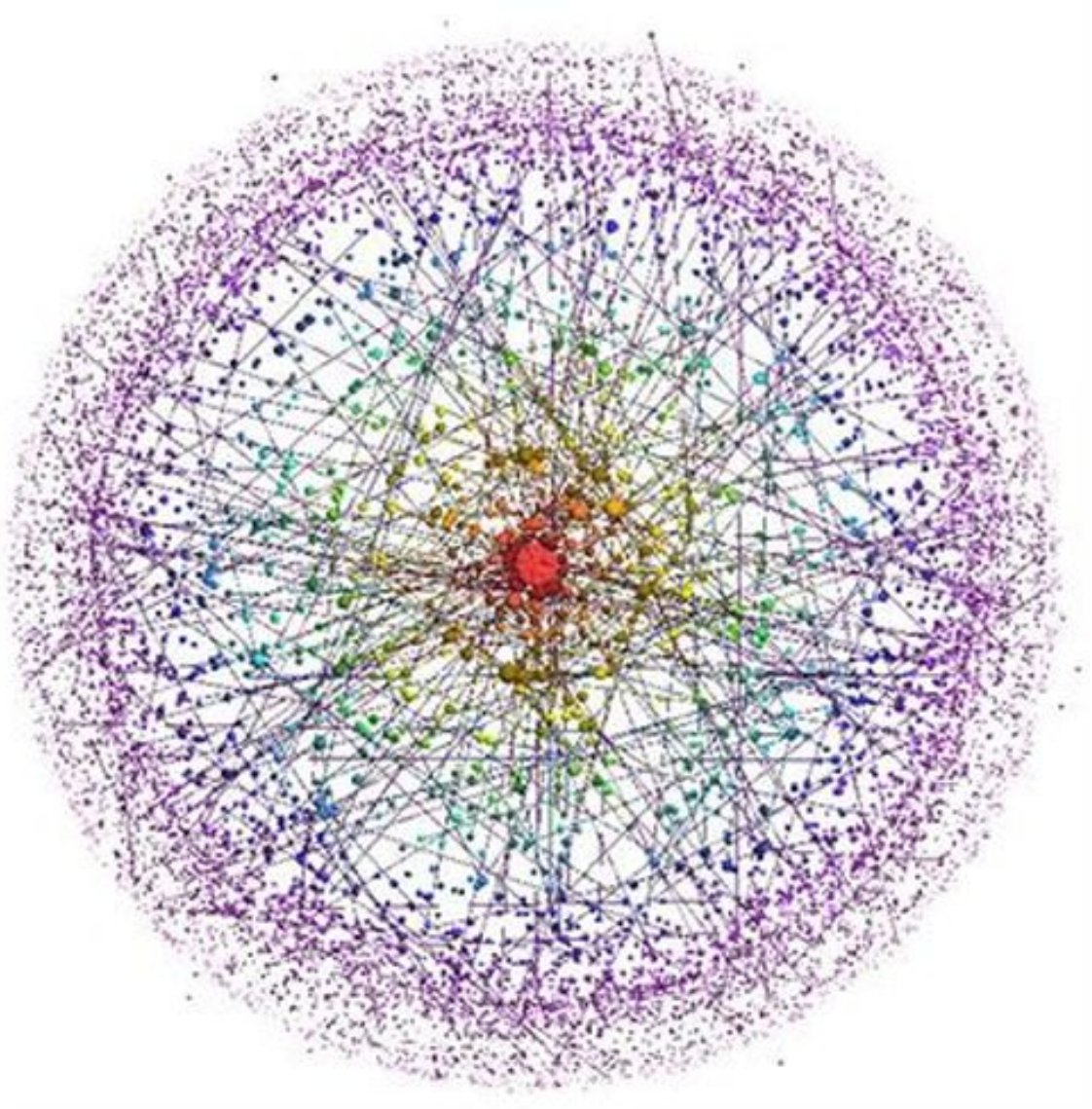
Part 2.
Recommendation for Real Systems



Pragmatic issues

I. SCALE-RELATED ISSUES

II. ABSTRACTION ISSUES

I. Scale-related issues in Recommendation

Real recommender systems are complicated because:

- User states are not just one event, but a timeseries of historical events
- Multiple recommendations are delivered at once (we do an argsort, not an argmax).
- We need to do this fast! Fast maximum inner product search / online KNN.

.

Real World Reco: An Example

Product view

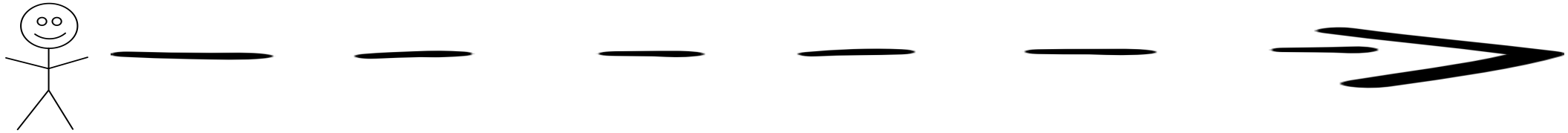
Product view

Product view

Product view

Product view

Recommend Banner



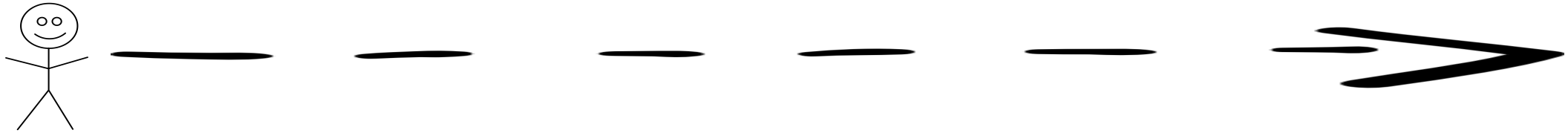
Product view

Product view

Recommend Banner

Banner =

$$argsort(g(v_1 = \text{  }, v_2 = \text{  })^T \beta)_{1:3}$$



Product view

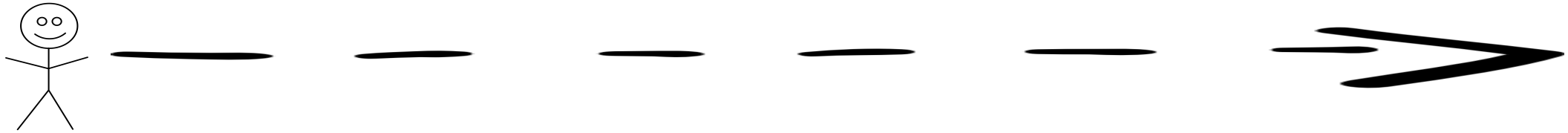
Product view

Recommend Banner

No click

Banner =

$$argsort(g(v_1 = \text{  }, v_2 = \text{  })^T \beta)_{1:3}$$



Product view

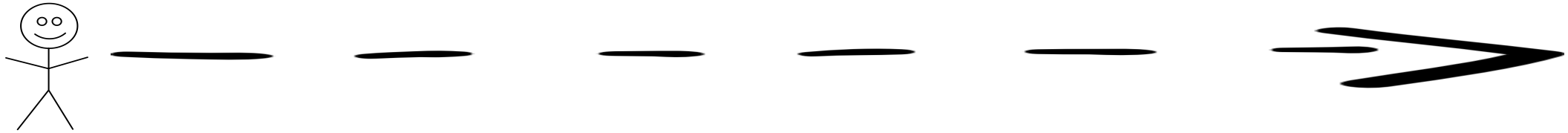
Product view

Recommend Banner

Product view

Banner =

$$argsort(g(v_1 = \text{  }, v_2 = \text{  })^T \beta)_{1:3}$$



Product view

Product view

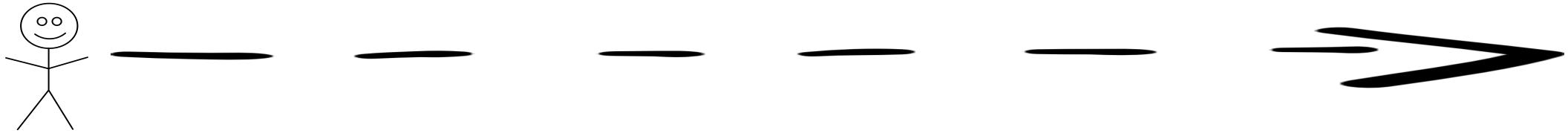
Recommend Banner

Product view

Recommend Banner

Banner = $argsort(g(v_1 =$  $, v_2 =$  $, v_3 =$  $)^T \beta)_{1:2}$

Banner = $argsort(g(v_1 =$  $, v_2 =$  $)^T \beta)_{1:3}$



Product view

Product view

Recommend Banner

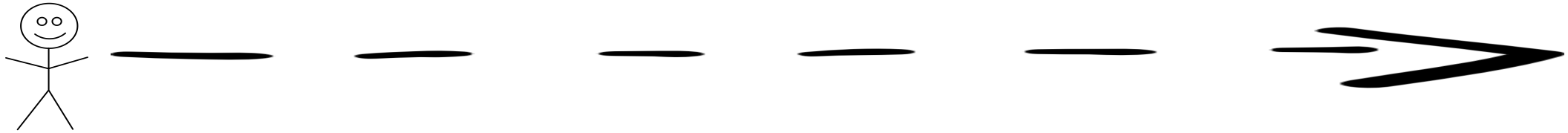
Product view

Recommend Banner

No click

Banner = $argsort(g(v_1 =$  $, v_2 =$  $, v_3 =$  $)^T \beta)_{1:2}$

Banner = $argsort(g(v_1 =$  $, v_2 =$  $)^T \beta)_{1:3}$



Product view

Product view

Recommend Banner

Product view

Recommend Banner

Sale

The recommender system is described by: $g(\cdot), \beta$

Banner = $argsort(g(v_1 =$  $, v_2 =$  $, v_3 =$  $)^T \beta)_{1:2}$

Banner = $argsort(g(v_1 =$  $, v_2 =$  $)^T \beta)_{1:3}$

How do we find the best $g(\cdot), \beta$?

- A/B test
  - Split traffic into $g_a(\cdot), \beta_a$ and $g_b(\cdot), \beta_b$ - we can then measure anything we want over the two populations
  - Not especially noisy!

- … but if $g_a(\cdot), \beta_a$ is production and $g_b(\cdot), \beta_b$ is our new proposal, how do we get $g_b(\cdot), \beta_b$ with a decent chance of success?

- The decision rule based and model based approaches are not easy to apply here…. but let's try …

On the old system $(g_a(\cdot), \beta_a)$, we know how it works from running it..

Product view

Product view

Recommend Banner

Product view

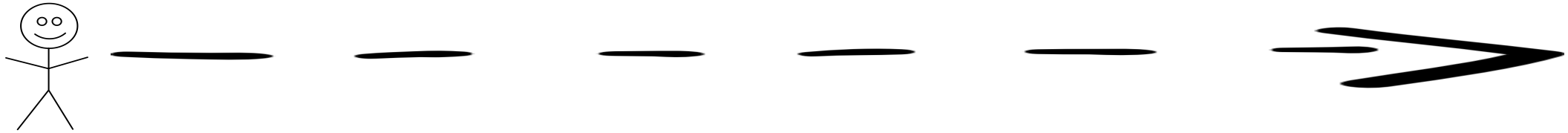
Recommend Banner

Sale

Banner = $argsort(g_a(v_1$  $, v_2$ =  $, , v_3$  $)^T \beta_a)_{1:2}$

Banner = $argsort(g_a(v_1$ =  $, v_2$ =  $)^T \beta_a)_{1:3}$

The new system $(g_b(\cdot), \beta_b)$ does new recommendations



Product view

Product view

Recommend Banner

Product view

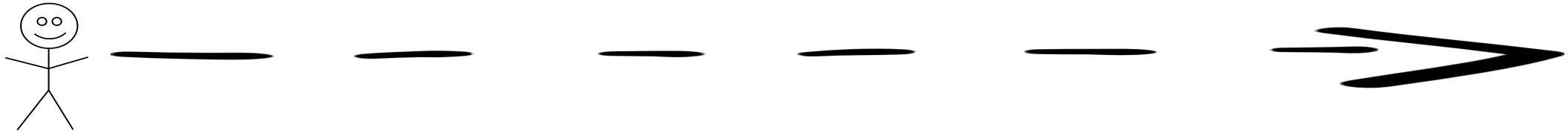
Recommend Banner

Sale

Banner = $argsort(g_b(v_1$  $, v_2$  $, , v_3$  $)^T \beta_b)_{1:2}$

Banner = $argsort(g_b\ (v_1$  $, v_2$  $)^T \beta_b)_{1:3}$

We now do not know anything after the new reco



Product view

Product view

Recommend Banner

Product view

Recommend Banner

Sale

Banner = $argsort(g_b(v_1$  $, v_2 =$  $,, v_3$  $)^T \beta_b)_{1:2}$

Banner = $argsort(g_b\ (v_1 :$  $, v_2$  $)^T \beta_b)_{1:3}$

Keeping it simple – let's just evaluate the first bit.. Is this a good banner (is the ranking good)?

Product view

Product view

Recommend Banner

We need to know:

$P(c|v_1 =$  $, v_2 =$  $, a_1 =$  $a_2 =$  $, a_3 =$  $)$

One option.. Let's look back in history and find all the occurrences where we observed precisely this sequence (it may not be many!!).

Banner =

$argsort(g_b\ (v_1 :$  $, v_2$  $)^T \beta_b)_{1:3}$

Keeping it simple – let's just evaluate the first bit.. Is this a good banner (is the ranking good)?

Product view

Product view

Recommend Banner

Banner =

$argsort(g_b\ (v_1 =$  $, v_2 =$  $)^T \beta_b)_{1:3}$

We need to know:

$P(c|v_1 =$  $, v_2 =$  $, a_1 =$  $a_2 =$  $, a_3 =$  $)$

The naive option: Let's look back in history and find all the occurrences where we observed precisely this sequence!

We can then count how often we obtain a click / sale.

Obviously, we have no hope to compute the click through rate using: clicks/impressions.

Keeping it simple – let's just evaluate the first bit.. Is this a good banner?

Product view

Product view

Recommend Banner

$$P(c|v_1 = \text{  }, v_2 = \text{  }, a_1 = \text{  } \; a_2 = \text{  }, a_3 = \text{  })$$

Modelling solution: For example, we could use **BLOB: Bayesian Latent Organic Bandit** (Sakhi, Otmane, et al 2020), could be used. This has advantages in the sense that similar items can be used to borrow strength. In BLOB, we leverage three key similarities, between actions, between histories and between action and history.

Banner =

$$argsort(g_b\ (v_1 = \text{  }, v_2 = \text{  })^T \beta_b)_{1:3}$$

Keeping it simple – let's just evaluate the first bit.. Is this a good banner (is the ranking good)?

Product view

Product view

Recommend Banner

$P(c|v_1 =$  $, v_2 =$  $, a_1 =$  $a_2 =$  $, a_3 =$  $)$

Policy solution: A **TOPK-REINFORCE** (Chen, Minmin, et al 2019) style algorithm could be used. The combinatorial nature of user contexts and banner actions can be tackled by embedding them in a vectorial space where all products are co-represented.

Banner =

$argsort(g_b\ (v_1 :$  $, v_2$  $)^T \beta_b)_{1:3}$

Complex Model / Simple Decision Rule

- As we get closer to real-world objectives, the model of reality becomes more and more complex (banner actions, delayed rewards)
- On the other hand, due to scaling limitations, most of the time the decision rule will stay in the same class of simple ranking functions

Proposal:

- Build a complex model on which we fit a simple decision rule
- If we skip the model, we won't be able to have offline metrics fully aligned with online performance!

Challenges of a real system

Producing an accurate model is hard:

- Modelling approaches could be *very* complicated and computationally expensive
- IPS methods will have significant variance issues as the actions become combinatorial

The decision rule is not just selecting a single item:

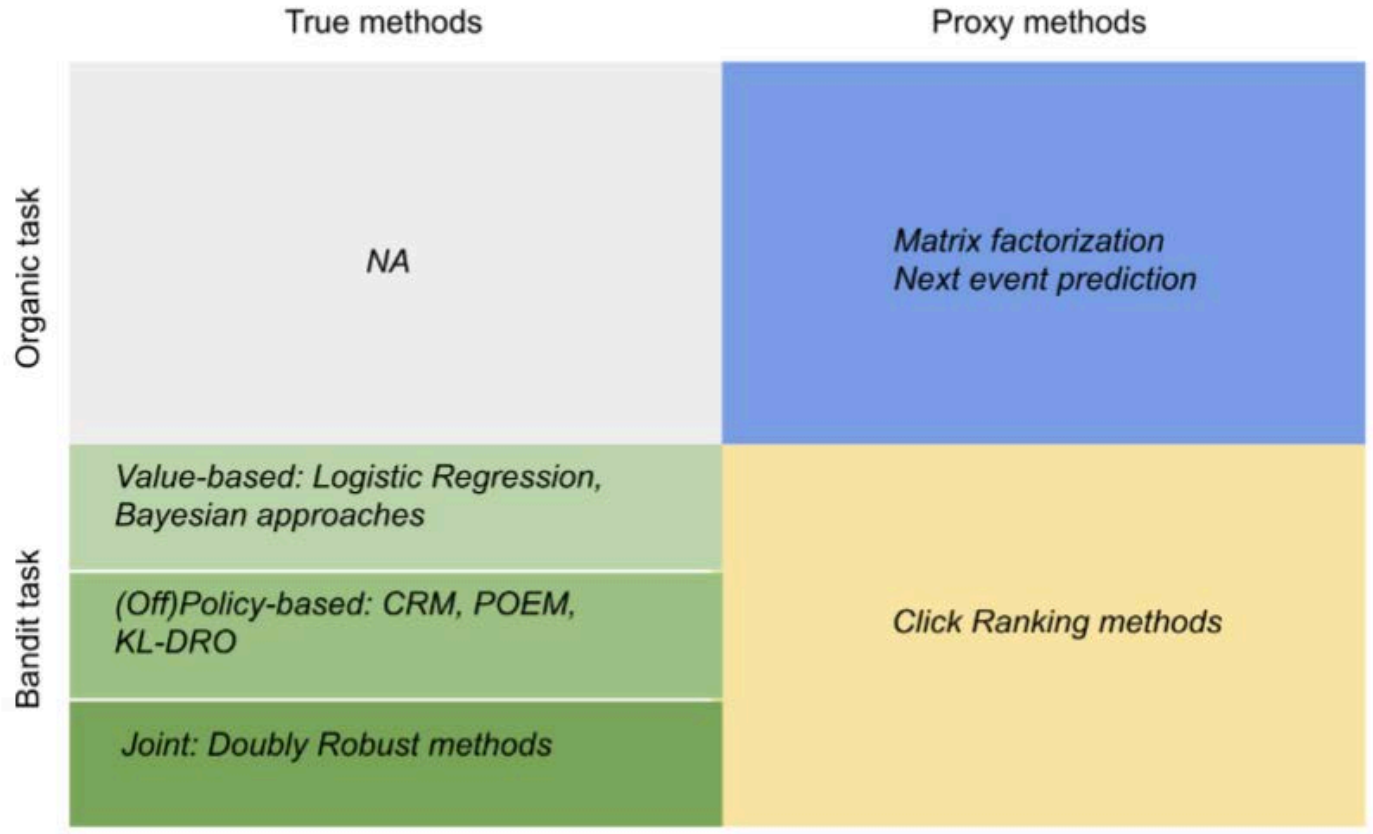
- The ranker will return K items
- Softmax no longer applies, need a differentiable argsort

Some fixes

- **To reduce variance/bias:** Blending the IPS and model-based solutions via Doubly Robust, CAB-like estimators (Su, Yi, et al. 2019)
- **To return top K:** use the gradient re-weighting trick proposed in TOPK-REINFORCE (Chen, Minmin, et al 2019)
- **To handle missing support for IPS**: reduce the support of the new policy, use a model to impute missing values as in (Sachdeva, Noveen et al 2020)

Or… we could **use proxy methods!**

Real World Solution: Use Proxies & ABTest



Real World Solution: Use Proxies & ABTest

Organic-based Proxy Methods

Collaborative Filtering via Matrix Factorization (reconstruction error)

- SVD
- NMF
- FFM
- VAEs

Next user event prediction via Sequential Modeling (ranking metrics)

- Prod2Vec, MetaProd2Vec
- RNNs, 2DCNN, TCN, ATTN

Real World Solution: Use Proxies & ABTest

Bandit-based Proxy Methods

- Based on methods from Search Optimization Literature.

- As in the learning-to-rank task, where the user is presented with a list of results, if the recommendation returns k items, the clicked item is considered a positive and the non-clicked items as negatives.

- However, this is not measuring the actual utility of showing the entire list of items and its therefore a proxy method.

Real World Solution: Use Proxies & ABTest

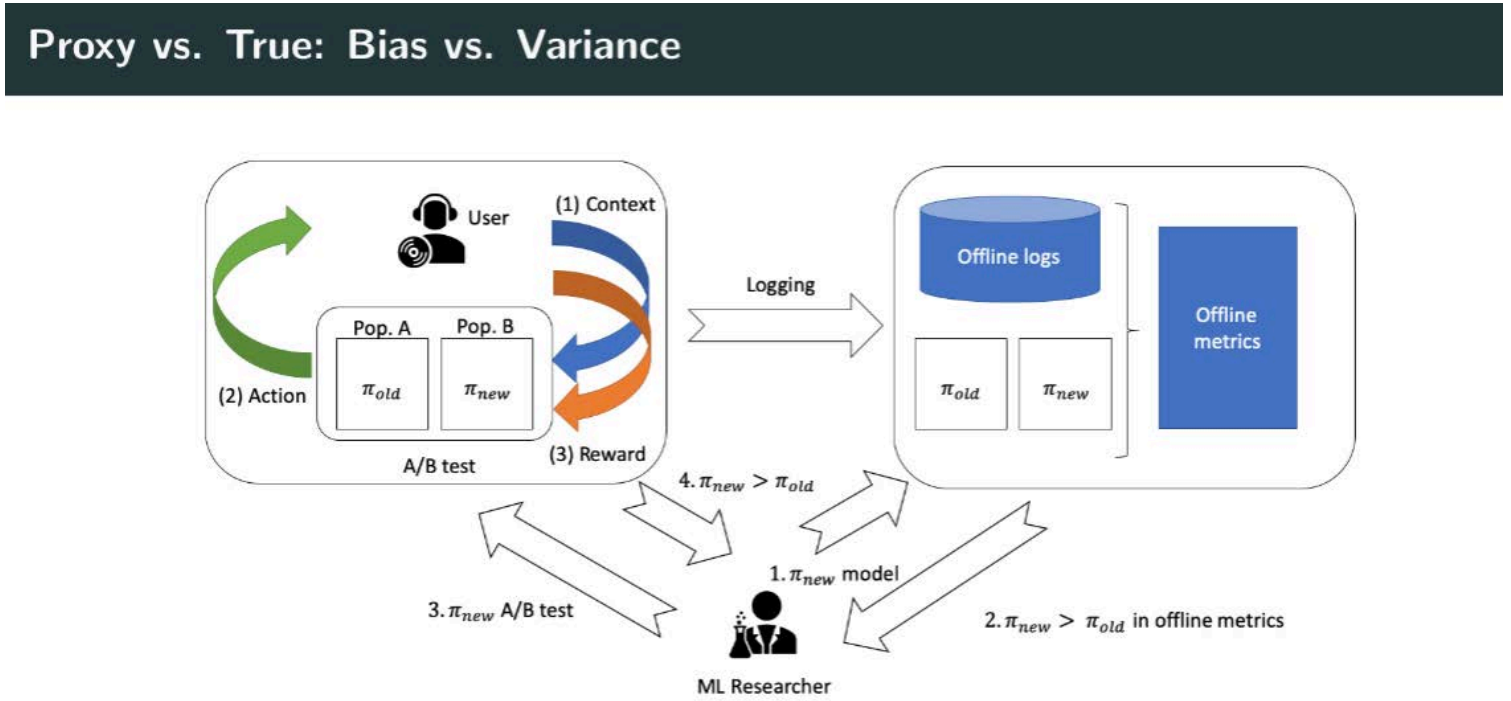
Bandit-based Proxy Methods

- At Criteo, we use **DeepKNN**, a deep learning framework for optimising a decision rule/ranker of the form:

Best k recommendations = $[\text{argsort}(g_a(\cdot)\beta_a)]_{1:K}$

- The model is optimized for a clickrank loss

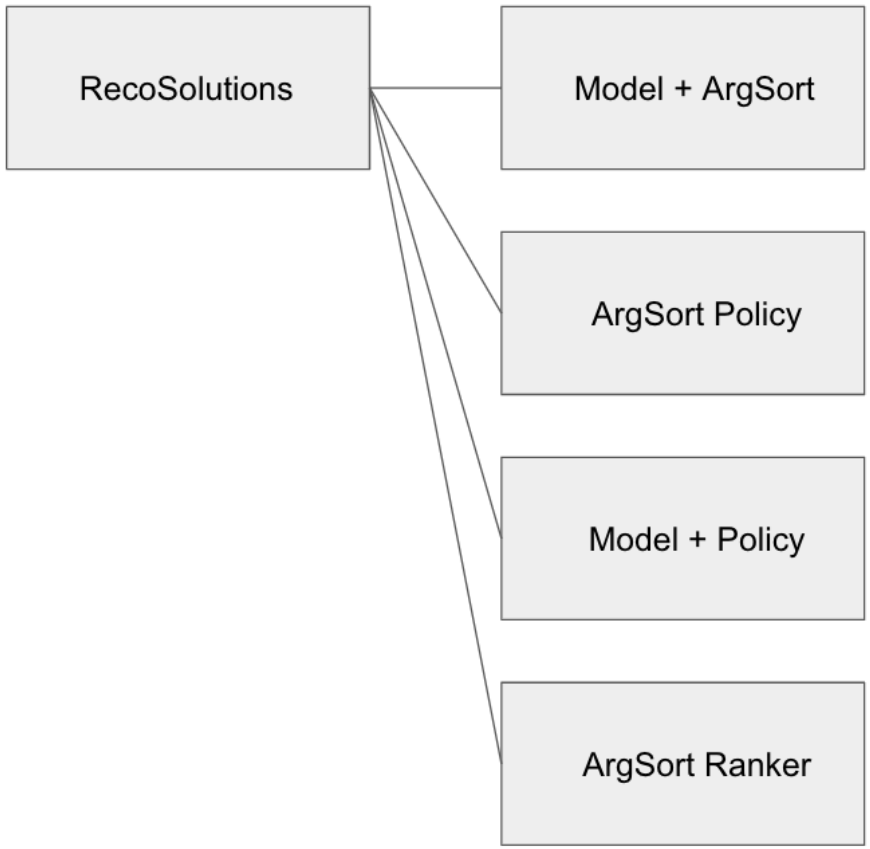
Real World Solution: Use Proxies & ABTest



The pragmatic compromise of proxy methods

- In practice **True reco methods** face these challenges:
  - Getting a realistic reward estimate for every action is difficult
  - IPS-based methods will suffer for extreme variance and might get stuck due to not sufficient exploration/diversity
- **Proxy methods sidestep** these issues:
  - Heuristics are used in order to assess quality of the ranking
  - Optimisation uses ranking which:
    - Produces a ranking with some reasonable properties for K>1
    - Do not require the slow evaluation of a softmax
- It is difficult to do theory or answer specific questions about how to tune proxy methods, but there is a reason why they are widely used.

TO RECAP…



Pros: Low Variance, True offline metrics
Cons: Bias due to misspecification
(It's hard to get a good model for complex problems)

Pros: Low Bias, True offline metrics
Cons: High Variance

Pros: Theoretically best of both worlds
Cons: The reduction in variance might not be that big. The resulting solution is complex

Pros: Low Bias, Proxy offline metrics
Cons: Cant work for complex objectives (banner level actions, delayed rewards)

II. Abstraction Scope Issues

There are cases where the myopic application of Decision Theory on the predefined Decision System scope is not correct due to ignoring/forgetting some aspects:

- **Decision System & Time:** Forgetting that the Decision System will be repeatedly used over time and ignoring how the immediate reward plays into the global reward
- **Adversarial situations:** Forgetting that the Decision System will be used in an adversarial environment. Examples: repeated auctions, item recommendation/ranking gaming, spam

.

Abstraction scope issues: DS + time

3 Games where we want to optimize a DS (RecSys) with different objectives:

Game1 (Reinvesting): The local reward at time [t] gets reinvested at time [t+1]

Game2 (Stay profitable): The sum of cumulative rewards at any time t needs to be > 0, some actions can have big negative rewards

Game3 (Make the most money over time, full bailout): Maximize the cumulative reward over time

Abstraction scope issues: DS + time

Game1: The local reward at time [t] gets reinvested at time [t+1]

Potential solution: Kelly's Criterion (e.g. penalize by the variance of the reward)

Motivation: The global reward is a multiplication of local rewards so it is as weak as its worst one.

Abstraction scope issues: DS + time

Game2: The sum of cumulative rewards at any time t needs to be > 0, some actions can have big negative rewards

Potential solution: Pessimism in the face of Uncertainty, e.g. penalize by variance or other measure of uncertainty:

See the work on:

- **Sample Variance Penalization (SVP)** (Swaminathan, Adith, and Thorsten Joachims, 1995)
- **Distributional Robust Optimization (DRO)** (Faury, Louis, et al.,2020)

Another solution: Using a concave/diminishing-returns utility function can accomplish the same thing (Jeunen, Olivier, et al.2020)

Motivation: Even if the global reward is the sum of local rewards, the >0 constraint makes the local DS want to be risk-aversive.

Abstraction scope issues: DS + time

Game3: Maximize the cumulative reward over time

Potential solution: Optimism in the face of uncertainty

See the work on:

- Upper Confidence Bound (UCB) (P. Auer, N. Cesa-Bianchi, 2002)

Motivation: Since you are simply interested in maximizing cumulative rewards, it pays to explore and spend time to look for the best possible action early on.

Adversarial Situations in Recommendation

- There are cases where we can expect that the reward signal can be tempered with for the goal of changing the relative rankings of items (e.g. inflate the relative popularity of an item in the app store)

- **Potential solution:** Frame the problem as an adversarial attack on the reward with limited budget which under some assumptions becomes a DRO/SVP problem (with a minmax solution)

In Conclusion

- Pragmatic application of Decision Theory comes with a set of pitfalls that we need to be aware of.
- The abstraction and problem definition steps can be sometimes crucial
- A lot of the time, knowing exactly what we want to solve is half of the work

Coming after the break

- Practical Notebooks on Decision Theory for Reco
- Policy vs. Model – Convergence Properties / Joint approaches

Thanks!