Recommender Systems
Learning to Rank

Professor Robin Burke

Spring 2019

Thanks to Alexandros Karatzoglou, Linas Baltrunas, Yue Shi for some materials

Learning to Rank

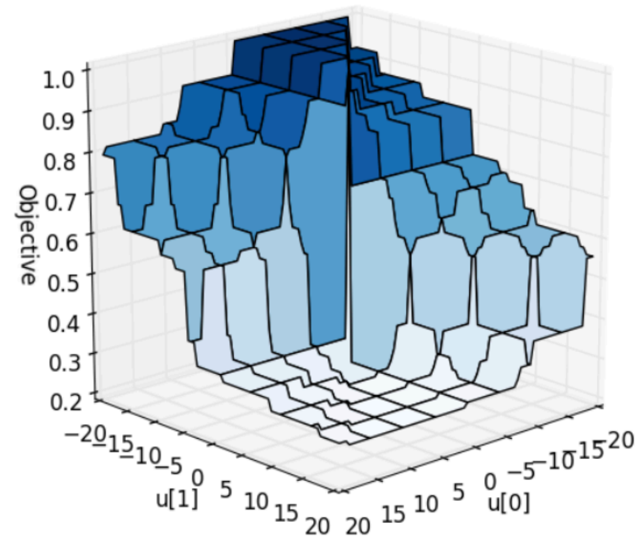
- Fundamental question for factorization models
  - Why are we optimizing $||R-\check{R}||^2$
  - If what we are really trying to do is get good lists
- Rating-oriented loss
  - Cares just as much about low ratings as high ratings
  - May correlate poorly with "good" lists
- If we measure a ranking metric (MAP or NDCG)
  - It makes sense to try to optimize such metrics
  - Can get big gains (up to 30%) by optimizing directly

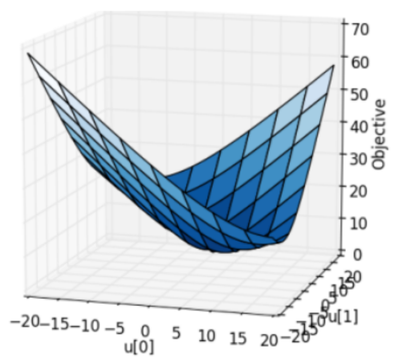
+
Example: MAP

- Suppose we want to build a loss function based on MAP
  - $MAP = \frac{\sum_{k=1}^{|S|} P(k)}{|S|}$
- But this is not a continuous function
  - When items change rank, that makes a discontinuous change in the objective
  - Discontinuity = no gradient at those points

Example: 1 latent factor dim 2

- Change the values of the factor
- Get different MAP values
- Compare with equivalent for RMSE
- Cannot apply gradient descent





Alternate formulation of MAP

- Average precision at the rank of each item
  - For all the items that are "relevant" (= rated)
- Sort all the items
  - Each relevant item has some rank
- For each item
  - Compute the precision of the smallest list containing that item
  - 8 items but only three relevant ones
  - (P@1 + P@2 + P@4) / 3
- Think about this in one pass over all items
  - Start from the bottom
  - We don't do any thing with ranks 5-8 (no relevant items)
  - When we get to item with rank 4, we count the items with lower or equal rank
  - Divide by the rank of i = precision at i's rank

Decompose the objective

- Assume unary data
  - y_{ui} = 1 if the item i is relevant to user u
  - s_{ui} = the rank of item i
  - $\mathcal{I}$ is the indicator function 1 if $s_{uj} <= s_{ui}$

- $$MAP_u = \frac{1}{\sum_{i=1}^{N} y_{ui}} \sum_{i=1}^{N} \frac{y_{ui}}{r_{ui}} \sum_{j=1}^{N} y_{uj} \mathcal{I}(s_{uj} \leq s_{ui})$$

- Discontinuous because of the indicator function and $1/s_{ui}$

Getting continuity

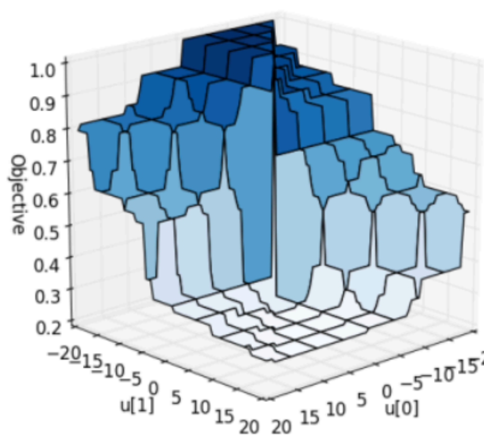
- Goal: a continuous function of the predicted rating p
- Continuous version of $1/s_{ui}$
- As the score of an item goes up
  - Its 1/rank will also go up
  - But the score has no inherent limits and the 1/rank has a minimum and maximum
- Good candidate = logistic function
  - $g_1 = 1/(1+e^{-p})$

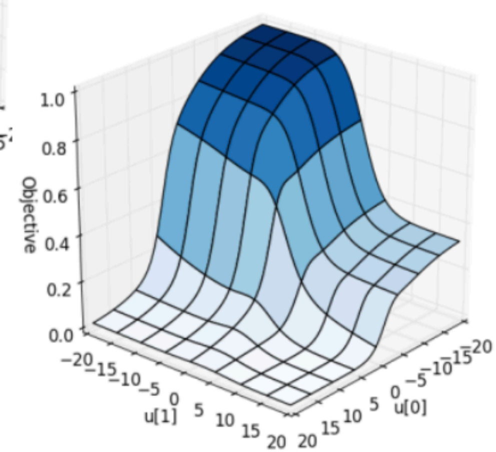
Getting continuity, part 2

- Continuous version of $I(s_{mj} <= s_{mi})$
- s_j will have lower rank than s_i
  - if s_j has a higher predicted rating p_j, than s_i (p_i)
- This is like a step function that depends on the difference between p_j and p_i
- Again logistic function is useful
  - $g_2 = 1/(1 + e^{-(pj-pi)})$

New objective

- $MAP_u = \frac{1}{\sum_{i=1}^{N} y_{ui}} \sum_{i=1}^{N} y_{ui} g_1(p(i)) \sum_{j=1}^{N} y_{uj} g_2(p(i), p(j))$
- Where p(i) is the predicted rating of item i





Caution

- Approximation is not always good
- Reason 1
  - Factorization is low-rank
  - May fail to capture important distinctions if the number of factors is too small
- Reason 2
  - Possible for the function approximations to give results not consistent with the approximated expressions
  - A > B and B > C but C > A

General strategy

- Take the ranking-oriented objective O
  - Inherently discontinuous because ranking creates discrete answer sets
- Find some continuous approximation C that is a function of the predicted rating
- Use C to create a continuous loss function over R
- Use factorization to create a prediction function that optimizes for C

Basic approaches

- Point-wise approach
  - f(user,item) ⇒ score
  - Convert ranking to regression, classification, ordinal regression, etc.
- Pair-wise approach
  - f(user, item1, item2) ⇒ score
  - Convert ranking to pair-wise classification
    - Is item1 ranked higher than item2?
- List-wise approach
  - f(user, item1, item2, …, itemn) ⇒ score
  - Direct optimization of the ranking metric over the list

Examples

- MAP optimization (originally proposed for context-aware recommendation as TFMAP)
  - List-wise approach – optimizing MAP
- BPR and RankALS
  - Pair-wise approaches
  - Different optimization criteria