margin hyperplanes, the upper bound of the expectation of the probability of test error is as follows:

\[
E_{\text{error}} \leq E \left\{ \min \left( \frac{sv}{m}, \frac{1}{m} \left[ \frac{R^2}{\Delta} \right], \frac{n}{m} \right) \right\}
\]  

(5)

where \( sv \) is the number of support vectors, \( R \) is the radius of the sphere containing the data and \( \Delta \) is the margin size, \( n \) is the number of dimensions.

It means that the generalization ability of the maximal margin hyperplane is justified in terms of

\[
\min \left( \frac{1}{m} \left[ \frac{R^2}{\Delta} \right] \right)
\]  

(6)

For the training task, the tSVR algorithm trains the decision tree regressor to split the full dataset having \( m \) datapoints into \( k \) terminal-nodes (the terminal-node size \( m_k \) is about \( \text{minobj} = m/k \)). And then the generalization capacity of local SVR models trained by
the tSVR algorithm is assessed in terms of

\[
\min \left( \frac{1}{m_k} \left[ \frac{R_k^2}{\Delta_k} \right] \right)
\]

(7)

The justification of the generalization ability is based on the comparison between Equation (6) of the global SVR model trained for the full dataset and Equation (7) of the local SVR model trained from a terminal-node. According to Theorem 1 in Do and Poulet (2016b) and Theorem 2, 3 in Do and Poulet (2017), the margin size \( \Delta_X \) of the local SVR model is greater than the margin size \( \Delta_X \) of the global one. Nevertheless, the use of the terminal-node (subset) in the training task of tSVR leads to \( R_k \leq R \) and \( m_k \leq m \). These allow concluding that there is a compromise between the locality (i.e. the terminal-node size \( \text{minobj} \) and the radius of the sphere containing the data) and the generalized ability (the margin size). Therefore, a local SVR model trained by the tSVR algorithm can endorse the prediction performance compared to the global SVR one.

The performance analysis in terms of the algorithmic complexity (4) and the generalization capacity (7) shows that the parameter \( \text{minobj} \) is used in the tSVR to give a trade-off between the generalization capacity (the prediction correctness) and the computational cost. This point can be understood as follows:

- If \( \text{minobj} \) is small then the tSVR algorithm significantly reduces training time. And then, the model learnt from such small terminal-node size has a very low capacity (i.e. the low prediction correctness).
- If \( \text{minobj} \) is large then the tSVR algorithm reduces insignificant training time. However, the size of a terminal-node is large; It improves the capacity (i.e. the high prediction correctness).

It leads to set \( \text{minobj} \) large enough (e.g. 200 proposed in Bottou & Vapnik, 1992; Vapnik & Bottou, 1993).

4. Learning algorithm of \( r \) random tSVR models for dealing with large datasets

The performance analysis can state that there is a trade-off between the algorithmic complexity and the generalization capacity in the learning algorithm tSVR. If the parameter \( \text{minobj} \) is set to small then the tSVR algorithm significantly speed up the training time but also making the low prediction correctness, compared to the learning global SVR algorithm. To address this problem, our proposed algorithm rtSVR learns the random forest of \( r \) decision trees using local prediction SVR models to improve the generalization capacity of the tSVR alone.

According to the Bias-Variance framework proposed by Breiman (1996, 2001), the performance of a learning model can be understood through Bias term and Variance term. Bias is the systematic error component and independent upon the learning sample. Variance is the error component with respect to the variability of the learning model due to the randomness of the learning sample. The ensemble-based learning algorithms (Breiman, 1996, 1998, 2001; Dietterich, 2000) aim to reduce Variance component and/or Bias component by way of using the randomization of the learning sample. And then,
the ensemble-based learning algorithm can improve the generalization capacity of the use of the single one model. Therefore, our proposed ensemble learning algorithm \( rtSVR \) trains \( r \) random \( tSVR \) models for reducing Variance. It means that the \( rtSVR \) improves the generalization ability of the \( tSVR \).

### 4.1. Learning \( r \) random \( tSVR \) models

The \( rtSVR \) algorithm trains the ensemble of \( r \) random \( tSVR \) models using the \( tSVR \) algorithm (described in Algorithm 1 and Figure 3).

The \( tSVR \) algorithm learning the \( i \)th random \( tSVR \) model from a \( i \)th bootstrap sample (sampling with replacement from the original dataset). For each non-terminal node in the training phase of the tree regressor, randomly choose \( n' \) dimensions from \( n \) original ones are used to calculate the best split among these \( n' \) dimensions. The tree regressor is grown with the early stopped \( minobj = m/k \) and without pruning. Finally, the algorithm \( tSVR \) learns \( k \) non-linear \( SVR \) models for \( k \) terminal-nodes.

As described in Algorithm 2 and Figure 5, the \( rtSVR \) constructs independently \( r \) random local \( tSVR \) models. This allows parallelizing the learning task with OpenMP on multi-core computers.

Thus, the complexity of parallel learning \( rtSVR \) on a \( P \)-core processor is as follows:

\[
O\left(\frac{r.m^2}{k.P}\right) = O\left(\frac{r.minobj.m}{P}\right) \tag{8}
\]

---

**Figure 5.** Training algorithm of \( r \) random \( tSVR \) models.
### 4.2. Prediction of a new datapoint $x$ using $r$ random tSVR models

The prediction for a new individual $x$ is the average of the prediction results obtained by $r$ random tSVR models.

```plaintext
Algorithm 2: Parallel training algorithm of $r$ random tSVR models

input:
- training dataset $D$ having $m$ individuals in $n$ dimensions
- number of tSVR models $T$
- minimum number of individuals $minobj$ for a split to be tried
- $n'$ random dimensions used for splitting data in tSVR
- tolerance $\epsilon$ for a SVR model
- hyper-parameter $\gamma$ of RBF kernel function
- positive constant $C$ for tuning margin and errors

output:
- $kr$ random tSVR models \{tSVR-1, tSVR-2, \ldots, tSVR-$r$\}

begin
  for $i \leftarrow 1$ to $r$ do
    Sampling a bootstrap $D_i$ from $D$
    Learning random tSVR model from $D_i$,*
    tSVR$_i$ = tSVR($D_i$, minobj, $\gamma$, $C$, $\epsilon$)
    for each non-terminal node, randomly choosing $n'$ dimensions from
    $n$ original ones and calculate the best split among these $n'$ dimensions
    the tree is grown with the early stopped minobj and without pruning
  end
return $r$ random tSVR models = \{tSVR-1, tSVR-2, \ldots, tSVR-$r$\}
end
```

### 5. Experimental evaluation

We are interested in the experimental performance of our decision trees (tSVR and rtSVR) using local support vector regression models for predicting large datasets. Therefore it is necessary to set up experimental tests to evaluate the performance in terms of training time and prediction correctness so that these numerical tests are consistent with the performance analysis in terms of the algorithmic complexity and the generalization ability.

#### 5.1. Software programs

Our algorithms tSVR krSVR are implemented in Python programming language, using Scikit-learn library (Machine Learning in Python Pedregosa et al., 2011), pymp package (OpenMP-like functionality for Python Lassner, 2017).

The highly efficient standard library SVM, LibSVM (Chang & Lin, 2011) with OpenMP (OpenMP Architecture Review Board, 2008) is also used to train global SVR models.

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 tSVR, rtSVR for local SVR models with LibSVM for global SVR models.

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.

#### 5.2. Datasets

Experiments are conducted with the five datasets from UCI repository (Lichman, 2013). 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.

### 5.3. Tuning parameters

We propose to use RBF kernel type in tSVR, rtSVR and LibSVM from training SVR models because it is general and efficient (Lin, 2003). The cross-validation protocol (twofold) is used to tune the regression tolerance $\epsilon$, the hyper-parameter $\gamma$ 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.

Our tSVR uses the parameter $minobj$ (the minimum number of datapoints for a split to be tried by the decision tree regression). We propose to set $minobj = 200$ (according to the advise of Vapnik & Bottou, 1993). The rtSVR algorithm is set the parameters to train 10 random tSVR models ($r=10$) with the number of randomly chosen dimensions being the square root of the full set ($n' = \sqrt{n}$ as advised by Breiman, 2001).

Table 2 presents the hyper-parameters of tSVR, rtSVR and LibSVM for training regression models.

### Table 2. Hyper-parameters of tSVR and SVR.

| ID | Datasets                        | $\gamma$ | $C$   | $\epsilon$ | minobj |
|----|--------------------------------|----------|-------|-------------|--------|
| 1  | Appliances energy prediction   | 0.02     | 100 000 | 0.1         | 200    |
| 2  | Facebook comment volume        | 0.001    | 100 000 | 0.1         | 200    |
| 3  | BlogFeedback                   | 0.4      | 100 000 | 0.05        | 200    |
| 4  | Buzz in social media (Twitter) | 0.1      | 100 000 | 0.1         | 200    |
| 5  | YearPredictionMSD              | 0.01     | 100 000 | 0.1         | 200    |

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

| ID   | Datasets                        | LibSVM Mean absolute error (MAE) | tSVR Mean absolute error (MAE) | rtSVR Mean absolute error (MAE) | LibSVM Training time (min) | tSVR Training time (min) | rtSVR Training time (min) |
|------|--------------------------------|---------------------------------|--------------------------------|--------------------------------|---------------------------|--------------------------|--------------------------|
| 1    | Appliances energy prediction   | 47.81                           | 43.56                          | 37.42                          | 2.55                      | 0.016                    | 0.15                     |
| 2    | Facebook comment volume        | 9.85                            | 23.85                          | 23.16                          | 53.78                     | 0.057                    | 0.11                     |
| 3    | BlogFeedback                   | 8.97                            | 5.31                           | 5.20                           | 27.91                     | 0.170                    | 0.40                     |
| 4    | Buzz in social media (Twitter) | 235.25                          | 45.72                          | 40.24                          | 5193.59                   | 2.392                    | 7.46                     |
| 5    | YearPredictionMSD              | 8.18                            | 7.38                           | 6.55                           | 2477.91                   | 3.297                    | 7.96                     |
| 6    | Average                        | 62.01                           | 25.16                          | 22.51                          | 1551.15                   | 1.187                    | 3.216                    |
5.4. Regression results

The regression results of LibSVM, $t$SVR and $rt$SVR on the datasets are given in Table 3 and Figures 6–8.

5.4.1. Comparison in training time

The training time in Table 3 show that our decision tree algorithms using local SVR models ($t$SVR and $rt$SVR) outperform LibSVM in terms of training time. The average of $t$SVR, $rt$SVR and LibSVM training time are 1.19, 3.22 and 1551.15 min, respectively. It means that:

- training time of $t$SVR is 1303.49 times faster than LibSVM;
- $rt$SVR is 481.72 times faster than LibSVM;
- training time of $rt$SVR is about 2.71 times longer than $t$SVR.

The comparison of training time for 3 first small datasets (Appliances energy prediction, Facebook comment volume, BlogFeedback) illustrates that the training speed improvements of $t$SVR and $rt$SVR against LibSVM are 346.67 and 127.64 times, respectively.

For two very large datasets (Buzz in social media -- Twitter and YearPredictionMSD), the learning time improvements of $t$SVR and $rt$SVR compared to LibSVM is more significant. These training speed improvements of $t$SVR and $rt$SVR versus LibSVM are 2261.39 times and 834.31 times, respectively.
5.4.2. Comparison in prediction correctness

Prediction correctness (mean absolute error – MAE) presented in Table 3 show that the error average made by \( tSVR \), \( rtSVR \) and LibSVM are 25.16, 22.51 and 62.01, respectively. The comparison of prediction correctness, dataset by dataset, shows that \( tSVR \) and \( rtSVR \) are beaten only once (with BlogFeedback dataset) by LibSVM; \( tSVR \) and \( rtSVR \) has 4 wins and 1 defeat against LibSVM. \( rtSVR \) is more accurate than \( tSVR \) with 5/5 wins. Prediction results demonstrate that our \( tSVR \), \( rtSVR \) are more efficient than LibSVM.

These numerical test results demonstrate that our \( tSVR \), \( rtSVR \) improve not only the training time, but also the prediction correctness when dealing with large datasets.

5.4.3. Comparison with decision tree algorithms

We are interested in studying the impact of using local SVR rules against the average prediction ones of classical decision trees. Therefore, we would like to compare the performance of \( tSVR \), \( rtSVR \) with the decision tree regressor (DT) and the random forest regressor (RF). The regression results of \( tSVR \), \( rtSVR \) compared to DT, RF are presented in Table 4, Figures 9 and 10. The training time of \( tSVR \) is about 2.51 times slower than DT with the
improvement of the prediction correctness being 1.21. In comparison to RF, $rtSVR$ also reduces 1.40 of the prediction error but requires 3.26 times of the training time.

The experimental results allow to believe that decision tree algorithms $tSVR$ and $rtSVR$ using local SVR models are efficient for dealing with such large datasets.

6. Discussion on related works

Our proposed $tSVR$ is in machine learning techniques related to local SVM algorithms. A theoretical analysis studied in Bottou and Vapnik (1992) and Vapnik and Bottou (1993) illustrates that there is the trade-off between the generalization capacity (the prediction correctness) and locality of local learning algorithms. The approaches of local SVMs can be categorized into two kinds of strategies (the hierarchical strategy and the nearest neighbours strategy).
The kind of local algorithms in the hierarchical strategy performs the training task with two main stages (the partition of the training set and learning local models). The first stage of learning task is to split the full training dataset into partitions (subsets) and then the second stage is to train the local supervised models from subsets. The learning algorithm proposed in Jacobs, Jordan, Nowlan, and Hinton (1991) uses the expectation–
maximization (EM) clustering algorithm (Dempster, Laird, & Rubin, 1977) to split the full training dataset into $k$ joint clusters (the EM clustering algorithm uses the posterior probabilities to make a soft cluster assignment Bishop, 2006); And then the algorithm learns neural network (NN) models from clusters to locally classify the individuals in clusters. Instead of building local NN models as done by Jacobs et al. (1991), the parallel mixture of SVMs algorithm in Collobert, Bengio, and Bengio (2002) proposes to learn local SVM models to predict the individuals in clusters. More recent, the Latent-ISVM (Do & Poulet, 2016a, 2017) uses Latent Dirichlet Allocation (LDA Blei, Ng, & Jordan, 2003) to perform the clustering task for sparse data representation. CSVM (Gu & Han, 2013), $k$SVR (Bui, Tran-Nguyen, Kim, & Do, 2017), and $k$SVM (Do, 2015) propose to uses $k$means algorithm (MacQueen, 1967) to split the full training dataset into $k$ disjoint clusters; And then $k$SVR and $k$SVM learn local non-linear SVM models in a parallel way, instead of training weighted local linear SVMs from clusters as done in CSVM. $k$rtSVR (Do & Poulet, 2015) is to learn the random ensemble of $k$SVM models. DTSVM (Chang, Guo, Lin, & Lu, 2010; Chang & Liu, 2012) and $t$SVM (Do & Poulet, 2016b) use decision tree algorithms (Breiman, Friedman, Olshen, & Stone, 1984; Quinlan, 1993) to split the full training dataset into $t$ terminal-nodes (tree leaves); follow which the $t$SVM algorithm builds local SVM models for classifying impurity terminal-nodes (with a mixture of labels) while DTSVM learns local SVM models from all tree leaves. These algorithms are shown to reduce the computational cost for dealing with large datasets while maintaining the prediction correctness.

The kind of local algorithms in the nearest neighbours strategy tries to find $k$ nearest neighbours ($k$NN) of a new testing individual from the training dataset and then it only learns local supervised model from these $k$NN to classify the new testing individual. First local learning algorithm proposed in Bottou and Vapnik (1992) and Vapnik and Bottou (1993) is to train the neural network model from $k$ neighbourhoods to predict the label of the testing individual. The investigation of Vincent and Bengio (2001) is to train $k$-local hyperplane and convex distance nearest neighbour (large margin nearest neighbours). Recent local SVM algorithms, including SVM-kNN (Zhang, Berg, Maire, & Malik, 2006), ALH (Yang & Kecman, 2008), FaLK-SVM (Segata & Blanzieri, 2010) find $k$NN of the testing individual with different techniques. SVM-kNN tries to use different metrics. ALH algorithm is to combine the weighted distance and features to predict the label of the testing individual. FaLK-SVM uses the cover tree (Beygelzimer, Kakade, & Langford, 2006) to improve the computational cost for finding the $k$NN in the feature kernel space.

7. Conclusion and future works

We presented decision tree algorithms using local SVR models that achieve high performances for the non-linear regression of large datasets. The training task of $t$SVR is to partition the full training dataset into $k$ terminal-nodes. This aims at reducing data size in training local SVR. And then it easily learns $k$ non-linear SVR models in a parallel way on multi-core computers in which an SVR model is trained in each terminal-node to predict the data locally. The algorithm $r$tSVR is to learn the random forest of decision trees using local prediction SVR models for improving the prediction correctness against the $t$SVR model alone. The performance analysis and the numerical test results on datasets from UCI repository
showed that proposed algorithms tSVR and rtSVR are efficient in terms of training time and prediction correctness compared to the standard learning algorithm LibSVM for global SVR models. The learning time improvements of tSVR and rtSVR versus LibSVM are 1282.66 and 482.29 times; Furthermore, tSVR and rtSVR also improve 59.43%, 63.70% of the relative prediction correctness compared to the standard LibSVM. An example of tSVR’s effectiveness is given with the non-linear regression of YearPredictionMSD dataset (having 400,000 datapoints, 90 dimensions) in 3.297 min and 7.38 mean absolute error obtained on the testset.

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

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Minh-Thu Tran-Nguyen was born in Vinh Long in 1980. She received her master degree in Computer Science from the University of Nantes, France in 2006 and Ph.D degree in Medical Informatics from the University of Paris 13, France in 2012. She is currently a senior lecturer, and a head of the Computer Science Department at the College of Information Technology, Cantho University, Vietnam. Her research focuses on data mining, machine learning, and recommender system.

Le-Diem Bui was born in An Giang in 1978. She got an engineering degree in Computer Science in 2000 at Can Tho University, followed by a master degree in Computer Science at Gyeongsang National University, Korea in 2005. Her research interests include fuzzy relational products, data mining in education and ICT in education. Currently, she is working at the School of Education, Can Tho University, Viet Nam and she is a PhD candidate at AI Lab, Computer Science Department, Gyeongsang National University, Korea.

Thanh-Nghi Do was born in Can Tho in 1974. He received his Ph.D./M.S. degree in Computer Science from the University of Nantes in 2004 and 2002, respectively. He is currently head of the computer networks department, and senior lecturer at the College of Information Technology, Can Tho University, Vietnam. He is also an associate researcher at UMI UMMISCO 209 (IRD/ UPMC), Sorbonne University, Pierre and Marie Curie University, France. His research interests include data mining with support vector machines, kernel-based methods, decision tree algorithms, ensemble-based learning, and information visualization. He has served on the program committees of international conferences and is a reviewer for the journals in his fields.

ORCID

Thanh-Nghi Do http://orcid.org/0000-0002-9328-3358

References

Beygelzimer, A., Kakade, S., & Langford, J. (2006). Cover trees for nearest neighbor. Proceedings of the 23rd international conference on machine learning (pp. 97–104). ACM.

Bi, J., & Bennett, K. P. (2003). A geometric approach to support vector regression. Neurocomputing, 55 (1–2), 79–108.

Bishop, C. M. (2006). Pattern recognition and machine learning. New York, NY: Springer-Verlag.
Blei, D. M., Ng, A. Y., Jordan, M. I. (2003, March). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022.

Bottou, L., & Vapnik, V. (1992). Local learning algorithms. *Neural Computation*, 4(6), 888–900.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics*, 26(3), 801–849.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. (1984). *Classification and regression trees*. Wadsworth International.

Bui, L.-D., Tran-Nguyen, M., Kim, Y.-G., & Do, T. (2017). Parallel algorithm of local support vector regression for large datasets. *Future data and security engineering – fourth international conference, FDSE 2017, HCM City, Vietnam, 2017, Proceedings* (pp. 139–153). Springer.

Chang, F., Guo, C. Y., Lin, X. R., Lu, C. J. (2010). Tree decomposition for large-scale SVM problems. *Journal of Machine Learning Research*, 11, 2935–2972.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27), 1–27.

Chang, F., & Liu, C. C. (2012). Decision tree as an accelerator for support vector machines. In X. Ding (Ed.), *Advances in character recognition*. InTech.

Collobert, R., Bengio, S., & Bengio, Y. (2002). A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5), 1105–1114.

Committee on the Analysis of Massive Data; Committee on Applied and Theoretical Statistics; Board on Mathematical Sciences and Their Applications; Division on Engineering and Physical Sciences; National Research Council (2013). *Frontiers in massive data analysis*. The National Academies Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines: And other kernel-based learning methods*. New York, NY: Cambridge University Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of The Royal Statistical Society, Series B*, 39(1), 1–38.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple classifier systems* (pp. 1–15).

Do, T. (2015). Non-linear classification of massive datasets with a parallel algorithm of local support vector machines. In *Advanced computational methods for knowledge engineering* (pp. 231–241). Springer International Publishing.

Do, T., & Poulet, F. (2015). Random local SVMS for classifying large datasets. In *Future data and security engineering – second international conference, FDSE 2015, Ho Chi Minh City, Vietnam, November 23–25, 2015, proceedings* (Vol. 9446, pp. 3–15). Springer.

Do, T., & Poulet, F. (2016a). Classifying very high-dimensional and large-scale multi-class image datasets with latent-LSVM. 2016 *intelligent IEEE conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (uic/atc/scalcom/cbdcom/iop/smartworld)* (pp. 714–721), Toulouse, France, July 18–21, 2016.

Do, T., & Poulet, F. (2016b). Parallel learning of local SVM algorithms for classifying large datasets. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, 31, 67–93.

Do, T., & Poulet, F. (2017). Latent-LSVM classification of very high-dimensional and large-scale multi-class datasets. *Concurrency and Computation: Practice and Experience*, e4224.

Gu, Q., & Han, J. (2013). Clustered support vector machines. *Proceedings of the sixteenth international conference on artificial intelligence and statistics* (Vol. 31, pp. 307–315), AISTATS 2013, Scottsdale, AZ, USA, April 29–May 1, 2013.

Guyon, I. (1999). *Web page on svm applications*. Retrieved from http://www.clopinet.com/isabelle/Projects/-SVM/app-list.html

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1), 79–87.

Lassner, C. (2017). *pymp: Easy, OpenMP style multiprocessing for Python on Unix*. Retrieved from https://github.com/classner/pymp

Lichman, M. (2013). *UCI machine learning repository*. Retrieved from http://archive.ics.uci.edu/ml

Lin, C. (2003). *A practical guide to support vector classification*. 
Lyman, P., Varian, H. R., Swearingen, K., Charles, P., Good, N., Jordan, L., & Pal, J. (2003). How much information. Retrieved from http://groups.ischool.berkeley.edu/archive/how-much-info-2003/

MacQueen, J. (1967, January). Some methods for classification and analysis of multivariate observations. Proceedings of 5th Berkeley symposium on mathematical statistics and probability (Vol. 1, 281–297). Berkeley: University of California Press.

OpenMP Architecture Review Board. (2008). OpenMP application program interface v3.0. Retrieved from http://www.openmp.org/mp-documents/spec30.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Duchesnay, E., & Vanderplas, J. (2011). Scikit-learn: machine learning in python. Journal of Machine Learning Research, 12, 2825–2830.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, & A. Smola (Eds.), Advances in kernel methods – support vector learning (pp. 185–208).

Quinlan, J. R. (1993). C4.5: Programs for machine learning. Morgan Kaufmann.

Segata, N., & Blanzieri, E. (2010). Fast and scalable local kernel machines. Journal of Machine Learning Research, 11, 1883–1926.

Tran-Nguyen, M., Bui, L., Kim, Y., & Do, T. (2018). Decision tree using local support vector regression for large datasets. Intelligent information and database systems – 10th asian conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19–21, 2018, proceedings, part I (pp. 255–265). Springer.

Vapnik, V. (1991). Principles of risk minimization for learning theory. Advances in neural information processing systems 4 [NIPS conference, Denver, Colorado, USA, December 2–5, 1991] (pp. 831–838).

Vapnik, V. (1995). The nature of statistical learning theory. Springer-Verlag.

Vapnik, V. (2000). The nature of statistical learning theory. 2nd ed. Springer-Verlag.

Vapnik, V., & Bottou, L. (1993). Local algorithms for pattern recognition and dependencies estimation. Neural Computation, 5(6), 893–909.

Vincent, P., & Bengio, Y. (2001). K-local hyperplane and convex distance nearest neighbor algorithms. In Advances in neural information processing systems (pp. 985–992). The MIT Press.

Yang, T., & Kecman, V. (2008). Adaptive local hyperplane classification. Neurocomputing, 71(13–15), 3001–3004.

Zhang, H., Berg, A., Maire, M., & Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. 2006 IEEE computer society conference on computer vision and pattern recognition (Vol. 2, pp. 2126–2136).