Qeios

Peer Review

Review of: "Integrating Functionalities To A System Via Autoencoder Hippocampus Network"

Laurenz Wiskott[1]

1. Institute for Neural Computation, Faculty of Computer Science, Ruhr Universität Bochum, Germany

You present a sketchy conceptual framework for an RL system that can learn to solve different tasks and store the corresponding parameter vectors. It can thus learn to solve different tasks by first retrieving the corresponding parameter vector and then applying the resulting policy to the task at hand.

The description of fMRI is a bit lengthy, given that the article is relatively short overall and fMRI is not central to it. I guess most readers will know fMRI in any case.

You write, "Autoencoder is more suitable for memorization module over other neural networks because the encoder-decoder architecture of autoencoder make it more convenient and straightforward to retrieve stored information." I disagree: An autoencoder learns a continuous mapping from input to output that optimizes something. It does not learn particular vectors to store. There is another class of neural networks, hetero-associative networks, such as the Willshaw network, that learn discrete key-value associations. Such architectures work better as memories than autoencoders due to their discrete nature. But you leave the question open of where the key should come from. Thus, it is hard to tell what kind of network you would need here.

You write, "The essence of this design lies in utilizing the autoencoder hippocampus network to derive a single policy function, parameterized by a set {W_i}, that is capable of executing actions tailored to various corresponding tasks." I do not understand. You write that the policy is represented by a single vector W. Now you speak of a single policy but with a set of vectors. Sounds like a contradiction to me.

You write, "S=\cal{E}(W)}". What is the calligraphic E? It looks like it is the same as the regular E above. I find this confusing.

doi.org/10.32388/SS8KQM

You write, "The skill vectors within the latent layer of an autoencoder embody the concept of Euclidean distance, where a shorter distance signifies a higher degree of skill relevance and, consequently, a greater similarity in parameter values." Sounds like a nice concept, but it does not necessarily come automatically. You have to train the system in a particular way to get this nice property in the compressed space. What are your ideas about that?

Overall, I feel these are nice but quite sketchy thoughts on the problem of maintaining a skill set in an RL system. Many details are left aside. I think several of the assumptions you make do not work like that, see above, and I thus think that your system will not work the way you anticipate.

Furthermore, while you cite literature on hierarchical RL, you do not mention literature addressing the problem you are discussing here, because I would see a difference between a hierarchical approach and learning quite different tasks that do not depend on each other. Literature on meta-learning in RL systems might be a good entry point.

As far as I see, you also leave out the question of how the system should actually decide which task it is confronted with and which parameter vector to retrieve.

The language could be improved.

Declarations

Potential competing interests: No potential competing interests to declare.

doi.org/10.32388/SS8KQM