Approaches to Recommendation in Industry

Xavier Amatriain
xavier@amatriain.net
@xamat
Quora/Xavier-Amatriain
Recsys 2017 Summer School
Bolzano, IT
Outline

1. The Traditional Recommender Problem
2. The Netflix Prize
3. Beyond Rating Prediction
4. Lessons Learned
5. A Recsys Architectural Blueprint
6. Building a state-of-the-art recommender system in practice
7. Hands-on tutorial
8. Future research Directions
9. Conclusions
10. Some references
1. The Recommender Problem
The “Recommender problem”

- Traditional definition: Estimate a utility function that automatically predicts how much a user will like an item.
- Based on:
  - Past behavior
  - Relations to other users
  - Item similarity
  - Context
  - ...

Recommendation as data mining

The core of the Recommendation Engine can be assimilated to a general data mining problem

Data Mining + all those other things

- User Interface
- System requirements (efficiency, scalability, privacy....)
- Serendipity
- Diversity
- Awareness
- Explanations
- ...

...
Serendipity

- Unsought finding
- **Don't recommend** items the user already knows or would have found anyway.
- Expand the user's taste into neighboring areas by improving the obvious
- Serendipity ~ Explore/exploit tradeoff
Explanation/Support for Recommendations
Diversity & Awareness

Personalization awareness

Top 10 for Xavier

Diversity
What works

- Depends on the **domain** and particular **problem**
- However, in the general case it has been demonstrated that the best isolated approach is CF.
  - Other approaches can be hybridized to improve results in specific cases (cold-start problem...)
- **What matters:**
  - **Data preprocessing:** outlier removal, denoising, removal of global effects (e.g. individual user's average)
  - “Smart” **dimensionality reduction** using MF
  - **Combining methods** through ensembles
2. The Netflix Prize
What we were interested in:

- High quality *recommendations*

Proxy question:

- Accuracy in predicted rating

\[
\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2}
\]
2007 Progress Prize

- Top 2 algorithms
  - SVD - Prize RMSE: 0.8914
  - RBM - Prize RMSE: 0.8990
- Linear blend Prize RMSE: 0.88
- Currently in use as part of Netflix’ rating prediction component
- Limitations
  - Designed for 100M ratings, not XB ratings
  - Not adaptable as users add ratings
  - Performance issues
What about the final prize ensembles?

- Offline studies showed they were too computationally intensive to scale
- Expected improvement not worth engineering effort
- Plus.... Focus had already shifted to other issues that had more impact than rating prediction.
3. Beyond Rating Prediction
Everything is a recommendation
Evolution of the Recommender Problem

Rating → Ranking → Page Optimization → Context-aware Recommendations
3.1 Ranking
Ranking

- Most recommendations are presented in a sorted list
- Recommendation can be understood as a ranking problem
- Popularity is the obvious baseline
- What about rating predictions?
Ranking by ratings

Niche titles
High average ratings... by those who would watch it
RMSE
Example: Two features, linear model

Linear Model:
\[ f_{\text{rank}}(u,v) = w_1 p(v) + w_2 r(u,v) + b \]
Example: Two features, linear model
Learning to rank

- Machine learning problem: goal is to construct ranking model from training data
- Training data can be a partial order or binary judgments (relevant/not relevant).
- Resulting order of the items typically induced from a numerical score
- Learning to rank is a key element for personalization
- You can treat the problem as a standard supervised classification problem
Learning to rank - Metrics

- Quality of ranking measured using metrics as
  - Normalized Discounted Cumulative Gain
  - Mean Reciprocal Rank (MRR)
  - Fraction of Concordant Pairs (FCP)
  - Others...

- But, it is hard to optimize machine-learned models directly on these measures (e.g. non-differentiable)

- Recent research on models that directly optimize ranking measures
Goal: Present most interesting stories for a user at a given time

Interesting = topical relevance + social relevance + timeliness

Stories = questions + answers

ML: Personalized learning-to-rank approach

Relevance-ordered vs time-ordered = big gains in engagement
3.2 Similarity
• Displayed in many different contexts
  ○ In response to user actions/context (search, queue add...)
  ○ More like... rows
Similars: Related Questions

- Given interest in question A (source) what other questions will be interesting?
- Not only about similarity, but also “interestingness”
- Features such as:
  - Textual
  - Co-visit
  - Topics
  - ...
- Important for logged-out use case
Graph-based similarities

- The Fighter: 0.2 to Rango
- Mad Men: 0.8 to The Fighter, 0.3 to Rango, 0.3 to How I Met Your Mother
- How I Met Your Mother: 0.7 to Rango
- Rango: 0.4 to The Fighter, 0.3 to How I Met Your Mother
Example of graph-based similarity: SimRank

- **SimRank (Jeh & Widom, 02):** “two objects are similar if they are referenced by similar objects.”

\[
s(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{I(a)} \sum_{j=1}^{I(b)} s(I_i(a), I_j(b))
\]

![Figure 1](image)

Figure 1: A small Web graph \( G \) and simplified node-pairs graph \( G^2 \). SimRank scores using parameter \( C = 0.8 \) are shown for nodes in \( G^2 \).
Similarity ensembles

- Similarity can refer to different dimensions
  - Similar in metadata/tags
  - Similar in user play behavior
  - Similar in user rating behavior
  - ...

- Combine them using an ensemble
  - Weights are learned using regression over existing response
  - Or... some MAB explore/exploit approach

- The final concept of “similarity” responds to what users vote as similar
3.3 Social Recommendations
Recommendations - Users

- **Goal: Recommend new users to follow**
- **Based on:**
  - Other users followed
  - Topics followed
  - User interactions
  - User-related features
  - ...
User Trust/Expertise Inference

● **Goal:** Infer user’s trustworthiness in relation to a given topic

● We take into account:
  ○ Answers written on topic
  ○ Upvotes/downvotes received
  ○ Endorsements
  ○ ...

● Trust/expertise propagates through the network

● Must be taken into account by other algorithms
Social and Trust-based recommenders

- A social recommender system recommends items that are “popular” in the social proximity of the user.
- Social proximity = trust (can also be topic-specific)
- Given two individuals - the source (node A) and sink (node C) - derive how much the source should trust the sink.
- Algorithms
  - Advogato (Levien)
  - Appleseed (Ziegler and Lausen)
  - MoleTrust (Massa and Avesani)
  - TidalTrust (Golbeck)
Other ways to use Social

- Social connections can be used in combination with other approaches
- In particular, “friendships” can be fed into collaborative filtering methods in different ways
  - replace or modify user-user “similarity” by using social network information
  - use social connection as a part of the ML objective function as regularizer
  - ...
3.4 Explore/Exploit
One of the key issues when building any kind of personalization algorithm is how to trade off:
- **Exploitation**: Cashing in on what we know about the user right now
- **Exploration**: Using the interaction as an opportunity to learn more about the user

We need to have informed and optimal strategies to drive that tradeoff
- **Solution**: pick a reasonable set of candidates and show users only “enough” to gather information on them
Multi-armed Bandits

- Given possible strategies/candidates (slot machines) pick the arm that has the maximum potential of being good (minimize regret)

- Naive strategy: $\varepsilon$-greedy
  - Explore with a small probability $\varepsilon$ (e.g. 5%) -> choose an arm at random
  - Exploit with a high probability $(1 - \varepsilon)$ (e.g. 95%) -> choose the best-known arm so far

- Translation to recommender systems
  - Choose an arm = choose an item/choose an algorithm (MAB testing)

- Thompson Sampling
  
  Given a posterior distribution, sample on each iteration and choose the action that maximizes the expected reward
Multi-armed Bandits

Explore-Exploit in Top-N Recommender Systems via Gaussian Processes

Hastagiri P Vanchinathan
ETH Zurich
hastagiri@inf.ethz.ch

Isidor Nikolov
Microsoft, Zurich
inikolov@microsoft.com

Andreas Krause
ETH Zurich
krausea@ethz.ch

Fabio De Bona
Google, Zurich
fdb@google.com

A Contextual-Bandit Approach to Personalized News Article Recommendation

Lihong Li, Wei Chu,
Yahoo Labs
lihong.chuwei@yahoo-inc.com

John Langford
Yahoo Labs
jl@yahoo-inc.com

Robert E. Schapire*
Dept of Computer Science
Princeton University
schapire@cs.princeton.edu

Context Adaptation in Interactive Recommender Systems

Negar Hariri
DePaul University
Chicago, IL 60604, USA
nhariri@cs.depaul.edu

Bamshad Mobasher
DePaul University
Chicago, IL 60604, USA
mobasher@cs.depaul.edu

Robin Burke
DePaul University
Chicago, IL 60604, USA
rburke@cs.depaul.edu

LinkedIn

Recommending Items to Users: An Explore Exploit Perspective

Deepak Agarwal, Director Machine Learning and Relevance Science, LinkedIn, USA

CIKM, 2013
3.5 Page Optimization
10,000s of possible rows

Variable number of possible videos per row (up to thousands)

1 personalized page

10-40 rows per device
From “Modeling User Attention and Interaction on the Web” 2014 - PhD Thesis by Dmitry Lagun (Emory U.)
User Attention Modeling

Web Search (Google)   Social Network (Twitter)

News (CNN)           Shopping (Amazon)

From “Modeling User Attention and Interaction on the Web” 2014 - PhD Thesis by Dmitry Lagun (Emory U.)
Page Composition

Accurate vs. Diverse
Discovery vs. Continuation
Depth vs. Coverage
Freshness vs. Stability
Recommendations vs. Tasks

- To put things together we need to combine different elements
  - Navigational/Attention Model
  - Personalized Relevance Model
  - Diversity Model

Beyond Ranking: Optimizing Whole-Page Presentation

Fair and Balanced: Learning to Present News Stories
3.6 Beyond user/rating
N-dimensional model

[Adomavicius et al., 2005]
Tensor Factorization

HOSVD: Higher Order Singular Value Decomposition

$$U \in \mathbb{R}^{n \times d_U}, \quad M \in \mathbb{R}^{m \times d_M} \quad \text{and} \quad C \in \mathbb{R}^{c \times d_C}$$

$$S \in \mathbb{R}^{d_U \times d_M \times d_C}$$

$$R[U, M, C, S] := L(F, Y) + \Omega[U, M, C] + \Omega[S]$$
Tensor Factorization


Where:

\[ \Omega[F] = \lambda_M \|M\|_F^2 + \lambda_U \|U\|_F^2 + \lambda_C \|C\|_F^2 \]

\[ \Omega[S] := \lambda_S \|S\|_F^2 \]

- We can use a simple squared error loss function:
  \[ l(f, y) = \frac{1}{2} (f - y)^2 \]

- Or the absolute error loss
  \[ l(f, y) = |f - y| \]

- The loss function over all users becomes
  \[ L(F, Y) = \sum_{i}^{n} \sum_{j}^{m} l(f_{ij}, y_{ij}) \]
Factorization Machines

- Generalization of regularized matrix (and tensor) factorization approaches combined with linear (or logistic) regression

- Problem: Each new adaptation of matrix or tensor factorization requires deriving new learning algorithms
  - Hard to adapt to new domains and add data sources
  - Hard to advance the learning algorithms across approaches
  - Hard to incorporate non-categorical variables
Factorization Machines

• Approach: Treat input as a real-valued feature vector
  – Model both linear and pair-wise interaction of $k$ features (i.e. polynomial regression)
  – Traditional machine learning will overfit
  – Factor pairwise interactions between features
  – Reduced dimensionality of interactions promote generalization
  – Different matrix factorizations become different feature representations
  – Tensors: Additional higher-order interactions

• Combines “generality of machine learning/regression with quality of factorization models”
Factorization Machines

- Each feature gets a weight value and a factor vector
  - $O(dk)$ parameters

$$b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, \mathbf{V} \in \mathbb{R}^{d \times k}$$

- Model equation:

$$f(\mathbf{x}) = b + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} x_i x_j \mathbf{v}_i^T \mathbf{v}_j$$

$$= b + \sum_{i=1}^{d} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left( \left( \sum_{i=1}^{d} x_i v_{i,f} \right)^2 - \sum_{i=1}^{d} x_i^2 v_{i,f}^2 \right)$$

$O(d^2)$ $O(kd)$
Factorization Machines

- Two categorical variables \((u, i)\) encoded as real values:

- FM becomes identical to MF with biases:

\[
f(x) = b + w_u + w_i + v_u^T v_i
\]

*From Rendle (2012) KDD Tutorial*
Factorization Machines

- Makes it easy to add a time signal

- Equivalent equation:

\[
f(x) = b + w_u + w_i + x_t w_t + \mathbf{v}_u^T \mathbf{v}_i + x_t \mathbf{v}_u^T \mathbf{v}_t + x_t \mathbf{v}_i^T \mathbf{v}_t
\]

From Rendle (2012) KDD Tutorial
Factorization Machines (Rendle, 2010)

- L2 regularized
  - Regression: Optimize RMSE
  - Classification: Optimize logistic log-likelihood
  - Ranking: Optimize scores
- Can be trained using:
  - SGD
  - Adaptive SGD
  - ALS
  - MCMC

Gradient:
\[
\frac{\partial}{\partial \theta} f(\mathbf{x}) = \begin{cases} 
1 & \text{if } \theta \text{ is } b \\
\mathbf{x}_i & \text{if } \theta \text{ is } w_i \\
\mathbf{x}_i \sum_{j=1}^{d} v_{j,f} x_j - v_{i,f} x_i^2 & \text{if } \theta \text{ is } v_{i,f}
\end{cases}
\]

Least squares SGD:
\[
\theta' = \theta - \eta \left( (f(\mathbf{x}) - y) \frac{\partial}{\partial \theta} f(\mathbf{x}) + \lambda \theta \right)
\]
Factorization Machines (Rendle, 2010)

- Learning parameters:
  - Number of factors
  - Iterations
  - Initialization scale
  - Regularization (SGD, ALS) – Multiple
  - Step size (SGD, A-SGD)
  - MCMC removes the need to set those hyperparameters
3.7 Deep Learning
(See Balázs Hidasi’s slides)
4. Lessons Learned
1. Implicit signals beat explicit ones (almost always)
Implicit vs. Explicit

- Many have acknowledged that implicit feedback is more useful
- Is implicit feedback really always more useful?
- If so, why?
Implicit vs. Explicit

● Implicit data is (usually):
  ○ More dense, and available for all users
  ○ Better representative of user behavior vs. user reflection
  ○ More related to final objective function
  ○ Better correlated with AB test results

● E.g. Rating vs watching
However

○ It is not always the case that direct implicit feedback correlates well with long-term retention

○ E.g. clickbait

Solution:

○ Combine different forms of implicit + explicit to better represent long-term goal
2. BE THOUGHTFUL ABOUT YOUR TRAINING DATA
Defining training/testing data

- Training a simple binary classifier for good/bad answer
  - Defining positive and negative labels -> Non-trivial task
  - *Is this a positive or a negative?*
    - funny uninformative answer with many upvotes
    - short uninformative answer by a well-known expert in the field
    - very long informative answer that nobody reads/upvotes
    - informative answer with grammar/spelling mistakes
    - ...
3. Your Model will learn what you teach it to learn.
Training a model

- Model will learn according to:
  - Training data (e.g. implicit and explicit)
  - Target function (e.g. probability of user reading an answer)
  - Metric (e.g. precision vs. recall)

- Example 1 (made up):
  - Optimize probability of a user going to the cinema to watch a movie and rate it “highly” by using purchase history and previous ratings. Use NDCG of the ranking as final metric using only movies rated 4 or higher as positives.
Example 2 - Quora’s feed

- Training data = implicit + explicit
- Target function: Value of showing a story to a user \( \sim \) weighted sum of actions:
  \[
  v = \sum_a v_a 1\{y_a = 1\}
  \]
  - predict probabilities for each action, then compute expected value: 
    \[
    v_{\text{pred}} = E[V | x] = \sum_a v_a p(a | x)
    \]
- Metric: any ranking metric
4. Explanations might matter more than the prediction.
Social Support

Explanation/Support for Recommendations

Sarah Smith, Richard Henry and 3 more upvoted this • 7h

How can I complain about my roommate who is cheating on his Google phone interviews?

Ben Garrison, Software Engineer at Google
304.3k Views • Upvoted by Jeremy Miles, Quantitative analyst at Google, Mayeesha Tahsin, Sarah Smith, and 3 others you follow

First off, I really appreciate your trying to make sure the right thing happens. I think that's great. Cheating sucks. However, the answer is "don't worry about it". Phone screens here at Google ar... (more)
5. Learn to deal with Presentation Bias
2D Navigational modeling

More likely to see

Less likely
The curse of presentation bias

● User can only click on what you decide to show
  ○ But, what you decide to show is the result of what your model predicted is good

● Simply treating things you show as negatives is not likely to work

● Better options
  ○ Correcting for the probability a user will click on a position -> Attention models
  ○ Explore/exploit approaches such as MAB
6. If You Have to Pick one single approach, Matrix factorization is your best bet.
Matrix Factorization

● MF can be interpreted as
  ○ Unsupervised:
    ■ Dimensionality Reduction a la PCA
    ■ Clustering (e.g. NMF)
  ○ Supervised:
    ■ Labeled targets \( \sim \) regression

● Very useful variations of MF
  ○ BPR, ALS, SVD++
  ○ Tensor Factorization, Factorization Machines

● However...
7. Everything is an ensemble
Ensembles

- Netflix Prize was won by an ensemble
  - Initially Bellkor was using GDBTs
  - BigChaos introduced ANN-based ensemble

- Most practical applications of ML run an ensemble
  - Why wouldn’t you?
  - At least as good as the best of your methods
  - Can combine different approaches (e.g. CF and content-based)
  - Can use different models at the ensemble layer: LR, GDBTs, RFs, ANNs...
Ensembles & Feature Engineering

- Ensembles are the way to turn any model into a feature!

- E.g. Don’t know if the way to go is to use Factorization Machines, Tensor Factorization, or RNNs?
  - Treat each model as a “feature”
  - Feed them into an ensemble
8. Building Recommender Systems is also about Feature Engineering
Need for feature engineering

In many cases an understanding of the domain will lead to optimal results.
Feature Engineering Example - Quora Answer Ranking

What is a good Quora answer?

- truthful
- reusable
- provides explanation
- well formatted
- ...
How are those dimensions translated into features?

- Features that relate to the answer quality itself
- Interaction features (upvotes/downvotes, clicks, comments...)
- User features (e.g. expertise in topic)
Feature Engineering

- Properties of a well-behaved ML feature:
  - Reusable
  - Transformable
  - Interpretable
  - Reliable
9. Why you should care about answering questions (about your recsys)
Model debuggability

- Value of a model = value it brings to the product
- Product owners/stakeholders have expectations on the product
- It is important to answer questions to why did something fail
- Model debuggability is so important it can determine:
  - Particular model to use
  - Features to rely on
  - Implementation of tools
Model debuggability

- E.g. Why am I seeing or not seeing this on my homepage feed?
10. Data and Models are great. You know what’s even better? The right evaluation approach!
Offline/Online testing process
11. You don’t need to distribute your Recsys
Distributing Recommender Systems

● Most of what people do in practice can fit into a multi-core machine
  ○ As long as you use:
    ■ Smart data sampling
    ■ Offline schemes
    ■ Efficient parallel code
● (... but not Deep ANNs)

● Do you care about costs? How about latencies or system complexity/debuggability?
Matrix Factorization Example
12. The UI is the only communication channel with what matters the most: **Users**
UI->Algorithm->UI

- The UI generates the user feedback that we will input into the algorithms
- The UI is also where the results of our algorithms will be shown
- A change in the UI might require a change in algorithms and vice versa
5. A Recsys Architectural Blueprint
- We want the same code/systems/tools to work for both experimentation & production.

- But we need to carefully “control” the production code to keep it be fast.

- So need to “control” offline experimentation systems too.
6. Building a state-of-the-art Recsys
6.1 Training, testing, and metrics
Training, testing, metrics

- As mentioned in the lessons, this is essential
- Choose implicit data and metrics that connect to your business goal
- Sample negatives smartly
- Select validation and test set carefully (e.g. avoid time traveling)
Training, testing, metrics

- For metrics, prefer ranking or ranking-related metrics
6.2 Implicit Matrix Factorization
Experience says, best single (simple) approach: implicit matrix factorization:

- ALS. Alternating Least Squares (Hu et al. 2008)
- BPR. Bayesian Personalized Ranked Ranking (Rendle et al. 2009)
Recommended Implementations

- Quora’s QMF
  - Efficient compiled C++ code
  - Supports many evaluation metrics
Recommended Implementations

- Implicit
  - Efficient
  - Python
  - Well-maintained
Sorry to say, but I cannot recommend any others (no, not Mahout)
6.3 A/B Test
AB Test

- So, you have your first implementation
  - Have tuned hyperparameters to optimize offline metric
  - How do you know this is working?
- Run AB Test!
  - Make sure offline metric (somewhat) correlates to online effect
AB Test

- Ideally, you would run several AB tests with different offline metrics and data sampling strategies.
6.4 Ensemble
Ensemble

- Now, it’s time to turn the model into a signal
- Brainstorm about some simple potential features that you could combine with implicit MF
  - E.g. user tenure, average rating for the item, price of the item…
- Add to MF through an ensemble
Ensemble

- What model to use at the ensemble layer?
  - Always favor most simple -> L2-regularized Logistic Regression
  - Eventually introduce models that can benefit from non-linear effects and many features -> Gradient Boosted Decision Trees
  - Explore Learning-to-rank models -> LambdaRank
6.5 Iterate, Feature Engineering
Iterate

- Experiment/add more features
- Experiment with more complex models
- Do both things in parallel
- Continue AB testing
7. Practical exercise
Exercise

- Train an ALS implicit matrix factorization recommender system
- Do basic feature engineering to add other features
- Add the mix to an XGBoost-based ensemble
- This is very close to what you could be using in real-life (minus scalability/performance issues)
- Detailed instructions [here](#)
8. Future Research Directions
Many interesting future directions

1. Indirect feedback
2. Value-awareness
3. Full-page optimization
4. Personalizing the how

Others
- Intent/session awareness
- Interactive recommendations
- Context awareness
- Deep learning for recommendations
- Conversational interfaces/bots for recommendations
- ...

Others
- Intent/session awareness
- Interactive recommendations
- Context awareness
- Deep learning for recommendations
- Conversational interfaces/bots for recommendations
- ...

Others
Indirect Feedback

Challenges

User can only click on what you show
But, what you show is the result of what your model predicted is good
No counterfactuals
Implicit data has no real “negatives”

Potential solutions

Attention models
Context is also indirect/implicit feedback
Explore/exploit approaches and learning across time

...
Value-aware recommendations

- Recsys optimize for probability of action
- Not all clicks/actions have the same “reward”
  - Different margin in ecommerce
  - Different “quality” of content
  - Long-term retention vs. short-term clicks (clickbait)
  - …
- In Quora, the value of showing a story to a user is approximated by weighted sum of actions:
  \[ v = \sum_a v_a 1[y_a = 1] \]
- Extreme application of value-aware recommendations: suggest items to **create** that have the highest value
  - Netflix: Which shows to produce or license
  - Quora: Answers and questions that are not in the service
Recommendations are rarely displayed in isolation
  ○ Rankings are combined with many other elements to make a page

Want to optimize the whole page

Jointly solving for set of items and their placement

While incorporating
  ○ Diversity, freshness, exploration
  ○ Depth and coverage of the item set
  ○ Non-recommendation elements (navigation, editorial, etc.)

Needs work hand-in-hand with the UX
Personalizing how we recommend (not just what)

- **Algorithm level**: Ideal balance of diversity, novelty, popularity, freshness, etc. may depend on the person.

- **Display level**: How you present items or explain recommendations can also be personalized.
  - Select the **best information** and presentation for a user **to quickly decide** whether or not they want an item.

- **Interaction level**: Balancing the needs of lean-back users and power users.
Example rows and beyond
9. Conclusions
Conclusions

- Recommendation is about much more than just predicting a rating
- All forms of recommendation require of a tight connection with the UI
  - Capture the right kind of feedback
    - Explicit/implicit feedback
    - Correct for presentation bias
    - ...
  - Present the recommendations correctly
    - Explanations
    - Diversity
    - Exploration/Exploitation
    - ....
Conclusions

● For the algorithm:
  ○ Use implicit feedback if possible
  ○ Build a Matrix Factorization recommender system
  ○ Think of using ensembles and turning your problem into a feature engineering problem
  ○ Always think of the metric you are optimizing to and the data you are using

● Whatever you do in the lab, you should trust your AB tests
10. References
Other resources

● 4 hour video of my lecture at MLSS at CMU (Youtube)
● “Recommender systems in industry: A netflix case study” (X. Amatriain, J. Basilico) in Recommender System Handbook
● “Mining large streams of user data for personalized recommendations” (X. Amatriain) - ACM SigKDD Explorations Newsletter
● “Big & personal: data and models behind netflix recommendations” (X. Amatriain) - ACM Workshop on Big Data
● Visit my slideshare page: https://www.slideshare.net/xamat