The Journal of Open Source Software

fABBA: A Python library for the fast symbolic approximation of time series

Xinye Chen [1] **and Stefan Güttel** [2]

1 Department of Numerical Mathematics, Charles University Prague, Czech Republic **2** Department of Mathematics, The University of Manchester, United Kingdom

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 06 December 2023
Published: unpublished

Summary

Adaptive Brownian bridge-based aggregation (ABBA) (Elsworth & Güttel, 2020a) is a symbolic time series representation approach that is applicable to general time series. It is based on a tolerance-controlled polygonal chain approximation of the time series, followed by a mean-based clustering of the polygonal pieces into groups. With the increasing need for faster time series processing, lots of efforts have been put into deriving new time series representations in order to reduce the time complexity of similarity search or enhance forecasting performance of machine learning models. Compared to working on the raw time series data, symbolizing time series with ABBA provides numerous benefits including but not limited to (1) dimensionality reduction, (2) smoothing and noise reduction, and (3) explainable feature discretization. The time series features extracted by ABBA enable fast time series forecasting (Elsworth & Güttel, 2020b), anomaly detection (Chen & Güttel, 2023; Elsworth & Güttel, 2020a), event prediction (Gogineni et al., 2022), classification (Nguyen & Ifrim, 2023; Taktak et al., 2024), and other data-driven tasks in time series analysis (Harris et al., 2021; Wang et al., 2023). An example illustration of an ABBA symbolization is shown in Figure 1.

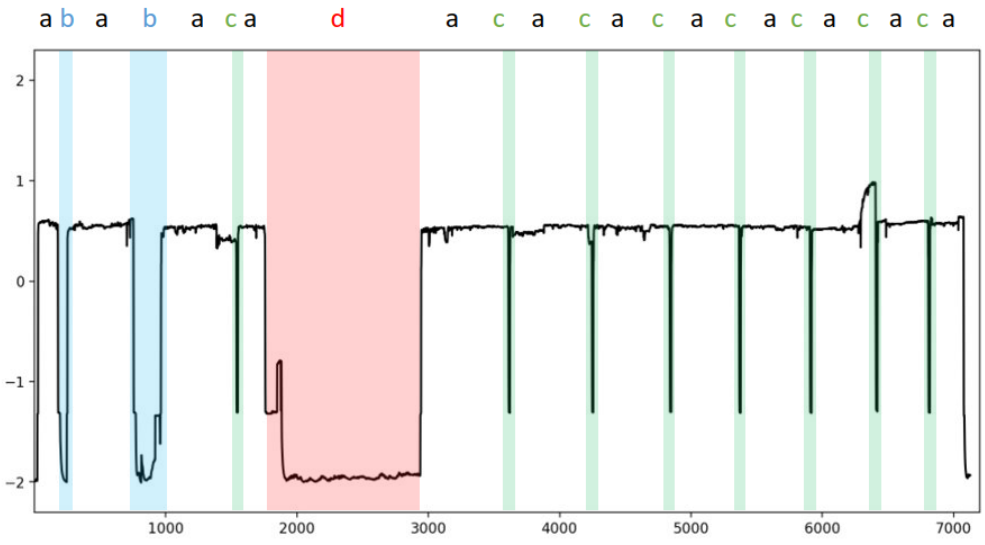

Figure 1: ABBA symbolization with 4 symbols.

ABBA follows a two-phase approach to symbolize time series, namely compression and digitization. The first phase aims to reduce the time series dimension by polygonal chain approximation, and the second phase assigns symbols to the polygonal pieces. Both phases operate together

to ensure that the essential time series features are best reflected by the symbols, controlled by a user-chosen error tolerance. The advantages of the ABBA representation against other symbolic representations include (1) better preservation of essential shape features, e.g., when compared against the popular SAX representation (Elsworth & Güttel, 2020a; Lin et al., 2003); (2) effective representation of local up and down trends in the time series which supports motif detection; (3) demonstrably reduced sensitivity to hyperparameters of neural network models and the initialization of random weights in forecasting applications (Elsworth & Güttel, 2020b).

fABBA is a Python library to compute ABBA symbolic time series representations on Linux, Windows, and MacOS systems. With Cython compilation and typed memoryviews, it significantly outperforms existing ABBA implementations. The *fABBA* library also includes a new ABBA variant, fABBA (Chen & Güttel, 2023), which uses a fast alternative digitization method (i.e., greedy aggregation) instead of k-means clustering (Lloyd, 1982), providing significant speedup and improved tolerance-based digitization (without the need to specify the number k of symbols a priori). The experiments in Chen & Güttel (2023) demonstrate that fABBA runs significantly faster than the original ABBA module at `https://github.com/nla-group/ABBA/`. *fABBA* is an open-source library and licensed under the 3-Clause BSD License. Its redistribution and use, with or without modification, are permitted under conditions described in `https://opensource.org/license/bsd-3-clause/`.

Examples

fABBA can installed via the Python Package Index or conda forge. Detailed documentation for its installation, usage, API reference, and quick start examples can be found on `https://fabba.readthedocs.io/en/latest/`. Below we provide a brief demonstration.

Compress and reconstruct a time series

The following example approximately transforms a time series into a symbolic string representation (using method `transform()`) and then converts the string back into a numerical format (using method `inverse_transform()`). fABBA requires two parameters, `tol` and `alpha`. The tolerance `tol` determines how closely the polygonal chain approximation follows the original time series. The parameter `alpha` controls how similar time series pieces need to be in order to be represented by the same symbol. A smaller `tol` means that more polygonal pieces are used and the polygonal chain approximation is more accurate; but on the other hand, it will increase the length of the string representation. Similarly, a smaller `alpha` typically results in more accurate symbolic digitization but a larger number of symbols.

```python
import numpy as np
import matplotlib.pyplot as plt
from fABBA import fABBA

# original time series
ts = [np.sin(0.05*i) for i in range(1000)]
fabba = fABBA(tol=0.1, alpha=0.1, sorting='2-norm', scl=1, verbose=0)

# symbolic representation of the time series
string = fabba.fit_transform(ts)
# prints aBbCbCbCbCbCbCbCA
print(string)

# reconstruct numerical time series
inverse_ts = fabba.inverse_transform(string, ts[0])
```

More ABBA variants}

Other clustering-based ABBA variants are also provided, supported by the clustering methods in the *scikit-learn* library (Pedregosa et al., 2011). Below is a basic code example.

```
import numpy as np
from sklearn.cluster import KMeans
from fABBA import ABBAbase

# original time series
ts = [np.sin(0.05*i) for i in range(1000)]
# k-means clustering with 5 symbols
kmeans = KMeans(n_clusters=5, random_state=0, init='k-means++', verbose=0)
abba = ABBAbase(tol=0.1, scl=1, clustering=kmeans)

# symbolic representation of the time series
string = abba.fit_transform(ts)
# prints BbAaAaAaAaAaAaAaC
print(string)

# reconstruct numerical time series
inverse_ts = abba.inverse_transform(string)
```

Statement of Need

Symbolic representations enhance time series processing by a large number of powerful techniques developed, e.g., by the natural language processing or bioinformatics communities (Lin et al., 2003, 2007). *fABBA* is a Python module for computing such symbolic time series representations very efficiently, enabling their use for downstream tasks such as time series classification, forecasting, and anomaly detection.

Acknowledgement

Stefan Güttel acknowledges a Royal Society Industry Fellowship IF/R1/231032.

References

Chen, X., & Güttel, S. (2023). An efficient aggregation method for the symbolic representation of temporal data. *ACM Transactions on Knowledge Discovery from Data*, *17*(1), 1–22.

Elsworth, S., & Güttel, S. (2020a). ABBA: adaptive Brownian bridge-based symbolic aggregation of time series. *Data Mining and Knowledge Discovery*, *34*, 1175–1200.

Elsworth, S., & Güttel, S. (2020b). *Time series forecasting using LSTM networks: A symbolic approach* (No. arXiv:2003.05672v1; p. 12).

Gogineni, K., Derasari, P., & Venkataramani, G. (2022). Foreseer: Efficiently forecasting malware event series with long short-term memory. *IEEE International Symposium on Secure and Private Execution Environment Design*, 97–108.

Harris, J. J., Chen, C.-H., & Zaki, M. J. (2021). A framework for generating summaries from temporal personal health data. *ACM Transactions on Computing for Healthcare*, *2*(3).

Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2–11.

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, *15*(2), 107–144.

Lloyd, S. P. (1982). Least squares quantization in PCM. *Transactions on Information Theory*, *28*, 129–137.

Nguyen, T. L., & Ifrim, G. (2023). Fast time series classification with random symbolic subsequences. *Advanced Analytics and Learning on Temporal Data: 7th ECML PKDD Workshop*, 50–65.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Taktak, M., Ltifi, H., & Ayed, M. B. (2024). ECG classification with learning ensemble based on symbolic discretization. *Information Systems*, *120*, 102294.

Wang, C., Dou, M., Li, Z., Outbib, R., Zhao, D., Zuo, J., Wang, Y., Liang, B., & Wang, P. (2023). Data-driven prognostics based on time-frequency analysis and symbolic recurrent neural network for fuel cells under dynamic load. *Reliability Engineering & System Safety*, *233*, 109123.