Stable Coactive Learning via Perturbation

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

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

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

[3]Stuttgart University *tbs49@cornell.edu*

June 19, 2013

Learning model

Repeat forever:

- System receives context $\mathbf{x}_t$.

Coactive Learning

e.g. : Search Engine

Learning model

Repeat forever:

- System receives context $\mathbf{x}_t$.

Coactive Learning

e.g. : Search Engine

Learning model

Repeat forever:

- System receives context $\mathbf{x}_t$.

User Query

Coactive Learning

e.g. : Search Engine

Learning model

Repeat forever:

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

User Query

Ranking

Coactive Learning

e.g. : Search Engine

Learning model

Repeat forever:

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

User Query

Ranking

User utility

Coactive Learning

e.g. : Search Engine

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: $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \ldots$

User Query

Ranking

User utility

Coactive Learning

e.g. : Search Engine

Learning model

Repeat forever:

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

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

Coactive Learning

e.g. : Search Engine

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**: $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \ldots$~~
  Bandit: $U(\mathbf{x}_t, \mathbf{y}_t)$

User Query

Ranking

User utility

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

Coactive Learning

e.g. : Search Engine

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**: $U(\mathbf{x}_t, \mathbf{y}^{(1)}), U(\mathbf{x}_t, \mathbf{y}^{(2)}), \ldots$~~
  ~~**Bandit**: $U(\mathbf{x}_t, \mathbf{y}_t)$~~
  Coactive: $U(\mathbf{x}_t, \bar{\mathbf{y}}_t) \geq_\alpha U(\mathbf{x}_t, \mathbf{y}_t)$

User Query

Ranking

User utility

Coactive Learning

e.g. : Search Engine

Learning model

Repeat forever:

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

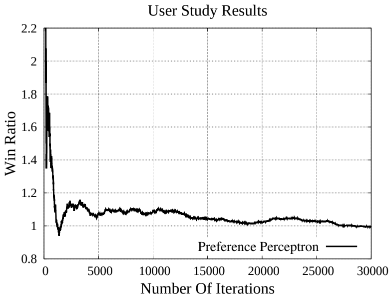
Perceptron has regret $O(\frac{1}{\alpha\sqrt{T}})$ for linear utility ($U(\mathbf{x}, \mathbf{y}) = \mathbf{w}_*^\top \phi(\mathbf{x}, \mathbf{y})$).

User Study: Learning Rankings using Perceptron

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

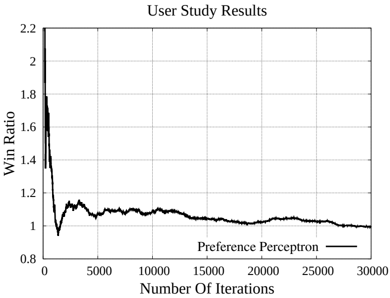
User Study: Learning Rankings using Perceptron

- 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).



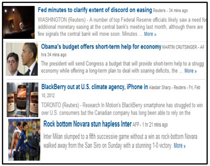
Perceptron performs poorly!

User Study: Learning Rankings using Perceptron

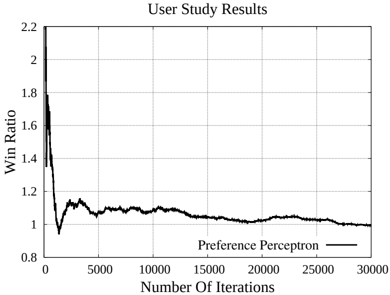
Preference Perceptron Algo:

1. Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
2. Given context $\mathbf{x}_t$ present $\mathbf{y}_t \leftarrow \mathrm{argmax}_{\mathbf{y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$.

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!

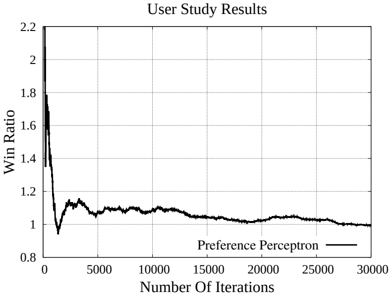
User Study: Learning Rankings using Perceptron

Preference Perceptron Algo:

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

- 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)



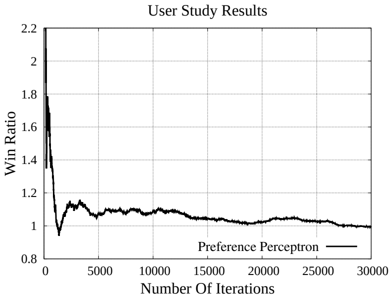
Perceptron performs poorly!

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 $\mathbf{y}_t \leftarrow \text{argmax}_{\mathbf{y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$.
3. Observe clicks and construct feedback ranking $\bar{\mathbf{y}}_t$.

- 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!

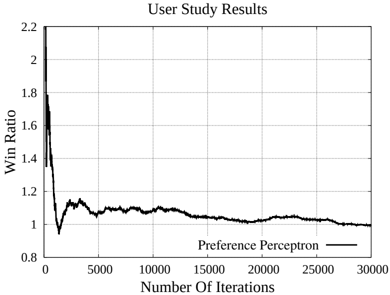
User Study: Learning Rankings using Perceptron

Preference Perceptron Algo:

1. Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
2. Given context $\mathbf{x}_t$ present $\mathbf{y}_t \leftarrow \mathrm{argmax}_{\mathbf{y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$.
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.

Presented Ranking (y) Feedback Ranking ($\bar{\text{y}}$)

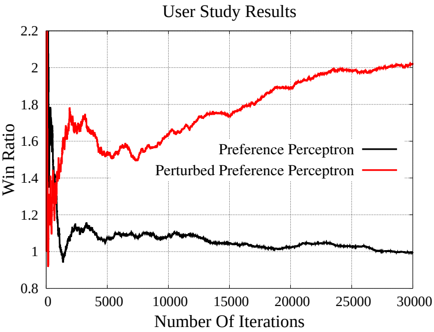
- On live search engine.
- Goal: Learn ranking function from user clicks.
- Interleaved comparison against hand-tuned baseline.
- Win ratio of 1 means no better than baseline (Higher = Better).



Perceptron performs poorly!

Perturbed Preference Perceptron

1. Initialize weight vector $\mathbf{w}_1 \leftarrow \mathbf{0}$.
2. Given context $\mathbf{x}_t$ compute $\hat{\mathbf{y}}_t \leftarrow \mathrm{argmax}_{\mathbf{y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$.
3. Present $\mathbf{y}_t \leftarrow \mathit{Perturb}(\hat{\mathbf{y}}_t)$ (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.



Predicted Ranking ($\hat{y}$)

Presented Ranking (y)

Please come to our poster

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?