Optimization for Data Science: final project

Frank-Wolfe methods for Recommender Systems

Kokot Maksim (ID: 2072065)

Date: 17/09/2023

1 Introduction

In this project we explore two important algorithms designed for constraint optimization tasks: Frank-Wolfe and Projected Gradient.

The project features theoretical analysis of the algorithms referencing to the existing papers dedicated to their computational cost and application. Moreover, the project is aimed to demonstrate the algorithms performance for project real world Recommender System task. In order to do that, we have selected Netflix Prize as such a task, formalized it as constrained and convex optimization problem, applied the algorithms to that problem.

2 Formalization

In this section we formulate convex and constrained optimization problem which solution is aimed to build Recommender System. Further analysis in mainly based on [1] and [2]. [1] describes algorithms' computational costs per iteration with respect to different convex sets commonly used in optimization. Meanwhile, [2] gives stronger convergence results Frank-Wolfe algorithm.

2.1 Low rank matrix completion

The majority of Recommender System problems provide user-item rating table, which contains large amount of missing values. In this case, the problem could be formalized as matrix completion task.

First of all, we will consider the constrained optimization problem

$$\min_{x \in C} f(x), \tag{1}$$

where:

x – solution of the problem

$C \subset R^n$ – a compact convex set

n – dimension of vector x

f – function to minimize

For given sparse rating matrix M, we are required to obtain matrix X whose entries suitably match to corresponding entries of matrix M in case if those entries are not missed. Moreover, we expect the entries of matrix X, which correspond to missing values in matrix M, to be plausible predictions of unknown ratings.

For this purpose, we impose low-rank constraint to matrix X. This type of constraint is aimed to preserve interdependency between the entries.

Thus, we may transform (1) into

$$\min_{rank(x)\leq r} f(x), \tag{2}$$

where:

r – threshold for rank of x

However, the low-rank constraint is hard to handle and it makes the problem non-convex. In order to overcome this problem, we can replace it with the nuclear norm constraint as follows

$$\min_{\|x\|_{nuc}\leq \beta} f(x), \tag{3}$$

where:

β– threshold for nuclear norm of x

By imposing this constraint, we obtain a convex optimization problem. The set C satisfying this constrain could be described as a convex hull of points described in expression bellow

$$C = \{X \in R^{n\times m} \mid \|X\|_{nuc} \leq \beta\} = \{\beta uv^T \mid u \in R^n, v \in R^m, \|u\|_2 = \|v\|_2 = 1\} \tag{4}$$

2.2 Loss function

As a function to minimize f, we can use squared Frobenius norm of difference between matrices M and X. Note, that matrix $M - X$ is also sparse and its entry is defined if and only if corresponding entry of matrix M is defined.

$$f(X) = \|M - X\|_F^2, \tag{5}$$

Thus, expression (3) could be rewritten as

$$\min_{\|x\|_{nuc}\leq \beta} \|M - X\|_F^2, \tag{6}$$

2.3 Gradient

The gradient matrix for current solution is calculated as follows:

$$\nabla f(X) = -2(M - X) \tag{7}$$

2.4 Frank-Wolfe algorithm

In this project we apply Frank-Wolfe algorithm to the convex minimization problem described in expression (6). This is first-order algorithm which is purposed for solving constrained optimization problems. It is described in Algorithm 1.

Algorithm 1 Frank-Wolfe

Input: Start point $\boldsymbol{x_0} \in \boldsymbol{C}$, step-size strategy $(\boldsymbol{\gamma_t})_{t \in N} \subset [\mathbf{0}, \mathbf{1}]$

for $\boldsymbol{t = 0}$ **to** $\boldsymbol{T - 1}$ **do**

$$\boldsymbol{w_t} := \underset{v \in C}{\arg\min} \langle v, \nabla f(x_t) \rangle$$

$$\boldsymbol{x_{t+1}} := \boldsymbol{x_t} + \boldsymbol{\gamma_t}(\boldsymbol{w_t} - \boldsymbol{x_t})$$

End for

For matrix completion with the nuclear norm constraint, we can clarify the computation of w_t for each iteration. First, we need to obtain u_t and v_t which are the top left and right singular vectors of $-\nabla f(x_t)$. This could be done by using truncated singular value decomposition (truncated SVD) with ARPACK solver. Second, we need to compute w_t which equals $\beta u_t v_t^T$. The whole algorithm is described in Algorithm 2.

Algorithm 2 Frank-Wolfe for matrix completion with the nuclear norm constraint

Input: Start point $\boldsymbol{x_0} \in \boldsymbol{C}$, step-size strategy $(\boldsymbol{\gamma_t})_{t \in N} \subset [\mathbf{0}, \mathbf{1}]$

for $\boldsymbol{t = 0}$ **to** $\boldsymbol{T - 1}$ **do**

Compute $(\boldsymbol{u_t}, \boldsymbol{v_t})$, the top singular vectors of $-\nabla \boldsymbol{f}(\boldsymbol{x_t})$

$$\boldsymbol{w_t} := \boldsymbol{\beta u_t v_t^T}$$

$$\boldsymbol{x_{t+1}} := \boldsymbol{x_t} + \boldsymbol{\gamma_t}(\boldsymbol{w_t} - \boldsymbol{x_t})$$

End for

The huge advantage of Frank-Wolfe algorithm is that it uses linear minimizations over given convex set and it is projection-free algorithm, hence its usage does not suppose solving quadratic programming problem [1]. Moreover, it doesn't require to use full SVD and relies on using truncated SVD, which is suitable for sparse data.

During the classes it has been proven that if f is a convex function with Lipschitz continuous gradient having constant L and for each iteration t we use $\gamma_t = \frac{2}{t+2}$, we have the following convergence result:

$$f(x_{t+1}) - f(x^*) \leq \frac{2LD^2}{t+1}, \quad (8)$$

where:

x^* – optimal solution of the problem

L – Lipschitz constant

D – diameter of the convex set C

Diameter of the convex set C can be computed by following formula:

$$D = \max_{x,y \in C} \|x - y\|_2 \quad (9)$$

According to paper [1], linear minimization over nuclear norm-ball performed each iteration has a following complexity:

$$O(\nu \ln(m+n)\sqrt{\sigma_1}/\sqrt{\varepsilon}), \quad (10)$$

where:

ν – the number of nonzero entries of $-\nabla f(x_t)$

σ_1 – the top singular value of $-\nabla f(x_t)$

ε – the additive error which satisfies $\sigma_1 - u^T\left(-\nabla f(x_t)\right)v \leq \varepsilon$

Note that $\nu \leq mn$. As the rule, in Recommender System tasks matrix $-\nabla f(x_t)$ is highly sparse, so $\nu \ll mn$. The expression (10) shows that sparsity of $-\nabla f(x_t)$ reduces the complexity of linear minimization. This implies effectiveness of Frank-Wolfe for Recommender System tasks.

2.5 Projected Gradient algorithm

In addition, we are going to apply Projected Gradient algorithm to the same problem (6). Unlike Frank-Wolfe algorithm, this algorithm is projection-based and it requires to compute projection onto C every iteration. Projected Gradient algorithm is described in Algorithm 3.

Algorithm 3 Projected Gradient

Input: Start point $\boldsymbol{x_0} \in \boldsymbol{C}$, step-size strategies $(\boldsymbol{s_t})_{t\in N} > \boldsymbol{0}$, $(\boldsymbol{\alpha_t})_{t\in N} \subset [\boldsymbol{0}, \boldsymbol{1}]$

for $\boldsymbol{t = 0}$ **to** $\boldsymbol{T - 1}$ **do**

$\hat{\boldsymbol{x}}_{\boldsymbol{t}} := \boldsymbol{proj}(\boldsymbol{x_t} - \boldsymbol{s_t}\nabla \boldsymbol{f}(\boldsymbol{x_t}))$

$\boldsymbol{x_{t+1}} := \boldsymbol{x_t} + \boldsymbol{\alpha_t}(\hat{\boldsymbol{x}}_{\boldsymbol{t}} - \boldsymbol{x_t})$

End for

A projection onto the nuclear norm-ball task can be transformed into projection onto the simplex task. For this purpose, we need first to obtain $(U, \sigma, V) \in R^{n\times k} \times R^{k} \times R^{m\times k}, k = min\{n, m\}$ as the results of SVD of $x_t - s_t\nabla f(x_t)$. Then we need to compute $\hat{\sigma}$ as projection of σ onto the simplex $\Delta_k = \{x \in R^k \mid x_l \geq 0 \; for \; 1 \leq l \leq k, \sum_{l=1}^{k} x_l = \beta\}$. Finally, a projection onto the nuclear norm-ball could be calculated as $\hat{x}_t := U\, diag(\hat{\sigma})V^T$. The whole algorithm is described in Algorithm 4.

Algorithm 4 Projected Gradient for matrix completion with the nuclear norm constraint

Input: Start point $\boldsymbol{x_0} \in \boldsymbol{C}$, step-size strategies $(\boldsymbol{s_t})_{t\in N} > \boldsymbol{0}$, $(\boldsymbol{\alpha_t})_{t\in N} \subset [\boldsymbol{0}, \boldsymbol{1}]$

for $\boldsymbol{t = 0}$ **to** $\boldsymbol{T - 1}$ **do**

$\boldsymbol{w_t} := \boldsymbol{x_t} - \boldsymbol{s_t}\nabla \boldsymbol{f}(\boldsymbol{x_t})$

Compute $(\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V})$, the results of SVD of $\boldsymbol{w_t}$

Compute $\hat{\boldsymbol{\sigma}} = \boldsymbol{proj}(\boldsymbol{\sigma}, \Delta_{\boldsymbol{k}})$

$\hat{\boldsymbol{x}}_{\boldsymbol{t}} := \boldsymbol{U}\, \boldsymbol{diag}(\hat{\boldsymbol{\sigma}})\boldsymbol{V^T}$

$\boldsymbol{x_{t+1}} := \boldsymbol{x_t} + \boldsymbol{\alpha_t}(\hat{\boldsymbol{x}}_{\boldsymbol{t}} - \boldsymbol{x_t})$

End for

The projection of σ onto the simplex Δ_k can be obtained by using an algorithm proposed here [3]. In practice, this algorithm has complexity $O(n)$.

The course materials provides us a proof for a statement that if f is a convex function with Lipschitz continuous gradient having constant L and for each iteration t we use $\alpha_t = \frac{1}{L}$, we have the following convergence result:

$$f(x_{t+1}) - f(x^*) \leq \frac{2L}{t}\|x_1 - x^*\|^2 \tag{11}$$

Finding projection on nuclear norm-ball performed each iteration has a following complexity [1]:

$$O(mn\min\{m,n\} + \min\{m^3, n^3\}) \quad (12)$$

3 Data

In order to demonstrate developed solutions, we chose MovieLens dataset for Netflix Prize competition. This dataset has several available variants. We used the smallest variant [4] due to limited memory. Thus, selected dataset contains 100 000 ratings provided by 600 users to 9000 movies.

The set of unique available ratings is $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$.

The project is mainly aimed to demonstrate the way selected algorithms converge to optimal solution. However, we believe that showing predictive ability of obtained solution is also necessary. For this reason, we split given rating into train and test part in such a way that test size equals 25% of all dataset.

4 Conducting experiment

4.1 Experiment's configuration

Before we start conducting the experiment, we have to properly select several algorithms' input parameters such as number of iteration and step-size strategies.

For each algorithm step-size step-size strategy has been carefully chosen.

Multiple papers including [2] describes Frank-Wolfe variant with step-size strategy $\gamma_t = \frac{2}{t+2}$ where t is number of iteration. In our case, γ_t values obtained this way turned out to be large and validation curves looked noisy. In order to make validation curves look smooth and obtain better convergence results, we have selected step-size strategy $\gamma_t = \frac{0.4}{t+2}$.

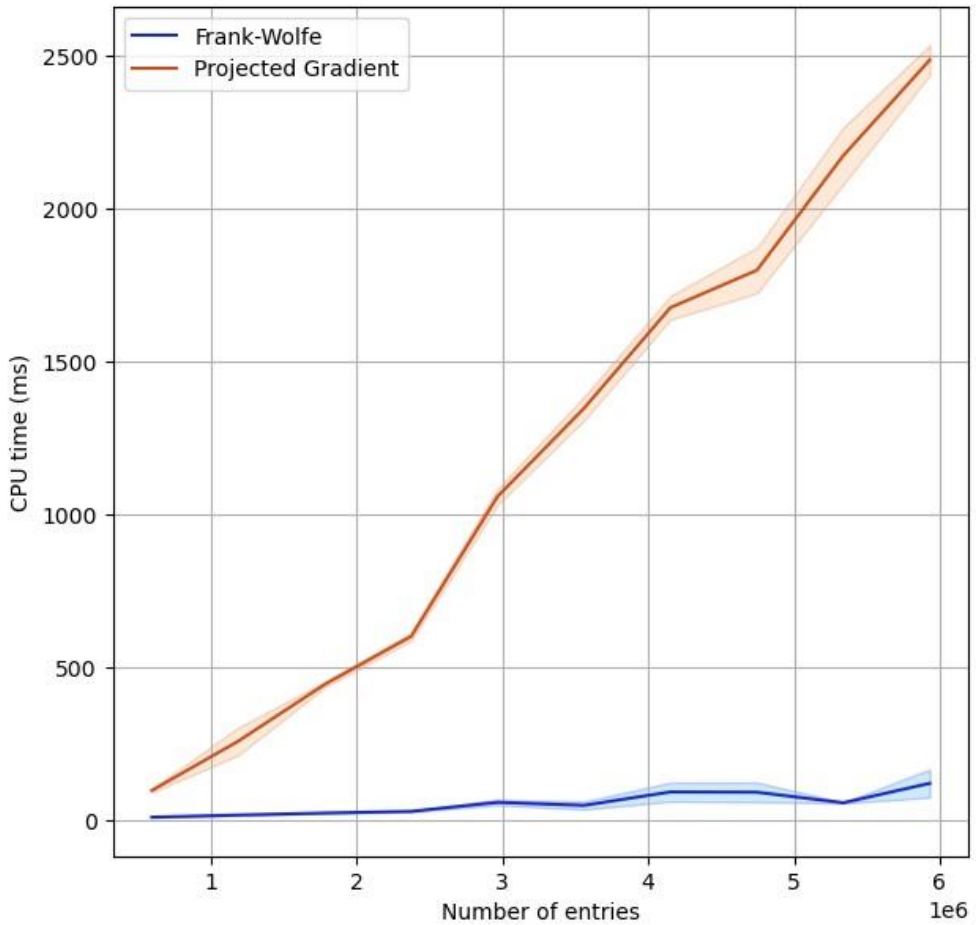
Minimized loss function f is quadratic convex function with Lipschitz constant $L = 2$. During the classes we proved faster convergence rate of Projected Gradient algorithm for the case when we minimize a convex function with Lipschitz continuous gradient and fix $s_t = \alpha_t = 1/L$. For this reason, we set $s_t = \alpha_t = 1/2$.

It’s possible to show that at least 2000 iterations are sufficient to reach a solution closed to optimal one for both algorithms given introduced step-size strategies.

Finally, we have to set parameter β for with the nuclear norm constraint. Empirically we have selected $\beta = 10000$. This value is enough to make the predictions match corresponding ratings if they exist. Moreover, this value is small enough for preserving interdependency between entries, which leads to plausibility of predictions corresponding to unknown ratings.

4.2 Cost per iteration

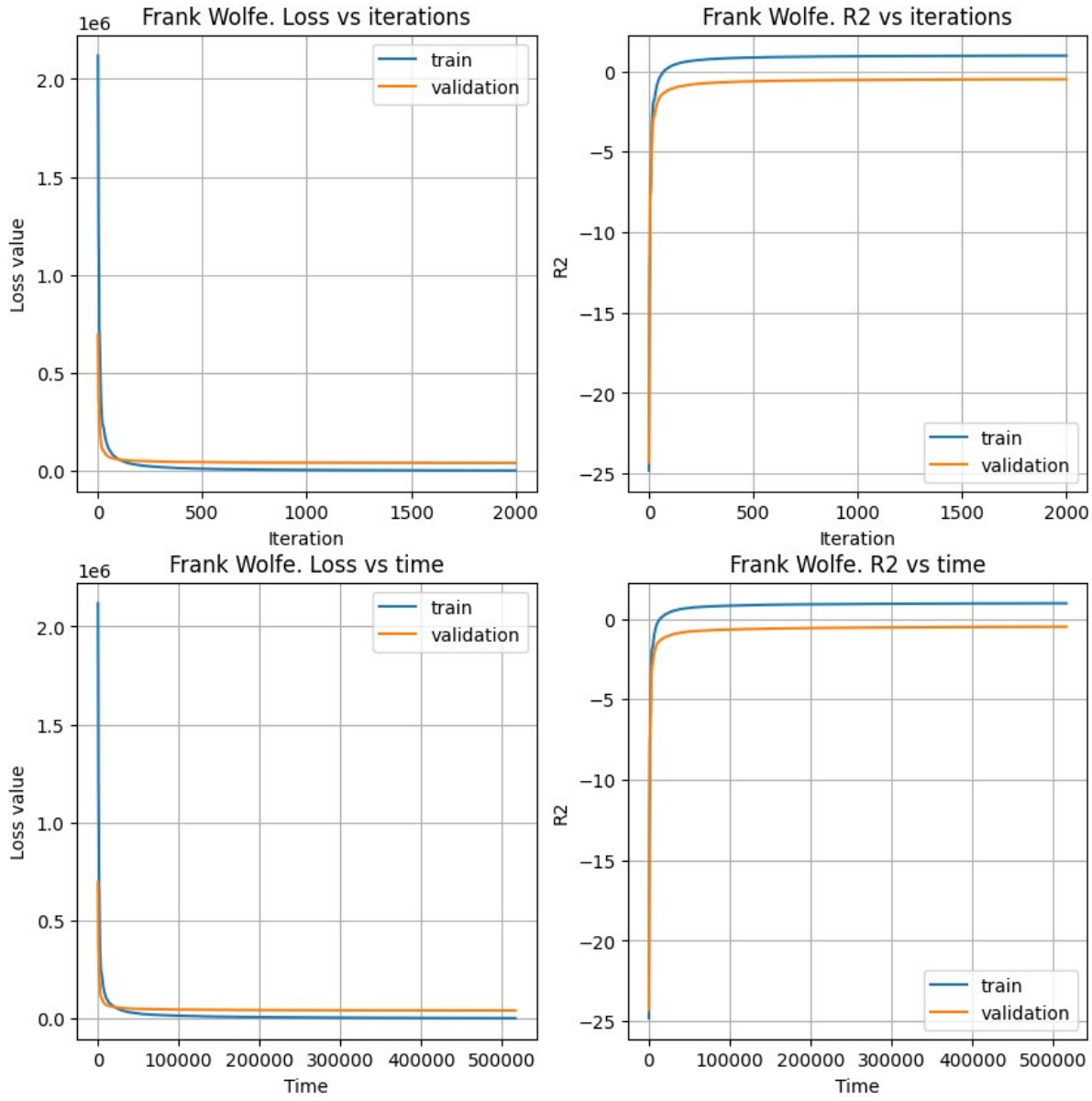
First we are going to compare computational costs per iteration for both algorithms. The comparison will be held with respect to number of entries in matrix X. After conducted analysis we get the following plot:

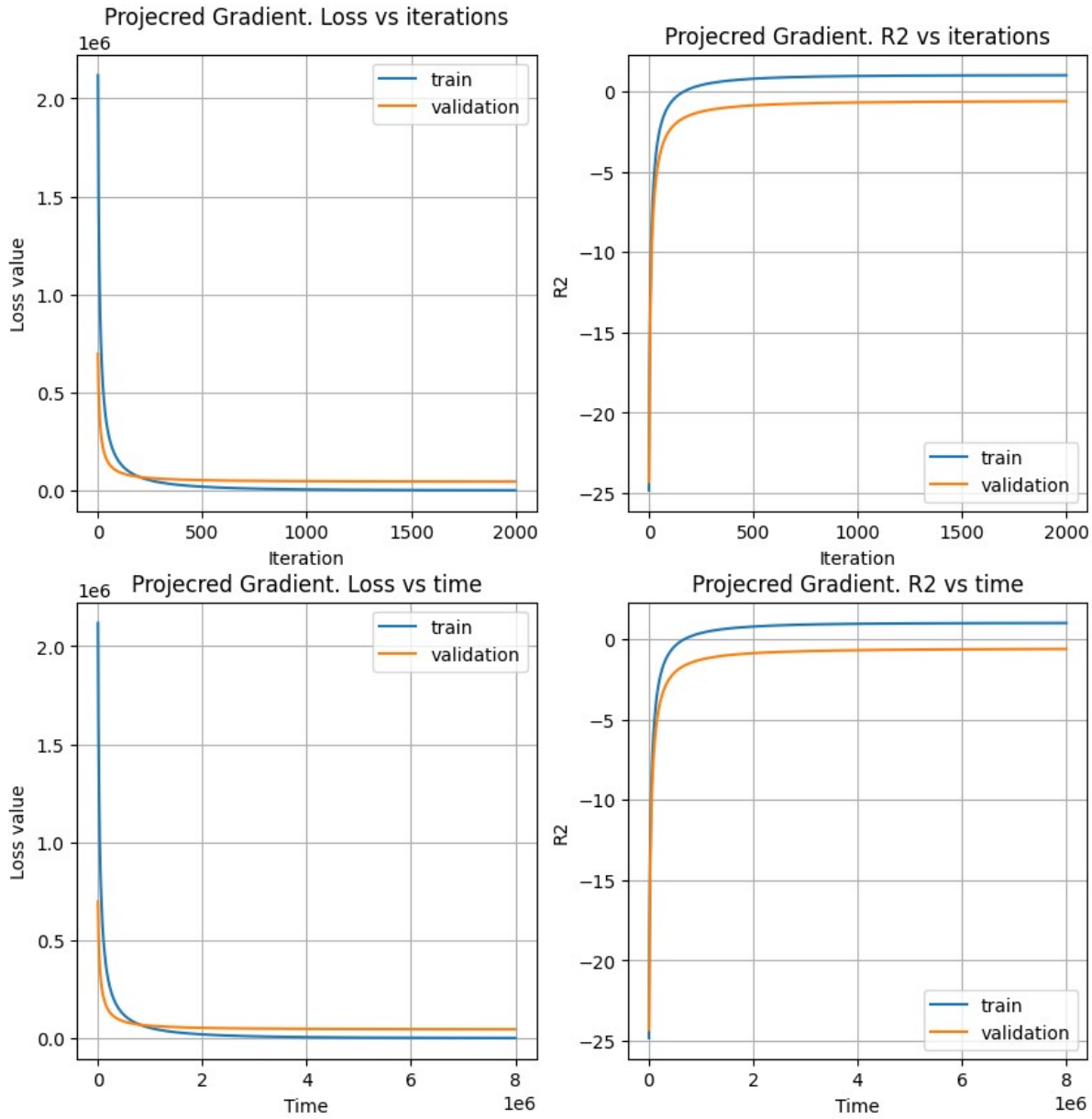


Here we can clearly observe that Projected Gradient iteration costs much more than Frank-Wolfe iteration, especially if we deal with large scale problem. Those results are consisted with theoretical analysis conducted bellow.

4.3 Loss function and R2 values

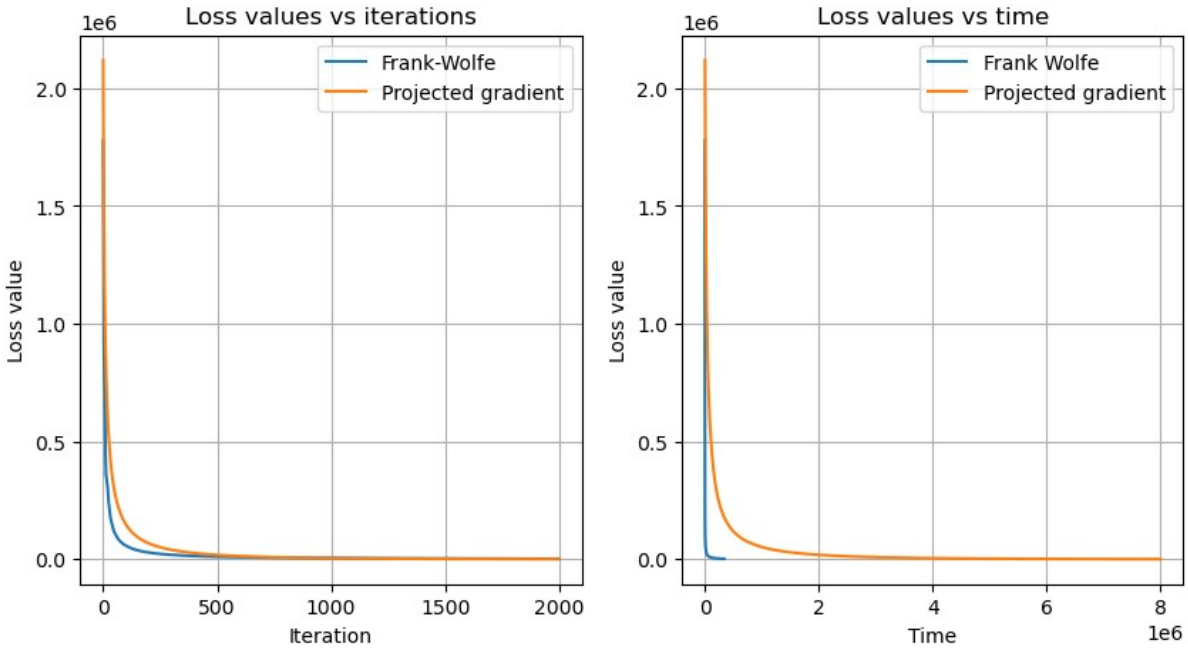
Now we are going to see the performance of both algorithms related to chosen Recommender System problem. For this purpose, we plot loss function and R2 values against number of iterations and CPU time. In addition, we make those plots for train and test datasets separately.





During 2000 iterations, both algorithms lead to sufficient decrease loss function value.

Now we compare convergence results of the algorithms.



Frank-Wolfe algorithm seems to have better convergence results in terms of iterations. However, it's impossible to notice, that final loss function value after using Projected Gradient algorithm (480) is smaller than one obtained after using Frank-Wolfe algorithm (2073).

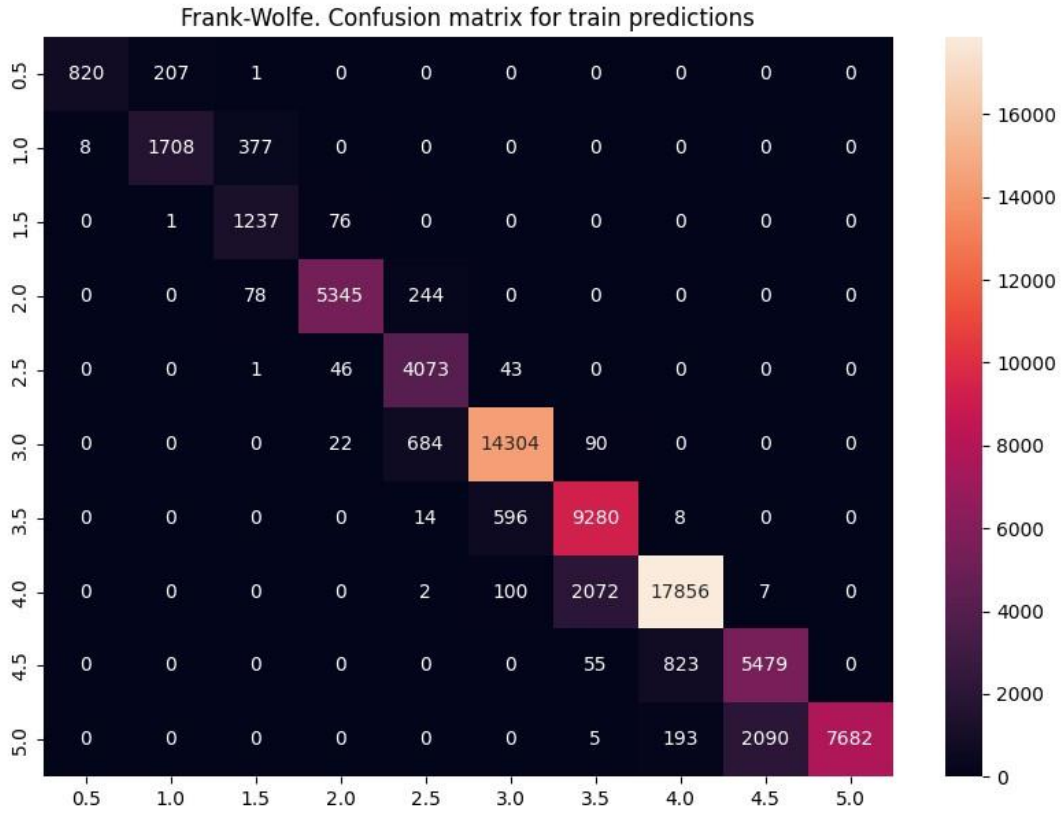
The plot above suggests that Projected Gradient requires less iterations to reach good solution for the problem. However, due to the fact that Projected Gradient iteration is costlier than Frank-Wolfe iteration, Frank-Wolfe algorithm consumes much less CPU time to find the solution.

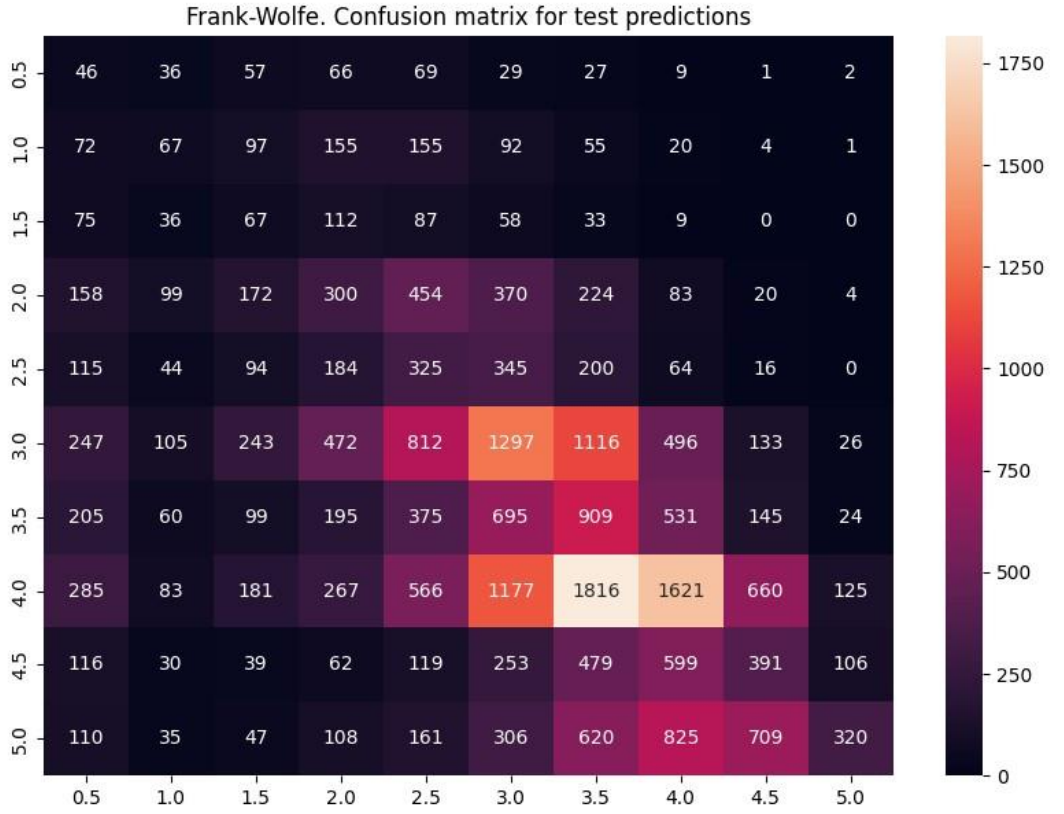
Final R2 scores are listed in the table below:

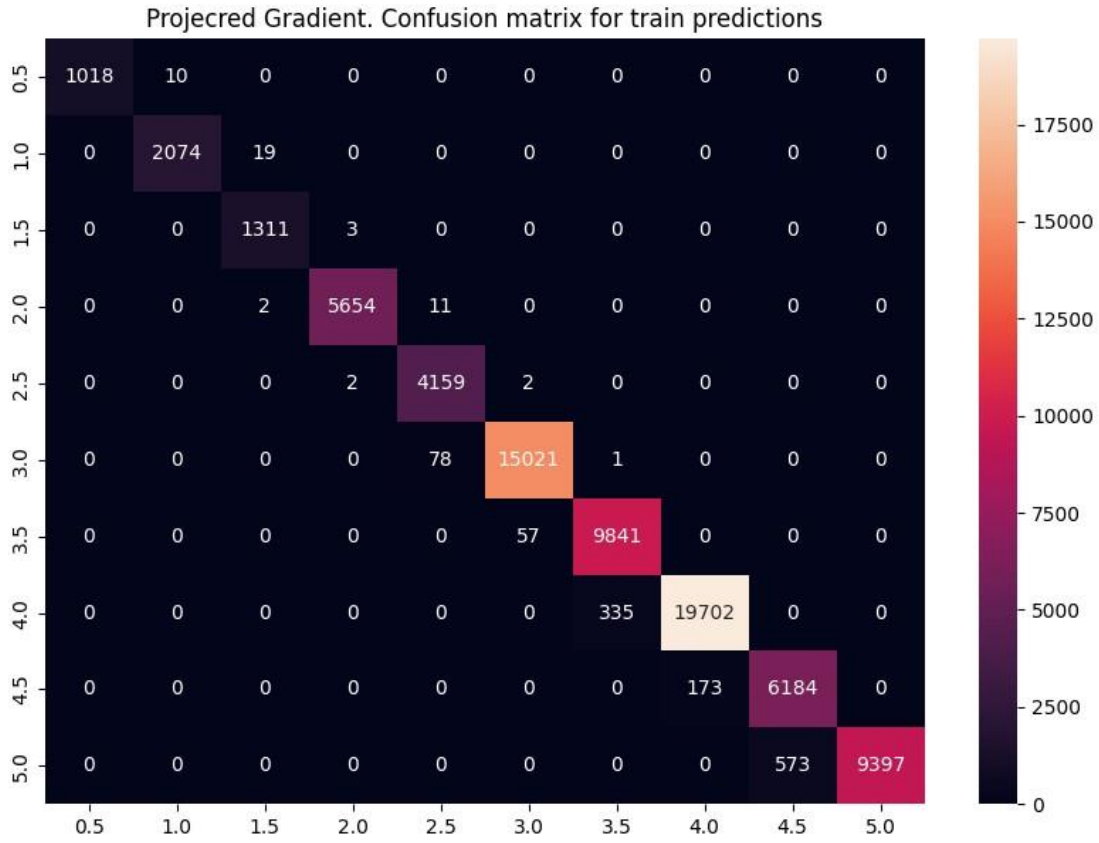
	Dataset	
Algorithm	train	test
Frank-Wolfe	0.97	-0.37
Projected Gradient	0.99	-0.49

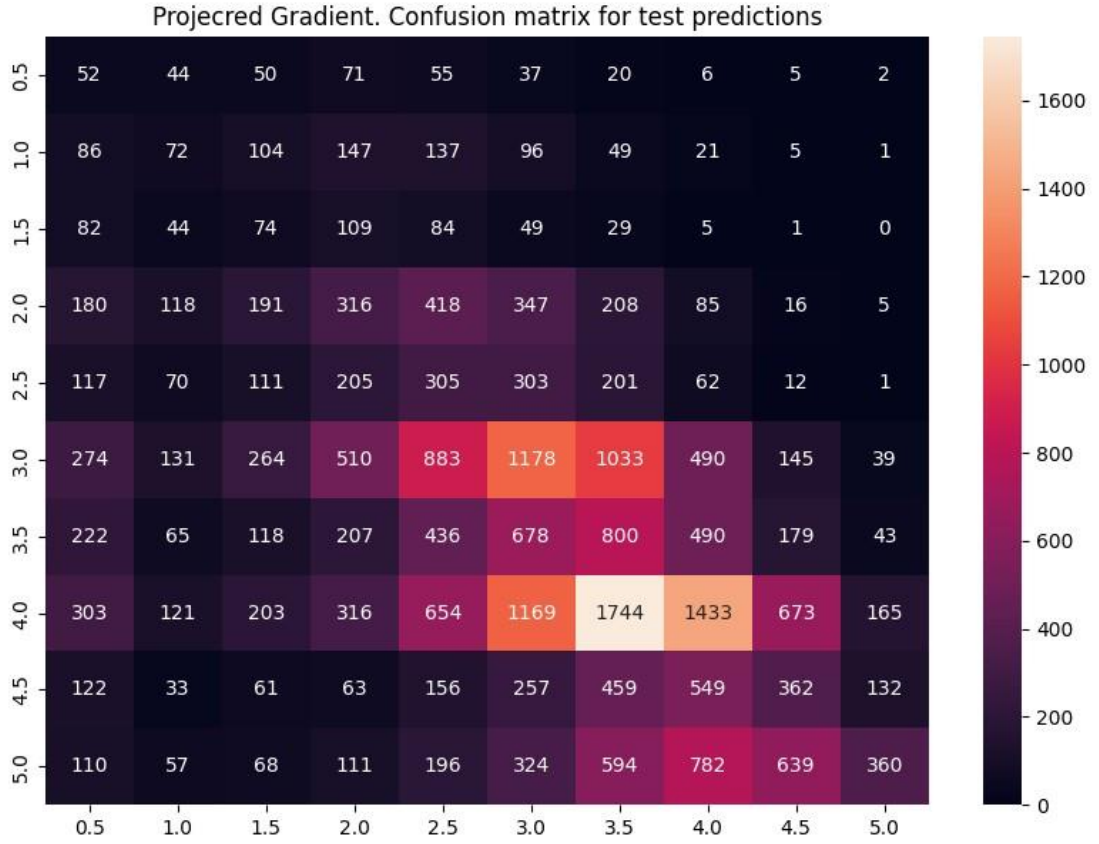
4.4 Confusion matrices

In order to explicitly estimate the plausibility of obtained solutions in terms of providing recommendations, we plot confusion matrices for train ant test predictions.


Frank-Wolfe. Confusion matrix for train predictions
0.5 820 207 1 0 0 0 0 0 0 0
1.0 8 1708 377 0 0 0 0 0 0 0
1.5 0 1 1237 76 0 0 0 0 0 0
2.0 0 0 78 5345 244 0 0 0 0 0
2.5 0 0 1 46 4073 43 0 0 0 0
3.0 0 0 0 22 684 14304 90 0 0 0
3.5 0 0 0 0 14 596 9280 8 0 0
4.0 0 0 0 0 2 100 2072 17856 7 0
4.5 0 0 0 0 0 0 55 823 5479 0
5.0 0 0 0 0 0 0 5 193 2090 7682
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
16000 14000 12000 10000 8000 6000 4000 2000 0


Frank-Wolfe. Confusion matrix for test predictions
0.5 46 36 57 66 69 29 27 9 1 2
1.0 72 67 97 155 155 92 55 20 4 1
1.5 75 36 67 112 87 58 33 9 0 0
2.0 158 99 172 300 454 370 224 83 20 4
2.5 115 44 94 184 325 345 200 64 16 0
3.0 247 105 243 472 812 1297 1116 496 133 26
3.5 205 60 99 195 375 695 909 531 145 24
4.0 285 83 181 267 566 1177 1816 1621 660 125
4.5 116 30 39 62 119 253 479 599 391 106
5.0 110 35 47 108 161 306 620 825 709 320
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
1750 1500 1250 1000 750 500 250 0





By looking at main diagonals of confusion matrices built for train prediction, we can conclude, that values of matrix X corresponding to train ratings

of matrix M have sufficiently become closer to given ratings. However, this could not be said about test predictions, even though it's possible to see moderate values close to part of main diagonals related to good and neutral ratings. Generally, such results are frequently obtained collaborative by using collaborative filtering methods for real world Recommender System problems.

5 Conclusion

In this project we explored performance of Frank-Wolfe and Projected Gradient Algorithms in constrained and convex optimization problem related to Recommender Systems. By analyzing papers describing algorithms, their complexity, convergence results and conducting computational experiments, we may conclude, that Projected Gradient algorithm is computationally expensive with respect to one single iteration compared with Frank-Wolfe algorithm due to several reasons. First, Projected Gradient algorithm requires to find projection which is quadratic programming problem, whereas Frank-Wolfe is projection-free algorithm. Second, Projected Frank-Wolfe exploits truncated SVD on sparse matrix, and this is computationally less expensive than using full SVD for Projected Gradient algorithm. Those aspects make Projected Gradient algorithm converge slower to optimal solution in term of CPU time. Meanwhile, one Projected Gradient iteration gives a new solution X which is closer to optimal solution than the one given by Frank-Wolfe iteration. However, this advantage doesn't make Projected Gradient algorithm provide the fastest CPU time solution.

References

[1] Cyrille W. Combettes, Sebastian Pokutta. «Complexity of Linear Minimization and Projection on Some Sets». In: (2021).

[2] Martin Jaggi. «Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization».

[3] L. Condat. «Fast projection onto the simplex and the l1 ball». In: (2016).

[4] MovieLens dataset. URL: https://files.grouplens.org/datasets/movielens/ml-latest-small.zip.

[5] Lijun Ding, Madeleine Udell. «Frank-Wolfe Style Algorithms for Large Scale Optimization».