Introduction to Computational Neuroscience 2017

Lecturers
Boris Gutkin (boris.gutkin@ens.fr)
Matty Chalk(matthewjchalk@gmail.com)
Gregory Dumont (gregory.dumont@ens.fr)

Tuesdays 17-19, Salle Prestige
HW: Help sessions TBA
http://iec-lnc.ens.fr/group-for-neural-theory/teaching-260/article/co6-course

Textbook: **Dayan & Abbott. Theoretical Neuroscience.** MIT Press 2001.

Introduction to Computational Neuroscience 2017

Lecturers
Boris Gutkin (boris.gutkin@ens.fr)
Matty Chalk(matthewjchalk@gmail.com)
Gregory Dumont (gregory.dumont@ens.fr)

Tuesdays 17-19, Salle Prestige
HW: Help sessions TBA
http://iec-lnc.ens.fr/group-for-neural-theory/teaching-260/article/co6-course

Textbook: **Dayan & Abbott. Theoretical Neuroscience.** MIT Press 2001.

Behavior
1.1 The Rescorla-Wagner Rule (Gutkin)
1.2 Exploration-Expoitation/ Reinforcement Learning: Model-free approaches (Gutkin)
1.3 Reinforcement Learning: Model-based approaches (Gutkin)
1.4 Seminar in RL

Neural Biophysics
3.1 Binary Neurons and networks (Dumont)
3.2 Firing rate models(Dumont)
3.3 Biophysics of Neurons(Dumont)
3.4 Hodgkin-Huxley(Dumont)

Neural Computation
2.1 Neural Decoding (Chalk)
2.2 Population Coding (Chalk)
2.3 Decision-Making (Chalk)
2.4 Neural Encoding (Chalk)
2.5 Seminar (Deneve)

Introduction to Computational Neuroscience 2017

Lecturers
Boris Gutkin (boris.gutkin@ens.fr)
Matty Chalk(matthewjchalk@gmail.com)
Gregory Dumont (gregory.dumont@ens.fr)

Tuesdays 17-19, Salle Prestige
HW: Help sessions TBA
http://iec-lnc.ens.fr/group-for-neural-theory/teaching-260/article/co6-course

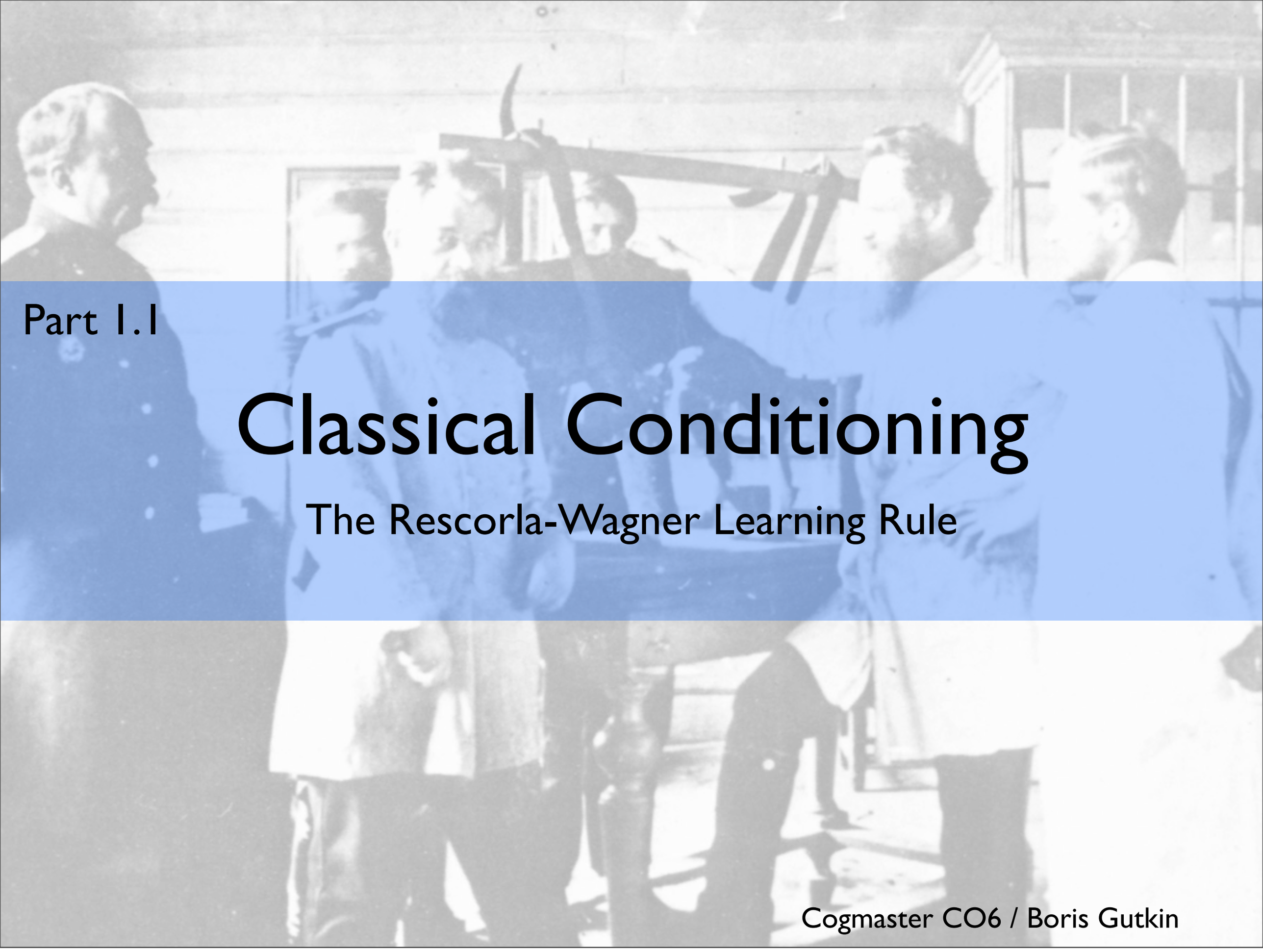
Textbook: **Dayan & Abbott. Theoretical Neuroscience.** MIT Press 2001.

Behavior
1.1 The Rescorla-Wagner Rule (Gutkin)
1.2 Exploration-Expoitation/ Reinforcement Learning: Model-free approaches (Gutkin)
1.3 Reinforcement Learning: Model-based approaches (Gutkin)
1.4 Seminar in RL

Neural Biophysics
3.1 Binary Neurons and networks (Dumont)
3.2 Firing rate models(Dumont)
3.3 Biophysics of Neurons(Dumont)
3.4 Hodgkin-Huxley(Dumont)

Neural Computation
2.1 Neural Decoding (Chalk)
2.2 Population Coding (Chalk)
2.3 Decision-Making (Chalk)
2.4 Neural Encoding (Chalk)
2.5 Seminar (Deneve)

Evaluation:
Homeworks 80% : taken up next week, returned 2 weeks later. No late HW!!!!!!!
Written Exams: 20% June 6



Part 1.1

Classical Conditioning

The Rescorla-Wagner Learning Rule

Cogmaster CO6 / Boris Gutkin

Classical conditioning

Ivan Pavlov (1849-1936)

- Work on digestive tract of dogs
- Nobel prize in 1903 for work on digestive glands
- Most famous for "classical conditioning"

Classical conditioning

testing	1.	food	→	salivation
testing	2.	sound	→	no salivation
training	3.	sound	→ food	→	salivation
testing	4.	sound	→	salivation

Extinction

training	1.	sound	food	→	salivation
testing	2.	sound		→	salivation
training	3.	sound	→ no food	→	salivation disappears
testing	4.	sound		→	no salivation

Interpretation

Why is this happening?

- involuntary behavior
- learning of associations
- temporal proximity of stimuli

Ivan Pavlov
(1849-1936)
work: 1890s

John B Watson
(1878-1958)
work: 1920s

Interpretation

Why is this happening?

- involuntary behavior
- learning of associations
- temporal proximity of stimuli

Ivan Pavlov
(1849-1936)
work: 1890s

John B Watson
(1878-1958)
work: 1920s

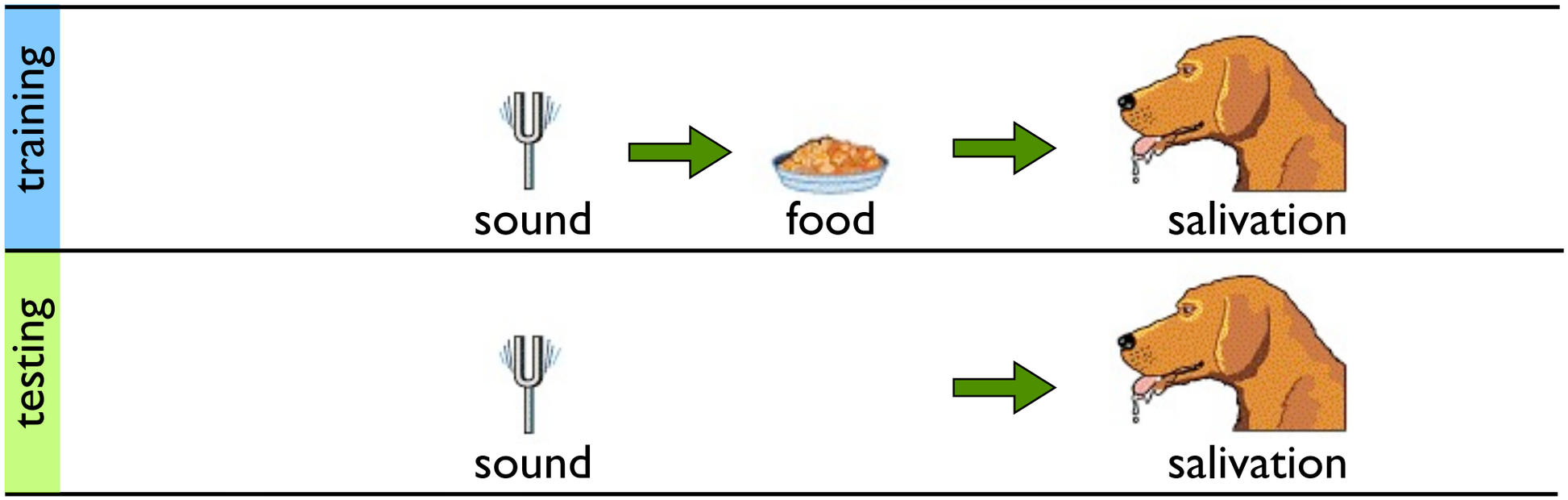
Why is this happening?

- learning to predict

Robert Rescorla

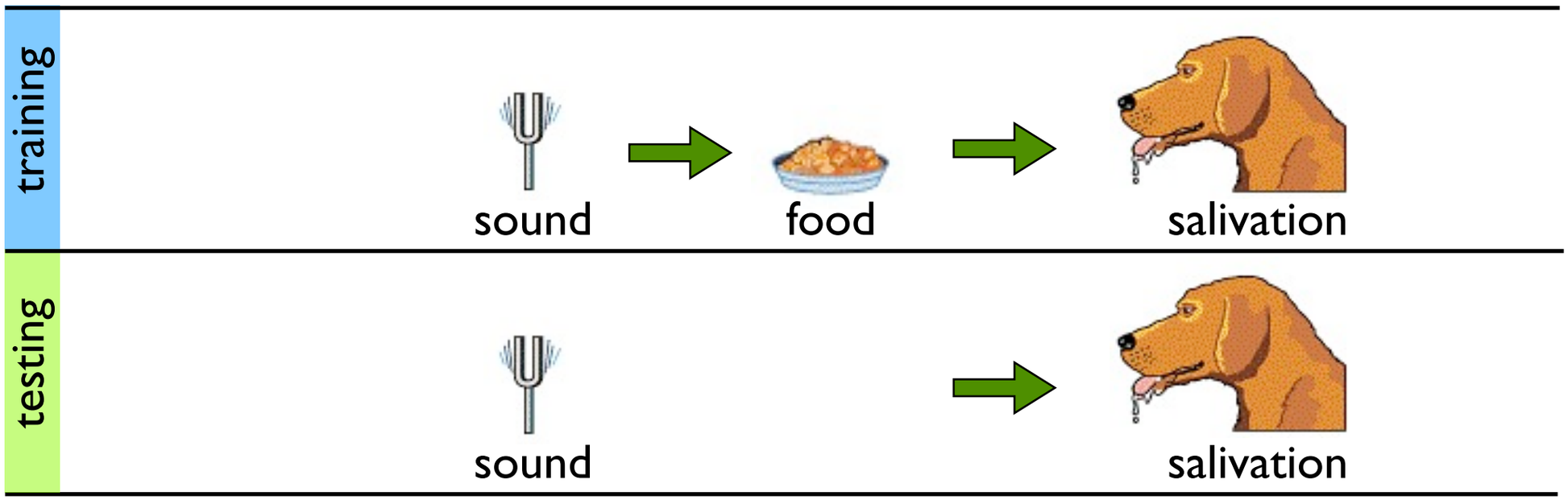
Allan R Wagner

The Rescorla-Wagner model



Assume: The dog wants to be able to predict the reward!

The Rescorla-Wagner model

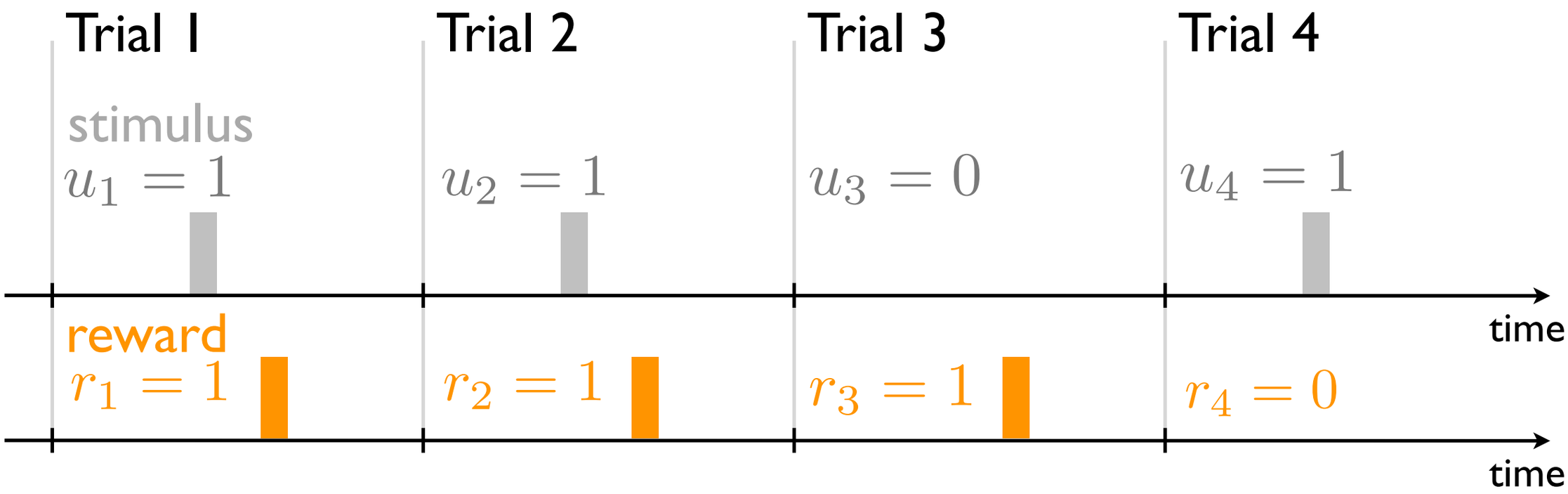


Assume: The dog wants to be able to predict the reward!

u_i stimulus ( ) in trial i: $u_i = 0$ or $u_i = 1$

r_i reward ( ) in trial i: $r_i = 0$ or $r_i = 1$

The Rescorla-Wagner model

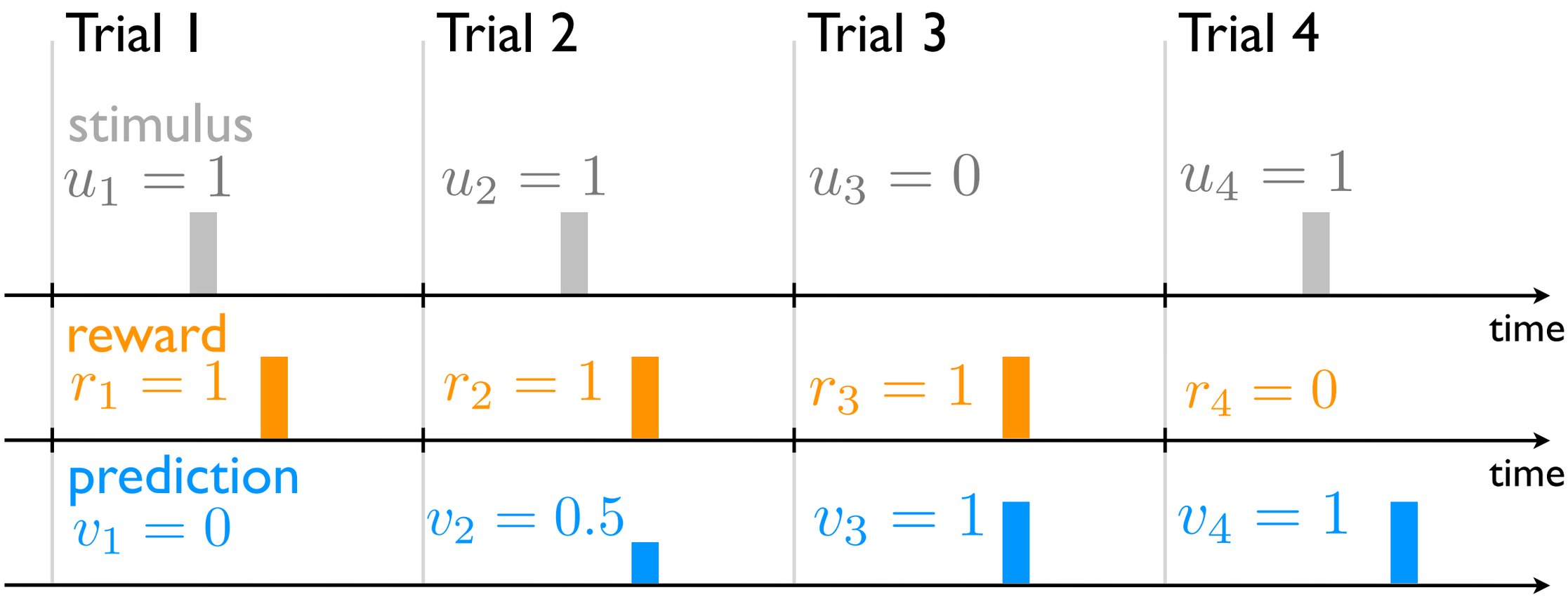


Assume: The dog wants to be able to predict the reward!

u_i stimulus ( ) in trial i: $u_i = 0$ or $u_i = 1$

r_i reward ( ) in trial i: $r_i = 0$ or $r_i = 1$

The Rescorla-Wagner model



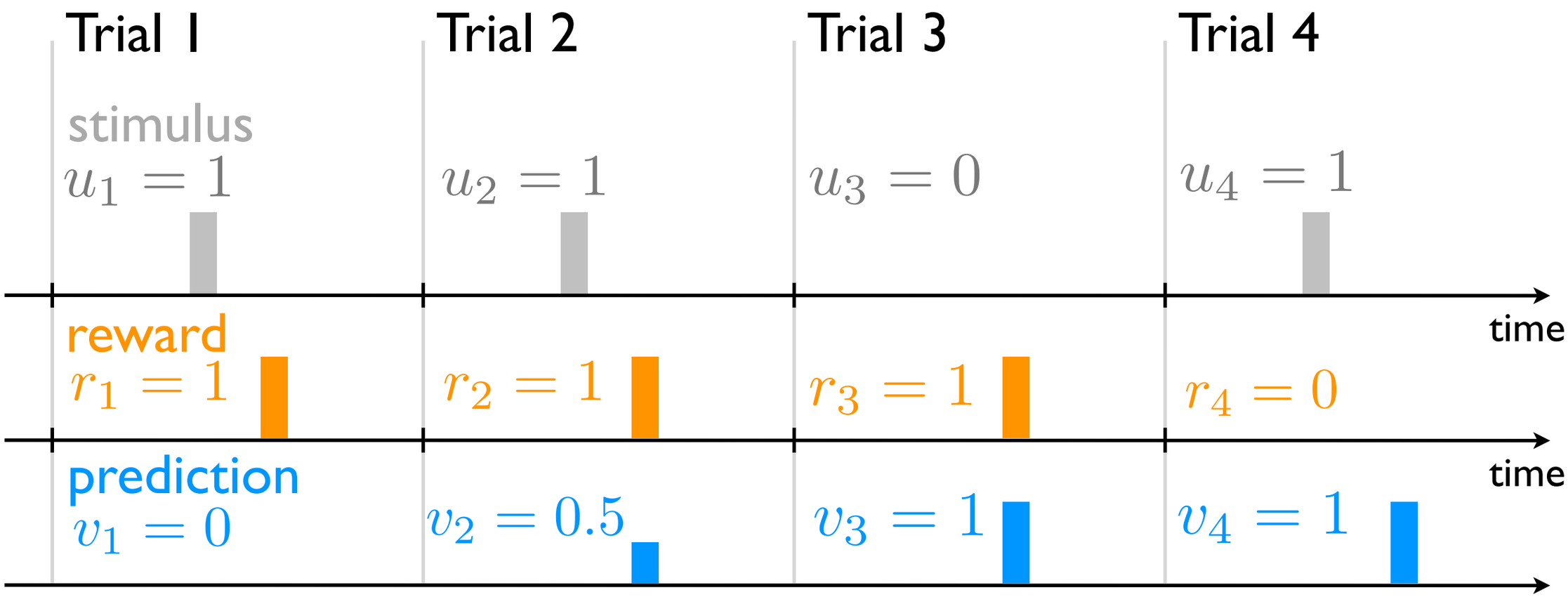
Assume: The dog wants to be able to predict the reward!

u_i stimulus ( ) in trial i: $u_i = 0$ or $u_i = 1$

r_i reward ( ) in trial i: $r_i = 0$ or $r_i = 1$

v_i reward that the dog expects ( ) in trial i

The Rescorla-Wagner model



Assume: The dog wants to be able to predict the reward!

u_i stimulus ( ) in trial i: $u_i = 0$ or $u_i = 1$

r_i reward ( ) in trial i: $r_i = 0$ or $r_i = 1$

v_i reward that the dog expects ( ) in trial i

The Rescorla-Wagner model

Measure the ability of the dog to predict the reward!
Prediction error in the i-th trial:

$$\delta_i = r_i - v_i$$

actual reward

predicted reward

The Rescorla-Wagner model

Measure the ability of the dog to predict the reward!
Prediction error in the i-th trial:

$$\delta_i = r_i - v_i$$

actual reward

predicted reward

$\delta_i > 0$: more reward than predicted
$\delta_i < 0$: less reward than predicted

The Rescorla-Wagner model

Measure the ability of the dog to predict the reward!
Prediction error in the i-th trial:

$$\delta_i = r_i - v_i$$

actual reward — predicted reward

$\delta_i > 0$: more reward than predicted
$\delta_i < 0$: less reward than predicted

"Loss" in the i-th trial:

$$L_i = \delta_i^2 = (r_i - v_i)^2$$

→ Minimize this loss function to maximize the ability to predict the reward !!

The Rescorla-Wagner model

Minimize the loss:

$$L_i = (r_i - v_i)^2$$

Assume: dog's model of the world

$$v_i = w u_i$$

The Rescorla-Wagner model

Minimize the loss:

$$L_i = (r_i - v_i)^2$$

Assume: dog's model of the world

$$v_i = wu_i$$

parameter that the dog needs to learn from observations

The Rescorla-Wagner model

Minimize the loss:

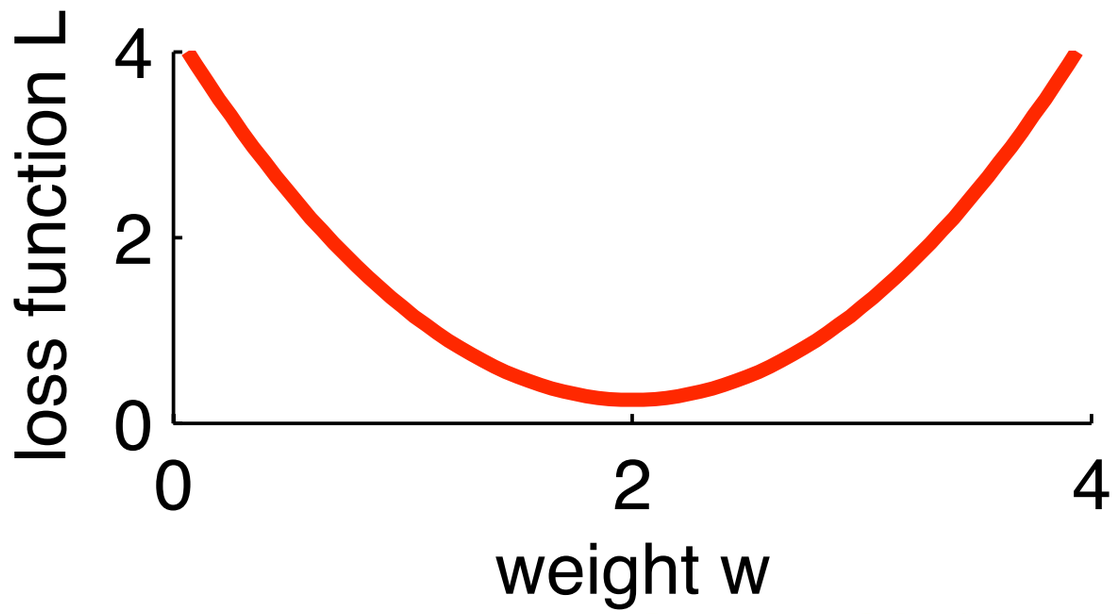
$$L_i = (r_i - v_i)^2$$

Assume: dog's model of the world

$$v_i = w u_i$$

Then:

$$L_i = (r_i - w u_i)^2$$

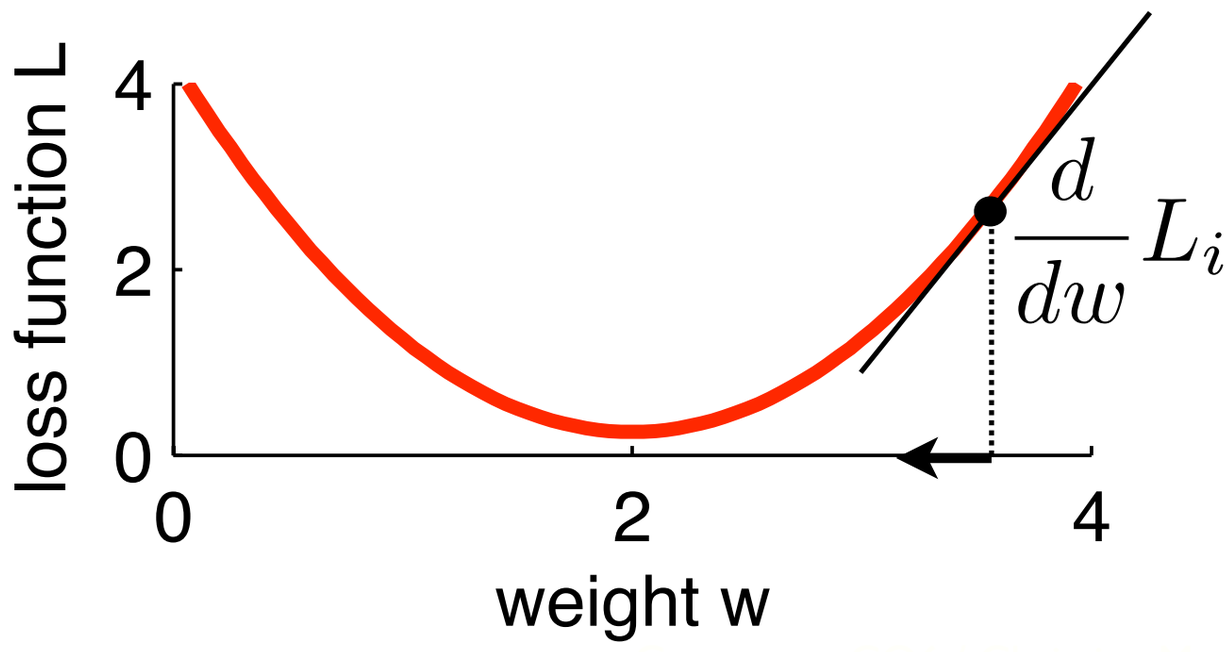


The Rescorla-Wagner model

Minimize the loss:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!



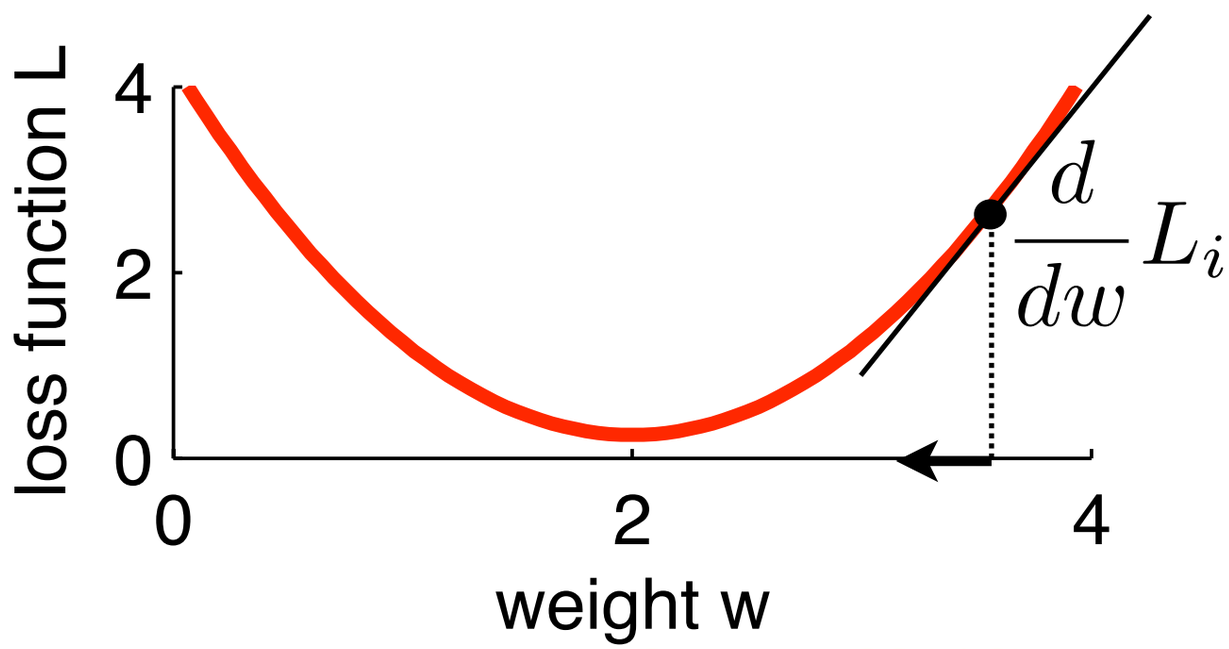
The Rescorla-Wagner model

Minimize the loss:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!

$$w \rightarrow w - \epsilon \frac{d}{dw} L_i$$



The Rescorla-Wagner model

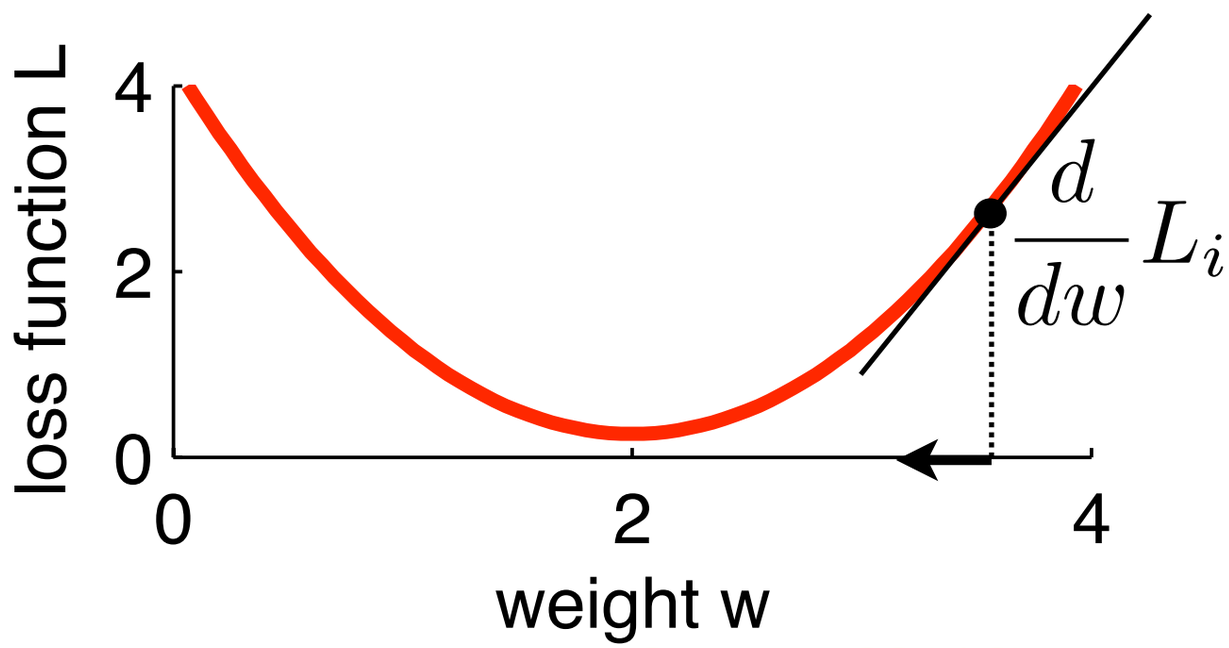
Minimize the loss:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!

$$w \rightarrow w - \epsilon \frac{d}{dw} L_i$$

learning rate



The Rescorla-Wagner model

Minimize the loss:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!

$$w \rightarrow w - \epsilon \frac{d}{dw} L_i$$

Compute the derivative:

$$\frac{d}{dw} L_i = \frac{d}{dw} (r_i - wu_i)^2$$

The Rescorla-Wagner model

"Loss" in the i-th trial:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!

$$w \rightarrow w - \epsilon \frac{d}{dw} L_i$$

Compute the derivative:

$$\frac{d}{dw} L_i = \frac{d}{dw} (r_i - wu_i)^2$$

$$= \underbrace{(-u_i)}_{\text{inner derivative}} \underbrace{2(r_i - wu_i)}_{\text{outer derivative}}$$

"Chain rule"

The Rescorla-Wagner model

"Loss" in the i-th trial:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!

$$w \rightarrow w - \epsilon \frac{d}{dw} L_i$$

Compute the derivative:

$$\begin{aligned} \frac{d}{dw} L_i &= \frac{d}{dw} (r_i - wu_i)^2 \\ &= (-u_i)\ 2(r_i - wu_i) \\ &= -2u_i \delta_i \end{aligned}$$

prediction error

$\delta_i = r_i - wu_i = r_i - v_i$

The Rescorla-Wagner model

"Loss" in the i-th trial:

$$L_i = (r_i - wu_i)^2$$

Update parameter w to decrease loss!

$$w \to w - \epsilon \frac{d}{dw} L_i \qquad \frac{d}{dw} L_i = -2u_i\delta_i$$

Therefore, $\epsilon \to \epsilon/2$

$$w \to w + \epsilon\delta_i u_i$$

The Rescorla-Wagner model

"Loss" in the i-th trial:

$$L_i = (r_i - w u_i)^2$$

Update parameter w to decrease loss!

$$w \rightarrow w - \epsilon \frac{d}{dw} L_i$$

$$\frac{d}{dw} L_i = -2 u_i \delta_i$$

Rescorla-Wagner-rule

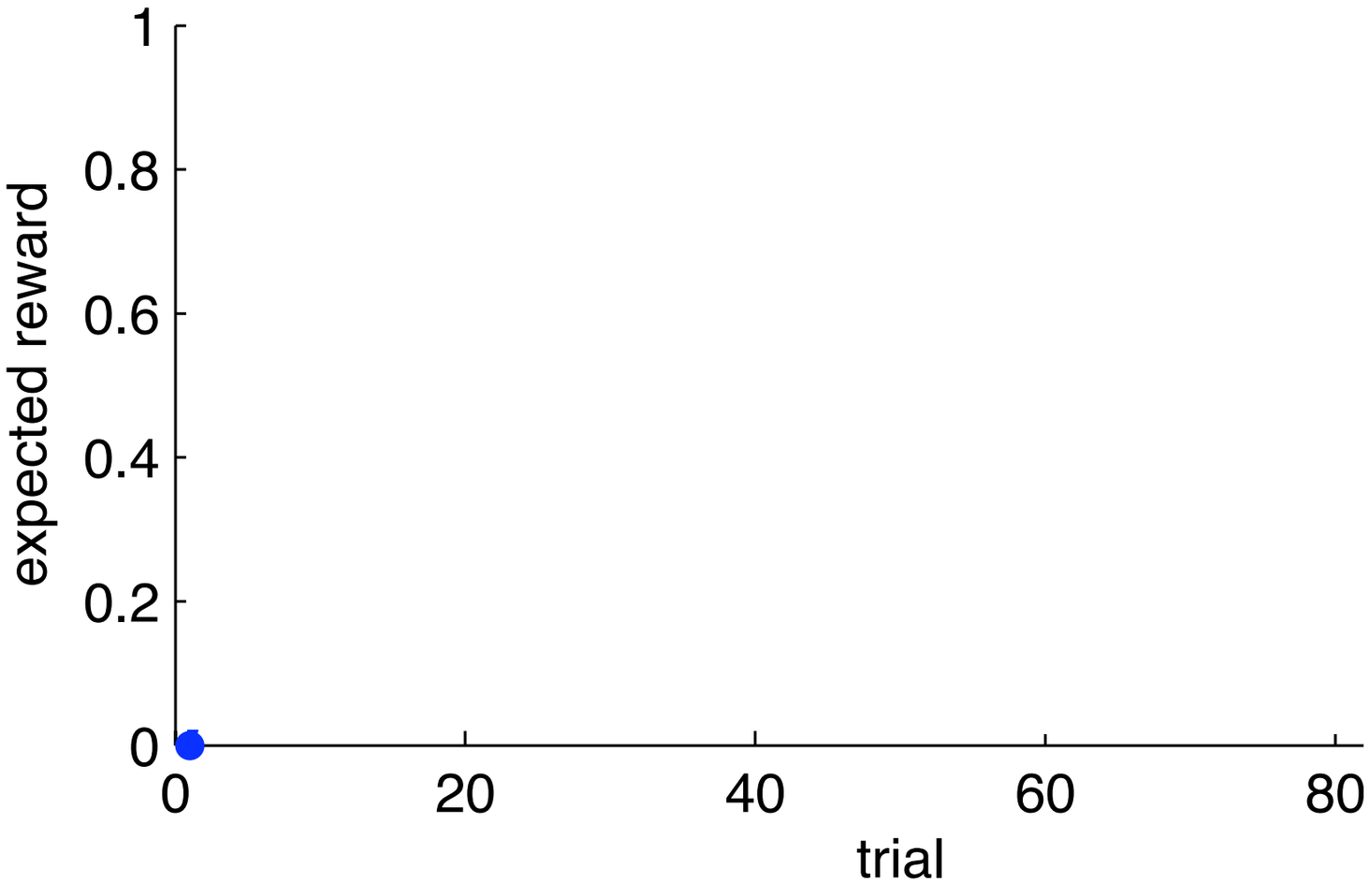
$$w \longrightarrow w + \epsilon \delta_i u_i$$

"delta-rule"

Conditioning explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 1$



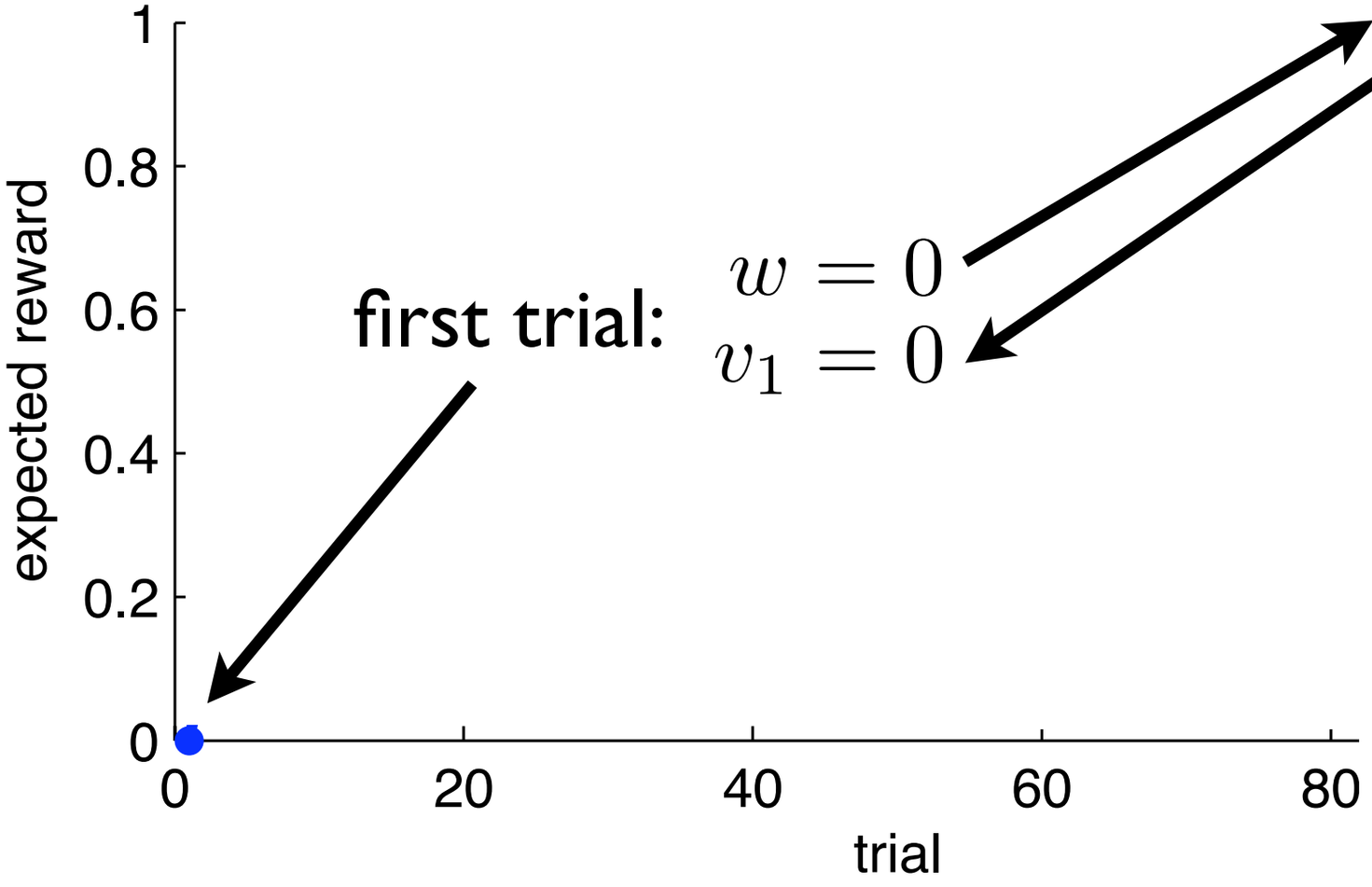
Conditioning explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 1$

expected reward

$v_i = w u_i$



Conditioning explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 1$

expected reward

$v_i = w u_i$

prediction error

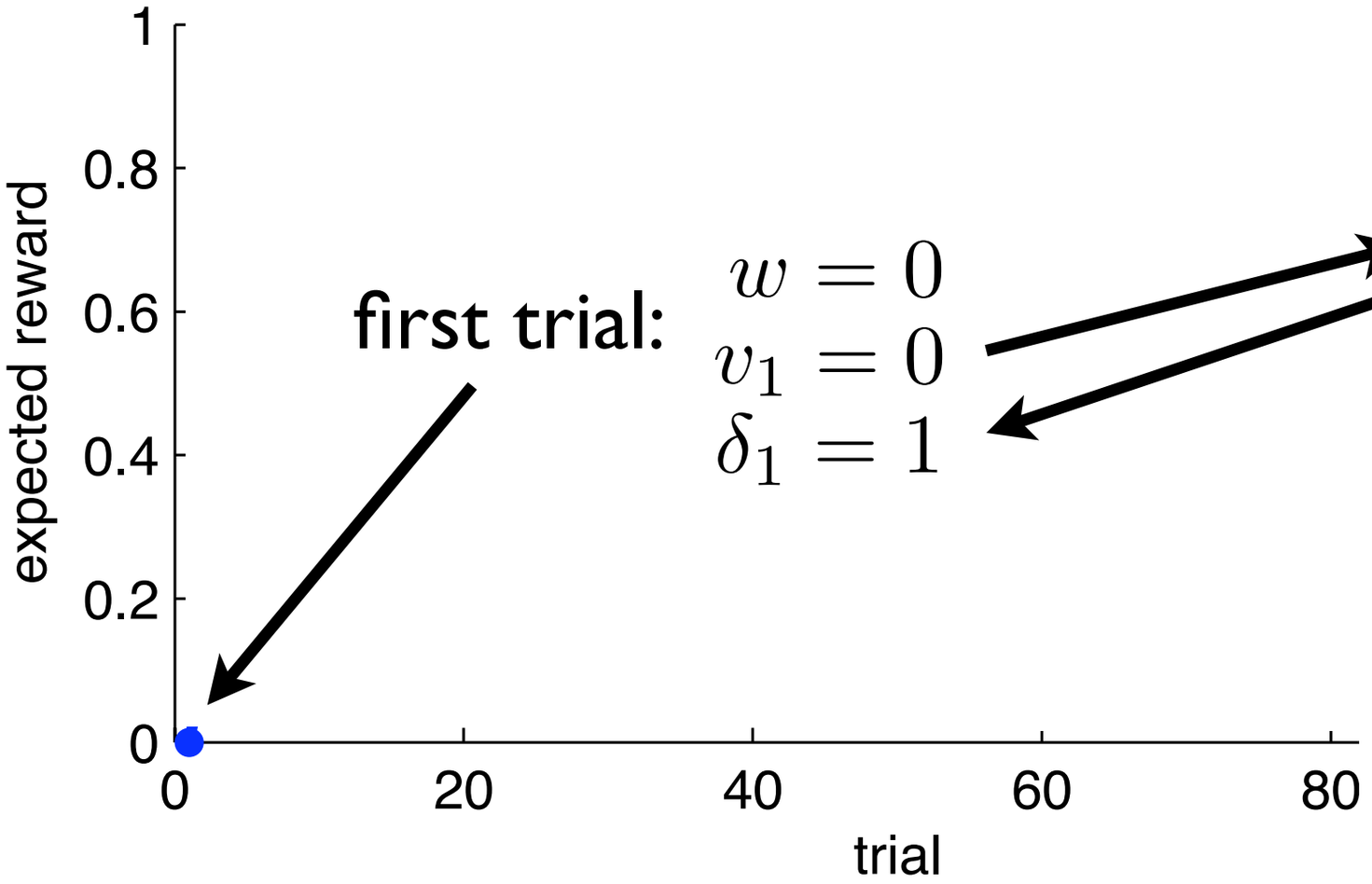
$\delta_i = r_i - v_i$

first trial:

$w = 0$

$v_1 = 0$

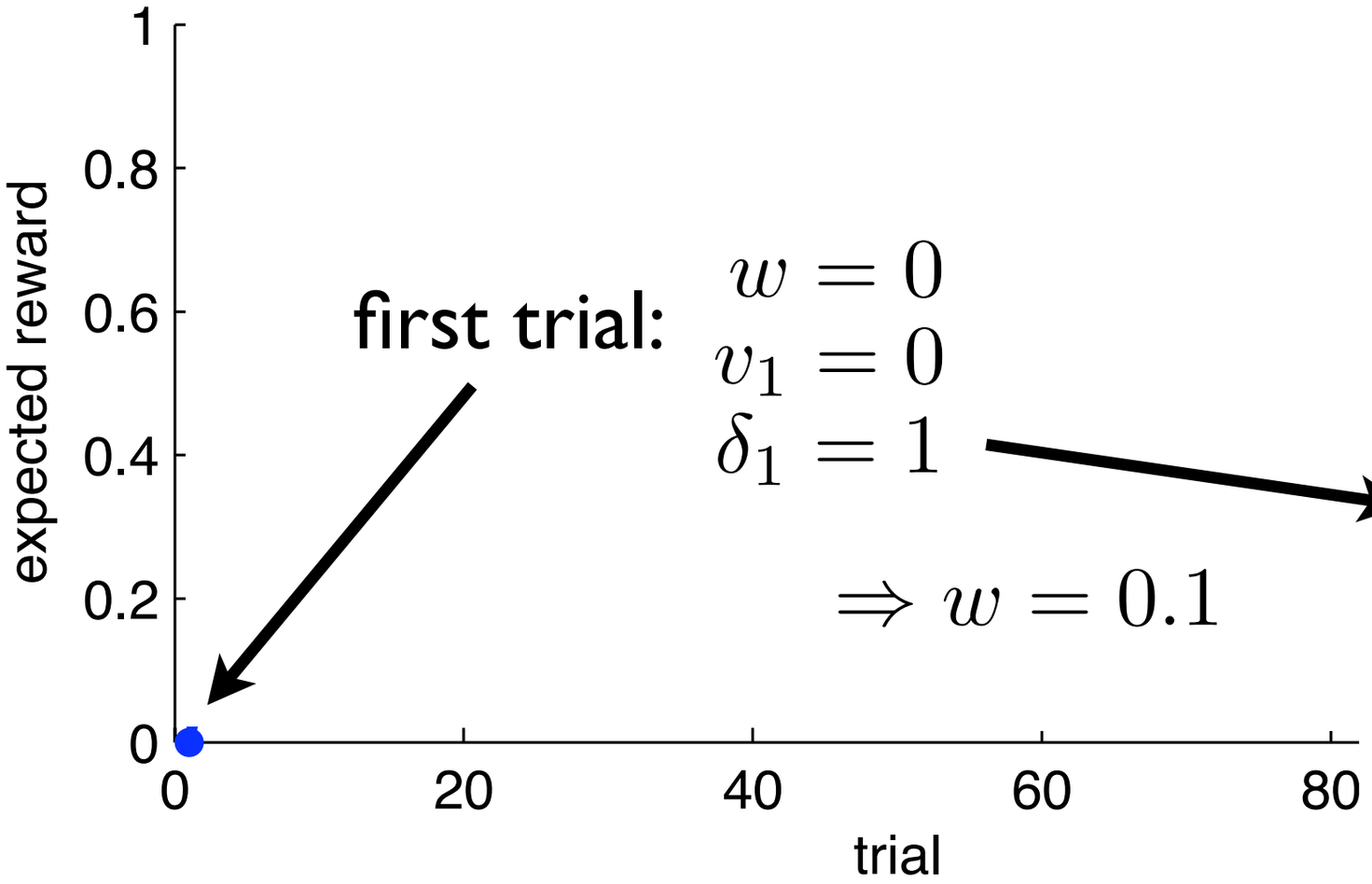
$\delta_1 = 1$



Conditioning explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 1$



first trial:

$w = 0$

$v_1 = 0$

$\delta_1 = 1$

$\Rightarrow w = 0.1$

expected reward

$v_i = w u_i$

prediction error

$\delta_i = r_i - v_i$

learning rule

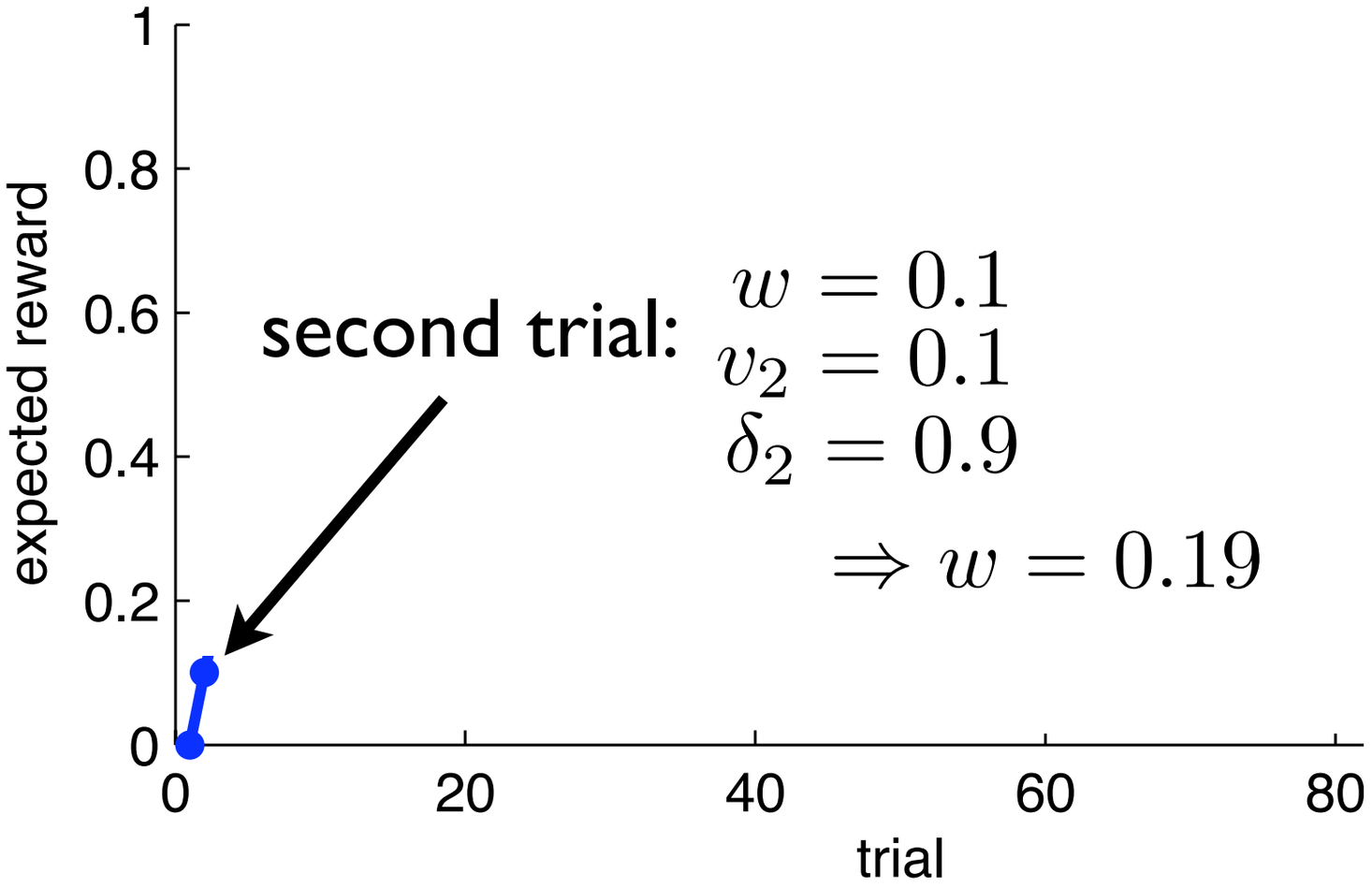
$w \rightarrow w + \epsilon \delta_i u_i$

$\epsilon = 0.1$

Conditioning explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 1$



expected reward

$$v_i = w u_i$$

prediction error

$$\delta_i = r_i - v_i$$

learning rule

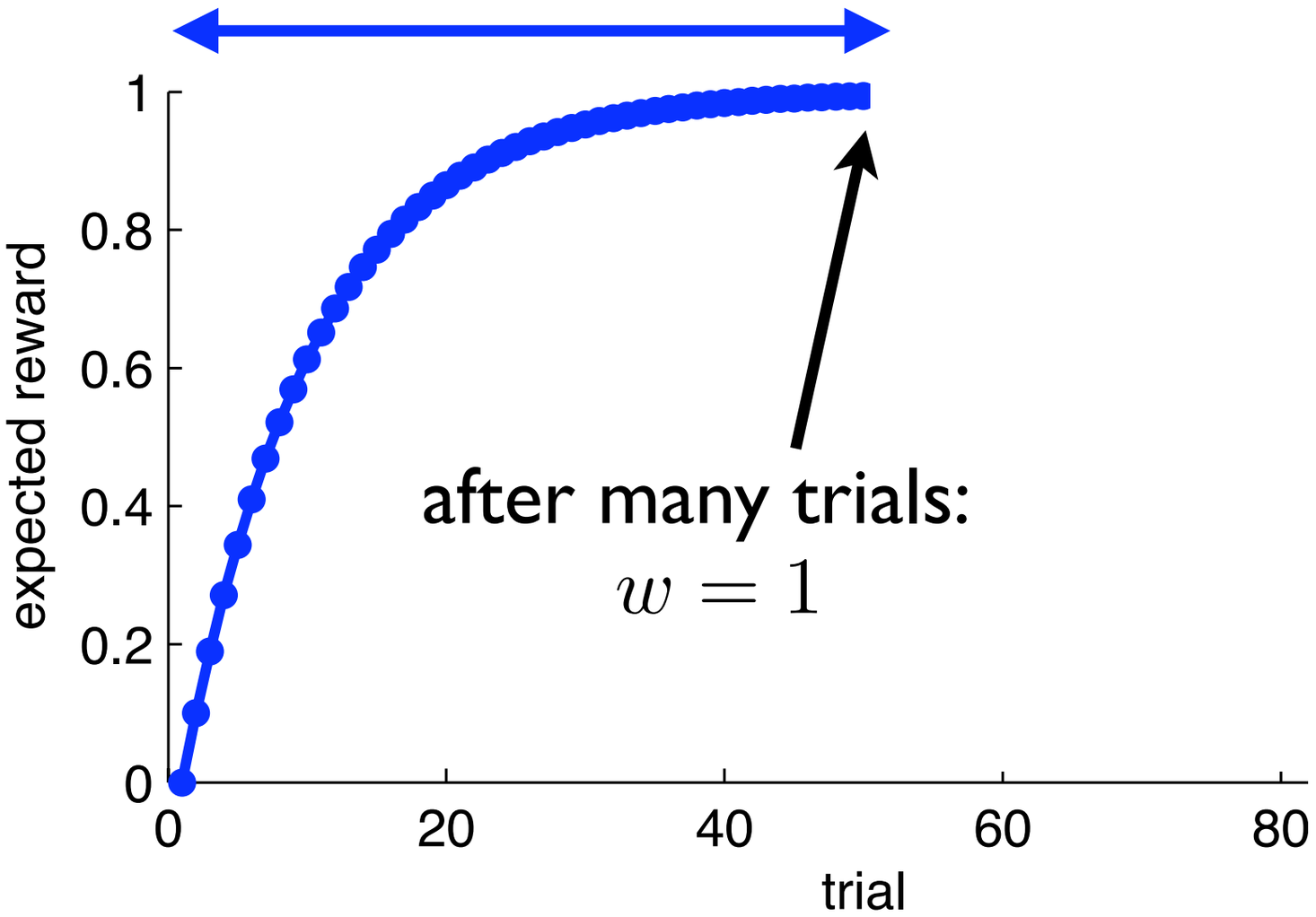
$$w \longrightarrow w + \epsilon \delta_i u_i$$

$$\epsilon = 0.1$$

Conditioning explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 1$



after many trials:

$w = 1$

expected reward

$v_i = w u_i$

prediction error

$\delta_i = r_i - v_i$

learning rule

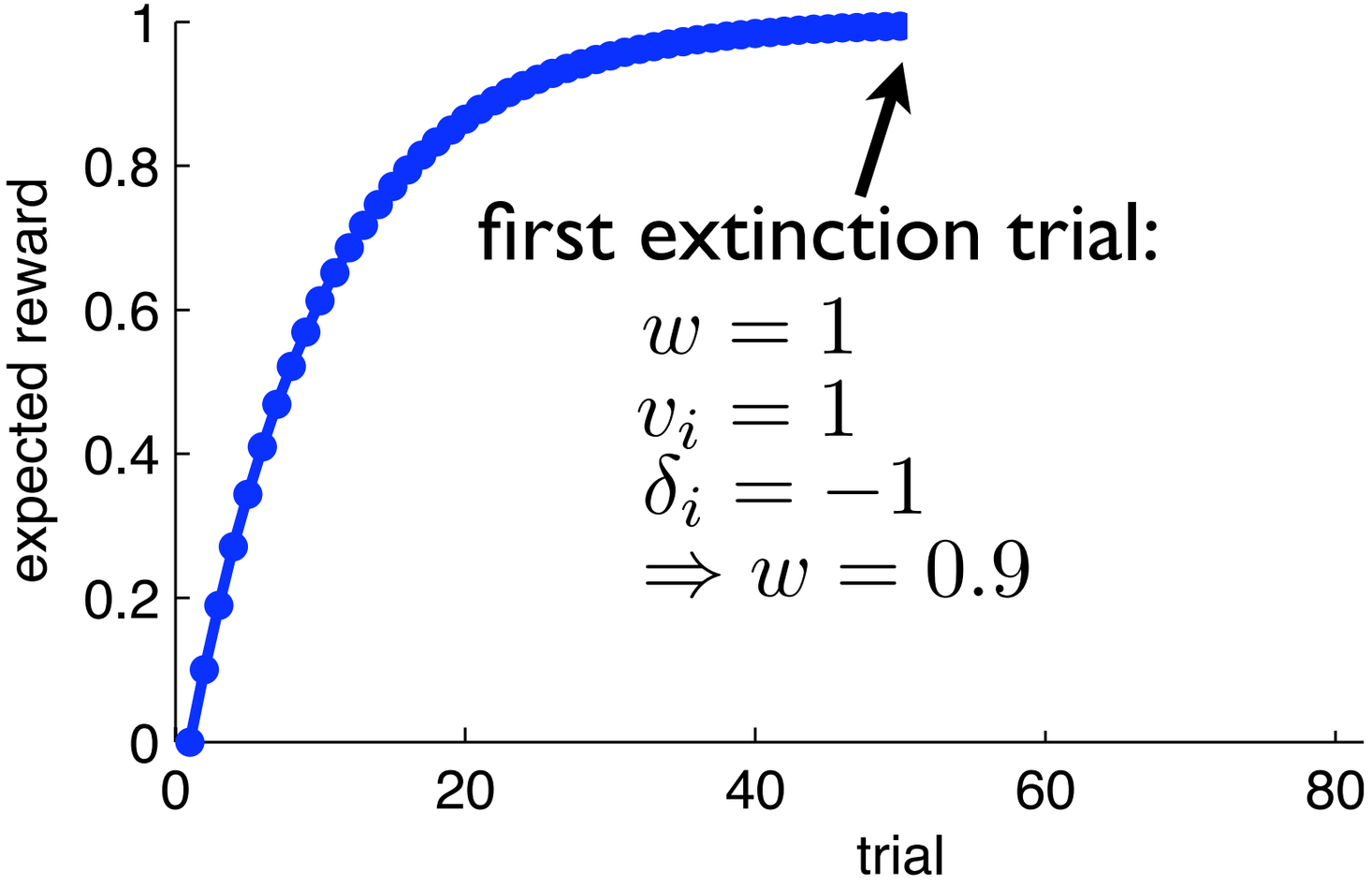
$w \longrightarrow w + \epsilon \delta_i u_i$

$\epsilon = 0.1$

Extinction explained

stimulus (CS) $u_i = 1$

reward (US) $r_i = 0$



first extinction trial:

$$w = 1$$
$$v_i = 1$$
$$\delta_i = -1$$
$$\Rightarrow w = 0.9$$

expected reward

$$v_i = w u_i$$

prediction error

$$\delta_i = r_i - v_i$$

learning rule

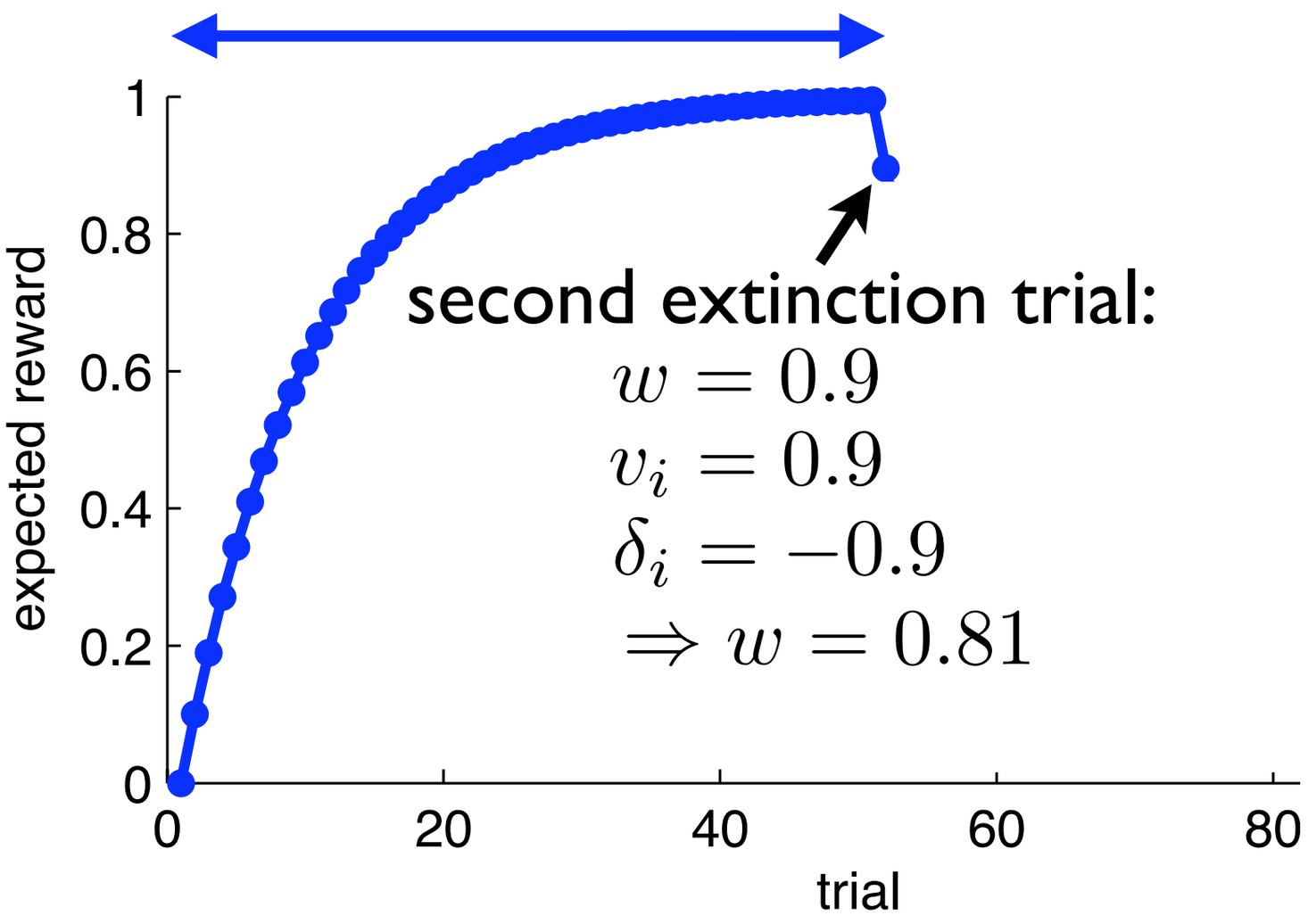
$$w \longrightarrow w + \epsilon \delta_i u_i$$
$$\epsilon = 0.1$$

Extinction explained

stimulus + reward (CS + US)

$u_i = 1$

$r_i = 0$



second extinction trial:

$w = 0.9$

$v_i = 0.9$

$\delta_i = -0.9$

$\Rightarrow w = 0.81$

expected reward

$v_i = w u_i$

prediction error

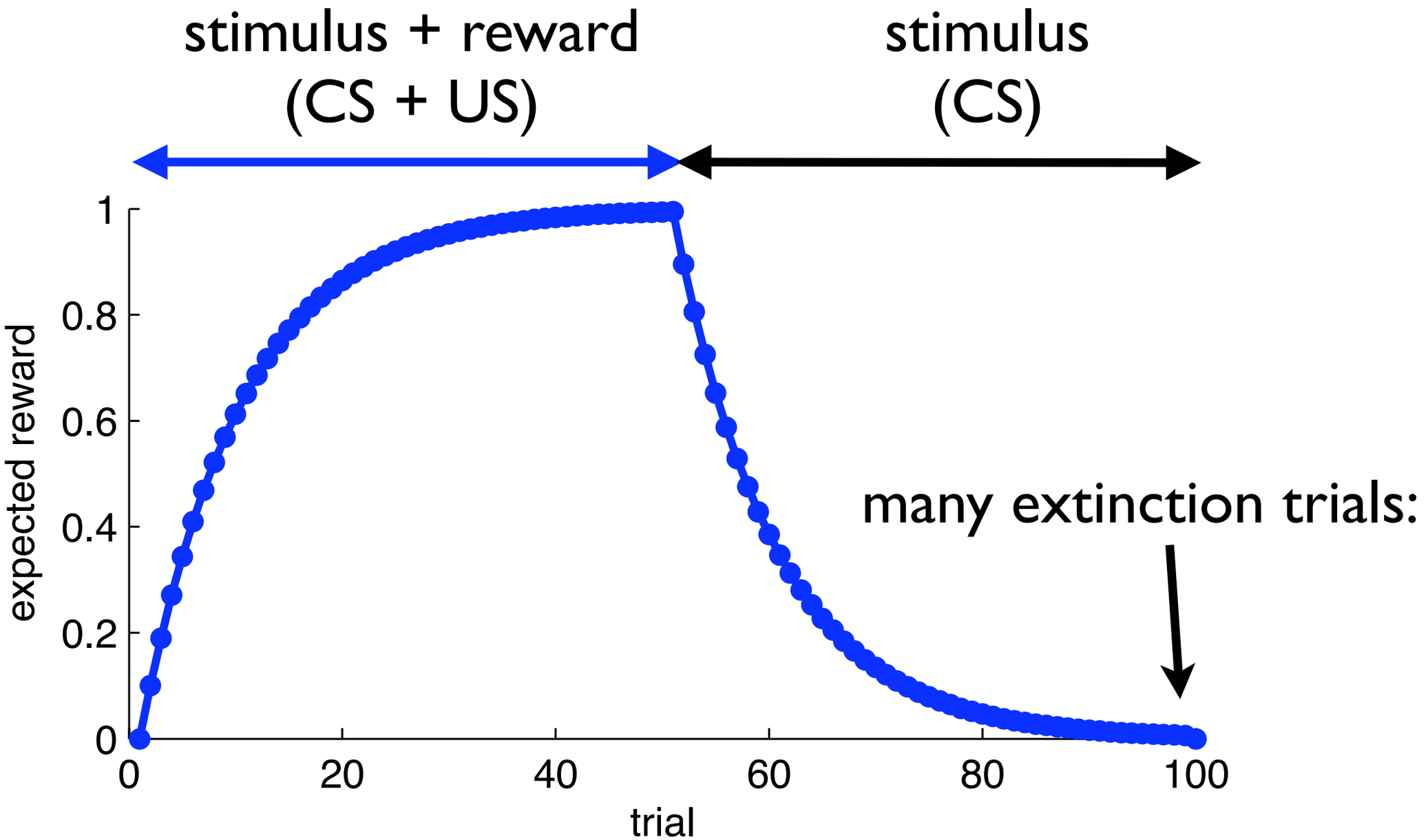
$\delta_i = r_i - v_i$

learning rule

$w \longrightarrow w + \epsilon \delta_i u_i$

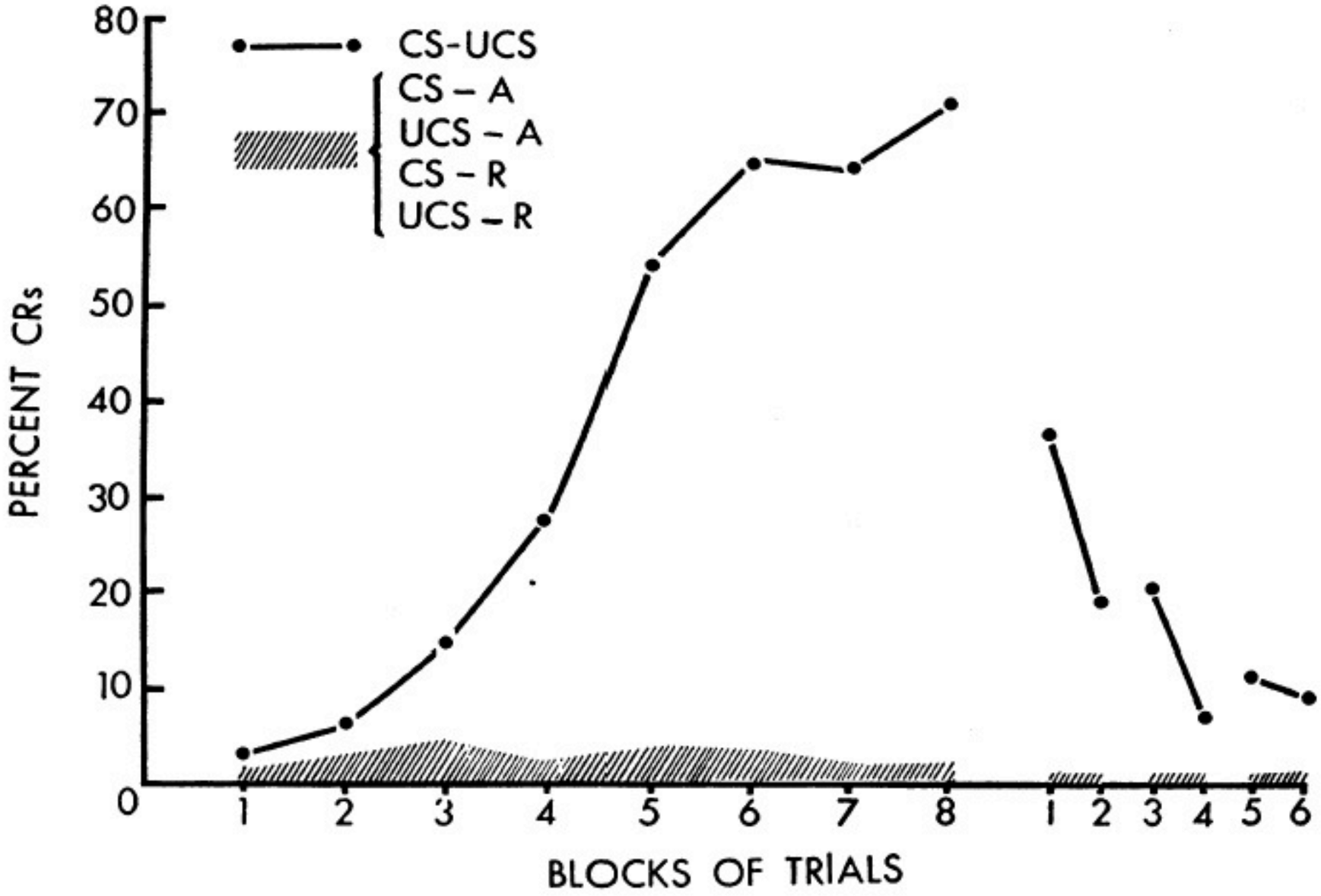
$\epsilon = 0.1$

Extinction explained



A Conditioning Experiment

Rabbit eye blinking



Schneiderman et al, Science, 1962

Classical conditioning: blocking

training	1.	sound	→ food	→ salivation
	2.	clapping sound	→ food	→ salivation
testing	3.	sound	→	salivation
	4.	clapping	→	no salivation

Rescorla-Wagner rule vectorized

reward prediction with two stimuli (i-th trial)

$$v_i = w_1 u_{1i} + w_2 u_{2i}$$

expected reward

weight for stimulus 1

stimulus 1

weight for stimulus 2

stimulus 2

Rescorla-Wagner rule vectorized

reward prediction with two stimuli (i-th trial)

$$v_i = w_1 u_{1i} + w_2 u_{2i} = \mathbf{w} \cdot \mathbf{u}_i$$

expected reward

weight vector

stimulus vector

Rescorla-Wagner rule vectorized

reward prediction with two stimuli (i-th trial)

$$v_i = w_1 u_{1i} + w_2 u_{2i} = \mathbf{w} \cdot \mathbf{u}_i$$

expected reward

weight vector

stimulus vector

Rescorla-Wagner learning rule ("delta rule")

$$w_1 \longrightarrow w_1 + \epsilon \delta_i u_{1i}$$
$$w_2 \longrightarrow w_2 + \epsilon \delta_i u_{2i}$$

prediction error

$$\delta_i = r_i - v_i$$

Rescorla-Wagner rule vectorized

reward prediction with two stimuli (i-th trial)

$$v_i = w_1 u_{1i} + w_2 u_{2i} = \mathbf{w} \cdot \mathbf{u}_i$$

expected reward

weight vector

stimulus vector

Rescorla-Wagner learning rule ("delta rule")

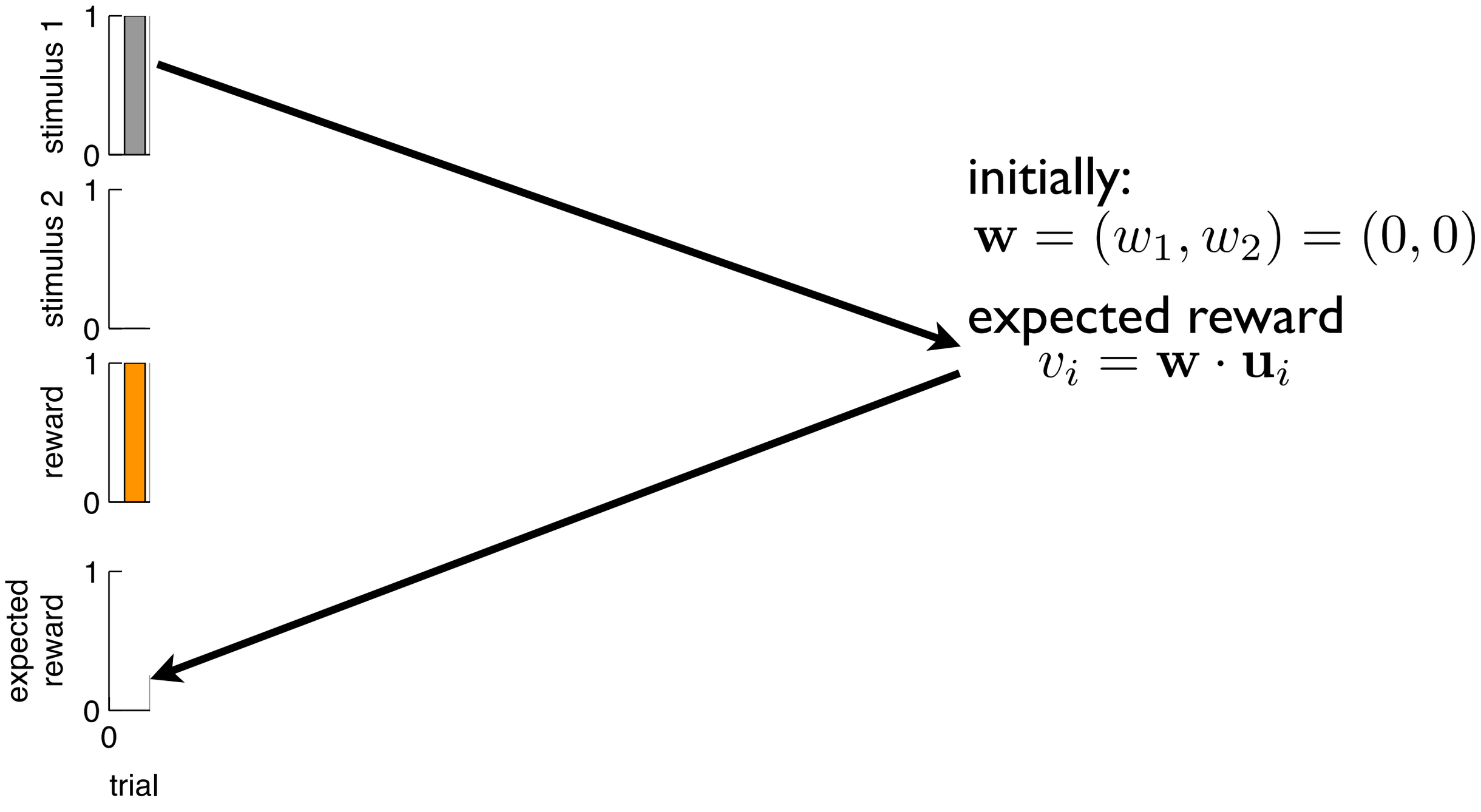
$$w_1 \longrightarrow w_1 + \epsilon \delta_i u_{1i}$$
$$w_2 \longrightarrow w_2 + \epsilon \delta_i u_{2i}$$

prediction error
$\delta_i = r_i - v_i$

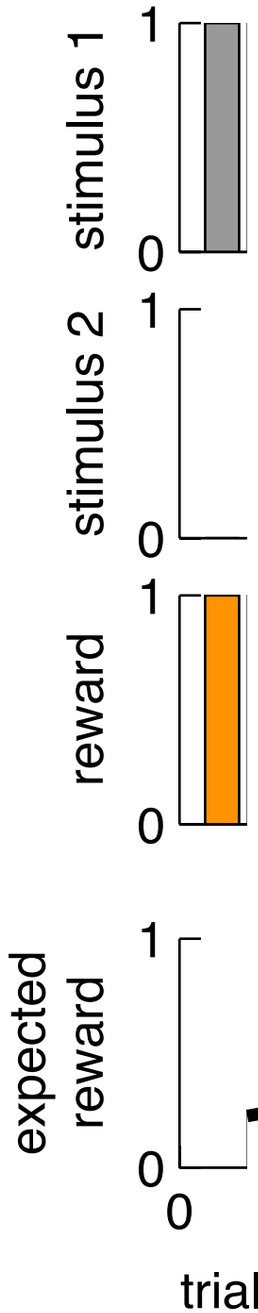
Short-hand vector notation:

$$\mathbf{w} \longrightarrow \mathbf{w} + \epsilon \delta_i \mathbf{u}_i$$

Classical conditioning: blocking



Classical conditioning: blocking



initially:

$$\mathbf{w} = (w_1, w_2) = (0, 0)$$

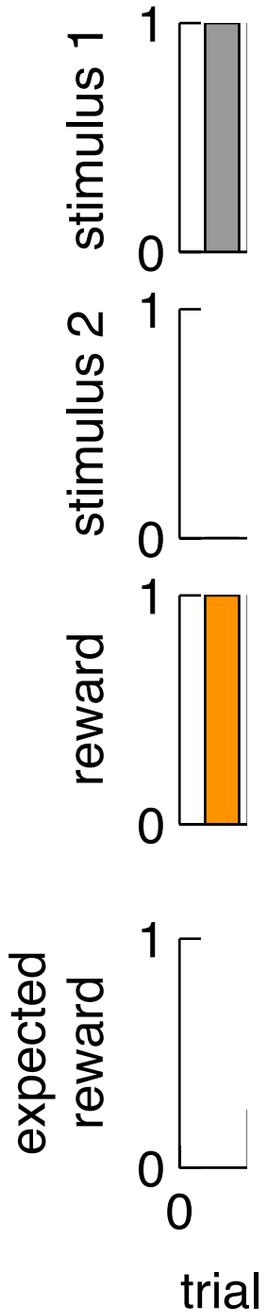
expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

$$\delta_i = r_i - v_i$$

Classical conditioning: blocking



initially:

$$\mathbf{w} = (w_1, w_2) = (0, 0)$$

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

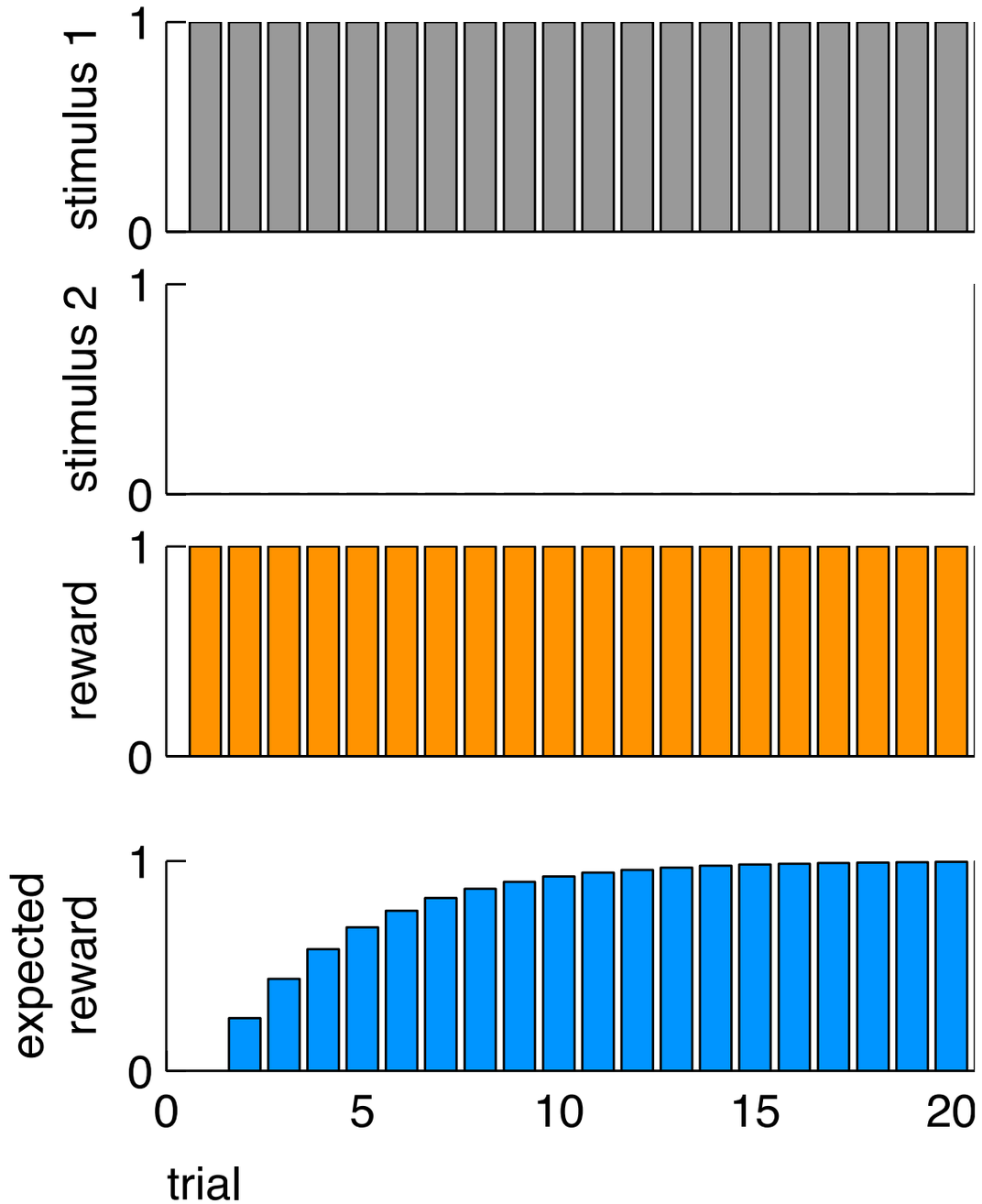
prediction error

$$\delta_i = r_i - v_i$$

learning rule

$$\mathbf{w} \longrightarrow \mathbf{w} + \epsilon \delta_i \mathbf{u}_i$$

Classical conditioning: blocking



initially:

$$\mathbf{w} = (w_1, w_2) = (0, 0)$$

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

$$\delta_i = r_i - v_i$$

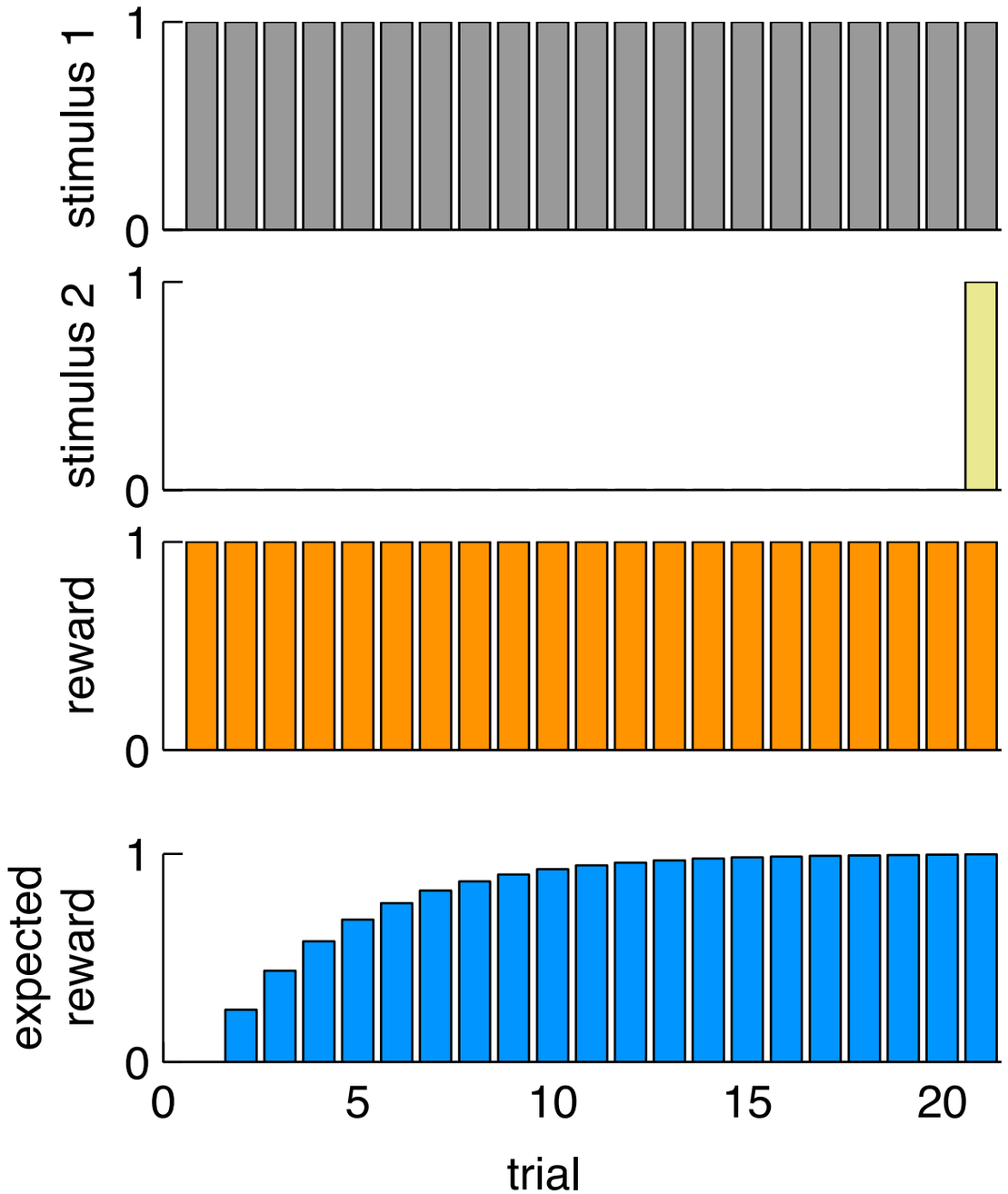
learning rule

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta_i \mathbf{u}_i$$

after several trials:

$$\mathbf{w} = (w_1, w_2) = (1, 0)$$

Classical conditioning: blocking



new stimulus, but:
no prediction error!!

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

$$\delta_i = r_i - v_i$$

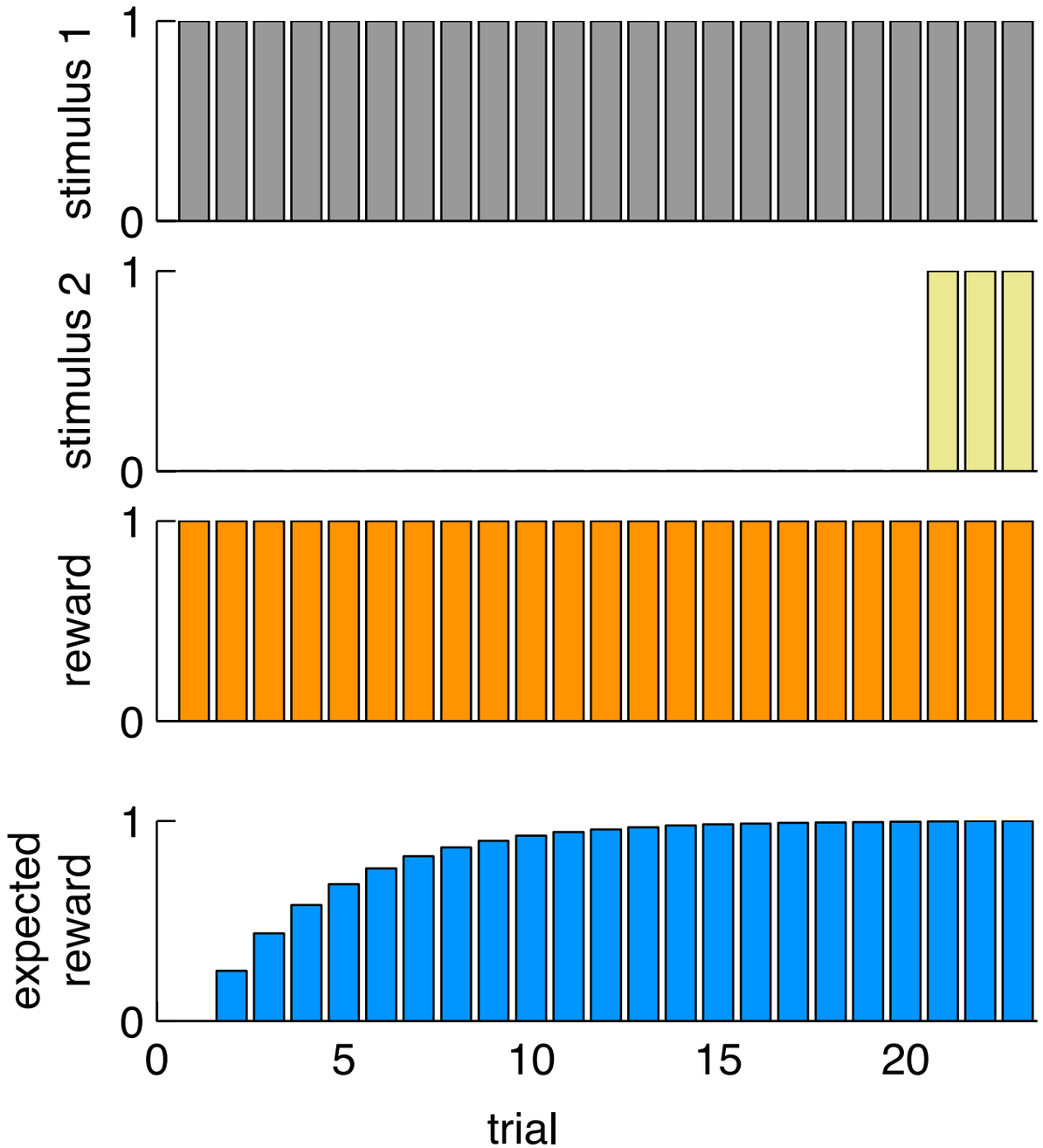
learning rule

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta_i \mathbf{u}_i$$

after several trials:

$$\mathbf{w} = (w_1, w_2) = (1, 0)$$

Classical conditioning: blocking



new stimulus, but:
no prediction error!
→ no learning!

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

$$\delta_i = r_i - v_i$$

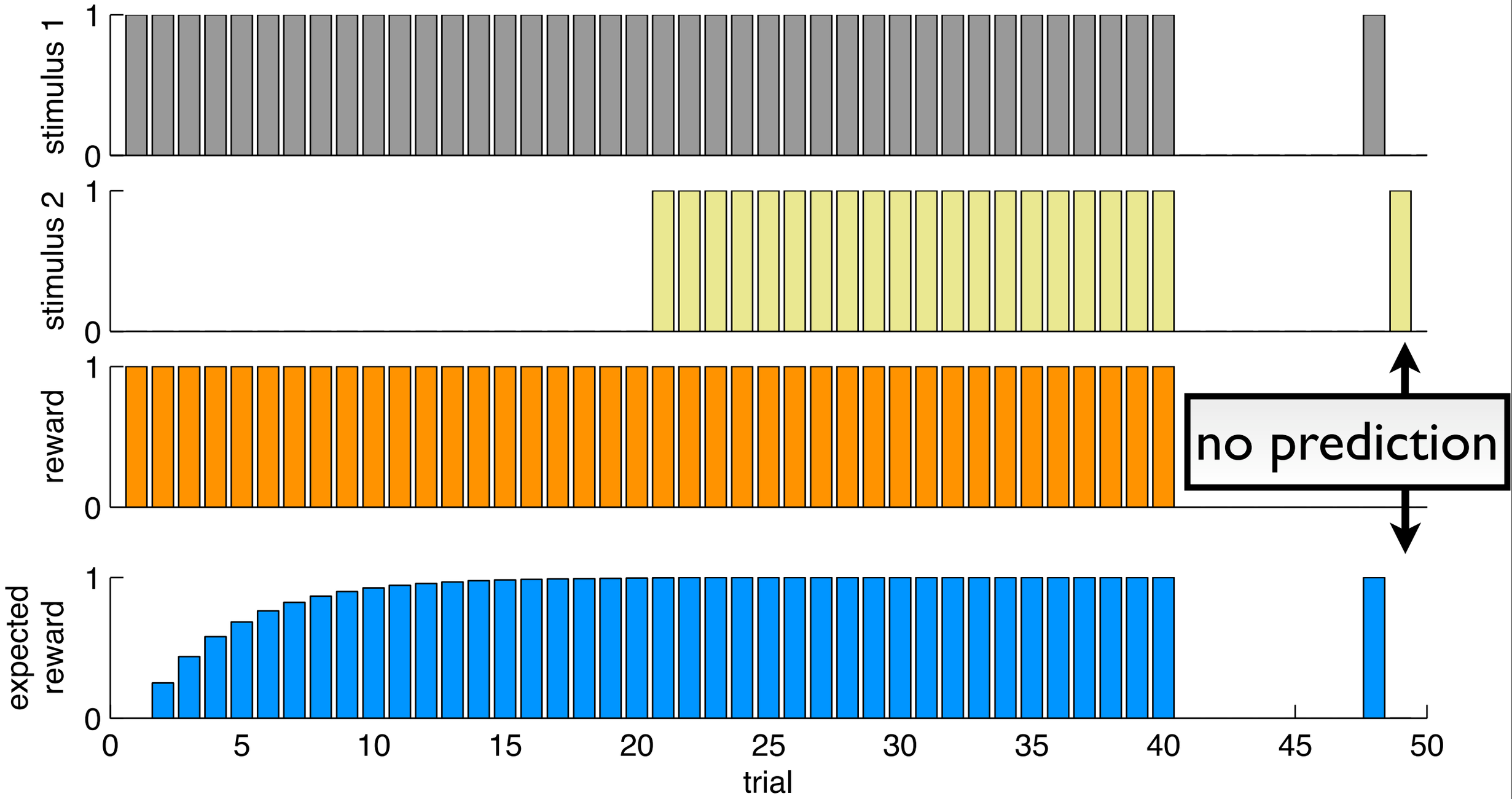
learning rule

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta_i \mathbf{u}_i$$

after several trials:

$$\mathbf{w} = (w_1, w_2) = (1, 0)$$

Classical conditioning: blocking



Inhibitory conditioning

training

1.

sound

food

salivation

2.

clapping

food

salivation

3.

light

clapping

salivation

disappears

testing

4.

sound

salivation

5.

light

sound

reduced salivation

Rescorla-Wagner rule vectorized

Simple linear model (i-th trial)

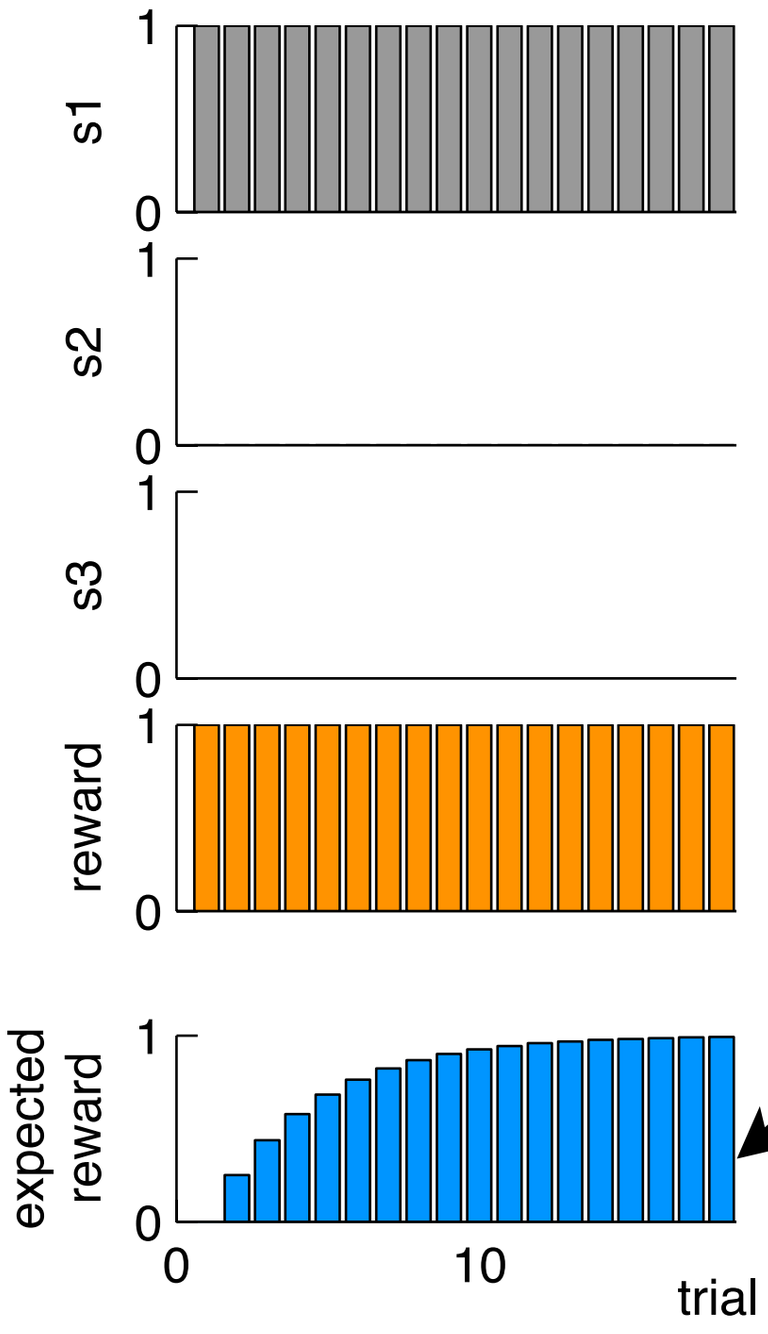
$$v_i = \mathbf{w} \cdot \mathbf{u}_i = w_1 u_{1i} + w_2 u_{2i} + w_3 u_{3i}$$

expected reward

weight vector

stimulus vector

Inhibitory conditioning



initially:

$$\mathbf{w} = (w_1, w_2, w_3) = (0, 0, 0)$$

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

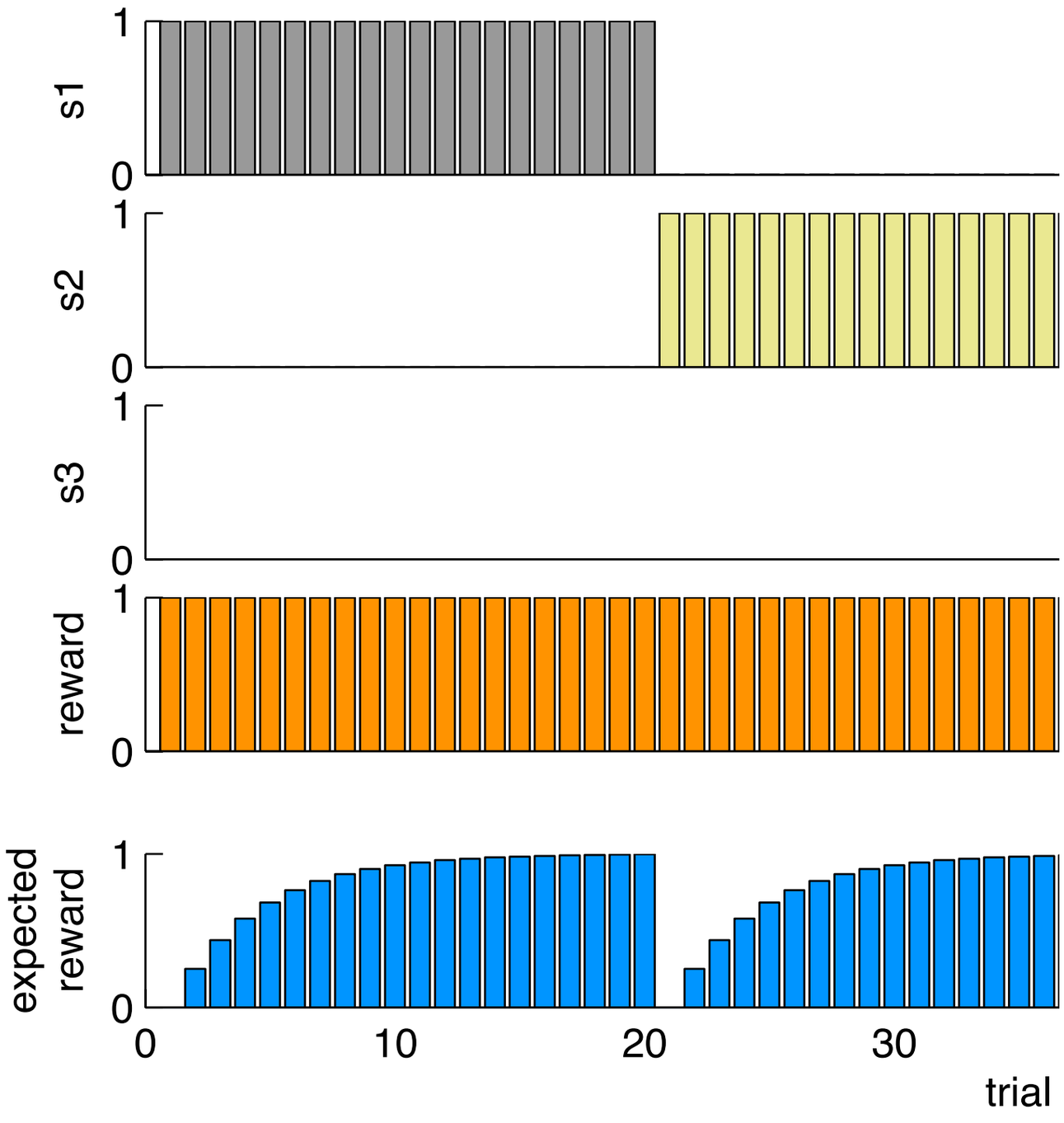
$$\delta_i = r_i - v_i$$

learning rule

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta \mathbf{u}_i$$

→ learn $w_1 = 1$

Inhibitory conditioning



previously:

$$\mathbf{w} = (1, 0, 0)$$

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

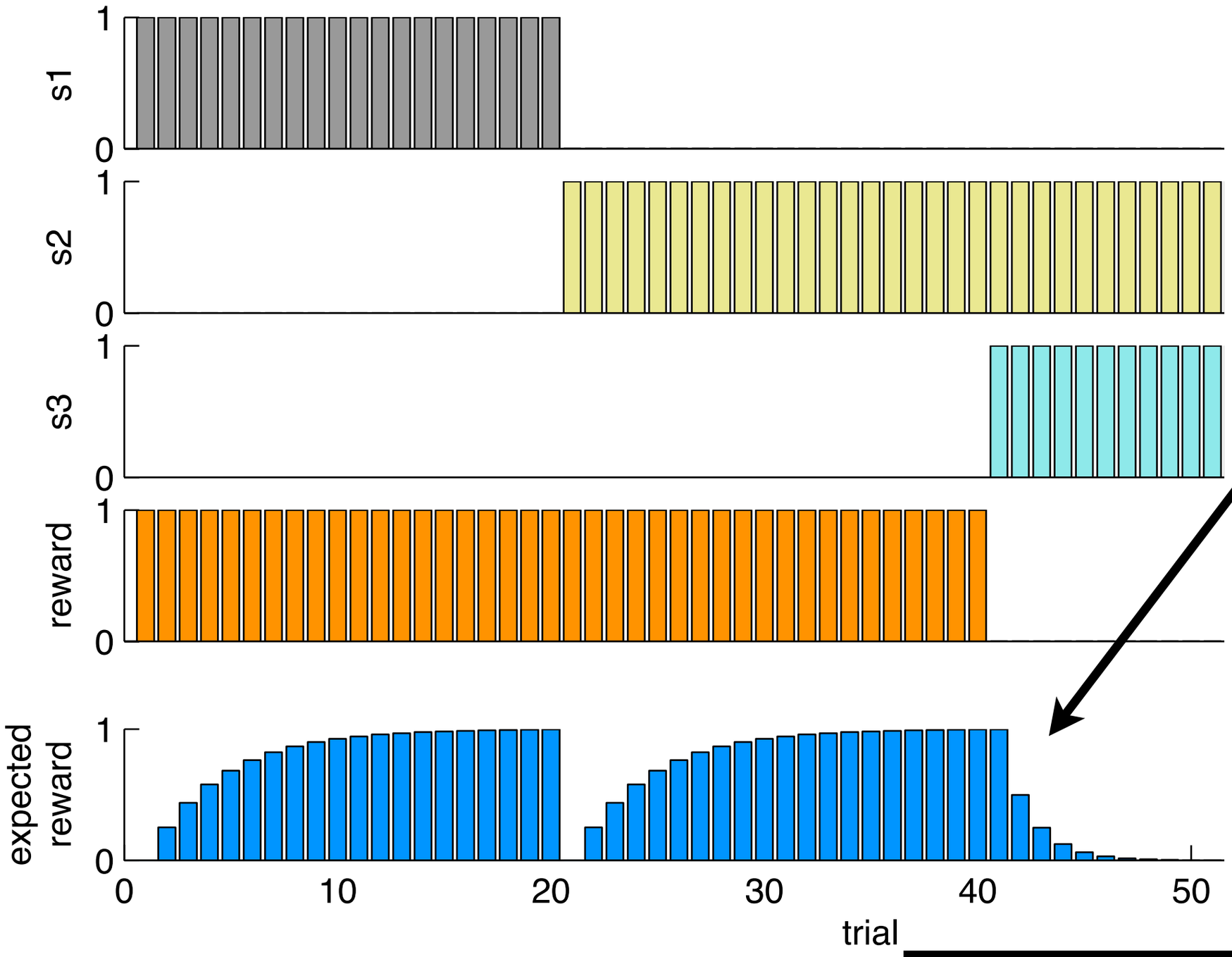
$$\delta_i = r_i - v_i$$

learning rule

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta \mathbf{u}_i$$

→ learn $w_2 = 1$

Inhibitory conditioning



previously:

$$\mathbf{w} = (1, 1, 0)$$

expected reward

$$v_i = \mathbf{w} \cdot \mathbf{u}_i$$

prediction error

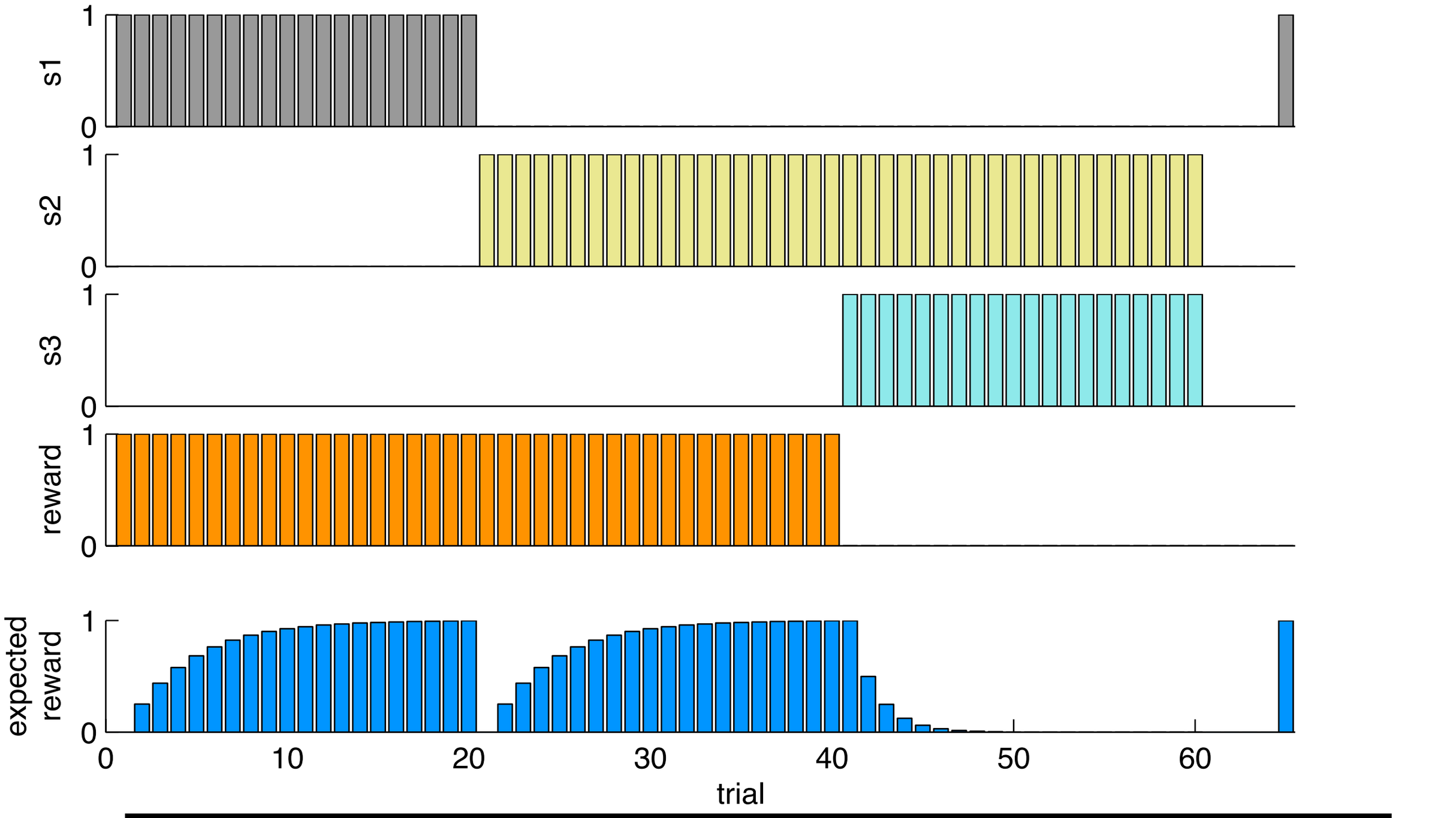
$$\delta_i = r_i - v_i$$

learning rule

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta \mathbf{u}_i$$

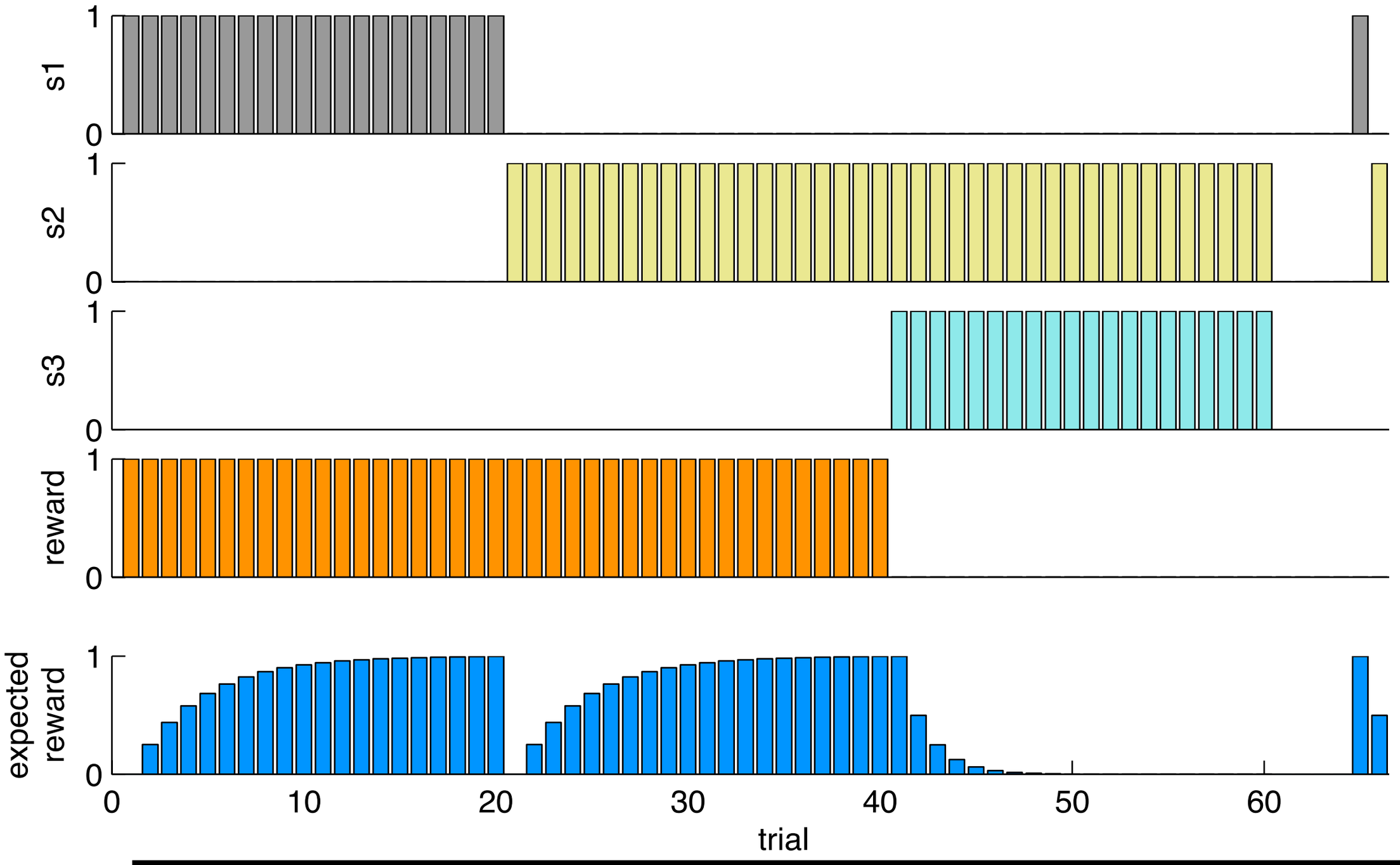
→ learn $\mathbf{w} = (1, 0.5, -0.5)$

Inhibitory conditioning



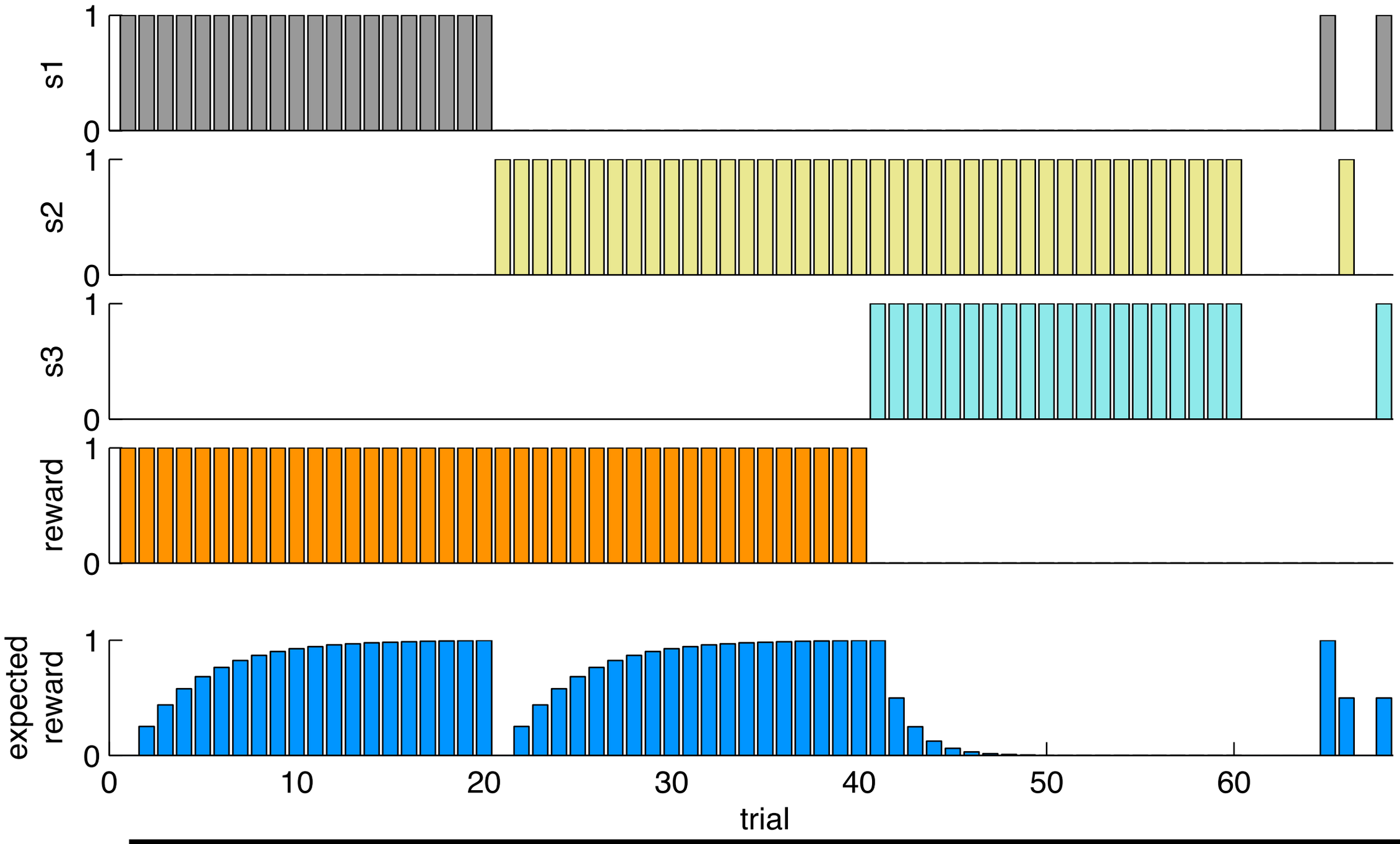
learned weights: $w_1 = 1 \quad w_2 = 0.5 \quad w_3 = -0.5$

Inhibitory conditioning



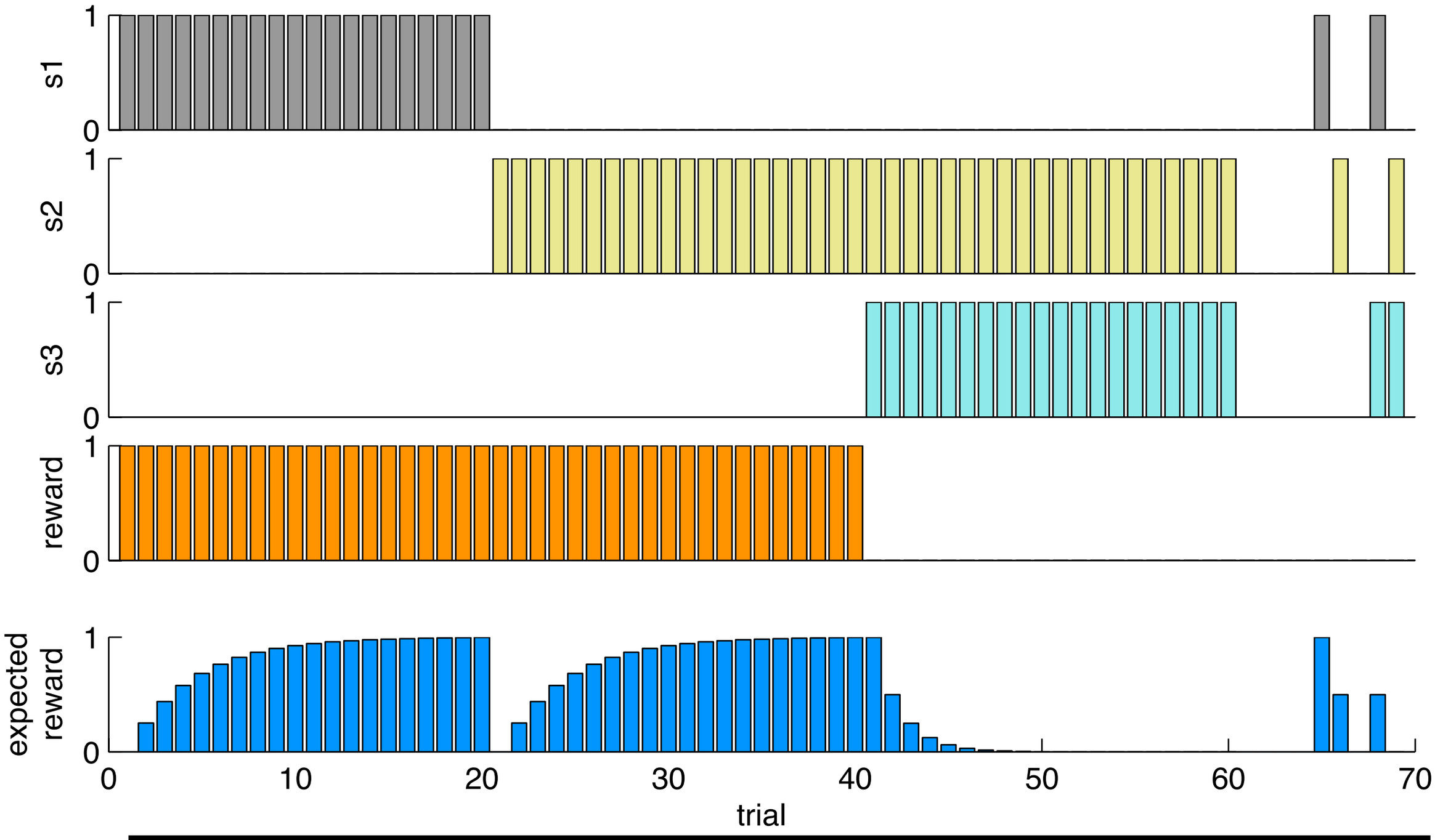
learned weights: $w_1 = 1 \quad w_2 = 0.5 \quad w_3 = -0.5$

Inhibitory conditioning



learned weights: $w_1 = 1 \quad w_2 = 0.5 \quad w_3 = -0.5$

Inhibitory conditioning



learned weights: $w_1 = 1 \quad w_2 = 0.5 \quad w_3 = -0.5$

Secondary conditioning

training

1.

sound

food

salivation

2.

clapping

sound

salivation

*

testing

3.

sound

salivation

4.

clapping

salivation

* if you do this too often,
you get extinction, of course!

Now what?

HOW DO WE ACT BASED ON VALUES WE LEARN?
(OPERANT CONDITIONING)

Part 1.2

The Exploration-Exploitation Dilemma

Choosing Actions in a Changing World

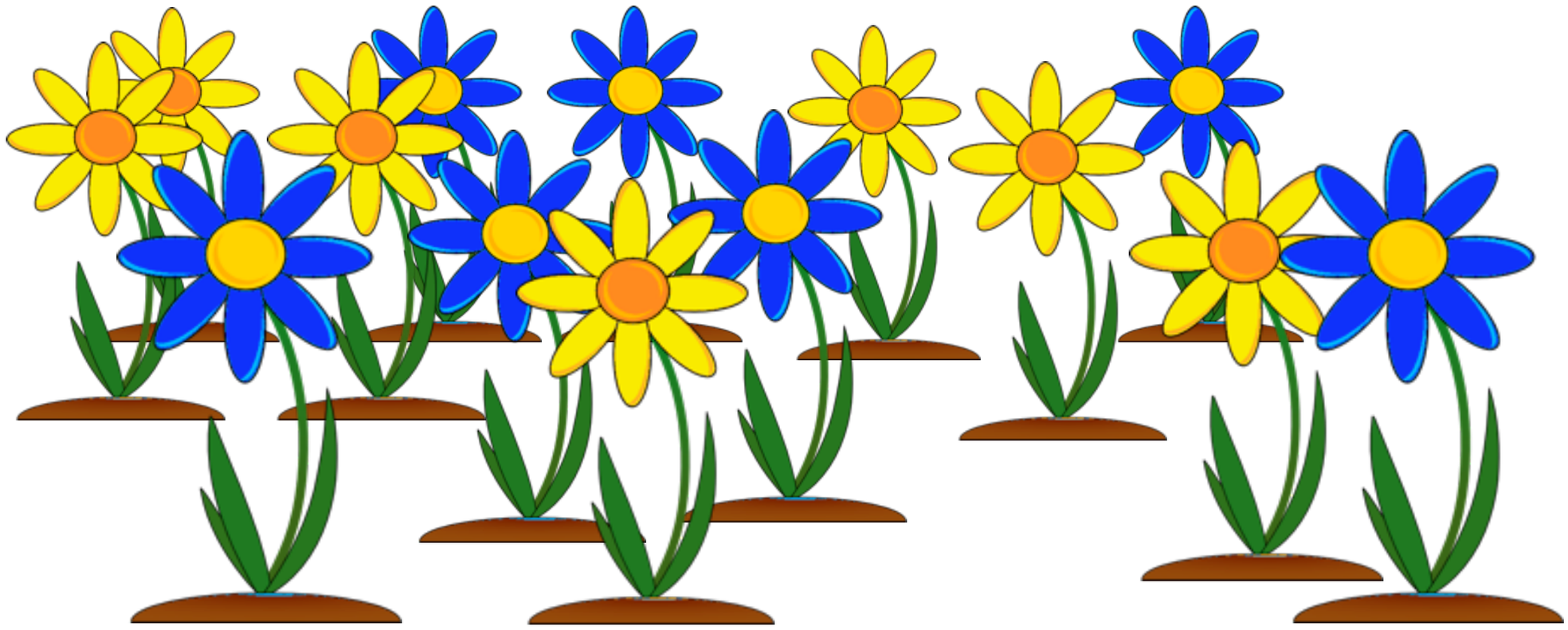
Cogmaster CO6 / Boris Gutkin

Exploration-Exploitation Dilemma

Lunch break

Exploration-Exploitation Dilemma

Possible choices (actions) of a bee:
land on a blue or yellow flower

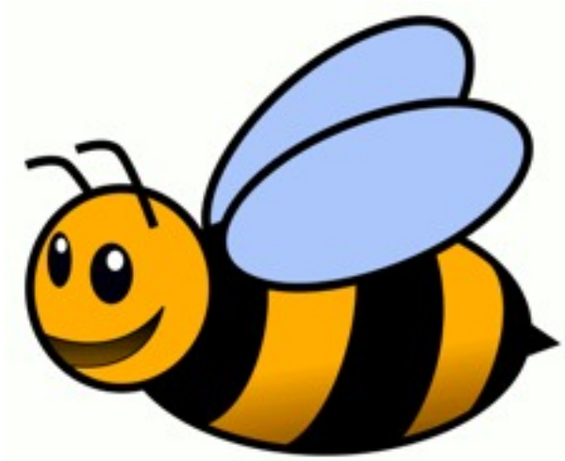


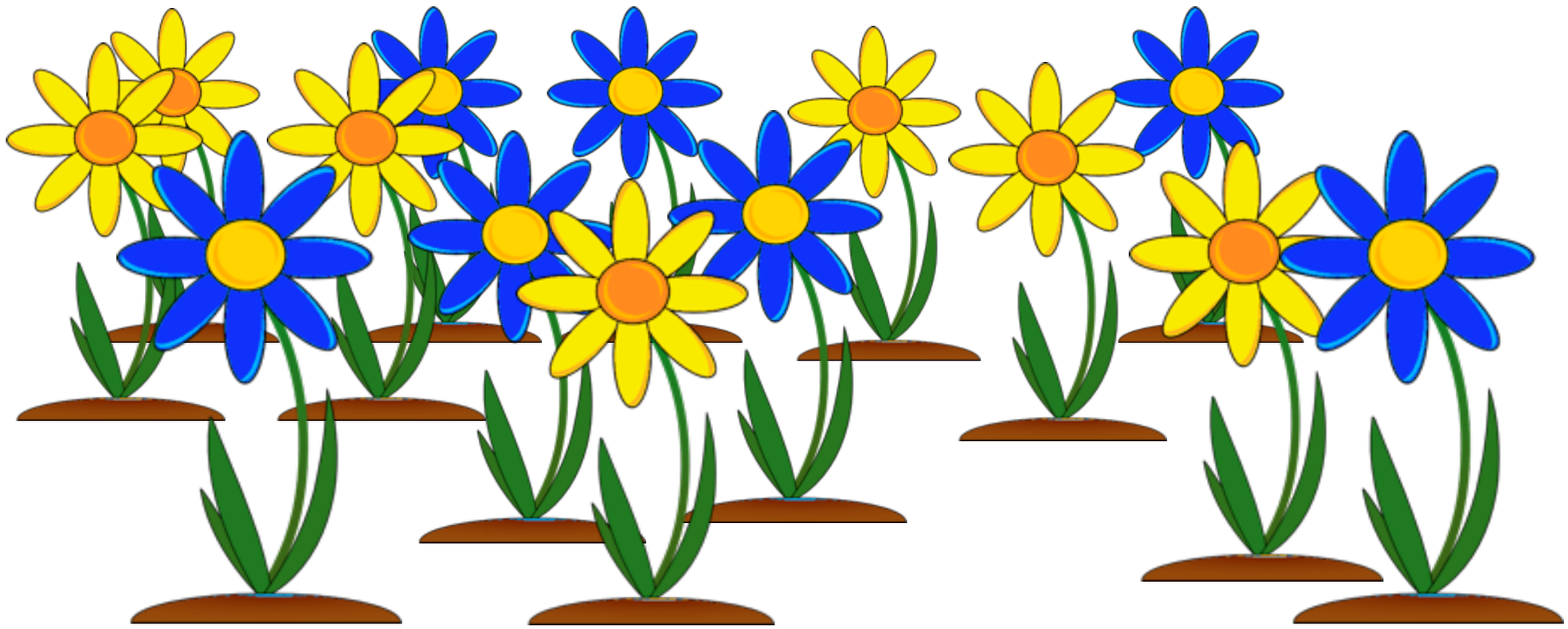
Bee searching for nectar

Possible choices (actions) of the bee:
land on a blue or yellow flower
rewards (in drops of nectar)

$r_b = 8$

$r_y = 2$



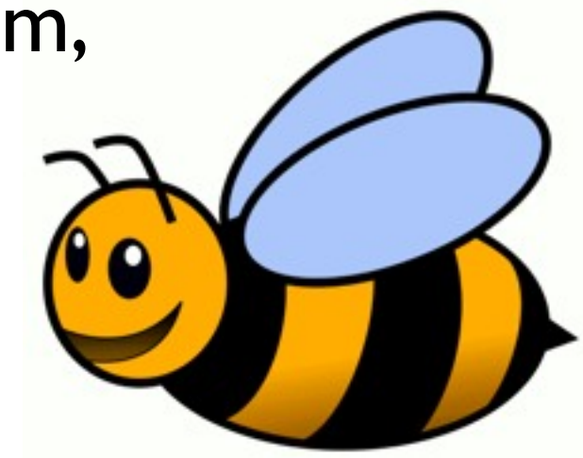


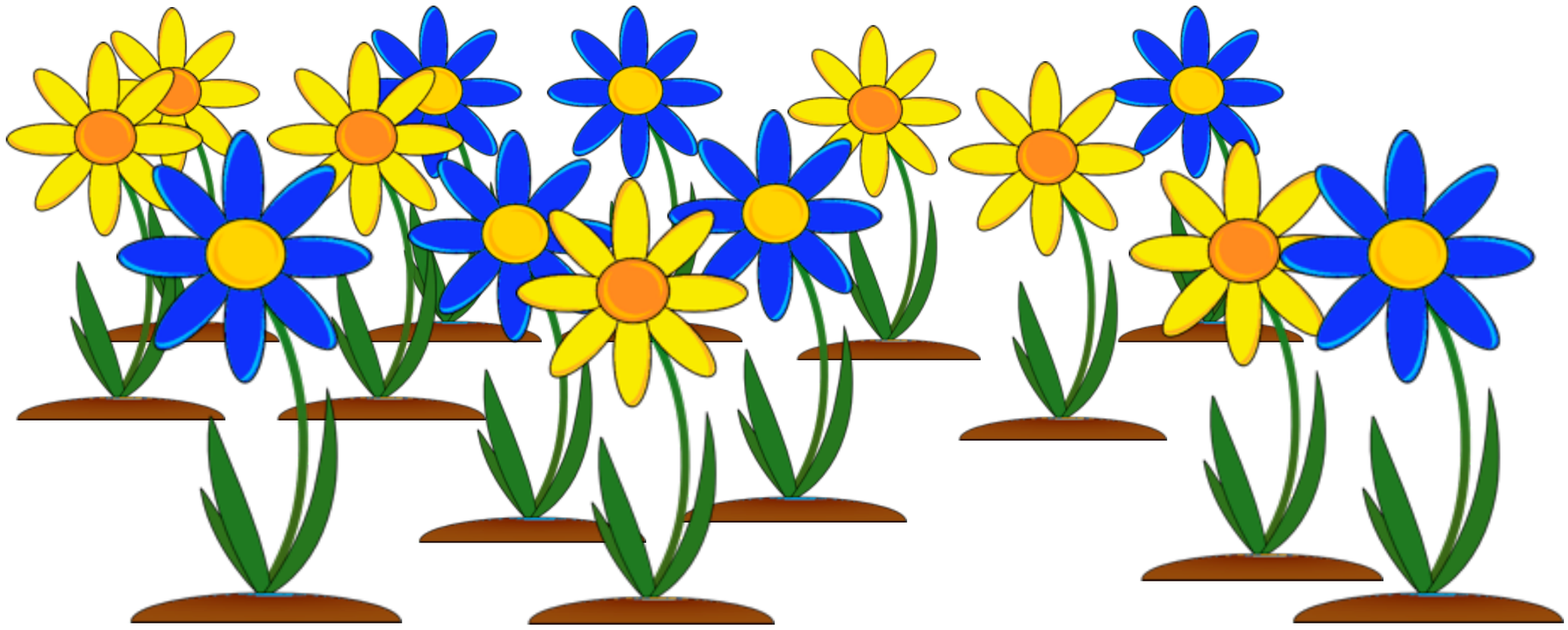
“Policy”: Bee’s plan of action

Assume: choices or actions a are taken at random, according to a probabilistic “policy”:

$$p(a = \text{yellow})$$

$$p(a = \text{blue})$$





Math reminder

Probabilities: Dice Example

Probability that you get A=1

$$p(A = 1) = 1/6$$

Math reminder

Playing dice

Probability that you get A=1

$$p(A = 1) = 1/6$$

Probability that you get either A=1 or A=6

$$p(A = 1 \text{ or } A = 6) = 1/6 + 1/6 = 1/3$$

Math reminder

Playing dice

Probability that you get A=1

$$p(A = 1) = 1/6$$

Probability that you get either A=1 or A=6

$$p(A = 1 \text{ or } A = 6) = 1/6 + 1/6 = 1/3$$

Prob. to get A=1 on the first throw and A=6 on the second:

$$p(A = 1 \text{ and } A = 6) = 1/6 * 1/6 = 1/36$$

Math reminder

Playing dice

Probability that you get A=1

$$p(A = 1) = 1/6$$

Probability that you get either A=1 or A=6

$$p(A = 1 \text{ or } A = 6) = 1/6 + 1/6 = 1/3$$

Prob. to get A=1 on the first throw and A=6 on the second:

$$p(A = 1 \text{ and } A = 6) = 1/6 * 1/6 = 1/36$$

Probability that you will **not** get A=1

$$p(\text{not } A = 1) = 1 - p(A = 1) = 5/6$$

Math reminder

Probabilities: general rules

Probability of event A:

$$p(A) \in [0, 1]$$

Probability of two (mutually exclusive) events:

$$p(A \text{ or } B) = p(A) + p(B)$$

Probability of two independent events:

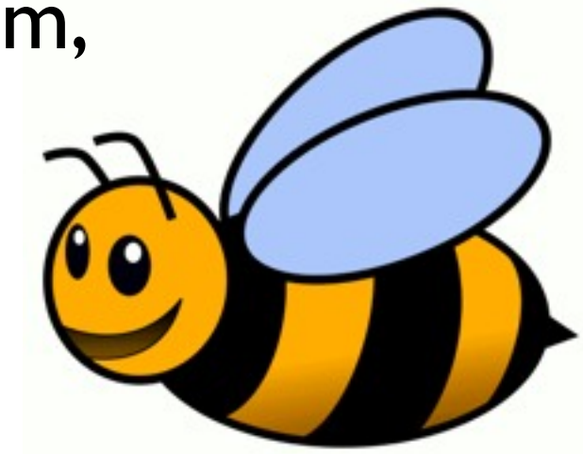
$$p(A \text{ and } B) = p(A)p(B)$$

Tertium non datur: (either A happens or not)

$$p(A) + p(\text{not } A) = 1$$

“Policy”: Bee’s plan of action

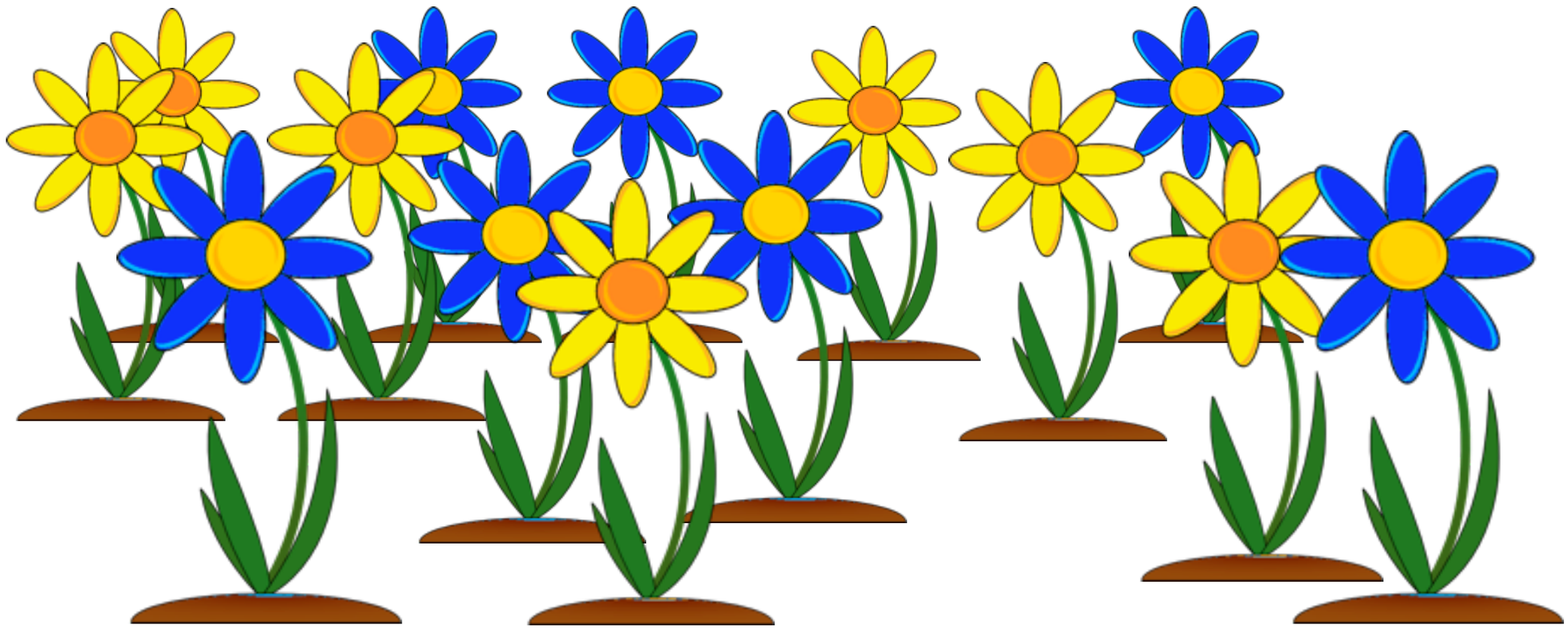
Assume: choices or actions a are taken at random, according to a probabilistic “policy”:



$$p(a = \text{yellow})$$

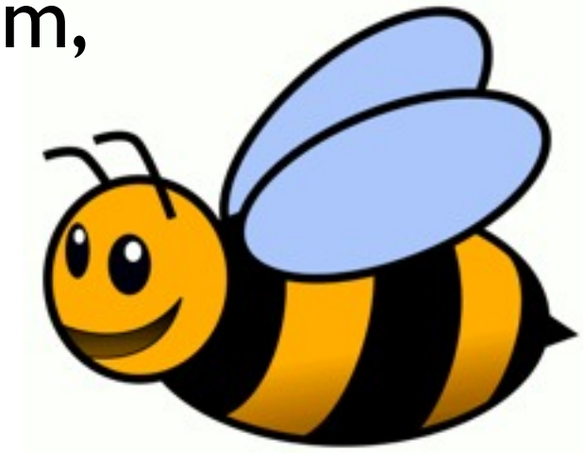
$$p(a = \text{blue})$$

$$p(a = \text{blue}) + p(a = \text{yellow}) = 1$$



“Policy”: Bee’s plan of action

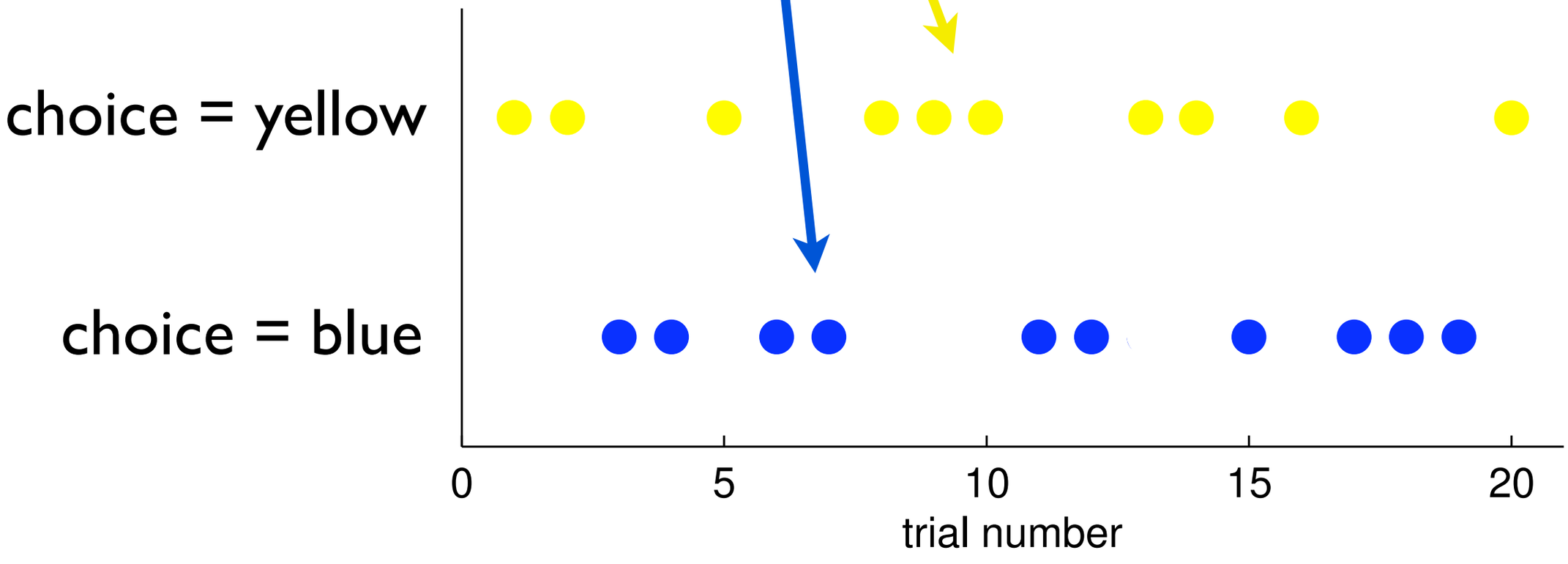
Assume: choices or actions a are taken at random, according to a probabilistic “policy”:



$$p(a = \text{yellow}) = 0.5$$

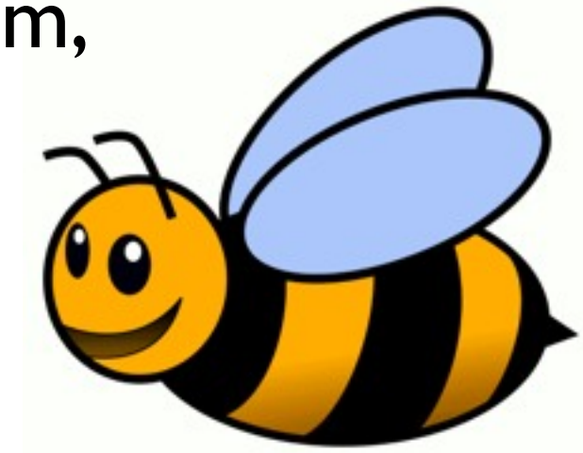
$$p(a = \text{blue}) = 0.5$$

$$p(a = \text{blue}) + p(a = \text{yellow}) = 1$$



“Policy”: Bee’s plan of action

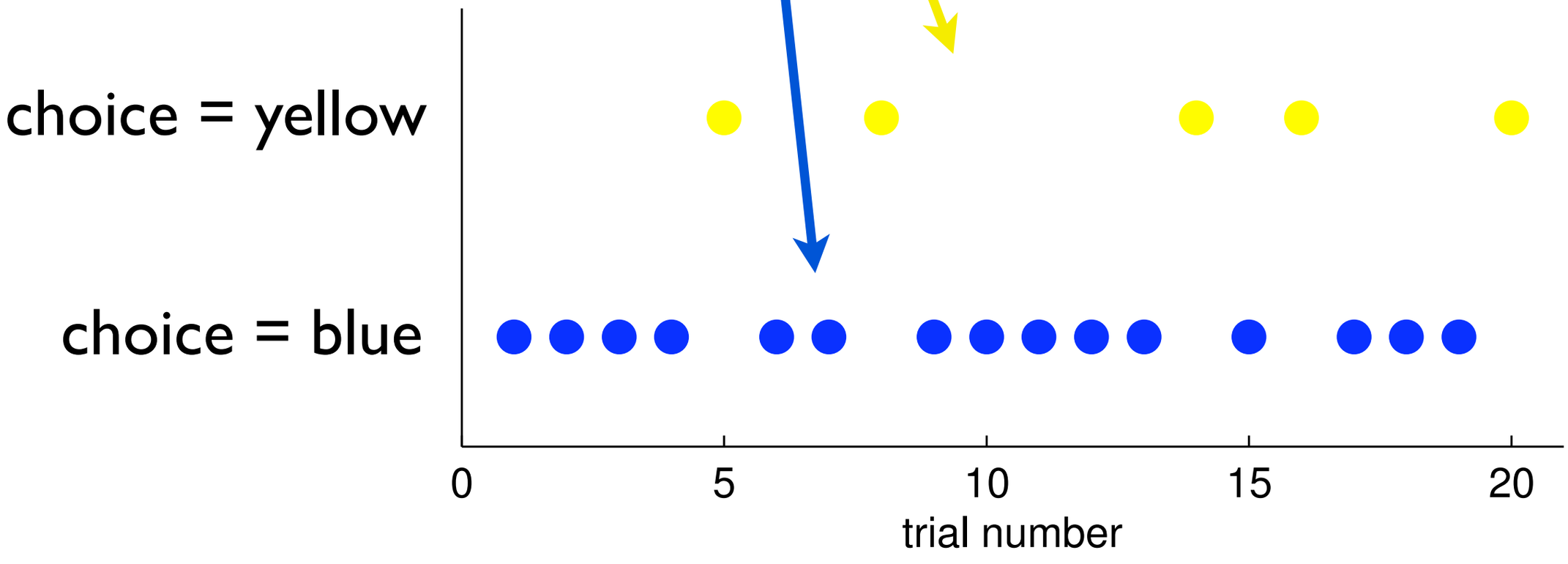
Assume: choices or actions a are taken at random, according to a probabilistic “policy”:



$$p(a = \text{yellow}) = 0.2$$

$$p(a = \text{blue}) = 0.8$$

$$p(a = \text{blue}) + p(a = \text{yellow}) = 1$$



“Optimal Policy”: The greedy bee

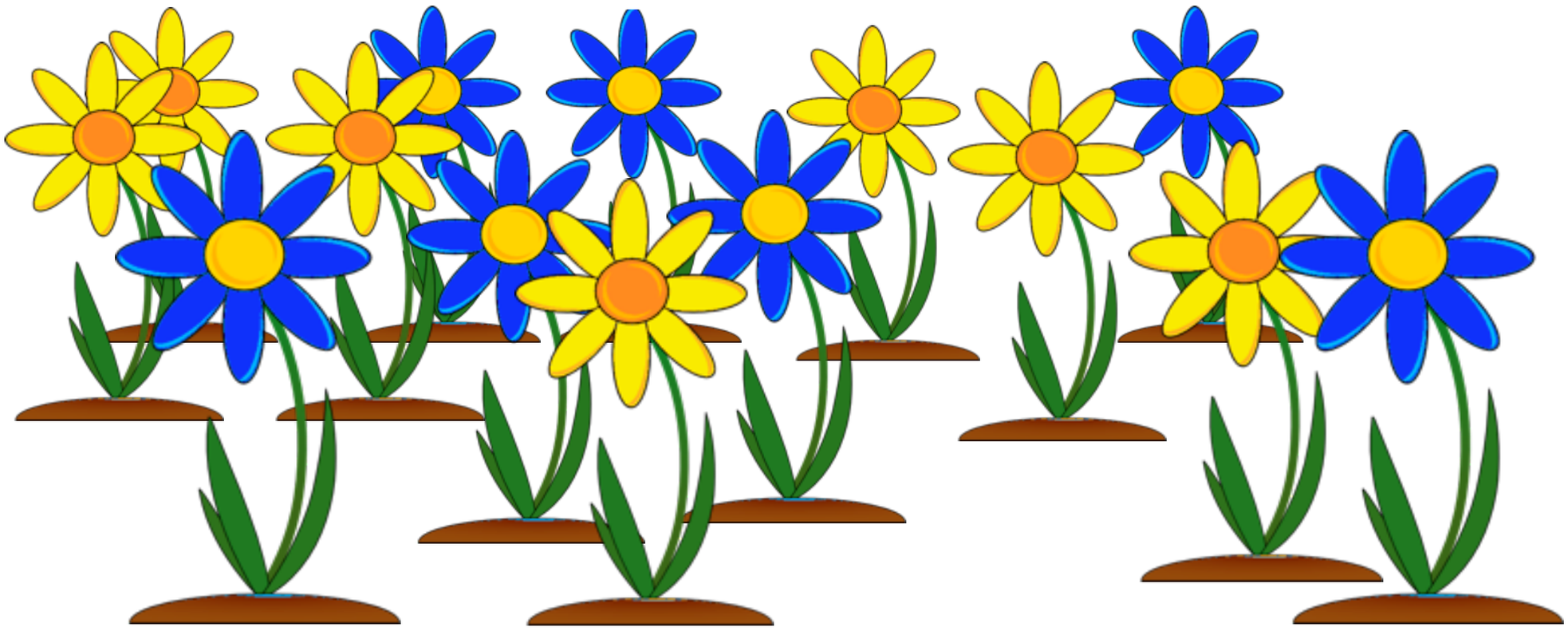
Optimal policy:

$$p(a = \text{blue}) = 1$$
$$p(a = \text{yellow}) = 0$$

Rewards:

$r_b = 8$

$r_y = 2$



“Optimal Policy”: The greedy bee

Optimal policy:

$$p(a = \text{blue}) = 1$$
$$p(a = \text{yellow}) = 0$$

Rewards:

$r_b = 8$

$r_y = 2$

BUT: What happens if the environment changes?

Day	1	2	3	...
r_b	8	2	3	...
r_y	2	8	5	...

Bee needs to explore and exploit

"greedy" policy

$$p(a = \text{blue}) = 1$$
$$p(a = \text{yellow}) = 0$$

BAD IF THINGS CHANGE!

"ε-greedy" policy ($\epsilon \ll 1$)

$$p(a = \text{blue}) = 1 - \epsilon$$
$$p(a = \text{yellow}) = \epsilon$$

exploratory trials

Bee needs to explore and exploit

"greedy" policy

$$p(a = \text{blue}) = 1$$
$$p(a = \text{yellow}) = 0$$

BAD IF THINGS CHANGE!

"ε−greedy" policy ($\epsilon \ll 1$)

$$p(a = \text{blue}) = 1 - \epsilon$$
$$p(a = \text{yellow}) = \epsilon$$

softmax Gibbs-policy (depends on rewards!)

$$p(a = \text{blue}) = \exp(\beta r_b) / \big(\exp(\beta r_b) + \exp(\beta r_y) \big)$$
$$p(a = \text{yellow}) = \exp(\beta r_y) / \big(\exp(\beta r_b) + \exp(\beta r_y) \big)$$

Softmax-Gibbs Policy

$$p(b) = \frac{\exp(\beta r_b)}{\exp(\beta r_b) + \exp(\beta r_y)}$$

$$p(y) = \frac{\exp(\beta r_y)}{\exp(\beta r_b) + \exp(\beta r_y)}$$

short-hand notation: we write $p(b)$ for $p(a = \text{blue})$ etc.

Softmax-Gibbs Policy

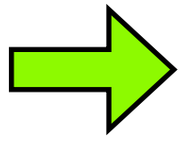
$$p(b) = \frac{\exp(\beta r_b)}{\exp(\beta r_b) + \exp(\beta r_y)}$$

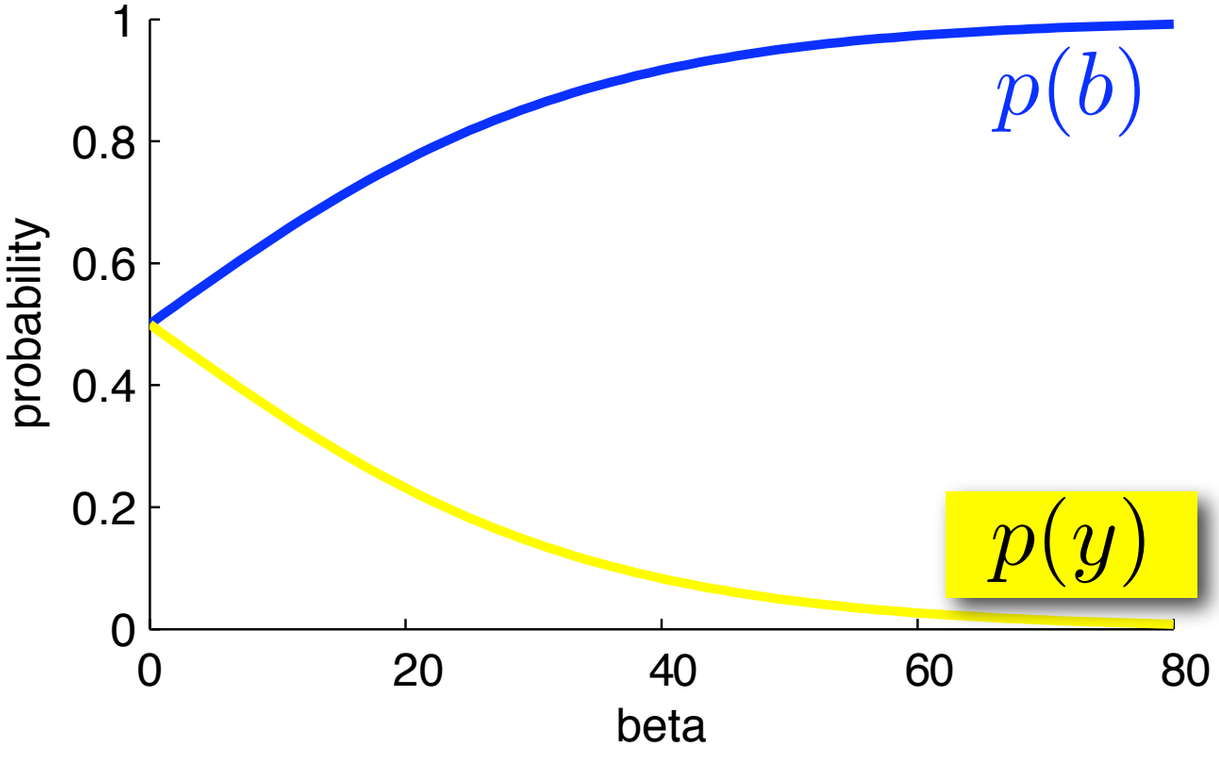
$$p(y) = \frac{\exp(\beta r_y)}{\exp(\beta r_b) + \exp(\beta r_y)}$$

Rewards:

$$r_b = 8$$

$$r_y = 2$$





Softmax-Gibbs Policy

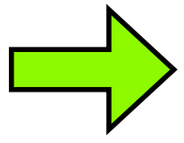
$$p(b) = \frac{\exp(\beta r_b)}{\exp(\beta r_b) + \exp(\beta r_y)}$$

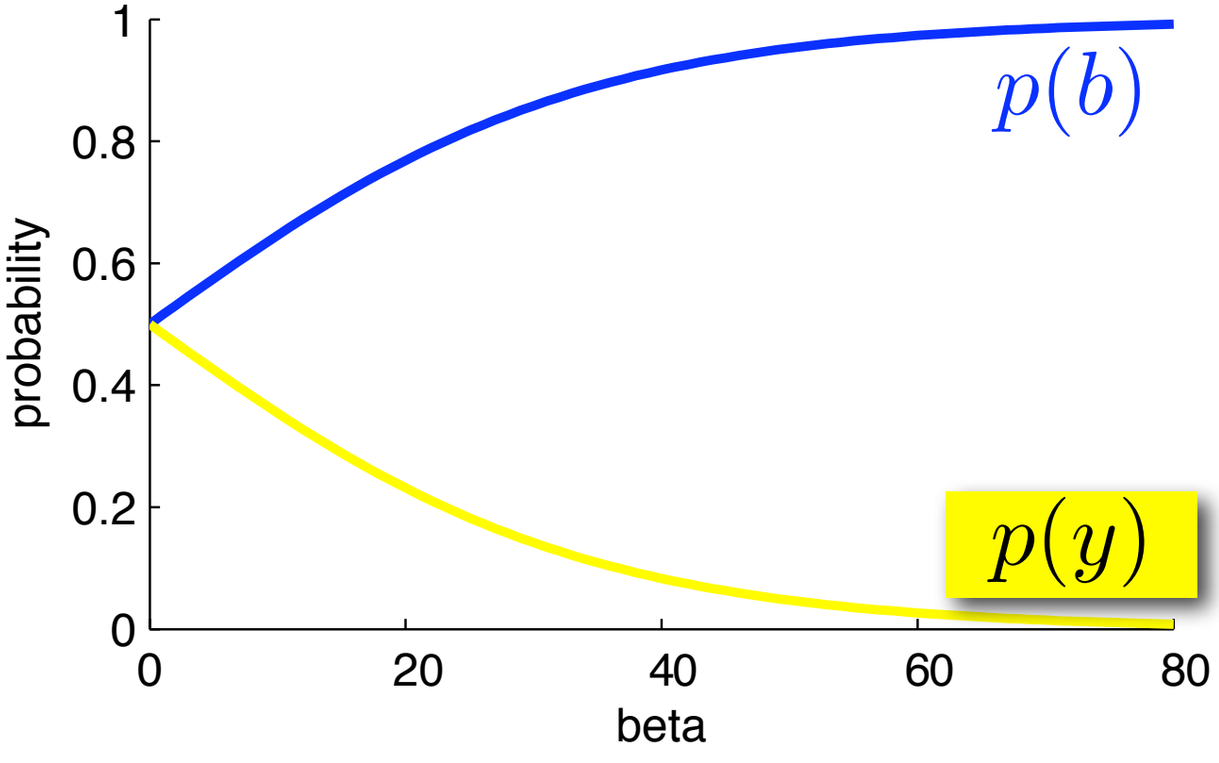
$$p(y) = \frac{\exp(\beta r_y)}{\exp(\beta r_b) + \exp(\beta r_y)}$$

Rewards:

$$r_b = 8$$

$$r_y = 2$$





← exploration

exploitation →

What does the bee know?

	actual reward	internal estimate
	r_b	m_b
	r_y	m_y

How can the bee learn the rewards?

"greedy" update:

$m_b = r_{b,i}$

last reward on a blue flower

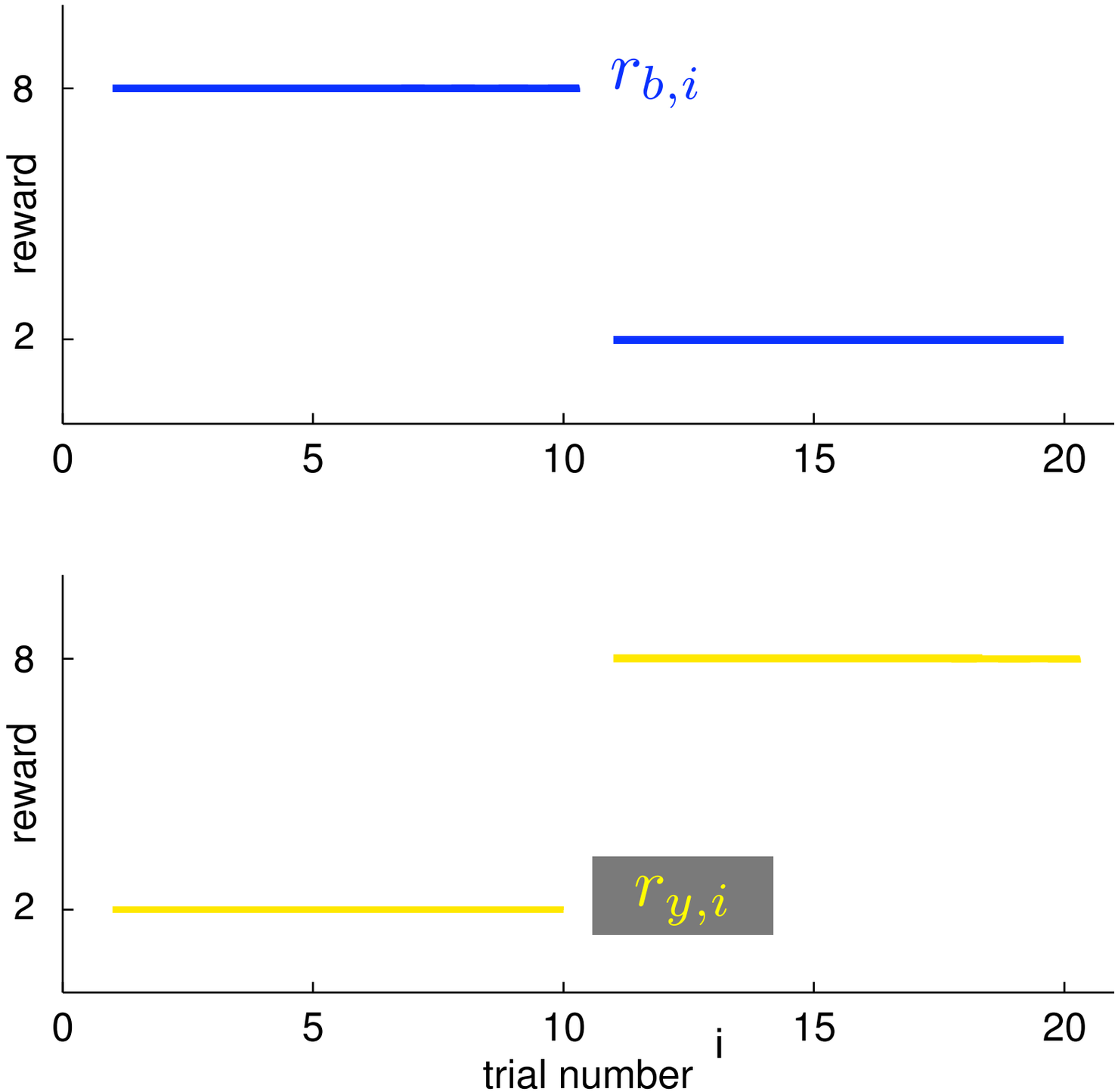
$m_y = r_{y,i}$

last reward on a yellow flower

How can the bee learn the rewards?

"greedy" update:

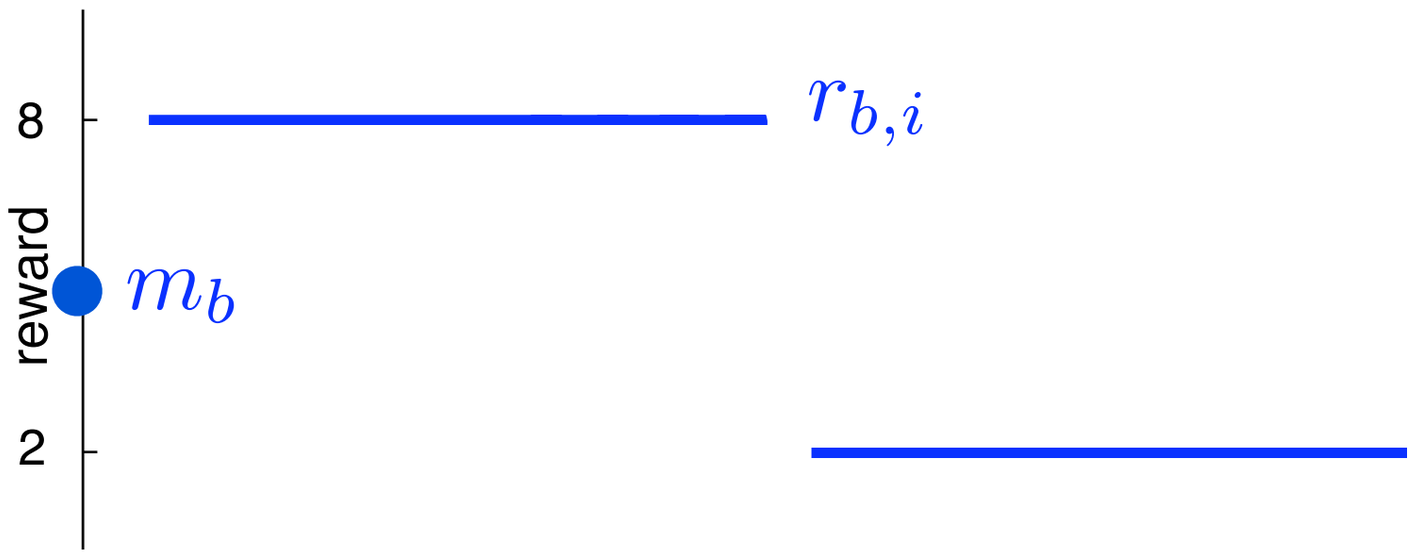
$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



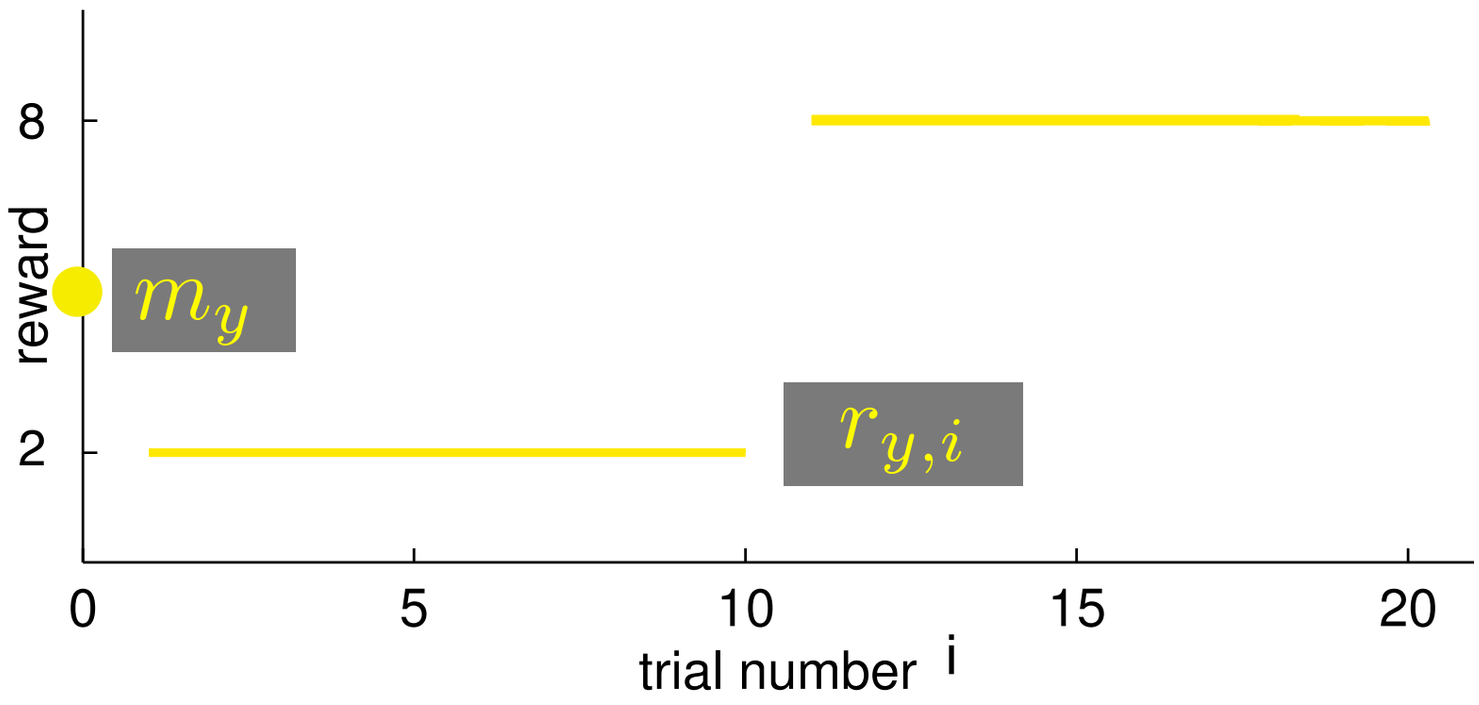
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



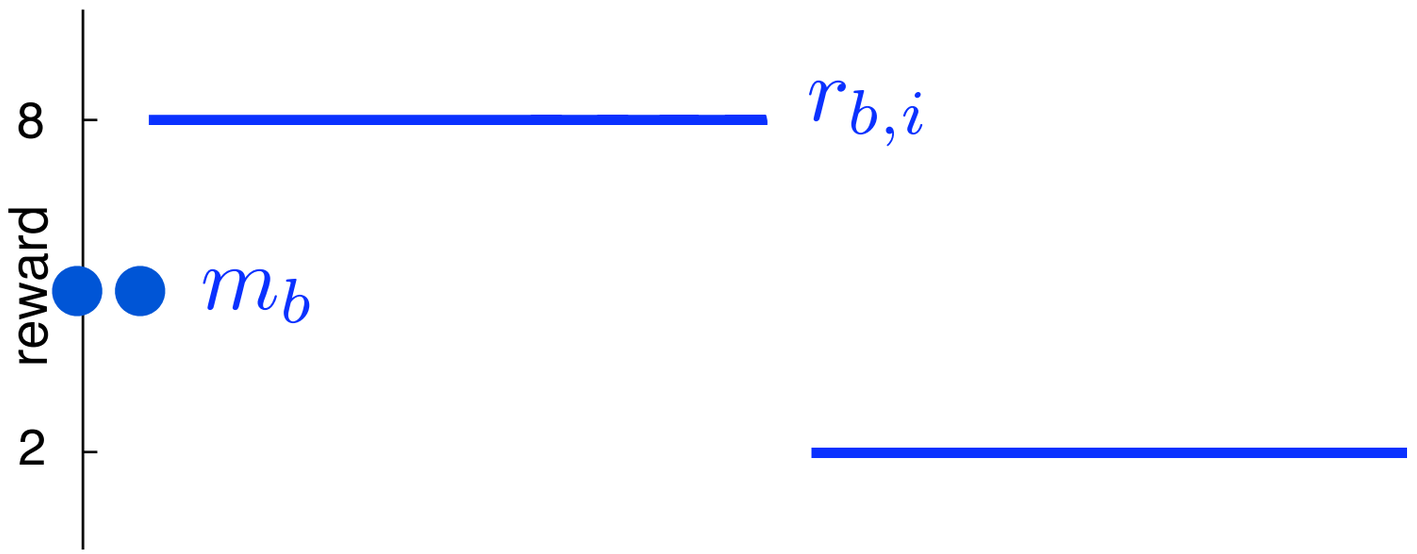
choices: ◆



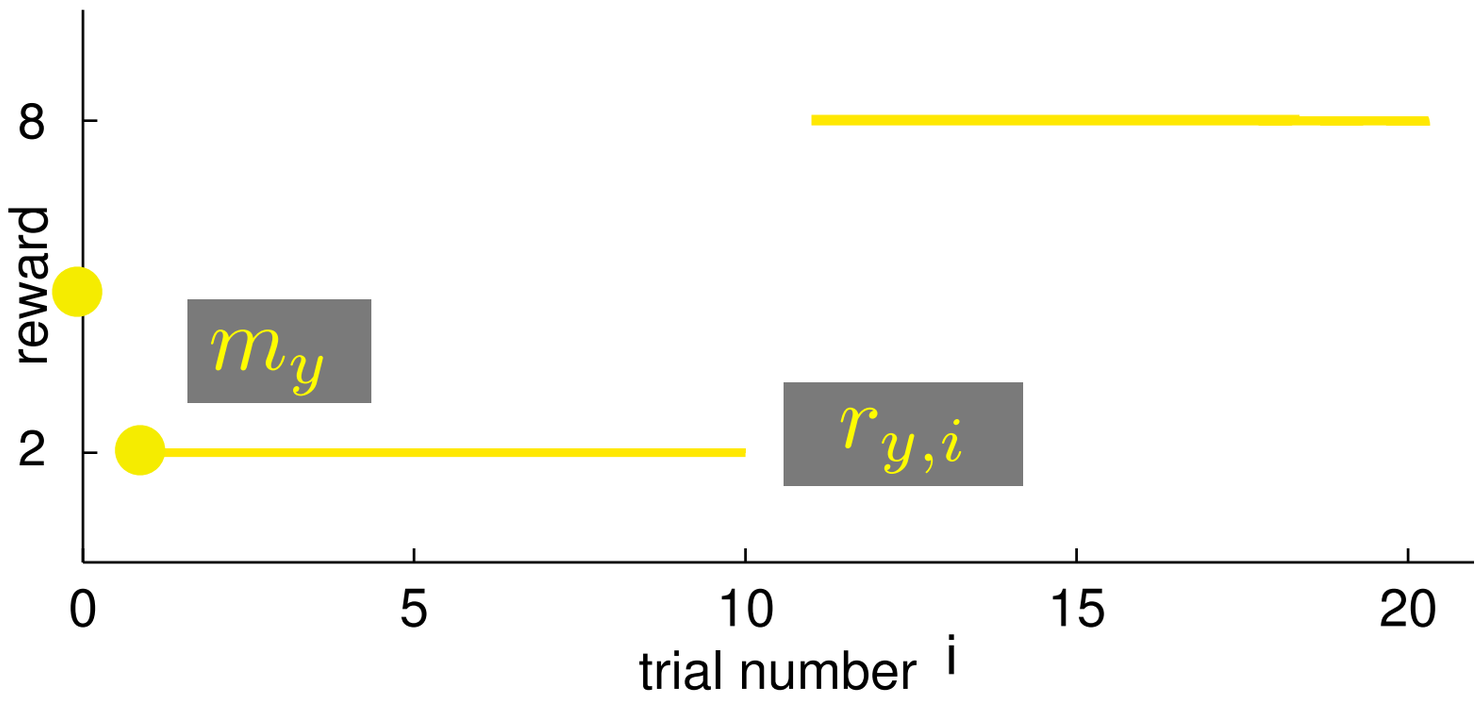
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



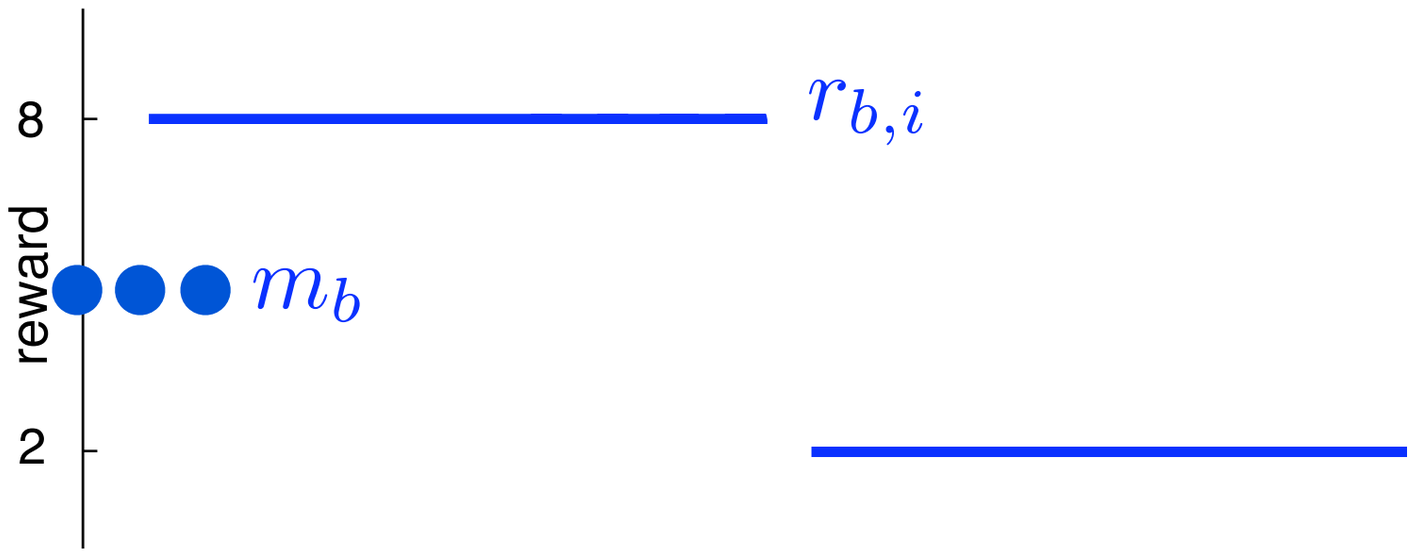
choices:



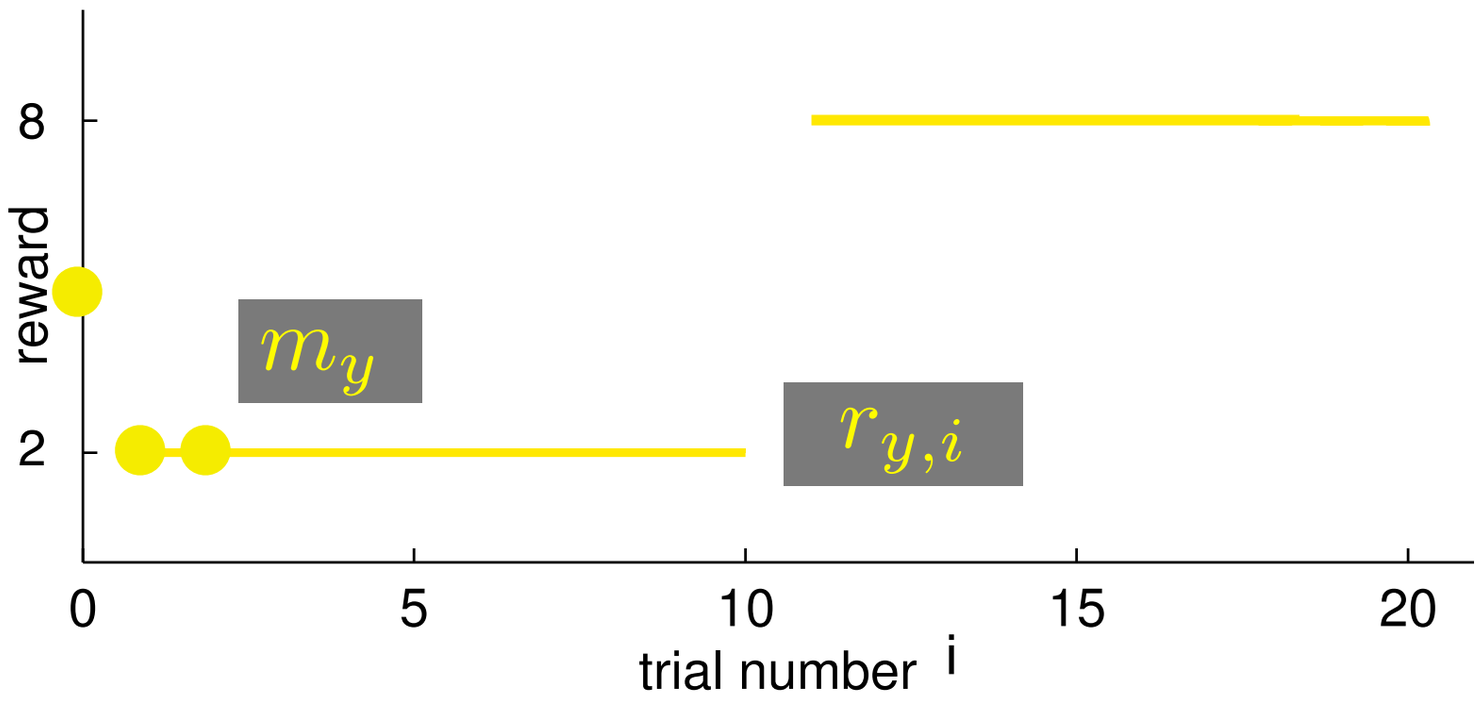
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



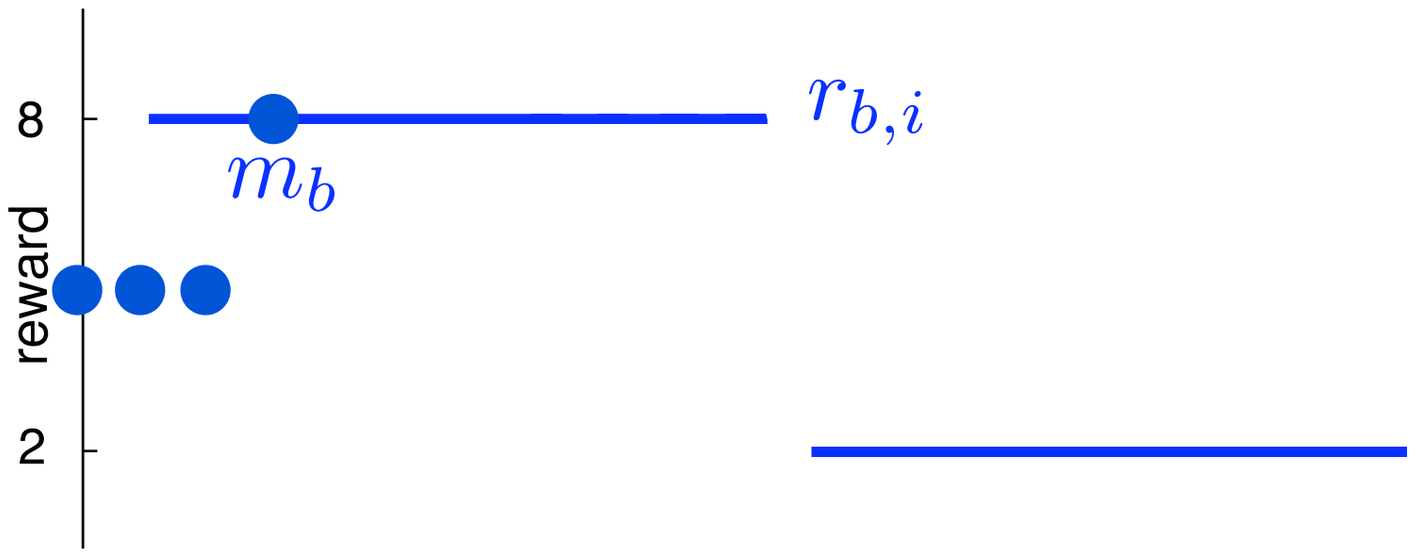
choices:



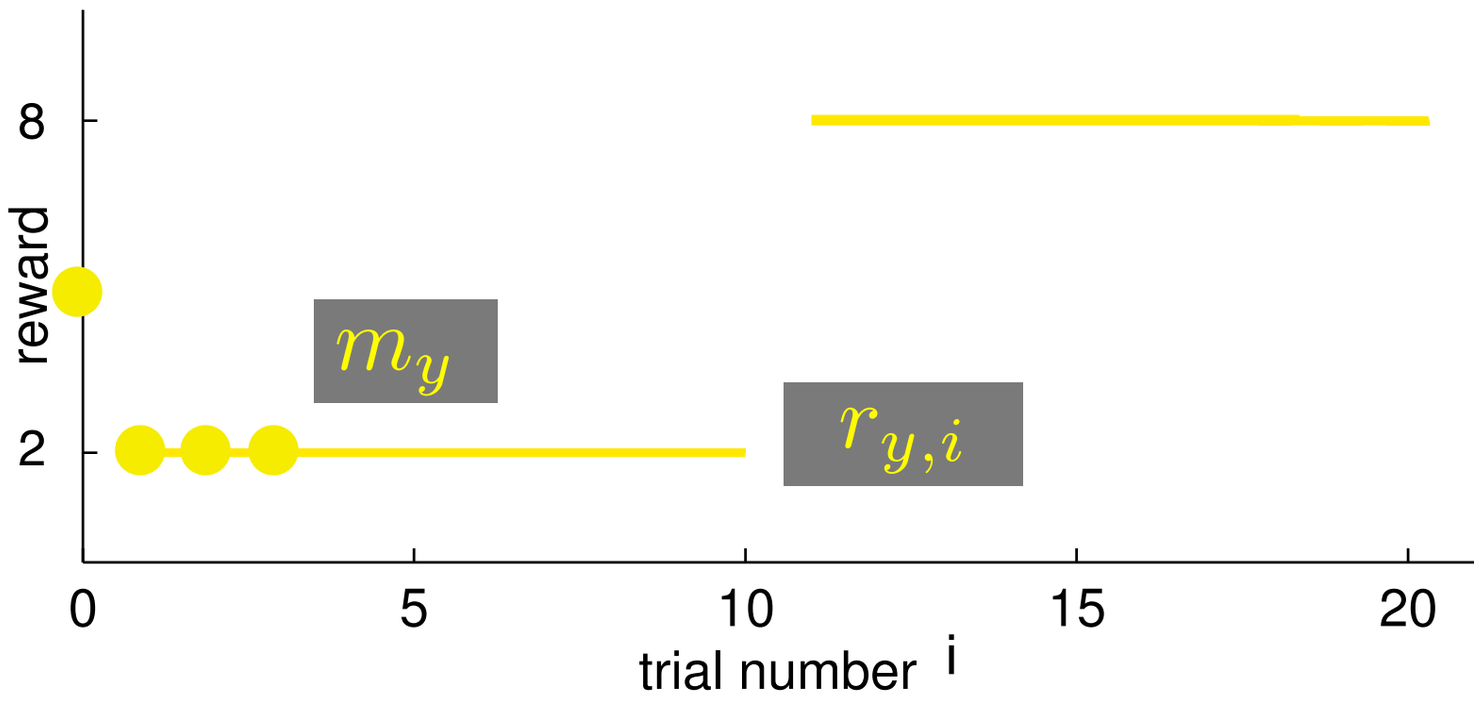
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



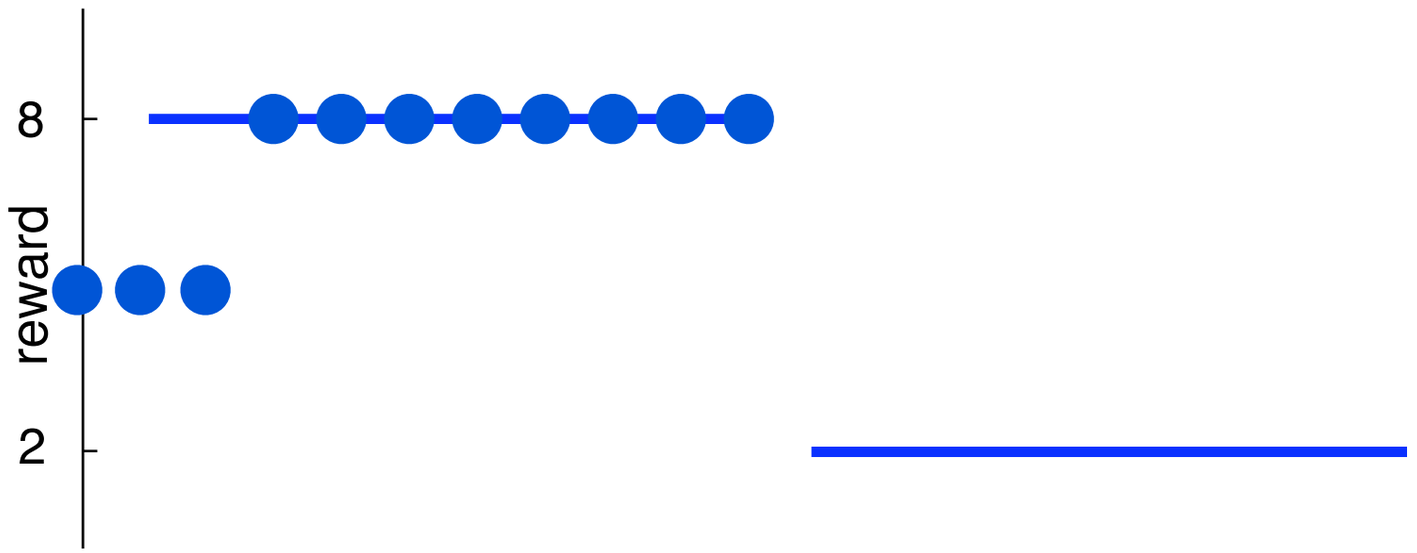
choices:



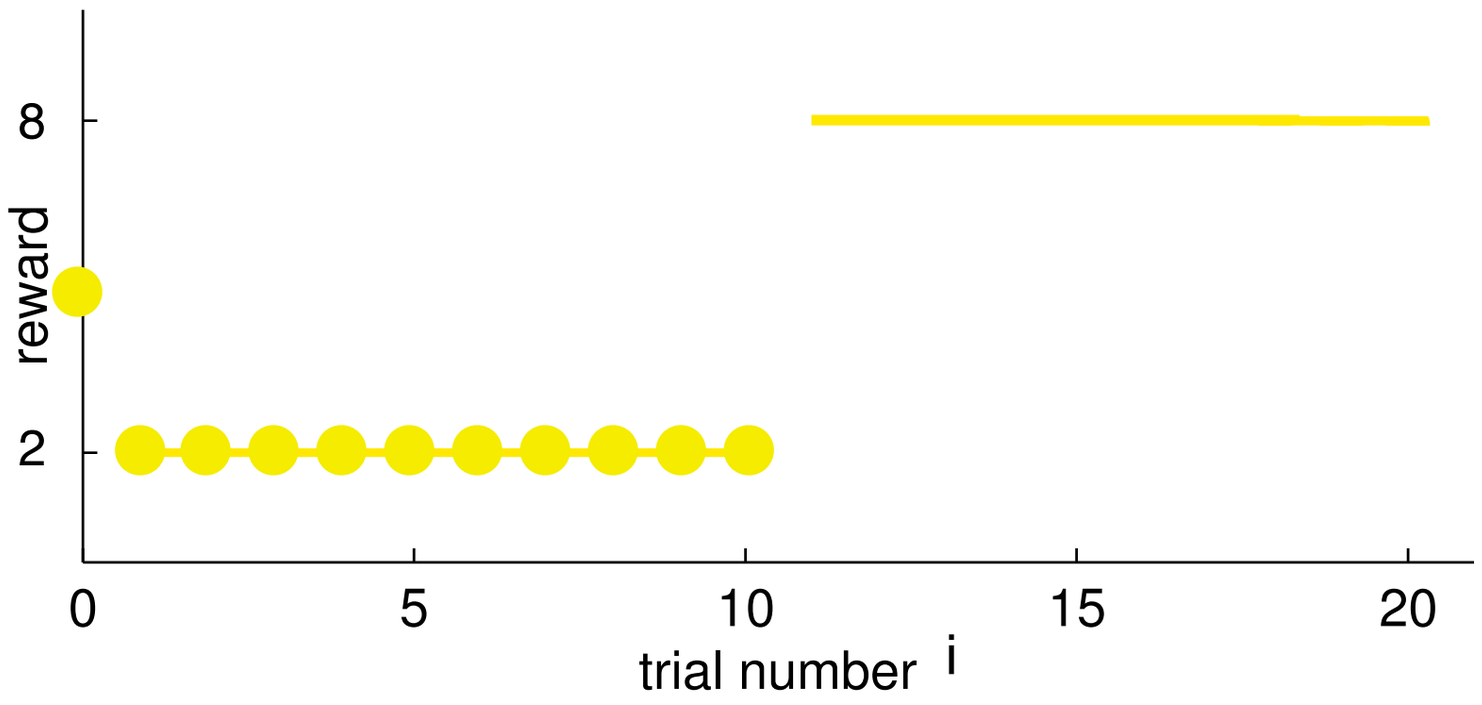
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



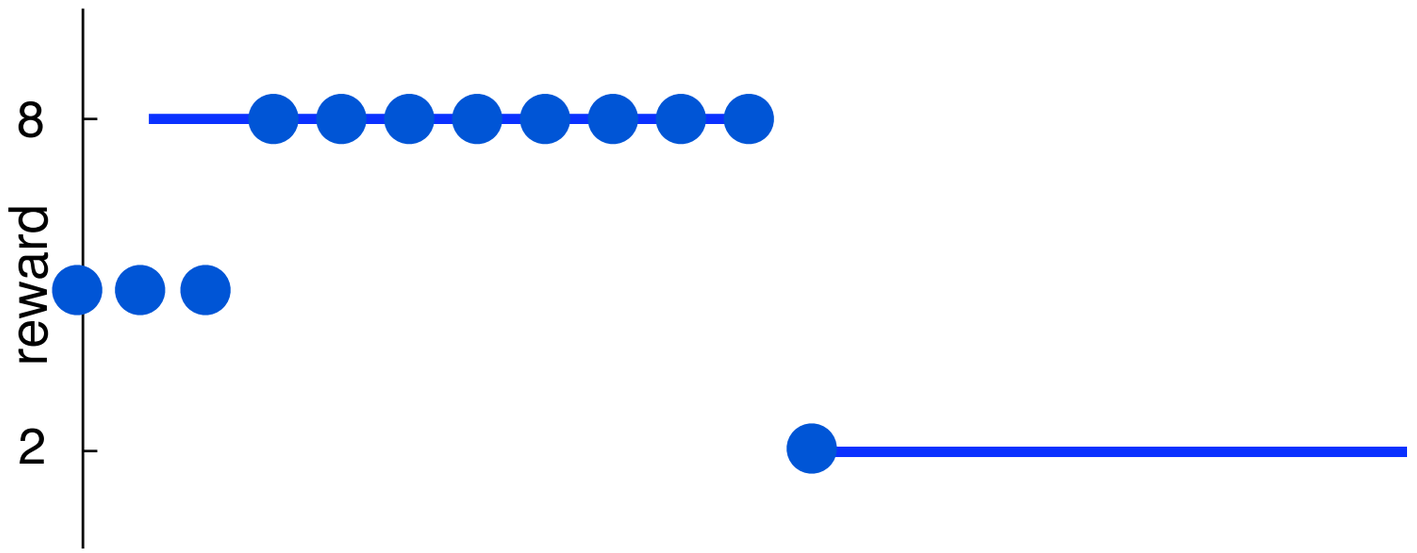
choices:



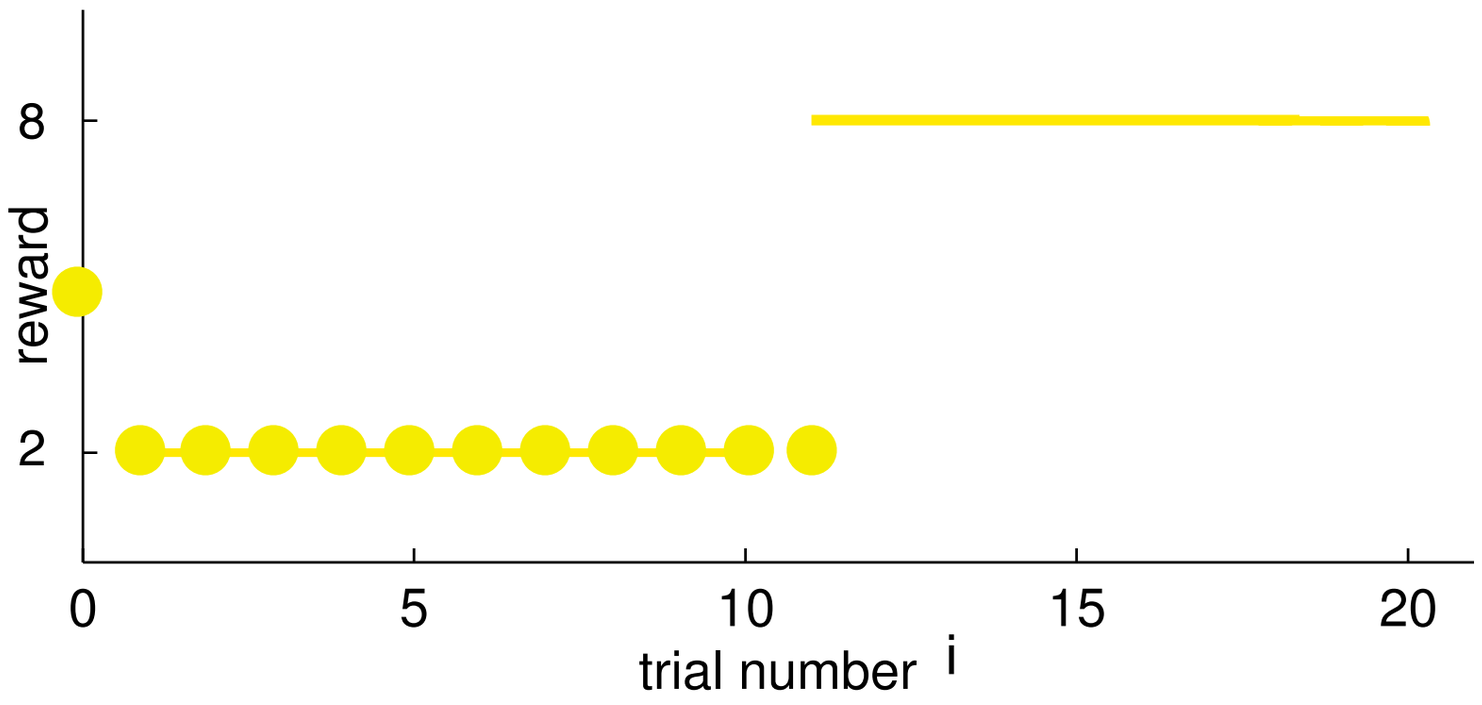
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



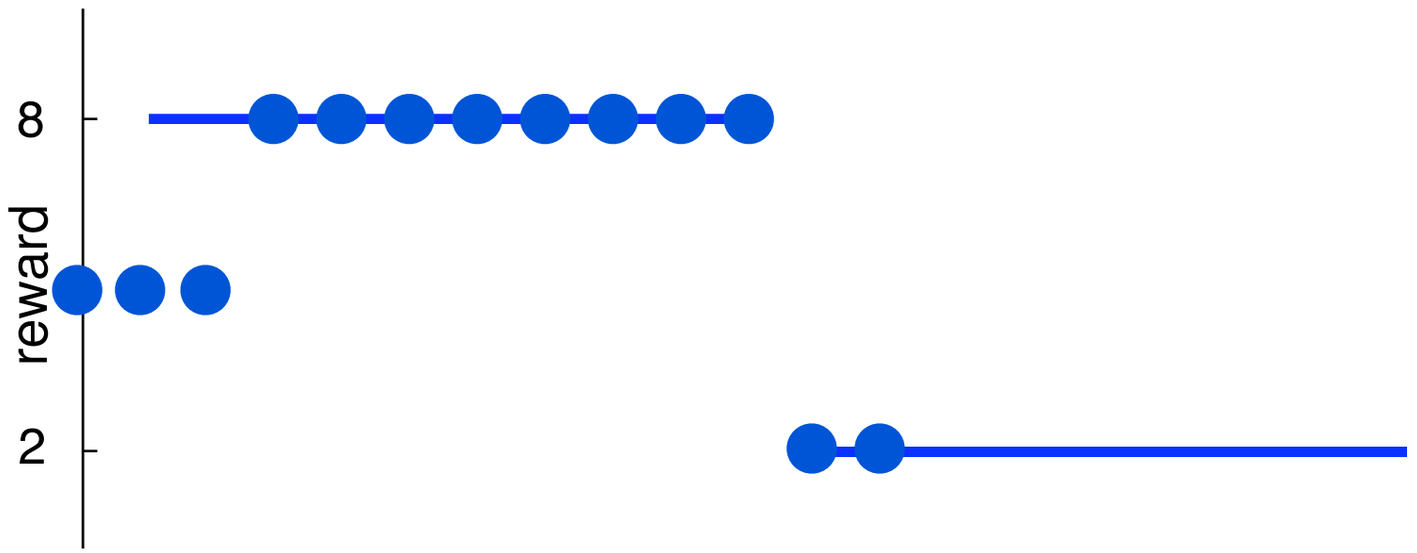
choices:



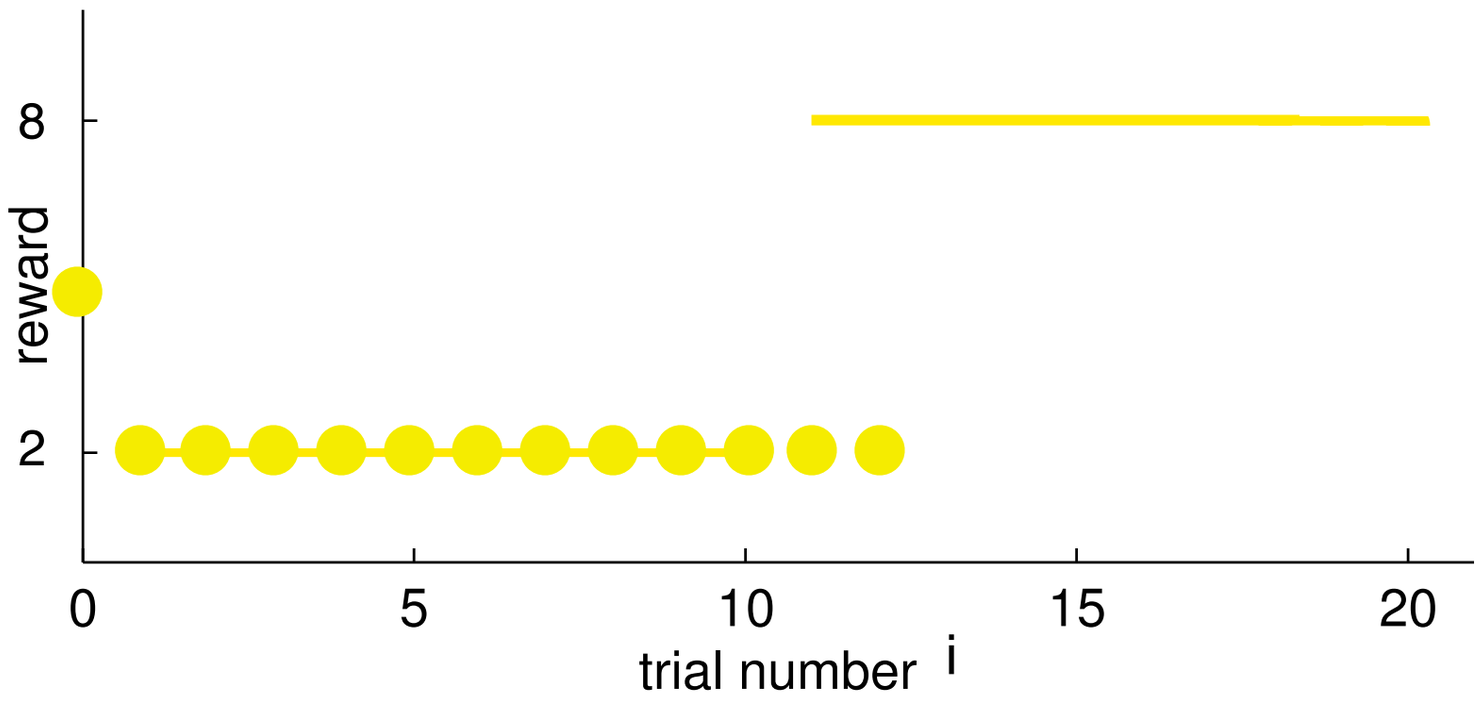
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



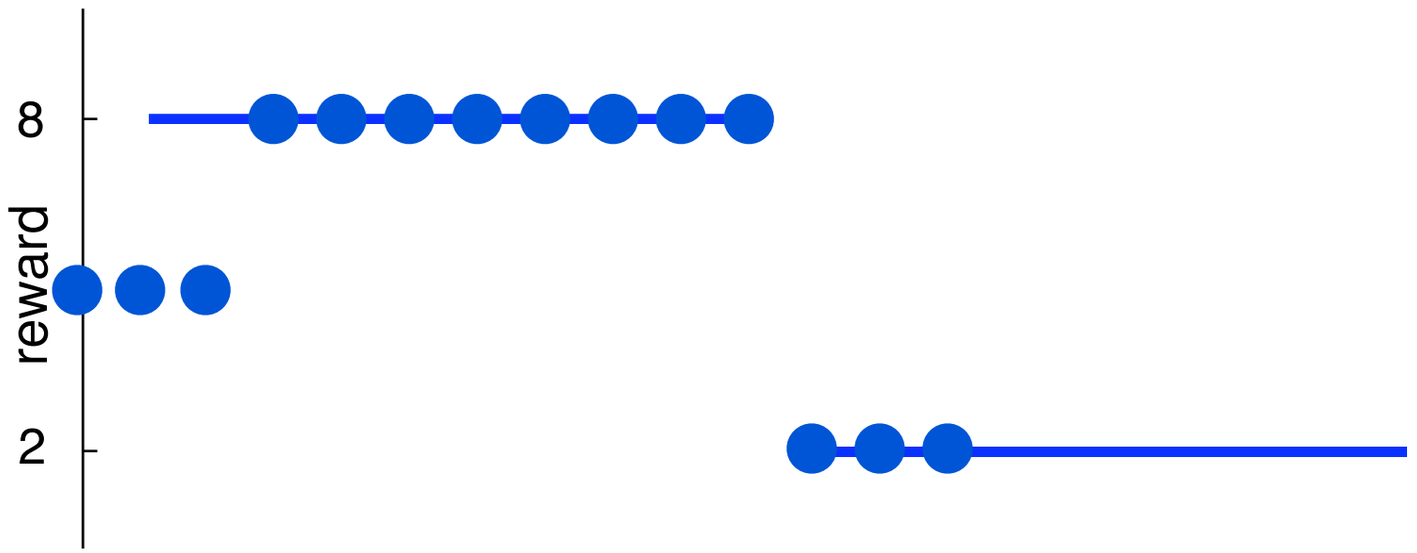
choices:



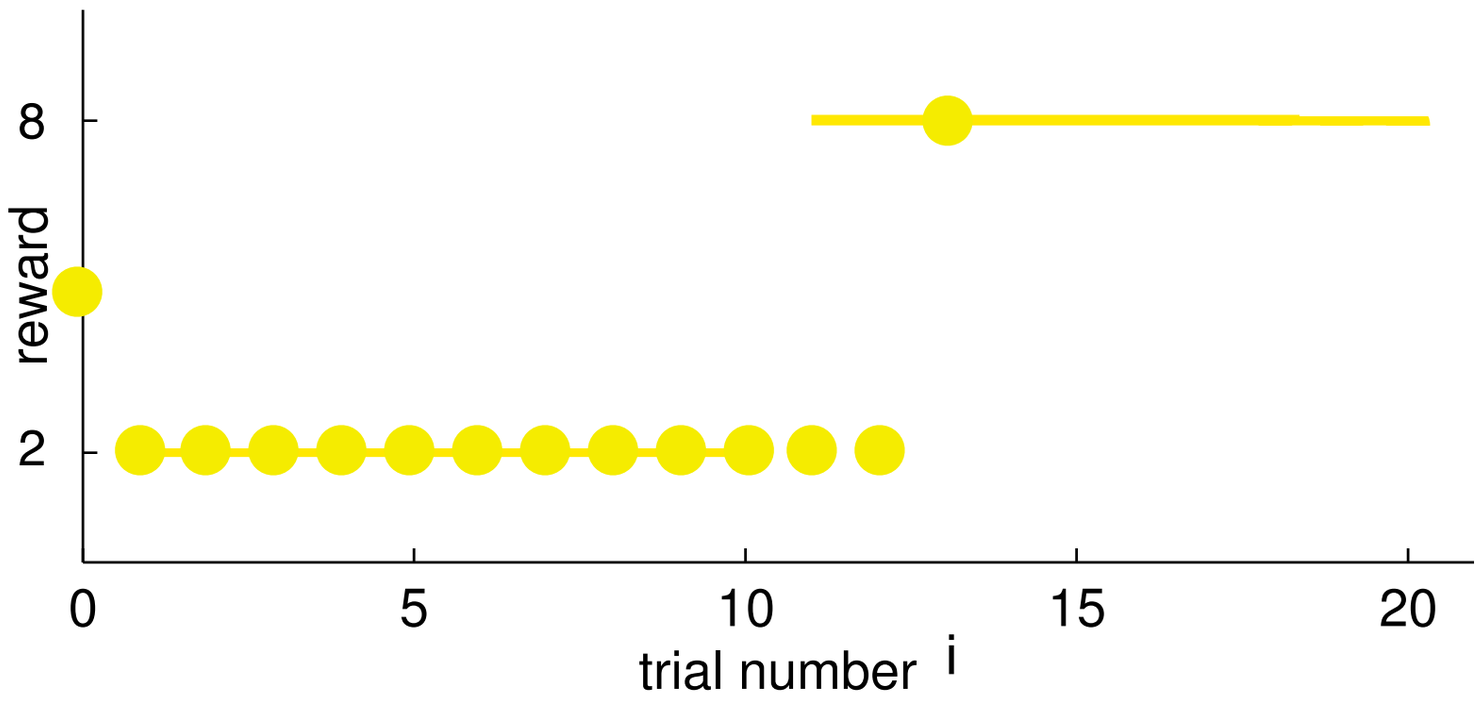
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



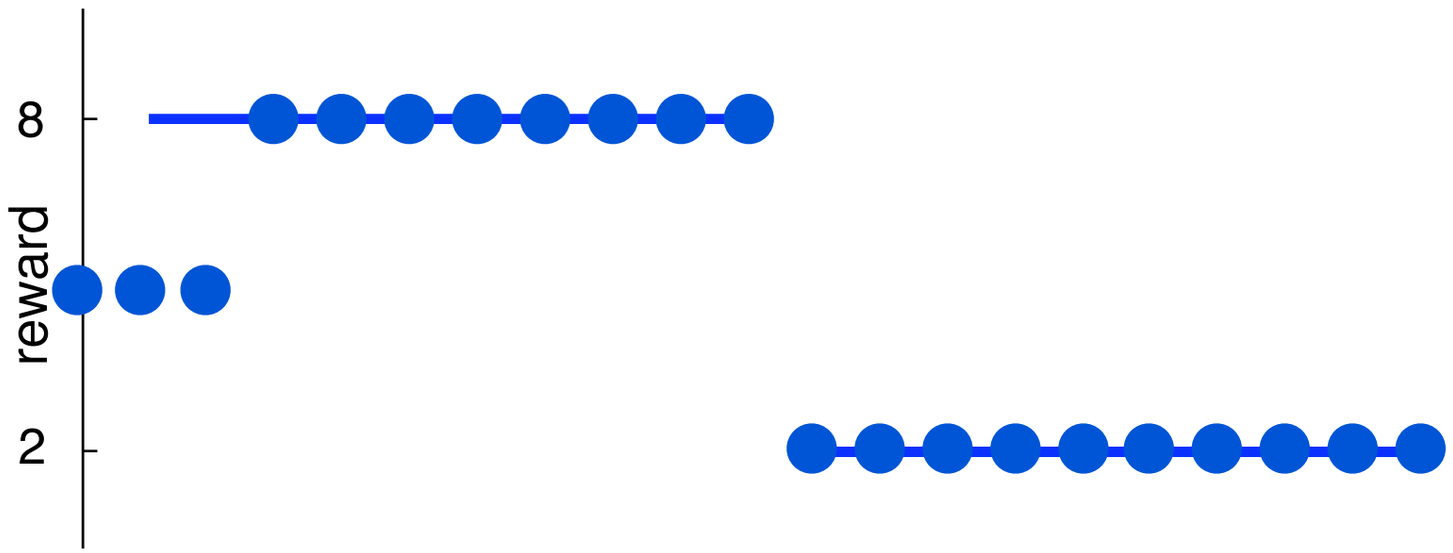
choices:



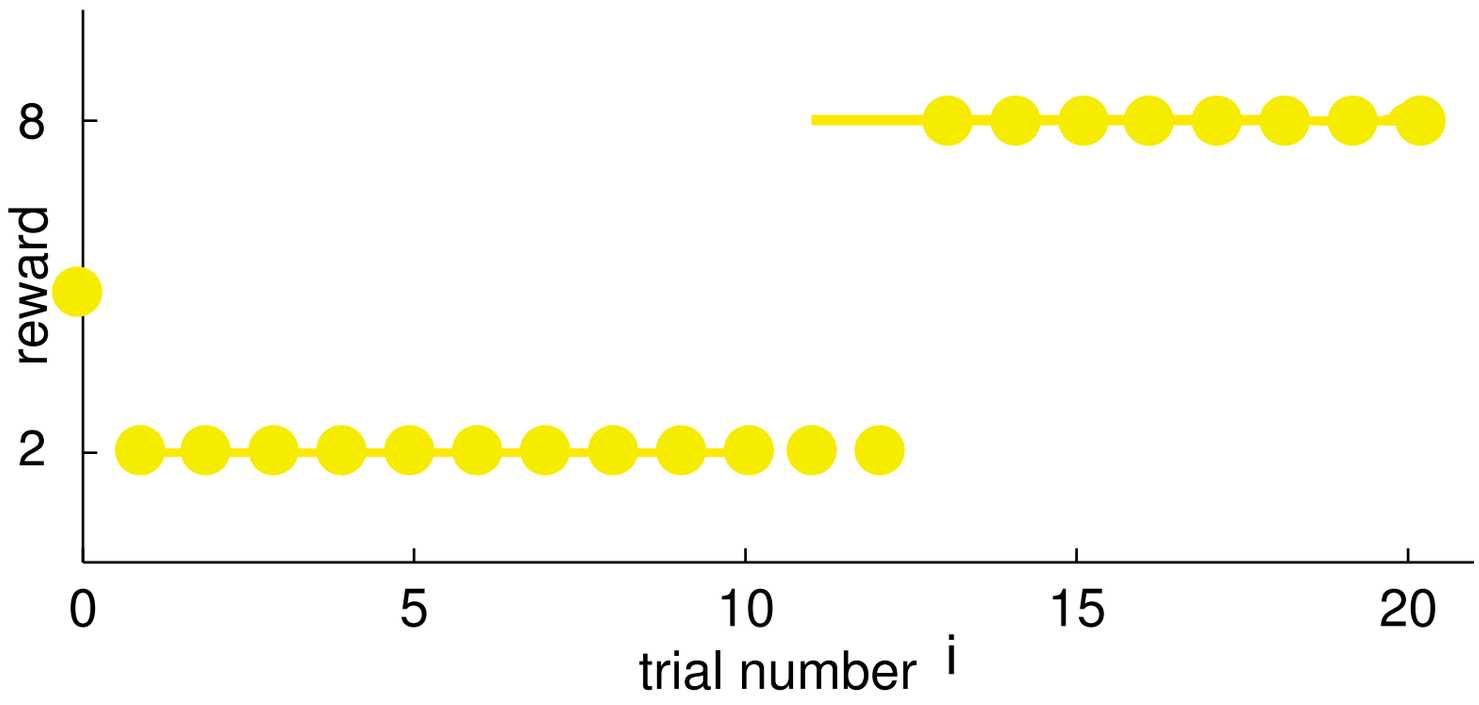
How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i}$$
$$m_y = r_{y,i}$$



choices:



How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i} \qquad m_y = r_{y,i}$$

"batch" update:

$$m_b = \frac{1}{N}\sum_{i=1}^{N} r_{b,i} \qquad m_y = \frac{1}{N}\sum_{i=1}^{N} r_{y,i}$$

average reward on last N visits to a blue flower

average reward on last N visits to a yellow flower

How can the bee learn the rewards?

"greedy" update:

$$m_b = r_{b,i} \qquad m_y = r_{y,i}$$

"batch" update:

$$m_b = \frac{1}{N}\sum_{i=1}^{N} r_{b,i} \qquad m_y = \frac{1}{N}\sum_{i=1}^{N} r_{y,i}$$

"online" update:

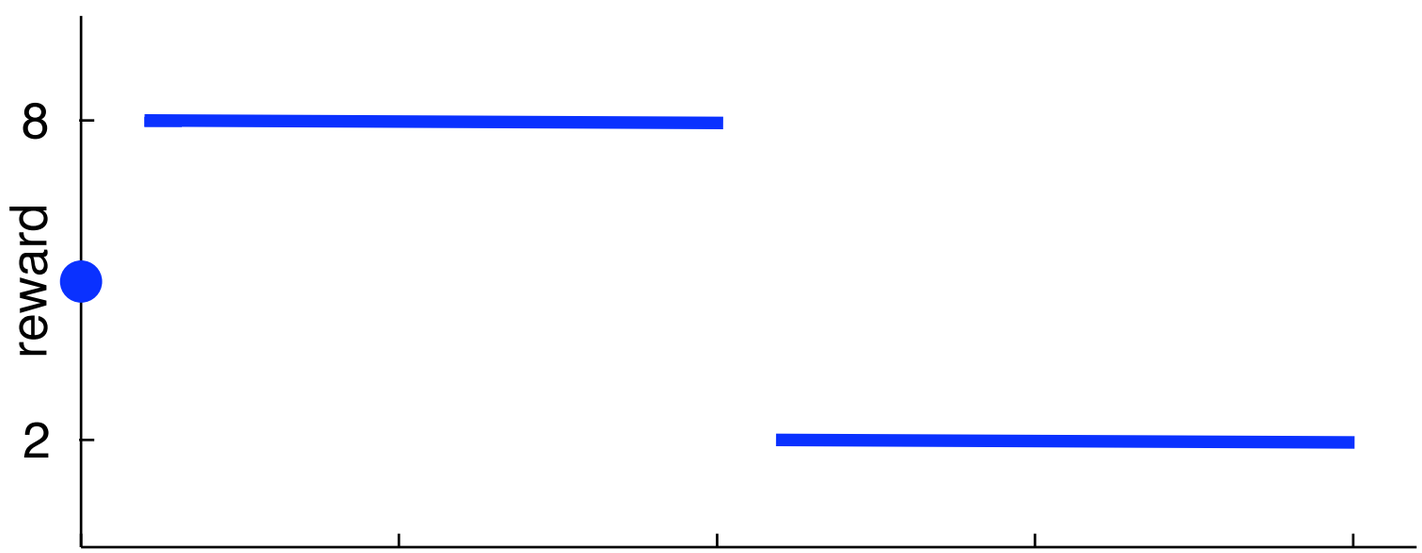
$$m_b \rightarrow m_b + \epsilon(r_{b,i} - m_b)$$

⇨ **"delta"- rule** $m_b \rightarrow m_b + \underbrace{\epsilon\delta}_{\text{learning rate}}$ **with** $\delta = \underbrace{r_{b,i} - m_b}_{\text{prediction error}}$

How can the bee learn the rewards?

"online" update:

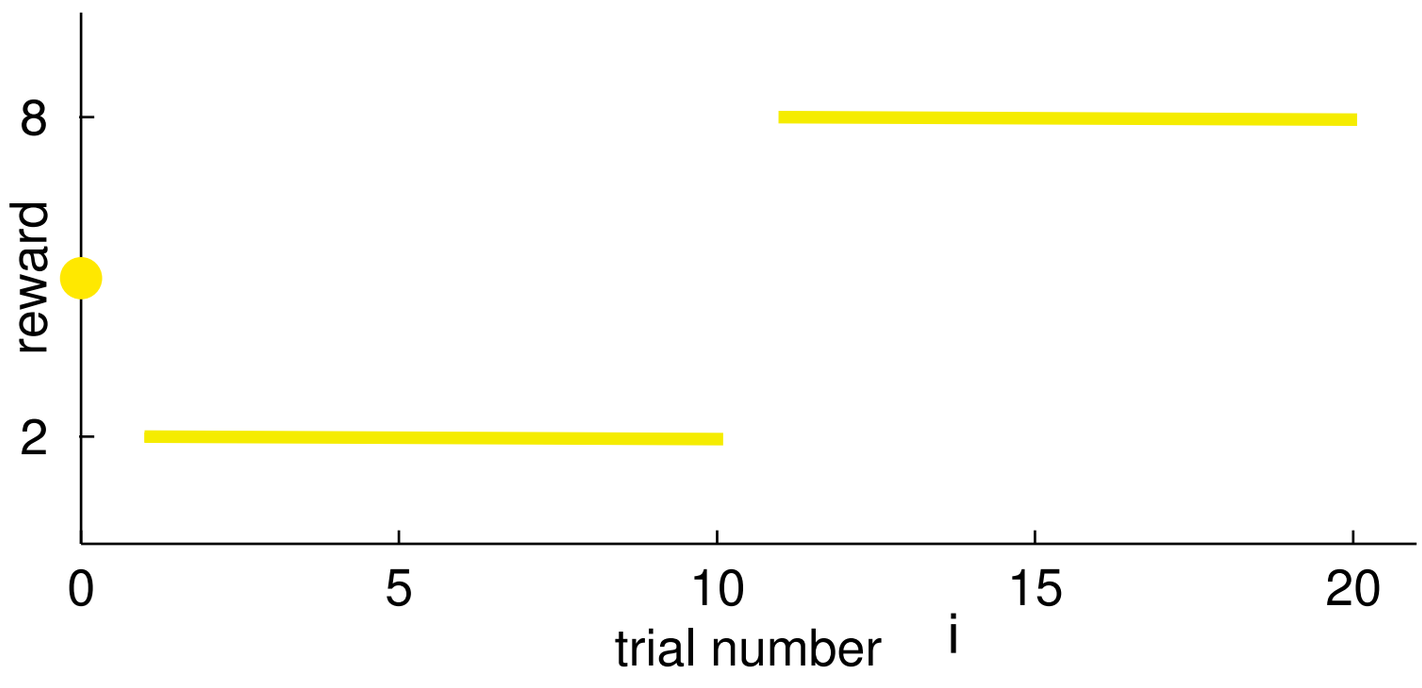
$$m_b \longrightarrow m_b + \epsilon\delta_b$$
$$m_y \longrightarrow m_y + \epsilon\delta_y$$



choices: ◆

prediction error:

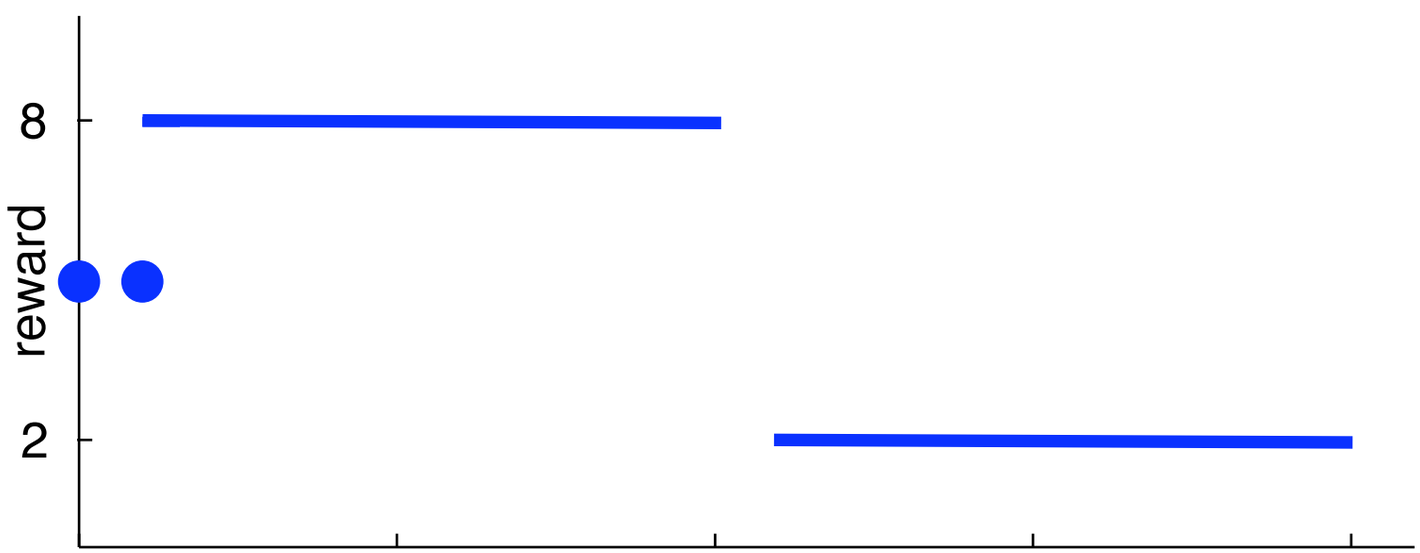
$$\delta_b = r_{b,i} - m_b$$
$$\delta_y = r_{y,i} - m_y$$



How can the bee learn the rewards?

"online" update:

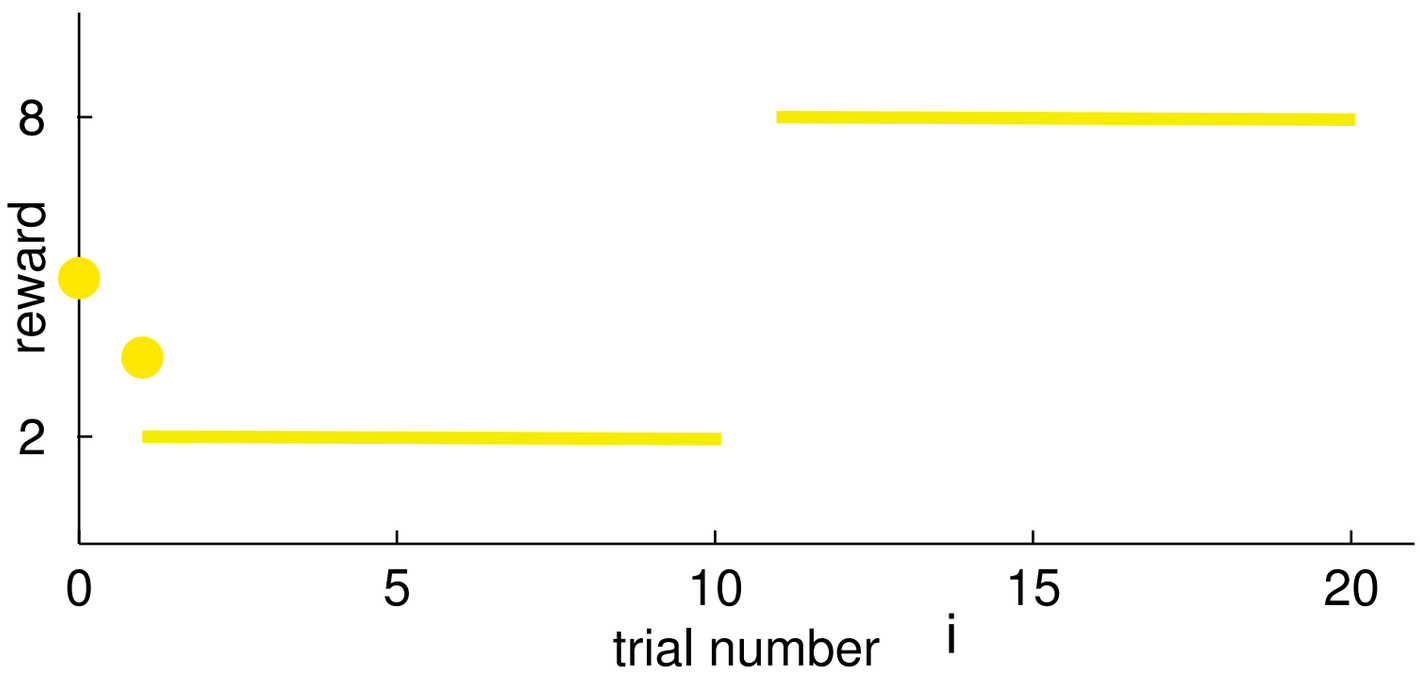
$$m_b \longrightarrow m_b + \epsilon\delta_b$$
$$m_y \longrightarrow m_y + \epsilon\delta_y$$



choices: ◆

prediction error:

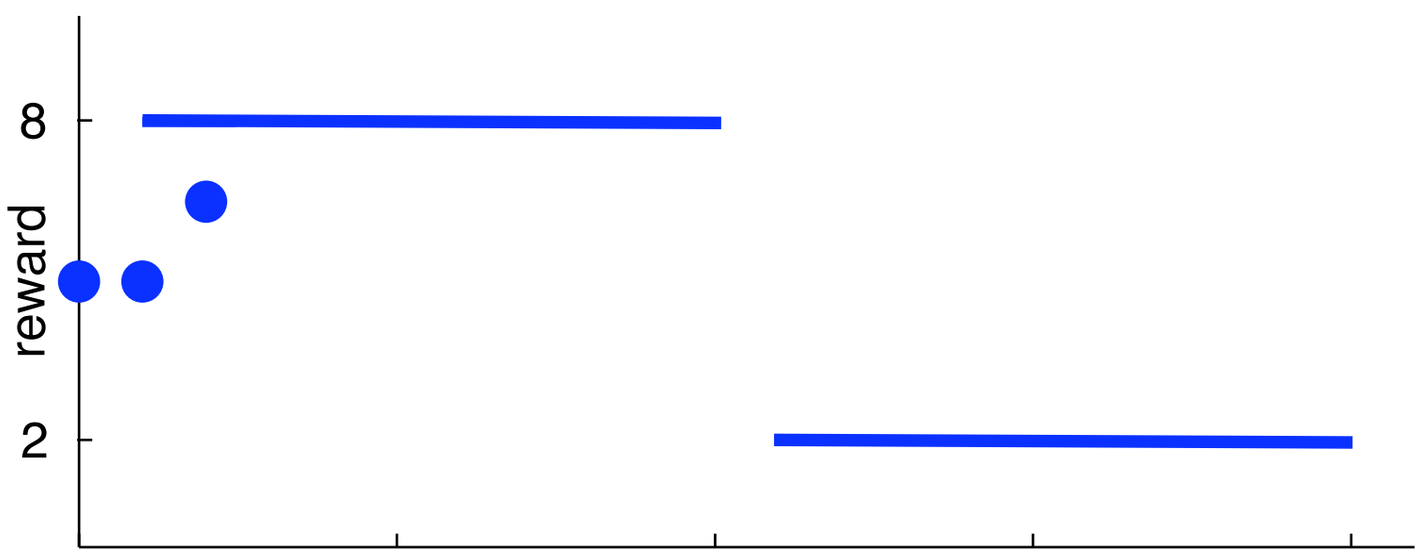
$$\delta_b = r_{b,i} - m_b$$
$$\delta_y = r_{y,i} - m_y$$



How can the bee learn the rewards?

"online" update:

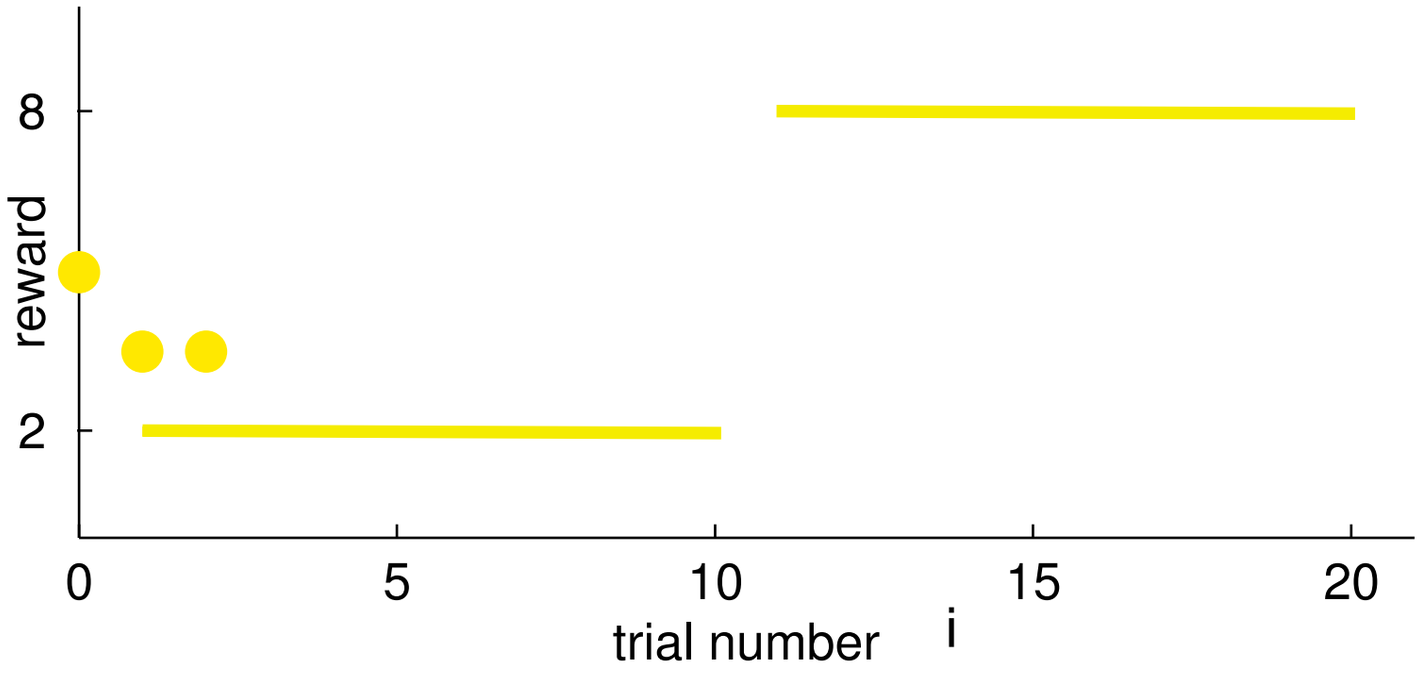
$$m_b \longrightarrow m_b + \epsilon\delta_b$$
$$m_y \longrightarrow m_y + \epsilon\delta_y$$



choices:

prediction error:

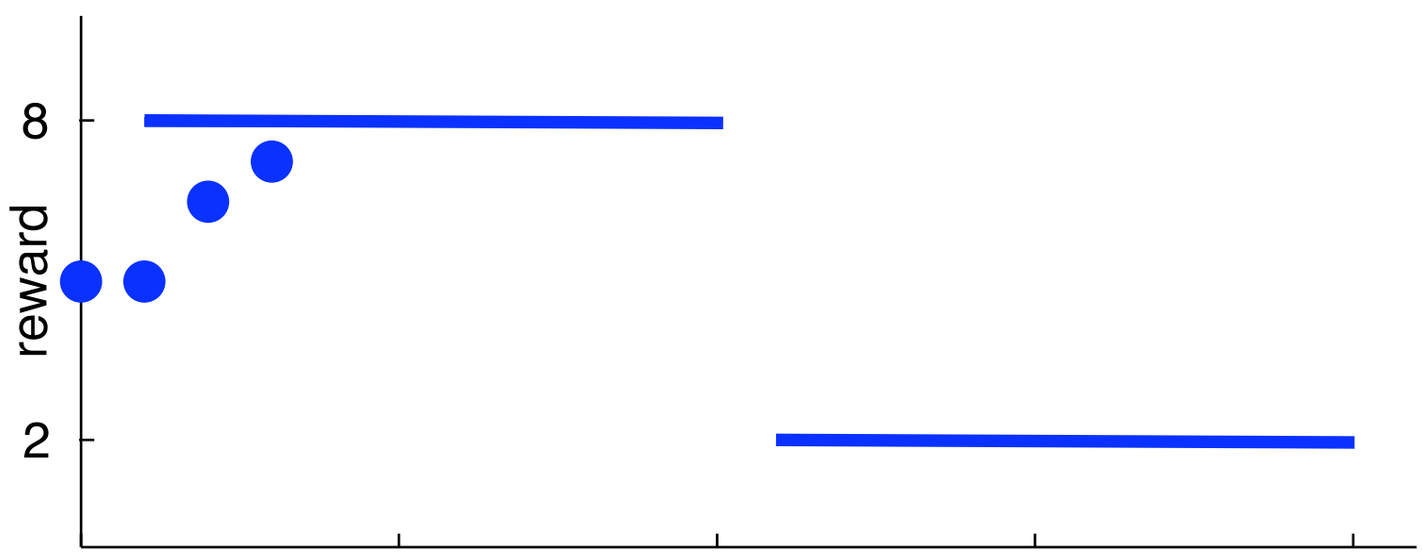
$$\delta_b = r_{b,i} - m_b$$
$$\delta_y = r_{y,i} - m_y$$



How can the bee learn the rewards?

"online" update:

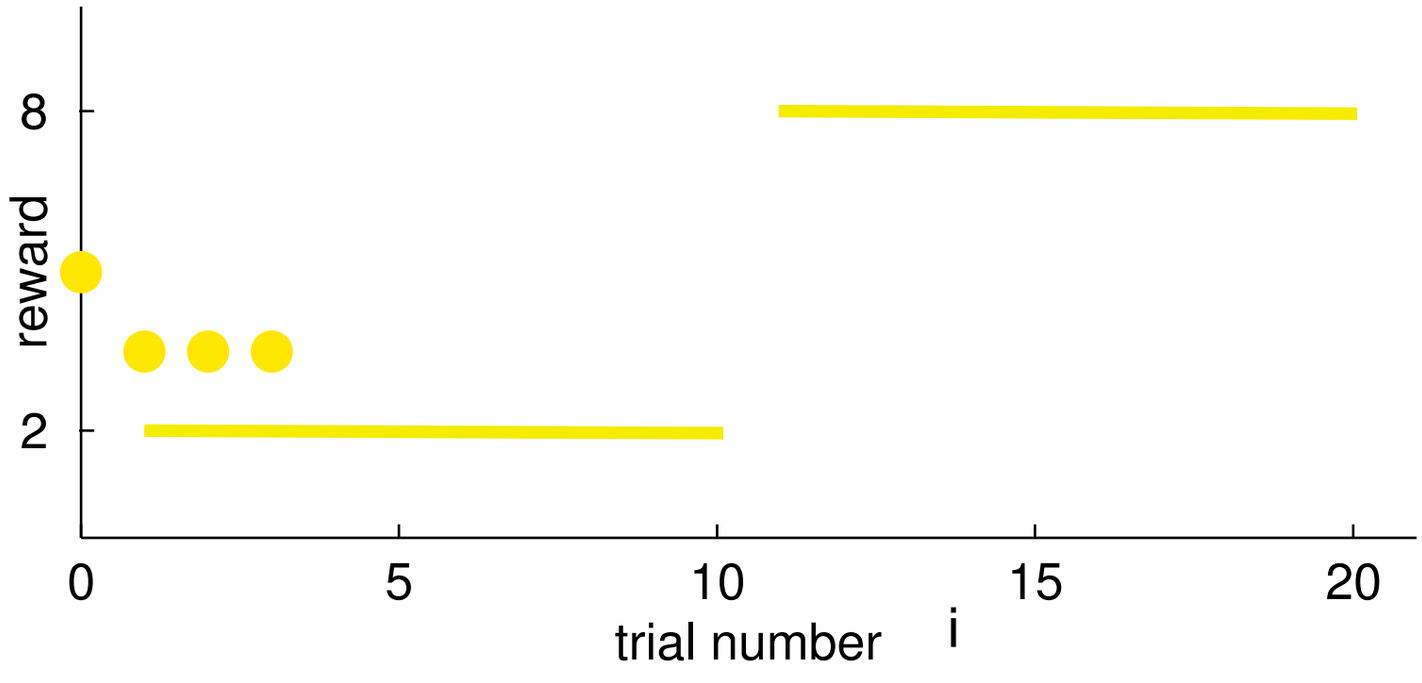
$$m_b \longrightarrow m_b + \epsilon\delta_b$$
$$m_y \longrightarrow m_y + \epsilon\delta_y$$



choices:

prediction error:

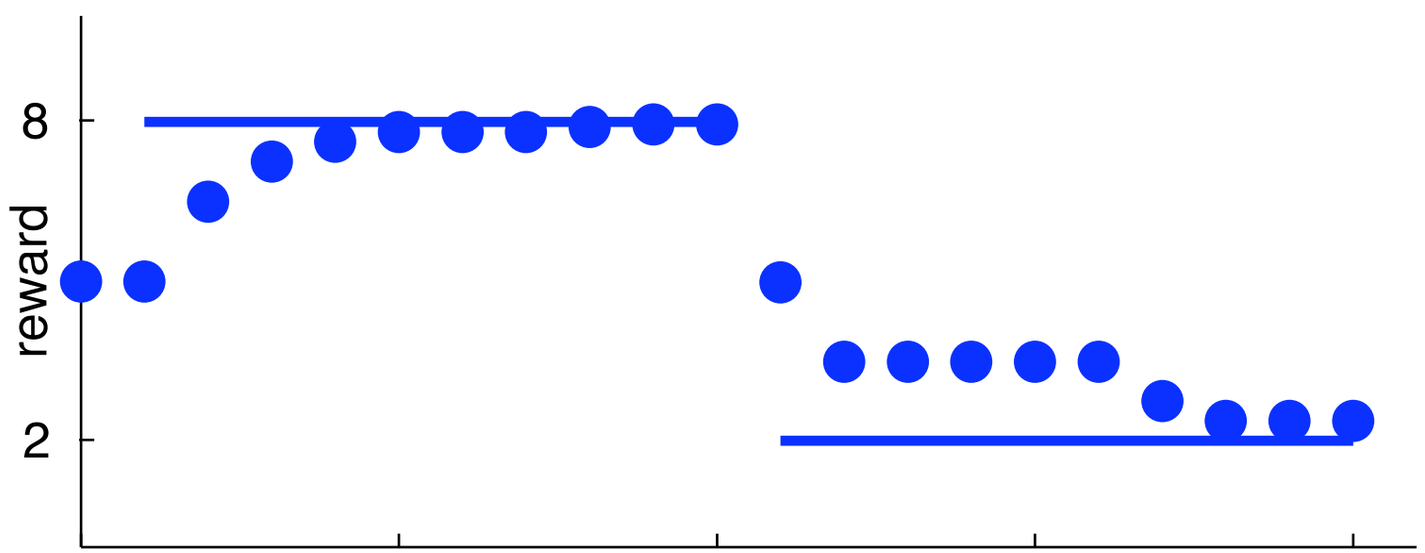
$$\delta_b = r_{b,i} - m_b$$
$$\delta_y = r_{y,i} - m_y$$



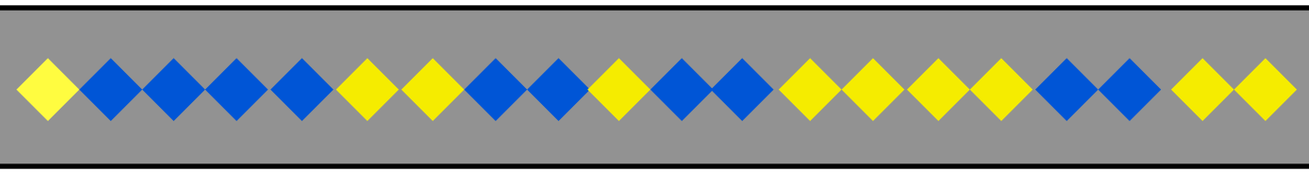
How can the bee learn the rewards?

"online" update:

$$m_b \longrightarrow m_b + \epsilon\delta_b$$
$$m_y \longrightarrow m_y + \epsilon\delta_y$$

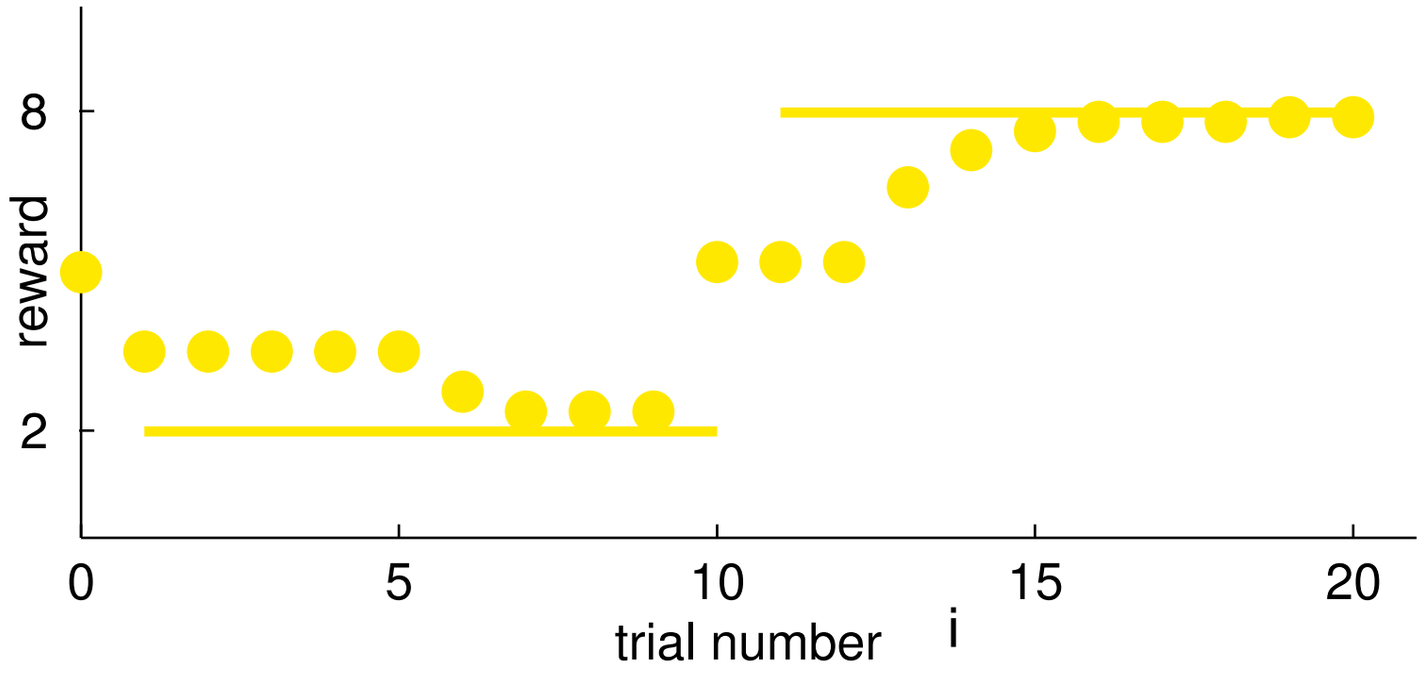


choices:



prediction error:

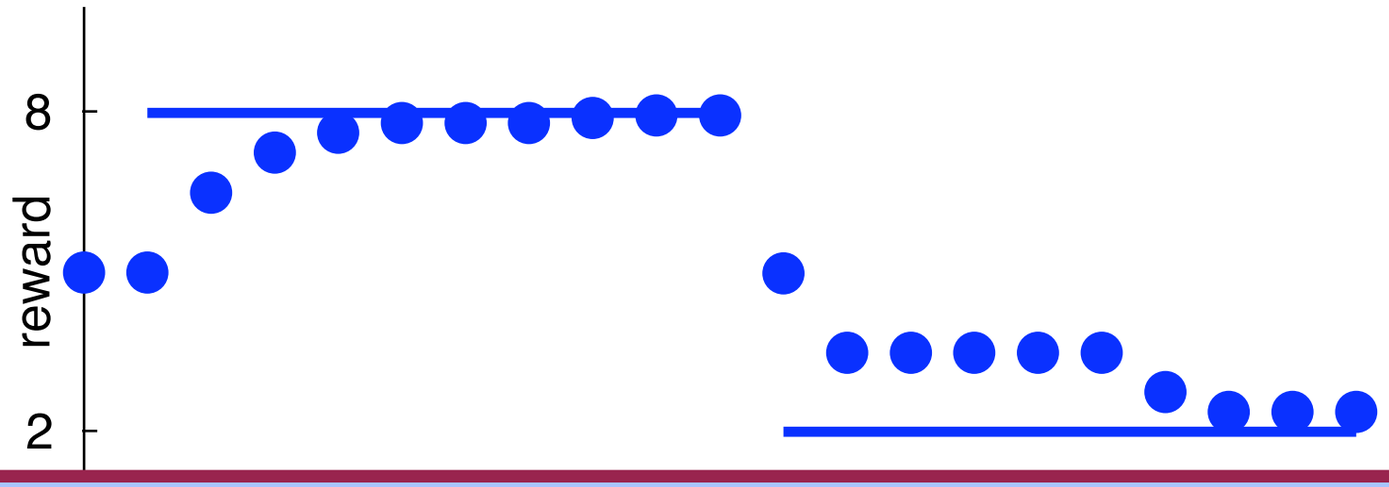
$$\delta_b = r_{b,i} - m_b$$
$$\delta_y = r_{y,i} - m_y$$



How can the bee learn the rewards?

"online" update:

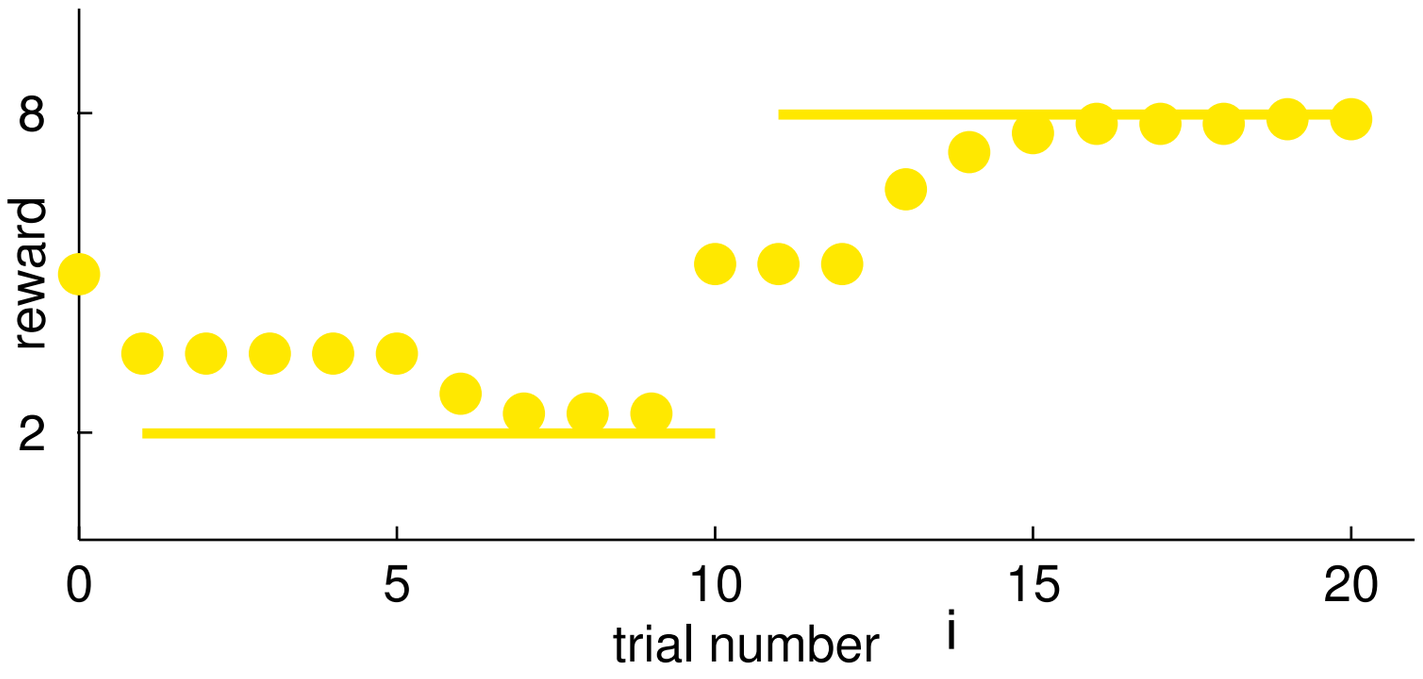
$$m_b \longrightarrow m_b + \epsilon\delta_b$$
$$m_y \longrightarrow m_y + \epsilon\delta_y$$



Idea: the bee could use its estimates about the flower rewards to change its policy!

prediction error:

$$\delta_b = r_{b,i} - m_b$$
$$\delta_y = r_{y,i} - m_y$$



Changing the policy online

$$p(b) = \frac{\exp(\beta m_b)}{\exp(\beta m_b) + \exp(\beta m_y)}$$

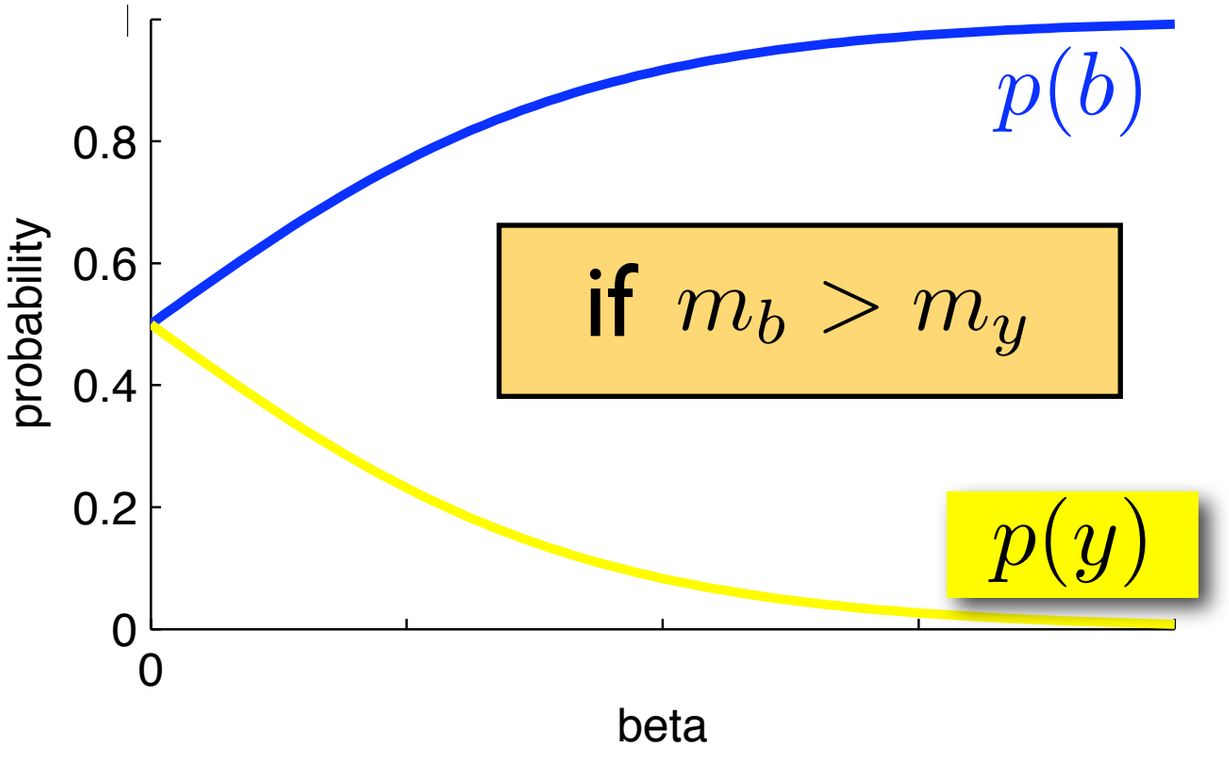
$$p(y) = \frac{\exp(\beta m_y)}{\exp(\beta m_b + \exp(\beta m_y)}$$

	actual reward	internal estimate
	r_b	m_b
	r_y	m_y

Changing the policy online

$$p(b) = \frac{\exp(\beta m_b)}{\exp(\beta m_b) + \exp(\beta m_y)}$$

$$p(y) = \frac{\exp(\beta m_y)}{\exp(\beta m_b + \exp(\beta m_y)}$$



← exploration

exploitation →

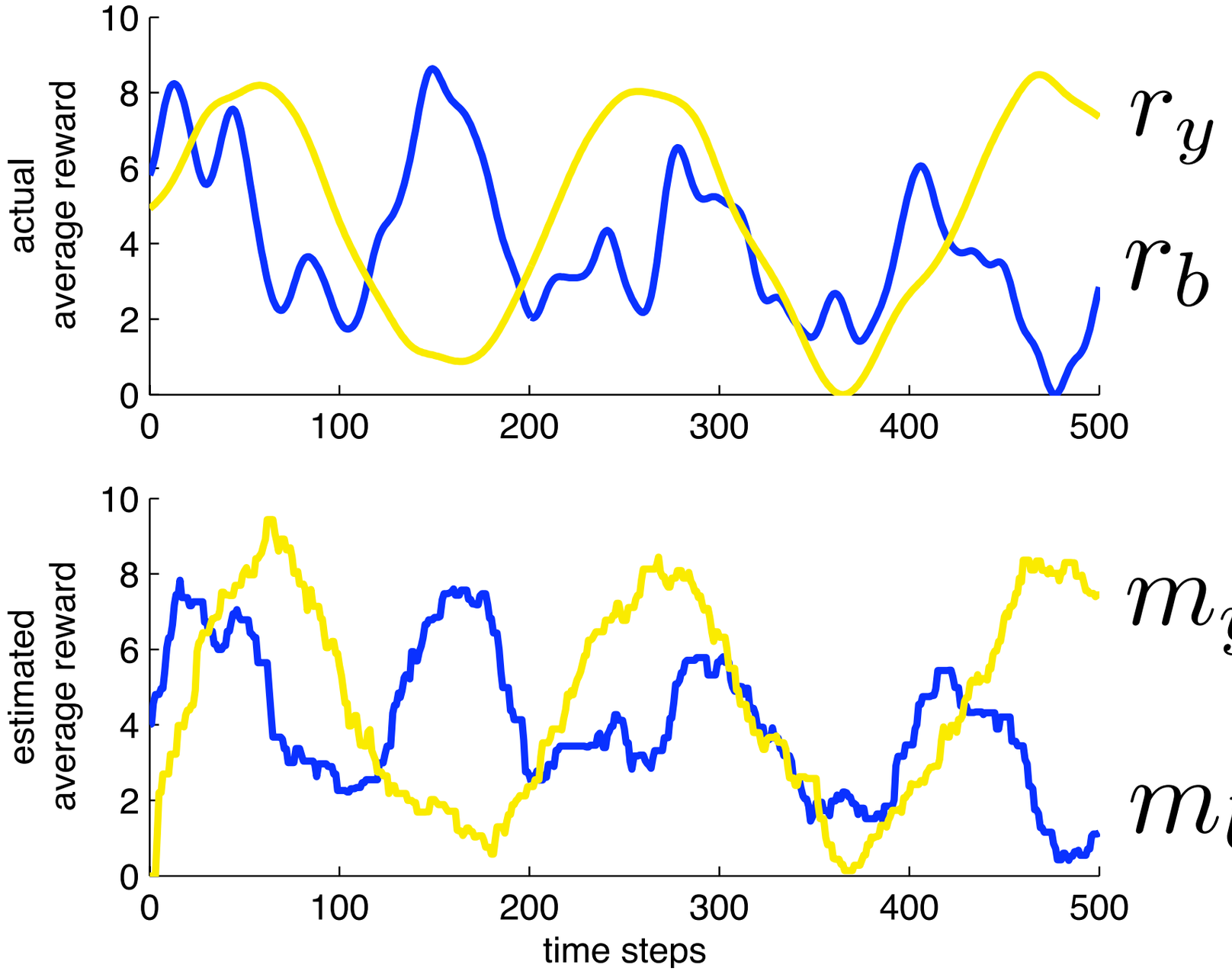
Learning the nectar reward: exploration-exploitation trade-off

softmax

Gibbs-policy with

$$\beta = 0$$

(50/50) policy

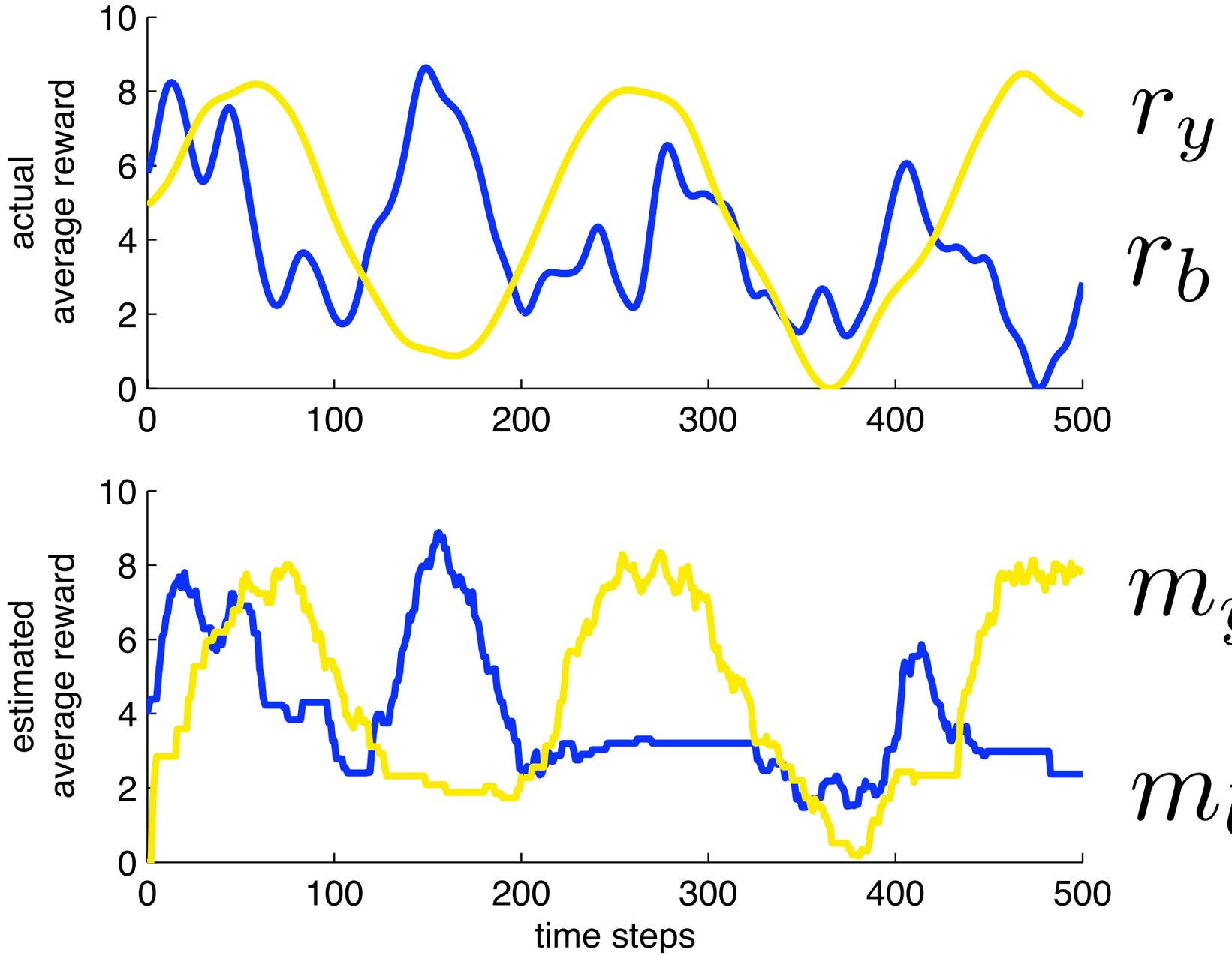


Learning the nectar reward: exploration-exploitation trade-off

softmax

Gibbs-policy with

$$\beta = 0.5$$

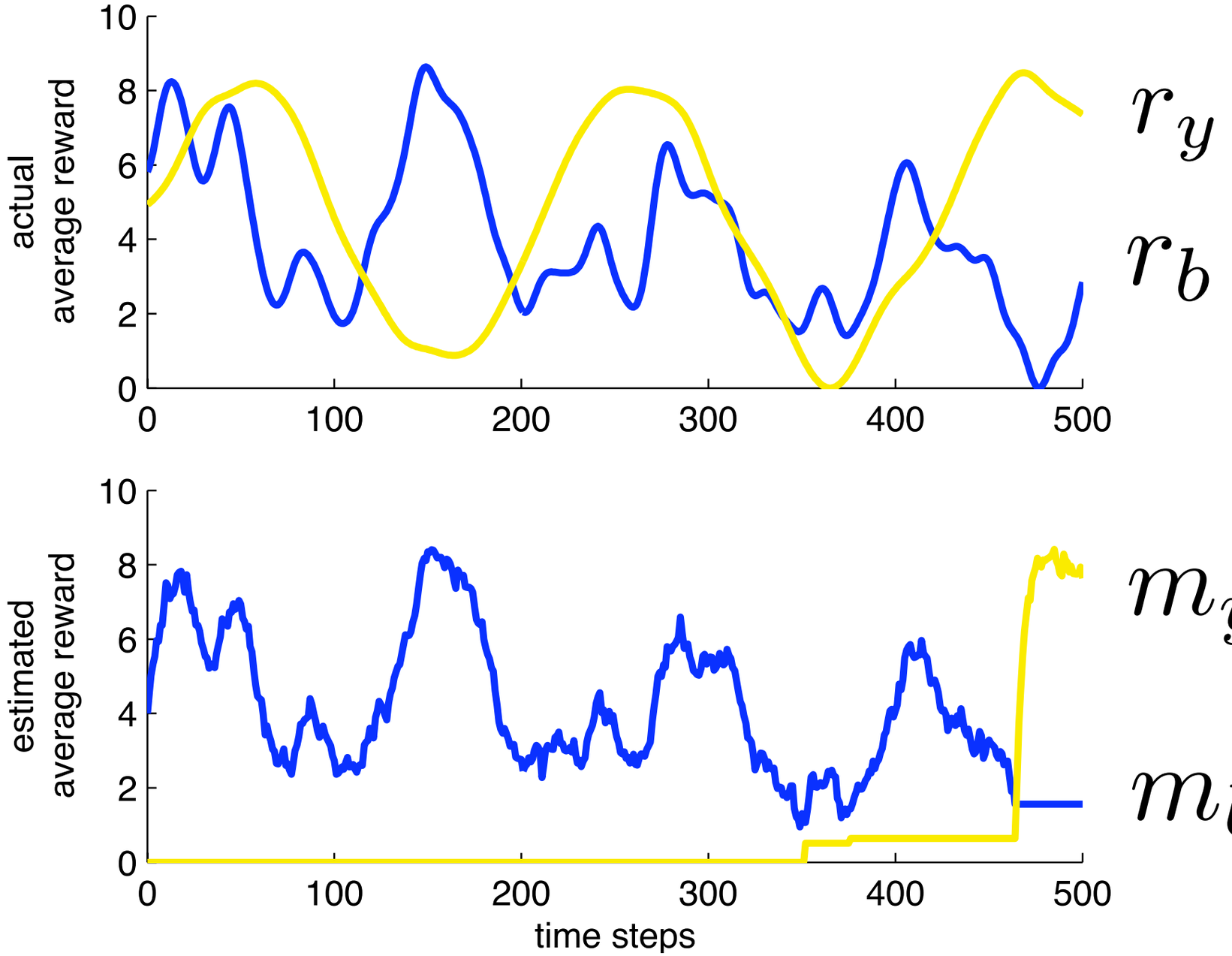


Learning the nectar reward: exploration-exploitation trade-off

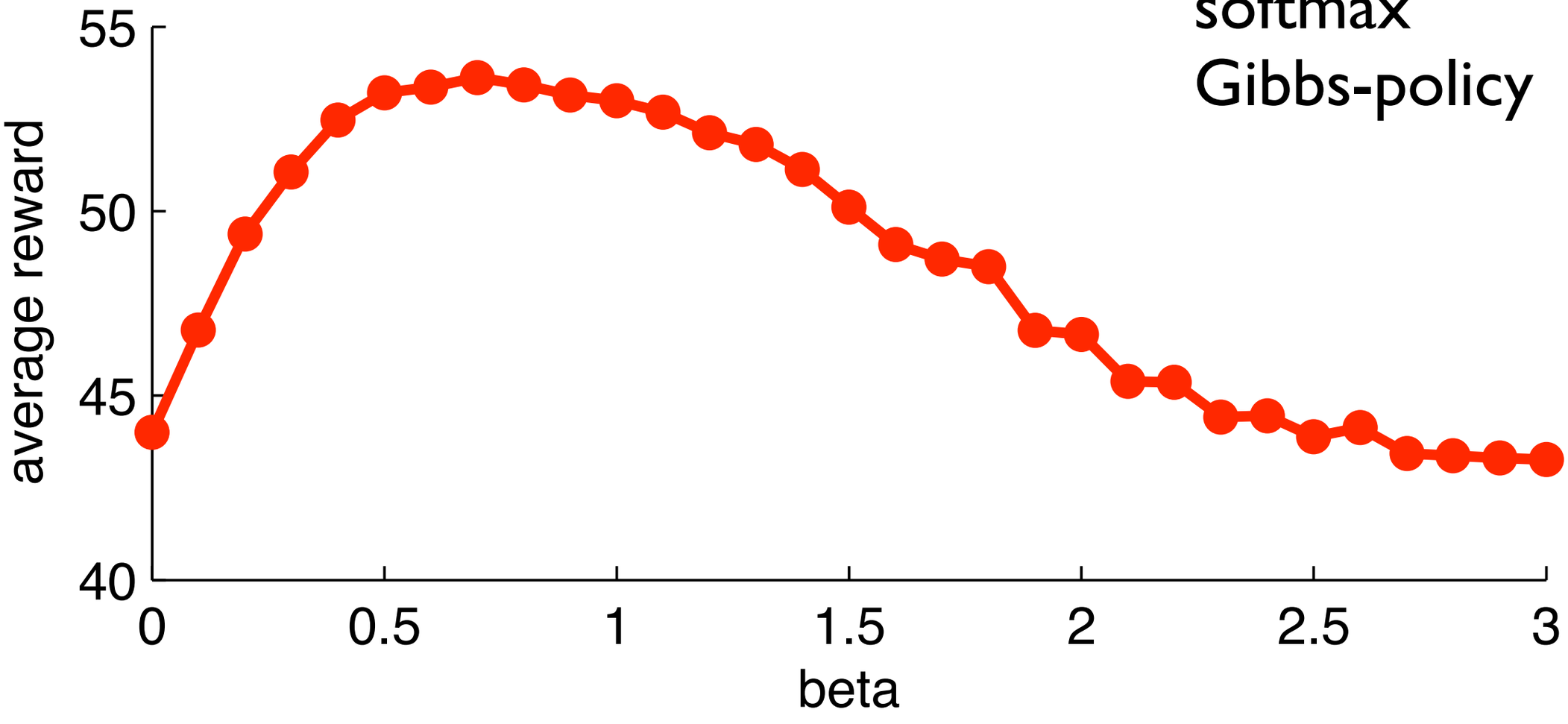
softmax

Gibbs-policy with

$$\beta = 2$$

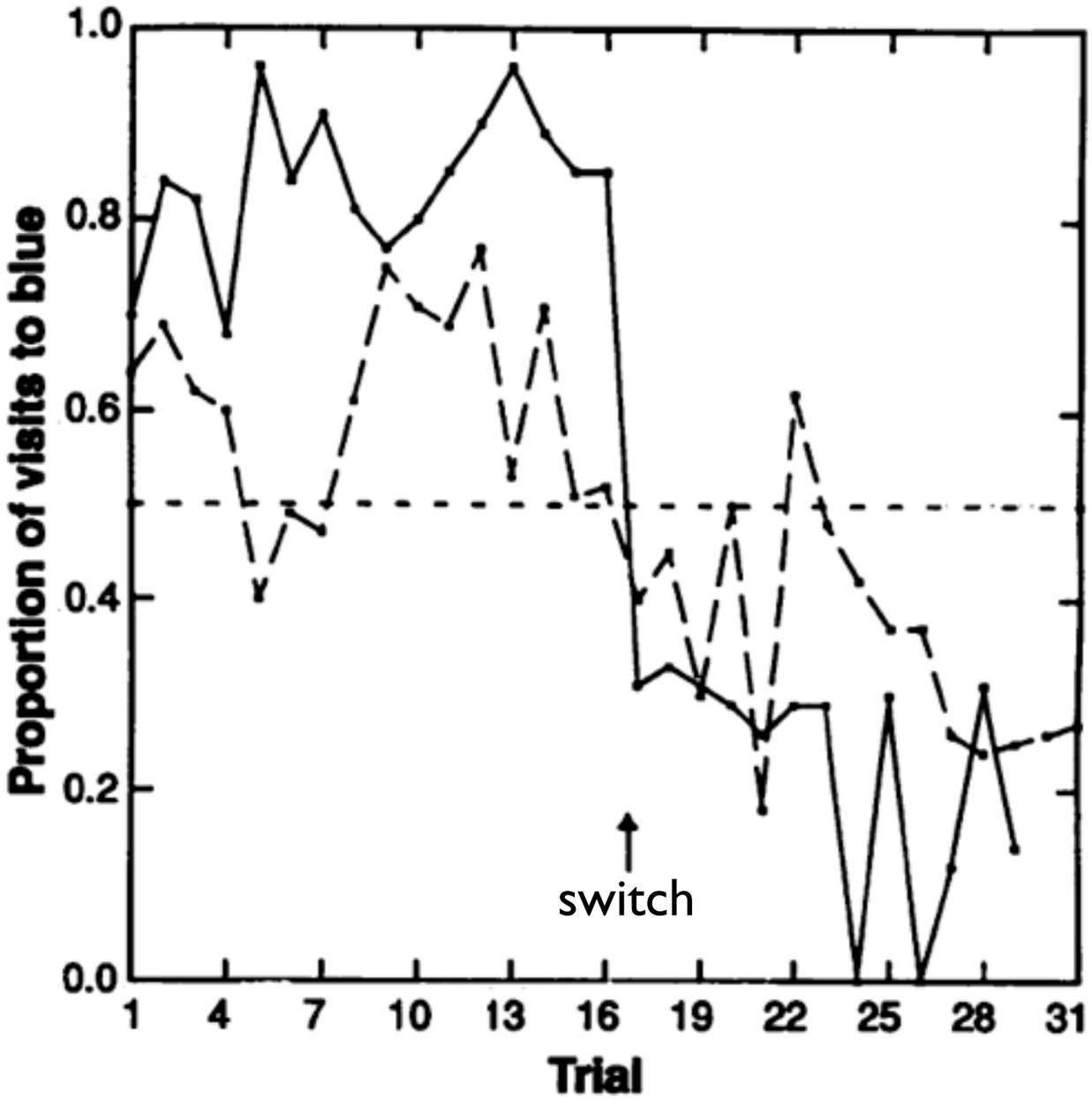


Exploration-exploitation tradeoff



softmax
Gibbs-policy

What do real bees do?



bumblebees (N=5)

solid line:

blue=constantly rewarded

yellow=randomly rewarded

switch at trial 17!

blue=randomly rewarded

yellow=constantly rewarded

dashed line:
less variability on randomly rewarded flowers

Real, 1991, Science 253:980-986