We show that it is always best to use all available pairwise comparisons if possible (subsampling gives worse performance).

The loss function $\ell$ we used is $L_2$ hinge loss $\ell(a) = \max(0, 1 - a)^2$.

The set $(i,j,k) \in \Omega$: - User $i$ rates item $j >$ item $k \Rightarrow Y_{i,j,k} = 1$
- If user $i$ rates $d_2$ movies, there will be $O(d_2^2)$ pairs per user.

Time Complexity Comparison - Classical matrix factorization: $O(d_1 d_2 r)$ time, $O(d_1 d_2)$ memory
- Previous collaborative ranking: $O(d_1 d_2^2 r)$ time, $O(d_1 d_2^2)$ memory
- Our Primal-CR: $O(d_1 d_2^2 + d_1 d_2 r)$ time, $O(d_1 d_2)$ memory
- Our Primal-CR++: $O(d_1 d_2 (r + \log(d_2)))$ time, $O(d_1 d_2)$ memory

where $d_1$ is number of users, $d_2$ is averaged ratings per user, $r$ is the target rank.

**Method**

$$\nabla V f(V) = \sum_{i=1}^{d_1} \sum_{(j,k) \in \Omega_i} 2 \max \left( 0, 1 - (u_i^T v_j - u_i^T v_k) \right) (u_i e_j^T - u_i e_k^T) + \lambda V$$

- **Primal-CR**: Pre-compute $u_i^T v_j$ for all $j$; Initial $c_j = 0$ for all $j$; Update $c_j$ for each $(j, k) \in \Omega_i$; Finally compute $\nabla f(V) = \nabla f(V) + \sum_{j} c_j u_i e_j$
- **Primal-CR++**: Fix $k$, do a linear scan of $j$ after sorting; Initially $P_j = \sum_{i=1}^{d_1} s_j$; For $j = \{1, 2, \ldots, d_2\}$ Add $\{\sum_{i=1}^{d_1} P_j \cdot u_i \}$ to $g_j$; Update $P_{t_j} \leftarrow P_{t_j} - s_j$ (Can be done in $O(\log(T))$ time per query for $T$ levels of ratings using Segment tree or Fenwick tree)

**Results**

- Single Core Subsampled Data (Plots Leftmost and Center): - We subsampled each user to have exactly 200 ratings in training set and used the rest of ratings as test set, since previous approaches cannot scale up - Users with fewer than 200 + 10 ratings not included
- Single Core Full Data (Plot Rightmost): - Due to the $O(\Theta(r))$ complexity, existing algorithms always sub-sample a limited number of pairs per user - Our algorithm is the first ranking-based algorithm that can scale to full Netflix data set using a single core, and without sub-sampling
- A natural question: Does using more training data help us predict and recommend better? The answer is yes!

**Source Code**

- Julia codes: https://github.com/wuliwei9278/ml-1m
- C++ codes (2x faster than Julia codes): https://github.com/wuliwei9278/primalCR