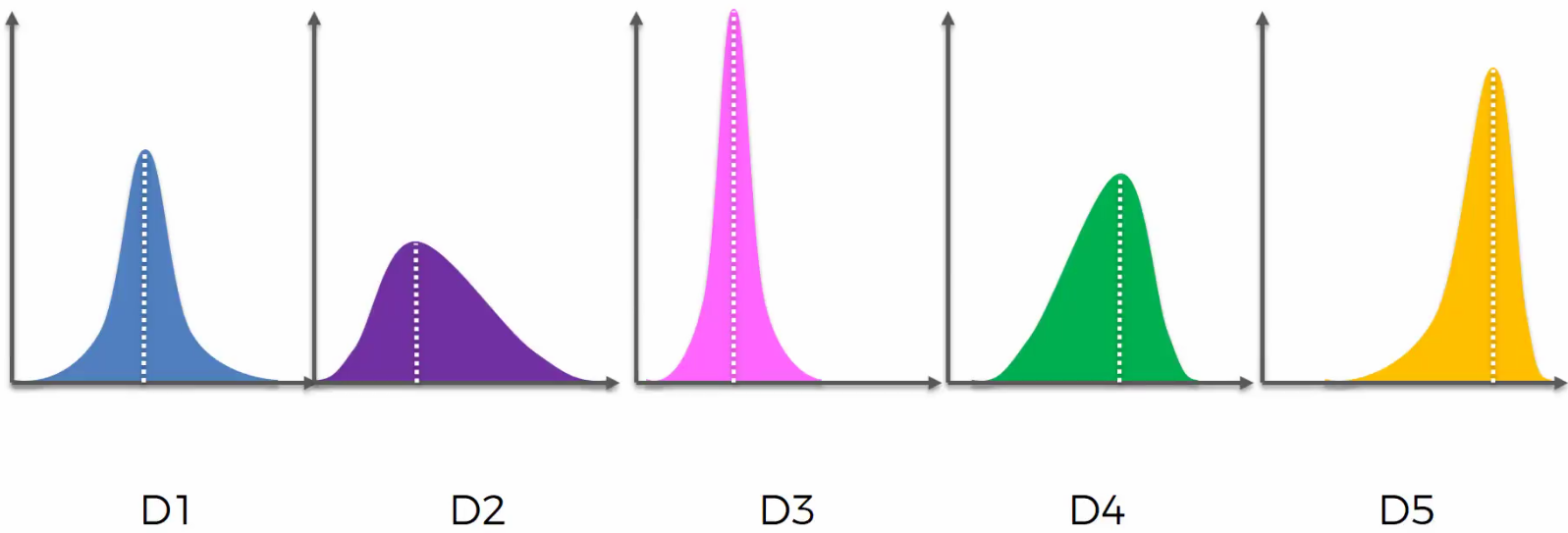
* **Reinforcement Learning** = branch of ML, also called **Online Learning**, used to solve interacting problems where data observed up to time **t** is considered to decide which action to take at time t + 1
* It is also used for AI when training machines to perform tasks such as walking.
* Desired outcomes provide the AI w/ reward, undesired outcomes w/ punishment, as machines learn through trial +error.

**Multi-Armed Bandit Problem**

* 1-armed bandit = a slot machine, the cause of 1 of the quickest ways to lose $ in casinos
* **Multi-armed bandit** = challenge one faces when they come up to multiple slot machines + want to find out how to play them to maximize return from the # of games one is going to play
* Assume each machine has a distribution of #’s/outcomes behind it out of which the machine picks results when pulled (tells you if you win/lose based on distribution w/in the machine)
* Problem = we don’t know the distributions + they’re different in each machine
* Goal = figure out which machine’s distribution is best for us



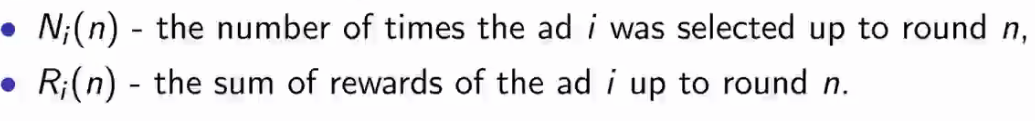
* D5 is the best distribution with the most left-skew (most positive outcome values = most “winning” outcome values)
* But, we’re spending $ trying to figure this out 🡺 the longer we take to figure it out, the more confident we are but the less $ we have (greater cost)
* Want to do so as fast as possible
* 2 ways = **exploration** vs. **exploitation**
* Explore machines to find best one + at the same time exploit findings we have to get max return
* **Regret** = suffered when we use a non-optimal method (using any machine that’s not D5) + can be quantified as the difference between the best outcome + the non-best outcome as well as opportunity costs of exploring other machines
* Longer time exploring non-optimal machines = more regret
* But if we don’t spend *enough* time exploring, we might choose a sub-optimal machine as our optimal machine
* Want to minimize amount of time exploring all machines as we’re still earning money, but not optimizing results
* Most common modern application = advertising
* Coke wants to run a campaign + find out which ad maximizes ROI



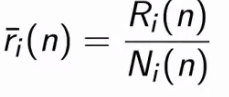
* Distribution only become known when thousands of people click on these ads
* To approach this, we could do an A/B test + wait until we have large enough sample to choose which ad is best, but the problem here = costs a lot of time + $ as an A/B test is pure exploration of ALL ads, even the non-optimal ones
* Want to exploit the best one while exploring all the ads (find out best ad while doing the campaign)

**Upper Confidence Bound (UCB)**

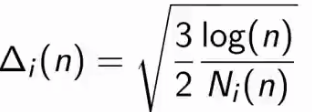
* Multi-armed bandit problem = **d** arms/ads 🡺 each time users connects to web page, that makes a **round, n** 🡺 in each round n, choose 1 ad, **i**, to display to user
* ad i gives **reward =**  if user clicks on ad i + reward = 0 if not
* Want to maximize total reward over many rounds
* 1) At each round n, consider 2 #’s for each add i:



* 2) From these 2 #’s, compute:
* Average reward of ad i up to round n:



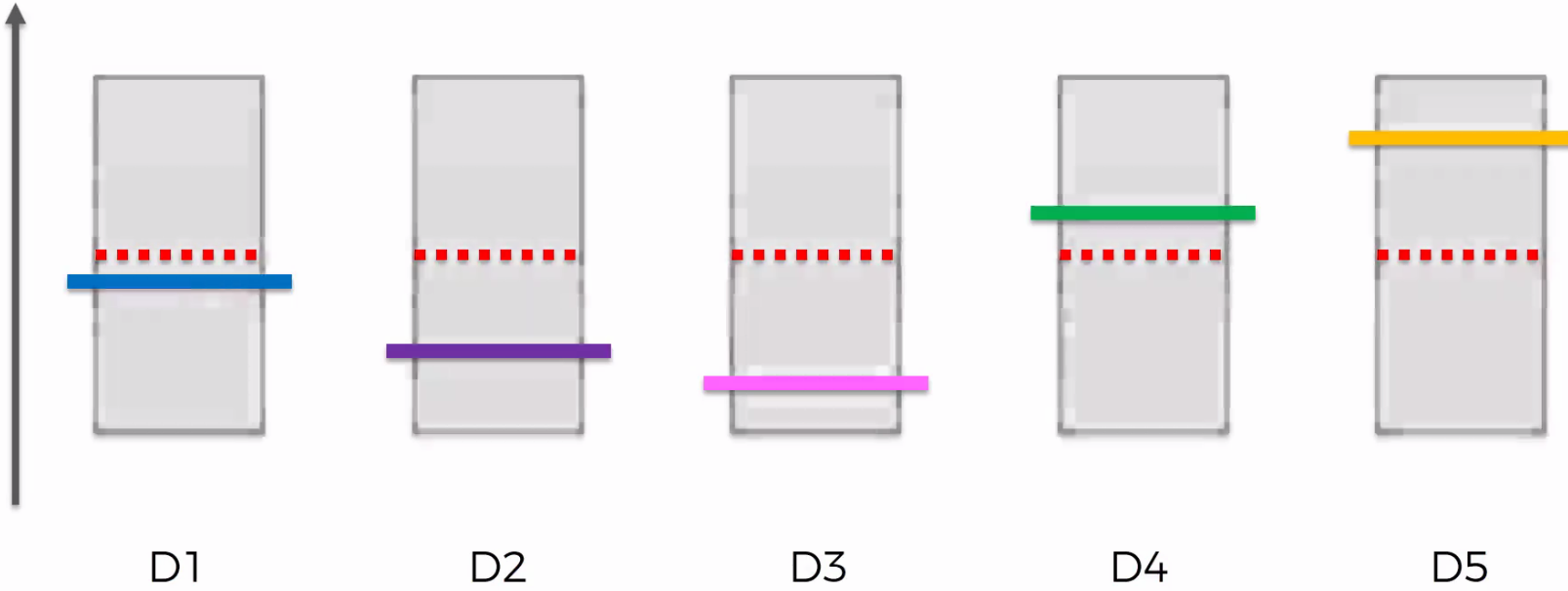
* Confidence interval =  at round n:



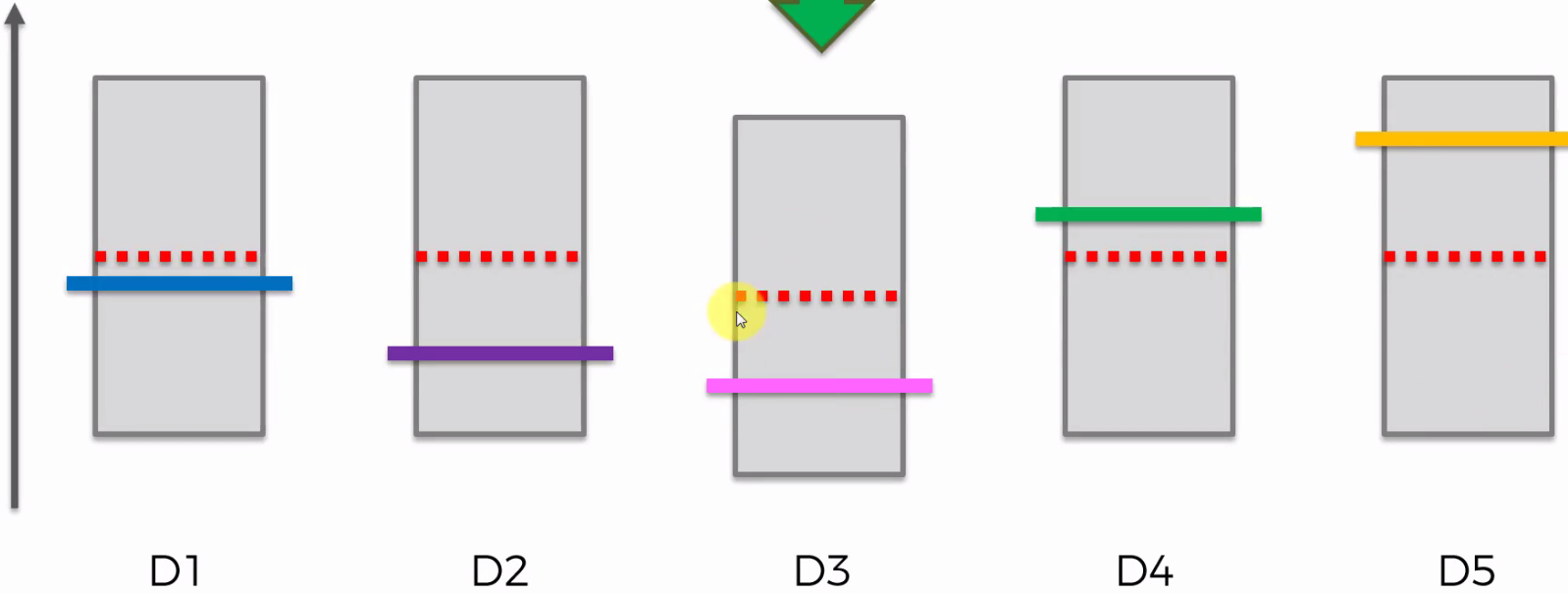
* 3) select ad i w/ highest/maximum 
* Look at the expected return values for the distributions above



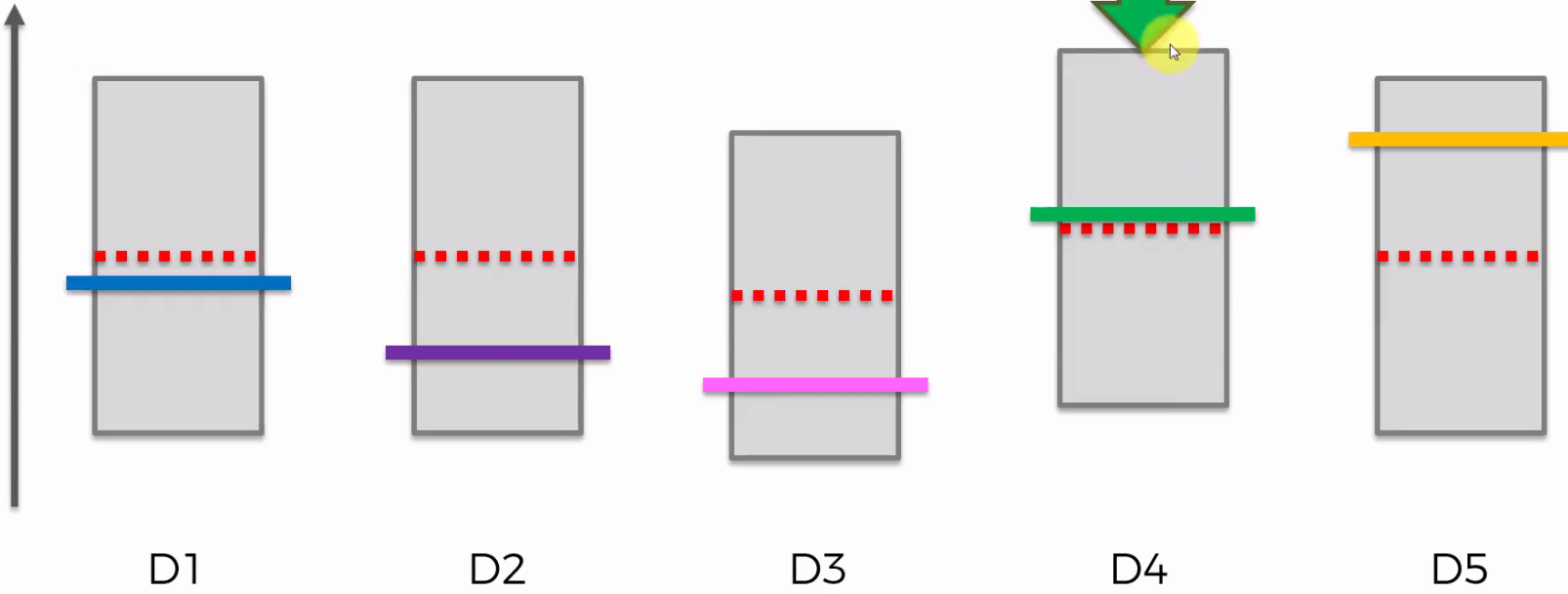
* We do not know this info.
* The algorithm assumes some starting point for each distribution, and since we can’t discriminate against these machine yet, it’s the same of each distribution (red dotted line)
* Then, the confidence band is calculated w/ a very high confidence that it contains the true expected return



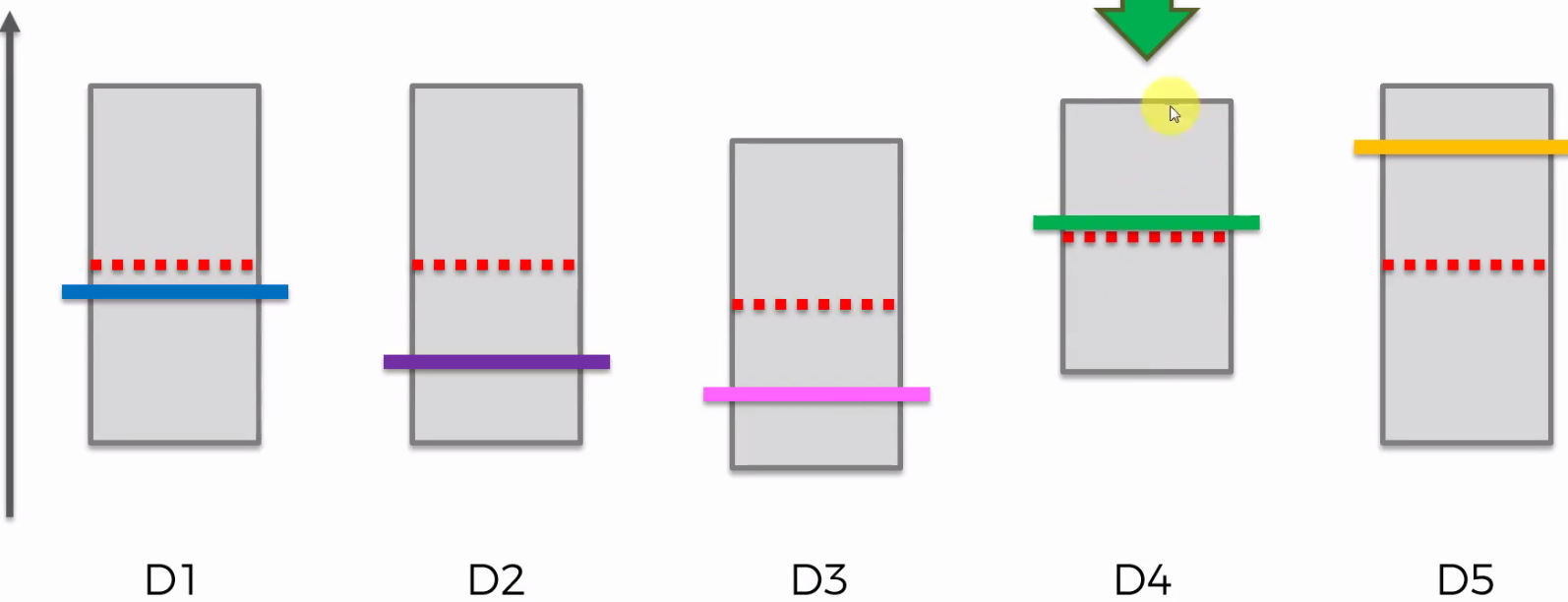
* The first couple of rounds are trials (each machine once for example)
* The algorithm then picks the machine w/ the highest confidence bound (upper), but here they’re all the same so we pick any
* Say we pick M3 🡺 pull its level/place this ad 🡺 if user doesn’t click, confidence band goes down + *becomes shorter*



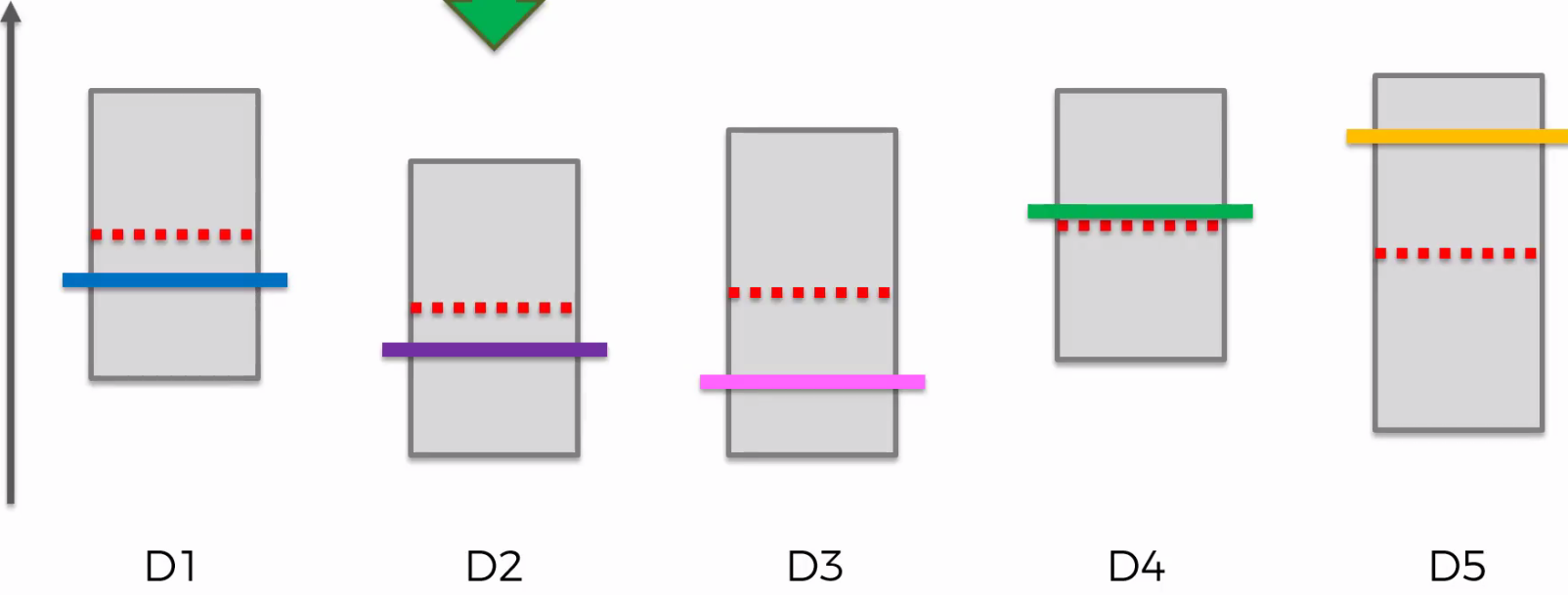
* The observed average return (red line) goes down (law of large #’s = in long run, observed average converges to expected return (pink)) + we have a new DP in this distribution so we’re more confident in it + the band shrinks
* Now we can pick 1 of the 4 higher confidence bands to work on next, say M4, and the user clicks



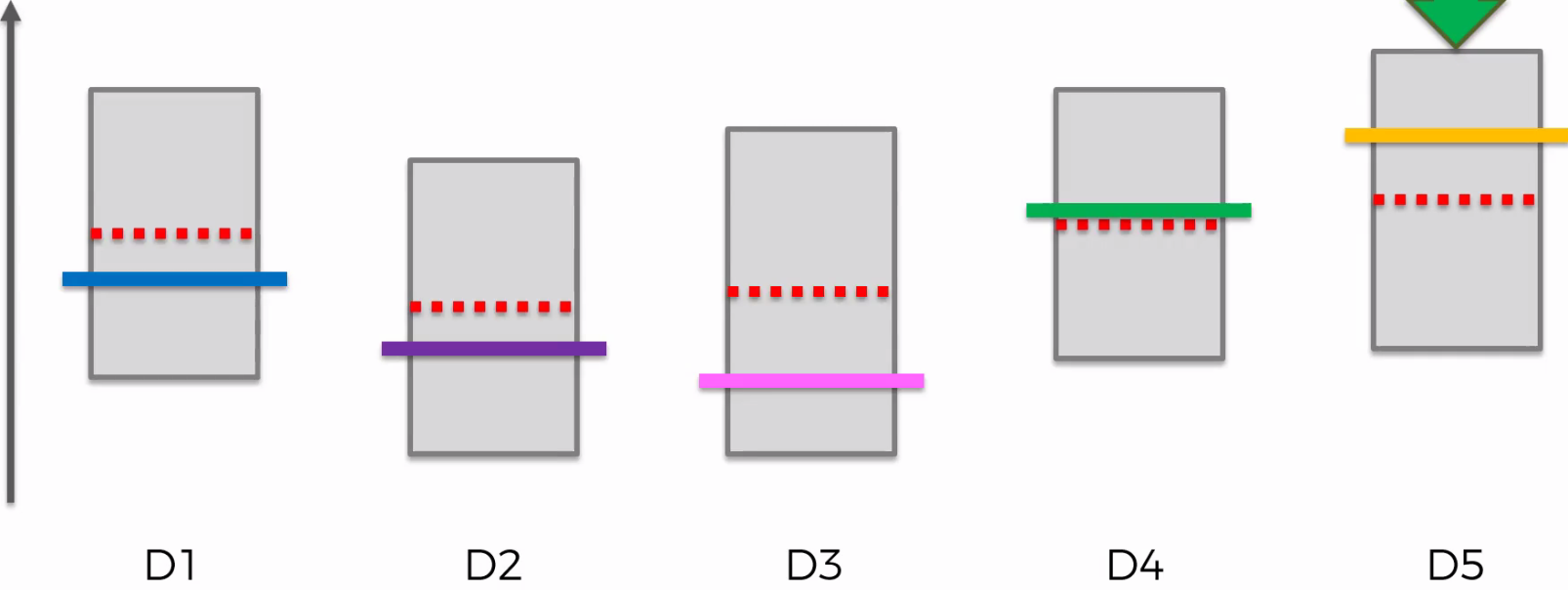
* Now M4 is the highest band, and if we ended the algorithm here, this would be our optimal machine
* But remember the band shrinks b/c we’re more confident due to the extra DP



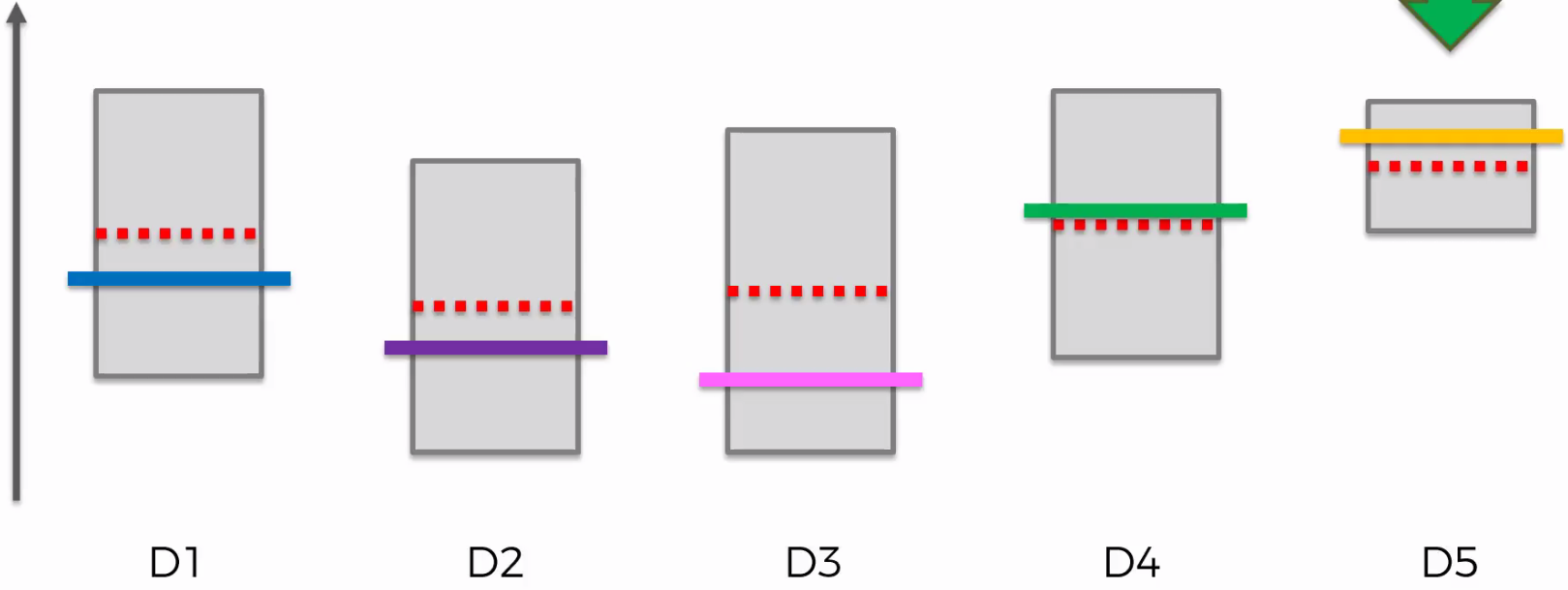
* Next user clicks on ad 1 + next user does not click on ad 2



* Now we’d say M5 is the “best” machine, but the user does not know that
* The algorithm is starting to unconsciously exploit it, as user does click on ad 5:



* It’s still the best, so we display it again, and repeat this:



* By exploiting this best option, we’re decreasing the width of the band/increasing our confidence that the interval contains the true return value

