Knowledge and Information Systems (2021) 63:589–619
https://doi.org/10.1007/s10115-020-01533-5

REGULAR PAPER

Warped softmax regression for time series classification

Brijnesh Jain[1]

Received: 22 October 2019 / Revised: 3 November 2020 / Accepted: 7 November 2020 /
Published online: 2 January 2021

Abstract
Linear models are a mainstay in statistical pattern recognition but do not play a role in time series classification, because they fail to account for temporal variations. To overcome this limitation, we combine linear models with dynamic time warping (dtw). We analyze the resulting warped-linear models theoretically and empirically. The three main theoretical results are (i) the Representation Theorem, (ii) the Matrix Complexity Lemma, and (iii) local Lipschitz continuity of the warped softmax function. The Representation Theorem roughly states that warped-linear models correspond to polytope classifiers in Euclidean spaces. This key result is useful because it simplifies analysis of warped-linear models. For example, it provides a geometric interpretation, points to the label dependency problem, and justifies application of warped-linear models not only on temporal but also on multivariate data. The Representation Theorem together with the Matrix Complexity Lemma reveals that warped-linear models implement a weight trick by weight selection and massive weight sharing. Local Lipschitz continuity of warped softmax functions admits a principled training of warped-linear models by stochastic subgradient methods. Empirical results show that replacing the inner product of linear models with a dtw-score substantially improves its predictive performance. The theoretical and empirical contributions of this article provide a simple and efficient first-trial alternative to nearest-neighbor methods and open up new perspectives for more sophisticated classifiers such as warped deep learning.

Keywords Time series · Dynamic time warping · Linear classifier · Softmax regression

1 Introduction

Two simple mainstays in pattern recognition with opposite characteristics are linear models and the k-nearest neighbors method [10]. The situation is different for time series classification. Here, the prime approach is the nearest-neighbor (NN) classifier using the *dynamic time warping* (dtw) distance [1,5], whereas linear models that directly operate on time series do not matter.

One explanation for the different significance of both approaches in time series classification is as follows: Often, time series vary in speed and phase. Lock-step measures such

✉ Brijnesh Jain
brijnesh.jain@oth-regensburg.de

[1] Department of Computer Science and Mathematics, OTH Regensburg, Regensburg, Germany

as the Euclidean distance are sensitive to these variations, because the element-wise comparison between time series is fixed. In contrast, elastic measures such as the *dynamic time warping* (dtw) distance [6,26] account for temporal variations by non-linearly aligning the elements of both time series to be compared (see Fig. 1). Empirical results on time series classification problems suggest that NN classifiers with elastic distances often have better predictive performance than NN classifiers with lock-step measures [5,23]. The Euclidean distance is induced by an inner product, and the inner product is the defining component of linear models. In the same way as the Euclidean distance, the inner product is also a lock-step measure and therefore too inflexible to cope with temporal variations. This in turn adversely affects linear models for time series classification and partly explains their uncompetitive predictive performance [5].

Replacing the Euclidean distance with the dtw-distance substantially improved the predictive performance of NN classifiers in time series classification. Inspired by this finding, elastic linear classifiers have been proposed that replace the inner product in linear models by a dtw-similarity score [11]. Elastic linear models have been tested on two-category problems only. The empirical results suggest that (i) elastic linear classifiers can better cope with temporal variations than linear models, (ii) the predictive performance of elastic linear classifiers is comparable to dtw-NN classifiers, and (iii) elastic linear classifiers are computationally much more efficient than dtw-NN classifiers. As opposed to the well-explored linear models, a theoretical understanding of elastic linear models is missing. Consequently, elastic linear models fall short of their potential as we will see in the course of this article. In addition, the behavior and performance of elastic linear models on multi-category problems is unknown.

In this article, we narrow the theoretical and empirical gap of elastic linear classifiers. We propose warped-linear models as a slight but more convenient modification of elastic linear classifiers. The main theoretical result is the Representation Theorem. It roughly states that replacing the inner product of a linear function by a dtw-score results in a pointwise

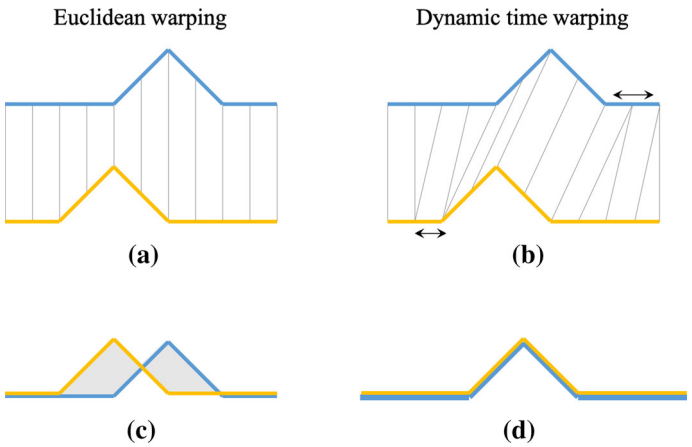

Fig. 1 Euclidean versus dtw-distance between two time series that are out of phase. Plot **a** shows the alignment between elements by the Euclidean distance and Plot **b** by the dtw-distance. The Euclidean distance aligns the *i*th element of the first with the *i*th element of the second time series. The dtw-distance allows many-to-many alignments but preserves the sequential order. The bi-directional arrows in Plot **b** indicate which segments of a time series are expanded, that is, which elements of a time series are aligned to more than one element of the other time series. The grey-shaded area in Plot **c** reflects the Euclidean distance. Plot **d** shows that the dtw-distance is zero

maximum of a finite set of linear functions. Based on this theorem, we show the following results:

1. Weight trick: We show that warped-linear classifiers almost always implement a weight trick to reduce extensive computations. The weight trick consists in weight selection and massive weight sharing.
2. Polytope classifiers: The Representation Theorem relates warped-linear models to the well-explored family of polytope classifiers, which are based on pointwise maximizers (or minimizers) of a finite set of linear functions [2,9,14,17,18,20,25,27]. This connection provides access to existing results that can be directly applied to warped-linear functions such as geometric interpretations, the label dependency problem, and subgradient optimization.
3. Cross-domain applicability: The Representation Theorem justifies to apply warped-linear models to multivariate data and standard polytope classifiers to temporal data.

To assess the behavior and performance of warped-linear models, we conducted experiments on temporal and multivariate data. For this, we adopted the softmax loss function. We analyzed the label dependency problem, compared warped softmax regression against standard softmax regression, polytope classifiers, and nearest neighbor classifiers with dtw.

2 Preliminaries

This section describes both components we want to combine, dynamic time warping and softmax regression. In addition, we introduce the terminology and notations used throughout this article.

2.1 Warping paths

For a given $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. A real-valued *time series* is a sequence $x = (x_1, \ldots, x_n)$ with elements $x_i \in \mathbb{R}$ for all $i \in [n]$. We denote the length of x by $|x|$ and write $\mathcal{T}$ for the set of all real-valued time series of finite length.

A technique to cope with temporal variations in time series is dynamic time warping. This technique is based on the concept of warping path. To define warping paths, we consider the grid

$$[m] \times [n] = \{(i, j)\, i \in [m],\, j \in [n]\}.$$

A *warping path* through the grid $[m] \times [n]$ is a sequence $p = (p_1, \ldots, p_\ell)$ of ℓ grid-points $p_l = (i_l, j_l) \in [m] \times [n]$ for all $l \in [\ell]$ such that the following conditions are satisfied:

1. $p_1 = (1, 1)$ and $p_\ell = (m, n)$ (boundary conditions)
2. $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for all $l \in [\ell - 1]$. (step condition)

The number ℓ is the length of p. We denote the set of all warping paths through $[m] \times [n]$ by $\mathcal{P}_{m,n}$. Figure 2 shows examples of warping paths through the grid $[3] \times [3]$. The boundary conditions enforce that the path starts at the upper left corner and ends in the lower right corner of the grid. The step condition demands that a transition from one point to the next point moves exactly one unit in either down, right, or diagonal direction.

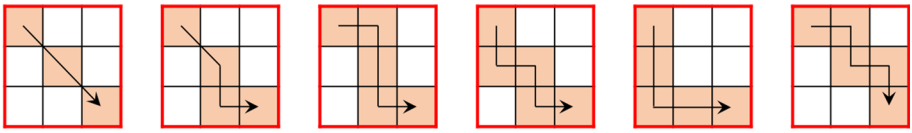

Fig. 2 Examples of warping paths through a 3×3 grid. Arrows indicate the directions of the paths

2.2 Softmax regression

Softmax regression is a linear classifier for multi-category problems. It is based on linear functions of the form

$$f : \mathbb{R}^{n-1} \to \mathbb{R}, \quad x \mapsto w^{\mathsf{T}} x + b,$$

where $w \in \mathbb{R}^{n-1}$ is the weight vector and $b \in \mathbb{R}$ is the bias. We represent linear functions $f(x)$ in homogeneous form $f(x) = \tilde{w}^{\mathsf{T}} \tilde{x}$, where $\tilde{w} = (b, w_1, \ldots, w_{n-1}) \in \mathbb{R}^n$ and $\tilde{x} = (1, x_1, \ldots, x_{n-1}) \in \mathbb{R}^n$ are the augmented vectors of w and x, respectively.

Suppose that $\mathcal{X} = \{1\} \times \mathbb{R}^{n-1}$ is the (augmented) input space and $\mathcal{Y} = \{1, \ldots, c\}$ is a set of $c \geq 2$ class labels. Softmax regression models the conditional probabilities $P_\theta(k \mid x)$ of class $k \in \mathcal{Y}$ after we have observed input $x \in \mathcal{X}$ as

$$P_\theta(k \mid x) = \frac{\exp w_k^{\mathsf{T}} x}{\sum_{l \in \mathcal{Y}} \exp w_l^{\mathsf{T}} x},$$

where matrix $\theta = (w_1, \ldots, w_c) \in \mathbb{R}^{n \times c}$ summarizes the c weight vectors w_k. In the following, we refer to $h_k(x) = P_\theta(k \mid x)$ as the softmax function of the kth class.

Let $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \subseteq \mathcal{X} \times \mathcal{Y}$ be a training set consisting of N input examples x_i with corresponding class labels y_i. Then, softmax regression aims at minimizing the negative log-likelihood

$$J(\theta; \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{c} \delta_{k y_i} \cdot \log h_k(x_i),$$

where δ_{ky} is the Kronecker delta satisfying $\delta_{ky} = 1$ if $k = y$ and $\delta_{ky} = 0$, otherwise. One approach to minimize $J(\theta; \mathcal{D})$ is stochastic gradient descent. After observing a training example $(x, y) \in \mathcal{D}$, the stochastic gradient descent algorithm updates the weight vectors w_k according to the rule

$$w_k \leftarrow w_k + \eta \cdot \big(\delta_{ky} - h_k(x)\big) x, \tag{1}$$

where $\eta \in (0, 1]$ is the learning rate. Algorithm 1 in Appendix 2 outlines the stochastic gradient descent method for minimizing the error function $J(\theta; \mathcal{D})$ of softmax regression.

3 Warped-linear functions

In this section, we propose warped-linear functions and analyze their properties.

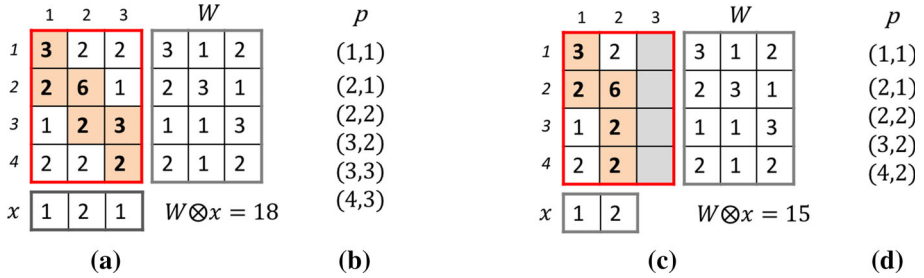

Fig. 3 Examples of warped products $W \otimes x$ with optimal warping paths p. In both examples, the weight matrices $W \in \mathbb{R}^{4\times 3}$ are identical. The number of columns of W bounds the maximum length of admissible time series x to $|x| \leq 3$. Plot **a** shows W and a time series x of maximum length $|x| = 3$ together with the red-bordered (4×3)-matrix $S = (s_{ij})$. The elements of S are the local scores $s_{ij} = w_{ij}x_j$. The sum of the local scores along the orange-colored optimal warping path p yields the warped product $W \otimes x = 3+2+6+2+3+2 = 18$. Plot **b** shows the optimal warping path p in (**a**) as a sequence of points $(i, j) \in p$. Plot **c** depicts the warped product $W \otimes x$ between W and an input time series x of length $|x| = 2 < 3$. In this case, the local score matrix $S = (s_{ij})$ consists of two columns. The sum of the local scores along the highlighted optimal warping path gives $W \otimes x = 3+2+6+2+2 = 15$. Plot (**d**) shows the optimal warping path p in (**c**) as a sequence of points $(i, j) \in p$

3.1 Definition of Warped-linear functions

A warped-linear function combines linear functions with dynamic time warping to obtain a function that has similar properties as a linear function but can cope with temporal variations. To construct a warped-linear function, we replace the inner product of linear functions by a warped product.

For the sake of presentation, we assume that all time series are of fixed length n. The corresponding algorithms listed in Appendix 2 consider the general case of time series with varying length. Since n is fixed, we briefly write $\mathcal{P}_m$ instead of $\mathcal{P}_{m,n}$.

A warped product is a warped version of a sparse matrix-vector product. Suppose that $W \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. A warping path $p \in \mathcal{P}_m$ defines a warping of x into matrix W by aligning element x_j with element w_{ij} if $(i, j) \in p$. The sum

$$s_p(W, x) = \sum_{(i,j)\in p} w_{ij}x_j$$

is the score of warping x into W along p. The *warped product* is a function $\otimes : \mathbb{R}^{m\times n} \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$W \otimes x = \max\left\{s_p(W, x) \,:\, p \in \mathcal{P}_m\right\}$$

for all $W \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. We say, $p \in \mathcal{P}_m$ is an *optimal warping path* if $s_p(W, x) = W \otimes x$, that is $s_p(W, x) \geq s_q(W, x)$ for all $q \in \mathcal{P}_m$.

Figure 3 depicts two examples of warped products and their corresponding optimal warping paths. Algorithm 4 in Appendix 2 presents a dynamic program of complexity $\mathcal{O}(m|x|)$ to compute the warped product $W \otimes x$, where x is a time series of arbitrary length. A necessary condition to apply $W \otimes x$ is that the number n of columns of W is not less than the length of x. This should not be a serious limitation in practice from a conceptual perspective because we can choose n arbitrarily large. Finally, Algorithm 5 describes how to compute an optimal warping path of $W \otimes x$.

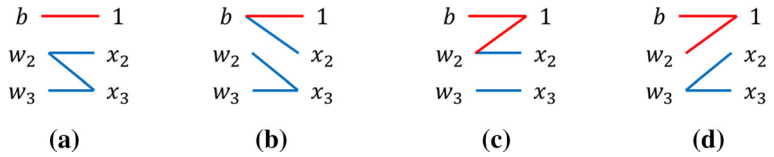

Fig. 4 Different roles of weights and bias. Shown are four different warping paths between an augmented weight matrix $W = (b, w_1, w_2)$ consisting of a single row and an augmented time series $x = (1, x_1, x_2)$. Plot **a** shows the traditional roles of bias and weights, Plot **b** shows that b takes the role of a bias and a weight, Plot **c** shows that w_1 takes the role of a bias and a weight, and finally Plot **d** shows that b and w_1 take both roles, as bias and as weight

To define warped-linear function, we replace the inner product of linear functions by a warped product. Let $\mathcal{X} = \{1\} \times \mathbb{R}^{n-1}$ be the augmented input space. A *warped-linear function* of elasticity m is a function of the form

$$f : \mathcal{X} \to \mathbb{R}, \quad x \mapsto W \otimes x,$$

where $W = (w_{ij}) \in \mathbb{R}^{m \times n}$ is the weight matrix. The number m of rows of W is the *elasticity* of f. The elasticity of a warped-linear function serves to control both, the capacity of the function class and the computational complexity. The first column of W represents the bias $b = (w_{11}, \ldots, w_{m1})$. In this formulation, weights and bias can share their roles as illustrated in Fig. 4. This makes warped-linear functions more flexible than functions of the form $f(x) = W \otimes x + b$, where $b \in \mathbb{R}$ and x is not augmented.

3.2 Warped Softmax regression

In this section, we propose warped softmax regression. We obtain warped softmax regression from softmax regression by replacing linear functions with warped-linear functions.

Let $\mathcal{X} = \{1\} \times \mathcal{T}$ be the augmented input space and let $\mathcal{Y} = \{1, \ldots, c\}$ be a set of class labels. Suppose that $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \subseteq \mathcal{X} \times \mathcal{Y}$ is a training set. Then, warped softmax regression minimizes the non-differentiable and non-convex cost function

$$J(\theta; \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{c} \delta_{ky_i} \cdot \log h_k(x)$$

where $\theta = (W_1, \ldots, W_c)$ collects the augmented weight matrices of the warped softmax regression model and

$$h_k(x) = \frac{\exp W_k \otimes x_i}{\sum_{l=1}^{c} \exp W_l \otimes x_i}$$

is the softmax function of class $k \in \mathcal{Y}$. To minimize the cost function $J(\theta; \mathcal{D})$, we repeatedly update the weights $W_k = (w_{ij}^k)$ of every class $k \in \mathcal{Y}$ in an incremental fashion on the basis of a single training example $(x, y) \in \mathcal{D}$. For this, we first determine an optimal warping path p_k between W_k and x. Then, we apply the update rule

$$w_{ij}^k \leftarrow w_{ij}^k + \eta \cdot \Big(\delta_{ky} - h_k(x)\Big) \cdot x_j, \tag{2}$$

where η is the learning rate and δ_{ky} is the Kronecker-delta (see Sect. 2.2). The update rule adjusts only weights w_{ij}^k of W_k along the optimal warping path p_k. All other weights remain unchanged. Figure 5 illustrates the update rule (2), Theorem 3 theoretically justifies update

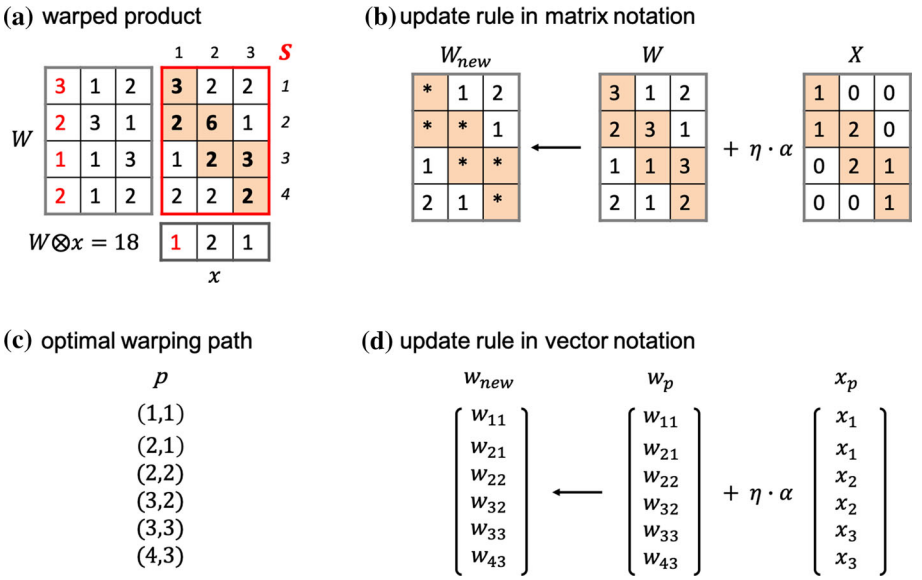

Fig. 5 Illustration of the update rule of warped softmax regression. Plot **a** shows the warped product $W \otimes x$ from Figure 3. We assume that $W = W_k$ is the augmented weight matrix of class k. The first column (red) of W is the bias b. The first element (red) of input x is the constant value 1. The optimal warping path p of $W \otimes x$ is highlighted in the local score matrix (framed in red). Plot **b** shows the optimal warping path p of $W \otimes x$ as a sequence of points (i, j). Plot **c** shows the update rule of warped softmax regression. The value α is a shortcut for the term $\delta_{ky} - h_k$. Only elements from W along the highlighted optimal warping path p are selected and summarized to a vector w_p. Thus, w_{ij} is an element of w_p iff $(i, j) \in p$. The vector x_p is obtained by warping x along p. Thus, x_j is an element of x_p iff $(i, j) \in p$. Finally, w_{new} is the updated weight vector. All other weights of W remain unchanged (color figure online)

rule (2), and Algorithm 4 in Appendix 2 presents a pseudo code of warped softmax regression for time series of varying rather than fixed length.

The update rule (2) is simple but suffers from the following limitations: the convergence behavior is unclear, it is not necessarily a descent method, a practicable termination criterion is unknown, the choice of a suitable learning rate scheme is difficult, and if it converges, the convergence speed may be poor. To cope with these limitations, numerous variations and alternative approaches have been devised [24]. In a practical setting, we recommend to record the best solution found so far and terminate the algorithm after a pre-specified number of epochs (cycles through the training set) without improvement.

3.3 A representation theorem

The key result of this article is the Representation Theorem. It states that a warped-linear function is a pointwise maximum of a finite set of linear functions. This result is useful for two reasons: (i) it considerably simplifies the analysis of warped-linear functions and (ii) it justifies application of warped-linear function not only on temporal but also on multivariate data.

Let $m, n \in \mathbb{N}$ and let $\mathcal{X} = \{1\} \times \mathbb{R}^{n-1}$. A *max-linear function* of *elasticity* m is a function of the form

$$f : \mathcal{X} \to \mathbb{R}, \quad x \mapsto \max \left\{ w_i^\mathsf{T} x \ : \ i \in [m], w_i \in \mathbb{R}^n \right\}.$$

The linear functions $g_i(x) = w_i^\mathsf{T} x$ are the *components* of the max-linear function $f(x)$. Note that the components $g_i(x)$ are in homogeneous form and the number of components is the elasticity of $f(x)$. A max-linear function $f(x)$ of elasticity m is specified by a weight matrix

$$W = (w_1, \ldots, w_m)^\mathsf{T} \in \mathbb{R}^{m \times n}$$

whose m rows $w_i^\mathsf{T} \in \mathbb{R}^n$ are the augmented weights of the ith component $g_i(x)$.

For the sake of convenience, we present a descriptive version of the Representation Theorem. Later in Sect. 4.1, we provide an equivalent but constructive formulation of Theorem 1.

Theorem 1 (Representation Theorem) *Every warped-linear function on $\mathbb{R}^{n-1}$ of elasticity m is a max-linear function on $\mathbb{R}^{n-1}$ of elasticity $m' \leq |\mathcal{P}_m|$.*

We conclude this section with two remarks: First, the Representation Theorem justifies application of warped-linear (max-linear) functions on multivariate (temporal) data. Second, it is unclear whether the converse statement of the Representation Theorem is also valid.

3.4 The weight trick

From the Representation Theorem together with the Matrix Complexity Lemma, it follows that warped-linear functions *almost always* implement a weight trick. The weight trick is a technique to avoid intensive computations by weight selection and massive weight sharing.

Both a warped-linear and max-linear function of elasticity m have $m \cdot n$ adjustable weights, and evaluating either of both functions has complexity $\mathcal{O}(m \cdot n)$. Consider the warped-linear function f with elasticity m. According to the Representation Theorem, we can express f as a max-linear function f' of elasticity m' such that $f \equiv f'$. In the following, we are interested in the magnitude of the number m'.

The Representation Theorem bounds the number m' of linear components by the number $|\mathcal{P}_m|$ of different warping paths through an $m \times n$ grid. The number $|\mathcal{P}_m|$ corresponds to the Delannoy number [4]

$$D_{m,n} = \sum_{k=0}^{\min(m,n)-1} \binom{m-1}{k} \binom{n-1}{k} 2^k. \tag{3}$$

The Delannoy number increases exponentially with m and n. To provide an intuition, Table 1 shows the Delannoy numbers for the smallest values of m and n. Thus, f' is computationally unfeasible for all but the smallest values m and n.

For a fixed number m, the number $C = m \cdot n$ grows linearly and the number $C' = D_{m,n} \cdot n$ grows exponentially in n. Comparing the number of adjustable weights (C vs. C') and the complexities ($\mathcal{O}(C)$ vs. $\mathcal{O}(C')$) suggests that a warped-linear function with $D_{m,n}$ linear components is a compact and efficient implementation of its max-linear representation. The weight trick implemented by such a warped-linear function consists in weight selection and massive weight sharing. Weights are selected along an optimal warping path. Weight sharing means that the weights of the $D_{m,n}$ components overlap. Different components share at least the weights w_{11} and w_{mn} by the boundary conditions of a warping path. Consequently, warped-linear functions have less flexibility than max-linear functions with the same number of linear components and their massive weight sharing results in high interdependence between different component functions.

Table 1 Delannoy numbers $D_{m,n}$ for all $m \in [10]$ and all $n \in [9]$

$m \backslash n$	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1
2	1	3	5	7	9	11	13	15	17
3	1	5	13	25	41	61	85	113	145
4	1	7	25	63	129	231	377	575	833
5	1	9	41	129	321	681	1289	2241	3649
6	1	11	61	231	681	1683	3653	7183	13, 073
7	1	13	85	377	1289	3653	8989	19, 825	40, 081
8	1	15	113	575	2241	7183	19, 825	48, 639	108, 545
9	1	17	145	833	3649	13, 073	40, 081	108, 545	265, 729
10	1	19	181	1159	5641	22, 363	75, 517	224, 143	598, 417

So far, we omitted an important point: What is a typical magnitude of m'? The Representation Theorem only states that $m' \leq D_{m,n}$. Thus, the number m' of linear components can take any value between one and $D_{m,n}$. For example, consider a warped-linear function $f(x) = W \otimes x$ with zero weight matrix $W \in \mathbb{R}^{m \times n}$. Then, $f'(x) = \max\left\{0^\mathsf{T} x\right\}$ is a max-linear representation of f consisting of a single linear component with weight vector $0 \in \mathbb{R}^n$. In this case, we have $m' = 1$. In addition, the warped-linear function f is inefficient in the sense that it uses $m \cdot n$ zero weights, whereas its max-linear representation f requires only n zero weights. This extreme example challenges the general validity of the claim that warped-linear functions implement a weight trick.

Fortunately, the Matrix Complexity Lemma comes at our rescue. It states that max-linear representations of warped-linear functions *almost always* consist of $m' = D_{m,n}$ linear components. Conversely, the case $m' < D_{m,n}$ is *negligible*. The terms *almost always* and *negligible* have precise measure-theoretic meanings. For the sake of convenience, we delegate the necessary measure-theoretic concepts and the Matrix Complexity Lemma to Sect. 4.2.

We conclude this section with illustrating the weight trick. For this, we used the Gunpoint dataset from the UCR repository [7]. The Gunpoint dataset consists of time series of length $n = 150$ from two classes and comes with a pre-specified training and test set. We trained a warped softmax regression model with elasticity $m = 5$ on the training set by using two softmax functions, one for each class.[1] Thus, the warping matrices for each class are of dimension 5×150. After training, we classified all training and all test examples. In doing so, we counted how often a weight was visited by optimally warping a training and test examples into the warping matrices of both classes.

Figure 6 shows the relative frequencies with which the weights of the first class were visited. Only 51% (52%) of the weights were used for classifying the training and test examples. Since the training error is 0%, it is not necessary to exploit the full weight matrix. As indicated by the right heatmap, the training and test examples largely share the same weights but with different relative frequencies. The test error of 14.7% indicates overfitting, that is, the model is too flexible for the given data.

[1] Section3.6 explains why to use two instead of a single softmax function.

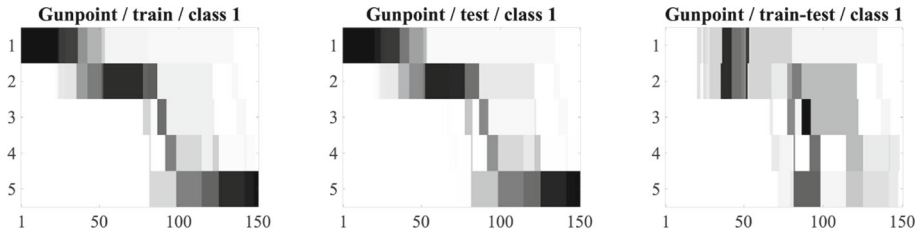

Fig. 6 Heatmaps showing the relative frequencies of visited weights of the first class of the Gunpoint dataset. Darker (brighter) shadings mean more (less) optimal warping paths visited the corresponding weights. Left: Heatmap generated by training examples. Middle: Heatmap generated by test examples. Right: Heatmap of absolute differences between the relative frequencies generated by the training and test set

3.5 Decision boundaries

In this section, we study the form of the decision boundaries that can be implemented by warped-linear functions. By using the Representation Theorem, we can reduce the analysis to decision boundaries of max-linear functions.

We assume that $\mathcal{Y} = \{0, 1\}$ consists of two class labels, where 0 refers to the negative and 1 to the positive class. Suppose that $f : \mathcal{X} \to \mathbb{R}$ is a max-linear function. The surface $\mathcal{S}(f) = \{x \in \mathcal{X} : f(x) = 0\}$ is the decision boundary of f. The decision boundary $\mathcal{S}(f)$ partitions the input space $\mathcal{X}$ into two class regions

$$\mathcal{R}_0(f) = \{x \in \mathcal{X} : f(x) \leq 0\} \quad \text{and} \quad \mathcal{R}_1(f) = \{x \in \mathcal{X} : f(x) > 0\},$$

where the closed set $\mathcal{R}_0(f)$ is the region of the negative class and the open set $\mathcal{R}_1(f)$ is the region of the positive class.

Let $\mathcal{U}, \mathcal{V} \subseteq \mathcal{X}$ be two subsets. We say, $\mathcal{U}$ is *max-lin separable* from $\mathcal{V}$ if there is a max-linear function $f : \mathcal{X} \to \mathbb{R}$ such that

1. $f(x) < 0$ for all $x \in \mathcal{U}$
2. $f(x) > 0$ for all $x \in \mathcal{V}$.

In this case, $\mathcal{U}$ is a subset of the negative class region $\mathcal{R}_0(f)$ and $\mathcal{V}$ is included in the positive class region $\mathcal{R}_1(f)$. We are interested under which conditions two finite sets are max-lin separable. For this, we need the notion of convex hull. The *convex hull* of a finite set $\mathcal{A} = \{x_1, \ldots, x_N\} \subseteq \mathcal{X}$ is defined by

$$\text{conv}(\mathcal{A}) = \left\{\lambda_1 x_1 + \cdots + \lambda_N x_N : \lambda_1 + \cdots + \lambda_N = 1 \text{ and } \lambda_1, \ldots, \lambda_N \in \mathbb{R}_{\geq 0}\right\}.$$

The next result presents a necessary and sufficient condition of max-lin separability.

Proposition 1 *Let $\mathcal{U}, \mathcal{V} \subseteq \mathcal{X}$ be two finite non-empty sets. Then, $\mathcal{U}$ is max-lin separable from $\mathcal{V}$ iff*

$$conv(\mathcal{U}) \bigcap \mathcal{V} = \emptyset.$$

Proof [2], Prop. 2.2.

We show the form of decision boundaries that can be implemented by max-linear functions. From the equivalence of max-lin separability and polytope (polyhedral) separability [18], it follows that the negative class regions $\mathcal{R}_0(f)$ of a max-linear function $f(x)$ on $\mathcal{X}$ are exactly

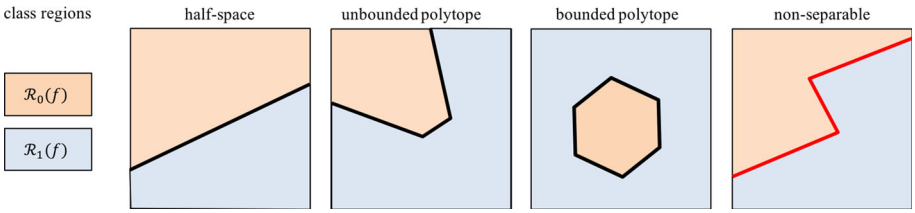

Fig. 7 First three examples show decision boundaries that can be implemented by a max-linear function $f(x)$, because the negative class region $\mathcal{R}_0(f)$ is a convex polytope. The decision boundary of the fourth example cannot be implemented by $f(x)$, because $\mathcal{R}_0(f)$ is non-convex

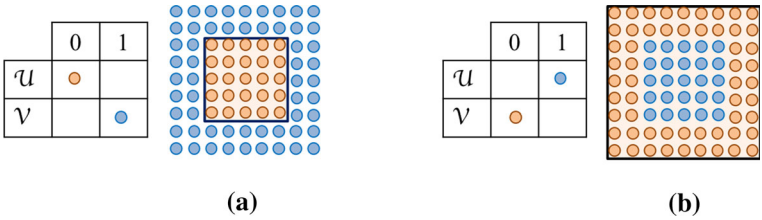

Fig. 8 Label dependency problem of max-linear functions for separating two finite sets $\mathcal{U}$ and $\mathcal{V}$. The elements of $\mathcal{U}$ and $\mathcal{V}$ are depicted by filled circles and can be distinguished by their color. Apricot refers to the negative and blue to the positive class. Plot **a**: The elements of $\mathcal{U}$ are labeled as negative (apricot) and the elements of $\mathcal{V}$ as positive (blue). The convex hull of $\mathcal{U}$ is the shaded square outlined by black lines. The sets $\text{conv}(\mathcal{U})$ and $\mathcal{V}$ are disjoint. Thus, $\mathcal{U}$ is max-lin separable from $\mathcal{V}$. Plot **b**: The elements of $\mathcal{U}$ are labeled as positive (blue) and the elements of $\mathcal{V}$ as negative (apricot). The convex hull of $\mathcal{V}$ includes the set $\mathcal{U}$. Thus, from $\mathcal{U} \subseteq \text{conv}(\mathcal{U})$ follows that $\mathcal{V}$ is not separable from $\mathcal{U}$ (color figure online)

the convex polytopes in $\mathcal{X}$. A convex polytope is the intersection of finitely many closed half-spaces. Figure 7 depicts examples of decision boundaries that can and cannot be implemented by a max-linear function. The decision boundaries implemented by warped-linear functions form a subset of the decision boundaries implemented by max-linear functions. Note that it is unclear whether the subset relationship is proper or not. For further details, we refer to the preprint [12, Prop. 3.6].

3.6 The label dependency problem

Standard softmax regression as introduced in 2.2 is overparameterized. Therefore, it is sufficient to learn $c - 1$ instead of c (augmented) weight vectors. For two-category problems ($c = 2$), it is common practice to learn a single weight vector (logistic regression). Such a weight vector determines a linear discriminant function with the following decision rule: Decide for the negative class, if the discriminant is negative and for the positive class, otherwise. Since the discriminant is linear, we do not care about which of both classes is negative (positive).

The situation is different for max-linear functions. The necessary and sufficient conditions of max-lin separability point to the *label dependency problem*. The label dependency problem is the problem that max-lin separability depends on which class is labeled as negative (positive). As an example, Fig. 8 shows two subsets $\mathcal{U}$ and $\mathcal{V}$ for which "$\mathcal{U}$ *is max-lin separable from* $\mathcal{V}$" but "$\mathcal{V}$ *is* ***not*** *max-lin separable from* $\mathcal{U}$".

The label dependency problem also affects the error rate of max-linear classifiers such as warped softmax regression with a single discriminant function. To demonstrate this, we we

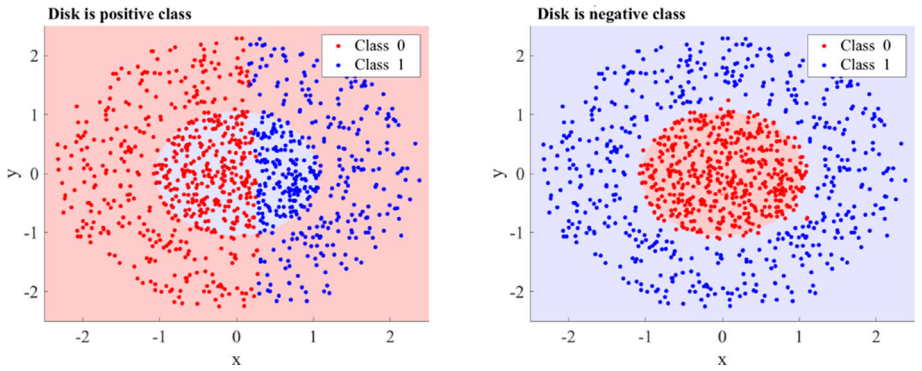

Fig. 9 Effect of different labelings on the classification error. Light blue (red) shaded areas show the true positive (negative) class region. Blue (red) dots show the test points classified as positive (negative) by the classifier (color figure online)

Table 2 Results of

Labeling	WSR_1		WSR_2	
	Train	Test	Train	Test
Disk is positive	45.0	44.2	99.35	98.90
Disk is negative	99.0	98.1	97.10	96.80

consider the task of separating points of a unit disk $\mathcal{D}$ from its complement $\overline{\mathcal{D}}$ (see Fig. 9). We randomly sampled 1000 training and 1,000 test examples with equal class distribution. Then, we labeled the data in two different ways. The first (second) labeling assumes that the unit disk $\mathcal{D}$ represents the positive (negative) class region. For each labeling, we applied warped softmax regresssion (WSR_1) using a single discriminant function $f(x) = W \otimes x$ of elasticity $m = 10$.

Figure 9 illustrates the results of both WSR classifiers subject to different labelings. Table 2 shows the classification accuracies on the training and test set of both WSR_1 classifiers.

For the first labeling (disk is positive), the WSR_1 classifier failed to separate both classes. In contrast, the WSR_1 classifier almost perfectly separated both classes under the second labeling (disk is negative). Thus, the first straightforward approach to cope with the label dependency problem is to select the WSR_1 classifier with the lower error rate (or loss) on the training set. In this example, we would select the WSR_1 model obtained from the second labeling (disk is negative).

For the second straightforward approach, we observe that warped softmax regression is not overparametrized and assume a single softmax regression model with two discriminant functions (WSR_2), one for each class. We also tested this WSR_2 classifier for both labelings. Table 2 shows the classification accuracies on the training and test set. The results show that WSR_2 also almost perfectly separated both classes. Differences of the results for different labelings are due to the random sampling process.

4 Theoretical results

In this section, we prove the Representation Theorem (Theorem 2), the Matrix Complexity Lemma (Lemma 1), and the local Lipschitz continuity of the warped softmax loss function (Theorem 3).

4.1 The representation theorem

This section proves the Representation Theorem (Theorem 2) using a different formulation than in Sect. 3.

The Representation Theorem states that every warped-linear function is a max-linear function that is a pointwise maximizer of a finite set of linear functions. In Definition 1, we construct the weight vectors of the linear component functions. For this, we need the following notation: Let $p \in \mathcal{P}_{m,n}$ be a warping path. For every $j \in [n]$, we define the set $\mathcal{A}_p(j) = \{i \in [m] : (i, j) \in p\}$ consisting of all elements $i \in [m]$ that are aligned to j by warping path p.

Definition 1 Let $W \in \mathbb{R}^{m \times n}$ be a matrix and let $p \in \mathcal{P}_{m,n}$ be a warping path. The *p-embedding* of W along p is a vector $w_p = (\tilde{w}_1, \ldots, \tilde{w}_n) \in \mathbb{R}^n$ with elements

$$\tilde{w}_j = \sum_{i \in \mathcal{A}_p(j)} w_{ij}$$

for all $j \in [n]$.

We use p-embeddings to prove the Representation Theorem.

Theorem 2 (Representation Theorem) *Let $W \in \mathbb{R}^{m \times n}$ and let $x \in \mathbb{R}^n$. Then,*

$$W \otimes x = \max \left\{ w_p^T x \ : \ p \in \mathcal{P}_{m,n} \right\},$$

where w_p is the p-embedding of W along warping path $p \in \mathcal{P}_{m,n}$.

Proof Suppose that $W = (w_{ij})$ and $x = (x_1, \ldots, x_n)$. Let $p \in \mathcal{P}_{m,n}$ be a warping path.

$$w_p^{\mathsf{T}} x = \sum_{j=1}^{n} \tilde{w}_j x_j = \sum_{j=1}^{n} \left(\sum_{i \in \mathcal{A}_p(j)} w_{ij} \right) x_j = \sum_{(i,j) \in p} w_{ij} x_j = s_p(W, x).$$

Then, the assertion follows from

$$W \otimes x = \max \left\{ s_p(W, x) \ : \ p \ \in \mathcal{P}_m \right\} = \max \left\{ w_p^{\mathsf{T}} x \ : \ p \ \in \mathcal{P}_m \right\}.$$

To explicate the Representation Theorem, we assume that $p = (p_1, \ldots, p_\ell)$ is a warping path of length ℓ with points $p_l = (i_l, j_l)$. Then, the score $s_p(W, x)$ can be equivalently expressed as

$$s_p(W, x) = \begin{pmatrix} w_{i_1 j_1} \\ \vdots \\ w_{i_\ell j_\ell} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} x_{j_1} \\ \vdots \\ x_{j_\ell} \end{pmatrix} = \sum_{l=1}^{\ell} w_{i_l j_l} x_{j_l}.$$

This shows that the score $s_p(W, x)$ is an inner product between expanded sequences in the ℓ-dimensional space $\mathbb{R}^\ell$ with $\ell \geq n$. Consequently, the component functions $s_p(W, x)$ of the warped-linear function $f(x) = W \otimes x$ are inner products in different higher-dimensional spaces with dimensions ℓ from $\max(m, n)$ to $m + n - 1$. The Representation Theorem states that these inner products in different higher dimensional spaces can all be reduced to inner products on the common space $\mathbb{R}^n$.

4.2 The matrix complexity Lemma

The *complexity* of weight matrix $W \in \mathbb{R}^{m \times n}$ is the number

$$\chi(W) = \left| \left\{ w_p^{\mathsf{T}} x \ : \ p \in \mathcal{P}_{m,n} \right\} \right|$$

of different linear component functions defined by all possible p-embeddings w_p. By definition, we have

$$\chi(W) \leq \left| \mathcal{P}_{m,n} \right| = D_{m,n},$$

where $D_{m,n}$ is the Delannoy number defined in Eq. (3). The Matrix Complexity Lemma states that $\chi(W) = D_{m,n}$ holds "almost always" in a measure-theoretic sense.

To state the Matrix Complexity Lemma, we need to specify the term "almost always". For this, we identify the isomorphic vector spaces $\mathbb{R}^{m \times n}$ and $\mathbb{R}^d$ with $d = m \cdot n$. Let $\left(\mathbb{R}^d, \mathcal{B}, \mu\right)$ be the Borel–Lebesgue measure space, where $\mathcal{B}$ is the σ-algebra generated by the open sets of $\mathbb{R}^d$ and μ is the Lebesgue-measure on $\mathcal{B}$. A set $\mathcal{N} \subset \mathcal{X}$ is *μ-negligible* if there is a set $\mathcal{N}' \in \mathcal{B}$ such that $\mu(\mathcal{N}') = 0$ and $\mathcal{N} \subseteq \mathcal{N}'$. A property of $\mathcal{X}$ is said to hold *μ-almost always* if the set of points in $\mathcal{X}$ where this property fails is μ-negligible.

Lemma 1 *(Matrix Complexity Lemma) Let $\left(\mathbb{R}^{m \times n}, \mathcal{B}, \mu\right)$ be the Borel–Lebesgue measure space. The property $\chi(W) = D_{m,n}$ holds μ-almost always on $\mathbb{R}^{m \times n}$.*

To prove Lemma 1, we introduce some technicalities. Let $e_1, \ldots, e_d$ be the standard basis of $\mathbb{R}^d$. A trivalent vector $w \in \mathbb{R}^d \backslash \{0\}$ is a nonzero vector of the form

$$w = \lambda_1 e_1 + \cdots + \lambda_d e_d \ \text{ with } \ \lambda_1, \ldots, \lambda_d \in \{-1, 0, 1\}.$$

A vector is reproducible if it is orthogonal to a trivalent vector. We denote the set of reproducible vectors by $\mathcal{R}_d \subseteq \mathbb{R}^d$. Consequently, a weight matrix $W \in \mathbb{R}^{m \times n}$ is reproducible if its vector representation in $\mathbb{R}^d$ is reproducible. Being reproducible is the desired property P outlined above. Then, the following auxiliary results hold:

Lemma 2 *Consider the Borel–Lebesgue measure space $\left(\mathbb{R}^d, \mathcal{B}, \mu\right)$. Then, we have $\mathcal{R}_d \in \mathcal{B}$ with $\mu(\mathcal{R}_d) = 0$.*

Lemma 3 *Let $x \in \mathbb{R}^d$ and let $\lambda \in \{-1, 0, 1\}^d$. If x is non-reproducible and $\lambda \neq 0$, then $\lambda^T x \neq 0$.*

Proof of the matrix complexity Lemma 1

The set $\mathcal{R}_{m \times n}$ of reproducible matrices is μ-negligible by Lemma 2. Thus, the property that a matrix is non-reproducible holds μ-almost always. We prove that every non-reproducible matrix W satisfies $\chi(W) = D_{m,n}$. Then, the property $\chi(W) = D_{m,n}$ also holds μ-almost always.

Let $p = (p_1, \ldots, p_\ell)$ and $q = (q_1, \ldots, q_k)$ be two arbitrary but different warping paths from $\mathcal{P}_{m,n}$. It is sufficient to show that $w_p \neq w_q$.

From $p \neq q$ follows that there is a smallest index λ such that $p_l = q_l$ for all $l \in [\lambda - 1]$ and $p_\lambda \neq q_\lambda$. From the boundary condition of a warping path follows that $1 < \lambda < \min\{\ell, k\}$. Suppose that $p_{\lambda-1} = q_{\lambda-1} = (i, j)$. We set $i' = i + 1$ and $j' = j + 1$. Observe that $i' \leq m$ and $j' \leq n$. From the step condition of a warping path follows that six different combinations of p_λ and q_λ with $p_\lambda \neq q_\lambda$ can occur:

1. $p_\lambda = (i', j)$ and $q_\lambda = (i, j')$
2. $p_\lambda = (i', j)$ and $q_\lambda = (i', j')$
3. $p_\lambda = (i, j')$ and $q_\lambda = (i', j')$
4. $p_\lambda = (i, j')$ and $q_\lambda = (i', j)$
5. $p_\lambda = (i', j')$ and $q_\lambda = (i', j)$
6. $p_\lambda = (i', j')$ and $q_\lambda = (i, j')$

It is sufficient to consider the first three cases for showing $w_p \neq w_q$. The other three cases are symmetric and their proofs are analogous. Let $w_p = \left(w_1^p, \ldots, w_n^p\right)$ be the p-embedding of W along p and let $w_q = \left(w_1^q, \ldots, w_n^q\right)$ be the q-embedding of W along q.

Case 1: $p_\lambda = (i', j)$ and $q_\lambda = (i, j')$. There are $r, s \in \mathbb{N}$ with $0 \leq r$ and $0 < s$ such that $\mathcal{A}_p(j) = \{i - r, \ldots, i + s\}$. Then, by assumption, we have $\mathcal{A}_q(j) = \{i - r, \ldots, i\}$. The jth elements of w_p and w_q are of the form

$$\begin{aligned} w_j^p &= w_{i-r,j} + \cdots + w_{i,j} + w_{i+1,j} + \cdots + w_{i+s,j} \\ w_j^q &= w_{i-r,j} + \cdots + w_{i,j}. \end{aligned}$$

Consider the nonzero matrix $\Lambda = (\lambda_{ij}) \in \{-1, 0, 1\}^{m \times n}$ with $\lambda_{i+1,j} = \cdots = \lambda_{i+s,j} = 1$ and all other entries of Λ are zero. From Lemma 3 follows

$$0 \neq \sum_{i,j} \lambda_{ij} w_{ij} = w_{i+1,j} + \cdots + w_{i+s,j} = w_j^p - w_j^q.$$

This shows $w_p \neq w_q$ for the first case.

Case 2: $p_\lambda = (i', j)$ and $q_\lambda = (i', j')$. The proof is identical to the first case.

Case 3: $p_\lambda = (i, j')$ and $q_\lambda = (i', j')$. There are $r, s \in \mathbb{N}$ with $0 \leq r$ and $0 < s$ such that

$$\begin{aligned} w_{j'}^p &= w_{i,j'} + \cdots + w_{i+r,j'} \\ w_{j'}^q &= w_{i+1,j'} + \cdots + w_{i+s,j'}. \end{aligned}$$

We distinguish between three cases:

$$\begin{aligned} w_{j'}^p - w_{j'}^q &= w_{i,j'} + w_{i+s+1,j'} + \cdots + w_{i+r,j'} && (s < r) \\ w_{j'}^p - w_{j'}^q &= w_{i,j'} && (s = r) \\ w_{j'}^p - w_{j'}^q &= w_{i,j'} - w_{i+r+1,j'} - \cdots - w_{i+s,j'} && (s > r) \end{aligned}$$

Similarly as in Case 1, we have $w_{j'}^p - w_{j'}^q \neq 0$ by Lemma 3 for all three combinations of r and s. Thus, we have $w_p \neq w_q$.

It follows that every non-reproducible W satisfies $\chi(W) = D_{m,n}$ and the proof is complete. □

4.3 Properties of the warped Softmax loss

In this section, we theoretically justify update rule (2) of warped softmax regression. We show that the negative log-likelihood $J(\theta; \mathcal{D})$ of warped softmax regression is locally Lipschitz continuous and repeated application of update rule (2) implements a stochastic subgradient method.

We first introduce the necessary concepts from non-smooth optimization. For details, we refer to [3]. An open ball with center $x_0 \in \mathbb{R}^n$ and radius $r > 0$ is defined by the set $\mathcal{B}(x_0, r) = \{x \in \mathbb{R}^n : \|x - x_0\| < r\}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *locally Lipschitz*

continuous at $x_0 \in \mathbb{R}^n$ if there are numbers $L > 0$ and $\varepsilon > 0$ such that

$$\big|f(x) - f(x')\big| \leq L\big\|x - x'\big\|$$

for all $x, x' \in \mathcal{B}(x_0, \varepsilon)$. The function $f(x)$ is said to be locally Lipschitz continuous, if it is locally Lipschitz continuous at every point $x \in \mathbb{R}^n$. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x_0 \in \mathbb{R}^n$. The *generalized directional derivative* of f at x_0 in the direction of $d \in \mathbb{R}^n$ is defined by

$$f^\circ(x_0; d) = \limsup_{\substack{x \to x_0 \\ t \downarrow 0}} \frac{f(x + td) - f(x)}{t}.$$

As proposed by [8], the (*Clarke*) *subdifferential* of f at x_0 is the set

$$\partial f(x_0) = \Big\{\xi \in \mathbb{R}^n \,:\, f^\circ(x_0; d) \geq \xi^\mathsf{T} d \text{ for all } d \in \mathbb{R}^n\Big\}.$$

Every element $\xi \in \partial f(x_0)$ is called a *subgradient* of f at x_0.

Next, we describe the general form of the cost function of warped softmax regression as a piecewise differentiable function following [22]. Let $\mathcal{U} \subseteq \mathbb{R}^n$ and let $f_1, \ldots, f_k : \mathcal{U} \to \mathbb{R}^m$ be continuous functions. A function $f : \mathcal{U} \to \mathbb{R}^m$ is a *continuous selection* of $f_1, \ldots, f_k$ on the set $\mathcal{V} \subseteq \mathcal{U}$ if f is continuous on $\mathcal{V}$ and $f(x) \in \{f_1(x), \ldots, f_k(x)\}$ for every $x \in \mathcal{V}$.

A function $f : \mathcal{U} \to \mathbb{R}^m$ defined on an open set $\mathcal{U} \subseteq \mathbb{R}^n$ is *piecewise differentiable*, if for every $x_0 \in \mathcal{U}$ there exists an open neighborhood $\mathcal{V} \subseteq \mathcal{U}$ and a finite number of continuously differentiable functions $f_1, \ldots, f_k : \mathcal{V} \to \mathbb{R}^m$ such that f is a continuous selection of $f_1, \ldots, f_k$ on $\mathcal{V}$. The active set $\mathcal{I}_f(x_0)$ and *essentially active set* $\mathcal{I}_f^*(x_0)$ of f at x_0 are defined by

$$\begin{aligned}
\mathcal{I}_f(x_0) &= \big\{i \in [k] \,:\, f(x_0) = f_i(x_0)\big\} \\
\mathcal{I}_f^e(x_0) &= \big\{i \in [k] \,:\, x_0 \in \mathrm{cl}(\mathrm{int}\{x \in \mathcal{U} \,:\, f(x) = f_i(x)\})\big\}.
\end{aligned}$$

The following result states that piecewise smooth functions are locally Lipschitz and their subdifferential is the convex hull of finitely many gradients each of which corresponds to an essentially active selection function [22].

Proposition 2 *Let $f : \mathcal{U} \to \mathbb{R}$ be a piecewise differentiable function with continuously differentiable selection functions $f_1, \ldots, f_k : \mathcal{V} \to \mathbb{R}$ at $x_0 \in \mathcal{V} \subseteq \mathcal{U}$. Then, f is locally Lipschitz continuous and*

$$\partial f(x_0) = conv\big\{\nabla f_i(x_0) \,:\, i \in \mathcal{I}_f(x_0)\big\}.$$

Proof [22, Cor. 4.1.1. and Prop. 4.3.1].

To apply Prop. 2 to the cost function of warped softmax regression, we introduce some additional notations: Let $c \in \mathbb{N}$ and let $\mathcal{P} = \mathcal{P}_{m,n}$. A configuration is a tuple $\pi = (p_1, \ldots, p_c) \in \mathcal{P}^c$ consisting of warping paths $p_k \in \mathcal{P}$ for all $k \in [c]$. Furthermore, let $\Theta = \mathbb{R}^{m \times n}$, let $W = (W_1, \ldots, W_c) \in \Theta^c$, and let $x \in \mathbb{R}^n$. We define the sets

$$\begin{aligned}
\mathcal{I}_x(W) &= \big\{(p_1, \ldots, p_c) \in \mathcal{P}^c \,:\, W_k \otimes x = s_{p_k}(W_k, x) \text{ for all } k \in [c]\big\} \\
\mathcal{I}_x^e(W) &= \big\{\pi \in \mathcal{P}^c \,:\, W \in \mathrm{cl}\big(\mathrm{int}\big\{W' \in \Theta^c \,:\, \pi \in \mathcal{I}(W', x)\big\}\big)\big\}.
\end{aligned}$$

Finally, a vector-to-matrix embedding along warping path $p \in \mathcal{P}_{m,n}$ is a map

$$\Phi_p : \mathbb{R}^n \to \mathbb{R}^{m \times n}, \quad x \mapsto X_p = \left(x_{ij}^p\right)$$

such that the elements of X_p are of the form

$$x_{ij}^p = \begin{cases} x_j & : \ (i,j) \in p \\ 0 & : \ \text{otherwise} \end{cases}$$

for all $i \in [m]$ and all $j \in [n]$. The next result provides a theoretical underpinning of update rule (2) for warped softmax regression.

Theorem 3 *Let $c \in \mathbb{N}_{\geq 2}$, let $\Theta = \mathbb{R}^{m \times n}$ let $W = (W_1, \ldots, W_c) \in \Theta^c$, and let*

$$h_k(x) = \exp W_k \otimes x \big/ (\exp W_1 \otimes x + \cdots + \exp W_c \otimes x)$$

for all $k \in [c]$. Then, the function

$$\ell \colon \mathbb{R}^n \times [c] \to \mathbb{R}, \quad (x, y) \mapsto -\sum_{k=1}^{c} \delta_{ky} \log h_k(x)$$

is locally Lipschitz continuous as a function of W_k and

$$\partial_{W_k} \ell(x) = \left\{ -\big(\delta_{ky} - h_k(x)\big) X_{p_k} \ : \ (p_1, \ldots, p_c) \in \mathcal{I}_x^e(W) \right\}$$

for all $k \in [c]$.

Proof Let $(x, y) \in \mathbb{R}^n \times [c]$ and let $\mathcal{P} = \mathcal{P}_{m,n}$. The score of warping x into $W = (w_{ij}) \in \Theta$ along $p \in \mathcal{P}$ can be written as

$$s_p(W, x) = \sum_{(i,j) \in p} w_{ij} x_j = \sum_{(i,j) \in p} w_{ij} x_{ij}^p = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij}^p = \langle W, X_p \rangle,$$

where $\langle W, X_p \rangle$ denotes the Frobenius inner product. Let $k \in [c]$, let $\pi \in \mathcal{P}^c$ be a configuration, and let

$$H_k \colon \mathbb{R}^c \to \mathbb{R}, \quad z \mapsto \exp z_k \big/ (\exp z_1 + \cdots + \exp z_c)$$

be the kth softmax function. We define the continuously differentiable functions

$$\begin{aligned} F_{k\pi} &\colon \Theta \to \mathbb{R}, \quad W_k \mapsto \langle W_k, X_{p_k} \rangle \\ F_\pi &\colon \Theta^c \to \mathbb{R}^c, \quad W \mapsto (F_{1\pi}(W_1), \ldots, F_{c\pi}(W_c)) \\ \ell_{k\pi} &\colon \Theta^c \to \mathbb{R}, \quad W \mapsto -\delta_{ky} \log H_k(F_\pi(W)) \\ \ell_\pi &\colon \Theta^c \to \mathbb{R}, \quad W \mapsto \ell_{1\pi}(W) + \cdots + \ell_{c\pi}(W), \end{aligned}$$

where $W = (W_1, \ldots, W_c) \in \Theta^c$. To derive the gradients $\nabla_{W_k} \ell_\pi(W)$ with respect to W_k, we note that

$$\nabla_{W_k} \log \frac{\exp F_{q\pi}(W_k)}{\sum_{l=1}^{c} \exp F_{l\pi}(W_l)} = \nabla_{W_k} F_{q\pi}(W_q) - \nabla_{W_k} \log \sum_{l=1}^{c} \exp F_{l\pi}(W_l)$$

for all $q \in [c]$. We distinguish between two cases:

Case 1: $k = q$. We have

$$\begin{aligned} \nabla_{W_k} \ell_{q\pi}(W) &= -\delta_{qy} \left(X_{p_k} - \frac{\exp F_{k\pi}(W_k)}{\sum_{l=1}^{c} \exp F_{l\pi}(W_l)} X_{p_k} \right) \\ &= -\delta_{qy} (1 - H_k(F_\pi(W))) X_{p_k}. \end{aligned}$$

Case 2: $k \neq q$. We have

$$\begin{aligned}\nabla_{W_k} \ell_{q\pi}(W) &= -\delta_{qy}\left(0 - \frac{\exp F_{k\pi}(W_q)}{\sum_{l=1}^{c} \exp F_{l\pi}(W_l)} X_{p_k}\right) \\ &= \delta_{qy} H_k(F_\pi(W)) X_{p_k}.\end{aligned}$$

Note that $y = q$ for exactly one $q \in [c]$. Combining both cases yields

$$\nabla_{W_k} \ell_\pi(W) = -\big(\delta_{ky} - H_k(F_\pi(W))\big) X_{p_k}.$$

Next, consider the continuous functions

$$\begin{aligned} F_k &: \Theta \to \mathbb{R}, \quad W_k \mapsto \max\left\{\langle W_k, X_p\rangle \,:\, p \in \mathcal{P}_{m,n}\right\} \\ F &: \Theta^c \to \mathbb{R}^c, \quad W \mapsto (F_1(W_1), \ldots, F_c(W_c)) \\ \ell_k &: \Theta^c \to \mathbb{R}, \quad W \mapsto \delta_{ky} \log H_k(F(W)) \\ \ell &: \Theta^c \to \mathbb{R}, \quad W \mapsto \ell_1(W) + \cdots + \ell_c(W) \end{aligned}$$

for all $k \in [c]$. Observe that $\mathcal{P}^c$ is finite and $\ell(W) \in \{\ell_\pi(x) \,:\, \pi \in \mathcal{P}_W(x)\}$. Since $\ell(W)$ is continuous and all $\ell_p i(W)$ are continuously differentiable, the function $\ell(W)$ is piecewise smooth with essentially active index set $\mathcal{I}_x^e(W)$. Then, the assertion follows from Proposition 2.

To derive update rule (2) for all weights $W = (W_1, \ldots, W_c)$, we assume that $(p_1, \ldots, p_c) \in \mathcal{I}_W^e(x)$. Then, $p_k \in \mathcal{P}_{m,n}$ is an optimal warping path W_k and x and $\xi = -\delta_{ky}(1 - h_k(x))X_p \in \partial_{W_k} \ell(x)$ is a subgradient. The update rule (2) in matrix notation takes the form

$$W_k \leftarrow W_k + \eta\, \delta_{ky}(1 - h_k(x)) X_p.$$

With $W_k = (w_{ij}^k)$ and $X_p = (x_{ij}^p)$, we have

$$w_{ij}^k \leftarrow w_{ij}^k + \eta\, \delta_{ky}(1 - h_k(x)) x_{ij}^p.$$

Finally, update rule (2) follows from $x_{ij}^p = x_j$ for all $(i, j) \in p$ and $x_{ij}^p = 0$ otherwise.

It is important to note that there are configurations $\pi \in \mathcal{I}_x(W) \backslash \mathcal{I}_x^e(W)$. For these configurations, update rule (2) is not necessarily a stochastic subgradient update rule.

5 Experiments

The goal of the experiments is to assess the performance of warped softmax regression. Three experiments were conducted: (i) experiments on label dependency, (ii) comparison of max-linear classifiers, and (iii) comparison of warped softmax regression against nearest neighbor methods.

5.1 Label dependency

The goal of this experiment is to investigate the effects of label dependency on warped softmax regression (WSR) for temporal data.

Table 3 Test classification accuracies in percentage

UCR dataset	NN	WSR_1^-	WSR_1^+	WSR_2
BirdChicken	75.0	‡60.0	**80.0**	*__80.0__
Coffee	**100.0**	‡89.3	96.4	*__100.0__
DistalPhalanxOutlineAgeGroup	79.2	‡68.0	**82.0**	*__82.0__
FordA	56.2	59.3	61.8	*__71.8__
GunPoint	**90.7**	‡82.7	**90.7**	85.3
Ham	46.7	†69.5	70.5	*__72.4__
HandOutlines	79.8	83.9	85.7	*__86.5__
ItalyPowerDemand	95.0	†96.4	**97.3**	96.9
Lighting2	**86.9**	†62.3	62.3	*72.1
MoteStrain	83.5	82.6	**86.4**	84.4
ProximalPhalanxOutlineCorrect	78.4	85.6	**86.9**	85.2
SonyAIBORobotSurface	72.5	‡73.9	83.7	*__85.4__
TwoLeadECG	**91.4**	‡77.2	82.8	*85.0
Wafer	98.0	†97.1	97.7	*__99.4__
Yoga	**83.6**	65.9	70.7	*80.3
Number of wins	5	0	6	8
Average rank	2.7	3.5	1.9	1.6

‡: WSR_1^- is at least five percentage points worse than WSR_1^+
†: WSR_1^- is at most one percentage points worse than WSR_1^+
∗: WSR_2 is equal to or better than WSR_1 (= WSR_1^+)

5.1.1 Data and experimental setup

We used 15 two-category problems from the UCR time series repository [7]. We tested two approaches (see Sect. 3.6):

1. WSR_1: two softmax regression models, each of which has one discriminant function.
2. WSR_2: one softmax regression model with two discriminant functions, one for each class.

We applied Algorithm 3 with elasticity $m = 5$ using *Adaptive Moment Estimation* (ADAM) proposed by [15] to minimize the warped softmax loss and selected the initial learning rate according to Algorithm 6. For WSR_1, we recorded the test accuracy of both classifiers, where WSR_1^- (WSR_1^+) refers to the classifier with the lower (higher) classification accuracy on the training set. As baseline, we applied the nearest neighbor classifier (NN) using the dtw-distance. The experimental protocol was holdout-validation using the train-test splits provided by the UCR repository.

5.1.2 Results

Table 3 summarizes the results. The results confirm that WSR classifiers with a single discriminant function suffer from the label dependency problem. On 6 (4) out of 15 datasets, WSR_1^- was at least 5 (at most 1) percentage points worse than WSR_1^+. In the worst case, the accuracy of WSR_1^- on the BirdChicken dataset was 20 percentage points less than the accuracy of

WSR_1^+. These findings suggest that unfavorable labelings can result in degraded classification accuracies of warped softmax regression with a single discriminant function.

The results show that both WSR_1 and WSR_2 are suitable to solve the label dependency problem. The WSR_2 model exhibited superior performance than WSR_1 ($= \text{WSR}_1^+$) with 9 wins, 2 draws, 4 losses, and an average rank of 1.6 vs. 1.9. In addition, WSR_2 was approximately 10 percentage points better than WSR_1 on three datasets (FordA, Lighting2, Yoga). A possible explanation for the better performance of WSR_2 could be that the decision boundary of WSR_1 is actually based on a single discriminant function, whereas two discriminant functions contribute to the decision boundary of WSR_2.

Both, WSR_1 and WSR_2 performed better than the baseline NN on two-category problems with substantially lower average ranks (1.6 and 1.9) than the average rank 2.7 of NN. In addition, wins, draws, and losses against NN were (9, 1, 5) for WSR_1 and (10, 1, 4) for WSR_2.

In contrast to NN, the results of WSR_1^- and WSR_1^+ provide additional information about the class regions. If WSR_1^- and WSR_1^+ have high accuracies, then the convex hulls of both classes have small overlap. In this case, the classes are (nearly) linearly separable and we could also apply a linear classifier. If WSR_1^- has low and WSR_1^+ has high accuracy, then the classes are not (nearly) linearly separable but max-linearly separable. In this case, one class is contained in a convex region, whereas the other class lies in a non-convex region whose convex hull has a strong overlap with the convex class. Finally, low accuracies of WSR_1^- and WSR_1^+ indicate that neither of both class regions is convex.

5.1.3 Discussion

For practical applications, we recommend to prefer WSR_2 over WSR_1, because it exhibited a better performance, naturally complies with the setting for multi-category problems, and is simpler to handle, because it requires no relabeling of the data.

The proposed WSR models advance prior work on elastic linear classifiers proposed by [11], which are a slightly modified variant of warped-linear models. In that prior work, the label dependency problem went unnoticed, because the relationship to max-linear models via the Representation Theorem was unknown. Consequently, elastic linear classifiers have been applied to two-category problems using a single discriminant function and an arbitrary labeling. The findings of the previous paragraph suggest that elastic linear classifiers missed to exploit their potential for the following reasons: First, according to the results of WSR_1^- and WSR_1^+, the predictive performance of elastic linear models substantially depends on the random choice of labeling. Second, the decision boundary was unknown and properties of the form of class region could not be derived.

Knowledge about the geometric form of the decision boundary and the class regions can guide the choice of an appropriate model. For example, choose a linear model if WSR_1^- and WSR_1^+ have high accuracy; choose a warped-linear model such as WSR_2, if WSR_1^+ has high accuracy and WSR_1^- has substantially lower accuracy. Otherwise, we propose to choose pointwise minimizers of a finite set of warped-linear (min-warped-linear) models or even warped deep learning. In warped deep learning, the first and optionally subsequent layers replace the inner product with a warped product.

From the Representation Theorem follows that min-warped-linear functions are min-max linear functions. Such functions are more expressive than max-linear functions and can implement arbitrary piecewise linear decision boundaries [3]. We leave a theoretical and empirical investigation of min-warped-linear functions as a direction of future research.

5.2 Comparison of Softmax regression classifiers

The Representation Theorem justifies application of warped-linear models on multivariate and max-linear models on temporal data. The goal of this experiment is to assess the performance of standard, max-linear, and warped softmax regression to both, temporal and multivariate data.

5.2.1 Data

We considered 15 time series datasets from the UCR repository [7] and 12 multivariate datasets from the UCI Machine Learning repository [16]. Table 4 displays the selected datasets.

5.2.2 Experimental protocol

In all experiments, we applied holdout validation. We used the train–test splits provided by the UCR repository and randomly split the datasets of the UCI repository into a training and test set with ratio 2 : 1. We compared the following classifiers:

ACR	Name	Type	Algorithm
SR	Softmax regression	Multivariate	Algorithm 1
MLSR	Max-linear softmax regression	Multivariate	Algorithm 2
WSR	Warped softmax regression	Temporal	Algorithm 3

All classifiers applied the same stochastic subgradient method with *Adaptive Moment Estimation* (ADAM) proposed by [15]. We set the maximum number of epochs to 5,000, the maximum number of consecutive epochs without improvement (stagnation) to 250, and the first and second momentum of ADAM to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, respectively. To choose the initial learning rates, we applied Algorithm 6. For MLSR and WSR, we additionally selected the elasticity-parameter as follows: with the above settings, we trained the three classifiers for every elasticity $m \in \{1, 2, 3, 4, 5, 7, 10, 15, 20\}$. Then, we selected the model with the elasticity that minimized the negative log-likelihood.

5.2.3 Results

Table 4 summarizes the results and Table 5 displays the rank distributions. The results show that WSR exhibited the best overall performance, followed by MLSR, whereas the performance of the simple SR classifer was worse, as expected. On temporal data (UCR), WSR substantially outperformed MLSR and SR with better average rank 1.1 vs. 1.9 (MLSR) and 2.5 (SR) and more first ranks 13 vs. 4 (MLSR) and 2 (SR). On multivariate data, WSR performed comparable with MLSR (1.7 vs. 1.8 average rank; 6 vs. 4 first ranks) and superior than SR (1.7 vs. 2.1 average rank; 2 vs. 6 third ranks).

5.2.4 Discussion

As expected, the more complex classifiers WSR and MLSR better account for temporal variation than SR. In addition, we found that the weight-trick implemented by WSR appears to be

Table 4 Classification accuracy (in %) of SR, WSR, and MLSR

UCR dataset	SR	MLSR	WSR
DistPhalOutAG	76.0	79.3	**85.0**
ECG5000	92.3	92.7	**93.4**
ElectricDevices	45.9	42.2	**56.1**
GunPoint	80.7	84.0	**85.3**
ItalyPowerDemand	96.9	**97.2**	97.1
MedicalImages	55.4	63.8	**64.2**
MiddlePhalanxTW	61.4	**61.7**	61.4
Plane	95.2	95.2	**96.2**
ProxPhalOutAG	82.4	84.9	**85.4**
ProxPhalOutCor	84.2	86.9	**88.3**
ProxPhalTW	**78.8**	**78.8**	**78.8**
SonyAIBORobSurf	76.5	70.1	**83.5**
SwedishLeaf	79.2	78.6	**81.1**
Synthetic_control	80.0	86.3	**92.3**
TwoLeadECG	**93.9**	**93.9**	**93.9**
Balance	90.9	91.8	**92.3**
Banknote	98.7	**100.0**	**100.0**
Ecoli	**81.6**	73.7	75.4
Eye	60.4	**88.7**	82.4
Glass	62.0	**64.8**	63.4
Ionosphere	87.2	88.0	**89.7**
Iris	**94.1**	**94.1**	92.2
Occupancy	**99.0**	98.9	**99.0**
Pima	**77.7**	71.5	67.2
Sonar	71.0	84.1	**88.4**
Whitewine	53.5	53.3	**53.8**
Yeast	**58.6**	54.6	57.4

particularly useful for temporal data, whereas its effect on multivariate data is substantially less pronounced. These findings suggest that replacing the inner product of linear models by warped product has a similar beneficial effect as replacing the Euclidean distance in nearest-neighbor classification with the dtw-distance.

The superior results of SR on the multivariate datasets ecoli and pima indicate that WSR and MLSR are both prone to overfitting (which is also confirmed by comparing training and test results). One standard approach to cope with overfitting is regularization. We defer the issue of regularization of WSR models to future work.

5.2.5 Comparison of classifiers in DTW spaces

The two simple mainstays in statistical pattern recognition are linear models and nearest neighbors (NN) methods. In this experiment, we compare the performance of their dtw-enhanced counterparts on temporal data. Specifically, we compare the performance of the proposed warped softmax regression model against nearest-neighbor methods under dtw.

Table 5 Rank distribution, average rank, and standard deviation

UCR	Rank			Avg	Std	UCI	Rank			avg	Std
	1	2	3				1	2	3		
SR	2	4	9	2.5	0.74	SR	5	1	6	2.1	1.00
MLSR	4	8	3	1.9	0.70	MLSR	4	6	2	1.8	0.72
WSR	13	2	0	1.1	0.35	WSR	6	4	2	1.7	0.78

Ranks go from 1 (best) to 3 (worst)

5.2.6 Data

We selected 29 datasets from the UCR time series repository [7]. The datasets were chosen to cover various characteristics such as application domain, length of time series, number of classes, and sample size. Table 6 shows the datasets. Restriction to a subset of the full UCR repository was due to limited computational resources for conducting the experimental protocol as described below.

5.2.7 Algorithms

We compared the following classifiers:

Notation	Algorithm	References
NN	Nearest neighbor classifier with dtw	[10]
GLVQ	Generalized LVQ with dtw	[13]
SR	Softmax regression	Algorithm 1
WSR	Warped softmax regression	Algorithm 3

We considered NN, because it is the prime approach in time series classification [5]. Two disadvantages of NN are its high storage and computation requirements. For this reason, we considered the protoype-based classifier GLVQ. According to [13], GLVQ outperformed state-of-the-art data reduction methods for time series classification such as the unsupervised prototype generation method based on k-means [21]. As a baseline and to assess its sensitivity to temporal variations, we also applied SR.

5.2.8 Experimental protocol

For every dataset, we conducted tenfold cross-validation and reported the mean classification accuracy. We used the following setting of SR and WSR: Both classifiers applied the same stochastic subgradient method using ADAM [15]. We set the maximum number of epochs to 5,000, the maximum number of consecutive epochs without improvement to 100, and the first and second momentum of ADAM to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, respectively. To choose the initial learning rates, we applied Algorithm 6. For WSR we additionally selected the elasticity-parameter as follows: with the above settings, we trained WSR for every elasticity $m \in \{1, 2, 3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$. Then, we selected the elasticity that minimized the negative log-likelihood.

Table 6 Mean accuracy in percentage averaged over tenfold cross-validation

UCR dataset	NN	GLVQ	SR	WSR
Beef	53.3	63.3	**84.0**	81.0
CBF	**99.9**	99.6	97.9	98.9
ChlorineConcentration	99.6	71.6	86.9	**99.8**
Coffee	**100.0**	98.2	96.7	**100.0**
ECG200	83.5	80.5	86.0	**88.4**
ECG5000	93.3	**94.3**	94.1	93.3
ECGFiveDays	99.2	99.1	99.6	**99.7**
ElectricDevices	**79.2**	78.2	53.9	74.7
FaceFour	**92.9**	92.0	91.7	92.3
FacesUCR	**97.8**	97.2	86.8	94.0
FISH	80.3	**90.9**	86.3	90.0
GunPoint	91.5	**97.0**	87.0	96.0
Ham	72.4	74.8	81.8	**84.9**
ItalyPowerDemand	95.8	95.3	**97.0**	95.9
Lighting2	**89.3**	76.0	61.6	65.8
Lighting7	71.3	**82.5**	56.9	63.1
MedicalImages	**80.7**	71.7	63.9	73.2
OliveOil	**85.0**	**85.0**	52.7	81.0
ProximalPhalanxOutlineAgeGroup	75.5	**83.6**	82.2	**83.6**
ProximalPhalanxOutlineCorrect	82.0	85.1	78.9	**86.3**
ProximalPhalanxTW	77.5	81.2	81.0	**82.2**
RefrigerationDevices	60.7	**62.9**	37.5	47.9
Strawberry	96.5	94.1	96.1	**97.9**
SwedishLeaf	82.0	87.6	80.7	**87.7**
Synthetic control	99.2	**99.5**	83.0	94.2
ToeSegmentation1	85.8	**92.5**	52.6	73.8
Trace	**100.0**	**100.0**	70.0	89.0
Wafer	99.4	96.5	94.0	**99.8**
yoga	**93.9**	73.2	69.5	93.2

5.2.9 Results

Table 6 shows the mean classification accuracies of the tenfold cross-validation protocol, Table 7 summarizes the rank distributions, and Fig. 10 displays the pairwise comparison of the four classifiers.

The three classifiers WSR, NN, and GLVQ exhibited comparable classification performance with slight advantages for WSR on rankings (Table 7) and for NN on pairwise comparisons (Fig. 10). The rank distributions and pairwise comparisons suggest that the performance of WSR, NN, and GLVQ are complementary. The linear classifier SR performed worst by a large margin with respect to ranking, winning percentages, and mean percentage difference.

Table 7 Rank distribution, average rank, and standard deviation

Classifier		Rank				Avg	Std
		1	2	3	4		
NN	Nearest-neighbor	10	7	7	5	2.2	1.1
GLVQ	Generalized LVQ	10	7	7	5	2.2	1.1
SR	Softmax regression	2	4	5	18	3.3	1.0
WSR	Warped softmax regression	11	7	11	0	2.0	0.9

Ranks go from 1 (best) to 4 (worst)

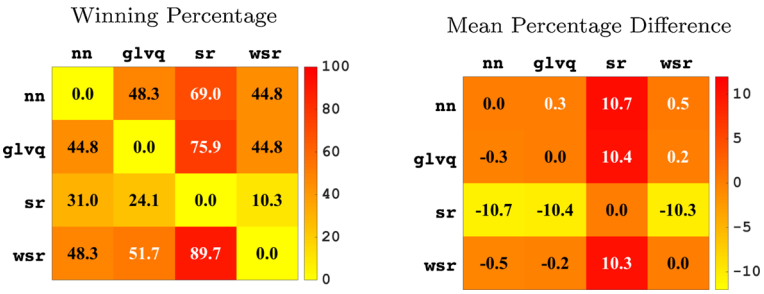

Fig. 10 Pairwise comparison of NN, GLVQ, SR, and WSR. Left: Pairwise winning percentages w_{ij}, where classifier in row i wins w_{ij} percentages of all competitions against the classifier in column j. Right: Pairwise mean percentage difference a_{ij} in accuracy, where the accuracy of classifier in row i is a_{ij} percentages better on average than the accuracy of the classifier in column j. We refer to Appendix 1 for a definition of both measures

Table 8 Running times for classifying a single test example in big $\mathcal{O}$ notation under the assumption that all time series are of length n. The first row shows the growth rates. The second row shows the asymptotic speed-up factor of a classifier compared to NN. The last row shows the asymptotic mean speed-up factor over all UCR datasets from Table 6

classifier	NN	GLVQ	SR	WSR
Growth rate	$\mathcal{O}\left(N \cdot n^2\right)$	$\mathcal{O}\left(c \cdot n^2\right)$	$\mathcal{O}(c \cdot n)$	$\mathcal{O}(c \cdot n \cdot m)$
Speed-up factor	1	N/c	$n \cdot N/c$	$(n \cdot N)/(m \cdot c)$
Mean speed-up factor	1	123.3	21112.7	844.5

N = # training examples • c = # classes • n = length of time series • m = elasticity

5.2.10 Discussion

The results confirm earlier findings that standard linear models is not competitive in time series classification [5]. The predictive performance of WSR suggests that warped products are a useful modification of the inner product to cope with temporal variations. The advantage of WSR over NN and GLVQ is its computational efficiency. Table 8 summarizes the asymptotic times required for classifying a single test example. The limiting behaviors of the run times indicate that WSR is almost three orders of magnitudes faster than NN and about two orders of magnitude faster than GLVQ. These findings suggest that WSR models could serve as a fast and simple first-trial alternative to NN classifiers and as an elementary building block for more sophisticated classifiers such as min-warped-linear models and warped deep learning.

6 Conclusion

Warped-linear models are time-warp invariant analogues of linear models. The key result to gain insight into warped-linear models is the Representation Theorem. It states that every warped-linear model is max-linear. From the Representation Theorem together with the Matrix Complexity Lemma follows that warped-linear models almost always implement an efficient weight trick by weight selection and massive weight sharing. From the Representation Theorem directly follows that warped-linear models on time series of identical length correspond to polytope classifiers. As such, they suffer from the label dependency problem, which can be easily solved by using the same number of discriminant functions as class labels. Since the warped softmax loss is locally Lipschitz continuous, we can apply (stochastic) subgradient methods to minimize the loss. Empirical results suggest that replacing the inner product of linear models by warped product has a similar beneficial effect as replacing the Euclidean distance in nearest neighbor classifiers by the dtw-distance. In addition, we suggest warped-linear models as an efficient first-trial alternative to the dtw-nearest neighbor prediction rule.

For future work, we suggest the following directions: First, empirically explore warped-linear models. For example, (i) apply different loss functions, (ii) devise appropriate regularization schemes, (iii) set different elasticities for different classes, and (iv) apply different optimization schemes. Second, theoretically analyze properties of warped-linear models such as its VC dimension. Third, warped-linear models open up new opportunities to more sophisticated classifiers such as (i) min-warped-linear models and (ii) warped deep learning.

Acknowledgements The author was funded by the DFG Sachbeihilfe `JA 2109/4-2`. This work was undertaken at the TU Berlin and finalized at the OTH Regensburg.

A Performance measures

This section describes the pairwise winning percentage and pairwise mean percentage difference.

A1.1 Winning percentage

The pairwise winning percentages are summarized in a matrix $W = (w_{ij})$. The winning percentage w_{ij} is the fraction of datasets for which the accuracy of the classifier in row i is strictly higher than the accuracy of the classifier in column j. Formally, the winning percentage w_{ij} is defined by

$$w_{ij} = 100 \cdot \frac{|\{d \in \mathcal{D} : \mathrm{acc}_d(j) < \mathrm{acc}_d(i)\}|}{|\mathcal{D}|}$$

where $\mathrm{acc}_d(i)$ is the accuracy of the classifier in row i on dataset d and $\mathrm{acc}_d(j)$ is the accuracy of the classifier in column j on d. The percentage w_{ij}^{eq} of ties between classifiers i and j can be inferred by

$$w_{ij}^{eq} = 100 - w_{ij} - w_{ji}.$$

A1.2 Pairwise mean percentage difference

The pairwise mean percentage differences are summarized in a matrix $A = (a_{ij})$. The mean percentage difference a_{ij} between the classifier in row i and the classifier in column j is defined by

$$a_{ij} = 100 \cdot \frac{2}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \cdot \frac{\mathrm{acc}_d(i) - \mathrm{acc}_d(j)}{\mathrm{acc}_d(i) + \mathrm{acc}_d(j)},$$

Positive (negative) values a_{ij} mean that the average accuracy of the row classifier was higher (lower) on average than the average accuracy of the column classifier.

B Algorithms

This section outlines the implemented algorithms in pseudo-code:

1. Algorithm 1: softmax regression
2. Algorithm 2: max-linear softmax regression
3. Algorithm 3: warped softmax regression
4. Algorithm 4: computation of warped products
5. Algorithm 5: computation of optimal warping paths
6. Algorithm 6: selection of initial learning rate

Note that Algorithms 4 is a variation in the dynamic program for the dynamic time warping distance and Algorithm 5 is the corresponding back-track procedure to derive an optimal warping path. The proof of correctness of both algorithms follows in the same way as for the corresponding algorithms for the dynamic time warping distance.

Algorithm 1 Softmax Regression

```
1: procedure SR(D, η)
2:     for k ← 1 to c do
3:         initialize w_k = (w_1^k, ..., w_n^k)          // w_1^k = b_k is the bias
4:     repeat
5:         randomly select (x, y) ∈ D
6:         for k ← 1 to c do
7:             h_k = exp w_k^T x / Σ_{l∈Y} exp w_l^T x
8:         for k ← 1 to c do
9:             for i ← 1 to n do
10:                w_i^k ← w_i^k + η · (δ_ky − h_k) x_i
11:        adjust learning rate η
12:    until termination
13:    return θ = (w_1, ..., w_c)
```

Softmax regression using stochastic gradient descent to minimize the cost function. Line 10 describes the stochastic gradient rule (1) in element-wise rather than in vector notation.

Algorithm 2 Max-Linear Softmax Regression

1: **procedure** MLSR($\mathcal{D}, \eta, \lambda, m$)
2: **for** $k \leftarrow 1$ **to** c **do**
3: initialize $W_k = (w_{ij}^k) \in \mathbb{R}^{m \times n}$ // w_{i1}^k = bias of ith component of $f_k(x)$
4: **repeat**
5: randomly select $(x, y) \in \mathcal{D}$
6: **for** $k \leftarrow 1$ **to** c **do**
7: $h_k \leftarrow \exp(-f_k(x)) / \sum_{l=1}^{c} \exp(-f_l(x)))$
8: $k \leftarrow \operatorname{argmax}_l h_l$
9: **for** $k \leftarrow 1$ **to** c **do**
10: $i \leftarrow \operatorname{argmax}_l g_{kl}(x)$ // $g_{kl}(x)$ is lth component of $f_k(x)$
11: **for** $j \leftarrow 1$ **to** n **do**
12: $w_{ij}^k \leftarrow w_{ij}^k + \eta \cdot (\delta_{ky} - h_k) x_j$ // $g_k = \nabla_{W_k} R(W_k)$
13: **until** termination
14: **return** $\theta = (W_1, \ldots, W_c)$

Max-linear softmax regression using the stochastic subgradient method to minimize the cost function. Line 10 selects for every class k the component $g_{kl}(x)$ that maximizes the max-linear function $f_k(x)$, that is $f_k(x) = g_{kl}(x)$. Then, lines 11 and 12 update the weights of component g_{kl}.

Algorithm 3 Warped Softmax Regression

1: **procedure** WSR($\mathcal{D}, \eta, \lambda, m$)
2: $n \leftarrow \max\{|x| : (x, y) \in \mathcal{D}\}$
3: **for** $k \leftarrow 1$ **to** c **do**
4: initialize $W_k = (w_{ij}^k) \in \mathbb{R}^{m \times n}$ // first column w_{i1}^k forms elastic bias
5: **repeat**
6: randomly select $(x, y) \in \mathcal{D}$
7: **for** $k \leftarrow 1$ **to** c **do**
8: $h_k \leftarrow \exp(W_k \otimes x) / \sum_{l=1}^{c} \exp(W_l \otimes x)$
9: $p_k \leftarrow$ optimal warping path of $W_k \otimes x$
10: **for** $k \leftarrow 1$ **to** c **do**
11: **for all** $(i, j) \in p_k$ **do** // for every point (i, j) of path p_k
12: $w_{ij}^k \leftarrow w_{ij}^k + \eta \cdot (\delta_{ky} - h_k) x_j$
13: **until** termination
14: **return** $\theta = (W_1, \ldots, W_c)$

Warped softmax regression using the stochastic subgradient method to minimize the cost function. Line 2 determines the maximum length n of all augmented input examples from $\mathcal{D}$. The number n together with the elasticity m specify the dimension of the augmented weight matrices $W_k = (w_{ij}^k)$. Line 8 invokes Algorithm 4 to evaluate the softmax functions h_k and line 9 calls Algorithm 5 to determine optimal warping paths between W_k and x. Line 10–12 present the stochastic subgradient rule.

Algorithm 4 Warped product

function $W \otimes x$
  $m \times n =$ dimension of $W = (w_{ij})$
  $n' =$ length of x // assert $n' \leq n$
  $\sigma_{11} \leftarrow w_{11} x_1$
  for $i \leftarrow 2$ **to** m **do**
    $\sigma_{i1} \leftarrow \sigma_{i-1,1} + w_{i1} x_1$
  for $j \leftarrow 2$ **to** n' **do**
    $\sigma_{1j} \leftarrow \sigma_{1,j-1} + w_{1j} x_j$
  for $i \leftarrow 2$ **to** m **do**
    for $j \leftarrow 2$ **to** n' **do**
      $\sigma_{ij} \leftarrow w_{ij} x_j + \max\left(\sigma_{i-1,j},\ \sigma_{i,j-1},\ \sigma_{i-1,j-1}\right)$
  return $\sigma_{mn'}$

Dynamic program to compute warped products. The accumulated local scores σ_{ij} correspond to the warped products between the sub-matrices $W_{[1:i,1:j]}$ and sub-series $x_{[1:j]}$ finally giving $\sigma_{mn'} = W \otimes x$.

Algorithm 5 Optimal warping path

1: **function** OPTIMALWARPINGPATH($\Sigma = (\sigma_{ij})$)
2: $p \leftarrow \langle (m, n') \rangle$
3: **while** $i > 1$ and $j > 1$ **do**
4: **if** $i = 1$ **then**
5: $j \leftarrow j - 1$
6: **else if** $j = 1$ **then**
7: $i \leftarrow i - 1$
8: **else**
9: $\sigma^* = \max\left(\sigma_{i-1,j},\ \sigma_{i,j-1},\ \sigma_{i-1,j-1}\right)$
10: **if** $\sigma_{i-1,j-1} = \sigma^*$ **then**
11: $i \leftarrow i - 1$
12: $j \leftarrow j - 1$
13: **else if** $\sigma_{i-1,j} = \sigma^*$ **then**
14: $i \leftarrow i - 1$
15: **else**
16: $j \leftarrow j - 1$
17: p.addFirst(i, j)
18: **return** p

Algorithm to compute an optimal warping path of the warped product $W \otimes x$, where $m \times n$ is the dimension of W and $n' \leq n$ is the length of x. The algorithm requires the output $\Sigma = (\sigma_{ij})$ of Algorithm 4 as input and returns an optimal warping path p. The optimal warping path p is computed in reverse order of the indices. Line 2 initializes p as a list $\langle (m, n') \rangle$ containing the last point $p_L = (m, n')$ of p as first element. In the while-loop, the algorithm backtracks the matrix one unit in opposite direction of the step condition (up, left, diagonal) until it reaches the first point $p_1 = (1, 1)$ of p. Lines 4–16 determine the next point to be added to p. If $p_l = (i, j)$ is the most recent point added to p, then backtracking chooses

$$p_{l-1} = \begin{cases} (1, j-1) & : \ p_l = (1, j) \\ (i-1, 1) & : \ p_l = (i, 1) \\ \operatorname{argmax}\left(\sigma_{i-1,j},\ \sigma_{i,j-1},\ \sigma_{i-1,j-1}\right) & : \ \text{otherwise} \end{cases}$$

as next point to be added to p. In case of ties, the order of the next step is diagonal, left, up. Thus, shorter optimal warping paths are preferred. Finally, line 17 prepends p_{l-1} to the current list p.

Algorithm 6 Selection of initial learning rate

```
1: function SELECTLEARNINGRATE
2:     η ← 0.8                          // learning rate
3:     ρ ← 1.0                          // ratio of stagnation to number of
   epochs
4:     initialize classifier
5:     while ρ ≥ 0.2 do
6:         η ← η/2
7:         s ← 0
8:         compute cost J_0
9:         for t ← 1 to 100 do          // t = number of epochs
10:            apply stochastic learning rule
11:            compute cost J_t
12:            if J_{t−1} ≤ J_t then
13:                s ← s + 1            // increase stagnation
14:            ρ ← s/t                  // update ratio
15:    return η
```

Selection of initial learning rate. The algorithm starts with learning rate $\eta = 0.4$ (see line 2 and 6) and then halves the learning rate in each iteration of the while-loop until the fraction of stagnation during the first 100 epochs (cycles through the training set) is less than 20%.

References

1. Abanda A, Mori U, Lozano JA (2018) A review on distance based time series classification. Data Min Knowl Discov 2:94
2. Astorino A, Gaudioso M (2002) Polyhedral separability through successive LP. J Optim Theory Appl 112(2):265–293
3. Bagirov A, Karmitsa N, Mäkelä MM (2014) Introduction to nonsmooth optimization. Springer International Publishing, Berlin
4. Banderier C, Schwer S (2005) Why Delannoy numbers? J Stat Plan Inference 135(1):40–54
5. Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Disc 31(3):606–660
6. Berndt DJ, Clifford J (1996) Finding patterns in time series: a dynamic programming approach. Adv Knowl Discov Data Min 5:229–248
7. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The UCR time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/
8. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, London
9. Dundar MM, Wolf M, Lakare S, Salganicoff M, Raykar VC (2008) Polyhedral classifier for target detection: a case study: colorectal cancer. In: International conference on machine learning (ICML)
10. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer, New York
11. Jain B (2015) Generalized gradient learning on time series. Mach Learn 100(2–3):587–608
12. Jain B (2017) Warped-linear models for time series classification arXiv:1711.09156
13. Jain B, Schultz D (2018) Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces. Pattern Recogn 76:349–366
14. Kantchelian A, Tschantz MC, Huang L, Bartlett PL, Joseph AD, Tygar JD (2014) Large-margin convex polytope machine. In: Neural information processing systems (NIPS)
15. Kingma DP, Ba JL (2015) Adam: a method for Stochastic optimization
16. Lichman M (2013) UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml
17. Manwani N, Sastry PS (2010) Learning polyhedral classifiers using logistic function. Asian Conf Mach Learn 13:17–30
18. Megiddo N (1988) On the complexity of polyhedral separability. Discrete Comput Geom 3:325–337
19. Norkin V (1986) Stochastic generalized-differentiable functions in the problem of nonconvex nonsmooth stochastic optimization. Cybern Syst Anal 22(6):804–809
20. Orsenigo C, Vercellis C (2007) Accurately learning from few examples with a polyhedral classifier. Comput Optim Appl 38(2):235–247

21. Petitjean F, Forestier G, Webb GI, Nicholson AE, Chen Y, Keogh E (2016) Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. Knowl Inf Syst 47(1):1–26
22. Scholtes S (2012) An introduction to piecewise differentiable equations. Springer, Berlin
23. Serra J, Arcos JL (2014) An empirical evaluation of similarity measures for time series classification. Knowl-Based Syst 67:305–314
24. Shor NZ (1985) Minimization methods for nondifferentiable functions. Springer, Berlin
25. Takacs G (2009) Convex polyhedron learning and its applications. Ph.D. thesis, Budapest University of Technology and Economics, Budapest, Hungary
26. Vintsyuk TK (1968) Speech discrimination by dynamic programming. Cybern Syst Anal 4(1):52–57
27. Yujian L, Bo L, Xinwu Y, Yaozong F, Houjun L (2011) Multiconlitron: a general piecewise linear classifier. IEEE Trans Neural Netw 22(2):276–289

Brijnesh Jain studied mathematics and computer science at the University Göttingen and TU Berlin, Germany. He received his Ph.D. in machine learning at the TU Berlin. He is professor at the OTH Regensburg, Germany. His research interest is machine learning in non-Euclidean spaces.