Preference Elicitation for Interface Optimization

Krzysztof Gajos and **Daniel S. Weld**
University of Washington, Seattle

Krzysztof Gajos

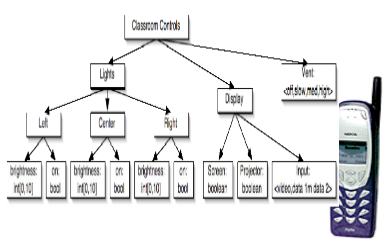
Motivation:
Supple Model-Based Interface Renderer

Func. Interface Spec. + Device Model + User Trace → Custom Interface Rendering

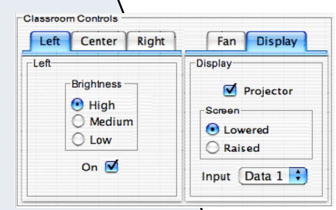
Hierarchy of State Vars + Methods

Screen Size, Available Widgets & Interaction Modes

Model of an Individual User's Behavior (or that of a Group)

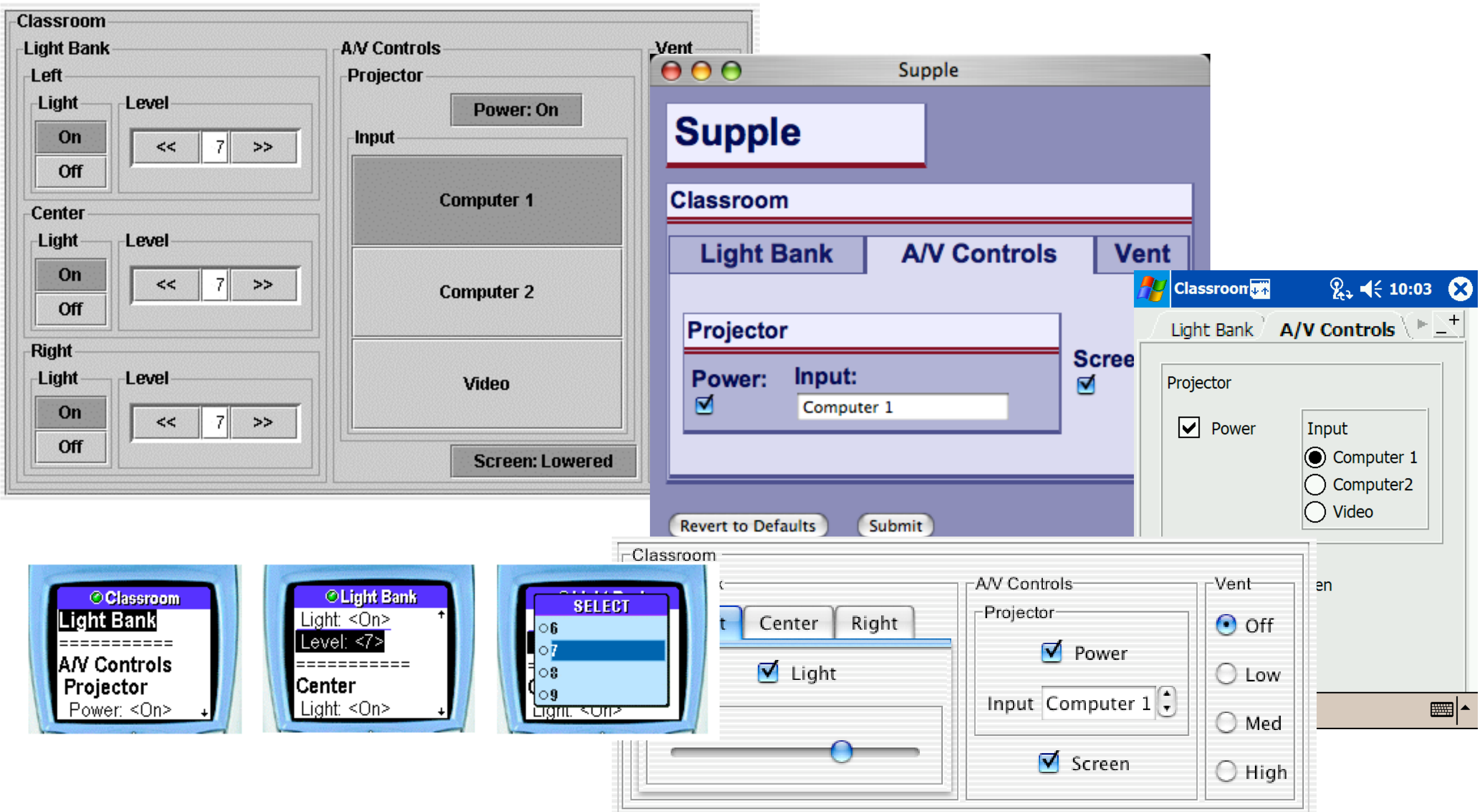


{root, -, ->}
⟨LeftLight:Power, off, on⟩
⟨Vent, 1, 3⟩
⟨Projector:Input, video, computer⟩
... }



Decision Theoretic Optimization

Supple Output



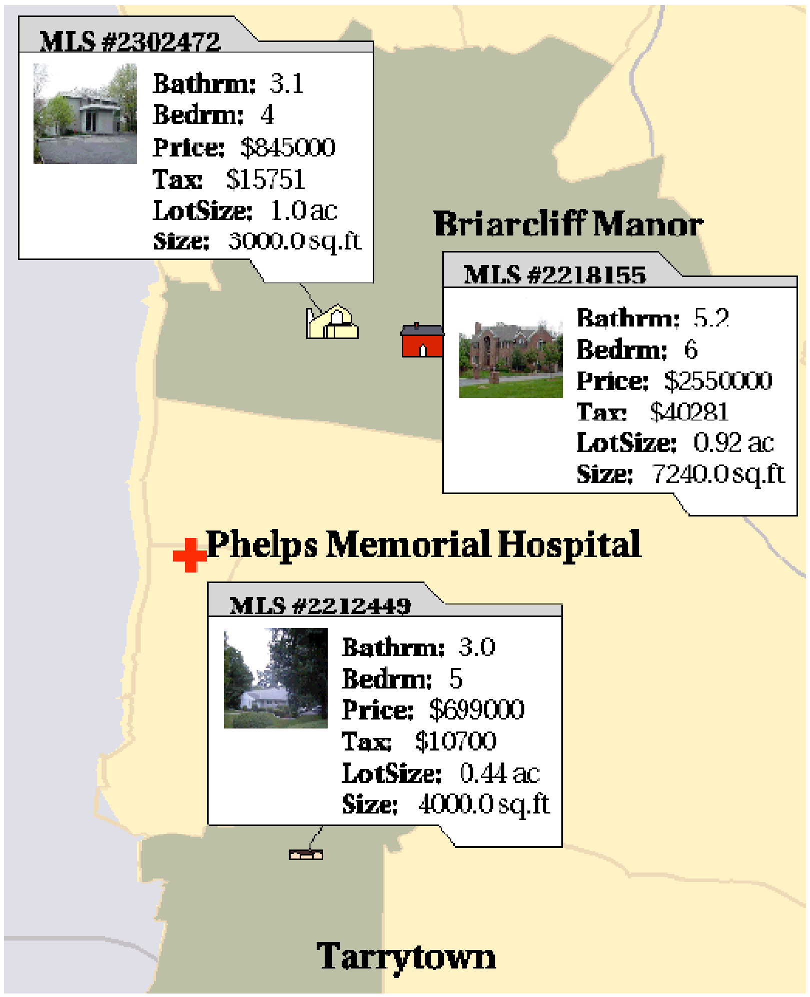
[Gajos & Weld, IUI'04]

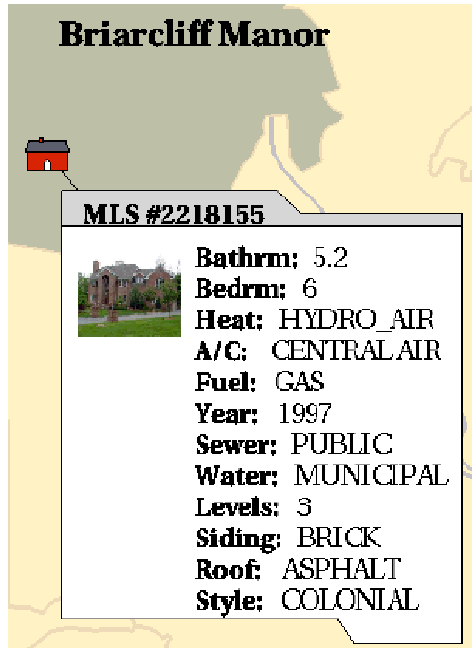
Supple Depends on Weights

Container factor weight:	0.0
Tab Pane factor weight:	100.0
Popup factor weight:	1.0
Spinner for integers factor weight:	5.0
Spinner (domain size) factor weight:	49.5238
Spinner for non-integers factor weight:	6.0
Slider factor weight:	45.7143
Progress bar factor weight:	0.0
Checkbox factor weight:	0.0
Radio button factor weight:	0.5
Horizontal radio button factor weight:	10.0
Radio button (>=4 values) factor weight:	0.0
Radio button (>=8 values) factor weight:	74.2857
Radio button for booleans factor weight:	14.2857

[Gajos & Weld, IUI'04]

RIA





U1	*Speech*: Show houses near Phelps Memorial Hospital
R1	*Speech*: I found 3 houses near Phelps Memorial Hospital *Graphics*: Display (a)
U2	*Speech*: Tell me more about it *Gesture*: Point to the house on the right
R2	*Speech*: Here are the attributes of this 6-bedroom home. *Graphics*: Display (b)

[Zhou +, UIST’04; IUI’05]

$$R(d, U) = \boxed{w_1} \times K(d, U) + \boxed{w_2} \times I(d, U)$$

$$F_3(d) = \sum_i \boxed{u_i} \times R(d, x_i)$$

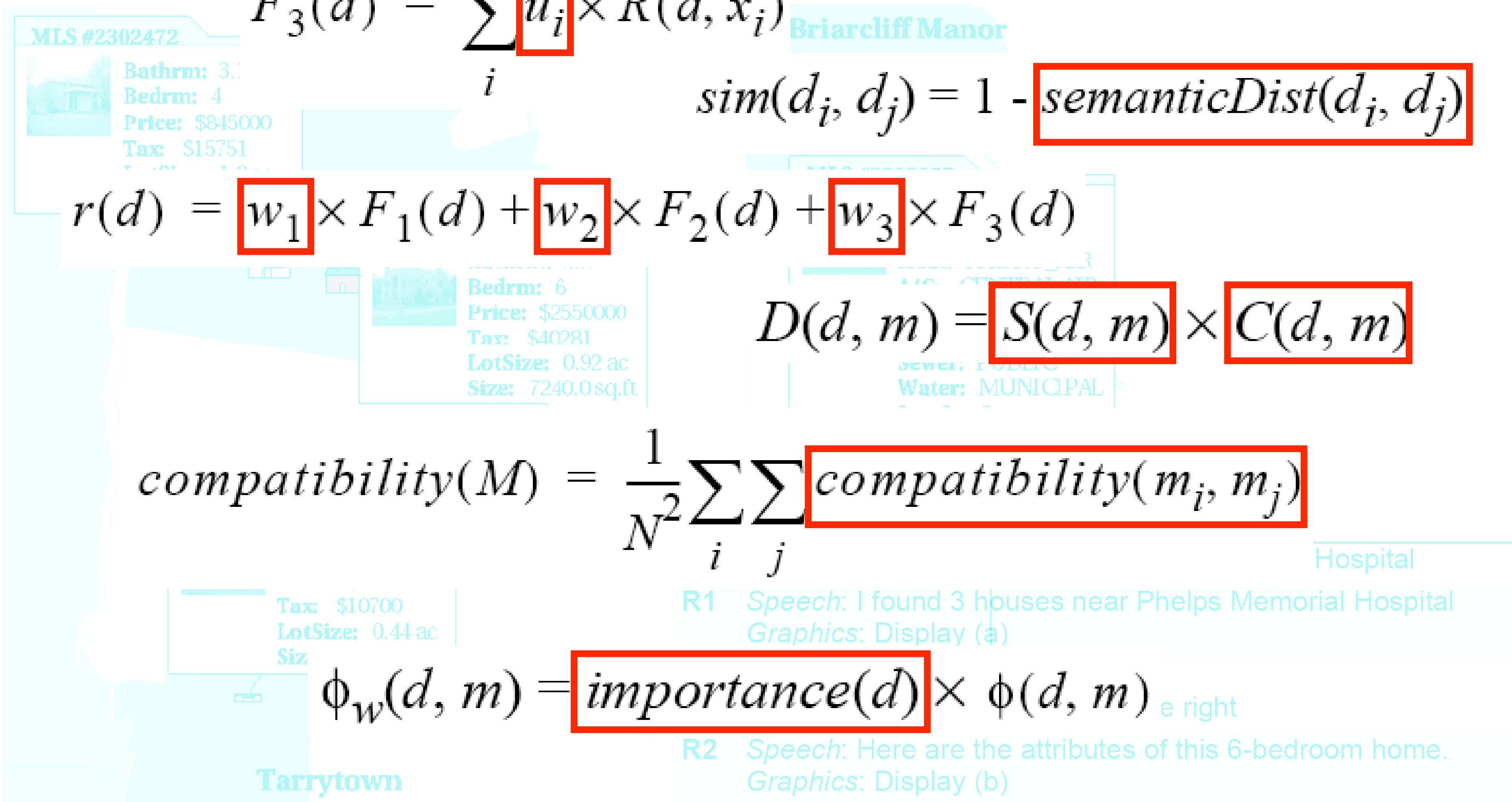
$$sim(d_i, d_j) = 1 - \boxed{semanticDist(d_i, d_j)}$$

$$r(d) = \boxed{w_1} \times F_1(d) + \boxed{w_2} \times F_2(d) + \boxed{w_3} \times F_3(d)$$

$$D(d, m) = \boxed{S(d, m)} \times \boxed{C(d, m)}$$

$$compatibility(M) = \frac{1}{N^2} \sum_i \sum_j \boxed{compatibility(m_i, m_j)}$$

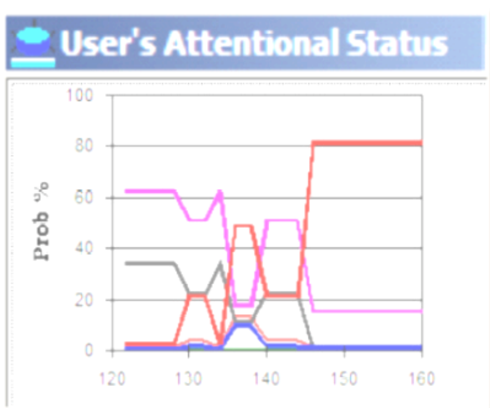
$$\phi_w(d, m) = \boxed{importance(d)} \times \phi(d, m)$$

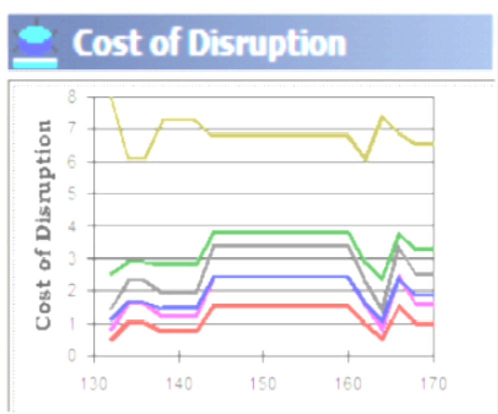


[Zhou +, UIST’04; IUI’05]

BusyBody

$$\underbrace{ECI}_{\text{Expected Cost of Interruption}} = \sum_i \underbrace{p(I_i \mid E)}_{\text{Probability of an interruptability state } I_i} \underbrace{C(I_i)}_{\text{Cost of interrupting if user is in state } I_i}$$





[Horvitz +, CSCW'04]

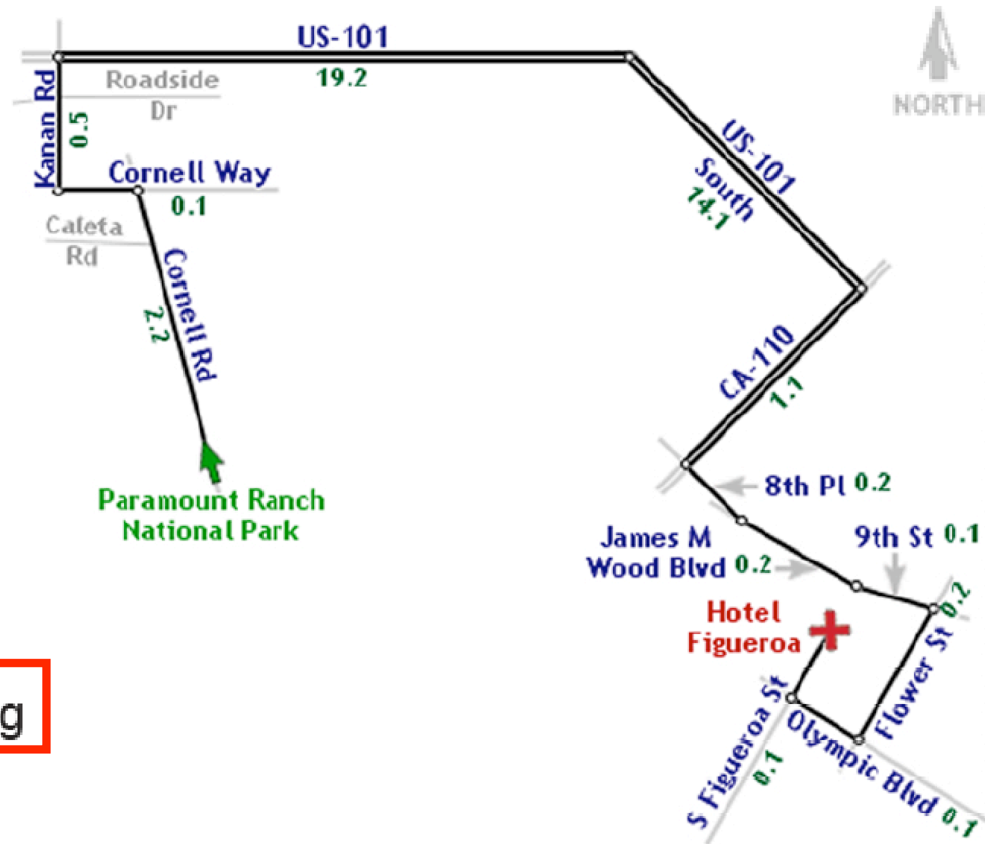
BusyBody

$$\underbrace{ECI}_{\text{Expected Cost of Interruption}} = \sum_i \underbrace{p(I_i \mid E)}_{\text{Probability of an interruptability state } I_i} \underbrace{\boxed{C(I_i)}}_{\text{Cost of interrupting if user is in state } I_i}$$

Needs to be elicited from the user for every interruptability state I_i

[Horvitz +, CSCW'04]

LineDrive



$$score(r_i, r_k) = d * \boxed{W_{misplaced}}$$

$$score(r_i, r_k) = d * \boxed{W_{missing}}$$

$$score(r_i) = |\alpha_{curr} - \alpha_{orig}| * \boxed{W_{orient}}$$

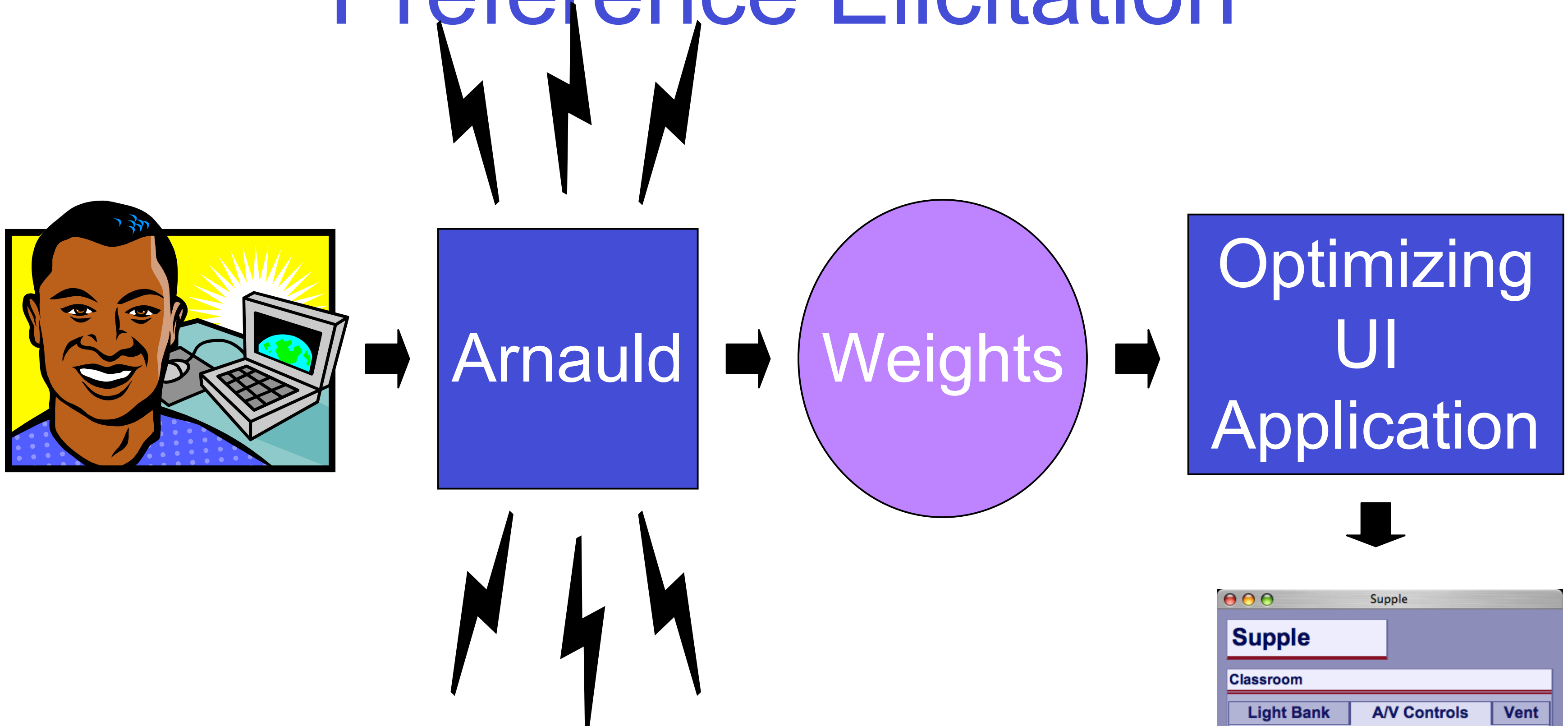
$$score(r_i, r_k) = d * \boxed{W_{extended}}$$

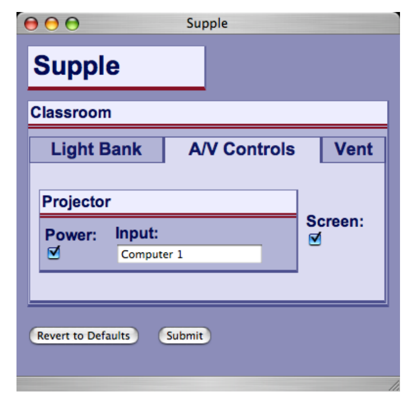
$$score(r_i) = ((l(r_i) - L_{min})/L_{min})^2 * \boxed{W_{small}}$$

$$score(r_i, r_k) = \min(d_{orig}, d_{dest}) * \boxed{W_{false}}$$

[Agrawala +, SIGGRAPH'01]

Arnauld: A Tool for Preference Elicitation



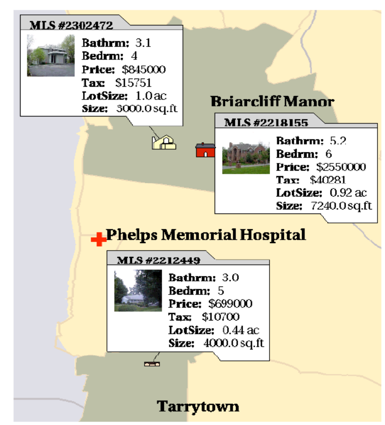


Raises level of abstraction:

- instead of directly choosing weights…,
- designers now interact with concrete outcomes

Arnauld: A Tool for Preference Elicitation

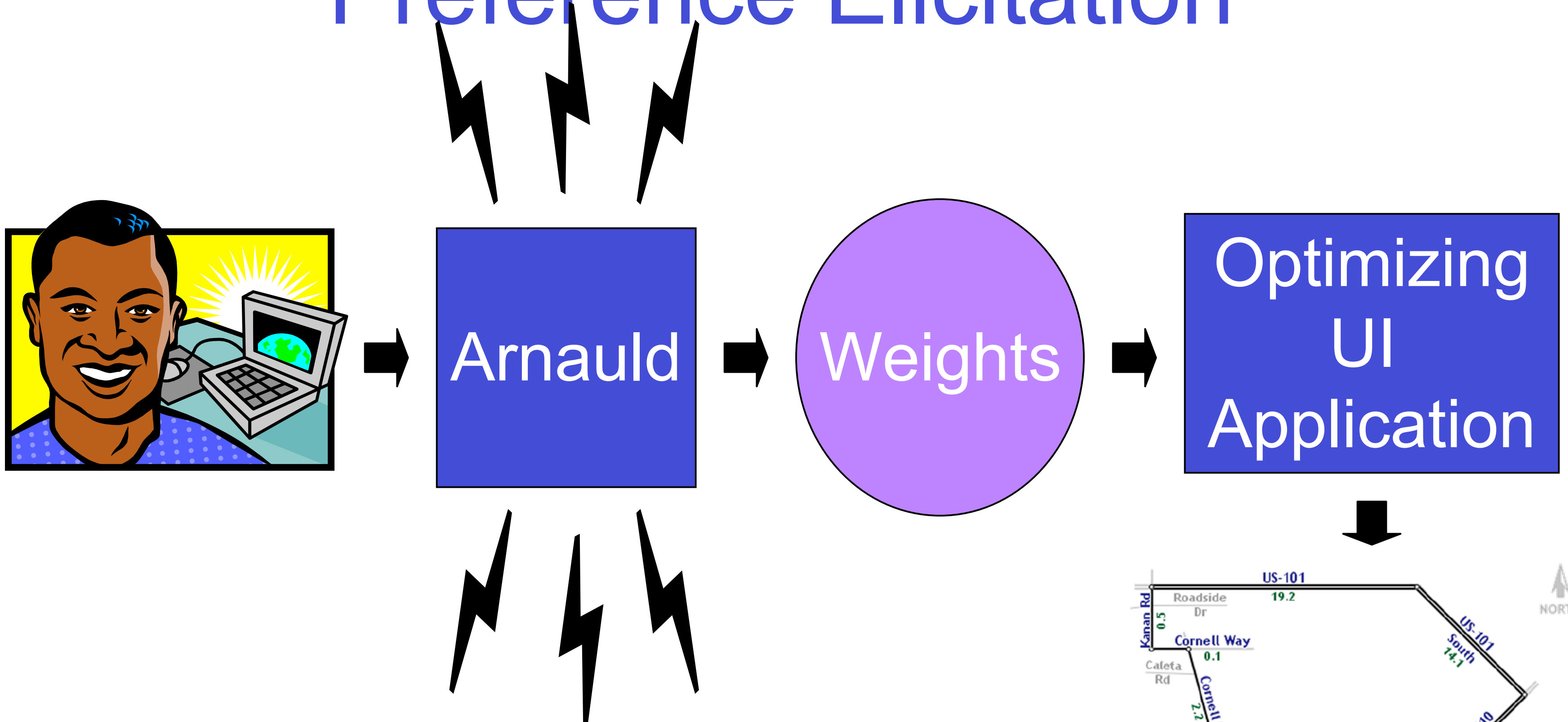
Arnauld → Weights → Optimizing UI Application

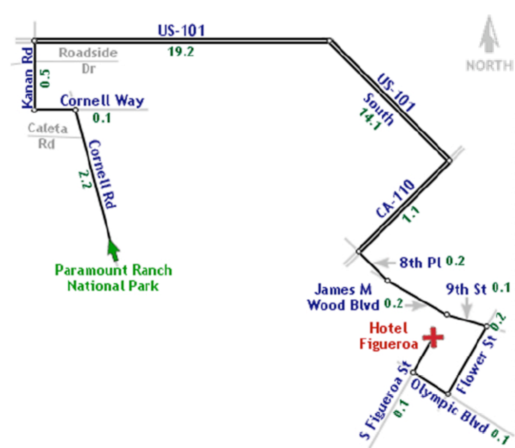


Raises level of abstraction:

- instead of directly choosing weights…,
- designers now interact with concrete outcomes

Arnauld: A Tool for Preference Elicitation

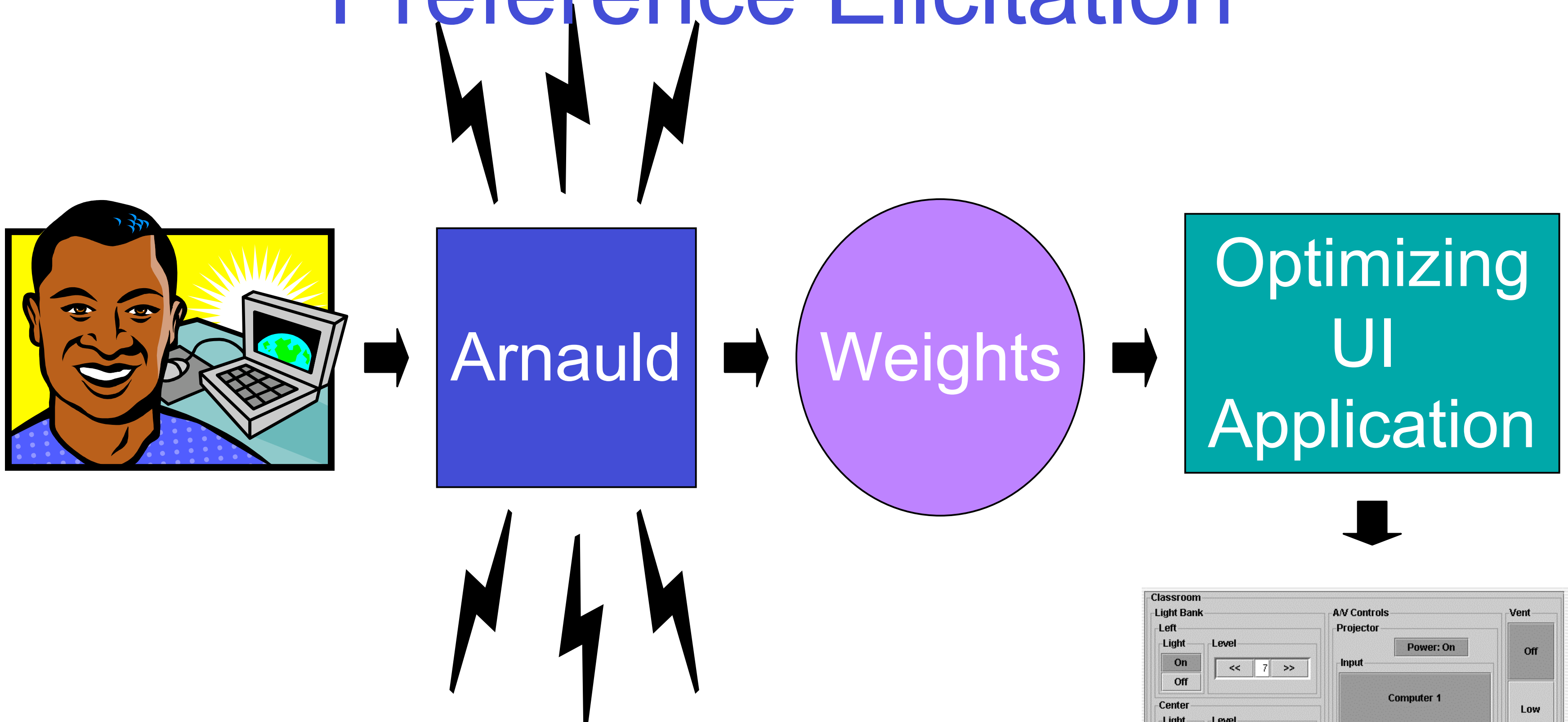


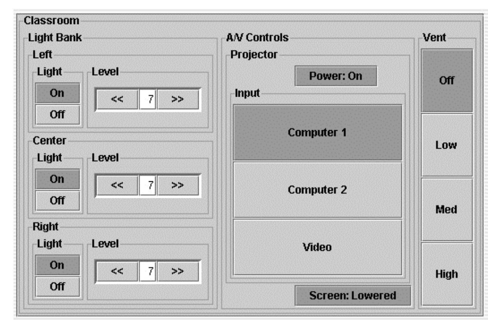


Raises level of abstraction:

- instead of directly choosing weights…,
- designers now interact with concrete outcomes

Arnauld: A Tool for Preference Elicitation





Raises level of abstraction:
- instead of directly choosing weights…,
- designers now interact with concrete outcomes

Benefits

- Saves Developers Time
  - By factor of 2-3x
- Improves Quality of Weights
  - Learned weights out-perform hand-tuned
- Users May Want to Override Default Params
  - Individual preferences
  - Multiple uses

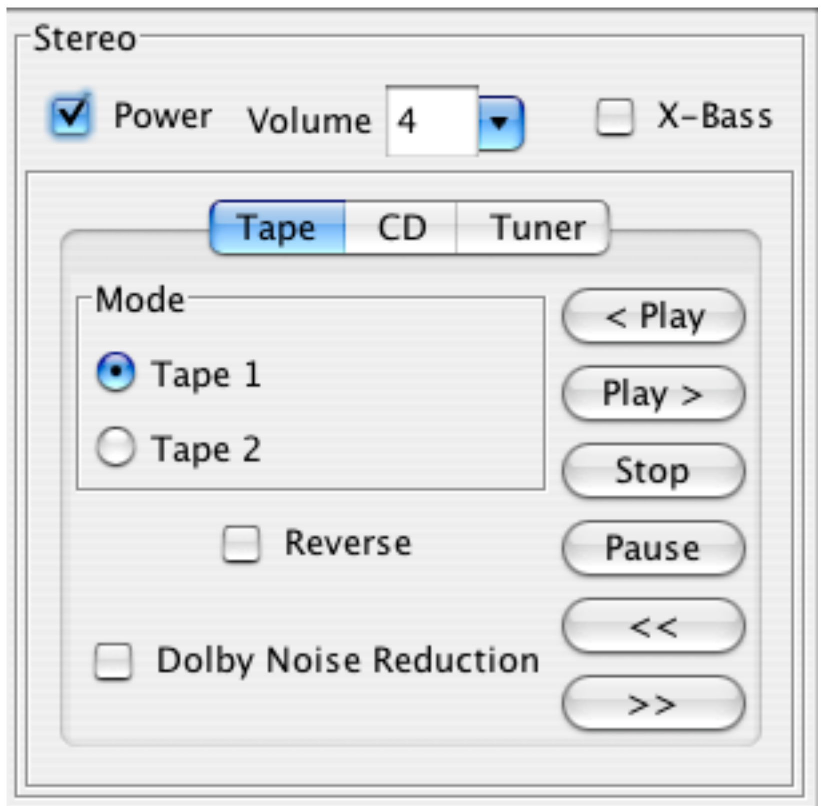
Our Contributions

- Implemented ***Arnauld*** system for preference elicitation
  - Applicable to most optimization-based HCI applications
  - Implemented on SUPPLE
- Based on two ***interaction methods*** for eliciting preferences
- Developed a fast ***machine learning algorithm*** that learns the best set of weights from user feedback
  - Enables interactive elicitation
- Investigated two ***query generation algorithms***
  - Keep the elicitation sessions short

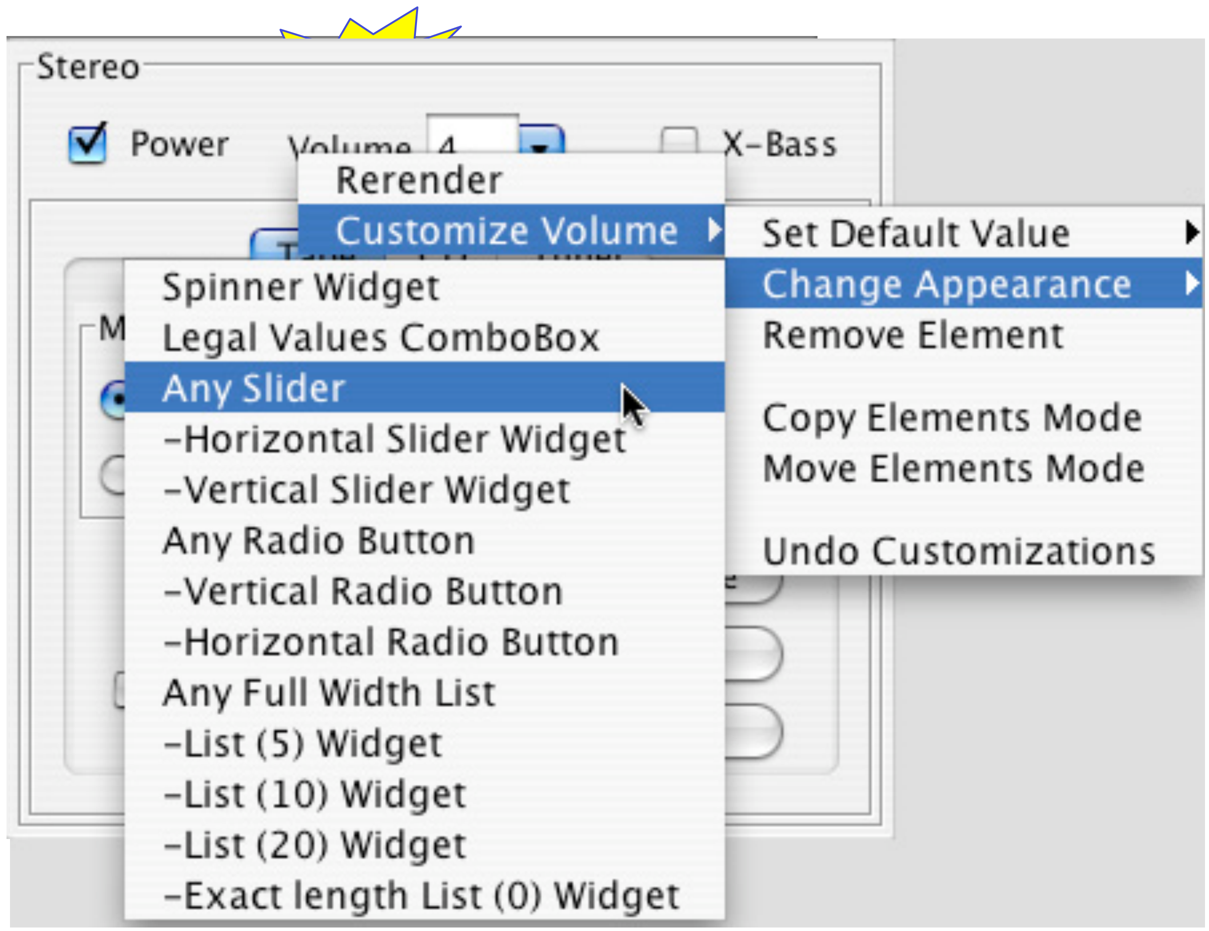
Outline

- Motivation
- **Elicitation techniques**
  - **Example critiquing**
  - **Active elicitation**
- User responses → constraints
- Learning from user responses
- Generating queries
- Results & Conclusions

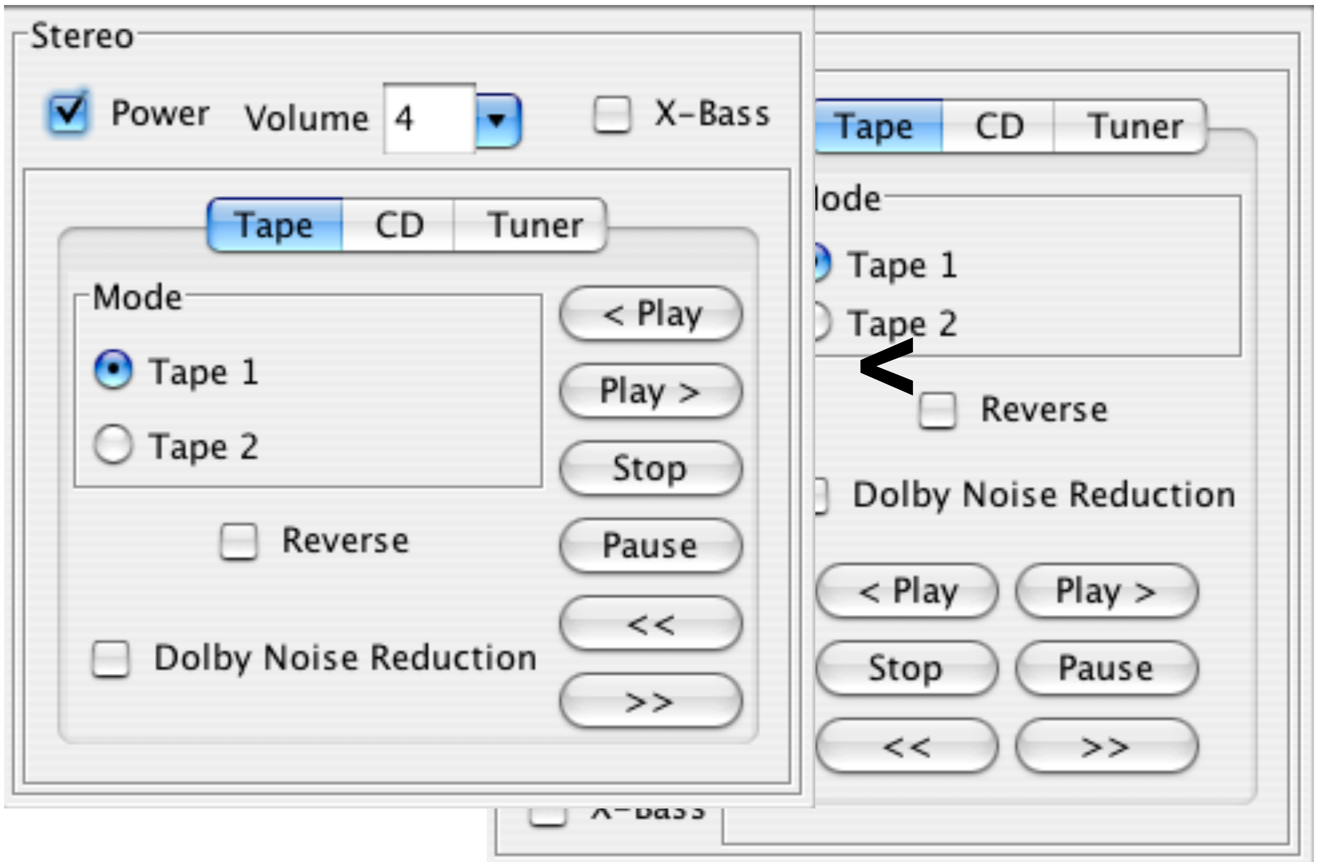
Example Critiquing



Via Customization Facilities



Result of Customization
Provides Training Example!



before

after

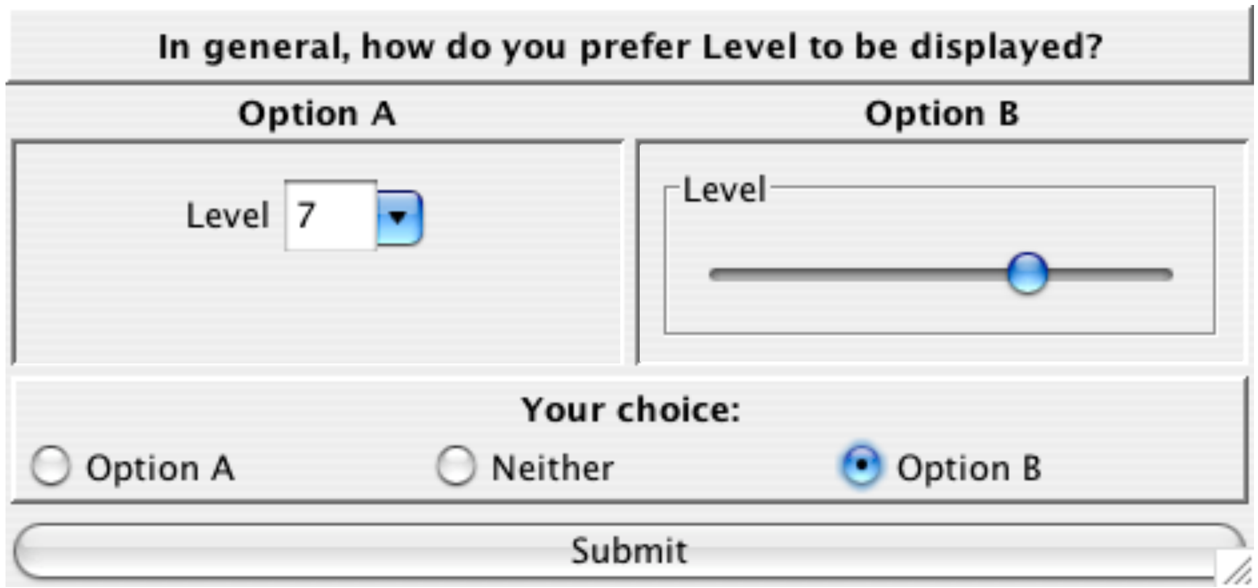
Example Critiquing

- Exploits **natural** interaction
  - Occuring during process of customizing interface
- **Effective** when cost function is almost correct

But…

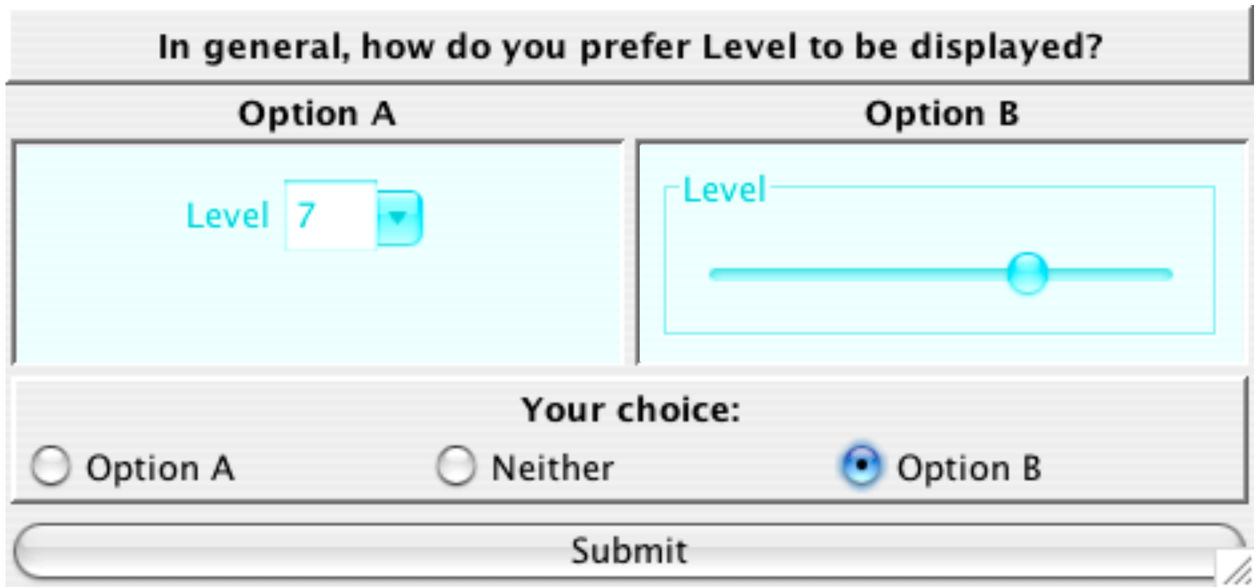
- Can be **tedious** during early stages of parameter learning process
- **Requires** customization support to be provided by the UI system (e.g. RIA, SUPPLE, etc.)

Active Elicitation



Active Elicitation

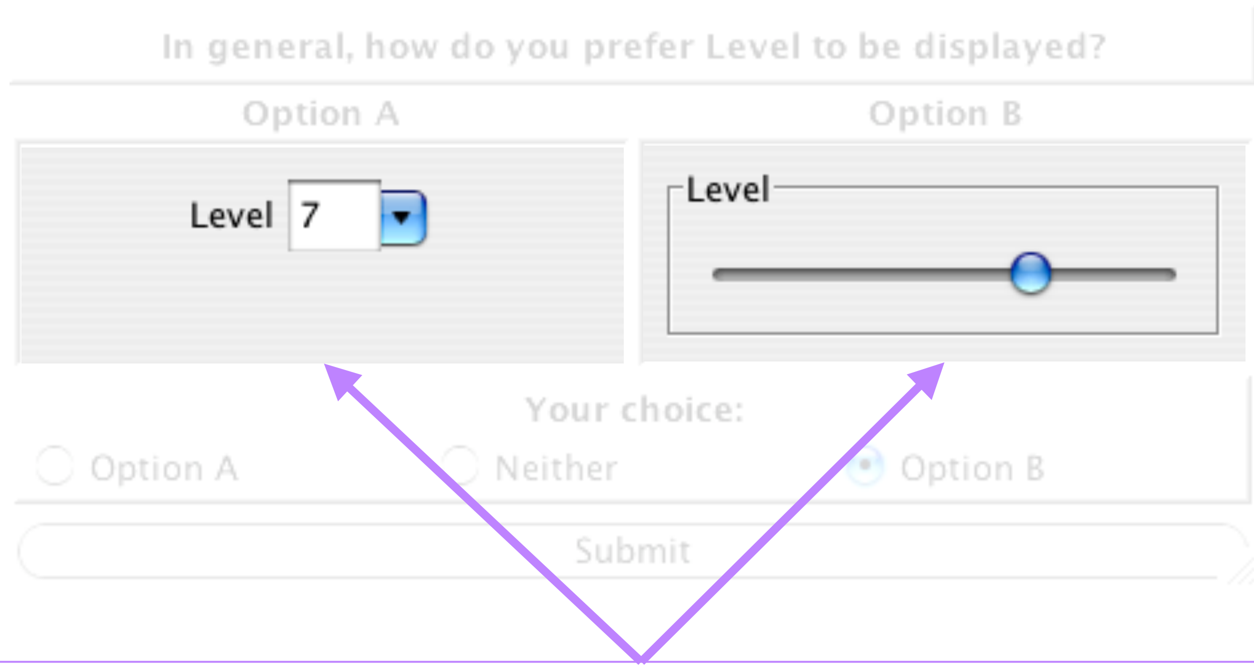
UI in Two Parts



Structure provided by ARNAULD

Active Elicitation

UI in Two Parts



Content provided by the interface system for which we are learning weights

Active Elicitation

- **Convenient** during early stages of parameter learning process
- Binary comparison queries **easy for user**
- **Doesn’t require** any additional support from UI system, for which parameters are generated

But

- **Doesn’t allow** designer to direct learning process
- Choice of Best Question is Tricky

Limitations of Isolated Feedback

Both examples so far provided feedback of the form

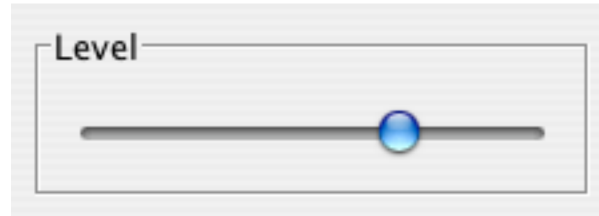
"All else being equal, I prefer sliders to combo boxes"

<

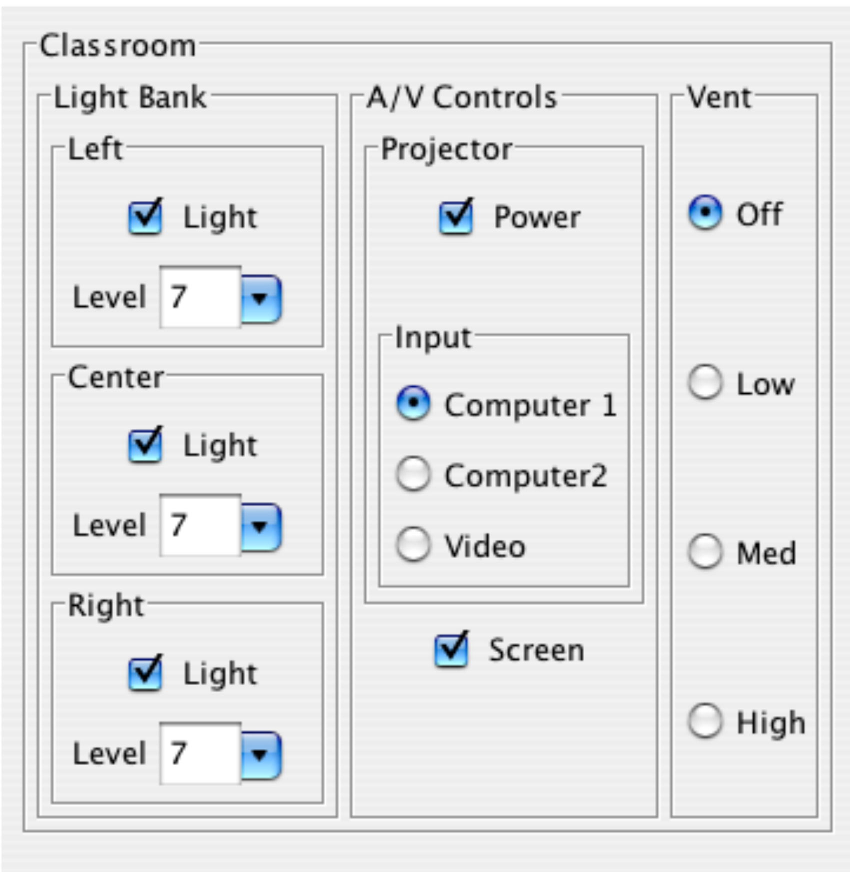
But what if using a better widget in one place
Makes another part of the interface crummy?!

In isolation, sliders are preferred

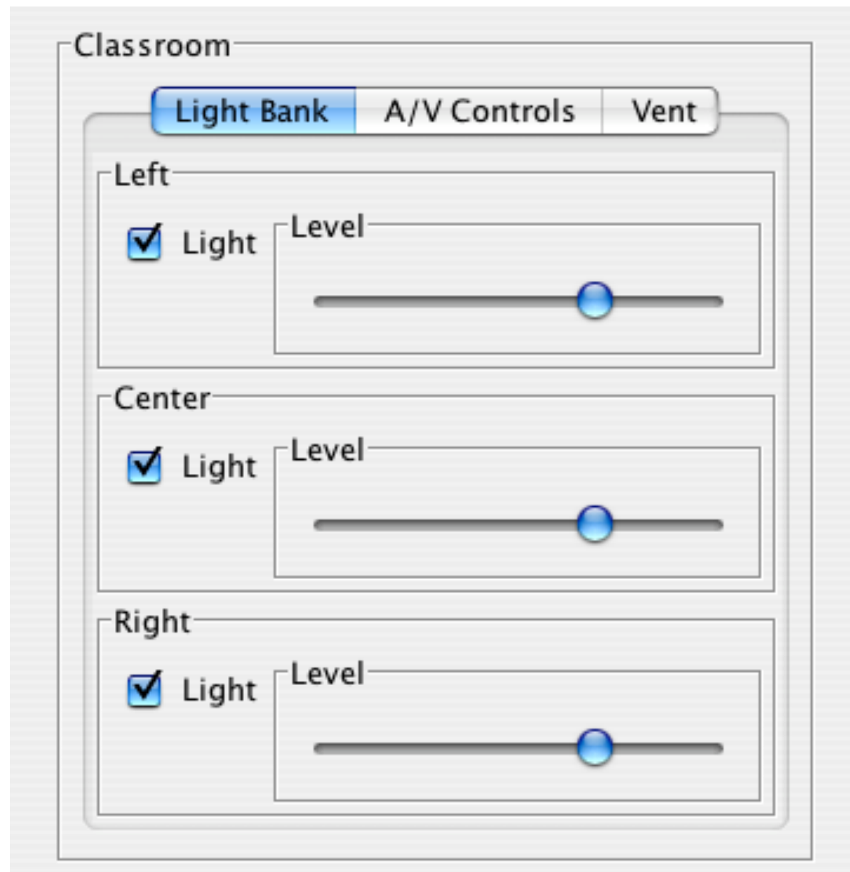
<



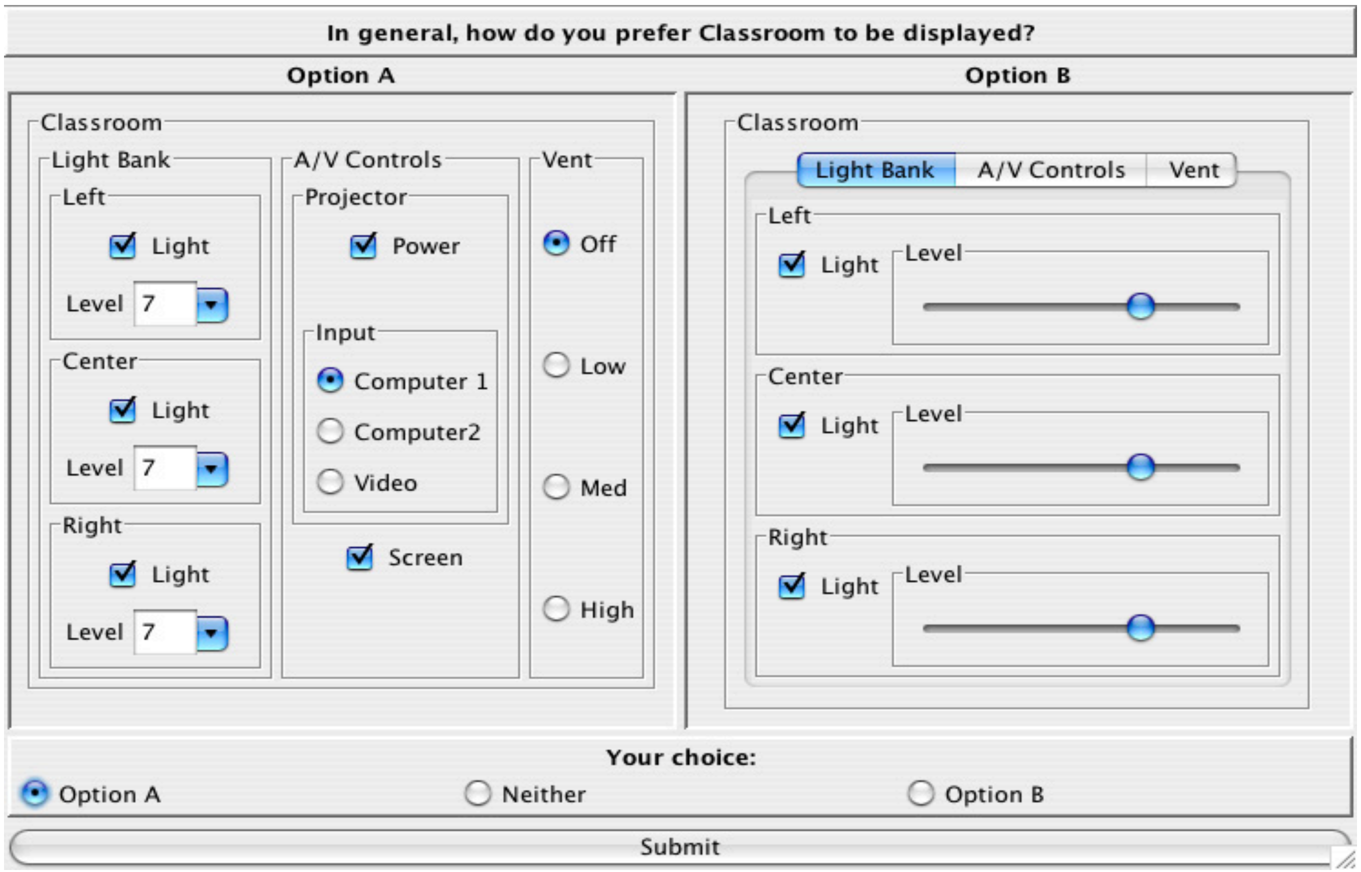
But using them may cause badness elsewhere



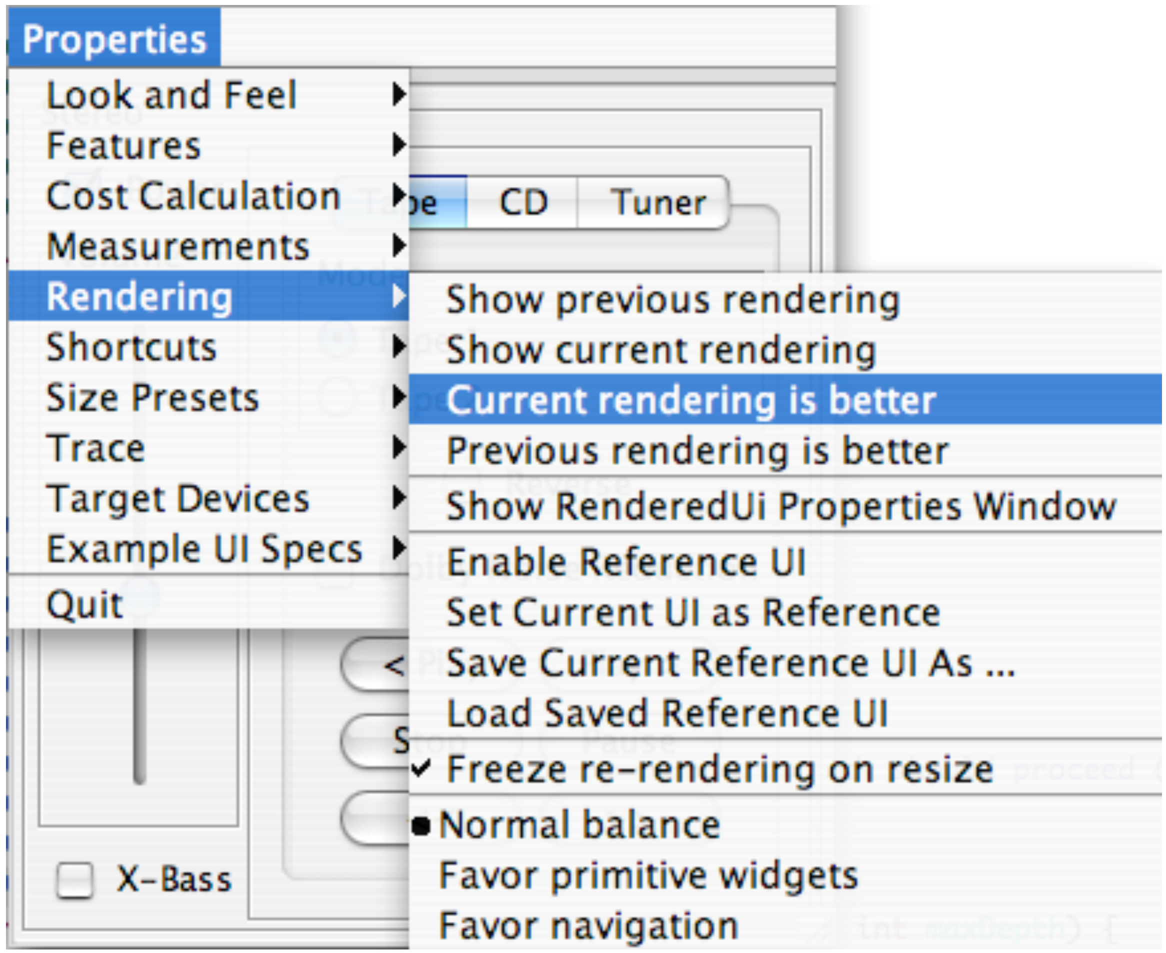
>



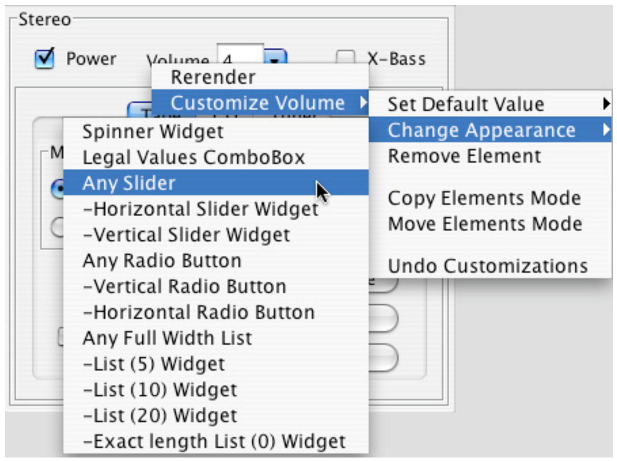
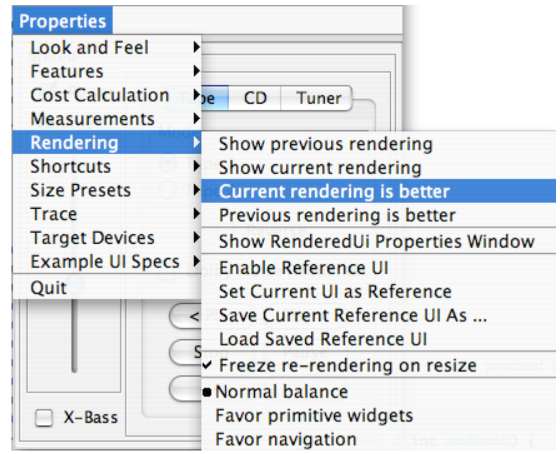
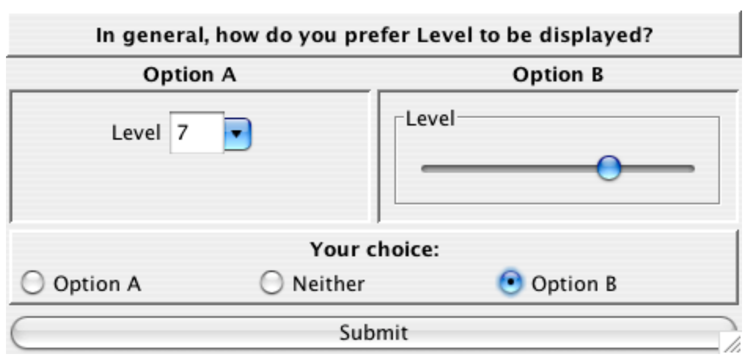
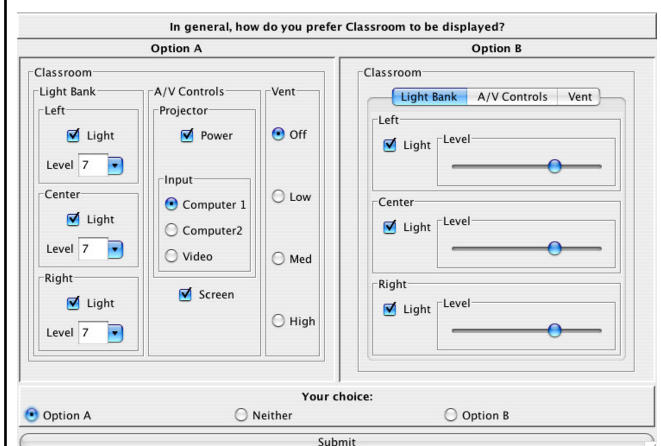
Situated Feedback
with Active Elicitation



Situated Feedback
with Example Critiquing



Summary of Elicitation Interactions

	Isolated	Situated
Example Critiquing		
Active Elicitation		

Outline

- Motivation
- Elicitation techniques
- **User responses → constraints**
- Learning from user responses
- Generating queries
- Results & Conclusions

Turning User Responses Into Constraints

All systems studied had **linearly decomposable** cost functions; these can be expressed as:

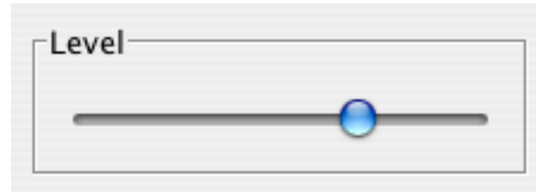
$$\text{cost}(interface) = \sum_{k=1}^{K} \boxed{u_k} \boxed{f_k(interface)}$$

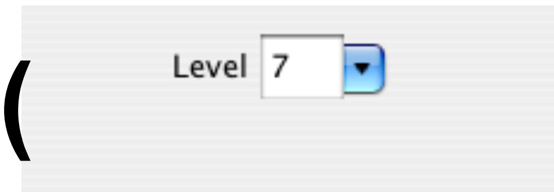
A ***weight*** associated with a factor

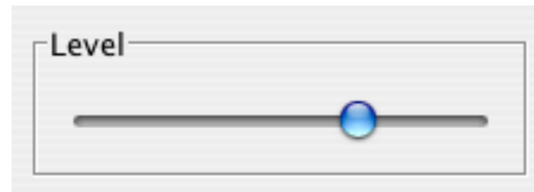
A "***factor***" reflecting presence, absence or intensity of some interface property

From User Responses to Constraints

<



cost(

) ≥ cost(

)

$$f_{combo\_box} = 1$$

$$f_{combo\_box\_for\_number} = 1$$

$$f_{slider} = 1$$

$$f_{horizontal\_slider} = 1$$

$$u_{combo\_box} + u_{combo\_box\_for\_number} \geq u_{slider} + u_{horizontal\_slider}$$

$$\sum_{k=1}^{K} u_k f_k(interface_1) \geq \sum_{k=1}^{K} u_k f_k(interface_2)$$

Outline

- Motivation
- Elicitation techniques
- User responses → constraints
- **Learning from user responses**
- Generating queries
- Results & Conclusions

Learning Algorithm

Given constraints of the form:

$$\sum_{k=1}^{K} u_k f_k(interface_1) \geq \sum_{k=1}^{K} u_k f_k(interface_2)$$

Find values of weights u_k

Satisfying a maximum number of constraints

And by the greatest amount

Our Approach

Use a ***max-margin*** approach

- Essentially a linear Support Vector Machine

Reformulate constraints:

$$\sum_{k=1}^{K} u_k f_k(interface_1) - \sum_{k=1}^{K} u_k f_k(interface_2) \geq margin + slack_i$$

Our Approach

Use a ***max-margin*** approach

Essentially a linear Support Vector Machine

Shared margin by which all constraints are satisfied

Reformulate constraints:

$$\sum_{k=1}^{K} u_k f_k(interface_1) - \sum_{k=1}^{K} u_k f_k(interface_2) \geq \boxed{margin} + \boxed{slack_i}$$

Per-constraint slack that accommodates unsatisfiable constraints

Learning as Optimization

Set up an optimization problem that maximizes:

$$margin - \sum_i slack_i$$

Subject to the constraints:

$$\sum_{k=1}^{K} u_k f_k(interface_1) - \sum_{k=1}^{K} u_k f_k(interface_2) \geq margin + slack_i$$

Learning as Optimization

Set u … n that maximizes:

Solved with standard
linear programming methods
in less than 250 ms.

$$\sum_i slack_i$$

Subject to the constraints:

$$\sum_{k=1}^{K} u_k f_k(interface_1) - \sum_{k=1}^{K} u_k f_k(interface_2) \geq margin + slack_i$$

Outline

- Motivation
- Elicitation techniques
- User responses → constraints
- Learning from user responses
- **Generating queries**
- Results & Conclusions

Generating Queries

- Important part of ***Active Elicitation***
  - Like game of 20 questions, order is key
- Optimality is intractable
- Introducing two heuristic methods
  - Searching $\Re^n$ space of weights
    - General method: applies to all opt-based UI
  - Search space of semantic differences
    - Faster
    - Requires tighter integration with the UI appl'ctn

Generating Queries

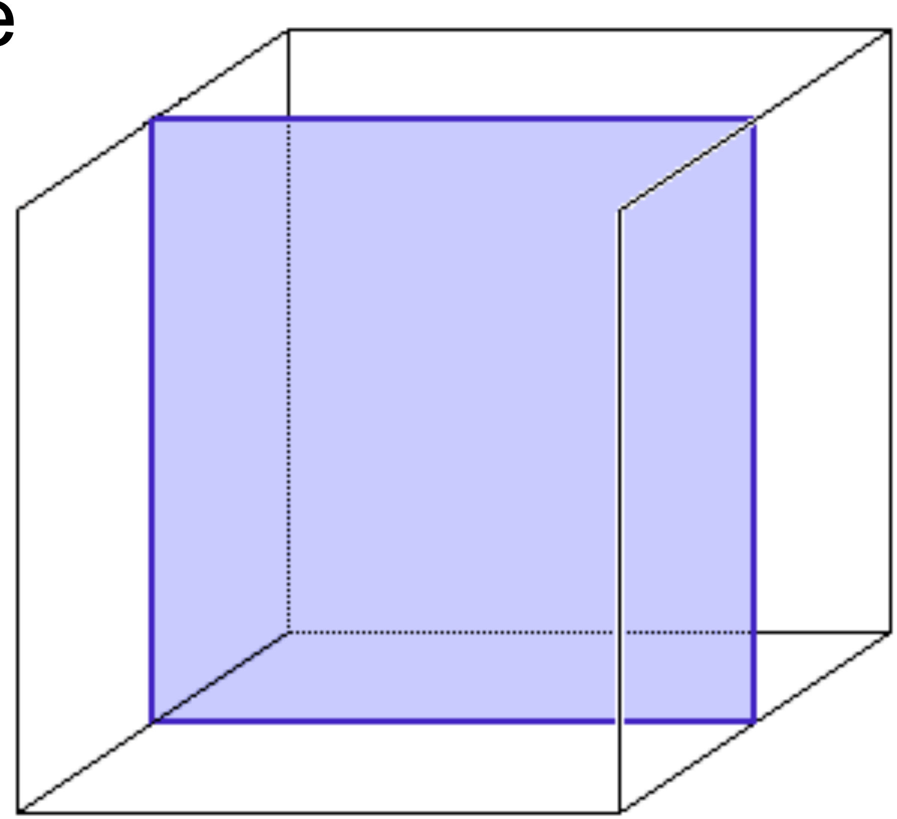
- Why is it important?
  - Like game of 20 questions, order is key
- Optimality is intractable
- Introducing two heuristic methods
  - Searching $\Re^n$ space of weights
    - General method: applies to all opt-based UI
  - Search space of semantic differences
    - Faster
    - Requires tighter integration with the UI appl'ctn

Visualizing the search thru $\Re^n$ space of weights

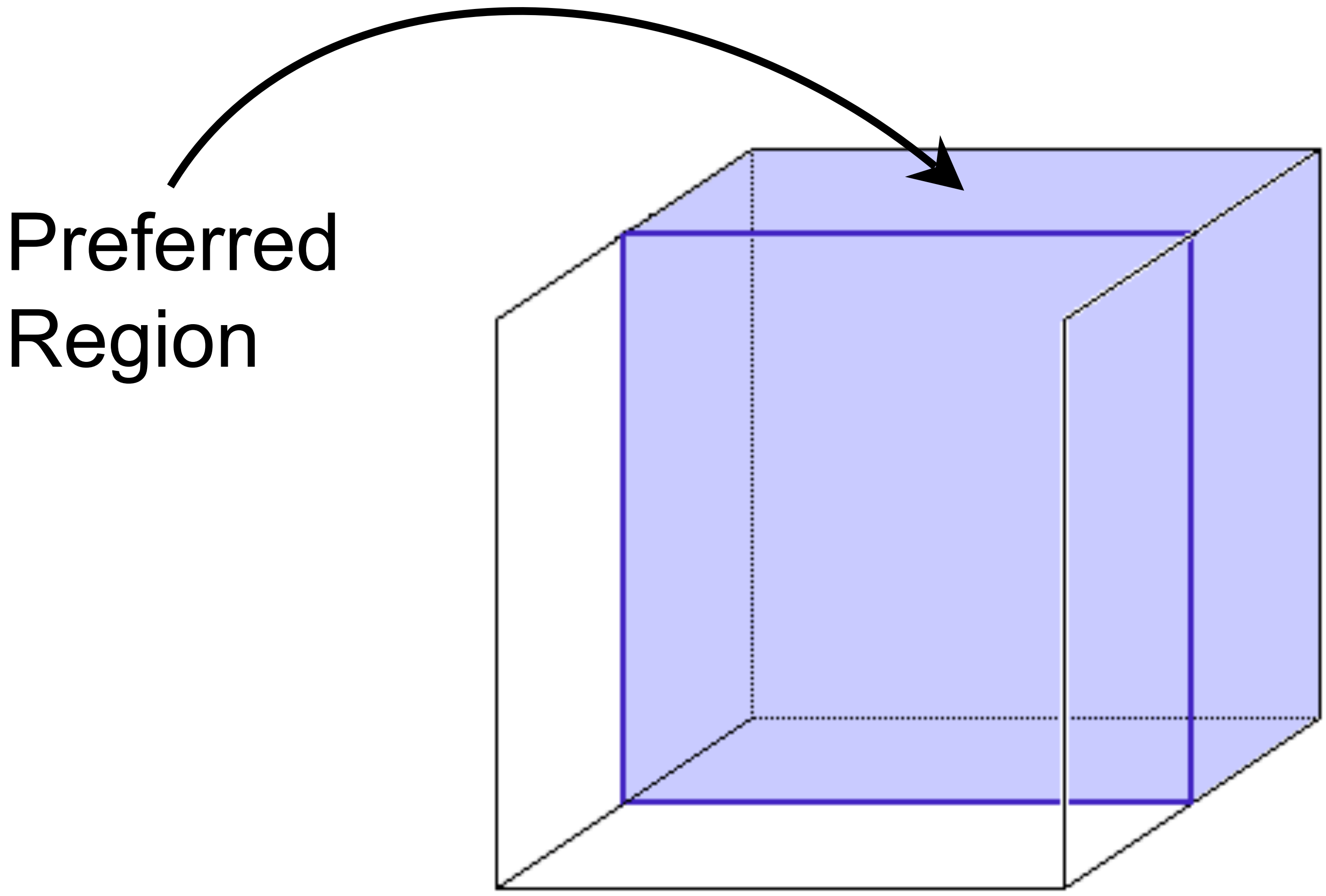
A binary preference question

cleaves

the space

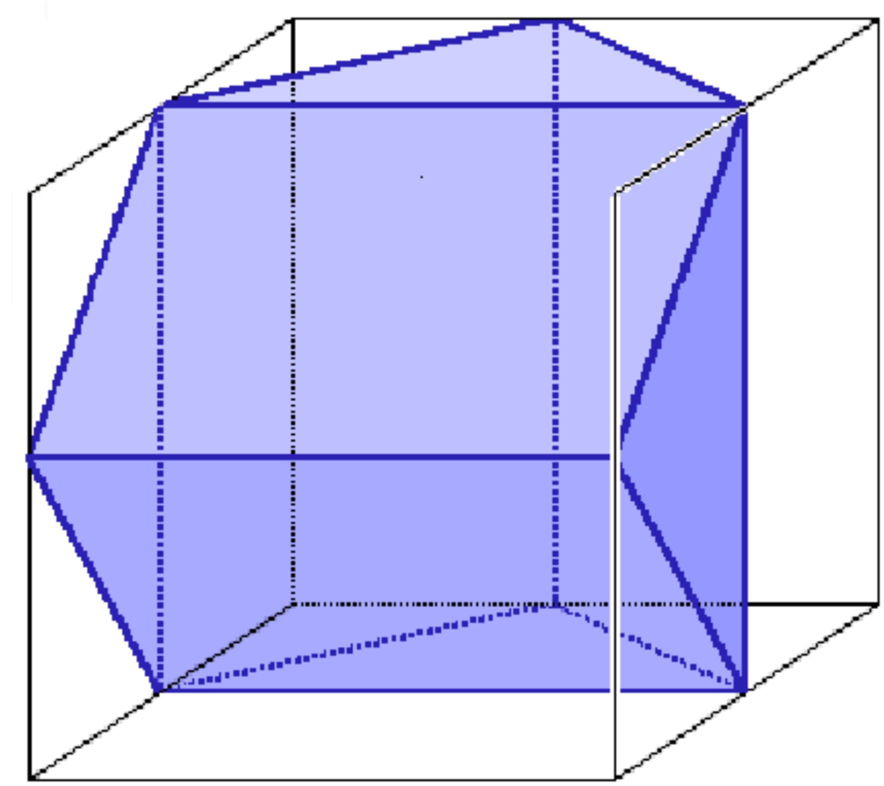


Answering Question Creates ***Region***



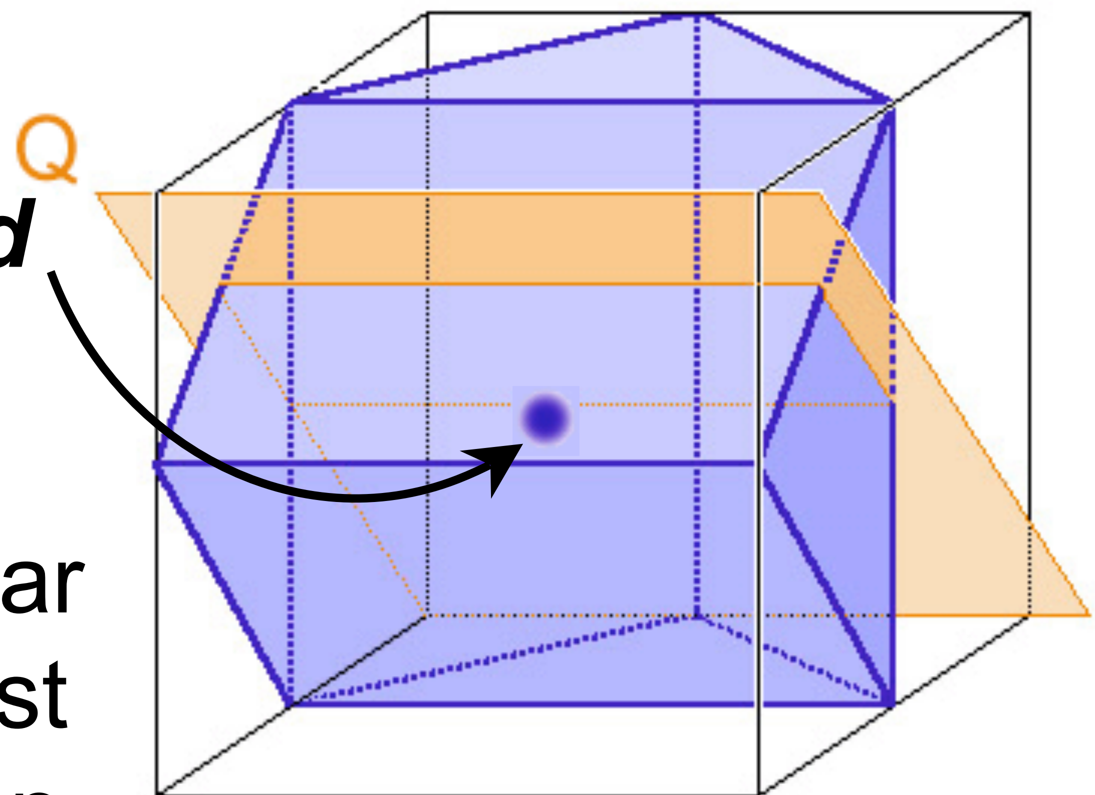
Midway thru the Q/A Process…

What is the best ***immediate*** (greedy) question for cleaving?



Good Heuristics for Cleaving

1. As close to the ***centroid*** as possible



2. Perpendicular to the longest axis of region

Outline

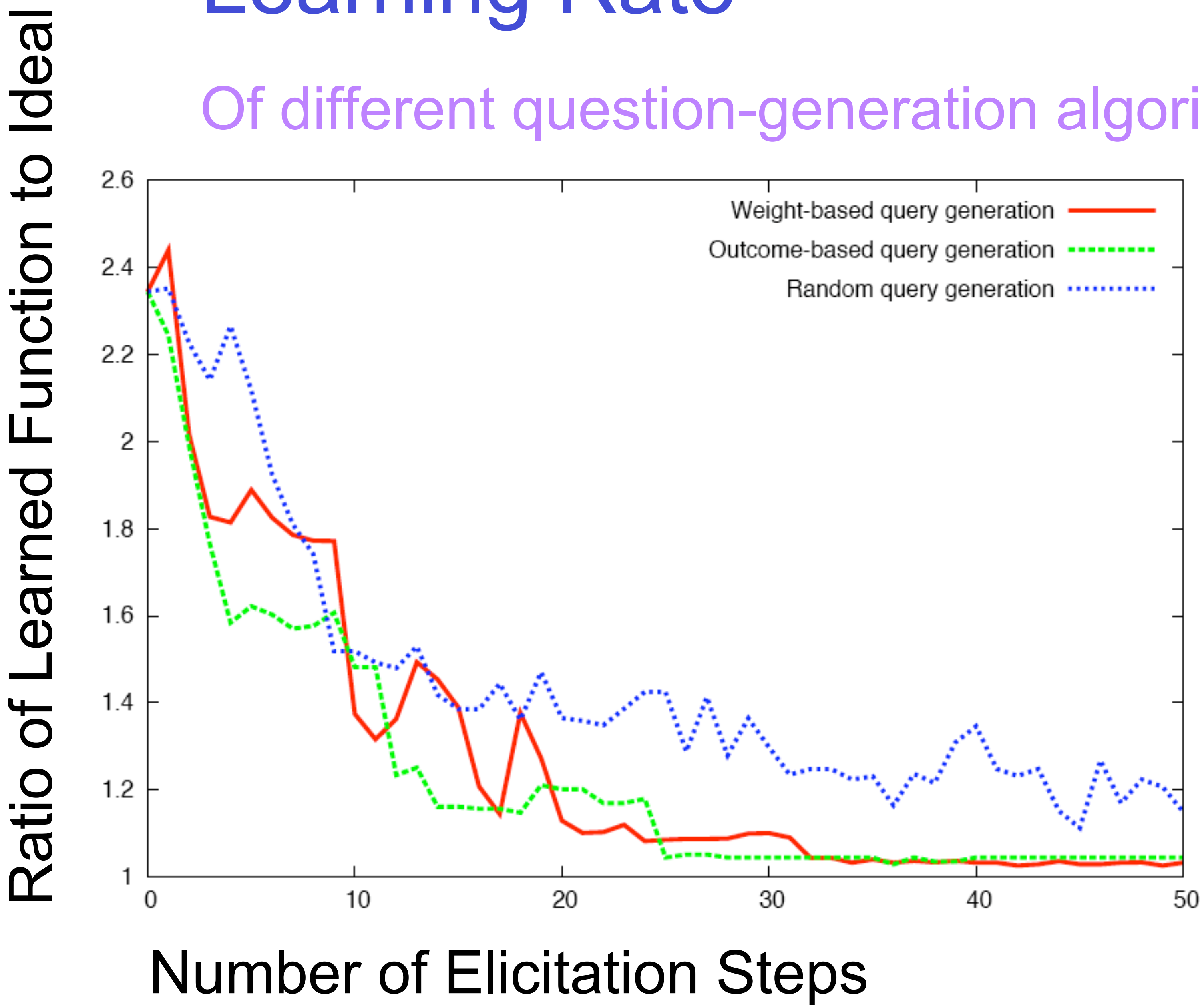
- Motivation
- Elicitation techniques
- User responses → constraints
- Learning from user responses
- Generating queries
- **Results & Conclusions**

Informal User Study

- Four users
  - Two Supple developers
  - Two “sophisticated users”
    - I.e. programmers w/o Supple experience
- Developers asked to hand-build cost function
  - Hand-coding took 2-3x longer
  - Resulting function “wrong” 35% of the time!
- Using Arnauld to create cost function
  - Got robust cost function in 10-15 minutes
  - All said Arnauld much easier & more accurate

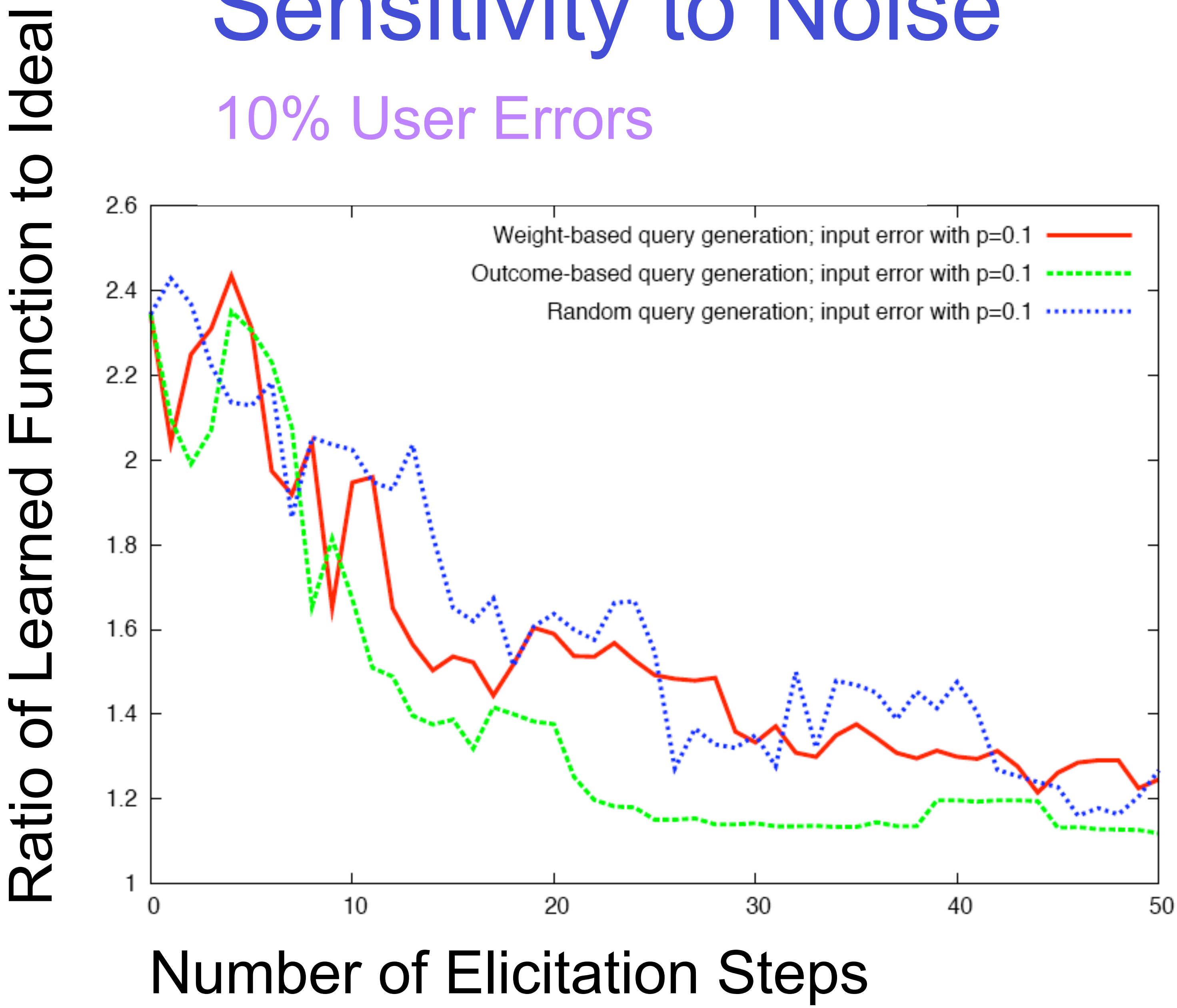
Learning Rate

Of different question-generation algorithms



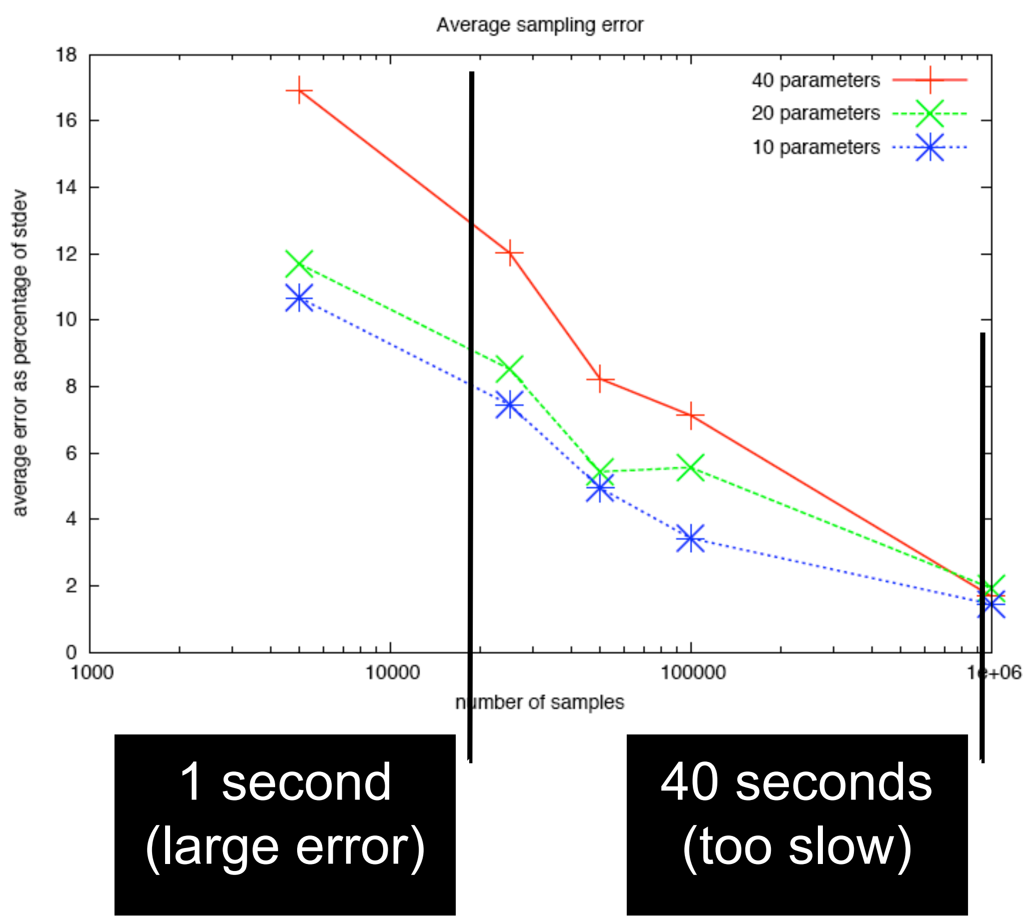
Sensitivity to Noise

10% User Errors



Related Work

- Gamble Queries
  - Outcome$_x$ *vs.* *p*Best + (1-*p*)*Worst*
- Bayesian Learning
  - [Chajewska, ICML'01]
  - Too slow for interactive use



Conclusions

- Implemented ***Arnauld*** system for preference elicitation
  - Applicable to most optimization-based HCI applications
  - Saves developers time
  - Creates better weights
- Based on ***two interaction methods***
  - ***Example Critiquing***
  - ***Active Elicitation***
  - Investigated two ***query generation algorithms***
- Novel ***machine learning algorithm***
  - Learns good weights from user feedback
  - Fast enough for interactive elicitation