Chapter 11

Time series classification with low-level attributes

In this chapter, we tackle the time series classification problem by the use of classical learning algorithms with low-level features. First, several algorithms are tested on non temporal attributes extracted from time series by simple sampling approaches. Then, a new method is investigated which reduces both bias and variance by classifying subsequences instead of the whole time series.

11.1 Introduction

In this chapter, we will experiment with methods that handle directly low-level features with classical learning algorithms. This is certainly the simplest and most direct approach since it does not demand to build new automatic learning algorithms or to extend existing ones. The motivations for such a study are twofold: first, show that temporal problems are often more difficult than non-temporal problems and thus justify the interest in them, second highlight the shortcomings of these approaches and thus narrow our future research in this domain in consequence.

We consider two ways to use a classical learning algorithm with numerical attributes extracted from time series:

- Simple sampling approaches: the simplest way to extract numerical attributes is certainly to sample the time series at equally spaced instants. Another method is proposed which divides the time series into non overlapping equal length segments and uses the average on each segment as candidate attributes.

- Segment and combine: this method starts by segmenting the original time series into *overlapping* subseries of a given length, then builds a model for the classification of such subseries (using low-level attributes and standard automatic learning methods), and then classifies a new scenario by aggregating the predictions given by this model for all its subseries. Although this approach uses low-level features as well, it takes advantage to some extent of the temporal structure of the problem; we will show that it improves results very significantly with respect to the first class of methods.

11.2 Simple sampling approaches

The simplest way to use classical automatic learning methods for temporal signal classification is to extract some simple scalar feature sets from signals and then consider them as

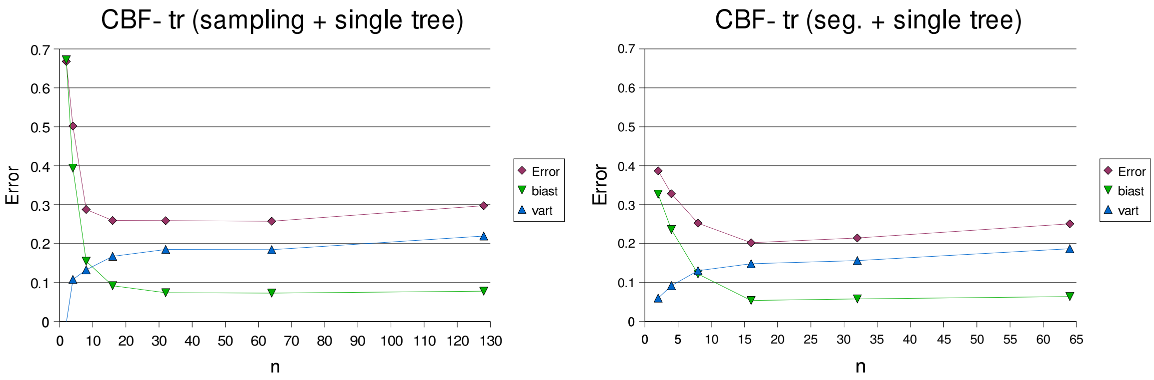

Figure 11.1: Sampling and segmentation on the CBF-tr problem with single trees

candidate input attributes for learning a classification model. In this section, experiments are carried out with two very simple sets of features:

- Sampled values of the time series at equally spaced instants. To handle time series of different durations, time instants are taken relative to the duration of the scenario. More precisely, if n is the number of time instants to take into account, each temporal attribute A gives rise to n scalar attributes given by: $A(o, t_f(o)\frac{i}{n-1})$, $i = 0, 1, \cdots, n-1$. In practice, to compute these values from the time series representation, the signals are interpolated in a linear fashion between two successive time points t_i.

- Segmentation of the time interval and averaging (also proposed in [Kad99]). The time axis of each scenario is divided into n segments of equal length and the average value of each temporal attribute along these segments are taken as attributes. In practice, a time series $(a_0, a_1, \ldots, a_{N-1})$ is described by the n following numbers:

$$\bar{a}_i = \frac{n}{N} \sum_{j=i\frac{N}{n}}^{(i+1)\frac{N}{n}-1} a_i, i = 0, 1, \ldots, n-1, \tag{11.1}$$

  when N is a multiple of n (otherwise the last segment receives the remaining time points).

These two approaches give $n \cdot m$ scalar attributes from m temporal ones. Note that the second approach exploits the temporal ordering of values to compute the average on each segment and so is doing some noise filtering.

In the first group of experiments, the effect of these two sets of low-level attributes in terms of bias and variance are computed on the CBF-tr and Two-pat problems. Then, further experiments are carried out on the remaining problems. Three classes of methods are tested: single decision trees, variance reduction techniques (boosting and extra-trees averaging), and the one-nearest neighbor rule.

11.2.1 Bias/variance study

Decision trees

We first build decision trees with these two feature sets. Figures 11.1 and 11.2 plot Tibshirani's bias and variance terms for increasing number n of non temporal attributes, respectively on the CBF-tr and Two-pat problems, left with sampling and right with

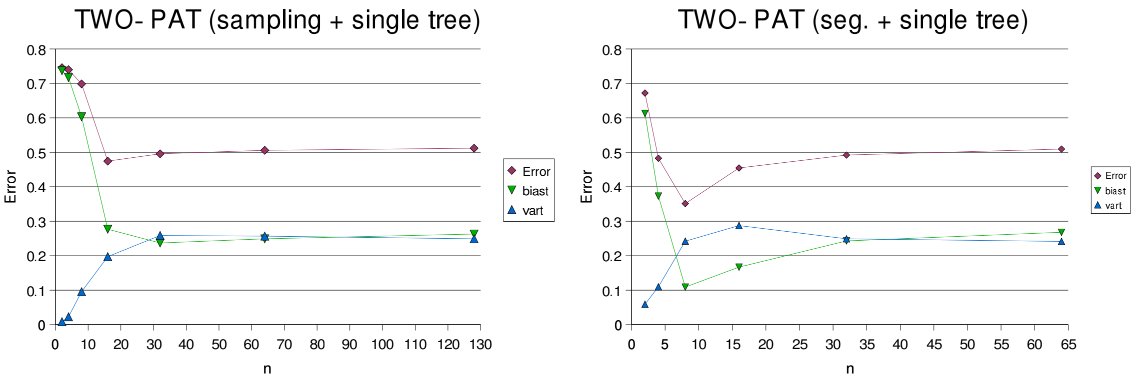

Figure 11.2: Sampling and segmentation on the Two-pat problem with single trees

Table 11.1: Both sampling strategies with single trees on CBF-tr and Two-pat

	Mean error	compl	bias_T	var_T	var_{KW}	$Err_{\hat{P}}$	$\text{bias}^2_{\hat{P}}$	$\text{var}_{\hat{P}}$
CBF-tr (single trees)								
sampling ($n = 64$)	0.2575	62	0.0730	0.1845	0.1735	0.1717	0.0560	0.1157
Segmentation ($n = 16$)	0.2023	71	0.0540	0.1483	0.1378	0.1349	0.0431	0.0918
Two-pat (single tree)								
Sampling ($n = 16$)	0.4744	139	0.2770	0.1974	0.2637	0.2372	0.1054	0.1318
Segmentation ($n = 8$)	0.3512	128	0.1090	0.2422	0.2193	0.1756	0.0660	0.1096

segmentation. In each problem, 50 learning samples of size 300 are drawn from a pool sample of 4000 case and bias and variance terms are estimated from a test sample of size 1000. Trees are fully grown.

One can observe from the figures that bias monotonically decreases with n and variance increases, except for the Two-pat problems and the attributes obtained by segmentation where bias increases for large values of n. Overall, the figures show that there exists an optimal value of n which realizes the best tradeoff between bias and variance. The bias/variance profile corresponding to this optimum is reported in Table 11.1 in each case (the value of n corresponding to this optimum is in the first column). The approach by segmentation is in each case significantly better than the approach by sampling, mostly from the point of view of bias. In both problems, the variance is high. But the bias is also high, especially on the Two-pat problem. This high bias is also present for the highest value of n, although, in this case, all the information available in the dataset is provided to the decision tree learner.

The high bias of decision trees is here due to the fact that on the one hand the relationship between the low-level attributes and classes can be represented accurately only by very complex decision trees, and on the other hand the rather small sample sizes used here (300) strongly bind the complexity of trees induced by learning. Note that the shift invariant properties of the two problems are certainly the first responsible of the complexity of the relationship between these low-level attributes and classes (we will further discuss this issue in Section 11.2.3).

Extra-trees and boosting

Since variance and bias are high, it seems to be a good idea to repeat the experiment with extra-tree averaging and boosting (which can reduce both bias and variance). Since segmentation is better than sampling, we report here only the results obtained with this

Table 11.2: Segmentation with extra-tree averaging and boosting on CBF-tr and Two-pat

	Mean error	compl	$bias_T$	var_T	var_{KW}	$Err_{\hat{P}}$	$bias^2_{\hat{P}}$	$var_{\hat{P}}$
CBF-tr (segmentation)								
Single tree ($n = 16$)	0.2023	71	0.0540	0.1483	0.1378	0.1349	0.0431	0.0918
Extra-tree avg. ($n = 32$)	0.0666	5766	0.0390	0.0276	0.0376	0.0576	0.0506	0.0070
Boosting ($n = 32$)	0.0863	857	0.0410	0.0453	0.0538	0.0710	0.0621	0.0089
Two-pat (segmentation)								
Single tree ($n = 8$)	0.3512	128	0.1090	0.2422	0.2193	0.1756	0.0660	0.1096
Extra-tree avg. ($n = 8$)	0.2357	8833	0.0810	0.1547	0.1526	0.0908	0.0777	0.0131
Boosting ($n = 8$)	0.2621	953	0.1070	0.1551	0.1645	0.1045	0.0943	0.0102

Table 11.3: Segmentation with 1-NN on CBF-tr and Two-pat

	Mean error	compl	$bias_T$	var_T	var_{KW}	$Err_{\hat{P}}$	$bias^2_{\hat{P}}$	$var_{\hat{P}}$
CBF-tr (segmentation)								
Extra-tree avg. ($n = 32$)	0.0666	5766	0.0390	0.0276	0.0376	0.0576	0.0506	0.0070
1-NN ($n = 32$)	0.0429	-	0.0170	0.0259	0.0277	0.0286	0.0101	0.0185
TWO-PAT (segmentation)								
Extra-tree avg. ($n = 8$)	0.2357	8833	0.0810	0.1547	0.1526	0.0908	0.0777	0.0131
1-NN ($n = 8$)	0.2222	-	0.0780	0.1442	0.1439	0.1111	0.0392	0.0720

approach. Table 11.2 shows the bias/variance profile corresponding to the optimum of error with respect to the number n of segments. In both cases, 25 trees are averaged.

With respect to a single decision tree, the improvement of error with the two techniques is important (especially on the CBF-tr problem) and is mostly due to a reduction of variance. Notice that, like in our experiments of Chapter 7, extra-trees yield also here lower error rates than boosting. However, even with this strong reduction of variance, bias and hence error remains quite high on both problems. We conclude that these techniques do not succeed in decreasing sufficiently the high bias of decision trees for these problems, because they are still limited by the small size of the induced trees in comparison to the number of low-level attributes which would be needed to classify these signals.

1-NN

One method which could possibly cope with a large number of relevant low-level attributes is the nearest neighbor technique. Indeed, provided that the vectors of low-level attributes corresponding to different classes of signals are farther away from each other than the vectors of the same class, this method is not directly sensitive to the dimension of the input space. So, it is interesting to try this method in this context. Table 11.3 compares 1-NN[1] with extra-tree averaging (which yields the best result so far). Again, the value of n reported in the table is the one which minimizes the error. The improvement with respect to extra-trees is quite important and is almost only due to a reduction of the bias. Still, the error rates obtained in the case of the Two-pat problem are very high (22%, to be compared with a residual error of 0%).

11.2.2 Further experiments

These two sets of attributes have been tried on the other four problems described in section 10.4 as inputs to the same four learning algorithms. Results in terms of error

[1]With respect to the algorithm described in Chapter 2, we do not normalize attribute values before computing the distance.

Table 11.4: Results with simple sampling methods (error rates in % ± the standard deviation)

Datasets	Single trees		Rand. tree avg.		Boosting		1-NN	
	Err.	n	Err.	n	Err.	n	Err.	n
CC - sampling	6.83 ± 2.29	60	1.67 ± 0.75	30	**1.50 ± 1.17**	60	1.83 ± 1.38	60
CC - segment.	3.50 ± 2.52	10	1.17 ± 1.50	10	1.50 ± 1.38	10	**0.50 ± 1.07**	10
CBF - sampling	7.33 ± 2.49	64	2.25 ± 1.34	32	2.17 ± 1.83	64	**1.16 ± 1.30**	128
CBF - segment.	2.67 ± 1.33	16	0.88 ± 1.26	32	0.67 ± 1.11	16	**0.50 ± 0.76**	16
JV - sampling	14.59	3	**2.97**	2	5.14	3	3.24	2
JV - segment.	12.97	3	4.05	2	6.22	5	**3.51**	2
Auslan - sampling	16.00 ± 8.31	8	**6.50 ± 5.50**	16	7.50 ± 4.03	10	11.00 ± 5.83	16
Auslan - segment.	21.50 ± 9.50	4	**5.50 ± 3.50**	8	8.50 ± 4.50	4	8.00 ± 6.00	4

rates are summarized in Table 11.4. Each line corresponds to a dataset and one sampling approach. Error rates were obtained by ten-fold cross-validation for the CBF, CC and Auslan problems (in which case the result is given in terms of the mean error rate and the standard deviation of this error) and by validation on an independent test set of size 370 for JV[2]. For each method, the error which is reported is the minimum of error with respect to n[3]. The value of n which realizes this error is also reported in the table for each method. The best result in every row is boldfaced.

There are several things to say about this experiment. Like for the first two problems, there exists an optimal value of n which corresponds to the best tradeoff between bias and variance and this value is different from one method to another on the same problem. For the CBF and CC problems, segmentation rather than simple sampling is highly beneficial while on the two other problems, the difference is less pronounced. This can be explained by the presence of an independent Gaussian noise on the CBF and CC problems which is somehow filtered out by segmentation.

Decision trees do not work well but extra-trees averaging and boosting are quite effective. 1-NN is still the best method on CC and CBF but it is beaten by extra-tree averaging on JV and Auslan. On CC, CBF, and JV, the best error rates we get with this simple approach are very good with respect to previously published results on these problems (i.e. with dedicated temporal approaches). On JV, the fact that the best results is obtained by sampling with only two values (measured at the beginning and the end of each scenario) shows that in this problem, the trajectory of the signal is not important for classification and hence improvement will be difficult to obtain with temporal approaches.

Notice that in our reporting of experiments, we have selected the best value of n on the basis of the results on the test sample. Note that, in practice, however, this value should be fixed, either by cross-validation from the learning sample or from an independent validation sample, and certainly not on the basis of the test sample.

11.2.3 Discussion

The results with the simple sampling approaches are not plainly satisfactory. In some problems, we get very good results but in other problems, like CBF-tr and Two-pat, which

[2]This division was suggested by the donors of this dataset [KTS99].

[3]n is chosen in $\{3, 5, 10, 30, 60\}$ on CC, in $\{8, 16, 32, 64, 128\}$ on CBF , in $\{2, 3, 5, 7\}$ on JV, and in $\{2, 4, 8, 10, 16, 24\}$ on Auslan.

present typical properties of the temporal domain by construction, the results are far from good even when applying boosting and extra-tree averaging. Also, the best method in terms of accuracy is in several problems the 1-NN method, which has several other well-known drawbacks (in particular the necessity to store the database for prediction and the large computation times required during prediction) which reduce the practical interest of this technique in the context of automatic signal classification tasks which often have some strong real-time constraints.

Another drawback of these approaches based on low-level attributes is that they do not provide interpretable models, which would be useful in order to help a human expert to understand the differences between the different classes of signals. Indeed, among the four studied methods, only decision trees would in principle be able to provide such interpretable results. However, in the present context, their interpretability is very questionable because of the choice of low-level attributes. Indeed, how to understand for example a rule like "if $a(o, 32) < 2.549$ and $a(o, 22) < 3.48$ then $c(o) =$ bell" which was induced from the CBF dataset? Although very simple and accurate, this rule does not make obvious the temporal increase of the signal peculiar to the bell class (see Figure 10.3).

The lack of accuracy and interpretability with decision trees comes from the fact that some very simple and common temporal features are not easily represented as a function of such attributes. For example, consider a set of sequences of n random numbers in $[0, 1]$, $\{a_1, a_2, ..., a_n\}$, and classify a sequence into the class c_1 if three consecutive numbers greater than 0.5 can be found in the sequence whatever the position. With a logical rule inducer (like decision trees) and using a_1, a_2,...,a_n as input attributes, a way to represent such a classification is the following rule: if ($a_1 > 0.5$ and $a_2 > 0.5$ and $a_3 > 0.5$) or ($a_2 > 0.5$ and $a_3 > 0.5$ and $a_4 > 0.5$) or ... or ($a_{n-2} > 0.5$ and $a_{n-1} > 0.5$ and $a_n > 0.5$) then return class c_1. Although the initial classification rule is very simple, the induced rule has to be very complex. This representation difficulty will result in a high variance of the resulting classifier and thus in low accuracy and poor interpretability. The use of variance reduction techniques like boosting or extra-trees will not be enough to restore the accuracy. In the next section, we propose a new variance reduction technique which is more adapted to temporal data.

11.3 Segment and combine approach

There are two problems in using low-level features with decision trees. When the number n of features is small, the variance is small but the bias is very high. On the other hand, when n is large, the variance is high but the bias is also high because it is difficult to represent scale and shift invariant properties with low-level attributes and further the tree complexity is limited by the learning sample size. So, all in all, the optimal value of n does not give good classifiers.

In this section, we propose another technique to control the bias/variance tradeoff which is based on subsignals classification. The idea is to segment a scenario into all its overlapping subsignals of a given length L, derive a model to classify these subsignals on the basis of low-level attributes and then combine the predictions given by this model for all the subsignals of a scenario to make a prediction. We call this approach "segment and combine". The algorithm is described in Section 11.3.1 and then validated on our test problems in Sections 11.3.2 and 11.3.3. In Section 11.3.4, we show that some interpretable results may be extracted when subsignals are classified by decision trees.

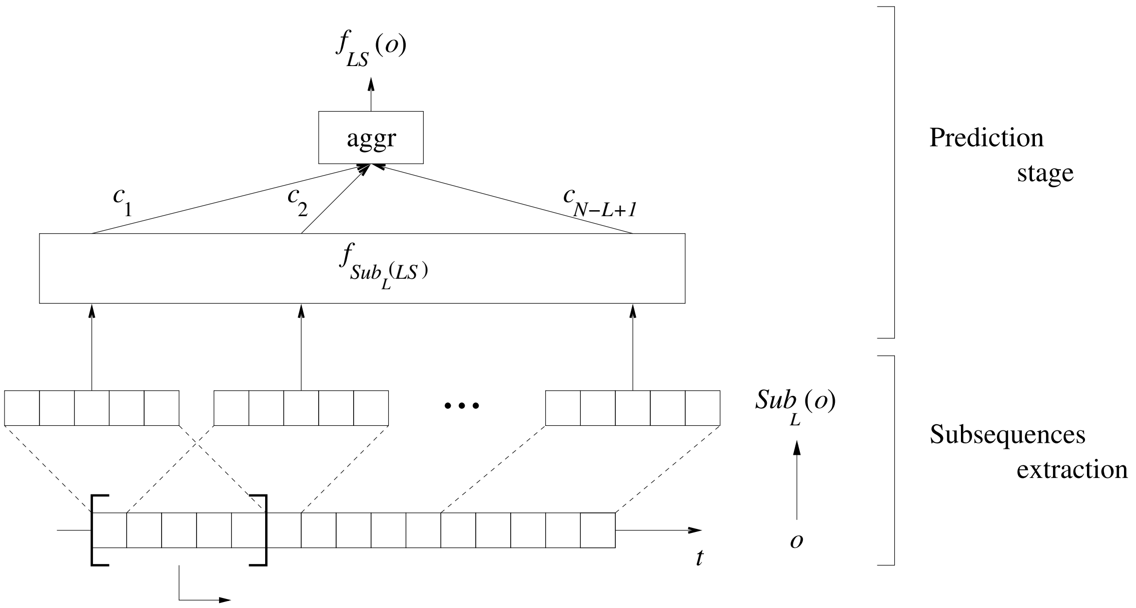

Figure 11.3: Illustration of the segment and combine approach for time series classification (with L=5)

11.3.1 Algorithm

Learning stage

Table 11.5 describes the algorithm which builds a model for subsequences from a learning sample. This algorithm explicitly assumes that temporal attributes are represented by time series with a constant discretization step $\triangle t$. A window of L time points is defined and moved along the time axis to extract all subsequences of L time points from every scenario of the learning sample (see Figure 11.3). Each subsequence receives the classification of its parent scenario and a learning algorithm is used to build a model of this relation. Any learning algorithm could be used in this goal. Input attributes used by this algorithm are all the $m \cdot L$ numbers which describe the subsequence in the time series representation.

Prediction stage

This model is then used to classify new scenarios as described in Table 11.6. The tested scenario is splitted into all its subsignals of length L and its classification is obtained by aggregating the predictions given for each of these subsignals (see Figure 11.3). Possible aggregation operators are majority vote among the predictions of subsignals or average of conditional class probability estimates assuming the learning algorithm provides such estimates. In our experiments with decision trees, we will only use averages of probability estimates.

With respect to simple sampling approaches, subsequences are here always represented by low-level features at the highest resolution (corresponding to the time step of the time series representation). The variance is regulated by modifying the length of subsequences. Indeed, when L decreases, the number of features which is proportional to L decreases and the number of subsequence forming the new learning sample increases. Both these effects should reduce the variance. On the other hand, bias should increase when the length of the subsequence decreases. In the limit case where L is equal to the number of time points,

Table 11.5: learning the subsignals classification model

Given a learning set ls, a length of subsignals L and a learning algorithm:

- Compute the set of all subsequences of size L appearing in each scenario $o \in ls$:

$$Sub_L(o) = \{(\underline{A}(o,t_i), \underline{A}(o,t_{i+1}), ..., \underline{A}(o,t_{i+L-1}))|i = 0,1,\ldots,n(o)-L\}, \qquad (11.2)$$

  and the set of all subsequences of size L appearing in the learning set:

$$Sub_L(ls) = \bigcup_{o \in ls} Sub_L(o) \qquad (11.3)$$

  Each object generates $n(o)-L+1$ subsequences. Each subsequence is fully described by $m \cdot L$ different numbers (where m is the number of temporal attributes)

- Give to each subsequence of $Sub_L(ls)$ the classification of the source scenario.
- Use the learning algorithm to build a model $f_{Sub_L(ls)}$ which reproduces this latter classification using the $m \cdot L$ numbers as input attributes.

Table 11.6: Test procedure with the segment and combine approach

Given a scenario o and a classification model $f_{sub_L(ls)}$ for subsignals of length L

- Compute the set $Sub_L(o)$ of all subsequences of size L appearing in the scenario $o \in ls$;
- Classify each subsignals s in $sub_L(o)$ with the model and aggregate these predictions:

$$f_{ls}(o) = \mathrm{aggr}_{s \in Sub_L(o)}\{f_{Sub_L(ls)}(s)\} \qquad (11.4)$$

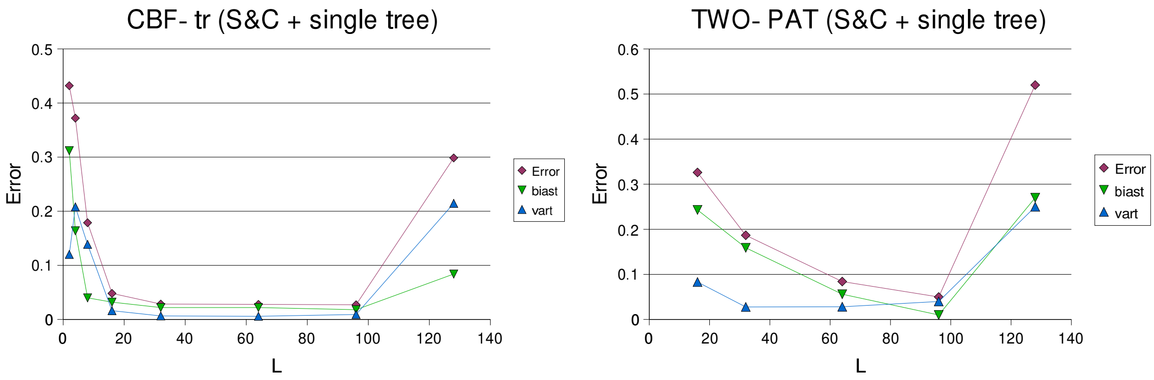

Figure 11.4: bias/variance evolution with the subsequence length L left on the CBF-tr problem, right on the Two-pat problem

only one subsequence is extracted from each scenario and this method is strictly equivalent to the sampling approach with the maximum number of time points. Notice that, although this method yields a computational overhead with respect to standard trees, its prediction stage remains intrinsically very efficient.

11.3.2 Bias/variance study

We first carried out experiments on the CBF-tr and Two-pat problems in computing bias and variance. As subsequence classification algorithm, we use fully grown trees. Since the set of all subsequences, $Sub_L(ls)$, may be very large (especially when L is small), its size is limited to 5000 in our experiments. So, If the number of subsequences exceeds 5000 cases, then 5000 samples are randomly drawn from the whole set to build the model.

Figure 11.4 plots Tibshirani's bias and variance terms with respect to the subsequence length L, left on the CBF-tr problem, right on the Two-pat problem. As expected, the bias decreases when the subsequence length increases. However, even for large values of L (close to the maximum number of time points), it is much lower than the bias which is obtained when L is maximum (i.e. exactly matches the number of time points).

The evolution of the bias may be explained by an example. Let us suppose that a scenario is characterized by the pattern at the top of Figure 11.5 which actually may appear at every position in the signal. Figure 11.5 shows three smaller subsequences extracted from the initial scenario. These subsequences correspond to three different positions of the pattern. Segment and combine is thus a way to augment the learning sample with shifted versions of the initial signals. This increase of the learning sample allows to build more complex decision trees which are more tolerant to the shifting of patterns and hence this translates to a reduction of the bias. When L is small, subsequences do not contain enough information about the patterns and the bias is more important and when L is very large, subsequences are very close to the initial scenario and hence the bias increases as well. Note that on the CBF-tr problem, the minimum of the bias appears earlier than on the Two-pat problems. Indeed, from our knowledge of the problem, it is clear that the CBF-tr may be resolved by looking only to small portions of the scenario. However, for the Two-pat problem, a larger portion of the signal must be observed (at least some parts of the two patterns must be seen to classify a scenario).

On the other hand, the variance decreases when L decreases but at some point it also starts increasing. Indeed, for small values of L, it is difficult to separate subsequences of different classes and the model gives class probability estimates which are close to the

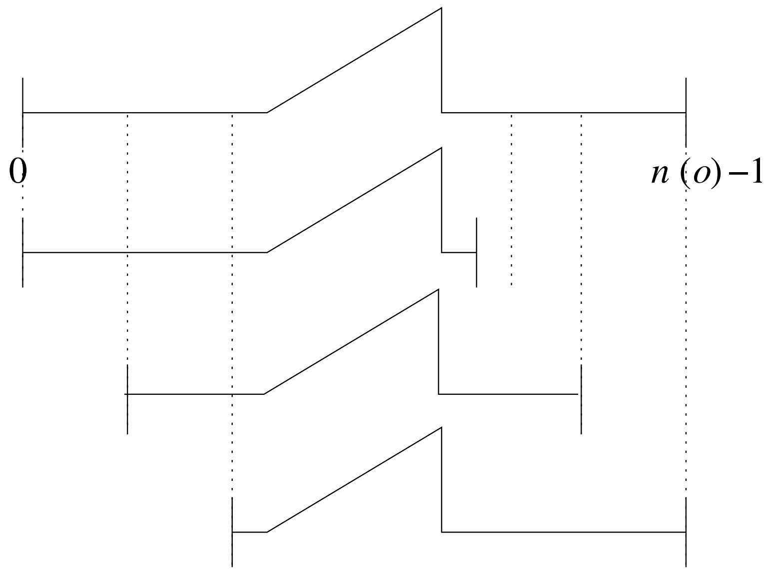

Figure 11.5: A signal and three subsequences extracted from it.

Table 11.7: Best results with the S&C approach on CBF-tr and Two-pat

	Mean error	compl	bias_T	var_T	var_{KW}	$Err_{\hat{P}}$	$\text{bias}^2_{\hat{P}}$	$\text{var}_{\hat{P}}$
CBF-tr								
Extra-tree avg. ($n = 32$)	0.0666	5766	0.0390	0.0276	0.0376	0.0576	0.0506	0.0070
1-NN ($n = 32$)	0.0429	-	0.0170	0.0259	0.0277	0.0286	0.0101	0.0185
S&C ($L = 96$)	0.0272	622	0.0180	0.0092	0.0135	0.0280	0.0244	0.0036
Two-pat								
Extra-tree avg. ($n = 8$)	0.2357	8833	0.0810	0.1547	0.1526	0.0908	0.0777	0.0131
1-NN ($n = 8$)	0.2222	-	0.0780	0.1442	0.1439	0.1111	0.0392	0.0720
S&C ($L = 96$)	0.0499	1015	0.0100	0.0399	0.0390	0.0400	0.0320	0.0080

decision frontier. So, the final decision for a tested scenario may easily goes from one class to another according to the particular learning set choice and this yields a high classification variance.

All in all, there exists an optimal value of L which realizes the best bias/variance trade-off. The bias/variance profile corresponding to this value on both problems is reported in Table 11.7. For comparison, the best errors obtained by extra-tree averaging and 1-NN are reported in the same table. The improvement obtained with segment and combine is very impressive, especially on the Two-pat problems. It comes from the reduction of both bias and variance. Another advantage of this technique is that the decision tree that classifies subsequences is very small with respect to the size of the set of extra-trees. With respect to 1-NN, a scenario is also classified in much less time and this classification needs only access to a rather compact set of trees.

11.3.3 Further experiments

We carried out further experiments on the other problems, with fully grown trees and with extra-tree averaging. The same limitation to 5000 of the number of subsequences for learning was applied. Table 11.8 reports the best result according to L[4] for each problem and method.

[4] L is chosen in $\{3, 5, 10, 30, 45, 60\}$ on CC, in $\{8, 16, 32, 64, 96, 128\}$ on CBF , in $\{2, 3, 5, 7\}$ on JV, and in $\{2, 4, 8, 10, 16, 24\}$ on Auslan.

Table 11.8: Results with S&C on small datasets (error rates in % ± the standard deviation)

Datasets	S&C with single trees		S&C with Extra-tree avg.	
	Err.	L	Err.	L
CC	0.33 ± 0.67	30	0.66 ± 0.82	45
CBF	1.50 ± 1.46	96	1.25 ± 1.37	96
JV	7.84	5	2.16	2
Auslan	3.00 ± 4.00	16	1.00 ± 2.00	24

Like for the CBF-tr and Two-pat problems, the value of L which realizes the best compromise on each problem is close to the maximum length of the scenarios. Thus, from our experiments it looks as if it was sufficient to take the length L a little bit (say 30%) smaller than the maximal length of the scenarios to improve significantly the results. Whether this remains true in a broader context remains to be investigated.

With respect to the direct use of low-level attributes with single decision trees, the improvement with segment and combine is very important on all problems. On CC and Auslan, segment and combine with single trees also improves the accuracy with respect to extra-trees and 1-NN with sampling approaches. On CC, the improvement is not very important since the error was already close to the minimum with 1-NN. On Auslan, however, the improvement is important (from 5.5% to 3% error rate). On the other hand, on CBF and JV, extra-trees or 1-NN with sampling give better results than segment and combine with decision trees. Nevertheless, a more powerful learning algorithm can be used in combination with segment and combine to further improve the results. To see if something can be gained this way, we ran segment and combine with extra-tree averaging (of 25 trees) in the same conditions on all problems. On CBF and CC where we already get almost perfect score with simple sampling or segment and combine with single trees, the use of extra-trees does affect accuracy. On JV, with subsequences of length 2, the error rate becomes 2.16%. On Auslan, the error rate was reduced to $1 \pm 2\%$ with a subsequence length of 24. This shows that further improvements are still possible with this algorithm.

11.3.4 Interpretability

Some authors [Oat99, DLM$^+$98] have proposed methods to extract a small subset of relevant subsequences from a set of time series (or a long one) which then can be used for different purposes, e.g. classification or rule discovery (i.e. finding small groups of subsequences which frequently appear together in a time series). As the number of such subsequences can be very huge, these authors have proposed to first use clustering to find relevant prototypes among subsequences. They fix a subsequence length, L, and use a distance measure (e.g. Euclidian or dynamic time warping, which is a distance measure between time series invariant to change in time scale) and a clustering algorithm (e.g. k-means) to cluster all time series subsequences of length L. The relevance of these prototypes for a given purpose is then assessed a posteriori. If we want to find useful subsequences for classification, there are two drawbacks with this approach: we do not know the right number of clusters and we are not ensured to find discriminative patterns with non-supervised clustering.

Actually, a side result of segment and combine with decision trees is that it provides a clustering of subsequences. Indeed, to some extend, a decision tree gives a partition of its input space into regions where the classification is constant. In the context of

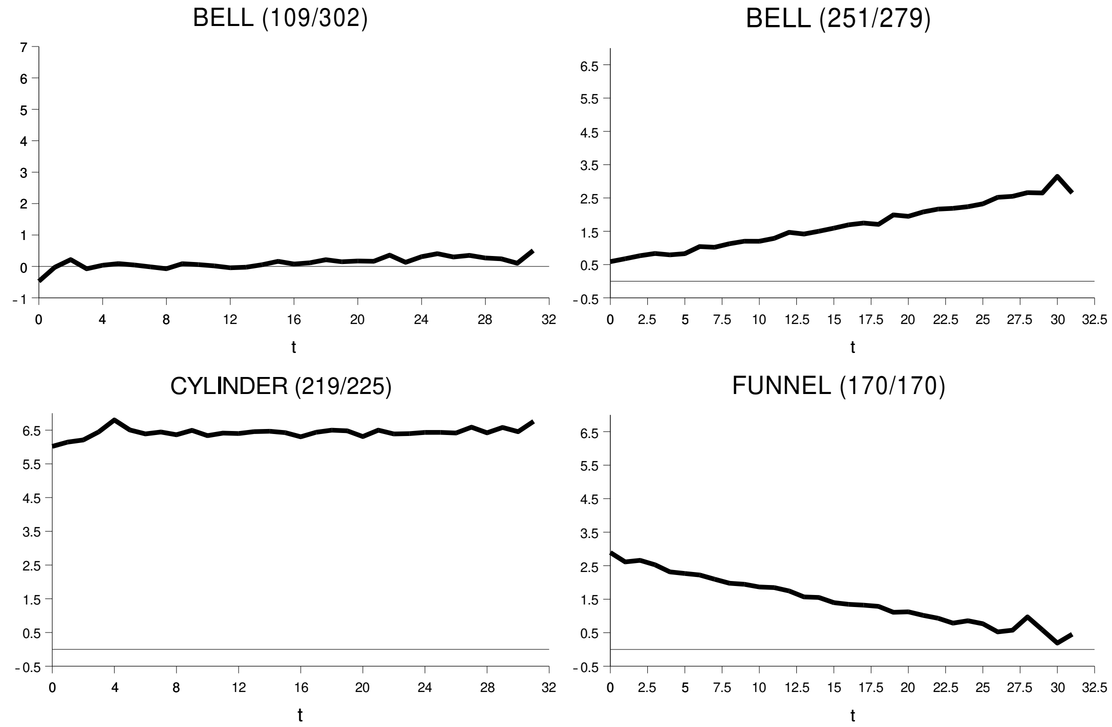

Figure 11.6: Four patterns extracted from a decision tree on the CBF problem

segment and combine, objects which are classified by the decision trees are subsequences of a given length L. So, each branch of the tree tries to characterize a set of subsequences which are representative of one class. Therefore, at no additional cost, the decision tree classifier gives clusters of subsequences which are as pure as possible in terms of the output classification variable. To visually inspect these clusters, a prototype can be computed by averaging the subsequences which reach a terminal node (attributes per attributes and time points per time points).

To illustrate this idea, we have built a tree for classifying subsequences of length 32 on the CBF problem. This tree was fully grown from 2500 subsequences extracted from a learning sample of size 500 and then pruned from 2500 other subsequences. Figure 11.6 shows the four prototypes corresponding to the four terminal nodes which contain the largest number of subsequences. The majority class in this node is given on the top of the prototype, as well as the number of subsequences of this class and the total number of subsequences in the node. When the node is not pure, only elements of the majority class have been used to compute the prototype. The first prototype corresponds to a "garbage" node which contains actually about the same number of elements each one of the tree classes; the bell class happens to be the majority class in this node, but the pattern corresponding to the subsequences arriving at this node is characteristic of no particular class. The interpretation of the other three terminal nodes is straightforward. Each one of them corresponds to one of the three classes and the degree of purity is very high. Also, the prototypes extracted from the corresponding subsequences exactly correspond to the patterns which are used to define the different classes in this problem (see Section 10.4 and Figure 10.3).

Although prototypes in Figure 11.6 are interpretable on this problem, this technique nevertheless assumes that there exist patterns typical of one class. For problems like Two-pat where several patterns have to be combined to classify scenarios, this technique does not give interpretable results.

11.3.5 Related work

The algorithm proposed here can be seen as a combination of jitter (see Section 4.2.2) and the dual perturb and combine algorithm proposed in the first part of this thesis. Indeed, every subsequence appearing in a time series may be seen as a perturbation of the initial time series (at least for large value of the length L). Like in jitter, the learning sample size is increased by the introduction of new perturbed learning cases. Like in dual perturb and combine, the prediction of a new case is obtained by aggregating the predictions given for several perturbations of it. Eventually, like for these two methods, the noise level (here, the length of the subsequence) regulates some bias/variance compromise.

At least two recent papers propose similar algorithms both of them for image classification in the context of handwriting recognition:

- in [FH00a], a new scheme for character classification is proposed which consists in moving a small rectangular window over a pixel-array representing the character image, classify every such part of the array and then combine these predictions to give a final classification of the image. Among other things, they analyze the effect of the window size on accuracy. Although, in this work the window classification is apparently not using decision trees, the overall idea is similar in spirit to our "segment and combine" method transposed to the bidimensional signal classification case;

- in [DKN01], in order to make a classification model independent of some transformations of the input, the learning sample is augmented with new learning cases which are obtained from existing ones by applying such transformations. When testing, a "virtual" test sample is constructed by applying these transformations to the test case and the predictions given by a unique model for these new cases are aggregated to give the final prediction. In their application to handwritten digits classification, transformations are ± 1 pixel shifts of images in x and y directions. Our algorithm can be seen as an instance of their method in the context of time series prediction, where the transformation is the shifting of patterns.

In [Die00a], a divide and conquer manifesto is discussed as the proper way to handle complex data of which time series are a particular case. For classification problems, this manifesto suggests to divide the complex data into smaller subparts which individual classifications are merged to provide a solution to the initial classification problem. The paper focuses on examples where the merge step is carried out by models that exploit the relative positions of subparts in the complex object (such as hidden Markov models for time series). Nevertheless, our "segment and combine" method can be considered as a particular instance of this divide and conquer framework where time series are divided into subsequences which classifications are aggregated by a simple (non temporal) averaging of conditional class probability estimates.

11.3.6 Discussion

On our test problems, the segment and combine approach for time series classification gives very good results. It works by reducing both bias and variance. Furthermore, the technique is rather general since it makes few explicit hypotheses about the problem. Its complexity depends on the automatic learning method which is used to classify subsequences and the number of subsequences which are used to learn this model.

All possibilities of this method have not yet been explored. For example, other learning algorithms may be used instead of decision trees or extra-tree averaging. Whatever the basic learning algorithm used in the approach, this algorithm may take advantage of the generally large number of subsequences which are available after the segmentation of time series. For example, some part of these subsequences may be used as a validation sample to estimate some meta-parameters of the base learner (the method used to derive models from subsequences). In the context of decision trees, this validation sample may be used for instance for post-pruning.

Given our analysis on bias and variance, it seems that this method mainly works by making the classification more robust to shift invariant properties. Following the work in [DKN01], other methods could be imagined to perturb scenarios that would make the method robust to other temporal peculiarities. For example, random non linear time axis transformations could be applied in order to make the classification model invariant to changes in time scale. Notice that in many automatic learning problems (not necessarily temporal ones) such invariance properties are known to hold from a priori information about the problem and the approach just discussed provides a generic way to incorporate such prior knowledge in the learning process. This is certainly a very interesting field for further research.

Another research direction concerns the study of alternative methods to combine the predictions of subsequences, for example by taking into account their temporal ordering. Actually, the segment and combine method is able to classify scenarios of any length greater than L. It could be applied in an on-line scheme by feeding the segment classifier with a window of size L moving forward as time progresses, and thus provide a classification that would evolve with time. Such a time varying output could then be used to build an on-line model which would classify a scenario as soon as the probability of one class goes above a given threshold. We can even use it to solve the early detection problem discussed in the first chapter. To this end, the model could be forced to favor the classification of early subsequences by assigning more important weights to them when learning the subsequence classifier.

11.4 Conclusion

In this chapter, we have explored different ways to exploit low-level features to solve the time series classification problem. First, simple sampling methods which extract scalar attributes from temporal ones have been used in combination with several algorithms. Second, we have proposed a "segment and combine" approach and have validated and analyzed its performance on the six test problems at our disposal. This method consists in building a model for subsequences of a time series of a given length, and then aggregating the predictions of each subsequence to classify time series. The appropriate length of the subsequences can be automatically adjusted by cross-validation so as to optimize the resulting bias/variance tradeoff.

On several problems, the results obtained by simple sampling methods are already very good. On the problems where this approach is suboptimal, the segment and combine algorithm improves significantly the results. Although our experiments are limited to six problems, they show that it is possible to get good results by using low-level attributes, even when problems present typical properties of the temporal domain.

The use of low-level attributes with classical learning algorithms however suffers from one important drawback: induced models are often not interpretable. Even if it is possible to interpret in some way the results given by the segment and combine algorithm with decision trees, this possibility remains limited and has still to be investigated in problems where classes are characterized by the occurrence of more than one single pattern.

Since the bias/variance problem seems to be solved in a satisfactory way at this stage (at least for our test problems), while the problem of interpretability remains to some extent, our goal in the next chapter is to design an algorithm which would help us to extract interpretable knowledge from time series data. Anticipating on this, let us say that the adopted method will consist in extending the vocabulary of candidate tests used by decision tree induction so as to enable such tests to detect patterns in time series. Nevertheless, given the unavoidable compromise which exists between interpretability and accuracy in automatic learning, we do not have the ambition of improving, with interpretable methods, the very good results obtained with the segment and combine approach.