Trees and Random Forests for Univariate/Multivariate Time Series Prompt Classification

Pietro Sala

Biomedical Decision Support Systems 24/25

1 Basic Definitions

In this section, we introduce the foundational concepts for prompt time series classification using tree-based methods.

Given:

- x is a time series
- y is a label
- $z = (x, y)$ is a sample

e.g., $z = ((0.1, 0.5, 1.2, 3), -1)$, where $z \in \mathbb{R}^\ell \times \mathcal{L}$

Definition 1 (Classification)**.** *A classification function* $h : \mathbb{R}^\ell \to \mathcal{L}$

Definition 2 (Early Classification)**.** *An early classification function* $h : \mathbb{R}^{\ell'} \to \mathcal{L}$*, where* $\ell' < \ell$

Definition 3 (Prompt Classification)**.** *A prompt classification function* $h : \bigcup_{\ell=1}^{\infty} \mathbb{R}^\ell \to \mathcal{L} \cup \{\textit{too-early}\}$

Definition 4 (Slice Test)**.** *Given a time series* x*, a* slice test *is a triple* (f, b, e) *where:*

- $f : \mathbb{R}^{e-b} \to \{true, false\}$ *is a boolean function*
- b *is the beginning index of the slice*
- e *is the ending index of the slice, with* $b < e$

The slice test evaluates to:

$$\lambda x.f(x[b : e])$$

This test applies function f *to the slice* $x[b : e]$ *of the current time series.*

Definition 5 (Reference Slice Test (RST))**.** *A* reference slice test *is a specific slice test where* f *is defined using a reference time series. It is specified by the tuple* $(\overline{x}, b, e, \delta, \varepsilon)$*, where:*

- $\overline{x}$ *is a reference time series*
- b *is the beginning index of the slice*
- e *is the ending index of the slice*
- δ *is a distance function*
- ε *is a threshold*

This test evaluates to:

$$\lambda x.(\delta(x[b : e], \overline{x}) \leq \varepsilon)$$

This test returns true when the distance between the slice $x[b : e]$ *of the current time series and the reference time series* $\overline{x}$ *is less than or equal to the threshold* ε*.*

Definition 6 (Prompt Tree)**.** *A* prompt tree *is a decision tree* $T = (V, E, S, \mathcal{L})$ *where:*

- V *is the set of vertices*
- $E \subseteq V \times V$ *is the set of edges*
- $S : V \to RST \cup \{\textit{is-leaf}\}$ *assigns a reference slice test or indicates a leaf node*
- $\mathcal{L} : V \to$ *Distribution over* $\mathcal{L}$ *assigns a distribution over possible labels to each node*

Let $P(v) = \{v_0, v_1, ..., v_k = v\}$ *be the path from the root* v_0 *to node* v *in the tree. For any node* $v \in V$*, we formally define:*

- $B(v) = \{b_i \mid S(v_i) = (\overline{x}_i, b_i, e_i, \delta_i, \varepsilon_i), v_i \in P(v)\}$*, the set of all beginning indices of RSTs along the path from the root to node* v*, inclusive of* v*.*
- $E(v) = \{e_i \mid S(v_i) = (\overline{x}_i, b_i, e_i, \delta_i, \varepsilon_i), v_i \in P(v)\}$*, the set of all ending indices of RSTs along the path from the root to node* v*, inclusive of* v*.*

The set $B(v)$ contains the beginning indices of all RSTs along the path from the root to node v, while $E(v)$ contains the ending indices of all RSTs along the same path. We have the following structural constraints. Let $(\overline{x}, b, e, \delta, \varepsilon) = S(root)$ then $b = 0$ (the root node test starts at the beginning of the time series). For each pair of internal non-root nodes v, v' such that $(v, v') \in E$ we have: let $S(v) = (\overline{x}, b, e, \delta, \varepsilon)$ and $S(v') = (\overline{x}', b', e', \delta', \varepsilon')$, then $b' \in B(v) \cup \{\max(E(v)) + 1\}$ (each child node either reuses a beginning index from its ancestors or moves forward by one with respect to the maximum ending index reached so far).

Multivariate trees are structurally the same the only difference is that they deal with Channel Reference Slice Tests (CSRTs).

Definition 7 (CRST (Channel Reference Slice Test))**.** *A channel reference slice test is specified by the tuple $(\overline{x}, c, b, e, \delta, \varepsilon)$, where:*

- *$\overline{x}$ is a reference time series*
- *c is the channel index*
- *b is the beginning index of the slice*
- *e is the ending index of the slice*
- *δ is a distance function*
- *ε is a threshold*

This test evaluates to:

$$\lambda x.(\delta(x[c][b:e], \overline{x}) \leq \varepsilon)$$

This test returns true when the distance between the slice $x[c][b:e]$ of channel c of the current time series and the reference time series $\overline{x}$ is less than or equal to the threshold ε.

2 Algorithm for Prompt Tree Fitting

In this section, we present the algorithm for fitting a prompt tree. Unlike traditional decision trees, this algorithm allows for flexible distance functions at each node and works in both supervised and unsupervised settings through customizable functions.

2.1 Explanation of Components

The PromptTreeFit algorithm builds a prompt tree for time series classification with the following customizable components:

- **Promptness Function (f_p):** Determines which intervals in the time series to consider for testing at each node. It accepts the set of beginning indices from the current path, the maximum ending index plus one, and the current dataset, and returns potential (channel, interval) pairs for testing.
- **Sampling Function (f_s):** Generates a set of candidate tests based on the intervals selected by f_p. Each test consists of a channel, reference time series, threshold, interval, and distance function.
- **Classification Function (f_c):** Determines the output distribution at leaf nodes. This can be used in both supervised settings (based on Y values) and unsupervised settings.
- **Distance Functions (Δ):** A set of distance functions that can be used to compare time series slices. Different nodes can use different distance functions.
- **Optimization Function (f_o):** Selects the optimal test from the candidates generated by f_s. This determines the split at each node.
- **Stopping Criterion (f_e):** Determines when to stop expanding the tree and create a leaf node instead.

3 Assignment: Time Series Classification Using Tree-Based Methods

This assignment focuses on implementing and evaluating tree-based methods for time series classification, including decision trees and random forests. You will work with both univariate and multivariate time series data, and explore supervised and unsupervised approaches.

3.1 Part 1: Implementation

3.1.1 Decision Tree Implementation

Implement a decision tree class inspired by the PromptTreeFit algorithm discussed in class. Your

Algorithm 1 PromptTreeFit

Require:

X - Set of time series data

Y - Labels (optional, can be None for unsupervised learning)

Path - Current path in the tree (None for root node creation)

f_p - Promptness function: $2^{\mathbb{N}} \times \mathbb{N} \times \text{Dom}(X) \times \text{Dom}(Y) \to 2^{\mathbb{N} \times \mathbb{I}(\mathbb{N})}$

f_s - Sampling function: $\text{Dom}(X) \times \text{Dom}(Y) \times 2^{\mathbb{N} \times \mathbb{I}(\mathbb{N})} \to 2^{\mathbb{N} \times \mathbb{R}^k \times \mathbb{R} \times \mathbb{I} \times \Delta}$

f_c - Classification function for leaf nodes

Δ - Set of distance functions: $\cup_{k=1}^{\infty} \mathbb{R}^k \to \mathbb{R}$

f_o - Optimization function: $\text{Dom}(X) \times \text{Dom}(Y) \times 2^{\mathbb{N} \times \mathbb{R}^k \times \mathbb{R} \times \Delta} \to \mathbb{N} \times \mathbb{R}^k \times \mathbb{R} \times \mathbb{I} \times \Delta$

f_e - Stopping criterion: $2^{\text{Path}} \times \text{Dom}(X) \times \text{Dom}(Y) \to \{\text{true}, \text{false}\}$

Ensure: A fitted prompt tree node and its subtree

if Path = None **then**
  Path $\leftarrow \emptyset$ ▷ Initialize empty path for root node
  $B \leftarrow \{0\}$ ▷ Root test starts at beginning of time series
else
  $B \leftarrow \{b_i \mid (\overline{x}_i, c_i, b_i, e_i, \delta_i, \varepsilon_i) \in \text{Path}\}$ ▷ Extract beginning indices from path
  $E \leftarrow \{e_i \mid (\overline{x}_i, c_i, b_i, e_i, \delta_i, \varepsilon_i) \in \text{Path}\}$ ▷ Extract ending indices from path
end if
if $f_e(\text{Path}, X, Y) = \text{true}$ **then**
  return LeafNode($f_c(X, Y)$) ▷ Create leaf with classification distribution
end if
CandidateIntervals $\leftarrow f_p(B, \max(E) + 1, X, Y)$ ▷ Get candidate intervals
CandidateTests $\leftarrow f_s(X, Y, \text{CandidateIntervals})$ ▷ Generate candidate tests
$(c, \overline{x}, \varepsilon, [b, e], \delta) \leftarrow f_o(X, Y, \text{CandidateTests})$ ▷ Select optimal test
Node $\leftarrow$ InternalNode$(c, \overline{x}, b, e, \delta, \varepsilon)$ ▷ Create node with selected test
$X_{\text{true}}, Y_{\text{true}} \leftarrow \text{Split}(X, Y, (c, \overline{x}, b, e, \delta, \varepsilon), \text{true})$
$X_{\text{false}}, Y_{\text{false}} \leftarrow \text{Split}(X, Y, (c, \overline{x}, b, e, \delta, \varepsilon), \text{false})$
$\text{Path}_{\text{true}} \leftarrow \text{Path} \cup \{(c, \overline{x}, b, e, \delta, \varepsilon)\}$ ▷ Extend path with current test
Node.left $\leftarrow$ PromptTreeFit$(X_{\text{true}}, Y_{\text{true}}, \text{Path}_{\text{true}}, f_p, f_s, f_c, \Delta, f_o, f_e)$
$\text{Path}_{\text{false}} \leftarrow \text{Path} \cup \{(c, \overline{x}, b, e, \delta, \varepsilon)\}$ ▷ Extend path with current test
Node.right $\leftarrow$ PromptTreeFit$(X_{\text{false}}, Y_{\text{false}}, \text{Path}_{\text{false}}, f_p, f_s, f_c, \Delta, f_o, f_e)$
return Node

implementation should:

- Store necessary information in the nodes (e.g., class distribution for supervised learning)
- Include an optional validation method for post-pruning using a validation dataset
- Provide a method for returning the paths of the tree (for distance calculations)

3.1.2 Random Forest Implementation

Implement a random forest approach based on your decision tree implementation. Your random forest should:

- Support the three voting mechanisms (majority, weighted, and track-record)
- Allow for both supervised and unsupervised (isolation) approaches

3.2 Part 2: Evaluation

Test your implementations on at least four datasets with the following combinations:

- Univariate + Binary Classification
- Univariate + Multi-class Classification (3+ classes)
- Multivariate + Binary Classification
- Multivariate + Multi-class Classification (3+ classes)

3.2.1 Supervised Learning Evaluation

For the classical random forest with the three voting mechanisms:

- Perform standard train-test evaluation using appropriate accuracy measures
- Create a conformal classifier by using part of the training data:
  - Check miscalibration
  - Compute efficiency

3.2.2 Unsupervised Learning Evaluation

Implement an isolation forest (unsupervised) and cluster the data according to the three distances:

- Breiman distance
- Zhu distance
- RatioRF distance

For each distance measure, perform the following clustering procedure:

- Start with a number of clusters equal to the number of samples
- Iteratively merge the two clusters with minimal average inter-distance between their samples
- Continue until the number of clusters equals the number of classes in the dataset

Evaluate the clustering results using:

- Internal validation: Calculate intra-cluster and inter-cluster distances
- External validation: Compute either purity or entropy metrics
- Clustering comparison: Calculate the Adjusted Rand Index (ARI) among the three clusterings

3.3 Part 3: Data Sources

Use the following datasets for your evaluation:

- Univariate time series: Select from datasets available at `http://www.timeseriesclassification.com/aeon-toolkit/Archives/Univariate2018_ts.zip`
- Multivariate time series: Select from datasets available at `http://www.timeseriesclassification.com/aeon-toolkit/Archives/Multivariate2018_ts.zip`

3.4 Submission Requirements

Your submission should include:

- Complete, well-documented source code for your implementations
- A report detailing your implementation choices, experimental setup, and evaluation results
- Visualizations of the decision trees and clustering results where appropriate
- Analysis of the performance differences between the different methods and distance measures