

Comparison of Frank-Wolfe Variants for White-Box Adversarial Attacks

Tanner Aaron Graves - 2073559 Alessandro Pala - 2107800

June 2024

1 Abstract

With deep neural networks becoming ubiquitous in application, adversarial attacks have received much attention, as it has proved remarkably easy to create adversarial examples- genuine data that undergoes a minimal and unobtrusive corruption process in order to maximally harm the performance of a model. Access to a model's architecture can enable white-box attacks, where gradients of loss with respect to input examples are exploited to create adversarial examples. The requirement that examples be minimally perturbed is a constraint on the optimization. The ability of Frank-Wolfe and variants have gathered much attention for their ability to efficiently create adversarial examples while staying in a constraint set. In the paper we introduce and discuss the application of Frank-Wolfe and two variants to this non-convex constrained optimization problem. Furthermore we discuss popular optimizations and their effect on convergence and attack efficacy, comparing performance on attacks on models trained on MNIST, FashionMNIST, and CFAIR-10 datasets. Finally, there is a discussion of the theoretical underpinnings of each algorithm.

2 Introduction

This is typically stated as the following constrained optimization problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & \|x\|_p \leq \epsilon \end{aligned} \tag{1}$$

In the case of untargeted attacks on a classifier, we perturb an example with the aim it be incorrectly predicted as any other class. Here $f(x)$ is the loss function of the attacked model $-\ell(x, \hat{y})$. In the case of targeted attacks, we aim to maximize the likelihood of another class $y \neq \hat{y}$. The cost then is $f(x) = \ell(x, y)$. We have implemented both targeted and untargeted attacks, and come to focus on targeted attacks as the algorithms are seen to require more iterations

to converge. The L_p constraint $\|x\|_p \leq \epsilon$ directly restricts the size of perturbations made to the example. An inherent problem of DNNs is often attacks can be consistently successful even with very small, even imperceptible ϵ . Different choices of p may be made giving $\|x\|_p = (\sum_i x_i^p)^{1/p}$. Or commonly, as we use here $L_\infty(x) = \max_i |x_i|$. We define the constraint set $\mathcal{M} = \{x : L_p(x) \leq \epsilon\}$. Of particular note is when \mathcal{M} is a polytope, as is the case when $p \in \{1, \infty\}$. This corresponds to models making perturbations that are either sparse or have a maximal disturbance along each element. In these cases, the constraint set can be expressed as the convex combination of a finite set of vertices $\mathcal{M} = \text{Conv}\{\mathcal{A}\}$. This will be of particular relevance in the discussion of Away-step and pairwise variants. Otherwise, \mathcal{A} is taken to be the boundary of \mathcal{M} . The constrained nature of this problem limits the applicability of a method like gradient descent, and requires integrating knowledge of the constraint space for effective optimization. Methods like Fast Signed Gradient attacks and projected gradient descent are popular choices, but either create unsifted adversarial examples or require wasteful projection onto the constraint space.

We explore Frank-Wolfe variants which are well suited to this problem by ensuring feasibility within the constraint set at each iteration with the efficient solving of a Linear Minimization Oracle (LMO).

$$LMO_{\mathcal{A}}(\nabla f(x_t)) \in \arg \min_{x \in \mathcal{A}} \langle \nabla f(x_t), x \rangle$$

The LMO is responsible for solving what is called the "Frank-Wolfe Subproblem" and at each iteration provides an optimal s_t such that updating x_{t+1} to move in the direction of s_t will remain in \mathcal{M} . Given it moves no more than some maximum stepsize. By solving the LMO efficiently, the algorithm ensures that each iteration makes significant progress towards the optimal solution while respecting the constraints, thereby maintaining feasibility and accelerating convergence. Efficient solving of the LMO requires exploiting the structure of the constraint set \mathcal{M} . In the case of the L_∞ norm it has a closed form solution:

$$LMO_{\mathcal{A}} = s_t = -\epsilon \text{sign}(\nabla f(x_t)) + x_0$$

This is clearly of $O(n)$ complexity where n is the number of elements in the gradient. This can be interpreted as defining an attack direction where each element is the maximum allowable perturbation $\pm\epsilon$ according to the gradient. With one iteration, this is exactly the outcome of a Fast Signed Gradient Attack. At each iteration, we can see that the LMO will give a vertex on the boundary of the constraint set \mathcal{M} which was optimally chosen to be as close to the true gradient as possible while permitting subsequent iterations stay within the feasible set.

The constrained nature of the Adversarial Attack problem means that the norm of the gradient $\|\nabla_x f(x)\|$ is not a suitable convergence criterion, as boundary points need not have zero gradient. Instead, we adopt the Frank-Wolfe gap as a measure of both optimality and point feasibility. This gap measures the maximum improvement over the current iteration x_t within the constraints \mathcal{M}

and is defined as:

$$g_t^{\text{FW}} := \max_{s \in \mathcal{M}} \langle x_t - s, \nabla f(x_t) \rangle = \langle -\nabla f(x_t), d_t^{\text{FW}} \rangle$$

The Frank-Wolfe gap g_t^{FW} is always non-negative $g_t^{\text{FW}} \geq 0$ and is zero if and only if x_t is a stationary point. This makes it a useful convergence criterion, particularly since stationary points need not have zero gradients in constrained problems.

The loss functions of Deep Neural Networks (DNNs), which are commonly the subject of adversarial attacks, are highly non-convex. Thus, the convergence of Frank-Wolfe methods in these applications is complicated by the fact that we are not guaranteed to find a global optimum or a successful attack, but rather convergence to a stationary point. It is also worth noting that, in the context of adversarial attacks, the Frank-Wolfe gap serves as an imprecise surrogate for success. In many cases, Frank-Wolfe methods generate successful attacks several iterations before convergence. This is because achieving an incorrect class probability higher than the correct class is often sufficient for a successful attack, whereas convergence indicates the new output class probability has been maximized.

3 Algorithms

3.1 Frank-Wolfe

Algorithm 1 FW for adversarial attacks

Require: maximum iterations T , stepsizes $\{\gamma_t\}$, convergence tolerance δ

Ensure: $y = x^n$

```

1:  $x_0 = x_{\text{ori}}$ 
2: for  $t = 1, \dots, T$  do
3:    $s_t = \arg \min_{x \in \mathcal{M}} \langle x, \nabla f(x_t) \rangle$  ▷ LMO step
4:    $d_t = s_t - x_t$ 
5:    $x_{t+1} = x_t + \gamma_t d_t$ 
6:   if  $\langle d_t, -\nabla f(x) \rangle < \delta$  then return ▷ FW gap convergence criterion
7:   end if
8: end for
```

Observing oscilation in Frank Wolfe convergence is common and consequence of optimal points lying on a face of \mathcal{M} . Since at each iteration the method is moving twords a vetex of polytope \mathcal{M} , in \mathcal{S} , the method "zigzags", moving twords different points in effort to gradually approach the face on which the optimum lies. in the convex case, Frank Wolfe is seen to have linear complexity when optimal point x^* lies in the interior of \mathcal{M} , the oscilation causes sub-linear convergence when x^* on the boundry. We implement varients that aim to address this problem to provide better convergence. The simplest of which

is adding momentum to standard Frank-Wolfe which replaces the gradient in the LMO calculation in line (4) with a momentum term $m_t = \beta m_{t-1} + (1-\beta)\nabla f(x_t)$ and initialize $m_0 = \nabla f(x_0)$. By considering this exponentially weighted average of gradient information, momentum variants are empirically observed to have nicer convergence.

The FW algorithm has a useful interpretation that serves as basis for the following variants: Away-Step FW and Pairwise FW. Namely that at each iteration x_t is perturbed in some direction s_t , making x_{t+1} a convex combination of x_t and s_t . We record these directions in an active set $S_{t+1} = S_t \cup \{s_t\}$ and observe that initially x_0 is a convex combination of active set $S_0 = \{x_0\}$. Then by induction, x_t is a convex combination of directions in S_t , admitting coefficients α such that $\sum \alpha_{s_i} s_i = x_t$. Each iteration of the FW algorithm is seen to increase or introduce the contribution of s_t in the convex combination while shrink the α coefficients of all other vertices, or atoms uniformly. The innovation of the Away-Step and Pairwise variant is to recognize contribution of "bad atoms" can prevent convergence to an optimum on the boundary. These variants more directly diminish such atoms contributions by either taking steps away from selected atoms, or transferring mass between two selected atoms at each iteration as the case with the pairwise variant.

3.2 Away-Step Frank-Wolfe

The away-step FW algorithm (AFW) defines a gap in two possible directions: the typical FW step d_t^{FW} and an away direction $d_t^{\text{A}} = x_t - v_t$ where $v_t \in \arg \max_{v \in S_t} \langle v, \nabla f(x_t) \rangle$. By comparing the gaps associated with the two directions, the algorithm compares the cost of taking a typical FW step to the cost associated with moving away from active atoms in S_t . Should this gap be less than that of moving towards the Frank-Wolfe direction, the influence of the selected away atom $v_t \in S_t$ will be diminished, and even dropped if the corresponding $\alpha_{v_t} \leq 0$. The α updates can be seen to be inverse for that of a normal FW step. Here influence of v_t is distributed uniformly to other active atoms. New coefficients for the convex combination are for v_t , $\alpha_{v_t} := (1 + \gamma_t)\alpha_{v_t} - \gamma_t$, and otherwise $\alpha_{v_{t+1}} := (1 + \gamma_t)\alpha_v$. The algorithm enables a "drop step" where the contribution of an atom is completely removed from the convex combination, alleviating the problem of oscillation when converging to an optimum on the boundary. This algorithm is proved to have linear convergence for convex problems

Algorithm 2 Away-Step FW for Adversarial Attacks

Require: maximum iterations T , stepsizes $\{\gamma_t\}$, convergence tolerance δ , $x_0 \in \mathcal{M}$

- 1: Define $S_0 := \{x_0\}$ with $\alpha_{x_0} = 1$
- 2: **for** $t = 1, \dots, T$ **do**
- 3: $s_t := \arg \min_{x \in \mathcal{M}} \langle x, \nabla f(x_t) \rangle$ ▷ LMO step
- 4: $d_t^{\text{FW}} := s_t - x_t$
- 5: $v_t := \arg \max_{v \in S_t} \langle v, \nabla f(x_t) \rangle$
- 6: $d_t^{\text{A}} := x_t - v_t$
- 7: **if** $\langle d_t^{\text{FW}}, -\nabla f(x) \rangle < \delta$ **then** return x_t ▷ FW gap convergence criterion
- 8: **end if**
- 9: **if then** $\langle d_t^{\text{FW}}, -\nabla f(x) \rangle < \langle d_t^{\text{A}}, -\nabla f(x) \rangle$
- 10: $d_t = d_t^{\text{FW}}, \gamma_{\max} := 1$
- 11: **else**
- 12: $d_t := d_t^{\text{A}}, \gamma_{\max} := \frac{\alpha_{v_t}}{1 - \alpha_{v_t}}$
- 13: **end if**
- 14: $x_{t+1} = x_t + \gamma_t d_t$
- 15: Update α, S_{t+1} s.t. $\langle \alpha, S_{t+1} \rangle = x_{t+1}$ (See below)
- 16: **end for**

3.3 Pairwise Frank-Wolfe

4 Results

Introduce Datasets

4.1 Momentum

4.2 Stepsize

The methods for stepsize were implemented as follows: Lipschitz constant-based (fixed) stepsize, where the stepsize is determined using the Lipschitz constant L with $\gamma_t = \frac{1}{L}$; exact inesearching, which solves the optimization problem $\arg \min_{\gamma} f(x + \gamma d_t)$ where $\gamma \in (0, 1]$; Decaying stepsize, where the stepsize decreases over time according to $\gamma_t = \frac{2}{t+2}$; and Armijo-rule search, which chooses γ_t to satisfy the Armijo rule $f(x + \gamma_t d_t) \leq f(x) + \delta \gamma_t \nabla f(x)^T d_t$, where $\delta \in (0, 1)$ controls the sufficient decrease condition.

4.3 ϵ Choice

Our three different models were pretrained on datasets with differing image scales. To allow for comparison of the sensitivity of each model to attacks with different ϵ , at each iteration we normalize to ensure images x_0 have mean 0.5 and variance 1 to ensure equal relative perturbation of the same ϵ on all three models.

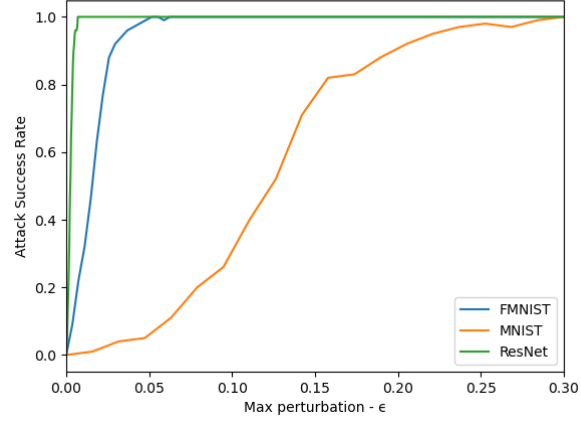


Figure 1: Effect of ϵ on attack success for untargeted FW for each dataset with L_∞ constraint.

As figure 3 shows, the different models have vastly different sensitivities to adversarial attacks. LeNet is the most robust of the three requiring large maximum perturbations for reliable attacks. The larger, more complex models can be reliably attacked with imperceptible examples in the case of ResNet or borderline perceptible in the case of the FMNIST model. Representative attacks are shown in figure 2.

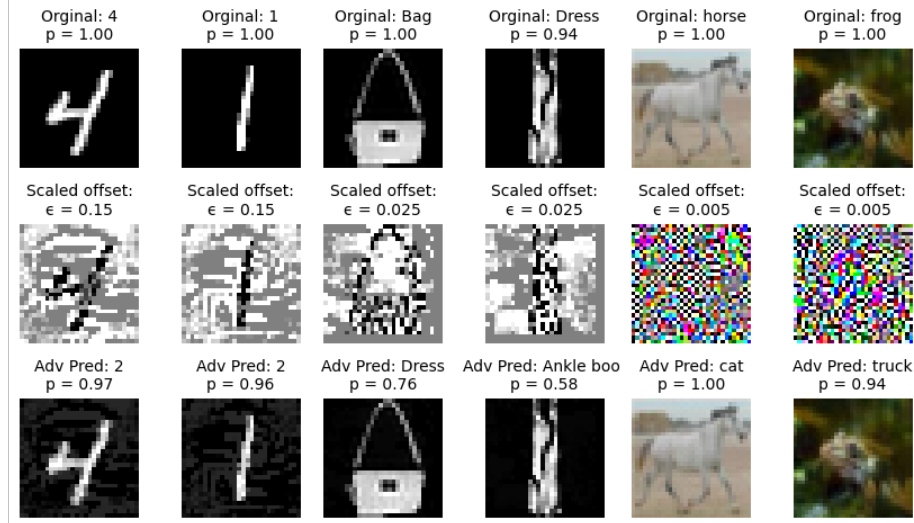


Figure 2: Representative attacks on LeNet, FMNIST, and ResNet 18.

4.4 Targeted vs. Untargeted Attacks

We implemented both untargeted and targeted attacks, requiring defining a custom loss function for the pretrained models. where $f(x) = -\ell(x, y)$ for untargeted attacks, and $f(x) = \ell(x, y_{\text{adv}})$ where y is the true label and y_{adv} the adversarial target class chosen at random to be any class other than y . We find, unsurprisingly, that targeted attacks are invariably a more challenging problem, resulting in lower success rates and higher average iterations for all variants and datasets.

For example, attacks on a set of 100 examples with ResNet18 and $\epsilon = 0.005$ with normal FW algorithm using decaying rule, achieved a success rate of 0.93 requiring 5.44 iterations on average in the untargeted case, but a success rate of 0.63 and 13.26 iterations in the targeted case.

4.5 Variants

Algorithm	MNIST		F-MNIST		CIFAR-10	
	$\epsilon = 0.15$		$\epsilon = 0.025$		$\epsilon = 0.005$	
	ASR (%)	Avg Iter	ASR (%)	Avg Iter	ASR (%)	Avg Iter
FW	85.4	50	86.1	48	84.7	52
AFW	87.3	45	88.0	43	86.9	47
PFW	86.5	47	87.2	44	85.8	49

Table 1: Attack Success Rate (ASR) and Average Iterations of Frank-Wolfe Algorithm and Variants for Targeted Adversarial Attacks on Three Datasets with 20 maximum iterations.

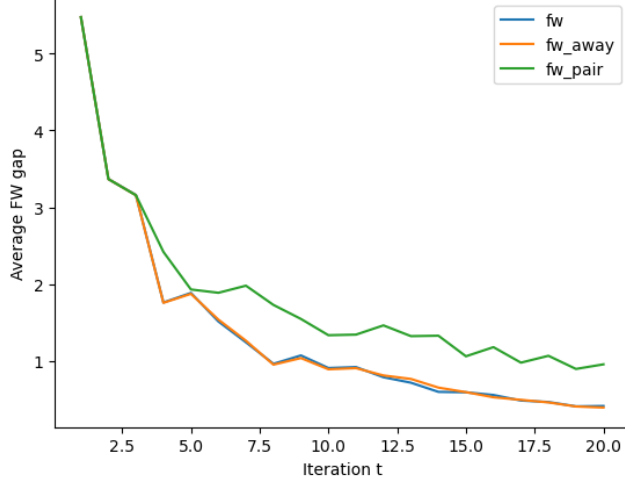


Figure 3: Average g_t^{FW} for each Algorithm by Iteration for Targeted Attacks on ResNet-18 (CIFAR-10). $\epsilon = 0.005$.

5 Convergence Analysis

For the following proof we assume that f has L -Lipschitz continuous gradient on \mathcal{M} . This is to say that

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2} \|y - x\|_2^2$$

. This has been shown to be a reasonable assumption for DNN. Lipschitz continuous gradient gives us bound on curvature constant C_f Where C_f is the smallest constant such that We additionally require that our constraint space \mathcal{M} is convex and compact with diameter D . Satisfying $\|x - y\|_2 \leq D$ for $x, y \in \mathcal{M}$. This is trivially satisfied in the application of adversarial attacks, considering the definition of \mathcal{M} . Where Chen et al. 2020 used this strategy to prove the convergence of FW with momentum. We the result holds for normal FW. At each iteration since $s_t \in \mathcal{M}$, we get that $\|\nabla f(x_t) - s_t\|_2$

5.1 Frank-Wolfe

5.2 Pairwise Frank-Wolfe

5.3 Away-Step Frank-Wolfe