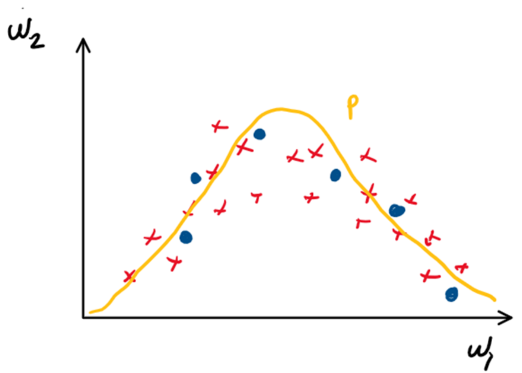
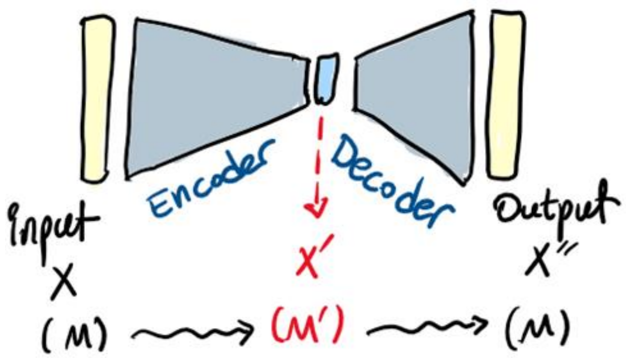
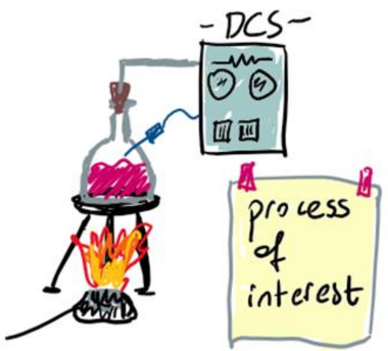
Data Driven Engineering I: Machine Learning for Dynamical Systems

Introduction to Generative Learning: Autoencoders

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer







Previously on DDE-1 :

* Supervised learning → Classification
  → Regression

* Unsupervised learning → Clustering
  → Dim. Reduction

+ Deep Learning + Time Series Analysis

Previously on DDE-1 :

* Supervised learning

⇨ [X], y } Learn a function to map [X] → y } "Discriminative models"

* Unsupervised learning

⇨ [X] ⇄ [X] learn some underlying hidden structures

⇨ Feature Extraction ?

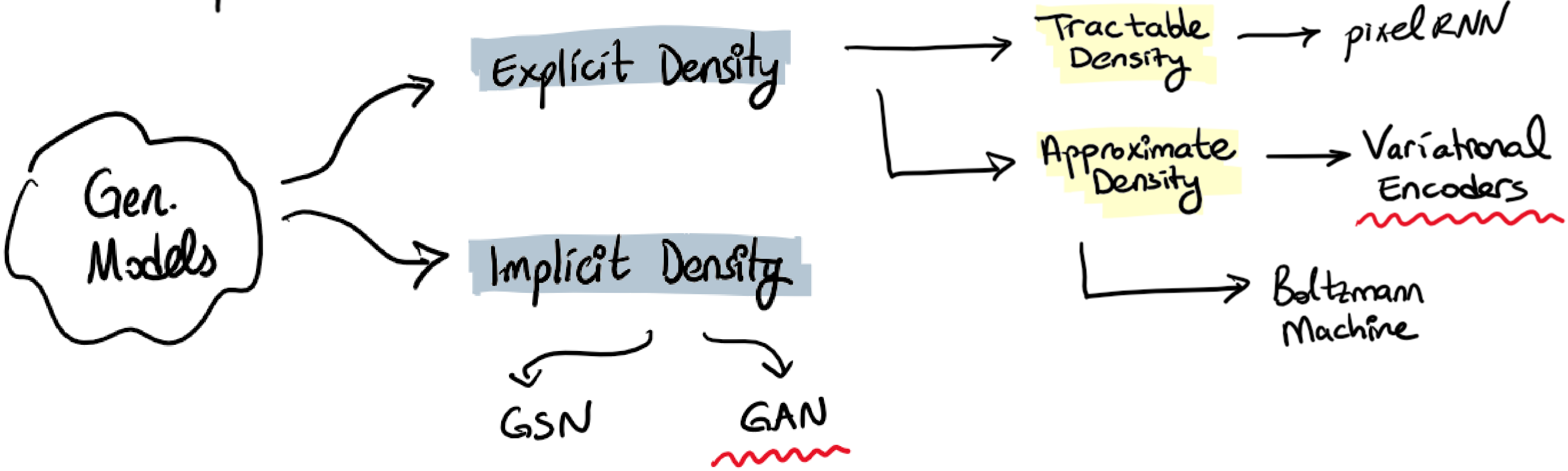
⇨ Promising ⇐ [X] "available"

UL → Generative Models :

* model tells how [x] is formed at the beginning
* It is probabilistic in nature
    * StyleGAN
    * GPT language model
    * Virtual training (RL)

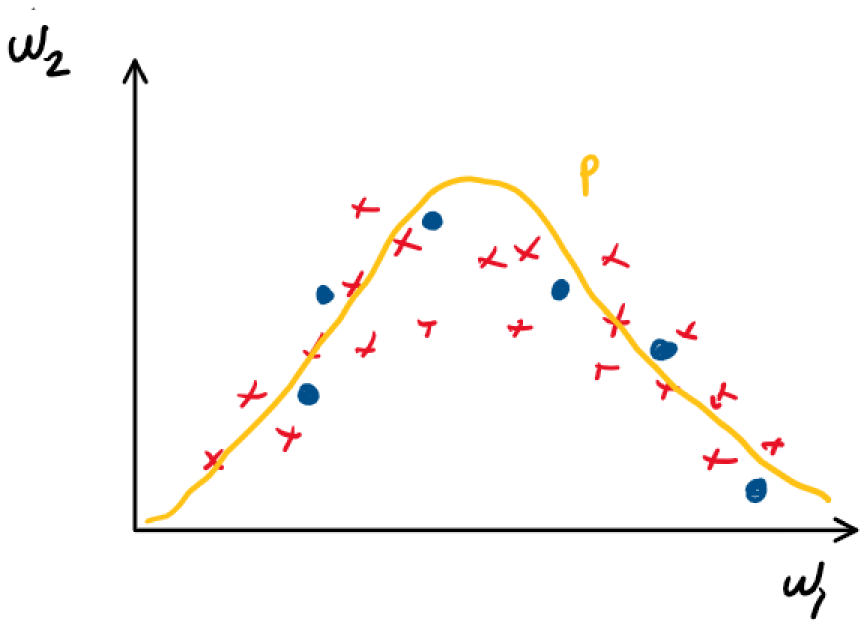
UL → Generative Models :

* It is probabilistic in nature



How does it work?

- we have a dataset X.
- we assume that it is generated according to a rule p
- Model mimics p to create new points.
- Model should not reproduce what it has seen.



How does it work?

① Sample space: Values an observation can take $x = [\cdot \cdot \cdot \cdot]$

② Probability Density Function: function maps x in sample space; $PDF = [0, 1]$

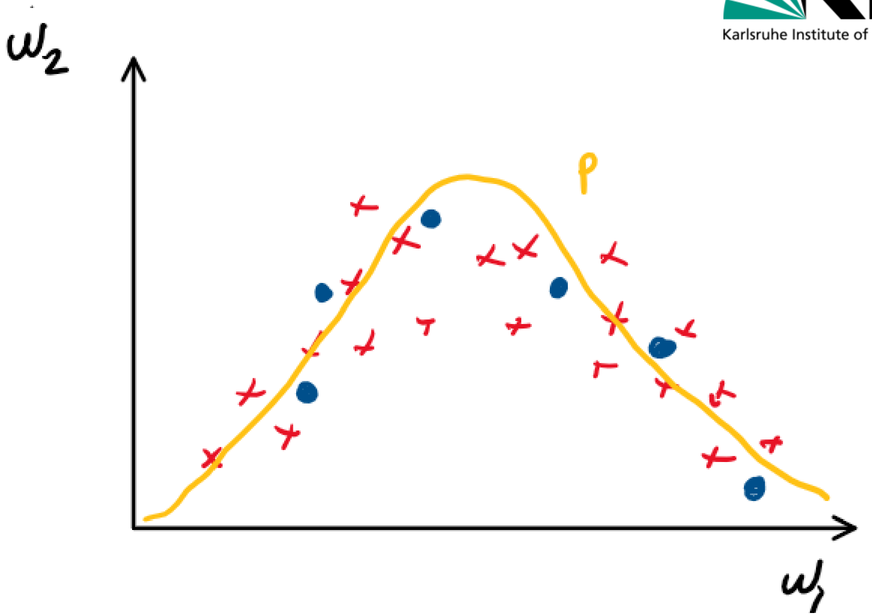
it is well-defined $\Rightarrow$ $\int PDF = 1.0$

③ Parametric Modeling: $\underbrace{p_\theta(\theta_1, \theta_2, \theta_3)}_{\text{find } [\theta_i]} \longrightarrow$ true p

$\Rightarrow$ "maximum likelihood estimation"

How does it work?

- we have a dataset X.
- we assume that it is generated according to a rule p
- Model mimics p to create new points.
- Model should not reproduce what it has seen.



- how can we infer the rules (p_θ) from data?

↳ Representation Learning

Representation Learning

$[X] \Longrightarrow [X']$ } latent space $\Rightarrow$ learn $p_\theta \Rightarrow$ learn p'_θ

high dim. (M) space

low dim. space (M')

Eg

2 47 16 14 21

72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57

5 # ✓

16 # ✗

(72, 57, 1) → ✗

→ ✓

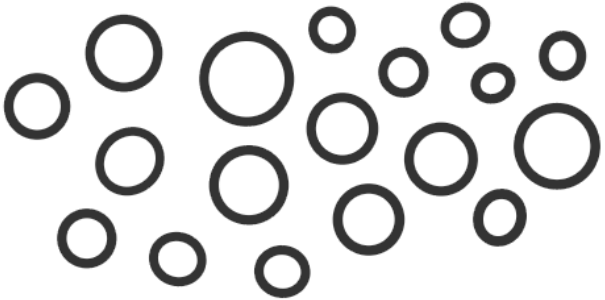
Representation Learning

$[X] \Longrightarrow [X']$ } latent space ⇨ learn p_θ ⇒ learn p'_θ

high dim. space (M) — low dim. space (M')

Perception in chess, 1973

Eg



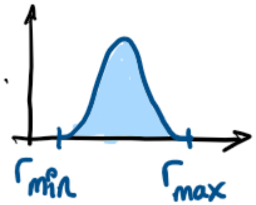
$x_0 =$ 'circle'

$x_1 = (r_0, r_1, r_2, \ldots r_N)$

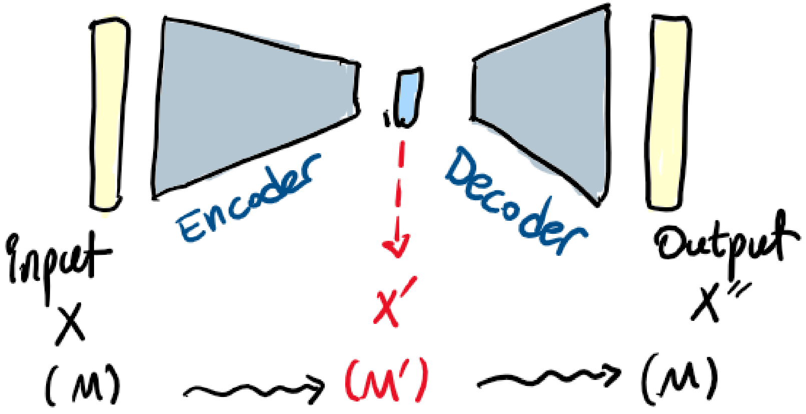
Task: Draw these objects?

Gen. Model

$[X] \rightarrow \boxed{} \rightarrow$ Circle, $p_\theta = \phi_r$



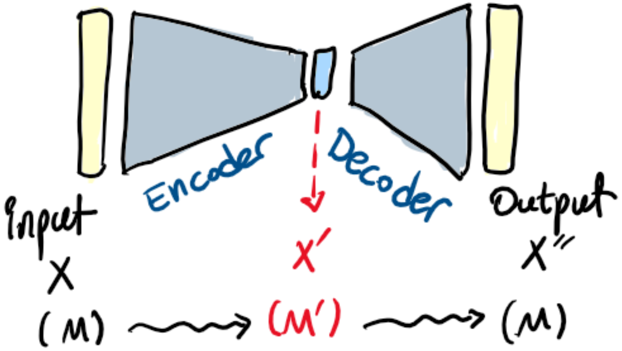
Autoencoders :

* unsupervised approch
* Obj : lower dim. representaton (higher ?)
* 

- Reconstructed X″
- By-product ⇒ X′
- Cost function : $\|X'' - X\|^2$

Autoencoders :

*



* Use cases :

- [x] Dimensionality Reduction
- [x] Anomaly detection

* Use cases :

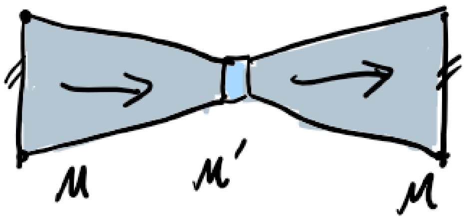
- [x] Forecasting ~ generative
- [x] Sparse Representation
- [x] Denoising dataset
- [x] "Interpolator"
- [x] Unsupervised pre-training
- [x] "Generative Model" (+ pdf)

Base model: Under-complete linear AE

- MLP → linear activation function
  - ↳ cost := MSE

⇓

≈ PCA

- 

  $M > M'$

  Symmetrical layer(s)

- Many layers ⇒ Deep Autoencoder
- Good for large [X] with high M.
- can be used for outlier detection

Today's Agenda

Basic Steps to Follow:

0.) Understand the business/task.
1.) Understand the data.
2.) Explore & prepare the data.
3.) Shortlist candidate models.
4.) ~~Training the model~~
5.) Evaluate the model predictions
6.) "Serve" the model!

Still valid

#0 Understanding the task

- **Problem**: Manufacturing error in a production line

- **Modified sensory input:** 28 variables including sensory input

- 280,000 instances, where only a **small fraction** (~500) of products are **defective**.

- **Heuristic**: <0.5% is defective

A similar example for you:

"Bosch Production Line Performance Reduce manufacturing failures"

#1 Understanding the data

- ❑ Check the data source: understand what the data refers to
- ❑ Objective: understand the characteristics of the data
- ❑ Look at the feature columns:
  - ❑ Any missing values?
  - ❑ Any features with NaN values?
  - ❑ Uniqueness of the dataset? (“cardinality”)

=> Colab

```
 23   S23      284807 non-null  float64
 24   S24      284807 non-null  float64
 25   S25      284807 non-null  float64
 26   S26      284807 non-null  float64
 27   S27      284807 non-null  float64
 28   S28      284807 non-null  float64
 29   Class    284807 non-null  object
dtypes: float64(29), object(1)
memory usage: 65.2+ MB
time: 54.5 ms
```

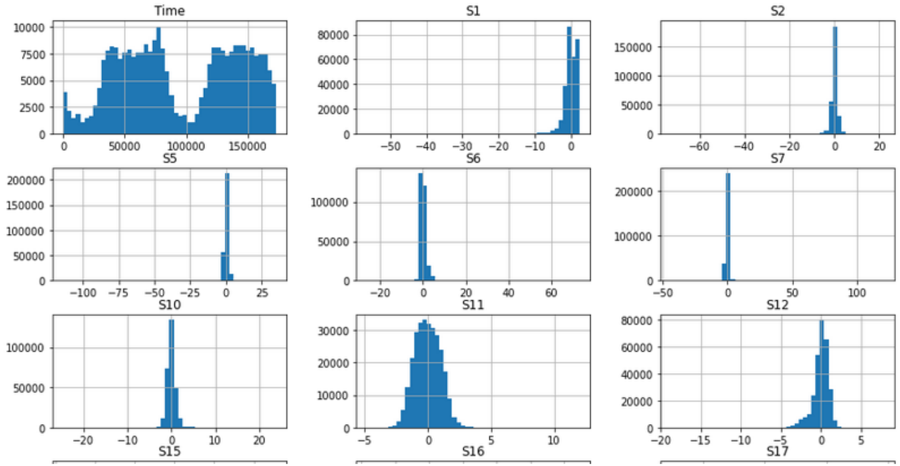
	Time	S1	S2	S3	S4	S5	S6
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.758743e-12	-8.252298e-13	-9.636929e-13	8.316157e-13	1.591952e-13	4.247354e-13
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01

```
time: 447 ms
```

#2 Exploring the data

- ❑ **Objective**: generate a data quality report

- ❑ Using standard statistical measures of central tendency and variation
  - ❑ tabular data and visual plots
  - ❑ mean, mode, and median
  - ❑ standard deviation and percentiles
  - ❑ bars, histograms, box and violin plots



- ✓ Missing values,
- ✓ Irregular cardinality problems,
  - ▪ 1 or comparably small
- ✓ Outliers
  - ▪ invalid outliers and valid outliers

#2 Exporing the data: Correlation Matrix

- Shows the correlation between each pair of features

$$\mathrm{Cov}(\underset{\text{Features}}{a,b}) = \frac{1}{n-1}\sum_{i=1}^{n}\left[(a_i - \bar{a}) \times (b_i - \bar{b})\right]$$

Features ↓ i → instance; $\bar{a}$ → mean; $\bar{b}$ → mean

- Normalized form of "covariance"

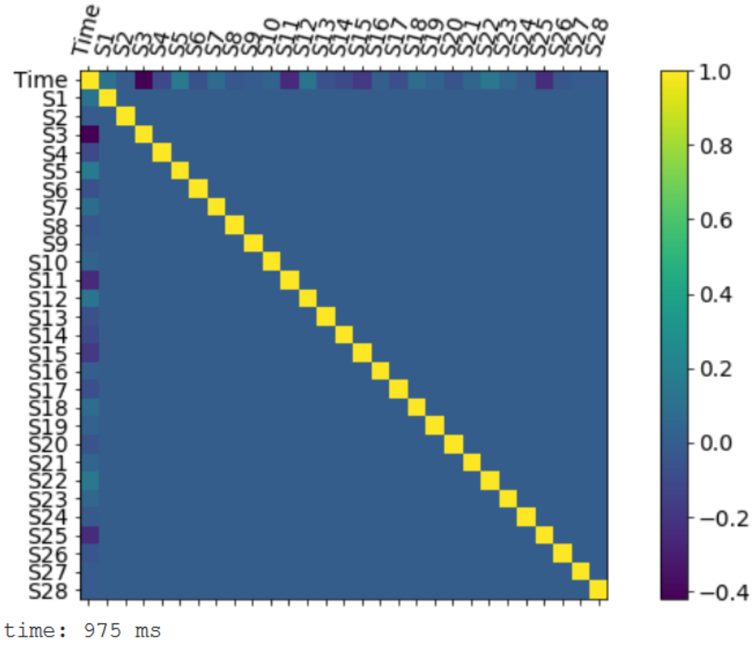
$$\Downarrow$$

$$\mathrm{Corr}(a,b) = \frac{\mathrm{Cov}(a,b)}{SD(a) \times SD(b)}$$

* Normalized
* Dimensionless

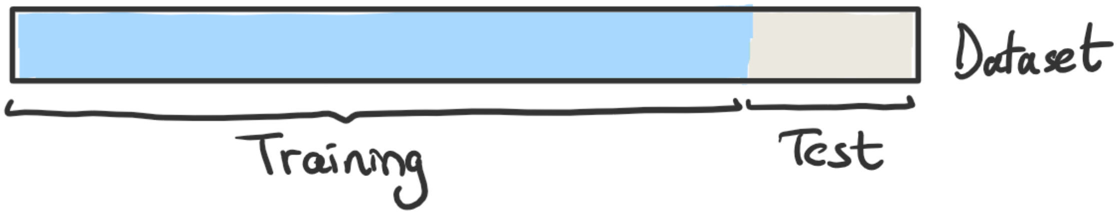
Easy to interpret

- Ranges between −1 and +1



#2 Preparing the Data

❑ Clustering >> unsupervised >> **training & test split not needed**



❑ We will use it to **reduce the volume of the data** when needed:

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(dataX,
     dataY, test_size=0.9,
     random_state=2020, stratify=dataY)

     time: 188 ms
```

#5 Evaluating the Results: Reconstruction error

Encoding Decoding

$$X \rightarrow \boxed{D.R.} \rightarrow \tilde{X} \rightarrow \boxed{DR^T} \rightarrow X'$$

physical coordinates "latent space" physical coordinates

if DR is capable of learning the patterns in the physical system;

$$X \approx X'$$

* $loss = \sum_{m=1}^{M} (x_m - x'_m)^2 \Rightarrow N$ elements

Normalization:

* $$loss' = \frac{loss - \min(loss)}{\max(loss) - \min(loss)} \Rightarrow [0, 1]$$

Interpretation:

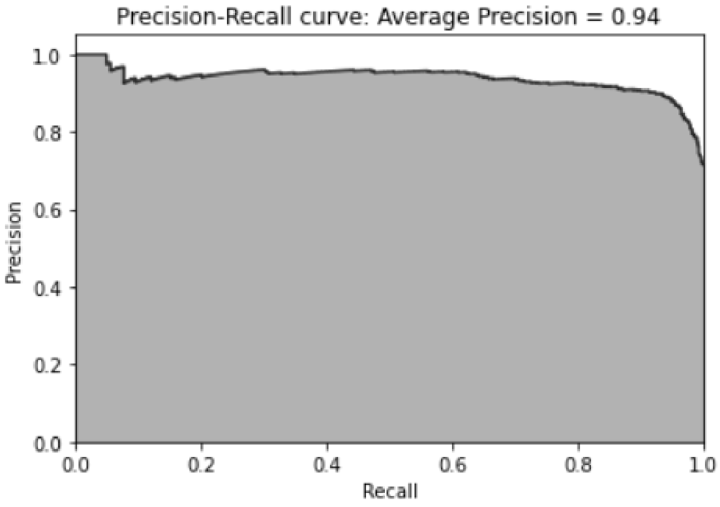
* $loss' \rightarrow 0 \Rightarrow$ Regular Product
$loss' \rightarrow 1 \Rightarrow$ Anomaly, defective

#5 Evaluation of the predictions

Precision Recall Curve (for imbalanced data)

$$\text{Precision} := \frac{\text{True Positive}}{\text{TP} + \text{False Positive}} \Rightarrow \frac{\text{It is positive}}{\text{“It is positive”}}$$

$$\text{Recall} := \frac{\text{True Positive}}{\text{TP} + \text{False Negative}} \Rightarrow \frac{\#\,\text{Correct Predict.}}{\#\,\text{True Cases}}$$

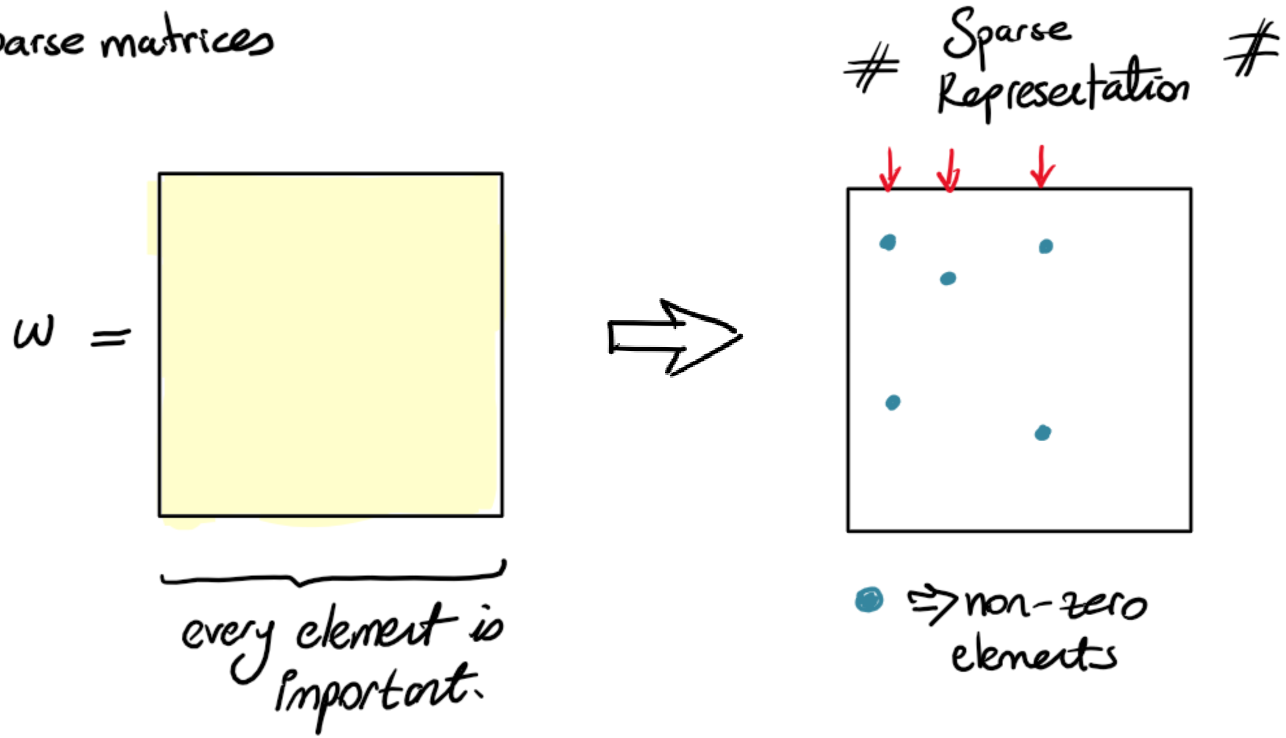


- **Precision** captures how often, when a model makes a positive prediction, this prediction turns out to be correct.
- **Recall** tells us how confident we can be that all the instances with the positive target level have been found by the model.

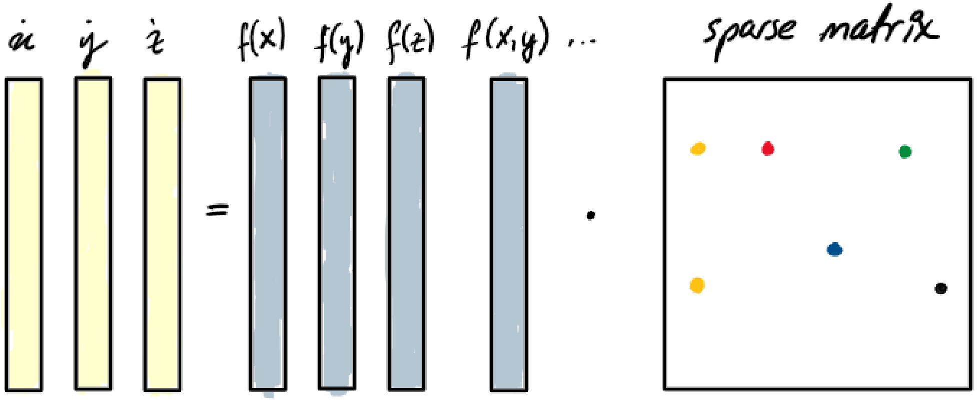
colab

Sparse Autoencoders:

* Sparse matrices



Data Driven Model Discovery

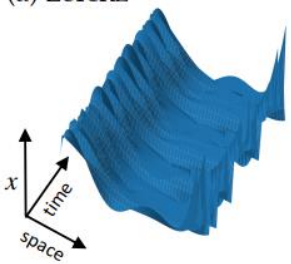


$$\dot{x} = ax - bxy^2 + \phi_i$$
$$\dot{y} = cyz + \phi_i$$
$$\dot{z} = axy + \sin\phi_i$$

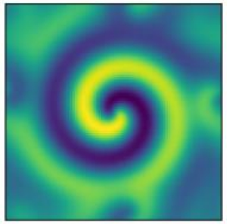
Reg. Linear Regression

$$\dot{z}_1 = -10.0z_1 + 10.0z_2$$
$$\dot{z}_2 = 27.7z_1 - 0.9z_2 - 5.5z_1z_3$$
$$\dot{z}_3 = -2.7z_3 + 5.5z_1z_2$$

(a) Lorenz

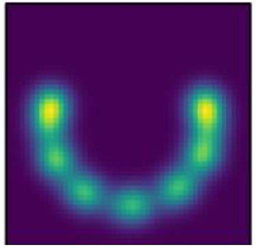


(b) Reaction-diffusion



$$\dot{z}_1 = -0.85z_2$$
$$\dot{z}_2 = 0.97z_1$$

(c) Nonlinear pendulum



$$\ddot{z} = -0.99\sin z$$

Sparse Autoencoders:

* Modify the cost function to enforce sparse "neurons".
  - ↳ l_1 regularization
  - ↳ sigmoid activation functions
  - ↳ large encoders

simpler tools

⇒

(i) have a target sparsity

(ii) measure actual sparsity

(iii) Give penalty

Check average neuron activity in batch.

[KL Divergence]

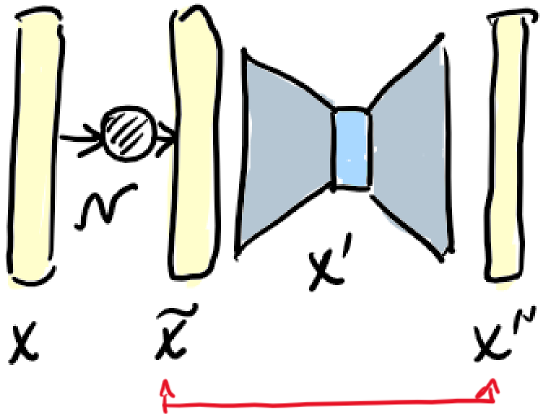
↳ Add sparsity loss to total loss.


Ateliers & Saveurs in Montreal

colab

De-noising Your Data:

- PIV Measurements
- Simulation data ⇒ FTLE -

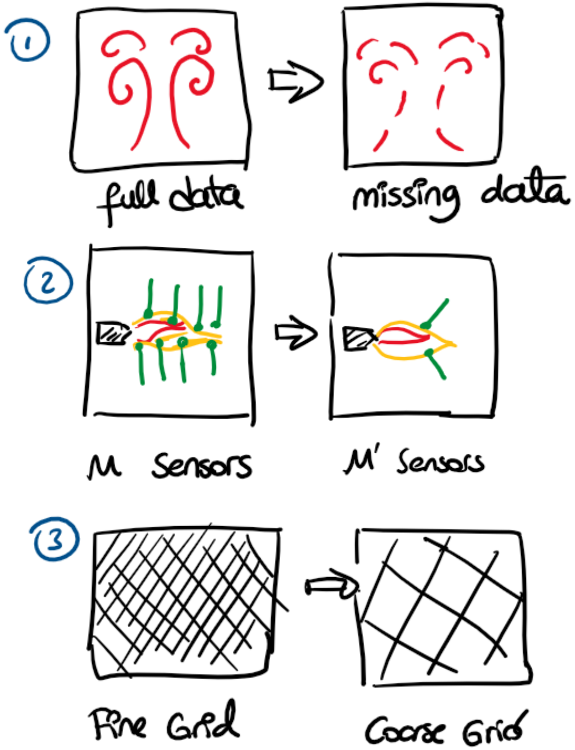


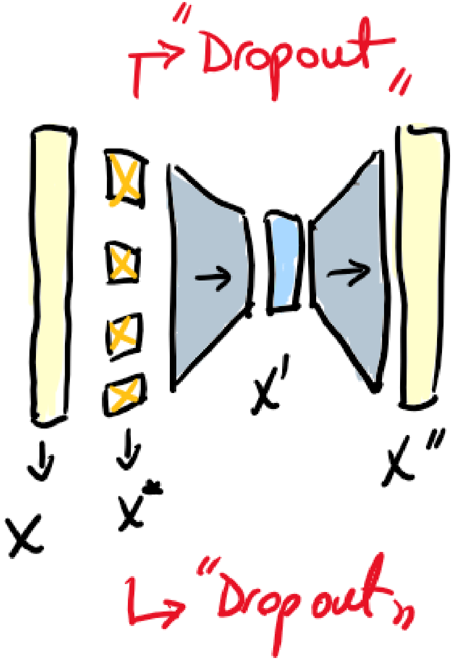
! "Clean data" is needed.

Obj: Learn to clean the data from noise.

colab

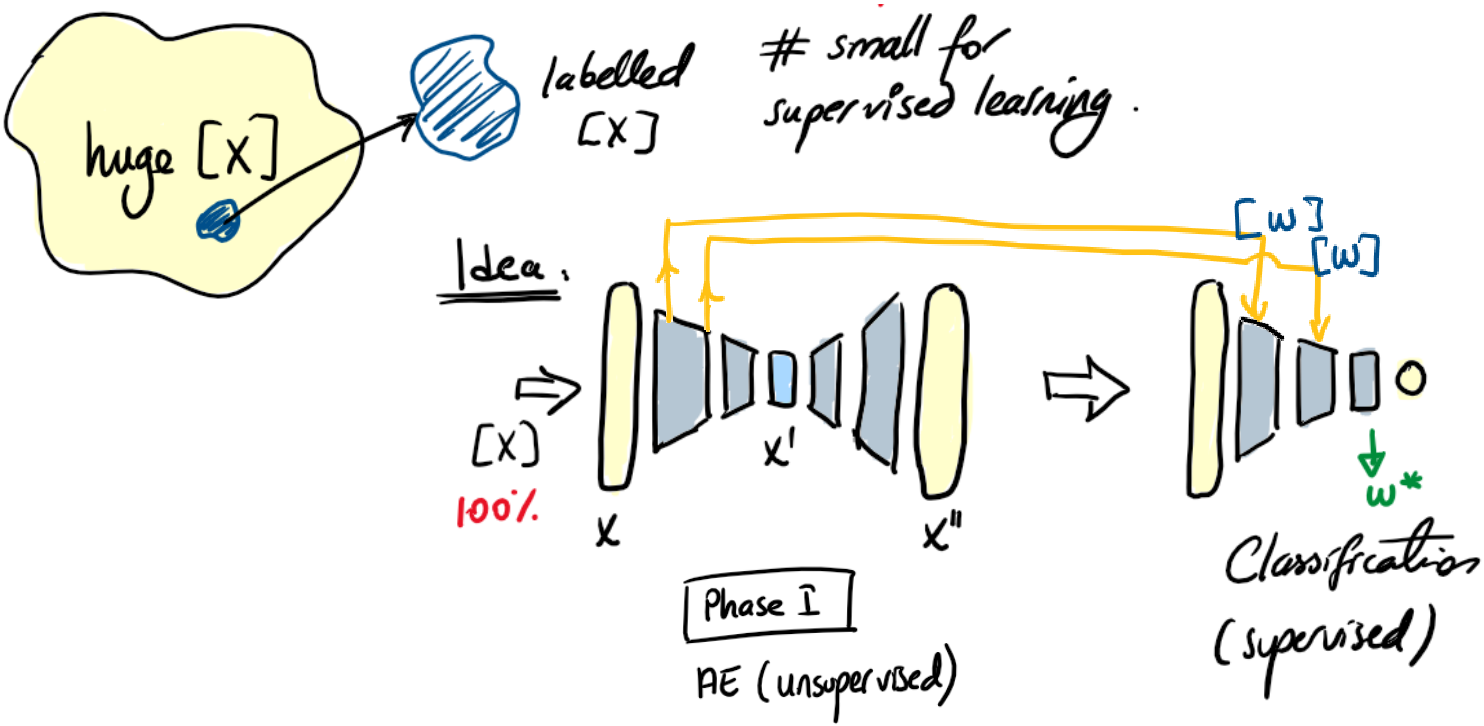
Interpolation Tool:





colab

Unsupervised Pretraining



colab

Additional Notes

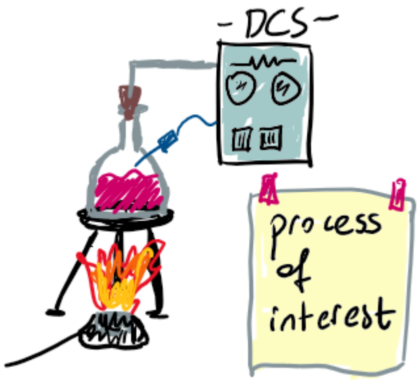
Dim. Reduction:

Computational
– preprocessing –

Feature Extraction
~ pattern recognition ~

Visualization

Idea:



$$\Rightarrow \quad X = \begin{bmatrix} — & \cdots & — \\ — & \text{product}_n & — \\ — & \cdots & — \end{bmatrix}$$

→ information m on the production line

Idea:

Assume:

- $\text{product} = \sum_{i=1}^{K} \text{process}_i$
- Features m is correlated to K steps in the production line;

Then:

$$X \rightarrow \boxed{D.R.} \rightarrow \tilde{X} \rightarrow \boxed{DR^T} \rightarrow X'$$

physical coordinates

"latent space"

physical coordinates

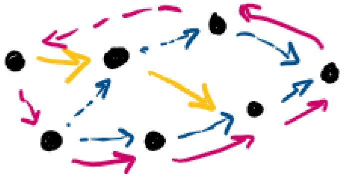
Encoding Decoding

if DR is capable of learning the patterns in the physical system;

$$X \approx X'$$

Idea:

Interpreting Patterns



- Physical system is composed of logical steps;
- Logical steps followed ⇒ "Regular product"
- Failure at some point ⇒ "Defect"

"Outlier Detection"

A.I.

"Something is wrong here."

Aim ⇒ Learn enough to detect outliers;