COLLÈGE DE FRANCE
1530

Chaire de Physique de la Matière Condensée
Antoine Georges

Réseaux de Neurones, Apprentissage et Physique Quantique

Cours 1 – Introduction:

Apprentissage, réseaux de neurons, survol des applications en physique quantique

Cycle 2022-2023
9 mai 2023

Image Credit: Javier Robledo-Moreno

COLLÈGE DE FRANCE
1530

Chaire de Physique
de la Matière Condensée
Antoine Georges

Machine Learning and Neural Networks for Quantum Physics

Lecture 1 – Overview.

Introduction to Machine Learning and Neural Networks

Cycle 2022-2023
9 mai 2023

Image Credit: Javier Robledo-Moreno

Mailing List

(Weekly announcement of lecture and seminar, etc.)

Send email to: listes-diffusion.cdf@college-de-france.fr
Subject line: subscribe chaire-pmc.ipcdf
…or: unsubscribe chaire-pmc.ipcdf

You can also just send me an email to be placed on the list

Website:

https://www.college-de-france.fr/site/antoine-georges/index.htm

Lectures are video recorded
and available on the website

Aim Of These Lectures

- Merely a broad-band introduction to some aspects of the field
- Aimed at:
- Stimulating interest
- Paving the way to understanding the seminars and reading further literature
- I am currently learning the field too! ☺
- NOT meant to be a technical introduction to ML
- NOT meant to be exhaustive

Mardi 9 mai

COURS (9h30) :

Introduction à l'apprentissage par réseaux de neurones et survol des applications en physique quantique.

COURS (11h30) :

Représentation des états quantiques par réseaux de neurones (*Neural Quantum States*).

SÉMINAIRE (14h30 -16h) :

Filippo Vicentini (École Polytechnique, Paris et EPFL, Lausanne)

Neural Quantum States for Finite Temperature and Open systems, with a practical introduction to NetKet

Mardi 16 mai

COURS (9H30) :

Introduction à la tomographie quantique

SÉMINAIRE (11h30) :

Giuseppe Carleo (EPFL, Lausanne)

Time-Dependent Neural Quantum States

Mardi 23 mai

COURS (9h30) :

Représentations des états quantiques fermioniques par réseaux de neurones

SÉMINAIRE (11h30) :

Juan Carrasquilla (Vector Institute, Toronto)

Quantum States with Neural Networks : Representations and Tomography

Mardi 30 mai

COURS (9H30) :

Réseaux de neurones, apprentissage et fonctionnelle de densité: applications à la structure electronique (1)

SÉMINAIRE (11h30) :

Giulio Biroli (ENS, Paris)

Renormalization Group Theory and Machine Learning

Mardi 6 juin

COURS (9h30) :

Réseaux de neurones, apprentissage et fonctionnelle de densité: applications à la structure electronique (2)

SÉMINAIRE (11h30) :

Ambroise van Roekeghem (CEA-LITEN, Grenoble)

Machine Learning Force Fields for Materials Science

Colloque

12,13 et 14 juin

Precision Many Body Physics 2023

Les 12 et 14 juin dans l'amphithéâtre Maurice Halbwachs
Le 13 juin – matin : dans l'amphithéâtre Guillaume Budé
Le 13 juin – après-midi : session posters en salles 7 et 8

Crédit image : Javier Robledo-Moreno (NYU et CCQ)

Acknowledgements

Johannes Flick
CCQ & CUNY/CCNY

Anna Dawid
CCQ

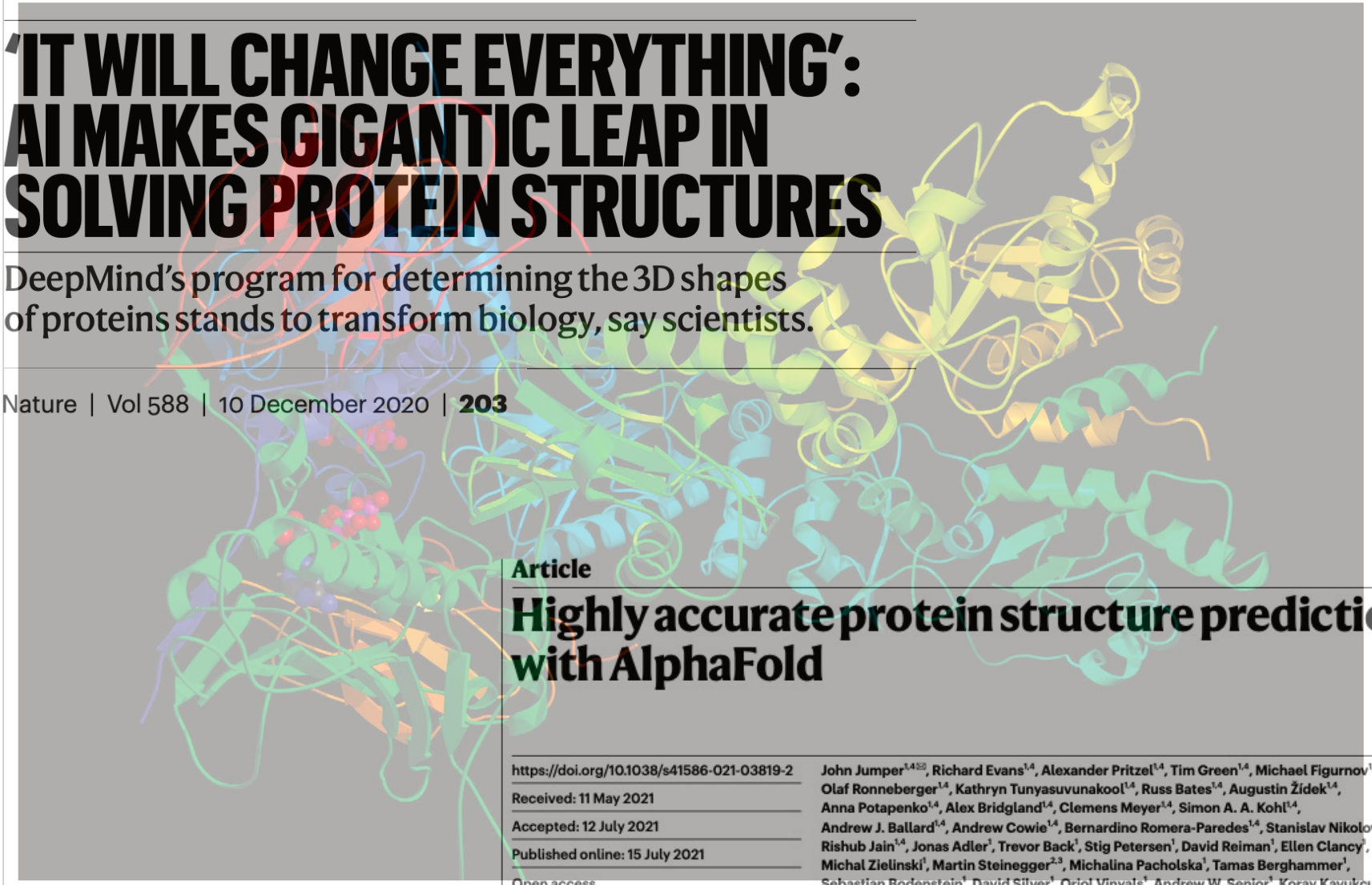
James Stokes
CCQ → U.Michigan

AI is taking the world by storm… and sciences too!

‘IT WILL CHANGE EVERYTHING’: AI MAKES GIGANTIC LEAP IN SOLVING PROTEIN STRUCTURES

DeepMind’s program for determining the 3D shapes of proteins stands to transform biology, say scientists.

Nature | Vol 588 | 10 December 2020 | 203



A protein’s function is determined by its 3D shape.

Article

Highly accurate protein structure prediction with AlphaFold

https://doi.org/10.1038/s41586-021-03819-2
Received: 11 May 2021
Accepted: 12 July 2021
Published online: 15 July 2021
Open access
Check for updates

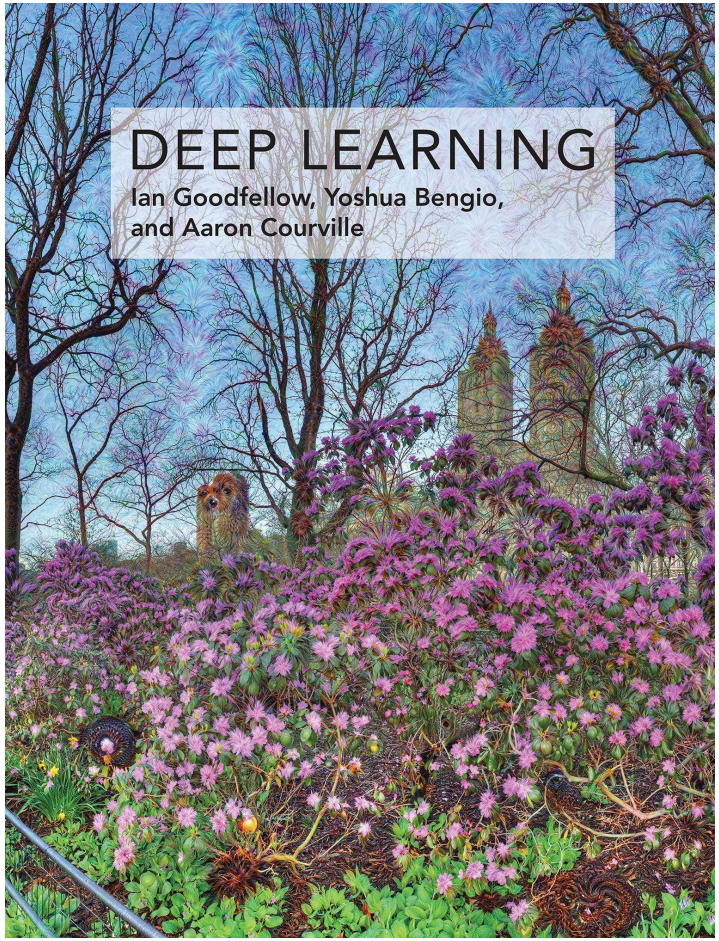
John Jumper[1,4✉], Richard Evans[1,4], Alexander Pritzel[1,4], Tim Green[1,4], Michael Figurnov[1,4], Olaf Ronneberger[1,4], Kathryn Tunyasuvunakool[1,4], Russ Bates[1,4], Augustin Žídek[1,4], Anna Potapenko[1,4], Alex Bridgland[1,4], Clemens Meyer[1,4], Simon A. A. Kohl[1,4], Andrew J. Ballard[1,4], Andrew Cowie[1,4], Bernardino Romera-Paredes[1,4], Stanislav Nikolov[1,4], Rishub Jain[1,4], Jonas Adler[1], Trevor Back[1], Stig Petersen[1], David Reiman[1], Ellen Clancy[1], Michal Zielinski[1], Martin Steinegger[2,3], Michalina Pacholska[1], Tamas Berghammer[1], Sebastian Bodenstein[1], David Silver[1], Oriol Vinyals[1], Andrew W. Senior[1], Koray Kavukcuoglu[1], Pushmeet Kohli[1] & Demis Hassabis[1,4✉]

Two DALL.E (OpenAI) pictures of
`Erwin Schrödinger, Matisse-style’

What is ML? (In a Nutshell)

- An ML algorithm is an algorithm able to 'learn' from data (i.e. evolve/adapt as more data is provided) in order to perform a specific task
- Mitchell (*Machine Learning, 1997):* `A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E'
- We will briefly review:
- T: Task
- E: Experience/Training Methods
- P: Performance

Some Useful General Refs on ML



- General ML:
- Goodfellow, Bengio and Courville, *Deep Learning,* MIT Press
- Lectures by Stéphane Mallat at the Collège de France (online and notes)
- Hasti, Tibshirani and Friedman, *The elements of Statistical Learning*, Springer
- Michael Nielsen's online book http://neuralnetworksanddeeplearning.com
- In relation to Physics:
- Mehta et al. A high-bias, low variance introduction to Machine Learning for physicists Phys. Reports 810, 1 (2019)
- Carleo et al. *Machine Learning and the Physical Sciences* Rev Mod Phys 91, 045002 (2019)

The website `Papers with Code' is also very useful: https://paperswithcode.com

FIDLE: Une plateforme de formation en français

https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle/-/wikis/home

FIDLE

Formation Introduction au Deep Learning

by CNRS / MIAI / UGA

1re Journée

Deep learning pour la Science

Save the date !

ORSAY
12 MAI 2023

9h00 Accueil café

9h30 Introduction

9h45 **BLOOM**, un modèle linguistique autorégressif (LLM) capable de parler 46 langues et 13 langages de programmation.
Lucile Saulnier (Hugging Face), François Yvon (CNRS)

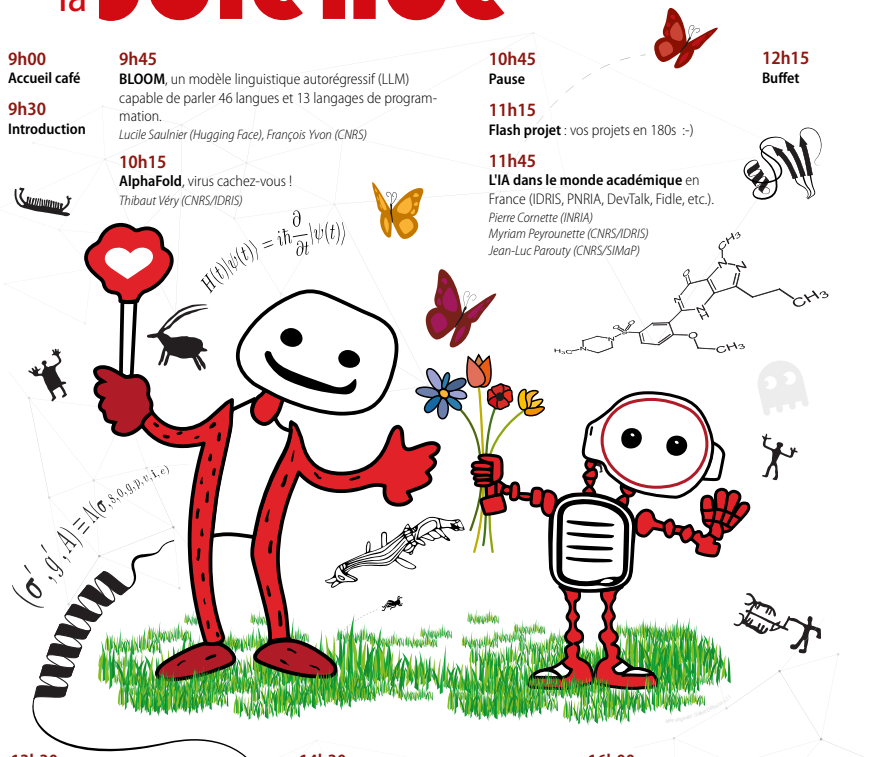
10h15 **AlphaFold**, virus cachez-vous !
Thibaut Véry (CNRS/IDRIS)

10h45 Pause

11h15 **Flash projet** : vos projets en 180s :-)

11h45 **L'IA dans le monde académique** en France (IDRIS, PNRIA, DevTalk, Fidle, etc.).
Pierre Cornette (INRIA)
Myriam Peyrounette (CNRS/IDRIS)
Jean-Luc Parouty (CNRS/SIMaP)

12h15 Buffet



13h30 **L'enfer des données :** je n'ai pas assez de donnée, mes données sont fausses ou mes données sont exotiques, que faire ?
Laurent Risser (CNRS/IMT), Achile Mbogol Touye (UGA/EFELIA)

14h00 **Gérer un tsunami de données :** building new brains for giant astronomical instruments using AI.
Damien Gratadour (CNRS/OBSPM)

14h30 **PlugAI :** simplifier l'usage de l'IA pour l'imagerie médicale.
Michaël Sdika (Creatis)

15h00 **Détection de brouillard à Paris :** developing and using HazeNet to forecast particulate pollution related low visibility days in Paris.
Chien Wang (CNRS/OBS-MIP)

15h30 Pause

16h00 **IA : qualité, méthode et reproductibilité**
Mauricio Diaz (INRIA)

16h30 **L'IA d'aujourd'hui et de demain :** état de l'art et perspectives pour la Science.
Bertrand Cabot (CNRS/IDRIS)

17h00 **Mot de la fin**
Jean-Luc Parouty (CNRS/SIMaP)
Sylvie Thérond (CNRS/IDRIS)

https://fidle.cnrs.fr/jdls2023

FIDLE

VENDREDI 12 MAI 2023
ORSAY Blandin LPS

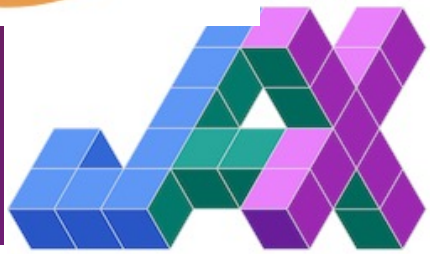
Some Useful General Refs on ML with a Focus on Quantum Systems

- J.Carrasquilla *Machine learning for quantum matter Advances in Physics 2020, Vol 5 1797528*
- A.Dawid et al. *Modern applications of machine learning in quantum sciences (Lecture Notes) arXiv:2204.04198*
- J.Carrasquilla and G.Torlai *How To Use Neural Networks To Investigate Quantum Many-Body Physics PRX Quantum 2, 040201 (2021) (Tutorial)*
- J.Schmidt et al. *Recent advances and applications of machine learning in solid-state materials science npj Computational Materials (2019)*

Online Journal Club (every 2 weeks): http://ultracold.org/menu/

Some Code Libraries

- NumPy https://numpy.org
- Scikit https://scikit-learn.org/stable/
- PyTorch: https://github.com/pytorch/pytorch
- Tensorflow (Google): https://github.com/tensorflow
- Jax (Google): https://github.com/google/jax
- NetKet: https://www.netket.org
- See e.g. https://www.coursera.org/articles/python-machine-learning-library



What is ML? (In a Nutshell)

- An ML algorithm is an algorithm able to 'learn' from data (i.e. evolve/adapt as more data is provided) in order to perform a specific task
- Mitchell (*Machine Learning, 1997):* `A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E'
- We will briefly review:
- T: Task
- E: Experience/Training Methods
- P: Performance

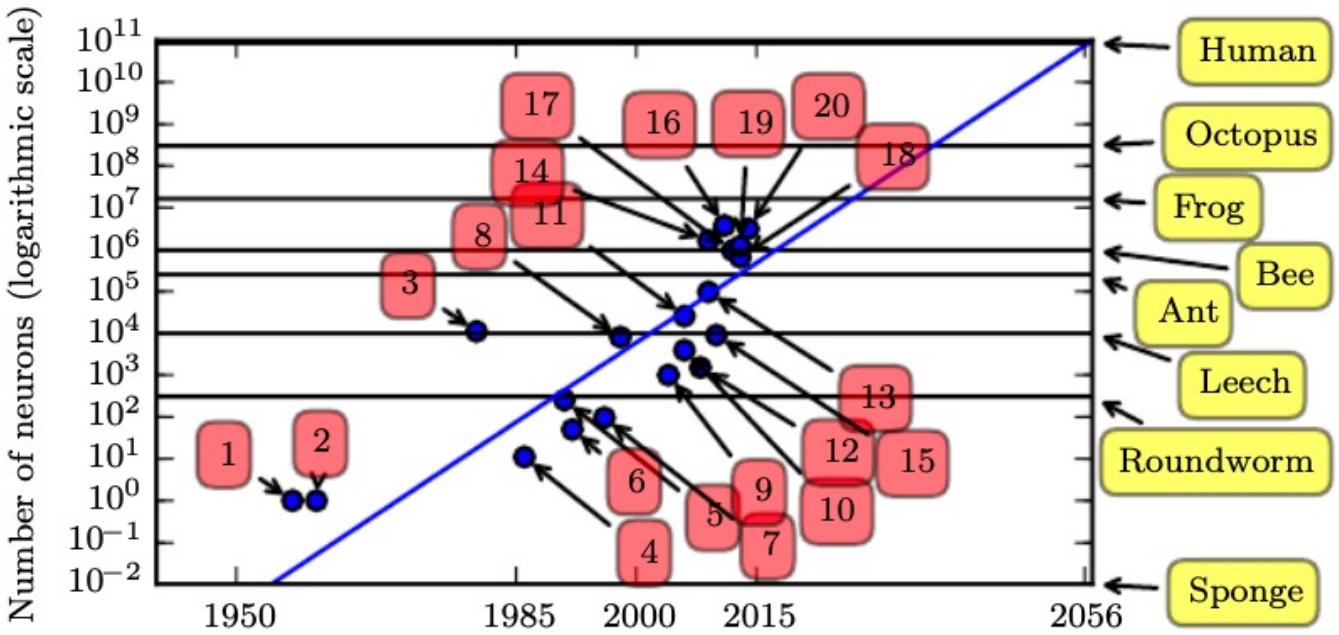
The Task T

- Data: a vector **x**=(x_1,…,x_n) in a space of (high) dimension n (n=number of `features')
- Example: **x** is an image, n the number of pixels
- We want to `learn' a function: $y = f(\mathbf{x})$
- Example: classification $y \in \{0, 1\}$; or $y \in \mathbb{R}$
- Find a parametrization of this function:

$$y = f(\mathbf{x}; \boldsymbol{\theta})$$

  - Learning: optimize the parameters θ
    - Architecture of neural network

      + Trained parameters = `Model'



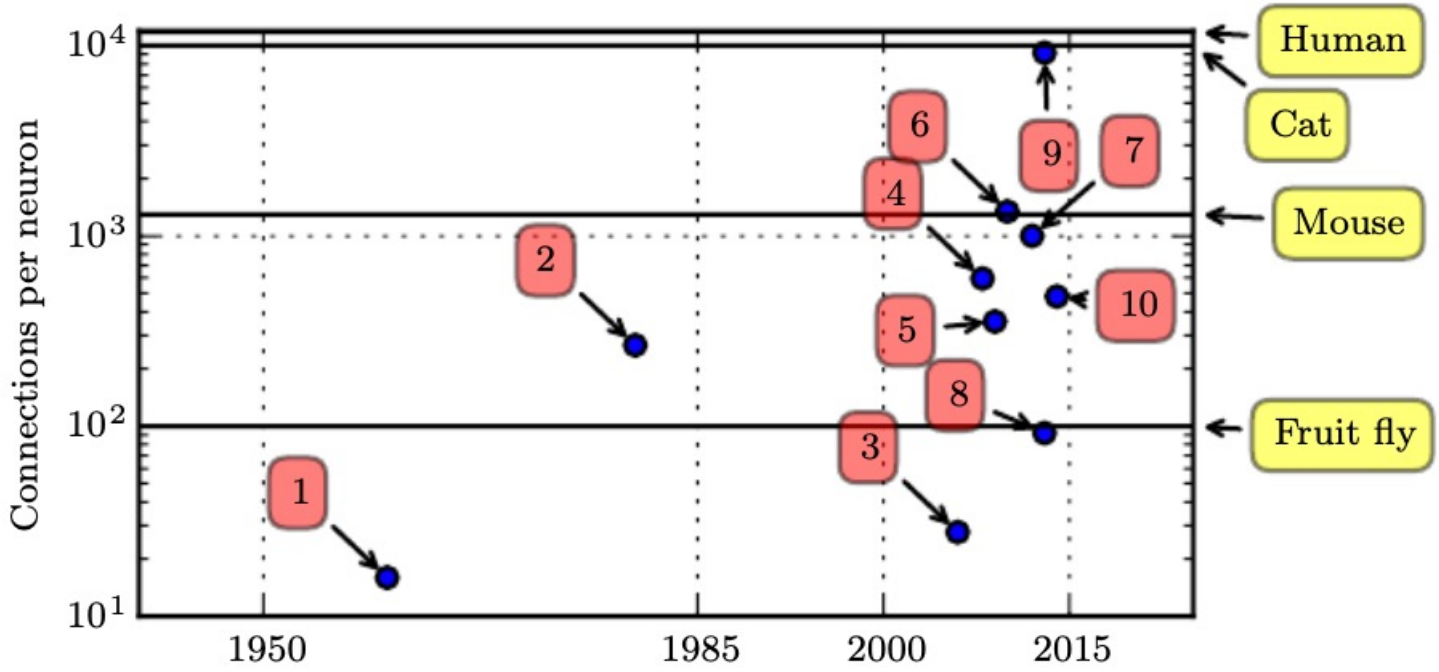
Increase of the neural network size over time

Doubling every ~ 2.4 years

(Figure from Goodfellow, Bengio and Courville `Deep Learning' MIT Press)

Sizes of biological NNs from Wikipedia (2015)

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)



Increase of the number of connections per neuron

(Figure from Goodfellow, Bengio and Courville `Deep Learning' MIT Press)

1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

Examples of Tasks (Classic ML Examples)

- Classification. The task is to specify to which of k different categories a given input $\mathbf{x}$ belongs to. In this case: $f : \mathbb{R}^n \to \{1, \cdots, k\}$.

  Examples: cat or dog? $(k = 2)$, tagging pictures with faces (face recognition), handwritten letter recognition, object recognition etc. Various variants of classification can be considered (e.g. classification with missing inputs).

- Regression. The task is to predict a numerical value of the output $\mathbf{y}$ given some input $\mathbf{x}$. The output can be just a real number: $f : \mathbb{R}^n \to \mathbb{R}$ - More generally could also be a vector: $f : \mathbb{R}^n \to \mathbb{R}^m$ (sometimes referred to as 'structured output').

  Example: speech recognition. The input is an audio recording (waveform) and the output is a sequence of characters or word codes.

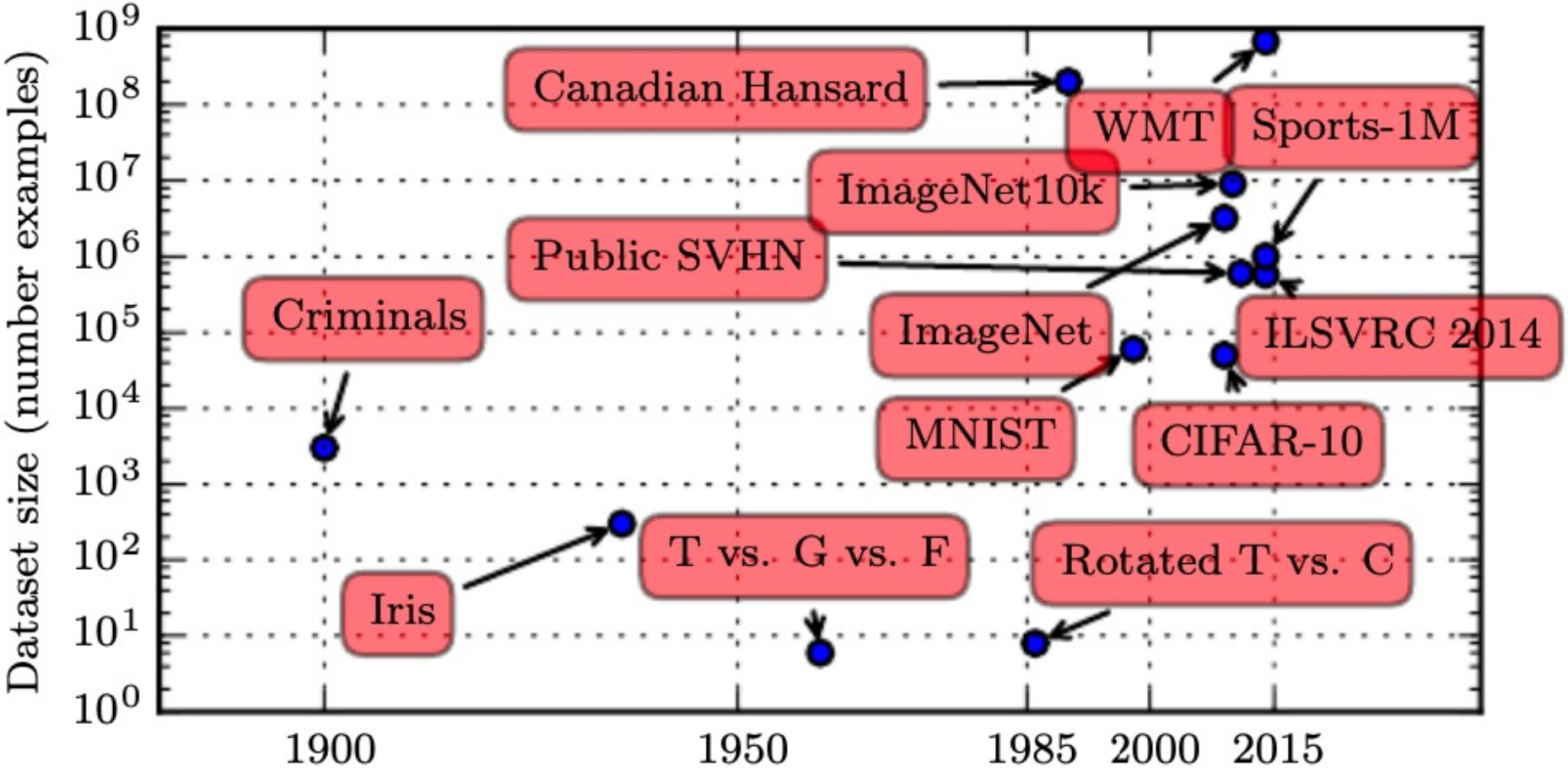
- Translation. Input is a sentence in one language, output is a sentence in another one.

- Probability Density Estimation (or density estimation, in short). The task is to learn a probability density $p(\mathbf{x})$, $p : \mathbb{R}^n \to \mathbb{R}$ ... or a complex valued probability amplitude such as a wave function: $\Psi : \mathbb{R}^n \to \mathbb{C}$ or more generally $\Psi : H^n \to \mathbb{C}$ where H is a local Hilbert space (for example $H = \{0, 1\}$ for a qubit or spin-1/2), n being the number of qubits/spins.

- In the context of density estimation, supervised learning means, roughly speaking, reconstructing $p(\mathbf{y}|\mathbf{x})$, while unsupervised learning means learning $p(\mathbf{y}, \mathbf{x})$. The Bayesian relation: $p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})\, p(\mathbf{x})$ relates unsupervised and supervised learning and hence allows for mixed strategies, blurring the distinction between the two. The decomposition of a probability measure into a chain of conditional probabilities is key to recurrent neural networks (RNNs), for example.

Training (The experience E)

- Supervised Learning. A data set (y_i, x_i) is provided to train the model (training set)
- Unsupervised learning. No `labeled/tagged' data set is provided. The model must identify patterns by itself.
- Reinforcement learning. No tagged data set, but a performance indicator is provided along the optimization process.
- Frontier between these different strategies somewhat blurred.



Increase of dataset sizes over time

(Figure from Goodfellow, Bengio and Courville `Deep Learning' MIT Press)

Why Use Machine Learning in the Context of Quantum Physics ?

- ML is applied linear algebra in a large vector space
- Quantum physics is also formulated as linear algebra in a large vector space of functions (Hilbert space)
- Both ML and QM use probabilistic/statistical concepts and methods
- Hence, at a general level, it is natural to use ML for QM problems involving a large number of degrees of freedom

Tasks in the Context of Quantum Physics (appetizer! – more at the end of the lecture)

- Phase recognition [C]
- Quantum State Tomography [DE]*
- Hamiltonian Certification [DE]
- Control of Quantum Systems [RL]
- Neural Quantum States: Parametrization of Variational Wave Function [R, self-generative unsupervised]*
- Density Functionals from ML [R, supervised]*
- Molecular Dynamics: Force Fields from ML [R, supervised]*
- Applications of ML to Materials Informatics/Databases

C=Classification; DE=(Probability) Density Estimation; RL=Reinforcement Learning; R=Regression.
(*): Topic of one of the lectures or seminars

Performance Metrics

- To be evaluated on the test set.
- Aim: Evaluate generalization performance
- Not to be confused with the *loss function* optimized during training. If differentiable, a performance metric can also be used as the loss function.
- Below, y_i is the predicted value, a_i the `ground truth' (exact) value.
- Examples of performance metrics:
- 0-1 Loss error rate (in classification):

$$\frac{1}{N}\sum_{j=1}^{N}(1-\delta_{y_j,a_j})$$

Warning: non differentiable!

- Mean Square Error (MSE)

$$\mathrm{MSE} = \frac{1}{N}\sum_{j=1}^{N}(a_j - y_j)^2 \equiv (\mathrm{RMSE})^2$$

- *Widely used and quite natural. The MSE has the reputation to be more sensitive to outliers (because of the squaring)*

- Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Percentage Error (MPE):

$$\mathrm{MAE} = \frac{1}{N}\sum_{j=1}^{N}|a_j - y_j| \ ,$$

$$\mathrm{MAPE} = \frac{100\%}{N}\sum_{j=1}^{N}\left|\frac{a_j - y_j}{a_j}\right|$$

$$\mathrm{MPE} = \frac{100\%}{N}\sum_{j=1}^{N}\frac{a_j - y_j}{a_j}$$

Density Estimation: Kullback-Leibler Divergence (or `Relative Entropy')

Set X of samples/events/messages/measurements x_i drawn according to P(x)

Quantity of information associated with each event, according to Shannon:

$$-\ln P(x_i)$$

Hence, average quantity of information contained in X – the Shannon entropy:

$$H = -\sum_{x \in X} P(x) \ln P(x)$$

We do not have access to P (the data generator), only to some data generated with P.

If we have an approximate determination Q(x) of P, the average amount of Information that we will estimate for X will be: $-\sum_{x \in X} P(x) \ln Q(x)$

The difference between this estimated amount and the actual amount of information is the *Kullback-Leibler divergence:*

$$-\sum_{x \in X} P(x) \left[\ln Q(x) - \ln P(x)\right] = \sum_{x \in X} P(x) \ln \frac{P(x)}{Q(x)} \equiv D_{\mathrm{KL}}(P||Q)$$

Extra amount of info to send message drawn according to P using a transmission channel optimized for Q

Properties of the Kullback-Leibler Divergence:

$$D_{\mathrm{KL}}(P||Q) = \sum_{x\in X} P(x)\ln\frac{P(x)}{Q(x)} \quad , \quad D_{\mathrm{KL}}(P||Q) = \int dx\, p(x)\ln\frac{p(x)}{q(x)}$$

$$\boxed{D_{\mathrm{KL}}(P||Q) \geq 0 \quad , \quad D_{\mathrm{KL}}(P||Q) = 0 \Leftrightarrow \; P = Q}$$

But the KL divergence it is *not* a distance
(not symmetric, triangle inequality not satisfied)

Note: in minimizing D_{KL} over a trial Q, only the second term -P lnQ matters

$$-D_{KL} = \sum_x P\ln\frac{Q}{P} \leq \sum_x P[\frac{Q}{P} - 1] = \sum_x [Q(x) - P(x)] = 0$$

For the last equivalence to hold: X = {all values that the random variable x can take}

Proof of D_{KL}=0 → P=Q. If g(x) strictly convex (i.e. above its tangent for all values of x) then we have the strict inequality: <g(x)> > g(<x>) [Jensen's inequality]

Hence if P and Q are different (except on a zero measure set) D_{KL} is strictly positive:

$$D_{KL} = \langle -\ln\frac{Q}{P}\rangle_P > -\ln\langle\frac{Q}{P}\rangle_P = 0$$

AG

Prove that the Kullback-Leibler divergence is zero if and only if the two probability distributions are identical

The Kullback-Leibler (KL) divergence is a measure of how different two probability distributions are from each other. The KL divergence between two probability distributions P and Q is defined as:

$KL(P || Q) = \sum_x P(x) \log(P(x) / Q(x))$

where x is the outcome of a random variable, P(x) is the probability of x under distribution P, and Q(x) is the probability of x under distribution Q.

To prove that the KL divergence is zero if and only if the two probability distributions are identical, we need to show two things:

1. If the KL divergence is zero, then the two probability distributions are identical.
2. If the two probability distributions are identical, then the KL divergence is zero.

This is correct, and the logic is fine but…

ChatGPT did not pass the exam… ☹

Proof of (1):

Assume that the KL divergence between P and Q is zero, i.e., $KL(P || Q) = 0$. Then, we have:

$\sum_x P(x) \log(P(x) / Q(x)) = 0$

Since the logarithm function is non-negative, we must have $P(x) = Q(x)$ for all x where $P(x) > 0$. Otherwise, if $P(x) > 0$ and $Q(x) = 0$, the logarithm term is undefined. Therefore, if $KL(P || Q) = 0$, then $P(x) = Q(x)$ for all x where $P(x) > 0$. Since the probability distributions must sum to 1, this implies that P and Q are identical.

Errare ~~humanum~~ mechanicum est… perseverare ~~diabolicum~~ mechanicum ☺

AG this proof is incorrect: the logarithm function can be negative

You are correct. Thank you for pointing that out. The logarithm function can be negative for values less than 1, so the proof needs to be revised. Here is a corrected proof:

Proof of (1):

Assume that the KL divergence between P and Q is zero, i.e., KL(P || Q) = 0. Then, we have:

$\sum x\ P(x) \log(P(x) / Q(x)) = 0$

Since the logarithm function is negative for values of P(x) / Q(x) less than 1, we can conclude that P(x) / Q(x) = 1 for all x where P(x) > 0. Otherwise, if P(x) > 0 and Q(x) = 0, the logarithm term is undefined. This implies that P(x) = Q(x) for all x where P(x) > 0. Since the probability distributions must sum to 1, this implies that P and Q are identical.

A Key Learning Algorithm: Minimizing a Loss Function by (Stochastic) Gradient Descent (SGD)

Simple Gradient Descent:

Define Loss Function (e.g. MSE, KL divergence etc.) - additive over data in the training set:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_j), y_j\right)$$

Taylor expand around a point in parameter space:

$$\mathcal{L}(\boldsymbol{\theta}+\delta\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \delta\boldsymbol{\theta}^{\intercal}\cdot\nabla_{\boldsymbol{\theta}}\mathcal{L} + \frac{1}{2}\delta\boldsymbol{\theta}^{\intercal}\cdot\mathbf{H}\cdot\delta\boldsymbol{\theta} + \cdots$$

$$= \mathcal{L}(\boldsymbol{\theta}) + \langle\delta\boldsymbol{\theta}|\nabla_{\boldsymbol{\theta}}\mathcal{L}\rangle + \frac{1}{2}\langle\delta\boldsymbol{\theta}|\mathbf{H}|\delta\boldsymbol{\theta}\rangle + \cdots$$

with:

$$[\nabla_{\theta}\mathcal{L}]_i \equiv \frac{\partial\mathcal{L}}{\partial\theta_i}\,(\text{Gradient}) \quad , \quad H_{ij} \equiv \frac{\partial^2\mathcal{L}}{\partial\theta_i\partial\theta_j}\,(\text{Hessian})$$

Make a (small) move in parameter space:

$$\delta\boldsymbol{\theta} = -\eta \nabla_{\boldsymbol{\theta}} \mathcal{L} \ , \ \ \boldsymbol{\theta} \to \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}$$

$$\mathcal{L}(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) - \eta \, ||\nabla_{\boldsymbol{\theta}} \mathcal{L}||^2 + \frac{\eta^2}{2} \langle \nabla_{\boldsymbol{\theta}} \mathcal{L} | \mathbf{H} | \nabla_{\boldsymbol{\theta}} \mathcal{L} \rangle + \cdots$$

< 0

Eta is the *learning rate.*
It should be chosen small enough,
so that the quadratic term is negligible
and one effectively lowers the loss function
- but not *too* small to avoid slow convergence
and wasting computing time…

Gradient Descent can (and does) of course get stuck in local minima etc.
Improvements on the simple GD method are available and commonly used,
such as the ADAM optimizer for example (Adaptative Moment Estimation)
- see Kingma and Ba arXiv:1412.6980 (2014) and textbooks/reviews

Computation of the Gradient in practice: Automatic Differentiation, Back Propagation (based on the chain rule of differentiation) – not described here, see references

A Key Learning Algorithm: Minimizing a Loss Function by (Stochastic) Gradient Descent (SGD)

Gradient Descent:

[4]Pour trouver les valeurs de $x, y, z, \ldots$, qui vérifieront l'équation $u = 0$, il suffira de faire décroître indéfiniment la fonction u, jusqu'à ce qu'elle s'évanouisse.

A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *C. R. Acad. Sci. Paris*, 25:536–538, 1847.

Augustin Louis Cauchy
1789 - 1857

`Cauchy and the gradient method'
https://www.math.uni-bielefeld.de/documenta/vol-ismp/40_lemarechal-claude.pdf

Choosing the Learning Rate

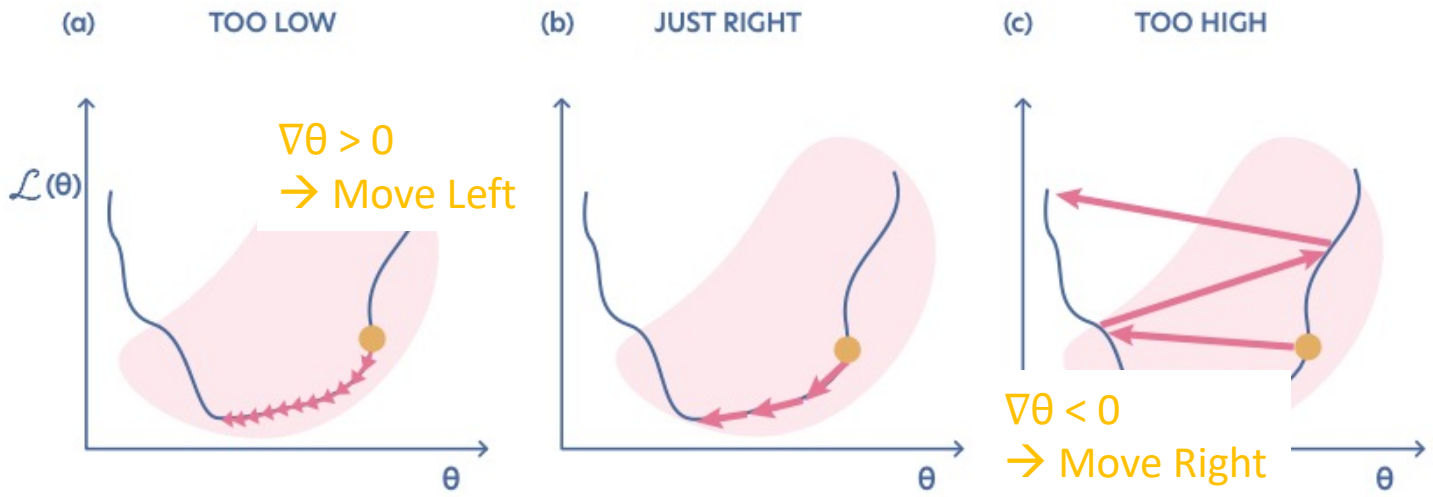


Figure 2.2: Choosing a learning rate has an impact on convergence to the minimum. (a) If η is too small, the training needs many epochs. (b) The right η allows for a fast convergence to a minimum and needs to be found. (c) If η is too large, optimization can take you away from the minimum (you "overshoot"). This figure suggests that loss function is convex which is rarely true.

Image Credit: A.Dawid et al. arXiv:2204.04198

Stochastic Gradient Descent (SGD)

Recall that the loss function is a sum over the training data set: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_j), y_j\right)$

Hence, computing the gradient can be very costly when this set is large
Instead, we can compute this stochastically over a randomly chosen subset (`mini-batch') of the training set. This also helps not getting stuck in local minima.

Mini-batch contains m_b data points,
σ denotes a permutation of 1…m:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} \simeq \frac{1}{m_b}\sum_{j=1}^{m_b} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_{\sigma(j)}), y_{\sigma(j)}\right)$$

Algorithm 1 Minibatch stochastic gradient descent (SGD)

Require: Learning rate η
  Initialize $\boldsymbol{\theta}$ to random values
  for epoch = 1 to no_epochs **do**
    Shuffle $\mathcal{D}_{\text{train}}$
    for i = 1 to m (where m is a minibatch size) **do**
      $\boldsymbol{x}_i, y_i \sim \mathcal{D}_{\text{train}}$ ▷ Draw random data point from data set without replacement
      $\mathcal{L} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(y_i, f(\boldsymbol{x}_i))$ ▷ Compute loss function on the minibatch
      $(\nabla\mathcal{L})_j \leftarrow \frac{\partial \mathcal{L}}{\partial \theta_j}$ ▷ Compute gradients
      $\theta_j \leftarrow \theta_j - \eta \frac{\partial}{\partial \theta_j}\mathcal{L}$ ▷ Update parameters
    end for
  end for
  return $\boldsymbol{\theta}$

Pseudocode credit:
A.Dawid et al.
arXiv:2204.04198

Natural Gradient and the Metric in Parameter Space

ARTICLE Communicated by Steven Nowlan and Erkki Oja

Neural Computation 10, 251–276 (1998)

Natural Gradient Works Efficiently in Learning

Shun-ichi Amari
RIKEN Frontier Research Program, Saitama 351-01, Japan

When a parameter space has a certain underlying structure, the ordinary gradient of a function does not represent its steepest direction, but the natural gradient does. Information geometry is used for calculating the natural gradients in the parameter space of perceptrons, the space of matrices (for blind source separation), and the space of linear dynamical systems (for blind source deconvolution). The dynamical behavior of natural gradient online learning is analyzed and is proved to be Fisher efficient, implying that it has asymptotically the same performance as the optimal batch estimation of parameters. This suggests that the plateau phenomenon, which appears in the backpropagation learning algorithm of multilayer perceptrons, might disappear or might not be so serious when the natural gradient is used. An adaptive method of updating the learning rate is proposed and analyzed.

Also related to
`Stochastic Reconfiguration'
in Variational MC

VOLUME 80, NUMBER 20 PHYSICAL REVIEW LETTERS 18 MAY 1998

Green Function Monte Carlo with Stochastic Reconfiguration

Sandro Sorella
Istituto Nazionale di Fisica della Materia and International School for Advanced Studies, Via Beirut 4, 34013 Trieste, Italy
(Received 17 November 1997)

A new method for the stabilization of the sign problem in the Green function Monte Carlo technique is proposed. The method is devised for real lattice Hamiltonians and is based on an iterative "stochastic reconfiguration" scheme which introduces some bias but allows a stable simulation with constant sign. The systematic reduction of this bias is possible in principle. The method is applied to the frustrated $J_1 - J_2$ Heisenberg model, and tested against exact diagonalization data. Evidence of a finite spin gap for $J_2/J_1 > \sim 0.4$ is found in the thermodynamic limit. [S0031-9007(98)06070-0]

Idea: The Euclidean distance in parameter space may not be the `natural' one. Derive a learning/descent rule for a general distance

Distance specified by a metric tensor g:

$$d(\boldsymbol{\theta}, \boldsymbol{\theta} + \delta\boldsymbol{\theta})^2 = \sum_{lm} g_{lm}(\boldsymbol{\theta}) \delta\theta_l \delta\theta_m$$

Minimize the loss function under the constraint that one travels a distance d=$\boldsymbol{\epsilon}$ and hence introduce a Lagrange multiplier μ and (keeping a first-order estimator for the loss function) minimize Lagrangian:

$$\sum_l \frac{\partial \mathcal{L}}{\partial \theta_l} \delta\theta_l + \mu \left[\sum_{lm} g_{lm}(\boldsymbol{\theta}) \delta\theta_l \delta\theta_m - \epsilon^2 \right]$$

Leads to:

$$\delta\boldsymbol{\theta} = -\frac{1}{2\mu} \mathbf{g}^{-1} \nabla_{\boldsymbol{\theta}} \mathcal{L}$$

Note: the inversion of g may require a regularization

Remarks: Adapts learning procedure to local metric

- Euclidean distance g=1 → recover simple steepest descent
- Change of loss function can be shown to be invariant under change of coordinates in parameter space (reparametrizations)

Additional remarks on NGD – Adaptative Methods

The Lagrange multiplier is obtained from the constraint $d^2=\epsilon^2$:

$$\epsilon^2 = \frac{1}{4\mu^2} \sum_{lm} \frac{\partial \mathcal{L}}{\partial \theta_l} [g^{-1}]_{lm} \frac{\partial \mathcal{L}}{\partial \theta_m} = \frac{1}{4\mu^2} \langle \nabla_\theta \mathcal{L} | g^{-1} | \nabla_\theta \mathcal{L} \rangle$$

We can keep the distance constant by adapting the learning rate at each step:

$$\delta \boldsymbol{\theta} = -\epsilon \frac{g^{-1} \nabla_\theta \mathcal{L}}{\sqrt{\langle \nabla_\theta \mathcal{L} | g^{-1} | \nabla_\theta \mathcal{L} \rangle}} \quad , \quad \delta \mathcal{L} = -\epsilon \sqrt{\langle \nabla_\theta \mathcal{L} | g^{-1} | \nabla_\theta \mathcal{L} \rangle}$$

This can be extended to second-order schemes – damped Newton-Raphson (then involves the Hessian in the learning rate)

State of the art adaptative methods:
e.g. ADADELTA , ADAM etc.
see M. Zeiler arXiv:1212.5701
(involves gradients from previous steps)

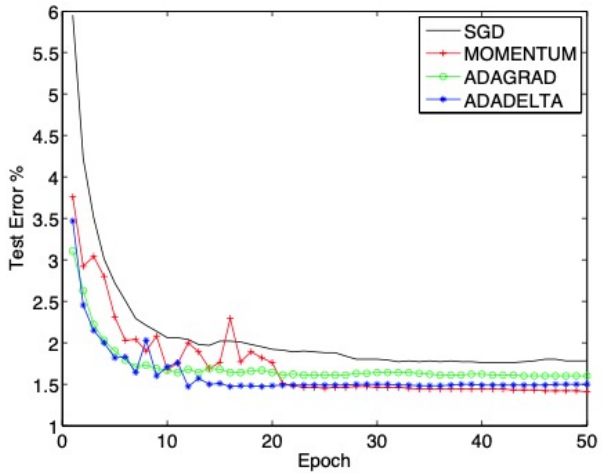


Fig. 1. Comparison of learning rate methods on MNIST digit classification for 50 epochs.

NGD for (Probability) Density Estimation

Choose the KL divergence as a measure of distance:

$$d^2(\theta, \theta') = D_{KL}(P_\theta || P_{\theta'}) = \sum_{x \in X} P_\theta \left[\ln P_\theta - \ln P_{\theta'} \right]$$

A simple calculation leads to:

$$d^2(\theta, \theta + \delta\theta) = \sum_{lm} g_{lm} \delta\theta_l \delta\theta_m$$

Fisher information matrix/metric

$$g_{lm} = \sum_{x \in X} P_\theta(x) \frac{\partial \ln P_\theta(x)}{\partial \theta_l} \frac{\partial \ln P_\theta(x)}{\partial \theta_m}$$

Fisher information: how much information about a given parameter θ we can get from a sample X?

Variance of the derivative of the log-likelihood:

$$\mathrm{FI}[\theta] = \mathrm{Var}_X \left[\frac{\partial}{\partial \theta} \ln P_\theta(x) \right] = \sum_{x \in X} P_\theta(x) \left(\frac{\partial \ln P_\theta(x)}{\partial \theta} \right)^2$$

Computation of Gradients in Practice

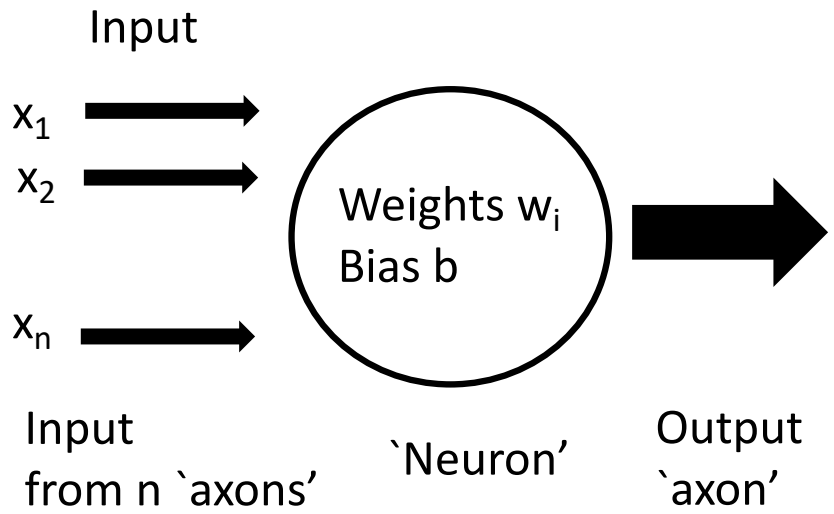
- Automatic Differentiation
- Backpropagation (based on recursive application of the chain rule for derivatives)
- *Not covered in detail in these lectures – see references and code docs.*

Neural Networks: Parametrizing $f(\mathbf{x};\theta)$

- Different NN architectures correspond to different ways of constructing and parametrizing $f(x,\theta)$.
- During the course of these lectures, we will encounter several different architectures:
- Multilayer Perceptrons
- Restricted Boltzmann Machines (RBM)
- Convolutional Neural Networks (CNN)
- Autoregressive Recurrent NNs (RNN)
- And more…

Archetypal Feed-Forward NN: The Multilayer Perceptron

- Basic unit: an `artificial neuron' which:
- Aggregates inputs by weighting the different components x_i
- Adds a bias b
- And returns an output signal (or activation) by applying to the result a non-linear function



Output (activation)

$$a = s\left(\sum_i w_i x_i + b\right)$$

Typical choices for the function s:

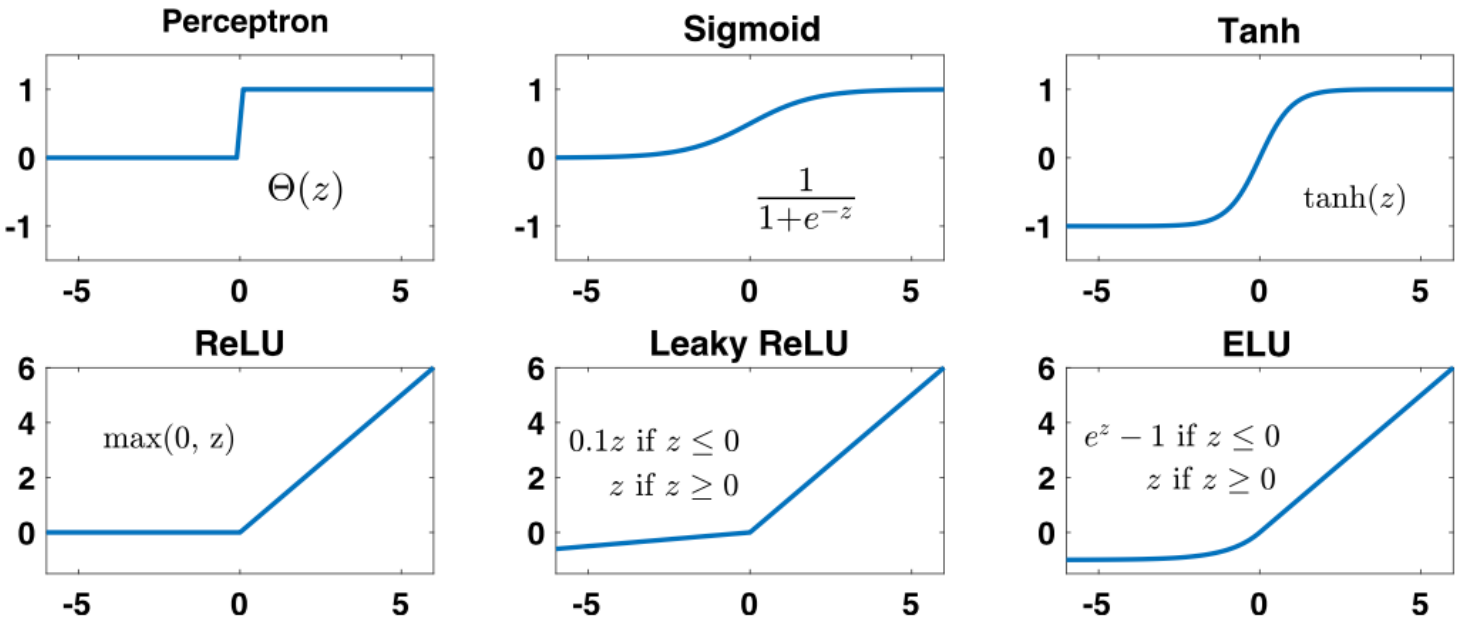
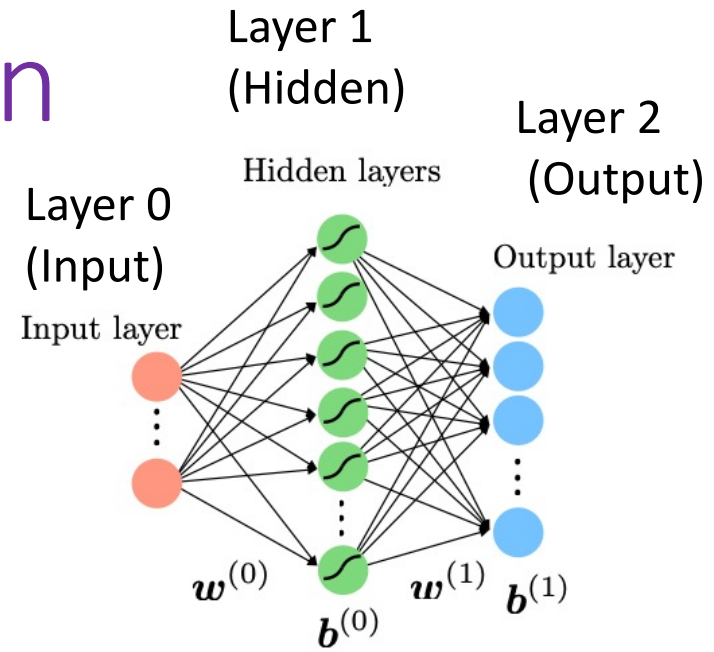


Fig. 36. Possible non-linear activation functions for neurons. In modern DNNs, it has become common to use non-linear functions that do not saturate for large inputs (bottom row) rather than saturating functions (top row).

Image Credit: Mehta et al. Physics Reports 810 1 (2019)

Multilayer Perceptron

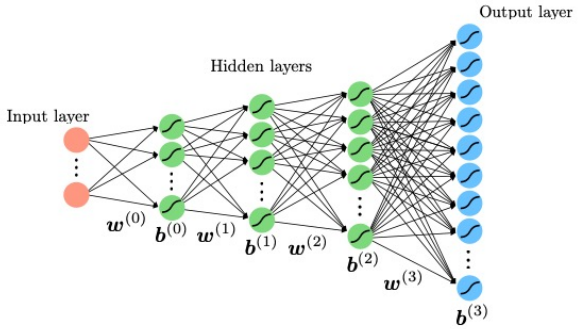
Example with:
1 input layer, 1 output layer
1 hidden layer



$$y_{i_2} = \sum_{i_1 \in 1} a_{i_1} w^{(1)}_{i_1 i_2} + b^{(1)}_{i_2}$$

$$= \sum_{i_1 \in 1} s \left(\sum_{i_0 \in 0} x_{i_0} w^{(0)}_{i_0 i_1} + b^{(0)}_{i_1} \right) w^{(1)}_{i_1 i_2} + b_{i_2}$$

Can continue this recursively by composing successive units, e.g with 2 hidden units:



$$s \left(\sum_{i_1 \in 1} s \left(\sum_{i_0 \in 0} x_{i_0} w^{(0)}_{i_0 i_1} + b^{(0)}_{i_1} \right) w^{(1)}_{i_1 i_2} + b_{i_2} \right) w^{(2)}_{i_2 i_3} + b^{(2)}_{i_3}$$

Welcome to the world of Deep Learning !

Image Credit: Javier Robledo-Moreno

NNs as Universal Approximators

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

Math. Control Signals Systems 2, 303 (1989)

See also more recent:
Lu et al.
arXiv:1709.02540

See Stéphane Mallat's Collège de France Lectures (videos on website) especially the February 20, 2019 Lecture.

Neural Networks 4, 251 (1991)

Approximation Capabilities of Multilayer Feedforward Networks

KURT HORNIK

Technische Universität Wien, Vienna, Austria

(*Received* 30 *January* 1990; *revised and accepted* 25 *October* 1990)

Abstract—*We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures μ, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.*

Keywords—Multilayer feedforward networks, Activation function, Universal approximation capabilities, Input environment measure, $L^p(\mu)$ approximation, Uniform approximation, Sobolev spaces, Smooth approximation.

Theorem: (Cybenko, 1989)

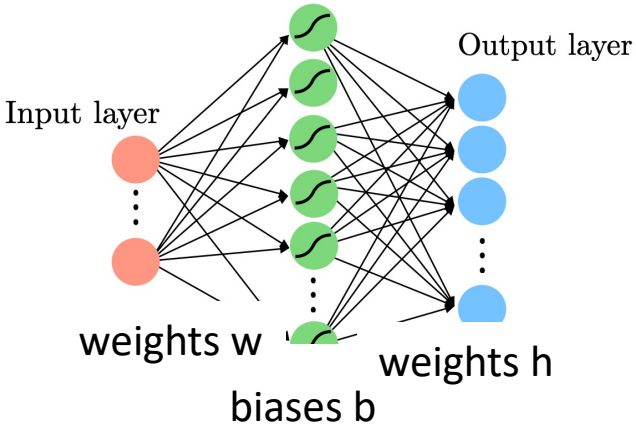
Arbitrary sigmoidal function s(x) – such as $1/(1+e^{-x})$ but need not be monotonous

$$s(x \to -\infty) \to 0 \ , \ \ s(x \to +\infty) \to 1$$

$$\mathbf{f}(\mathbf{x}) : [0,1]^{D_i} \to \mathbb{R}^{D_o}$$ Arbitrary function to be approximated

Single hidden layer:

$$j \in \mathrm{Out}\,;\ N(\mathbf{x})_j = \sum_{k \in H} s\left(\sum_{i \in \mathrm{In}} x_i w_{ik} + b_k\right) h_k$$

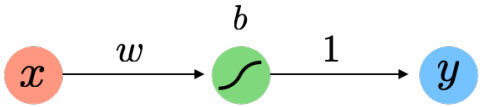


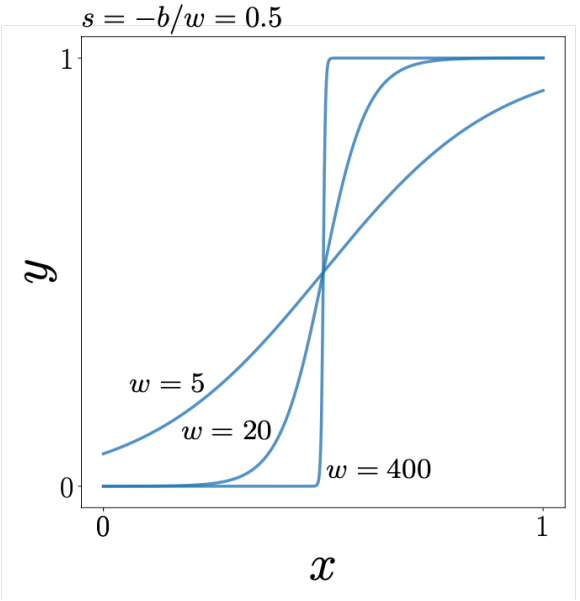
For all ε > 0, there is a 1-hidden layer NN approximant N(x) such that:

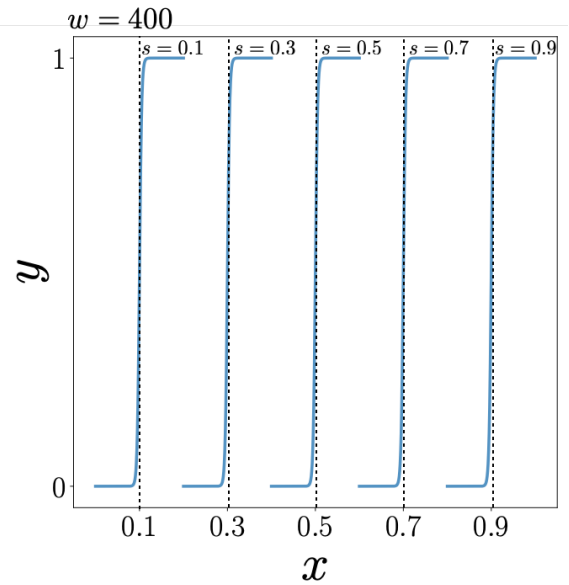
$$|N(\mathrm{x}) - f(\mathrm{x})| < \varepsilon \ , \ \forall \mathrm{x} \in [0,1]^{D_i}$$

Intuitive view of universal approximation theorem

See M.Nielsen's online book Chap 4
and Javier Robledo-Moreno (PhD thesis, in preparation) –
- gratefully acknowledged for discussions and pictures

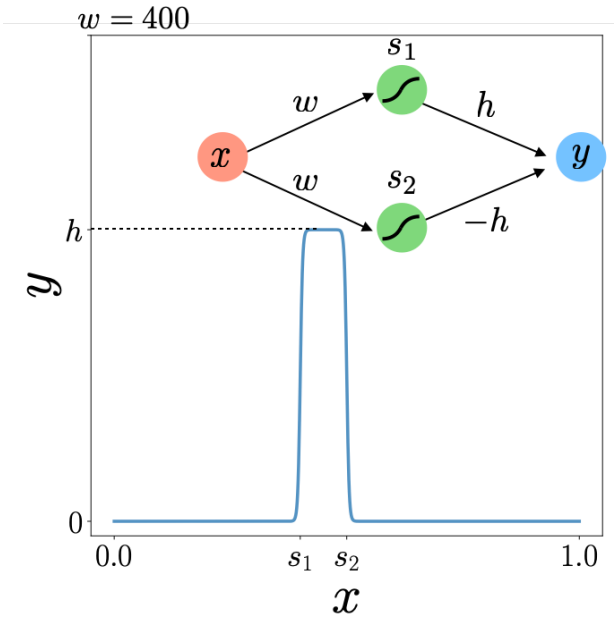






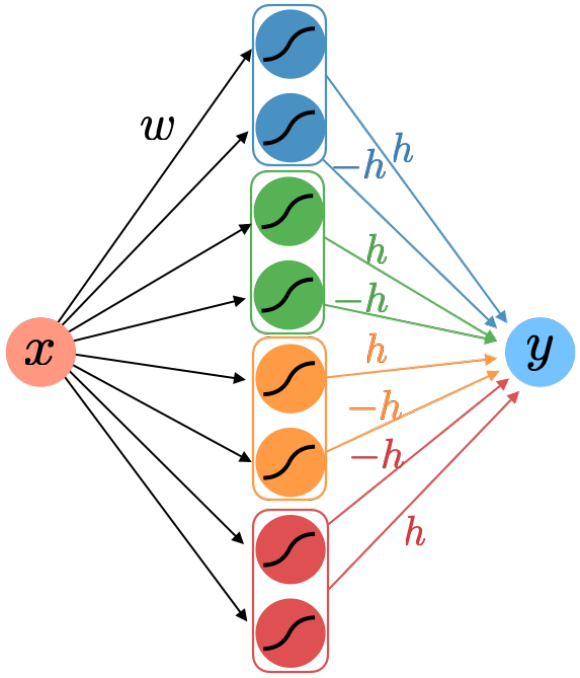
One unit → step behavior: w controls width, s=-b/w controls location of step
Here the sigmoid $1/(1+e^{-x})$ is used

Image Credit: Javier Robledo-Moreno



2 Neurons with opposite output weights
→ `Crenel' function

$$\approx h[sign(x + b_1/w) - sign(x + b_2/w)]$$



Pairwise
(h,-h)
Weights →

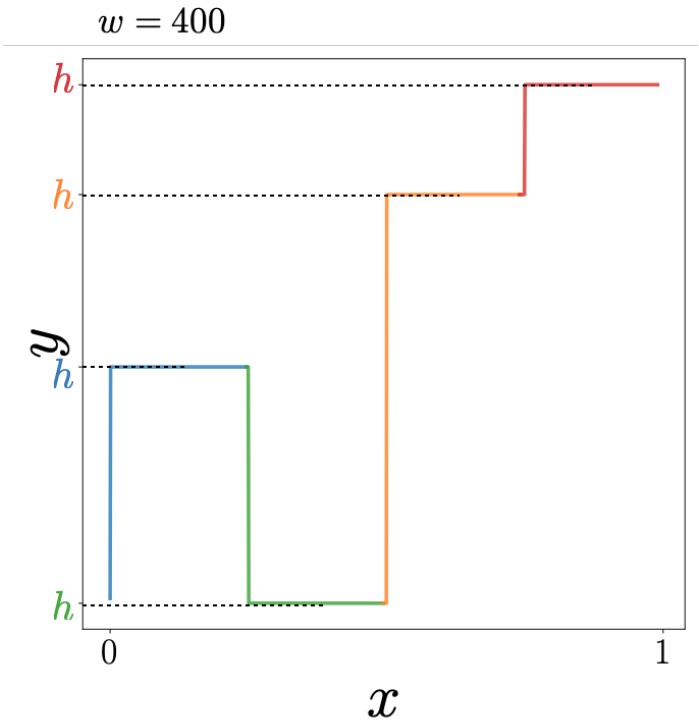
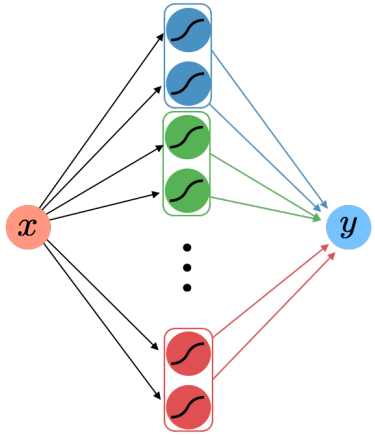
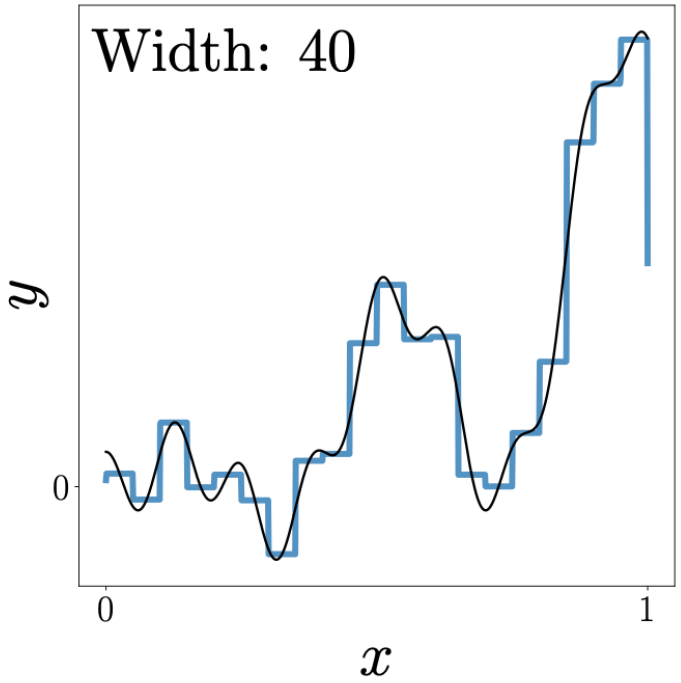


Image Credit: Javier Robledo-Moreno

Better approximants for increasing hidden layer width:





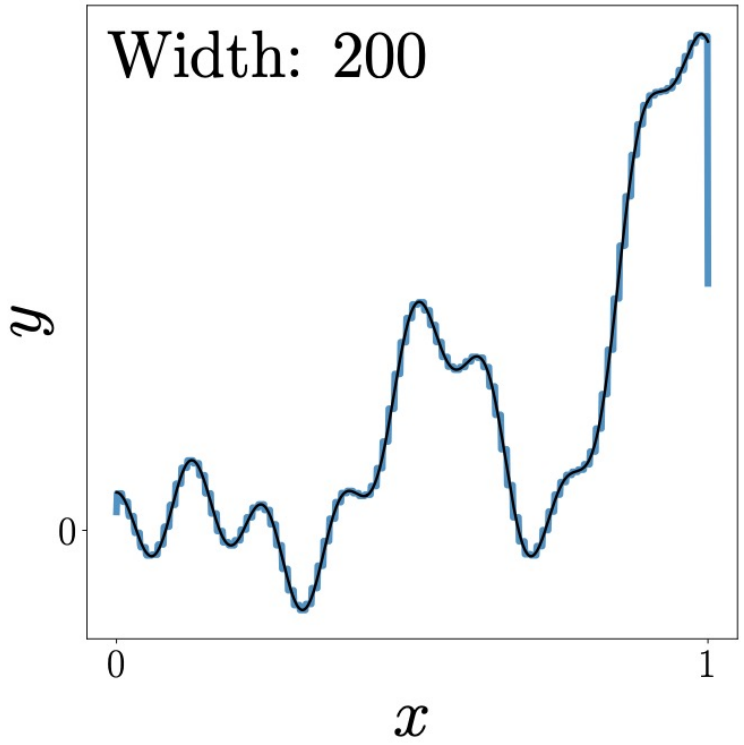


Image Credit: Javier Robledo-Moreno

The approximation theorem says nothing about how many parameters we need to reach a given precision!

Cybenko, 1989

While the approximating properties we have described are quite powerful, we have focused only on existence. The important questions that remain to be answered deal with feasibility, namely how many terms in the summation (or equivalently, how many neural nodes) are required to yield an approximation of a given quality? What properties of the function being approximated play a role in determining the number of terms? At this point, we can only say that we suspect quite strongly that the overwhelming majority of approximation problems will require astronomical numbers of terms. This feeling is based on the *curse of dimensionality* that plagues multidimensional approximation theory and statistics. Some recent progress concerned with the relationship between a function being approximated and the number of terms needed for a suitable approximation can be found in [MSJ] and [BH], [BEHW], and [V] for related problems. Given the conciseness of the results of this paper, we believe that these avenues of research deserve more attention.

cf. Ergodicity in Statistical Mechanics: good to know when it applies,
But its not the fundamental reason why the Boltzmann/Gibbs equilibrium ensemble works!

For recent progress and conjectures on this, see Lu et al. arXiv:1709.02540

Underfitting and Overfitting: The Bias-Variance Trade-Off

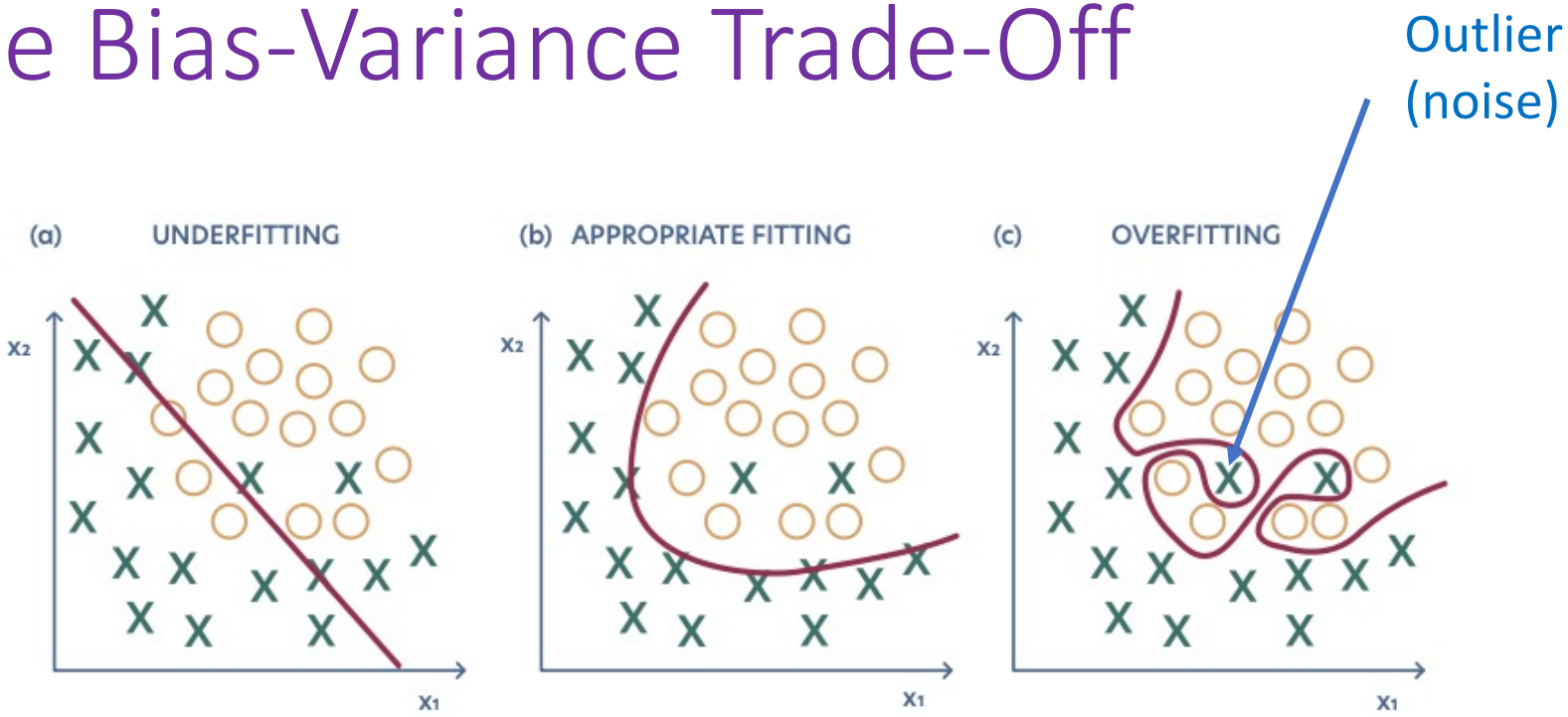


Figure 2.3: Scheme of under- and overfitting. (a) When the model capacity is too low, the model cannot fit the training data properly. (b) With the model capacity corresponding to the task complexity, the fitting is optimal. (c) When the model capacity exceeds the task complexity, the model tends to overfit and the generalization error increases.

Image Credit: A.Dawid et al. arXiv:2204.04198

Underfitting and Overfitting: The Bias-Variance Trade-Off

Function to be approximated (ground-truth): f(x)
Approximant for a given set of parameters: $f_\theta(x)$

Parameters optimized over noisy training sets

$$D = \{(y_i, x_i)\} \ , \ y_i = f(x_i) + \varepsilon$$

In the following averages <…> are taken over both the training sets and the random noise.

Estimate the generalization error by averaging over many training sets

Prediction error for a given value of x:

$$\mathrm{Err}(x) \ = \ \langle [f(x) + \varepsilon - f_\theta(x)]^2 \rangle$$

Underfitting and Overfitting: The Bias-Variance Trade-Off

An easy calculation yields:

$$\mathrm{Err}(x) = \mathrm{Bias}^2 + \mathrm{Var} + \langle \varepsilon^2 \rangle$$

$$\mathrm{Bias} = \langle f_\theta(x) \rangle - f(x)$$

$$\mathrm{Var} = \left\langle \left(f_\theta(x) - \langle f_\theta(x) \rangle \right)^2 \right\rangle$$

Underfitting and Overfitting: The Bias-Variance Trade-Off

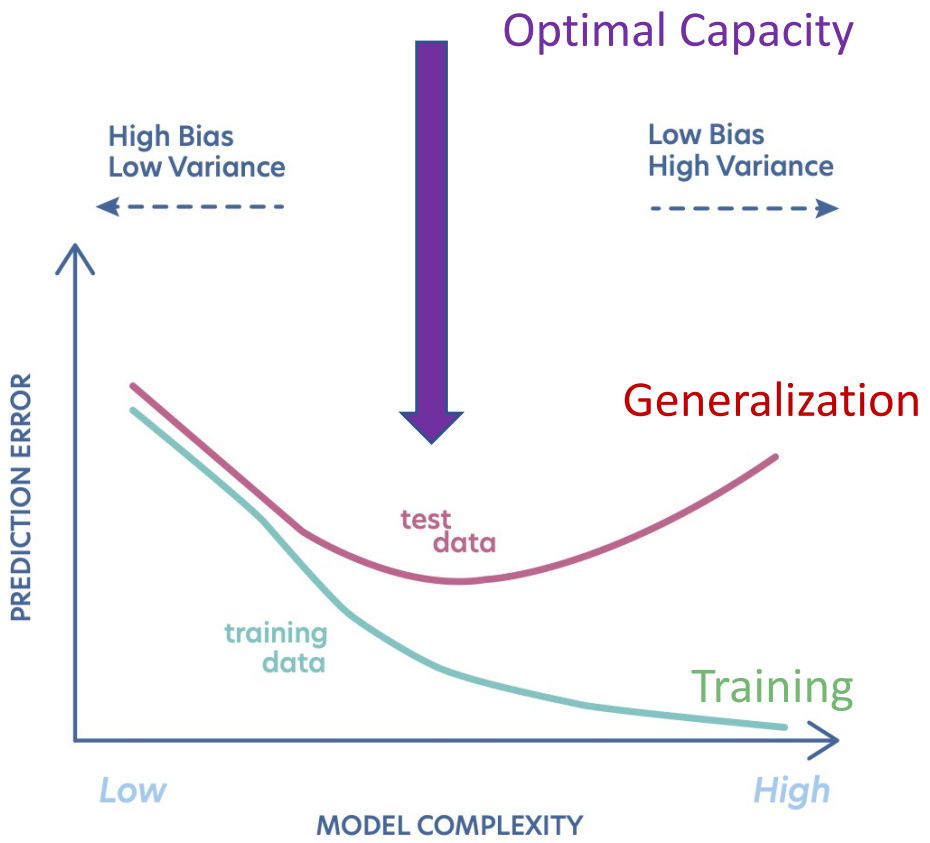


Figure 2.4: Illustration of the bias-variance trade-off and its relation to the prediction error observed on training (green curve) and test sets (red curve). The ideal model which results in the lowest test error has both intermediate model complexity (e.g. capacity) and training error. Adapted from Ref. [56].

The bias-variance trade-off: conventional view

Image Credit: A.Dawid et al. arXiv:2204.04198
Adapted from Hasti et al. (Springer)

Over-Parametrization Is Good! `Double Descent'

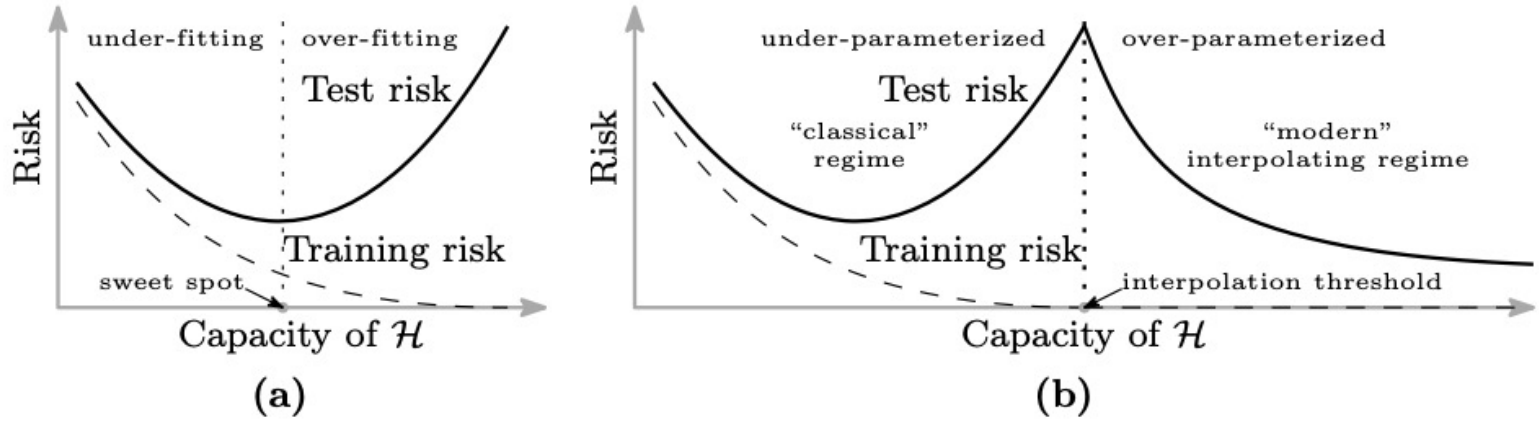


Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Belkin et al. arXiv:1812.111118, PNAS 116 (2019)
See comment by Loog et al. PNAS 117 (2020)

Our intuition is biased by polynomial fitting: in ML, higher capacity may lead to smoother interpolation – hence better generalization

Optimisation Landscape: Topology of the space of parameters?

Much yet to be understood…

- Many nearly-degenerate minima with zero or low training error
- Some have poor generalization properties (non-smooth interpolation in the space of data)
- Some have good generalization (smooth interpolations)
- Minima disconnected → Glassy
- Minima actually connected by valleys ?

Discussions on these points with Javier Robledo-Moreno gratefully acknowledged

Jamming transition as a paradigm to understand the loss landscape of deep neural networks

Mario Geiger,[1,*,†] Stefano Spigler,[1,*] Stéphane d'Ascoli,[2,3] Levent Sagun,[2,1] Marco Baity-Jesi,[4]
Giulio Biroli,[2,3] and Matthieu Wyart[1]

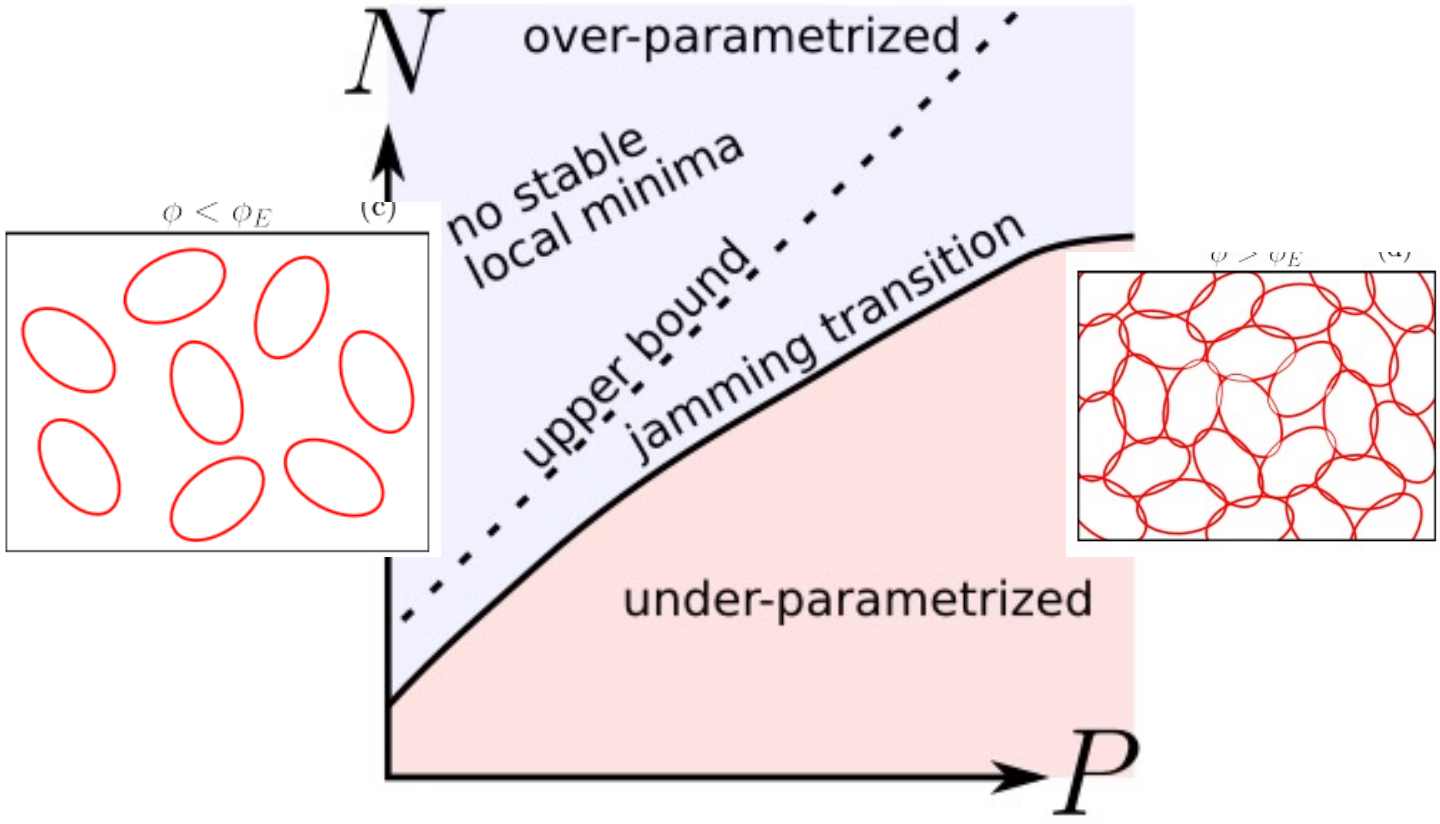
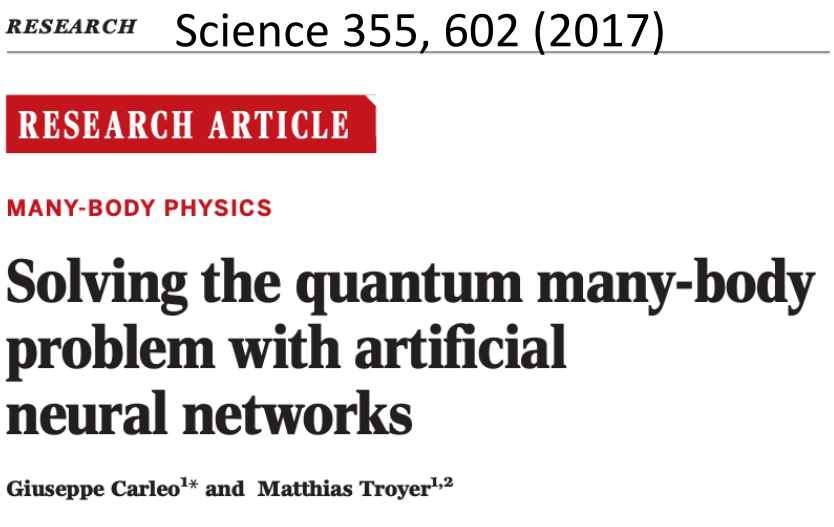


FIG. 1. N: degrees of freedom, P: training examples.

Applications of ML to Quantum Physics (Selected Examples – with some of the papers to be discussed during these lectures)

- Neural Quantum States:
- Parametrize the (ground-state) wave-function of a many-body quantum system $\psi(x_1,\ldots,x_N)$ by a NN and optimize the parameters to lower the energy
- Loss function: variational energy

→ Lectures 2 and 4
Seminars by F.Vicentini, G.Carleo, J.Carrasquilla

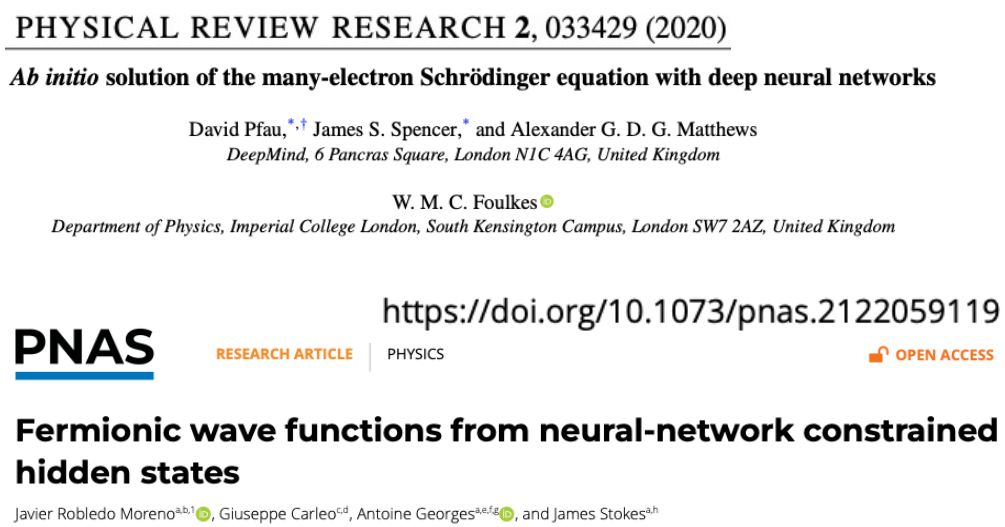

RESEARCH Science 355, 602 (2017)

RESEARCH ARTICLE

MANY-BODY PHYSICS

Solving the quantum many-body problem with artificial neural networks

Giuseppe Carleo[1]* and Matthias Troyer[1,2]


PHYSICAL REVIEW RESEARCH 2, 033429 (2020)

***Ab initio* solution of the many-electron Schrödinger equation with deep neural networks**

David Pfau,[*,†] James S. Spencer,[*] and Alexander G. D. G. Matthews
DeepMind, 6 Pancras Square, London N1C 4AG, United Kingdom

W. M. C. Foulkes
Department of Physics, Imperial College London, South Kensington Campus, London SW7 2AZ, United Kingdom

https://doi.org/10.1073/pnas.2122059119

PNAS RESEARCH ARTICLE | PHYSICS OPEN ACCESS

Fermionic wave functions from neural-network constrained hidden states

Javier Robledo Moreno[a,b,1], Giuseppe Carleo[c,d], Antoine Georges[a,e,f,g], and James Stokes[a,h]

Reconstructing The Wave-Function From a Sequence of (Projective) Measurements: `Quantum Tomography'

→ Lecture 3
and seminar by J.Carrasquilla

nature physics
LETTERS
https://doi.org/10.1038/s41567-018-0048-5

Neural-network quantum state tomography

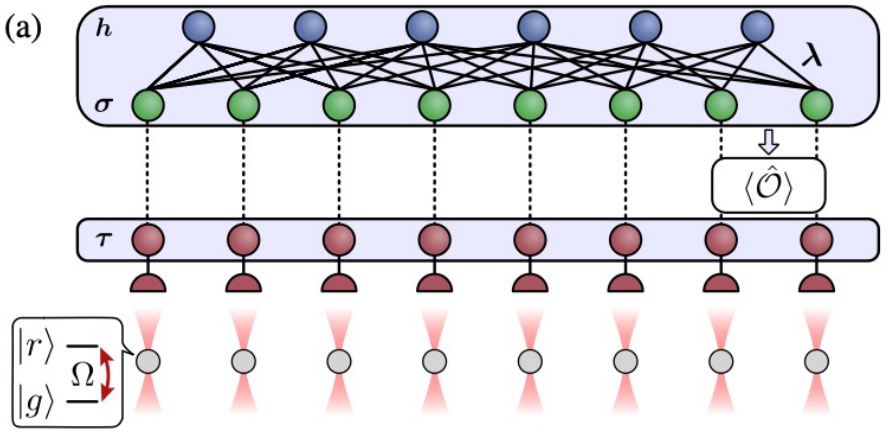
Giacomo Torlai[1,2], Guglielmo Mazzola[3], Juan Carrasquilla[4,5], Matthias Troyer[3,6], Roger Melko[1,2] and Giuseppe Carleo[3,7]*

PHYSICAL REVIEW LETTERS **123,** 230504 (2019)

Editors' Suggestion

Integrating Neural Networks with a Quantum Simulator for State Reconstruction

Giacomo Torlai,[1,2,3] Brian Timar,[4] Evert P. L. van Nieuwenburg,[4] Harry Levine,[5] Ahmed Omran,[5] Alexander Keesling,[5] Hannes Bernien,[6] Markus Greiner,[5] Vladan Vuletić,[7] Mikhail D. Lukin,[5] Roger G. Melko,[2,3] and Manuel Endres[4]



Learning the Hamiltonian from Measurements (not covered in these lectures)

Scalably learning quantum many-body Hamiltonians from dynamical data

F. Wilde,[1] A. Kshetrimayum,[2,1] I. Roth,[3] D. Hangleiter,[4,5] R. Sweke,[1,*] and J. Eisert[1,2,6]

arXiv:2209.14328

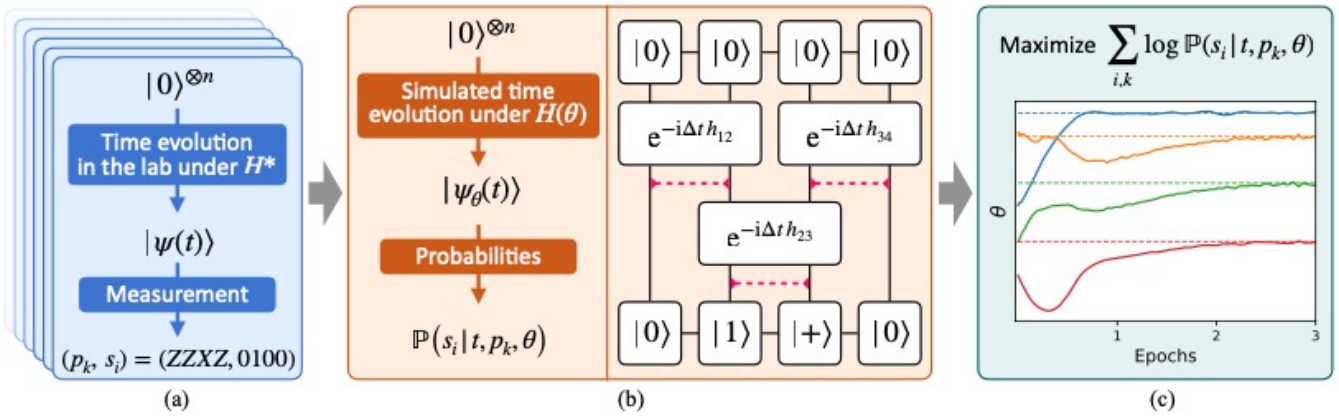


FIG. 1. The core idea of the method. **(a)** Data are accumulated from repeated measurements at different times. Each data point corresponds to a randomly chosen Pauli basis and a string of binary measurement outcomes. **(b)** The time evolution is simulated under a Hamiltonian $H(\theta) \in \mathcal{C}$ using the TEBD algorithm. After each contraction with a local time step $e^{-i\Delta t\, h_{\alpha,\beta}}$ the resulting rank-4 tensor is split and the bond dimensions are truncated to their original size, as indicated by the dashed red lines. **(c)** The best suitable parameters $\theta^{\mathcal{D}} \in \mathbb{R}^{\nu}$—according to the negative log-likelihood cost function—matching the observed data are learned using automatic differentiation and gradient-based optimization. In the case shown here $\nu = 4$ and the true parameter values (dashed lines) are recovered successfully.

For reference - most likely, we won't have time to discuss this in these lectures

Phase Classification
(not covered in these lectures)

Antiferromagnetic Transition in the Hubbard Model

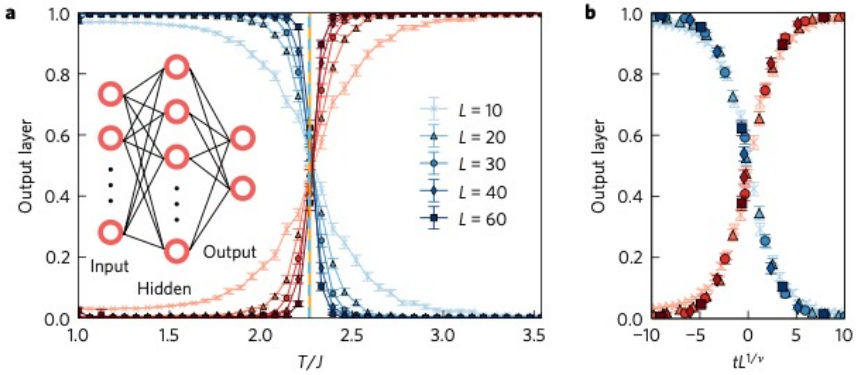
In those examples:
From Monte Carlo Data

PHYSICAL REVIEW X 7, 031038 (2017)

Machine Learning Phases of Strongly Correlated Fermions

Kelvin Ch'ng,[1] Juan Carrasquilla,[2] Roger G. Melko,[2,3] and Ehsan Khatami[1]
[1]*Department of Physics and Astronomy, San José State University, San José, California 95192, USA*
[2]*Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada*
[3]*Department of Physics and Astronomy, University of Waterloo, Ontario N2L 3G1, Canada*
(Received 14 June 2017; published 30 August 2017)

Classical Ising Model



nature physics
LETTERS
PUBLISHED ONLINE: 13 FEBRUARY 2017 | DOI: 10.1038/NPHYS4035

Machine learning phases of matter

Juan Carrasquilla[1]* and Roger G. Melko[1,2]

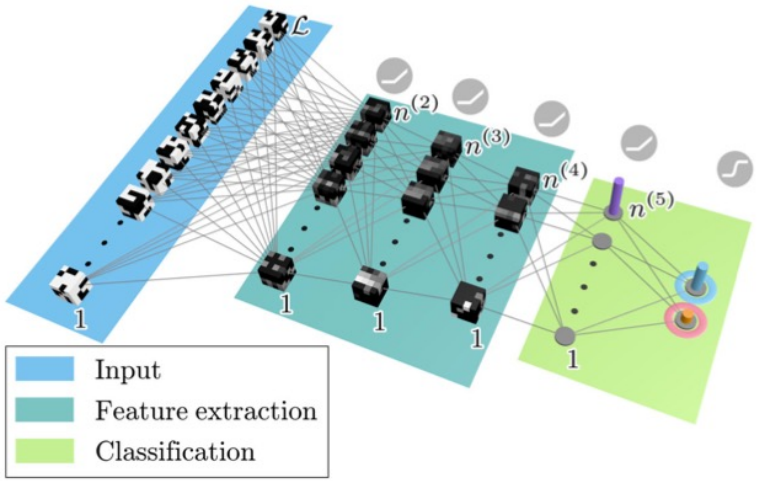


FIG. 1. Architecture of the 3D convolutional neural network used to obtain T_N for the 3D Hubbard model. For input, we use the auxiliary field configurations in a four-dimensional grid, three spatial dimensions of size 4 (total of $N = 4^3$ sites), and one imaginary time dimension of size $\mathcal{L} = 200$. Numbers of volu-

Control of Quantum Systems

(not covered in these lectures)

PHYSICAL REVIEW X **12,** 011059 (2022)

Model-Free Quantum Control with Reinforcement Learning

V. V. Sivak,[1,*] A. Eickbusch,[1] H. Liu,[1] B. Royer,[2] I. Tsioutsios,[1] and M. H. Devoret[1,†]
[1]*Department of Applied Physics, Yale University, New Haven, Connecticut 06520, USA*
[2]*Department of Physics, Yale University, New Haven, Connecticut 06520, USA*

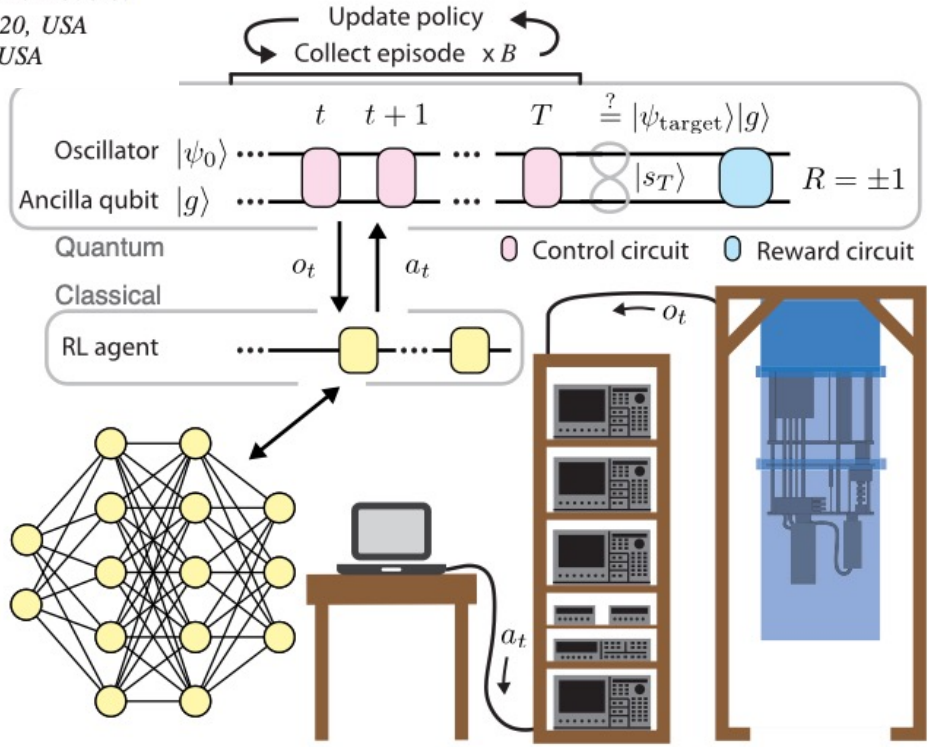


FIG. 1. Pipeline of classical reinforcement learning applied to a quantum-observable environment. The agent (yellow box), whose policy is represented with a neural network, is a program implemented in a classical computer controlling the quantum system. The quantum environment of the agent consists of a harmonic oscillator and its ancilla qubit, implemented with superconducting circuits and cryogenically cooled in the dilution refrigerator. The goal of the agent is to prepare the target state $|\psi_{\text{target}}\rangle$ of the oscillator after T time steps, starting from initial state $|\psi_0\rangle$. Importantly, the agent does not have access to the quantum state of the environment; it can only observe the environment through intermediate projective measurements of the ancilla qubit yielding binary outcomes o_t. The agent controls the environment by producing, at each time step, the action vector a_t of parameters of the control circuit (pink box). The reward R for the RL training is obtained by executing the reward circuit (blue box) on the final state $|s_T\rangle$ prepared in each episode. This circuit is designed to probabilistically answer the following question: "Is the prepared state $|s_T\rangle$ equal to $|\psi_{\text{target}}\rangle|g\rangle$?" A batch of B episodes is collected per training epoch and used in the classical optimization loop to update the policy.

For reference only - we won't have time to discuss this in these lectures

Better Density Functionals from ML
(Lectures 5-6, time permitting…)

PRL 108, 253002 (2012) PHYSICAL REVIEW LETTERS week ending 22 JUNE 2012

Finding Density Functionals with Machine Learning

John C. Snyder,[1] Matthias Rupp,[2,3] Katja Hansen,[2] Klaus-Robert Müller,[2,4] and Kieron Burke[1]

Nature Reviews 4, 357 (2022)

Machine learning and density functional theory

Ryan Pederson[1], Bhupalee Kalita[2] and Kieron Burke[1,2]

Over the past decade machine learning has made significant advances in approximating density functionals, but whether this signals the end of human-designed functionals remains to be seen. Ryan Pederson, Bhupalee Kalita and Kieron Burke discuss the rise of machine learning for functional design.

ARTICLE

https://doi.org/10.1038/s41467-020-17265-7 OPEN

Machine learning accurate exchange and correlation functionals of the electronic density

Sebastian Dick[1,2] & Marivi Fernandez-Serra[1,2]

Nature Communications (2020)

RESEARCH

QUANTUM CHEMISTRY

Science 374, 1385 (2021)

Pushing the frontiers of density functionals by solving the fractional electron problem

James Kirkpatrick[1]*†, Brendan McMorrow[1]†, David H. P. Turban[1]†, Alexander L. Gaunt[1]†, James S. Spencer[1], Alexander G. D. G. Matthews[1], Annette Obika[1], Louis Thiry[2], Meire Fortunato[1], David Pfau[1], Lara Román Castellanos[1], Stig Petersen[1], Alexander W. R. Nelson[1], Pushmeet Kohli[1], Paula Mori-Sánchez[3], Demis Hassabis[1], Aron J. Cohen[1,4]*

Accelerating Molecular Dynamics by Learning Force Fields from DFT

Seminar by Ambroise van Roekeghem, June 6

PRL 98, 146401 (2007) PHYSICAL REVIEW LETTERS week ending 6 APRIL 2007

Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces

Jörg Behler and Michele Parrinello

Department of Chemistry and Applied Biosciences, ETH Zurich, USI-Campus, Via Giuseppe Buffi 13, CH-6900 Lugano, Switzerland

(Received 27 September 2006; published 2 April 2007)

pubs.acs.org/CR Review

Machine Learning Force Fields

Oliver T. Unke,△ Stefan Chmiela,△ Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko,* and Klaus-Robert Müller*

Cite This: *Chem. Rev.* 2021, 121, 10142–10186 Read Online

Online lecture by Gabor Csanyi:
https://www.youtube.com/watch?v=ZjBff6-5amo

Some Useful General Refs on ML with a Focus on Quantum Systems

- J.Carrasquilla *Machine learning for quantum matter Advances in Physics 2020, Vol 5 1797528*
- A.Dawid et al. *Modern applications of machine learning in quantum sciences (Lecture Notes) arXiv:2204.04198*
- J.Carrasquilla and G.Torlai *How To Use Neural Networks To Investigate Quantum Many-Body Physics PRX Quantum 2, 040201 (2021) (Tutorial)*
- J.Schmidt et al. *Recent advances and applications of machine learning in solid-state materials science npj Computational Materials (2019)*

Online Journal Club (every 2 weeks): http://ultracold.org/menu/