Chapter 5
Accelerating the Big Data Analytics by GPU-Based Machine Learning: A Survey

K. Bhargavi and B. Sathish Babu

Abstract Today a large volume of structured and unstructured data is being generated online; the main sources for big data are social media profiles, MOOC (massive open online courses) log, social influencer, Internet of Things (IoT) data, the web, transactional applications, stream monitoring technologies, NoSQL (not only structured query language) stored data, log files, legacy document, and so on. There is a need to analyze such huge volume of data at a faster rate by uncovering the hidden patterns and correlation between the data to provide intelligent business decisions with high accuracy. The GPU (graphics processing unit)-enabled machine learning-based techniques are the strongest solution being used to perform big data analytics operation at an accelerated speed. This paper discusses selective GPU-based machine learning algorithms like decision tree, neural network, random forest, Q-learning, SARSA learning, K-means, NB (naive Bayes), AdaBoost, deep learning, support vector machine (SVM), linear regression, logistic regression, Apriori, and HMM (hidden Markov model) being used for big data analysis.

Keywords Machine learning · Big data · GPU · Acceleration · Analytics

5.1 Introduction

The term big data refers to the data made up of three Vs, i.e., high volume, high velocity, and high variety, which cannot be handled by traditional relational database tool. It covers a large volume of data including structured, unstructured, batch processed, or live streaming whose size varies from terabytes to petabytes. Big data analytics is the process of analyzing the huge volume of data to draw

K. Bhargavi (✉)
Department of CSE, Siddaganga Institute of Technology, Tumkur, India

B. S. Babu
Department of CSE, RV College of Engineering, Bengaluru, India
e-mail: bsbabu@rvce.edu.in

N. S. V. Rao et al. (eds.), *Proceedings of International Symposium on Sensor Networks, Systems and Security*, https://doi.org/10.1007/978-3-319-75683-7_5

meaningful inferences by identifying hidden patterns in the existing data. The big data analysis operation involves several implementation level and processing level challenges like increased noise levels in high-dimensional data, computational cost, incomplete data with high degree of uncertainty, heterogeneity in data, improper extraction of hidden patterns from outliers of big data, experimental variation due to statistical inclination exhibited by unstructured data, and so on. The GPUs consisting of thousands of cores with massively parallel architecture are becoming a key tool for accelerating big data analytics to get a proper insight into the existing data to draw meaningful inferences. Today there are GPUs with around 5000 cores which are 32 times faster than conventional CPUs (central processing units), and a single core can have a memory of 192 GB which provides significant support for analysis of huge datasets. MapD is the leading service provider of GPUs for big data analytics [1] (https://www.rtinsights.com/gpu-computing-big-data-visualization-mapd-nvidia/; http://blogs.parc.com/2015/11/the-new-kid-on-the-block-gpu-accelerated-big-data-analytics/).

Many advanced big data analysis techniques like predictive analytics, machine learning, data mining, natural language processing, and so on are being used. Out of all advanced data analysis techniques, machine learning is one of the oldest and popular methods for big data analysis that continuously learn from the past data by recursively building analytical data models. Although the machine learning techniques for data analysis are being used for decades, the availability of a large amount of the social media data and the use of GPU with several thousands of cores for parallel processing changed the conventional look of machine learning. Many machine learning libraries are released to support big data analysis on GPU, which includes BIDMach, Capio, Deepgram, PolyAnalyst, UETorch, Theano, TensorFlow, Meson, Keras, H2O, Dextro, Caffe, nvGRAPH, and so on [2, 3] (https://www.ngdata.com/machine-learning-and-big-data-analytics-the-perfect-marriage/).

5.2 Machine Learning Algorithms

The machine learning algorithms have evolved from the concept of computational learning theory and have the capability to learn from the data without explicit programming by constructing learning models with frequent interaction with the environment. Based on the methodology of learning, the machine learning techniques are grouped into various categories like supervised learning, reinforcement learning, and unsupervised learning. The supervised learning algorithms learn from labeled pair of input-output examples; the reinforcement learning algorithms learn from the sequence of reward and punishment sequences, and the unsupervised learning algorithms learn from an unlabeled pair of input-output examples. The taxonomy representing the classification of machine learning techniques and distribution of algorithms among these categories is depicted in Fig. 5.1.

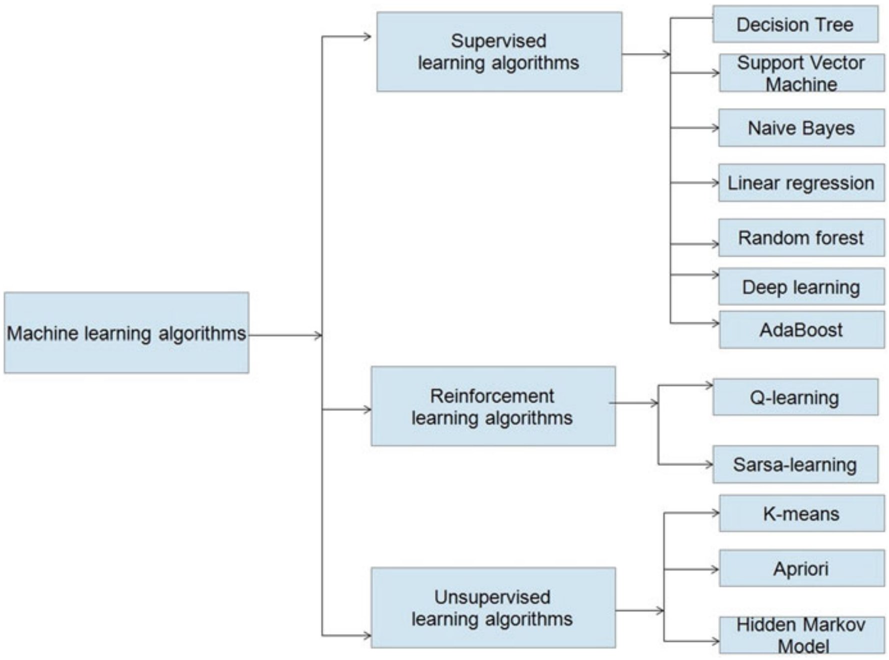

Fig. 5.1 Taxonomy of machine learning algorithms

5.3 GPU Empowering Machine Learning

The machine learning algorithms consist of complex mathematical operations involving a lot of vectors and matrices of very large size. The efficiency of matrix operations on GPU is more when compared to CPU as the GPUs are made up of several cores and each core is capable of spooling multiple threads in parallel. The efficiency of the learning algorithms improves over time by fine-tuning the input parameters; this feature of machine learning algorithm suites well with the parallel architecture of GPU made up of several computational units which are capable enough to perform several floating point operations per second. The GPUs are now being used to train machine learning algorithms with large labeled or unlabeled data samples in multiple levels to do data analysis at high speed with fewer infrastructures. Some of the parallel implementation of machine learning algorithms on GPU are discussed below.

5.3.1 Parallel Decision Tree

It is the supervised algorithm based on tree form of a graph which is used for classification purpose in data mining and analysis applications. The CPU-based implementation of the decision tree algorithm is found to be more time-consuming in nature; hence parallel initiatives of decision tree algorithms have been started. Some of the parallel implementation of decision tree algorithms are SPRINT and CUDAT. The SPRINT basically splits the dataset into several subsets and constructs the decision tree in parallel; the speed of splitting and sequencing the dataset is very high in SPRINT which in turn reduces the memory consumption rate also. The prefix-sum and CUDT stands for (CUDA based Decision Tree) library functions are used in CUDAT to support the parallel implementation of decision tree algorithms. The CUDAT consists of two functions, one is finding the split point in the input dataset, and another is splitting the identified input attribute list. After splitting the identified input attributes, decision trees are constructed and stored in host memory. By storing the tree in host memory, it can be scaled to larger data easily. Compared to SPRINT, CUDAT performance is found to be better as it achieves optimal trade-offs between CPU and GPU computation [4]. The illustration of decision tree construction process by iteratively splitting the dataset is shown in Fig. 5.2. The merits and demerits of parallel implementation of decision tree algorithm are listed below.

Merits

- It works well when the data items exhibit a high degree of correlation.
- The interpretation of tree results is easy.
- The accuracy of the results generated using parallel decision trees is high compared to the sequential ones.

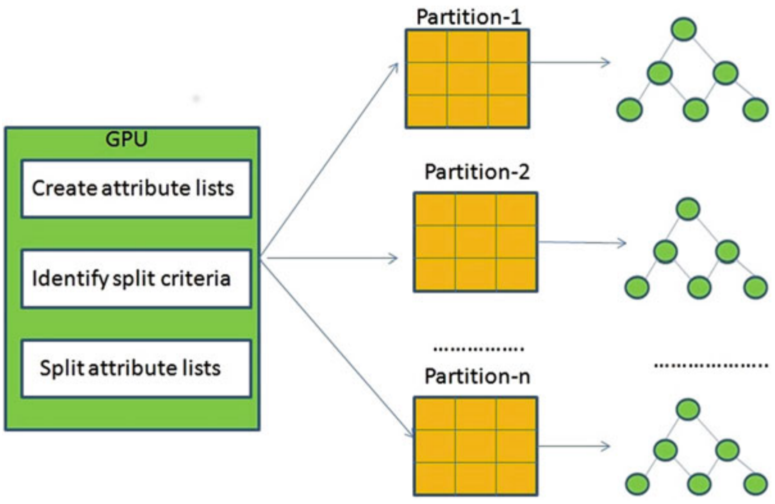

Fig. 5.2 Parallel decision tree

Demerits

- The technique is unstable with respect to the noisy dataset as the synchronization between the trees is less.
- It does linear classification of the dataset; the performance decreases when there is a need to draw soft margin.
- On creation of more bushy trees in parallel, the load imbalance ratio is high.

5.3.2 Parallel Neural Network

It is one of the fastest adaptive learning models with the ability to remember the patterns by adjusting the weight of input variables through block mode backpropagation learning mechanism. Here the parallelism can be achieved in two ways; one is by parallelizing the input data with the help of POSIX threads, and other is parallelizing the data nodes with the help of SIMD (single instruction, multiple data) combined with CUDA (Compute Unified Device Architecture). The GotoBLAS linear algebra library function is used to accelerate the data partitioning and summing up activities.

In input data parallelization method, the input is partitioned into several threads, and the weight synchronization is being carried out periodically for every thread and summed up at last. The GotoBLAS linear algebra library function is used to accelerate the data partitioning and summing up activities. In the case of node parallelization method, the network layers are divided into disjoint partitions of neurons; those neuron outputs are combined to provide output. The CUDA and CUBLAS framework along with CULA library function is used to accelerate the linear and nonlinear operations inside the kernel to achieve node-level parallelism. Figure 5.3 shows the parallel implementation of a neural network using multiple POSIX threads and multiple cores of GPU [5]. The merits and demerits of parallel implementation of parallel neural network are listed below.

Merits

- The neural networks achieve a high degree of parallelism by doing a proper trade-off between CPU and GPU performances.
- The parallel implementation of artificial neural networks with multiple hidden layers makes the analysis process of complex problems easier.

Demerits

- The activation function of the parallel neural network is harder to implement.
- The performance of the parallel neural network is worse compared to its nonparallel implementation when the size of the input is large.

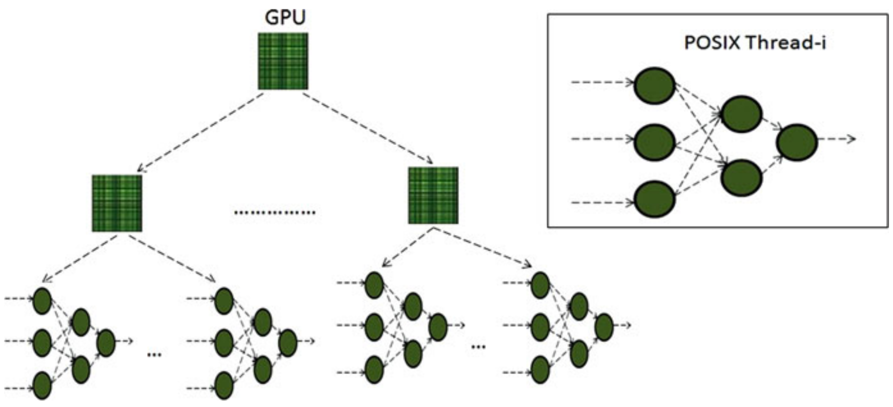

Fig. 5.3 Parallel neural network

5.3.3 *Optimized Random Forest*

Random forest is a supervised machine learning-based classifier; it is a collection of decision trees arranged in the random order used to classify big datasets. The parallelization of individual decision trees is usually carried out in two ways; one is achieving data parallelism through depth first search mechanism, and another is task parallelism through breadth first search mechanism [6].

CudaTree is GPU implementation of random forest which speeds up the decision tree creation process by parallelizing the decision tree creation process on multiple cores of GPU and prevents frequent switching between coarse-grained and fine-grained tasks. BIDMachRF is a GPU-CPU framework of random forest with high scalability, and it is ten times faster than CudaTree. The BIDMachRF works on the principle of achieving maximum parallelism at the microlevel and also removes unnecessary data transfer operation between cores. CURFIL is GPU based random forest used for image labeling which basically predicts the RGB colors in the image and does hyper parameter optimization to achieve higher throughput. Figure 5.4 shows ensemble learning process of a set of decision trees in every core of GPU. The merits and demerits of random forest algorithm are listed below.

Merits

- The over-fitting problem occurrence is less as the summation of several tree outputs are taken into consideration.
- The accuracy of the output achieved is high with the ensemble of many decision trees.
- Practical feasibility is more as it can be used for both classification and regression related jobs.

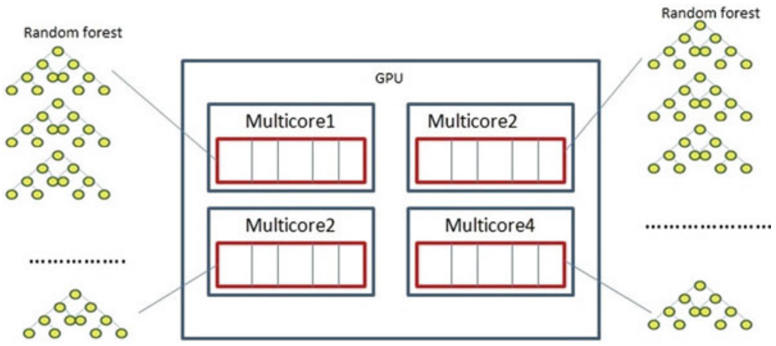

Fig. 5.4 GPU based random forest

Demerits

- Its parallel implementation is tedious as there exists; it requires a lot of synchronization between individual decision trees.
- The training time of the algorithm is high, and the accurate predictions are drawn very slowly.

5.3.4 Deep SARSA Learning

Deep SARSA learner is a State-Action-Reward-State-Action form of reinforcement learning algorithm. It overcomes the Q-learning mechanism as Q-learning always tries to find local optimal solution, whereas SARSA learning takes each and every action of the agent into control and keeps updating its actions to find a globally optimal solution. Julia is a dynamic programming tool which supports SARSA learning through CPU and GPU parallelization with the help of packages like OpenCL.jl, CUDAnative.jl, CUDAdrv.jl, CUDArt.jl, and GPUArrays.jl. To achieve distributed parallelism for SARSA learning among multiple TensorFlow GPUs, OpenAI Gym packages are used [7]. A multiple GPU-based deep SARSA parallel learning environment with stages from Q1 to Qn is depicted in Fig. 5.5. The merits and demerits of SARSA algorithm are listed below.

Merits

- Achieves high speed by employing exploratory behavior policy during the learning stage of the agents.
- Supports asynchronous learning process by doing n step learning activities.
- The usage of locality of state space while computation accelerates the convergence rate of the algorithm.

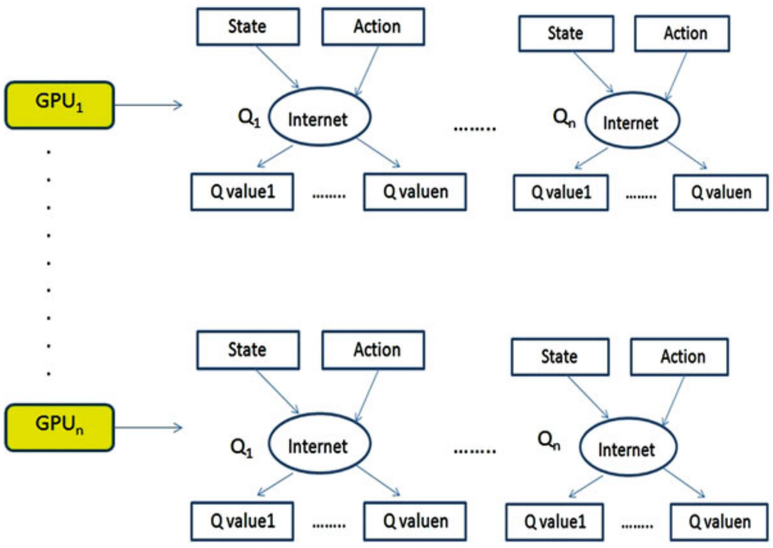

Fig. 5.5 Deep SARSA learning

Demerits

- The communication cost is high due to the exchange of large volume of information between the agents.
- The frequent updating of agent learning state in several cores leads to a deadlock situation.

5.3.5 *Deep Q-Learning*

The deep Q-learning is also another reinforcement learning algorithm used by the agents to perform optimal actions using Markov decision process. The Q-learning process in depth is supported by Deeplearning4j machine learning library, which is popularly referred as RL4J (reinforcement learning 4J) version library. The procedure to train a robot to play Atari game using deep Q-learning was first introduced by Google in 2013 with the help of GPU-based machine learning packages like Keras and Gym [8]. The pictorial representation of a deep Q-learning network with one input layer, two hidden layers, and one output layer is shown in Fig. 5.6; the nodes in the output layer are enabled with Q-learning algorithm. The merits and demerits of Q-learning algorithm are listed below.

Merits

- Accurate decision-making ability as it takes soft actions by exploring best policies.

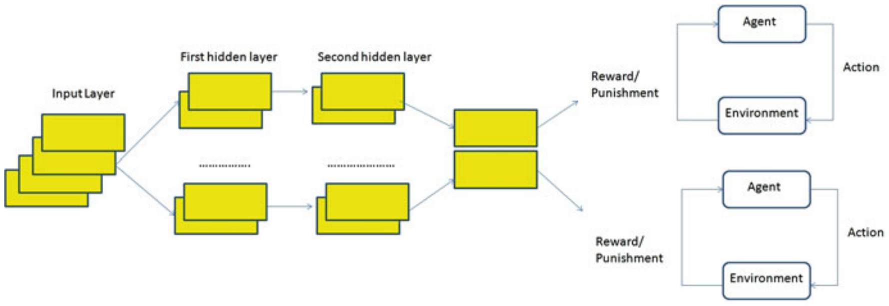

Fig. 5.6 Deep Q-learning

- The learning ability of the agent is flexible as it always tries to find a best optimal solution.
- The parallel implementation of Q-learning agents is successful on continuous and discontinuous input datasets.

Demerits

- Chances of getting trapped into local minimal are more.
- The policies made by the agents are fast but in terms of stability, it is weak.

5.3.6 Parallel K-Means

It is an unsupervised clustering algorithm used to cluster high-dimensional vectors nearly of size 128 dimensions by achieving parallelism at the task level. The K-means implementation on GPU uses master-slave architecture, the master node sends the incoming data to worker nodes by computing centroid, and the worker nodes in parallel compute new centroids to perform data-intensive operation. Here the job of worker nodes is equally distributed among n CPUs in a sequential manner, and their mean is calculated by the master node in a GPU. It also provides differentiated Quality of Service (QoS), by classifying the input dataset as low-dimensional and high-dimensional dataset based on their service requirements. For low-dimensional dataset, the processing is carried out in CPU registers, whereas for high-dimensional dataset both GPU and CPU registers are used. NVIDIA G80 PCI board with CUDA framework supports the parallel implementation of K-means on GPU. Many varieties of K-means algorithms are implemented on CUDA which includes GPU Miner, UVKMeans, and HPKMeans [9]. The implementation of K-means algorithm using master-slave architecture in shared memory of GPU is given in Fig. 5.7. The merits and demerits of parallel implementation of K-means algorithm are listed below.

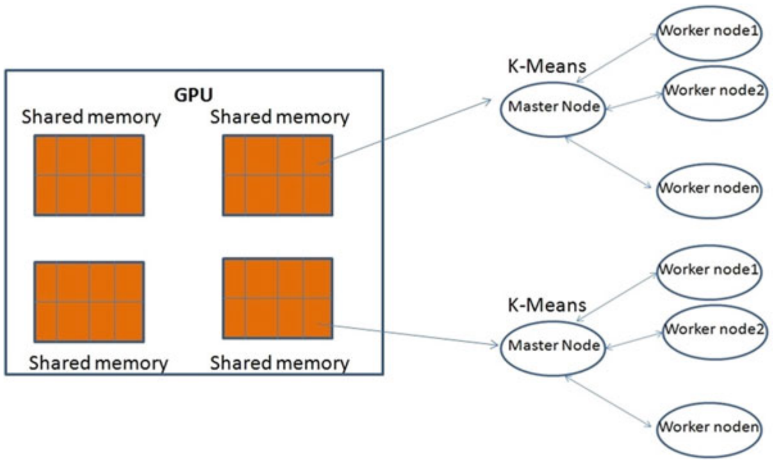

Fig. 5.7 Parallel K-means

Merits

- The multi-core implementation of K-means is simple, fast, and easy to implement.
- Creates the clusters with uniform variance, as a result, the cluster result interpretation is easy.
- The speed of clustering is faster as it recursively divides the input space into disjoint subspaces.

Demerits

- The wrong selection of cluster centers leads to inappropriate results.
- The performance is weak on the nonlinear and noisy dataset.
- Uneven size input partitions lead to the formation of global clusters and predicting K-value for such clusters is difficult.

5.3.7 GPU-NB

GPU-NB is most widely used parallel version of naive Bayes classifier for classifying the documents. It decides the probability of assigning a document to a particular category based on the probabilistic model. Both training and testing phases of the algorithm are parallelized; first, the training phase is parallelized by using GPU kernels which improve the training speed by reducing the data dependency between the training phases. The dense probability matrix is formed by using the learning GPU kernel; at last, the documents are classified to a particular class which exhibited the highest probability. The CUDA kernels are more commonly used for training and classification purpose along with the NVIDIA GeForce GTX 460 GPU cards [10, 11]. The GPU kernel learning and testing phases using NB algorithm over

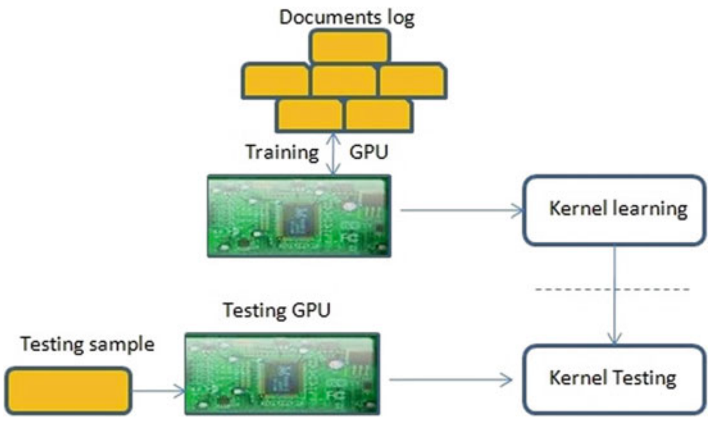

Fig. 5.8 GPU-NB

huge document log are shown in Fig. 5.8. The merits and demerits of parallel implementation of NB algorithm are listed below.

Merits

- It is the simplest form of classifier.
- The performance of the algorithm is good with respect to multi-class problems.
- The classifier robustly handles the bell shaped or spherical shaped input dataset.
- The parallel version of NB is widely accepted classifier for classification of text documents.

Demerits

- The prediction output of NB is weak when there is scarcity of available dataset.
- The practical feasibility of the approach is less as it assumes that the input predictors are totally independent of each other.
- The process of mining continuous features from the dataset for likelihood estimation is not clear.

5.3.8 Accelerated AdaBoost

AdaBoost is used to recognize objects (car, face, pedestrian, iris, fingerprint, and so on) precisely even when the input data sample is huge, highly volatile, and uncertain in nature. The object recognition and classification is carried out at high speed in parallel with the help of GPUs. The training of the algorithm is carried out without any dependencies by modifying the weights of the input samples, and the optimal sample is identified by spooling multiple GPU streams without any overlapping situations [12]. Figure 5.9 illustrates a sample image recognition scenario by mining every pixel of the target object using the AdaBoost algorithm. The merits and demerits of AdaBoost algorithm are listed below.

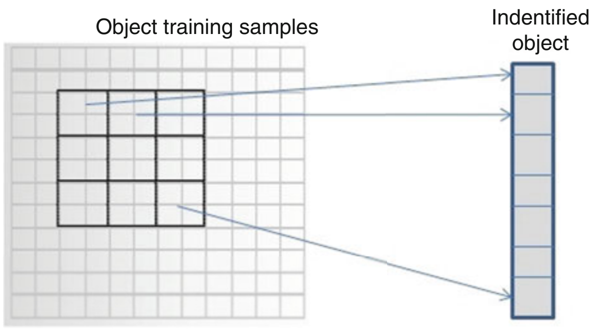

Fig. 5.9 Image recognition using AdaBoost

Merits

- The parallel implementation of the classifier is fast due to its self-adaptive nature.
- It is computationally efficient against illumination changes of the input data.
- The classifier by default uses hierarchical models which increases its computational efficiency.
- The accuracy of the result is high due to fairly good generalization mechanism.

Demerits

- It is cost expensive whenever subjected to large instances of input data samples.
- The parallel implementation is sensitive to outliers and noisy input data.
- The algorithm often gets stuck in suboptimal solutions due to interpolation problems.

5.3.9 Deep Belief Network

It is the process of training a computer in depth with hierarchical models by dividing the original dataset into multiple subsets based on different characteristics exhibited by the subsets. The deep algorithms usually contain many nonlinear operations which are arranged in multiple levels to form deep belief networks. Some of the recent deep learning techniques used to learn complex decision-making tasks are Adadelta, Adagrad, Adam, affine neural network, Alexnet, backpropagation with respect to time, bidirectional recurrent neural network, Deep Dream, GloVe (Global Vectors for Word Representation), Keras, negative log likelihood, and so on. The AlchemyAPI is an IBM-based company, which is the leading provider of deep learning oriented cloud services with many APIs like AlchemyData News API, AlchemyLanguage, AlchemyVision, and so on http://timdettmers.com/2017/04/09/which-gpu-for-deep-learning/. The concept of deep learning in multiple layers with increasing depth levels ranging from one to n in every layer is shown in Fig. 5.10. The merits and demerits of deep belief network are listed below.

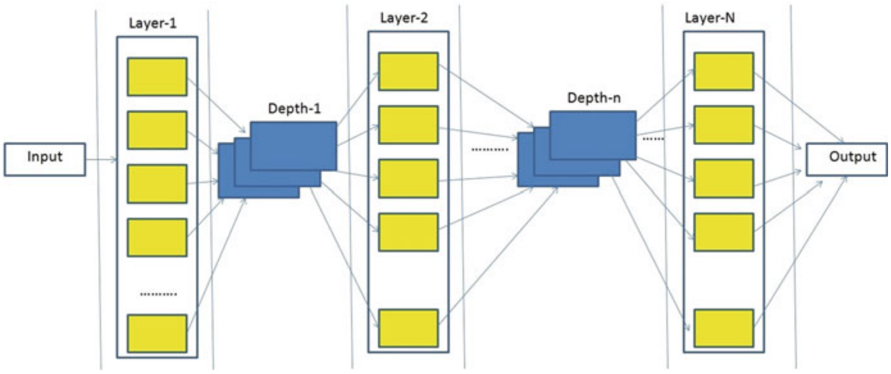

Fig. 5.10 Deep belief network

Merits

- The deep belief networks are structured uniformly with many layers of depth which make it highly suitable for computation-intensive jobs, and they are able to perform billions of computation per unit of time
- The architecture of deep networks suites well with the GPU; the computation speed offered by the combination of GPU and deep network is extremely high.

Demerits

- For accurate predictions, the algorithm needs to be trained with more number of training examples which results in more training time.
- The process of determining the exact values for hyper parameters to train the deep belief network is not transparent.
- Parallelizing deep networks over multiple GPUs is a difficult operation.

5.3.10 Parallel Support Vector Machine

It is one of the powerful supervised machine learning-based classifiers to perform high computation involved jobs. The SVMs are classified into two types: one is binary SVM and multi-class SVM. The binary SVM is used mainly to solve a convex optimization problem using SMO (sequential minimal optimization) algorithm using GPU devices. The SMO algorithms inherently make use of second order WSS (working set selection) logic to do classification at a faster rate. For multi-class classification using SVM, multi-threaded approach with GPU at task level but the memory allocation and launching functions are assigned for CPUs only [13, 14]. The trade-off use of CPU and GPU allows the multiple SVM based classifier to scale at a higher rate. Figure 5.11 depicts the training and testing phases of multiple SVM-based classifiers in a parallel programming environment. The merits and demerits of parallel SVM is listed below.

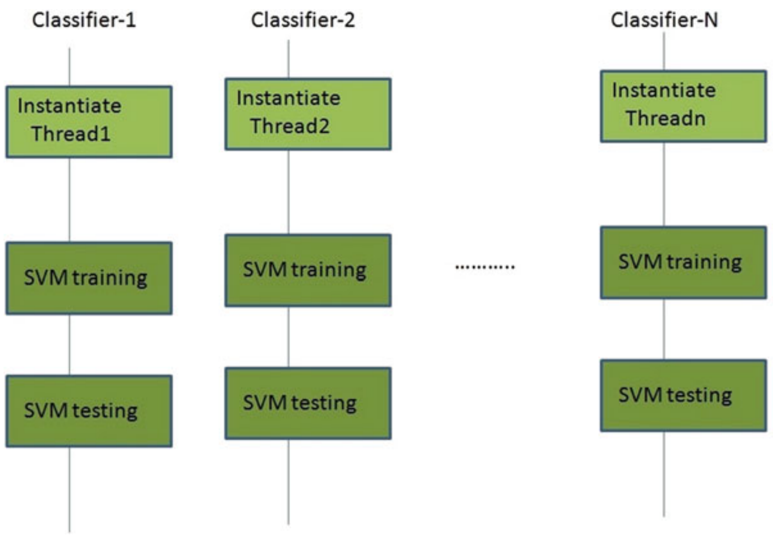

Fig. 5.11 Parallel SVM

Merits

- Delivers accurate novel solution for convex optimal problems.
- With the Gaussian kernel, the simplicity of the algorithm increases, and it evenly separates the input data samples.
- It enforces efficient capacity control scheme among multiple support vector machines by optimizing the margin and reduces the chances of a locally optimal solution.

Demerits

- The interpretation of the result is difficult due to the lack of transparency.
- The learning time of support vector machine is in polynomial times.
- It is suitable for binary classification problems as the approach fails on multi-class problems.
- The misclassification rate is high; it does not consider the history of domain knowledge.

5.3.11 Large Scale Linear Regression

The least median of square model is one of the most commonly used linear regression models for multi-threaded GPU programming. The parallel implementation of linear regression models does mining and analysis operations on the large dataset at a faster rate and achieves high forecasting accuracy also. The GPGPU (general

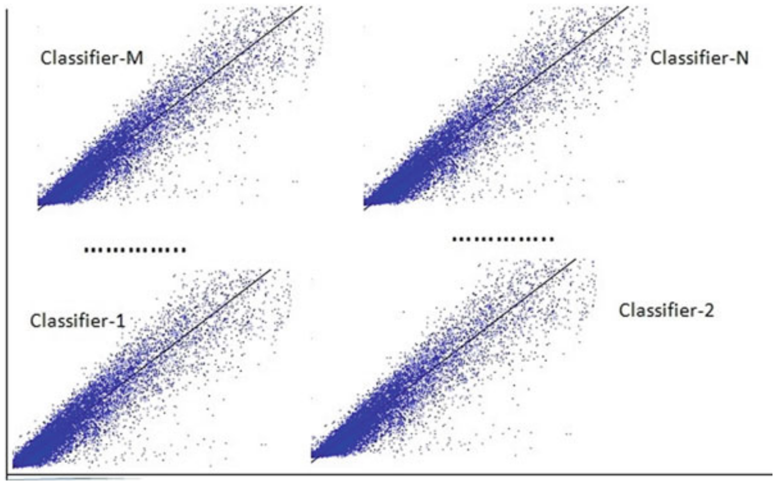

Fig. 5.12 Parallel linear regression

purpose GPU) kernels are most commonly used to handle big data and to solve several floating point arithmetic operations with more precision using machine learning algorithms, and by statistically modeling the power between GPU kernels, the regression operations are carried out robustly even in the presence of several outliers to arrive at timely solutions [15]. The simultaneous operation of multiple linear regression model-based classifiers is illustrated in Fig. 5.12. The merits and demerits of linear regression are listed below.

Merits

- The linear model is simple and easy to implement.
- The forecasting accuracy is high as it efficiently mines the hidden patterns in the dataset.
- Most commonly used modeling tool which can automatically map the N-dimensional datasets.

Demerits

- Its scope is limited to datasets exhibiting a linear relationship as the approach fails on the dependent dataset.
- The accuracy of the prediction goes down with the presence of outliers.
- It can only map the N-dimensional input space to one-dimensional output space, which makes it suitable only for linear separable data items.

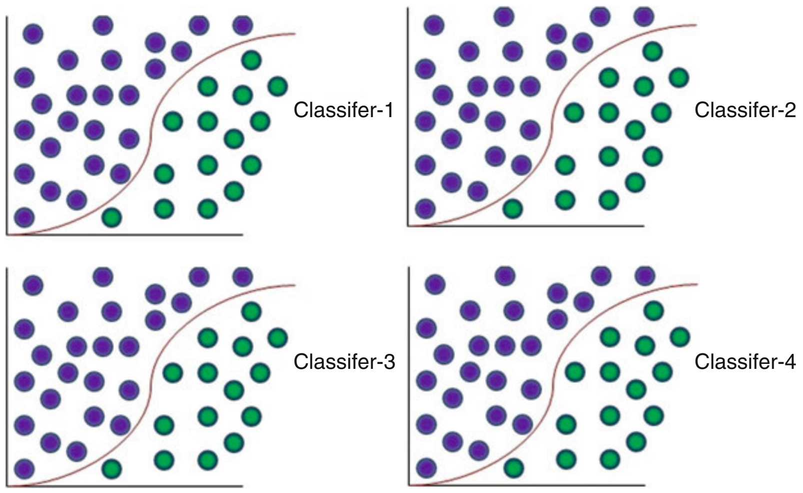

Fig. 5.13 Parallel logistic regression

5.3.12 Large Scale Logistic Regression

Logistic regression is one of the oldest binary classification algorithm used to forecast the chances of occurrence of an event using logit function. The sequential implementation of logistic regression is slow with respect to high-dimensional problems, but the parallel implementation of logistic regression model using GPU exhibits a high degree of suitability towards high-dimensional problems. The CUDNN.torch is the popular GPU-based library which facilitates the parallel implementation of logistic regression using torch learning framework [16, 17]. The simultaneous operation of multiple logistic regression classifiers is depicted in Fig. 5.13. The merits and demerits of logistic regression are listed below.

Merits

- It is capable enough to represent nonlinear relationships between the data items.
- Robust enough to handle the noise and floating point variance in the input as the independent data items are equally distributed among the groups.

Demerits

- The training time of the algorithm is high as it needs to be trained with more examples in order to attain the stable state.
- The prediction accuracy is less as it achieves performance at a lower bound.

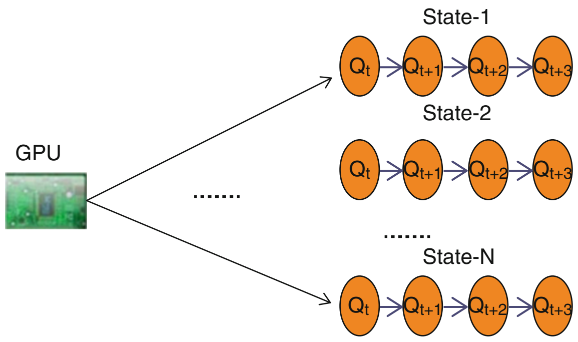

Fig. 5.14 Parallel HMM

5.3.13 Massively Parallel Hidden Markov Model (HMM)

The HMM is usually used to perform sequential classification operations based on Markov property using a set of hidden states, but nowadays the parallel implementation of HMM is used to perform computation-intensive jobs like handwriting recognition, face recognition, signal detection, pattern mining, gene analysis, and so on. The parallel version of HMM-based algorithms is Baum-Welch, Viterbi, and forward, in which the performance of Viterbi and Baum-Welch are found to be good. The CUDA and OpenCL offer general purpose frameworks for HMM applications and for parallel implementation of HMM; the thread blocks are considered where the Markov model is evaluated by considering the forward probability of every thread in each thread block. The HMMER, HMMoC, and GPU-HMMER are the popular tool to create GPU-based hidden Markov models [18]. A working of HMM with N states on every core of a GPU is shown in Fig. 5.14. The merits and demerits of parallel HMM are listed below.

Merits

- The simultaneous execution of multiple Markov models in several cores of GPU increases the rate of completion of jobs.
- The data parallelism is achieved at a higher rate by exploiting the data independence feature of the Markov chain.
- The learning model is most accurate as it can learn from a raw sequence of dataset and exhibit efficient generalization capability even in the presence of variable length input dataset.

Demerits

- It is basically a sequential classifier, parallelizing it takes time.
- The performance is weak with a discrete set of data samples.
- The model fails to capture higher order correlation among the items datasets.
- The computation power required to evaluate the success or failure of the model is too expensive.

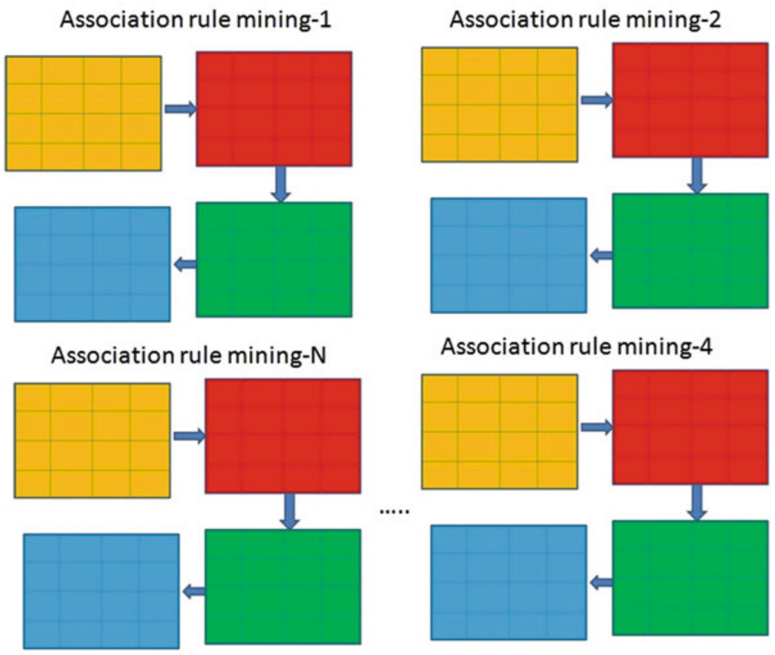
Association rule mining-1
Association rule mining-2
Association rule mining-N
Association rule mining-4
.....

Fig. 5.15 Parallel Apriori

5.3.14 Parallel Apriori

Apriori algorithm is used to determine associated data items in transaction database by mining the hidden patterns among the data items using association rules. The parallel implementation of Apriori algorithm is in CUDA architecture; the algorithm speed increases with the increase in dataset size as it exhibits superior floating point computation capability. The GPApriori is a GPU-based implementation of Apriori on NVIDIA Tesla T10 GPU card, whose speed is 100 times better than sequential Apriori; the high speed is achieved by representing the candidate dataset in the vertical bit set form. OpenCL (Open Compute Language) is the most commonly used Java based GPGPU framework for implementation of Apriori algorithm [19–21]. A simple example showing the parallel association rule mining process of Apriori algorithm is given in Fig. 5.15. The merits and demerits of parallel Apriori algorithm are listed below.

Merits

- The classifier is easily parallelizable.
- The implementation of it is easy.
- Exhibits a high degree of scalability by exploiting spatial locality property of the data items.
- Communication cost is less due to the fewer interaction between processors.

Demerits

- Due to the construction of entire hash tree in every processor, the memory utilization is not optimal.
- Processing overhead increases with the increase in dataset size.
- The tendency of extracting not so formal association rules in two-dimensional databases is more, which decreases the performance of the classifier.

5.4 Accelerated Machine Learning Based Big Data Analytic Applications

Some of the modern big data analytics applications, which use GPU-based machine learning algorithms for acceleration and accuracy are given below.

Modern Facebook Faces Recognition with Deep Learning Earlier to recognize friends in the Facebook, the friend name was supposed to be tagged, but now it is able to automatically recognize the friends and pop up their name by just seeing the photograph. This is achieved by using a nine-layered neural network with around 120 million synaptic weights which are trained with four million images of the Facebook users.

Emerging Twitter Hashtag Prediction The machine learning classifier algorithms like a decision tree, SVM, NB, k-nearest neighbors, and logistic regression are used to predict newly emerging hashtags based on the content features of hashtag string and contextual features of users.

Twitter Users Latent Attributes Prediction The Twitter usually doesn't provide personal information while creating the user account, but with the help of more user information, it is possible to recommend relevant content to users. Supervised machine learning technique based on user LinkedIn profile or social networking website and unsupervised learning technique based on Markov Chain Monte Carlo algorithm are used to predict attributes related to users.

Smartphones Deloitte Global predicts that around 300 million smartphones provide machine learning services by understanding the user of the phone during 2017. The phone uses chips on which low-power machine learning processors run to provide artificial intelligence-based services like profiling the driver behavior to prevent accidents, providing health predictions to the owner, dynamic navigation guidelines while driving, etc.

Driverless Cars Reinforcement learning algorithms are used in autonomous car based on the data collected from the IoT devices to provide services like automatic navigation on road, redirecting the car to hospital if the driver's condition is bad, applying brakes during emergency situations, estimating the cars approaching quickly from behind, playing songs according to the mood of the owner, and so on.

Precision Medicine Unsupervised machine learning algorithms K-means and supervised algorithm like the random forest are most widely used to identify multi-factorial disease and provide treatment by keeping track of the individual lifestyle and gene characteristics. The services offered by precision medicine include finding suitable blood group while donating blood, identifying the type of cancer based on genetic profiles, finding the root cause of disease via diagnostic analysis, etc.

E-Commerce E-commerce is made up of lots of data; supervised machine learning algorithms like decision tree, naive Bayes, and support vector machine are executed on the numerous business-related data to provide smart services like identifying the likelihood of purchasing the items, clustering the customers based on their interest, content-based filtering to offer personalized products, detecting anomaly in the shopping data, and so on.

Natural Language Processing Convoluted or recurrent neural networks are being used to automatically process natural language and have yielded successful output in academic and industrial point of view.

Computer Vision Computer vision enabled with machine learning operations are used to solve image processing problems in an orthogonal manner. Some of the image processing problems solved are recognizing the objects from the noisy text, scaling the image, removal of background clutter, carrying out intra-class variations in the image, identifying analogy of documents, etc.

5.5 Conclusion

The paper provides a brief discussion on the basics of big data analytics, highlights issues while analyzing the big data, and provides a comprehensive discussion on some of the popular GPU-based machine learning algorithms used for accelerated big data analytics applications.

References

1. Hua, F., Zhaoyang, Z., Chanpaul, J.W., Mahmoud, D., Chonggang, W., Honggang, W.: A survey of big data research. IEEE Netw. **29**(5), 6–9 (2015)
2. Acharjya, D.P., Kauser, A.P.: Survey on big data analytics: challenges, open research issues and tools. Int. J. Adv. Comput. Sci. Appl. 7(2), 1–11 (2016)
3. Win-Tsung, L., Yue-Shan, C., Ruey-Kai, S., Chun-Chieh, C., Shyan-Ming, Y.: CUDT: a CUDA based decision tree algorithm. Sci. World J. **2014**, 745640 (2014)
4. Toby, S.: Implementing decision trees and forests on a GPU. In: Computer Vision-ECCV 2008. Lecture Notes in Computer Science, vol. 5305, pp. 595–608. Springer, Berlin (2008)
5. Raghavendra, D.P.: GNeuron: parallel neural networks with GPU. In: International Conference on High Performance Computing, posters (2007)

6. Mitchell, L., Sloan, T.M., Mewissen, M., Ghazal, P., Forster, T., Ptotwski, M., Andtrew, A.S.: A parallel random forest classifier for R. In: Proceedings of the Second International Workshop on Emerging Computational Methods for the Life Sciences (2011)
7. Dongbin, Z., Haitao, W., Shao, K., Yuanheng, Z.: Deep reinforcement learning with experience replay based on SARSA. In: IEEE Symposium Series on Computational Intelligence (SSCI). This work was supported in part by National Natural Science Foundation of China, IEEE (2016)
8. Iuri, F., Stephen, T., Jason, C., Jan, K.: GA3C: GPU-based A3C for deep reinforcement learning. In: 30th Conference on Neural Information Processing Systems (NIPS 2016)
9. Mario, Z., Michael, G.: Accelerating K-means on the graphics processor via CUDA. In: First International Conference on Intensive Applications and Services, IEEE (2009)
10. Lei, Z., Hai, J., Ran, Z., Xiaowen, F.: Effective naive bayes nearest based image classification on GPU. J. Supercomput. **68**(2), 820–848 (2014)
11. Felipe, V., Guilherme, A., Jussara, A., Gabriel, R., Leonardo, R.: GPU-NB: a fast CUDA-based implementation of naive bayes. In: International Symposium on Computer Architecture and High Performance Computing (2013)
12. Pin, Y.T., Yarsun, H., Ching-Te, C., Tsai-Te, C.: Accelerating AdaBoost algorithm using GPU for multi-object recognition. In: IEEE International Symposium on Circuits and Systems (ISCAS) (2015)
13. Bryan, C., Narayanan, S., Kurt, K.: Fast support vector machine training and classification on graphics processors. In: 25th international Conference on Machine Learning. ACM (2008)
14. Quan, L., Jibo, W., Yue, W., Watson, I.A.: GPU accelerated support vector machines for mining high-throughput screening data. J. Chem. Inf. Model. **49**(12), 2718–2725 (2009)
15. Vaibhav, M., Mayank, G: Data regression with normal equation on GPU using CUDA. Int. J. Comput. Sci. Inf. Technol. Secur. **2**(2), 418–422 (2012)
16. John, C.: Extreme machine learning with GPUs. Computer Science Division, University of California, Berkeley (2014)
17. Larsen, A.B.L.: CUDArray: CUDA-based NumPy. DTU Compute Technical Report (2014)
18. Chuan, L.: cuHMM: a CUDA implementation of hidden Markov model training and classica-tion. The Chronicle of Higher Education (2009)
19. Spandana, K., Sirisha, D., Shahida, S.: Parallelizing Apriori algorithm on GPU. Int. J. Comput. Appl. **155**(10), 22–27 (2016)
20. Fan, Z., Yan, Z., Jason, B.: GPApriori: GPU-accelerated frequent itemset mining. In: IEEE International Conference on Cluster Computing (2011)
21. William, A., Fayaz, K., Veerabhadra, B.: HSApriori: high speed association rule mining using apriori based algorithm for GPU. Int. J. Multidiscip. Curr. Res. **2**, 759–763 (2014)