Further ideas

Genetic algorithms

If you go to the ANNA Git repository and switch to the master branch, you will see the src/main/scala/anna/epengine folder with a lot of code dedicated to mutate and cross networks, but these experiments so far were focused on neurons – rewiring them, changing weights, changing thresholds, etc. – and not on blocks of neurons. I plan to build a way to describe the network as a sequence of “create” and “connect” orders, just as I described it in the main article in the “Learning” chapter, and refactor this code, so it can work on that. The goal is to recreate the S.O.S. network from a simple template consisting of one input and two output neurons, where neuron blocks will be added and modified by a genetic algorithm.

Graphical User Interface

Simply for presentation purposes: I thought it would be interesting to be able to send Morse codes by hitting the spacebar and see the results – both logs from the network and the output as it is generated – in separate text panels. The GUI might be either in the form of a window, made in Java Swing library for example, or some sort of a command-line interface with a few separate panels.

Implementation of other letters in the Morse code

Also for presentation purposes, but codes consisting of both dots and lines, where the order of short and long signals is important, will require a new type of a neuron block. Together with GUI it will be a good device for showing how an asynchronous network works. Anyone will be able to run it, type in some text using only the spacebar, and see how their input is processed into letters.

Forgetting functionality

One of the properties of Spiking Neural Networks is that if a neuron is stimulated by a signal, but not enough to activate it, the signal will diminish in time. In case of an asynchronous network discussed in the article it can be achieved by adding to the neuron a function which would slowly reduce the value in the buffer, from the moment of the last signal to the moment when the value in the buffer is again equal to 0. This functionality is not needed for the S.O.S. network described in the article, but it could be used in order to build blocks which would recognize signals more precisely. For example, the S.O.S. network recognizes an “s” no matter how long are the time gaps between three consecutive short signals, or even if there will be a long signal between them. The only condition is that three signals recognized as dots should appear before three signals recognized as lines. With the forgetting functionality we could build a version of a Signal Sum block which would forget its internal state after a given number of iterations, so if the time gap between three signals recognized as dots was too big, it would not recognize it as “s”.

Naive and non-naive of neuron blocks

More generally, we can say that the DOT and LINE blocks used in the S.O.S. example are “naive” in the sense that they will try to recognize the incoming signal when the positive conditions are fulfilled: any signal coming to DOT will be recognized as a dot after a certain number of iterations; and any two signals coming to LINE, no matter how big is the time gap between them, will be recognized as a line. The two have to communicate with each other through their silencing neurons in order to achieve an agreement.

In the S.O.S. example the LINE block has a priority over the DOT block, implemented as a longer delay for DOT, so LINE has time to send its silencing request. Therefore, we can describe the interaction between them as “if the input sequence contains two signals, recognize a line (LINE); or during the third iteration recognize a dot (DOT)”.

Please note that the description does not say anything about when not to recognize the signal. Any non-empty signal sequence will be recognized as either dot or line. This is what I define as a naive neuron block – a block whose description of functionality lacks negative clauses. Non-naive neuron blocks would contain such clauses. For example, the functionality of a version of LINE with the afore-

mentioned forgetting function could be described as “if the input sequence contains two signals, with the time gap between them not longer than T iterations, recognize a line”.

Integration with Akka Streams

Currently the network works in fact on strings, not real streams. I would like to rebuild the API to allow a better integration with Akka Streams.

Maciej Gorywoda (gorywodamaciej@gmail.com)

November 12, 2016