

Stable Coactive Learning via Perturbation

Karthik Raman ¹ Thorsten Joachims ¹ Pannaga Shivaswamy ²
Tobias Schnabel ³

¹Cornell University {*karthik,tj*}@*cs.cornell.edu*

²AT&T Research *pannaga@research.att.com*

³Stuttgart University *tbs49@cornell.edu*

June 19, 2013

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .

→ e.g. : Search Engine

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .

e.g. : Search Engine

User Query

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .

→ e.g. : Search Engine

→ User Query

→ Ranking

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .
- $\text{Regret} = \text{Regret} + U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$

e.g. : Search Engine

User Query

Ranking

User utility

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .
- Regret = Regret + $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$
- System gets feedback:
Full information: $U(\mathbf{x}_t, \mathbf{y}^{(1)})$, $U(\mathbf{x}_t, \mathbf{y}^{(2)})$, ...

e.g. : Search Engine

User Query

Ranking

User utility

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .
- Regret = Regret + $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$
- System gets feedback:
Full information: $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \dots$

e.g. : Search Engine

User Query

Ranking

User utility

Unrealistic for users to provide (e.g., implicit feedback).

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .
- $\text{Regret} = \text{Regret} + U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$
- System gets feedback:
~~Full information: $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \dots$~~
Bandit: $U(\mathbf{x}_t, \mathbf{y}_t)$

e.g. : Search Engine

User Query

Ranking

User utility

Unrealistic for users to provide (e.g., implicit feedback).

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .
- Regret = Regret + $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$
- System gets feedback:
~~Full information: $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \dots$~~
~~Bandit: $U(\mathbf{x}_t, \mathbf{y}_t)$~~
Coactive: $U(\mathbf{x}_t, \bar{\mathbf{y}}_t) \geq_{\alpha} U(\mathbf{x}_t, \mathbf{y}_t)$

e.g. : Search Engine

User Query

Ranking

User utility

Coactive Learning

Learning model

Repeat forever:

- System receives context \mathbf{x}_t .
- System makes prediction \mathbf{y}_t .
- Regret = Regret + $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$
- System gets feedback:

~~Full information:~~ $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \dots$

~~Bandit:~~ $U(\mathbf{x}_t, \mathbf{y}_t)$

Coactive: $U(\mathbf{x}_t, \bar{\mathbf{y}}_t) \geq_{\alpha} U(\mathbf{x}_t, \mathbf{y}_t)$

e.g. : Search Engine

User Query

Ranking

User utility

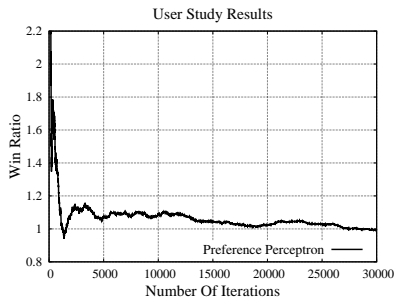
Perceptron has regret $O\left(\frac{1}{\alpha\sqrt{T}}\right)$ for linear utility ($U(\mathbf{x}, \mathbf{y}) = \mathbf{w}_*^T \phi(\mathbf{x}, \mathbf{y})$).

User Study: Learning Rankings using Perceptron

- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.

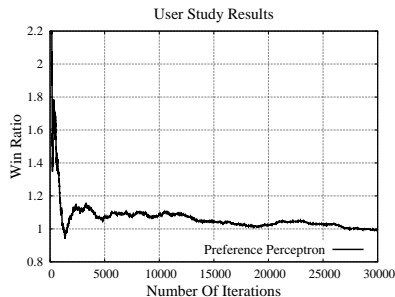
User Study: Learning Rankings using Perceptron

- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).



User Study: Learning Rankings using Perceptron

- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).



Perceptron performs poorly!

User Study: Learning Rankings using Perceptron

Preference Perceptron Algo:

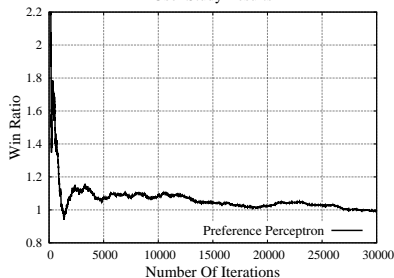
- 1 Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
- 2 Given context \mathbf{x}_t present $\mathbf{y}_t \leftarrow \operatorname{argmax}_y \mathbf{w}_t^T \phi(\mathbf{x}_t, \mathbf{y})$.

- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).

Presented Ranking (y)



User Study Results



Perceptron performs poorly!

User Study: Learning Rankings using Perceptron

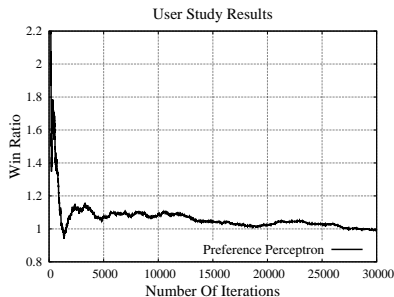
Preference Perceptron Algo:

- 1 Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
- 2 Given context \mathbf{x}_t present $\mathbf{y}_t \leftarrow \operatorname{argmax}_{\mathbf{y}} \mathbf{w}_t^T \phi(\mathbf{x}_t, \mathbf{y})$.
- 3 Observe clicks and construct feedback ranking $\bar{\mathbf{y}}_t$.

Presented Ranking (y)



- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).



Perceptron performs poorly!

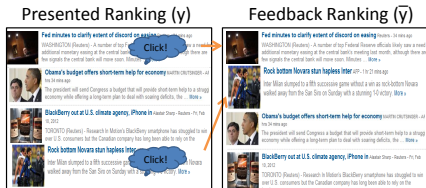
User Study: Learning Rankings using Perceptron

Preference Perceptron Algo:

- 1 Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
- 2 Given context \mathbf{x}_t present $\mathbf{y}_t \leftarrow \operatorname{argmax}_y \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$.
- 3 Observe clicks and construct feedback ranking $\bar{\mathbf{y}}_t$.

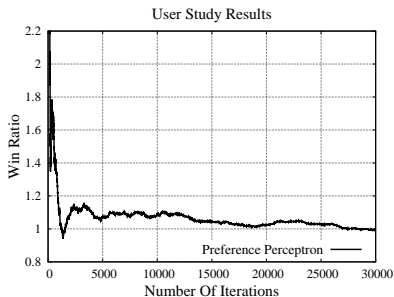
Presented Ranking (\mathbf{y})

Feedback Ranking ($\bar{\mathbf{y}}$)



The diagram illustrates the process of learning from user clicks. On the left, a 'Presented Ranking' shows a list of news items. Two items are highlighted with blue circles and labeled 'Click!'. Arrows point from these clicks to the corresponding items in the 'Feedback Ranking' on the right, which are also highlighted with blue circles. The feedback ranking shows the items in a different order, reflecting the user's preference.

- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).



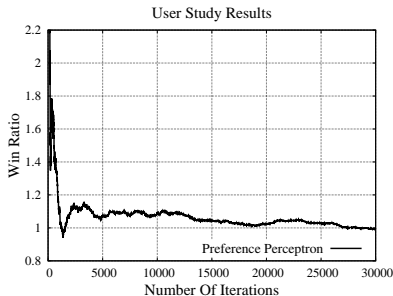
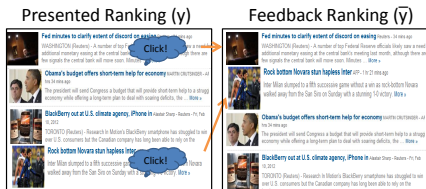
Perceptron performs poorly!

User Study: Learning Rankings using Perceptron

Preference Perceptron Algo:

- 1 Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
- 2 Given context \mathbf{x}_t present $\mathbf{y}_t \leftarrow \operatorname{argmax}_y \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$.
- 3 Observe clicks and construct feedback ranking $\bar{\mathbf{y}}_t$.
- 4 $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$.
- 5 Repeat from step 2.

- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).

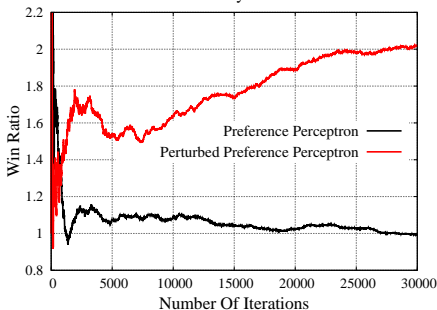


Perceptron performs poorly!

Perturbed Preference Perceptron

- 1 Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
- 2 Given context \mathbf{x}_t compute $\hat{\mathbf{y}}_t \leftarrow \operatorname{argmax}_{\mathbf{y}} \mathbf{w}_t^T \phi(\mathbf{x}_t, \mathbf{y})$.
- 3 Present $\mathbf{y}_t \leftarrow \text{Perturb}(\hat{\mathbf{y}}_t)$
(Randomly swap adjacent pairs).
- 4 Observe clicks and construct feedback ranking $\bar{\mathbf{y}}_t$.
- 5 $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$.
- 6 Repeat from step 2.

User Study Results



Predicted Ranking ($\hat{\mathbf{y}}$)

A list of news items in a predicted ranking order. The items are: 'Fed minutes to clarify extent of discord on easing', 'Obama's budget offers short-term help for economy', 'BlackBerry out at U.S. climate agency, iPhone', and 'Rock bottom Novara stun hapless Inter'. Each item includes a small image and a brief text snippet.

PERTURB

Presented Ranking (\mathbf{y})

A list of news items in a presented ranking order, which is a perturbed version of the predicted ranking. The items are: 'Obama's budget offers short-term help for economy', 'Fed minutes to clarify extent of discord on easing', 'BlackBerry out at U.S. climate agency, iPhone', and 'Rock bottom Novara stun hapless Inter'. A blue arrow indicates a swap between the first and second items.

I will tell you:

- Why the preference perceptron performs poorly?
- Why does perturbation fix the problem?
- What are the regret bounds for the algorithm?
- How do we do this more generally for non-ranking problems?