Time Series Clustering

Hunar Preet (160301)
Mayank Anupam(160388)
Ritesh Kumar (160575)
Suryateja BV (160729)

12th November, 2019

1 References

1. Xiong, Yimin, and Dit-Yan Yeung. "Mixtures of ARMA models for model-based time series clustering." 2002 IEEE International Conference on Data Mining, 2002. Proceedings.. IEEE, 2002.

2. Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–A decade review." Information Systems 53 (2015): 16-38.

2 Introduction

Clustering is a method for classifying massive data if there is no early knowledge of classes. Since new concepts such as cloud computing and big data have emerged in recent years, research work on unsupervised approaches such as clustering algorithms to derive information from this flood of data has been extended. In the case of large data sets, it is almost impossible to use supervised classification methods, although clustering can solve this problem using unsupervised approaches. In various scientific fields, clustering time series data is used to discover patterns that allow data analysts to extract valuable information from large and complex data sets.

Data in many applications is stored as time-series, for example sales data, stock prices, exchange rates in finance, weather data, biomedical measurements (e.g., blood pressure and electrocardiogram measurements), biometrics data (image data for facial recognition), particle tracking in physics, etc. Such a massive amount of time-series data has opened up an opportunity for various researchers in the data mining community to extract and analyse valuable information that is hidden in the data.

2.1 Time-Series Clustering

Given a dataset of n time-series $D = \{F_1, F_2, \ldots, F_n\}$, the methodology of partitioning D into $C = C_1, C_2, \ldots, C_k$ such that homogeneous time-series are grouped together based on some similarity measure. Here C_i is a cluster such that $D = \cup_{i=1}^{k} C_i$ and $C_j \cap C_j = \phi \forall i \neq j$.

Time-series clustering is a challenging issue because first of all, time-series data are often far larger than memory size and consequently they are stored on disks. This leads to an exponential decrease in speed of the clustering process. Second challenge is that time-series data are often

high dimensional which makes handling these data difficult for many clustering algorithms and also slows down the process of clustering. Finally, the third challenge addresses the similarity measures that are used to make the clusters. To do so, similar time-series should be found which needs time-series similarity matching that is the process of calculating the similarity among the whole time-series using a similarity measure. This process is also known as "whole sequence matching" where whole lengths of time-series are considered during distance calculation.

However, the process is complicated, because time-series data are naturally noisy and include outliers and shifts, at the other hand the length of time-series varies and the distance among them needs to be calculated. These common issues have made the similarity measure a major challenge for data miners.

2.2 Applications

Time-series clustering can be broadly divided into two groups: The first one is a set of algorithms that are used to find patterns that frequently appear in the data. The second one is related to finding patterns that occur by chance in the data. Various real world problems related to this task are as follows:

- **Anomaly/novelty detection:** For eg., in sensor databases, clustering of time-series which are produced by sensor readings of a mobile robot in order to discover the events
- **Recognizing dynamic changes** For eg., in financial databases, it can be used to find the companies with similar stock price movements
- **Prediction and recommendation** For eg., in scientific databases, it can address problems such as finding the patterns of solar magnetic wind to predict today's pattern
- **Pattern identification** For eg., in marketing database, different daily patterns of sales of a specific product in a store can be discovered

2.3 Different Types of Clustering

There are three categories for time series clustering:

2.3.1 Whole time-series clustering

Whole time-series clustering is considered as clustering of a set of individual time-series with respect to their similarity. Here, clustering means applying conventional (usually) clustering on discrete objects, where objects are time-series.
Their are four components of whole time-series clustering.

1. Time-Series Representation
2. Similarity or Distance Measures
3. Clustering Prototypes
4. Time-Series Clustering

2.3.2 Subsequence clustering

Subsequence clustering means clustering on a set of subsequences of a time-series that are extracted via a sliding window, that is, clustering of segments from a single long time-series.

2.3.3 Time point clustering

Time point clustering is another category of clustering. It is clustering of time points based on a combination of their temporal proximity of time points and the similarity of the corresponding values. This approach is simialr to time series segmentation. However, it is different from segmentation as all points do not need to be assigned to clusters, id., some of them are considered as noise.

2.4 Our Work

We study and implement a model based approach for this task using mixtures of autoregressive moving average (ARMA) models. This can be thought of as a time series equivalent of the Gaussian Mixture Model, that is a well known probabilistic model for time independent data clustering. We derive the expectation-maximization(EM) algorithm for this mixture model. For the model selection problem ,i.e to find an optimum value of number of clusters, we use the Bayesian information criterion (BIC). The clustering experiments were conducted on various simulated datasets. Our novelty lies in implementing the ARMA mixtures model in Python

3 Related Work

3.1 Markov chains

Finite mixtures of Markov chains have been proposed for clustering time series. The EM algorithm is used to learn mixing coefficients as well as the parameters of the component models. Then number of clusters can be determined by comparing different choices of the number based on some scoring scheme.

3.2 Hidden Markov Models

Some time series can be modeled better using HMM due to their ability of handling uncertainty in temporal and spatial dimensions simultaneously. For example, HMMs have been very successfully used for speech recognition, handwriting recognition and bioinformatics.

3.3 Regression models

Regression models, mixtures of regression models or regression mixtures and their extensions are another type of models that can be used for time series modeling and clustering. Typically, a regression model provides a projection from the baseline status to some relevant demographic variables. Curve-type time series data are quite common examples of these kinds of variables. For example, mixtures of standard regression models and the accompanying EM algorithm have been used for the clustering of trajectory data.

3.4 Autoregressive moving average (ARMA) models

In addition to Markov chains, HMMs and regression models, ARMA and autoregressive integrated moving average (ARIMA) models have also been used extensively for time series analysis. However, clustering applications bases on such mixtures models were not studied.

4 Model-based clustering with ARMA mixtures

4.1 Standard ARMA models

Given a time series $\{x_t\}_{t=1}^n$, the fitted $ARMA(p,q)$ model takes the following form

$$x_t = \phi_0 + \sum_{j=1}^{p} \phi_j x_{t-j} + \sum_{j=1}^{q} \theta_j e_{t-j} + e_t, \quad t = 1, 2, \ldots, n \tag{1}$$

where n is the length of the time series $\Phi = \{\phi_0, \ldots, \phi_p, \theta_1, \ldots \theta_q\}$ are model parameters and $\{e_t\}_{t=1}^n$ is a sequence of i.i.d. gaussian white noise with variance σ^2.

4.2 ARMA mixtures

The ARMA models can be extended to mixtures of ARMA models. In this setting we assume the time-series data is generated from M different ARMA models, which correspond to the cluster M clusters of interest. The clusters are denoted as $\omega_1, \ldots, \omega_k$. The likelihood function of mixture model can be expressed in the form of a mixture density as follows

$$P(\mathbf{x}|\Theta) = \sum_{k=1}^{M} P(\mathbf{x}|\omega_k, \Phi_k) P(\omega_k) \tag{2}$$

Here $P(\omega_k)$ is the prior probability that time series comes from cluster k, $P(\mathbf{x}|\omega_k \Phi_k)$ denotes the conditional likelihood function for cluster k with parameters Φ_k and $\Theta = \Phi_1, \ldots, \Phi_M, P(\omega_1, \ldots, \omega_M)$.

Suppose the complete dataset $D = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ of N time-series is given. Under the assumption that different time series are conditionally independent given the underlying parameters, we can write the likelihood of D as

$$P(D|\Theta) = \prod_{i=1}^{N} P(x_i|\Theta) \tag{3}$$

Model parameter learning is equivalent to finding the MAP estimate i.e.

$$\hat{\Theta} = \arg\max_{\Theta}(P(D|\Theta)P(\Theta)) \tag{4}$$

If we take a uniform prior the objective turns into maximum likelihood estimation (MLE)

$$\hat{\Theta} = \arg\max_{\Theta} P(D|\Theta) \tag{5}$$

We will solve the MLE objective using EM algorithm.

4.3 EM Algorithm for ARMA mixtures

In the context of ARMA mixture modeling for clustering, the latent data corresponds to the cluster id. The log-likelihood $l(\Theta, D)$ can be written as

$$l(\Theta, D) = \sum_{i=1}^{N} \ln P(\mathbf{x}_i|\omega_{z_i,\Phi_{z_i}}) + \sum_{i=1}^{N} \ln P(\omega_{z_i}) \tag{6}$$

Given the observed data D and the current parameter estimates $\Theta(t)$, the expected complete log data likelihood becomes

$$Q(\Theta|\Theta(t)) = \sum_{i=1}^{N}\sum_{k=1}^{M} P(\omega_k|\mathbf{x}_i, \Theta(t)) \ln P(x_i|\omega_k, \Phi_k) + \sum_{i=1}^{N}\sum_{k=1}^{M} P(\omega_k|\mathbf{x}_i, \Theta(t)) \ln P(\omega_k) \quad (7)$$

where the posterior probabilities can be computed as

$$P(\omega_k|\mathbf{x}_i, \Theta) = \frac{P(\mathbf{x}_i|\omega_k, \Phi_k)P(\omega_k)}{\sum_{u=1}^{M} P(\mathbf{x}_i|\omega_u, \Phi_u)P(\omega_u)} \qquad i = 1, 2, ..., N \quad and k = 1, 2, ..., M \quad (8)$$

The EM algorithm interactively maximizes the function $Q(\Theta|\Theta(t))$ until convergence. At each iteration we compute the posterior $P(\omega_k|\mathbf{x}_i, \Theta(t))$ and given the current parameter estimates $\Theta(t)$ in the E-step and update the parameter estimate by maximizing the expected complete log data likelihood $Q(\Theta|\Theta(t))$ over Θ in the M-step. The algorithm can be summarized as follows:

- **initialize** Θ, $t \leftarrow 0$
- **do** $t \leftarrow t + 1$
  - **E-Step**: compute posterior probabilities $P(\omega_k|\mathbf{x}_i, \Theta(t))$ and $Q(\Theta|\Theta(t))$ using the current parameter estimates
  - **M-Step**: $\Theta(t+1) \leftarrow \arg\max_{\Theta} Q(\Theta|\Theta(t))$
- **until** some convergence condition is satisfied
- return $\Theta(t+1)$

One of the possible convergence condition would be to check the difference in the log-likelihood between two time-steps. Another possibility can be to observe the change in the estimates of Θ between two time steps.

The M-Step update equations are as follows -

$$\begin{aligned}
\hat{P}(\omega_k) &= \frac{1}{N}\sum_{i=1}^{N} P(\omega_k|x_i, \mathbf{\Theta}(t)) \\
\hat{\sigma_k}^2 &= \frac{\sum_{i=1}^{N}\left[P(\omega_k|x_i, \mathbf{\Theta}(t))\sum_{i=1}^{n} e_{i,t}^2\right]}{\sum_{i=1}^{N}\left[nP(\omega_k|x_i, \mathbf{\Theta}(t))\right]} \\
\hat{\delta_k} &= \left(\hat{\mathbf{W}}_k^{-1}\right)\hat{\mathbf{U}}_k \quad \text{where} \\
\hat{\delta_k} &= (\phi_{k,0}, \phi_{k,1}, \cdots, \phi_{k,p}, \theta_{k,1}, \cdots, \theta_{k,q})^T \\
\hat{\mathbf{W}}_k &= \sum_{i=1}^{N}\left[P(\omega_k|x_i, \mathbf{\Theta}(t))\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix}\right] \\
\hat{\mathbf{U}}_k &= \sum_{i=1}^{N}\left[P(\omega_k|x_i, \mathbf{\Theta}(t))(a_0, a_1, a_2, \cdots, a_p, c_1, c_2, \cdots, c_q)^T\right]
\end{aligned}$$

4.4 Model selection using BIC

The learning problem discussed in the previous section assumes that the model has already been selected, ie, the number of clusters has already been specified by the user before clustering is preformed. However, in may real-world problems, the actual model size is unknown. We have to select the most appropriate model (size) for clustering problems. Two families of model selection methods in common use:

1. Cross Validation

2. Bayesian model selection (BIC)

The Bayesian approach to model selection is to compute the posterior model probabilities of all possible models in the model space, and to select the model with the highest posterior probability. BIC is one such approach. The BIC, which is an approximation of the Bayes factor, is based on the maximized log-likelihood minus a penalty term to estimate the posterior model probability quickly and efficiently. The BIC takes the form

$$BIC = \log P\left(\mathbf{D}|\hat{\mathbf{\Theta}}\right) - \frac{1}{2}V \log N$$

where $\hat{\mathbf{\Theta}}$ is the MLE parameter of the model and V is the number of independent parameters to be estimated in the model. The first term is the maximized log-likelihood which tends to favor larger models with more parameters, while the second is the penalty term which favors smaller models with less parameters. **The BIC criterion tries to strike a balance between the simplicity of a model and its fit to the data**. The larger the BIC value, the better the model.
Under our framework, the best mixture model for clustering has the maximum marginal likelihood probability $P(\mathbf{D}|\mathbf{\Theta})$. Given a partition with M clusters, the BIC criterion is expressed as

$$BIC = \sum_{i=1}^{N} \log \left[\sum_{k=1}^{M} P\left(\mathbf{x}_i|\omega_k, \hat{\mathbf{\Phi}}_k\right) \hat{P}(\omega_k) \right] - \frac{1}{2}(M\nu + M - 1) \log N$$

where ν is the number of parameters in each component model.

5 Experiments

We generate the dataset from 3 ARMA(2, 1) models of same variance. The specifications are given in Table 1. To check if our simulation code is correct, we fit the simulated data with statsmodels.tsa models and compare the obtained coefficients. For each ARMA model, we generate 10 time series with 500 points each, thus obtaining 30 time series sequences to be clustered.

Model	ϕ_1	ϕ_2	θ_1	σ^2
ARMA(2, 1)$_1$	-0.05	0.52	0.44	0.27
ARMA(2, 1)$_2$	0.36	0.10	0.06	0.27
ARMA(2, 1)$_3$	0.34	0.27	-0.25	0.27

Table 1: Model specification for data generation

Next, we implement the EM algorithm for $T = 20$ iterations. In the E-Step, we use the log-sum-exp trick to ensure the values are bounded. We consider the case where the number of clusters are unknown. To determine the number of clusters and the appropriate ARMA models, we compare the BIC values. Table 2 shows the *normalized* BIC values we obtained.

For higher values of cluster number K and (p, q), the algorithm took a lot of time to converge, or there were cases of formation of singular matrix in the M-step. Hence, we resorted to only the hyperparameters as shown in Table 2.

Params	(1, 1)	(1, 2)	(2, 1)	(3, 1)	(4, 1)
1	0.27383783	0.	0.61050917	0.71063051	0.70347159
2	0.52299774	0.14770012	0.91758181	0.98996471	0.90732037
3	0.57297075	0.20764925	0.95114377	1.	0.98725254
4	0.40846709	0.087751	0.83125219	0.85980297	0.82738821
5	0.45894073	0.02780187	0.79863557	0.79017958	0.74745604

Table 2: Normalized BIC Values - each row corresponds to k clusters as indicated by row number; the column name refers to (p, q) values for ARMA model mixture

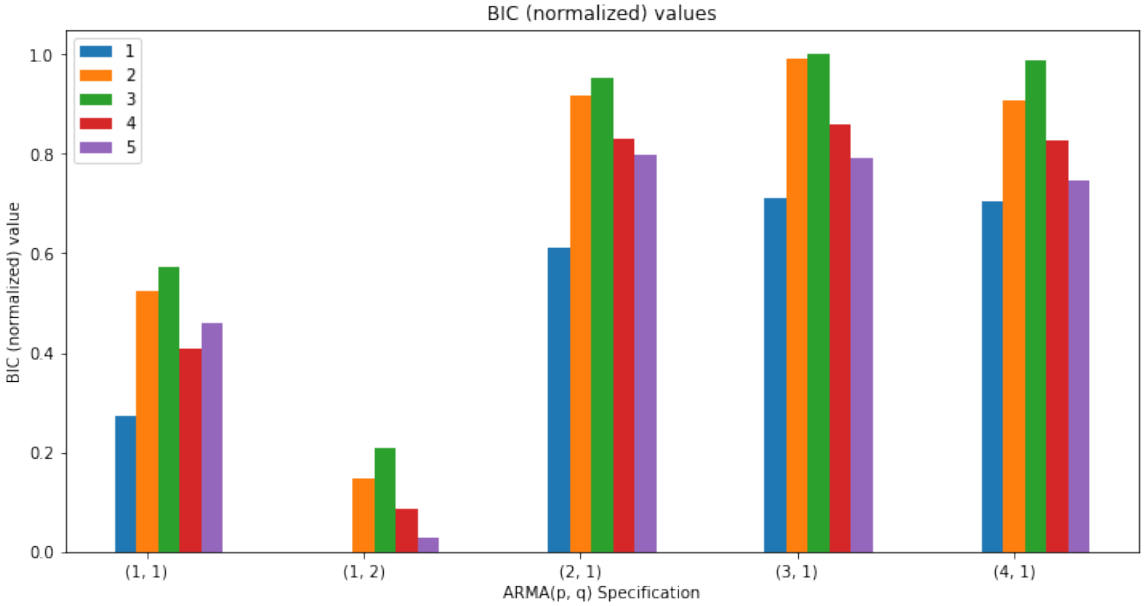

Figure 1: BIC normalized values for various cluster numbers and parameter numbers (p, q)

From Figure 1, , we can see that for K = 3, the BIC values are higher. This holds good with the fact that our data has been generated from 3 ARMA models.

While (3, 1) and (4, 1) seem to be on the higher end of BIC values, it can be attributed to overfitting (ie, larger number of parameters). Note that (2, 1) has almost comparable values as that of (3, 1) showing the efficacy of our work.

Some ways to improve our current experimental results -

1. Better **initialization**: We have initialized the $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ values using standard normal distribution. However, there are better ways to initialize. One such method followed in the paper is to use Stochastic EM Algorithm to obtain an initial set of values for $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, and then apply the general EM algorithm.

2. Use **k-means**: Apply k-means clustering to obtain an initial set of cluster means, which can be used as initializations for general EM algorithm.

5.1 The Specific Case of K = 3, (p, q) = (2, 1)

In this subsection, we look into the specific case of K = 3, (p, q) = (2, 1), since this is how the data was actually generated. We measure the cluster similarity, and look at the graph of complete log-likelihood. We also compare the estimated parameter values with that of the actual parameter values.

1. **Cluster Similarity**: We use the cluster similarity formulation as defined in the paper, given by

$$sim\left(G, A\right) = \frac{1}{K}\sum_{i=1}^{K} \max_{1\leq j\leq K} sim\left(G_i, A_j\right)$$
$$sim\left(G_i, A_j\right) = \frac{2|G_i \cap A_j|}{|G_i| + |A_j|}$$

   where $G_1, G_2, \cdots, G_K$ are the ground truth clusters, and $A_1, A_2, \cdots, A_K$ are the clusters obtained from algorithm. Note that, due to label switching, A_1 of obtained clusters need not correspond to G_1 of ground-truth clusters. So, to account for this exchangeability, we take a max in the similarity formula. In our experiments, we obtained a cluster similarity of 0.624 for the specific case in discussion. A reason for this somewhat low value is that our algorithm assigns very low probability to one of the clusters, effectively resulting in only two clusters - $[\hat{P}\left(\omega_1\right), \hat{P}\left(\omega_2\right), \hat{P}\left(\omega_3\right)] = [0.63, 0.033, 0.33]$

2. **Complete-log-likelihood**: From Figure 2, we can see that the CLL is always increasing. This can be used to track the progress of EM algorithm. We can also determine the number of iterations required by looking at this graph, as in, if CLL plateaus beyond a point, we need not iterate beyond that point.

3. **Estimated parameters**: Note that while simulating the data we took $\phi_0 = 0$. However, while estimating, we do obtain values for ϕ_0 albeit very small. The estimated values of σ^2 match very well with those of the actual values. Comparing Table 1 and 3, it seems that the third cluster values are quite close to ARMA(2, 1)$_3$, while the first cluster values are close to ARMA(2, 1)$_2$. The poor overlap of estimated model parameters can be attributed to the fact that EM algorithm converges to a local maxima and heavily depends on how good the initialization. In our experiments, we have not taken any particular care to initialize well. Better initializations are bound to yield much significant overlap of parameter values.

$\hat{\phi_0}$	$\hat{\phi_1}$	$\hat{\phi_2}$	$\hat{\theta_1}$	$\hat{\sigma}^2$
0.007	0.536	0.185	-0.19	0.27
0.012	0.508	0.452	-0.346	0.288
-0.005	0.46	0.26	-0.36	0.264

Table 3: Estimated parameters

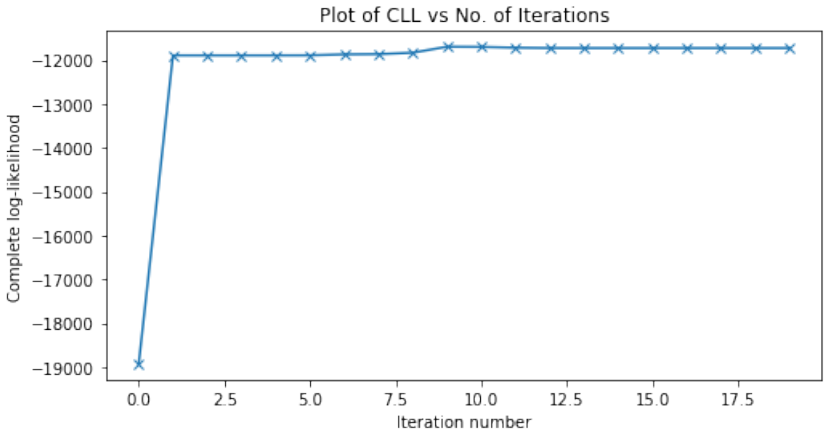

Figure 2: Plot of Complete log-likelihood vs no. of iterations