

Vowpal Wabbit 2017 Update



John Langford

<http://hunch.net/~vw/>

git clone

[git://github.com/JohnLangford/vowpal_wabbit.git](https://github.com/JohnLangford/vowpal_wabbit.git)

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)
5. Sparse Models

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)
5. Sparse Models
6. Baseline (Alberto)

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)
5. Sparse Models
6. Baseline (Alberto)
7. Optimized Exploration algorithms (Alberto)

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)
5. Sparse Models
6. Baseline (Alberto)
7. Optimized Exploration algorithms (Alberto)
8. Cost Sensitive Active Learning (Akshay)

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)
5. Sparse Models
6. Baseline (Alberto)
7. Optimized Exploration algorithms (Alberto)
8. Cost Sensitive Active Learning (Akshay)
9. Active Learning to Search (Hal)

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
2. Online Learning (*)
3. Active Learning (*)
4. Learning Reduction (*)
5. Sparse Models
6. Baseline (Alberto)
7. Optimized Exploration algorithms (Alberto)
8. Cost Sensitive Active Learning (Akshay)
9. Active Learning to Search (Hal)
10. Java Interface (Jon Morra)

(*) Old stuff

What is Vowpal Wabbit

1. Large Scale linear regression (*)
 2. Online Learning (*)
 3. Active Learning (*)
 4. Learning Reduction (*)
 5. Sparse Models
 6. Baseline (Alberto)
 7. Optimized Exploration algorithms (Alberto)
 8. Cost Sensitive Active Learning (Akshay)
 9. Active Learning to Search (Hal)
 10. Java Interface (Jon Morra)
 11. JSON/Decision Service (Markus)
- (*) Old stuff

Community

1. BSD license.

Community

1. BSD license.
2. Mailing list >500, Github >1K forks, >1K, >1K issues, >100 contributors

Community

1. BSD license.
2. Mailing list >500, Github >1K forks, >1K, >1K issues, >100 contributors
- 3.

amazon



AOL

Baidu 百度

GraphLab



FTI
CONSULTING

IBM

Microsoft



YAHOO! Яндекc

Sparse Models

Scenario: You want to train a model with many potential parameters but use little RAM at test time.

Sparse Models

Scenario: You want to train a model with many potential parameters but use little RAM at test time.

Step 1: `vw -b 26 --l1 1e-7 <training set>`
(memory footprint is 1GB)

Sparse Models

Scenario: You want to train a model with many potential parameters but use little RAM at test time.

Step 1: `vw -b 26 --l1 1e-7 <training set>`

(memory footprint is 1GB)

Step 2: `vw -t --sparse_weights <test set>`

(memory footprint is 100MB)

Baseline and Contextual Bandits

Alberto Bietti (Inria)
alberto@bietti.me

Setting: **online regression**

- ▶ **Problem:** range of targets (e.g. offset) is unknown
- ▶ Bias term (weight for “constant” features) can be slow to learn
- ▶ Hurts performance of learning / exploration algorithms

Setting: **online regression**

- ▶ **Problem:** range of targets (e.g. offset) is unknown
- ▶ Bias term (weight for “constant” features) can be slow to learn
- ▶ Hurts performance of learning / exploration algorithms

Goal: adapt quickly and automatically to the range of targets

Baseline

Solution:

- ▶ Learn baseline regressor separately from rest
 - ▶ From constant features on example `--baseline`
 - ▶ Or separate global constant example `--baseline --global_only`

Baseline

Solution:

- ▶ Learn baseline regressor separately from rest
 - ▶ From constant features on example `--baseline`
 - ▶ Or separate global constant example `--baseline --global_only`
- ▶ Residual regression on the other features

Baseline

Solution:

- ▶ Learn baseline regressor separately from rest
 - ▶ From constant features on example `--baseline`
 - ▶ Or separate global constant example `--baseline --global_only`
- ▶ Residual regression on the other features

Note: learning rate multiplied by max label to converge faster than other normalized updates

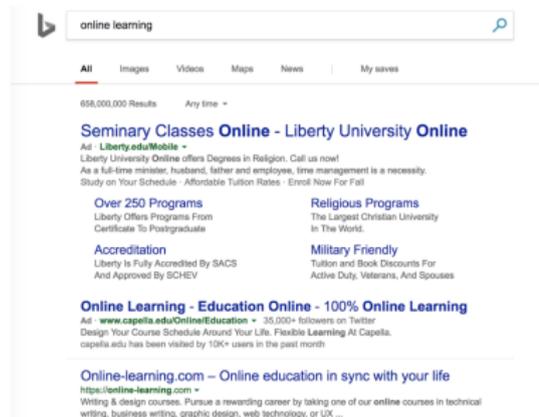
Contextual Bandits

Repeat:

- Get some **context** x
 - Search query, user info, user's interests
- Choose **action** a
 - Advertisement
 - News story
 - Medical treatment
- Observe **reward/loss** $r(a)$
 - Click/no click
 - Revenue
 - Treatment outcome

Goal: maximize **cumulative reward**

How? Balance **exploration/exploitation**



The screenshot shows a search engine interface with the query "online learning". The results are categorized by "All", "Images", "Videos", "Maps", "News", and "My saves". The top result is for "Seminary Classes Online - Liberty University Online", which includes a sub-headline "Liberty University Online offers Degrees in Religion. Call us now!" and a description: "As a full-time minister, husband, father and employee, time management is a necessity. Study on Your Schedule - Affordable Tuition Rates - Enroll Now For Fall". Below this are two columns of related links: "Over 250 Programs" (Liberty Offers Programs From Certificate To Postgraduate) and "Religious Programs" (The Largest Christian University In The World). The second main result is "Online Learning - Education Online - 100% Online Learning" from Capella University, with a sub-headline "Design Your Course Schedule Around Your Life. Flexible Learning At Capella." and a description: "capella.edu has been visited by 10k+ users in the past month". A third result is for "Online-learning.com - Online education in sync with your life" with a sub-headline "Writing & design courses. Pursue a rewarding career by taking one of our online courses in technical writing, business writing, graphic design, web technology, or UX ...".

Baseline: example for cb loss estimates

Reward/loss estimation is key (e.g. doubly robust)

Baseline: example for cb loss estimates

Reward/loss estimation is key (e.g. doubly robust)

```
> vw ds.txt --cbify 10 --cb_explore_adf --cb_type dr --epsilon 0.05  
0.682315
```

```
> vw ... --loss0 9 --loss1 10  
0.787594
```

```
> vw ... --loss0 9 --loss1 10 --baseline  
0.710636
```

```
> vw ... --loss0 9 --loss1 10 --baseline --global_only  
0.636140
```

Contextual bandits: bagging

Bagging: “bootstrapped Thompson sampling”

- ▶ Each update is performed *Poisson*(1) times
- ▶ For only one policy, **greedy** performs better (always update once)

Contextual bandits: bagging

Bagging: “bootstrapped Thompson sampling”

- ▶ Each update is performed $Poisson(1)$ times
- ▶ For only one policy, **greedy** performs better (always update once)
- ▶ `--bag n --greedify` treats first policy like greedy
- ▶ Often works better, especially for small n

Contextual bandits: cover

Cover: maintains set of diverse policies good for explore/exploit

- ▶ New parameterization: `--cover n [--psi 0.01] [--nounif]`
 - ▶ ψ controls diversity cost for training policies ($\psi = 0 \rightarrow$ all ERM policies)
 - ▶ $\epsilon_t = 1/\sqrt{Kt}$ always
 - ▶ `--nounif` disables exploration on ϵ actions (not chosen by any policy)

Contextual bandits: miscellaneous

- ▶ Most changes are only in the **ADF** code
- ▶ `--cbify K --cb_explore_adf` for using ADF code in cbify
 - ▶ `--loss0 [0] --loss1 [1]` to specify different loss encodings
- ▶ **Cover** + **MTR** uses MTR for the first (ERM) policy, DR for the rest
- ▶ **Upcoming**: reduce uniform ϵ exploration in ϵ -greedy using disagreement test (from active learning)