Inverse Preference Elicitation for Sequential Decision Making

Dan Lizotte
Postdoctoral Fellow, University of Michigan Statistics

Joint work with Michael Bowling†, Susan A. Murphy‡
†University of Alberta, ‡University of Michigan

INFORMS Healthcare
Montreal, QC
19 June 2011

Evidence-Based Medicine

- "It's about integrating individual clinical expertise and the best external evidence."[1]

[1] D.L. Sackett et al., "Evidence-based Medicine: What It Is and What It Isn't" (Editorial), British Medical Journal 312, no. 7023 (1996): 71-72.

How do we provide the "best external evidence?"

- One approach: Collect and analyze data, recommend treatments

How do we provide the “best external evidence?”

- One approach: Collect and analyze data, recommend treatments
    - Input: Patient state, Output: Recommended treatment
        - “Personalized medicine”

How do we provide the "best external evidence?"

- One approach: Collect and analyze data, recommend treatments
    - Input: Patient state, Output: Recommended treatment
        - "Personalized medicine"
- This does not integrate with individual clinical expertise

How do we provide the "best external evidence?"

- One approach: Collect and analyze data, recommend treatments
  - Input: Patient state, Output: Recommended treatment
    - "Personalized medicine"
- This does not integrate with individual clinical expertise
  - Input: Patient state, Output: Salient information about available treatments that reflects the evidence in the data.

How do we provide the "best external evidence?"

- One approach: Collect and analyze data, recommend treatments
  - Input: Patient state, Output: Recommended treatment
    - "Personalized medicine"
- This does not integrate with individual clinical expertise
  - Input: Patient state, Output: Salient information about available treatments that reflects the evidence in the data.
- Approach: Modify methods and algorithms that recommend a single treatment to produce richer information about available treatments

Example: Reinforcement Learning for Chronic Disease Management

- Chronic diseases are *managed*, not cured
  - Major depressive disorder
  - Schizophrenia
  - ...

- Treatment decisions should be:
- Personalized
  - Current treatment is chosen based on current patient state
- Non-myopic
  - Current treatment is chosen conditioned on future treatment strategy

- Reinforcement Learning (RL) methods can be used to learn a personalized, non-myopic treatment policy from data...
- ...but almost all current RL methods recommend a single treatment.

Example: Reinforcement Learning for Chronic Disease Management

- Chronic diseases are *managed*, not cured
  - Major depressive disorder
  - Schizophrenia
  - ...
- Treatment decisions should be:
- Personalized
  - Current treatment is chosen based on current patient state
- Non-myopic
  - Current treatment is chosen conditioned on future treatment strategy
- Reinforcement Learning (RL) methods can be used to learn a personalized, non-myopic treatment policy from data...
- ...but almost all current RL methods recommend a single treatment.

Example: Reinforcement Learning for Chronic Disease Management

- Chronic diseases are *managed*, not cured
  - Major depressive disorder
  - Schizophrenia
  - ...
- Treatment decisions should be:
- Personalized
  - Current treatment is chosen based on current patient state
- Non-myopic
  - Current treatment is chosen conditioned on future treatment strategy
- Reinforcement Learning (RL) methods can be used to learn a personalized, non-myopic treatment policy from data...
- ...but almost all current RL methods recommend a single treatment.

Example: Reinforcement Learning for Chronic Disease Management

- Chronic diseases are *managed*, not cured
  - Major depressive disorder
  - Schizophrenia
  - ...
- Treatment decisions should be:
- Personalized
  - Current treatment is chosen based on current patient state
- Non-myopic
  - Current treatment is chosen conditioned on future treatment strategy
- Reinforcement Learning (RL) methods can be used to learn a personalized, non-myopic treatment policy from data...
- ...but almost all current RL methods recommend a single treatment.

Preferences in Schizophrenia Treatment

- Many treatments available for managing schizophrenia (dozens)
- Consider two important objectives or rewards:
  - Symptom reduction, weight control
- No treatment is best by both measures
- Different doctors and patients have very different preferences about relative importance of rewards, and preference information is absent from large schizophrenia datasets
- Recommending a single treatment based on available data is not appropriate.

The Inverse Preference Elicitation Project

- How can we provide salient information about available treatments that is non-myopic and that accommodates these preferences?

1. Augment Q-Learning to allow for different reward preferences
   - Formalize preferences as a multi-objective optimization problem
2. Develop an algorithm tailored to randomized trial data that provides information for each treatment for *all* preferences simultaneously

Example: Decision Aid for Choosing Antipsychotics

- Possible decision aid:

Initial Symptoms	Preference: Symptom Relief		Preference: Weight Control	
	Strong	Mild	Mild	Strong
Good	Olan	Olan or Zip	Zip	Zip
Moderate	Olan	Olan or Zip	Zip	Zip
Bad	Olan	Olan	Risp or Zip	Zip

Olan = Olanzapine, Zip = Ziprasidone, Risp = Risperidone

- This is harder than it looks

Q-Learning - Scalar Reward, Two Time Points

- Randomized trial data: (S_1, A_1, S_2, A_2, R) for each individual
  - $S_t \in \mathcal{S}_t$ - "State" - Patient features (prior treatments, test results, ...)
  - $A_t \in \mathcal{A}_t$ - "Action" - Treatment assigned by exploration policy
  - $R \in \mathbb{R}$ - "Reward" - Scalar clinical outcome, depends on (S_2, A_2)

- Want to find π^* that produces maximal expected reward

- A "policy" $\pi = \{\pi_1, \pi_2\}$ chooses actions given state
  - $\pi_t : \mathcal{S}_t \to \mathcal{A}_t$
  - π_1 influences distribution of S_2 by choosing A_1
  - π_2 influences distribution of R by choosing A_2

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \text{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \text{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \text{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \text{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \text{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \text{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \text{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \text{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \text{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \text{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \operatorname{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \operatorname{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R|S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \mathrm{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \mathrm{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning - Dynamic Programming

- Q-Learning is Dynamic Programming. Determine π_2^*, then π_1^*.

Time 2

- Define $Q_2(s_2, a_2) = E[R | S_2 = s_2, A_2 = a_2]$
  - For state s_2, this is *quality* of each $a_2 \in \mathcal{A}_2$.
- $\pi_2^*(s_2) = \text{argmax}_{a_2} Q_2(s_2, a_2)$
- $V_2^{\pi_2^*}(s_2) = \max_{a_2} Q_2(s_2, a_2)$ the *value* of being in s_2

Time 1

- Define $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2) | S_1 = s_1, A_1 = a_1]$
  - For state s_1, this is *quality* of each $a_1 \in \mathcal{A}_1$.
- $\pi_1^*(s_1) = \text{argmax}_{a_1} Q_1(s_1, a_1)$

- Q-functions are conditional expectations

Q-Learning from Data: Repeated Regression

- Estimate each Q_t by linear regression on features $\phi_t(s_t, a_t)$

Time 2: $Q_2(s_2, a_2) = E_R[R|S_2 = s_2, A_2 = a_2]$

- Regress R on features $\phi_2(S_2, A_2)$ [R might be symptom reduction] to obtain $\hat{Q}_2(s_2, a_2; \hat{\beta}_2) = \hat{\beta}_2^\top \phi_2(s_2, a_2)$
- Set $\hat{\pi}_2^* = \text{argmax}_{a_2} \hat{Q}_2(s_2, a_2)$, $\quad \hat{V}_2^{\hat{\pi}_2^*}(s_2) = \max_{a_2} \hat{Q}_2(s_2, a_2)$

Time 1: $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$

- Regress $\hat{V}_2^{\hat{\pi}_2^*}(s_2)$ on features $\phi_1(S_1, A_1)$ to obtain $\hat{Q}_1(s_1, a_1; \hat{\beta}_1) = \hat{\beta}_1^\top \phi_1(s_1, a_1)$
- $\hat{\pi}_1^*(s_1) = \text{argmax}_{a_1} \hat{Q}_1(s_1, a_1)$

Q-Learning from Data: Repeated Regression

- Estimate each Q_t by linear regression on features $\phi_t(s_t, a_t)$

Time 2: $Q_2(s_2, a_2) = E_R[R|S_2 = s_2, A_2 = a_2]$

- Regress R on features $\phi_2(S_2, A_2)$ [R might be symptom reduction] to obtain $\hat{Q}_2(s_2, a_2; \hat{\beta}_2) = \hat{\beta}_2^\mathsf{T} \phi_2(s_2, a_2)$
- Set $\hat{\pi}_2^* = \text{argmax}_{a_2} \hat{Q}_2(s_2, a_2)$, $\quad \hat{V}_2^{\hat{\pi}_2^*}(s_2) = \max_{a_2} \hat{Q}_2(s_2, a_2)$

Time 1: $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$

- Regress $\hat{V}_2^{\hat{\pi}_2^*}(s_2)$ on features $\phi_1(S_1, A_1)$ to obtain $\hat{Q}_1(s_1, a_1; \hat{\beta}_1) = \hat{\beta}_1^\mathsf{T} \phi_1(s_1, a_1)$
- $\hat{\pi}_1^*(s_1) = \text{argmax}_{a_1} \hat{Q}_1(s_1, a_1)$

Q-Learning from Data: Repeated Regression

- Estimate each Q_t by linear regression on features $\phi_t(s_t, a_t)$

Time 2: $Q_2(s_2, a_2) = E_R[R|S_2 = s_2, A_2 = a_2]$

- Regress R on features $\phi_2(S_2, A_2)$ [R might be symptom reduction] to obtain $\hat{Q}_2(s_2, a_2; \hat{\beta}_2) = \hat{\beta}_2^\mathsf{T}\phi_2(s_2, a_2)$
- Set $\hat{\pi}_2^* = \text{argmax}_{a_2} \hat{Q}_2(s_2, a_2)$, $\quad \hat{V}_2^{\hat{\pi}_2^*}(s_2) = \max_{a_2} \hat{Q}_2(s_2, a_2)$

Time 1: $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$

- Regress $\hat{V}_2^{\hat{\pi}_2^*}(s_2)$ on features $\phi_1(S_1, A_1)$ to obtain $\hat{Q}_1(s_1, a_1; \hat{\beta}_1) = \hat{\beta}_1^\mathsf{T}\phi_1(s_1, a_1)$
- $\hat{\pi}_1^*(s_1) = \text{argmax}_{a_1} \hat{Q}_1(s_1, a_1)$

Q-Learning from Data: Repeated Regression

- Estimate each Q_t by linear regression on features $\phi_t(s_t, a_t)$

Time 2: $Q_2(s_2, a_2) = E_R[R|S_2 = s_2, A_2 = a_2]$

- Regress R on features $\phi_2(S_2, A_2)$ [R might be symptom reduction]
  to obtain $\hat{Q}_2(s_2, a_2; \hat{\beta}_2) = \hat{\beta}_2^\top \phi_2(s_2, a_2)$
- Set $\hat{\pi}_2^* = \operatorname{argmax}_{a_2} \hat{Q}_2(s_2, a_2)$, $\quad \hat{V}_2^{\hat{\pi}_2^*}(s_2) = \max_{a_2} \hat{Q}_2(s_2, a_2)$

Time 1: $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$

- Regress $\hat{V}_2^{\hat{\pi}_2^*}(s_2)$ on features $\phi_1(S_1, A_1)$
  to obtain $\hat{Q}_1(s_1, a_1; \hat{\beta}_1) = \hat{\beta}_1^\top \phi_1(s_1, a_1)$
- $\hat{\pi}_1^*(s_1) = \operatorname{argmax}_{a_1} \hat{Q}_1(s_1, a_1)$

Q-Learning from Data: Repeated Regression

- Estimate each Q_t by linear regression on features $\phi_t(s_t, a_t)$

Time 2: $Q_2(s_2, a_2) = E_R[R|S_2 = s_2, A_2 = a_2]$

- Regress R on features $\phi_2(S_2, A_2)$ [R might be symptom reduction] to obtain $\hat{Q}_2(s_2, a_2; \hat{\beta}_2) = \hat{\beta}_2^\top \phi_2(s_2, a_2)$
- Set $\hat{\pi}_2^* = \operatorname{argmax}_{a_2} \hat{Q}_2(s_2, a_2)$, $\quad \hat{V}_2^{\hat{\pi}_2^*}(s_2) = \max_{a_2} \hat{Q}_2(s_2, a_2)$

Time 1: $Q_1(s_1, a_1) = E_{S_2}[V_2^{\pi_2^*}(S_2)|S_1 = s_1, A_1 = a_1]$

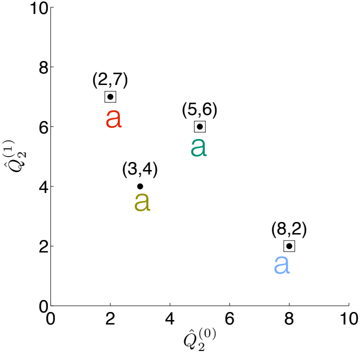
- Regress $\hat{V}_2^{\hat{\pi}_2^*}(s_2)$ on features $\phi_1(S_1, A_1)$ to obtain $\hat{Q}_1(s_1, a_1; \hat{\beta}_1) = \hat{\beta}_1^\top \phi_1(s_1, a_1)$
- $\hat{\pi}_1^*(s_1) = \operatorname{argmax}_{a_1} \hat{Q}_1(s_1, a_1)$

Q-Learning - Beyond Scalar Rewards and Values

- Learned policy $\hat{\pi}_t^*(s_t) = \text{argmax}_{a_t} \hat{Q}_t(s_t, a_t)$ is constructed to maximize long-term expected reward, i.e. is *non-myopic*
- Dynamic programming “trick” is to maximize over a_2 *first*
  - Key step: Set $\hat{V}_2^{\hat{\pi}_2^*}(s_2) = \max_{a_2} \hat{Q}_2(s_2, a_2)$
  - Only makes sense if R is scalar
- What if there is more than one R of interest?

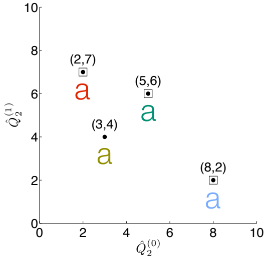
Time 2 Policy - Two Rewards

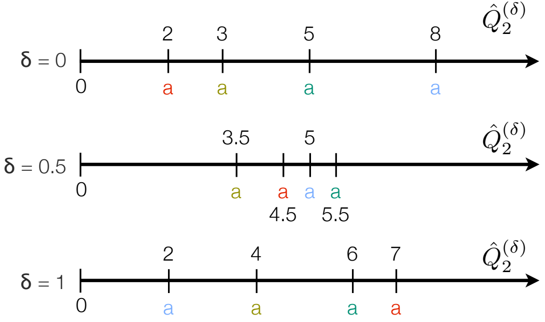
- Suppose two rewards $R^{(0)}$ and $R^{(1)}$ are of interest, e.g. symptom reduction and weight control
- Below, $(\hat{Q}_2^{(0)}, \hat{Q}_2^{(1)})$ for patient with $S_2 = s_2$, four different actions
- What should $\hat{\pi}_2^*(s_2)$ be?



Formalizing Preference

- Define a scalar reward $R^{(\delta)} = (1-\delta)R^{(0)} + \delta R^{(1)}$
- $0 \leq \delta \leq 1$
- Proceed as before to get $\hat{Q}_2^{(\delta)}$





- δ represents "How much do I care about $R^{(1)}$?"

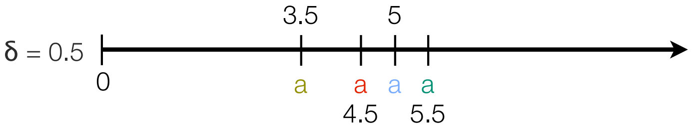
Preference Elicitation

- Function $R^{(\delta)}$ is an *Aggregate Objective Function* (many names...) familiar in multi-objective optimization

- Preference Elicitation Approach:
  - Figure out the decision maker's δ
  - Define a scalar reward $R^{(\delta)} = (1 - \delta) \cdot R^{(0)} + \delta \cdot R^{(1)}$
  - Use Q-learning to estimate the optimal policy for that reward

- Resulting policy is Pareto optimal

- E.g., for $\delta = 0.5$



Preference Elicitation

- E.g., "Consider two actions. You can have $(8, 5)$, or you can have $(5, x)$. What value of x makes you indifferent to this choice?"[2]
- Find δ so that $R^{(\delta)}$ is equal for the two points
  - $(1 - \delta) \cdot 8 + \delta \cdot 5 = (1 - \delta) \cdot 5 + \delta \cdot x$
- Doubt about whether or not this actually works
- Has nothing to do with the actions that are actually available, i.e. does not provide salient information about available treatments.

[2] Actual question would be more subtle.

Contribution: Inverse Preference Elicitation

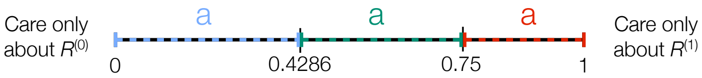
- Preference Elicitation
  - "Give me your δ, I will tell you the right action."

- Infinite number of δ, but only 4 actions

- Inverse Preference Elicitation
  - *"Given each available action,*
    I will tell you the δ for which that action is optimal."
  - This is our salient information

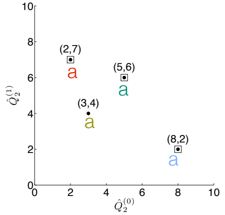
Contribution: Inverse Preference Elicitation

- *"Given each available action,*
  I will tell you the δ for which that action is optimal."
- Each action is optimal over a range of δ



Contribution: Inverse Preference Elicitation

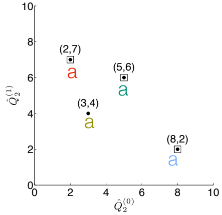
- *"Given each available action, I will tell you the δ for which that action is optimal."*
- Each action is optimal over a range of δ



- Note action a does not appear

Contribution: Inverse Preference Elicitation

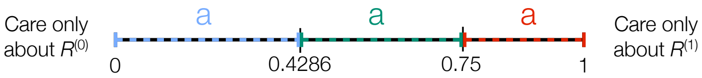
- *"Given each available action,*
  I will tell you the δ for which that action is optimal."
- Each action is optimal over a range of δ

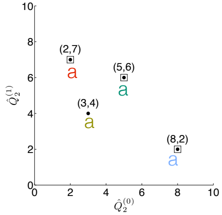


- Note action a does not appear
- Can see sensitivity of action choice w.r.t. preference

Contribution: Inverse Preference Elicitation

- *"Given each available action,*
  I will tell you the δ for which that action is optimal."
- Each action is optimal over a range of δ





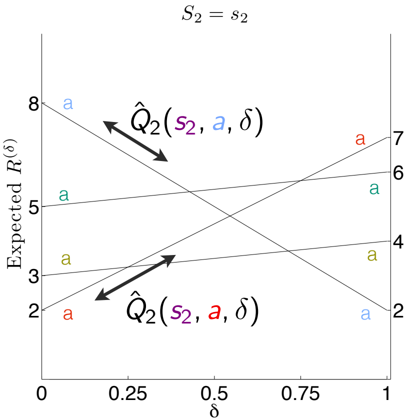
- Note action a does not appear
- Can see sensitivity of action choice w.r.t. preference
- Decision aid from the beginning is a coarsened version of a picture like this

Non-Myopic Inverse Preference Elicitation

- At each timepoint t, define $\hat{Q}_t(s_t, a_t; \hat{\beta}_t(\delta)) = \hat{\beta}_t^\top(\delta)\phi_t(s_t, a_t)$
  - Before, each $\hat{Q}_t(s_t, a_t)$ was a scalar. Now they are functions of δ.
- Perform Q-learning *for all δ simultaneously*
- At Time 2, finding ranges of δ is straightforward
  - Use convex hull to identify regions of δ
- At Time 1, things get interesting
- **Challenge:** represent $\hat{Q}_1(s_1, a_1; \hat{\beta}_1(\delta))$
  - Exactly
  - Economically

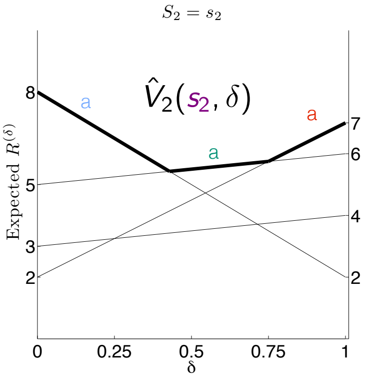
Q-Learning for all δ: Time 2

- Notice that $\hat{Q}_2(s_2, a_2; \hat{\beta}_2(\delta)) = \phi_2(s_2, a_2)^\mathsf{T}\hat{\beta}_2(\delta)$ is linear in δ, so only compute $\hat{\beta}_2(0)$ and $\hat{\beta}_2(1)$.



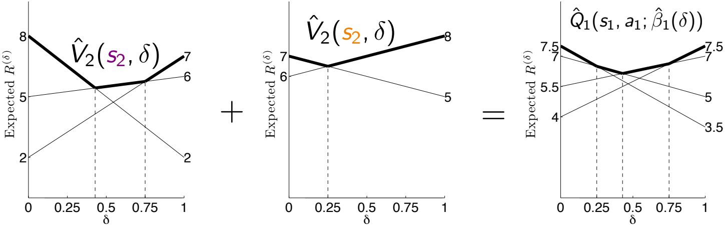
Q-Learning for all δ: Time 1

- Notice $\hat{V}_2(s_2;\delta)$ is piecewise linear in δ.



Q-Learning for all δ: Time 1

- Notice that $\hat{\beta}_1(\delta)$ and $\hat{Q}_1(s_1, a_1; \hat{\beta}_1(\delta))$ are linear over regions of δ where $\hat{V}_2(s_2; \delta)$ is simultaneously linear for all s_2.

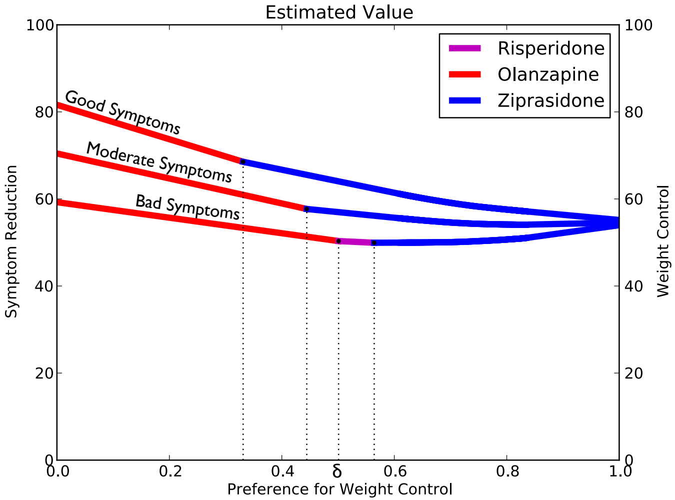


- Only need to evaluate $\hat{\beta}_1(\delta)$ at union of knots in $\hat{V}_2(S_2; \delta)$

Example: CATIE

- Large (N = 1460) comparative effectiveness trial
- Most patients randomized two times:
  - First to one of 5 actions
  - Then, if desired, to one of 5 different actions
- Following is a *highly* simplified analysis
- Overall, the results are consistent with the literature
- Rewards: symptoms relief, weight control

Example: CATIE Inverse Preference Elicitation



Example: CATIE-based Decision Aid

- Possible decision aid: Coarse version of the plots

Initial Symptoms	Preference: Symptom Relief		Preference: Weight Control	
	Strong	Mild	Mild	Strong
Good	Olan	Olan or Zip	Zip	Zip
Moderate	Olan	Olan or Zip	Zip	Zip
Bad	Olan	Olan	Risp or Zip	Zip

Olan = Olanzapine, Zip = Ziprasidone, Risp = Risperidone

- Thanks to Holly Wittemann, Brian Zikmund-Fisher, UMich SPH

Future Work

- Algorithms and Methods for Generating Evidence
  - More flexible models / approximation algorithms for preferences
  - Measures of uncertainty - requires interesting optimization
    - Ask me about this!
  - “Classical” ML problems (feature selection/dimensionality reduction/model selection, feature extraction via NLP, accommodating missing data...)
  - Must still provide salient information!

- Clinical Science Applications
  - Schizophrenia - CATIE
  - Major Depressive Disorder - STAR*D
  - ICU data (non-randomized) - MIMIC, MIMIC II
  - EHR?

Thank You

- Supported by National Institute of Health grants R01 MH080015 and P50 DA10075
- Daniel J. Lizotte, Michael Bowling, and Susan A. Murphy. *Efficient Reinforcement Learning with Multiple Reward Functions for Randomized Clinical Trial Analysis*. Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML), 2010.
- Related work:

  Barrett, L. and Narayanan, S. *Learning all optimal policies with multiple criteria.* In Proceedings of the 25th International Conference on Machine Learning 2008.

Confidence Intervals for Q-Learning

- Question: In state s_t, is there evidence that a is really better than a?
- Classical approach: get confidence interval for $\hat{\beta}_t^\mathsf{T} \cdot (\phi(s_t, a) - \phi(s_t, a))$
- For $t = T$, under mild assumptions on R, can use normal approximation or bootstrap
- For $t < T$, standard methods can fail even as $n \to \infty$
- Trouble arises when statistics (e.g. $\hat{\beta}_t$) are non-differentiable functions of the dataset
- $\hat{\beta}_1$ based on $\hat{V}_2(s_2) = \max_a \hat{Q}_2(s_2, a)$

Confidence Intervals for Q-Learning

- Question: In state s_t, is there evidence that a is really better than a?
- Classical approach: get confidence interval for $\hat{\beta}_t^\mathsf{T} \cdot (\phi(s_t, a) - \phi(s_t, a))$
- For $t = T$, under mild assumptions on R, can use normal approximation or bootstrap
- For $t < T$, standard methods can fail even as $n \to \infty$
- Trouble arises when statistics (e.g. $\hat{\beta}_t$) are non-differentiable functions of the dataset
- $\hat{\beta}_1$ based on $\hat{V}_2(s_2) = \max_a \hat{Q}_2(s_2, a)$

Confidence Intervals for Q-Learning

- Question: In state s_t, is there evidence that a is really better than a?
- Classical approach: get confidence interval for $\hat{\beta}_t^\mathsf{T} \cdot (\phi(s_t, a) - \phi(s_t, a))$
- For $t = T$, under mild assumptions on R, can use normal approximation or bootstrap
- For $t < T$, standard methods can fail even as $n \to \infty$
- Trouble arises when statistics (e.g. $\hat{\beta}_t$) are non-differentiable functions of the dataset
- $\hat{\beta}_1$ based on $\hat{V}_2(s_2) = \max_a \hat{Q}_2(s_2, a)$

Adaptive Confidence Intervals for Q-Learning

- A method that produces correct coverage:
  - Re-sample a dataset $\mathcal{D}'$ with replacement
  - Compute $\tilde{\beta}_t = \arg \max_{\beta \text{ near } \hat{\beta}_t} f(\beta, \mathcal{D}')$
  - Repeat
- Use distribution of $\tilde{\beta}_t$ to make C.I.

- The $\arg \max_{\beta \text{ near } \hat{\beta}_t} f(\beta, \mathcal{D}')$ problem is interesting
  - Non-convex
  - Piecewise linear but possibly not continuous
  - Can formulate as MIP, but maybe we can do better...