Merks - a small learning world

Bruno Sciolla

July 8, 2018

1 Current Merk

The world of merks contains individuals called merks. Each has a genome, a neural network, and a physical existence in the stage, that describes where the merk is, and what it does.

1.1 Neurons

A neuron with index $i \in [0, N]$ is connected to other neurons via links stored in a (signed) adjacency matrix l_{ij}. $l_{ij} = \{-1, 0, 1\}$. A neuron has an activation value a_i. There are three types of neurons: **sensor** neurons, **action** neurons and normal neurons. Each neuron also has a binary label L_i.

At each time step the activation is updated according to:

$$a_i(t+1) = \sigma\left(\sum_j l_{ij} a_j(t) + s_i(t)\right) \tag{1}$$

where $\sigma(x) = \{-1, x, 1\}$ on the intervals $[-\infty, -1]$, $[-1, 1]$, $[1, \infty]$ respectively. The external sensor contribution $s_i(t)$ is added if the neuron is a sensor neuron, the value of which is fixed by the stage (can be any external impulse like bumping on a wall, the proximity to resources, or any 'physical sense' that the merk can have of the world).

1.2 Actions

If the neuron is an action neuron, its activation value $a_i(t)$ results in an action on the stage. For example, if the forward action neuron has activation value 1, the merk steps forward. Other possible actions are turning left or right, jumping, kicking, attempting to kill or reproduce, etc.

1.3 Genome

The genome is a binary sequence of fixed universal length. $b = \{011000101110 \cdots\}$, typically 1000 binary numbers.

1.4 Rules

The genome is read by a sequence reader to produce rules of the form:

$$\texttt{Predecessor} \rightarrow \texttt{Successor}, \texttt{Extra}, \texttt{Codes} \tag{2}$$
$$\text{Example} : 010 \rightarrow \{\}, 00, 0111 \tag{3}$$

The predecessor, the successor and extra are three binary label. In the example, 010, an empty label $\{\}$, and 00. The codes block is a succession of 4 bits.

1.5 From genome to rules

The genome is read in order, starting from the beginning of the sequence.

A valid rule starts with a 11. The next segment defines `Predecessor`, the next segment defines `Successor`, the next segment defines `Extra` and finally 4 bits are used to define `Codes`. Because segments do not have a fixed length, a special rule is used to define a segment. The rule is that the odd rank bites (first, third, fifth, etc.) define the length of the segment: the segment extends as far as there are zeros in odd bites. The even bites define the actual value of the segment. So for example:

$$1 \Rightarrow \{\} \tag{4}$$
$$001 \Rightarrow 0 \tag{5}$$
$$011 \Rightarrow 1 \tag{6}$$
$$0a0b1 \Rightarrow ab \tag{7}$$
$$0a0b0c1 \Rightarrow abc \tag{8}$$
$$0a0b0c0d1 \Rightarrow abcd \tag{9}$$
$$\dots$$

The genome is traversed to produce rules using non-overlapping segments. Only some rules are kept: a valid rule must have a nonempty `Predecessor` and either `Successor` or `Extra` must be nonempty. A label longer than 6 bits is truncated to a length of 6 bits. Otherwise no limitations are applied.

1.6 From rules to network

The network is initialized with 3 neurons labelled $001, 000, 01$ and links $001 \xrightarrow{1} 000$, $000 \xrightarrow{1} 01$. The number above the arrow means that the link value is 1 in the adjacency matrix, e.g. $l_{01} = 1$. Notice that several neurons can have the same label in the following.

The network is built, iterating over each neuron. For each neuron `Neuron`, the `Predecessor` of the rule matching the label of the neuron best is found. Basically matching means that every bit of the rule agrees from left to right with the corresponding bit of the label. Thus the rule can be at most as long as the label to match. For example for a label `Neuron` $= 001$:

rule with `Predecessor` 0 matches
rule with `Predecessor` 1 does not match
rule with `Predecessor` 00 matches
rule with `Predecessor` 01 does not match
rule with `Predecessor` 001 matches
rule with `Predecessor` 0011 does not match

So for each neuron `Neuron`, one rule is choosed, the one with longest `Predecessor` that matches the label. If there is no rule, we go to the next neuron. If there is a rule, the network is modified as follows. If label `Extra` is empty, the neuron is just labelled with a new name: `Successor`. If label `Successor` is empty, then let `Neighbors` be every neuron whose name matches `Extra`. Then if the bit `Codes`[1] is 1, then a link is made `Neuron` $\xrightarrow{c}$ `Neighbors` where $c = 1$ if `Codes`$[2] = 0$ and $c = -1$ if `Codes`$[2] = 1$. Also, if the bit `Codes`[3] is 1, then a link is made `Neighbors` $\xrightarrow{c}$ `Neuron` where $c = 1$ if `Codes`$[4] = 0$ and $c = -1$ if `Codes`$[4] = 1$. If both `Successor` and `Extra` are nonempty, a new neuron with label `Extra` is created. The new neuron is created with every link that `Neuron` had with other neurons. Afterwards, `Neuron` is renamed to `Successor`. Also, a link `Neuron` $\xrightarrow{c_2}$ `Extra` is created if `Codes`$[1] = 1$ and a link `Extra` $\xrightarrow{c_4}$ `Neuron` if `Codes`$[3] = 1$ with signs c_2 and c_4 fixed by the bits `Codes`[2] and `Codes`[4] respectively.