Universal Adversarial Perturbation starring Frank-Wolfe

Chiara Bigarella
Student nr. 2004248

Silvia Poletti
Student nr. 1239133

Gurjeet Singh
Student nr. 2004251

Francesca Zen
Student nr. 2010640

1. Introduction

1.1. Adversarial Attacks

TODO: What are adversarial attacks?

1.2. Universal Adversarial Perturbations

Universal perturbations are small perturbations that, when applied to the input images, are able to fool a state-of-the-art deep neural network classifier. These perturbations are quasi-imperceptible for humans, due to their small norm, and therefore they are difficult to detect. However, what makes universal perturbations special, is their capability to generalize well both on a large dataset and across different deep neural network architectures. In fact, an important property of universal perturbations is that they are image-agnostic. This means that they don't depend on a single image, but rather they are able to cause label estimation change for most of the images contained in a dataset.

It is important to notice that there's no unique universal perturbation: different random shuffling of the data used to compute the perturbation can lead to a diverse set of universal perturbations.

Furthermore, universal perturbations mostly make natural images classified with specific labels, called *dominant labels*. These dominant labels are not determined a priori, but rather they are automatically found by the algorithm that computes the universal perturbation. In paper XXX, the authors hypothesize that ”these dominant labels occupy large regions in the image space, and therefore represent good candidate labels for fooling most natural images”.

Finally, although some fine-tuning strategies can be adopted to make deep neural networks more robust to adversarial attacks, they are not sufficient to make the classifiers immune to the attacks. In fact, the fine-tuning of a classifier with universal perturbations leads to an improvement in the classification of perturbed images, however it is always possible to find a small universal perturbation that can fool the network.

2. Gradient Free Frank Wolfe

Gradient free optimization algorithms find application in settings where the explicit closed form of the loss function is not available or the gradient evaluation is computationally prohibitive. A prime example is black-box adversarial attacks on neural networks, where only the model output is known, while the architecture and weights remain unknown. In fact, gradient free methods exploit just zeroth-order oracle calls (i.e. loss function evaluations) to solve an optimization problem.
In particular, this report focuses on zeroth-order Frank Wolfe algorithms for constrained optimization problems. Unlike the Projected Gradient method that requires expensive projection operations, the Frank Wolfe framework provides computational simplicity by making use of instances of linear minimization.
Considering the application of interest, that is black-box adversarial attacks, a ”good enough” feasible solution is often adequate. Therefore, the use of biased gradient estimates obtained from zeroth-order information is suitable for performing this task.

1) SPIEGARE ZEROTH-ORDER OPTIMIZATION + STOCHASTIC AND CONSTRAINED OPTIMIZATION
2) SPIEGARE FW FROM 1ST ORDER TO 0TH ORDER
3) INTRODURRE IL CONCETTO DI DECENTRALIZED E DISTRIBUTED SETTINGS

3. Materials and Methods

4) VISTO CHE GLI ALGORITMI CHE ABBIAMO IMPLEMENTATO SONO SPECIFICI PER GLI ADVERSARIAL ATTACKS SU MNIST, INTRODURRE/DEFINIRE IL DATASET MNIST E IL PROBLEMA DI OTTIMIZZAZIONE (39) PRIMA DI INTRODURRE GLI ALGORITMI

3.1. Decentralized Stochastic Gradient Free Frank Wolfe

In this section we discuss the Stochastic Gradient Free FW in a decentralized setup. In this setting we have some devices, that we call workers, connected to a master node to read, write and exchange information.

In our case we have M workers to which is spread the data, they compute their local gradient, using the loss function and the parameters given from I-RDSA, and send them to the master node who uses the gradients collected to return the new perturbated data.

Given an input image δ, the algorithm initializes both the starting point and the gradient to a zero d-vector and $M \times d$ matrix, respectively. Then each worker computes its own gradient estimation based on the parameters given from I-RDSA in the following way:

$$\mathbf{g}_i = (1 - \rho_t)\mathbf{g}_{i,t-1} + \rho_t \mathbf{g}_i(\delta_t, \mathbf{y}).$$

When all of the workers finish their tasks, they send the results to the master node, which computes the average of the ones collected and get back the obtained value as new initial point δ_{t+1}:

$$\delta_{t+1} = (1 - \gamma_t)\delta_t + \gamma_t \mathbf{v}_t$$

When all of the iterations are done, the algorithm returns the average of the gradient computation done by the master node.

Algorithm 1 Decentralized Stochastic Gradient Free FW

```
Require: Input image $\delta$, labels $y$, Loss Function $F(\delta; y)$, number of queries $T$, number of workers $M$, image dimension $d$, tolerance $\varepsilon$, number of directions $m$.
Ensure: $\delta^T$
Initialize $\delta_0 = 0$.
for $t = 0, \dots, T-1$ do
    Master node computes parameters required for the computation of the I-RDSA scheme:
        $(\rho_t, c_t)_{I-RDSA} = \left( \frac{4}{(1 + \frac{d}{m})^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}} \right)$
    For each worker $i$ compute I-RDSA:
    Sample $\{\mathbf{z}_{n,t}\}_{n=1}^{m} \sim N(0, \mathbf{1}_d)$
    $\mathbf{g}_i(\delta_t; \mathbf{y}) = \frac{1}{m} \sum_{n=1}^{m} \frac{F(\delta_t + c_t \mathbf{z}_{n,t}; \mathbf{y}) - F(\delta_t; \mathbf{y})}{c_t} \mathbf{z}_{n,t}$
    Workers compute
        $\mathbf{g}_i = (1 - \rho_t)\mathbf{g}_{i,t-1} + \rho_t \mathbf{g}_i(\delta_t, \mathbf{y}).$
    Push $\mathbf{g}_{i,t}$ to the master node.
    Master node computes
        $\mathbf{g}_t = \frac{1}{M} \sum_{i=1}^{M} \mathbf{g}_{i,t}.$
    Master node computes $\mathbf{v}_t = -\varepsilon sign(\mathbf{g}_t)$.
    Master node computes $\delta_{t+1} = (1 - \gamma_t)\delta_t + \gamma_t \mathbf{v}_t$ and sends it to all nodes.
end for
```

3.2. Decentralized Variance-Reduced Stochastic Gradient Free Frank Wolfe

In this section we employ the SPIDER variance reduction technique, which is a built for dynamic tracking, while avoiding excessive querying to oracles and ultimately reducing query complexity.

3.3. Distributed Stochastic Gradient Free Frank Wolfe

In this section we discuss the Distributed Stochastic Grandient Free FW in a distributed setup. In this setting we have that the M workers do not have a central coordinator, instead they exchange information in a peer-to-peer manner. The internode comunication network used by the workers is modeled as an undirected simple connective graph $G = (V, E)$, with $V = \{1, \dots, M\}$ the set of nodes and E the set of comunication links. Each node comunicates and exchange information with its own neighbors, and given a node n we indicate with $\Omega_n = \{l \in V | (n, l) \in E\}$ its neighborhood. Node n has degree $d_n = |\Omega_n|$. Also, we use the $M \times M$ adjacency matrix $A = [A_{ij}]$ to describe the edges of the graph G: we have $A_{ij} = 1$ if $(i, j) \in E$, otherwise we get $A_{ij} = 0$. We define the diagonal matrix $D = (d_1 \dots d_M)$ in order to compute the graph Laplacian $L = D - A$. The Laplacian of L is define to be the matrix

$$\mathcal{L}(u, v) = \begin{cases} 1 & \text{if } u = v \text{ and } d_v \neq 0 \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

We can write $\mathcal{L} = D^{-1/2} L D^{-1/2}$, with the convention that $D^{-1}(v, v) = 0$ if $d_v = 0$. The Laplacian $\mathcal{L}$ is used to compute the weghted matrix $W = 1 - \mathcal{L}$.

At every time instat t in Algorithm 3, an agent i exchange its current iterate with its neighbors and average the iterates as follows:

$$\bar{\delta}_t^i \leftarrow \sum_{j=1}^{M} W_{ij} \delta_t^i.$$

After this, each worker i use I-RDSA to compute its local gradient estimation g_t^i, for $i = 1, \dots, M$, using $c_t = \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}}$. When all of them are calculated, they are

Algorithm 2 Decentralized Variance-Reduced Stochastic Gradient Free FW

```
Require: Input image $\delta$, labels $y$, Loss Function $F(\delta; y)$, number of queries $T$, number of workers $M$, image dimension $d$, tolerance $\varepsilon$, number of images $S_1$, number of function components $S_2$, total number of function component $n$, period $q$, duality gap tolerance $tol$.
Ensure: Universal prturbation's history.
Initialize $\delta_0 = 0$.
for $t = 0, \ldots, T-1$ do
    Each worker $i$ computes:
    if $mod(t, q) = 0$ then
        Draw $S_1' = \frac{S_1 d}{M}$ samples for each dimension at each worker $i$ and compute its local gradient $\mathbf{e}_k^T \mathbf{g}_i(\delta_t) = \frac{1}{n}\sum_{j=1}^{n} \frac{F_{i,j}(\mathbf{x}_t + \eta \mathbf{e}_k) - F_{i,j}(\mathbf{x}_t)}{\eta}$ along each canonical basis vector $\mathbf{e}_k$.
        Each worker updates $\mathbf{g}_{i,t} = \mathbf{g}_i(x)_t$.
    else
        Draw $S_2$ pairs of component functions and Gaussian random vectors $\{\mathbf{z}\}$ at each worker $i$ and update
        $\mathbf{g}_i(x)_t = \frac{1}{|S_2|}\sum_{j \in S_2} \frac{F_{i,j}(\mathbf{x}_t + \eta \mathbf{e}_k) - F_{i,j}(\mathbf{x}_t)}{\eta}\mathbf{z}$ - $\frac{F_{i,j}(\mathbf{x}_{t-1} + \eta \mathbf{e}_k) - F_{i,j}(\mathbf{x}_{t-1})}{\eta}\mathbf{z}$
        Each worker updates
            $$\mathbf{g}_{i,t} = \mathbf{g}_i(\mathbf{x}_t)\mathbf{g}_{i,t-1}.$$
    end if
    Each worker pushes $\mathbf{g}_{i,t}$ to the master node.
    Master node computes
        $$\mathbf{g}_t = \frac{1}{M}\sum_{i=1}^{M} \mathbf{g}_{i,t}.$$
    Master node computes $\mathbf{v}_t = -\varepsilon sign(\mathbf{g}_t)$.
    Master node computes $\delta_{t+1} = (1 - \gamma_t)\delta_t + \gamma_t \mathbf{v}_t$ and sends it to all nodes.
end for
```

collected as follow:

$$\bar{\mathbf{g}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij} \mathbf{G}_t^j.$$

After the update, the algorithm computes the Frank-Wolfe step, calculating the new iterate δ_{t+1}^i using $\gamma_t = t^{-1/2}$. A distributed FW algorithm requires two round of communication: one for the iterate and one for the gradient estimate.

The performance of Algorithm 3 depends on how well the averaged iterates and the averaged gradient estimates are tracked across the network.

Algorithm 3 Distributed Stochastic Gradient Free FW

```
Require: Input image $\delta_0$, labels $y$, Loss Function $F(x; y)$, number of queries $T$, number of workers $M$, image dimension $d$, tolerance $\varepsilon$, number of directions $m$, adjacency matrix $A$.
Ensure: Average perturbated images.
Initialize $\delta_0^i = 0$ for $i = 1, \ldots, M$.
for $t = 1, \ldots, T$ do
    Approximate the average iterate:
        $$\bar{\delta}_t^i \leftarrow \sum_{j=1}^{M} W_{ij} \delta_t^i$$
    At each node $i$, employ I-RDSA to estimate the gradient $\mathbf{g}_t^i = \sum_{j=1}^{d} \frac{F(\bar{\delta}_t^i + c_t \mathbf{e}_j) - F(\bar{\delta}_t^i)}{c_t}\mathbf{e}_j$.
    Approximate the average gradient:
        $$\mathbf{G}_t^i = \bar{\mathbf{g}}_{t-1}^i + \mathbf{g}_t^i - \mathbf{g}_{t-1}^i$$
        $$\bar{\mathbf{g}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij} \mathbf{G}_t^j$$
    Each worker computes $\mathbf{v}_t = -\varepsilon sign(\mathbf{g}_t)$.
    *Frank-Wolfe Step*: update
        $$\delta_{t+1}^i \leftarrow (1 - \gamma_t)\bar{\mathbf{x}}_t^i + \gamma_t \mathbf{v}_t^i$$
    for all worker $i = 1, \ldots, M$.
end for
```

4. Experiments

5. Conclusions