Invariant Time-Series Classification

Josif Grabocka[1], Alexandros Nanopoulos[2], and Lars Schmidt-Thieme[1]

[1] Information Systems and Machine Learning Lab
Samelsonplatz 22, 31141 Hildesheim, Germany
[2] University of Eichstaett-Ingolstadt, Germany
{josif,schmidt-thieme}@ismll.de, alexandros.nanopoulos@ku.de

Abstract. Time-series classification is a field of machine learning that has attracted considerable focus during the recent decades. The large number of time-series application areas ranges from medical diagnosis up to financial econometrics. Support Vector Machines (SVMs) are reported to perform non-optimally in the domain of time series, because they suffer detecting similarities in the lack of abundant training instances. In this study we present a novel time-series transformation method which significantly improves the performance of SVMs. Our novel transformation method is used to enlarge the training set through creating new transformed instances from the support vector instances. The new transformed instances encapsulate the necessary intra-class variations required to redefine the maximum margin decision boundary. The proposed transformation method utilizes the variance distributions from the intra-class warping maps to build transformation fields, which are applied to series instances using the Moving Least Squares algorithm. Extensive experimentations on 35 time series datasets demonstrate the superiority of the proposed method compared to both the Dynamic Time Warping version of the Nearest Neighbor and the SVMs classifiers, outperforming them in the majority of the experiments.

Keywords: Machine Learning, Time Series Classification, Data-Driven Transformations, Invariant Classification.

1 Introduction

Time series classification is one of the most appealing research domains in machine learning. The generality of interest is influenced by the large number of problems involving time series, ranging from financial econometrics up to medical diagnosis [1]. The most widely applied time series classifier is the nearest neighbor (NN) empowered with a similarity/distance metric called Dynamic Time Warping (DTW), hereafter jointly denoted as DTW-NN [2]. Due to the accuracy of DTW in detecting pattern variations, DTW-NN has been characterized as a hard-to-beat baseline [3].

Support Vector Machines (SVMs) are successful classifiers involved in solving a variety of learning and function estimation problems. Yet, experimental studies show that it performs non-optimally in the domain of time series [4]. We explore

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 725–740, 2012.

an approach to boost the classification of time series using SVMs, which is directly inspired by the nature of the problem and the reasons why SVMs fail to build optimal decision boundaries. In most time series datasets, the variations of instances belonging to the same class, denoted intra-class variation, are considerably numerous. Variations appear in different flavors. A pattern of a signal/time series can start at various time points (translational variance) and the duration of a pattern can vary in length (scaling variance). Even more challenging, such variances can partially occur in a signal, in multiple locations, by unexpected direction and magnitude of change. There exist more, theoretically infinitely many, possible variations of a particular class pattern, compared to the present number of instances in the dataset. Ergo, such lack of sufficient instances to cover all the possible variations can affect the maximum margin decision boundary in under-representing the ideal decision boundary of the problem.

In order to overcome the lack of instances, the insertion of virtual transformed instances to the training set has been proposed. In the case of SVMs, support vectors are transformed/deformed, the new virtual support vectors added back to the training set and the model is finally retrained [5,6]. An illustration of the effect of inserting virtual instances and its impact on redefining the decision boundary is shown in Figure 1. The most challenging aspect of this strategy is to define efficient transformation functions, which create new instances from existing ones, enabling the generated instances to represent the necessary variations in the feature space.

The main contribution of our study is in defining a novel instance transformation method which improves the performance of SVMs in time series classification. In our analysis the transformations should possess four characteristics. First the transformations should be data-driven, concretely an analysis of the intra-class variance distributions should be taken into account. Secondly, localized variations are required since the variations of series instances appears in forms of local deformations. Thirdly, in order to overcome the time complexity issues, only a representative subset of the instances should be selected for producing variations. Finally the transformations should accurately redefine the decision boundary without creating outliers or over-fit the training set.

The novel transformation method we introduce satisfies all the above raised requirements. The proposed method analyses the translational variance distributions by constructing warping alignment maps of intra-class instances. The time series is divided into a number of local regions and transformation fields/vectors are created to represent the direction and magnitude of the translational variance at every region center, based on the constructed variance distributions. Finally, the application of the transformation fields to time series is conducted using the Moving Least Squares algorithm [7].

The efficiency of the proposed method is verified through extensive experimentation on 35 datasets from the UCR collection[1]. Our method clearly outperforms DTW-NN on the vast majority of challenging datasets, while being on a par competitive with DTW-NN in the easy (low error) ones. Furthermore, the

[1] www.cs.ucr.edu/~eamonn/time_series_data

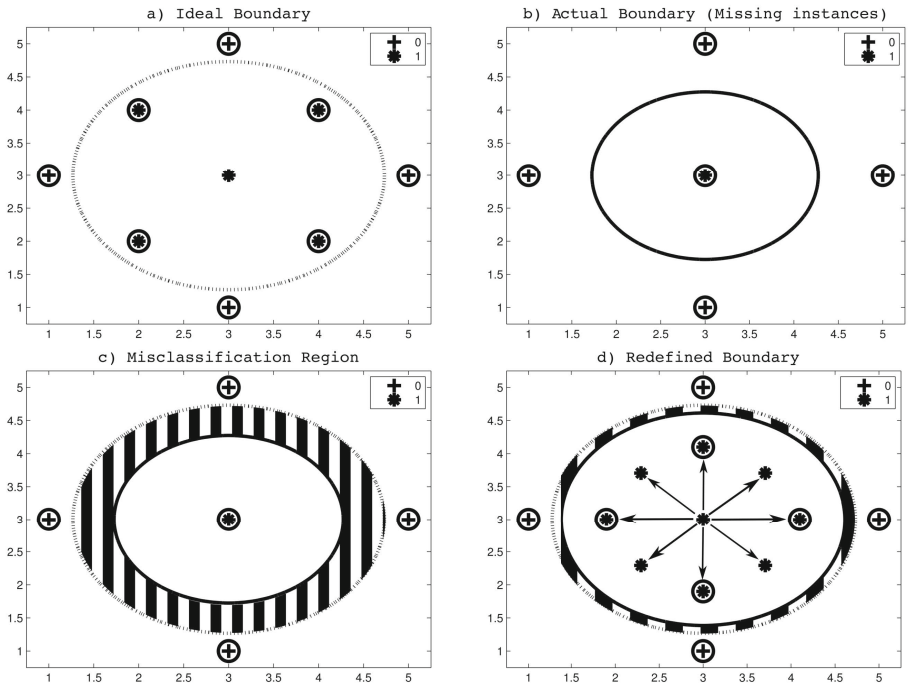

Fig. 1. a) An ideal max-margin decision boundary for the depicted binary (0/1) classification problem. *Circled* points denote support vectors. **b)** An actual version of the problem where most class 1 instances are missing. The actual max-margin decision boundary differs from the ideal boundary in a). **c)** The under-fitting of the actual boundary (*solid*) to represent the ideal boundary (*dots*) produces the *shaded* misclassification region. **d)** Transforming the class 1 instance in coordinate (3,3) and inserting the transformed instances (*pointed by arrows*) back to the dataset, helps redefine the new max-margin boundary (*solid*). Consecutively, the area of the misclassification region is reduced.

results indicate that our proposed method always improves the default SVM. The principal contributions of the study can be summarized as:

- A novel time series transformation method is presented
- For the first time, the approach of Invariant SVMs is proposed in time-series domain
- Extensive experimentations are conducted to demonstrate the superiority of the proposed method

2 Related Work

2.1 Time Series Classification

Time series classification has been elaborated for more than two decades and a plethora of methods has been proposed for the task. Various classifiers have

been applied, ranging from neural networks [8,9] to bayesian networks [10], from decision trees [11] to first-order logic rules [12], and from Hidden Markov Models [13] to tailored dimensionality reduction [14].

Dynamic Time Warping. Despite the major effort spent in building accurate time series classifiers, still the nearest neighbor classifier combined with a similarity technique called Dynamic Time Warping (DTW) was reported to produce more accurate results [15]. DTW overcomes the drawback of other methods because it can detect pattern variations, such as translations/shifts, size and deformations. The metric builds an optimal alignment of points among two time series by dynamically constructing a progressive cost matrix. It computes the the path of the minimal overall point pairs' distance [2]. Adding warping window size constraints have been reported to occasionally boost the classification [16]. In contrast to DTW-NN, our study aims at building a competitive max-margin classifier.

2.2 Invariant SVMs Classification

Even though the challenge of handling invariance in time series classification is rather new, it has, however, been applied long ago to the domain of image classification. Significant effort to the problem of invariant classification was followed by the SVM community, where one of the initial and most successful approaches relies on creating virtual instances by replicating instances. Typically the support vectors are transformed creating new instances called Virtual Support Vectors (VSV), with the aim of redefining the decision boundary [5,17]. VSV has been reported to achieve optimal performance in image classification [6]. An alternative technique in handling variations relies in modifying the kernel of the SVM, by adding a loss term in the dual objective function. Such a loss enforces the decision boundary to be tangent to the transformation vector [6,18]. Other approaches have been focused on selecting an adequate set of instances for transformation [19]. Studies aiming the adoption of SVM to the context of time series have been primarily addressing the inclusion of DTW as a kernel [20,21]. Unfortunately, to date, all proposed DTW-based kernels we are aware of, are not efficiently obeying the positively semi-definite requirement [4]. The DTW based kernels have been reported to not perform optimally compared to state-of-art [22]. Generating new instances based on the pairwise similarities has also been applied [4], with limited success compared to DTW-NN. In comparison, our method applies a novel transformation technique along with the VSV approach.

2.3 Instance Transformations

Intentional transformations and deformations of time series has shown little interest because of the limited studies of VSV classifiers in the domain. Among the initiatives, morphing transformation from a time series to another has been inspected [23]. However, deformations have been more principally investigated

in the domain of images. Moving Least Squares is a state-of-art technique to produce realistic deformations [7]. In this work we use the Moving Least Squares method for applying the transformations over series.

3 Proposed Method

3.1 Principle

In order to boost the classification accuracy our method needs to generate new instances via transformations. In order to capture the necessary patterns' intra-class variations, the transformation technique should aim for certain characteristics and requirements. In our analysis, the transformations should obey to the following list of properties:

- **Data-Driven:** Variance should be generated by analyzing the similarity distribution of instances inside a class.
- **Localized:** Intra-class variations are often expressed in local deformations, instead of global variations.
- **Selective:** Transforming all the instances becomes computationally expensive and many instances can be redundant w.r.t to the decision boundary. Therefore it is crucial to select only a few class-representative instances for generating variations.
- **Accurate:** The transformed instances should help redefine the decision boundary, however care should be payed to avoid excessive magnitudes of transformation, in order to avoid generating outliers.

3.2 Method Outline

The transformation method and the instance generation technique we are introducing, does answer all the requirements we raised in section 3.1. Initially we define the local transformation fields in subsection 3.3, which are used to transform time series using the Moving Least Squares algorithm. The transformation fields are constructed by measuring the translational variance distributions subsection 3.5. The variance distributions are obtained by building the intra-class warping maps, defined in subsection 3.4. Finally, the transformation fields are applied only to the support vectors following the Virtual Support Vector classification approach, defined in subsection 3.6.

3.3 Transformation Fields and Moving Least Squares

In this subsection we present only the technicalities of how a time-series can be transformed by a particular localized transformation, while the actual method that creates intelligent transformation magnitudes will be introduced in forthcoming subsections. Variations of time series are often occurring in localized forms, meaning that two series differ only in the deformation of a particular subsequence rather than a global difference. In order to include the mechanism

of localized variances we first introduce the concept of a **transformation field**, denoted F. We split the time series into K many regions, and then we define the left/right translation that is required for each region. Each region will be transformed dedicatedly, while the transformation will be applied to the centroid of the region. Such centroids are denoted as **control points**. The amount of translational transformation applied to every control point (hence every region) is denoted as the transformation field vector $F \in \mathbb{R}^K$. For instance Figure 2 shows the effect of applying a transformation field on two regions of a time series, where each region is denoted by its representative control point.

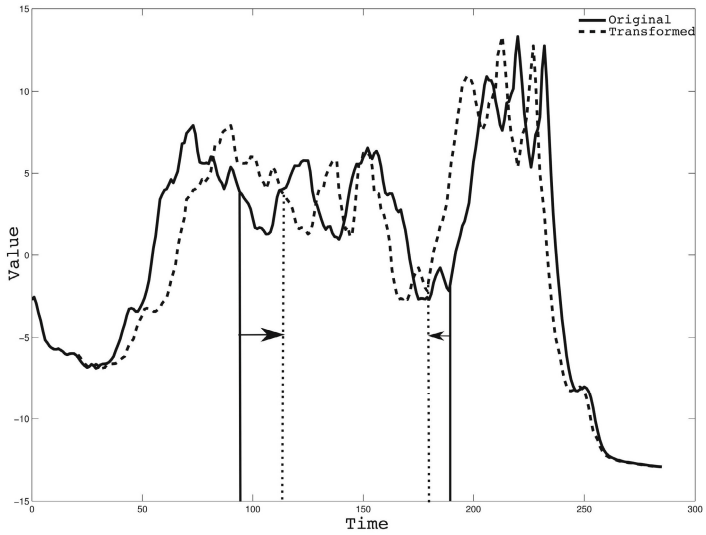

Fig. 2. Demonstrating the effect of applying a transformation field vector of values [+20 -10] to the control points positioned at [95 190] highlighted with vertical lines, on an instance series from the *Coffee* dataset. The transformation algorithm (MLS) is described in Algorithm 1.

The mechanism of applying a transformation field to a series is conducted via the deformation algorithm called Moving Least Squares (MLS), which is described in Algorithm 1. This algorithm is used to transform one signal that passes through a set of points P, called control points. The transformation is defined by a new set of control points Q, which are the transformed positions of the control points P [7]. The control points Q are obtained, in our implementation, by applying transformation fields F translations to the original control points P. MLS applies the transformation by initially creating one local approximation function l_v for each point v of the time series. Thus, for every point we solve the best affine transformations that approximates the new control points Q [line 3 of Alg 1]. There is a weight decay in the importance of the control points compared to the point for which we are defining a local transformation, approximation of

Algorithm 1. MLS [7]

Require: A series: S, A transformation field: F, Control Points: P
Ensure: New transformed instance: T
1: $Q \leftarrow [P_1 + F_1, P_2 + F_2, ...]$
2: **for** $v = 1$ to $|S|$ **do**
3: Search l_v that minimizes
$\text{argmin}_{l_v} \sum_{i=1}^{|P|} w_i |l_v(P_i) - Q_i|^2$, where $w_i = \frac{1}{|P_i - v|^{2\alpha}}$
4: $T[v] \leftarrow l_v(S[v])$
5: **end for**
6: **return** T

near control points get more impact. The speed of decay is controlled by a hyper parameter α.

Once the local transformation function l_v is computed, then the value at point v in the transformed series is computed by applying the searched transformation function over the value in the original series. In order for the transformed series to look as realistic compared to the original series, the transformation should be as rigid as possible, that is, the space of deformations should not even include uniform scaling, therefore we follow the rigid transformations optimization [7] in solving line 3 of Alg. 1. This subsection only introduced the transformation fields and the underlying algorithm used to transform a series, while the successive subsections will show how to search for the best magnitudes of the transformation fields vector elements, in order for the transformed instances to encapsulate intra-class variations.

3.4 Warping Maps

Before introducing the main hub of our method concerning how the variance-generating transformation fields are created, we initially need to present some necessary concepts and means, which are used in the successive subsection to analyze the intra-class variance.

DTW is an algorithm used to compute the similarity/distance between two time series. A cost matrix, denoted $\mathbf{W}$, is build progressively by computing the subtotal warping cost of aligning two series A and B. The cost is computed incrementally backwards until reaching a stopping condition in aligning the first points of the series. The overall cost is accumulatively computed at the topmost index value of the matrix, whose indices correspond to the length of series.

$$\begin{aligned} \text{DTW}(A, B) &= \mathbf{W}_{\text{length}(\mathbf{A}),\text{length}(\mathbf{B})} \\ \mathbf{W}_{1,1} &= (A_1 - B_1)^2 \\ \mathbf{W}_{i,j} &= (A_i - B_j)^2 + \min(\mathbf{W}_{i-1,j}, \mathbf{W}_{i,j-1}, \mathbf{W}_{i-1,j-1}) \end{aligned} \tag{1}$$

An optimal **warping path** between series A and B, defined as $\tau_{A,B}$ or here shortly τ, is a list of aligned indexes of points along the cost matrix. The sum of distances of the aligned points along the warping path should sum up to the

exact distance cost of DTW. The list of the warping path indexes pairs (i, j) corresponds to the chain of the recursive calls in the cost computation $W_{i,j}$. The sum of the distances among the values of the aligned indexes of two series, yields the minimum distance, which is equal to the DTW formulation.

$$\tau(A, B) = \{(i, j) \mid W_{i,j} \text{ called in the chain of recursion of } DTW(A, B)\} \quad (2)$$

A **warping map**, denoted M, is a square matrix whose elements are built by overlapping the warping paths of all-vs-all instances in an equi-length time series dataset. In this overlapping context, a cell of the warping map matrix denotes how often a warping alignment occur at that indexe. Equation 3 formalizes the procedure of building a warping map as a superposition (frequency) of warping paths of all time series pairs A, B from dataset S.

$$\mathbf{M}(i, j) \leftarrow \mid \{(A, B) \in S^2 \mid (i, j) \in \tau(A, B)\} \mid \quad (3)$$

A filtered warping map is created similarly as shown in Algorithm 2, where we filter only those warping paths that are either right or left aligned at a specific point. For instance, if we need to filter for right alignment at a point P, we need to build the DTW warping of any two series pairs, denoted τ, and then check if the aligned index at the second series is higher than (right of) the index on the first series. For instance, the notation $\tau(A, B)_P$ denotes the aligned index at series B corresponding to time P of first series A.

Algorithm 2. FilteredWarpingMap

Require: Dataset of time series **S**, Control Point P, Direction D
Ensure: Filtered warping map **M**
1: **if** $D = right$ **then**
2: $\quad \mathbf{M}(i, j) \leftarrow \mid \{(A, B) \in S^2 \mid (i, j) \in \tau(A, B) \wedge P < \tau(A, B)_P\} \mid$
3: **else**
4: $\quad \mathbf{M}(i, j) \leftarrow \mid \{(A, B) \in S^2 \mid (i, j) \in \tau(A, B) \wedge P \geq \tau(A, B)_P\} \mid$
5: **end if**
6: **return M**

In our forthcoming analysis we build warping paths by providing a filtered dataset of instances belonging to only one class. Therefore we will construct one warping map per class.

3.5 Variance Distribution Analysis and Creation of Transformation Fields

In this section we present the main method of creating the transformation, which is be based on the analysis of the variance distributions of warping paths. The transformation fields represent local perturbation vectors of the predefined regions, R many, by translating the representative control points. Each control point/region is translated both left and right, therefore creating $2 \times R$ total transformation fields. The amount of translational transformation to be applied

to every region is computed by making use of the warping maps. For each region, R_i, we filter in turn the right and left warping paths of instance warping alignments at that region, in order to analyze the general left/right translational variances of the warping alignments on other regions as an impact of a transformation at R_i. An illustration is found on Figure 3. The time series is divided into three regions defined by their centroid control points. In the image, part b), c), d) we show the filtered warping maps for the right warping alignments at every control points. Please note that three more filtered warping maps could be created for left alignments, but are avoided due to lack of space.

Once we built the warping map, we can successively construct the distribution of the warped alignments at every control points. For every region where we apply left/right translational perturbations, the *transformation field is created to be equal to the means of the warped points*, as an impact of the perturbation. The means are selected as transformation field, because they represents the tendency of variations at every control point. An illustration of the distributions is depicted in Figure 4. Only two distribution plots are shown belonging to the warping maps in Figure 3, part **b)** and **c)**.

Algorithm 3. ComputeTransformationFields

Require: Dataset of time series: S, A class: *label*, A list of control points CP
Ensure: List of transformation fields: L

1: $L \leftarrow \emptyset$
2: $S_{label} \leftarrow \{A \in S \mid A \text{ has class } label\}$
3: **for** $P \in$ CP **do**
4: **for** direction $\in \{\text{left}, \text{right}\}$ **do**
5: $M \leftarrow$FilteredWarpingMap$(S_{label}, P, \text{direction})$ from Algorithm 2
6: **for** $P' \in$ CP **do**
7: $F_j \leftarrow \frac{1}{\|M_{j,*}\|} \sum_{k=1}^{|S_{label}|} \left(M_{j,k} \cdot (k - P')\right), \quad j \in [1...|CP|]$
8: **end for**
9: $L \leftarrow L \cup \{F\}$
10: **end for**
11: **end for**
12: **return** L

For instance, the means of the distributions, which also form transformation fields at image a), represents the warping distributions as an impact of perturbation of R_1 are [34 24 0]. We can conclude that a right perturbation at R_1 causes a right translational impact on R_2, but fades away at R_3. Therefore the transformations of instances at R_1, will be in proportional to this distribution. Similarly in image b) there is a perturbation on R_2, which has a stronger impact on R_1 than on R_3.

The Algorithm 3 describes the creation of transformation fields. For every control point [line 3], we analyze the right and left variations [line 4] and get respective filtered warping maps [line 5]. The impact of such variation on other control points [line 6] is taken into consideration by the weighted mean variance at each other control point [line 7].

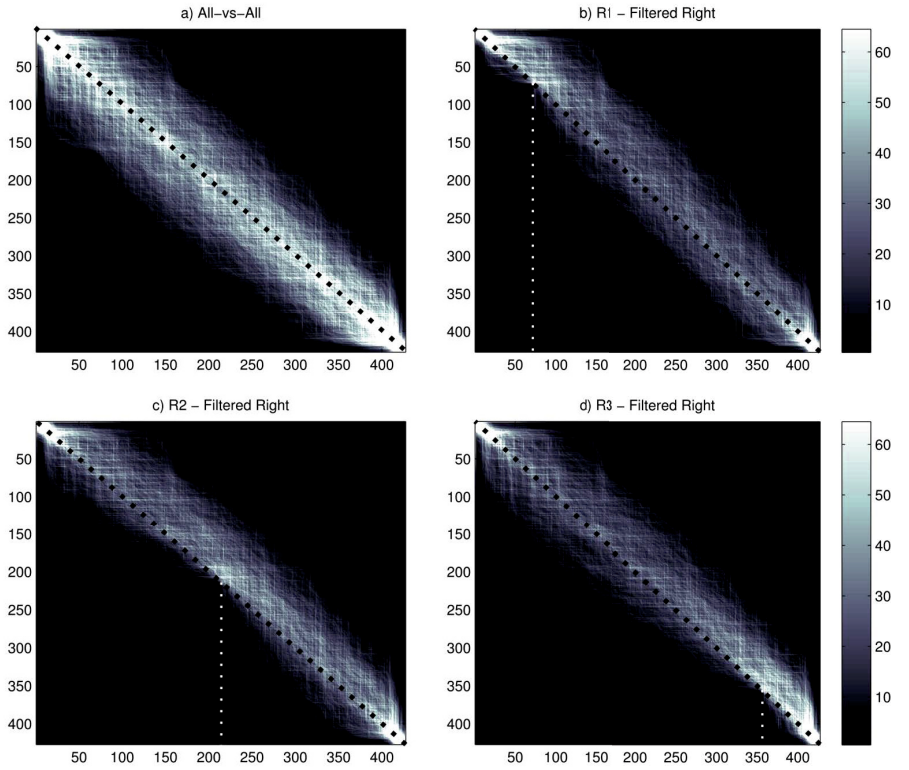

Fig. 3. An illustration concerning the warping map belonging to label 0 of the *OSULeaf* dataset. The time series are divided into three region R_1, R_2, R_3 for analysis. The center of each region is defined as a control point on time indices [71,213,355]. **a)** All-vs-all warping map. **b)** Warping map created by filtering only right warping paths at control point of R_1 at index 71. **c)** Warping map at control point of R_2 at index 213. **d)** A similar right warping filter of R_3 at 355.

3.6 Learning and Virtual Support Vectors

The defined transformation fields are used during the classification of time series. Even though in principle various classifiers can benefit from larger training set, still transforming all instances deteriorates the learning time of methods. SVMs have a crucial advantage because they point out the important instances (support vectors) which are needed to be transformed. In our study only the support vectors of a SVM model are transformed, called Virtual Support Vectors (VSV) [5]. Such selective approach ensures that the decision boundary is redefined only by instances close to it, hence the support vectors. The training set is extended to include MLS transformations of the support vectors as shown in Equation 4.

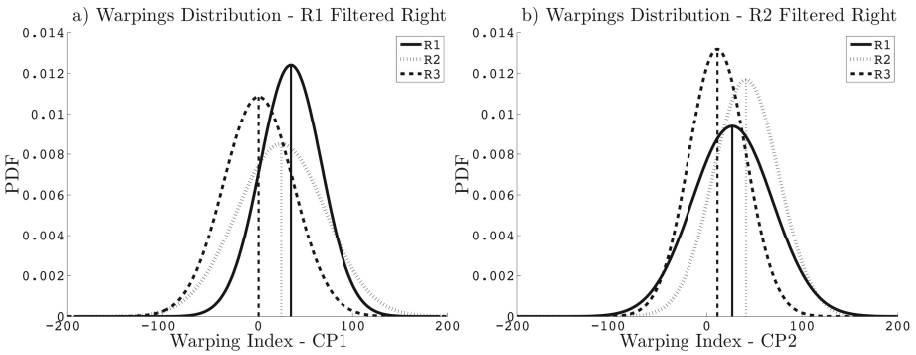

Fig. 4. Approximation with Gaussian Probability Density Functions (PDF) regarding the warping distribution of filtered warping maps in Figure 3, images b and c. **a)** The warping distribution as a result of right-warping occuring at R1/CP1. **b)** The warping distribution as a result of right-warping occuring at R2/CP2.

Given a list of control points, denoted CP, and a list of transformation fields, denoted TF, a transformation scale factor μ, then Equation 4 represents the addition of transformed support vectors obtained by building a model, denoted $svmModel$, from the training set S_{train}.

$$S^*_{train} = S_{train} \cup \{\ MLS\ (sv, \mu \cdot v, \text{CP}) \mid sv \in supportVectors(svmModel) \\ \wedge \quad v \in TF_{label(sv)}\} \tag{4}$$

Algorithm 4 describes the classification procedure in pseudo code style. The values of the transformation scales are computed by hyper-parameter search on a validation split search during the experiments.

4 Experimental Setup

In order to evaluate the performance of our proposed method, denoted as Invariant SVM, or shortly ISVM, we implemented and evaluated the following set of baselines:

- **SVM:** The default SVM is a natural choice for a baseline. The performance of our method compared to the standard SVM will give us indications on the success of redefining the decision boundaries, by injecting transformed support vectors.
- **DTW-NN:** Characterized as a hard-to-beat baseline in time series classification, which has been reported to achieve hard-to-beat classification accuracy [15]. The relative performance of our method compared to DTW-NN will give hints whether a refined maximum-margin is competitive, or not.

Algorithm 4. LearnModel

Require: Training set of time series: S_{train}, SVM hyper parameters: θ, Transformation Fields: TF, Control Points: CP, Transformation Scale: μ
Ensure: SVM model: $svmModel$
$svmModel \leftarrow svm.train(S_{train}, \theta)$
for $sv \in supportVectors(svmModel)$ **do**
  $label \leftarrow sv.label$
  $TF_{label} \leftarrow$ Field vectors of TF for class $label$
  for $v \in TF_{label}$ **do**
    $VSV \leftarrow MLS(sv, \mu \times v, CP)$ from Algorithm 1
    $S_{train} \leftarrow S_{train} \cup \{VSV\}$
  end for
end for
$svmModel \leftarrow svm.train(S_{train}, \theta)$
return $svmModel$

The UCR collection of time series dataset was selected for experimentation. Very few large datasets whose transformation fields creation was excessively time consuming were omitted. All the datasets were randomly divided into five subsets/folds of same size (5-folds cross-validation). Each random subset was stratified, meaning that, the number of instances per label was kept equal on all subsets. In turn each fold was used once for testing the method, while three out of the remaining four for training and one for validation. The inhomogeneous polynomial kernel, $k(x, y) = (\gamma\, x \cdot y + 1)^d$, was applied for both the standard SVM as well as our method. The degree d was found to perform overall optimal at value of 3. A hyper parameter search was conducted in order to select the optimal values of the kernel's parameter γ and the methods transformation scale μ of Algorithm 4, by searching for maximum performance on the validation set after building the model on the train set. SVM's parameter C was searched among $\{0.25, 0.5, 1, 2, 4\}$. The number of regions/control points was found to be optimal around 10. The performance is finally tested with a cross-validation run over all the splits[2].

5 Results

A cumulative results table involving the experimentation results is found in Table 1. For every dataset, the mean and standard deviations of the cross validation error rates is reported in columns. The non-overlapping 1-σ confidence interval results, representing significance of ISVM/SVM results versus DTW-NN, are annotated with a circle (°). We grouped the datasets into two categories, easy ones and challenging ones. The criteria of the split is based on an error rate threshold, where values greater than 5% of the default SVM are grouped as easy dataset. The values in bold indicate the best mean error rate for the respective

[2] The authors provide the source code upon request.

dataset/row. The last row indicates a sum of the wins for each method, where draw points are split to ties. In brackets we denote the wins with significant and non-significant intervals.

The first message of the experiments is that the performance of our method is improving the accuracy of a standard SVM. In various cases like 50words, Cricket_X, Cricket_Y, Cricket_Z, Lighting7, OSULeaf, WordsSynonyms, the improvement is very significant ranging from +5% up to +11% accuracy. Thus it is appropriate to use our method for boosting the SVM accuracy, without adding noise.

The second and more important message is that our method produces better mean error rates than DTW-NN, winning on the majority of the datasets. The performance on the easy datasets is even (7 to 7). However, our method outperforms DTW-NN on the majority (11 to 5) of the challenging datasets. The invariant SVM looses significantly only on Cricket_* and Lighting2. In contrast, it significantly outperforms DTW-NN on 50words, Fish, OliveOil, SwedishLeaf, uWaveGestureLibrary_* and WordsSynonyms. Thus, our experiments demonstrate the superiority of our approach to DTW-NN.

The transformation scale parameter introduced in Algorithm 3, controls the scale of the transformation fields perturbations to be applied to the instances. Intuitively, optimal transformation fields redefine the decision boundary, while excessive magnitudes of transformations deteriorate into noisy instances. A demonstration of the transformation fields' scale parameter behavior is presented in Figure 5.

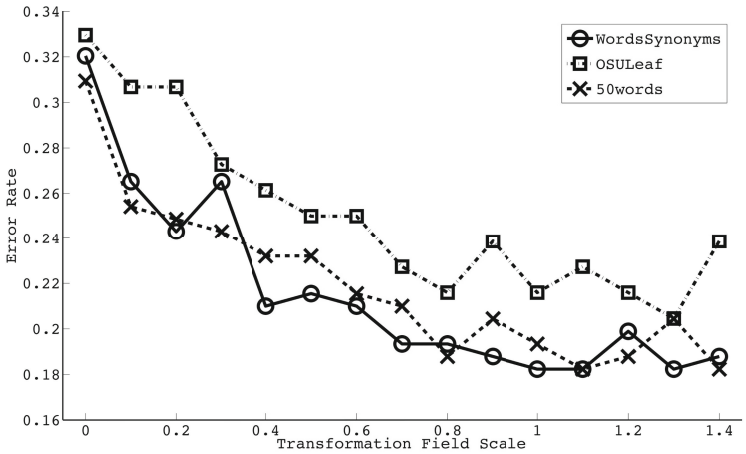

Fig. 5. The effect of increasing the transformation field scale on three typical datasets. The accuracy improves proportionally with the scale, until a minimum point is reached. After the optimal error rate, the large transformations produce noise and deteriorate accuracy, as for instance in OSULeaf, after minimum at scale value 1.3.

Table 1. 5-fold Cross-Validation Experiments Results - Error Rate Fractions

(i) **ISVM** *: Our proposed method Invariant Support Vector Machines*
(ii) **DTW-NN** *: Nearest Neighbor with Dynamic Time Warping*
(iii) **SVM***: The default Support Vector Machines*

Dataset	**ISVM**		**DTW-NN**		**SVM**	
	mean	*st.dev.*	*mean*	*st.dev.*	*mean*	*st.dev.*
Easy Datasets						
CBF	0.002	0.003	**0.000**	0.000	0.002	0.003
Coffee	°**0.000**	0.000	0.073	0.010	°**0.000**	0.000
DiatomSizeReduction	°**0.000**	0.000	0.006	0.000	°**0.000**	0.000
ECGFiveDays	°**0.001**	0.003	0.007	0.000	°**0.001**	0.003
FaceAll	**0.020**	0.007	0.024	0.000	0.027	0.005
FaceFour	**0.036**	0.038	0.054	0.006	0.045	0.056
FacesUCR	°**0.021**	0.007	0.031	0.000	0.026	0.010
Gun_Point	°**0.030**	0.027	0.075	0.001	0.050	0.040
ItalyPowerDemand	°**0.026**	0.019	0.051	0.000	°**0.026**	0.019
MoteStrain	0.050	0.016	**0.045**	0.000	0.060	0.020
SonyAIBORobotSurface	°**0.008**	0.011	0.027	0.000	°**0.008**	0.011
SonyAIBORobotSurfaceII	0.004	0.004	0.029	0.000	°**0.003**	0.005
Symbols	0.027	0.016	**0.019**	0.000	0.029	0.012
synthetic_control	0.020	0.013	°**0.007**	0.000	0.022	0.015
Trace	0.030	0.027	°**0.000**	0.000	0.045	0.048
TwoLeadECG	0.002	0.002	**0.001**	0.000	0.002	0.002
Two_Patterns	0.001	0.001	°**0.000**	0.000	0.006	0.001
wafer	°**0.002**	0.002	0.006	0.000	°**0.002**	0.001
Wins (Sig./Non-Sig.)	**7 (5/2)**		**7 (3/4)**		**4 (4/0)**	
Challenging Datasets						
50words	°**0.199**	0.008	0.287	0.001	0.272	0.035
Adiac	°**0.202**	0.038	0.341	0.001	0.206	0.037
Beef	°**0.267**	0.109	0.467	0.009	°**0.267**	0.109
Cricket_X	0.300	0.025	°**0.197**	0.001	0.391	0.033
Cricket_Y	0.271	0.021	°**0.213**	0.001	0.388	0.045
Cricket_Z	0.306	0.043	°**0.188**	0.000	0.399	0.031
ECG200	**0.110**	0.052	0.160	0.003	0.125	0.040
Fish	°**0.094**	0.031	0.211	0.000	0.103	0.031
Lighting2	0.289	0.025	°**0.099**	0.002	0.297	0.013
Lighting7	**0.252**	0.054	0.285	0.010	0.357	0.068
OliveOil	**0.083**	0.083	0.133	0.006	**0.083**	0.083
OSULeaf	0.296	0.039	**0.285**	0.003	0.346	0.059
SwedishLeaf	°**0.079**	0.017	0.185	0.001	0.082	0.014
uWaveGestureLibrary_X	°**0.193**	0.012	0.251	0.000	0.197	0.013
uWaveGestureLibrary_Y	°**0.253**	0.009	0.342	0.000	0.259	0.009
uWaveGestureLibrary_Z	°**0.249**	0.008	0.301	0.000	0.256	0.011
WordsSynonyms	°**0.200**	0.038	0.270	0.000	0.261	0.027
Wins (Sig./Non-Sig.)	**11 (9/2)**		**5 (4/1)**		**1 (0.5/0.5)**	

Finally, it is worth mentioning that the time complexity of our method is obviously worse than that of a normal SVM, because of the enlarged training set. Yet, the computational time is not prohibitive in terms of run-time feasibility. In Table 2 you can find some test run times in minutes, of typical prototypes of easy and challenging datasets, with the aim of demonstrating the run time feasibility of our method compared to DTW-NN. The run-time minutes shown on the last column are measured over the same random dataset fold.

Table 2. Classification Run Times of Typical Dataset

Dataset	**# labels**	**# instances**	**length**	**ISVM** Time(min)
Coffee	2	56	286	0.01
ECG200	2	200	96	0.04
wafer	2	7174	152	5.48
FacesUCR	14	2250	131	8.73
50words	50	905	270	14.17
Cricket_X	12	780	300	24.00

6 Conclusion

This study we introduced a novel instance transformation method, which is used to boost the performance of SVM via transforming the support vectors. The proposed method utilizes the distribution of warping variances, yield from warping alignment maps, in order to define transformation fields, which represent variances at a predefined set of local regions of a particular class. Therefore the virtual support vectors which are generated by applying the transformation fields, represent the necessary intraclass variation and redefines the maximum margin decision boundary. The superiority of our method is demonstrated by extensive experimentations on 35 datasets of the UCR collection. In a group of easy datasets, the presented method is on a par competitive to the baselines, while being clearly superior on a set of challenging datasets.

As a planned future work, we will focus on analyzing the joint operation of our method with possible inclusion of efficient similarity based metric into the SVM kernel. In parallel, other state-of-art invariant classifiers will be explored, in particular Convolutional Neural Networks.

References

1. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications With R Examples. Springer (2011) ISBN 978-1-4419-7864-6
2. Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: KDD, pp. 285–289 (2000)
3. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB 1(2), 1542–1552 (2008)

4. Gudmundsson, S., Runarsson, T.P., Sigurdsson, S.: Support vector machines and dynamic time warping for time series. In: IJCNN, pp. 2772–2776. IEEE (2008)
5. Schölkopf, B., Burges, C., Vapnik, V.: Incorporating invariances in support vector learning machines, pp. 47–52. Springer (1996)
6. DeCoste, D., Schölkopf, B.: Training invariant support vector machines. Machine Learning 46(1-3), 161–190 (2002)
7. Schaefer, S., McPhail, T., Warren, J.D.: Image deformation using moving least squares. ACM Trans. Graph. 25(3), 533–540 (2006)
8. Kehagias, A., Petridis, V.: Predictive modular neural networks for time series classification. Neural Networks 10(1), 31–49 (1997)
9. Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time-series data. International Journal of Computer Research 10, 49–61 (2001)
10. Pavlovic, V., Frey, B.J., Huang, T.S.: Time-series classification using mixed-state dynamic bayesian networks. In: CVPR, p. 2609. IEEE Computer Society (1999)
11. Rodríguez, J.J., Alonso, C.J.: Interval and dynamic time warping-based decision trees. In: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC 2004, pp. 548–552. ACM, New York (2004)
12. Rodríguez, J.J., Alonso, C.J., Boström, H.: Learning First Order Logic Time Series Classifiers: Rules and Boosting. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 299–308. Springer, Heidelberg (2000)
13. Kim, S., Smyth, P., Luther, S.: Modeling waveform shapes with random effects segmental hidden markov models. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI 2004, Arlington, Virginia, United States, pp. 309–316. AUAI Press (2004)
14. Mizuhara, Y., Hayashi, A., Suematsu, N.: Embedding of time series data by using dynamic time warping distances. Syst. Comput. Japan 37(3), 1–9 (2006)
15. Xi, X., Keogh, E.J., Shelton, C.R., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: ICML. ACM International Conference Proceeding Series, vol. 148, pp. 1033–1040. ACM (2006)
16. Keogh, E.J., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowl. Inf. Syst. 7(3), 358–386 (2005)
17. Niyogi, P., Girosi, F., Poggio, T.: Incorporating prior information in machine learning by creating virtual examples. Proceedings of the IEEE, 2196–2209 (1998)
18. Loosli, G., Canu, S., Vishwanathan, S.V.N., Smola, A.J.: Invariances in classification: an efficient svm implementation. In: Proceedings of the 11th International Symposium on Applied Stochastic Models and Data Analysis (2005)
19. Loosli, G., Canu, S., Bottou, L.: Training invariant support vector machines using selective sampling. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines, pp. 301–320. MIT Press, Cambridge (2007)
20. Shimodaira, H.: Noma, K.-i., Nakai, M., Sagayama, S.: Support vector machine with dynamic time-alignment kernel for speech recognition. In: Dalsgaard, P., Lindberg, B., Benner, H., Tan, Z.H. (eds.) INTERSPEECH, ISCA, pp. 1841–1844 (2001)
21. Bahlmann, C., Haasdonk, B., Burkhardt, H.: On-line handwriting recognition with support vector machines - a kernel approach. In: Proc. of the 8th IWFHR, pp. 49–54 (2002)
22. Zhang, D., Zuo, W., Zhang, D., Zhang, H.: Time series classification using support vector machine with gaussian elastic metric kernel. In: ICPR, pp. 29–32. IEEE (2010)
23. Rahimi, A., Recht, B., Darrell, T.: Learning to transform time series with a few examples. IEEE Trans. Pattern Anal. Mach. Intell. 29(10), 1759–1775 (2007)