Gmail **Saby S <ssahoo.infinity@gmail.com>**

Queries related to your NIPS 2017 paper "Thy Friend is My Friend"

Christina Lee <celee@mit.edu> Sun, Jan 21, 2018 at 10:21 AM
To: Saby S <ssahoo.infinity@gmail.com>
Cc: Jennifer Chayes <jchayes@microsoft.com>, Christian Borgs <Christian.Borgs@microsoft.com>, Devavrat Shah <devavrat@mit.edu>

Hello Sabyasachi,

Thanks for reaching out. I've written our responses below in blue. Feel free to ask us as any further questions arise.

- In the "Model" section on page 5, you talk about *Mu,v* and (cap)*Fu,v*. So, the only difference between these two values is that *Mu,v* represents the already present ratings(or edges) in the dataset and (cap)*Fu,v* represents the estimated ratings obtained from collaborative filtering. Is my understanding correct here?
  Yes, M_{uv} represents the ratings present in the dataset, and \hat{F}_{uv} represents the estimated ratings from collaborative filtering such as in the equation (2) (5) and (6).

- In the "Algorithm Details" section on Page 6, you have step 1 of sample splitting, where you split all the edges of the graph *E* into three sub graphs, *E1*, *E2* and *E3*, such that any edge from E, ends up in either *E1*, *E2* or *E3* with probabilities *c1*, *c2* and *1 - c1 - c2*. I don't understand the intuition behind this step. I can see the values *c1*, *c2* being used in theorems, but can you tell me why are we splitting an already *sparse* graph into even more *sparser* sub graphs? What advantage do we gain? Or am I misunderstanding something?
  The purpose of sample splitting was to reduce dependence/correlation in the datapoints used across different steps of the algorithm (important for the analysis and theoretical results). Without splitting, it could introduce bias.

- In step 3 on page 6, (cap) *Nu,r* is the normalized neighborhood boundary. My understanding was that it is a scalar value but looking at the formula for calculating *dist1*, it looks like it maybe a vector/matrix. Then, is it a matrix of normalized ratings (or product of values along all the edges of path) from user(or vertex) *u* to movie (or vertex) *i*? Is my understanding correct or is *Nu,r* something else?
  N_{ur} is an n-dimensional vector, where the i-th entry is indicated by the equation specified in step 2. We are using the notation for a symmetric n-by-b matrix, so in the example of a rectangular user-movie rating matrix, we would consider the larger symmetric (n+m)-by-(n+m) matrix, and N_{ur} would be a (n+m)-dimensional vector. The sparsity would alternate with respect to the radius r between users and movies, since it represents the product of weights along paths of distance exactly r from vertex u.

- Also, in step 3, you talk about two ways of calculating distances. Can you give little more insight or some intuitive explanation on what they are and how to calculate them from *Nu,r* matrix?
  Think of the first variant as computing an inner product between vectors (N_{ur} - N_{vr}) and (N_{u,r+1} - N_{v,r+1}) with respect to the matrix M_2. In our analysis, we show that this quantity approximates ||\Lambda^{r+1} Q (e_u-e_v)||_2^2.
  The second variant involves solving a linear system of equations for a d'-dimensional vector z, where d' is the number of distinct eigenvalues in \Lambda. An implementation of this would require estimates of the spectrum. However, if you were able to solve for z, then this variant would be able to approximate ||\Lambda Q (e_u-e_v)||_2^2 (from our analysis). The computation is a linear combination of the "inner product"-like quantities used in variant 1 but with different radius lengths.
  The disadvantage of variant 2 is the extra computation required and the need to know the spectrum/eigenvalues. The advantage is that the quantity it approximates, specifically ||\Lambda Q (e_u-e_v)||_2^2, is directly related to the difference in functional values of interest. The first variant approximates ||\Lambda^{r+1} Q (e_u-e_v)||_2^2, which is a skewed measurement, especially when there is large variation between the eigenvalue magnitudes.

- In the "Practical Implementation" section on Page 9, you say "The key computational step of our algorithm involves comparing the expanded local neighborhoods of pairs of vertices to find the “nearest neighbors”." I just wanted to verify, if this meant the step3 of your algorithm (on page 6) or some other step of your algorithm?
  Yes, we mean step 3.

- In the "Non-uniform sampling" section on Page 9, you have proposed the following method to account for non uniform sampling, "If we let matrix X indicate the presence of an observation or not, then we can apply our algorithm twice, first on matrix X to estimate function g, and then on data matrix M to estimate f times g." What exactly is matrix *X*? Is it a matrix which has value 1 if an edge is present between the vertices or not (i.e. user

has rated a movie or not)?
Sorry about the confusion, I realized that I left out many important details. Our method requires being able to distinguish from an observed zero and a blank unobserved entry in the data matrix (as the observed entries are used to build the neighborhoods), so we are only able to partially deal with non-uniform sampling. Specifically, suppose you had a setting such as a recommendation system which recommends movies at random with probability p (thus satisfying uniform sampling), however users decide to provide a rating or not with probability g(.,.) which is a function of the user and movie latent features. Then the rating matrix will be non-uniformly sampled due to users having varied response rates on different movies, however you would still have a uniformly sampled matrix of movies that were recommended to users. In this setting, X would be a matrix where 1 entry indicates that a movie was recommended and a rating was provided, 0 entry indicates that a movie was recommended and a rating was NOT provided, and blank indicates that the movie was not recommended. The algorithm could be applied to this matrix X to estimate the probability that a user provides a rating for a given movie (the function g). The rating matrix M contains the information for provided ratings, 0 entries for recommended movies not rated, and blank entries indicating the movie was not recommended. The algorithm applied to this matrix M would estimate the expected rating a user has for a movie scaled by the probability a user provided a rating given the movie was recommended. Therefore the user-movie rating would be estimated by dividing by the probability from function g(.,.). In this sense we are only partially able to deal with non-uniform sampling since it still requires a uniform probability p of the movie being recommended. The general case is still an open question. Sorry if this is a long winded explanation, but I'd be happy to follow up with more discussion if you have further questions.

Sincerely,
Christina Lee
[Quoted text hidden]