Model-Based Similarity Estimation of Multidimensional Temporal Sequences

Romain Tavenard `rtavenar@irisa.fr` *IRISA/ENS Cachan*

Laurent Amsaleg `lamsaleg@irisa.fr` *IRISA/CNRS*

Guillaume Gravier `ggravier@irisa.fr` *IRISA/CNRS*

Abstract

Content-based queries in multimedia sequence databases where information is sequential is a tough issue, especially when dealing with large scale applications. One of the key points is similarity estimation between a query sequence and elements of the database. In this paper, we investigate two ways to compare multimedia sequences, one – that comes from the literature – being computed in the feature space while the other one is computed in a model space, leading to a representation less sensitive to noise. We compare these approaches by testing them on a real audio dataset, which points out the utility of working in the model space.

Keywords: Multidimensional feature sequences, Support Vector Regression, Temporal aspects, Similarity estimation in a model space

1 Motivations

Our everyday life is now flooded by multimedia sequence databases: access to TV archives, production and sharing of personal videos, podcasts.

Nevertheless, there are still few techniques allowing fast content-querying of such databases. The need for such techniques is especially important in the field of multimedia archiving where one wants to annotate streams in order to allow future documentary search. This implies segmentation of a (video or audio) stream into distinct programmes that can later be annotated, which requires the ability to query-by-content huge databases in order to temporally localize items such as advertisement, TV show credits, Mr X's interventions on radio channel Y, and so on.

Until now, there has been no technique efficient enough to process real-size audiovisual archives – tens or hundreds of thousands of hours. One of the main problems comes from the temporal aspect of such sequences: working with audio or video implies to deal with multidimensional feature sequences, a sequence being a series of features whose order is important. Because of the amount of data that is concerned, indexing is necessary. There has been numerous works on the indexing of still images [AI06, NS06, LÁJA09] and the indexing of temporal media is often viewed as a simple extension of these [LTCJ$^+$07]. Nevertheless, these works only use post-processing temporal robustification, instead of considering the sequential nature of the media at each step of the process. There has also been a great interest, in the database community, for the mining of time series data [DTS$^+$08] that is often dedicated to the study of symbolic sequences, while the problems we want to address require the use of numerical features.

We intend, in this paper, to present similarity measures that could, in future works, be used in a fully temporal indexing scheme with application to audiovisual data. To do so, one has to investigate ways to compare multidimensional feature sequences extracted from multimedia sequences. The point here is to estimate similarity between a query sequence and a database of reference sequences. We explore two ways of achieving such a goal. The first one comes from the state-of-the-art and is based on a direct comparison of the

features by means of the computation of an alignment cost given by dynamic time warping (DTW) [SC78]. The second one is based on models of sequences and the similarity is computed at the model level. The models we suggest to use are based on support vector machine theory [Vap95] and, more precisely, on support vector regression (SVR). DTW and SVR are essentially different: while the former is very close to the actual features, trying to directly align sequences, the latter computes similarity estimation at a higher abstraction level, in the model space. In other words, instead of comparing sequences, we propose to model these sequences using SVR and then compare the resulting models, stating that if models are similar, then sequences on which they were fitted should also be similar.

The aim of this study is to evaluate the ability of each method to cope with distortions such as usual temporal distortions (stretching, shrinking, timeshifting) or the presence of outliers in the sequences (due to compression for example). We also intend to test the ability of each method to retrieve subsequences, which correspond to real-life issues such as recognizing a song from which a sample was recorded. For this study to be statistically sounded, we performed our experiments on a real dataset made of 100 hours of radio recordings, leading to more reliable conclusions than in [TAG07].

DTW in its original form is incapable of comparing timeshifted sequences. Because of that, segmentation of the stream into well-defined pieces, which is a tough issue [BGG99], is necessary as DTW is very sensitive to shifts on initial and ending times. On the contrary, DTW is known to cope efficiently with local and global temporal distortions in non-timeshifted sequences. We intend to study the behaviour of SVR in presence of these same distortions. We also investigate the characteristics of SVR and DTW based similarity measures in order to point out their advantages and drawbacks for later application to the indexing of large temporal datasets.

This paper contribution is then double: first we suggest to use SVR to model sequences and process comparison at the model level, and we then validate this approach on a real audio dataset.

This paper is structured as follows. Sections 2 and 3 explain both methods. Section 4 gives a theoretical comparison of these, while section 5 describes the experimental setup and the results. Section 6 concludes and gives prospects.

2 Dynamic Time Warping (DTW)

The dynamic time warping (DTW) algorithm was introduced by [SC78] and is now widely used to compare sequences of possibly different lengths in databases [Keo02, YJF98]. The principle on which DTW relies is to locally stretch both sequences in order to achieve the most probable alignment between them and compensate for their possible time distortions. To do so, one allows for instertions and deletions in both streams, each of these operations having a cost. The aim is then to find the set of operations that will lead to a minimal cost.

Using DTW to compare two sequences boils down to building a similarity matrix that contains the distances between all elements of the 2 sequences to compare and finding the optimal path inside this matrix. Note that since DTW relies on a similarity matrix, it is very sensitive to outliers.

This section gives an overview of the basic idea behind DTW, presents typical optimizations that can be used and a way to release DTW border constraints.

2.1 Basics

Let $Q = q_1, q_2, \ldots, q_n$ and $C = c_1, c_2, \ldots, c_m$ be two sequences of features to compare. The first step consists in building a similarity matrix S of size $n \times m$, such that:

$$\forall (i,j) \in [1,n] \times [1,m], s_{i,j} = dist(q_i, c_j) \tag{1}$$

where $dist(.,.)$ is the distance defined in the feature space (typically Euclidian distance). Let us denote W a path, of length K such that $w_1 = (1,1)$ and $w_K = (n,m)$, leading to a minimum cost:

$$DTW(Q,C) = \min_W \left(\sum_{i=1}^{K} S(w_i) \right) \tag{2}$$

Such a path has to be continuous and monotone, which can be written:

$$(w_k = (a, b) \text{ and } w_{k-1} = (a', b')) \Rightarrow \begin{cases} (a = a' + 1 \text{ and } b = b' + 1) \text{ or} \\ (a = a' + 1 \text{ and } b = b') \text{ or} \\ (a = a' \text{ and } b = b' + 1) \end{cases} \quad (3)$$

This path and its cost can be computed using dynamic programming to evaluate the partial cost $\gamma_{i,j}$ of the best path up to (i, j) at each step:

$$\gamma_{i,j} = s_{i,j} + \min\{\gamma_{i-1,j-1}, \gamma_{i-1,j}, \gamma_{i,j-1}\} \quad (4)$$

This gives a simple way to compute DTW:

$$DTW(Q, C) = \gamma_{n,m} \quad (5)$$

Note that, for this algorithm to return the path it used, one just has to store each element's predecessor (*i.e.* , $\arg\min\{\gamma_{i-1,j-1}, \gamma_{i-1,j}, \gamma_{i,j-1}\}$) at each step.

2.2 Typical optimizations

This algorithm is quadratic in the length of the sequences, which is one of its limitations. Several optimizations have been proposed to reduce computing time. They all rely on approximations of DTW.

Some of them [SC78] aim at not fully building the similarity matrix, considering that diagonal paths are more likely than the other ones. This can be done in two ways: the first one is to set the elements of S that are far from the diagonal to ∞, the other is to slightly modify possible transitions, changing insertions (respectively deletions) into a combination of insertions (respectively deletions) and matches.

Other methods are based on the computation of lower bounds for DTW. These lower bounds can then be used to filter out dissimilar enough sequences. Indeed, in the case of ε-search in a database given a query Q, if LB denotes the considered lower bound, the following implication stands:

$$\forall C, (LB(Q, C) > \varepsilon) \Rightarrow (DTW(Q, C) > \varepsilon) \quad (6)$$

Then, exact DTW can be computed on sequences from the database for which $LB(Q, C)$ is lower than the threshold ε. The point here is then to find a lower bound that is both easy to compute and close to the actual DTW. [YJF98] proposes a way to compute such a lower bound by referring to extrema of both sequences, which has been improved by [Keo02] considering the high probability of diagonal paths. In both cases the computational cost of the lower bound is linear, but the second one has the advantage of focusing on a small portion of C, leading to a more accurate approximation of the DTW. Note that the lower bound introduced by [Keo02] is a lower bound for the DTW with restriction to almost diagonal paths, not for the standard DTW.

2.3 Releasing constraints

The basic implementation of DTW shows one major drawback: as the path is supposed to start from $w_1 = (1, 1)$ and end in $w_K = (n, m)$, sequences are considered as not timeshifted. When sequences to compare are timeshifted, what one would want to do would be to compare similar parts of the sequences, instead of comparing the whole sequences. This implies to find anchor points in sequences to extract similar subsequences to be compared, which is very time consuming [AGM+90].

[MGB09] propose to solve this issue by slightly modifying the algorithm described above in such a way that the border constraints become $w_1 = (i_1, 1), i_1 \in [1, n]$ and $w_K = (i_2, m), i_2 \in [1, n]$. The principle here is just to alter initialization by stating that the path can start from any element of sequence Q:

$$\forall i_1 \in [1, m], \gamma_{0,i_1} = 0 \quad (7)$$

and normalize the cumulative similarity measure $\gamma_{i,j}$ by the length of the best matching path ending at point (i, j). The similarity measure then becomes the minimal cost path ending in any element of sequence Q:

$$DTW(Q, C) = \min_{i_2 \in [1,n]} \frac{\gamma_{i_2,m}}{l(i_2)} \tag{8}$$

where $l(i_2)$ is the length of the path leading to (i_2, m).

Note that this released version of DTW is incompatible with the computation of the lower bound introduced in [Keo02], as the diagonal path cannot be considered as the most probable.

3 Support Vector Regression (SVR)

We propose another approach to sequence comparison that consists in modeling sequences and then process comparison at the model level. The class of models we investigate in this paper comes from support vector machine (SVM) theory [Vap95]. SVM are known to be very efficient classifiers. The core idea with SVM is to find the separating hyperplane maximizing the gap between two sets of elements using kernel functions that allow for non-linear classification. SVM can also be used in regression problems [VGS96, SS98] (it is then called SVR) consisting in the approximation of a function $g : \mathbb{R}^H \to \mathbb{R}$ given a set of N observations $\{\mathbf{x_i}, y_i\}_{i=1}^N$ such that $y_i = g(\mathbf{x_i}) + \eta$, η being considered as noise.

For both classification (for which $sgn(g)$ is the classifier) and regression, g is modeled as

$$\hat{g}(\mathbf{x}) = \sum_{i=1}^{L} c_i \phi_i(\mathbf{x}) + b \tag{9}$$

where $\{\phi_i\}_{i=1}^L$ denotes a set of basis functions and coefficients c_i and b have to be optimized. The optimization step consists in minimizing

$$\frac{1}{2} \sum_{i=1}^{L} c_i \phi_i(\mathbf{x_i}) + C \sum_{i=1}^{L} (\xi_i + \xi_i^*) \tag{10}$$

subject to

$$\begin{aligned} y_i - \hat{g}(\mathbf{x_i}) &\leq \varepsilon + \xi_i & (11)\\ \hat{g}(\mathbf{x_i}) - y_i &\leq \varepsilon - \xi_i^* & (12)\\ \xi_i, \xi_i^* &\geq 0 & (13) \end{aligned}$$

where the constant $C > 0$ determines the trade-off between the flatness of $\hat{g}$ and the amount up to which deviations larger than ε are tolerated. [Vap95] showed that the function minimizing the quantity in (10) can be expressed as

$$\hat{g}(\mathbf{x}, \alpha) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b \tag{14}$$

under the condition

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^H K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{L} \phi_i(\mathbf{x}) \phi_i(\mathbf{y}) \tag{15}$$

The minimization of the quantity in (10) is done iteratively and loops while the approximation error is greater than a threshold ε_2.

Only a few terms of the sum in (14) will be non null. The corresponding $\mathbf{x_i}$ will therefore be called support vectors. The more support vectors, the better the approximation. Figure 1 presents the model of the *sinc* function using two different values for ε and C, showing the effect of these parameters in the number of support vectors in the model.

Note that (14) does not refer to the set of ϕ_i functions anymore, but only to its associated kernel K that has to be a scalar product in some unknown space. Mercer's theorem [Mer09] gives a caracterization of all acceptable kernel functions. The most common ones are summarized in table 1.

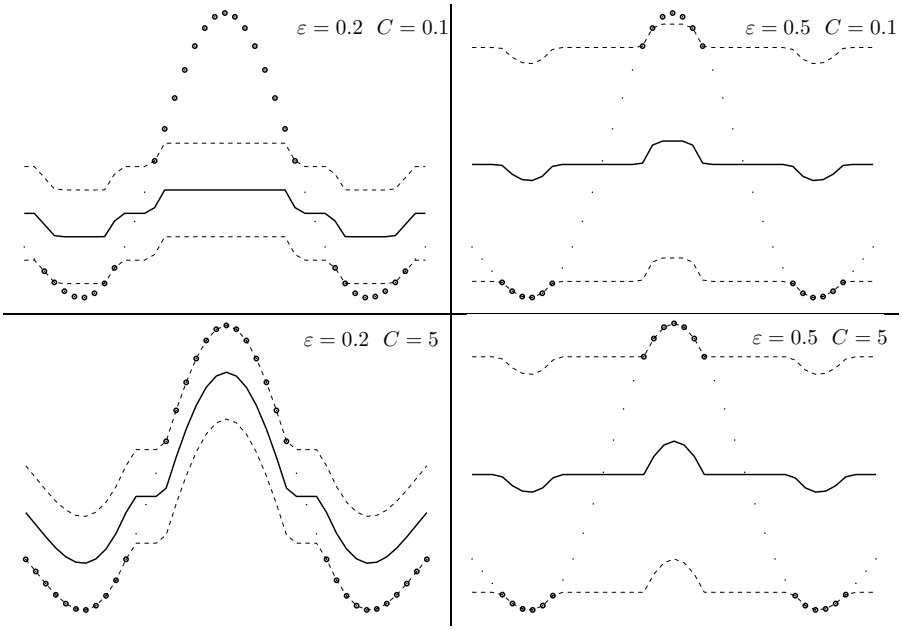

Fig. 1: Approximation of a *sinc* function. The function is sampled and plotted as dots, supports vectors as filled circles, the extrapolated function as solid line and the dash lines represent the ε enveloppe of the optimized function. One can easily notice that support vectors are concentrated in the regions where the function is more difficult to predict. Moreover, the smaller ε, the better the approximation and the larger C, the fewer points outside the enveloppe.

Denomination	Form of K
Linear kernel	$K(x, y) = (x \cdot y)$
Polynomial function of degree k	$K(x, y) = [(x \cdot y) + 1]^k$
Radial basis function (RBF)	$K(x, y) = \exp[-\gamma \|x - y\|^2]$
2-level neural networks	$K(x, y) = \tanh[(x \cdot y)v + c]$

Tab. 1: Usual kernels for SVM.

3.1 Modeling sequences using SVR

Assuming independance between the d dimensions of a sequence, modeling the latter using SVR can be done by fitting one model per dimension. The set of d models then defines a super-model of the evolution of our features along time.

For each $i \in [1, d]$, a model is built that explicits, for each timestamp $t \in [1, m]$ – where m is the length of the corresponding sequence – the relationship between $x_{i,t}$ and its neighbours $\{x_{i,\tilde{t}}\}_{\tilde{t} \in v_t}$ where $v_t = \{t - \frac{H}{2}, \cdots, t-1, t+1, \cdots, t + \frac{H}{2}\}$ denotes the temporal neighbourhood of $x_{i,t}$, of size H, that we choose to be centered around t. The set of $\{g_i\}_{i \in [1,d]}$ functions is then such that $x_{i,t} = g_i(\{x_{i,\tilde{t}}\}_{\tilde{t} \in v_t}) + \eta$ for each i.

Each sequence is now represented by a set of d models. Comparison between two sequences can then be achieved by estimating the similarity between their respective models. Unfortunately, there is no known parametric distance between models, except for the linear kernel. For other kernels, one has to compute cross-similarity [BMM03].

In other words, if $G_Q = \{g_{Q,i}\}_{i \in [1,d]}$ is the set of estimated models for sequence $\{\mathbf{Q}_t\}_{t=1}^n = \{\{q_{1,t}, \cdots, q_{d,t}\}\}_{t=1}^n$ and $\{\mathbf{C}_t\}_{t=1}^m$ is another sequence one wants to compare with $\{\mathbf{Q}_t\}$, let us build a new sequence $\{\hat{\mathbf{C}}_t\}_{t=1}^m$ such

that, for each $t \in [1, m]$,

$$\hat{\mathbf{C}}_t = G_Q(\mathbf{C}_{t-\frac{H}{2}}, \ldots, \mathbf{C}_{t-1}, \mathbf{C}_{t+1}, \ldots, \mathbf{C}_{t+\frac{H}{2}}) \tag{16}$$

$\{\hat{\mathbf{C}}_t\}_{t=1}^m$ is the sequence of values predicted by G_Q for $\{\mathbf{C}_t\}$. Both sequences are then considered as similar if $\{\hat{\mathbf{C}}_t\}$ is close to $\{\mathbf{C}_t\}$, or, in other words, if the model learnt on $\{\mathbf{Q}_t\}$ could predict $\{\mathbf{C}_t\}$ precisely enough.

As $\{\mathbf{C}_t\}$ and $\{\hat{\mathbf{C}}_t\}$ are, by construction, aligned, L_2-norm can then be used for similarity estimation:

$$Sim(\{\mathbf{Q}_t\}, \{\mathbf{C}_t\}) = \sqrt{\frac{1}{nd}\sum_{i=1}^{d}\sum_{t=1}^{n}(\hat{c}_{i,t} - c_{i,t})^2} \tag{17}$$

We chose L_2-norm instead of L_1 or L_∞ because we wanted to strongly penalize large prediction errors without focusing on a single outlier.

4 Compared study of the DTW and SVR approaches

As seen in section 1, SVR and DTW need to have certain characteristics in order to be useable, in a later step, for multimedia indexing. These characteristics can be tightly linked to those of semi-distances, which are

- Weak separation: $\forall x, dist(x, x) = 0$
- Symetry: $\forall (x, y), dist(x, y) = dist(y, x)$
- Triangular inequality: $\forall (x, y, z), dist(x, z) \leq dist(x, y) + dist(y, z)$

These three items correspond to "natural" characteristics that one can expect from a similarity measure. Moreover, the last one can be helpful in order to avoid some comparisons between elements of the database and the query knowing similarity between elements of the database that can be computed offline [CPZ97]. Two other items can be added to this good property list that are low computational complexity and robustness to distortions natural sequences could suffer.

Both approaches (relaxed DTW and SVR) are based on dissymmetric similarity measures. This means that there are two ways of estimating the similarity between two sequences $\{\mathbf{Q_t}\}$ and $\{\mathbf{C_t}\}$: $S_1 = Sim(\{\mathbf{C_t}\}, \{\mathbf{Q_t}\})$ and $S_2 = Sim(\{\mathbf{Q_t}\}, \{\mathbf{C_t}\})$, where $Sim(.,.)$ is the related similarity measure. Considering DTW, S_1 (resp. S_2) denotes alignment of $\{\mathbf{C_t}\}$ (resp. $\{\mathbf{Q_t}\}$) with a subsequence of $\{\mathbf{Q_t}\}$ (resp $\{\mathbf{C_t}\}$). Regarding SVR, S_1 (resp. S_2) is the error obtained while trying to predict $\{\mathbf{C_t}\}$ (resp. $\{\mathbf{Q_t}\}$) using a model fitted on $\{\mathbf{Q_t}\}$ (resp. $\{\mathbf{C_t}\}$).

We then study the behaviour of DTW and SVR with respect to the above items.

4.1 Weak separation

Regarding the weak separation paradigm, DTW in all the forms presented in section 2 satisfies it: one just has to use the diagonal path which is made of null values in the case of auto-similarity estimation (*dist* being a distance function, $dist(q_i, q_i) = 0$ for all i). SVR does not rigorously satisfy this weak separation paradigm but the following inequality stands:

$$\forall \{\mathbf{Q}_t\}, Sim(\{\mathbf{Q}_t\}, \{\mathbf{Q}_t\}) \leq \varepsilon_2 \tag{18}$$

For small values of ε_2, SVR will then follow the philosophy of this paradigm.

4.2 Enabling symetry

Note that, for both DTW and SVR, S_1 should give more accurate estimation of the similarity when $\{\mathbf{C_t}\}$ is a subsequence of $\{\mathbf{Q_t}\}$, whereas S_2 should be more efficient when $\{\mathbf{Q_t}\}$ turns to be a subsequence of $\{\mathbf{C_t}\}$. Both similarity measurements are then complementary and using only one of them would induce information loss. That is why we introduce two new similarity measures in order to fuse information between S_1 and S_2:

$$\begin{aligned} S_{avg}(\{\mathbf{Q_t}\},\{\mathbf{C_t}\}) &= \frac{S_1+S_2}{2} \\ S_{min}(\{\mathbf{Q_t}\},\{\mathbf{C_t}\}) &= \min(S_1,S_2) \end{aligned}$$

S_{avg} and S_{min} are then symetric similarity measures.

4.3 Triangular inequality

As written above, similarity measure's satisfying the triangular inequality can be very useful for application to indexing of large databases.

In the case of numerical sequences (which is the one we are interested in), neither DTW nor its associated lower bounds satisfy triangular inequality, as insertions, deletions and substitutions do not have a constant cost, as for the Levenshtein distance. That is why [CN04] introduced a new distance measure based on DTW with constant cost for insertions and deletions. Its main drawback is that it weakens DTW's powerful principle that the returned cost is equal to the exact sum of all costs needed to align both sequences.

Concerning SVR, the similarity measure we introduced does not satisfy triangular inequality neither. Indeed, it is not possible to state that

$$Sim(\{\mathbf{Q}_t\},\{\mathbf{C}_t\}) \leq Sim(\{\mathbf{Q}_t\},\{\mathbf{Y}_t\}) + Sim(\{\mathbf{Y}_t\},\{\mathbf{C}_t\}) \tag{19}$$

for any $\{\mathbf{Y}_t\}$ as the sequence $\{\hat{\mathbf{Y}}_\mathbf{t}\}$ is built using a model optimized on $\{\mathbf{Y}_t\}$ and, thus, not fully predictable.

In other words, for both DTW and SVR, the similarity measures we introduced do not satisfy triangular inequality.

4.4 Computational complexity

Even if our study is exclusively focused on similarity estimation, one has to keep in mind that these similarity measures are to be used in a complete data indexing scheme. Therefore, computational complexity of both similarity measures is a crucial point for application to large databases.

Let us suppose that both sequences are made of N elements of dimension d. DTW requires to fill coefficients for the similarity matrix, leading to the computation of N^2 terms, each being a distance between features in dimension d. Thus, temporal complexity is $\mathrm{O}(d \times N^2)$. Yet, there are possible improvements, as presented in section 2.2, consisting in evaluating a lower bound for DTW [Keo02] (temporal complexity becomes $\mathrm{O}(d \times N)$ but the result is only an estimate of the real DTW similarity measure) or constraining the path to be as diagonal as possible (temporal complexity is still quadratic but the constant term is much lower[1]).

SVR necessitates prediction of $N \times d$ values and, for each prediction, one needs to compute the sum of n terms, where n denotes the number of support vectors in the model. In practice, we have $n \sim N$ in our experiments and the temporal complexity is then $\mathrm{O}(d \times N^2)$.

4.5 Robustness

Both approaches have to be robust to some usual distortions that sequences could suffer. These distortions are, most of the time, due to post-production operations such as compression (that might lead to the presence of outliers) or tempo modification. Timeshifting is another kind of time distortions that comes from our will not to use a costly segmentation pre-process.

[1] Note that this method might overestimate dissymilarity if the natural path contains a significant amount of non-diagonal parts.

Outliers. In the case of real sequences, some features can be noisy. Hence, sensitivity of the similarity measure to outliers in the sequences is a crucial point. DTW, as it is directly computed on signal descriptions, suffers from such distortions. Indeed, a single outlier in one of the sequences to compare seriously modifies the optimal path and therefore alters the similarity measure. On the contrary, SVR based similarity estimation should be more robust to this kind of distortions, as SVMs are well-known to be capable of avoiding overfitting.

Stretching. Real sequences can be stretched or shrinked for several reasons. For example, radio channels sometimes shrink tunes they broadcast in order to catch up with their timeline. DTW is designed to deal with this kind of distortions, as it allows insertions and deletions in sequences. SVR is not as devoted to that task as DTW, but as it models the temporal evolution of the signal, it should also be able to deal with such time distortions.

Timeshifting. Another kind of time distortion that similarity measures have to face is timeshifting. SVR can deal with it quite accurately as it only requires a small neighbourhood of the point to estimate the value in this point, without considering its global time location. DTW, even with released contraints, will compute the shortest path to align a whole sequence inside the other one whereas, in the case of timeshifed sequences the problem would consist in aligning parts of both sequences.

5 Experiments

In this section, we aim at empirically comparing the two methods presented above (namely relaxed DTW and SVR) for a task of content-based sequence retrieval in a database. We first used a subset of the database to fit the parameters for the SVR and then compared both methods on the full database.

5.1 Experimental Setup

The database consists of 100 hours of audio divided into 10 hours of a French radio channel called *France Info* and of 90 hours of another French radio channel called *FIP*. These records contain show jingles and other repeating parts.

The two data streams were segmented into pieces of 5 seconds each without any overlap between successive chunks, each chunk being an entry in our database. These chunks are then represented as sequences of features, each feature being made of the 12 first Mel Frequency Cepstral Coefficients (MFCC) [DM80] computed over a window of 20 ms length with an overlap of 10 ms. Cepstral coefficients encode the spectral enveloppe of the signal, which caracterizes sounds, over a short time period. Moreover, the Euclidian distance between two MFCC vectors measures the spectral distance between the two corresponding spectral enveloppes. In our experiments, each dimension of the features is normalized for the global distribution to have zero mean and unit variance.

We used 3 different samples that are repeated in the *France Info* stream but not present in the *FIP* stream and distorted them in order to get 7 different versions of each sample, leading to a set of 21 different queries. The chunks from the database can then be similar to the queries in the sense that they share similar parts, but are, in general, timeshifted. The first version, which is the original one, is 5 second long. We extracted from the stream a sequence of 7 seconds that contains the original version of the query and a subsequence of the latter that is 3 second long. We also altered the tempo of the original sequence to get a version that is 10% faster and another one that is 10% slower. Two sequences are MP3-encoded (while the *France Info* stream from which the query was extracted was in WAVE format) at different rates: 32kbps for the low-quality version and 128kbps for the high-quality one. Finally, we used a sample that is not present in the database in order to evaluate the ability of both approaches to give information about the absence of a sequence in the database.

Experiments consist in ranking elements from the database by decreasing similarity to a query. To do so, we estimate the similarity between the considered query and every elements of the database. Results are

		$C = 0.5$	$C = 1$	$C = 2$
$\varepsilon = 0.005$	$\varepsilon_2 = 0.005$	0.921	0.939	0.951
	$\varepsilon_2 = 0.05$	0.924	0.939	0.950
	$\varepsilon_2 = 0.5$	0.859	0.913	0.938
$\varepsilon = 0.05$	$\varepsilon_2 = 0.005$	0.924	0.939	0.951
	$\varepsilon_2 = 0.05$	0.924	0.939	0.949
	$\varepsilon_2 = 0.5$	0.863	0.913	0.936
$\varepsilon = 0.5$	$\varepsilon_2 = 0.005$	0.928	0.936	**0.954**
	$\varepsilon_2 = 0.05$	0.918	0.939	0.953
	$\varepsilon_2 = 0.5$	0.809	0.870	0.909

Tab. 2: MAP scores for all parameter sets of SVR on the small database.

evaluated using the mean average precision (MAP) of the resulting ranked list given a hand-made ground-truth.

5.2 Setting the SVR parameters

A first series of experiments aims at understanding and setting the parameters of the SVR. For that task we use only a part of the above described database (5 hours of *France Infos* recordings) and a single query without distortion so that the parameters are not completely database-specific. Here, we use the S_{min} similarity measure as it proved to be the most stable in our experiments.

MFCC features are designed to be compared using the Euclidean distance. For that reason, the only acceptable kernel for our task is the RBF one (see table 1). The parameters to adjust are therefore: the cost-function threshold ε, the termination criterion ε_2 and the trade-off between the number of support vectors and the quality of reconstruction C.

As there is no proof of independance between these parameters, we tested all possible combinations, with 3 possible values for each parameter. Results can be found in table 2, showing that the best is to set $\varepsilon = 0.5$, $\varepsilon_2 = 0.005$ and $C = 2$. Using this parameter set, we reached a MAP of 0.954 which means that most of the samples from the ground truth were ranked at a high position in the list. We observed that this parameter set led to a very precise representation of the samples, which was achieved thanks to a high number of support vectors in the model.

The optimal value for ε can be quite surprising: as our data has been scaled to follow a standard normal distribution, this value of 0.5 could be considered as being large, but one can see in table 2 that ε has little influence on the MAP. The small value for ε_2 indicates that going for numerous runs of optimization is useful, while setting $C = 2$ shows that deviations from the acceptable enveloppe (defined by $g \pm \varepsilon$) have to be strongly penalized.

This parameter set will be used in the next section to evaluate SVR performances.

5.3 Comparison between DTW and SVR

Using the optimal parameter set for SVR, we can compare the results of the two methods in terms of precision/recall graph, query-specific MAP and average MAP. To perform this comparison, we use the entire database and the whole set of queries.

Figure 2 presents precision/recall curves for both methods and both symetric similarity measures S_{avg} and S_{min}. This shows that using S_{avg} gives poorer results than the ones obtained with S_{min} for both methods. This is due to the fact that S_{avg} always takes into account information from both S_1 and S_2 while S_{min} only relies, for each similarity estimation, on the minimum of S_1 and S_2, which is supposed to be the best representation. Moreover, one can notice that the curves for DTW and SVR using S_{min} intersect several times, showing that each method can be useful, depending on the range of precision and recall one expects.

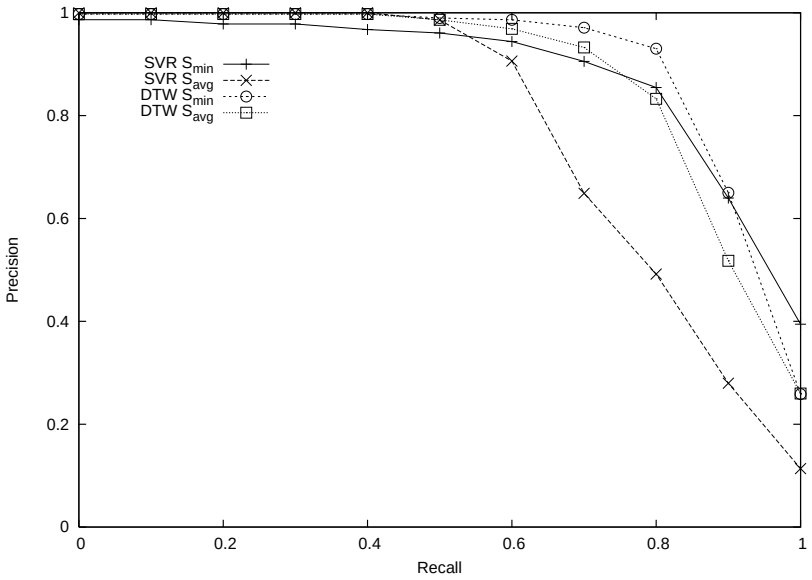

Fig. 2: Precision/recall curves for SVR and DTW using the entire database and the whole set of queries.

Table 3 presents MAP values for both methods, showing that SVR performs better than DTW, especially on original sequences. This proves the effectiveness of SVR for similarity estimation. Moreover, SVR also outperforms DTW on MP3 encoded samples. This is due to a better robustness to outliers. Indeed, encoding the chunks using MP3 alters the spectrum of the sequences, on which MFCC are based, generating outliers in the features. Performances are comparable for subsequences and oversequences, while, as predicted, DTW still performs better on tempo-modified sequences. This is due to the nature of DTW that is built to deal with such distortions. The good point here is that SVR is reasonably robust to stretching, as the MAP does not fall down for these chunks. We performed a non-parametric Wilcoxon test [Wil45] on the data. The results were that DTW is not significantly better than SVR on this dataset. Using S_2, which, in the case of SVR, corresponds to predicting the query using a model fitted on the elements of the database, performs significantly better than S_1, leading to a bias in the computation of S_{avg} and S_{min}.

Indeed, the audio descriptors we are using are very sensitive to the presence of speech in the audio signal (so sensitive that they are often used for speech/non-speech classification). Therefore, the descriptors tend to be located in very different parts of the multidimensional feature space, depending on whether they describe speech or not. The point here is that our database was truncated into pieces of 5 seconds each. Because of this arbitrary truncation, some of the chunks contain parts of the query (which is music) as well as speech. These sequences were evaluated by the human expert as occurences of the query. Then, given what we know about the differences between speech descriptors and non-speech ones, a model fitted on a music query (which is the case for S_2) is poor at predicting speech parts of the signal, and these chunks are then considered as negative examples, while they should not. This explanation is supported by table 4 that shows the comparative impact on S_1 and S_2 of truncating the false negative chunks so that they do not contain speech anymore.

We also tested the ability of both approaches to give information about whether a sequence is present or not in the database. To do so, we used a sequence that was not present in the database as a query q and computed, for both approaches, the following quantity

$$F(q) = \frac{\text{score}(\text{First}(q))}{\frac{1}{n_{\text{queries}}} \sum_{i=1}^{n_{\text{queries}}} \text{score}(\text{First}(q_i))} \quad , \tag{20}$$

where score(First(q)) is the similarity measure given by the considered approach for the best match in the

	DTW				SVR			
	S_1	S_2	S_{avg}	S_{min}	S_1	S_2	S_{avg}	S_{min}
original	0.751	0.904	0.877	0.903	0.608	0.948	0.822	0.938
subsequence	0.714	0.942	0.928	0.941	0.613	0.937	0.865	0.938
oversequence	0.881	0.729	0.844	0.885	0.692	0.922	0.821	0.898
slow version	0.796	0.880	0.878	0.904	0.383	0.925	0.690	0.722
fast version	0.720	0.913	0.874	0.913	0.489	0.933	0.695	0.828
MP3 32kbps	0.735	0.880	0.852	0.879	0.536	0.938	0.760	0.918
MP3 128kbps	0.757	0.901	0.877	0.907	0.597	0.955	0.825	0.946
Average	0.765	0.878	0.876	0.905	0.560	0.937	0.782	0.884

Tab. 3: MAP scores for SVR and DTW using the entire database for each kind of versions of the set.

Similarity measure	Original version	Truncated version
S_1	15.72	11.53 (-27%)
S_2	12.26	12.65 (+3%)

Tab. 4: Impact of the truncation on the similarity measurements. These results were obtained for SVR with optimal parameter set and the first query of the set.

list returned for query q and n_{queries} is the total number of queries we used in the previous experiments, that is 21 (7 versions of 3 different samples). Therefore, the higher $F(q)$, the better.

Results, presented in Table 5 show that, for each smilarity measure, SVR is better than DTW at discriminating between sequences from the database and the others.

6 Conclusion

This paper aimed at assessing the feasability of estimating similarity between sequences in a model space for later application to the indexing of large temporal databases. Performances of similarity estimation using SVR models were compared to the ones of usual DTW similarity measure, in both theoretic and experimental ways.

The study carried out above shows that using S_2 similarity measure, SVR outperforms DTW for almost all types of queries, exhibiting an interesting robustness to distortions such as tempo-modifications, MP3 encoding and subsequence extraction. Moreover, SVR is better than DTW at distinguishing between a sequence that is present in the database and another one that is not.

These encouraging results point out the fact that SVR could be used as part of a complete sequence indexing scheme. Before that, we will need to understand better the reasons why S_2 performs significantly better than S_1 while we expected them to give similar results. Then, a special effort should be done to turn these similarity measures into distances so that the process of querying can be fastened.

References

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[AI06] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468, October 2006.

[BGG99] P. Bouthemy, M. Gelgon, and F. Ganansia. A unified approach to shot change detection and cam-

	DTW				SVR			
	S_1	S_2	S_{avg}	S_{min}	S_1	S_2	S_{avg}	S_{min}
$F(q)$	5.948	9.220	7.772	8.083	16.581	20.443	20.575	14.309

Tab. 5: Comparison of $F(q)$ for both approaches.

era motion characterization. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7):1030–1044, October 1999.

[BMM03] E. Bruno and S. Marchand-Maillet. Prédiction temporelle de descripteurs visuels pour la mesure de similarité entre vidéos. In *Proceedings of the GRETSI'03*, France, September 2003.

[CN04] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 792–803, August 2004.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23th International Conference on Very Large Data Bases*, pages 426–435, Athens, Greece, August 1997. Morgan Kaufmann Publishers, Inc.

[DM80] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Audio, Speech, and Language Processing*, 28(4):357–366, 1980.

[DTS+08] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. In *Proceedings of the 34th International Conference on Very Large Data Bases*, 2008.

[Keo02] E. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 406–417, 2002.

[LÁJA09] H. Lejsek, F. H. Ásmundsson, B. Þ. Jónsson, and L. Amsaleg. Nv-tree: An efficient disk-based index for approximate search in very large high-dimensional collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):869–883, 2009.

[LTCJ+07] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. Video copy detection: a comparative study. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 371–378, New York, NY, USA, July 2007. ACM.

[Mer09] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

[MGB09] A. Muscariello, G. Gravier, and F. Bimbot. Variability tolerant audio motif discovery. January 2009. Submitted to the 15th International Multimedia Modeling Conference.

[NS06] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2006.

[SC78] H. Sakoe and S. Chiba. Dynamic programming optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[SS98] A.J. Smola and B. Schoelkopf. A tutorial on support vector regression, 1998.

[TAG07] R. Tavenard, L. Amsaleg, and G. Gravier. Machines à vecteurs supports pour la comparaison de séquences de descripteurs. In *Proceedings of the 12th CORESA*, pages 247–251, 2007.

[Vap95] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[VGS96] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, 1996.

[Wil45] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, December 1945.

[YJF98] B. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th International Conference on Data Engineering*, pages 201–208, February 1998.