Evolutionary Algorithms with Neural Networks to optimize Big Data Cache

Tracolli Mirco

Ph.D. student in MATHEMATICS, COMPUTER SCIENCE, STATISTICS
at University of Florence and University of Perugia
Tutors: Marco Baioletti, Valentina Poggioni

Research grant holder at INFN section of Perugia
Tutor: Daniele Spiga

Abstract. With this project I want to develop an Artificial Intelligence (AI) for smart data cache orchestrator. This AI will be used to optimize the access to scientific experiment data. These Big Data will be stored in a data-lake environment and they have to be available for different type of tasks. I want to explore the use of Neural Networks (NNs) to optimize the cache, also using the Evolutionary Algorithms. The NNs with memory such as Long Short-Term Memory Networks are the best technique to approach these kinds of data (time series). Another possibility is a Generative Adversarial Network, that could benefit from the features of evolutionary strategies during the learning phase. Reinforcement learning could be exploited to have a real-time agent for the smart cache management. Furthermore, I want to use Coevolution approach on these networks, including the possibility to deploy the algorithm in a distributed system. These techniques will be disclosed and analyzed to improve the performance both in computational time and in the accuracy of the model.

Keywords: Neuroevolution · Coevolution · Evolutionary Algorithms · Recurrent Neural Networks · Long Short Term Memory Networks · Generative Adversarial Network · Reinforcement Learning · data-lake · Cache.

Introduction

Big Data changed the way we store, access and process information. We are distributing data in multiple sources and we still want to use them as a centralized management. This method can raise problems due to the infrastructure network and computational resources, e.g. denial of service problems or timeout issues, more in general latency related issues. Among various techniques and solution, data cache system allows mitigating problems above mentioned.

A geo-distributed cache system could be a possible solution to serve users, particularly When computational resources are dynamically created as in the case of opportunistic computing.

Cache systems could be created manually and configured with preplaced data, but the best solution is an automatic environment that do all that work in

a transparent way for the end users. In this context something that auto-adapt and reacts in the base of the actual state is necessary. For these reasons, Neural Networks (NNs) are one of the main targets of interest.

Furthermore, an approach with Evolutionary algorithms could also be notable. These algorithms that are inspired by biological evolution and solve problems using a metaheuristic approach. They are usually used to find a global optimization to a problem with several constraints.

Such algorithms operate on a population that is evolved with processes like reproduction, mutation, recombination, selection and more, in a way similar to what we see in Nature.

These strategies are also applied in Neuroevolution (NE) field to train Artificial Neural Networks (ANNs). Neuroevolution techniques are suitable methods in research fields like artificial life and evolutionary robotics, especially because they are not only bound to supervised learning algorithms. By searching the space of solutions directly, NE can be applied to reinforcement learning problems, where neural network controllers map observations from the environment directly to actions.

Motivation

Data management in the next few years will be a critic domain because of the increasing size of the data. Several technical solutions already enable transparent data access, however, processing efficiency seems a common issues. Moreover, this effect can only increase as computing power increases. A key to minimize I/O inefficiency would be a highly effective dynamic cache management.

A system of caches can mitigate this problem and get easier the access to data, but it has to be also smarter than a normal cache system. Manage a distributed and on-demand group of cache requires an object that can also overcome unpredictable events without human intervention. Neural Networks (NNs) with memory suit this kind of problem and could be applied for a prototype of smart cache manager.

Special classes of NNs that allow information to persist is necessary. We can find such kind of feature in Recurrent Neural Networks (RNNs). They are born to elaborate sequences (or stream) of data.

Another useful type of RNN are the Long Short-Term Memory networks (LSTM)[6]. These networks are capable of learning long-term dependencies because they are explicitly designed to remember information for long periods of time.

Using NE to evolve an ANN it could be useful to overcome to some stochastic gradient descent limits (SGD). E.g. NE avoids the problem of vanishing error gradients[5] that affect recurrent network learning algorithms.

Besides the most used supervised learning techniques, two different approaches could be used: Generative adversarial networks (GANs) and Reinforcement learning (RL). The first ones are a class of generative models in which a generator is trained to optimize a cost function that is being simultaneously learned. They

belong to unsupervised machine learning and they are capable to generate new data after learning from a training set. These networks could be used to forecast cache states or to create new tests for a simulation.

Reinforcement learning (RL) research has seen a number of significant advances over the past few years. These advances have allowed agents to explore several domains, such as in robotics, and not only virtual worlds like games[8]. This technique could be useful to create a reactive cache that evolves during the time and changing in the base of the user requests.

Among Evolutionary Algorithms, there is a special class of Genetic Algorithms (GAs) that uses a different strategy to evolve: Coevolution. There are principally two different kinds of Coevolution, the cooperative one and the competitive. This method is similar to a natural ecosystem where organisms struggle for resources and survival and, e.g. it is used to face complex behaviors in GAs.

State of the Art

Optimize the content delivery with Artificial Intelligence is not a well defined field but still there are several companies that already do that. Amazon uses AI for Redshift service self-healing; Netflix uses an AI to improve and manage the stream quality; Algorithmia uses a layer of AI to account resources.

A formal model have to be investigated and the classical algorithms such Least Recently Used (LRU), Most Recently Used (MRU), Adaptive Replacement Cache (ARC), can not forecast or guess the user requests. This kind of approach is better suited to Neural Networks and Evolutionary Algorithms.

Regarding the Reinforcement Learning and the new type of networks available we can see in [10] that an evolutionary approach, against all expectations, performs better on some domains and worse on others, but turn out to be a competitive algorithm for RL.

Problem Statement and Contributions

In my master degree thesis work, I used a specific evolutionary algorithm: the Differential Evolution (DE). The work was a part of a larger project named DENN: an experimentation of the DE as an alternative of the Gradient Descent to train Neural Networks in a supervised environment. The project involves more parts and is still in progress[1]. In my thesis, I investigated some boosting techniques used to enhance the training phase of the network. With that work, I could experiment the behavior of such algorithm using real problems, e.g. the handwritten digits recognition, marketing campaigns of a Portuguese banking institution[7], biodegradation of chemicals[7] and gamma telescope images[7].

The results obtained[1] stimulated me to continue in this field, trying to resolve the common problems I found like *(i)* computational time, *(ii)* solution exploitation and *(iii)* algorithm parameters optimization.

Recently, I'm involved in the Worldwide LHC Computing Grid (WLCG) environment because of INFN (National Institute for Nuclear Physics) related

projects. WLCG target is to access data as a data-lake[2][9] to overcome the next storage requirements.

In fact, the amount of data will be too big that it needs to decouple data and CPUs management. Often, the amount of data is too big that the computing centers do not have enough space to contain all experiment data [4][3]. This means that it's necessary to change the paradigm and split the resources: data will be stored on a few (highly controlled) sites and CPUs will be found elsewhere. This type of organization, named data-lake, uses the internet as a low latency bus to connect the computational resources with the data.

This kind of model proposed in WLCG project allows facing the increasing resource request with the current funding used to maintain the platform. Of course, it's needed to manage how the data flows through the computational clusters and for this problem caches come to support the environment. A good managed cache layer allows to reduce the load of main data centers (avoid "Denial of Service"), plus we can apply a predictive provisioning load of data and move hot data close to the users. The AI has to be as independent as possible to not require human intervention.

Research Methodology and Approach

The research project is divided into several phases: analysis, experiment, test and apply. Each phase produces a piece of the final target, but they are not fully sequential. The first steps will be used to study the main use case and to formalize the problem. For this target, I will request access to CMS (Compact Muon Solenoid) experiment environment at LHC (Large Hadron Collider) at CERN. Their log data of the past years would be a good base for the analysis and the model creation. Also, I would ask to have access to INFN national distributed cache data, and other scenario related to the opportunistic computing.

With a proper model, I will generate a prototype to experiment and simulate the smart cache management. I will compare the results with the log data that I already requested and measure the performances. After that, a testbed will be used as platform before the real application in a working environment with real tasks.

All the phases could be iterate more than one time to adapt the model to the desired target. The model will be as much as possible independent of the original data used for the training. There will be created meta-features to give as input to the artificial intelligence. The output will have the same treatment because we want to describe the possible action that this smart cache can do and they have to be personalized in the base of the technology implemented with the cache and not related to a specific software.

Evaluation Plan

Cache access data are strongly time dependent. You have to treat them as a time series. For such kind of input is needed a Neural Network with memory such RNNs and LSTMs.

In my research project, I would explore the use of memory within the Evolutionary Algorithms. The use of GANs could be helpful to generate a correct configuration of data cache and also to simulate future sessions.

The Neural Network model created will be compared with the current algorithms used for cache management and also with the software used in the scientific environment such CMS to manage content delivery.

The first prototype could not have directly an evolutionary approach because I want also verify if cache management could benefit of the most recent AI techniques.

Conclusions

Analyze and manage Big Data is not trivial and a cache system can alleviate the effort to maintain a local storage for the data. A smart orchestration of a distributed cache system allows users to access data more efficiently without taking care of the opportunistic management of the resources.

Furthermore, with modern techniques, it could be possible to create adaptable cache according to the situation, without the human intervention.

References

1. Baioletti, M., Di Bari, G., Poggioni, V., Tracolli, M.: Can differential evolution be an efficient engine to optimize neural networks? In proc. MOD (2017)
2. Bird, I.: Computing for the large hadron collider. Annual Review of Nuclear and Particle Science **61**, 99–118 (2011)
3. Boccali, T., Donvito, G., Diacono, D., Marzulli, G., Pompili, A., Della Ricca, G., Mazzoni, E., Argiro, S., Gregori, D., Grandi, C., et al.: An xrootd italian federation. In: Journal of Physics: Conference Series. vol. 513, p. 042013. IOP Publishing (2014)
4. Gardner, R., Hanushevsky, A., Vukotic, I., Yang, W.: Caching servers for atlas. In: Journal of Physics: Conference Series. vol. 898, p. 062017. IOP Publishing (2017)
5. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. Lichman, M.: UCI machine learning repository (2013), website: http://archive.ics.uci.edu/ml
8. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
9. Robertson, L.: Computing services for lhc: from clusters to grids. In: From the web to the grid and beyond, pp. 69–89. Springer (2011)
10. Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O., Clune, J.: Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567 (2017)