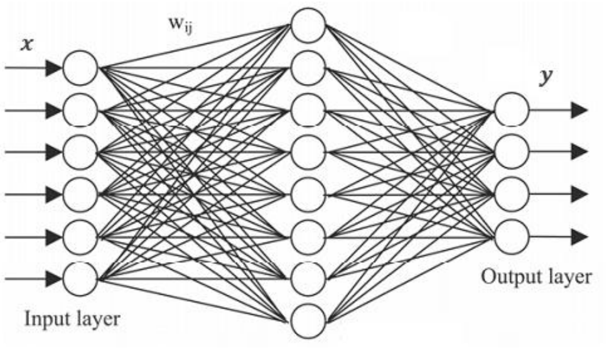
EPFL

Control and Operation of Tokamaks
Machine Learning for plasma control



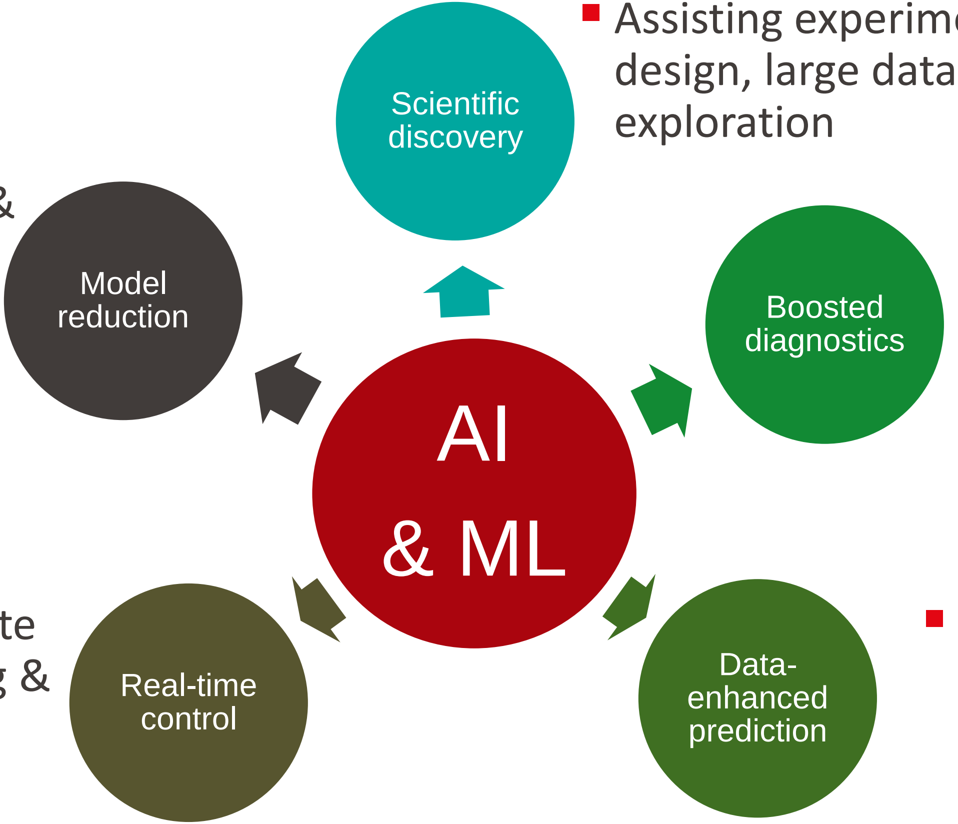
Alessandro Pau

12/02/2025, Lausanne

Swiss
Plasma
Center

Advancing fusion by leveraging AI and ML

- Assisting experimental design, large dataset exploration
- Reduced & Surrogate models
- Information extraction, data fusion

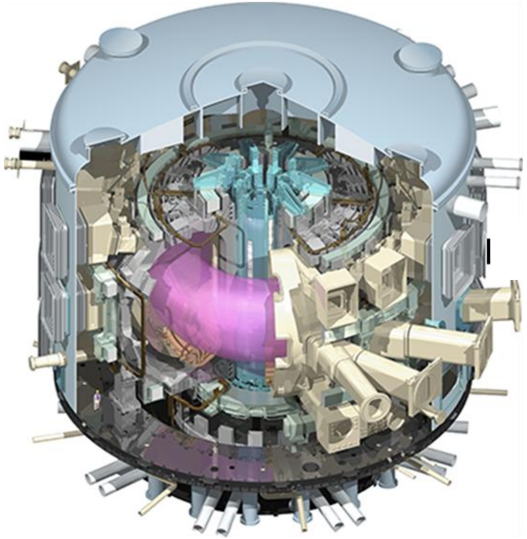


- Plasma state monitoring & control
- Disruptions, transient events

Swiss Plasma Center

Data in fusion: a challenge in itself

- **Massive amount of data** (Big data – 2PB/day at ITER, high bandwidth diagnostics) ✓



- **High-dimensional and heterogenous data** (many diagnostics measuring various plasma properties) ✓

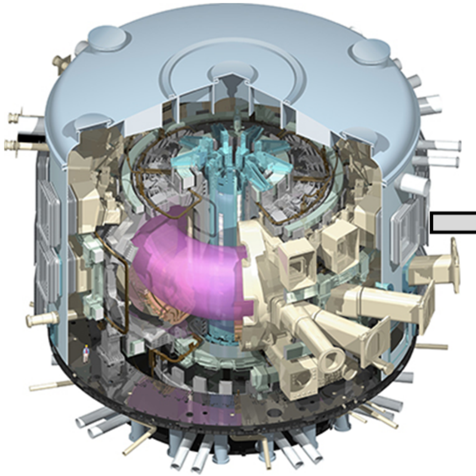
- **Clear formulation** of the problem, and **well-defined targets**? Not always easy to translate high-level fusion research objectives in a well-defined machine learning formulation…

- **Well-curated and annotated datasets:** do we have a well-defined vocabulary?

Swiss Plasma Center

Typical Machine Learning workflow

- Massive volume of data
- High-dimensional;



DATA PREPARATION

DATA PROCESSING

FEATURE EXTRACTION & ENGINEERING

HUMAN LABELING

REINFORCEMENT LEARNING

MACHINE LEARNING MODEL

UNSUPERVISED LEARNING

DATA/DIM-REDUCTION

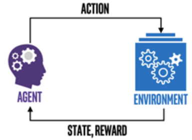
CLUSTERING

SUPERVISED LEARNING

CLASSIFICATION

REGRESSION

PROB. ESTIMATION



PLASMA CONTROL & SIMULATORS

DATA VISUALIZATION

PATTERN RECOGNITION

GENERATIVE MODELS

ANOMALY DETECTION

EVENT/OBJECT DETECTION

IMAGE PROCESSING

TIME SERIES FORECASTING

- Heterogenous
- multiple timescales

...Type of learning: Supervised Learning

Training data

$$\mathcal{D}: (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)$$

***Batch Learning*:**
- *training on a **dataset entirely available** to the learning algorithm, with model's parameters being updated after each iteration through the data.*
  - *Typically, **more computationally efficient**, but less flexible to adapt to new data distributions.*

***Active learning*:**
- *the learning algorithm is able to **interactively query** an information source to obtain the **desired outputs on new data** points (most informative data points to learn from)*
  - *often used when there is a limited amount of labelled data available: selecting which data points to learn from, the model can learn more effectively and efficiently.*

***Online Learning*:**
- *the algorithm receives **one example at a time**, with model's parameters being updated incrementally as new data comes in.*
  - *Useful in case of limitations on computing and storage*

EPFL

...Type of learning: Reinforcement Learning

Training data

$$(input, output, reward/penalty)$$

***Reinforcement learning*:**

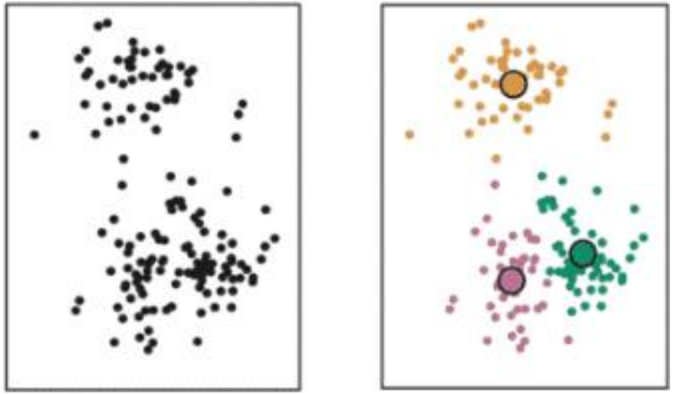
- *an "**agent**" learns to make decisions by continuously interacting with an <u>environment</u> and receiving **feedback** in the form of **rewards** or **penalties**.*
- *The goal of reinforcement learning is to learn a "**policy**", which is a **mapping from states to actions**, that maximizes the cumulative reward the agent receives over time.*

- ***Training data** consists of **sequences of states**, **actions**, and **rewards**.*
- *Learning by **trial-and-error**, where the agent takes actions, receives rewards, and updates its policy based on the observed rewards until convergence to an optimal solution*

…Type of learning: Unsupervised Learning

Training data

$$(x_1, \dots), (x_2, \dots), \dots, (x_N, \dots)$$



Unsupervised Learning:

- *useful to discover **patterns** or **structure** in the data, with **no labelled data**. The learning algorithm task is to identify structure in the data, such as **grouping** similar examples according to a well-defined **metric**.*
- *Some common unsupervised learning techniques:*
  - ***Clustering:** grouping of similar examples into clusters,*
  - ***dimensionality_reduction**: projection of the data into a lower-dimensional space while preserving as much of the structure of the data as possible*
  - ***anomaly detection**: identification of examples that are significantly different from the majority of the data (…novelty detection).*

ML foundations: fitting/training a model

Model fitting, or training:

(Training) data $\mathcal{D}: (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$

- *Learn the unknown target function describing the relation* $f(x, \boldsymbol{\theta}) \to y$
- *find the set of parameters* $\boldsymbol{\theta}$ *that best describe the mapping between the input and output variables in the data.*
- *Given the input data* $\mathcal{D}$*, solve an optimization problem in terms of minimization of an* ***objective*** *or* ***loss function***

Training examples

$\mathcal{D}: \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$

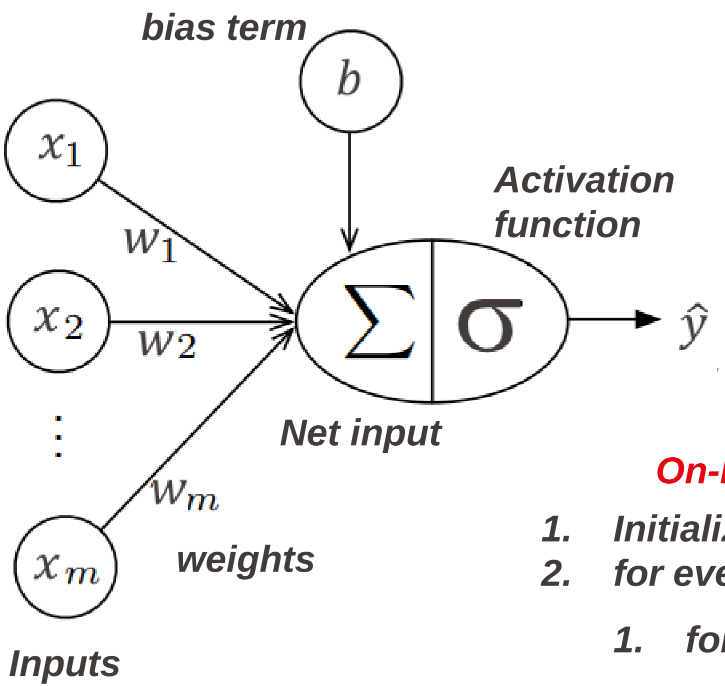
Loss function

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, \mathcal{L}(\mathrm{D}|\boldsymbol{\theta})$$

- *What we call* ***inference*** *depends on the context: quantify the uncertainty or confidence in the estimate* $\hat{\boldsymbol{\theta}}$*, or making prediction with a training model;*
  - *More in general: process of drawing conclusions about the underlying data-generating process*

EPFL

ML foundations: fitting/training a Perceptron



$$\hat{y} = \sigma\left(\left(\sum_{i=1}^{m} w_i \cdot x_i\right) + b\right) = \sigma(\boldsymbol{x}^T \boldsymbol{w} + b)$$

Given a training set:

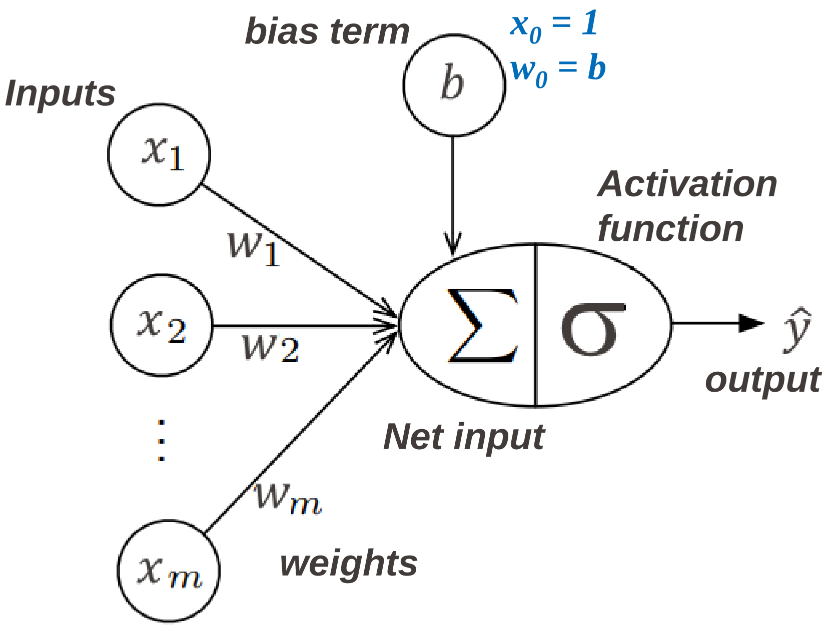
$$\mathcal{D}: \left(\boldsymbol{x}^{[1]}, y^{[1]}\right), \left(\boldsymbol{x}^{[2]}, y^{[2]}\right), \dots, \left(\boldsymbol{x}^{[N]}, y^{[N]}\right) \in \Re^m$$

On-line mode with Gradient Descent

1. ***Initialize $\boldsymbol{w}, \boldsymbol{b}$.*** $(with\ \ x^{[0]} = \boldsymbol{1}\ for\ \boldsymbol{b}\,)$
2. ***for every training epoch:***
    1. ***for every*** $\left(x^{[j]}, y^{[j]}\right)$ ***in*** $\mathcal{D}$: (or over mini-batches)
        1. $\hat{y}^{[j]} = \sigma\left(\boldsymbol{w}^T \boldsymbol{x}^{[j]} + b\right)$ ***compute prediction*** *(forward)*
        2. $err = \left(y^{[j]} - \hat{y}^{[j]}\right)$ ***compute error*** *(backward)*
        3. $\boldsymbol{w}, \boldsymbol{b} = \boldsymbol{w}, \boldsymbol{b} + err \cdot \boldsymbol{x}^{[j]}$ ***update parameters***

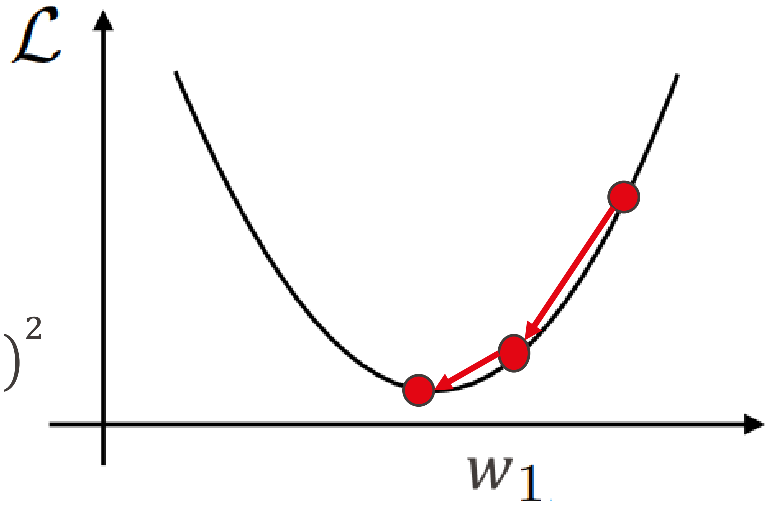
Swiss Plasma Center

ML foundations: gradient descent



$$\hat{y} = \sigma\left(\left(\sum_{i=1}^{m} w_i \cdot x_i\right) + b\right) = \sigma(\boldsymbol{x}^T \boldsymbol{w} + b)$$

$$\mathcal{D}: \left(\boldsymbol{x}^{[1]}, y^{[1]}\right), \left(\boldsymbol{x}^{[2]}, y^{[2]}\right), \dots, \left(\boldsymbol{x}^{[N]}, y^{[N]}\right) \in \Re^m$$



function

$$\mathcal{L}(\boldsymbol{w}, b) = \sum_j \left(\widehat{y^{[j]}} - y^{[j]}\right)^2$$

ML foundations: learning « modes »

On-line mode:

- *Learning faster but noisier (shuffling each epoch) – update after each $\left(x^{[j]}, y^{[j]}\right)$*

Batch mode:

- *Slower but less sensitive to noise*
- *update after the entire data "batch"*

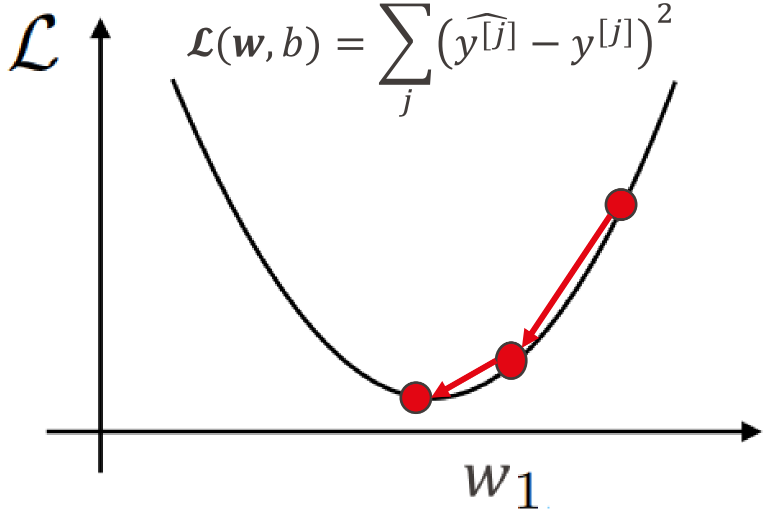
$$\hat{y} = \sigma\left(\left(\sum_{i=1}^{m} w_i \cdot x_i\right) + b\right) = \sigma(\boldsymbol{x}^T\boldsymbol{w} + b)$$

$$\mathcal{D}: \left(\boldsymbol{x}^{[1]}, y^{[1]}\right), \left(\boldsymbol{x}^{[2]}, y^{[2]}\right), \dots, \left(\boldsymbol{x}^{[N]}, y^{[N]}\right) \in \Re^m$$

Mini-batch mode (*typically used in DL*)

- *In between the previous two: with respect to batch settings, the update is done for each* ***"mini-batches".***
- *Advantage:* ***vectorization*** *(GPUs)*
- *Less noisy than online-mode & learning faster than batch*

$$\mathcal{L}(\boldsymbol{w}, b) = \sum_j \left(\widehat{y^{[j]}} - y^{[j]}\right)^2$$

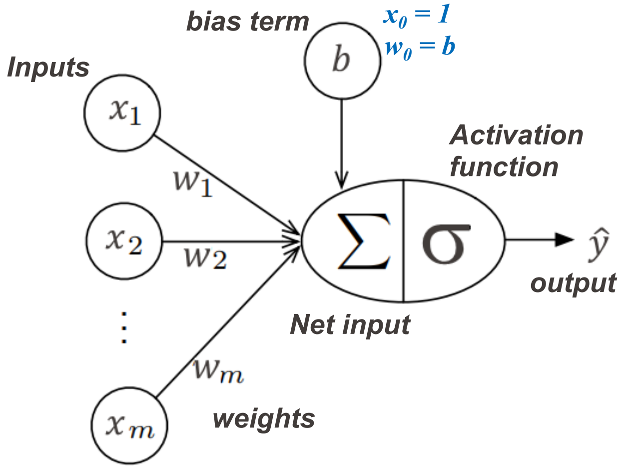


Other training paradigm:

- ***Stochastic Gradient Descend*** *(SGD)*
- ***Batch Normalization*** *(BN)*

ML foundations: Linear Regression (Least-squares)

Optimization problems with Least-Squares *normal equation:* $w = (X^T X)^{-1} X^T y$



$$\hat{y} = \sigma(\boldsymbol{x}^T \boldsymbol{w} + b); \quad \sigma = I;$$

matrix form

$$\hat{y} = \boldsymbol{X}\boldsymbol{w} \qquad \mathcal{L}(\boldsymbol{w}) = \frac{1}{2m}\sum_j \left(\hat{y}^{[j]} - y^{[j]}\right)^2$$

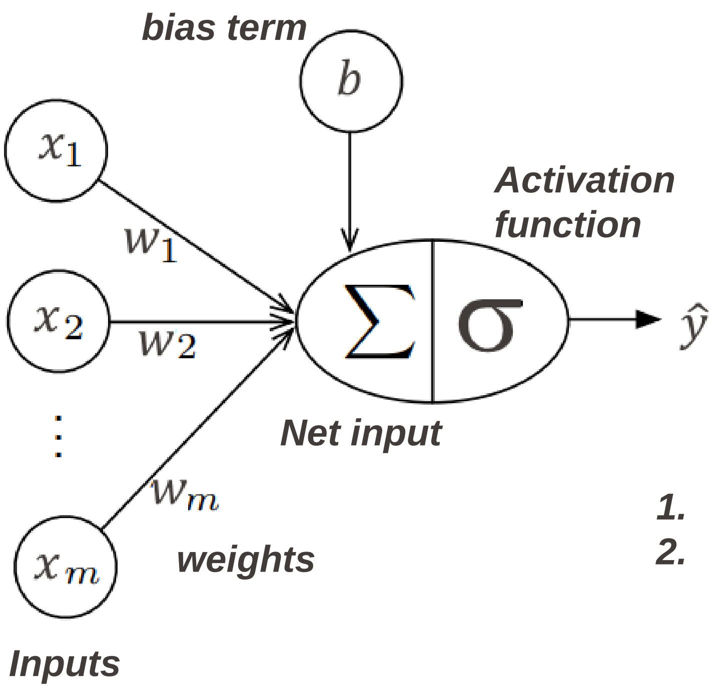
$$\nabla\mathcal{L}(\boldsymbol{w}) = \frac{1}{2m}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$= \frac{1}{2m} 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$ *(using chain rules)*

$$\nabla\mathcal{L}(\boldsymbol{w}) = 0 \;\rightarrow\; \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0 \;\rightarrow\; \boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T y$$

- *We have to fit basically a **linear regression model***
- *Reasons: Sometimes closed-form solution (matrix inversion) computationally expensive (large $\mathcal{D}$)*
- *We can learn this parameters **iteratively**, fitting (deep) **neural networks** and (**non-)convex loss functions***

ML foundations: Linear Regression (Least-squares)



$$\hat{y} = \sigma\left(\left(\sum_{i=1}^{m} w_i \cdot x_i\right) + b\right) = \sigma(\boldsymbol{x}^T \boldsymbol{w} + b)$$

Given a training set:

$$\mathcal{D}: \left(\boldsymbol{x}^{[1]}, y^{[1]}\right), \left(\boldsymbol{x}^{[2]}, y^{[2]}\right), \dots, \left(\boldsymbol{x}^{[N]}, y^{[N]}\right) \in \Re^m$$

Convex loss function

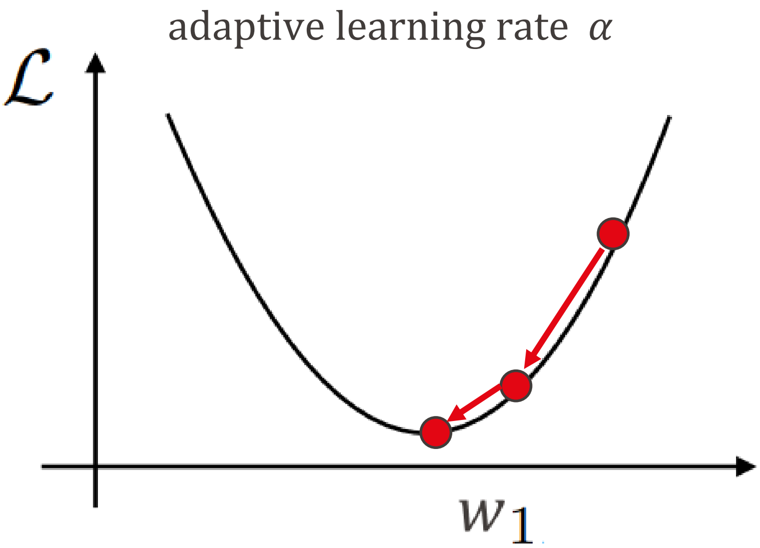
$$\mathcal{L}(\boldsymbol{w}, b) = \sum_j \left(\hat{y}^{[j]} - y^{[j]}\right)^2$$

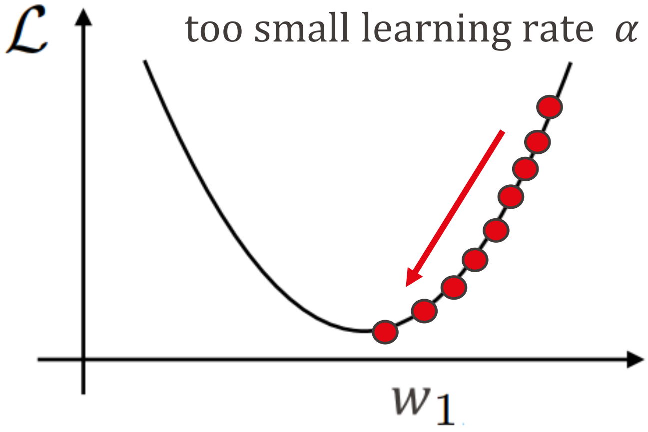
On-line mode with (Stochastic) Gradient Descent)

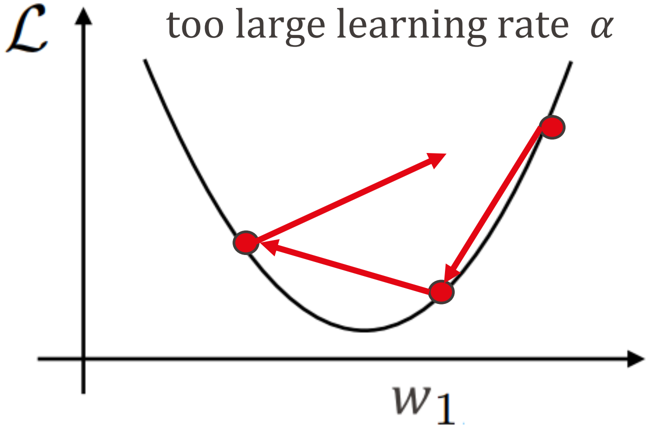
1. *Initialize* $\boldsymbol{w}, b$
2. *for every training epoch:*
   1. *for every* $\left(x^{[j]}, y^{[j]}\right)$ *in* $\mathcal{D}$: (or over mini-batches)
      1. $\hat{y}^{[j]} = \sigma\left(\boldsymbol{w}^T \boldsymbol{x}^{[j]} + b\right)$ *compute prediction*
      2. $\nabla_{\boldsymbol{w},\boldsymbol{b}}\mathcal{L} = \left(y^{[j]} - \hat{y}^{[j]}\right) \cdot \boldsymbol{x}^{[j]}$ *calculate error*
      3. $\boldsymbol{w}, \boldsymbol{b} = \boldsymbol{w}, \boldsymbol{b} + \alpha \cdot \left(-\nabla_{\boldsymbol{w},\boldsymbol{b}}\mathcal{L}\right)$. *update parameters*

learning rate

ML foundations: fitting/ Gradient Descent (GD)







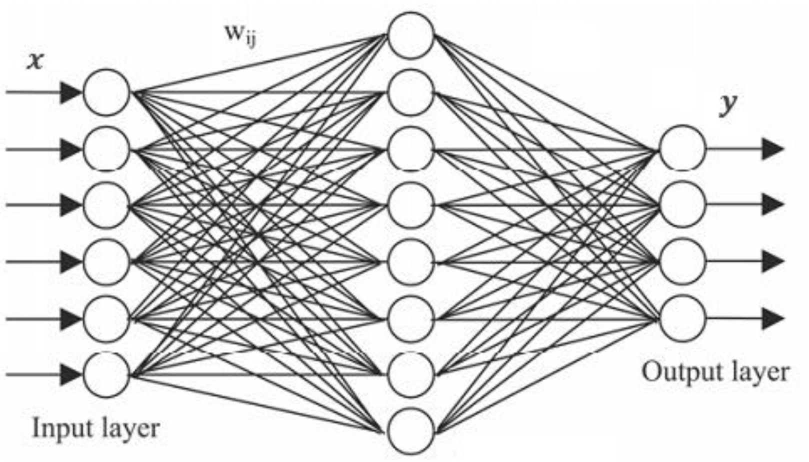
Convex Loss function (with a global minimum)

$$\mathcal{L}(\boldsymbol{w}, b) = \sum_j \left(\hat{y}^{[j]} - y^{[j]}\right)^2$$

Swiss Plasma Center

ML foundations: Linear Regression (Least-squares)

Back-propagation (Jacobians)



Convex Loss function

$$\mathcal{L}(\boldsymbol{w}, b) = \frac{1}{2N}\sum_j \left(\widehat{y^{[j]}} - y^{[j]}\right)^2$$ ***(MSE)***

$$\hat{y} = \sigma(\boldsymbol{w}^T \boldsymbol{x} + b)$$ ***(prediction)***

On-line mode with Stochastic Gradient Descent

1. $$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial\left(\frac{1}{2N}\sum_j\left(\widehat{y^{[j]}} - y^{[j]}\right)^2\right)}{\partial w_i}$$

2. $$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial\left(\frac{1}{n}\sum_j \frac{1}{2}\cdot\left(\sigma\left(\boldsymbol{w}^T x^{[j]}\right) - y^{[j]}\right)^2\right)}{\partial w_i}$$

(chain rule for $f\left(\sigma(h(\boldsymbol{w}))\right)$)

$$\frac{\partial f\left(\sigma(\boldsymbol{h}(\boldsymbol{w}))\right)}{\partial w_i} = \frac{\partial f}{\partial \sigma}\cdot\frac{\partial \sigma}{\partial h}\cdot\frac{\partial h}{\partial w_i}$$

where $$\begin{cases} f = (\sigma - y_i)^2 \\ \sigma = I(h) \\ h = \boldsymbol{w}^T x^{[j]} \end{cases}$$

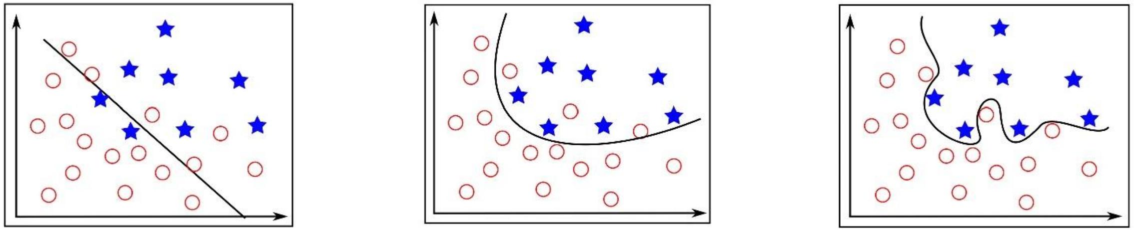
Outer → Inner

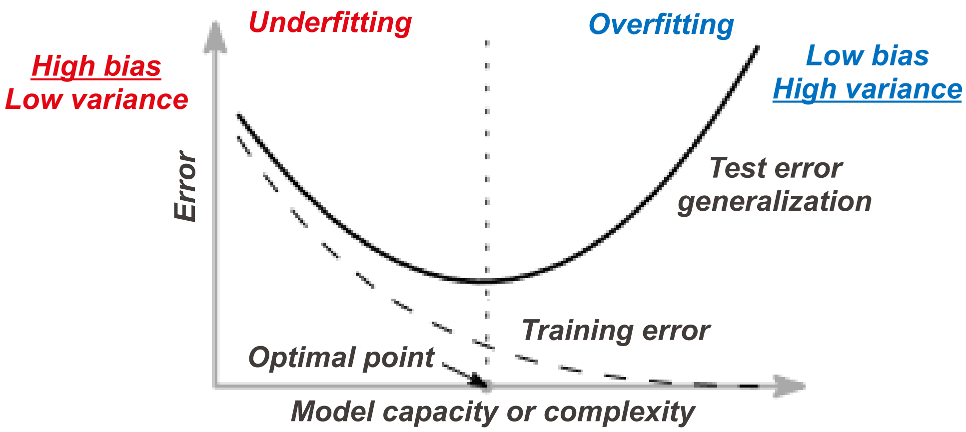
3. $$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{1}{n}\sum_j\left(\sigma(h) - y^{[j]}\right)\cdot\frac{d\sigma}{dh}\cdot\frac{\partial\left(\boldsymbol{w}^T x^{[j]}\right)}{\partial w_i}$$

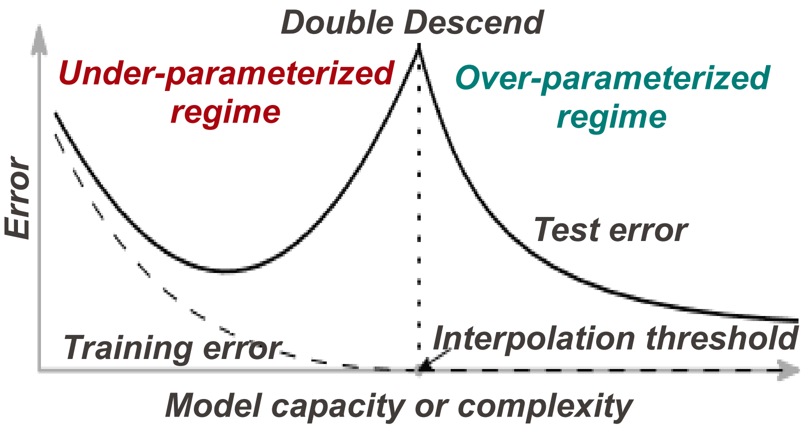
4. $$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{1}{n}\sum_j\left(\sigma\left(\boldsymbol{w}^T x^{[j]}\right) - y^{[j]}\right)\cdot x_i^{[j]}$$

Swiss Plasma Center

Underfitting and Overfitting: bias/variance trade-off







Underfitting and Overfitting: bias/variance intuition

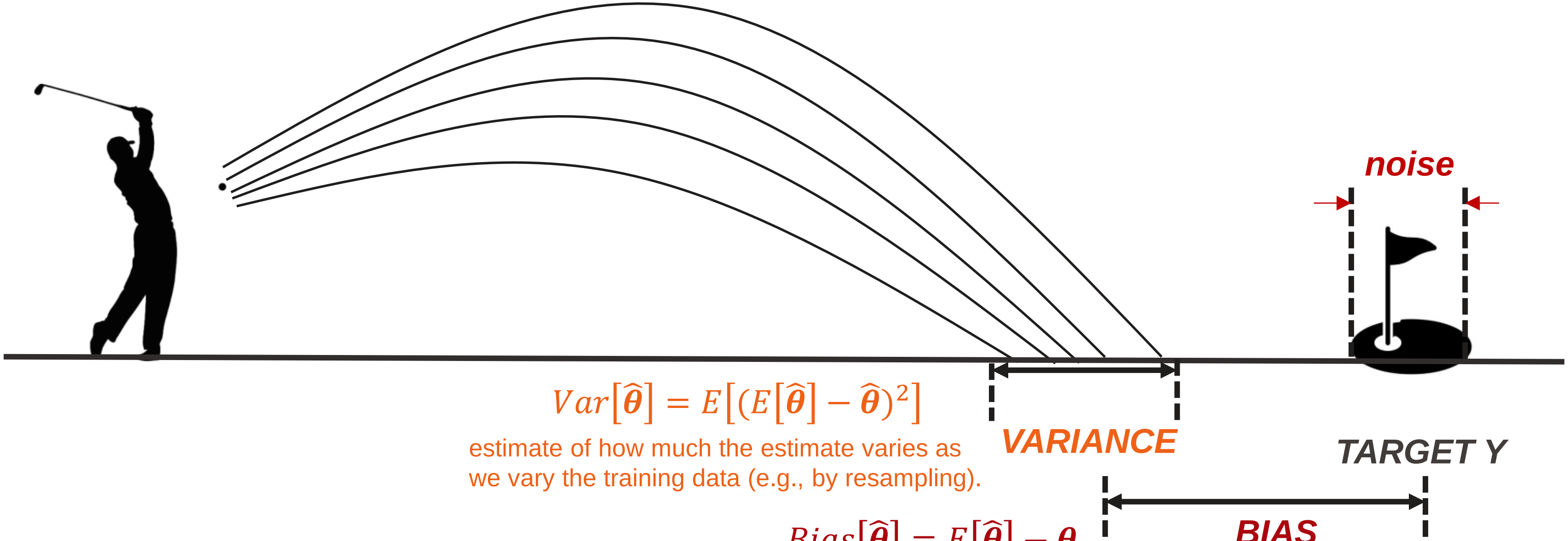
Bias/variance decomposition of the squared error [derivation]

$$Err(x) = (Bias[\hat{f}(x)])^2 + Var[\hat{f}(x)] + \sigma^2$$

How far the learned model is from the true function

Changes when the model is trained on different data samples

Irreducible error



$$Var[\hat{\boldsymbol{\theta}}] = E[(E[\hat{\boldsymbol{\theta}}] - \hat{\boldsymbol{\theta}})^2]$$

estimate of how much the estimate varies as we vary the training data (e.g., by resampling).

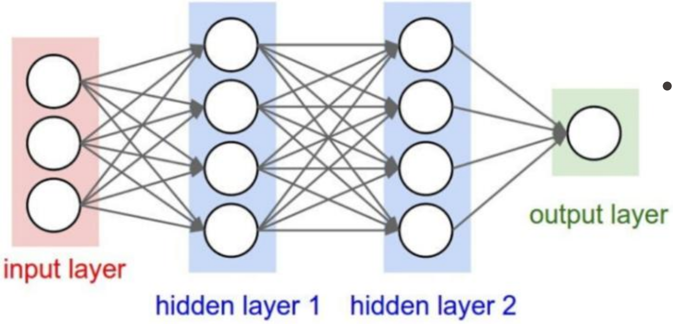
$$Bias[\hat{\boldsymbol{\theta}}] = E[\hat{\boldsymbol{\theta}}] - \boldsymbol{\theta}$$

difference between the average estimator from different training samples and the true value.

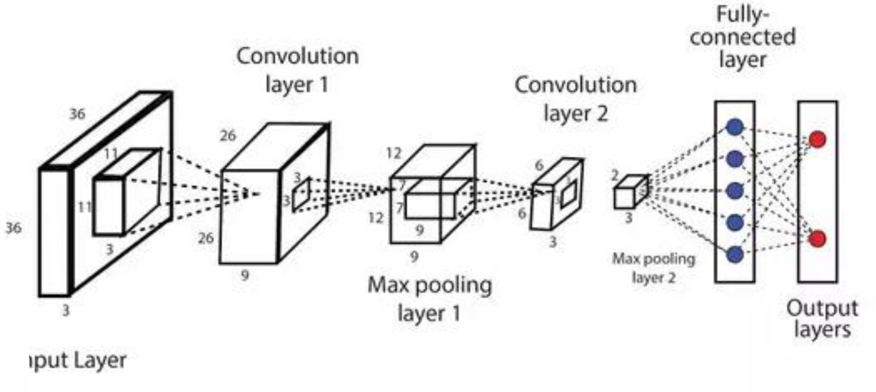
Swiss Plasma Center

Relational Inductive Biases

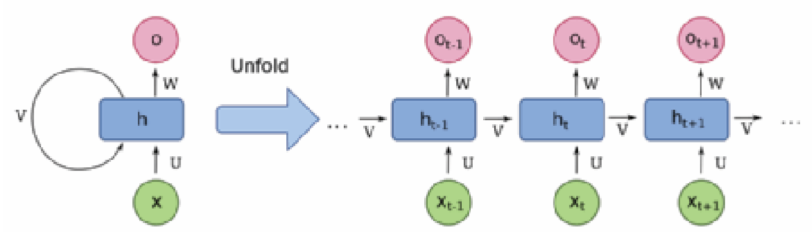
Inductive Bias: set of assumptions a learning algorithm uses to generalize from the training data to unseen examples.



- ***Multilayer NN:*** *feedforward (**shuffling** & **independence**)*

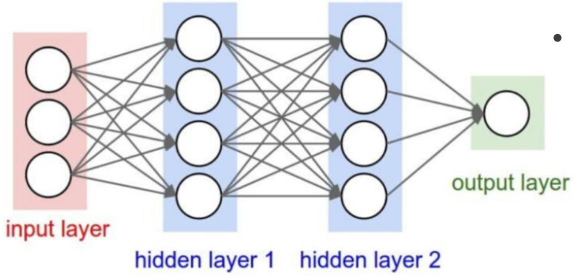


- ***cNN:*** *convolution filters (spatial/time **locality** & **equivariance**)*



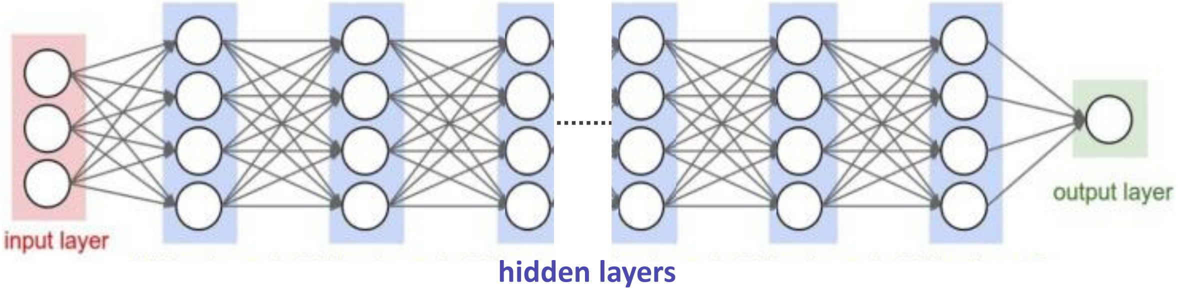
- ***RNN:*** *recurrent relation at each time step to process a sequence (**sequentiality**)*
- ***Back propagation** through time*

Relational Inductive Biases



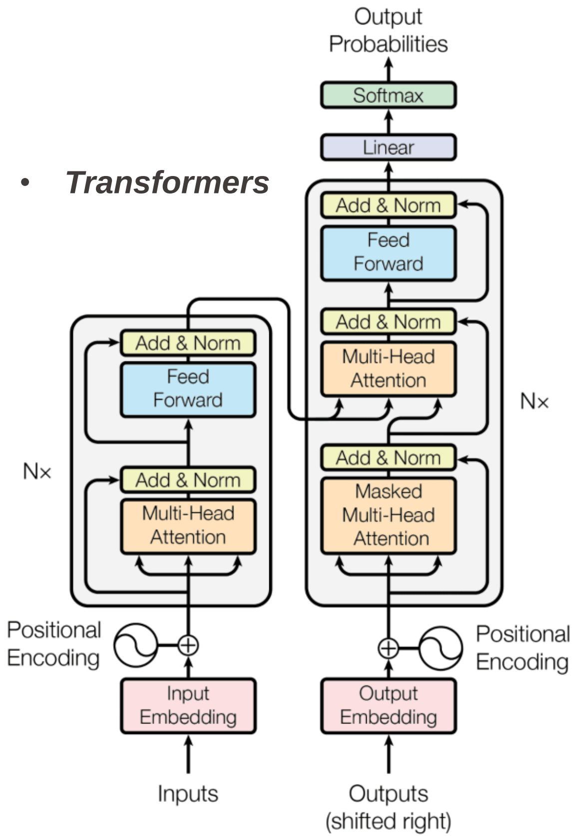
- ***Multilayer NN:*** *feedforward (**shuffling** & **independence**)*

- ***Deep Learning***



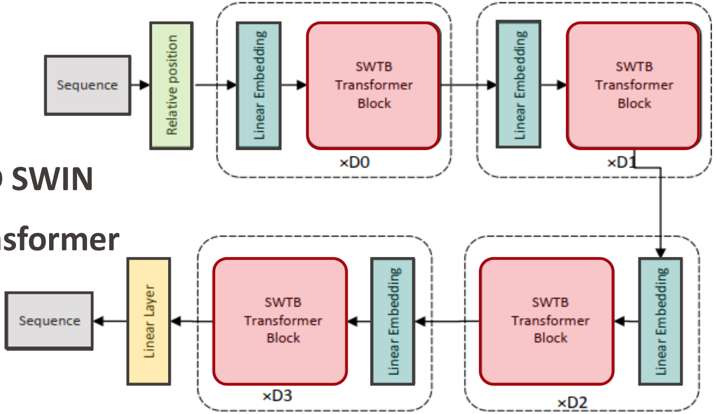
- Neural Networks with many layers (**deep architectures**)
- learn **representations** of data through a process of model abstraction, automatically discovering the representations needed for detection or classification
- it replaces **feature learning** or feature engineering
- Originally hard to train (but now we have GPUs) & **less interpretable**

- ***Transformers***

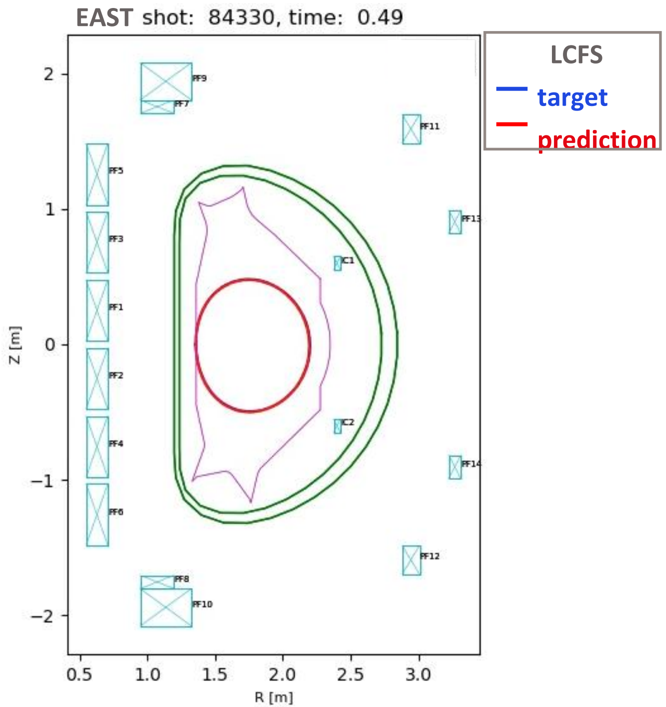


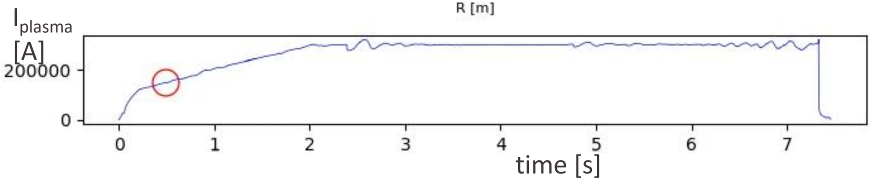
ML foundations: Linear Regression (Least-squares)

- **magnetic equilibrium** reconstruction:
  - complex time-varying, non-linear, **multi-scale**…
  - Modeling **sequences** with large variations in the time scales… "**attention** is all you need"!
  - …**one-step ahead** prediction of the magnetic field evolution in time (**L**ast **C**losed **F**lux **S**urface)



1D SWIN Transformer

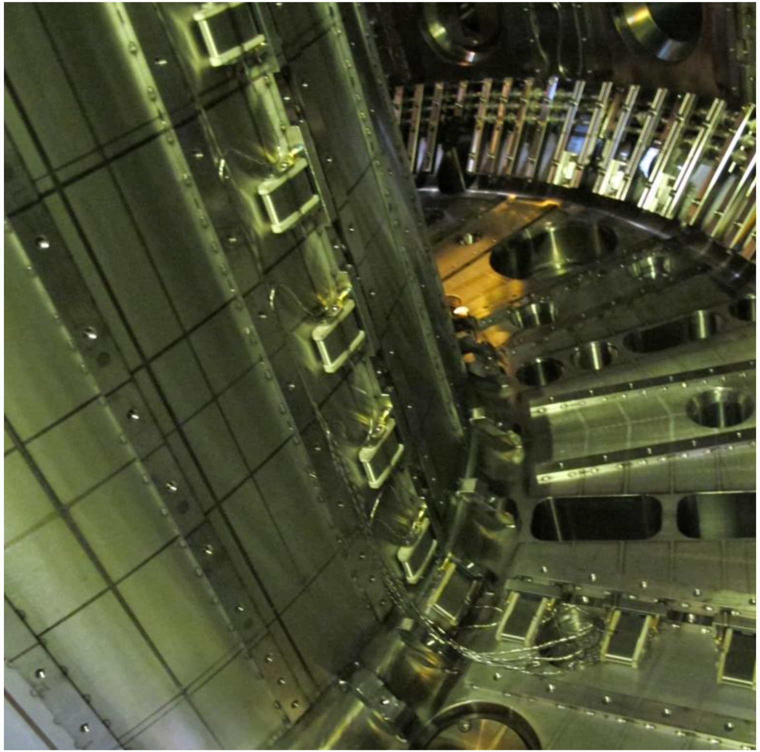




REF:[C. Wan, A. Pau, O Sauter et al 2022 (in review)]

SVD, PCA and MHD modes detection

Extracting Physics from Sensor Data

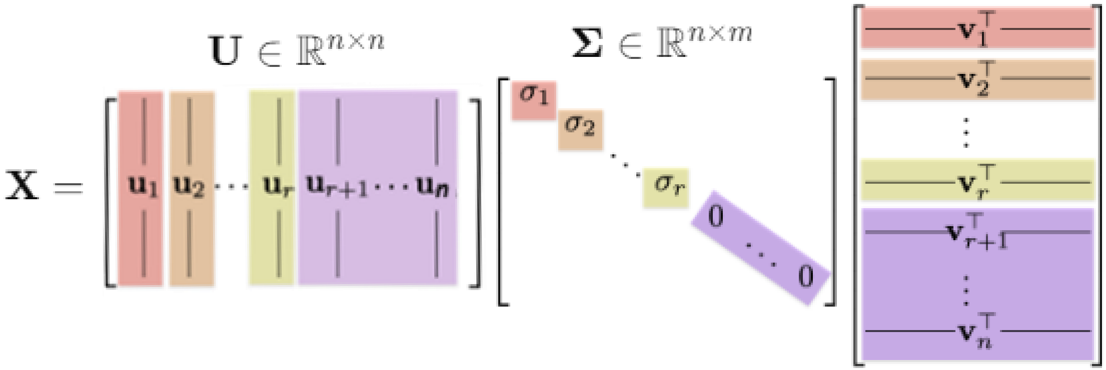


- Data fusion techniques enhance insights from multiple sensors.
- Machine learning aids in identifying significant patterns, extracting temporal and spatial correlations.
- Interpret dominant patterns to extract physics knowledge
- Real-time analysis improves control strategies.

Singular Value Decomposition (SVD)

- $\mathbf{X} \in \mathbb{R}^{n \times m}$:
- **Rows (n):** each row represents a measurement at a specific time;
- **Columns (m):** each column corresponds to one sensor placed in a spatial array (e.g., magnetic probes)

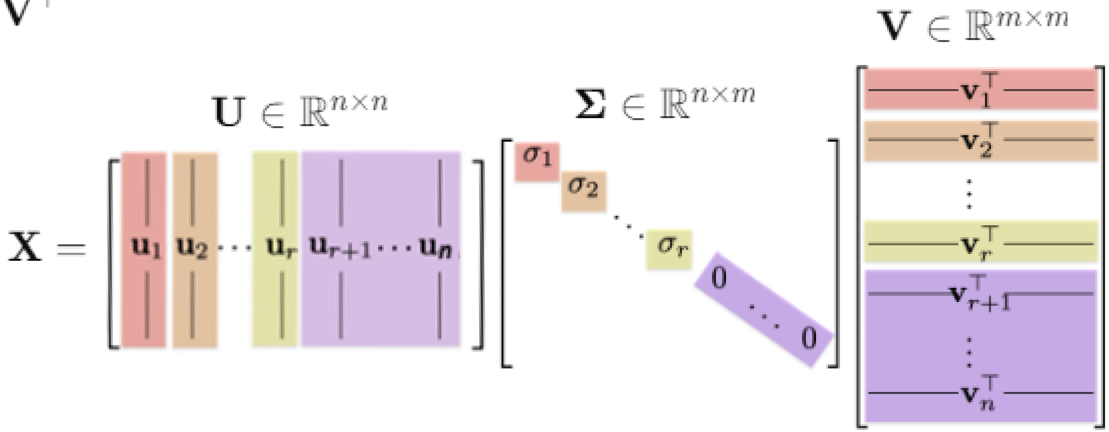
$$\mathbf{X} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^{\top}$$



EPFL

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top$$

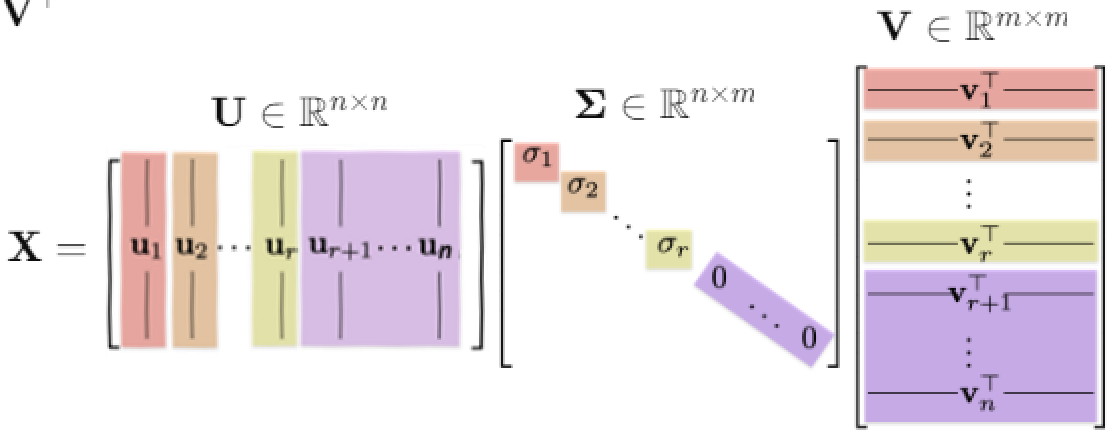


- $\mathbf{U} \in \mathbb{R}^{n\times n}$: Temporal modes (left singular vectors).
- $\mathbf{\Sigma} \in \mathbb{R}^{n\times m}$: Diagonal matrix with nonnegative singular values σ_i in descending order.
- $\mathbf{V} \in \mathbb{R}^{m\times m}$: Spatial modes (right singular vectors).

Swiss Plasma Center

Singular Value Decomposition (SVD)

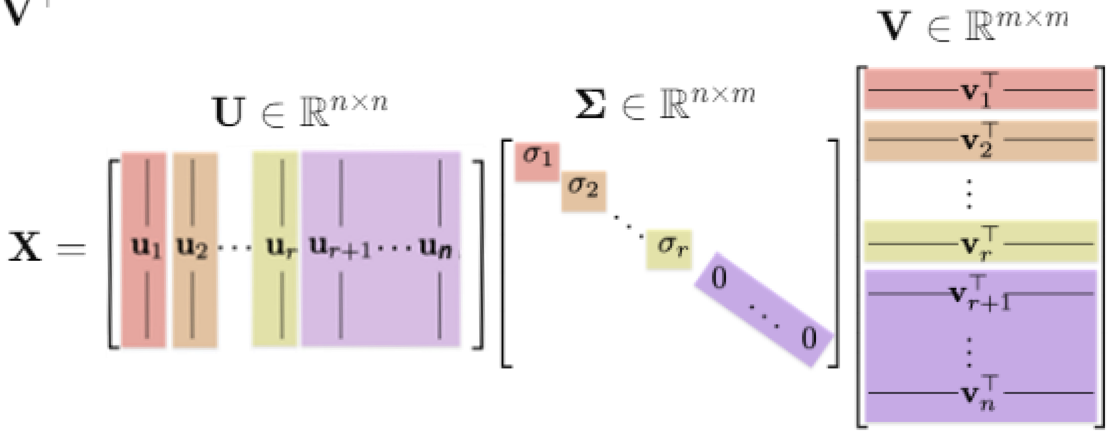
$$\mathbf{X} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^\top$$



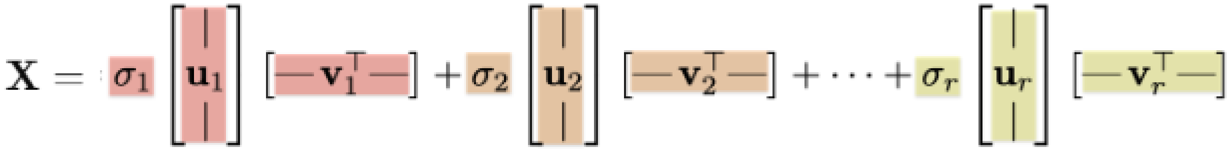
- **Temporal modes**: left singular vectors (**U**) capturing the temporal evolution of the sensor signals
- **Singular values**: non-negative values (**Σ**) arranged in a descending order, corresponding to the energy or importance of each mode
- **Spatial modes**: rigth singular vectors (**V**) revealing patterns and correlations across the sensors (e.g., coherent magnetic fluctuations).

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top$$

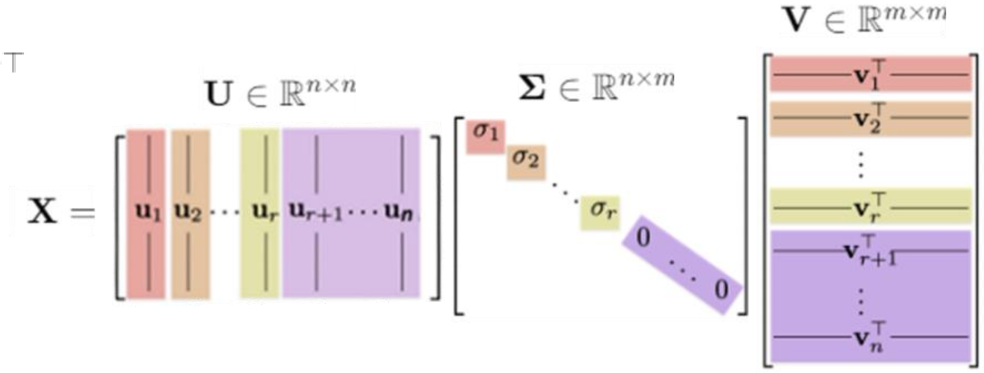


- **Singular values energy ranking**: dominant spatio-temporal modes together with their relative importance allowing for low-rank approximations (useful for noise reduction and dimensionality reduction).



Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^\top$$



MATLAB

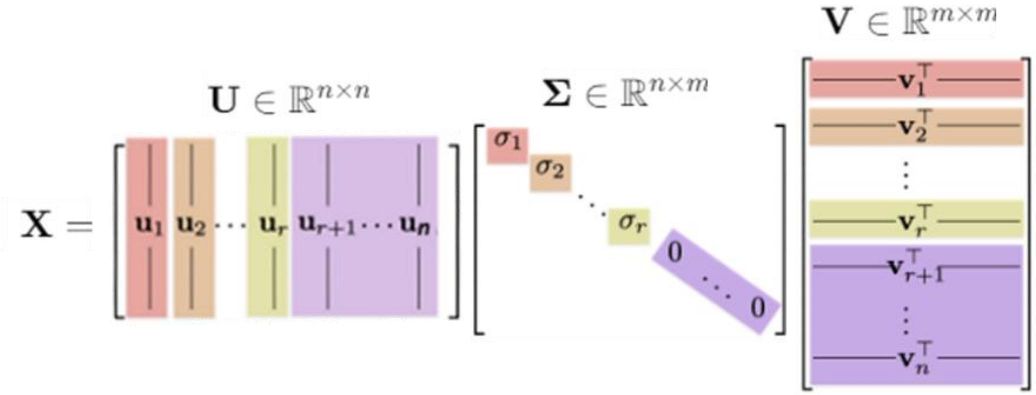
```
>> X = randn(100, 7); % Create a random data matrix
>> [U,S,V] = svd(X);  % full SVD
>> [U,S,V] = svd(X, 'econ');
```

python

```
>>> import numpy as np
>>> X = np.random.rand(100, 7)  # create random data matrix
>>> U, S, V = np.linalg.svd(X,full_matrices=True) # full SVD
>>> Uhat, Shat, Vhat = np.linalg.svd(X, full_matrices=False) # economy SVD
```

Singular Value Decomposition – correlation matrix

$$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top$$

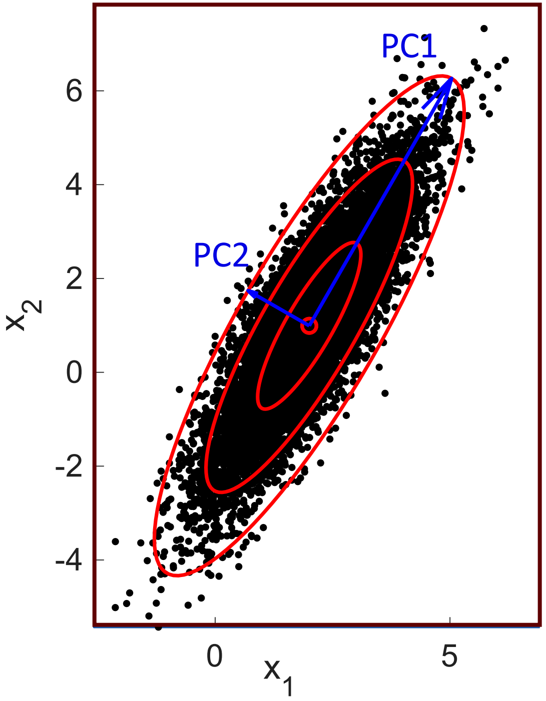


- ❑ Compute sensor correlation matrix: $\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{m\times m}$
- ❑ Substitute the **SVD** of $\mathbf{X}$:

$$\mathbf{X}^\top\mathbf{X} = \left(\mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top\right)^\top\left(\mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top\right)$$
$$= \mathbf{V}\,\mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top = \mathbf{V}\,\mathbf{\Sigma}^2\,\mathbf{V}^\top$$

- ❑ $\mathbf{X}^\top\mathbf{X} = \mathbf{V}\,\mathbf{\Sigma}^2\,\mathbf{V}^\top$ eigenvalue decomposition of the correlation matrix
- ❑ Each non-singular singular value is the positive square root of an eigenvalue of the correlation matrix $\sigma_i = \sqrt{\lambda_i}$ (i.e., $\lambda_i = \sigma_i^2$).

Link with Principal Component Analysis (PCA)



- Replacing the matrix $\mathbf{X}$ with them **mean subtracted matrix** (row-wise subtraction) $\mathbf{X} - \bar{\mathbf{X}} \rightarrow \mathbf{X}$

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij} \qquad \bar{\mathbf{X}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \bar{\mathbf{x}}$$

- From the covariance matrix $\mathbf{X}^\top \mathbf{X}$ we get directly the **principal "directions"** by performing an eigen-decomposition of the matrix itself:

$$\mathbf{X}^\top \mathbf{X} = \mathbf{V}\,\mathbf{\Sigma}^2\,\mathbf{V}^\top$$

- The **eigenvectors** (columns of $\mathbf{V}$) indicate the **directions** of maximum variance, and the **eigenvalues** represent the **variance** explained by each **principal component**.
- The **principal component scores** are the projections of the data onto the principal directions.

Scores: $\mathbf{X}\mathbf{V} = \mathbf{U}\Sigma$

(**principal components** in the observation space)

Least square and regression with SVD

- We want to find the slope 'a' that best fits **y = ax**
- "Best fit" here means minimizing the sum of squared errors → minimize $\|\mathbf{y} - \mathbf{x}a\|^2$
- Taking the derivative and setting it to zero gives us the normal equations: **x'xa = x'y**

$$[x] = U\Sigma V^T$$

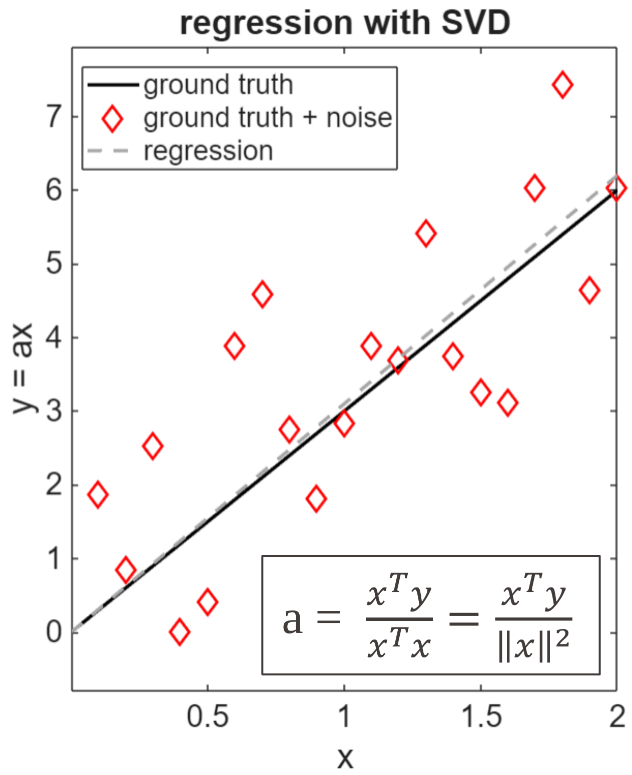
$$[y] = [x]a = U\Sigma V^T a$$

$$a = V\Sigma^{-1}U^T y$$

$$\Sigma = \|x\|$$ Length of x

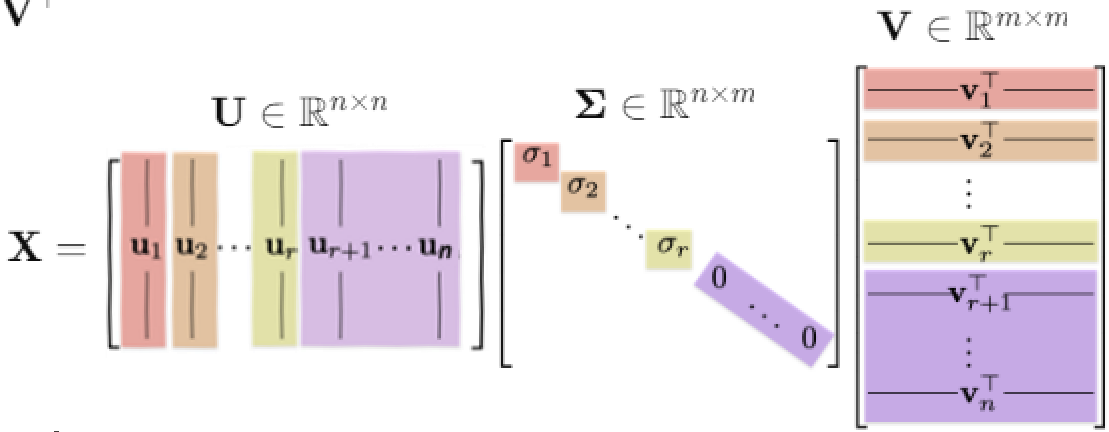
$$V = 1$$ Unit vector

$$U = x/\|x\|$$ Unit vector in the direction x



MHD perturbations in fusion plasmas TCV

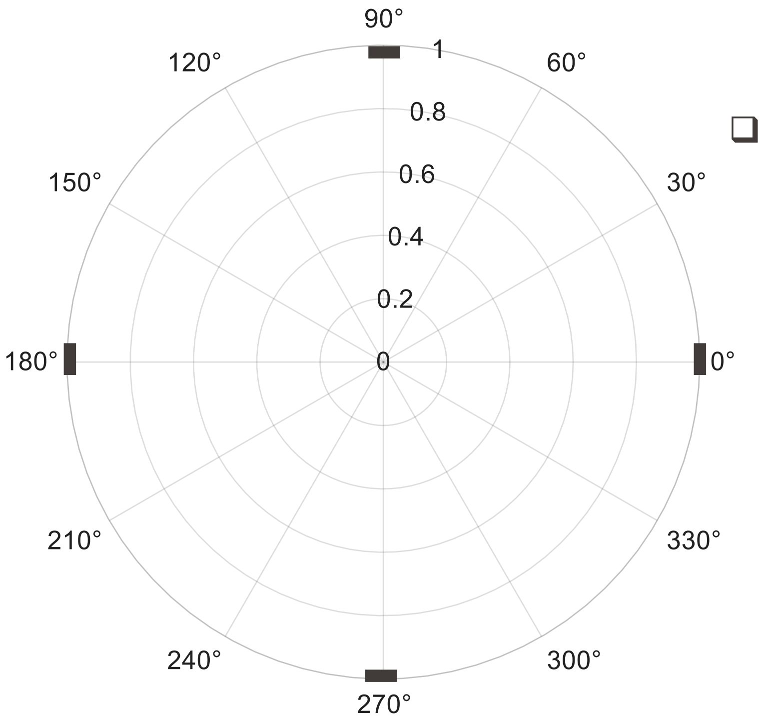
$$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{\top}$$



Singular values:

- ❑ allows decoupling **Spatial** and **Temporal patterns**:
- ❑ **Physical Interpretation**
  - ❑ **singular values** → energy and coherency of the MHD perturbation
  - ❑ **dominant spatial mode(s)** → dominant patterns across sensors, indicating a large-scale magnetic perturbation;
  - ❑ **dominant temporal mode(s)** → time evolution of the perturbation (e.g., oscillations, rotations).

MHD modes with SVD analysis



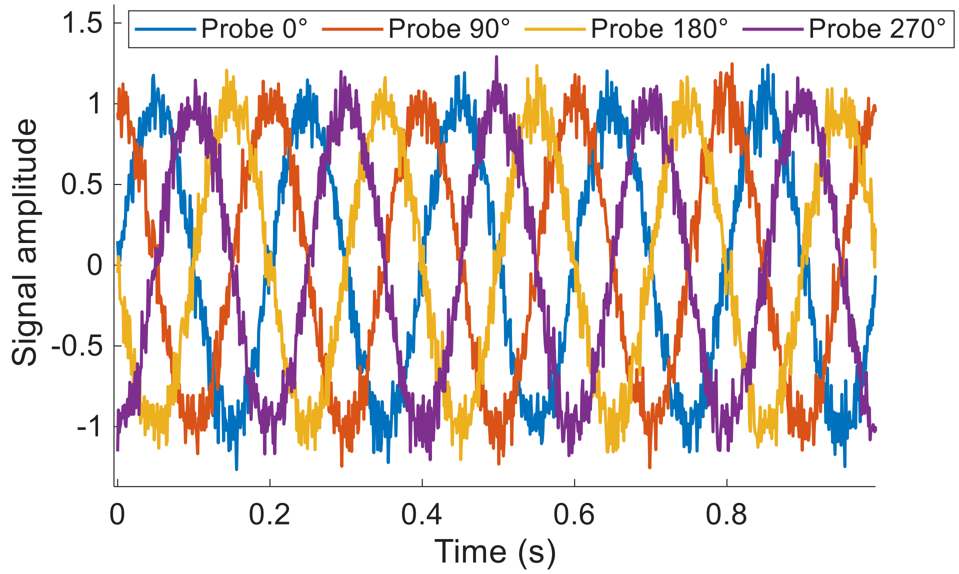
Phase shift (coil position, mode periodicity)

Coils phase offset

Amplitude

Rotation frequency

- **Coil measurement**: $y_k(t) = A_k \cos(\theta_k - 2\pi \, v \, t + \Theta)$

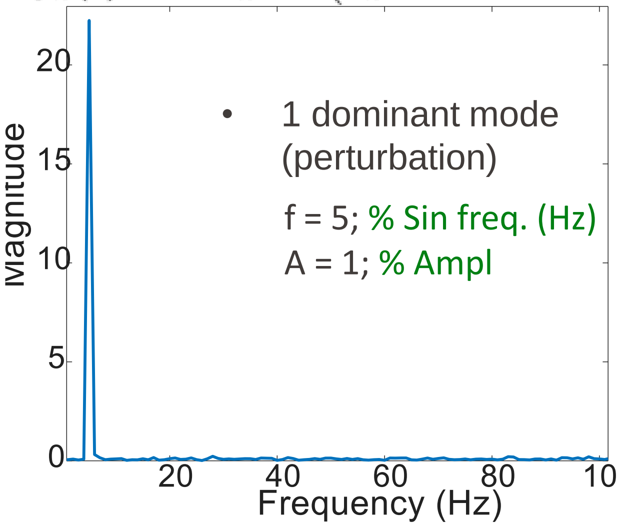


- **Matrix measurements**: $(Y)_{i,j} = y(x_j, t_i)$
  - rows = time indices
  - columns = sensor positions

EPFL

MHD modes with SVD analysis

$$y_k(t) = A_k \cos(\theta_k - 2\pi v t + \Theta)$$



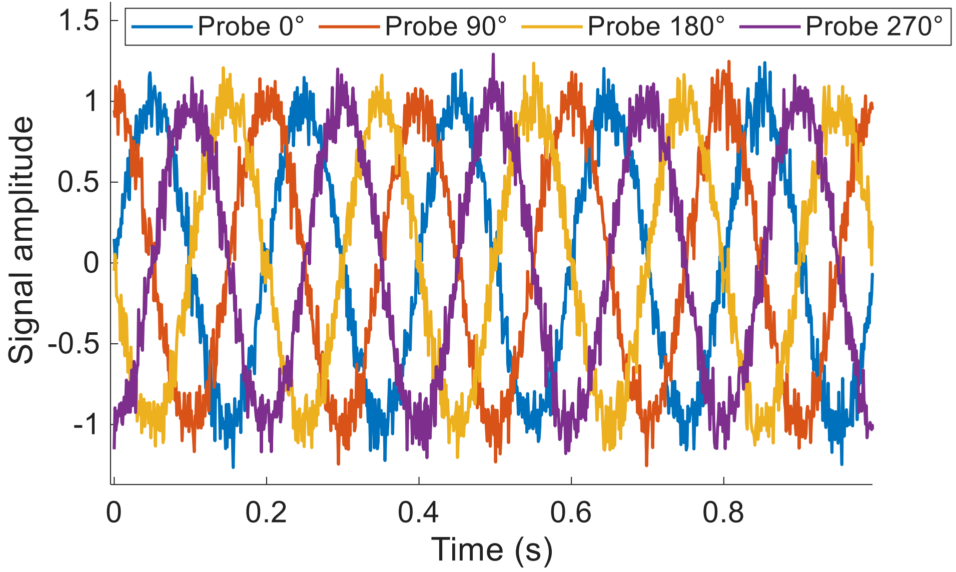
❑ **Coil measurement**: $y_k(t) = A_k \cos(\theta_k - 2\pi v t + \Theta)$

Amplitude (A_k)

Phase shift (coil position, mode periodicity) (θ_k)

Rotation frequency ($2\pi v t$)

Coils phase offset (Θ)



❑ **Perform SVD** : $Y = U\,S\,V^\top$

• Q: How should the singular values like?

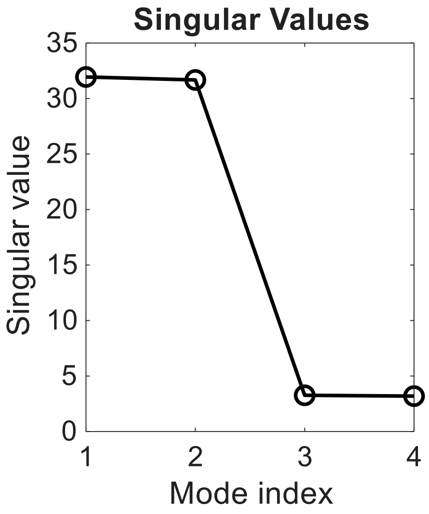
❑ **Matrix measurements**: $(Y)_{i,j} = y(x_j, t_i)$

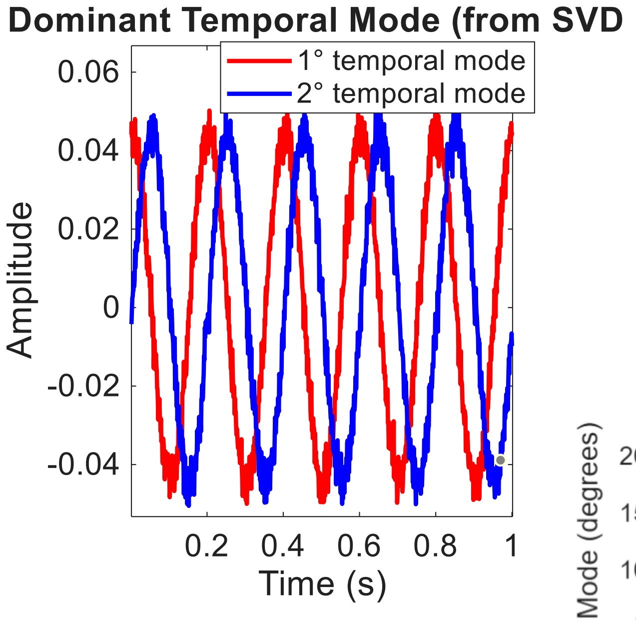
EPFL

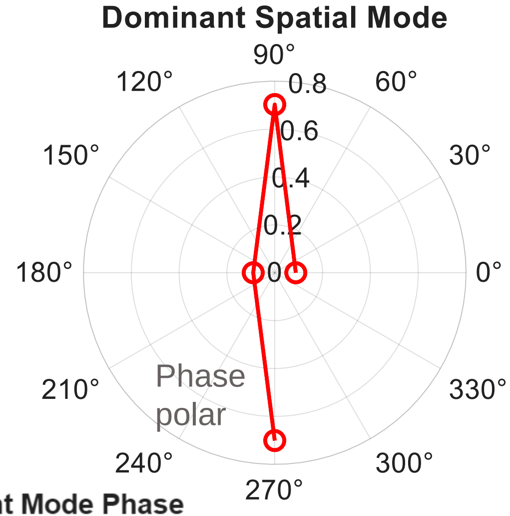
MHD modes with SVD analysis

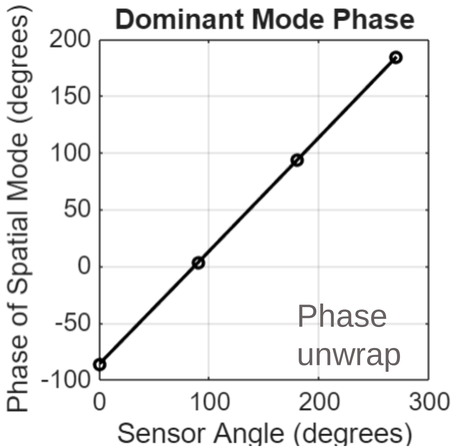
- ❑ **Perform SVD :**

$$Y = U\,S\,V^{\top}$$









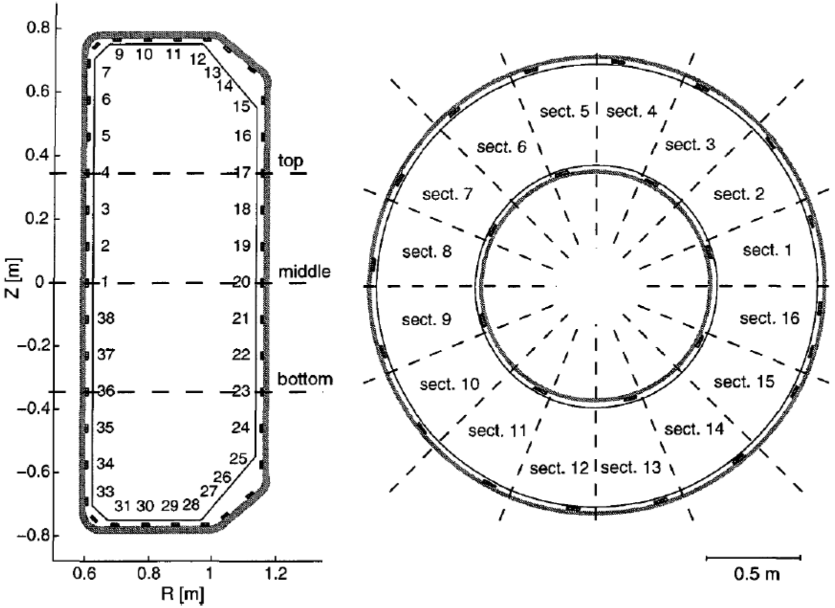
Complex Spatial Mode

$$pc1 + i * pc2;$$

- ❑ **Mode Frequency** applying ***fft*** to **U(:,1)**
- ❑ **Mode number** (periodicity) applying ***fft*** to **V(:,1)**

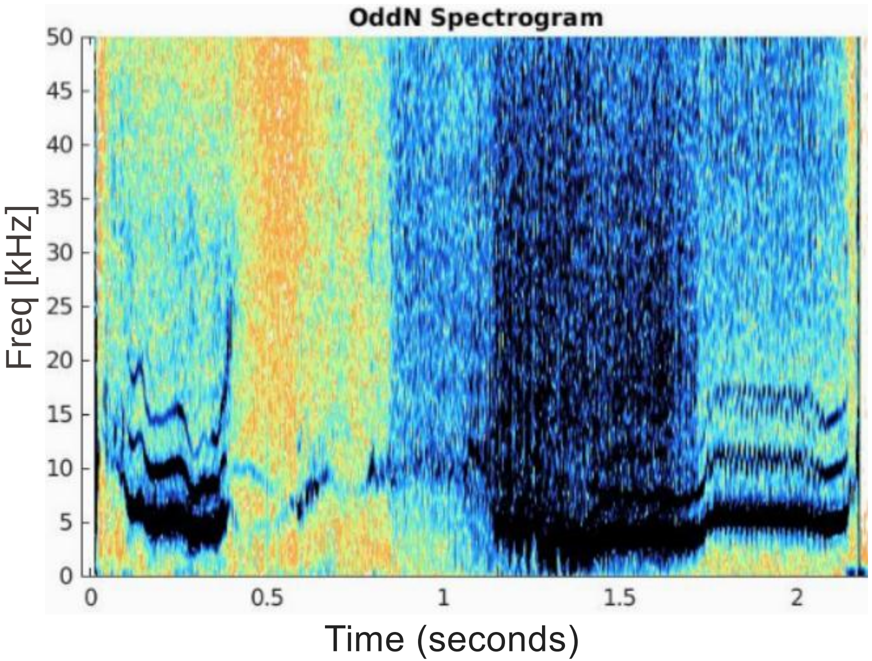
Swiss Plasma Center

MHD modes with SVD analysis



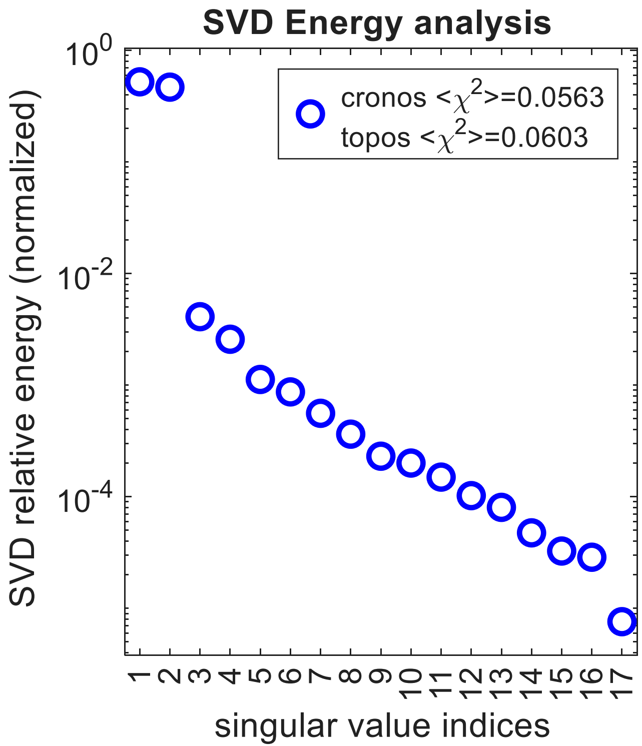
- ❑ Toroidal and poloidal arrangement of the magnetic probes on TCV

- ❑ Toroidal mode with “Odd” periodicity

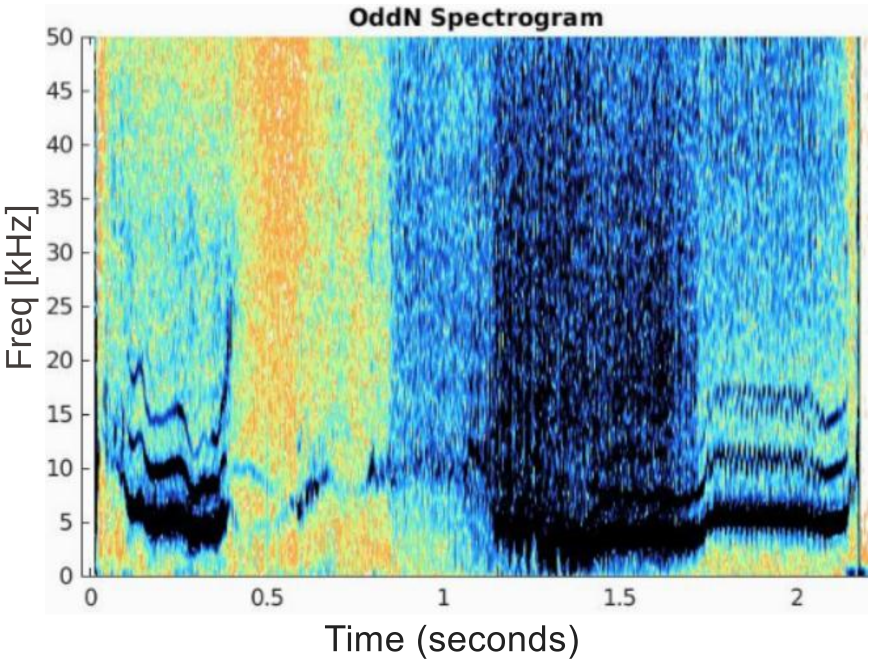


EPFL

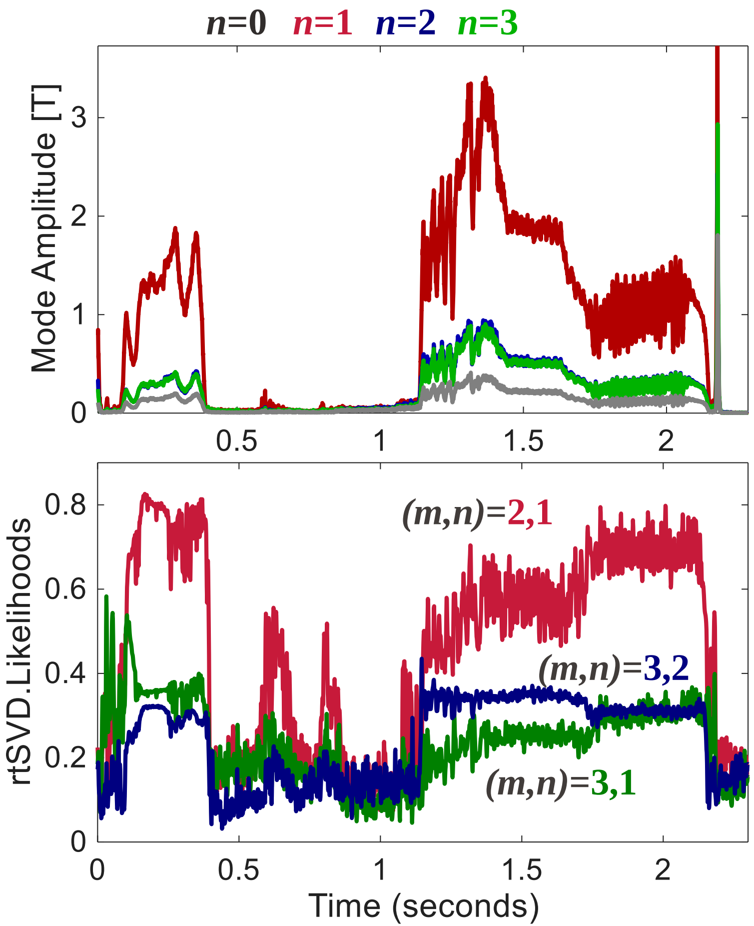
MHD modes with SVD analysis



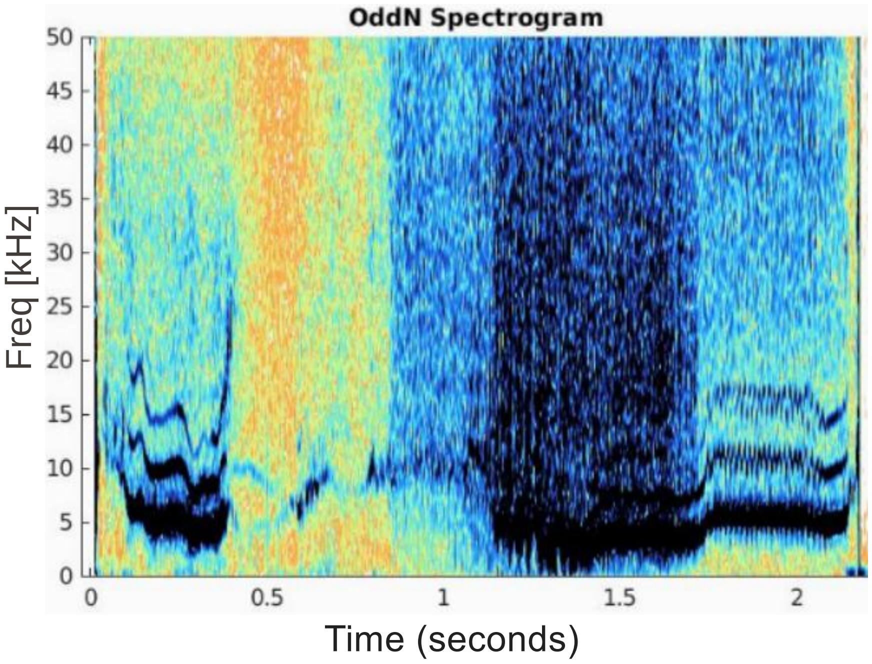
- Toroidal mode with "Odd" periodicity



MHD modes with SVD analysis

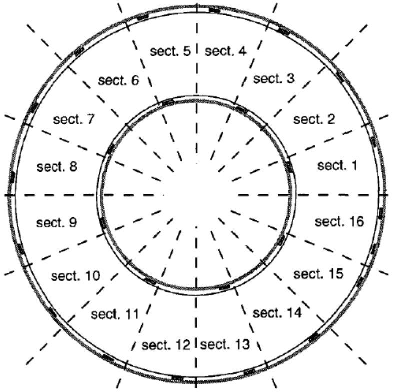


❑ Toroidal mode with “Odd” periodicity



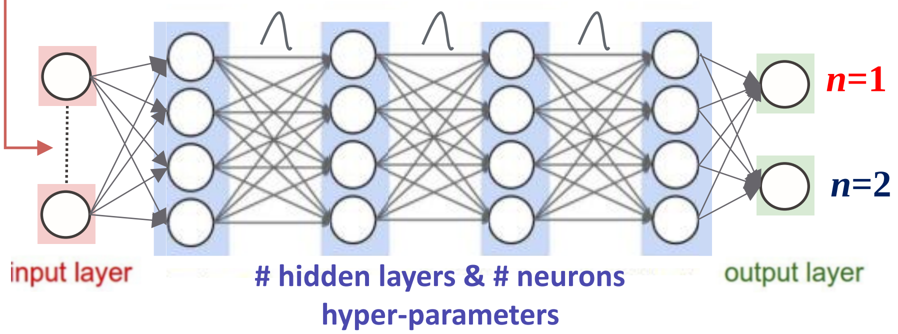
Swiss Plasma Center

MHD- RT observers with Neural Networks



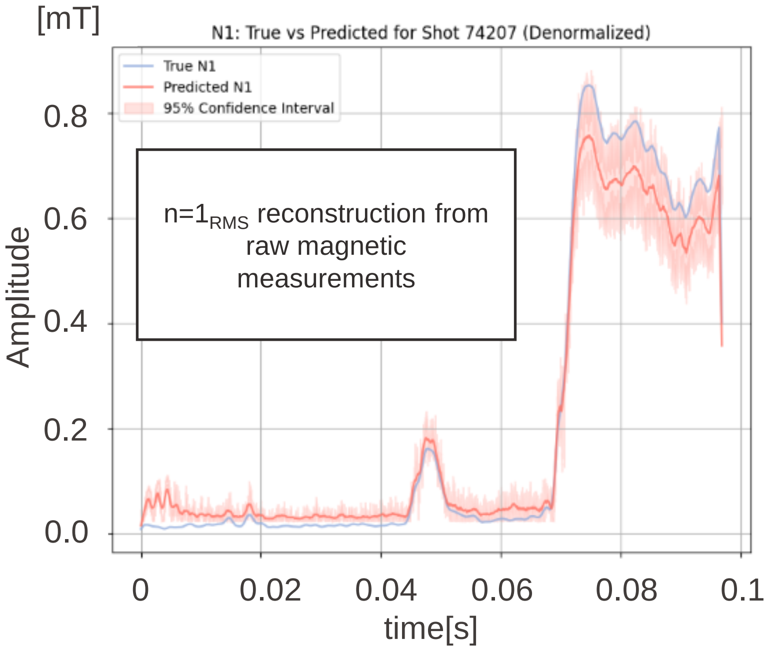
Toroidal array: 16 probes

Input normalization



Algorithm	N1_rms Time (seconds)
Spatial FFT	4.970490
Our Algorithm	0.000265

***REF**: L. Harrison, J.P. Svantner, A. Pau*



A probabilistic perspective

Bayesian inference

- **Bayes' Theorem** that describes how to update the probability of an event (or hypothesis) based on new evidence or information.
- What do we mean with **Bayesian Inference**?

Given a dataset:

$$\mathcal{D}: \left(\boldsymbol{x}^{[1]}, y^{[1]}\right), \left(\boldsymbol{x}^{[2]}, y^{[2]}\right), \dots, \left(\boldsymbol{x}^{[N]}, y^{[N]}\right)$$

Posterior, Likelihood, Prior, Evidence or Marginal:

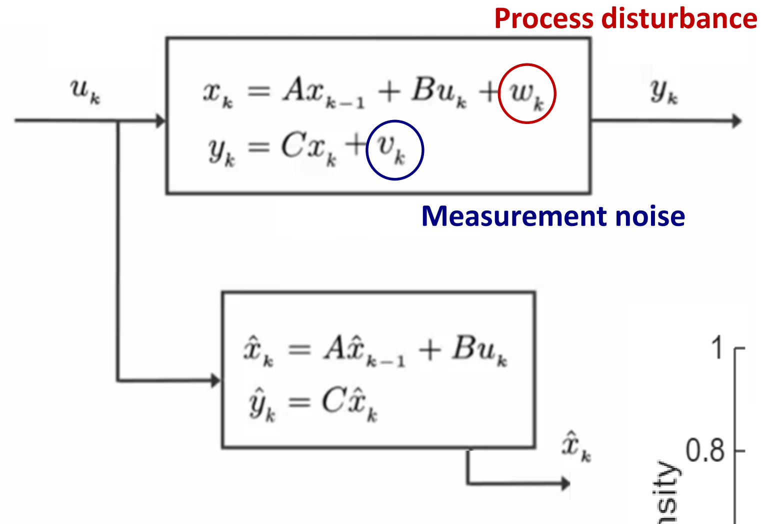
$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta})}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta})}{\int P(\mathcal{D}, \boldsymbol{\theta}')\, p(\boldsymbol{\theta}')\, d\boldsymbol{\theta}'}$$

$\boldsymbol{\theta}$ is an unknown random variable

- The **posterior** gives an indication of the **uncertainty** about our fitting parameter $\boldsymbol{\theta}$ given the data $\mathcal{D}$, according to the **prior knowledge** we have.
- Extremely powerful for **online learning**:
  - $P(\boldsymbol{\theta}|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}|\mathcal{D}_{1:t-1})$

Swiss Plasma Center

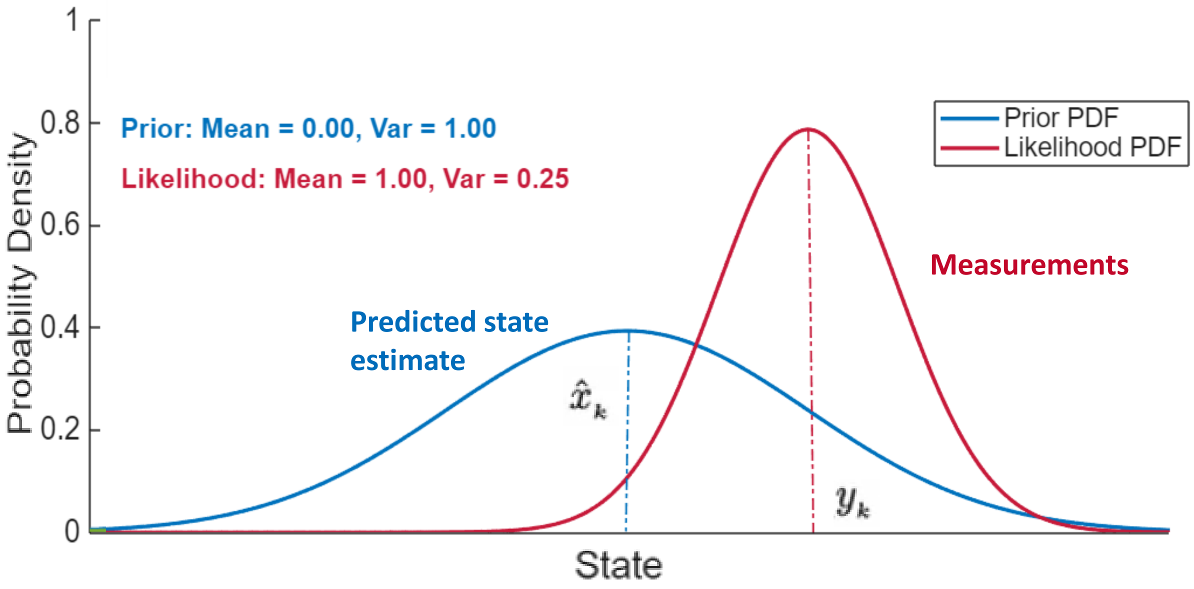
Behind Kalman filters: Bayesian inference



Stochastic state observer

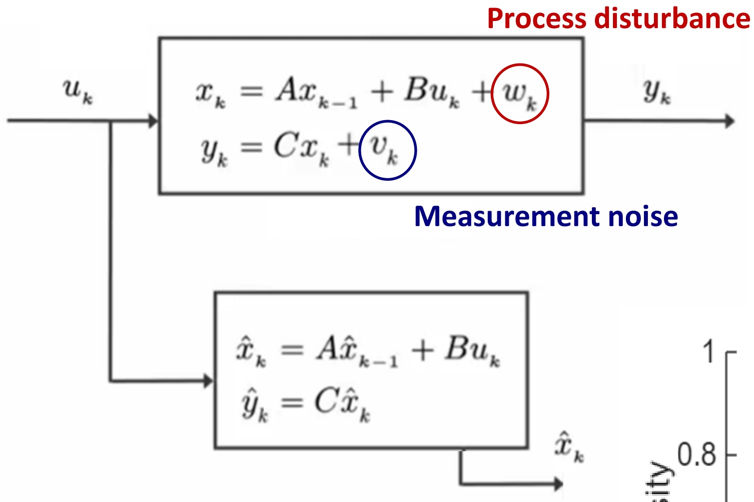
$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$$

A priori estimate



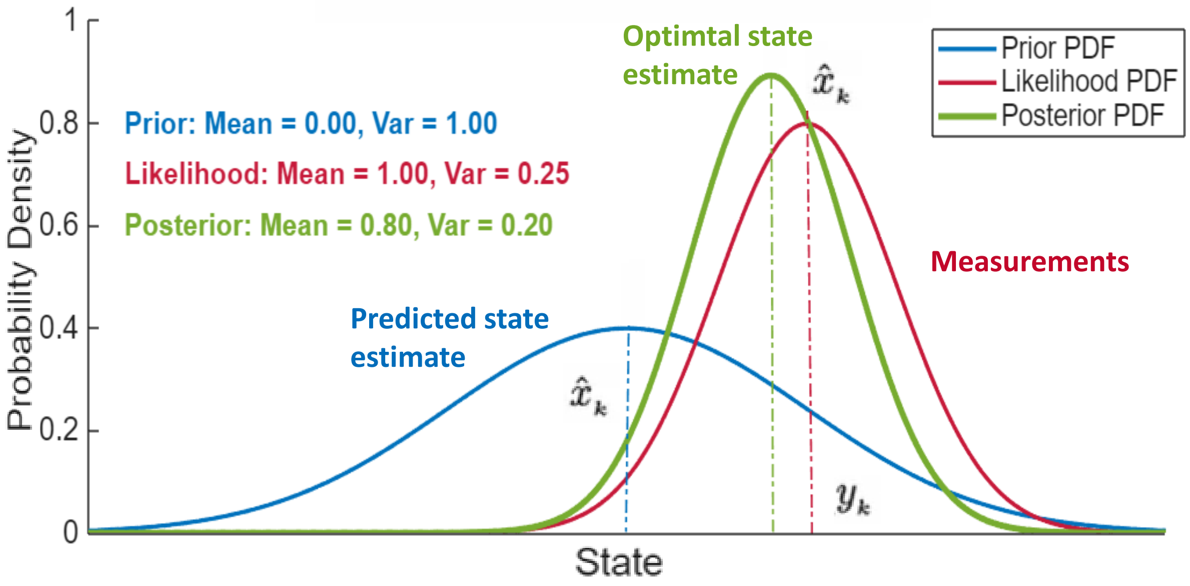
EPFL

Behind Kalman filters: Bayesian inference

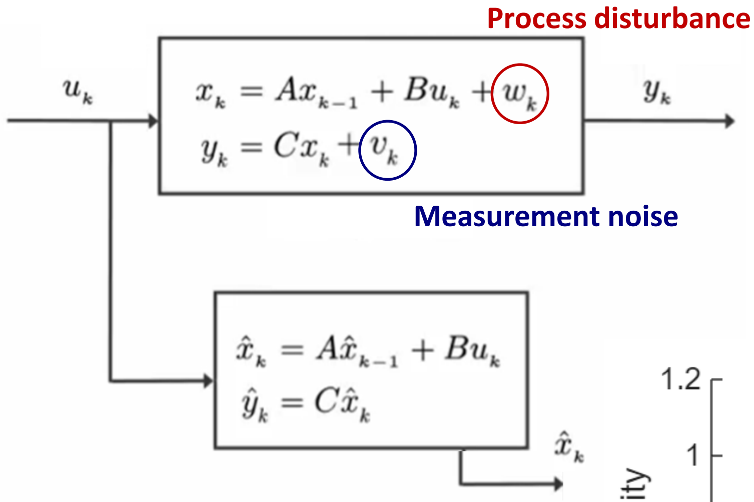


$$\hat{x}_k = \underbrace{A\hat{x}_{k-1} + Bu_k}_{\text{A priori estimate}} + K_k(y_k - C\underbrace{(A\hat{x}_{k-1} + Bu_k)}))$$

A priori estimate

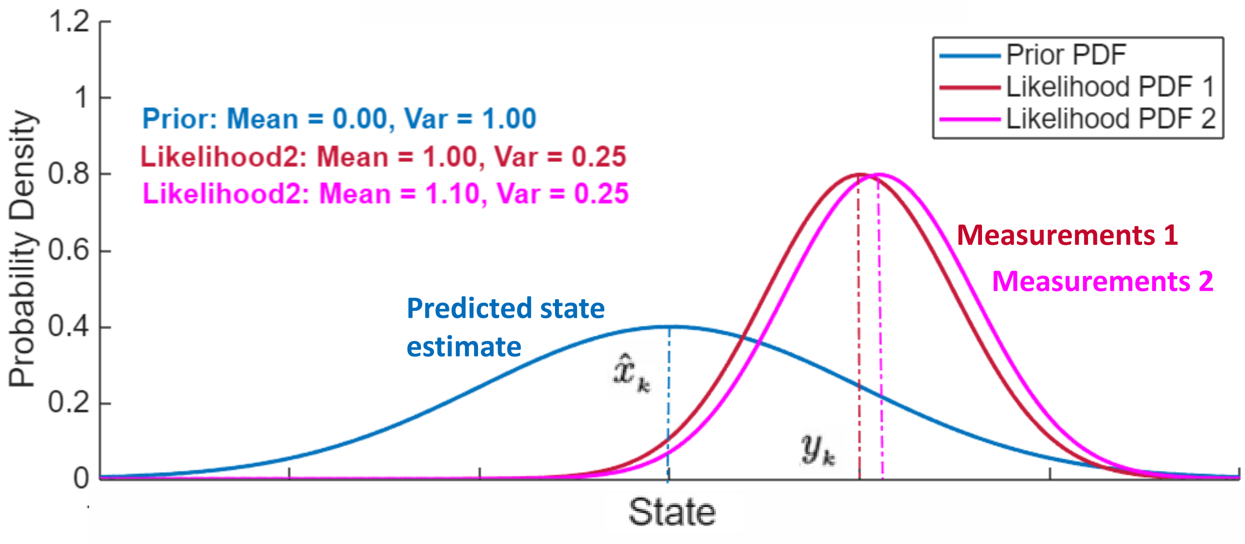


Behind Kalman filters: Bayesian inference

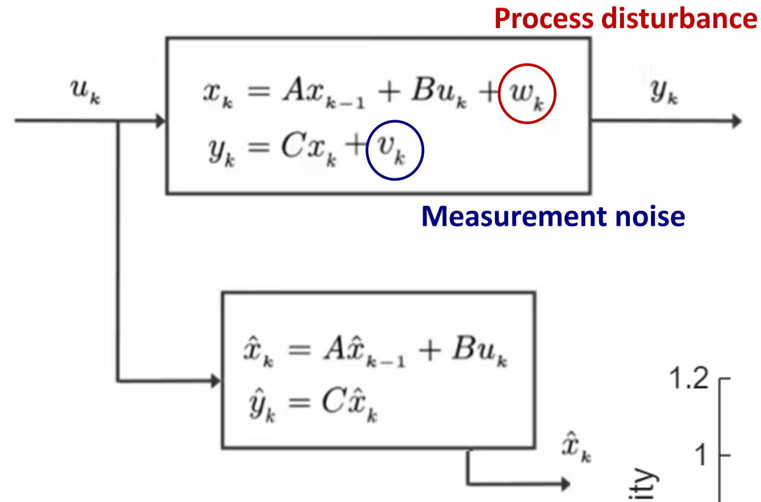


$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$$

A priori estimate

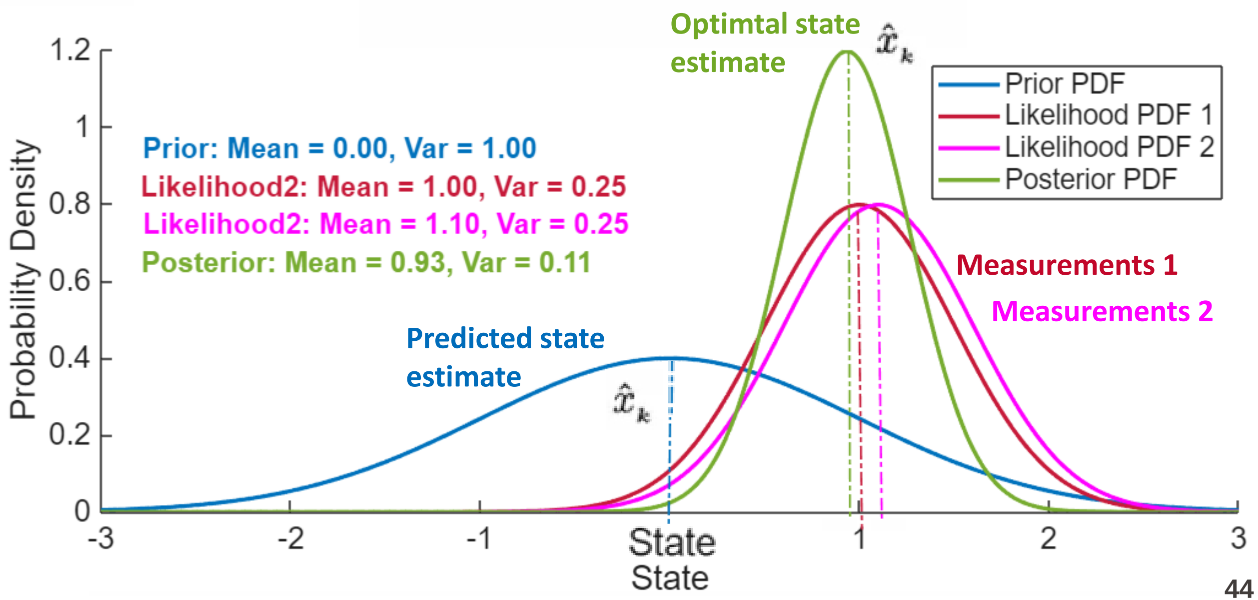


Behind Kalman filters: Bayesian inference

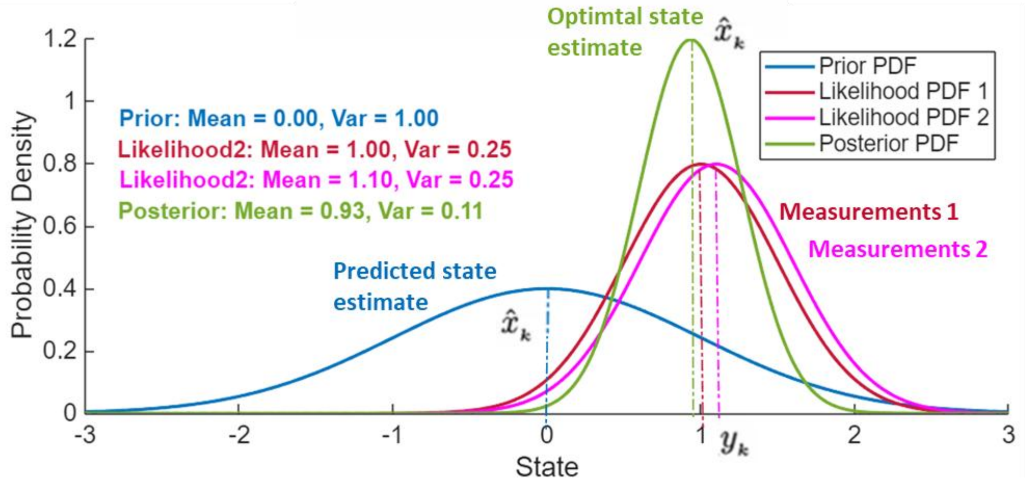


$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$$

A priori estimate



Behind Kalman filters: Bayesian inference



$$\begin{cases} f(x) = \dfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\dfrac{(x-\mu)^2}{2\sigma^2}\right) \\ \tau = \dfrac{1}{\sigma^2} \end{cases}$$

$$\tau_{\text{posterior}} = \tau_{\text{prior}} + \tau_{\text{likelihood}}$$

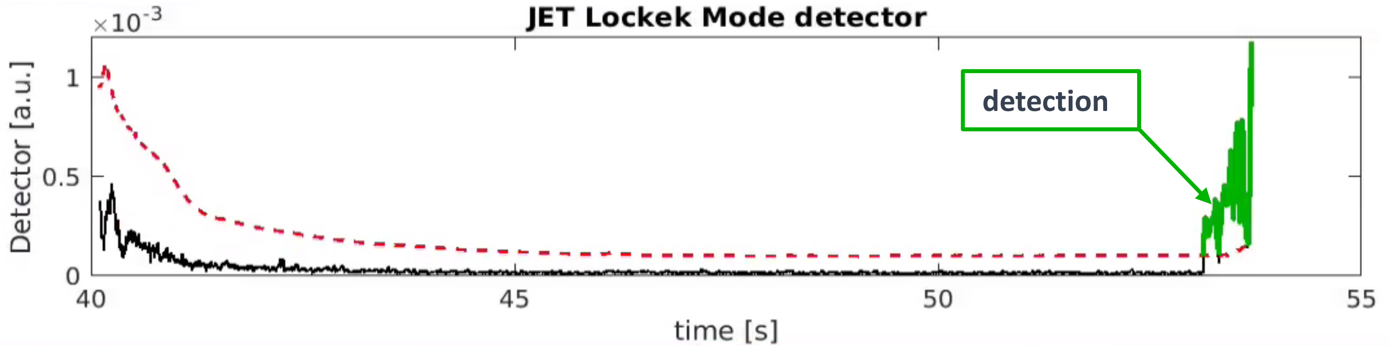
$$\mu_{\text{posterior}} = \frac{\tau_{\text{prior}}\mu_{\text{prior}} + \tau_{\text{likelihood}}\mu_{\text{likelihood}}}{\tau_{\text{prior}} + \tau_{\text{likelihood}}}$$

prior distribution: $\mathcal{N}(\mu_1, \sigma_1^2)$

likelihood distribution: $\mathcal{N}(\mu_2, \sigma_2^2)$

$$\mathcal{N}\left(\frac{\mu_1\tau_1 + \mu_2\tau_2}{\tau_1 + \tau_2}, \frac{1}{\tau_1 + \tau_2}\right)$$

Bayesian inference: event-detection problem



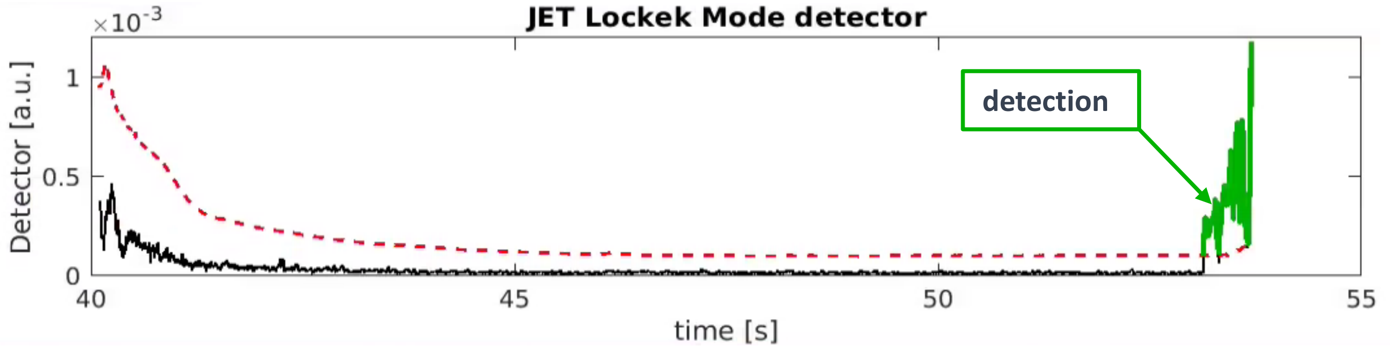
- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges exhibits a Locked Mode

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta})}{P(\mathcal{D})}$$

- Consider the case where we run a discharge, and the locked mode detector triggers an alarm.
  - What is the probability that there was a locked mode?

Swiss Plasma Center

Bayesian inference: event-detection problem

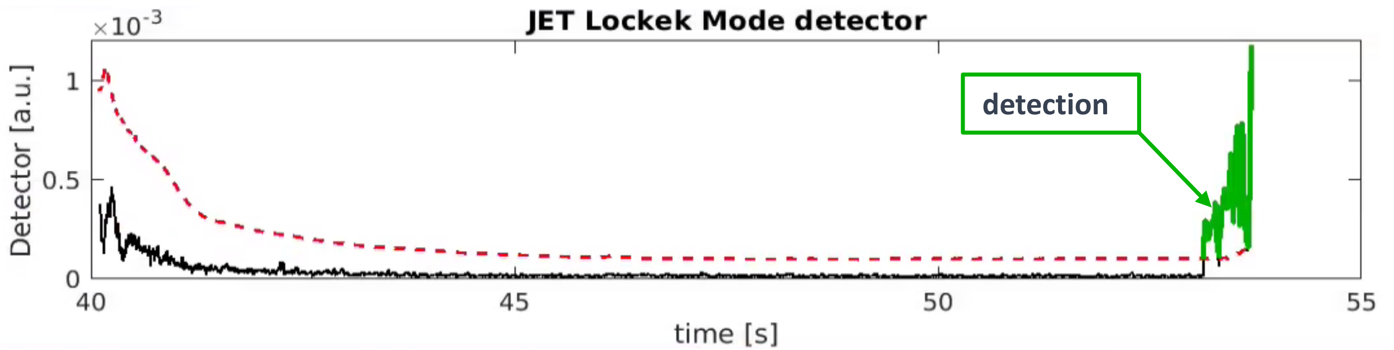


- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges exhibits a Locked Mode
    - What is the probability that there was actually a locked mode?

$$P(LM|detect) = \frac{\overset{0.99}{P(detect|LM)} \cdot \overset{0.02}{P(LM)}}{\underset{?\%}{P(detect)}}$$

EPFL

Bayesian inference: event-detection problem



- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges exhibits a Locked Mode
    - What is the probability that there was actually a locked mode?

0.99 **0.02**

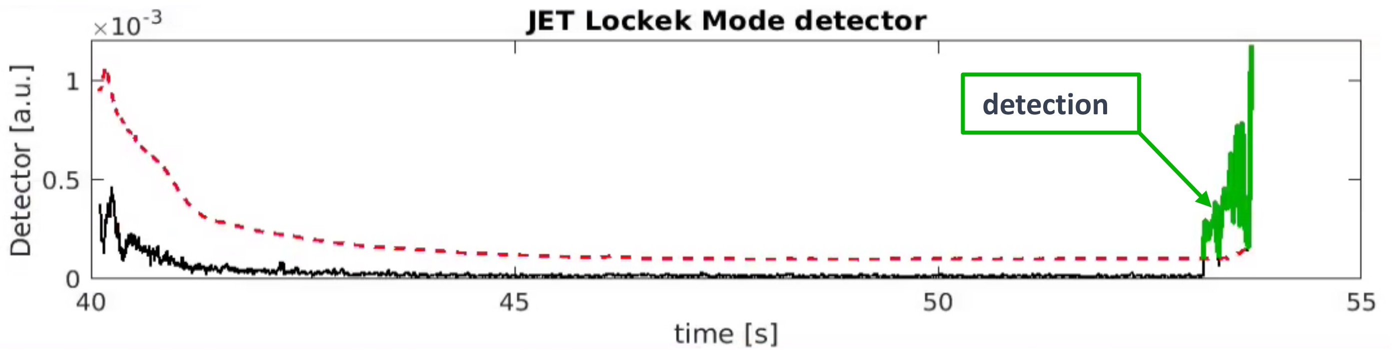
$$P(LM|detect) = \frac{P(detect|LM) \cdot P(LM)}{P(detect)}$$

?%

$$P(detect) = P(detect \mid LM) \cdot P(LM) + P(detect \mid {\sim}LM) \cdot P({\sim}LM)$$

Swiss Plasma Center

Bayesian inference: event-detection problem



- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges exhibits a Locked Mode
    - What is the probability that there was actually a locked mode? **95%**

$$P(LM|detect) = \frac{\overset{0.99}{P(detect|LM)} \cdot \overset{0.02}{P(LM)}}{\underset{?\%}{P(detect)}}$$

$$P(detect) = \overset{0.99}{P(detect\,|\,LM)} \cdot \overset{0.02}{P(LM)} + \overset{0.001}{P(detect\,|\,{\sim}LM)} \cdot \overset{1-0.02}{P({\sim}LM)}$$

$$P(LM|detect) = \frac{\overset{0.99}{P(detect|LM)} \cdot \overset{0.02}{P(LM)}}{\underset{0.0208}{P(detect)}} = {\sim}0.953$$

Swiss Plasma Center

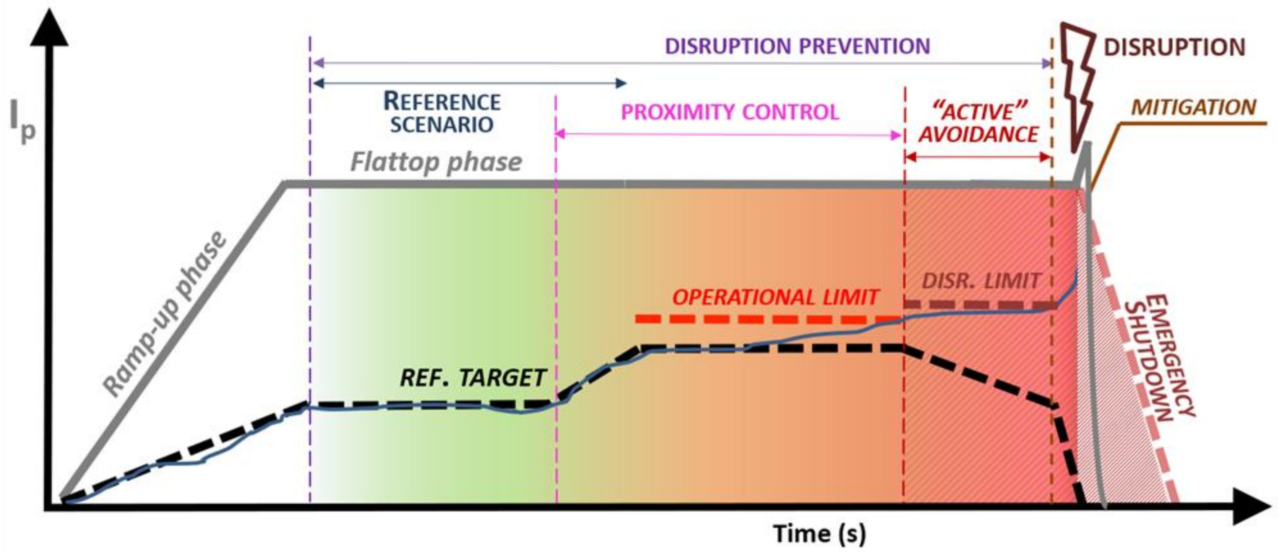
Plasma trajectories & Latent variable models

Swiss Plasma Center

Latent variable models for plasma state monitoring

- **DISRUPTION PREVENTION**

Break down in different "control phases":



REF. SCENARIO

- keep the target scenario stable again disturbances (ST, ELM, MHD modes, etc.)

PROXIMITY CONTROL

- keep stability while pushing performance by regulating proximity to stability & controllability boundaries

ACTIVE AVOIDANCE

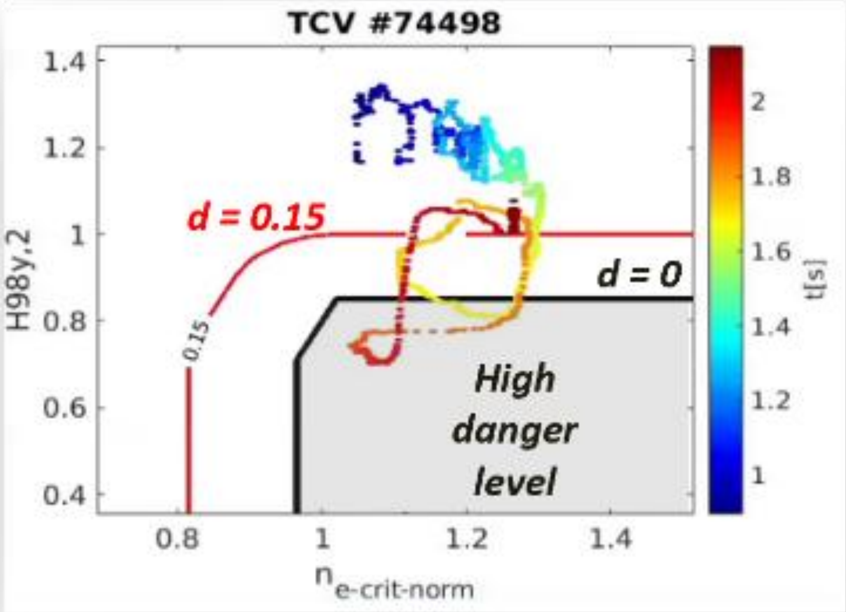
- asynchronous response when crossing operational boundaries (danger levels)

EMERGENCY SHUTDOWN

- Fast controlled shutdown
- mitigation

Plasma trajectories in physics phase spaces

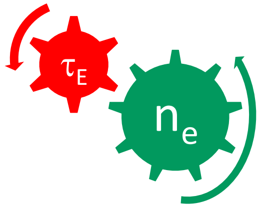
DISRUPTION



- **High-performance Density Limit** dynamics described through trajectories in a physics-based **"state space" [H98y,2-$n_{e\text{-crit-norm}}$]**

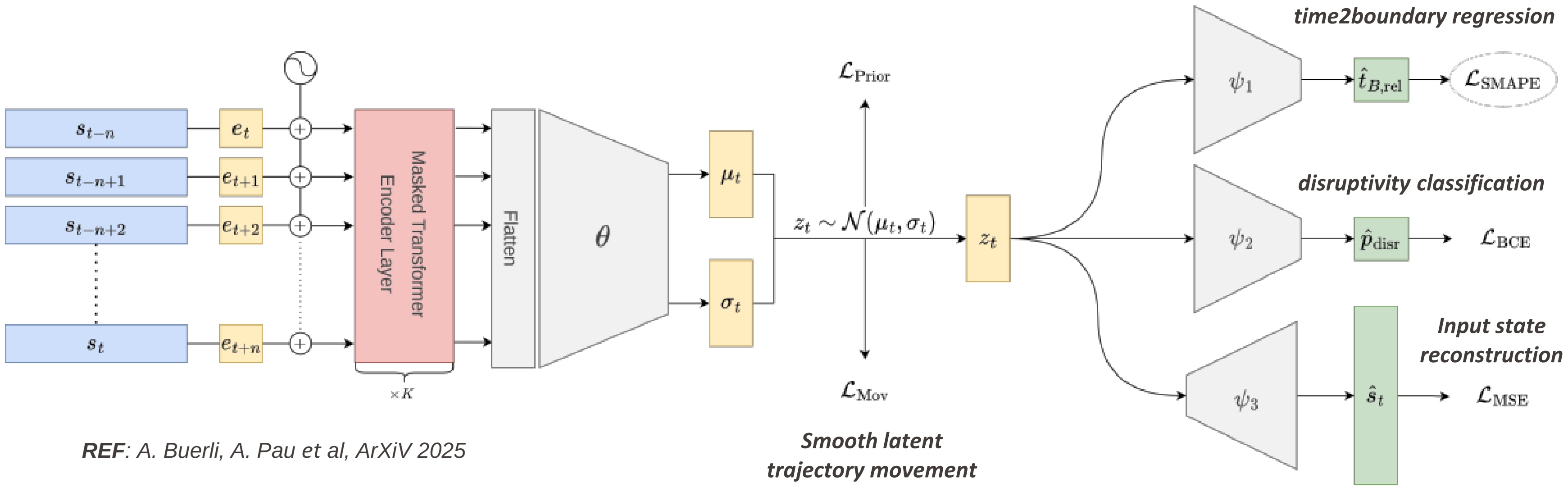
REF: [M. Bernert PPCF 2015]

- **Conflicting control objectives** in high density regimes



- **Proximity to boundaries** with increasing **probability of disruptions** (H-L, MARFE, etc.)

Latent variable models for plasma state monitoring



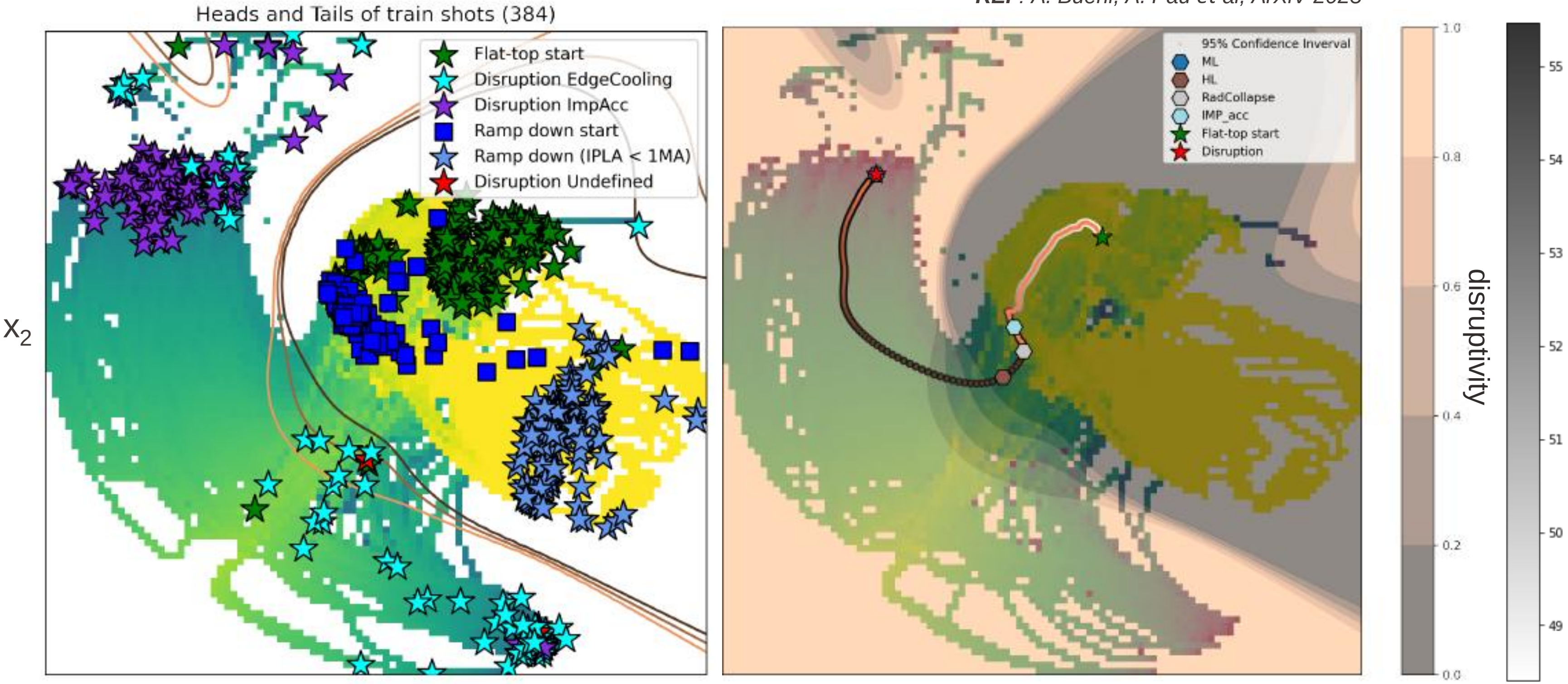
REF: *A. Buerli, A. Pau et al, ArXiV 2025*

- **Sequence-based** model: a variational autoencoder (transformer, GPT-alike architecture)
- **Multi-task learning:** by learning tasks jointly(supervised and unsupervised), the model can discover common features or structures across tasks (shared representation).

Swiss Plasma Center

Plasma State Monitoring with sequence-based DL

REF: A. Buerli, A. Pau et al, ArXiV 2025



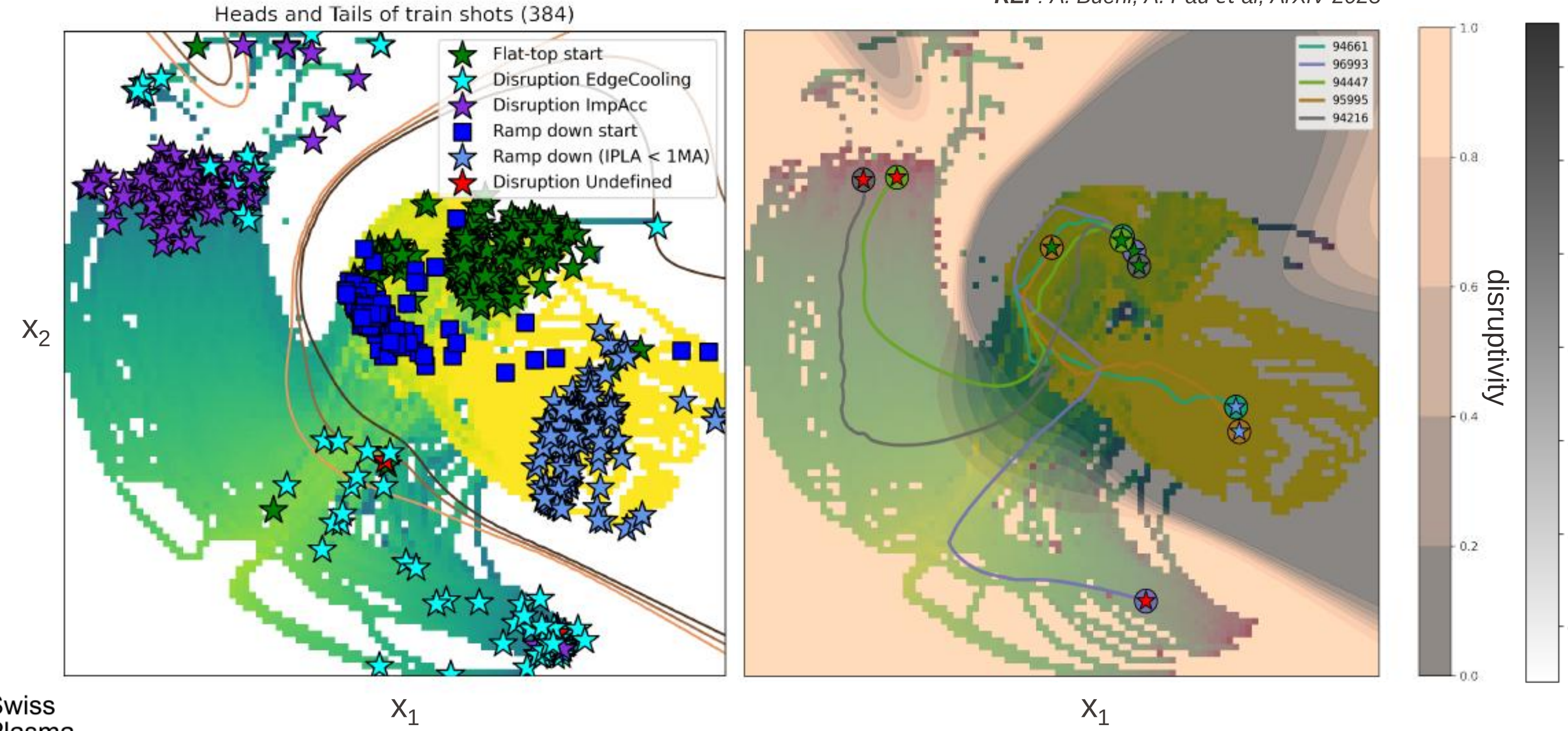
x_2

x_1 x_1

•The goal is to discover and learn the **hidden/latent** variables or states that better explain or predict observable signals, transitions, or anomalies in plasma behavior.

Plasma State Monitoring with sequence-based DL

REF: *A. Buerli, A. Pau et al, ArXiV 2025*

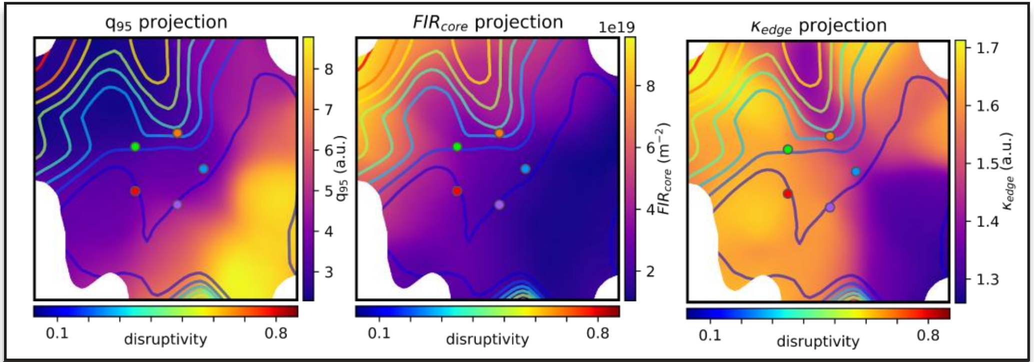


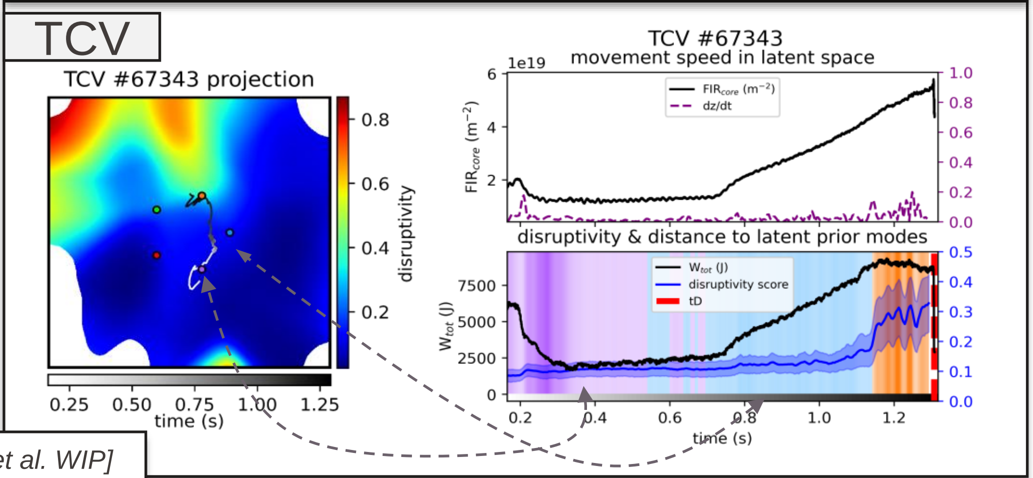
•The goal is to discover and learn the **hidden/latent** variables or states that better explain or predict observable signals, transitions, or anomalies in plasma behavior.

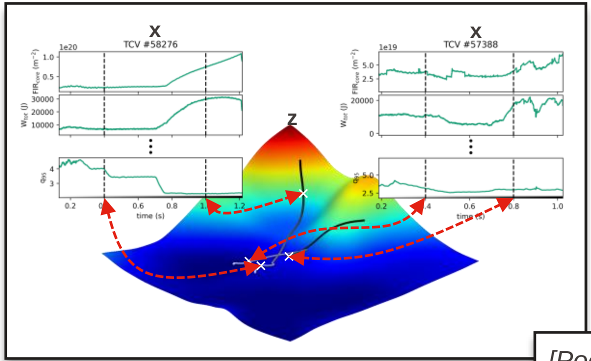
Latent variable models for disruption monitoring

Sequential VAE with multimodal prior

- Project **disruptive boundaries** & physics quantities to inspect connections
- Project full discharges to track proximity to disruption
- *Future: Investigate identified modes in posterior distribution*
- *Future: Discretize projections as sequences of states*





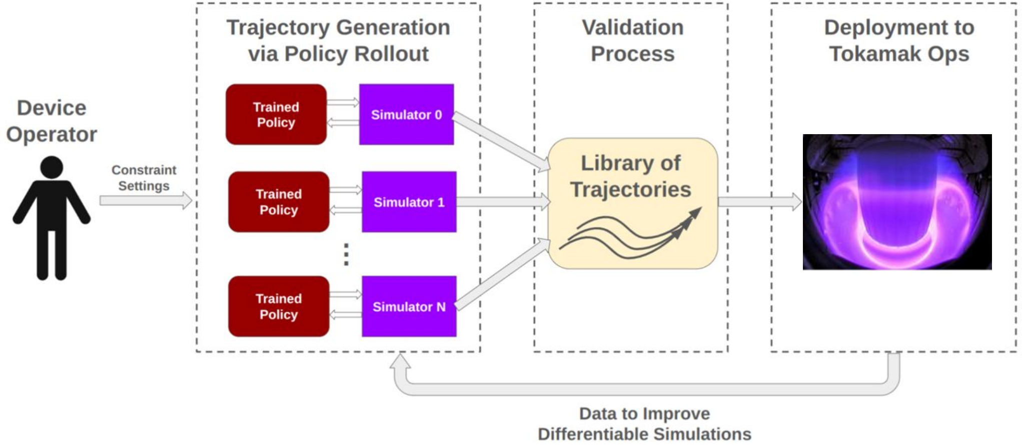


[Poels et al. WIP]

Plasma trajectories optimization with physics constraints

COLLABORATION WITH MIT

- Scientific machine learning for building simulators that **combine physics + machine learning**
- Reinforcement learning to design **trajectories** and **controllers** to meet operator specifications that are robust to **physics uncertainty**



A. Wang, A. Pau, et al. (paper to be submitted)

- **Trajectory**: sequence of states, actions, and rewards that an agent experiences as it interacts with the environment.
- **Neural State Space Models** (**NSSM**) to learn the temporal dynamics of some observed quantities in response to actions (physics structure and data-driven models).

Swiss Plasma Center

Plasma trajectories optimization with physics constraints

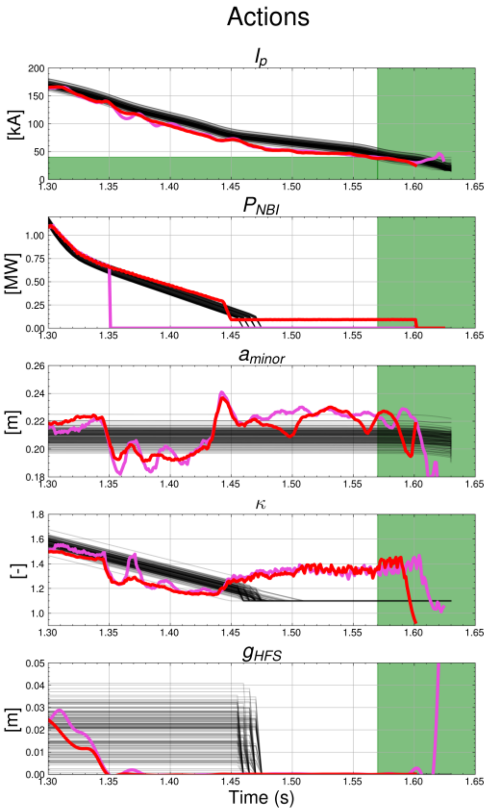
COLLABORATION WITH MIT

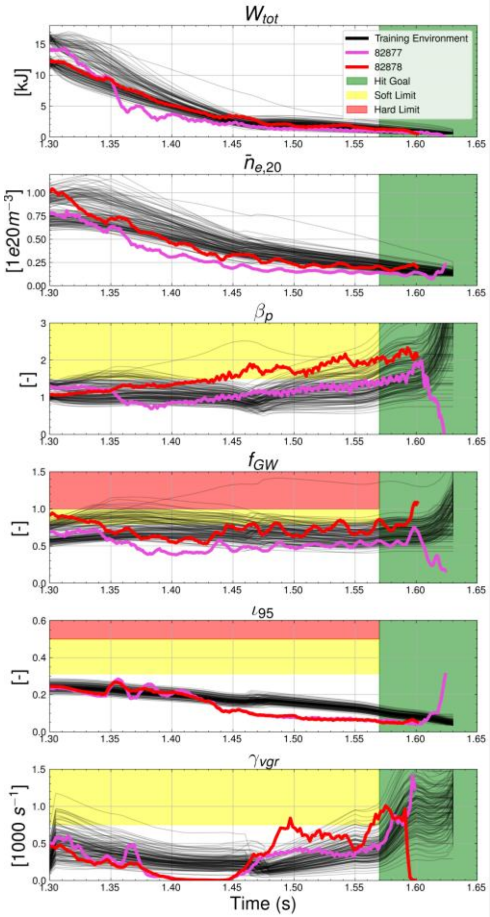
Reward function

$$r(\mathbf{x}(t),\mathbf{a}(t)) = \underbrace{-c_{time}}_{\text{Penalty for time}} - \underbrace{c_W W_{tot}(t) - c_{I_p} I_p(t)}_{\text{Penalty for current and energy}} - \underbrace{\sum_{i=1}^{n_{soft}} c_{soft} s_i(\mathbf{x}(t))}_{\text{Soft chance-constraints}} - \underbrace{\sum_{i=1}^{n_{hard}} c_{hard} h_i(\mathbf{x}(t))}_{\text{Hard chance-constraints}}$$

Reward function parameters

Category	Parameter	Value
Hard Limits	f_{GW}	1.0
	ι_{95}	0.5
Soft Limits	f_{GW}	0.8
	β_p	1.75
	γ_{vgr}	0.75
	ι_{95}	0.313
Parameters	c_{time}	5.0
	c_{I_p}	1.0
	c_W	1.0
	c_{soft}	1.0×10^3
	c_{hard}	5.0×10^4





Swiss Plasma Center

Introduction to the exercises

EPFL

Control augmentation in modern Plasma Control Systems

- Magnetic control via DRL

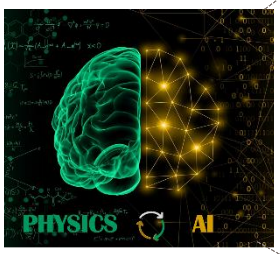
*REF: [FJ. Degrave, F. Felici et al. **Nature** 2022]*

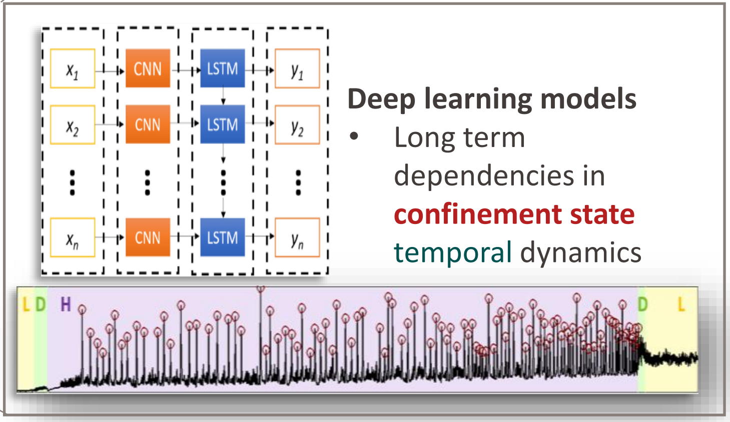
- plasma state **monitoring** and **forecasting** for control augmentation
- Detection of **off-normal events** to react with specific control tasks in **real-time**
- **Proximity** to **operational limits**

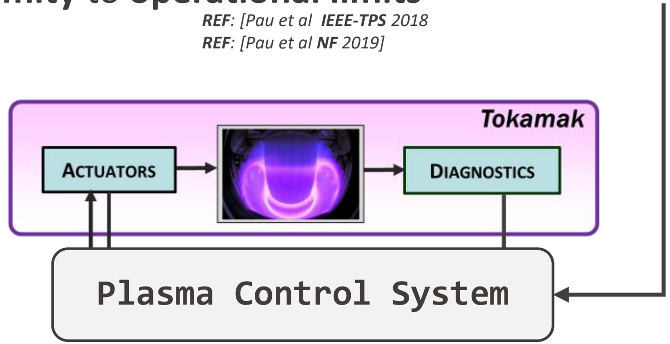
*REF: [Pau et al **IEEE-TPS** 2018*
*REF: [Pau et al **NF** 2019]*

*REF: [F. Matos et al. **NF** 2020]*
*REF: [F. Matos et al. **NF** 2021]*
*REF: [G. Marceca et al. **NeurIPS** 2021]*

EVENT DETECTION







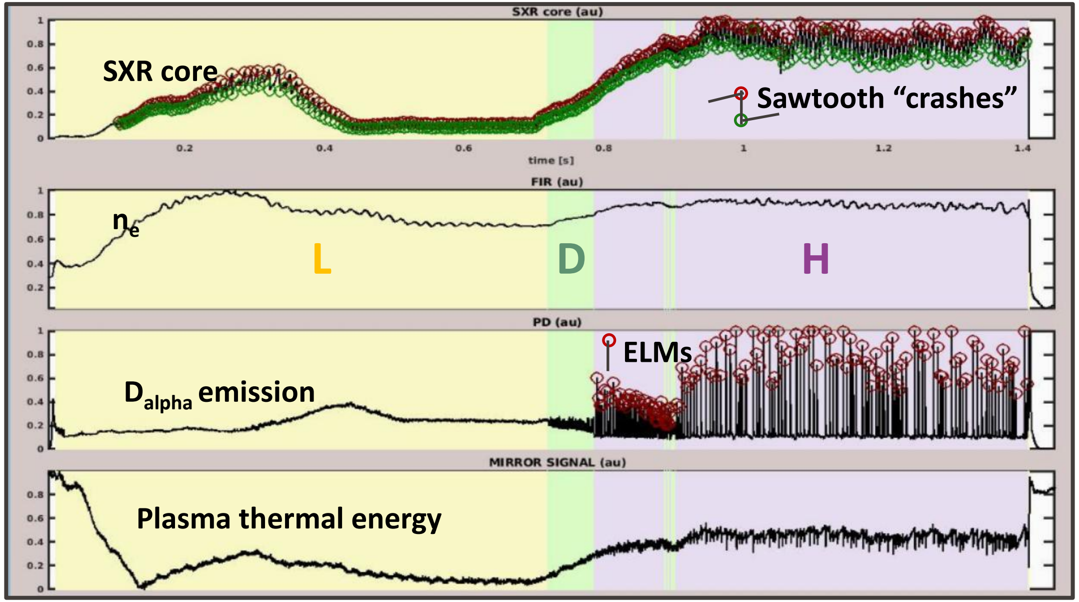
…combination & integration of:

- **Physics-, model-** & **ML**-based approaches

EPFL

Event detection and plasma state classification

- **Plasma confinement states [L=Low; D=Dithering; H=High]**

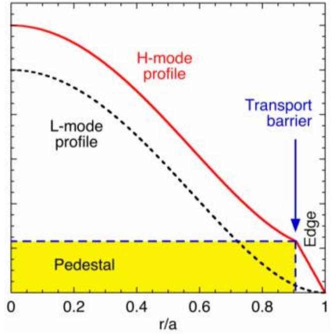


Time [s]

LHD phases from validated-labeled file
A total of 138 ELMs were found in the loaded data
A total of 249 ST_MDs were found in the loaded data

An experiments have potentially **hundreds of events**….

- Plasma can evolve in one of several possible **confinement states** (typical categorization in L=Low; D=Dithering; H=High).
- By applying **sufficient heating power**, the plasma spontaneously transitions from a low to a high confinement state
- **H-mode:** improved energy confinement state with **reduced particles and energy transport** outwards formation of an edge transport barrier (**ETB**) and a cyclic MHD instability called Edge Localized Modes (**ELMs**).



Backup slides

ML foundations: a probabilistic perspective

- *We call* ***inference***[*] *the procedure with which we quantify of the* *uncertainty* *or confidence in the estimate* $\hat{\boldsymbol{\theta}}$. $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, \mathcal{L}\,(\mathcal{D}|\boldsymbol{\theta})$
- *On a probabilistic perspective we reason in terms of* ***Probability Density Estimation*** *for the joint probability distribution of our dataset* $\mathcal{D}$ *(a sample from the population)*
- *Under* ***i.i.d assumption*** *(training examples sampled independently and identically from the population representing the input domain* $\mathcal{D}$*:*

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|x_n, \boldsymbol{\theta}) \qquad LL(\mathcal{D}|\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{n=1}^{N} p(y_n|x_n, \boldsymbol{\theta})$$

- *Therefore the optimization problem can be seen as maximizing a probability*

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \, p\,(\mathcal{D}|\boldsymbol{\theta})$$

- ***inference***[*] *in the deep learning community refers to predicting* $p(y_n|x_n, \hat{\boldsymbol{\theta}})$

ML foundations: a probabilistic perspective

- *Therefore, the optimization problem translates in maximizing the* ***Log-Likelihood (LL)***,

$$LL(\mathcal{D}|\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{n=1}^{N} p(y_n|x_n, \boldsymbol{\theta}) \qquad \widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, LL\,(\mathcal{D}|\boldsymbol{\theta})$$

- *Which can be also seen as minimizing the* ***Negative Log-Likelihood (NNL)****:*

$$NLL(\mathcal{D}|\boldsymbol{\theta}) \triangleq -\log p(\mathcal{D}|\boldsymbol{\theta}) = -\sum_{n=1}^{N} p(y_n|x_n, \boldsymbol{\theta}) \qquad \widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, NLL(\mathcal{D}|\boldsymbol{\theta})$$

- *Estimating the probability density function (high uncertainty if the sampling distribution is small) is usually done with two common approaches:*
  - ***Maximum Likelihood Estimation (MLE)***
  - ***Maximum a Posteriori (MAP)****:*

Swiss Plasma Center

EPFL

ML foundations: a probabilistic perspective

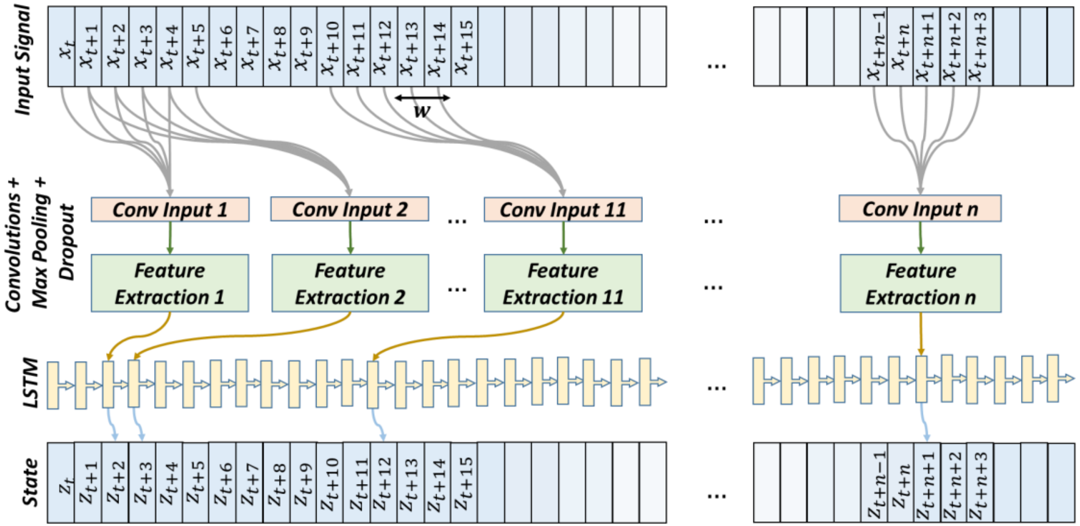
- ***Maximum Likelihood Estimation (MLE):*** *frequentist approach for estimating the set of parameters $\hat{\boldsymbol{\theta}}$ of a model by finding the values that maximize the log-likelihood $LL(\mathcal{D}|\boldsymbol{\theta})$.*
- *Interpretation: $LL(\mathcal{D}|\boldsymbol{\theta})$ describes the probability of observing the data given the model parameters $\hat{\boldsymbol{\theta}}$. The likelihood function is known if data are* ***i.i.d****.*
  $\hat{\boldsymbol{\theta}}$ resulting from MLE are the most probable values given the data.

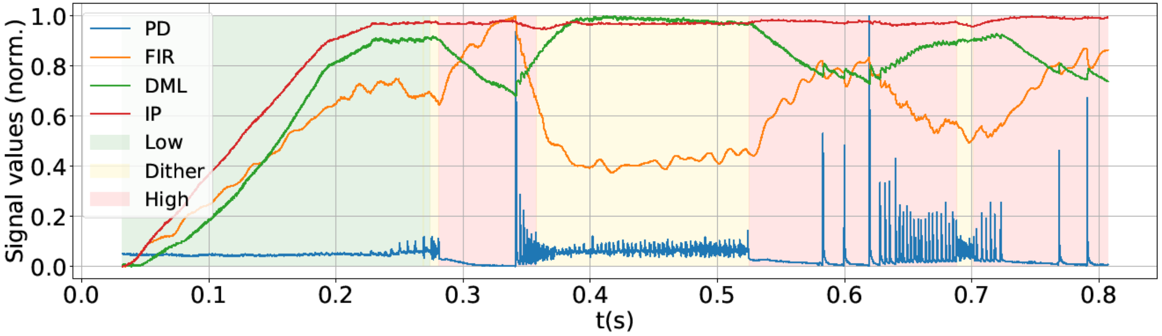
- ***Maximum a Posteriori (MAP)****: Bayesian approach for estimating the values of the parameters $\hat{\boldsymbol{\theta}}$ that maximize the posterior probability,*
- *Interpretation: MAP describes probability of the parameters given the data and allows incorporating* ***prior knowledge*** *about the parameters into the estimation process. This prior knowledge is specified as a probability distribution and allows us to account for uncertainty in the data.*

Swiss Plasma Center

Event detection and plasma confinement state classification

- **Deep Learning** model based on a **convolutional-RNN (LSTM)**
- **Probability** of the plasma of being in a given **confinement state** (accounting for temporal evolution)
- **RT implementation** (nice example of integration with physics-based models in the framework of **off-normal events handling & disruption avoidance**

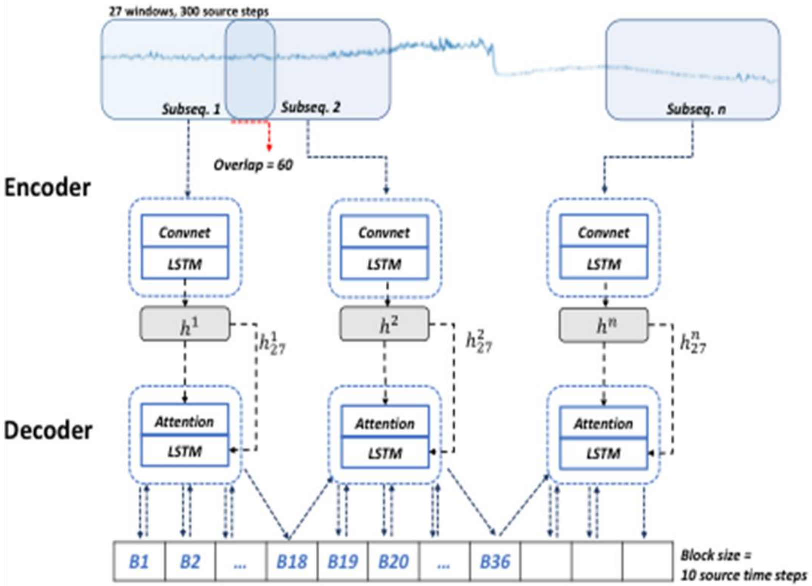




REF: *[Matos et al NF 2020]*

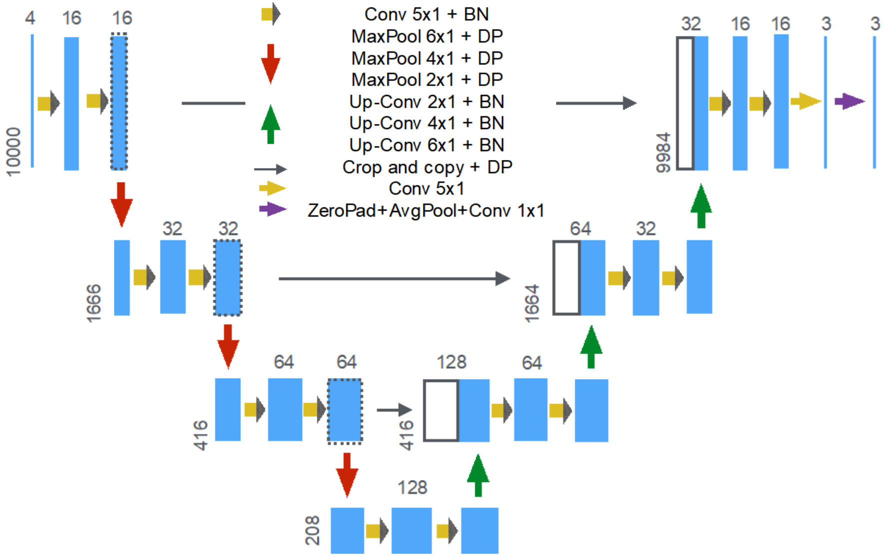
Event detection and plasma confinement state classification

- **SEQUENCE 2 SEQUENCE MODEL:**
  - Model not constrained to have same **source/target resolutions**.
  - Decoder was extended with an **attention** layer to capture **larger context** of long input sequences.



REF: [Matos et al NF 2021]

- **UTIME MODEL:**
  - **Multi-scale convolutional** structure allows to capture **patterns at different scales** present in the plasma.
  - processing the whole signal at once (offline) with the ability to see at a **wider contextual information.**



REF: [Marceca et al NEURIPS 2021]