Pattern Recognition 76 (2018) 349–366

Contents lists available at ScienceDirect

Pattern Recognition

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

Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces

Brijnesh J. Jain*, David Schultz

TU Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

ARTICLE INFO

Article history:
Received 26 May 2017
Revised 4 September 2017
Accepted 23 October 2017
Available online 7 November 2017

Keywords:
Learning vector quantization
Time series
Dynamic time warping

ABSTRACT

The nearest neighbor method together with the dynamic time warping (DTW) distance is one of the most popular approaches in time series classification. This method suffers from high storage and computation requirements for large training sets. As a solution to both drawbacks, this article extends learning vector quantization (LVQ) from Euclidean spaces to DTW spaces. The proposed generic LVQ scheme uses asymmetric weighted averaging as update rule. We theoretically justify the asymmetric LVQ scheme via subgradient techniques and by the margin-growth principle. In addition, we show that the decision boundary of two prototypes from different classes is piecewise quadratic. Empirical results exhibited superior performance of asymmetric generalized LVQ (GLVQ) over other state-of-the-art prototype generation methods for nearest neighbor classification.

1. Introduction

The nearest neighbor (NN) classifier endowed with the dynamic time warping (DTW) distance is one of the most popular methods in time series classification [10,48]. Application examples include electrocardiogram frame classification [17], gesture recognition [2,36], speech recognition [27], and voice recognition [26].

Two disadvantages of the naive NN method are high storage and computation requirements. Storage requirements are high, because the entire training set needs to be retained for being able to execute its classification rule. Computation requirements are high, because classifying a test example demands calculation of DTW distances between the test and all training examples. One solution to both limitations are data reduction methods that infer a small set of prototypes from the training examples [46]. These methods aim at scaling down storage and computational complexity, while maintaining high classification accuracy. Two common reduction approaches are prototype selection [11] and prototype generation [44]. Prototype selection methods choose a suitable subset of the original training set. Examples of prototype selection algorithms that have been applied in DTW spaces are min-max centroids [33], k-Medoids [20,31,32] and the DROP-family [46,47].

Prototype generation methods infer new artificial prototypes from the training examples. We distinguish between three directions to prototype generation methods for NN classification in DTW spaces:

1. *Unsupervised prototype generation* [1,29,32–34,39,43,45]: Unsupervised methods cluster the training examples of every class separately. Centroids of the clusters are computed by averaging warped time series, which is non-trivial as compared to averaging vectors [30,40]. The resulting centroids form a reduced set of prototypes for NN classification.
2. *Symmetric LVQ1* [42]: Symmetric LVQ1 is a supervised prototype generation method that extends the LVQ1 algorithm [22] from Euclidean to DTW spaces by using a symmetric update rule. Starting with an initial set of prototypes, LVQ1 repeatedly applies the following steps: (i) select the next training example x; (ii) identify the prototype p closest to x; and (iii)“attract” p to x if the class labels of both agree, otherwise “repel” p from x.
3. *Relational LVQ* [12,14,15,25]: Relational LVQ methods extend state-of-the-art LVQ methods from Euclidean spaces to pseudo-Euclidean spaces via pairwise dissimilarity data. An important example that has been extended to relational learning is generalized LVQ (GLVQ) [38]

An empirical comparison of different methods across the three directions is missing. Consequently, it is unclear which direction is best suited for which situation. Relational methods are theoretically best developed and the most general approach, because they can be applied to any distance space.

Unsupervised methods are currently the most popular direction in DTW spaces. Their usefulness has been first demonstrated in the 1970ies for speech recognition [33,34] and recently

* Corresponding author.
E-mail address: jain@dai-labor.de (B.J. Jain).

https://doi.org/10.1016/j.patcog.2017.10.029
0031-3203/© 2017 Elsevier Ltd. All rights reserved.

confirmed for general time series classification tasks [32,39]. Moreover, empirical result showed that k-means together with the DBA algorithm for time series averaging exhibited the best generalization performance over different prototype selection and unsupervised prototype generation methods [31,32]. A limitation of unsupervised methods is that they learn prototypes of every class separately without considering the decision boundaries to the respective neighboring classes.

In contrast to unsupervised methods, supervised approaches aim at directly approximating the true but unknown decision boundaries. As far as we know, symmetric LVQ1 [42] is the only existing supervised prototype generation method that operates in DTW spaces. However, there are four major issues related to symmetric LVQ1:

1. The update rule of the Euclidean LVQ1 method can be formulated as a weighted average of two points. In DTW spaces there are two common forms of averages: a symmetric and an asymmetric form [23,40]. The symmetric form computes the arithmetic mean of warped time series, whereas the asymmetric form is a minimizer of a sum of squared DTW distance criterion [18,40]. The asymmetric mean is theoretically better grounded than the symmetric mean [40] and exhibited superior performance in unsupervised learning [30,41].
2. A basic principle of LVQ methods in Euclidean spaces is to attract a selected prototype p to the current input x if the class labels of both agree, otherwise repel p from x. We call this principle the margin-growth principle. Symmetric LVQ1 in DTW spaces emulates the Euclidean LVQ1 update rule but in fact lacks a theoretical guarantee for satisfying the margin-growth principle.
3. Symmetric LVQ1 has been proposed as a heuristic without explicit link to a cost function. It extends the first and simplest LVQ variant formulated in Euclidean spaces [22]. The Euclidean LVQ1 showed unsatisfactory generalization performance, slow convergence, and numerical instabilities. Improved methods such as GLVQ define explicit cost functions that are minimized by stochastic gradient descent [38].
4. As we will show in Section 3, it is possible to extend cost-based LVQ methods such as GLVQ from Euclidean spaces to DTW spaces using symmetric update rules. Since a theoretical foundation is missing, it is unclear whether and according to which principles symmetric updating minimizes a cost function.

Due to the better theoretical foundation and the above four issues, it seems natural to ask whether asymmetric extensions of LVQ methods are beneficial for supervised prototype generation in DTW spaces.

The contributions of this article are as follows:

1. We propose a generic asymmetric LVQ scheme for DTW spaces. Under mild assumptions, every Euclidean LVQ algorithm can be directly extended to DTW spaces. As examples, we derive asymmetric versions of LVQ1 and GLVQ.
2. In a theoretical analysis, we first show that the decision boundary defined by two prototypes from different classes is piecewise quadratic. Then we relate asymmetric LVQ to stochastic subgradient methods and prove that asymmetric LVQ satisfies the margin-growth principle almost everywhere.
3. In experiments, we compare the performance of the proposed asymmetric LVQ1 and GLVQ method against prototype generation methods from the above mentioned three directions. The experiments fill the gap of a missing comprehensive study of the performance of different prototype generation methods across different directions for nearest neighbor classification. The main finding is that asymmetric GLVQ best trades classification accuracy against computation time by a large margin.

The implications of this contribution are twofold: First, asymmetric GLVQ is well suited for online settings and in situations where storage and computation requirements are an issue. Second, the generic asymmetric LVQ scheme can serve as a blueprint for directly extending unsupervised prototype learning methods to DTW spaces, such as vector quantization, self-organizing maps, and neural gas.

The rest of this paper is structured as follows: Section 2 introduces LVQ in Euclidean space. Section 3 proposes asymmetric LVQ for DTW spaces. Section 4 presents the theoretical results and Section 5 discusses experiments. Finally, Section 6 concludes with a summary of the main results and an outlook to further research.

2. Learning vector quantization

This section introduces learning vector quantization in Euclidean spaces in such a way that most concepts can be directly extended to other distance spaces.

2.1. Nearest neighbor classification

Let $(\mathcal{X}, \delta)$ be a distance space with distance function $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$. Furthermore, let $\mathcal{Y} = \{1, \ldots, C\}$ be a set consisting of C class labels. We assume that there is an unknown labeling function

$$\ell : \mathcal{X} \to \mathcal{Y}, \quad x \mapsto \ell(x)$$

that assigns a class label $\ell(x) \in \mathcal{Y}$ to every element $x \in \mathcal{X}$. A nearest neighbor classifier approximates the unknown function ℓ by using a set $\mathcal{C} = \{(p_1, z_1), \ldots, (p_K, z_K))\}$ of K reference elements $p_k \in \mathcal{X}$ with class labels $z_k = \ell(p_k)$. The set $\mathcal{C}$ is called *codebook* and the elements p_k are the *prototypes*. We demand that there is at least one prototype for every class, that is $\mathcal{Y} = \{z_1, \ldots, z_K\}$. For the sake of convenience, we occasionally write $p \in \mathcal{C}$ instead of $(p, z) \in \mathcal{C}$.

Suppose that

$$p_*(x) \in argmin_{p \in \mathcal{C}} \delta(p, x)$$

denotes a nearest prototype $p_*(x) \in \mathcal{C}$ of $x \in \mathcal{X}$. Then the nearest neighbor classifier with respect to codebook $\mathcal{C}$ is a function $h_{\mathcal{C}} : \mathcal{X} \to \mathcal{Y}$ of the form $h_{\mathcal{C}}(x) = \ell(p_*(x))$. The function $h_{\mathcal{C}}(x)$ assigns element x to the class of its nearest prototype $p_*(x)$. To ensure that $h_{\mathcal{C}}$ is well-defined, we assume that $p_*(x)$ is a deterministic selection in case of ties (the nearest prototype is not uniquely determined).

2.2. Learning vector quantization

Let $\mathcal{X} = \mathbb{R}^d$ be the d-dimensional Euclidean space endowed with the Euclidean metric $\delta(p, x) = \|p - x\|$. Suppose that $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \subseteq \mathcal{X} \times \mathcal{Y}$ is a training set. The goal of LVQ is to learn a codebook $\mathcal{C}$ of size $K \ll N$ on the basis of the training set $\mathcal{D}$ such that the corresponding nearest neighbor classifier $h_{\mathcal{C}}$ minimizes the expected classification error on unseen data.

Learning can be performed in batch or incremental (stochastic) mode. Here, we focus on incremental learning. During learning, the prototypes p are adjusted in accordance with the distortion

$$D_x(p) = \delta^2(p, x),$$

where x is the current input example. The distortion $D_x(p)$ is differentiable as a function of p. Its gradient is given by $\nabla D_x(p) = 2(p - x)$. Algorithm 1 outlines the generic LVQ algorithm.

To explain Algorithm 1, we assume that $p = p_k$ and $p' = p'_k$. In every iteration, Algorithm 1 first randomly selects a training example $(x, z) \in \mathcal{D}$ and then updates the codebook. Update rule (1) moves the current prototype p to a new prototype p' in a direction defined by the gradient $\nabla D_x(p) = 2(p - x)$. The learning rate