УДК 681.3: 519.68

TIME SERIES CLASSIFICATION THROUGH HETEROGENEOUS FEATURE SELECTION

© Martyanov V.Ju., Eruhimov V.L.

Intel corporation
Nizhnhy Novgorod, Russia

e-mail: *Vladimir.Martyanov@intel.com*

Abstract. The paper investigates a generic method of time series classification. A heterogeneous set of features is extracted from each signal, including statistical moments, wavelets, Chebyshev polynomials, PCA, and DTW-based features. An ensemble of boosted trees is learned on a subset of this set of features. Particle filtering is used to choose a good feature subset and parameters of the learning engine based on cross-validation error.

Introduction

The problem of time series classification (TSC) has experienced a burst in the number of publications during the last decade. Various domains such as computer vision, medicine, biology, manufacturing and many others generate an enormous amount of signal data that can be used to segment or predict important events such as human actions or diseases or manufacturing tool malfunctions. There is a strong need for a generic TSC engine that, having small or no knowledge about the domain, can learn to classify signals from labeled samples. About a decade ago, [4] introduced a measure called Dynamic Time Warping (DTW) that is based on matching two signals with dynamic programming. Together with one-nearest-neighbor (1NN) it remains a competitive method for time series classification [22]. A large group of papers is devoted to extracting generic features from signals and transforming a TSC problem into a classical machine learning problem of predicting signal class from a given feature set. A list of features includes SVD (Singular Value Decomposition) features [19], DFT (Discrete Fourier Transform) [2], coefficients of the decomposition into Chebyshev Polynomials [8], DWT (Discrete Wavelet Transform) [9, 21], PLA (Piecewise Linear Approximation) [17], ARMA (AutoRegression Moving Average) coefficients [10], various symbolic representations [15, 13, 20]. An excellent review of the TSC techniques is given in [16].

Each of the methods has its own faults. Euclidean/DTW based methods suffer from the curse of dimensionality – 1-NN is known to perform poorly on high-dimensional problems (i.e. long signals) [14]. Generic feature extraction methods balance between low-dimensional signal representations that have less information and thus often possess lower predictive capabilities, and high dimensional representations that are hard to learn from (although here we are not restricted to 1-NN, as in the case of Euclidean/DTW distance). Also, each generic feature is salient for only a subset of TSC problems – each TSC problem has its own classification-optimal set of features.

Recent advances in feature selection methods [5, 1] enable us to dicuss the following generic approach to a TSC problem. We pick several generic feature sets – statistical moments, wavelets, Chebyshev coefficients, PCA coefficients, and the original values of signal. Then we run Gradient Boosting Trees (GBT) [11, 12] with imbedded feature weighting scheme. We show that although the extracted features supplement each other and removal of one feature types may result in a considerably large increase of test error for some TSC tasks, due to limited number of training samples, learning the model on an appropriate feature subset (suited to this particular TSC dataset) can be advantageous, as compared to learning it on all available features. We show that Particle Filtering (PF) can be successfully used to select a subset of features and simultaneously optimize model parameters. The results are obtained on publicly available UCR datasets [18].

1. Machine Learning on Massive Sets of Features

This section provides a description of Gradient Boosting Trees that we used for supervised learning, and a feature selection algorithm that we used to reduce the dimensionality of the learning problem.

1.1. **Gradient Boosting Trees.**

Gradient Boosting Trees (GBT) [11], [12] has been proven to be among the most accurate and versatile state-of-the-art learning machines. GBT is an iteratively learned serial ensemble where every new tree is fitted to the generalized residuals of the current ensemble. GBT builds shallow trees using all variables (on a subsample of the training data), and hence, it can handle large datasets with a moderate number of inputs. A modification of GBT [5] suggests a different ensemble learning strategy so that processing of very high dimensional datasets is feasible with almost no loss in prediction accuracy.

Our implementation of GBT is very close to [12] with feature weighting [5] on top of it. Each tree was trained on a randomly chosen 60% portion of the training dataset, the probability threshold was equal to 0.5.

1.2. **GBT with dynamic feature weighting.**

The main idea of this approach is to apply the variable-sampling technique used by Random Forest [6] for high-dimensional problems. We sample with replacement a small subset S from a set M of all input variables but keep only distinct elements. We sample S for each split in each tree independently and a standard greedy optimization is used to select a feature from S to split on. However, as shown in the [5], the uniform sampling used in the Random Forest could cause a significant performance degradation for sequentially boosted trees. We sample with probabilities that depend on feature importance and both are updated as a new tree is added to the ensemble. The sampling probability for a variable x_i at iteration l (where each iteration corresponds to learning a new tree so that at iteration l we already have an ensemble of l trees) is proportional to the corresponding weight

$$p(x_i, l) = w(x_i, l) / \sum_j w(x_j, l) \tag{1}$$

Weights have two components

$$w(x_i, l) = I(x_i, l) + SVI(x_i, l), \tag{2}$$

where $SVI(x_i, l) = \sum_{j=1}^{l} VI(x_i, j)$, $VI(x_i, j)$ is the influence of the i-th variable in the j_{th} tree:

$$VI(x_i, T) = \sum_{t \in T} \Delta I(x_i, t) \tag{3}$$

where $\Delta I(x_i, t) = I(t) - p_L I(t_L) - p_R I(t_R)$ is the decrease in impurity due to an actual (or potential) split on variable x_i at a node t of the tree T [7]. $I(x_i, l)$ is a contribution of the initial influence $I(x_i, 0)$ for the i_{th} variable at the l_{th} iteration. We used exponentially decreasing initial influences

$$I(x_i, l) = I(x_i, 0) \cdot (1 - |S| \, / \, |M|)^{\alpha l}\,, \tag{4}$$

where α is an adjustable parameter controlling how fast initial weights decrease (empirically chosen in range 0.5-2, equal to 1 throughout this paper). Here, $I(x_i, 0)$ represents prior knowledge about the variable influences, and this governs the sampling weights for a number of initial iterations. In this paper we used $I(x_i, 0)$ equal to the sum of variances of log-odds for response classes.

It is obvious that $I(x_i, l)$ decreases and $SVI(x_i, l)$ grows with the number of iterations l. Therefore, for sufficiently low initial influences, the learned variable importance will dominate the sampling weights after a number of iterations. Sampling with replacement (versus without) reduces the computational time up to 5 times [5]. However, it poses additional challenges related to potential "overweighting"effect for a small group of influential variables preventing other relevant variables from entering the model. This effect could be controlled by a gradual transition from initial to learned importance based upon the weights.

Here is the formal description of the dynamic feature weighting algorithm.

Gradient boosting trees with dynamic feature selection

1. Set $I(x_i, 0)$ for $i = 1, \ldots, n = |M|$ to the initial response deviation. Initialize $w(x_i, 0) = I(x_i, 0)$. Set current residuals (responses) to output variable values. Set $SVI(x_i, 0) = 0$.
2. Fit the next GBT tree to the current residuals using $p(x_i, l) = w(x_i, l) / \sum_j w(x_j, l)$ as the selection weights. At each tree node, a small fixed number $n_0 << n$ of variables is selected with replacement using selection probabilities $p(x_i, l)$ and the best split is searched only amongst this subset. l is the current iteration number.
3. Calculate variable importance $VI(x_i, l)$ on the i-th variable as in (3).
4. Calculate $SVI(x_i, l+1) = SVI(x_i, l) + VI(x_i, l)$. Update variable weights as $w(x_i, l+1) = I(x_i, 0) \cdot (1 - S/M)^{\alpha \cdot (l+1)} + SVI(x_i, l+1)$.
5. Update residuals with the difference between the predicted values and the old residuals.
6. Return to step 2 if the maximum iteration number is not exceeded ($l < l_{max}$).

1 Raw values,
2 Wavelet coefficients,
3 Chebyshev coefficients,
4 PCA coefficients,
5 Statistical moments,
6 DTW distances to base signals,
7 DTW+1NN predicted class.

2. Selecting the optimal feature classes

We extracted several types of features from each time series:

First we want to make sure that each of the feature classes helps to improve classification accuracy on at least one dataset. Here we are assessing the quality of only large feature classes – raw, wavelets, chebyshev and PCA – that, being irrelevant to the response, can decrease classification accuracy. In order to evaluate the contribution of each of the feature classes, we compare the performance of two models: the model learned on all features F and the model learned on all features except for a specific feature class F_X. Each model is learned 10 times with different subsampling sets.

The results are summarized in Table 1. The columns corresponding to feature types show the median of the ratio of test errors ϵ_X to the test error on the full feature set ϵ. We run a t-test for each set of experiments checking if $\epsilon_X/\epsilon > 1$. The values in bold correspond to p-values less than 0.05 indicating that the given type of features is important for prediction on the given dataset. Table 1, columns 2–6, shows that all feature types except for PCA provide a statistically significant change in test errors. Although it is hard to explain the influence of a specific feature type on the response through a wide variety of UCR datasets, we could speculate that PCA features are less important because most of UCR datasets do not have enough samples to robustly estimate more than few PCA components.

We have found that for each feature class X there is at least one UCR dataset where the removal of feature set $F_X$has a negative impact on test error (not statistically significant for PCA). However we simultaneously get a decrease of test error on other datasets. Although the GBT model with dynamic feature weighting can handle massive sets of features, the number of training samples is usually limited, and filtering out irrelevant feature types prior to model construction can increase accuracy. Exhaustive search of the subset of feature classes that minimizes CV errors is too expensive. Also our experience of working with GBT indicates that sometimes the choice of GBT parameters N and ν is crucial. We solve both optimization problems with one algorithm based on particle filtering with simulated annealing.

3. Particle Filtering

The objective of the Particle Filtering (PF) algorithm is to optimize cross-validation error as a function of GBT model complexity N, shrinkage ν and feature subsets $\hat{F} \subset F$. Since iterating through all possible feature classes is infeasible we limit our search to classes of features so that $\hat{F}$ can be represented as $\hat{F} = F_{X_1} \cup F_{X_2} \cup .. \cup F_{X_k}$, where each F_{X_i}

Table 1. Medians of test errors on various feature sets normalized by the test error ϵ on F. Columns 2–6: for each feature type X we plot test error ϵ_X on feature set $F \backslash F_X$. "Random" column: test error ϵ_{RND} on random set of features. "1NN" column: the test error ϵ_{DTW} of DTW+1NN method. Boxes where t-test indicates that $\epsilon_X > \epsilon$ (columns 2–6) or $\epsilon_{RND,DTW}/\epsilon > 1$ (last two columns) with p-values greater than 0.05 are in bold.

Dataset	Raw	Wavelets	Chebyshev	PCA
ECG	0.789	**1.508**	0.829	0.950
Yoga	0.999	**1.025**	0.999	0.996
Two_patterns	1.000	2.750	1.000	1.000
wafer	0.925	**1.487**	**1.359**	1.093
Synthetic_Control	1.000	1.000	1.000	1.000
Swedish_Leaf	1.013	0.994	**1.066**	1.000
OSU_Leaf	0.926	1.000	**1.137**	0.978
Face(all)	1.017	1.008	**1.186**	1.004
Gun_Point	0.888	0.854	1.000	0.894
CBF	0.564	0.820	0.822	1.079
Trace	0.000	0.000	0.000	0.000
Face_Four	0.774	**1.524**	0.762	1.083
Lighting2	**1.192**	**1.069**	0.969	1.000
Lighting7	**1.323**	1.000	0.938	0.967
Olive_Oil	0.600	1.000	1.000	1.000
Coffee	0.000	**Inf**	0.000	0.000
Fish	1.000	1.063	1.000	0.920
Beef	0.837	**1.300**	0.905	0.900
Adiac	1.010	**1.034**	1.007	0.993

represents a class of features such as raw, wavelets etc. Each feature class is removed or added to a feature set with a fixed probability on a resampling phase. Particle weight is proportional to the difference between cv error of the particle and cv error averaged over all particles on the current iteration. Each weight is also multiplied by an "overlapping" factor that gives priority to the particles whose classification errors are consistently lower than errors observed on the previous iteration. The details of the method are summarized in Algorithm 1.

Algorithm 1: Particle filtering with simulated annealing

```
   Notation:
   • $1(x)$ is a step function: $1(x) = 1$ if $x > 0$, otherwise 0
   • $ln$ is natural logarithm
   • $U(a, b)$ denotes a real number sampled from a uniform distribution in $[a, b]$
   • $X$ denotes a class of features $f_X \subset F$ (wavelets, raw, etc)
   • $F$ denotes the full feature set
1. Initialize: $f_{sa} = f_0 = 0.99$, $\beta = 50$, $t_{max} = 10$, $t = 0$, $q = 5$, $m = 20$, $p = 0.2$,
   $N_{min} = 30$, $N_{max} = 2000$, $\nu_{min} = 0.01$, $\nu_{max} = 0.5$
2. Initialize $m$ particles with input parameters $N = \#\text{samples}/3, \nu = 15/N, f = F$
3. Evaluate particle weights:
   For each particle $p_i$
     calculate q-fold cv error $\{\epsilon_i^{(k)}\}_{k=1..q}$, $\epsilon_i = \sum_{k=1}^{q} \epsilon_i^{(k)}$
     calculate the overlap factor $\gamma_i$ (equal to 1 for $t = 1$):
       calculate the average $E(\epsilon_{min})$ and standard deviation $Std(\epsilon_{min})$ of the errors
       distribution of the maximal weight particle from the previous iteration $\{\epsilon_{min}^{(k)}\}$
       Calculate the number $J$ of CV fold errors that fall below
           $\epsilon_b = E(\epsilon_{min}) - 3 \cdot Std(\epsilon_{min})$: $J = \sum_{k=1}^{q} 1(\epsilon_b - \epsilon_i^{(k)})$.
       $\gamma_i = (J + 1)/q$
   EndFor
   Calculate average error $\bar{\epsilon} = \frac{1}{m} \sum_{1}^{m} \epsilon_i$
   For each particle $p_i$ calculate weight $w_i = (\bar{\epsilon} - \epsilon_i)1(\bar{\epsilon} - \epsilon_i)\gamma_i$.
4. Resample particles
   For each $i = 1..m$
     Sample a number $j$ from $1..m$ with probabilities proportional to $w_i$
     Set a new particle $p_i^{(n)}$ with $\{N_i^{(n)}, \nu_i^{(n)}, f_i^{(n)}\}$, equal to parameters of $p_j$
     Resample particle parameters:
       $N_i^{(n)} = N_i^{(n)} + \exp U(ln(N_{min}), ln(N_{max})) f_{sa} 1(U(0, 1) - 0.5)$
       $\nu_i^{(n)} = \nu_i^{(n)} + \exp U(ln(\nu_{min}), ln(\nu_{max})) f_{sa} 1(U(0, 1) - 0.5)$
       For each feature class $X$
         If $U(0, 1) < p$ then do
             [If $f_X \subset f_i^{(n)}$ then remove $f_X$ from $f_i^{(n)}$ otherwise add $f_X$ to $f_i^{(n)}$]
       EndFor
     $f_{sa} = f_0^{\beta t}$
   $t = t + 1$
   If $t > t_{max}$ End otherwise Goto 3
```

We test our TSC method on several UCR datasets. We run the algorithm on each dataset 10 times with different GBT random seed to reduce possible effect of fluctuations. The results are summarized in Table 2. One can see that GBT model with parameters and feature subset optimized by Particle Filtering is almost always notably superior to one built without such optimization. Test error values of simple DTW+1NN classifier are listed for comparison.

Table 2. Test errors.

Dataset	Average test error on all features	Average test error with PF	DTW+1NN error
Beef	0.167	0.13	0.467
CBF	0.0392	0.0186	0.004
Coffee	0.0214	0.00357	0.179
ECG200	0.068	0.052	0.12
Face(all)	0.14	0.191	0.192
Face(four)	0.124	0.0557	0.114
Fish	0.167	0.147	0.160
Gun-Point	0.0793	0.0793	0.087
Lightning-2	0.251	0.131	0.131
Lightning-7	0.290	0.256	0.288
OliveOil	0.2	0.17	0.167
OSU Leaf	0.395	0.355	0.384
Swedish Leaf	0.101	0.107	0.157
Synthetic Control	0.025	0.012	0.017
Trace	0.0	0.0	0.01
Two Patterns	0.0075	0.0	0.0015
Wafer	0.017	0.00393	0.005
Yoga	0.161	0.163	0.155

Conclusions

This work deals with TS classification problems. The proposed approach creates a massive number of features including original signals, by-class warped signals, wavelet and chebychev decomposition coefficients of warped signals, summary statistical moments of warped signals, and even predicted by DTW-1-NN labels used as input features. Gradient boosting of trees with imbedded dynamic feature weighting capable of handling hundreds of thousands predictors is then used for classification. Model parameters and the subset of features that the model is trained on are optimized using Particle Filtering. A set of experiments on UCR datasets show that this combination provides a superior learner relative to the well know state of the art approach. The future work will concentrate on refining of this approach for important industrial applications and porting the methodolgy to the time series regression problems.

References

1. B. A., K.Torkkola, and T. E. *Best Subset Feature Selection for Massive Mixed-Type Problems*, volume 4224/2006, pages 1048–1056. Springer, 2006.
2. R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *In proceedings of the 4th Int'l Conference on Foundations of Data Organization and Algorithms*, pages 69–84, 1993.
3. Anonymous.A. Signal classification through massive feature extraction from warped signals. In *Submitted to ECML*, 2007.
4. D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. *In Working Notes of the Knowledge Discovery in Databases Workshop*, pages 359–370, 1994.
5. A. Borisov, V. Eruhimov, and E. Tuv. Dynamic soft feature selection for tree-based ensembles. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications.* Springer, New York, 2005.
6. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
7. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, MA, 1984.
8. Y. Cai and R. T. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD*, 2004.
9. K. Chan and A. W. Fu. Efficient time series matching by wavelets. In *In proceedings of the 15th IEEE Int'l Conference on Data Engineering*, pages 126–133, 1999.
10. K. Deng, A. Moore, and M. Nechyba. Learning to recognize time series: Combining arma models with memory-based learning. In *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, volume 1, pages 246 – 250, 1997.
11. J. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University, 1999.
12. J. Friedman. Stochastic gradient boosting. Technical report, Dept. of Statistics, Stanford University, 1999.
13. P. Geurts. Pattern extraction for time series classification. In *In proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–127, Freiburg, Germany, Sep 3-7 2001.
14. T. Hastie, R. Tibshirani, and J. Friedman. *The elemetns of statistical learning: Data mining, inference, prediction.* Springer, 2001.
15. Y. Huang and P. S. Yu. Adaptive query processing for time-series data. In *In proceedings of the 5th Int'l Conference on Knowledge Discovery and Data Mining*, pages 282–286, San Diego, CA, Aug 15-18 1999.
16. E. Keogh. Data mining and machine learning in time series databases, 2004.
17. E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *In Proceedings of IEEE International Conference on Data Mining*, pages 289–296, 2001.
18. E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage, 2006.
19. F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *In proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 289–300, 1997.

20. J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, June 13 2003.
21. I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *In proceedings of the 18th Int'l Conference on Data Engineering*, pages 212–221, San Jose, CA, Feb 26-Mar 1 2002.
22. C. A. Ratanamahatana and K. E. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data, in conjunction with the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.
23. H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, ASSP-26:43–49, 1978.

Статья поступила в редакцию 25.04.2008