Genifer 7.0 — white paper

Yan King Yin general.intelligence@gmail.com

March 23, 2019

Top-level architecture = reinforcement learning. This is explained in the my paper *Wandering in the Labyrinth of Thinking*.

Inside the RL model:

- state = mental state = set of logic propositions
- environment = state space = mental space
- actions = logic rules

Basically, an action = a logic rule is of the form:

$$\mathsf{xxx} \wedge \mathsf{xxx} \wedge .... \Rightarrow \mathsf{xxx} \tag{1}$$

where xxx denotes a logic **proposition**.

Each proposition is a composition of 3 atomic concepts (think of these as word vectors as in Word2Vec):

$$\text{proposition} = \mathsf{xxx} = \mathsf{x} \cdot \mathsf{x} \cdot \mathsf{x}. \tag{2}$$

$\mathsf{x} \in \mathbb{R}^n$ where n is the dimension of a single word-vector (or atomic concept).

An **action** is the conclusion of a rule, ie, the right-hand side of (1).

We use a "free" neural network (ie, standard feed-forward NN) to approximate the set of all rules.

The **input** of the NN would be the state vector:

$$\mathsf{xxx}_1 \wedge \mathsf{xxx}_2 \wedge ....\mathsf{xxx}_m \tag{3}$$

where we fix the number of conjunctions to be m.

The **output** of the NN would be the **probability** of an action:

$$p(\mathsf{xxx}). \tag{4}$$

Note that we don't just want the action itself, we need the **probability distribution** over these actions. The **Bellman update** of reinforcement learning should update the probability distribution over such actions.