MLMI10
Designing Intelligent Interactive Systems

Lecture 6

Per Ola Kristensson

Lent 2022

Key Concepts

- Adaptive interaction
- Interpreting user input
- Traps and tactics when designing for uncertainty
- Automation
- Interactive machine learning

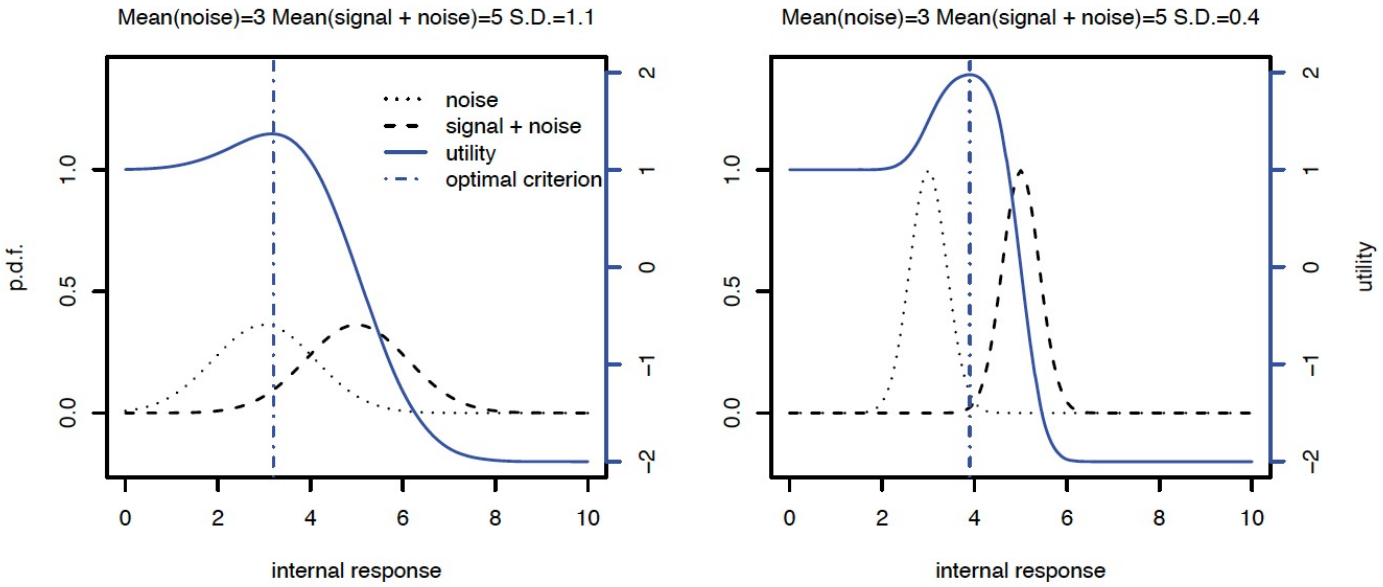
Adaptive Interaction

All slides on adaptive interaction from: Payne, S.J. and Howes, A. 2003. *Adaptive Interaction*. Morgan & Claypool Publishers.

Adaptive interaction

- The adaptive interaction framework hypothesizes that users are maximally adaptive by seeking out a strategy under the following constraints:
  - **Utility**: what a user wants to do, what they find valuable; in other words the user's own internal subjective utility functions
  - **Ecology**: the constraints imposed by the environment
  - **Mechanism**: the cognitive information processing mechanisms available to the user
- **Users choose an interaction strategy that strives to maximise utility given the constraints**
- Interaction is adaptive as it is constrained by these three constraints
- If a theory removes one of the above three constraints, the space of possible interaction strategies becomes unbounded and much more difficult to explain

Adaptive interaction example: signal detection theory



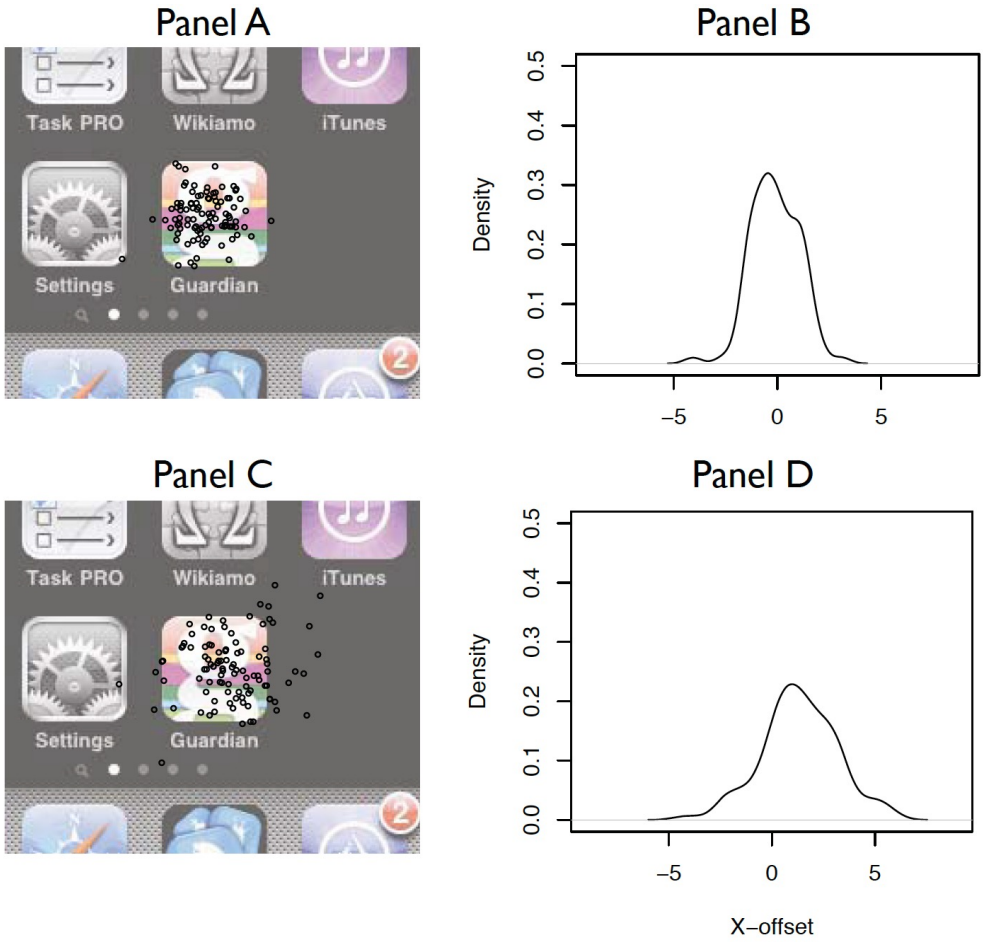
- Binary classification problem (the diagnosis problem): separating signal from noise by setting a threshold along a single dimension of measurement
- Noise in the left-hand figure have been reduced in the right-hand figure (perhaps due to a user interface redesign)
- As a result, if the user changes criterion (strategy), a higher payoff is possible
- Environment and cognitive mechanisms combine to define the user's sensitivity

Adaptive interaction example: target acquisition (1/2)

- Target acquisition: moving an implement (stylus, mouse, finger) to a target and triggering (acquiring it)
- A motor-perceptual task subject to noise:
  - Device sensor noise
  - Noise in software processing (modelling assumptions)
  - Noise in cognitive processing, such as vision affected by environmental noise affecting rate of photons arriving at the retina
  - Noise due to perceptual amplification processes in the human visual system
  - Neuron firing noise
  - Noise in the human neuromuscular system
  - Noise in muscle fibers converting electrical signals to mechanical movement

Adaptive interaction example: target acquisition (2/2)

- Touch point distributions for target (the *Guardian* icon)
- Utility should reflect intrinsic variability and the cost of an error
- Acquiring the distractor target *Settings* to the left of the target incurs a higher negative utility than merely missing the target to the right, which will only result in the user having to try again
- Users with higher variability have to offset their touch point distributions more to ensure they maximise utility by avoiding the distractor target
- Touch point variance, spatial layout and rewards/penalties for acquiring correct targets are good predictors of where users tend to aim



Adaptive interaction example: movement duration (1/2)

- We are often interested in the movement duration when a user is attempting to acquire a target
- This is fundamentally governed by a speed-accuracy tradeoff (faster movement is less precise)
  - For an optimality analysis, how the variability in targeting changes with movement duration
  - In general, variability will decrease as movement duration increase
- While part of the targeting movement is (probably) ballistic, part of it is guided by closed-loop control via the human vision and proprioception systems
- A user's objective is to choose muscular actuation/force for the desired target to be selected as fast as possible **and** with acceptable variance
  - There is another trade-off here: faster movements results in higher variance
- The utility currency is **time**

Adaptive interaction example: target acquisition (2/2)

- A user chooses a strategy S by minimising expected movement duration T_S and average time cost for targeting error T_E:
- $S = \arg\min_S (T_S + T_E)$
  - $T_S = b \log \frac{D}{V_S'} + (b - a) \log C$
  - where a and b are empirically determined parameters, D is the distance from the starting point to the **end** of the target and V_S' is the variability for strategy S
  - $b \log \frac{D}{V_S'}$ describes the first submovement as a ballistic motion
  - $(b - a) \log C$ describes the second submovement as a small corrective submovement under closed-loop control
  - $C = \frac{V_S}{D}$, where V_S is the variability in movement for the strategy S at distance D
  - the parameters V_S' and V_S are under the user's control and form the strategy
- The above model allows for reasoning about the strategy users choose when facing task constraints, which is in contrast to Fitts' law, which assumes users will achieve a fixed level of accuracy

Interpreting Input

Levels of interpretation of user input

1. **Biomechanical models of human movement**
   - Generative models (can generate human-like behavior)
   - Richer analytical ability, such as potential to predict fatigue or energy expenditure for human movement sequences
2. **Control theoretic models of human motor control**
   - Predictive and mechanistic explanations of observed human performance
   - Seldom used in practice; one example: transfer functions for continuous pressure-sensitive scrolling buttons on mice (e.g. IBM ScrollPoint)
3. **Information theoretic models of human information processing capacity (for example, Fitts' law)**
   - Predictive models (typically *time*)
   - No explanation of governing mechanisms
4. **State diagram models of interaction**
   - No predictive ability
   - Used to design systems

Example: bubble cursor

- How to optimize using Fitts' law:
  - Decrease the distance *D* to targets
  - Increase the width *W* of targets
- The bubble cursor does the latter
- The idea is to dynamically resize targets in relation to nearby targets such that only one target is available at any time
- Increases **bandwidth**

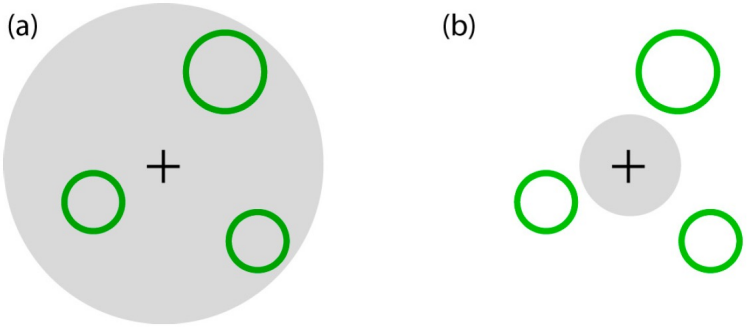


Figure 3. Effect of size on performance of area cursors. (a) A large area cursor tends to encompass multiple targets, thus completely negating its benefits. (b) A small area cursor does not always capture a target, thus reducing its benefits.

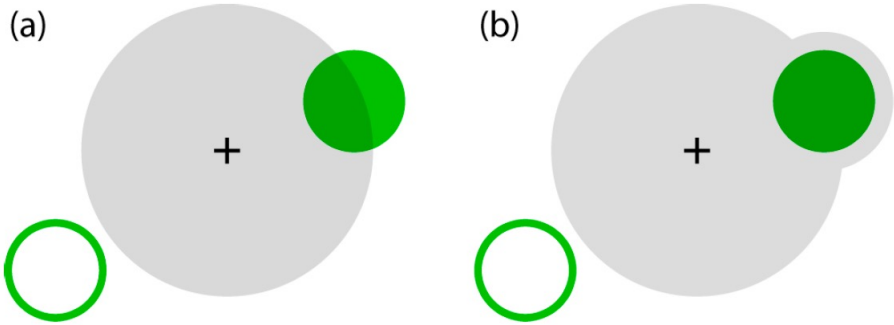


Figure 4. Bubble cursor. (a) Cursor size is dynamically adjusted such that only one target is captured at any time. (b) Cursor is morphed to envelop the target when it is not completely contained by the main bubble.

Grossman, T. and Balakrishnan, R. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. ACM Conference on Human Factors in Computing Systems* (CHI 2005). ACM Press: 281-290.

Example: keyboard optimization 1/2

- Idea: use simulated annealing to identify an optimized keyboard that minimizes the objective function (Equation 2: right)
- Use Fitts' law as fundamental component of objective function
- Use Equation 2 as the energy function; gradually lower temperature until a near-optimal keyboard layout is found that minimizes the objective function

According to Fitts' law (Figure 2), the time to move the tapping stylus from one key i [1] to another j for a given distance (D_{ij}) and key size (W_j) is

$$MT = a + b\, Log_2(D_{ij}/W_j + 1), \qquad (1)$$

where a and b are empirically determined coefficients. In order to be able to make comparisons to the results in [11], we choose the same values a = 0, b = 1 / 4.9. In other words, we consider the Fitts' index of performance (IP) [4] to be 4.9 bits per second. We will return to this choice of this parameter later.

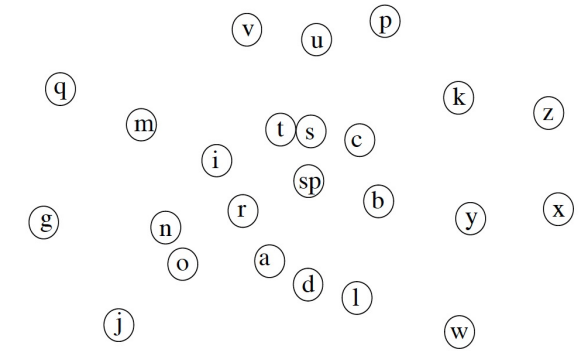
If the frequency of letter j to follow letter i (digraph i-j) among all digraphs is P_{ij}, then the mean time in seconds for typing a character is:

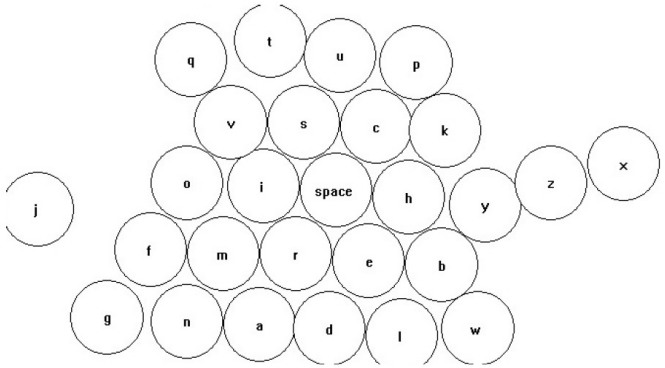
$$\bar{t} = \sum_{i=1}^{27} \sum_{j=1}^{27} \frac{P_{ij}}{IP} \left[Log_2 \left(\frac{D_{ij}}{W_i} + 1 \right) \right] \qquad (2)$$

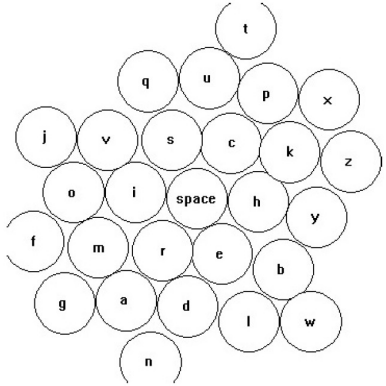
Zhai, S., Hunter, M. and Smith, B.A. 2000. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proc. 13th Annual ACM Symposium on User Interface Software and Technology* (*UIST 2000*). ACM Press: 119–128.

Example: keyboard optimization 2/2

- The optimisation landscape is a valley with many near-optimal keyboard layouts
- Final layout (after some minor layout editing to make the layout practical) results in an estimated ~43 word-per minutes (wpm)
- Having all keys on the same spot results in 95 wpm (estimated)
- Assuming the next key is always adjacent result in 59 wpm (estimated)
- Final design can use ancillary keys to pad the layout to make it fit a rectangular window for deployment to a mobile device







Zhai, S., Hunter, M. and Smith, B.A. 2000. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proc. 13th Annual ACM Symposium on User Interface Software and Technology* (*UIST 2000*). ACM Press: 119–128.

Binary Classification of User Input

Binary classification of user input

- In practice, input is often classified in a binary fashion: “Did the user push the button or not?”
- Some reasons:
  - User interface action sequences are often modelled as state diagrams where an action is a transition from one state to another
  - It is easier to design a binary decision-based system compared to a system that must relay additional information, such as posterior distributions, etc.
  - User interfaces are largely event-driven; this means that at some point the system has to make a decision on whether an event has occurred or not in order to trigger a corresponding action
  - Software architecture is greatly simplified if input is compatible with established software development norms

Confusion matrix

	Predicted 0	Predicted 1	
Actual 0	30	20	50
Actual 1	10	40	50
	40	60	100

Confusion matrix

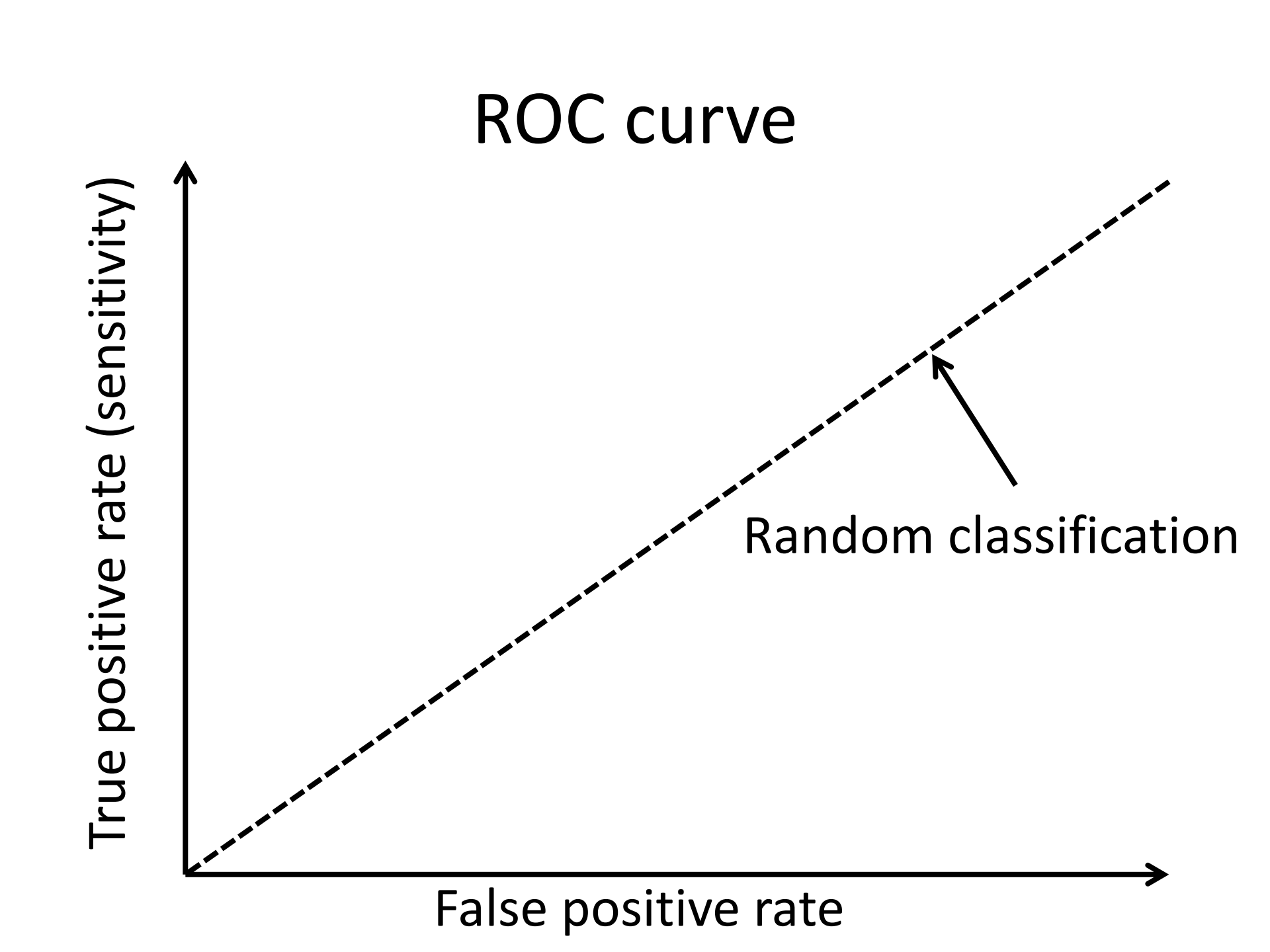
	Predicted 0	Predicted 1	Margin
Actual 0	30	20	50
Actual 1	10	40	50
	40	60	100

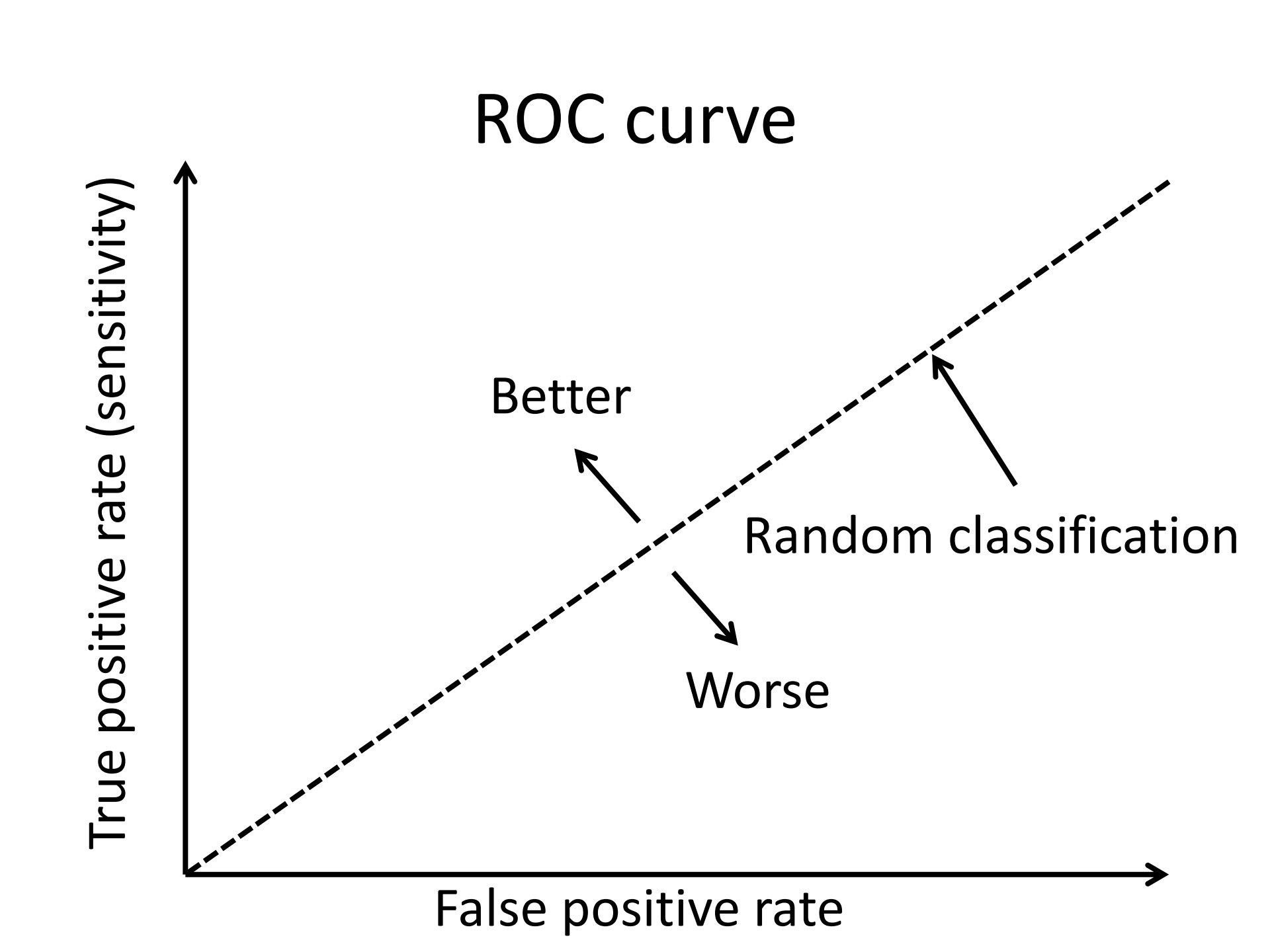
Metrics

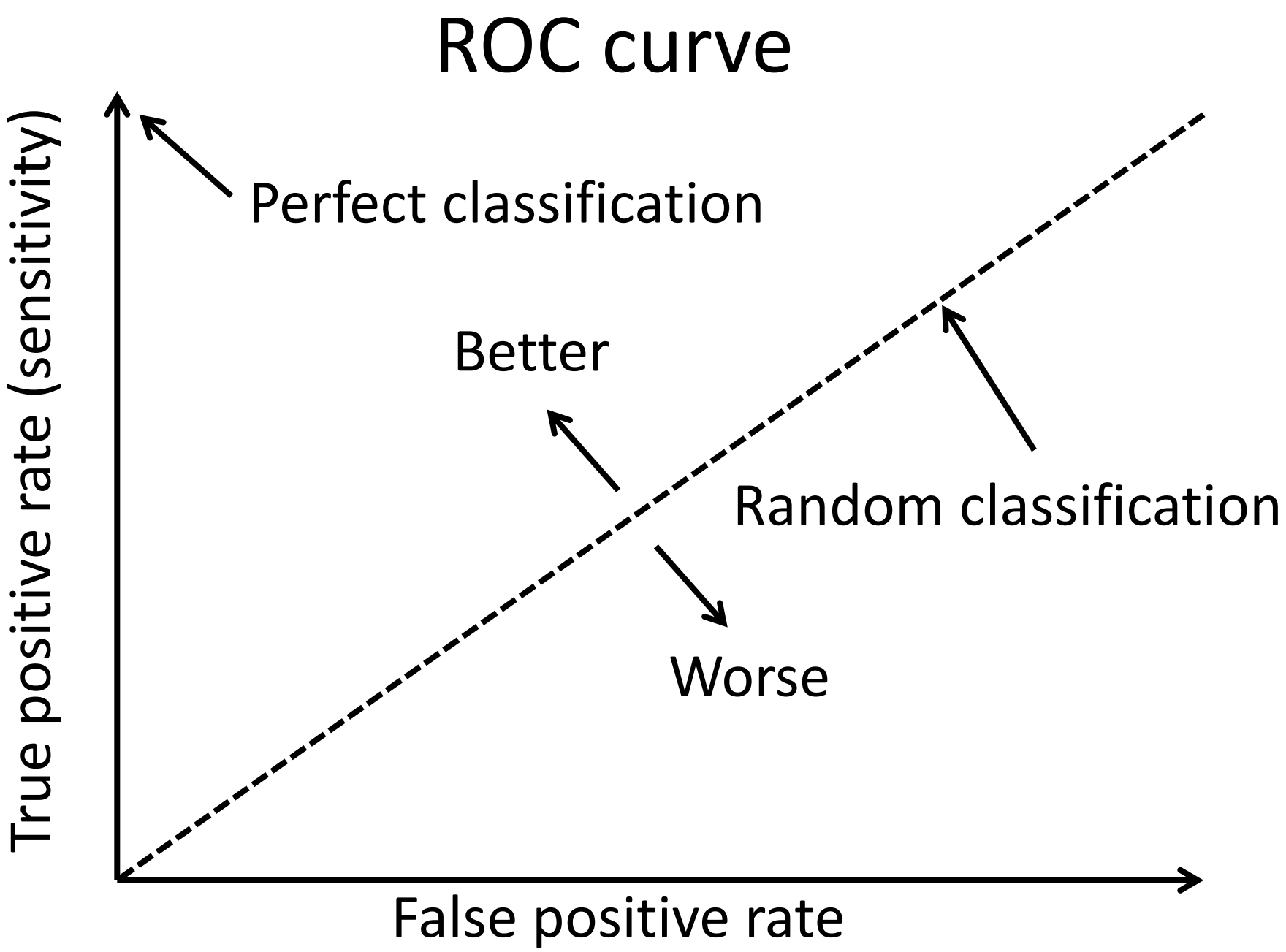
- True positive (TP): correct identification
- True negative (TN): correct rejection
- False positive (FP): incorrect identification
- False negative (FN): incorrect rejection

- True positive rate: TP/(TP + FN) **Sensitivity**
- False positive rate: FP/(FP + TN) **False alarm rate**
- Accuracy: (TP + TN)/(TP + TN + FP + FN)

ROC curve


ROC curve
True positive rate (sensitivity)
Random classification
False positive rate


ROC curve
True positive rate (sensitivity)
Better
Random classification
Worse
False positive rate


ROC curve
True positive rate (sensitivity)
Perfect classification
Better
Random classification
Worse
False positive rate

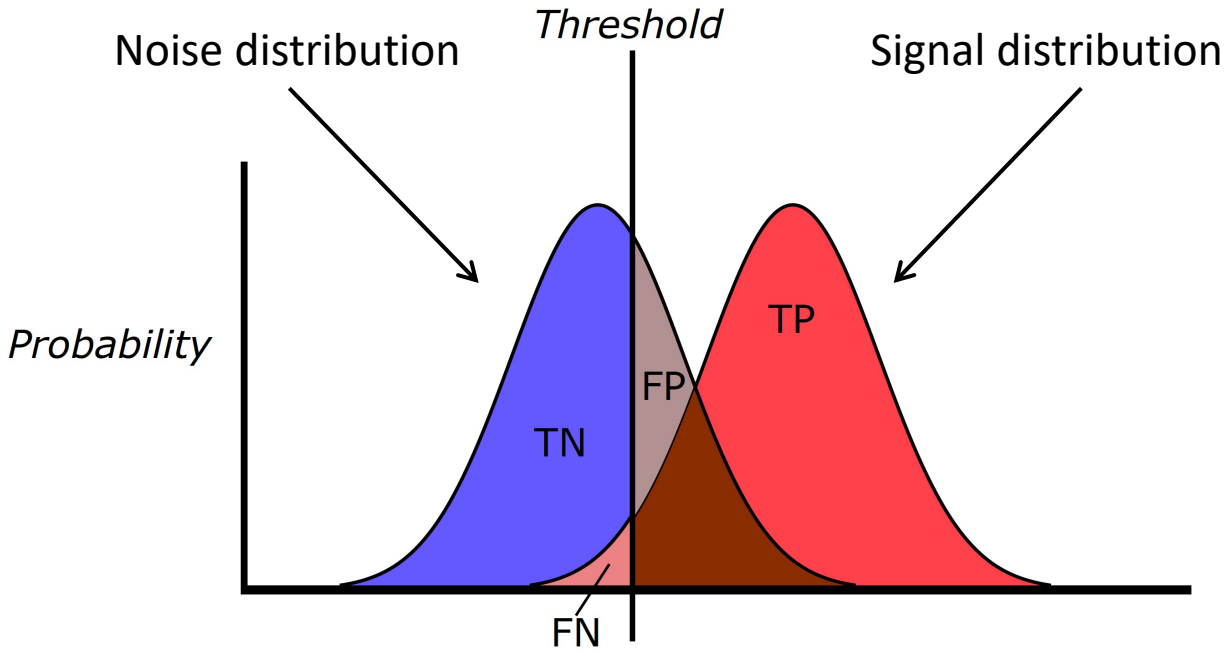
Signal detection theory

- Signal detection theory is the investigation into human ability to detect signal (stimuli) in the presence of noise
- The typical task is to expose the participant to trials where a stimulus is either present or not present and the participant is instructed to state at each trial whether the stimulus was present
  - This results in a confusion matrix categorizing the four possible outcomes as either a *Miss*, *Hit*, *Correct Rejection* or a *False Alarm*
  - It also allows the calculation of the **sensitivity index** (often called *d*'), a dimensionless statistic defined as:

$$d' = \frac{\mu_S - \mu_N}{\sqrt{\frac{1}{2}(\sigma_S^2 + \sigma_N^2)}}$$

  - where μ_S, μ_N, σ_S, and σ_N are the corresponding means and standard deviations of the signal and noise distributions
  - Assumes normally distributed signal and noise distributions with equal variances
  - Can be estimated from an observed hit rate and false alarm rate by subtracting the *Z* score of the hit rate from the *Z* score of the false alarm rate
  - A higher *d*' indicates better ability in discriminating the signal in the presence of noise

Illustration: signal and noise distribution with a threshold



True positive rate and false alarm rate: design implications

- A high true positive rate is imperative for two reasons:
  1. Efficiency in using the system to carry out actions
  2. It results in high perceived usability as it induces a sense of agency in the user (agency captures the degree of user perception that a system outcome was the result of their actions)
- A high false alarm rate is, on the other hand, highly problematic:
  1. It results in the system triggering actions that are unintended by the user: users' sense of agency is reduced; frustration increased
  2. It is typically more costly to correct errors, as unintended actions must first be undone and may, in addition, appear perplexing to the user
  3. It reduces the user's trust in the system
- As the ROC curve makes explicit, a binary classification system is at a particular operating point, balancing true positive rate and false alarm rate
  - Design needs to consider the utility (cost/benefit) of potential operating points as it is unrealistic to expect perfect classification

Traps and Tactics when Designing for Uncertainty

Naïve belief in machine learning

- Consider a user tracking activities via a set of displays
  - Each display continuously fed sensor data
- The task of the user is to detect some objects of interest according to certain parameters (suspicious behaviour, etc.)
- Possibly this task can be aided by a machine learning algorithm trained to identify objects of interest and call for user attention (e.g. visually highlighting objects)
- The problem is this algorithm will not be perfect, it will always form an operating point on a ROC-curve
  - For objects of interest that are critical to detect, the algorithm's operating point on the ROC-curve must then move to a point with a very high true positive rate
  - However, this inevitably results in a high false positive rate; giving rise to distractors that consume the user's attention
- Such a system easily introduces two problems:
  - Problem 1: the machine learning algorithm overloads the user with false positives
  - Problem 2: the user may, partially due to cognitive overload, stop scanning for objects on their own, which has implications if the algorithm's true positive rate is less than 100%

Naïve belief in confidence scores

- Machine learning systems are often able to quantify the uncertainty around their decisions
  - There are typically called confidence scores
- Such scores can be used to for example underline words outputted by a speech recogniser that have a low confidence score
- Such a solution could possibly help users identifying words that are misrecognised
- The problem is that such a solution tends not to work well for two reasons:
  1. Confidence scores are not highly indicative of words that are actually misrecognised (the system's belief in an output is generally high regardless)
  2. Once a system starts underlining words users tend to **only** look at the underlined words
  - However, the non-underlined words will also have errors since confidence scores are imperfect
  - The net effect is that underlining words typically result in no performance gain whatsoever

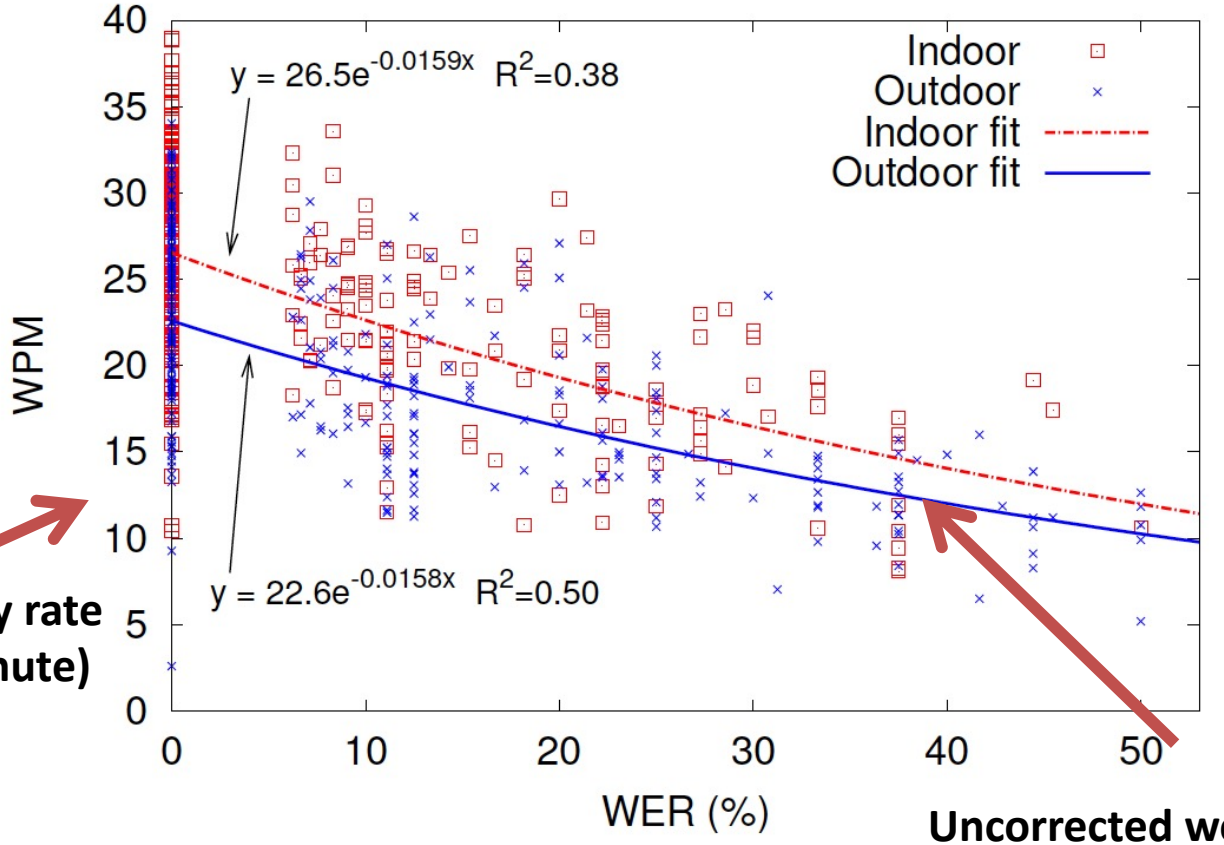
Vertanen, K. and Kristensson, P.O. 2008. On the benefits of confidence visualization in speech recognition. In *Proceedings of the 26th ACM Conference on Human Factors in Computing Systems (CHI 2008)*. ACM Press: 1497-1500.

Lack of fallback method

- Any system relaying on uncertain inference of user intentions risk reaching a situation in which it is impossible or unviable for the user to interact with the system
- The most famous example is a study by IBM in the late 1990s on speech recognition and user's use of "speech repair" to fix speech recognition errors with speech
  - Such speech repairs were frequently misrecognised, leading to further errors in the text that the user had to fix
  - Such cascades could go down several levels before the user gave up
- A fallback method provides a method for the user to escape such cycles, which often only arise at certain malign operating points

Karat, C.M., Halverson, C., Horn, D. and Karat, J., 1999, May. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 568-575).

Example: touchscreen correction fallback for a speech recognition system



Corrected entry rate (words-per-minute)

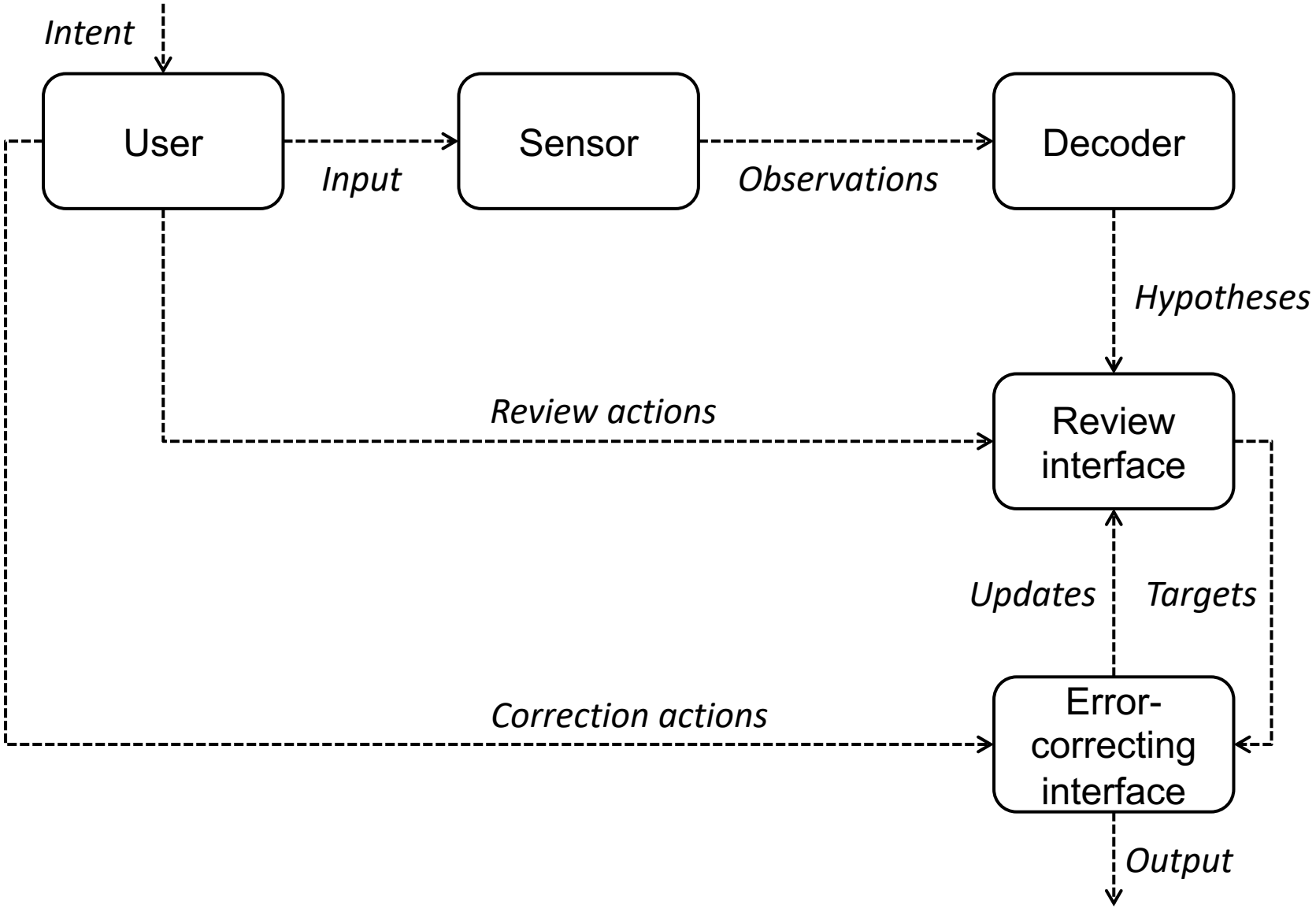
Uncorrected word error rate

Vertanen, K. and Kristensson, P.O. 2009. Parakeet: a continuous speech recognition system for mobile touch-screen devices. In Proceedings of the 14th ACM International Conference on Intelligent User Interfaces (IUI 2009). ACM Press: 237-246.

Avoiding user interface errors

- UI errors are **inevitable** and **noise is pervasive**
  - Human neuromuscular system
  - Human cognition
  - Sensor readings
  - Choice of models
  - Modelling assumptions
  - Data underpinning models
  - Etc.
- UI errors can be counteracted:
  - By allowing users to provide more information than needed (**redundancy**; analogous to error correcting codes)
  - By providing users with ways to easily **identify and rectify errors**
  - By allowing users to **regulate their uncertainty** (for instance, by allowing users to push harder the more certain they are that they want to trigger a particular touchscreen button on pressure-sensitive screen)
  - By choosing **appropriate models of user interaction** (for instance, Gaussian Processes are known to better model target acquisition)

Uncertain input pipeline



The verification-validation trap with sensors and machine learning

- Step 1: Device sensors fed sensor data from lab environment with people simulating actual activity, data split into development, training and testing data
- Step 2: Kitchen sink machine learning evaluation on development data, results in selection of highest performing sensor-inference combination on development data
- Step 3: Verification on separate training-testing data, sensor-inference combination succeeds in verification
- Step 4: Deployment: sensor-inference combination **fails in validation**

Noise mitigation strategies

- Sensor fusion (redundancy)
- Isolating noise sources
- Redesign of workflow to involve a second-pass with alternative sensing solution, possibly human (for example, crowdsourced)
- Redesign of function model to reduce exposure to noise
  - Example: SNPS: reduce fall damage in hospital
  - Solution 1: Fall detector sensor with wearable device (lots of noise to worry about)
  - Solution 2: Redesign the hospital environment to reduce fall risk (remove slopes, move furniture, provide railings, etc.)
- Provide fall-back solution
- Avoid reliance on accurate sensing
- Adapt model online from sensor data
- Distribute local models and identify a best-fit local model to adjust prior in model to improve sensing (example: speech recognition, acoustic signal is matched to best existing model on server to improve recognition)
- Isolate system from external noise sources (if possible)

Automation

Function automation

- Objective: a design that automates some aspect of one or several functions
- Steps:
  - Step 1: Arrive at a functional architecture
  - Step 2: For each function, identify the **type** of automation (acquisition, analysis, decision, action or adaptive)
  - Step 3: For each function, identify the **level** of automation (from no automation to fully automatic)
  - Step 4: Apply primary evaluation criteria: what are the **human performance consequences?**
  - Step 5: Arrive at suitable initial types and levels of automation
  - Step 6: Apply secondary evaluation criteria: how **reliable is automation** and what are the **costs for decisions/outcomes**
  - Step 7: Arrive at final types and levels of automation

Automation: design problem

- A human operator is using a complex system that requires integrating information from dynamic and possibly heterogeneous sources in order to make a decision to achieve a system goal efficiently and safely
  - Example 1: A submarine operator must make a decision on undersea mine detection signals from sonar sensors
  - Example 2: An air-traffic control (ATC) operator must continuously scan several displays showing real-time ATC information and make decisions on which instructions and what information to communicate to pilots and other ATC operators
  - Example 3: A stock analyst is monitoring several sources of real-time information to make decisions about whether to buy or sell stocks
- **System design problem**: Which system functions should be automated and to what extent should they be automated?

Automation: definition

- "…a device or system that accomplishes (partially or fully) a function that was previously, or conceivably could be, carried out (partially or fully) by a human operator"
  - R. Parasuraman and V. A. Riley. 1997. Humans and automation: use, misuse, disuse, abuse. *Human Factors* (39): 230–253.

Function allocation approach

- Strategy 1: maximum automation
  - Each function that can be automated is allocated to a machine
  - Driver: increase efficiency and/or reduce cost
  - Major strategy in automation
  - Implication 1: human operator left with functions that the designer finds hard, expensive or difficult to automate; thus this strategy defines the roles and and responsibilities of human operators in terms of automation
  - Implication 2: as automation approaches a maximum, the role of the human operator becomes more critical (for example: automated errors magnify until human intervention); famous reference: Bainbridge, L. 1983. Ironies of automation. *Automatica* **19** (6): 775–779.
- Strategy 2: Each function in the function model should be assigned to the most capable agent (human or machine)
  - Difficult to carry out in practice
- Strategy 3: Each function is allocated to maximise economical efficiency
  - Requires accurate modelling, which may be very difficult to achieve
- The principle of **human-centred automation** maintains a human has final control (the so-called authority problem), which may not always hold for the above strategies

Types and levels of automation approach

- A popular provides a framework for system design problem using a model of **types** of automation and **levels** of automation (Parasuraman et al. 2000)
  - Human performance consequences on choices of types and levels of automation form primary evaluation criteria
  - Secondary evaluation criteria include automation reliability and cost of decisions/outcomes
- Primary reference used in the remaining of this section: Parasuraman, R., Sheridan, T.B. and Wickens, C.D., 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, *30*(3), pp. 286-297

Levels of automation

Automation level		
High	10	The computer decides everything, acts autonomously, ignoring the human.
	9	The computer informs the human, only if it, the computer, decides to.
	8	The computer informs the human only if asked.
	7	The computer executes automatically then necessarily informs the human.
	6	The computer allows the human a restricted time to veto before automatic execution.
	5	The computer makes a decision to perform automatic execution if the human approves.
	4	The computer suggests one alternative action.
	3	The computer narrows the selection of alternative actions down to a few.
	2	The computer offers a complete set of decisions/action alternatives.
Low	1	The computer offers no assistance: the human must take all decisions and actions.

Slightly paraphrased from Parasuraman, R., Sheridan, T.B. and Wickens, C.D., 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, *30*(3), pp. 286-297

Types of automation

- By viewing a human operator as a simple four-stage model of human information processing, we arrive at the following four stages:

1. Sensory processing
2. Perception/working memory
3. Decision making
4. Response selection

- This four-stage model of human information processing has the following corresponding types of system functions that can be automated:

1. Acquisition automation
2. Analysis automation
3. Decision automation
4. Action automation

- In addition, the following emergent type of system function automation exists, which encompasses a combination of the above types of automation:

- Adaptive automation

Parasuraman, R., Sheridan, T.B. and Wickens, C.D., 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, *30*(3), pp. 286-297.

Acquisition automation

- Sensing and registering input data, corresponding to the first human information processing stage
- Low level automation: assistance of sensor adjustment, such as mechanically moving a radar sensor to lock on detected targets
- Moderate level of automation: organising information according to some criteria, such as a priority list or by highlighting information based on criteria (either static or dynamic criteria)

Analysis automation

- Automation of information analysis involves inferential processes
- Low level automation: extrapolate/predict data over time (for example, trend prediction of an industrial plant based on sensor reading history)
- Moderate/high level of automation: integration of multiple sources/input variables (for example: displays with emergent perceptual features, such as a optical see-through display with polygonal information overlaid (such a landing strip)
- High level of automation: context-dependent summaries of data

Decision automation

- Deciding and selecting appropriate actions among decision alternatives augmenting or replacing human decision making
- Examples include: route planning and route adaptation (for instance, to avoid bad weather), systems providing medical diagnosis support
- The difference between analysis automation (inference) and decision automation is that decision automation means the system must make implicit and/or explicit assumptions but costs and values inherent in all decisions
- The levels of automation show the various levels of automation that can be applied to decision automation
  - Example 1: Decision automation in avoiding air-craft ground collisions is at level 4 of automation as a decision is recommended but the pilot can choose to ignore it
  - Example 2: An automatic ground collision avoidance system would be at level 7 as automation assumes control if the pilot does not

Action automation

- Action execution of an action choice
- Various levels of machine execution automating human actuation by hands, etc. or commands by voice
- Common examples include automating a series of manual steps into a single key press (as opposed to several key presses)
- Photocopier example: manual sorting, automatic sorting, automatic collation and automatic stapling
- High levels of automation include automatic landing systems in air-crafts and systems that track user interaction and automatically execute context-specific sub-tasks

Adaptive automation

- Type and level of automation can be allowed to vary depending on context, such as situational demands
- For example, air-craft navigation automation can be either high or low in a critical situation depending on how quickly the joint system (air-craft + crew) must react to an event in order to avoid a catastrophic scenario

Primary evaluation criteria: human performance consequences

- **Mental workload**
  - Automation can reduce mental workload, for example:
    1. Automatic organization of information to make it easier to understand
    2. Facilities to avoid search or communication (such as showing pilot data directly to the ATC controller)
    3. Collating and presenting information sources together
    4. Prediction facilities can sometimes reduce workload (such as predictive flight path displays in aircrafts which have been shown to reduce pilots' mental workload
  - …and increase mental workload, for example, systems that are difficult to initiate or engage, difficult to understand or require extensive physical operations
- **Situation awareness**
  - Automation may reduce operators' awareness of the system and dynamic (and/or emergent) properties of the system
  - Humans tend to be less aware of changes in a system or environment if those changes are under the control of another agent
  - Automatic decision making based on sources of information may make the operator less aware of the information sources as the operator is not actively engaged in evaluating the information as a basis of the decisions
- **Complacency** (over-trust in automation)
  - Failure of an operator to monitor automation or information sources if automation is highly, but not perfectly, reliable
  - Imperfect automated information analysis (filtering, prediction) is a source of operator errors as operators tend to overly trust an imperfect system
  - Attention-guiding systems, such as automatic cueing systems, can result in operators not paying attention to uncued areas
- **Skill degradation**
  - Automation resulting in operators forgetting how to carry out functions, or skill decay

Secondary evaluation criteria

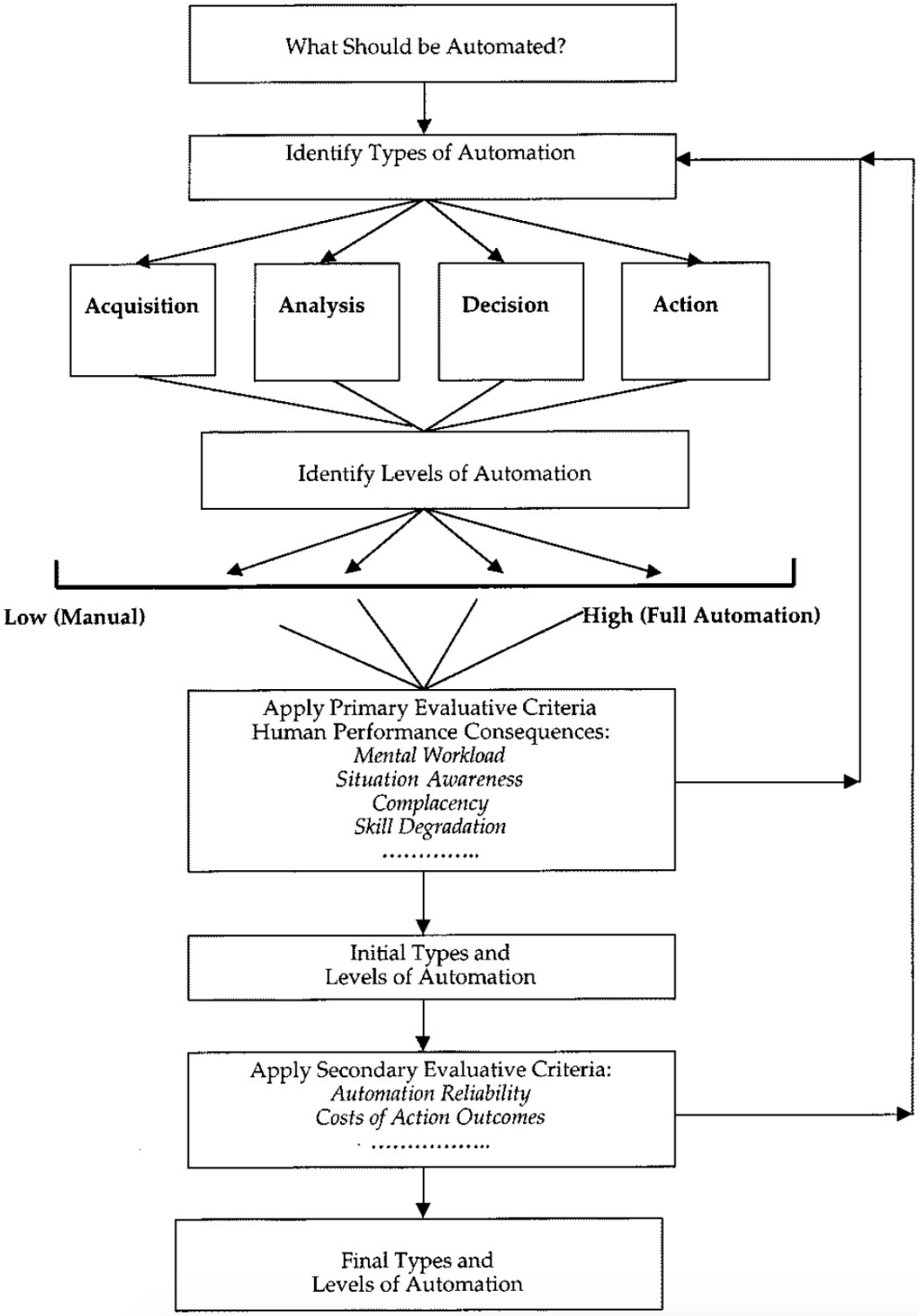
- Automation reliability
  - True positive rate
  - False alarm rate
  - Ability to detect hazardous events reliably
  - Lower reliability may be acceptable if raw data is available to well-trained and alert human operators
  - System that predict the future have inherent errors built in and thus the very real eventuality of prediction errors must be designed in
  - Providing feedback about when automation fails to the human operator helps build an appropriate level of operator trust in the system
- Costs of decision/action outcomes
  - Benefits of automation has to be weighted against possible disadvantages, such as increased mental workload, reduced situation awareness, complacency and skill degradation
  - Risk analysis can be used to analyse the impact of incorrect decisions

    risk = cost of error × probability of error
  - A high level of automation may be considered in high-risk decisions **if** the human operators are given time to respond
    - The cost of these highly costly errors form major evaluation criteria as for the appropriate level of automation

Automation framework

1. Identify **types** of automation
2. Identify **levels** of automation
3. Evaluate based on primary evaluation criteria: human performance consequences
4. Arrive at initial types and levels of automation
5. Apply secondary evaluation criteria: automation reliability and costs of decisions/outcomes
6. Arrive at final types and levels of automation

- This is a **guide** only



Practical example

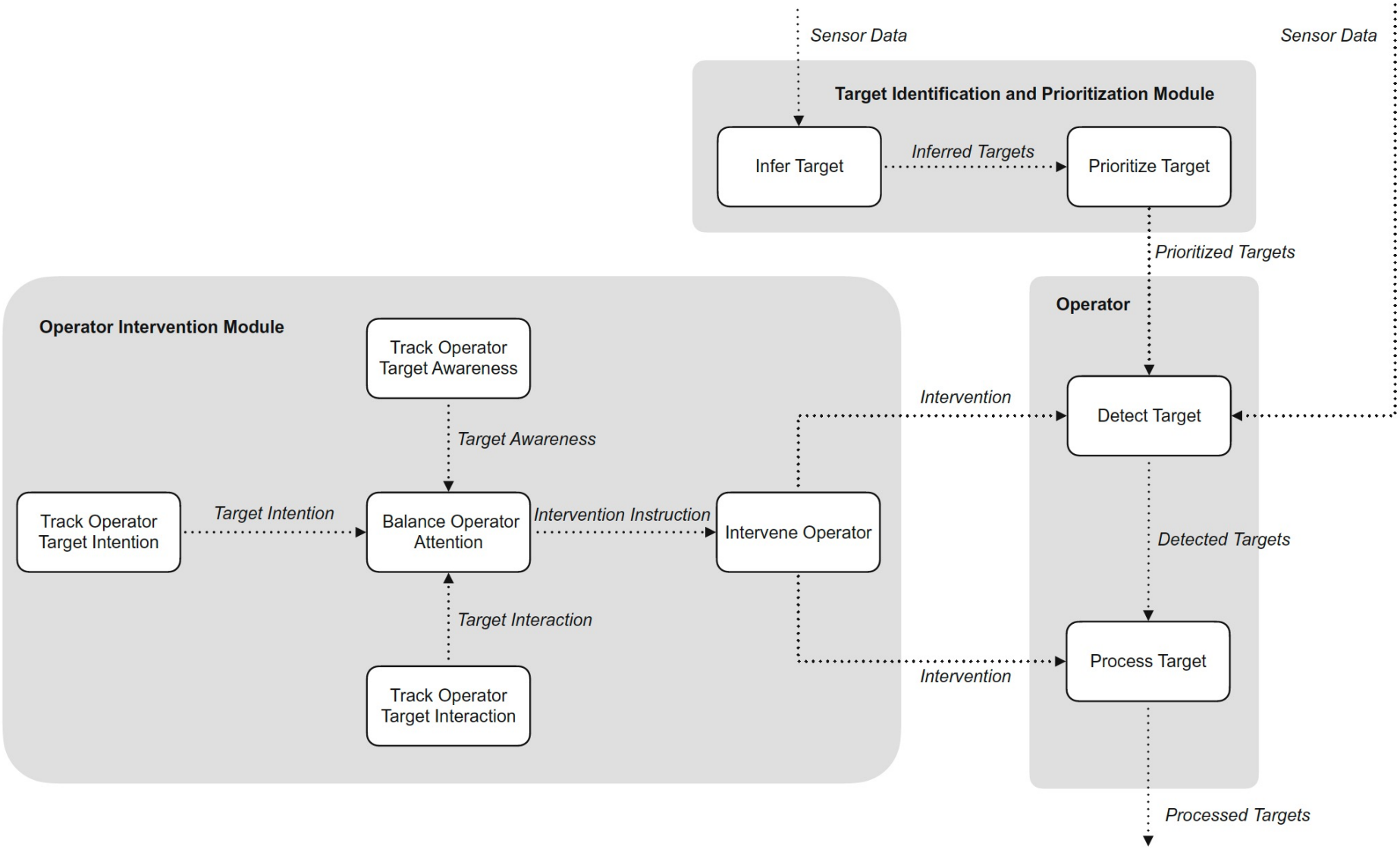
Problem context

- An operator is tasked with monitoring several displays with continuous sensor streams and detect objects of interest and process them
- Example application areas: submarine demining vehicle, CCTV surveillance, manufacturing operations monitoring
- Example SNPS: Devise a means to assist an operator in detecting and processing objects of interest in a continuous sensor stream

Human-in-the-loop AI-assisted continuous monitoring

- Now assume it is known that the cognitive load on the operator is high and thus an operator may struggle to detect all intended objects (targets) among the plethora of distractor targets in the sensor stream
- An AI-assisted system is proposed that automatically detects targets and reveals them to the user
- How do we best automate this task?

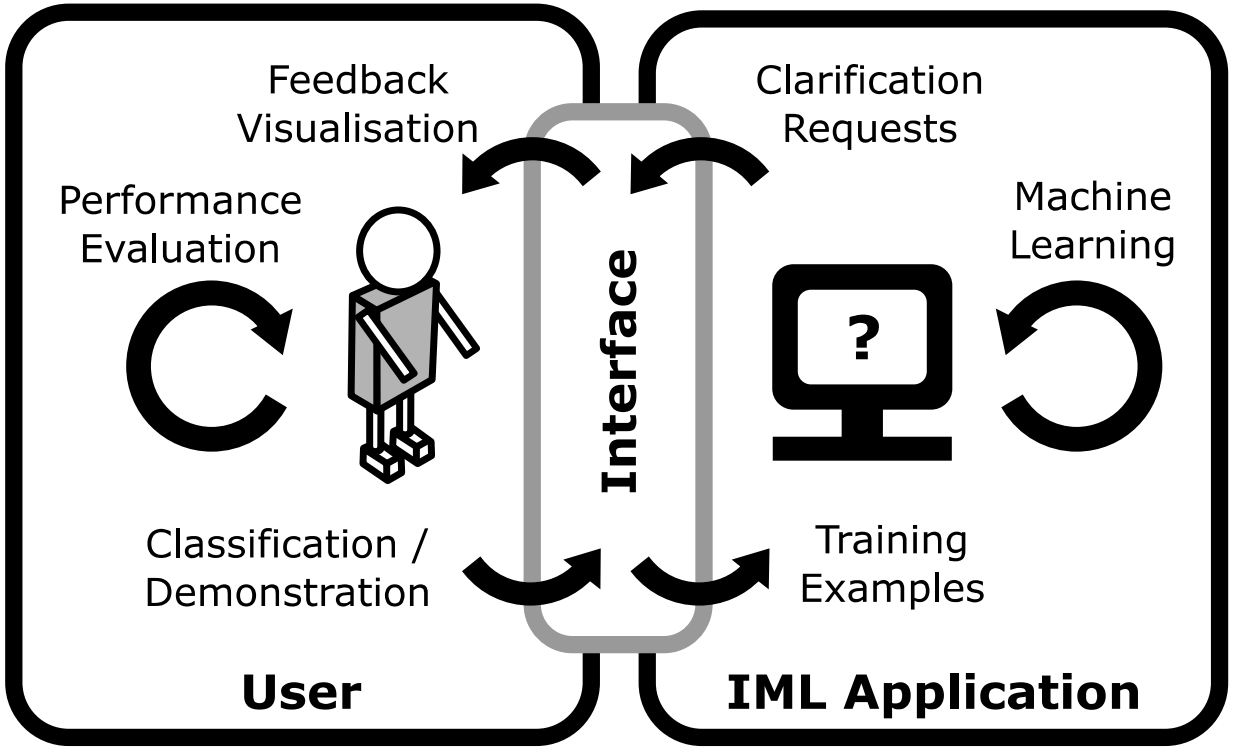
Functional architecture



Nicosia, M. and Kristensson, P.O. 2021. Design principles for AI-assisted attention aware systems in human-in-the-loop safety critical applications. In Lawless, W.F., Llinas, J., Sofge, D.A. and Mittu, R. (Eds.), *Engineering Artificially Intelligent Systems: A Systems Engineering Approach to Realizing Synergistic Capabilities*. Cham: Springer Nature.

Interactive Machine Learning

Definition



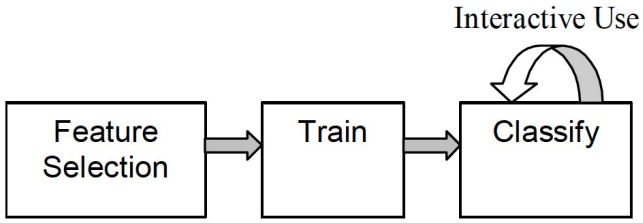
Interactive Machine Learning is an interaction paradigm in which a user or user group refines a mathematical model to describe a concept through iterative cycles of input and review.

Origin of term

- The term **interactive machine learning** was coined in the following research paper:
  - Fails, J.A. and Olsen, D.R. 2003. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (*IUI 03*). ACM Press: 39–45.

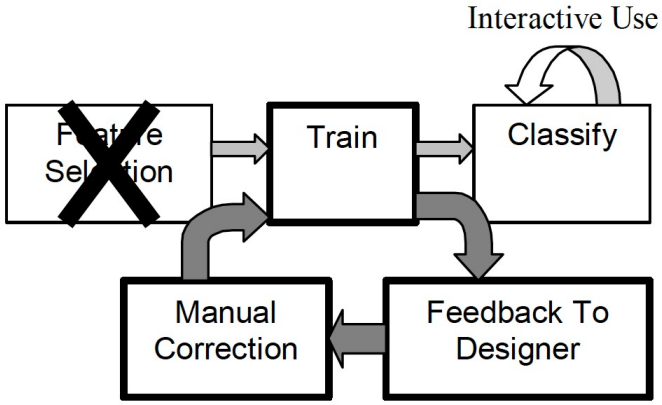
ML vs. IML: classic (2003) view

- ML:



- Perform feature engineering to select features (difficult for non-experts to do, such as user interface designers)
- Train model offline (time intensive, introduces time lag in the design process)
- Use classifier in an interactive setting
- Demands too much technical knowledge from user interface designers

- IML:

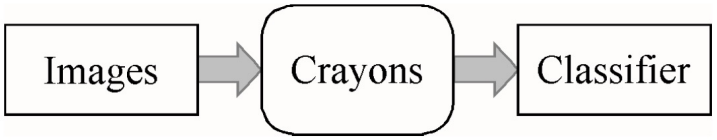


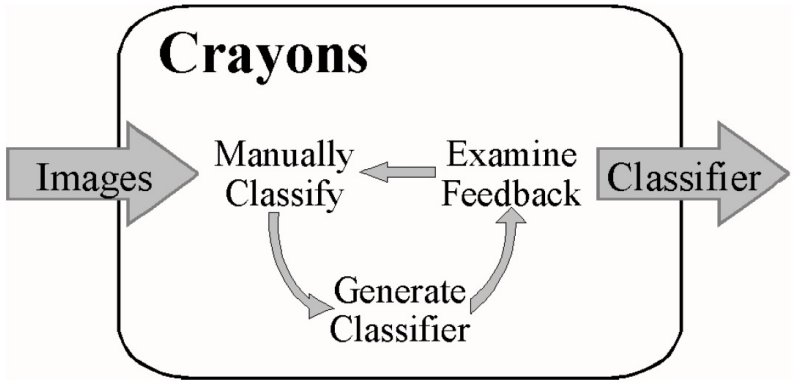
- Feature selection is subsumed into the training (learning) stage
- A large repository of features are fed to the learning algorithm to learn the best features
- The designer rapidly creates training data that correct errors made by the learning algorithm

Fails, J.A. and Olsen, D.R. 2003. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (*IUI 03*). ACM Press: 39–45.

IML (2003) example

- IML was first exemplified with system called *Crayons* with two primary objectives:
  1. Allow the user to quickly create an effective pixel-based image classifier
  2. Allow the user to focus on classification problems rather than image processing or other algorithms
- The system accepts images and outputs image classifiers
- Workflow:
  1. The user imports images into Crayons
  2. The user performs manual classification (labelling)
  3. The system uses manual classification to train a classifier and provide visual feedback on classification accuracy
  4. The user can refine the classifier by adding additional manual classifications
  5. When the user is happy, the user can export a classifier

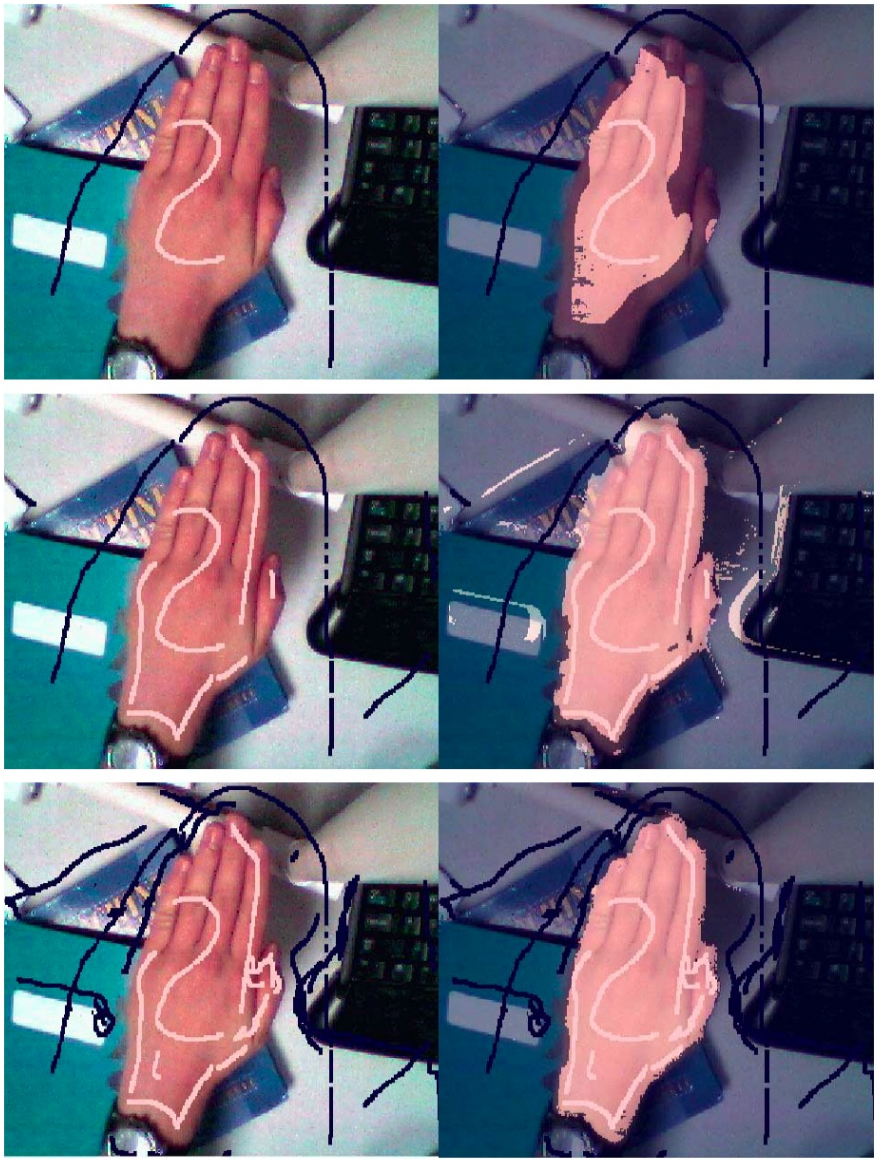




Fails, J.A. and Olsen, D.R. 2003. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (*IUI 03*). ACM Press: 39–45.

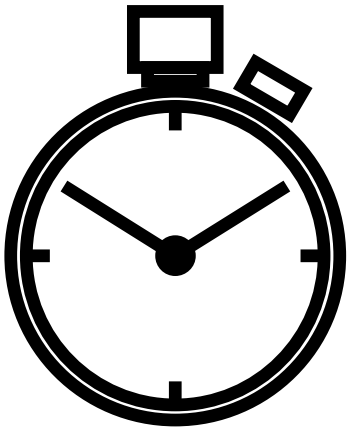
IML (2003) example

- Implementation based on a custom variant of decision trees
- The image to the right shows visual samples of user-system iterations where the user is gradually providing more classification data to the system in order for the system to be able to separate the image of the hand from the background
- The designer is a human-in-the-loop, instructing the classifier of parts of the image that are missed and preventing over-generalisation
- A small experiment in the paper also showed empirical evidence that designers were able to build a better classifier faster using IML than a classic ML baseline



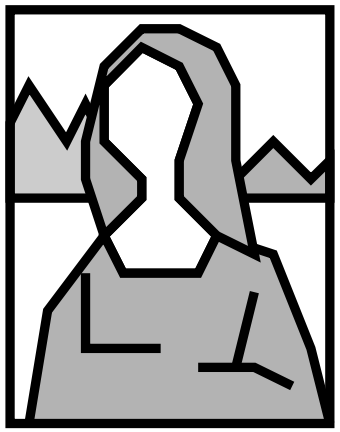
Fails, J.A. and Olsen, D.R. 2003. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (*IUI 03*). ACM Press: 39–45.

Opportunities for accessible IML



Streamline workflows

Directly apply domain expertise



Extend creativity

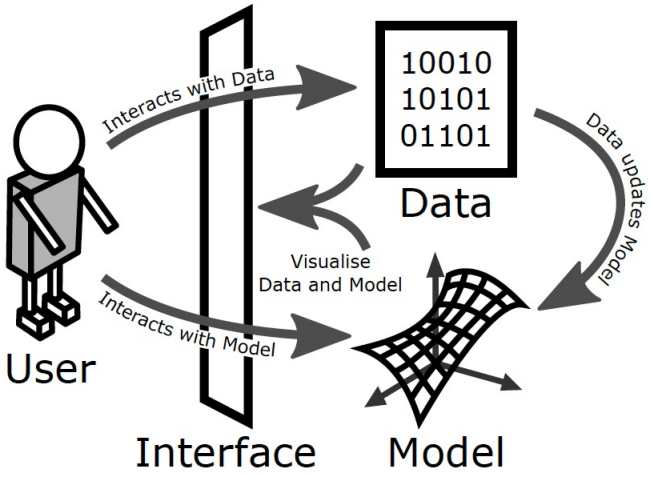
Key challenges*

- Establishing a productive dialog
  - Framing feedback
  - Capturing input
  - Explaining decisions
- Communicating state
  - Representing quality
  - Knowing when to stop

***from an interface design perspective**

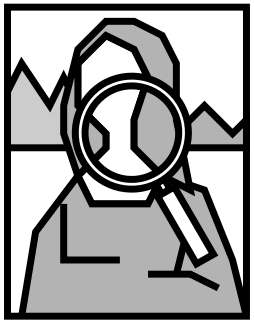
Structure of a generic IML system

1. **User:** A dynamic but potentially unreliable component that provides domain expertise
2. **Model:** A mapping between inputs and outputs based on the system's current understanding of the concept or process the user desires to capture
3. **Data:** The user selects, describes or generates data in order to indicate how the system should behave in response to certain inputs
4. **Interface:** The interface provides bidirectional feedback between the user and the model/data and must provide both input and output mechanisms to allow this functionality



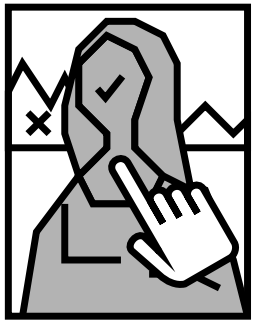
Dudley, J.J. and Kristensson, P.O. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Intelligent Interactive Systems* **8**(2): Article No. 8.

Composition of an IML interface



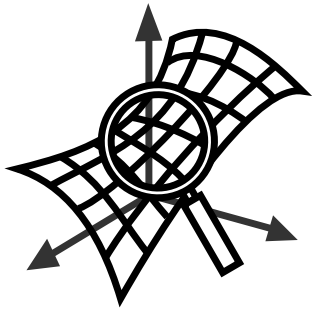
Sample Review

Visualization of output sample(s) to assess how well desired concept operates at the instance level.



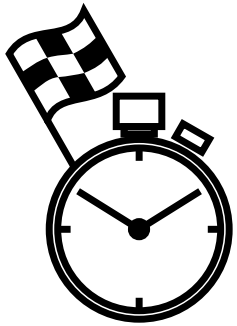
Feedback Assignment

Assignment or correction of labels and/or creation of new samples to improve match with desired concept.



Model Inspection

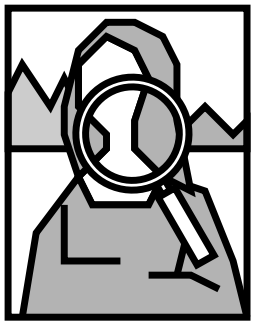
View of overall model quality and coverage to evaluate how well concept is captured.



Task Overview

View of current task status contexualized by coverage of training data and improvement potential relative to cost.

Sample Review



Sample Review

Visualization of output sample(s) to assess how well desired concept operates at the instance level.

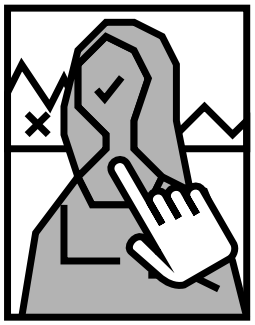
Challenges:

- ❑ Pestering
- ❑ Doubt
- ❑ Irrelevance

Potential solutions:

- ❑ Support assessment of current state of concept
- ❑ Understanding why
- ❑ Avoid overly abstract/condensed representations

Feedback Assignment



Feedback Assignment

Assignment or correction of labels and/or creation of new samples to improve match with desired concept.

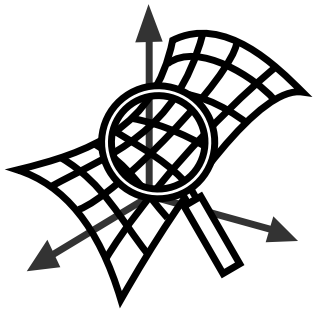
Challenges:

- ❑ Overwhelming users with ‘knobs’
- ❑ Capturing high level attributes with low level features
- ❑ Concept shift

Potential solutions:

- ❑ Avoid constraining the labelling or demonstration process
- ❑ Carefully frame input requests

Model Inspection



Model Inspection

View of overall model quality and coverage to evaluate how well concept is captured.

Challenges:

- ❑ Understanding where error comes from
- ❑ Delivering an overview

Potential solutions:

- ❑ Highlight potential sources of error
- ❑ Provide test methods

Model Inspection

Task Overview

View of current task status contexualized by coverage of training data and improvement potential relative to cost.

Challenges:

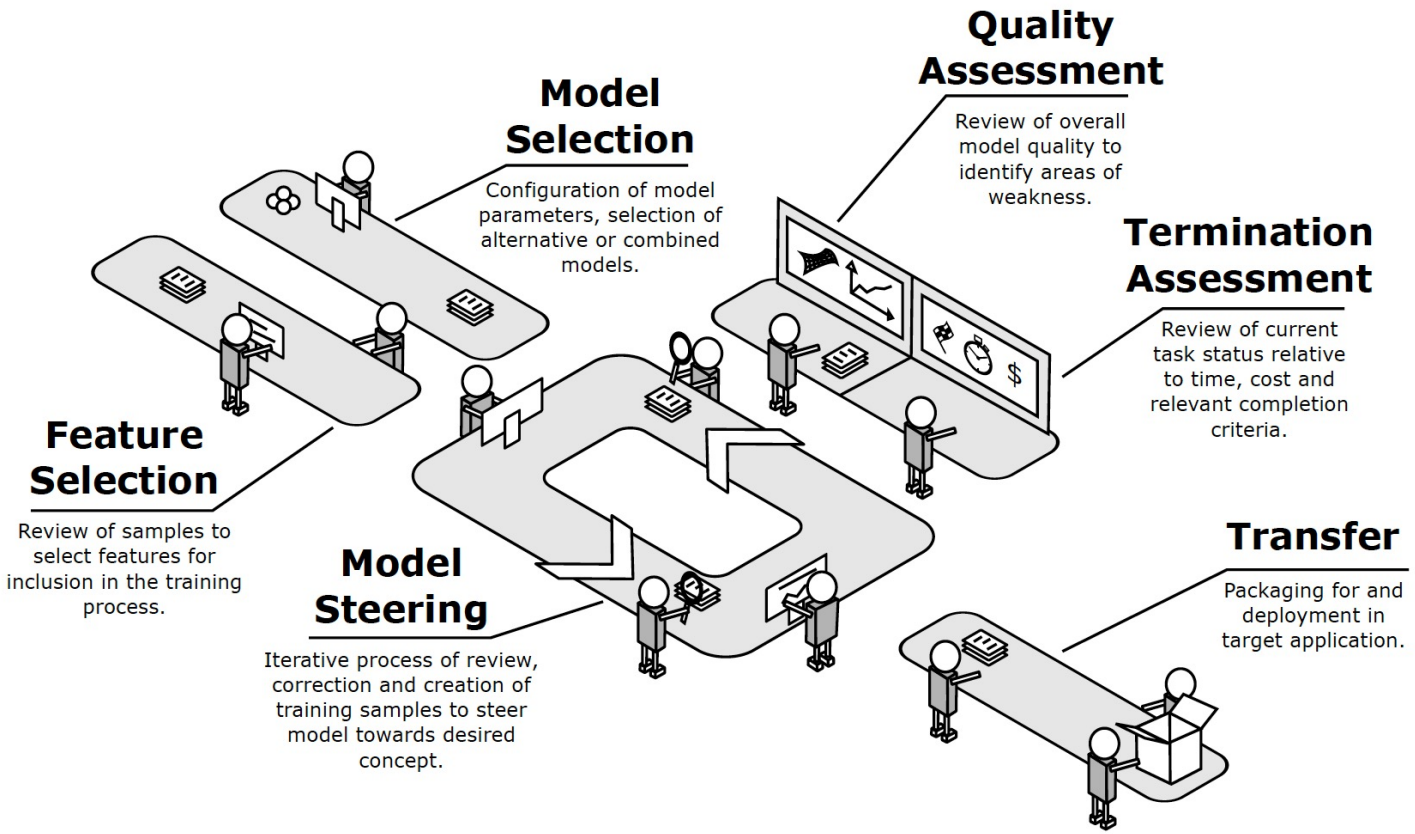
- ❑ Diminishing returns
- ❑ Knowing when to stop

Potential solutions:

- ❑ Provide visibility of the global objective
- ❑ Contextualise with other task considerations

Workflow for IML processes

- **Model steering** is the iterative feedback-review cycles in the centre of the IML process
- **Model selection** and **feature selection** allow the user to initialise or reconfigure the process
- **Quality assessment** and **termination assessment** is a parallel activity to **model steering** that can suspend **model steering** momentarily
- Once a model has been successfully trained then the **transfer** activity is engaged to deploy a developed model into a target application



Dudley, J.J. and Kristensson, P.O. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Intelligent Interactive Systems* **8**(2): Article No. 8.

Solution principles for IML

1. Make task goals and constraints explicit
   - Since users tend to be imprecise and training is open-ended, vital that non-experts are provided with clear task goals
2. Support user understanding of model uncertainty and confidence
   - Non-experts are likely building incorrect mental models of the underlying models in the IML system and are unlikely to understand the nature of the uncertainty that is inevitable in data-driven real-world applications
3. Capture intent rather than input
   - Reduce the uncertainty between a user's intent and a user's noisy input
4. Provide effective data representations
   - Since model steering (reviewing output samples) is essentially a comprehension task, any visualization aid that can improve comprehension is likely to improve performance
5. Exploit interactivity and promote rich interactions
   - More interactivity allow users to fully express their intentions
   - Design interaction to promote understanding, for example, a **mixed-initiative interface** allow a user to probe and engage with intelligent services using principles of direct manipulation
   - Make the most out of the user: allow the user to experiment and reverse actions and in general provide the user with control and autonomy
6. Engage the user
   - For example, showing partial predictions can help the user stay engaged by encouraging them to actively improve prediction quality
   - Reduce effort to interpret model outputs and/or effort to express feedback

Dudley, J.J. and Kristensson, P.O. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Intelligent Interactive Systems* **8**(2): Article No. 8.