Neurocomputing 72 (2008) 3–14

Contents lists available at ScienceDirect

Neurocomputing

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

Improving radial basis function kernel classification through incremental learning and automatic parameter selection

Carlos Renjifo *, David Barsic, Craig Carmen, Kevin Norman, G. Scott Peacock

The Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, MD 20723, USA

ARTICLE INFO

Available online 25 September 2008

Keywords:
Support vector machine
Run-time complexity
Parameter estimation
Simultaneous perturbation stochastic approximation
Greedy search

ABSTRACT

Training algorithms for radial basis function kernel classifiers (RBFKCs), such as the support vector machine (SVM), often produce computationally burdensome classifiers when large training sets are used. Furthermore, the developer cannot directly control this complexity. The proposed incremental asymmetric proximal support vector machine (IAPSVM) employs a greedy search across the training data to select the basis vectors of the classifier, and tunes parameters automatically using the simultaneous perturbation stochastic approximation (SPSA) after incremental additions are made. The resulting classifier accuracy, using an *a priori* chosen run-time complexity, compares favorably to SVMs of similar complexity whose parameters have been tuned manually.

1. Background

1.1. Notation

In this paper, scalars will be denoted by lowercase italic letters, vectors by lowercase bold letters, and matrices by uppercase bold letters. Functions are defined with italicized letters followed by a list of parameters in parentheses. A vector of ones of arbitrary length is denoted $\mathbf{e}$. The training data are represented by an $m \times n$ matrix, where m is the number of training samples and n is the number of features. $\mathbf{A}_i$ denotes the ith row (sample) in $\mathbf{A}$. This notation differs from the standard convention, where vectors are defined as columns, but was chosen to preserve notational consistency with the papers that form the foundation of this work, namely [9,10]. $\bar{\mathbf{A}}$ denotes a selected subset of rows from $\mathbf{A}$. Vertical concatenation of two matrices $\mathbf{A} \in \Re^{m \times n}$ and $\mathbf{B} \in \Re^{p \times n}$ is expressed as $[\mathbf{A};\ \mathbf{B}] \in \Re^{(m+p) \times n}$. Horizontal concatenation of two matrices with $\mathbf{A} \in \Re^{m \times n}$ and $\mathbf{B} \in \Re^{m \times p}$ is expressed as $[\mathbf{A}, \mathbf{B}] \in \Re^{m \times (n+p)}$. A quantity with a superscript in parentheses, such as $\mathbf{X}^{(i)}$ for a matrix, $\mathbf{x}^{(i)}$ for a vector, or $x^{(i)}$ for a scalar, is the ith instance of some repeated process. Variables with carets, such as $\hat{\mathbf{Y}}$ for a matrix, $\hat{\mathbf{y}}$ for a vector, or $\hat{y}$ for a scalar, represent estimates of some quantity. The training data in $\mathbf{A}$ have associated class labels $\{-1, 1\}$ along the main diagonal of the $m \times m$ matrix $\mathbf{D}$. The radial basis function (RBF) kernel matrix with tunable width parameter γ for any two matrices $\mathbf{A} \in \Re^{m \times n}$ and $\mathbf{B} \in \Re^{p \times n}$ is denoted by the matrix $\mathbf{K} \in \Re^{m \times p}$, where the ith row ($i = 1, \ldots, m$) and jth column ($j = 1, \ldots, p$) of $\mathbf{K}$ are defined as follows:

$$\mathbf{K}_{ij} = K(\mathbf{A}_i, \mathbf{B}_j) = \exp(-\gamma ||\mathbf{A}_i - \mathbf{B}_j||^2) \tag{1}$$

1.2. Problem overview

This paper focuses on building radial basis function kernel classifiers (RBFKCs) under constraints commonly encountered in real-world situations. These classification scenarios often contain data sets with large numbers of training points ($>10^4$ samples) and high feature dimensionality (10^1–10^3 features).

In a resource-constrained scenario, training may be done “off line” with any preferred method (i.e., there are no appreciable constraints placed on time or memory), but the resulting classifier must be capable of being *implemented* into a system with limited computational, power, and/or memory resources. Current methods focus on improving classifier accuracy or reducing training time or memory usage, but they do not consider the run-time implications. When the classifier is used in systems such as embedded devices or autonomous distributed sensors, the run-time efficiency of the classifier becomes a key constraint on model selection. The methods proposed in this paper focus on this aspect of the problem.

1.3. RBF kernel-based classification

The support vector machine (SVM) [31] is one of the most popular and robust nonparametric classification algorithms in current literature. Its method of “maximizing the margin” or “Large-Margin Classification” has shown excellent performance

* Corresponding author. Tel.: +1 443 778 4527.
E-mail addresses: Carlos.Renjifo@jhuapl.edu (C. Renjifo), David.Barsic@jhuapl.edu (D. Barsic), Craig.Carmen@jhuapl.edu (C. Carmen), Kevin.Norman@jhuapl.edu (K. Norman), Scott.Peacock@jhuapl.edu (G.S. Peacock).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.04.053

on a variety of classification problems. The canonical SVM algorithm is a linear classifier; nonlinear classification is achieved via a kernel transform satisfying Mercer's conditions, such as the RBF kernel defined in Eq. (1). Other kernels that satisfy Mercer's conditions include the polynomial kernel and the hyperbolic tangent kernel. Burges provides an excellent tutorial on these concepts in [5].

Many variants based on the original SVM have been proposed. A least-squares concept is developed for both the least-squares support vector machine (LS-SVM) [25] and the proximal support vector machine (PSVM) [10]. Similar to the SVM, LS-SVM and PSVM are linear classifiers that use kernel transforms to permit nonlinear classification. For all of these related methods, classification of unknown patterns is achieved via the following:

$$f(\mathbf{x}) = \operatorname{sign}(K(\mathbf{x}, \bar{\mathbf{A}})\mathbf{w} + b) \tag{2}$$

where $\bar{\mathbf{A}} \in \Re^{p \times n}$ is a subset of p rows of $\mathbf{A}$, and $\mathbf{w} \in \Re^p$ and b are the weight and bias terms computed by the respective algorithms. In this paper, any classifier that can be evaluated using Eq. (2), with K defined by Eq. (1), is called a RBFKC.

In the SVM, the rows of $\bar{\mathbf{A}}$ are the support vectors, which are defined as points that either lie on the canonical hyperplanes or are incorrectly classified. The physical meaning of the "support vectors" is not consistent between the SVM and the least-squares SVM methods. Therefore, the more general term "kernel center" is used to denote any point about which the RBF kernel transform must be evaluated to classify an unknown data sample. Running the classifier on a new data point requires calling the kernel function once for each kernel center. Thus, the run-time computation of Eq. (2) is directly proportional to the number of kernel centers contained in $\bar{\mathbf{A}}$.

1.4. *Limitations of the standard SVM*

Theoretically, Steinwart demonstrated that a linear relationship exists between the expected number of kernel centers in a trained classifier and the size of the training set for a RBFKC. In particular, as the size of the training set approaches infinity, the fraction of training data points used as kernel centers approaches twice the Bayesian misclassification rate for the L1-SVM, the noise probability for the L2-SVM, and 1 for the LS-SVM [24]. A consequence of this discovery is that SVMs using RBF kernel transforms trained with large data sets typically yield classifiers with large numbers of kernel centers.

A further limitation of the SVM is that the number of kernel transforms that needs to be computed at run time to classify unknown data cannot be specified during the training stage. Although it is possible via trial and error to reduce/increase the size of the training set to produce a classifier with a desired number of kernel centers, such an approach is tedious, imprecise, and likely to yield suboptimal results.

The preceding limitations render the standard SVM impractical for use in resource- or power-constrained systems, where upper bounds in computational complexity often exist, except in the unlikely event that linear classification yields acceptable performance or the available training data are very limited.

1.5. *Survey of SVM modifications to reduce run-time complexity*

Various modifications to the SVM have been proposed to reduce run-time complexity. Such methods often attempt to remove unnecessary members of the training data set. Tsang et al. [29] built the core vector machine (CVM) using a "core vector set" found by solving a minimum enclosing ball (MEB) problem. Editing techniques were employed by Bakir et al. in [1]. Zhan and Shen [33] used adaptive techniques that penalize extreme outliers. Others have developed incremental approaches for increasing SVM efficiency, such as Keerthi et al. [12] and Bordes et al. [4]. Fung and Mangasarian [9] built an incremental version of the PSVM to allow an existing linear classifier to be modified to reflect the solution using the original training data with points added or removed—their approach is quite different from the approach to be presented here.

Osuna and Girosi [17] proposed a two-stage algorithm involving the solution of the original SVM problem followed by a support vector machine regression (SVMR). Although their method provides controllable sparseness, it has been shown to be effective only in highly separable, low-dimensional classification environments. Furthermore, it is limited to small training sets ($<$10,000 points) due to memory limitations.

Finally, the relevance vector machine (RVM) of Tipping [27] is a RBFKC that achieves substantial efficiency. However, even with recent improvements to the training time and sparseness of the final solution [28], the algorithm still requires a substantially higher training time (on the order of m^3, where m is the number of training points) than competing SVM techniques, making it impractical for use with large training sets (10,000–100,000 points). Additionally, the RVM does not contain explicit controls to pre-specify the final complexity of the classifier solution.

Various least-squares algorithms have been developed that use a two-norm rather than one-norm objective function. These SVM variants lead to efficient, analytic solutions, but lack sparseness. Without sparseness, the trained classifiers are slow at run time. Suykens et al. [26] attempted to improve the sparseness by pruning training data points corresponding to small (in magnitude) weights. Alternatively, Lee and Mangasarian [15] applied reduced kernel techniques in their PSVM algorithm. Although the method used all data points as constraints, it employed only a small subset of points as kernel centers.

1.6. *Publication overview*

The remainder of this publication focuses on the proposed incremental asymmetric proximal support vector machine (IAPSVM) algorithm. Beginning with a discussion of the least-squares adaptations to the SVM, Section 2 describes the formulation of the IAPSVM method. In Section 3, the problem of parameter estimation is discussed, and a parameter-tuning scheme based on Spall's simultaneous perturbation stochastic approximation (SPSA) algorithm [21] is introduced to eliminate manual tuning. Finally, Section 4 compares the performance of the IAPSVM algorithm with several SVM formulations on two benchmark data sets: the UCI Forest Covertype data set [3] and the MNIST handwritten digit data set [14].

2. Methods to build efficient RBF kernel classifiers

2.1. *Overview of the proximal SVM adaptation*

The canonical linear SVM is the solution to the quadratic programming problem given as

$$\begin{aligned} \min_{(\mathbf{w},b,\xi)\in\Re^{n+1+m}} \quad & \nu\mathbf{e}^{\mathrm{T}}\xi + \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \\ \text{s.t.} \quad & \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{e}b) \geqslant \mathbf{e} - \xi \\ & \xi \geqslant 0 \end{aligned} \tag{3}$$

Three significant changes are made to convert the SVM in Eq. (3) to a PSVM. First, the two-norm of the margin ξ error replaces the