Applying Machine Learning to String Theory

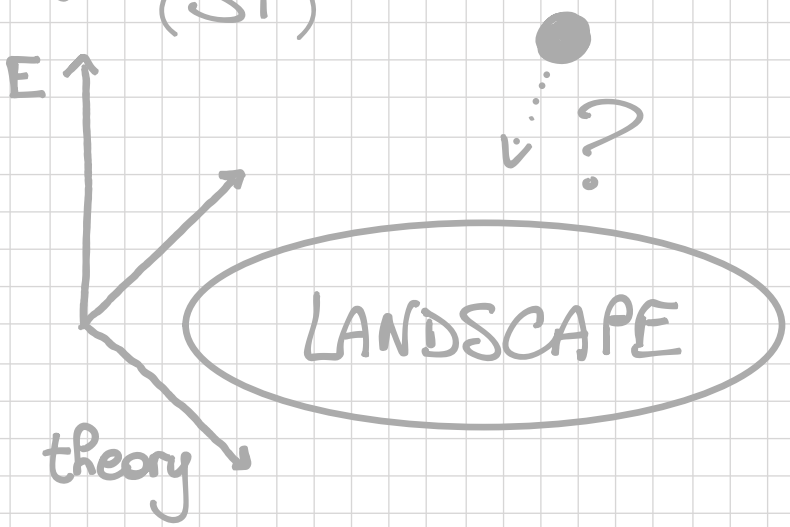
1. Motivations

Complex analytical problem $\Leftrightarrow$ "more feasible" optimisation problem $\Rightarrow$ Insights ?

1.1 Machine Learning for Physics

Pattern Exploration

e.g. 1) String Theory (ST): $\mathbb{R}^{1,9} \to \mathbb{R}^{1,3} \times X_6$



How many possibilities ?

F-theory: $10^{272\,000}$ [Taylor, Wang, 1511.03209]

How difficult?

NP-complete [Denef, Douglas, hep-th/0602072]

NP-hard

undecidable [Halverson, Ruehle, 1809.08279]

Non deterministic searches?

[Abel, Rizos, 1404.7359]

[Ruehle, 1706.07024]

[He, 1706.02714]

[Altman et al., 1811.06490]

[Ruehle, Phys. Rep. 839 (2020)]

e.g. 2) algebraic geometry and ST:

TOPOLOGICAL SPACE $\xrightarrow{?}$ TOPOLOGICAL INVARIANTS

[2007.15706]

[2108.02221]

e.g. 3) Symmetry detection

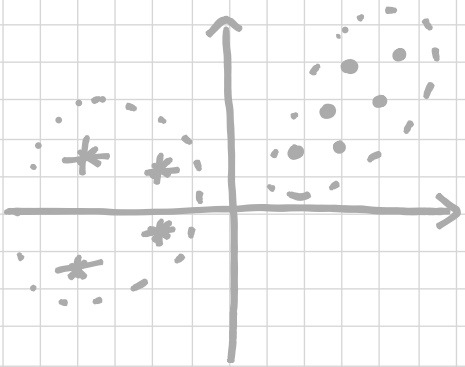
$$S: A \longrightarrow A, \quad x \longmapsto S(x)$$

$$f: A \longrightarrow \mathbb{R}, \quad z \longmapsto f(z)$$

Q: how to detect $x \in A \mid f(S(x)) = f(x)$?

A: $x \longmapsto f(x)$

HIGH DIMENSIONAL EMBEDDING through ML



⇒ find clusters of symmetric objects

[Krippendorf, Syvaeri, 2003.13679]

1.2 PHYSICS FOR MACHINE LEARNING

EXPLANATION and BEHAVIOURAL ANALYSIS

e.g. 1.1) NN - QFT duality

Let $f_{\theta,N}: \mathbb{R}^{d_{IN}} \longrightarrow \mathbb{R}^{d_{out}}$

in the limit $N \to \infty$:

$$P[f] \sim e^{-S_{GP}}$$

where

$$S_{GP} = \frac{1}{2}\int d^{d_{in}}x\, d^{d_{in}}x'\, f(x)\Sigma(x,x')f(x')$$

is a GAUSSIAN PROCESS:

$$G^{(k)}(x_1,\dots,x_k) = \frac{\int \mathcal{D}f\, f(x_1)\dots f(x_k) e^{-S_{GP}}}{\int \mathcal{D}f\, e^{-S_{GP}}}$$

KEY POINT: propagator $K(x,x')$

$$\int d^{d_{in}}x'\, \underbrace{K(x,x')}\Sigma(x';x'') = \delta^{d_{in}}(x-x'')$$

COVARIANCE of the output — strongly dep. on type of NN

What for $N < \infty$?

$$S_{GP} \to S = S_{GP} + \text{INTERACTIONS}$$

[Halverson, Maiti, Stoner, 2008.08601]
[Maiti, Stoner, Halverson, 2106.00694]
[Di Luo, Halverson, 2112.00723]

e.g. 2) $S = S_{GF} + \boxed{\Delta S}$

↓

coarse graining

↓

RG

Non pert. approaches show links with NN behaviour.

[Lahoche, Samary, Tamaazousti, 2002.10547]

[Erbin, Lahoche, Samary, 2108.01403]

1.3 MACHINE LEARNING FOR "ANALYTICAL METHODS"

e.g. 1) study F-theory compactif.

ML { INPUT: reflexive polytopes
OUTPUT: Fano toric 3-folds

↳ conjecture on E_6 compactif.

↳ analytical proof

[Carifio et al., 1707.00655]

e.g. 2) triangulations of reflexive polytopes

ML { INPUT: 4D reflexive polytopes
OUTPUT: triangulations

↳ estimate no. of triang. in the KS dataset → proof

[Altman et al., 1811.06490]

e.g. 3) analytic rules from 4D F-theory

ML { INPUT: divisors of Complex 3-folds base
OUTPUT: non-Higgsable gauge groups

↳ analytic rules → proof

[Wang, Zhang, 1804.07296]

e.g. 4) generation of superpotentials

ML { INPUT: known Superp.
OUTPUT: unknown Superp.

[Erbin, Krippendorf, 1809.02612]

2. NEURAL NETWORKS

- impact
- building blocks
- how to use them in physics

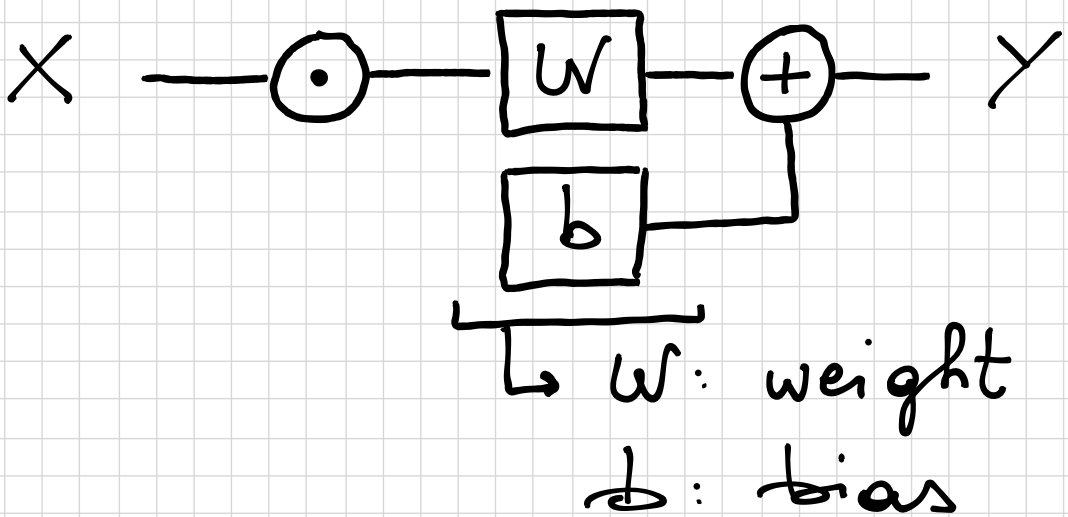
2.1 PERCEPTRON

First conceived by Rosenblatt, 1958 (Cornell Aeronautical Lab.):

i) pattern recognition

ii) first HARDWARE (software implementation came later)

iii) provide linear classification



In formulae:

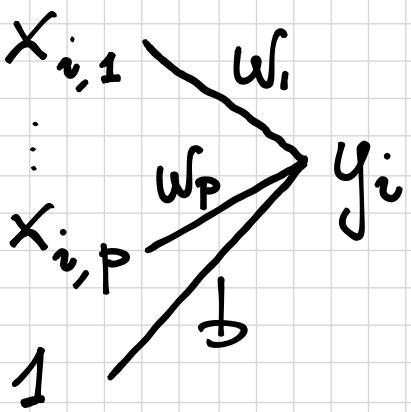
let $x_i \in \mathbb{R}^p$, $i = 1, \dots, N$ and

$$X = \begin{pmatrix} x_1^T \\ \vdots \\ x_N^T \end{pmatrix} \in \mathbb{R}^{N \times p}$$

then

$$P(X) = XW + b\mathbb{I}_N \in \mathbb{R}^N$$

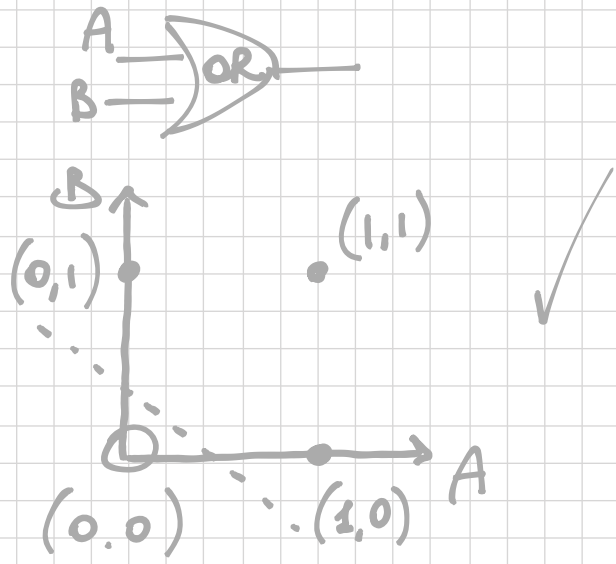
that is



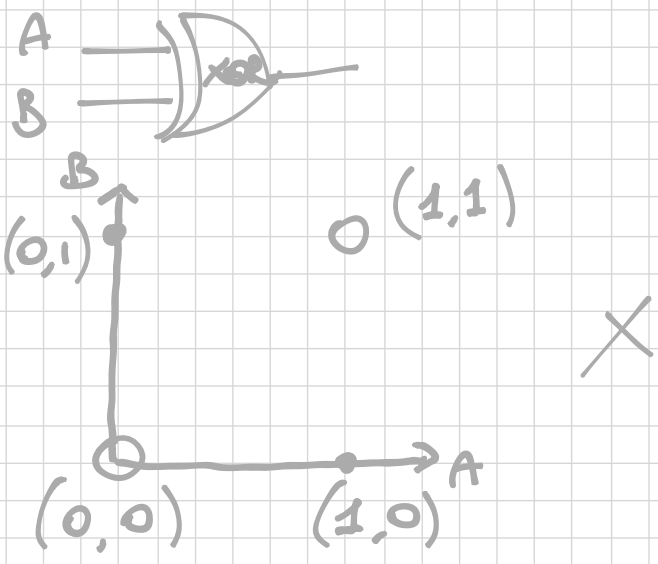
N.B: LINEAR ALGORITHM

[Historically, it lead to abandon NN research for ~20 years]

e.g.: LOGIC GATES



✓



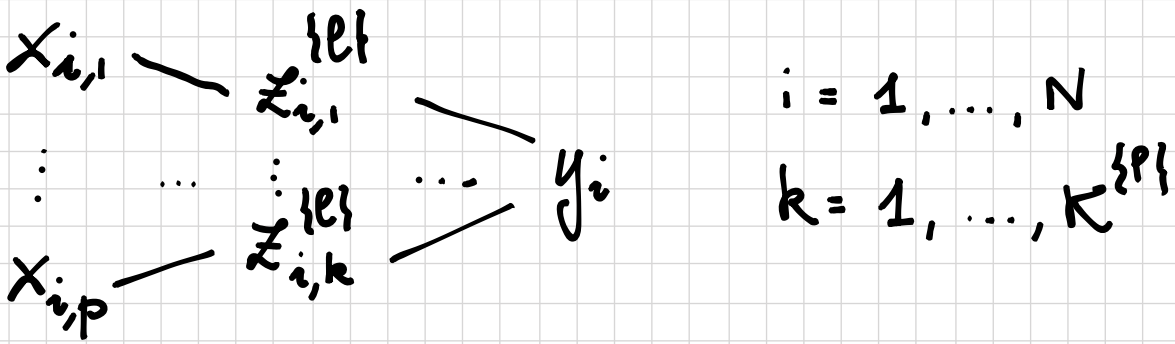
✗

2.2 Multi-Layered Perceptron and Fully Connected Networks

New function

$$Y = MLP_{\{(W^{[l]}, b^{[l]})\}_{l \in \{1 \dots L\}}}(X)$$

where :



(the bias term is omitted for simplicity)

In formulae,

def: $W^{[l]} \in \mathbb{R}^{K^{[l-1]} \times K^{[l]}}$

then: $Y = ((X W^{[1]}) W^{[2]}) W^{[3]} \dots$

$= X \tilde{W}$

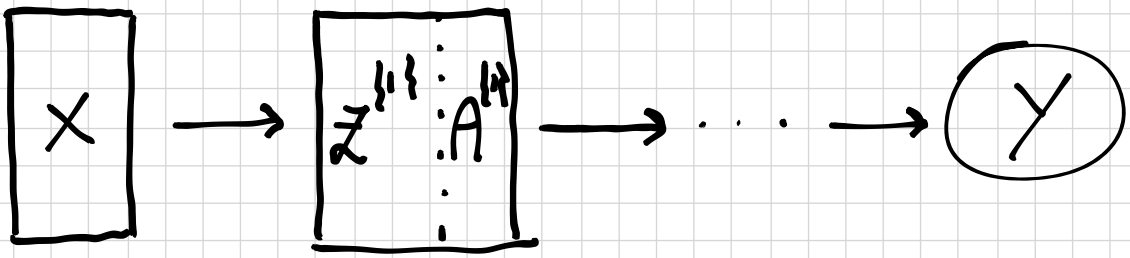
LINEAR

New ingredient:

$$Y = MLP_{\{(W^{\{\ell\}}, b^{\{\ell\}}, f^{\{\ell\}})\}_{\ell \in \{1 \dots L\}}}(X)$$

for each layer:

$$\begin{cases} Z^{\{\ell\}} = A^{\{\ell-1\}} W^{\{\ell\}} \\ A^{\{\ell\}} = f^{\{\ell\}}(Z^{\{\ell\}}) \\ A^{\{0\}} = X \end{cases}$$



"ACTIVATION FUNCTIONS"

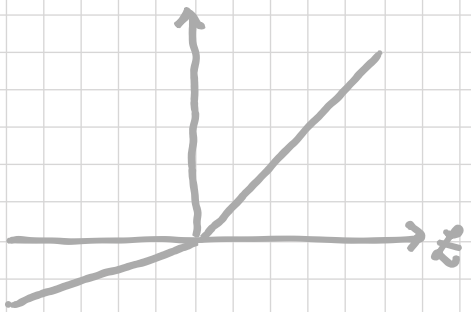
- non linear
- differentiable

e.g. 1) RECTIFIED LINEAR UNIT

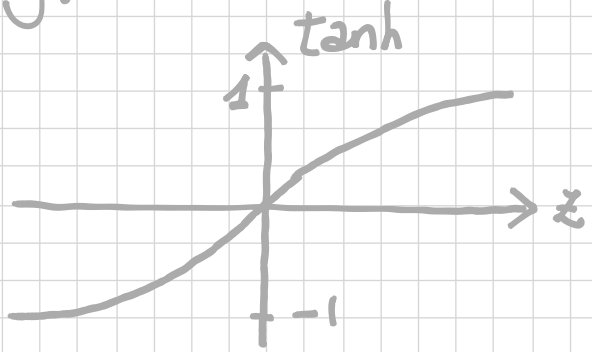
$$ReLU(z) = max(0, z)$$

e.g. 2) "LEAKY" - RELU

$$\text{LeakyReLU}_\alpha(z) = \max(\alpha z, z), \quad \alpha > 0$$



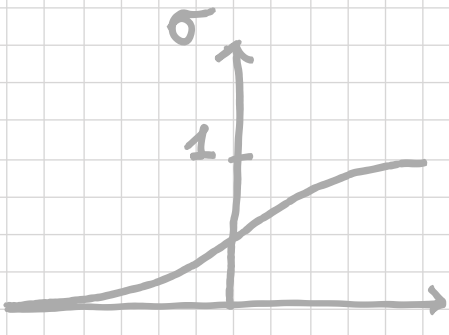
e.g. 3) Hyp. Tan.



constrain in $[-1, 1]$

e.g. 4) Sigmoid (CLASSIFICATION)

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad z \in \mathbb{R}^N$$



model probabilities

let $p \in [0, 1)$ be the prob. of an event:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \sigma^{-1}(p).$$

e.g. 4) Softmax

$$S(Z)_{i,k} = \frac{e^{z_{i,k}}}{\sum_{k=1}^{K} e^{z_{i,k}}}$$

where $Z \in \mathbb{R}^{N \times K}$, $k = 1, \dots, K$.

Used in MULTICLASS CLASS.:

- one class

$$y_i \in \{0, 1\}$$

- multiclass

$$y_i \in \{0, 1, \dots, K\}$$

Introduce ONE-HOT ENCODING

$$y_i \longrightarrow \tilde{y}_{i,k} = \begin{cases} 1 \text{ if } y_i = k \\ 0 \text{ otherwise} \end{cases}$$

finally $\tilde{y}_i \in \mathbb{R}^K$.

$$Y = \begin{pmatrix} 0 \\ 3 \\ 1 \end{pmatrix} \rightarrow \tilde{Y} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

2.4 Loss Functions

So far: FORWARD PASS

[Input → NN → Output]

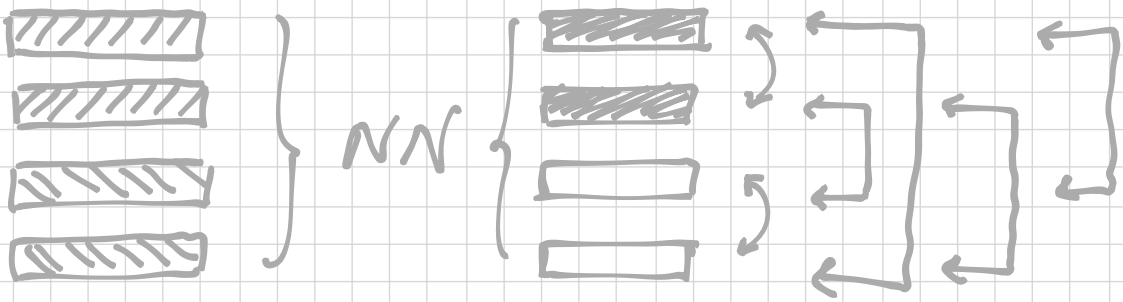
How to guide the "learning"?

CASE 1: you do not know the true result (the "ground truth")

→ UNSUPERVISED LEARNING

(tricky for neural networks)

e.g.



[Chen et al., 2002.05709]

[Koshla et al., 2004.11362]

"CONTRASTIVE LEARNING"

CASE 2: you know the answer
$\Rightarrow$ SUPERVISED LEARNING

- define a "distance" from the ground truth:

$$L(Y, \hat{Y}_\theta) = L(\theta)$$

where $\hat{Y}_\theta = f_\theta(X) \leftarrow NN_\theta(X)$.

- criterion based on LIKELIHOOD:

$$P(\hat{Y}_\theta | \theta) := \mathcal{L}(\theta | \hat{Y}_\theta)$$

$\Rightarrow$ joint prob. of the outp. given the model param. SEEN AS A FUNCTION OF THE PARAMS! (NOT a PDF!)

$\Rightarrow$ "which params are more likely to generate that output?"

$\Rightarrow$ $$\hat{\theta} := \arg\max_\theta \mathcal{L}(\theta | \hat{Y}_\theta)$$

MAXIMUM LIKELIHOOD ESTIMATE

or $$\hat{\theta} := \arg\min_\theta L(\theta), \quad L(\theta) := -\log \mathcal{L}(\theta | \hat{Y}_\theta)$$

a) CLASSIFICATION

$$y_i \in \mathbb{R}^K \quad \text{(OH enc.)}$$
$$i = 1, \dots, N$$

then
$$\mathcal{L}(\theta | \hat{y}_{\theta,i}) = \prod_{k=1}^{K} \hat{y}_{\theta,i,k}^{\,y_{i,k}}$$

(only the observation for which $y_{i,k} = 1$ appears)

e.g. BINARY CLASS. $K=2$

$$\begin{cases} y_0 + y_1 = 1 \\ \hat{y}_0 + \hat{y}_1 = 1 \end{cases} \rightarrow \mathcal{L}(\theta | \hat{y}_\theta) = \hat{y}_0^{\,y_0} (1-\hat{y}_0)^{1-y_0}$$

then the NLL or "loss function":

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(-\log \mathcal{L}(\theta | \hat{y}_{\theta,i}) \right)$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \sum_{i=1}^{K} y_{i,k} \log \hat{y}_{\theta,i,k}$$

$$= -\mathbb{E}_y [\log \hat{y}_\theta] := \text{CROSS-ENTROPY}$$

b) REGRESSION

$$y_i \in \mathbb{R} \quad i = 1, \dots, N$$

s.t. $$y_i = \hat{y}_{\theta,i} + \varepsilon_i$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$

"no depend. on the indep. var."

Then

$$\mathcal{L}(\theta \mid \hat{y}_{\theta,i}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{y}_{\theta,i})^2}{2\sigma^2}}$$

$$\Rightarrow L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \left\{ -\frac{(y_i - \hat{y}_{\theta,i})^2}{2\sigma^2} - \frac{1}{2} \log 2\pi\sigma^2 \right\}$$

$$\Rightarrow \hat{\theta} = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_{\theta,i})^2$$

"MEAN SQUARED ERROR"

Other more general examples:

i) MEAN ABSOLUTE ERROR

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_{\theta,i}|$$

ii) HUBER LOSS

$$L_\delta(\theta) = H_\delta(\theta) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} \frac{1}{2}(y_i - \hat{y}_{\theta,i})^2 & \text{if } |y_i - \hat{y}_{\theta,i}| < \delta \\ \delta(|y_i - \hat{y}_{\theta,i}| - \frac{\delta}{2}) & \text{otherwise} \end{cases}$$

iii) KULLBACK-LEIBLER DIV.

- let $P(x)$ be the cumulative distrib of an input x
- Q: how similar are two distrib. $P(z|x)$ and $Q(z)$?

$$\begin{aligned} \text{EVIDENCE} &:= \log P(x) \\ &= \log \int dz\, P(x|z) P(z) \\ &= \log \int dz\, \frac{Q(z)}{Q(z)} P(x,z) \\ &= \log \mathbb{E}_{z\sim Q}\left[\frac{P(x,z)}{Q(z)} \right] \end{aligned}$$

Jensen's inequality $\geq \mathbb{E}_{z\sim Q}\left[\log \frac{P(x,z)}{Q(z)}\right]$

$:=$ EVIDENCE LOWER BOUND

In other words (let θ be params):

$$\text{ELBO} := \mathbb{E}_{z\sim Q}[\log \mathcal{L}(\theta|x,z)] - \mathbb{E}_{z\sim Q}[Q(z)]$$

Now introduce the KL div.:

$$KL(Q(z)\,\|\,P(z|x)) :=$$

$$:= \mathbb{E}_{z\sim Q}\left[\log \frac{Q(z)}{P(z|x)}\right]$$

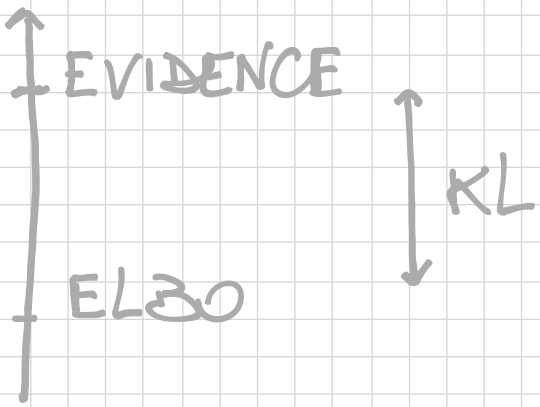
$$= \mathbb{E}_{z\sim Q}[\log Q(z)] - \mathbb{E}_{z\sim Q}\left[\log \frac{P(x,z)}{P(x)}\right]$$

With params θ:

$$KL(Q(z)\,\|\,P(z|x)) = \log \mathcal{L}(\theta|x) - \left[\mathbb{E}_{z\sim Q}[\log \mathcal{L}(\theta|x,z)] - \mathbb{E}_{z\sim Q}[\log Q(z)]\right]$$

$:=$ EVIDENCE - ELBO

Visually



at the heart of (Bayes) VARIATIONAL INFERENCE:

: if $Q(z)$ is impossible to compute we use $P(z|x)$ to approximate:

$$\hat{Q} = \arg\max_{Q} ELBO(P, Q)$$

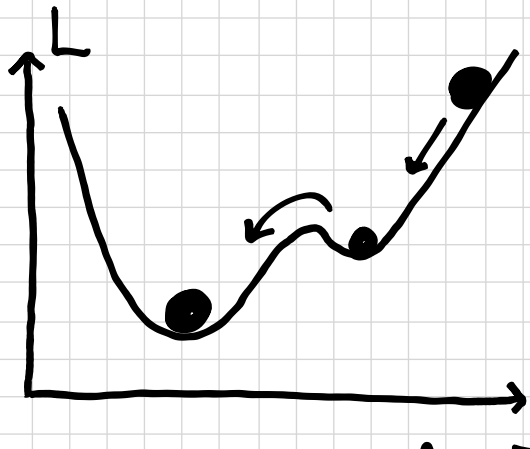
$$\Rightarrow \hat{Q} = \arg\min_{Q} KL(Q \| P).$$

2.5 Backpropagation

So far

FORWARD PASS $\longrightarrow$ LOSS FUNCTION

Where is training ?



$$\hat{\theta} := \operatorname*{argmin}_{\theta} L(\theta)$$

"GRADIENT DESCENT"

i) $\dot{\theta} = -\nabla_\theta L(\theta)$

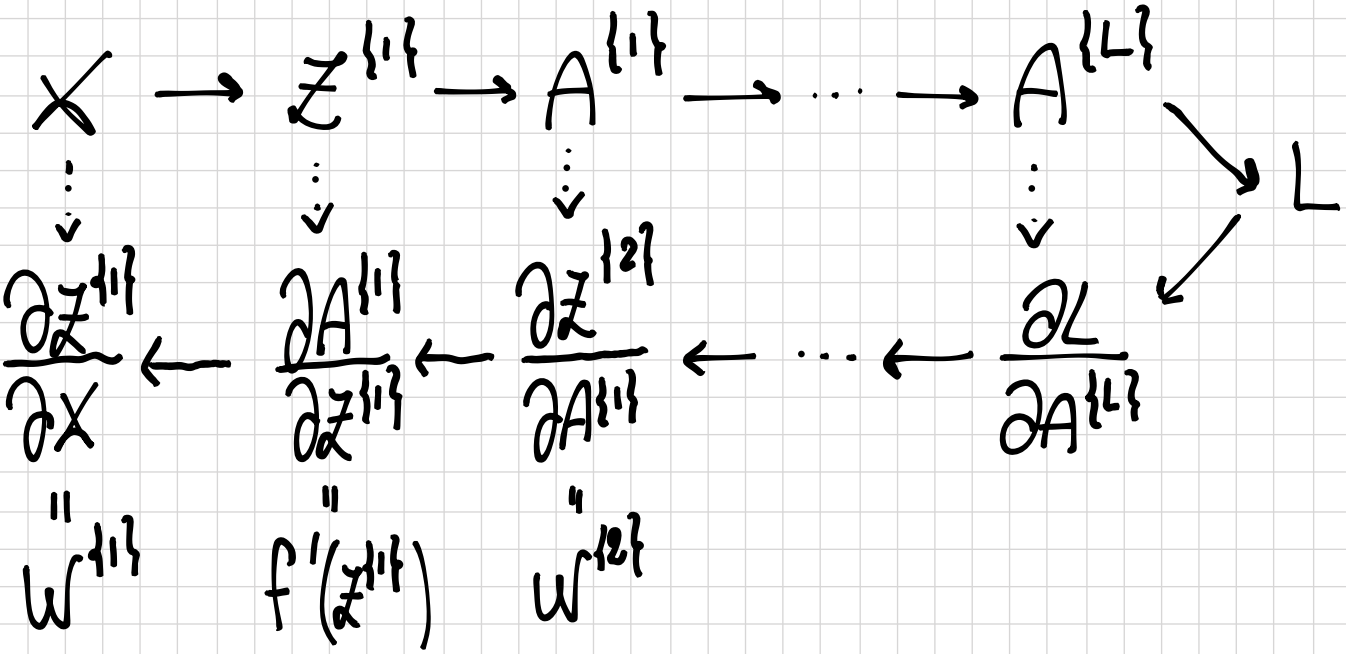
ii) let $\alpha > 0$ "LEARNING RATE"

iii) update $\theta \leftarrow \theta + \alpha\dot{\theta}$

How to do this efficiently?

BACKPROPAGATION

[Rumelhart, Hinton, Williams, Nature 323 (1986)]



Then let

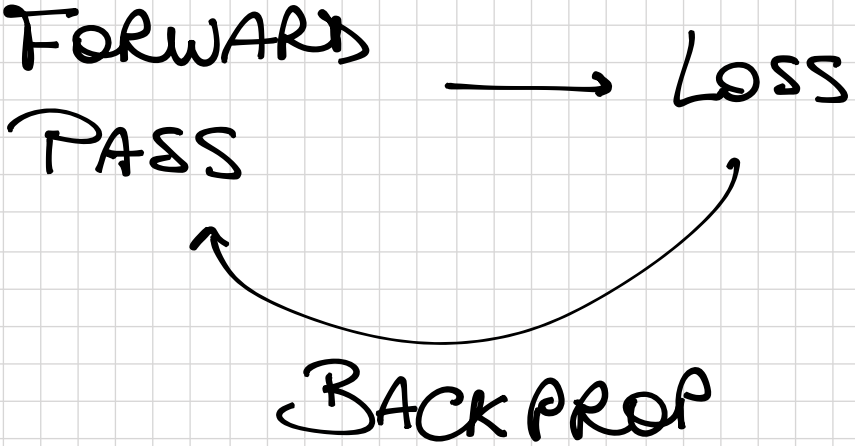
$$\delta^{[l]} = f'(z^{[l]}) W^{[l+1]} \cdots f'(z^{[L]}) \nabla_{A^{[L]}} L$$

and

$$\frac{\partial L}{\partial W^{[l]}} = \frac{\partial z^{[l]}}{\partial W^{[l]}} \cdot \frac{\partial L}{\partial z^{[l]}}$$
$$= A^{[l-1]} \cdot \delta^{[l]}$$

N.B.: frameworks such as PyTorch or Tensorflow are auto-grad. systems!

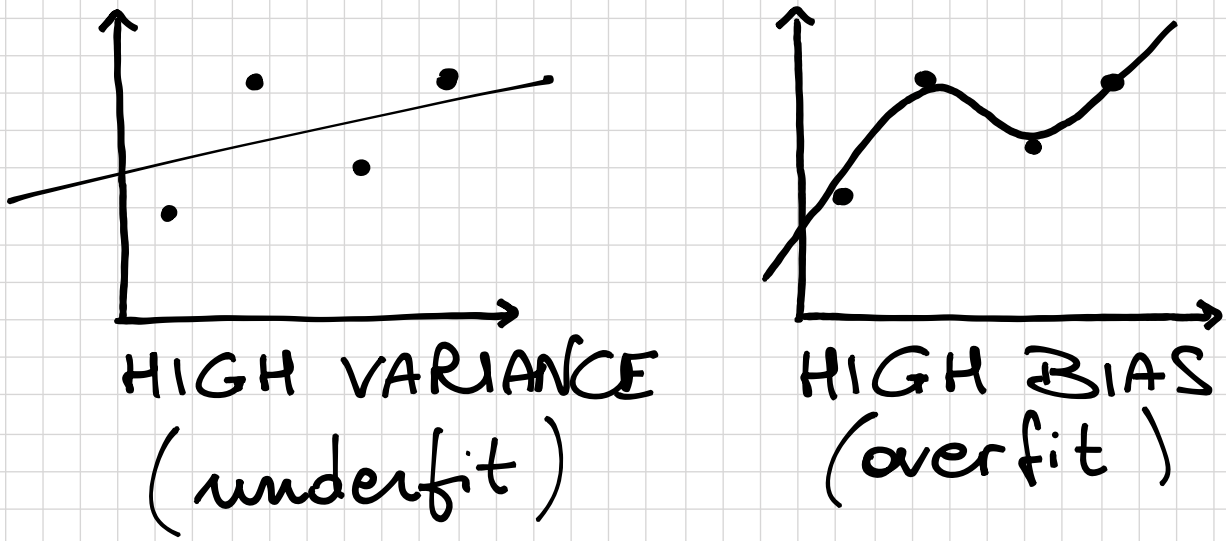
Thus we have



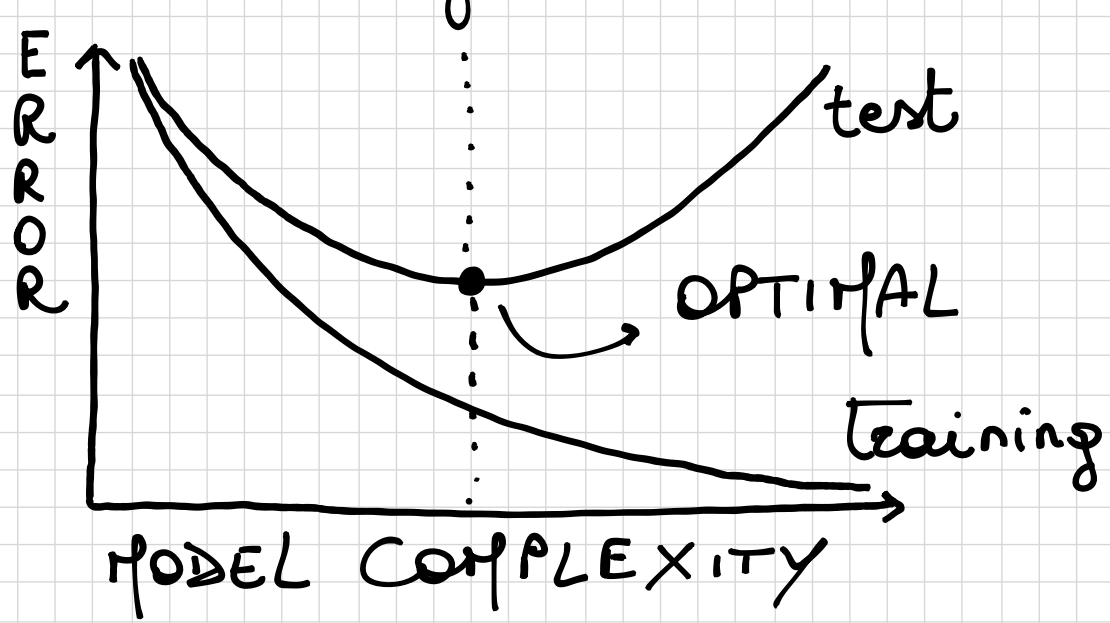
N.B.: there are good optimisations of GD, such as ADAM [Kingma, Ba, 1412.6980]

2.7 Variance - Bias Tradeoff

GD may induce issues:

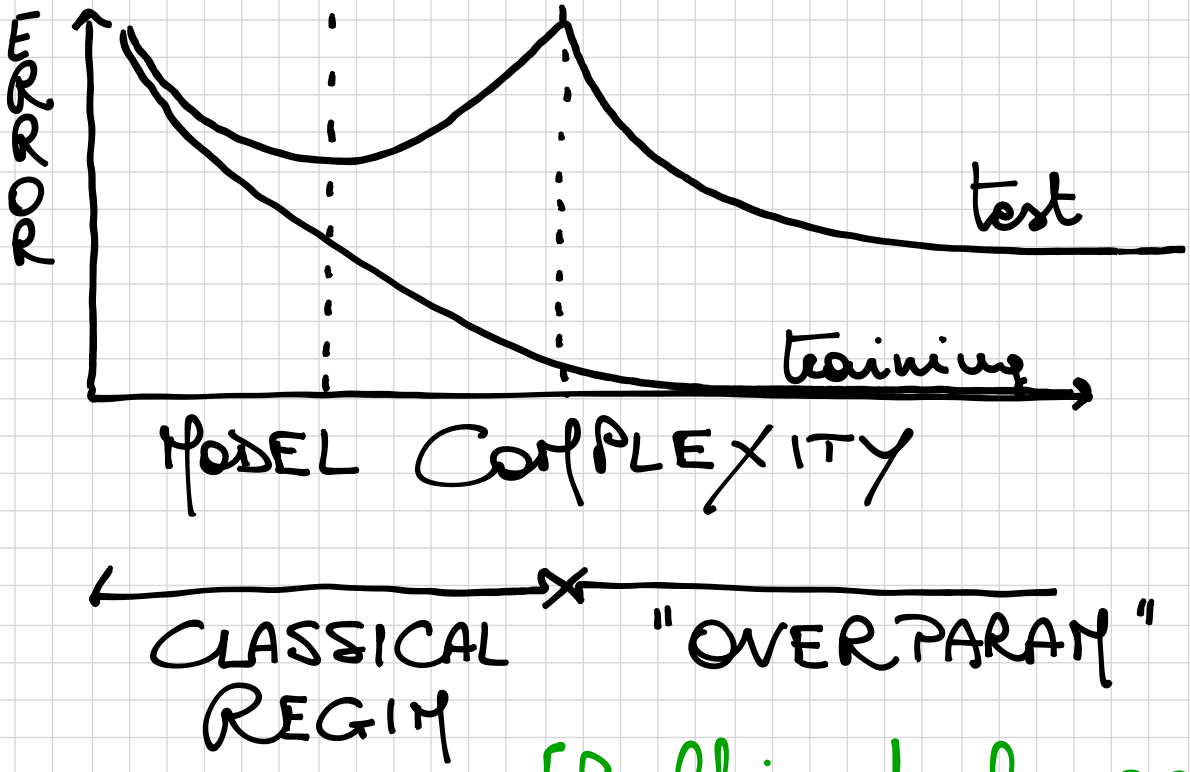


Classical argument:



⇒ must find the optimal point

Modern argument



[Belkin et al., 1812.11118]

NB for physicists:

- high complexity = hard interpr.
- less params = easier train.
- geometrical tasks are "easier": do we need all those parameters?

NB: the training error can be made arbitrarily small: NNs are UNIVERSAL APPROXIM.

[Hornik, Stinchcombe, White, Neural Networks 2 (1989)]

3. Modern Applications

Requirements for training:

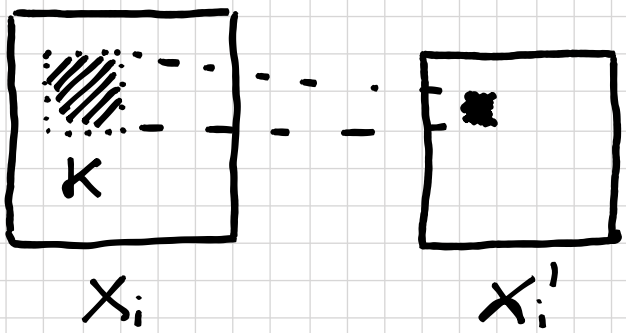
i) differentiable forward pass

ii) Convex loss function (at least locally)

Are there more interesting functions, other than MLPs?

3.1 Convolutional Neural Networks

Convolution filter:



$X_i \in \mathbb{R}^{N \times M}$

$K \in \mathbb{R}^{R \times S}$

$i = 1, \dots, W$

then

$$X'_{i,uv} = \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} X_{i,u+r,v+s} K_{r,s}$$

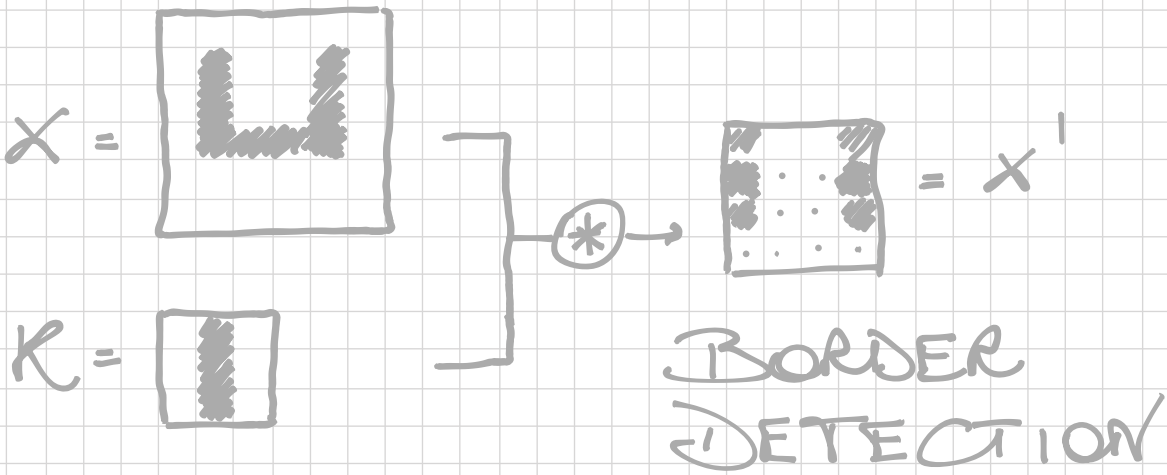
thus

$$X_i' \in \mathbb{R}^{N-R+1,\ M-S+1}$$

Why useful?

- local features
- shared parameters
- neighbouring properties

eg



As it is a DIFFERENTIABLE OPERATION, we can <u>learn</u> the filters (more than one at a time).

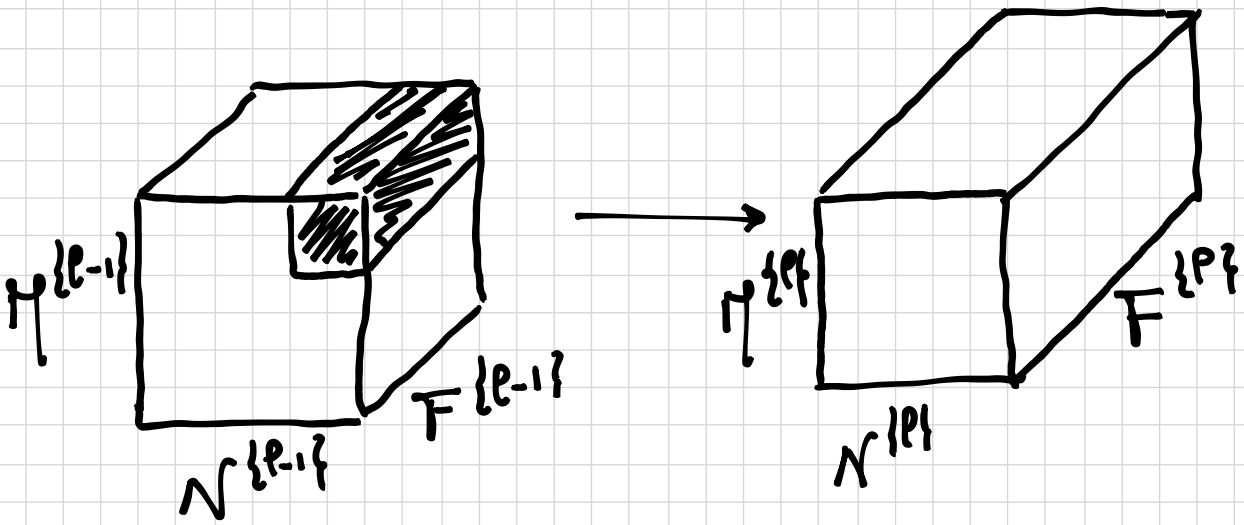
For each layer $l \in \{1, \dots, L\}$:

$$a_i^{[l]} \in \mathbb{R}^{N^{[l]} \times H^{[l]} \times F'^{[l]}}$$

$$K^{[l]} \in \mathbb{R}^{F'^{[l]} \times F^{[l]} \times R^{[l]} \times S^{[l]}}$$

we have:

$$\begin{cases} z_{i,uv}^{\{l\}[f]} = \sum_{r=0}^{R^{\{l-1\}}-1} \sum_{s=0}^{S^{\{l-1\}}-1} \sum_{q=0}^{F^{\{l-1\}}-1} a_{i,u+r,v+s}^{[l-1\}[q]} K_{r,s}^{\{l\}[q,f]} \\ a_{i,uv}^{\{l\}[f]} = f\left(z_{i,uv}^{\{l\}[f]}\right) \end{cases}$$



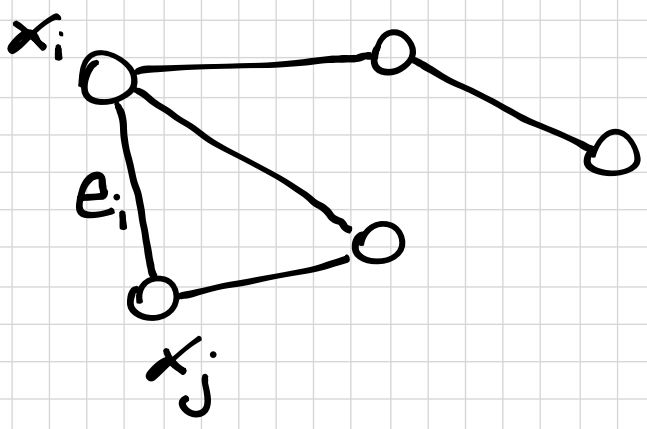
[LeCun et al., Neur. Comp. 1 (1989)]

3.2 Graph Neural Network

Graph $G = (V, E)$

vertices

edges



How to perform graph computations?

1) EDGE UPDATE

$$e'_k = \phi^e\left(e_k, x_k^{(r)}, x_k^{(s)}, u\right)$$

"compute forces"

2) EDGE AGGREGATION

$$\bar{e}'_i = \rho^{e \to v}\left(E_{[i]}\right)$$

"compute resulting forces"

3) NODE UPDATE

$$x'_i = \phi^v\left(\bar{e}'_i, x_i, u\right)$$

"update pos. and vel."

4) NODE AGGREGATION

$$\bar{e} = \rho^{e \to u}\left(E'\right)$$

"Sum ALL forces"

5) GLOBAL UPDATE

$$u' = \phi^u\left(\bar{e}', \bar{x}', u\right)$$

"kinetic energy"

5) GLOBAL AGGREGATION

$$\bar{x}' = \rho^{v \to u}(v') \quad \text{"total en."}$$

How to use NNs?

$$\phi^{\{e,v,u\}} \longrightarrow NN_\theta^{\{e,v,u\}}$$

and for instance

$$\rho^{e \to v}_{(i)} = \sum_{\{k \mid r_k = i\}} e'_{k(i)}$$

$$\rho^{v \to u} = \sum_i x_i'$$

$$\rho^{e \to u} = \sum_i e'_k$$

This is encoded in MESSAGE PARSING

[Gilmer et al., 1704.01212]

$$x_i' = \gamma\left(x_i, \square_{j \in N(i)} \phi(x_i, x_j, e_{j \to i})\right)$$

update ↙ ↳ aggregation ↳ message

e.g.: GRAPH CONVOLUTION

let $A = (a_{ij})$ s.t.

$$a_{ij} = \begin{cases} 1 & \text{if } x_i \text{ conn. to } x_j \\ 0 & \text{otherwise} \end{cases}$$

then

$$D = (d_{ij}) \Leftarrow d_{ii} = \sum_j a_{ij}$$

$$\hat{A} = A + I$$

then:

$$x' = f\left[\left(D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}\right) x W + b\right]$$

- only local features
- aggregation of neighbours.

4. Applications to String Theory

4.1 Computer Vision for Complete Intersections Calabi-Yau Manifolds

DEF: CY : complex manifold X

s.t. $\dim_{\mathbb{C}} X = N$

and $Hol(X) \subseteq SU(N)$

(or $c_1(X) = 0$)

$\Rightarrow$ ST: no known metrics for COMPACT CY

A simple way to build CY is through COMPLETE INTERSECTIONS:

let $\mathcal{A} = \mathbb{P}^{n_1} \times \dots \times \mathbb{P}^{n_m}$

det $\begin{cases} p_{a_k^i}(z) = \sum_{I_{1k} \dots I_{a_k^i}} z^{I_{1k}} \dots z^{I_{a_k^i}} = 0 \\ p_{a_k^i}(\lambda z) = \lambda^{a_k^i} p_{a_k^i}(z) \end{cases}$

Topologically:

$$X = \left[\begin{array}{c|ccc} \mathbb{P}^{n_1} & a_1^1 & \cdots & a_k^1 \\ \vdots & \vdots & & \vdots \\ \mathbb{P}^{n_m} & a_1^m & \cdots & a_k^m \end{array}\right] \in \mathbb{N}^{m \times k}$$

s.t. $\dim_{\mathbb{C}} X = \sum_{r=1}^{m} n_r - k = N$

$$n_r + 1 - \sum_{\alpha=1}^{k} a_\alpha^r = 0 \quad \forall r = 1, \ldots, m$$

Why?

$X \rightarrow$ predict topological invariants

$\Rightarrow$ direct connection to EFT in ST.

Particularly:

$$h^{(r,s)} = H_{\partial}^{(r,s)}(X, \mathbb{C})$$

e.g.: $\mathcal{N}=3 \Rightarrow h^{(1,1)}$ and $h^{(2,1)}$ count the no. of chiral mult. in HE or hyper- and vector mult. in Type II

[He, 1706.02714]
[He, Lukas, 2009.02544]

Datasets

$\mathcal{N}=3$ [Candelas, Dale, Lutken, Nucl. Phys. B 298 (1988)]
[Green, Hubsch, Lutken, Class. Quant. Grav. 6 (1989)]
[Anderson et al., 1708.07907]

$\mathcal{N}=4$ [Gray, Haupt, Lukas, 1303.1832 and 1405.2073]

Attempts (through NNs)

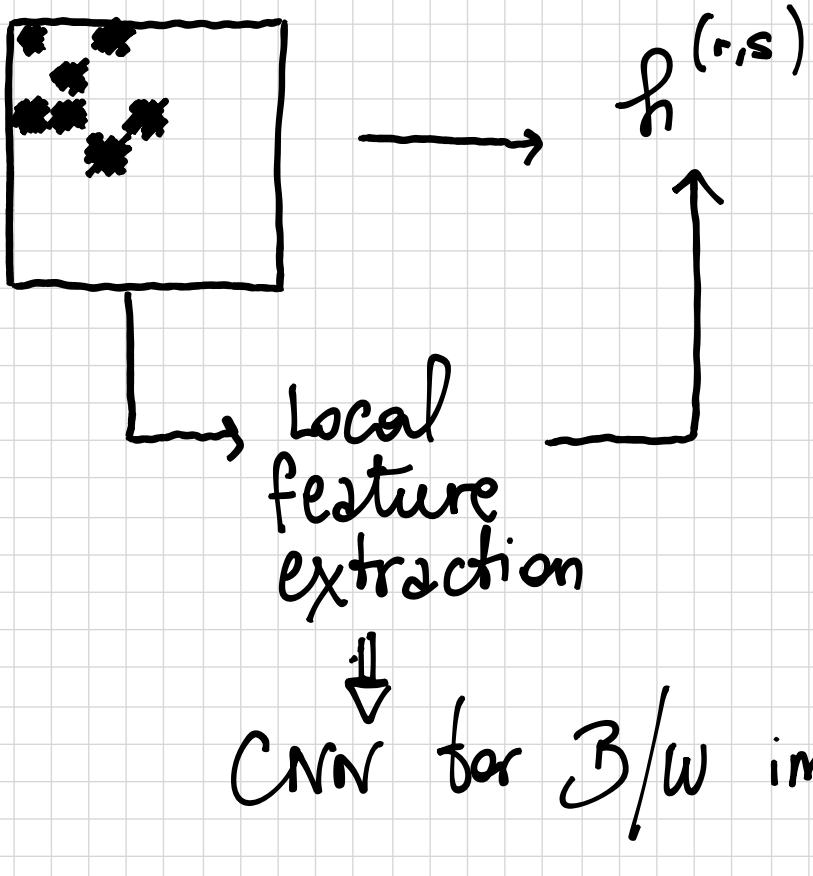
$$f : \mathbb{N}^{m,k} \longrightarrow \mathbb{N}$$
$$x \longmapsto f(x) = h^{(r,s)}$$

replace f by NN_θ :

FULLY CONNECTED $\Rightarrow \sim 75\%$ acc. on $h^{(1,1)}$

[Bull et al.,]

However :



How to build such arch. ?

1) DATA ANALYSIS [2007.15706]

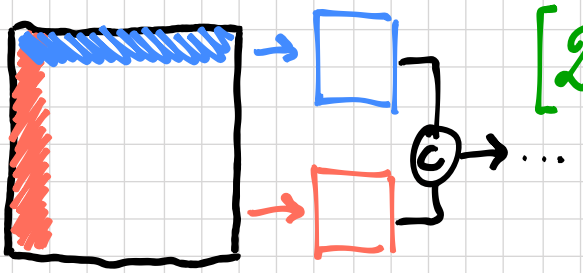
i) "good info" in small portions of X

ii) strong influence of the no. of proj. spaces and the dim. of the spaces (notice: rows and columns of X)

iii) engineered var. are strongly corr. => no need for all of them

2) INCEPTION MODULE

- 1d kernels [Szegedy et al., 1409.4842]
- maximal size [2007.13379]

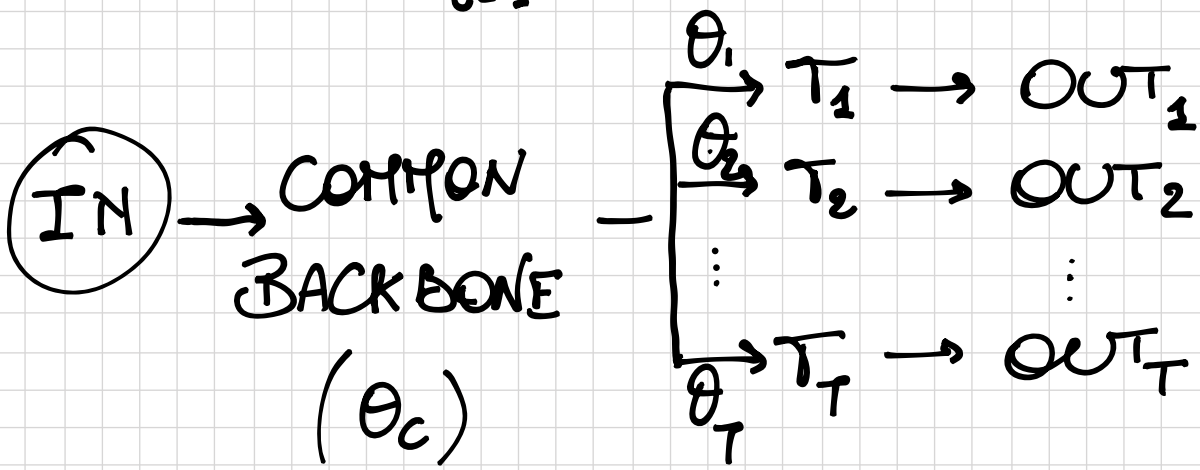


- let AI decide everything
- no prior math knowledge

$$N = 3 \Rightarrow h^{(1,1)} \sim 100\%$$
$$(h^{(2,1)} \sim 50\%)$$

- Deep multi-task learning: [2108.02221]

$$L(\theta) = \sum_{t=1}^{T} L_t(\theta), \quad \theta = \{\theta_c, \theta_1 \dots\}$$



- only 1 model for 92 1497 conf. matr.
- highlight dep.
- no overfit + outliers

$$N = 4 \Rightarrow \left(h^{(1,1)}, h^{(2,1)}, h^{(3,1)}, h^{(2,2)}\right) \sim (100\%, 100\%, 96\%, 85\%)$$

using 80 : 10 : 10