Exploitation vs Exploration in self improving online cognitive systems using Thompson Sampling.

Desislav Bonchev

Johns Hopkins Whiting School of Engineering  
dbonche2@jh.edu

Abstract

Thomspon sampling (1933) is well known and studied heuristics for solving the multi-armed bandit problem by proving a state-of-the-art algorithmic solution of the “Exploit-Explore” dilemma. In the setting of multi-armed bandit problem an intelligent agent is constantly trying to improve its behavior by balancing between its knowledge of the past and exploring the unknown and by this to maximize the expected reward of its actions from a given environment. This scenario fits naturally into any online cognitive system employing cognition for building a strategy for optimal performance by learning from the feedback of its actions and exploring new horizons. However, high level of complexity coming from environmental conditions is a very common challenge for the architecture design of such system and must be addressed by employing non-linear methods for data modeling and dependencies discovery. In this paper we are going to design a reference architecture for building online cognitive systems by using different studies and variations of the Contextual Thompson Sampling (CTS) algorithm. Our work will combine the linear approach proposed by (Agrawal & Goyal 2012), Neuro-linear approach proposed by (Riquelme & Tucker & Snoek 2018) and the multi-agent approach proposed by (Bargiacchi & Libin & Helsen & Roijers & Nowé) into a reusable abstract architecture that scales well in different real-life scenarios.

Introduction

As in any other system in the core of a cognitive system lies the synergy of its components and the connections between them.

[[1]](#footnote-1)It is a collaborative effort of its components towards collective cognition and cognition implies an ability to make inferences about events in the world around you. These events include those from the environment, the cognitive agent itself, its actions, and the consequences of those actions (Vernon). It is the actions consequences that contributes at most for building solid understanding of the environment and establishing strategies for success. Being successful on the other hand implies the ability of finding optimal solutions of the Exploitation-Exploration dilemma by balancing between curiosity and experience.

Curiosity is natural and important part of every real-life cognition but not trivial to implement into any artificial intelligence system where the system design must employ algorithmic approach and mathematical reasoning as solid foundations for its strategies for choosing between exploiting what is known and discovering new opportunities. And yet any cognitive system potential is limited to the potential of its most poorly performing element. Even the best state-of-art algorithm cannot guarantee state-of-art system without proper system architecture and careful selection of its components and the connections between them.

In this paper as a first step, we are going to discuss a state-of-art algorithms in the field of online cognitive systems by looking into the theoretical work already done for solving the so called Multi-armed Bandit problem where an agent must find an optimal strategy for playing the Multi-armed Bandit game, receiving the best possible collective reward. In more specific our work will focus on the Contextual Thompson Sampling (CTS) algorithm and its application to the online cognitive systems design. Our aim is to combine several variations of the algorithm and by this achieve a design that will address environmental contexts with non-linear complexity and multiple agents’ setup where the agents must act in collaborative manner for achieving maximum reward.

As a second step our focus will move on providing strong reference architecture for building online cognitive systems capable of collective cognition based on historical knowledge and collective experience. The goal of this exercise is providing a guidance for building successful designs of systems that implements the algorithmic framework provided in step one in a scalable underlying infrastructure that maps the algorithm requirements to reusable abstract infrastructure. We will discuss and address the architecture significant requirements and quality attributes in generalized manner describing the desired set of components characteristics and the connections between them.

Part 1. Algorithmic framework

Problem setting

Consider autonomous cognitive system setting following (Agrawal & Goyal 2012) definition for contextual multi-armed bandit problem where at time step the system observes environment context and a set of actions A = {}. Given the reward for action comes from unknow probability distribution with mean where is fixed but unknow parameter. Let history we have:

where is optimal.

For every time step from finite but possibly unknown horizon the goal of the system is to pick action which minimizes the total regret R = .

Contextual Thompson Sampling (CTS)

Building a strategy for solving the multi-armed bandit problem lies within the “Exploration-Exploitation dilemma” where at each time step the system must choose whether it needs to relay on its knowledge of the past or explore new possibilities.

Thompson (1933) provides a probabilistic approach for building a strategy where at each time step the system generates samples of from a prior distribution and then selects the best action which minimizes expected , receives a reward and finally computes the posterior distribution used for the next round.

Linear CTS

In the case of Linear CTS described by (Agrawal & Goyal 2012) where we add contextual information coming form the environment, we assume that the likelihood of reward for action at time having context and parameter is given by the pdf of a:

) where with

.

Then let:

where is indicator.

We then consider the prior to be ) which leads to computing the posterior at time as . Following the Bayes rule, we have:

|  |  |
| --- | --- |
| **Algorithm 1.** Linear CTS | |
| , set  **for** **do**  ~ )  and observe reward  **end** **for** | |

Neural Linear CTS

The Algorithm 1. provides efficient strategy for dealing with the exploit-explore dilemma whenever the context fits naturally into a linear model. The main problem of such linear models is their lack of representational power (Riquelme & Tucker & Snoek 2018) for contexts with non-linear complexity.

One possible solution for this challenge is using a non-linear representation of the context by employing DNN. However, re-training a DNN after each time step is unattractive option form a computational perspective. In this paper we follow the Neural Linear CTS approach proposed by (Riquelme & Tucker & Snoek 2018) where a DNN is trained to provide representation of the context . The system then applies CTS to for its decision-making process.

In this new setting the predicted reward for each action is given by . On each subset of time steps the system retrains its DNN using the time steps experiences from history . Let’s update the algorithm with this new approach:

|  |  |
| --- | --- |
| **Algorithm 2.** Neuro - Linear CTS | |
| , set , DNN = trained DNN model  **for** **do**  )  ~ )  and observe reward  **end** **for** | |

Multi-agent Thompson sampling (MATS)

Next, we consider an extension to our problem definition as suggested by (Bargiacchi & Libin & Helsen & Roijers& Nowé) where instead of having a single agent acting within the environment, we have a set of enumerated loosely coupled agents. The set D is factorized into possibly overlapping, subsets of agents . Every agent has its own set of actions and we denote by the set of joint actions. Further we denote the the set of local joint actions for the groups .

Having this setup, our is now a function of the join action instead of a single agent action as it was previously defined. The global reward is decomposed into local rewards functions, i.e., where only depends on the local joint action .

Now we can construct a coordination bipartite graph where set of edges .

At each time step the Multi-agent Thompson sampling algorithm proposed by (Bargiacchi & Libin & Helsen & Roijers& Nowé) draws a sample of from its posterior distribution and the selects joint action that:

Solving this equation brings the challenge of finding optimal actions for agents in possibly overlapping groups. The MATS algorithm proposed by (Bargiacchi & Libin & Helsen & Roijers& Nowé) employs variable elimination (VE) strategy for computing joint action reward without enumerating explicitly over the whole joint action set which guarantees computational complexity that is combinatorial in terms of the width of the graph .

Multi-agent Multi-armed CTS (MACTS)

Finally let’s put all this together and alter the algorithm used further for the needs of this paper combining everything we discovered so far. In **Algorithm 3** we use a combination of Neural CTS for dealing with the context and then MATS for optimizing the multiple agent’s setup. In our new definition the context must be considered from the prism of the joint actions instead of individual agent as it was defined before in the Linear CTX definition. We already discussed the usage of the context representation which remains valid. In this new approach however the parameter is associated to a joint action instead of individual actions.

|  |
| --- |
| **Algorithm 3.** MACTS |
| , set , DNN = trained DNN model,  **for** **do**  )  ~ )  **end** **for** |

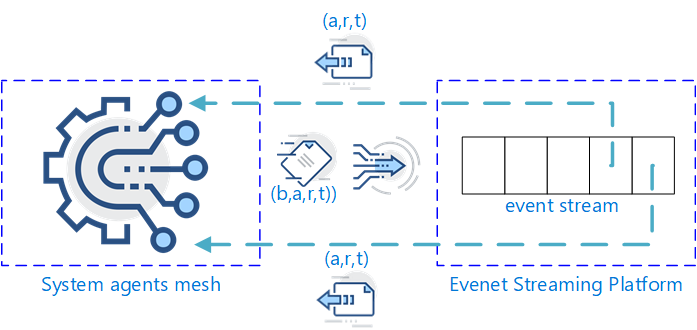
Graphical user interface, application, Word

Description automatically generatedPart 2. System Architecture

**Figure 1** System Architecture

While a detailed performance analysis of the algorithms presented in the first part of this paper is not a subject of this work, we focus our efforts on proposing a reference architecture that prescribes a recipe for building scalable and reusable architectures of an online autonomous cognitive system using the algorithmic framework provided by the MACTS algorithm proposed earlier. **Figure 1** represents the diagram of the proposed architecture.

This reference architecture is an abstract artefact composed of a set of architecture significant capabilities, constraints and quality attributes grouped into components and the connections between them each describing different part of the desired architecture.

 Components of the proposed architecture are defined as follows:

Event Streaming Platform

Traditional architectures designs relying on a single multi purpose persistence layer is no longer sufficient for systems containing high levels of complexity. Different parts of the system require data persistence components with different sets of capabilities and quality attributes. Many modern architecture styles and patterns dealing with Big Data volumes envision architecture that splits the persistence storage into at least two logical groups: one for real-time data access and one for batch processing and analytics. Those two groups come with their own set of attributes, constraints, and desired capabilities.

**Figure 2** Event Streaming Platform

For real-time data persistence, we envision component which allows the system agents to ingest and recover their interactions within the environment in real-time providing high levels of scalability, availability, and performance. The component does not need to provide relational data representations or rich querying and analytics API since data at this point is not relational in its nature – every agent needs to be able to recreate the state of its environment, actions applied to it and rewards obtained.

Our reference architecture adopts the Event Sourcing Architecture pattern (Fowler 2005) for its real-time data persistence layer where the current state of the system can be fully described by its historical events. All state changing events are preserved in a data store in a form of an event stream as demonstrated in **Figure 2**. The event stream store must be capable of ingesting events from multiple system agents by appending the events to the end of the stream. Each agent holds a read pointer for the stream allowing it to read and traverse over the stream on its own pace. Further the platform must be capable of partitioning and replicating the stream for better performance, scalability, and availability. Example of such event streaming platform is the open-source platform **Apache Kafka**.

The architecture implies that all agents acting within the system will push event tuples in the form of . Each agent holds a read pointer for the stream used to recreate or update its history by pulling event tuples in the form of tuples.

Extract Transform Load (ETL) Pipeline

The real-time data persistence layer described above is not sufficient for the purpose of model training, analytics, and business insights. Data coming from this layer is non-relational, denormalized and its solo purpose is serving the needs of the agents of recreating the state of their environment. Before pushing the data for permanent storage and analytics certain data transformations and enrichments must be performed. For this the system must employ a solid ETL Data platform providing scalability, reliability, easy of use and automation.

Building a scalable and reliable ETL Data Pipeline is crucial for every data centric system. Our architecture envisages an ETL platform providing the following capabilities:

* It is essential for the platform to be capable to apply an event streaming technique from the events store for further processing due to the specifics of the real-time persistence layer described previously. Late coming data handling, grouping, and joining are essential capabilities.
* The platform must be capable of providing scalable solution for building the data transformation steps including data cleaning, reshaping, and normalization.
* ETL pipeline is the right place for data enrichments as important part of the data transformation process where additional insights coming from third party sources and machine learning models (anomaly detection, classification, clustering) are incorporated as integral part of the data stream.

Example for such open-source platform is **Apache Spark**.

Data warehousing

Next, since we described the pipeline for turning our real-time data into normalized, transformed, and enriched assets we are ready to discuss the second half of our data persistence layer in the form of data warehousing solution.

Cognitive systems can be characterized as data centric since cognition is built upon data meaning that the data assets collected are of highest importance for the success of the system. However, having these assets persisted is not of a great use without a proper set of capabilities and quality attributes providing specific guaranties and analytical insights.

For data warehousing, we envisage a data store capable of long-term, possibly permanent persistence, of potentially huge amounts of data providing horizontal scalability, distributed storage, availability, data integrity, confidentiality, data-loss prevention, and accessibility. The platform must provide a rich set of querying capabilities on top of large amounts of data with decent performance levels. Data must be stored using both highly normalized schemas as well as demoralized and preloaded views for specific analytical needs.

At the time of this writing there are many open source and commercial solutions providing distributed storage capabilities including **Azure Synapse, Amazon Redshift, Google BigQuery** etc.

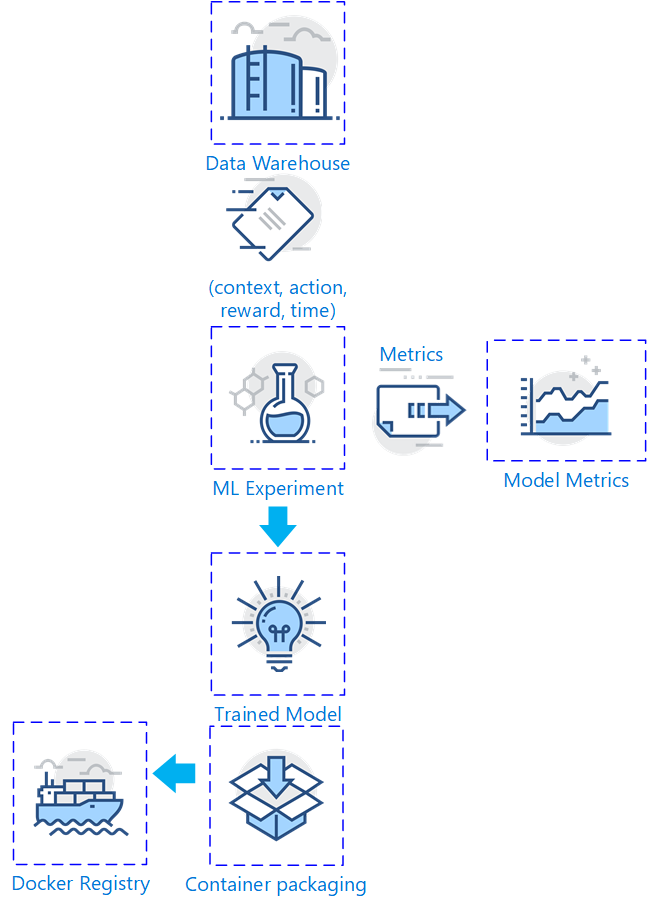
DNN Model training

The usage of a specific type of DNN may vary and depends heavily on the specifics of the concrete use case and implementation of the proposed reference architecture. A CNN model can be employed for visual types of contexts while RNN model is a good fit for textual or sequential cases.

The architecture allows the usage of transfer learning techniques for the purpose of model training by using pretrained models coming from the community and then adding a use case specific layer. For the time of this writing there are many open-source frameworks for building DNNs including **TensorFlow**, **Keras**, **YOLO**, **PyTorch** just to name a few.

In this paper we recommend packaging of the trained model as container image by using any of the available containerization solutions like Docker. Containerization provides great benefits towards deployability, portability and ML Ops best practices.

Once the model is trained and packaged into container image it is being published to container registry from where it can be distributed by automation pipelines to all active agents within the system. **Figure 3** provides a good example of this process.

Model metrics

Evaluation of the model performance metrics during training the models is equally important as the models themselves. The training platforms and machine learning frameworks needs to be chosen carefully for capabilities of providing explainable models. Knowing the performance of the model is essential for building any cognitive system that relays on this model for its decision-making process.

The architecture requires dedicated component for storing, querying, and analyzing the metrics generated during the training process. This platform must provide a specific set of capabilities allowing business stakeholders and domain experts to access the insights needed for the decision-making process. Such capabilities include rich querying features, statistical visualizations, and dashboards.

System agents

The autonomous online setup of our imaginary system is composed of a mesh of agents. Agents observes and interacts with the environment, receive feedback provided because of this interaction and learn from each other as well as from their own behavior.

In this work we suggest that every agent holds a local copy of the DNN model trained by the backend system in the form of container image. The agents per se uses the model as abstract black box and are not concerned with its internals. With this we aim for high levels of decoupling and abstraction and align with the single responsibility principle.

Agents are divided into possibly overlapping groups. As soon as one agent receives a new stimulus in a form of a context from its environment it uses its local DNN copy to create a good representation of the context features. It then uses its local instance of the MACTS for the decision-making process and signals the agents in the group selected to act. After taking specific action the agent from the acting group receives feedback in the form of reward . The agents update their MACTS posterior believes and sends an event to the centralized Event Streaming Store in the form of . All other agents in the mesh subscribed to the event stream receives the event in the form of and each updates its local MACTS posterior. After specific number of times steps the backend system pushes a new retrained DNN image to its agents.

Now as described above the mesh of agents requires a capability for signaling each other to act. Although there are many ways to satisfy this requirement it highly de-pends on the types and capabilities of the agents them-selves. Some agents may not be capable of providing such functionality themselves and may have to relay on a centralized gateway component acting as a cluster manager dispatching the work to the appropriate agents.

**Figure 3** ML Ops Pipeline

Our vision however is that whenever possible agents needs to broadcast the signal by their own in a pass-forward manner meaning that no matter which agent receives the stimulus in a form of new context it calculates the joint action group and broadcast this conclusion to all nearby agents in a form of a signal with specific unique id. The near agents receiving the signal observe if they need to act or not, store locally the id of the signal and then broadcast the signal further. An agent may receive the signal multiple times from different agents but since it stores the id of the signal locally it wont broadcast or act on it again.

Conclusion

In this paper we discussed the importance of embedding curiosity, inside online systems using cognitive skills for their decision-making process. We described how the Thomspon Sampling algorithm can be employed for building strategies for dealing with the “Exploitation-Exploration” dilemma in environments with non-linear complexity. Our work combined number of variations of the algorithm into a single solution that deals with the multi-agent-multi-armed-contextual-bandit problem using deep neural networks for context generalization. We then envisioned a reference architecture as an abstract artefact that generalizes the business case behind online cognitive systems and maps the combined algorithm into real-life system by describing the desired set of capabilities and quality attributes required for autonomous cognitive behavior. We addressed architecture significant requirements of this generalized system by envisioning its components and the connections between them.

As a result of our work, we provided a solid guidance for building reliable, scalable, and reusable implementations of online cognitive systems capable of self-improvement trough delicate balance between historical learning and unknown opportunities exploration.

However, our aim is not to provide detailed performance analysis of the algorithms neither to provide an exact recipe that fits to every use case of online cognitive system but rather to provide abstract vision behind significant drivers for the design process.

References

*N*. Gupta, O. Granmo and A. Agrawala, "Thompson Sampling for Dynamic Multi-armed Bandits," *2011 10th International Conference on Machine Learning and Applications and Workshops, Honolulu, HI, USA, 2011, pp. 484-489, doi: 10.1109/ICMLA.2011.144.*

Nonglei Liu, Anuj Kumar, Wenhai Yang, Benoit Dumoulin, “Explore-Exploit: A Framework for Interactive and Online Learning”, *Facebook Conversational AI*, 1 Hacker Way, Menlo Park, CA 94025 USA

Shipra Agrawal, Navin Goyal, “Analysis of Thompson Sampling for the Multi-armed Bandit Problem”, *JMLR: Workshop and Conference Proceedings vol 23 (2012) 39.1–39.26, 25th Annual Conference on Learning Theory*

Agrawal, Shipra & Goyal, Navin. (2012). Thompson Sampling for Contextual Bandits with Linear Payoffs. *30th International Conference on Machine Learning*, ICML 2013.

Riquelme, Carlos & Tucker, George & Snoek, Jasper. (2018). Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling.

Zahavy, Tom & Mannor, Shie. (2019). Deep Neural Linear Bandits: Overcoming Catastrophic Forgetting through Likelihood Matching.

Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, “A Tutorial on Thompson Sampling”

Timothy Verstraeten, Eugenio Bargiacchi, Pieter J. K. Libin, Jan Helsen, Diederik M. Roijers, Ann Nowé, “Multi-Agent Thompson Sampling for Bandit Applications with Sparse Neighborhood Structures”

Olivier Chapelle, Lihong Li, “An Empirical Evaluation of Thompson Sampling”, Yahoo! Research

David Vernon, “Artificial Cognitive Systems A Primer”, The MIT Press

Martin Fowler, (2005), “Event Sourcing”, *Development of Further Patterns of Enterprise Application Architecture*

Clements, Paul & Garlan, David & Bass, Len & Stafford, Judith & Nord, Robert & Ivers, James & Little, Reed. (2002). Documenting Software Architectures: Views and Beyond.

1. Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved. [↑](#footnote-ref-1)