Adversarial Attacks nei sistemi di Deep Learning

FGSM, PGD e approcci difensivi

Gianmarco Russo mat.887277
MSc Data Science

Report esame **Cybersecurity for Data Science**

February 15, 2023

Contents

1 Introduzione **1**

2 Adversarial Attacks **1**
2.1 Fast Gradient Sign Method (FGSM) 2
2.2 Projected Gradient Descent (PGD) 2

3 Contromisure **2**
3.1 Reactive - Denoising 3
3.2 Proactive - SVD 3

4 Demo **4**
4.1 CleverHans 4
4.2 Dimostrazione metodo SVD 4

5 Conclusioni **5**

6 Fonti **5**

1 Introduzione

Gli algoritmi di Machine Learning vengono utlilizzati per creare modelli matematici per sistemi basati su dati di addestramento. Un modello di questo tipo è in grado di effettuare predizioni accurate senza essere stato esplicitamente programmato per far ciò. Queste tecniche hanno un vasto ambito di applicazioni: dalla digital economy all'intelligenza artificiale, includendo anche ambiti di importanza critica come la guida autonoma o ISDS. Gli ISDS (intelligent security detection systems) sono sistemi progettati per identificare e mitigare l'attività malevola. Il rilevamento delle anomalie è un esempio comune in cui modelli di machine learning vengono impiegati per ottenere un modello dal comportamento affidabile. Il modello analizza il comportamento *normale* per essere in grado di classificare in seguito le attività *anomale*, e segnalare quindi l'attività malevola. Questi sistemi di sicurezza sono vulnerabili ad attacchi avversari.

2 Adversarial Attacks

Gli adversarial attacks rappresentano un serio rischio per gli ISDS. Questi ultimi possono essere classificati in 3 categorie:

a) **Evasive**: manipolazione dati per far si che il classificatore performi male;

b) **Poisoning**: iniezione di dati malevoli per sabotare il training del modello (richiedono l'accesso ai suddetti dati);

c) **Confedentiality violations**: tramite le analisi di modelli di ML addestrati.

Di seguito vengono discussi due metodi di poisoning per sistemi di deep learning basati su immagini, con conseguente dimostrazione in python (vedi sezione Demo).

2.1 Fast Gradient Sign Method (FGSM)

Come inizialmente introdotto da I. J. Goodfellow, Shlens e Szegedy[1], le reti neurali, a causa delle non linearità presenti nelle loro architetture, sono molto fragili a perturbazioni lineari avversarie. FGSM è un tipico attaco one-step, che performa aggiornamenti one-step lungo la direzione (il segno) del gradeinte della adversarial loss, per incrementare la perdita nella direzione più ripida. Formalmente un campione avversario FGSM è così formulato

$$x' = x + \epsilon sgn(\nabla_x J(\theta, x, y)) \tag{1}$$

Dove ϵ è la magnitudine della perturbazione. FGSM può essere facilmente esteso ad un attacco mirato (targeted FGSM) lavorando col gradiente di $J(\theta, x, y')$ dove y' è l'etichetta target. Questa procedura di aggiornamento decresce la cross-enthropy tra il vettore di probabilità predette e il vettore di probabilità target se la cross-enthropy è applicata come la adversarial loss. La regola di update può essere formulata come segue:

$$x' = x - \epsilon sgn(\nabla_x J(\theta, x, y')) \tag{2}$$

In un adeguato spazio ad alta dimensionalità, tante piccole modifiche infinitesime, ognuna di queste non più grandi di ϵ, possono sommarsi fino a produrre una perturbazione abbastanza grande in grado di alterare le previsioni del modello[1].

2.2 Projected Gradient Descent (PGD)

È considerato uno dei migliori algoritmi robusti di generazione avversaria[2][3]. Il funzionamento alla base è molto simile a quello di FGSM, ma viene reso iterativo. Inoltre, l'inizializzazione avviene in un punto casuale nella sfera di interesse ed effettua riavvii automatici. Quindi ottiene esempi avversari utilizzando una variante multi-step di FGSM, inizializzando $x^0 = x$. I dati perturabati al passo t-esimo possono essere espressi come:

$$x^t = \prod_{x+s}(x^{t-1} + \alpha sgn(\nabla_x L(f_\theta(x^{t-1}, y))) \tag{3}$$

dove $\prod_{x+s}$ rappresenta le perturbazioni proiettate nel set s ed α denota lo step-size. Si definisce PGD vincolato con l_∞ come *PGD-inf attack* ed il PGD vincolato con l_2 come *PGD-l2 attack*.

3 Contromisure

In questo elaborato ci concentriamo su una difesa definita **proactive** (SVD), ovvero approcci che puntano a rinforzare il modello e a renderlo più robusto agli attacchi e men-

zioniamo alcune difese **reactive**, ovvero tecniche che mirano a identificare un attacco e a reagire di conseguenza.

3.1 Reactive - Denoising

È una classe di difese reattive che mira a mitigare l'efficacia di attacchi, rimuovendo o riducendo le perturbazioni introdotte dalla generazione avversaria. Esistono due approcci: uno è quello di ripulire l'input originale prima di passarlo al modello, un altro è quello di lavorare sulle feature estratte all'interno della rete neurale. Alcune tecniche appartenenti a questa categoria sono[2]:

- *Conventional Input retification*,
- *GAN-based Input cleansing*,
- *Autoencoder-based Input denoising*.

3.2 Proactive - SVD

In uno spazio ad alta dimensionalità, anche la piccola modifica di un pixel può drammaticamente modificare lo spazio delle features, portando la rete a classificare in maniera errata. Singular Value Decomposition è un metodo di riduzione della dimensionsionalità che può essere utilizzato come difesa avversaria per capire se l'input sia stato modificato. Computa una approssimazione ottimale della matrice dell'immagine in termini di square loss. Senza entrare nel merito del funzionamento della SVD, ci limiatiamo a dire che una matrice di valori A (un'immagine), a cui viene applicata questa trasformazione, può essere espressa come:

$$A = UsV^T \tag{4}$$

La distribuzione caratteristica di una matrice è descritta dai suoi valori singolari (contenuti in s). Nel caso di un segnale 2D (un'immagine) può essere anche espressa come l'energia del segnale. I valori singolari hanno diverse proprietà che li rendono invarianti a trasformazioni geometriche. Questo significa che applicando una SVD alle immagini di training, dovremmo eliminare le eventuali perturbazioni che un attacco avversario può aver causato. [4]

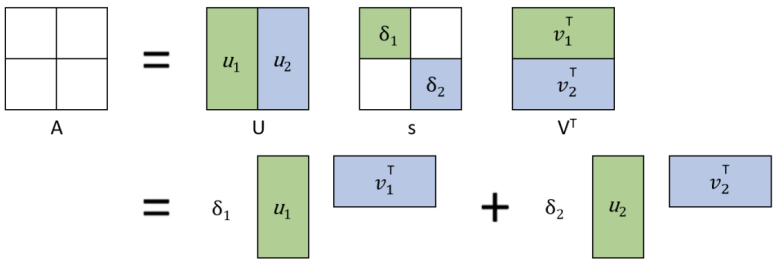

Figure 1: SVD graphic explanation

4 Demo

4.1 CleverHans

La dimsotrazione pratica è avvenuta per lo più utilizzando le funzioni presenti nella libreria **CleverHans**[5]. La libreria fornisce dei tutorial di attacchi avversari su **MNIST** e **CIFAR-10**. In questo elaborato vengono utilzzate le funzioni scritte per *tensorflow* sul dataset MNIST, ma sono anche presenti esempi in *pytorch* e *jax*. In particolare, le immagini che seguono sono state realizzate con attacchi FGSM (*fast_gradient_method.py*) e PGD (*projected_gradient_descent.py*) con valori di ϵ diversi, per valutarne l'effetto visivo.

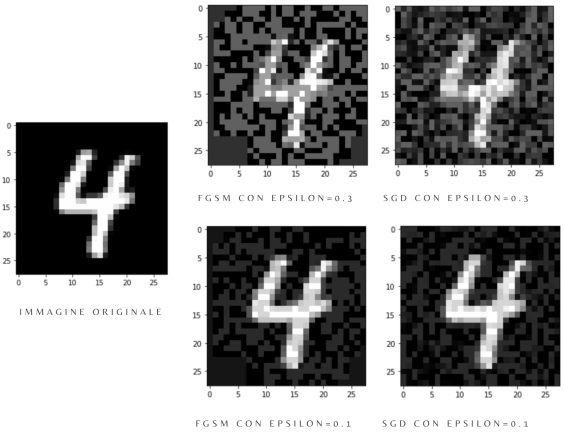

Figure 2: Immagine Campione MNIST dopo FGSM e PGD

Ovviamente, trattandosi di immagini a bassa risoluzione (28x28px) si nota anche ad occhio nudo la modifica dell'immagine. Possiamo facilmente immaginare che in immagini più risolute si noti molto meno. La rete neurale è stata addestrata su un train set modificato con FGSM e testato sia su test set originale che anch'esso modificato con FGSM. Di seguito i risultati.

Accuracy Clean	Accuracy FGSM	ϵ
8.070	9.880	0.1
8.960	9.760	0.3

Come possiamo vedere l'attacco ha avuto successo: la rete ha ottenuto una accuratezza inferiore al 10% (che in questo caso è il minimo ottenibile se la rete classificasse in maniera casuale, trattandosi di 10 classi). L'implementazione qui riportata utilizza lo strumento Google Colab, in modo da poter effettuare i training e le operazioni descritte utilizzando le risorse computazionali del suddetto strumento e non la propria macchina.

4.2 Dimostrazione metodo SVD

Per dimostrare quanto indicato nel paragrafo 3.2 è stata applicata la SVD su un campione originale e su uno soggetto ad attacco FGSM. Successivamente l'immagine è stata ricostruita

usando i primi 7 valori singolari (trattandosi di immagini 28x28, i valori singolari sono 28). I primi valori singolari sono i più significativi del contenuto informativo dell'immagine, via via perdendo di significatività. Sia s il vettore dei valori singolari dell'immagine originale e s_fgsm il corrispettivo dell'immagine attaccata, calcoliamo la loro differenza in questo modo: $|sum(s) - sum(s\_fgsm)|$. Utilizziamo questa operazione sia su tutto il contenuto del vettore s e s_fgsm e poi unicamente sui primi 7 valori singolari. Di seguito i risultati:

$$|sum(s) - sum(s\_fgsm)| = 6.467; \quad |sum(s\_7) - sum(s\_fgsm7)| = 0.694 \tag{5}$$

Con s_7 e s_fgm7 il vettore s ricostruito con i primi 7 valori singolari. Questo piccolo esperimento dimostra che considerando unicamente i valori singolari più significativi il contenuto informativo di questi ultimi non viene alterato dall'attacco avversario. Quindi se immaginiamo di applicare questo metodo a tutte le immagini a nostra disposizione, e quindi ricostruirle riducendo la loro dimensionalità, nell'operazione stiamo anche rimuovendo eventuali perturbazioni avversarie. Il codice contenente il test con CleverHans su Google Colab e la SVD è disponibile in questa repo.

5 Conclusioni

In questo elaborato sono stati descritti alcuni degli attacchi avversari ai sistemi di ML/DL, e delle relative tecniche difensive. Ci sono inoltre alcune precisazioni da fare: gli attacchi trattati richiedono l'accesso ai dati di training del modello, aspetto tutt'altro che scontato, e che sfocia in altre nozioni di sicurezza informatica. Un'altra puntualizzazione da fare è sull'utilizzo di metodi come la SVD, in quanto se immaginiamo l'utilizzo di questa tecnica sull'interezza dei dati a disposizione (nel caso di deep learning solitamente i dataset sono molto grandi), possiamo ben immaginare come il peso computazionale di questa operazione diventi piuttosto oneroso.

6 Fonti

Bibliografia

[1] C.Szegedy I.GoodFellow J.Shlens. "Explaining and harnessing adversarial examples". In: *ICLR 2015* (2015).

[2] K. Ren et al. "Adversarial attacks and defenses in Deep learning". In: (2020).

[3] T. Huang et al. "Bridging the Performance Gap between FGSM and PGD Adversarial Training". In: (2022).

[4] Cato Pauling et al. "A Tutorial on Adversarial Learning Attacks and Countermeasures". In: (2022).

[5] *CleverHans: a Python library to benchmark machine learning systems' vulnerability to adversarial examples*. Available on line. URL: https://github.com/cleverhans-lab/cleverhans.