Multi-attribute utility models (started last time)
- preferential and utility independence
- additive and generalized addition models

Classical preference elicitation
- standard gambles
- additive and GAI models

Queries and partial elicitation
Utility Representations

- Utility function $u: X \rightarrow [0,1]$
  - decisions induce distribution over outcomes
  - or we simply choose an outcome (no uncertainty), but constraints on outcomes
- If $X$ is combinatorial, sequential, etc.
  - representing, eliciting $u$ difficult in explicit form
Product Configuration

Luggage Capacity?
Two Door? Cost?
Engine Size?
Color? Options?
Utility Representations

- Utility function \( u: X \rightarrow [0,1] \)
  - decisions induce distribution over outcomes
  - or we simply choose an outcome (no uncertainty), but constraints on outcomes

- If \( X \) is combinatorial, sequential, etc.
  - representing, eliciting \( u \) difficult in explicit form
  - is the following representation reasonable, comprehensible?

<table>
<thead>
<tr>
<th>Car 1</th>
<th>Toyota Prius</th>
<th>Silver</th>
<th>125hp</th>
<th>5.6l/100k</th>
<th>...</th>
<th>0.82</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car 2</td>
<td>Acura TL</td>
<td>Black</td>
<td>286hp</td>
<td>8.9l/100k</td>
<td>...</td>
<td>1.0</td>
</tr>
<tr>
<td>Car 3</td>
<td>Acura TL</td>
<td>Blue</td>
<td>286hp</td>
<td>8.9l/100k</td>
<td>...</td>
<td>0.96</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
COACH*

- POMDP for prompting Alzheimer’s patients
  - solved using factored models, value-directed compression of belief space
- Reward function (patient/caregiver preferences)
  - indirect assessment (observation, policy critique)
Winner Determination in Combinatorial Auctions

Expressive bidding in auctions becoming common
- expressive languages allow: combinatorial bids, side-constraints, discount schedules, etc.
- direct expression of utility/cost: economic efficiency

Advances in winner determination
- determine least-cost allocation of business to bidders
- new optimization methods key to acceptance
- applied to large-scale problems (e.g., sourcing)
Non-price Preferences

A and B for $12000.  
C and D for $5000...

A for $10000.  
B and D for $5000 if A; 
B and D for $7000 if not A...

A, C to Fred.  
B, D, G to Frank.  
F, H, K to Joe...

Cost: $57,500.

That gives too much business to Joe!!
Non-price Preferences

- WD algorithms minimize cost alone
  - but preferences for non-price attributes play key role
  - Some typical attributes in sourcing:
    - percentage volume business to specific supplier
    - average quality of product, delivery on time rating
    - geographical diversity of suppliers
    - number of winners (too few, too many), …

- Clear utility function involved
  - difficult to articulate precise tradeoff weights
  - “What would you pay to reduce \( \%volume_{Joe} \) by 1%?”
Manual Scenario Navigation*

- Current practice: manual *scenario navigation*
  - impose constraints on winning allocation
    - *not a hard constraint!*
  - re-run winner determination
  - new allocation satisfying constraint: higher cost
  - assess tradeoff and repeat (often hundreds of times) until satisfied with some allocation

Here's a new allocation with less business to Joe.

Cost is now: $62,000.
Utility Representations

- Utility function $u: X \rightarrow [0, 1]$
  - decisions induce distribution over outcomes
  - or we simply choose an outcome (no uncertainty), but constraints on outcomes
- If $X$ is combinatorial, sequential, etc.
  - representing, eliciting $u$ difficult in explicit form
- Some structural form usually assumed
  - so $u$ parameterized compactly (weight vector $w$)
  - e.g., linear/additive, generalized additive models

- Representations for qualitative preferences, too
  - e.g., CP-nets, TCP-nets, etc. [BBDHP03, BDS05]
Flat vs. Structured Utility Representation

- Naïve representation: vector of values
  - e.g., *car7*:1.0, *car15*:0.92, *car3*:0.85, …, *car22*:0.0

- Impractical for combinatorial domains
  - e.g., can’t enumerate exponentially many cars, nor expect user to assess them all (choose among them)

- Instead we try to exploit independence of user preferences and utility for different attributes
  - the relative preference/utility of one attribute is independent of the value taken by (some) other attributes

- Assume *X* \( \subseteq \text{Dom}(X_1) \times \text{Dom}(X_2) \times \ldots \times \text{Dom}(X_n) \)
  - e.g., *car7*: Color=red, Doors=2, Power=320hp, LuggageCap=0.52m³
Preferential, Utility Independence

- **X** and **Y = V-X** are *preferentially independent* if:
  - \(x_1y_1 \geq x_2y_1\) iff \(x_1y_2 \geq x_2y_2\) (for all \(x_1, x_2, y_1, y_2\))
  - e.g., *Color: red* > *blue* regardless of value of *Doors, Power, LugCap*
  - conditional P.I. given set **Z**: definition is straightforward

- **X** and **Y = V-X** are *utility independent* if:
  - \(l_1(Xy_1) \geq l_2(Xy_1)\) iff \(l_1(Xy_2) \geq l_2(Xy_2)\) (for all \(y_1, y_2\), all distr. \(l_1, l_2\))
  - e.g., preference for *lottery(Red, Green, Blue)* does not vary with value of *Doors, Power, LugCap*
  - implies existence of a “utility” function over local (sub)outcomes
  - conditional U.I. given set **Z**: definition is straightforward
### Question

- Is each attribute P.I. of others in preference relation 1, 2?

<table>
<thead>
<tr>
<th>Preferences #1</th>
<th>Preferences #2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Better</strong></td>
<td><strong>Better</strong></td>
</tr>
<tr>
<td>a b c</td>
<td>a b c</td>
</tr>
<tr>
<td>a b c</td>
<td>a b c</td>
</tr>
<tr>
<td>a b c</td>
<td>a b c</td>
</tr>
<tr>
<td>a b c</td>
<td>a b c</td>
</tr>
<tr>
<td>a b c</td>
<td>a b c</td>
</tr>
</tbody>
</table>

- Does UI imply PI? Does PI imply UI?
Additive Utility Functions

- **Additive representations** commonly used \[\text{[KR76]}\]
  - breaks exponential dependence on number of attributes
  - use sum of *local utility functions* \(u_i\) over attributes
  - or equivalently *local value functions* \(v_i\) plus scaling factors \(\lambda_i\)

\[
u(x) = \sum_{i=1}^{n} u_i(x_i) = \sum_{i=1}^{n} \lambda_i v_i(x_i).
\]

- This will make elicitation/optimization much easier

\[
u(\text{red,2dr,280hp}) = 0.85
\]
Additive Utility Functions

- An additive representation of $u$ exists iff decision maker is indifferent between any two lotteries where the marginals over each attribute are identical
  - $l_1(X) \sim l_2(X)$ whenever $l_1(X_i) = l_2(X_i)$ for all $X_i$

We’ll look at a rough proof sketch when we discuss elicitation of additive functions in a few minutes

<table>
<thead>
<tr>
<th>Outcome</th>
<th>$I_1$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_2$</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>$x'_1x_2$</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>$x_1x'_2$</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>$x'_1x'_2$</td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th>$I_2$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_2$</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>$x'_1x_2$</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>$x_1x'_2$</td>
<td>0.42</td>
<td></td>
</tr>
<tr>
<td>$x'_1x'_2$</td>
<td>0.28</td>
<td></td>
</tr>
</tbody>
</table>

Under additivity, two lotteries equally preferred, since marginals over $X_1$, $X_2$ are the same in each:
- $\Pr(X_1) = <.6, .4>$
- $\Pr(X_2) = <.3, .7>$
Generalized Additive Utility

- **Generalized additive models** more flexible
  - **interdependent value additivity** [Fishburn67], GAI [BG95]
  - assume (overlapping) set of $m$ subsets of vars $X[j]$
  - use sum of *local utility functions* $u_j$ over attributes

$$u(x) = \sum_{j=1}^{m} u_j(x_j)$$

- This can make elicitation/optimization much easier

<table>
<thead>
<tr>
<th>Color</th>
<th>Drs</th>
<th>$u_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>blue</td>
<td>4</td>
<td>0.9</td>
</tr>
<tr>
<td>red</td>
<td>4</td>
<td>0.6</td>
</tr>
<tr>
<td>blue</td>
<td>2</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pwr</th>
<th>Drs</th>
<th>$u_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>350</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>350</td>
<td>4</td>
<td>0.7</td>
</tr>
<tr>
<td>280</td>
<td>2</td>
<td>0.65</td>
</tr>
<tr>
<td>280</td>
<td>4</td>
<td>0.55</td>
</tr>
</tbody>
</table>

$u(\text{red,2dr,280hp}) = 0.79$
GAI Utility Functions

- An GAI representation of $u$ exists iff decision maker is indifferent between any two lotteries where the marginals over each factor are identical
  - $l_1(X) \sim l_2(X)$ whenever $l_1(X[i]) = l_2(X[i])$ for all $i$

$$u(x) = \sum_{j=1}^{m} u_j(x_j)$$

- Reasoning is similar to the additive case (but more involved)
Utility Elicitation

- Now, how do we assess a user’s utility function?

- First, we’ll look at classical elicitation
  - we’ll focus on additive models
  - review slides on generalized additive models if interested

- Then we’ll look at a couple “AI approaches” to assessing utility functions using:
  - predicting a user’s utility using learning (classification/clustering)
  - eliciting partial utility information (identifying “relevant” information)
Basic Elicitation: Flat Representation

- “Typical” approach to assessment
  - normalization: set best outcome utility 1.0; worst 0.0
    - \( u(x^\top) = 1 \quad u(x^\perp) = 0 \)
  - standard gamble queries: ask user for probability \( p \) with which indifference holds between \( x \) and \( SG(p) \)
    \[
    x \sim \langle p, x^\top; 1 - p, x^\perp \rangle
    \]
    \[
    u(x) = p \ u(x^\top) + (1 - p) \ u(x^\perp) = p
    \]
  - e.g., \( car3 \sim <0.85, car7; 0.15, car22 > \)

- SG queries: require precise numerical assessments

- Bound queries: fix \( p \), ask if \( x \) preferred to \( SG(p) \)
  - yes/no response: places (lower/upper) bound on utility
  - easier to answer, much less info (narrows down interval)
Elicitation: Additive Models

- First: assess local value functions with *local SG queries*
  - calibrates on \([0,1]\)
  
  \[ x_i \sim \langle p, x_i^\top; 1 - p, x_i^\perp \rangle \iff v_i(x_i) = p \]

- For instance,
  - ask for best value of Color (say, *red*), worst value (say, *grey*)
  - then ask local standard gamble for each remaining Color to assess it’s local value (*note: user specifies probability… difficult*)
    - *blue* \(\sim <0.85, \text{red}; 0.15, \text{grey}>\)
    - *green* \(\sim <0.67, \text{red}; 0.33, \text{grey}>\), …

- Bound queries can be asked as well
  - only refine intervals on local utility
Elicitation: Additive Models

Second: assess *scaling factors* with “global” queries

- define *reference* outcome \( x^0 = (x^0_1, x^0_2, \ldots, x^0_n) \)
  - could be worst global outcome, or any salient outcome, ...
  - e.g., user’s current car: *(red, 2door, 150hp, 0.35m^3)*
- define \( x^j \) by setting \( X_j \) to best value, others to reference value
  - e.g., for doors: *(red, 4door, 150hp, 0.35m^3)*
  - *by independence, best value 4door must be fixed (whatever ref. values)*
- compute scaling factor
  \[
  \lambda_j = u(x^j) - u(x^{\perp j})
  \]
- assess these \( 2n \) utility values with (global) SG queries

Altogether: gives us full utility function

\[
u(x) = \sum_{i=1}^{n} u_i(x_i) = \sum_{i=1}^{n} \lambda_i v_i(x_i).
\]
Why Does the Additive Rep’n Suffice?

Let \( \succeq \) be a pref order with utility f’n \( u \). Want to show (MELEP) iff (ADD)
- (MELEP) any pair of marginal-equivalent lotteries are equally preferred
- (ADD) \( u \) has an additive decomposition \( u(x) = \sum u_i(x_i) \)

(ADD) implies (MELEP) is obvious (exercise)

Sketch other direction. Assume two variables \( X_1, X_2 \) (generalizes easily)
- MELEP implies \( [\frac{1}{2}(x_1, x_2), \frac{1}{2}(x'_1, x'_2)] \sim [\frac{1}{2}(x_1, x'_2), \frac{1}{2}(x'_1, x_2)] \) for any \( x_1, x_2, x'_1, x'_2 \)
- Let \( x^* = (x^*_1, x^*_2) \) be an arbitrary reference outcome.
- Set \( u_1(x^*_1) + u_2(x^*_2) = u(x^*) \) (however you want)
- For all other \( x_1, x_2 \), define \( u_1(x_1) = u((x_1, x^*_2)) - u_2(x^*_2) \) & \( u_2(x_2) = u((x^*_1, x_2)) - u_1(x^*_1) \)
- By (2) and (3): \( u_1(x_1) + u_2(x_2) = u((x_1, x^*_2)) + u((x^*_1, x_2)) - u(x^*) \)
- By (1): \( [\frac{1}{2}(x_1, x_2), \frac{1}{2}(x^*_1, x^*_2)] \sim [\frac{1}{2}(x_1, x'_2), \frac{1}{2}(x'_1, x_2)] \)
- So by EU and (5): \( \frac{1}{2}u(x_1, x_2) + \frac{1}{2}u(x^*_1, x^*_2) = \frac{1}{2}u(x_1, x'_2) + \frac{1}{2}u(x'_1, x_2) \)
- Rearranging (6): \( u(x_1, x_2) = u(x_1, x^*_2) + u(x^*_1, x_2) - u(x^*_1, x^*_2) \)
- Plugging (4) into (7): \( u(x_1, x_2) = u_1(x_1) + u_2(x_2) \)

Step (3) is key: Define \( u_1(x_1) = u((x_1, x^*_2)) - u_2(x^*_2) \) to be the marginal contribution of \( x_1 \) to utility of an outcome given reference value \( x^*_2 \); similarly for \( u_2(x_2) \).
Normalizing Local Utility Functions

- Given an additive $u(x)$, normalization is easy:
  - Need to define local value functions $v_i(x_i)$ normalized in $[0,1]$
  - Need to define scaling constants $\lambda_i$ that sum to one
  - Let’s assume reference outcome is $x^\perp$

- Set $u^*(x) = \frac{u(x) - u^\perp}{u^\top - u^\perp}$; just an affine transformation of $u$.

$$u^*(x) = \frac{u(x) - u^\perp}{u^\top - u^\perp} = \frac{\sum u_i(x_i) - \sum u_i^\perp}{\sum u_i^\top - \sum u_i^\perp} = \frac{\sum (u_i(x_i) - u_i^\perp)}{\sum (u_i^\top - u_i^\perp)}$$

$$= \sum \frac{u_i^\top - u_i^\perp}{\sum (u_i^\top - u_i^\perp)} \frac{u_i(x_i) - u_i^\perp}{u_i^\top - u_i^\perp}$$

$$= \sum \lambda_i v_i(x_i),$$
Elicitation: GAI Models (Classical)

- Assessment is subtle (won’t get into gory details)
  - overlap of factors a key issue [F67, GP04, DB05]
  - cannot rely on purely local queries: values cannot be fixed without reference to others!
  - seemingly “different” local preferences correspond to the same $u$

\[
u(\text{Color}, \text{Doors}, \text{Power}) = u_1(\text{Color}, \text{Doors}) + u_2(\text{Doors}, \text{Power})
\]

\[
u(\text{red}, 2\text{door}, 280\text{hp}) = u_1(\text{red}, 2\text{door}) + u_2(2\text{door}, 280\text{hp})
\]

\[
u(\text{red}, 4\text{door}, 280\text{hp}) = u_1(\text{red}, 4\text{door}) + u_2(4\text{door}, 280\text{hp})
\]
Fishburn’s Decomposition [F67] Optional

- Define *reference outcome*: \( x^0 = (x_1^0, x_2^0, x_3^0, \ldots, x_n^0) \)

- For any \( x \), let \( x[I] \) be restriction of \( x \) to vars \( I \), with remaining replaced by default values:

\[
x[{1, 2}] = (x_1, x_2, x_3^0, \ldots, x_n^0)
\]

- Utility of \( x \) can be written [Fishburn67]

\[
u(x) = \sum_{j=1}^{m} (-1)^{j+1} \sum_{1 \leq i_1 < i_2 < \cdots < i_j \leq m} u \left( x \left[ \bigcap_{s=1}^{j} I_{i_s} \right] \right)
\]

- sum of utilities of certain related “key” outcomes
Key Outcome Decomposition Optional

- Example: GAI over I={ABC}, J={BCD}, K={DE}

  - \( u(x) = u(x[I]) + u(x[J]) + u(x[K]) \)
    - \( - u(x[I \cap J]) - u(x[I \cap K]) - u(x[J \cap K]) \)
    - \( + u(x[I \cap J \cap K]) \)

  - \( u(abcde) = u(x[abc]) + u(x[bcd]) + u(x[de]) \)
    - \( - u(x[bc]) - u(x[\]) - u(x[d]) \)
    - \( + u(x[\]) \)

  - \( u(abcde) = u(abcd0e0) + u(a0bcd0e0) + u(a0b0c0de) \)
    - \( - u(a0bcd00e0) - u(a0b0c0de0) \)
This leads to canonical decomposition of $u$:

$$u(x_1, x_2, x_3) = u(x_1, x_2, x_3^0) + u(x_1^0, x_2, x_3) - u(x_1, x_2, x_3^0).$$

\[u_1(x_1, x_2)\quad u_2(x_2, x_3)\]

E.g., $I=\{ABC\},\ J=\{BCD\},\ K=\{DE\}$

$$u(abcde) = u(abcd^0e^0) + u(a^0bcd^0e^0) - u(a^0bcd^0e^0) + u(a^0b^0c^0de) - u(a^0b^0c^0de^0)$$

$$= u_1(abc) + u_2(bcd) + u_3(de)$$
Local Queries [Braziunas, B. UAI05] Optional

- We wish to avoid queries on whole outcomes
  - can’t be purely local; but condition on a *subset* of reference values

- **Conditioning set** $C_i$ for factor $u_i(X_i)$:
  - vars (excl. $X_i$) in any factor $u_k(X_k)$ where $X_i \cap X_k \neq \emptyset$
  - setting $C_i$ to reference values renders $X_i$ independent of remaining variables
    - e.g., *Power=280hp* shields *<Color, Door>* from any other vars

- Define *local* best/worst for $u_i$ assuming $C_i$ set at reference levels

- Ask SG queries relative to local best/worst with $C_i$ fixed
  - e.g., fix *Power=280hp* and ask SG queries on *<Color, Door>* conditioned on *280hp*
Local Queries [BB05] Optional

- **Theorem:** If for some \( y \) (where \( Y = X - X_i - C(X_i) \))

\[
(x_i, x_{C_i}^0, y) \sim \langle p, (x_i^\top, x_{C_i}^0, y); 1 - p, (x_i^\top, x_{C_i}^0, y) \rangle
\]

then for all \( y' \)

\[
(x_i, x_{C_i}^0, y') \sim \langle p, (x_i^\top, x_{C_i}^0, y'); 1 - p, (x_i^\top, x_{C_i}^0, y') \rangle
\]

- Hence we can legitimately ask *local* queries:

\[
(x_i, x_{C_i}^0) \sim \langle p, (x_i^\top, x_{C_i}^0); 1 - p, (x_i^\top, x_{C_i}^0) \rangle
\]
Conditioning Sets Optional

\[ AE = a^0e^0 \]
\[ D = d^0 \]
\[ BCF = b^0c^0f^0 \]
\[ EJ = e^0j^0 \]
\[ DGHJ = d^0g^0h^0j^0 \]
\[ EH = e^0h^0 \]
Local Standard Gamble Queries

Local standard gamble queries
- use “best” and “worst” local outcome—conditioned on default values of conditioning set
  - e.g., $x^{T}[1] = abcd^0$ for factor ABC; $x^{⊥}[1] = ~abcd^0$
- SG queries on other parameters relative to these
- gives local value function $v(x[i])$ (e.g., $v(ABC)$)

Can use bound queries as well

But local VFs not enough: must calibrate
- requires global scaling
Global Scaling \textbf{Optional}

- Assess scaling factors with "global" queries
  - exactly as with additive models
  - define \textit{reference} outcome \( \mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0) \)
  - define \( \mathbf{x}^\top_j \) by setting \( X[j] \) to best value, others to ref
  - compute scaling factor
    \[
    \lambda_j = u(\mathbf{x}^\top_j) - u(\mathbf{x}^\perp_j)
    \]
  - assess the \( 2n \) utility values with (global) SG queries
  - can use bound queries as well
Elicitation: Beyond the Classical View

- The classic view involving standard gambles difficult:
  - large number of parameters to assess (structure helps)
  - unreasonable precision required (SGQs)
  - queries over full outcomes difficult (structure helps)
  - cost (cognitive, communication, computational, revelation) may outweigh benefit
    - *can often make optimal decisions without full utility information*

- General approach to practical, automated elicitation
  - cognitively plausible forms of interaction
  - incremental elicitation until decision possible that is *good enough*
  - collaborative/learning models to allow generalization across users
Beyond Standard Gamble Queries

**Bound queries**

- a boolean version a (global/local) SG query
  - **global**: “Do you prefer \( x \) to \([(p, x^T), (1-p, x^\perp)]\)?”
  - **local**: “Do you prefer \( x[k] \) to \([(p, x^T[k]), (1-p, x^\perp[k])]\) ?”
    - need to fix reference values \( C_k \) if using GAI model
- response tightens bound on specific utility parameter

**Comparison queries** (is \( x \) preferred to \( x' \) ?)

- **global**: “Do you prefer \( x \) to \( x' \)?”
- **local**: “Do you prefer \( x[k] \) to \( x'[k] \)?”
- impose linear constraints on parameters
  - \( \Sigma_k u_k(x[k]) > \Sigma_k u_k(x'[k]) \)
- interpretation is straightforward
Other Modes of Interaction

- **Stated choice (global or local)**
  - choose $x_i$ from set \{x_1, \ldots x_k\}
  - imposes $k-1$ linear constraints on utility parameters

- **Ranking alternatives (global or local)**
  - order set \{x_1, \ldots x_k\} : similar

- **Graphical manipulation of parameters**
  - bound queries: allow tightening of bound (user controlled)
    - generally must show implications of moves made
  - approximate valuations: user-controlled precision
    - useful in quasi-linear settings

- **Passive observation/revealed preference**
  - if choice $x$ made in context $c$, $x$ as preferred as other alternatives

- **Active, but indirect assessment**
  - e.g., dynamically generate Web page, with $k$ links
  - assume response model: $Pr(link_j | u)$
Local Queries: Comparison

This is a **comparison** query. Please carefully consider the two outcomes below and indicate which outcome is of higher value by clicking on the question mark.

You prefer Outcome 2 to Outcome 1.
Local Query: Bound

This is a local **bound** query. Below, the outcome on the left (in blue) is the worst outcome (in some factor), and the outcome on the right (in red) is the best. Now, assume a scale from 0 to 100, with the worst outcome rated 0, and the best outcome rated 100. You are asked to decide where the outcome in question (directly below) falls on this scale. If its value is between 0 and the tip of the slider, please drag it to the left bin; otherwise, drag it to the right bin.
Local Query: Bound
Global Query: Anchor Comparison

This is a comparison query. Please carefully consider the two outcomes below and indicate which outcome is of higher value by clicking on the question mark.

You prefer Outcome 1 to Outcome 2

User selects > or < (from ?)
Global Query: Anchor Bound

This global **bound** query asks you to provide a monetary bound on the value of the outcome below.

Is the value of this outcome greater than $1650?

- **Yes, greater than $1650**
- **No, less than $1650**
Cognitive Biases: Anchoring

- Decision makers susceptible to context in assessing preferences (and other relevant info, like probabilities)
- **Anchoring**: assessment of utility dependent on arbitrary influences
- Classic experiment [ALP03]:
  - (business execs) write last 2 digits of SSN on piece of paper
  - place bids in mock auction for wine, chocolate
  - those with SSN>50 submitted bids 60-120% higher than SSN<50
- Often explained by focus of attention plus adjustment
  - holds for estimation of probabilities (Tversky, Kahneman estimate of # African countries), numerical quantities, …
- How should this impact the design of elicitation methods?
Cognitive Biases: Framing

- How questions/choices are *framed* is critical
- Classic Tversky, Kahneman experiment (1981); disease predicted to kill 600 people, choose vaccination program
  - Choose between:
    - Program A: "200 people will be saved"
    - Program B: "there is a one-third probability that 600 people will be saved, and a two-thirds probability that no people will be saved"
  - Choose between:
    - Program C: "400 people will die"
    - Program D: "there is a one-third probability that nobody will die, and a two-third probability that 600 people will die"
  - 72 percent prefer A over B; 78 percent prefer D over C
  - Notice that A and C are equivalent, as are B and D
- How should this impact design of elicitation schemes?
Cognitive Biases: Endowment Effect

- People become “attached” to their possessions
  - e.g., experiment of Kahneman, et al. 1990
- Randomly assign subjects as buyers, sellers
  - sellers given a coffee mug (sells for $6); all can examine closely
  - sellers asked: “at what price would you sell?”
  - buyers asked: “at what price would you buy?”
  - median asking price: $5.79; median offer price: $2.25
    - would expect these to be identical given random asst to groups
  - if sellers are given *tokens* with a monetary value (can be used later to buy mugs/chocolate in bookstore), no difference between offers and ask prices
- How should this impact the design of elicitation methods?
Utility Elicitation as a Classification Problem. Chajewksa, et al. (1998)

- Want to make decisions: but utility elicitation is difficult
  - Large outcome space (exponential, hard to wrap head around complete outcomes)
  - Hard to assess quantitatively
- Problem 2: std. gambles, esp. bound queries, can help
- Problem 1: additive independence (or GAI) helps

Still very difficult, intensive
- Can we focus our elicitation effort on only utility information relevant to decision at hand?
- If elicitation costly, might be better off making assumptions or predictions and living with approximately optimal decisions
CGNS Motivation

- Medical decision scenario (prenatal testing, termination)
  - Consequences of decisions are significant

- Basic model is this:
  - **Offline**: find clusters of *similar utility functions* (case database)
    - **Similar**: a *single* decision is close to optimal for each element
    - Good clusters assumed to exist
  - **Online**: take steps to identify a user’s cluster, propose optimal decision for that cluster
    - Should help ease elicitation burden
Influence Diagram (PANDA)

From: Chajewksa, et al., UAI 1998)
CGNS: High Level Picture

- Clusters produced using simple agglomerative methods
- Elicitation policy: find a decision tree that distinguishes the clusters using very few queries
  - Plops you into a cluster, makes decision using prototype utility f’n

Queries:
- Feature: is age < 40?
- Comparison: is o1 > o2?

Clusters: in each cluster C there is some strategy s, s.t. for all u in C, s is approx. optimal for u (we will define)
Basic Inputs

- **Set of strategies** $S = \{s_1, \ldots s_m\}$
  - Conditional plans, e.g., “Test A. If obs Z, test B; …; if Obs Z’, do X”
  - 18 strategies, only 4 useful for DB
  - Sequential component of decisions abstracted away

- **Set of outcomes** $O = \{o_1, \ldots o_n\}$
  - E.g., “healthy baby, no future conception, …” (22 outcomes)

- **History**: observable prior patient info (health status, etc.)

- **Outcome distribution**: $P(O|S,H)$

- **$EU(S|H) = \sum_o P(o|S,H) u(o)$ (assuming known utility $u$)**
Strategies (only 4 optimal)

Strategy 1

- no → no → no → no
- normal → no → no

Strategy 2

- yes → abnormal → yes → normal
- uncertain → no → yes → abnormal or uncertain

Strategy 3

- yes → abnormal → yes → normal
- uncertain → no → yes → abnormal or uncertain

Strategy 4

- yes → normal → no → no

Decisions

CVS → CVS results → early TAB → AMNIO → AMNIO results → late TAB

From: Chajewksa, et al., UAI 1998)
Clustering

- N utility functions in DB, each a vector \([u(o_1), \ldots, u(o_n)]\)
  - elicited by clinical decision analysts (70 in DB, 55 used)
  - question: why use utilities in DB instead of all possible utility f’ns?
- Want to find \(k\) clusters of \(u\)’s, elements in a cluster similar
- Similar? Want to treat all \(u\)’s in any \(C\) indistinguishably
  - Same strategy applied to all, so there should be one strategy that is optimal, or at least very good, for every \(u\) in \(C\)
Clustering: Distance Function

- **Fix history** $h$
  - Define $EU(s|h,u_i) = \sum_o P(o|s,h) u_i(o)$
  - $s^*(u_i)$ is best strategy for $u_i$ given $h$
  - If we use prototype utility $u_p$ for the cluster containing $u_i$ instead of $u_i$ itself, $s^*(u_p)$ would be performed
  - **Loss:** $UL(u_i, u_p |h) = EU(s^*(u_i) |h,u_i) - EU(s^*(u_p) |h,u_i))$
  - **Distance:** $d(u_i, u_j | h) = Avg \{ UL(u_i, u_j |h) , UL(u_j, u_i |h) \}$

- **Comments**
  - Why fixed history? Must cluster online (once $h$ known)
    - Otherwise would need to perform clustering for all $h$ a priori
  - Other alternatives? $d(u_i, u_j) = \sum_h d(u_i, u_j | h) Pr(h)$ ?
    $d(u_i, u_j) = max_h d(u_i, u_j | h)$?
Agglomerative Clustering

- Initially, each \( u \) in its own cluster (recall: \( h \) is fixed)
- Then repeatedly merge two clusters that are most similar
  - \( d(C_i, C_j) \) is avg of the pairwise distances between \( u \)'s in each \( C \)
- Merge until we have \( k \) clusters (or use some validation method)
- \( \text{Score}(u_i) \) in cluster \( C \): \( \sum \{ UL(u_i, u_j | h) : u_j \in C \} \)
- Choose prototype utility for \( C \): the \( u_i \in C \) with min score

Comments
- Why choose prototype utility, and use \( s^*(u_i) \)?
- What about: \( \min_s \sum_{u_i \in C} \{ EU(s^*(u_i) | h, u_i) - EU(s | h, u_i) \} \)
Classification

- **Goal:** minimize elicitation effort
- **Technique:** build a decision tree that asks various questions/tests so that any sequence of answers "uniquely" determines a cluster (hence prototype)

**CGNS do the following:**

- Data is set of utility functions in DB, *labeled* by cluster it is in
- Now try to find predictor for cluster membership
- Possible splits (features for classification):
  - Is $o_i > o_j$?: implicit in $u$, $O(n^2)$ such Boolean tests
  - Is $o_i > [p, o_T; 1-p, o_\perp]$?: equiv to Is $u(o_i) > p$?
    - Note: boolean, but infinitely many such splits (values of $p$)
    - Trick: no more than $n$ values of $u(o_i)$ in DB; so consider midpoints between such values (and ignore small intervals)
  - Note: no history/patient features used! Tree is for fixed $h$
Resulting Decision Tree (h = “Teen”)

From: Chajewksa, et al., UAI 1998)
Empirical Results

Figure 6: Learning curves (average of 10,000 runs).

Figure 7: Leave-one-out cross-validation for number of clusters.

From: Chajewksa, et al., UAI 1998)
Discussion Points

- Queries over full outcomes: OK?
- Are utility function clusters legitimate?
  - cover cases in DB, but how different could other u’s be?
  - high error rate for 45YO: very sensitive to small changes in u (!)
- Could we use other features for prediction?
  - CGNS assume utility independent of observable history
- How do you account for all observable histories?
- Distributional information about preferences?
- Cost/effort of questions?
- Myopic nature of decision tree construction
Further Background Reading

Interactive Decision Making

- General framework for interactive decision making:

  \( B: \text{beliefs about user's utility function } u \)
  \( \text{Opt}(B): \text{“optimal” decision given incomplete, noisy, and/or imprecise beliefs about } u \)

  - Repeat until \( B \) meets some termination condition
    - ask user some query (propose some interaction) \( q \)
    - observe user response \( r \)
    - update \( B \) given \( r \)
  - Return/recommend \( \text{Opt}(B) \)
Regret-Based Elicitation

- Elicitation model that gives guarantees on decision quality
  - contrast data-driven approach of CGNS (and learning models)

- In *regret-based* methods:
  - uncertainty represented by a *set of utility functions*
    - those utility functions consistent with query responses
  - decisions made using *minimax regret*
    - robustness criterion well-suited to utility function uncertainty
    - provides bounds on how far decision could be from optimal
  - queries are asked to drive down minimax regret as quickly as possible

- *Constraint-based Optimization and Utility Elicitation using the Minimax Decision Criterion*. Boutilier, et al. 2006:
  - attack constraint-based combinatorial optimization problems
Decision Problem: Constraint Optimization

- **Standard constraint satisfaction problem (CSP):**
  - outcomes over variables $X = \{X_1 \ldots X_n\}$
  - constraints $C$ over $X$ : feasible decisions/outcomes
    - generally compact, e.g., $X_1 \& X_2 \supset \neg X_3$
    - e.g., $\text{Power} > 280\text{hp} \& \text{Make}=\text{BMW} \supset \text{FuelEff} > 9.5\text{l/100km}$
    - e.g., $\text{Volume}(\text{Supplier27}) > 10,000,000$

- **Feasible solution:** a satisfying variable assignment

- **Constraint-based/combinatorial optimization:**
  - add to $C$ a *utility function* $u: \text{Dom}(X) \rightarrow \mathbb{R} / [0,1]$
  - $u$ parameterized compactly (weight vector $w$)
    - e.g., linear/additive, generalized additive models

- Solved using search (B&B), integer programming, variable elimination, etc.
Strict Utility Function Uncertainty

- User’s utility parameters $w$ unknown
- Assume feasible set $W$
  - e.g., $W$ defined by a set of linear constraints on $w$

$$u(\text{red, 2 door, 280hp}) > 0.4$$
$$u(\text{red, 2 door, 280hp}) > u(\text{blue, 2 door, 280hp})$$

- allows for unquantified or “strict” uncertainty

- How should one make a decision? elicit info?
  - regret-based approaches
  - polyhedral approaches (and other heuristics)
Minimax Regret

• Regret of $x$ under $w$

$$R(x, w) = \max_{x' \in X} u(x'; w) - u(x; w)$$

• Max regret of $x$ under $W$

$$MR(x, W) = \max_{w \in W} R(x, w)$$

• Minimax regret and optimal allocation

$$x^*_W = \arg \min_{x \in X} MR(x, W)$$
Computing MMR

- **Direct factored representation:**
  - minimax program (rather than straight min or max)
  - potentially quadratic objective

\[
\text{MMR}(U) = \min_{x \in \text{Feas}(X)} \text{MR}(x, U) \\
= \min_{x \in \text{Feas}(X)} \max_{u \in U} \max_{x' \in \text{Feas}(X)} u(x') - u(x)
\]

- **Solution:**
  - natural structure that allows direct integer program formulation
  - Bender’s style decomposition/constraint generation
Pairwise Regret (Bounds)

- Graphical (GAI) model with factors $f_k$
- Assume \textit{bounds} $u_{x[k]} \uparrow$ and $u_{x[k]} \downarrow$ on parameters

<table>
<thead>
<tr>
<th>Factor(_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>red</td>
</tr>
<tr>
<td>blue</td>
</tr>
<tr>
<td>red</td>
</tr>
<tr>
<td>blue</td>
</tr>
</tbody>
</table>
Pairwise Regret (Bounds)

- Graphical (GAI) model with factors $f_k$
- Assume *bounds* $u_{x[k]}^{↑}$ and $u_{x[k]}^{↓}$ on parameters

Pairwise regret of $x$ and $x'$ can be broken into sum of *local regrets*:

- $r_{x[k]x'[k]} = u_{x'[k]}^{↑} - u_{x[k]}^{↓}$ if $x[k] \neq x'[k]$
  
- $= 0$ otherwise

- $R(x, x') = r_{xx'} = \sum_k r_{x[k]x'[k]}$

- no need to maximize over $U$ explicitly

<table>
<thead>
<tr>
<th>Color</th>
<th>Drs</th>
<th>$u_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>2</td>
<td>[0.7, 1.0]</td>
</tr>
<tr>
<td>blue</td>
<td>4</td>
<td>[0.8, 0.95]</td>
</tr>
<tr>
<td>red</td>
<td>4</td>
<td>[0.2, 0.7]</td>
</tr>
<tr>
<td>blue</td>
<td>2</td>
<td>[0.35, 0.4]</td>
</tr>
</tbody>
</table>
Computing Max Regret

- Max regret $MR(x, W)$ computed as an IP
  - number of vars *linear* in GAI model size
  - number of (precomputed) constants (i.e., local regret terms for all possible $x$) *quadratic* in GAI model size

\[
\max_{\{I_{x[k]}, X_i\}} \sum_k \sum_{x'[k]} r_{x[k]x'[k]} I_{x'[k]} \quad \text{subj. to } A, C
\]
Minimax Regret in GAI Models

- We convert minimax to min (standard trick)
  - obtain a MIP with one constraint per feasible config
  - linearly many vars (in utility model size)
- Key question: can we avoid enumerating all $x'$ ?

\[
MMR(U) = \min_{\{I_{x[k]}, X_i\}} \max_{x' \in Feas(X')} \sum_k \sum_{x[k]} r_{x[k], x'[k]} I_{x[k]} \quad \text{subject to } A \text{ and } C
\]

\[
= \min_{\{I_{x[k]}, X_i, M\}} M \\
\text{subject to } \left\{ \begin{array}{c}
M \geq \sum_k \sum_{x[k]} r_{x[k], x'[k]} I_{x[k]} \\
\forall x' \in Feas(X') \end{array} \right. \\
\text{A and } C
\]
Constraint Generation

- Very few constraints will be active in sol’n
- Iterative approach:
  - solve relaxed IP (using a subset of constraints)
  - if any constraint violated at solution, add it and repeat

- Let $Gen = \{x'\}$ for some feasible $x'$
- Solve MMX-IP using only constraints for $x' \in Gen$
  - let solution be $x^*$ with objective value $m^*$
- Solve MR-IP for $x^*$ obtaining solution $x', r$
- If $r > m^*$, add $x'$ to $Gen$ and repeat;
  - else terminate
  - note: $x'$ is *maximally* violated constraint
real estate: 20 vars (47m configs); 29 factors in utility model (1-3 vars per), with 160 parameters (320 bounds)
Regret-based Elicitation

- Minimax optimal solution may not be satisfactory
- Improve quality by asking queries
  - new bounds on utility model parameters
- Which queries to ask?
  - what will reduce regret most quickly?
  - myopically? sequentially?
- BPPS develop a heuristic: the current solution strategy
  - explored for bound queries on GAI model parameters
  - Intuition: ask user to refine our knowledge to utility parameters that impact utility of the minimax optimal solution or the adversarial witness; if we don’t change those, we won’t reduce pairwise max regret between them (and these determine MMR currently)
Elicitation Strategies (Bound): Simple GAI

- **Halve Largest Gap (HLG)**
  - ask if parameter with largest gap > midpoint
  - $\text{MMR}(U) \leq \text{maxgap}(U)$, hence $n \cdot \log(\text{maxgap}(U)/\varepsilon)$ queries needed to reduce regret to $\varepsilon$
  - bound is tight
  - like polyhedral-based conjoint analysis [THS04]

---

Domain: $A, B, C$

MMR-opt: $a \ b \ c$

Adv. w/ wtn: $a \ b \ c$

```
f_1(a,b) f_1(a,b) f_1(\overline{a},b) f_1(\overline{a},b) f_2(b,c) f_2(b,c) f_2(b,c) f_2(b,c)
```
Elicitation Strategies (Bound): Simple GAI

**Current Solution (CS)**

- only ask about parameters of optimal solution $x^*$ or regret-maximizing witness $x^w$
- intuition: focus on parameters that contribute to regret
  - reducing u.b. on $x^w$ or increasing l.b. on $x^*$ helps
- use early stopping to get regret bounds (CS-5sec)

Domain: \{A,B,C\}

MMR-opt: \{a,b,c\}

Adv. witn: \{a,b,c\}
Elicitation Strategies (Bound): Simple GAI

- **Optimistic**
  - query largest-gap parameter in optimistic soln $x^o$

- **Pessimistic**
  - query largest-gap parameter in pessimistic soln $x^p$

- **Optimistic-pessimistic (OP)**
  - query largest-gap parameter $x^o$ or $x^p$

- **Most uncertain state (MUS)**
  - query largest-gap parameter in uncertain soln $x^{mu}$

- **CS needs minimax optimization; HLG needs no optimization; others require standard optimization**
- **None except CS knows what MMR is (termination is problematic)**
Results (Small Rand, Unif)

Small Random Problem -- Uniform Prior

- 10 factors, at most 3 vars
- Users drawn using uniform prior over parameters (45 trials)
- Gaussian priors similar

Minimax regret vs. max utility

Number of queries

HLG, CS-5, CS, OP, MUS, SB
Results (Car Rental, Unif)

Car Rental Problem -- Uniform Prior

- 26 vars; 61 billion configs
- 36 factors, at most 5 vars; 150 parameters
- Users drawn using uniform prior over parameters (45 trials)
- Gaussian priors similar
Results (Real Estate, Unif)

House Buying Problem -- Uniform Prior

Minimax regret / max utility

Number of queries

Minimax regret

20 vars; 47 million configs

29 factors, at most 5 vars; 100 parameters

Users drawn using uniform prior over parameters (45 trials)

Gaussian priors similar
Results (Large Rand, Unif)

Large Random Problem -- Uniform Prior

- HLG
- CS-5
- CS
- OP
- SB
- MUS

- 25 vars; < 5 vals
- 20 factors, at most 3 vars
- Users drawn using uniform prior over parameters (45 trials)
- Gaussian priors similar
Elicitation Strategies: Summary

- Comparison queries can be generated using CSS too
  - HLG is harder to generalize to comparisons (see polyhedral)
- CSS: ask user to compare minimax optimal solution $x^*$ with regret-maximizing witness $x^w$
  - easy to prove this query is never “vacuous”

- CS works best on test problems
  - time bounds (CS-5): little impact on query quality
  - always know max regret (or bound) on solution
  - time bound adjustable (use bounds, not time)
- OP competitive on most problems
  - computationally faster (e.g., 0.1s vs 14s on RealEst)
  - no regret computed so termination decisions harder
- Other strategies less promising (incl. HLG)
Apartment Search [Braziunas, B, EC-10]

- Are users comfortable with MMR?
- Study with UofT students
  - search subset of student housing DB (100 apts) for rental
  - GAI model over 9 variables, 7 factors
  - queries generated using CSS (bound, anchor, local, global)
    - continue until MMR=0 or user terminates (“happy”)
  - post-search: through entire DB to find best 10 or so apartments

- Qualitative Results:
  - system-recommended apartment almost always in top ten
  - if MMR-apartment not top ranked, error (how much more is top apartment worth) tends to be very small
  - very few queries/interactions needed (8-40); time taken roughly 1/3 of that of searching through DB with our tools
  - user feedback: comfortable with queries, MMR, felt search was efficient
Further Background Reading