Neurocomputing 131 (2014) 43–51

Contents lists available at ScienceDirect

Neurocomputing

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

Learning vector quantization for (dis-)similarities

Barbara Hammer*, Daniela Hofmann, Frank-Michael Schleif, Xibin Zhu

CITEC Center of Excellence, Bielefeld University, Germany

ARTICLE INFO

Article history:
Received 30 October 2012
Received in revised form
5 March 2013
Accepted 8 May 2013
Available online 27 November 2013

Keywords:
Learning vector quantization
Pseudo-Euclidean space
Dissimilarity data

ABSTRACT

Prototype-based methods often display very intuitive classification and learning rules. However, popular prototype based classifiers such as learning vector quantization (LVQ) are restricted to vectorial data only. In this contribution, we discuss techniques how to extend LVQ algorithms to more general data characterized by pairwise similarities or dissimilarities only. We propose a general framework how the methods can be combined based on the background of a pseudo-Euclidean embedding of the data. This covers the existing approaches kernel generalized relevance LVQ and relational generalized relevance LVQ, and it opens the way towards two novel approach, kernel robust soft LVQ and relational robust soft LVQ. Interestingly, also unsupervised prototype based techniques which are based on a cost function can be put into this framework including kernel and relational neural gas and kernel and relational self-organizing maps (based on Heskes' cost function). We demonstrate the performance of the LVQ techniques for similarity or dissimilarity data in several benchmarks, reaching state of the art results.

1. Introduction

Since electronic data sets increase rapidly with respect to size and complexity, humans have to rely on automated methods to access relevant information from such data. Apart from classical statistical tools, machine learning has become a major technique in the context of data processing since it offers a wide variety of inference methods. Today, a major part of applications is concerned with the inference of a function or classification prescription based on a given set of examples, accompanied by data mining tasks in unsupervised machine learning scenarios and more general settings as tackled, e.g. in the frame of autonomous learning. In this contribution, we focus on classification problems.

There exist many different classification techniques in the context of machine learning ranging from symbolic methods such as decision trees to statistical methods such as Bayes classifiers. Because of its often excellent classification and generalization performance, the support vector machine (SVM) constitutes one of the current flagships in this context, having its roots in learning theoretical principles as introduced by Vapnik and colleagues [4]. Due to its inherent regularization of the result, it is particularly suited if high dimensional data are dealt with. Further, the interface to the data is given by a kernel matrix such that, rather than relying on vectorial representations, the availability of the Gram matrix is sufficient to apply this technique.

With machine learning techniques becoming more and more popular in diverse application domains and the tasks becoming more and more complex, there is an increasing need for models which can easily be interpreted by practitioners: for complex tasks, often, practitioners do not only apply a machine learning technique but also inspect and interpret the result such that a specification of the tackled problem or an improvement of the model becomes possible [39]. In this setting, a severe drawback of many state-of-the-art machine learning tools such as the SVM occurs: they act as black-boxes. In consequence, practitioners cannot easily inspect the results and it is hardly possible to change the functionality or assumptions of the model based on the result of the classifier.

Prototype-based methods enjoy a wide popularity in various application domains due to their very intuitive and simple behavior: they represent their decisions in terms of typical representatives contained in the input space and a classification is based on the distance of data as compared to these prototypes [19]. Thus, models can be directly inspected by experts since prototypes can be treated in the same way as data. Popular techniques in this context include standard learning vector quantization (LVQ) schemes and extensions to more powerful settings such as variants based on cost functions or metric learners such as robust soft LVQ (RSLVQ) or generalized LVQ (GLVQ), for example [32,37,38,36]. These approaches are based on the notion of margin optimization similar to SVM in case of GLVQ [37], or based on a likelihood ratio maximization in case of RSLVQ [38]. For GLVQ and RSLVQ, a behavior which closely resembles standard LVQ2.1 results in limit cases. The limit case of RSLVQ does not necessarily achieve optimum behavior already in simple model situations similar to LVQ2.1, as has been investigated in the

* Corresponding author.
E-mail address: bhammer@techfak.uni-bielefeld.de (B. Hammer).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.05.054

context of the theory of online learning [2]. Nevertheless, it displays excellent generalization ability in the standard intermediate case, see e.g. [36] for an extensive comparison of the techniques.

With data sets becoming more and more complex, input data are often no longer given as simple Euclidean vectors, rather structured data or dedicated formats can be observed such as sequences, graphs, tree structures, time series data, functional data, and relational data as occurs in bioinformatics, linguistics, or diverse heterogeneous databases. Several techniques extend statistical machine learning tools towards non-vectorial data: kernel methods such as SVM using structure kernels, recursive and graph networks, functional methods, relational approaches, and similar [9,33,11,31,14].

Recently, popular prototype-based algorithms have also been extended to deal with more general data. Several techniques rely on a characterization of the data by means of a matrix of pairwise similarities or dissimilarities only rather than explicit feature vectors. In this setting, median clustering as provided by median self-organizing maps, median neural gas, or affinity propagation characterizes clusters in terms of typical exemplars [10,20,8]. More general smooth adaptation is offered by relational extensions such as relational neural gas or relational learning vector quantization [13]. A further possibility is offered by kernelization such as proposed for neural gas, self-organizing maps, or different variants of learning vector quantization [29,5,30]. By formalizing the interface to the data as a general similarity or dissimilarity matrix, complex structures can be easily dealt with: structure kernels for graphs, trees, alignment distances, string distances, etc. open the way towards these general data structures [27,11].

In this contribution, we consider the question how to extend cost function based LVQ variants such as RSLVQ (11) or GLVQ (2) to similarity or dissimilarity data, respectively. We propose a general way based on an implicit pseudo-Euclidean embedding of the data, and we discuss in how far instantiations of this framework differ from each other. Using this framework, we cover existing techniques such as kernel GLVQ [30] and relational GLVQ [15], and investigate novel possibilities such as kernel and relational RSLVQ. These techniques offer valid classifiers and training methods for an arbitrary symmetric similarity or dissimilarity. Some mathematical properties, however, such as an interpretation via a likelihood ratio or interpretation of learning as exact gradient, are only guaranteed in the Euclidean case for some of the possible choices, as we will discuss in this paper. In this context, we investigate the effect of corrections of the matrix to make data Euclidean. The effectivity of the novel techniques is demonstrated in a couple of benchmarks.

Now, we first introduce standard LVQ for Euclidean vectors, in particular the two cost-function based variants GLVQ and RSLVQ. Afterwards, we review facts about similarity and dissimilarity data and their pseudo-Euclidean embedding. Based on this embedding, kernel and relational variants of LVQ can be introduced for similarities or dissimilarities. Training can take place essentially in two ways, mimicking the corresponding Euclidean counterparts or via direct gradients, whereby the same local optima of the cost function are present in the Euclidean case, but a numerical scaling of the gradients is observed. We exemplarily derive new models, kernel RSLVQ and relational RSLVQ in this framework. Experiments are based on the setting as proposed in [6], investigating the effect of different preprocessing steps and learning techniques in comparison to the results of SVM and a k-nearest neighbor classifier. We conclude with a discussion.

2. Learning vector quantization

Learning vector quantization (LVQ) constitutes a very popular class of intuitive prototype based learning algorithms with successful applications ranging from telecommunications to robotics [19]. Basic algorithms as proposed by Kohonen include LVQ1 which is directly based on Hebbian learning, and improvements such as LVQ2.1, LVQ3, or OLVQ which aim at a higher convergence speed or better approximation of the Bayesian borders. These types of LVQ schemes have in common that their learning rule is essentially heuristically motivated and a valid cost function does not exist [3]. One of the first attempts to derive LVQ from a cost function can be found in [32] with an exact computation of the validity at class boundaries in [36]. Later, a very elegant LVQ scheme which is based on a probabilistic model and which can be seen as a more robust probabilistic extension of LVQ2.1 has been proposed in [38]. We shortly review these two proposals.

2.1. Generalized learning vector quantization

Assume data $\xi_i \in \mathbb{R}^n$ with $i=1,\ldots,N$ are labeled y_i where labels stem from a finite number of different classes. A GLVQ network is characterized by m prototypes $w_j \in \mathbb{R}^n$ with priorly fixed labels $c(w_j)$. Classification takes place by a winner takes all scheme:

$$\xi \mapsto c(w_j) \quad \text{where } d(\xi, w_j) \text{ is minimum} \tag{1}$$

with squared Euclidean distance $d(\xi, w_j) = \|\xi - w_j\|^2$, breaking ties arbitrarily.

For training, it is usually assumed that the number and classes of prototypes are fixed. In practice, these are often determined using cross-validation, or a further wrapper technique is added to obtain model flexibility. Training aims at finding positions of the prototypes such that the classification accuracy of the training set is optimized. GLVQ also takes the generalization ability into account, using the costs

$$\sum_i \frac{d(\xi_i, w^+) - d(\xi_i, w^-)}{d(\xi_i, w^+) + d(\xi_i, w^-)} \tag{2}$$

where w^+ constitutes the closest prototype with the same label as ξ_i and w^- constitutes the closest prototype with a different label than ξ_i. The nominator is negative iff ξ_i is classified correctly, thus GLVQ tries to maximize the number of correct classifications. In addition, it aims at an optimization of the hypothesis margin $d(\xi_i, w^-) - d(\xi_i, w^+)$ which determines the generalization ability of the method [37].

Training takes place by a simple stochastic gradient descent, i.e. given a data point ξ_i, adaptation takes place via

$$\Delta w^+ \sim -\frac{2 \cdot d(\xi_i, w^-)}{(d(\xi_i, w^+) + d(\xi_i, w^-))^2} \cdot \frac{\partial d(\xi_i, w^+)}{\partial w^+} \tag{3}$$

$$\Delta w^- \sim \frac{2 \cdot d(\xi_i, w^+)}{(d(\xi_i, w^+) + d(\xi_i, w^-))^2} \cdot \frac{\partial d(\xi_i, w^-)}{\partial w^-} \tag{4}$$

From an abstract point of view, we can characterize GLVQ as a classifier, which classification rule is based on a number of quantities

$$D(\xi, w) := (d(\xi_i, w_j))_{i=1,\ldots,N, j=1,\ldots,m} \tag{5}$$

Training aims at an optimization of a cost function of the form

$$f(D(\xi, w)) \tag{6}$$

by means of the gradients

$$\frac{\partial f(D(\xi, w))}{\partial w_j} = \sum_{i=1}^{m} \frac{\partial f(D(\xi, w))}{\partial d(\xi_i, w_j)} \cdot \frac{\partial d(\xi_i, w_j)}{\partial w_j} \tag{7}$$

with respect to the prototypes w_j or the corresponding stochastic gradients for one point ξ_i.