Engineering Applications of Artificial Intelligence 25 (2012) 1203–1211

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

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

Low-cost model selection for SVMs using local features

Miguel Lázaro-Gredilla [a,b,*], Vanessa Gómez-Verdejo [a], Emilio Parrado-Hernández [a]

[a] Signal Proc. and Comm., Universidad Carlos III de Madrid, 28911 Leganés, Spain
[b] Department of Communications Engineering, Universidad de Cantabria, 39005 Santander, Spain

ARTICLE INFO

Article history:
Received 14 March 2011
Received in revised form
26 April 2012
Accepted 31 May 2012
Available online 23 June 2012

Keywords:
Support Vector Machines
Model selection
RBF kernels

ABSTRACT

Many practical engineering applications require the usage of accurate automatic decision systems, usually operating under tight computational constraints. Support Vector Machines (SVMs) endowed with a Radial Basis Function (RBF) as kernel are broadly accepted as the current state of the art for decision problems, but require cross-validation to select the free parameters, which is computationally costly. In this work we investigate low-cost methods to select the spread parameter in SVMs with an RBF kernel. Our proposal relies on the use of simple local methods that gather information about the local structure of each dataset. Empirical results in UCI datasets show that the proposed methods can be used as a fast alternative to the standard cross-validation procedure, with the additional advantage of avoiding the (often heuristic) task of a priori fixing the values of the spread parameter to be explored.

1. Introduction

Decision making, also known as classification, is a pervasive task that is present not only in many (if not all) industrial processes, but also in many other scenarios such as medicine and finance. Most of the time, the accuracy of the decisions made within a process is critical for its success as a whole. Due to this criticality and complexity, this task has been traditionally undertaken by experts in the relevant field, who had to make a choice based on the limited available information. While humans can handle complex tasks reasonably well, they are vulnerable to psychological biases, lack of consistency and are unable to consider large volumes of data simultaneously. With the advent of ever-cheaper computational resources and machine learning algorithms, experts can now rely on guidance from automatic systems which do not have the aforementioned drawbacks. These new decision making algorithms focus on achieving high accuracy at low computational cost. In this work we will strive to reduce computational cost while keeping state-of-the-art accuracy.

Support Vector Machines (SVMs) (Boser et al., 1992; Cristianini and Shawe-Taylor, 2000) are accepted as a standard *de facto* in automatic classification. An SVM is a maximum margin linear classifier, that can be extended to nonlinear problems by the use of the *kernel trick* (Schölkopf and Smola, 2002). Most practitioners find SVMs attractive because of their classification accuracy reported in the literature. But it should also be noticed that another major key behind SVMs success is that one only needs to tune one or two hyper-parameters before the main optimisation takes place. These hyper-parameters are the kernel design hyper-parameters and the omnipresent trade-off between regularisation and training set error minimisation, C. In this sense, most practitioners bear the following rule of thumb: use an SVM with a Radial Basis Function (RBF) as kernel and a 10-fold cross-validation process to determine C and the Gaussian kernel width σ.

The above mentioned hyper-parameter selection involves a huge increase in the computational cost of the training: one has to a priori define a grid of (C,σ) pairs and then solve 10 SVM problems of size 90% of the complete dataset per grid pair. Of course more sophisticated ways of surfing the grid can help reduce this computational burden (Momma and Bennett, 2002; Ortiz-García et al., 2009), but the question about which range of values to explore for both parameters remains open.

Although many authors have explored the use of different parameter setting techniques in machine learning algorithms (Shaheen et al., 2010; Pavón et al., 2008; Bengio, 2000) and, in particular, in SVMs (Friedrichs and Igel, 2004; Samanta et al., 2003), most of these methods are based on high computational cost techniques, such as genetic algorithms (Sardiñas et al., 2006; Samanta et al., 2003). This fact has prevented these techniques from becoming mainstream, and most practitioners still prefer to cross-validate their algorithms' parameters, see, for instance Kohavi and John (1995).

In this paper we focus on the estimation of the Gaussian kernel width. The motivation for this choice is twofold: on the one hand, σ determines the effective rank of the kernel matrix, and thus the

* Corresponding author at: Signal Proc. and Comm., Universidad Carlos III de Madrid, 28911 Leganés, Spain. Tel.: +34 916248769; fax: +34 916248749.
E-mail addresses: miguel@tsc.uc3m.es, miguellg@gtas.dicom.unican.es (M. Lázaro-Gredilla), vanessa@tsc.uc3m.es (V. Gómez-Verdejo), emipar@tsc.uc3m.es (E. Parrado-Hernández).

0952-1976/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2012.05.021

degrees of freedom that we may use to build a linear classifier in the feature space. In this sense, very small values of σ lead to ultra-localised kernels, resulting in a classifier equivalent to the 1-Nearest Neighbour; while very big values of σ lead to a classifier that always predicts the most populated class. On the other hand, once σ is fixed, the kernel matrix will not change with the variations on parameter C. This enables the use of a shared kernel matrix cache among all the explored values of C, saving computational cost.

Bearing in mind that the value of σ relates to the locality of the dataset, we propose to explore three methods to estimate σ based on the performance of two cheap non-linear classifiers that somehow bring out the local structure of the dataset, enabling us to choose a value for σ that captures this locality. The first cheap classifier is based on K-Nearest Neighbours (Cover and Hart, 1967). The second one is based on the distance from each data point to its nearest opposite-class neighbour. The third one employs a fast clustering algorithm to extract the local geometry of the problem.

The three methods compare favourably against 10-fold cross-validation in the selection of the σ in several UCI classification tasks (Blake and Merz, 1998).

The rest of the paper is organised as follows. Section 2 reviews the SVM and the algorithms that are used to implement the proposed σ estimation methods. Section 3 motivates and presents in detail the three estimation algorithms. Section 4 includes an empirical evaluation of the SVM classification performance under the proposed model selection schemes; we use 10-fold cross-validation as baseline method. Finally, Section 5 closes the paper with the main conclusions and the description of our ongoing research.

2. Background

This section includes a brief review of all the techniques and algorithms employed in the proposed methods to estimate σ.

2.1. Machine classification

Consider a classification problem defined in terms of a set of labelled examples $\{(\mathbf{x}_i,y_i)\}_{i=1}^n$, with observations $\mathbf{x}_i \in \mathbb{R}^d$ and labels $y_i \in \{1,2,\ldots,L\}$ indicating the class to which observation $\mathbf{x}_i$ belongs. There are two general approaches to tackle the design of an automatic classifier for this problem (Webb, 2002)

- *Generative approach*: Data are used to learn a probabilistic model for the distribution of each class. This model consists of the following probability density functions: $p(\mathbf{x}|y=l)$ and $p(y=l)$, $l=1,\ldots,L$. The classifier results from the application of the Bayes optimal classification rule

  $\hat{y}(\mathbf{x}) = \arg\max_l p(y=l|\mathbf{x})$

- *Discriminative approach*: One selects a parametric classification rule (also denominated discriminant function) $f_{\mathbf{w}}(\mathbf{x})$ whose output determines the class of instance $\mathbf{x}$, and uses the data to fit the parameters $\mathbf{w}$.

Despite the optimality underlying the former, in practice the latter is more common since the fitting of the discriminant function posses an easier optimisation problem (with a smaller number of variables to optimise) than the learning of a more or less complex probabilistic model. This intuition can be also borrowed from daylife experience; it is well known that children are able to separate dogs from cats without coming up with a precise definition of each class.

A machine learning version of Occam's Razor principle translates to stick to the simplest discriminant function able to solve the classification problem. This principle helps to prevent from a bad generalisation due to overfitting. One of the simplest discriminant functions for binary problems is linear classifiers

$$o(\mathbf{x}) = \text{sign}\{f(\mathbf{x})\} = \text{sign}\{\mathbf{w}^T\mathbf{x}+b\} \quad (1)$$

since the number of parameters is equal to the size of the observations (plus one). Moreover, it is well known from the machine learning community experience that linear classifiers work reasonably well in a good number of applications. Linear classifiers as formulated in Eq. (1) solve binary problems (they are able to discriminate between two classes). The linear classification rule defines a separating hyperplane (or boundary) that divides the input space in two halfspaces, each one associated with one output class (see Fig. 1). For one class, usually termed positive class, one expects that $f(\mathbf{x})>0$, whilst for the other class, termed negative class, one expects that $f(\mathbf{x})<0$. The quantity $yf(\mathbf{x})$ measures how well is $\mathbf{x}$ classified if y is its label. A much greater than zero value of $yf(\mathbf{x})$ indicates that $\mathbf{x}$ is on the correct side of the boundary and well away from it. Analogously, a negative value of $yf(\mathbf{x})$ indicates that the pattern is on the wrong side of the boundary. The more negative the value of $yf(\mathbf{x})$ is, the further from the boundary (and from being correctly classified) $\mathbf{x}$ lies.

Nevertheless, it is straightforward to reformulate any multi-class problem as a pool of binary classifications following one of these strategies:

One vs. all: In a problem with L classes, the multiclass classifier is composed of L classifiers, each one trained to separate one class from the rest. Each pattern is fed to all the classifiers and assigned to the class corresponding to the classifier whose classification is more reliable. In the linear classification case of Eq. (1) the most usual criterion is to assign the pattern to the class with largest $f(\mathbf{x})$, since the larger this number is, the more clearly classified as positive sample is $\mathbf{x}$.

One vs. one: In a problem with L classes, this strategy involves training $L(L-1)/2$ binary classifiers, each one trained to separate a pair of input classes. Since each class can be predicted by at most $L-1$ classifiers, the usual overall classification rule is to assign each pattern to the most voted class among the $L(L-1)/2$ classifiers.

The training of a linear classifier for a linearly separable problem consists in determining the value of its weight vector $\mathbf{w}$ and bias term b. It usually involves the optimisation of a functional that includes a penalty term favoring a reduced number of errors among the training data set plus some regularisation terms that ensure a good generalisation capability (good classification rates with testing

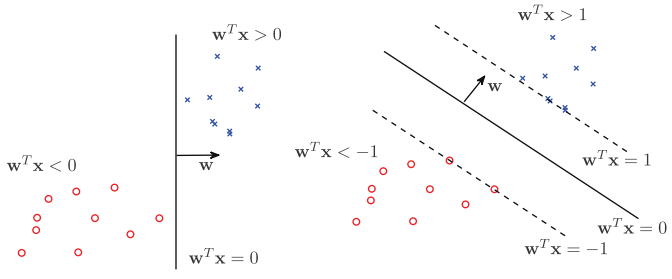

Fig. 1. Two different linear discriminant functions classifying the same dataset. The weight vector $\mathbf{w}$ defines the separating plane. Both separating planes have zero errors in the training set, but the maximal margin plane on the right presents a smaller risk of incurring in misclassification of test samples. In the right hand case, the dotted lines represent the classification margin.