Optimization within Compucell-3D: a deep learning proof of principle

Gabriel J. Severino

Abstract

In this application, we present a deep learning approach for optimizing bacterial chemotaxis within the CompuCell3D framework, a powerful agent-based simulation platform for studying multi-scale cellular phenomena. Our study focuses on the run-and-tumble model of bacterial motion, which is characterized by periods of straight "runs" interspersed with random "tumbles" that change the cell's direction. We aim to optimize the chemotactic behavior of the bacteria in response to a spatially distributed food source by modulating three key output parameters: the probability of tumbling, the persistence of the cell, and its sensitivity to environmental stimuli. Importantly, this work serves as a foundation and a proof of principle that deep learning techniques as derived from Keras and TensorFlow can be implemented *within* CC3D for a variety of different purposes.

Introduction

From the collective behavior seen in swarms, to the ontogenesis seen in the development of multicellular life, self-organization is a ubiquitous property of biological systems. Seen in many areas of complexity science, self-organization refers to the ability of individual components to interact and organize themselves into higher-level structures without external direction or control. Colloquially, this is often described as leading to emergent properties witnessed in a systam that do not appear to be present at the individual component level. Understanding the principles that govern self-organization will not only be crucial in our understanding of biological phenomena such as tissue regeneration, morphogenesis, and development but will help us to build a more solid foundation in complex systems science generally.

On both experimental and computational fronts, there has been much recent work done to understand the mechanisms or processes underlying self-organization (Levin, 2021a; 2021b; 2012; Moore et al., 2017). For instance, experimentally, researchers have found the role of electrical signaling, gene expression, and mechanical forces including tissue tension all play important roles in the self-organization process (LeGoff and Lecuit, 2016; Levin, 2012; 2021a). One particularly fruitful area of research has been to look into the local neighbor-to-neighbor interactions between cells. Cellular automata have been an effective tool for modeling these interactions and have yielded many insights into local interactions between cells (Graner and Glazier, 1992). Specifically, since its origin in the 1980s, the Cellular Potts Model (CPM) has been widely used to study cell behavior and interactions in a variety of biological contexts, including tissue morphogenesis and tumor growth. The CPM allows for a flexible representation of cells and their interactions and has been used to study a wide range of biological phenomena, such as tissue morphogenesis, tumor growth, and developmental mechanisms (Boas and Merks, 2015; Summers et al., 2012; Gao et al., 2011).

In recent years, the incorporation of machine learning techniques into biological simulations has emerged as a promising avenue for optimizing self-organization processes and further understanding the underlying principles governing them (Butler et al., 2018; Meuwly, 2021; Klaine et al., 2017). Deep learning, a subset of machine learning that focuses on the use of large neural networks, has demonstrated impressive results in various fields such as image recognition, natural language processing, and even drug discovery. By incorporating deep learning algorithms into cellular models like the CPM, we can further refine our understanding of the intricacies involved in cellular self-organization and potentially uncover novel strategies for optimizing their behavior.

In this study, we focus on bacterial chemotaxis, a well-established example of self-organization, within the CompuCell3D (CC3D) framework. We employ deep learning techniques to optimize the chemotactic behavior of the bacteria in response to a spatially distributed food source. To achieve this, we use a deep learning architecture based on the Keras and TensorFlow libraries and implement it *within* the CC3D environment. Our approach serves as a foundation for future research that may expand upon these methods to address other cellular processes and phenomena. In the following sections, we detail the methods, including the implementation of the deep learning architecture and the run-and-tumble model. We then present the results and learning objectives of our application. Lastly, we highlight some of the future avenues of additional research for optimization within CC3D.

Run and Tumble Simulation

Cellular migration plays a crucial role in numerous biological processes, the run-and-tumble behavior seen in bacteria being one such method. This behavior enables cells to explore their surroundings and respond to dynamic chemical fields by alternating between periods of persistent movement (runs) and reorientation (tumbles). We implement this behavior using the CPM in the open-source multicellular modeling software CompuCell3D. The CPM is a lattice-based model that represents cells as collections of connected lattice sites. Each cell has an associated volume and surface area, and the behavior of each cell is governed by an energy function. The energy function determines the behavior of cells in response to various stimuli, such as chemical gradients or mechanical forces, etc. The Hamiltonian for our simulation is then defined as

$$H = \sum_{i,j\ neighbors} J(\tau(\sigma_i), \tau(\sigma_j))(1 - \delta(\sigma_i, \sigma_j)) + \lambda\sum_{\sigma_i}(v(\sigma_i) - V(\sigma_i))^2$$

where *i,j* are lattice sites, σ is a cell at lattice site *i, J* is the adhesion coefficient, τ is the cell type, δ is the Kronecker delta, *v* is the volume, *V* is the target volume, and λ is the strength of the volume constraint. Cell polarization reorientation during tumbling depends on the persistence parameter, defined in our simulation as

$$Persistance = P_0(tan(S \bullet \Delta F) + 1) / 2$$

where P_0 is the base persistence value, *S* is the sensitivity parameter, and *ΔF* is the field change over time. The cell's external potential acting over the cell's center of mass is influenced by the force parameter and the direction of polarization. Additionally, the model modulates the probability to tumble (*PT*) based on the field change in time and the sensitivity parameter, described as

$$PT = PT_0(- tan(S \bullet \Delta F) + 1) / 2$$

where PT_0 is the base probability to tumble. At the start of the simulation, the field object is invoked, and the cells are initialized with random polarization directions, field values, persistence values, and *PT*. The cells store their field history with a memory length of 10. This memory length is used to reduce the effect of Potts's noise in the field change information. At each time step, the run and tumble process occurs by updating the current field value at the cell's center of mass and the field history. The field change during the previous 10-time steps is calculated, and the persistence and PT values are modulated based on the field change using the tanh function.

If a random uniform value is less than the cell's *PT*, the cell tumbles. The cell's polarization direction is updated based on a random angle picked from a von Mises distribution, weighted by the cell's persistence value, as

$$\theta' = \theta + d\theta$$

where *θ i*s the current polarization angle and *dθ* is the random angle. The force vector is then updated based on the cell's new polarization direction using

$$Fx = - F_0 \bullet cos(\theta')$$

$$Fy = - F_0 \bullet sin(\theta')$$

where F_0 is the base force value, and *Fx* and *Fy* are the *x* and y components of the force vector, respectively.

To Train A Cell

Here, we implement a deep learning network using the TensorFlow library to optimize the run-and-tumble behavior of bacteria in response to a chemical gradient. The deep learning network is designed to modulate three key output parameters: the probability of tumbling (PT), the persistence of the cell, which is related to how much the cell will change direction as it tumbles, and its polarization.

To achieve this optimization, we implement a Deep Q-Learning algorithm. This approach combines the advantages of Q-Learning with the power of deep neural networks, enabling the agent to learn complex patterns and relationships between its state and the optimal action to take in a given situation. Specifically, we implement a *simple* version of the basic Q-Learning equation denoted as:

$$New\,Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma\, maxQ'(s',a') - Q(s,a)]$$

where *s, a* are the state and action of a given Q-value, R is the reward for taking an action within a state, α is the learning rate and γ is the discount factor. The cell's state is defined by its field history and the change in chemical concentration over time. We define the state of our cell to be the following:

$$State = Field\ History\ and\ \Delta Field$$

The cell's policy is represented by a neural network, which takes the agent's state as input and outputs the Q-values for each possible action. The action is then chosen based on the highest Q-value:

$$Action = argmax(model.predict(state))$$

The model.predict function takes the state as input and returns a list of Q values, from which the action with the highest Q-value is selected.

The reward function used in this problem is designed to guide the cell toward the highest point in the chemical gradient while also considering the distance penalty. The reward is calculated as follows:

$$Reward = \left(\frac{Field}{max(Field)} - \frac{penalty\ weight}{distance\ penalty}\right)$$

The distance penalty is calculated as distance_to_max / np.sqrt(x_dim2 + y_dim2), where distance_to_max is the distance to the highest concentration. By incorporating both the field value and a distance penalty, the agent is encouraged to move towards regions of higher concentration while also avoiding unnecessary movement.

Implemented in TensorFlow, the neural network consists of an 11 x 64 x 32 x 6 architecture that utilizes ReLU activation with linear activation for the final output layer. It modulates the output parameters (PT, persistence, and polarization) based on the selected action, which in turn influences the cell's behavior in response to the chemical gradient. These output parameters can be increased or decreased depending on the action.

Discussion

In this study, we presented a proof of principle for optimizing bacterial chemotaxis using deep learning techniques within the CC3D framework. Our approach demonstrates the feasibility of implementing deep learning methods in CC3D. By employing deep learning algorithms to optimize the run-and-tumble behavior of bacteria in response to a spatially distributed food source, we pave the way for future research that may leverage these methods to address a variety of cellular processes and phenomena in the context of self-organization. The implementation of the Deep Q-Learning algorithm within the CC3D framework allows us to optimize the chemotactic behavior of the bacteria by modulating key output parameters, such as the probability of tumbling, the persistence of the cell, and its sensitivity to environmental stimuli. Our approach offers a novel perspective on bacterial chemotaxis and highlights the potential of deep learning techniques for optimizing self-organization processes in biological systems.

The main goal of our application is to introduce optimization methods *within* CC3D but this also offers itself a wonderful opportunity to provide an introduction to reinforcement learning to interested users. We highlight the following learner objectives for users interacting with our application.

Learning Objectives/Exercises

Activity 1: Observing Bacterial Run and Tumble Behavior in a Chemical Gradient

a. Watch the untrained CompuCell3D simulation of bacterial run-and-tumble behavior as the bacteria swim up a chemical gradient. Take note of the key features of the behavior, such as the duration and frequency of running and tumbling.

b. Analyze how the bacteria's movement is affected by the chemical gradient. Record your thoughts on the role of chemotaxis in their behavior and how they optimize their search for resources.

c. Observe the differences in individual bacterial trajectories and write down possible reasons for the variability in their movement patterns. What parameters are causing the bacteria to change direction?

Activity 2: Identifying Key Simulation Parameters

a. Review the information provided on the polarity, persistence, sensitivity, and probability of tumbling parameters that were optimized in the CompuCell3D simulation. Record the roles of these parameters in a run-and-tumble behavior.

b. Analyze how each parameter value contributes to the efficiency of the bacteria's chemotactic response within the simulation.

c. Take a moment to compare the behavior between different trained models. Are they all the same?

d. On the steering panel, select the option to start from an untrained model. How does its behavior change over time?

Activity 3: Comparing Simulated Behavior with Real-Life Observations

a. Watch video recordings of real bacteria exhibiting run-and-tumble behavior in response to chemical gradients. Compare and contrast the observed behaviors with the simulated behavior in the CompuCell3D model.

b. Identify and list any discrepancies between the real-life observations and the simulated behavior. Explore possible reasons for these differences, such as limitations in the model or differences in environmental conditions.

c. Investigate recent research articles on bacterial run and tumble behavior and chemotaxis. Write a summary of how the simulation model in CompuCell3D aligns with current scientific understanding.

Activity 4: Reflecting on the Interplay of Biology and Computer Science

a. How might computer simulations, such as the CompuCell3D model shown here, enhance our understanding of complex biological processes like bacterial run-and-tumble behavior? How can different methods in cognitive science help us in our pursuit to understand biological systems?

b. Explore the challenges and limitations associated with developing accurate simulations of biological systems. Consider the role of parameter optimization and deep learning techniques, such as TensorFlow, in addressing these challenges.

c. Reflect on the potential applications of computer simulations in advancing research and innovation in fields such as microbiology, medicine, and biotechnology. Specifically, consider ways in which optimization techniques like this could be used for different biological simulations. Discuss the potential benefits and limitations of these techniques for the scientific community.

Conclusion

Lastly, our study contributes to the growing body of research on the integration of machine learning techniques, particularly deep learning, into biological simulations. The incorporation of such methods has the potential to significantly enhance our understanding of the underlying principles governing self-organization in biological systems. This knowledge, in turn, could be applied to a range of applications, including drug discovery, tissue engineering, and the development of novel therapies for various diseases. Future research should explore alternative deep learning architectures and algorithms, as well as other machine learning techniques, to further optimize both the chemotactic behavior of bacteria and importantly to explore other cellular processes within the CC3D framework. Additionally, the scalability of the proposed approach should be investigated, as well as its applicability to other self-organization phenomena in biological systems. Ultimately, the successful integration of deep learning techniques into CC3D has the potential to transform our understanding of the complex and dynamic processes underlying cellular self-organization.

References

Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., & Walsh, A. (2018). Machine learning for molecular and materials science. Nature, 559(7715), 547-555.

Boas, S. E., & Merks, R. M. (2015). Tip cell overtaking occurs as a side effect of sprouting in computational models of angiogenesis. BMC systems biology, 9, 1-17.

Gao, X., Tangney, M., & Tabirca, S. (2011). A multiscale model for hypoxia-induced avascular tumor growth. In Proceedings of the International Conference on Bioscience, Biochemistry and Bioinformatics (IPCBEE): 26–28 February 2011; Singapore. Volume (Vol. 5, pp. 53-58).

Graner, F., & Glazier, J. A. (1992). Simulation of biological cell sorting using a two-dimensional extended Potts model. Physical review letters, 69(13), 2013.

Klaine, P. V., Imran, M. A., Onireti, O., & Souza, R. D. (2017). A survey of machine learning techniques applied to self-organizing cellular networks. IEEE Communications Surveys & Tutorials, 19(4), 2392-2431.

LeGoff, L., & Lecuit, T. (2016). Mechanical forces and growth in animal tissues. Cold Spring Harbor perspectives in biology, 8(3), a019232.

Levin, M. (2021). Bioelectric signaling: Reprogrammable circuits underlying embryogenesis, regeneration, and cancer. Cell, 184(8), 1971-1989.

Levin M. (2021). Bioelectrical approaches to cancer as a problem of the scaling of the cellular self. Progress in biophysics and molecular biology, 165, 102–113. https://doi.org/10.1016/j.pbiomolbio.2021.04.007

Levin, M. (2012). Morphogenetic fields in embryogenesis, regeneration, and cancer: non-local control of complex patterning. Biosystems, 109(3), 243-261.

Meuwly, M. (2021). Machine learning for chemical reactions. Chemical Reviews, 121(16), 10218-10239.

Moore, D., Walker, S. I., & Levin, M. (2017). Cancer as a disorder of patterning information: computational and biophysical perspectives on the cancer problem. Convergent Science Physical Oncology, 3(4), 043001.

Summers, R., Abdulla, T., Imms, R. A., & Schleich, J. M. (2012). 3D Simulation of an in vitro Epithelial to Mesenchymal Transition. In 5th European Conference of the International Federation for Medical and Biological Engineering: 14–18 September 2011, Budapest, Hungary (pp. 1287-1290). Springer Berlin Heidelberg.