ADVERSARIAL LEARNING FOR RECOMMENDATION:

APPLICATIONS FOR SECURITY AND GENERATIVE TASKS – CONCEPT TO CODE

Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia and Felice Antonio Merra

25/09/2020 – Tutorial at ACM RecSys 2020, Virtual Event, Brazil

Polytechnic University of Bari , Bari, Italy

ABOUT US

Vito Walter ANELLI

@walteranelli

Yashar DELDJOO

@yashardel

Tommaso DI NOIA

@TommasoDiNoia

Felice Antonio MERRA

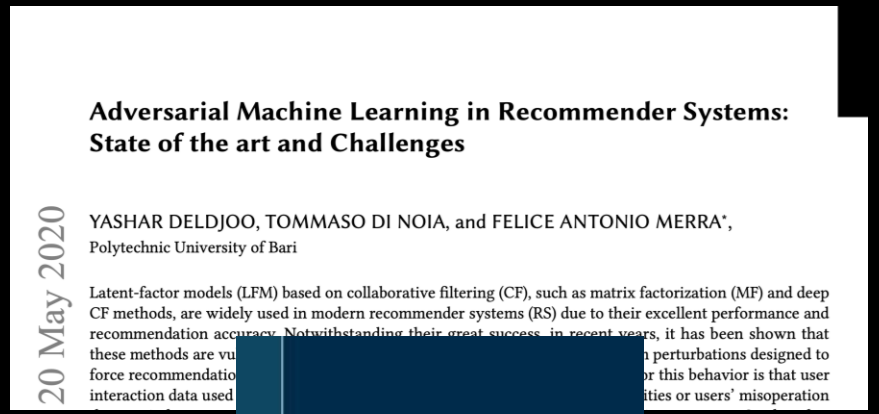
@merrafelice

SisInf Lab,
Polytechnic **U**niversity of **B**ari, Italy

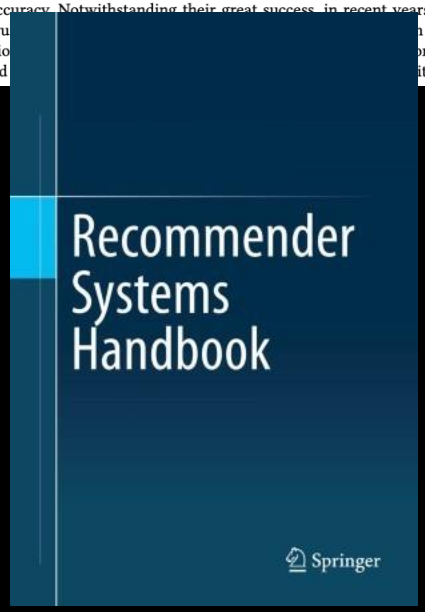
Politecnico di Bari

RELATED PUBLICATIONS FOR THIS TUTORIALS BY SISINFLAB

- **Survey**
  - submitted to Journal of ACM Computing Surveys in 2020 (arXiv version availiable)



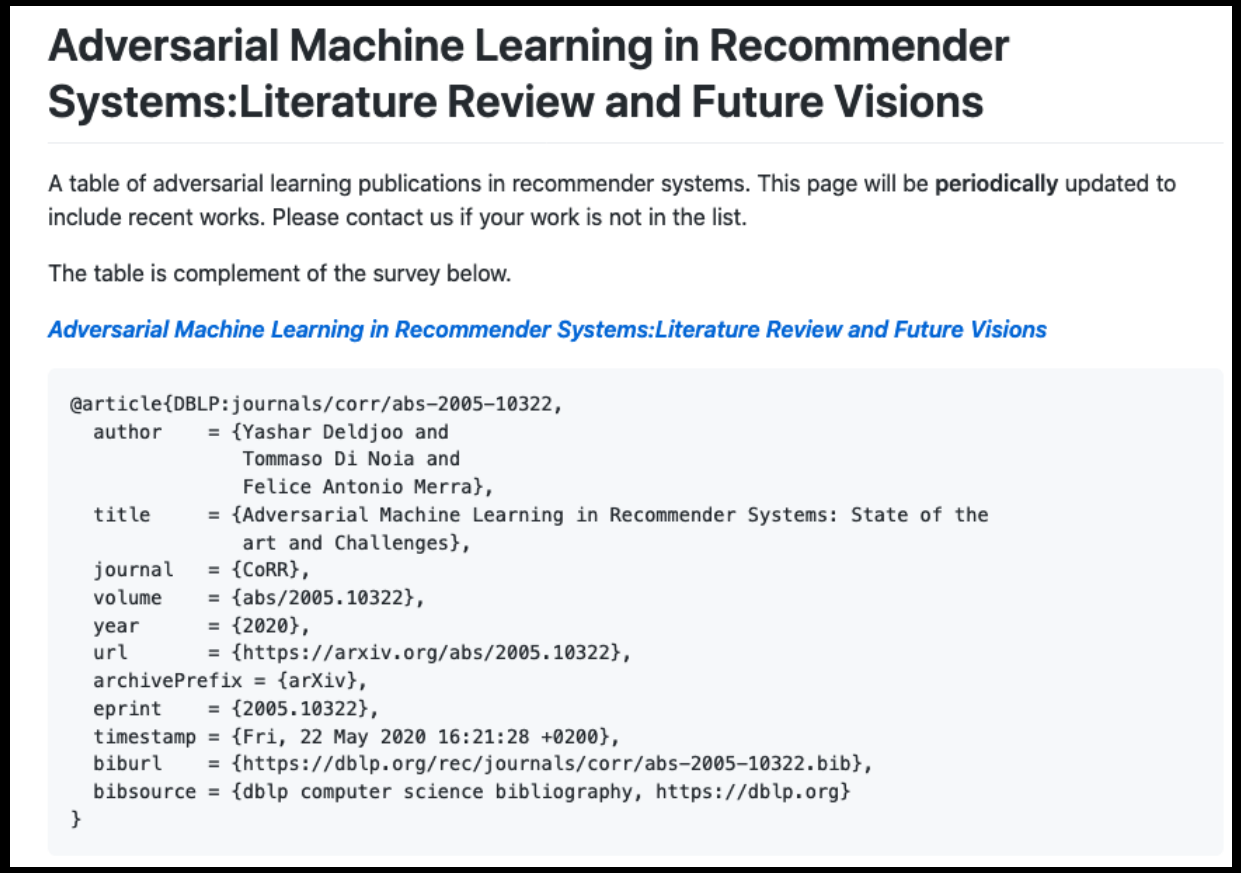
- **Book Chapter**
  - **Title: "**Adversarial Recommender Systems: Attack, Defense, and Advances"
  - Sep 2020, submitted to the 3rd Edition of Recommender System Handbook



Politecnico di Bari

UP TO DATE REPOSITORY

https://github.com/sisinflab/adversarial-recommender-systems-survey



Adversarial Machine Learning in Recommender Systems:Literature Review and Future Visions

A table of adversarial learning publications in recommender systems. This page will be **periodically** updated to include recent works. Please contact us if your work is not in the list.

The table is complement of the survey below.

Adversarial Machine Learning in Recommender Systems:Literature Review and Future Visions

```
@article{DBLP:journals/corr/abs-2005-10322,
  author    = {Yashar Deldjoo and
               Tommaso Di Noia and
               Felice Antonio Merra},
  title     = {Adversarial Machine Learning in Recommender Systems: State of the
               art and Challenges},
  journal   = {CoRR},
  volume    = {abs/2005.10322},
  year      = {2020},
  url       = {https://arxiv.org/abs/2005.10322},
  archivePrefix = {arXiv},
  eprint    = {2005.10322},
  timestamp = {Fri, 22 May 2020 16:21:28 +0200},
  biburl    = {https://dblp.org/rec/journals/corr/abs-2005-10322.bib},
  bibsource = {dblp computer science bibliography, https://dblp.org}
}
```

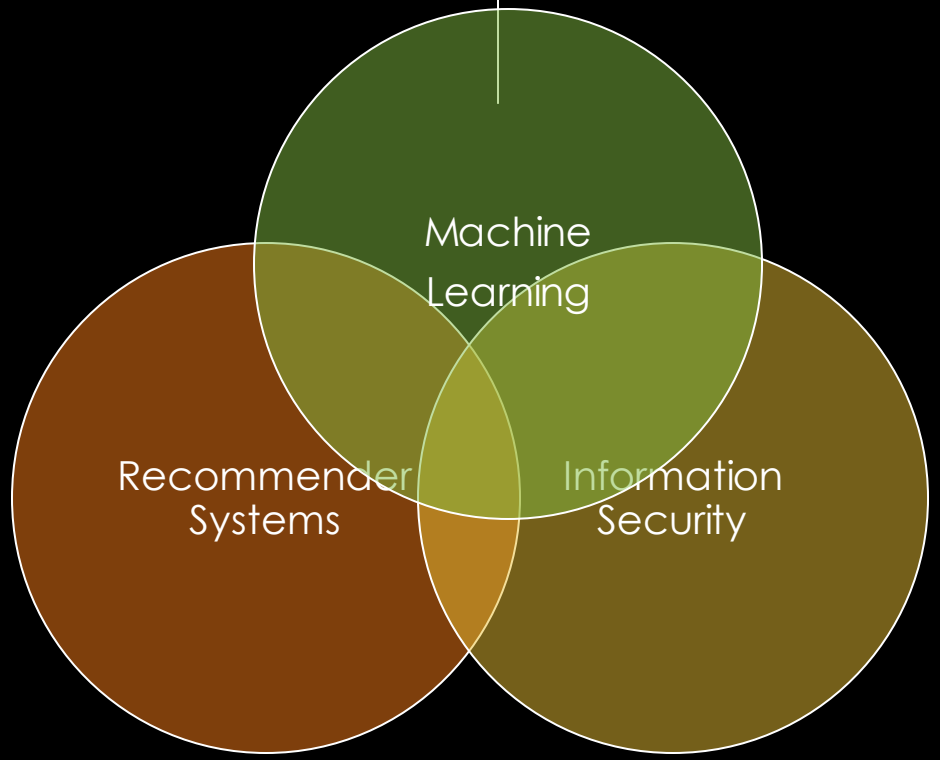
Politecnico di Bari

PLAN FOR TODAY

1. Part I: Introduction
    1. Goal, Statistics,
    2. Foundations
2. Part II: AML for Security of RS
    1. Foundations to AML for security of ML systems
    2. Adversarial Learning for attack-defense strategies in Recommender Systems
    3. Hands-on Session
3. Part III
    1. GAN-based Recommendation Framework (GAN-RF)
    2. Applications
    3. Hands-on Session

THE FOCUS OF OUR TUTORIAL

Adversarial Learning for RS security



Concept of Adversarial Learning USED in

Generative Adversarial Networks (GANs)

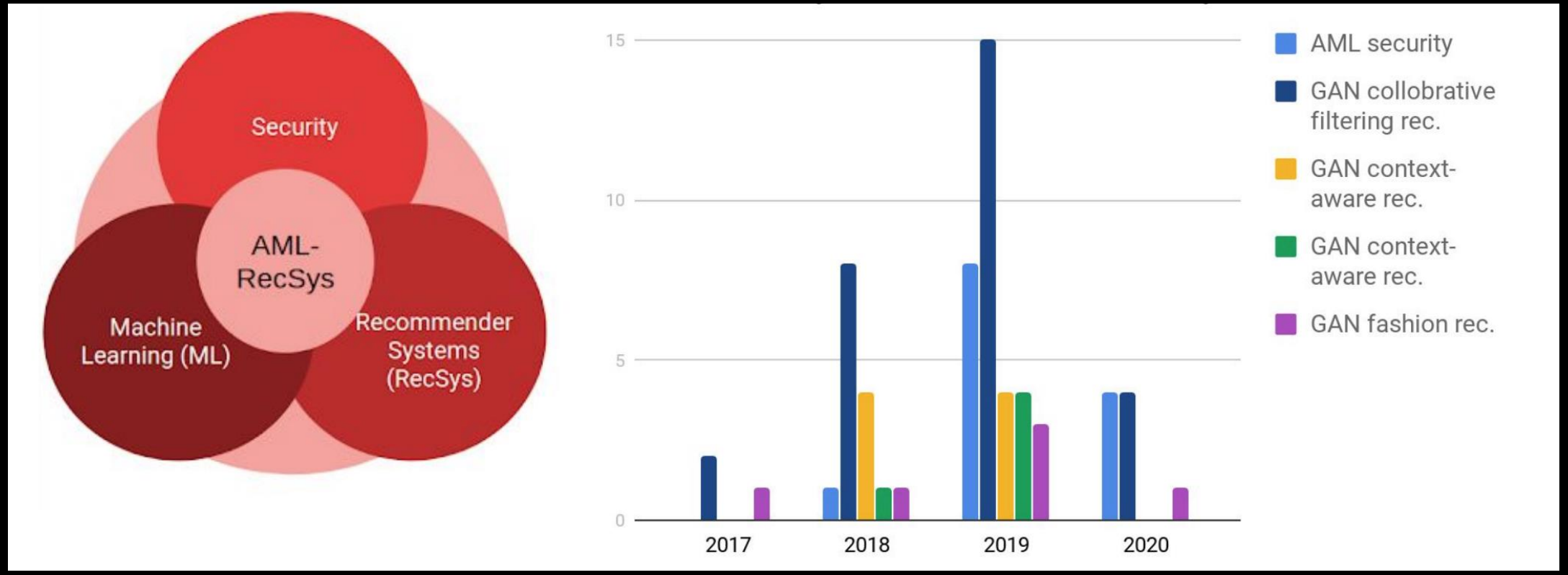
Politecnico di Bari

ADVERSARIAL LEARNING FOR RS

- More than **80** papers in the last **3** years
- Top Conferences/Journals involved

- WSDM
- SIGIR
- RecSys
- KDD
- WWW
- NIPS
- KDD
- IJCAI
- ICML
- CIKM
- AAAI
- TKDE
- ACL
- MM
- TOIS
- TIST
- CSUR
- …

AML FOR SECURITY OF AND GAN-BASED RS



Picture Source: Anell, Vito Walter, Deldjoo, Yashar, Tommaso Di Noia, and Felice Antonio Merra. "Adversarial Learning for Recommendation: Applications for Security and Generative Tasks—Concept to Code", Tutorial at RecSys'20

Politecnico di Bari

RECSYS + ADVERSARIAL LEARNING

Evaluation Goals

- Accuracy
- Coverage
- Confidence and Trust
- Novelty
- Serendipity
- Diversity
- Security
- Privacy
- Fairness
- Scalability

Increases the ability of learning in adversarial setting

Recomendation Models

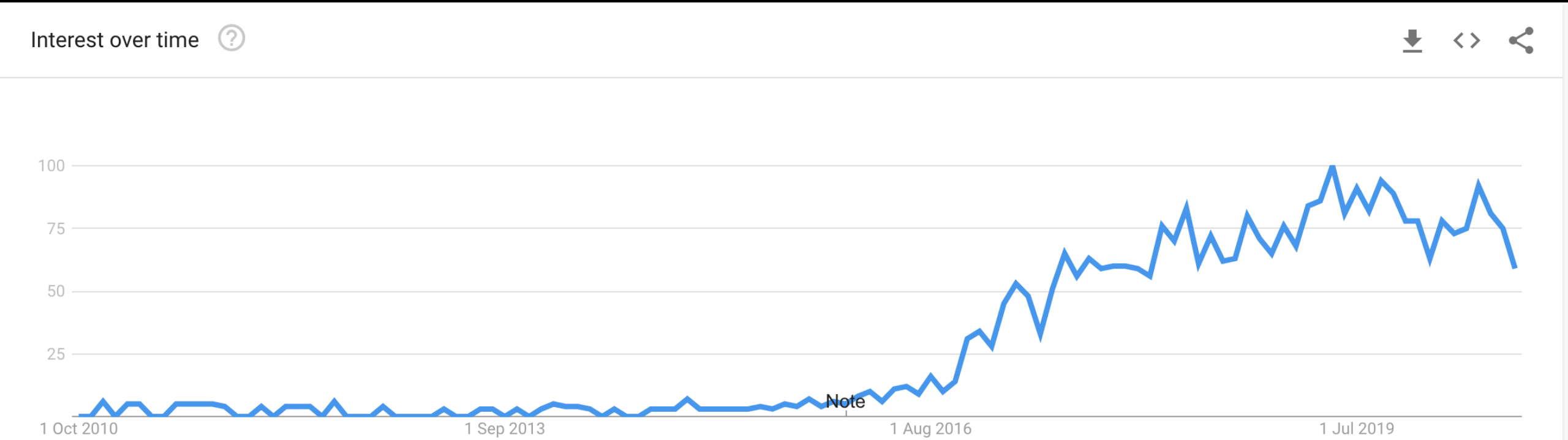
- Collaborative Filtering
  - Model-based
  - Memory-based
  - Graph-based
  - Deep
- Content-based Filtering
  - Metadata
  - Multimedia (audio and visual)
  - Knowledge-base
- Context-aware
  - Social
  - Temporal
- Hybrid

TUTORIAL GOALS

- Bridge the gap between advances made in the field of RecSys and Security
- Understand key concepts in adversarial machine learning
- Adversarial machine learning AND generative adversarial network (GANs)
- Present AML-RecSys hands-on for security and generative scenarios.
- Present future directions of AML for RecSys

Motivation

ADVERSARIAL LEARNING: SEARCH TERM FREQUENCY OVER TIME



Politecnico di Bari

[SOURCE GOOGLE TRENDS]

ADVERSARIAL LEARNING: SUGGESTED RELATED TOPICS

Related topics	Rising
1 Learning - Topic	Breakout
2 Generative adversarial networks - Topic	Breakout
3 Adversarial machine learning - Field of study	Breakout
4 Generative model - Topic	Breakout
5 Deep learning - Topic	Breakout

Related queries	Rising
1 deep learning	Breakout
2 adversarial networks	Breakout
3 adversarial machine learning	Breakout
4 generative adversarial networks	Breakout
5 gan	Breakout

Politecnico di Bari

[SOURCE GOOGLE TRENDS]

EVERYTHING STARTED WITH A PANDA

"panda"

+

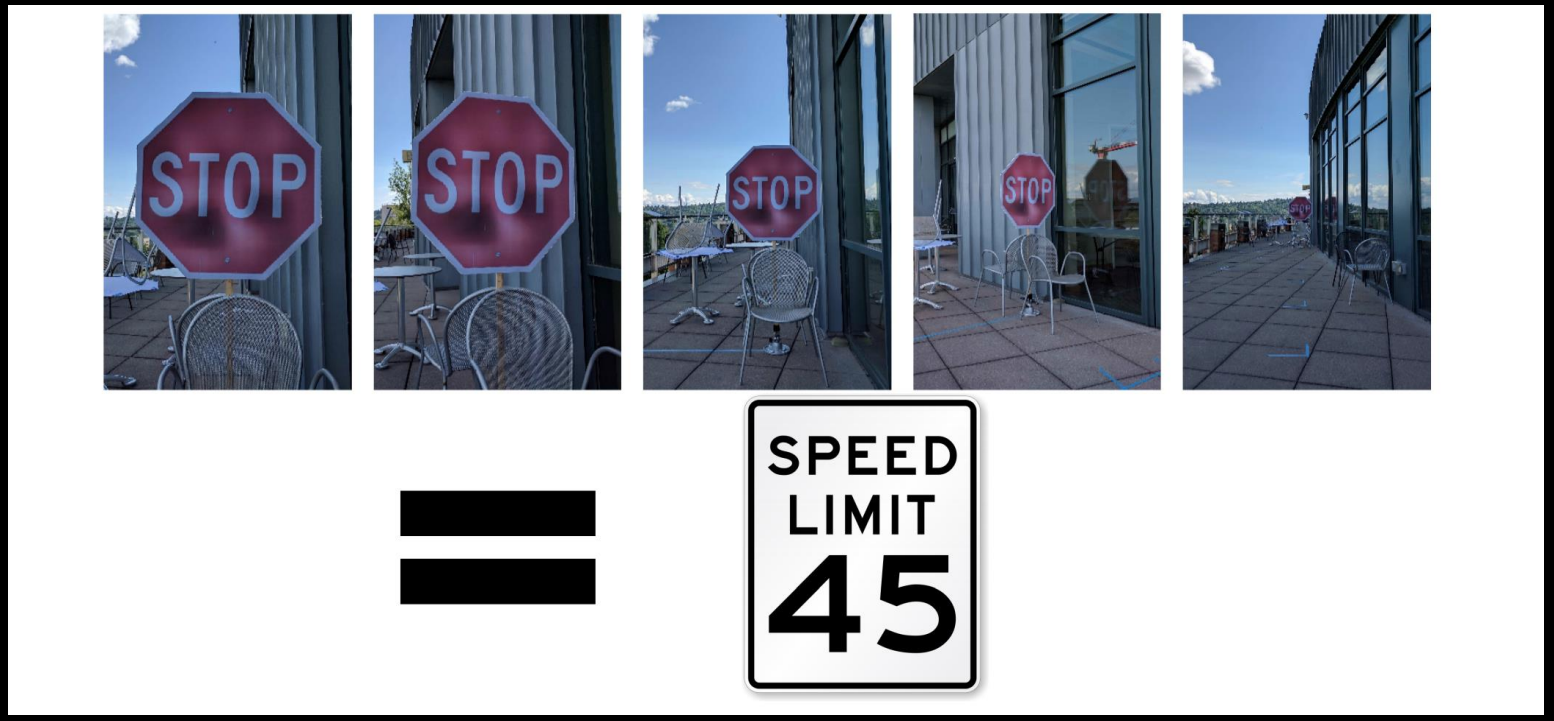
Adversarial Noise

=

"gibbon"

ADVERSARIAL EXAMPLES IN PHYSICAL WORLD

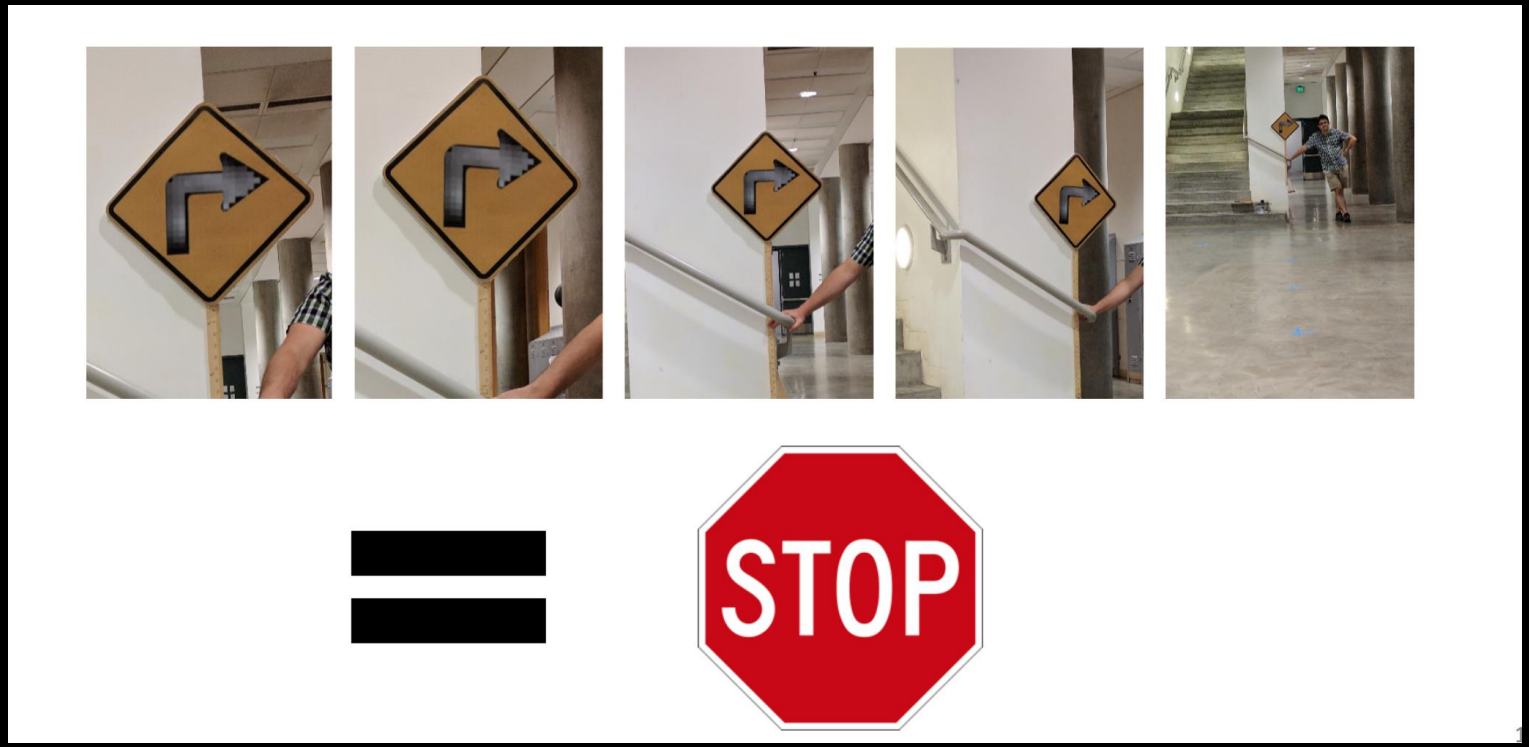
Subtle Perturbations



Source: Song, Dawn. "AI and Security: Lessons, Challenges & Future Directions", UC Berkeley, 2017
Evtimov, Ivan et. al. "Robust Physical-World Attacks on Machine Learning Models." arXiv preprint arXiv:1707.08945 (2017).

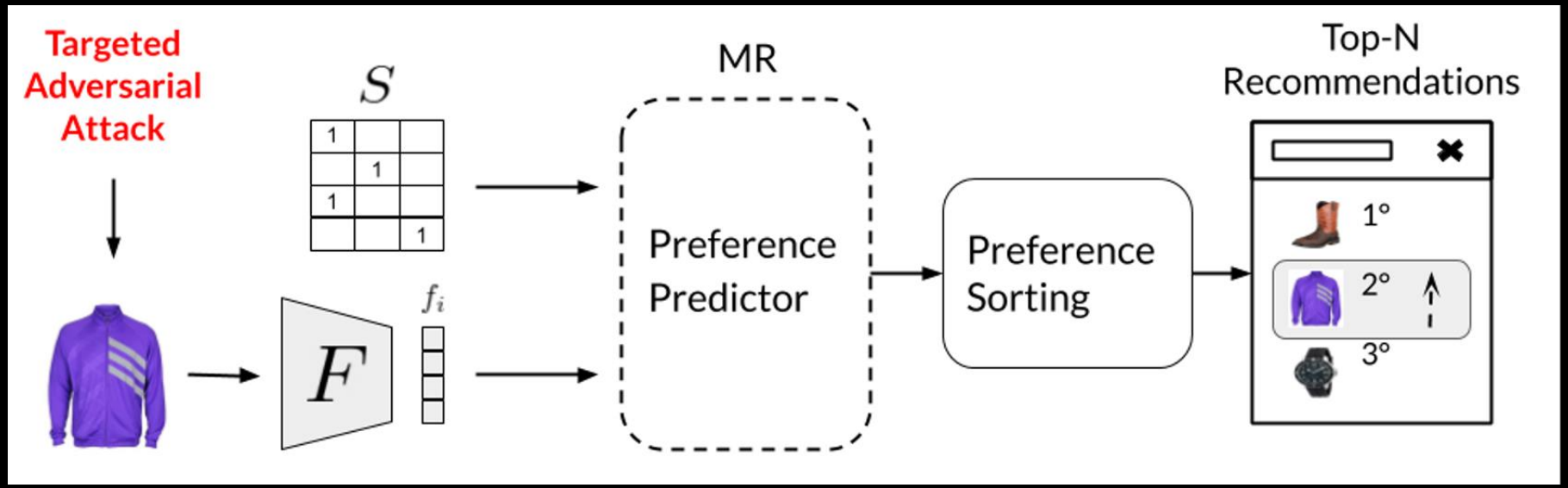
ADVERSARIAL EXAMPLES IN PHYSICAL WORLD

Subtle Perturbations



Source: Song, Dawn. "AI and Security: Lessons, Challenges & Future Directions", UC Berkeley, 2017
Evtimov, Ivan et. al. "Robust Physical-World Attacks on Machine Learning Models." arXiv preprint arXiv:1707.08945 (2017).

ADVERSARIAL EXAMPLES IN RS



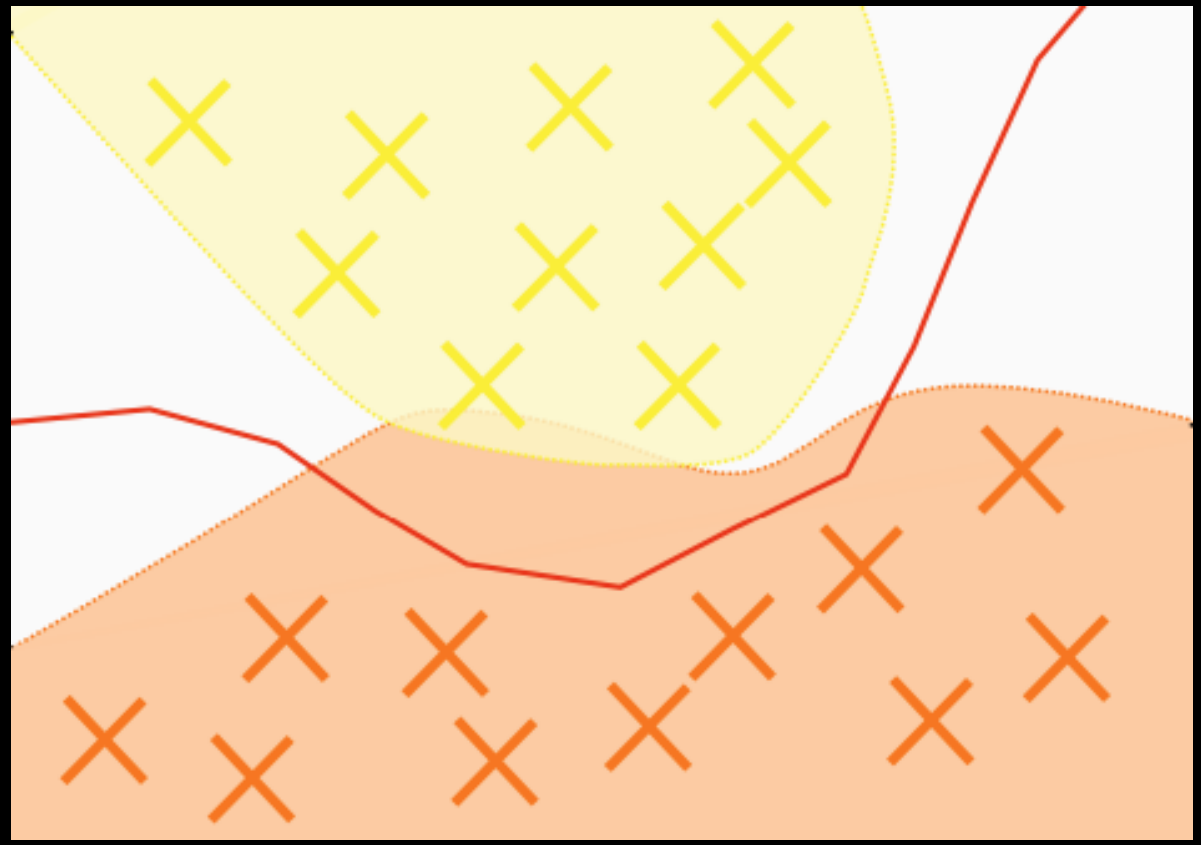
Simulation of Targeted Adversarial Attacks against Multimedia Recommender Systems can push low recommended product categories even **3 times more recommended** by **perturbing** product images in a **human-imperceptible way.**

[Di Noia, Tommaso, Daniele Malitesta, and Felice Antonio Merra. "TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems." the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-DSML'20). 2020.]

Politecnico di Bari

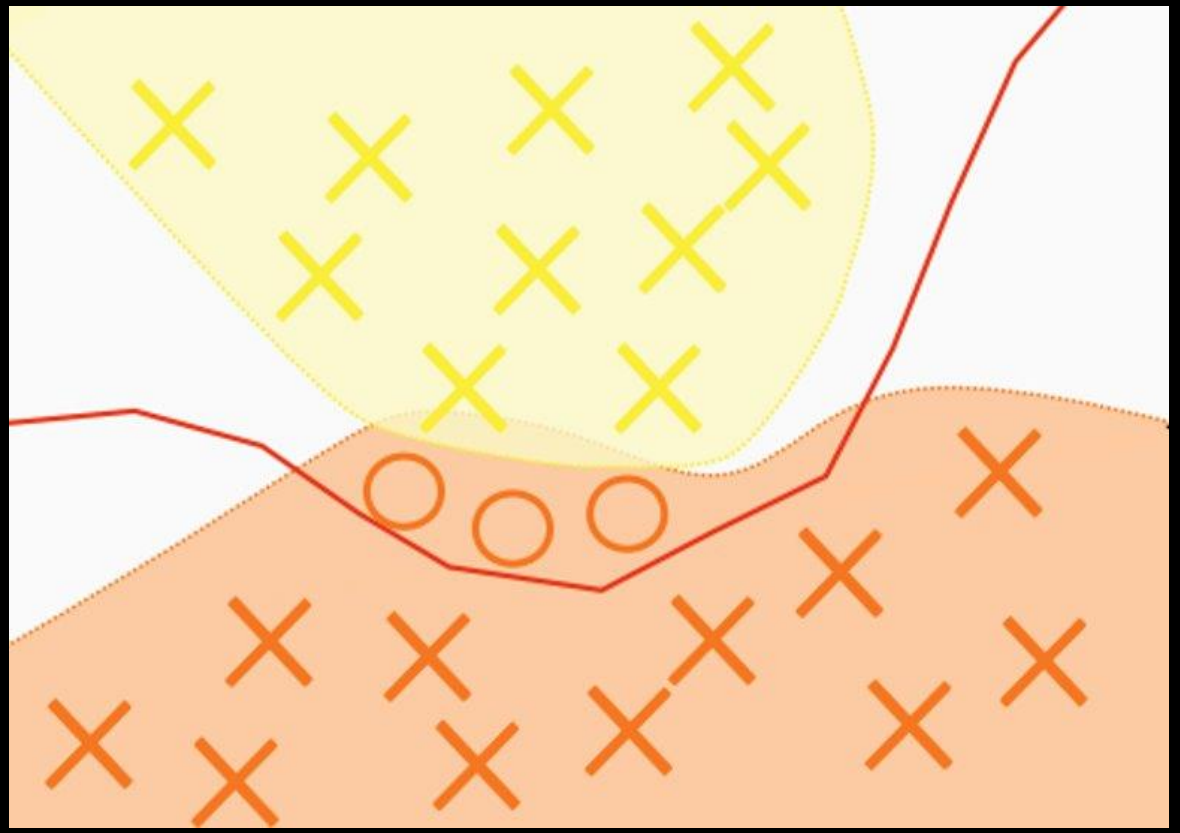
WHAT IS THE ORIGIN OF SUCH FAILURE?

IDEA: DECISION BOUNDARY OF THE MODEL



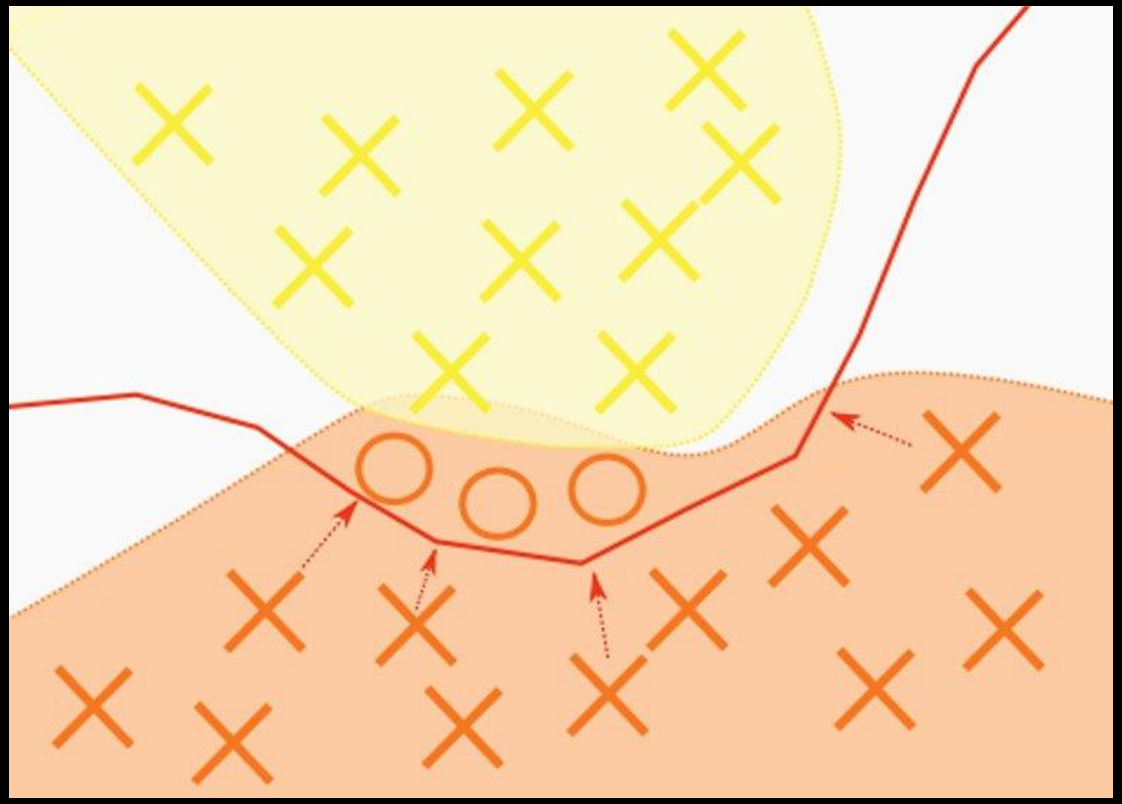
The **learned model** slightly differs from the true data distribution …

IDEA: THE SPACE OF ADVERSARIAL EXAMPLES



…. which makes room for adversarial examples.

ATTACK: USE THE ADVERSARIAL DIRECTIONS



Most attacks try to move inputs across the boundary.

Attacking with a random noise does not work well in real experiment.

IT'S ALL ABOUT DATA

«Yet, we found that adversarial examples are relatively robust, and are shared by neural networks with varied number of layers, activations or trained on different subsets of the training data.

That is, if we use one neural net to generate a set of adversarial examples, we find that these examples are still statistically hard for another neural network even when it was trained with different hyperparameters or, most surprisingly, when it was trained on a different set of examples»

[C. SZEGEDY ET AL. INTRIGUING PROPERTIES OF NEURAL NETWORKS, ICLR'14]

ACTUALLY NOT REALLY…

«… adversarial examples can be directly attributed to the presence of *non-robust features*: features (derived from patterns in the data distribution) that are highly predictive, yet brittle and (thus) incomprehensible to humans.»

«*Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data.*»

«…this perspective establishes adversarial vulnerability as a human-centric phenomenon, since, from the standard supervised learning point of view, non-robust features can be as important as robust ones »

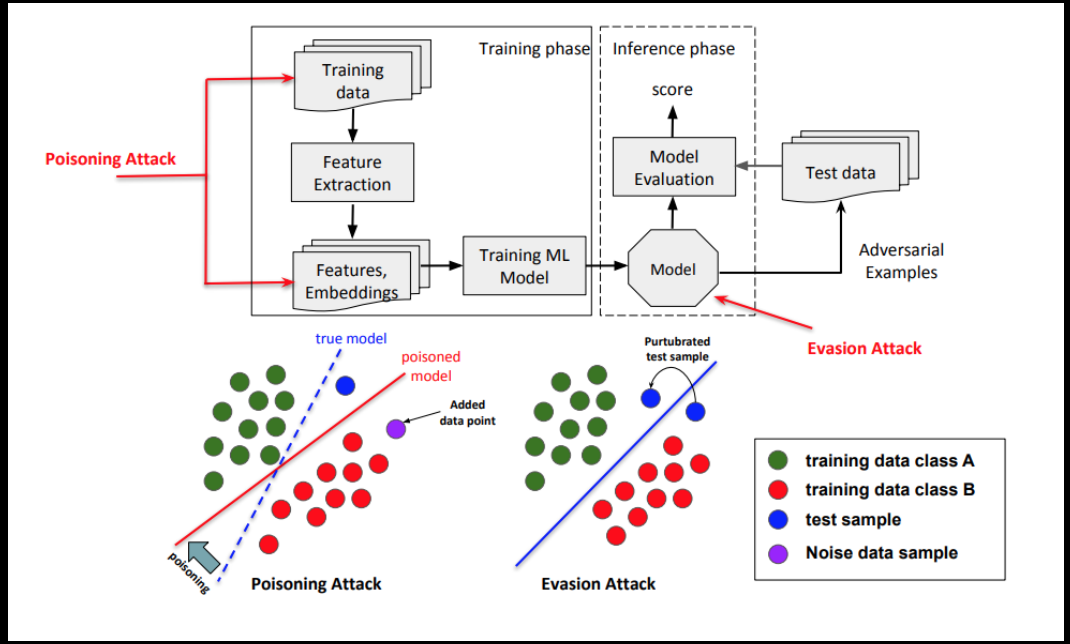
[A. ILYAS ET AL. ADVERSARIAL EXAMPLES ARE NOT BUGS, THEY ARE FEATURES, NIPS'19]

Foundations

TAXONOMY OF ADVERSARIAL ATTACKS

(1) Attack's timing:

- Training time (data poisoning)
  - occur before the ML model is trained or in the inference phase
- Inference time (evasive attack)
  - occur after the ML model is trained or in the inference phase
  - aim to evade detection — or evade the decisions made by the learned model

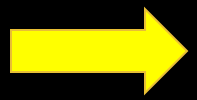


Picture Source: Deldjoo, Yashar, Tommaso Di Noia, and Felice Antonio Merra. "Adversarial Machine Learning in Recommender Systems: State of the art and Challenges. CoRR abs/2005.10322 (2020)." *arXiv preprint arXiv:2005.10322* (2020).

Politecnico di Bari

POISONING ATTACK APPLICATION IN LITERATURE

- Attack on Binary Classification
  - Label flipping attack
  - Kernel SVM
- Attack on unsupervised learning
  - Clustering
  - Anomaly detection
- Attack on matrix completion
    - Alternating minimization
      - Projected gradianet decent (PGA)
    - Nuclear norm normalization
    - Mimicking user behavior



Popularized by Bioggio et al

Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389.

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

POISIONING ATTACK APPLICATION IN LITERATURE

- Attack on Binary Classification
  - Label flipping attack
  - Kernel SVM
- Attack on unsupervised learning
  - Clustering
  - Anomaly detection
- Attack on matrix completion
  - Hand Engineered Poisoning
  - Mimicking user behavior
  - ML-optimized Strategies

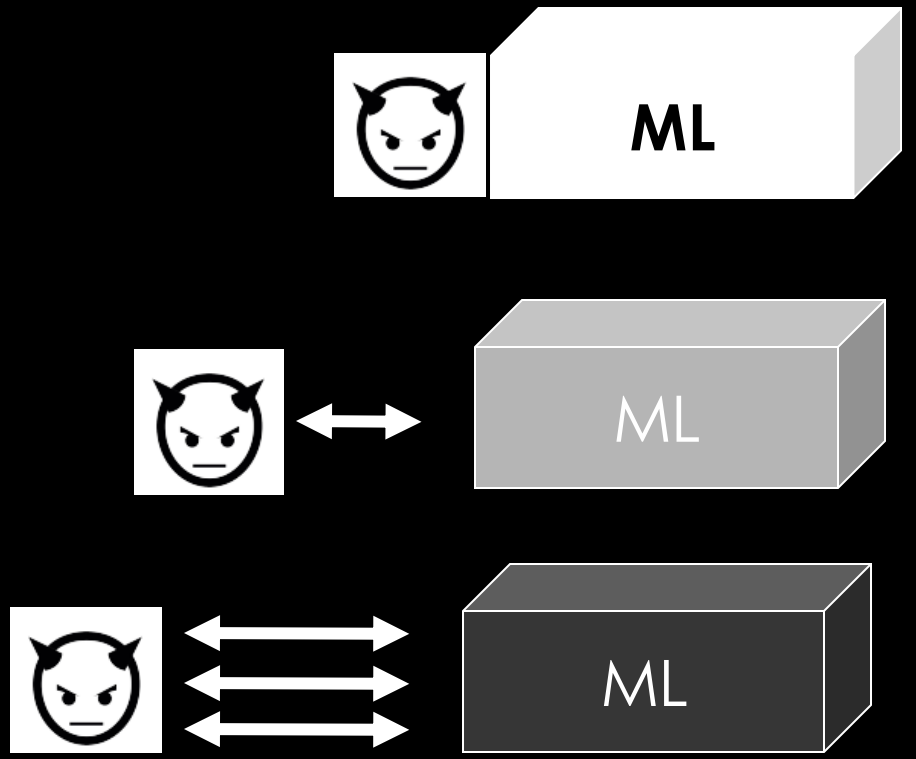
Commonly known as "shilling attack" in RecSys.

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

TAXONOMY OF ADVERSARIAL ATTACKS

(2) Attacker's knowledge:

- White-box attack
- Gray-box attack
- Black-box attack



Politecnico di Bari

TAXONOMY OF ADVERSARIAL ATTACKS

(3) Attacker's goal:

- Targeted attack: forces the classifier (ML model) to make predictions into a target class label.
- Untargeted attack (reliability attack): forces the classifier to make predictions into any incorrect class label.

CLASSICAL MACHINE LEARNING

Supervised learning (classification) problem

$$\min_{\Omega} J(\Omega, x, y)$$

ADVERSARIAL PERSPECTIVE

Supervised learning (classification) problem

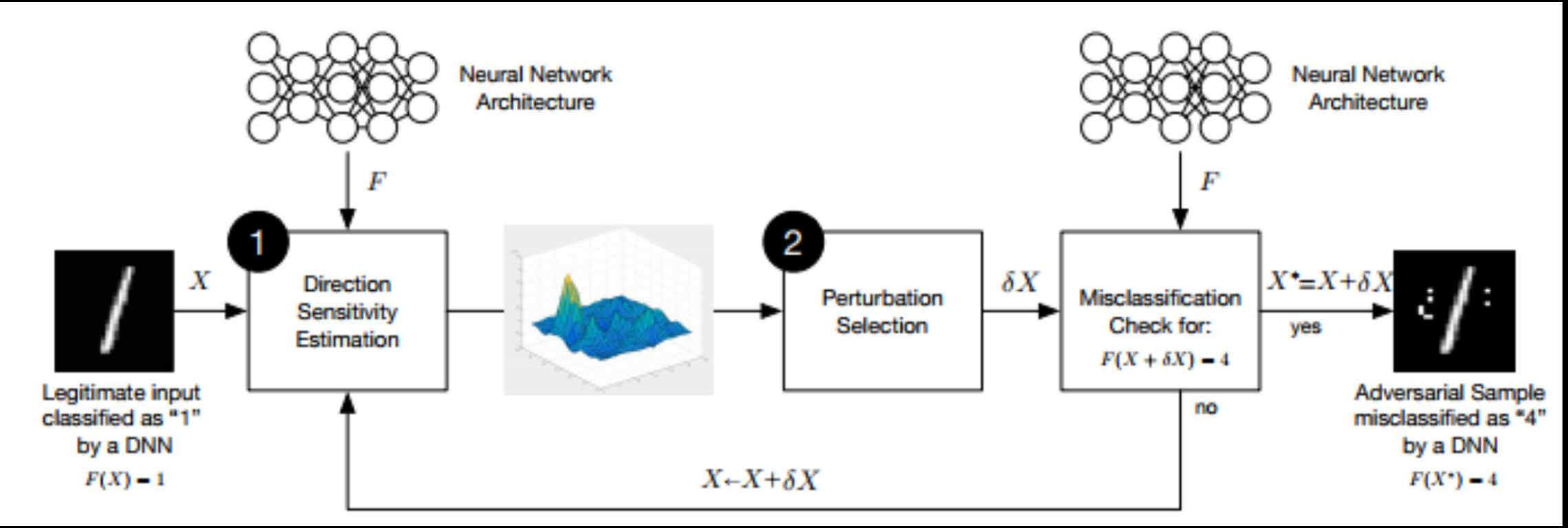
$$\arg\max_{\Delta_{adv}} J(\Omega, x + \Delta_{adv}, y) \;\; s.t., \|\Delta_{adv}\|_p \leq \epsilon$$

Adversarial perturbation of sample x

perturbation budget

Algorithms that aim to find such adversarial perturbations are referred to as adversarial attacks.

THE FRAMEWORK



CREDITS: N. Papernot et al. *Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks*. IEEE Symposium on Security and Privacy, SP 2016

ATTACK METHODS IN COMPUTER VISION DOMAIN

- **L-BFGS** [Szegedy et al., 2013]
  - Uses Limited-memory BFGS (L-BFGS) algorithm to solve the problem with **linear memory requirement**
- **FSGM -** Fast Gradient Sign Method [Goodfellow et al., ICLR '15]
  - Use the **gradient** of the loss function to work on the costraint problem.
- **Carlini-Wagner** [Carlini and Wagner, 2017a]
  - Refine the L-BFGS attack to **defeat Defensive distillation**
- **JSMA -** Jacobian Saliency Map Attack [Papernot et al., 2015a]
  - Construct an **input-output mapping** (forward derivatives) to find the minimal perturbation
- **DeepFool** [Moosavi-Dezfooli et al., 2015]
  - Perform an iterative attack to find **the closest decision boundary** to a given input

FGSM(INTUITION 1)
[GOODFELLOW ET AL., ICLR'15]

$$x' = x_0 + \Delta_{adv}$$

$$\omega^T \cdot x' = \omega^T \cdot x_0 + \omega^T \cdot \Delta_{adv}$$

GOAL: Maximize the factor $\omega^T \cdot \Delta_{adv}$ under the constraint $\|\Delta_{adv}\|_\infty \leq \epsilon$

IDEA: $\Delta_{adv} = \epsilon \cdot sign(\omega)$

FGSM(INTUITION 2)
[GOODFELLOW ET AL., ICLR'15]

$$\omega^T \cdot \Delta_{adv} = \epsilon \cdot (|\omega_1| + \cdots + |\omega_n|)$$

If we consider $m = avg(\omega_i)$ then the influence of the perturbation $\omega^T \cdot \Delta_{adv}$ grows as $\epsilon \cdot m \cdot n$

RESULT: the perturbation linearly grows with n

FGSM
[GOODFELLOW ET AL., ICLR'15]

Let Ω be the parameters of a ML model, x the input to the model, y the label of x, and $J(\Omega, x, y)$ be the loss function used for the model.

FSGM linearizes J around the fixed value of Ω, obtaining an optimal max-norm constrained adversarial perturbation.

$$\Delta_{avd} = \epsilon \cdot \mathrm{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\Omega}, \boldsymbol{x}, y)\right)$$

UNTARGETED ADVERSARIAL ATTACK

$$\min_{\Delta_{adv}} \|\Delta_{adv}\|$$

$$s.t.: f(x_0 + \Delta_{adv}, \Omega) \neq y_0$$

Constrained optimization problem formulation

$$=$$

$$\max_{\|\Delta_{adv}\| \leq \epsilon} J(\Omega, x_0 + \Delta_{adv}, y')$$

UnConstrained optimization problem formulation

Politecnico di Bari

TARGETED ADVERSARIAL ATTACK

$$\min_{\Delta_{adv}} \|\Delta_{adv}\|$$

$$s.t.: f(x_0 + \Delta_{adv}, \Omega) = y_T$$

Constrained optimization problem formulation

=

$$\max_{\|\Delta_{adv}\| \leq \epsilon} J(\Omega, x_0 + \Delta_{adv}, y_T)$$

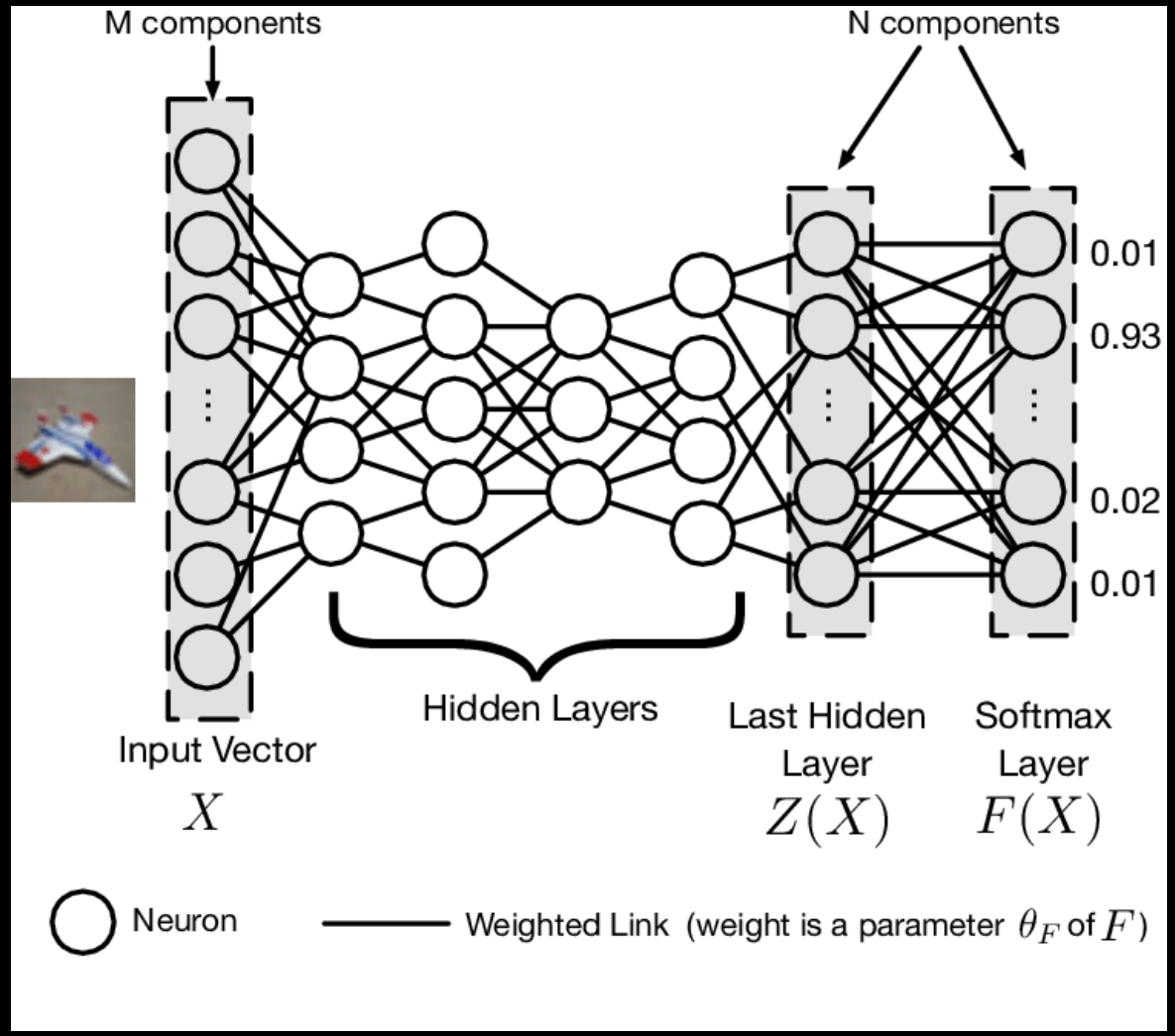
UnConstrained optimization problem formulation

COUNTERMEASURES

- **Proactive** countermeasures
  - **Adversarial Training** [Goodfellow et al., ICLR '15]
    - Additional training epochs with adversarial exmaples
  - **Defensive Distillation** [Papernot et al., ISS'16]
    - Adapt distillation to increase the robusteness of the network
  - **Robust Optimization** [Madry et al., ICLR'18]
    - design robust DNN to prevent a speciic class of adversarial examples
- **Reactive** countermeasures
  - **Adversarial Detecting**
  - **Input Reconstruction**
  - **Network Verification**

DEFENSIVE DISTILLATION

[PAPERNOT ET AL., ISS'16]

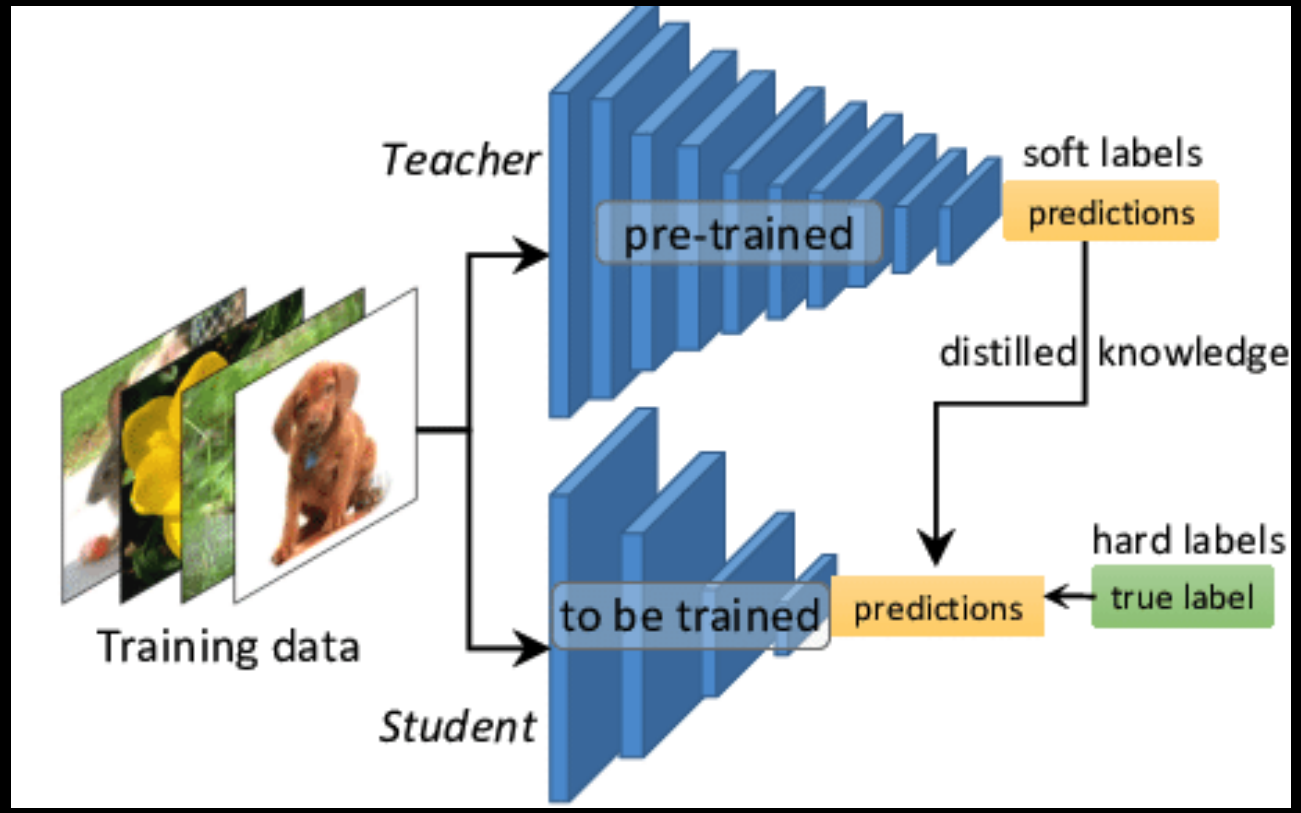


$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

Politecnico di Bari

DEFENSIVE DISTILLATION

[PAPERNOT ET AL., ISS'16]

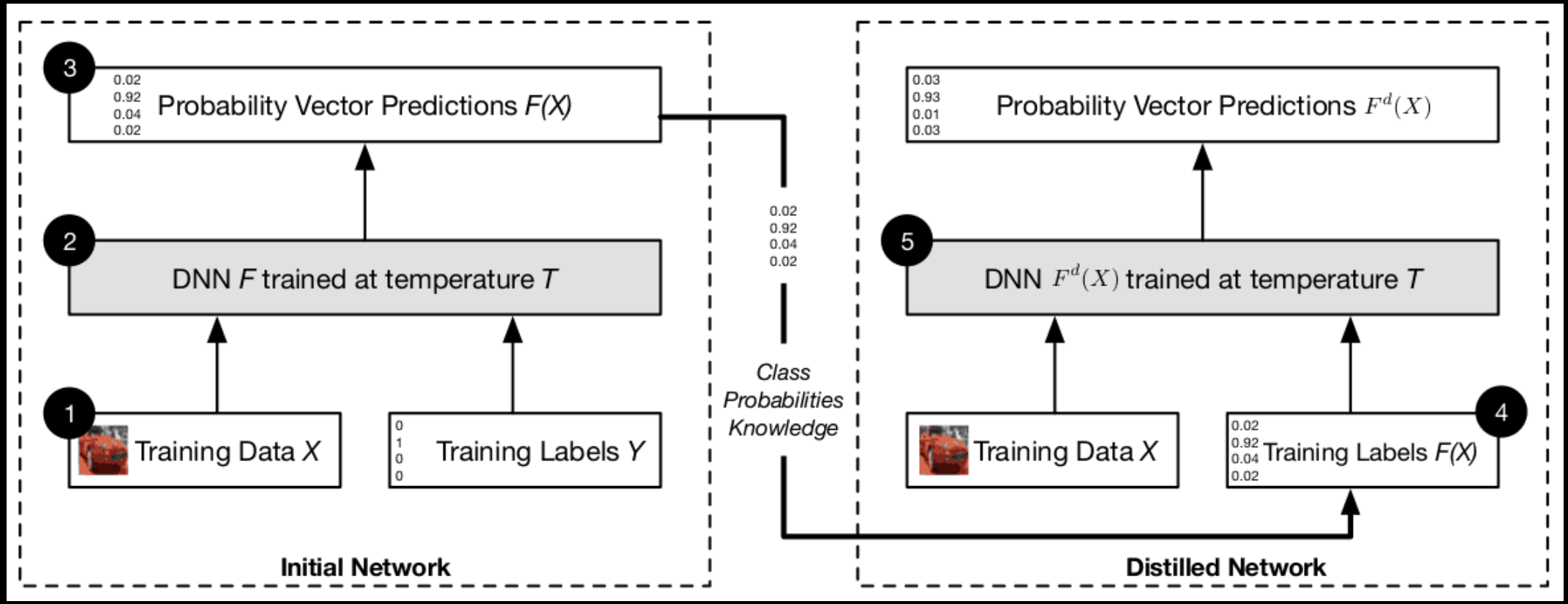


Credits: https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Originally proposed for model compression

DEFENSIVE DISTILLATION

[PAPERNOT ET AL., ISS'16]



Politecnico di Bari

ADVERSARIAL ROBUSTNESS

Supervised learning (classification) problem

$$\min_{\Omega} \max_{\Delta_{adv}} J(\Omega, x + \Delta_{adv}, y)$$

The above min-max formulation is used to capture the notion of security against adversarial attacks = Robust Optimization

ADVERSARIAL TRAINING
[GOODFELLOW ET AL., ICLR'15]

Including adversarial samples in the **training** of a model makes it **more robust**.
The objective function of the model **adversarially-trained** is:

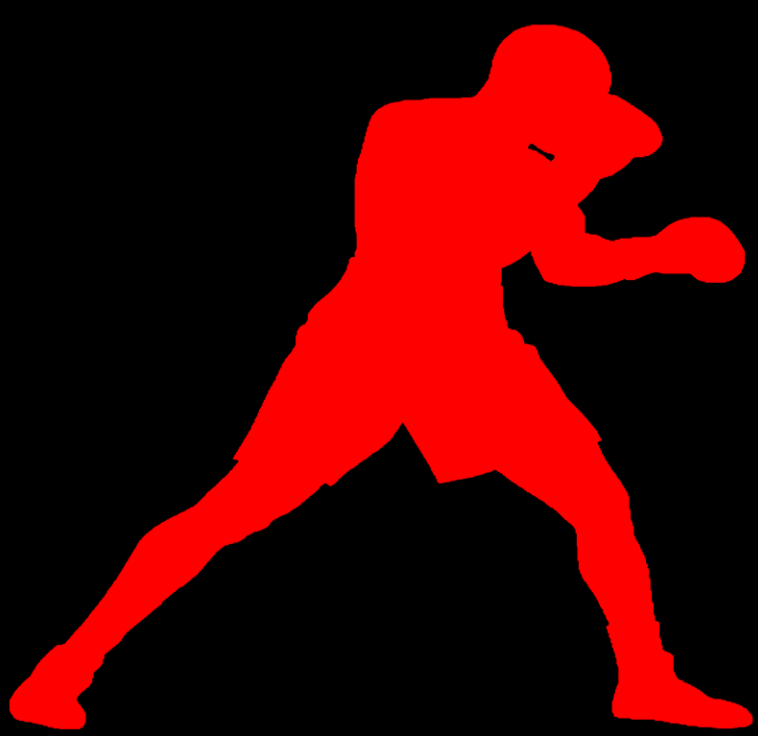
Adversarial Perturbation

$$J(\boldsymbol{\Omega}, \boldsymbol{x}, y) + \lambda J(\boldsymbol{\Omega}, \boldsymbol{x} + \epsilon \cdot \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\Omega}, \boldsymbol{x}, y)))$$

Adversarial Regularization term

Adversarial training provides better generalization performance
[Miyato et al., ICLR'17]

ATTACK-DEFENSE GAME

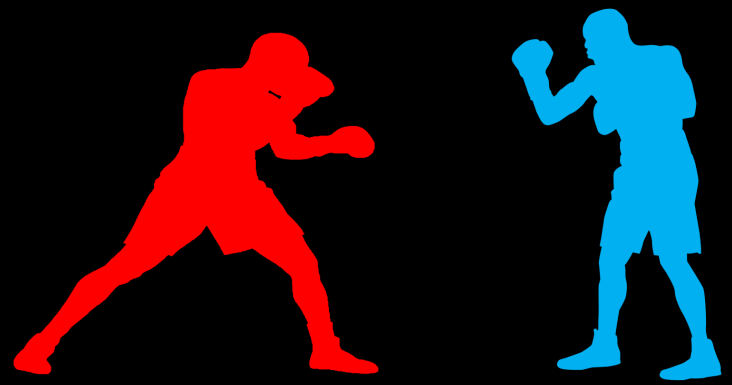


Politecnico di Bari

ATTACK-DEFENSE GAME

The attack-defence game is a **MINIMAX GAME:**

- in the **literature** for each **ATTACK** there is a **DEFENSE** strategy
- in the **adversarial training** we **MINIMIZE** the Loss and **MAXIMIZE** the perturbation
- **Generative Adversarial Network (GAN)** is trained with an **ATTACKER** that tries to alter a **DEFENDER**



Politecnico di Bari

WHY STUDY GAN?

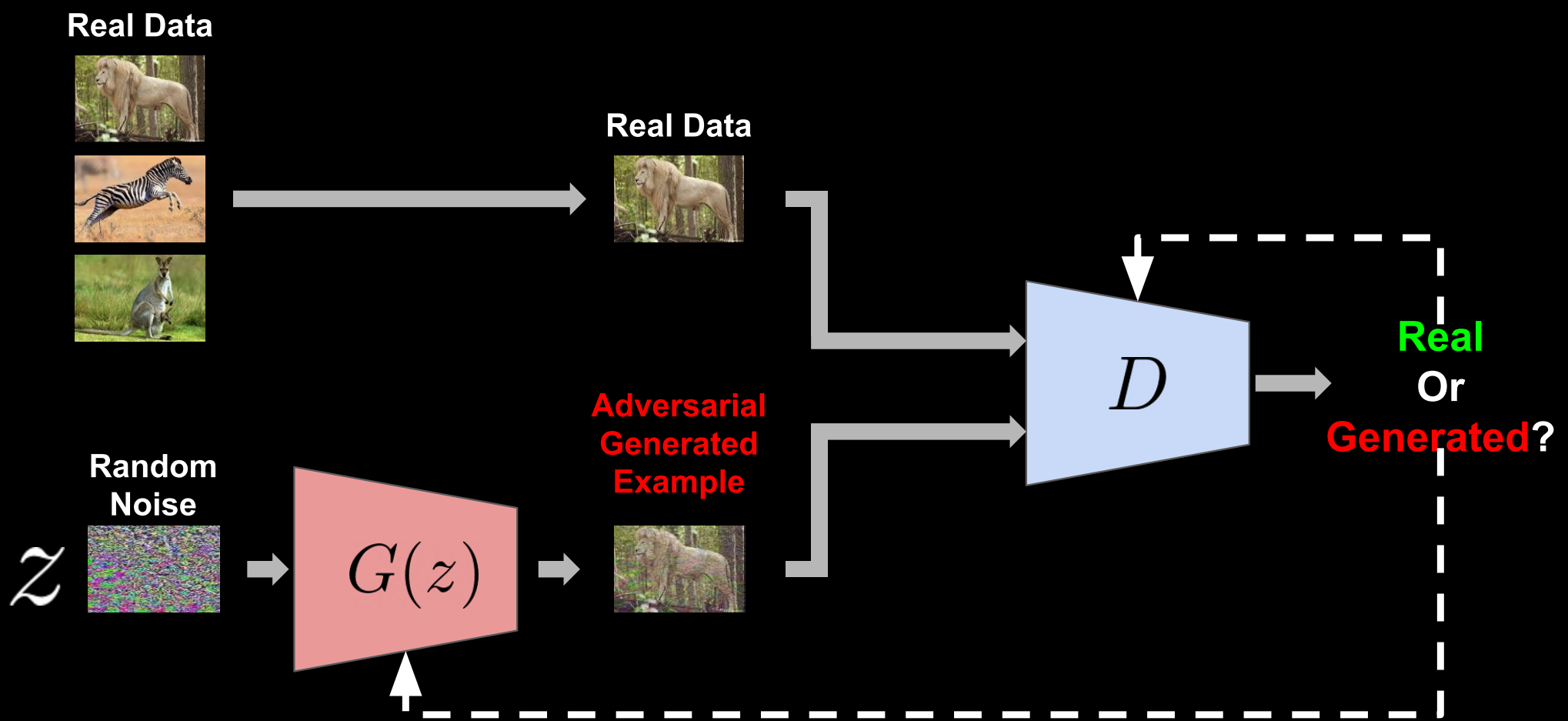
- Generate Examples for Image Datasets
- Generate Photographs of Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Pose Guided Person Image Generation
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation

- Photos to Emojis
- Photograph Editing
- High-resolution image synthesis
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting (Style-transfer)
- Clothing Translation
- Video-motion Prediction
- 3D Object Generation

HOW DO GANS WORK?

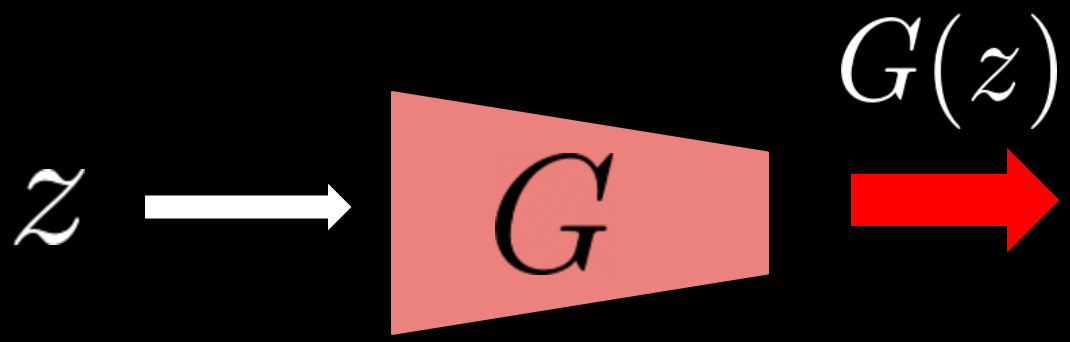
Normal Classifier

HOW DO GANS WORK?



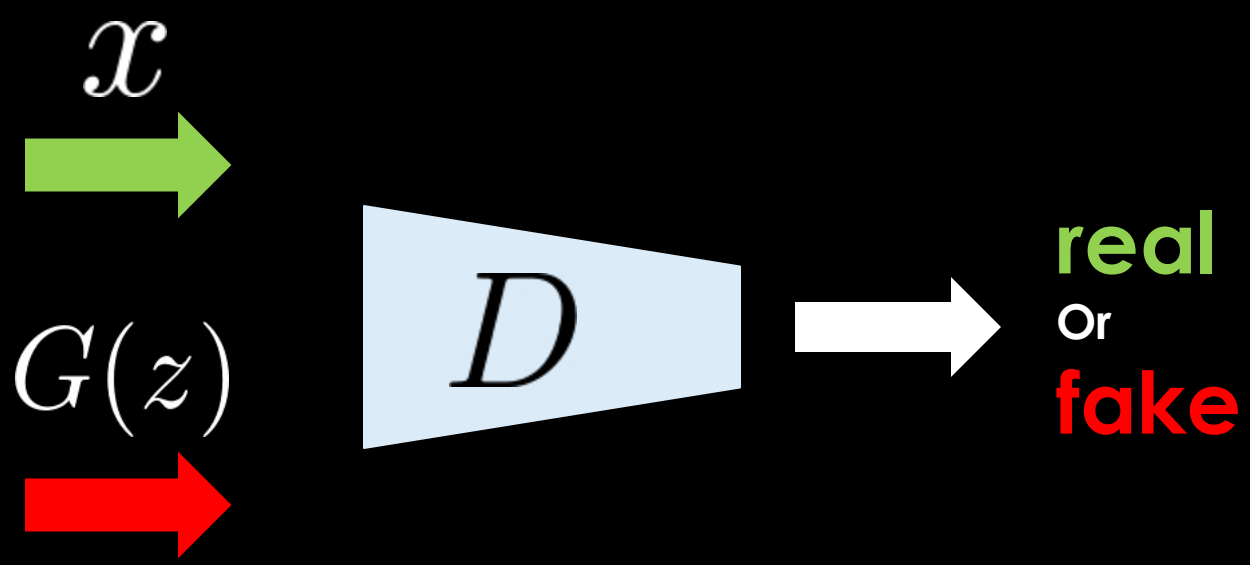
GENERATOR

GOAL? Maps a gaussian random noise z to a **Fake** data to **FOOL** the discriminator

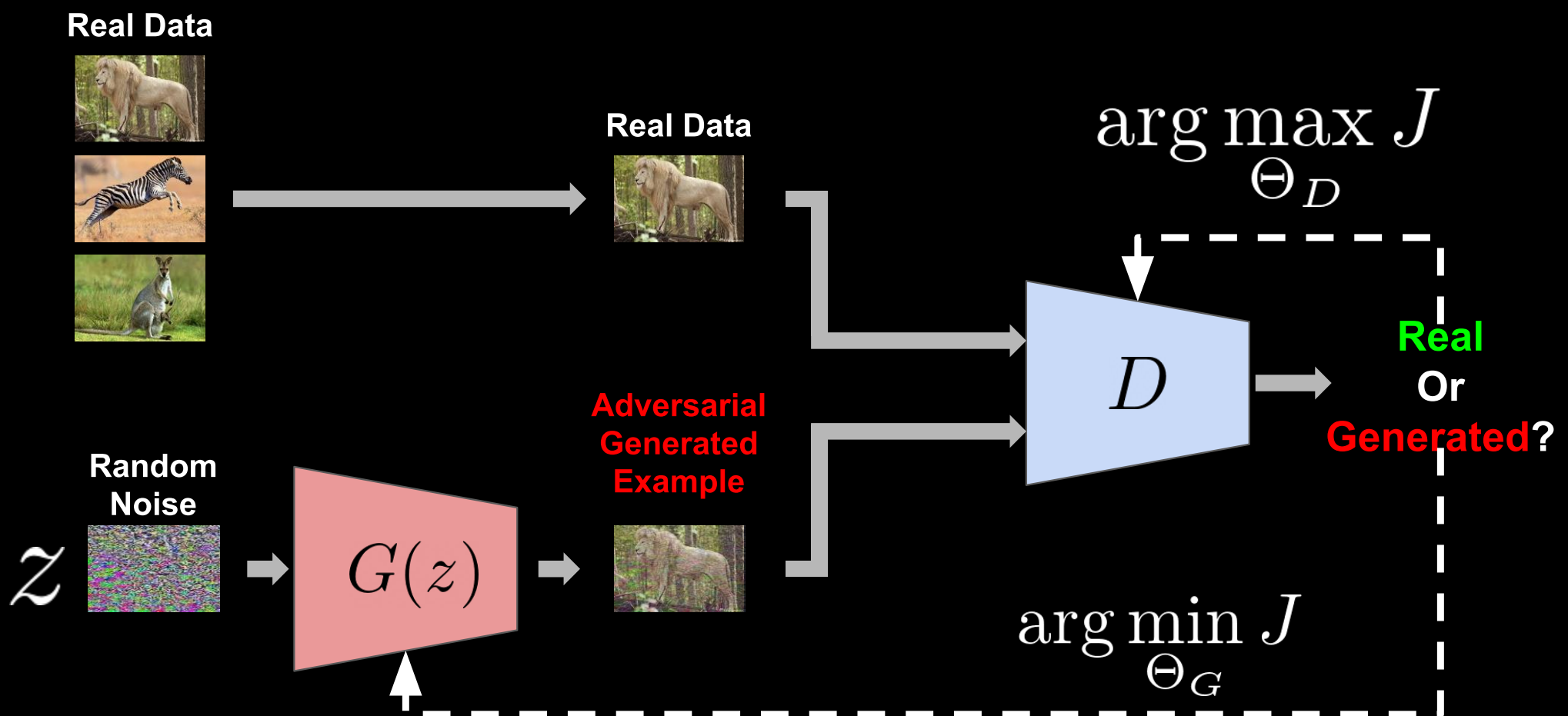


DISCRIMINATOR

GOAL? Output the probability that its input is **real** rather than **fake**



HOW DO GANS WORK?



Politecnico di Bari

MINIMAX GAME

$$J = \mathbb{E}_{\boldsymbol{x} \sim \rho_{\text{data}}\,(\boldsymbol{x})} \left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(z)} \left[\log\left(1 - D\left(G(\boldsymbol{z})\right)\right)\right]$$

DISCRIMINATOR

$$\arg\max_{\Theta_D} J$$

GENERATOR

$$\arg\min_{\Theta_G} J$$

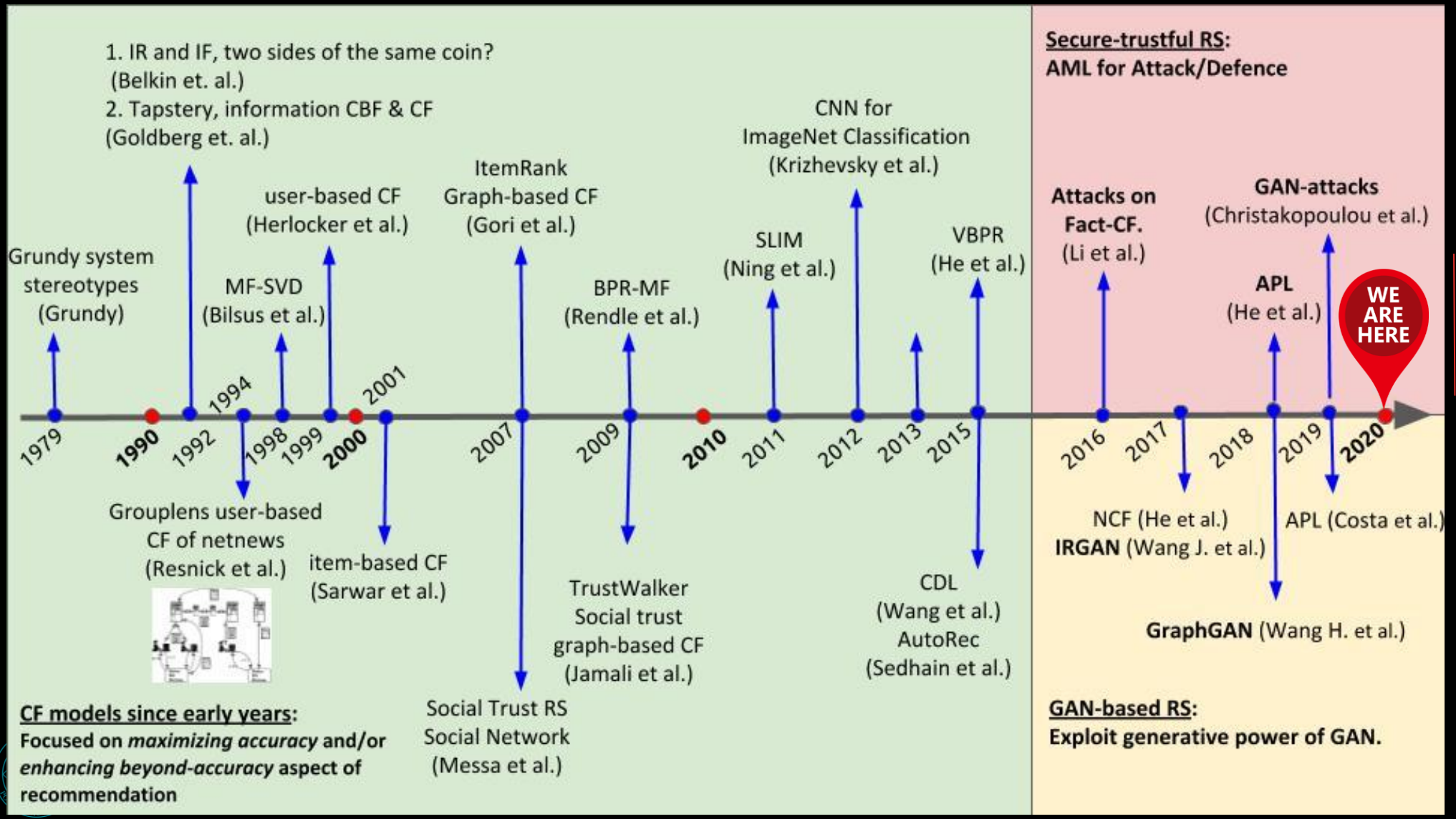
MINIMAX GAME

$$\arg\min_{\Theta_G} \max_{\Theta_D} J$$

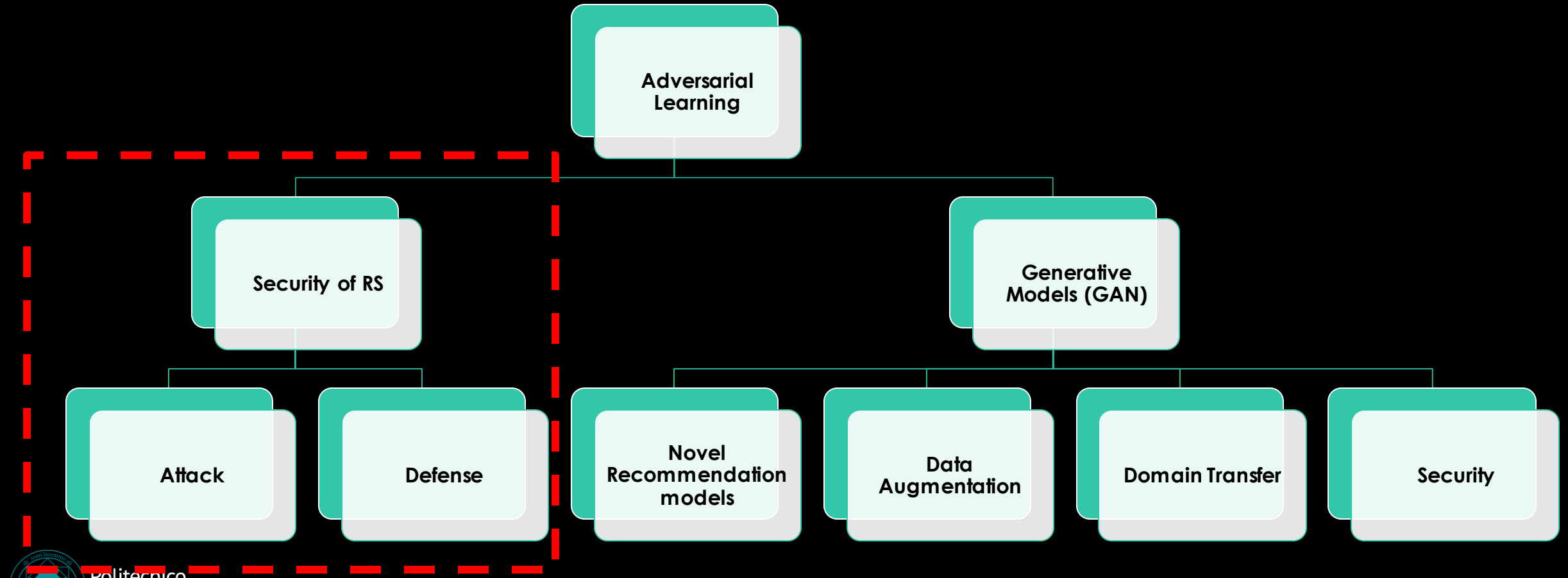
Politecnico di Bari

PART 2
ADVERSARIAL LEARNING: SECURITY OF RS

WHERE AML+RECSYS STANDS IN MILSTONES OF RS DEVELOPMENT?



ADVERSARIAL LEARNING FOR RS



SECURITY OF RS

Attack Type	Years
Hand-engineered shilling attacks	**Early 2000-till now**
• Attack by leveraging interaction data • Attack by exploiting semantic data • Studying the impact of data characteristics on shilling attacks • Detection and defense of shilling attack	
Machine-learned Data Poisoning Optimization	**Recently emerging**
• Factorization-based models • Reinforcement Learning models • Other recommendation models • Defense	
Adversarial machine-learned attacks	**2016 till now (in RS field)**
• Adversarial perturbations on model parameters • Adversarial perturbation on content data • Defense and robustification	

SECURITY OF RS

Attack Type	Years
Hand-engineered shilling attacks	**Early 2000-till now**
• Attack by leveraging interaction data • Attack by exploiting semantic data • Studying the impact of data characteristics on shilling attacks • Detection and defense of shilling attack	
Machine-learned Data Poisoning Optimization	**Recently emerging**
• Factorization-based models • Reinforcement Learning models • Other recommendation models • Defense	
Adversarial machine-learned attacks	**2016 till now (in RS field)**
• Adversarial perturbations on model parameters • Adversarial perturbation on content data • Defense and robustification	

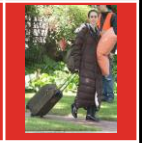
OUR MAIN FOCOUS

Hand-Crafted Shilling Attacks against RS

HAND-CRAFTED SHILLING ATTACKS

- **Problem:** Given a URM with 'n' users and 'm' items, the goal is to add $\lfloor \alpha.n \rfloor \, (\alpha << 1)$ fake (malicious) user profile.
  - **Constraint:** Each profile can have maximum 'C' ratings
- **Different attack types:** random, popular, bandwagon, love-hate:
  - Constructed based on the composition a of user profile

I_S			I_F			$I_\emptyset$			I_T
$i_s^{(1)}$	...	$i_s^{(\alpha)}$	$i_f^{(1)}$	...	$i_f^{(\phi)}$	$i_\emptyset^{(1)}$	...	$i_\emptyset^{(\chi)}$	i_t

					
	3			5	
		4			
			2		4
	1		5	5	
			4	4	
	2				5

fake user profiles

[Gunes, Ihsan, et al. "Shilling attacks against recommender systems: a comprehensive survey." Artificial Intelligence Review 42.4 (2014): 767-799.]

HAND-CRAFTED SHILLING ATTACKS

Recent advances have been dedicated on:

- Study of the **Impact of Dataset Characteristics on the Robustness of Popular Collaborative Recommenders**

  [Deldjoo, Di Noia, Di Sciascio, and Merra, How Dataset Characteristics Affect the Robustness of Collaborative Recommendation Models. In SIGIR 2020]

  - **Density** and **Space** **have** **Negative Impact**
    - **Increasing** the **density** of the dataset **REDUCES** the attacks' effectiveness.
  - **Space** has **Positive Impact**
    - Having **more users than items** may generate a lack of robustness
- Proposal of **Novel Defensive Strategies**
  - Using SVM method [Zhou, Wei, et al. "SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems." Neurocomputing 10]
  - Using user/item side information [Cai et al. "Detecting shilling attacks in recommender systems based on analysis of user rating behavior." Knowledge-Based Systems 2019]

Machine-Learned Data Poisoning Attacks

DATA POISONING OPTIMIZATION

- "Classic" Poisoning Attacks:
  - Label Modification
  - Poison Insertion
  - Data Modification
  - Boiling Frog

- Main limits:
  - Empirical techniques
  - Heuristics
  - No optimization procedure (neither Adversarial Learning) to maximize the attacker's utility

DATA POISONING OPTIMIZATION

Which kinds of attacker's utilities could be maximized?

- **Availability attack:** Attacker tries to maximize the estimated error in the target domain
- **Integrity attack:** Attacker's goal is to increase (or decrease) the popularity of a subset of items
- **Hybrid attack:** A mixture of availability attack and integrity attack

Huiyuan Chen and Jing Li. Data poisoning attacks on cross-domain recommendation. In Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundenst einer, David Carmel, Qi He, and Jeffrey Xu Yu, editors, Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019, pages 2177–2180. ACM, 2019

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

- Data Poisoning Optimization is DIFFERENT (or better it is another level of optimization) from the classic optimization for a recommendation utility

- What we need are:
  - the target recommendation model;
  - an attacker utility;
  - an optimization procedure for that utility.

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

- In principle, every model could be optimized for data poisoning attacks

- In the literature, we find (in descending popularity order):
  1. Factorization-based models
  2. Reinforcement Learning-based models
  3. Graph-based models
  4. K-NN models
  5. others

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

- In principle, every model could be optimized for data poisoning attacks

- In the literature, we find (in descending popularity order):
  1. **Factorization-based models**
  2. **Reinforcement Learning-based models**
  3. Graph-based models
  4. K-NN models
  5. others

POISONING FACTORIZATION-BASED MODELS

Attacker Utilities/strategies observed in the literature:

- Availability, Integrity, Hybrid attacks
- Hit ratio maximization/minimization to promote/demote items (Warning: Hit ratio is not directly optimizable) usually optimized by using the Wilcoxon-Mann-Whitney loss
- Exploitation of a subset of Influencial users to modify top-N recommendations

POISONING FACTORIZATION-BASED MODELS

Defensive strategies observed in the literature:

- SVM classifier to detect fake profiles trained on specific features/indicators: RDMA,WDMA,WDA,TMF,FMTD, and MeanVar
- Trim Learning to conduct a "clean" learning of the recommendation model
- Robustness analyzer to certify a factorization model as robust

POISONING REINFORCEMENT LEARNING-BASED MODELS

What we need to define:

- The recommendation model (or a "proxy", or.. nothing)
- The **attacker's knowledge** and capability(that are common even to the other attack families);
- The **state space** (usually the sequence of actions, or an embedding);
- The **action space** (usually, the items that can be chosen, or an embedding);
- The **reward utility**.

Adversarial Attacks and Defenses

CHALLENGES

1. Unlike **images** composed of **continous features**, the input to RS are discrete (rating, (u,i,j) in BPR)

   Gradient-based approaches to find perturbations
   → FGSM, BIM, PGD, C&W
   → are NOT suited

2. Adversarial examples on images aim to be **UNNOTICEABLE**.

How can we model the notion of 'unoticable' in (binary, discerete) RS?

ANSWERS TO CHALLANGES IN RS

- Most popular apporach so far has been to add **adversarial perturbations** are **added** to the **embeddings/model parameters** of Recommender Models to study:
  - **The STABILITY of TRAINING to MINIMAL ALTERATION of PARAMETERS**
- However, it remains as an open research question!

Politecnico di Bari

[He, Xiangnan, et al. "Adversarial personalized ranking for recommendation." The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2018.]

SECURITY OF RS

Attack Type	Years
Hand-engineered shilling attacks	**Early 2000-till now**
• Attack by leveraging interaction data • Attack by exploiting semantic data • Studying the impact of data characteristics on shilling attacks • Detection and defense of shilling attack	
Machine-learned Data Poisoning Optimization	**Recently emerging**
• Factorization-based models • Reinforcement Learning models • Other recommendation models • Defense	
Adversarial machine-learned attacks	**2016 till now (in RS field)**
• Adversarial perturbations on model parameters • Adversarial perturbation on content data • Defense and robustification	

Adversarial Perturbation on Model Parameters

ADVERSARIAL PERTUBRATION OF MODEL PARAMTERS

- **Adversarial Perturbation on Model Parameters**
  - **Reduce** the accuracy performance of the Recommender (**Attacks** point-of-view)
  - Study the stability of the training through the minimal-perturbation of learned parameters (**Defender** point-of-view)

[Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommenda-tion. In SIGIR.]

He et al. showed that by exposing the model parameters of BPR to

- adversarial perturbations: NDCG decreases -21.2%
- random perturbations: NDCG decreases -1.6%

ADVERSARIAL PERTUBRATION OF MODEL PARAMTERS

- **Adversarial Perturbation on Model Parameters**
  - **Reduce** the accuracy performance of the Recommender (**Attacks** point-of-view)
  - Study the stability of the training through the minimal-perturbation of learned parameters (**Defender** point-of-view)

[Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommenda-tion. In SIGIR.]

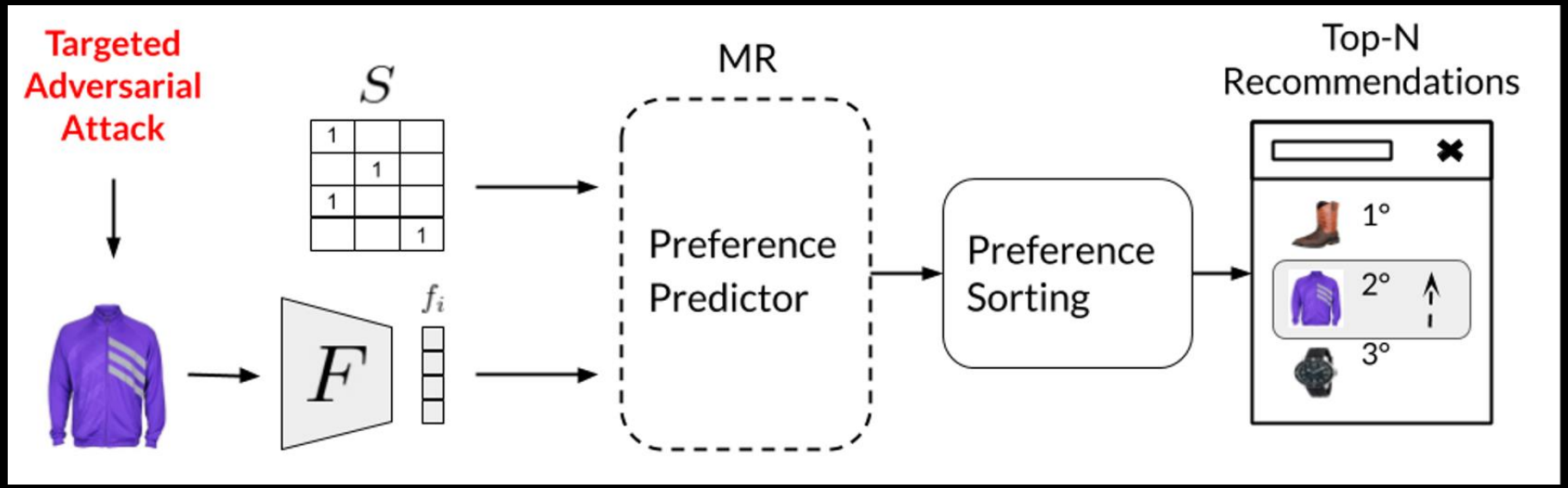
He et al. showed that by exposing the model parameters

- adversarial perturbations: NDCG decreases -21.2%
- random perturbations: NDCG decreases -1.6%

13 times difference!

Adversarial Perturbation on Content Data

TAAMR: TARGETED ADVERSARIAL ATTACK AGAINST MULTIMEDIA RECOMMENDER SYSTEMS



Simulation of Targeted Adversarial Attacks against Multimedia Recommender Systems can push low recommended product categories even **3 times more recommended** by **perturbing** product images in a **human-imperceptible way.**

Politecnico di Bari

[Di Noia, Tommaso, Daniele Malitesta, and Felice Antonio Merra. "TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems." the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-DSML'20). 2020.]

Defense against Adversarial Samples

DEFENSE AGAINST ADVERSARIAL SAMPLES

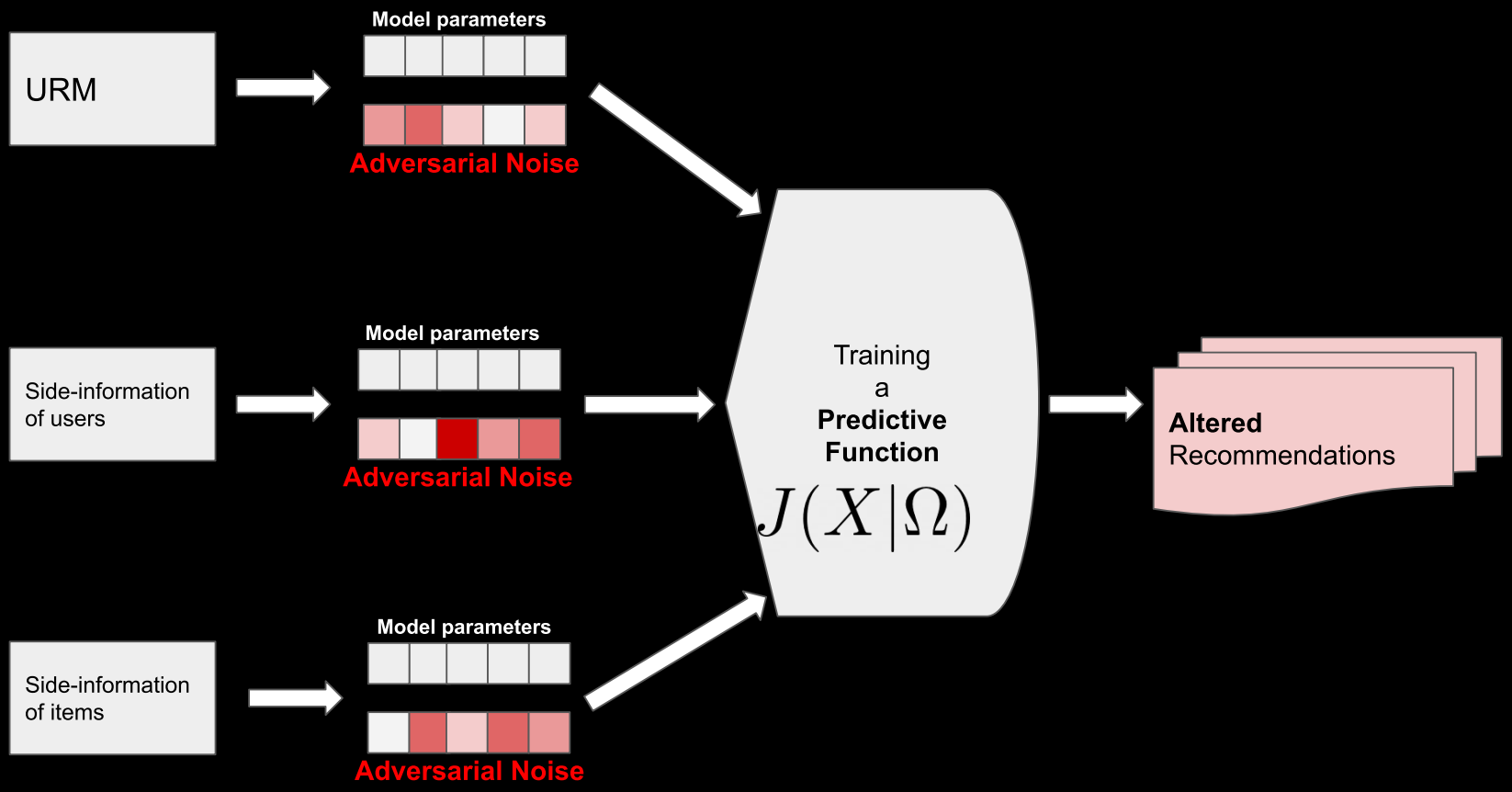
- **Goal**: Build ML models that can make robust prediction even in prescence of adversial examples.
- Main approaches:

Most Popular in RecSys

1. **Robust optimization**
   a) **Adversarial regularization** (aka adversarial training)
   b) Robust gradient descent
   c) Certified robutsness
2. **Defensive Distilliation**

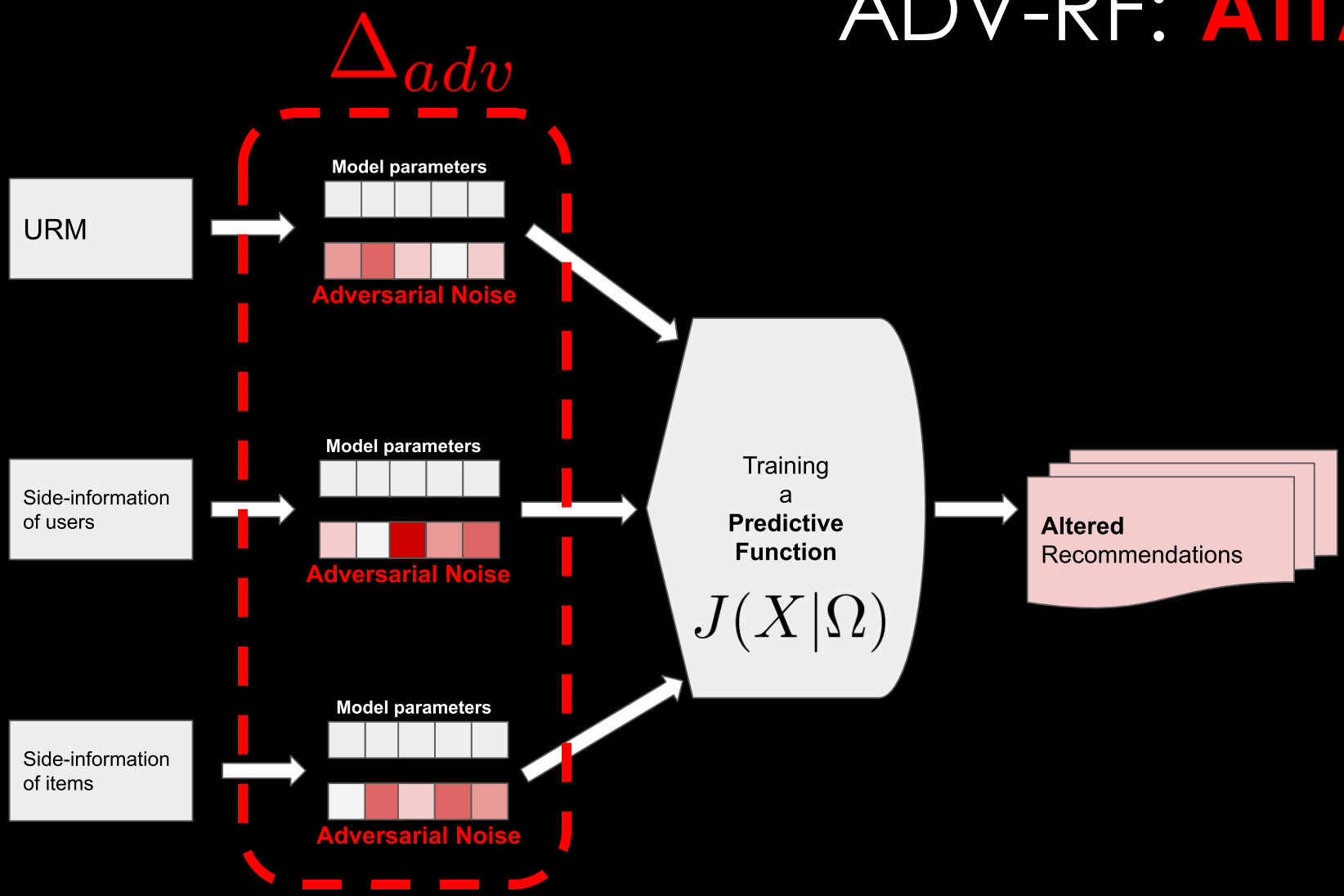
Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

OPTIMIZATION FRAMEWORK FOR ADVERSARIAL ROBUST RECOMMENDATION



Politecnico di Bari

ADV-RF: **ATTACKS**



Politecnico di Bari

ADV-RF: **ATTACKS**

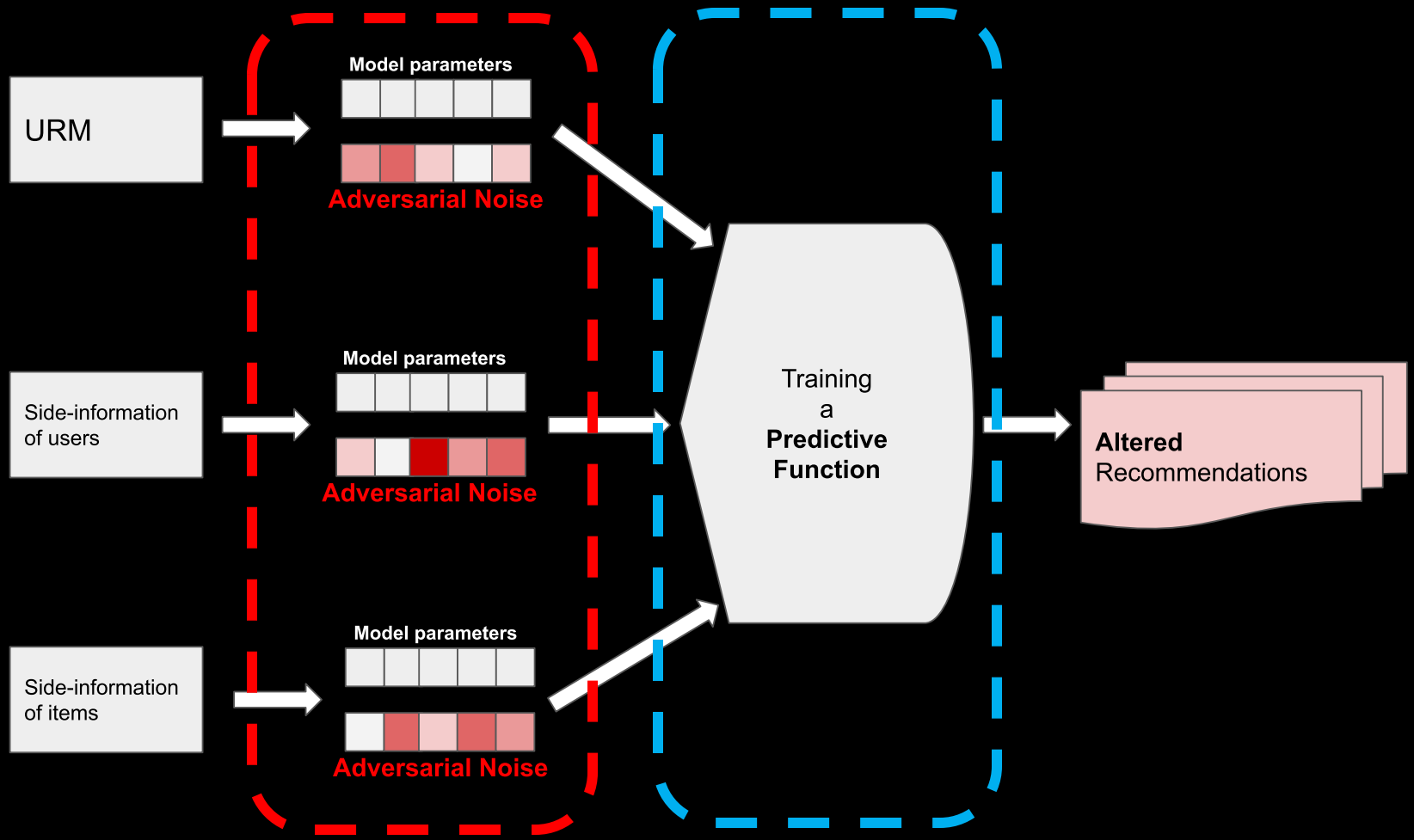
- **ATTACKER Goal?** **Maximize** the recommender model objective

$$\Delta_{adv} = \arg \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

- **Attack Method?** **FGSM**

$$\Delta_{adv} = \epsilon \frac{\Pi}{\|\Pi\|} \quad \text{where} \quad \Pi = \frac{\partial J(X|\Omega + \Delta)}{\partial \Delta}$$

ADV-RF: **DEFENSE**



Politecnico di Bari

ADV-RF: **DEFENSE**

- **Defender Goal? Minimize** the attack influence
- **Defence Method? Adversarial (re)training**

$$\arg\min_{\Omega} J(X|\Omega) + \lambda J\left(X|\Omega + \Delta_{adv}\right)$$

$$where \quad \Delta_{adv} = \arg\max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

MINIMAX GAME

The training process is a **MINIMAX GAME**

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

ADV-RF: **ALGORITHM**

Input: Training data $X, \epsilon, \lambda, \alpha$

Pretrain The Recommender Model

While ***stopping-criteria*** **do**:

Randomly select an x (or a mini-batch) from X

Construct adversarial perturbations

Update Model Parameters

$$\arg\min_{\Omega} \max_{\Delta, ||\Delta|| \leq \epsilon} \mathcal{L}(X|\Omega) + \lambda \mathcal{L}(X|\Omega + \Delta)$$

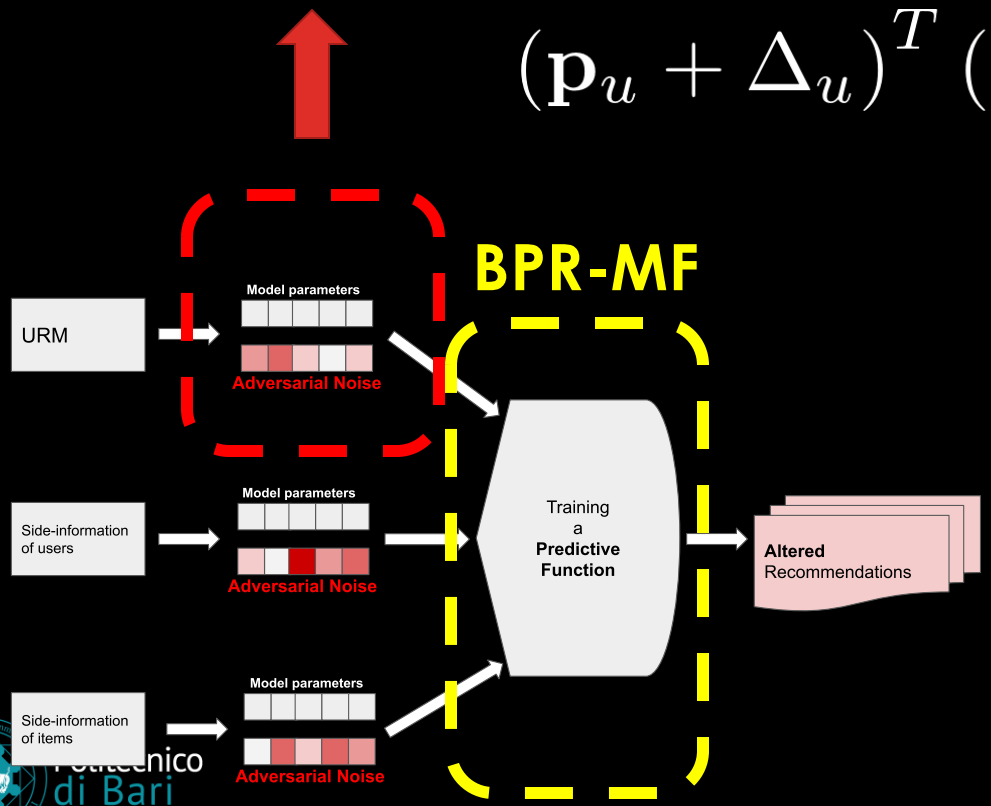
Politecnico di Bari

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Adversarial Perturbation on each **embedding** vector of user and item

$$(\mathbf{p}_u + \Delta_u)^T (\mathbf{q}_i + \Delta_i)$$



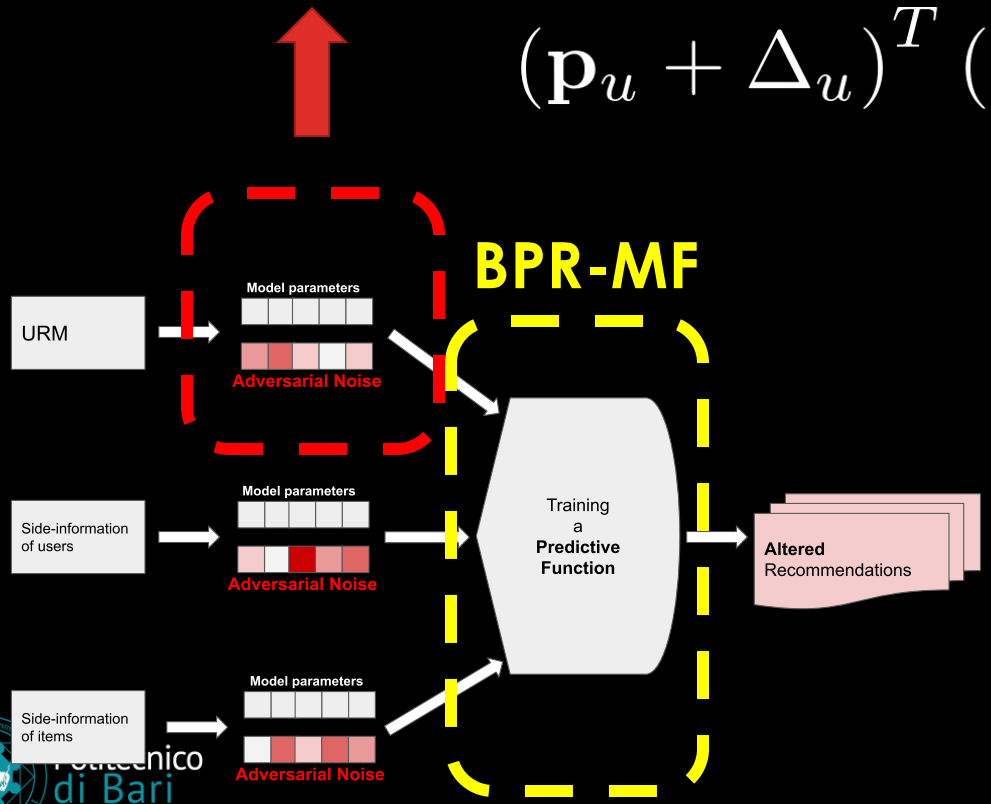
Politecnico di Bari

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Adversarial Perturbation on each **embedding** vector of user and item

$$(\mathbf{p}_u + \Delta_u)^T (\mathbf{q}_i + \Delta_i)$$



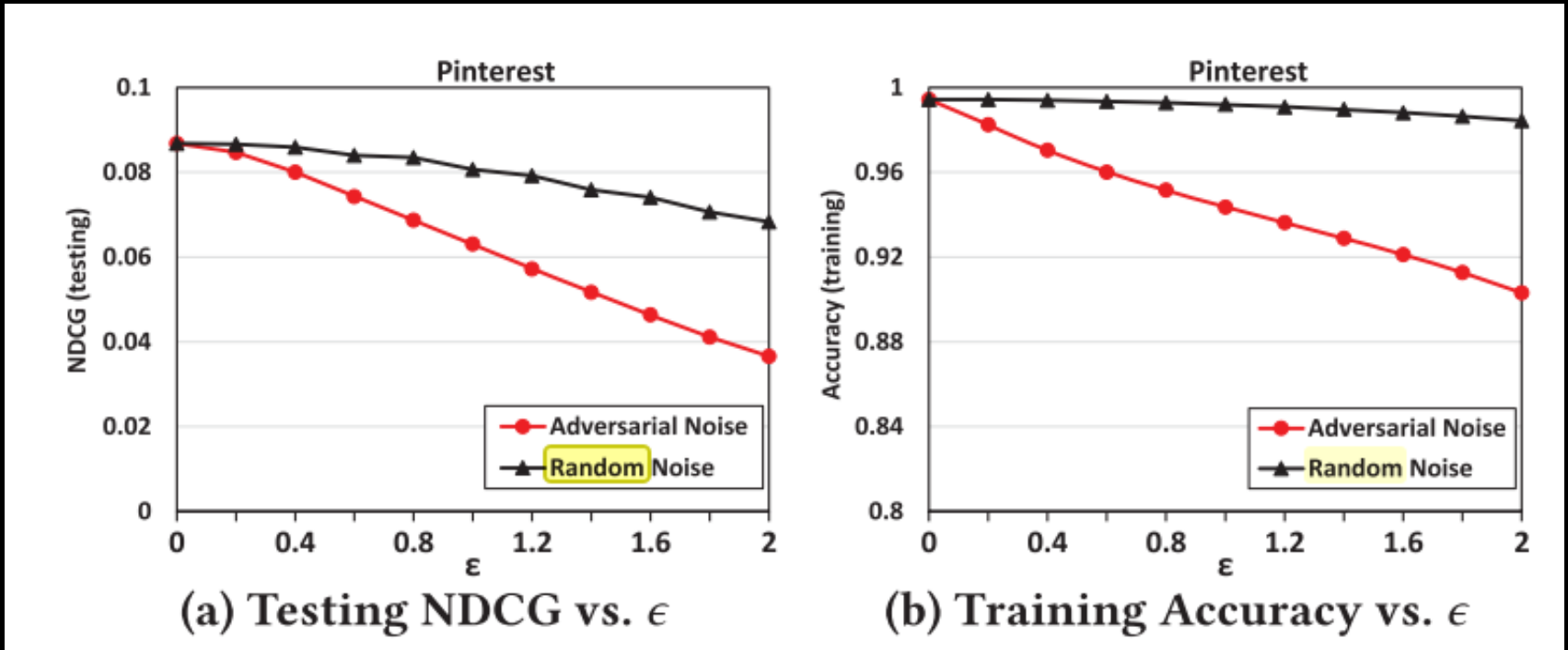
The impact of applying adversarial perturbation

reduction of NDCG@100			
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$
Dataset	BPR-MF	BPR-MF	BPR-MF
Yelp	-22.1%	-42.7%	-63.8%
Pinterest	-9.5%	-25.1%	-55.7%
Gowalla	-26.3%	-53.0%	-78.0%

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Are **Adversarial Perturbation** more **effective** than **random perturbation**?



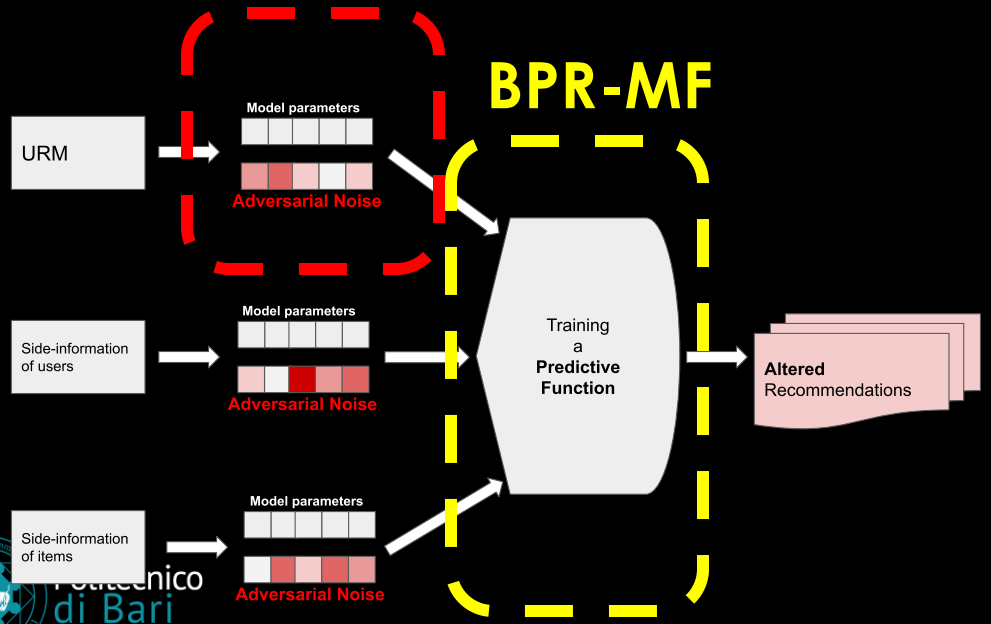
Politecnico di Bari

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Adversarial Perturbation on each **embedding** vector of user and item

$$(\mathbf{p}_u + \Delta_u)^T (\mathbf{q}_i + \Delta_i)$$



Apply Adversarial Training

Adversarial Regularization

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

where $J = J_{BPR}$

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Do **Adversarial (re)training** improve the **robustness**?

	NDCG@100					
	$\epsilon = 0.5$		$\epsilon = 1$		$\epsilon = 2$	
Dataset	BPR-MF	APR	BPR-MF	APR	BPR-MF	APR
Yelp	-22.1%	-4.7%	-42.7%	-12.5%	-63.8%	-31.0%
Pinterest	-9.5%	-2.6%	-25.1%	-7.2%	-55.7%	-23.4%
Gowalla	-26.3%	-2.9%	-53.0%	-13.2%	-78.0%	-29.2%

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Can **Adversarial (re)training** improve the **recommendation performance**?

	Yelp, HR		Yelp, NDCG		Pinterest, HR		Pinterest, NDCG		Gowalla, HR		Gowalla, NDCG		RI
	K=50	K=100	K=50	K=100	K=50	K=100	K=50	K=100	K=50	K=100	K=50	K=100	
ItemPop	0.0405	0.0742	0.0114	0.0169	0.0294	0.0485	0.0085	0.0116	0.1183	0.1560	0.0367	0.0428	+416%
MF-BPR	0.1053	0.1721	0.0312	0.0420	0.2226	0.3403	0.0696	0.0886	0.4061	0.5072	0.1714	0.1878	+11.2%
CDAE [35]	0.1041	0.1733	0.0293	0.0405	0.2254	0.3495	0.0672	0.0873	0.4435	0.5483	0.1837	0.2007	+9.5%
IRGAN [31]	0.1119	0.1765	**0.0361***	**0.0465***	0.2254	0.3363	0.0724	0.0904	0.4157	0.518	0.1853	0.2019	+5.9%
NeuMF [17]	0.1135	0.1817	0.0335	0.0445	0.2342	0.3526	0.0734	0.0925	0.4558	0.5642	0.1962	0.2138	+2.9%
AMF	**0.1176***	**0.1885***	0.0350	**0.0465***	**0.2375***	**0.3595***	**0.0741***	**0.0938***	**0.4693***	**0.5763***	**0.2039***	**0.2212***	-

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Can **Adversarial (re)training** improve the **recommendation performance**?

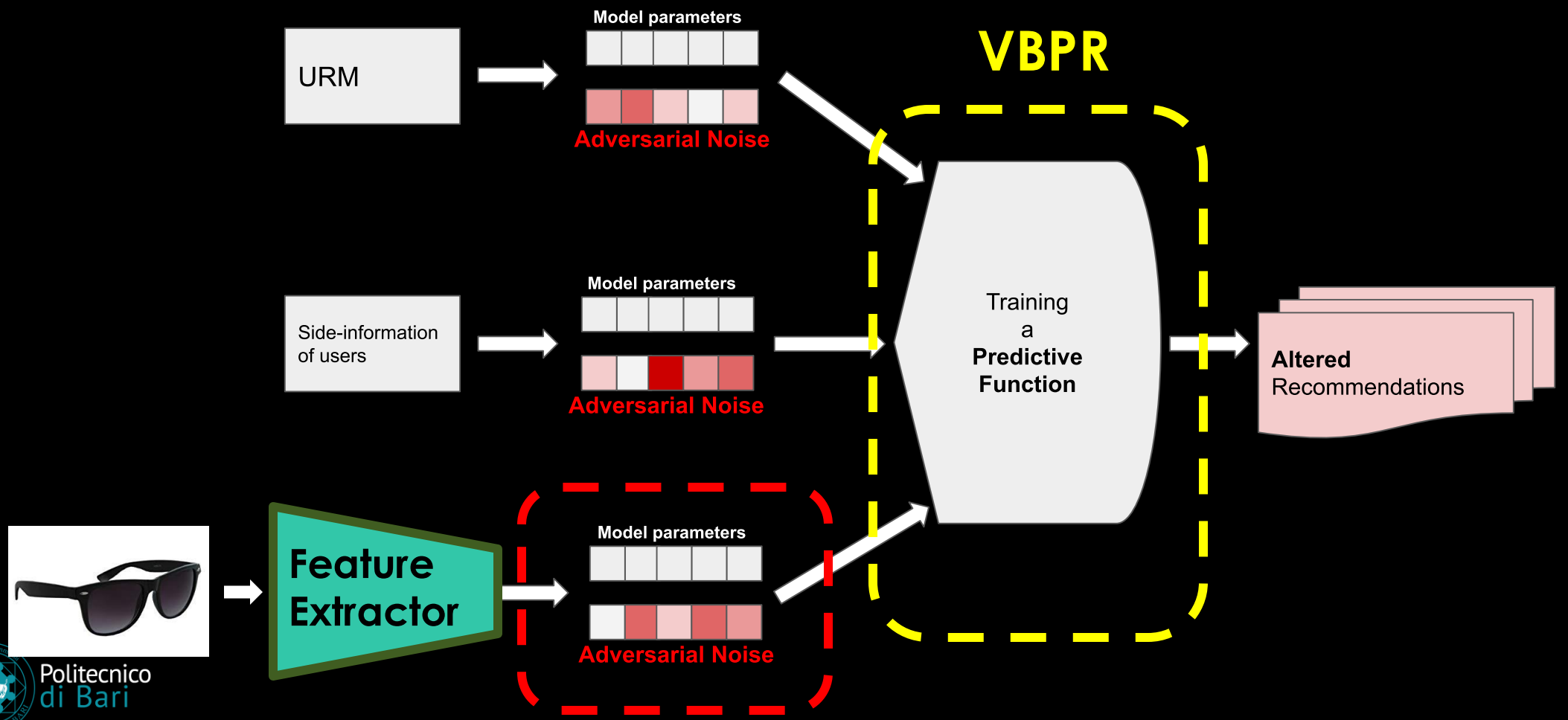
	RI
ItemPop	+416%
MF-BPR	+11.2%
CDAE [35]	+9.5%
IRGAN [31]	+5.9%
NeuMF [17]	+2.9%
AMF	-

Relative Improvement on HR@k and NDCG@k

Politecnico di Bari

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]



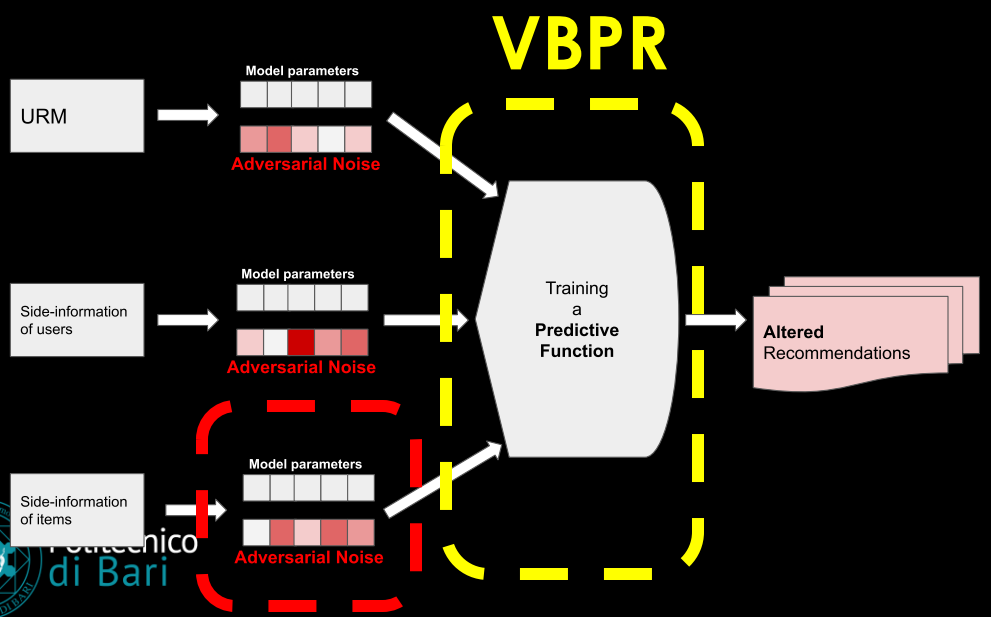
Politecnico di Bari

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Adversarial Perturbation on each **CONTENT** embedding.

$$\mathbf{p}_u^T(\mathbf{q}_i + \boldsymbol{E} \cdot (\boldsymbol{c}_i + \boldsymbol{\Delta}_i))$$

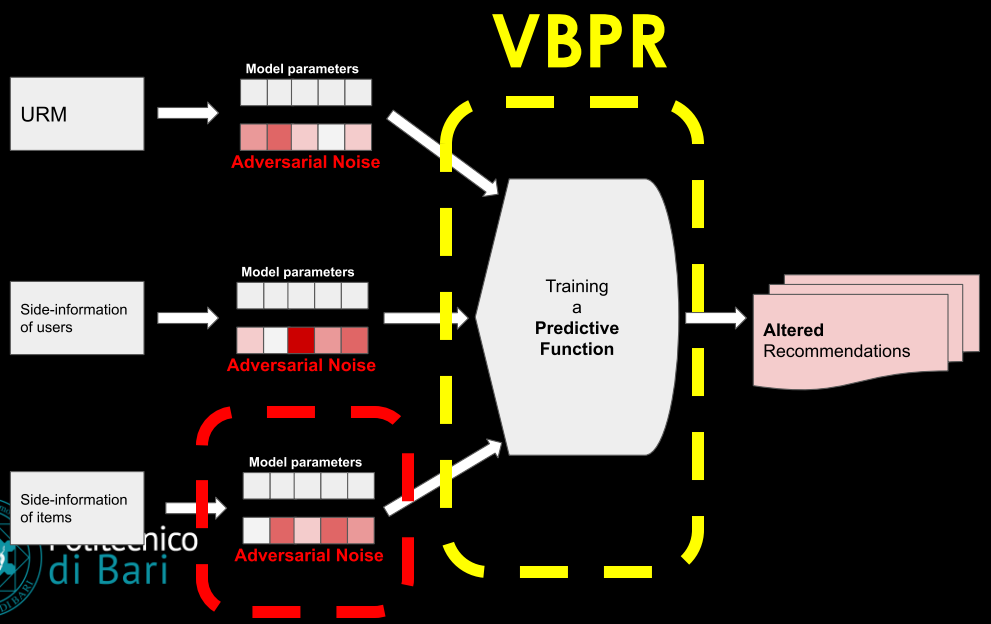


ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Adversarial Perturbation on each **CONTENT** embedding.

$$\mathbf{p}_u^T(\mathbf{q}_i + \boldsymbol{E} \cdot (\boldsymbol{c}_i + \boldsymbol{\Delta}_i))$$



The impact of applying adversarial perturbation

reduction of NDCG@10			
	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.2$
Dataset	VBPR	VBPR	VBPR
Amazon	-8.7%	-30.4%	-67.7%
Pinterest	-4.2%	-11.9%	-31.8%

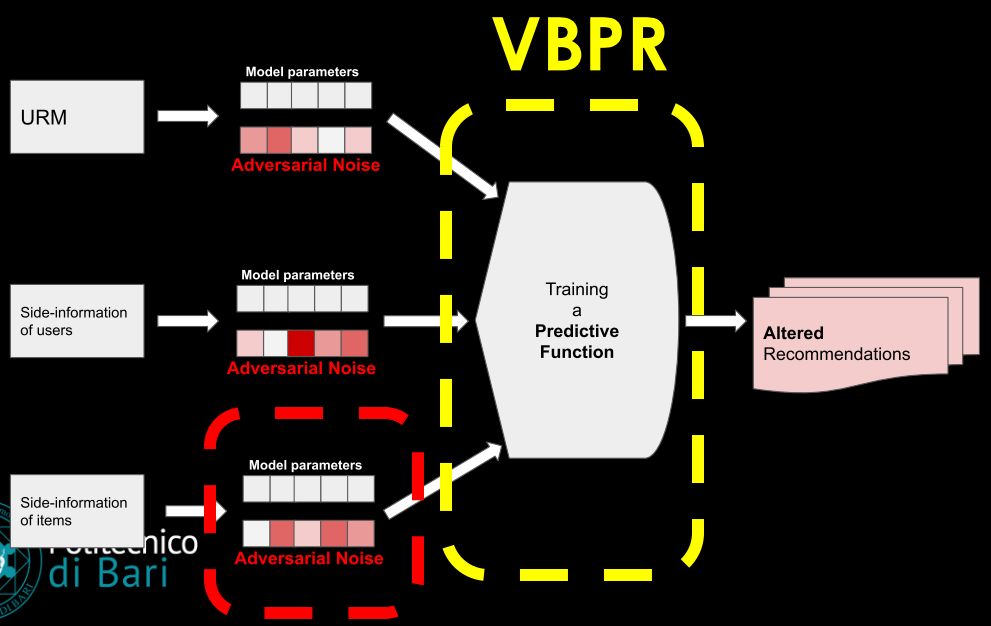
Politecnico di Bari

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Adversarial Perturbation on each **CONTENT** embedding.

$$\mathrm{p}_u^T(\mathrm{q}_i + \boldsymbol{E} \cdot (\boldsymbol{c}_i + \boldsymbol{\Delta}_i))$$



Apply Adversarial Training

Adversarial Regularization

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

where $J = J_{VBPR}$

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Do **Adversarial (re)training** improve the **robustness**?

	$\epsilon = 0.05$		$\epsilon = 0.1$		$\epsilon = 0.2$	
Dataset	**VBPR**	**AdvReg**	**VBPR**	**AdvReg**	**VBPR**	**AdvReg**
Amazon	-8.7%	-1.4%	-30.4%	-5.3%	-67.7%	-20.2%
Pinterest	-4.2%	-2.6%	-11.9%	-6.2%	-31.8%	-18.4%

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Can **Adversarial (re)training** improve the **recommendation performance**?

	RI
Pop	34.27%
MF-eALS	7.98%
MF-BPR	7.91%
DUIF	52.61%
VBPR	5.14%
AMR	-

Relative Improvement on HR@k and NDCG@k

Politecnico di Bari

OTHER APPLICATIONS

[Yuan, F. et al., 2019]

- attack/defense on **Deep Neural Model/ Auto-Encoder**

[Chen and Li, 2019]

- attack/defense on Factorization Machine

[Li R. et al., 2019]

- adv. regularization to improve **Sequential recommendations**

[Park et al., 2019]

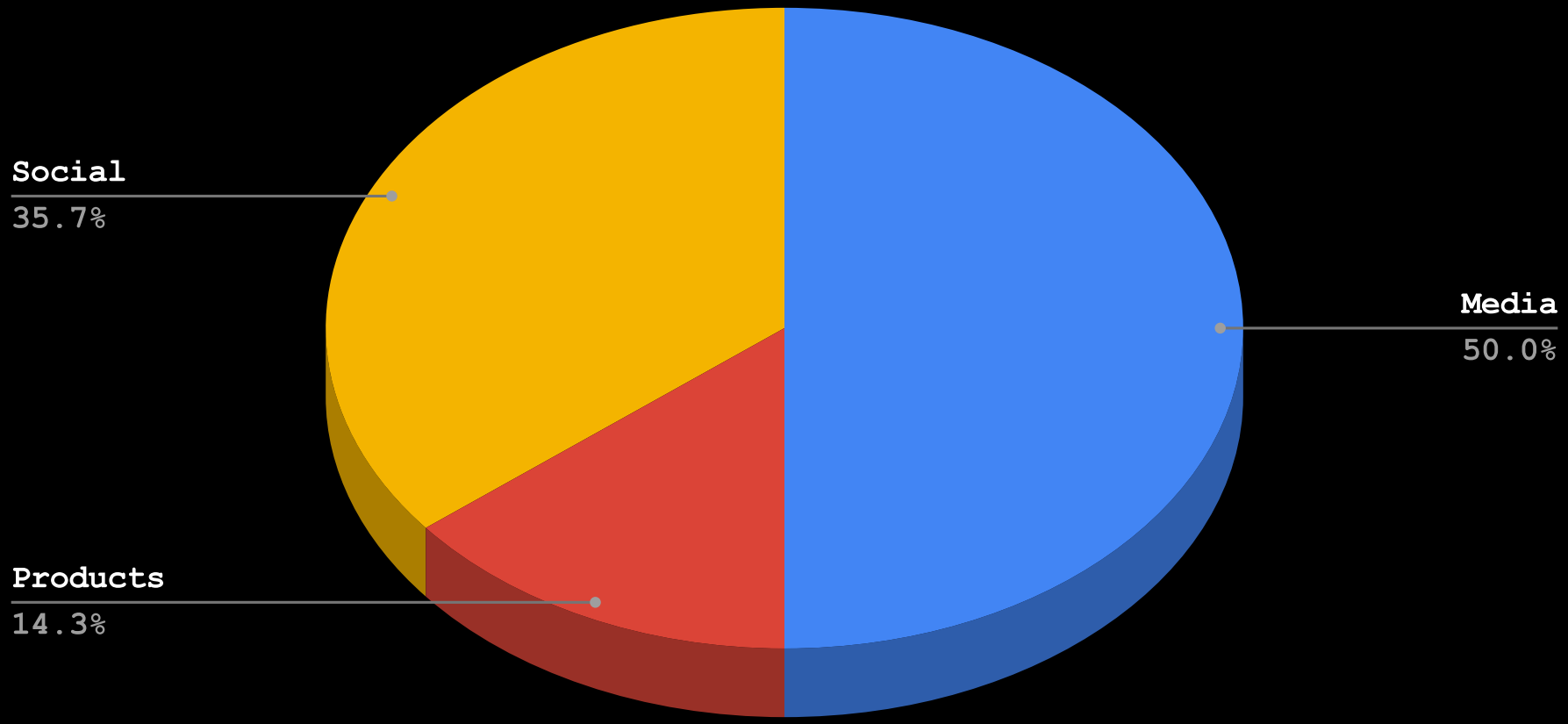
- perturb **user-continuous input**

[Du et al., 2019]

- attack with **C&W**, defense with **Defensive Distillation**

For further study check our Survey!

DOMAINS



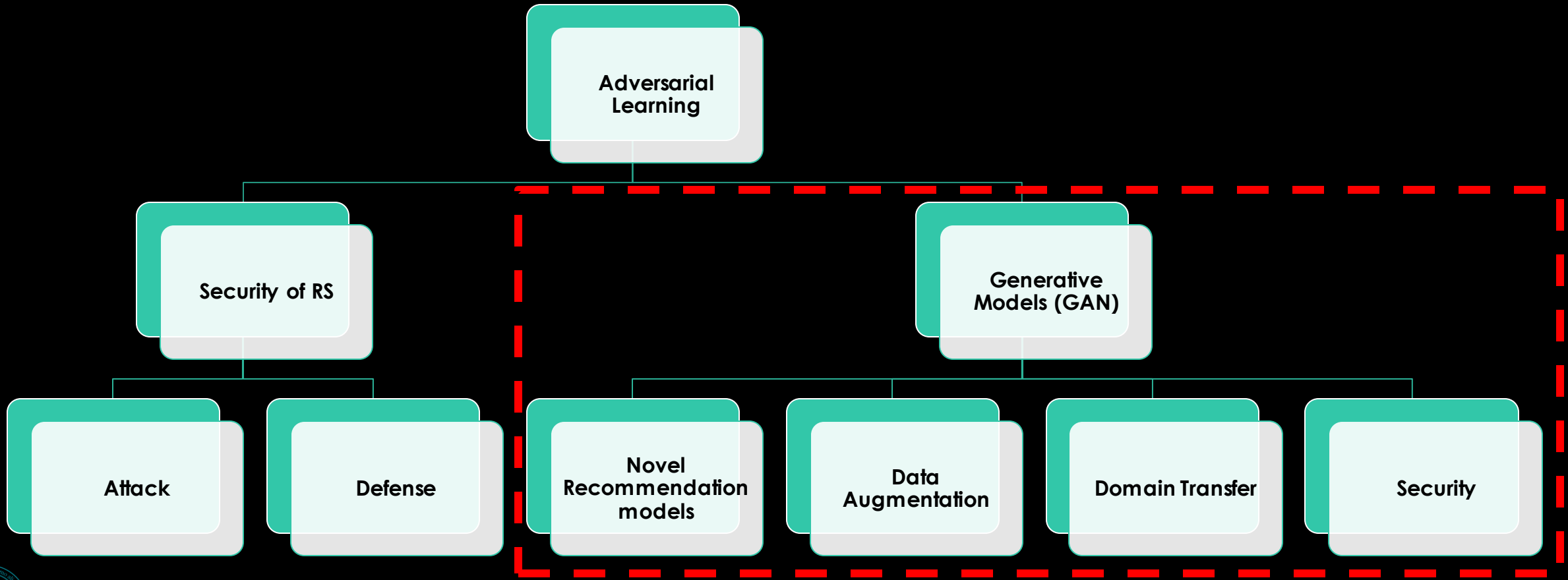
Politecnico di Bari

OPEN DIRECTIONS

- Other **attacks strategies**
  - Use state-of-the-art adv. Attack strategies
  - Implement perturbation direct on the input:
    - user-rating profile
    - Imitation of implicit feedback
    - images, audio, videos
- Other **defence approaches**
- Verify and Extend the **AVD-RF** on other recomenders
- Other **domains**

PART 3
ADVERSARIAL LEARNING FOR GAN-BASED RECOMMENDATION

ADVERSARIAL LEARNING FOR RS



Politecnico di Bari

3.1 GAN-BASED RECOMMENDATION FRAMEWORK (**GAN-RF**)

PIONEERING WORKS

- **IRGAN** [Wang J. et al., *SIGIR'17*]
  - Solution for CF based on **MF**
  - Combine **generative** and **discriminative** **IR** schools of thinking under a GAN-framework
- **GraphGAN** [Wang H. et al., *AAAI'18*]
  - Solution for CF based on **graph-representation learning**
  - Combine **generative** and **discriminative** **graph-representation learning** models under a GAN-framework

IRGAN: A MINIMAX GAME FOR UNIFYING GENERATIVE AND DISCRIMINATIVE INFORMATION RETRIEVAL MODELS

[WANG J. ET AL., SIGIR'17]

Generative model

- **Assumption:** Exists a stochastic generative process between i and u

$$u \longrightarrow i$$

- **Model function:** $p_\theta(i|u,r)$ tries to generate **relevant items**, from the candidate pool for the given **user**

IRGAN: A MINIMAX GAME FOR UNIFYING GENERATIVE AND DISCRIMINATIVE INFORMATION RETRIEVAL MODELS

[WANG J. ET AL., SIGIR'17]

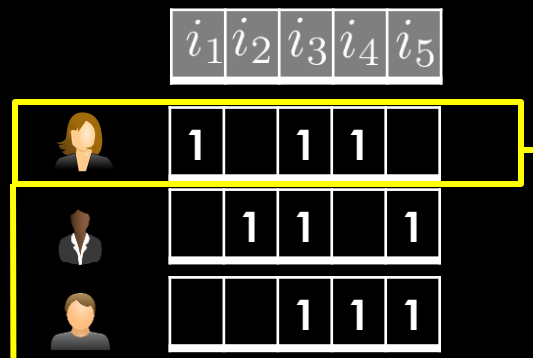
Discriminative model

- **Assumption:** The rating is predicted as a label given i and u

$$u + i \rightarrow r$$

- **Model function:** $p_\phi(u, i)$ tries to discriminate well-matched user-item pairs from ill-matched ones

GAN-RF



u

Real item

i_1

G is user-conditioned

u

G

$p_\theta(i|u,r)$

Generated item

i_7

D

Distinguish the true and generated relevant item

Politecnico di Bari

GAN-RF: MODEL FORMULATION

- We model GAN-RF with a **conditional GAN** (CGAN)
- G is user-conditioned (e.g., the user-id) to generate **user-personalized recommendations**

GAN-RF loss function:

the discriminative retrieval model

$$J(\theta, \phi) = \mathbb{E}_{i \sim p_{\text{true}}(i|u,r)}[\log D_\phi(i|u)] + \mathbb{E}_{\hat{i} \sim p_\theta(\hat{i}|u,r)}[\log(1 - D_\phi(\hat{i}|u))]$$

the generative retrieval model

Politecnico di Bari

GAN-RF: MODEL FORMULATION

Optimising **Discriminative Retrieval**

$$\arg\max_{\phi} J(\theta, \phi)$$

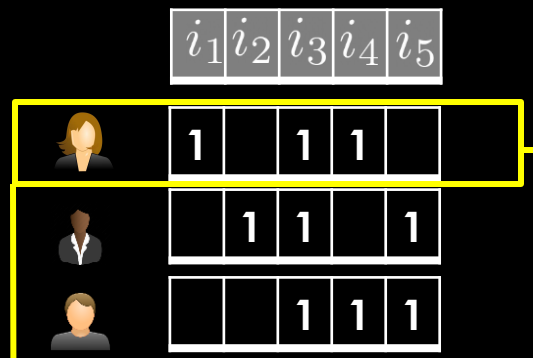
Optimising **Generative Retrieval**

$$\arg\min_{\theta} J(\theta, \phi)$$

The **MINIMAX GAME** to train the **generator** and the **discriminator** is formally defined as:

$$\arg\min_{\theta}\max_{\phi} J(\theta, \phi)$$

GAN-RF



u

Real item

i_1

$\arg\max_{\phi} J(\theta, \phi)$

D

Distinguish the true and generated relevant item

G is user-conditioned

u

G

$p_\theta(i|u, r)$

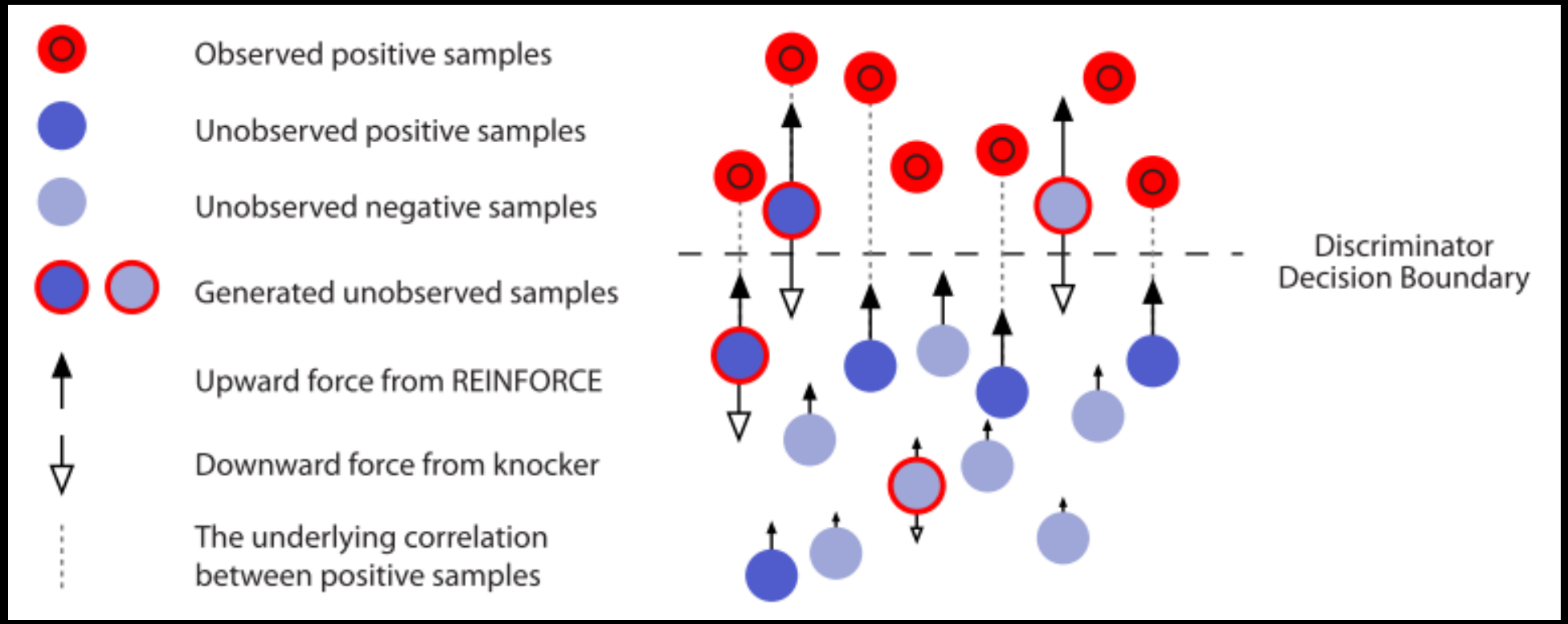
Generated item

i_7

$\arg\min_{\theta} J(\theta, \phi)$

Politecnico di Bari

GAN-RF


Observed positive samples
Unobserved positive samples
Unobserved negative samples
Generated unobserved samples
Upward force from REINFORCE
Downward force from knocker
The underlying correlation between positive samples
Discriminator Decision Boundary

ADV-RF: **ALGORITHM**

Input: Training data **X**
Initialize G and D
Pretrain G and D
While *stopping-criteria* do:

MINIMAX

For g-steps do:
Generate K relevant items for each
Update G parameters
end for
For d-steps do:
Combine Real relevant items with Generated Relevant items.
Update D parameters
end for

WE NEED A DIFFERENTIABLE ITEM SAMPLING PROCEDURE

Politecnico di Bari

DIFFERENTIABLE SAMPLING STRATEGY

The generation of recommendation lists is a **discrete item sampling** operation.

The gradients that are derived from $J(\theta, \phi)$ **cannot** be directly used to optimize the **generator** via **gradient descent**

Two main approaches:

1. The **policy gradient based reinforcement learning algorithm (REINFORCE)**

   first proposed by [Wang J. et al., *SIGIR2017*]

2. The **Gumbel-Softmax differentiable sampling procedure**

   first proposed by [Yangdong Ye et al., *EXPERT SYST APPL 2019*]

REINFORCE

The **gradient** of the generator is derived as follows:

$$
\begin{aligned}
\nabla_\theta J &= \nabla_\theta \mathbb{E}_{i \sim p_{\text{ true }}(i|u,r)}[\log D_\phi(i|u)] + \mathbb{E}_{\hat{i} \sim p_\theta(\hat{i}|u,r)}[\log(1 - D_\phi(\hat{i}|u)))] \\
&= \nabla_\theta \mathbb{E}_{\hat{i} \sim p_\theta(\hat{i}|u,r)}[\log(1 - D_\phi(\hat{i}|u)))] \\
&= \sum_{m=1}^{M} \nabla_\theta p_\theta(\hat{i}_m|u,r) \log(1 - D_\phi(\hat{i}_m|u)) \\
&= \sum_{m=1}^{M} p_\theta(\hat{i}_m|u,r) \nabla_\theta \log p_\theta(\hat{i}_m|u,r) \log(1 - D_\phi(\hat{i}_m|u))) \\
&= \mathbb{E}_{\hat{i} \sim p_\theta(\hat{i}|u,r)}[\nabla_\theta \log p_\theta(\hat{i}|u,r) \log(1 - D_\phi(\hat{i}|u)))] \\
&\simeq \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p_\theta(\hat{i}_k|u,r) \log(1 - D_\phi(\hat{i}_k|u)))
\end{aligned}
$$

reward for the generative function

$\hat{i}_k$ is the ***k*-th relevant item** sampled from the current version of the generative function

GUMBEL-SOFTMAX

REINFORCE suffers of **instability** in gradient estimation with large set of items

Gumbel-Softmax builds a **differentiable bridge** between **G** and **D**

We obtain an **approximate one-hot representation** of the sampled item:

$$v_k = \frac{\exp\left(\left(\log p_\theta\left(i_k|u,r\right) + g_k\right)/\tau\right)}{\sum_{j=1}^{K}\exp\left(\left(\log p_\theta\left(i_j|u,r\right) + g_j\right)/\tau\right)} \qquad k = 1,\ldots,K$$

τ Temperature parameter

where g_j are i.i.d **sampled noise** from

$$Gumbel(0,1) = -\log(-\log(Uniform(0,1)))$$

Politecnico di Bari

3.2 APPLICATIONS

CATEGORIZATION

1. **Collaborative** Recommendation
    1. Pure Collaborative Filtering
    2. Graph-based Recommendation
    3. Hybrid
2. **Contextual** Recommendation
    1. Temporal-aware
    2. Geographical
3. **Cross-domain** Recommendation
4. **Complementary** Recommendation
5. Other applications

COLLABORATIVE RECOMMENDATION - **PURE**:

CFGAN

[DONG-KYU CHAE, CIKM '18]

PROBLEM

1. Generator's **performance degredation** as training progresses
2. Difficulties in capturing **users's preference** on **0/1-sparse-vector**

PROPOSAL

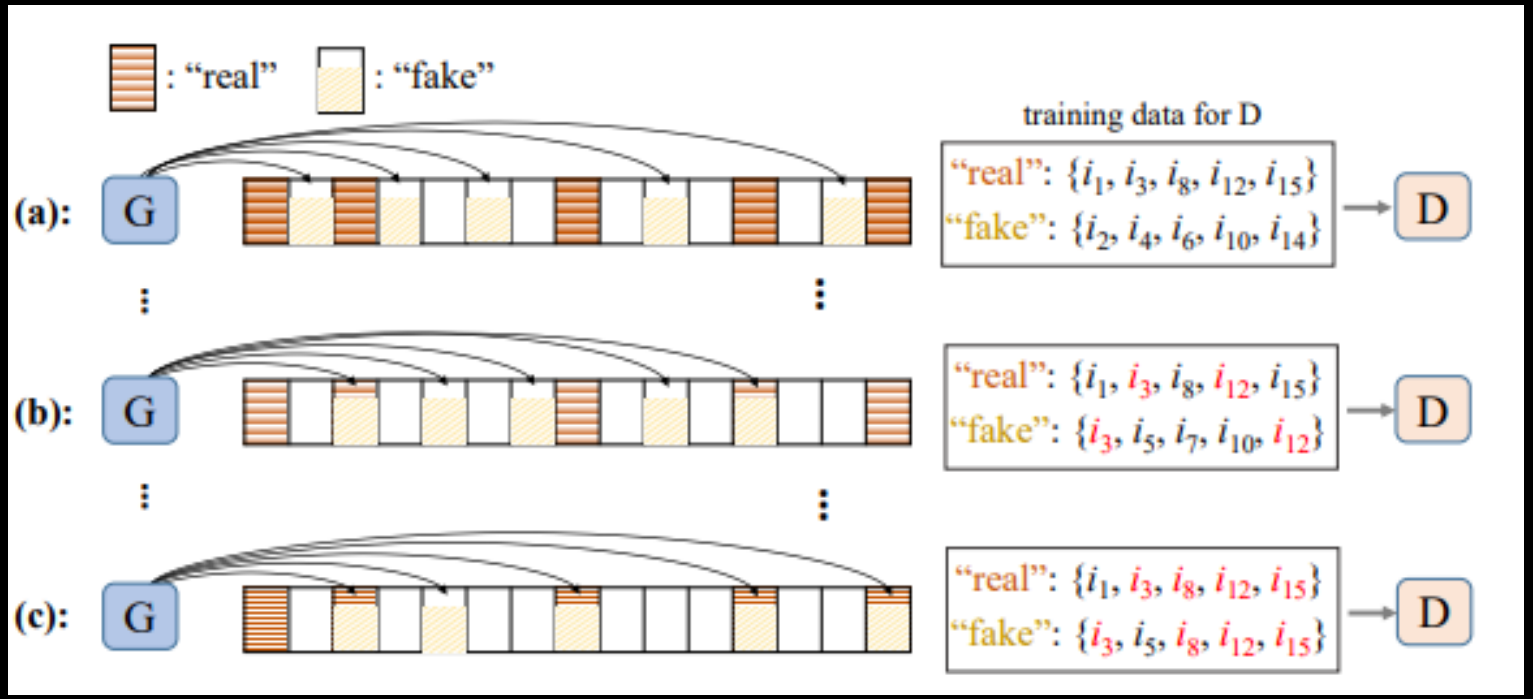
1. **Vector-wise adversarial training** where G generates ***real-valued vectors***
2. Novel **negative-sampling methods to capture the** user's relative preferences

COLLABORATIVE RECOMMENDATION - **PURE:**

CFGAN

[DONG-KYU CHAE, CIKM '18]

Generator's performance degredation because it samples 'real' items with **contradicting labels**

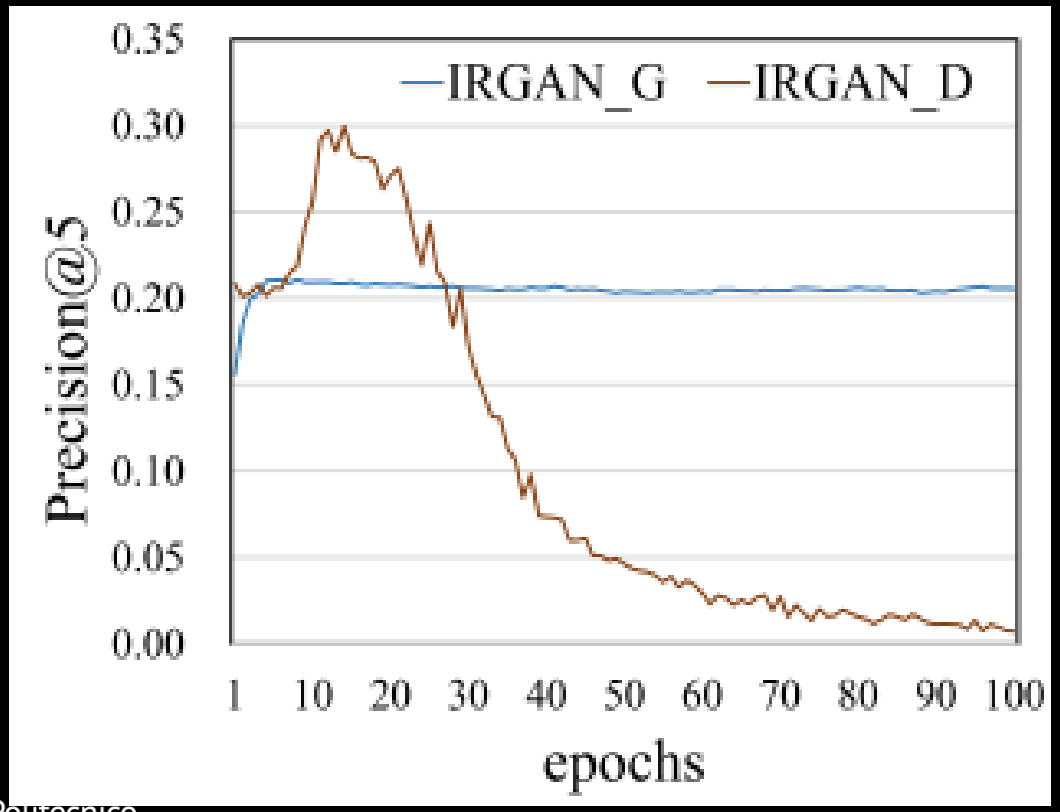


COLLABORATIVE RECOMMENDATION - **PURE:**

CFGAN

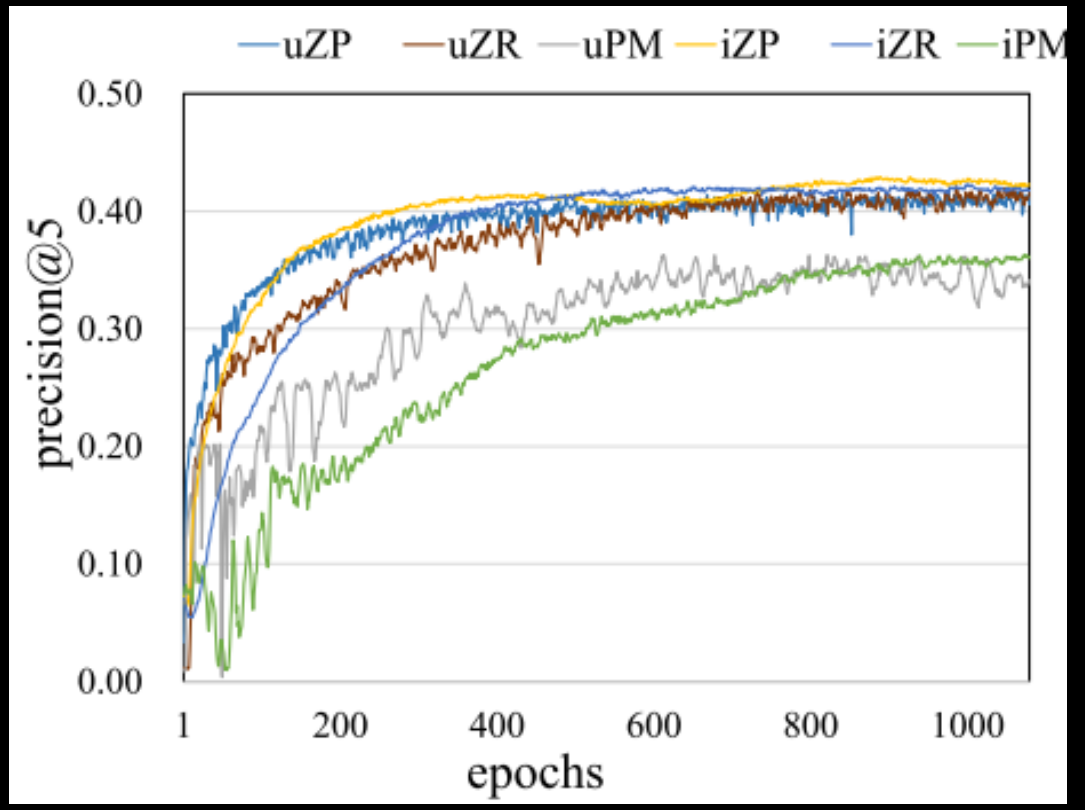
[DONG-KYU CHAE, CIKM '18]

Single-item IRGAN implementation



Results on Movielens 100K

Vector-wise CF-GAN implementation



Results on Movielens 100K

Politecnico di Bari

COLLABORATIVE RECOMMENDATION - **PURE:**

CFGAN

[DONG-KYU CHAE, CIKM '18]

datasets	Movielens 100K								Movielens 1M							
metrics	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20
ItemPop	.181	.138	.102	.251	.163	.195	.254	.292	.157	.121	.076	.197	.154	.181	.252	.297
BPR	.348	.236	.116	.287	.370	.380	.556	.574	.341	.252	.077	.208	.349	.362	.537	.556
FISM	.426	.285	.140	.353	.462	.429	.674	.685	.420	.302	.107	.270	.443	.399	.637	.651
CDAE	.433	.287	.144	.353	.465	.425	.664	.674	.419	.307	.108	.272	.439	.401	.629	.644
GraphGAN	.212	.151	.102	.260	.183	.249	.282	.312	.178	.194	.070	.179	.205	.184	.281	.316
IRGAN	.312	.221	.107	.275	.342	.368	.536	.523	.263	.214	.072	.166	.264	.246	.301	.338
Ours	**.444**	**.294**	**.152**	**.360**	**.476**	**.433**	**.681**	**.693**	**.432**	**.309**	**.108**	**.272**	**.455**	**.406**	**.647**	**.660**

G@N = normalized Discounted Cumulative Gain

M@N = Mean Reciprocal Rank

+ 2.8% against FISM [Kabbur et al., KDD'13]

COLLABORATIVE RECOMMENDATION - GRAPH:

GRAPHGAN

[WANG H. ET AL., AAAI'18]

PROBLEM

1. Unify Generative-Discriminative models in graph representation learning
2. Computationally inefficiency for **G** (softmax) gradient update

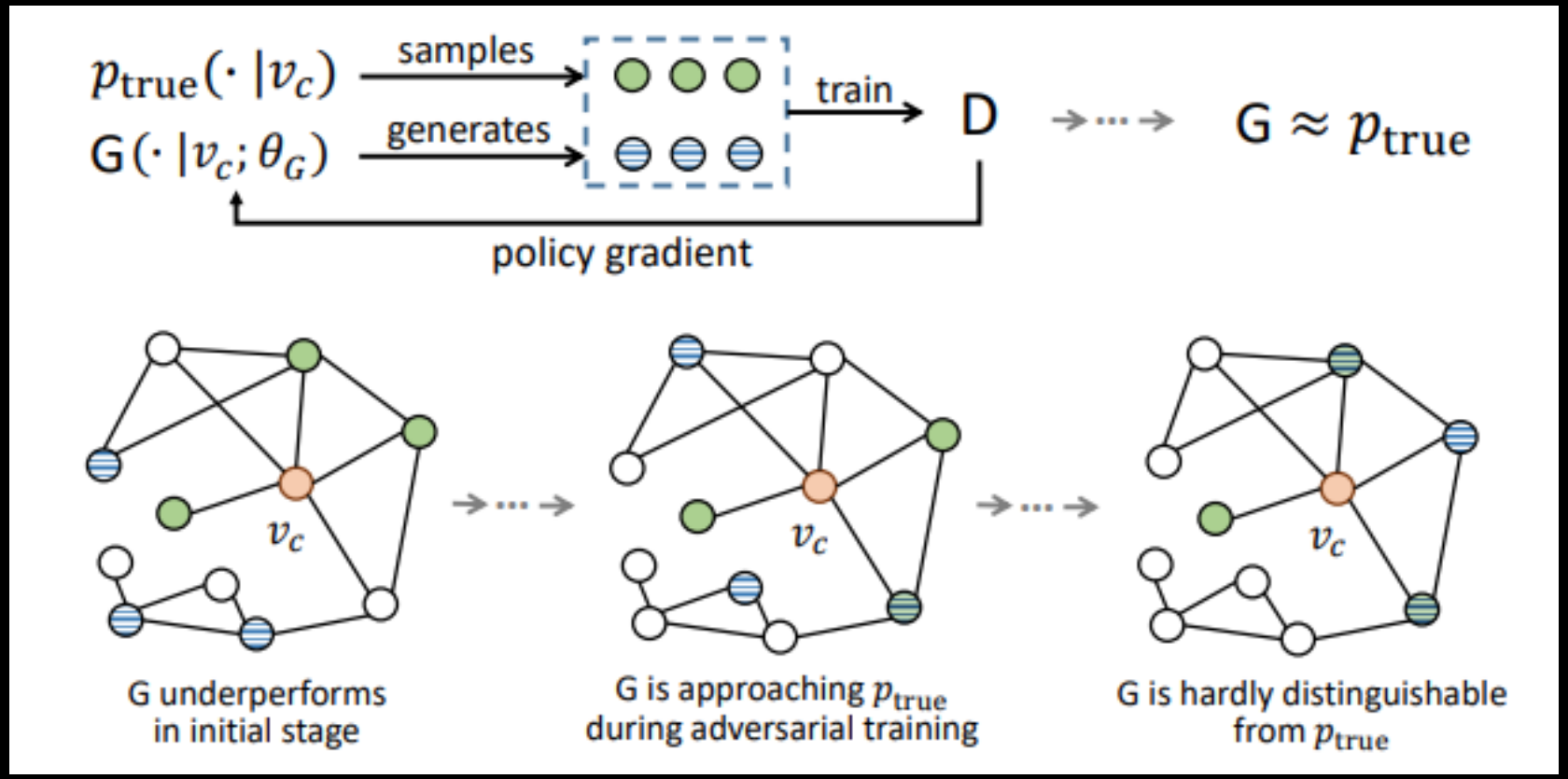
PROPOSAL

1. GAN-based approach (can be used as recommender model is data are represented as **bipartite-graph** and the recommendation problem as a **link prediction task**)
2. **Graph-softmax** to compute connectivity distribution in an **efficient** and **graph-structural-aware**

COLLABORATIVE RECOMMENDATION - GRAPH:

GRAPHGAN

[WANG H. ET AL., AAAI'18]



Discrete Item Sampling?
REINFORCE

Code

Politecnico di Bari

COLLABORATIVE RECOMMENDATION - HYBRID:

AUGCF

[WANG Q. ET AL., KDD'19]

PROBLEM

1. Data sparsity problem (**cold-users**)
2. Current **hybrid CF** models:
   - apply the expensive process of exploiting the side information to all users (**unnecessary computational resources**)
   - are **not general** (complex to adapt at varying of dataseta/domaina)

PROPOSAL

1. GAN-based framework to train a **generator** to be used as a **data augmenter to generate "real" interaction data for cold-users exploting side information**
2. **End-to-end** training procedure [data augmenter + CF model]

COLLABORATIVE RECOMMENDATION - **HYBRID:**

AUGCF

[WANG Q. ET AL., KDD'19]

Two-phase Procedure for the End-to-End Model

PHASE 1: BUILD the Data Augmenter

Minimax game between **G** and **D.**

The **Generator** aims to create a fake-interacted item by exploting the **side information**

PHASE 2: CF Recommendation

G is fixed and acts as a **data augmentator** with data **indistinguishable to D**

D learns to generate users' **relevant items based pm real or sampled items**

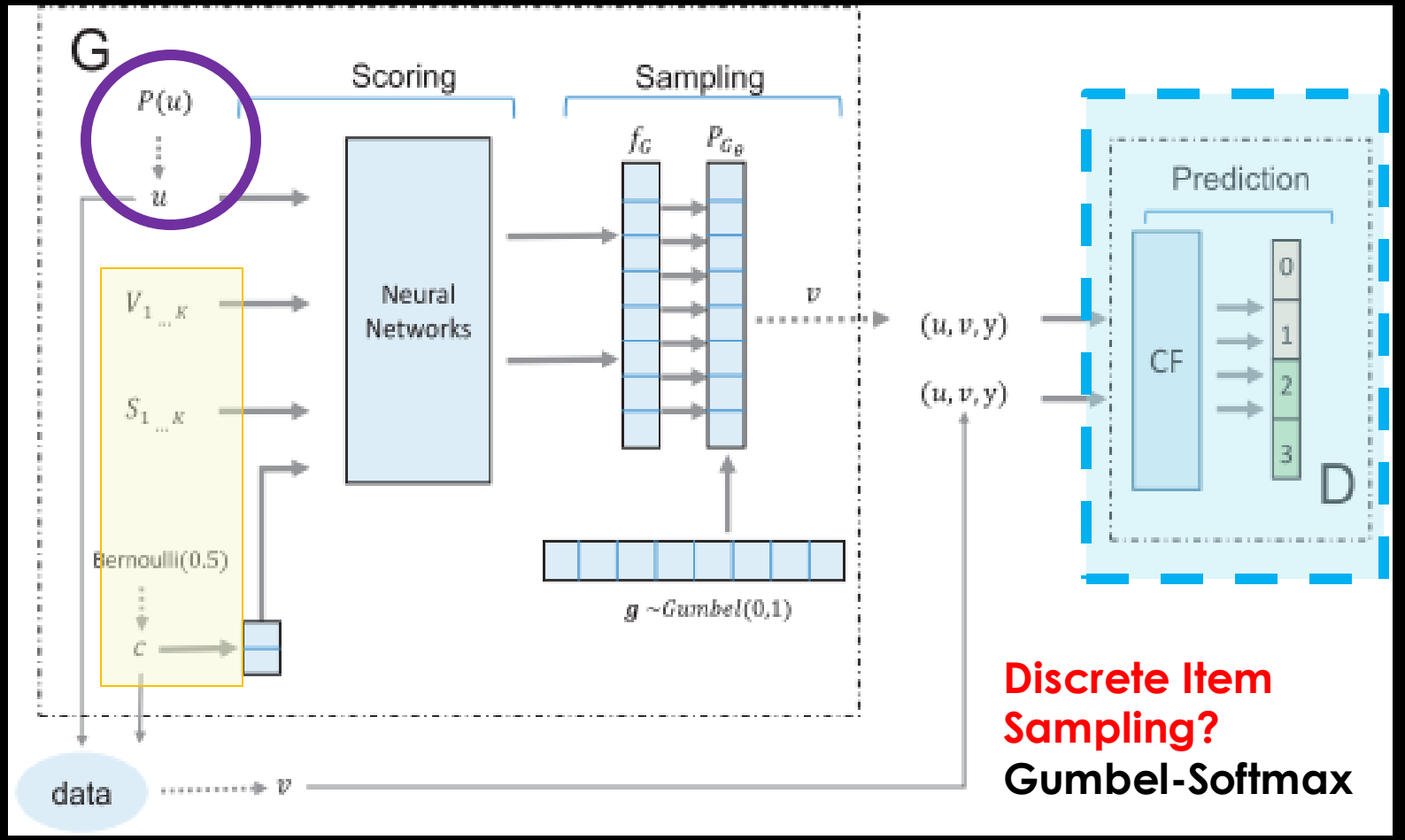
Acts as the RECOMMENDER

COLLABORATIVE RECOMMENDATION - HYBRID:

AUGCF

[WANG Q. ET AL., KDD'19]

less active users have higher probabilities to be sampled



COLLABORATIVE RECOMMENDATION - HYBRID: AUGCF

[WANG Q. ET AL., KDD'19]

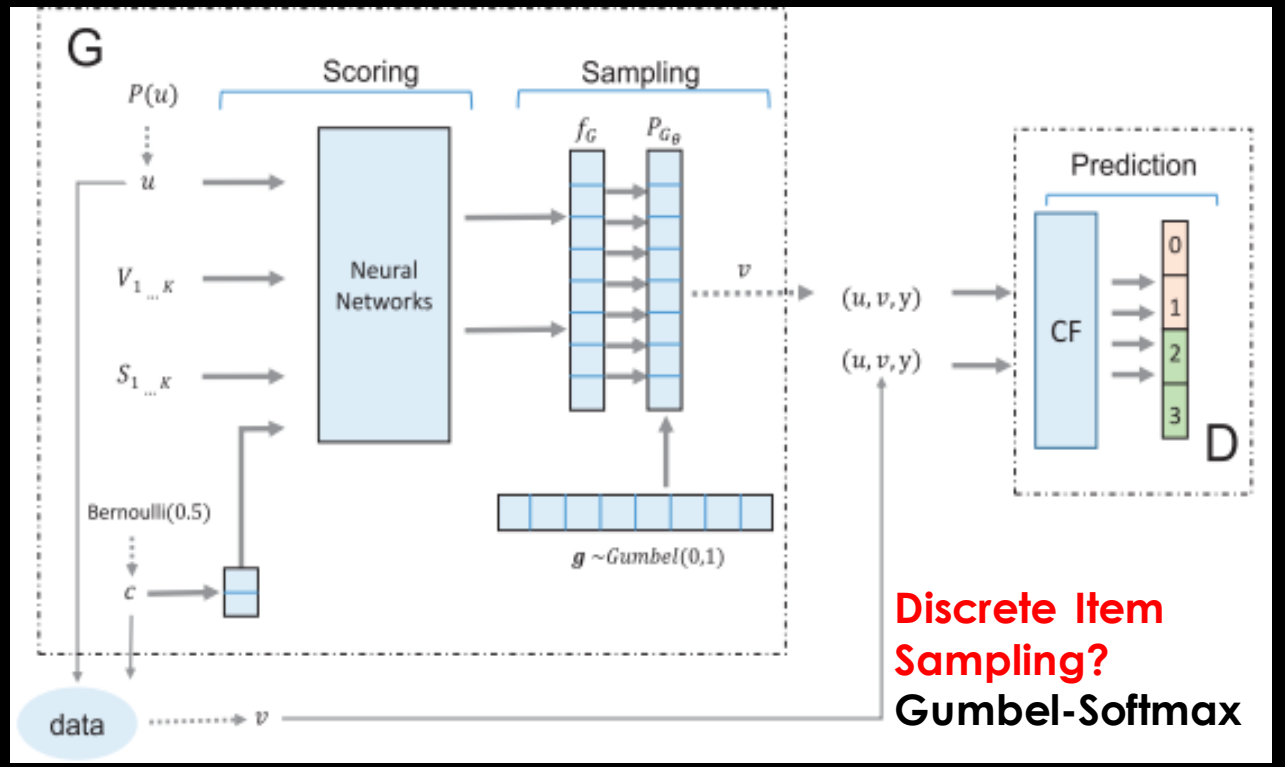
The authors tested the generalizability with 2 hybrid-models:

1. Content-Based AugCF

- Items reviews to model users' behaviors and items properties
- G is implemented with a DeepCoNN (Content-RS)

2. Sparse Feature-Based AugCF

- Sparse features like tags, timestamps
- G is implemented with **Wide & Deep learning framework**



Politecnico di Bari

CONTEXTUAL RECOMMENDATION - **TEMPORAL:**
PLASTIC
[MIN YANG ET AL., IJCAI'19]

PROBLEM

Leverage Long And Short- Term Information in top-*k* recommendation scenario

PROPOSAL

Use adversarial learning to train a generator G to generate **highly rewarded** recommendation list

CONTEXTUAL RECOMMENDATION - **TEMPORAL:**

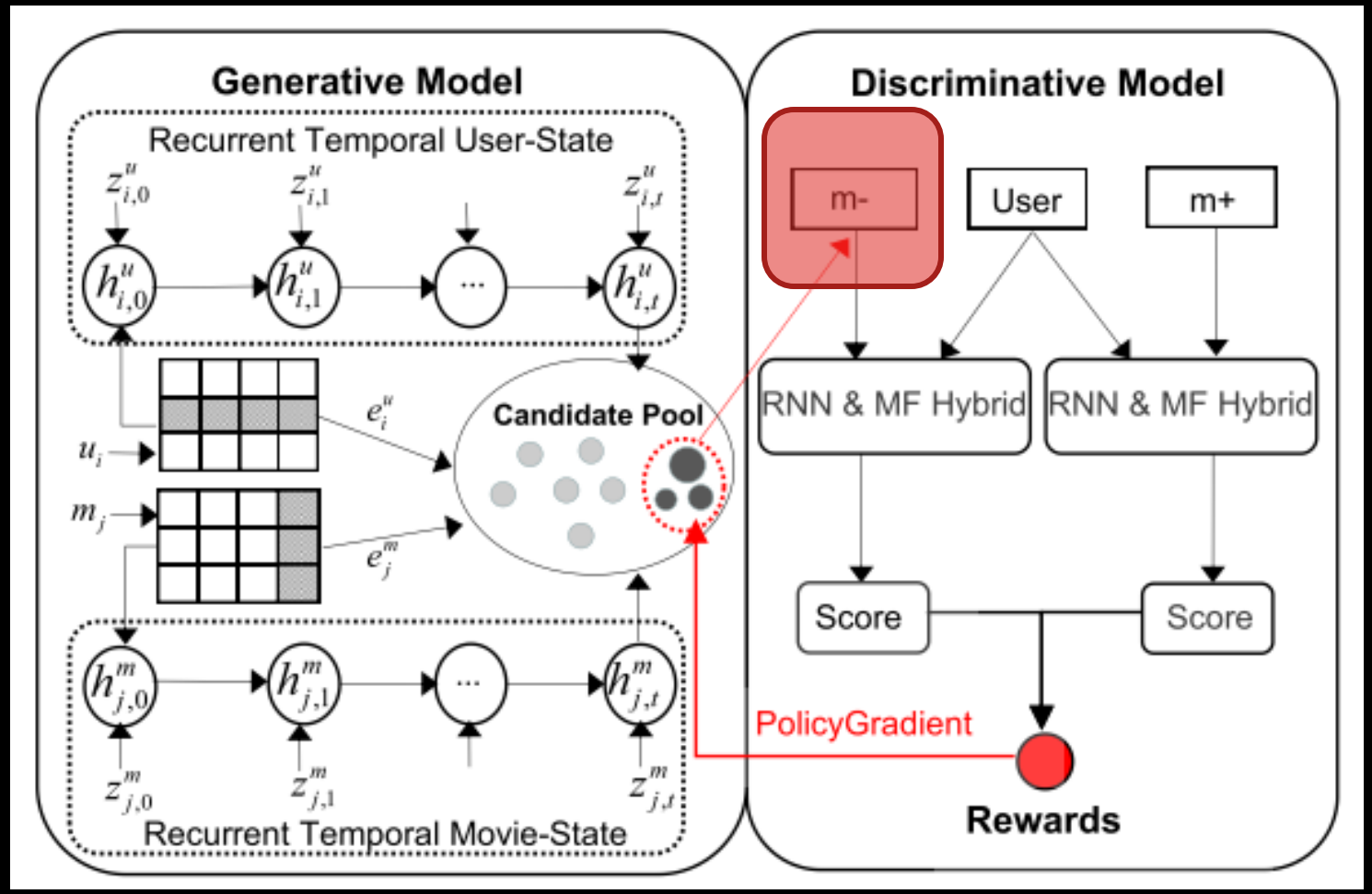
PLASTIC

[MIN YANG ET AL., IJCAI'19]

G takes a user at a specific time as input, **predicts** the recommendation list based on the **historical user-item interactions**

D is **SIAMESE** with two **point-wise** models (**MF + RNN**) that share parameters and are updated by minimizing a **pair-wise** loss. (hinge loss)

Discrete Item Sampling?
REINFORCE



Politecnico di BariCode

CONTEXTUAL RECOMMENDATION - **GEOGRAPHICAL:**

GEO-ALM

[ZHI-JIE WANG ET AL., IJCAI'19]

PROBLEM

1. **Negative samples** for **pair-wise** learning are randomly chosen and are **not informative**
2. How geographical information can be included in **POI recommendation task**

PROPOSAL

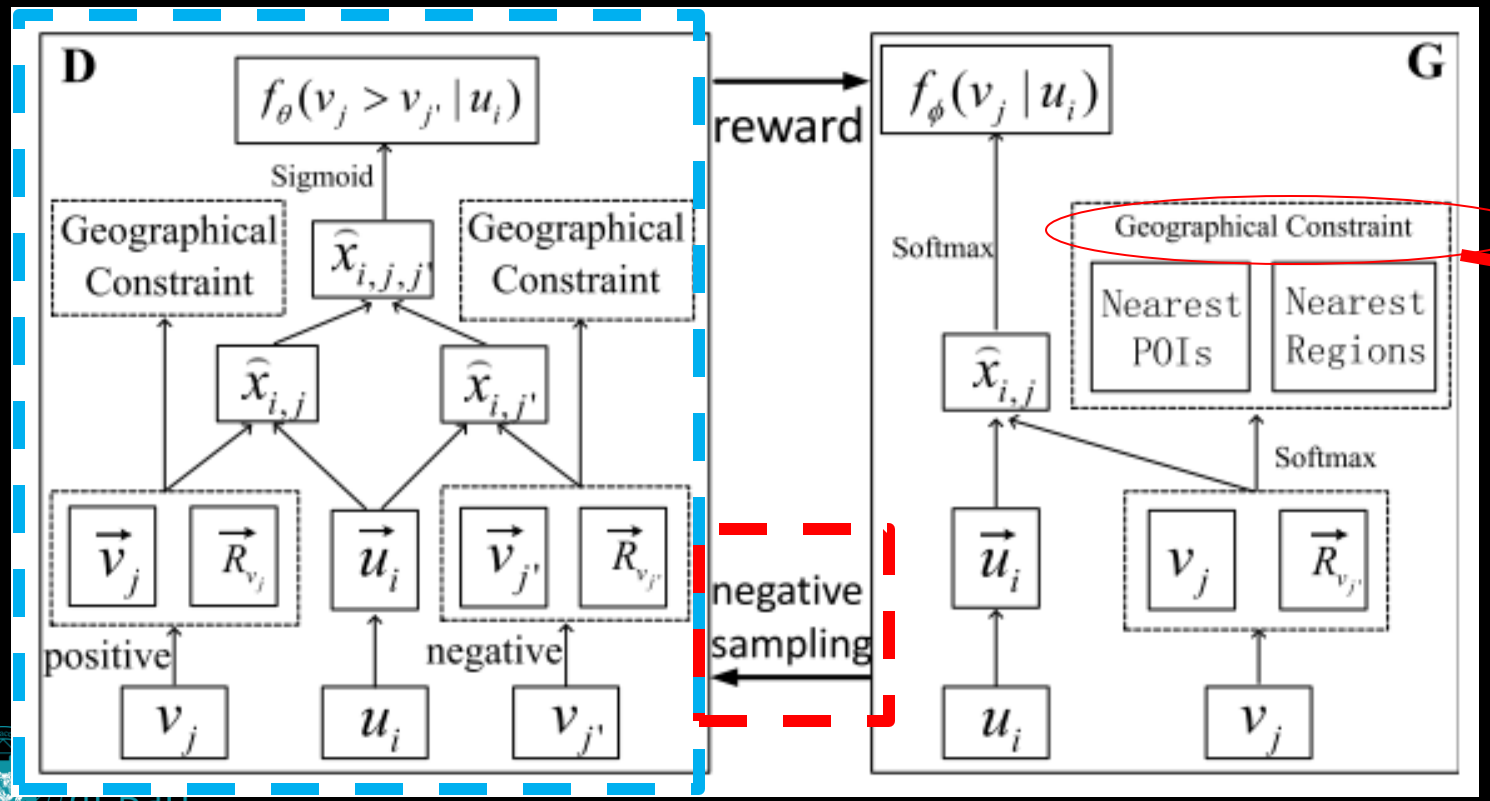
1. Exploit the generator to generate **more critical negative samples**
2. Integrate **POI** feature and **region** feature

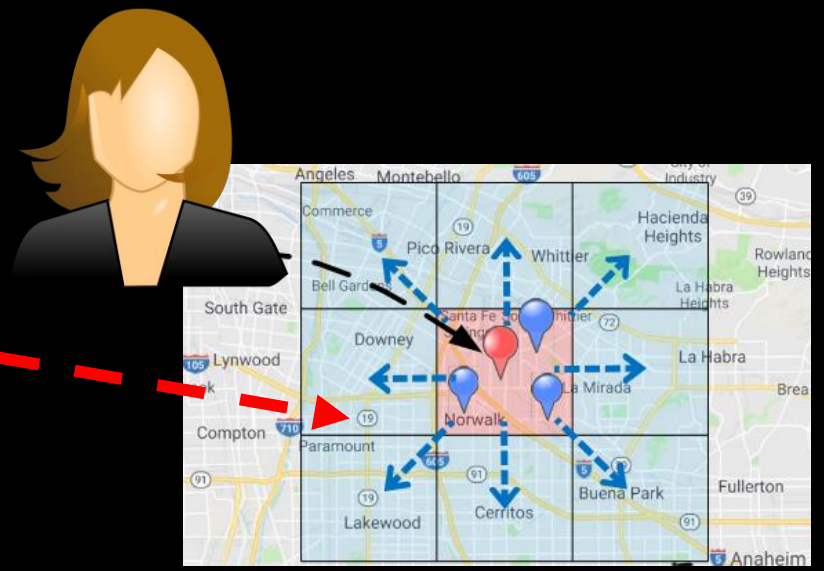
CONTEXTUAL RECOMMENDATION - **GEOGRAPHICAL:**

GEO-ALM

[ZHI-JIE WANG ET AL., IJCAI'19]

Acts as the RECOMMENDER





Discrete Item Sampling?
REINFORCE

CROSS-DOMAIN RECOMMENDATION:

DEEP ADVERSARIAL SOCIAL RECOMMENDATION

[WENQI FAN ET AL., IJCAI'19]

PROBLEM

1. Unify user representation for the **user-item interactions** (item domain) and **user-user connections** (social domain).
2. Negative sampling techniques to provide **informative** guidance during training

PROPOSAL

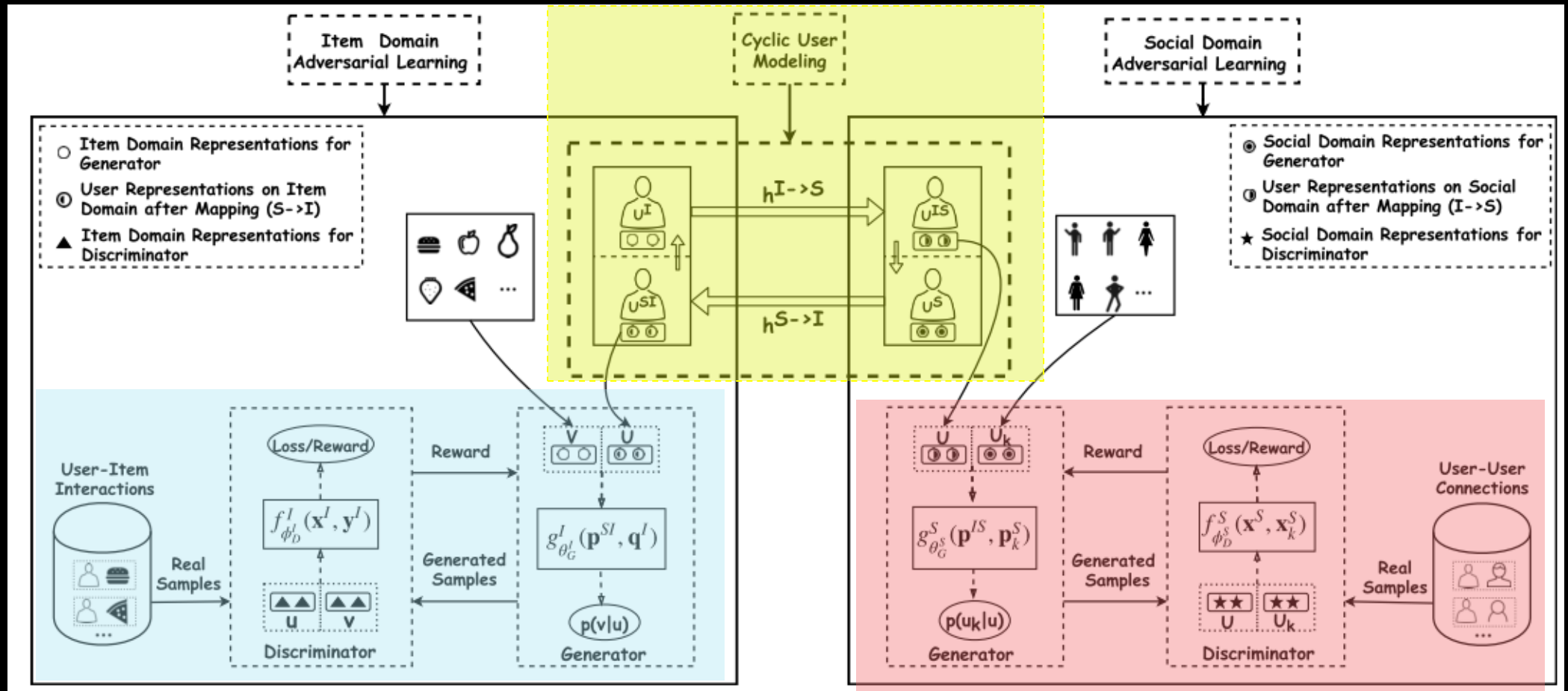
1. Adopts a nonlinear mapping method to **transfer** users' information between **social domain** and **item domain** using **adversarial learning**.
2. Generate **informative** negative samples to guide the training

CROSS-DOMAIN RECOMMENDATION:

DEEP ADVERSARIAL SOCIAL RECOMMENDATION

[WENQI FAN ET AL., IJCAI'19]

The model = **cyclic user modeling** + **item domain** and **social domain** adversarial learning.



Politecnico di Bari

CROSS-DOMAIN RECOMMENDATION:

DEEP ADVERSARIAL SOCIAL RECOMMENDATION

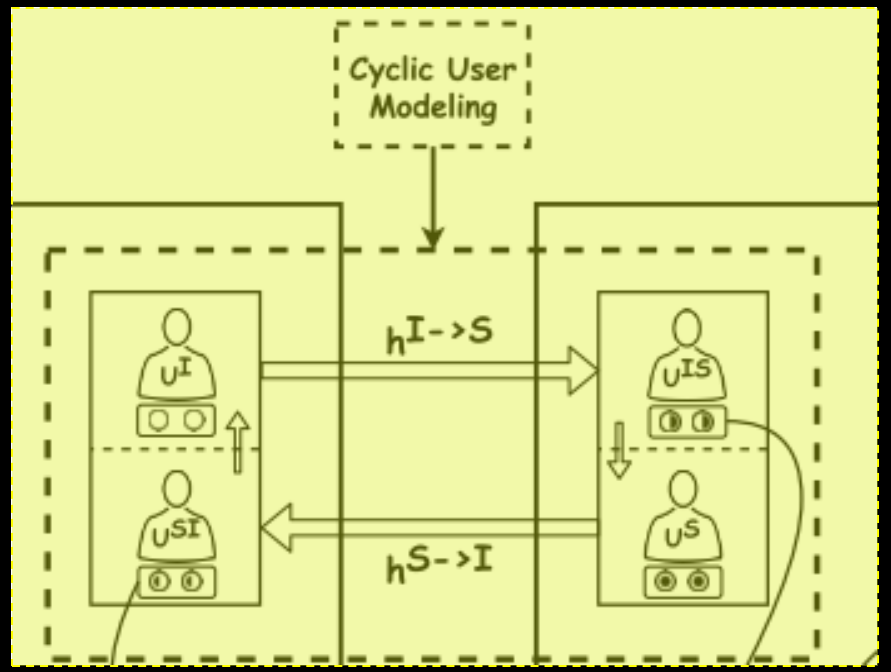
[WENQI FAN ET AL., IJCAI'19]

Cyclic user modeling

- transfer **user's information** from the **social domain** to the **item domain**
- transfer **user's information** from the **item domain** to the **social domain**

Learn an:

intra-domain user representation



Politecnico di Bari

CROSS-DOMAIN RECOMMENDATION:

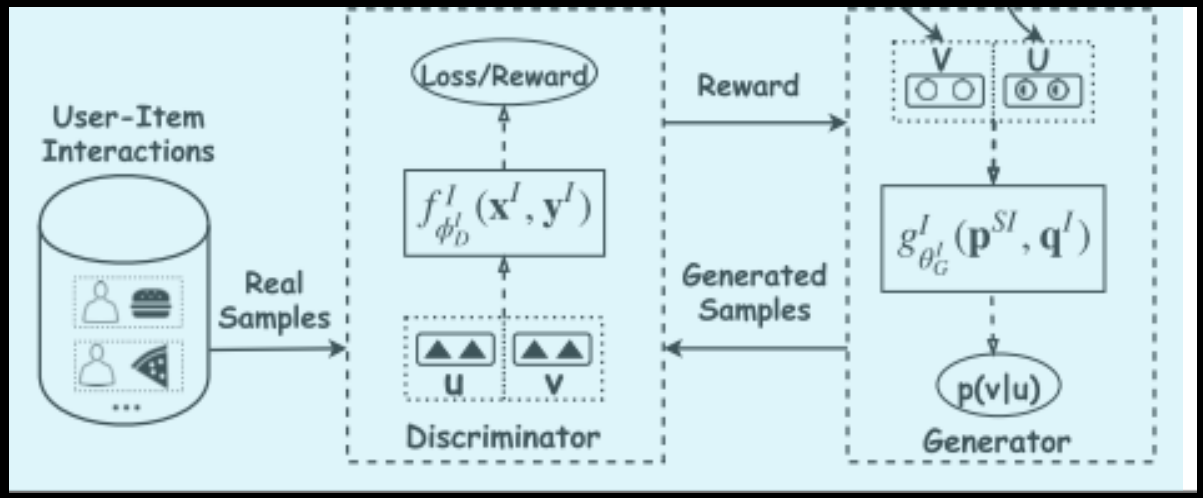
DEEP ADVERSARIAL SOCIAL RECOMMENDATION

[WENQI FAN ET AL., IJCAI'19]

Item domain adversarial learning

- Address the limitation of **negative sampling** for ranking recommendation.

G learns to produce **more informative negative** examples under the influence of **transferred social information**



Discrete Sampling?
REINFORCE

Politecnico di Bari

CROSS-DOMAIN RECOMMENDATION:

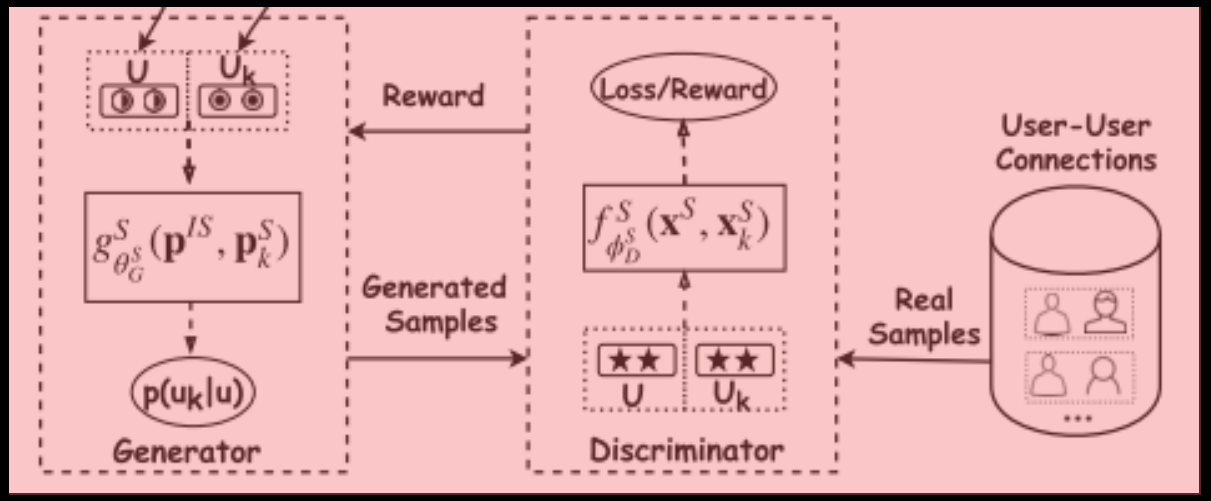
DEEP ADVERSARIAL SOCIAL RECOMMENDATION

[WENQI FAN ET AL., IJCAI'19]

Social domain adversarial learning

- Address the limitation of **negative sampling** for social recommendation.

G learns to produce **more informative negative** (most relevant social-connections) examples under the influence of **transferred item information**



Discrete Sampling? REINFORCE

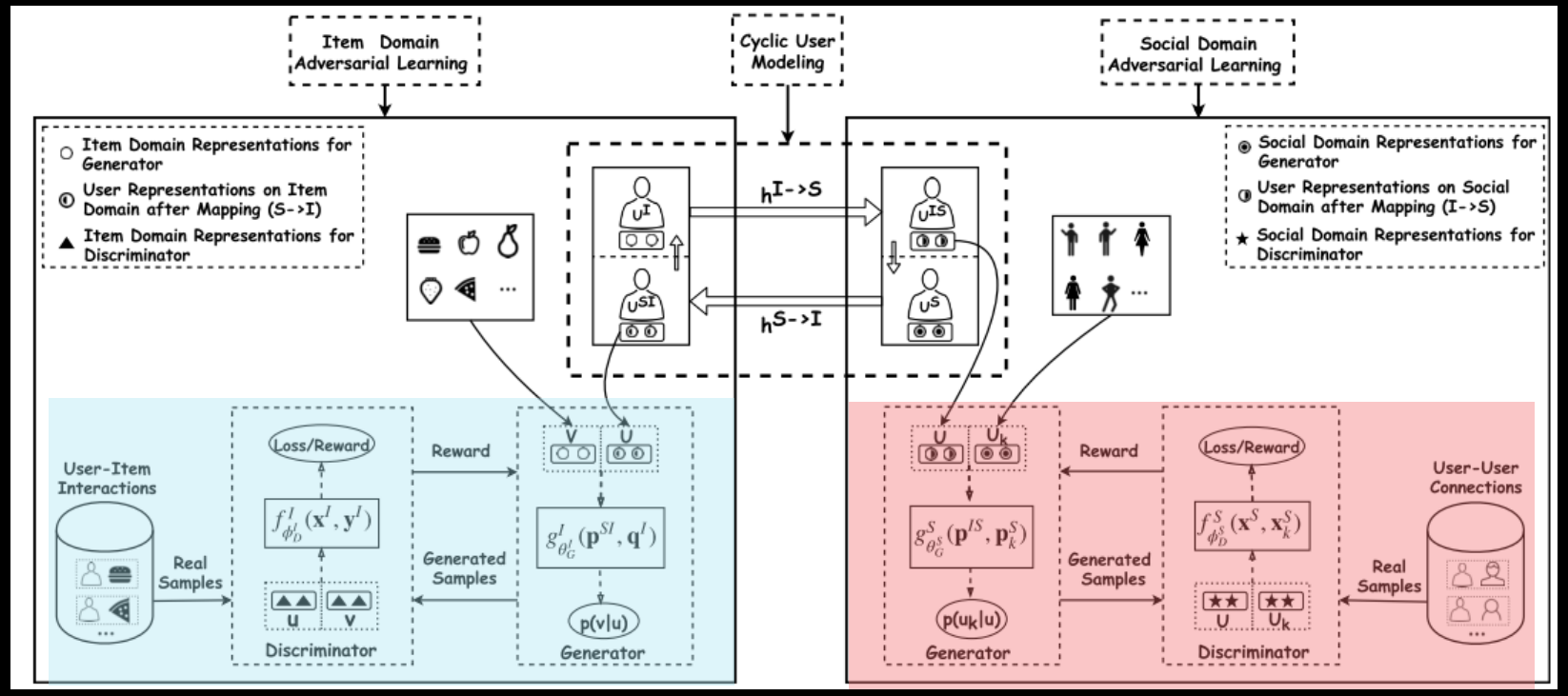
Politecnico di Bari

CROSS-DOMAIN RECOMMENDATION:

DEEP ADVERSARIAL SOCIAL RECOMMENDATION

[WENQI FAN ET AL., IJCAI'19]

When the training is finished, the embedding learned by the **social** and **item** generators are used to perform recommendation



Politecnico di Bari

CROSS-DOMAIN RECOMMENDATION:

DEEP ADVERSARIAL SOCIAL RECOMMENDATION

[WENQI FAN ET AL., IJCAI'19]

Datasets	Metrics	Algorithms						
		BPR	IRGAN	SBPR	SocialMF	DeepSoR	GraphRec	**DASO**
Ciao	Precision@3	0.0154	0.0274	0.0211	0.0260	0.0310	0.0374	**0.0462**
	Precision@5	0.0137	0.0245	0.0204	0.0218	0.0240	0.0326	**0.0451**
	Precision@10	0.0102	0.0239	0.0178	0.0155	0.0201	0.0265	**0.0375**
	NDCG@3	0.0254	0.0337	0.0316	0.0312	0.0380	0.0392	**0.0509**
	NDCG@5	0.0299	0.0350	0.0335	0.0364	0.0356	0.0373	**0.0514**
	NDCG@10	0.0315	0.0376	0.0379	0.0373	0.0396	0.0382	**0.0518**
Epinions	Precision@3	0.0046	0.0138	0.0096	0.0100	0.0105	0.0156	**0.0208**
	Precision@5	0.0042	0.0104	0.0089	0.0090	0.0098	0.0123	**0.0173**
	Precision@10	0.0035	0.0080	0.0066	0.0071	0.0086	0.0102	**0.0140**
	NDCG@3	0.0099	0.0175	0.0136	0.0176	0.0160	0.0183	**0.0226**
	NDCG@5	0.0128	0.0177	0.0152	0.0196	0.0183	0.0182	**0.0217**
	NDCG@10	0.0169	0.0202	0.0198	0.0202	0.0200	0.0217	**0.0234**

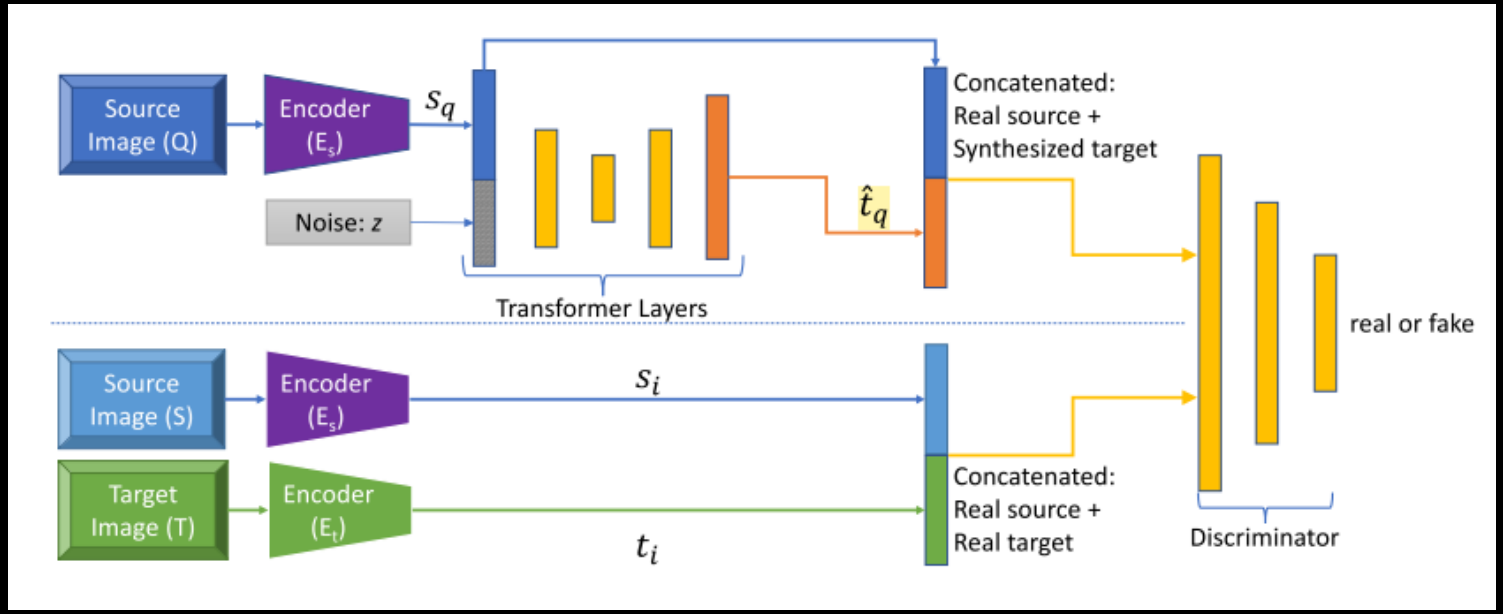
COMPLEMENTARY RECOMMENDATION: CRAFT

[HUYNH, CIPTADI ET AL., ECCV'18]

Generator: Transformer Network

learns to **generate** features of the **complementary items** conditioned on the query item.

Architecture for the CRAFT framework.

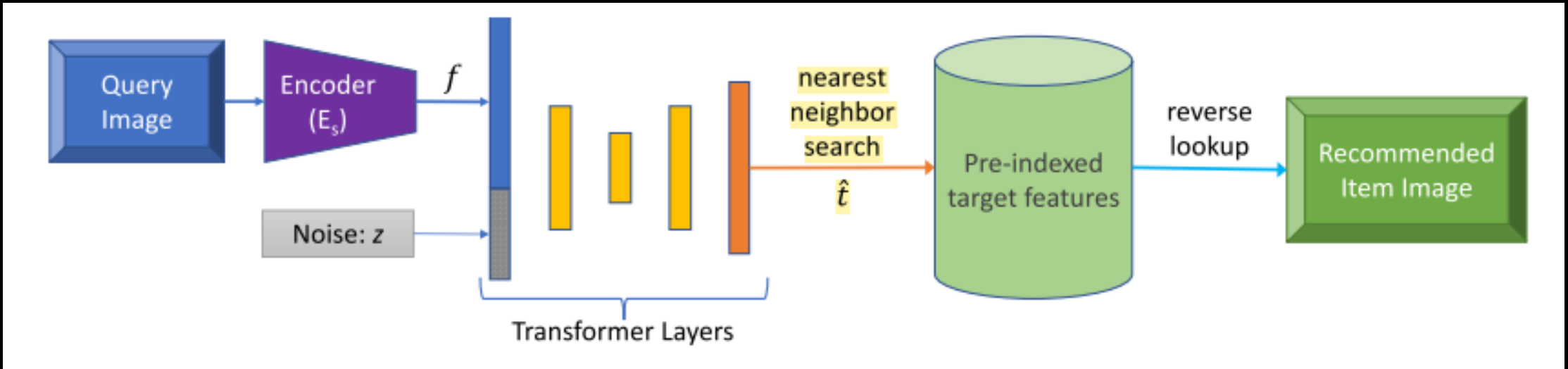


Politecnico di Bari

COMPLEMENTARY RECOMMENDATION: CRAFT

[HUYNH, CIPTADI ET AL., ECCV'18]

Generate Recommendations



COMPLEMENTARY RECOMMENDATION: CRAFT

[HUYNH, CIPTADI ET AL., ECCV'18]

Complementary Recommendations

QUERY

CRAFT

Nearest-neighbors

Incompatible

Green box are the accepted recommendation by a fashion specialist

Politecnico di Bari

COMPLEMENTARY RECOMMENDATION:
C+GAN
[KUMAR AND DAS GUPTA, AI FOR FASHION@KDD'19]

GOAL? find best fashion clothes when the user sets a query

$$p(bottom|top)$$

The **Generator** combines several loss components to model distinct characteristics:

1. **MSE** Loss: generated image stays close to the target image
2. **Perceptual** Loss: reconstructed image and the original input match each other under
3. **Adversarial** Loss: encourages the network to prefer solutions that reside on the manifold of natural images

COMPLEMENTARY RECOMMENDATION:

C+GAN

[KUMAR AND DAS GUPTA, AI FOR FASHION@KDD'19]

Politecnico di Bari

OTHER APPLICATIONS: SECURITY

ADVERSARIAL ATTACKS ON AN OBLIVIOUS RECOMMENDER

[CHRISTAKOPOULOU AND BANERJEE, RECSYS'19]

PROBLEM

1. user-rating matrix poisoning attack in an optimized way
2. Unnoticeability of fake profiles

PROPOSAL

1. Zero-order optimization techniques to overcome the challenge that the adversary does not have access to the recommender's gradient
2. Use GANs for generating realistic fake-user samples

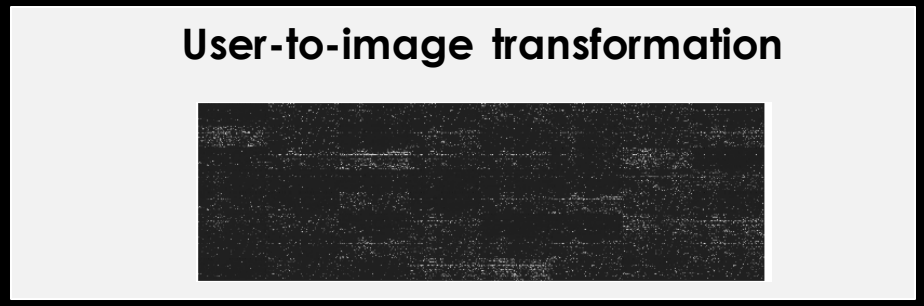
OTHER APPLICATIONS: SECURITY

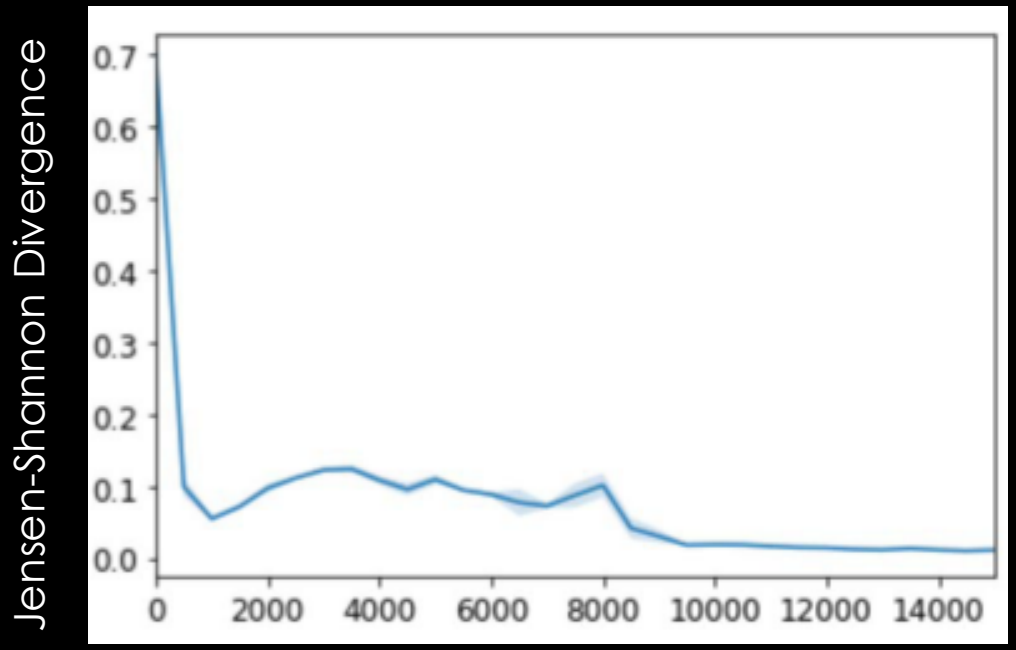
ADVERSARIAL ATTACKS ON AN OBLIVIOUS RECOMMENDER

[CHRISTAKOPOULOU AND BANERJEE, RECSYS'19]

METHOD:

Transform each user profile in an image to work with standard GAN for computer vision.





"GANs can produce fake user samples whose distribution is close to the real user distribution."

[CHRISTAKOPOULOU AND BANERJEE, RECSYS'19]

Politecnico di Bari

Name	Year	Generator							Discriminator					
		Linear	MLP	CNN	AE	VAE	RNN-LSTM	RNN-GRU	Linear	MLP	CNN	AE	RNN-LSTM	RNN-GRU
Collaborative Rec.														
IRGAN[126]	2017	✓							✓					
CFGAN[12]	2018		✓							✓				
Chae et a.[13]	2018				✓				✓					
CAAE[14]	2019				✓				✓					
CGAN[115]	2019					✓				✓				
CALF[20]	2019			✓							✓			
PD-GAN[133]	2019	✓							✓					
LambdaGAN[129]	2019	✓							✓					
VAEGAN[140]	2019					✓				✓				
APL[109]	2019	✓							✓					
RsyGAN[138]	2019				✓					✓				
Graph-based Collaborative Rec.														
GraphGAN[124]	2018	✓							✓					
GAN-HBNR [7]	2018				✓							✓		
VCGAN[148]	2018													
Hybrid Collaborative Rec.														
VAE-AR[62]	2017					✓				✓				
DVBPR[53]	2017			✓							✓			
RGD-TR[66]	2018		✓							✓				
aae-RS[137]	2018				✓					✓				
SDNet[18]	2019	✓							✓					
ATR[90]	2019							✓			✓			
AugCF[128]	2019		✓						✓					
RSGAN[139]	2019				✓				✓					
PRGAN[16]	2019	✓								✓				
CGAN[130]	2019	✓							✓					

SUMMARY

Prevalence of

Linear Models

and

Auto-Encoder

SUMMARY

Name	Year	Generator							Discriminator					
		Linear	MLP	CNN	AE	VAE	RNN-LSTM	RNN-GRU	Linear	MLP	CNN	AE	RNN-LSTM	RNN-GRU
Contextual Rec.														
Temporal-aware														
RecGAN[4]	2018							✓						✓
NMRN-GAN[127]	2018		✓										✓	
AAE[118]	2018				✓					✓				
PLASTIC[150]	2018	✓					✓		✓				✓	
LSIC[149]	2019	✓					✓		✓				✓	
GAN-LSTM[17]	2019						✓			✓				
Geographical-aware														
Geo-ALM[69]	2019	✓							✓					
APOIR[151]	2019							✓		✓				
Cross-domain Rec.														
VAE-GAN-CC[77]	2018					✓				✓				
RecSys-DAN[122]	2019			✓						✓				
CnGAN[86]	2019				✓					✓				
DASO[26]	2019		✓							✓				
Complementary Rec.														
CRAFT[48]	2018		✓							✓				
Yang et al.[136]	2018			✓							✓			
MrCGAN[106]	2018			✓							✓			
a^+GAN[61]	2019			✓							✓			

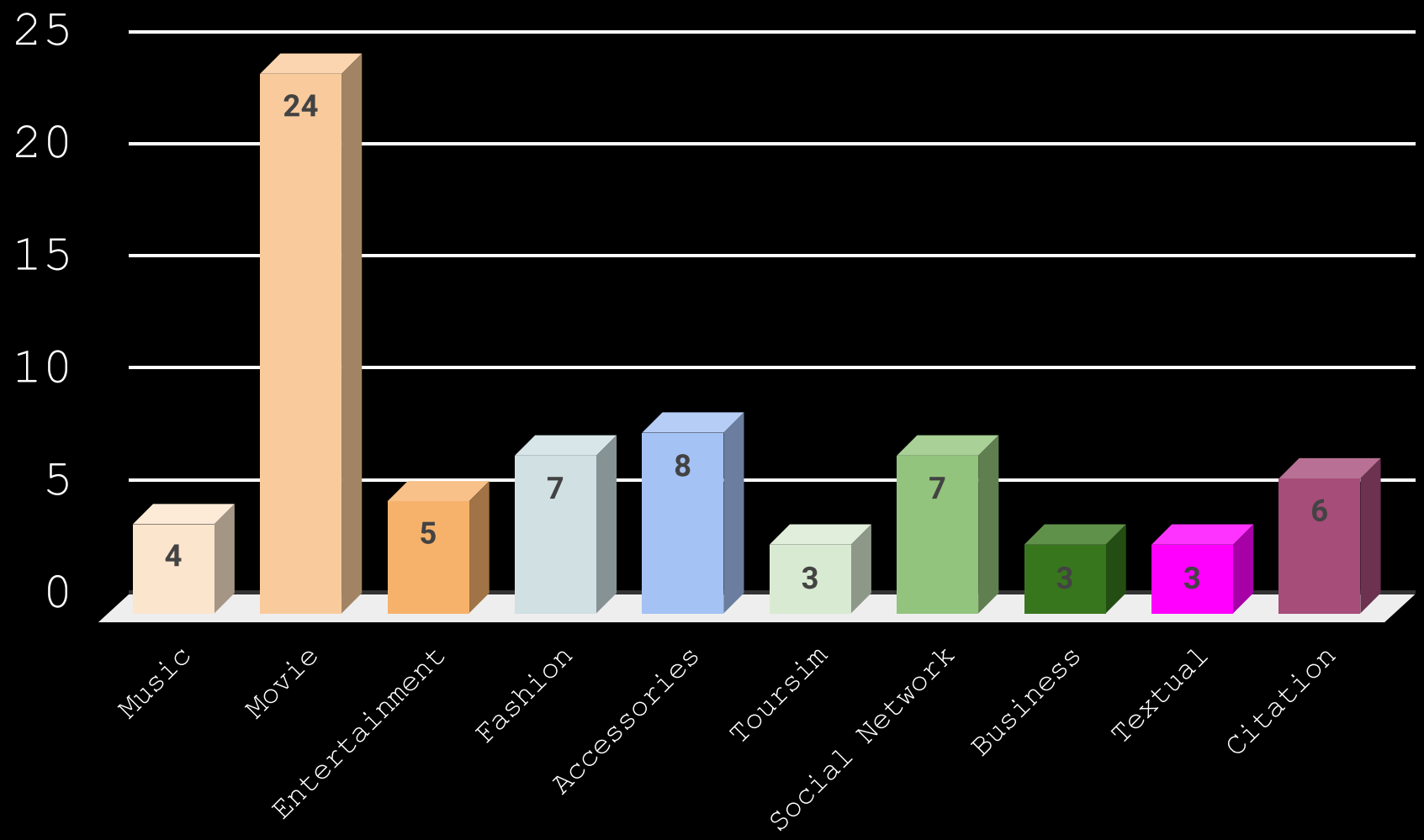
Prevalence **RNN**

Prevalence **CNN**

Politecnico di Bari

3.3 DOMAIN AND OPEN DIRECTIONS

DOMAIN



Politecnico di Bari

OPEN DIRECTIONS

- Novel solution for the discrete sampling non-differentiability problems
- Integrate user-personalization in complementary recommendation
- Extend GAN-RF to Knowledge-aware recommender systems
- Explore GAN-based attacks to Deep-CF RS
- Verify the effectiveness of GAN-RF with an online evaluation

LEVEL OF KNOWLEDGE OF THE ATTACKER

- **Black-box**: No access to the model only knows the **OUTPUT**
- **White-box**: knowledge of the data, architecture and parameters (**gradient**).
- **Grey-box:** any variation between white- and black-box

Who goes first?

- **Attacker** designs attacks based on the defense strategy
- **Defender** designs defense based on the attacker's knowledge

GOAL OF THE ATTACKER

- **Targeted**
  - Mislead the **classifier** to predict a **target** label
- **Non-targeted**
  - Mislead the **classifier** to predict a **arbitrary** label

****** and in RS? ******

DATA POISONING OPTIMIZATION

- "Classic" Poisoning Attacks:
  - Label Modification
  - Poison Insertion
  - Data Modification
  - Boiling Frog

- Main limits:
  - Empirical techniques
  - Heuristics
  - No optimization procedure (neither Adversarial Learning) to maximize the attacker's utility

DATA POISONING OPTIMIZATION

Which kinds of attacker's utilities could be maximized?

- **Availability attack:** Attacker tries to maximize the estimated error in the target domain
- **Integrity attack:** Attacker's goal is to increase (or decrease) the popularity of a subset of items
- **Hybrid attack:** A mixture of availability attack and integrity attack

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

- Data Poisoning Optimization is DIFFERENT (or better it is another level of optimization) from the classic optimization for a recommendation utility

- What we need are:
  - the target recommendation model;
  - an attacker utility;
  - an optimization procedure for that utility.

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

- In principle, every model could be optimized for data poisoning attacks

- In the literature, we find (in descending popularity order):
  1. Factorization-based models
  2. Reinforcement Learning-based models
  3. Graph-based models
  4. K-NN models
  5. others

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

- In principle, every model could be optimized for data poisoning attacks

- In the literature, we find (in descending popularity order):
  1. **Factorization-based models**
  2. **Reinforcement Learning-based models**
  3. Graph-based models
  4. K-NN models
  5. others

POISONING FACTORIZATION-BASED MODELS

A Bi-level optimization problem

- Constrained optimization: first-order Karush-Kuhn-Tucker (KKT) conditions
- Outer optimization maximizes the attackers' utilities
- Inner optimization maximizes the recommendation utility of the poisoned data models

POISONING FACTORIZATION-BASED MODELS

Attacker Utilities/strategies observed in the literature:

- Availability, Integrity, Hybrid attacks
- Hit ratio maximization/minimization to promote/demote items (Warning: Hit ratio is not directly optimizable) usually optimized by using the Wilcoxon-Mann-Whitney loss
- Exploitation of a subset of Influencial users to modify top-N recommendations

POISONING FACTORIZATION-BASED MODELS

Defensive strategies observed in the literature:

- SVM classifier to detect fake profiles trained on specific features/indicators: RDMA,WDMA,WDA,TMF,FMTD, andMeanVar
- Trim Learning to conduct a "clean" learning of the recommendation model
- Robustness analyzer to certify a factorization model as robust

POISONING REINFORCEMENT LEARNING-BASED MODELS

What we need to define:

- The recommendation model (or a "proxy", or.. nothing)
- The **attacker's knowledge** and capability(that are common even to the other attack families);
- The **state space** (usually the sequence of actions, or an embedding);
- The **action space** (usually, the items that can be chosen, or an embedding);
- The **reward utility**.

POISONING REINFORCEMENT LEARNING-BASED MODELS

How can we represent the target recommender?

- Suppose you know it (not realistic rarely adopted with reinforcement learning attack strategies)
- A recommender simulator as an ensemble of several representative recommendation models
- A black-box: we only take information by exploiting proxy indicators

POISONING REINFORCEMENT LEARNING-BASED MODELS

What about the reward utilities?

- predicted **weighted average influence** of the target samples
- the number of **Page View (PV)** for certain items

POISONING REINFORCEMENT LEARNING-BASED MODELS

How to defend a reinforcement learning-based recommender system?

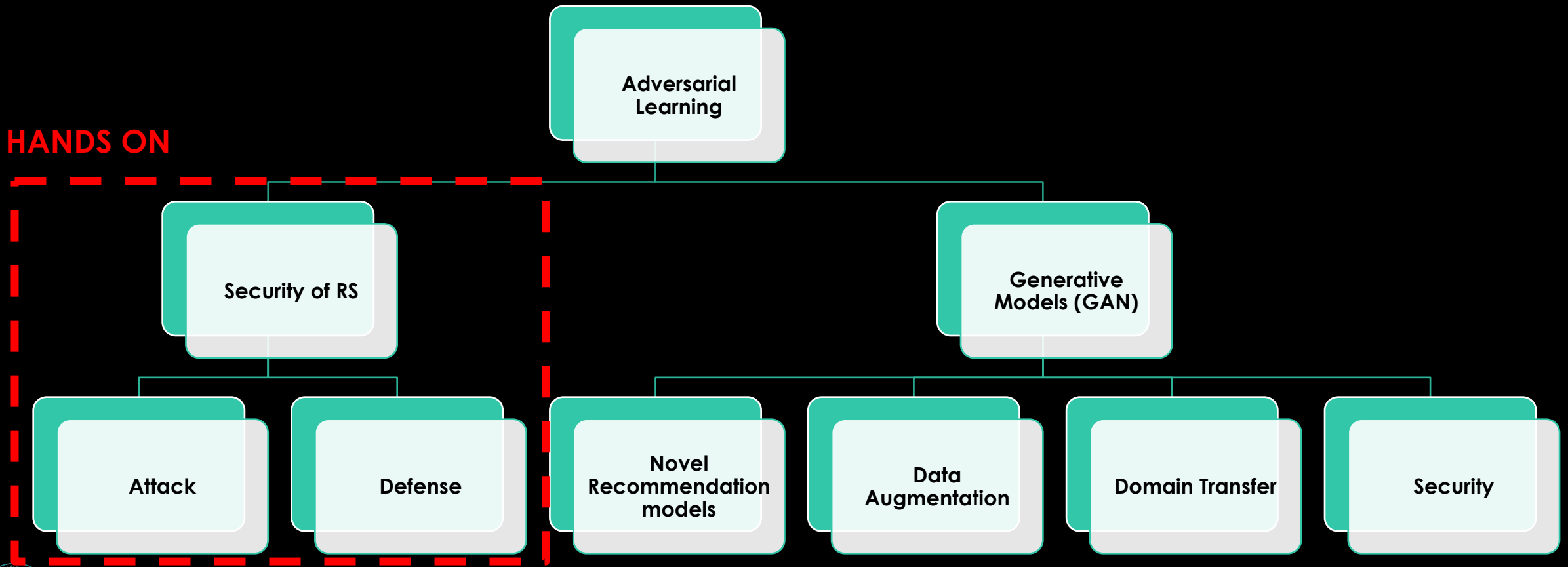
Supervised classifier that detects the adversarial examples by exploiting the reinforcement agent's actions:

- In detail, leveraging a **GRU encoder** and an **attention-based decoder** to detect the adversarial examples.
- The **GRU encoder** encodes the action methods into a low dimensional feature vector.
- The **attention-based decoder** takes as input the action embedding, the hidden states, and the encoder output

HANDS-ON SESSION
ADVERSARIAL REGULARIZATION

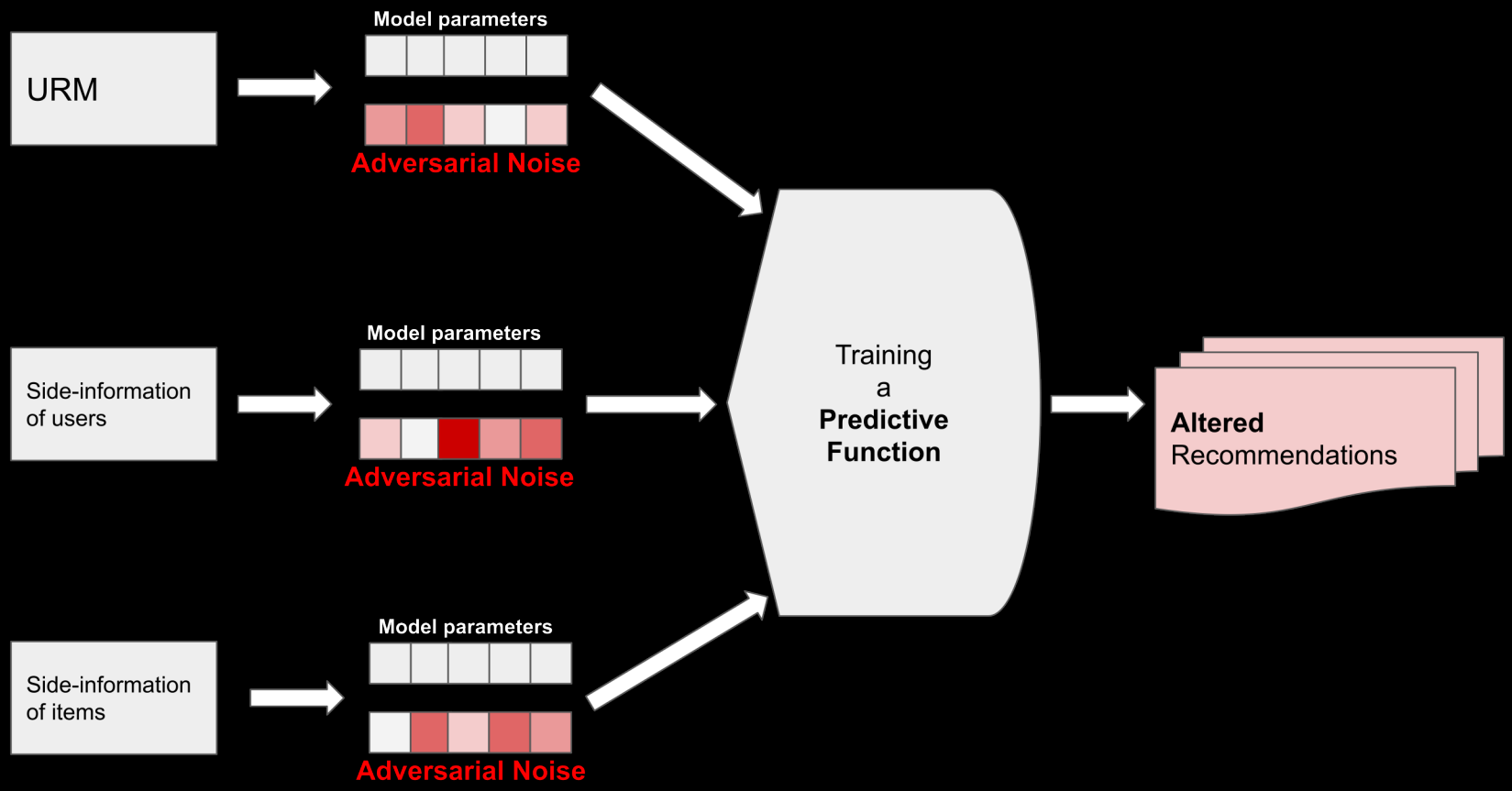
Presenter: Felice Antonio Merra

ADVERSARIAL LEARNING FOR RS



Politecnico di Bari

ADVERSARIAL ROBUST RECOMMENDATION



Politecnico di Bari

HOW TO ACCESS THE JUPYTER NOTEBOOK

README.md

Adversarial Learning for Recommendation: Applications for Security and Generative Tasks – Concept to Code

To start the ***Hands-on Session*** access the Jupyter Notebook: launch binder

Source: 🤖Adversarial Learning for Recommendation: Applications for Security and Generative Tasks – Concept to Code repository_

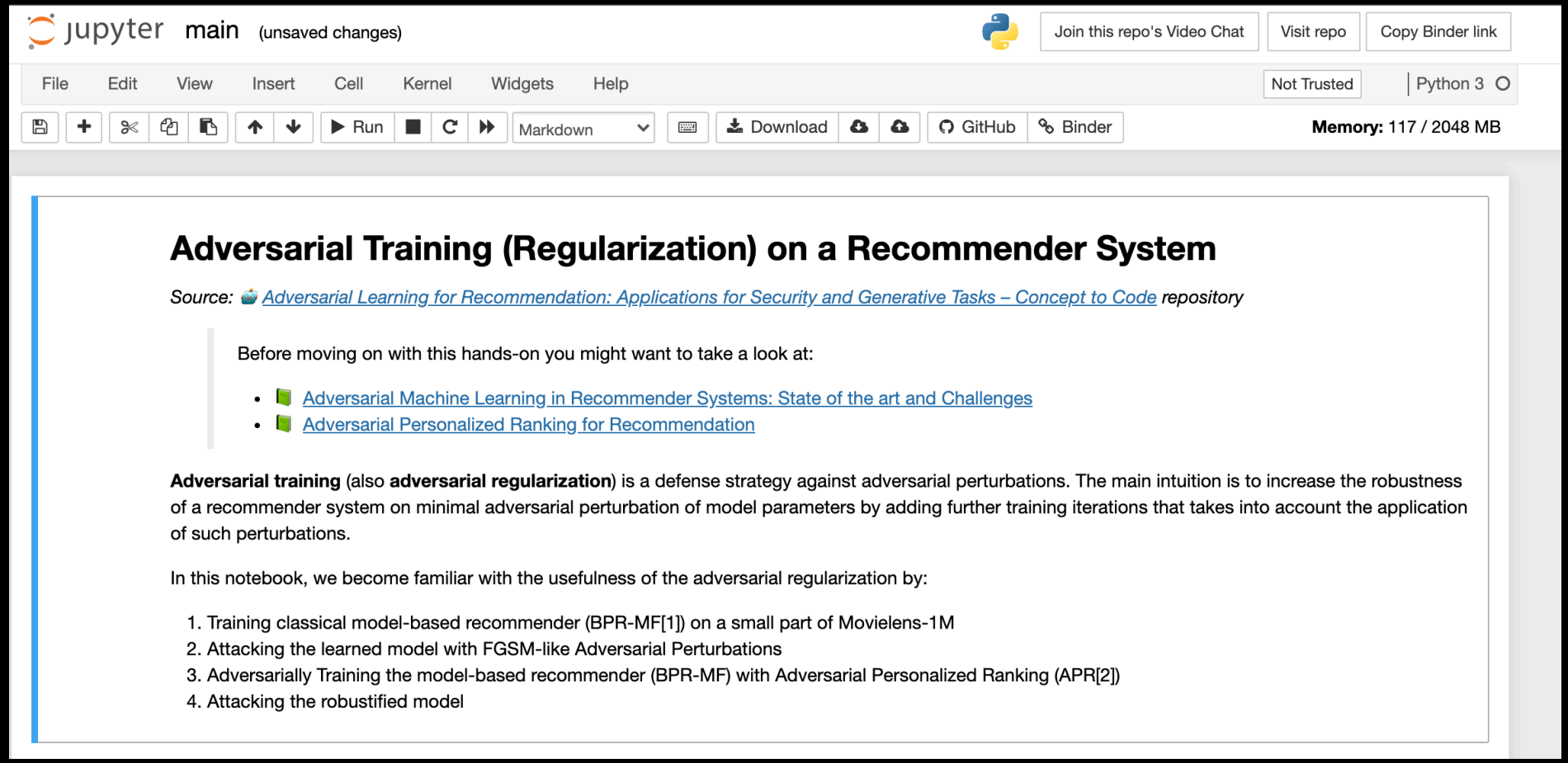
> Before moving on with this hands-on you might want to take a look at:
>
> - 📗 Adversarial Machine Learning in Recommender Systems: State of the art and Challenges
> - 📗 Adversarial Personalized Ranking for Recommendation

Click The Binder Widget

**It takes few minutes to build the interactive environment.*

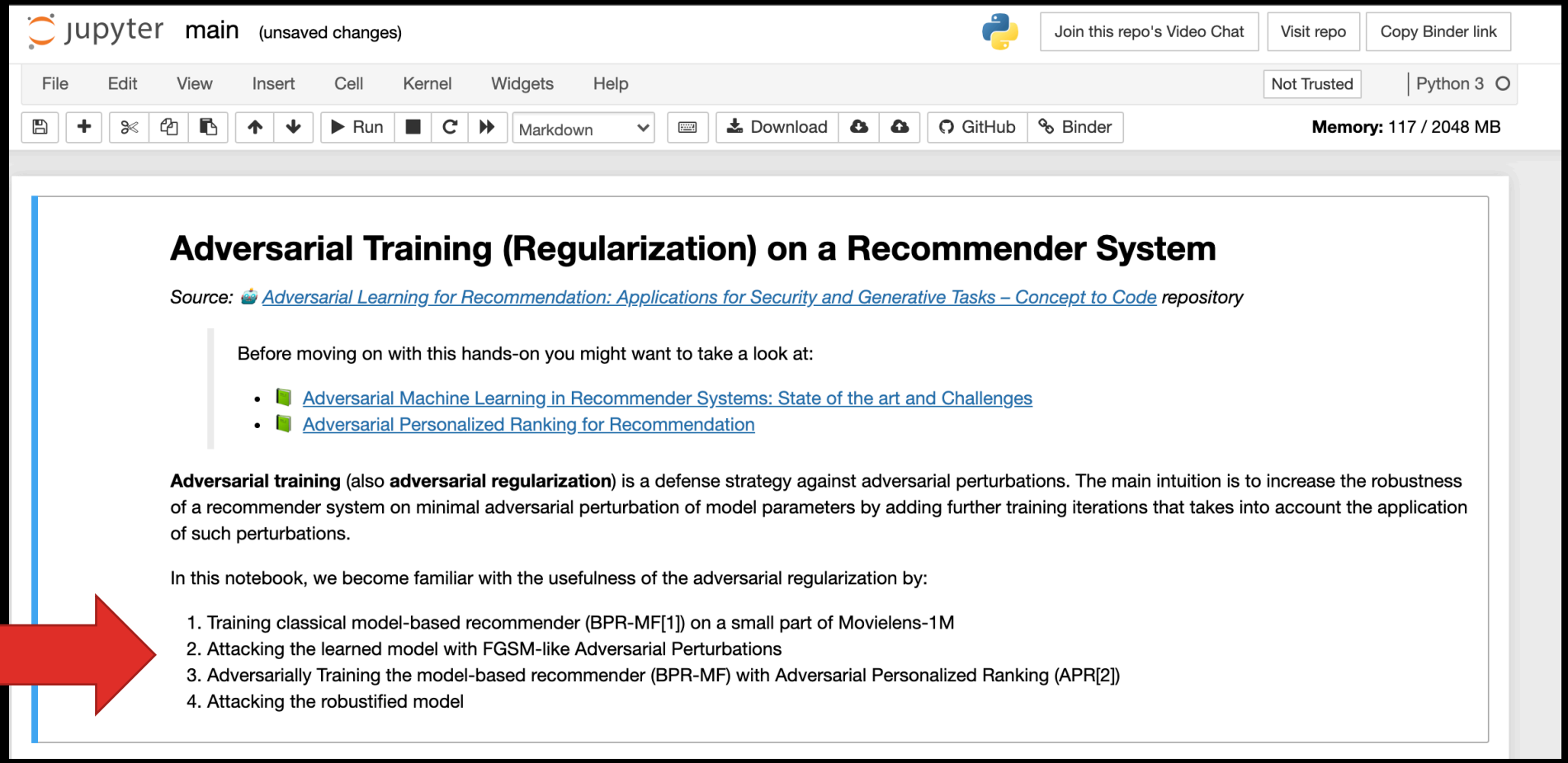
Politecnico di Bari

JUPYTER NOTEBOOK

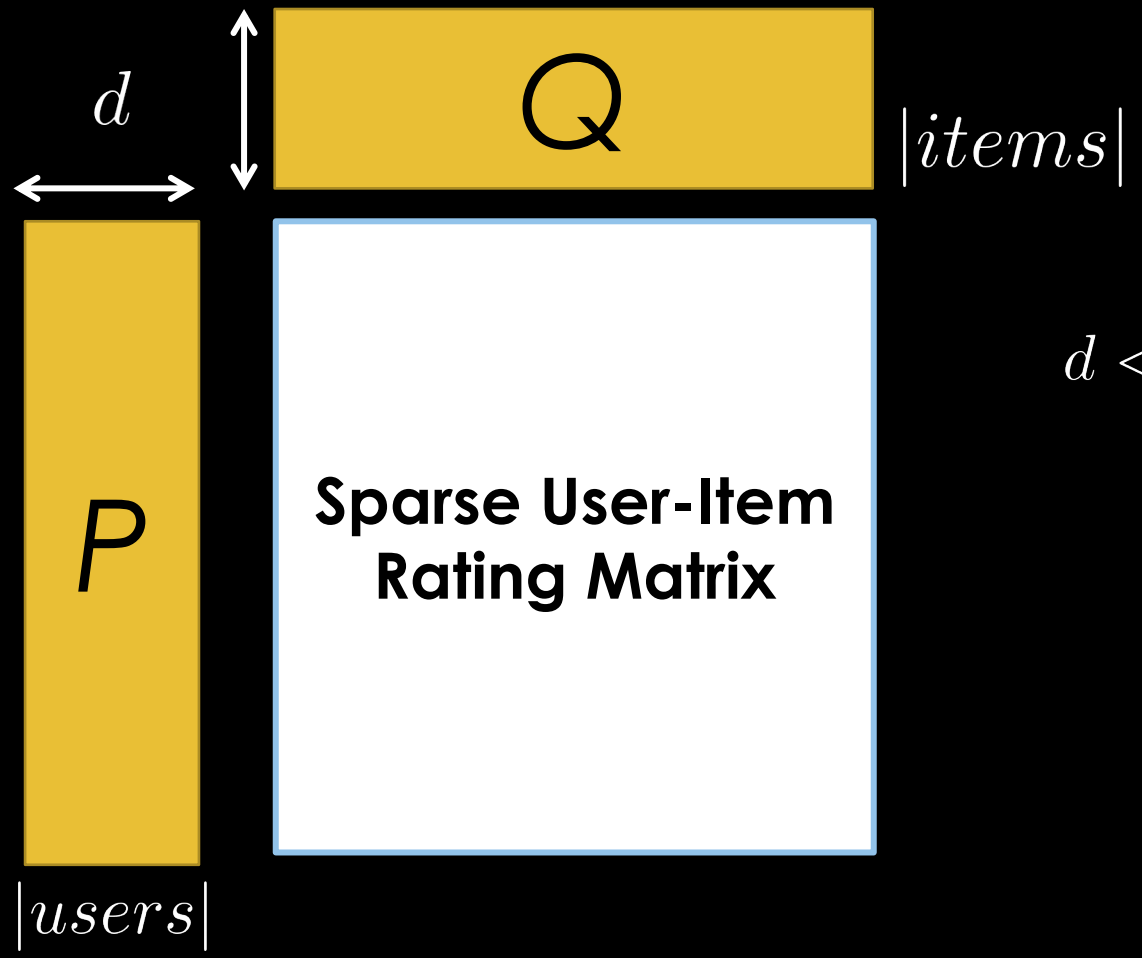


Politecnico di Bari

JUPYTER NOTEBOOK



THE RECOMMENDER MODEL
BPR-MF



$$d << |users|, |items|$$

IMPLEMENTATION BPR-MF

Define The Model

We will define a new Tensorflow 2 model class to define the model (BPR-MF). For a matter of simplicity we have also implemented the adversarial attack and defense strategies,, that will be used in the later sections.

In [ ]:

```
from src.recommender.RecommenderModel import RecommenderModel

TOPK = 100 # Top-K

class BPRMF(RecommenderModel):
    def __init__(self, data_loader, path_output_rec_result, path_output_rec_weight):
        super(BPRMF, self).__init__(data_loader, path_output_rec_result, path_output_rec_weight, 'bprmf')
        self.embedding_size = 64
        self.learning_rate = 0.05
        self.reg = 0
        self.epochs = 5
        self.batch_size = 512
        self.verbose = 1
        self.evaluator = Evaluator(self, data, TOPK)

        self.initialize_model_parameters()
        self.initialize_perturbations()
        self.initialize_optimizer()

    def initialize_model_parameters(self):
        """
            Initialize Model Parameters
        """
        self.embedding_P = tf.Variable(tf.random.truncated_normal(shape=[self.num_users, self.embedding_size], mean=0.0
        self.embedding_Q = tf.Variable(tf.random.truncated_normal(shape=[self.num_items, self.embedding_size], mean=0.0
```

Politecnico di Bari

TRAIN THE CLEAN MODEL

TRAIN STEP

TRAIN LOOP

```python
@timethis
def _train_step(self, batches):
    """ Apply a Single Training Step (across all the batches in the dataset). """
    user_input, item_input_pos, item_input_neg = batches

    for batch_idx in range(len(user_input)):
        with tf.GradientTape() as t:
            t.watch([self.embedding_P, self.embedding_Q])

            # Model Inference
            self.output_pos, embed_p_pos, embed_q_pos = self.get_inference(user_input[batch_idx],
                                                                           item_input_pos[batch_idx])
            self.output_neg, embed_p_neg, embed_q_neg = self.get_inference(user_input[batch_idx],
                                                                           item_input_neg[batch_idx])
            self.result = tf.clip_by_value(self.output_pos - self.output_neg, -80.0, 1e8)

            self.loss = tf.reduce_sum(tf.nn.softplus(-self.result))

            # Regularization Component
            self.reg_loss = self.reg * tf.reduce_mean(tf.square(embed_p_pos) + tf.square(embed_q_pos)
                                                      + tf.square(embed_q_neg))

            # Loss Function
            self.loss_opt = self.loss + self.reg_loss

        gradients = t.gradient(self.loss_opt, [self.embedding_P, self.embedding_Q])
        self.optimizer.apply_gradients(zip(gradients, [self.embedding_P, self.embedding_Q]))

@timethis
def train(self):
    for epoch in range(self.epochs):
        batches = self.data.shuffle(self.batch_size)
        self._train_step(batches)
        print('Epoch {0}/{1}'.format(epoch+1, self.epochs))
```

Politecnico di Bari

EVALUATE THE MODEL

Initialize and Train The Model

Now, we are ready to initialize and train the model.

```
recommender_model = BPRMF(data, '../rec_result/', '../rec_weights/')
recommender_model.train()
```

Evaluate The Model

The evaluation is computed on TOPK recommendation lists (default K = 100).

```
before_adv_hr, before_adv_ndcg, before_adv_auc = recommender_model.evaluator.evaluate()
```

ADV-RF: ATTACKS

- **ATTACKER Goal? Maximize** the recommender model objective

$$\Delta_{adv} = \arg \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

- **Attack Method? FGSM**

$$\Delta_{adv} = \epsilon \frac{\Pi}{\|\Pi\|} \quad \text{where} \quad \Pi = \frac{\partial J(X|\Omega + \Delta)}{\partial \Delta}$$

IMPLEMENTING THE ADVERSARIAL ATTACK

```python
def execute_adversarial_attack(self, epsilon):
    user_input, item_input_pos, item_input_neg = self.data.shuffle(len(self.data._user_input))
    self.initialize_perturbations()

    with tf.GradientTape() as tape_adv:
        tape_adv.watch([self.embedding_P, self.embedding_Q])
        # Evaluate Current Model Inference
        output_pos, embed_p_pos, embed_q_pos = self.get_inference(user_input[0],
                                                                  item_input_pos[0])
        output_neg, embed_p_neg, embed_q_neg = self.get_inference(user_input[0],
                                                                  item_input_neg[0])
        result = tf.clip_by_value(output_pos - output_neg, -80.0, 1e8)
        loss = tf.reduce_sum(tf.nn.softplus(-result))
        loss += self.reg * tf.reduce_mean(
            tf.square(embed_p_pos) + tf.square(embed_q_pos) + tf.square(embed_q_neg))
    # Evaluate the Gradient
    grad_P, grad_Q = tape_adv.gradient(loss, [self.embedding_P, self.embedding_Q])
    grad_P, grad_Q = tf.stop_gradient(grad_P), tf.stop_gradient(grad_Q)

    # Use the Gradient to Build the Adversarial Perturbations (https://doi.org/10.1145/3209978.3209981)
    self.delta_P = tf.nn.l2_normalize(grad_P, 1) * epsilon
    self.delta_Q = tf.nn.l2_normalize(grad_Q, 1) * epsilon
```

Politecnico di Bari

EVALUATING THE ATTACK

Adversarial Attack Against The Model

We can attack the model with adversarial perturbation and measure the performance after the attack. Epsilon is the perturbation budget.

```
epsilon = 0.5
print('Running the Attack with Epsilon = {0}'.format(epsilon))
recommender_model.execute_adversarial_attack(epsilon=epsilon)
print('The model has been Adversarially Perturbed.')
```

Evaluate the Effects of the Adversarial Attack

We will now evaluate the performance of the attacked model.

```
after_adv_hr, after_adv_ndcg, after_adv_auc = recommender_model.evaluator.evaluate()

print('HR decreases by %.2f%%' % ((1-after_adv_hr/before_adv_hr)*100))
print('nDCG decreases by %.2f%%' % ((1-after_adv_ndcg/before_adv_ndcg)*100))
print('AUC decreases by %.2f%%' % ((1-after_adv_auc/before_adv_auc)*100))
```

Politecnico di Bari

ADV-RF: **DEFENSE**

- **Defender Goal? Minimize** the attack influence
- **Defence Method? Adversarial (re)training**

$$\arg\min_{\Omega} J(X|\Omega) + \lambda J\left(X|\Omega + \Delta_{adv}\right)$$

$$where \quad \Delta_{adv} = \arg\max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

MINIMAX GAME

The training process is a **MINIMAX GAME**

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

IMPLEMENTING THE DEFENSE

BPR-MF Loss

Regularization

Adversarial Attack

Adversarial Regularizer

Adversarial Regularized Loss Function

```
self.loss = tf.reduce_sum(tf.nn.softplus(-self.result))

# Regularization Component
self.reg_loss = self.reg * tf.reduce_mean(tf.square(embed_p_pos) + tf.square(embed_q_pos) + tf.square(e

# Adversarial Regularization Component
##  Execute the Adversarial Attack on the Current Model (Perturb Model Parameters)
self.execute_adversarial_attack(epsilon)
##  Inference on the Adversarial Perturbed Model
self.output_pos_adver, _, _ = self.get_inference(user_input[batch_idx], item_input_pos[batch_idx])
self.output_neg_adver, _, _ = self.get_inference(user_input[batch_idx], item_input_neg[batch_idx])

self.result_adver = tf.clip_by_value(self.output_pos_adver - self.output_neg_adver, -80.0, 1e8)
self.loss_adver = tf.reduce_sum(tf.nn.softplus(-self.result_adver))

# Loss Function
self.adversarial_regularizer = adv_reg * self.loss_adver # AMF = Adversarial Matrix Factorization
self.bprmf_loss = self.loss + self.reg_loss

self.amf_loss = self.bprmf_loss + self.adversarial_regularizer
```

Politecnico di Bari

EVALUATING THE ATTACK ON THE DEFENDED MODEL

Evaluated The Adversarially Defended Model before the Attack

```
before_adv_hr, before_adv_ndcg, before_adv_auc = recommender_model.evaluator.evaluate()
```

Adversarial Attack Against The Defended Model

```
recommender_model.execute_adversarial_attack(epsilon=0.5)
```

Evaluate the Effects of the Adversarial Attack against the Defended Model

```
after_adv_hr, after_adv_ndcg, after_adv_auc = recommender_model.evaluator.evaluate()

print('HR decreases by %.2f%%' % ((1-after_adv_hr/before_adv_hr)*100))
print('nDCG decreases by %.2f%%' % ((1-after_adv_ndcg/before_adv_ndcg)*100))
print('AUC decreases by %.2f%%' % ((1-after_adv_auc/before_adv_auc)*100))
```

Politecnico di Bari

LIVE HANDS-ON SESSIONS

Tutorial 6A Adversarial Learning	Sep 25 14:30 Pacific
	Sep 25 23:30 CET
	Sep 26 05:30 Shangai
Tutorial 6B Adversarial Learning	Sep 26 01:30 Pacific
	Sep 26 10:30 CET
	Sep 26 16:30 Shangai

Politecnico
di Bari