Comparison of Frank-Wolfe Varients 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 haverecieved much attention, as it has proved remarkably easy to create adversarial examples- genuine data that undergoes a minimal and unobtrusive corruption process inorder to maximally harm the performance of a model. Access to a models architecture can enable white-box attacks, where gradients of loss with respect to input examples are exploided to create adversarial examples. The requirement that examples be minimally preturbed is a constraint on the optimization. The ability of Frank-wolf and vairents have gathered much attention for their ability to be efficiently create adversarial examples while staying in a constraint set. In the paper we introduce discuss the application of Frank-Wolfe and two varients to this non-convex constrained optimization problem. Furthermore we discuss popular optimzations and their effect convergence and attack efficacy, comparing performance on attacks on models trained on MNSIT, FashionMNIST, and CFAIR-10 datasets. Finally, there is a discussion of the theoretical underpinnings of each algorithim.

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 liklihood 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 algorithioms are seen to require more iterations to converge. The L_p constraint $||x||_p \leq \epsilon$ directly restricts the size of perturbations made the to the example. An inherient problem of DNNs is often attacks can be consistently sucessful even with very small, even imperceptable ϵ. Different choices of p may be made giving $||x||_p = (\sum_i x_i^p)^{1/p}$. We have implemented and provide results for the L_2 and the $L_\infty(x) = \max_i |x_i|$ norms, focusing mostly on the ladder. We define the constraint set $\mathcal{M} = \{x : L_p(x) \leq \epsilon\}$. Of pirticular 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 distubance along each element. In these cases, the constraintset can be expressed as the convex combination of a finite set of verticies $\mathcal{M} = \text{Conv}\{\mathcal{A}\}$. This will be of pirticular relevance in the discussion of Away-step and pairwise varients. Otherwise, $\mathcal{A}$ is taken to be the boundry of $\mathcal{M}$. The constrained nature of this problem limits the applicabiliy 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 unsifisticated adversarial examples or require wasteful prejection 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 and 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 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 intrepreted 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 boundry of the constraint set $\mathcal{M}$ which was optimally chosen to be as close to the true gradient as possible while permitting subsequent iterations staty 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. In the convex case, 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 Algorithims

3.1 Frank-Wolfe

Algorithm 1 FW for adversarial attacks

```
Require: maximum iterations $T$, stepsizes $\{\gamma_t\}$, convergence tolerence $\delta$
Ensure: $y = x^n$
$x_0 = x_{\text{ori}}$
for $t = 1, ..., T$ do
    $s_t = \arg\min_{x\in\mathcal{M}} \langle x, \nabla f(x_t)\rangle$                ▷ LMO step
    $d_t = s_t - x_t$
    $x_{t+1} = x_t + \gamma_t d_t$
    if $\langle d_t, -\nabla f(x)\rangle < \delta$ then return        ▷ FW gap convergence criterion
    end if
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 osicilation 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 standerd frank wolf which replaces the gradient in the LMO calulation 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 varients are emperically observed to have nicer convergence.

The FW algorthim has a useful intrepretation that serves as basis for the following varients: 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 $S_{t+1} = S_t \bigcup \{s_t\}$ and observe that initially x_0 is a convex combination of 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 algorithim is seen to increase or introduce the contribution of s_t in the convex combination while shrink the α coefficients of all other verticies, or atoms uniformly. The innovation of the Away-Step and Pairwise varient is to recognize contribution of "bad atoms" can prevent convergence to an optimum on the boundry. These varients more directly diminish such atoms contributions by either taking steps away from selected atoms, or transfering mass between two selected atoms at each iteration as the case with the pairwise varient.

3.2 Away-Step Frank-Wolfe

The away-step FW algorithim (AFW) defines a gap in two possible directions: the typical FW step d_t^{FW} and an away direction $d_t^{\mathrm{A}} = x_t - v_t$ 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 algorithim 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 twords the frank wolfe direction, the influence of the selected away atom $v_t \in S_t$ will be deminished, 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 infulene of v_t is distributed uniformly to other active atoms. New coefficients for the comvex 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 algorithim enables a "drop step" where the contribution of an atom is compleatly removed from the convex combination, aleviating the problem of osicllation when converging to an optimum on the boundry. This algorithim 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 tolerence $\delta$, $x_0 \in \mathcal{M}$
Define $S_0 := \{x_0\}$ with $\alpha_{x_0} = 1$
for $t = 1, ..., T$ do
    $s_t := \arg\min_{x \in \mathcal{M}} \langle x, \nabla f(x_t) \rangle$    ▷ LMO step
    $d_t^{\mathrm{FW}} := s_t - x_t$
    $v_t := \arg\max_{v \in S_t} \langle v, \nabla f(x_t) \rangle$
    $d_t^{\mathrm{A}} := x_t - v_t$
    if $\langle d_t^{\mathrm{FW}}, -\nabla f(x) \rangle < \delta$ then return $x_t$    ▷ FW gap convergence criterion
    end if
    if then$\langle d_t^{\mathrm{FW}}, -\nabla f(x) \rangle < \langle d_t^{\mathrm{A}}, -\nabla f(x) \rangle$
        $d_t = d_t^{\mathrm{FW}}$, $\gamma_{\max} := 1$
    else
        $d_t := d_t^{\mathrm{A}}$, $\gamma_{\max} := \frac{\alpha_{v_t}}{1-\alpha_{v_t}}$
    end if
    $x_{t+1} = x_t + \gamma_t d_t$
    Update $\alpha$, $S_{t+1}$ s.t. $\langle \alpha, S_{t+1} \rangle = x_{t+1}$ (See below)
end for
```

3.3 Pairwise Frank-Wolfe

Pairwise Frank-Wolfe (PFW) is similar to AFW in that it computes the same away step. However, step taken by the algorithim is not a choice between the two, but rather their sum $d_t^{\mathrm{PFW}} = d_t^{\mathrm{FW}} + d_t^{\mathrm{A}} = s_t - v_t$.

Algorithm 3 Pairwise FW for Adversarial Attacks

```
Require: maximum iterations $T$, stepsizes $\{\gamma_t\}$, convergence tolerence $\delta$, $x_0 \in \mathcal{M}$
Define $S_0 := \{x_0\}$ with $\alpha_{x_0} = 1$
for $t = 1, ..., T$ do
    $s_t := \arg\min_{x \in \mathcal{M}} \langle x, \nabla f(x_t) \rangle$    ▷ LMO step
    $d_t^{\mathrm{FW}} := s_t - x_t$
    $v_t := \arg\max_{v \in S_t} \langle v, \nabla f(x_t) \rangle$
    if $\langle d_t^{\mathrm{FW}}, -\nabla f(x) \rangle < \delta$ then return $x_t$    ▷ FW gap convergence criterion
    end if
    Require: $\gamma_t \leq \alpha_{v_t}$
    $d_t := s_t - v_t$
    $x_{t+1} = x_t + \gamma_t d_t$
    Update $\alpha$, $S_{t+1}$ s.t. $\langle \alpha, S_{t+1} \rangle = x_{t+1}$ (See below)
end for
```

The behaivor of the of PFW can be understood to be transfering mass in the convex combination from the worst atom to the active set to the atom corresponding the the FW direction. At each iteration the α coefficients are updated as follows: $\alpha_{v_{t+1}} = \alpha_{v_t} - \gamma_t$, $\alpha_{s_{t+1}} = \alpha_{s_t} + \gamma_t$ and all other α values are unchanged. To maintian x_t *convex* combination of atoms in active set, and consequentily not affine or infeasible, its required $\gamma_t \leq \gamma_{\max} := \alpha_{v_t}$. By directly minimizing, and potentially dropping bad atoms from the active set, the osicilation observed with FW is mitigated. In the convex case this has been shown to give the PFW varient a linear convergence rate.

4 Results

Algorithims were tested on three pretrained models, each performing a classification task on a different dataset. An implementation of LeNet5 described by LeCun et. al 1998) is a CNN with 62,706 parameters trained on the MNIST dataset to classify handwritten didgets. Secondly we found a medium sized CNN pretrined for the Fashion MNIST dataset. Designed to be a more difficult problem than MNIST, the model classifies images as one of T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. Finally, ResNet 20 is a comparitively large CNN with 20 layers. We use a pretrained implementation adapted for the CIFAR-10 dataset which classifies 32x32 examples into one of 10 categories: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. All three proved suseptible to Advarsarial attacks created by FW methods to varying degrees. Representitve sucessful attacks are shown in figure 1.

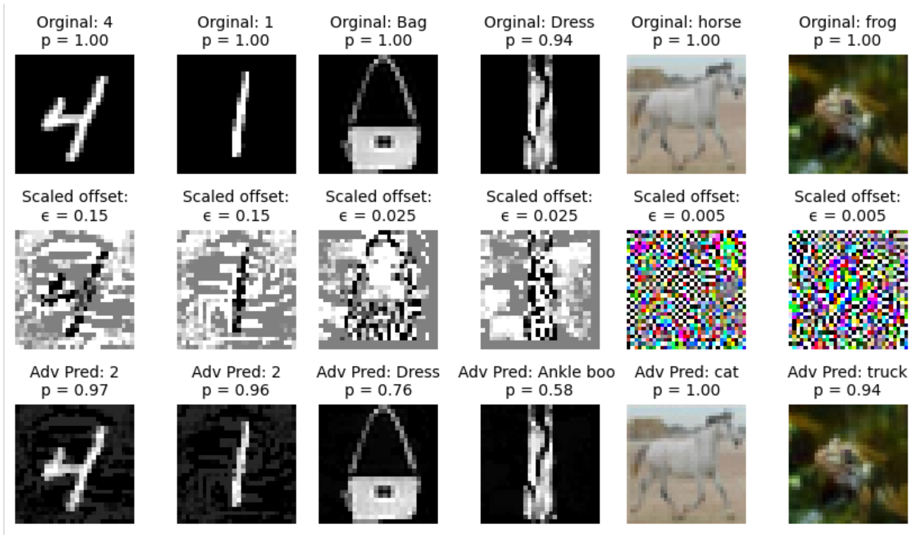

Figure 1: Representative attacks on LeNet, FMNIST, and ResNet-20.

We observed that the larger and more complicated a model, the more suseptible it is to adversarial attack. LeNet5 being a notably small and simple model by contempory standards, showed remarkable robustness and required attacks well within perceptible levels ($\epsilon \approx 0.3$) before the tested FW methods could reliably produce adversarial examples. While ResNet and FMNIST model can be rilably attacked with ϵ above 0.07 and 0.05 respectively (see Figure 2).

4.1 ϵ Choice

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

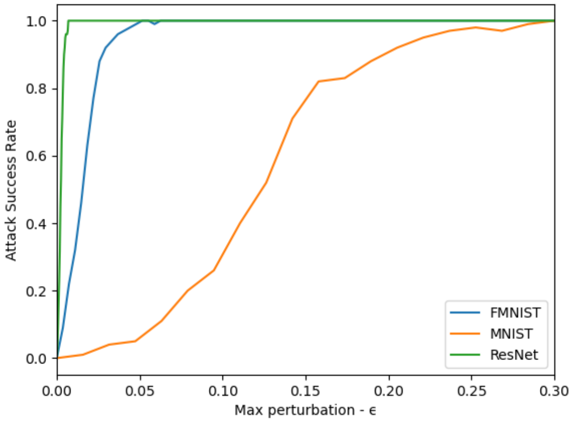

Figure 2: Effect of ϵ on attack success for untargeted FW for each dataset with L_∞ constraint.

As figure 5 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 attakcs. 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. Representitive attacks are shown in figure 2.

4.2 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, unsuprisingly, that targeted attacks are invariably a more challanging problem, resulting in lower success rates and higher average iterations for all varients and datasets.

For example, attacks on a set of 100 examples with ResNet18 and $\epsilon = 0.005$ with normal FW algorithim using decaying rule, acheived 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.

4.3 L_∞ vs L_2 Constraints

We implement attacks Linear Minimization orcales to enable both L_∞ and L_2 constrained attacks. Choices of ϵ for the two norms are clearly not comparable as L_∞ is conserned by the maximum pertrubation a pixel is perturbed by, doing nothing to disencentivise most if not all pixels from being pertrubed, where L_2 constrains the sum of squares of pertrubations.

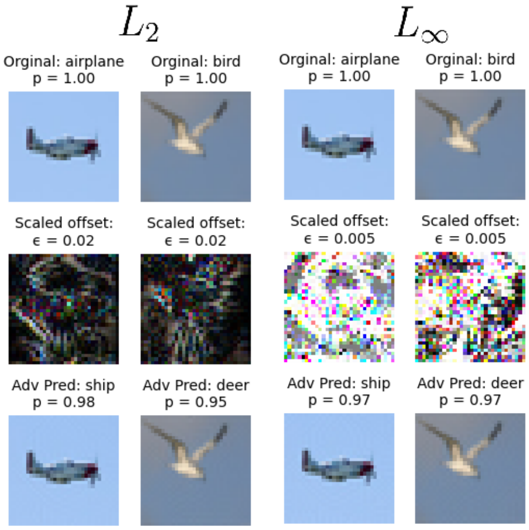

Figure 3: Effect of constraint norm on attacks on CIFAR-10.

To compare the resulting adversarial examples generated by attacks I chose ϵ values that gave approximately the same attack success rate of aboout 62% on CIFAR-10 ($\epsilon_\infty = 0.005, \epsilon_2 = 0.02$) and compare the $L1$ norm of the generated perturbations. For the to get the same success rate we expectedly see that $L2$ constrained attacks are more efficient at producting attacks, with an average $L1$ of 6.84 ($\sigma = 0.69$) where L_∞ constrained attacks created attacks with average perturbation L_1 of 10.5. However, we note that both cases produced attacks below perceptible levels with L_2 attacks having an average L_∞ of 0.013 and it being difficult to think of applicaitons were this efficiency is worth the trade off. We visualize the magnitude of attacks made by the attacks using the different constraints in figure 3.

4.4 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.

We found the application of linesearching techniques to be marginally benificial when compared to decaying rule.

4.5 Momentum

Momentum proved highly effective at reducing the number of iterations required for convergence with minimal effect attack sucess rate. On a test of 1000 examples from CIFAR-10. Normal FW provided an ASR of 88.7% with average of 11.39 iterations. FW with momentum ($\beta = 0.8$) acheived 89.5% ASR and 8.47 mean iterations, indicating a significant improvement in convergence (see Figure 4). This is seen to outperform all other varients with the same parameters, achiving appreciably better ASR and mean convergence time.

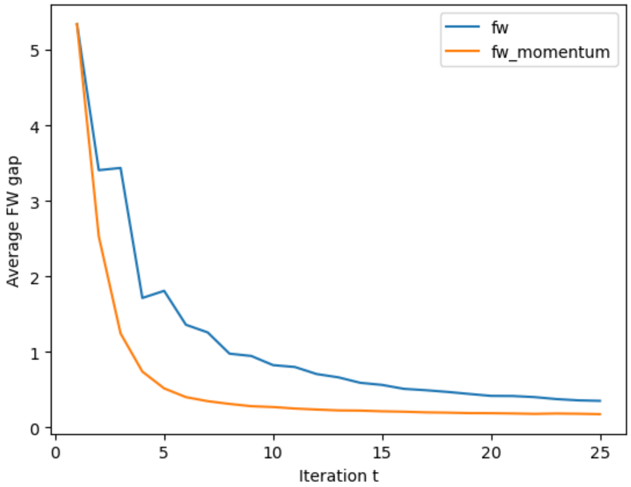

Figure 4: Comparison of g_t^{FW} of FW and momentum varient targeted attacks on ResNet-20 with $\epsilon = 0.007$ and $\beta = 0.8, N = 1000$.

4.6 Varients

Algorithm	**MNIST** $\epsilon = 0.15$		**F-MNIST** $\epsilon = 0.025$		**CIFAR-10** $\epsilon = 0.005$	
	ASR (%)	**Avg Iter**	**ASR (%)**	**Avg Iter**	**ASR (%)**	**Avg Iter**
FW	85.4	50	86.1	48	87.4	10.2
AFW	87.3	45	88.0	43	86.2	10.2
PFW	86.5	47	87.2	44	82.2	11.2

Table 1: Attack Success Rate (ASR) and Average Iterations of Frank-Wolfe Algorithm and Variants for Targeted Adversarial Attacks on 1000 Examples from Three Datasets with 20 maximum iterations and decaying stepsize rule.

We note that ϵ values were chosen to create a difficult optimization problem that highlighted diffrences in convergence between algorithims, and ϵ could have been easily chosen to ensure 100% ASR on all datasets.

We observe that the difficulity of the optimization problem, as influecned by the choice of dataset to attack, ϵ choice, and perfroming either targeted or untargeted attacks will effect the number of Away Steps taken by AFW. Targeted attacks on CIFAR-10 as described in table 1. For Both attacks on LeNet5 and FMNIST model, AFW was never seen to take an away step. However, when attacking ResNet-20, it is seen to have taken Away Steps 17.1% of the time. As it takes majority FW steps, the convergence of AFW is seen to closely follow that of normal FW (see Figure 5).

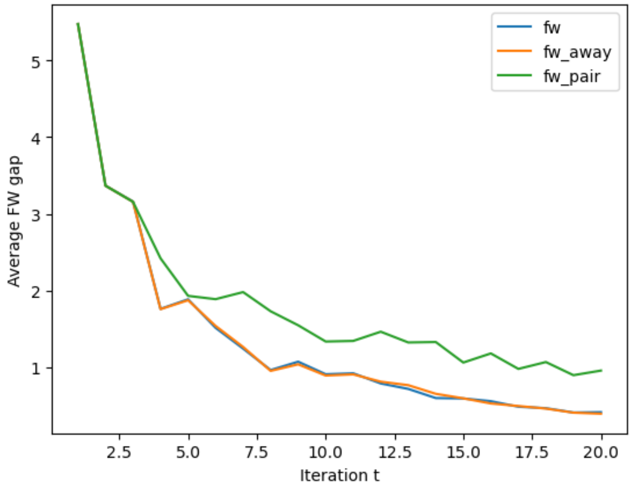

Figure 5: Average g_t^{FW} for each Algorithim by Iteration for Targeted Attacks on ResNet-20 (CIFAR-10) with decaying rule and $\epsilon = 0.005$.

We observe that this application does not favor the use of PFW. Which we attribute to the lack of linesearching, which we found to be inappropiate for the non-convex application. Analysis of the away costs at each iteration showed that g_t^{A} is often negative, indicating that it is still a good atom in the active set S_t. In the absense of linesearching, PFW with decaying rule frequently will over minimize, or even drop the contribution v_t to x_t harming convergence and attack sucess rate. We observe that the use of linesearching harms the of PFW less so than the other two algorithims, making their performance more comparable. But still suffers form the afformentioned problems associated with non-convexity and linesearching.

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. Lipshitz continuious 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 dinameter D. Satisifying $||x-y||_2 \leq D$ for $x, y \in \mathcal{M}$. This is trivially satisified in the application of adversarial attacks, considering the defintion 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