A gentle introduction to computational modelling

A step by step tutorial

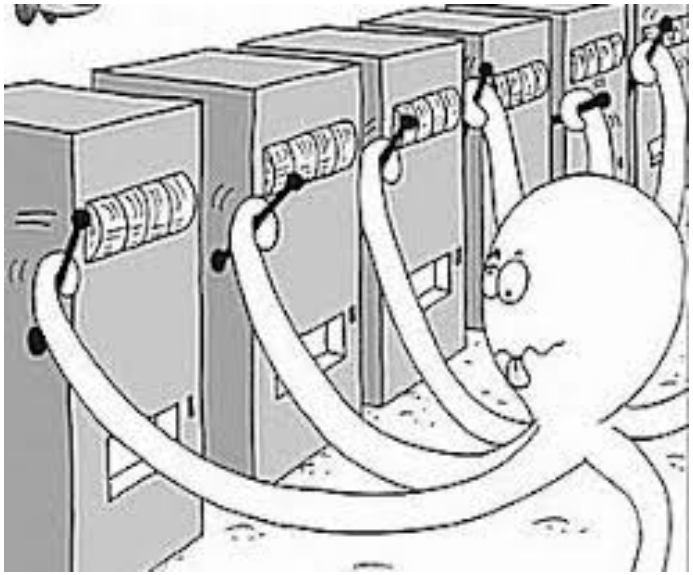
Vincent Valton

Neuroscience of Mental Health Group
Institute of Cognitive Neuroscience
University College London, UCL

Task: (multi-arm bandit problem)

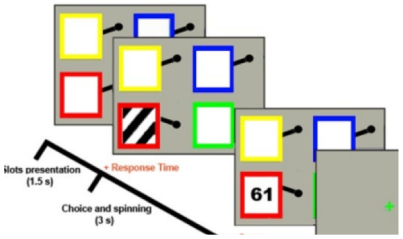
Typically defined as a reinforcement learning problem

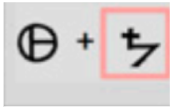
- Imagine we asked participants to play a game with three slot machines.
  - Each slot machine can either give a reward (1) or no-reward (0)
  - Each slot machine has a different probability of returning a reward



Instructions to participant:

Accumulate as many rewards
as possible by the end of the task





Task

- Imaginary task (3 slot machines)
  - Instructions: Accumulate as many rewards as possible

80% 5% 15%

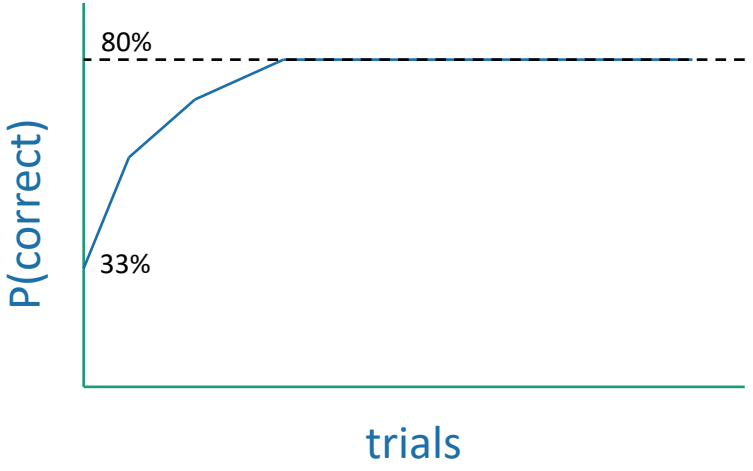
A B C

Task

- Imaginary task (3 slot machines)
  - Instructions: Accumulate as many rewards as possible

80% 5% 15%

A B C



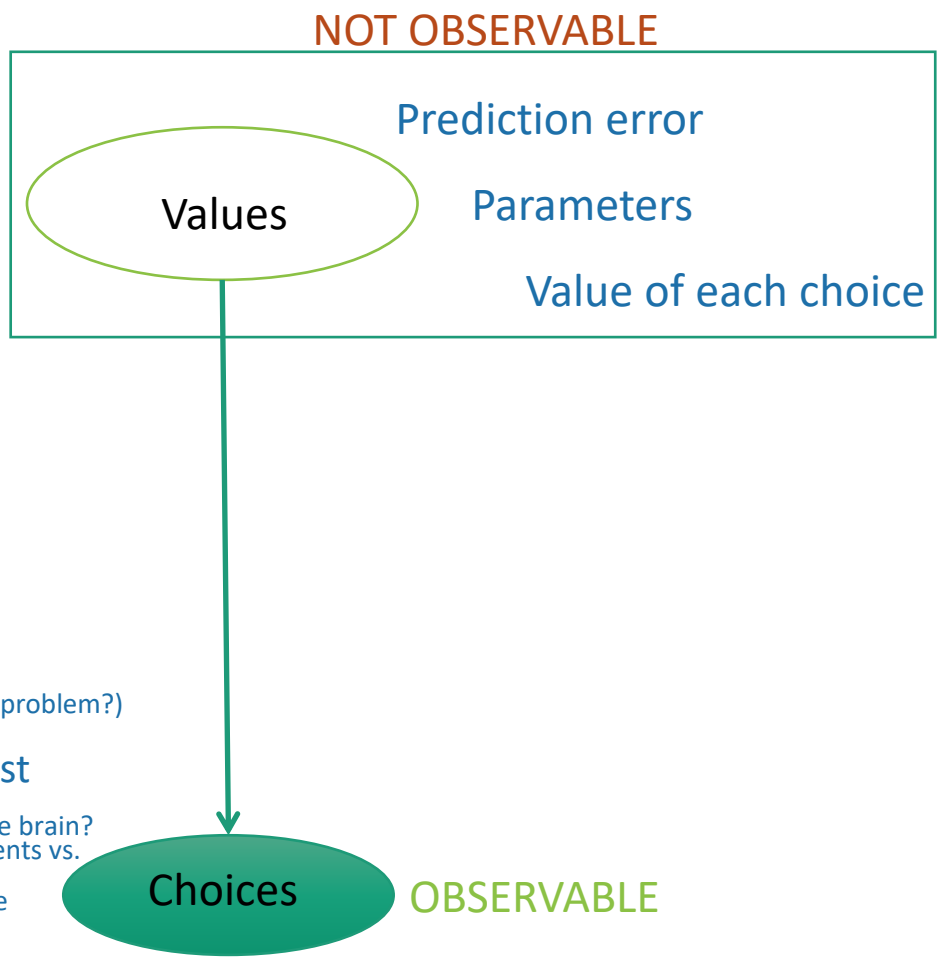
Trial by trial analysis of behaviour

- What are we interested in ?

- We want to:
  - Make assumptions as to how they learn/solve the task (heuristic/strategy)
  - Formalize mathematically that strategy
  - Fit the observation model to the choices of the participants

- To estimate their:
  - Option values (e.g. Q-values)
  - Model parameters (e.g. learning-rate, temperature)
  - Reward prediction errors (RPEs)

- Why:
  - Compare strategies/heuristics
    (i.e. what strategies are participants using to solve this task/problem?)
  - Regress these to other variables of interest
    (e.g. Where is RPE encoded in the brain?
    Where is the value of each bandit/action encoded in the brain?
    Does any of the model parameters differ between patients vs. controls?
    Does it correlate with symptoms scores or questionnaire measures, etc.)



Formalize 'Learning model' mathematically

'How do people learn the values from experience?'

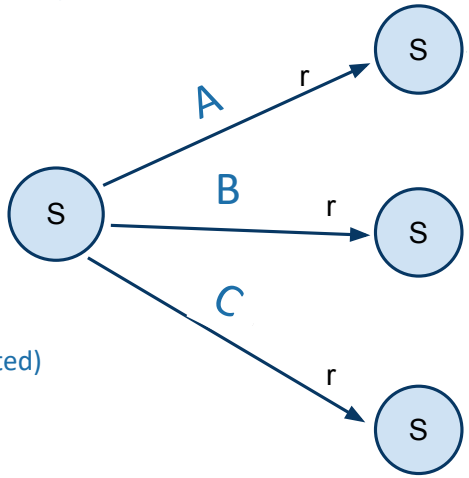
- Goal:
  - Learn which actions (or slot machines) lead to the most amount of money
  - How: Assign a value that represent the 'goodness' of each state (or state-action pair) based on experience.

Learning Model

- We have : a model of the task (states, actions, rewards)

Q-learning

- $Q(s,a)= Q(s,a) + \alpha (r-Q(s,a))$
  - α = learning rate
  - r-Q(s,a) = Reward Prediction Error (RPEs)
    (difference between what was received as a reward and what was expected)



Formalize 'Learning model':
A one time-step example

Learning Model

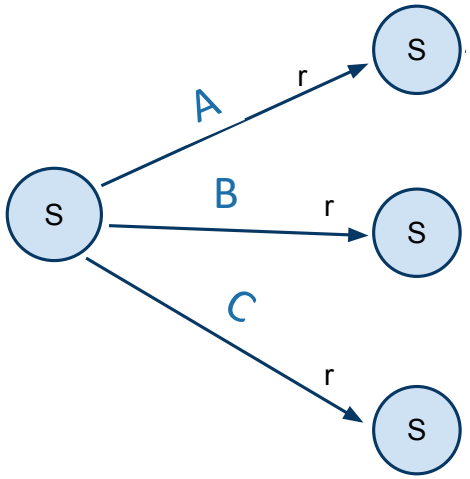
Q-learning

- $Q(s,a)= Q(s,a) + \alpha (r-Q(s,a))$
  - α = learning rate
  - r-Q(s,a) = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- $Q\_new(s,'A') = Q\_old(s,'A') + \alpha \times (r - Q\_old(s,'A'))$



Formalize 'Learning model':
A one time-step example

Learning Model

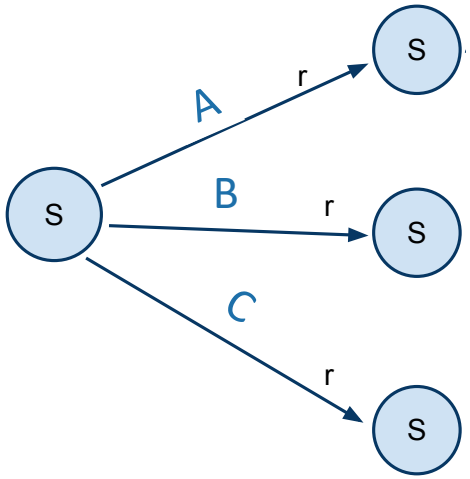
Q-learning

- $Q(s,a)= Q(s,a) + \alpha\ (r-Q(s,a))$
  - α = learning rate
  - r-Q(s,a) = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- $Q\_new(s,'A') = \underbrace{Q\_old(s,'A')}_{0.33} + \alpha \times (\ r\ - \underbrace{Q\_old(s,'A')}_{0.33})$



Formalize 'Learning model':
A one time-step example

Learning Model

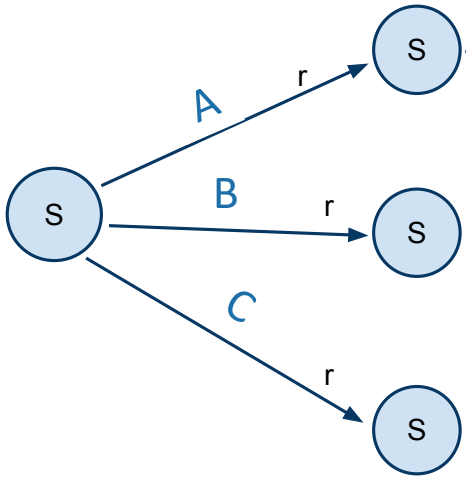
Q-learning

- $Q(s,a)= Q(s,a) + \alpha\ (r-Q(s,a))$
  - α = learning rate
  - r-Q(s,a) = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)



Formalize 'Learning model':
A one time-step example

Learning Model

Q-learning

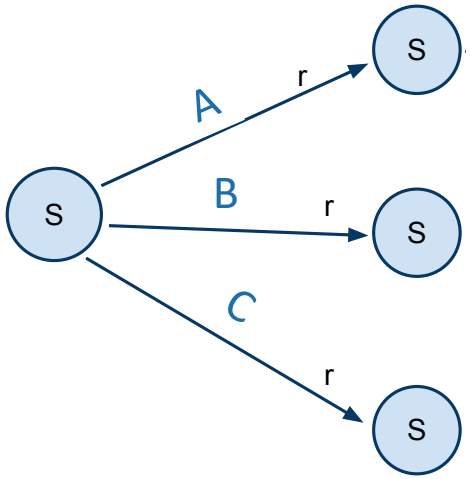
- $Q(s,a) = Q(s,a) + \alpha\,(r - Q(s,a))$
  - α = learning rate
  - r-Q(s,a) = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- $Q_{new}(s,'A') = Q_{old}(s,'A') + \alpha \times (r - Q_{old}(s,'A'))$
- $Q_{new}(s,'A') = 0.33 + \alpha \times (1 - 0.33)$

  α = 0.7



Formalize 'Learning model':
A one time-step example

Learning Model

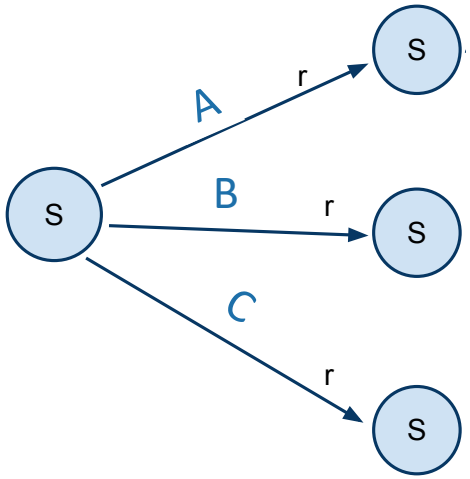
Q-learning

- $Q(s,a) = Q(s,a) + \alpha\,(r - Q(s,a))$
  - α = learning rate
  - $r - Q(s,a)$ = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (1 - 0.33)



Formalize 'Learning model':
A one time-step example

Learning Model

Q-learning

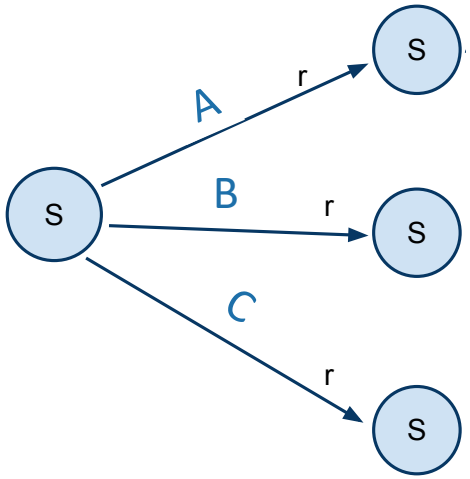
- $Q(s,a) = Q(s,a) + \alpha\,(r - Q(s,a))$
  - α = learning rate
  - $r - Q(s,a)$ = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (1 - 0.33)

0.67



Formalize 'Learning model':
A one time-step example

Learning Model

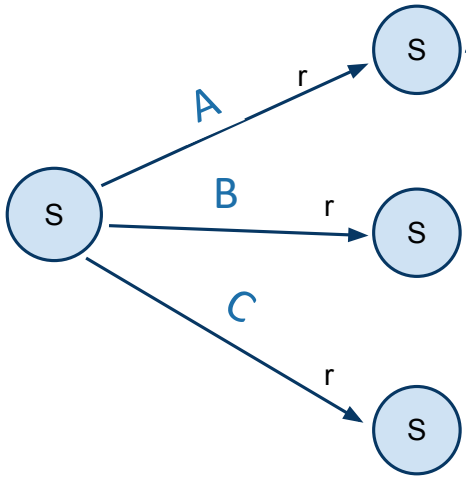
Q-learning

- $Q(s,a) = Q(s,a) + \alpha\,(r - Q(s,a))$
  - α = learning rate
  - $r - Q(s,a)$ = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (0.67)



Formalize 'Learning model':
A one time-step example

Learning Model

Q-learning

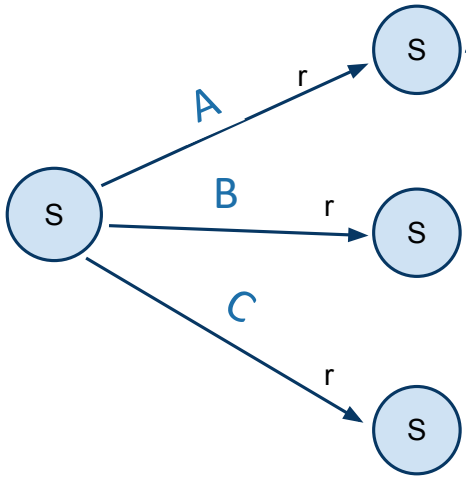
- $Q(s,a) = Q(s,a) + \alpha\,(r - Q(s,a))$
  - α = learning rate
  - $r - Q(s,a)$ = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (0.67)

0.469



Formalize 'Learning model':
A one time-step example

Learning Model

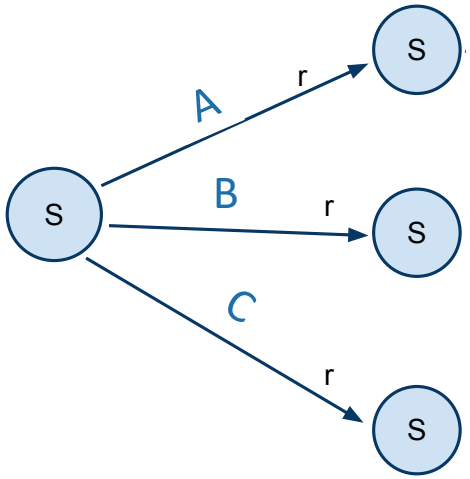
Q-learning

- $Q(s,a)= Q(s,a) + \alpha\,(r-Q(s,a))$
  - α = learning rate
  - r-Q(s,a) = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (0.67)
- Q_new(s,'A') = 0.33 + 0.469



Formalize 'Learning model':
A one time-step example

Learning Model

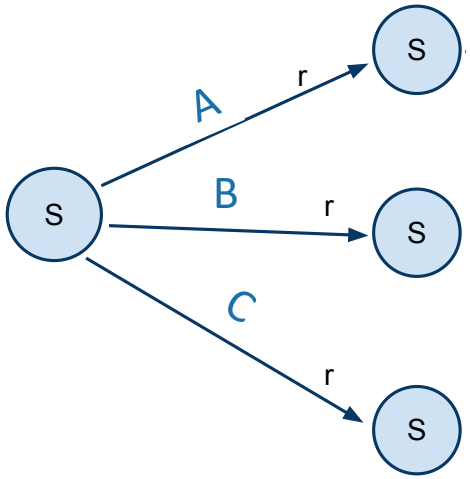
Q-learning

- $Q(s,a) = Q(s,a) + \alpha\,(r - Q(s,a))$
  - α = learning rate
  - $r - Q(s,a)$ = Reward Prediction Error (difference between what you received as a reward and what you expected to receive)

1) Rewards=[0,1], all state-action pairs initialized to '0.33' to make all options equally good (represent no a-priori expectation that one state-action pair being better than another)

2) We select action/bandit 'A' and get a reward r=1; assume α=0.7:

- Q_new(s,'A') = Q_old(s,'A') + α x (r - Q_old(s,'A'))
- Q_new(s,'A') = 0.33 + α x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (1 - 0.33)
- Q_new(s,'A') = 0.33 + 0.7 x (0.67)
- Q_new(s,'A') = 0.33 + 0.469
- Q_new(s,'A') = 0.799



Observation model : Action selection

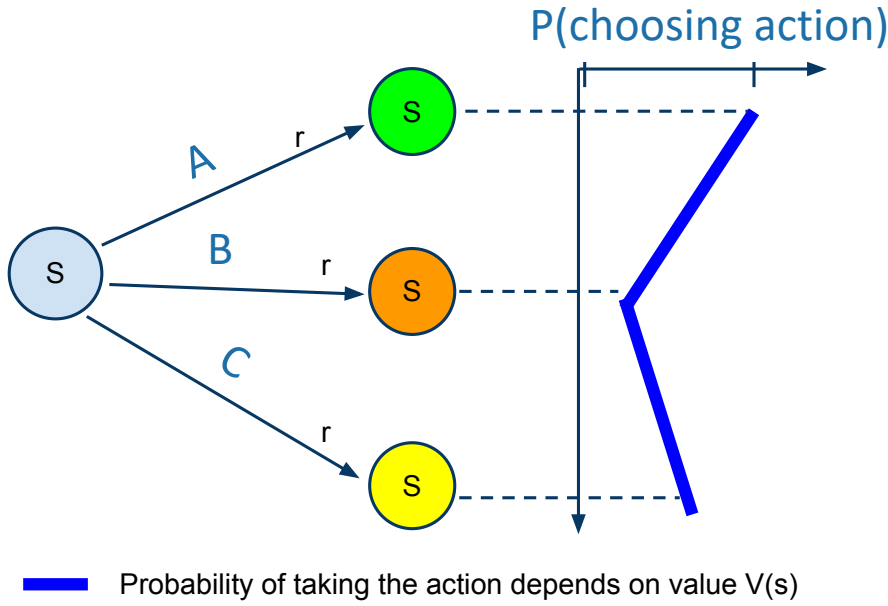
'How do people make choices?'

Softmax

Returns a probability distribution over actions as a function of their value and a temperature parameter:

- Exploration is not random but depends on the values of each action

- 'τ' is the temperature parameter that scales the values of each action in comparison to each other.



$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{\sum_{b=1}^{3} exp\left(V\left(b\right)/\tau\right)}$$

(attention: τ vs β in literature): 'τ' is often used to describe 'temperature', while 'β' is often used to describe 'inverse temperature'

detail of implementation that change the interpretation of the parameter values (small τ = large β = deterministic 'exploitative' behaviour, large τ = small β = random 'exploratory' behaviour)

Observation model : Action selection

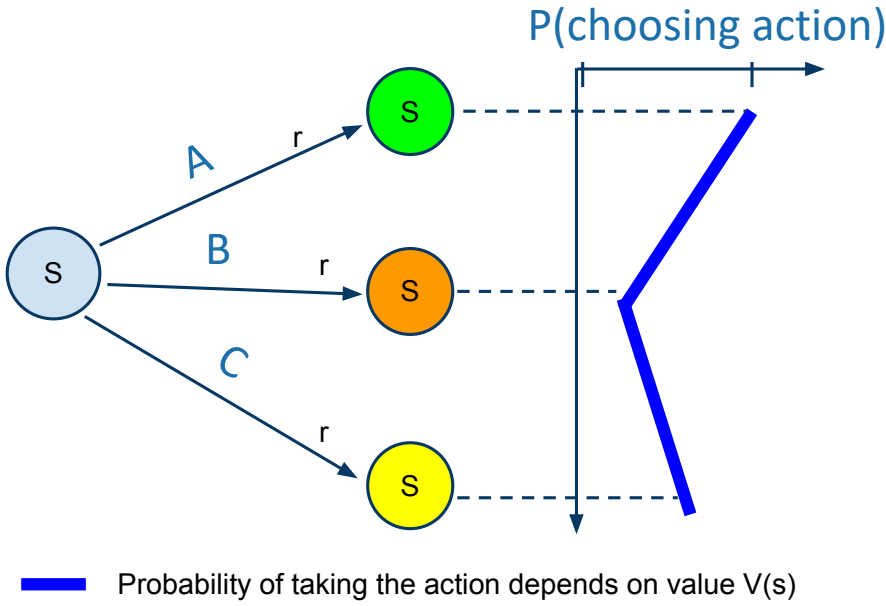
'How do people make choices?'

Softmax

Returns a probability distribution over actions as a function of their value and a temperature parameter:

- Exploration is not random but depends on the values of each action

- 'τ'is the temperature parameter that scales the values of each action in comparison to each other.



$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{\sum_{b=1}^{3} exp\left(V\left(b\right)/\tau\right)}$$

(attention: τ vs β in literature): 'τ' is often used to describe 'temperature', while 'β' is often used to describe 'inverse temperature'

$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{exp\left(V\left(A\right)/\tau\right) + exp\left(V\left(B\right)/\tau\right) + exp\left(V\left(C\right)/\tau\right)}$$

detail of implementation that change the interpretation of the parameter values (small τ = large β = deterministic 'exploitative' behaviour, large τ = small β = random 'exploratory' behaviour)

Observation model : Action selection

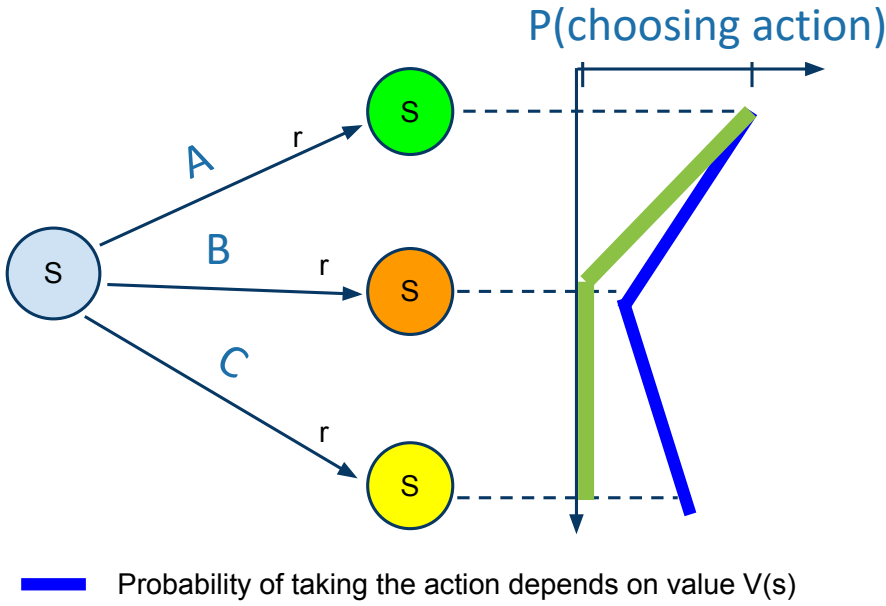
'How do people make choices?'

Softmax

Returns a probability distribution over actions as a function of their value and a temperature parameter:

- Exploration is not random but depends on the values of each action

- 'τ'is the temperature parameter that scales the values of each action in comparison to each other.



$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{\sum_{b=1}^{3} exp\left(V\left(b\right)/\tau\right)}$$

$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{exp\left(V\left(A\right)/\tau\right) + exp\left(V\left(B\right)/\tau\right) + exp\left(V\left(C\right)/\tau\right)}$$

(attention: τ vs β in literature): 'τ' is often used to describe 'temperature', while 'β' is often used to describe 'inverse temperature'

detail of implementation that change the interpretation of the parameter values (small τ = large β = deterministic 'exploitative' behaviour, large τ = small β = random 'exploratory' behaviour)

Observation model : Action selection

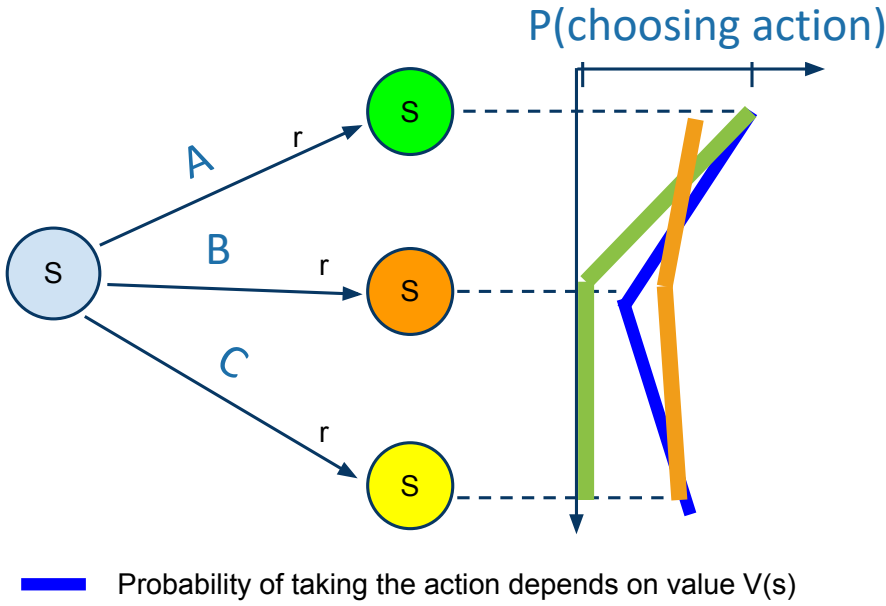
'How do people make choices?'

Softmax

Returns a probability distribution over actions as a function of their value and a temperature parameter:

- Exploration is not random but depends on the values of each action

- 'τ'is the temperature parameter that scales the values of each action in comparison to each other.



$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{\sum_{b=1}^{3} exp\left(V\left(b\right)/\tau\right)}$$

$$p\left(action = A\right) = \frac{exp\left(V\left(A\right)/\tau\right)}{exp\left(V\left(A\right)/\tau\right) + exp\left(V\left(B\right)/\tau\right) + exp\left(V\left(C\right)/\tau\right)}$$

(attention: τ vs β in literature): 'τ' is often used to describe 'temperature', while 'β' is often used to describe 'inverse temperature'

detail of implementation that change the interpretation of the parameter values (small τ = large β = deterministic 'exploitative' behaviour, large τ = small β = random 'exploratory' behaviour)

Parameter estimation (model inversion)

'Now that we have defined a model, how do we estimate the parameters for each participant?'

- We have a model (M) which is the composite of:
  - the learning model (Q-learning)
  - the observation model (Softmax)

- The resulting model (M) has a set of free parameters:
  - the learning rate: α (from the learning model)
  - the temperature parameter: $\boldsymbol{\tau}$ (from the observation model)

- We have a data-set (D) for each subject:
  - Choices made on each trial
    (e.g. : A, C, A, A, A, B, C, A, A, A, …)

- Problem: We want to find the parameter values (α,$\boldsymbol{\tau}$) for each subject, such that the model fits the choices of the participant as closely as possible

$$p(\alpha,\boldsymbol{\tau}|D,M)$$

Parameter estimation

'Example in 1D, with a single parameter to estimate'

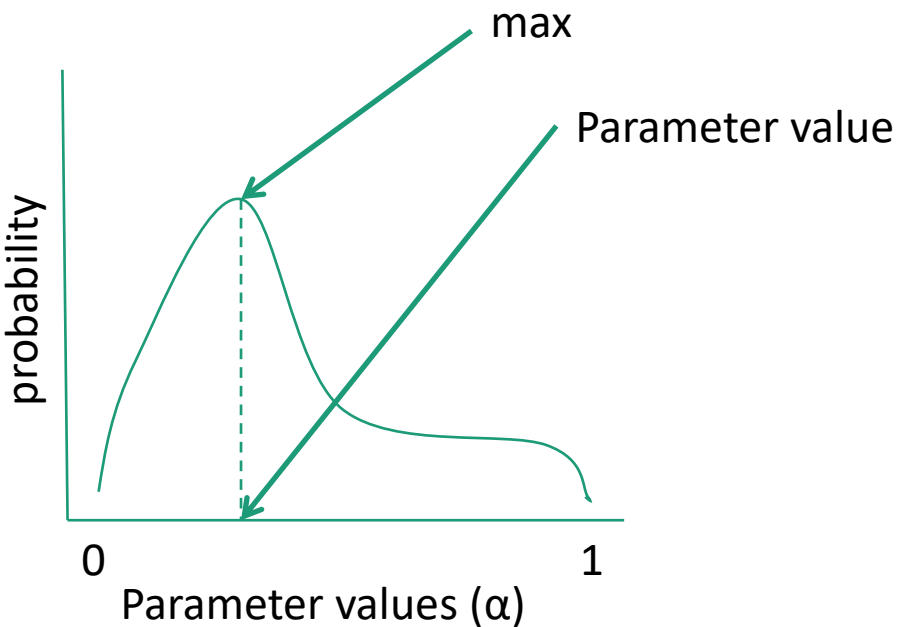
- Problem: We want to find the parameter values (α,$\boldsymbol{\tau}$) for each subject, such that the model fits the choices of the participant as closely as possible

$$p(\alpha,\boldsymbol{\tau}|D,M)$$

Example with only 1 parameter:

$$p(\alpha|D,M)$$

Find parameter value that maximize the probability of having choices similar to that of the subject



Parameter estimation

'Example in 1D, with a single parameter to estimate'

No a-priori over which parameters are more likely than others
(uninformative prior – flat)

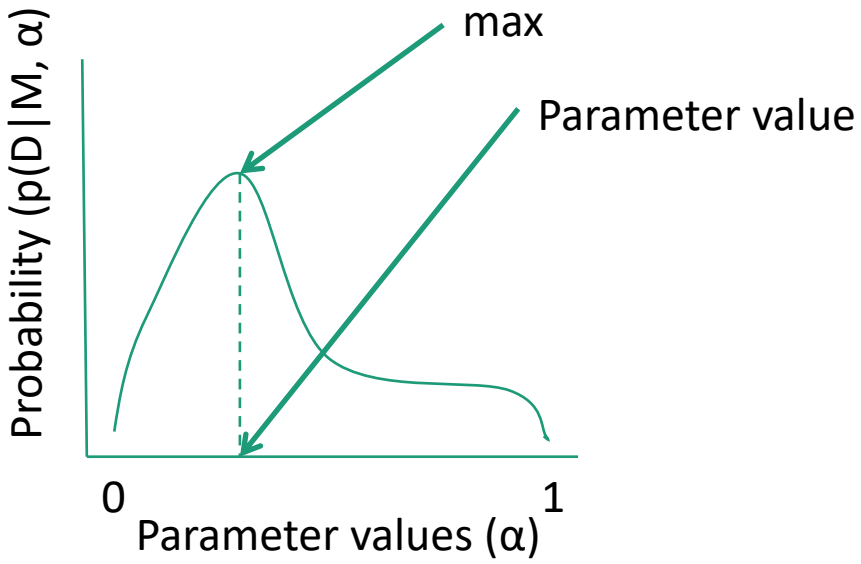
Maximum Likelihood:

$$\underbrace{p(\alpha,\boldsymbol{\tau}|D,M)}_{\text{Posterior}} \propto \underbrace{p(D|M,\alpha,\boldsymbol{\tau})}_{\text{Likelihood}} . \underbrace{P(\alpha,\boldsymbol{\tau}|M)}_{\text{Prior}}$$

Example with only 1 parameter:

$$p(\alpha|D,M)$$

Find parameter value that maximize the probability of having choices similar to that of the subject



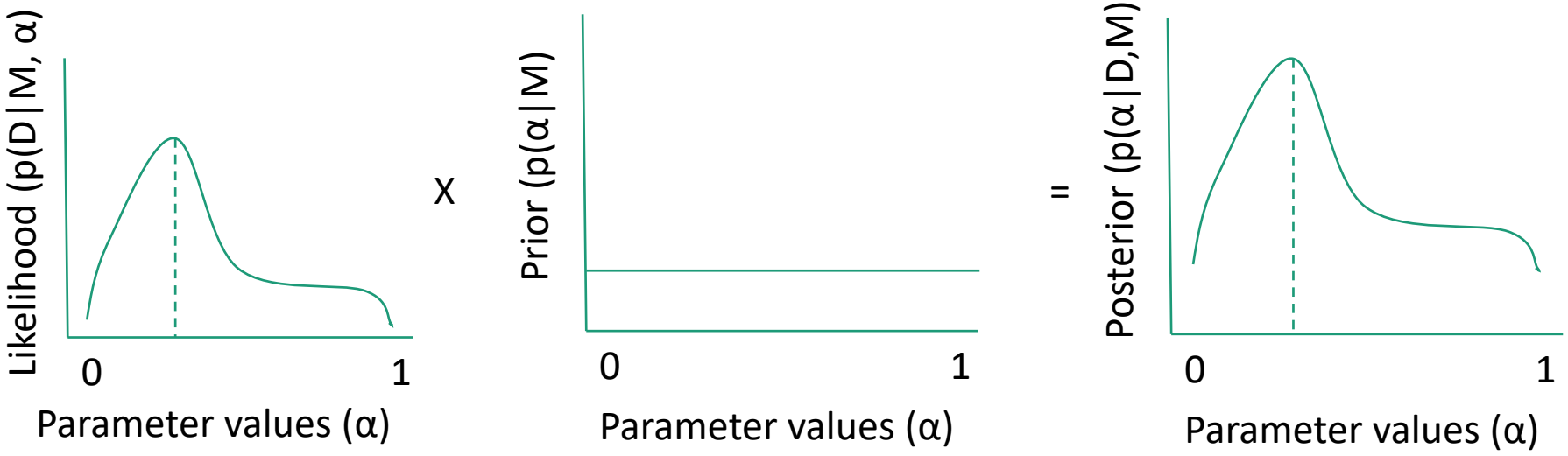
Parameter estimation

'Example in 1D, with a single parameter to estimate'

Maximum Likelihood:

Posterior | Likelihood | Prior

$$p(\alpha,\boldsymbol{\tau}|D,M) \propto p(D|M,\alpha,\boldsymbol{\tau}) \cdot \cancel{P(\alpha,\boldsymbol{\tau}|M)}$$



Parameter estimation

'How do I calculate this likelihood?'

Maximum Likelihood:

$$p(\alpha,\tau|D,M) \propto p(D|M,\alpha,\tau) \cdot P(\alpha,\tau|M)$$

What is $p(D|M,\alpha,\tau)$? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, C, A, A, A, …

And a set of parameter values for 'α', and 'τ':

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M,\alpha,\tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

$$p(action = A) = \frac{exp(V(A)/\tau)}{exp(V(A)/\tau) + exp(V(B)/\tau) + exp(V(C)/\tau)}$$

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,$\boldsymbol{\tau}$) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33

$$p(D \mid M, \alpha, \tau) =$$

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33

$$p(D \mid M, \alpha, \tau) = 0.33$$

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2			

Update Q-Value of A given reward just received & re-calculate using Softmax for each action

$$p(D \mid M, \alpha, \tau) = 0.33$$

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20

Update Q-Value of A given reward just received & re-calculate using Softmax for each action

$$p(D \mid M, \alpha, \tau) = 0.33$$

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20

Update Q-Value of A given reward just received & re-calculate using Softmax for each action

$p(D \mid M, \alpha, \tau) =$ 0.33 x 0.20

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20

Update Q-Value of A given reward just received & re-calculate using Softmax for each action

$p(D \mid M, \alpha, \tau) =$ 0.33 x 0.20

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Update Q-Value of A given reward just received & re-calculate using Softmax for each action

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20
3	0.75	0.15	0.10

$p(D \mid M, \alpha, \tau) =$ 0.33 x 0.2 x 0.75

Parameter estimation

‘How do I calculate this likelihood?’ – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

$$p(D \mid M, \alpha, \tau) = 0.33 \times 0.2 \times 0.75 \times 0.79 \times 0.64 \times \ldots$$

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20
3	0.75	0.15	0.10
4	0.79	0.11	0.04
5	0.64	0.26	0.10

Parameter estimation

'How do I calculate this likelihood?' – a worked example for 5 trials

What is p(D|M,α,τ) ? How do I calculate it ?

Given the sequence of choices of a subject (your data D):

E.g.: A, C, A, A, A, B, …

Your likelihood is the product (multiplication) of the probability of selecting the action at each trial.

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

$$p(D \mid M, \alpha, \tau) = \text{0.33 x 0.2 x 0.75 x 0.79 x 0.64 x …}$$

Problem! Will quickly result in a very small number
(e.g.: after 5 trials only, the likelihood is equal to 0.025)

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20
3	0.75	0.15	0.10
4	0.79	0.11	0.04
5	0.64	0.26	0.10

Parameter estimation

'Why do I keep hearing about log-likelihoods instead of likelihoods?'

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

$$p(D \mid M, \alpha, \tau) = \text{0.33 x 0.2 x 0.75 x 0.79 x 0.64 x ...}$$

Problem! Will quickly result in a very small number
(e.g.: after 5 trials only, the likelihood is equal to 0.025)

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20
3	0.75	0.15	0.10
4	0.79	0.11	0.04
5	0.64	0.26	0.10

We could potentially run into issues when we have a lot more trials per subjects.

This number becomes so small we may end up hitting the floating-point precision of the computer (resulting in underflows)

An elegant solution is to use the natural logarithm. The Log is monotonically increasing function, which enables us to apply a monotone transformation to the likelihood. Because of the monotone nature of the transformation, the parameters that maximize the likelihood will be the same as those that maximize the log-likelihood.

Also we get to use sums of log-likelihoods rather than a product of likelihoods, which is typically faster for a computer to perform than multiplications.

Parameter estimation

'Why do I keep hearing about log-likelihoods instead of likelihoods?'

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

$$p(D \mid M, \alpha, \tau) = \underbrace{0.33 \times 0.2 \times 0.75 \times 0.79 \times 0.64 \times \ldots}$$

Problem! Will quickly result in a very small number
(e.g.: after 5 trials only, the likelihood is equal to 0.025)

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20
3	0.75	0.15	0.10
4	0.79	0.11	0.04
5	0.64	0.26	0.10

We could potentially run into issues when we have a lot more trials per subjects.

This number becomes so small we may end up hitting the floating-point precision of the computer (resulting in underflows)

An elegant solution is to use the natural logarithm. The Log is monotonically increasing function, which enables us to apply a monotone transformation to the likelihood. Because of the monotone nature of the transformation, the parameters that maximize the likelihood will be the same as those that maximize the log-likelihood.

Also we get to use sums of log-likelihoods rather than a product of likelihoods, which is typically faster for a computer to perform than multiplications.

$$ln \quad p(D \mid M, \alpha, \tau) = \sum_{trial=1}^{N} ln \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

Parameter estimation

'Why do I keep hearing about log-likelihoods instead of likelihoods?'

Value of selected action on trial t

$$p(D \mid M, \alpha, \tau) = \prod_{trial=1}^{N} \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

$$p(D \mid M, \alpha, \tau) = \underbrace{0.33 \times 0.2 \times 0.75 \times 0.79 \times 0.64 \times \ldots}$$

Problem! Will quickly result in a very small number
(e.g.: after 5 trials only, the likelihood is equal to 0.025)

Trial	P(select A)	P(select B)	P(select C)
1	0.33	0.33	0.33
2	0.60	0.20	0.20
3	0.75	0.15	0.10
4	0.79	0.11	0.04
5	0.64	0.26	0.10

We could potentially run into issues when we have a lot more trials per subjects.

This number becomes so small we may end up hitting the floating-point precision of the computer (resulting in underflows)

An elegant solution is to use the natural logarithm. The Log is monotonically increasing function, which enables us to apply a monotone transformation to the likelihood. Because of the monotone nature of the transformation, the parameters that maximize the likelihood will be the same as those that maximize the log-likelihood.

Also we get to use sums of log-likelihoods rather than a product of likelihoods, which is typically faster for a computer to perform than multiplications.

$$ln \quad p(D \mid M, \alpha, \tau) = \sum_{trial=1}^{N} ln \left(\frac{exp(V(s)/\tau)}{\sum_{b=1}^{3} exp(V(b)/\tau)} \right)$$

= Ln(0.33) + Ln(0.2) + Ln(0.75) + …

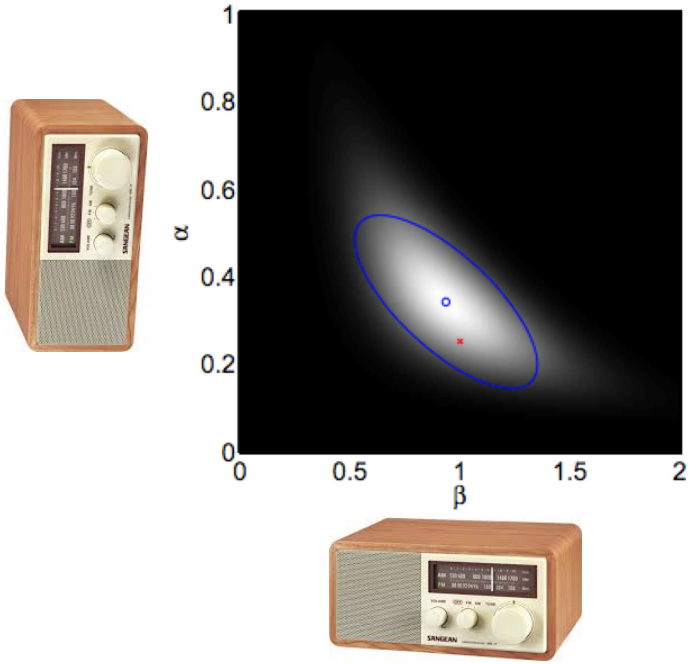
Maximum Likelihood – Implementation

'Now that I can calculate the log-likelihood, how do I find the best parameters?'

$$\hat{\alpha}, \hat{\tau} = \arg\max \ln p(D \mid M, \alpha, \tau)$$

- Could discretize our search space over parameter values (Grid-search):
  - We could calculate the likelihood at all parameter combinations of 'α' and 'τ' from within a given range and small enough step-size, and pick the parameter values that return the highest likelihood.

    (e.g. try all combination of alpha = [0:0.01:1], tau = [0:0.01:2])



But this is not efficient, and would not scale well if we have more than a handful of free parameters

Daw 2009 "Trial by trial data analysis using computational models"

Maximum Likelihood – Implementation

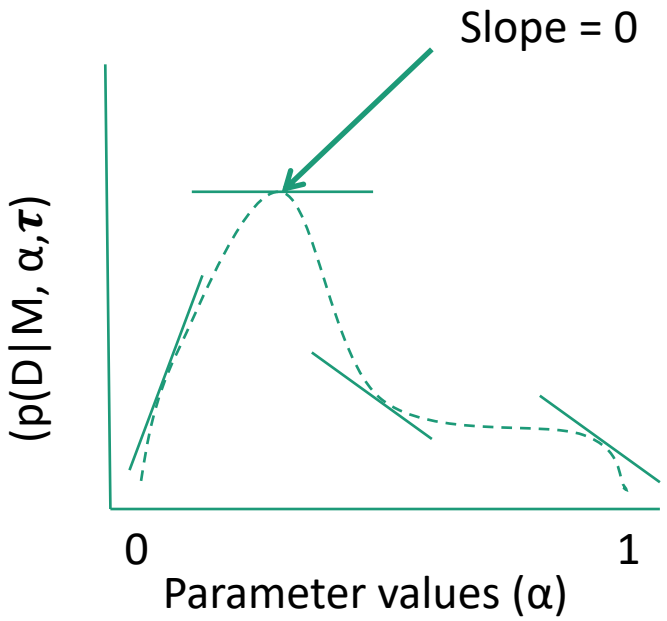
'Now that I can calculate the log-likelihood, how do I find the best parameters?'

- Better approach:
  - Use standardized optimizers (Matlab: fminsearch, fminbound, fmincon, etc…)
    req. optimization toolbox

Idea behind optimizers:

Search through the parameter space without trying all the combinations of parameter values.

For e.g. could use the gradient of the likelihood function to find the optimal value of the likelihood without searching through the entire space of parameter values.



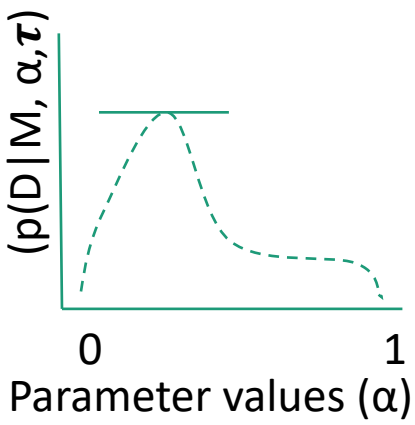
Maximum Likelihood – Implementation

Optimization functions: 'what to watch out for?'

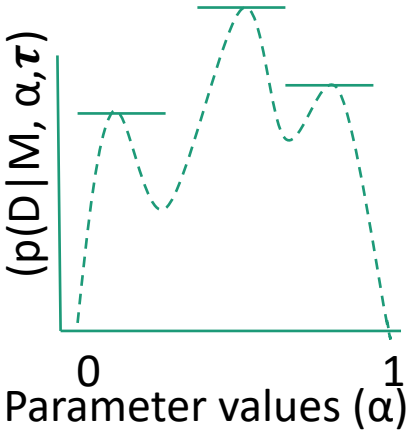
- - Use standardized optimizers (Matlab: fminsearch, fminbound, fmincon, etc…)

- Works well if likelihood is relatively smooth and has only one optima that is the global optima

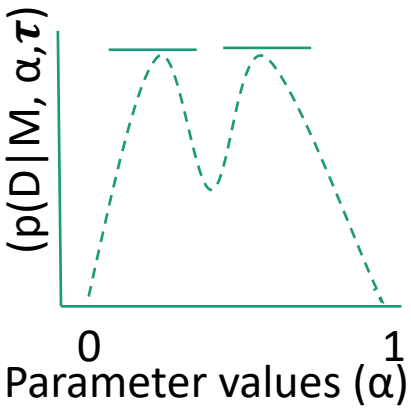
Global = local optima



Multiple local optima,
One global optima



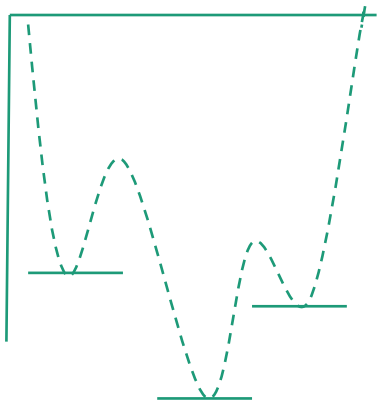
Multiple global optima



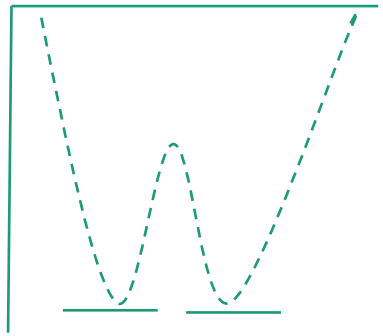
Maximum Likelihood – Implementation

Optimization functions: 'what to watch out for?'

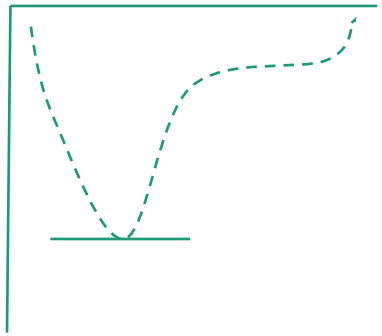
- - Use standardized optimizers (Matlab: fminsearch, fminbound, fmincon, etc…)
- Works well if likelihood is relatively smooth and has only one optima that is the global optima
- **Solution:** Try multiple times the same optimizer with different starting value (random) and take the one with the highest maximum likelihood
- **Attention:** most optimizers attempt to find the local minima, not maxima, so we need to reverse the likelihood function



Global = local optima



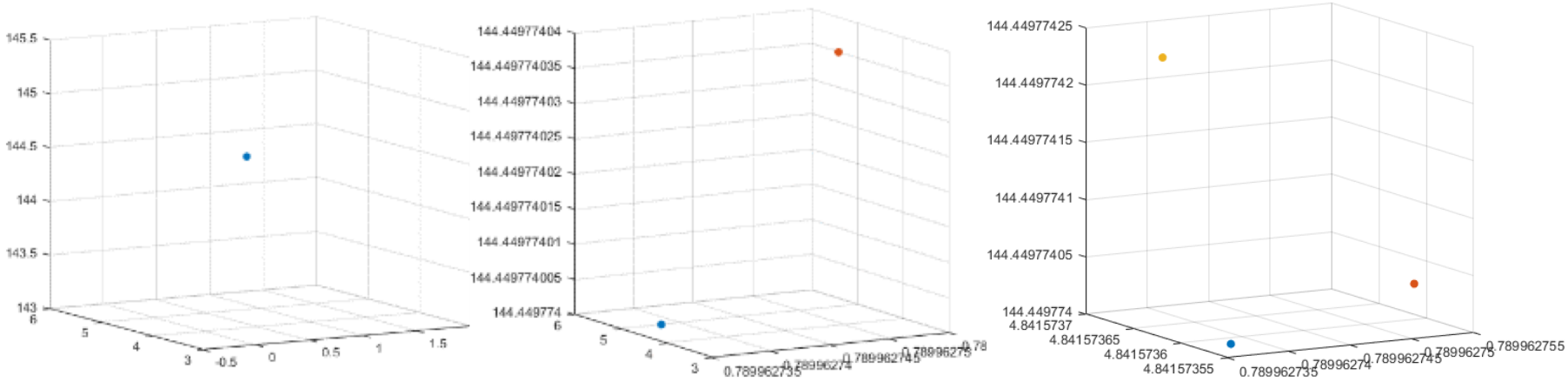
Multiple local optima,
One global optima



Multiple global optima

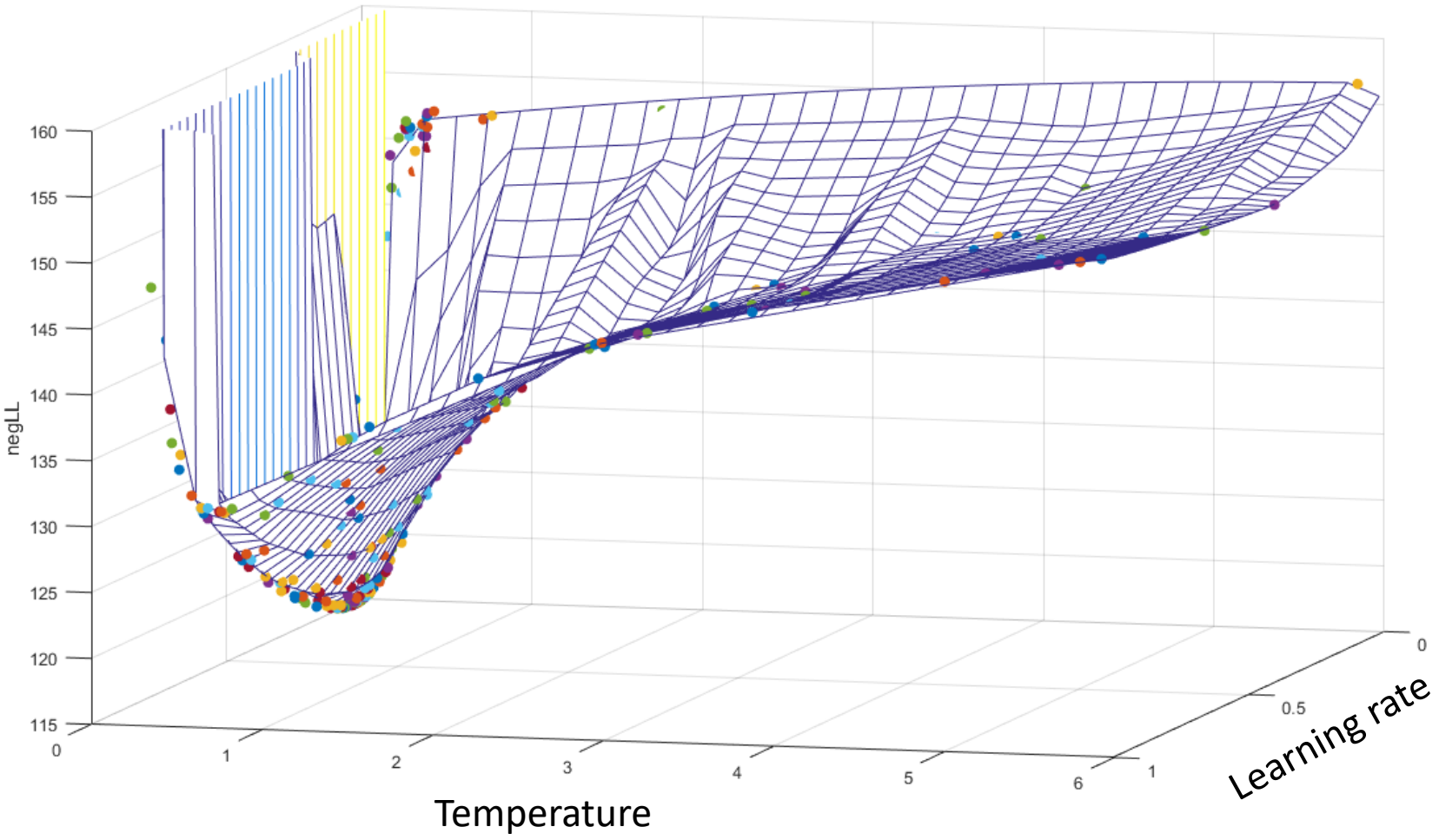
Maximum Likelihood – Implementation

Optimization functions: Example



Maximum Likelihood – Implementation

Optimization functions: Example



Once we have parameters

- Can estimate (per participant):
  - Reward prediction error for each trial
  - Q-values for each trial
  - Probabilities for each trial
  - All parameters
- Can either use these to compare:
  - Different groups (controls vs patients)
  - Different conditions (stationary vs. volatile environment) *
- Regress these parameters to other measures of interest:
  - Does learning rate correlate with questionnaire measures?
  - Where is RPE located in the brain?
  - Etc…

Model comparison

‘What strategy are my participants using to solve the task?’

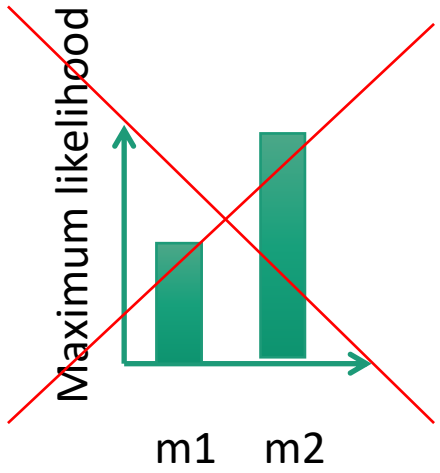
- Why compare models ?
  - Different learning models – different hypotheses
    - Q-learning
    - Win Stay Lose Shift
    - Different learning rates for rewards and punishments
  - Different link functions (observation models) – also different hypotheses
    - Softmax
    - ϵ-greedy
    - ϵ-Softmax
    - Softmax with ‘choice-stickiness’ parameter (choice autocorrelation)

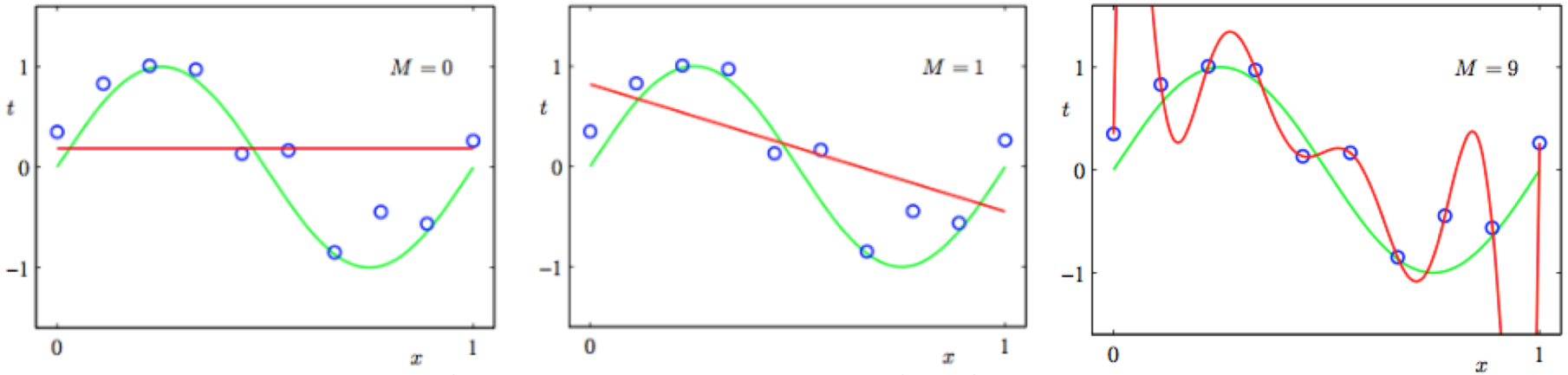
Model comparison

'Why can I not just compare the likelihoods of different models ?'

- Cannot just look at the likelihood function
- Why?
  - Because the more parameters you will have

  in your model the better will be the fit to your data (hence a more complex model will always fit your data better (over-fitting – fitting noise))





Bishop 2008, Pattern Recognition and Machine Learning

Model comparison

'How to compare models ?' – use Bayesian Information Criterion

Ideally want to have ratio between 2 model evidence:

Marginal Likelihood

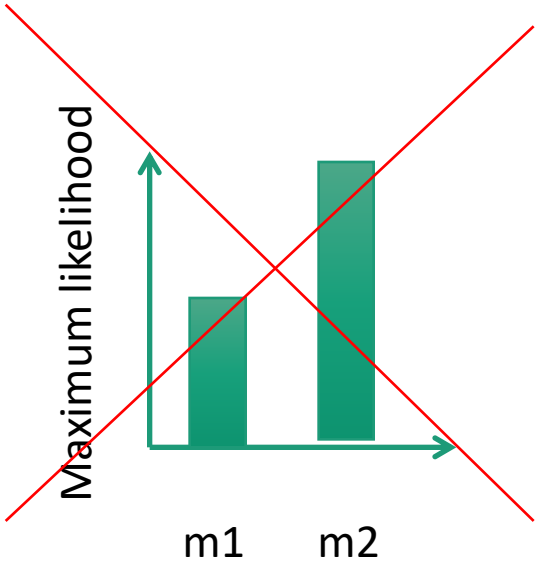
$$Bayes\ Factor = \frac{p(M1 \mid D)}{p(M2 \mid D)} = \frac{p(D \mid M1)p(M1)}{p(D \mid M2)p(M2)}$$

Approximate Bayes factor with Bayesian Information Criterion

BIC(M1)= -2 * log-likelihood M1 + n * log m

Where **n** is the number of free parameters and
m is the number of data points fitted

Calculate the BIC for each subject and sum the BIC of each subject



Model comparison

'What model explains the data best?'

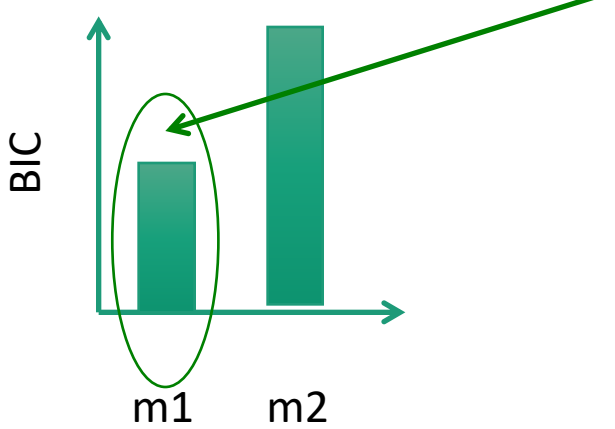
Approximate Bayes factor with Bayesian Information Criterion

$$BIC = -2 * \text{log-likelihood } M1 + n * \log m$$

Where n is the number of free parameters and
m is the number of observations (trials)

AicBic function in matlab:

[aic, bic] = aicbic(LogL, numParams, numObs);



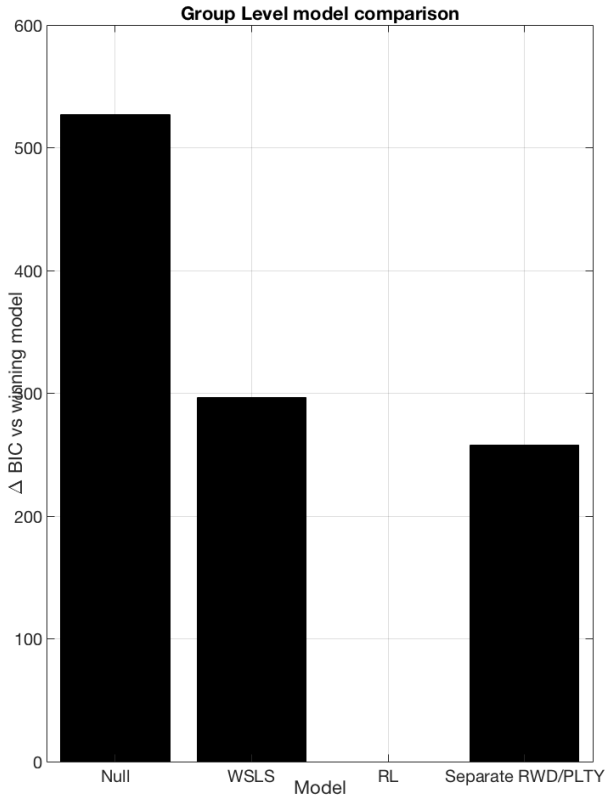
Model with the lowest BIC score is the model to prefer and report as the best model

I.e. model that fits best the data without over-fitting

2 ln *K*	*K*	Strength of evidence
0 to 2	1 to 3	not worth more than a bare mention
2 to 6	3 to 20	positive
6 to 10	20 to 150	strong
>10	>150	very strong

Model Comparison

‘What if I have only one model?’



You should never have just a single model

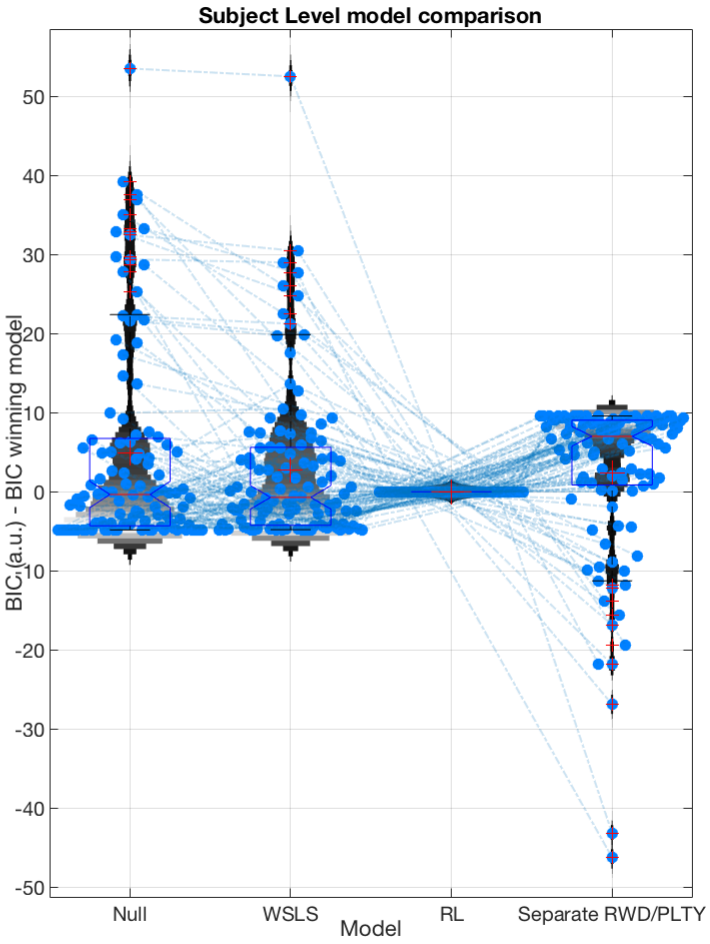
- Good practice:
  - Try to check for a number of models
  - Fit at least:
    - a ‘Null model’ (e.g. learning rate = 0)
    - a simpler version of your existing model
      (e.g. fix some parameters of your model: ‘Win Stay Lose Shift’ learning-rate =1)

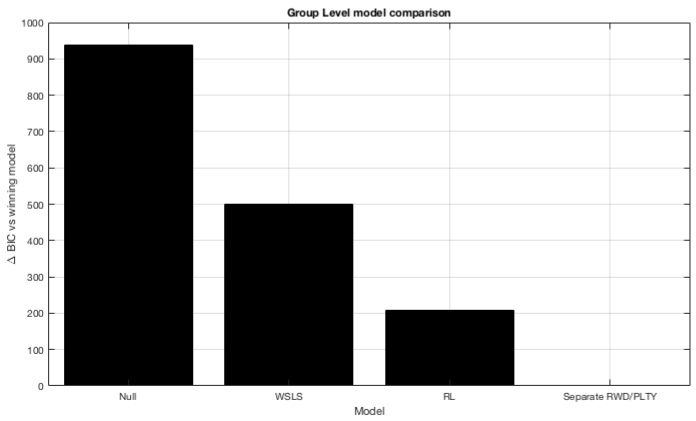
- This is to ensure that:
  1. Your participants are actually doing the task (not performing at random)
  2. your model is better at explaining the participants’ data than some simpler heuristic

Model Comparison

Things to look for!

- Plot your subject-level BIC for each model:
  - Compare whether each participant (data-point) is better modelled by the winning model or a null-model.
  - If the participant is better 'explained'/fitted by the 'Null-model' it may be worth looking at the behaviour of these participants. It could be that they're not doing the task at all and behaving randomly (in which case their parameters are useless).
  - If a large proportion of participants are behaving at random (i.e. favouring the null) this could bias your model selection in favour of a simple model!



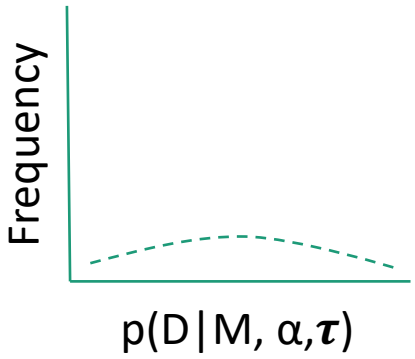


Predictive checks

‘Now that I’ve found the best model I am done, no?’

You may have found a model that best explains your data, but it does not guarantee that it is a good model of how your data was generated.

Imagine the likelihood function of your winning model is very flat. This would suggest that no matter what parameter values we have, it does not have much influence on the fit of the data. This could be indicative of a poor model of behaviour.



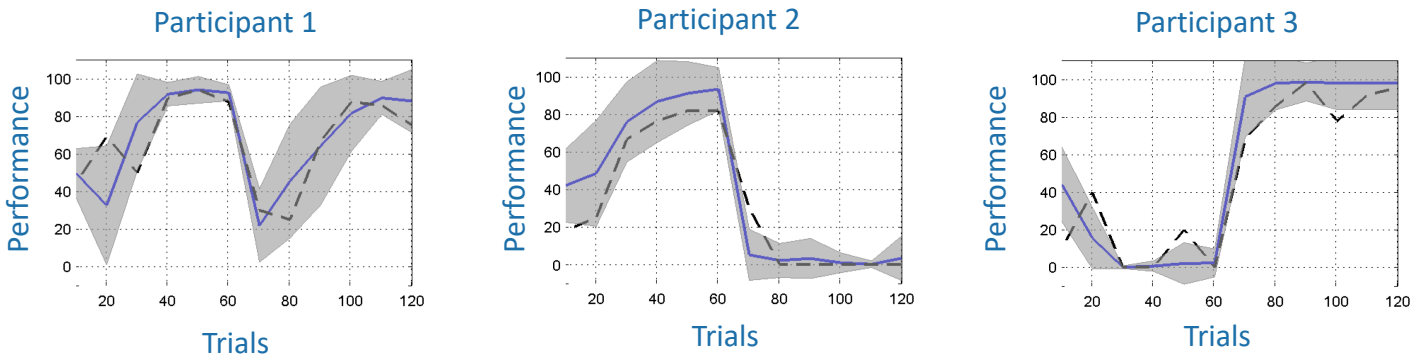
Predictive checks – overlay your model fit to data

Check that your model can recapitulate your data well

Overlay your model fits to your participant data:

- For each subject, use the parameters estimated for that participant
- Run the model with these parameters multiple times.
- Plot your behavioural data, and overlay the average model performance on top of your data

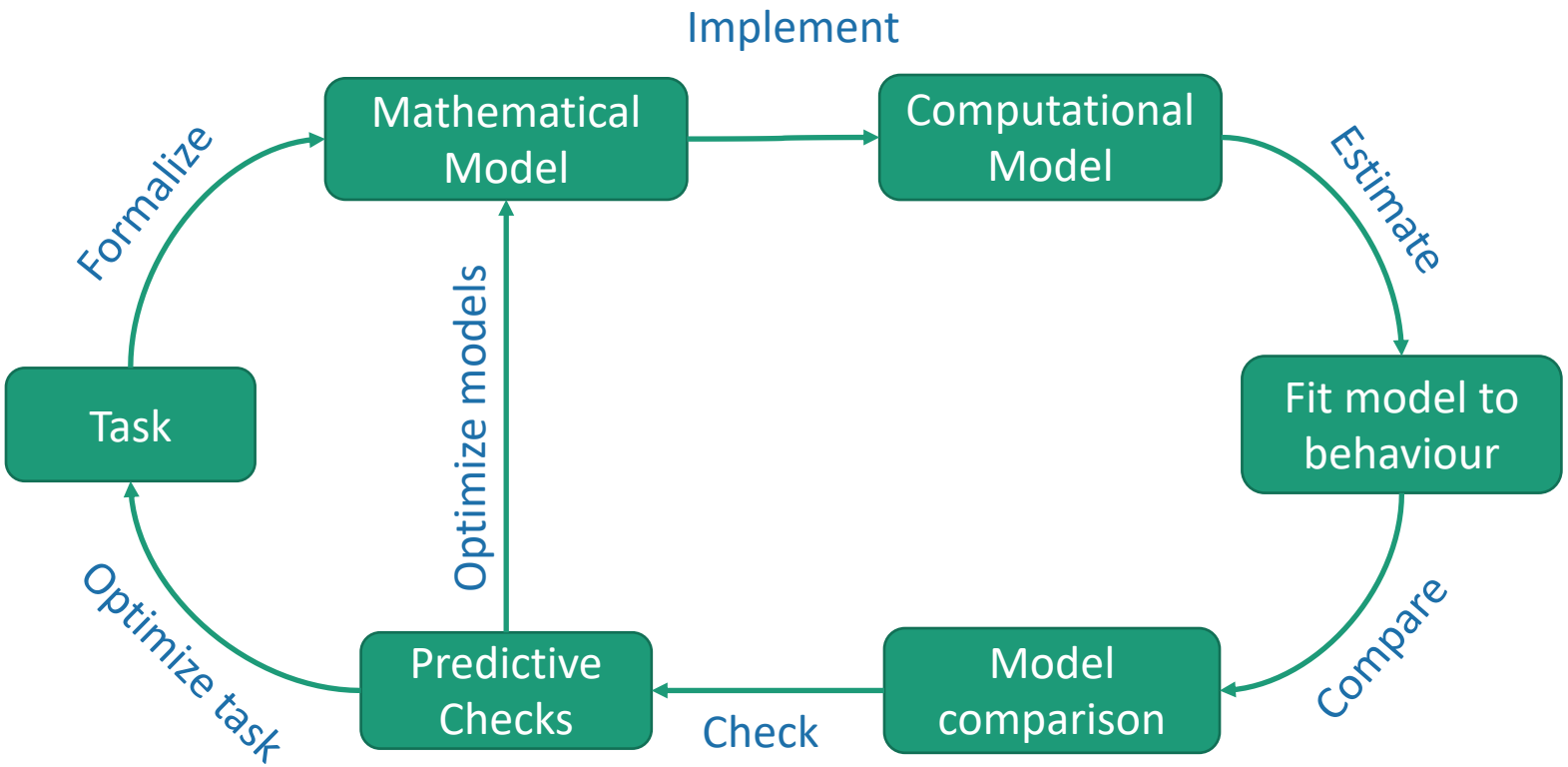
- If the model and data match, it is a good sign!
- If they don't, we need to re-think our models
- If there are areas that could be improved, go back to your model and try to improve it so that it can capture the elements of behaviour it cannot account for at the moment



Rivalan et al., 2009. Elucidating poor decision-making in a Rat Gambling Task

Take home message

Building models and fitting them to data is an iterative process:



Supported by
wellcome trust

Thanks

Prof. J. Roiser *(supervision)*
Prof. P. Dayan *(supervision)*