Ethics in Artificial Intelligence (Module 3)

Last update: 19 April 2025

Academic Year 2024 – 2025
Alma Mater Studiorum · University of Bologna

Contents

1 Human agency and oversight

AI act, article 14 Article related to human oversight. It states that: AI act, article 14

- Human centric AI is one of the key safeguarding principles to prevent risks.
- AI systems must be designed and developed with appropriate interfaces to allow humans to oversee them.

Human agency AI systems should empower human beings such that they can: Human agency

- Make informed decisions.
- Foster their fundamental rights.

This can be achieved with methods like:

- Human-centric approaches,
- AI for social good,
- Human computation,
- Interactive machine learning.

Human oversight Oversight mechanisms to prevent manipulation, deception, conditioning from AI systems. Human oversight

Possible methods are:

- Human-in-the-loop,
- Human-on-the-loop,
- Human-in-command.

Human-centered AI framework Approach centered on high autonomy while keeping human control. Human-centered AI framework

Remark. Human agency and oversight happens at different levels:

Development team Responsible for the technical part.

Organization Decides who is in charge of accountability, validation, ...

External reviewers (e.g., certification entities).

1.1 Governance and methodology

Human-out-of-the-loop The environment is static and cannot integrate human knowledge. The AI system is a black-box that cannot be used in safety-critical settings. Human-out-of-the-loop

Human-in-the-loop (HITL) The environment is dynamic and can use expert knowledge. The AI system is explainable or transparent and suitable for safety-critical settings. Human-in-the-loop (HITL)

In practice, the AI system stops and waits for human commands before making a decision.

Society-in-the-loop The society, with its conflicting interests and values, is taken into account. Society-in-the-loop

Human-on-the-loop (HOTL) The AI system operates autonomously and the human can intervene if needed. Human-on-the-loop (HOTL)

Remark. Limitations of human-centric AI are:

- It does not scale well as human intervention is involved.
- It is hard to evaluate its effectiveness.
- Performance of the AI system might degrade.

1.2 HITL state-of-the-art approaches

1.2.1 Active learning

Active learning The system is in control of the learning process and the human acts as an oracle for labeling data. Active learning

The learner can query, following some strategy, the human for the ground-truth of unlabeled data. A general algorithm works as follows:

1. Split the data into an initial (small) pool of labeled data and a pool with the remaining unlabeled ones.
2. The model selects an example(s) to be labeled by the oracle.
3. The model is trained on the available labeled data.
4. Repeat until a stop condition is met.

The selection strategy can be:

Random

Uncertainty-based Select examples classified with the least confidence according to some metric.

Diversity-based Select examples that are rare or representative according to some metric.

Remark. This approach is effective in settings with lots of unlabeled data and annotating all of it is expensive.

Remark. This approach is sensitive to the choice of the oracle.

1.2.2 Interactive machine learning

Interactive machine learning Users interactively supply information that influences the learning process. Interactive machine learning

Remark. Compared to active learning, with interactive machine learning it is the human that selects the learning data.

1.2.3 Machine teaching

Machine teaching Human experts are completely in control of the learning process. There can be different types of teachers: Machine teaching

Omniscient teacher Complete access to the components of the learner (i.e., feature space, parameters, loss, optimization algorithm, ...).

Surrogate teacher Access to the loss.

Imitation teacher The teacher uses a copy of the learner that it can query to create a surrogate model.

Active teacher The teacher queries the learner and evaluates it based on the output.

Adaptive teacher The teacher selects examples based on the current hypothesis of the learner.

2 Technical robustness and safety

AI act, article 15 Article related to accuracy, robustness, and cybersecurity. It states that high-risk AI systems should: AI act, article 15

- Be benchmarked and evaluated adequately.
- Be resilient to errors.
- Have measures to prevent and respond to attacks.

Technical robustness and safety AI systems should be secured to prevent unintentional harm and minimize the consequences of intentional harm. These requirements can be achieved by: Technical robustness and safety

- Improving resilience to attacks.
- Introducing fallback plans.
- Improving general safety.
- Improving accuracy, reliability, and reproducibility.

> **Remark.** Robustness is required as the real-world distribution is usually different from the training one.

> **Remark** (Reliability vs robustness vs resilience)**.**
>
> **Reliability** Perform similarly on any test set from the same distribution.
>
> In practice, reliable design aims at obtaining a probability of failure below some threshold.
>
> **Robustness** Perform reasonably well on test sets from a slightly different distribution.
>
> In practice, robust design aims at obtaining a model insensitive to small changes.
>
> **Resilience** Adapt to unexpected inputs from unknown distributions.

Robustness levels Robustness can be ranked on different levels: Robustness levels

Level 0 No robustness measures or mitigation functionalities.

Level 1 Generalization under distribution shift. It aims at mitigating data shifts and out-of-distribution data.

Level 2 Robustness against a single risk.

Level 3 Robustness against multiple risks.

Level 4 Universal robustness against all known risks.

Level 5 Level 4 system with human-aligned and augmented robustness.

AI safety Build a system less vulnerable to adversarial attacks. This can be achieved by: AI safety

- Identifying anomalies.
- Defining safety objectives.

Reproducibility Build a system that exhibits the same behavior under the same conditions. Reproducibility

Remark (Repeatability vs replicability vs reproducibility)**.**

Repeatability The same team can repeat the results under the same experimental setup.

Replicability A different team can repeat the results under the same experimental setup.

Reproducibility A different team can repeat the results with some tolerance under a different experimental setup.

Robustness requirements

Robustness requirements Two aspects have to be considered for robustness:

Performance Capability of a model to perform a task reasonably well (humans can be used as baseline).

Vulnerability Resistance of the model to attacks. Possible sources of attack are: data poisoning, adversarial examples, flaws in the model.

Robustness approaches

Robustness approaches Robustness can be imposed with different methods at different moments of the lifecycle of the system:

- Data sanitization,
- Robust learning,
- Extensive testing,
- Formal verification.

2.1 Robust learning

Robust learning Learn a model that is general enough to handle slightly out-of-distribution data.

Remark. It is impossible (and possibly unwanted) to have a system that models everything.

Theorem 2.1.1 (Fundamental theorem of machine learning)**.**

$$\text{error rate} = \frac{\text{model complexity}}{\text{sample size}}$$

Corollary 2.1.1.1. If the sample size is small, the model should be simple.

Remark (Uncertainty in AI)**.** Knowledge in AI can be divided into:

Known knowns Well-established and understood areas of research:

- Theorem proving.
- Planning in deterministic and fully-observed worlds.
- Games of perfect information.

Known unknowns Areas whose understanding is incomplete:

- Probabilistic graphical models to represent and reason on uncertainty in complex systems.
- Probabilistic machine learning that is able to quantify uncertainty.
- Planning in Markov decision problems for decision-making under uncertainty.

- Computational game theory to analyze and solve games.

Unknown unknowns Areas that we do not know are unknown. They are the natural step toward robust AI.

Remark. Robustness in biology is achieved by means of a diverse and redundant population of individuals.

2.1.1 Robustness to model errors

Robust optimization Handle uncertainty and variability through optimization methods: Robust optimization

- Assign ranges to parameters to account for uncertainty.
- Optimize in a max-min formulation aiming at maximizing the performance of the worst-case.

Remark. There is a trade-off between optimality and robustness.

Model regularization Add a penalty term to the training loss to encourage simple models. Model regularization

Theorem 2.1.2. Regularization can be interpreted as robust optimization.

Optimize risk-sensitive objectives Consider, when optimizing a reward, the variability and uncertainty associated to it (e.g., minimize variance of rewards). Optimize risk-sensitive objectives

Robust inference Deal with uncertainty, noise, or variability at inference time. Robust inference

2.1.2 Robustness to unmodeled phenomena

Model expansion Expand the models with a knowledge base. Model expansion

Remark. New knowledge might contain errors or not improve the model at all.

Causal models Define causal relations. Causal models

Portfolio of models Have multiple solvers available and use a selection method to choose the most suited in any situation. Portfolio of models

Remark. Ideally, given an instance, there should be at least a solver that performs well on it.

Anomaly detection Detect instances that deviate from the expected distribution. Anomaly detection

2.2 Data sanitization

Data sanitization Methods to ensure that data is deleted and unrecoverable. Data sanitization

NIST guidelines Guidelines for media sanitization provided by the National Institute of Standards and Technology: NIST guidelines

1. Determine the level of sanitization based on the sensitivity of the information.
2. Choose a sanitization method such as:

   Clearing Remove data so that it is not recoverable from software.

   Purging Physically remove the data on the media (e.g., degaussing a hard disk).

   Destroying Physically destroy the media.

3. Document the sanitization process.
4. Verify and validate the sanitization process.
5. Create training and awareness programs on the importance of media sanitization.

2.3 Extensive testing

Robustness testing Software test to evaluate the capability of a system to maintain its functionalities under unexpected scenarios. Robustness testing

Key aspects to take into account are:

- Unexpected inputs,
- Edge cases,
- Stress testing and resource exhaustion,
- Fault tolerance,
- Error handling,
- Regression testing.

2.3.1 Model-based testing

Model-based testing Test the system on test cases generated based on a reference behavior model (i.e., expected output from a given input). Model-based testing

| **Remark.** To exhaustively test all possible cases, the solution space is very large.

Search-based testing Use meta-heuristics to generate test cases. Search-based testing

2.3.2 Results analysis testing

Passive testing Add observation mechanisms to a system under test to collect and analyze execution traces. A possible approach works as follows: Passive testing

1. Collect traces in case of failure.
2. Define the properties the system should verify in case of failures.
3. Check whether the execution trace satisfies the property.

2.4 Formal verification

Formal specification Unambiguous description of the system and its required properties. Formal specification

Formal verification Exhaustive comparison between the formal specification and the system. Formal verification

2.4.1 Theorem proving

Theorem proving Model the system as a set of logical formulae Γ and the properties as theorems Ψ. Verification is done through logical reasoning: Theorem proving

$$\models (\Gamma \Rightarrow \Psi)$$

Remark. Formalizing the system though logical formulae can be difficult.

2.4.2 Model checking

Model checking Model the system as a finite state machine M and the properties as formal representations Ψ. Verification is done through logical reasoning: Model checking

$$M \models \Psi$$

Remark. In practice, formal verification methods are difficult to scale due to the large state space.

2.5 Adversarial attacks

Adversarial attack Techniques to intentionally manipulate the result of a machine learning model. Adversarial attack

White-box attack The adversary has complete knowledge of the attacked system. White-box attack

Black-box attack The attacker can only interact with the system through input-output queries. Black-box attack

2.5.1 Poisoning

Data poisoning Manipulate the training data. Data poisoning

Model poisoning Attack the model at training time (e.g., inject malicious gradients, modify loss, manipulate hyperparameters, ...). Model poisoning

Algorithm poisoning Modify the underlying algorithm, introduce backdoors, ... Algorithm poisoning

2.5.2 Model-based

Inversion attack Reconstruct information about the training data based on the model output. Inversion attack

Extraction attack Recover information of the underlying model (e.g., parameters, architecture, training data, ...). Extraction attack

2.5.3 Evasion

Score-based attack Manipulate the output logits to induce a misclassification. Score-based attack

Patch attack Add imperceptible perturbations to the input to induce a misclassification. Patch attack

Gradient attack Compute or estimate the gradient of the training loss function to determine a perturbation of the input to induce a misclassification. Gradient attack

Decision attack Perturb the input to move across decision boundaries and induce a misclassification. Decision attack

Adaptive attack Dynamically adjust the attack strategy based on the model. Adaptive attack

3 Explainability

Transparency Ensure that appropriate information reaches the relevant stakeholders. Transparency

Explanation Evidence, support, or reasoning related to a system's output or process. Explanation

An explanation can be assessed by the following properties:

Quality Related to the accuracy of the explanation.

Quantity Related to the amount of information delivered.

Relation Whether it only contains relevant information.

Manner How the information is delivered.

Context-oriented Whether it accounts for the knowledge capabilities of the recipient of the explanation.

Knowledge limit Whether it is limited to the training data.

An explanation can be:

Attribute based Describe the contribution of the input features.

Rule based If-then rules based on the input features.

Counterfactual Determine which input features would have made the prediction different.

Argumentation based Produce the explanation by extracting and processing arguments.

3.1 Explanation taxonomy

3.1.1 Global vs local

Global explanation Explain the model as a whole. Global explanation

Local explanation Explain the output of the model for a particular instance. Local explanation

3.1.2 Approaches

Model (global) explanation Create an interpretable predictor that mimics the one to be explained on the entire input space. Model (global) explanation

Outcome (local) explanation Create an interpretable predictor that mimics the one to be explained on a portion of the input space. Outcome (local) explanation

Model inspection Create a representation that models the behavior of the system. Model inspection

Transparent box design Use an interpretable predictor. Transparent box design

3.2 XAI abstract framework

Interpretation Associate a (subjective) meaning to an object. Interpretation

Explanation Extract relevant aspects of an object to ease interpretation. Explanation

XAI abstract framework System composed of: XAI abstract framework

- A model M to explain with representation R,
- An explanation function E.

The explanation function should produce another model $M' = E(M)$ such that its representation R' is more interpretable and the performance difference between M and M' should be minimized.

3.3 Explanation via feature importance

Feature importance explanation Method that quantifies the importance score of each input feature for either local or global explanation. Feature importance explanation

3.3.1 Local interpretable model-agnostic explanations (LIME)

LIME Model-agnostic method for post-hoc (after training) explanation. Given a model f to explain and an input $\mathbf{x}$, LIME works as follows: LIME

1. Sample N points $\mathbf{z}_1, \dots, \mathbf{z}_N$ around $\mathbf{x}$ according to some proximity measure.
2. Form a dataset of the sampled points $\langle \mathbf{z}'_i, y_i \rangle$ where $\mathbf{z}'_i$ is the one-hot encoding of $\mathbf{z}_i$ and $y_i = f(\mathbf{z}_i)$.
3. Train an interpretable local surrogate model g on the sampled data.
4. Repeat with different hyperparameters of g and pick the one that maximizes the fidelity with f and minimizes the complexity of g.
5. Use the coefficients of g to measure feature importance.

| **Remark.** Global explanation can be performed by aggregating over multiple points.

3.4 Explanation via symbolic knowledge extraction

Symbolic knowledge extraction explanation Method that given a sub-symbolic model produces a symbolic representation of it (e.g., rule list, decision tree, decision table). Symbolic knowledge extraction explanation

The expressiveness of the extracted knowledge can be:

Propositional Boolean statements and logical connectives.

Fuzzy Hierarchical set of if-then-else statements with comparison between variables and constants.

Oblique Propositional logic with arithmetic comparison.

M-of-N Propositional, fuzzy, or oblique with the addition of statements in the form of m of $\{\phi_1, \dots, \phi_n\}$.

3.5 Symbolic knowledge injection

Symbolic knowledge injection Method to modify a predictor so that it is consistent with some symbolic knowledge provided by the user. Symbolic knowledge injection

Symbolic knowledge can be injected through:

Guided learning Encode input knowledge as a cost factor and include it in the training loss.

Structuring Modify the architecture of the predictor to mimic the knowledge.

Embedding Embed knowledge and inject it into the training set.

3.6 Argumentation

Argumentation Approach that, given some input, extracts the arguments and their semantics allowing to study their properties. Argumentation

3.6.1 Computational argumentation

Abstract argumentation Directed graph where nodes are arguments and arcs are relationships between arguments (i.e., support or attack). Abstract argumentation

There are two common approaches to classify arguments:

Extension-based Determine extensions (i.e., set of arguments):

Complete Set of arguments that is able to defend itself and includes all the arguments it defends.

Grounded Set of arguments whose defended by the initial arguments.

Stable Set of arguments that attack all the arguments not included in it.

Preferred Set of arguments that is as large as possible and able to defend itself.

Labeling-based Determine labels (e.g., the states of an argument).

Structured argumentation Explicitly model the relationship between premises and conclusions of the arguments. Structured argumentation

3.6.2 Defeasible logic as argumentation

Conclusive reasoning A reasoning schema is conclusive if its conclusions are always true when the premises hold. Conclusive reasoning

Defeasible reasoning A reasoning schema is defeasible if, under certain conditions, its conclusions are not true when the premises hold. Defeasible reasoning

Defeasible logic argumentation Arguments are defined as proof trees. Their relationships can be: Defeasible logic argumentation

Attack An argument A attacks a defeasible argument B if the conclusion of A is the complement of the conclusion of B and the conclusion of B is not part of a strict sub-argument of B.

Support A set of arguments S supports a defeasible argument A if every proper sub-argument of A is in S.

Undercut A defeasible argument A is undercut by a set of arguments S if S supports an argument B that attacks A.