Journal of Machine Learning Research 23 (2022) 1-7 Submitted 1/21; Revised 5/22; Published 9/22

Feature selection for time series

Mansoor Rezghi REZGHI@MODARES.AC.IR
Department of Computer Science
Tarbiat Modares University

Sepide Moradi SEPIDE.MORADI73@GMAIL.COM
Department of Computer Science
Tarbiat Modares University

Editor: My editor

Abstract

Feature selection is a crucial task when working with time series data as it helps to identify the most relevant features that can improve the performance of machine learning models. In recent years, with the rapid advance in technology, the amount of time series data has exploded, making feature selection even more important. However, feature selection for time series data is different from that of non-time dependent data due to the temporal nature of such data.

In this paper, we propose two novel methods for feature selection in time series data that utilize local features to identify the most dominant series. Specifically, we create a network that illustrates the relationship between local features based on their similarity. We then apply PageRank algorithm on this network to extract the most important features.

Our experiments show that the proposed models outperform existing feature selection methods for time series data, demonstrating their effectiveness in improving the performance of time series models. By utilizing local features and network analysis techniques, our methods are able to identify the most relevant features that contribute to the predictive power of time series models.

Keywords: time series, feature selection

1 Introduction

Multidimensional time series data is characterized by the presence of multiple variables or dimensions that change over time. These types of data are commonly encountered in a range of fields, from finance and economics to healthcare and engineering. When analyzing multidimensional time series data, it is often useful to identify the dominant series or variables that are driving the overall behavior of the system. By doing so, we can gain insights into the underlying dynamics of the system and potentially use this information for feature selection or other downstream analysis.

The goal of feature selection is to identify the most relevant features for a particular machine learning task, while removing irrelevant or redundant ones. This can help in reducing the dimensionality of the data, improve model accuracy and generalization, and reduce overfitting. For time series data, feature selection can be different from traditional feature selection methods because the data is structured in a time-dependent manner. The main difference is that features are often selected based on their ability to predict future values

of the time series, rather than just their correlation with the target variable.Furthermore, unlike traditional feature selection where all data points are independent, in time series data, each observation is dependent on past observations. Therefore, the choice of relevant features becomes more critical as the impact of each feature on the prediction changes over time. Hence, feature selection for time series data requires specialized techniques such as lag features, rolling statistics, and time-based transformations.

In this study, we propose two novel feature selection methods for time series analysis. In the following sections, we provide a detailed explanation of both methods and compare their performance with other baseline methods in section 3.

2 Methodology

In time series analysis, we often encounter data where patterns can change over time. In order to identify dominant patterns that persist over longer periods, it is necessary to observe how similar these patterns are across different time periods. One way to achieve this is through the comparison of the patterns of the same short period of time in all series.

For example, let's say we have three time series data sets A, B, and C, with each containing observations at different time points. To determine which of these series has the most dominant pattern, we can take a sliding window of a fixed length (e.g., one week), and compute the cosine similarity between the patterns of A, B, and C during that window. This allows us to compare the similarity of the patterns of each series at different time periods. By using cosine similarity, we can analyze time series data to identify dominant patterns that persist over longer periods of time.

The size of the moving window is an important hyperparameter that must be chosen carefully based on the characteristics of the time series data. The choice of window size depends on various factors, such as the frequency of the data, the length of the time series, the complexity of the patterns, and the objectives of the analysis. A larger window size can provide a broader view of the data, allowing us to capture longer-term patterns, but it may not be sensitive enough to detect changes in short-term patterns. Conversely, a smaller window size can detect short-term changes more accurately, but it may not capture longer-term patterns with enough granularity. Choosing an appropriate window size requires balancing these considerations while taking into account the specific goals of the analysis. In some cases, it may be necessary to experiment with different window sizes to find the one that best captures the patterns of interest in the data.

2.1 Method1

After dividing the main time series $X \in R^{n \times T}$ into smaller time series, we obtain a set containing

$$\tilde{X}_i = [x_{(i-1)w+1}, \ldots, x_{(i-1)w+w}] \in \mathbb{R}^{n \times w}, \quad i = 1, \ldots, k = \left\lfloor \frac{T}{w} \right\rfloor \tag{1}$$

where w is the size of the window.

we can compare the similarity of the patterns between each pair of time series using cosine similarity. This results in a matrix of pairwise similarities for each time period. If we stack these matrices in order of their time periods, we obtain a 3D tensor $\mathcal{S} \in R^{n \times n \times K}$,

where n is the number of features in the time series and K is the number of smaller time series, so:

$$\mathcal{S}_{:,:,i} = |\tilde{X}_i \tilde{X}_i^\top|, \quad i = 1, \ldots, k \tag{2}$$

Based on the observation that each mode 2 fiber (i.e., $\mathcal{S}_{i,j,:}$) in the similarity tensor reflects changes in the similarity of the same pair over time, the next step could be to analyze these fibers to gain insights into how the similarity between pairs evolves over time. To gain insights into how the relationships between pairs of objects evolve over time, we analyzed the similarity changes of fibers in our similarity tensor. Specifically, we used cosine similarity to calculate the similarity between each pair of fibers. We examined the evolution of relationships between pairs of objects over time by analyzing the changes in similarity of fibers within our similarity tensor. Specifically, we utilized cosine similarity to calculate the similarity between each pair of fibers. To keep track of which similarity value corresponds to which pair in the final similarity matrix, the tensor was unfolded along its third mode (fibers). Specifically, the fibers were arranged under each other in a specific order, with $\mathcal{S}_{1,1,:}$, $\mathcal{S}_{2,1,:}$, ..., $\mathcal{S}_{n,1,:}$ for the first set of fibers, $\mathcal{S}_{1,2,:}$, $\mathcal{S}_{2,2,:}$, ..., $\mathcal{S}_{n,2,:}$ for the second set of fibers, and so on until the last set of fibers was listed as $\mathcal{S}_{1,n,:}$, $\mathcal{S}_{2,n,:}$, ..., $\mathcal{S}_{n,n,:}$. The resulting matrix $H \in \mathbb{R}^{n^2 \times n^2}$ is the cosine similarity matrix of $unfold(\mathcal{S})_3$ so:

$$H_{n(i-1)+j,n(p-1)+q} = \frac{\mathcal{S}_{i,j,:}\mathcal{S}_{p,q,:}^\top}{||\mathcal{S}_{i,j,:}||.||\mathcal{S}_{p,q,:}||} \tag{3}$$

This matrix can be considered as an adjacency matrix of a graph, where each node corresponds to a fiber and edges represent the similarity between pairs of fibers. By treating the similarity matrix as a graph, we can use various graph theory-based techniques to analyze it further.

After constructing the similarity graph, one approach we can use to further analyze and understand the data is static rank. To compute the static rank of each node, one common method is to use the PageRank algorithm. PageRank computes a score for each node based on the number and quality of incoming edges, where higher scores indicate greater importance.

By identifying which features were involved in the static rank and relate them back to the original data, we can use this formula to calculate the total importance of each node:

$$\text{score}(i) = \sum_{j=1}^{n} r_{n(i-1)+j}$$

where r is pagerank vector of matrix H. After computing the scores for each node in our similarity graph, we selected the features with the highest scores to use in our modeling.

We call our method "feature selection based on order 2 similarity" to emphasize its use of order 2 similarity measures to identify important features in high-dimensional datasets. Figure 1 provides a visual representation of the actions taken in each step of the process.

2.2 method2

as mentioned earlier, When analyzing time series data, it is common to use sliding windows to capture subsets of the data over a given period. The length of the sliding window can

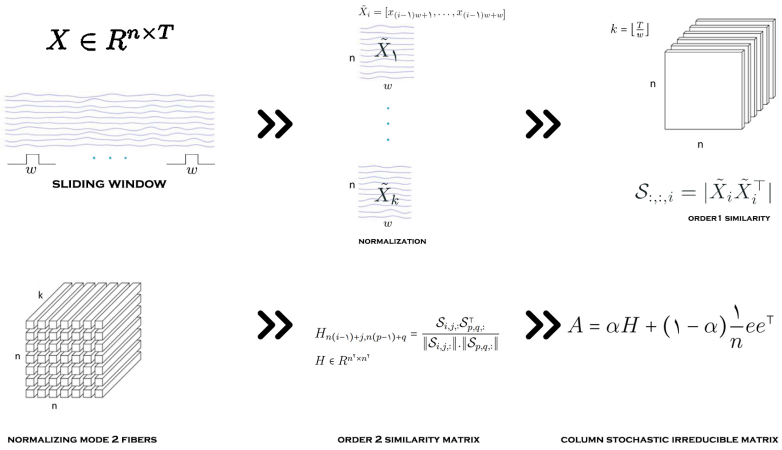

Figure 1: feature selection based on order 2 similarity

vary depending on the analysis being performed and the characteristics of the data being analyzed.

After we have created the similarity tensor ,$\mathcal{S} \in R^{n \times n \times k}$ with the frontal slices arranged in time order, the next step is to project each frontal slice onto an orthogonal space using SVD (singular value decomposition).suppose:

$$\mathcal{S}_{:,:,i} = U_i \Sigma_i V_i \quad \forall i \in \{1, \ldots, k\} \tag{4}$$

Then $\mathcal{A}$, the denoised version of S is:

$$\mathcal{A}_{:,:,i} = U_i \mathcal{S}_{:,:,i} \quad \forall i \in \{1, \ldots, k\} \tag{5}$$

It's important to note that the main goal of this projection step is not to reduce the dimensionality of the data, but rather to reduce the amount of noise in the data. By removing noise, we can more accurately identify and analyze the underlying patterns and features in the time-series data.

By treating this tensor as an adjacency tensor of a dynamic graph, we can analyze the behavior of the features over time.moreover, It is worth mentioning that interpreting tensors as graphs allows us to leverage the mathematical properties of graph theory to perform analysis and modeling tasks on our data. For example The Temporal PageRank algorithm is a powerful tool in graph theory that extends the traditional PageRank algorithm to dynamic networks.

Using the temporal PageRank values to select important nodes as a feature selection technique has several advantages. First, it allows us to capture the time-varying behavior of the network and identify nodes that are consistently important over time. Second, it takes into account the importance of both direct and indirect connections between nodes in determining their overall influence within the network.

Specifically, Polina Rozenshtein and Aristides Gionis use the concept of "temporal walk" in their paper on Temporal PageRank. A temporal walk is a path in a temporal network that follows the edges of the network in chronological order, with each step only moving to

nodes that are reachable from the current node at the current time or later. This allows them to take into account the time dimension of the network when computing the transition probabilities between nodes. By analyzing such temporal walks, they develop a method for assigning time-dependent weights to the edges in the network, which is the basis for the Temporal PageRank algorithm.

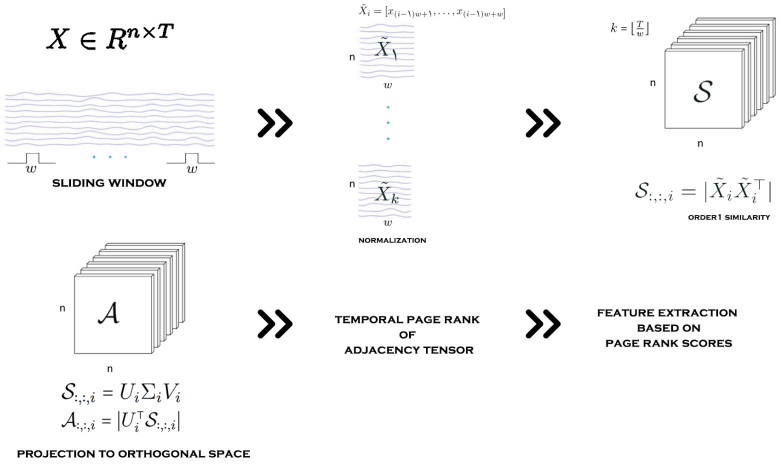

Figure 2: feature selection based on order one similarity

In Figure2, the process of what we do in each step is illustrated. Finally, once we have selected important nodes based on their temporal PageRank values, we can use these nodes as features for further analysis or modeling tasks.

3 Experiments

In this study, we investigate the effectiveness of feature selection algorithms for time series data using a real-world stock market dataset. The dataset contains 1,602,808 records and 76 features, representing daily stock market prices over a period of several years. Each record includes information such as ?? .our ultimate goal is to predict a target variable for each record.

To compare the performance of different feature selection algorithms, we will divide the time series data into successive subsets, with each subset containing 2000 records. We will train and evaluate our XGBoost models on these subsets and report their performance in terms of root mean squared error (RMSE) and running time.

To provide a baseline for comparison, we will also compare the performance of our XGBoost models to our feature extraction techniques: principal component analysis (PCA) and temporally regularized matrix factorization (TRMF). By comparing the performance of our XGBoost models to these baseline approaches, we can determine if feature selection improves the accuracy of our predictive models.

In first experiment, to select an appropriate number of features for our different subsets, we will use principal component analysis (PCA) to extract a reduced set of features based on energy measurements. Specifically, we will select the minimum number of principal

Table 1: energy= 90%

	PCA	TRMF	Temporal_pr	Static_pr
Var	0.029832	0.227876	**0.012063**	0.014211
Mean	1.070715	1.086047	**1.008805**	1.025914
Min	0.029832	0.227876	**0.012063**	0.014211
Max	3.307063	13.278481	**1.421221**	1.878058

		PCA	trmf rmse	temp PR rmse	static PR rmse
energy .70	Var	0.020002	0.051363	**0.011731**	0.013636
	Mean	1.085063	1.090146	1.016642	**0.962086**
	Min	0.020002	0.051363	**0.011731**	0.013636
	Max	2.070929	3.120672	**1.336473**	2.930720

components that explain at least 90% of the variance in the data for each subset. We will then choose the corresponding number of features for other feature selection methods applied to that subset.

References

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3):462–467, 1968.