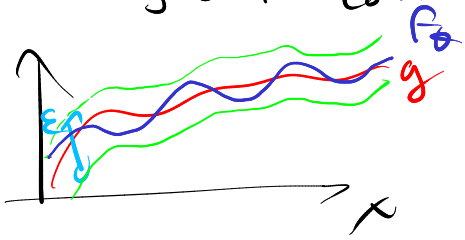
Deep Learning vs. classical ML & optimiz°

Properties (or lack of) of DL

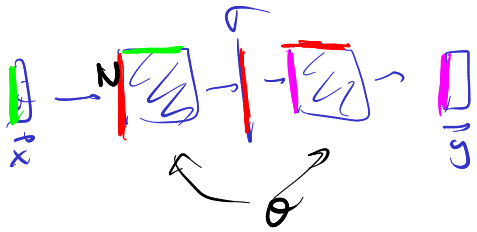
- no guarantee

Universal approxim° theorems (expressive power)

[Cybenko 89, Hornik 91] with just one hidden layer, one can approximate any cº fº (on a compact set) arbitrarily well



neural net



$\forall f, \forall$ cº $g, \forall \varepsilon > 0, \exists N, \exists \theta$ s.t.

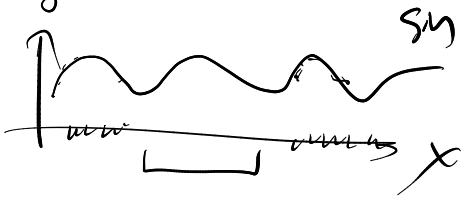
$$\forall x, \quad \| f_\theta(x) - g(x) \| < \varepsilon$$

⚠ Existence network

→ doesn't provide the solut°

→ " tell wether it is easy to find it

→ " " anything about generalizº



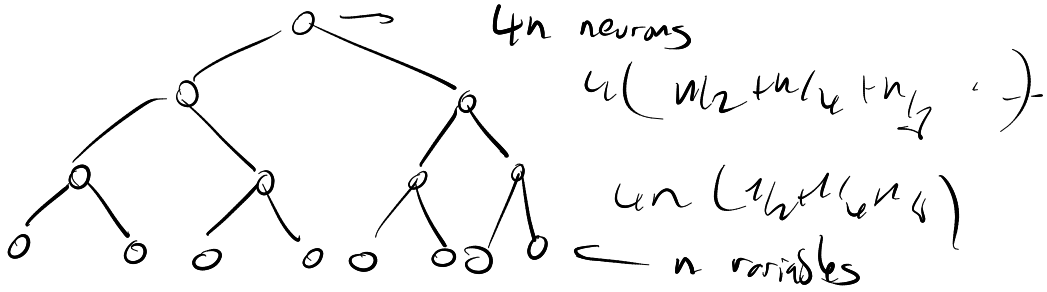
Depth simplifies the task

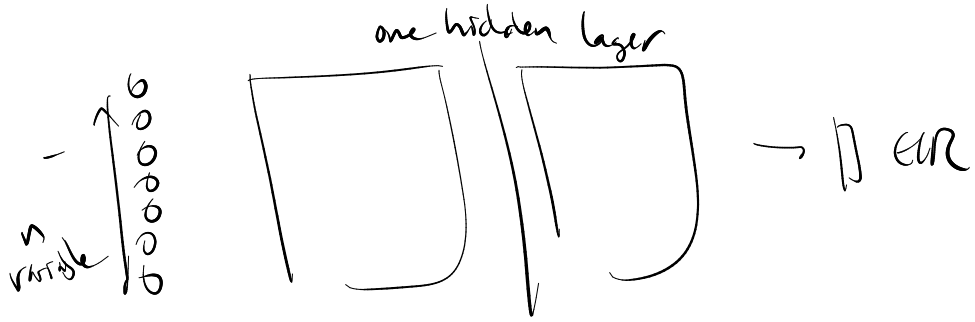
[Lin & co, 2017] multiplication of variables

- ≃ multiplicat° of 2 variables with 4 neurons

$\sigma(a(x+y)) \quad \sigma(a(x-y)) \quad - \quad a^2 \quad 4^2$

small





need 2^n neurons to be correct on n-ary inputs

[Telgarsky 2015]

$\Rightarrow 2^n$ triangles

Flat network:

2 layers $\rightarrow$ $2^{n/2}$ neurons

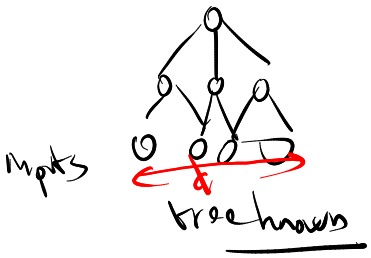
depth $\sqrt{n}$ $\rightarrow$ $2^{\sqrt{n}}$ neurons

width

[Poggio & al, 2016] Th: if target P° = compositional tree of input variables

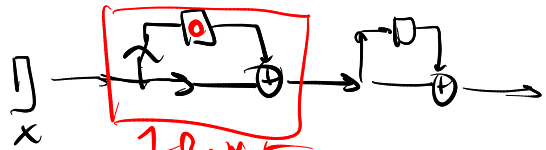
then #samples required to train a similar-shaped network: $O(d)$

Compare with e^d



[Lin & Jegelka, 2018] : depth is sufficient

flat network



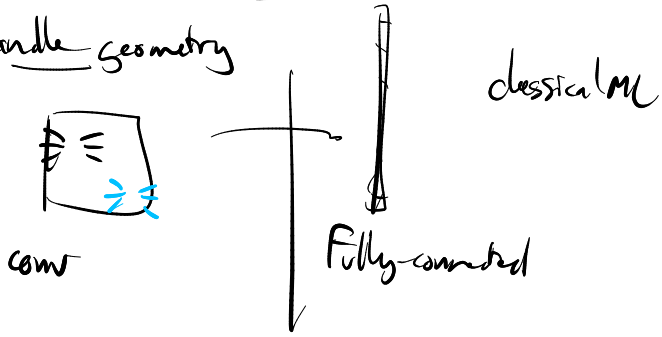
universal approx property (with $\nearrow$ depth)

Does it work? when?

- computer vision
- NLP (text)

} huge success

ex: vision → CNN → equiv invariant to translations
→ hierarchical models
→ handle geometry



- on the opposites:
  data not structured
  ex: medical file $\begin{pmatrix} \text{age} \\ \#\text{cigarettes (day)} \\ \vdots \end{pmatrix}$ small

⚠ NN need many samples to be trained
⇒ dataset small ⇒ classical ML = better

- **Alpha Go : RL**
  ↳ 2-0 ! - 44 millions of training games (>> one life)
  ≃ total humanity ?
  - trained for only 4 hours on 5 000 TPU
  ≃ very good GPU graphic card
  > 2 years of 1 TPU
  ≥ 1000 years of 1 GPU

- **Computational power :** 10^{11} parameters in GPT-4

Gap between classical ML & optim° vs. DL

Reminder : classical ML

- dataset samples (x_i, y_i)
- estimate a f : $f : x_i \to \simeq y_i$
- quantify "goodness" with a "loss" f° training criterion

$$\Rightarrow \inf_{f \in \mathcal{F}_{(\theta)}} \mathbb{E}\left[\text{loss}(f(x_i), y_i)\right] + R(f)$$

predefined parameterized functional space

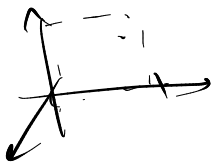
avoid overfit ! generalize

regularizer ↳ smooth

Minimum Description Length principle (MDL):
- Occam's razor: simpler explanations are more probable

Potential issues with DL:
- 10^6 parameters → MDL?
- models able to overfit (easily)
- no regularizer
- possible to learn huge models without overfitting?
- - - - - - - - with fewer samples than parameters (estimator convergence)
- highly non-convex optim° pb in a high-dim space

vs. classical optim°:

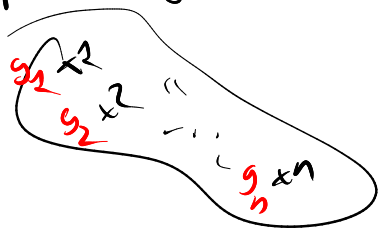


- add noise to optim° process ⇒ better results
- we want to optim: accuracy
  we train: cross-entropy } ≠ criterion!
- common recommandation: new task / architecture : check you are able to overfit! (small part of data)

A closer look at overfitting:

[Zhang & co, 2017] → huge models can do the job (might not overfit)
→ " " " completely overfit also

Th: dataset of n samples in $\mathbb{R}^d$
⇒ 2-layer network with $2n+d$ weights (with ReLU activ° fct)
that can represent any f on that dataset

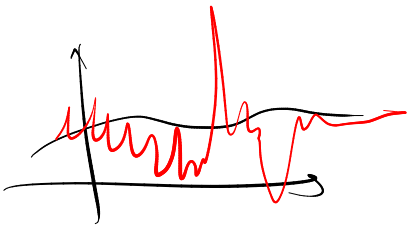


- capacity of a network = not an issue
↳ Rademacher Vapnik-Ch. complexities

Palliatives for regularizⁿ

- About adding a regularizer:

$$\int_{x \in \Omega} \|\nabla_x f(x)\|^2 \, dx \qquad p(x)\,dx$$



$x \notin \Omega$

$$\simeq \frac{1}{N} \sum_{x \in D} (\ldots)$$

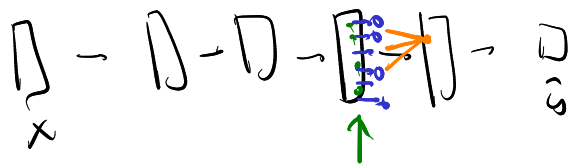
↑ of size N

← tractable approx of the regularizer

Th: just checking that, over all possible binary inputs, a NN outputs always 0 is NP-hard

Dropout

select randomly 50% of activities



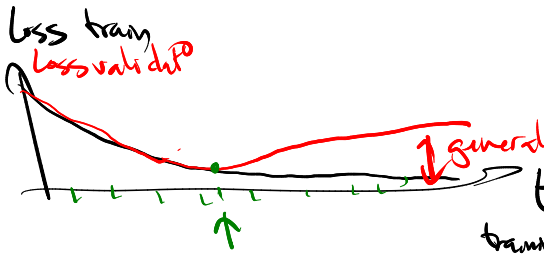
activities = outputs of layer

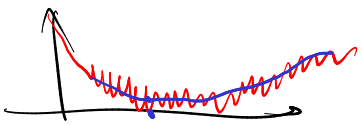
ensemble methods = robustify

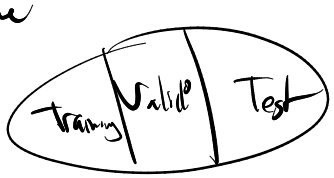
at test time: activities / 2

Early stopping

model selection

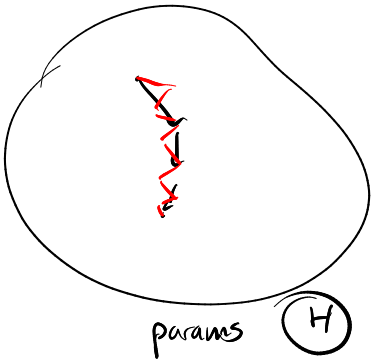






Optimiz° noise acts as a regularizer

source of noise: SGD : train on minibatches

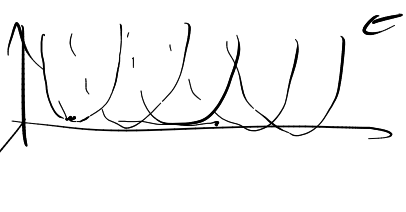


[Passio] around convergence → local minimums

Loss ≃ Hessian matrix

$\frac{d^2 \text{Loss}}{d\theta^2}$ ↘ if >0 : then no pb

↘ if not:

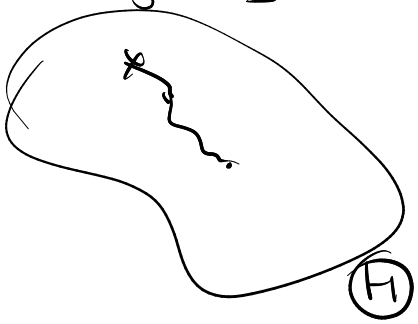


⇒ make the $H > 0$

↳ add a weight regularizer:

$+\alpha \sum_{\theta_i} |\theta_i|^2$ ← weight decay = L^2 norm on parameters

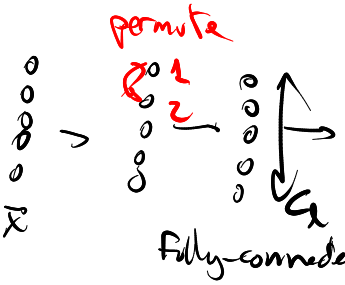
$\theta_i \leftarrow \eta\left(-\nabla L - 2\alpha\theta_i\right)$



≡ prior that favors smooth functions

Optimization Landscape

Local minima :



$C_1! \, C_2! \, C_3! \ldots C_L!$

huge

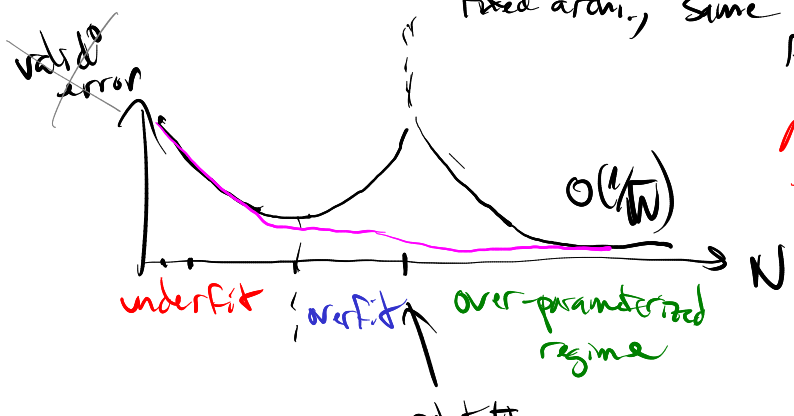
- So many parameters: ⇒ strong quality of local minima
  - Fixed ML task
  - add more & more parameters ⇒ the more params, the easier
    - ↳ $\nabla L = 0$ means none of $\frac{\partial L}{\partial \theta}$ improves the loss
    - billions of directions

Over-parameterization helps

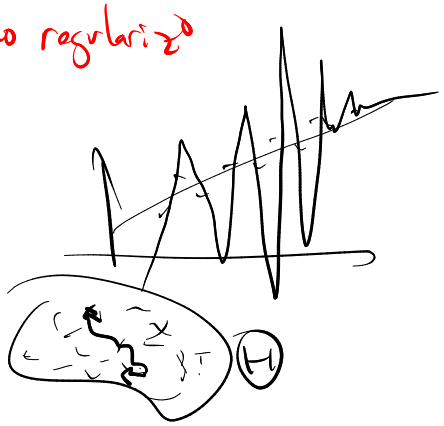
[Geiger et al, 2019]

- Neural Tangent Kernel
- "double ∇ descent"

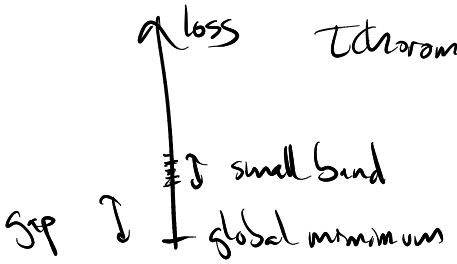
Fixed archi., same width N for all layers



1) no regularization



Gradient of local minima



[Choromanska, 2015]

spin glasses

About MD2

→ lost ?

→ redundancies in parameters

[Welling] action recog. videos real-time on smartphone

↳ /100 with ~ accuracy

→ tensorize
→ discr.
→ ~~distillation~~
→ pruning

} compression