Apprentice Learner - Memory Overview

Downloading and Use

The current implementation of the memory mechanism is available here. In order to use the memory mechanism, modify your JSON file to include the agent-level argument `use_memory: True` (see `generate_json.py`, and then run AL as normal. Currently, the agent relies on specific features in the BRD, in order to account for initial activation level based on question type.

Description of Memory Mechanism

The memory mechanism is based on the ACT-R modeling system. In particular, the memory mechanism in AL replicates the activation and decay models described in the Pavlik & Anderson paper. The activation for a given skill at time n is determined by the equation:

$$m_n = \beta_i + \ln\left(\sum_{k=1}^{n} t_k^{-d_k}\right)$$

where t_k is the age of the k^{th} trial, d_k is the decay for the k^{th} trial, and β_i represents differences in activation increase between different types of questions.

Decay for a a given skill at time k is given by the equation:

$$d_k = ce^{m_{k-1}} + \alpha$$

where c is a decay scale parameter, m_{k-1} is the activation level of the previous trial, and α is the intercept of the decay function.

The probability that a skill will be recalled during testing is dependent on its activation:

$$p(m) = \frac{1}{1 + e^{\frac{\tau - m}{s}}}$$

where τ is a threshold parameter, and s controls the sensitivity of recall to changes in activation.

Implementation

Because the decay function used in this model is recursive and depends on previous activations, the activations for every skill are stored upon every time step (time increases whenever AL takes an action). Activation for a specific skill increases whenever AL uses it on a problem, or requests a bottom-out hint. The activation increase depends on a constant coefficient, β_i. Currently, $\beta_i = 0$ for retrieval practice questions, and $\beta_i = 4$ for worked examples, as we hypothesize that completing worked examples causes higher initial activation than retrieval practice, but causes faster decay due to the recursive nature of the model. The decay scale parameter is the same for both types of questions, $c = 0.277$. The activations for the other skills that AL has learned is then recalculated, according to the decay function (in this experiment, α = 0.177).

When the agent requests an answer/action from AL, the where/when learners return a list of applicable skills for the problem state. The probability of AL recalling the first skill in the list is calculated according to the equation above, with $\tau = -0.7, s = 1$. If the skill is retrieved, AL performs the action (and possibly receives feedback depending on the question type, triggering

activation recalculation). If the skill is not retrieved, AL attempts to retrieve each subsequent skill in the list. If no skills are retrieved, AL requests a bottom-out hint from the tutor.

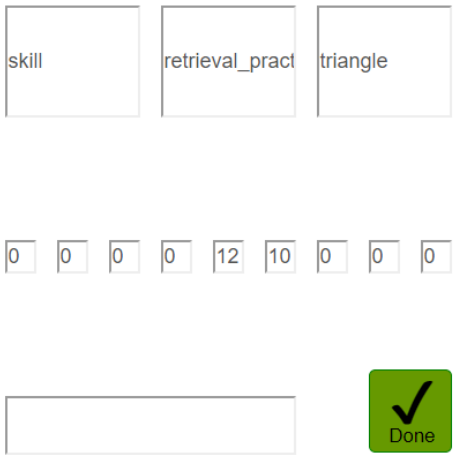
Code Locations

The activation and decay models is implemented within the `train` function in the `ModularAgent`. The `activations` dictionary keeps track of sequential activations for each skill, and the `exp_inds` dictionary keeps track of the times at which each skill is activated. Upon first activation of a skill, it is added to `activations`, and initialized with an activation of `-Inf`. This is needed, as decay depends on the previous activation (with $m_0 =$ `-Inf`, $d_1 = \alpha$). Whenever a skill is activated, the current time is logged in the `exp_inds` dictionary. Afterwards, activation (including decay) is recalculated for all skills using the recursive model, with age for each skill given by `self.t - self.exp_inds[str_exp]` (note that this produces an array for each skill), and time is incremented.

The β_i coefficient is determined by examining a specific element in the state, and changing it based on the value: `state["?ele-practice_type"]["value"]`. This is specific to this project.

Retrieval is calculated in the `request` function.

BRD information



Agent Name:R_3pyDfCcyZTbcfmx
Agent Type:ModularAgent
question_file:../UnifiedTemplate/tria_s_t_1.brd

- This is an example of the current BRD setup. The nine boxes in the middle represent the individual problem parameters (for example, in the above image, 12 represents the base length, and 10 represents the height of the triangle). There may be extraneous information encoded within the BRD.

Current status

Experiment Design

- Experiment modeled on 2019 Mechanical Turk data. Each AL agent completes 16 training questions, and 10 post-test questions. Each question tests geometry skills related to rectangles, triangles, trapezoids, and circles.
- Each question is either a **fact** or a **skill**: facts test knowledge of the area formula for the given shape, while skills involve computing the area for the given shape.

Implementation

- Facts and skills receive different activation boosts (see beta coefficient above), which results in skills having higher initial activation upon every recall, in addition to higher decay.
- AL operators are one-shot. For skills, this means that AL computes the area of a problem in one step, rather than multi-step processes. For facts, this means that the AL operator is simply the corresponding string for the area formula of a shape (e.g. "A=l*w").
- Worked example training questions automatically request a bottom-out hint from the tutor, while retrieval practice questions only a request a bottom-out hint if AL fails to find an appropriate skill to apply.
- Post-tests currently use the *test_mode* parameter, which ensures that feedback is not given.

Future changes

- Adding some sort of prior knowledge
- Add retention interval
- Disallow multiple attempts for retrieval practice questions