**UNIVERSIDAD COMPLUTENSE DE MADRID**
FACULTAD DE CIENCIAS MATEMÁTICAS


# TESIS DOCTORAL

## Contribuciones a la Seguridad del Aprendizaje Automático

## (Contributions to the Security of Machine Learning)

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

**Roi Naveiro Flores**

Directores

**David Ríos Insua**
**David Gómez-Ullate**

Madrid

Programa de doctorado en Ingeniería Matemática,
Estadística e Investigación Operativa por la
Universidad Complutense de Madrid y la
Universidad Politécnica de Madrid
FACULTAD DE CIENCIAS MATEMÁTICAS (UCM)



---

# Contribuciones a la Seguridad del Aprendizaje Automático

(Contributions to the Security of Machine Learning)

Tesis Doctoral

**Roi Naveiro Flores**

DIRECTORES

David Ríos Insua

David Gómez-Ullate

Año 2020

# Agradecimientos

El apoyo de mi director David Ríos Insua ha sido condición necesaria (y diría que suficiente) para la realización de esta tesis doctoral. Gracias. Hago extensivo mi agradecimiento a todos los investigadores con los que he tenido el placer –y muchas veces el honor– de trabajar, con mención especial a mi co-director David Gómez-Ullate, y al profesor David Banks, que me brindó la oportunidad de realizar dos estancias en Duke University, donde tanto he aprendido.

Eskerrak bihotzez nire ama Lourdesi, jakintzaz beterik zaudelako eta hutsegiteen aurrean arduraz jokatzen irakatsi didazulako, zuregan eredu izan ditudan irmotasun eta ausardiaz. Grazas ao meu pai, por contaxiarme a súa sensibilidade, a súa admiración pola natureza, o seu gusto pola literatura (se é en galego, mellor), e por facerme unha persoa loitadora e combativa. Gracias a mi hermana Amaia, a quien secretamente admiro, has sido siempre un apoyo indispensable, particularmente en tiempos de pandemia; a mi hermana Mariña, un referente inalcanzable; y a mis sobrinas Helena, Inés y Alicia, por tantos momentos.

Para concluír, quiero agradecer al Ministerio de Educación la financiación de mi investigación a través de la beca FPU15-03636. Además, quiero reconocer el apoyo brindado por el Ministerio de Ciencia mediante el proyecto MTM2017-86875-C3-1-R AEI/ FEDER EU, la cátedra AXA-ICMAT, el proyecto EU's Horizon 2020 número 740920 CYBECO (Supporting Cyberinsurance from a Behavioural Choice Perspective) y el Statistical and Applied Mathematical Sciences Institute (SAMSI).

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# Notation

## General comments

Most part of this PhD thesis is devoted to model confrontations between two agents: a Defender (she) and an Attacker (he). The Defender is usually denoted as $D$. Her actions belong to the set $\mathcal{D}$. Individual actions are generally denoted as $d$. Similarly, the Attacker is denoted as $A$, his decision set is $\mathcal{A}$, and his actions are denoted as $a$.

Unless noted, the gradient will be denoted as $\mathrm{d}_x$; the partial derivative as $\partial_x$. Similarly, second partial derivatives will be denoted as $\partial_x^2$ and $\partial_x\partial_y$. We shall use this notation indistinctly for the unidimensional and multidimensional cases. For instance, if $f(x, y)$ is a scalar function, $x$ is a $p$-dimensional vector and $y$ is a $q$-dimensional vector, then $\partial_x^2 f(x, y)$, the Hessian matrix, is the $p \times p$ matrix whose $(i, j)$ entry is $\partial_{x_i}\partial_{x_j} f(x, y)$, where $x_i$ is the $i$-th component of the vector $x$. Similarly, $\partial_x\partial_y f(x, y)$ is a $p \times q$ matrix whose $i, j$ entry is $\partial_{x_i}\partial_{y_j} f(x, y)$.

## Abbreviations

**ABC**: Approximate Bayesian computation
**AC**: Adversarial classification
**ACRA**: AC based on adversarial risk analysis
**AD**: Automatic Differentiation
**ADS**: Autonomous driving system
**AI**: Artificial intelligence
**AML**: Adversarial machine learning
**APP**: Adversarial prediction problem
**APS**: Augmented probability simulation
**AR**: Autoregressive
**ARA**: Adversarial risk analysis
**ARIMA**: Autoregressive integrated moving average
**AT**: Adversarial training
**BAID**: Bi-agent influence diagram
**BGR**: Brooks-Gelman-Rubin
**BNE**: Bayes-Nash equilibrium
**CART**: Classification and regression trees
**CK**: Common knowledge
**CNN**: Convolutional neural network
**DDoS**: Distributed denial-of-service
**DLM**: Dynamic linear model
**DM**: Decision maker
**DNN**: Deep neural network
**ES**: Exponential smoothing
**FNR**: False negative rate
**FPR**: False positive rate
**GWI**: Good Word Insertion
**GESD**: Generalized extreme Student deviation
**ICD**: Internet connected device
**ICT**: Information and communication technologies
**ID**: Influence diagram
**IoT**: Internet of things

**KKT**: Karush-Kuhn-Tucker
**LSTM**: Long-short term memory
**MAE**: Mean absolute error
**MAPE**: Mean absolute percentage error
**ML**: Machine learning
**MC**: Monte Carlo
**MCMC**: Markov chain Monte Carlo
**MH**: Metropolis-Hastings
**MR**: Multinomial regression
**MSE**: Mean squared error
**NB**: Naive Bayes
**NE**: Nash equilibrium
**NN**: Neural network
**PDE**: Partial Differential Equation
**PGD**: Projected gradient descent
**RL**: Reinforcement learning
**RSME**: Root mean squared error
**SG-MCMC**: Stochastic gradient Markov chain Monte Carlo
**SLLN**: Strong law of large numbers
**TMDP**: Threatened Markov decision process
**VAE**: Variational autoencoder

# Abstract

Machine learning (ML) applications have experienced an unprecedented growth over the last two decades. However, the ever increasing adoption of ML methodologies has revealed important security issues. Among these, vulnerabilities to adversarial examples, data instances targeted at fooling ML algorithms, are especially important. Examples abound. For instance, it is relatively easy to fool a spam detector simply misspelling spam words. Obfuscation of malware code can make it seem legitimate. Simply adding stickers to a stop sign could make an autonomous vehicle classify it as a merge sign. Consequences could be catastrophic.

Indeed, ML is designed to work in stationary and benign environments. However, in certain scenarios, the presence of adversaries that actively manipulate input data to fool ML systems to attain benefits break such stationarity requirements. Training and operation conditions are not identical anymore. This creates a whole new class of security vulnerabilities that ML systems may face and a new desirable property: adversarial robustness. If we are to trust operations based on ML outputs, it becomes essential that learning systems are robust to such adversarial manipulations.

This thesis explores the topic of secure ML focusing on security against intentional adversarial threats. Since first observations by Dalvi et al. 2004, who noted that it was relatively easy to fool linear classifiers in spam detection systems simply by adding "good" words to emails to make them be classified as legitimate, research in security issues of ML has grown exponentially. This research is not only targeted at exploring new vulnerabilities of learning systems, but also aims at providing efficient defenses. Two types of defense methods have been proposed: *reactive* ones aimed at mitigating or eliminating the effects of an eventual attack, whereas *proactive* ones aimed at preventing the attack execution.

In this thesis we propose novel defense mechanisms to adversarial attacks and study their relevant algorithmic aspects. The first objective of the thesis (O1), accomplished in Chapter 2, is that of developing a novel reactive defense for time series security problems. We have worked on advanced detection of threats in the domain of predictive network monitoring, so as to mitigate the effect of potential attacks. In this domain it is common to deal with a huge number of high frequency time series. Thus, our predictive system, apart from being accurate, needs to be scalable, automatic and versatile. We have provided a framework to identify safety and security issues within a large number of internet connected devices that fulfills these conditions.

The second objective (O2), attained in Chapter 3, is that of providing a proactive

defense in classification problems. Among proposed proactive defenses in secure ML, security-by-design approaches constitute an important subclass. These entail modeling adversarial actions explicitly when designing the learning system. Stemming from the pioneering work by Dalvi et al. 2004, most approaches model the confrontation between adversary and ML system within the framework of game theory with underlying common knowledge hypothesis, unrealistic in most security settings. In Chapter 3, we provide a general, Bayesian probabilistic framework based on adversarial risk analysis (ARA) for modeling such confrontation. Our approach mitigates standard common knowledge assumptions by modeling explicitly, not only the presence of an adversary, but also our uncertainty about his elements.

Finding equilibria, or seeking ARA solutions, of typical games used in secure ML is very challenging from a computational perspective, as we need to face a new paradigm: while in classical game theory intervening agents were humans whose decisions are generally discrete and low dimensional, in secure ML decisions are made by algorithms and are usually continuous and high dimensional, e.g. choosing the weights of a neural network. As a result, scalable numerical algorithms to solve these type of games are required. Chapters 4 and 5 provide gradient based and simulation based solution approaches, respectively, and study their scalability with the dimension of decision sets, thus accomplishing the two last objectives of this thesis (O3 and O4).

For the sake of reproducibility, the code used in the experiment of this thesis is open-sourced and available at `https://github.com/roinaveiro`. In addition, the following papers, with the most significant results of this PhD thesis, have already been published:

> Naveiro, R.; Rodríguez, S. & Ríos Insua, D. (2019). Large scale automated forecasting for network safety and security monitoring. *Applied Stochastic Models in Business and Industry*, 35(3), 431–447, `https://doi.org/10.1002/asmb.2436`.
>
> Naveiro, R.; Redondo, A.; Ríos Insua, D. & Ruggeri, F. (2019). Adversarial classification: An adversarial risk analysis approach. *International Journal of Approximate Reasoning*, 113, 133–148, `https://doi.org/10.1016/j.ijar.2019.07.003`.
>
> Naveiro, R. & Ríos Insua, D. (2019). Gradient Methods for Solving Stackelberg Games.*Algorithmic Decision Theory. ADT 2019. Lecture Notes in Computer Science*, 11834, Springer, Cham, `https://doi.org/10.1007/978-3-030-31489-7_9`.
>
> Banks, D.; Gallego, V.; Naveiro, R. & Ríos Insua, D. (2020). Adversarial Risk Analysis (Overview).*Wiley Interdisciplinary Reviews: Computational Statistics*, Wiley Online Library.

While the next still await to be published:

Ekin, T.; Naveiro, R.; Torres Barrán, A. & Ríos Insua, D. (2019). Augmented Probability Simulation Methods for Non-cooperative Games. *arXiv preprint arXiv:1910.04574*.

Ríos Insua, D.; Naveiro, R.; Gallego, V. & Poulos, J. (2020). Adversarial Machine Learning: Perspectives from Adversarial Risk Analysis. *arXiv preprint arXiv:2003.03546*.

Gallego, V.; Naveiro, R.; Redondo, A.; Ruggeri, F. & Ríos Insua, D. (2020). Protecting Classifiers From Attacks. A Bayesian Approach. *arXiv preprint arXiv:2004.08705*.

Gallego, V.; Naveiro, R.; & Ríos Insua, D. (2020). Perspectives on Adversarial Classification.

Additional related work to that of this PhD thesis has been conducted in

Gallego, V.; Naveiro, R. & Ríos Insua, D. (2019). Reinforcement Learning under Threats.*AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 9939-9940, `https://doi.org/10.1609/aaai.v33i01.33019939`.

Besides, a number of talks have also been derived from its contents:

Naveiro, R. (June, 2017). Monitoring for anomalous behaviour in massive traffic time series. Oral presentation at the *Bayesian Inference in Stochastic Processes Workshop (BISP10)*, Bocconi University, Milano, Italy.

Naveiro, R. (November, 2017). Adversarial Classification: An Adversarial Risk Analysis approach. Oral presentation at the *1st Spanish Young Statisticians and Operational Researchers Meeting (SYSORM)*, University of Granada, Granada, Spain.

Naveiro, R. (May, 2018). Adversarial Classification: An Adversarial Risk Analysis approach. Oral presentation at the *XXXVII National Congress of Statistics and Operational Research*, Oviedo, Spain.

Naveiro, R. (July, 2019). Augmented Probability Simulation Methods for Non-cooperative Games. Poster presentation at the *Second Conference on Risk Analysis in the Digital Era*, University at Buffalo, Buffalo, USA.

Naveiro, R. (August, 2019). Augmented Probability Simulation Methods for Non-cooperative Games. Poster presentation at the *Games, Decisions, Risk and Reliability Opening Workshop*, NC State University, Raleigh, USA.

Naveiro, R. (August, 2019). Gradient Methods for Solving Stackelberg Games. Poster presentation at the *Deep Learning Opening Workshop*, Duke University, Durham, USA.

Naveiro, R. (October, 2019). Gradient Methods for Solving Stackelberg Games. Oral presentation at the *Algorithmic Decision Theory Conference*, Duke University, Durham, USA.

Naveiro, R. (November, 2019). Security Games in the New Paradigm: Solution Techniques. Invited speaker at the George Washington University, Washington D.C., USA.

# Resumen

Las aplicaciones del aprendizaje automático o *machine learning* (ML) han experimentado un crecimiento sin precedentes en las últimas dos décadas. Sin embargo, la adopción cada vez mayor de metodologías de ML ha revelado importantes problemas de seguridad. Entre estos, destacan las vulnerabilidades a ejemplos adversarios, es decir; instancias de datos destinadas a engañar a los algoritmos de ML. Los ejemplos abundan: es relativamente fácil engañar a un detector de spam simplemente escribiendo mal algunas palabras características de los correos basura. La ofuscación de código malicioso (*malware*) puede hacer que parezca legítimo. Agregando unos parches a una señal de stop, se podría provocar que un vehículo autónomo la reconociese como una señal de dirección obligatoria. Cómo puede imaginar el lector, las consecuencias de estas vulnerabilidades pueden llegar a ser catastróficas.

Y es que el *machine learning* está diseñado para trabajar en entornos estacionarios y benignos. Sin embargo, en ciertos escenarios, la presencia de adversarios que manipulan activamente los datos de entrada para engañar a los sistemas de ML (logrando así beneficios), rompen tales requisitos de estacionariedad. Las condiciones de entrenamiento y operación de los algoritmos ya no son idénticas, quebrándose una de las hipótesis fundamentales del ML. Esto crea una clase completamente nueva de vulnerabilidades que los sistemas basados en el aprendizaje automático deben enfrentar y una nueva propiedad deseable: la robustez adversaria. Si debemos confiar en las operaciones basadas en resultados del ML, es esencial que los sistemas de aprendizaje sean robustos a tales manipulaciones adversarias.

Esta tesis explora cómo garantizar un aprendizaje automático seguro, enfocándose en la seguridad contra amenazas adversarias intencionales. Desde las primeras observaciones por parte de Dalvi et al. 2004, quiénes probaron que era relativamente sencillo engañar a los clasificadores lineales de los sistemas de detección de spam, simplemente agregando palabras "buenas" a los correos basura para que fuesen clasificados como legítimos; la investigación en la seguridad del ML ha crecido exponencialmente. Esta no solo se centran en explorar nuevas vulnerabilidades de los sistemas de aprendizaje automático, sino que también tiene como objetivo proporcionar defensas eficientes ante dichas vulnerabilidades. En la literatura, se han propuesto dos clases de métodos de defensa: los *reactivos*, que pretenden mitigar o eliminar los efectos de un posible ataque, y los *proactivos*, cuyo objetivo es evitar la ejecución del ataque.

En esta tesis proponemos nuevos mecanismos de defensa ante ataques adversarios y estudiamos aspectos algorítmicos relevantes de los mismos. Nuestro primer objetivo

(O1), abordado en el Capítulo 2, es el de desarrollar una nueva defensa reactiva aplicable a problemas de seguridad que utilicen el análisis de series temporales. En concreto, hemos trabajado en la detección avanzada de amenazas en el dominio de la monitorización predictiva de grandes redes de dispositivos, para así mitigar el efecto de posibles ataques. En este dominio, es común tener que tratar con una gran cantidad de series temporales de alta frecuencia. Por lo tanto, nuestro sistema predictivo, además de ser preciso, necesita ser escalable, automático y versátil. Hemos proporcionado un marco para garantizar la seguridad en grandes redes de dispositivos conectados a Internet que cumple con las características mencionadas.

El segundo objetivo (O2), abordado en el Capítulo 3, es el de proporcionar una nueva defensa proactiva en problemas de clasificación estadística. Entre las defensas propuestas en la literatura, las consistentes en la *seguridad por diseño* constituyen una subclase importante. Estas requieren la modelización explícita de las acciones de posibles adversarios en el diseño del sistema de aprendizaje. Partiendo del trabajo pionero de Dalvi et al. 2004, la mayoría de los enfoques sucesivos modelizan esta confrontación entre el adversario y el sistema de ML dentro del marco de la teoría de juegos con la consiguiente hipótesis de conocimiento común, poco realista en la mayoría de las aplicaciones en seguridad. En el Capítulo 3, proporcionamos un marco probabilístico bayesiano basado en el análisis de riesgos adversarios (ARA) para estudiar tal confrontación. Nuestro enfoque mitiga los efectos de la hipótesis de conocimiento común al modelizar explícitamente, no solo la presencia de adversarios, sino también nuestra incertidumbre acerca de sus elementos.

Encontrar equilibrios, o buscar soluciones de ARA, de los juegos típicamente utilizados en el campo de la seguridad del ML es muy costoso desde una perspectiva computacional, ya que necesitamos enfrentarnos a un nuevo paradigma: mientras que en la teoría de juegos clásica, los jugadores eran humanos cuyas decisiones son generalmente discretas y de baja dimensionalidad, en el campo del ML seguro, las decisiones las toman algoritmos y generalmente son contínuas y de alta dimensión, por ejemplo, elegir los pesos de una red neuronal. En consecuencia, se requieren algoritmos numéricos escalables para resolver este tipo de juegos. Los capítulos 4 y 5 proporcionan soluciones basadas en el gradiente y en simulación, respectivamente, y estudian su escalabilidad con la dimensión de los conjuntos de decisiones, cumpliendo así con los dos últimos objetivos de esta tesis doctoral (O3 y O4).

En aras de la reproducibilidad, el código utilizado en los experimentos de la presente tesis doctoral es de libre acceso y está disponible en `https://github.com/roinaveiro`. Además, los siguientes artículos, con los resultados más significativos de esta tesis, ya han sido publicados:

> Naveiro, R.; Rodríguez, S. & Ríos Insua, D. (2019). Large scale automated forecasting for network safety and security monitoring. *Applied Stochastic Models in Business and Industry*, 35(3), 431–447, `https://doi.org/10.1002/asmb.2436`.
>
> Naveiro, R.; Redondo, A.; Ríos Insua, D. & Ruggeri, F. (2019). Adversarial classification: An adversarial risk analysis approach. *International*

*Journal of Approximate Reasoning*, 113, 133–148, https://doi.org/10.1016/j.ijar.2019.07.003.

Naveiro, R. & Ríos Insua, D. (2019). Gradient Methods for Solving Stackelberg Games.*Algorithmic Decision Theory. ADT 2019. Lecture Notes in Computer Science*, 11834, Springer, Cham, https://doi.org/10.1007/978-3-030-31489-7_9.

Banks, D.; Gallego, V.; Naveiro, R. & Ríos Insua, D. (2020). Adversarial Risk Analysis (Overview). *Wiley Interdisciplinary Reviews: Computational Statistics*, Wiley Online Library.

Mientras que los próximos aún esperan a ser publicados:

Ekin, T.; Naveiro, R.; Torres Barrán, A. & Ríos Insua, D. (2019). Augmented Probability Simulation Methods for Non-cooperative Games. *arXiv preprint arXiv:1910.04574*.

Ríos Insua, D.; Naveiro, R.; Gallego, V. & Poulos, J. (2020). Adversarial Machine Learning: Perspectives from Adversarial Risk Analysis. *arXiv preprint arXiv:2003.03546*.

Gallego, V.; Naveiro, R.; Redondo, A.; Ruggeri, F. & Ríos Insua, D. (2020). Protecting Classifiers From Attacks. A Bayesian Approach. *arXiv preprint arXiv:2004.08705*.

Gallego, V.; Naveiro, R.; & Ríos Insua, D. (2020). Perspectives on Adversarial Classification.

Trabajo relacionado con el de esta tesis doctoral ha sido realizado en:

Gallego, V.; Naveiro, R. & Ríos Insua, D. (2019). Reinforcement Learning under Threats.*AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 9939-9940, https://doi.org/10.1609/aaai.v33i01.33019939.

Además, una serie de presentaciones en conferencias internacionales también se han derivado de su contenido:

Naveiro, R. (June, 2017). Monitoring for anomalous behaviour in massive traffic time series. Oral presentation at the *Bayesian Inference in Stochastic Processes Workshop (BISP10)*, Bocconi University, Milano, Italy.

Naveiro, R. (November, 2017). Adversarial Classification: An Adversarial Risk Analysis approach. Oral presentation at the *1st Spanish Young Statisticians and Operational Researchers Meeting (SYSORM)*, University of Granada, Granada, Spain.

Naveiro, R. (May, 2018). Adversarial Classification: An Adversarial Risk Analysis approach. Oral presentation at the *XXXVII National Congress of Statistics and Operational Research*, Oviedo, Spain.

Naveiro, R. (July, 2019). Augmented Probability Simulation Methods for Non-cooperative Games. Poster presentation at the *Second Conference on Risk Analysis in the Digital Era*, University at Buffalo, Buffalo, USA.

Naveiro, R. (August, 2019). Augmented Probability Simulation Methods for Non-cooperative Games. Poster presentation at the *Games, Decisions, Risk and Reliability Opening Workshop*, NC State University, Raleigh, USA.

Naveiro, R. (August, 2019). Gradient Methods for Solving Stackelberg Games. Poster presentation at the *Deep Learning Opening Workshop*, Duke University, Durham, USA.

Naveiro, R. (October, 2019). Gradient Methods for Solving Stackelberg Games. Oral presentation at the *Algorithmic Decision Theory Conference*, Duke University, Durham, USA.

Naveiro, R. (November, 2019). Security Games in the New Paradigm: Solution Techniques. Invited speaker at the George Washington University, Washington D.C., USA.

# Chapter 1

# Introduction

## 1.1 Motivation

Over the last decade, an increasing number of processes are being automated through machine learning (ML) algorithms (Breiman 2001). Applications abound in areas such as bioinformatics, Ghosh and Parai (2008); spam detection, Goodman and Heckerman (2004); credit scoring, Hand and Henley (1997); computer vision, Chen (2015); and genomics, Mallick et al. (2005); to mention but a few. As a consequence, it becomes essential that ML algorithms are robust and reliable if we are to trust operations based on their output.

State-of-the-art ML algorithms perform extraordinarily well on standard data, but have recently been shown to be vulnerable to adversarial examples, data instances targeted at fooling those algorithms (Goodfellow et al. 2015a). The presence of adaptive adversaries has been pointed out in areas such as spam detection (Zeager et al. 2017); fraud detection (Kołcz and Teo 2009); and computer vision (Goodfellow et al. 2015a). In those contexts, algorithms should acknowledge the presence of possible adversaries to defend against their data manipulations. Comiter (2019) provides a review from the policy perspective showing how many artificial intelligence (AI) societal systems, including content filters, military systems, law enforcement systems and autonomous driving systems (ADS), are susceptible and vulnerable to attacks. As motivating examples, we next consider attacks to computer vision and spam detection systems.

**Attacking computer vision algorithms** Vision algorithms are at the core of many AI systems such as ADS (Bojarski et al. 2016) and (McAllister et al. 2017). The simplest and most notorious attack examples to such algorithms consist of modifications of images in such a way that the alteration becomes imperceptible to the human eye, yet drives a model trained on millions of images to misclassify the modified ones, with potentially relevant security consequences.

With a relatively simple convolutional neural network (CNN) model, we are able to accurately predict 99% of the handwritten digits in the MNIST data set (LeCun et al. 1998). However, if we attack those data with the fast gradient sign method in

Szegedy et al. (2014), its accuracy is reduced to 62%. Figure 1.1 provides an example of an original MNIST image and a perturbed one. Although to our eyes both images look like a 2, our CNN classifier rightly identifies a 2 in the first case (Fig. 1.1a), whereas it suggests a 7 after the perturbation (Fig. 1.1b).

(a) Original image (b) Perturbed image

**Figure 1.1**: An original input and its attacked version.

This type of attacks are easily built through low cost computational methods. However, they require the attacker to have precise knowledge about the architecture and weights of the corresponding predictive model. This is debatable in security settings, a main driver of this thesis. △

**Attacking spam detection algorithms** Consider spam detection problems, an example of content filter systems. We employ the utility sensitive naive Bayes (NB) classifier, a standard spam detection approach (Song et al. 2009), studying its degraded performance under the 1 Good Word Insertion attacks in Naveiro et al. (2019a). In this case, the adversary just attacks spam emails adding to them a good word, that is, a word which is common in legit emails but not in spam ones.

We use the Spambase Data Set from the UCI ML repository (Lichman 2013). Accuracy and false positive and negative rates (FPR and FNR respectively) are reported in Table 1.1, estimated via repeated hold-out validation over 100 repetitions (Kim 2009). We provide as well the standard deviation of each metric, also estimated through repeated hold-out validation.

| | **Accuracy** | **FPR** | **FNR** |
|---|---|---|---|
| **NB-Plain** | $0.886 \pm 0.009$ | $0.680 \pm 0.100$ | $0.19 \pm 0.02$ |
| **NB-Tainted** | $0.761 \pm 0.101$ | $0.680 \pm 0.100$ | $0.500 \pm 0.250$ |

**Table 1.1**: Performance of utility sensitive NB in clean and attacked data.

NB-Plain and NB-Tainted refer to results of NB on clean and attacked data, respectively. Observe how the presence of an adversary degrades NB accuracy: FPR coincides for NB-Plain and NB-Tainted as the adversary is not modifying innocent instances. However, false negatives undermine NB-Tainted performance, as the classifier is not able to identify a large proportion of attacked spam emails.

△

Both examples show how the performance of ML based systems may degrade under attacks. This suggests the need for a framework that guarantees robustness

of ML against adversarial manipulations in a principled way, as pointed out in Fan et al. (2020). During the last decade, research towards the achievement of such framework for secure machine learning has gained a lot of interest, constituting a new research area called adversarial machine learning (AML). Within this field, the pipeline typically followed to improve security of ML systems, proposed in Biggio and Roli (2018), consists of three key steps: modeling threats, taking into account the adversary's goals, knowledge and capabilities; simulating them to better prepare against attacks; and, finally, defending ML systems from such attacks. We describe each of them in more detail.

**Modelling threats.** This activity is critical to ensure ML security in adversarial environments. It is obvious that any algorithm could be fooled if adversarial data modifications are not somehow restricted. Consider for instance an extreme case in which an instance in a binary classification problem is modified so that it is indistinguishable from an instance of the other class. Clearly, the algorithm would misclassify such instance. However, the adversary is probably not interested in making such data modifications. Thus, appropriate adversary modeling is key in AML. This is a problem specific task.

In general, we should assess three attacker features. First, we should consider his *goals*, which vary depending on the setting, ranging from money to causing fatalities, going through damaging reputation (Couce-Vieira et al. 2019). Prior to deploying an ML system, it is crucial to guarantee its robustness against attackers with the most common goals. For instance, in fraud detection the attacker usually obfuscates fraudulent transactions to make the system classify them as legitimate ones in search of an economic benefit: a fraud detection system should be robust against such attacks. In general, these are classified following two criteria regarding their goals: *violation type* and *attack specificity*. For the first criterion, we distinguish between *integrity violations*, aimed at moving the prediction of particular instances towards the attacker's target, e.g. have malicious samples misclassified as legitimate; *availability violations*, aimed at increasing the predictive error to make the system unusable; and *privacy violations* (a.k.a, exploratory attacks) to gain information about the ML system. In relation with the second one, we distinguish between *targeted*, which address just a few, even one, defenders, and *indiscriminate* attacks, affecting many defenders in a random manner.

Next, we consider that at the time of attacking, the adversary could have *knowledge* about different aspects of the ML system such as the training data used, or the features. Thus, we classify adversarial threats depending on which aspects is the attacker assumed to have knowledge about. At one end of the spectrum, we find *white box* or *perfect knowledge* attacks: the adversary knows every aspect of the ML system. This is almost never the case in actual scenarios, except perhaps for insiders. Yet they could be useful in sequential settings where the ML system moves first, training an algorithm to find its specific parameters. The adversary, who moves afterwards, has some time to observe the behaviour of the system and learn about it.[1] At the other

[1]However, although the adversary may have some knowledge, assuming that this is perfect is

end, *black box* or *zero knowledge* attacks assume that the adversary has capabilities to query the system but does not have any information about the data, the feature space or the particular algorithms used. This is the most reasonable assumption in which attacking and defending decisions are eventually made simultaneously. In between attacks are called *gray box* or *limited knowledge*. This is the most common type of attacks in security settings, especially when attacking and defending decisions are made sequentially but there is private information that the intervening agents are not willing to share.

Finally, we classify the attacks depending on the *capabilities* of the adversary to influence on data. In some cases, he may obfuscate training data to induce errors during operation, called *poisoning attacks*. On the other hand, *evasion attacks* have no influence on training data, but perform modifications during operation, for instance when trying to evade a detection system. These data alteration or crafting activities are the typical in AML and we designate them as coming from a *data-fiddler*. But there could be attackers capable of changing the underlying structure of the problem, affecting process parameters, called *structural attackers*. Moreover, some adversaries could be making decisions in parallel to those of the defender with the agents' losses depending on both decisions, which we term *parallel attackers*.[2]

**Simulating attacks.** The standard approach, Biggio and Roli (2018), formalizes poisoning and evasion attacks in terms of constrained optimization problems with different assumptions about the adversary's knowledge, goals and capabilities. In general, the objective function in such problems assesses attack effectiveness, taking into account the attacker's goals and knowledge. The constraints frame assumptions such as the adversary wanting to avoid detection or having a maximum attacking budget.[3]

**Protecting learning algorithms.** Two types of defence methods have been proposed. *Reactive defences* aim to mitigate or eliminate the effects of an eventual attack. They include timely detection of attacks, e.g. Naveiro et al. (2019b); frequent retraining of learning algorithms; or verification of algorithmic decisions by experts. *Proactive defences* aim to prevent attack execution. They can entail security-by-design approaches such as explicitly accounting for adversarial manipulations, e.g., Naveiro et al. (2019a), or developing provably secure algorithms against specific perturbations, e.g., Gowal et al. (2018); or security-by-obscurity techniques such as randomization of the algorithm response, or gradient obfuscation to make attacks less likely to succeed (Athalye et al. 2018).

---

not realistic and has been criticized, even in the pioneering Dalvi et al. (2004).

[2]Some attackers could combine the three capabilities in certain scenarios. For example, in a cybersecurity problem an attacker might add spam modifying its proportion (structural); alter some spam messages (data-fiddler); and, in addition, undertake his own business decisions (parallel).

[3]A more natural and general formulation of the attacker's problem is through a statistical decision theoretic perspective (French and Rios Insua 2000), see Section 1.4.3.

In this PhD dissertation, we provide contributions to the security of Machine Learning by proposing novel reactive and proactive defences. Before specifying the particular research objectives, we provide a generic literature review of Adversarial Machine Learning, Section 1.2, that is relevant for framing most of the work conducted in this PhD. This revision suggests the specific topics covered in this thesis, which are introduced in Sections 1.3, 1.4 and 1.5.

## 1.2 Adversarial Machine Learning: a review

In this section we present, following a historical perspective, key results and concepts in AML. Further perspectives may be found in recent reviews by Vorobeychik and Kantarcioglu (2018), Joseph et al. (2019), Biggio and Roli (2018), Dasgupta and Collins (2019) and Zhou et al. (2019).

### 1.2.1 Adversarial classification

Classification is one of the most widely used instances of supervised learning (Bishop 2006). In recent years, the field has experienced an enormous growth becoming a major research area in statistics and machine learning (Efron and Hastie 2016). Most efforts in classification have focused on obtaining more accurate algorithms which, however, largely ignore a relevant issue in many application areas: the presence of adversaries who can actively manipulate data to fool the classifier so as to attain a benefit. As a motivating example consider the case of fraud detection. As machine learning algorithms are incorporated to such detection task, fraudsters begin to learn how to evade them. For instance, they could find out that making a huge transaction increases the probability of being detected and start issuing smaller transactions more frequently rather than a single big one. Thus, in contexts such as fraud detection, algorithms should take into account possible modifications on the behavior of adversaries so as to be robust against adversarial data manipulations.

Dalvi et al. (2004) introduced adversarial classification (AC), a pioneering approach to enhance classification algorithms when an adversary is present. They view AC as a game between a classifier, also referred to as defender ($D$, she), and an adversary, ($A$, he). The classifier aims at finding an optimal classification strategy against $A$'s optimal attacks. Computing Nash equilibria (NE) in such general games quickly becomes very complex. Thus, they propose a forward myopic version in which $D$ first assumes that the data is untainted, computing her optimal classifier; then, $A$ deploys his optimal attack against it; subsequently, $D$ implements the best response classifier against such attack, and so on. This approach assumes common knowledge (CK), i.e. all parameters of both players are known to each other. Although standard in game theory, this assumption is actually unrealistic in the security settings typical of AML.

Stemming from this work, there has been an important literature in AC, reviewed in Biggio et al. (2014) or Li and Vorobeychik (2014). Subsequent approaches have focused on analyzing attacks over classification algorithms and assessing their

robustness against such attacks. To that end, some assumptions about the adversary are made. For instance, Lowd and Meek (2005) consider that the adversary is able to send membership queries to the classifier, the entire feature space being known to issue optimal attacks; then, they prove the vulnerability of linear classifiers against adversaries. Similarly, Zhou et al. (2012) consider that the adversary seeks to push his malicious instances into innocuous ones, assuming that the adversary can estimate such instances.

A few methods have been proposed to robustify classification algorithms in adversarial environments. Most of them have focused on application-specific domains, as Kołcz and Teo (2009) on spam detection. Vorobeychik and Li (2014) study the impact of randomization schemes over different classifiers against adversarial attacks proposing an optimal randomization scheme as best defense. Other approaches have focused on improving the game theoretic model in Dalvi et al. (2004) but, to our knowledge, none has been able to overcome the unrealistic common knowledge assumptions, as may be seen in recent reviews by Biggio and Roli (2018) and Zhou et al. (2019), who have also pointed out the importance of this issue. As an example, Kantarcıoğlu et al. (2011) use a Stackelberg game in which both players know each other payoff functions. Only Großhans et al. (2013) have attempted to relax common knowledge assumptions in adversarial regression settings, reformulating the corresponding problem as a Bayesian game.

### 1.2.2 Adversarial prediction

An important source of AML cases are adversarial prediction problems (APPs), Brückner and Scheffer (2011).They focus on building predictive models where an adversary exercises some control over the data generation process, jeopardising standard prediction techniques. APPs model the interaction between the predictor and the adversary as a two agent game, a defender that aims at learning a parametric predictive model and an adversary trying to transform the distribution governing data at training time. The agents' costs depend on both the predictive model and the adversarial data transformation. Both agents aim at optimizing their operational costs, which is a function of their expected losses under the perturbed data distribution. As this distribution is unknown, the agents actually optimize their regularized empirical costs, based on the training data. The specific optimization problem depends on the case considered.

First, in *Stackelberg prediction games*, Brückner and Scheffer (2011) assume full information of the attacker about the predictive model used by the defender who, in addition, is assumed to have perfect information about the adversary's costs and action space. $D$ acts first choosing her parameter; then, $A$, after observing this decision, chooses the optimal data transformation. Finding NE in these games requires solving a bi-level optimization problem, optimizing the defender's cost function subject to the adversary optimizing his, after observing the defender's choice. As nested optimization problems are intrinsically hard, the authors restrict to simple classes where analytical solutions can be found. On the other hand, in *Nash*

*prediction games* (Brückner et al. 2012) both agents act simultaneously. The main concern is then seeking for NE. The authors provide conditions for their existence and uniqueness in specific classes of games.

### 1.2.3 Adversarial unsupervised learning

Much less AML work is available in relation with unsupervised learning. A relevant proposal is Kos et al. (2018) who describe adversarial attacks to generative models such as variational autoencoders (VAEs) or generative adversarial networks used in density estimation. Their focus is on slightly perturbing the input to the models so that the reconstructed output is completely different from the original input.

To the best of our knowledge, Biggio et al. (2013) first studied clustering under adversarial disturbances. They suggest a framework to create attacks during training that significantly alter cluster assignments, as well as an obfuscation attack that slightly perturbs an input to be clustered in a predefined assignment, showing that single-link hierarchical clustering is sensitive to these attacks.

Lastly, adversarial attacks on autoregressive (AR) models have started to attract interest. Alfeld et al. (2016) describe an attacker manipulating the inputs to drive the latent space of a linear AR model towards a region of interest. Papernot et al. (2016) propose adversarial perturbations over recurrent neural networks.

### 1.2.4 Adversarial reinforcement learning

The prevailing solution approach in reinforcement learning (RL) is $Q$-learning (Sutton and Barto 2018). Deep RL has faced an incredible growth (Silver et al. 2017); however, the corresponding systems may be targets of attacks and robust methods are needed (Huang et al. 2017).

An AML related field of interest is multi-agent RL (Buşoniu et al. 2010). Single-agent RL methods fail in multi-agent settings, as they do not take into account the non-stationarity due to the other agents' actions: $Q$-learning may lead to suboptimal results. Thus, we must reason about and forecast the adversaries' behaviour. Several methods have been proposed in the AI literature (Albrecht and Stone 2018). A dominant approach draws on fictitious play (Brown 1951) and consists of assessing the other agents computing their frequencies of choosing various actions. Using explicit representations of the other agents' beliefs about their opponents could lead to an infinite hierarchy of decision making problems, as explained in Gallego et al. (2019b).

The application of these tools to $Q$-learning in multi-agent settings remains largely unexplored. Relevant extensions have rather focused on Markov games. Three well-known solutions are minimax-$Q$ learning (Littman 1994), which solves at each iteration a minimax problem; Nash-$Q$ learning (Hu and Wellman 2003), which generalizes to the non-zero sum case; or friend-or-foe-$Q$ learning (Littman 2001), in which the agent knows in advance whether her opponent is an adversary or a collaborator.

### 1.2.5 Adversarial examples

One of the most influential concepts triggering the recent interest in AML are *adversarial examples*. They were introduced by Szegedy et al. (2014) within neural network (NN) models, as perturbed data instances obtained through solving certain optimization problem. NNs are highly sensitive to such examples (Goodfellow et al. 2015a).

The usual framework for robustifying models against these examples is adversarial training (AT) (Madry et al. 2018), based on solving a bi-level optimization problem whose objective function is the empirical risk of a model under worst case data perturbations. AT approximates the inner optimization through a projected gradient descent (PGD) algorithm, ensuring that the perturbed input falls within a tolerable boundary.[4] The complexity of this attack depends on the chosen norm. However, recent pointers urge modellers to depart from using norm based approaches (Carlini et al. 2019) and develop more realistic attack models, as in Brown et al. (2017) adversarial patches.

### 1.2.6 Comments

As a result of the previous revision there are three topics of interest that will be covered in this thesis.

1. While a lot of research has been undertaken in security aspects of supervised learning, security of time series based systems has been largely ignored. Thus, the first topic of interest will be to deal with reactive defences in time series security problems.

2. Most of the research in adversarial supervised learning has been framed within a standard game theory approach. However, this entails strong common knowledge assumptions which are hard to maintain in the security contexts typical of AML. In this PhD thesis, we propose a formal Bayesian decision theoretic approach to solve AML problems, adopting an adversarial risk analysis (ARA) perspective, Banks et al. (2015); to model the confrontation between attackers and defenders mitigating questionable CK assumptions.

3. Given the importance of game theoretic approaches in AML, we devote the last part of the thesis to develop efficient algorithmic approaches to solve typical games appearing in AML.

These topics are introduced in the following sections.

[4] It is possible to frame the inner optimization problem as a *mixed integer linear program* and use general purpose optimizers to search for adversarial examples (Kolter and Madry 2018). Note though that for moderate to large networks in mainstream tasks, exact optimization is still computationally intractable.

## 1.3 Reactive Defences in Time Series Security Problems

As outlined in Mortenson et al. (2015), leveraging big data and real time analytics constitute two main current research venues. Information and communication technologies (ICT) have experienced an exponential growth in the last few decades and most human activities, businesses and devices strongly depend on them, Dimelis and Papaioannou (2011). With the advent of the Internet of things (IoT), this interrelation will become even more evident and change dramatically the way in which different components of business and service systems interact. In parallel, risks concerning the security of ICT systems are also growing, as pointed out e.g. by The Geneva Association (2016). Such cyber risks can be of natural origin or man-made, the latter potentially originating from human failure or intentional threats, as in cyber crime. These actually constitute a current major global trend, as reflected e.g. in the Global Risks Map, World Economic Forum (2020). As an example, MacAfee (2014) catalogs around 70 new potential cyber threats per minute, and estimates the annual cost to the global economy from cyber crime to be above 400 billion USD. This impact highlights the need for developing solid cybersecurity risk management frameworks and the central role that these will play in business and industrial security in the near future. In addition, the increasing significance and availability of data in every business-related activity emphasizes data-driven approaches to improve cybersecurity decision making. As an example, Sedgewick (2014) provides a set of standards and guidelines supporting such frameworks, covering key cybersecurity activities. Two of them refer to continuous safety monitoring of Internet connected devices (ICDs) and anomaly detection, key reactive defences in cybersecurity. This has led to the development of tools that periodically collect high frequency information from the ICDs of an organisation to support their monitoring. Given the increasing relevance of ICT, we may need to face organisations with several hundred thousands of such devices from which we obtain tens of variables every few minutes. This poses tremendous challenges in processing such enormous amounts of data and making the relevant forecasts and decisions in real time to mitigate, sufficiently in advance, potential or actual safety and security issues in a network. In particular, it is virtually impossible to analyze the time series of each individual device by human intervention, creating the need for an automated framework and system. Moreover, within this context, many standard time series analysis models become useless, either because they cannot tackle a huge amount of high frequency data, or because they cannot be used in an automated fashion due to the versatility of the series that need to be faced.

In Chapter 2 of this thesis, we develop a novel reactive defense for network safety and security, proposing a framework for time series monitoring and anomaly detection. This framework serves as basis for a system for large scale safety and security network monitoring. Functionally, we require the system to be:

- *Automatic.* Given the huge amount of series to be monitored, intervention of

humans in the process should be kept to a minimum.

- *Versatile*. Time series may be of different nature and have different characteristics such as linear growth, seasonality or outbursts. The approach should be able to deal with all these issues in an automated fashion. In addition, the time series features may change over time and, thus, the framework should be able to adapt fast and automatically.

- *Scalable*. The approach should scale both in time and memory space. It should be fast enough to be able to cope with many very high frequency time series. In addition, due to the huge amount of time series to be monitored, it should be able to summarize each series with just a few parameters, avoiding storage of the whole series.

- *Accurate*. Besides, we would also like the system to fulfill the required good statistical properties in relation with the predictive accuracy of the provided forecasts.

Earlier work in network monitoring does not fully cover the above requirements. In the classic Brutlag (2000), the author proposed a simple approach based on Holt-Winters forecasting; however, model parameters need to be set and tuned for each model to work well, complicating automation. In addition, his technique is not capable of tracking several seasonal periods, reducing its versatility. In Taylor and Letham (2018), the authors recently proposed an analyst-in-the-loop algorithm which makes use of human and automated tasks, precluding full automation. Moreover, they essentially frame the monitoring problem as a curve-fitting exercise using Generalized Additive Models, not fully taking into account the temporal dependence structure in the data. This way, the dynamic nature of the algorithm and its adaptability to sudden changes are essentially lost. In Vallis et al. (2014) a tool is presented for anomaly detection based on a seasonal trend loess decomposition together with the generalized extreme Student deviation (GESD) test to detect anomalies. This algorithm handles anomaly detection issues, but does not accomplish other important monitoring tasks in relation with forecasting potentially dangerous future events. The main shortcoming is its rigidity in adapting to situations in which the signal features change considerably. Classification and regression trees (CART), Breiman et al. (1984), are also popular methods for anomaly detection. They may be used to provide point and interval forecasts as well as detect anomalies through GESD or Grubbs' tests; their main disadvantage is that a growing number of features can impact computational performance, quickly jeopardizing scalability. ARIMA models have also been widely used in this area (Bianco et al. 2001). These models assume that the signal under study is stationary, possibly after differencing, which is not always the case in many time series network monitoring. In addition, numerous parameters such as the number of differences should be selected in advance, complicating automation. Finally, long short-term memory (LSTM) neural networks, if properly built, may perform successfully in anomaly detection tasks (Malhotra et al. 2015). However,

their main drawback stems from the complex work required to adjust them to reach a proper performance level, rendering automation virtually infeasible.

## 1.4 Proactive Defences in Classification Problems

As highlighted in Section 1.2, most of AML research has been framed within a standard game theory approach pervaded by NE and refinements. However, these entail CK assumptions which are hard to maintain in the security contexts typical of AML applications. We could argue that CK is too commonly assumed.

In this PhD thesis, we propose a formal Bayesian decision theoretic approach to solve AML problems, adopting an ARA perspective to model the confrontation between attackers and defenders mitigating questionable CK assumptions, Rios Insua et al. (2009) and Banks et al. (2015).

ARA makes operational the Bayesian approach to games, Kadane and Larkey (1982) and Raiffa (1982), facilitating a procedure to predict adversarial decisions. Compared with standard game theoretic approaches, it does not assume their CK hypothesis according to which agents share information about utilities and probabilities. ARA provides one-sided prescriptive support to a decision maker maximizing her subjective expected utility by treating the adversaries' decisions as random variables. To forecast them, we model the adversaries' problems; our uncertainty about their probabilities and utilities is propagated and leads to the corresponding random optimal adversarial decisions which provide the required prediction.

In Section 1.4.1, we provide a comparison of game-theoretic and ARA through three template AML models associated, respectively, to white, black and gray box attacks. This suggests a decision theoretic pipeline for AML, illustrated in Section 1.4.2. Finally, in Section 1.4.3 we particularize this methodology to adversarial supervised learning.

### 1.4.1 ARA templates for AML

We provide a comparison of game-theoretic and ARA concepts over three template AML models associated, respectively, to white, black and gray box attacks. They constitute basic structures which may be simplified or made more complex through removing or adding nodes, and combined to accommodate specific AML problems. In all models there is a defender who chooses her decision $d \in \mathcal{D}$ and an attacker who chooses his attack $a \in \mathcal{A}$. In the AML jargon, the defender would be the learning system (or the organization deploying it) and the decisions she makes could refer to the various choices required, including the data set used, the models chosen, or the algorithms employed to estimate the parameters. In turn, the attacker would be the attacking organization (or their corresponding attacking system); his decisions refer to the data sets chosen to attack or the period at which he decides to make it, but could also refer to structural or parallel decisions. The involved agents are assumed

to maximize expected utility (or minimize expected loss) (French and Rios Insua 2000).

We use bi-agent influence diagrams (BAIDS) (Banks et al. 2015) to describe the problems: they include circular nodes representing uncertainties; double nodes representing deterministic aspects; hexagonal utility nodes, modeling preferences over consequences; and, square nodes portraying decisions. The arrows point to decision nodes (meaning that such decisions are made knowing the values of predecessors) or chance and value nodes (the corresponding events or consequences are influenced by predecessors). Arcs pointing to decision nodes are dashed. Different colors suggest issues relevant to just one of the agents (white, defender; gray, attacker); striped ones are relevant to both agents.

### Sequential defend-attack games

We start with sequential games. $D$ chooses her decision $d$ and, then, $A$ chooses his attack $a$, after having observed $d$. This exemplifies white-box attacks, as the attacker has full information of $D$'s action at the time of making his decision. As an example, a classifier chooses and estimates a parametric classification algorithm and an attacker, who has access to the specific algorithm, sends examples to try to fool the classifier. These games have received various names like sequential Defend-Attack (Brown et al. 2006) or Stackelberg (Gibbons 1992). Their BAID is in Figure 1.2. Arc $D$-$A$ reflects that $D$'s choice is observed by $A$. The consequences for both systems depend on an attack outcome $\theta \in \Theta$. Each agent has its assessment on the probability of $\theta$, which depends on $d$ and $a$, respectively called $p_D(\theta|d,a)$ and $p_A(\theta|d,a)$. Similarly, their utility functions are $u_D(d,\theta)$ and $u_A(a,\theta)$.

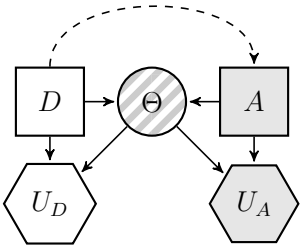



**Figure 1.2**: Basic two player sequential defend-attack game

The basic game theoretic solution does not require $A$ to know $D$'s judgements, as he observes her decisions. However, $D$ must know those of $A$, the CK condition in this case. For its solution, we compute both agents' expected utilities at node $\Theta$: $\psi_A(a,d) = \int u_A(a,\theta)\, p_A(\theta|d,a)\, \mathrm{d}\theta$, and $\psi_D(a,d) = \int u_D(d,\theta)\, p_D(\theta|d,a)\, \mathrm{d}\theta$. Next, we find $a^*(d) = \mathrm{argmax}_{a\in\mathcal{A}}\ \psi_A(d,a)$, $A$'s best response to $D$'s action $d$. Then, $D$'s optimal action is $d^*_{\mathrm{GT}} = \mathrm{argmax}_{d\in\mathcal{D}}\ \psi_D(d, a^*(d))$. The pair $(d^*_{\mathrm{GT}},\, a^*(d^*_{\mathrm{GT}}))$ is a NE and, indeed, a sub-game perfect equilibrium.

**Example.** The Stackelberg game in Brückner and Scheffer (2011), Section 1.2.2, modeling the confrontation in an APP is a particular instance in which costs are minimized with no uncertainty about the outcome $\theta$. $D$ chooses the parameters $d$ of a predictive model. $A$ observes $d$ and chooses the transformation, converting data $\mathcal{T}$ into $a(\mathcal{T})$. Let $\widehat{c}_i(d, a(\mathcal{T}))$ be the (regularized) empirical cost of the $i$-th agent, $i \in \{A, D\}$. Then, they propose a pair $[d^*, a^*(\mathcal{T}(d^*))]$ solving

$$\begin{aligned} \underset{d}{\operatorname{argmin}} \quad & \widehat{c}_D(d, a^*(\mathcal{T}(d))) \\ \text{s.t.} \quad & a^*(\mathcal{T}(d)) \in \underset{a(\mathcal{T})}{\operatorname{argmin}}\, \widehat{c}_A(d, a(\mathcal{T})), \end{aligned}$$

which provides a NE. △

**Example.** *Adversarial examples* can be cast as well through sequential games. $A$ finds the best attack which leads to perturbed data instances obtained from solving the problem

$$\min_{\|\delta\| \leq \epsilon} \widehat{c}_A(h_\theta(a(x)), y),$$

with $a(x) = x + \delta$, a suitable perturbation of the original data instance $x$; $h_\theta(x)$, the output of a predictive model with parameters $\theta$; and $\widehat{c}_A(h_\theta(x), y) = -\widehat{c}_D(h_\theta(x), y)$, the cost of classifying $x$ as of being of class $h_\theta(x)$ when the actual label is $y$.

In turn, robustifying models against those perturbations through AT aims at solving the problem $\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\|\delta_x\| \leq \epsilon} \widehat{c}_D(h_\theta(a(x)), y)\right]$: we minimize the empirical risk of a model under worst case perturbations of the data $\mathcal{D}$. The inner maximization problem is solved through PGD, with iterations $x_{t+1} = \Pi_{B(x)}(x_t - \alpha \nabla_x \widehat{c}_A(h_\theta(x_t), y))$, where $\Pi$ is a projection operator ensuring that the perturbed input falls within a tolerable boundary $B(x)$. After $T$ iterations, we make $a(x) = x_T$ and optimize with respect to $\theta$. Madry et al. (2018) argue that the PGD attack is the strongest one using only gradient information from the target model. However, there has been evidence that it is not sufficient for full defence of neural models (Gowal et al. 2018). The complexity of the above attack depends on the chosen norm; for instance, if we resort to small perturbations under $\ell_\infty$ norm, the update simplifies to $x := x - \epsilon\, \text{sign} \nabla_x \widehat{c}_A(h_\theta(x), y)$, making it attractive because of its low computational burden. The $\ell_2$ norm can also be considered, leading to updates $x := x - \epsilon \frac{\nabla_x \widehat{C}_A(h_\theta(x), y)}{\|\nabla_x \widehat{C}_A(h_\theta(x), y)\|_2}$. △

The above CK condition is weakened if we assume only partial information, leading to games under incomplete information (Harsanyi 1967), but we defer their discussion until we discuss simultaneous games. These CK conditions, and those used under incomplete information, are doubtful as the attacker's judgments are not available. Moreover, the CK hypothesis could lack robustness to perturbations in such judgments, as we shall see in Chapter 5.

Alternatively, we perform a Bayesian decision theoretic approach based on ARA. We weaken the CK assumption: the defender does not know $(p_A, u_A)$ and faces the problem in Figure 1.3a. To solve it, she needs $p_D(a|d)$, her assessment of

the probability that $A$ will implement attack $a$ after having observed $d$. Then, her expected utility would be $\psi_D(d) = \int \psi_D(a, d)\, p_D(a|d)\, \mathrm{d}a$ with optimal decision $d^*_{\mathrm{ARA}} = \arg\max_{d\in\mathcal{D}} \psi_D(d)$. This solution does not necessarily correspond to a NE (both solutions are based on different information and assumptions).[5]


(a) Decision problem seen by defender. (b) Defender analysis of attacker problem.

**Figure 1.3**: Influence Diagrams for defender and attacker problems.

To elicit $p_D(a|d)$, $D$ benefits from modeling $A$'s problem, with his ID in Figure 1.3b. For this, she would use all information available about $p_A$ and $u_A$; her uncertainty about $(p_A, u_A)$ is modeled through a distribution $F = (U_A, P_A)$ over the space of utilities and probabilities. This induces a distribution over $A$'s expected utility, where his random expected utility would be $\Psi_A(a, d) = \int U_A(a, \theta) P_A(\theta|a, d)\, \mathrm{d}\theta$. Then, $D$ would find $p_D(a|d) = \mathbb{P}_F\left[a = \mathrm{argmax}_{x\in\mathcal{A}}\, \Psi_A(x, d)\right]$, in the discrete case and, similarly, in the continuous one. In general, we would use Monte Carlo (MC) simulation to approximate $p_D(a|d)$, as we illustrate in Chapter 5.

### Simultaneous defend-attack games

Consider next simultaneous games: the agents decide their actions without knowing the one chosen by each other. Black-box attacks are assimilated to them. As an example, a defender fits a classification algorithm; an attacker, who has no information about it, sends tainted examples to try to outguess the classifier. Their basic template is in Fig. 1.4.

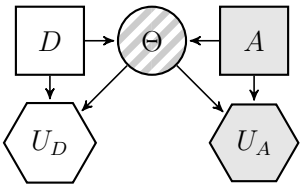



**Figure 1.4**: Basic two player simultaneous defend-attack game.

Suppose the judgements from both agents, $(u_D, p_D)$ and $(u_A, p_A)$ respectively, are disclosed. Then, both $A$ and $D$ know the expected utility that a pair $(d, a)$ would provide them, $\psi_A(d, a)$ and $\psi_D(d, a)$. A NE $(d^*, a^*)$ in this game satisfies $\psi_D(d^*, a^*) \geq \psi_D(d, a^*)\ \ \forall\, d \in \mathcal{D}$ and $\psi_A(d^*, a^*) \geq \psi_A(d^*, a)\ \ \forall\, a \in \mathcal{A}$.

[5] See a cybersecurity example in Chapter 5.

**Example.** Nash prediction games are particular instances of simultaneous defend-attack games with sure outcomes. As in an APP, agents minimize regularized empirical costs, $\widehat{c}_D(d, a(\mathcal{T}))$ and $\widehat{c}_A(d, a(\mathcal{T}))$, with $a(\mathcal{T})$ being the attacked dataset. Under CK of both players' cost functions, a NE $[d^*, a^*(\mathcal{T})]$ satisfies

$$d^* \in \arg\min_d \widehat{C}_D(d, a^*(\mathcal{T})) \qquad a^*(\mathcal{T}) \in \arg\min_{a(\mathcal{T})} \widehat{C}_A(d^*, a(\mathcal{T})).$$

△

If utilities and probabilities are not CK, we may proceed modelling the game as one with incomplete information using the notion of types: each player will have a type known to him but not to the opponent, representing private information. Such type $\tau_i \in T_i$ determines the agent's utility $u_i(d, \theta, \tau_i)$ and probability $p_i(\theta|d, a, \tau_i)$, $i \in \{A, D\}$. Harsanyi proposes Bayes-Nash equilibria (BNE) as their solutions, still under a strong CK assumption: the adversaries' beliefs about types are CK through a common prior $\pi(\tau_D, \tau_A)$ (moreover, the players' beliefs about other uncertainties in the problem are also CK). Define strategy functions by associating a decision with each type, $d : \tau_D \to d(\tau_D) \in \mathcal{D}$, $a : \tau_A \to a(\tau_A) \in \mathcal{A}$. $D$'s expected utility associated with a pair of strategies $(d, a)$, given her type $\tau_D \in T_D$, is

$$\psi_D(d(\tau_D), a, \tau_D) = \int\int u_D(d(\tau_D), \theta, \tau_D) p_D(\theta|d(\tau_D), a(\tau_A), \tau_D) \pi(\tau_A|\tau_D) \, \mathrm{d}\tau_A \mathrm{d}\theta.$$

Similarly, we compute the attacker's expected utility $\psi_A(d, a(\tau_A), \tau_A)$. Then, a BNE is a pair $(d^*, a^*)$ of strategy functions satisfying

$$\begin{aligned} \psi_D(d^*(\tau_D), a^*, \tau_D) &\geq \psi_D(d(\tau_D), a^*, \tau_D), \quad \forall\, \tau_D \\ \psi_A(d^*, a^*(\tau_A), \tau_A) &\geq \psi_A(d^*, a(\tau_A), \tau_A), \quad \forall\, \tau_A \end{aligned}$$

for every $d$ and every $a$, respectively.

The common prior assumptions are still unrealistic in AML security contexts. We thus weaken them in supporting $D$. She should maximize her expected utility through

$$d^* = \arg\max_{d \in \mathcal{D}} \int\int u_D(d, \theta) p_D(\theta \mid d, a) \pi_D(a) d\theta da. \tag{1.4.1}$$

where $\pi_D(a)$ models her beliefs about the attacker's decision $a$, which we need to assess. Suppose $D$ thinks that $A$ maximizes expected utility,

$$a^* = \operatorname*{argmax}_{a \in \mathcal{A}} \int \left[ \int u_A(a, \theta) \right] p_A(\theta|d, a) \, \mathrm{d}\theta \pi_A(d) \, \mathrm{d}d.$$

In general, she will be uncertain about $A$'s $(u_A, p_A, \pi_A)$ required inputs. If we model all information available to her about it through a probability distribution $F \sim (U_A, P_A, \Pi_A)$, mimicking (1.4.1), we propagate such uncertainty to compute the distribution of $A$'s (random) action $\mathscr{A}$, conditioned on $D'$s (random) action $\mathscr{D}$

$$\mathscr{A} \mid \mathscr{D} \sim \operatorname*{argmax}_{a \in \mathcal{A}} \int \left[ \int U_A(a, \theta) \; P_A(\theta \mid d, a) \, \mathrm{d}\theta \right] \Pi_A(\mathscr{D} = d) \, \mathrm{d}d. \tag{1.4.2}$$

$(U_A, P_A)$ could be directly elicited from $D$. However, eliciting $\Pi_A$ may require further analysis leading to an upper level of recursive thinking: she would need to think about how $A$ analyzes her problem (this is why we condition in (1.4.2) by the distribution of $\mathscr{D}$).

In the above, in order for $D$ to assess (1.4.2), she would elicit $(U_A, P_A)$ from her viewpoint, and assess $\Pi_A(\mathscr{D})$ through the analysis of her decision problem, as thought by $A$. This reduces the assessment of $\Pi_A(\mathscr{D})$ to computing $\mathscr{D} \mid \mathscr{A}^1 \sim \text{argmax}_{d\in\mathcal{D}} \int\int U_D(d,\theta)\ P_D(\theta \mid d,a)d\theta\ \Pi_D(\mathscr{A}^1 = a)\,\mathrm{d}a$, assuming she is able to assess $\Pi_D(\mathscr{A}^1)$, where $\mathscr{A}^1$ represents $A$'s random decision within $D$'s second level of recursive thinking. For this, $D$ needs to elicit $(U_D, P_D) \sim G$, representing her knowledge about how $A$ estimates $u_D(d,a)$ and $p_D(\theta|d,a)$, when she analyzes how the attacker thinks about her decision problem. Again, eliciting $\Pi_D(\mathscr{A}^1)$ might require further thinking from $D$, leading to a recursion of nested models, connected with the level-$k$ thinking concept in Stahl and Wilson (1994), which would stop at a level in which $D$ lacks the information necessary to assess the corresponding distributions. At such point, she could assign a non-informative distribution (French and Rios Insua 2000).

**Sequential defend-attack games with private information**

Our final template is the sequential Defend-Attack model with defender private information. Gray-box attacks are assimilated to them. As an example, a defender estimates the parameters of a classification algorithm and an attacker, with no access to the algorithm but knowing the data used to train it, sends examples to try to fool the classifier. Fig. 1.5 depicts the template, with private information represented by $V$. Arc $V - D$ reflects that $v$ is known by $D$ when she makes her decision; the lack of arc $V - A$, that $v$ is not known by $A$ when making his decision. The uncertainty about the outcome $\theta$ depends on the actions by $A$ and $D$, as well as on $V$. The utility functions are $u_D(d,\theta,v)$ and $u_A(a,\theta,v)$.

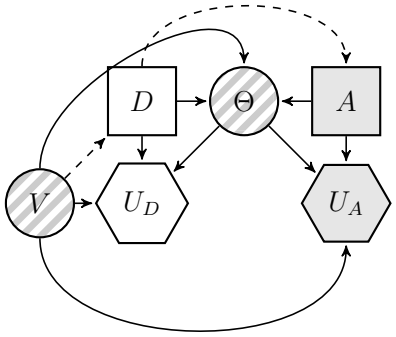



**Figure 1.5**: Basic template for sequential defend-attack game with private information.

Standard game theory solves this model as a signaling game (Aliprantis and

Chakrabarti 2002). For a more realistic approach, we weaken the required CK assumptions. Assume for now that $D$ has assessed $p_D(\theta|d,a,v)$, $u_D(d,\theta,v)$ and $p_D(a|d)$. Then, she obtains her optimal defence through

`At node` $\Theta$`, compute for each` $(d,a,v)$`,`
$\psi_D(d,a,v) = \int u_D(d,\theta,v)\, p_D(\theta \mid d,a,v)\, \mathrm{d}\theta.$

`At node` $A$`, compute` $(d,v) \to \psi_D(d,v) = \int \psi_D(d,a,v)\, p_D(a \mid d)\, \mathrm{d}a$

`At node` $D$`, solve` $v \to d^*(v) = \arg\max_{d\in\mathcal{D}} \psi_D(d,v)$.

To assess $p_D(a|d)$, $D$ could solve $A$'s problem from her perspective. As $A$ does not know $v$, his uncertainty is represented through $p_A(v)$, describing his (prior) beliefs about $v$. Arrow $V - D$ can be inverted to obtain $p_A(v|d)$. Note that we would still need to assess $p_A(d|v)$ for this. Should $D$ know $A$'s utility function $u_A(a,\theta,v)$ and probabilities $p_A(\theta|d,a,v)$ and $p_A(v|d)$, she would anticipate his attack $a^*(d)$ for any $d \in \mathcal{D}$ by

`At node` $\Theta$`, compute for each` $(d,a,v)$`,`
$\psi_A(d,a,v) = \int u_A(d,\theta,v)\, p_i(\theta \mid d,a,v)\, \mathrm{d}\theta$`.`

`At node` $V$`, compute for each` $(d,a)$, $\psi_A(d,a) = \int \psi_A(d,a,v)\, p_A(v \mid d)\, \mathrm{d}v$`.`

`At node` $A$`, solve` $d \to a^*(d) = \arg\max_{a\in\mathcal{A}} \psi_A(d,a)$.

However, $D$ does not know $(p_A, u_A)$. She has beliefs about them, say $F \sim (P_A, U_A)$, which induce distributions on $A$'s expected utilities (which now are random) through

$$\Psi_A(d,a,v) = \int U_A(a,\theta,v)\; P_A(\theta \mid d,a,v)\; \mathrm{d}\theta, \quad \Psi_A(d,a) = \int \Psi_A(d,a,v)\; P_A(v \mid d)\; \mathrm{d}v.$$

Then, $D$'s predictive distribution about $A$'s response to her defense choice $d$ would be defined through

$$p_D(a|d) = P_F\Big[a = \operatorname*{argmax}_{x\in\mathcal{A}}\; \Psi_A(d,x)\Big],\ \forall a \in \mathcal{A}.$$

To sum up, the elicitation of $(P_A(\theta|d,a,v), P_A(v|d), U_A(a,\theta,v))$ allows the defender to solve her problem of assessing $p_D(a|d)$. The defender may have enough information to directly assess $P_A(\theta|d,a,v)$ and $U_A(a,\theta,v)$. Yet the assessment of $P_A(v|d)$ requires a deeper analysis, since it has a strategic component, and would lead to a recursion similar to that in the simultaneous game case.

### 1.4.2 A decision theoretic pipeline for AML

Based on the above three templates, we revisit now the AML pipeline introduced before, proposing an ARA based decision theoretic approach with the same steps.

**1. Model system threats.** This entails modelling the attacker problem from the defender perspective through an influence diagram (ID). The attacker key features are his goals, knowledge and capabilities. Assessing these require determining which are the actions that he may undertake and the utility that he perceives when performing a specific action, given a defender's strategy. The output is the set of attacker's decision nodes, together with the value node and arcs indicating how his utility depends on his decisions and those of the defender. Assessing the attacker knowledge entails looking for relevant information that he may have when performing the attack, and his degree of knowledge about this information, as we do not assume CK. This entails not only a modelling activity, but also a security assessment of the ML system to determine which of its elements are accessible to the attacker. The outputs are the uncertainty nodes of the attacker ID, the arcs connecting them and those ending in the decision nodes, indicating the information available to $A$ when attacking. Finally, identifying his capabilities requires determining which part of the defender problem the attacker has influence on. This provides the way the attacker ID connects with that of the defender.

**2. Simulating attacks.** Based on step 1, a mechanism is required to simulate reasonable attacks. The state of the art solution assumes that the attacker will be acting NE, given strong CK hypothesis. The ARA methodology relaxes such assumptions, through a procedure to simulate adversarial decisions. Starting with the adversary model (step 1), our uncertainty about his probabilities and utilities is propagated to his problem and leads to the corresponding random optimal adversarial decisions which provide the required simulation.

**3. Adopting defences.** In this final step, we augment the defender problem incorporating the attacker one produced in step 1. As output, we generate a BAID reflecting the confrontation big picture. Finally, we solve the defender problem maximizing her subjective expected utility, integrating out all attacker decisions, which are random from the defender perspective given the lack of CK. In general, the corresponding integrals are approximated through MC, simulating attacks consistent with our knowledge level about the attacker using the mechanism of step 2.

### 1.4.3 AML from an ARA perspective

We illustrate now how the previous templates can be combined and adapted through the proposed pipeline to provide support to AML supervised learning models.

Almost every supervised ML problem entails the tasks reflected in Figure 1.6: an *inference* (learning) stage, in which relevant information is extracted from training data $\mathcal{T}$, and a *decision* (operational) stage, in which a decision $y_D$ is made based on the gathered information.

In a general supervised learning problem under a Bayesian perspective, the first stage requires computing the posterior $p(\beta|\mathcal{T}) \propto p(\beta)p(\mathcal{T}|\beta)$, where $p(\beta)$ is the prior on the model parameters; $\mathcal{T}$, the data; and $p(\mathcal{T}|\beta)$ the likelihood. Based on it, the

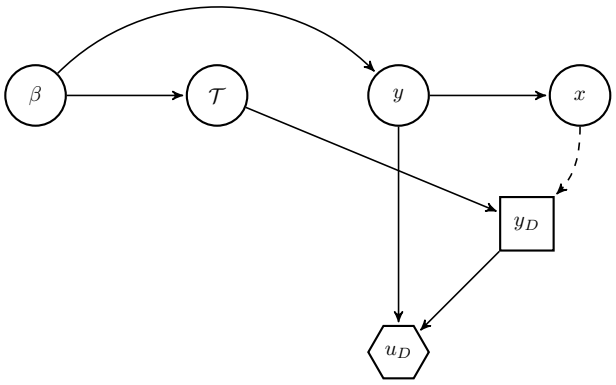



**Figure 1.6**: Influence diagram for a supervised learning problem

predictive distribution is $p(y|x,\mathcal{T}) = \int p(y|x,\beta)p(\beta|\mathcal{T})\,\mathrm{d}\beta$. At the second stage, given an input $x$, the response $y_D$ has to be decided. If the actual value is $y$, the attained utility is $u_D(y,y_D)$ and, globally, the expected utility is

$$\psi(y_D|x,\mathcal{T}) = \int_y u_D(y,y_D)p(y|x,\mathcal{T})\,\mathrm{d}y = \int_y u_D(y,y_D)\left[\int p(y|x,\beta)p(\beta|\mathcal{T})\,\mathrm{d}\beta\right]\mathrm{d}y.$$

We aim at finding $\operatorname{argmax}_{y_D}\psi(y_D|x,\mathcal{T})$.

An attacker might be interested in modifying the inference stage, through poisoning attacks, and/or the decision stage, through evasion attacks. Figure 1.7 represents both possibilities (step 3). In it, $\mathcal{T}'$ denotes the poisoned data and $a_T$ the poisoning attack; $a_O$ denotes the evasion attack decision and $x'$ the attacked feature vector actually observed by the defender. Finally, $u_A$ designates the attacker's utility function. If we just consider evasion attacks, $a_T$ is the identity, $\mathcal{T}' = \mathcal{T}$, and inference will be as in Figure 1.6; when considering only poisoning attacks, $a_O$ would be the identity, the observed instance $x'$ will coincide with $x$, and the decision stage is that in Figure 1.6. Assume that attacks are deterministic transformations of data: applying an attack to a given input will always lead to the same output.

Suppose the defender is not aware of the presence of the adversary. Then, she would receive the training data $\mathcal{T}'$; use the posterior $p(\beta|\mathcal{T}')$, compute the predictive $p(y|x,\mathcal{T}')$; receive the data $x'$ at operation time; and solve $\operatorname{argmax}_{y_D}\psi_D(y_D|x',\mathcal{T}')$. This will typically differ from $\operatorname{argmax}_{y_D}\psi_D(y_D|x,\mathcal{T})$, leading to an undesired performance degradation, as shown in Section 1.1. Should the defender know how she has been attacked, and assuming that attacks are invertible, she would know $a_O^{-1}(x')$ and $a_T^{-1}(\mathcal{T}')$ and maximise $\psi_D(y_D|a_O^{-1}(x'),a_T^{-1}(\mathcal{T}'))$. However, she will not typically know neither the attacks, nor the original data. Thus, upon becoming adversary aware, she would deal with the problem in Figure 1.8, where now $A$'s decisions appear as random. In such case, if her uncertainty is modelled through $p(a_T|\mathcal{T}')$ and $p(a_O|x')$, the defender would optimise

$$\int\left[\int \psi_D(y_D|a_O^{-1}(x'),a_T^{-1}(\mathcal{T}'))p(a_T|\mathcal{T}')da_T\right]p(a_O|x')da_O. \tag{1.4.3}$$

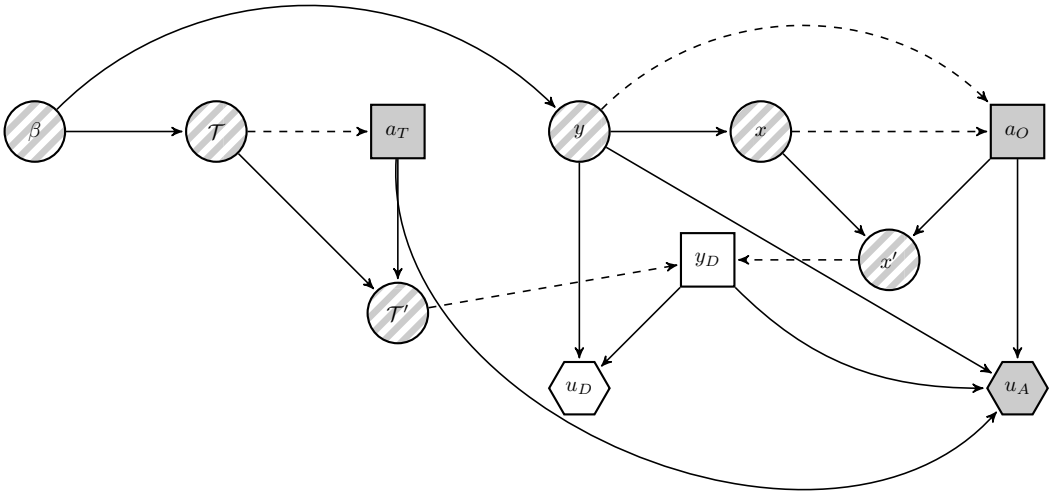



**Figure 1.7**: Influence diagram for a generic adversarial supervised learning problem

We describe a procedure to assess the required distributions following the methodology in Section 1.4.2: (step 1) considers the problem that the attacker would be solving; (step 2) assesses our uncertainty about his problem to simulate from it; then, (step 3) the optimal defense (1.4.3) is proposed.

The attacker aims at modifying data to maximize his utility $u_A(y_D, y, a_T, a_O)$, that depends on $y_D$, $y$, and the attacks $a_T$ and $a_O$, as these may have implementation costs. The form of his utility function depends on his goals. Given that the adversary observes $(\mathcal{T}, x, y)$, if we assume that he aims at maximizing expected utility when trying to confuse the defender, he would find his best attacks through

$$\max_{a_T, a_O} \int u_A(y_D, y, a_T, a_O) p_A(y_D | a_O(x), a_T(\mathcal{T}))\, \mathrm{d}y, \tag{1.4.4}$$

where $p_A(y_D|a_O(x), a_T(\mathcal{T}))$ describes the probability that the defender says $y_D$ if she observes the training data $a_T(\mathcal{T})$ and features $a_O(x)$, from the adversary's perspective. At this point, $D$ must model her uncertainty about $A$'s utilities and probabilities. She will use random utilities $U_A$ and probabilities $P_A$ and look for the random optimal adversarial transformations

$$(A_T, A_O)^*(\mathcal{T}, x, y) = \arg\max_{a_T, a_O} \int U_A(y_D, y, a_T, a_O) P_A(y_D | a_O(x), a_T(\mathcal{T}))\, \mathrm{d}y. \tag{1.4.5}$$

Finally, she computes the probability of the attacker choosing attacks $a_T$ and $a_O$ when observing $\mathcal{T}, x, y$ as

$$p(a_T, a_O | \mathcal{T}, x, y) = \mathbb{P}\left[(A_T, A_O)^*(\mathcal{T}, x, y) = (a_T, a_O)\right]. \tag{1.4.6}$$

Using this, she would compute the required quantities $p(a_T|\mathcal{T}')$ and $p(a_O|x')$.

This last step is problem dependent. In Chapter 3 we illustrate some specificities through an application to AC under evasion attacks.

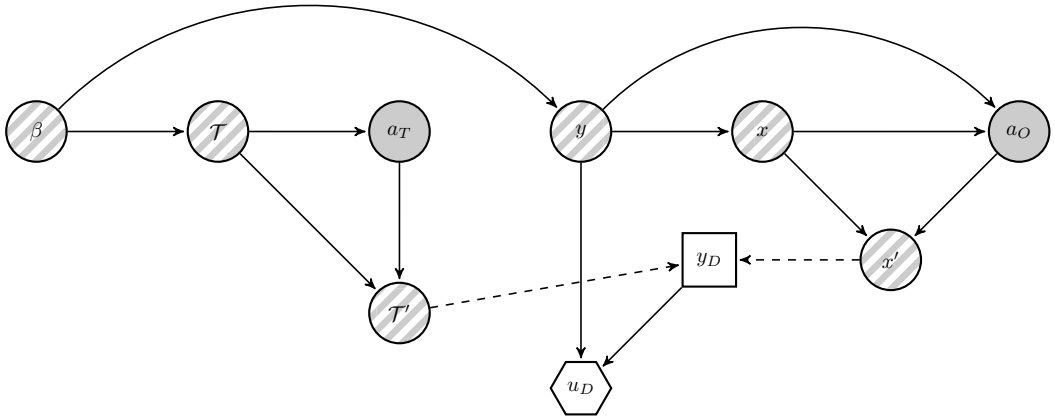



**Figure 1.8**: Supervised learning from the defender's perspective

## 1.5 Algorithmic Approaches in AML

As a byproduct of the raise of importance of game theoretic approaches to AML, algorithmic game theory is also gaining relevance in the last years. In this thesis, we make two important contributions to this field: a gradient-based and a sampling-based solution techniques. We will introduce both of them in the following Sections.

### 1.5.1 Gradient Methods for Stackelberg Games in AML

As reviewed in Section 1.2, the standard approach to guarantee the security of machine learning algorithms against adversarial perturbations, consists of modeling the interaction between the learning algorithm and the adversary as a game in which one agent controls the predictive model parameters while the other manipulates input data. Different game theoretic models of this problem have been proposed. In particular, as introduced in the first example of Section 1.4.1, Brückner and Scheffer (2011) view adversarial learning as a Stackelberg game. Finding Nash equilibria of games in the context if AML is really challenging from a computational perspective as the decision spaces involved are continuous and high dimensional (e.g. choosing parameters of a model, or choosing a data transformation). Surprinsingly, little attention has been paid to algorithmic issues for solving the type of games appearing in AML.

Nash equilibria in general cannot be calculated analytically and numerical approaches are required. However, standard techniques are not able to deal with continuous and high dimensional decision spaces, as those appearing in AML applications. In particular, finding equilibira of Stackelberg games as the ones proposed in Brückner and Scheffer (2011), requires solving a bilevel optimization problem. State of the art numerical solution methods for this kind of problems, Sinha et al. (2018), do not scale well with the dimension of the decision spaces of the intervening agents. In Chapter 4, we propose two gradient-based procedures to solve Stackelberg games in the new paradigm of AML and study their time and space scalability. In

particular, one of the proposed solutions scales efficiently in time with the dimension of the decision space, at the cost of more memory requirements. The other scales well in space, but requires more time.

However, Stackelberg games require strong common knowledge assumptions which are unrealistic in AML as we discuss in Chapter 3. In Chapter 4, we extend the previously proposed algorithmic techniques to deal with Bayesian Stackelberg Games, thus partially mitigating the common knowledge assumption and enhancing robustness.

### 1.5.2 APS Methods for Non-cooperative Games

The gradient-based solution methods to be proposed in Chapter 4, are restricted to games with certain outcomes and continuous decision spaces. In addition, they can just be used to approximate Nash equilibria in Stackelberg games, but they cannot compute ARA solutions, which are more suitable when we lack common knowledge.

In Chapter 5, we propose a simulation-based approach to solve general Stackelberg games paying special attention to games in which the number of decision alternatives is huge or decision spaces are continuous. In particular we analyse how Augmented Probability Simulation, Bielza et al. (1999), may be expanded to efficiently compute game theoretic solutions in both the standard and ARA settings. On the whole, we present a comprehensive robust decision support framework with novel computational algorithms for decision makers in a non-cooperative sequential setup. We cover approaches to approximate subgame perfect equilibria under common knowledge conditions, assess the robustness of such solutions and, finally, approximate ARA solutions when lacking common knowledge.

## 1.6 Research objectives and dissertation structure

Given the ideas introduced in Sections 1.3, 1.4 and 1.5, the main contributions in this thesis are thus summarized in the following objectives.

**O1.** Provide scalable, automatic, versatile and accurate reactive defenses for time series security problems.

**O2.** Develop robust proactive defenses for Adversarial Classification that mitigate the common knowledge assumption.

**O3.** Present scalable gradient based methods for solving Stackelberg games and Bayesian Stackelberg games in Adversarial Machine Learning.

**O4.** Develop Augmented Probability Simulation methods for solving general non-cooperative games.

The above objectives are covered as follows. In Chapter 2, we propose a reactive defense for safety and security monitoring of high frequency time series measuring

performance metrics of hundreds of thousands of Internet Connected Devices, thus fulfilling objective O1. In Chapter 3 we provide a proactive defense for adversarial classification problems, that does not assume common knowledge assumptions, thus fulfilling O2. Finally, in light of the computational challenges arising in AML, we provide two novel algorithmic contributions to approximate solutions of typical problems in AML: a gradient-based approach for solving Adversarial Prediction Problems, Chapter 4 fulfilling objective O3 and a sampling-based approach for solving general non-cooperative games, 5, fulfilling O4.

# Chapter 2

# Reactive Defences in Time Series Security Problems

## 2.1 Motivation

Following objective O1, in this chapter we develop a novel reactive defense for network safety and security by proposing a completely automated, scalable and versatile system framework for time series monitoring and anomaly detection. In the domain of network safety and security, aberrant behavior identification is often based on heuristics developed by analysts and usually lacks predictive capabilities. The framework we propose aims at providing such predictive power to the monitoring system in an automated way. For that purpose, we use a Bayesian approach differentiating between continuous-valued and discrete-valued series. For continuous-valued ones, we use a modified version of dynamic linear models (DLM) (West and Harrison 1997), that incorporates the eventual existence of regular outbursts originated by physical processes such as backups or compressions, specially relevant in our application domain. A main advantage of DLMs is that they can be constructed using different blocks capturing each of the specific features of the time series typical of our domain. We provide an effective way to automatically identify the involved models. In addition, these summarize the relevant aspects of the series in a few parameters, facilitating space scalability. Moreover, computation of the predictive distributions is relatively fast, thus making the approach scalable in time after appropriate tuning. For discrete-valued series, we use discrete time Markov chains (Rios Insua et al. 2012), which fulfill also our above desiderata. We emphasize that the aim of our approach is not develop new, more accurate models for time series, but rather to mix existing solutions in this automated fashion to create a framework that meets the above mentioned requirements.

The structure of the chapter is as follows. A description of the models and their goals are given in Section 2.2, together with how they are identified and how forecasts are made. This allows us to address the versatility and automaticity requirements. We then discuss implementation details in Section 2.3, covering scalability requirements, while in Section 2.4 we analyze the predictive accuracy performance through empirical

tests in different series. We end up with some remarks.

We would like to highlight that the framework exposed in this chapter, was motivated by an industrial application. In particular, it is the solution given to a cybersecurity company based in Madrid, that reached our research group at ICMAT asking for a way to monitor massive traffic time series for safety and security, with strong time and space scalability requirements.

## 2.2 Problem formulation and model description

Consider, for the moment, the case of monitoring a single time series which, in our domain, will refer to some performance measure of an Internet connected device (ICD), say the number of users connected to a device or the percentage of disk storage occupied. Associated with the series, there are two reference values, designated $W$ (*warning*) and $C$ (*critical*), linked with observation levels such that, if exceeded, should lead to issuing warning and critical signals, respectively. For example, for a storage disk we could be monitoring its usage and set $C = 0.95$, meaning that when we reach a saturation of 95%, disk performance might degrade and even collapse, inducing a loss. Such thresholds will depend on the device and its overall relevance. Observe that we have established a two-level alarm system to try to provide richer information about potential failures of the monitored device. In the above example, setting $W = 0.9$ would allow us to raise awareness before it is too late.

As pointed out, several key network monitoring cybersecurity activities are related with safety monitoring and anomaly detection within time series, which, in turn, we shall base on:

i. Making short term forecasts. These allow us to:

   – Identify anomalous behavior of the series when observed values do not lie within predictive intervals. This is related with security and could be useful in pointing critical issues in advance or detecting intruders in a system.

   – Identify unsafe behavior when the predictive intervals actually cover the $W$ and/or $C$ thresholds.

ii. Making long term forecasts. These allow us to foresee critical issues sufficiently in advance when the $W$ and/or $C$ thresholds appear in long-term predictive intervals: critical values that lie within them point to potentially problematic behavior in the distant future.

To accomplish such tasks, we need procedures to make point and interval predictions of the involved time series. The width of the intervals should be adaptable to control for false positives, as well as to take into account the value of the corresponding asset. Whenever the predictions reach either level $W$ or level $C$, an alarm should be issued, being stronger in the latter case. In addition, we should emphasize alarms whenever several of them occur at consecutive time periods.

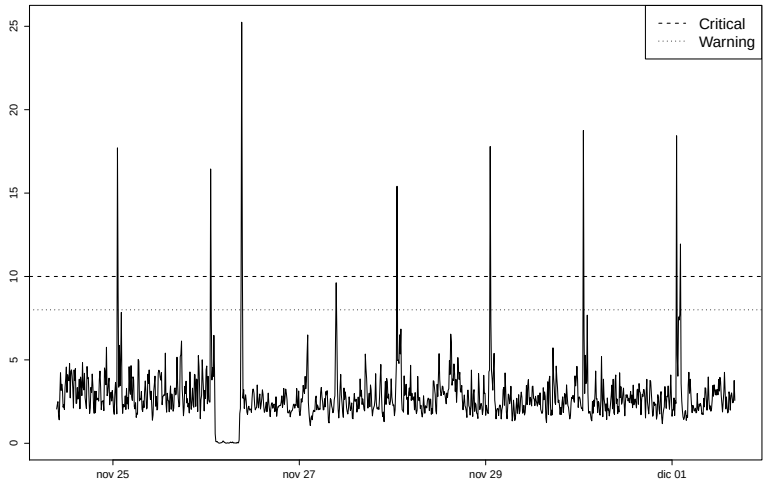



**Figure 2.1**: Continuous time series with linear and outburst terms.

Depending on the specific nature of the series considered, a suitable model will be proposed. We first describe the models used for continuous time series and, then, those for discrete series. Their specific structure is based on extensive analysis performed with actual series in our domain, as illustrated in Section 2.4.

### 2.2.1 Continuous valued time series

Typical continuous time series examples in our domain include device load and disk storage. For a given organization, let $Z_n$ be the $n$-th observation of a relevant continuous variable monitored. If $h$ is the monitoring period, as configured by the user, $Z_n$ represents the observation at time $n \cdot h$. $D_n$ represents the values observed until such time and is recursively defined through $D_n = D_{n-1} \cup \{z_n\}$.

#### Model Definition

We have identified the following relevant types of continuous time series in high frequency network traffic monitoring:

- Series with a level or linear trend, possibly varying in time.
- Series with a level or linear trend together with one (or more) seasonal terms, typically describing daily and/or weekly patterns.
- Series with a level or linear trend, together with several outburst terms, typically associated with compression or backup processes.
- Series with a level or linear trend together with one (or more) seasonal blocks as well as several outburst terms.

As an example, Figure 2.1 shows the average load of a device that goes through a backup process every night around 02:00am producing instantaneous load outbursts. This series would lie within the third type above.

As a consequence, the general expression that we shall adopt for our models will be

$$Z_n = Y_n + S_n + B_n + \epsilon_n,$$

where $Y_n$ designates a linear trend block; $S_n$ designates the seasonal block(s); $B_n$ designates the outburst block(s); and, finally, $\epsilon_n$ designates a noise term. Note that not all three blocks will need to be included in all cases. We describe a procedure to automatically identify the required blocks below. We sketch first their definition.

The trend and seasonal terms are specifications of DLMs, West and Harrison (1997). We consider for them the general, normal DLM with univariate observation $X_n$, where $X_n$ corresponds either to the linear trend ($Y_n$) or the seasonal term ($S_n$). They are characterized by the quadruple $\{F_n, G_n, V_n, W_n\}$: for each $n$, $F_n$ is a known vector of dimension $m \times 1$, $G_n$ is a known $m \times m$ matrix, $V_n$ is a known variance, and $W_n$ is a known $m \times m$ variance matrix. The model is then succinctly written as

$$\begin{array}{ll} \text{Observation:} & X_n|\theta_n \sim N(F_n'\theta_n, V_n), \\ \text{State:} & \theta_n|\theta_{n-1} \sim N(G_n\theta_{n-1}, W_n), \\ \text{Prior:} & \theta_0|D_0 \sim N(m_0, C_0), \end{array}$$

where $\theta_n$ represents the state variable at time $n$. We specify now the general model for the required blocks, which we may combine using the superposition principle (West and Harrison 1997, p. 186–188). The trend model is a specification of the DLM with constant $F_n$ and $G_n$ through

$$F = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad G = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

In turn, the basic seasonal model with period $s$ is a specification of the DLM with constant $F_n$ and $G_n$

$$F = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}}_{s-1}, \qquad G = \underbrace{\begin{bmatrix} -1 & -1 & -1 & \cdots & -1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{s-1}.$$

Typical periods that we incorporate are $s = 288$ for 5 minute data and $s = 144$ for 10 minute data. For larger periods, we might use a Fourier decomposition and work with a few of the Fourier components (Petris et al. 2009, p. 102–109). For both models we use $V_n = 1$; however, this parameter could be assessed using e.g. maximum likelihood estimation. The variance matrix $W_n$ is defined through the discount principle with discount factor of 0.95 (West and Harrison 1997, p. 193–200).

In case some outburst processes are detected, then at the corresponding times, the DLM model is *turned off* and the analysis of these points is made separately. To this end we use a normal model for each particular outburst process

$$B_{n_p} \sim \mathcal{N}(\mu, \sigma^2),$$

with mean $\mu$ and variance $\sigma^2$. Here, $n_p$ would represent the index of the $p$-th outburst of a given type.

### Model Identification

By default, we use as baseline a linear trend block on top of which we add seasonal and outburst blocks when such effects seem relevant. These are identified as follows:

- **Seasonal component:** We store the lags $t_i$ at which sign changes in the sample autocorrelation function of the series take place. We then calculate the difference between the closest non-consecutive lags, $(t_{i+2} - t_i)$ for $i = \{1, \cdots, j-2\}$, assuming that there are $j$ changes, and compute the sample mean $(m)$ and variance $(s^2)$ of differences. If $s/m < r$, where $r$ is an adjustable threshold, we include a seasonal component with nearest integer to $m$ as the estimated seasonality. Figure 2.2 represents the autocorrelation of a series of period 48 in a specific example.

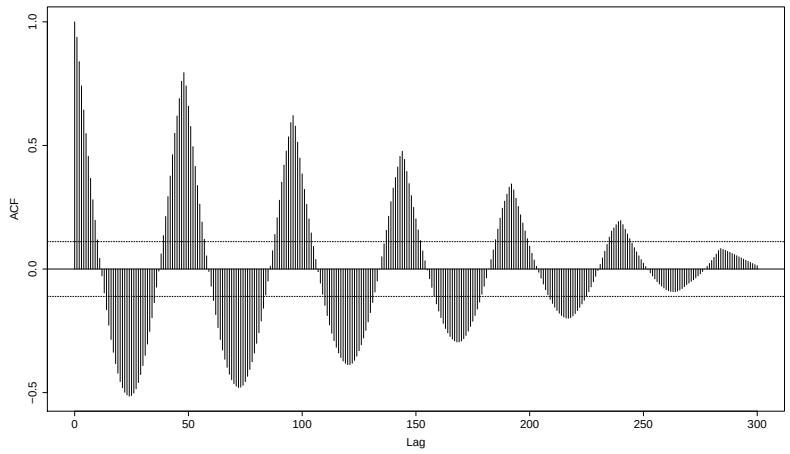



**Figure 2.2**: Sample autocorrelation function of a network monitoring series with seasonal component. The period is 48, corresponding to measuring the series twice every hour for a full day.

- **Outbursts:** We first search through the dataset looking for times in which the data lie outside $\gamma$ standard deviations from the estimated mean of the series, with $\gamma$ an adjustable threshold. For these times we record how many instances $b_j$ of the same type of outburst occur. For example, when analyzing daily-regular outbursts we take a certain number of days and count in how

many of them does a peak appear at the same hour. After a sufficiently long time, we declare that this is an identified regular outburst process if

$$\frac{b_j}{b} > q, \tag{2.2.1}$$

where $b$ is the number of periodical time intervals in the data used for model identification purposes, e.g. the total number of days in our previous example. In addition, $q$ is a *repetition threshold* that can be tuned to make peak selection stricter. Condition (2.2.1) suggests that peaks need to appear, at least, $q \times b$ times throughout the data timespan to be considered part of a regular outburst process.

For this identification process, we use a large enough amount of data, able to capture the main features of the time series under study. In our application domain, where daily and weekly effects are the most relevant, 5 weeks have been usually sufficient.

### Priors

It could be the case that we have available prior information for specific monitored series. This will typically entail improved performance in terms of faster fitting. However, to automate the approach we need to be generic and have adopted default diffused priors as follows:

*Linear term.* The adopted prior is $\mathcal{N}(m_0, C_0)$, where

$$m_0 = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad , \quad C_0 = \begin{bmatrix} 10^7 & 0 \\ 0 & 10^7 \end{bmatrix}.$$

*Seasonal term.* We use again a $\mathcal{N}(m_0, C_0)$ prior. We adopt as $m_0$ a vector of 0's of dimension $s-1$, $s$ being the period. The covariance matrix, of dimension $(s-1) \times (s-1)$, takes the form

$$C_0 = \begin{bmatrix} A & B & \cdots & B \\ B & \ddots & & \vdots \\ \vdots & & \ddots & B \\ B & \cdots & B & A \end{bmatrix},$$

where $A$ is a large positive value and $B$ is a negative value defined so that the sum of every row and column is zero. Thus, we set $B = -\frac{A}{s-2}$.

*Outburst term.* We use a noninformative prior

$$\Pi(\mu, \sigma^2) \propto \frac{1}{\sigma^2}.$$

**Model Forecasting**

We describe how we make forecasts with the above models. Petris et al. (2009, p. 53–55) provides closed forms of the one-step ahead predictive distribution $\mathcal{N}(f_n, Q_n)$ of $X_n|D_{n-1}$, affecting both the trend and seasonal terms, and their superposition thereof. As a consequence:

- The pointwise forecast at the next period $n$ is $E[X_n|D_{n-1}] = f_n$.
- If $u_n$ and $l_n$ represent, respectively, the upper and lower bounds of the expected predictive interval with probability content $\alpha$, they are defined through

$$\begin{aligned} u_n &= f_n + z_{1-\alpha/2}\, Q_n^{1/2}, \\ l_n &= f_n - z_{1-\alpha/2}\, Q_n^{1/2}, \end{aligned} \tag{2.2.2}$$

  where $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the standard normal distribution, being $\alpha$ the desired probability level of the predictive interval, which may be chosen by design, e.g. depending on the asset value of the corresponding device. By default, we use $\alpha = 0.95$.

$k$-step ahead forecasts are also relevant in our context. If we use $a_n(k) = E[\theta_{n+k}|D_n]$, $f_n(k) = E(Y_{n+k}|D_n)$, and $Q_n(k) = Var(Y_{n+k}|D_n)$, (Petris et al. 2009, p. 70–71) provide closed forms for such means and variances. As a consequence:

- The $k$-step ahead point forecast for the trend term at time $n$ is

$$f_n(k) = a_n(0) + a_n(1)k. \tag{2.2.3}$$

- The $k$-step point forecast for the seasonal term at time $n$ is

$$f_n(k) = a_n(0)_{k \bmod s}. \tag{2.2.4}$$

Prediction intervals take a form similar to (2.2.2).

For the outbursts, for simplicity, we index the peaks of a particular type with $p$, the number of observed outbursts of such type until the corresponding time. Then

$$B_{p+1}|D_p \sim \mu_p + t_{p-1}\sqrt{\left(1+\frac{1}{p}\right)\sigma_p^2},$$

where $\mu_p$ is the sample mean after observing $p$ outbursts of the corresponding type, and $\sigma_p^2$ is the sample variance. We update recursively the parameters through

$$\mu_{p+1} = \frac{p\mu_p + X_{p+1}}{p+1}, \qquad \sigma_{p+1}^2 = \frac{\sigma_p^2(p-1) + k\mu_p^2 + X_p^2 - (p+1)\mu_{p+1}^2}{p},$$

where $X_p$ is the value of the series at the time corresponding to the $p$-th outburst of the given type. Then,

- The *point* forecast is $E[B_{p+1}|D_p] = \mu_p$.
- The *interval* forecast is

$$\begin{aligned} u_{p+1} &= \mu_p + t_{1-\frac{\alpha}{2},p+1}\sqrt{\left(1+\frac{1}{p}\right)\sigma_p^2}, \\ l_{p+1} &= \mu_p - t_{1-\frac{\alpha}{2},p+1}\sqrt{\left(1+\frac{1}{p}\right)\sigma_p^2}, \end{aligned} \tag{2.2.5}$$

  where $t_{1-(\alpha/2),p+1}$ is the $1-\alpha/2$ quantile of the $t$-distribution with $p-1$ degrees of freedom, with $\alpha$ as above.

Finally, for general forecasts, we first distinguish between regular points and outbursts. For regular points, the prediction is based on the superposition of the involved DLM components. However, if the prediction refers to an outburst, the DLM model is switched off and the outburst forecast is used.

### 2.2.2 Discrete valued time series

We briefly sketch the approach with this type of series. For further details see Rios Insua et al. (2012). We use finite, time homogeneous Markov chains $\{X_n\}$, with states $\{1,\ldots,K\}$, where $K = C + c$, is the stated critical level plus a small integer $c$. We write the transition matrix as $\mathbf{P} = (p_{ij})$ where $p_{ij} = P(X_n = j|X_{n-1} = i)$, for $i, j \in \{1,\ldots,K\}$. Should it exist, the stationary distribution $\pi$ is the unique solution of $\pi = \pi\mathbf{P}$, $\pi_i \geq 0, \sum \pi_i = 1$.

We assimilate the monitored data $D_n$ to observing $n$ successive transitions of the Markov chain, say $X_1 = x_1,\ldots,X_n = x_n$, given the known initial state $X_0 = x_0$. A natural prior for $\mathbf{P}$ is defined by letting $\mathbf{p}_i = (p_{i1},\ldots,p_{iK})$ have a Dirichlet distribution $\mathbf{p}_i \sim \mathrm{Dir}(\boldsymbol{\alpha}_i)$ where $\boldsymbol{\alpha}_i = (\alpha_{i1},\ldots,\alpha_{iK})$ for $i = 1,\ldots,K$ are independent vectors. Then, the posterior is $\mathbf{p}_i|D_n \sim \mathrm{Dir}(\boldsymbol{\alpha}_i')$ where $\alpha_{ij}' = \alpha_{ij} + n_{ij}$ for $i, j = 1,\ldots,K$; being $n_{ij} \geq 0$ the number of observed transitions from state $i$ to state $j$. Specifically, the prior adopted is given by the coefficients $\boldsymbol{\alpha}_i$ of the corresponding Dirichlet distributions presented in the following $K \times K$ matrix

$$\begin{bmatrix} \alpha & \beta & \gamma & \gamma & \cdots & \gamma \\ \beta & \alpha & \beta & & \cdots & \gamma \\ \gamma & \beta & \ddots & & & \vdots \\ \vdots & \vdots & & & & \beta \\ \gamma & \gamma & \cdots & & \beta & \alpha \end{bmatrix}.$$

In our case, due to the high frequency of the time series involved, the most-likely behavior for a certain state is to remain in its position, followed by one-state transitions. The rest of transitions are less likely. We model this behavior by setting $\alpha > \beta > \gamma$. In particular, our default values are $\alpha = 10$, $\beta = 8$ and $\gamma = 2$. As before,

the proposed prior is general, flexible and useful to deal with the cases we have found in network monitoring, facilitating a generic automated approach.

We predict the next value of the Markov chain, at time $n+1$, using our estimates of the transition probabilities

$$P(X_{n+1} = j|D_n) = \int p_{x_n j} f(\mathbf{P}|D_n)\, d\mathbf{P} = \frac{\alpha_{x_n j} + n_{x_n j}}{\alpha_{x_n \bullet} + n_{x_n \bullet}} \equiv \widehat{p}_{X_n,j},$$

where $\alpha_{i\bullet} = \sum_{j=1}^{K} \alpha_{ij}$ and $n_{i\bullet} = \sum_{j=1}^{K} n_{ij}$. The pointwise prediction will then be $\sum_{j=1}^{K} j \cdot \widehat{p}_{X_n,j}$. Once the prediction of the next value is performed, we can also calculate the prediction interval around such estimation. If $i$ is the last visited state

**Algorithm 1:** Predictive interval calculation for discrete time series.

**Data:** Last visited state: $i$. Transition probabilities: $\widehat{p}_{i,j}$ for $j = 1, 2, \ldots, K$.
**Initialization:** sum $= p_{i,i}$, $u_{n+1} = i$, $l_{n+1} = i$ ;
**while** *sum* $< \zeta$ **do**
  **if** $u_{n+1} = K$ **then**
    continue, interval stops growing upwards;
  **else**
    $u_{n+1} = u_{n+1} + 1$;
    sum $=$ sum $+ \widehat{p}_{i,u_{n+1}}$;
  **end**
  **if** $l_{n+1} = 1$ **then**
    continue, interval stops growing downwards;
  **else**
    $l_{n+1} = l_{n+1} - 1$;
    sum $=$ sum $+ \widehat{p}_{i,l_{n+1}}$;
  **end**
**end**
**Result:** $u_{n+1}$, $l_{n+1}$

and $\zeta$ is the probability of the one step ahead predictive interval, we get its upper and lower bounds $u_{n+1}$ and $l_{n+1}$, using Algorithm 1. Note that discrete time series have both *maximum* and *minimum* states, here $K$ and 1 respectively.

$k > 1$ steps ahead predictions are more complex. For small $k$, we can use

$$P(X_{n+k} = j|D_n) = \int \left(\mathbf{P}^k\right)_{x_n j} f(\mathbf{P}|D_n)\, d\mathbf{P},$$

giving a sum of Dirichlet expectation terms. However, as $k$ increases, the evaluation of this expression becomes computationally infeasible in our domain. Thus, we perform approximately by calculating the $k^{th}$ power $\widehat{P}^k$ of the matrix of expected values of the probabilities and using

$$P(X_{n+k} = j|D_n) \approx \left(\widehat{P}^k\right)_{nj}.$$

Our interest also lies in the stationary distribution of the chain. For high dimensional chains as the ones we need to deal with, we use the approximation

$$\begin{aligned} \hat{\pi} &= \hat{\pi}\hat{\mathbf{P}}, \\ \sum_i \hat{\pi}_i &= 1, \\ \hat{\pi}_i &\geqslant 0. \end{aligned} \tag{2.2.6}$$

Once this equation is solved, we can use $\hat{\pi}_i \geqslant 0$, the approximate stationary distribution, to produce long-term forecasts.

### 2.2.3 General scheme

The general strategy followed for each time series is described in Figure 2.3 (this whole procedure is applied in an automated fashion to each new time series inputted to the system). Once the series is in, a first distinction is made between whether it is continuous or discrete. In this last case, the system uses the Markov model in Section 2.2.2. Otherwise, the model identification process presented is applied. Once the blocks have been identified, the appropriate dynamic model in Section 2.2.1 is used. After fitting the corresponding models based on a sufficient amount of data, forecasts may be made and alarms raised in case an anomalous behavior is detected or reaching critical values is predicted, both in the short or medium terms.

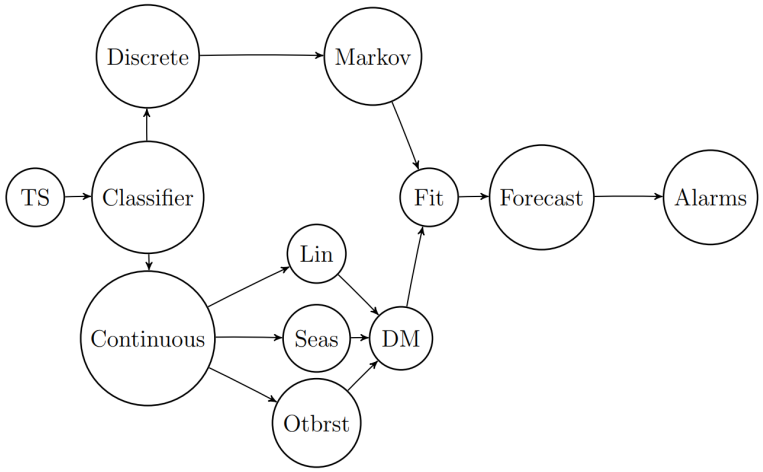



**Figure 2.3**: General scheme.

## 2.3 Implementation

We now describe the implementation and use of our approach. We present here issues in relation with having to deal with several hundred thousands of monitored series sampled over periods ranging from 1 to 10 minutes, depending on the criticality of

the corresponding device. We thus cover the speed and space scalability requirements mentioned above.

The implementation has been carried out through the creation of an object-oriented python package linked with the core monitoring system. Each monitored time series will be associated with an object that includes all methods required to accomplish predictive monitoring tasks. The package is structured as in Figure 2.4.

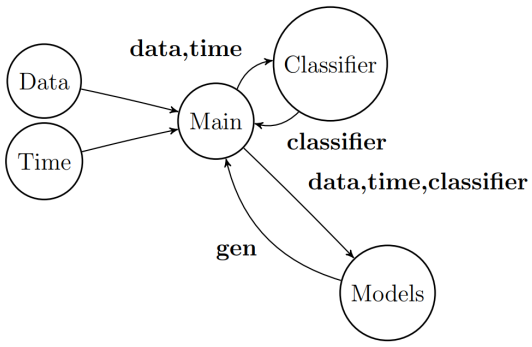
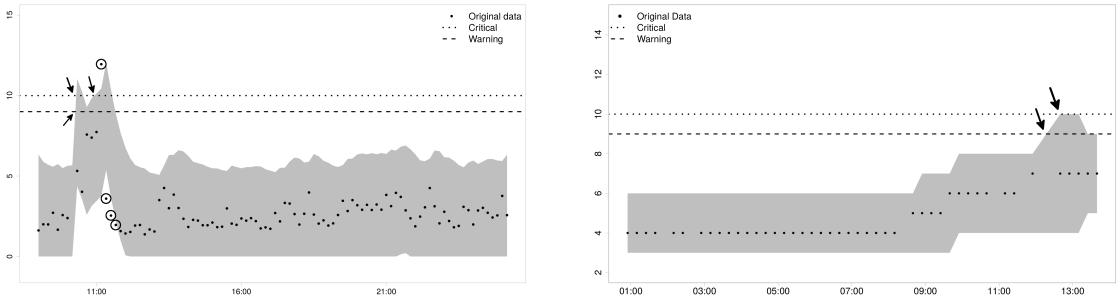



**Figure 2.4**: Structure of the python package for time series monitoring and anomaly detection.

The main code blocks are the classes *gen* and *classifier*. The main code receives the data and timestamps of the training period of each series and generates a *classifier* object. Its attributes contain the characteristics needed to identify the class of models to be used in the analysis of that particular time series. Once this object is constructed, using it together with the data and timestamps, a *gen* object is created. This is the general class whose methods depend on the model relevant for the given time series, which will ultimately be used to undertake the forecasting and anomaly and critical value detection tasks. Among these, we highlight the *update* method, which incorporates new data to the current model and updates the parameters in a Bayesian manner, and the *predict* method, which performs $k$-step ahead predictions, returning point and interval forecasts, providing the basis to construct the specific critical and anomaly detection functions that raise the alarms.

Short term forecasts are needed for safety and security purposes. Given $D_n$, we produce forecasts using the *predict* method for future observations $x_{n+1}$, $x_{n+2}$, $x_{n+3}, \cdots$ up to a certain time $(n+k)\cdot h$, with $k$ defined by the user. $k = 3$ is the default value. Forecasts are delivered as predictive intervals $[l_{n+i}, u_{n+i}]$ with probability level configurable by the user. These could be used to detect aberrant behavior within the time series: when the next observation $x_{n+1}$ does not lie in the predictive interval $[l_{n+1}, u_{n+1}]$, a warning concerning unexpected behavior is displayed pointing to a potential security issue. Our implementation modulates the warning depending on the number of consecutive intervals in which it needs to be launched. In order to control for false positives, alarms could be issued just when the number of violations exceeds certain threshold within a moving window with fixed number of time steps, as suggested in Brutlag (2000). Short term predictions also serve for

safety monitoring tasks. Specifically, when $W$ or $C$ lie within a predictive interval $[l_{n+j}, u_{n+j}]$ for one $j \in \{1, 2, ..., k\}$, an alarm pointing to potentially high values of $x$ should be issued. The higher the number of intervals covering the particular level, the more intense the alarm would be. We illustrate both functionalities through a




(a) Continuous time series. (b) Discrete time series.

**Figure 2.5**: Short term forecasting using 95% one-step ahead predictive intervals.

practical example in Figure 2.5(a). In this case, an intense unexpected behavior alarm would be issued around 2 PM as there are four measurements outside the corresponding predictive intervals. In addition, an alarm concerning a potentially high value of the series would be issued at 1:40 PM, as the predictive intervals exceed the warning and critical levels at 1:50 PM and 2:00 PM, respectively. A similar example using discrete time series is displayed in Figure 2.5(b).

Long term forecasts are used to accomplish safety monitoring tasks: we try to ascertain whether critical levels will be reached with sufficiently high probability in the long term. In principle, it could be done using predictive intervals as we do with short term predictions and illustrated in Section 2.2.1. However, since the system must monitor so many high frequency time series in real time and, consequently, must perform under a very narrow time window, we need to make a compromise: although the calculation of predictive intervals is feasible, here we will just focus on point forecasts. This allows for a drastic improvement in the computational costs of running the involved algorithms, as we can use explicit expressions to make the forecasts, obviating the costly computation of predictive variances. We use as point forecasts $z_{n+j}$, the midpoint of intervals $[l_{n+j}, u_{n+j}]$, and try to find out the first $j_1$ such that $z_{n+j_1} > W$ and the first $j_2$ such that $z_{n+j_2} > C$. This allows us to identify, well in advance, time instants in which critical values might be reached, as we may provide explicit expressions of such inequalities. Indeed, in the linear case, from (2.2.3), we try to find the first $j_i, i = 1, 2$, such that

$$j_1 > \frac{W - a_n(0)}{a_n(1)}, \quad j_2 > \frac{C - a_n(0)}{a_n(1)},$$

assuming that $a_n(1) > 0$. Similarly, with a seasonal block, from (2.2.4), we try to find the first $j_i, i = 1, 2$ such that

$$a_n(0)_{j_1 \bmod s} > W, \quad a_n(0)_{j_2 \bmod s} > C,$$

if any. Finally, when the model contains linear and seasonal blocks, we compute $j_1$ (and similarly $j_2$) through Algorithm 2, where $a'_n(0)$ are the parameters of the pointwise $k$-step ahead forecasts for the seasonal term, and $a_n(0)$ and $a_n(1)$ are those of the linear trend.

**Algorithm 2:** Long term forecast linear + seasonal trend.

**while** $a_n(0) + a_n(1)k + a'_n(0)_{k \ mod \ s} < W$ **do**
    $k = k + 1$;
    $j_1 = k$;
**end**

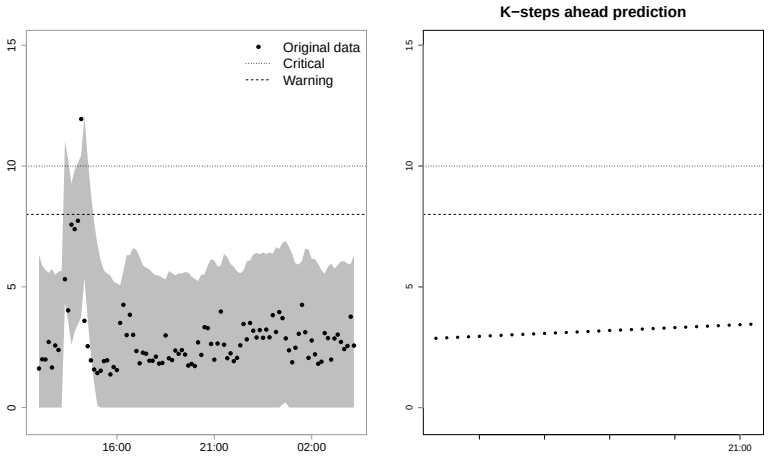



**Figure 2.6**: Long term forecast for continuous time series

A practical example of this type of forecast is illustrated in Figure 2.6. There, the time series is not expected to cross neither warning nor critical levels in the next five hours. However, if the linear growth in the main trend continues, both levels could possibly be reached in the future. In order to control for false positives, we could define a *relevance time window*, only raising an alarm when anomalous behavior occurs repeatedly inside it and neglecting them otherwise.

For long term forecasting with discrete time series, we use the stationary distribution in (2.2.6). If the sum of probabilities assigned to the states that lie above the warning (critical) level is higher than a certain prestablished threshold, an alarm would be raised. This is illustrated in Figure 2.7. In this case, an alarm should be raised if the warning (critical) threshold is at 0.07 or lower.

We assessed the time performance of our approach with stress tests. For different models, we have sequentially added 15000 new data points through the *update* method, performing a short term prediction (30 minutes) and a long term prediction (5 hours) at each iteration. In particular, we tested four models: one with a linear trend; one with a linear trend and a seasonal block with period 144; a model with a linear trend and an outburst block; and, finally, a Markov chain model with 50 states. In Table 2.1, we show the mean, median, min and max times of the whole operation (update + short term prediction + long term prediction), for each of the

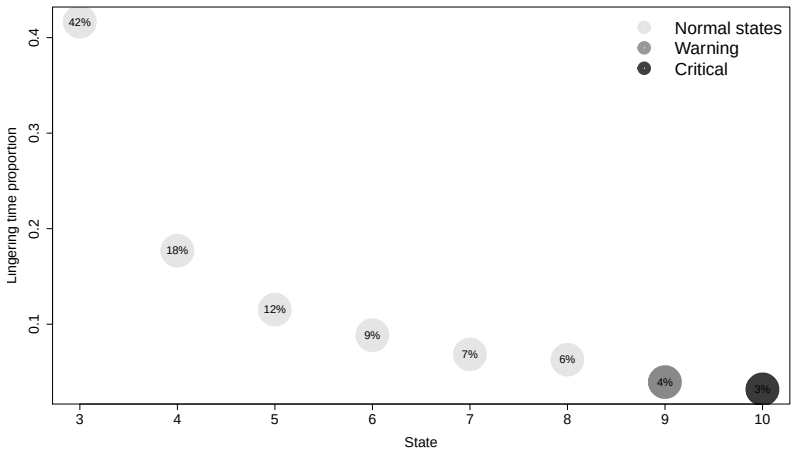



**Figure 2.7**: Long term forecast for discrete time series

models used. Time calculations have been performed in an Intel Core i7-3630UM, $2.40GHz \times 8$ machine.

| | **Mean** | **Median** | **Min** | **Max** |
|---|---|---|---|---|
| Linear | $4.03 \cdot 10^{-4}$ | $3.97 \cdot 10^{-4}$ | $3.76 \cdot 10^{-4}$ | $9.12 \cdot 10^{-4}$ |
| Linear + Seasonal | $2.50 \cdot 10^{-2}$ | $2.49 \cdot 10^{-2}$ | $2.43 \cdot 10^{-2}$ | $4.60 \cdot 10^{-2}$ |
| Linear + Outburst | $4.42 \cdot 10^{-4}$ | $4.35 \cdot 10^{-4}$ | $2.73 \cdot 10^{-4}$ | $8.38 \cdot 10^{-4}$ |
| Markov Chain | $8.81 \cdot 10^{-4}$ | $8.06 \cdot 10^{-4}$ | $7.70 \cdot 10^{-4}$ | $1.96 \cdot 10^{-3}$ |

**Table 2.1**: Mean, median, min and max times in seconds of update, short term and long term prediction for different models.

The algorithm is fast enough, and consequently, able to cope with the typical high frequency data in the network monitoring domain. However, note that the model including the seasonal trend is remarkably slower than the other ones. This is to be expected since it includes a seasonal term with long period, which involves performing several operations with high dimensional matrices. Should better performance be required, we could use a Fourier decomposition of the seasonal trend, and work with a few of the most relevant Fourier components (Petris et al. 2009, p. 102–109), although this would require a method to automate identification of such components.

In terms of memory, the whole approach is constructed so that a fixed number of parameters are stored at any step of the calculations to further improve its performance. In the case of continuous time series, the linear part of the DLM stores two 2-component vectors, the vector $F$ and the mean of the state distribution $m$; three $2 \times 2$ matrices, the matrix $G$, the covariance matrix $C$ of the state distribution and the system covariance; and a parameter corresponding to the observation variance. Similarly, the seasonal part stores two $(s-1)$-component vectors, three $(s-1) \times (s-1)$ matrices and one parameter. The outburst term holds only two parameters corresponding to the mean and variance of the corresponding distribution,

for each recurrent peak detected. Finally, the Markov model for discrete time series employs a $k \times k$ matrix that is updated at each step, where $k$ corresponds to the number of states considered in the chain. This means that in the worst case scenario, the model needs to store in memory $\mathcal{O}(s^2)$ parameters for the continuous case and $\mathcal{O}(k^2)$ for the discrete case, which is feasible within the architecture developed.

These remarks, together with the time performance measures previously presented, suggest that the algorithm fulfills the time and space scalability requirements.

## 2.4 Empirical test for accuracy

This section provides empirical support for the accuracy of the proposed framework for network monitoring, fulfilling our fourth requirement. It also illustrates how the approach adopted was designed based on modeling numerous series in our domain to obtain the relevant features.

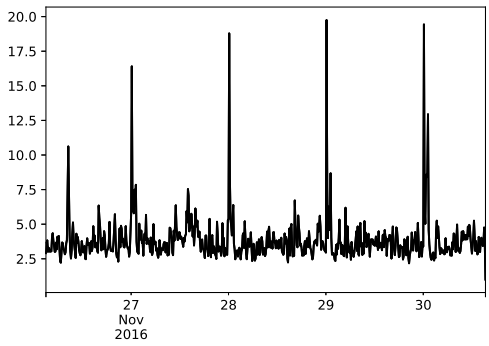
To that end, we use as benchmark the four time series plotted in Figure 2.8, which are representative of the above mentioned characteristics in network traffic monitoring (linear trends, seasonal behaviors, outbursts and discrete processes). In this way, we show that our automated framework meets the aforementioned properties of scalability and versatility, being able to deal with time series of very different nature.

To assess the overall accuracy of our approach, we use some of the traditional performance metrics for point forecasts. In particular, we use the mean absolute error (MAE), the mean square error (MSE) and the mean absolute percentage error (MAPE). We compare our modeling framework with some of the traditional methodologies mentioned in the introduction, specifically ARIMA and exponential smoothing (ES), as there are open source automated implementations available. Specifically, we use the *forecast* R package, Hyndman and Khandakar (2008) and Hyndman et al. (2018). In our case, for the comparison, it is important to use monitoring methods that are able to deal in a completely automated fashion with large amounts of different time series.

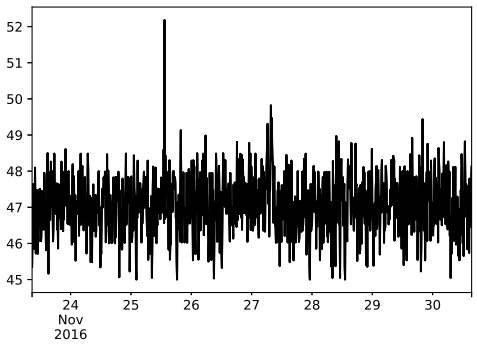
Note first that our forecast, in contrast to those provided by the other methods, is a full predictive distribution rather than a single point forecast. Then, if we want to compute the above mentioned performance metrics, we need to summarize our predictive distribution with a single point. To that end, we shall choose the point that minimizes the corresponding expected loss under the predictive distribution[1].

Figure 2.9 plots the different performance metrics versus the forecast horizon from one to ten steps ahead for the first three time series in Figure 2.8. Notice that the accuracy of our method is deemed competitive. In particular, our approach notably outperforms ARIMA and exponential smoothing in cases 1 and 3. In the first case, this is due to the fact that we are capable of accounting for the outburst process in

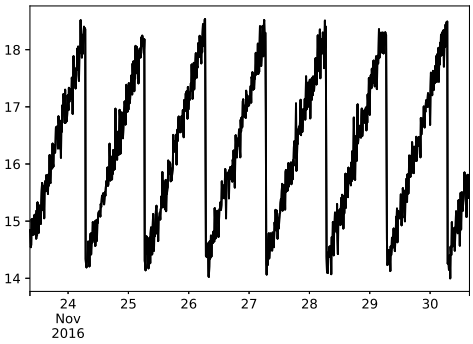
[1]It is well known that the predictive median is optimal under absolute loss; the mean, under squared loss, and the $(-1)$-median under the absolute percentage loss, that is the median of the p.d.f. $g(y) \propto p(y)/y$ where $p(y)$ is the forecast distribution.



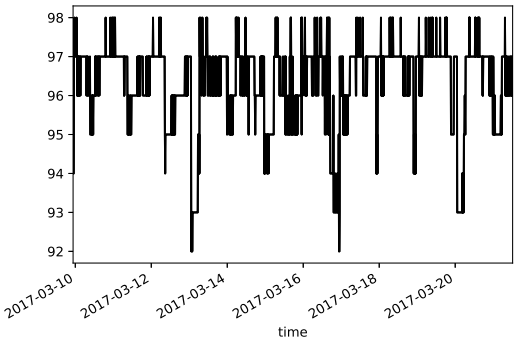

Case 1: Continuous series with outbursts.




Case 2: Continuous series.




Case 3: Seasonal series.




Case 4: Discrete series.

**Figure 2.8**: Time series used for the comparisons.

such sample. Modeling these regular outbursts separately makes all the difference here. For sample 2, all methods have a similar performance in all three metrics, due to its simpler nature. However, for sample 3, the automated approach implemented in the *forecast* R package for ARIMA and ES was not capable of recognizing the seasonal component in such data. Therefore, the difference in performance is due to the fact that our approach is correctly identifying the seasonal component and the error remains more stable through the number of steps ahead of the predictions.

To further study sample 3 results we also computed these performance metrics for ARIMA and ES, tuning them manually to explicitly account for the corresponding seasonal behavior. Results are presented in Figure 2.10. Here we see that once ARIMA and ES are given this information, their performance in the long term predictions improves sharply. However, even in this case our approach, still being completely automatic, remains competitive. Moreover, if we also consider the scalability requirement, both ES and ARIMA have much higher running times, being specially high in the case of ARIMA. Therefore, even though we had to intervene the ARIMA and ES models to correct them and gave them much more running time to produce the forecasts, our method still performs well enough while fulfilling the requirements of automaticity and scalability.

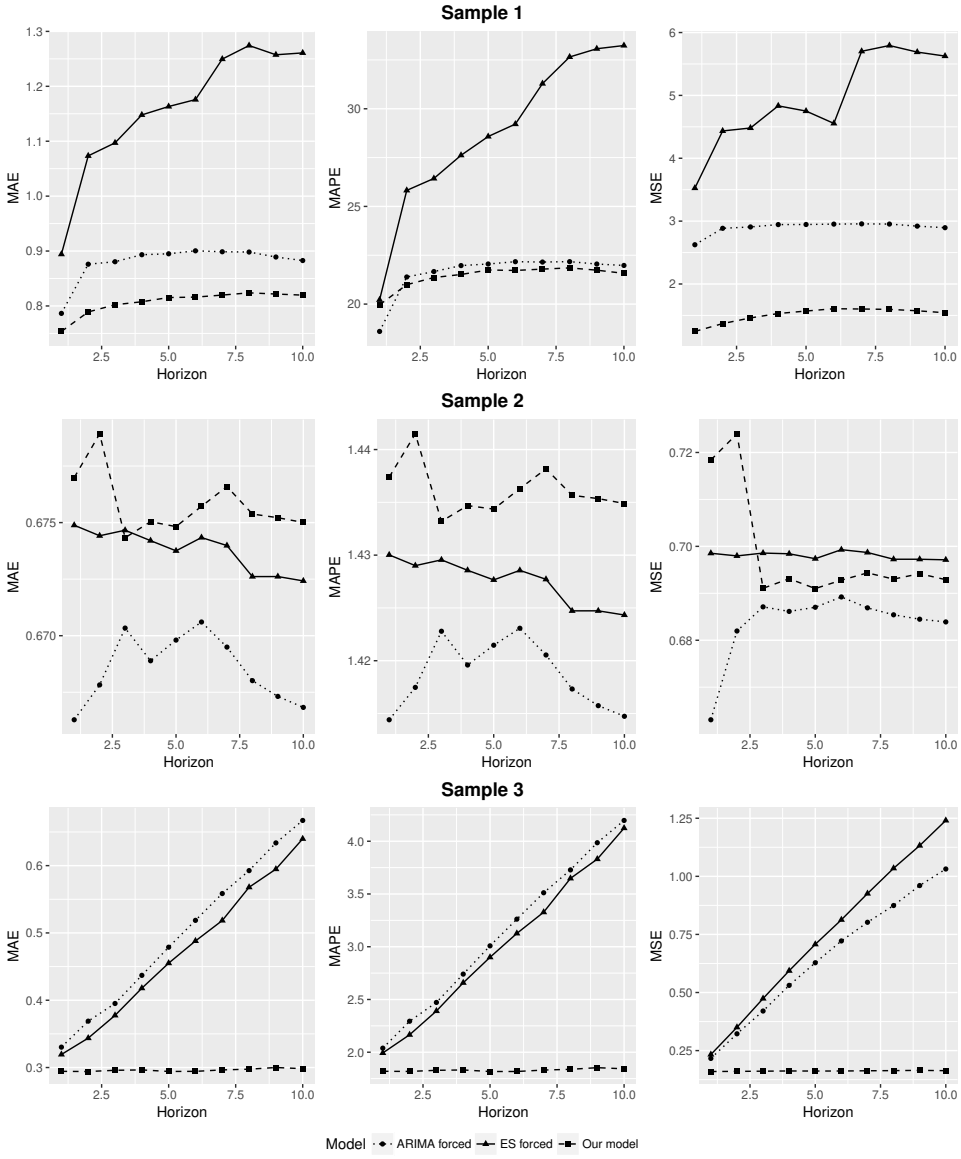



**Figure 2.9**: Point forecast performance comparison for the continuous time series.

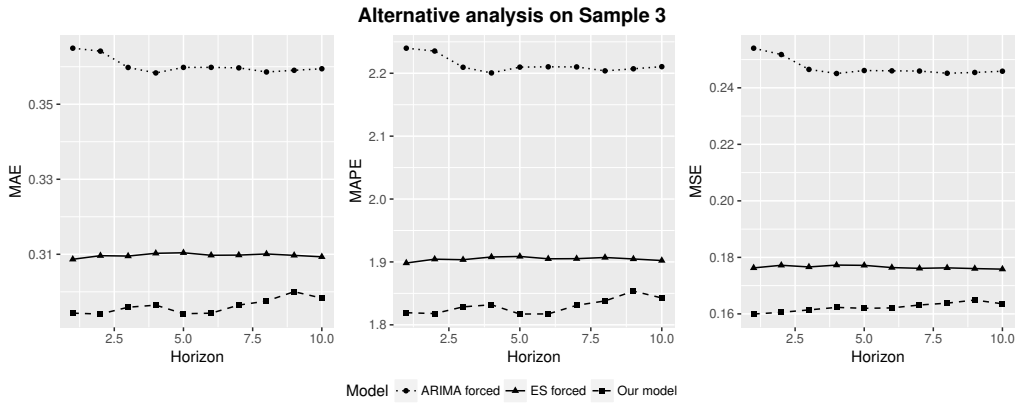



**Figure 2.10**: Point forecast performance comparison for sample 3 using *hand-crafted* features for ARIMA and ES models versus the automatic fitting of our approach (forcing seasonality and giving the period of this seasonal component).

Finally, we also computed the performance metrics for the fourth sample in Figure 2.11. However, neither ARIMA nor ES are prepared to deal with discrete time series. Therefore, here we only present the results for our method. As expected, our method

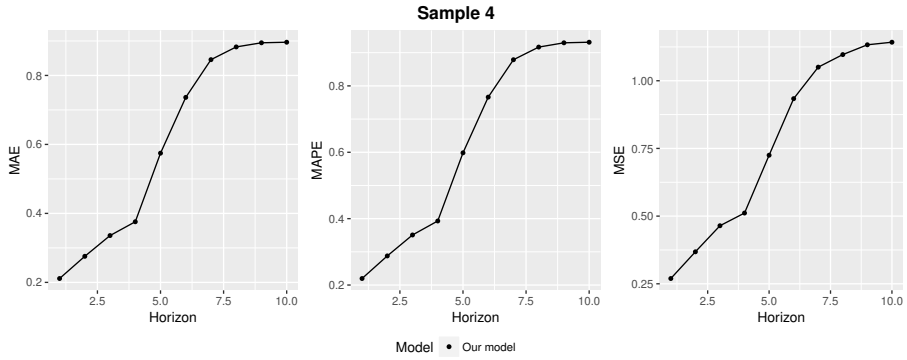



**Figure 2.11**: Point forecast performance comparison for the discrete time series.

performs better in short term predictions than in the long term ones. As mentioned, when dealing with long term forecasts in the discrete case, it may be more sensible to work with the stationary distribution described in (2.2.6).

A key aspect that differentiates our methodology from the rest is that it produces full predictive distributions, with reasonable uncertainty quantification, rather than point forecasts. Such distributions play an important role in anomaly detection as pointed out in Section 2.2. In order to perform probabilistic forecast evaluations, we plot in Figure 2.12 the empirical coverage against the width of the predictive distributions, for 1, 5 and 10 steps ahead predictions in each of the cases studied. As can be seen, our approach is working properly in the continuous case as the coverage curve is very close to the 45 degree line (ideal coverage).

In the discrete time series case, we observe over-coverage, specially for 5 and 10 steps ahead forecasts. This is a consequence of how the credible intervals are

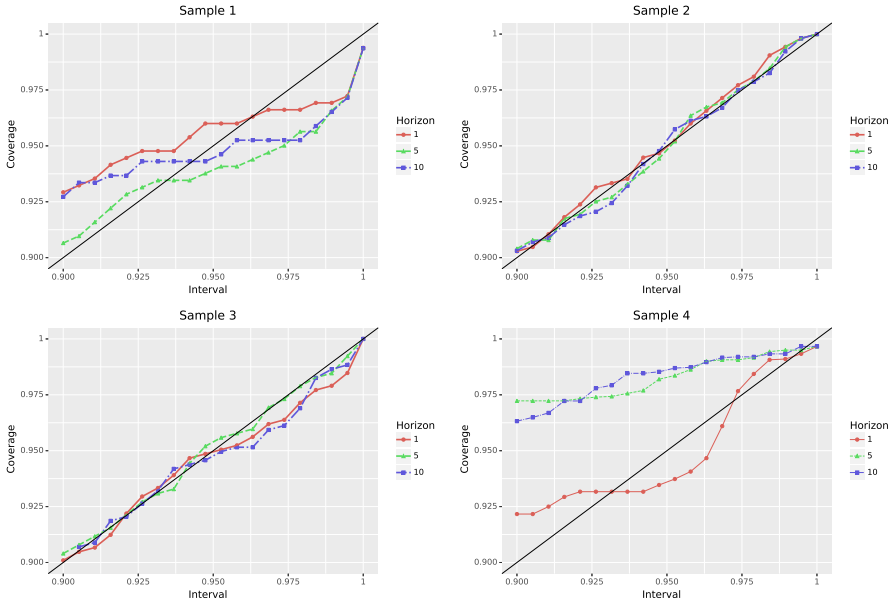



**Figure 2.12**: Empirical coverage for 1, 5 and 10 steps ahead predictive distributions.

constructed in such case (Algorithm 1). If we want to obtain an $\alpha$ probability predictive interval, we will keep on adding states to the interval until the probability of the system staying in any of those states is at least $\alpha$. However, we have point probability masses due to the discrete nature of the series and, therefore, the addition of a discrete number of these will make the real probability of the predictive interval bigger than $\alpha$ since we only stop adding states once $\alpha$ is surpassed to make sure that we cover *at least* such level.

## 2.5 Discussion

In this Chapter a reactive defense for time series security problems has been proposed. In particular, based on the features typical of network monitoring time series, we have provided a framework to identify safety and security issues within a large number of internet connected devices. The framework has been implemented and operates as part of a system controlling a network with more than three hundred thousand devices, even taking into account that each of them provides several very high-frequency time series.

The framework presented is very flexible when it comes to identify and model different behaviors that can be described in terms of basic components, as it happens in our application domain. This would allow for wide use through different environments beyond it, as we may use the same approach to monitor very distinct time series with intrinsic varying nature. Moreover, since the procedure just needs to store a few parameters for each time series, we can say that it is scalable, both in terms of memory and runtime. The framework is amenable to parallel processing, allowing to monitor different batches of series in different cores, thus reinforcing

the scalability request. In addition, thanks to our model identification procedure, our approach is able to work automatically without human supervision. Another interesting advantage refers to updating series with missing data, which happens relatively frequently in our domain. In presence of such missing values, the state carries no information and, therefore, the filtering distribution at such time is just the one-step-ahead predictive distribution of the previous step.

# Chapter 3

# Proactive Defences in Classification Problems

## 3.1 Introduction

Following objective O2, in this chapter we present novel proactive defenses for adversarial classification based on adversarial risk analysis (ARA) (Rios Insua et al. 2009). ARA is an emergent paradigm supporting decision makers who confront adversaries in problems with random consequences that depend on the actions of all participants. It provides one-sided prescriptive support to a decision maker maximizing her subjective expected utility by treating the adversaries' decisions as random variables. To forecast them, we model the adversaries' problems. However, our uncertainty about their probabilities and utilities is propagated leading to the corresponding random optimal adversarial decisions which provide the required forecasts. ARA operationalizes the Bayesian approach to games, Kadane and Larkey (1982) and Raiffa (1982), facilitating a procedure to predict adversarial decisions. Compared with standard game theoretic approaches, ARA does not assume the standard common knowledge hypothesis, according to which agents share information about utilities and probabilities.

As reviewed in Section 1.4 of Chapter 1, most proactive defenses for adversarial classification have been framed within a standard game theory approach, with strong common knowledge assumptions. In this chapter, we apply the decision theoretic pipeline proposed in Section 1.4.2 and 1.4.3 to adversarial classification (AC). Thus, we propose ACRA, an approach to robustify classification in adversarial settings based on ARA, which stems from the pioneering work by Dalvi et al. (2004), but avoids common knowledge assumptions prevalent in the literature.

The chapter is organized as follows: ACRA is presented in Section 2, followed by a simple numerical example in Section 3 aimed at showcasing concepts and illustrating robustness issues with adversarial examples. Section 4 presents computational enhancements illustrated with larger examples in Section 5. We end up with a discussion.

## 3.2 Adversarial Classification based on Adversarial Risk Analysis

In classification settings, an agent that we call classifier ($C$, she) may receive objects belonging to $k$ different types designated with a label $y = y_i$, $i = 1, ..., k$. Objects have features $x$ whose distribution depends on their type $y$. Classification problems can be broken down into two separate stages (Bishop 2006): an inference stage for learning $p_C(y|x)$, the classifier beliefs about the instance type given the features; and a decision stage in which the agent, based on these posterior probabilities, makes a class assignment decision $y_C$ perceiving some utility $u_C(y_C, y)$. The agent decides by maximizing expected utility, French and Rios Insua (2000). This problem may be formulated through an influence diagram, Jensen and Gatti (2012), as in Figure 3.1. As in BAIDs, Section 1.4.1, square nodes describe decisions; circle nodes, uncertainties; double nodes represent deterministic aspects; and finally, hexagonal nodes refer to the associated utilities. Arcs have the same interpretation as in Shachter (1986); those arcs pointing to decision nodes are dashed and represent information available when the corresponding decisions are made.

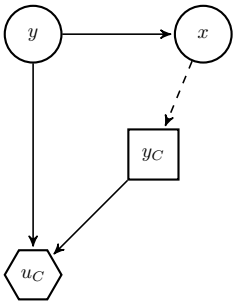



**Figure 3.1**: Classification as an Influence Diagram.

In adversarial settings, another agent called adversary ($A$, he) is present. Thus, the classification problem must be reformulated, leading to the BAID in Figure 1.7. In this chapter we focus on evasion attacks, defined to have influence just over operational data but not over training data. Thus, they only affect the decision stage of the classification problem. Within this stage, the adversary chooses an attack $a$ which, applied to the features $x$, leads to the perturbed data $x' = a(x)$ actually observed by $C$. A general transformation from $x$ to $x'$ will be designated $a_{x \to x'}$. The ID describing the classification problem must be augmented to incorporate adversarial decisions, leading to the bi-agent influence diagram in Figure 3.2. Notice that this coincides with the right part of the BAID in Figure 1.7. The adversary and classifier decisions are represented through nodes $a$ (chosen attack) and $y_C$ (classification choice), respectively. The impact of the data transformation over $x$ implemented by $A$ is described through node $x'$. The utilities of $A$ and $C$ are represented with nodes $u_A$ and $u_C$, respectively. Upon observing a particular $x'$, $C$ needs to determine the object class $y$. Her guess $y_C$, which we shall also denote $c(x')$, provides her with utility $u_C(y_C, y)$. As before, she aims at maximizing expected utility. However, $A$

also aims at maximizing his expected utility trying to confuse the classifier. His utility has the form $u_A(y_C, y, a)$, when $C$ says $y_C$, the actual label is $y$ and the attack is $a$, which has an implementation cost.

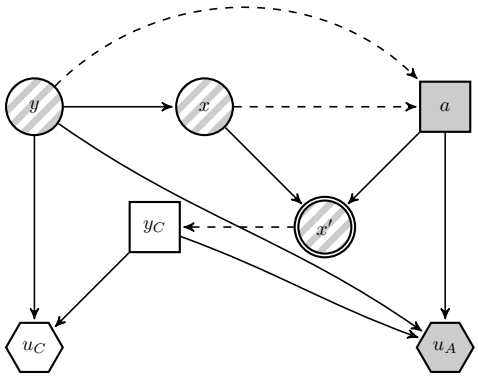



**Figure 3.2**: Adversarial classification as a Bi-agent Influence Diagram

In this chapter, we develop a framework to support $C$ in choosing her classification decision taking into account possible adversarial modifications of the data she observes. As in Dalvi et al. (2004), we study cases in which the attacker considers interesting for him instances belonging to the first $l$ classes (call them malicious), the other ones being irrelevant for him (innocent): the attacker is interested in modifying data associated with instances belonging to the first $l$ classes to make $C$ believe that they belong to the remaining ones. Thus, $A$ will not modify good instances. These types of attacks are denominated *integrity-violation attacks*, and are the most common ones in security scenarios. Finally, we restrict our attention to deterministic attacks, in the sense that their output is not random. Huang et al. (2011) and Barreno et al. (2006) provide taxonomies of attacks against classifiers.

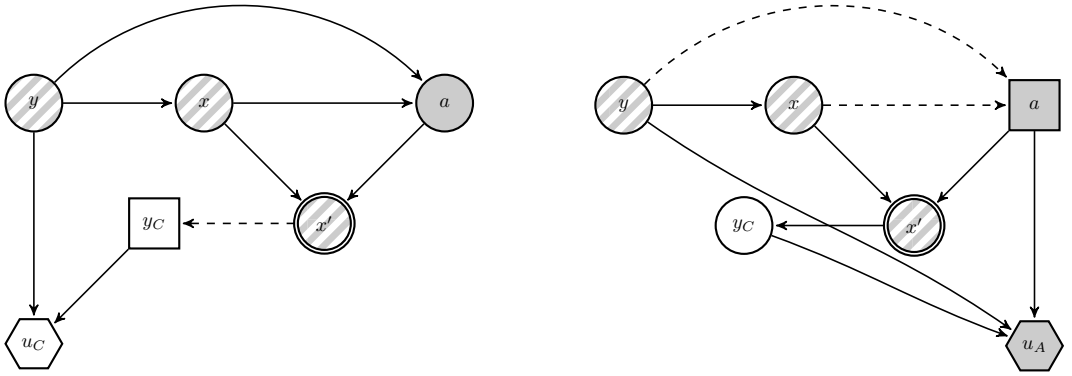
We shall need to forecast the attacker's actions to support $C$ in her decision making process. For that we consider his problem. As we lack common knowledge, we shall model our uncertainty about $A$'s beliefs and preferences and compute $A$'s random optimal attack, which provides the required forecasting distribution.

### 3.2.1 The classifier problem

We present first the classification problem faced by $C$ as a Bayesian game in Figure 3.3a, deduced from Figure 3.2. We formulate a decision problem for $C$ in which $A$'s decision appears as random to the classifier, since she does not know how the adversary will attack the data. Section 3.2.2 provides a procedure to estimate the corresponding probabilities making this approach operational.

Suppose for now that we are capable of assessing from the classifier:

1. $p_C(y)$, which describes her beliefs about the class distribution, with $\sum_{i=1}^{k} p_C(y_i) = 1$, and $p_C(y_i) \geq 0 \quad \forall i$.




(a) Classifier problem (b) Adversary problem

**Figure 3.3**: Influences Diagrams for the Classifier and the Adversary problems.

2. $p_C(x|y)$, modeling her beliefs about the feature distribution given the class, when $A$ is not present. Thus, we need $p_C(x|Y_i) \quad \forall i$. Since we focus on exploratory attacks, we can estimate $p_C(x|y)$ and $p_C(y)$ training a generative classifier, Bishop and Lasserre (2007), on data which is clean by assumption.

3. $p_C(x'|a, x)$, which models her beliefs about the transformation results. Since we consider only deterministic transformations, it will actually be the case that $p_C(x'|a, x) = I(x' = a(x))$, where $I$ is the indicator function.

4. $u_C(y_C, y)$, describing $C$'s utility when she classifies as $y_C$ an instance whose actual label is $y$.

5. $p_C(a|x, y)$, portraying $C$'s beliefs about $A$'s action, given that he receives and instance of class $y$ with features $x$.

In addition, we assume that $C$ is able to compute the set $\mathcal{A}(x)$ of possible attacks over a given instance $x$. When she observes $x'$, she could compute the set $\mathcal{X}' = \{x : a(x) = x' \text{ for some } a \in \mathcal{A}(x)\}$ of instances potentially leading to $x'$. She should then aim at choosing the class $y_C$ with maximum predictive expected utility. In our context, this means that she must find the class $c(x')$ such that

$$\begin{aligned} c(x') &= \operatorname*{argmax}_{y_C} \sum_{i=1}^{k} u_C(y_C, y_i) p_C(y_i|x') = \\ &= \operatorname*{argmax}_{y_C} \sum_{i=1}^{k} u_C(y_C, y_i) p_C(y_i) p_C(x'|y_i) = \\ &= \operatorname*{argmax}_{y_C} \sum_{i=1}^{k} u_C(y_C, y_i) p_C(y_i) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} p_C(x', x, a|y_i). \end{aligned}$$

For the second equation, we apply Bayes formula to compute $p(y_i|x')$, but ignore the denominator, which is irrelevant for optimization purposes. Then, we expand

the term $p(x'|y_i)$ taking into account the possible attacks. The presence of $A$ thus modifies $p_C(x'|y)$, preventing us from directly using the training set estimates of these elements. Therefore, we need to take into account $A$'s modifications through the probabilities $p_C(x', x, a|y)$. Furthermore, expanding the last expression and using the conditional independence assumptions implied by Figure 3.3a, we have

$$\begin{aligned} c(x') &= \underset{y_C}{\operatorname{argmax}} \sum_{i=1}^{k} \left[ u_C(y_C, y_i) p_C(y_i) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} p_C(x'|x, a, y_i) p_C(x, a|y_i) \right] = \\ &= \underset{y_C}{\operatorname{argmax}} \sum_{i=1}^{k} \left[ u_C(y_C, y_i) p_C(y_i) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} p_C(x'|x, a) p_C(a|x, y_i) p_C(x|y_i) \right]. \end{aligned}$$

Recalling now that we consider only integrity-violation attacks, we have $p_C(a|x, y_i) = I(a = id)$ for $i = l + 1, \ldots, k$, where $id$ stands for the identity attack leaving $x$ unchanged and $I$ is the indicator function. Then, taking also into account assumption 3, the problem to be solved by $C$ is

$$\begin{aligned} c(x') &= \underset{y_C}{\operatorname{argmax}} \left[ \sum_{i=1}^{l} u_C(y_C, y_i) p_C(y_i) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} I(x' = a(x)) p_C(a|x, y_i) p_C(x|y_i) \right. \\ &+ \left. \sum_{i=l+1}^{k} u_C(y_C, y_i) p_C(y_i) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} I(x' = a(x)) I(a = id) p_C(x|y_i) \right] = \\ &= \underset{y_C}{\operatorname{argmax}} \left[ \sum_{i=1}^{l} u_C(y_C, y_i) p_C(y_i) \sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x, y_i) p_C(x|y_i) \right. \\ &+ \left. \sum_{i=l+1}^{k} u_C(y_C, y_i) p_C(x'|y_i) p_C(y_i) \right], \end{aligned} \tag{3.2.1}$$

where $p_C(a_{x \to x'}|x, y_i)$ designates the probability that $A$ will execute an attack that transforms $x$ into $x'$, when he receives $(x, y = y_i)$, according to $C$.

Note that if the above mentioned game-theoretic common knowledge assumptions held, $C$ would be able to compute the set of instances that, with probability 1, $A$ would change to $x'$: we would know $A$'s beliefs and preferences and, therefore, would be able to compute the attacks he is actually implementing. Thus, we find out that for those potential attacks, $p_C(a_{x \to x'}|x, y_i)$ would be 1, and 0 for every other. Then, the model would just take into account instances with probability 1, ignoring the others. But common knowledge is not available, so we actually lack $A$'s beliefs and preferences. ACRA models our uncertainty around them. As we shall see, in doing so we enhance model robustness. Notice that in (3.2.1), we are summing $p_C(x|y_i)$ over all possible originating instances, weighing each element by $p_C(a_{x \to x'}|x, y_i)$, thus taking into account the uncertainty about $A$'s decision problem.

The ingredients 1-4 required in the analysis are standard in the decision analytic practice, Clemen and Reilly (2013). However, the fifth element $p_C(a_{x \to x'}|x, y)$, demands strategic thinking from $C$. To facilitate the corresponding forecast and make the approach operational, we consider $A$'s decision making process.

### 3.2.2 The attacker problem

We assume that $A$ aims at modifying $x$ to maximize his expected utility by making $C$ classify malicious instances as innocent. The decision problem faced by $A$ is presented in Figure 3.3b, deduced from Figure 3.2. In it, $C$'s decision appears as an uncertainty to $A$. Suppose for now that we have available from him:

1'. $p_A(x'|a, x)$, describing his beliefs about the transformation results. As with $C$, we make $p_A(x'|a, x) = I(x' = a(x))$.

2'. $u_A^{ci}(a) := u_A(y_c, y_i, a)$, which describes the utility that $A$ attains when $C$ says $y_c$, the actual label is $y_i$ and the attack is $a$. Note that this reflects certain attack implementation costs.

3'. $p_A(y_c|x' = a(x))$, which is the probability that $A$ concedes to $C$ saying that the instance belongs to class $y_c$, given that she observes the attacked data $x' = a(x)$. Let us call $p_c$ this probability. As the Attacker will typically be uncertain about it, we model this uncertainty with a density $f_A(p_c|a(x))$, with mean $p_A^c[a(x)]$.

In addition, we are assuming that $A$ just modifies malicious instances. Thus, when receiving an instance with features $x$ and label $y_i$ with $i = 1, \ldots, l$, among all attacks, $A$ would choose that maximizing his expected utility

$$\begin{aligned} a^*(x, y_i) &= \operatorname*{argmax}_a \sum_{c=1}^{k} \int \left[u_A^{ci}(a) p_c\right] f_A(p_c|a(x))\, \mathrm{d}p_c = \\ &= \sum_{c=1}^{k} u_A^{ci}(a) p_A^c[a(x)] \end{aligned} \tag{3.2.2}$$

However, the classifier does not know the involved utilities $u_A^{ci}(a)$ and expected probabilities $p_A^c[a(x)]$ from the adversary. Suppose we may model her uncertainty through random utility functions $U_A^{ci}(a)$ and a random expectations $P_A^c[a(x)]$. Then, we could solve for the random optimal attack, optimizing the random expected utility

$$A^*(x, y_i) = \operatorname*{argmax}_a \sum_{c=1}^{k} U_A^{ci}(a) P_A^c[a(x)],$$

and make $p_C(a_{x \to x'}|x, y_i) = Pr(A^*(x, y_i) = a_{x \to x'})$, assuming that the set of attacks is discrete.

We have therefore provided a way to approximate the remaining fifth ingredient in the classifier problem in Section 3.2.1, which is now operational. In general, to approximate it we use simulation drawing $K$ samples $\left(U_A^{ci,k}(a) P_A^{c,k}[a(x)]\right)$, $k = 1, \ldots, K$ from the random utilities and probabilities, finding

$$A_k^*(x, y_i) = \operatorname*{argmax}_a \sum_{c=1}^{k} U_A^{ci,k}(a) P_A^{c,k}[a(x)],$$

and estimating it through

$$\widehat{p_C}(a_{x\to x'} \mid x, +) = \frac{\#\{A_k^*(x,+) = a_{x\to x'}\}}{K}. \tag{3.2.3}$$

It is easy to prove that (3.2.3) converges almost surely to $p_C(a_{x\to x'}|x,+)$.

Of the required random elements, it is relatively easy to model the random utility $U_A^{ci}(a)$. One option would be to scale these utilities between 0 and 1 and use $U_{ci} \sim Beta(\alpha_{ci}, \beta_{ci})$. If information about the likely values of the utilities is available, we may assess them through appropriate $\alpha_{ci}$ and $\beta_{ci}$ values; if information about possible perceived rankings of the utilities is available, we may introduce them as constraints and sample by rejection. Another option, that allows us to explicitly model attacking costs, would typically include two components. The first one refers to $A$'s gain from $C$'s decision. If we adopt the notation $Y_{y_c,y_i}$ to represent the gain when $C$ decides $y_c$ and the actual label is $y_i$ we use: $-Y_{y_c,y_i} \sim Ga(\alpha_{ci}, \beta_{ci})$ when $c = 1, \ldots, l$, reflecting that, the gain obtained by $A$ when when $C$ classifies a truly malicious instance as malicious is negative. Similarly, $Y_{y_c,y_i} \sim Ga(\alpha_{ci}, \beta_{ci})$ when $c = l+1, \ldots, k$. The second component refers to the random cost $B$ of implementing an attack. Then, the gain of the attacker would be $Y_{y_c,y_i} - B$. Finally, assuming that $A$ is risk prone, French and Rios Insua (2000), the random utility could be computed as $U_A^{ci}(a) = \exp(\rho\,(Y_{y_c,y_i} - B))$ with, say, $\rho \sim U[a_1, a_2]$, $a_1 > 0$, reflecting $C$'s believes about $A$'s risk proneness coefficient.

On the other hand, modeling $P_A^c[a(x)]$, $A$'s (random) expected probability that $C$ declares an instance as belonging to class $y_c$ when she observes $x' = a(x)$, is more delicate. It entails strategic thinking as $C$ needs to understand his opponent's beliefs about what classification she will make when she observes $x'$. This could be the beginning of a hierarchy of decision making problems, as described in Rios and Rios Insua (2012) in a much simpler context. We illustrate here the initial stage of such hierarchy in our problem area. First, $A$ does not know the terms in the decision making problem (3.2.1) faced by the classifier. By assuming uncertainty over them through the random distributions $P_C^A(y_i)$, $P_C^A(x|y_i)$, $P_C^A(a_{x\to x'}|x, y_i)$ and utilities $U_C^A(y_C, y_i)$, he would get the corresponding random optimal decision replacing the incumbent elements in (3.2.1) to obtain $P_A^c[a(x)]$. However, observe that this requires the assessment of $P_C^A(a_{x\to x'}|x, y_i)$ (what $C$ believes that $A$ thinks about her beliefs concerning the action he would implement given the observed data) for which there is a strategic component, leading to the next stage in the announced hierarchy. One would typically stop at a level in which no more information is available. At that stage, we could use a non-informative prior over the involved probabilities and utilities.

For the first stage of this hierarchy, a relevant heuristic to assess $P_A^c[a(x)]$ may be based on the probability $Pr_C(c(a(x)) = y_c|a(x))$ that $C$ assigns to the object received being malicious assuming that she observed $x'$, with some uncertainty around it. Thus, we can model $P_A^c[a(x)]$ using a Beta distribution with parameters chosen in such a way that the mean is $Pr_C(c(a(x)) = y_c|a(x))$ for a given variance. Specifics on how to compute $Pr_C(c(a(x)) = y_c|a(x))$ would depend on the case at hand.

### 3.2.3 Algorithmic implementation

Once we have a Monte Carlo routine to estimate $p_C(a_{x\to x'}|x, y_i)$ as in Algorithm 4, which entails the availability of a routine to generate from the random utility function (Algorithm 3), we implement the scheme described above as in Algorithm 5, where $\hat{x}$ indicates estimate of $x$.

**Algorithm 3:** GENERATE $U_A^{ci}(a)$.

Generate $B^k \sim B$, $\rho^k \sim U[a_1, a_2]$.
Generate $Y^k_{y_c,y_i}$ from the corresponding Gamma distribution.
$U_A^{ci}(a) = \exp(\rho^k(Y^k_{y_C,y} - B^k))$.s
**return** $U_A^{ci}(a)$

**Algorithm 4:** ESTIMATE $p_C(a_{x\to x'}|x, y_i)$.

Compute $\mathcal{A}(x)$.
**for** $k = 1, 2, \ldots, K$ **do**
  **for** $a \in \mathcal{A}(x)$ **do**
    Generate $U_A^{ci,k}(a)$.
    Generate $P_A^{c,k}(a(x))$ from the corresponding Beta distribution.
    Compute $\psi^k(a) = \sum_{c=1}^k U_A^{ci,k}(a) P_A^{c,k}(a(x))$.
  **end**
  Compute $a_k^* = \text{argmax}_{a\in\mathcal{A}(x)}\, \psi^k(a)$.
**end**
Compute

$$\widehat{p}_C(a_{x\to x'} \,|\, x, +) = \frac{\#\{a_k^* = a_{x\to x'}\}}{K}.$$

**return** $\widehat{p}_C(a_{x\to x'} \,|\, x, y_i)$

## 3.3 A case study in spam detection

We illustrate the ACRA approach with a spam detection problem. We have data referring to $m$ emails characterized through the *bag-of-words* representation, Zhang et al. (2010): binary features indicate the presence (1) or not (0) of $n$ relevant words in a dictionary. Additionally, a label indicates whether the message is spam $y = +$ or not $y = -$. Thus, an email is assimilated with an $n$-dimensional vector $x$ of 0's or 1's, together with a label $y$. In this example, we consider only good word insertion (GWI) attacks, Dalvi et al. (2004). For simplicity, we illustrate the method in detail only when adding at most one word (1-GWI), but our method can be applied, with precautions discussed later, to the case of $k$ added words ($k$-GWI). An example asseses how the method works in the case 2-GWI in Section 3.4.3. 1-GWI entails converting at most one of the 0's of the originally received message into a 1.

**Algorithm 5:** ACRA scheme.

**Preprocessing**

Train a generative classifier to estimate $p_C(y)$ and $p_C(x|y)$, assuming that the training set has not been tainted.

**Operation**

Read $x'$ and estimate $\mathcal{X}'$.

ESTIMATE $p_C(a_{x\to x'}|x, y_i)$ for $i = 1, \ldots, l$ and for all $x \in \mathcal{X}'$.

Solve

$$c(x') = \operatorname*{argmax}_{y_C} \left[ \sum_{i=1}^{l} u_C(y_C, y_i)\widehat{p}_C(y_i) \sum_{x\in\mathcal{X}'} \widehat{p}_C(a_{x\to x'}|x, y_i)\widehat{p}_C(x|y_i) + \sum_{i=l+1}^{k} u_C(y_C, y_i)\widehat{p}_C(x'|y_i)\widehat{p}_C(y_i) \right].$$

**return** $c(x')$

Given a message $x = (x_1, x_2, .., x_n)$, with $x_i \in \{0, 1\}$, let us designate by $I(x)$ the set of indices such that $x_i = 0$. Then, the set of possible attacks in this case is $\mathcal{A}(x) = \{a_0 = id, a_i; \forall i \in I(x)\}$, where $a_i$ transforms the $i$-th 0 into a 1. In turn, given a message $x'$ received by $C$, we designate by $J(x')$ the indices of features with value 1 in $x'$. If we designate by $x'_j$ a message potentially leading to $x'$, derived by changing the $j$-th 1 in $x'$ with a 0, the set of possibly originating messages would be $\mathcal{X}' = \{x', x'_j; \forall j \in J(x')\}$.

### 3.3.1 Classifier elements

The elements required to solve the classifier problem, Section 3.2.1, include the utility function $u_C(y_C, y)$, which is standard, and the distributions $p_C(y)$ and $p_C(x|y)$, also standard if we just consider, as we do here, exploratory attacks. As we mentioned, we could use our favorite generative classifier to estimate them. Finally, $p_C(a_{x\to x'}|x, y)$ has a strategic component and we use ARA to approximate it.

### 3.3.2 Adversary elements

The adversary's random utilities follow the general arguments in Section 3.2.2. We also need to assess $P_A^+(a(x))$ (having this $P_A^- = 1 - P_A^+(a(x))$). We use the heuristic there proposed, with the following caveat. If the original label is $-$, the mail is innocent and the adversary does not change it, thus coinciding with the received one; we denote by $q_0 = p_C(x'|-)p_C(-)$ the probability of this event happening, according to $C$. If the original label is $+$, the mail is malicious, and $A$ might change it in an attempt to fool the classifier; according to $C$, the original message $x'_j$ happens with probability $q_j = p_C(x'_j|+)p_C(+), \forall j \in J(x')$. However, the adversary might decide not to attack even if the email is spam; according to $C$, this happens with probability

$q_{n+1} = p_C(x'|+)p_C(+)$. Then

$$r_a = \frac{\sum_{i \in J[a(x)]} q_i + q_{n+1}}{q_0 + \sum_{i \in J[a(x)]} q_i + q_{n+1}} \tag{3.3.1}$$

is the probability of $C$ believing that the observation $a(x)$ has label $+$, when she is aware of the presence of $A$. For such attack $a$, we could make $\delta_1^a/(\delta_1^a + \delta_2^a) = r_a$ and $(\delta_1^a \delta_2^a)/[(\delta_1^a + \delta_2^a)^2(\delta_1^a + \delta_2^a + 1)] = var$ and solve for $\delta_1^a$ and $\delta_2^a$ in order to get the parameters of the Beta distribution modeling $P_A^+(a(x))$.

### 3.3.3 Example

The above ingredients allow us to implement Algorithms 3 and 4, to generate from the random utility function and estimate $p_C(a_{x \to x'}|x, +)$, respectively. With this, we follow the scheme in Algorithm 5. We illustrate it in a simplified spam filtering problem[1], and compare it against the utility sensitive naive Bayes (NB) classifier, a standard non adversarial generative approach in this application area, Song et al. (2009). We use the Spambase Data Set from the UCI Machine Learning repository, Lichman (2013). It consists of 4601 emails, out of which 1813 are spam. For each email, the database contains information about 54 relevant words. The *bag-of-words* representation with binary features assimilates each email with a 54 dimensional vector $x$ of 0's and 1's. The dataset will be divided into training and hold-out test sets, respectively comprising 75% and 25% of the data.

We first train a utility sensitive NB classifier using the training data, unaltered by assumption. For comparison purposes, and in order to check utility robustness, we use four utility functions. One of them is the 0/1 utility, i.e. the utility is 1 if the instance is correctly classified and 0 otherwise. For the rest, we chose utility 1 for correctly classified instances and -1 for spam classified as legitimate. The penalty for classifying non-spam mail as spam was set, respectively, to -2, -5 and -10 in the other three cases. The corresponding NB classifier will serve for comparison as well as basis for the ACRA approach providing the required $\widehat{p}_C(y)$ and $\widehat{p}_C(x|y)$.

To compare ACRA with NB on tampered data, we simulate attacks over the instances in the test set. For this purpose, we solved the adversary problem (3.2.2) for each test email. Uncertainty in the adversary's utility function is not present from his point of view, thus, we fixed $-u_A(+,+,a) = 5$, $u_A(-,+,a) = 5$, $u_A(-,-,a) = u_A(+,-,a) = 0$. The cost for implementing an attack was set to $b = 0.5 \cdot d(a)$, where $d(a)$ is the number of word changes (0 or 1) associated with attack $a$. The risk proneness coefficient was set to $\rho = 0.5$. Finally, the adversary would have uncertainty about $p^A_{a(x)}$, as this quantity depends on the classifier decision. We test ACRA against a worst case adversary, who knows the true value of $p^A_{a(x)}$. With this, we attacked each test email to generate the attacked test set.

[1]For the sake of reproducibility, we provide the open source version of the code used for the examples at `https://github.com/roinaveiro/ACRA_spam_experiment.git`. The data is publicly available at `https://archive.ics.uci.edu/ml/datasets/spambase`.

From the classifier's point of view, the adversary's parameters were fixed at: $-U_A(+,+,a) \sim Ga(\alpha_1, \beta_1)$ with $E[-U_A(+,+,a)] = 5$ and $Var[-U_A(+,+,a)] = 0.01$, entailing $\alpha_1 = 2500$, $\beta_1 = 0.002$; $U_A(-,+,a) \sim Ga(\alpha_2, \beta_2)$, again with $\alpha_2 = 2500$, $\beta_2 = 0.002$; $U_A(-,-,a) = U_A(+,-,a) = \delta_0$. The random cost of implementing a particular attack $a$ was set to $B = d(a) \cdot \alpha$, where $d(a)$ is the number of word changes (0 or 1) associated with attack $a$, and $\alpha \sim U[0.4, 0.6]$. The random risk proneness coefficient was set to $\rho \sim U[0.4, 0.6]$. Observe that the attacker values are set as the means of the classifier distributions used to model them. We study later on how departures from the assumed adversary behaviour affects both ACRA and the corresponding game theoretic solution performance.

In order to obtain $P_A^+(a(x))$ for a given attack $a$, we need to generate from a beta distribution with mean $r = Pr_C(c(a(x)) = +|a(x))$, requiring its density to be concave in its support. Otherwise, we would be believing that the probability that $A$ concedes to $C$ deciding the instance $a(x)$ is malicious is peaked around 0 and 1 and low in between, which makes no sense in our context. Then, its variance must be bounded from above by $\Delta = \min\Big\{[r^2(1-r)]/(1+r), [r(1-r)^2]/(2-r)\Big\}$. We fix the adjustable variance *var* at $k\Delta$ with $k \in [0,1]$: the bigger $k$ is, the bigger $C$'s uncertainty will be about $A$'s behavior. We ran experiments for each $k \in \{0.01, 0.1, 0.2, \cdots, 0.9\}$. Finally, we fixed $K = 1000$, the Monte Carlo sample size in (3.2.3).

As performance metrics, we used the accuracy, utility, false positive (FPR) and false negative (FNR) rates, estimated via repeated hold-out validation over 100 repetitions, Kim (2009). We represent the results of the ACRA algorithm over the tampered test set with a solid line. The dashed line corresponds to the results of the utility sensitive NB on the attacked test, referred to as NB-Tainted. The error bars represent the standard deviation of each metric, also estimated through repeated hold-out validation. In addition, as a benchmark, we show the results of the utility sensitive NB over the original, untampered test set with a dotted line. We refer to these as NB-Plain. Obviously, NB-Plain and NB-Tainted metrics do not depend on $k$.

Figures 3.4 and 3.5 respectively present the average accuracies and utilities for various values of $k$ and the four utility models. Observe first that the presence of an adversary considerably degrades NB performance both in accuracy and average utility, as NB-Plain is consistently above NB-Tainted. This one is still correctly classifying the same proportion of non spam as NB-Plain, as such emails have not been attacked. However, NB-Tainted is not able to identify a large proportion of attacked spam emails. Consequently, as we increase the cost of misclassifying non-spam, reducing the relative importance of misclassifying spam, the performance of NB clearly degrades. This lack of robustness to attacks confirms the need to take into account the presence of adversaries.

In contrast, ACRA is robust to attacks and identifies most of the spam. Its overall accuracy is above 0.9, thus identifying most non-spam emails. Observe though that ACRA degrades as $k$ grows: the bigger $k$ is, the less precise the knowledge that $C$ has about $A$ and the classifier performance will degrade. One of our contributions is

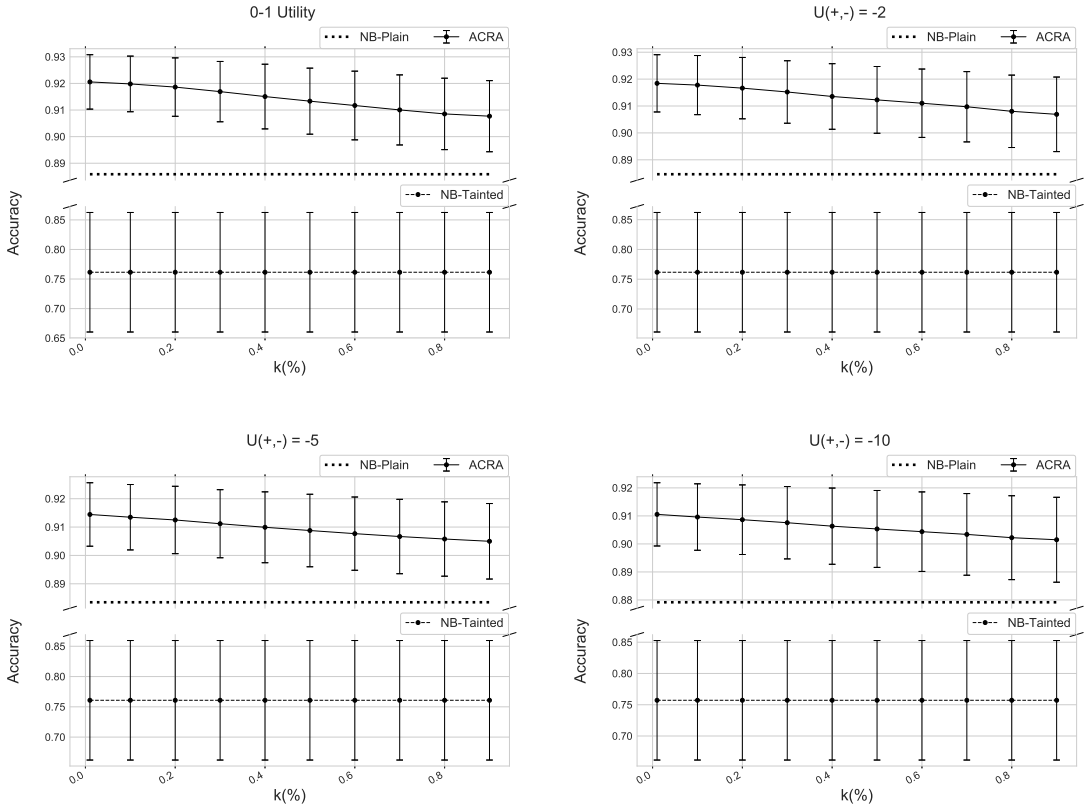



**Figure 3.4**: Average accuracy versus $k$ for different utility models.

providing parameters that may be tuned to adapt to the knowledge that the classifier could have about her opponent.

Very interestingly, note that in Figures 3.4 and 3.5 ACRA beats NB-Plain in both accuracy and utility. This effect has been observed by Dalvi et al. (2004) and Goodfellow et al. (2015b) for different algorithms and different application areas. The latter argues that taking into account the presence of an adversary has an effect similar to that of a regularizer, being able to improve the original accuracy of the base algorithm and making it more robust.

To better understand these results, we plotted FPR and FNR in Figures 3.6 and 3.7, respectively. FPR coincides for NB-Plain and NB-Tainted as the adversary is not modifying innocent instances. Both FPR and FNR grow with $k$ for ACRA: the more the classifier knows about the adversary strategy, the better she protects, and lower FPR and FNR are attained. In addition, an increase of $u_C(+,-)$ raises the cost of false positives, reducing FPR at the expense of increasing FNR.

Regarding the conceptual comparison of different algorithms, observe that false negatives undermine the performance of NB on tampered data. In contrast, ACRA seems more robust, presenting smaller FNR than NB-Tainted. ACRA has also significantly lower FPR than NB, causing the overall performance to raise up. The reason for this is that the adversary is very unlikely to apply the identity attack

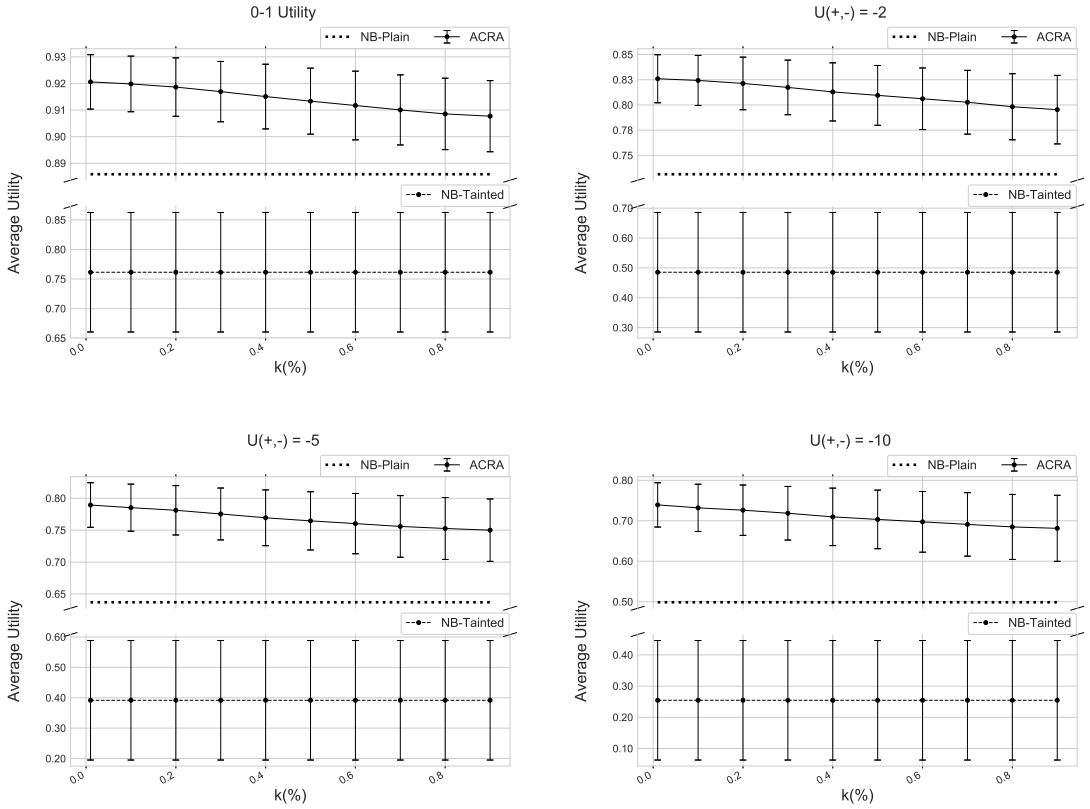



**Figure 3.5**: Average attained utility versus $k$ for different utility models.

to a spam, as the cost difference between such attack and 1-GWI attacks is small in terms of utility gain. Then, for a legitimate email that NB classifies as positive, i.e. has high $p_C(x|+)$, ACRA will give a very low weight to $p_C(x|+)$, thus reducing the probability of classifying such email as spam. Reducing FPR is crucial in spam detection, as filtering out a non-spam is typically more undesirable than letting spam reach the user. Interestingly enough, ACRA also has lower FNR than NB-Plain, specially for low values of $k$, although this improvement is not as remarkable as that for FPR. Both effects together produce ACRA to outperform utility sensitive NB both in tainted and untainted data: we enhance the classifier performance by taking advantage of the information we may have about the adversary, a core idea underlying ACRA.

### 3.3.4 Robustness

We have tested the ACRA algorithm against an adversary whose parameters were fixed to the expected values of the distributions assumed by the classifier. It is natural to ask how departures from such assumptions about the adversary's behaviour would affect performance. In this section, we compare the robustness of ACRA and the corresponding game theoretic solution, to departures from the assumed value of

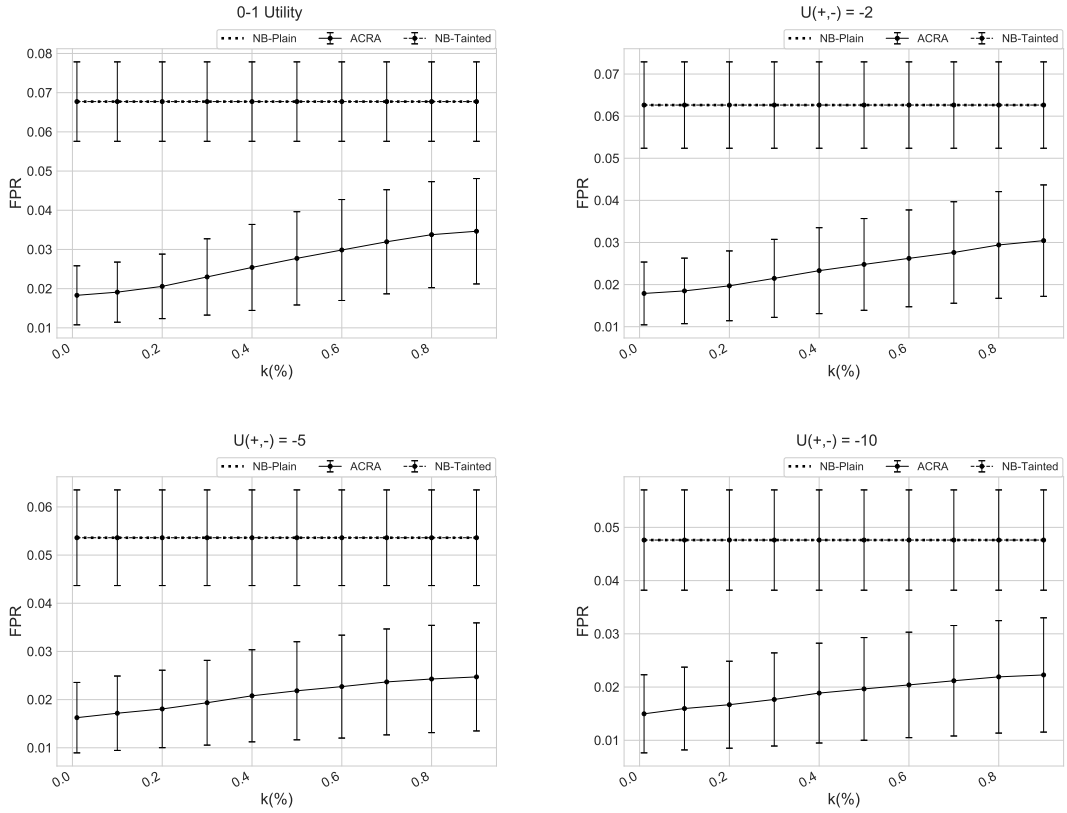



**Figure 3.6**: Average false positive rate versus $k$ for different utility models.

$p_A^+(a(x))$, the expected value of the probability that $A$ concedes to $C$ saying that the instance $a(x)$ is malicious. As we discussed above, this is clearly the most challenging assessment.

The game theoretic solution assumes common knowledge. For the case of $p_A^+(a(x))$ this means that this quantity is fixed at a certain value, shared by both $A$ and $C$. Thus $C$, when modelling $A$, assumes that he is using a certain value of $p_A^+(a(x))$ that coincides with the one that he is actually using. For robustness purposes, we attacked the test set solving the attacker problem (3.2.2) for each test instance $x$, but using this time perturbed values of $p_A^+(a(x))$ around the one assumed by $C$. To do so, we sample $p_A^+(a(x))$ from a beta distribution centered at the assumed value, with variance $k_A\Delta$, where $k_A$ is the proportion of the maximum allowed variance $\Delta$ (again, we require the density to be concave in its support). We compare performance against this attacker of both the ACRA solution, which models uncertainty on $p_A^+(a(x))$ by placing a beta distribution centered at the assumed value of $p_A^+(a(x))$ and variance $k\Delta$, and the game theoretic solution for which $p_A^+(a(x))$ is fixed at its assumed value. We performed experiments for different values of $k$ and $k_A$. Results for the percentage accuracy gain of ACRA, i.e. ACRA percentage accuracy minus game theory percentage accuracy, are presented in Figure 3.8a. As can be seen, if the attacker behaves closely to the common knowledge assumptions (i.e. low $k_A$),

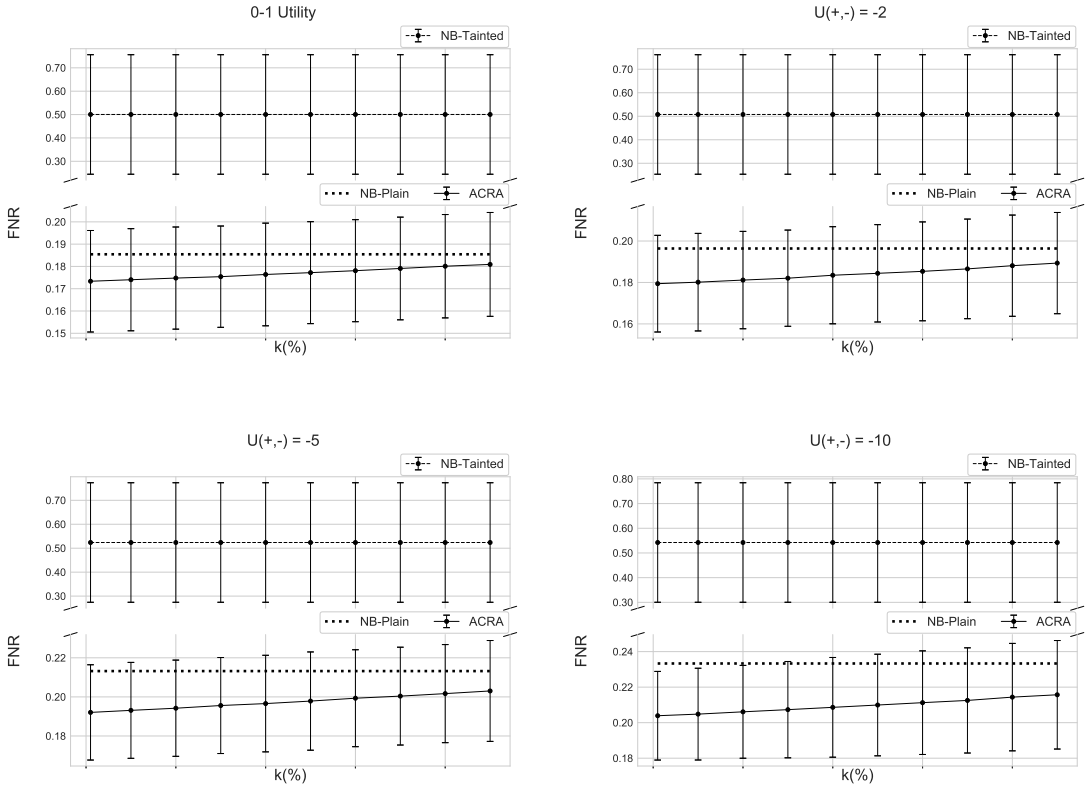
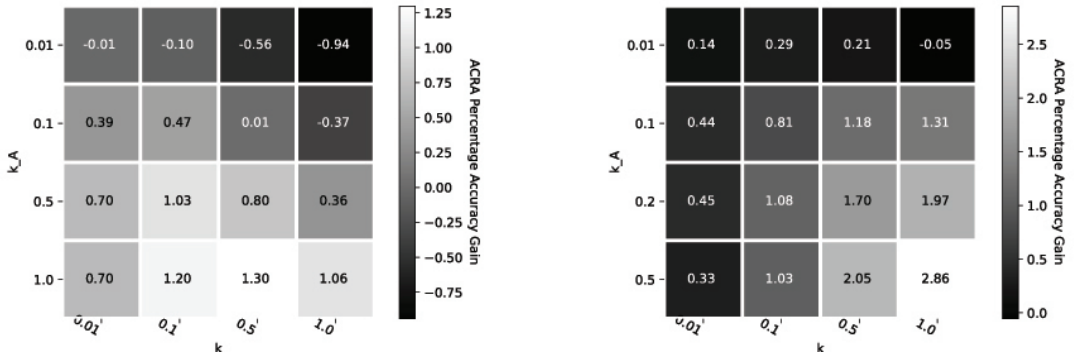



**Figure 3.7**: Average false negative rate versus $k$ for different utility models.

then the game theoretic solution and ACRA with low variance behave similarly. If in this case, we increase the variance assumed by ACRA, then its performance degrades, as it is overestimating the uncertainty. If the variance of the distribution that the adversary puts in $p_A^+(a(x))$ is high (i.e. high $k_A$), then big deviations from the common knowledge value of $p_A^+(a(x))$ are more likely, thus degrading the performance of the game theoretic solution. Nevertheless, in this case ACRA remains robust to these perturbations, thanks to accounting for uncertainty on the adversary's value of $p_A^+(a(x))$. In addition, although not shown in the Figure, ACRA's accuracy was above 0.89 for every pair $(k, k_A)$, beating always the NB algorithm. One could argue that this comparison is not sufficiently fair, as we are sampling the adversary's values of $p_A^+(a(x))$ from a beta distribution, that is the one assumed by ACRA. Figure 3.8b shows the results of an alternative experiment. In this case, to perturb the values of $p_A^+(a(x))$ of each test instance, we add a number, uniformly distributed in the interval $[-k_A, k_A]$, to the common knowledge value. As can be seen, in this case ACRA is again more robust to this imprecision in the attacker's model, thus providing better generalization. In addition, ACRA's accuracy was not excessively damaged despite using perturbed values of $p_A^+(a(x))$, being above 0.87 for every pair $(k, k_A)$.




(a) (b)

**Figure 3.8**: ACRA accuracy gain with respect to the game theoretic solution.

## 3.4 Computational issues

The raw version of ACRA presented above may turn out to be extremely heavy computationally in some application domains in which little assumptions about the adversary behaviour are made. In this section we asses ACRA computationally and then propose several solutions. For the sake of illustration, we just discuss the binary case with possible classes $y = +$ (malicious) and $y = -$ (innocent).

### 3.4.1 Computational assessment

Note first that if no assumptions about the attacks are made, the sets $\mathcal{A}(x)$, and consequently $\mathcal{X}'$, could grow rapidly. The size of these sets strongly depends on the application domain. For instance, in spam detection the number of possible adversarial manipulations is $2^n$, where $n$ is the number of words considered by $C$ to undertake the classification.

In fact, the size of $\mathcal{X}'$ affects critically the number of computations. Notice that in order to classify a given instance $x'$, we need to estimate $p_C(a_{x\to x'}|x, +)$ for each $x \in \mathcal{X}'$ to compute the summation in (3.2.1). In addition, estimating each $p_C(a_{x\to x'}|x, +)$ requires a MC simulation with size $K$, see Algorithm 4. Thus, if $G$ represents the computational cost of each MC sample, the overall cost to classify one instance would be $K \cdot |\mathcal{X}'| \cdot G$. Moreover, $G$ could depend on the size of $\mathcal{A}(x)$.

We study now how to reduce the computational burden of ACRA by means of reducing the size of $\mathcal{X}'$ and the MC size $K$ without affecting performance too much.

### 3.4.2 Computational enhancements

One possibility to reduce $|\mathcal{X}'|$ could be to use application-specific information to restrict the class of possible attacks. For instance, if we consider only $k$-GWI attacks in spam detection, the size of $\mathcal{X}'$ is reduced to $\mathcal{O}(n^k)$. In addition, case-specific constraints about the adversary behaviour could be used to achieve a greater decrease.

For example, in GWI we may reduce $n$ assuming that some words cannot be modified by the adversary, or limit the number of words inserted, either explicitly, or implicitly through penalising the insertion of additional words.

Apart from using application-specific information, we present now several general suggestions which can be used to alleviate the computational burden. Note first that the optimisation problem (3.2.1) may be reformulated as setting $c(x') = +$ if and only if $\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x,+)p_C(x|+) > t$, where

$$t = \frac{\Big[u_C(-,-) - u_C(+,-)\Big]p_C(x'|-)p_C(-)}{\Big[u_C(+,+) - u_C(-,+)\Big]p_C(+)}.$$

Rather than going through the whole $\mathcal{X}'$, we can approximate the left hand side summation through Monte Carlo. Should $\{x_n\}$ be a sample of size $N$ from $p_C(x|+)$, with $N$ lower than the cardinality of $\mathcal{X}'$, the condition would be approximated through

$$I = \frac{1}{N}\sum_{n=1}^{N} p_C(a_{x_n\to x'}|x_n,+)I(x_n \in \mathcal{X}') > t. \tag{3.4.1}$$

A potential problem with this approach is that $p_C(x|+)$ for $x \in \mathcal{X}'$ is generally small and a standard MC sample might contain few points in $\mathcal{X}'$. We could use importance sampling, Owen and Zhou (2000), to mitigate this issue for example using the restriction of $p(x|+)$ to $\mathcal{X}'$ as importance distribution. Let $\tilde{p}(x|+)$ be the probability distribution defined by

$$\tilde{p}(x|+) = \frac{p(x|+)}{Q}\cdot I(x \in \mathcal{X}')$$

with $Q = \sum_{x\in\mathcal{X}'} p(x|+)$ . Then

$$\begin{aligned}\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x,+)p_C(x|+) &= \sum_{x\in\mathcal{X}'} \frac{p_C(a_{x\to x'}|x,+)p_C(x|+)}{\tilde{p}(x|+)}\tilde{p}(x|+)\\ &= Q\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x,+)\tilde{p}(x|+).\end{aligned}$$

Now, if $\{x_n\}$ is a sample of size $N$ from $\tilde{p}(x|+)$, we could approximate our target quantity $I$ by

$$\tilde{I} = \frac{Q}{N}\sum_{n=1}^{N} p_C(a_{x_n\to x'}|x_n,+). \tag{3.4.2}$$

We should take into account, though, the inherent uncertainty in the above MC approximations when checking inequality (3.4.1). An estimate $\tilde{\Delta}$ of its standard deviation would be

$$\tilde{\Delta} = \sqrt{\frac{\sum_{i=1}^{N} Q^2 p_C^2(a_{x_i\to x'}|x_i,+) - N\tilde{I}^2}{N-1}}.$$

We would then declare $c(x') = +$ if $\tilde{I} - 2\tilde{\Delta} > t$. Observe that we could test this condition sequentially, before reaching the maximum size $N$ allowed by our computational budget. By making $\tilde{I}_m$ and $\tilde{\Delta}_m$ depend on the sample size $m$, we would check sequentially whether

$$\tilde{I}_m - 2\tilde{\Delta}_m > t, \tag{3.4.3}$$

and stop if verified, with $\tilde{I}_m$ and $\tilde{\Delta}_m$ defined sequentially, since

$$\begin{aligned} \tilde{I}_m &= \frac{(m-1)\tilde{I}_{m-1} + p_C(a_{x_m \to x'}|x_m, +)}{m}, \\ \tilde{\Delta}_m^2 &= \frac{(m-2)\tilde{\Delta}_{m-1}^2 + Q^2 p_C^2(a_{x_m \to x'}|x_m, +) + (m-1)\tilde{I}_{m-1}^2 - m\tilde{I}_m^2}{m-1}. \end{aligned}$$

Note that with the proposed enhancement, the number of computations is $K \cdot N$ rather than $K \cdot |\mathcal{X}'|$ , where $N$ is the chosen sample size. Thus, we manage to eliminate the dependence on $|\mathcal{X}'|$. $N$ should be fixed to trade off between accuracy and speed.

A complementary possibility to speed up computations is to use a relatively small MC size $K$ for estimating $p_C(a_{x \to x'}|x, +)$. To mitigate getting null probabilities, we could adopt a Dirichlet-multinomial model with non-informative prior $Dir(1, 1, ..., 1)$ over the probabilities of the attacks in $\mathcal{A}(x)$ and approximate such probability through

$$\widehat{p}_C(a_{x \to x'} \,|\, x, +) = \frac{\#\{a_k^* = a_{x \to x'}\} + 1}{K + |\mathcal{A}(x)|}.$$

Moreover, we could use a regression metamodel, Kleijnen (1992), based on approximating in detail $p_C(a_{x \to x'}|x, +)$ at some pairs $(x, x')$, as allowed by our computational budget, fit a regression model $\psi(x, x')$ to $(x, x', \widehat{p}_C(a_{x \to x'}|x, +))$ and use it to replace $p_C(a_{x \to x'}|x, +)$ in the above expressions.

Last, but not least, ACRA is amenable to parallelization in at least two respects. Observe first that the terms in summations (3.4.1) or (3.4.2) may be evaluated independently. To improve performance, we may compute batches of those terms in parallel by running different processes at different nodes of a multi-core cluster. Whenever we use the sequential approach in (3.4.3), this parallelization strategy would require a master node that checks such condition periodically when the computation of any batch of terms finishes in the corresponding worker node. Finally, recall that computing terms in summations (6) or (3.4.2) entails a simulation. We could accelerate the computation of each simulation by running in parallel different processes for different batches of MC samples. Both parallelization strategies could be combined by sending different simulations to different nodes in the cluster, and parallelizing each simulation within the cores of each node.

The combination of the above approaches alleviates tremendously the computational burden and largely makes ACRA computationally feasible as we show next.

### 3.4.3 Application

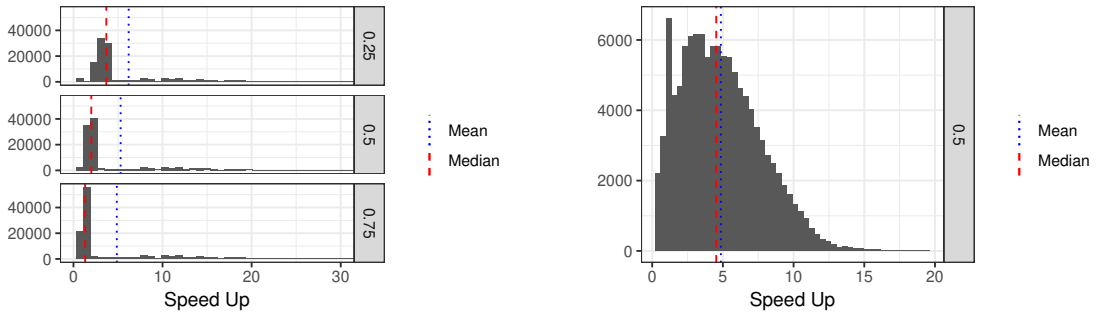
We illustrate several of the proposed enhancements with the example in Section 3.3.3. We start by testing the ACRA framework using MC simulation (MC ACRA) with importance sampling and the sequential strategy (3.4.3) against the raw algorithm (ACRA). We tried different MC sample sizes $N$, measuring them as proportions of the cardinal of $\mathcal{X}'$: e.g. an MC sample size of 0.5 corresponds to considering $N = |\mathcal{X}'|/2$ values. We fixed $K = 1000$, the MC size in the ESTIMATE $p_C(a_{x \to x'}|x, +)$ function, the adjustable var parameter $k$ to 0.1, and used the 0/1 utility. Table 3.1 shows

| | Size | Accuracy | FPR | FNR |
|---|---|---|---|---|
| **ACRA** | 1.00 | $0.919 \pm 0.010$ | $0.019 \pm 0.008$ | $0.177 \pm 0.022$ |
| **MC ACRA** | 0.75 | $0.912 \pm 0.012$ | $0.032 \pm 0.009$ | $0.174 \pm 0.023$ |
| **MC ACRA** | 0.50 | $0.905 \pm 0.016$ | $0.027 \pm 0.009$ | $0.199 \pm 0.032$ |
| **MC ACRA** | 0.25 | $0.885 \pm 0.029$ | $0.021 \pm 0.007$ | $0.260 \pm 0.067$ |
| **MC ACRA** | 0.10 | $0.841 \pm 0.047$ | $0.016 \pm 0.005$ | $0.370 \pm 0.120$ |
| **NB-Tainted** | - | $0.761 \pm 0.101$ | $0.680 \pm 0.100$ | $0.500 \pm 0.250$ |

**Table 3.1**: Comparison between MC ACRA, raw ACRA and NB.

the average performance metrics, with standard deviations, of the algorithms over 100 experiments. Note that as the sample size $N$ increases, accuracy also increases. Nevertheless, we get fairly good results for relatively small sample sizes. For example, with just a 0.1 sample size we manage to beat NB in accuracy as well as in FPR and FNR. Considering a 0.5 sample size, we almost recover the original performance levels.

To compare execution times, we computed the speed-up (quotient between execution times of ACRA and MC ACRA over all 100 experiments). Figure 3.9a presents the speed-up histogram, Table 3.2 shows mean and median speed-ups for the MC sizes (0.25, 0.5, 0.75). As expected, the median is close to the inverse of




(a) (b)

**Figure 3.9**: Speed-up histograms.

the MC size, e.g. when size is 0.5, MC ACRA performs approximately twice faster

than ACRA. Nevertheless, the speed-up distributions (Figure 3.9a) are skewed to the right suggesting that MC ACRA performs much faster on average. This is due to the sequential rule (3.4.3): for some instances, such condition is reached in a few iterations and, consequently, over those instances MC ACRA performs much faster than ACRA.

| Size | Mean | Median |
|---|---|---|
| 0.25 | 6.20 | 3.69 |
| 0.50 | 5.30 | 2.00 |
| 0.75 | 4.86 | 1.31 |

**Table 3.2**: Mean and median speed-ups.

We have also tested the first parallelization approach in Section 3.4, computing in parallel the terms in the MC approximation (3.4.2). We used a 16 core processor for this purpose. We performed 100 experiments fixing the variance parameter $k = 0.1$, MC size to 0.5 and 0/1 utility. The histogram of speed-ups is in Figure 3.9b. In this case, both the mean (4.856) and median (4.530) are close. We do not use the sequential approach (3.4.3) and consequently, extreme values do not occur. Nevertheless, we obtain a huge improvement in time performance, almost 5 times faster both in mean and median.

The combination of the above approaches induces considerable improvements rendering ACRA largely feasible, as we illustrate with the results of an experiment under 2-GWI attacks with different databases[2] in Table 3.3. As in previous examples, we report averages over 100 experiments performed under different train-test splits. Observe that MC ACRA with size 0.5 consistently beats utility sensitive NB.

| | Dataset | Accuracy | FPR | FNR |
|---|---|---|---|---|
| **MC 0.5 ACRA** | UCI | $0.904 \pm 0.012$ | $0.037 \pm 0.007$ | $0.187 \pm 0.023$ |
| **NB-Tainted** | UCI | $0.724 \pm 0.088$ | $0.066 \pm 0.008$ | $0.601 \pm 0.022$ |
| **MC 0.5 ACRA** | Enron-Spam | $0.824 \pm 0.017$ | $0.132 \pm 0.012$ | $0.305 \pm 0.073$ |
| **NB-Tainted** | Enron-Spam | $0.534 \pm 0.011$ | $0.283 \pm 0.013$ | $1.000 \pm 0.000$ |
| **MC 0.5 ACRA** | Ling-Spam | $0.958 \pm 0.008$ | $0.039 \pm 0.001$ | $0.057 \pm 0.030$ |
| **NB-Tainted** | Ling-Spam | $0.800 \pm 0.016$ | $0.040 \pm 0.001$ | $1.000 \pm 0.000$ |

**Table 3.3**: Comparison between size 0.5 MC ACRA and NB under 2-GWI attacks.

[2]Besides the UCI Spam Data Set, we used the Enron-Spam Data Set at `https://www.cs.cmu.edu/~enron` and the Ling-Spam Data Set at `http://csmining.org/index.php/ling-spam-datasets.html`

## 3.5 Dealing with discriminative classifiers

In Section 3.2 we have provided an ARA based framework for Adversarial Classification. Note though, that such framework could deal just with generative classifiers, that is, classifiers that model explicitly the feature distribution given the class $p_C(x|y)$. In this section, we present a general framework that may be used with both discriminative and generative classifiers.

Assume for a moment that the classifier knows the attack that she has suffered and that it is invertible, in the sense that we may recover, when convenient, the original $x$, designated $a^{-1}_{x\to x'}(x')$, being $x'$ the observed instance. Then, rather than classifying based on $\text{argmax}_{y_C} \sum_{i=1}^{k} u_C(y_C, y_i)p_C(y_i|x')$, as an adversary unaware classifier would do, she should classify based on

$$\arg\max_{y_C} \sum_{i=1}^{k} u_C(y_C, y_i)p_C(y_i|x = a^{-1}_{x\to x'}(x')).$$

However, we do not know the attack $a$, neither, more generally, the originating $x$.

Suppose we model our uncertainty about the origin $x$ of the attack through a distribution $p_C(x|x')$ with support over the set $\mathcal{X}'$ of reasonable originating features $x$. Then, the expected utility that the classifier would get for her classification decision $y_C$ would be

$$\begin{aligned}\psi(y_C) &= \int_{\mathcal{X}'} \left( \sum_{i=1}^{k} u_C(y_C, y_i)p_C(y_i|x = a^{-1}_{x\to x'}(x') \right) p_C(x|x')dx \\ &= \sum_{i=1}^{k} u_C(y_C, y_i) \left[ \int_{\mathcal{X}'} p_C(y_i|x = a^{-1}_{x\to x'}(x'))p_C(x|x')dx \right],\end{aligned}$$

then having to solve

$$\arg\max_{y_C} \psi(y_C). \tag{3.5.1}$$

Here, evaluating $p_C(x|x')$ is delicate, as it demands strategic thinking. In Section 3.2 we provided a mechanism to evaluate $p_C(a_{x\to x'}|x, y_i)$, the strategic element in that case. In this Section we provide an alternative strategy that requires sampling from $p_C(x|x')$ instead of directly evaluating this probability. Thus, if we could sample from this distribution we could approximate the expected utilities by Monte Carlo (MC) using a sample $\{x_n\}_{n=1}^{N}$ from $p_C(x|x')$ so that

$$\widehat{\psi}(y_C) = \frac{1}{N} \sum_{i=1}^{k} u(y_C, y_i) \left[ \sum_{n=1}^{N} p(y_i|x_n) \right]. \tag{3.5.2}$$

Algorithm 6 summarizes the general procedure that we later specify.

For this approach to be operational, we need to be able to estimate $\mathcal{X}'$ and $p_C(x|x')$ or, at least, sample from this distribution. Assuming that we can define a metric $\lambda$ in the feature space, a first heuristic would be to define the set $\mathcal{X}'$ of reasonable originating features as those $x$ such that $\lambda(x, x') < \rho$ for a certain threshold

**Algorithm 6:** General ARA procedure for Adversarial Classification

**Input:** $N$, training data.
**Output:** A classification decision $c(x')$.
**Training**
Based on the training data compute $p_C(y_i|x)$ for all $i$.
**End Training**
**Operation**
Read instance $x'$
Draw sample $\{x_n\}_{n=1}^N$ from $p_C(x|x')$.

Find $c(x') = \text{argmax}_{y_C} \frac{1}{N} \sum_{i=1}^k \left(u_C(y_C, y_i) \left[\sum_{n=1}^N p_C(y_i|x_n)\right]\right)$

**End Operation**
**Return** $c(x')$

$\rho$. We could then take $p_C(x|x')$ as a uniform distribution over $\mathcal{X}'$. Alternatively, we could make $p_C(x|x') = \frac{h}{\lambda(x,x')}$, where $\frac{1}{h} = \sum_{x\in\mathcal{X}'} \frac{1}{\lambda(x,x')}$, ignoring $x'$ as possible origin. These heuristics formalize the fact that changing instances entails some cost for the adversary that probably increases with the number of features changed. However, as shown in Ríos Insua et al. (2020), better forecasts are typically attained if we explicitly model the attacker's behavior using the information available about him, as we do next.

### 3.5.1 An Approximate Bayesian Computation sampling approach

An approach to sample from $p_C(x|x')$ that leverages information available about the attacker is now discussed. We call it AB-ACRA, being based on approximate Bayesian computation (ABC), Csilléry et al. (2010). As basic ingredients, it requires us to be able to generate samples from $x \sim p_C(x)$ and $x' \sim p_C(x'|x)$.

#### Basic ingredients

Estimating $p_C(x)$ is possible using training data, which is untainted by assumption. For this, we can use an implicit generative model, such as a generative adversarial network (Goodfellow et al. 2014) or an energy-based model (Grathwohl et al. 2019).

On the other hand, sampling from $p_C(x'|x)$ entails strategic thinking, which we shall treat with the ARA methodology.

Recall that the attacker only attacks malicious instances, those with labels $y_i$ with $i = 1, \ldots, l$. Thus, we base our analysis on the decomposition

$$p_C(x'|x) = \sum_{i=1}^{k} p_C(x'|x, y_i)p_C(y_i|x) = \sum_{i=1}^{l} p_C(x'|x, y_i)p_C(y_i|x) + \sum_{i=l+1}^{k} I(x' = x)p_C(y_i|x),$$

where $I$ is the indicator function. We can easily generate samples from $p_C(y_i|x)$, as we can estimate those probabilities based on training data. Then, we can obtain samples from $p_C(x'|x)$ by sampling $y_i \sim p_C(y_i|x)$ first and, then, if $i > l$ return $x$ or sample $x' \sim p_C(x'|x, y_i)$ otherwise.

To sample from $p_C(x'|x, y_i)$, the ARA methodology helps us to model the Attacker's decision problem when he has available an instance $x$ with label $y_i$. As we are not assuming common knowledge, we need to model our uncertainty about the Attacker's elements. As in Section 3.2.2, assume that agent $A$ also aims at maximizing his expected utility when trying to confuse $C$. His utility function has the form $u_A(y_c, y_i)$, when $C$ says $y_c$ and the actual label is $y_i$. For simplicity, we assume that the attacks have no cost, this is the reason why the utility now does not depend on $a$. With this, the Attacker would choose feature modification that maximizes his expected utility by making $C$ classify instances as most beneficial as possible to him. We scale utilities to lie in the interval $[0, 1]$. As we are focusing in integrity violation attacks, the utility that $A$ derives from $C$'s decision has the following structure:

$$u_A^{ci} := u_A(y_c, y_i) = \begin{cases} 0, & \text{if } i \leq l \text{ and } j \leq l \\ u_A^{ji} \neq 0, & \text{if } i \leq l \text{ and } j > l \\ 0, & \text{if } i > l \end{cases}$$

This reflects that the Attacker just obtains benefit when he makes the defender classify a bad instance as if it was a good one.

By transforming instance $x$ with label $y_i$ for $i = 1, \ldots, l$ into instance $x'$, the attacker would get an expected utility

$$\sum_{c=1}^{k} u_A(y_c, y_i) p_A(y_c|x') = \sum_{c=l+1}^{k} u_A^{ci}\, p_A(y_C = c|x'), \tag{3.5.3}$$

where $p_A(y_c|x')$ describes the probability that $C$ says type $y_c$ if she observes $x'$, from $A$'s perspective. As in Section 3.2.2, the Attacker will typically be uncertain about such probability. Suppose we model it with a density $f_A(p_c|x')$, with mean $p_A^c(x')$. Taking expectations in (3.5.3), the expected utility he would get is $\sum_{c=l+1}^{k} u_A^{ci}\, p_A^c(x')$. Thus, the attacker would choose his action through

$$x'(x, y_i) = \arg\max_{z \in \mathcal{X}'} \sum_{c=l+1}^{k} u_A^{ci} p_A^c(z), \tag{3.5.4}$$

and craft object $(x, y_i)$ into $(x'(x, y_i), y_i)$.

However, we do not know $u_A$ neither $p_A^c$ with certitude. If we model our uncertainty about these elements with, respectively, random utilities $U_A$ and random expected probabilities $P_A^c$, defined over an appropriate probability space. We would look for the random optimal transformation defined by

$$X'(x, y_i) = \arg\max_{z \in \mathcal{X}'} \sum_{c=l+1}^{k} U_A^{ci} P_A^c(z), \tag{3.5.5}$$

and make $p_C(x|x', y_i) = Pr(X^{'}(x, y_i) = x')$. Then, by construction, if we sample $u_A \sim U_A$ and $p_A^c \sim P_A^c$ and solve

$$x' = \underset{z \in \mathcal{X}'}{\operatorname{argmax}} \sum_{c=l+1}^{k} u_A^{ci} p_A^c(z),$$

$x'$ would be distributed according to $p_C(x'|x, y_i)$.

We can model random utilities and probabilities as in Section 3.2.2. As we announced, modeling $P_A^c(x')$ is more delicate as it entails strategic thinking. Here, we provide another relevant heuristic that consists of modelling $P_A^c(x')$ using a distribution based on $p(y_c|x')$ with some uncertainty around it. For this, given $x'$, consider the set $\mathcal{X}'$ of reasonable origins. Imagine we assess a distribution $p^*(x|x')$ over it, for example using the metric based approach. Let $mean_c = \sum_x p(y_c|x)p^*(x|x')$ and, for a given variance $var_c$, choose $P_A^c(x') \sim Beta(\alpha, \beta)$ having the above $mean_c$ and $var_c$, for which we just make $\alpha^c = \left(\frac{1-mean_c}{var_c} - \frac{1}{mean_c}\right) mean_c^2$ and $\beta^c = \alpha^c \left(\frac{1}{mean_c} - 1\right)$. In the above expression for $mean_c$, $p(y_c|x)$ would come from the estimates based on untainted data; to reduce the computational cost, we could approximate $mean_c$ through $\frac{1}{M} \sum_{n=1}^{M} p(y_C|x_n)$, for a sample $\{x_n\}_{n=1}^{M}$ from $p^*(x|x')$.

### AB-ACRA

Once we are able to sample from $p_C(x)$ and $p_C(x'|x)$, we need a procedure to sample from $p_C(x|x')$. In the discrete case[3], we can use rejection sampling (Casella et al. 2004). This entails generating $x \sim p_C(X)$, $\tilde{x}' \sim p_C(X'|X = x)$, and accepting $x$ only if $\tilde{x}'$ coincides with the actually observed instance $x'$. It is straightforward to prove that $x \sim p_C(X|X' = x')$. We can think of this procedure as generating instances $x$ and indicators $I$, where $I = 0(= 1)$ if we reject (accept) the sample. Accepted instances are distributed according to

$$p_C(X = x|I = 1) \propto p_C(I = 1|X = x)p_C(X = x) \propto p_C(X' = x'|X = x)p_C(X = x)$$

which, using Bayes rule, coincides with the desired distribution.

When $x'$ is continuous and/or high dimensional, the acceptance rate would typically be very low, making the above approach inefficient. In such cases, we can leverage ABC techniques. This entails accepting the sample $x$ if $\phi(\tilde{x}', x') < TOL$, for a given distance $\phi$ and tolerance $TOL$. The $x$ generated in this manner is distributed approximately according to $p_C(x|x')$. However, the probability of generating samples for which $\phi(\tilde{x}', x') < TOL$ decreases as the dimensionality of $x'$ increases. A common solution replaces the acceptance criterion by $\phi(s(\tilde{x}'), s(x')) < TOL$, where $s(x)$ is a set of summary statistics that capture the relevant information in $x$. The particular choice of summary statistics is problem specific. We summarize the whole procedure in Algorithm 7, which would be integrated within Algorithm 1.

[3]Here, for convenience, we distinguish between random variables and realizations using upper and lower cases, respectively. Thus, $X'$ refers to the actually observed instance and $X$ to the originating one.

**Algorithm 7:** ABC scheme to sample from $p_C(x|x')$

**Input:** Observed instance $x'$, data model $p_C(x)$, $U_A$, $P_A^c$, family of statistics $s$, $TOL$. **Output:** A sample approximately distributed according to $p_C(x|x')$.

**while** $\phi(s(x'), s(\tilde{x}')) > TOL$ **do**
  Sample $x \sim p_C(x)$ Sample $y_i \sim p_C(y_i|x)$ **if** $i > l$ **then**
    $\tilde{x}' = x$ **else**
     S
    **end**
    ample $u_A \sim U_A$ and $p_A^c \sim P_A^c$;
    Compute $\tilde{x}' = \text{argmax}_{z \in \mathcal{X}'} \sum_{c=l+1}^{k} u_A^{ci} p_A^c(z)$
  **end**
  Compute $\phi(s(x'), s(\tilde{x}'))$
**end**
**return** $x$

### 3.5.2 A case study in multiclass malware detection

We illustrate the proposed approach with a malware detection problem[4]. Malware of different types is increasingly being delivered by attackers to obtain a benefit, being a current major global cyber threat (ENISA 2019). Malware types include, among others: trojans, aimed at misleading the victim of its real intention of accessing personal information such as passwords; adware, which releases advertisements through the victim interface; or virus, that can replicate itself modifying other programs causing system failures, wasting host resources or corrupting data. It is crucial to detect the appropriate type of malware to decide the relevant countermeasures and mitigate its consequences. Recently, obfuscation attacks on malware binaries (You and Yim 2010) have gained relevance as they affect critically the performance of detection algorithms as shown in Example 2. We test the performance of AB-ACRA as a defence mechanism against obfuscation attacks, based on a 0-1 utility for the Defender.

For these experiments, we use a dataset containing malware and benign binaries. Malware was provided by Virus Total (Chronicle 2018) and contains trojans, adware and virus. Benign binaries were obtained from clean copies of the Program Files folder of MS Windows 7 and 8. The proportion of malware binaries in the data is 50%. Features were extracted from binaries through: the *Assembly Language Source*, from which we extract registers, operation codes, API calls and keywords; and, the *Portable Executable Header*, providing the symbols and imports. In total, we use 76 binary features coded with 1 (0) indicating the presence (absence) of the corresponding characteristic. Additionally, a label indicates whether the binary is trojan ($y_i = 1$), adware ($y_i = 2$), virus ($y_i = 3$) or benign ($y_i = 4$). We randomly split the dataset into train and test subsets: 80%, 20%; respectively.

[4]All code to reproduce these experiments is available at `https://github.com/roinaveiro/ACRA_2`.

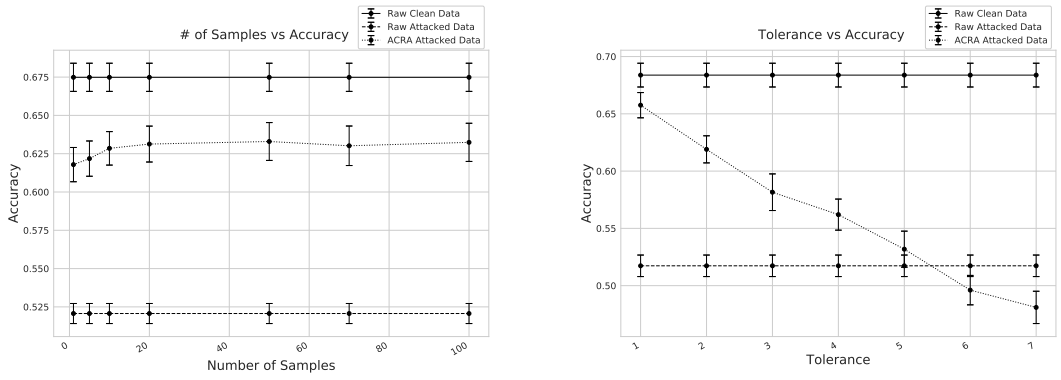
As underlying classification algorithm we deploy multinomial regression (MR) with L1 regularization. This is equivalent to performing maximum a posteriori estimation in an MR model with a Laplace prior, Park and Casella (2008). The regularization parameter was chosen using cross validation. Mean and standard deviations of accuracies in all experiments are estimated via repeated hold-out validation over ten repetitions (Kim 2009). The accuracy of this approach on clean test data is $0.68 \pm 0.01$.

To compare AB-ACRA with raw MR on tampered data, we simulate attacks over the instances in the test set. For this purpose, we solved problem (3.5.5) for each test malware binary, removing the uncertainty that is not present from the adversary's point of view. We restrict to attacks that involve changing at most the value of one of the features. The utility that the attacker perceives when he makes the defender misclassify a malware binary is 0.7 for all malware types. Finally, the adversary would have uncertainty about $p_A^c(x')$, as this quantity depends on the defender's decision. We test AB-ACRA against a worst case adversary who knows the true value of $p(y_C|x_n)$ and estimates $p_A^c(x')$ through $\frac{1}{M}\sum_{n=1}^{M} p(y_C|x_n)$ for a sample $\{x_n\}_{n=1}^{M}$ from $p^*(x|x')$. We set $M = 40$. For $p^*(x|x')$ we use a uniform distribution on the set of all instances at distance 1 from the observed $x'$, using as distance $\lambda(x, x') = \sum_{i=1}^{76} |x_i - x'_i|$.

To model the uncertainty about $p_A^c(x')$ and the attacker's utility function from the defender's perspective, we use beta distributions centered at the attacker's values of probabilities and utilities, respectively, with variances chosen to guarantee that the distribution is concave in its support: they must be bounded from above by $\min\left\{[\mu^2(1-\mu)]/(1+\mu), [\mu(1-\mu)^2]/(2-\mu)\right\}$, were $\mu$ is the corresponding mean. We set the variance to be 10% of this upper bound.

For the AB-ACRA algorithm, we used the 12 most relevant features (in terms of their coefficients having the highest posterior mode in absolute value) as summary statistics $s$. Figure 3.10a compares the accuracy of ACRA and MR for different sample sizes $N$ in Algorithm 6, and tolerance $TOL = 2$. As we can see, ACRA beats MR in tainted data with just 5 samples. The accuracy saturates quickly as we increase the number of samples. Thus, we get good performance with a relatively small sample size. Figure 3.10b plots the accuracy of ACRA against MR for different values of $TOL$. As expected, as this parameter decreases, accuracy increases, albeit at a higher computational cost.

Finally, we compare AB-ACRA with tolerance 1 (this value was found to give reasonable results for a moderate computational time), with MR and a heuristic approach that assumes that $p(x|x')$ is a uniform distribution over all possible instances at distance one from the observed $x'$. MR and the heuristic approach both obtained accuracy $0.52 \pm 0.01$, while AB-ACRA obtained $0.66 \pm 0.01$. As anticipated, defences that do not model explicitly the attacker's behavior have worse performance. However, AB-ACRA outperforms the heuristic approach and the raw MR as it explicitly models the attacker's behaviour.




(a) Experiment for different number of samples.

(b) Experiment for different values of tolerance.

**Figure 3.10**: Accuracy comparison LR vs AB-ACRA.

## 3.6 Discussion

Adversarial classification is an important example of provision of proactive defense that aims at enhancing classification algorithms to achieve robustness in presence of adversarial examples, as usually encountered in many security applications. The pioneering work of Dalvi et al. (2004) framed most of later approaches to adversarial classification within the standard game theoretic paradigm, in spite of the unrealistic common knowledge assumptions required, actually even questioned by the authors. This motivated us to focus on an ARA perspective to the problem.

In this chapter, we have presented ACRA, a general, Bayesian probabilistic framework for adversarial classification that models explicitly the presence of an adversary and our uncertainty about his elements. Our framework is general in the sense that application-specific assumptions are kept to a minimum. ACRA can naturally deal with generative classifiers. We have provided an extension based on approximate Bayesian computation, able to deal with both, generative and discriminative classifiers.

Also, we have provided extensive empirical evidence of the performance of our framework through case studies in spam and malware detection. In particular we have shown the robustness of ACRA to imprecisions in the assumptions made about the adversary. ACRA has been shown to be more robust than its common knowledge, purely game theoretic counterpart. Finally, we have presented computational enhancements that have significantly improved ACRA performance, allowing us to solve large problems and use it in operational settings.

# Chapter 4

# Algorithmic approaches: Gradient Methods for Stackelberg Games in AML

## 4.1 Introduction

As emphasized in Section 1.4.1, Stackelberg Games have been gaining importance in recent years due to their application in modeling confrontations within Adversarial Machine Learning problems. In this context, a new paradigm must be faced: while in classical game theory, intervening agents were humans whose decisions are generally discrete and low dimensional; in AML, decisions are made by algorithms and are usually continuous and high dimensional, e.g. choosing the weights of a neural network. As a result, scalable numerical algorithms to solve Stackelberg games are needed.

Following objective O3, in this chapter, we propose two procedures to solve Stackelberg games within the new paradigm of AML and study their time and space scalability. In particular, one of the proposed solutions scales efficiently in time with the dimension of the decision space (at the cost of more memory requirements). The other scales well in space, but requires more time. As Stackelberg games rely on strong common knowledge assumptions, unrealistic in AML, we extend the proposed methodologies to Bayesian Stackelberg games, in which that assumption is weakened.

The chapter is organized as follows: in Section 4.2 we define Stackelberg games. Section 4.3 presents the proposed solution methods as well as a discussion of their scalability. Section 4.4 extends the previous methodologies to the case of Bayesian Stackelberg games. The proposed solutions are illustrated with an AML experiment in Section 4.5. Finally, we conclude with a discussion.

## 4.2 Stackelberg games

As in Section 1.4.1, in this chapter we focus in sequential defend-attacks (Stackelberg) games. In particular, we restrict ourselves to sequential games with certain outcomes played between two agents: the first one makes her decision and then, after having observed the decision, the second one implements his response. As an example, consider the adversarial prediction problems introduced in Section 1.2.2. In them, the first agent chooses the parameters of a certain predictive model; the second agent, after having observed such choice, selects an optimal data transformation to fool the first agent, so as to gain some benefit.

As we focus on applications of Stackelberg games to AML, we restrict ourselves to the case in which the Defender ($D$) chooses her defense $\alpha \in \mathbb{R}^n$ and, then, the Attacker ($A$) chooses his attack $\beta \in \mathbb{R}^m$, after having observed $\alpha$. The corresponding bi-agent influence diagram is shown in Fig. 4.1. The dashed arc between nodes $D$ and $A$ reflects the fact that the Defender choice is observed by the Attacker. The utility function of the Defender, $u_D(\alpha, \beta)$, depends on both, her decision and the Attacker's decision. Similarly, the Attacker's utility function has the form $u_A(\alpha, \beta)$. In these games, the common knowledge hypothesis is that the Defender knows $u_A(\alpha, \beta)$.

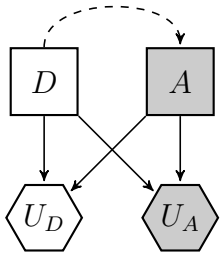



**Figure 4.1**: The two-player sequential decision game with certain outcome.

Mathematically, finding Nash equilibrium in Stackelberg games requires solving a bilevel optimization problem (Bard 1991). The defender's utility is called *upper level* or *outer* objective function while the attacker's one is referred to as the *lower level* or *inner* objective function. Similarly, the upper and lower level optimization problems, correspond to the defender's and the attacker's problem, respectively. These problems are also referred to as outer and inner problems.

As in Section 1.4.1, we assume that the attacker will act rationally in the sense that he will choose an action that maximizes his utility (French and Rios Insua 2000), given the disclosed defender's decision $\alpha$. Assuming that there is a unique global maximum of the attacker's utility for each $\alpha$, and calling it $\beta^*(\alpha)$, a Stackelberg equilibrium is identified using backward induction: the defender has to choose $\alpha^*$ that maximizes her utility subject to the attacker's best response $\beta^*(\alpha)$. Mathematically, the problem to be solved by the defender is

$$\begin{aligned} &\underset{\alpha}{\text{argmax}} && u_D[\alpha, \beta^*(\alpha)] \\ &\text{s.t.} && \beta^*(\alpha) = \underset{\beta}{\text{argmax}}\, u_A(\alpha, \beta). \end{aligned} \tag{4.2.1}$$

The pair $(\alpha^*, \beta^*(\alpha^*))$ is a Nash equilibrium and, indeed, a sub-game perfect equilibrium (Hargreaves-Heap and Varoufakis 2004).

When the attacker problem has more than one global maximum, several types of equilibria have been proposed. The two more important ones are the optimistic and the pessimistic solutions (Sinha et al. 2018). In an optimistic position, the defender expects the attacker to choose the optimal solution which gives the higher upper level utility. On the other hand, the pessimistic approach suggests that the defender should optimize for the worst case attacker solution. In this thesis, we just deal with the case in which the inner utility has a unique global maximum.

## 4.3 Solution Method

Bilevel optimization problems can rarely be solved analytically. Indeed, even extremely simple instances of bilevel problems have been shown to be NP-hard (Jeroslow 1985). Thus, numerical techniques are required. Several classical and evolutionary approaches have been proposed to solve (4.2.1), as reviewed by Sinha et al. (2018). When the inner problem satisfies certain regularity conditions, it is possible to reduce the bilevel optimization problem to a single level one replacing the inner problem with its Karush-Kuhn-Tucker (KKT) conditions (Gordon and Tibshirani 2012). Then, evolutionary techniques could be used to solve this single-level problem, thus making it possible to relax the upper level requirements. As, in general, this single-level reduction is not feasible, several other approaches have been proposed, such as nested evolutionary algorithms or metamodeling-based methods. However, most of these approaches lack scalability: increasing the number of upper level variables produces an exponential increase on the number of lower level tasks required to be solved being thus impossible to apply these techniques to solve large scale problems as the ones appearing in the context of AML.

In Brückner and Scheffer (2011) the authors face the problem of solving Stackelberg games in the AML context. However, they focus on a very particular type of game which can be reformulated as a quadratic program. In this chapter, we provide more general procedures to solve Stackelberg games that are useful in the AML paradigm in which decision spaces are continuous and high dimensional. To this end, we focus on gradient ascent techniques to solve bilevel maximization problems.

As we are assuming that for any $\alpha$ the solution of the inner problem is unique, we can define an implicit function $\beta^*(\alpha)$ that maps $\alpha$ into the corresponding solution. Thus, problem (4.2.1) may be viewed solely in terms of the defender's decisions $\alpha$. The underlying idea behind gradient ascent techniques is the following: given a defender decision $\alpha \in \mathbb{R}^n$ a direction along which the defender's utility increases while maintaining feasibility must be found, and then, we move $\alpha$ in that direction. Thus, the major issue of ascent methods is to find the gradient of $u_D(\alpha, \beta^*(\alpha))$. In Kolstad and Lasdon (1990), the authors provide a method to approximate such gradient that works for relatively large classical optimization problems but is clearly insufficient to deal with the typical bilevel problems appearing in AML.

Recently, Franceschi et al. (2017) proposed forward and reverse-based methods for computing the gradient of the validation error in certain hyperparamenter optimization problems that appear in Deep Learning. Structurally, hyperparameter optimization problems are similar to Stackelberg games. We adapt their methodology to this domain. In particular we propose two alternative approaches to compute the gradient of $u_D[\alpha, \beta^*(\alpha)]$ with different memory and running time requirements. We refer to these approaches as backward and forward solutions, respectively.

### 4.3.1 Backward solution

We propose here a new gradient ascent approach to solve the bilevel problem (4.2.1) whose running time scales well with the defender's decision space dimension. In particular, we propose to approximate problem (4.2.1) by the following PDE-constrained optimization problem (Hinze et al. 2008)

$$\begin{aligned} \underset{\alpha}{\text{argmax}} \quad & u_D\left[\alpha, \beta(\alpha, T)\right] \\ \text{s.t.} \quad & \partial_t \beta(\alpha, t) = \partial_\beta u_A[\alpha, \beta(\alpha, t)] \\ & \beta(\alpha, 0) = 0. \end{aligned} \tag{4.3.1}$$

The idea is formalized in the next proposition, that can be proved using results in Bottou (1998).

**Proposition 4.1.** *Suppose that the following assumptions hold*

*1'. The attacker problem, the inner problem in* (4.2.1)*, has a unique solution* $\beta^*(\alpha)$ *for each defender decision* $\alpha$*.*

*2'. For all* $\epsilon > 0$ *and all* $\alpha$*,*

$$\inf_{\|\beta^*(\alpha)-\beta\|_2^2>\epsilon} \langle \beta - \beta^*(\alpha), \partial_\beta u_A[\alpha, \beta] \rangle > 0.$$

*If* $\beta(\alpha, t)$ *satisfies the differential equation*

$$\partial_t \beta(\alpha, t) = \partial_\beta u_A[\alpha, \beta(\alpha, t)] \tag{4.3.2}$$

*then* $\beta(\alpha, t) \to \beta^*(\alpha)$ *as* $t \to \infty$*, with rate* $\mathcal{O}\left(\frac{1}{t}\right)$.

The idea in (4.3.1) is thus to constrain the trajectories $\beta(\alpha, t)$ to satisfy (4.3.2) and approximate the defender's problem using $\beta(\alpha, T)$ with $T \gg 1$, instead of $\beta^*(\alpha)$.

We propose solving problem (4.3.1) using gradient ascent and the adjoint method (Pontryagin 2018) to compute the total derivative of the defender utility function with respect to her decision. The adjoint method defines an *adjoint function* $\lambda(t)$ satisfying the *adjoint equation*

$$\mathrm{d}_t \lambda(t) = -\lambda(t)\ \partial_\beta^2 u_A[\alpha, \beta(\alpha, t)]. \tag{4.3.3}$$

Then, we can compute the derivative of the defender utility function with respect to her decision using Theorem 4.1.

**Theorem 4.1.** *If $\lambda(t)$ satisfies the adjoint equation* (4.3.3)*, the derivative of the defender utility with respect to her decision may be written as*

$$\mathrm{d}_\alpha u_D[\alpha, \beta(\alpha, T)] = \partial_\alpha u_D[\alpha, \beta(\alpha, T)] - \int_0^T \lambda(t) \partial_\alpha \partial_\beta u_A[\alpha, \beta(\alpha, t)] \, \mathrm{d}t. \tag{4.3.4}$$

*Proof.* The Lagrangian of problem (4.3.1) is

$$\mathcal{L} = u_D[\alpha, \beta(\alpha, T)] + \int_0^T \lambda(t) \left\{ \mathrm{d}_t \beta(\alpha, t) - \partial_\beta u_A[\alpha, \beta(\alpha, t)] \right\} dt + \mu \beta(\alpha, 0).$$

As the constraints hold, by construction we have that $d_\alpha \mathcal{L} = d_\alpha u_D$ and

$$\begin{aligned} \mathrm{d}_\alpha \mathcal{L} &= \partial_\alpha u_D[\alpha, \beta(\alpha, T)] + \partial_\beta u_D[\alpha, \beta(\alpha, T)] \, \mathrm{d}_\alpha \beta(\alpha, T) + \mu \, \mathrm{d}_\alpha \beta(\alpha, 0) \\ &+ \int_0^T \lambda(t) \left\{ \mathrm{d}_t \, \mathrm{d}_\alpha \beta(\alpha, t) - \partial_\alpha \partial_\beta u_A[\alpha, \beta(\alpha, t)] - \partial_\beta^2 u_A[\alpha, \beta(\alpha, t)] \, \mathrm{d}_\alpha \beta(\alpha, t) \right\} \mathrm{d}t. \end{aligned} \tag{4.3.5}$$

Integrating by parts, we have

$$\int_0^T \lambda(t) \, \mathrm{d}_t \, \mathrm{d}_\alpha \beta(\alpha, t) \, \mathrm{d}t = [\lambda(t) \, \mathrm{d}_\alpha \beta(\alpha, t)]_0^T - \int_0^T \mathrm{d}_t \lambda(t) \, \mathrm{d}_\alpha \beta(\alpha, t) \, \mathrm{d}t$$

Inserting this in (4.3.5), and grouping the terms conveniently, we have

$$\begin{aligned} \mathrm{d}_\alpha \mathcal{L} &= \partial_\alpha u_D[\alpha, \beta(\alpha, T)] + \left\{ \partial_\beta u_D[\alpha, \beta(\alpha, T)] + \lambda(T) \right\} \mathrm{d}_\alpha \beta(\alpha, T) \\ &+ \{\mu - \lambda(0)\} \, \mathrm{d}_\alpha \beta(\alpha, 0) + \int_0^T \left\{ - \mathrm{d}_t \lambda(t) - \lambda(t) \partial_\beta^2 u_A[\alpha, \beta(\alpha, t)] \right\} \mathrm{d}_\alpha \beta(\alpha, t) \, \mathrm{d}t \\ &- \int_0^T \lambda(t) \partial_\alpha \partial_\beta u_A[\alpha, \beta(\alpha, t)] \, \mathrm{d}t \end{aligned}$$

Since the constraints hold, we may choose the Lagrange multipliers freely. In particular, we may choose them so that we can avoid calculating the derivatives of $\beta(\alpha, t)$ with respect to $\alpha$ (as this is computationally expensive). Thus, we have that $\lambda$ satisfies the adjoint equation

$$\mathrm{d}_t \lambda(t) = -\lambda(t) \partial_\beta^2 u_A[\alpha, \beta(\alpha, t)]$$

with $\lambda(T) = -\partial_\beta u_D[\alpha, \beta(\alpha, T)]$, and $\mu = \lambda(0)$. Using this, the derivative is computed as

$$\mathrm{d}_\alpha \mathcal{L} = \partial_\alpha u_D[\alpha, \beta(\alpha, T)] - \int_0^T \lambda(t) \partial_\alpha \partial_\beta u_A[\alpha, \beta(\alpha, t)] \, \mathrm{d}t.$$

□

**Algorithm 8:** Approximate total derivative of defender utility function with respect to her decision using backward solution

Set $T$ to be the number of iterations;
$\alpha$ is the point where we compute the derivative;
$\beta_0(\alpha) = 0$;
**for** $t = 1, 2, \ldots, T$ **do**
  $\beta_t(\alpha) = \beta_{t-1}(\alpha) + \eta \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}(\alpha)}$;
**end**
$\lambda_T = -\partial_\beta u_D(\alpha, \beta)\Big|_{\beta_T(\alpha)}$;
$\mathrm{d}_\alpha u_D = \partial_\alpha u_D[\alpha, \beta_T(\alpha)]$;
**for** $t = T-1, T-2, \ldots, 0$ **do**
  $\mathrm{d}_\alpha u_D = \mathrm{d}_\alpha u_D - \eta \lambda_{t+1} \partial_\alpha \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t+1}(\alpha)}$;
  $\lambda_t = \lambda_{t+1} \left[ I + \eta \partial_\beta^2 u_A(\alpha, \beta)\Big|_{\beta_{t+1}(\alpha)} \right]$;
**end**
**return** $\mathrm{d}_\alpha u_D$;

Algorithmically, we can proceed by discretizing (4.3.3) via Euler method, and approximate the derivative (4.3.4) discretizing the integral on the left hand side. This leads to Algorithm 8. Once we are able to compute this derivative, we can solve the defender's problem using gradient ascent.

Regarding its complexity, note that by basic facts of Automatic Differentiation (AD) (Griewank and Walther 2008), if $\tau(n, m)$ is the time required to evaluate $u_D(\alpha, \beta)$ and $u_A(\alpha, \beta)$, then computing derivatives of these functions requires time $\mathcal{O}(\tau(n, m))$. Thus the first for loop in Algorithm 8 requires time $\mathcal{O}(T\tau(n, m))$. In the second loop, we need to compute second derivatives which appear always multiplying the vector $\lambda_t$. By basic results of AD, Hessian vector products have the same time complexity as function evaluations. Thus in our case, we can compute second derivatives in time $\mathcal{O}(\tau(n, m))$, being the time complexity of the second for loop $\mathcal{O}(T\tau(n, m))$. Thus, overall, Algorithm 8 runs in time $\mathcal{O}(T\tau(n, m))$. Regarding space complexity, as it is necessary to store the values of $\beta_t(\alpha)$ produced in the first loop for later usage in the second one, if $\sigma(n, m)$ is the space requirement for storing each $\beta_t(\alpha)$, then $\mathcal{O}(T\sigma(n, m))$ is the space complexity of the backward algorithm.

In certain applications where space complexity is critical, the backward solution could be infeasible as it requires storing the whole trace $\beta_t(\alpha)$ within each iteration. In this case, the forward solution proposed in the next section solves this issue at a cost of loosing time scalability.

### 4.3.2 Forward solution

In this case, we approximate (4.2.1) through the problem

$$\begin{aligned} \underset{\alpha}{\text{argmax}} \quad & u_D\left[\alpha, \beta_T(\alpha)\right] \\ \text{s.t} \quad & \beta_t(\alpha) = \beta_{t-1}(\alpha) + \eta_t \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}} \quad t = 1, \dots, T \\ & \beta_0(\alpha) = 0. \end{aligned} \tag{4.3.6}$$

The idea here is that, for each defense $\alpha$, we condition on a dynamical system that, under certain conditions, converges to $\beta^*(\alpha)$, the optimal solution for the attacker when the defender implements $\alpha$. Thus, we can approximate the defender's utility by $u_D\left[\alpha, \beta_T(\alpha)\right]$, with $T \gg 1$. This idea is formalized in the next proposition that can be proved using the results of Bottou (1998).

**Proposition 4.2.** *Suppose that the following assumptions hold*

*1'. The attacker problem (inner problem in* (4.2.1)*) has a unique solution* $\beta^*(\alpha)$ *for each defender decision* $\alpha$*.*

*2'. For all* $\epsilon > 0$ *and* $\alpha$

$$\inf_{\|\beta - \beta^*(\alpha)\|_2^2 > \epsilon} \langle \beta - \beta^*(\alpha), \partial_\beta u_A[\alpha, \beta] \rangle > 0.$$

*3'. For some* $C, D > 0$ *and all* $\alpha$

$$\|\partial_\beta u_A[\alpha, \beta]\|_2^2 \leq C + D\|\beta - \beta^*(\alpha)\|_2^2.$$

*If for all* $t$, $\beta_t(\alpha)$ *satisfies*

$$\beta_t(\alpha) = \beta_{t-1}(\alpha) + \eta \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}}, \tag{4.3.7}$$

*then,* $\beta_t(\alpha)$ *converges to* $\beta^*(\alpha)$*, with rate* $\mathcal{O}\left(\frac{1}{t}\right)$*.*

We propose solving problem (4.3.6) using gradient ascent. To that end, we need to compute $\mathrm{d}_\alpha u_D(\alpha, \beta_T(\alpha))$. Using the chain rule we have

$$\mathrm{d}_\alpha u_D[\alpha, \beta_T(\alpha)] = \partial_\alpha u_D[\alpha, \beta_T(\alpha)] + \partial_{\beta_T} u_D[\alpha, \beta_T(\alpha)]\, \mathrm{d}_\alpha \beta_T(\alpha).$$

To obtain $\mathrm{d}_\alpha \beta_T(\alpha)$, we sequentially compute $\mathrm{d}_\alpha \beta_t(\alpha)$ taking derivatives in (4.3.7)

$$\mathrm{d}_\alpha \beta_t(\alpha) = \mathrm{d}_\alpha \beta_{t-1}(\alpha) + \eta_{t-1} \left[ \partial_\alpha \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}} + \partial_\beta^2 u_A(\alpha, \beta)\Big|_{\beta_{t-1}} \mathrm{d}_\alpha \beta_{t-1}(\alpha) \right].$$

This induces a dynamical system in $\mathrm{d}_\alpha \beta_t(\alpha)$ that can be iterated in parallel to the dynamical system in $\beta_t(\alpha)$. The whole procedure is described in Algorithm 9. Once

**Algorithm 9:** Approximate total derivative of defender utility function with respect to her decision using forward solution.

Set $T$ to be the number of iterations;
$\alpha$ is the point where we compute the derivative;
$\beta_0(\alpha) = 0$;
$\mathrm{d}_\alpha \beta_0(\alpha) = 0$;
**for** $t = 1, 2, \ldots, T$ **do**
  $\beta_t(\alpha) = \beta_{t-1}(\alpha) + \eta \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}}$;
  $\mathrm{d}_\alpha \beta_t(\alpha) = \mathrm{d}_\alpha \beta_{t-1}(\alpha) + \eta_{t-1} \left[ \partial_\alpha \partial_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}} + \partial^2_\beta u_A(\alpha, \beta)\Big|_{\beta_{t-1}} \mathrm{d}_\alpha \beta_{t-1}(\alpha) \right]$;
**end**
$\mathrm{d}_\alpha u_D = \partial_\alpha u_D[\alpha, \beta_T(\alpha)] + \partial_{\beta_T} u_D[\alpha, \beta_T(\alpha)] \, \mathrm{d}_\alpha \beta_T(\alpha)$;
**return** $\mathrm{d}_\alpha u_D$;

we are able to compute this derivative, we solve the defender's problem using gradient ascent.

Regarding time complexity, the bottleneck in Algorithm 9 is that we need to compute the second derivatives of $u_A(\alpha, \beta)$. In particular, computing $\partial^2_\beta u_A(\alpha, \beta)$ requires time $\mathcal{O}(m\tau(m, n))$ as it requires computing $m$ Hessian vector products, one with each of the $m$ unitary vectors. On the other hand, computing $\partial_\alpha \partial_\beta u_A(\alpha, \beta)$ requires computing $n$ Hessian vector products and, thus, time $\mathcal{O}(n\tau(m, n))$. If we compute the derivative in the other way, first we derive with respect to $\beta$ and then with respect to $\alpha$, the time complexity is $\mathcal{O}(m\tau(m, n))$. Thus, we derive first with respect to the variable with the largest dimension. Then, the time complexity of computing $\partial_\alpha \partial_\beta u_A(\alpha, \beta)$ is $\mathcal{O}(\min(n, m)\tau(m, n))$. Finally, as $\partial^2_\beta u_A(\alpha, \beta)$ and $\partial_\alpha \partial_\beta u_A(\alpha, \beta)$ could be computed in parallel, then the overall time complexity of the forward solution is $\mathcal{O}(\max[\min(n, m), m]T\tau(m, n)) = \mathcal{O}(mT\tau(m, n))$. Regarding space, as in this case the values $\beta_t(\alpha)$ are overwritten at each iteration, we do not need to store all of them and the overall space complexity is $\mathcal{O}(\sigma(m, n))$.

## 4.4 An extension to Bayesian Stackelberg games

The Stackelberg games presented in Section 4.2 rely on an important common knowledge assumption: the defender must know the attacker's utility to solve her problem. This common knowledge condition is doubtful, specially in security settings as in AML in which the adversary has incentives to hide information to the defender. Moreover, the Nash equilibrium computed solving (4.2.1) could lack robustness to perturbations in $A$'s judgments (Ekin et al. 2019).

We can relax this assumption modeling the Defender's uncertainty about the attacker's utility function. Let us assume that the defender lacks precise knowledge

about a set of parameters of the attacker's utility function referred to as $\tau$. The Defender models her uncertainty through a prior $\pi(\tau)$ on the parameters. This is a particular instance of a Bayesian Stackelberg Game, (Jain et al. 2008).

A rational adversary with parameters $\tau$ would choose $\beta^*(\alpha, \tau) = \operatorname{argmax}_\beta u_A(\alpha, \beta, \tau)$ as the best response to the observed defender's decision $\alpha$. Similarly, a rational defender, knowing how the attacker with parameters $\tau$ could react to her decision, should choose $\alpha^*$ maximising her expected utility

$$\int u_D[\alpha, \beta^*(\alpha, \tau)]\pi(\tau)\,\mathrm{d}\tau$$

The pair $(\alpha^*, \beta^*(\alpha^*, \tau))$ constitutes a Bayes-Nash equilibrium (Harsanyi 1967). Thus, finding Bayes-Nash equilibria requires solving the following problem

$$\begin{aligned} \underset{\alpha}{\operatorname{argmax}} \quad & \int u_D[\alpha, \beta^*(\alpha, \tau)]\pi(\tau)\,\mathrm{d}\tau \\ \text{s.t.} \quad & \beta^*(\alpha, \tau) = \underset{\beta}{\operatorname{argmax}}\, u_A(\alpha, \beta, \tau) \quad \forall \tau \end{aligned} \tag{4.4.1}$$

This problem can rarely be solved analytically, so numerical techniques are required. The methodologies presented in Sections 4.3.1 and 4.3.2, can be easily extended to deal with Bayesian Stackelberg games as the one presented above. To that end, we approximate problem (4.4.1) by

$$\begin{aligned} \underset{\alpha}{\operatorname{argmax}} \quad & \frac{1}{N}\sum_{i=1}^{N} u_D[\alpha, \beta^*(\alpha, \tau_i)] \\ \text{s.t.} \quad & \beta^*(\alpha, \tau_i) = \underset{\beta}{\operatorname{argmax}}\, u_A(\alpha, \beta, \tau_i) \quad i = 1, \ldots, N \end{aligned} \tag{4.4.2}$$

where $\tau_1, \ldots, \tau_N$ are samples from $\pi(\tau)$. That is, we approximate the defender's utility using a Monte Carlo estimate (Hammersley 2013). By the strong law of large numbers, when $N \to \infty$, the quantity $\frac{1}{N}\sum_{i=1}^{N} u_D[\alpha, \beta^*(\alpha, \tau_i)]$ converges almost surely to the expected utility of the defender.

We propose using gradient ascent to solve (4.4.2). The total derivative of the MC estimator of the defender's utility function could be computed easily noting that $\mathrm{d}_\alpha \frac{1}{N}\sum_{i=1}^{N} u_D[\alpha, \beta^*(\alpha, \tau_i)] = \frac{1}{N}\sum_{i=1}^{N} \mathrm{d}_\alpha u_D[\alpha, \beta^*(\alpha, \tau_i)]$, and approximating each term of the sum using either Algorithm 8 or 9. The computation of $\frac{1}{N}\sum_{i=1}^{N} \mathrm{d}_\alpha u_D[\alpha, \beta^*(\alpha, \tau_i)]$ can be parallelized easily, thus maintaining the scalability advantages of the backward and forward solution methods mentioned above.

## 4.5 Experiments

We illustrate the proposed approaches. We start with a conceptual example in which we empirically test the scalability properties of the forward and backward solution methods. Then, we apply these algorithms to solve a problem in the context of adversarial regression. Finally, we apply the methodology in Section 4.4 to an

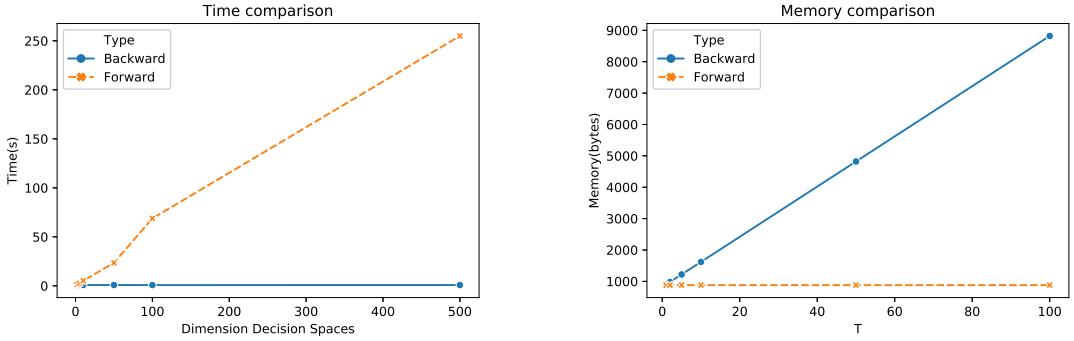
adversarial regression problem where the defender has limited knowledge of the attacker.

All the code used for these examples has been written in python using the pytorch library for Automatic Differentiation (Paszke et al. 2017) and is available at `https://github.com/roinaveiro/GM_SG`.

### 4.5.1 Conceptual Example

We use a simple example to illustrate the scalability of the proposed approaches. Consider that the attacker's and defender's decisions are both vectors in $\mathbb{R}^n$. The attacker's utility takes the form $u_A(\alpha, \beta) = -\sum_{i=1}^{n} 3(\beta_i - \alpha_j)^2$ and the defender's one is $u_D(\alpha, \beta) = -\sum_{i=1}^{n}(7\alpha_i + \beta_j^2)$. In this case, the equilibrium can be computed analytically using backward induction: for a given defense $\alpha \in \mathbb{R}^n$, we see that $\beta^*(\alpha) = \alpha$; substituting in the outer problem, the equilibrium is reached at $\alpha_j^* = -3.5, \beta_j^*(\alpha^*) = -3.5$ with $j = 1, \ldots, m$.

We apply the proposed methods to this problem to test their scalability empirically. The parameters were chosen as follows: the learning rate $\eta$ of Algorithms 8 and 9 was set to 0.1; similarly, the learning rate of gradient ascent used to solve the outer problem was also set to 0.1. Finally, all gradient ascents were run for $T = 40$, sufficient to reach convergence.




(a) Backward and Forward running times versus the dimension of decision spaces.

(b) Backward and Forward running memory usage versus length of trace $\beta_t(\alpha)$.

**Figure 4.2**: Empirical comparison of time and space scalability.

Figure 4.2a shows running times for increasing number of dimensions of the decision spaces (in this problem both the attacker's and the defender's decision space have the same dimension). As discussed, the forward running time increases linearly with the number of dimensions while the backward solution remains approximately constant. This obviously comes at the cost of having more memory requirements as in Algorithm 8 we need to store the whole trace $\beta_t(\alpha)$. Figure 4.2b shows how memory consumption increases linearly with the length $T$ of that trace for the backward method, while it remains constant for the forward one.

### 4.5.2 An application to adversarial regression

We illustrate an application of the proposed methodology to adversarial regression problems (Großhans et al. 2013). They are a specific class of prediction games, Brückner and Scheffer (2011), played between a *learner* of a regression model and a *data generator* who tries to fool the learner modifying input data at operation time, inducing a change between the data distribution at training and test time with the aim of confusing the data generator and attain a benefit.

Given a feature vector $x \in \mathbb{R}^p$ and its corresponding target value $y \in \mathbb{R}$, the learner's decision is to choose the weight vector $w \in \mathbb{R}^p$ of a linear model $f_w(x) = x^\top w$, that minimizes the theoretical costs at application time, given by

$$\theta_l(w, \bar{p}, c_l) = \int c_l(x, y)(f_w(x) - y)^2 \, \mathrm{d}\bar{p}(x, y),$$

where $c_l(x, y) \in \mathbb{R}^+$ reflects instance-specific costs and $\bar{p}(x, y)$ is the data distribution at test time which is different from the one a training time, that we shall call $p(x, y)$. To do so, the learner has a training matrix $X \in \mathbb{R}^{n \times p}$ and a vector of target values $y \in \mathbb{R}^n$, that is a sample from distribution $p(x, y)$.

The data generator aims at changing features of test instances to induce a transformation in the data distribution from $p(x, y)$ to $\bar{p}(x, y)$. Let $z(x, y)$ be the data generator's target value for an instance $x$ with real value $y$, i.e. he aims at transforming $x$ to make the learner predict $z(x, y)$ instead of $y$. The data generator aims at choosing the transformation that minimizes the theoretical costs given by

$$\theta_d(w, \bar{p}, c_d) = \int c_d(x, y)(f_w(x) - z(x, y))^2 \, \mathrm{d}\bar{p}(x, y) + \Omega_d(p, \bar{p}),$$

where $\Omega_d(p, \bar{p})$ is the incurred cost when transforming $p$ to $\bar{p}$ and $c_d(x, y)$ are instance specific costs.

As the theoretical costs defined above depend on the unknown distributions $p$ and $\bar{p}$, we focus on their regularized empirical counterparts, given by

$$\begin{aligned} \widehat{\theta}_l(w, \bar{X}, c_l) &= \sum_{i=1}^{n} c_{l,i}(f_w(\bar{x}_i) - y_i)^2 + \Omega_l(f_w), \\ \widehat{\theta}_d(w, \bar{X}, c_d) &= \sum_{i=1}^{n} c_{d,i}(f_w(\bar{x}_i) - z_i)^2 + \Omega_d(X, \bar{X}), \end{aligned}$$

where $\Omega_l(f_w)$ is a regularizer that accounts for the fact that $\bar{X}$ is not the future test data but a training sample transformed to reflect the test distribution and used to learn the model parameters.

In addition, we assume that the learner acts first, choosing a weight vector $w$. Then the data generator, after observing $w$, chooses his optimal data transformation. Thus, the problem to be solved by the learner is

$$\begin{aligned} \underset{w}{\operatorname{argmin}} \quad & \widehat{\theta}_l(w, T(X, w, c_d), c_l) \\ \text{s.t.} \quad & T(X, w, c_d) = \underset{X'}{\operatorname{argmin}} \, \widehat{\theta}_d(w, X', c_d), \end{aligned} \tag{4.5.1}$$

where $T(X, w, c_d)$ is the attacker's optimal transformation for a given choice $w$ of weight vector. (4.5.1) has the same form as (4.2.1), except that it is formulated in terms of costs rather than utilities. In addition, it is easy to see that if $\Omega_d(X, \bar{X})$ is equal to the squared Frobenius norm of the difference matrix $\|X - \bar{X}\|_F^2$, then the attacker's problem has a unique solution. Thus, we can use the proposed solution techniques to look for Nash equilibria in this type of game, taking care of performing gradient descent instead of gradient ascent, as we are minimizing costs here.

We apply the results to the UCI white wine dataset (Lichman 2013). This contains information about 4898 wines consisting of 11 quality indicators plus a wine quality score. $R_J$ and $R_D$ are two competing wine brands. $R_D$ has implemented a system to automatically measure wine quality using a regression over the available quality indicators: each wine is described by a vector of 11 entries, one per quality indicator. Wine quality is a discrete variable that ranges between 0 and 10. $R_J$, aware of the actual superiority of its competitor's wines, decides to hack $R_D$'s system by manipulating the value of several quality indicators to artificially decrease $R_D$'s quality rates. However, $R_D$ is aware of the possibility of being hacked, and decides to use adversarial methods to train its system. In particular, $R_D$ models the situation as a Stackelberg game. It is obvious that the target value of his enemy is $z(x, y) = 0$ for every possible wine. In addition, $R_D$ was able to filter some information about $R_J$'s wine-specific costs $c_{d,i}$.

As basic underlying model, a regular ridge regression (Friedman et al. 2001), was trained using 11 principal components as features. The regularization strength was chosen using repeated hold-out validation (Kim 2009), with ten repetitions. As performance metric we used the root mean squared error (RMSE), estimated via repeated hold-out.

We compare the performance of two different learners against an adversary whose wine specific costs $c_{d,i}$ are fixed. The first one, referred to as Nash, assumes that the wine specific costs are common knowledge and plays Nash equilibria in the Stackelberg game defined in (4.5.1). The second learner, referred to as raw, is a non adversarial one and uses a ridge regression model. To this end, we split the data in two parts, 2/3 for training purposes and the remaining 1/3 for test. The training set is used to compute the weights $w$ of the regression problem. Those weights are observed by the adversary, and used to attack the test set. Then, the RMSE is computed using this attacked test set and the previously computed weights.

In order to solve (4.5.1), we use the backward solution solution method of Section 4.3.1 due to its time scalability. The hyperparameters were chosen as follows: number of epochs $T$ to compute the gradient in Algorithm 8, 100; the learning rate $\eta$ in this same Algorithm, was set to 0.01. Within the gradient descent optimization used to optimize the defender's cost function, the number of epochs was set to 350 and the learning rate to $10^{-6}$. Finally, we assumed that the wine specific costs were the same for all instances and called the common value $c_d$. We studied how $c_d$ affects the RMSE for different solutions.

Notice that, in this case, the dimension of the attacker's decision space is huge. He has to modify the training data to minimize his costs. If there are $k$ instances

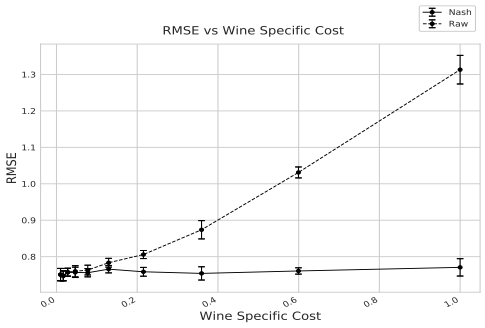



**Figure 4.3**: Performance comparison.

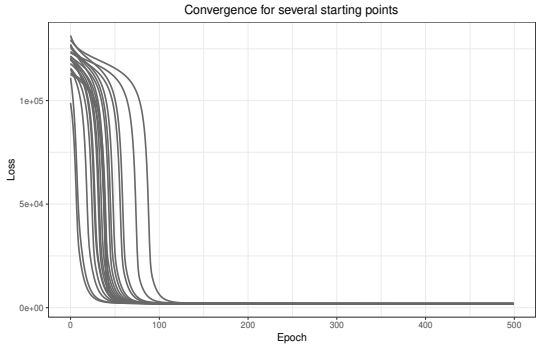



**Figure 4.4**: Convergence for several initial points.

in the training set, each of dimension $n$, the dimension of the attacker's decision space is $n \times k$. In this case $k = 3263$ (2/3 of 4898) and $n = 11$. Thus the forward solution is impractical, and we did not compute it. We show in Figure 4.3, the RMSE for different values of the wine specific cost. We observe that Nash outperforms systematically the adversary unaware regression method. When $c_d \to 0$, we see that $\widehat{\theta}_d(w, \bar{X}, c_d) \to \Omega_d(X, \bar{X})$. Thus, in this situation, the adversary will not manipulate the data. Consequently the Nash and ridge regression solutions will coincide, as shown in Figure 4.3. However, as $c_d$ increases, data manipulation is bigger, and the RMSE of the adversary unaware method also increases. On the other hand, the Nash solution RMSE remains almost constant.

We have also computed the average and standard deviation of training times. In an Intel Core i7-3630UM, 2.40GHz $\times$ 8 computer, the average training time was 131.6 seconds with 2.7 seconds as standard deviation. This corresponds approximately to 2.66 seconds per outer epoch. Each outer epoch involves running Algorithm 8 with 100 inner epochs.

Finally, to illustrate the convergence of the proposed approach, we solve (4.5.1) using gradient descent with the backward method for 20 different random initializations of the defender's decisions $\omega$. Results are depicted in Figure 4.4. As can be seen, all paths converge with less than 150 epochs.

### 4.5.3 An application to adversarial regression with limited knowledge

As in Großhans et al. (2013), we can relax the common knowledge assumption in the adversarial regression problem of Section 4.5.2, introducing uncertainty in the learner's knowledge about the data generator instance specific costs $c_d$. If $\pi(c_d)$ is the prior reflecting this uncertainty, the problem to be solved by the defender is

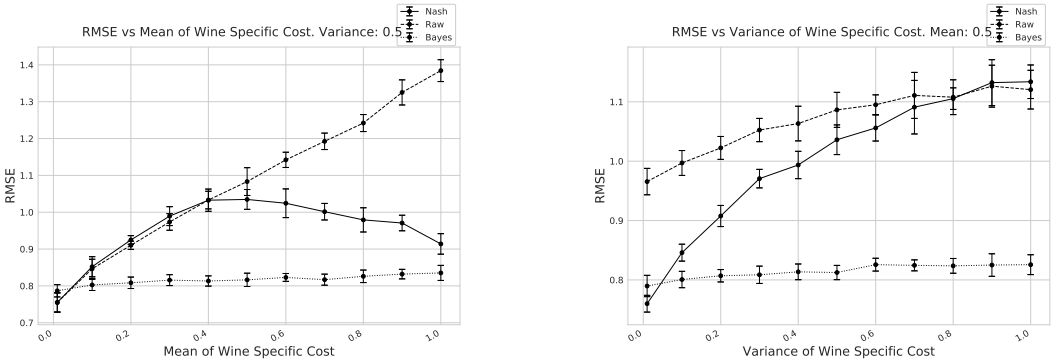
$$\begin{aligned} \underset{w}{\text{argmin}} \quad & \int \widehat{\theta}_l(w, T(X, w, c_d), c_l) \\ \text{s.t.} \quad & T(X, w, c_d) = \underset{X'}{\text{argmin}}\, \widehat{\theta}_d(w, X', c_d), \end{aligned} \tag{4.5.2}$$

Note that this problem has the same form as (4.4.1), except that it is formulated in terms of costs rather than utilities. Thus, we can use the technique in Section 4.4 to compute the Bayes-Nash equilibrium of this game.

In Section 4.5.2, we assumed that $R_D$ was able to filter information about $R_J$'s wine-specific costs $c_{d,i}$. However, in realistic settings, it is not reasonable to assume that this information will be precise, as the competitors have incentives to conceal information to each other.

In this section we study the performance of a more realistic learner, referred to as Bayes, that decides to put a gamma prior $\pi(c_{d,i})$ on $c_{d,i}$ reflecting the lack of precise knowledge about the wine specific costs. This learner plays Bayes-Nash equilibrium in the Bayesian Stackelberg game defined in (4.5.2), computed using the methodology in Section 4.4 coupled with the backward solution method.

The experimental setting is the same as that of Section 4.5.2. We compare the Bayes learner with two other players: one, referred to as Nash, that assumes that $R_J$'s wine-specific costs are the mean value of the previous gamma distribution and plays Nash equilibrium of (4.5.1), and another non adversarial learner, referred to as raw, that uses a ridge regression model. We compare the performance of the three different learners against an adversary whose wine specific costs are sampled from the conjectured gamma distribution.




(a) (b)

**Figure 4.5**: Performance comparison for different means and variances of the wine specific costs.

In Figure 4.5a, we represent the RMSE of the different solutions against the mean of the wine specific costs, with fixed variance 0.5. In Figure 4.5b we represent RMSE versus the variance of the wine specific costs for a fixed mean value of 0.5. We see that Bayes consistently outperforms the other players, specially for high values of variance, where the uncertainty is bigger and consequently, the Bayesian approach is clearly better.

## 4.6 Discussion

The demand for scalable solutions of Stackelberg Games has increased in the last years due to the use of such games to model confrontations within Adversarial Machine Learning problems. In this chapter, we have focused on gradient methods for solving Stackelberg Games, providing two different approaches to compute the gradient of the defender's utility function: the forward and backward solutions. In particular, we have shown that the backward solution scales well in time with the defender's decision space dimension, at a cost of more memory requirements. On the other hand, the forward solution scales poorly in time with this dimension, but has better space scalability. In addition, we have extended these approaches to the case of Bayesian Stackelberg games.

We have provided empirical support of the scalability properties of both approaches using a simple example. In addition, we have solved an AML problem using the backward solution in a reasonable amount of time. In this problem, the defender's decision space is continuous with dimension 11. The attacker's decision space is also continuous with dimension $\mathcal{O}(10^4)$, as we showed in Section 4.5.2. To the best of our knowledge, none of previous numerical techniques for solving Stackelberg games could deal, in reasonable time, with such high dimensional continuous decision spaces.

Apart from scalability properties, a major advantage of the proposed framework is that it could be directly implemented in any Automatic Differentiation library such as PyTorch (the one used in this example) or TensorFlow, and thus benefit from the computational advantages of such implementations. Finally, we highlight that one of the most important contributions of this chapter is the derivation of the backward solution formulating the Stackelberg game (4.2.1) as a PDE-constrained optimization problem and using the adjoint method. This provides a general and scalable framework that could be used to seek for Nash equilibria in other types of sequential games. Exploring this, is another possible line of future work.

# Chapter 5

# Algorithmic approaches: APS Methods for Non-cooperative Games

## 5.1 Introduction

As discussed in Chapter 5, efficient and scalable algorithmic methods for solving game theoretic problems are gaining importance due to the rise of adversarial machine learning (AML). Our realm in this one is precisely within algorithmic decision (Rossi and Tsoukias 2009) and game (Nisan et al. 2007) theories, in that we propose efficient algorithms to approximate solutions for game theoretic problems. Chapter 4 dealt with gradient-based solution methods. This chapter focuses on simulation based approaches. Among these, Monte Carlo (MC) methods are straightforward to use and widely implemented. However, they can be inefficient under certain conditions such as in presence of a high number of decision alternatives for the agents. For instance, counter-terrorism, adversarial machine learning and (cyber) security problems may involve thousands of possible decisions, and there could be large uncertainties associated with the goals and resources of the attackers (Zhuang and Bier 2007). This can result in computational challenges especially in cases where model uncertainty dominates (Rios Insua et al. 2009). Sampling procedures that focus on high-probability events, would have the potential to handle such computational challenges. Augmented probability simulation (APS), (Bielza et al. 1999) is a powerful simulation based methodology used to approximate optimal solutions in decision analytic problems. In particular, following objective O4, we analyze how APS may be used to efficiently compute game theoretic solutions in both the standard and ARA settings.

On the whole, we present a comprehensive robust decision support framework with novel computational algorithms for decision makers in a non-cooperative sequential setup. The proposed approach can be especially beneficial in application domains such as cybersecurity, AML and counter-terrorism. In particular, Sections 5.2 and 5.3 provide approaches to approximate subgame perfect equilibria under complete

information, assessing their robustness and, finally, approximate ARA solutions under incomplete information. A computational assessment and the solution of a cybersecurity case study are presented in Sections 5.4 and 5.5. The chapter concludes with a discussion in Section 5.6. Code to reproduce the results in this chapter is available in the GitHub repository `https://github.com/roinaveiro/aps`, including parameter choices. In the Appendix we present additional results and algorithms relevant in particular settings.

## 5.2 Sequential non-cooperative games with complete information

This section focuses on computational methods for finding equilibria in sequential non-cooperative games with complete information. These games have received various names including sequential defend-attack (Brown et al. 2006) and Stackelberg games (Gibbons 1992). As an example, consider a company that must determine the cybersecurity controls to deploy given that a hacker could observe them and launch a distributed denial-of-service (DDoS) attack.

Sequential non-cooperative games were introduced in Section 1.4.1 of Chapter 1. Recall that we considered a Defender ($D$, she) who chooses her defense $d \in \mathcal{D}$. Then, an Attacker ($A$, he) chooses his attack $a \in \mathcal{A}$, after having observed $d$. Both $\mathcal{D}$ and $\mathcal{A}$ are assumed finite, unless noted. Figure 1.2 shows the corresponding bi-agent influence diagram. The consequences for both agents depend on the outcome $\theta \in \Theta$ of the attack. The agents have their own assessment on the outcome probability, respectively $p_D(\theta \,|\, d, a)$ and $p_A(\theta \,|\, d, a)$, dependent on $d$ and $a$. The Defender's utility function $u_D(d, \theta)$ depends on her chosen defense and the attack result. Similarly, the Attacker's utility function is $u_A(a, \theta)$.

As noted in Section 1.4.1, if the game is under complete information, the basic game-theoretic solution does not require the Attacker to know the Defender's probabilities and utilities, as he observes her actions. However, the Defender must know $(u_A, p_A)$, the common knowledge assumption in this case. Then both agents' expected utilities are computed at node $\Theta$, $\psi_A(d, a) = \int u_A(a, \theta)\, p_A(\theta \mid d, a)\, \mathrm{d}\theta$ and $\psi_D(d, a) = \int u_D(d, \theta)\, p_D(\theta \,|\, d, a)\, \mathrm{d}\theta$. Next, the Attacker's best response to $D$'s action $d$, is $a^*(d) = \mathrm{argmax}_{a \in \mathcal{A}}\ \psi_A(d, a)$. This is used to find the Defender's optimal action $d^*_{\mathrm{GT}} = \mathrm{argmax}_{d \in \mathcal{D}}\ \psi_D(d, a^*(d))$. The pair $(d^*_{\mathrm{GT}},\, a^*(d^*_{\mathrm{GT}}))$ is a Nash equilibrium and, indeed, a sub-game perfect equilibrium (Hargreaves-Heap and Varoufakis 2004).

As mentioned in Chapter 4, the solution of such games requires solving a bilevel optimization problem (Bard 1991), which can rarely be solved analytically. Existing numerical techniques lack scalability: increasing the number of upper level variables produces an exponential increase on the number of lower level tasks required. However, problems in emerging areas such as cybersecurity and adversarial machine learning (Ríos Insua et al. 2019) may require dealing with high dimensional and/or continuous decision spaces, and, consequently, can hardly be solved using standard methods. Some scalable gradient based solution approaches have been introduced in 4. However,

they are restricted to games in which expected utilities can be computed analytically. When this is not the case, MC simulation methods, see e.g. Ponsen et al. (2011) and Johanson et al. (2012) for pointers, could be used as briefly described next.

### 5.2.1 Monte Carlo simulation for games

Simulation based methods for sequential games typically approximate expected utilities using MC and, then, optimize with respect to decision alternatives, first to approximate Attacker's best responses, then to approximate the optimal defense. Algorithm 10 reflects a generic MC based approach to solve non-cooperative games where $Q$ and $P$ are the sample sizes required to respectively approximate the expected utilities $\psi_A(d, a)$ and $\psi_D(d, a)$ to the desired precision, as discussed in Appendix A.1. Convergence of Algorithm 10, detailed in A.1, follows under mild conditions and is

**Algorithm 10:** MC approach for non-cooperative sequential games with complete information

**input:** $P$, $Q$
**for** $d \in \mathcal{D}$ **do**
  **for** $a \in \mathcal{A}$ **do**
    Generate samples $\theta_1, \ldots, \theta_Q \sim p_A(\theta \mid d, a)$;
    Compute $\widehat{\psi}_A(d, a) = \frac{1}{Q} \sum_i u_A(a, \theta_i)$;
  **end**
  Find $a^*(d) = \text{argmax}_a \, \widehat{\psi}_A(d, a)$;
  Generate samples $\theta_1, \ldots, \theta_P \sim p_D(\theta \mid d, a^*(d))$;
  Compute $\widehat{\psi}_D(d) = \frac{1}{P} \sum_i u_D(d, \theta_i)$;
**end**
Compute $\widehat{d}^*_{\text{GT}} = \text{argmax}_d \, \widehat{\psi}_D(d)$;

based on two applications (for the inner and outer loops within Algorithm 10) of a uniform version of the strong law of large numbers (SLLN) (Jennrich 1969). It shows uniform convergence to the expected utilities as well as to the attacker's best responses and defender's optimal decision.

From a computational perspective, the algorithm requires generating $|\mathcal{D}| \times (|\mathcal{A}| \times Q + P)$ samples, where $|\cdot|$ designates the cardinality of the corresponding set, in addition to the cost of the final optimization and $|\mathcal{D}|$ inner loop optimizations. When the decision sets are continuous, they need to be discretized to solve the problem to the desired precision, as exemplified in Section 5.4.2. In the end, MC approaches could turn out to be computationally expensive when dealing with decision dependent uncertainties, as is the case in the games in this study: they require sampling from $p_D(\theta \mid d, a)$ and $p_A(\theta \mid d, a)$ for each possible pair of $d$ and $a$, entailing loops over the decision spaces $\mathcal{D}$ and $\mathcal{A}$. When these are high dimensional, considering the whole decision space as in MC will typically be inefficient. APS mitigates this issue.

### 5.2.2 Augmented probability simulation for games

APS solves for maximization of expected utility by converting the tasks of sequential estimation and optimization into simulation from an augmented distribution in the joint space of decisions and outcomes, not requiring a separate optimization step. Bielza et al. (1999) introduced it to solve decision analysis problems and Ekin et al. (2014) extended it to solve constrained stochastic optimization models. It can be advantageous in problems with expected utility surfaces that are expensive to estimate rendering the optimization step inefficient. In this chapter, we first use APS to solve sequential games dealing with the Attacker's and Defender's decision problems sequentially.

For the Attacker, we introduce an augmented distribution $\pi_A(a, \theta \mid d)$ over $(a, \theta)$ for a given defender action $d$, defined as proportional to the product of the utility function and the original distribution, $u_A(a, \theta)\, p_A(\theta \mid d, a)$. If $u_A(a, \theta)$ is positive and $u_A(a, \theta)\, p_A(\theta \mid d, a)$ is integrable, then $\pi_A(a, \theta \mid d)$ is a well-defined distribution. Simulating from it solves simultaneously for the expectation of the objective function and its optimization since its marginal over actions $a$, given by $\pi_A(a \mid d) = \int \pi_A(a, \theta \mid d) d\theta$, is proportional to the Attacker's expected utility $\psi_A(d, a) = \int u_A(a, \theta)\, p_A(\theta \mid d, a)\, \mathrm{d}\theta$. Consequently, the Attacker's best response given $d$ can be computed as $a^*(d) = \text{mode}\,[\pi_A(a \mid d)]$. Using backward induction, and assuming that $u_D(d, \theta)$ is positive and $u_D(d, \theta) p_D(\theta \mid d, a)$ is integrable, the Defender's problem is solved sampling from the augmented distribution $\pi_D(d, \theta \mid a^*(d)) \propto u_D(d, \theta) p_D(\theta \mid d, a^*(d))$: its marginal $\pi_D(d \mid a^*(d))$ in $d$ is proportional to the Defender's expected utility $\psi_D(d, a^*(d))$ and, consequently, $d^*_{\mathrm{GT}} = \text{mode}\,[\pi_D(d \mid a^*(d))]$. This leads to a solution approach for non-cooperative games that includes the steps of sampling from the augmented distributions, marginalising to the corresponding decision variables and estimating the mode of the marginal sample.

It is generally impossible to sample directly from the augmented distributions. However, Markov chain Monte Carlo (MCMC) methods, e.g. Gamerman and Lopes (2006), serve for such purpose. They construct a Markov chain in the space of the target distribution, the augmented distributions in our case, guaranteed to converge in distribution to the target under mild conditions. After convergence is detected, samples from the chain can be used as approximate samples from the target. Various approaches are available to construct the chains. For instance, Appendix B.1 discusses Gibbs based algorithms. Here we adopt more versatile Metropolis-Hastings (MH) variants (Chib and Greenberg 1995) as in Algorithm 11. This facilitates sampling approximately from $\pi_D(d, \theta \mid a^*(d))$ (outer APS) to solve the Defender's problem. Within that, the Attacker's best response $a^*(d)$ is estimated for any given $d$ using another APS (inner APS) on $\pi_A(a, \theta \mid d)$. Details of the acceptance/rejection step follow. Let $d$ and $\theta$ be the current samples in the MH scheme of the outer APS. Within each iteration, a candidate $\tilde{d}$ for the Defender's decision is sampled from a proposal generating distribution $g_D(\tilde{d} \mid d)$. We choose this to be symmetric in the sense that it satisfies $g_D(\tilde{d} \mid d) = g_D(d \mid \tilde{d})$. Then, the Attacker's problem is solved using

an inner APS to estimate $a^*(\tilde{d})$. The state $\theta$ is next sampled using $p_D(\theta \mid \tilde{d}, a^*(\tilde{d}))$. The candidate samples are accepted with probability $\pi_D(\tilde{d}, \tilde{\theta} \mid a^*(\tilde{d}))/\pi_D(d, \theta \mid a^*(d))$, which, after simplification, adopts the form $\frac{u_D(\tilde{d},\tilde{\theta})}{u_D(d,\theta)}$. Algorithm 11 thus defines a

**Algorithm 11:** MH APS for non-cooperative sequential games with complete information.

**function** `solve_attacker(`$M$, $d$, $g_A$`)`**:**
  **initialize :** $a^{(0)}$
  Draw $\theta^{(0)} \sim p_A(\theta \mid d, a^{(0)})$;
  **for** $i = 1$ **to** $M$ **do** ▷ `Inner APS`
    Propose new attack $\tilde{a} \sim g_A(\tilde{a} \mid a^{(i-1)})$.;
    Draw $\tilde{\theta} \sim p_A(\theta \mid d, \tilde{a})$;
    Evaluate acceptance probability $\alpha = \min\left\{1, \frac{u_A(\tilde{a},\tilde{\theta})}{u_A(a^{(i-1)},\theta^{(i-1)})}\right\}$;
    With probability $\alpha$ set $a^{(i)} = \tilde{a}$, $\theta^{(i)} = \tilde{\theta}$. Otherwise, set $a^{(i)} = a^{(i-1)}$, and $\theta^{(i)} = \theta^{(i-1)}$.;
  **end**
  Discard the first $K$ samples and estimate mode of rest of draws $\{a^{(i)}\}$. Record it as $a^*(d)$.;
  **return** $a^*(d)$;
**input:** $d$, $M$, $K$, $N$, $R$, $g_D$ and $g_A$ symmetric proposal distributions
**initialize :** $d^{(0)}$, $a^*(d^{(0)})$ = `solve_attacker(`$M$, $d^{(0)}$, $g_A$`)`
Draw $\theta^{(0)} \sim p_D(\theta \mid d^{(0)}, a^*(d^{(0)}))$;
**for** $i = 1$ **to** $N$ **do** ▷ `Outer APS`
  Propose new defense $\tilde{d} \sim g_D(\tilde{d} \mid d^{(i-1)})$;
  $a^*(\tilde{d})$ = `solve_attacker(`$M$, $\tilde{d}$, $g_A$`)` if not previously computed;
  Draw $\tilde{\theta} \sim p_D(\theta \mid \tilde{d}, a^*(\tilde{d}))$.;
  Evaluate acceptance probability $\alpha = \min\left\{1, \frac{u_D(\tilde{d},\tilde{\theta})}{u_D(d^{(i-1)},\theta^{(i-1)})}\right\}$;
  With probability $\alpha$ set $d^{(i)} = \tilde{d}$, $a^*(d^{(i)}) = a^*(\tilde{d})$ and $\theta^{(i)} = \tilde{\theta}$. Otherwise, set $d^{(i)} = d^{(i-1)}$, and $\theta^{(i)} = \theta^{(i-1)}$.;
**end**
Discard first $R$ samples and estimate mode of rest of draws $\{d^{(i)}\}$. Record it as $\widehat{d}^*_{\mathrm{GT}}$.;

Markov chain in $(d, \theta)$ such that $(d^{(N)}, \theta^{(N)}) \xrightarrow{d} \pi_D(d, \theta \,|\, a^*(d))$ where $\xrightarrow{d}$ represents convergence in distribution. Proposition 1 provides necessary conditions for the convergence of its output to the decision $d^*_{\mathrm{GT}}$.

**Proposition 1.** *If the Attacker's and Defender's utility functions are positive; $p_A(\theta|\, d, a)$, $p_D(\theta|\, d,\, a) > 0\ \forall a, \theta$; $u_A(a, \theta)p_A(\theta|\, d, a)$ and $u_D(d, \theta)p_D(\theta|\, d,\, a)$ are integrable; $\mathcal{A}$, $\mathcal{D}$, $\Theta$ are either discrete sets or intervals in $\mathbb{R}^n$; and the proposal generating distributions $g_A$ and $g_D$ are symmetric, Algorithm 11 defines a Markov Chain with stationary distribution $\pi_D(d, \theta \mid a^*(d))$, and a consistent mode estimator based on its marginal samples in $d$ converges to $d^*_{GT}$ almost surely.*

*Proof.* Under the hypothesis, for each $d$, $\pi_A(a, \theta \mid d)$ is a well-defined distribution and the samples generated within the inner APS loop in Algorithm 11 define a

Markov chain with $\pi_A(a, \theta \mid d)$ as stationary distribution (Gamerman and Lopes 2006). Once convergence is detected (at iteration $K$), the remaining $M - K$ marginal samples $a^{(i)}$ of the Markov chain are approximate samples from $\pi_A(a \mid d)$. For large enough $M - K$, a consistent sample mode estimator (in the sense of Romano (1988)) converges almost surely to $a^*(d)$. Similarly, under the hypothesis, $\pi_D(d, \theta \mid a^*(d))$ is a well-defined distribution and the samples generated in the outer APS loop in Algorithm 11 define a Markov chain with stationary distribution $\pi_D(d, \theta \mid a^*(d))$. Once MCMC convergence is detected (at iteration $R$), the remaining $N - R$ marginal samples $d^{(i)}$ of the Markov chain are approximate samples from $\pi_D(d \mid a^*(d))$. Hence, their sample mode estimator converges to $d^*_{\mathrm{GT}}$ almost surely (Romano 1988).

□

In practice, as continuous candidate proposal generation $g_D$ and $g_A$ distributions, we use heavy tailed $t$ distributions, centered at the current solutions (Gamerman and Lopes 2006). When facing discrete or ordinal decisions, we display those in a circular list and generate from neighbouring states with equal probability. Practical convergence (for discarding the first $K$ or $R$ samples) may be assessed with various statistics like Brooks-Gelman-Rubin's (BGR) (Brooks and Roberts 1998). Once the chain is judged to have converged, the initial samples are discarded as burn-in and the remaining simulated values are used as an approximate sample from the distribution of interest. In particular, the marginal draws $d^{(R+1)}, ..., d^{(N)}$ would correspond to an approximate sample from the marginal $\pi_D(d \mid a^*(d))$. The sample mode must be estimated with a consistent estimator in the sense of Romano (1988), see also the classical work in Parzen (1962) and Grenander (1965).

Computationally, Algorithm 11 removes the loops over both $\mathcal{D}$ and $\mathcal{A}$. Thus, its complexity does not depend on the dimensions of those decision spaces providing an intrinsic advantage over MC approaches in problems with large or continuous spaces. In particular, Algorithm 11 requires $N \times (2 \times M + 3) + 2M + 2$ samples plus the cost of convergence checks and (at most) $N + 1$ mode approximations.

### 5.2.3 Sampling from a power transformation of the marginal augmented distribution

In cases in which either the Attacker's or the Defender's expected utility surfaces are very flat, identifying the mode of their corresponding marginal augmented distributions could be very challenging. Very flat expected utilities result in flat marginal augmented distribution, that requires many samples (APS iterations) to resolve the mode. In these cases, one possibility to increase efficiency of APS in finding the mode is to sample from a power transformation of the marginal augmented distribution (more peaked round the mode) rather than sampling from the original distribution. Next, we provide a demonstration of such sampling and its convergence guarantees. For the sake of illustration, we just deal with the Attacker's problem (inner APS in Algorithm 11) for a given defense $d$. The application of this new sampling approach to the Defender's problem is straightforward.

Assume we are interested in maximizing the Attacker's expected utility for a given defense $d$, $\psi_A(a,\theta) = \int u_A(a,\theta)p_A(\theta|d,a)\,\mathrm{d}\theta$. We define an augmented distribution $\pi_H(a,\cdot)$ such that its marginal distribution in $a$ is proportional to $\psi_A^H(a,\theta)$, with $H \geq 1$ defined as the augmentation parameter. Now, $\psi_A^H(a,\theta)$ is more peaked around the optimal attack and, consequently the marginal on $a$ of $\pi_H(a,\cdot)$ will be more peaked around such mode, thus facilitating its identification. Let us define this augmented distribution creating $H$ identical copies of the state $\theta$ and using

$$\pi_H(a,\theta_1,\ldots,\theta_H|d) \propto \prod_{i=1}^{H} u_A(a,\theta_i)p_A(\theta_i|d,a).$$

(Note that when $H=1$ we recover the original formulation of APS). Its marginal in $a$ is

$$\pi_H(a|d) \propto \left[\int u_A(a,\theta)p_A(\theta|d,a)\,\mathrm{d}\theta\right]^H = \psi_A^H(a,\theta)$$

as requested. Samples from this marginal will cluster tightly around the mode as $H$ increases, since this distribution will be more peaked. To get those samples, for a given value of $H$, we sample from $\pi_H(a,\theta_1,\ldots,\theta_H|d)$ and just keep the samples of $a$, using the following MH scheme: suppose the current state of the Markov chain is $(a,\theta_1,\ldots,\theta_H)$; generate a candidate $\tilde{a}$ from a symmetric proposal $g_A(\cdot|a)$, and propose $\tilde{\theta}_i \sim p_A(\theta|d,\tilde{a})$ for $i=1,\ldots,H$; accept $\tilde{a},\tilde{\theta}_1,\ldots,\tilde{\theta}_H$ with probability

$$\min\left\{1, \prod_{i=1}^{H}\frac{u_A(\tilde{a},\tilde{\theta}_i)}{u_A(a,\theta_i)}\right\}, \tag{5.2.1}$$

otherwise, maintain the current $(a,\theta_1,\ldots,\theta_H)$. This defines a Markov chain with stationary distribution $\pi_H(a,\theta_1,\ldots,\theta_H|d)$ (Tierney 1994). Indeed, the acceptance probability of a Metropolis-Hastings scheme for a target distribution $\pi_H(a,\theta_1,\ldots,\theta_H|d)$ with proposal distribution $q(\tilde{a},\tilde{\theta}_1,\ldots,\tilde{\theta}_H|a,\theta_1,\ldots,\theta_H) = g_A(\tilde{a}|a)\prod_{i=1}^{H}p_A(\tilde{\theta}_i|d,\tilde{a})$ is

$$\begin{aligned}
& \min\left\{1, \frac{\pi_H(\tilde{a},\tilde{\theta}_1,\ldots,\tilde{\theta}_H)|d}{\pi_H(a,\theta_1,\ldots,\theta_H|d)}\cdot\frac{q(a,\theta_1,\ldots,\theta_H|\tilde{a},\tilde{\theta}_1,\ldots,\tilde{\theta}_H)}{q(\tilde{a},\tilde{\theta}_1,\ldots,\tilde{\theta}_H|a,\theta_1,\ldots,\theta_H)}\right\} \\
= & \min\left\{1, \frac{\prod_{i=1}^{H}u_A(\tilde{a},\tilde{\theta}_i)p_A(\tilde{\theta}_i|d,\tilde{a})}{\prod_{i=1}^{H}u_A(a,\theta_i)p_A(\theta_i|d,a)}\cdot\frac{g_A(a|\tilde{a})\prod_{i=1}^{H}p_A(\theta_i|d,a)}{g_A(\tilde{a}|a)\prod_{i=1}^{H}p_A(\tilde{\theta}_i|d,\tilde{a})}\right\} \\
= & \min\left\{1, \prod_{i=1}^{H}\frac{u_A(\tilde{a},\tilde{\theta}_i)}{u_A(a,\theta_i)}\right\}.
\end{aligned}$$

Thus, samples from $\pi_H(a,\theta_1,\ldots,\theta_H|d)$ can be generated running this MH scheme for a certain number of iterations, and discarding the initial ones as burn-in samples. The rest of $a$ samples are approximately distributed as $\pi_H(a|d)$. As a consequence, the sample mode estimator computed using these samples converges almost surely to the optimal attack $a^*(d)$, the mode of $\pi_H(a|d)$. A similar procedure could be defined for the Defender's problem to increase efficiency in finding $d^*_{GT}$. The inner and outer APS of Algorithm 11, could thus be substituted by sampling mechanisms as the

one explained to facilitate mode identifications. Convergence of this new approach occurs under the same conditions of Proposition 1 and follows from the fact that the defined Markov chains have the desired stationary distributions.

When the decision sets are high dimensional, direct application of the previous approach might not be feasible as mode identification becomes even more challenging. Müller et al. (2004) offer a solution for this issue that consists of embedding the previous MH scheme within an annealing schedule. Thus, within each APS iteration we increase $H$ using an appropriate cooling schedule. This produces an inhomogeneous Markov chain that converges to the mode of $\pi_H(a|d)$ (the optimal attack). A proof can be found in Müller et al. (2004). The same strategy could be used for the Defender's APS.

**Example.**

We demonstrate the framework with a simple cybersecurity problem. An organization (Defender) has to choose among ten security protocols: $d = 0$ (no extra defensive action); $d = i$ (level $i$ protection protocol with increasing protection), $i = 1, \ldots, 8$; $d = 9$ (a safe but cumbersome protocol). The Attacker has two alternatives: attack ($a = 1$) or not ($a = 0$). Successful (unsuccessful) attacks are denoted with $\theta = 1$ ($\theta = 0$). Clearly, when there is no attack, $\theta = 0$.

| | $\theta$ | |
|---|---|---|
| $d$ | 0 | 1 |
| 0 | 0.05 | 7.05 |
| 1 | 0.10 | 7.10 |
| 2 | 0.15 | 7.15 |
| 3 | 0.20 | 7.20 |
| 4 | 0.25 | 7.25 |
| 5 | 0.30 | 7.30 |
| 6 | 0.35 | 7.35 |
| 7 | 0.40 | 7.40 |
| 8 | 0.45 | 7.45 |
| 9 | 0.50 | 7.50 |

(a)

| | $a$ | |
|---|---|---|
| $d$ | 0 | 1 |
| 0 | 0.0 | 0.50 |
| 1 | 0.0 | 0.40 |
| 2 | 0.0 | 0.35 |
| 3 | 0.0 | 0.30 |
| 4 | 0.0 | 0.25 |
| 5 | 0.0 | 0.20 |
| 6 | 0.0 | 0.15 |
| 7 | 0.0 | 0.10 |
| 8 | 0.0 | 0.05 |
| 9 | 0.0 | 0.01 |

(b)

| | $\theta$ | |
|---|---|---|
| $a$ | 0 | 1 |
| 0 | 0.00 | 0.00 |
| 1 | -0.53 | 1.97 |

(c)

| $d$ | $\alpha_d$ | $\beta_d$ |
|---|---|---|
| 0 | 50.0 | 50.0 |
| 1 | 40.0 | 60.0 |
| 2 | 35.0 | 65.0 |
| 3 | 30.0 | 70.0 |
| 4 | 25.0 | 75.0 |
| 5 | 20.0 | 80.0 |
| 6 | 15.0 | 85.0 |
| 7 | 10.0 | 90.0 |
| 8 | 5.0 | 95.0 |
| 9 | 1.0 | 99.0 |

(d)

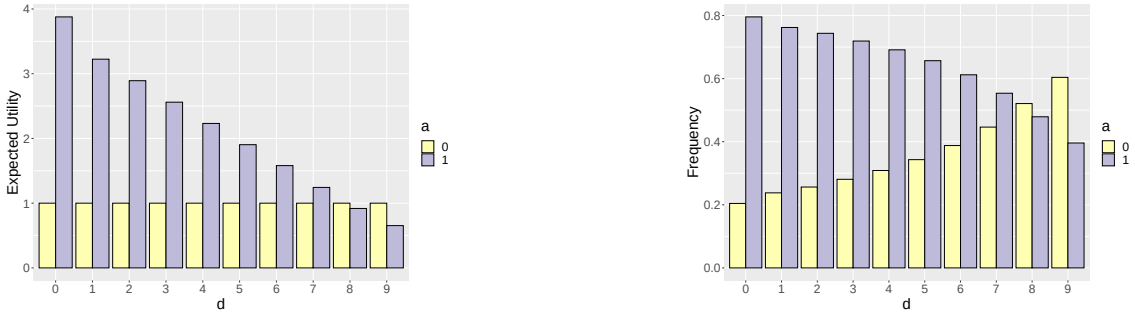
**Table 5.1**: (a) Defender's net costs; (b) Successful attack probabilities; (c) Attacker's net benefits; (d) Beta distribution parameters

**Defender non strategic judgments.** Table 5.1a presents net costs $c_D$ associated with each decision and outcome, covering a 7M€ business valuation, and 0.05M€ base security cost plus 0.05M€ per each security level increase. When the attack is successful, the defender loses the whole business value. The probability $p_D(\theta = 1 \,|\, d, a)$

of successful attack given $d$ and $a$ is in Table 5.1b (with complementary probabilities for unsuccessful attacks). The Defender is constant risk averse in costs, with utility strategically equivalent to $u_D(c_D) = -\exp(c \times c_D)$ with $c = 0.4$.

**Attacker judgments.** The average attack cost is estimated at 0.03M€. The average benefit (due to market share captured, ransom, etc.) is 2M€. An unsuccessful attack has an extra cost of 0.5M €. Table 5.1c presents the Attacker's net benefit $c_A$ associated with each attack and outcome. $D$ thinks that $A$ is constant risk prone over benefits. His utility is strategically equivalent to $u_A(c_A) = \exp(e \times c_A)$, with $e > 0$.

Start with the complete information case. To fix ideas, assume that $p_A(\theta = 1 \mid d, a) = p_D(\theta = 1 \mid d, a)$ (Table 5.1b) and $e = 1$. MC and APS approximate optimal



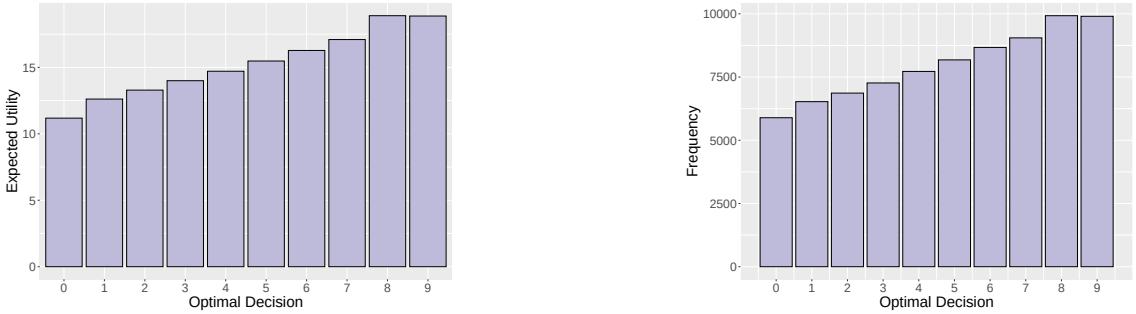

(a) MC solutions (b) APS solutions

**Figure 5.1**: Attacker problem solutions for each defense

decisions using Algorithms 10 and 11, respectively. First, Figure 5.1a represents MC estimates of $A$'s expected utility for each $d$ and $a$. The optimal response $a^*(d)$ for each $d$ is the alternative with maximum expected utility. For example, for $d = 5$, $A$'s optimal decision is to attack; for $d = 8$, he should not attack. Next, Figure 5.1b represents the frequencies of marginal samples of $a$ from the augmented distribution $\pi_A(a, \theta \mid d)$ for each $d$. Its mode coincides with the optimal attack decision. From $d = 0$ (no defense) until 7, the Attacker should attack. With stronger defenses $d = 8$ and $d = 9$, the mode is $a = 0$ and hence an attack is not advised. The Attacker's best responses $a^*(d)$ for each defense $d$ are thus identical with both approaches.

Armed with $a^*(d)$, the optimal defense is computed again using MC and APS. Figure 5.2a presents the MC estimation of $\psi_D(d, a^*(d))$ for each $d$. Figure 5.2b shows sample frequencies from the marginal augmented distribution $\pi_D(d \mid a^*(d))$. Both methods agree that $d^*_{\mathrm{GT}}$ is acquiring level 8 protection, with level 9 a close competitor.

It could be argued that finding the exact optimal decision is not that crucial since the expected utilities for protection levels 8 and 9 are very close. Moreover, as the expected utilities of $d = 8$ and $d = 9$ are very close, it is challenging to find the exact optimal decision. APS is useful in checking that, indeed, $d^*_{\mathrm{GT}} = 8$. We repeat the experiment using APS but replacing the defender's marginal augmented




(a) MC solution (b) APS solution

**Figure 5.2**: Solutions of Defender problem

distribution $\pi_D$ by its power transformation $\pi_D^H$, which is more peaked around the mode, as explained in Section 5.2.3. In addition, as in simulated annealing, within each APS iteration we increase $H$ using an appropriate cooling schedule (Müller et al. 2004). Figure 5.3 displays the results of the simulation when $H = 450$, showing that, indeed, the game-theoretic solution under complete information is $d^*_{\mathrm{GT}} = 8$.

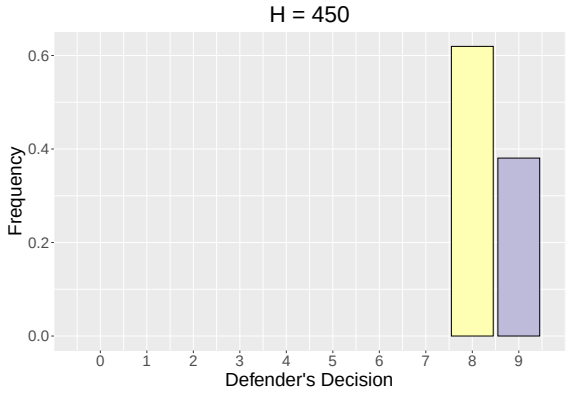



**Figure 5.3**: Defender optimal solutions for the game with complete information using power augmented distributions.

This also emphasizes another advantage of APS: despite eventual flatness of expected utility, APS provides a method to find the optimal solution with little extra computational cost. △

### 5.2.4 Sensitivity analysis for games

The Defender's judgments, expressed through $(u_D, p_D)$, could be argued to be well assessed, as she is the supported agent in the game. However, as cogently argued in Keeney (2007), our knowledge about $(u_A, p_A)$ may not be that precise as it would require $A$ to reveal his beliefs and preferences. This is doubtful in domains such as cybersecurity and counter terrorism where information is concealed and hidden to adversaries.

One could conduct a sensitivity analysis to mitigate this issue, considering that $A$'s preferences and beliefs are modeled through classes of utilities $u \in \mathcal{U}_A$ and probabilities $p \in \mathcal{P}_A$, summarizing the information available to $D$ possibly obtained from leakage, earlier interactions or informants. The stability of the proposed solution $d^*_{\text{GT}}$ could be assessed by comparing Nash defenses $d^*_{u,p}$ computed for each pair $(u, p)$. Several criteria have been proposed to assess the stability of solutions in the areas of decision making under uncertainty and sensitivity analysis (Insua 1990) and robust Bayesian analysis (Insua and Ruggeri 2012). Of them, we shall use the regret $r_{u,p}(d^*_{\text{GT}}) = \psi_D(d^*_{\text{GT}}, a^*(d^*_{\text{GT}})) - \psi_D(d^*_{u,p}, a^*(d^*_{u,p}))$, as it reflects the loss in expected utility for the choice of the proposed $d^*_{\text{GT}}$ instead of $d^*_{u,p}$ that should have been chosen for the actual judgements; $(u, p) \in \mathcal{U}_A \times \mathcal{P}_A$. Small values of $r_{u,p}(d^*_{\text{GT}})$ would indicate robustness with respect to the Attacker's utility and probability: any pair $(u, p) \in \mathcal{U}_A \times \mathcal{P}_A$ could be chosen with no significant changes in the attained expected utilities and $d^*_{\text{GT}}$ is thus robust. Otherwise, the relevance of the proposed Nash defense $d^*_{\text{GT}}$ should be criticized and further investigated. Operationally, a *threshold* on the maximum acceptable regret would be specified, as sketched in Algorithm 12.

**Algorithm 12:** Robustness assessment of solutions for games with complete information

**input:** $d^*_{\text{GT}}$, $\mathcal{U}_A$, $\mathcal{P}_A$, $R$, *threshold*
**for** $i = 1$ **to** $R$ **do**
  Randomly sample $u$ from $\mathcal{U}_A$ and $p$ from $\mathcal{P}_A$;
  Compute $d^*_{u,p}$ using Algorithm 11;
  Compute $r_{u,p}(d^*_{\text{GT}})$;
  **if** $r_{u,p}(d^*_{GT}) >$ *threshold* **then**
    Robustness requirements not satisfied;
    Stop
  **end**
**end**
Robustness requirements satisfied.

**Example (cont.)**

We next check the robustness of $d^*_{\text{GT}}$ with respect to the utility and probability assumptions. The optimal defense is computed for $10,000$ perturbations of $u_A(c_A)$ (sampling $e' \sim \mathcal{U}(0, 2)$ and using $u'_A(c_A) = \exp(e' \times c_A)$) and the probability $p_A(\theta \mid d, a = 1)$ of successful attack in case $A$ attacks for each $d$ (sampling from a Beta distribution with mean equal to the original value and variance 0.1% of the corresponding mean for each $d$). Figure 5.4 reflects the frequency with which each $d$ is found optimal. The proposed $d^*_{\text{GT}} = 8$ emerges 25% of the times as optimal. However, it is unstable as inducing small perturbations in the utilities and probabilities leads to other solutions: $d = 9$ appears 42% of the times as optimal and $d = 7$, 16%. More importantly, large variations in optimal expected utilities are observed 33% of the time, with maximum regret 42.5% of the total optimal expected utility: $d^*_{\text{GT}} = 8$ seems too sensitive to changes in $u_A$ and $p_A$. △

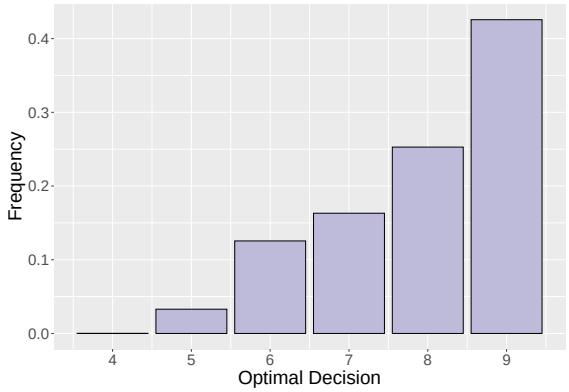



**Figure 5.4**: Sensitivity analysis of the solution of the game with complete information

This will be addressed next by relaxing the complete information assumption.

## 5.3 Sequential non-cooperative games with incomplete information: ARA

When the game theoretic solution lacks robustness or the complete information assumption does not hold, the problem may be handled as a game with incomplete information. The most common approach in such games is based on the BNE concept. Alternatively, as in Section 1.4.1, we use a decision analytic approach based on ARA. Rios and Rios Insua (2012) discuss the differences between both concepts in simultaneous games showing that they may lead to different solutions. An interesting feature of ARA is that it mitigates the common prior assumption (Antos and Pfeffer 2010). We describe the relation between both solution concepts in sequential games below.

As highlighted in Section 1.4.1, ARA considers that the Defender actually has uncertainty about $(u_A, p_A)$. Her problem was depicted in Figure 1.3a as an influence diagram, where $A$'s action appears as an uncertainty. Her expected utility is $\psi_D(d) = \int \psi_D(d,a)\, p_D(a \mid d)\, \mathrm{d}a$ which requires $p_D(a \mid d)$, her assessment of the probability that the Attacker will choose $a$ after having observed $d$. Then, her optimal decision is $d^*_{\mathrm{ARA}} = \mathrm{argmax}_{d\in\mathcal{D}}\ \psi_D(d)$. Our example will show a solution that does not coincide with a Nash equilibrium.

Eliciting $p_D(a \mid d)$, which has a strategic component, is facilitated by analyzing $A$'s problem from $D$'s perspective, depicted in Figure 1.3b. For that, she would use all information and judgment available about $A$'s utilities and probabilities. However, instead of using point estimates for $u_A$ and $p_A$ to find $A$'s best response $a^*(d)$ given $d$ as in Section 5.2, her uncertainty about the attacks would derive from her uncertainty about $(u_A, p_A)$ modeled through a distribution $F = (U_A, P_A)$ on the space of utilities and probabilities. Without loss of generality, assume that both $U_A$ and $P_A$ are defined over a common probability space $(\Omega, \mathcal{A}, \mathcal{P})$ with atomic elements $\omega \in \Omega$ (Chung 2001). This induces a distribution over the Attacker's expected utility $\psi_A(d, a)$, where the ran-

dom expected utility for $A$ would be $\Psi_A^\omega(d,a) = \int U_A^\omega(a,\theta) P_A^\omega(\theta \,|\, d,a)\,\mathrm{d}\theta$. In turn, this induces a random optimal alternative defined through $A^*(d)^\omega = \mathrm{argmax}_{x\in\mathcal{A}}\, \Psi_A^\omega(d,x)$. Then, the Defender would find $p_D(a \mid d) = \mathbb{P}_F\left[A^*(d) = a\right] = \mathcal{P}(\omega : A^*(d)^\omega = a)$ in the discrete case (and, similarly in the continuous one). Observe that $\omega$ and $\mathcal{P}$ could be re-interpreted, respectively, as the type and the common prior in Harsanyi's doctrine. Then, $P_A^\omega$ and $U_A^\omega$ respectively correspond to $A$'s probability and utility given his type, and $(d^*, \{A^*(d^*)^\omega\})$ would constitute a BNE. Thus, in the sequential Defend-Attack game, we can operationally reinterpret the ARA approach in terms of Harsanyi's, although the underlying principles are different. Computationally, ARA models entail integration and optimization procedures that can be challenging in many cases. Therefore, we explore simulation based methods for ARA.

### 5.3.1 MC based approach for ARA

MC simulation approximates $p_D(a \mid d)$ for each $d$, drawing $J$ samples $\{(u_A^i, p_A^i)\}_{i=1}^J$ from $F$ and setting $\hat{p}_D(a|d) = \frac{\#\{A^*(d)=a\}}{J}$ where $A^*(d) = \mathrm{argmax}_a \int u_A^i(a,\theta) p_A^i(\theta|d,a)\,\mathrm{d}\theta$. This is then used as an input to the Defender's expected utility maximization, as reflected in Algorithm 13.

**Algorithm 13:** MC based approach to solve the ARA problem

**input:** $J$, $P$, $Q$
**for** $d \in \mathcal{D}$ **do**
  **for** $i = 1$ **to** $J$ **do**
    Sample $u_A^i(a,\theta) \sim U_A^\omega(a,\theta)$, $p_A^i(\theta \mid d,a) \sim P_A^\omega(\theta \mid d,a)$;
    **for** $a \in \mathcal{A}$ **do**
      Generate samples $\theta_1, \ldots, \theta_Q \sim p_A^i(\theta \mid d,a)$;
      Approximate $\widehat{\psi}_A^i(d,a) = \frac{1}{Q}\sum u_A^i(a,\theta_i)$;
    **end**
    Find $a_i^*(d) = \mathrm{argmax}_a\, \widehat{\psi}_A^i(d,a)$;
  **end**
  $\hat{p}_D(a \mid d) = \frac{1}{J}\sum_{i=1}^J I[a_i^*(d) = a]$;
**end**
**for** $d \in \mathcal{D}$ **do**
  Generate samples $(\theta_1, a_1), \ldots, (\theta_P, a_P) \sim p_D(\theta \mid d,a)\hat{p}_D(a \mid d)$ ;
  Approximate $\widehat{\psi}_D(d) = \frac{1}{P}\sum u_D(d,\theta_i)$;
**end**
Compute $\widehat{d}^*_{\mathrm{ARA}} = \mathrm{argmax}_d\, \widehat{\psi}_D(d)$;

From a computational perspective, it requires generating $|\mathcal{D}| \times [J \times (|\mathcal{A}| \times Q + 2) + 2P]$ samples where $Q$ and $P$ are the number of samples required to respectively approximate $\int u_A^i(a,\theta)\, p_A^i(\theta \mid d,a)\,\mathrm{d}\theta$ and $\int\int u_D(d,\theta)\, p_D(\theta \mid d,a)\, \hat{p}_D(a \mid d)\,\mathrm{d}\theta\,\mathrm{d}a$ to the desired precision. Convergence follows from two applications of a uniform version of the SLLN as reflected in Appendix A.2. In high dimensional cases, and when model uncertainty dominates, methods that automatically focus on high-probability-high impact events could be faster and more robust. Hence, APS to solve ARA is

investigated as a scalable alternative to MC.

### 5.3.2 APS for ARA

APS solves the ARA model by constructing augmented distributions for the Attacker's and Defender's problems. To solve the Attacker's decision problem, this study constructs an APS in the state space of the Attacker's random utilities and probabilities. For a given $d$, this study builds the random augmented distribution $\Pi_A^\omega(a, \theta \mid d) \propto U_A^\omega(a, \theta) P_A^\omega(\theta \mid d, a)$, whose marginal $\Pi_A^\omega(a \mid d) = \int \Pi_A^\omega(a, \theta \mid d)\, \mathrm{d}\theta$ is proportional to the random expected utility $\Psi_A^\omega(d, a)$. Then, the random optimal attack $A^*(d)^\omega$ coincides almost surely with the mode of the marginal $\Pi_A^\omega(a \mid d)$ of this random augmented distribution. Consequently, by sampling $u_A(a, \theta) \sim U_A(a, \theta)$ and $p_A(\theta \mid d, a) \sim P_A(\theta \mid d, a)$, one can build $\pi_A(a, \theta \mid d) \propto u_A(a, \theta) p_A(\theta \mid d, a)$, which is a sample from $\Pi_A(a, \theta \mid d)$. Then, $\mathrm{mode}(\pi_A(a \mid d))$ is a sample of $A^*(d)$, whose distribution is $\mathbb{P}_F\left[A^*(d) = a\right] = p_D(a \mid d)$. Thus, this study provides a mechanism to sample from $p_D(a \mid d)$.

Next, using backward induction, an augmented distribution for the Defender's problem is introduced as $\pi_D(d, a, \theta) \propto u_D(d, \theta)\ p_D(\theta \mid d, a)\ p_D(a \mid d)$. Its marginal $\pi_D(d) = \int\int \pi_D(d, a, \theta)\, \mathrm{d}a\, \mathrm{d}\theta$ is proportional to the expected utility $\psi_D(d)$ and, consequently, $d^*_{\mathrm{ARA}} = \mathrm{mode}\left(\pi_D(d)\right)$. Thus, one just needs to sample $(d, a, \theta) \sim \pi_D(d, a, \theta)$ and estimate its mode in $d$.

Algorithm 14 summarizes a nested MH based procedure for APS. Let $d$, $a$ and $\theta$ be the current state of the Markov Chain. This study samples a candidate defense $\tilde{d}$ from a proposal generating distribution $g_D(\tilde{d} \mid d)$, a candidate $\tilde{a}$ from $p_A(a \mid \tilde{d})$ using the Attacker's APS as explained before, and $\tilde{\theta} \sim p_D(\theta \mid \tilde{d}, \tilde{a})$. These samples are accepted with probability $\alpha = \min\left\{1, \frac{u_D(\tilde{d}, \tilde{\theta})}{u_D(d, \theta)}\right\}$. The stationary distribution of this Markov Chain converges to $\pi_D(d, a, \theta)$ as reflected in Proposition 2, which provides conditions for the convergence of the output of Algorithm 14 to the optimal decision $d^*_{\mathrm{ARA}}$.

**Proposition 2.** *If $u_D$ and almost all the utilities in the support of $U_A$ are positive; $p_D(\theta \mid d, a)$ and almost all distributions in the support of $P_A(\theta \mid d, a)$ are also positive; the products of utilities and probabilities are integrable; $\mathcal{A}$, $\mathcal{D}$ and $\Theta$ are either discrete or intervals in $\mathbb{R}^n$; and the proposal generating distributions $g_A$ and $g_D$ are symmetric, Algorithm 14 defines a Markov Chain with stationary distribution $\pi_D(d, \theta, a)$. Moreover, the mode of the marginal samples of d from this Markov chain approximates the solution $d^*_{ARA}$.*

*Proof.* Given our hypothesis about $U_A^\omega(a, \theta)$ and $P_A^\omega(\theta \mid d, a)$ for each $d$, the distributions $\pi_A(a, \theta \mid d) \sim \Pi_A^\omega(a, \theta \mid d)$ given by $\pi_A(a, \theta \mid d) \propto u_A(a, \theta) p_A(\theta \mid a, d)$ with $u_A \sim U_A^\omega$ and $p_A \sim P_A^\omega$, are well defined a.s. Moreover, the samples $a$ generated through `sample_attack` in Algorithm 14 are distributed according to $\mathbb{P}_F\left[A^*(d) = a\right] = p_D(a \mid d)$. Indeed, as we are sampling $u_A(a, \theta) \sim U_A^\omega(a, \theta)$ and $p_A(\theta \mid d, a) \sim P_A^\omega(\theta \mid d, a)$; $\pi_A(a, \theta \mid d) \propto u_A(a, \theta) p_A(\theta \mid d, a)$, is a sample from $\Pi_A^\omega(a, \theta \mid d)$. Then, $\mathrm{mode}(\pi_A(a \mid d))$ is a sample from $A^*(d)^\omega$, whose distribution

is $\mathbb{P}_F[A^*(d) = a]$. To compute this mode, we sample $a, \theta \sim \pi_A(a, \theta \mid d)$ using MH, defining a Markov chain $\{a^{(i)}, \theta^{(i)}; i = 1, ..., M\}$ (loop in function `sample_attack`) whose stationary distribution is $\pi_A(a, \theta \mid d)$, (Roberts and Smith 1994). Once MCMC convergence is assessed, the first $K$ samples of $a$ are discarded as burn-in samples and the remaining $M - K$ marginal samples are approximate samples from $\pi_A(a \mid d)$. For large enough $M - K$, the mode of such marginal samples approximates the mode of $\pi_A(a, \theta \mid d)$, based on a consistent mode estimator (Romano 1988).

Next, as $u_D$ is positive and $u_D(d, \theta)p_D(\theta|d, a)p_D(a|d)$ integrable, $\pi_D(d, a, \theta)$ is well-defined and is the stationary distribution of the Markov chain defined by the outer APS in Algorithm 14. Once MCMC convergence is detected, the first $R$ samples of $d$ are discarded as burn-in and the remaining $N - R$ samples are approximately distributed as $\pi_D(d)$. Hence, a consistent mode estimation of these samples approximates $d^*_{\text{ARA}}$. □

Computationally, Algorithm 14 requires generating $N \times (2M + 5) + 2M + 4$ samples from multivariate distributions in addition to the cost of the convergence checks and mode computations. Again, this algorithm removes the need for loops over $\mathcal{A}$ and $\mathcal{D}$. This would be an excellent choice when facing a problem where the cardinality of these spaces is large or the decisions are continuous. Note though that, in the discrete case, an alternative could be to combine MC and APS. If the cardinality of the $\mathcal{D}$ is low, it could be more convenient to estimate the value of $p_D(a \mid d)$ for each $d$, drawing $J$ samples $a \sim p_D(a \mid d)$ and counting frequencies as in Section 5.3.1. Then, in the Defender's APS, instead of invoking the attacker's one each time we need a sample $a \sim p_D(a \mid d)$, we would directly sample from the estimate $\hat{p}_D(a \mid d)$. Finally, Appendix B.2 provides a Gibbs based algorithm.

**Example (cont.)**

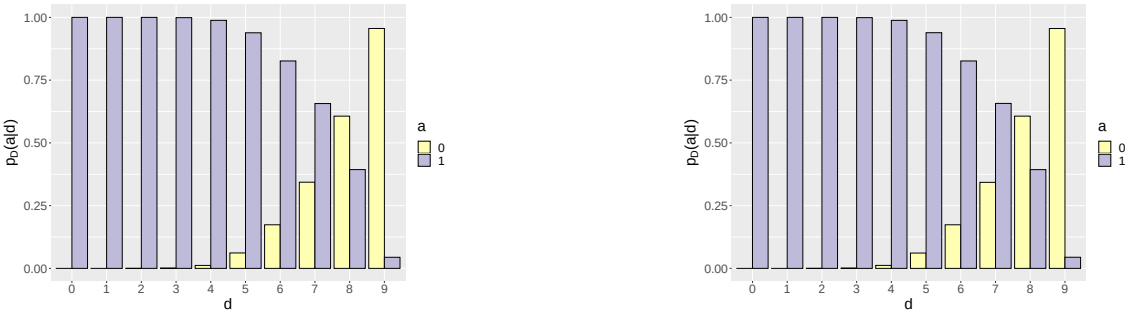
We continue with the example of Section 5.2 studying the case in which complete information is no longer available. $D$'s beliefs over $A$'s judgements are described through $P_A$ and $U_A$. Assume $A$'s random probability of success is modeled as $P_A(\theta = 1 \mid d, a = 1) \sim Beta(\alpha_d, \beta_d)$ with parameters $\alpha_d$ and $\beta_d$ in Table 5.1d (their expected values are equal to $p_D(\theta = 1 \mid d, a)$ from Table 5.1b). In addition, $A$'s risk coefficient $e$ is uncertain, with $e \sim \mathcal{U}(0, 2)$, inducing the random utility $U_A(c_A)$.

In this case, for APS, as the cardinality of $\mathcal{D}$ is small, one can estimate the value of $p_D(a \mid d)$ for each $d$. Figure 5.5 presents the estimates $\hat{p}_D(a \mid d)$, obtained using MC and APS. They coincide up to numerical errors. Next, the ARA solution for the Defender is computed. Figure 5.6a shows the MC estimation of the Defender's expected utility; Figure 5.6b presents the frequency of samples from the marginal $\pi_D(d)$. Its mode coincides with the optimal defense, $d^*_{\text{ARA}} = 9$, in agreement with the MC solution. The ARA decision does not correspond to the Nash equilibrium $d^*_{\text{GT}}$ since its informational assumptions are different. In this case, it appears to be more conservative, as it suggests a safer but more expensive defense.

As in Section 5.2.2, decisions 8 and 9 have similar expected utilities. To check that, indeed, $d^*_{\text{ARA}} = 9$, we repeated the experiment replacing the defender's marginal

**Algorithm 14:** MH APS to approximate ARA solution in the sequential game.

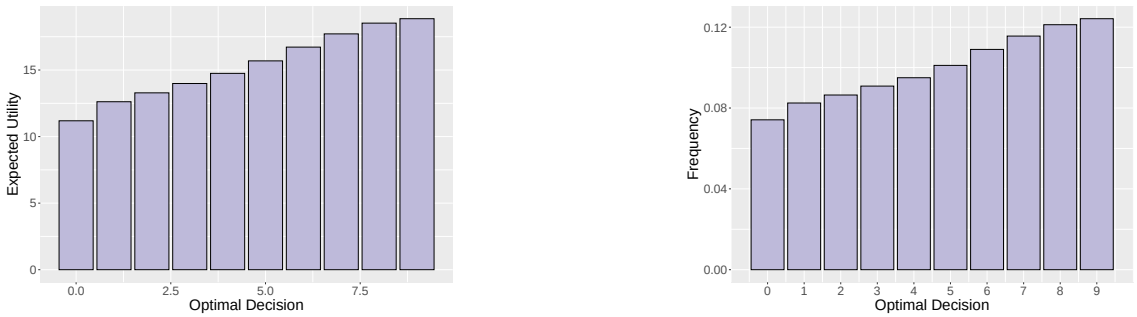
**function** `sample_attack(`$d$, $M$, $K$, $g_A$, $U_A$, $P_A$`)`**:**
  **initialize:** $a^{(0)}$
  Draw $u_A(a, \theta) \sim U_A^{\omega}(a, \theta)$;
  Draw $p_A(\theta \mid a, d) \sim P_A^{\omega}(\theta \mid d, a)$;
  Draw $\theta^{(0)} \sim p_A(\theta \mid a^{(0)}, d)$;
  **for** $i = 1$ **to** $M$ **do** ▷ `Inner APS`
    Propose new attack $\tilde{a} \sim g_A(\tilde{a} \mid a^{(i-1)})$ ;
    Draw $\tilde{\theta} \sim p_A(\theta \mid d, \tilde{a})$;
    Evaluate acceptance probability $\alpha = \min\left\{1, \frac{u_A(\tilde{a}, \tilde{\theta})}{u_A(a^{(i-1)}, \theta^{(i-1)})}\right\}$;
    With probability $\alpha$ set $a^{(i)} = \tilde{a}$, $\theta^{(i)} = \tilde{\theta}$. Otherwise, set $a^{(i)} = a^{(i-1)}$ and $\theta^{(i)} = \theta^{(i-1)}$.;
  **end**
  If convergence, discard first $K$ samples and compute mode $a^*(d)$ of rest of draws $\{a^{(i)}\}$;
  **return** $a^*(d)$;
**input:** $d$, $U_A$, $P_A$, $M$, $K$, $N$, $R$, $g_D$ and $g_A$ symmetric distributions
**initialize:** $d^{(0)} = d$
Draw $a^{(0)} \sim p_A(a \mid d^{(0)})$ using `sample_attack(`$d^{(0)}$, $M$, $K$, $g_A$, $U_A$, $P_A$`)`;
Draw $\theta^{(0)} \sim p_D(\theta \mid d^{(0)}, a^{(0)})$;
**for** $i = 1$ **to** $N$ **do** ▷ `Outer APS`
  Propose new defense $\tilde{d} \sim g_D(\tilde{d} \mid d^{(i-1)})$;
  Draw $\tilde{a} \sim p_A(a \mid \tilde{d})$ using `sample_attack(`$\tilde{d}$, $M$, $K$, $g_A$, $U_A$, $P_A$`)`;
  Draw $\tilde{\theta} \sim p_D(\theta \mid \tilde{d}, \tilde{a})$;
  Evaluate acceptance probability $\alpha = \min\left\{1, \frac{u_D(\tilde{d}, \tilde{\theta})}{u_D(d^{(i-1)}, \theta^{(i-1)})}\right\}$;
  With probability $\alpha$ set $d^{(i)} = \tilde{d}$, $a^{(i)} = \tilde{a}$ and $\theta^{(i)} = \tilde{\theta}$. Otherwise, set $d^{(i)} = d^{(i-1)}$, $a^{(i)} = a^{(i-1)}$ and $\theta^{(i)} = \theta^{(i-1)}$.;
**end**
If convergence, discard first $R$ samples and compute mode $\widehat{d}^*_{\text{ARA}}$ of rest of draws $\{d^{(i)}\}$




(a) MC estimation of $p_D(a \mid d)$

(b) APS estimation of $p_D(a \mid d)$

**Figure 5.5**: Estimation of $p_D(a \mid d)$ through ARA




(a) MC solution (b) APS solution

**Figure 5.6**: ARA solutions for the Defender

augmented distribution $\pi_D$ by its power transformation $\pi_D^H$, more peaked around the mode. Results in Figure 5.7 confirm that $d = 9$ is the optimal ARA decision.

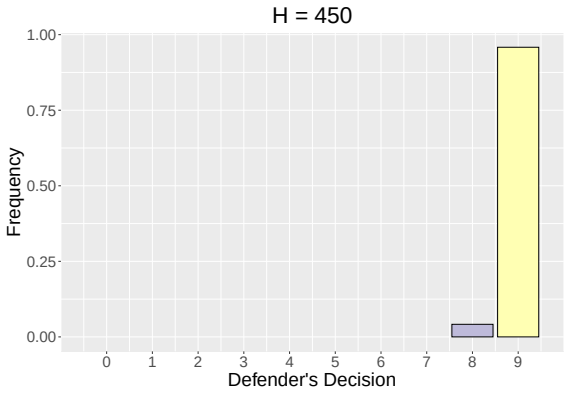
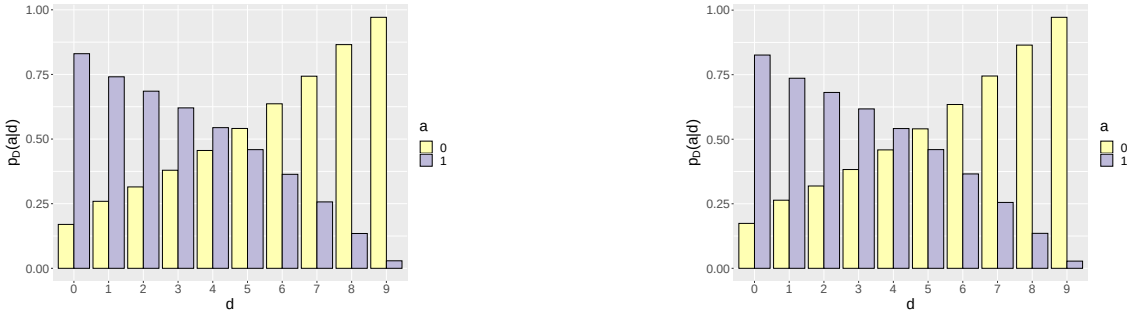



**Figure 5.7**: Defender optimal solutions for ARA using power augmented distributions.

Finally, we repeat the experiment to test the robustness of the ARA solution with respect to our uncertainty level about the attacker. The random probability of success for the Attacker is modeled again as $P_A(\theta = 1 \mid d, a = 1) \sim Beta(\alpha_d, \beta_d)$ but, instead of using parameter values of $\alpha_d$ and $\beta_d$ from Table 5.1d, we divide both by 100. That way, the variances of the resulting beta distributions are more than 50 times larger, thus inducing more uncertainty about the attacker's probabilities. The resulting $p_D(a \mid d)$, obtained with MC and APS, is shown in Figures 5.8a and 5.8b, respectively. As expected, these $p_D(a \mid d)$ distributions are more spread than those in Section 5.3.2, reflecting the fact that the Defender is more uncertain about the Attacker's behaviour. However, the optimal ARA solution ($d^*_{ARA} = 9$) remains stable, as can be seen in Figures 5.9a and 5.9b.

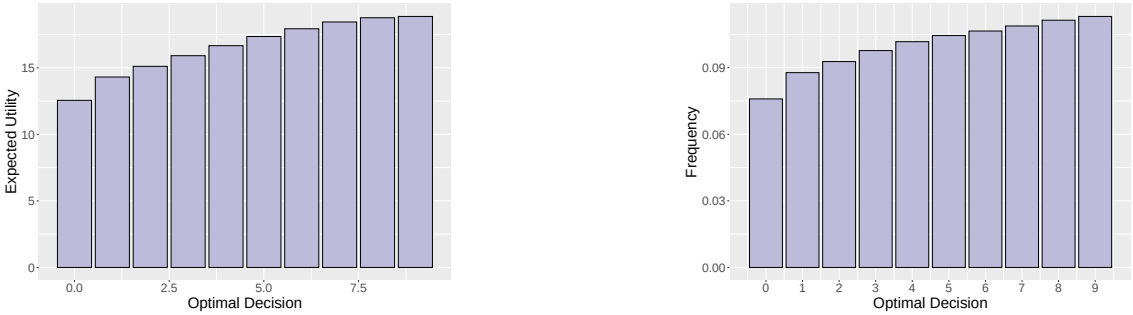
△




(a) MC estimation of $p_D(a \mid d)$

(b) APS estimation of $p_D(a \mid d)$

**Figure 5.8**: Estimation of $p_D(a \mid d)$ through ARA




(a) MC solution

(b) APS solution

**Figure 5.9**: ARA solutions for the Defender

### 5.3.3 Sensitivity analysis of the ARA solution

The ARA approach leads to a decision analysis problem with the peculiarity of including a sampling procedure to forecast $A$'s actions. A sensitivity analysis should be conducted with respect to its inputs ($u_D(d,\theta)$, $p_D(\theta \mid d,a)$, $p_D(a \mid d)$). However, focus should be on $p_D(a \mid d)$, the most contentious element as it comes from adversarial calculations based on the random utility $U_A(a,\theta)$ and probability distribution $P_A(\theta \mid d,a)$. We would proceed similarly to Section 5.2.4 evaluating the impact of the imprecision on $U$ and $P$ over the attained expected utility $\psi(d^{*UP}_{\text{ARA}})$ using classes $\mathscr{U}_A$, $\mathscr{P}_A$ of random utilities and probabilities, and for each pair $(U,P)$ from such classes, $p^{UP}_D(a \mid d)$ would be obtained to compute $d^{*UP}_{\text{ARA}}$, estimating then the maximum regret.

## 5.4 Computational assessment

This section discusses computational complexity results of the proposed algorithms.

### 5.4.1 Computational complexity

Table 5.2 summarizes the computational complexity of MC and APS for solving games with complete and incomplete information compiled from earlier sections. Recall that parameters $P$, $Q$, $N$ and $M$ would typically depend on the desired precision, as outlined in EC.1.1.

| | MC | APS |
|---|---|---|
| Complete | $\lvert\mathcal{D}\rvert \times (\lvert\mathcal{A}\rvert \times Q + P)$ | $N \times (2M+3) + 2M + 2$ |
| Incomplete | $\lvert\mathcal{D}\rvert \times [J \times (\lvert\mathcal{A}\rvert \times Q + 2) + 2P]$ | $N \times (2M+5) + 2M + 4$ |

**Table 5.2**: Required sample sizes by MC and APS algorithms for games with complete and incomplete information

With continuous decision variables, the decision space is discretized to approximate the MC solution, as will be done in Section 5.4.2. This discretization step impacts the precision of the solution and the cardinalities of $\mathcal{D}$ and $\mathcal{A}$ which, in turn, affect complexity. The main lesson from Table 5.2 is that the number of MC samples depends on the cardinality of the Defender's and Attacker's decision spaces, whereas this dependence is not present in APS. Thus, this approach would be expected to be more efficient than MC for problems with large decision spaces as is illustrated next.

### 5.4.2 A computational comparison

A simple game with continuous decision spaces is used to compare the scalability of both methods. Each agent makes a decision $d, a \in [0,1]$ about the proportion of resources respectively invested to defend and attack a server with value $s$. Let

$\theta$ designate the proportion of losses for the defender under a successful attack. It is modeled with a Beta distribution with parameters $\alpha(d, a)$ and $\beta(d, a)$, with $\alpha$ ($\beta$) increasing in $a$ ($d$) and decreasing in $d$ ($a$). $D$'s payoff function is $f(d, \theta) = (1 - \theta) \times s - c \times d$, where $c$ denotes her unit resource cost. She is constant risk averse with utility strategically equivalent to $1 - \exp(-h \times f(d, \theta))$, $h > 0$. $A$'s payoff is $g(a, \theta) = \theta \times s - e \times a$, where $e$ denotes $A$'s unit resource cost. He is constant risk prone with utility strategically equivalent to $\exp(-k \times g(a, \theta))$, $k > 0$.

Figure 5.10 provides the MC estimates of $D$'s expected utility, which is arguably flat. To increase efficiency of the APS algorithm in finding the mode we apply the trick introduced in Section 5.2.3: instead of sampling from the marginal augmented distribution $\pi_D(d \mid a^*(d))$, we sample from its power transformation $\pi_D^H(d \mid a^*(d))$, where $H$ is defined as the augmentation parameter. This distribution is more peaked around the mode (Müller 2005). As we shall see, this provides another advantage of APS over MC, by improving the efficiency of direct MC sampling from flat regions. $A$'s problem includes a similar augmentation. The augmentation parameters for $A$ and $D$ are referred to as inner and outer powers, respectively.

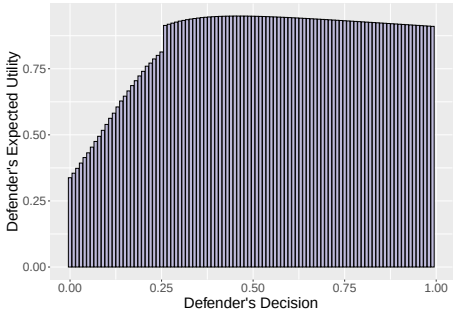



**Figure 5.10**: Defender's expected utility surface

We therefore investigate the trade-off between precision and required sample size finding a limit precision such that MC (APS) is faster for smaller (bigger) precision. The minimum number of required MC and APS samples for optimality and the time to achieve an optimal decision with a given precision are computed for both $A$ and $D$ problems, respectively designated as inner and outer samples. APS also includes choosing the minimum inner and outer powers for which chains are judged to have converged. For a fair comparison, this study conducts several parallel replications of MC and APS with an increasing number of samples until 90% of the solutions coincide with the optimal decision (computed with MC for a large number of samples) and the algorithm is declared to have reached the desired precision for such number of iterations.

Table 5.3 presents MC and APS performance, in terms of computational times, for precisions 0.1 and 0.01, computed using a server node with 16 cores Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz. For instance, with precision 0.1, we discretise $\mathcal{D}$ through $d = \{0, 0.1, 0.2, ..., 1\} \in \mathcal{D}$, and the cardinality of the decision spaces is small, $|\mathcal{D}| = |\mathcal{A}| = 11$. Computational runs show the need for only $1,000$ and 100 MC samples to reach optimality with the required precision to solve $D$ and $A$

| | | Samples | | Power | | |
|---|---|---|---|---|---|---|
| Precision | Algorithm | Outer | Inner | Outer | Inner | Time (s) |
| 0.1 | MC | 1000 | 100 | - | - | 0.007 |
| | APS | 60 | 100 | 900 | 20 | 0.240 |
| 0.01 | MC | 717000 | 100 | - | - | 13.479 |
| | APS | 300 | 100 | 6000 | 100 | 2.461 |

**Table 5.3**: Computational time, minimum number of required MC and APS samples and augmentation parameters at optimality for different precisions

problems, respectively: MC outperforms APS. However, the performance of MC diminishes for higher precision. For instance, for 0.01, $|\mathcal{D}| = |\mathcal{A}| = 101$ and MC becomes more demanding: APS outperforms MC, as we get rid of the dependence on $|\mathcal{D}|$ and $|\mathcal{A}|$. Indeed, for MC, there is a factor of 200 between the time needed to obtain the solutions with precision 0.01 and 0.1. For APS, this factor is just 10, suggesting that it scales much better with precision. For smaller precision, such as 0.001, we could not even get a stable solution using MC even with a large number of samples ($P = 10$M, $Q = 100$k). Finally, observe that as the expected utility is flat around the optimal decision, see Figure 5.10, MC requires a higher number of samples to converge to the optimal solution than the peaked version of APS. To sum up, in problems with large or continuous decision spaces and/or flat expected utilities, APS would be preferred over MC for its scalability.

## 5.5 A cybersecurity application

We illustrate the proposed framework by solving a real cybersecurity problem. Figure 5.11 presents the influence diagram, which simplifies the case study in Rios Insua et al. (2019) by retaining only the adversarial cyber threat. An organisation (Defender) faces a competitor (Attacker) that may attempt a DDoS to undermine the Defender site's availability and compromise her customer services.

The Defender has to determine which security controls to implement: she has to decide about the level of subscription to a monthly cloud-based DDoS protection system, with choices including 0 (not subscribing), $5, 10, 15, \ldots, 190$, and 195 gbps. The Attacker must decide on the intensity of his DDoS attack, viewed as the number of days (from 0 to 30) that he will attempt to launch it. The duration of the DDoS may impact the Defender's market share, due to reputational loss. The Attacker gains all market share lost by $D$, which determines his earnings. However, he runs the risk of being detected with significant costs. Both agents aim at maximizing expected utility. Specific details on the required models at various nodes are the following.

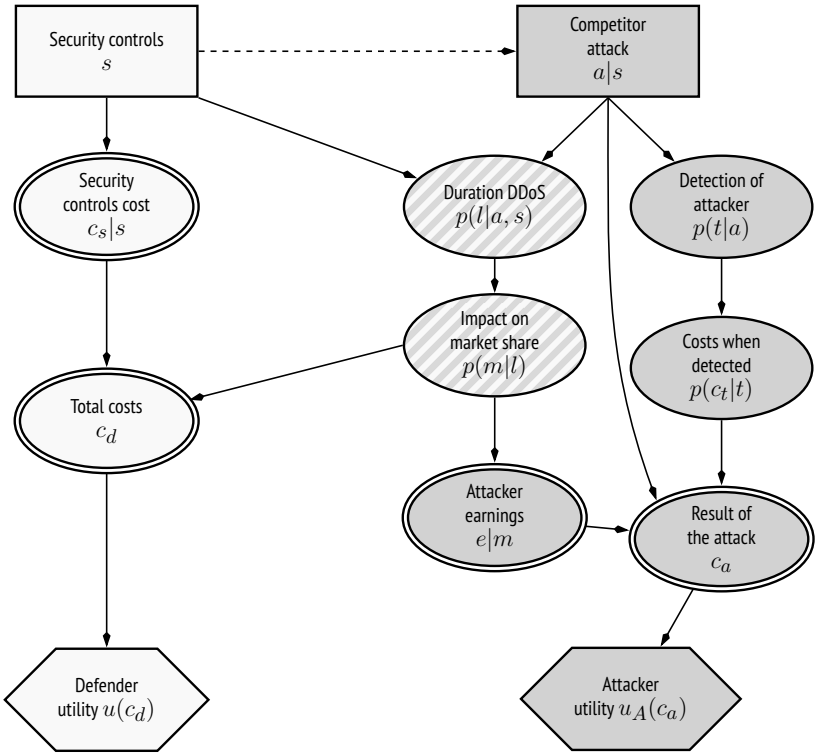



**Figure 5.11**: Bi-agent influence diagram of the cybersecurity application.

**Security controls cost:**

The Defender has to determine the security controls, the protection level of a cloud-based DDoS protection system, with choices including 0 (not subscribing), $5, 10, 15, \ldots, 190$, and 195 gbps. Subscription costs $c_s$ are presented in Figure 5.12.

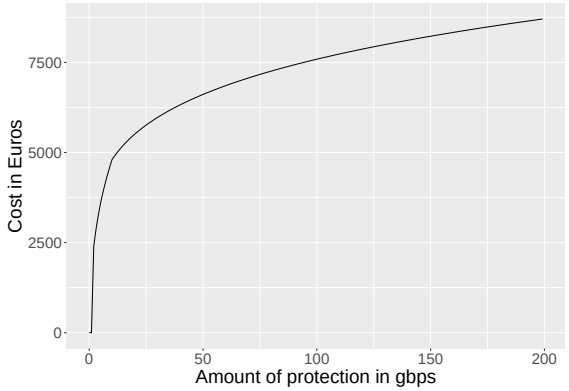



**Figure 5.12**: Costs of DDoS protection given protection hired

**Duration DDoS:**

The duration $l$ in hours of all DDoS attacks depends on both $d$, the cloud-based protection hired by the organization, and the number of attacking attempts, $a$. We model the length $l_j$ of the $j$-th individual attack as a $\Gamma(4,1)$ distribution (its average duration is 4 hours). This duration is conditional on whether the attack actually saturates the target, which depends on the capacity of the attacker's DDoS platform minus the absorption of the cloud-based system. We assume that the Attacker uses a professional platform capable of producing attacks of 5 gbps, modelled through a $\Gamma(5,1)$ distribution. We then subtract the traffic $d$ absorbed by the protection system to determine whether the attack was successful which happens when its traffic overflows the protection system. Thus, the total duration is $l = \sum_{j=1}^{a} l_j$, with $l_j \sim \Gamma(4,1)$ when $\Gamma(5,1) - d > 0$ and $l_j = 0$, otherwise.

To model the Attacker random beliefs about the duration of the attack, we base our estimate on that of the Defender. We model the length of the $j$-th individual DDoS attack as a random gamma distribution $\Gamma_{\text{length}}(\upsilon, \upsilon/\mu)$ with $\upsilon \sim \mathcal{U}(3.6, 4.8)$ and $\upsilon/\mu \sim \mathcal{U}(0.8, 1.2)$ so that we add uncertainty about the average duration (between 3 and 6 hours) and the dispersion. Similarly, we model the attack gbps, refer to as $Q_{\text{gbps}}$, through a random gamma distribution $Q_{\text{gbps}} \sim \Gamma_{\text{gbps}}(\omega, \omega/\eta)$ with $\omega \sim \mathcal{U}(4.8, 5.6)$ and $\omega/\eta \sim U(0.8, 1.2)$. Next, we subtract the protection $d$ from $Q_{\text{gbps}}$ to determine whether the DDoS is successful. We then use $l = \sum_{j}^{a} l_j$, with $l_j \sim \Gamma_{\text{length}}(\upsilon, \upsilon/\mu)$ if $Q_{\text{gbps}} - d > 0$, and $l_j = 0$ otherwise.

**Impact on market share:**

The DDoS duration might cause a reputational loss that would affect the organisation market share. The current market share is 50% valued at 1,500,000 €. We assume that all market share is fully lost at a linear rate until lost in, say, $5-8$ days of unavailability ($120-192$ hours of DDoS duration): in the fastest case the loss rate would be $0.5/120 = 0.00417$ per hour, whereas in the slowest one it would be 0.0026. We model this with a uniform distribution $\mathcal{U}(0.0026, 0.00417)$. Thus, the monetary loss $m$ due to a reduced market share is $m \sim \min[1500000, 3000000 \times l \times \mathcal{U}(0.0026, 0.00417)]$.

For the Attacker, we base our estimate on that of the Defender, adding some uncertainty. The market share value and percentage are not affected by uncertainty, as this information is available to both agents. However, we model the uncertainty in the market loss rate so that the fastest one (5 days in the Defender problem) is between 4 and 6 days in the Attacker problem and the slowest one (8 for Defender) is between 7 and 9. Therefore, the random distribution describing the market loss m is $m \sim \min\left[1500000, 3000000 \times l \times \mathcal{U}(\alpha, \beta)\right]$ with $\alpha \sim \mathcal{U}(0.0021, 0.0031)$ and $\beta \sim \mathcal{U}(0.00367, 0.00467)$.

**Total costs:**

The costs $c_d$ suffered by the Defender include the security control (subscription) costs $c_s$, and the market share lost: $c_d = m + c_s$.

**Defender utility:**

The organisation is constant risk averse over costs. Its utility function is strategically equivalent to $u(c_d) = a - b\exp(k(c_d))$. We rescale the costs to the (0,1) range and calibrate the utility function to $u(c_d) = \frac{1}{\mathrm{e}-1}\left[\exp\left(1 - \frac{c_d}{7000000}\right) - 1\right]$.

**Attacker earnings:**

Being the sole competitor, the Attacker gains $e$ in terms of market share is $e = m$, all the market share lost by the Defender.

**Attacker detection:**

The Attacker runs the risk of being detected with significant costs. Detection probability is estimated via expert judgment at 0.2%, should the Attacker attempt a DDoS attack. Should there be $a$ attacks, the detection has a binomial distribution $\mathcal{B}(a, 0.002)$. To add some uncertainty, we model the detection probability for each attack through a $\beta e(2, 998)$ (Its mean is 0.002.). Thus, we model attacker's detection $t$ through a random binomial distribution that outputs *detected* if $\mathcal{B}(a, \phi) > 0$ with $\phi \sim \beta e(2, 998)$, and *not detected*, otherwise.

**Costs when detected:**

If the attack is detected, there is a further cost for the Attacker deriving from legal issues, discredit, etc. We assume that $c_t \mid t = 1 \sim \mathcal{N}(2430000, 400000)$ for the detection costs, where $t = 1$ indicates that attack is detected. If $t = 0$, there are no further costs.

**Result of the attack:**

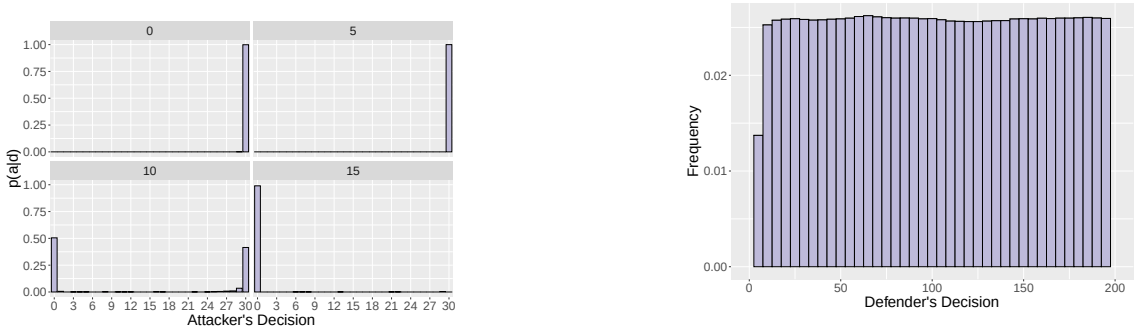
Regarding costs, using a botnet to launch the DDoS attack would cost on average 792 € per day. Overall, the Attacker gains are $c_a = e - c_t - 792a$.

**Attacker utility:**

The Attacker utility function is strategically equivalent to $u_A(c_a) = (c'_a)^{k_a}$, where $c'_a$ are the $c_a$ costs normalised to $[0, 1]$, and $k_a$ is the risk proneness parameter. We add uncertainty on $k_a$ assuming it follows a $\mathcal{U}(8, 10)$ distribution.

This is a problem with incomplete information and large decision spaces. We compute the ARA solution using APS. First, we estimate the probabilities $p_D(a|d)$ for each defense $d$. As in Section 5.4.2, we replace $A$'s marginal augmented distribution $\pi_A$ by its power transformation $\pi_A^H$ to increase APS efficiency in finding the mode. In addition, as in simulated annealing, within each APS iteration we increase $H$ using an appropriate cooling schedule (Müller et al. 2004). Figure 5.13a displays $p_D(a \mid d)$

for four possible defenses $(0, 5, 10, 15)$. When no defensive action is adopted ($d = 0$), $D$ is convinced that $A$ will launch the worst DDoS attack (30 days). Subscribing to a low protection plan ($d = 5$) makes little practical difference. However, when increasing the protection to 10 gbps, the attack forecast (from $D$'s perspective) becomes a mixture of high and low intensity values. The reason for this is that, when $d = 10$, small perturbations in the Defender's assumptions about the Attacker's elements, induce big changes in the optimal attack. Finally, a 15 gbps protection convinces $D$ that she will avoid the attack, attaining a deterrence effect. The optimal solution remains the same for $d > 15$, and therefore results are not displayed.



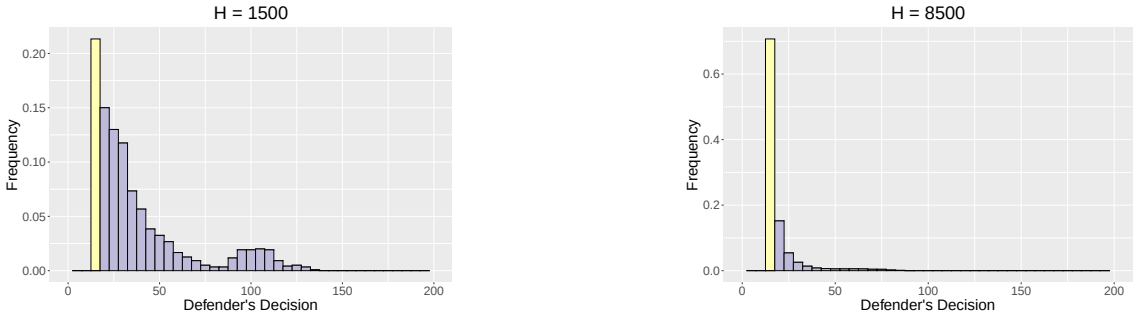

(a) Attack intensity for each decision

(b) APS solution of the Defender problem

**Figure 5.13**: ARA solution computed using APS

Figure 5.13b shows a histogram of the APS samples for $D$'s decision. As expected, the frequency of samples with value 15 is similar to the ones with higher values. As the histogram, and consequently the expected utility surface, is very flat, we cannot resolve the mode. Thus, as we did in $A$'s problem, we sample from increasing $H$ powers of $D$'s marginal augmented distribution. As illustrated in Figure 5.14, increasing $H$ makes the distribution more peaked around the optimal decision $d^*_{ARA} = 15$, the cheapest plan that avoids the attack from $D$'s perspective.

One can argue that finding the exact optimal decision may not be that crucial since the expected utilities for different protections are close. That is a valid point. However, the flat expected optimal utility region might be too big. By sampling from a power transformation of the marginal augmented distribution, APS permits finding the optimal solution even when facing such flat expected utilities at little extra computational cost. Moreover, we can emphasize another advantage of APS: it provides the distribution $\pi_D(d \mid a^*(d))$ as part of the solution, providing sensitivity analysis at no extra cost. The decision maker may want to take such sensitivity of the optimal decision into account, and could consider a defense which could be practically more robust.




(a) (b)

**Figure 5.14**: APS solutions for different augmentation parameter values

## 5.6 Discussion

We have considered the problem of supporting a decision maker against adversaries in an environment with random consequences depending on the actions of all participants. The proposed procedure is summarized as follows. Under complete information, we compute the game-theoretic solution and conduct a sensitivity analysis. If stable, such solution may be used with confidence and no further analysis is required. Otherwise, or if complete information is lacking, we relax the above assumption and use ARA as an alternative decision analytic approach. If the ARA solution is stable, one may use it with confidence and stop the analysis. Otherwise, one must gather more data and refine relevant probability and utility classes, eventually declaring the robustness of the ARA solution. If not sufficient, one could undertake a minimum regret (or other robust) analysis.

We have provided MC and APS methods to solve for these games. With large decision spaces, APS would be more efficient as its complexity does not depend on decision sets' cardinality. It should be also noted that MC errors associated with approximating the expected utility could overwhelm the calculation of the optimal decision. Samples from $p(\theta \mid d, a)$ will typically need to be recomputed for each pair $(d, a)$. In contrast, APS performs the expectation and optimization simultaneously, sampling $d$ from regions with high utility with draws of $\theta$ from the utility-tilted augmented distribution. This reduces MC error as optimization effort in parts of the parameter space with low utility values is limited, resulting in reduced sample sizes for the same precision. In problems with continuous decision sets, an extension of the proposed approaches could use MC to limit the area of the decision space where the optimum is located, and then switching to APS to search within that area in more detail. Exploiting gradient information of the utility functions as in Chapter 4 could also be useful. Apart from computational issues, when the expected utility surface is flat, MC simulation may need many draws or results in poor estimates, being also inefficient for random variables with skewed distributions. APS could handle those cases better as it is based on sampling from the optimizing portions of the decision space. Moreover, APS could sample from a power transformation of the marginal augmented distribution, which is less flat and more peaked around the mode.

Finally, we would like to note that to the best of our knowledge, the APS approach for solving Stackelberg games presented in this chapter is the first implementation of APS in sequential decision problems.

# Chapter 6

# Conclusions

## 6.1 Introduction

Security of ML is essential to protect systems increasingly based on such technology, Comiter (2019). Throughout this PhD thesis we have provided an extensive review of the main topics in ML security. In addition, we have contributed both to the theoretical part, by proposing novel reactive and proactive defences to time series analysis and classification algorithms respectively, and to the algorithmic side, by providing new scalable techniques to solve typical game theoretic problems appearing in ML security. This Chapter recaps the PhD thesis developments in Chapters 2, 3, 4 and 5 and proposes future research lines to extend this work, respectively in Sections 6.2, 6.3, 6.4 and 6.5. Besides, 6.6 provides a general overview of future research lines within the area of secure ML.

## 6.2 Reactive Defences in Time Series Problems

In Chapter 2 a reactive defense for time series security problems has been proposed, from a predictive perspective. In particular, we have worked on advanced detection of threats in the domain of predictive network monitoring, where it is common to deal with a huge number of high frequency time series. Thus, any timely detection system, apart from being accurate, needs to scalable, automatic and versatile. We have provided a framework to identify safety and security issues within a large number of internet connected devices that fulfills those conditions. Our solution uses a Bayesian approach that models continuous time series using a wide range of dynamic linear models, and deals with discrete ones using discrete time Markov chains. This way we fulfill the versatility requirement. Moreover, our procedure just needs to store a few parameters for each time series, we can say that it is scalable, both in terms of memory and runtime. In addition, we have developed a model identification procedure, to guarantee that our approach is able to work automatically without human supervision.

Improvements in the algorithm can still be performed, specially if we take

into account specifics of the cases monitored. To this extent, using the eventual hierarchy between monitored devices, model identification could be optimized, for example allowing for a faster regular outburst detection. Moreover, should there exist correlated time series in the database, such correlations could be used similarly to our advantage. For this we would need coupling-decoupling strategies of the type described in e.g. Berry and West (2018). It is also possible to carry out an automatic performance evaluation of the algorithm. This could be done by computing the autocorrelation function of the residuals within a certain time-window. If, for instance, a strong correlation in the first few lags is encountered, an autoregressive term could be added to the model. In other cases, when model performance deteriorates too much, an alarm could be issued, finally demanding the intervention of a human analyst. It may be also interesting to adjust in real time the width of the predictive probability intervals taking into account the particular potential economical losses of false positives and negatives of the system. From this point of view, we may also find it interesting to re-build the structure of the algorithm to try to produce long-term forecasts including the probability intervals as well, moving beyond the point-wise predictions here proposed for practical implementation.

Finally, we have just focused on reactive defenses for time series security problems. An interesting future research line would be to explore proactive defenses that model explicitly the presence of an adversary. These could be interesting for cases in which an attacker might be interested in intruding a defender network without alerting a monitoring system or when adversaries deliberately manipulate the value of time series to drive the predictions towards a particular objective. Thus, an adversarial framework extending that of Chapter 2 would be of great interest.

## 6.3 Proactive Defences in Classification Problems

Chapter 3 presented ACRA, a general, Bayesian probabilistic framework for adversarial classification that does not assume standard common knowledge assumptions by modeling explicitly, not only the presence of an adversary, but also our uncertainty about his elements. We have provided extensive empirical evidence of the performance of our framework, through case studies in spam and malware detection, showcasing, among other things, the robustness of ACRA to imprecisions in the assumptions made about the adversary. In particular, ACRA has been shown to be more robust than its common knowledge, purely game theoretic counterparts.

Our framework may be extended in several ways. First of all, we could extend the ACRA approach to situations in which there is repeated play of the adversarial classification game, thus introducing the possibility of learning the adversarial utilities and probabilities in a Bayesian way. Also, we have only considered exploratory attacks, but we could extend the approach to take into account attacks over the training data, called poisoning attacks, Biggio et al. (2012). In addition, we have just considered the case in which the attacker performs intentional attacks. In some problems, there could be, in addition, random attacks. The proposed framework could be adapted to

take those into account as well as to the case in which there are several attackers. Also, the Approximate Bayesian Computation framework in Section 3.5.1, is based on a vanilla version of ABC. An interesting line of future research is the adaptation of more sophisticated versions of ABC as MCMC based ABC, Marjoram et al. (2003).

Finally, our work could be extended to the case of attacks to innocent instances (not just integrity violation ones). In this case, when computing $p_C(x'|y_i)$ in (3.2.1), with $i > l$ we must proceed similarly as we did when computing $p_C(x'|y_i)$, $i \leq l$: we consider all possible originating instances $x$ leading to $x'$, and sum them weighting each of them with their $p_C(a_{x\to x'}|x, y_i)$, the probability of the attacker choosing the attack linking each of them with $x'$, given that they are innocent. This would just involve replacing $\sum_{i=l+1}^{k} p_C(x'|y_i)p_C(y_i)$ by $\sum_{i=l+1}^{k}\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x, y_i)p_C(x|y_i)p_C(y_i)$ in (3.2.1). Finally, as computing $p_C(a_{x\to x'}|x, y_i)$ demands strategic thinking, we need to consider the attacker's problem as we did in Section 3.2.2, but this time allowing the attacker to modify also innocent instances.

Regarding computational aspects, note that in ACRA we go through a simulation stage to forecast attacks and an optimization stage to determine optimal classification. The whole process might be performed in a single stage, possibly based on augmented probability simulation, extending the results in Chapter 5.

## 6.4 Algorithmic approaches: Gradient Methods for Stackelgerg Games in AML

Stackelberg Games have been gaining importance in recent years due to the use of such games to model confrontations within Adversarial Machine Learning problems. Within this context, a new paradigm must be faced: in classical game theory, intervening agents were humans whose decisions are generally discrete and low dimensional. In AML, decisions are made by algorithms and are usually continuous and high dimensional, e.g. choosing the weights of a neural network. As closed form solutions for Stackelberg games generally do not exist, it is mandatory to have efficient algorithms to search for numerical solutions. In Chapter 4, we have focused on gradient solution methods, providing two different approaches to compute the gradient of the defender's utility function in a Stackelberg game: the forward and backward solutions. The backward solution scales well in time with the defender's decision space dimension, at a cost of more memory requirements. On the other hand, the forward solution scales poorly in time with this dimension, but has better space scalability. Backward and forward methods could be use to solve both, Stackelberg Games and Bayesian Stackelberg Games.

We could extend the framework in several ways. First, as we discussed, the backward solution has poor space scalability. This is generally not an issue in most applications. Nevertheless, if space complexity is critical it is possible to reduce it at a cost of introducing a numerical error, as proposed in Maclaurin et al. (2015) in hyperparameter optimization problems. Instead of storing the whole trace $\beta_t(\alpha)$ in the first for loop of Algorithm 8 to use it in the second loop, we could sequentially

undo its gradient update at each step of the second for loop. Obviously, this would introduce some numerical errors. Another possible line of work would be to extend the methodology proposed in Section 4.4 to solve adversarial risk analyisis problems in AML, Chapter 3. The ARA framework fully removes the common knowledge assumptions present in game theoretic approaches.

We have focused on exact gradient methods. However, it would be interesting to extend the proposed algorithms to work with stochastic gradient methods. In addition, in Mokhtari et al. (2019) the authors propose several variants of Gradient Ascent to solve saddle point problems. It could be worth investigating how to extend such techniques to general Stackelberg Games.

## 6.5 Algorithmic approaches: APS Method for Non-cooperative Games

Chapter 5 considered the problem of supporting a decision maker against adversaries in an environment with random consequences that depend on the actions of all participating agents. We proposed a general procedure in which the game-theoretic solution is computed and subjected to a sensitivity analysis. If stable, such solution may be used with confidence and no further analysis is required. Otherwise, we relax the common knowledge assumption and use ARA as an alternative decision analytic approach. We have provided MC and APS methods to solve the proposed problems. With large decision spaces, APS would be more efficient as its complexity does not depend on their cardinality.

The proposed algorithmic approaches could be extended in several ways. First, in problems with continuous decision sets, an extension could consist of using MC to limit the area of the decision space where the optimum is located, and then switching to APS to search within a local neighborhood of such solution in more detail. Also, exploiting information of the gradient of utility functions could be useful as illustrated in Chapter 4. Finally, in addition to the sequential two stage games, these ideas could be extended to other game types such as simultaneous defend-attack games (Rios and Rios Insua 2012) and general bi-agent influence diagrams (González-Ortega et al. 2019).

## 6.6 Future research lines

In this section, we provide a general overview of additional future research lines that we find specially interesting within the area of secure ML.

### 6.6.1 Robustifying ML algorithms through Bayesian ideas

Bayesian methods have been shown to provide enhanced robustness in AML. Chapter 5 shows how game theory solutions based on point estimates of preferences and

beliefs, potentially lead to unstable solutions, while ARA solutions tend to be more robust acknowledging uncertainties in such judgments. Here are some additional ideas.

**Bayesian methods and AML.** It has been empirically shown that model ensembling is an effective mechanism for designing powerful attack models (Tramèr et al. 2018). The Bayesian treatment of ML models offers a way of combining the predictions of different models via the predictive distribution. Gal and Smith (2018) show how certain idealized Bayesian NNs properly trained lack adversarial examples. Thus, a promising research line consists of developing efficient algorithms for approximate Bayesian inference with robustness guarantees. Indeed, there are several ways in which the Bayesian approach may increase security of ML systems. Regarding opponent modelling, in sequential decision making, an agent has uncertainty over her opponent type initially; as information is gathered, she might be less uncertain about her model via Bayesian updating (Gallego et al. 2019a). Uncertainty over attacks in supervised models can also be considered to obtain a more robust version of AT, as in Ye and Zhu (2018) who sample attacks using a SG-MCMC method. Combining their approach with ARA opponent modeling may further increase robustness. Lastly, there are alternative approaches to achieve robustness in presence of outliers and perturbed observations, as through the robust divergences for variational inference (Futami et al. 2018).

**Robust Bayesian methods.** The robust Bayesian literature burgeoned in the period 1980-2000 (Berger 1982; Rios Insua and Ruggeri 2000). In particular, there has been relevant work in Bayesian likelihood robustness (Shyamalkumar 2000) referring to likelihood imprecision, reminiscent of the impact of attacks over the data received. Note that Bayesian likelihood robustness focuses around random or imprecise perturbations and contaminations in contrast to the purposeful perturbations in AML. Not taking into account the presence of an adversary affecting data generation is an example of model misspecification; robustness of Bayesian inference to such issue has been revisited recently in Miller and Dunson (2019). Their ideas could be used to robustify ML algorithms in adversarial contexts.

### 6.6.2 Modeling and computational enhancements

We discuss modeling and computational enhancements aimed at improving operational aspects of the proposed framework.

**Characterizing attacks.** A core element in the AML pipeline is the choice of the attacker perturbation domain. This is highly dependent on the nature of the data attacked. In computer vision, a common choice is an $\ell_p$ ball of radius $\delta$ centered at the original input. For instance, an $\ell_\infty$ norm implies that an attacker may modify any pixel by at most $\delta$. These perturbations, imperceptible to the human eye, may not be representative of threats actually deployed. As an example, Brown et al.

(2017) designed a circular sticker that may be printed and deployed to fool state of the art classifiers. Thus, it is important to develop threat models that go beyond $\ell_p$ norm assumptions. Moreover, we discussed only problems with two agents. It is relevant to deal with multiple agents in several variants (one defender vs. several attackers, several defenders vs. several attackers) including cases in which agents on one of the sides somehow cooperate.

**New algorithmic approaches.** Exploring gradient-based techniques for bi-level optimization problems arising in AML is a fruitful line of research (Naveiro and Insua 2019). However the focus has been on white box attacks. It would be interesting to extend those results to the framework here proposed. On the other hand, Bayesian methods are also hard to scale to high dimensional problems or large datasets. Recent advances in accelerating SG-MCMC samplers, e.g., Gallego and Insua (2019) and Gallego and Insua (2018), are crucial to leverage the benefits of a Bayesian treatment.

### 6.6.3 Applications

As presented in Comiter (2019), applications abound. We mention three of interest to us.

**Fake news.** The rise in computer power used in deep learning is leading to quasi-realistic automatic text generation (Radford et al. 2019), which can be used for malicious purposes such as fake reviews generation (Juuti et al. 2018). At present, state of the art defences consist mostly of statistical analyses of token distributions (Gehrmann et al. 2019). These are attacker-agnostic. An ARA treatment may be beneficial to inform the model of which are the most likely attack patterns.

**Autonomous driving systems.** ADS directly benefit from developments in computer vision and RL. However, accidents still occur because of lack of guarantees in verifying the correctness of deep NNs. Alternative solutions include the use of Bayesian NNs, as there is evidence that uncertainty measures can predict crashes up to five seconds in advance (Michelmore et al. 2018). McAllister et al. (2017) propose propagating the uncertainty over the network to achieve a safer driving style. As mentioned, adversaries may interact with the visual system through adversarial examples. Thus, developing stronger defences are of major importance.

**Malware detection.** Its methods are traditionally classified in three categories (Nath and Mehtre 2014): static, dynamic and hybrid approaches. Shabtai et al. (2009) describe how ML algorithms may accomplish accurate classification to detect new malware. Note though that, besides the illustrative case study in Chapter 3, no ARA based AML methods have been used yet in this domain.

# Appendices

# Appendix A

# Proofs of Monte Carlo based approaches for solving sequential games

We briefly present convergence of MC approximations in Algorithms 10 and 13 of Chapter 5.

## A.1 Algorithm 10. Subgame perfect equilibria

Suppose that $\mathcal{A}$ is a compact set in an Euclidean space; $\Theta$ is an Euclidean space; $u_A(a, \theta)$ is continuous in $a$ for each $\theta$ and measurable in $\theta$ for each $a$; there exists a function $h(\theta)$, integrable with respect to $p_A(\theta|d, a)$, such that $|u_A(a, \theta)| \leq h(\theta)$. Then, for almost all sequences $\{\theta_i\}$ forming a sample from $p_A(\theta \mid d, a)$, $\widehat{\psi}_A(d, a) = \frac{1}{Q} \sum_{i=1}^{Q} u_A(a, \theta_i) \to \psi_A(d, a)$ when $Q \to \infty$ uniformly in $a$, as in Thm.2 in Jennrich (1969). Then, assuming that the best response $a^*(d)$ is unique, we can prove that $\operatorname{argmax}_a \widehat{\psi}_A(d, a)$ converges almost surely to $a^*(d)$ as $Q \to \infty$, with arguments similar to Thm.1 in Shao (1989).

Next, under similar conditions for $u_D(d, \theta)$, $p_D(\theta|d, a)$ and $\mathcal{D}$, we have the almost sure uniform convergence of the Monte Carlo averages $\widehat{\psi}_D(d) = \frac{1}{P} \sum_{i=1}^{P} u_D(d, \theta_i)$ to the expected utilities $\psi_D(d)$ as $P \to \infty$ and, therefore, the almost sure convergence of $\operatorname{argmax}_d \widehat{\psi}_D(d)$ to $d^*_{\mathrm{GT}}$.

Several relevant comments are:

- When $\mathcal{A}$ and $\mathcal{D}$ are finite, the same argument applies.
- When $a^*(d)$ is not unique, corresponding to a case of alternative optima, convergence to the optima may not hold (although it would hold for the maximal expected utilities), but we may disentangle several convergent subsequences to alternative best responses in case the set of alternative optima is finite. Then, the argument would hold for the inner loop. And similarly, for the outer loop. Note that with an infinite set of alternative optima we could have extreme cases, although we would estimate correctly the optimal expected utilities.

- The uniform convergence condition facilitates the use of regression metamodels (Chen et al. 2013) allowing to replace expected utilities in just a few values of $a$ or $d$ and then optimizing the metamodel as an alternative computational approach.
- Recall that $P$ and $Q$ are essentially dictated by the required precision. Based on the Central Limit Theorem (Chung 2001), MC sums approximate integrals with probabilistic bounds of the order $\sqrt{\frac{var}{N}}$ where $N$ is the MC sum size. To obtain a variance estimate, we run a few iterations and estimate the variance, then choosing the required size based on such bounds. In our case, in which there are multiple integrals to approximate associated to the various $a$ and/or $d$, we would run a few MC iterations at several $a$ or $d$ values and use the maximum variance to estimate the required MC simulation sizes $Q$ and $P$.

## A.2 Algorithm 13. ARA solutions

Convergence follows a similar path to A.1. Suppose that, almost surely, $U_A^\omega(a,\theta)$ is continuous in $a$ for each $\theta$ and measurable in $\theta$; there is $h(\theta)$ integrable such that for each $a$ and $|U_A^\omega(a,\theta)| \leq h(\theta)$; and $\mathcal{A}$ is compact. Then, with the same argument $\operatorname{argmax}_a \widehat{\Psi}_A^\omega(a,d)$ converges almost surely to $A^*(d)^\omega$. Next, by construction $p_D(a \mid d) = P(A^*(d) = a)$. Using the SLLN $\hat{p}_D(a \mid d)$ converges almost surely to $p_D(a \mid d)$. The rest of the approximation follows by another application of a uniform version of the SLLN.

# Appendix B

# Gibbs sampling based APS methods

In Chapter 5, the core APS based algorithms 11 and 14 are of the MH type. This section covers Gibbs sampler based APS versions. Gibbs sampling (GS) (Roberts and Smith 1994) can be utilized in cases where samples from full conditional distributions are available. This typically requires a more substantial preliminary analysis than MH, but tends to converge faster (when full conditionals are available). They iteratively sample from the conditional distributions resulting in samples from the joint distribution in the limit under mild conditions (Casella and George 1992).

## B.1 Subgame perfect equilibria

Based on the analysis in Section 5.2.2, Algorithm 15 summarizes a GS procedure. For each $d$, the Attacker's APS samples iteratively from $\pi_A(a|\theta, d)$ and $\pi_A(\theta|d, a)$, whereas the Defender's APS samples iteratively from $\pi_D(d \mid \theta, a^*(d))$ and $\pi_D(\theta \mid d, a^*(d))$. Convergence of Algorithm 15 follows from arguments similar to Proposition 1. It requires $2 \times (|\mathcal{D}| \times M + N)$ samples plus the cost of convergence checks and $|\mathcal{D}| + 1$ mode approximations. The computational complexity of Algorithm 15 does not depend on the dimension of $\mathcal{A}$, which could be crucial when $|\mathcal{A}|$ is very large or $\mathcal{A}$ is continuous.

## B.2 ARA for incomplete information games

Similarly, we consider a Gibbs sampler based APS approach for the ARA model in incomplete information games as an alternative to Algorithm 14. The convergence of the Algorithm 16 follows similarly as from Proposition 2.

Overall, MCMC sampling can result in fast convergence such as geometric convergence in many cases and polynomial time in some cases in contrast to standard central limit theorem type convergence of MC estimation, as also detailed in Jacquier et al. (2007).

**Algorithm 15:** Gibbs based APS to solve a game with complete information.

**input:** $N$, $M$, $K$
**initialize:** $a^{(0)}$, $\theta^{(0)}$
**for** $d \in \mathcal{D}$ **do**
  **for** $j = 1$ **to** $M$ **do**
    Draw $\theta_A^{(j)}$ from $\pi_A(\theta \mid d, a^{(j-1)})$;
    Draw $a^{(j)}$ from $\pi_A(a \mid \theta_A^{(j)}, d)$;
  **end**
  Compute mode of $M$ draws $\{a^{(j)}\}$ and record it as $a^*(d)$;
**end**
**initialize:** $d^{(0)}$, $\theta^{(0)}$
**for** $i = 1$ **to** $N$ **do**
  Draw $\theta_D^{(i)}$ from $\pi_D(\theta \mid d^{(i-1)}, a^*(d^{(i-1)}))$;
  Draw $d^{(i)}$ from $\pi_D(d \mid \theta_D^{(i)}, a^*(d))$
**end**
Discard first $K$ samples, compute mode of rest of draws $\{d^{(i)}\}$ and propose it as $\widehat{d}^*_{\mathrm{GT}}$;

**Algorithm 16:** Gibbs based APS approach to solve the ARA problem

**input:** $N, M, J$
**for** $d \in \mathcal{D}$ **do**
  **for** $j = 1$ **to** $J$ **do**
    Sample $U_A^j$, $P_A^j$ and define $\Pi_A^j$;
    Initialize $\theta^0$;
    **for** $i = 1$ **to** $M$ **do**
      Sample $a^{(i)}$ from $\Pi_A^j(a \mid \theta^{(i-1)}, d)$;
      Sample $\theta^{(i)}$ from $\Pi_A^j(\theta \mid a^{(i)}, d)$;
    **end**
    Estimate $a_j^*$ as mode of $\{a^{(i)}\}$;
  **end**
  Estimate $p_D(a \mid d)$ from $\{a_j^*\}$;
**end**
Initialize $(d^{(0)}, \theta^{(0)})$;
**for** $i = 1$ **to** $N$ **do**
  Draw $d^{(i)}$ from $\pi_D(d \mid a^{(i-1)}, \theta_D^{(i-1)})$;
  Draw $\theta_D^{(i)}$ from $\pi_D(\theta \mid a^{(i-1)}, d^{(i)})$;
  Draw $a^{(i)}$ from $\pi_D(a \mid d^{(i)}, \theta_D^{(i)})$;
**end**
Estimate $d^*$ as mode of $\{d^{(i)}\}$, record it as $d^*_{\mathrm{ARA}}$.

# Bibliography


Albrecht, S. V. and Stone, P. (2018). "Autonomous agents modelling other agents: A comprehensive survey and open problems". In: *Artif. Intell.* 258, pp. 66–95.

Alfeld, S., Zhu, X., and Barford, P. (2016). "Data poisoning attacks against autoregressive models". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1452–1458.

Aliprantis, C. D. and Chakrabarti, S. K. (2002). *Games and Decision Making*. Oxford University Press.

Antos, D. and Pfeffer, A. (2010). "Representing Bayesian Games without a Common Prior". In: *Proc. AAMAS 2010*. Ed. by L. van der Hoek Kaninka and Sen. IFAMAS.

Athalye, A., Carlini, N., and Wagner, D. (2018). "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *International Conference on Machine Learning*, pp. 274–283.

Banks, D. L., Aliaga, J. M. R., and Insua, D. R. (2015). *Adversarial risk analysis*. Chapman and Hall/CRC.

Bard, J. F. (1991). "Some properties of the bilevel programming problem". In: *Journal of optimization theory and applications* 68.2, pp. 371–378.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. (2006). "Can machine learning be secure?" In: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*. ACM, pp. 16–25.

Berger, J. (1982). "The robust Bayesian point of view". In: *Robustness*. Springer Verlag.

Berry, L. and West, M. (2018). "Bayesian forecasting of many count-valued time series". In: *arXiv preprint arXiv:1805.05232*.

Bianco, A. M., Garcia Ben, M, Martinez, E., and Yohai, V. J. (2001). "Outlier detection in regression models with arima errors using robust estimates". In: *Journal of Forecasting* 20.8, pp. 565–579.

Bielza, C., Müller, P., and Rios Insua, D. (1999). "Decision analysis by augmented probability simulation". In: *Management Science* 45, pp. 995–1007.

Biggio, B. and Roli, F. (2018). "Wild patterns: Ten years after the rise of adversarial machine learning". In: *Pattern Recognition* 84, pp. 317–331.

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

Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., and Roli, F. (2013). "Is data clustering in adversarial settings secure?" In: *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*. ACM, pp. 87–98.

Biggio, B., Fumera, G., and Roli, F. (2014). "Security evaluation of pattern classifiers under attack". In: *IEEE Transactions on Knowledge and Data Engineering* 26, pp. 984–996.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387310738.

Bishop, C. M. and Lasserre, J. (2007). "Generative or discriminative? Getting the best of both worlds". In: *Bayesian Statistics* 8.3, pp. 3–24.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316*.

Bottou, L. (1998). "Online learning and stochastic approximations". In: *On-line learning in neural networks* 17.9, p. 142.

Breiman, L. (2001). "Statistical modeling, the two cultures". In: *Statistical Science* 16, pp. 199–231.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. CRC Press.

Brooks, S. P. and Roberts, G. O. (1998). "Assessing convergence of Markov chain Monte Carlo algorithms". In: *Statistics and Computing* 8.4, pp. 319–335.

Brown, G. W. (1951). "Iterative Solution of Games by Fictitious Play". In: *Activity Analysis of Production and Allocation*, pp. 374–376.

Brown, G., Carlyle, M., Salmerón, J., and Wood, K. (2006). "Defending critical infrastructure". In: *Interfaces* 36.6, pp. 530–544.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. (2017). "Adversarial patch". In: *arXiv preprint arXiv:1712.09665*.

Brückner, M. and Scheffer, T. (2011). "Stackelberg games for adversarial prediction problems". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 547–555.

Brückner, M., Kanzow, C., and Scheffer, T. (2012). "Static prediction games for adversarial learning problems". In: *Journal of Machine Learning Research* 13.Sep, pp. 2617–2654.

Brutlag, J. D. (2000). "Aberrant Behavior Detection in Time Series for Network Monitoring." In: *Large Installation System Administration.* Pp. 139–146.

Buşoniu, L., Babuška, R., and De Schutter, B. (2010). "Multi-agent reinforcement learning: An overview". In: *Innovations in multi-agent systems and applications-1.* Springer, pp. 183–221.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., and Madry, A. (2019). "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705.*

Casella, G. and George, E. I. (1992). "Explaining the Gibbs sampler". In: *The American Statistician* 46.3, pp. 167–174.

Casella, G., Robert, C. P., and Wells, M. T. (2004). "Generalized Accept-Reject sampling schemes". In: *A Festschrift for Herman Rubin.* Ed. by A. DasGupta. Vol. Volume 45. Lecture Notes–Monograph Series. Beachwood, Ohio, USA: Institute of Mathematical Statistics, pp. 342–347. URL: `https://doi.org/10.1214/lnms/1196285403`.

Chen, C.-H. (2015). *Handbook of Pattern Recognition and Computer Vision.* World Scientific.

Chen, X., Ankenman, B. E., and Nelson, B. L. (2013). "Enhancing stochastic kriging metamodels with gradient estimators". In: *Operations Research* 61.2, pp. 512–528.

Chib, S. and Greenberg, E. (1995). "Understanding the Metropolis-Hastings algorithm". In: *The American Statistician* 49.4, pp. 327–335.

Chronicle (2018). *Virus Total.* `https://www.virustotal.com/`.

Chung, K. L. (2001). *A Course in Probability Theory.* Ac Press.

Clemen, R. T. and Reilly, T. (2013). *Making hard decisions with DecisionTools.* Cengage Learning.

Comiter, M. (2019). *Attacking Artificial Intelligence.* Belfer Center Paper.

Couce-Vieira, A., Rios Insua, D., and Kosgodagan, A. (2019). "Assessing and forecasting cybersecurity impacts". In: *Technical Report.*

Csilléry, K., Blum, M. G., Gaggiotti, O. E., and François, O. (2010). "Approximate Bayesian Computation (ABC) in practice". In: *Trends in Ecology & Evolution* 25.7, pp. 410–418.

Dalvi, N., Domingos, P., Mausam, Sumit, S., and Verma, D (2004). "Adversarial classification". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04, 99–108. ISBN: 1-58113-888-1.

Dasgupta, P. and Collins, J. B. (2019). "A Survey of Game Theoretic Approaches for Adversarial Machine Learning in Cybersecurity Tasks." In: *AI Magazine* 40.2.

Dimelis, S. P. and Papaioannou, S. K. (2011). "ICT growth effects at the industry level: A comparison between the US and the EU". In: *Information Economics and Policy* 23.1, pp. 37–50.

ENISA (2019). *Threat Landscape Report 2018 15 Top Cyberthreats and Trends*. Tech. rep. European Union Agency For Network and Information Security. URL: `http://www3.weforum.org/docs/WEF_Global_Risks_Report_2019.pdf`.

Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference*. Vol. 5. Cambridge University Press.

Ekin, T., Polson, N. G., and Soyer, R. (2014). "Augmented Markov chain Monte Carlo simulation for two-stage stochastic programs with recourse". In: *Decision Analysis* 11.4, pp. 250–264.

Ekin, T., Naveiro, R., Torres-Barrán, A., and Ríos-Insua, D. (2019). "Augmented Probability Simulation Methods for Non-cooperative Games". In: *arXiv preprint arXiv:1910.04574*.

Fan, J., Ma, C., and Zhong, Y. (2020). "A selective overview of deep learning". In: *Statistical Science (to appear)*.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. (2017). "Forward and reverse gradient-based hyperparameter optimization". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1165–1173.

French, S. and Rios Insua, D. (2000). *Statistical Decision Theory*. Wiley.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*. Vol. 1. Springer Series in Statistics New York.

Futami, F., Sato, I., and Sugiyama, M. (2018). "Variational Inference based on Robust Divergences". In: *International Conference on Artificial Intelligence and Statistics*, pp. 813–822.

Gal, Y. and Smith, L. (2018). "Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with Bayesian neural networks". In: *arXiv preprint arXiv:1806.00667*.

Gallego, V. and Insua, D. R. (2018). "Stochastic Gradient MCMC with Repulsive Forces". In: *Bayesian Deep Learning Workshop, Neural Information and Processing Systems (NIPS)*.

Gallego, V. and Insua, D. R. (2019). "Variationally Inferred Sampling Through a Refined Bound". In: *Advances in Approximate Bayesian Inference (AABI)*.

Gallego, V., Naveiro, R., Insua, D. R., and Oteiza, D. G.-U. (2019a). "Opponent Aware Reinforcement Learning". In: *arXiv preprint arXiv:1908.08773*.

Gallego, V., Naveiro, R., and Insua, D. R. (2019b). "Reinforcement Learning under Threats". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 9939–9940.

Gamerman, D. and Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC.

Gehrmann, S., Strobelt, H., and Rush, A. (2019). "GLTR: Statistical Detection and Visualization of Generated Text". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, pp. 111–116.

Ghosh, A. and Parai, B. (2008). "Protein secondary structure prediction using distance based classifiers". In: *International Journal of Approximate Reasoning* 47.1, pp. 37–44.

Gibbons, R. (1992). *A Primer in Game Theory*. Harvester Wheatsheaf.

González-Ortega, J., Insua, D. R., and Cano, J. (2019). "Adversarial risk analysis for bi-agent influence diagrams: An algorithmic approach". In: *European Journal of Operational Research* 273.3, pp. 1085 –1096. ISSN: 0377-2217. URL: http://www.sciencedirect.com/science/article/pii/S0377221718307756.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015a). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*. URL: http://arxiv.org/abs/1412.6572.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015b). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*. URL: http://arxiv.org/abs/1412.6572.

Goodman, J. and Heckerman, D. (2004). "Fighting spam with statistics". In: *Significance* 1.2, pp. 69–72.

Gordon, G. and Tibshirani, R. (2012). "Karush-kuhn-tucker conditions". In: *Optimization* 10.725/36, p. 725.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T. A., and Kohli, P. (2018). "On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models". In: *CoRR* abs/1810.12715. arXiv: 1810.12715. URL: http://arxiv.org/abs/1810.12715.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2019). "Your classifier is secretly an energy based model and you should treat it like one". In: *International Conference on Learning Representations*.

Grenander, U. (1965). "Direct estimates of the mode". In: *The Annals of Mathematical Statistics* 36.1, pp. 131–138.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Vol. 105.

Großhans, M., Sawade, C., Brückner, M., and Scheffer, T. (2013). "Bayesian games for adversarial regression problems". In: *International Conference on Machine Learning*, pp. 55–63.

Hammersley, J. (2013). *Monte carlo methods*. Springer Science & Business Media.

Hand, D. J. and Henley, W. E. (1997). "Statistical classification methods in consumer credit scoring: a review". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160.3, pp. 523–541.

Hargreaves-Heap, S. and Varoufakis, Y. (2004). *Game Theory: A Critical Introduction*. Routledge.

Harsanyi, J. C. (1967). "Games with incomplete information played by "Bayesian" players, I–III Part I. The basic model". In: *Management science* 14.3, pp. 159–182.

Hinze, M., Pinnau, R., Ulbrich, M., and Ulbrich, S. (2008). *Optimization with PDE constraints*. Vol. 23. Springer Science & Business Media.

Hu, J. and Wellman, M. P. (2003). "Nash Q-learning for general-sum stochastic games". In: *Journal of Machine Learning Research* 4.Nov, pp. 1039–1069.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. (2011). "Adversarial machine learning". In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. AISec '11. Chicago, Illinois, USA, pp. 43–58. ISBN: 978-1-4503-1003-1.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. (2017). "Adversarial attacks on neural network policies". In: *arXiv preprint arXiv:1702.02284*.

Hyndman, R. J. and Khandakar, Y. (2008). "Automatic time series forecasting: the forecast package for R". In: *Journal of Statistical Software* 26.3, pp. 1–22. URL: http://www.jstatsoft.org/article/view/v027i03.

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., and Yasmeen, F. (2018).

*forecast: Forecasting functions for time series and linear models*. R package version 8.4. URL: http://pkg.robjhyndman.com/forecast.

Insua, D. R. (1990). *Sensitivity Analysis in Multiobjective Decision Making*. Springer Verlag.

Insua, D. R. and Ruggeri, F. (2012). *Robust Bayesian Analysis*. Vol. 152. Springer Science & Business Media.

Jacquier, E., Johannes, M., and Polson, N. (2007). "MCMC maximum likelihood for latent state models". In: *Journal of Econometrics* 137.2, pp. 615–640.

Jain, M., Pita, J., Tambe, M., Ordónez, F., Paruchuri, P., and Kraus, S. (2008). "Bayesian Stackelberg games and their application for security at Los Angeles International Airport". In: *ACM SIGecom Exchanges* 7.2, pp. 1–3.

Jennrich, R. (1969). "Properties on non-linear least squares estimators". In: *Annals of Mathematical Statistics* 40.2, pp. 633–643.

Jensen, F. V. and Gatti, E. (2012). "Information enhancement—A tool for approximate representation of optimal strategies from influence diagrams". In: *International Journal of Approximate Reasoning* 53.9, pp. 1388–1396.

Jeroslow, R. G. (1985). "The polynomial hierarchy and a simple model for competitive analysis". In: *Mathematical programming* 32.2, pp. 146–164.

Johanson, M., Bard, N., Lanctot, M., Gibson, R. G., and Bowling, M. (2012). "Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization". In: *AAMAS*, pp. 837–846.

Joseph, A., Nelson, B., Rubinstein, B., and Tygar, J. (2019). *Adversarial Machine Learning*. Cambridge University Press. ISBN: 9781107043466. URL: https://books.google.com/books?id=XkSntAEACAAJ.

Juuti, M., Sun, B., Mori, T., and Asokan, N (2018). "Stay on-topic: Generating context-specific fake restaurant reviews". In: *European Symposium on Research in Computer Security*. Springer, pp. 132–151.

Kadane, J. B. and Larkey, P. D. (1982). "Subjective probability and the theory of games". In: *Management Science* 28, pp. 113–120.

Kantarcıoğlu, M., Xi, B., and Clifton, C. (2011). "Classifier evaluation and attribute selection against active adversaries". In: *Data Mining and Knowledge Discovery* 22, pp. 291–335.

Keeney, R. (2007). "Modeling values for anti-terrorism analysis". In: *Risk Analysis* 27.3, pp. 585–596.

Kim, J.-H. (2009). "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap". In: *Computational Statistics and Data Analysis* 53.11, pp. 3735–3745.

Kleijnen, J. P. (1992). "Regression metamodels for simulation with common random numbers: comparison of validation tests and confidence intervals". In: *Management Science* 38, pp. 1164–1185.

Kołcz, A. and Teo, C. H. (2009). "Feature Weighting for Improved Classifier Robustness". In: *CEAS'09: Sixth Conference on Email and Anti-Spam*.

Kolstad, C. D. and Lasdon, L. S. (1990). "Derivative evaluation and computational experience with large bilevel mathematical programs". In: *Journal of optimization theory and applications* 65.3, pp. 485–499.

Kolter, Z. and Madry, A. (2018). *Adversarial Robustness - Theory and Practice*. `https://adversarial-ml-tutorial.org/adversarial_examples/`.

Kos, J., Fischer, I., and Song, D. (2018). "Adversarial examples for generative models". In: *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, pp. 36–42.

LeCun, Y., Cortes, C., and Burges, C. (1998). *THE MNIST DATABASE of handwritten digits*. `http://yann.lecun.com/exdb/mnist/`.

Li, B. and Vorobeychik, Y. (2014). "Feature cross-substitution in adversarial classification". In: *Advances in Neural Information Processing Systems*, pp. 2087–2095.

Lichman, M. (2013). *UCI Machine Learning Repository*. `http://archive.ics.uci.edu/ml`.

Littman, M. L. (1994). "Markov games as a framework for multi-agent reinforcement learning". In: *Machine Learning Proceedings 1994*. Elsevier, pp. 157–163.

Littman, M. L. (2001). "Friend-or-Foe Q-learning in General-Sum Games". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., pp. 322–328.

Lowd, D. and Meek, C. (2005). "Adversarial learning". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD '05. Chicago, Illinois, USA, pp. 641–647. ISBN: 1-59593-135-X.

MacAfee (2014). *Net Losses: Estimating the Global Cost of Cybercrime*. Tech. rep. Intel Security.

Maclaurin, D., Duvenaud, D., and Adams, R. (2015). "Gradient-based hyperparameter optimization through reversible learning". In: *International Conference on Machine Learning*, pp. 2113–2122.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=rJzIBfZAb`.

Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. (2015). "Long short term memory networks for anomaly detection in time series". In: *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. P. 89.

Mallick, B. K., Ghosh, D., and Ghosh, M. (2005). "Bayesian classification of tumours by using gene expression data". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 219–234.

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. (2003). "Markov chain Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 100.26, pp. 15324–15328.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. V. (2017). "Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning". In: *International Joint Conferences on Artificial Intelligence, Inc.*

Michelmore, R., Kwiatkowska, M., and Gal, Y. (2018). "Evaluating uncertainty quantification in end-to-end autonomous driving control". In: *arXiv preprint arXiv:1811.06817*.

Miller, J. W. and Dunson, D. B. (2019). "Robust Bayesian inference via coarsening". In: *Journal of the American Statistical Association* 114.527, pp. 1113–1125.

Mokhtari, A., Ozdaglar, A., and Pattathil, S. (2019). "A Unified Analysis of Extra-gradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach". In: *arXiv preprint arXiv:1901.08511*.

Mortenson, M. J., Doherty, N. F., and Robinson, S. (2015). "Operational research from Taylorism to Terabytes: A research agenda for the analytics age". In: *European Journal of Operational Research* 241.3, pp. 583–595.

Müller, P. (2005). "Simulation based optimal design". In: *Handbook of Statistics* 25, pp. 509–518.

Müller, P., Sansó, B., and De Iorio, M. (2004). "Optimal Bayesian design by inhomogeneous Markov chain simulation". In: *Journal of the American Statistical Association* 99.467, pp. 788–798.

Nath, H. V. and Mehtre, B. M. (2014). "Static malware analysis using machine learning methods". In: *International Conference on Security in Computer Networks and Distributed Systems*. Springer, pp. 440–450.

Naveiro, R. and Insua, D. R. (2019). "Gradient Methods for Solving Stackelberg Games". In: *International Conference on Algorithmic DecisionTheory*. Springer, pp. 126–140.

Naveiro, R., Redondo, A., Insua, D. R., and Ruggeri, F. (2019a). "Adversarial classification: An adversarial risk analysis approach". In: *International Journal of Approximate Reasoning* 113, pp. 133–148.

Naveiro, R., Rodríguez, S., and Ríos Insua, D. (2019b). "Large-scale automated forecasting for network safety and security monitoring". In: *Applied Stochastic Models in Business and Industry*.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Vol. 1. Cambridge University Press Cambridge.

Owen, A. and Zhou, Y. (2000). "Safe and effective importance sampling". In: *Journal of the American Statistical Association* 95, pp. 135–143.

Papernot, N., McDaniel, P., Swami, A., and Harang, R. (2016). "Crafting adversarial input sequences for recurrent neural networks". In: *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, pp. 49–54.

Park, T. and Casella, G. (2008). "The Bayesian lasso". In: *Journal of the American Statistical Association* 103.482, pp. 681–686.

Parzen, E. (1962). "On Estimation of a probability density function and mode". In: *The Annals of Mathematical Statistics* 33.3, pp. 1065–1076.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). "Automatic differentiation in PyTorch". In:

Petris, G., Petrone, S., and Campagnoli, P. (2009). *Dynamic Linear Models with R*. useR! Springer-Verlag, New York.

Ponsen, M, Jong, S de, and Lanctot, M (2011). "Computing approximate Nash equilibria and robust best responses using sampling". In: *Journal of AI Research* 42, pp. 575–605.

Pontryagin, L. S. (2018). *Mathematical theory of optimal processes*. Routledge.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8.

Raiffa, H. (1982). *The Art and Science of Negotiation*. Harvard University Press.

Rios Insua, D. and Ruggeri, F. (2000). "Robust Bayesian Analysis". In: *Lecture Notes in Statistics* 152.

Rios Insua, D., Rios, J., and Banks, D. (2009). "Adversarial risk analysis". In: *Journal of the American Statistical Association* 104.486, pp. 841–854.

Rios Insua, D., Ruggeri, F., and Wiper, M. (2012). *Bayesian Analysis of Stochastic Process Models*. Wiley.

Ríos Insua, D., Naveiro, R., Gallego, V., and Poulos, J. (2019). "Adversarial Machine Learning: Perspectives from adversarial risk analysis". In: *arXiv e-prints*, arXiv:1908.06901, arXiv:1908.06901. arXiv: 1908.06901 [cs.GT].

Rios Insua, D., Couce-Vieira, A., Rubio, J. A., Pieters, W., Labunets, K., and G. Rasines, D. (2019). "An adversarial risk analysis framework for cybersecurity". In: *Risk Analysis*.

Ríos Insua, D., Banks, D., Ríos, J., and González-Ortega, J. (2020). "Adversarial Risk Analysis to support expert judgement elicitation". In: *Expert Judgement in Decision and Risk Analysis*. Ed. by H. B. French Nane. Springer.

Rios, J. and Rios Insua, D. (2012). "Adversarial Risk Analysis for Counterterrorism Modeling". In: *Risk Analysis* 32, pp. 894–915.

Roberts, G. O. and Smith, A. F. (1994). "Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms". In: *Stochastic processes and their applications* 49.2, pp. 207–216.

Romano, J. P. (1988). "On weak convergence and optimality of kernel density estimates of the mode". In: *The Annals of Statistics*, pp. 629–647.

Rossi, F. and Tsoukias, A. (2009). *Algorithmic Decision Theory: First International Conference, ADT 2009, Venice, Italy, October 2009, Proceedings*. Vol. 5783. Springer.

Sedgewick, A. (2014). *Framework for Improving Critical Infrastructure Cybersecurity*. Tech. rep. NIST.

Shabtai, A., Moskovitch, R., Elovici, Y., and Glezer, C. (2009). "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey". In: *information security technical report* 14.1, pp. 16–29.

Shachter, R. D. (1986). "Evaluating Influence Diagrams". In: *Operations Research* 34, pp. 871–882.

Shao, J. (1989). "Monte Carlo approximations in Bayesian decision theory". In: *Journal of the American Statistical Association* 84.407, pp. 727–732.

Shyamalkumar, N. (2000). "Likelihood robustness". In: *Robust Bayesian Analysis*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). "Mastering the game of go without human knowledge". In: *Nature* 550.7676, p. 354.

Sinha, A., Malo, P., and Deb, K. (2018). "A review on bilevel optimization: from classical to evolutionary approaches and applications". In: *IEEE Transactions on Evolutionary Computation* 22.2, pp. 276–295.

Song, Y., Kołcz, A., and Giles, C. L. (2009). "Better Naive Bayes classification for high-precision spam detection". In: *Software: Practice and Experience* 39, pp. 1003–1024.

Stahl, D. O. and Wilson, P. W. (1994). "Experimental evidence on players' models of other players". In: *Journal of Economic Behavior & Organization* 25.3, pp. 309–327.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). "Intriguing properties of neural networks". In: *International Conference on Learning Representations*. URL: http://arxiv.org/abs/1312.6199.

Taylor, S. J. and Letham, B. (2018). "Forecasting at scale". In: *The American Statistician* 72.1, pp. 37–45.

The Geneva Association (2016). *Ten Key Questions on Cyber Risk and Cyber Risk Insurance*. Tech. rep. The Geneva Association.

Tierney, L. (1994). "Markov chains for exploring posterior distributions". In: *The Annals of Statistics*, pp. 1701–1728.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2018). "Ensemble Adversarial Training: Attacks and Defenses". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rkZvSe-RZ.

Vallis, O., Hochenbaum, J., and Kejariwal, A. (2014). "A Novel Technique for Long-Term Anomaly Detection in the Cloud." In: *HotCloud*.

Vorobeychik, Y. and Kantarcioglu, M. (2018). "Adversarial machine learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12.3, pp. 1–169.

Vorobeychik, Y. and Li, B. (2014). "Optimal randomized classification in adversarial settings". In: *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*. AAMAS '14. Paris, France, pp. 485–492. ISBN: 978-1-4503-2738-1.

West, M. and Harrison, J. (1997). *Bayesian Forecasting & Dynamic Models*. Springer.

World Economic Forum (2020). *The Global Risks Report*. Tech. rep. World Economic Forum.

Ye, N. and Zhu, Z. (2018). "Bayesian adversarial learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., pp. 6892–6901.

You, I. and Yim, K. (2010). "Malware obfuscation techniques: A brief survey". In: *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*. IEEE, pp. 297–300.

Zeager, M. F., Sridhar, A., Fogal, N., Adams, S., Brown, D. E., and Beling, P. A. (2017). "Adversarial learning in credit card fraud detection". In: *Systems and Information Engineering Design Symposium (SIEDS), 2017*. IEEE, pp. 112–116.

Zhang, Y., Jin, R., and Zhou, Z.-H. (2010). "Understanding bag-of-words model: a statistical framework". In: *International Journal of Machine Learning and Cybernetics* 1.1-4, pp. 43–52.

Zhou, Y., Kantarcioglu, M., Thuraisingham, B., and Xi, B. (2012). "Adversarial support vector machine learning". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1059–1067.

Zhou, Y., Kantarcioglu, M., and Xi, B. (2019). "A survey of game theoretic approach for adversarial machine learning". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.3, e1259.

Zhuang, J. and Bier, V. M. (2007). "Balancing terrorism and natural disasters—Defensive strategy with endogenous attacker effort". In: *Operations Research* 55.5, pp. 976–991.