Proceedings of the Federated Conference on Computer Science and Information Systems pp. 15–18
DOI: 10.15439/2019F362
ISSN 2300-5963 ACSIS, Vol. 18

Efficient Support Vector Regression with Reduced Training Data

Ling Cen
EBTIC, Khalifa University, UAE
cen.ling@kustar.ac.ae

Quang Hieu Vu
Zalora, Singapore
quanghieu.vu@zalora.com

Dymitr Ruta
EBTIC, Khalifa University, UAE
dymitr.ruta@kustar.ac.ae

***Abstract*—Support Vector Regression (SVR) as a supervised machine learning algorithm have gained popularity in various fields. However, the quadratic complexity of the SVR in the number of training examples prevents it from many practical applications with large training datasets. This paper aims to explore efficient ways that maximize prediction accuracy of the SVR at the minimum number of training examples. For this purpose, a clustered greedy strategy and a Genetic Algorithm (GA) based approach are proposed for optimal subset selection. The performance of the developed methods has been illustrated in the context of Clash Royale Challenge 2019, concerned with decks' win rate prediction. The training dataset with 100,000 examples were reduced to hundreds, which were fed to SVR training to maximize model prediction performance measured in validation R^2 score. Our approach achieved the second highest score among over hundred participating teams in this challenge.**

***Index Terms*—Support Vector Regression (SVR), K-means clustering, greedy search, R-squared metric, Clash Royale**

I. Introduction

Support Vector Regression (SVR) shares the same set of properties as Support Vector Machine (SVM) does for classification. Examples include tolerating some errors, characterizing hyper-plane that maximizes the margin, etc. Because of these good properties, during the past decades, SVR as well as SVM have attracted increasing interest and successfully solved supervised machine learning problems in various fields [1], [2], [3]. Its quadratic complexity in the number of training examples, however, eliminates the SVR from training on large datasets, especially if frequent retraining is required [4], [5]. High computational cost associated with the large number of support vectors is a critical drawbacks in comparison with other supervised machine learning algorithms [6], [7], [8].

To improve model efficiency, some approaches for model simplification have been proposed in the literature, e.g. eliminating support vectors linearly dependent on the other support vectors [9], selectively removing examples from training data using probabilistic estimates related to editing algorithms [10], reducing the number of support vectors using smoothed separable case approximation [8] or k-mean clustering [5], etc.

One efficient way for fast SVR training is to maximize its prediction accuracy at the minimum number of training examples. To address this challenge, a multi-step clustered greedy strategy is proposed for selecting a small data subset fed to SVR training fitted with automated robust hyper-parameter selection. Its performance has been illustrated in the context of Clash Royale Challenge 2019 with an aim to build an efficient win-rate prediction model on a relatively small subset of decks. The 100,000 labelled data examples in the training dataset were reduced to hundreds, over which a SVR model can be trained with near-maximal validation R^2 score. Our method achieved the second highest score among over hundred participating teams. In addition, a Genetic Algorithm (GA) based approach is also proposed for subset selection to explore global search in training data reduction.

The remainder of the paper is organized as follows. The Clash Royale Challenge 2019 is described in Section II. The clustered greedy selection strategy is elaborated in Section III, followed with the GA based selection approach in Section IV. The experiment results are discussed in Section V. Finally, concluding remarks are given in Section VI.

II. Competition description

Clash Royale is a popular video game, where players build decks consisting of 8 cards representing playable troops, buildings, and spells to attack opponent's towers and defend against their cards. Building good decks is, therefore, critical to win the game. The intention of the challenge is to find out whether it is possible to build an efficient win-rate prediction model on a relatively small subset of decks, whose win rates were estimated in the past.

The competition training dataset includes 100,000 decks comprising 8 cards out of the total of 90 unique possible cards, which were most commonly used by players during 3 consecutive league seasons in 1v1 ladder games, with accompanied win-rates computed over 160m games. The validation set contains 6000 randomly selected decks with their corresponding win-rates, which was extracted from the 3 next game seasons after the training data period. The testing data extracted from the same period as the validation set, which were unrevealed to participants, were used to evaluate the solutions submitted to the competition,

The task of the competition was to select 10 subsets from the 100,000 training decks, on which 10 efficient SVR models can be trained with best performance of win-rate prediction. Besides the 10 subsets, the hyper-parameter values of the SVR models with radial kernels, including ϵ, C, and γ, were required together.

The Performance of a SVR model is assessed by the R^2 metric of the model, which is defined as

$$R^2 = 1 - \frac{\sum_i (y_i - p_i)^2}{\sum_i (y_i - \frac{1}{N}\sum_i y_i)^2}, \quad (1)$$

where y_i and p_i are the true and predicted values of the win-rate of the i^{th} data point, respectively, and N is the size of the testing dataset. The score of a solution is the average of the R^2 scores of the 10 SVR models.

The facility to score derived model solutions on a part of the testing set was provided via the web-based KnowledgePit platform. Although the submission had to be evaluated for the whole testing set, the feedback in a form of the R^2 score was received based on a small subset of the testing examples, fixed for the competitors in the preliminary stage

III. Clustered Greedy Selection Strategy

The clustered greedy strategy has been developed for selecting optimal training subsets, which consists of 4 steps, i.e. k-means clustering, forward greedy search, sequence optimization, and fine-tuning process. The implementation details of the method will be elaborated in this section.

A. Data preparation

Estimation of future average win-rates for every deck are enforced to be done with the SVR model trained on the bag-of-cards represented decks and their historically computed win-rates. Given 90 unique cards the training dataset is transformed to a binary matrix with a dimension of $100k \times 90$ representing 100k (examples) by 90 (card presence indicators), while the output vector with a dimension of $100k \times 1$ contains corresponding win-rates. Similarly, the validation set (6000×90) and its corresponding outputs (6000×1) are prepared in the same way.

There is a big gap between the validation R^2 values of our submission-ready solutions and the leaderboard scores, e.g. the former is in the range of 0.4-0.5 while the latter is in 0.2-0.25. To avoid over-fitting and achieve robust models, the validation dataset are extended by combining the original validation set and training data in 4 ways, denoted as E1, E2, E3, and E4, which are:

- E1: 6000 data examples in the original validation dataset;
- E2: 6000 data examples in the original validation dataset and 6000 examples having the largest number of games in the training dataset;
- E3: all data examples in the training dataset, and 16 copies of the original validation dataset for balanced involvement of training and evaluation dataset;
- E4: removing the training points having the same decks as those in the validation set from E3 due to the big discrepancies between the two sets.

The performance of SVR models obtained during search will be evaluated on one of the 4 validation sets.

B. Hyper-parameters of SVR

The hyper-parameters of the SVR models with radial basis function (RBF) kernel, including ϵ, C, and γ, are achieved in below ways:

- C, the constraint to the alpha coefficients, is set as $C = iqr(Y)/1.349$, where $iqr(Y)$ is the inter-quartile range of the response variable, Y.
- ϵ is set to be an estimate of 0.1 of Y's standard deviation, i.e. $\epsilon = iqr(Y)/13.49$.
- γ is selected using the heuristic procedure internally implemented in MATLAB.

C. k-means clustering

The idea here is to constitute a subset with the data points distributed in the full space of training data. To achieve this, the data are firstly divided into k groups by k-means clustering. A subset is composed by selecting data points equally from each of the k clusters. Smaller k leads to high computational cost and possibly over-fitting caused by concentrated distribution of the selected data, while bigger k may overlook unique distribution of the training data. We have made a comparison on different values of k, e.g. 20, 50, and 100, from which it can be seen that $k = 50$ gives the best results.

D. Forward greedy search (FGS)

After dividing the training data into k clusters, a forward greedy algorithm is applied to select the best subset, which follows a simple strategy of adding the best possible data point from one cluster at each time. After a round is completed, in which k points have been added respectively from k clusters, a new round is started if the subset is not fully filled. The flowchart of the search process are shown in Fig. 1.

This search ensures near-optimal performance at the high computational cost of testing the addition of all remaining data points before selecting the best at each search. The advantage of our method is exhaustive evaluation is only performed on data points within a cluster, which, compared to testing all points in the full training dataset, reduces computational cost to $1/k$. In addition, such search is deterministic hence it can be implemented in parallel.

Below list compares the regression performance of the SVR models trained over the subsets selected with different values of k and using different validation sets for performance evaluation of any model yielded in search:

- $R^2 = 0.2158$, $k = 100$, validation set: E1,
- $R^2 = 0.2277$, $k = 50$, validation set: E1,
- $R^2 = 0.2566$, $k = 20$, validation set: E2,
- $R^2 = 0.2593$, $k = 50$, validation set: E2,

where R^2 is the leaderboard score received in the preliminary stage.

E. Sequence optimization (SO)

The greedy search that chooses what appears to be the optimal immediate choice at each time cannot ensure global optimal performance since the current best point may not lead to global best path. To improvement this, after 1500 training

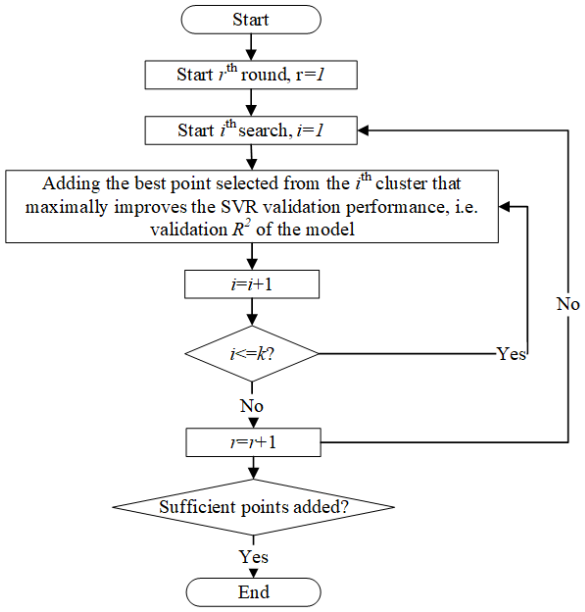

Figure 1. Flowchart of forward greedy search, where i and r denote the indices of a search and a around, respectively, i.e. the i^{th} search is to find the best point from all available remaining data in the i^{th} cluster, and a round is to find k points in k clusters respectively.

data points are selected, the sequence of the selected data points are re-arranged by starting a new round of forward greedy search within the 1500 points. Searching within a compressed set, model evaluation can be performed on E3 or E4 validation set at a much lower computational cost than exhaustive evaluation of all available points in a full clustered set. The prediction performance in the steps of FGS and SO are compared below ($k = 50$):

- $R^2 = 0.2277$ in FGS with a validation set of E1 –> $R^2 = 0.2445$ in SO with a validation set of E3;
- $R^2 = 0.2593$ in FGS with a validation set of E2 –> $R^2 = 0.2655$ in SO with a validation set of E3;
- $R^2 = 0.2593$ in FGS with a validation set of E2 –> $R^2 = 0.2702$ in SO with a validation set of E4.

After sequence optimization, the first n data points in the selected subset are the best n points of the set. From this set choosing the best 600,700,..,1500 is readily given by taking the incrementally growing chunk of the data.

F. Fine-tuning process

The final step of our selection approach is a fine-tuning process, which constitutes a new subset by combining the support vectors of the SVR model from the previous step with the training examples outside ϵ-intensive band with smaller deviation between ground truth and corresponding prediction. The improvement, however, is not always quite obvious. The solution with $R^2 = 0.2702$ can only be improved to $R^2 = 0.2703$, while some solutions achieved in previous steps can be improved a little more, e.g. the solution with $R^2 = 0.2593$ can be improved to $R^2 = 0.2606$.

IV. GA BASED SELECTION APPROACH

In addition to the main method that was presented in the previous section, we also implemented another approach using Genetic Algorithm (GA). In this section, we will present our GA based approach to select training decks.

A. Population, individual (chromosome), and gene

In GA, at any point of time, there is a population consisting of individuals each of which is a possible solution that includes ten different sets of training decks together with the three required parameters to train an SVM: ϵ, C, and γ. In other words, an individual in our GA population is a possible solution or submission to the competition.

Given the above definition for an individual in GA, we can see that there are a couple of ways to define a gene in the individual (as an individual is a chromosome that contains a set of genes).

- A possible definition is to consider each training set of deck indices together with the parameters ϵ, C, and γ as a gene. In this way, we have exactly 10 genes from 10 training sets of decks in each individual. This definition, however, has an issue as the gene is too big to efficiently and effectively perform different variation operations.
- Instead of applying the above definition, the smallest unit of the individual is considered as a gene in which a gene could be a specific ϵ, C, and γ to train a model with a set of training decks, or even a training deck in this training set. While this definition gives us a finer granulation for the gene, it requires some tricks to support crossover and mutation that we will discuss later to make generation evolve.

B. First generation

As in a typical approach, the first generation of GA should be generated randomly.

- A random ϵ in the range: 0.0 to 1.0
- A random C in the range: 0.0 to 1000.0
- A random γ in the range: 0.0 to 10.0
- A random set of indices in the range: 1.0 to 100000.0

However, in order to help the GA involve faster, in addition to randomly generated individuals, we also employ few simple approaches to get some seed (good) individuals for the first generation. Note that these approaches are only used to select (possible) better training decks (indices). For ϵ, C, and γ, we use default values (specifically, $\epsilon = 0.1$, $C = 1.0$, and $\gamma = 1.0/90$). Training deck indices were generated for the seed individuals in the following approaches

- Using indices from decks having the highest number of games in the training data.
- Using indices from decks having the highest number of players in the training data.
- Using k-means algorithm to cluster training data into different groups (e.g. 60 -> 150) and again selecting the top-10 indices from each group having the highest number of games or the highest number of players.

C. Fitness measurement

As each individual in our GA is a possible solution or submission, the straightforward fitness score is the prediction score of the validation data using model trained by parameters and indexed data specified in the individual. In addition to this fitness measurement, another way is to evaluate the model using both training and validation data sets (using different way to give higher weights to the validation data then the training data – as ultimately, we still need to mainly rely on the validation data set). While this validation seems to be better to avoid over-fitting, the trade-off, however, is that it takes significantly more time for the evaluation as the model needs to be evaluated for a much bigger set of data.

D. Mutation

Given an individual, we first randomly select a set of training decks for the mutation. Then, we will choose to change one of the following components:

- Changing ϵ to a random number between 0.0 and 1.0, changing C to a random number between 0.0 and 1000.0, and changing γ to a random number between 0.0 and 10.0 all with a grid of 10^{-6}.
- Initially, we randomly selected a single training deck to be replaced by another one outside the training set. However, this approach makes the GA extremely slow in progress. Thus, instead of selecting one training deck, to make the GA evolve faster, we chose to randomly select 5% of training decks from the existing indices for replacement.

Note that in each generation, we randomly select 25% of the population to apply mutation for generating new individuals.

E. Crossover

Given a pair of individuals, we first randomly select a set of training decks for crossover. Then, we choose to perform crossover in the following components:

- Choosing an ϵ, C, γ independently from a randomly selected individual.
- We first randomly select a training set of indices having the same size from both individual (e.g. training set of 1000 indices from both sides). Then, from the two training set of indices, we select half of them from each individual. Note that there could be overlapping in the selected indices from both individual, and hence generate less than the number of required indices. In this case, we have two ways to fill the missing indices: to continuously select indices from two individuals to fill or randomly select new indices from outside to fill. In our approach, we randomly use one of the two methods.

Note that in each generation, we randomly select 50% of the population to apply crossover for generating new individuals.

F. Selection

We follow the traditional approach to select individuals from a generation to the next one. Basically, the probability of an individual to be selected is proportional to the fitness score it has. It means that the stronger (higher fitness score) of an individual, the higher chance it is being selected to be in the next generation. In our implementation, we choose to maintain a population of 50 individuals in each generation.

V. Experiment Results

By applying the proposed methods, the best solution was achieved by clustered greedy selection with below settings:

- $k = 50$ in clustering,
- validation set: E2 in the step of FGS,
- validation set: E4 in the step of SO.

Its leaderboard R^2 score is 0.2593 from forward greedy search and improved to 0.2703 via sequence optimization and fine-tuning. The SO contributes most to score improvement from 0.2593 to 0.2702. The final score evaluated on the full testing dataset is 0.253017. Both the preliminary and final scores of the solution are the 2^{nd} highest among over hundred participating teams, showing robustness of the method against over-fitting.

VI. Conclusions

This paper explores the possibility of training a Support Vector Regression (SVR) model using a minimal number of training data samples. Two approaches, i.e. clustered greedy strategy, and Genetic Algorithm (GA) based method, are proposed for the selection of data subset fed to SVR training to maximize validation performance. The details of the implementation are elaborated in the paper. The proposed methods successfully selected hundreds of points from 100,000 labeled data samples for efficient SVR training in decks' win-rate prediction and scored 2^{nd} place among over hundred participating teams in the Clash Royale Challenge 2019.

References

[1] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," *Proc. Fifth Annual Workshop of Computational Learning Theory*, vol. 5, pp. 144–152, Pittsburgh, 1992.

[2] V. Vapnik, "The Nature of Statistical Learning Theory," Springer, New York, 1995.

[3] V. Vapnik, S. Golowich and A. Smola, "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing," in M. Mozer, M. Jordan, and T. Petsche (eds.), Neural Information Processing Systems, vol. 9, MIT Press, Cambridge, MA., 1997.

[4] A. Smola, and B. Schölkopf, "A Tutorial on Support Vector Regression," *Statistics and computing*, vol. 14, pp. 199-222, 2003.

[5] X. Xia,M. Lyu, T. Lok, G. Huang, "Methods of Decreasing the Number of Support Vectors via k-Mean Clustering," *Proc. Int. Conf. Intelligent Computing*, pp. 717-726, 2005.

[6] C. Burges, "Simplified support vector decision rules," Proc. 13th Int. Conf. Mach. Learning, pp. 71-77, 1996.

[7] E. Osuna and F. Girosi, "Reducing the run-time complexity of support vector machines," Int. Conf. Pattern Recognition, Australia, 1998.

[8] D. Geebelen, J. Suykens, J. Vandewalle, "Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation," *IEEE Trans Neural Netw Learn Syst.*, vol. 23, no. 4, pp. 682-688, 2012.

[9] T. Downs, K. Gates, and A. Masters, "Exact simplification of support vector solutions," Journal of Machine Learning Reseaerch, vol. 1, pp. 293-297, 2001.

[10] G. Bakir, J. Weston, and L. Bottou, "Breaking SVM complexity with cross-training," Advances in Neural Information Processing Systems, vol. 17, pp. 81-88, 2005.