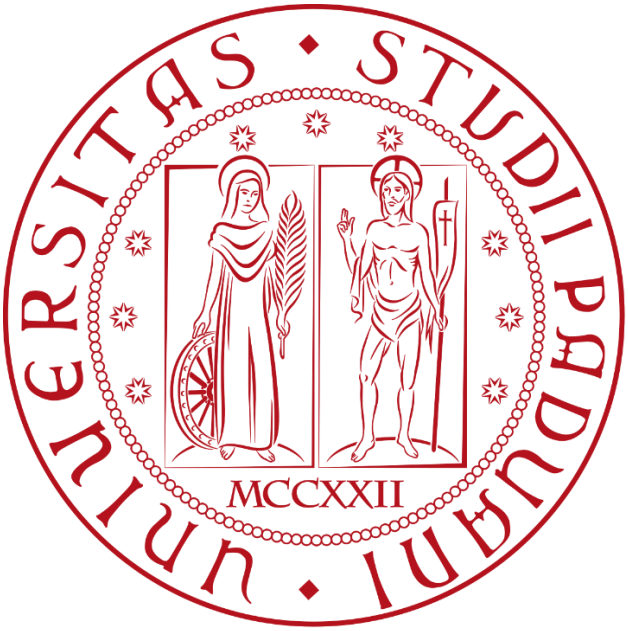

Analysis of Frank-Wolfe and Pairwise Frank-Wolfe Algorithms on the LASSO Problem

Optimization for Data Science (2023/2024)

Roya Ghamari

ID: 2071969

Contents

1 Introduction 3

2 Theoretical Foundations 3

2.1 LASSO Problem 3

2.2 Frank Wolfe Algorithm 4

2.3 Pairwise Frank Wolfe Algorithm 4

2.4 The Duality Gap 5

2.5 Optimizing over Vectors 5

2.6 Armijo Line Search: 5

3 Datasets 6

3.1 Concrete Strength Dataset 6

3.2 Boston Housing Dataset 6

4 Implementation and Results 7

4.1 Concrete Strength Dataset 7

4.2 Boston Housing Dataset 9

5 Conclusion 10

6 References 11

1 Introduction

This project studies the application of the Frank-Wolfe (FW) algorithm together with its variant, Pairwise Frank-Wolfe (PFW), in minimizing the LASSO (Least Absolute Shrinkage and Selection Operator) regression problem. Convergence behavior and optimization efficiency of these algorithms are evaluated using two real-world datasets, which include Concrete Compressive Strength dataset and Boston Housing dataset. The LASSO problem, to be able to explicitly recover sparse solution in high-dimensional data, is an ideal method for feature selection. The results demonstrate that although FW algorithm is fast and decreases the duality gap and objective function rapidly, it tends to plateau early, resulting in suboptimal solutions. On the other hand, the PFW algorithm, although being more time consuming, produces more optimal solutions than the other algorithm by making more improvements constantly. It can be observed that the discrepancies in the results of FW and PFW are greater in Concrete dataset than in Boston Housing dataset, which demonstrate how the characteristics of a dataset can influence the performance of these optimization algorithms.

2 Theoretical Foundations

2.1 LASSO Problem

In this project, the LASSO problem is used to assess the implementation of the FW and PFW algorithms. The problem is characterized to find a sparse solution for high-dimensional data, ensuring that the solution remains within the ℓ_1-norm constraint [1].

To implement the LASSO problem, the objective function is defined as:

$$\min_{x \in R^n} f(x) := \|Ax - b\|^2$$

$$s.t. \|x\|_1 \leq \tau$$

Given the training set:

$$T = \{(r_i, b_i) \in \mathbb{R}^n \times \mathbb{R} : i \, \epsilon [1:m]\}$$

r_i^T are the rows of an m × n matrix A and parameter τ controls the amount of shrinkage that is applied to model. The feasible set is:

$$C = \{x \in \mathbb{R}^n : \|x\|_1 \leq \tau\} = conv\{\mp \tau e_i : i \in [1:n]\}$$

We then compute the gradient, and utilize the Linear Minimization Oracle (LMO) to find the appropriate search directions:

$$LMO_C(\nabla f(x_k)) = sign(-\nabla_{i_k} f(x_k)) \cdot \tau e_{i_k},$$

$$i_k \in \arg\max_i |\nabla_i f(x_k)|$$

2.2 Frank Wolfe Algorithm

The Frank Wolf algorithm can be used to address convex optimization problems with constrained feasible sets. Frank Wolfe, a projection-free algorithm, uses LMO to solve the problem which is less computationally expensive than the projected gradient method. Below, the Pseudocode of the algorithm is presented:

Algorithm 1 Frank-Wolfe method

Choose a point $x_0 \in C$
For k = 0, …
    If x_k satisfies some specific condition (duality gap), then STOP
    Compute $s_k \in LMO_C(\nabla f(x_k))$
    Set $d_k^{FW} = s_k - x_k$
    Set $x_{k+1} = x_k + \alpha_k d_k^{FW}$, with $\alpha_k \in (0,1]$ a suitably chosen stepsize
End for

In this project, which we are dealing with the LASSO problem, our objective function is $f(x) := \|Ax - b\|^2$ and we use $LMO_C\big(\nabla f(x_k)\big) = sign\big(-\nabla_{i_k} f(x_k)\big) \cdot \tau e_{i_k}$ to compute s_k. Moreover, the duality gap and number of iterations are our stopping criteria.

2.3 Pairwise Frank Wolfe Algorithm

The Pairwise Frank Wolfe algorithm is a variant of the classical Frank Wolfe Algorithm, which was introduced by Mitchell and colleagues in 1974. This alternative aims at improving optimization by concentrating on shifting weight mass between two chosen atoms only as opposed to altering all active weights. PFW algorithm performs one move of weight from an "away" atom v_t to a Frank Wolfe atom s_t, while keeping other weights unchanged [2]. The Pseudocode of the algorithm is given below:

Algorithm 2 Pairwise Frank-Wolfe method

Let $x^{(0)} \in \mathcal{A}$ and $\mathcal{S}^{(0)} := \{x^{(0)}\}$
For t = 0 to T do
    Let $s_t := LMO_{\mathcal{A}}\big(\nabla f(x^t)\big)$ and $d_t^{FW} := s_t - x^{(t)}$
    Let $v_t \epsilon\, argmax\langle \nabla f(x^t), v\rangle, v \epsilon \mathcal{S}^{(t)}$ and $d_t^A = x^{(t)} - v_t$
    If $g_t^{FW} := \langle -\nabla f(x^t), d_t^{FW}\rangle \leq \epsilon$ then return x_t
    Else
        $d_t = d_t^{PFW} = d_t^{FW} + d_t^A = s_t - v_t$, $\gamma_{max} := |\alpha_{v_t}|$
        Determine $\gamma_t \in [0, \gamma_{max}]$ a suitably chosen stepsize
        Update $x^{(t+1)} := x^t + \gamma_t d_t$
        Update $\mathcal{S}^{(t+1)} := \left\{ v \in \mathcal{A} \middle| |\alpha_v^{(t+1)}| > 0 \right\}$
    End
End

Compared to the classical Frank Wolfe algorithm which uniformly shrinks all active weights at each iteration, the PFW algorithm's selective adjustment makes it more efficient in certain cases especially for sparse solutions. Also, since our feasible set in this project can contain negative values as well, a few considerations should be taken regarding the implementation of general framework of PFW. A positive value should be considered for γ_{max} and the active set in each iteration should be chosen from non-zero weights associated with the atoms.

2.4 The Duality Gap

For any constrained convex optimization problem of the form $\min_{x \in \mathcal{D}} f(x)$, where the objective function f is convex and the domain $\mathcal{D}$ is a compact convex subset of any vector space and a feasible point $x \in \mathcal{D}$, the duality gap can be defined as:

$$g(x) = \max_{s \in \mathcal{D}} \nabla f(x)^T (x - s) = \max_{s \in \mathcal{D}} -\nabla f(x)^T (s - x)$$

The objective function f is convex so we have that $g(x) \geq f(x) - f(x^*), \ \forall x \in \mathcal{D}$. Since x^* is unknown and so we do not have the value of $f(x) - f(x^*)$, the duality gap can be considered as a good optimality measure for a stopping criterion. This duality gap is easy to compute and we have the $g(x)$ for free at each iteration of the frank wolfe algorithm [3].

2.5 Optimizing over Vectors

In this project, the focus is on the optimization of sparse vectors over the ℓ_1-ball constraint, which is a basic approach in high dimensional optimization tasks like LASSO regression. Sparsity is a very important feature of the ℓ_1-ball constraint as it enhances efficient feature selection and model simplicity.

The convex hull of the signed unit basis vectors $\mathcal{A} = \{\mp e_i | i \in [n]\}$ in $\mathbb{R}^n$ is the unit ball of the ℓ_1-norm. The ℓ_1-ball in $\mathbb{R}^n$ is defined as:

$$B_1(\tau) = \{x \in \mathbb{R}^n : \|x\|_1 \leq \tau\}$$

Where $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ is the ℓ_1-norm of x. The ℓ_1-ball is commonly used in optimization to enforce sparsity. Frank-Wolfe-type greedy algorithms are well-suited for finding sparse vectors that optimize a convex function over such domains.

2.6 Armijo Line Search:

Armijo line search is a popular rule for determining the stepsize. This method shrinks the step size iteratively and guarantees a sufficient reduction of the objective function. In this method, we fix the parameters $\delta \in (0,1)$ and $\gamma \in (0,1/2)$ and given a starting stepsize $\triangle_k$, the steps $\alpha = \delta^m \triangle_k$ with $m = \{0, 1, 2, \dots\}$ are tried until the sufficient decrease inequality:

$$f(x_k + \alpha d_k) \leq f(x_k) + \gamma \alpha \nabla f(x_k)^T d_k$$

holds and so set $\alpha_k = \alpha$.

3 Datasets

In this project, two real world datasets have been utilized which their description and the reason for their selection are explained in sections 3.1 and 3.2.

3.1 Concrete Strength Dataset

Concrete is a fundamental material in civil engineering. Civil engineering uses concrete as a core material which is necessary for strong and long-lasting structures. The various factors that affect the compressive strength of concrete and are considered in this measurement include the age of the concrete in question and its composition. This dataset is employed when forecasting the compressive strength of the cement using these variables [4]. It has 1030 instances, each consisting of eight input variables and one output variable. All variables are numeric, with no missing values at all. These variables are:

- **Cement:** Amount of cement in the mixture (kg/m³)
- **Blast Furnace Slag:** Amount of blast furnace slag in the mixture (kg/m³)
- **Fly Ash:** Amount of fly ash in the mixture (kg/m³)
- **Water:** Amount of water in the mixture (kg/m³)
- **Superplasticizer:** Amount of superplasticizer in the mixture (kg/m³)
- **Coarse Aggregate:** Amount of coarse aggregate in the mixture (kg/m³)
- **Fine Aggregate:** Amount of fine aggregate in the mixture (kg/m³)
- **Age:** Age of the concrete (days)
- **Compressive Strength:** Compressive strength of the concrete (MPa)

This dataset is suitable for the project because the relationship between the target variable, the compressive strength, and its predictors is most likely nonlinear, which will give the optimization algorithms such as Frank-Wolfe and Pairwise Frank-Wolfe a good test case. This is in line with the abilities of such algorithms to work with big data and especially high dimensionality since it has eight input variables. Also, the absence of missing values in the dataset minimizes the disruptions of processing the data and the accuracy of the results obtained. The purpose of this work is to contribute to the improvement of the predictive modeling in civil engineering with focus on the practical cases of material quality evaluation.

3.2 Boston Housing Dataset

The Boston Housing Dataset, available from the StatLib library of Carnegie Mellon University, has been widely utilized for testing algorithms in regression analytic projects. This data was originally created by Harrison and Rubinfeld to be used in their study of hedonic pricing and the demand for clean air [5].

This dataset consists of 506 instances and 14 attributes. There are 13 continuous attributes and one binary attribute. The descriptions of variables are:

- **CRIM:** Per capita crime rate by town
- **ZN:** Proportion of residential land zoned for lots over 25,000 sq.ft.

- **INDUS:** Proportion of non-retail business acres per town
- **CHAS:** Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- **NOX:** Nitric oxides concentration (parts per 10 million)
- **RM:** Average number of rooms per dwelling
- **AGE:** Proportion of owner-occupied units built prior to 1940
- **DIS:** Weighted distances to five Boston employment centers
- **RAD:** Index of accessibility to radial highways
- **TAX:** Full-value property-tax rate per $10,000
- **PTRATIO:** Pupil-teacher ratio by town
- **B:** 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- **LSTAT:** Percentage of lower status of the population
- **MEDV:** Median value of owner-occupied homes in $1000's

Containing 13 predictor variables, makes it a high dimensional data which is well suited for LASSO problem. Also, the diverse range of attributes allows LASSO to perform feature selection effectively. This also aligns closely with FW and PFWs objectives of efficiently solving convex optimization challenges. Moreover, the practical relevance of the dataset guarantees that the models and optimization methods developed can be applied in real world situations offering a context, for evaluating LASSO, FW and PFW performance. Also, the lack of missing data simplifies the use of LASSO and optimization algorithms ensuring that the primary focus remains on optimizing processes themselves.

4 Implementation and Results

For both datasets, the following parameters were considered: number of iterations was set to 1000, shrinkage control parameter was set to 1 and threshold for duality gap, our criterion for approximation quality, was set to 10e-8. Moreover, Armijo Rule was employed as a line search to make the code and implementation more efficient.

Compressive Strength for Concrete Strength Dataset and Median value of owner-occupied homes were considered as target variables. Also, all variables are normalized to the range [0, 1] for consistency.

4.1 Concrete Strength Dataset

Plots related to logarithm of Gap and logarithm of objective function vs number of iteration for concrete data are shown in figure 1.

From the plot of log (Gap) vs iteration, it can be interpreted that the frank-wolfe algorithm shows a more rapid initial decrease in the duality gap. However, with respect to pairwise frank-wolfe, it struggles to make substantial progress in reducing the duality gap further after the initial iterations. The slower initial decrease in pairwise frank-wolfe might be due to the algorithm taking smaller, more refined steps by adjusting weights between pairs of vertices. The PFW algorithm continues to reduce the duality gap significantly more than FW and once it reaches a certain level of duality

gap, it also flattens out, indicating convergence. Overall, it can be concluded that FW efficiency diminishes rapidly, while, PFW proves to be more effective in the long run, achieving a much lower duality gap which indicates a better optimization performance.

Plot of log (f(x)) vs iteration shows that both FW and PFW show a rapid initial decrease in the objective function value, indicating that they quickly find a direction that significantly reduces the objective function in the early iterations. However, the PFW consistently decreases the objective function value more than FW, achieving a lower value within approximately same number of iterations which indicates that PFW is more effective at exploring the feasible region and optimizing the objective function.

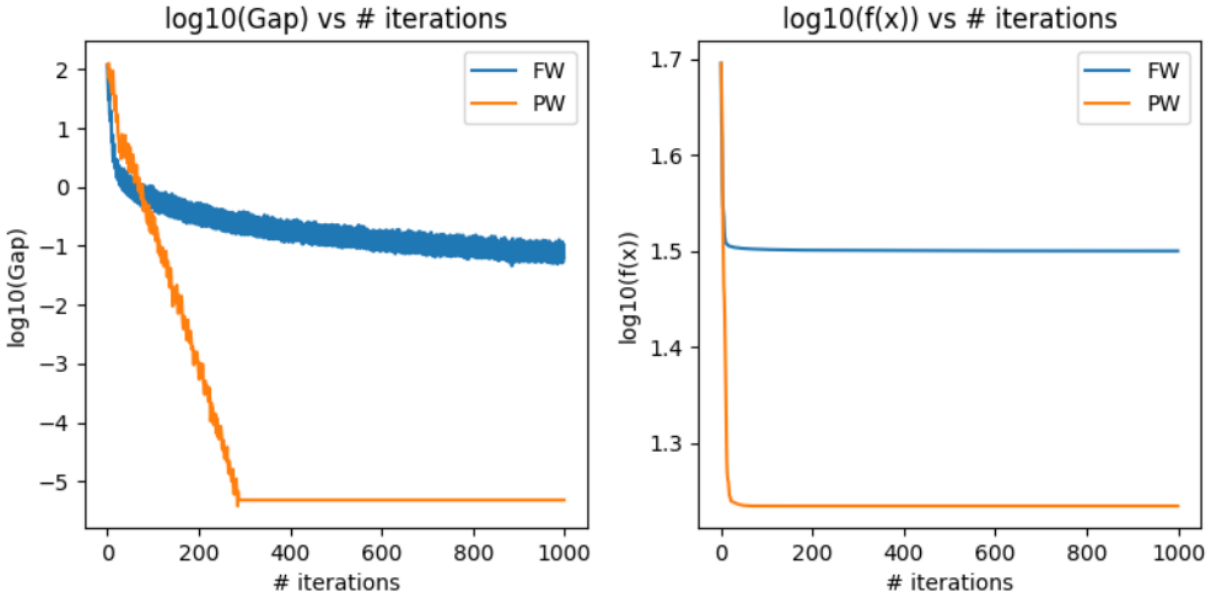

Figure 1. Results of Concrete dataset based on iterations

Plots related to logarithm of Gap and logarithm of objective function vs CPU time for concrete data are shown in figure 2.

It can be interpreted from the plot of log (Gap) vs CPU time that both FW and PFW can rapidly find a feasible direction which significantly reduces the duality gap within a short amount of CPU time. The FW stops with fewer CPU resources and thus more computationally efficient. On the other hand, PFW, although slower per iteration, is more effective in reducing the duality gap to a much lower value.

Based on the plot of log (f(x)) vs CPU time, both FW and PFW show a rapid initial decrease in the objective function value and then they both reach a suboptimal solution and then make minimal further progress. However, like the previous plot, FW shows being quicker to finish while PFW takes longer in terms of CPU time to reach its lowest point. It decreases the objective function value more substantially and continues to refine the solution, indicating that it achieves a more optimal solution over time.

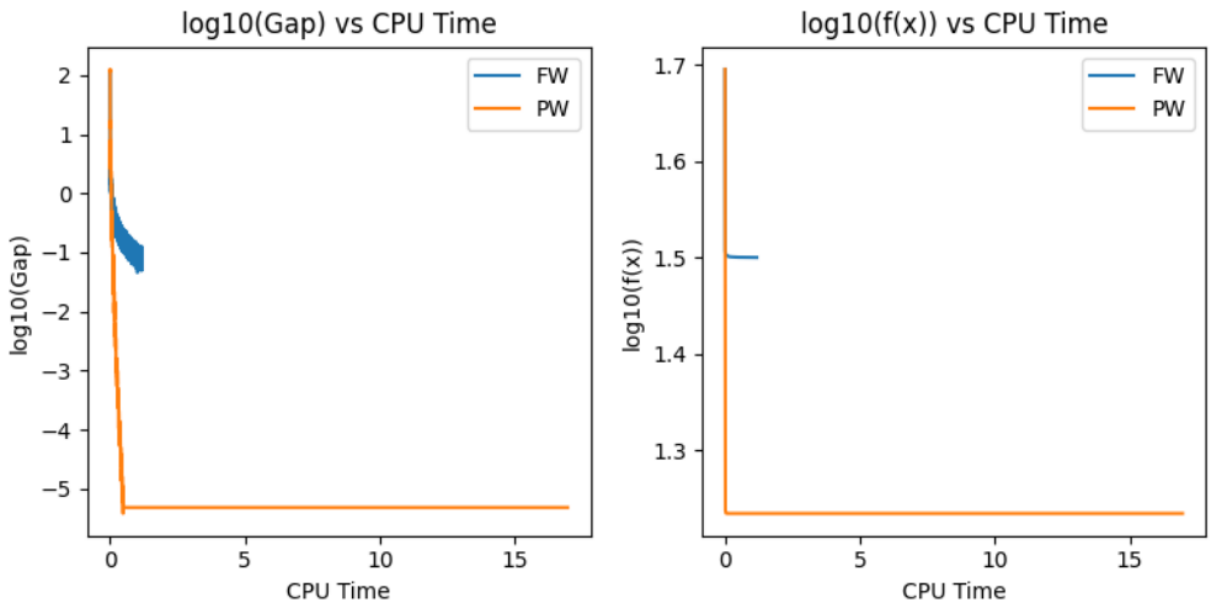

Figure 2. Results of Concrete dataset based on CPU time

4.2 Boston Housing Dataset

Plots related to logarithm of Gap and logarithm of objective function vs number of iterations for Boston Housing data are shown in figure 3.

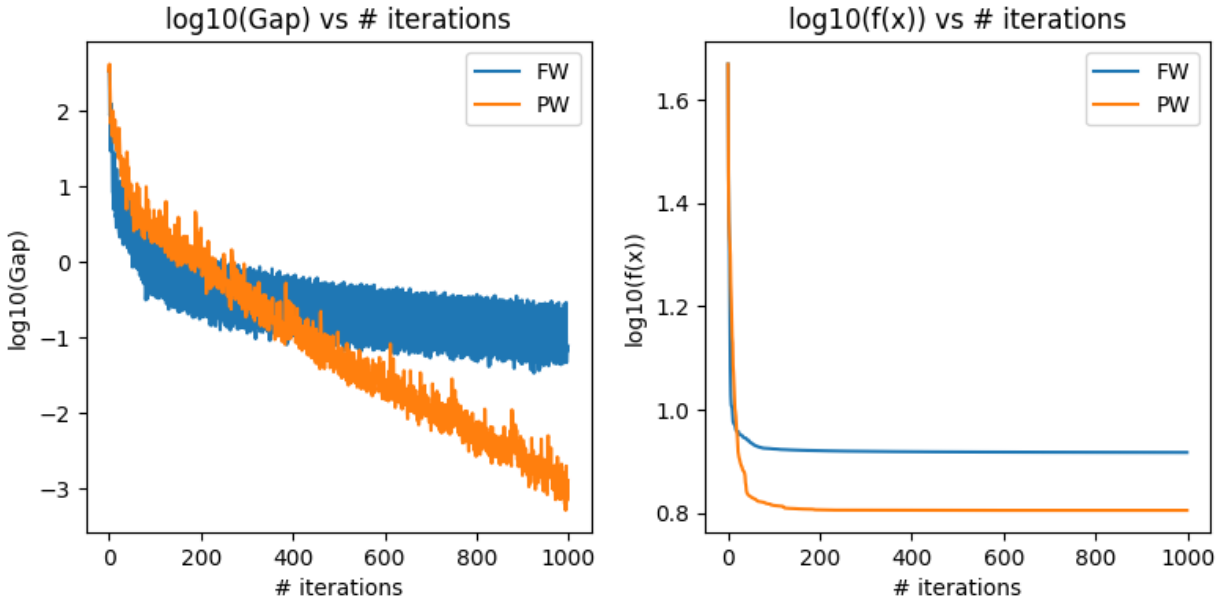

Figure 3. Results of Boston Housing dataset based on iterations

The results show almost similar trend that we had for Concrete data but here the difference between FW and PFW in finding the lower value for both gap duality and objective function is less. The reason can be due to several facts such as higher dimensionality of Boston Housing data. For

concrete data, the lower-dimensional space with more instances can cause the ability of PFW to make fine-grained adjustments which are significantly better in sparsity and feature selection.

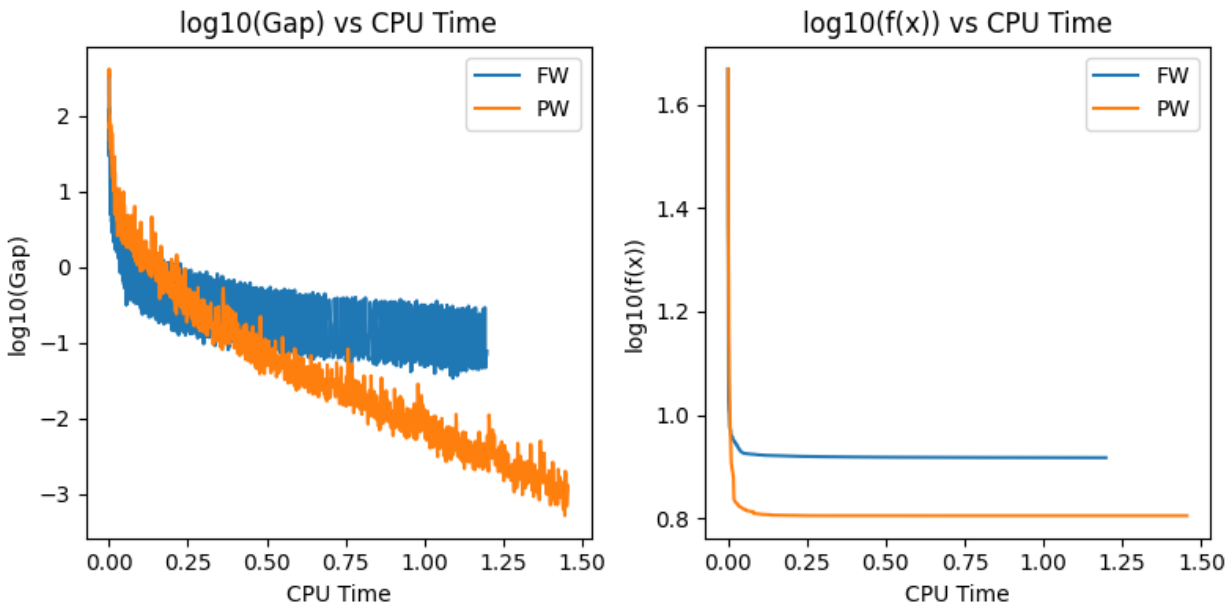

Figure 4. Results of Boston Housing dataset based on CPU time

Also, from figure 4 it can be seen that similarly to figure 2, FW algorithm is faster, while PFW performs better in case of finding the lowest point. But the difference between the FW and PFW performance is less than the case for Concrete data in both the time of CPU and the lowest point they find.

5 Conclusion

In this project, we used two real world datasets namely Concrete Strength Data and Boston Housing Data to compare the performance of the Frank-Wolfe and the Pairwise Frank-Wolfe algorithms on the LASSO problem. We investigated these algorithms in terms of their performance, the convergence rate, computational complexity, and the quality of the solutions obtained.

For the Concrete data, we had a clearer distinction between the FW and PFW algorithms results. Both algorithms demonstrated sharp initial decrease in the duality gap and the objective function. However, FW plateaued quickly, and it managed to achieve a smaller change in both two metrics. On the other hand, PFW was rapidly decreasing and constantly had much lower values than those of the other two indicators.

In terms of CPU time, FW was proved to be rather efficient as it performed the iterations in significantly less CPU time. PFW required more iterations than the other algorithms but reached a better solution as it had a lower final duality gap and objective function values.

We had almost the same results for Boston Housing data, but the differences between FW and PFW performance were less noticeable. This can be because of the higher dimensionality and the lower number of instances of the Boston Housing data that made the optimization landscape more complex.

In terms of algorithm efficiency, FW demonstrated computational efficiency by quickly reducing the duality gap and objective function values. However, it often converged to suboptimal solutions.

In terms of algorithm effectiveness, although PFW was more computationally intensive, it consistently achieved more optimal solutions. Its ability to continue making improvements over a longer CPU time was evident, especially in the Concrete Data.

To summarize, it can be concluded that the FW algorithm is more efficient in terms of CPU time which might be important for the cases when significant improvements should be obtained quickly. However, it tends to plateau early, resulting in suboptimal solutions in more complex datasets. It is highlighted that the PFW algorithm is better in finding more global optimal solutions especially in less complex optimization landscapes. Although PFW has higher computational cost, it is preferable for applications where the quality of the final solution is more important.

6 References

[1] Bomze, I. M., Rinaldi, F., & Zeffiro, D. (2021). Frank–Wolfe and friends: a journey into projection-free first-order optimization methods. *4OR*, 19, 313–345. https://doi.org/10.1007/s10288-021-00493-y

[2] Lacoste-Julien, S., & Jaggi, M. (2015). On the Global Linear Convergence of Frank-Wolfe Optimization Variants.

[3] Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In Proceedings of the 30th International Conference on Machine Learning (ICML), Atlanta, Georgia, USA.

[4] Maajdl. (n.d.). Yeh Concrete Data. Retrieved from Kaggle: https://www.kaggle.com/datasets/maajdl/yeh-concret-data

[5] Dua, D., & Graff, C. (2019). UCI Machine Learning Repository: Housing Data Set. Irvine, CA: University of California, School of Information and Computer Science. Retrieved from https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.names and https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data.