Coactive Learning
Rohan Batra, Avishek Dutta, Nand Kishore, Siddharth Murching
Roadmap

1. Introduction
2. Related Work
3. Coactive Learning Model
4. Coactive Learning Algorithms
5. Experiments
6. Robots
Introduction

● Interaction between humans and most systems today:
  ○ User issues a command
  ○ User receives a result
  ○ User interacts with the results

● In this way, the user provides implicit feedback about his/her utility function
Web Search

- User types the query ‘Who is Yisong Yue?’
- Search engine presents the ranking to the right
- User clicks on documents B and D
Movie Recommendation

- User watches a number of movies
- Netflix makes recommendations
- User rents movie D after viewing all options
Machine Translation

- User requests an online machine translator to translate a wiki page from language A to B
- System returns translated page
- User manually corrects some of the translated text
Implicit Feedback

- In all the previous examples, the user provides implicit feedback

- Typically, this feedback is only an incremental improvement
  - Web page B is better than web page A

- Often very difficult for a human to specify the optimal result/prediction
  - Web pages should be ranked B,D,C,A,E
Key Contributions

1. Formulate Coactive Learning as a model of interaction between a learning system and its user
   - Define notion of regret
   - Validate assumption of implicit user feedback

2. Derive learning algorithms for the Coactive Learning Model
   - Linear utility models
   - Convex cost functions
   - Show $O\left(\frac{1}{\sqrt{T}}\right)$ regret bounds

3. Provide empirical evaluations of algorithms
   - Web search

4. Robotic Application
Roadmap

1. Introduction
2. Related Work
3. Coactive Learning Model
4. Coactive Learning Algorithms
5. Experiments
6. Robots
Related Work

- The Coactive Learning Model bridges the gap between two previously studied forms of feedback
  - Multi-armed Bandit Model
    - We choose one arm and learn the utility of that arm only
  - Learning with Experts Model
    - We choose one arm, but learn the utility of all arms
- Our model reveals information about two arms at every timestep
Related Work

- Both the aforementioned models can be relaxed to the continuous setting
  - Multi-armed Bandit Model
    - Online Convex Optimization in the Bandit Setting
  - Learning with Experts Model
    - Online Convex Optimization

- Our model explores the continuous case with the Convex Preference Perceptron Algorithm
Related Work

- The most closely related problem is the Dueling Bandits Problem.
- Recall that in this setting, the learning algorithm presents two arms to the user in some interleaved format.
- The user gives implicit feedback that can be used to construct a pairwise ordering of the arms.
- Our setting is different in that only one arm is given to the user.
- The other is implicitly determined from user feedback.
  - More details about this when we explain the model and experimental results.
### Comparison of Similar Frameworks

<table>
<thead>
<tr>
<th>Framework</th>
<th>Algorithm</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandits Experts</td>
<td>pull an arm</td>
<td>observe cardinal reward for the arm pulled</td>
</tr>
<tr>
<td>Dueling Bandits</td>
<td>pull two arms</td>
<td>observe feedback on which one is better</td>
</tr>
<tr>
<td>Coactive Learning</td>
<td>pull an arm</td>
<td>observe another arm which is better</td>
</tr>
</tbody>
</table>
Roadmap

1. Introduction
2. Related Work
3. Coactive Learning Model
4. Coactive Learning Algorithms
5. Experiments
6. Robots
In the coactive learning model, both the human and the learning algorithm have the same goal of obtaining good results.

In each round $t$:
- Learning algorithm observes context $x_t \in \mathcal{X}$
- Algorithm presents a structured object
- User “returns” an improved object $\tilde{y}_t \in \mathcal{Y}$
The utility of $y_t$ within the context $x_t$ is given as the unknown function $U(x_t, y_t)$.

Generally, the learned feedback, $\bar{y}_t$ satisfies:

$$U(x, \bar{y}_t) > U(x, y_t)$$

We will also allow violations of this condition through $\alpha$-informative user feedback.
● Our algorithm should return objects with utility close to that of the optimal

● Thus, if our algorithm presents object $y_t$ under context $x_t$ at time $t$, it suffers a regret:

$$\sum_{t=1}^{T} U(x_t, y^*_t) - U(x_t, y_t)$$

● The average regret over $T$ steps is:

$$REG_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y^*_t) - U(x_t, y_t)$$

● Goal of the algorithm is to minimize this average regret
User generates feedback $\bar{y}_t$ through an approximate utility-maximizing search over a subset $\tilde{Y}_t$ of $Y$.

Usually, the returned feedback is **NOT** the optimal (unobservable) label $y^*_t := \arg\max_{y \in Y} U(x_t, y)$.

Thus, we model settings where:
- The user searches using various tools (i.e. query reformulation, browsing)
- The user cannot manually optimize the argmax

Model assumes that reliable preference feedback can be derived from observable user behavior.
Quantifying User Feedback

- Need to quantify how much improvement $\bar{y}_t$ provides in the utility space
- Not needed for algorithm, but necessary for theoretical analysis

- Simplest case: strictly $\alpha$-informative
  - $\alpha \in (0, 1]$ is an unknown parameter

\[
U(x_t, \bar{y}_t) - U(x_t, y_t) \geq \alpha (U(x_t, y^*_t) - U(x_t, y_t))
\]

- Utility of $\bar{y}_t$ is higher than that of $y_t$ by a fraction $\alpha$ of the max possible utility range
Violations of the above feedback model are allowed by introducing slack variables $\xi_t \geq 0$
  ○ Quantifies to what extent the strict $\alpha$-informative modeling assumption is violated

$$U(x_t, \bar{y}_t) - U(x_t, y_t) \geq \alpha(U(x_t, y^*_t) - U(x_t, y_t)) - \xi_t$$

Refer to this model as simply $\alpha$-informative feedback

Our regret bounds in the next sections will contain the term $\xi_t$ and $\alpha$
  ○ Note that we can express feedback of any quality, even the strict case by choosing $\xi_t = 0$
Now we consider the expected \( \alpha \)-informative feedback

○ Even weaker feedback model
○ Positive utility gain is only achieved in expectation over user actions

\[
E_t \left[ U(x_t, \bar{y}_t) - U(x_t, y_t) \right] \geq \alpha (U(x_t, y_t^*) - U(x_t, y_t)) - \bar{\xi}_t
\]

● Expectation is over the user’s choice of \( \bar{y}_t \) given \( y_t \) under context \( x_t \)
  ○ i.e. under a distribution \( P_{x_t}[\bar{y}_t \mid y_t] \)

● Allows for analysis of expected regret
User Study: Preferences from Clicks

- Experimentally validate that users actually exhibit a preference for predictions with higher utility (implicit feedback is reflective of changes in utility).

- Preference feedback is from clicks in web-search.

- Asked subjects to answer 10 questions using Google search:
  - Google results: \( y \)
  - User feedback (links clicked): \( \bar{y} \)
  - Relevance of each document: \( r(x, y[i]) \)
    - Manually ranked by assessors, reflects ground-truth utility of link
    - \( r(x, d) \in [0...5] \)
Who is Yisong Yue?

Yisong Yue | Machine Learning Researcher @ Caltech

Yisong Yue is an assistant professor in the Computing and Mathematical Sciences Department at the California Institute of Technology. His research interests lie in:

Research
- His research interests lie primarily in the theory and application of machine learning and data.

About
- Yisong Yue is an assistant professor in the Computing and...

Please read before emailing
- Please clearly state WHAT...

More results from yisongyue.com

Google (y)

Google (y)

User Clicks

Feedback Vector (y̅)

r' (x, y[i])

r' (x, y̅ [i])

A 4 5 3 2 3

B 5 4 3 3 2
Need to assess whether \( U(x, \bar{y}_t) > U(x, y_t) \)

Use standard metric of information retrieval quality (Manning et al., 2008)

\[
DCG@10(x, y) = \sum_{i=1}^{10} \frac{r(x, y[i])}{\log(i+1)}
\]

<table>
<thead>
<tr>
<th>Google ( y )</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>DCG = 10.38</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r(x, y[i]) )</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feedback Vector ( \bar{y} )</th>
<th>B</th>
<th>A</th>
<th>E</th>
<th>C</th>
<th>D</th>
<th>DCG = 10.82</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r(x, \bar{y}[i]) )</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Three Experimental Conditions

- Also check whether quality of feedback affected by quality of current prediction
  - How $y$ influences $\bar{y}$
- Normal
  - Top 10 google results in order
- Reversed
  - Top results in reverse order
- Swapped
  - Top 2 results switched

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google (Normal)</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$r(x, y[i])$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google (Reversed)</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>$r(x, y[i])$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google (Swapped)</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$r(x, y[i])$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results

- All CDF’s shifted to right
  - Implicit feedback can indeed produce improvements in utility
  - Statistically significant
Additionally, the previous graph shows that users provide accurate preferences across a range of retrieval qualities (normal, random, swapped).

Intuitively, a worse retrieval system may make it harder to find good results, but it also makes an easier baseline to improve upon.

This intuition is captured by $\alpha$-informative feedback.

Tradeoff between $\alpha$ and $\xi$ is application-specific.

- The following algorithms do not require knowledge of $\alpha$ or $\xi$. 
Roadmap

1. Introduction
2. Related Work
3. Coactive Learning Model
4. Coactive Learning Algorithms
5. Experiments
6. Robots

28
Modeling Utility

\[ U(x, y) = w^T_\star \phi(x, y) \]

- Linear function of \( \phi(x, y) \) parameterized by \( w^T_\star \)

- \( \phi(x, y) \): feature map dependent on \( x \) (context) and \( y \) (prediction)

- \( w^T_\star \): optimal weight vector
Example: Robot Manipulation

\[ U(x, y) = w^T \phi(x, y) \]

- **Context (x):** Environment (positions of other objects, etc)

- **Prediction (y):** Trajectory (list of waypoints)

- \( \phi(x, y) \)
  - Features such as distance from each waypoint to objects in environment
  - Can capture interaction between x and y
Preference Perceptron

Algorithm 1 Preference Perceptron.

| Initialize $w_1 \leftarrow 0$
| for $t = 1$ to $T$ do
| Observe $x_t$
| Present $y_t \leftarrow \arg\max_{y \in Y} w_t^T \phi(x_t, y)$
| Obtain feedback $\tilde{y}_t$
| Update: $w_{t+1} \leftarrow w_t + \phi(x_t, \tilde{y}_t) - \phi(x_t, y_t)$
| end for

- Goal: learn optimal weight vector $w^*_T$
- Update weight vector using difference in feature representations
Regret Bound of Preference Perceptron

\[
REG_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y_t^*) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}}
\]

- \( \alpha \) is parameter governing assumption that user feedback is \( \alpha \)-informative

- Recall: User feedback is \( \alpha \)-informative, with slack variables \( \xi_t \):

\[
U(x_t, y_t) - U(x_t, y_t^*) \geq \alpha(U(x_t, y_t^*) - U(x_t, y_t)) - \xi_t
\]

- \( R \) is upper bound on \( \|\phi\| \) \((\|\phi(x, y)\| \leq R)\)
Interpreting the Regret Bound

\[ REG_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y_t^*) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}} \]

Larger \( \alpha \) → more informative feedback → lower regret
Interpreting the Regret Bound

\[
REG_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y^*_t) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}}
\]

Larger $\alpha \rightarrow$ more informative feedback $\rightarrow$ lower regret

Larger $\xi_t \rightarrow$ worse violations of $\alpha$-informative assumption $\rightarrow$ higher regret
Interpreting the Regret Bound

\[ \text{REG}_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y^*_t) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}} \]

- Larger \( \alpha \) → more informative feedback → lower regret
- Larger \( \xi_t \) → worse violations of \( \alpha \)-informative assumption → higher regret
- Upper bound on \( U(x, y) \)
Interpreting the Regret Bound

\[ REG_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y^*_t) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}} \]

- If strict $\alpha$-informative assumption holds, slack terms vanish $\rightarrow \mathcal{O} \left( \frac{1}{\sqrt{T}} \right)$ regret

- Note: algorithm does not know $\alpha$; it just factors into the analysis
Outline of Proof

\[ REG_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y_t^*) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}} \]

1. Show \( \|w_{T+1}\|^2 \leq 4R^2T \)

2. Upper bound \( \sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t) \) in terms of \( \|w_{T+1}\| \)

3. Upper bound \( \sum_{t=1}^{T} U(x_t, y_t^*) - U(x_t, y_t) \) in terms of \( \sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t) \)

(using alpha-informative assumption)
1) Bounding $\|w_{T+1}\|^2$

- Expanding $\|w_{T+1}\|^2$ using our perceptron update rule $w_{T+1} = w_T + (\phi(x_T, y_T) - \phi(x_T, y_T))$ gives us:

$$w_{T+1}^T w_{T+1} = w_T^T w_T + 2w_T^T (\phi(x_T, y_T) - \phi(x_T, y_T)) + (\phi(x_T, y_T) - \phi(x_T, y_T))^T (\phi(x_T, y_T) - \phi(x_T, y_T))$$
1) Bounding $\|w_{T+1}\|^2$

$$w_{T+1}^T w_{T+1} = w_T^T w_T + 2w_T^T (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))$$

$$+ (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))^T (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))$$

$$\leq w_T^T w_T + 4R^2 \leq 4R^2 T.$$
1) Bounding $\|w_{T+1}\|^2$

$$
\begin{align*}
\mathbf{w}_{T+1}^T \mathbf{w}_{T+1} &= \mathbf{w}_T^T \mathbf{w}_T + 2\mathbf{w}_T^T (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T)) \\
&\quad + (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))^T (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T)) \\
&\leq \mathbf{w}_T^T \mathbf{w}_T + 4R^2 \leq 4R^2T.
\end{align*}
$$

Negative because our algorithm picked $\bar{y}_t$ over $\bar{y}_t$. 
1) Bounding $\|w_{T+1}\|^2$

\[
\begin{align*}
\mathbf{w}_{T+1}^\top \mathbf{w}_{T+1} &= \mathbf{w}_T^\top \mathbf{w}_T + 2\mathbf{w}_T^\top (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T)) \\
&\quad + (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))^\top (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T)) \\
&\leq \mathbf{w}_T^\top \mathbf{w}_T + 4R^2 \leq 4R^2T.
\end{align*}
\]

- Negative because our algorithm picked $\bar{y}_t$ over $\bar{y}_t$
- We have $\|\phi(x, y)\| \leq R$ so $\|\phi(x_T, y_T) - \phi(x_T, y_T)\| \leq 2R$ (triangle inequality)
1) Bounding $\|w_{T+1}\|^2$

$$w_{T+1}^T w_{T+1} = w_T^T w_T + 2w_T^T (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))$$
$$+ (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))^T (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))$$
$$\leq w_T^T w_T + 4R^2 \leq 4R^2T.$$ 

From $w_1 = 0$ and repeated application of the inequality $w_{T+1}^T w_{T+1} \leq w_T^T w_T + 4R^2$.

Negative because our algorithm picked $\bar{y}_t$ over $\bar{y}_t$.

We have $\|\phi(x, y)\| \leq R$ so $\|\phi(x_T, y_T) - \phi(x_T, y_T)\| \leq 2R$ (triangle inequality).
2) Bounding $\sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t)$

Using our previous result, we can find a bound on $\sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t)$

$$w_{T+1}^T w_* = w_{T}^T w_* + (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))^T w_*$$

$$= \sum_{t=1}^{T} (U(x_t, \bar{y}_t) - U(x_t, y_t)) .$$  \hspace{1cm} (9)
Using our previous result, we can find a bound on

\[
\sum_{t=1}^{T} U(x_t, y_t) - U(x_t, y_t)
\]

**Update rule multiplied by** \( w_* \)

\[
\begin{align*}
\mathbf{w}_{T+1}^T \mathbf{w}_* &= \mathbf{w}_T^T \mathbf{w}_* + (\phi(x_T, \bar{y}_T) - \phi(x_T, y_T))^T \mathbf{w}_* \\
&= \sum_{t=1}^{T} (U(x_t, \bar{y}_t) - U(x_t, y_t)). 
\end{align*}
\]

\( w_* \phi(x_T, y_T) \) is our true utility function, summation follows from iterating equation on first line
2) Bounding $\sum_{t=1}^{T} \left(U(x_t, \bar{y}_t) - U(x_t, y_t)\right) = w_{T+1}^T w_* \leq \|w_{T+1}\| \|w_*\|$

From prev. slide

$\|w_{T+1}\|^2 \leq 4R^2T$

$\sum_{t=1}^{T} \left(U(x_t, \bar{y}_t) - U(x_t, y_t)\right) \leq 2R\sqrt{T}\|w_*\|
2) Bounding $\sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t)$

$$\sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t) = w_{T+1}^T w_* \leq \|w_{T+1}\| \|w_*\|$$

Cauchy-Schwarz inequality

$$\|w_{T+1}\|^2 \leq 4R^2 T$$

From step 1

$$\sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t) \leq 2R\sqrt{T}\|w_*\|$$
3) Bounding \( \text{REG}_T \)

\[
\alpha(U(x_t, y_t^*) - U(x_t, y_t)) - \xi_t \leq U(x_t, \bar{y}_t) - U(x_t, y_t)
\]

\[
\sum_{t=1}^{T} U(x_t, \bar{y}_t) - U(x_t, y_t) \leq 2R\sqrt{T}\|w_*\|
\]

\[
\alpha \sum_{t=1}^{T} (U(x_t, y_t^*) - U(x_t, y_t)) - \sum_{t=1}^{T} \xi_t \leq 2R\sqrt{T}\|w_*\|
\]

\[
\text{REG}_T = \frac{1}{T} \sum_{t=1}^{T} U(x_t, y_t^*) - U(x_t, y_t) \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w^*\|}{\alpha \sqrt{T}}
\]
Comparison to Standard Perceptron

- **Standard perceptron (for multi-class classification)**
  - Requires true label $y_t^*$
  - Analyzed in terms of number of mistakes made

- **Preference perceptron**
  - Uses implicit feedback $\tilde{y}_t$
  - Analyzed in terms of utility
● Some applications have high volumes of feedback
  ○ Might not be possible to do an update after every round

● Consider a variant of Algorithm 1 that makes an update every k iterations
  ○ Uses $w_t$ obtained from the previous update until the next update

● It is easy to show the following regret bound for batch updates:

$$ REG_T \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R||w_*||\sqrt{k}}{\alpha \sqrt{T}}. $$
Corollary 3 Under expected $\alpha$-informative feedback model, the expected regret (over user behavior distribution) of the preference perceptron algorithm can be upper bounded as follows:

$$E[REG_T] \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|w_*\|}{\alpha \sqrt{T}}. \quad (10)$$

- Follow the argument of Theorem 1, but take expectations over user feedback
● We can generalize our results to minimize convex losses defined on the linear utility differences

● At every time step $t$, there is an (unknown) convex loss function $c_t : \mathcal{R} \rightarrow \mathcal{R}$
  ○ Determines the loss $c_t(U(x_t, y_t) - U(x_t, y_t^*))$
  ○ The functions $c_t$ are assumed to be non-increasing
  ○ Sub-derivatives of the $c_t$'s are assumed to be bounded
    ■ $c'_t(\theta) \in [-G, 0]$ for all $t$ and for all $\theta \in \mathcal{R}$

● The vector $w_*$ which determines the utility of $y_t$ under context $x_t$
  ○ Assumed from a closed and bounded convex set $B$ whose diameter is $|B|$
Convex Loss Minimization

Main differences from Algorithm 1

○ There is a rate $\eta_t$ associated with the update at time $t$
○ After every update, the resulting vector $\vec{w}_{t+1}$ is projected back to the set $B$
Theorem 4

For the convex preference perceptron, we have, for any \( \alpha \in (0, 1] \) and any \( \mathbf{w}_* \in \mathcal{B} \),

\[
\frac{1}{T} \sum_{t=1}^{T} c_t(U(x_t, y_t) - U(x_t, y^*_t)) \leq \frac{1}{T} \sum_{t=1}^{T} c_t(0)
\]

\[
+ \frac{2G}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{1}{\alpha} \left( \frac{2 |\mathcal{B}|G}{2\sqrt{T}} + \frac{|\mathcal{B}|G}{T} + \frac{4R^2G}{\sqrt{T}} \right).
\]

(11)

- Main differences from Theorem 1
  - \( c_t(0) \) is the minimum possible convex loss
  - Under strict \( \alpha \)-informative feedback, average loss approaches best achievable loss:
    - Larger constant factors than Theorem 1
Testing

- Empirically evaluate Preference Perceptron
  - Structured objects (rankings)

- Strong vs Weak Feedback
  - See how regret of algorithm changes with feedback quality
  - Different levels of α-informativity

- Noisy feedback
  - Directly uses user feedback
  - Compare to SVM
• Evaluated perceptron on Yahoo! Learning to rank dataset

• Query-url feature vectors
  ○ $x_i^q$ for query $q$ and URL $i$
  ○ Relevance rating $r_i^q \in [0, 4]$
  ○ $y_i$: index of URL at position $i$ in the ranking

• Joint feature map: $w^\top \phi(q, y) = \sum_{i=1}^{5} \frac{w^\top x_{y_i}^q}{\log(i + 1)}$
Procedure

- Query $q_t$ at time $t$

- Perceptron presents ranking $y_t^q$ that maximizes $w_t^T \phi(q_t, y)$
  - Equivalent to sorting URL’s by $w_t^T x_i^{q_t}$

- Utility regret:

$$\frac{1}{T} \sum_{t=1}^{T} w_*^T (\phi(q_t, y_{q_t}^{q_t*}) - \phi(q_t, y^{q_t}))$$
Strong vs. Weak Feedback

- Goal: see how regret of algorithm changes with feedback quality
  - Different values of $\alpha$

- Given predicted ranking $y_t$, user goes down list until they find URL’s such that resulting $\tilde{y}_t$ satisfies $\alpha$-informativity

- Update $w_{t+1}$ and repeat
- Regret with $\alpha = 1.0$ less than with $\alpha = 0.1$

- Difference less than factor of 10
Noisy Feedback

- Previous experiment: feedback based on actual utility values from optimal $w^*$
- Goal: use actual relevance labels from users
- Produces noisy feedback
  - No linear model can perfectly fit relevance labels
Noisy Feedback

- Regret with Preference Perceptron significantly less than with SVM
- Preference Perceptron runs orders of magnitude more quickly
- Regret converges to non-zero value
Comparison with Dueling Bandits

- Performs better than dueling bandits
Roadmap

1. Introduction
2. Related Work
3. Coactive Learning Model
4. Coactive Learning Algorithms
5. Experiments
6. Robots
• Teaching a robot to produce desired motions has been a long standing goal

• Past research has focused on mimicking expert’s demonstrations
  ○ Autonomous helicopter flights
  ○ Ball-In-A-Cup Experiment

• Applicable to scenarios when it is clear to an expert what constitutes a good trajectory
  ○ Extremely challenging in some scenarios, especially involving high DoF manipulators
  ○ Users have to give
    ■ End-effector’s location at each time-step
    ■ Full configuration of the arm in a way that is spatially and temporally consistent
http://www.youtube.com/watch?v=uLktpkd7ojA
● User never discloses optimal trajectory (or provides optimal feedback) to the robot

● Robot learns preferences from sub-optimal suggestions on how trajectory can be improved

● Authors design appropriate features that consider
  ○ Robot Configurations
  ○ Object-object relations
  ○ Temporal behavior

● Learns score functions reflecting user preferences from implicit feedback
Robot Learning Model

$s(y, \text{context} \mid \text{task})$

$\hat{s}(y, \text{context} \mid \text{task})$

Robot

User

Goal: Learn user preferences
Scoring Function

\[ s(x, y; w) = w_O^T \phi_O(x, y) + w_E^T \phi_E(x, y) \]

Object-object Interactions

Robot configuration and Environment Interactions

Connecting waypoints to neighboring objects
Object attributes: \{electronic, fragile, sharp, liquid, hot, \ldots\}

E.g. Laptop: \{electronic, fragile\}

Knife: \{sharp\} \ldots.

Hermans et. al. ICRA w/s 2011
Koppula et. al. NIPS 2011

Object-object Interactions

\[ w_o^T \phi_o(x, y) \]

\[ \sum_{\text{edges}} \sum_{l,k \in \text{labels}} 1(\text{edge}, l, k)w_{lk}^T\phi_{o-0}(x, y; \text{edge}) \]

Trajectory graph

Distance features
Features

\[ s(x, y; w) = w_O^T \phi_O(x, y) + w_E^T \phi_E(x, y) \]

Features ∈ \( \mathbb{R}^{75} \)

Object-object Interactions

Robot configuration and Environment Interactions

1. Spectrogram

2. Object’s distance from horizontal and vertical surfaces

3. Object’s angle with vertical axis

4. Robot’s wrist and elbow configuration in cylindrical co-ordinate
Computing Trajectory Rankings

- For a given task with context $x$, we would like to maximize the current scoring function

$$y^* = \arg\max_y s(x, y; w_O, w_E).$$

- Trajectory space is continuous and needs to be discretized to maintain argmax tractable
  - We can sample trajectories from the continuous space
    - Rapidly Exploring Random Tree (RRT)

- We can sort the trajectories by their trajectory scores to find the argmax
Trajectory Preference Perceptron (TPP)

**Algorithm 1** Trajectory Preference Perceptron. (TPP)

Initialize $w_O^{(1)} \leftarrow 0$, $w_E^{(1)} \leftarrow 0$

for $t = 1$ to $T$ do

Sample trajectories $\{y^{(1)}, ..., y^{(n)}\}$

$y_t = \text{argmax}_{y_s} (x_t, y; w_O^{(t)}, w_E^{(t)})$

Obtain user feedback $\bar{y}_t$

$w_O^{(t+1)} \leftarrow w_O^{(t)} + \phi_O(x_t, \bar{y}_t) - \phi_O(x_t, y_t)$

$w_E^{(t+1)} \leftarrow w_E^{(t)} + \phi_E(x_t, \bar{y}_t) - \phi_E(x_t, y_t)$

end for

- Almost the same as algorithm in previous paper except we sample trajectories
- Proof can be adapted to show that the expected average regret is upper-bounded by

$$E[\overline{REG_T}] \leq O\left(\frac{1}{\alpha \sqrt{T}} + \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t\right)$$
• Evaluate approach on 16 pick-and-place robotic tasks in a grocery store checkout setting

• For each task, train and test on scenarios with different objects being manipulated and/or with a different environment

• An expert labeled 1300 trajectories on a Likert scale of 1-5 (where 5 is the best) on the basis of subjective human preferences

• Evaluate quality of trajectories after robot has grasped the items and while it moves them for checkout
Baseline Algorithms

- **Geometric**: It plans a path, independent of the task, using a BiRRT planner
- **Manual**: It plans a path following certain manually coded preferences
- **Oracle-svm**: This algorithm leverages the expert’s labels on trajectories and is trained using SVM-rank in a batch manner
- **MMP-online**: This is an online implementation of Maximum margin planning (MMP)
- **TPP**: This is the algorithm from the paper

Where applicable, algorithms are applied to two different settings
- **Untrained** setting: algorithm learns preferences for the new task from scratch without observing any previous feedback
- **Pretrained** setting: algorithms are pre-trained on other similar tasks, and then adapt to the new task
Results

Table 1: Comparison of different algorithms and study of features in untrained setting. Table contains average nDCG@1(nDCG@3) values over 20 rounds of feedback.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Manipulation centric</th>
<th>Environment centric</th>
<th>Human centric</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric Manual</td>
<td>0.46 (0.48)</td>
<td>0.45 (0.39)</td>
<td>0.31 (0.30)</td>
<td>0.40</td>
</tr>
<tr>
<td>TPP Features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj-obj interaction</td>
<td>0.68 (0.68)</td>
<td>0.80 (0.79)</td>
<td>0.79 (0.73)</td>
<td>0.76</td>
</tr>
<tr>
<td>Robot arm config</td>
<td>0.82 (0.77)</td>
<td>0.78 (0.72)</td>
<td>0.80 (0.69)</td>
<td>0.80</td>
</tr>
<tr>
<td>Object trajectory</td>
<td>0.85 (0.81)</td>
<td>0.88 (0.84)</td>
<td>0.85 (0.72)</td>
<td>0.86</td>
</tr>
<tr>
<td>Object environment</td>
<td>0.70 (0.69)</td>
<td>0.75 (0.74)</td>
<td>0.81 (0.65)</td>
<td>0.75</td>
</tr>
<tr>
<td>TPP (all features)</td>
<td><strong>0.88 (0.84)</strong></td>
<td><strong>0.90 (0.85)</strong></td>
<td><strong>0.90 (0.80)</strong></td>
<td><strong>0.89 (0.83)</strong></td>
</tr>
<tr>
<td>MMP-online</td>
<td>0.47 (0.50)</td>
<td>0.54 (0.56)</td>
<td>0.33 (0.30)</td>
<td>0.45</td>
</tr>
</tbody>
</table>

- TPP performs better than baseline algorithms
- All features combined together give the best performance
Figure 5: Study of generalization with change in object, environment and both. **Manual**, Oracle-SVM, Pre-trained MMP-online (—), Untrained MMP-online (— —), Pre-trained TPP (—), Untrained TPP (— —).
Experiment: User Study

- Five users used system to train Baxter for grocery checkout tasks
- A set of 10 tasks of varying difficulty level was presented to users one at a time
- Users were instructed to provide feedback until they were satisfied with the top ranked trajectory
  - Zero-G was provided kinesthetically on the robot
  - Re-rank was elicited in a simulator
- To quantify the quality of learning each user evaluated on a Likert scale of 1-5 (5 is the best)
  - Their own trajectories (self score)
  - The trajectories learned of the other users (cross score)
  - Those predicted by Oracle-svm
- Time a user took for each task was also recorded
User Feedback

Re-rank

Zero-G
Results: User Study

- Within 5 feedbacks the users were able to improve over Oracle-svm
- Re-rank feedback was popular for easier tasks
- As difficulty increased the users relied more on zero-G feedback
- Each user took on average 5.5 minutes per-task
- Shows algorithm is realizable in practice on high DoF manipulators
Any Questions?


● Cornell University Robotics Learning Lab (http://pr.cs.cornell.edu/coactive/)
Robot Learning Model

- The robot is given a context $x$
  - Describes the environment, the objects, and any other input relevant to the problem

- The robot has to figure out what is a good trajectory $y$ for this context

- We assume that the user has a scoring function $s^*(x, y)$
  - Reflects how much he values each trajectory $y$ for context $x$
  - Higher score → better trajectory
  - Scoring function cannot be observed directly
  - User can provide us with preferences that reflect this scoring function

- The robot’s goal is to learn a function $s(x, y; w)$
  - Approximates the user’s true scoring function $s^*(x, y)$ as closely as possible
We compute features capturing robot’s arm configuration
  ○ Location of its elbow and wrist, w.r.t. to its shoulder, in cylindrical coordinate system, (r,θ,z)

Features to capture orientation and temporal behavior of the object to be manipulated
  ○ Cosine of the object’s maximum deviation, along the vertical axis, from its final orientation at the goal location
  ○ Spectrogram for each one-third part for the movement of the object in x, y, z directions
  ○ Compute the average power spectral density in the low and high frequency part

Features for object-environment
  ○ Captures temporal variation of vertical and horizontal distances from its surrounding surfaces
  ○ Minimum vertical distance from the nearest surface below it
  ○ Minimum horizontal distance from the surrounding surfaces
  ○ Minimum distance from the table, on which the task is being performed
  ○ Minimum distance from the goal location
  ○ Feature from time-frequency spectrogram of object’s vertical distance from the nearest surface below it
Features for Object-Object Interactions

- We enumerate waypoints of trajectory $y$ as $y_1, \ldots, y_N$
- Objects in the environment as $O = \{o_1, \ldots, o_K\}$
- The robot manipulates the object $\bar{o} \in O$
- A few of the trajectory waypoints would be affected by the other objects in the environment
  - We connect an object $o_k$ to a trajectory waypoint
    - If the minimum distance to collision is less than a threshold
    - If $o_k$ lies below $\bar{o}$
  - The edge connecting $y_j$ and $o_k$ is denoted as $(y_j, o_k) \in E$
Features for Object-Object Interactions

- Attributes of an object determine trajectory quality
  - For every object $o_k$, we consider a vector of $M$ binary variables $[l_k^1, \ldots, l_k^M]$
  - Each $l_k^m = \{0, 1\}$ indicating whether object $o_k$ possesses property $m$ or not

- If the set of possible properties are \{heavy, fragile, sharp, hot, liquid, electronic\}, then
  - Laptop can have labels $[0, 1, 0, 0, 0, 1]$
  - Glass Table can have labels $[0, 1, 0, 0, 0, 0]$

- For every $(y_j, o_k)$ edge, we extract following four features $\phi_{oo}(y_j, o_k)$:
  - Projection of minimum distance to collision along $x$, $y$ and $z$ (vertical) axis
  - Binary variable, that is 1, if $o_k$ lies vertically below $o$, 0 otherwise

- We now define the score $s_O(\cdot)$ over this graph as follows:

$$s_O(x, y; w_O) = \sum_{(y_j, o_k) \in \mathcal{E}} \sum_{p, q=1}^{M} l_k^p l_k^q [w_{pq} \cdot \phi_{oo}(y_j, o_k)]$$
We model the user’s scoring function with the following parameterized family of functions

\[ s(x, y; w) = w \cdot \phi(x, y) \]

We further decompose the score function in two parts

- \( s_O \) - Objects the trajectory is interacting with
- \( s_E \) - Object being manipulated and the environment

\[ s(x, y; w_O, w_E) = s_O(x, y; w_O) + s_E(x, y; w_E) = w_O \cdot \phi_O(x, y) + w_E \cdot \phi_E(x, y) \]
Atomic Predictions

- Movie recommendations from MovieLens dataset
  - 3090 movies rated by 6040 users
  - Over 1 million pairs

- Split data in half
  - First set:
    - Obtain feature vector $\mathbf{m}_j$ for each movie $j$ using SVD
  - Second set:
    - Consider problem of recommending movies based on features $\mathbf{m}_j$

- Simulates recommending movies to new user based on movie features from old users
Procedure

- For each user $i$
  - Best least-squares approximation $w^T_{i*} m_j$ to user utility

- At each $t$
  - Best available movie: $m_{t*}$
  - Recommended movie: $m_t$
  - User reveals preference
  - Recommended movie and feedback movie removed from subsequent set of candidate movies

- Utility regret:
  $$\frac{1}{T} \sum_{t=1}^{T} w^T_{i*} (m_{t*} - m_t)$$
Strong vs. Weak Feedback

- Goal: see how regret of algorithm changes with feedback quality
  - Different values of $\alpha$

- We recommend movie with maximum utility from current $w_t$

- User returns a movie with smallest utility that satisfies $\alpha$-informativity

- Update $w_{t+1}$ and repeat

- Regret calculated at each step and averaged over all users
• Regret with \( \alpha = 1.0 \) less than with \( \alpha = 0.5 \) and \( \alpha = 0.1 \)
• Difference less than factor of 2 and 10, respectively
• Regret converges faster for higher values of \( \alpha \)
Noisy Feedback

- Previous experiment: feedback based on actual utility values from optimal $w_*$
- Goal: use actual feedback from users
- User receives recommendation
- User returns a movie one rating higher
Noisy Feedback

- Regret with Preference Perceptron significantly less than with SVM
- Preference Perceptron runs orders of magnitude more quickly