Applying Computational Neuroscience Methods to Explore the Selectivities of a Trained Deep Q-Network Playing Atari

Maria Kesa, January, 2020

Computational neuroscience and machine learning-- what benefits can be derived from the cross-talk between these two fields? The mathematical methods that have been developed for studying learning in animals can also be useful for studying how machine learning algorithms learn to achieve their goals or developing entirely new algorithms. Consequently, the cross-talk between these two fields has been intense. For example, the original convolutional neural network called Neocognitron developed by Fukushima in 1979 was based on the experimental work from Huebel and Wiesel, who measured the characteristic responses of neurons in the primary visual cortex of the cat. Reinforcement learning is heavily inspired by conditioning experiments in animals where animals are rewarded or punished based on a sequence of actions they take. The aim of the current project is to contribute to the field of “Explainable AI” by applying mathematical techniques from Computational Neuroscience to study how reinforcement learning algorithms learn to associate stimulus-action pairs with value, the long-term expected reward from taking a particular action in a particular state.

Learning algorithms have surpassed human performance in certain domains, for example beating the World Champion at the game of GO, the state space of which is larger than the number of atoms in the Universe. These algorithms often find original ways to solve problems that are clever and unorthodox. But can we learn to understand why they operate as they do and to explain how they learn from experience? This question becomes particularly pertinent as these methods gain widespread use in the industry and it becomes increasingly important to justify their decisions and make sure that the models don’t go against any ethical principles. This is why for my capstone project I decided to apply state-of-the-art methods from Computational Neuroscience to study how a deep reinforcement learning agent, encodes information and develops selectivities for particular game states in order to evaluate the value of the state it’s in.

The DQN reinforcement learning algorithm (Mnih et al, 2015) was ground-breaking because it devised a way to do reinforcement learning in situations with very large state spaces, where using a tabular dynamic programming method fails because the storage of state-action pair values simply becomes too prohibitive. It uses a universal function approximator, a neural network, to pass a game state through it and predict the q-value, a value of a state-action pair. Using the neural network as a data structure permits storing the logic of the game and rewards in the weights of the neural network-- a much more economical solution than amassing a comprehensive database of state-action value pairs. In this project, my plan is to train a DQN (Mnih et al, 2015) agent, consisting of a neural network and a reinforcement learning algorithm, on an Atari game available for experimentation from the OpenAI Gym and use this learned model to study how the algorithm achieves its goals.

The problem that I intend to approach in this project is understanding how the convolutional neural network in the DQN algorithm develops selectivities towards particular elements of the

game. The idea is to train the algorithm until it produces good results (this is an evaluation criterium) and then freeze the model and let it play the game for at least 10,000 frames and record the activations (feature maps) in the neural network as the DQN is playing the game. We would then reduce the dimensionality of the neural network responses in each layer and in all the layers in aggregate using unsupervised algorithms, Principal Component Analysis as a benchmark and EnsemblePursuit as a novel algorithm developed for clustering high-dimensional neural measurements to be evaluated (Kesa, Stringer, Pachitariu, 2019). The dimensionality reduction algorithms will give time-courses of components of the activity in the convolutional neural network in response to the situations in the game screen. Thus the game screen is a stimulus being presented to the neural network, forming a time-series of game screens and we have a neural response of the network components as the targets. We can then fit a receptive field model to this data to see what in the stimulus makes components of neurons activate. Fitting a receptive field is nothing more than carrying out ridge regression, regressing the stimuli to the neural responses (Dayan and Abbott, 2001). The regression weights form the receptive fields with the higher weights corresponding to the stimulus that activated a neuron or a cluster of neurons.

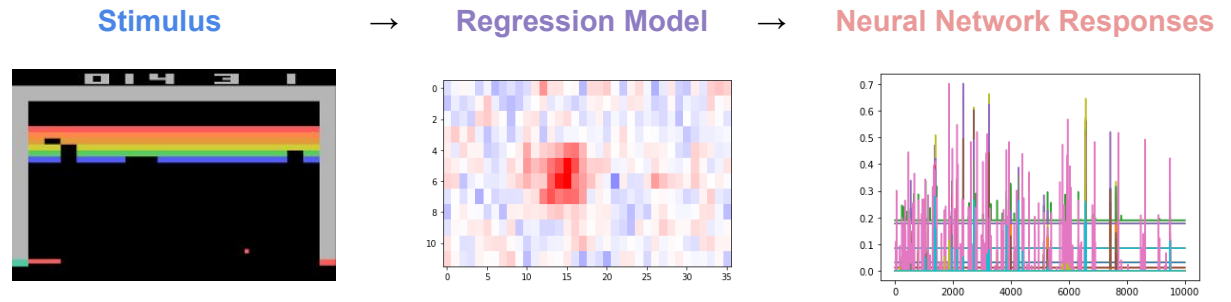

Figure 1. We fit a time-series of game images through a ridge regression model to predict the observed DQN neural network responses. Regression coefficients form a “receptive field”.

To benchmark the model we will train it in a similar procedure on MNIST digits, where a convolutional neural network will predict not values of state-action pairs, but the class of the digit. We will apply the same dimensionality reduction procedure to the neural activations on the test set and fit a regression model to obtain receptive fields for neural components. We will evaluate the shapes and properties of the receptive fields by visual inspection. It will be interesting to compare how the selectivities of the convolutional neural network differ when it is trained to predict values rather than discrete classes.

To evaluate the performance of the algorithm, as a first control we will evaluate the performance of the DQN on an Atari game it was trained on. After fitting the receptive field models, we will evaluate the variance explained of ridge regression, i.e. the amount of information in the neural network response regression predicts from the stimulus (using

sklearn.variance_explained_score). To calculate the variance explained we will split collected data into a training and test set (both coming from timepoints that have been collected after the network was trained, i.e. letting it run in free form). We will also build a JavaScript interface to view the receptive fields of the neural components and animations to quantify what the neural network "sees" in the game as in (Hilton et al, 2020). Making an explorable interface for visualizing how the DQN network makes sense of the game it plays is a step towards understanding how the algorithm works and will bring us a tiny bit closer to Explainable AI.

As a summary, we propose to use machine learning methods that are in use in Computational Neuroscience to understand the feature maps in a deep convolutional neural network trained to play an Atari game (DQN). As a benchmark we first train a convolutional neural network to predict the classes of the MNIST images. We collect the activations of the artificial neurons to the test set, do dimensionality reduction and compute the receptive fields of the reduced neural components. These analyses will be performed by splitting the data, training on the training set and testing on the test set. These receptive fields and their variance explained will be consequently compared to the DQN receptive fields and variance explained values. We first train a DQN and make sure that it achieves good accuracy in the game. We then let a trained network play rounds of games and collect the activations from the convolutional neural network and the game images (stimuli) that evoked them. We split the obtained data into a training and test set. We use dimensionality reduction algorithms (both a traditional algorithm, Principal Components Analysis and a recent state-of-the-art algorithm, EnsemblePursuit) to organize the neural activations into correlated components (fit the unsupervised model on the training set and transform the test set data according to the trained model). We fit a regression model from the time-series of game images to the neural component activity to see what each neural component is sensitive to in the game (i.e. the elements of the images that evoke a neural network response) on the training set and evaluate the predictive power of the model on the test set using variance explained and correlation. Lastly, we will create a browsable interface to explore the activations and receptive fields (regression coefficients) of the network using JavaScript.

References

Mnih et al, 2015, "Human-level control through deep reinforcement learning", Nature

Kesa, Stringer, Pachitariu, 2019, "EnsemblePursuit: an algorithm for finding overlapping clusters of correlated neuros in large-scale recordings", Statistical Analysis of Neural Data workshop, Carnegie Mellon, https://github.com/MouseLand/EnsemblePursuit

Dayan and Abott, 2001, "Theoretical Neuroscience", MIT Press

Hilton et al, 2020, "Understanding RL Vision", Distill, https://distill.pub/2020/understanding-rl-vision/