Expert Systems with Applications 41 (2014) 4989–4995

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Approximating support vector machine with artificial neural network for fast prediction

Seokho Kang, Sungzoon Cho [*]

Department of Industrial Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-742, Republic of Korea

A R T I C L E I N F O

Keywords:
Support vector machine
Artificial neural network
Hybrid neural network
Approximation
Run-time speed

A B S T R A C T

Support vector machine (SVM) is a powerful algorithm for classification and regression problems and is widely applied to real-world applications. However, its high computational load in the test phase makes it difficult to use in practice. In this paper, we propose hybrid neural network (HNN), a method to accelerate an SVM in the test phase by approximating the SVM. The proposed method approximates the SVM using an artificial neural network (ANN). The resulting regression function of the ANN replaces the decision function or the regression function of the SVM. Since the prediction of the ANN requires significantly less computation than that of the SVM, the proposed method yields faster test speed. The proposed method is evaluated by experiments on real-world benchmark datasets. Experimental results show that the proposed method successfully accelerates SVM in the test phase with little or no prediction loss.

1. Introduction

Support vector machine (SVM), based on the structural risk minimization principle, is one of the most widely used learning algorithms, and achieves superior generalization performance for both classification and regression problems (Burges, 1998; Smola & Schölkopf, 2004; Vapnik, 1995, 1998). SVM has been successfully applied to various applications, including text categorization, handwritten digit recognition, and financial forecasting (Burges, 1998; Smola & Schölkopf, 2004). Its major drawback is low speed in the training and test phases. To overcome this, several studies have been conducted using various approaches.

In the training phase, the SVM solves a quadratic programming (QP) problem with computational complexity $\mathcal{O}(N^3)$, where N is the number of training datapoints. Commonly used approaches are improving the efficiency of QP (Joachims, 1999; Platt, 1998, 1999). Other approaches such as eliminating non-support vectors early, before or during the QP process, have also been proposed (Kim & Cho, 2012; Shin & Cho, 2007). They successfully reduce the training time of the SVM, and widely used nowadays.

The majority of studies focus on accelerating the SVM in the training phase. Run-time speed of the trained SVM, however, is much more important in real-world deployments. In most cases, training is carried out before applying the SVM, and is often not subject to a time limit. On the other hand, the SVM works under time constraints when deployed to make predictions on new datapoints, hence, fast run-time speed is needed. Moreover, the importance of run-time speed is more strengthened when the SVM have to predict for real-time stream data. This problem can be interpreted as acceleration of the SVM in the test phase.

For the test phase, the SVM takes $\mathcal{O}(N_{SV})$ of its computational complexity, where N_{SV} is the number of support vectors. By alleviating its complexity, the usability of the SVM for real-time applications would be enhanced. To lessen the computational burden of the SVM in the test phase, many researchers have focused on reducing the number of support vectors directly (Burges, 1996; Downs, Gates, & Masters, 2002; Li, Zhang, & Lin, 2006; Li, Jiao, & Hao, 2007; Liang et al., 2013; Lin & Yeh, 2009), and approximating kernel functions (Maji, Berg, & Malik, 2008, 2013; Vedaldi & Zisserman, 2012).

These efforts just tried to modify the SVM partially, whereupon they were limited to the structural characteristics of the SVM. In order to obtain better results, hybrid approaches, which put together with different models, can be taken into account.

There are two major types of hybrid approaches to increase run-time speed of the complex algorithms. The first approach is to use a simple, fast model, which roughly processes the test datapoints. Only uncertain or crucial datapoints are processed by this complex, high-performance model. This approach reduces the number of test datapoints directly processed by the complex model to lessen its test complexity. Several studies employed an SVM as a complex model. Xu, Zhang, and Zhong (2005) employed a rocchio classifier as a simple model for text categorization problems. Kumar and

* Corresponding author. Tel.: +82 2 880 6275.
E-mail addresses: prokids@snu.ac.kr (S. Kang), zoon@snu.ac.kr (S. Cho).

http://dx.doi.org/10.1016/j.eswa.2014.02.025
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

Gopal (2010) and Ji and Zhao (2013) used decision tree (DT) and k-nearest-neighbor (k-NN), respectively, for the binary classification problems. This approach successfully reduced the test complexity for classification problems, but was not applicable for regression problems.

The second approach is to have a simple, fast model that replicates a complex model. This approach can be thought of as a function approximation problem. Schmitz, Aldrich, and Gouws (1999) presented a DT-approximation of an artificial neural network (ANN) to yield fast test speed and high interpretability with little prediction loss. Chen and Chen (2004) used multiple simple classifiers to approximate an SVM for classification problems. Applying this approach to an SVM would be beneficial to improve the test speed of the SVM.

In this paper, we propose hybrid neural network (HNN), which is an ANN regression-based approximation method for two SVM algorithms, support vector classification (SVC) and support vector regression (SVR). The proposed method follows the second hybrid approach and seeks to approximate an SVM using an ANN to achieve a high run-time speed without sacrificing prediction accuracy. For a binary classification or regression problem, the SVC or SVR is simply replaced by a single-output ANN. For a multi-class classification problem, we adopt a multiple-output ANN whose output nodes correspond to the several decision functions of the multi-class SVC.

The test phases of an ANN and SVM are similar in computation. The number of hidden nodes of an ANN is comparable with the number of support vectors of an SVM. However, the number of hidden nodes is generally much smaller than the number of support vectors for a dataset. Thus, an ANN is even faster in the test phase than an SVM. By employing an ANN as an approximation of an SVM, we can yield similar predictions with less computational burden. In order to verify the effectiveness of the proposed method, we conducted experiments on several benchmark datasets.

The remainder of this paper is organized as follows. Section 2 contains a review of SVC, SVR, and ANN. In Section 3, we present our proposed method, and Section 4 reports experimental results. Conclusion and future work are presented in Section 5.

2. Backgrounds

In this section, we review the foundations of the proposed method, support vector classification (SVC), support vector regression (SVR), and artificial neural network (ANN). For further details, refer to Bishop and Nasrabadi (2006), Burges (1998), and Smola and Schölkopf (2004).

2.1. Support vector classification (SVC)

We first describe a standard SVC algorithm. Consider a set of N training datapoints $\mathfrak{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where $\boldsymbol{x}_i$ is the i-th input feature vector and $y_i \in \{-1, 1\}$ is the corresponding target class label. SVC aims to find the maximum margin hyperplane $\mathbf{w}^T\varphi(\boldsymbol{x}) + b = 0$ that separates the positive datapoints from the negative datapoints. This can be formulated into the following primal optimization problem:

$$\begin{aligned} \underset{\mathbf{w},b,\xi_i}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\varphi(\boldsymbol{x}_i) + b) \geqslant 1 - \xi_i, \\ & \xi_i \geqslant 0, i = 1, \ldots, N, \end{aligned} \tag{1}$$

where $C > 0$ is the parameter determining the tradeoff between the training errors and the model complexity, ξ_i are the slack variables, and φ is a non-linear mapping from an input space into a feature space. By introducing the Lagrangian function for the primal problem, we derive its dual problem into the following quadratic programming (QP) problem:

$$\begin{aligned} \underset{\alpha_i}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_i \alpha_i \\ \text{subject to} \quad & \sum\nolimits_i \alpha_i y_i = 0, \\ & 0 \leqslant \alpha_i \leqslant C, i = 1, \ldots, N, \end{aligned} \tag{2}$$

where α_i are Lagrange multipliers and k is a kernel function. One of the most popular kernel functions is the radial basis function (RBF) kernel function which is $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$. Once the dual QP problem is solved, the resulting decision function is:

$$f(\boldsymbol{x}) = \mathbf{w}^T\varphi(\boldsymbol{x}) + b = \sum_{i=1}^N \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b = \sum_{i \in SV} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{3}$$

The decision function is described by only training datapoints with only nonzero α_i. These datapoints are called support vectors (SVs). In the test phase, we decide the class label of the test datapoint $\boldsymbol{x}$ to be $sign(f(\boldsymbol{x}))$.

Note that SVC is originally a binary classification algorithm. To extend SVC for multi-class classification problems, several approaches such as one-against-one and one-against-rest can be applied (Hsu & Lin, 2002).

2.2. Support vector regression (SVR)

In SVR, we would like to find a regression function $f(x) = \mathbf{w}^T\varphi(x) + b$ that fits the training datapoints. Given $\mathfrak{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where $\boldsymbol{x}_i$ is an input feature vector and $y_i \in \mathbb{R}$ is its target value to be estimated by the regression function, a standard SVR is formulated as follows:

$$\begin{aligned} \underset{\mathbf{w},b,\xi_i,\xi_i^*}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_i \xi_i + \sum_i \xi_i^*\right) \\ \text{subject to} \quad & y_i - (\mathbf{w}^T\varphi(\boldsymbol{x}_i) + b) \leqslant \epsilon + \xi_i, \\ & (\mathbf{w}^T\varphi(\boldsymbol{x}_i) + b) - y_i \leqslant \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geqslant 0, i = 1, \ldots, N, \end{aligned} \tag{4}$$

where C determines the tradeoff between the training error and the flatness, ϵ is the size of the ϵ-insensitive tube, and xi_i, xi_i^* are the slack variables that are zero when the corresponding training datapoint is inside the ϵ-insensitive tube. Similar to the SVC, the primal problem can be transformed to a dual QP problem. The QP problem is:

$$\begin{aligned} \underset{\alpha_i,\alpha_i^*}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & -\epsilon\sum_i(\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \quad & \sum_i(\alpha_i - \alpha_i^*) = 0, \\ & 0 \leqslant \alpha_i, \alpha_i^* \leqslant C, i = 1, \ldots, N, \end{aligned} \tag{5}$$

where α_i are Lagrange multipliers and k is a kernel function. As with the SVC, the RBF kernel function is commonly used. After solving the problem, we can obtain the following regression function:

$$\begin{aligned} f(\boldsymbol{x}) = \mathbf{w}^T\varphi(\boldsymbol{x}) + b &= \sum_{i=1}^N(\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}) + b \\ &= \sum_{i \in SV}(\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}) + b. \end{aligned} \tag{6}$$

As before, only those datapoints for which $\alpha_i > 0$ or $\alpha_i^* > 0$ are the SVs, and they define the regression function.

2.3. Artificial neural network (ANN)

In this subsection, we introduce multilayer perceptron (MLP) architecture for regression among various types of ANN. An ANN consists of three layers, each of which contains several processing units: the input layer, the hidden layer, and the output layer. The nodes of the input layer take input features and are then followed by one or more hidden layers. The final output layer produces the regression function values. The regression function f of an ANN is based on the linear combination of nonlinear basis functions h_i in the form:

$$f(\boldsymbol{x}) = w_0 + \sum_i w_i h_i(\boldsymbol{x}) \tag{7}$$

where w_i are weights. Each h_i corresponds to the i-th hidden node that exists in a hidden layer, and they are generally chosen to be sigmoidal functions. The following is the logistic sigmoid function:

$$h_i(\boldsymbol{x}) = \frac{1}{1 + exp\left(w_{i,0} + \sum_{j=1}^{p} w_{i,j} x^j\right)} \tag{8}$$

where x^j are the j-th input variables, and $w_{i,j}$ are weights. Given a training set $\mathfrak{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N, y_i \in \mathbb{R}$, we would like to minimize the error function:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N} \|f(\boldsymbol{x}_i) - y_i\|^2 \tag{9}$$

In order to derive optimal weights for the ANN, a back propagation algorithm is generally employed. A back propagation algorithm updates the weights iteratively to minimize the error function. After training, the output of a test datapoint $\boldsymbol{x}$ can be estimated by the regression function with the weights.

3. The proposed method

In this section, we analyze SVM and ANN in the test phase, and present hybrid neural network (HNN) which is a approximation method to accelerate SVM in the test phase.

3.1. Analysis of SVM and ANN in the test phase

In the test phase, SVM takes $\mathcal{O}(N_{SV})$ of its computational complexity. The support vectors are chosen among training datapoints in the training process. For an SVC, a set of support vectors determines the decision boundary in the feature space. For an SVR, the datapoints located outside the ϵ-insensitive tube will be the support vectors. Thus, the set of support vectors tends to grow as the size of the dataset increases. In some cases, almost the whole training datapoints can be the support vectors.

The computational complexity of ANN in the test phase is $\mathcal{O}(N_{HN})$, where N_{HN} is the number of hidden nodes. Unlike SVM, the number of hidden nodes of an ANN highly depends on the nature of the dataset. For this reason, the number of support vectors of an SVM is generally smaller than that of hidden nodes of an ANN for a dataset. Moreover, more gaps appear as the dataset grows.

The major operations of the SVM and ANN in the test phase are the kernel and sigmoidal functions computation, respectively. Considering the RBF kernel as the kernel function of the SVM and logistic sigmoid function as the sigmoidal function of the ANN, these two computations are similar. Thereby, ANN is more advantageous than SVM in terms of the amount of computation. Especially in larger datasets, ANN is much faster in the test phase.

3.2. Hybrid neural network (HNN)

In subsection 3.1, the reason that the ANN is faster than the SVM in the test phase is explained. In this subsection, we introduce a method, named HNN, to accelerate the SVM based on an ANN. It is known that an ANN, on its own, shows good fitting performance for regression problems. In addition, the fitting performance of the regression algorithm for a function is better than that for a noisy dataset. Utilizing these points, we replace the decision or regression function of the trained SVM with the regression function of the ANN that fits the function values of the SVM. That is, the SVM plays a role in removing noise from the dataset, and the ANN is trained for the refined dataset. The prediction accuracy of the trained ANN is the same as that of the SVM, but it is able to predict new datapoints faster.

The HNN method consists of two training phases as shown in Fig. 1. In the first training phase, an SVM is trained with the given original training dataset $\mathfrak{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where $\boldsymbol{x}_i$ is a input feature vector and y_i is its corresponding target value. Next, we obtain the function values $f(\boldsymbol{x}_i) \in \mathbb{R}$ for each datapoint $\boldsymbol{x}_i$ using the decision or regression function of the trained SVM. In the second training phase of the HNN, a new refined training dataset $\mathfrak{D}' = \{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1}^N$ is constructed by the input feature vectors and the function values. Then an ANN is trained for $\mathfrak{D}'$ to fit the function values. Finally, the regression function of the trained ANN is used for testing new datapoints. That is, the decision function of the SVC or the regression function of the SVR is substituted by the regression function of the ANN.

We first introduce the proposed HNN for classification problems. For a binary classification problem, a single-output ANN is sufficient while a multiple-output ANN, whose output nodes are exactly matched with decision functions of a multi-class SVC, is used for a multi-class classification problem. The HNN method for classification is presented in Algorithm 1.

Algorithm 1. HNN for classification

Given a training dataset $\mathfrak{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N, y_i \in \{1, \dots, c\}$, the detailed procedure for the proposed method is as follows.

(Step 1 – Train the SVC for the dataset $\mathfrak{D}$.)If c is equal to 2, a binary SVC is trained with $\mathfrak{D}$. If c is greater than 2, a multi-class SVC is trained following the selected multi-class classification approach. After this, several decision functions $f_j \in \mathbb{R}$ of the SVC are obtained.

(Step 2 – Construct a new training dataset $\mathfrak{D}'$ using the decision functions of the SVC.) In this step, we compute the decision function values $f_j(\boldsymbol{x}_i)$ for the original training dataset $\mathfrak{D}$. After this, a new refined training dataset $\mathfrak{D}'$ is defined as $\mathfrak{D}' = \{\boldsymbol{x}_i, \{f_1(\boldsymbol{x}_i), \dots, f_c(\boldsymbol{x}_i)\}\}_{i=1}^N$

(Step 3 – Train the ANN for the dataset $\mathfrak{D}'$.) A multiple-output ANN is trained with the $\mathfrak{D}'$. The j-th output node of the ANN corresponds to the j-th decision function f_j. In addition, the number of output nodes of the ANN is equal to the number of decision functions.

(Step 4 – Classify the test datapoints using the ANN.) The values for the decision functions of the multi-class SVC are approximated by the regression functions using the ANN. The final classification results are obtained based on the strategy taken by the multi-class SVC.

In Algorithm 1, several decision functions of the multi-class SVC are integrated into an ANN. Fig. 2 shows the example network diagram of the HNN for a 3-class classification problem based on the one-against-one approach. For a multi-class SVC, each decision

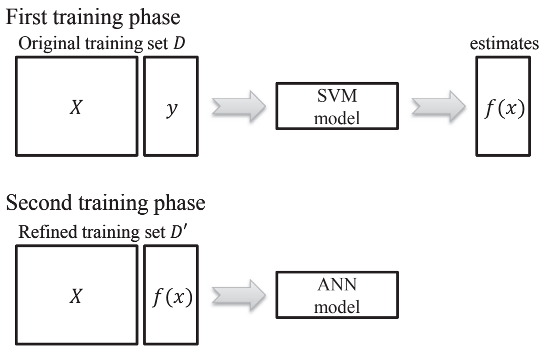

Fig. 1. Framework of the proposed method.

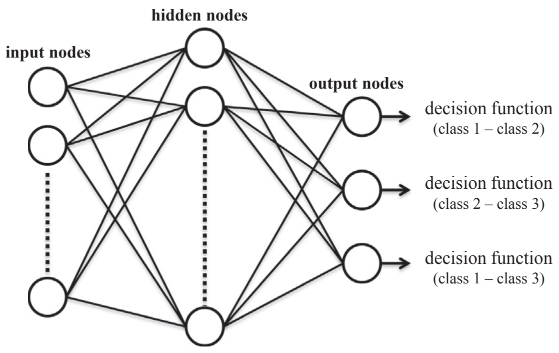
hidden nodes
input nodes
output nodes
decision function (class 1 – class 2)
decision function (class 2 – class 3)
decision function (class 1 – class 3)

Fig. 2. Network diagram of HNN for 3-class classification problem based on one-against-one approach.

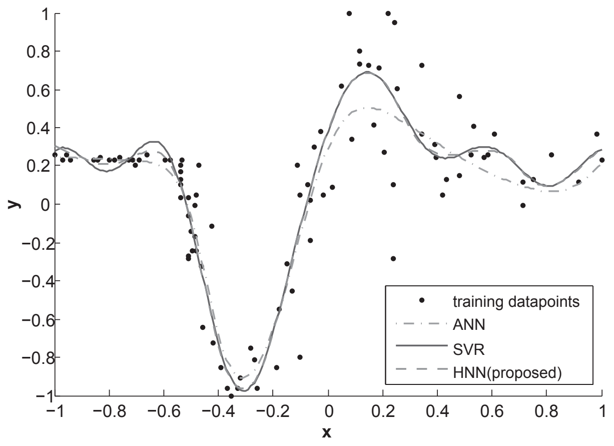

Fig. 3. Estimated regression function of ANN, SVR and HNN on *Motorcycle* dataset.

function is associated with a different set of support vectors while the regression functions of the ANN are determined by a common set of hidden nodes with different weights.

For regression problems, a similar procedure as used for the classification problem can be applied. The HNN for regression is given belows.

Algorithm 2. HNN for regression

Given a training dataset $\mathfrak{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N, y_i \in \mathbb{R}$, the detailed procedure for the proposed method is as follows.

(Step 1 – Train the SVR for the dataset $\mathfrak{D}$.) The SVR is trained with the original training dataset $\mathfrak{D}$ to obtain the regression function $f \in \mathbb{R}$.

(Step 2 – Construct a new training dataset $\mathfrak{D}'$ using the regression function of the SVR.) The regression function values $f(\boldsymbol{x}_i)$ of the original training dataset $\mathfrak{D}$ using the trained SVR are computed. Using this, a new refined training dataset $\mathfrak{D}' = \{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1}^N$ is constructed.

(Step 3 – Train the ANN for the dataset $\mathfrak{D}'$.) A single-output ANN, whose regression function corresponds to the regression function of the SVR, is trained with the dataset $\mathfrak{D}'$.

(Step 4 – Compute the regression estimates of the test datapoints using the ANN.) The final regression results are obtained by the regression function of the trained ANN.

In Algorithm 2, the regression function of the SVR is approximated with an ANN. The difference from the classification case is that the ANN always has a single output node. Fig. 3 illustrates the regression functions obtained by SVR, ANN, and HNN, respectively, on the *Motorcycle* dataset (Silverman, 1985). For this figure, the ANN is trained with the original training datapoints. As shown, the regression function of HNN almost completely fits that of the SVR, and is different from that of the ANN.

In HNN method, an ANN effectively approximates both SVC and SVR because the number of hidden nodes of the ANN is generally less than the number of support vectors of the SVC or SVR on a dataset. Therefore the HNN shows faster speed in the test phase while maintaining the good prediction accuracy of the SVC or SVR.

4. Experiments

To demonstrate the effectiveness of our proposed HNN method for the classification and regression problem, we conducted experiments with a total of 16 benchmark datasets: ten datasets for classification and six datasets for regression. All datasets were partitioned into 70% for the training set and 30% for the test set. Additionally, all numerical features were normalized to be in $[-1,1]$.

For the SVC and SVR, the RBF kernel function was adopted as a kernel function. For the ANN, we employed the Levenberg–Marquardt back propagation algorithm that performs better for regression problems (Hagan & Menhaj, 1994). The number of hidden layers was set to 1. In addition, the logistic sigmoid function was used as the sigmoidal function. For all algorithms, the best parameters were found using 10-fold cross validation on the training dataset with a grid search mechanism. All experiments were performed in MATLAB 7.14.

4.1. Classification results

To validate the classification performance of the proposed method, we chose the following ten benchmark datasets from the UCI repository (Bache & Lichman, 2013): *Iris*, *Wine*, *Sonar*, *Glass*, *Ionosphere*, *BreastCancer*, *Vechicle*, *Vowel*, *Yeast*, and *Segment*. In this paper, the one-against-one approach, which has been reported to show higher accuracy than other approaches (Hsu & Lin, 2002), was adopted to extend the conventional SVC algorithm for the multi-class classification problem. For the one-against-one approach, $c(c-1)/2$ different SVCs are trained for each pair of classes. Thus, the decision functions of the SVCs for each class pair were replaced with $c(c-1)/2$ output nodes of an ANN. The best parameters for the SVC were explored on a two dimensional grid search with $C = \{2^{-3}, 2^{-2}, \ldots, 2^{10}\}$ and $\sigma = \{2^{-5}, 2^{-4}, \ldots, 2^{5}\}$.

We evaluated the performances using the misclassification error rate (%) which is defined by $1/N\sum_{i=1}^N 1_{y_i \neq \hat{y}_i} \times 100$, where N is the number of test datapoints, y_i is actual class of the i-th

Table 1
Comparison of classification results on benchmark datasets.

Dataset [#data × dim.]	#class	SVC			HNN (proposed)			Comparison	
		Error rate (%)	Test time (ms)	# support vectors	Error rate (%)	Test time (ms)	# hidden nodes	Error rate (%) difference	Test time reduction rate (%)
Iris [150 × 4]	3	4.444	0.178	55	4.444	0.027	13	0.000	85.039
Wine [178 × 13]	3	3.774	0.273	77	3.774	0.015	4	0.000	94.557
Sonar [208 × 60]	2	11.290	0.500	105	12.903	0.040	11	1.613	92.098
Glass [214 × 9]	6	35.938	0.386	106	35.938	0.037	12	0.000	90.425
Ionosphere [351 × 34]	2	7.619	0.339	54	7.619	0.050	10	0.000	85.327
BreastCancer[683 × 9]	2	2.451	0.376	52	2.451	0.071	10	0.000	81.100
Vehicle [846 × 18]	4	16.996	1.492	263	16.601	0.179	19	0.395	88.021
Vowel [990 × 10]	11	1.347	2.633	422	1.684	0.250	38	0.337	90.513
Yeast [1484 × 8]	10	41.124	7.881	752	41.573	0.276	12	0.449	96.498
Segment [2310 × 19]	7	2.742	4.383	287	2.886	0.323	21	0.144	92.629
Average		–	–	–	–	–	–	0.294	89.621

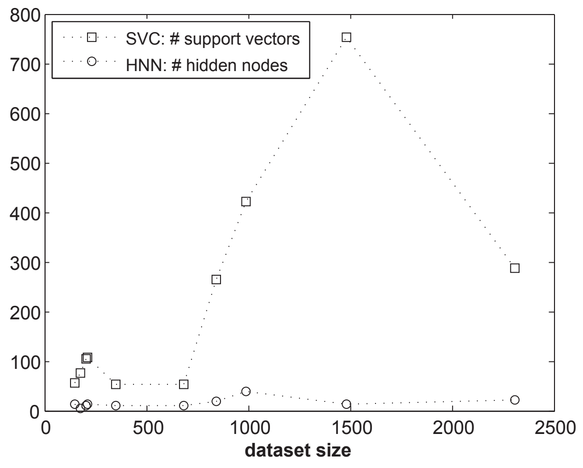

Fig. 4. Comparison between the number of support vectors for SVC and the number of hidden nodes for HNN.

datapoint, $\hat{y}_i$ is predicted class of the i-th datapoint, and $1_{y_i \neq \hat{y}_i}$ is an indicator function that has value 1 when $y_i \neq \hat{y}_i$. In addition, test time and the number of support vectors or hidden nodes were measured to validate the effectiveness of the proposed method.

Table 1 presents the classification results that compare the proposed HNN with a conventional SVC. HNN has a similar error rate to the SVC, but shows significantly faster test time on every dataset. On average, the proposed method achieved approximately 89.6% reduction in test time. Moreover, the test time was further reduced for large-scale datasets. For example, for the *Yeast* dataset, the number of support vectors for the SVC was 752 while there were only 12 hidden nodes for the HNN. Therefore, the test time was reduced by 96.5%.

Fig. 4 plots the number of support vectors and hidden nodes against dataset size for each dataset. As shown in Fig. 4, as the dataset size grows, the number of support vectors of SVC greatly increases while the number of hidden nodes of HNN does not.

4.2. Regression results

For the regression experiments, *Motorcycle* from (Silverman, 1985), *Bodyfat* and *SpaceGa* from the StatLib datasets archive (Vlachos, 2013), and *Autompg*, *Housing*, and *ConcreteCS* from the UCI repository (Bache & Lichman, 2013) were used. For all the datasets, the best parameters for the SVR were selected from the set of values $C = \{2^{-3}, 2^{-2}, \ldots, 2^{10}\}, \sigma = \{2^{-5}, 2^{-4}, \ldots, 2^{5}\}$, and $\epsilon = \{2^{-10}, 2^{-9}, \ldots, 2^{-1}\}$. The performance was measured by root mean squared error (RMSE)=$\sqrt{1/N\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, where N is the number of test datapoints, y_i is an i-th actual value, and $\hat{y}_i$ is a predicted value. We also observed the test time and the number of support vectors or hidden nodes as indicators of their computational load in the test phase.

The regression results of the benchmark datasets are shown in Table 2. It can be shown that the test time of the HNN is much faster than that of the SVR. However, there is no significant difference between the RMSE of the HNN and SVR. On average, the test time of the proposed method was reduced by about 88.5% with only a 0.002 of RMSE difference. In addition, the reduction rate of the test time tended to increase with the size of the dataset. In the case of the largest dataset, *SpaceGA*, that achieved a 98.9% reduction rate, the number of hidden nodes for the HNN was 21 while the number of support vectors for the SVR was 1721.

Fig. 5 represents the relationship of the number of support vectors for the SVR and the hidden nodes for the HNN according to the size of the dataset. Similar to the classification results, it can be seen that the number of hidden nodes is always less than the

Table 2
Comparison of regression results on benchmark datasets.

Dataset [#data × dim.]	SVR			HNN (proposed)			Comparison	
	RMSE	Test time (ms)	# support vectors	RMSE	Test time (ms)	# hidden nodes	RMSE difference	Test time reduction rate (%)
Motorcycle [133 × 1]	0.237	0.168	55	0.235	0.027	8	0.001	83.774
Bodyfat [252 × 14]	0.021	0.493	110	0.021	0.037	6	0.000	92.511
Autompg [392 × 7]	0.123	0.594	82	0.125	0.116	15	0.002	80.490
Housing [506 × 13]	0.136	1.067	176	0.133	0.185	19	0.003	82.673
ConcreteCS [1030 × 8]	0.147	5.834	499	0.145	0.420	25	0.002	92.797
SpaceGA[3107 × 6]	0.066	53.067	1721	0.065	0.606	21	0.001	98.857
Average	–	–	–	–	–	–	0.002	88.517

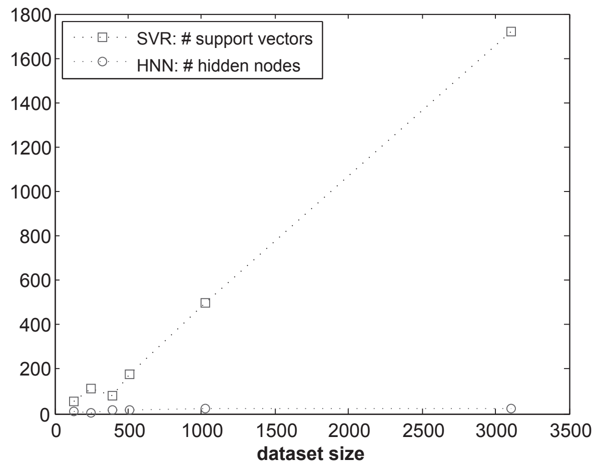

Fig. 5. Comparison between the number of support vectors for SVR and the number of hidden nodes for HNN.

number of support vectors. Furthermore, the larger the size of the dataset, the greater the difference achieved.

5. Concluding remarks

The major drawback of the SVM is its low speed in the training and test phases. Most studies focus on accelerating the SVM in the training phase while relatively few studies have been conducted on the test phase of the SVM, in spite of the importance of run-time speed of the SVM.

In this paper, HNN, an ANN based hybrid method to accelerate support vector machines was proposed. The proposed method was motivated by the characteristics that the computation of the RBF kernel function of an SVM and the computation of the sigmoidal function of an ANN are similar but the number of hidden nodes of an ANN is generally smaller than the number of support vectors of an SVM, for any dataset.

The proposed method was applied to two SVM algorithms, SVC and SVR. The regression functions of the ANN approximated the decision functions of the SVC or the regression function of the SVR. That is, the trained SVC or SVR was replaced by an ANN. From the results, both the SVC and SVR were efficiently approximated. Experimental results on several benchmark datasets showed that the proposed method achieved significant test time reductions compared to conventional SVM without compromising prediction accuracy. Moreover, the proposed method was much more effective for large-scale datasets.

The distinguishing point of the proposed method over existing work is that the approximation problem of both the SVC and SVR is converted into the regression-based function approximation problem. In other words, the SVC and SVR generate noise-filtered functions and the ANN just approximates these functions. It merits further investigations. The proposed method can approximate other complex algorithms that generate noise-filtered functions, and can be improved by using any kinds of regression algorithms including ANN.

The proposed method replaces the SVM with the approximation of the ANN for predicting new datapoints, based on a hybrid approach. Thus, it is not strongly restricted by the structure of the SVM. Instead, the structure of the ANN should be considered. In our experiments, we chose the best number of hidden nodes of the ANN through cross validation. If high run-time speed is required and some prediction losses are allowable, we can manually employ the smaller number of hidden nodes. Despite the advantages, the proposed method causes more computational load in the training phase because it requires additional training of the ANN, which is the common limitation of most hybrid approaches.

Overall, the main research contribution of the proposed method is that it approximates the SVM with the ANN through regression-based function approximation, hence giving higher run-time speed compared to the conventional SVM without sacrificing the prediction accuracy of the SVM. Therefore, the application of the proposed method can improve the practical usability of SVM, especially in real-time applications.

In real-time applications, the run-time speed is critical. One example is a driver state monitoring system. It has to detect in real-time from the facial image if a driver is drowsy. Another example is fault detection and diagnosis of equipment in manufacturing systems. This system has to monitor the conditions of equipment in real-time, and the abnormal events have to be detected immediately. High run-time speed as well as high prediction accuracy is greatly required because they commonly deal with large data stream in real-time. Thus, the proposed method, which has a fast run-time speed with taking the prediction accuracy of the SVM, could be very helpful for these examples.

As future work, we intend to apply other non-linear regression algorithms that are faster than the ANN in terms of run-time. By doing so, a further improvement of run-time speed of the SVM is expected. In addition, we intend to apply the proposed method not only to the SVC and SVR which are covered in this paper but also to other complex algorithms that are time consuming in the test phase such as one-class SVM and Gaussian process regression.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIP) (No. 2011-0030814), and the Brain Korea 21 PLUS Project in 2014. This work was also supported by the Engineering Research Institute of SNU.

References

Bache, K., & Lichman, M. (2013). UCI machine learning repository.
Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
Burges, C. J. C. (1996). Simplified support vector decision rules. In *Proceedings of the 13th international conference on machine learning* (pp. 71–77).
Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*, 121–167.
Chen, J.-H., & Chen, C.-S. (2004). Reducing SVM classification time using multiple mirror classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34*, 1173–1183.
Downs, T., Gates, K. E., & Masters, A. (2002). Exact simplification of support vector solutions. *The Journal of Machine Learning Research, 2*, 293–297.
Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks, 5*, 989–993.
Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*, 415–425.
Ji, J., & Zhao, Q. (2013). A hybrid SVM based on nearest neighbor rule. *International Journal of Wavelets, Multiresolution and Information Processing, 11*, 1350048.
Joachims, T. (1999). Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods: support vector learning* (pp. 169–184). Cambridge, MA, USA: MIT Press.
Kim, D., & Cho, S. (2012). Pattern selection for support vector regression based response modeling. *Expert Systems with Applications, 39*, 8975–8985.
Kumar, Arun, & Gopal, M. (2010). A hybrid SVM based decision tree. *Pattern Recognition, 43*, 3977–3987.
Liang, X., Ma, Y., He, Y., Yu, L., Chen, R.-C., Liu, T., et al. (2013). Fast pruning superfluous support vectors in SVMs. *Pattern Recognition Letters, 34*, 1203–1209.
Li, Q., Jiao, L., & Hao, Y. (2007). Adaptive simplification of solution for support vector machine. *Pattern Recognition, 40*, 972–980.
Lin, H.-J., & Yeh, J. P. (2009). Optimal reduction of solutions for support vector machines. *Applied Mathematics and Computation, 214*, 329–335.
Li, Y., Zhang, W., & Lin, C. (2006). Simplify support vector machines by iterative learning. *Neural Information Processing: Letters and Reviews, 10*, 11–17.

Maji, S., Berg, A., & Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 1–8). IEEE.

Maji, S., Berg, A., & Malik, J. (2013). Efficient classification for additive kernel SVMs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*, 66–77.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14 Microsoft Research.

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

Schmitz, G. P., Aldrich, C., & Gouws, F. S. (1999). ANN-DT: An algorithm for extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks, 10*, 1392–1401.

Shin, H., & Cho, S. (2007). Neighborhood property-based pattern selection for support vector machines. *Neural Computation, 19*, 816–855.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society Series B (Methodological)*, 1–52.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*, 199–222.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Vedaldi, A., & Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*, 480–492.

Vlachos, P. (2013). Statlib datasets archive.

Xu, X., Zhang, B., & Zhong, Q. (2005). Text categorization using SVMs with rocchio ensemble for internet information classification. In *Proceedings of the third international conference on networking and mobile computing* (pp. 1022–1031). Springer.