QUIZ 2: Modern recommender systems: MAB and MF

Let's talk about matrix factorization and multi-armed bandits

Email address *

Your name *

Q1. Which of the following is a benefit of applying Multi-armed bandit (MAB) algorithms to recommendations? *

- ○ MAB models are not susceptible to overfitting the training data
- ○ They can typically be trained entirely on historical data
- ○ Prior knowledge does not need to be encoded into MAB models
- ◉ MAB models gracefully handle new items being added to the candidate set

Q2. Which of the following is NOT an advantage of dueling bandits over A/B tests? *

- ○ Dueling bandits dynamically allocate traffic towards winning ranking strategies
- ○ They allow direct comparison of rankers while controlling for more experimental variables
- ◉ A/B tests are more computationally intensive for the back-end platform
- ○ Interleaved search results mitigate against a bad user experience from poor rankers
- ○ All of them are advantages of MAB

Q3. Which of the following is a benefit of applying Multi-armed bandit (MAB) algorithms to recommendations over Matrix Factorization *

- ☑ MAB models can gracefully handle new items
- ☐ MAB can be used to discover latent features underlying the interactions between two different kinds of entities
- ☑ MAB models allow faster experimentation
- ☐ MAB provide a simple way to dimension reduction
- ☐ None of the above is

Q4. What is the basic idea behind matrix factorization recommender algorithms such as SVD *

- ◉ To transform the user by item ratings space into user by feature preference and item by feature expression for some set of latent features
- ○ To help take into account a user's context, such as user's emotional state or current mood, location and time and date
- ○ To effectively distinguish between desirable and undesirable items
- ○ I have no idea

Q5. What is folding-in in a singular value decomposition? *

- ○ Mapping values outside the normal range to minimum or maximum values
- ◉ Incorporating new ratings, users, or items into the model by using the existing factorization and updating/computing a feature vector from the ratings
- ○ Factoring the ratings matrix into two matrices multiplied by a matrix of eigenvalues
- ○ Directly predicting a rating by taking the dot product of a user-vector and an item-vector
- ○ I have no idea

Q6. There is no reason SVD based approaches to recommendation are guaranteed to reduce dimensionality, but in the end, they almost always do substantially. Mathematically this is because *

- ◉ Most of the singular values are almost always very close to zero, indication that not much is lost by reducing those dimensions
- ○ Incorporating new ratings, users, or items into the model by using the existing factorization and updating/computing a feature vector from the ratings
- ○ Because you need different algorithms to compute predictions v. top-N recommendation
- ○ The evaluation can only judge whether the returned items are among the rated/consumed/purchased ones. having too many other items just increases the number of desirable but not-yet consumed items, making it harder to tell whether the recommendations are good

Q7. In addition to providing good recommendations, matrix factorization techniques can improve privacy and robustness over nearest neighbor techniques. why? *

- ○ They're designed to measure how effectively a recommender can be used to distinguish between desirable and undesirable items
- ○ When there are many more users than items in the system, user similarities can change rapidly as users interact with the system
- ◉ The relationship between the input and output is less direct, going through latent features, and is harder to directly manipulate
- ○ If the recommender picks something the user didn't purchase we don't know if they like it (bad rec) or didn't know about it (potentially great rec)

Q8. The basic idea behind matrix factorization recommender algorithms such as SVD is that they're designed to measure how effectively a recommender can be used to distinguish between desirable and undesirable items *

- ○ True
- ◉ False

Q9. Which of these is NOT an advantage of SVD-based algorithms compared with user-user collaborative filtering? *

- ○ Faster time for building the model
- ○ Faster time for computing individual recommendations.
- ○ Potential to avoid overfitting and resulting odd recommenations.
- ◉ Potential to get recommendations based on users with similar tastes who don't have items in common
- ○ I have no idea

Q10. What is the core idea behind dimensionality reduction recommenders *

- ○ To reduce the computation from polynomial to linear
- ○ To strip off any product attributes so products appear simpler
- ○ To reduce the computation time from O(n^3) to O(n^2).
- ◉ To transform of high-dimensional data into a meaningful representation of reduced dimensionality

Q11. Recommendation with Matrix Factorization (mark true statements) *

- ☑ In low-rank MF, the scaling factor is absorbed into both matrices, representing users and items
- ☑ A rating r_{ui} can be estimated by the dot product of user vector p_u and item vector q_i
- ☐ A rating r_{ui} can be estimated as the cosine similarity of the user vector p_u and item vector q_i

Q12. Asymmetric matrix factorization -AMF (mark true statements) *

- ☑ if there are many more user-vectors than item-vectors, these can be the slow/inaccurate part to learn in basic matrix factorization
- ☑ Instead of representing a user as their own Vector, the basic idea in AMF is to learn a separate set of item vectors that, when summed, give the user vector
- ☐ Asymmetric matrix factorization is viewed as a simple nearest-neighbor method
- ☐ None of them is true

Q13. Which statements are true for SVD? *

- [ ] It does not perform quite well for most data sets
- [x] Results are not always the best for visualization
- [x] It is difficult to interpret
- [x] De-facto method for dimensionality reduction in generic datasets

Q14. The left singular vectors obtained from SVD are also: *

- ( ) The transpose of the right singular vectors
- ( ) Orthonormal
- ( ) The eigenvectors of RR(T)
- (•) B and C
- ( ) A and B

Q15. To get a low-rank approximation of R of rank K using SVD, you: *

- ( ) Randomly select K rows and columns of R
- (•) Take the K largest singular values along with their associated left and right singular vectors
- ( ) Take the K smallest singular values along with their associated left and right singular vectors
- ( ) Transpose of the singular values matrix

Q16. Unlike SVD, Matrix Factorization relaxes: *

- ○ The two matrices, P and Q, are no longer required to be orthonormal
- ○ The singular values aren't decomposed, instead they are absorbed into P and Q
- ○ Bias terms can be added to the model
- ◉ All of above

Q17. What are the standard ways to train Matrix Factorization? *

- ○ Alternating Least Squares
- ○ Stochastic Gradient Descent
- ○ Variance reduction
- ◉ B and C
- ○ A and B

Q18. Asymmetric Matrix Factorization replaces the user vector with: *

- ○ The square root of the user vector
- ◉ The sum of a separately learnt set of item vectors, for items the user has purchased
- ○ The average cosine distance between all item vectors
- ○ A random vector