AI at the Webscale Project Results

Bas Bootsma & Fenno Vermeij

Radboud University Nijmegen

30th June 2015

Approach

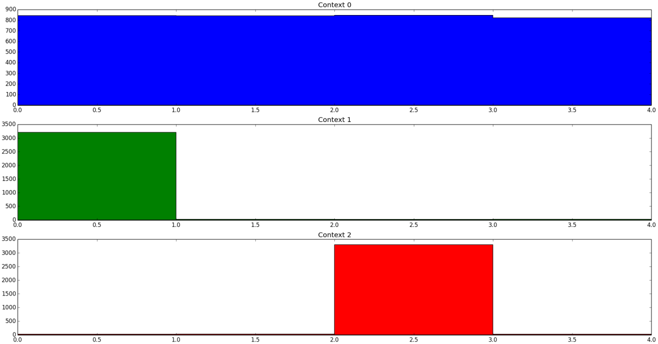
- Epsilon-greedy - Good baseline
- Gibbs-sampling - Too computationally expensive
- Thompson-sampling

Model

$$\begin{aligned} r = \beta_0 + \beta_{x_1} c_1 + \ldots + \beta_{x_k} c_k + \\ \beta_{y_1} a_1 + \ldots + \beta_{y_l} a_l + \\ \beta_{z_1} c_1 a_1 + \ldots + \beta_{z_m} c_k a_l \end{aligned}$$

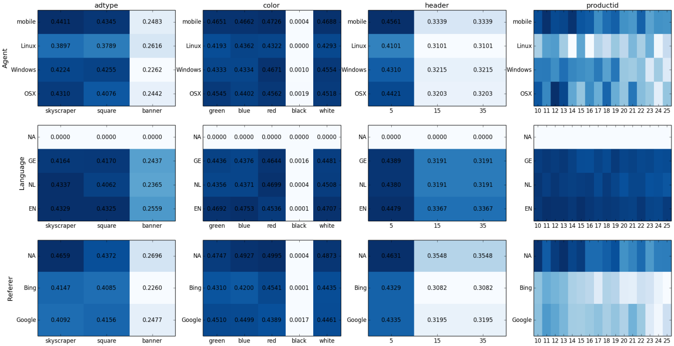
- Reward for update: use *price* · *effect* instead of *effect*
- Price: Maximize polynomial: $\beta_0 + \beta_1 \cdot p + \beta_2 \cdot p^2$ instead of bucketing: [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

Visualization of policy



- Pre-defined distribution for 3 context parameters, and 4 arms

Visualization of context vs. proposal



- Every possible combination of proposal parameters, except *price* = 1

Miscellaneous improvements

- Multivariate Gaussian speedup: using Cholesky transformation
- Use 5000 random interactions to give model 'warm start' before doing actual predictions
- Add features for user ID: average price user paid previously, and whether the user actually bought anything

Miscellaneous improvements

- Multivariate Gaussian speedup: using Cholesky transformation
- Use 5000 random interactions to give model ‘warm start’ before doing actual predictions
- Add features for user ID: average price user paid previously, and whether the user actually bought anything

Miscellaneous improvements

- Multivariate Gaussian speedup: using Cholesky transformation
- Use 5000 random interactions to give model 'warm start' before doing actual predictions
- Add features for user ID: average price user paid previously, and whether the user actually bought anything

Miscellaneous improvements

- Multivariate Gaussian speedup: using Cholesky transformation
- Use 5000 random interactions to give model ‘warm start’ before doing actual predictions
- Add features for user ID: average price user paid previously, and whether the user actually bought anything

Results

- Algorithm tested on 16 runIDs
- Average reward: 16.75
- Standard deviation: 5.07
- Time taken: ∼01:25h per run
- Any questions?