Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Institute for AI and
Autonomous Systems

Safe Robot Learning

An Overview

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure

- Safe learning preliminaries
- Safe learning methods

Safe Learning Motivation

Safety Preliminaries

Safety is "the state of being safe and protected from danger or harm, the state of not being dangerous" (Oxford Dictionary)

- Safety is a property that defines whether a robot behaves in a manner that **prevents it from performing dangerous actions**

Safety Preliminaries

Safety is "the state of being safe and protected from danger or harm, the state of not being dangerous" (Oxford Dictionary)

- Safety is a property that defines whether a robot behaves in a manner that **prevents it from performing dangerous actions**
- Safety is **an essential property of robot behaviour** — deploying unsafe robots is simply irresponsible

Safety Preliminaries

Safety is "the state of being safe and protected from danger or harm, the state of not being dangerous" (Oxford Dictionary)

- Safety is a property that defines whether a robot behaves in a manner that **prevents it from performing dangerous actions**
- Safety is **an essential property of robot behaviour** — deploying unsafe robots is simply irresponsible
- The meaning of safety is **context-dependent**, i.e. discussing safety only makes sense **with respect to concrete context-specific constraints**
  - An autonomous car can be safe if it obeys traffic rules, even when it drives at $100km/h$
  - Driving at such high speeds is clearly unsafe for a robot navigating in an indoor environment

Safety Preliminaries

Safety is "the state of being safe and protected from danger or harm, the state of not being dangerous" (Oxford Dictionary)

- Safety is a property that defines whether a robot behaves in a manner that **prevents it from performing dangerous actions**
- Safety is **an essential property of robot behaviour** — deploying unsafe robots is simply irresponsible
- The meaning of safety is **context-dependent**, i.e. discussing safety only makes sense **with respect to concrete context-specific constraints**
  - An autonomous car can be safe if it obeys traffic rules, even when it drives at $100km/h$
  - Driving at such high speeds is clearly unsafe for a robot navigating in an indoor environment
- For learning-based robots, safety needs to be considered:
  - To prevent damages during learning **if data collection is performed on a physical robot**
  - During deployment so that **learned models can be applied without the danger of performing hazardous actions**, particularly in the presence of humans

System Model

- Let us consider **a robot with state s_t that can apply actions a_t, potentially under the influence of noise w**, such that its operation is governed by a dynamical system of the form

$$\boldsymbol{s}_{t+1} = \boldsymbol{f}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right)$$

System Model

- Let us consider **a robot with state s_t that can apply actions a_t, potentially under the influence of noise w**, such that its operation is governed by a dynamical system of the form

$$\boldsymbol{s}_{t+1} = \boldsymbol{f}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right)$$

- The robot's performance on a task is evaluated by **an additive cost function J** defined as

$$J(\boldsymbol{s}_{0:T}, \boldsymbol{a}_{1:T-1}) = l(\boldsymbol{s}_T) + \sum_{t=0}^{T-1} l\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$$

System Model

- Let us consider **a robot with state s_t that can apply actions a_t, potentially under the influence of noise w**, such that its operation is governed by a dynamical system of the form

$$\boldsymbol{s}_{t+1} = \boldsymbol{f}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right)$$

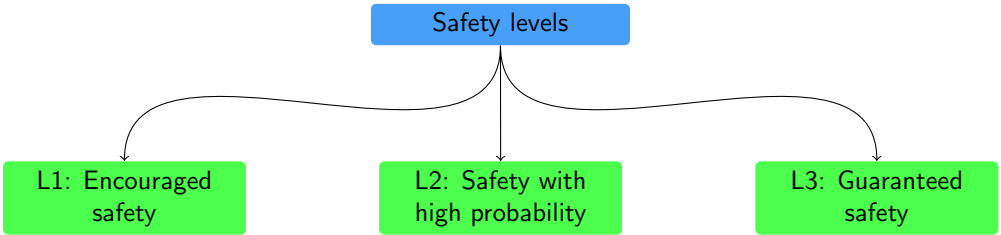
- The robot's performance on a task is evaluated by **an additive cost function J** defined as

$$J(\boldsymbol{s}_{0:T}, \boldsymbol{a}_{1:T-1}) = l(\boldsymbol{s}_T) + \sum_{t=0}^{T-1} l\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$$

- The robot's operation needs to be performed under **safety constraints**, which are described by N_c **constraint functions** $c_k\left(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{w}\right), 1 \leq k \leq N_c$ and can be of multiple types:
  - **State constraints**: A subset of safe states $S_c \subseteq S$
  - **Input constraints**: A subset of safe actions $A_c \subseteq A$
  - **Stability guarantees**: Ensure that the robot's operation is robust to small perturbations

Safety Levels

There is no single notion of safety that is always applicable — safety is best observed at different levels



Safety Level 1: Encouraged Safety (Soft Constraints)

- The robot is **encouraged** (rather than enforced) **to satisfy the safety constraints**

Safety Level 1: Encouraged Safety (Soft Constraints)

- The robot is **encouraged** (rather than enforced) **to satisfy the safety constraints**
- Can be achieved in multiple ways:

Safety Level 1: Encouraged Safety (Soft Constraints)

- The robot is **encouraged** (rather than enforced) **to satisfy the safety constraints**
- Can be achieved in multiple ways:
  - **Encouraging safety at every step along a trajectory** by defining the constraints as

$$c_k\left(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{w}\right) \leq \epsilon_k, 1 \leq k \leq N_c$$

Safety Level 1: Encouraged Safety (Soft Constraints)

- The robot is **encouraged** (rather than enforced) **to satisfy the safety constraints**
- Can be achieved in multiple ways:
  - **Encouraging safety at every step along a trajectory** by defining the constraints as

$$c_k\left(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{w}\right) \leq \epsilon_k, 1 \leq k \leq N_c$$

  - **Guaranteeing expected safety on a complete trajectory**:

$$J = E\left[\sum_{t=0}^{T-1} c_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right)\right] \leq d_k, 1 \leq k \leq N_c$$

Safety Level 1: Encouraged Safety (Soft Constraints)

- The robot is **encouraged** (rather than enforced) **to satisfy the safety constraints**
- Can be achieved in multiple ways:
  - **Encouraging safety at every step along a trajectory** by defining the constraints as

$$c_k\left(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{w}\right) \leq \epsilon_k, 1 \leq k \leq N_c$$

  - **Guaranteeing expected safety on a complete trajectory**:

$$J = E\left[\sum_{t=0}^{T-1} c_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right)\right] \leq d_k, 1 \leq k \leq N_c$$

  - **Adding a high cost to $J(\boldsymbol{s}_{0:T}, \boldsymbol{a}_{1:T-1})$ for constraint violation**

Safety Level 2: Safety With High Probability

- In this case, **safety is guaranteed with a given probability**

Safety Level 2: Safety With High Probability

- In this case, **safety is guaranteed with a given probability**
- Can be specified as

$$P\left(c_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) \leq 0\right) \geq p_k \; \forall t \in [0, T]$$

Safety Level 2: Safety With High Probability

- In this case, **safety is guaranteed with a given probability**
- Can be specified as

$$P\left(c_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) \leq 0\right) \geq p_k \; \forall t \in [0, T]$$

- Here, p_k **is a probability threshold** that defines the **certainty level** with which the constraint is satisfied

Safety Level 3: Guaranteed Safety

- At this level, **the compliance with safety constraints is formally guaranteed**

Safety Level 3: Guaranteed Safety

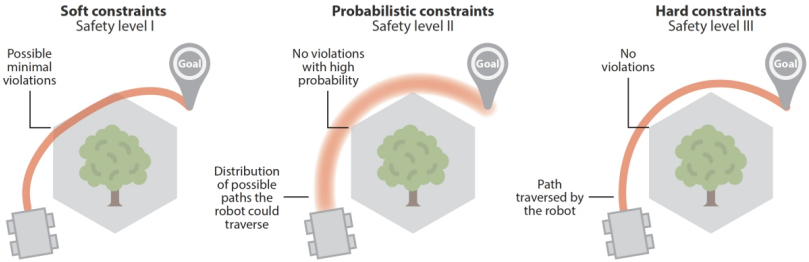
- At this level, **the compliance with safety constraints is formally guaranteed**
- Specified by hard constraints of the form

$$c_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) \leq 0 \; \forall t \in [0, T]$$

Safety Level 3: Guaranteed Safety

- At this level, **the compliance with safety constraints is formally guaranteed**
- Specified by hard constraints of the form

$$c_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) \leq 0 \; \forall t \in [0, T]$$



System and Property Specification Formalisms


Formal Specification and Verification of Autonomous Robotic Systems: A Survey

Authors: Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, Michael Fisher Authors Info & Claims

ACM Computing Surveys, Volume 52, Issue 5 • Article No.: 100, pp 1–41 • https://doi.org/10.1145/3342355

- For systems that are not learning-based, there are many **formal specification frameworks** that can be used to specify dynamical systems and the (safety) properties to be satisfied
- **State transition diagrams**, such as state machines or Petri nets, are predominant for modelling the operation of dynamical systems
- For specifying properties about the operation, **temporal logic**, such as linear temporal logic, is often used

Table 2. Summary of the Types of Formalisms Found in the Literature for Specifying the System and the Properties to Be Checked

	System		Property	
Formalism	**References**	**Total**	**References**	**Total**
Set Based	[81], [89], [117], [178], [179]	5		0
State-Transition	[1], [20], [18], [25], [36], [39], [40], [41], [48], [49], [61], [62], [66], [70], [83], [88], [97], [98], [51], [99], [104], [108], [110], [111], [115], [116], [129], [141], [146], [174], [175], [182], [186]	33		0
Logics		6		32
Temporal Logic		0	[29], [36], [57], [55], [61], [62], [70], [71], [83], [95], [97], [98], [51], [99], [104], [108], [110], [111], [115], [116], [123], [129], [138], [141], [176], [174], [175]	27
Dynamic Logic	[113], [125]	2	[113], [125]	2
Other Logics	[29], [64], [80], [164]	4	[64], [80], [164]	3
Process Algebra	[6], [123], [132]	3	[132]	1
Ontology	[11], [129], [137], [166]	4		0
Other	[57], [55], [71], [138], [176]	5	[1], [20], [18], [25], [40], [41], [66], [180]	8

Verification Approaches


Formal Specification and Verification of Autonomous Robotic Systems: A Survey

Authors: Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, Michael Fisher

ACM Computing Surveys, Volume 52, Issue 5 • Article No.: 100, pp 1–41 • https://doi.org/10.1145/3342355

- For systems that are not learning-based, there are also various techniques that can be used for **formal verification** with respect to desired systems properties
- **Model checking**, particularly in conjunction with temporal logic, is the most commonly used verification technique
- In addition to techniques for offline verification, **runtime monitoring** is also commonly used to verify the online operation of a robot — this enables a robot to trigger **recovery behaviours** or **mitigating actions**

Table 3. Summary of the Formal Verification Tools, and the Type of Tools, Identified in the Core Set of 63 Papers Surveyed

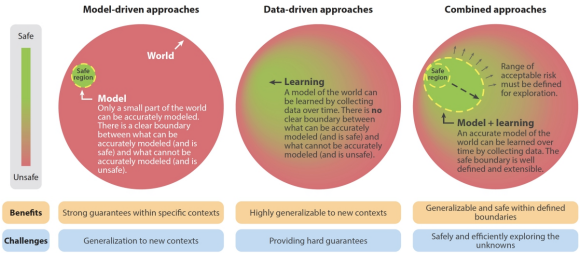
Type of Tool	Tool	References	Total	Type Total
Model Checkers	PRISM	[110], [123], [111], [36]	4	25
	NuSMV	[61], [62]	2	
	UPPAAL	[97], [98], [51]	3	
	SAL	[89]	1	
	SPIN	[174], [175], [50], [70], [176]	5	
	Beryl	[138], [139]	2	
	Aldebaran	[43]	1	
	Dfinder	[24], [18], [25], [1]	4	
	Unspecified	[40], [30], [104]	3	
Program Model Checkers	AJPF	[71], [55], [57], [176]	4	7
	MCMAS	[114], [44], [129]	3	
Theorem Provers	KeyMaera	[125], [113]	2	3
	SteP	[89]	1	
Others	Bio-PEPA Tool Suite	[123]	1	14
	TmeNET	[49]	1	
	TuLiP	[116]	1	
	LTLMoP	[116], [141]	2	
	Alloy	[29], [81]	2	
	Evaluator	[20]	1	
	minisat	[20]	1	
	MissionLab (VIPARS)	[132]	1	
	RV-BIP	[66]	1	
	Community Z Tools	[117],[178], [179]	3	

Models, Safety, and Learning-Based Methods

- But then there is learning, where applying the widespread methods for formal specification and verification is challenging

Models, Safety, and Learning-Based Methods

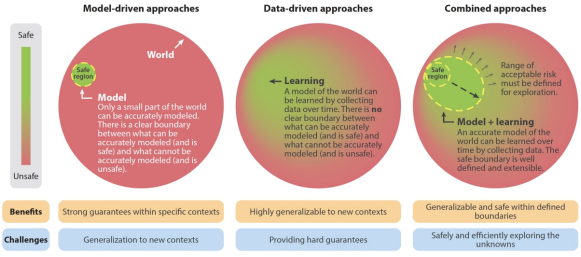
- But then there is learning, where applying the widespread methods for formal specification and verification is challenging



- In general, model-based techniques **operate within a well-defined safety region**, and operation outside that region is unsafe

Models, Safety, and Learning-Based Methods

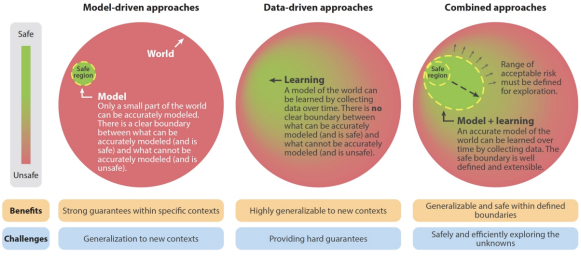
- But then there is learning, where applying the widespread methods for formal specification and verification is challenging



- In general, model-based techniques **operate within a well-defined safety region**, and operation outside that region is unsafe
- Pure learning-based methods, on the other hand, are **more flexible in terms of the valid operating region**, but **blur the boundary between regions of safe and unsafe operation**

Models, Safety, and Learning-Based Methods

- But then there is learning, where applying the widespread methods for formal specification and verification is challenging



- In general, model-based techniques **operate within a well-defined safety region**, and operation outside that region is unsafe
- Pure learning-based methods, on the other hand, are **more flexible in terms of the valid operating region**, but **blur the boundary between regions of safe and unsafe operation**

- The challenge is thus that of defining learning methods that **preserve the flexibility and generalisability of data-driven approaches**, but which are also **safe to apply during exploration and deployment**

Safe Learning Methods

Safe Learning Formalisation

- Formally, the safe learning problem is similar to that of a typical learning problem as discussed in our previous lectures — but with one key difference
- The objective is to **find an optimal policy**, but by minimising $J\left(\boldsymbol{s}_{0:T}, \boldsymbol{a}_{0:T-1}\right)$ not only over the dynamic model, but also over the safety constraints
- This is specified as follows:

$$
\begin{aligned}
J^{\pi^*}(\overline{\boldsymbol{s}}_0) = \min_{\pi} \; & J\left(\boldsymbol{s}_{0:T}, \boldsymbol{a}_{0:T-1}\right) + l(\boldsymbol{\epsilon}) \\
\text{subject to} \quad & \boldsymbol{s}_{t+1} = f\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) \\
& \boldsymbol{a}_t = \pi(\boldsymbol{s}_t) \\
& \boldsymbol{s}_0 = \overline{\boldsymbol{s}}_0 \\
& \text{safety constraints as defined} \\
& \quad \text{before for different safety levels}
\end{aligned}
$$

where $\boldsymbol{\epsilon}$ is only used with soft constraints

Safe Learning and Classical Control

- In safe learning, the dynamics model f is typically assumed to be composed of **a known prior model** $\overline{f}$ and **an uncertain component** $\hat{f}$:

$$f\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) = \overline{f}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right) + \hat{f}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{w}_t\right)$$

- Approaches to safe learning are closely related to **adaptive control** and **robust control**; some particularly use MDP extensions that are based on adaptive and robust control

- For instance, **robust MDPs** solve an optimisation problem **over the policy and the dynamics uncertainty**:

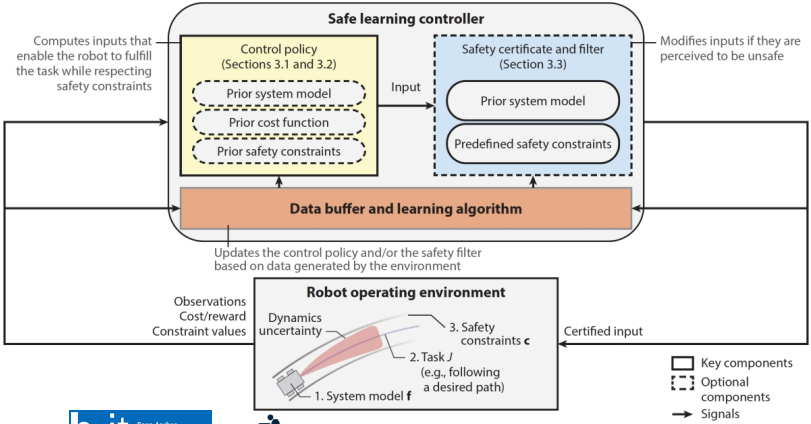
$$J^{\pi^*}\left(\overline{\boldsymbol{s}}_0\right) = \min_{\pi} \max_{\hat{f} \in \mathbb{D}} J\left(\boldsymbol{s}_{0:T}, \boldsymbol{a}_{0:T-1}\right)$$

subject to the same conditions are before

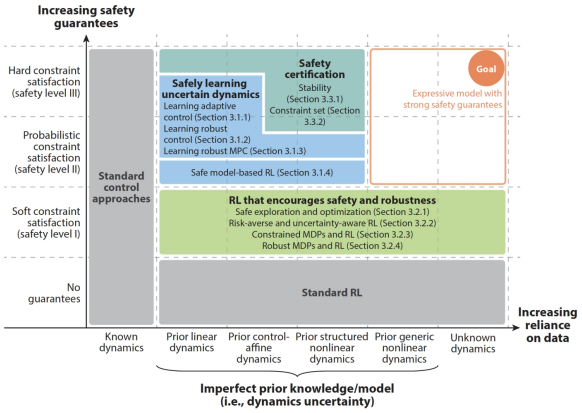
where $\mathbb{D}$ defines an uncertainty set for $\hat{f}$

Safe Learning Problems

- Safe learning can focus on two main aspects:
  - **Policy learning**, which is concerned with learning a **policy with safety guarantees**
  - **Learning safety filters**, where the policy itself may not have safety guarantees, but where the outputs are post-processed with respect to safety properties



Safety Levels and Safe Learning Approaches



- The safe learning approaches that are used in practice differ based on the safety level that should be achieved:
  - Soft constraint satisfaction can be achieved using approaches such as **safe exploration**, **uncertainty-aware RL**, or **robust MDPs**
  - Probabilistic safety can be achieved using **safe model-based RL** or **safe learning of dynamics models**
  - Hard constraint satisfaction can also be achieved using safe dynamics model learning, but also using **safety verification**
- Thus, the suitability of different approaches is application-dependent, namely it depends on the safety levels that are acceptable in a given context

Lipschitz Continuity

- One commonly used criterion when analysing learned models is Lipschitz continuity

Lipschitz Continuity

- One commonly used criterion when analysing learned models is Lipschitz continuity
- A real-valued function f is called **Lipschitz continuous** if there is a constant C such that

$$\|f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1)\| \leq C\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|$$

Lipschitz Continuity

- One commonly used criterion when analysing learned models is Lipschitz continuity
- A real-valued function f is called **Lipschitz continuous** if there is a constant C such that

$$\|f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1)\| \leq C\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|$$

- Essentially, Lipschitz continuity defines **a limit on the absolute value on the slope of** f
  - This is useful for robot models because it would mean that small input changes would not cause large output changes

Lipschitz Continuity

- One commonly used criterion when analysing learned models is Lipschitz continuity
- A real-valued function f is called **Lipschitz continuous** if there is a constant C such that

$$\|f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1)\| \leq C\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|$$

- Essentially, Lipschitz continuity defines **a limit on the absolute value on the slope of f**
  - This is useful for robot models because it would mean that small input changes would not cause large output changes
- The smallest C for which the above condition holds is referred to as **the Lipschitz constant of f**

Lipschitz Continuity and Safe Learning

- The concept of Lipschitz continuity can be utilised for safe learning by **learning a policy with a desired Lipschitz constant**

[1] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: analysis and efficient estimation", in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Lipschitz Continuity and Safe Learning

- The concept of Lipschitz continuity can be utilised for safe learning by **learning a policy with a desired Lipschitz constant**
- There are a few ways in which this can be done:
  - **Enforcing a Lipschitz constant** during training (requires a dedicated training procedure)
  - Training a policy iteratively **until a desired Lipschitz constant is achieved** (requires a procedure for estimating the constant)

[1] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: analysis and efficient estimation", in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Lipschitz Continuity and Safe Learning

- The concept of Lipschitz continuity can be utilised for safe learning by **learning a policy with a desired Lipschitz constant**
- There are a few ways in which this can be done:
  - **Enforcing a Lipschitz constant** during training (requires a dedicated training procedure)
  - Training a policy iteratively **until a desired Lipschitz constant is achieved** (requires a procedure for estimating the constant)
- **Estimating the Lipschitz constant is NP-hard in general**[1]
  - The policy class needs to be restricted so that the constant can be calculated — the method is not applicable in general
  - There are, however, procedures for computing upper bounds of the Lipschitz constant

[1] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: analysis and efficient estimation", in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Ergodicity and Safe Exploration

- A commonly desired property of MDPs is **ergodicity**: An MDP is ergodic if **every state can be reached from every other state given some policy**
  - But ergodicity is violated in terminal states (e.g. there is no recovery from breaking an egg during manipulation) and in some unsafe states

[2] T. M. Moldovan and P. Abbeel, "Safe Exploration in Markov Decision Processes", in *Proc. 29th Int. Conf. Machine Learning (ICML)*, 2012, pp. 1451–1458.

Ergodicity and Safe Exploration

- A commonly desired property of MDPs is **ergodicity**: An MDP is ergodic if **every state can be reached from every other state given some policy**
  - But ergodicity is violated in terminal states (e.g. there is no recovery from breaking an egg during manipulation) and in some unsafe states
- Ergodicity is a property that can be used to **define safe exploration procedures**: By enforcing ergodicity on the model, the policy can be guaranteed to avoid unsafe states

[2] T. M. Moldovan and P. Abbeel, "Safe Exploration in Markov Decision Processes", in *Proc. 29th Int. Conf. Machine Learning (ICML)*, 2012, pp. 1451–1458.

Ergodicity and Safe Exploration

- A commonly desired property of MDPs is **ergodicity**: An MDP is ergodic if **every state can be reached from every other state given some policy**
  - But ergodicity is violated in terminal states (e.g. there is no recovery from breaking an egg during manipulation) and in some unsafe states
- Ergodicity is a property that can be used to **define safe exploration procedures**: By enforcing ergodicity on the model, the policy can be guaranteed to avoid unsafe states
- **Finding whether there is such a safe, ergodic policy is also an NP-hard problem**[2]
  - In practice, this can only be guaranteed with a given probability

[2] T. M. Moldovan and P. Abbeel, "Safe Exploration in Markov Decision Processes", in *Proc. 29th Int. Conf. Machine Learning (ICML)*, 2012, pp. 1451–1458.

Risk-Averse Reinforcement Learning

- Safe learning can also be performed by **using an estimate of risk during learning**
  - The estimate can be used to encourage **risk-averse robot behaviour**

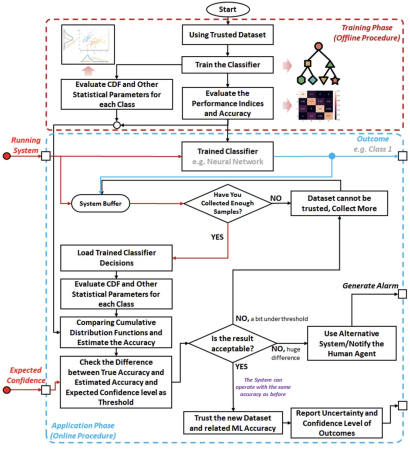
Risk-Averse Reinforcement Learning

- Safe learning can also be performed by **using an estimate of risk during learning**
  - The estimate can be used to encourage **risk-averse robot behaviour**
- This can be done with the help of a **predictive model** that calculates risk based on (i) **the robot's state** and (ii) **a sequence of actions that would be applied in subsequent steps**

Risk-Averse Reinforcement Learning

- Safe learning can also be performed by **using an estimate of risk during learning**
  - The estimate can be used to encourage **risk-averse robot behaviour**
- This can be done with the help of a **predictive model** that calculates risk based on (i) **the robot's state** and (ii) **a sequence of actions that would be applied in subsequent steps**
- Estimates provided by f are useful for computing the risk
  - If these are not available, **a risk estimation model can be learned in a preliminary, unsafe learning phase** (e.g. done in simulation or using offline data)

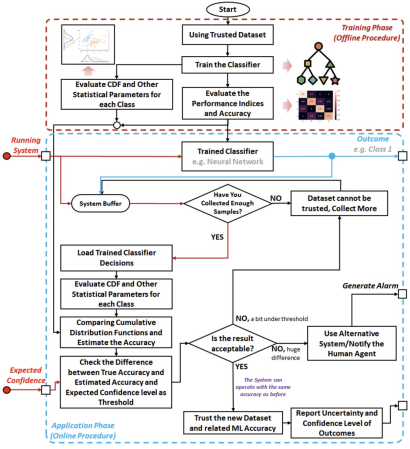
Discovering Distribution Shifts for Safe Operation



K. Aslansefat et al. "SafeML: Safety Monitoring of Machine Learning Classifiers Through Statistical Difference Measures," in *Int. Symp. Model-Based Safety and Assessment*, 2020, pp. 197–211.

- When using trained models, **distribution shift** can occur, which refers to a **discrepancy between the training and testing distributions**
  - Distribution shift is one reason why direct sim-to-real transfer is usually not possible

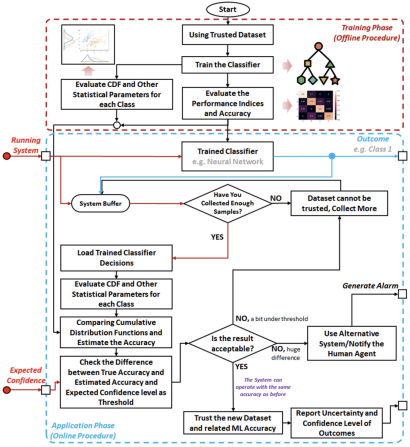
Discovering Distribution Shifts for Safe Operation



K. Aslansefat et al. "SafeML: Safety Monitoring of Machine Learning Classifiers Through Statistical Difference Measures," in *Int. Symp. Model-Based Safety and Assessment*, 2020, pp. 197–211.

- When using trained models, **distribution shift** can occur, which refers to a **discrepancy between the training and testing distributions**
  - Distribution shift is one reason why direct sim-to-real transfer is usually not possible
- To ensure safe decisions based on trained models, it is useful to **compute an estimate of this shift**

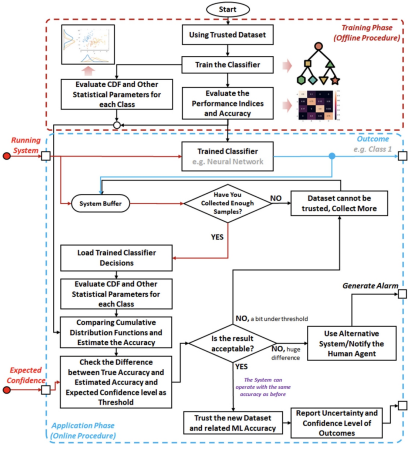
Discovering Distribution Shifts for Safe Operation



K. Aslansefat et al. "SafeML: Safety Monitoring of Machine Learning Classifiers Through Statistical Difference Measures," in *Int. Symp. Model-Based Safety and Assessment*, 2020, pp. 197–211.

- When using trained models, **distribution shift** can occur, which refers to a **discrepancy between the training and testing distributions**
  - Distribution shift is one reason why direct sim-to-real transfer is usually not possible
- To ensure safe decisions based on trained models, it is useful to **compute an estimate of this shift**
- This can be done by comparing the training distribution with the distribution observed during online application, for instance based on the **cumulative distribution functions** of the two datasets

Discovering Distribution Shifts for Safe Operation



K. Aslansefat et al. "SafeML: Safety Monitoring of Machine Learning Classifiers Through Statistical Difference Measures," in *Int. Symp. Model-Based Safety and Assessment*, 2020, pp. 197–211.

- When using trained models, **distribution shift** can occur, which refers to a **discrepancy between the training and testing distributions**
  - Distribution shift is one reason why direct sim-to-real transfer is usually not possible
- To ensure safe decisions based on trained models, it is useful to **compute an estimate of this shift**
- This can be done by comparing the training distribution with the distribution observed during online application, for instance based on the **cumulative distribution functions** of the two datasets
- An estimate of the online distribution requires **online samples** to be collected — **sufficient evidence** should be available before a model can be trusted

Summary

- Safety is an essential property of robot behaviour and is expressed by safety constraints that need to be satisfied throughout a robot's operation
- Safety can be enforced during learning or on an already learned policy
- There are different levels of safety, defined based on how the constraint satisfaction is guaranteed: safety based on soft constraints, probabilistic safety, and safety based on hard constraints
- Safe learning is based on a wide variety of methods, such as those enforcing Lipschitz continuity, ergodicity of a policy, or risk minimisation