Lecture 10. Outlook on AI Dependability and Course Summary

IL2233 Embedded Intelligence

Zhonghai Lu

KTH Royal Institute of Technology

May 12, 2024

Outline

Outlook on AI dependability

- AI dependability issues
- Approaches addressing AI dependability

Deep learning

- Deep learning drives the new wave of AI
- AI dependability: Deep learning can cause deep trouble
- Examples

Image Recognition: ImageNet

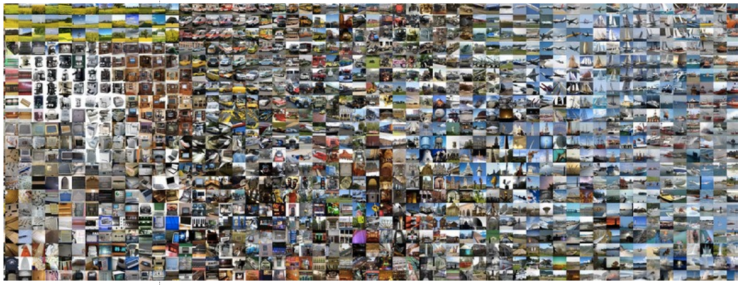


Figure: ImageNet dataset

- 256x256 pixels (color)
- 1000 Classes
- More than 14 million images 1.3M Training 100,000 Testing (50,000 Validation)

Image Recognition

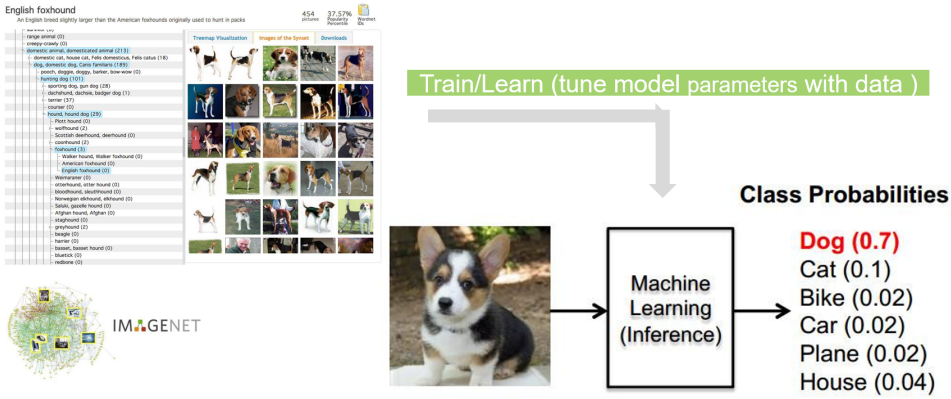


Figure: Recognition from training to inference

Image Recognition

Large Scale Visual Recognition Challenge (ILSVRC) starting from 2010.

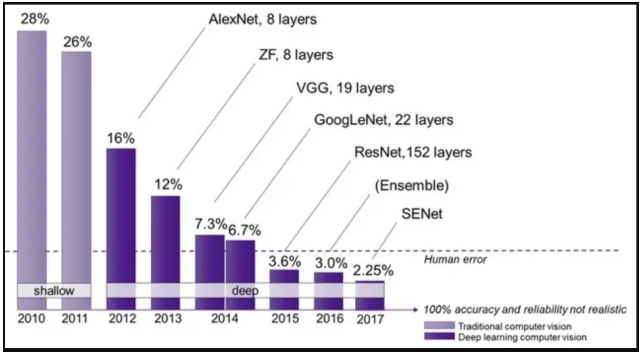
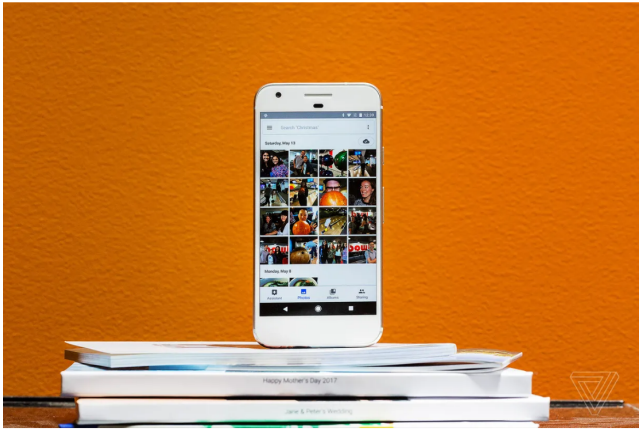


Figure: ImageNet Recognition Challenge Results

https://semiengineering.com/new-vision-technologies-for-real-world-applications/

Google Photos



The AI algorithms in Google Photos sort images by a number of categories. | Photo by Vjeran Pavic / The Verge

Figure: Image Recognition by Google Photos

www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai

Real-life failure examples

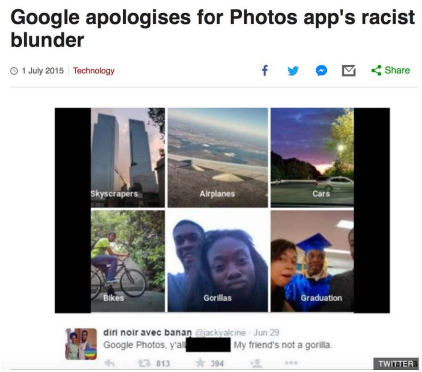


Figure: Google classification mistake (2015)

- Google ‘fixed’ its algorithm by removing gorillas from its image-labeling tech (2018).

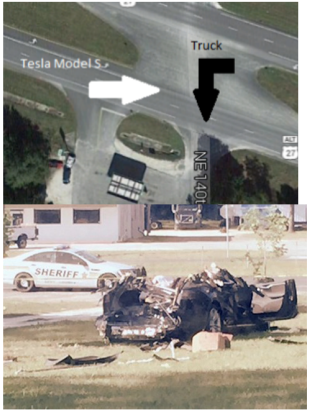


Figure: Tesla S fatal crash, radar/cameras fail to recognize a white car (2016)

How can it go wrong? (1)

- Rotating objects in an image confuses DNNs, probably because they are too different from the types of image used to train the network.

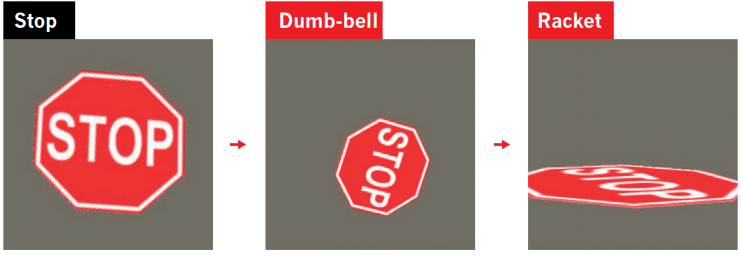


Figure: Stop sign or Dumb-bell, Racket?

Figure: Dumb-Bell (Wikipedia)

How can it go wrong? (2)

- Recognize pattern? Scientists have evolved images that look like abstract patterns — but which DNNs see as familiar objects.

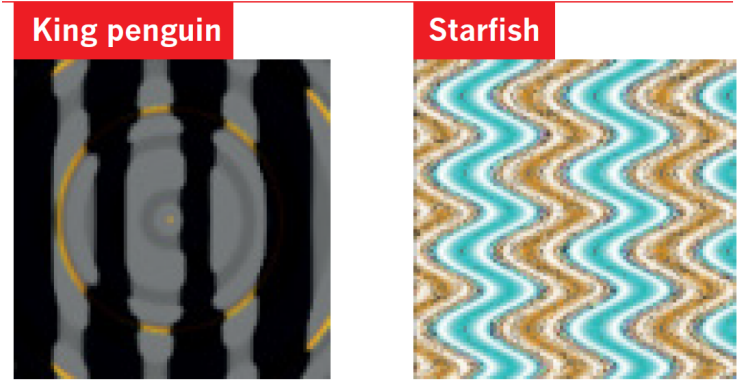


Figure: King penguin, Starfish or just cheating patterns?

Figure: King Penguin (Wikipedia)

Figure: Star fish (Wikipedia)

How can it go wrong? (3)

- Even natural images can fool a DNN, because it might focus on the picture's colour, texture or background rather than picking out the salient features a human would recognize.?

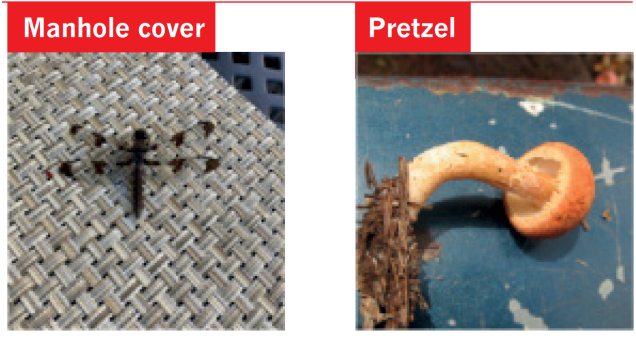


Figure: Mis-classification of natural objects

Figure: Manwhole cover (Wikipedia)

Figure: Pretzel

How can it go wrong? (4)

- These stickers made an AI system read this stop sign as 'speed limit 45'.

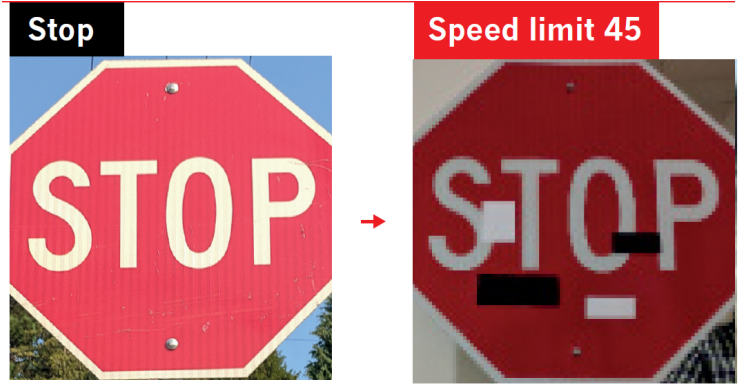


Figure: Stop or Drive?

Figure: Speed limit 45 (Wikipedia)

How can it go wrong? (6)

- Adding carefully crafted noise to a picture can create a new image that people would see as identical, but a DNN sees it as utterly different.

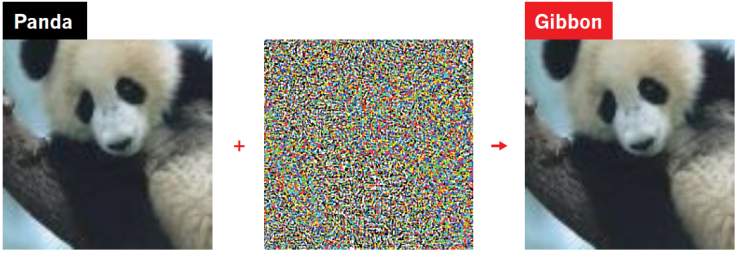


Figure: Panda + Noise = Gibbon ?

Figure: Gibbon (Wikipedia)

How can it go wrong? (7)

- Any starting image can be tweaked so a DNN misclassifies it as any target image a researcher chooses.

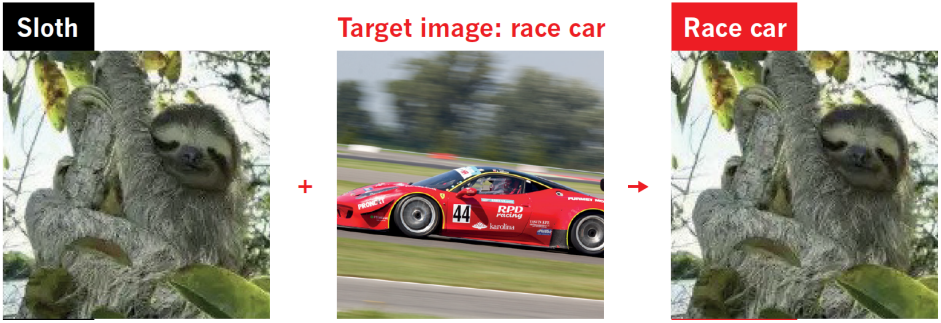


Figure: Sloth + Race car = Race car ?

How can it go wrong? (8)

- State-of-the-art DNNs can increasingly recognize natural images (left panel). However, they can be easily fooled (center).
- Images are produced by evolutionary algorithms (right panel). The dataset the DNN is trained on is ImageNet.

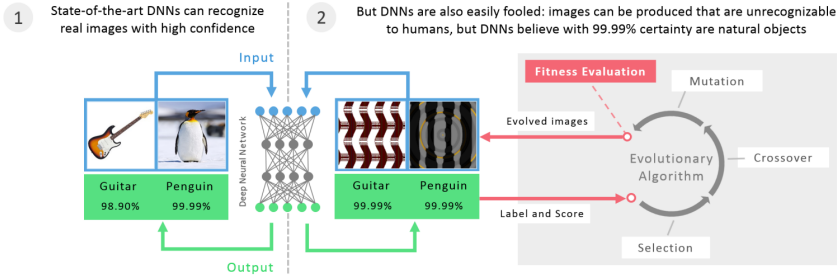


Figure: How easily DNN can be fooled!

Nguyen A, Yosinski J, Clune J. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In Computer Vision and Pattern Recognition (CVPR'15), IEEE, 2015.

Addressing AI dependability

We have seen two broad approaches in addressing the AI dependability challenges.

- Make an AI model more robust, e.g. against fake images in the case of face recognition
  - Generative Adversarial Network (GAN)
- Make an AI model explainable

GAN

- Generative Adversarial Network (GAN) is an old idea arising from the game theory. It was introduced to the machine learning community in 2014.
  - Two neural networks contest with each other in a game (in the form of a zero-sum game, where one agent's gain is another agent's loss)
  - Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). "Generative Adversarial Nets". Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.
- "Generative Adversarial Network— the most interesting idea in the last ten years in machine learning" by Yann LeCun, VP & Chief AI Scientist at Facebook.

GAN

A GAN has three primary components:

- A generator model for generating new data, e.g. fake faces.
- A discriminator model for classifying whether generated data are real or fake, and real data are real.
- An adversarial training process that pits them against each other.

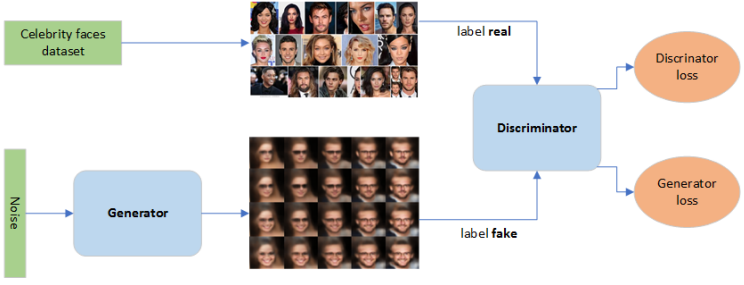


Figure: GAN for celebrity faces

https://towardsdatascience.com/generative-adversarial-network-gan-for-dummies-a-step-by-step-tutorial-fdefff170391

GAN

An iterative adversarial training process:

1. Select a number of real images from the training set.
2. Generate a number of fake images. This is done by sampling random noise vectors and creating images from them using the generator.
3. Train the discriminator for one or more epochs using both fake and real images. This updates only the discriminator's weights by labeling all the real images as 1 and the fake images as 0.
4. Generate a new set of fake images.
5. Train the full GAN model for one or more epochs using only fake images. This will update only the generator's weights by labeling all fake images as 1.

GAN: Train the discriminator

During the discriminator training:

- Ignore the generator loss and just use the discriminator loss, which penalizes the discriminator for misclassifying real faces as fake or generated faces as real.
- The discriminator's weights are updated through error backpropagation.
- Generator's weights are not updated.

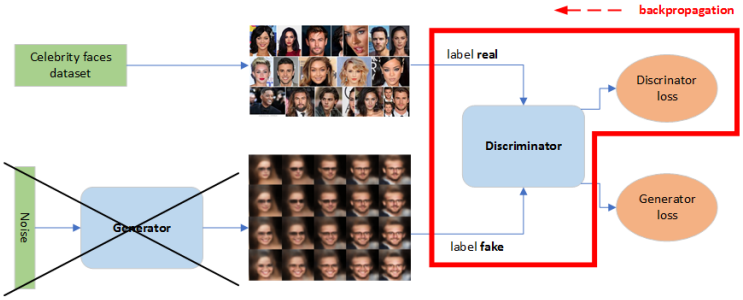


Figure: GAN: Train the discriminator

GAN: Train the generator

During the generator training:

- Use the generator loss, which penalizes the generator for failing to fool the discriminator and generating a face that the discriminator classifies as fake.
- The discriminator is frozen and only the generator's weights are updated through error backpropagation.

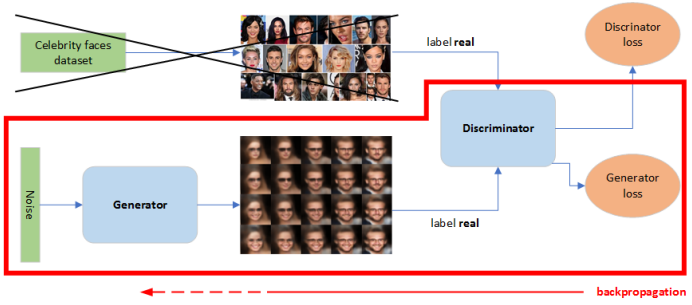


Figure: GAN: Train the generator

Code example, see also "Deep Convolutional GAN tutorial"
https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

GAN: fake images

GANs can create images that look like photographs of human faces, but these faces are fake ones. They don't belong to any real person.

Figure: GAN-generated fake faces

https://research.nvidia.com/sites/default/files/pubs/2017-10_Progressive-Growing-of/karras2018iclr-paper.pdf

Need for AI explainability

Two goals occur to prevail across the users:

- need for model understanding
- need for regulatory compliance

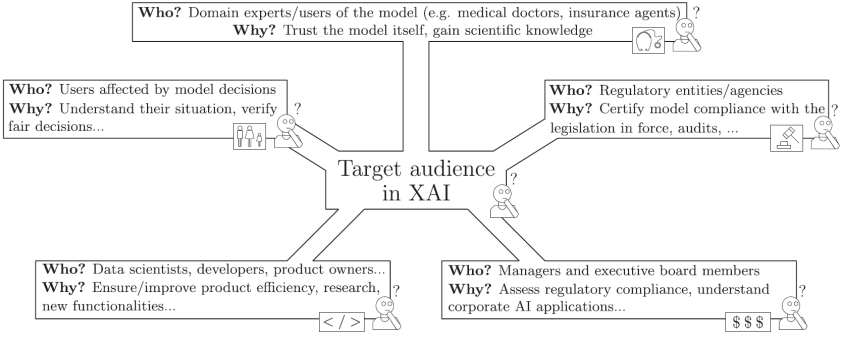


Figure: Explainability needs.

AI explainability

Make AI models explainable.

- Model transparency: Open the black box
- Post-hoc explainability

A. Barredo Arrieta *et al.* "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". Information Fusion, Volume 58, 2020, Pages 82-115. – This paper has cited 426 papers.

Model transparency: Overview

- Transparent models convey some degree of interpretability by themselves.
- Levels of transparency in machine learning models:
  - Algorithmic transparency
  - Decomposability
  - Simulatability
- Each of these classes contains its predecessors, e.g. a simulatable model is at the same time a model that is decomposable and algorithmically transparent.

Model transparency levels: Rough definition

- **Algorithmic transparency** deals with the ability of the user to understand the process followed by the model to produce any given output from its input data.
  - A linear model is deemed transparent because its error surface can be understood and reasoned about, allowing the user to understand how the model will act in every situation.
  - Contrarily, it is not possible to understand it in deep architectures as the loss landscape might be opaque, since it cannot be fully observed and the solution has to be approximated through heuristic optimization (e.g. stochastic gradient descent).
- **Decomposability** stands for the ability to explain each of the parts of a model (input, parameter and calculation).
  - The added constraint for an algorithmically transparent model to become decomposable is that every part of the model must be understandable by a human without the need for additional tools.
- **Simulatability** denotes the ability of a model of being simulated or thought about strictly by a human. Hence complexity takes a dominant place in this class.
  - Simple but extensive (i.e., with too large amount of rules) rule based systems fall out of this characteristic, whereas a single perceptron neural network falls within.
  - Sparse linear models are more interpretable than dense ones.

Model transparency levels: Example

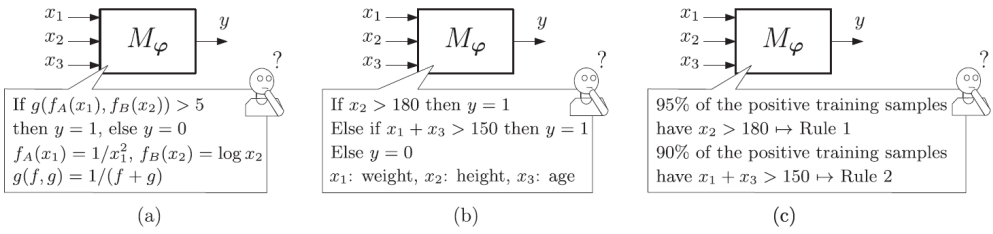


Figure: Conceptual diagram exemplifying the three levels of transparency characterizing a ML model M_ϕ, with ϕ denoting the model parameters: (a) simulatability; (b) decomposability; (c) algorithmic transparency.

Model transparency

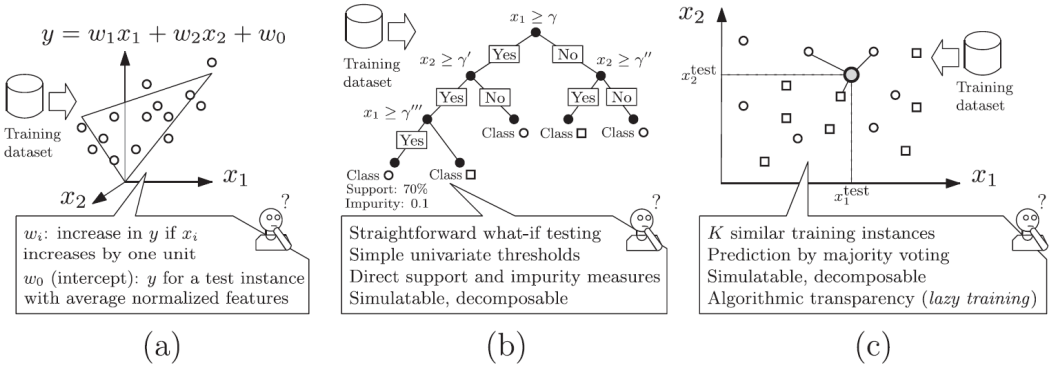


Figure: Transparent models. (a) Linear regression; (b) Decision trees; (c) K-Nearest Neighbors.

Model transparency

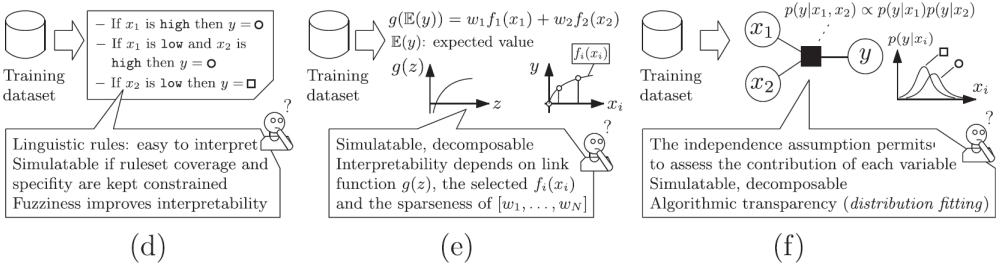


Figure: Transparent models. (d) Rule-based Learners; (e) Generalized Additive Models; (f) Bayesian Models.

Post-hoc explainability

- When AI models do not meet any of the criteria imposed to declare them transparent, a separate method must be devised and applied to the model to explain its decisions.
- The purpose of post-hoc explainability techniques (also referred to as post-modeling explainability) aims at communicating understandable information about how an already developed model produces its predictions for any given input.

Post-hoc explainability

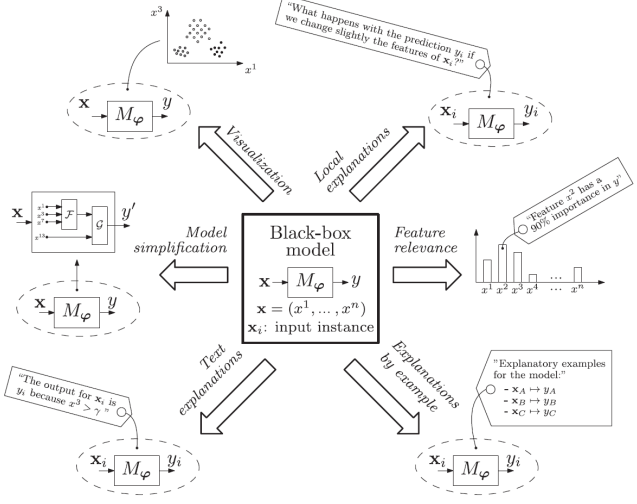


Figure: Post-hoc explainability

What is the course about?

- The course grips representative data analysis techniques for machine intelligence, touching upon both statistical and deep learning approaches, and *their application in embedded systems*.
- The course focuses on *applied time-series data analysis*, even though most models are general for both independent data and time-series data.
  - Intelligent embedded systems often deal with time-series data from various sensors such as temperature, EEG, ECG, lidar, radar etc., much more often than time-independent data.
  - Time-series analysis has its own set of theories and tools, from both statistical and machine learning domains.

Course structure

The course is designed to have 18 teaching & learning activities.

- 10 Lectures organized as 3 modules
- 3 Labs: 2 students per group
  - Lab 1. Time-series data visualization and feature extraction
  - Lab 2. ARIMA model and prediction
  - Lab 3. Clustering with K-means and SOM, and DTW
- 2 Seminars: 3 students per group
  - Seminar I. Time-series data mining and anomaly detection
  - Seminar II. Anomaly detection and AI challenges
- 1 Project: individual work
  - Project. Time-Series Prediction and Anomaly Detection

  A common application of time-series analysis for embedded system: anomaly detection

Modules

- Module I. Basic concepts, EDA (Exploratory Data Analysis) and feature extraction
- Module II. Statistical and Supervised learning: Modeling and Forecasting
- Module III. Unsupervised learning: Clustering algorithms

We deal with both time-series data and independent data, while weighing more on time-series analysis.

KLPs of Module I (Time-series analysis basics)

- Common graphs for EDA
  - Line (run sequence) plot
  - Historgram & Density plot
  - Heatmap
  - Box plot
  - Lag plot
  - ACF plot
  - PACF plot
- Typical features
  - Statistical features: Mean, Variance (Standard deviation) etc,
  - Temporal features: Auto-covariance, Auto-correlation etc.
  - Spectral features: Power spectrum, Spectrogram etc.

KLPs of Module I (Time-series analysis basics)

- Basic concepts
  - Randomness: White noise vs Random walk
    - How to check whether a series is random or not?
  - Stationarity
    - How to check whether a series is stationary or not?
  - Decomposition: trend, season, remainder
    - Which algorithms are used for decomposition?
- Hypothesis testing
  - Ljung-Box test for randomness
  - Augmented Dickey-Fuller (ADF) test for stationarity of *non-seasonal* series

Module II. Statistical and deep learning

- Statistical time-series modeling and forecasting
  - AR, MA, ARMA, ARIMA, Seasonal ARIMA models
  - Box-Jenkins methodology
- Deep Learning
  - Perceptron and MLP
  - RNN and LSTM

KLPs of ARMA Models

- Stationarity of an AR(p) process depends on its coefficients
- Invertibility of an MA(q) process depends on its coefficients
- Backshift notation $By_t = y_{t-1}$
- ARIMA model representation in backshift notation

KLPs of ARIMA and Seasonal ARIMA

ARIMA assumes that the series is stationary. If not, transformation is needed.

- Basic data transformations to stabilize trend and variance.
  - Differencing: d-th differencing, d-th order differencing
  - Square-root, log transforms
  - General Box-Cox transform
- Remember: A back-transformation is needed to recover to the original value scale. E.g., if the prediction is done for a differenced series, there is an integration step to obtain the predicted values in the original scale.
- Seasonality affects stationary, but its consideration is embedded into the model, Seasonal ARIMA (p, d, q) (P, D, Q).

KLPs of ARIMA(p, d, q)

Box-Jenkins methodology for ARIMA(p, d, q) modeling and prediction.

Starting point: The series is stationary but not random.

- Model identification: determine d, p, q
  - Check PACF for AR(p): cut off after lag p
  - Check ACF for MA(q): cut off after lag q
- Parameter estimation and model selection (e.g. AIC)
- Model validation: check the randomness of the remainder series
- Model prediction: In-sample and out-sample predictions.

Ending point: The remainder series is ideally random.

KLPs of deep learning

- Perceptron
  - The original perceptron used the step function as the activation function.
- Feed-forward structure: MLP
  - Why go deeper?
- Feed-back structure: RNN
  - Why feedback?
- LSTM
  - What problem does it intend to address?
  - How? 3 Gates: Forget gate, Input gate, output gate

Deep learning for time-series analysis

- Do not have the assumption on stationarity of the time series
- Organize the data set into input-output pairs for supervised learning
  - This applies to both non-sequence and sequence learning.

Module III Clustering algorithms (Unsupervised learning)

From supervised to unsupervised learning

- Statistical clustering
  - K-means clustering
  - Hierarchical clustering $\longrightarrow$ dendrogram
  - Dynamic time warping (DTW). Euclidean distance vs. DTW distance
- Neural network-based clustering
  - Self-Organizing Map (SOM)

KLPs of statistical clustering

- K-means clustering
  - Initialization
  - Iterative Expectation-Maximization
- Hierarchical clustering to obtain a dendrogram
- Various distance measures
- Dynamic time warping distance vs. Euclidean distance

KLPs of neural-network clustering

From statistical to self-organizing unsupervised learning

- SOM is a two-layer neural network conducting competitive unsupervised learning.
- SOM algorithm steps: best matching, cooperation, and adaptation
- SOM clustering vs. K-means clustering
  - For K-means algorithm, we need to specify ”K”. Any of the K clusters has mapped input samples.
  - Need to specify the output layer structure (hyper-parameters), but an output cell (neuron) can be empty (no samples mapped to this neuron).
  - Visualization: Map high-dimensional data set to a low-dimension (2D) space while capturing their neighborhood relation in the high-dimension space

About the project

- Individual completion
- Prepare the following
  1. Technical Report,
  2. Code deliverable: A zip file with all your code/result files and a readme file for executing your code, and
  3. Oral Presentation in pdf or ppt slides
- **Deadline for completion**: Submit the technical report, deliverable and presentation slides to the Canvas course room the day before the Final Workshop.

About the final workshop

- Time and place:
  Monday, May 27 14:00 to 18:00 in Room 203;
  Tuesday, May 28, 14:00 to 18:00 in Room 203
  (The exact finish time depends on number of students.)
- Registration: Required, in Canvas course room, Self registration
- Mandatory Participation: Physical presence.
- Individual presentation of the project.
- You will be asked to present part of your project work, results and answer questions. However, **You should prepare slides for all tasks.**