Master's Degree in Data Science

Academic Year 2024/2025

# Optimization for Data Science

## Comparative Analysis of Frank-Wolfe, Pairwise Frank-Wolfe, and Projected Gradient for Matrix Completion in Recommender Systems

**Ilaria Boschetto**
ID: 2140725

**Federico Clerici**
ID: 2140726

**Raffaele D'Agostino**
ID: 2159093

**Gabriele Villa**
ID: 2157873

## Contents

## 1 Introduction

In this project, various optimization algorithms for **Matrix Completion** [1] were explored within the domain of Recommender Systems. The performance of **Frank-Wolfe (FW)**, **Pairwise Frank-Wolfe (PFW)**, and **Projected Gradient (PG)** methods was specifically evaluated. The goal was to assess their effectiveness in recovering a low-rank matrix from sparse observations, a common challenge in building efficient recommender systems. To this end, these algorithms were applied to **three datasets with varying degrees of sparsity and different dimensions** to observe and compare their behavior.

## 2 Presentation of the problem

This project focuses on **matrix completion**, a field where the core challenge stems from the common real-world scenario where data matrices, such as user-item rating matrices, are inherently sparse; that is, only a small fraction of their entries are observed.

The core of this work is the task of recovering a complete matrix, denoted as $X \in \mathbb{R}^{n_1 \times n_2}$, starting from a set of data that is only partially observed.

The goal is to find the matrix $X$ that minimizes the squared error between the predicted values and those actually observed. To this minimization, however, a crucial constraint is added: the rank of the matrix $X$ must be less than or equal to a certain value $\delta$. This constraint is key to ensuring that the found solution is both accurate and ”simple” and generalizable.

The mathematical formulation of the problem is as follows ([2]):

$$\min_{X \in \mathbb{R}^{n_1 \times n_2}} f(X) := \sum_{(i,j) \in J} (X_{ij} - U_{ij})^2$$

$$s.t. \quad \operatorname{rank}(X) \leq \delta$$

Where:

- $X$ is the complete low-rank matrix to be recovered.
- $U_{ij}$ are the values that have actually been observed.
- $J$ is the set of indices for the positions where data is available (the observations).
- The constraint $\operatorname{rank}(X) \leq \delta$ indicates that a solution $X$ whose rank is at most $\delta$ is being sought, reflecting the low-rank assumption of the underlying matrix.
- $\delta$: a parameter that controls the complexity (effective rank) of the solution.

For this project, the low-rank constraint is approximated by imposing a constraint on the nuclear norm (i.e. the sum of its singular values) effectively relaxing the problem into a convex formulation. Therefore, the actual formulation of the problem is the following:

$$\min_{X \in \mathbb{R}^{n_1 \times n_2}} f(X) := \sum_{(i,j) \in J} (X_{ij} - U_{ij})^2$$

$$s.t. \quad \|X\|_* \leq \delta$$

The set of possible solutions is the **convex hull of rank-1 matrices**:

$$\mathcal{C} = \left\{ X \in \mathbb{R}^{n_1 \times n_2} : \|X\|_* \leq \delta \right\} = \operatorname{conv} \left\{ \delta \, uv^\top : u \in \mathbb{R}^{n_1}, \, v \in \mathbb{R}^{n_2}, \, \|u\| = \|v\| = 1 \right\}.$$

If we denote by $A_J$ the matrix that coincides with $A$ on the indices $J$ and is zero otherwise, then we can write the gradient $\nabla f(X)$ as follows:

$$\nabla f(X) = \begin{cases} 2(X_{ij} - U_{ij}) & \text{if } (i,j) \in J \\ 0 & \text{otherwise} \end{cases}$$

## 3 Frank-Wolfe

To address the matrix completion problem, the **Frank-Wolfe** algorithm was initially employed. This algorithm is particularly effective when the feasible set is a convex compact set and the objective function is convex with a Lipschitz continuous gradient. A key component of this algorithm is the use of the **Linear Minimization Oracle (LMO)** to minimize the following function and whose solution is $\hat{X}_k$:

$$\mathrm{LMO}_C(\nabla f(X_k)) \in \arg\min\{\mathrm{tr}(\nabla f(X_k)^\top X) : \|X\|_* \leq \delta\}$$

The `ARPACK`[1] library was utilized to efficiently approximate the dominant singular value via the Lanczos method, following the approach suggested by [3]. This results in an inexact linear minimization oracle, which balances computational tractability with solution accuracy.

At a certain iteration, this method reconstructs the target matrix as a sparse combination of rank-1 matrices, generating a new descent direction at $X_k$:

$$d_k = \hat{X}_k - X_k.$$

Next, the new iterate can be defined as

$$X_{k+1} = X_k + \alpha_k d_k,$$

where $\alpha_k \in (0, 1]$ is computed through various line-search procedures.

We report below the scheme of the method:

[1]https://docs.scipy.org/doc/scipy/tutorial/arpack.html

**Algorithm 1** Frank–Wolfe method

```
Require: Observed matrix $U \in \mathbb{R}^{n_1 \times n_2}$ (whose observed indices are contained in $J$),
    nuclear norm threshold $\delta > 0$, maximum number of iterations $K > 0$, duality gap
    threshold $\epsilon > 0$
1: Choose a starting matrix $X_1 \in C$
2: for $k = 1, \dots K$ do
3:     Calculate the gradient on the observed indices $J$:
           $$\nabla f(X_k) = \begin{cases} 2(X_{ij}^{(k)} - U_{ij}) & \text{if } (i,j) \in J \\ 0 & \text{otherwise} \end{cases}$$
4:     Solve the Linear Minimization Oracle:
           $$\hat{X}_k \in \arg\min \left\{ \mathrm{tr}(\nabla f(X_k)^\top X) : \|X\|_* \le \delta \right\}$$
       where $\hat{X}_k = \delta \cdot u_1 v_1^\top$ with $u_1, v_1$ being the first singular vectors of $-\nabla f(X_k)$
5:     if $\langle X_k - \hat{X}_k, \nabla f(X_k) \rangle < \varepsilon$ then STOP
6:     end if
7:     Choose $\alpha_k \in (0, 1]$ (various step-size rules were implemented)
8:     Update:
           $$X_{k+1} = X_k + \alpha_k(\hat{X}_k - X_k)$$
9: end for
```

The stopping criterion adopted is based on the value of the **duality gap**, which quantifies how much the objective function could **potentially decrease** if an exact minimization over the feasible set $C$ were possible in the current gradient direction [4]. A small duality gap indicates that the current solution is close to optimality within the feasible region. In conclusion, the choice of the step-size will be discussed later in this paper.

## 4 Pairwise Frank-Wolfe

The basic idea is similar to that of the classical Frank-Wolfe algorithm, with the difference that the search direction here is given by the sum of the Frank-Wolfe direction $d_k^{FW}$ and the away-step direction $d_k^{AS}$.

The away-step direction is computed by identifying the vertex $\hat{X}_k^{AS}$ in the current active set that maximizes the directional derivative in the opposite direction of the gradient, i.e., it corresponds to the vertex that yields the largest increase of the objective function when moving towards it. Formally, $\hat{X}_k^{AS}$ is selected such that

$$\hat{X}_k^{AS} = \arg\max_{X \in S_k} \langle \nabla f(X_k), X \rangle,$$

where $S_k$ is the set of active vertices at iteration $k$. The away-step direction is then

$$d_k^{AS} = X_k - \hat{X}_k^{AS}.$$

**Algorithm 2** Pairwise Frank-Wolfe for Matrix Completion

```
Require: Observed matrix $U \in \mathbb{R}^{n_1 \times n_2}$ (whose observed indices are contained in $J$),
    nuclear norm threshold $\delta$, maximum number of iterations $K$, duality gap threshold $\epsilon$
 1: Choose a starting matrix $X_1 \in C$ and initialize a set of the currently used vertices
    $S_1 = \emptyset$
 2: for $k = 1$ to $K$ do
 3:     Calculate the gradient on $J$:
            $$\nabla f(X_k) = \begin{cases} 2(X_{ij}^{(k)} - U_{ij}) & \text{if } (i,j) \in J \\ 0 & \text{otherwise} \end{cases}$$
 4:     Calculate $\hat{X}_k^{FW} = \arg\min_{X \in \mathcal{C}} \langle \nabla f(X_k), X \rangle$
 5:     Calculate $\hat{X}_k^{AS} = \arg\max_{X \in S_k} \langle \nabla f(X_k), X \rangle$
 6:     if $\langle X_k - \hat{X}_k^{FW}, \nabla f(X_k) \rangle < \varepsilon$ then STOP
 7:     end if
 8:     Define the following directions:
            $$d_k^{FW} = \hat{X}_k^{FW} - X_k, \quad d_k^{AS} = X_k - \hat{X}_k^{AS}$$
 9:     The overall descent direction is computed as:
            $$d_k = d_k^{FW} + d_k^{AS}$$
10:     Compute the maximum step size $\beta$ such that $X_k + \beta d_k \in C$:
            $$\bar{\alpha} = \max \{\beta : X_k + \beta d_k \in C\}$$
11:     Choose $\alpha_k \in (0, \bar{\alpha}]$ (various step-size rules were implemented)
12:     Update:
            $$X_{k+1} = X_k + \alpha_k d_k$$
13:     Update the set of the currently used vertices $S_{k+1}$ adding $\hat{X}_k^{FW}$
14: end for
```

In this case as well, the stopping criterion is based on the duality gap: when the duality gap falls below a predefined tolerance, the algorithm is considered to have converged.

## 4.1 Interpretation of $\bar{\alpha}$ in the Pairwise Frank-Wolfe algorithm

The value of $\bar{\alpha}$ represents the maximum admissible step in the direction $d_k = \hat{X}_k - X_k$ such that the convex combination of the *active atoms* remains valid, that is:

$$\sum_i \lambda_i = 1, \quad \lambda_i \geq 0$$

In the *pairwise* variant of the Frank-Wolfe algorithm, the step is performed by transferring weight ($\lambda_i$) from an existing atom $\hat{X}_k^{AS}$ (called the *away atom*) towards a new atom $\hat{X}_k^{FW}$ (obtained through the Linear Minimization Oracle, LMO). This movement results in an update of the form:

$$X_{k+1} = X_k + \alpha(\hat{X}_k^{FW} - \hat{X}_k^{AS})$$

In terms of the coefficients of the convex combination, this corresponds to:

$$\lambda_s \leftarrow \lambda_s + \alpha, \qquad \lambda_v \leftarrow \lambda_v - \alpha$$

where $\lambda_v$ is the weight of the away atom, while $\lambda_s$ is the weight of the atom obtained through the LMO. To ensure that $\lambda_v \geq 0$ after the update, it is required that:

$$\lambda_v - \alpha \geq 0 \quad \Rightarrow \quad \alpha \leq \lambda_v$$

Therefore, the maximum admissible value of $\alpha$, denoted as $\bar{\alpha}$, is:

$$\bar{\alpha} = \lambda_v$$

This value represents the feasibility constraint imposed by the structure of the domain, and it ensures that the solution remains within the feasible region of the problem.

If $\alpha = \lambda_v$, the weight of the atom $v$ becomes zero, and the atom can be removed from the active set.

## 5 Projected Gradient

The key idea behind this algorithm is to combine the standard gradient descent update with a projection step that ensures the iterates remain within a feasible constraint set.

The detailed scheme of the projected gradient algorithm is given below:

**Algorithm 3** Projected gradient method

```
Require: Observed matrix $U \in \mathbb{R}^{n_1 \times n_2}$, nuclear norm threshold $\delta > 0$, maximum number
    of iterations $K > 0$, solution change threshold $\epsilon > 0$
1: Initialize $X_1 \in C$
2: for $k = 1, \dots K$ do
3:     Choose a fixed $s_k = s > 0$
4:     Set $\hat{X}_k = \rho_C(X_k - s\nabla f(X_k))$
5:     if $\frac{\|\hat{X}_k - X_k\|_F}{\|X_k\|_F} < \epsilon$ then STOP
6:     end if
7:     Choose $\alpha_k \in (0, 1]$ (various step-size rules were implemented)
8:     Set $X_{k+1} = X_k + \alpha_k(\hat{X}_k - X_k)$
9: end for
```

The algorithm stops when the relative change between successive iterates, measured in Frobenius norm, falls below a predefined tolerance: when the solution change is small, it indicates that the solution has stabilized and further updates are negligible. With regard to the value of $s_k$, we selected $s_k = \frac{1}{L}$. Since the gradient is $L$-Lipschitz continuous, selecting a step size $s = 1/L$ ensures the convergence of the projected gradient algorithm to an optimal solution of the optimization problem.

To calculate the projection of the matrix onto the set $C$, we developed a `projection` function, modifying Algorithm 1 proposed by Laurent Condat [5] and adapting it to our setting.

**Algorithm 4** Projection onto Nuclear Norm Ball $C = \{X \mid \|X\|_* \leq \delta\}$

```
Require: A matrix $Y \in \mathbb{R}^{n_1 \times n_2}$, and a radius $\delta > 0$.
Ensure: The projected matrix $\hat{X} \in C$.
 1: // Step 1: Compute Singular Value Decomposition (SVD)
 2: $Y = U\Sigma V^T$, where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_k)$ with $\sigma_1 \geq \cdots \geq \sigma_k \geq 0$, and $k = \min(n_1, n_2)$
 3: Let $s$ be the vector of singular values: $s = [\sigma_1, \dots, \sigma_k]^T$.
 4: if $\sum_{i=1}^{k} \sigma_i \leq \delta$ then
 5:     // Step 2: If already in the ball, return Y
 6:     $\hat{X} = Y$
 7: else
 8:     // Step 3: Project singular values using a variation of Algorithm 1
 9:     // (Note: $s$ is already sorted in descending order as $u$ in Algorithm 1)
10:     for $j = 1, \dots, k$ do
11:         $current\_threshold = (\sum_{r=1}^{j} \sigma_r - \delta)/j$
12:         if $\sigma_j - current\_threshold > 0$ then
13:             $\tau = current\_threshold$
14:         else
15:             break
16:         end if
17:     end for
18:     Set $\sigma_i' = \max\{0, \sigma_i - \tau\}$ for $i = 1, \dots, k$.
19:     Let $\Sigma' = \text{diag}(\sigma_1', \dots, \sigma_k')$.
20:     // Step 4: Reconstruct the projected matrix
21:     $\hat{X} = U\Sigma' V^T$
22: end if
23: return $\hat{X}$
```

The core idea of this algorithm is to compute the SVD decomposition of the matrix, shrinking its singular values by projecting $\Sigma$ onto $C$ and then reconstructing the matrix.

## 5.1 Lipschitz constant calculation

Considering the gradient, we computed the Lipschitz constant $L$ such that

$$\|\nabla f(X_1) - \nabla f(X_2)\|_F \leq L\|X_1 - X_2\|_F$$

for every $X_1, X_2$.

We calculated the difference of the gradients:

$$\begin{aligned} \|\nabla f(X_1) - \nabla f(X_2)\|_F &= \|2(X_1 - U)_J - 2(X_2 - U)_J\|_F \\ &= \|2((X_1 - U) - (X_2 - U))_J\|_F \\ &= \|2(X_1 - X_2)_J\|_F \end{aligned}$$

Let $M = X_1 - X_2$. Then the expression becomes $\|2M_J\|_F$. By the properties of the Frobenius norm, we factored out the constant:

$$\|2M_J\|_F = 2\|M_J\|_F.$$

We know that for any matrix $M$, the Frobenius norm of the masked matrix $M_J$ cannot be greater than the Frobenius norm of the full matrix $M$. This is because $M_J$ contains only a subset (or all) of the elements of $M$, while the others are set to zero:

$$(M_J)_{ij} = \begin{cases} M_{ij} & \text{if } (i,j) \in J \\ 0 & \text{otherwise} \end{cases}$$

Therefore,

$$\|M_J\|_F^2 = \sum_{(i,j)\in J} M_{ij}^2 \leq \sum_{i,j} M_{ij}^2 = \|M\|_F^2.$$

This implies

$$\|M_J\|_F \leq \|M\|_F.$$

Applying this inequality to our expression:

$$2\|M_J\|_F \leq 2\|M\|_F.$$

Substituting back $M = X_1 - X_2$:

$$\|\nabla f(X_1) - \nabla f(X_2)\|_F \leq 2\|X_1 - X_2\|_F.$$

Comparing this with the definition of Lipschitz continuity, we identified the Lipschitz constant $L$:

$$L = 2.$$

## 6 Methodologies

### 6.1 Initialization of the matrix

The initialization of the matrix for the three algorithms is the following:

$$X_1 \in \left\{ \delta u v^T \in \mathbb{R}^{n_1 \times n_2} : u \in \mathbb{R}^{n_1}, v \in \mathbb{R}^{n_2}, \|u\| = \|v\| = 1 \right\}$$

where $u$ and $v$ are randomly generated vectors. This choice ensures that the algorithm starts from a matrix on the boundary of the convex hull.

### 6.2 Step-size strategy

The step-size chosen in our code is the **exact line search**, which guarantees the best results one can obtain in practice. Although it's more computationally expensive than other methods such as Armijo line search or Lipschitz constant dependent step-size, we chose this one to aim at the best possible results. However, other three step-size strategies were included in the code, in case the goal is to achieve faster performance with reduced computational cost. A detailed description of these three strategies can be found in Appendix A.

**Exact line search**: Let $X_k$ be the current estimate of the matrix and $d_k$ the update direction. We aim to solve:

$$\min_{\alpha \in [0, \bar{\alpha}]} f(X_k + \alpha d_k)$$

where the objective function is the quadratic loss:

$$f(X) = \|(X - U)_J\|_F^2$$

Define:

$$R_k = (X_k - U)_J \quad \text{(residual on observed entries)}$$

$$D_k = (d_k)_J \quad \text{(projected direction onto observed entries)}$$

Then:

$$f(X_k + \alpha d_k) = \|R_k + \alpha D_k\|_F^2 = \left(\|R_k\|_F^2 + 2\alpha\langle R_k, D_k\rangle + \alpha^2\|D_k\|_F^2\right)$$

Differentiating with respect to $\alpha$:

$$\frac{d}{d\alpha} f(X_k + \alpha d_k) = 2\langle R_k, D_k\rangle + 2\alpha\|D_k\|_F^2$$

Setting the derivative equal to zero yields the minimizer:

$$\alpha^\star = -\frac{\langle R_k, D_k\rangle}{\|D_k\|_F^2}$$

Finally, by enforcing that $\alpha^\star$ lies within the admissible interval $[0, \bar{\alpha}]$, we obtained:

$$\alpha_{\text{opt}} = \min\left(\bar{\alpha}, \max(0, \alpha^\star)\right)$$

## 6.3 Experiment setting

For the execution of the experiments, we decided to set the maximum number of iterations to 400 and the stopping condition to `1e-4` for both the duality gap (for Frank-Wolfe-based methods) and the difference change of loss (for the Projected Gradient Method). All experiments were conducted on a MacBook Air equipped with an Apple M3 chip and 8 GB of RAM.
To determine the optimal value of the parameter $\delta$, we followed a validation-based selection strategy for all the three algorithms. Specifically, we divided the entries of the observed matrix into training set, validation set and test set by masking different proportions of them. We then trained each algorithm on the training set using various $\delta$ values and selected the one that resulted in the lowest RMSE on the validation set. Once the optimal $\delta$ was identified, we retrained the algorithms on the combined training and validation sets to fully leverage the available data.
Finally, we evaluated the performance of each algorithm on a separate test set in order to assess their generalization capabilities on previously unseen data.

# 7 Real-world datasets

As required, we tested the three algorithms on three different datasets. For completeness, we chose a relatively small dataset, a medium dataset and a massive dataset to analyze the performances in these three different situations.

## 7.1 Amazon Gift Cards Dataset

We evaluated our three algorithms on the *Amazon Gift Cards Dataset*[2]. This dataset contains user ratings specifically related to gift cards, with rating values ranging from 1 to 5. It comprises 377 users and 129 items with a density of 4.9%, indicating that only a small fraction of all possible user-item interactions are observed. The data-splitting strategy used in this case picks the latest two item interactions for each user to evaluate model performance. Specifically, the first $N$–2 interactions are used for training, the ($N$–1)-th for validation, and the $N$-th for testing. In this way, the prediction of the model is compared with the most recent user interaction. This approach yields a more accurate estimation of model performance in a real scenario, where the objective is to predict future user behavior based on historical interactions. The validation set and the test set respectively represent approximately 15% of the data.

| | |
|---|---|
| Training non-zero ratings | 1675 |
| Validation ratings | 377 |
| Test ratings | 377 |

Table 1: Amazon Gift Cards Dataset statistics

| **Algorithm** | $\boldsymbol{\delta}$ | **RMSE Train** | **RMSE Val** | **Rank** |
|---|---|---|---|---|
| Projected Gradient | 880 | 0.780 | 1.337 | 8 |
| Pairwise FW | 600 | 1.724 | 1.877 | 124 |
| Standard FW | 685 | 1.390 | 1.554 | 118 |

Table 2: Best configuration for each algorithm (based on validation RMSE) on the Amazon Gift Cards Dataset.

The best result with this dataset was achieved using the standard Frank-Wolfe method, which not only achieved strong performance but also proved to be the fastest approach. In particular, it reconstructed a matrix of rank 2, effectively capturing the low-dimensional structure. Importantly, the method successfully converges, demonstrating stable optimization behavior in real-world settings.

The Projected Gradient method showed the most accurate reconstruction, with an RMSE of 0.930, but required significantly more computation time. In contrast, standard Frank-Wolfe reached an RMSE of 1.537 and completed in just 0.653 seconds, making it a highly efficient and effective choice.

The differences in rank reduction also provide meaningful insight into each method's ability to exploit the underlying structure of the data. Projected Gradient converged to a rank of 4, standard Frank-Wolfe to rank 2, while Pairwise Frank-Wolfe produced a much higher rank of 87.

| **Algorithm** | **RMSE Test** | **Rank** | **Time(sec)** |
|---|---|---|---|
| Projected Gradient | 0.930 | 4 | 3.210 |
| Pairwise FW | 1.857 | 87 | 4.162 |
| Standard FW | 1.537 | 2 | 0.653 |

Table 3: Results on the Amazon Gift Cards dataset

[2]https://amazon-reviews2023.github.io/data_processing/5core.html#leave-lastoutsplitting

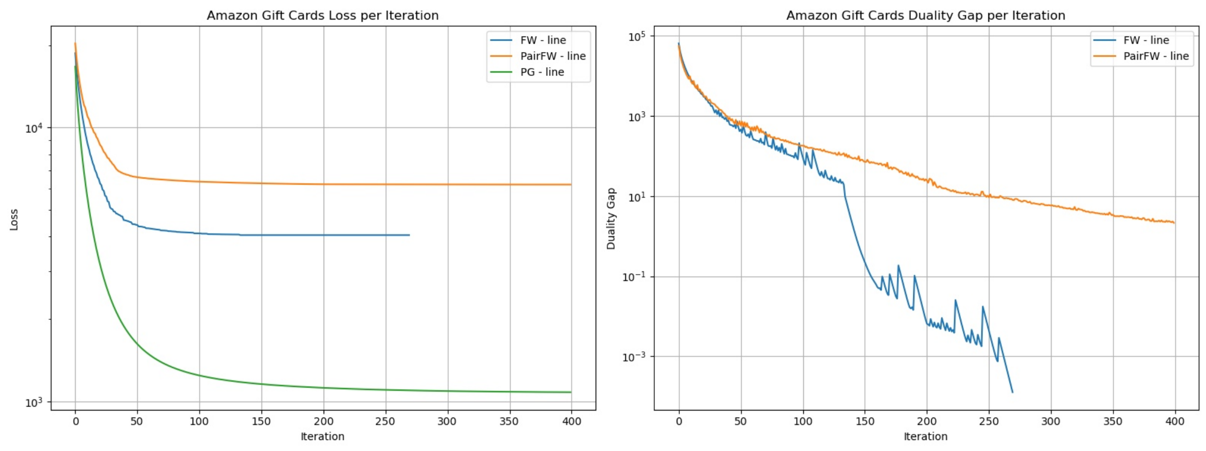

Figure 1: Loss function and duality gap for Amazon Gift Cards dataset versus iterations. The y-axis scale is logarithmic. The Frank Wolfe Standard algorithm reaches convergence before reaching the iterations limit.

## 7.2 Netflix Prize Dataset

We evaluated our three algorithms on the *Netflix Prize Dataset* [3] next, focusing specifically on the first out of the four data files provided for the competition. The dataset contains integer movie ratings from 1 to 5, collected between 1999 and 2005, and is characterized by a high level of sparsity due to many missing entries, which we initially set to zero.

To obtain a more manageable and less sparse subset, we filtered the data by selecting only the movies and users falling in the top 95% and 99% quantiles, respectively, in terms of number of ratings. Furthermore, we restricted our analysis to ratings given in the year 2005, in order to mitigate potential biases introduced by the evolution of Netflix's recommendation systems over time.

After preprocessing, we obtained a dataset consisting of 4,695 users and 225 movies, with an overall density (i.e., the proportion of non-zero entries) of approximately 22%.

Since the dataset includes the timestamps of each rating, we constructed the validation and test sets based on temporal order. Specifically, for each movie, we selected the penultimate rating as part of the validation set and the most recent rating as part of the test set. This setup allows us to evaluate the algorithms' ability to predict the most recent ratings using only the preceding ones.

In Table 4 we can see the statistics of the dataset splitted into train, validation and test.

| | |
|---|---:|
| Training non-zero ratings | 232,510 |
| Validation ratings | 225 |
| Test ratings | 225 |

Table 4: Netflix Prize Dataset statistics

In Table 5 we can see the best hyperparameter selected, together with the train and validation RMSE and the rank of the matrix obtained from the training.

[3]https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data/data?select=combined_data_1.txt

| **Algorithm** | $\delta$ | **RMSE Train** | **RMSE Val** | **Rank** |
|---|---|---|---|---|
| Projected Gradient | 4700 | 0.627 | 0.895 | 96 |
| Pairwise FW | 3010 | 0.962 | 1.047 | 179 |
| Standard FW | 2600 | 1.106 | 1.193 | 77 |

Table 5: Best configuration for each algorithm (based on validation RMSE) on the Netflix Prize Dataset.

Based on the final evaluation on the test set, the results reveal a clear trade-off between accuracy, computational efficiency, and the ability to exploit low-rank structure. The Projected Gradient method achieved the best reconstruction accuracy with an RMSE of 0.889, converging to a matrix of rank 96 requiring 19.380 seconds. The Pairwise Frank-Wolfe algorithm, while achieving a decent RMSE of 1.089, performed poorly in both efficiency and rank reduction. It required 101.972 seconds and converged to a high rank of 188.

On the other hand, the standard Frank-Wolfe offered the most balanced performance: although its RMSE of 1.263 was the highest, it achieved this result in just 11.622 seconds, and with the lowest rank among the three methods. This suggests that standard Frank-Wolfe was the most effective at identifying a compact, low-dimensional structure with relatively little computational cost.

| **Algorithm** | **RMSE Test** | **Rank** | **Time(sec)** |
|---|---|---|---|
| Projected Gradient | 0.889 | 96 | 19.380 |
| Pairwise FW | 1.089 | 188 | 101.972 |
| Standard FW | 1.263 | 68 | 11.622 |

Table 6: Results on the Netflix dataset

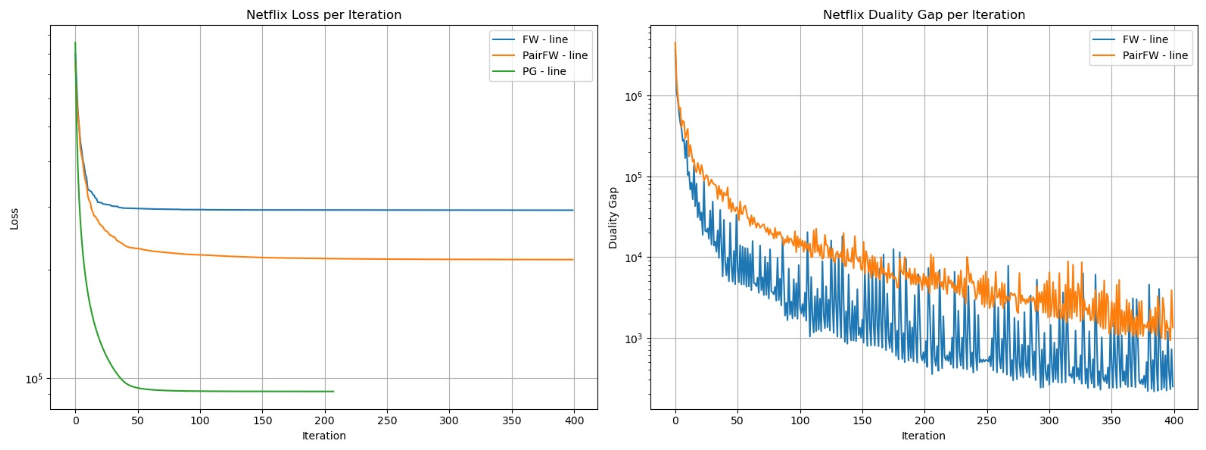

Figure 2: Loss function and duality gap for Netflix Prize dataset. The y-axis scale is logarithmic. The projected gradient algorithm converges before reaching the iterations limit.

## 7.3 MovieLens Dataset

Eventually, we evaluated our three algorithms on the *MovieLens 100k Dataset*[4]. The dataset comprises 100,000 integer movie ratings, ranging from 1 to 5, for 943 users across 1682 films with a density of 6.3%. The observed values have been divided into training and test sets with a proportion of 90-10. From the training set, we selected a 20% for the validation set to choose the proper value of $\delta$.

| | |
|---|---|
| Training non-zero ratings | 72,000 |
| Validation ratings | 18,000 |
| Test ratings | 10,000 |

Table 7: MovieLens Dataset statistics

The Projected Gradient algorithm achieved the best overall performance, yielding the lowest test RMSE and the smallest recovered rank. The Pairwise Frank-Wolfe algorithm performed slightly worse in terms of test error and returned a significantly higher rank, while also being the slowest method, with a running time exceeding 2700 seconds. The standard Frank-Wolfe algorithm exhibited a higher rank than the Pairwise Frank-Wolfe, although it worked in a very small running time.

| **Algorithm** | $\boldsymbol{\delta}$ | **RMSE Train** | **RMSE Val** | **Rank** |
|---|---|---|---|---|
| Projected Gradient | 5000 | 0.521 | 1.002 | 112 |
| Pairwise FW | 5000 | 0.642 | 1.008 | 381 |
| Standard FW | 6000 | 0.595 | 1.010 | 401 |

Table 8: Best configuration for each algorithm (based on validation RMSE) on the MovieLens dataset.

| **Algorithm** | **RMSE Test** | **Rank** | **Time (sec)** |
|---|---|---|---|
| Projected Gradient | 0.951 | 115 | 260.403 |
| Pairwise FW | 0.980 | 379 | 2760.050 |
| Standard FW | 0.980 | 401 | 47.662 |

Table 9: Results on the MovieLens dataset

[4]https://grouplens.org/datasets/movielens/100k/

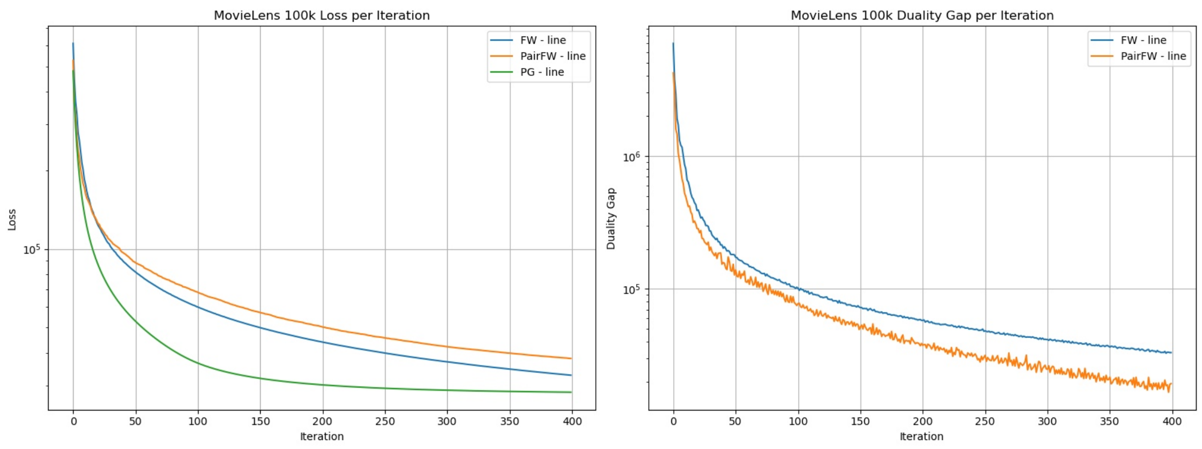

Figure 3: Loss function and duality gap for MovieLens dataset. The y-axis scale is logarithmic.

## 8 Conclusions

In this project, we achieved a good trade-off between prediction correctness and low-rank approximations. By increasing the number of iterations it is possible to further improve the quality of the reconstructed matrix, which would come at the cost of higher recovered matrix ranks. Additionally, it is possible to tune the $\delta$ parameter alongside other step size rules to better balance convergence speed and solution quality. Overall, the algorithms produced relatively low test errors while maintaining low-rank solutions, confirming the effectiveness of matrix completion methods in recommendation system applications.

## A Step-size strategy descriptions

The other three strategies that were implemented in our project to find a suitable step-size at each iteration were:

- **Diminishing stepsize**: it's the simpler one to compute and it gets smaller as the number of iterations increases:

  $$\alpha_k = \min\left\{\frac{2}{k+2}, \bar{\alpha}\right\}$$

  Therefore, if the value $\frac{2}{k+2}$ is greater than $\bar{\alpha}$, the latter will be selected as the step-size.

- **Lipschitz constant dependent step-size**: using a Lipschitz constant dependent step-size offers several advantages. It ensures stable and controlled progress by preventing steps that are too large, and it also adapts to the local curvature of the function, allowing larger steps in flatter regions and smaller ones in highly curved areas.

  $$\alpha_k = \alpha_k(L) := \min\left\{-\frac{\nabla f(X_k)^\intercal d_k}{L\|d_k\|^2}, \bar{\alpha}\right\}$$

- **Armijo line search**: the method iteratively shrinks the step size in order to guarantee a sufficient reduction of the objective function. In this project, we fixed parameters $\delta = \frac{1}{2}$ and $\gamma = \frac{1}{10}$. $\bar{\alpha}$ was set to 1 in the Frank-Wolfe algorithm and the projected gradient algorithm, while in the Pairwise Frank-Wolfe algorithm it was set to the value $\bar{\alpha}$ calculated at line 10 of Algorithm 2: then we tried the steps $\alpha = \delta^m \bar{\alpha}$ with $m \in \{0, 1, 2, \ldots\}$ until the sufficient decrease inequality

  $$f(X_k + \alpha d_k) \le f(X_k) + \gamma\alpha\nabla f(X_k)^\intercal d_k \tag{1}$$

  held, and set $\alpha_k = \alpha$

## References

[1] Immanuel M Bomze, Francesco Rinaldi, and Damiano Zeffiro. Frank–wolfe and friends: a journey into projection-free first-order optimization methods. *4OR*, 19:313–345, 2021.

[2] Robert M Freund, Paul Grigas, and Rahul Mazumder. An extended frank–wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on optimization*, 27(1):319–346, 2017.

[3] Cyrille W Combettes and Sebastian Pokutta. Complexity of linear minimization and projection on some sets. *Operations Research Letters*, 49(4):565–571, 2021.

[4] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pages 427–435. PMLR, 2013.

[5] Laurent Condat. Fast projection onto the simplex and the $l_1$ ball. *Mathematical Programming*, 158(1):575–585, 2016.