IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 01-05
www.iosrjournals.org

Periodically Data Mining (Business) Use Auto-correlation

Monali L.Gaidhane

Gaidhane_monali.ghrcemtechcse@raisoni.net , Department of Computer Science and Engineering , G.H. Raisoni College of Engg(Autonomous Institute), Nagpur, India

Abstract: *The rapid growth in data and databases increased a need of powerful data mining technique that will guide to analyze, forecast and predict behaviour of events. Periodicity mining needs to give more attention as its increased need in real life applications. In this paper, we are going to discuss on various periodicity mining techniques in Time Databases. Here, we propose a periodicity mining technique that will detect various periodic patterns.*

Keywords: *Time Series Database, Periodic Pattern, Periodicity Mining.*

I. INTRODUCTION

The explosive growth in data and databases has generated an urgent need for new techniques and tools that can intelligently and automatically transform the processed data into useful information and knowledge. The Data Mining techniques provide an intelligent solution by discovering implicit and meaningful knowledge which can be used for further development of various applications of real life such as marketing, stock market, supermarket etc. The data mining can be described as the process of discovering patterns or trends in data . TIME series data captures the evolution of a data value over time. Life includes several examples of time series data. Examples are meteorological data containing several measurements, e.g., temperature and humidity, stock prices depicted in financial market, power consumption data reported in energy companies, and event logs monitored in computer networks. Periodicity mining is a tool that helps in predicting the behavior of time series data.

Nowadays, we have different types of databases which can be distinguish on the basis of stored data like Transactional data, Legacy data, Time Series data, Spatial data, Multimedia Data, Temporal Data, Spatiotemporal data, relational data etc. In this paper we present a review on the periodicity mining techniques in Time Series Data. Time Series Database is used to store sequence of events which is obtained over repeated measurement of time such as hourly measurements, daily measurements and weekly measurements.

1.1. Background

1.1.1. Support vector machine

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis . Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier . An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser , Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.

1.1.2. Motivation

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a *new* data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of $p$numbers), and we want to know whether we can separate such points with a (p-1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the *maximum-margin hyperplane* and the linear classifier it defines is known as a *maximum margin classifier*; or equivalently, the *perceptron of optimal stability.*

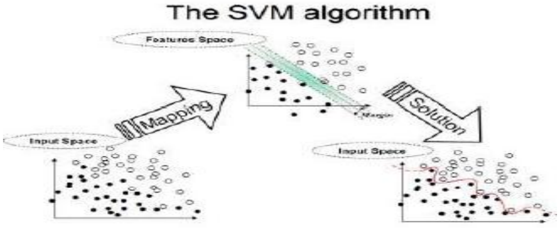


Fig. 1

1.1.3. Equations

We are given a training dataset of $n$points of the form

$$(\vec{x}_1, y_1),\ \ldots,\ (\vec{x}_n, y_n)$$

Where, the $y_i$are either 1 or −1, each indicating the class to which the point $\vec{x}_i$belongs. Each $\vec{x}_i$is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0,$$

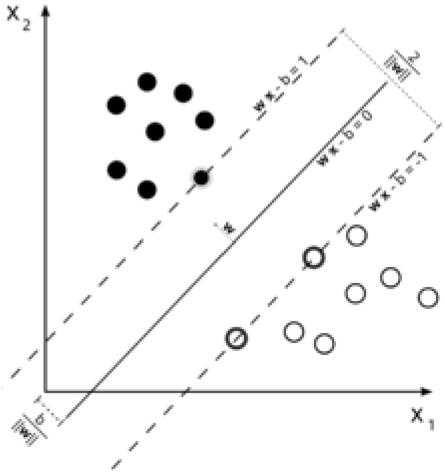


Fig. 2

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

Where $\vec{w}$is the (not necessarily normalized) normal vector to the hyperplane. The parameter $\tfrac{b}{\|\vec{w}\|}$determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$.

1.2. Computing the SVM classifier

Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i + b)\right)\right] + \lambda\|w\|^2. \qquad (2)$$

We focus on the soft-margin classifier since, as noted above, choosing a sufficiently small value for λyields the hard-margin classifier for linearly classifiable input data. The classical approach, which involves reducing (2) to a quadratic programing problem, is detailed below. Then, more recent approaches such as sub-gradient descent and coordinate descent will be discussed.

1.2.1. Primal

Minimizing (2) can be rewritten as a constrained optimization problem with a differentiable objective function in the following way.

For each $i \in \{1, \ldots, n\}$we introduce the variable ζ_i, and note that $\zeta_i = \max(0, 1 - y_i(w \cdot x_i + b))$if and only if $\zeta_i$is the smallest nonnegative number satisfying $y_i(w \cdot x_i + b) \geq 1 - \zeta_i$.

Thus we can rewrite the optimization problem as follows

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n} \zeta_i + \lambda \|w\|^2$$

$$\text{subject to } y_i(x_i \cdot w + b) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.$$

This is called the *primal* problem.

1) Dual

By solving for the Lagrangian dual of the above problem, one obtains the simplified problem

$$\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j,$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

This is called the *dual* problem. Since the dual minimization problem is a quadratic function of the $c_i$subject to linear constraints, it is efficiently solvable by quadratic programming algorithms. Here, the variables $c_i$are defined such that

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i$$

.

Moreover, $c_i = 0$exactly when $\vec{x}_i$lies on the correct side of the margin, and $0 < c_i < (2n\lambda)^{-1}$ when $\vec{x}_i$lies on the margin's boundary. It follows that $\vec{w}$can be written as a linear combination of the support vectors. The offset, b, can be recovered by finding an $\vec{x}_i$on the margin's boundary and solving

$$y_i(\vec{w} \cdot \vec{x}_i + b) = 1 \iff b = y_i - \vec{w} \cdot \vec{x}_i.$$

1.2.2. Kernel trick

Suppose now that we would like to learn a nonlinear classification rule which corresponds to a linear classification rule for the transformed data points $\varphi(\vec{x}_i)$.Moreover, we are given a kernel function $k$which satisfies $k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$.

We know the classification vector $\vec{w}$in the transformed space satisfies

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \varphi(\vec{x}_i),$$

where the $c_i$are obtained by solving the optimization problem

$$\begin{aligned}\text{maximize } f(c_1 \ldots c_n) &= \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (\varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)) y_j c_j \\ &= \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i k(\vec{x}_i, \vec{x}_j) y_j c_j\end{aligned}$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

The coefficients $c_i$can be solved for using quadratic programming, as before. Again, we can find some index i such that $0 < c_i < (2n\lambda)^{-1}$, so that $\varphi(\vec{x}_i)$lies on the boundary of the margin in the transformed space, and then solve

$$b = \vec{w} \cdot \varphi(\vec{x}_i) - y_i = \left[\sum_{k=1}^{n} c_k y_k \varphi(\vec{x}_k) \cdot \varphi(\vec{x}_i) \right] - y_i$$
$$= \left[\sum_{k=1}^{n} c_k y_k k(\vec{x}_k, \vec{x}_i) \right] - y_i.$$

Finally, new points can be classified by computing

$$\vec{z} \mapsto \operatorname{sgn}(\vec{w} \cdot \varphi(\vec{z}) + b) = \operatorname{sgn}\left(\left[\sum_{i=1}^{n} c_i y_i k(\vec{x}_i, \vec{z}) \right] + b \right).$$

II. AUTOCORRELATION TECHNIQUE

Autocorrelation refers to the correlation of a time series with its own past and future values. autocorrelation is also sometimes called "lagged correlation" or "serial correlation", which refers to the correlation between members of a series of numbers arranged in time. positive autocorrelation might be considered a specific form of "persistence", a tendency for a system to remain in the same state from one observation to the next. For example, the likelihood of tomorrow being rainy is greater if today is rainy than if today is dry. geophysical time series are frequently autocorrelated because of inertia or carryover processes in the physical system. For example, the slowly evolving and moving low pressure systems in the atmosphere might impart persistence to daily rainfall. or the slow drainage of groundwater reserves might impart correlation to successive annual flows of a river or stored photo syntheses might impart correlation to successive annual values of tree-ring indices. autocorrelation complicates the application of statistical tests by reducing the number of independent observations. autocorrelation can also complicate the identification of significant covariance or correlation between time series (e.g., precipitation with a tree-ring series). autocorrelation can be exploited for predictions: an autocorrelated time series is predictable, probabilistically, because future values depend on current and past values. three tools for assessing the autocorrelation of a time series are (1) the time series plot, (2) the lagged scatterplot, and (3) the autocorrelation function.

2.1. Equations

The auto-correlation function measures the correlation of a signal *x(t) with itself shifted by some time delay τ:*

$$C(\tau) = \frac{1}{t-\tau} \int_0^{t-\tau} x(t) x(t+\tau) dt$$

The auto-correlation function can be used to detect repeats or periodicity in a signal. Here, we use the auto-correlation to assess the effect of fluctuations(noise) on a periodic signal.

Calculating Accuracy:

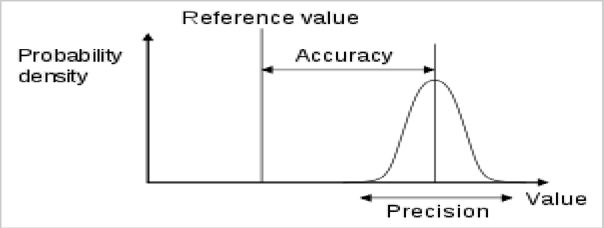
Accuracy is often the starting point for analyzing the quality of a predictive model, as well as an obvious criterion for prediction.

Accuracy measures the ratio of correct predictions to the total number of cases evaluate.

Accuracy is also used as a statistical measure of how well a binary classification test correctly identifies or

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total of all cases to be predicted}}$$

excludes a condition.



Accuracy is the proximity of measurement results to the true value; precision, the repeatability, or reproducibility of the measurement

Fig. 3

III. Conclusion

In this paper, we have discussed different algorithms that can detect periodicity in time series database. Some authors proposed separate algorithm for different periodicity. Again we can approach to make it more generalized for different inputs. The periodicity detection in time series play an important role in many application. We performed several experiments to show the time behavior, accuracy, and noise resilience characteristics of the data. We run the algorithm on both real and synthetic data. The reported results demonstrated the power of the employed pruning strategies.

References

[1]. Faraz Rasheed, Mohammed Alshalalfa, and Reda Alhajj, "Efficient Periodicity Mining in Time Series Databases Using Suffix Trees",*Knowledge And Data Engineering, Vol. 23, No. 1,* January 2015

[2]. M.G. Elfeky, W.G. Aref, and A.K. Elmagarmid, "Periodicity Detection in Time Series Databases," *Knowledge and Data Eng., vol. 17, no. 7, pp. 875-887*, July 2012.

[3]. M.G. Elfeky, W.G. Aref, and A.K. Elmagarmid, "WARP: Time Warping for Periodicity Detection", *Proc. Fifth IEEE Int'l Conf. Data Mining, Nov.*, 2015.

[4]. Amruta Mahatre, Mridula Verma, Durga Toshniwal, "Privacy Preserving Sequential Pattern In p rogressive databases using Noisy Data", 13th International Conference Information Visualization, 2011

[5]. Kuo-Yu Huang and Chia-Hui Chang, Member, IEEE Computer Society, " SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temp oral Databases",*Knowledge and Data Eng., vol. 17*, no. 6, June 2013.

[6]. Anita Sant'Anna, Nicholas Wickstr¨om, "Symbolization of time-series: An evaluation of SAX, Persist, and ACA", 4th International Conf. on Image and Signal Processing, 2011

[7]. Yi Jiang, Tuo Lan, Dongzhan Zhang, "A New Representation and Similarity Measure of Time Series on Data Mining", *Similarity Search in Sequence Databases. In Proc. Of the 4th International conference Foundations of Data Organization and Algorithms, Chicago, Illinois*, October 14, 2014

[8]. R. Agrawal, K. Lin, H. Sawhney, and K. Shim. Fast," Similarity Search in the Presence of Noise, Scaling, and Translation in Time Series Databases". *In Proc. of the 21st Int. Conf. on Very Large Databases*, September 2005.

[9]. R. Agrawal, G. Psaila, E. Wimmers, and M. Zait. Querying Shapes of Histories. In proceedings VC'1l Large Databases, Zurich, Switzerland, September 2011.

[10]. R. Agrawal and R. Srikant, " Fast Algorithms for Mining Association Rules", *proceedings 20th International Conference Very Large Databases, Santiago, Chile*, September 2014.