Säkerhet i AI-system (DV2607)

Inlämningsuppgift

Martin Boldt
Sadi Alawadi
Anton Kanerva
Institutionen för Datavetenskap (DIDA)
Blekinge Tekniska Högskola

2024-10-01

Introduktion

I likhet med alla mjukvarubaserade system kan även maskininlärningsystem (ML-system) och dess tränade modeller attackeras av personer med illasinnat uppsåt, t.ex. genom att skapa bakdörrar som lurar klassificeringsmodeller eller som tar bort nyttan med ML-modellen. Därför krävs också säkerhetsåtgärder för att kunna säkerställa att ML-modellerna kan utföra sina uppgifter på de sätt de är ämnade att utföras.

Denna inlämningsuppgift är uppdelad i två delar där ni i respektive del får stifta bekantskap med två vanliga attackertyper på ML-system, nämligen centraliserade *Adversarial input attacker* (i del 1) respektive distribuerade *Adversarial input attacker* (i del 2). Notera att del 2 beskrivs på engelska, medan del 1 beskrivs på svenska.

Praktiskt upplägg och betygsättning

Respektive del av inlämningsuppgiften innehåller förklaringar, uppgifter ni ska utföra samt frågor ni ska besvara i en rapport. Rapporten betygssätts därefer med betygsskalan *A-F*. För att erhålla något av betygen *C, D eller E* behöver ni minst genomföra de obligatoriska uppgifterna och frågorna under Del 1 respektive Del 2 nedanför. För att ha en chans att få betyg *A eller B* krävs dessutom att ni genomför det som beskrivs i de frivilliga delarna för både Del 1 och Del 2 nedan.

Betygskriterierna är som följer. Om ni siktar på betyg i skalan C, D eller E måste ni slutföra uppgifterna 1.1 t.o.m. 1.3 och 2.1 t.o.m. 2.2. För betyg B måste ni dessutom slutföra uppgift 1.4 och 2.3. För betyg A måste ni dessutom slutföra uppgift 1.4 och 2.4.

Notera att det är obligatoriskt att jobba i grupper om **två (2)** studenter då ni genomför denna inlämningsuppgift respektive skriver er gemensamma rapport.

När ni är klara med er skriftliga rapport lämnar ni in en ZIP-fil som innehåller er: (1) Pythonkod samt (2) er rapport i PDF-format. ZIP-filen skickas in på kursens Canvas-sida innan deadline, se mer info på kurssidan. Innan ni skickar in er inlämningsuppgift så kontrollera att ni:

1. använder den obligatoriska rapportmallen som finns på kurssidan,

2. utförligt besvarat uppgifterna i både Del 1 och Del 2 nedan,

3. inkluderar all er Pythonkod som behövs för att köra era uppgifter (inkl. Jupyter Notebooks från Del 2),

4. skrivit båda personernas namn och e-postaddress i er rapport samt i headern i all källkod,

5. lämnar in er rapport i PDF-format, och

6. Packat ihop er kod och er PDF-rapport i en ZIP-fil som ni lämnar in.

Part 1: Adversarial input attacker och skydd

Evasion attack är en attack-typ där maskininlärningsmodellen attackeras genom att modellen luras till att ta felaktiga beslut. När man talar om *evasion attacks* syftar man oftast på en attack som heter *adversarial input attack*. *Adversarial inputs* är instanser vars värden har manipulerats för att klassificeras till en annan klass utan att värdena egentligen representerar den klassen. Attacken är enklast utförd och demonstrerad på bilddata, då man enkelt kan lägga ett så pass lite brus på en bild att det är osynligt för människoögat, men får modellen att klassificera bilden som något helt annat än vad vi ser på bilden. Den går dock även att utföra inom andra datatyper såsom text- eller tabelldata, även om den senare medför en del utmaningar för att obfuskera "bruset".

Denna del av inlämningsuppgiften går ut på att få en grundläggande förståelse kring hur maskininlärningsmodeller kan attackeras via input i klassifikationssteget genom *adversarial input attacks*, samt vilka säkerhetsåtgärder man kan vidta för att försvåra eller kanske t.o.m. förhindra sådana attacker. Ni kommer att träna en modell på ett dataset för att sedan attackera den resulterande modellen med adversarial inputs som ni genererar för att lura modellen. Ni ska även undersöka och läsa in er på tillgängliga säkerhetsåtgärder mot *adversarial input attacks* och redogöra för dessa.

Uppgiften delas upp i följande två huvudsakliga delar: i) en teoretisk sammanfattning om adversarial input attacks och säkerhetsåtgärder som kan/bör vidtas, samt ii) en praktisk implementation av attacken och en eller flera försvarsmekanismer.

1.1 - Instudering

Innan ni implementerar attack och försvar för adversarial input attacks är det nödvändigt med en grundläggande förståelse för dessa. Ni ska här efterforska lite mer i detalj hur adversarial input attacks går till och vad det finns för skydd mot dem. Då det finns flera olika varianter räcker det med att välja en attacktyp och en skyddsåtgärd. Skriv ditt svar i rapportens del 1.1.

1.2 - Implementation av attack

I detta steg ska ni implementera en attack som gör så att bilden på koalan klassificeras som en traktor, utan att bilden ser annorlunda ut för mänskliga ögat. Mer info finns i Jupyter Notebooken under `Part 1`). Skriv er kod direkt i Notebooken och exekvera den där så att svaren syns i Notebooken när denna lämnas in. Så länge som ni kan förklara metoden får ni använda valfri metod, och såklart även redan tillgängliga paket, t.ex. Adversarial Robustness Toolbox (ART), för att lösa uppgiften.

1.3 - Säkerhetsåtgärder

I denna deluppgift ska ni välja ett så effektivt skydd som möjligt mot just er adversarial input attack. Eftersom det finns många olika typer av skydd ska ni motivera varför ni valt just det skydd ni gjort, istället för något annat skydd. Skriv ner era svar i rapportens del 1.3, och var noga med att motivera valet av skyddd väl.

1.4 - Implementation av skydd (betyg A eller B)

I rapportens del 1.4 summerar ni vilket skydd ni uppnådde då skyddsmekanismen hade implementerats. Det finns många olika varianter av skydd, och det är vanligt att skydden resulterar i en ny modell som är härdad mot attacker. Vid implementation av ett sådant skydd redovisar ni en jämförelse av resultaten

från er attack på originalmodellen med resultaten på den härdade modellen. Huvudsaken här är att ni tydligt kan uppvisa skyddets effekt och att ni kort kan redogöra för hur skyddsmekanismen fungerar.

Part 2: Attacks in federated learning scenario

In this task, you will use the FLOWER federated learning framework (`https://flower.ai/`) and the CIFAR-10 dataset (`https://flower.ai/docs/datasets/how-to-use-with-pytorch.html`) to simulate an attack scenario of your choice. The simulation will involve 50 communication rounds with 5 clients collaboratively training a global model. Run this setup without any attack in order to have it as the baseline to compare with later, and try to - Log the following metrics for each round:

1. Kappa
2. F1 Score
3. Accuracy
4. ROC value

Part 2.1: Mandatory task 1

Use the Flower framework to simulate the chosen attack and use FedAvg as an aggregation strategy to construct the final global model following two main scenarios: one with Independent and Identically Distributed (IID) data and another with non-IID data. Please check this link to how they do it in Flower: `https://flower.ai/docs/datasets/tutorial-use-partitioners.html`
In the first scenario, you must attack one client and log all the metrics mentioned during the training process. Then, increase the number of attack clients to two and compare both results against your baseline (federated learning without attacks), to evaluate the impact of the attacks on model performance. In the second scenario, repeat the process described in the previous scenario but with a non-IID data distribution across the clients and evaluate the model performance using the same metrics. Then, perform the same comparison you did previously. Analyze and compare the results obtained from previous scenarios (IID and non-IID), focusing on how the attacks affect model performance. Conclude the vulnerability of federated learning systems under different data distributions (IID vs. non-IID) when attacked by one or more clients.

Part 2.2: Mandatory task 2

In this task, you will repeat the same experiment as described previously, but instead of using FedAvg as the aggregation algorithm, you will use FedProx (Check Flower documentation regarding aggregation algorithms). The setup remains the same, with the Cifar10 dataset, the Flower framework, and both IID and non-IID data distributions. This will allow you to evaluate the impact of different aggregation algorithms on model performance, specifically comparing FedAvg and FedProx.

Part 2.3: Optional for A and/or B grade

Develop one of the defence techniques against your chosen attack, and compare the results with the baseline and the attacked model. Please analyse the results and draw a conclusion based on your own reflection.

Part 2.4: Optional task for possibility to get grade A

Develop another defence techniques against your chosen attack, and compare against the results obtained by the previous defence technique. Please analyse the results and draw a conclusion based on your own reflection.
Note: you still also need to do part 2.3 for an option to get grade A.

Good luck!