Neurocomputing (xxxx) xxxx–xxxx

Contents lists available at ScienceDirect

Neurocomputing

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

A fast algorithm for sparse support vector machines for mobile computing applications

Jian-Xun Peng, Karen Rafferty*, Stuart Ferguson

The School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, UK

ARTICLE INFO

Communicated by Dorothy Ndedi Monekosso

Keywords:
Data classification
Sparse support vector machines
Recursive algorithm
Sequential training algorithm

ABSTRACT

This research presents a fast algorithm for projected support vector machines (PSVM) by selecting a basis vector set (BVS) for the kernel-induced feature space, the training points are projected onto the subspace spanned by the selected BVS. A standard linear support vector machine (SVM) is then produced in the subspace with the projected training points. As the dimension of the subspace is determined by the size of the selected basis vector set, the size of the produced SVM expansion can be specified. A two-stage algorithm is derived which selects and refines the basis vector set achieving a locally optimal model. The model expansion coefficients and bias are updated recursively for increase and decrease in the basis set and support vector set. The condition for a point to be classed as outside the current basis vector and selected as a new basis vector is derived and embedded in the recursive procedure. This guarantees the linear independence of the produced basis set. The proposed algorithm is tested and compared with an existing sparse primal SVM (SpSVM) and a standard SVM (LibSVM) on seven public benchmark classification problems. Our new algorithm is designed for use in the application area of human activity recognition using smart devices and embedded sensors where their sometimes limited memory and processing resources must be exploited to the full and the more robust and accurate the classification the more satisfied the user. Experimental results demonstrate the effectiveness and efficiency of the proposed algorithm. This work builds upon a previously published algorithm specifically created for activity recognition within mobile applications for the EU Haptimap project [1]. The algorithms detailed in this paper are more memory and resource efficient making them suitable for use with bigger data sets and more easily trained SVMs.

1. Introduction

The core aim of the research presented in this paper is to refine and extend a new generation of algorithms to underpin, much more accurately, the process of activity recognition using data derived from mobile sources. As the popularity of low cost portable hand-held computers and mobile phones increases, opportunities for novel context aware applications have grown. Mobile phones can be used along with wearable accelerometers to create valid and reliable measures of physical activity. However, to do this effective algorithms are also needed to interpret the data in the context of different activities. We do this by extending the algorithms published previously [1] which were tested on activity data collected from mobile sensors. Here we improve upon those algorithms and more thoroughly test them (using standard benchmarks) in terms of memory efficiency which is crucial for the mobile storage devices typically used for assisted living.

The algorithms we have developed are based on the Support vector machine(SVM). SVMs are a set of empirical data modelling techniques, which are firmly grounded in the *VC theory* proposed by Vapnik [2], and provide the start-of-the-art performance. The structural risk minimization (SRM) principle implemented by SVM overcomes the difficulties with generalization that have been suffered by traditional neural networks [3], and allows SVMs to provide very accurate solutions.

There has been increasing interest in seeking sparse representations of regular (accurate) SVMs to tackle this problem. Existing techniques proposed for reduced sizes of SVMs fall into two classes: *post-training algorithms* and algorithms that directly yield sparse SVMs, referred to as *sparse algorithms* or *direct algorithms*. Since the generalization performance of a regular SVM is guaranteed (by the SRM principle), post-training algorithms produce a standard SVM in advance and then approximate the normal vector to the separating hyperplane in the feature space, where the SVM discriminant function is expressed as a linear expansion of the support vectors (SVs). A family of linear expansions in the feature space of smaller sizes are used to approximate the normal vector which minimizes the Euclidean distance between the approximation normal and the original one in the

* Corresponding author.
E-mail address: k.rafferty@ee.qub.ac.uk (K. Rafferty).

http://dx.doi.org/10.1016/j.neucom.2015.11.136
Received 21 April 2015; Received in revised form 12 August 2015; Accepted 12 November 2015

feature space is identified. The approximated SVM discriminant function is thus expressed as the inner product of the approximating normal vector and an input vector in the feature space.

Downs [4] proposed an exact algorithm which prunes SVs from the full SVM solution. Given that the normal vector is a linear combination of the support vectors, all SVs that are linearly dependent (in the feature space) are removed. Obviously, the maximal size of the reduced support vector set is the number of dimensions of the feature space, although it is generally unknown in nonlinear cases. Exact algorithms yield sparse solutions without any loss of the ability to generalize (as the normal vector remains unchanged). However the exact method does not work when a further reduction to the support vector set (SVS) is desired.

Most existing post-training algorithms are approximation methods that approximate the normal vector in a linear expansion of much smaller sizes.

On the other hand sparse SVM algorithms directly minimize the primal objective function with the additional constraint that the normal vector will be a linear expansion of a given number of vectors in the feature space. This means that the search space for the normal vector is restricted, rather than the full feature space required for standard SVMs, and that the resulting reduced size SVM still have a maximal margin.

Lee and Mangasarian [5] randomly choose a subset (typically 1–10%) of the given training vectors as candidate SVs during the optimization while classification errors are evaluated over the full training set. In this way the scale of the problem of SVM training (the number of variables) is reduced, resulting in greatly reduced SVM (RSVM) classifiers. However as the expansion vectors are chosen from a random candidate set, and may not be good representatives of the training data, good classification performance can not be guaranteed when the randomly chosen subset is small [6].

Addressing this problem, Wu et al. [7,8] proposed algorithms for directly building sparse kernel classifiers. The normal vector to the separating hyperplane is expressed as a linear expansion of a given number of vectors in the feature space. Direct algorithms minimize the primal objective function for standard SVMs with the linear expansion substituted for the normal vector. In addition, the expansion vectors (XVs) for the normal vector are optimized using a gradient-based search, rather than selected from the training set. However optimization of the XVs is a hard non-convex nonlinear problem. Keerthi et al. [9] proposed a sequential incremental algorithm that selects one vector from the training set each time, hence avoiding the hard non-convex optimization problem for XVs. The corresponding expansion coefficients are optimized using a Newton-Raphson method such that the primal objective function is minimized. This incremental selection is iterated until a given number of vectors are selected.

Typically SVMs are not preferred for real-time applications with limited computational resources (e.g. available RAM or CPU speed) since a large set of support vectors (SVs) is needed to form the SVM classifier, making it computationally complex and expensive to implement. Whilst the advances in sparse SVMs have helped to overcome this issue from a software perspective it is also important to consider the hardware constraints especially when using smart devices. Anguita [10] introduced the concept of a hardware friendly SVM. This method exploits fixed point arithmetic in the feed-forward phase of the SVM. They then extended their models for multi-class problems [11]. They concluded that the use of fixed point calculations is useful in activity recognition applications because they require less memory, processor time and power consumption. Whilst this is important, it is also necessary to refine the basis of the SVM to make it more efficient whether or not it is based on fixed point integers. The main contribution of our paper in terms of algorithmic progress is to project the training points into the subspace spanned by a set of basis vectors selected in the feature space. In the subspace, the SVM is built with the projected training points, referred to as the projected SVM (PSVM). The basis vectors are initially selected from the training set incrementally, and then refined by combining decremental pruning and incremental selecting, resulting in a solution which is optimized over the training set. A condition for a vector that can be selected as an additional basis vector is proposed. This condition is checked recursively in the selection and refining procedures, thus confirming the linear independence of the selected basis set in the feature space. An advantage of PSVM over regular SVM is that the size of the PSVM expansion is determined by the size of the basis set, rather than the number of SVs. Compared with existing sparse algorithms, the SVM does not need to be built in advance and directly minimizes the primal objective function rather than approximating the SVM normal vector. It approaches locally optimal solutions while avoiding hard non-convex non-linear searches. This makes our algorithms particularly suited for implementation on mobile devices and therefore applicable to a range of health-care and assisted living applications.

In Section 2, the PSVM is presented following an outline of regular SVMs in the primal. Section 3 details a sequential algorithm to solve for the PSVM and optimize the basis set. An implementation of the PSVM algorithm follows in Section 4, where the computational complexity of the algorithm is analyzed. In Section 5, the proposed algorithm is tested over some public benchmark problems and compared with LibSVM for standard SVMs and the sparse SVM algorithm from [9]. Section 6 draws some conclusions on our algorithm. In particular, since in [1] we know the algorithms are suited to activity classification on mobile devices, we use these new benchmark tests to verify the advances made to the algorithms in terms of memory efficiency as clearly they will perform better on the activity classification data they were previously tested on.

2. Projected support vector machines

Given a data set of N point-label pairs $\{(\mathbf{x}_k, y_k), k = 1,\ldots,N\}$, referred to as the training set, each point is represented as a row vector $\mathbf{x}_k \in \Re^{1\times n}$, to which a label of either $+1$ or -1, i.e., $y_k \in \{+1, -1\}$, is attached. This means the training points fall into two categories. This is a binary data classification problem, where a classifier is to be found that can separate the points into two classes. For convenience, the training point set and the associated labels are denoted as $N \times n$ matrix $\mathbf{X} = [\mathbf{x}_1^T,\ldots,\mathbf{x}_N^T]^T$ and N×1 column vector $\mathbf{y} = [y_1,\ldots,y_N]^T$.

2.1. Regular SVMs in the primal

Conceptually, a SVM maps its input vector to a high-dimensional space through a kernel-induced map ϕ: $\mathbf{x} \to \mathbf{f} = \phi(\mathbf{x})$, where $\mathbf{x} \in \Re^{1\times n}$ is an arbitrary input point to the SVM, $\mathbf{f} = \phi(\mathbf{x})$ denotes its map in the feature space. The high-dimensional space is referred to as the *feature space*, while a point in the feature space, say $\mathbf{f}$, is referred to as a *feature*. In contrast, the space where input vectors $\mathbf{x}$ are from is referred to as the *input space*. Note that $\mathbf{f}$ is represented as a row vector as is $\mathbf{x}$. However the number of dimensions of $\mathbf{f}$ is normally unknown, thus $\mathbf{f}$ cannot be represented numerically.

A SVM defines two parallel hyperplanes $y = \mathbf{fw} + b \pm 1$ in the feature space that bound the two classes of points in the training set, namely

$$\begin{cases} \mathbf{f}_k\mathbf{w} + b \geq +1, & \text{if } y_k = +1 \\ \mathbf{f}_k\mathbf{w} + b \leq -1, & \text{if } y_k = -1 \end{cases} \tag{1}$$

hold for $k = 1,\ldots,N$, or equivalently

$$y_k(\mathbf{f}_k\mathbf{w} + b) \geq 1, \quad k = 1,\ldots,N \tag{2}$$

The hyperplane $y = \mathbf{fw} + b$ that lies midway between the two parallel bounding hyperplanes separates the training points, where $\mathbf{f}_k = \phi(\mathbf{x}_k)$ denotes the map of training point $\mathbf{x}_k$ (a point from the input space) in the feature space, $\mathbf{w}$ denotes the normal (column) vector common to