Reinforcement learning

crash course

Quentin Huys

Translational Neuromodeling Unit, University of Zurich and ETH Zurich
University Hospital of Psychiatry Zurich

Computational Psychiatry Course
Zurich, 1.9.2016
Overview

- Reinforcement learning: rough overview
  - mainly following Sutton & Barto 1998

- Dopamine
  - prediction errors and more

- Fitting behaviour with RL models
  - hierarchical approaches
Setup

After Sutton and Barto 1998

\[
\{a_t\} \leftarrow \text{argmax} \sum_{t=1}^{\infty} r_t
\]
State space

Electric shocks -1

Gold +1
A Markov Decision Problem

\[ s_t \in S \]
\[ a_t \in A \]
\[ \mathcal{T}_{ss'}^a = p(s_{t+1} \mid s_t, a_t) \]
\[ r_t \sim \mathcal{R}(s_{t+1}, a_t, s_t) \]
\[ \pi(a \mid s) = p(a \mid s) \]
A Markov Decision Problem

\[ s_t \in S \]
\[ a_t \in A \]
\[ T_{ss'}^a = p(s_{t+1} | s_t, a_t) \]
\[ r_t \sim R(s_{t+1}, a_t, s_t) \]
\[ \pi(a | s) = p(a | s) \]
Actions

Absorbing state -> max eigenvalue < 1

Noisy: plants, environments, agent
Markovian dynamics

\[ p(s_{t+1} | a_t, s_t, a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \cdots) = p(s_{t+1} | a_t, s_t) \]

Velocity

\[ s' = \text{[position]} \rightarrow s' = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix} \]
A Markov Decision Problem

\[ s_t \in S \]
\[ a_t \in A \]
\[ T_{ss'}^a = p(s_{t+1} | s_t, a_t) \]
\[ r_t \sim R(s_{t+1}, a_t, s_t) \]
\[ \pi(a|s) = p(a|s) \]
A Markov Decision Problem

\[ s_t \in S \]
\[ a_t \in A \]
\[ T_{s,s'}^{a} = p(s_{t+1} | s_t, a_t) \]
\[ r_t \sim \mathcal{R}(s_{t+1}, a_t, s_t) \]
\[ \pi(a | s) = p(a | s) \]
Tall orders

- Aim: maximise total future reward

\[
\sum_{t=1}^{\infty} r_t
\]

- i.e. we have to sum over paths through the future and weigh each by its probability

- Best policy achieves best long-term reward
Exhaustive tree search

Decision 1

Decision 2

\[ w_d \]
Decision tree

\[ \sum_{t=1}^{\infty} r_t \]

8
64
512
...
Policy for this talk

- Pose the problem mathematically
- Policy evaluation
- Policy iteration
- Monte Carlo techniques: experience samples
- TD learning

\[ \text{Policy} \quad \xrightarrow{\text{Evaluate}} \quad \xleftarrow{\text{Update}} \]
Evaluating a policy

- Aim: maximise total future reward
  \[ \sum_{t=1}^{\infty} r_t \]

- To know which is best, evaluate it first

- The policy determines the expected reward from each state

\[ V^\pi(s_1) = \mathbb{E} \left[ \sum_{t=1}^{\infty} r_t | s_1 = 1, a_t \sim \pi \right] \]
Discounting

- Given a policy, each state has an expected value

\[ V^\pi(s_1) = \mathbb{E} \left[ \sum_{t=1}^{\infty} r_t | s_1 = 1, a_t \sim \pi \right] \]

- But: \( \sum_{t=0}^{\infty} r_t = \infty \)

- Episodic \( \sum_{t=0}^{T} r_t < \infty \)

- Discounted
  - infinite horizons \( \sum_{t=0}^{\infty} \gamma^t r_t < \infty \)
  - finite, exponentially distributed horizons

\[ \sum_{t=0}^{T} \gamma^t r_t \sim \frac{1}{\tau} e^{t/\tau} \]
Markov Decision Problems

\[ V^\pi(s_t) = \mathbb{E} \left[ \sum_{t'=1}^{\infty} r_{t'} | s_t = s, \pi \right] \]

\[ = \mathbb{E} [r_1 | s_t = s, \pi] + \mathbb{E} \left[ \sum_{t=2}^{\infty} r_t | s_t = s, \pi \right] \]

\[ = \mathbb{E} [r_1 | s_t = s, \pi] + \mathbb{E} [V^\pi(s_{t+1}) | s_t = s, \pi] \]

This dynamic consistency is key to many solution approaches. It states that the value of a state \( s \) is related to the values of its successor states \( s' \).
Markov Decision Problems

\[ V^\pi(s_t) = \mathbb{E}[r_1 \mid s_t = s, \pi] + \mathbb{E}[V(s_{t+1}), \pi] \]

\[ r_1 \sim \mathcal{R}(s_2, a_1, s_1) \]

\[ \mathbb{E}[r_1 \mid s_t = s, \pi] = \mathbb{E} \left[ \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t)\mathcal{R}(s_{t+1}, a_t, s_t) \right] \]

\[ = \sum_{a_t} p(a_t \mid s_t) \left[ \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t)\mathcal{R}(s_{t+1}, a_t, s_t) \right] \]

\[ = \sum_{a_t} \pi(a_t, s_t) \left[ \sum_{s_{t+1}} \mathcal{T}_s^{a_t} s_{t+1} \mathcal{R}(s_{t+1}, a_t, s_t) \right] \]
Bellman equation

\[
V^\pi(s_t) = \mathbb{E}[r_1 | s_t = s, \pi] + \mathbb{E}[V(s_{t+1}), \pi]
\]

\[
\mathbb{E}[r_1 | s_t, \pi] = \sum_a \pi(a, s_t) \left[ \sum_{s_{t+1}} T_{s_t s_{t+1}}^a R(s_{t+1}, a, s_t) \right]
\]

\[
\mathbb{E}[V^\pi(s_{t+1}), \pi, s_t] = \sum_a \pi(a, s_t) \left[ \sum_{s_{t+1}} T_{s_t s_{t+1}}^a V^\pi(s_{t+1}) \right]
\]

\[
V^\pi(s) = \sum_a \pi(a|s) \left[ \sum_{s'} T_{ss'}^a [R(s', a, s) + V^\pi(s')] \right]
\]
Bellman Equation

\[ V^\pi(s) = \sum_a \pi(a \mid s) \left[ \sum_{s'} T_s^a \left[ R(s', a, s) + V^\pi(s') \right] \right] \]

All future reward from state \( s \) = \( \text{E} \) Immediate reward + All future reward from next state \( s' \)
\[ V^\pi(s) = \sum_a \pi(a|s) \left[ \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V^\pi(s') \right] \right] \]

\textbf{so we can define state-action values as:}
\[ Q(s, a) = \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V(s') \right] \]
\[ = \mathbb{E} \left[ \sum_{t=1}^{\infty} r_t \middle| s, a \right] \]

\textbf{and state values are average state-action values:}
\[ V(s) = \sum_a \pi(a|s) Q(s, a) \]
Bellman Equation

\[ V^\pi(s) = \sum_a \pi(a|s) \left[ \sum_{s'} T_{ss'}^a [R(s', a, s) + V^\pi(s')] \right] \]

- to evaluate a policy, we need to solve the above equation, i.e. find the self-consistent state values

- options for policy evaluation
  - exhaustive tree search - outwards, inwards, depth-first
  - value iteration: iterative updates
  - linear solution in 1 step
  - experience sampling
Solving the Bellman Equation

Option 1: turn it into update equation

\[ V^{k+1}(s) = \sum_a \pi(a, s_t) \left[ \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V^k(s') \right] \right] \]

Option 2: linear solution (w/ absorbing states)

\[ V(s) = \sum_a \pi(a, s_t) \left[ \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V(s') \right] \right] \]

\[ \Rightarrow v = R^\pi + T^\pi v \]

\[ \Rightarrow v^\pi = (I - T^\pi)^{-1} R^\pi \quad O(|S|^3) \]
Policy update

Given the value function for a policy, say via linear solution

\[ V^\pi(s) = \sum_a \pi(a|s) \left( \sum_{s'} T_{ss'}^a [R(s', a, s) + V^\pi(s')] \right) \]

Given the values \( V \) for the policy, we can improve the policy by always choosing the best action:

\[ \pi'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_a Q^\pi(s, a) \\ 0 & \text{else} \end{cases} \]

It is guaranteed to improve:

\[ Q^\pi(s, \pi'(s)) = \max_a Q^\pi(s, a) \geq Q^\pi(s, \pi(s)) = V^\pi(s) \]
**Policy iteration**

**Policy evaluation**

\[ v^\pi = (I - T^\pi)^{-1} R^\pi \]

**Value iteration**

\[ V^*(s) = \max_a \sum_{s'} T_{ss'}^a \left[ R_{ss}^a + V^*(s') \right] \]

**Greedy policy improvement**

\[ \pi(a|s) = \begin{cases} 
1 & \text{if } a = \text{argmax}_a \sum_{s'} T_{ss'}^a \left[ R_{ss}^a + V^{\pi}(s') \right] \\
0 & \text{else}
\end{cases} \]
Model-free solutions

- So far we have assumed knowledge of R and T
  - R and T are the ‘model’ of the world, so we assume full knowledge of the dynamics and rewards in the environment
- What if we don’t know them?
- We can still learn from state-action-reward samples
  - we can learn R and T from them, and use our estimates to solve as above
  - alternatively, we can directly estimate V or Q
Solving the Bellman Equation

Option 3: sampling

\[ V(s) = \sum_a \pi(a, s_t) \left[ \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V(s') \right] \right] \]

this is an expectation over policy and transition samples.

So we can just draw some samples from the policy and the transitions and average over them:

\[ a = \sum_k f(x_k) p(x_k) \]

\[ x^{(i)} \sim p(x) \rightarrow \hat{a} = \frac{1}{N} \sum_i f(x^{(i)}) \]

more about this later...
Learning from samples

A new problem: exploration versus exploitation
Monte Carlo

- **First visit MC**
  - randomly start in all states, generate paths, average for starting state only
  
  \[ V(s) = \frac{1}{N} \sum_i \left\{ \sum_{t'=1}^{T} r_{t'} | s_0 = s \right\} \]

- **More efficient use of samples**
  - Every visit MC
  - Bootstrap: TD
  - Dyna

- **Better samples**
  - on policy versus off policy
  - Stochastic search, UCT...
**Update equation: towards TD**

Bellman equation

\[
V(s) = \sum_a \pi(a, s) \left[ \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V(s') \right] \right]
\]

Not yet converged, so it doesn’t hold:

\[
dV(s) = -V(s) + \sum_a \pi(a, s) \left[ \sum_{s'} T_{ss'}^a \left[ R(s', a, s) + V(s') \right] \right]
\]

And then use this to update

\[
V^{i+1}(s) = V^i(s) + dV(s)
\]
TD learning

\[ dV(s) = -V(s) + \sum_a \pi(a, s) \left[ \sum_{s'} T^{a}_{ss'} [R(s', a, s) + V(s')] \right] \]

\[ a_t \sim \pi(a|s_t) \]
\[ s_{t+1} \sim T^{a_t}_{s_t,s_{t+1}} \]
\[ r_t = R(s_{t+1}, a_t, s_t) \]

\[ \delta_t = -V_{t-1}(s_t) + r_t + V_{t-1}(s_{t+1}) \]

\[ V^{i+1}(s) = V^i(s) + dV(s) \]
\[ V_t(s_t) = V_{t-1}(s_t) + \alpha \delta_t \]
TD learning

\[ a_t \sim \pi(a | s_t) \]
\[ s_{t+1} \sim T_{s_t, s_{t+1}}^{a_t} \]
\[ r_t = R(s_{t+1}, a_t, s_t) \]
\[ \delta_t = -V_t(s_t) + r_t + V_t(s_{t+1}) \]
\[ V_{t+1}(s_t) = V_t(s_t) + \alpha \delta_t \]
Do TD for state-action values instead:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \]

\[ s_t, a_t, r_t, s_{t+1}, a_{t+1} \]

convergence guarantees - will estimate \( Q^\pi(s, a) \)
Learn off-policy

- draw from some policy
- “only” require extensive sampling

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \]

- will estimate \( Q^*(s, a) \)
The effect of bootstrapping

- Markov (every visit)
  - $V(B) = \frac{3}{4}$
  - $V(A) = 0$

- TD
  - $V(B) = \frac{3}{4}$
  - $V(A) = \approx \frac{3}{4}$

Average over various bootstrappings: $TD(\lambda)$

after Sutton and Barto 1998
Conclusion

- Long-term rewards have internal consistency
- This can be exploited for solution
- Exploration and exploitation trade off when sampling
- Clever use of samples can produce fast learning
  - Brain most likely does something like this
Fitting models to behaviour

Quentin Huys

Translational Neuromodeling Unit, University of Zurich and ETH Zurich
University Hospital of Psychiatry Zurich

Computational Psychiatry Course
Zurich, 1.9.2016
Example task

Think of it as four separate two-armed bandit tasks
Analysing behaviour

- **Standard approach:**
  - Decide which feature of the data you care about
  - Run descriptive statistical tests, e.g. ANOVA

- **Many strengths**
- **Weakness**
  - Piecemeal, not holistic / global
  - Descriptive, not generative
  - No internal variables
Models

- **Holistic**
  - Aim to model the process by which the data came about in its “entirety”

- **Generative**
  - They can be run on the task to generate data as if a subject had done the task

- **Inference process**
  - Capture the inference process subjects have to make to perform the task.
  - Do this in sufficient detail to replicate the data.

- **Parameters**
  - replace test statistics
  - their meaning is explicit in the model
Reinforcement learning

Actions

- **Q values “the process”**
  \[ Q_t(a_t, s_t) = Q_{t-1}(a_t, s_t) + \epsilon (r_t - Q_{t-1}(a_t, s_t)) \]

- **Probabilities “link function”**
  \[ p(a_t | s_t, h_t, \beta) = p(a_t | Q(a_t, s_t), \beta) \]
  \[ = \frac{e^{\beta Q(a_t, s_t)}}{\sum_{a'} e^{\beta Q(a', s_t)}} \]

- **Features:**
  \[ p(a_t | s_t) \propto Q(a_t, s_t) \]
  \[ 0 \leq p(a) \leq 1 \]

- links learning process and observations
  - choices, RTs, or any other data
Fitting models I

- Maximum likelihood (ML) parameters

\[ \hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta) \]

- where the likelihood of all choices is:

\[
\mathcal{L}(\theta) = \log p(\{a_t\}_{t=1}^T | \{s_t\}_{t=1}^T, \{r_t\}_{t=1}^T, \underbrace{\theta, \beta, \epsilon}_{\text{parameters}}) \\
= \log p(\{a_t\}_{t=1}^T | \{Q(s_t, a_t; \epsilon)\}_{t=1}^T, \beta) \\
= \log \prod_{t=1}^T p(a_t | Q(s_t, a_t; \epsilon), \beta) \\
= \sum_{t=1}^T \log p(a_t | Q(s_t, a_t; \epsilon), \beta)
\]
Fitting models II

- No closed form
- Use your favourite method
  - gradients
  - fminunc / fmincon...
- Gradients for RW model

\[
\frac{d\mathcal{L}(\theta)}{d\theta} = \frac{d}{d\theta} \sum_t \log p(a_t|Q_t(a_t, s_t; \epsilon), \beta) \\
= \sum_t \frac{d}{d\theta} \beta Q_t(a_t, s_t; \epsilon) - \sum_{a'} p(a'|Q_t(a', s_t; \epsilon), \beta) \frac{d}{d\theta} \beta Q_t(a', s_t; \epsilon)
\]

\[
\frac{dQ_t(a_t, s_t; \epsilon)}{d\epsilon} = (1 - \epsilon) \frac{dQ_{t-1}(a_t, s_t; \epsilon)}{d\epsilon} + (r_t - Q_{t-1}(a_t, s_t; \epsilon))
\]
**Little tricks**

- **Transform your variables**
  \[
  \beta = e^{\beta'} \\
  \Rightarrow \beta' = \log(\beta) \\
  \epsilon = \frac{1}{1 + e^{-\epsilon'}} \\
  \Rightarrow \epsilon' = \log \left( \frac{\epsilon}{1 - \epsilon} \right)
  \]

- **Avoid over/underflow**
  \[
  y(a) = \beta Q(a) \\
  y_m = \max_a y(a) \\
  p = \frac{e^{y(a)}}{\sum_b e^{y(b)}} = \frac{e^{y(a)} - y_m}{\sum_b e^{y(b)} - y_m}
  \]
ML characteristics

- ML is asymptotically consistent, but variance high
  - 10-armed bandit, infer beta and epsilon

Mathematical notation:

\[ L(\beta = 10) \approx L(\beta = 100) \]

Graph: 200 trials, 1 stimulus, 10 actions, learning rate = .05, beta=2
Priors

Not so smooth

Smooth

Reinforcement learning

CPC Zurich 1/9/16

Quentin Huys, ETHZ / PUK
Maximum a posteriori estimate

\[ P(\theta) = p(\theta|a_1...T) = \frac{p(a_1...T|\theta)p(\theta)}{\int d\theta p(\theta|a_1...T)p(\theta)} \]

\[ \log P(\theta) = \sum_{t=1}^{T} \log p(a_t|\theta) + \log p(\theta) + \text{const.} \]

\[ \frac{\log P(\theta)}{d\alpha} = \frac{\log \mathcal{L}(\theta)}{d\alpha} + \frac{d p(\theta)}{d\theta} \]

If likelihood is strong, prior will have little effect

• mainly has influence on poorly constrained parameters
• if a parameter is strongly constrained to be outside the typical range of the prior, then it will win over the prior
Maximum a posteriori estimate

200 trials, 1 stimulus, 10 actions, learning rate = .05, beta=2
m_{beta}=0, m_{eps}=-3, n=1
But

What prior parameters should I use?
Hierarchical estimation - “random” effects

- **Fixed effect**
  - conflates within- and between-subject variability

- **Average behaviour**
  - disregards between-subject variability
  - need to adapt model

- **Summary statistic**
  - treat parameters as random variable, one for each subject
  - overestimates group variance as ML estimates noisy

- **Random effects**
  - prior mean = group mean

\[
p(A_i | \mu_\theta, \sigma_\theta) = \int d\theta_i p(A_i | \theta_i) p(\theta_i | \mu_\theta, \sigma_\theta) \
\]
Random effects

- See subjects as drawn from group

- Fixed models
  - all the same: fixed effect wrt model
  - parametrically nested
    \[ Q(a, s) = \omega_1 Q^1(a, s) + \omega_2 Q^2(a, s) \]
    - assumes within-subject mixture, rather than mixture of perfect types

- Random effects in models
Estimating the hyperparameters

- Effectively we now want to do gradient ascent on:
  \[
  \frac{d}{d\zeta} p(A|\zeta)
  \]

- But this contains an integral over individual parameters:
  \[
  p(A|\zeta) = \int d\theta p(A|\theta) p(\theta|\zeta)
  \]

- So we need to:
  \[
  \hat{\zeta} = \arg\max_{\zeta} p(A|\zeta)
  = \arg\max_{\zeta} \int d\theta p(A|\theta) p(\theta|\zeta)
  \]
\[ \hat{\zeta} = \arg\max_{\zeta} p(A|\zeta) \]
\[ = \arg\max_{\zeta} \int d\theta p(A|\theta) p(\theta|\zeta) \]

- analytical - rare
- brute force - for simple problems
- Expectation Maximisation - approximate, easy
- Variational Bayes
- Sampling / MCMC
Expectation Maximisation

\[
\log p(A | \zeta) = \log \int d\theta \, p(A, \theta | \zeta)
\]

\[
= \log \int d\theta \, q(\theta) \frac{p(A, \theta | \zeta)}{q(\theta)}
\]

\[
\geq \int d\theta \, q(\theta) \log \frac{p(A, \theta | \zeta)}{q(\theta)}
\]

\[k^{\text{th}}\text{ E step: } q^{(k+1)}(\theta) \leftarrow p(\theta | A, \zeta^{(k)})\]

\[k^{\text{th}}\text{ M step: } \zeta^{(k+1)} \leftarrow \text{argmax}_{\zeta} \int d\theta \, q(\theta) \log p(A, \theta | \zeta)\]

Iterate between

- Estimating MAP parameters given prior parameters
- Estimating prior parameters from MAP parameters
Bayesian Information Criterion

- Laplace's approximation (saddle-point method)

\[ \int dx \, f(x) \approx f^*(x_0) \sqrt{2\pi \sigma^2} \]

Just a Gaussian
EM with Laplace approximation

- **E step**: 
  \[ q^{(k+1)}(\theta) \leftarrow p(\theta | A, \zeta^{(k)}) \]
  - only need sufficient statistics to perform M step
  - Approximate \( p(\theta | A, \zeta^{(k)}) \sim N(m_k, S_k) \)
  - and hence:

\[
\begin{align*}
E \text{ step:} & \quad q_k(\theta) = \mathcal{N}(m_k, S_k) \\
& \quad m_k \leftarrow \text{argmax}_\theta p(a_k | \theta) p(\theta | \zeta^{(i)}) \\
& \quad S_k^{-1} \leftarrow \left. \frac{\partial^2 p(a^k | \theta) p(\theta | \zeta^{(i)})}{\partial \theta^2} \right|_{\theta = m_k}
\end{align*}
\]

Just what we had before: MAP inference given some prior parameters
EM with Laplace approximation

- Updates

M step: $\zeta^{(i+1)}_\mu = \frac{1}{K} \sum_k m_k$

$\zeta^{(i+1)}_{\nu^2} = \frac{1}{N} \sum_i \left[ (m_k)^2 + S_k \right] - \left( \zeta^{(i+1)}_\mu \right)^2$

Prior mean = mean of MAP estimates

Prior variance depends on inverse Hessian $S$ and variance of MAP estimates

- And now iterate until convergence
Parameter recovery

Rescorla-Wagner Model

2-step model

Simulation 1
truly uncorrelated, no GLM

Simulation 4
truly correlated, no GLM

A \( \beta' \) correlation true–inferred

B \( \alpha' \) correlation true–inferred

C \( \beta_1 \) correlation true–inferred

D \( \omega \) correlation true–inferred

E

F

G

H

\( N_{sj} = 10 \)
\( N_{sj} = 20 \)
\( N_{sj} = 50 \)
\( N_{sj} = 100 \)

EM-MAP
MAP0
ML

\( T \)

\( T \)

\( T \)

\( T \)
Correlations

**RW**

**A** Correlation ML $\beta'$ and $\alpha'$

**B** Correlation ML $\beta_1'$ and $\omega'$

**2-step**

**C** Correlation MAP $\beta''$ and $\alpha''$

**D** Correlation MAP $\beta_1''$ and $\omega''$

**E** Prior covariance MAP $\beta''$ and $\alpha''$

**F** Prior covariance MAP $\beta_1''$ and $\omega''$
Are parameters ok for correlations?

- Individual subject parameter estimates NO LONGER INDEPENDENT!
  - Change group -> change parameter estimates
- compare different params
  - if different priors
- correlations, t-tests
  - within same prior ok

![Graph showing distribution of p-values](image)
» So far
  - infer individual parameters
  - apply standard tests

» Alternative
  - View as variation across group
  - Specific - more powerful?

\[
\mu^i_\theta = \mu^\text{Group}_\theta + \beta \psi_i
\]
Group-level regressor

GLM

- Group-level regressor

**Figure 5**: GLM performance. A-D show the estimates of the regression coefficients when there was no true correlation (A,C) and when there was a correlation of size 1 (B,D) in the RW (A,B) and the 2-step models (C,D). Error bars are derived from the finite difference Hessians around the estimates. E,F show the p-value distribution obtained from the Hessians around the regression coefficients for the case were the true parameters and ψ were uncorrelated. For the RW model (E), this correctly yields a flat distribution. This is not the case for the ω parameter of the 2-step model (F). Hence, we cannot generally use (at least this version of) the estimated errors around the GLM regression coefficients to decide whether a correlation was significant or not.

An alternative is to perform model comparison, comparing a model without regressor (containing just the group means for each parameter) to a model including the GLM regressor. This is probably the most appropriate Bayesian approach. Estimating Bayes Factors is however difficult, and we hence examine how an approximation to this, the integrated BIC (iBIC) performs. The lower this is, the more parsimonious and hence “better” the model. For the RW data, model comparison using iBIC yields low false positive rates (Figure 6A), but very high false negative rates (Figure 6B). For the 2-step model, false positive and false negative rates appear to both be unacceptably high (Figure 6C,D).

Finally, the EM-MAP algorithm does provide approximate uncertainties for each individual parameter estimate. These uncertainties can be used as weights in a regression analysis to attempt to achieve some...
Fitting - how to

- Write your likelihood function
  - matlab examples attached with emfit.m
  - don’t do 20 ML fits!
  - pass it into emfit.m or julia version
    - www.quentinhuys.com/pub/emfit_151110.zip
  - validate: generate data with fitted params
    - compare, have a look, does it look right?
    - re-fit - is it stable?
  - model comparison
  - now: look at parameters, do correlations etc.

- Future:
  - GLM
  - full random effects over models and parameters jointly?
    - Daniel Schad
Hierarchical / random effects models

- **Advantages**
  - Accurate group-level mean and variance
  - Outliers due to weak likelihood are regularised
  - Strong outliers are not
  - Useful for model selection

- **Disadvantages**
  - Individual estimates $\theta_i$ depend on other data, i.e. on $A_{j \neq i}$ and therefore need to be careful in interpreting these as summary statistics
  - More involved; less transparent

- **Psychiatry**
  - Groups often not well defined, covariates better

- **fMRI**
  - Shrink variance of ML estimates - fixed effects better still?
How does it do?
Overfitting
Model comparison

- A fit by itself is not meaningful
- Generative test
  - qualitative
- Comparisons
  - vs random
  - vs other model -> test specific hypotheses and isolate particular effects in a generative setting
Model comparison

» Averaged over its parameter settings, how well does the model fit the data?

\[ p(A|\mathcal{M}) = \int d\theta \ p(A|\theta) \ p(\theta|\mathcal{M}) \]

» Model comparison: Bayes factors

\[ BF = \frac{p(A|\mathcal{M}_1)}{p(A|\mathcal{M}_2)} \]

» Problem:
  • integral rarely solvable
  • approximation: Laplace, sampling, variational...
Why integrals? The God Almighty test

\[
\frac{1}{N} \left( p(X|\theta_1) + p(X|\theta_2) + \cdots \right)
\]

These two factors fight it out
Model complexity vs model fit
Group-level BIC

\[
\log p(A|M) = \int d\zeta p(A|\zeta) p(\zeta|M) \\
\approx -\frac{1}{2} \text{BIC}_{\text{int}} \\
= \log \hat{p}(A|\hat{\zeta}^{ML}) - \frac{1}{2} |M| \log(|A|)
\]

- Very simple
  - 1) EM to estimate group prior mean & variance
    - simply done using fminunc, which provides Hessians
  - 2) Sample from estimated priors
  - 3) Average
How does it do?

Fitted by EM... too nice?
Group Model selection

Integrate out your parameters
Model comparison: overfitting?

Note: same number of parameters
Behavourial data modelling

- **Are no panacea**
  - statistics about specific aspects of decision machinery
  - only account for part of the variance

- **Model needs to match experiment**
  - ensure subjects actually do the task the way you wrote it in the model
  - model comparison

- **Model = Quantitative hypothesis**
  - strong test
  - need to compare *models*, not *parameters*
  - includes all consequences of a hypothesis for choice
Thanks

- Peter Dayan
- Daniel Schad
- Nathaniel Daw

- SNSF
- DFG