The Frank-Wolfe Algorithm and an Experiment on Its Convergence Rates

Gonghan Xu

1 Introduction

The Frank-Wolfe algorithm[2], also known as the conditional gradient method, is an iterative algorithm designed for convex optimization problems with compact, convex feasible sets. Originated in the 1950s by Marguerite Frank and Philip Wolfe, this method provides a unique approach by avoiding direct projections onto the feasible set, which can be computationally expensive or intractable for certain problems. Instead, it relies on linear approximations to the objective function, making it particularly useful for optimization over polyhedra or simple constraint sets. In recent years, with the upsurge of large-scale optimization problems in various fields like machine learning, data science, and operations research, there has been a revived interest in the Frank-Wolfe method. This is due to its simplicity, adaptability, and efficiency, especially in scenarios where other optimization methods struggle.

In this project, we will first review some important theories about the Frank-Wolfe algorithm and then conduct a Python programming experiment to study the convergence rates of this algorithm under different algorithmic variants. This project report is based on the review articles [1], [3], and [4]. The experimental code is available at `https://github.com/kiwi1236/sds631_project.git`.

2 Theory

2.1 Problem setup

The Frank-Wolfe algorithm is intended to solve constrained optimization problems where the feasible set is compact and convex. In this article, we will focus on convex optimization problems so we only discuss the case when the objective function to be minimized is also convex. Then the optimization problem is:

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

where P is a compact convex feasible set and $f(x)$ is a convex function. We also assume that $f(x)$ is differentiable and L-smooth, i.e. its gradient $\nabla f(x)$ is L-Lipschitz.

2.2 The Frank-Wolfe algorithm

The main intuition of the Frank-Wolfe algorithm comes from gradient descent. If we try to solve the optimization problem 1 directly with a vanilla gradient descent algorithm, as given in Algorithm 1 we may encounter a problem in line 4, where the next iterate x_{t+1} can go out of the feasible set P. One idea is to project the updated iterate x_{t+1} back into the feasible set P, which is the core of the projected gradient descent algorithm. However,

Algorithm 1 Vanilla Gradient Descent Algorithm

Initialize iterate $x_0 \in P$
Set some positive step size α and maximum number of iterations N
for $t = 0$ to $N - 1$ **do**
    $x_{t+1} = x_t - \alpha \cdot \nabla f(x_t)$
end for
return x_N

such a projection operation will typically involve another optimization problem, which can be hard to solve. Instead, the Frank-Wolfe algorithm provides an alternative approach. It attempts to obtain the new iterate x_{t+1} such that x_{t+1} automatically falls within P. At each step t, the Frank-Wolfe algorithm introduces an intermediate point $v_t \in P$, which we call the target point. Then $\forall v_t \in P$, the vector $v_t - x_t$ represents a potential update direction. Then we want to find a target point v_t such that $v_t - x_t$ aligns with the negative gradient of $f(x)$, i.e. $-\nabla f(x)$, as much as possible. This is achieved by maximization over a linear objective $v_t = \text{argmax}_{v \in P} \langle -\nabla f(x), v \rangle$, which is equivalent to $v_t = \text{argmin}_{v \in P} \langle \nabla f(x), v \rangle$. Then after obtaining v_t at each step, we can do a regular descent step:

$$x_{t+1} = x_t + \alpha(v_t - x_t) \tag{2}$$

where we move the current iterate x_t towards the target point v_t with step size α. Notice that the update equation 2 can equivalently be written as $x_{t+1} = (1 - \alpha)x_t + \alpha v_t$, which is a convex combination of x_t and v_t. This shows that given $x_t, v_t \in P$, the next iterate x_{t+1} will always fall within the feasible set P because P is convex. Then we can see the Frank-Wolfe algorithm is essentially tweaking the vanilla gradient descent algorithm with the condition that each new iterate x_{t+1} automatically falls within P, hence its alternative name conditional gradient method. Then we put forth the Frank-Wolfe algorithm in Algorithm 2. Notice that in line 5 of Algorithm 2, the step size α_t can be chosen by some rule. Constant

Algorithm 2 Frank-Wolfe Algorithm

1 Initialize iterate $x_0 \in P$
2 Set maximum number of iterations N
3 **for** $t = 0$ to $N - 1$ **do**
4     $v_t = \text{argmin}_{v \in P} \langle \nabla f(x_t), v \rangle$
5   Set step size $\alpha_t \in [0, 1]$ based on some rules
6     $x_{t+1} = (1 - \alpha_t)x_t + \alpha_t v_t$
7 **end for**
8 **return** x_N

step size can be used. Another strategy often used is setting $\alpha_t = \frac{2}{t+2}$, the so-called open loop step size. We will discuss the choice of step size later.

2.3 Properties

Now that we have introduced the basics of the Frank-Wolfe algorithm, we will take a closer look and discuss some of its important properties.

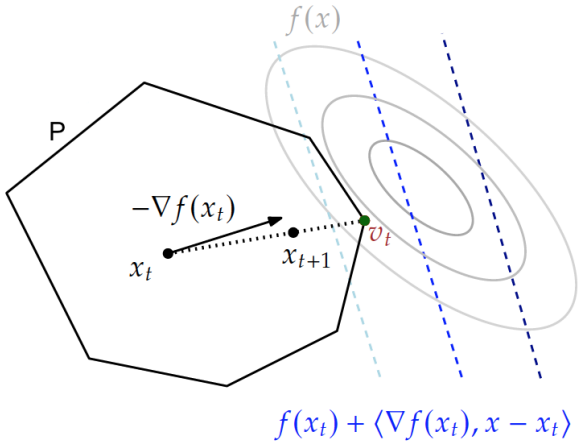

Figure 1: Illustration of the Frank-Wolfe algorithm

First, from Figure 1, we can see that the target point v_t at step t is the solution of minimizing the linear approximation of the objective function $f(x)$ at x_t. Then since $f(x)$ is convex, the linear approximation $f(x_t) + \nabla f(x_t)(x - x_t)$ always lies below the graph of $f(x)$. This property is valid for any $x_t \in P$, so if we set $x_t = x^*$, where x^* is the optimal solution of optimization problem 1, then we have

$$f(x) - f(x^*) \leqslant \langle \nabla f(x), x - x^* \rangle \leqslant \max_{v \in P} \langle \nabla f(x), x - v \rangle \tag{3}$$

where the second inequality comes trivially from the definition of max. The left hand side (LHS) of Equation 3 gives the suboptimality for the current iterate x, and the right hand side (RHS) gives an upper bound of this suboptimality that is ignorant of x^*. The LHS, the middle part, and the RHS are also referred to as the primal gap at x, the dual gap at x, and the Frank-Wolfe gap at x, respectively[4]. Since the Frank-Wolfe gap is ignorant of the optimal solution x^*, it can be used as a certificate for the suboptimality quality. Given an iterate x_t, the Frank-Wolfe gap $g(x_t)$ at x_t is

$$\begin{aligned} g(x_t) &\equiv \max_{v \in P} \langle \nabla f(x_t), x_t - v \rangle \\ &= \langle \nabla f(x_t), x_t \rangle - \min_{v \in P} \langle \nabla f(x_t), v \rangle \\ &= \langle \nabla f(x_t), x_t \rangle - \langle \nabla f(x_t), v_t \rangle \end{aligned} \tag{4}$$

From Equation 4, we can see the Frank-Wolfe gap is essentially obtained for free in line 4 of the Frank Wolfe algorithm. From the form of the Frank-Wolfe gap, we can relate it to the first-order optimality condition for constrained convex optimization problems. The first-order optimality condition states that for $\forall\, v \in P$, $\langle \nabla f(x^*), x^* - v \rangle \leqslant 0$. And when $v = x^*$, we have $\langle \nabla f(x^*), x^* - x^* \rangle = 0$. So the Frank-Wolfe gap at x^*, i.e. $\max_{v \in P} \langle \nabla f(x^*), x^* - v \rangle$, is exactly 0. Notice that this implies that the minimum Frank-Wolfe gap $\min_{x \in P} g(x)$ is zero, because a negative Frank-Wolfe gap $g(\tilde{x})$ means $\tilde{x} = x^*$ due to the first-order optimality condition, but we have just proved that $g(x^*) = 0$, thus a contradiction.

2.4 Convergence Analysis

First, we will prove convergence of the Frank-Wolfe algorithm. As mentioned in the beginning, the objective function $f(x)$ is assumed to be L-smooth in our scope of discussion. Then L-smoothness implies a quadratic upper bound. Specifically, for $\forall x, y \in P$,

$$f(y) - f(x) \leqslant \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|_2^2 \tag{5}$$

where we assumed the Lipschitz upper bound L is defined with respect to L^2 norm. Then choosing $x = x_t$, $y = x_{t+1}$, and plugging in the Frank-Wolfe update relation $x_{t+1} = x_t + \alpha_t(v_t - x_t)$ we can obtain

$$f(x_t) - f(x_{t+1}) \geqslant \alpha_t \langle \nabla f(x_t), x_t - v_t \rangle - \alpha_t^2 \frac{L}{2} \|x_t - v_t\|_2^2 \tag{6}$$

Using convexity of $f(x)$, we have $\langle \nabla f(x_t), x_t - v_t \rangle \geqslant f(x_t) - f(v_t) \geqslant f(x_t) - f(x^*)$. Then Inequality 6 can be used to derive

$$f(x_t) - f(x_{t+1}) \geqslant \alpha_t (f(x_t) - f(x^*)) - \alpha_t^2 \frac{L}{2} \|x_t - v_t\|_2^2 \tag{7}$$

Then subtracting $f(x^*)$ from both sides and bounding $\|x_t - v_t\|_2$ by the diameter D of the compact feasible set P, we can rearrange Inequality 7 to obtain

$$f(x_{t+1}) - f(x^*) \leqslant (1 - \alpha_t)[f(x_t) - f(x^*)] + \alpha_t^2 \frac{LD^2}{2} \tag{8}$$

Note that Inequality 8 relates the suboptimality at x_{t+1} with that at x_t. Let $\alpha_t = \frac{2}{t+2}$ as pertaining to the open loop step size strategy, then when $t = 0$, we have $\alpha_0 = 1$ and $1 - \alpha_0 = 0$. So $f(x_1) - f(x^*) \leqslant \frac{LD^2}{2} \leqslant \frac{2LD^2}{(1+2)}$. Now we will prove the following theorem by induction:

Theorem 1 (Suboptimality convergence of the Frank-Wolfe algorithm) *Let f be an L-smooth convex function and let P be a compact convex set with diameter D, then with open loop step size $\alpha_t = \frac{2}{t+2}$, the suboptimality of the iterates in Algorithm 2 satisfies*

$$f(x_t) - f(x^*) \leqslant \frac{2LD^2}{t+2} \tag{9}$$

Proof *The base case* $f(x_1) - f(x^*) \leqslant \frac{2LD^2}{(1+2)}$ *is already satisfied. Then suppose Theorem 1 is satisfied for* $1, 2, ..., t$*, then we have*

$$\begin{aligned}
f(x_{t+1}) - f(x^*) &\leqslant (1-\alpha_t)[f(x_t) - f(x^*)] + \alpha_t^2 \frac{LD^2}{2} && \textit{(Inequality 8)} \\
&= \frac{t}{t+2}[f(x_t) - f(x^*)] + \frac{4}{(t+2)^2}\frac{LD^2}{2} && \textit{(Using open loop step size)} \\
&\leqslant \frac{t}{t+2}\frac{2LD^2}{t+2} + \frac{4}{(t+2)^2}\frac{LD^2}{2} && \textit{(Using inductive hypothesis)} \\
&= \frac{2LD^2}{t+3}[\frac{(t+3)(t+1)}{(t+2)^2}] && (\frac{(t+3)(t+1)}{(t+2)^2} \leqslant 1) \\
&\leqslant \frac{2LD^2}{(t+1)+2}
\end{aligned} \tag{10}$$

then the inductive case is also satisfied, hence the proof. ■

From Theorem 1, we can see that the suboptimality $f(x_t) - f(x^*)$ for the Frank-Wolfe algorithm converges to 0 no slower than a rate of $O(\frac{1}{t})$. So we are guaranteed that for $\forall\, t \geqslant \frac{2LD^2}{\epsilon}$, $f(x_t) - f(x^*) \leqslant \epsilon$.

2.5 Choice of Step Size

Now we discuss the choice of step sizes. We have mentioned the options of choosing a constant step size and an open loop step size. Constant step size may be desired for some scenarios, but in practice it can often lead to slow convergence rate and even non-convergence. The open loop step size, $\alpha_t = \frac{2}{t+2}$ has two major benefits. First, it guarantees convergence based on Theorem 1. Second, it is very easy to implement because it does not depend on any other parameters other than the current number of iterations t. This step size choice usually has faster convergence rate than constant step size in practice.

Besides the aforementioned step size choices that are relatively simple, we introduce two other step size rules that operate more strategically. The first one is called a short step stepsize. From Inequality 6, we can see that the amount of descent at step t is lower-bounded by $\alpha_t\langle\nabla f(x_t), x_t - v_t\rangle - \alpha_t^2 \frac{L}{2}\left\|x_t - v_t\right\|_2^2$. Then we can view this lower bound on the descent amount as a quadratic function of α_t and maximize this quadratic function with respect to α. This leads to the optimal $\alpha_t = \frac{\langle\nabla f(x_t), x_t - v_t\rangle}{L\|x_t - v_t\|_2^2}$. Notice that we require $\alpha \in [0, 1]$ so that the next update x_{t+1} will fall within the feasible set P. So the short step rule is:

$$\alpha_t = \min\{\frac{\langle\nabla f(x_t),\, x_t - v_t\rangle}{L\left\|x_t - v_t\right\|_2^2}, 1\} \tag{11}$$

Notice that $\alpha_t \geqslant 0$ always hold because the Frank-Wolfe gap $g(x_t) = \langle\nabla f(x_t),\, x_t - v_t\rangle = \max_{v\in P}\langle\nabla f(x_t),\, x_t - v\rangle$ is always nonnegative. In practice, this short step rule sometimes leads to faster convergence rate than the open loop rule, but it relies on the knowledge of the Lipschitz constant L.

Another strategic rule to choose the step size is based on line search. Recall that the next iterate $x_{t+1} = x_t + \alpha_t(v_t - x_t)$. Then we can pick $\alpha_t = \text{argmin}_{\alpha_t \in [0,1]} f(x_t + \alpha_t(v_t - x_t))$. The line search rule often allows for more progress in the descent step, but it requires solving an extra optimization problem at each iteration. This approach can be very useful if a closed-form solution exists for this 1-dimensional optimization problem.

3 Experiments on Frank-Wolfe Convergence Rates with Different Step Size Rules

In this section, we will demonstrate the convergence rates of the Frank-Wolfe algorithm on the Lasso problem, using different rules of choosing the step size. We use Python and the CVXPY package to conduct our experiment. The experiment code is available at: `https://github.com/kiwi1236/sds631_project.git`

The problem setup for our experiment is

$$\min_{\|\beta\|_1 \leqslant \lambda} \|X\beta - y\|_2^2 \tag{12}$$

In our experiment, $X \in \mathbb{R}^{100\times 10}$, $y \in \mathbb{R}^{100}$, and $\beta \in \mathbb{R}^{10}$. We set $\lambda = 10$. Each entry of X was initialized randomly by a standard normal distribution. We also initialize an intermediate constant $\beta_0 \in \mathbb{R}^{10}$ whose entries were uniformly sampled from $[-10, 10)$. Then we set $y = X\beta_0$.

In our experiment, we first solved for the optimal solution x^* of this problem using CVXPY, then we ran the implemented Frank-Wolfe algorithm with four different step size rules (constant, open loop, short step, line search) and obtained four different sequences of the suboptimality $f(x_t) - f(x^*)$ over the iteration steps. We then plotted the four suboptimality sequences on the same plot along with the upper bound for suboptimality given in Theorem 1. We also fitted the open step suboptimality sequence with two curves (a power law $\frac{C}{t^p}$ curve and a $\frac{C}{t+2}$ curve). The core implementation of the Frank-Wolfe algorithm is shown in the code snippet below. Note that different step size rules can be chosen based on the values of the variables ol (for open loop), adapt (for short step) and ls (for line search). The default step size rule is constant step size of 0.001. The Lipschitz constant L was computed using the maximum eigenvalue of the Hessian of the objective function, $2X^\top X$. The diameter D of the feasible set P is just 2λ for the Lasso problem. The linear optimization problem in the Frank-Wolfe update was solved by CVXPY.

```
# Compute the gradient of the objective function
gradient = -2 * X.T.dot(y - X.dot(beta))

# Obatain the target point
target = update_direction_lasso(gradient, lambda_)

if ol:
    alpha = 2 / (i + 3)
elif adapt:
```

```
        alpha = adapt_alpha(beta, gradient, target, L)
    elif ls:
        alpha = ls_alpha(beta, target, X, y)

    beta = (1-alpha) * beta + alpha * target
```

The experimental results are shown in Figure 2. From Figure 2, we can see that for this

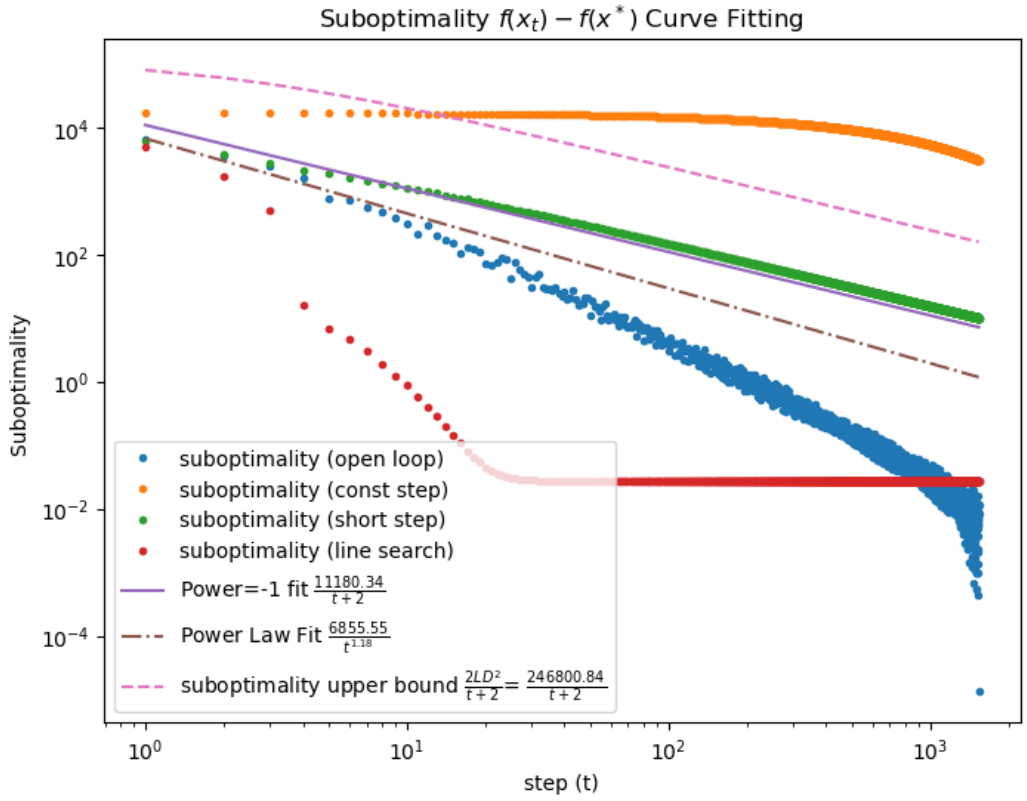

Figure 2: The suboptimality convergence results for different step size rules

particular problem, the suboptimality for the line search rule initially converges very quickly, but it seems to get stuck after 20 iterations. For the Lasso problem, α_t for the line search rule has a closed-form solution $\min\{\frac{[X(v_t-\beta_t)]^\top(y-X\beta_t)}{\|X(v_t-\beta_t)\|_2^2}, 1\}$. In our experiment, we found that the line search rule was not very numerically stable and we had to use $(1-10^{-5})$ rather than 1 in the previous min function to make the next iterate x_{t+1} fall within the feasible set P. This might be the reason why the suboptimalities for line search stop converging further after about 20 iterations.

Compared to the line search rule, the open loop rule converges much more slowly in the beginning but it catches up after about 1000 iterations. The power law curve that we fitted on the open loop suboptimalities has a fitted power of -1.18, which is close to the -1 power for the suboptimality upper bound. Note that Figure 2 is displayed in log-log scale, so the

power law and the $\frac{C}{t+2}$ fitting curves do not appear perfectly aligned with the open loop suboptimality curve.

Compared to the open loop rule, the short step loop performed worse. This is likely due to the fact that the suboptimality upper bound $\frac{2LD^2}{t+2}$ is much larger than the actual open loop suboptimalities and the short step rule was designed to minimize the quadratic upper bound at each iterate. As a result, the short step rule did not show better performance than the open loop rule. This fact is also alluded to by the closeness between slope of the short step optimality curve and that of the optimality upper bound.

Lastly, we can see that the constant step size rule performed the worst, which is not surprising. Overall, the suboptimality convergence phenomenon showcased in this experiment aligns with the theoretical results well.

4 Conclusion

In this project, we introduced the Frank-Wolfe algorithm and its properties. We analyzed its convergence rates under the open loop step size rule and also discussed other more sophisticated rules for choosing the Frank-Wolfe step size. We then conducted an experiment on a Lasso optimization problem where we verified the suboptimality convergence rates suggested by the theory. By the conclusion of this project, we achieved a comprehensive understanding of the Frank-Wolfe method, from both theoretical and practical perspectives.

References

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

[2] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

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

[4] S. Pokutta. The frank-wolfe algorithm: a short introduction, 2023.