Flashcard using combined Leitner-assisted HLR Algorithm

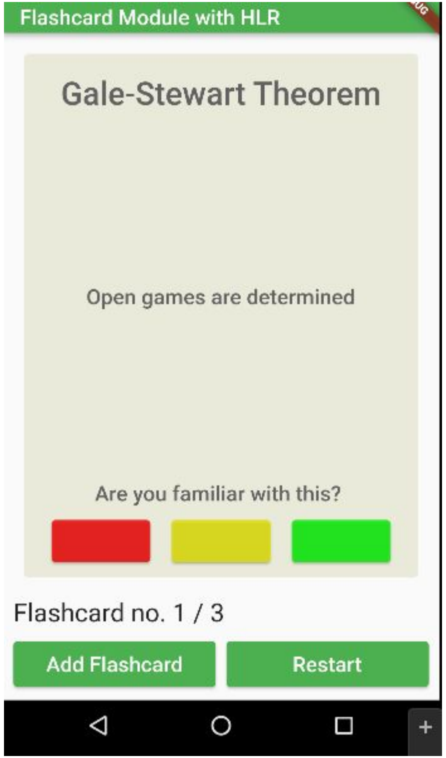
Owen

Problem

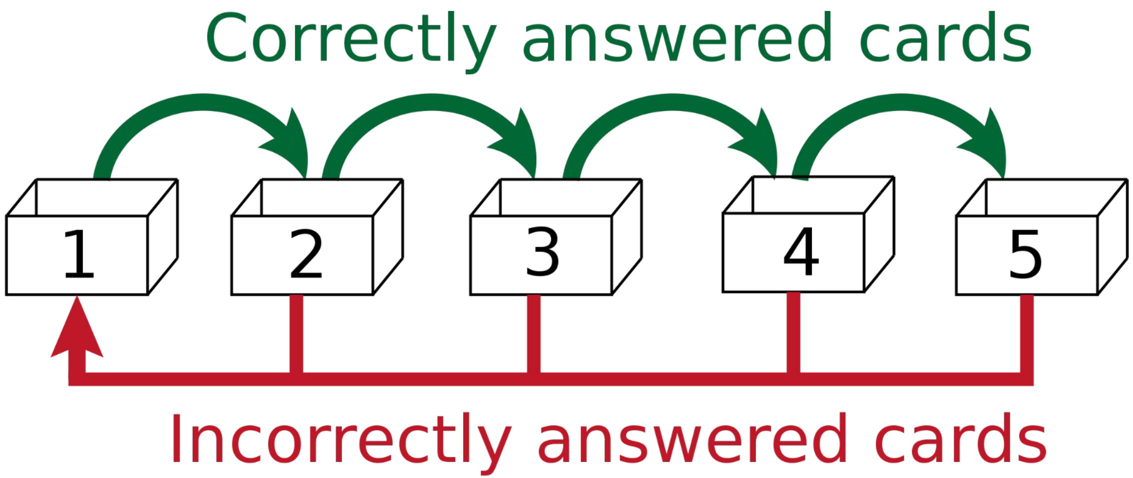
- Not many applications combine an efficient and modular learning-pattern optimization algorithm with a customizable word-list.

Solution

1. Needs an App
    a. Flutter framework
2. Needs algorithm
    a. Leitner
    b. HLR

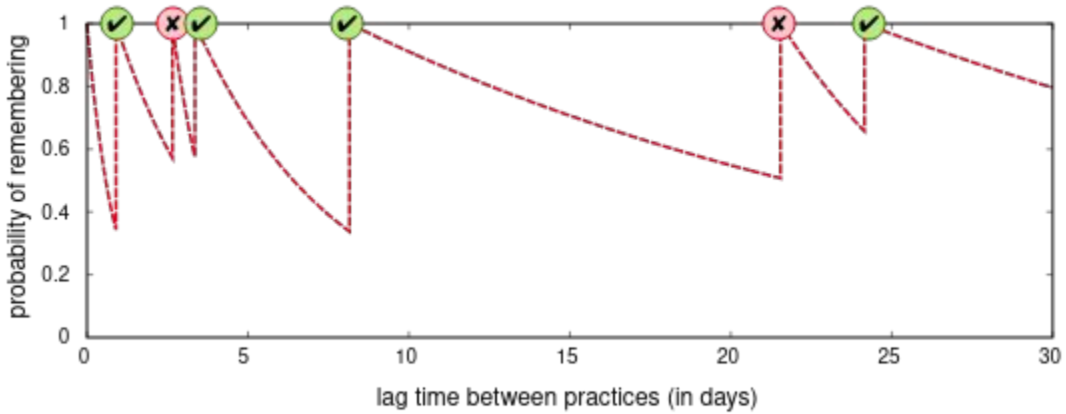

Flashcard Module with HLR
Gale-Stewart Theorem
Open games are determined
Are you familiar with this?
Flashcard no. 1 / 3
Add Flashcard
Restart

Leitner



Problem: not quite adaptive.

Half Life Regression (HLR) – Duolingo



Memory is assumed to exponentially decays with respect to time — we just need to figure out the rate of decay, and initiate reviews when the retention probability is close to 50%.

Duolingo's Formula

$$\begin{aligned}\ell(\langle p, \Delta, \mathbf{x}\rangle; \Theta) &= (p - \hat{p}_\Theta)^2 \\ &\quad + \alpha(h - \hat{h}_\Theta)^2 + \lambda\|\Theta\|_2^2 \, .\end{aligned}$$

Problem: half-life hard to obtain without accurate-enough probability of memory retention

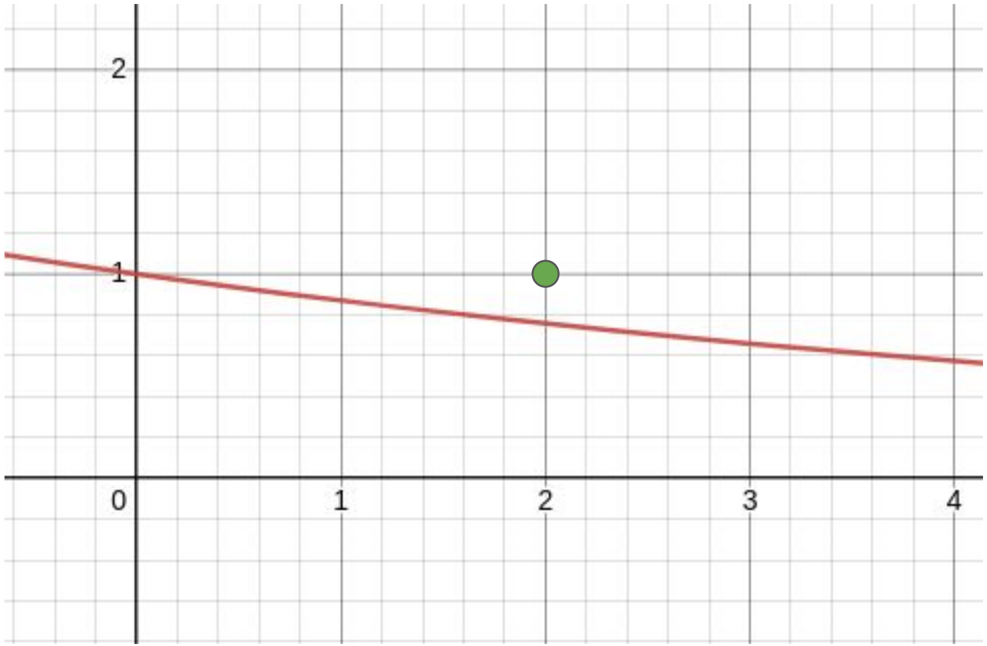
$$\ell(\langle p, \Delta, \mathbf{x}\rangle; \Theta) = (p - \hat{p}_\Theta)^2 + \boxed{\alpha(h - \hat{h}_\Theta)^2} + \lambda\|\Theta\|_2^2 \, .$$

First attempt: HLR

$$p = \begin{cases} 1.0, \text{recalled} \\ 0.5, \text{neutral} \\ 0.0, \text{forgot} \end{cases}$$

$$\begin{aligned}
\ell &= \sum_{i=1}^{n} (\hat{p}_i - p_i)^2 \\
&= \sum_{i=1}^{n} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right)^2 \\
\frac{d}{d\hat{t}}\ell &= \sum_{i=1}^{n} 2 \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \frac{d}{dt} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \\
&= \sum_{i=1}^{n} 2 \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \left((\frac{1}{2})^{t_i/\hat{t}} \ln(\frac{1}{2})(-\frac{t_i}{\hat{t}}) \right)
\end{aligned}$$

Problem: exploding gradient



In an attempt to approximate the forgetting curve when there’s a correct answer at t=2, the algorithm will try to make half-life infinite.

Solution 1: L2-Regularization

$$p = \begin{cases} 1.0, \text{recalled} \\ 0.5, \text{neutral} \\ 0.0, \text{forgot} \end{cases}$$

$$\begin{aligned}
\ell &= \hat{t}^2 + \sum_{i=1}^{n} (\hat{p}_i - p_i)^2 \\
&= \hat{t}^2 + \sum_{i=1}^{n} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right)^2 \\
\frac{d}{d\hat{t}}\ell &= 2\hat{t} + \sum_{i=1}^{n} 2 \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \frac{d}{dt} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \\
&= 2\hat{t} + \sum_{i=1}^{n} 2 \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \left((\frac{1}{2})^{t_i/\hat{t}} \ln(\frac{1}{2})(-\frac{t_i}{\hat{t}}) \right)
\end{aligned}$$

Does not work well in practice!

Leitners Estimation on Success

$$h' = 2h$$

$$(\frac{1}{2})^h = (\frac{1}{2})^{h'/2} = \sqrt{(\frac{1}{2})^{h'}}$$

$$p = \sqrt{p'}$$

when $p = 1/2$, we have $p' \approx 0.7$.

$$p = \begin{cases} {\color{red}0.7}, \text{recalled} \\ 0.5, \text{neutral} \\ 0.0, \text{forgot} \end{cases}$$

$$p = \begin{cases} 0.7, \text{recalled} \\ 0.5, \text{neutral} \\ 0.0, \text{forgot} \end{cases}$$

$$\begin{aligned}
\ell &= \sum_{i=1}^{n} (\hat{p}_i - p_i)^2 \\
&= \sum_{i=1}^{n} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right)^2 \\
\frac{d}{d\hat{t}}\ell &= \sum_{i=1}^{n} 2 \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \frac{d}{dt} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \\
&= \sum_{i=1}^{n} 2 \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \left((\frac{1}{2})^{t_i/\hat{t}} \ln(\frac{1}{2})(-\frac{t_i}{\hat{t}}) \right)
\end{aligned}$$

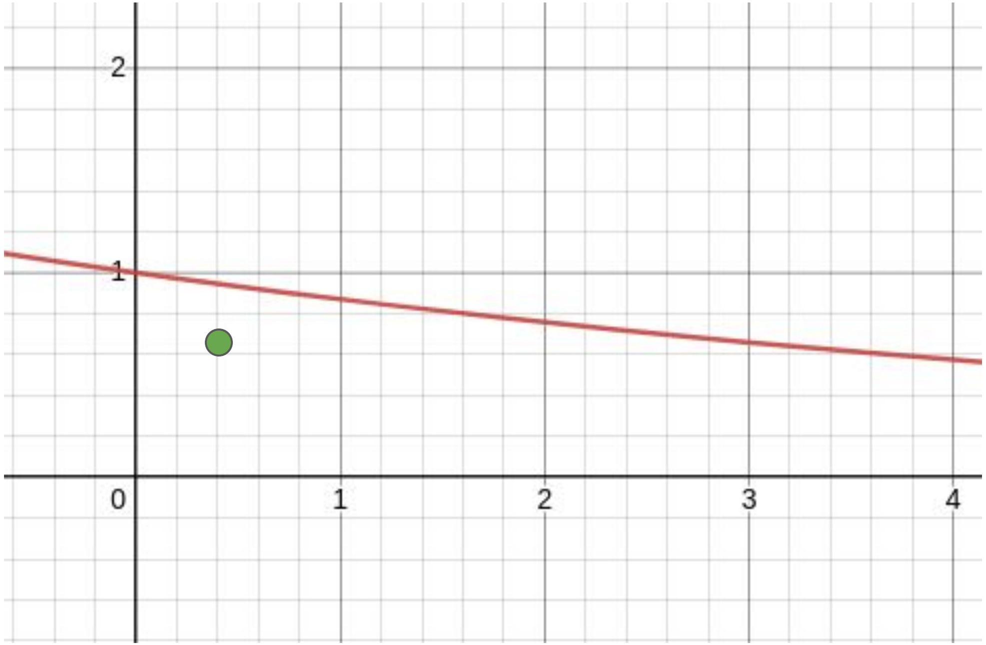
Problem: too dependent on history

Rolling Window

$$p = \begin{cases} 0.7, \text{recalled} \\ 0.5, \text{neutral} \\ 0.0, \text{forgot} \end{cases}$$

$$\begin{aligned}
\ell &= \hat{t}^2 + \sum_{i=n-k+1}^{n} (\hat{p}_i - p_i)^2 \\
&= \hat{t}^2 + \sum_{i=n-k+1}^{n} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right)^2 \\
\frac{d}{d\hat{t}}\ell &= 2\hat{t} + \sum_{i=n-k+1}^{n} 2\left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \frac{d}{dt} \left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \\
&= 2\hat{t} + \sum_{i=n-k+1}^{n} 2\left((\frac{1}{2})^{t_i/\hat{t}} - p_i \right) \cdot \left((\frac{1}{2})^{t_i/\hat{t}} \ln(\frac{1}{2})(-\frac{t_i}{\hat{t}}) \right)
\end{aligned}$$

Preventing negative feedback from clustered review



One more problem: If user made a correct recall shortly after a previous study session, the algorithm would try to reduce the half-life to fit such datapoint, which is not realistic as clustered learning sessions aren't known to actively make people forget what they learned.

Solution: ditch those cases!

```
// prevents unwanted negative feedback
// due to clustered study sessions
if (time[i] < hl && rate[i] == 2) continue;
```

And that is how I came up with such a version of HLR. Thanks for listening!