Learning with Humans in the Loop

CS4780/5780 – Machine Learning
Fall 2014

Thorsten Joachims
Cornell University

Optional Reading:
User-Facing Machine Learning

• Examples
  – Search Engines
  – Netflix
  – Smart Home
  – Robot Assistant

• Learning
  – Gathering and maintenance of knowledge
  – Measure and optimize performance
  – Personalization
Interactive Learning System

\[ y_t \text{ dependent on } x_t \]
(e.g. ranking for query)

Utility: \( U(y_t) \)

\[ x_{t+1} \text{ dependent on } y_t \]
(e.g. click given ranking, new query)

- Observed Data ≠ Training Data
  - Observed data is user’s decisions
  - Even explicit feedback reflects user’s decision process
- Decisions \( \rightarrow \) Feedback \( \rightarrow \) Learning Algorithm
Decide between two Ranking Functions

Distribution \( P(u,q) \) of users \( u \), queries \( q \)

Retrieval Function 1
\[ f_1(u,q) \rightarrow r_1 \]

Retrieval Function 2
\[ f_2(u,q) \rightarrow r_2 \]

Which one is better?

U(tj,”SVM”,r_1)

U(tj,”SVM”,r_2)

1. Kernel Machines
   http://svm.first.gmd.de/
2. SVM-Light Support Vector Machine
   http://svmlight.joachims.org/
3. School of Veterinary Medicine at UPenn
   http://www.vet.upenn.edu/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Service Master Company
   http://www.servicemaster.com/

1. School of Veterinary Medicine at UPenn
   http://www.vet.upenn.edu/
2. Service Master Company
   http://www.servicemaster.com/
3. Support Vector Machine
   http://jbolivar.freeservers.com/
4. Archives of SUPPORT-VECTOR-MACHINES
   http://www.jiscmail.ac.uk/lists/SUPPORT...
5. SVM-Light Support Vector Machine
   http://ais.gmd.de/~thorsten/svm_light/
## Measuring Utility

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Aggregation</th>
<th>Hypothesized Change with Decreased Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abandonment Rate</td>
<td>% of queries with no click</td>
<td>N/A</td>
<td>Increase</td>
</tr>
<tr>
<td>Reformulation Rate</td>
<td>% of queries that are followed by reformulation</td>
<td>N/A</td>
<td>Increase</td>
</tr>
<tr>
<td>Queries per Session</td>
<td>Session = no interruption of more than 30 minutes</td>
<td>Mean</td>
<td>Increase</td>
</tr>
<tr>
<td>Clicks per Query</td>
<td>Number of clicks</td>
<td>Mean</td>
<td>Decrease</td>
</tr>
<tr>
<td>Click@1</td>
<td>% of queries with clicks at position 1</td>
<td>N/A</td>
<td>Decrease</td>
</tr>
<tr>
<td>Max Reciprocal Rank*</td>
<td>1/rank for highest click</td>
<td>Mean</td>
<td>Decrease</td>
</tr>
<tr>
<td>Mean Reciprocal Rank*</td>
<td>Mean of 1/rank for all clicks</td>
<td>Mean</td>
<td>Decrease</td>
</tr>
<tr>
<td>Time to First Click*</td>
<td>Seconds before first click</td>
<td>Median</td>
<td>Increase</td>
</tr>
<tr>
<td>Time to Last Click*</td>
<td>Seconds before final click</td>
<td>Median</td>
<td>Decrease</td>
</tr>
</tbody>
</table>

(*): only queries with at least one click count
User Study in ArXiv.org

– Natural user and query population
– User in natural context, not lab
– Live and operational search engine
– Ground truth by construction

**ORIG** > **SWAP2** > **SWAP4**

- **ORIG**: Hand-tuned fielded
- **SWAP2**: ORIG with 2 pairs swapped
- **SWAP4**: ORIG with 4 pairs swapped

**ORIG** > **FLAT** > **RAND**

- **ORIG**: Hand-tuned fielded
- **FLAT**: No field weights
- **RAND**: Top 10 of FLAT shuffled

[Radlinski et al., 2008]
ArXiv.org: Experiment Setup

• Experiment Setup
  – Phase I: 36 days
    • Users randomly receive ranking from Orig, Flat, Rand
  – Phase II: 30 days
    • Users randomly receive ranking from Orig, Swap2, Swap4
  – User are permanently assigned to one experimental condition based on IP address and browser.

• Basic Statistics
  – ~700 queries per day / ~300 distinct users per day

• Quality Control and Data Cleaning
  – Test run for 32 days
  – Heuristics to identify bots and spammers
  – All evaluation code was written twice and cross-validated
Conclusions

• None of the absolute metrics reflects expected order.

• Most differences not significant after one month of data.

• Absolute metrics not suitable for ArXiv-sized search engines.

[Radlinski et al., 2008]
Yahoo! Search: Results

- **Retrieval Functions**
  - 4 variants of production retrieval function

- **Data**
  - 10M – 70M queries for each retrieval function
  - Expert relevance judgments

- **Results**
  - Still not always significant even after more than 10M queries per function
  - Only Click@1 consistent with DCG@5.

[Chapelle et al., 2012]
Interactive Learning System

- $y_t$ dependent on $x_t$ (e.g. ranking for query)
- $x_{t+1}$ dependent on $y_t$ (e.g. click given ranking, new query)
- Utility: $U(y_t)$

- Observed Data ≠ Training Data
- Decisions $\rightarrow$ Feedback $\rightarrow$ Learning Algorithm
  - Model the users decision process to extract feedback
  - Design learning algorithm for this type of feedback
Decide between two Ranking Functions

Distribution $P(u,q)$ of users $u$, queries $q$

Retrieval Function 1
$f_1(u,q) \rightarrow r_1$

Retrieval Function 2
$f_2(u,q) \rightarrow r_2$

Which one is better?

1. Kernel Machines
   [http://svm.first.gmd.de/](http://svm.first.gmd.de/)
2. SVM-Light Support Vector Machine
3. School of Veterinary Medicine at UPenn
   [http://www.vet.upenn.edu/](http://www.vet.upenn.edu/)
4. An Introduction to Support Vector Machines
5. Service Master Company

U(tj,"SVM",r_1)

1. School of Veterinary Medicine at UPenn
   [http://www.vet.upenn.edu/](http://www.vet.upenn.edu/)
2. Service Master Company
3. Support Vector Machine
4. Archives of SUPPORT-VECTOR-MACHINES
   [http://www.jiscmail.ac.uk/lists/SUPPORT...](http://www.jiscmail.ac.uk/lists/SUPPORT...)
5. SVM-Light Support Vector Machine

U(tj,"SVM",r_2)
A Model of how Users Click in Search

• Model of clicking:
  – Users explore ranking to position $k$
  – Users click on most relevant (looking) links in top $k$
  – Users stop clicking when time budget up or other action more promising (e.g. reformulation)
  – Empirically supported by [Granka et al., 2004]
Interleaving\((r_1, r_2)\)

Model of User:
Better retrieval functions is more likely to get more clicks.

Invariant:
For all k, top k of balanced interleaving is union of top \(k_1\) of \(r_1\) and top \(k_2\) of \(r_2\) with \(k_1= k_2 \pm 1\).

Interpretation: \((r_1 \gg r_2) \iff \text{clicks(topk}(r_1)) > \text{clicks(topk}(r_2))\)

\(\rightarrow\) see also [Radlinski, Craswell, 2012] [Hofmann, 2012]

[Joachims, 2001] [Radlinski et al., 2008]
Arxiv.org: Interleaving Experiment

- **Experiment Setup**
  - **Phase I:** 36 days
    - Balanced Interleaving of (Orig,Flat) (Flat,Rand) (Orig,Rand)
  - **Phase II:** 30 days
    - Balanced Interleaving of (Orig,Swap2) (Swap2,Swap4) (Orig,Swap4)

- **Quality Control and Data Cleaning**
  - Same as for absolute metrics
Arxiv.org: Interleaving Results

Conclusions

• All interleaving experiments reflect the expected order.
• All differences are significant after one month of data.
• Same results also for alternative data-preprocessing.
Yahoo and Bing: Interleaving Results

- Yahoo Web Search [Chapelle et al., 2012]
  - Four retrieval functions (i.e. 6 paired comparisons)
  - Balanced Interleaving
    - All paired comparisons consistent with ordering by NDCG.

- Bing Web Search [Radlinski & Craswell, 2010]
  - Five retrieval function pairs
  - Team-Game Interleaving
    - Consistent with ordering by NDGC when NDCG significant.
Interactive Learning System

$y_t$ dependent on $x_t$
(e.g. ranking for query)

Utility: $U(y_t)$

$x_{t+1}$ dependent on $y_t$
(e.g. click given ranking, new query)

- Observed Data ≠ Training Data
- Decisions → Feedback → Learning Algorithm
  - Model the users decision process to extract feedback
    → Pairwise comparison test $P( y_i \succ y_j \mid U(y_i) > U(y_j) )$
  - Design learning algorithm for this type of feedback
Learning on Operational System

• Example: 4 retrieval functions: A > B >> C > D
  – 10 possible pairs for interactive experiment
    • (A,B) → low cost to user
    • (A,C) → medium cost to user
    • (C,D) → high cost to user
    • (A,A) → zero cost to user
    • ...

• Minimizing Regret
  – Don’t present “bad” pairs more often than necessary
  – Trade off (long term) informativeness and (short term) cost
  – Definition: Probability of \((f_t, f_t')\) losing against the best \(f^*\)

\[
R(A) = \sum_{t=1}^{T} \left[ P(f^* \succ f_t) - 0.5 \right] + \left[ P(f^* \succ f_t') - 0.5 \right]
\]

⇒ Dueling Bandits Problem

[Yue, Broder, Kleinberg, Joachims, 2010]
First Thought: Tournament

- Noisy Sorting/Max Algorithms:
  - [Feige et al.]: Triangle Tournament Heap $O(n/\varepsilon^2 \log(1/\delta))$ with prob $1-\delta$
  - [Adler et al., Karp & Kleinberg]: optimal under weaker assumptions
### Algorithm: Interleaved Filter 2

#### Algorithm

InterleavedFilter1(T,W={f_1...f_K})

- **Pick random f’ from W**
- **δ=1/(TK^2)**
- **WHILE |W|>1**
  - FOR b ∈ W DO
    » duel(f’,f)
    » update P_f
  - t=t+1
  - c_t=(log(1/δ)/t)^0.5
  - Remove all f from W with P_f < 0.5-c_t
  - IF there exists f” with P_{f’} > 0.5+c_t
    » Remove f’ from W
    » Remove all f from W that are empirically inferior to f’
    » f’=f’’; t=0
- **UNTIL T: duel(f’,f’)**

---

#### Related Algorithms:

- Hofmann, Whiteson, Rijke, 2011
- Yue, Joachims, 2009
- Yue et al., 2009

---

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>f_2</td>
<td>f’=f_3</td>
<td>f_4</td>
<td>f_5</td>
</tr>
<tr>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_1</td>
<td>f_2</td>
<td>f’=f_3</td>
<td>f_4</td>
<td>f_5</td>
</tr>
<tr>
<td>8/2</td>
<td>7/3</td>
<td>4/6</td>
<td>1/9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_1</td>
<td>f_2</td>
<td>f’=f_3</td>
<td>f_4</td>
<td></td>
</tr>
<tr>
<td>13/2</td>
<td>11/4</td>
<td>7/8</td>
<td>XX</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f’=f_1</td>
<td>f_2</td>
<td></td>
<td>f_4</td>
<td></td>
</tr>
<tr>
<td>0/0</td>
<td>0/0</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
</tbody>
</table>
Assumptions

• Preference Relation: \( f_i \succ f_j \iff P(f_i \succ f_j) = 0.5 + \varepsilon_{i,j} > 0.5 \)

• Weak Stochastic Transitivity: \( f_i \succ f_j \) and \( f_j \succ f_k \rightarrow f_i \succ f_k \)

\[ \varepsilon_{1,2} = 0.01 \text{ and } \varepsilon_{2,3} = 0.01 \rightarrow \varepsilon_{1,3} \leq 0.02 \]

• \( \varepsilon \)-Winner exists: \( \varepsilon = \max_i \{ P(f_1 \succ f_i) - 0.5 \} = \varepsilon_{1,2} > 0 \)

\[ \frac{1}{T} E(R_T) \leq O \left( \frac{K \log T}{\varepsilon_{1,2} T} \right) \]
Theorem: Any algorithm for the dueling bandits problem has regret

$$R_T \leq \Omega \left( \frac{K}{\epsilon^{1.2} \log T} \right)$$

Proof: [Karp, Kleinberg, 2007] [Kleinberg et al., 2007]

Intuition:
- Magically guess the best bandit, just verify guess
- Worst case: $\forall f_i \succ f_j: P(f_i \succ f_j) = 0.5 + \epsilon$
- Need $O(1/\epsilon^2 \log T)$ duels to get $1-1/T$ confidence.
Interactive Learning System

- $y_t$ dependent on $x_t$
  - (e.g. ranking for query)

- $x_{t+1}$ dependent on $y_t$
  - (e.g. click given ranking, new query)

Utility: $U(y_t)$

- Observed Data ≠ Training Data
- Decisions $\rightarrow$ Feedback $\rightarrow$ Learning Algorithm
  - Model the users decision process to extract feedback
    - Pairwise comparison test $P(y_i \succ y_j \mid U(y_i) > U(y_j))$
  - Design learning algorithm for this type of feedback
    - Dueling Bandits problem and algorithms (e.g. IF2)
Who does the exploring?

Example 1
Who does the exploring?  
Example 2
Who does the exploring?

Example 3
Coactive Feedback Model

• Interaction: given x

Algorithm prediction

User explored

\( y \)

\( \bar{y} \)

Improved Prediction

Set of all y for context x

• Feedback:
  – Improved prediction \( \bar{y}_t \)
    \[
    U(\bar{y}_t | x_t) > U(y_t | x_t)
    \]
  – Supervised learning: optimal prediction \( y_t^* \)
    \[
    y_t^* = \arg\max_y U(y | x_t)
    \]
We propose Coactive Learning as a model of interaction between a learning system and a human user, where both have the common goal of providing results of maximum utility to the user.

Wir schlagen vor, koaktive Learning als ein Modell der Wechselwirkung zwischen einem Lernsystem und menschlichen Benutzer, wobei sowohl die gemeinsame Ziel, die Ergebnisse der maximalen Nutzen für den Benutzer.

Wir schlagen vor, koaktive Learning als ein Modell der Wechselwirkung des Dialogs zwischen einem Lernsystem und menschlichen Benutzer, wobei sowohl die beide das gemeinsame Ziel haben, die Ergebnisse der maximalen Nutzen für den Benutzer zu liefern.
Coactive Learning Model

- Unknown Utility Function: $U(y|x)$
  - Boundedly rational user

- Algorithm/User Interaction:
  - LOOP FOREVER
    - Observe context $x$ (e.g. query)
    - Learning algorithm presents $y$ (e.g. ranking)
    - User returns $\bar{y}$ with $U(\bar{y}|x) > U(y|x)$
    - Regret = Regret + $[ U(y^*|x) - U(y|x) ]$

- Relationship to other online learning models
  - Expert setting: receive $U(y|x)$ for all $y$
  - Bandit setting: receive $U(y|x)$ only for selected $y$
  - Dueling bandits: for selected $y$ and $\bar{y}$, receive $U(\bar{y}|x) > U(y|x)$
  - Coactive setting: for selected $y$, receive $\bar{y}$ with $U(\bar{y}|x) > U(y|x)$
Coactive Preference Perceptron

• Model
  – Linear model of user utility: \( U(y|x) = w^T \phi(x,y) \)

• Algorithm
  • FOR \( t = 1 \) TO \( T \) DO
    – Observe \( x_t \)
    – Present \( y_t = \arg\max_y \{ w_t^T \phi(x_t,y) \} \)
    – Obtain feedback \( \tilde{y}_t \) from user
    – Update \( w_{t+1} = w_t + \phi(x_t,\tilde{y}_t) - \phi(x_t,y_t) \)

• This may look similar to a multi-class Perceptron, but
  – Feedback \( \tilde{y}_t \) is different (not get the correct class label)
  – Regret is different (misclassifications vs. utility difference)

\[
R(A) = \frac{1}{T} \sum_{t=1}^{T} [U(y_t^*|x) - U(y_t|x)]
\]

[Shivaswamy, Joachims, 2012]
**α-Informative Feedback**

- **Definition: Strict α-Informative Feedback**
  \[ U(x_t, \bar{y}_t) \geq U(x_t, y_t) + \alpha(U(x_t, y^*_t) - U(x_t, y_t)) \]

- **Definition: α-Informative Feedback**
  \[ U(x_t, \bar{y}_t) = U(x_t, y_t) + \alpha(U(x_t, y^*_t) - U(x_t, y_t)) - \xi_t \]

[Shivaswamy, Joachims, 2012]
Preference Perceptron: Regret Bound

• Assumption
  – \( U(y|x) = w^T \phi(x,y) \), but \( w \) is unknown

• Theorem
  For user feedback \( \tilde{y} \) that is \( \alpha \)-informative, the average regret of the Preference Perceptron is bounded by

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

• Other Algorithms and Results
  – Feedback that is \( \alpha \)-informative only in expectation
  – General convex loss functions of \( U(y^*_|x)-U(\hat{y} | x) \)
  – Regret that scales \( \log(T)/T \) instead of \( T^{-0.5} \) for strongly convex

[Shivaswamy, Joachims, 2012]
Preference Perceptron: Experiment

Experiment:

- Automatically optimize Arxiv.org Fulltext Search

Model

- Utility of ranking $y$ for query $x$: $U_t(y|x) = \sum_t \gamma_i w^T_t \phi(x,y^{(i)})$ [~1000 features]
  - Computing argmax ranking: sort by $w^T_t \phi(x,y^{(i)})$

Feedback

- Construct $\tilde{y}_t$ from $y_t$ by moving clicked links one position higher.
- Perturbation [Raman et al., 2013]

Baseline

- Handtuned $w_{base}$ for $U_{base}(y|x)$

Evaluation

- Interleaving of ranking from $U_t(y|x)$ and $U_{base}(y|x)$

[Analogous to DCG]

[Graph: Cumulative Win Ratio vs. Number of Feedback]

[Coactive Learning]

[Baseline]
Summary and Conclusions

Algorithm

\[ y_t \text{ dependent on } x_t \]
(e.g. ranking for query)

Utility: \( U(y_t) \)

User

\[ x_{t+1} \text{ dependent on } y_t \]
(e.g. click given ranking, new query)

- Observed Data \( \neq \) Training Data
- Decisions \( \rightarrow \) Feedback \( \rightarrow \) Learning Algorithm
  - Dueling Bandits
    - Model: Pairwise comparison test \( P(y_i \succ y_j \mid U(y_i) > U(y_j)) \)
    - Algorithm: Interleaved Filter 2, \( O(|Y| \log(T)) \) regret
  - Coactive Learning
    - Model: for given \( y \), user provides \( \tilde{y} \) with \( U(\tilde{y} \mid x) > U(y \mid x) \)
    - Algorithm: Preference Perceptron, \( O(\|w\| T^{0.5}) \) regret