Frank-Wolfe Initializations for Recommender Systems

Joseph Janssen: 28738152; Vincent Guan: 18533281

December 21st, 2020

1 Introduction

Recommender systems aggregate user feedback of items, then provide recommendations to other users. Popular examples of recommender systems include Netflix's "Top Picks" tab, Amazon's "Products related to this item" tab, and Facebook's story ordering algorithm. Recommender system datasets can be massive. For example, Netflix has almost 200 million customers and over 5000 movies and TV shows.

To reduce computational and memory requirements, a natural question one may ask is if users can be summarized by a series of "archetypes". Instead of solving the entire problem exactly, we can approximate the system with a linear combination of k rank one archetypes where k is typically much smaller than the number of users n. Archetypal analysis therefore drastically reduces the problem's dimensionality, and has been used to cluster video game player behavior (Drachen et al., 2012), generate human-like bot game play (Sifa & Bauckhage, 2013), and recommend games (Sifa et al., 2014).

Although archetypal analysis with Frank-Wolfe works with any arbitrary initialization in the problem domain such as the zero or random matrices (Bauckhage, 2020; Pokutta et al., 2020; Mu et al., 2016), Bauckhage (2020) hypothesized that convergence and quality of results can be improved by better initializations. The goal of this paper is to explore how different initializations of Frank-Wolfe can affect the interpretation, convergence, and quality of results for a matrix completion recommender system.

2 Data

The Netflix-prize data was obtained from the Kaggle competition website (https://www.kaggle.com/netflix-inc/netflix-prize-data). We only used ratings that were made in the year 2000. Information on movie genres from IMDB was compiled and downloaded from github (https://github.com/bmxitalia/netflix-prize-with-genres). There are 882862 ratings with 3510 unique movies, 8203 users, and 23 genres. A movie can have multiple genres.

3 Methods

Simply put, Netflix's recommender system problem is a problem of low-rank matrix completion (Sindhwani et al., 2010; Cabral et al., 2013). Since constraining the nuclear norm can induce low rank solutions (Recht et al., 2010), our problem can be formulated as follows:

$$\min_X \frac{1}{2}||\Omega \circ X - B||_F^2 \text{ s.t. } ||X||_* \leq \tau. \tag{1}$$

Let there be n users (n=8203) who have access to m movies (m=3510) in the database. $B \in \mathbb{R}^{n \times m}$ is the matrix of user ratings where the entry $B_{i,j}$ is the rating that user i gave movie j. In the Netflix setting, these ratings are the integers $\{1, 2, 3, 4, 5\}$ and $B_{i,j} = 0$ if the user has not rated the movie. B will be a sparse matrix since, on average, users have only rated about 100 of the 3510 movies. We wish to recover X, the dense matrix of predicted user ratings. Ω is the binary mask with an entry of 1 when user i has rated product j and 0 otherwise. $\Omega \circ X$ can be viewed as the entrywise matrix product. Thus, equation (1) asks which low-rank matrix of predicted user ratings X best aligns with the partial data B.

The squared Frobenius norm in (1) is convex and continuously differentiable. Moreover, the nuclear norm ball $\{X \in ||X||_* \leq \tau\}$ is compact and convex. Thus, the Frank-Wolfe (conditional gradient) algorithm is an appropriate method to solve problem (1) (Jaggi, 2013). We choose it for its simplicity as well as its efficiency, since it scales well to large dimensions, which is important for handling the Netflix dataset. For instance, the projected gradient method scales poorly since the cost of the nuclear norm projection is $O(n^3)$ for a $n \times n$ matrix. This is shown in Figure 1, which contrasts the running-times of the conditional gradient method and the projected gradient method.

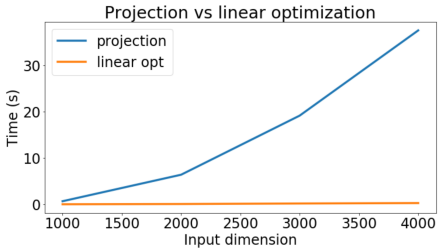

Figure 1: This figure is taken from https://ee227c.github.io/code/lecture5.html, which compares the computational costs of one iteration of projected gradient descent (blue) with one iteration of conditional gradient descent (orange) for different sized nuclear norm constrained matrix completion problems.

Given a problem of the form (2) with a convex continuously differentiable function, f, and a compact convex set D, the Frank-Wolfe algorithm can be expressed as Algorithm 1 (Jaggi, 2013).

$$\min_{X \in D} f(x) \tag{2}$$

Algorithm 1: General Frank-Wolfe with approximate linear sub-problems and line-search

Result: $x^{(K)}$
Choose $x^{(0)}$ such that $x^{(0)} \in D$;
for $k = 0...K$ **do**
  Find $s \in D$ s.t. $\langle s, \nabla f(x^{(k)}) \rangle = \min_{s* \in D} \langle s*, \nabla f(x^{(k)}) \rangle$;
  $\gamma = \frac{2}{k+2}$ or arg $\min_{\gamma \in [0,1]} f(x^k + \gamma(s - x^{(k)}))$;
  $x^{(k+1)} = (1-\gamma)x^{(k)} + \gamma s$;
end

For the Netflix problem, $f(X) = \frac{1}{2}||\Omega \circ X - B||_F^2$ and $D = \{X : ||X||_* \leq \tau\}$. The gradient of f at X_k, the minimizer, s, of $\langle s, \nabla f(X^k) \rangle$, and the optimal step size, γ, are derived in the Appendix. Solving for the top singular value and its associated singular vectors can be done efficiently via the power method (Algorithm 2). The Frank-Wolfe algorithm we use to solve problem (1) is given by Algorithm 3. Based on an exploratory analysis, we let $\tau = 50000$ and $K = 100$ when implementing the algorithm on the Netflix dataset (4.2).

Algorithm 2: Power Method

Result: u, v, σ
Initialize v_0 with a random vector of unit norm and mean 0;
for $k = 1...1000$ **do**
  $v = A^T A v_0$;
  $v = \frac{v}{||v||_2}$;
  if $(||v - v_0||_2 < \epsilon)$ break;
  $v_0 = v$;
end
$\sigma = ||Av||_2$;
$u = Av/\sigma$;
return u, v, σ

Algorithm 3: Frank-Wolfe for Netflix Problem

Result: X^{K+1}
Let $||X^0||_* \leq \tau$;
for $k = 0...K$ **do**
  $-\nabla f(X^k) = B - \Omega \circ X^k$;
  $[u, v, \sigma]$=power$(-\nabla f(X^k), \epsilon)$;
  $s = uv^T$;
  $U_{k+1} = \begin{bmatrix} U_k & u \end{bmatrix}$;
  $V_{k+1} = \begin{bmatrix} V_k & v \end{bmatrix}$;
  $\gamma = \frac{\sum_{i,j}(\Omega_{i,j} X^k_{i,j} - B_{i,j})(\Omega_{i,j} X^k_{i,j} - s_{i,j})}{\sum_{i,j}(s_{i,j} - \Omega_{i,j} X^k_{i,j})^2}$;
  if $\gamma < 0, \gamma = 0$;
  if $\gamma > 1, \gamma = 1$;
  $\Sigma_{k+1} = \begin{bmatrix} (1-\gamma)\Sigma_k & 0 \\ 0 & \gamma\tau \end{bmatrix}$;
  $X^{k+1} = U_{k+1} \Sigma_{k+1} V_{k+1}^T$;
end

Iterative methods for norm constrained problems often initialize at 0, however, there may be more reasonable options. For recommender systems, it would be reasonable to set X^0 as the constant matrix where each entry is equal to the mean rating in the dataset. However, the mean rating for some movies, ex. *The Matrix*, will be higher than for others, ex. *Star Wars: The Phantom Menace*, and some users may on average rate movies lower than other users. To account for this in our initial guess, we can set a unit vector u proportional to the mean rating given by each user, set a unit vector v proportional to the mean rating given to each movie, then set $X^0 = tuv^T$ where t is chosen such that the mean entry of X^0 is equal to the mean rating in the dataset. We call this initialization the "col/row mean rating". We test the sensitivity of results to these three different initializations in 4.2.

4 Results and Testing

4.1 Synthetic Testing

To optimize our algorithm's performance on the Netflix data, we first tested it on synthetic data. The main advantage of synthetic testing is that we can directly compare our approximations to the true full matrix X, which is obviously unknown for the real data.

To analyze convergence to the solution, we generate a full matrix of ratings X, conceal it with a binary mask Ω such that $B = \Omega \circ X$, and then compute $\frac{1}{2}||X - X^k||_F^2$ at each iteration k of Frank-Wolfe. Note that this is different from convergence in the objective function, which is given by $\frac{1}{2}||\Omega \circ X^k - B||_F^2$.

To simulate Netflix rating matrices, we generate a random normally distributed $n \times m$ matrix X of ratings with integer values from 1 to 5. Then, we generated a random filter Ω of 1s and 0s with an indicated density of non-zero entries, corresponding to the sparsity of the initial data, and set $B = \Omega \circ X$. Frank-Wolfe was then run with a specified number of iterations and value τ with the goal of minimizing $\frac{1}{2}||\Omega \circ X - B||_F^2$ on the τ-nuclear norm ball.

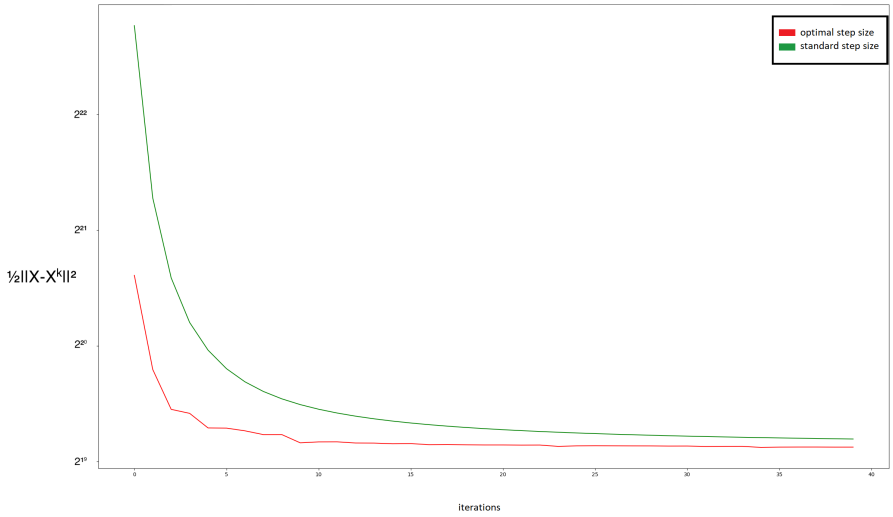

Figure 2: The convergence plot comparing the optimal step size (red) with the standard step size ($\frac{2}{2+k}$) (green) with $n, m = 500, 1000$, a density of 0.15, 40 iterations, and $\tau = 2000$.

The convergence is extremely rapid, with most of the convergence done in the first 10 iterations. The plots illustrate that using the optimal step size enables larger improvements at each iteration whereas the standard step size converges more smoothly and gradually. Another way of framing this is that the algorithm works by matching the singular vectors of X, and the optimal step size enables more efficient matching whereas other step sizes will chip away at the same singular vector for more iterations.

However, both variants perform well. In this example, $\frac{1}{2}||X^{40} - X||_F^2 \approx 55000$ when the step size is optimal and $\frac{1}{2}||X^{40} - X||_F^2 \approx 56000$ when the step size at iteration k is $\frac{2}{k+2}$. Also, $\frac{1}{2}||X||_F^2 \approx 27500000$, so the squared error is about one fiftieth of the squared norm of X, which seems to be a reasonable error for completing a 500×1000 matrix when only given 15% of the entries.

4.2 Testing on Netflix Data

We then tested our algorithm on the Netflix data using the three aforementioned initializations of X^0: 1) 0 matrix, 2) constant average rating matrix, 3) col/row mean matrix. To contrast their performances, we stored the singular vectors u, v corresponding to the Frank-Wolfe step at each iteration k and then plotted the correlation matrices (Figures 3 & 4). Naturally, if the left singular vectors u taken at different iterations of the algorithm are highly correlated, then the convergence is less efficient since the algorithm takes multiple steps in the same general direction and hence, costly matrix compression may be needed. However, if these vectors are uncorrelated, then the convergence is efficient. The same analysis holds for interpreting the correlation of the right singular vectors v. For each of the three initializations, we present the correlation matrices of u and v below. We see that the col/row mean initialization is best since the vectors are the least correlated, and the 0 initialization is the worst due to high correlation.

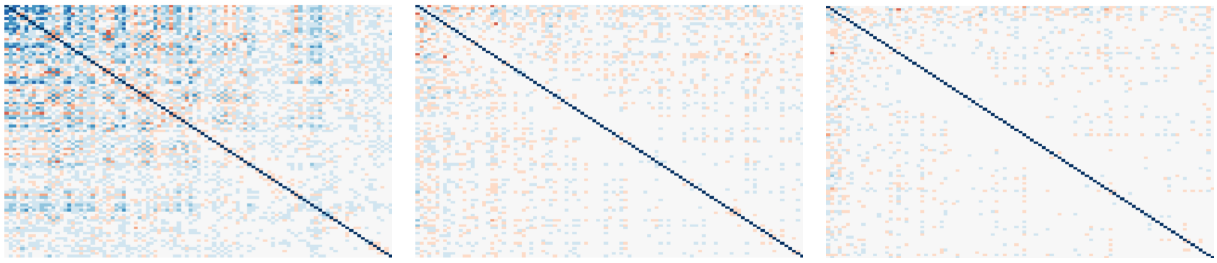

Figure 3: The correlation matrices of U after 100 iterations for different initializations. From left to right, the initializations are 0, mean rating, and col/row mean rating. Dark blue indicates a correlation of 1, dark red indicates a correlation of -1, and grey indicates a correlation of around 0.

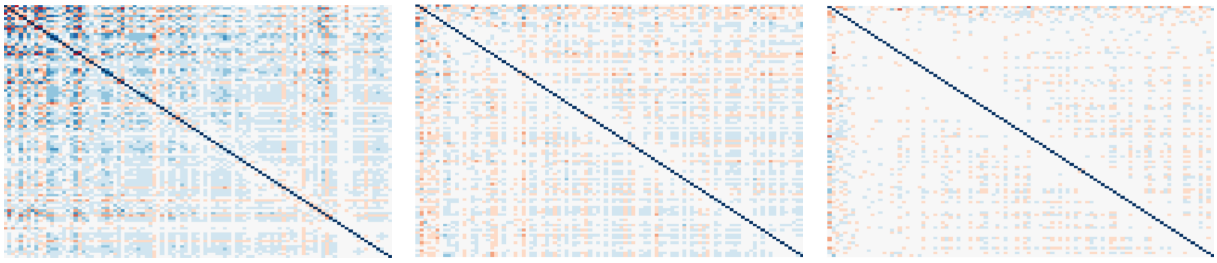

Figure 4: The correlation matrices of V after 100 iterations for different initializations. From left to right, the initializations are 0, mean rating, and col/row mean rating. Dark blue indicates a correlation of 1, dark red indicates a correlation of -1, and grey indicates a correlation of around 0.

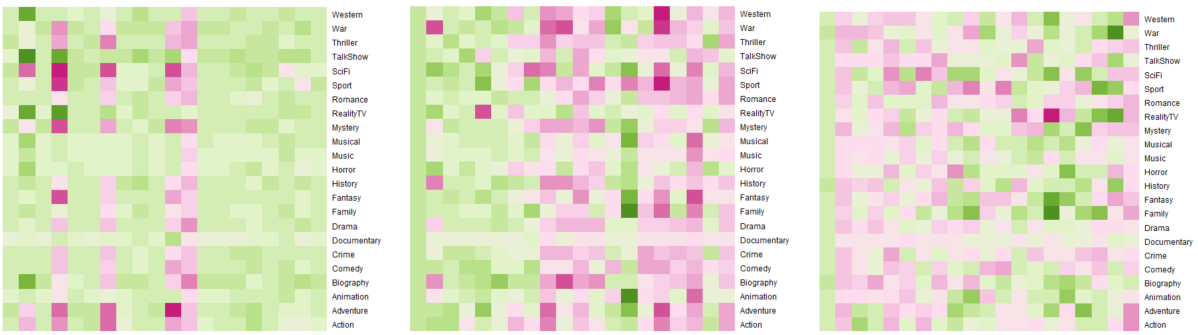

Figure 5: The mean entry in V aggregated by movie genre. The columns in each plot correspond to the columns of V (first 20 columns in total). The row of each plot corresponds to a movie genre. From left to right, the initializations are 0, mean rating, and col/row mean rating. Green indicates a positive average and pink indicates a negative average.

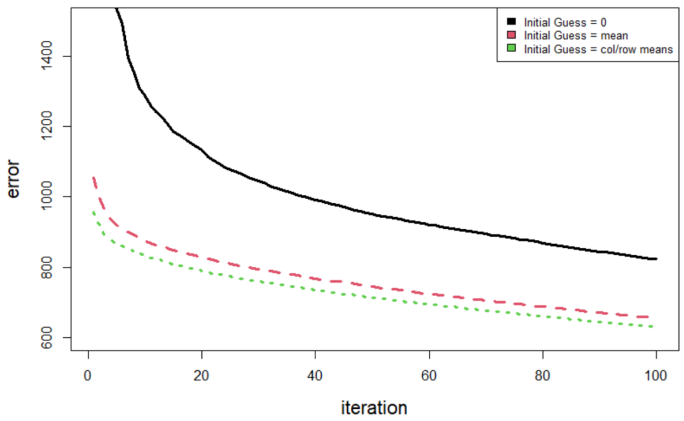

Figure 6: The Frobenius norm error of $\Omega \circ X^k - B$ at iteration k. The initializations are 0 (black), mean rating (red), and col/row mean rating (green).

Similarly, we interpret each step with regards to different film genres to see if we can identify archetypes (Figure 5). If some genres in a column show positive entries (green) and others are negative (red), then this column naturally corresponds with an interpretable archetype, whereas if an entire column is the same color, the interpretation is more ambiguous. For example, the last 8 columns of V do not contrast the genres from one another in Figure 5a, however, the second column contrasts ScFi, mystery, adventure, and action with the rest of the genres. This ability to contrast different genres for interpretation only happens for about 5 out of 20 of the columns with a 0 initialized algorithm. With a mean rating initialized algorithm 14 out 20 columns are interpretable. With a col/row mean rating initialized algorithm 18 out 20 columns are interpretable.

Converging on a solution happens faster when good initial guesses are chosen for Frank-Wolfe (Figure 6). In fact, it takes about 50 iterations of Frank-Wolfe with the zero-initialized algorithm to find an X with the same error as the first iteration of Frank-Wolfe with the col/row mean rating initialization. Further, within 10 iterations of the col/row mean initialized algorithm the error is less than the error after the 100^{th} iteration of the 0 initialized algorithm.

5 Discussion and Conclusions

As was suspected by Bauckhage (2020), choosing better initialization matrices can lead to independent iterations without matrix compression, easily interpretable results, and faster convergence. In particular, we saw that the 0 matrix performs far worse across each of these metrics than the other two initializations that we considered. Meanwhile, the difference between the constant mean rating matrix and the mean col/row matrix was more subtle but still noticeable. Predictably, the more information that the Frank-Wolfe algorithm is given at the start, the better the matrix completion results will be. This is similar to the fact that the algorithm naturally completes dense matrices better than sparse matrices. Since sparse matrices are the principal interest of matrix completion problems, we instead try to embed as much information as possible into the initialization matrix to compensate for the lack of information in the dataset. For example, when we were considering the 0 initialization earlier in the project, a major difficulty that we encountered was that movies with few ratings were predicted to have extremely low ratings due to the fact that these vectors of ratings are highly correlated with the standard basis vectors, so they are more difficult columns to complete. This problem is called "incoherence" (Candes & Recht, 2009; Candes & Tao, 2010; Recht, 2011). When we consider more appropriate initializations such as the mean col/row matrix, we no longer have near 0 ratings for movies with very few reviews, though the accuracy of these predicted ratings is almost certainly still worse than the predicted ratings of movies with more reviews. Although initialization cannot completely compensate for incoherence and general sparsity, we have shown that it is a powerful contributor to the relevant performance metrics of large matrix completion problems. We therefore believe that more complex initializations that embed more archetypal information should be studied for better results in matrix completion via Frank-Wolfe.

6 Appendix

6.1 Derivation of the gradient for the LMO in Frank-Wolfe

Let $f : \mathbb{R}^{n\times m} \to \mathbb{R}$ be given by $f(X) = \frac{1}{2}||\Omega \circ X - B||_F^2$. Recall that the gradient of a real valued function f on the matrix space $\mathbb{R}^{n\times m}$ is given by the matrix:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial X_{1,1}} & \cdots & \frac{\partial f}{\partial X_{1,m}} \\ \vdots & \ddots & \\ \frac{\partial f}{\partial X_{n,1}} & \cdots & \frac{\partial f}{\partial X_{n,m}} \end{bmatrix} \tag{3}$$

In our case, we have $f(X) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(\Omega_{i,j}X_{i,j} - B_{i,j})^2$. Differentiating in each entry gives us

$$\nabla f = \begin{bmatrix} \Omega_{1,1}X_{1,1} - B_{1,1} & \dots & \Omega_{1,m}X_{1,m} - B_{1,m} \\ \vdots & \ddots & \\ \Omega_{n,1}X_{n,1} - B_{n,1} & \dots & \Omega_{n,m}X_{n,m} - B_{n,m} \end{bmatrix} = \Omega \circ X - B \tag{4}$$

6.2 Nuclear norm ball as convex hull of rank one matrices

Let $A = \{uv^T : u \in \mathbb{R}^m, v \in \mathbb{R}^n, ||u||_2 = ||v||_2 = 1\}$ and $B_* = \{X \in \mathbb{R}^{m\times n} : ||X||_* \leq 1\}$. We show $\text{conv}(A) = B_*$. Recall $||X||_* = \text{tr}(\sqrt{X^TX}) = \sum_i \sigma_i(X)$.

First, we show $\text{conv}(A) \subseteq B_*$. Recall that all norm balls are convex, hence B_* is a convex set. It then suffices to prove that $A \subseteq B_*$. Let $X = uv^T \in A$. Then $X^TX = (vu^T)(uv^T) = v(u^Tu)v^T = vv^T$ since $u^Tu = ||u||_2^2 = 1$. In particular, we know $\sigma_i^2(X) = \lambda_i(X^TX) = \lambda_i(vv^T)$. We can use v itself as the eigenvectors, getting $vv^Tv = \lambda v$ implying $\lambda = 1$ since $v^Tv = 1$. Thus, $\sum_i \sigma_i(X) = 1 = ||X||_*$, implying $X \in B_*$, $A \subseteq B_*$ as desired.

Next, to show $B_* \subseteq \text{conv}(A)$, let $X = U\Sigma V^T \in B_*$. Suppose X is of rank l. Then, using the outer product form of SVD, we may write $X = \sum_{k=1}^{l} u_k\theta_k v_k^T$ where $\theta_k \geq 0$ (since singular values are nonnegative) is the k^{th} singular value of X and u_k and v_k are normal left and right singular vectors.

Since $||X||_* \leq 1$, we know that $\sum_{k=1}^{l} \theta_k \leq 1$. Since $0 = \frac{1}{2}uv^T + \frac{1}{2}(-uv^T) \in \text{conv}(A)$, we may trivially write X as a convex combination of elements $u_kv_k^T \in A$ and 0: $X = \sum_{k=1}^{l} \theta_k u_k v_k^T + (1 - \sum_{k=1}^{l} \theta_k)0$. Thus, $B_* \subseteq \text{conv}(A)$ and $\text{conv}(A) = B_*$ as desired. □

As a direct corollary, $B_{\tau_*} = \{X \in \mathbb{R}^{m\times n} : ||X||_* \leq \tau\} = \text{conv}(\tau A)$.

6.3 Derivation of Frank-Wolfe atom for problem (1)

From Jaggi, (2013) the atom needed in each iteration of Frank-Wolfe solves the optimization problem given by:

$$s = \arg\min_{||s||_* \leq \tau} < \nabla f(X^k), s >= \arg\max_{||s||_* \leq \tau} < -\nabla f(X^k), s > \tag{5}$$

Recall that maximizing over the convex hull is equivalent to maximizing over the original set. Thus, we may use 6.1 and 6.3 to rewrite our maximization problem as:

$$s = \arg\max_{\tau uv^T : ||u||_2, ||v||_2 = 1} < B - \Omega \circ X^k, \tau uv^T >:= \arg\max_{\tau uv^T : ||u||_2, ||v||_2 = 1} < A, \tau uv^T > \tag{6}$$

We now apply the trace inner product on real matrices, $< A, B >= \text{tr}(A^TB)$:

$$< A, \tau uv^T >= \text{tr}(A^T\tau uv^T) = \tau\text{tr}(A^Tuv^T) = \tau\text{tr}(vu^TA) = \tau\text{tr}(u^TAv) \tag{7}$$

We have used the properties: 1) $\text{tr}(cA) = c\text{tr}(A)$, 2) $\text{tr}(A^T) = \text{tr}(A)$, and 3) $\text{tr}(ABC) = \text{tr}(CAB) = \text{tr}(BCA)$. Now, the expression inside the trace is just a number, so we may remove the trace:

$$s = \arg\max_{\tau uv^T : ||u||_2, ||v||_2 = 1} \tau u^TAv \tag{8}$$

Let $A = U\Sigma V^T$ be its singular value decomposition. Then, by associativity of matrix multiplication:

$$s = \tau \arg\max_{\tau uv^T : ||u||_2, ||v||_2 = 1} u^T(U\Sigma V^T)v = \tau \arg\max_{\tau uv^T : ||u||_2, ||v||_2 = 1} (u^TU)\Sigma(V^Tv) \tag{9}$$

Now, note that since U and V^T are orthogonal normal matrices, u^TU and V^Tv are unit vectors. Next, we observe that Σ is a square diagonal matrix with its largest eigenvalue being σ_1, the largest singular value of A (always positive). Recall the Rayleigh quotient characterization of the largest eigenvalue σ_1 of a square Hermitian matrix such as Σ is given by:

$$\sigma_1 = \max_{||x||=1} < x, Ax > \tag{10}$$

Finally, since Σ is a diagonal matrix with its first entry being its largest eigenvalue σ_1, we see that arg max of (16) is given by $x = (1, 0, ..., 0)$. Equivalently, we see that for the argmax, u is parallel to the first row of U and v is parallel to the first row of V since we want $u^TU = (1, 0, ...0)$ and $V^Tv = (1, 0, ...0)^T$ (The fact that the other entries are 0 is due to the fact that the rows of U and V are normal). In other words, $s \in \tau A$ is the outer product of the top singular (unit) vectors of A, scaled by τ.

$$s = \tau uv^T \tag{11}$$

6.4 Derivation of the exact line-search step size

From Jaggi, (2013) line search requires us to solve the optimization problem given by:

$$\arg\min_{\gamma \in [0,1]} f(X^k + \gamma(s - X^k)) = \arg\min_{\gamma \in [0,1]} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(\Omega_{i,j}(X_{i,j}^k - \gamma X_{i,j}^k + \gamma s_{i,j}) - B_{i,j})^2 \tag{12}$$

We then solve $\frac{d}{d\gamma} f(X^k + \gamma(s - X^k)) = 0$ to find critical points:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(\Omega_{i,j}(X_{i,j}^k-\gamma X_{i,j}^k+\gamma s_{i,j})-B_{i,j})\Omega_{i,j}(s_{i,j}-X_{i,j}^k)=\sum_{i,j\in P}(X_{i,j}^k-\gamma X_{i,j}^k+\gamma s_{i,j}-B_{i,j})(s_{i,j}-X_{i,j}^k)=0 \tag{13}$$

$i,j \in P$ indicates the non-zero entries of Ω. Separating the terms with γ leaves:

$$\sum_{i,j\in P}(X_{i,j}^k-B_{i,j})(s_{i,j}-X_{i,j}^k)+\sum_{i,j\in P}(\gamma s_{i,j}-\gamma X_{i,j}^k)(s_{i,j}-X_{i,j}^k)=0 \tag{14}$$

Thus, we determine a single critical point γ_0:

$$\gamma_0=\frac{\sum_{i,j\in P}(X_{i,j}^k-B_{i,j})(X_{i,j}^k-s_{i,j})}{\sum_{i,j\in P}(s_{i,j}-X_{i,j}^k)^2}. \tag{15}$$

In fact, γ_0 always corresponds to the global minimum since the function f is a concave up quadratic. Thus, as long as $0 \leq \gamma_0 \leq 1$, we choose γ_0 as our step size. Otherwise, we take $\gamma = 0$ or 1 depending on which one corresponds to a lesser value.

7 References

Drachen, A., Sifa, R., Bauckhage, C., & Thurau, C. (2012, September). Guns, swords and data: Clustering of player behavior in computer games in the wild. In 2012 IEEE conference on Computational Intelligence and Games (CIG) (pp. 163-170). IEEE.

Sifa, R., & Bauckhage, C. (2013, August). Archetypical motion: Supervised game behavior learning with archetypal analysis. In 2013 IEEE Conference on Computational Inteligence in Games (CIG) (pp. 1-8). IEEE.

Sifa, R., Bauckhage, C., & Drachen, A. (2014, January). Archetypal Game Recommender Systems. In LWA (pp. 45-56).

Bauckhage, C. (2020). NumPy/SciPy Recipes for Data Science: Archetypal Analysis via Frank-Wolfe Optimization. researchgate. net, Oct.

Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. Naval research logistics quarterly, 3(1-2), 95-110.

Sifa, R., Yawar, R., Ramamurthy, R., Bauckhage, C., & Kersting, K. (2020). Matrix-and Tensor Factorization for Game Content Recommendation. KI-Künstliche Intelligenz, 34(1), 57-67.

Sindhwani, V., Bucak, S. S., Hu, J., & Mojsilovic, A. (2010, December). One-class matrix completion with low-density factorizations. In 2010 IEEE International Conference on Data Mining (pp. 1055-1060). IEEE.

Cabral, R., De la Torre, F., Costeira, J. P., & Bernardino, A. (2013). Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2488-2495).

Recht, B., Fazel, M., & Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM review, 52(3), 471-501.

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Proceedings of the 30th international conference on machine learning (No. CONF, pp. 427-435).

Mu, C., Zhang, Y., Wright, J., & Goldfarb, D. (2016). Scalable robust matrix recovery: Frank–Wolfe meets proximal methods. SIAM Journal on Scientific Computing, 38(5), A3291-A3317.

Pokutta, S., Spiegel, C., & Zimmer, M. (2020). Deep Neural Network Training with Frank-Wolfe. arXiv preprint arXiv:2010.07243.

Recht, B. (2011). A Simpler Approach to Matrix Completion. Journal of Machine Learning Research, 12(12).

Candès, E. J., & Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. IEEE Transactions on Information Theory, 56(5), 2053-2080.

Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. Foundations of Computational mathematics, 9(6), 717.