Subsymbolic Performance of ACT-R

Karen Rudin & Deniz Duek

April 9, 2015

1 Production Selection (Conflict Resolution)

Once a current goal has been set, ACT-R must choose a production rule to fire in order to achieve that goal.

The question is, if multiple production rules satisfy that goal,
how does the system select which production to fire?

First step: The production rules that match the goal - the conflict set - are ordered given their **expected gain**, or utility.

This is done in parallel.

(1) Expected Gain
E = PG - C

Where:

I. P = probability that goal will be achieved if production is chosen

II. G = goal value (time ACT-R is willing to spend on goal)

III. C = expected cost (time to achieve goal)

Intuitively, production rules are ranked highest the more the probability of achieving the goal outweighs the time it takes to achieve that goal if the production is chosen.

As the value of the goal increases more weight is given to the probability of success, that is, the more weight is placed on achieving the goal, the more important it is to achieve it, so the system will be more willing to spend more time on it. This results in a **speed-accuracy trade-off**.

All production rules in the conflict set are ordered with respect to each other, but given that the only criterion for membership is whether they match the current goal, it is possible that some of them will not be successful because chunk retrieval fails.

That is, the highest ranked production rule may fail, in which case the second highest one will be tried, and so on, for all production that match the goal and have greater than zero expected gain.

This process is done serially.

If no production in the conflict set has a positive expected gain, the goal is popped with failure, leading ACT-R to return to the higher goal.

1.1 Specifying P, G and C

The probability that the goal will be achieved if a given production rule is chosen is a based on both the probability that the production will be successful and that subsequent steps in achieving the goal will also be successful.

(2) P(robability that goal will be achieved if production is chosen)
$P = qr$

(3) q = probability that the production will work successfully
r = probability of achieving goal if production is successful

q will be less than 1 if a retrieval is attempted and fails, or if a subgoal the production sets fails.

The cost, C, is also determined by two parameters.

(4) Cost of Goal
$C = a + b$

(5) a = sum of match time and right-hand-side costs
b = time from production completion to goal being achieved

Right-hand-side costs are set by default as 0.05 seconds, but increases if the production sets subgoals.

q and a refer to both the production and any sugboals it sets. a refers to the amount of time spent in the entire span of production and subgoals, and q the probability of success over the whole span.

G, the goal value, is not derived based on other parameters, but estimated in fitting the data.

Once the initial value is set though, subgoals do not inherit the value of the supergoal. This is intuitive, if G is the amount of time ACT-R is willing to spend on that goal as a whole.

The value of subgoals is set as G' = rG - b: the probability of achieving the goal given the supergoal value, minus the time from production completion to the goal being achieved.

1.2 Introducing some noise

The production selection system given so far is entirely deterministic - there will always be a single highest ranked rule for each goal, so that production will be chosen every time.

ACT-R adds some noise to the evaluation process, so that a production is chosen with a certain probability, reflecting the distance between it and its competitors, relative to that noise.

The noise comes from a logistic distribution, which approximates a normal distribution.

ACT-R determines the probability of selection of production rule by running a series of Monte Carlos simulations, but it can also have the following description

$$\text{Probability of } i = \frac{e^{E_i/t}}{\sum_j e^{E_j/t}} \qquad \text{Conflict Resolution Equation 3.4}$$

Where E_i is the evaluation of alternative i, and t is related to the standard deviation of the noise.

Why add noise? I don't know, see chapter 8.

1.3 Experimental data

Probability-learning paradigm: subjects must make a single choice between two alternatives, learning their probability of success by receiving feedback on what was the correct choice over a number of trials.

Subject's behavior is often characterized as *probability matching*: if an option is right on a proportion p of the trials, subjects will select that option with probability p.

Probability matching is said to be irrational, because if an alternative has probability greater than 0.5, subjects would be able to maximize their correct choices by selecting that alternative all of the time.

Friedman et al (1964): subjects do not exactly probability match.

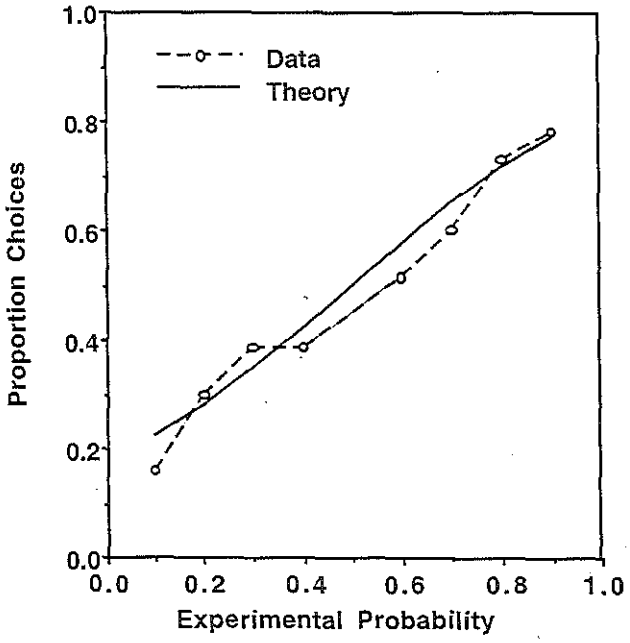


FIG. 3.2 Proportion of choices for experimental alternatives as a function of the probability. From Friedman et al. (1964).

ACT-R simulations match the experimental data quite well if the following model is assumed. There are two production rules, with identical conditions.

Choose-Button 1
IF the goal is to make a choice in the experiment
THEN press button 1
and pop the goal

Choose-Button 2
IF the goal is to make a choice in the experiment
THEN press button 2
and pop the goal

Assume the following values for the parameters:

$q = 1$ (rule always fires if selected)
r = true probability of that button
$a = 0.05$ default action cost
$b = 0$ since goal is popped and there are no more actions

Since q = 1, P = true probability, and since b= 0, C = a = 0.05

The expected gain (PG - C) of choice 1 is then P_1G - 0.05, and for choice 2 (1- P_1)G - 0..05.

Introducing noise to the selection based on the Conflict Resolution Equation, the probability of choosing alternative 1 is e to the expected gain or 1 over t, over e to the expected gain of 1 over t plus e to the expected gain or 2 over t, or

$$\text{Prob}(1) = \frac{1}{1 + e^{(1-2P_1)G/t}}$$

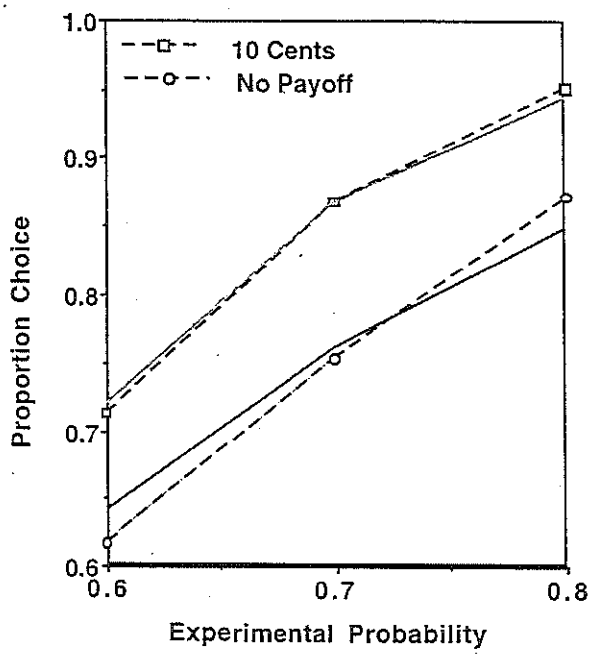
The probability of choosing alternative 1 thus dependents on the ration G/t. t i set to 1, and recall that G is just fitted to the data. Here the best fit for G is 1.54.

Is this jenky? The final probability depends on this value that is not derived from anything else, but just estimated to fit the data.

There is an interesting, an empirically grounded result, however. The higher the value of the goal, the greater the effect of P will be. This means that the higher the value given to the Goal, the further away from probability matching we get, and closer to the expected rational behavior.

This seems to be on the right track. Meyes, Fort, Katz and Suydam (1963): two subject groups, one paid for correct responses, the other not.

If financial incentive increases the value of the goal, then that subject group should behave more rationally. This was indeed the case:



2 Chunk Retrieval (Activation)

Chunk retrieval latency (and success) is a function of that chunks **activation**.

The activation A_i of some chunk i is calculated like this:

(6) $$A_i = B_i + \sum_j W_j S_{ji}$$

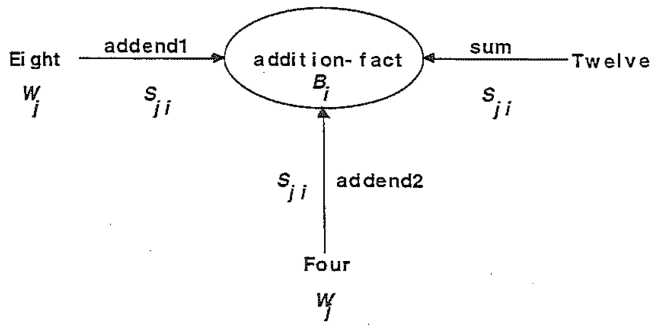
Where:

I. B_i is the base-level activation of the chunk

II. j are slot values of the chunk

III. W_j is the 'attentional weighting' of the slot values

IV. S_{ji} is the strength of association between j and i



B_i is going to be the most interesting to us for the time being; it reflects how frequently and how recently the chunk has been retrieved.

(7) $$B_i(t) = \beta \text{ - } d\ln(t) + \epsilon_1 + \epsilon_2$$

Where:

I. β is the initial expected value of B_i (usually set to 0)

II. d is the decay rate parameter (usually set to 0.5)

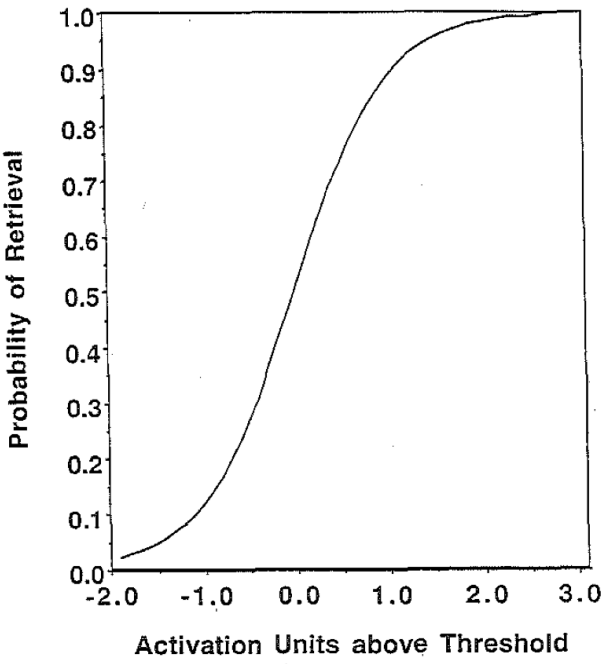
III. ϵ_1 is some random noise permanently associated with i

IV. ϵ_2 is some random noise added with each retrieval attempt

Upshot: activation decays as a logarithmic function of time since last retrieval. So activation spikes very briefly when a chunk is retrieved, declines very quickly as that retrieval recedes into the immediate past, and then decays much more slowly after the loss of the initial spike.

2.1 Retrieval Failure

Chunks will be retrievable if Ai is above the retrieval threshold τ; they will not be retrievable otherwise.



2.2 Partial Matching

Theres not always an exact match for a searched-for chunk; the system shouldnt just throw up its hands, it should search for approximate matches and see if they work. So ACT-R penalizes the activation of chunks that imperfectly match a request, instead of disregarding them entirely.

This is cached out by way of a Match Score M_{ip}:

(8) $M_{ip} = A_i$ - D_{ip}

Where D_{ip} is determined by the number of slots in which chunk i mismatches the goal chunk, and the degree of mismatch.

If the match is perfect, then $D_{ip} = 0$, and the $M_{ip} = A_i$.

Chunk choice works exactly like production rule choice:

(9) probability of retrieving i for $p = \frac{e^{M_{ip}/t}}{\sum_j e^{M_{jp}/t}}$

Takeaway: high-activation perfectly-matching chunks will always be the most likely to be chosen; as distribution of noise increases, the chances of low-activation or partially-matching chunks to be retrieved also increases.

Theres something interesting here that they dont discuss: its possible for a partially-matching chunk to be the objective (i.e. pre-noise) winner here if its a very high-activation chunk and perfectly-matching chunks are very low-activation.

There's a concrete example about addition mistakes by 4-year-olds on pp. 77-80 that could be discussed if there's interest.

2.3 Retrieval Latency

Activation doesn't just influence the chance of a chunk being retrieved; it also makes predictions about retrieval latency.

Retrieval time, relative to some production rule p, is the sum of Time_{ip} for all chunks i that have to be retrieved for production p (usually, thats just one chunk).

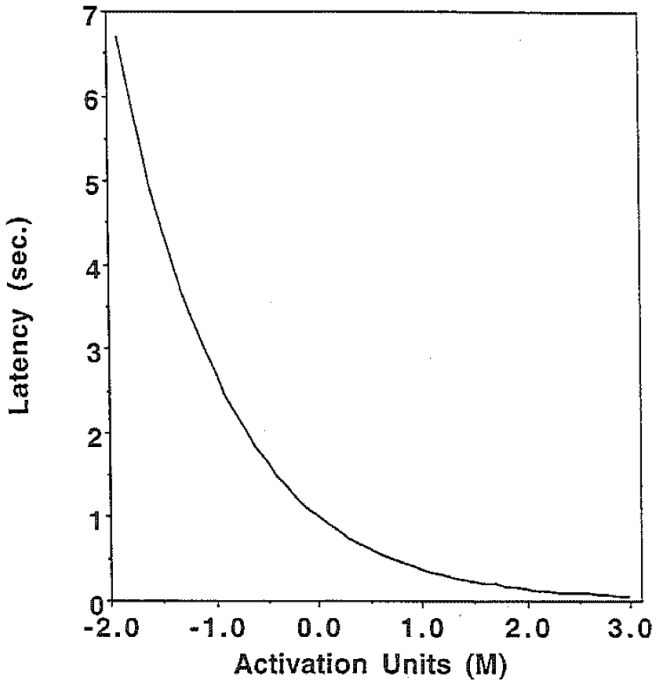
(10) $$\text{Time}_{ip} = Fe^{-f(\text{M}_{ip} + \text{S}_p)}$$

We usually assume that f is 1 and S_p is 0, so really its just:

(11) $$\text{Time}_{ip} = Fe^{-\text{M}_{ip}}$$

F is a latency scale parameter representing the time it takes to retrieve a chunk i when M_{ip} is zero. They assume that F = one second.

Whats crucial to understand about this is that the retrieval latency function is exponential, so retrieval time increases very quickly as activation (M_{ip}) becomes negative, but increases very slowly as activation becomes positive:



Again, there's a concrete example (pp.82-87) to walk through if the desire exists.

2.4 Retrieval Failure

Finally: if no chunks have a high enough match score M_{ip} to surpass the retrieval threshold τ, the production rule that initiated the search is jettisoned.