# ADVERSARIAL HYPOTHESIS TESTING (AHT)

- Use concepts from Adversarial Risk Analysis (ARA)
- Agent (Defender D, she) needs to ascertain which of several hypotheses holds, based on observations from a source
- Another agent (Attacker A, he) alters the observations to induce the Defender to make a wrong decision (and get a benefit)
- AHT problem studied from the Defender's perspective
- Defender needs to forecast the Attacker's decision, simulating from the corresponding Attacker's decision making problem

# AHT: SIMPLE EXAMPLE

- Defender D needs to decide whether a batch of e-mails includes spam or not
- D has beliefs about the standard flow of legit and spam messages
- Attacker A alters such flow in an attempt to confound the Defender and gain some benefit
- Both agents obtain different rewards depending on whether
  - batch is accepted or not by the Defender
  - batch includes just legit messages or not

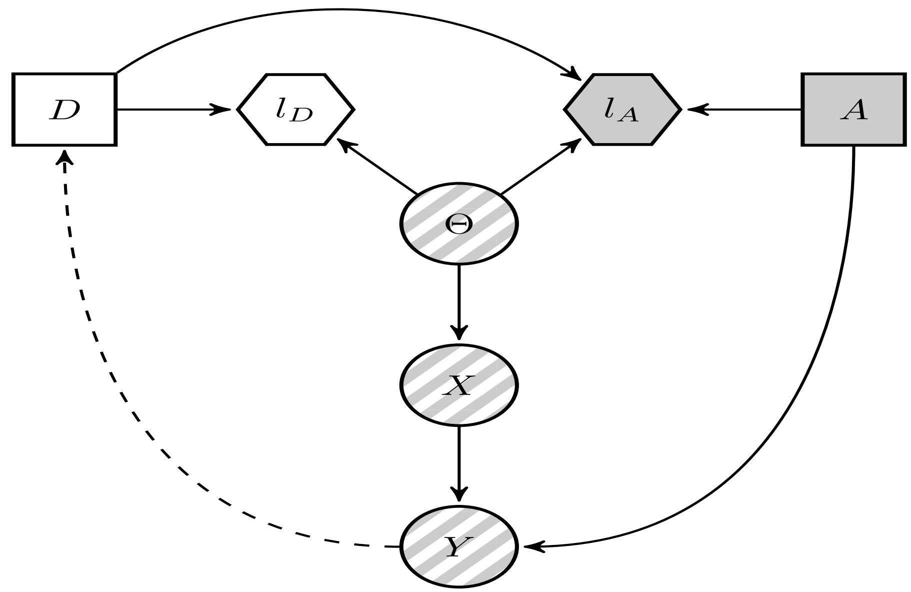
# ADVERSARIAL HYPOTHESIS TESTING

- Test of two simple hypotheses: $\Theta = \{\theta_0, \theta_1\}$
- Observation $x$ generated according to a model depending on $\theta$
- $x$ altered to $y$ by A's action $a$
- $y$ observed by D $\Rightarrow$ $D$'s decision $d$ on $\theta$ based on $y$, without observing $x$
- Depending on $d$ and actual $\theta \Rightarrow$ losses (utilities) for both agents
- Efforts by A in minimizing his loss
- Support for D in choosing $\theta$ to minimise her loss

# INFLUENCE DIAGRAMS

- Directed acyclic graph with three kinds of nodes:
  - Square: decision node
  - Circle: random node
  - Hexagon: value node (e.g. utility/loss)
- Arrows into a value or uncertainty node indicate functional and probabilistic dependence, respectively
  $\Rightarrow$ utility function at the value node depends on its immediately preceding nodes and probabilities at a chance node are conditional on the values of its direct predecessors
- Arrows into a decision node indicate that, when the decision is made, the values of its preceding nodes are known

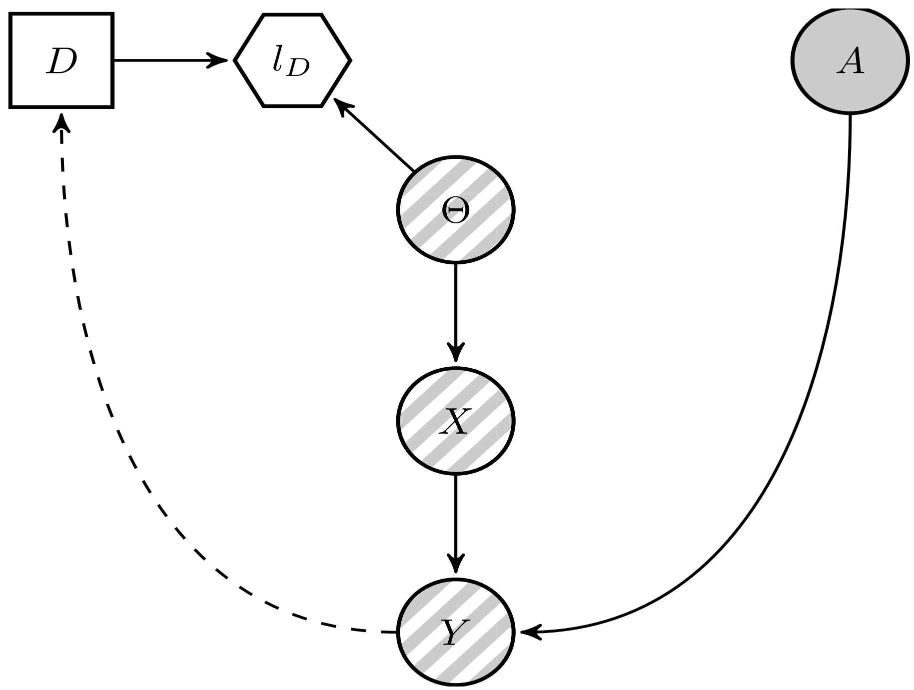
# AHT: BI-AGENT INFLUENCE DIAGRAM (BAID)




- Decisions: $D$ (depending on $Y$) and $A$
- Random: $\Theta \to X \to Y$ ($Y$ influenced also by the decision $A$)
- Losses: $l_D$ and $l_A$ depending on $\Theta$ and related decisions ($l_A$ also on decision $D$)

# SOLVING THE DEFENDER'S PROBLEM

Influence diagram of the Defender's decision problem




Attacker's node is now random

# SOLVING THE DEFENDER'S PROBLEM

Assessed by Defender D:

- Belief $\pi_D(\theta)$ on hypotheses:
  $p_D(\theta = \theta_i) = \pi_i^D, \quad i = 0, 1$, with $\pi_i^D \geq 0$ and $\pi_0^D + \pi_1^D = 1$
- Belief $\pi_D(x|\theta)$ on how data depend on the hypothesis:
  $X|\theta_i \sim \pi_D(x|\theta_i), \quad i = 0, 1$
- Belief $\pi_D(y|x, a)$ on how action $a \in \mathcal{A}$ by Attacker modifies actual $x$ into observed $y$
- Belief $\pi_D(a)$ on the attack $a$ performed by the Attacker
- Standard 0-1-$c_D$ loss function $l_D(d, \theta)$ with decision space $\mathcal{D} = \{d_0, d_1\}$ s.t.
  $d_j = \{\text{Defender supports } \theta_j\}, j = 0, 1$

# SOLVING THE DEFENDER'S PROBLEM

Defender's loss function

| | | Actual Hypothesis | |
|---|---|---|---|
| | | $\theta_0$ | $\theta_1$ |
| **D's Decision** | $d_0$ | 0 | 1 |
| | $d_1$ | $c_D$ | 0 |

- 0 best loss, associated with the *right* decision
- $c_D \leq 1$ (without loss of generality)

# SOLVING THE DEFENDER'S PROBLEM

- Solve: $\arg\min_{d \in \mathcal{D}} \sum_{i=0}^{1} l_D(d, \theta_i)\, \pi_D(\theta_i|y)$

- $\Rightarrow d_0$, i.e. support for $\theta_0$, optimal solution for D if and only if $\pi_D(\theta_1|y) \leq c_D\, \pi_D(\theta_0|y)$

- From

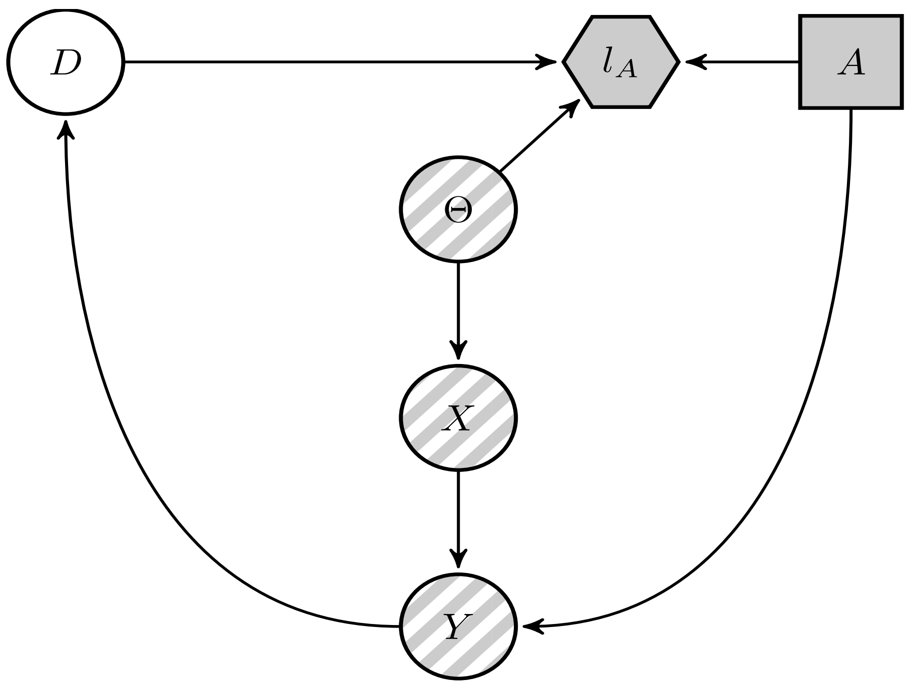
$$\begin{aligned}\pi_D(\theta_i|y) &= \frac{\pi_D(\theta_i, y)}{\pi_D(y)} = \frac{\iint \pi_D(\theta_i)\, \pi_D(y|x,a)\, \pi_D(x|\theta_i)\, \pi_D(a)\, \mathrm{d}x\, \mathrm{d}a}{\pi_D(y)} \\ &= \frac{\pi_i^D \iint \pi_D(y|x,a)\, \pi_D(x|\theta_i)\, \pi_D(a)\, \mathrm{d}x\, \mathrm{d}a}{\pi_D(y)}, \quad i = 0, 1\end{aligned}$$

- $\Rightarrow$ support for $\theta_0$, optimal decision for D if and only if

$$\pi_1^D \iint \pi_D(y|x,a)\, \pi_D(x|\theta_1)\, \pi_D(a)\, \mathrm{d}x\, \mathrm{d}a \leq c_D\, \pi_0^D \iint \pi_D(y|x,a)\, \pi_D(x|\theta_0)\, \pi_D(a)\, \mathrm{d}x\, \mathrm{d}a$$

# SOLVING THE ATTACKER'S PROBLEM

- All Defender's beliefs obtained in standard way, except for $\pi_D(a)$
- Defender's belief $\pi_D(a)$ on Attacker's action comes from considering his decision problem
- Defender's node is now random



# SOLVING THE ATTACKER'S PROBLEM

Needed for Attacker A:

- Belief $\pi_A(\theta)$ on hypotheses:
  $p_A(\theta = \theta_i) = \pi_i^A, \quad i = 0, 1$, with $\pi_i^A \geq 0$ and $\pi_0^A + \pi_1^A = 1$
- Belief $\pi_A(x|\theta)$ on how data depend on the hypothesis:
  $X|\theta_i \sim \pi_A(x|\theta_i), \quad i = 0, 1$
- Belief $\pi_A(y|x, a)$ on consequences of his action $a \in \mathcal{A}$, modifying actual $x$ into $y$
- Belief $\pi_A(d|y)$ on the decision $d$ taken by the Defender upon observing $y$
- Loss function $l_A(d, \theta, a) = l_{jk}(a)$, with
  - $j = 0, 1$ depending on Defender's decision $d_j$ (i.e., supporting $\theta_j$)
  - $k = 0, 1$ depending on actual $\theta_k$
  - No cost directly associated with chosen action $a$ (but only on consequences)

# SOLVING THE ATTACKER'S PROBLEM

Attacker's loss function

| | | **Actual Hypothesis** | |
|---|---|---|---|
| | | $\theta_0$ | $\theta_1$ |
| **D's Decision** | $d_0$ | $l_{00}(a)$ | $l_{01}(a)$ |
| | $d_1$ | $l_{10}(a)$ | $l_{11}(a)$ |

- Better for the Attacker if the Defender makes mistakes

  $\Rightarrow l_{00}(a) \geq l_{01}(a)$ and $l_{10}(a) \leq l_{11}(a)$

# SOLVING THE ATTACKER'S PROBLEM

Attacker's loss function

| | | Actual Hypothesis | |
|---|---|---|---|
| | | $\theta_0$ | $\theta_1$ |
| **D's Decision** | $d_0$ | 1 | 0 |
| | $d_1$ | $c_A^1$ | $c_A^2$ |

$0 \leq c_A^1 \leq c_A^2 \leq 1$

- Best loss for Attacker (0) when Defender supports $\theta_0$ and she should not
- Worst loss for Attacker (1) when Defender supports $\theta_0$ and she should
- Intermediate cases: worse for Attacker when Defender supports $\theta_1$ and actual hypothesis is $\theta_1$

# SOLVING THE ATTACKER'S PROBLEM

- Optimal decision for Attacker given by $a^*$ s.t.

  $a^* = \arg\min_{a \in \mathcal{A}} \sum_{j=0}^{1} \sum_{i=0}^{1} \iint l_A(d_j, \theta_i, a)\, \pi_A(d_j|y)\, \pi_A(\theta_i)\, \pi_A(y|x,a)\, \pi_A(x|\theta_i)\, \mathrm{d}y\, \mathrm{d}x$

- Defender does not know $\pi_A(\theta)$, $\pi_A(x|\theta)$, $\pi_A(y|x,a)$, $\pi_A(d|y)$ and $l_A(d,\theta,a)$

- $\Rightarrow$ model uncertainty around them through random probabilities and losses
  $F = (\Pi_A(\theta), \Pi_A(x|\theta), \Pi_A(y|x,a), \Pi_A(d|y), L_A(d,\theta,a))$

- $\Rightarrow$ find optimal random attack

  $A^* = \arg\min_{a \in \mathcal{A}} \sum_{j=0}^{1} \sum_{i=0}^{1} \iint L_A(d_j, \theta_i, a)\, \Pi_A(d_j|y)\, \Pi_A(\theta_i)\, \Pi_A(y|x,a)\, \Pi_A(x|\theta_i)\, \mathrm{d}y\, \mathrm{d}x$

- $\Rightarrow$ required distribution through $\pi_D(a) = \Pi(A^* = a)$ (assuming discrete $\mathcal{A}$, but possible also for continuous one)

# SOLVING THE ATTACKER'S PROBLEM

- $\pi_D(a)$ approximated through simulation, sampling from $F$
- Samples $\big(\Pi_A^k(\theta_i), \Pi_A^k(x|\theta_i), \Pi_A^k(y|x,a), \Pi_A^k(d_j|y), L_A^k(d_j,\theta_i,a)\big)$, $k = 1, ..., K$
- $\Rightarrow a_k^* = \arg\min_{a\in\mathcal{A}} \sum_{j=0}^{1} \sum_{i=0}^{1} \iint L_A^k(d_j,\theta_i,a)\, \Pi_A^k(d_j|y)\, \Pi_A^k(\theta_i)\, \Pi_A^k(y|x,a)\, \Pi_A^k(x|\theta_i)\, \mathrm{d}y\, \mathrm{d}x$
- $\Rightarrow \widehat{\pi}_D(a) \approx \#\{a_k^* = a\}/K$

# SOLVING THE ATTACKER'S PROBLEM

Choice of random probabilities and loss $F$

- $\Pi_A(\theta)$ based on $\pi_D(\theta)$ with some uncertainty around it
  - $\Pi_A(\theta)$ modelled as a Dirichlet distribution with mean $\pi_D(\theta)$, if discrete
  - $\Pi_A(\theta)$ modelled as Dirichlet process with base measure $\pi_D(\theta)$, if continuous
- $\Pi_A(x|\theta)$ based on $\pi_D(x|\theta)$ with some uncertainty around it
- $\Pi_A(y|x,a)$ based on $\pi_D(y|x,a)$ with some uncertainty around it
- Parametric form for $L_A(d,\theta,a)$ with distribution over such parameters
- On the contrary, $\Pi_A(d|y)$ requires strategic thinking as the Defender needs to assess the Attacker's beliefs about which decision $d$ she will make, given that she observes $y$
- $\Rightarrow$ could be the start of a hierarchy of decision making problems!

# SOLVING THE ATTACKER'S PROBLEM

- Defender should solve the problem

  $\arg\min_{d \in \mathcal{D}} \sum_{i=0}^{1} l_D(d, \theta_i)\, \pi_D(\theta_i | y)$ equivalent to

  $\arg\min_{d \in \mathcal{D}} \sum_{i=0}^{1} \int \int l_D(d, \theta_i)\, \pi_D(\theta_i)\, \pi_D(y | x, a)\, \pi_D(x | \theta_i)\, \pi_D(a)\, \mathrm{d}x\, \mathrm{d}a$

- Attacker does not know ingredients of above integral

- $\Rightarrow$ assume uncertainty over them through random loss $L_D^A(d, \theta)$ and random distributions $\Pi_D^A(\theta)$, $\Pi_D^A(y | x, a)$, $\Pi_D^A(x | \theta)$ and $\Pi_D^A(a)$

- $\Rightarrow$ get corresponding random optimal decision

- Assessment of $\Pi_D^A(a)$ (what Defender believes that Attacker thinks about her beliefs concerning the attack to be implemented)
  $\Rightarrow$ strategic component leading to the next stage in the hierarchy

- Iterate until no further information is available, then choosing non-informative prior over the involved probabilities and losses

# NUMERICAL EXAMPLE

- Two hypotheses: $\theta_0 = 2$ and $\theta_1 = 1$
- Two decisions: $d_0$ chooses $\theta_0 = 2$ and $d_1$ chooses $\theta_1 = 1$
- Priors over the hypotheses: $\pi_0^D = \pi_1^D = 1/2$
- Actual data $X|\theta_i$ exponentially distributed $\mathcal{E}(\theta_i)$, with uncertainty about $\theta_i$
- Data $x$ modified by Attacker into $y$, with actions
  - $a_0$: $x \to y = x$ (*keeping*)
  - $a_1$: $x \to y = 2x$ (*doubling*)
  - $a_{-1}$: $x \to y = x/2$ (*halving*)
- Suppose (for illustration) Defender knows probabilities $\pi_D(a)$ used by Attacker to choose actions:

  $\pi_D(a_0) = 1/2$, $\pi_D(a_1) = 1/6$ and $\pi_D(a_{-1}) = 1/3$

# NUMERICAL EXAMPLE

- Two decisions: $d_0$ chooses $\theta_0 = 2$ and $d_1$ chooses $\theta_1 = 1$
- Loss function $L(d, \theta)$

| | | Actual Hypothesis | |
|---|---|---|---|
| | | $\theta_0$ | $\theta_1$ |
| **D's Decision** | $d_0$ | 0 | 1 |
| | $d_1$ | 3/4 | 0 |

# NUMERICAL EXAMPLE

Adopt decision $d_0$ (i.e., accept $\theta_0 = 2$) if and only if

$$\pi_1^D \left[\theta_1\, e^{-\theta_1\, y}\, \pi_D(a_0) + \theta_1\, e^{-\theta_1\, \frac{y}{2}}\, \pi_D(a_1) + \theta_1\, e^{-\theta_1\, 2y}\, \pi_D(a_{\text{-}1})\right]$$

$$\leq$$

$$\frac{3}{4}\, \pi_0^D \left[\theta_0\, e^{-\theta_0\, y}\, \pi_D(a_0) + \theta_0\, e^{-\theta_0\, \frac{y}{2}}\, \pi_D(a_1) + \theta_0\, e^{-\theta_0\, 2y}\, \pi_D(a_{\text{-}1})\right]$$

- $\Leftrightarrow 2e^{-\frac{y}{2}} + 3e^{-y} - 5e^{-2y} - 6e^{-4y} \leq 0$

- $\Leftrightarrow y \lesssim 0.3723$ is observed

  (Note that $\theta = 2$ leads to a smaller mean w.r.t. $\theta = 1$, i.e. $1/2$ vs. $1$)

- Note that a small change in probabilities, i.e. $\pi_0^D = 1/3$ and $\pi_1^D = 2/3$ (and other probabilities and losses kept as before) $\Rightarrow d_1$ optimal regardless of observed $y$

# NUMERICAL EXAMPLE

Defender does not accurately know $\pi_D(a) \Rightarrow$ ARA

- $\Pi_A(\theta_1)$ drawn uniformly over $[1/4, 3/4]$, and $\Pi_A(\theta_0) = 1 - \Pi_A(\theta_1)$

- $\Pi_A(x|\theta)$, where $\theta \in \{\theta_0, \theta_1\}$, from a Gamma distribution $\mathcal{G}a(\alpha, \beta)$ with mean $\alpha/\beta = \theta$ and variance $\alpha/\beta^2 = \sigma^2$ uniformly chosen over $[1/2, 2]$ s.t. variance randomness induces that of $\Pi_A(x|\theta)$

- $\Pi_A(y|x, a)$ Dirac distributions coinciding with those of $\pi_D(y|x, a)$

- $\Pi_A(d|y)$ looking at the likelihood $h(y|d, a)$ of $y$ under different choices of $d$ and $a$, mixing them through a random allocation of probabilities to each action

# NUMERICAL EXAMPLE

- Attacker assumes the Defender is modelling the data with an exponential distribution
- Likelihood $h(y|d,a)$ of $y$ under different choices of $d$ and $a$
  - $d_0$ chooses $\theta_0 = 2$ and $d_1$ chooses $\theta_1 = 1$
  - $a_0$ (*keeping*), $a_1$ (*doubling*) and $a_{-1}$ (*halving*)
- Example
  - $y$ reported and $a_1$ chosen $\Rightarrow x = y/2$ true value
  - $d_0$ chosen $\Rightarrow h(y|d_0, a_1) = 2e^{-y}$

| | | Actions | | |
|---|---|---|---|---|
| | | $a_0$ | $a_1$ | $a_{-1}$ |
| **D's Decision** | $d_0$ | $2e^{-2y}$ | $2e^{-y}$ | $2e^{-4y}$ |
| | $d_1$ | $e^{-y}$ | $2e^{-y/2}$ | $e^{-2y}$ |

# NUMERICAL EXAMPLE

- Defender assessing the probabilities $(\epsilon_0, \epsilon_1, \epsilon_{\text{-}1})$ assigned by the Attacker to each strategy through a Dirichlet distribution $\mathcal{D}ir(1,1,1)$

- $\Rightarrow$

$$P_A(d = d_1 | \epsilon_0, \epsilon_1, \epsilon_{\text{-}1}, y) \qquad \frac{\sum_{j=\text{-}1}^{1} \epsilon_j \, h(y|d_1, a_j)}{\sum_{j=\text{-}1}^{1} \epsilon_j \, h(y|d_0, a_j) + \sum_{j=\text{-}1}^{1} \epsilon_j \, h(y|d_1, a_j)}$$

$$= \frac{\epsilon_0 \, e^{-y} \epsilon_1 \, e^{-\frac{y}{2}} + \epsilon_{\text{-}1} \, e^{-2y}}{2\,(\epsilon_0 \, e^{-2y} + \epsilon_1 \, e^{-y} + \epsilon_{\text{-}1} \, e^{-4y}) + \epsilon_0 \, e^{-y} + \epsilon_1 \, e^{-\frac{y}{2}} + \epsilon_{\text{-}1} \, e^{-2y}}$$

- Distribution of $(\epsilon_0, \epsilon_1, \epsilon_{\text{-}1})$ induces the randomness of $P_A(d = d_1|y)$

- $P_A(d = d_0|y) = 1 - P_A(d = d_1|y)$

# NUMERICAL EXAMPLE

Random loss function $L_A(d, \theta, a)$ based on table below

- $C_A^1$ fixed at 0
- $C_A^2$ uniformly drawn from $[1/2, 1]$

| | | Actual Hypothesis | |
|---|---|---|---|
| | | $\theta_0$ | $\theta_1$ |
| **D's Decision** | $d_0$ | 1 | 0 |
| | $d_1$ | $C_A^1$ | $C_A^2$ |

# NUMERICAL EXAMPLE

- Attacker's random expected losses for the three actions

-

$$\Psi_A(a_0) = \int \left[\Pi_A(d_0|y = x)\,\Pi_A(\theta_0)\,\Pi_A(x|\theta_0) + C_A^2\,\Pi_A(d_1|y = x)\,\Pi_A(\theta_1)\,\Pi_A(x|\theta_1)\right] \mathrm{d}x$$

$$\Psi_A(a_1) = \int \left[\Pi_A(d_0|y = 2x)\,\Pi_A(\theta_0)\,\Pi_A(x|\theta_0) + C_A^2\,\Pi_A(d_1|y = 2x)\,\Pi_A(\theta_1)\,\Pi_A(x|\theta_1)\right] \mathrm{d}$$

$$\Psi_A(a_{\text{-}1}) = \int \left[\Pi_A(d_0|y = \tfrac{x}{2})\,\Pi_A(\theta_0)\,\Pi_A(x|\theta_0) + C_A^2\,\Pi_A(d_1|y = \tfrac{x}{2})\,\Pi_A(\theta_1)\,\Pi_A(x|\theta_1)\right] \mathrm{d}x$$

- Random models induce randomness in these expected losses

- $K = 100,000$ observations drawn from the corresponding distributions

- $\Rightarrow$ Estimates $\widehat{\pi}_D(a_0) \approx 0.04$, $\widehat{\pi}_D(a_1) \approx 0.85$ and $\widehat{\pi}_D(a_{\text{-}1}) \approx 0.11$

- Optimal action: $d_0$ when $y \lesssim 0.7374$ (different from previous solution)

# NUMERICAL EXAMPLE

Set $p_j = 0, \quad -1 \leq j \leq 1$.

**For** $k = 1$ **to** $K$

Generate $\pi_A^{1,k} \sim \mathcal{U}(1/4, 3/4)$. Compute $\pi_A^{0,k} = 1 - \pi_A^{1,k}$.

Generate $\sigma_{0,k}^2 \sim \mathcal{U}(1/2, 2)$. Compute $\alpha_0^k = \theta_0^2/\sigma_{0,k}^2; \quad \beta_0^k = \theta_0/\sigma_{0,k}^2$.

Generate $\sigma_{1,k}^2 \sim \mathcal{U}(1/2, 2)$. Compute $\alpha_1^k = \theta_1^2/\sigma_{1,k}^2; \quad \beta_1^k = \theta_1/\sigma_{1,k}^2$

Generate $(\epsilon_0^k, \epsilon_1^k, \epsilon_{-1}^k) \sim \mathcal{D}ir(1, 1, 1)$..

Generate $C_A^{2,k} \sim \mathcal{U}(1/2, 1)$.

$$\psi_A^k(a_0) = \pi_A^{0,k} \int (1 - g(\epsilon_0, \epsilon_1, \epsilon_{-1}, x))\ f(x|\alpha_0^k, \beta_0^k)\, \mathrm{d}x + C_A^{2,k}\, \pi_A^{1,k} \int g(\epsilon_0, \epsilon_1, \epsilon_{-1}, x)\, f(x|\alpha_1^k, \beta_1^k)\, \mathrm{d}x$$

$$\psi_A^k(a_1) = \pi_A^{0,k} \int (1 - g(\epsilon_0, \epsilon_1, \epsilon_{-1}, 2x))\ f(x|\alpha_0^k, \beta_0^k)\, \mathrm{d}x + C_A^{2,k}\, \pi_A^{1,k} \int g(\epsilon_0, \epsilon_1, \epsilon_{-1}, 2x)\, f(x|\alpha_1^k, \beta_1^k)\, \mathrm{d}x$$

$$\psi_A^k(a_{-1}) = \pi_A^{0,k} \int (1 - g(\epsilon_0, \epsilon_1, \epsilon_{-1}, x/2))\ f(x|\alpha_0^k, \beta_0^k)\, \mathrm{d}x + C_A^{2,k}\, \pi_A^{1,k} \int g(\epsilon_0, \epsilon_1, \epsilon_{-1}, x/2)\, f(x|\alpha_1^k, \beta_1^k)\, \mathrm{d}x$$

Determine $j^* = \underset{-1 \leq j \leq 1}{\arg\min}\, \psi_A^k(a_j)$.

Set $p_{j^*} = p_{j^*} + 1$.

Set $\hat{\pi}_D(a_j) = p_j/K, \quad -1 \leq j \leq 1$.

# BATCH ACCEPTANCE MODEL

- Problem: deciding whether to accept a batch of items received over a period of time, some of which could be faulty, thus entailing potential security and/or performance problems
- Type of issues arising in areas such as screening containers at international ports, accepting batches of electronic messages or admitting packages of perishable products or electronic components, among others
- Consider different scenarios for a batch with $m$ items in a period;
  - Loss depending if at least one faulty item is included ($1$ or $m$ faulty items give the same loss)
  - Loss depending on the number of included faulty items among the $m$
- Consider different Attacker's strategies:
  - $\mathcal{S}_1$. Attacker adds some, new faulty items
  - $\mathcal{S}_2$. Attacker modifies few original items converting them into faulty ones
  - $\mathcal{S}_3$. Attacker combines strategies $\mathcal{S}_1$ and $\mathcal{S}_2$

# BATCH ACCEPTANCE MODEL

- Problem: deciding whether to accept a batch of items received over a period of time, some of which could be faulty, thus entailing potential security and/or performance problems

- Type of issues arising in areas such as screening containers at international ports, accepting batches of electronic messages or admitting packages of perishable products or electronic components, among others

- We first outline a non-adversarial hypothesis testing problem which we then modify to include adversaries

# BATCH ACCEPTANCE MODEL

- Decision maker $D$ (*Defender*) receives a batch with two types of items $x$
  - 0 (acceptable items)
  - 1 (faulty items)
- $D$ needs to decide whether to accept ($d_0$) or reject ($d_1$) the batch
- $D$ observes the batch size, modelled by a Poisson distribution $\mathcal{P}o(\lambda)$ over a unit period (or a homogeneous Poisson process, HPP, of parameter $\lambda$)
- Distribution on $\lambda$ as a consequence of past experience:
  - Gamma prior $\mathcal{G}a(a, b)$ on $\lambda$
  - $r$ items arrived after $t$ periods $\Rightarrow$ posterior $\lambda | t, r \sim \mathcal{G}a(a + r, b + t)$
- $\lambda$ will have no impact when $D$ observes the actual value of $m$

# BATCH ACCEPTANCE MODEL

- Item acceptable with probability $\theta$

  $Z$ designates item acceptability, s.t. $z = 0$ acceptable and $z = 1$ faulty

  $\Rightarrow p_D(z = 0|\theta) = \theta$ and $p_D(z = 1|\theta) = 1 - \theta$

- Acceptability of an item independent of the arrival process $\Rightarrow$ arrival of acceptable items is HPP of parameter $\lambda\theta$ (*Coloring or Thinning Theorem*)

- Beta prior $\mathcal{B}e(\alpha, \beta)$ for $\theta$

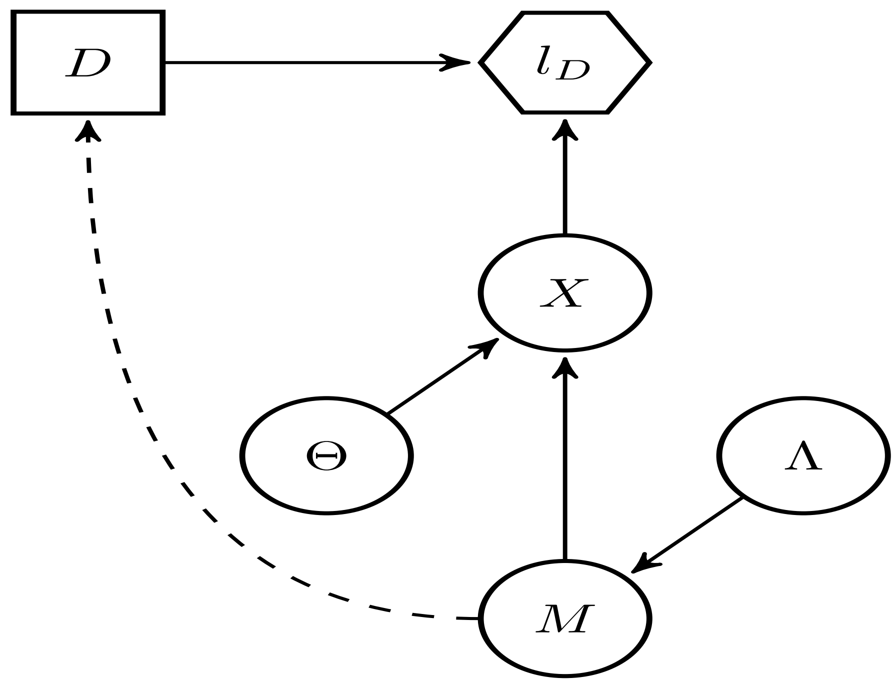
- Suppose $r$ received items with $s$ acceptable (and $r - s$ faulty)

  $\Rightarrow$ posterior $\theta|r, s \sim \mathcal{B}e(\alpha + s, \beta + r - s)$

- To fix ideas, in a unit period we shall have
  - Total number of items $m|\lambda \sim \mathcal{P}o(\lambda)$
  - Total number of acceptable items $x|\lambda, \theta \sim \mathcal{P}o(\lambda\theta)$
  - (Conditional on $m$) total number of acceptable items $x|m, \theta \sim \mathcal{B}in(m, \theta)$

# BATCH ACCEPTANCE MODEL

Influence diagram for batch acceptance problem without adversaries



# BATCH ACCEPTANCE MODEL

Scenario A: Winner takes it all

- Batch with $m$ items in a period
- Allowing one faulty item is as bad as allowing several of them, because of the entailed security or performance problems
- Loss function given by

| | | **Batch of $m$ Items** | | |
|---|---|---|---|---|
| | | All Acceptable | Some Faulty | |
| | | $p = \theta^m$ | $p = 1 - \theta^m$ | **Exp. Loss** |
| **D's Decision** | Accept, $d_0$ | 0 | 1 | $1 - \theta^m$ |
| | Reject, $d_1$ | $c$ | 0 | $c\,\theta^m$ |

# BATCH ACCEPTANCE MODEL

- Suppose batch size $m$ known to Defender $D \Rightarrow \lambda$ not relevant

- Expected losses of both decisions

$$l_D(d_0) = E_\theta\left[1 - \theta^m\right] = 1 - E_\theta\left[\theta^m\right]$$
$$l_D(d_1) = E_\theta\left[c\,\theta^m\right] = c\,E_\theta\left[\theta^m\right]$$

- Decision: accept the batch ($d_0$) if and only if

$$1 - E_\theta\left[\theta^m\right] \leq c\,E_\theta\left[\theta^m\right] \quad \Longleftrightarrow \quad E_\theta\left[\theta^m\right] \geq \frac{1}{1+c}$$

- $E_\theta\left[\theta^m\right]$ decreases as $m$ increases $\Rightarrow$ threshold value $m_A$
  $\Rightarrow$ rejection of the batch ($d_1$) if $m > m_A$

- $m_A$ recursively obtained for posterior $\mathcal{B}e(\alpha + s, \beta + r - s)$ on $\theta$ from
  $E_\theta\left[\theta^m\right] = \prod_{k=0}^{m-1} \dfrac{\alpha + s + k}{\alpha + \beta + r + k}$

# BATCH ACCEPTANCE MODEL

- Suppose batch size $m$ unknown to Defender $D$, with distribution $p(m|\lambda), m \in \mathcal{N}$
- Expected losses of both decisions (now summing over all possible values of $m$)

$$l_D(d_0) = 1 - E_\theta \left( E_\lambda \left( \sum_{m=0}^{\infty} \theta^m p(m|\lambda) \right) \right)$$
$$l_D(d_1) = c\, E_\theta \left( E_\lambda \left( \sum_{m=0}^{\infty} \theta^m p(m|\lambda) \right) \right)$$

- Decision: accept the batch ($d_0$) if and only if

$$E_\theta \left( E_\lambda \left( \sum_{m=0}^{\infty} \theta^m p(m|\lambda) \right) \right) > \frac{1}{c+1}$$

- If $p(m|\lambda)$ Poisson, then accept the batch ($d_0$) if and only if

$$E_\theta \left( E_\lambda \left( e^{\lambda(\theta-1)} \right) \right) > \frac{1}{c+1}$$

# BATCH ACCEPTANCE MODEL

- Gamma distribution $\mathcal{G}a(a,p)$ over $\lambda$ and Beta distribution $\mathcal{B}e(\alpha,\beta)$ over $\theta$

- $E_\lambda(e^{\lambda(\theta-1)}) = \int_0^\infty e^{-\lambda(1-\theta)} \frac{p^a}{\Gamma(a)} \lambda^{a-1} e^{-p\lambda} d\lambda = \frac{p^a}{(p+1-\theta)^a}$

$$
\begin{aligned}
E_\theta(E_\lambda(e^{\lambda(\theta-1)})) &= E_\theta\left(\frac{p^a}{(p+1-\theta)^a}\right) \\
&= \int_0^1 \frac{p^a}{(p+1-\theta)^a} \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} d\theta \\
&= \frac{p^a}{(p+1)^a B(\alpha,\beta)} \int_0^1 \theta^{\alpha-1}(1-\theta)^{\beta-1}\left(1-\frac{\theta}{p+1}\right)^{-a} d\theta \\
&= \frac{p^a}{(p+1)^a} {}_2F_1\left(a,\alpha;\alpha+\beta;\frac{1}{p+1}\right)
\end{aligned}
$$

- $\Rightarrow$ accept the batch when $\frac{p^a}{(p+1)^a} {}_2F_1\left(a,\alpha;\alpha+\beta;\frac{1}{p+1}\right) > \frac{1}{c+1}$

# BATCH ACCEPTANCE MODEL

Scenario B: Each fault counts

- Batch with $m$ items in a period
- Loss depending on the number of included faulty items
- Loss function given by

| | | **Batch of $m$ Items** | | |
|---|---|---|---|---|
| | | All Acceptable | $x$ Acceptable | |
| | | $p = \theta^m$ | $p = \binom{m}{x} \theta^x (1-\theta)^{m-x}$ | **Exp. Loss** |
| **D's Decision** | Accept, $d_0$ | 0 | $(m-x)\, c'$ | $m\, c' (1-\theta)$ |
| | Reject, $d_1$ | $c$ | 0 | $c\, \theta^m$ |

# BATCH ACCEPTANCE MODEL

- Suppose batch size $m$ known to Defender $D \Rightarrow \lambda$ not relevant
- Expected losses of both decisions

$$l_D(d_0) = E_\theta\left[m\,c'\,(1-\theta)\right] = m\,c'\,(1 - E_\theta\left[\theta\right])$$
$$l_D(d_1) = E_\theta\left[c\,\theta^m\right] = c\,E_\theta\left[\theta^m\right]$$

- Decision: accept the batch ($d_0$) if and only if

$$m\,c'\,(1 - E_\theta\left[\theta\right]) \leq c\,E_\theta\left[\theta^m\right] \quad \Longleftrightarrow \quad \frac{E_\theta\left[\theta^m\right]}{m} \geq \frac{c'}{c}\,(1 - E_\theta\left[\theta\right])$$

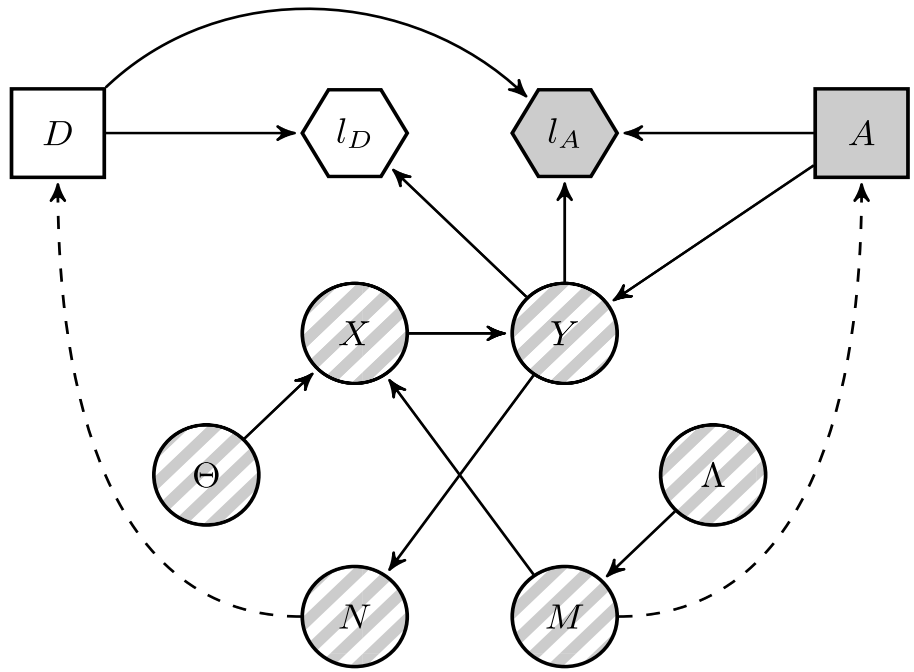
- $E_\theta\left[\theta^m\right]$ decreases as $m$ increases $\Rightarrow$ threshold value $m_B \Rightarrow$ rejection of the batch ($d_1$) if $m > m_B$
- $m_B$ recursively obtained for posterior $\mathcal{B}e(\alpha + s, \beta + r - s)$ on $\theta$ as the smallest integer satisfying

$$\frac{E_\theta\left[\theta^m\right]}{m} \leq \frac{c'}{c}\,\frac{\beta + r - s}{\alpha + \beta + r}$$

# ADVERSARIAL BATCH ACCEPTANCE MODEL

- Attacker might alter the batch $X$ to $Y$ and, thus, perturb the data flow process to confound the Defender and reach some objectives
- Batch of size $m$, with $m$ known by Attacker $A$
- Attacker $A$ might add items to get a final batch of size $n$
- Defender $D$ observes $n$ before making her decision
- Gain bigger for $A$ if $D$ accepts one of $A$'s faulty items rather than a faulty item from another source

# ADVERSARIAL BATCH ACCEPTANCE MODEL



# ADVERSARIAL BATCH ACCEPTANCE MODEL

We study three possible attack strategies, identifying

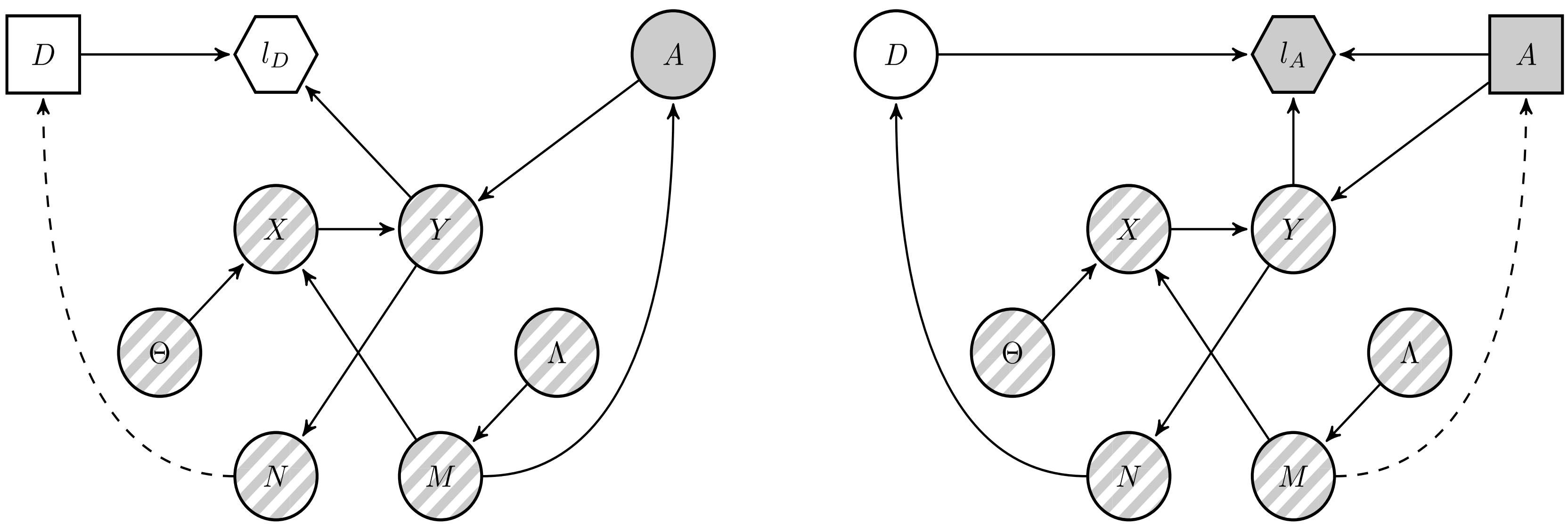
- Attacker's decision variables
- how the item arrival process changes
- Attacker's loss function
- how to solve the problem

The strategies are:

- $\mathcal{S}_1$. Attacker adds some, new faulty items
- $\mathcal{S}_2$. Attacker modifies few original items converting them into faulty ones
- $\mathcal{S}_3$. Attacker combines strategies $\mathcal{S}_1$ and $\mathcal{S}_2$

# ADVERSARIAL BATCH ACCEPTANCE MODEL

- $n$: number of items in a batch observed by Defender $D$
- $x$: acceptable items in the batch
- $m - x$: original faulty items ($O$-faults)
- $n - m$: faulty items produced by the Attacker $A$ ($A$-faults)




(a) Defender's problem (b) Attacker's problem

# ADVERSARIAL BATCH ACCEPTANCE MODEL

$\mathcal{S}_1$. Attacker adds $y_1$ new faulty items

- $m + y_1$ data received by Defender include
  - $x$ acceptable items
  - $m - x$ $O$-faults
  - $y_1$ $A$-faults
- Attacker needs to decide $y_1$, which is random to Defender
- Suppose first that Defender knows $p_D(y_1|m)$, distribution of $Y_1|m$
- Loss structure for Defender

| | | **Final Batch of $n$ Items** | | |
|---|---|---|---|---|
| | | All Acceptable | Some Faulty | |
| | | $p = q_1(n\|\lambda)$ | $p = 1 - q_1(n\|\lambda)$ | **Exp. Loss** |
| **D's Decision** | Accept, $d_0$ | 0 | 1 | $1 - q_1(n\|\lambda)$ |
| | Reject, $d_1$ | $c$ | 0 | $c\, q_1(n\|\lambda)$ |

# ADVERSARIAL BATCH ACCEPTANCE MODEL

- $n = m + y_1$

- Probability of having a final batch of $n$ items reflects all possible initial sizes of the batch and included faulty items, not just $m$ and $y_1$, respectively:

$$p_1(n|\lambda) = \sum_{m=0}^{n} p_D(m|\lambda)\, p_D(y_1 = n - m|m)$$

- Probability that all items are acceptable (i.e., $x = m = n$ and $y_1 = 0$)

$$q_1(n|\lambda) = \frac{p_D(m = n|\lambda)\, p_D(y_1 = 0|m = n)}{p_1(n|\lambda)}\, \theta^n$$

- $\lambda$ relevant here since it provides information on $m$

# ADVERSARIAL BATCH ACCEPTANCE MODE

| | | **Final Batch of $n$ Items** | | |
|---|---|---|---|---|
| | | All Acceptable | Some Faulty | |
| | | $p = q_1(n\|\lambda)$ | $p = 1 - q_1(n\|\lambda)$ | **Exp. Loss** |
| **D's Decision** | Accept, $d_0$ | 0 | 1 | $1 - q_1(n\|\lambda)$ |
| | Reject, $d_1$ | $c$ | 0 | $c\, q_1(n\|\lambda)$ |

- Expected losses of both decisions

$$l_D(d_0) = 1 - E_\theta\left[E_\lambda\left[q_1(n|\lambda)\right]\right]$$

$$l_D(d_1) = c\, E_\theta\left[E_\lambda\left[q_1(n|\lambda)\right]\right]$$

- Decision: accept the batch ($d_0$) if and only if

$$E_\theta\left[E_\lambda\left[q_1(n|\lambda)\right]\right] \geq \frac{1}{1+c}$$

- Decision obtained through simulation

# ADVERSARIAL BATCH ACCEPTANCE MODE

- $p_D(y_1|m)$ (and thus $q_1(n|\lambda)$) unknown to Defender $D \Rightarrow$ use ARA
- $x \in \{0, 1, \ldots, m\}$ acceptable items
- $y_1 \in \{0, 1, \ldots\}$ added $A$-faults
- $h$ unitary gain (for $A$) due to each $O$-fault
- $g$ unitary gain (for $A$) due to each $A$-fault
- $f$ unitary cost (for $A$) for adding each $A$-fault
- Attacker $A$'s loss function, depending on batch composition and decision by $D$

| | | **Final Batch Composition** | | |
|---|---|---|---|---|
| | | Acceptable | $O$-Fault | $A$-Fault |
| | | $x$ | $m - x$ | $y_1$ |
| **D's Decision** | Accept, $d_0$ | 0 | $-h$ | $f - g$ |
| | Reject, $d_1$ | 0 | 0 | $f$ |

# ADVERSARIAL BATCH ACCEPTANCE MODE

| | | Final Batch Composition | | |
|---|---|---|---|---|
| | | Acceptable | $O$-Fault | $A$-Fault |
| | | $x$ | $m-x$ | $y_1$ |
| **D's Decision** | Accept, $d_0$ | 0 | $-h$ | $f-g$ |
| | Reject, $d_1$ | 0 | 0 | $f$ |

- Attacker $A$'s losses associated to Defender $D$'s decisions when $A$ chooses $y_1$

$$l_A(d_0, y_1, x) = -h\,(m-x) + (f-g)\,y_1$$

$$l_A(d_1, y_1) = f\,y_1$$

# ADVERSARIAL BATCH ACCEPTANCE MODE

- Losses: $l_A(d_0, y_1, x) = -h\,(m - x) + (f - g)\,y_1$ and $l_A(d_1, y_1) = f\,y_1$
- Problem faced by $A$: choose $y_1$ to minimise expected loss for original batch size $m$

$$\begin{aligned}
\psi_A(y_1|m) &= p_A(d_0|m + y_1) \int \left( \sum_{x=0}^{m} p_A(x|m,\theta)\, l_A(d_0, y_1, x) \right) p_A(\theta)\, \mathrm{d}\theta \\
&\quad + (1 - p_A(d_0|m + y_1))\, l_A(d_1, y_1) \\
&= y_1\,(f - g\,p_A(d_0|m + y_1)) \\
&\quad - h\,p_A(d_0|m + y_1) \int \left( \sum_{x=0}^{m} p_A(x|m,\theta)\,(m - x) \right) p_A(\theta)\, \mathrm{d}\theta,
\end{aligned}$$

- $p_A(d_0|m + y_1)$ reflects $A$'s beliefs about $D$'s decision $d_0$ to accept the batch given that she knows the batch size is $n = m + y_1$

# ADVERSARIAL BATCH ACCEPTANCE MODE

- Defender does not know Attacker's probabilities and parameters of his loss function $\Rightarrow (F, G, H,\, P_A(d_0|n), P_A(\theta), P_A(x|m,\theta))$ random quantities

- Look for random optimal attack $Y_1^*(m)$ defined through

$$\arg\min_{y_1} \begin{cases} y_1\,(F - G\,P_A(d_0|m+y_1)) \\ -\,H\,P_A(d_0|m+y_1) \displaystyle\int \left( \sum_{x=0}^{m} P_A(x|m,\theta)\,(m-x) \right) P_A(\theta)\,\mathrm{d}\theta \end{cases}$$

- Draw from random quantities and get sample $\{Y_{1k}^*(m)\}_{k=1}^K$ of size $K$ from $Y_1^*(m)$

- Estimate $\widehat{p}_D(y_1|m) = P(y_1^*(m) = y_1) \approx \#\{Y_{1k}^*(m) = y_1\}/K$

  $\Rightarrow$ get the optimal amount of added faulty items (e.g. from the mode)

# ADVERSARIAL BATCH ACCEPTANCE MODE

Typical assumptions about Attacker's random utilities and probabilities

- Gains and costs uniformly distributed:
  - $F \sim \mathcal{U}(f_1, f_2)$
  - $G \sim \mathcal{U}(g_1, g_2)$
  - $H \sim \mathcal{U}(h_1, h_2)$
- $P_A(x|m, \theta)$ Binomial distribution $\mathcal{B}in(m, \theta)$ (i.e. not a random distribution)
- $P_A(\theta)$ from a Dirichlet process with Beta distribution $\mathcal{B}e(\alpha + s, \beta + r - s)$ as base parameter and concentration parameter $\rho$
- $P_A(d_0|n)$ modelled through a uniform distribution, although this might require further recursion if deeper strategic thinking is considered

# ADVERSARIAL BATCH ACCEPTANCE MODE

- Other two strategies:
  - $\mathcal{S}_2$. Attacker modifies few original items converting them into faulty ones
  - $\mathcal{S}_3$. Attacker modifies few original items converting them into faulty ones and adds some new ones
- Very similar approach: not presented here except for the Attacker $A$'s loss function, depending on batch composition and decision by $D$

# ADVERSARIAL BATCH ACCEPTANCE MODE

$\mathcal{S}_2$. Attacker modifies few original items converting them into faulty ones

- $h$ unitary gain (for $A$) due to each $O$-fault
- $g$ unitary gain (for $A$) due to each $A$-fault
- $e$ unitary cost (for $A$) for changing any item to make it faulty

| | | **Final Batch Composition** | | |
|---|---|---|---|---|
| | | Acceptable | $O$-Fault | $A$-Fault |
| | | $x - y_2^0$ | $m - x - y_2^1$ | $y_2$ |
| **D's Decision** | Accept, $d_0$ | 0 | $-h$ | $e - g$ |
| | Reject, $d_1$ | 0 | 0 | $e$ |

# ADVERSARIAL BATCH ACCEPTANCE MODE

$\mathcal{S}_3$. Attacker modifies few original items converting them into faulty ones and adds some new ones

- $h$ unitary gain (for $A$) due to each $O$-fault
- $g$ unitary gain (for $A$) due to each $A$-fault
- $e$ unitary cost (for $A$) for changing any item to make it faulty
- $f$ unitary cost (for $A$) for adding each $A$-fault

| | | Final Batch Composition | | | |
|---|---|---|---|---|---|
| | | Acceptable | $O$-Fault | $A$-Fault *Injected* | $A$-Fault *Modified* |
| | | $x - y_2^0$ | $m - x - y_2^1$ | $y_1$ | $y_2$ |
| **D's Decision** | Accept, $d_0$ | 0 | $-h$ | $f - g$ | $e - g$ |
| | Reject, $d_1$ | 0 | 0 | $f$ | $e$ |

# DISCUSSION

- New ARA approach to dealing with the AHT problem
- Symmetric losses and strong common knowledge assumptions typical of non-cooperative game theory have been avoided
- Multiple Attackers and/or multiple Defenders cases in the AHT problem are also of interest
  - need to differentiate when Attackers are completely independent or totally coordinated or are such that their attacks influence somehow each other
  - possibility of several Defenders, possibly cooperating but with different observations of the data flow
- New strategies, e.g. Attacker could add (apparently) acceptable items to confound the Defender
- Possible application in adversarial signal processing, such as in Electronic Warfare where pulse/signal environment is generally very complex with many different radars transmitting simultaneously and signals possibly jammed by hostile radars

# ACCEPTANCE SAMPLING

Work stemming from Lindley and Singpurwalla (1991)

- Manufacturer $M$ (she) is trying to sell a batch of items to a consumer $C$ (he) who may either accept ($\mathcal{A}$) or reject ($\mathcal{R}$) the batch provided by $M$
- $C$'s decision depends on the evidence provided by $M$ to $C$, based on a sample from an inspection that $M$ may perform
- The decision $M$ faces is whether to offer a sample to $C$ and, if so, the size of such sample
- Both $M$ and $C$ are assumed to be expected utility maximisers
- Lindley and Singpurwalla assume that $M$, who decides before $C$, knows $C$'s preferences and beliefs, as well as they share other relevant distributions, a too strong common knowledge assumption
- ARA allows us to overcome such issue (for Bernoulli acceptance sampling problem)
- Addressed also a life testing problem

# ACCEPTANCE SAMPLING: GAME THEORY

Sequential problem

- $M$ decides the sample size $n$ to offer to $C$ ($\Rightarrow C$ knows $n$)

- $C$ has available

  - $p_C(\theta)$, i.e., beliefs about the product quality $\theta$

  - $p_C(d|\theta, n)$, i.e., beliefs about the experiment result $d$ (number of defective items) given $\theta$ and decision $n$ of $M$

  - $u_C(c, \theta)$, i.e., utility function based on decision $c$: accept ($\mathcal{A}$) or reject ($\mathcal{R}$) the batch

# ACCEPTANCE SAMPLING: GAME THEORY

- $C$ computes for each $d$ and $n$
  - Posterior distribution $p_C(\theta|d,n) \propto p_C(\theta)p_C(d|\theta,n)$
  - Expected utility $\psi_C(d,n,c) = \int u_C(c,\theta)p_C(\theta|d,n)d\theta$
  - Optimal decision $c$, given $d$ and $n$:

$$c^*(d,n) = \operatorname*{arg\,max}_{c\in\{\mathcal{A},\mathcal{R}\}} \ \psi_C(d,n,c)$$

- All the above known by $M$ who switches to her problem

# ACCEPTANCE SAMPLING: GAME THEORY

$M$ knows $p_C(\theta|d,n)$, $\psi_C(d,n,c)$ and $c^*(d,n)$ for each $d$ and $n$

- $M$ has available
  - $p_M(\theta)$, i.e., beliefs about the product quality $\theta$
  - $p_M(d|\theta,n)$, i.e., beliefs about the experiment result $d$ (number of defective items) given $\theta$ and decision $n$ of $M$
  - $u_M(c,\theta)$, i.e., utility function based on decision $c$: accept ($\mathcal{A}$) or reject ($\mathcal{R}$) the batch

# ACCEPTANCE SAMPLING: GAME THEORY

- $M$ computes for each $d$ and $n$
  - $\psi_M(n,d,\theta) = u_M(c^*(d,n),n,\theta)$, i.e., utility based on $C$'s decision (known under the common knowledge assumption)
  - $\psi_M(n,\theta) = \int \psi_M(n,d,\theta)p_M(d|\theta,n)\ dd$, i.e., expected utility (w.r.t. $d$)
  - $\psi_M(n) = \int \psi_M(n,\theta)p_M(\theta)\,d\theta$, i.e., expected utility (w.r.t. $\theta$)
  - $n^* = \arg\max \psi_M(n)$, i.e. optimal decision by $M$

# ACCEPTANCE SAMPLING: ARA

- $p_M(\theta)$, $p_M(d|\theta, n)$ and $u_M(c, n, \theta)$ available as before
- Earlier $c^*(d, n)$ was known but now $p_M(c|d, n)$ is needed (and its computation requires thinking about $C$'s behaviour)
- $\Rightarrow$ Need to compute $\psi_M(n, d, \theta) = \sum_{c \in \{\mathcal{A}, \mathcal{R}\}} u_M(c, n, \theta) p_M(c|d, n)$ to get rid of $c$
- $p_C(\theta)$, $p_C(d|\theta, n)$, and $u_C(c, \theta)$ unknown to $M$ (no common knowledge)
- $\Rightarrow$ random utilities and probabilities generated from $F = (U_C(c, \theta), P_C(\theta), P_C(d|\theta, n))$
- Computation of random functional $\Psi^*_C(d, n, c) = \int U_C(c, \theta) P_C(\theta) P_C(d|\theta, n) d\theta$
- Computation of the random optimal alternative, given $d$ and $n$:

$$C^*(d, n) = \underset{c \in \{\mathcal{A}, \mathcal{R}\}}{\arg\max} \; \Psi^*_C(d, n, c)$$

- $\Rightarrow$ empirical distribution of $C^*(d, n)$ to estimate $p_M(c|d, n)$

# BERNOULLI ACCEPTANCE SAMPLING

The manufacturer's viewpoint

- Sample of size $n$ offered by manufacturer possibly defective with probability $\theta$
- Sampling model binomial for $d$ defective items with $p_M(d|\theta, n) \sim Bin(n, \theta)$
- $\theta$ with a beta distribution $p_M(\theta) \sim \beta e(\beta_1, \beta_2)$
- Utility function $u_M(c, n, \theta)$ as in Lindley and Singpurwalla (1991):
  - $u_M(\mathcal{A}, n, \theta) = b_1 + b_2\theta + b_4 n,$
  - $u_M(\mathcal{R}, n, \theta) = b_3 + b_4 n$
  - $b_4$ unit cost of providing each sample unit
  - $b_2$ penalty for defectiveness; the higher $\theta$, the worse the corresponding cost
  - $b_1 > b_3$: preference for accepted items rather than rejected
  - $b_3 > b_1 + b_2$: preference for rejection rather than acceptance of very low quality lot (for reputation)

# BERNOULLI ACCEPTANCE SAMPLING

Assumptions on $C$

- Same sampling model binomial for $d$ defective items with $p_M(d|\theta, n) \sim Bin(n, \theta)$
- Random distribution $P_C(\theta)$ given by
  - Beta distribution $p_c(\theta) \sim \beta e(\alpha_1, \alpha_2)$
  - Uniform distributions $\alpha_1 \sim \mathcal{U} \in [a_{11}, a_{12}]$, and $\alpha_2 \sim \mathcal{U} \in [a_{21}, a_{22}]$
  - Compare with Lindley and Singpurwalla (1991) who considered $p_c(\theta) \sim \beta e(\alpha_1, \alpha_2)$, with known $\alpha_1$ and $\alpha_2$
- Random utility $U_C(c, \theta)$, similar to Lindley and Singpurwalla (1991):
  - $u_C(\mathcal{A}, \theta) = a_1 + a_2\theta$,
  - $u_C(\mathcal{R}, \theta) = a_3$,
  - where $a_1 > a_3 > a_1 + a_2$ and $a_2 < 0$

# BERNOULLI ACCEPTANCE SAMPLING

An example (values of the parameters omitted)

| | | $n = 0$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| $\widehat{p_M}(\mathcal{A}\|d, n)$ | $d = 0$ | x | 0.4 | 0.49 | 0.55 | 0.61 | 0.65 | 0.68 | 0.71 | |
| | $d = 1$ | x | 0.22 | 0.34 | 0.42 | 0.49 | 0.54 | 0.58 | 0.62 | |
| | $d = 2$ | x | x | 0.19 | 0.29 | 0.37 | 0.44 | 0.49 | 0.53 | |
| | $d = 3$ | x | x | x | 0.16 | 0.26 | 0.33 | 0.4 | 0.45 | |
| | ... | x | x | x | x | 0.14 | 0.23 | 0.3 | 0.36 | |

Acceptance probabilities for various manufacturer decisions and experimental results

# BERNOULLI ACCEPTANCE SAMPLING

| | $n = 1$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\psi_M(n)$ | 4.25 | 4.325 | 4.374 | 4.408 | 4.43 | 4.444 |
| ... | 7 | 8 | 9 | 10 | 11 | 12 |
| $\psi_M(n)$ | 4.453 | 4.456 | 4.457 | 4.456 | 4.451 | 4.444 |

Expected utilities of various manufacturer decisions ($n = 9$ optimal decision)

# CLASSIFICATION

- Classification: widely used supervised learning method, applied, e.g., in computer vision, genomics, credit scoring and spam detection
- Currently, a major research area in Statistics and Machine Learning (ML)
- Most efforts focused on obtaining more accurate algorithms
- Less attention for a relevant aspect: presence of adversaries manipulating data to deceive the classifier in order to obtain a benefit (e.g. credentials of bank account)
- Example: Fraud detection
  - ML algorithms developed for detection $\Rightarrow$ fraudsters learn how to evade them
  - Detection more likely for huge transactions $\Rightarrow$ smaller ones more frequently
- No common knowledge $\Rightarrow$ Adversarial Risk Analysis (ARA)

# ADVERSARIAL HYPOTHESIS TESTING (AHT)

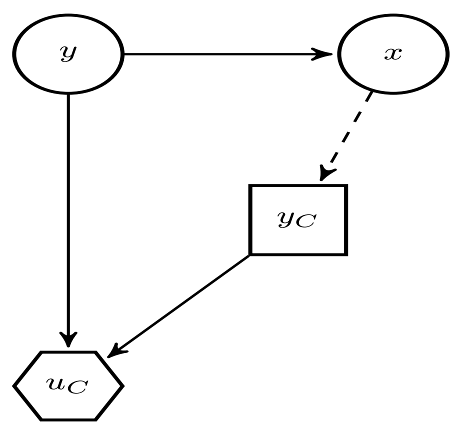
- Use concepts from Adversarial Risk Analysis (ARA)
- Agent (Defender D) needs to ascertain which of several hypotheses holds, based on observations from a source
- Another agent (Attacker A) alters the observations to induce the Defender to make a wrong decision (and get a benefit)
- AHT problem studied from the Defender’s perspective
- Lack of common knowledge about decision strategies
- Defender needs to forecast the Attacker’s decision, simulating from the guess about Attacker’s decision making problem (based on Defender’s decision problem)

# ADVERSARIAL HYPOTHESIS TESTING

- Test of two simple hypotheses: $\Theta = \{\theta_0, \theta_1\}$
- Observation $x$ generated according to a model depending on $\theta$
- $x$ altered to $y$ by A's action $a$
- $y$ observed by D $\Rightarrow$ $D$'s decision $d$ on $\theta$ based on $y$, without observing $x$
- Depending on $d$ and actual $\theta$ $\Rightarrow$ losses (utilities) for both agents
- Efforts by A in minimising the loss
- Support for D in choosing $\theta$ to minimise the loss

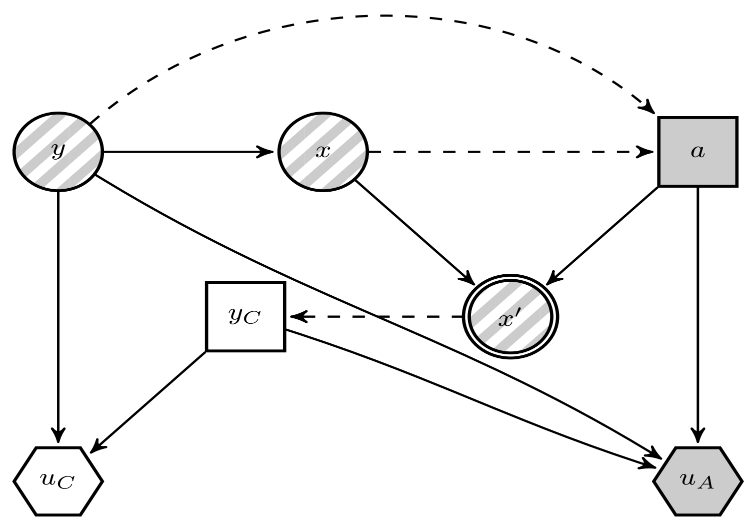
# BINARY CLASSIFICATION

- Classifier $C$ receives two types of objects: malicious ($y = +$) or innocent ($y = -$)
- Objects have features $x$ whose distribution depends on their type $y$
- Classification problems broken down into two separate stages:
  - inference about $p_C(y|x)$, $C$'s beliefs about type given features
  - decision about class assignment $y_C$, based on $p_C(y|x)$ and utility $u_C(y_C, y)$
- Node: decision (square), uncertainty (circle), deterministic (double), utility (hex.)
- Arrow: conditional relation (solid), information available at decision time (dashed)



# ADVERSARIAL CLASSIFICATION

- Adversary $A$ chooses attack $a$ s.t. actual $x \to x' = a(x)$ observed by $C$
- $A$ attacks only for malicious instances ($y = +$)
- Nodes in bi-agent influence diagram: grey ($A$), white ($C$), striped (both $A$ and $C$)
- Decisions: attack $a$ by $A$ and classification $y_C$ by $C$
- Utilities: $u_C(y_C, y)$ for $C$ and $u_A(y_C, y, a)$ for $A$



# CLASSIFIER PROBLEM

$$\begin{aligned}
\text{Find class } c(x') &= \arg\max_{y_C} \sum_{y \in \{+,-\}} u_C(y_C, y) p_C(y|x') \\
\text{(divide by } p_C(y)) &= \arg\max_{y_C} \Big[ u_C(y_C, -) p_C(x'|-) p_C(-) \\
&+ u_C(y_C, +) p_C(+) \sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x, +) p_C(x|+) \Big]
\end{aligned}$$

- Expected utility maximisation
- $\mathcal{A}(x)$: set of possible attacks for actual $x$
- $\mathcal{X}' = \{x : a(x) = x' \text{ for some } a \in \mathcal{A}(x)\}$: $x$'s potentially leading to observed $x'$
- $p_C(y)$: beliefs about the class distribution
- $p_C(x|y)$: beliefs about feature distribution given the class (under no attacks)
- $u_C(y_C, y)$: utility in classifying $y_C$ with actual $y$
- $p_C(a|x, y)$: beliefs about $A$'s action, given $x$ and $y$ (Think of $A$'s behaviour!)

# ATTACKER PROBLEM

- Find optimal attack

$$
\begin{aligned}
a^*(x,y) &= \arg\max_a \int \Big[ u_A(+,+,a)\ p + u_A(-,+,a)\ (1-p) \Big] f_A(p|a(x))dp \\
&= \arg\max_a \left[ u_A(+,+,a) - u_A(-,+,a) \right] p^A_{a(x)} + u_A(-,+,a)
\end{aligned}
$$

- $A$: modify $x$ so that $C$ classifies malicious instances as innocent ($A$'s maximum expected utility)
- $A$: modify only malicious instances, i.e. $y = +$, and not innocent, i.e. $y = -$
- $C$'s decision: uncertain for $A$
- $u_A(y_C, y, a)$: utility for $A$ when $C$ says $y_C$, actual label is $y$ and the attack is $a$
- $p_A(c(x')|x')$: $A$'s beliefs about the classification result when $C$ observes $x'$
- $p = p_A(c(a(x)) = +|a(x))$: $A$'s beliefs about $C$ classifying as malicious after observing $x' = a(x)$
- Uncertainty on $p$ modelled via density $f_A(p|a(x))$ with expectation $p^A_{a(x)}$.

# CLASSIFIER PROBLEM

- Find $a^*(x,y) = \text{arg max}_a \, [u_A(+,+,a) - u_A(-,+,a)] \, p^A_{a(x)} + u_A(-,+,a)$
- $C$ does not know $A$'s utilities $u_A$ and probabilities $p^A_{a(x)}$
- $C$'s uncertainty modelled through random utility $U_A$ and random expectation $P^A_{a(x)}$
- Solve for the random optimal attack, optimising the random expected utility
$$A^*(x,+) = \text{arg max}_a \left( [U_A(+,+,a) - U_A(-,+,a)] \, P^A_{a(x)} + U_A(-,+,a) \right)$$
- $\Rightarrow p_C(a_{x \to x'} | x, +) = Pr(A^*(x,+) = a_{x \to x'})$, assuming a discrete set of attacks
- Approximation through simulation of $K$ samples $\left(U^k_A(y_C,+,a), P^{A,k}_{a(x)}\right)$ from random utilities and probabilities
$$\Rightarrow A^*_k(x,+) = \text{arg max}_a \left( \left[U^k_A(+,+,a) - U^k_A(-,+,a)\right] P^{A,k}_{a(x)} + U^k_A(-,+,a) \right)$$
- Estimation: $\widehat{p_C}(a_{x \to x'} \,|\, x, +) = \#\{A^*_k(x,+) = a_{x \to x'}\}/K$

# RANDOM UTILITY

- Random utility $U_A(y_C, +, a)$ includes two components
  - $A$'s gain from $C$'s decision
  - random cost $B$ of implementing an attack
- $Y_{y_C y}$: gain when $C$ decides $y_C$ with $y$ actual label
- $-Y_{++} \sim Ga(\alpha_1, \beta_1)$ with expected gain $\alpha_1/\beta_1 = -d$ for $A$ and variance $\alpha_1/\beta_1^2$
- $Y_{-+} \sim Ga(\alpha_2, \beta_2)$ with expected gain $\alpha_2/\beta_2 = e$ for $A$, and variance $\alpha_2/\beta_2^2$
- $Y_{+-} = Y_{--} = \delta_0$, Dirac at 0: no gain for $A$ from innocent instances
- $\Rightarrow A$'s gain $(Y_{y_C y} - B)$
- If $A$ risk prone $\Rightarrow U_A(y_C, y, a) = \exp(\rho\,(Y_{y_C y} - B))$ with random risk proneness coefficient $\rho \sim U[a_1, a_2]$, $a_1 > 0$

# RANDOM PROBABILITY

- $P^A_{a(x)}$, $A$'s (random) expected probability that $C$ classifies as malicious for $x' = a(x)$
- $C$ guesses $A$'s beliefs about $C$'s classification when observing $x' \Rightarrow$ delicate
- Hierarchy of decisions: $A$ should know what $C$ does when knowing what $A$ does ...
- Probabilities to be specified at each stage until no more available information
$\Rightarrow$ non-informative distribution at that stage
- Heuristic at first stage based on $Pr_C(c(x') = +|x') = r$ ($C$ classifies as malicious observing $x'$), with some uncertainty around it
$\Rightarrow P^A_{a(x)} \sim \beta e(\delta_1, \delta_2)$, with mean $\delta_1/(\delta_1 + \delta_2) = r$ and adequate variance
- In general, given observed $x'$, consider all instances leading to it
  - $p_1$: proportion of instances originally malicious
  - $p_2$: proportion of instance originally innocent
  - $\Rightarrow r = p_1/(p_1 + p_2)$

# SPAM DETECTION

- $m$ emails as *bag-of-words*: binary features about presence (1) or not (0) of $n$ words
- Label indicates whether the message is spam $(+)$ or not $(-)$
- Email as $n$-dimensional vector $x = (x_1, x_2, .., x_n)$ of 0's or 1's, with label $y$
- Only word insertion attacks $\Rightarrow$ 0's replaced by 1's
- Interest in insertion of one word at most
- $I(x)$: set of indices s.t. $x_i = 0$ in $x \Rightarrow \mathcal{A}(x) = \{a_0, a_i; \forall i \in I(x)\}$ set of possible attacks with identity $a_0$ and $a_i$ transforming $i$-th 0 into 1
- $J(x')$: set of indices with value 1 in $x'$ received by $C \Rightarrow \mathcal{X}' = \{x', x'_j; \forall j \in J(x')\}$ and $x'_j$ message potentially leading to $x'$, with $j$-th 1 in $x'$ replaced with 0

# SPAM DETECTION

- $u_C(y_C, y)$ standard
- $p_C(y)$ and $p_C(x|y)$ standard if considering only exploratory attacks and using generative classifier to estimate them
- Strategic component for $p_C(a_{x \to x'}|x, y)$ and use of ARA to approximate it
- Adversary's random utilities obtained as before
- Beta distribution for $P^A_{a(x)}$ with adequate variance and mean $r_a$
  - $q_0 = p_C(x'|-)p_C(-)$: original label $-$ left unchanged by $A$
  - $q_j = p_C(x'_j|+)p_C(+), \forall j \in J(x')$: original label $+$ changed by $A$
  - $q_{n+1} = p_C(x'|+)p_C(+)$: original label $+$ left unchanged by $A$
  - $r_a = \dfrac{\sum_{i \in J[a(x)]} q_i + q_{n+1}}{q_0 + \sum_{i \in J[a(x)]} q_i + q_{n+1}}$

# SPAM DETECTION

- Spambase Data Set from UCI Machine Learning repository
  - 4601 emails, out of which 1813 are spam
  - 54 relevant words for each email $\Rightarrow$ 54 dimensional vector $x$ of 0's and 1's
  - data randomly split into training (75%) and test ($25\%$) sets, with 100 repetitions
- Training not affected by attacks $\Rightarrow \widehat{p}_C(y)$ and $\widehat{p}_C(x|y)$ from Naive Bayes classifier
- Simulations (sample size 1000) with 4 utilities for $C$ and different variances for random expected probability $P^A_{a(x)}$ (increasing percentages $k$ of maximum value)
- Comparison between ACRA and Naive Bayes: accuracy, utility, false positive (FPR) and false negative rates
- ACRA more robust w.r.t. attacks, identifying more attacked spam emails, even for larger $k$, i.e. variance, worsening the performance
- ACRA $\Rightarrow$ lower FPR, i.e. less non-spam are rejected as spam (more important than accepting spam)

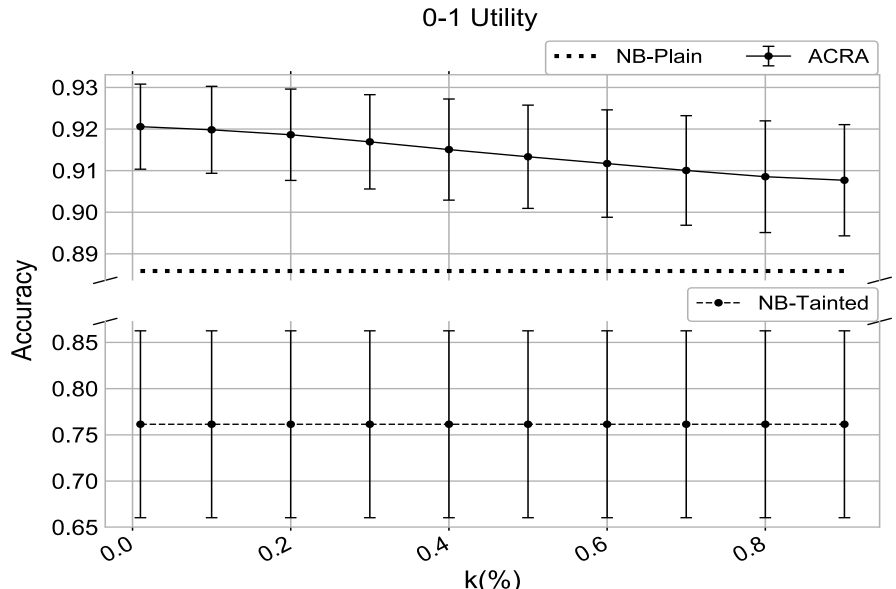
# SPAM DETECTION

- Checking utility robustness through 4 utilities for C:
  - $0/1$ Utility $\Rightarrow$ 1 if correctly classified and 0 o.w.
  - Three utilities taking values
    - 1 if correctly classified
    - -1 for spam classified as legit
    - $-2/-5/-10$ for legit classified as spam
- Random utilities for A ($m$=mean, $v$=variance)
  - $-U_A(+,+,a) \sim Ga(2500, 0.002) \Rightarrow m = 5, v = 0.01$
  - $U_A(-,+,a) \sim Ga(2500, 0.002) \Rightarrow m = 5, v = 0.01$
  - $U_A(-,-,a) = U_A(+,-,a) = \delta_0$
- Random cost $B = d(a) \cdot \alpha$, with $d(a) = \#$ word changes and $\alpha \sim U[0.4, 0.6]$
- Random risk proneness coefficient $\rho \sim U[0.4, 0.6]$

# SPAM DETECTION

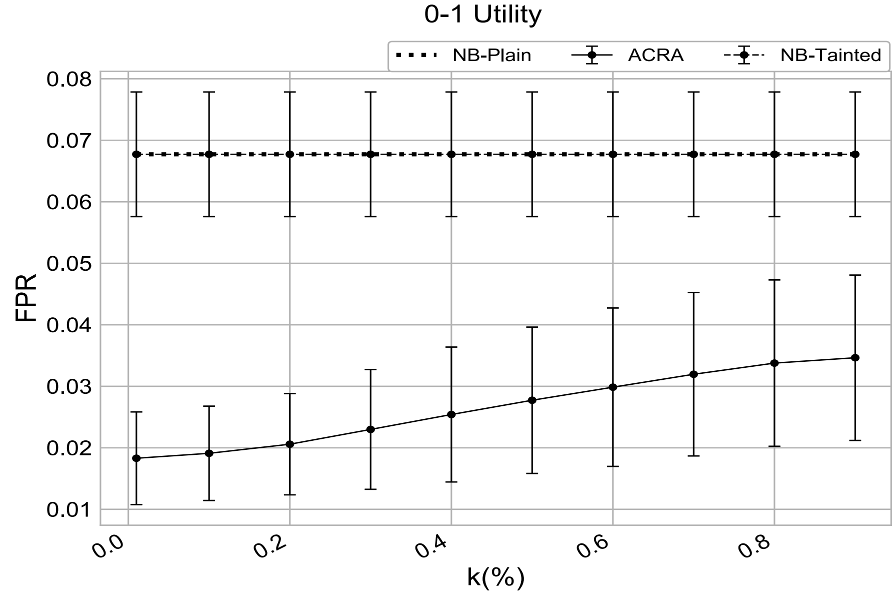
- Beta distribution for $P^A_{a(x)}$ with mean $r = Pr_C(c(a(x)) = +|a(x))$
  - Concave to avoid malicious $a(x)$ concentrated around 0 or 1
  - $\Rightarrow$ variance $\leq \Delta = \min\left\{[r^2(1-r)]/(1+r), [r(1-r)^2]/(2-r)\right\}$
  - Adjustable variance at $k\Delta$ with $k \in \{0.01, 0.1, 0.2, \cdots, 0.9\}$
- $K = 1000$ Monte Carlo sample size

# SPAM DETECTION




- Starting problem for $C$: find $c(x') = \arg\max_{y_C} \sum_{y \in \{+,-\}} u_C(y_C, y) p_C(y|x')$
- $0/1$ utility function, i.e. 1 for correctly classified instance and 0 otherwise
- Naive Bayes: NB-Plain for original data and NB-Tainted for attacked data
- $k$: percentage of maximum variance for $P^A_{a(x)}$

# SPAM DETECTION




- Naive Bayes: NB-Plain and NB-Tainted behave similarly since $A$ is not modifying innocent instances
- Increasing $k$ (and variance for $P^A_{a(x)}$) $\Rightarrow$ increases FPR
- Reducing FPR crucial in spam detection, as filtering out a non-spam is worse than letting spam reach the user

# DISCUSSION ABOUT ACRA

- So far ACRA tested with $A$'s distributions centered around the expected values of $C$'s, but it proves quite robust even when moving away
- Changing all words in the spam detection problem $\Rightarrow 2^n$ possible attacks
  - Ad hoc procedure, e.g., changing only one word and from 0 to 1
  - Smaller sample size
  - Approximations, parallelisation
- Further extensions
  - From binary classification to multi-label (e.g. malware: trojan, adware, virus)
  - From exploratory to poisoning attacks, i.e. attacks also during training
  - Attacks not only on malicious instances but also on innocent ones
  - From generative classifiers ($P(X, Y)$) to discriminative ones ($P(Y|X = x)$)

# DISCRIMINATIVE CLASSIFIERS

- In the earlier approach (generative classifier) we supposed to know $p(y)$ and $p(x|y)$, e.g. from a classifier applied to the training set

- Here we suppose to know only $p(y|x)$ and address the problem of classifying an instance when $x'$ is observed $\Rightarrow$ solve $\arg\max_{y_C} \psi(y_C)$ where

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

- $p(y|x)$ is based on untainted $x$

- $\mathcal{X}_{x'}$, the set of reasonable instances $x$ leading to $x'$ if attacked

- Optimisation solved via Monte Carlo using sample $\{x_n\}_{n=1}^{N}$ from $p(x|x')$ but ...

- ... there is a problem: we do not know $p(x|x')$ and we have to estimate it

# AB-ACRA

- Suppose $p(x)$ unknown and $p(x'|x)$ known as result of strategic thinking, as before, about the possible attacks
- Efficient approach to sample from $p(x|x')$ making use of samples from $p(x'|x)$
- Sample from $p(x|x') \propto p(x'|x)p(x)$ for $x$ and $x'$ discrete
  - Proposal $\tilde{x}$ from transition distribution $q(x \to \tilde{x})$
  - Sampled $\tilde{x}' \sim p(X'|X = \tilde{x})$
  - $\Rightarrow$ accept $\tilde{x}$ if $\tilde{x}' = x'$ with probability $\alpha = \min\left\{1, \frac{p(\tilde{x})q(\tilde{x} \to x_i)}{p(x_i)q(x_i \to \tilde{x})}\right\}$
  - Very slow convergence
- Sample from $p(x|x')$ for $x$ and $x'$ continuous
  - $\tilde{x}$ and $\tilde{x}'$ generated as above
  - Based on Approximate Bayesian Computation (ABC) techniques, accept $\tilde{x}$ if $\phi(\tilde{x}', x') < \epsilon$ for a given distance $\phi$ and tolerance $\epsilon$
  - For high dimensions, use summary statistics $s$ to accept $\tilde{x}$ if $\phi(s(\tilde{x}'), s(x')) < \epsilon$

# CONCLUSIONS ABOUT ACRA

- Here more emphasis on modelling and conceptual aspects whereas the papers contains many details about algorithmic ones and comparisons with classical classifiers

- Like in ABC, the choice of summary statistics in AB-ACRA might be critical

- AB-ACRA and ACRA become computationally expensive for large scale problems $\Rightarrow$ differentiable classifiers as an alternative

- Adaptive attackers can be dealt with changing random probability and random utility accordingly

- Here we have considered attacks to i.i.d. sequences but data could come, say, from an autoregressive model

# ADVERSARIAL SOFTWARE TESTING

- Software subject to (possibly expensive and dangerous) failures in programming or system design
- $\Rightarrow$ software must undergo rigorous testing, both during development and operation, to verify its reliability
- Optimal policies for software release $\Rightarrow$ important issue in software engineering
- Challenges due to several, often uncertain, complicating factors
- Endogenous factors
  - number of bugs in the software
  - skill in detecting bugs
- Exogenous factors
  - release decisions made by competitors
  - eventual purchasing decision by software buyers

# ADVERSARIAL SOFTWARE TESTING

- Monetary aspects
  - costs related to time on test
  - costs related to bugs discovering and their fixing during testing
  - costs related to bugs discovering and their fixing after the release
  - monetary gain for the software sale
- Reputational aspects
- Early software release $\Rightarrow$ larger commercial advantage over competitors
- Less intensely tested software $\Rightarrow$ possible lower quality $\Rightarrow$ potential advantage to competitors

# ADVERSARIAL SOFTWARE TESTING

- Singpurwalla and Wilson (2012): Review of software reliability and testing
- Anand, Singh, Das (2015): evaluation of two types (simple and serious) failures in successive versions of a software, during testing and operational phases
- Wilson and O'Riordain (2018): optimal release policy of new versions of Mozilla Firefox based on bug detection data
- Saraf and Iqbal (2019): software reliability model based on NHPP, performing fault detection, observation and correction in two stages and multiple versions
- Mishra, Kapur, Srivastava (2018): reliability growth of software over multiple versions
- Kenett, Ruggeri, Faltin (2018): thorough review of analytic methods in systems and software testing
- Ay, Landon, Ruggeri, Soyer (2022): software testing with possible introduction of bugs

# ADVERSARIAL SOFTWARE TESTING

- Ruggeri, Soyer (2018): overview of games and decision models for software testing
- Forman, Singpurwalla (1977, 1979) and Okumoto, Goel (1979): introduction of stopping time models to support software release decisions
- Dalal, Mallows (1988): pioneer work on decision theoretic models for release
- Morali, Soyer (2003): sequential Bayesian decision theoretic setup for developing optimal stopping policies for software testing
- Zeephongsekul, Chiera (1995): first game theoretic approach looking for optimal release policies through Nash equilibrium
  - Dohi, Teraoka, Osaki (2000): different approach since previous solution restricted to particular case and computationally intractable
  - Saito, Dohi (2022): uncovered faults in the earlier two papers showing the existence of Nash equilibrium under some parametric conditions

# ADVERSARIAL SOFTWARE TESTING

- Overview of Zeephongsekul and Chiera (1995)
- First work to consider also actions and costs of a competitor
- Two competitors ($i = 1, 2$) produce software performing the same set of tasks and with life cycle length non exceeding $T$
- Competitor $i$, $i = 1, 2$, decides to release the software at any time $t$ in $[0, T]$ and sells the product with probability $A_i(t)$ to the only buyer (who buys from one competitor at most)
- $A_i(t)$, $i = 1, 2$, continuously differentiable, concave and s.t. $A_i(0) = A_i(T) = 0$ with a unique maximum at time $\eta_i$
  - Choice of $A_i(t)$ not only for mathematical convenience but also justified by actual behaviour
  - Success probability expected to be close to $0$ both at the beginning and the end of the life cycle $[0, T]$, because of initial poor reliability and final obsolescence, respectively

# ADVERSARIAL SOFTWARE TESTING

- Introduction of expected cost function $c_i(t)$ incurred by player $i$ in releasing the software at time $t$
- $c_i(t) = c_{1i}t + c_{2i}m(t) + c_{3i}\left(m(T) - m(t)\right)$
  - $c_{1i}$ cost of testing per unit time
  - $c_{2i}$ cost of removing a fault during testing
  - $c_{3i}$ cost of removing a fault during operation, with $c_{3i} > c_{2i}$ since fixing an error is more expensive after release than before it
  - $m(t)$ expected number of faults detected up to time $t$
  - increasing, concave and differentiable $m(t)$, with $m(0) = 0$
- $\Rightarrow c_i(t)$ convex function with minimum at $\gamma_i$ s.t. $\Rightarrow m_i^{'}(\gamma_i) = \dfrac{c_{i1}}{(c_{3i} - c_{2i})}$
- $T$ is sufficiently large so that $\gamma_i < T$

# ADVERSARIAL SOFTWARE TESTING

- $p_i > 0$: selling price of the software produced by player $i$

- If player $1$ releases software at time $x$ and player $2$ at time $y \Rightarrow M_i(x, y)$ is the expected unit profit to player $i$, with

$$M_1(x,y) = \begin{cases} p_1 A_1(x) - c_1(x) & 0 \leq x < y \leq T \\ p_1(1 - A_2(y))A_1(x) - c_1(x) & 0 \leq y < x \leq T \end{cases}$$

- $M_2(x, y)$ can be described similarly and $M_1(x, y) \neq M_2(x, y)$ in general

- $\Rightarrow$ optimal release policies among Nash equilibrium points in this non-zero sum game (with concerns about the results as mentioned earlier)

- The paper, and all game theoretic work in the field, entails common knowledge assumptions, debatable in competitive business settings as in software development

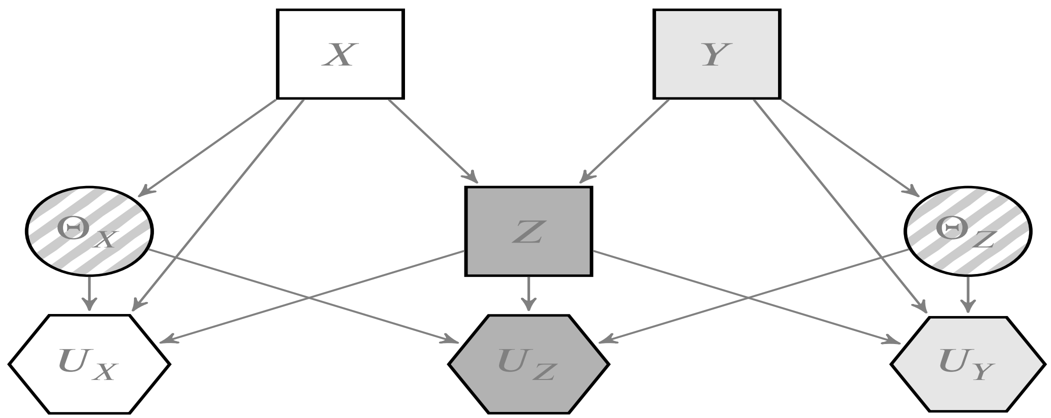
- $\Rightarrow$ Adversarial Risk Analysis $\Rightarrow$ Adversarial Software Testing

# ADVERSARIAL SOFTWARE TESTING

- Guevara, Pierce, Rios Insua, Ruggeri, Soyer (submitted)
- Support for producer $X$ against competitor $Y$, trying both to sell software to buyer $Z$ (purchasing from one producer at most)
- $X$ can release the software at any time $x \in [0, T]$
- In absence of competitors, $X$ would succeed in selling the product at the price $p_X$ with probability $A_X(x)$, with $A_X(0) = A_X(T) = 0$ (less restrictive than before)
- $Y$ releases at time $y \in [0, T]$ independently, succeeding to sell at fixed price $p_Y$ with probability $A_Y(y)$, with similar properties as $A_X$
- Consider a stochastic number $N_X(t)$ of faults found until time $t$, instead of the expected number $m_X(t) = E[N_X(t)]$
- $N_X(t)$ NHPP with intensity $\lambda_X(t)$ and mean value function $m_X(t) = \int_0^t \lambda_X(u) du$
- Similar definitions apply to $Y$

# ADVERSARIAL SOFTWARE TESTING

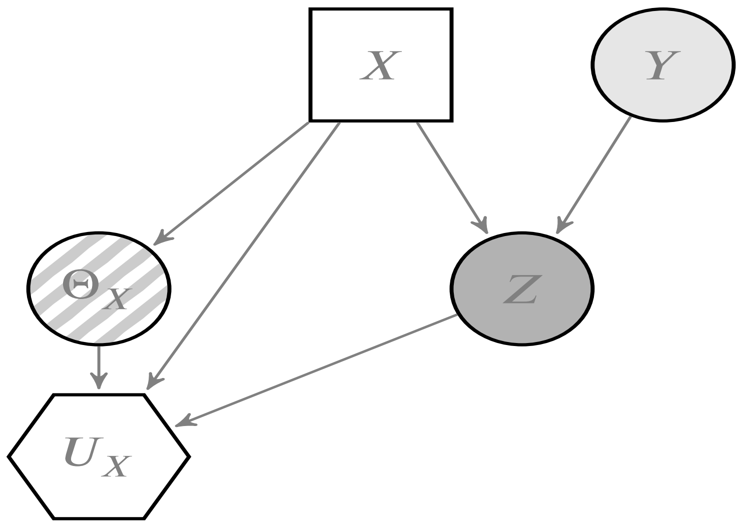
Tri-agent influence diagram representing the basic problem




- Global perspective
- Different colours for different agents
- Square nodes: Decisions by producers ($X$ and $Y$) and buyer ($Z$)
- Circle nodes: Uncertain features of $X$ ($\Theta_X$) and $Y$ ($\Theta_Y$), like number of bugs
- Hexagonal nodes: Utilities $U_X, U_Y, U_Z$ for $X, Y, Z$

# ADVERSARIAL SOFTWARE TESTING

Tri-agent influence diagram representing the basic problem




- Perspective from producer $X$, the one we are taking in the work
- $Y$'s decision now as a circle since it is uncertain for $X$

# ADVERSARIAL SOFTWARE TESTING

- $c_X(t) = c_{1X}t + c_{2X}N_X(t) + c_{3X}\left[N_X(T) - N_X(t)\right]$
  - $c_{1i}$ cost of testing per unit time
  - $c_{2i}$ cost of removing a fault during testing
  - $c_{3i} > c_{2i}$ cost of removing a fault during operation
- We assume that no new bugs are introduced during the debugging phase
- We assume that fault arrivals can be described by the same process during debugging and operational phase after the software has been released
- There are other assumptions leading to further developments, e.g., price fixed in advance, only two producers, only one buyer, fixed purchase probability

# ADVERSARIAL SOFTWARE TESTING

- $X$ and $Y$ release their software at times $x$ and $y$, respectively ($x \neq y$ a.s.)
- $X$ stops testing if the buyer does not purchase its software, either because it rejects the product or because it has already bought it from $Y$
- $g_X(x, y)$ (random) gain of producer $X$ given such release times
- Start with $x < y$ and rename $g_X$ as $g_{X1}$
- $\Rightarrow g_{X1}(x, y) = A_X(x)\,[p_X - c_X(x)] - [1 - A_X(x)]\,[c_{1X}\,x + c_{2X}N_X(x)]$
- First term: expected gain if $Z$ buys $X$'s software given by purchase probability at time $x$ times the difference between selling price and costs due to debugging until $x$ and fault removals after the release up to time $T$
- Second term: expected loss due to refusal by $Z$ and costs incurred until release time
- Note that $g_{X1}(x, y)$ does not depend on $y$

# ADVERSARIAL SOFTWARE TESTING

- Similarly, $Y$'s gain, for $y < x$, not dependent on $x$:
- $g_{Y1}(x, y) = A_Y(y)\,[p_Y - c_Y(y)] - [1 - A_Y(y)]\,[c_{1Y}\,y + c_{2Y}N_Y(y)]$
- When $x > y$, the $X$'s gain is renamed as $g_{X2}$

$$\begin{aligned} g_{X2}(x, y) \quad = \quad & -A_Y(y)\,[c_{1X}y + c_{2X}N_X(y)] + [1 - A_Y(y)]\,\{A_X(x)\,[p_X - c_X(x)] - \\ & - [1 - A_X(x)]\,[c_{1X}x + c_{2X}N_X(x)]\} \end{aligned}$$

- First term: $Z$ buys $Y$'s software and $X$ stops debugging its own
- Second and third term: like earlier, but after $Z$'s refusal of buying $Y$'s software
- Similar result for $Y$ when $y > x$

# ADVERSARIAL SOFTWARE TESTING

- Assuming risk neutrality $\Rightarrow$ expected gain $h_X(x, y)$ replacing $N_X(t)$ with its expectation, like for $x < y$

$$\begin{aligned} h_{X1}(x, y) &= A_X(x) \left[p_X - (c_{1X}x + c_{2X}m_X(x) + c_{3X}\left[m_X(T) - m_X(x)\right])\right] \\ &\quad - \left[1 - A_X(x)\right]\left[c_{1X}x + c_{2X}m_X(x)\right] \end{aligned}$$

- As an anticipation of what is next, $X$ can also consider $A_Y(y)$ as random and compute its expectation when $x > y$

$$h_{X2}(x, y) = -E(A_Y(y))[c_{1X}y + c_{2X}m_X(y)] + (1 - E(A_Y(y)))\times$$

$$\times[[A_X(x)[p_X - (c_{1X}x + c_{2X}m_X(x) + c_{3X}[m_X(T) - m_X(x)]] - [1 - A_X(x)]\times$$

$$\times[c_{1X}x + c_{2X}m_X(x)]]]$$

- Similar results apply to $Y$

# ADVERSARIAL SOFTWARE TESTING

- $\pi_Y^X(y)$: density modelling $X$'s beliefs about $Y$'s release decision being time $y$
- Expected gain associated with release decision $x$
  $M_X(x) = \int h_X(x,y)\pi_Y^X(y)dy = \int_0^x h_{X2}(x,y)\pi_Y^X(y)dy + \int_x^T h_{X1}(x,y)\pi_Y^X(y)dy$
- Optimal release time for $X$: $x^* = \arg\max_{0\leq x\leq T} M_X(x)$
- Above arguments slightly modified in absence of risk neutrality, i.e., when considering a utility function $u_X$

$$g_{X1}(x,y) = A_X(x)\times u_X(p_X - c_X(x)) + [1 - A_X(x)]\times u_X(-(c_{1X}(x) + c_{2X}N_X(x)))$$

$$\begin{aligned} g_{X2}(x,y) &= A_Y(y)\times u_X(-[c_{1X}y + c_{2X}N_X(y)]) + [1 - A_Y(y)]\times \\ &\times \{A_X(x)u_X([p_X - c_X(x)]) + [1 - A_X(x)]\, u_X(-[c_{1X}x + c_{2X}N_X(x)])\} \end{aligned}$$

# ADVERSARIAL SOFTWARE TESTING

- All the elements introduced above are standard in the decision analysis and software reliability literature and practice, except for those entailing strategic thinking:
  - $A_Y(y)$ (purchase probability of $Y$'s software)
  - $\pi_Y^X(y)$ ($X$'s beliefs about $Y$ releasing its product at time $y$)
- Need for procedures to facilitate their assessment, starting with $\pi_Y^X(y)$
- Look at $Y$'s perspective on product release
- Remember that $Y$ has a cost function $c_Y(t)$ and a purchase probability function $A_Y(t)$ for a fixed price $p_Y$, with similar properties and definitions than those of $X$
- Presenting now an approach to obtain an estimate $\widehat{\pi}_Y^X(t)$ of $\pi_Y^X(t)$ reflecting upon the optimisation problem faced by $Y$

# ADVERSARIAL SOFTWARE TESTING

- Suppose $X$ has complete knowledge about $Y$'s behaviour, i.e., $c_{1Y}, c_{2Y}, c_{3Y}$, $p_Y$, $\lambda_Y(t)$, $A_Y(t)$ and $\pi_X^Y(t)$ (which models $Y$'s beliefs about $X$'s release time)

- $\Rightarrow X$ could guess $Y$'s actual optimal release time $y^*$, using the previous computations by interchanging $X$ and $Y$

- But we have uncertainty about $Y$'s elements so that we
  - model such uncertainty through probability measures $\Pi_X^Y(t)$, $C_{1Y}$, $C_{2Y}$, $C_{3Y}$, $P_Y$, $\mathcal{A}_Y$ and $\mathcal{N}_Y(t)$ over the space of suitable densities $\pi_X^Y(t)$, constants $c_{1Y}$, $c_{2Y}$, $c_{3Y}$, $p_Y$, functions $A_Y$ and processes $N_Y(t)$, respectively
  - make a sufficiently large number of draws from these components, compute the corresponding optimal release time $y^*$ for each draw, and estimate an empirical distribution over $y^*$, which will be considered as the estimate $\widehat{\pi}_Y^X(y)$
  - $\Rightarrow X$ will be able to compute its optimal release time $x^*$

# ADVERSARIAL SOFTWARE TESTING

- The random ingredients could be specified gathering all information available and modelling with standard expert judgement

- Here we consider several heuristics based on adding some uncertainty to the judgements concerning $X$

- $Y$'s random beliefs about $X$'s decision $\Pi_X^Y(t)$
  - Transform the time interval $[0, T]$ into the unit interval via the transformation $t \to t/T$, $0 \leq t \leq T$
  - Consider suitable densities $\pi_X^Y(t)$ in the space of all beta densities over $[0, 1]$ or a proper subset, if $X$ feels capable of adding some constraints about their parameters, e.g. by fixing lower and/or upper bounds over mean and/or variance of the beta distributions
  - Randomly generate densities from such class, e.g., drawing a uniform distribution over both parameters of the beta distribution or its mean-variance pair

# ADVERSARIAL SOFTWARE TESTING

- $Y$'s random beliefs about $X$'s decision $\Pi_X^Y(t)$
  - Use distortion function as in Arias-Nicolas, Ruggeri and Suárez-Llorens (2016)
  - Start from an absolutely continuous (for simplicity) pdf $\pi_X(t)$ and its cdf $\Pi_X(t)$, expressing $X$'s opinion on $Y$'s release time and build a random space of cdf's $\pi_X^Y(t)$ around it
  - Consider distortion functions $h(t)$, i.e. non-decreasing functions such that $h : [0,1] \to [0,1], h(0) = 0, h(1) = 1$
  - Apply $h(\cdot)$ to $\Pi_X(t)$ and obtain random pdf's $\Pi_{hX}^Y(t) = h(\Pi_X(t))$ and cdf's $\pi_{hX}^Y(t) = h^{'}(\Pi_X(t))\pi_X(t)$
  - Consider a band around $\Pi_X(t)$ taking one convex and one concave distortion function to get, respectively, its lower and upper bounds
  - A useful choice for a distortion function is $h(t) = t^\alpha$, which is convex for $0 < \alpha < 1$ and concave for $\alpha > 1$
  - Randomness is induced by, say, considering that $\alpha$ follows a uniform distribution on a certain interval

# ADVERSARIAL SOFTWARE TESTING

- Uncertainty about $Y$'s costs
  - Model $X$'s uncertainty about $c_{1Y}, c_{2Y}$ and $c_{3Y}$ considering independent (Gaussian) distributions centered around the corresponding values $c_{1X}, c_{2X}, c_{3X}$
  - Alternatively, if $X$ can provide upper and lower bounds for $c_{1Y}$, $c_{2Y}$ and $d_Y = c_{3Y} - c_{2Y}$, then independent shifted beta distributions could be considered
  - The variances of those distributions will be determined by $X$ depending on the confidence about the chosen means
- Uncertainty about $Y$'s price $P_Y$
  - In absence of further information consider a (Gaussian) distribution with mean $p_X$ and variance $\sigma^2$ denoting the degree of uncertainty around $p_X$
- Uncertainty about $Y$'s purchase probability $A_Y(y)$
  - Transform $A_X(x) \rightarrow a\,[A_X(x)]^b$, with $a \in [0, 1]$ (decreasing effect) and $b \in [0, 1]$ (increasing effect)
  - $a$ and $b$ randomly generated to obtain values of $A_Y(y)$

# ADVERSARIAL SOFTWARE TESTING

- Uncertainty about $Y$'s fault discovery process $\mathcal{N}_Y(t)$
  - Suppose $X$ has chosen a functional form for $N_X(t)$ and estimated its parameters and obtained an estimate $\tilde{m}_X(t)$ for its mean value function
  - First alternative: generate values of the parameters of $\mathcal{N}_Y(t)$ from distributions centered around $X$'s estimated parameters (e.g. posterior distributions)
  - Second alternative: Bayesian non-parametric approach with mean value function as a random measure $M$, generated by a Gamma process, conjugate w.r.t. the Poisson process (Lo, 1982)
  - Gamma process centered around $\tilde{m}_X(t)$ so that at each interval $[t_0, t_1]$ the mean value function is generated by a Gamma distribution with mean $\tilde{m}_X(t_1) - \tilde{m}_X(t_0)$
  - The variance of the Gamma distribution could determine how close the fault discovery process $N_Y(t)$ is to $N_X(t)$
  - Further details can be found in Cavallo and Ruggeri (2001)

# ADVERSARIAL SOFTWARE TESTING: EXAMPLE

- Example based on Zeephongsekul and Chiera (1995)
- Life cycle length $T = 2000$ days
- Cost parameters: $c_{1X} = 0.5$, $c_{2X} = 1$, $c_{3X} = 5$
- Selling price $p_X = 5000$
- Purchase probability $A_X(t) = 0.0002t(10 - 0.005t)$
- Fault discovery process $N_X(t)$: NHPP with mean value function $m_X(t) = at^c$ (power law process) and MLEs of parameters given by $\widehat{a} = 0.256$ and $\widehat{c} = 0.837$, from Zeephongsekul and Chiera (1995) and based on data from Okumoto (1979)
- Cost function with utility function $u_X$ assumed to be the identity ($\Rightarrow$ Risk neutrality)

# ADVERSARIAL SOFTWARE TESTING: EXAMPLE

- Cost parameters follow distributions centered around the $c_X$ values:
  - $c_{1Y} \sim N(0.5, 0.02) = N(c_{1X}, 0.02)$
  - $c_{2Y} \sim N(1, 0.05) = N(c_{2X}, 0.05)$
  - $c_{3Y} \sim N(5, 0.5) = N(c_{3X}, 0.5)$
- Selling price $p_Y \sim N(5000, 250) = N(p_X, 250)$
- Random purchase probability $A_Y(t) \sim \tilde{d} A_X(t)^{\tilde{b}}$, with $\tilde{d} \sim U(0, 1)$ and $\tilde{b} \sim U(0, 1)$
- The random fault discovery process $N_Y(t)$ is a NHPP with random mean value function $m_Y(t) = \tilde{a} t^{\tilde{c}}$ with $\tilde{a} \sim N(0.256, 0.05)$ and $\tilde{c} \sim N(0.837, 0.05)$
- Beliefs of $Y$ over $X$'s release time $t$ given by $t/T \sim \beta e(\alpha, \alpha)$, with $\alpha \sim U(1, 3)$
- $Y$'s random cost function $c_Y(t) = c_{1Y} t + c_{2Y} N_Y(t) + c_{3Y} [N_Y(T) - N_Y(t)]$
- Deterministic utility function $U_Y$: identity $\Rightarrow$ risk neutrality

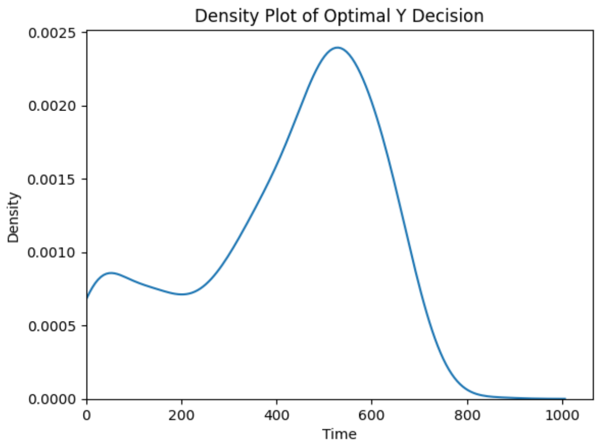
# ADVERSARIAL SOFTWARE TESTING: EXAMPLE

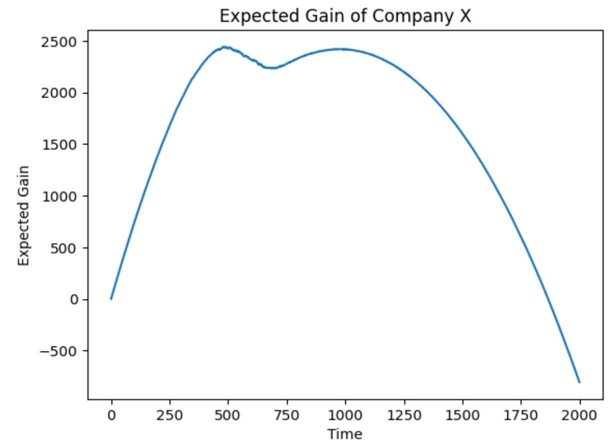
- Forecasting $Y$'s release decision
  - Maximise the objective function $M_Y(y) = \int h_Y(x,y)\pi_X^Y(x)dx$
  - For $i = 1, ..., K$
    - $*$ Sample $c_{1Y}$, $c_{2Y}$, $c_{3Y}$, $p_Y$, $A_Y$, $N_Y$, $\alpha$ (for $\pi_X^Y$, i.e. $Y$'s beliefs on $X$'s release)
    - $*$ Given the sampled $\alpha_i$
      - $\cdot$ generate a sample $z_j \sim \beta e(\alpha_i, \alpha_i)$, $j = 1, ..., N$
      - $\cdot$ get $x_j = z_j \times T$, $j = 1, ..., N$
    - $*$ Monte Carlo approximation $M_Y^i(y)$ through $\frac{1}{N}\sum_{j=1}^{N} h_Y(x_j, y) = \frac{1}{N}[\sum_{x_j<y} h_{Y2}(x_j, y) + \sum_{y<x_j} h_{Y1}(x_j, y)] =$ (omitted)
    - $*$ $\Rightarrow$ find $y_i^* = \arg\max_{0\leq x\leq T} M^i{}_Y(y)$
  - $\Rightarrow$ Get approximate df $\hat{\Pi}_Y^X(y) = card\{y_i^* : y_i^* \leq y\}/K$

# ADVERSARIAL SOFTWARE TESTING: EXAMPLE

- Deciding $X$'s optimal release
  - Find $x^* = \arg\max_{0 \le x \le T} M_X(x)$
  - Maximise the objective function $M_X(x) = \int h_X(x,y)\pi_Y^X(y)dy$
  - Approximate df $\widehat{\Pi}_Y^X(y) = card\{y_i^* : y_i^* \le y\}/K$
  - Monte Carlo approximation through $\frac{1}{K}\sum_{i=1}^{K} h_X(x,y_i^*) = \frac{1}{K}[\sum_{y_i^* \le x} h_{X2}(x,y_i^*) + \sum_{y_i^* \ge x} h_{X1}(x,y_i^*)] =$ (omitted)

# ADVERSARIAL SOFTWARE TESTING: EXAMPLE



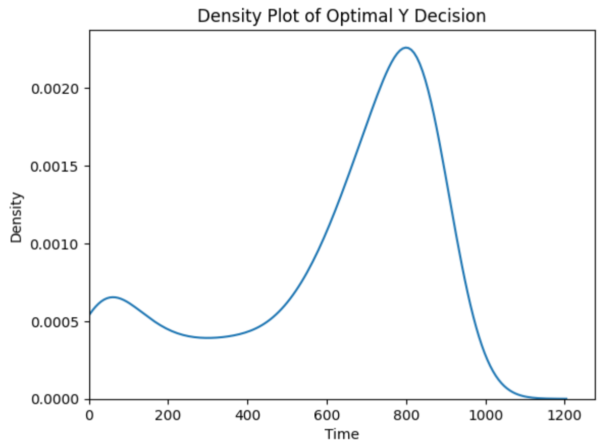

- $\beta e(\alpha, \alpha)$ distribution (mean 0.5) on $X$'s release $\Rightarrow$ guess $1000 = 0.5 * 2000$

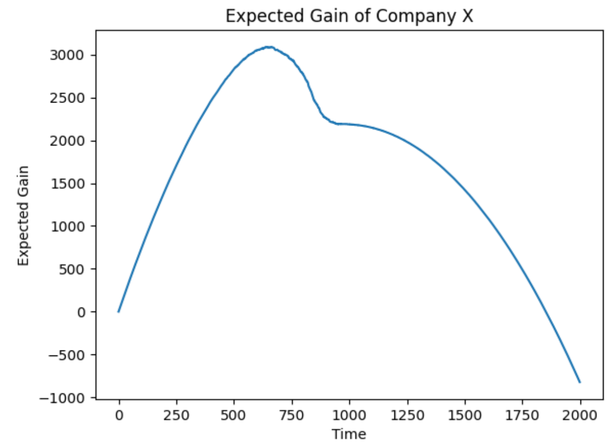
- LEFT: $Y$'s optimal release time up to 800 days (out of 2000) with some incentive to very early release but the optimal ones are between 300 and 700

- RIGHT: bimodality in $X$'s optimal release, with two possible strategies, one before $Y$'s release and one after it

- $X$'s optimal release occurs on day 483 for an expected gain of 2,442

# ADVERSARIAL SOFTWARE TESTING: EXAMPLE



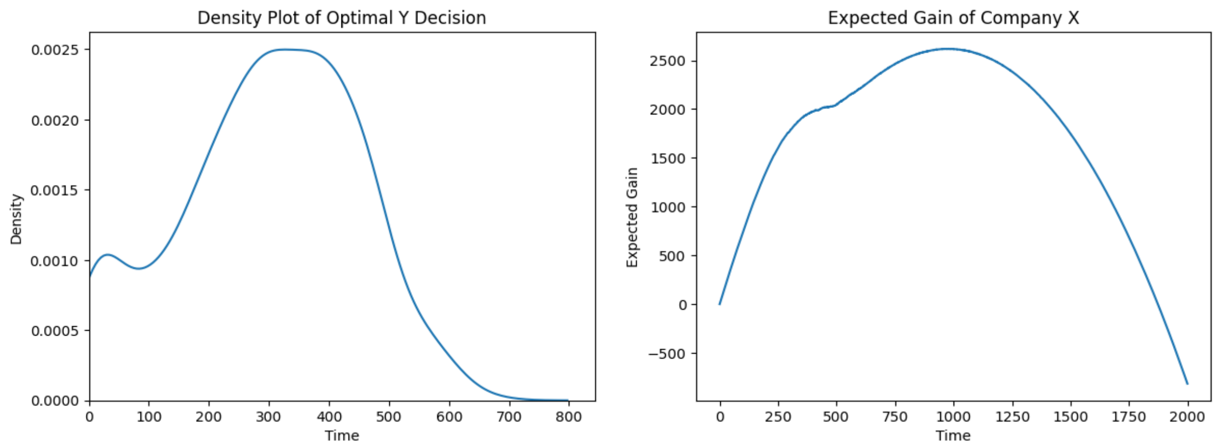

- $X$ thinks that $Y$ thinks that $X$ will release later $\Rightarrow \beta e(\alpha, \alpha)$ on $X$'s release replaced with $\beta e(3\alpha, \alpha) \Rightarrow$ guess $1,500 = 0.75{*}2000$

- LEFT: $Y$'s optimal release up to 1200 days with some incentive to very early release and optimal ones between 700 and 900 (compare with 300 and 700)

- RIGHT: $X$'s optimal release is before $Y$'s one

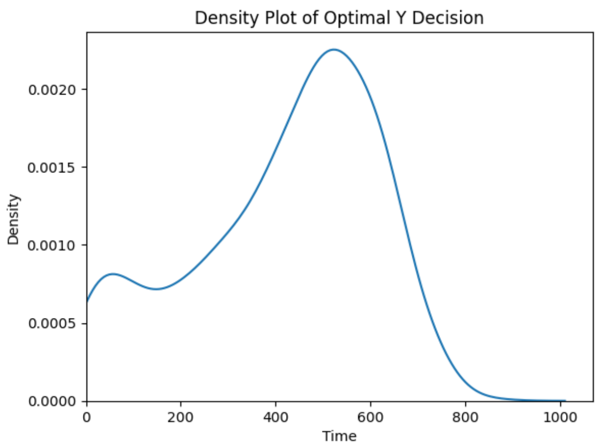
- $X$'s optimal release on day 663 for an expected gain of 3,091 (earlier 483 and 2,442)

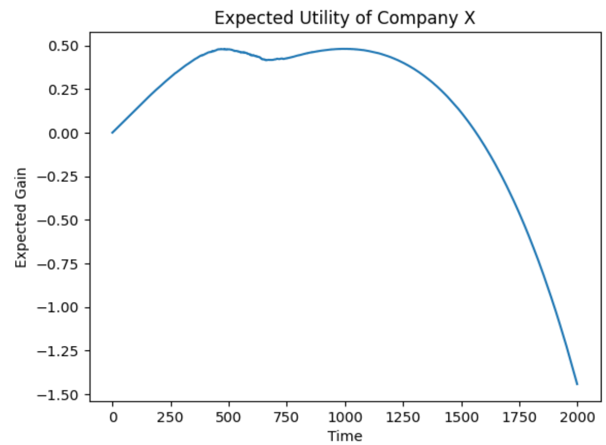
# ADVERSARIAL SOFTWARE TESTING: EXAMPLE




- $X$ thinks that $Y$ thinks that $X$ will release earlier
$\Rightarrow \beta e(\alpha, \alpha)$ on $X$'s release replaced with $\beta e(\alpha, 3\alpha) \Rightarrow$ guess $500 = 0.25 * 2000$
- LEFT: $Y$'s optimal release up to 800 days with some incentive to very early release and high-risk early release between 200 and 500 (earlier 300 & 700 and 700 & 900)
- RIGHT: $X$'s optimal release is well after the $Y$'s high-risk one
- $X$'s optimal release on day 978 with expected gain of 2,619 (earlier 483 & 2,442 and 663 & 3,091)

# ADVERSARIAL SOFTWARE TESTING: EXAMPLE




- Risk averse $X \Rightarrow$ identity utility replaced with constant absolute risk averse (CARA) model given by $u(x) = 1 - \exp(-\rho x)$, with risk aversion parameter $\rho = 0.001$

- LEFT: $Y$'s optimal release between 300 and 700 unchanged w.r.t. the first plot

- RIGHT: Still bimodal distribution for $X$'s optimal release, but tendency to be more conservative and wait more

- $X$'s optimal release on day 1003 (483 under identity) with expected utility (no more gain!) of 0.48

# AST: CURRENT WORK

- Multiple producers
  - Instead of $x < y$ and $x > y$, consider order statistics and position $X$'s release time between $x_{(i-1)}$ and $x_{(i+1)}$ for all $i$'s
  - Similar formulas w.r.t. previous ones
- Multiple decision variables
  - So far the $A_X$ purchase probability has been considered only as a function of the release time but it should depend also on other variables, like price and quality of the software
- Multiple buyers

# REFERENCES


- Banks, D., Rios, J., and Rios Insua, D. (2015). *Adversarial Risk Analysis* (Vol. 343). CRC Press.

- Rios Insua, D., Rios, J. and Banks, D. (2009). Adversarial risk analysis. *Journal of the American Statistical Association*, 104, 841-854.

- Gonzalez-Ortega, J., Soyer, R., Rios Insua, D. and Ruggeri, F. (2021), An Adversarial Risk Analysis Framework for Batch Acceptance Problem. *Decision Analysis*, 18, 25-40.

- Rios Insua, D., Ruggeri, F., Soyer, R. and Rasines, D.G. (2018), Adversarial issues in reliability. *European Journal of Operational Research*, 266, 1113-1119.

- Rios Insua, D., Ruggeri, F., Soyer, R. and Wilson S. (2020), Advances in Bayesian Decision Making in Reliability. *European Journal of Operational Research*, 282, 1-18.

# REFERENCES


- Gonzalez-Ortega, J., Rios Insua, D., Ruggeri, F. and Soyer, R. (2021), Hypothesis Testing in Presence of Adversaries. *The American Statistician*, 75, 31-40.

- Naveiro, R., Redondo, A., Rios Insua, D. and Ruggeri, F. (2019), Adversarial classification: An adversarial risk analysis approach. *International Journal of Approximate Reasoning*, 113, 133-148.

- Gallego, V., Naveiro, R., Redondo, A., Rios Insua, D. and Ruggeri, F., Protecting Classifiers From Attacks. Under revision for *Statistical Science*.

- Soyer, R., Ruggeri, F., Rios Insua, D., Pierce, C. and Guevara, C., An adversarial risk analysis framework for software release decision support. *Submitted.*

- Rios Insua, D., Ruggeri, F., Alfaro, C. and Gomez, J. (2016), Robustness for Adversarial Risk Analysis. In *Robustness Analysis in Decision Aiding, Optimization and Analytics*, M. Doumpos, C. Zopounidis and E. Grigoroudis Eds., Springer, 19-58.

# REFERENCES


- Arias, P., Ruggeri, F. and Suarez-Llorens, A. (2016), New classes of priors based on stochastic orders and distortion functions. *Bayesian Analysis*, 11, 1107-1136.

- Ruggeri, F., Sanchez-Sanchez, M., Sordo, M.A. and Suarez-Llorens, A. (2020), On a new class of multivariate prior distributions: theory and application in reliability. *Bayesian Analysis*, 16, 31-60.

- Rios Insua, D. and Ruggeri, F. Eds. (2000), *Robust Bayesian Analysis*, Springer, New York, USA.

- Cavallo, D. and Ruggeri, F. (2001), Bayesian models for failures in a gas network, *Safety and Reliability*, E. Zio, M. Demichela and N. Piccinini, Eds. , pp. 1963-1970, Politecnico di Torino Editore.