Journal of Big Data

RESEARCH Open Access

A graph-based big data optimization approach using hidden Markov model and constraint satisfaction problem

Imad Sassi*, Samir Anter and Abdelkrim Bekkhoucha

*Correspondence:
imadsassi7@gmail.com
Computer Science Laboratory (LIM), FSTM, Hassan II University, Casablanca, Morocco

Abstract
To address the challenges of big data analytics, several works have focused on big data optimization using metaheuristics. The constraint satisfaction problem (CSP) is a fundamental concept of metaheuristics that has shown great efficiency in several fields. Hidden Markov models (HMMs) are powerful machine learning algorithms that are applied especially frequently in time series analysis. However, one issue in forecasting time series using HMMs is how to reduce the search space (state and observation space). To address this issue, we propose a graph-based big data optimization approach using a CSP to enhance the results of learning and prediction tasks of HMMs. This approach takes full advantage of both HMMs, with the richness of their algorithms, and CSPs, with their many powerful and efficient solver algorithms. To verify the validity of the model, the proposed approach is evaluated on real-world data using the mean absolute percentage error (MAPE) and other metrics as measures of the prediction accuracy. The conducted experiments show that the proposed model outperforms the conventional model. It reduces the MAPE by 0.71% and offers a particularly good trade-off between computational costs and the quality of results for large datasets. It is also competitive with benchmark models in terms of the running time and prediction accuracy. Further comparisons substantiate these experimental findings.

Keywords: Machine learning, Big data analytics, Optimization, Metaheuristics, Time series forecasting, Graphical modeling

Introduction

Big data refers to huge amount of heterogeneous data. Due to the growing number of connected objects, the massive use of social networks and the advent of new generations of mobile technologies, the volume of data generated has increased at an exponential rate [1].

Big data has meaning and value when it is possible to derive relevant information from the data and discover hidden links and correlations. To process a large amount of data, there are analytical application tools that can also be called big data analytics [2]. One of the most powerful solutions in big data analytics is machine learning [3], which can be used to handle a large amount of data [4].

Since the beginning of the era of big data, and with the increasing multiplicity and diversity of data sources (social networks, log files, sensors, the IoT, mobile objects, etc.), multiple challenges have emerged [5]. These challenges are related to the complex characteristics of big data. The focus here is on the volume of data, which causes a great storage problem, and the variety of data, which complicates the operation of collecting data of different types that have heterogeneous formats (structured, unstructured, semistructured), without neglecting the speed of the generation, collection, processing/analysis and sharing of data [6].

Currently, the growing interest in big data applications requiring the processing of large amounts of heterogeneous data is bringing new opportunities to apply new optimization approaches [7, 8]. Big data optimization concerns the high dimensionality of data, dynamic changes in data and multiobjective problems and algorithms.

In machine learning, optimization algorithms are widely used to analyze large volumes of data and to calculate parameters of models used for prediction or classification [9]. Indeed, optimization plays an important role in the development of new approaches to solve machine learning problems, thanks to the high efficiency of optimization solutions and the multitude of applications that can be formulated as an optimization problem. Although optimization algorithms are effective in many application areas, the complex characteristics of big data, the size of the state space, and the variety of learning models require new optimization techniques and powerful methods capable of dealing with optimization problems that cannot be addressed today [10].

In recent years, metaheuristic algorithms have been frequently used in various data mining problems due to their ability to find optimal solutions for problems of reasonable size [11]. Furthermore, to deal with large optimization problems, metaheuristics constitute a very interesting alternative when optimality is not essential. Metaheuristics are an indispensable approach for difficult and complex optimization problems guaranteeing an equilibrium between the quality of the solutions and the computation time. However, despite the progress made, particularly in terms of the computation time, many metaheuristic algorithms are less efficient when dealing with large-scale problems. Thus, the application of metaheuristics to big data analytics problems is a challenging topic that attracts the attention of many researchers; thus, the study of these methods is currently in full development [12].

Some powerful machine learning algorithms are hidden Markov models (HMMs), which are commonly used in several machine learning problems [13]. HMMs have been applied successfully to speech recognition [14], face detection [15], bioinformatics [16], finance analysis [17], etc. The use of HMMs for big data applications (i.e., a high number of states and a high number of observations) is growing rapidly, which explains the focus of researchers on new methods of adapting HMMs to the big data context and improving their performance.

In this paper, we propose a new big data optimization method based on the use of the constraint satisfaction problem, one of the fundamental concepts of metaheuristics [18]. It is a CSP graph-based approach that is put into practice to reduce the state space of hidden Markov models to improve learning and prediction tasks using HMMs. In this approach, HMMs are treated as a CSP but are not limited to a formalism (i.e., constrained HMMs, as in some works), which explains the use of CSP solver algorithms to solve such problems. Furthermore, this concept can be applied not only to states or observations but also to both at the same time, unlike in other works. In addition, this approach is specifically designed for big data, but it is also suited to small standard data.

The main contributions of this work are as follows:

- The phenomenon of big data and the emergence of big data analytics are introduced, supporting the need for new machine learning algorithms to take advantage of this huge amount of data.
- A general overview of related works that focus on the application of machine learning to financial time series is provided; in particular, the use of HMMs is discussed, followed by the main approaches for the optimization of HMMs.
- We propose a new big data optimization method based on the use of the constraint satisfaction problem, consisting of a graph-based approach to enhance the learning and prediction tasks using HMMs.
- We experimentally evaluate the proposed approach on a real-world dataset and compare it to the conventional HMM and to the reference models using the complexity, running time, and MAPE and other metrics as measures of the prediction accuracy.

The remainder of the paper is organized as follows: "Problem formulation" section clarifies the problem statements. "Related work" section offers a general overview of recent related works. "Background" section provides some required background knowledge and establishes some basic notation used in HMM theory and then discusses big data optimization techniques and fundamental metaheuristics concepts. The proposed approach is presented in "Research methodology" section. In "Experiments and results" section, we describe the experiments and results for evaluating the proposed approach. Finally, in "Conclusion and future directions" section, we give conclusions, and we present some directions for future work in this area.

Problem formulation

The use of HMMs in financial time series applications faces a range of challenges. Researchers have been working on the main problems in connection with HMMs. Thus, many works have focused on the improvement of existing approaches or the search for new solutions to the prediction problem (using the Viterbi algorithm [19]) or the evaluation problem (using forward-backward or Viterbi training). The proposed approaches aim to find solutions to four problems: (1) the choice of model topology (e.g., the number of hidden states and observations and type of connections); (2) the search for the initial parameters; (3) the search for the model parameters; and (4) the reduction of the search space. Despite the progress made, these works nevertheless face multiple obstacles that hamper their speed and efficiency.

In the real world, a main challenge for hidden Markov model problems (e.g., stock market prediction) is the high dimensionality of the state space (N) and/or observation space (M). The objective is to provide a solution quickly enough that the system can give a result in a reasonable time without losing accuracy.

This paper addresses this problem by developing a complete approach, which is inspired by heuristic methods, that can provide solutions to a given big data problem.

The objective of this big data optimization approach is to reduce the state and/or the observation space so that this approach can be used in a big data context. The approach is based on treating the problem of dimensionality optimization as a constraint optimization problem, thus taking advantage of the power of CSP solvers, by giving a solution with a quality depending on the time allocated for computation. It consists of using an AC3 [20] variant of the arc consistency algorithm with added external constraints and a backtracking method for solving the CSP to:

- reject the nodes that cannot be reached under certain constraints and thereby reduce the state space dimension.
- delete unnecessary arcs between nodes and thereby reduce the transition probability matrix.
- speed up the learning process using the Baum-Welch algorithm while maintaining a very high level of accuracy.

Related work

There is a vast amount of published research involving the application of machine learning and deep learning techniques to related problems of time series [21], especially financial time series analysis. For example, [22–25] investigated the application of deep learning for stock market prediction. Many approaches based on the use of the support vector machine technique have also been proposed for stock price forecasting or stock trend prediction [26–29]. In some articles, artificial neural networks have been applied for stock price prediction in combination with genetic algorithms or metaheuristics [30–33]. [34–36] studied the implementation of dimensionality reduction techniques for forecasting the stock market. [37] presented an evaluation of ensemble learning techniques for stock market prediction. It is also noted that the graph-based technique [38] is an approach that has found a place in this field.

In comparison, there has been relatively little research focusing on applying HMMs to financial scenarios. In their work, the authors of [39] build a dynamic asset allocation (DAA) system using HMMs for regime detection. Then, they extend the DAA system by incorporating a feature saliency HMM algorithm that performs feature selection simultaneously with the training of the HMM to improve regime identification. Experiments across multiple combinations of smart beta strategies and the resulting portfolios show an improvement in risk-adjusted returns. Another work, [40], used an HMM to predict economic regimes, on the basis of which global stocks are evaluated and selected for optimizing the portfolio. In this study, they established a multistep procedure for using an HMM to select stocks from the global stock market. The results showed that global stock trading based on the HMM outperformed trading portfolios based on all stocks in ACWI, a single stock factor, or the equal-weighted method of five stock factors. Finally, [41] studied the use of HMMs to model the market situation, performed feature analysis on the hidden state of the model input, estimated the market situation, and proposed the Markov situation estimation trading strategy. The experimental data showed that the hidden Markov model estimates the trading strategy's overall return and is better than the double moving average strategy.

Additionally, different techniques and approaches have been studied with the aim of optimizing HMMs to adapt to the complexity of big data characteristics so that they can be used in different applications that deal with large amounts of heterogeneous and multisource data.

[42] demonstrates that it is possible to improve the training of HMMs by applying a constrained Baum–Welch algorithm along with a model selection scheme that imposes constraints on the possible hidden state paths in calculating the expectation. This proposed method has two main advantages. First, it enables the partial labels to be leveraged in the training sequences, thus increasing the log-likelihood of the given training sequences. Second, in each iteration of the constrained Baum–Welch algorithm, the decoding accuracy for the partially labeled training sequence can be calculated and factored into model selection.

In [43], the maximum mutual information (MMI) criterion combined with 4-fold cross-validation was used to optimize HMM hyperparameters. This paper further explored the use of a hand movement recognition framework based on an ergodic HMM that can model myoelectric activity transitions with multichannel sEMG signals. The experimental results showed that using MMI as the optimization criterion for hyperparameters significantly improved the average recognition accuracy.

[44] introduced the use of a genetic algorithm to optimize the parameters of an HMM and used the improved HMM in the identification and diagnosis of photovoltaic (PV) inverter faults. Thus, a genetic algorithm was used to optimize the initial value and to achieve global optimization. The experimental results showed that the correct PV inverter fault recognition rate achieved by the HMM was approximately 10% higher than that of traditional methods. Using the GHMM, the correct recognition rate was further increased by approximately 13%, and the diagnosis time was greatly reduced.

In recent years, another work of Bravzenas et al. [45], proposed three different EM-based fitting procedures that can take advantage of parallel hardware, such as graphics processing units, to reduce the computational complexity of fitting Markov arrival processes with the expectation-maximization (EM) algorithm. The performance evaluation showed that the proposed algorithms are orders of magnitude faster than the standard serial procedure.

In our recent paper [46], we presented two newly improved algorithms for Gaussian continuous HMMs and a mixture of Gaussian continuous HMMs for solving the learning problem for large-scale multidimensional data. These are parallel distributed versions of classical algorithms based on Spark as the main framework. The proposed solution enables the management of heterogeneous data in real time. The algorithms presented in this paper have two main advantages, a high computational time efficiency and a high scalability, since it is possible to add several nodes in a very simple way. In addition, this solution is easy to integrate into big data frameworks.

However, these alternatives present many disadvantages. In some proposed solutions, for higher accuracy, it is necessary to choose an HMM with the correct topology and apply the approach to data with special characteristics. An important issue is that the accuracy varies from one application to another because the efficiency of the final solution is strongly affected by the choice of the initial parameters. Other approaches with independent feature extraction algorithms have a major drawback;

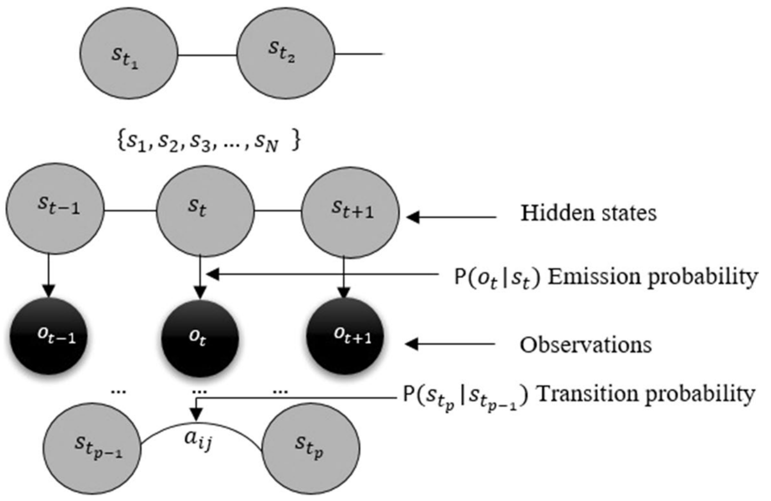
their optimization criteria could lead to inconsistency between feature extraction and the classification steps of a pattern recognition tool and thereby degrade the performance of the classifiers. Moreover, from the viewpoint of the constrained optimization problem, the optimization criteria do not consider the intrinsic and extrinsic constraints and are limited to the constraints induced by repeated experiments. Another disadvantage of the preceding solutions is that the required computation time is often prohibitive in practice and the learning time can increase considerably. The accuracy is certainly improved, but the complexity increases considerably, and in the best case, it keeps the same values as those of the classic model. For some solutions, the running time increases dramatically. In addition, the application areas of most of the proposed implementations are limited because they perform better only in specific use cases. Generally, the previous approaches focused only on HMM situations with a low number of states and observations, but they did not give any feasible solution for the estimation of HMM parameters with a high number of states or observations. The optimization of the hyperparameters primarily involves choosing the number of hidden states and the number of observations.

Background

Hidden Markov models

A hidden Markov model is a directed graph $\langle S, A\rangle$ with vertices representing states $S = \{s_1, s_2, ..., s_N\}$ and arcs $A = \{\langle i,j\rangle \mid s_i, s_j \in S\}$ showing transitions between states (see Fig. 1). Each arc $\langle i,j\rangle$ is labeled with a probability a_{ij} of transitioning from state s_i to state s_j. At any time t, one state is designated as the current state q_t. At time t, the probability of any future transitions depends only on s_t and no other earlier states (first-order HMM).

An HMM consists of states, transitions, observations and probabilistic behavior and is formally defined by the elements presented in Table 1. We may completely specify



s_{t_1}
s_{t_2}
$\{s_1, s_2, s_3, \dots, s_N\}$
s_{t-1}
s_t
s_{t+1}
Hidden states
P($o_t \mid s_t$) Emission probability
o_{t-1}
o_t
o_{t+1}
Observations
P($s_{t_p} \mid s_{t_{p-1}}$) Transition probability
$s_{t_{p-1}}$
a_{ij}
s_{t_p}

Fig. 1 Graphical representation of a hidden Markov model. A hidden Markov model consists mainly of hidden states (vertices s_i), transition probabilities (arcs a_{ij}), observations (o_t) and emission probabilities

Table 1 Explanation of the elements of an HMM

Element	Description
λ	DHMM model, $\lambda = (A, B, \Pi)$ or CHMM model, $\lambda = (A, c_{jm}, \mu_{jm}, \Sigma_{jm}, \Pi)$.
S	The state vectors of the HMM $S = \{s_1, s_2, ..., s_N\}$, ($N$ states).
V	the observation $V = \{v_1, v_2, ..., v_M\}$, ($M$ observations).
O	The observation sequence $O = \{o_1, o_2, ..., o_T\}$.
Q	The hidden state sequence $Q = \{q_1, q_2, ..., q_T\}, q_t \in S$.
A	The transition matrix $A = \{a_{ij}\}, a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$, where $1 \leq i, j \leq N$. q_t is the state at time t. a_{ij} is the transition probability from state s_i to state s_j. For every state s_i, $\sum_{j=1}^{N} a_{ij} = 1$ and $a_{ij} \geq 0$.
B	The observation matrix $B = \{b_j(o_t)\}, b_j(o_t) = P(o_t \mid q_t = s_j)$ is the probability of the t^{th} observation, which is observed in state s_j. For continuous observation, $b_j(o_t)$ is a probability density function (pdf) or a mixture of continuous pdfs. o_t is the observation feature vector recorded at time t. $b_j(o_t) = \sum_{m=1}^{M} c_{jm} N(o_t, \mu_{jm}, \Sigma_{jm})$, where $N(O, \mu_{jm}, \Sigma_{jm}) = \sum_{m=1}^{M} c_{jm} \frac{1}{((2\pi)^d \lvert\Sigma_{jm}\rvert)^{1/2}} \exp\left(-\frac{1}{2}(o_t - \mu_{jm})\Sigma_{jm}^{-1}(o_t - \mu_{jm})^T\right)$. M is the total number of mixtures and d is the dimension of o_t. For every state s_j, $\sum_{t=1}^{T} b_j(o_t) = 1$.
Π	The stochastic initial distribution vector $\Pi = \{\pi_i\}$, where π_i is the probability of s_i being the first state of a state sequence. $\pi_i = P(q_1 = s_i), 1 \leq i \leq N$. $\sum_{i=1}^{N} \pi_i = 1$.
$P(O \mid \lambda)$	The probability that a given sequence of observations $O = \{o_1, o_2, ..., o_T\}$ are generated by a model λ with a given HMM.
$\alpha_t(i)$	forward variable, defined as $\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = s_i \mid \lambda)$.
$\beta_t(i)$	Backward variable, defined as $\beta_t(i) = P(o_{t+1} o_{t+2} ... o_T \mid q_t = s_i, \lambda)$.
$\gamma_t(i)$	The probability of being in state s_i at time t given λ and O. $\gamma_t(i) = P(q_t = s_i \mid O, \lambda)$.
$\xi_t(i,j)$	The probability of being in state s_i at time t and in state s_j at time $t+1$ given the model parameter λ and the observation sequence O. $\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$.
$\gamma_t(j,m)$	The probability that given the model parameter λ, the observation o_t is generated from state s_j and accounted for by the m^{th} component of the Gaussian mixture density of state s_j.
μ_{jm}	The mean of the m^{th} mixture in state s_j.
Σ_{jm}	The covariance matrix of the m^{th} mixture in state s_j.
c_{jm}	m^{th} Mixture weights in state s_j, where $\sum_{m=1}^{M} c_{jm} = 1$ and $c_{jm} \geq 0$.
$\delta_t(i)$	The likelihood score of the optimal (most likely) sequence of hidden states of length t (ending in state s_i) that produce the first t observations for the given model. $\delta_t(i) = \max_{q_1, q_2, ..., q_{t-1}} P(q_1, q_2, ..., q_{t-1}, q_t = s_i, o_1, o_2, ..., o_{t-1} \mid \lambda)$.
$\psi_t(i)$	The array of back pointers that stores the node of the incoming arc that led to this most probable path.

an HMM by its parameters $\lambda = (A, B, \Pi)$, where A is the state transition probability matrix, B is the state emission probability matrix, and Π is the initial state probability matrix.

HMMs have been applied successfully to speech recognition, bioinformatics, finance analysis, etc. They are often used in machine learning problems. The use of HMMs in classification is a generative method of first training a model (Baum-Welch algorithm [46]) (see Algorithm 1) and then, for a given observation sequence, determining which model is, among those previously established, the most likely to produce this observation sequence.

In addition, for a given model, it is possible to find the sequence of hidden states that are the most likely to produce the observation sequence in question (Viterbi algorithm) (see Algorithm 2).

```
Algorithm 1: Baum-Welch algorithm
Input:
    $\lambda$: initial model $\lambda = (A, B, \Pi)$
    $O$: a sequence of observations $O = o_1, o_2, ..., o_T$
Output:
    $\overline{A} = \{a_{ij}\}$, $\overline{B} = \{c_{jm}, \mu_{jm}, \Sigma_{jm})\}$, $\overline{\Pi} = \{\pi_i\}$: optimal parameters of a Hidden Markov Model
    to maximize the probability $P(\lambda \mid O)$
 1: Initialization;
    $\alpha_1(i) = \pi_i b_i(o_1)$, $\beta_T(i) = 1$, for $1 \leq i \leq N$
 2: repeat
 3:     Estimation-Step
 4:     Forward-Backward recursive computation;
    $\alpha_{t+1}(i) = b_j(o_{t+1}) \sum_{j=1}^{N} \alpha_t(j) a_{ij}$,
    $\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})$, for $1 \leq i \leq N$, $1 \leq t \leq T-1$
 5:     calculate $\gamma_t(i)$, $\gamma_t(j,m)$ and $\xi_t(i,j)$
    $\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$
    $\gamma_t(i) = \sum_{i=1}^{N} \xi_t(i,j)$
    $\gamma_t(j,m) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right] \left[\frac{c_{jm} N(o_t, \mu_{jm}, \Sigma_{jm})}{\sum_{m=1}^{M} c_{jm} N(o_t, \mu_{jm}, \Sigma_{jm})}\right]$ for $1 \leq j \leq N$, $1 \leq m \leq M$,
    $1 \leq t \leq T-1$
 6:     Maximization-Step
 7:     calculate optimal parameters;
    $\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$
    $\overline{\pi}_i = \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N} \alpha_1(i)\beta_1(i)}$
    $\overline{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(j,m)}$
    $\overline{\mu}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m).o_t}{\sum_{t=1}^{T} \gamma_t(j,m)}$
    $\overline{\Sigma}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m).(o_t - \mu_{jm})(o_t - \mu_{jm})^T}{\sum_{t=1}^{T} \gamma_t(j,m)}$
 8:     set $\lambda \leftarrow \overline{\lambda}$
 9: until some convergence criterion is met
10: return $\overline{A} = \{a_{ij}\}$, $\overline{B} = \{c_{jm}, \mu_{jm}, \Sigma_{jm})\}$, $\overline{\Pi} = \{\pi_i\}$
```

```
Algorithm 2: Viterbi algorithm
Input:
    $\lambda$: initial model $\lambda = (A, B, \Pi)$
    $O$: a sequence of observations $O = o_1, o_2, ..., o_T$
Output:
    $Q$: the hidden state sequence that produced the observation sequence $Q = \{q_1, q_2, ..., q_T\}$
 1: Initialization;
    $\delta_1(i) = \pi_i b_i(o_1)$, $\psi_1(i) = 0$, for $1 \leq i \leq N$
 2: Recursion;
    $\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t)$, $\psi_t(j) = \underset{1 \leq i \leq N}{argmax} [\delta_{t-1}(i) a_{ij}] b_j(o_t)$, for $1 \leq j \leq N$, $2 \leq t \leq T$
 3: Termination;
    $P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$, $q_T^* = \underset{1 \leq i \leq N}{argmax} [\delta_T(i)]$, for $1 \leq t \leq T-1$
 4: Path backtracking;
    $q_t^* = \psi_{t+1}(j)(q_{t+1}^*)$, for $t = T-1, T-2, ..., 1$
 5: return $Q$
```

HMMs make it possible to model discrete or continuous observation sequences. They solve three main problems:

- Evaluation: Given a hidden Markov model λ and an observation sequence O, find $P(\lambda|O)$, the probability that the sequence O was generated by the model λ.

- Decoding (searching for the most likely path): Given a hidden Markov model λ and an observation sequence O, find the state sequence Q that maximizes the probability of observing this sequence, $P(\lambda|O)$.
- Learning: Given a hidden Markov model λ with unspecified transition/emission probabilities and a set of observation sequences, find the parameters A and B of the hidden Markov model to maximize the probabilities of these sequences, $P(\lambda|O)$.

Big data

Features of big data

A well-known definition of big data was proposed by the International Data Corporation (IDC) as follows: "big data technologies describe a new generation of technologies and architectures, designed to economically extract the value from very large volumes of a wide variety of data, allowing a high speed of capture, discovery and/or analysis" [6].

Big data is mainly characterized by:

(1) Volume: With the digitization of our lives and the advent of the Internet of Things, the data volume continues to grow exponentially. The amount of digital data will quadruple, from 45 zettabytes in 2019 to 175 zettabytes by 2025, and 49% of data will be stored in public cloud environments (see Fig. 2). However, with this large amount of data, studies show that only a tiny fraction of the digital universe has been explored for value analysis [47].

(2) Variety: Big data are available in three types: structured, semistructured and unstructured. However, 90% of current data, from sources such as social media and other user-generated content, are unstructured.

(3) Velocity: Due to recent technological developments, speed has increased significantly. The rate of generation, collection and sharing of big data reaches very high thresholds. This high speed requires that big data be processed and analyzed at a speed corresponding to the speed of their production.

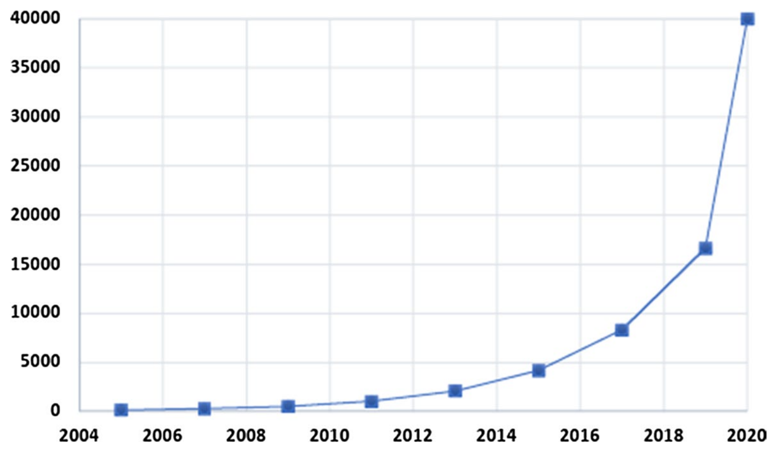


Fig. 2 Growth of digital data between 2005 and 2020 in exabytes. This figure shows the exponential increase in data volume according to the IDC

Big data challenges

In the literature, several studies have focused on big data issues [48, 49]. To solve these issues, there are many challenges that need to be addressed to improve the performance of machine learning algorithms:

- accelerating data processing and analysis as much as possible to reduce the computation time.
- developing new methods that can scale to big data and deal with heterogeneous and loosely structured data and with high computational complexity.
- performing efficient synchronization between different systems and overcoming bottlenecks at system binding points.
- proposing effective security solutions to increase the level of security and protect the privacy and information of individuals and companies, as security vulnerabilities can multiply with data proliferation.
- improving the efficiency of algorithms in managing the large amount of data storage required.
- producing real-time results by enabling real-time analysis of data flows from different sources, which gives more value to knowledge.
- improving the management and manipulation of multidimensional data to overcome the problem of incomplete or noisy information.
- reinforcing big data analytics.
- exploiting advances in computer hardware in the development of increasingly rich classes of models.
- taking full advantage of new approaches to distributed parallel processing (e.g., MapReduce, Spark [50], and GPGPUs) to effectively improve machine learning algorithms for big data [51–54].

Metaheuristics and big data optimization

Metaheuristics are approximate methods and techniques adaptable to a very large number of combinatorial problems, specifically large-scale problems [11]. A metaheuristic is a set of fundamental concepts used to solve data mining problems. They have proven to be very effective in providing good quality approximate solutions for many classical optimization problems and large real-world applications. Therefore, metaheuristics can be used for big data problems that induce complex data characteristics, i.e., data volume, data velocity, data variety, data veracity and data value.

Optimization problem

An optimization problem is defined by a set of instances. Each instance is associated with a discrete set of solutions S, where $S \neq 0$ represents the search space, a subset X of S representing the admissible (achievable) solutions and an objective function f (fitness function) which assigns to each solution $x \in X$ the real (or integer) number $f(x)$ [55].

We define an instance I of a minimization problem by a pair (X, f) where $X \subseteq S$ is a finite set of admissible solutions, and f an objective function to be minimized, defined as:

$$f : X \to \mathbb{R} \tag{1}$$

Solving this problem (more precisely this instance of the problem) consists of finding a solution $x^* \in X$ that optimizes the value of the objective function f. The problem is to find $x^* \in X$ such that for any element $x \in X$:

$$f(x^*) \leqslant f(x),\ x \in X \tag{2}$$

This solution is called an optimal solution or a global optimum.

Optimization problems vary depending on the domain in which S belongs. Thus, an optimization problem can be binary if $S \subseteq \mathbb{B}^*$, integer if $S \subseteq \mathbb{N}^*$, continuous if $S \subseteq \mathbb{R}^*$, or heterogeneous if $S \subseteq (\mathbb{B} \cup \mathbb{N} \cup \mathbb{R})^*$.

There are several types of optimization methods. To solve a combinatorial optimization problem, the choice of the method depends on its complexity. A polynomial optimization algorithm is used to solve P class problems. For NP class problems, two approaches are used: heuristic and exact approaches. The essential principle of an exact method generally consists of enumerating, often implicitly, all the solutions of the search space. Metaheuristics are powerful heuristic algorithms capable of solving combinatorial optimization problems. They are a very interesting alternative to deal with large optimization problems if optimality is not essential. They offer an acceptable solution in a reasonable computation time for complex problems.

Metaheuristics concepts

The main concepts of metaheuristics are linked to the methods that are required to solve an optimization problem.

- Representation/encoding: The encoding of possible solutions to an optimization problem is a very important concept when designing metaheuristics. For each problem, it is necessary to choose the suitable operators and optimization functions so that the encoding is efficient, and then to check if this encoding respects a set of properties to be achievable.
- Constraint satisfaction: The solution to an optimization problem can be described by assigning values to the variables of this problem and defining a set of constraints to be respected. Therefore, a solution is achievable if it respects all these constraints, which are generally difficult to formulate according to the studied problem and the chosen optimization criterion.
- Optimization criterion/objective function: To formulate a data mining task as an optimization problem, it is necessary to identify the optimization criterion and to carefully define the objective function. Correctly choosing these two basic concepts ensures the development of an efficient optimization method and guarantees a very good quality of the solutions.
- Performance analysis: Another fundamental concept that requires careful study is the performance analysis of metaheuristics. It is necessary to set the objectives of

the experiments, then to choose the appropriate performance measures and finally, depending on the purpose of the experiments, to identify and calculate the indicators in order to evaluate the quality of the solutions.

Constraint satisfaction problem

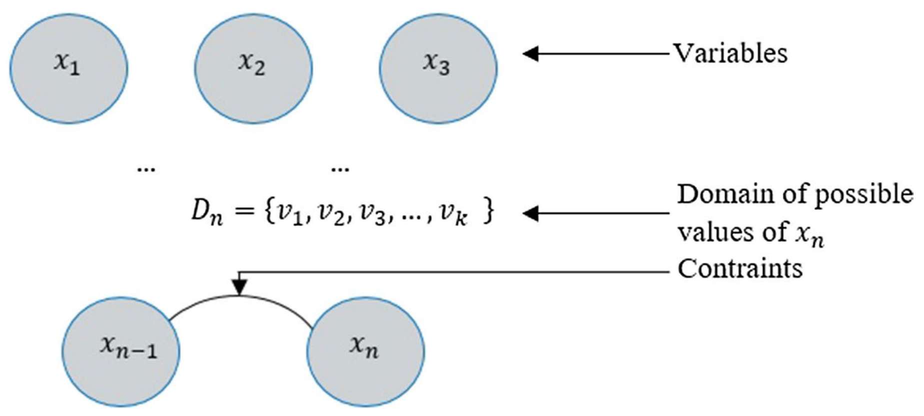
A constraint satisfaction problem (or CSP) is a triple $\langle X, D, C\rangle$, where X consists of a finite set of variables $X = \{x_1, x_2, ..., x_n\}$, D is an associated set of discrete-valued domains $D = \{D_1, D_2, ..., D_n\}$ that list the possible values of each variable $D_i = \{v_1, v_2, ..., v_k\}$ and C is a set of constraints $C = \{C_1, C_2, ..., C_m\}$. Each constraint C_i is a relation R_i defined on a subset of variables S_i, $S_i \subseteq X$, it specifies which combinations of values are compatible for the variables of S_i. Each relation denotes the simultaneous legal assignments of a set of variables. A constraint C_i is a pair (S_i, R_i), where R_i is a subset of the Cartesian product $D_{i_1} \times ... \times D_{i_k}$, $R_i \subseteq D_{i_1} \times ... \times D_{i_k}$ defined on a subset of variables $S_i = \{x_{i_1}, ..., x_{i_k}\}$ called the scope of C_i, $S_i = scope(C_i)$, consisting of all tuples of values of $\{x_{i_1}, ..., x_{i_k}\}$ that are compatible with each other.

A solution to the CSP consists of choosing, for each variable x_i ($x_i \in X$), a value a_i, chosen in its domain D_i, such that each constraint C_j, in which x_i participates, is satisfied. The set of solutions S is therefore represented by the Cartesian product of the domains $D_1 \times ... \times D_n$.

A constraint satisfaction problem can be represented by a constraint graph that contains a node for each variable and has an arc between two nodes if the corresponding variables participate in the same constraint (see Fig. 3).

Research methodology

To solve the problem defined above, the proposed approach is carried out in a cascading manner. The first phase consists of using CSP solvers to reduce the state space of the HMM. In the second phase, the resulting HMM is used as an initial model to estimate the optimal parameters and then to forecast the stock market.



x_1
x_2
x_3
Variables
...
...
$D_n = \{v_1, v_2, v_3, ..., v_k\}$
Domain of possible values of x_n
Contraints
x_{n-1}
x_n

Fig. 3 Graphical representation of a constraint satisfaction problem. A constraint satisfaction problem is defined by a set of variables (x_i), the corresponding set of domains (D_i) of the possible values of each variable and a set of constraints (C_j)

Link between the HMM and CSP

Given their power and their excellent performance in several applications, the use of HMMs in the big data context is common. However, to obtain good results, new methods must be found to optimize these algorithms. We are particularly interested in two characteristics of HMMs: the high number of states and the high number of observations. In this approach, we propose an optimization technique using metaheuristics based on the CSP. Treating a problem as a CSP confers several important benefits [56]. CSP resolution methods can be applied to reduce the state space of an HMM since the representation of a CSP conforms to a standard HMM model, i.e., a set of variables (states) with assigned values and a set of containers (state transitions and/or other external constraints). The use of CSPs to reduce the number of states or observations and therefore improve the performance of HMMs is justified by the power of their solver algorithms and because HMMs and CSPs can be seen and treated in the same way from a graphical point of view. Thus, the structure of the constraint graph can be used to simplify the solution process, in many cases conferring an exponential reduction in complexity and making it possible to improve the learning and prediction phases using HMMs.

The link between HMMs and CSPs can be explained as follows:

- Variables (states or nodes): $\{s_{t_1}, s_{t_2}, ..., s_{t_p}\}$, where $t_1, t_2, ..., t_p$ are fixed sequences of time. t_1 is the current time, and t_p is the time for which we want to predict the state of the system.
- Domains (state values): $\{D_1, D_2, ..., D_p\}$, where $D_k = \{s_1, s_2, ..., s_N\}$ for $k = 1, 2, ..., p$.
- Constraints (state transitions, arcs or edges):
  $\sum_{j=1}^{N} Pr\{s_t = s_i \mid s_{t+1} = s_j\} = 1$, $Pr\{s_t = s_i \mid s_{t+1} = s_j\} \geq 0$ for $i, j = 1, 2, ..., N$.

Resolution algorithm

The resolution algorithm is described by the steps below.

Data extraction

The data were extracted from the Yahoo Finance website.

Feature selection

Feature selection is a preprocessing method to select a feature subset from all the input features to make the constructed model better. Generally, in machine learning applications, the quantity of features is often very large; there may be irrelevant features, or the features may depend on each other. There are many benefits of feature selection, such as reducing the training time, storage needs and effects of the curse of dimensionality. However, effective data features can also improve the quality and performance, such as through redundancy reduction, noise elimination, improvement of the processing speed and even facilitation of data modeling. There are three major categories of feature selection: wrapper, filter, and embedded methods [57]. In our case, feature selection is implemented before the model training process to determine the most relevant stock market index variables for stock market prediction. This process determines features that are highly correlated with the index closing price but exhibit low correlation with each other [58]. Hence, we obtain a dataset of samples with 7 attributes.

Problem modeling

Topology selection: We consider an ergodic (fully connected state transition) topology for transitions between states. The number of states (N) and the number of mixture components per state (M) are chosen as fixed values initially. In our case, we construct an HMM with Gaussian mixtures as the observation density function. The HMM has 8 states ($N = 8$). Each state is associated with a Gaussian mixture that has 7 Gaussian probability distributions ($M = 7$).

Model parameter initialization: It is important to obtain good initial parameters so that the reestimation reaches the global maximum or is as close as possible to it. Therefore, we must first define the parameters of the initial model: the initial transition matrix, initial observation probability matrix and initial prior probability matrix. In general, several approaches are used to choose the initial parameters: (1) Choosing a random initial model: an initial model is chosen uniformly at random in the space of all possible models. For example, we can try several initial models to find different local likelihood maxima. (2) Choosing an informed initial model: an informed model allows us to obtain optimized initial values of the HMM parameters so that, after training, the HMM is the model best suited to the prediction phase. Here, we use a hybrid approach to initialize the model parameters. The initial state probabilities and state transition probabilities are chosen randomly so that they satisfy the criteria $\sum_{i=1}^{N} \pi_i = 1$ and $\sum_{j=1}^{N} a_{ij} = 1$. An adequate choice for Π and A is the uniform distribution when an ergodic model is used. Then, we used K-means to find the optimal initial parameters for the HMM with a Gaussian mixture. To initialize N mixtures each having M components, the training dataset X is fed into a K-means clustering algorithm, producing a clustering of the data into $K(= NM)$ groups. Thus, the training dataset is partitioned into K clusters $\{C_{11}, C_{12}, ..., C_{1M}, C_{21}, ..., C_{NM}\}$. Let x_i denote the mean of cluster C_{jm} and n_{jm} denote the number of instances in C_{jm}. Clustering is performed by choosing the cluster in such a way that the criterion value $SSE = \sum_{j=1}^{N} \sum_{m=1}^{M} \sum_{x_i \in C_{jm}} \|x_i - \mu_{jm}\|$ is minimized, where $\mu_{jm} = \frac{1}{n_{jm}} \sum_{x_i \in C_{jm}}$. We recompute the centroids μ_{jm} by taking the mean of the vectors that belong to each centroid. The mean value of each cluster is considered the initial cluster center. Once we obtain the initial cluster centers, K-means clustering is performed to obtain the final clusters. Finally, we obtain the initial parameters for the mixtures by calculating, for each cluster C_{jm}, the mean μ_{jm}, the covariance matrix Σ_{jm} and c_{jm} calculated by $c_{jm} = \frac{n_{jm}}{\sum_m n_{jm}}$.

HMM state space optimization using the CSP approach

In this step, we formulate the modeled problem using HMMs in the form of CSPs by exploiting the link between the two approaches, as previously described. We start with the hypothesis that we have a fully connected HMM; i.e., at the beginning, in the graph, all the transitions between the states are possible. The objective is to reject inaccessible nodes and reduce the transition matrix if possible, and ideally to obtain a diagonal matrix. Arc consistency verification can be applied as a preprocessing step before the beginning of the search process. To do this, we use a procedure based on the AC3 Arc consistency algorithm, namely, *CSP_Space_Reducer*(*csp*), which takes a CSP as input and returns a CSP with possibly reduced domains, hence an HMM with a reduced transition matrix. In a given CSP, the arc (x_i, x_j) is arc consistent if and only if for any value $v \in D_i$ that satisfies the unary constraint regarding x_i, there is a value $w \in D_j$ that satisfies the unary constraint regarding x_j and is such that the restriction between x_i and x_j is satisfied. Given discrete domains D_i and D_j, for two variables x_i and x_j that are node consistent, if $v \in D_i$ and there is no $w \in D_j$ that satisfies the restriction between x_i and x_j, then v can be deleted from D_i. When this has been done for each $v \in D_i$, arc (x_i, x_j) (but not necessarily (x_j, x_i)) is consistent [20]. To be able to eliminate, or not, a value from the domain of each variable, we perform a backtrack search. We choose a node and instantiate the corresponding variable as an occurrence belonging to its domain. Subsequently, we can discard the remaining values from its domain and run arc consistency algorithms to restore consistency. If the network succeeds, we fix an occurrence on another variable and run the local consistency algorithms again until each occurrence is fixed on the domain of each variable in the network. We obtain a solution corresponding to the set of occurrences fixed on the domain of each variable. If the network fails at some point, we backtrack and choose another occurrence on the domain of the last selected variable. By making the constraint graph arc-consistent, it is often possible to reduce the search space. For example, if $D1 = 1, 2, 3, 4$ and $D2 = 1, 2, 3, 4$ and the constraint is $x1 > x2$, what can be eliminated? We can eliminate the values of a domain that are not in any solution. In this example, we can eliminate 1 from $D1$ and 4 from $D2$. Thus, at a given instant t, the possible states of the system are only those that satisfy all the constraints, and therefore, the number of transitions from and to the state concerned is reduced. In addition to the constraint of the model, we inject the CSP solver with constraints from a sentiment dataset created by considering a financial news dataset and President Trump's tweets. Both the tweets and news are collected for the studied period, and a sentiment analysis algorithm is applied. The sentiments of the tweets and news, according to a binary classification, are integrated daily. AC3 runs in $O(d^3n^2)$ time, with n being the number of nodes and d being the maximum number of elements in a domain.

In summary, to predict the direction of the index closing price movement on day p, we proceed as follows: for the first day t_1, we choose the initial parameters of the models as previously described (i.e., the hybrid approach with a random and informed model), and for the next day, we take as the initial HMM parameters the result of the HMM parameter estimation of the day before, and so on. Initially, we consider a fully connected HMM, which means that in the constraint graph (i.e., the HMM modeled as a CSP), every node participates with other nodes in at least one constraint $(\forall\, node_k, node_l \in G_{csp}(X, D, C)\, \exists\, C_i \in C$, where C_i is a relation R_i defined in the subset $\{node_k, node_l\} \subseteq X)$. On the first day, all the variables' domains have the same elements (i.e., all states), and we have a dense transition matrix. That is, we start with a dense transition matrix, and, by following the CSP-based optimization procedure operating according to the snowball principle (iterating the optimization procedure), we end with a sparse transition matrix, ideally a diagonal transition matrix.

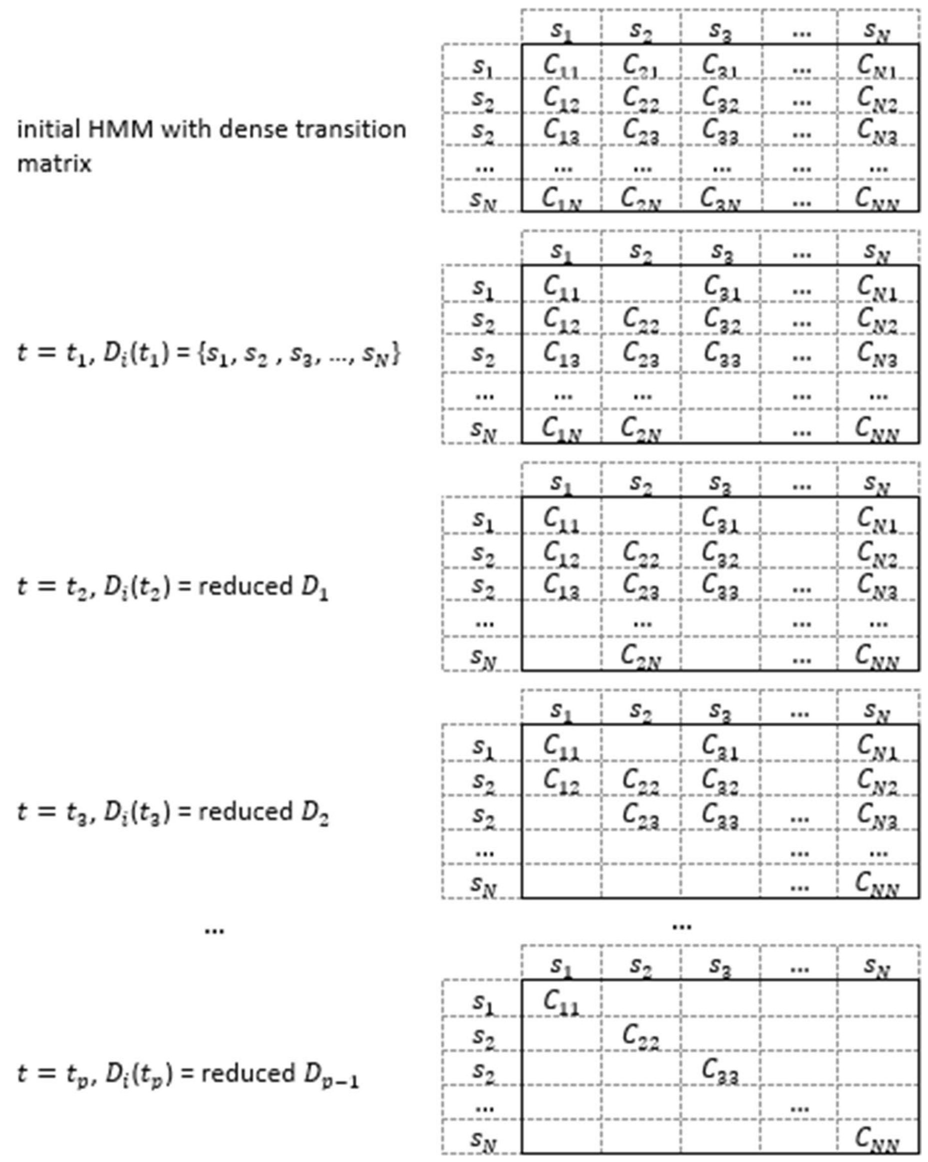

initial HMM with dense transition matrix
$t = t_1, D_i(t_1) = \{s_1, s_2, s_3, ..., s_N\}$
$t = t_2, D_i(t_2)$ = reduced D_1
$t = t_3, D_i(t_3)$ = reduced D_2
...
$t = t_p, D_i(t_p)$ = reduced D_{p-1}

Fig. 4 Transition matrix reduction. The figure shows the principle of optimization

The principle of optimization can be schematized as in Fig. 4.

Learning step using the Baum-Welch Algorithm

Now that the state space is reduced, we apply the Baum-Welch algorithm for the estimation of the model parameters. Baum-Welch runs in $O(N^2(T-1))$ time, with N being the number of states in the model and T being the length of the observation sequence.

Prediction step using the Viterbi Algorithm

After the training step, we use the Viterbi algorithm to predict the direction of the price movement. The Viterbi algorithm has a time complexity of $O(TN^2)$.

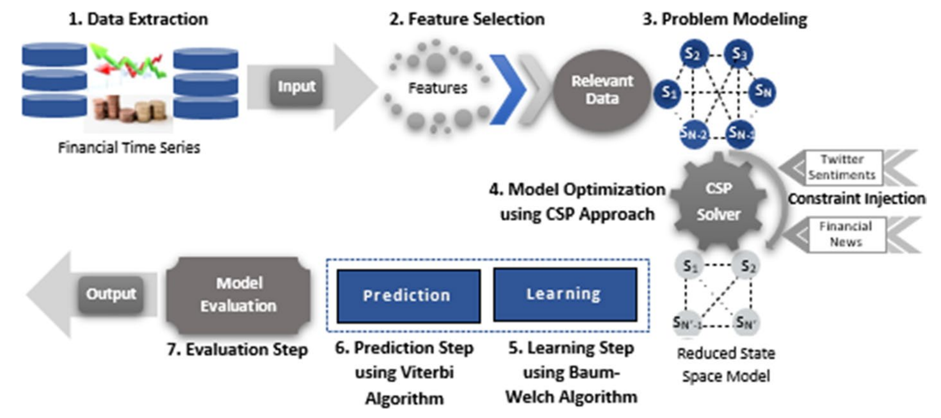


Fig. 5 Schematic representation of the proposed approach. The figure presents the steps of the proposed CSP-based approach for HMM optimization

Model evaluation

To evaluate the model, we perform experiments using real data. We evaluate the proposed algorithm compared to the conventional algorithm in terms of the computational complexity, running time, accuracy, recall, precision, and f-measure. To verify whether the result is significant, we use the McNemar test. The proposed approach is also compared with the main benchmark models for more meaningful evaluation.

An overview of the proposed approach is presented in Fig. 5.

The optimized algorithm cascaded with the algorithm of the learning and prediction phases of an HMM is described in Algorithm 3.

Here, in Algorithm 3, $G_{csp}(X, D, C)$ represents the constraint graph with the parameters X (set of variables), D (set of variable domains) and C (set of constraints). Backtrack search is a common backtracking algorithm used for solving CSPs. C_{kl} denotes a set of constraints defined on the subset of variables $\{node_k, node_l\}$.

```
Algorithm 3: CSP-Baum-Welch-Viterbi algorithm
Input:
    $\lambda$: initial model $\lambda = (A, B, \Pi)$
    $O$: a sequence of observations $O = o_1, o_2, ..., o_T$
    $N$: State number;
Output:
    $Q$: the hidden state sequence that produced the observation sequence $Q = \{q_1, q_2, ..., q_T\}$
1: Optimization of state space using CSP;
    Input: HMM ($N$ states)
    procedure $CSP\_Space\_Reducer(csp)$
    for each $node_k, node_l \in G_{csp}(X, D, C)$
        for each $v_r \in D_k$ Do Backtrack Search
            If there exists no $v_s \in D_l$ such that $C_{kl}\{node_k, node_l\}$ is satisfied (Backtrack Search
    returns failure) then remove $v_r$ from $D_k$
    end procedure
    Output: HMM' ($N'$ states)
2: Learning step using Baum-Welch algorithm;
3: Prediction step using Viterbi algorithm;
4: return $Q$
```

Experiments and results

Case study

To evaluate the proposed approach, we put it into practical use in a real case of financial time series analysis: stock market forecasting. The objective is to predict the direction of the price movement.

In this study, we considered the following states and observations:

- States: very small rise (0≤Variation<0.1%), small rise (0.1% ≤Variation<1%), large rise (1% ≤Variation<2%), very large rise (Variation≥2%), very small drop (– 0.1% < Variation≤0), small drop (– 1% <Variation≤– 0.1%), large drop (– 2% <Variation≤– 1%), and very large drop (Variation≤-2%), where Variation = Close price(t+1)-Close price(t).

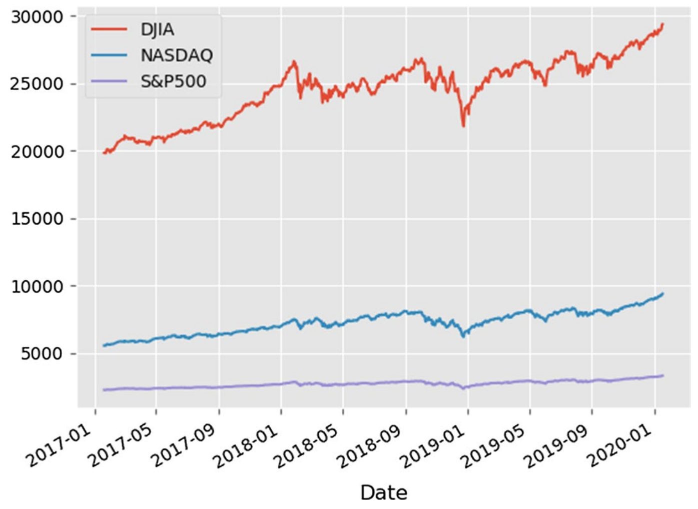


Fig. 6 Representation of the closing price of the three stock market indices, the Dow Jones Industrial Average (DJIA), NASDAQ and S&P 500, during the period between January 20, 2017 and January 20, 2020

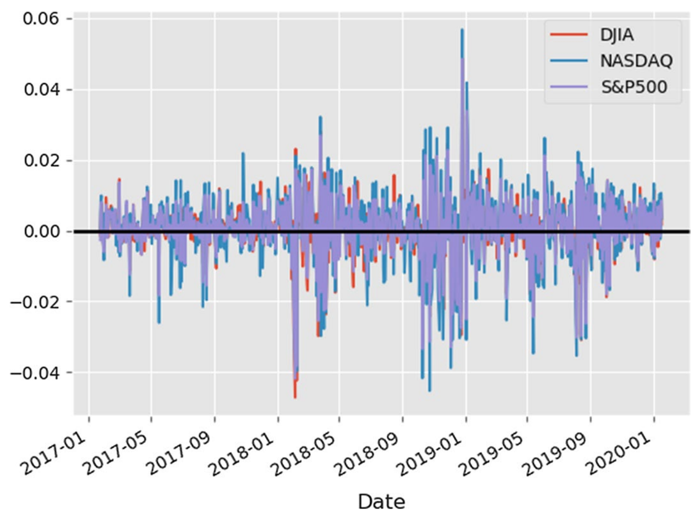


Fig. 7 Daily change in each stock of the three stock market indices, the Dow Jones Industrial Average (DJIA), NASDAQ and S&P 500, during the period between January 20, 2017 and January 20, 2020

Table 2 Part of DJIA index daily prices data

Date	Open	High	Low	Close	Adjusted	Volume
Jan 20, 2017	19795.0605	19843.9394	19759.1406	19827.2500	19827.2500	435260000
Jan 23, 2017	19794.7890	19833.9804	19732.3593	19799.8496	19799.8496	326690000
Jan 16, 2020	29131.9492	29300.3203	29131.9492	29297.6406	29297.6406	252110000
Jan 17, 2020	29313.3105	29373.6191	29289.9101	29348.0996	29348.0996	321820000

- Observations: djia-close-price, djia-high, djia-low, djia-variation, djia-volume, nasdaq-high, nasdaq-low, nasdaq-variation, nasdaq-volume, s&p500-high, s&p500-low, and s&p500-variation, s&p500-volume.

We construct an HMM with Gaussian mixtures as the observation density function. The HMM has 8 states. Each state is associated with a Gaussian mixture that has 10 Gaussian probability distributions. The HMM is a fully connected trellis, and each of the transition probabilities is assigned a real value at random between 0 and 0.125. The initial state probability distribution probabilities π_i are assigned a set of random values between 0 and 0.125. We used k-means algorithm to find the optimal initial parameters of the initial mixture weights c_{jm}, the initial covariance Σ_{jm} and the initial mean μ_{jm} of each Gaussian mixture component, as described previously.

Test data

In this work, we use historical daily data of three stock market indices: the Dow Jones Industrial Average (DJIA), NASDAQ, and S&P 500 during the period between January 20, 2017 (beginning of term of the American president Donald Trump) and January 20, 2020 obtained from the Yahoo Finance website [59] (Fig. 6). After data preprocessing, the dataset contains 2262 (754∗3) samples and has 7 attributes (date,

Table 3 Computational complexity comparison

Algorithms	Computational Complexity	
	Original algorithm	CSP-Optimized algorithm
Baum-Welch	$O(N^2(T-1))$	$O(N'^2(T-1))$
Viterbi	$O(N^2(T-1))$	$O(N'^2(T-1))$

open price, high price, low price, closing price, adjusted closing price and volume). The aim of the model is to combine observations from the three indices to predict the movement of the closing price of the Dow Jones Industrial Average Index.

The entire dataset is divided into two categories. The training dataset is 80% of the data (from 20 January 2017 to 13 June 2019), and the testing dataset is 20% of the data (from 14 June 2019 to 17 January 2020).

Fig. 7 shows the correlation between the three stock market indices used in this work.

Some of the data used in this paper are shown in Table 2.

Experimental setup

The experiments were performed on Ubuntu Linux 18.04.5 LTS with Linux Kernel 5.4. All tests were conducted using the same hardware: an Acer Aspire 5551G-P324G32Mnkk laptop with an AMD Athlon II Dual Core Processor P320, 2.3 GHz, 4Go DDR4 on an Integrated ATi Radeon HD5470 512Mo graphics card based on the Park XT graphics processor. All the programs were coded in the Python programming language. First, we performed experiments without optimization techniques. Then, we treated the HMM model as a CSP, specifically, as a constraint optimization problem, to reduce the state space by adding external constraints to the internal constraints of the model. Thus, we carried out experiments by adding constraints related to the economic and political situation of the United States of America at the time of prediction.

Table 4 Running time comparison of Viterbi algorithm

Algorithm	Running time (s)	
	Original algorithm	CSP-Optimized algorithm
Viterbi	0,1	0,08

Table 5 Running time comparison of Baum-Welch algorithm

Iterations number	Running time (s)	
	Original Baum-Welch	CSP-Optimized Baum-Welch
100000	2580	2520
10000	2054	1987
1000	1976	1933
600	1820	1768

Computational complexity

The computational complexity of both the original Baum-Welch and Viterbi algorithms is $O(N^2(T-1))$, where N is the number of states and T is the observation sequence length. For the improved algorithms using the CSP optimization approach, the time complexity is reduced to $O({N'}^2(T-1))$, where $N' < N$ (see Table 3).

Performance evaluation

Running time

We compared the running time of both the conventional and optimized Baum-Welch and Viterbi algorithms. Tables 4 and 5 show an improvement in terms of the running time of the optimized algorithms compared to that of the original algorithms. We also note that the improvement in the running time remains almost the same as the variation in the number of iterations.

Quality of prediction

We compared the quality of prediction of the two decoding algorithms in the standard and optimized versions using the occurrences of the direction of the index closing price movement that were correctly predicted. To investigate the prediction quality, we compared the decoding and the true hidden state (i.e., the state of the direction of the daily closing price movement) in the test set. In this paper, the performance metrics, namely, recall, precision, F-measure, accuracy, and mean absolute percentage error (MAPE), were considered to evaluate the performance of the proposed approach. Because these measures require binary data, we converted the obtained decoding results to binary data by classifying all closing price variation increases (i.e., very small rise, small rise, large rise, and very large rise) as a price increase and all closing price variation drops (i.e., very small drop, small drop, large drop, and very large drop) as a price decrease.

Depending on the obtained results, each predicted direction of the daily closing price movement can be classified as a true positive (*TP*) if it is predicted by the HMMs as a rise and it is truly a rise, a true negative (*TN*) if it is predicted by the HMMs as a drop and it is actually a drop, a false positive (*FP*) if it is predicted as a rise when it is not actually a rise and finally as a false negative (*FN*) if it is predicted as a drop and it is actually a rise. Using the total number of true positives, true negatives, false positives, and false negatives, we calculated the recall, precision, F-measure, accuracy and MAPE, defined as follows:

Recall, also known as sensitivity, is the fraction of relevant instances that are retrieved. Recall represents how sensitive the predictor is and serves as a measurement of predictor completeness. It is computed as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Precision is the fraction of retrieved instances that are relevant. Precision indicates the exactness of the predictor. It is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Table 6 Accuracy comparison

Iterations number	Accuracy of algorithms (%)	
	Standard HMM	**CSP-Optimized HMM**
100000	96.15	96.86
10000	94.36	94.36
1000	91.05	89.45
600	87.08	85.12

The F-measure, also known as the F-score or F1 score, can be interpreted as a weighted average of the precision and recall. It measures the weighted harmonic mean of the precision and recall, where the F-measure reaches the best value at 1 and the worst at 0. It is defined as:

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{5}$$

Accuracy reflects the percentage of correctly predicted closing price directions with respect to the total number of predictions. It is estimated using:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

The mean absolute percentage error (MAPE) is the percentage of accurate predicted values with respect to the computed actual values. The MAPE is defined as the sum of the differences between the actual values and the predicted data divided by the total number of test data. It is given by the following equation:

Table 7 Recall values comparison

Iterations number	Recall Value of algorithms (%)	
	Standard HMM	**CSP-Optimized HMM**
100000	91.85	90.93
10000	87.30	86.60
1000	87.40	87.95
600	84.50	86.20

Table 8 Precision values comparison

Iterations number	Precision Value of algorithms (%)	
	Standard HMM	**CSP-Optimized HMM**
100000	87.40	88.30
10000	85.90	86.07
1000	86.40	86.16
600	78.60	76.83

Table 9 F-measure values comparison

Iterations number	F-measure Value of algorithms (%)	
	Standard HMM	CSP-Optimized HMM
100000	89.56	89.59
10000	86.59	86.33
1000	86.89	87.04
600	81.44	81.24

Table 10 Mean absolute percentage error (MAPE) of prediction accuracy comparison

Iterations number	MAPE of prediction accuracy of algorithms (%)	
	Standard HMM	CSP-Optimized HMM
100000	03.85	03.14

$$MAPE = \frac{\sum_i^n |\frac{A_i - F_i}{A_i}|}{n} * 100 \tag{7}$$

where F_i is the forecast direction of the closing price movement on day i, A_i is the actual direction of the closing price movement on day i and n is the total number of test data.

Table 6 shows the accuracy of the standard and optimized HMMs. The first row shows the number of iterations, and the second and third rows show the accuracy statistics for the two compared algorithms. In terms of accuracy, we note that the optimized HMM using the CSP surpasses the standard HMM for a high number of iterations, while it presents almost the same performance for a small number of iterations.

In Table 7, the recall of the results is shown when the HMMs are trained with different numbers of iterations. A global view of the result shows that the recall value is not affected significantly by the optimization approach. Additionally, as the number of iterations increases, the recall increases. This recall improvement occurs because the increase in the number of iterations improves the learning of the optimal HMM parameters and consequently the quality of prediction.

Table 8 illustrates the results of the algorithm precision comparison. This table shows the impact of the CSP optimization on the precision of the algorithm. We note a clear improvement for high values of the number of iterations.

In Table 9, the results of the F-measure statistics for the standard and optimized HMM are illustrated. As this expresses the weighted average of the precision and recall, it also presents good results for a high number of iterations, especially for the optimized algorithms.

In Table 10, the result of the comparison of the mean absolute percentage error of the prediction accuracy is illustrated with only the best number of iterations for HMM learning (i.e., 100000 iterations). The overall MAPE of the optimized HMM for a number of iterations of 100000 is 03.14%, and it improved only slightly compared to the standard HMM for the same number of iterations.

The optimized model achieves an overall accuracy of 96.86%, a recall of 90.93% and a precision of 88.30% with an F-measure of 89.59% for a number of iterations of 100000.

The proposed model was compared to the main models proposed in the literature in terms of complexity, accuracy and speedup. The results show that the new model outperforms those of Li et al. [42] and Wen et al. [43] in terms of computational complexity; this can be explained by the efficiency of the optimization approach, which allows us to considerably reduce the state space and obtain better performance. Moreover, because there are fundamental differences in the setup of the experiments between the studies, we calculated the best percentage improvement in terms of accuracy instead of directly measuring the accuracy. Our approach has the worst result compared to those of Wen et al. [43] and Zheng et al. [44] since both models improved the accuracy of the classical HMM by 5.44% and 11.59%, respectively. This can be explained by the fact that the proposed model needs to be iterated several times to yield good results. In addition, the proposed approach is specifically designed for HMMs with large state spaces. Concerning speedup, we computed the relative speedup between the conventional algorithm and the proposed algorithm in each study and compared it with the other versions. The proposed optimization approach allows a significantly higher speedup of 3.73% compared to that of Zheng et al. [44], which drops dramatically by 4.40%.

Even though the proposed approach is effective compared to other optimization solutions, there are some limitations regarding the applicability, the characteristics of the model and the CSP concept.

The main drawback of the proposed big data optimization approach is that it does not improve the accuracy of the classic model when it is iterated a small number of times. The proposed model must be iterated several times for better results. This is primarily because forecasting the stock market is very challenging, especially under the fluctuation and instability of financial markets due to economic, political, and social factors. In addition, this approach does not seem to be efficient for low numbers of iterations in a relatively small state space because the rejection of even one node can significantly affect the accuracy of the model. Another main limitation of the model is its memoryless property since it allows the rejection of certain states without keeping a history of the initial structure of the model. Furthermore, the CSP paradigm on which this approach is based implies that the constraints used must be selected carefully; otherwise, the accuracy may deteriorate.

On the other hand, the proposed model allows an acceptable level of accuracy and precision. It is even faster when the number of iterations is high for a large state space. When the dimension of the state space increases, the acceleration of the model increases while keeping the accuracy close to that of the classical model. In addition, unlike other HMM optimization approaches that optimize either the initial parameters or the topology used, the approach presented here is complete. It aims to address the core of HMM problems in the big data context. It seeks to reduce the state transition matrix and the observation probability matrix.

This study's first concern is to improve the model so that it works well with big data. Generally, in the big data context, we are interested in a high number of iterations with a large state space. This leads us to conclude that the result obtained for a small number of iterations does not affect the efficiency of the proposed approach because even if we

Table 11 The contingency table of the McNemar test

	CSP-Optimezed HMM Correct Predictions	**CSP-Optimized HMM Incorrect Predictions**
Standard HMM Correct Predictions	a=714 (number of (Yes/Yes))	b=12 (number of (Yes/No))
Standard HMM Incorrect Predictions	c=25 (number of (No/Yes))	d=03 (number of (No/No))

increase the number of iterations, we always keep an acceptable running time because the state space is considerably reduced compared to that of conventional HMMs. Nevertheless, the drawbacks discussed previously need to be addressed in future research to improve our model. This motivates us to work on other alternative convergence criteria to overcome the problem of the results for a low number of iterations. We focus on a new graph-based HMM similarity measure to replace the number-of-iterations criterion with a more efficient convergence criterion.

Result validation

To validate the proposed model, we used the McNemar test [60] to compare the predictive accuracy of the two models to certify the fact that the results of the prediction accuracy are significant and are due mainly to the proposed approach and not to chance. In terms of comparing two algorithms, the test concerns whether the two models disagree in the same way. It does not concern whether one model is more or less accurate or error prone than another.

To run the McNemar test, we calculate the correct predictions of the two models daily. The results are placed into a 2×2 contingency table (see Table 11), with the cell frequencies equaling the number of pairs. The rows represent the correct predictions and incorrect predictions, respectively, of the standard HMM model, and the columns represent the correct predictions and incorrect predictions, respectively, of the CSP-optimized HMM model. For example, the intersection of the first row and the first column is the total number of instances where both models have correct predictions (Yes/Yes).

Using cells b and c, called “discordant” cells, the McNemar test calculates:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{8}$$

In the McNemar test, we formulate the null hypothesis of marginal homogeneity, that the two marginal probabilities for each outcome $p(b)$ and $p(c)$ are the same, or in other words, neither of the two models performs better than the other. Thus, the null and alternative hypotheses are:

$$H_0 : p(b) = p(c) \tag{9}$$

$$H_1 : p(b) \neq p(c) \tag{10}$$

With one degree of freedom and an alpha risk (Type I error) $\alpha = 0.05$, the critical value of χ^2 from the chi-square table is $\chi^2{}_{\alpha=0.05,dl=1} = 3.841$.

Given that

$$\chi^2 = \frac{(12-25)^2}{12+25} = 4.567 \tag{11}$$

and since $\chi^2 \geq {\chi^2}_{\alpha=0.05,dl=1}$, we can reject the null hypothesis H_0. Finally, we can say that the result of the proposed approach is significant.

Conclusion and future directions

In this paper, a big data optimization approach for the refinement of hidden Markov models is proposed. To enhance the results of the learning and prediction tasks of HMMs, we present a graph-based approach using a constraint satisfaction problem to reduce the state and/or observation space. The accuracy of the Baum-Welch and Viterbi algorithms is optimized through a dimension-reduction optimization approach. This hybrid approach allows us to improve the performance of hidden Markov models. The complexity of the learning and prediction phases is remarkably improved. The experimental results indicate the superiority of the proposed approach, which exhibits notable strengths over the classical algorithms and shows competitiveness compared to other benchmark algorithms in terms of the mean absolute percentage error, running time and computational complexity. The results demonstrate that the proposed algorithm reaches a comparable accuracy, although the conventional algorithm performs slightly better for a low number of iterations.

To deal with low numbers of iterations, we are interested in a new graph-based HMM similarity measure to replace the number-of-iterations criterion with another more efficient convergence criterion.

Future work will involve deeper analysis of the most important HMM problems, mainly the parameters of learning and prediction tasks. This approach can be adapted and applied to other real cases, and it will be interesting to use other metrics for a good evaluation.

To improve this approach, we will focus on parallel and/or distributed hidden Markov models using metaheuristics since they are parallelizable and can therefore be used with different big data technologies in a cloud environment.

Abbreviations
CSP: Constraint Satisfaction Problem; HMM: Hidden Markov Models; EM: Expectation Maximization; AC3: Arc Consistency Algorithm 3; IDC: International Data Corporation; GPGPUs: General-Purpose Graphics Processing Unit; DJIA: Dow Jones Industrial Average; NASDAQ: National Association of Securities Dealers Automated Quotations; S&P: Standard & Poor's; MAPE: Mean Absolute Percentage Error.

Acknowledgements
Not applicable.

Authors' contributions
IS was the main contributor of this work. He has done an initial literature review, data generation, experiments, prepare results, and drafted the manuscript. The other authors worked closely with IS to review, analyze and prepare the manuscript. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 23 January 2021 Accepted: 17 June 2021
Published online: 29 June 2021

References

1. Luengo J, García-Gil D, Ramírez-Gallego S, García S, Herrera F. Big data preprocessing. 1st ed. Switzerland AG: Springer; 2020. p. 186. https://doi.org/10.1007/978-3-030-39105-8.
2. Tsai CW, Lai CF, Chao HC, Vasilakos AV. Big data analytics: a survey. J Big Data. 2015;2(1):1–31. https://doi.org/10.1186/s40537-015-0030-3.
3. El-Alfy ESM, Mohammed SA. A review of machine learning for big data analytics: bibliometric approach. Technol Anal Strateg Manag. 2020;32(8):984–1005. https://doi.org/10.1080/09537325.2020.1732912.
4. Hariri RH, Fredericks EM, Bowers KM. Uncertainty in big data analytics: survey, opportunities, and challenges. J Big Data. 2019;6(1):1–16. https://doi.org/10.1186/s40537-019-0206-3.
5. Lee I. Big data: Dimensions, evolution, impacts, and challenges. Business Horizons. 2017;60(3):293–303. https://doi.org/10.1016/j.bushor.2017.01.004.
6. Sassi I, Anter S, Bekkhoucha A. An overview of big data and machine learning paradigms. Int Conf Adv Intell Syst Sustain Dev. 2018;915:237–51. https://doi.org/10.1007/978-3-030-11928-7_21.
7. Seethalakshmi V, Govindasamy V, Akila V. Hybrid gradient descent spider monkey optimization (HGDSMO) algorithm for efficient resource scheduling for big data processing in heterogenous environment. J Big Data. 2020;7(1):1–25. https://doi.org/10.1186/s40537-020-00321-w.
8. Al Jallad K, Aljnidi M, Desouki MS. Anomaly detection optimization using big data and deep learning to reduce false-positive. J Big Data. 2020;7(1):1–12. https://doi.org/10.1186/s40537-020-00346-1.
9. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH. The arithmetic optimization algorithm. Comput Methods Appl Mech Eng. 2021;376:113609. https://doi.org/10.1016/j.cma.2020.113609.
10. Emrouznejad A. Big data optimization: recent developments and challenges. vol. 18. 1st ed. Switzerland AG: Springer. 2018. p. 487. https://doi.org/10.1007/978-3-319-30265-2.
11. Dhaenens C, Jourdan L. Metaheuristics for big data. 1st ed. London: Wiley Online Library. 2016. p. 212. https://doi.org/10.1002/9781119347569.
12. Chopard B, Tomassini M. An introduction to metaheuristics for optimization. 1st ed. Switzerland AG: Springer. 2018. p. 266. https://doi.org/10.1007/978-3-319-93073-2.
13. Mor B, Garhwal S, Kumar A. A systematic review of hidden markov models and their applications. Arch Comput Methods Eng. 2020;28:1–20. https://doi.org/10.1007/s11831-020-09422-4.
14. Mao S, Tao D, Zhang G, Ching P, Lee T. Revisiting hidden markov models for speech emotion recognition. ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2019; 1:6715–9. https://doi.org/10.1109/ICASSP.2019.8683172.
15. Nasfi R, Amayri M, Bouguila N. A novel approach for modeling positive vectors with inverted dirichlet-based hidden markov models. Knowl Based Syst. 2020;192:105335. https://doi.org/10.1016/j.knosys.2019.105335.
16. Kwon BC, Anand V, Severson KA, Ghosh S, Sun Z, Frohnert BI, Lundgren M, Ng K. DPVis: Visual analytics with hidden markov models for disease progression pathways. IEEE Trans Vis Comput Graph. 2020. https://doi.org/10.1109/TVCG.2020.2985689.
17. Nystrup P, Lindström E, Madsen H. Learning hidden markov models with persistent states by penalizing jumps. Expert Syst Appl. 2020;150:113307. https://doi.org/10.1016/j.eswa.2020.113307.
18. Gao J, Wang J, Wu K, Chen R. Solving quantified constraint satisfaction problems with value selection rules. Front Comput Sci. 2020;14(5):1–11. https://doi.org/10.1007/s11704-019-9179-9.
19. Lember J, Sova J. Regenerativity of viterbi process for pairwise markov models. J Theor Probab. 2020;34:1–33. https://doi.org/10.1007/s10959-020-01022-z.
20. Wang R, Yap RH. Arc consistency revisited. Int Conf Integr Constraint Program Artif Intell Oper Res. 2019;11494:599–615. https://doi.org/10.1007/978-3-030-19212-9_40.
21. Hsieh T Y, Wang S, Sun Y, Honavar V. Explainable multivariate time series classification: a deep neural network which learns to attend to important variables as well as informative time intervals. arXiv preprint arXiv:2011.11631. 2020; 1:607–15. https://doi.org/10.1145/3437963.3441815.
22. Shen J, Shafiq MO. Short-term stock market price trend prediction using a comprehensive deep learning system. J Big Data. 2020;7(1):1–33. https://doi.org/10.1186/s40537-020-00333-6.
23. Sohangir S, Wang D, Pomeranets A, Khoshgoftaar TM. Big data: Deep learning for financial sentiment analysis. J Big Data. 2018;5(1):1–25. https://doi.org/10.1186/s40537-017-0111-6.
24. Budiharto W. Data science approach to stock prices forecasting in indonesia during covid-19 using long short-term memory (LSTM). J Big Data. 2021;8(1):1–9. https://doi.org/10.1186/s40537-021-00430-0.

25. Nti IK, Adekoya AF, Weyori BA. A novel multi-source information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction. J Big Data. 2021;8(1):1–28. https://doi.org/10.1186/s40537-020-00400-y.
26. Dash RK, Nguyen TN, Cengiz K, Sharma A. Fine-tuned support vector regression model for stock predictions. Neural Comput Appl. 2021;1:1–15. https://doi.org/10.1007/s00521-021-05842-w.
27. Sedighi M, Jahangirnia H, Gharakhani M, Farahani Fard S. A novel hybrid model for stock price forecasting based on metaheuristics and support vector machine. Data. 2019;4(2):75. https://doi.org/10.3390/data4020075.
28. Hao PY, Kung CF, Chang CY, Ou JB. Predicting stock price trends based on financial news articles and using a novel twin support vector machine with fuzzy hyperplane. Appl Soft Comput. 2021;98:106806. https://doi.org/10.1016/j.asoc.2020.106806.
29. Ren R, Wu DD, Liu T. Forecasting stock market movement direction using sentiment analysis and support vector machine. IEEE Syst J. 2018;13(1):760–70. https://doi.org/10.1109/JSYST.2018.2794462.
30. Vijh M, Chandola D, Tikkiwal VA, Kumar A. Stock closing price prediction using machine learning techniques. Procedia Comput Sci. 2020;167:599–606. https://doi.org/10.1016/j.procs.2020.03.326.
31. Chandar SK. Grey wolf optimization-elman neural network model for stock price prediction. Soft Comput. 2021;25(1):649–58. https://doi.org/10.1007/s00500-020-05174-2.
32. Nayak SC, Misra BB. A chemical-reaction-optimization-based neuro-fuzzy hybrid network for stock closing price prediction. Financial Innov. 2019;5(1):1–34. https://doi.org/10.1186/s40854-019-0153-1.
33. Gao P, Zhang R, Yang X. The application of stock index price prediction with neural network. Math Comput Appl. 2020;25(3):53. https://doi.org/10.3390/mca25030053.
34. Zhong X, Enke D. Forecasting daily stock market return using dimensionality reduction. Expert Syst Appl. 2017;67:126–39. https://doi.org/10.1016/j.eswa.2016.09.027.
35. Lv D, Wang D, Li M, Xiang Y. DNN models based on dimensionality reduction for stock trading. Intell Data Anal. 2020;24(1):19–45. https://doi.org/10.3233/IDA-184403.
36. Ghorbani M, Chong EK. Stock price prediction using principal components. PLoS ONE. 2020. https://doi.org/10.1371/journal.pone.0230124.
37. Nti IK, Adekoya AF, Weyori BA. A comprehensive evaluation of ensemble learning for stock-market prediction. J Big Data. 2020;7(1):1–40. https://doi.org/10.1186/s40537-020-00299-5.
38. Wu JMT, Li Z, Herencsar N, Vo B, Lin JCW. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimed Syst. 2021;1:1–20. https://doi.org/10.1007/s00530-021-00758-w.
39. Fons E, Dawson P, Yau J, Zeng XJ, Keane J. A novel dynamic asset allocation system using feature saliency hidden markov models for smart beta investing. Expert Syst Appl. 2021;163:113720. https://doi.org/10.1016/j.eswa.2020.113720.
40. Nguyen N, Nguyen D. Global stock selection with hidden markov model. Risks. 2021;9(1):9. https://doi.org/10.3390/risks9010009.
41. Chen P, Yi D, Zhao C. Trading strategy for market situation estimation based on hidden markov model. Mathematics. 2020;8(7):1126. https://doi.org/10.3390/math8071126.
42. Li J, Lee JY, Liao L. A new algorithm to train hidden markov models for biological sequences with partial labels. BMC Bioinformatics. 2021;22(1):1–21. https://doi.org/10.1186/s12859-021-04080-0.
43. Wen R, Wang Q, Ma X, Li Z. Human hand movement recognition based on HMM with hyperparameters optimized by maximum mutual information. 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). 2020; 1:944–951. https://ieeexplore.ieee.org/document/9306365.
44. Zheng H, Wang R, Xu W, Wang Y, Zhu W. Combining a HMM with a genetic algorithm for the fault diagnosis of photovoltaic inverters. J Power Electron. 2017;17(4):1014–26. https://doi.org/10.6113/JPE.2017.17.4.1014.
45. Bražėnas M, Horváth G, Telek M. Parallel algorithms for fitting markov arrival processes. Perform Eval. 2018;123:50–67. https://doi.org/10.1016/j.peva.2018.05.001.
46. Sassi I, Anter S, Bekkhoucha A. A new improved baum-welch algorithm for unsupervised learning for continuous-time hmm using spark. Int J Intell Eng Syst. 2020;13(1):214–26. https://doi.org/10.22266/ijies2020.0229.20.
47. Reinsel D, Gantz J, Rydning J, Data age 2025. the digitization of the world: From edge to core. an IDC white paper# US44413318. Tech. rep. IDC. 2018. https://resources.moredirect.com/white-papers/idc-report-the-digitization-of-the-world-from-edge-to-core.
48. Sassi I, Ouaftouh S, Anter S. Adaptation of classical machine learning algorithms to big data context: problems and challenges: Case study: Hidden markov models under spark. 2019 1st International Conference on Smart Systems and Data Science (ICSSD). 2019; 1:1–7. https://doi.org/10.1109/ICSSD47982.2019.9002857.
49. Zhou L, Pan S, Wang J, Vasilakos AV. Machine learning on big data: opportunities and challenges. Neurocomputing. 2017;237:350–61. https://doi.org/10.1016/j.neucom.2017.01.026.
50. Sassi I, Anter S. A study on big data frameworks and machine learning tool kits. Int Conf Big Data Anal Data Mining Comput Intel. 2019;1:61–8. https://doi.org/10.33965/bigdaci2019_201907l008.
51. Coimbra ME, Francisco AP, Veiga L. An analysis of the graph processing landscape. J Big Data. 2021;8(1):1–41. https://doi.org/10.1186/s40537-021-00443-9.
52. Jain P, Agarwal A, Behara R, Baechle C. HPCC based framework for COPD readmission risk analysis. J Big Data. 2019;6(1):26. https://doi.org/10.1186/s40537-019-0189-0.
53. Belcastro L, Marozzo F, Talia D, Trunfio P. ParSoDA: high-level parallel programming for social data mining. Soc Netw Anal Min. 2019;9(1):4. https://doi.org/10.1007/s13278-018-0547-5.
54. Xu L, Apon A, Villanustre F, Dev R, Chala A. Massively scalable parallel KMeans on the HPCC systems platform. 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS). 2019; 4:1–8. https://par.nsf.gov/servlets/purl/10201358.
55. Hamdia KM, Zhuang X, Rabczuk T. An efficient optimization approach for designing machine learning models based on genetic algorithm. Neural Comput Appl. 2021;33(6):1923–33. https://doi.org/10.1007/s00521-020-05035-x.
56. Norvig P, Russell S. Artificial intelligence: a modern approach, global edition. 4th ed. London: Pearson Education Limited; 2021. p. 1170.

57. El-Hasnony IM, Barakat SI, Elhoseny M, Mostafa RR. Improved feature selection model for big data analytics. IEEE Access. 2020;8:66989–7004. https://doi.org/10.1109/ACCESS.2020.2986232.
58. Chmielewski L, Amin R, Wannaphaschaiyong A, Zhu X. Network analysis of technology stocks using market correlation. 2020 IEEE International Conference on Knowledge Graph (ICKG). 2020; 1:267–274. https://doi.org/10.1109/ICBK50248.2020.00046.
59. Yahoo! Dow jones industrial average (^dji). 2020. finance.yahoo.com. Accessed 1 Feb 2020. https://finance.yahoo.com/quote/%5EDJI?p=^DJI.
60. Smith MQP, Ruxton GD. Effective use of the McNemar test. Behav Ecol Sociobiol. 2020;74(11):1–9. https://doi.org/10.1007/s00265-020-02916-y.

Publisher's Note