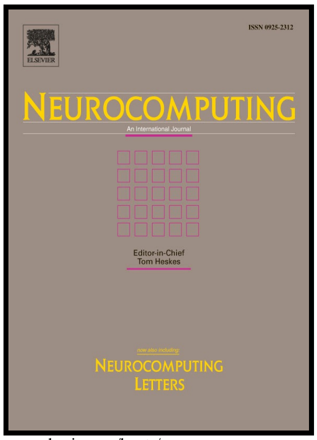
# Author's Accepted Manuscript




www.elsevier.com/locate/neucom

Computational Performance Optimization **of Support Vector Machine Based on Support Vectors**

Xuesong Wang, Fei Huang, Yuhu Cheng

PII: S0925-2312(16)30564-1
DOI: http://dx.doi.org/10.1016/j.neucom.2016.04.059
Reference: NEUCOM17176

To appear in: *Neurocomputing*

Received date: 31 July 2015
Revised date: 18 March 2016
Accepted date: 6 April 2016

Cite this article as: Xuesong Wang, Fei Huang and Yuhu Cheng, Computational Performance Optimization **of Support Vector Machine Based on Suppor Vectors**, *Neurocomputing,* http://dx.doi.org/10.1016/j.neucom.2016.04.059

# Computational Performance Optimization of Support Vector Machine Based on Support Vectors

Xuesong Wang, Fei Huang, Yuhu Cheng*

School of Information and Electrical Engineering, China University of Mining and Technology, 221116, Xuzhou, China

*Corresponding author. chengyuhu@163.com

**Abstract**

The computational performance of support vector machine (SVM) mainly depends on the size and dimension of training sample set. Because of the importance of support vectors in the determination of SVM classification hyperplane, a kind of method for computational performance optimization of SVM based on support vectors is proposed. On one hand, at the same time of the selection of super-parameters of SVM, according to Karush-Kuhn-Tucker condition and on the precondition of no loss of potential support vectors, we eliminate non-support vectors from training sample set to reduce sample size and thereby to reduce the computation complexity of SVM. On the other hand, we propose a simple intrinsic dimension estimation method for SVM training sample set by analyzing the correlation between number of support vectors and intrinsic dimension. Comparative experimental results indicate the proposed method can effectively improve computational performance.

**Keywords:** support vector machine; support vector; sample size; intrinsic dimension; computational performance

## 1. Introduction

Support vector machine (SVM) proposed by Vapnik [1] is one of important machine learning model. SVMs can achieve good performance by using the implementation of the structural risk minimization principle and the introduction of the kernel trick [2]. Due to the powerful abilities of generalization, small-sample and nonlinear processing, SVMs have been successfully applied to classification decision [3, 4], regressive modeling [5], fault diagnosis [6, 7] and bioinformatics [8, 9]. Although SVM can effectively avoid the problem of 'curse of dimensionality', the degraded computational performance caused by the increase of sample size or dimension cannot be solved, which are usually dealt by the following two ways: one is the improvement of learning algorithm, such as sequential minimal optimization (SMO) [10], successive overrelaxation (SOR) [11] and $LIBSVM_{CBE}$ [12] while the other is to simplify computation by reducing sample size or dimension of training sample set, which are respectively studied in this paper.

In dealing with large datasets, how to reduce the scale of training samples is one of the most important aspects to improve computation efficiency. There are some major difficulties when processing large data concerning a fully dense nonlinear kernel matrix. To overcome computational difficulties, some authors have proposed low-rank approximation to the full kernel matrix. As an alternative, Lee and Mangasarian [13] have proposed the method of reduced support vector machine (RSVM). The key ideas of the RSVM are as follows. Prior to training, it selects a small random subset as to generate a thin rectangular kernel matrix. Then, it uses this much smaller rectangular kernel matrix to replace the full kernel matrix in the nonlinear SVM formulation. Ke and Zhang [14] proposed a method called editing support vector machine (ESVM) by removing some samples near the boundary from the training set. Its basic scheme is similar to that editing nearest neighbor methods in statistical pattern recognition, which randomly divides the training set into subsets, using one subset to edit the other one and then get the final decision boundary