Effective Learning in Support Vector Machines using Bagging and Boosting

Yeboon YUN, Kansai University
Hirotaka NAKAYAMA, Konan University

Key Words: Support vector machines, Bagging, Boosting, Gaussian function, Parameter tuning

1. Introduction

In many optimization problems with unknown functions whose evaluation is expensive, it is difficult to get a precise mathematical model. In this case, generally we use a meta-model (or surrogate model) constructed based on a less number of samples. Those meta-models are improved in sequence by adding a few samples at one time (it is called a sequential approximate optimization) in order to obtain a good approximate model with as a less number of samples as possible. It has been observed that computational intelligence can be effectively applied to meta-modeling [8]. Above all, SVM (support vector machines) and RBFN (radial basis function networks) are promising for this purpose.

The performance of SVM and RBFN methods depends on values of parameters such as the width of Gaussian function and a regularization parameter. Therefore, it is important to choose appropriate values for those parameters (especially the width of Gaussian function). For this purpose, several criteria such as AIC (Akaike’s information criterion) and BIC (Bayesian information criterion) have been proposed regarding as one of model selection.

One of most popular methods for estimating parameters is cross validation (CV) test, however CV test is usually time consuming.

On the other hand, several simple formulas [3, 4, 5] have been suggested for estimating roughly the width of Gaussian function. Moreover, it has been shown that ensemble learning such as boosting and bagging has the effect on reducing the sensitivity to parameters [6].

In this research, we propose a sequential learning method using bagging and boosting in order to reduce the burden on choosing the width of Gaussian function as well as the sensitivity to it. Additionally, we show that comparing with the calculation time by a single machine, the one by the proposed learning method can be improved without losing a performance of generalization ability through several numerical experiments.

2. Support Vector Regression

In this section, we introduce SVR (support vector regression) in brief. Originally, support vector machine (SVM) was proposed by Vapnik et al [1] in the middle of 90’s, and has been recognized as one of the most effective tool for machine learning [2, 8, 9].

Later, SVM was extended to SVR for a regression problem introducing $\varepsilon$-insensitive loss function [9] in order to avoid over-learning.

Denote a training data set by $(x_i, y_i), i = 1, \ldots, \ell$, where $y_i$ is an output value for an input vector $x_i$ which is transformed to $z_i$ by some nonlinear function.

Various SVR models have been suggested. Among them, we use the following $\mu$-SVR [8]: for given parameters $\mu > 0$ and $\varepsilon \geq 0$,

\[
\minimize_{w, b, \xi, \xi'} \frac{1}{2} w^T w + \mu (\xi + \xi')
\]

subject to

\[
(w^T z_i + b) - y_i \leq \varepsilon + \xi, \quad i = 1, \ldots, \ell,
\]

\[
y_i - (w^T z_i + b) \leq \varepsilon + \xi', \quad i = 1, \ldots, \ell,
\]

\[
\xi, \xi' \geq 0,
\]

where $\mu$ is a parameter to control the amount of the error $\xi$ and $\xi'$. 

The Japan Society of Mechanical Engineers
Step 2. Set \( t := t + 1 \) and

2-1 update the width of Gaussian kernel function which is taken as \( r(t) < r(t-1) \). In this research, the following relation will be used:

\[
r(t) = 0.5r(t-1).
\]

2-2 update weights \( \alpha_i(t), i = 1, \ldots, \ell \):

\[
d_i(t) = \frac{d_i(t-1) + \epsilon_-}{\sum_{j=1}^{\ell} d_j(t-1) + \epsilon_-} \exp \left( \frac{-|y_i - \hat{f}(t-1)(x_i)|^2}{2r^2} \right),
\]

where \( \epsilon_+ = \frac{\sum_{j=1}^{\ell} d_j(t-1)}{\sum_{j=1}^{\ell} d_j(t-1) + \epsilon_-} \), and \( d_i(t-1) = \frac{1}{2} \log \left( \frac{1}{\epsilon_+ \epsilon_-} \right) \).

2-3 update a training data set \( S(t) \) and it is decomposed into several subsets \( S_k(t), k = 1, \ldots, M \), by the same way described in Step 1:

\[
S_k(t) = \{ (x_i, y_i) \mid y_i - \hat{f}(t-1)(x_i) | > \epsilon \}
\]

2-4 estimate each weak learner \( f_k(t)(x) \mid S_k(t) \), \( k = 1, \ldots, M \), by maximizing the revised \( \mu - \text{SVR} \) formulated by the following:

\[
\begin{align*}
\text{maximize} & \quad -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i' - \alpha_i)(\alpha_j' - \alpha_j)K(x_i, x_j) \\
& \quad + \sum_{i=1}^{\ell} (\alpha_i' - \alpha_i)y_i - \epsilon \sum_{i=1}^{\ell} (\alpha_i' + \alpha_i) \\
\text{subject to} & \quad \sum_{i=1}^{\ell} (\alpha_i' - \alpha_i) = 0, \\
& \quad \sum_{i=1}^{\ell} \alpha_i' \leq \mu, \\
& \quad 0 \leq \alpha_i' \leq \mu d_i(t), \quad 0 \leq \alpha_i \leq \mu d_i(t), \quad i = 1, \ldots, \ell.
\end{align*}
\]

As can be known from Eq. (7), the samples \( x_i \) with \( |y_i - \hat{f}(t-1)(x_i)| > \epsilon \) did not be approximated correctly, and thus such those samples gain a larger weight. At the next subsequent layer, it is needed to select a learner that can approximate better for the samples with large weights.

For this purpose, we revised \( \mu - \text{SVR} \) with the constraints that \( \alpha_i' \), \( \alpha_i \) is limited by the weight \( d_i(t) \). This is a parameter to control the amount of influence of the error from the viewpoint of the primal problem to the problem (9). The revised \( \mu - \text{SVR} \) can form a learner paying attention to not well-approximated samples training samples with large weights, because those samples at the preceding layer gain larger weights.
2-5 find the learner of $t$-th layer:

$$f^{(t)}(x) = f^{(t-1)}(x) + \frac{1}{M} \sum_{k=1}^{M} \xi_k^{(t)}(x) S_k^{(t)}$$ \hspace{1cm} (10)

In general, it is known that boosting (the case of the number of weak learner $M = 1$) is susceptible to noisy data and outliers, and sometimes, tends to make over-learning. Combining boosting (the first term in Eq. (10)) and bagging (the second term in Eq. (10)), it is expected that the proposed method can overcome for shortcomings and take advantages of two methods.

Step 3. Terminate if $t = T$. Otherwise, go to Step 2.

![Sequential Learning](image)

Fig. 1. Scheme of the proposed learning method

4. Experiments

Now, we will apply the proposed method to a test function and for better understanding, the following test function with a one-dimensional design variable will be used:

$$f(x) = \sin(2\pi x^4) + x, \; 0 \leq x \leq 2 \hspace{1cm} (11)$$

In this experiment, we use the parameters as follows:

- the number of training data : $\ell = 200$
- the size of $S_k^{(t)} : n = 100$
- the number of weak learner at each layer : $M = 3, 6$
- the number of layer : $T = 12$
- the initial width of Gaussian function : $\tau_0 = \frac{1}{\sqrt{2}}, \sqrt{2}$
- $\mu = 100, \varepsilon = 0.05$

For 200 training data randomly generated, we perform 10 times and the performance of generalization ability is evaluated for the test data with 400 linearly spaced points in the interval [0, 2]. The blue curve in Fig.2 (Fig.3) shows the average of RMSE (root means square error) of a single machine over several values of the width $r$ of Gauss kernel function. As can be easily known, appropriate values of $r$ are in a very small area (around $r = 0.0221$ in this problem) and also, the performance (RMSE) is sensitive to the value of the width $r$. Thus, finding such an appropriate value of the width $r$ is difficult and time consuming.

Table1 shows the averages of RMSE at each layer by the proposed method versus the initial width ($\tau_0 = \frac{1}{\sqrt{2}}, \sqrt{2}$) and the number of bagging ($# = 3, 6$). Also, RMSE is shown in Fig. 2 and Fig. 3 by red curves.

<table>
<thead>
<tr>
<th># bagging</th>
<th>3</th>
<th>6</th>
<th># bagging</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0 = 1.4142$</td>
<td>0.6181</td>
<td>0.6225</td>
<td>$r_0 = 0.7071$</td>
<td>0.6012</td>
<td>0.5984</td>
</tr>
<tr>
<td>$r_1 = 0.7071$</td>
<td>0.5996</td>
<td>0.5968</td>
<td>$r_1 = 0.3536$</td>
<td>0.5709</td>
<td>0.5688</td>
</tr>
<tr>
<td>$r_2 = 0.3536$</td>
<td>0.5669</td>
<td>0.5669</td>
<td>$r_2 = 0.1768$</td>
<td>0.5364</td>
<td>0.5324</td>
</tr>
<tr>
<td>$r_3 = 0.1768$</td>
<td>0.5379</td>
<td>0.5349</td>
<td>$r_3 = 0.0884$</td>
<td>0.5013</td>
<td>0.4817</td>
</tr>
<tr>
<td>$r_4 = 0.0884$</td>
<td>0.4911</td>
<td>0.4925</td>
<td>$r_4 = 0.0442$</td>
<td>0.4459</td>
<td>0.4310</td>
</tr>
<tr>
<td>$r_5 = 0.0442$</td>
<td>0.4425</td>
<td>0.4451</td>
<td>$r_5 = 0.0221$</td>
<td>0.3784</td>
<td>0.3630</td>
</tr>
<tr>
<td>$r_6 = 0.0221$</td>
<td>0.3703</td>
<td>0.3701</td>
<td>$r_6 = 0.0110$</td>
<td>0.3277</td>
<td>0.3143</td>
</tr>
<tr>
<td>$r_7 = 0.0110$</td>
<td>0.3174</td>
<td>0.3177</td>
<td>$r_7 = 0.0055$</td>
<td>0.3102</td>
<td>0.3083</td>
</tr>
<tr>
<td>$r_8 = 0.0055$</td>
<td>0.2995</td>
<td>0.2997</td>
<td>$r_8 = 0.0028$</td>
<td>0.3065</td>
<td>0.2946</td>
</tr>
<tr>
<td>$r_9 = 0.0028$</td>
<td>0.2956</td>
<td>0.2958</td>
<td>$r_9 = 0.0014$</td>
<td>0.3057</td>
<td>0.2939</td>
</tr>
<tr>
<td>$r_{10} = 0.0014$</td>
<td>0.2947</td>
<td>0.2951</td>
<td>$r_{10} = 0.0007$</td>
<td>0.3055</td>
<td>0.2938</td>
</tr>
<tr>
<td>$r_{11} = 0.0007$</td>
<td>0.2944</td>
<td>0.2949</td>
<td>$r_{11} = 0.0003$</td>
<td>0.3055</td>
<td>0.2938</td>
</tr>
</tbody>
</table>

Table 2. The number of training data vs. the calculation time

<table>
<thead>
<tr>
<th>calculation time</th>
<th>400</th>
<th>200</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.9 sec.</td>
<td>5.0 sec.</td>
<td>0.9 sec.</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. RMSE vs. the initial value of the width

(b) the number of bagging = 6
From these experiments, the proposed method can provide a stable good performance (RMSE converges after the layer = 10) without paying any particular attention on the initial value of \( r \) that can be given by rough estimation [3, 4, 5].

The approximate function at the final layer of the proposed method (Fig. 4) seems to be satisfactory comparing with the best result by a single machine (Fig. 5), even though the best RMSE of a single machine is better a little than the one of the proposed method.

Furthermore, Table 2 shows the relation of the calculation time versus the number of training data. As can be seen from the table, a single machine takes about 600 seconds by making CV test 10 times for finding an appropriate value of \( r \), while the proposed method needs about 9.7 seconds (the case of \# bagging number = 3 and \# layer = 12).

5. Concluding Remarks

In this research, we proposed an effective learning method using both bagging and boosting for regression problems in order to reduce the burden on choosing parameters in meta-learning as well as the computation time for learning. And we showed through several cases that the proposed method performs well without paying any particular attention to a value of the width \( r \) of Gaussian kernel function and with less computation time.

The proposed method can be applied to more complicated problems with noisy data (or outliers) and large scale data which are encountered in many practical problems.

References

Appendix

Approximate function at each layer by the proposed method:

(a) $t = 0$

(b) $t = 2$

(c) $t = 4$

(d) $t = 6$

(e) $t = 8$

(f) $t = 10$