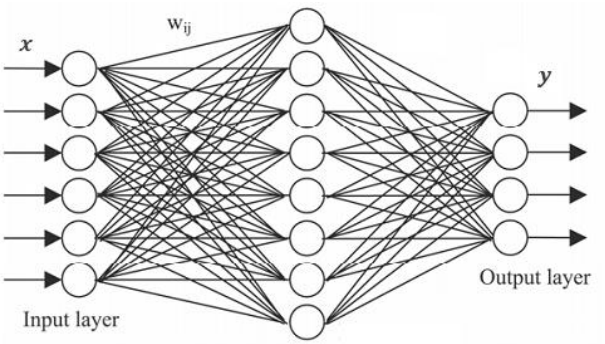
Control and Operation of Tokamaks
Machine Learning for plasma control



Alessandro Pau
alessandro.pau@epfl.ch

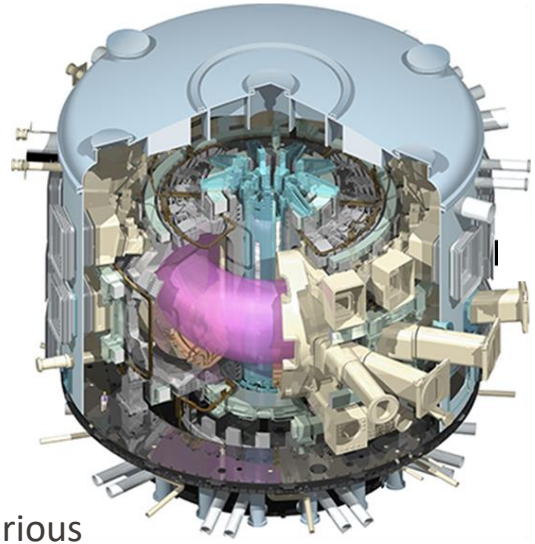
Swiss Plasma Center

16/02/2023, Lausanne

Data in fusion: a challenge in itself

- **Tokamaks** are one of the most complex system in nature…

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

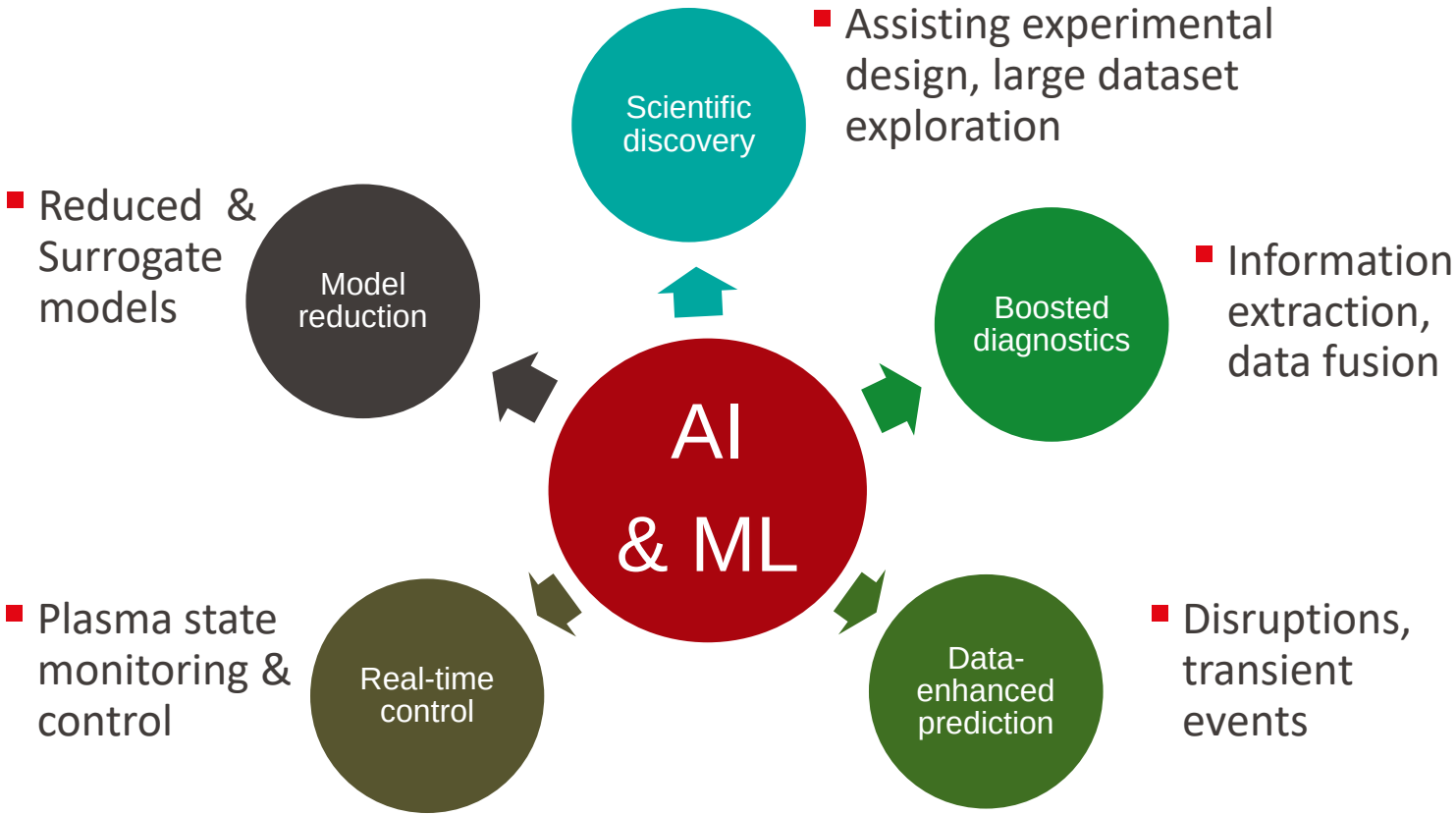


- **High-dimensionality** (many diagnostics measuring various plasma properties)

- **Multi-scales, multi-physics** (integrated tokamak modeling)

- **Heterogenous** (various formats, facility dependent data ecosystem)

Advancing fusion by leveraging AI and ML



Control augmentation in modern Plasma Control Systems

- Magnetic control via DRL

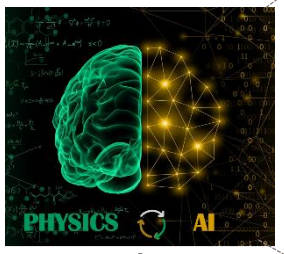
*REF: [FJ. Degrave, F. Felici et al. **Nature** 2022]*

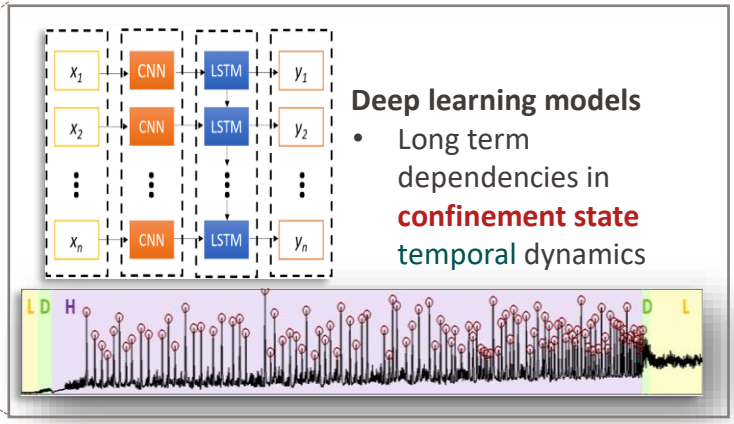
- plasma state **monitoring** and **forecasting** for control augmentation
- Detection of **off-normal events** to react with specific control tasks in **real-time**
- **Proximity** to **operational limits**

*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]*

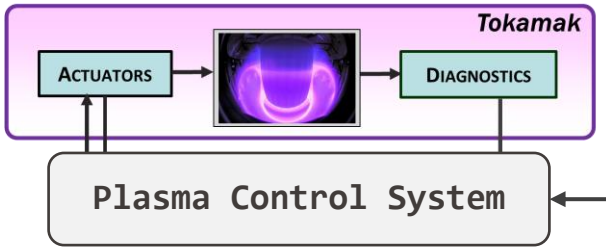
EVENT DETECTION





Deep learning models

- Long term dependencies in **confinement state** temporal dynamics

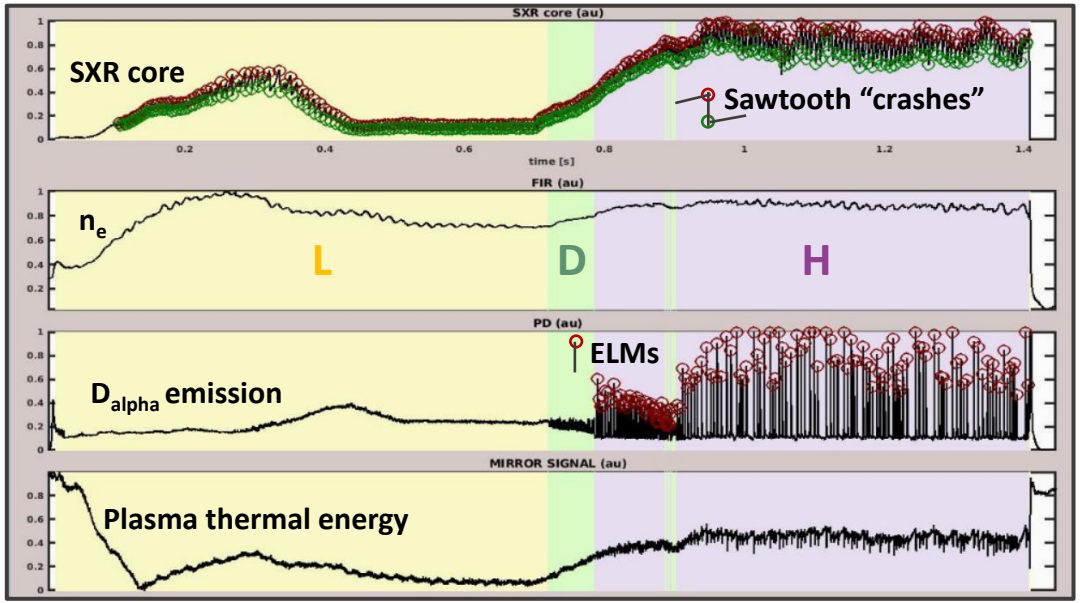


…combination & integration of:

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

Event detection and plasma state classification

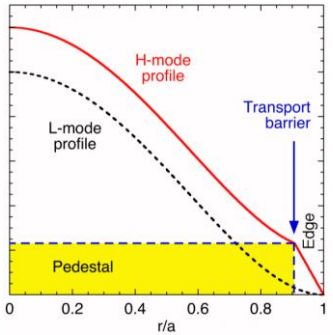
- **Plasma confinement states [L=Low; D=Dithering; H=High]**



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**….

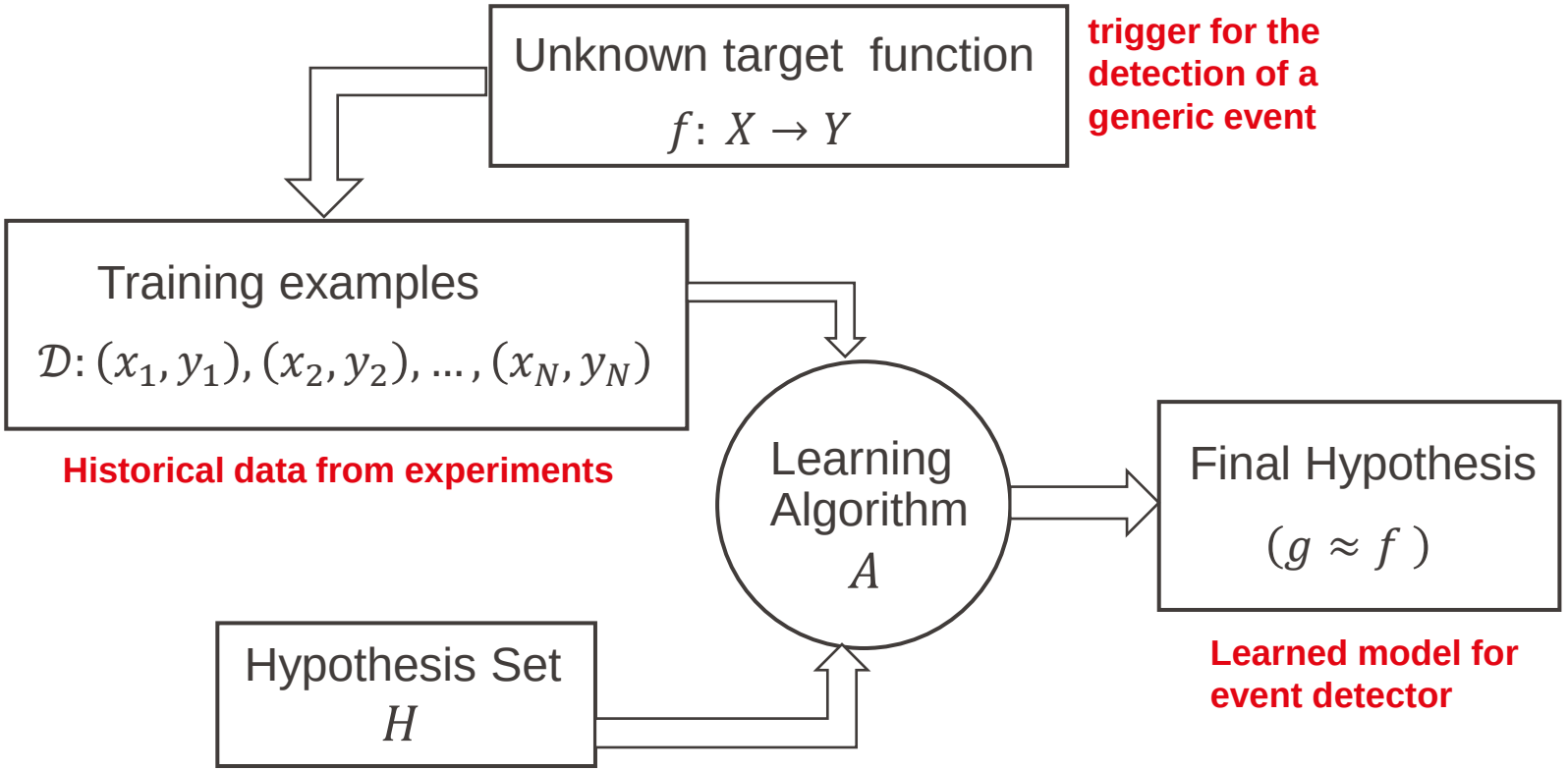
- Plasma can evolve in one of several possible **confinement states** (typical categorization in L=Low; D=Dithering; H=High).
- By applying **sufficient heating power**, the plasma spontaneously transitions from a low to a high confinement state
- **H-mode:** improved energy confinement state with **reduced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



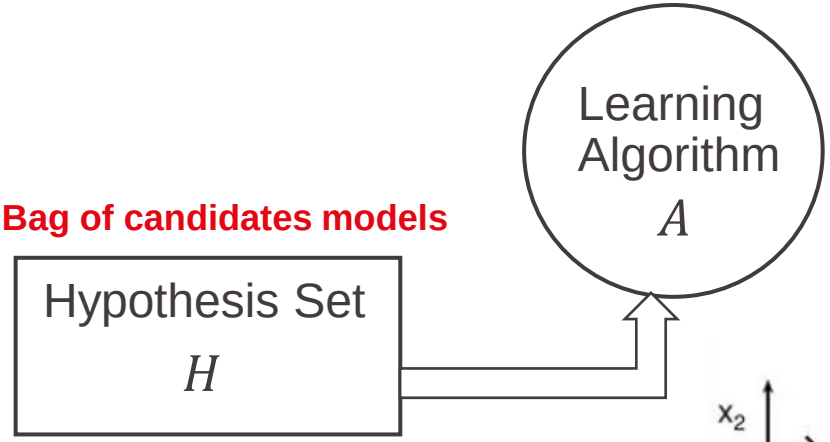
Machine Learning foundations

How can we formulate a control problem leveraging ML?

Formulation of the learning problem



Formulation of the learning problem

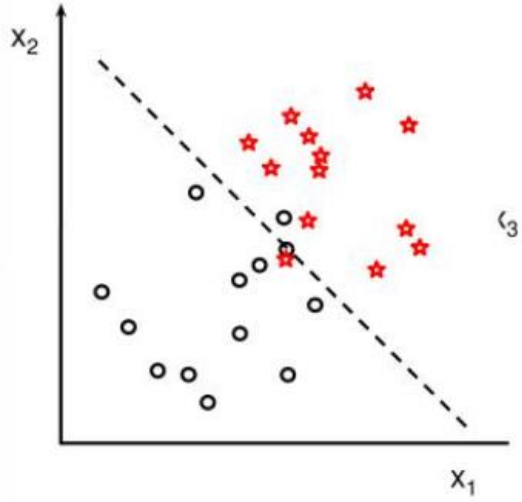


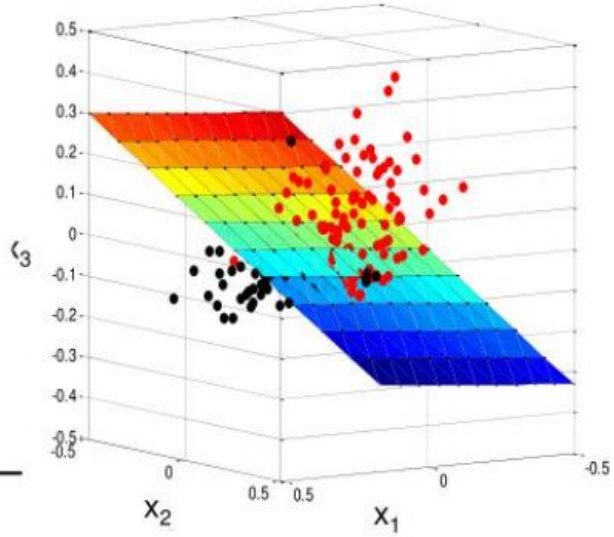
Bag of candidates models

Simple generic linear hypothesis:
Functional form *h(x)*

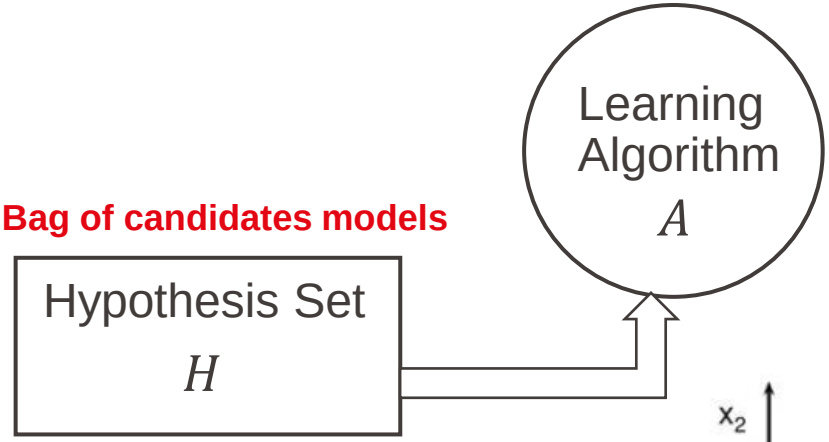
$$h(x) = sign\left(\left(\sum_{i=1}^{d} w_i \cdot x_i\right) + b\right)$$

Linear decision boundary





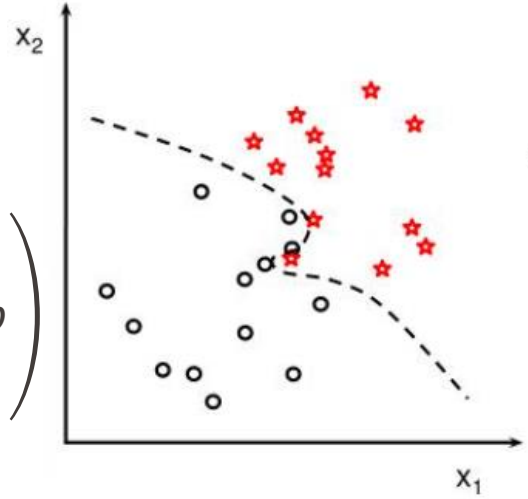
Formulation of the learning problem



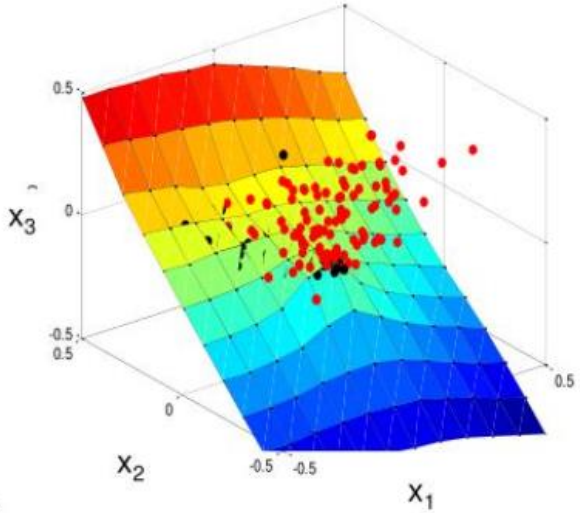
Bag of candidates models

Simple generic linear hypothesis:
Functional form *h(x)*

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i \cdot x_i\right) + b\right)$$



Non-linear decision boundary



Formulation of the learning problem

Functional form *h(x) -> perceptron*

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i \cdot x_i\right) + b\right)$$

Training examples

$\mathcal{D}: (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

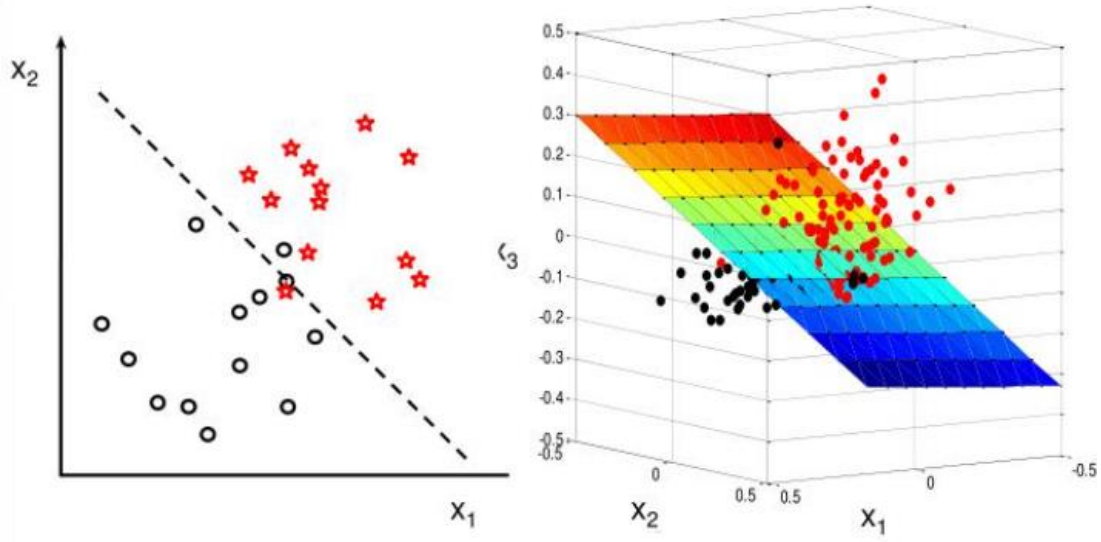
Linear decision boundary

If the problem is linearly separable:

- *Set of* $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$, and b *such that* $h(x) = y$ *is satisfied for all training samples.*

Iterative procedure:

$$w(t+1) = w(t) + y(t) \cdot x(t)$$



…Type of learning: Supervised Learning

Training data

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

Batch Learning:

- *training on a fixed **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*

…Type of learning: Reinforcement Learning

Training data

$(input, output, reward/penalty)$

Reinforcement learning:

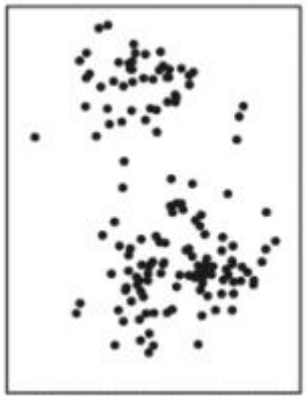
- *an "**agent**" learns to make decisions by continuously interacting with an environment and receiving **feedback** in the form of **rewards** or **penalties**.*
- *The goal of reinforcement learning is to learn a "**policy**", which is a **mapping from states to actions**, that maximizes the cumulative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

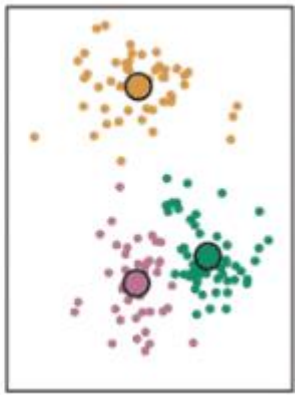
- ***Training data** consists of **sequences of states**, **actions**, and **rewards**.*
- *Learning by **trial-and-error**, where the agent takes actions, receives rewards, and updates its policy based on the observed rewards until convergence to an optimal solution*

…Type of learning: Unsupervised Learning

Training data

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



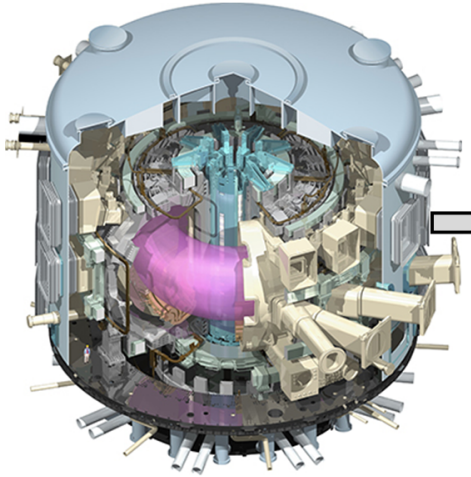


Unsupervised Learning:

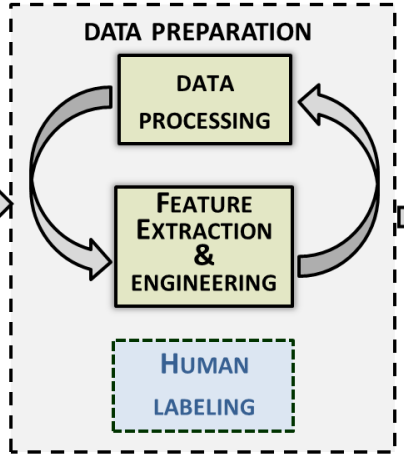
- *useful to discover **patterns** or **structure** in the data, with **no labelled data**. The learning algorithm task is to identify structure in the data, such as **grouping** similar examples according to a well defined **metric**.*
- *Some common unsupervised learning techniques:*
  - ***Clustering:** 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).*

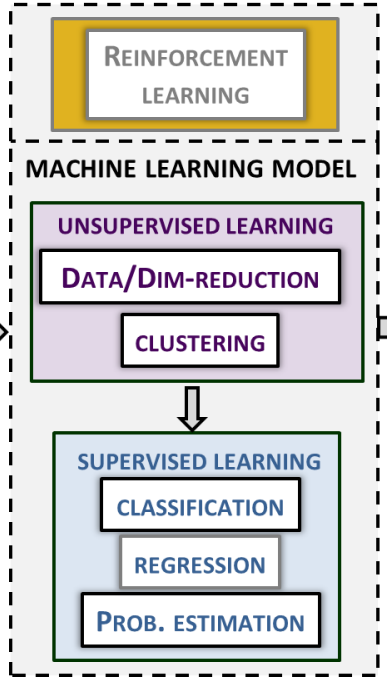
...Typical Machine learning workflow

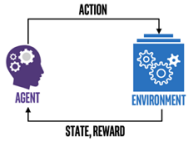
- Massive volume of data
- High-dimensional;



- Heterogenous
- multiple timescales







PLASMA CONTROL & SIMULATORS

DATA VISUALIZATION

PATTERN RECOGNITION

GENERATIVE MODELS

ANOMALY DETECTION

EVENT/OBJECT DETECTION

IMAGE PROCESSING

TIME SERIES FORECASTING

ML foundations: fitting/training a model

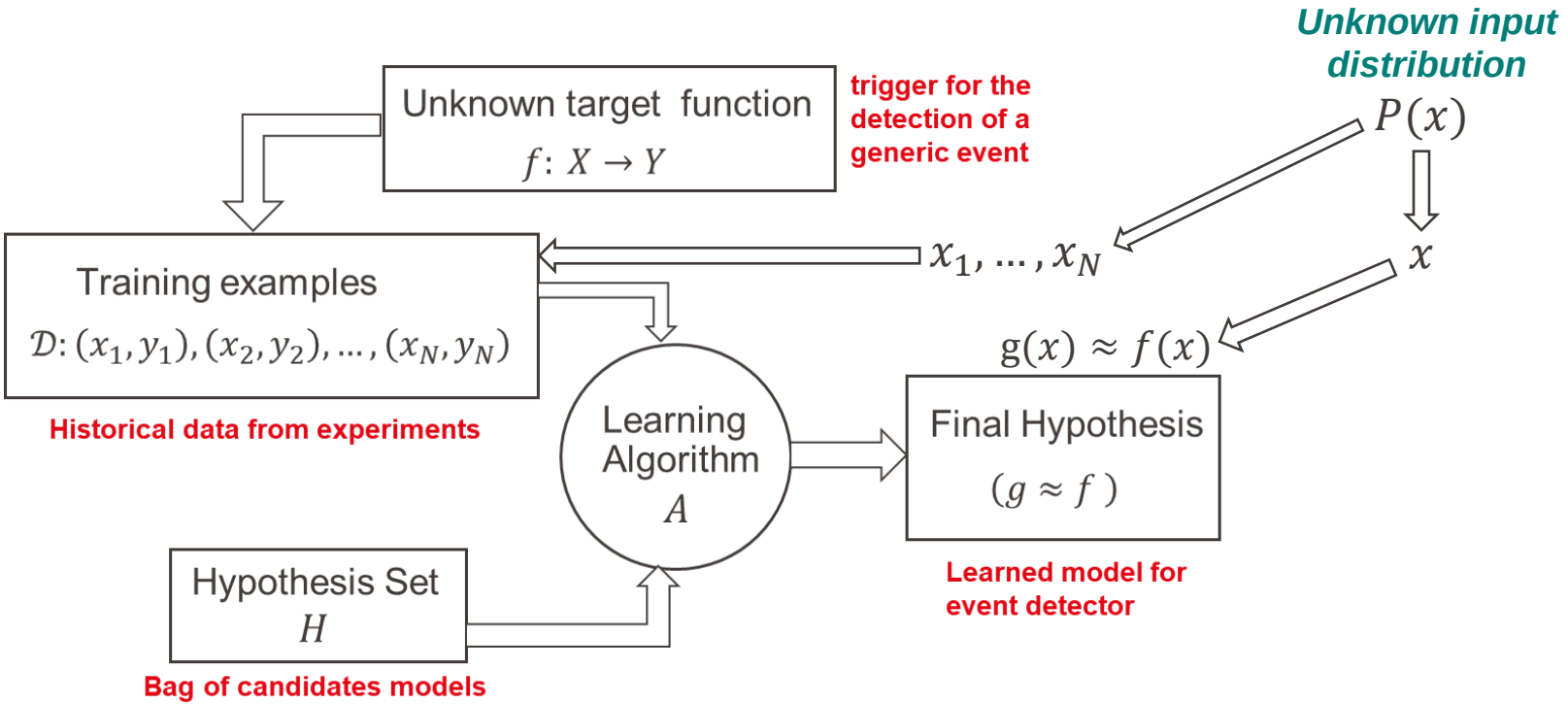
Unknown target function

$$f: X \rightarrow Y$$

Model fitting, or training:

- *Learn the **unknown target function** describing the mapping* $f(x, \boldsymbol{\theta}) \rightarrow y$
- *find the **set of parameters** $\boldsymbol{\theta}$ that best describe the mapping between the input and output variables in the data.*
- *A good fitting enables the model to **generalize** from the training data to new, unseen data.*
- *The **quality of the model fit** depends on a number of things:*
  - *choice of model;*
  - *quality of the input data (and labelling in supervised settings);*
  - *choice of optimization algorithm used to fit the model.*

When learning is feasible...



Hoeffding Inequality *upper bound*

$$\mathrm{P}([E_{in}h(x) - E_{out}h(x)] \geq \epsilon) \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

When learning is feasible

Unknown input distribution

$P(x)$

$\Downarrow$

$x_1, \dots, x_N$

$\Downarrow$

$g(x) \approx f(x)$

Hoeffding Inequality ***upper bound***

$$\mathrm{P}([E_{in}h(x) - E_{out}h(x)] \geq \epsilon) \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

$\Downarrow$

$$\mathcal{H} = \{h_1, \dots, h_M\} \rightarrow g$$

h is fixed before generating the dataset, g is our final hypothesis given $\mathcal{D}$

$\Downarrow$

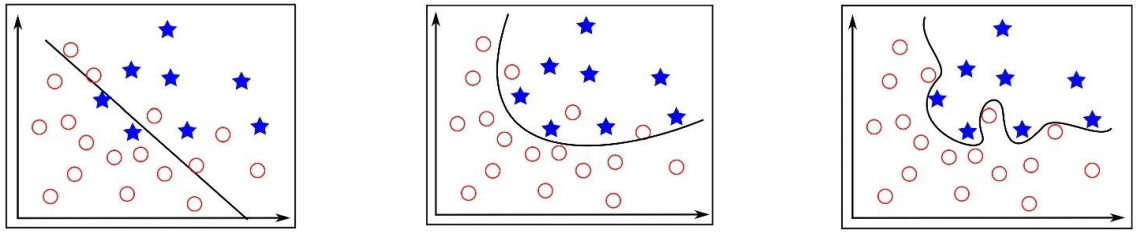
$$\mathrm{P}([E_{in}h(x) - E_{out}h(x)] \geq \epsilon) \leq 2Me^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

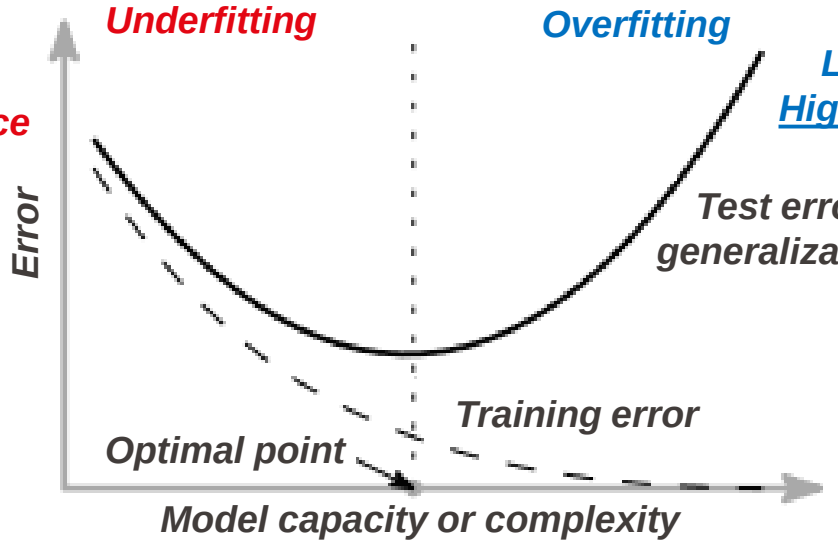
$\Downarrow$

Vapnik-Chervonenkis (VC) dimension: measure of the **complexity** of a learning problem.

- used to derive **bounds** on the **generalization** error of learning algorithms, and to compare the **expressive power** of different families of functions or models.

Underfitting and Overfitting: bias/variance trade-off





$\hat{\theta}$ is a "point estimator" (predicted "target")

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

$E[\hat{\theta}]$ is evaluated over the training sets

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

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

Underfitting and Overfitting: bias/variance of the squared error

$$Bias\left[\hat{f}(x)\right] = E\left[\hat{f}(x)\right] - f(x)$$

$$Var\left[\hat{f}(x)\right] = \left[(E\left[\hat{f}(x)\right] - f(x))^2\right]$$

$y = f(x) - ?$

- f: *true function*
- $?$: *noise error* ε *with mean* 0 *and variance* σ^2
- y: *true target (label)*

Error on unseen training data x

$$Err(x) = E\left[(y - \hat{f}(x))^2\right]$$

$\hat{f}$: *model learned*

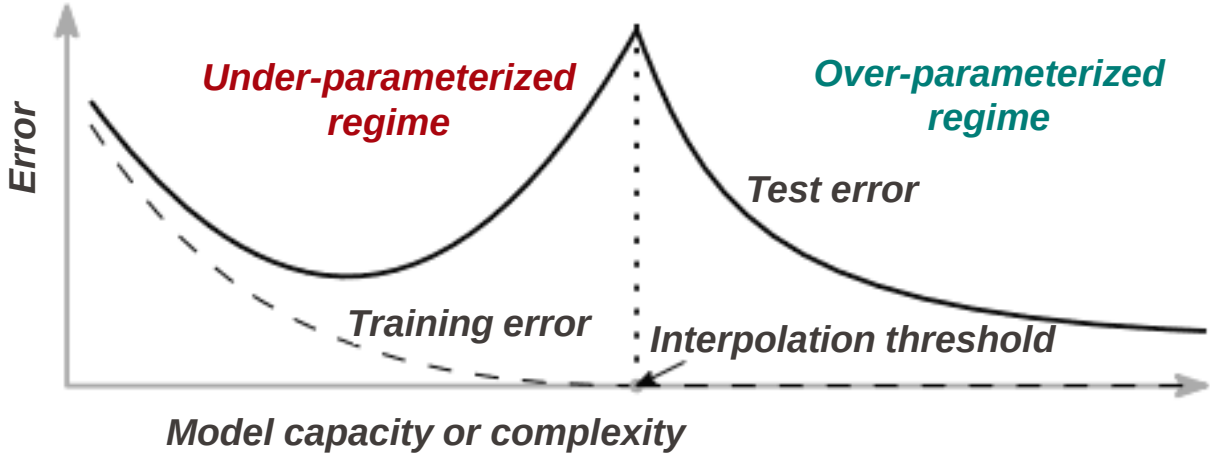
Bias/variance decomposition of the squared error [derivation]

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

- $(Bias\left[\hat{f}(x)\right])^2$: *How far the learned model is from the true function*
- $Var\left[\hat{f}(x)\right]$: *Changes when the model is trained on different data samples*
- σ^2: *Irreducible error*

Underfitting and Overfitting: bias/variance trade-off

- ***Classical machine learning*** *operates in the under-parameterized regime*
- ***Modern machine learning*** *operates in the over-parameterized regime*



Phenomenon known as: ***"double descend"***

- Model-wise double descend
- Epoch-wise double descend

ML foundations: fitting/training a model

Model fitting, or training:

- *Learn the unknown target function describing the relation* $f(x, \boldsymbol{\theta}) \rightarrow y$
- *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 a optimization problem in terms of minimization of a* ***objective*** *or* ***loss function***

Training examples

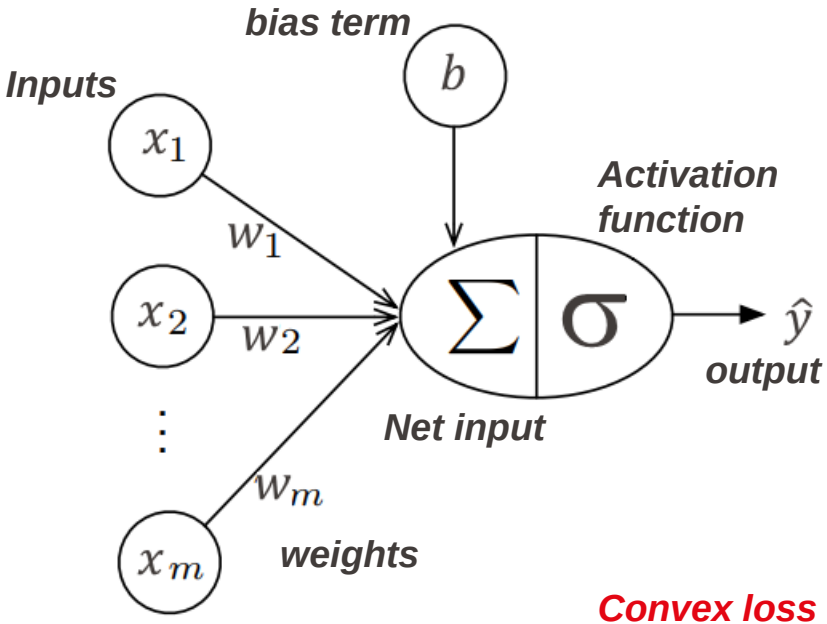
$\mathcal{D}: \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$

Loss function

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

- *We call* ***inference*** *the procedure with which we quantify the uncertainty or confidence in the estimate* $\widehat{\boldsymbol{\theta}}$

ML foundations: linear neurons & Perceptron

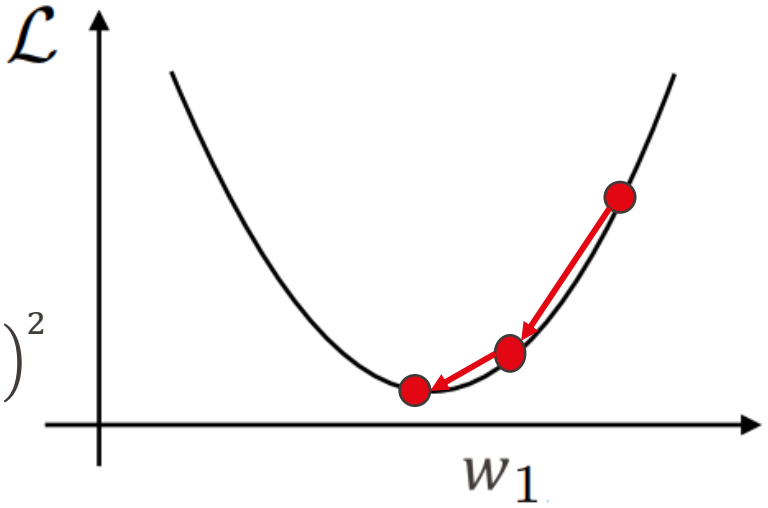


$$\hat{y} = \sigma\left(\left(\sum_{i=1}^{m} w_i \cdot x_i\right) + b\right) = \sigma(\boldsymbol{x}^T \boldsymbol{w} + b)$$

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

Convex loss 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 more noisy (shuffling each epoch)*

Batch mode:

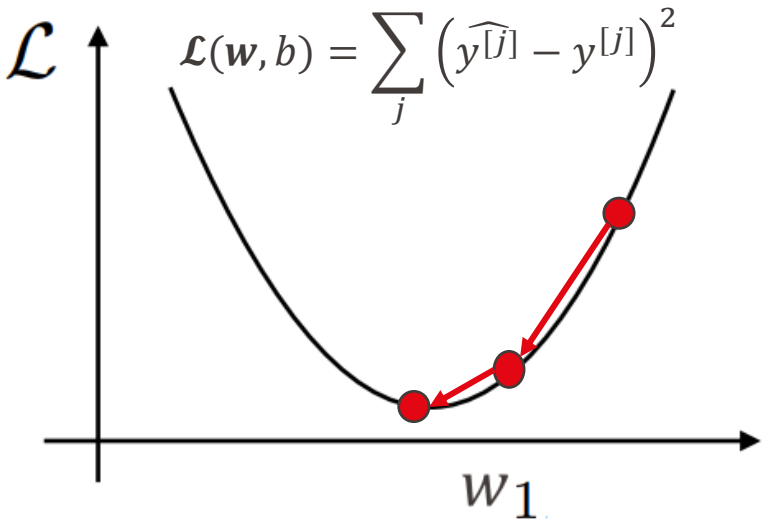
- *Slower but less sensitive to noise*

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".***
- *Vectorization*
- *Less noisy than online-mode & learning faster than batch*

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

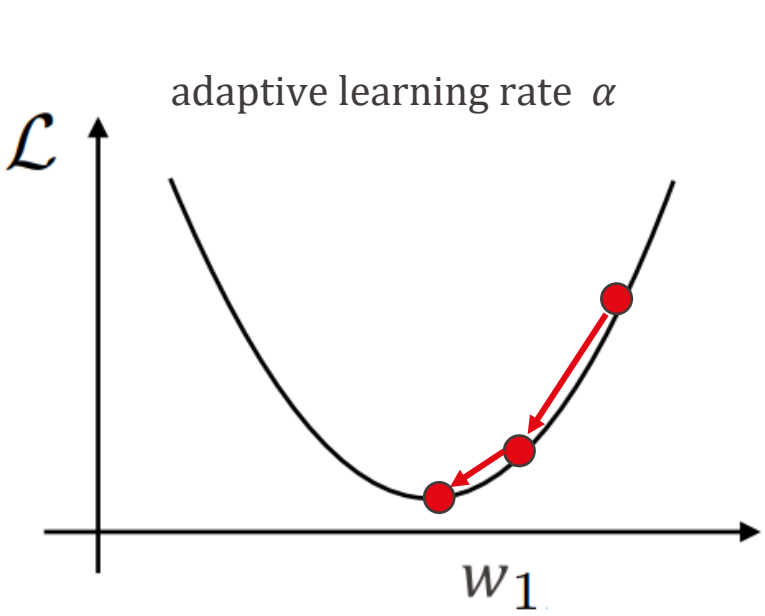
$$\mathcal{D}: \left(\boldsymbol{x}^{[1]}, y^{[1]}\right), \left(\boldsymbol{x}^{[2]}, y^{[2]}\right), \ldots, \left(\boldsymbol{x}^{[N]}, y^{[N]}\right) \in \Re^m$$

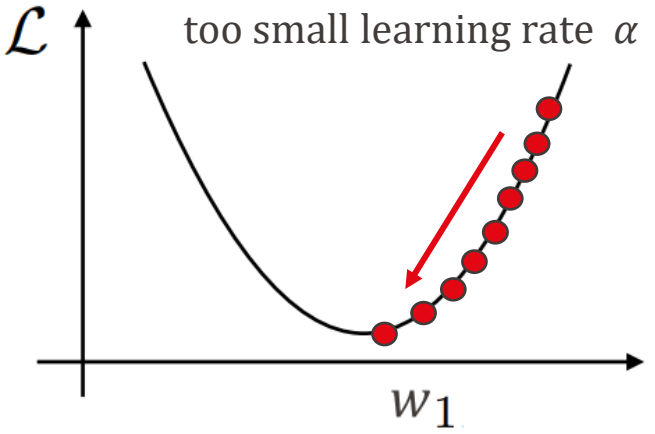


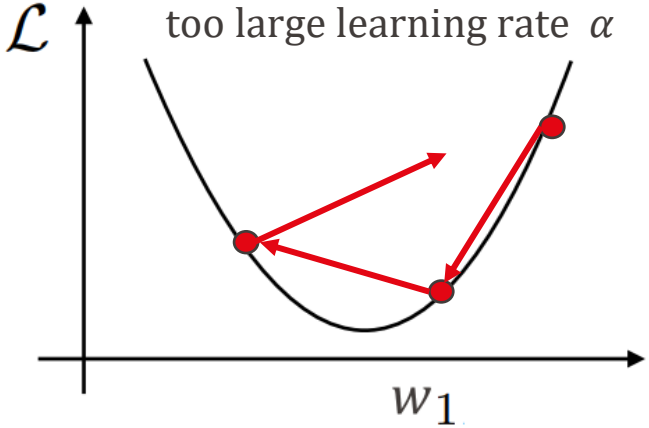
Other training paradigma:

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

ML foundations: fitting/ Gradient Descent (GD)







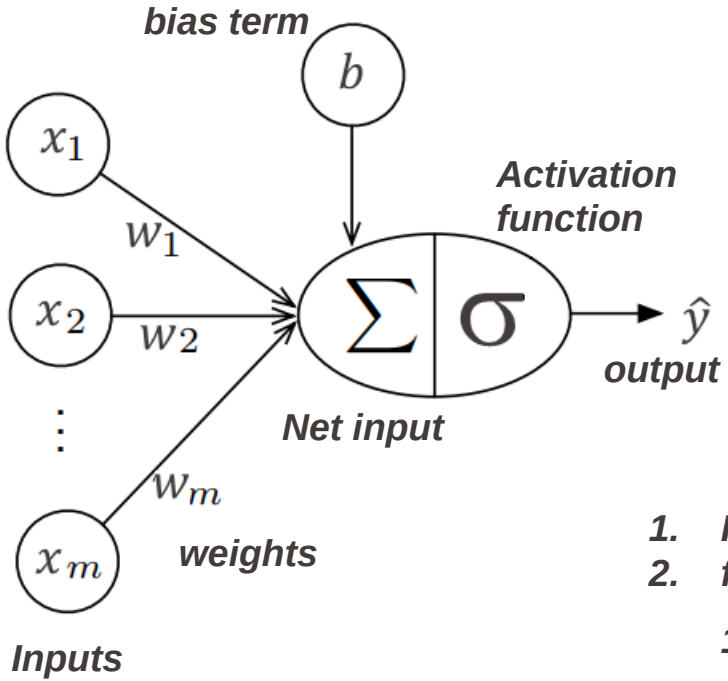
Convex Loss function

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

Machine Learning foundations

Fit/Train of ML models…

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:

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

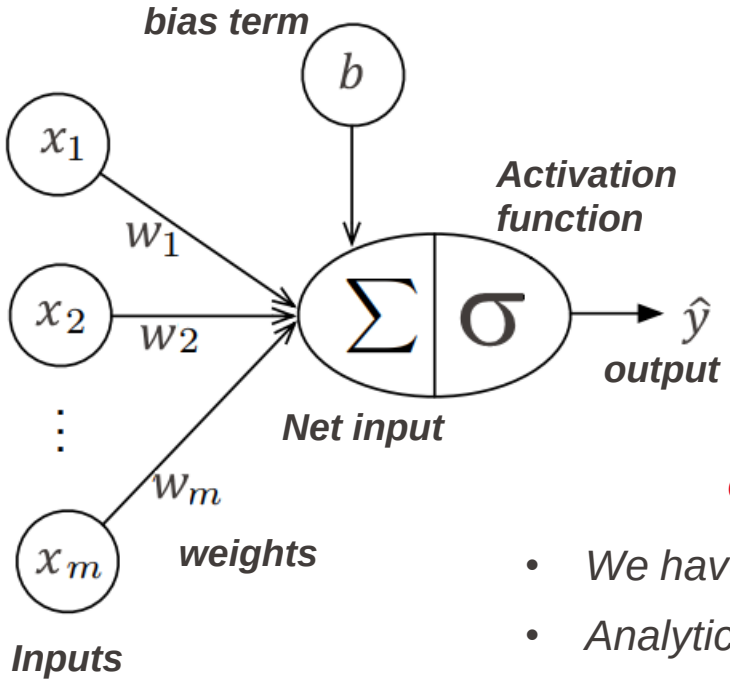
On-line mode with Stochastic Gradient Descent

1. ***Initialize*** $\boldsymbol{w}, \boldsymbol{b}$
2. ***for every training epoch:***
   1. ***for every*** $\left(x^{[j]}, y^{[j]}\right)$ ***in*** $\mathcal{D}$: (or over mini-batches)
      1. $\widehat{y^{[j]}} = \sigma\left(\boldsymbol{w}^T \boldsymbol{x}^{[j]} + b\right)$ ***compute prediction***
      2. $\nabla_{\boldsymbol{w},\boldsymbol{b}} \mathcal{L} = \left(y^{[j]} - \widehat{y^{[j]}}\right) \cdot \boldsymbol{x}^{[j]}$ $(with \;\; x^{[0]} = \mathbf{1}\; for\; \boldsymbol{b})$
      3. $\boldsymbol{w}, \boldsymbol{b} = \boldsymbol{w}, \boldsymbol{b} + \alpha \cdot \left(-\nabla_{\boldsymbol{w},\boldsymbol{b}} \mathcal{L}\right)$ (α is the learning rate)

Convex loss function

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

ML foundations: Linear Regression (Least-squares)



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

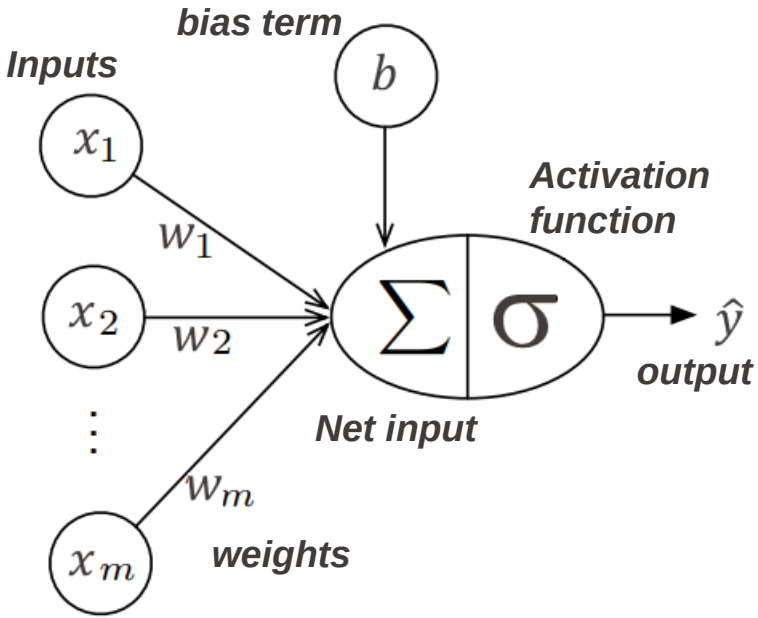
Given a training set:

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

Optimization problems with Least-Squares

- *We have to fit basically a **linear regression model***
- *Analytical solution (**normal equation**):* $\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T y$
- Sometimes **matrix inversion computationally expensive** (large datasets)
- We can **learn** this parameters **iteratively**, fitting neural networks…

ML foundations: Linear Regression (Least-squares)



On-line mode with Stochastic Gradient Descent

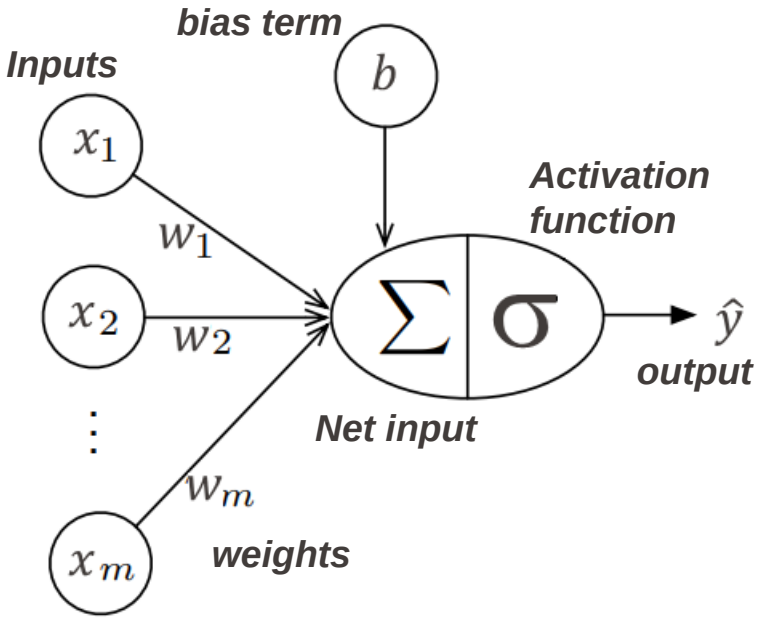
$$1.\quad \frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial\left(\frac{1}{2N}\sum_j\left(\widehat{y^{[j]}} - y^{[j]}\right)^2\right)}{\partial w_i}$$

Convex Loss function

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

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

ML foundations: Linear Regression (Least-squares)



Convex Loss function

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

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

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

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

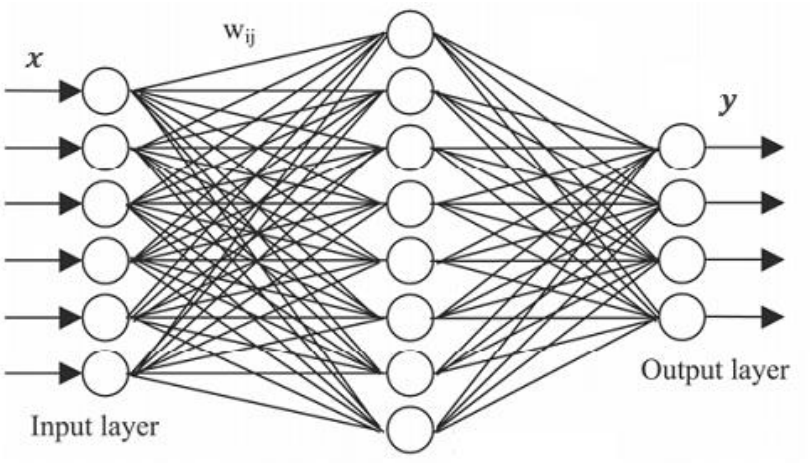
$$\frac{\partial f(\sigma(h(\boldsymbol{w})))}{\partial w_i} = \frac{\partial f}{\partial \sigma}\cdot\frac{\partial \sigma}{\partial h}\cdot\frac{\partial h}{\partial w_i} \qquad where \begin{cases} f = (\sigma - y_i)^2 \\ \sigma = I(h) \\ h = \boldsymbol{w}^T x^{[j]} \end{cases}$$

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

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

ML foundations: Linear Regression (Least-squares)

Back-propagation (Jacobians)



On-line mode with Stochastic Gradient Descent

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

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

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

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

Convex Loss function

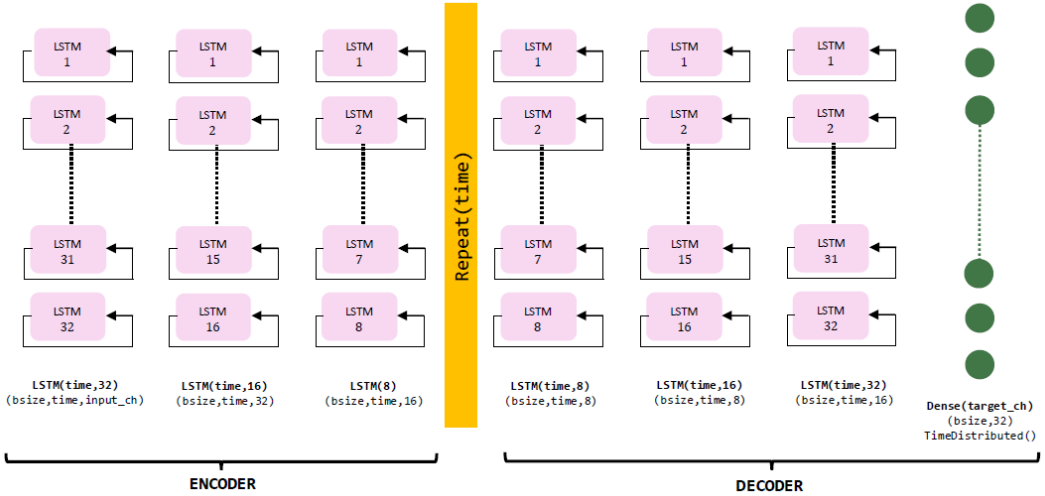
$$\mathcal{L}(\boldsymbol{w}, b) = \frac{1}{2N}\sum_j \left(\widehat{y^{[j]}} - y^{[j]}\right)^2 \textbf{\textit{(MSE)}}$$

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

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

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

Reconstruction of T_e profile from TS

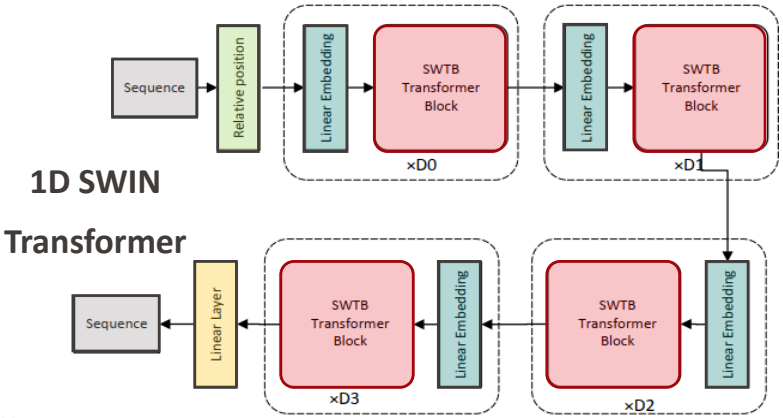


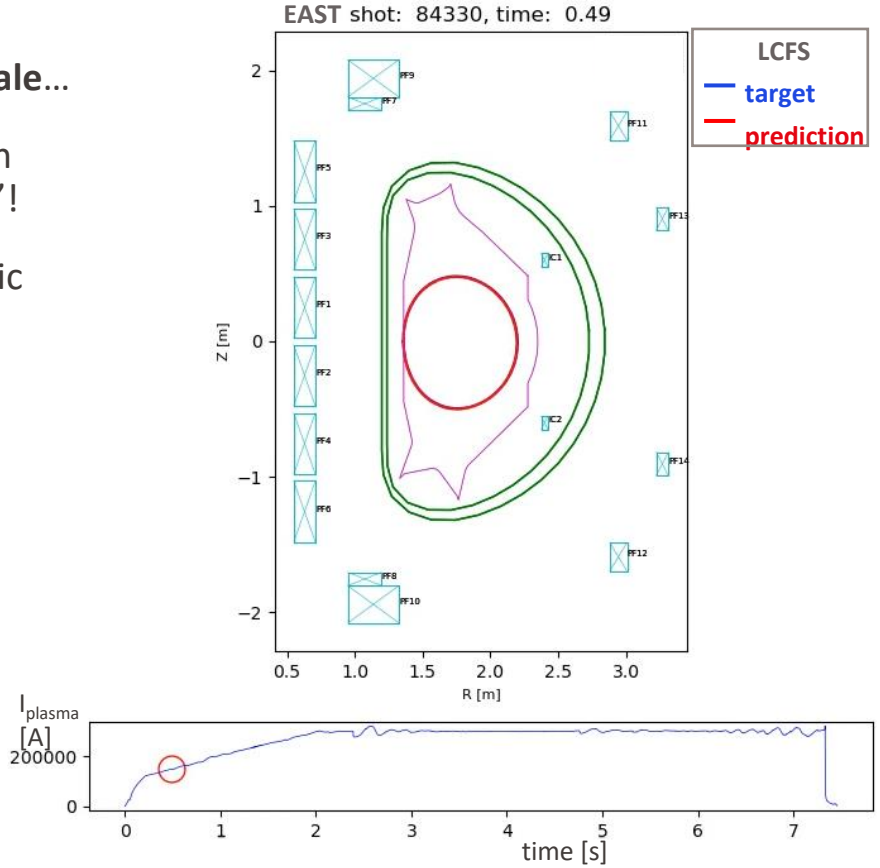
- **Encoder-Decoder**: embedding of high-dimensional space and reconstruction in the original input space
- Learning of **short** & **long-range dependencies** of the temporal dynamics of the profile evolution;

***REF**:[Cristina Venturini: Master Thesis]*

Magnetic equilibrium reconstruction (LCFS)

- **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)





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

Machine Learning foundations

A probabilistic perspective

ML foundations: fitting/training a model

- *We call **inference**(*) the procedure with which we quantify of the uncertainty or confidence in the estimate* $\widehat{\boldsymbol{\theta}}$. $\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\mathcal{D}|\boldsymbol{\theta})$
- *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*

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

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

ML foundations: fitting/training a model

- *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)**:*

ML foundations: fitting/training a model

- ***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.*

Bayesian inference

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

Given a dataset:

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

Posterior **Likelihood** **Prior**

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

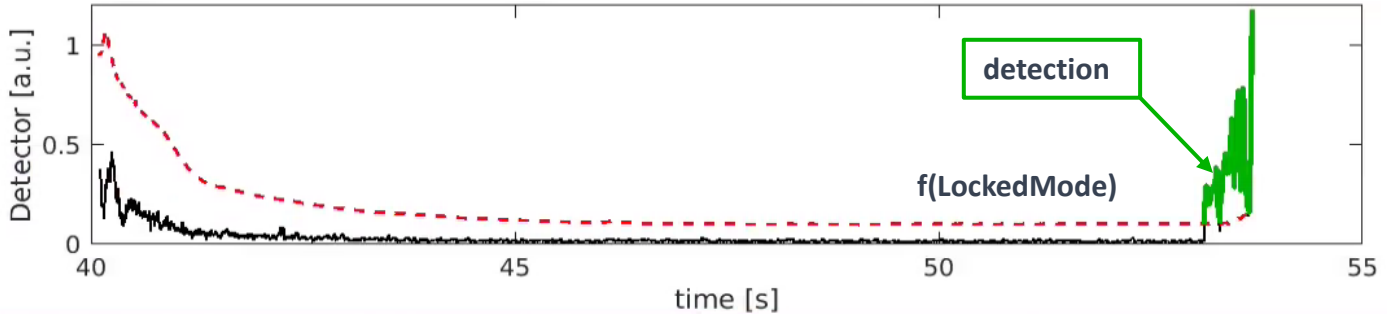
$\boldsymbol{\theta}$ is an unknown random variable

Evidence or Marginal

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

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

Bayesian inference: event-detection problem

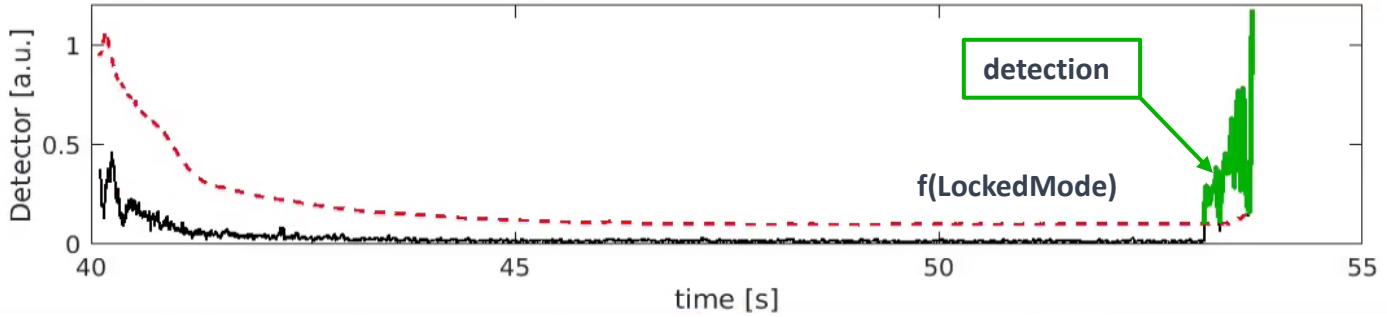


- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges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

Bayesian inference: event-detection problem

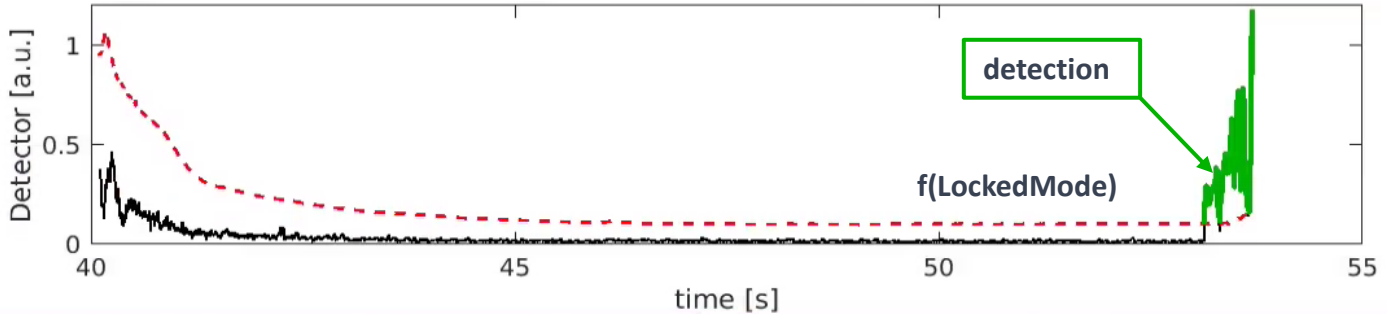


- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges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

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

0.99 (over $P(detect|LM)$), 0.02 (over $P(LM)$), ?% (over $P(detect)$)

Bayesian inference: event-detection problem



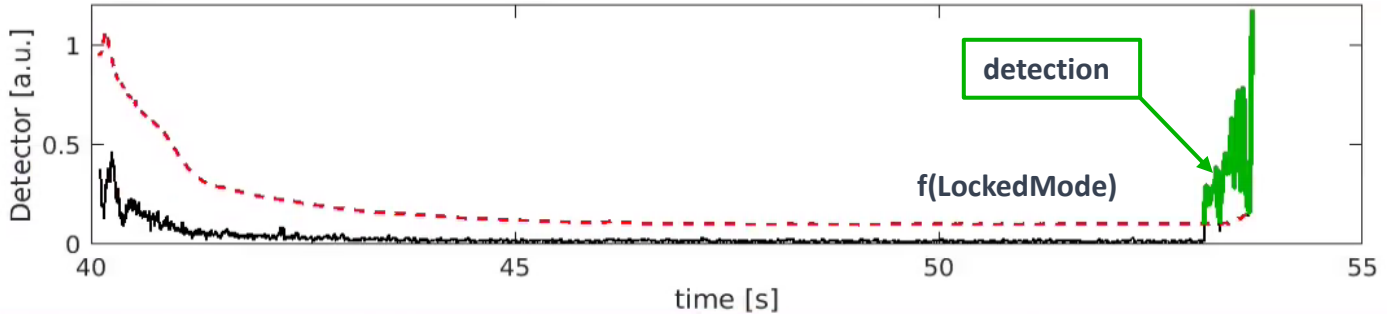
- We have a RT-detector for **Locked Modes** (LM - common disruption precursor:
- The detector works very well:
  - It has an **accuracy** of **99%** (correct detection when there actually is a LM)
  - It has a very low **false positive** rate **0.1%**
  - In our sampling distribution **2%** of the discharges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

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

0.99 (over $P(detect|LM)$), 0.02 (over $P(LM)$), ?% (over $P(detect)$)

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

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%**

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

(0.99 over $P(detect|LM)$, 0.02 over $P(LM)$, ?% under $P(detect)$)

$$P(detect) = P(detect\,|\,LM) \cdot P(LM) + P(detect\,|\,\sim LM) \cdot P(\sim LM)$$

(0.99, 0.02, 0.001, 1-0.02)

$$P(LM|detect) = \frac{P(detect|LM) \cdot P(LM)}{P(detect)} = \sim \mathbf{0.953}$$

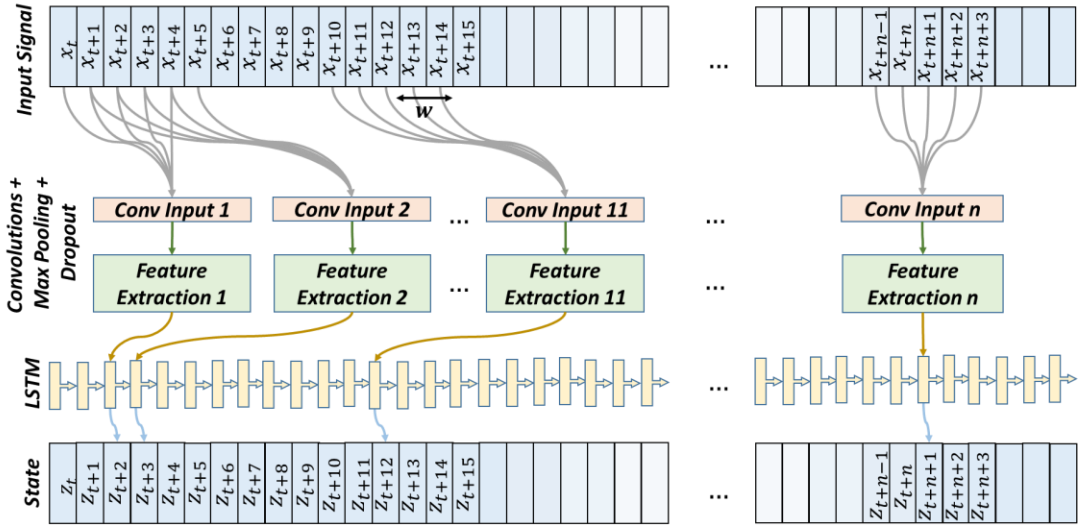
(0.99, 0.02, 0.0208)

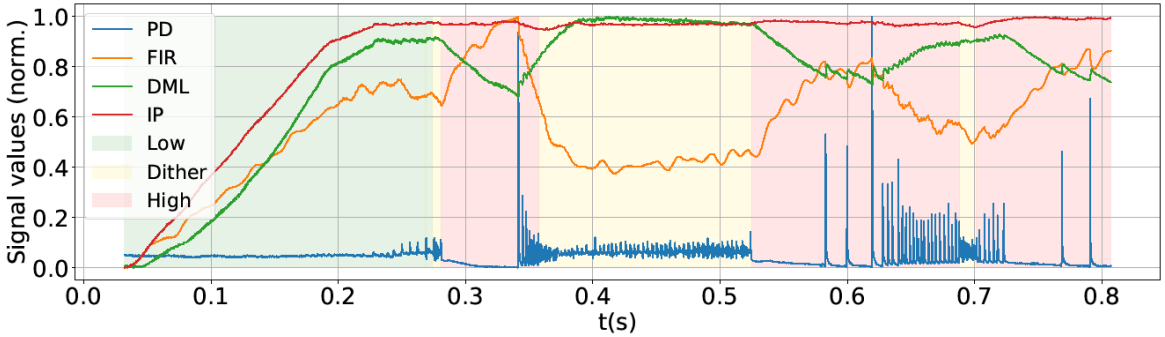
Machine Learning foundations

Event Detection & Disruption prevention…

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**

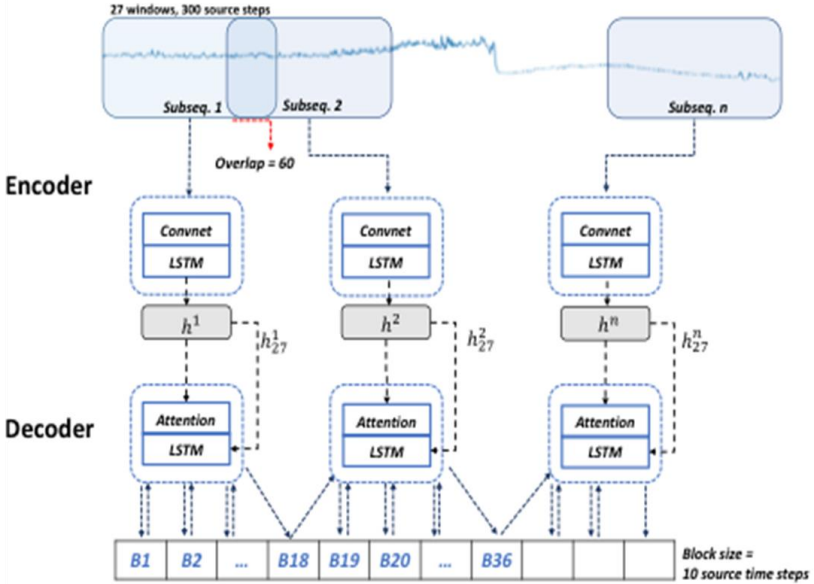




REF: [Matos et al NF 2020]

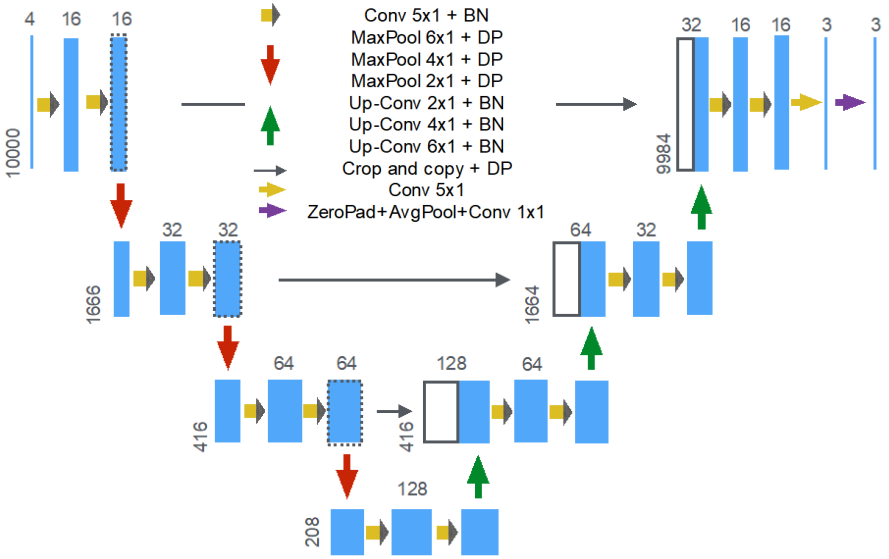
Event detection and plasma confinement state classification

- **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]

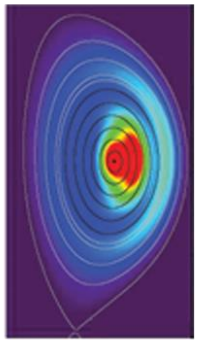
- **UTIME MODEL:**
  - **Multi-scale convolutional** structure allows to capture **patterns at different scales** present in the plasma.
  - processing the whole signal at once (offline) with the ability to see at a **wider contextual information.**

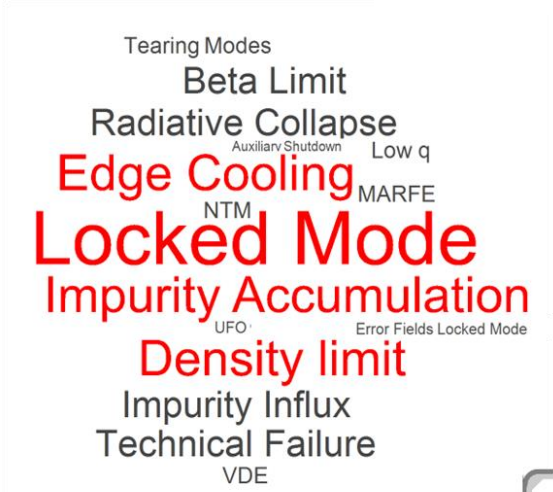


REF: [Marceca et al NEURIPS 2021]

Disruptions: a complex physics picture

Impurity Accumulation (radiation)

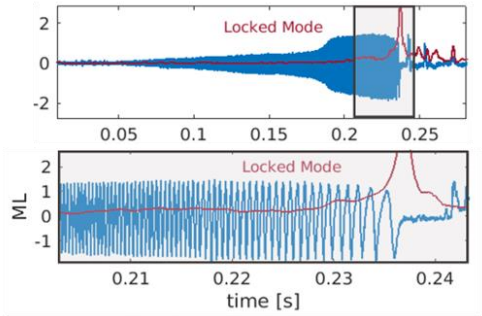




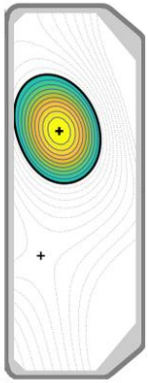
*REF: [P. de Vries et al. **NF** 2011]*

- Wide plethora of **physics** and **technical** cause (systems failure, etc.)
- … **loss of control** and potential **damages** to the machine.

Locked modes (MHD)



Vertical Displacement Events

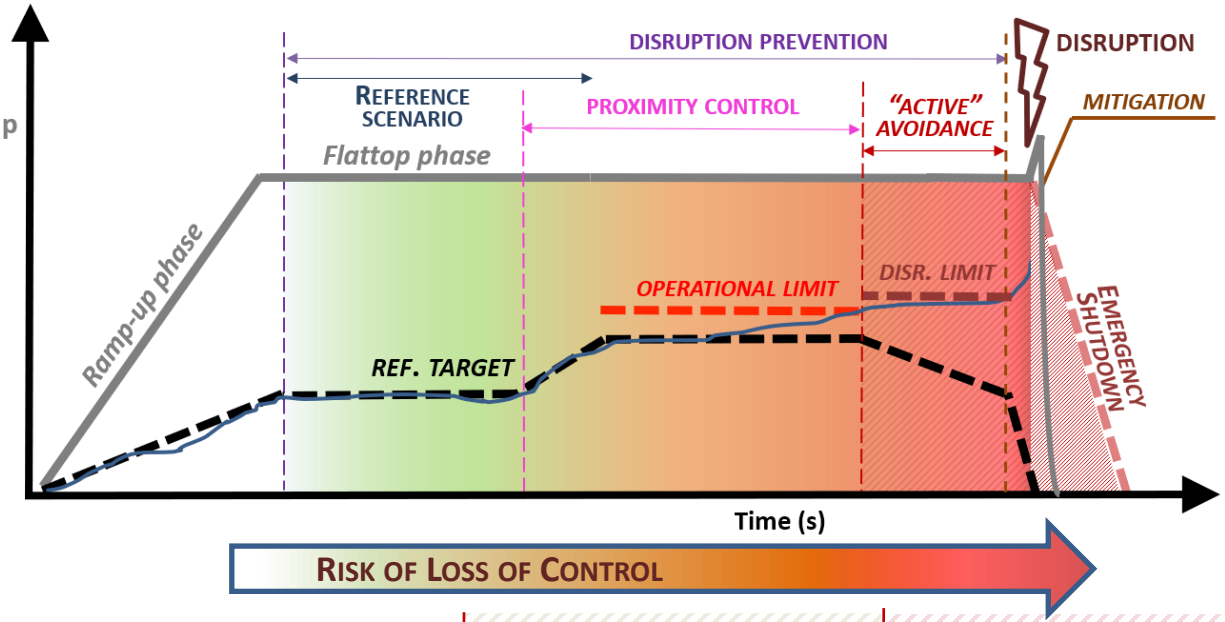


Disruption prevention in a nutshell

- **PLASMA STATE MONITORING & DISRUPTION PREVENTION**

a key requirement for plasma control throughout the entire evolution of the discharge

*REF: [A. Pau et al **EPS 2022**]*



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

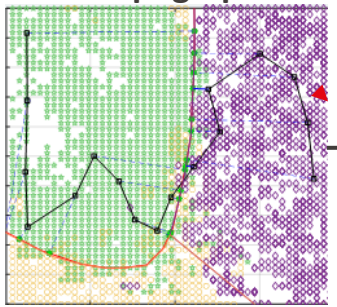
- asynchronous response when crossing operational boundaries (danger levels)

EMERGENCY SHUTDOWN

- Fast controlled shutdown
- mitigation

Feature Engineering & Generative models

Generative Topographic Mapping



*REF: [Bishop et al **IEEE** 1998]*

Posterior probability (Bayes!)over 2D latent space

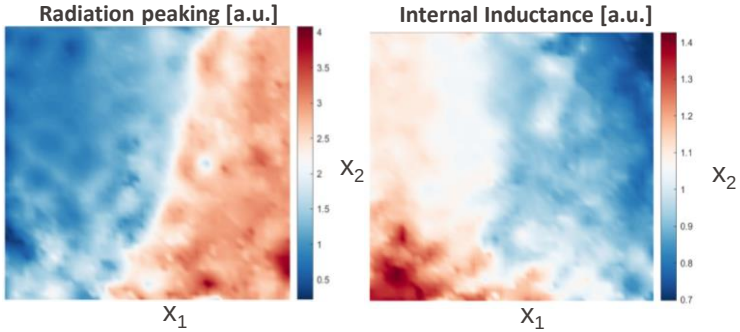
*REF: [Pau et al **IEEE-TPS** 2018*
*REF: [Pau et al **NF** 2019]*

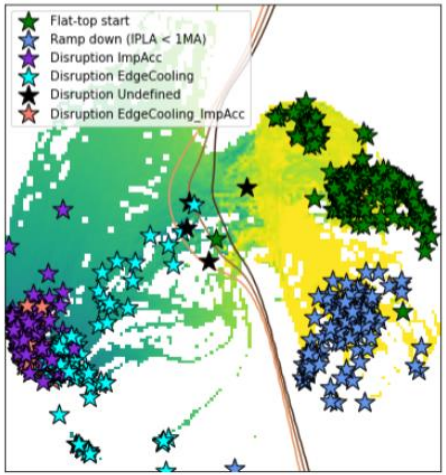
Off-normal plasma regimes

(towards model transparency and interpretability)

- **Trajectories** over a low-dimensional **latent space** encode different disruption paths
- **Clustering** of plasma regions with different **risk of disruptions** to unveil **boundaries**

2D Manifold key physics quantities distributions





REF:[Andrin Bürli: Master Thesis]

References

- **Learning From Data** by Yaser Abu-Mostafa
- **Pattern Recognition and Machine Learning** by Christopher M Bishop
- **Bayesian Reasoning and Machine Learning** by David Barber
- **Machine Learning: A Probabilistic Perspective** by Kevin P. Murphy
- **The Elements of Statistical Learning** by T. Hastie, R. Tibshirani, J. Friedman