Quantum computing, Machine Learning and Quantum Machine Learning at UiO

MHJ, Lasse Vines and other, Center for Materials Science and Nanotechnology, Center for Computing in Science Education and Department of Physics, UiO

March 20, 2024

What is this talk about?

The main emphasis is to give you a short introduction to present research and educational initiatives on Quantum Computing, Machine Learning and Quantum Machine Learning in physics.

These slides and more at http://mhjensenseminars.github.io/MachineLearningTalk/doc/pub/QuantumUiO

Thanks to many

Jane Kim (MSU), Julie Butler (MSU), Patrick Cook (MSU), Danny Jammooa (MSU), Daniel Bazin (MSU), Dean Lee (MSU), Witek Nazarewicz (MSU), Michelle Kuchera (Davidson College), Even Nordhagen (UiO), Robert Solli (UiO, Expert Analytics), Bryce Fore (ANL), Alessandro Lovato (ANL), Stefano Gandolfi (LANL), Francesco Pederiva (UniTN), and Giuseppe Carleo (EPFL). Niyaz Beysengulov and Johannes Pollanen (experiment, MSU); Zachary Stewart, Jared Weidman, and Angela Wilson (quantum chemistry, MSU) Jonas Flaten, Oskar, Leinonen, Christopher Linderälv, Øyvind Sigmundson Schøyen, Stian Dysthe Bilek, and Håkon Emil Kristiansen (UiO). Marianne Bathen, David Gongarra, Lasse Vines, and Justin Wells (experiments (UiO)). Excuses to those I have forgotten.

People

UiO

1. Morten Hjorth-Jensen (theory), Lasse Vines, Marianne Bathen Etzelmueller, Justin Wells and David Gongarra (experiment)
2. Four theory PhD students (2019-2025), one PD shared with Lasse Vines' QuTE project. And many MSc and PhD students at the SMN
3. Ten master of science students (theory), many-body physics, quantum computing, quantum machine learning and machine learning

MSU

1. Dean Lee, Scott Bogner, Angela Wilson and Heiko Hergert, theory and Johannes Pollanen and Niyaz Beysengulov, experiment
2. Four PhD students working on quantum computing and machine learning (theory)

Since 2020, final thesis of three PhD students (MSU, one QT and

Educational strategies

1. **New study direction on Quantum technology** in Bachelor program Physics and Astronomy, starts Fall 2024. Three new courses:
   - FYS1400 Introduction to Quantum Technologies
   - FYS3405/4405 Quantum Materials
   - FYS3415/4415 Quantum Computing
2. **Developed Master of Science program on Computational Science**, started fall 2018 and many students here work on quantum computing and machine learning
3. Developed courses on machine learning, from basic to advanced ones, FYS-STK3155/4155 and FYS5429/9429
4. Developed advanced course on quantum computing and quantum machine learning, FYS5419/9419
5. New study directions in Master of Science in Physics and Computational Science on Quantum technologies and more. Start fall 2025

Machine learning research

1. Solving complicated quantum mechanical many-body systems with deep learning, see references at the end
2. Developing new machine learning algorithms **with applications to quantum computing as well**
3. Analyzing experimental data from nuclear physics experiments, NIMA https://www.sciencedirect.com/science/article/abs/pii/S0168900221004460?via%3Dihub
4. Predicting solid state material platforms for quantum technologies, Nature Computational Materials https://www.nature.com/articles/s41524-022-00888-3

Quantum computing and quantum machine learning, main activities

How to use many-body theory to design quantum circuits (Quantum engineering)

1. Many-body methods like F(ull)C(onfiguration)I(nteraction) theory, Coupled-Cluster theory and other with
   - Adaptive basis sets
   - Time dependence
   - Optimization of experimental parameters
   - Feedback from experiment
2. Finding optimal parameters for tuning of entanglement, see https://arxiv.org/abs/2310.04927
3. Numerical experiments to mimick real systems, quantum twins
4. Constructing quantum circuits to simulate specific systems
5. Quantum machine learning to optimize quantum circuits

Candidate systems at UiO and MSU

1. **Quantum dots, experiments at MSU and UiO**
2. **Point Defects in semiconductors, experiments at UiO**
3. Recent article Coulomb interaction-driven entanglement of electrons on helium, see https://arxiv.org/abs/2310.04927, and PRX Quantum, under review

Many-body physics, Quantum Monte Carlo and deep learning

Given a hamiltonian H and a trial wave function Ψ_T, the variational principle states that the expectation value of $\langle H \rangle$, defined through

$$\langle E \rangle = \frac{\int d\boldsymbol{R} \Psi_T^*(\boldsymbol{R}) H(\boldsymbol{R}) \Psi_T(\boldsymbol{R})}{\int d\boldsymbol{R} \Psi_T^*(\boldsymbol{R}) \Psi_T(\boldsymbol{R})},$$

is an upper bound to the ground state energy E_0 of the hamiltonian H, that is

$$E_0 \leq \langle E \rangle.$$

In general, the integrals involved in the calculation of various expectation values are multi-dimensional ones. Traditional integration methods such as the Gauss-Legendre will not be adequate for say the computation of the energy of a many-body system. **Basic philosophy: Let a neural network find the optimal wave function**

Quantum Monte Carlo Motivation

Basic steps

Choose a trial wave function $\psi_T(\boldsymbol{R})$.

$$P(\boldsymbol{R},\boldsymbol{\alpha}) = \frac{|\psi_T(\boldsymbol{R},\boldsymbol{\alpha})|^2}{\int |\psi_T(\boldsymbol{R},\boldsymbol{\alpha})|^2 \, d\boldsymbol{R}}.$$

This is our model, or likelihood/probability distribution function (PDF). It depends on some variational parameters $\boldsymbol{\alpha}$. The approximation to the expectation value of the Hamiltonian is now

$$\langle E[\boldsymbol{\alpha}]\rangle = \frac{\int d\boldsymbol{R}\Psi_T^*(\boldsymbol{R},\boldsymbol{\alpha})H(\boldsymbol{R})\Psi_T(\boldsymbol{R},\boldsymbol{\alpha})}{\int d\boldsymbol{R}\Psi_T^*(\boldsymbol{R},\boldsymbol{\alpha})\Psi_T(\boldsymbol{R},\boldsymbol{\alpha})}.$$

Quantum Monte Carlo Motivation

Define a new quantity

$$E_L(\boldsymbol{R},\boldsymbol{\alpha}) = \frac{1}{\psi_T(\boldsymbol{R},\boldsymbol{\alpha})} H\psi_T(\boldsymbol{R},\boldsymbol{\alpha}),$$

called the local energy, which, together with our trial PDF yields

$$\langle E[\boldsymbol{\alpha}]\rangle = \int P(\boldsymbol{R}) E_L(\boldsymbol{R},\boldsymbol{\alpha}) d\boldsymbol{R} \approx \frac{1}{N}\sum_{i=1}^{N} E_L(\boldsymbol{R_i},\boldsymbol{\alpha})$$

with N being the number of Monte Carlo samples.

Monte Carlo methods and Neural Networks

Machine Learning and the Deuteron by Kebble and Rios and Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz by Adams et al.
Adams et al:

$$H_{LO} = -\sum_i \frac{\vec{\nabla}_i^2}{2m_N} + \sum_{i<j} \left(C_1 + C_2\, \vec{\sigma_i} \cdot \vec{\sigma_j}\right) e^{-r_{ij}^2 \Lambda^2/4} + D_0 \sum_{i<j<k} \sum_{\text{cyc}} e^{-\left(r_{ik}^2 + r_{ij}^2\right)\Lambda^2/4}, \quad (1)$$

where m_N is the mass of the nucleon, $\vec{\sigma_i}$ is the Pauli matrix acting on nucleon i, and $\sum_{\text{cyc}}$ stands for the cyclic permutation of i, j, and k. The low-energy constants C_1 and C_2 are fit to the deuteron binding energy and to the neutron-neutron scattering length

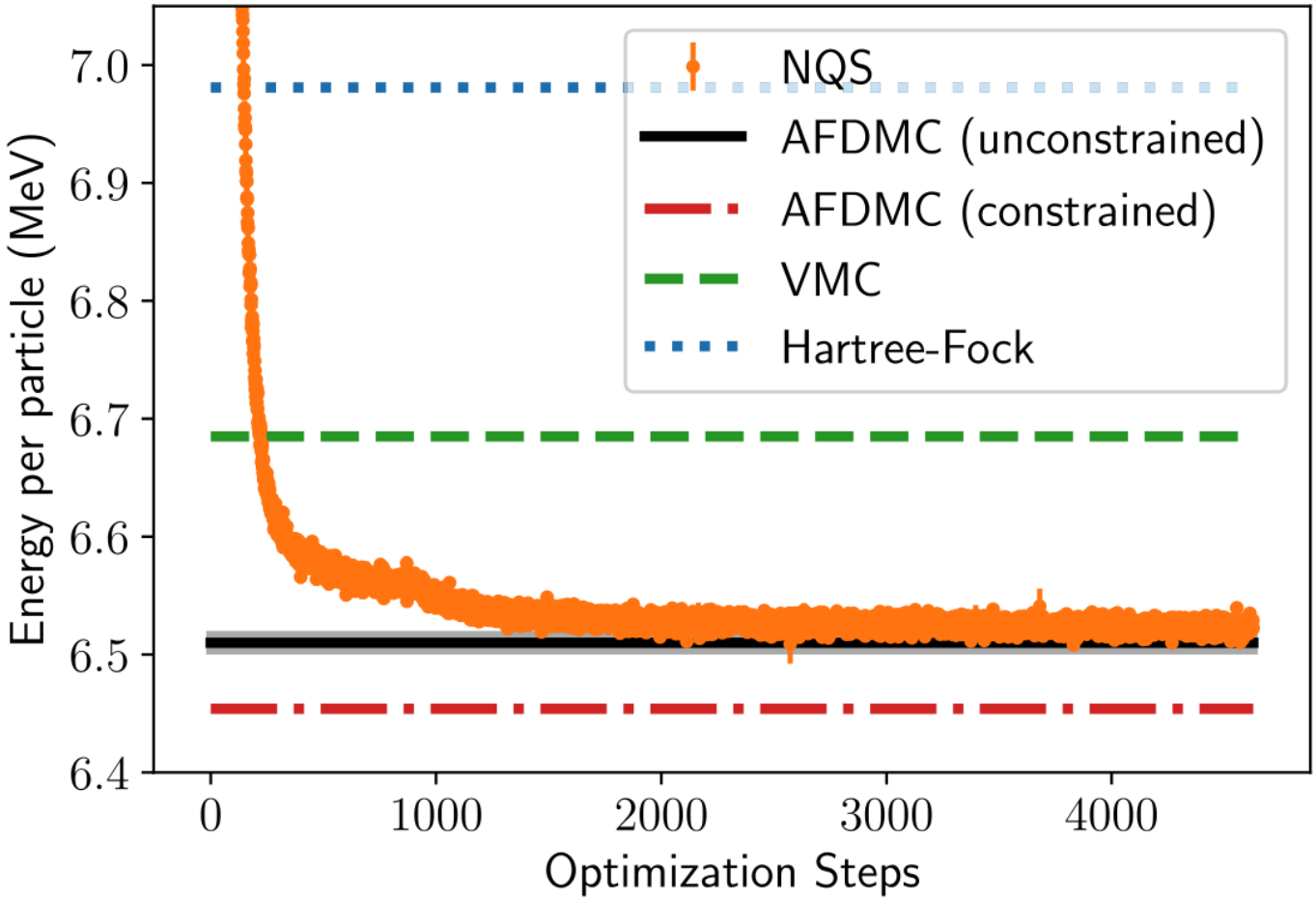
Deep learning neural networks, Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz by Adams et al.

An appealing feature of the neural network ansatz is that it is more general than the more conventional product of two- and three-body spin-independent Jastrow functions

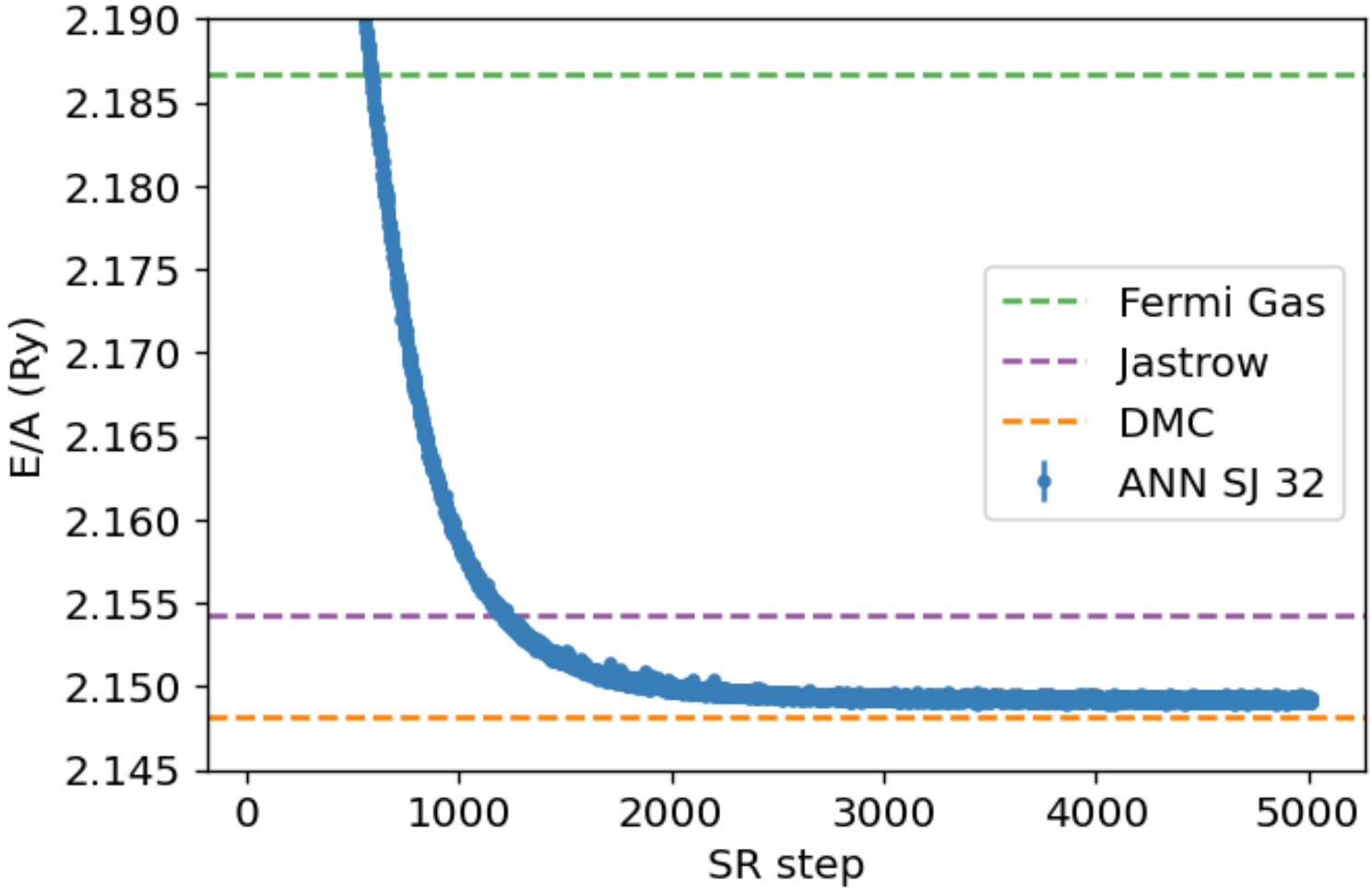
$$|\Psi_V^J\rangle = \prod_{i<j<k} \left(1 - \sum_{\text{cyc}} u(r_{ij})u(r_{jk})\right) \prod_{i<j} f(r_{ij})|\Phi\rangle\,, \qquad (2)$$

which is commonly used for nuclear Hamiltonians that do not contain tensor and spin-orbit terms. The above function is replaced by a four-layer Neural Network.

Dilute neutron star matter from neural-network quantum states by Fore et al, Physical Review Research 5, 033062 (2023) at density $\rho = 0.04$ fm^{-3}



The electron gas in three dimensions with $N = 14$ electrons (Wigner-Seitz radius $r_s = 2$ a.u.), Gabriel Pescia, Jane Kim et al. arXiv.2305.07240,



Selected references

- Artificial Intelligence and Machine Learning in Nuclear Physics, Amber Boehnlein et al., Reviews Modern of Physics 94, 031003 (2022)
- Dilute neutron star matter from neural-network quantum states by Fore et al, Physical Review Research 5, 033062 (2023)
- Neural-network quantum states for ultra-cold Fermi gases, Jane Kim et al, Nature Physics Communication, in press
- Message-Passing Neural Quantum States for the Homogeneous Electron Gas, Gabriel Pescia, Jane Kim et al. arXiv.2305.07240,
- Efficient solutions of fermionic systems using artificial neural networks, Nordhagen et al, Frontiers in Physics 11, 2023

More selected references

- Unsupervised learning for identifying events in active target experiments, R. Solli et al, Nuclear Instruments and Methods Physics A
- Coulomb interaction-driven entanglement of electrons on helium, PRX Quantum, under review
- Predicting solid state material platforms for quantum technologies, Hebnes et al, Nature Computational Materials, 2022

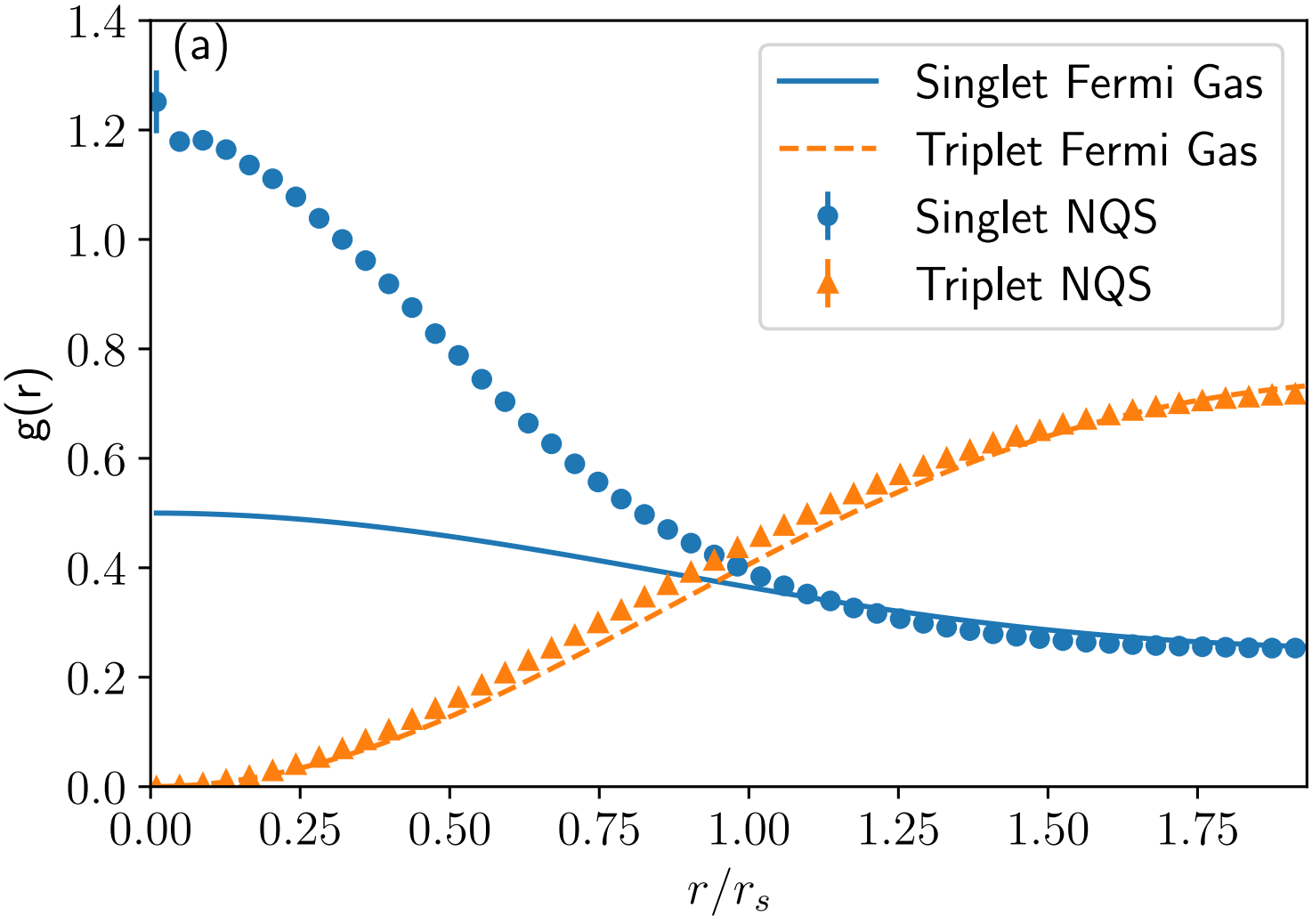
And sponsors

1. National Science Foundation, US (various grants)
2. Department of Energy, US (various grants)
3. Research Council of Norway (various grants) and University of Oslo and Michigan State University

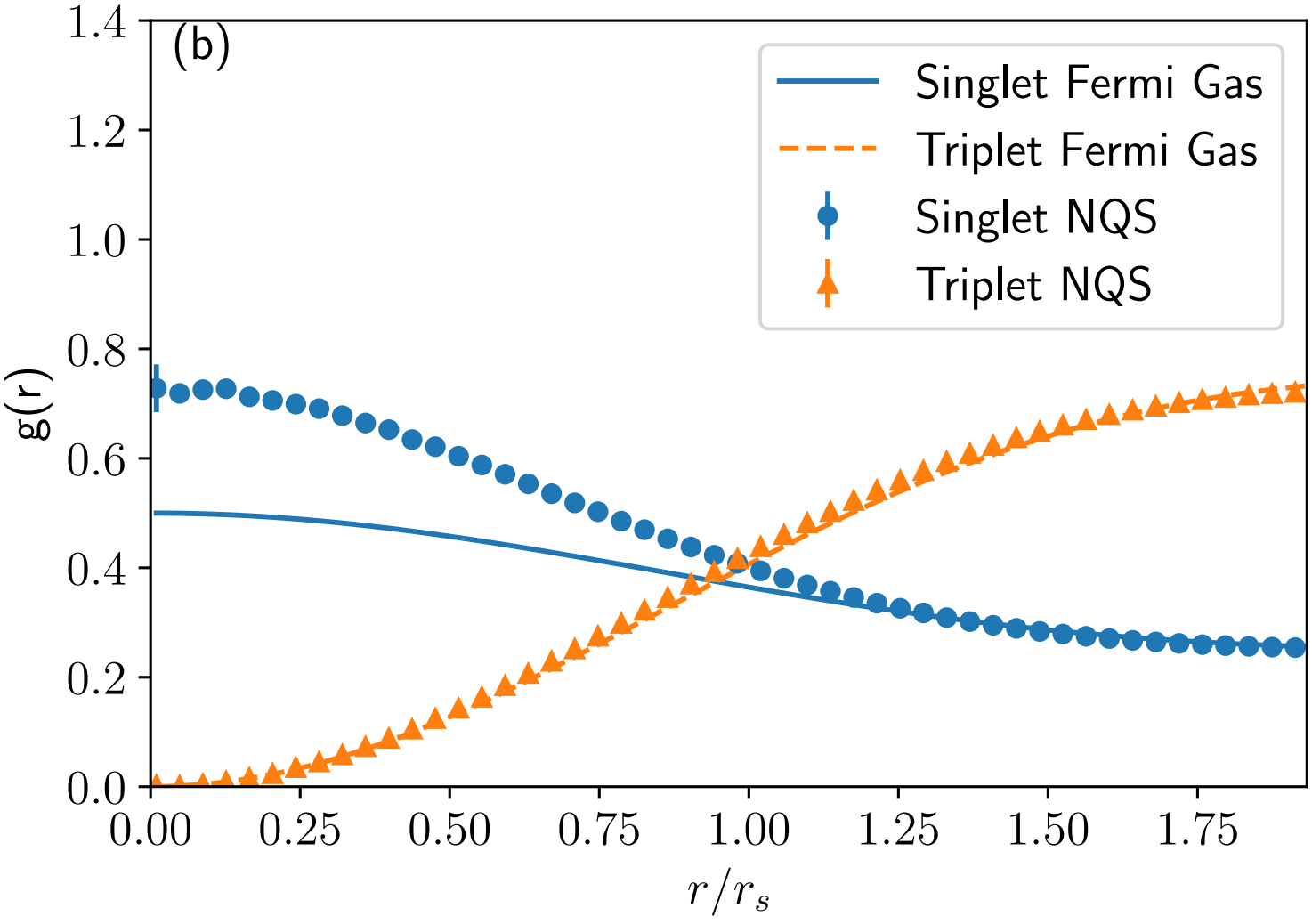
Appendix with additional material

Here follows a set of slides on various Machine Learning topics.

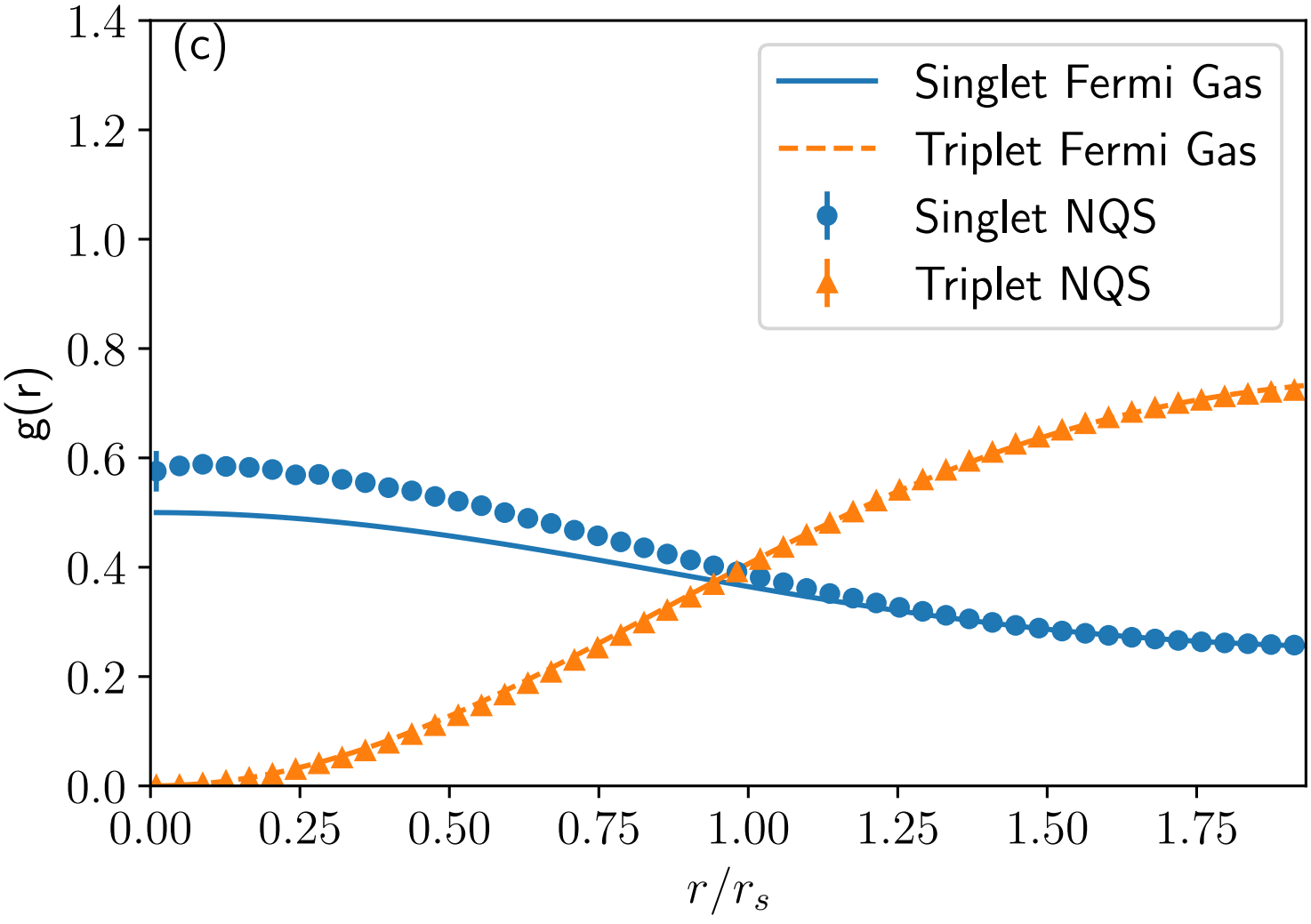
Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.01$ fm^{-3}



Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.04$ fm^{-3}



Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.08$ fm^{-3}



Universal approximation theorem

The universal approximation theorem plays a central role in deep learning. Cybenko (1989) showed the following:

Let σ be any continuous sigmoidal function such that

$$\sigma(z) = \begin{cases} 1 & z \to \infty \\ 0 & z \to -\infty \end{cases}$$

Given a continuous and deterministic function $F(\boldsymbol{x})$ on the unit cube in d-dimensions $F \in [0,1]^d$, $x \in [0,1]^d$ and a parameter $\epsilon > 0$, there is a one-layer (hidden) neural network $f(\boldsymbol{x}; \boldsymbol{\Theta})$ with $\boldsymbol{\Theta} = (\boldsymbol{W}, \boldsymbol{b})$ and $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, for which

$$|F(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\Theta})| < \epsilon \; \forall \boldsymbol{x} \in [0,1]^d.$$

The approximation theorem in words

Any continuous function $y = F(\boldsymbol{x})$ supported on the unit cube in d-dimensions can be approximated by a one-layer sigmoidal network to arbitrary accuracy.
Hornik (1991) extended the theorem by letting any non-constant, bounded activation function to be included using that the expectation value

$$\mathbb{E}[|F(\boldsymbol{x})|^2] = \int_{\boldsymbol{x}\in D} |F(\boldsymbol{x})|^2 p(\boldsymbol{x}) d\boldsymbol{x} < \infty.$$

Then we have

$$\mathbb{E}[|F(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\Theta})|^2] = \int_{\boldsymbol{x}\in D} |F(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\Theta})|^2 p(\boldsymbol{x}) d\boldsymbol{x} < \epsilon.$$

More on the general approximation theorem

None of the proofs give any insight into the relation between the number of of hidden layers and nodes and the approximation error ϵ, nor the magnitudes of $\boldsymbol{W}$ and $\boldsymbol{b}$.
Neural networks (NNs) have what we may call a kind of universality no matter what function we want to compute.

It does not mean that an NN can be used to exactly compute any function. Rather, we get an approximation that is as good as we want.

Types of machine learning

The approaches to machine learning are many, but are often split into two main categories. In *supervised learning* we know the answer to a problem, and let the computer deduce the logic behind it. On the other hand, *unsupervised learning* is a method for finding patterns and relationship in data sets without any prior knowledge of the system.

An important third category is *reinforcement learning*. This is a paradigm of learning inspired by behavioural psychology, where learning is achieved by trial-and-error, solely from rewards and punishment.

Main categories

Another way to categorize machine learning tasks is to consider the desired output of a system. Some of the most common tasks are:

- Classification: Outputs are divided into two or more classes. The goal is to produce a model that assigns inputs into one of these classes. An example is to identify digits based on pictures of hand-written ones. Classification is typically supervised learning.
- Regression: Finding a functional relationship between an input data set and a reference data set. The goal is to construct a function that maps input data to continuous output values.
- Clustering: Data are divided into groups with certain common traits, without knowing the different groups beforehand. It is thus a form of unsupervised learning.

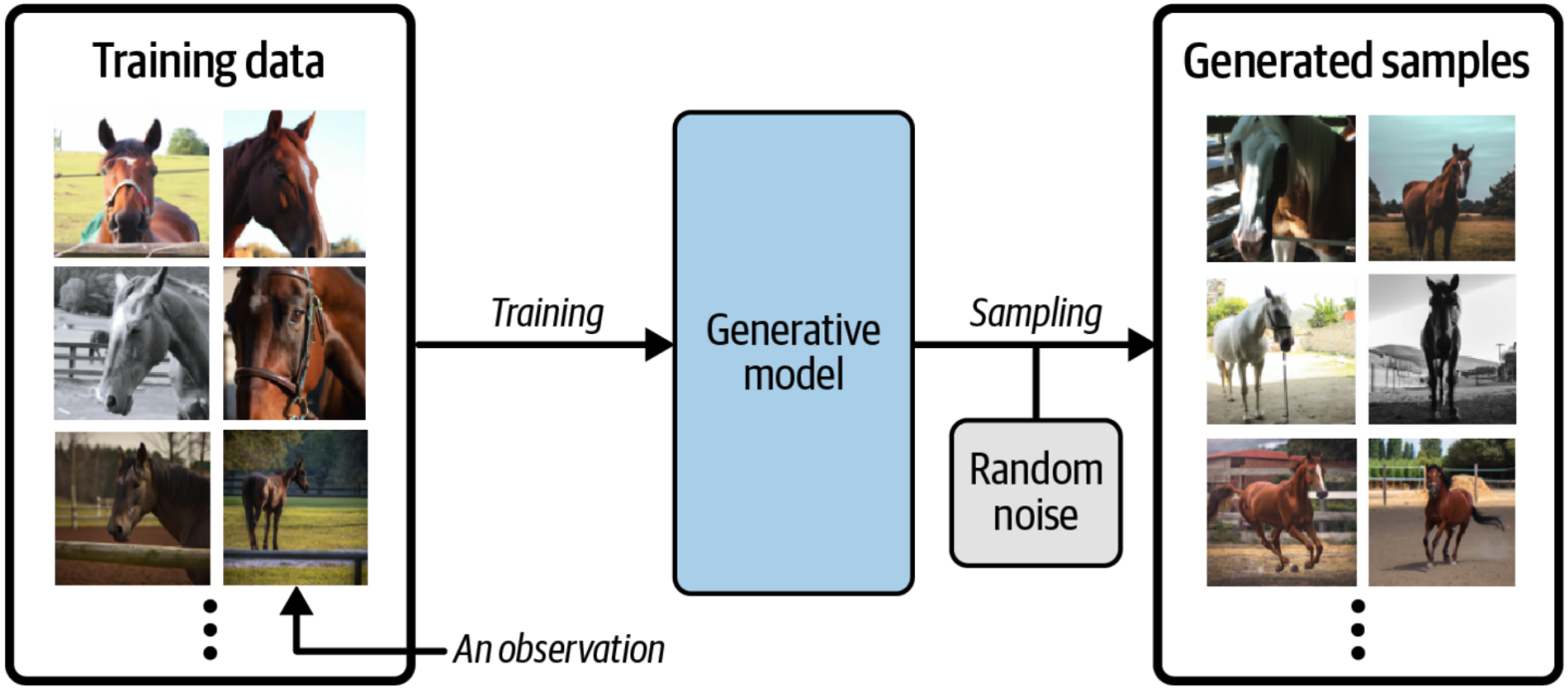
The plethora of machine learning algorithms/methods

1. Deep learning: Neural Networks (NN), Convolutional NN, Recurrent NN, Boltzmann machines, autoencoders and variational autoencoders and generative adversarial networks, stable diffusion and many more generative models
2. Bayesian statistics and Bayesian Machine Learning, Bayesian experimental design, Bayesian Regression models, Bayesian neural networks, Gaussian processes and much more
3. Dimensionality reduction (Principal component analysis), Clustering Methods and more
4. Ensemble Methods, Random forests, bagging and voting methods, gradient boosting approaches
5. Linear and logistic regression, Kernel methods, support vector machines and more
6. Reinforcement Learning; Transfer Learning and more

What Is Generative Modeling?

Generative modeling can be broadly defined as follows:
Generative modeling is a branch of machine learning that involves training a model to produce new data that is similar to a given dataset.
What does this mean in practice? Suppose we have a dataset containing photos of horses. We can train a generative model on this dataset to capture the rules that govern the complex relationships between pixels in images of horses. Then we can sample from this model to create novel, realistic images of horses that did not exist in the original dataset.

Example of generative modeling, taken from Generative Deep Learning by David Foster



Generative Modeling

In order to build a generative model, we require a dataset consisting of many examples of the entity we are trying to generate. This is known as the training data, and one such data point is called an observation.

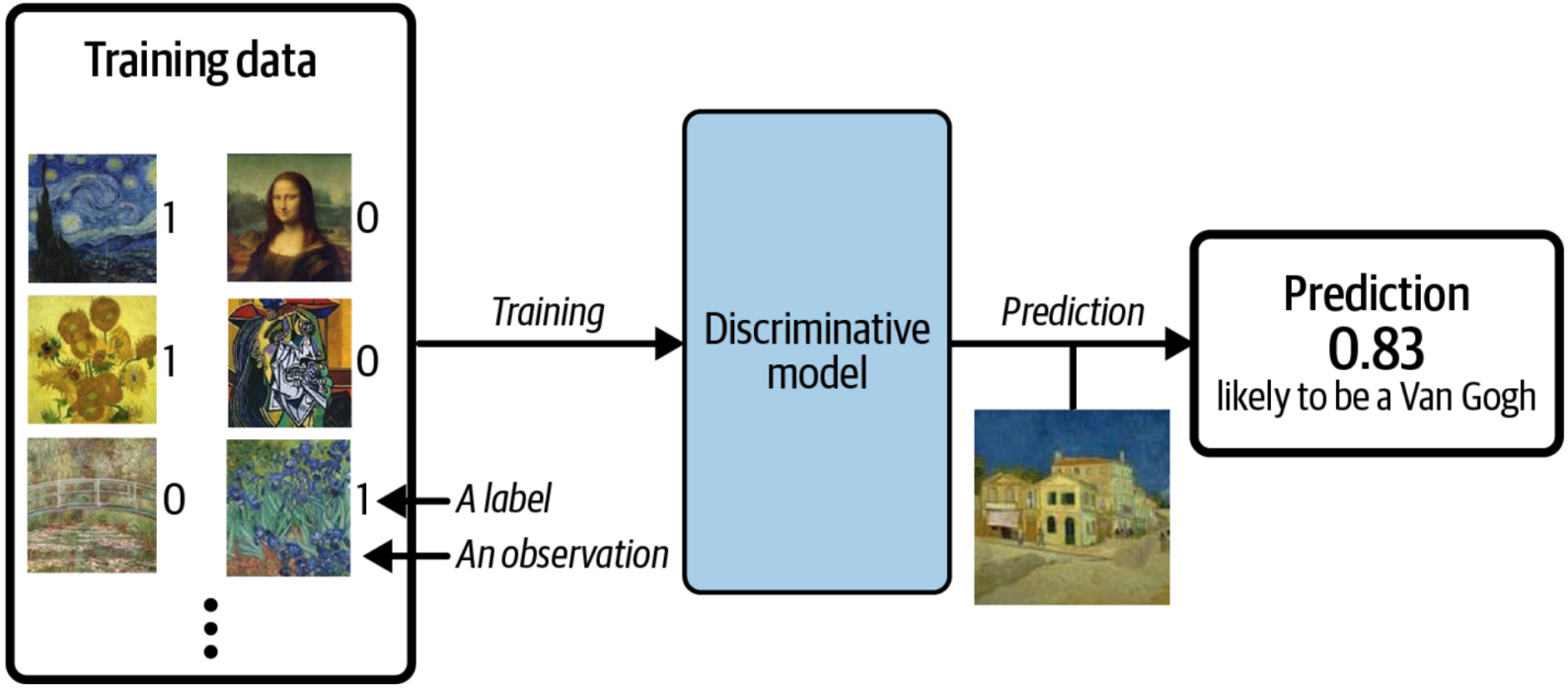
Each observation consists of many features. For an image generation problem, the features are usually the individual pixel values; for a text generation problem, the features could be individual words or groups of letters. It is our goal to build a model that can generate new sets of features that look as if they have been created using the same rules as the original data.

Conceptually, for image generation this is an incredibly difficult task, considering the vast number of ways that individual pixel values can be assigned and the relatively tiny number of such arrangements that constitute an image of the entity we are trying to generate.

Generative Versus Discriminative Modeling

In order to truly understand what generative modeling aims to achieve and why this is important, it is useful to compare it to its counterpart, discriminative modeling. If you have studied machine learning, most problems you will have faced will have most likely been discriminative in nature.

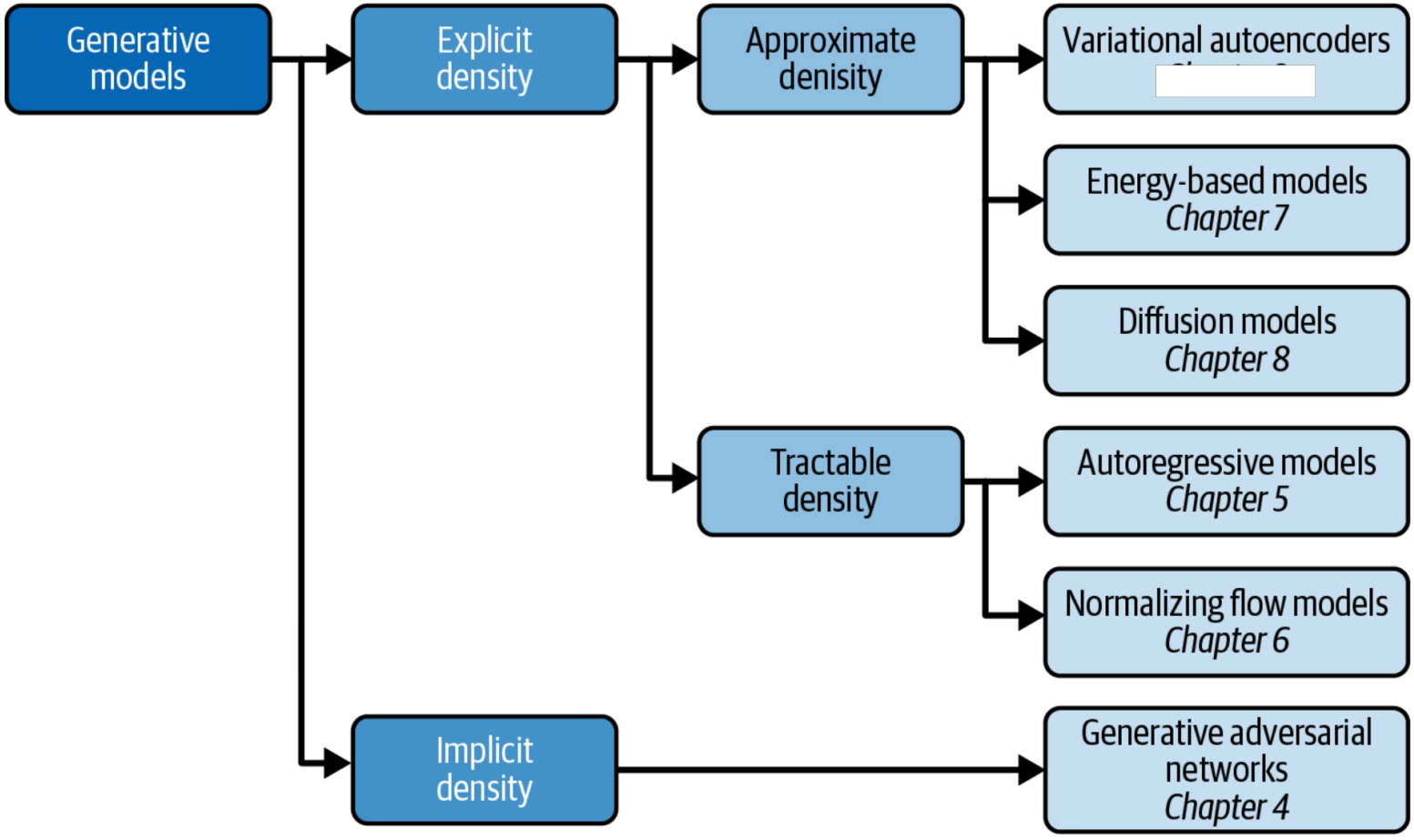
Example of discriminative modeling, taken from Generative Deeep Learning by David Foster



Discriminative Modeling

When performing discriminative modeling, each observation in the training data has a label. For a binary classification problem such as our data could be labeled as ones and zeros. Our model then learns how to discriminate between these two groups and outputs the probability that a new observation has label 1 or 0
In contrast, generative modeling doesn't require the dataset to be labeled because it concerns itself with generating entirely new data (for example an image), rather than trying to predict a label for say a given image.

Taxonomy of generative deep learning, taken from Generative Deep Learning by David Foster



Good books with hands-on material and codes

- Sebastian Rashcka et al, Machine learning with Sickit-Learn and PyTorch
- David Foster, Generative Deep Learning with TensorFlow
- Bali and Gavras, Generative AI with Python and TensorFlow 2

All three books have GitHub addresses from where one can download all codes. We will borrow most of the material from these three texts as well as from Goodfellow, Bengio and Courville's text Deep Learning

What are the basic Machine Learning ingredients?

Almost every problem in ML and data science starts with the same ingredients:

- The dataset $\boldsymbol{x}$ (could be some observable quantity of the system we are studying)
- A model which is a function of a set of parameters $\boldsymbol{\alpha}$ that relates to the dataset, say a likelihood function $p(\boldsymbol{x}|\boldsymbol{\alpha})$ or just a simple model $f(\boldsymbol{\alpha})$
- A so-called **loss/cost/risk** function $\mathcal{C}(\boldsymbol{x}, f(\boldsymbol{\alpha}))$ which allows us to decide how well our model represents the dataset.

We seek to minimize the function $\mathcal{C}(\boldsymbol{x}, f(\boldsymbol{\alpha}))$ by finding the parameter values which minimize $\mathcal{C}$. This leads to various minimization algorithms. It may surprise many, but at the heart of all machine learning algortihms there is an optimization problem.

Low-level machine learning, the family of ordinary least squares methods

Our data which we want to apply a machine learning method on, consist of a set of inputs $\boldsymbol{x}^T = [x_0, x_1, x_2, \dots, x_{n-1}]$ and the outputs we want to model $\boldsymbol{y}^T = [y_0, y_1, y_2, \dots, y_{n-1}]$. We assume that the output data can be represented (for a regression case) by a continuous function f through

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}.$$

Setting up the equations

In linear regression we approximate the unknown function with another continuous function $\tilde{\boldsymbol{y}}(\boldsymbol{x})$ which depends linearly on some unknown parameters $\boldsymbol{\theta}^T = [\theta_0, \theta_1, \theta_2, \ldots, \theta_{p-1}]$.
The input data can be organized in terms of a so-called design matrix with an approximating function $\tilde{\boldsymbol{y}}$

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\theta},$$

The objective/cost/loss function

The simplest approach is the mean squared error

$$C(\boldsymbol{\Theta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\left\{(\boldsymbol{y} - \tilde{\boldsymbol{y}})^T(\boldsymbol{y} - \tilde{\boldsymbol{y}})\right\},$$

or using the matrix $\boldsymbol{X}$ and in a more compact matrix-vector notation as

$$C(\boldsymbol{\Theta}) = \frac{1}{n}\left\{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})\right\}.$$

This function represents one of many possible ways to define the so-called cost function.

Training solution

Optimizing with respect to the unknown parameters θ_j we get

$$\boldsymbol{X}^T \boldsymbol{y} = \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\theta},$$

and if the matrix $\boldsymbol{X}^T \boldsymbol{X}$ is invertible we have the optimal values

$$\hat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^T \boldsymbol{X} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}.$$

We say we 'learn' the unknown parameters $\boldsymbol{\theta}$ from the last equation.

Ridge and LASSO Regression

Our optimization problem is

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \frac{1}{n}\left\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})\right\}.$$

or we can state it as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2 = \frac{1}{n}||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta}||_2^2,$$

where we have used the definition of a norm-2 vector, that is

$$||\boldsymbol{x}||_2 = \sqrt{\sum_i x_i^2}.$$

From OLS to Ridge and Lasso

By minimizing the above equation with respect to the parameters $\boldsymbol{\theta}$ we could then obtain an analytical expression for the parameters $\boldsymbol{\theta}$. We can add a regularization parameter λ by defining a new cost function to be optimized, that is

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \frac{1}{n}||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta}||_2^2+\lambda||\boldsymbol{\theta}||_2^2$$

which leads to the Ridge regression minimization problem where we require that $||\boldsymbol{\theta}||_2^2 \leq t$, where t is a finite number larger than zero. We do not include such a constraints in the discussions here.

Lasso regression

Defining

$$C(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2 + \lambda||\boldsymbol{\theta}||_1,$$

we have a new optimization equation

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2 + \lambda||\boldsymbol{\theta}||_1$$

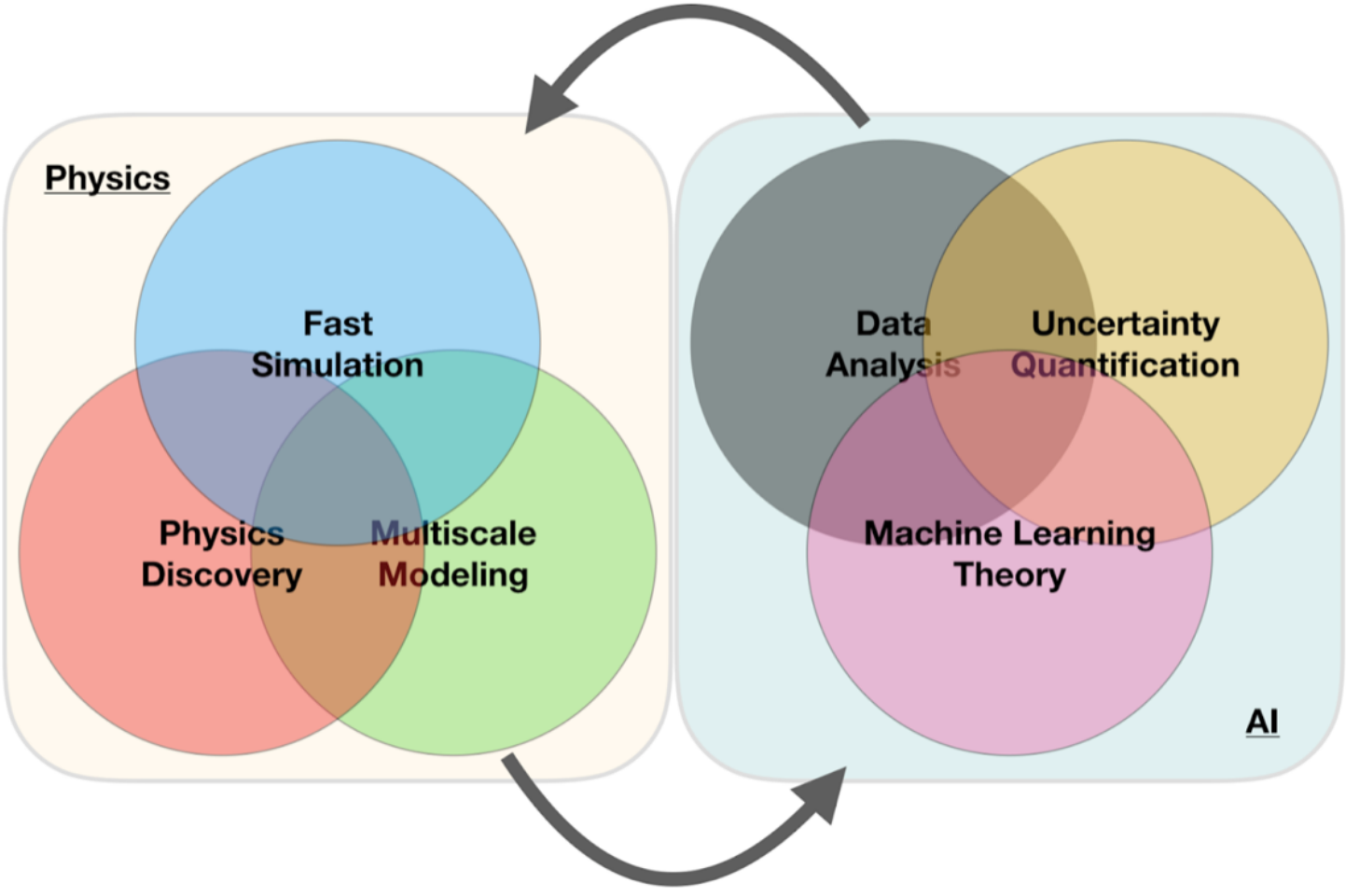
which leads to Lasso regression. Lasso stands for least absolute shrinkage and selection operator. Here we have defined the norm-1 as

$$||\boldsymbol{x}||_1 = \sum_i |x_i|.$$

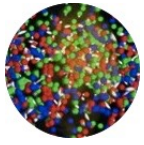
Selected references

- Mehta et al. and Physics Reports (2019).
- Machine Learning and the Physical Sciences by Carleo et al
- Artificial Intelligence and Machine Learning in Nuclear Physics, Amber Boehnlein et al., Reviews Modern of Physics 94, 031003 (2022)
- Dilute neutron star matter from neural-network quantum states by Fore et al, Physical Review Research 5, 033062 (2023)
- Neural-network quantum states for ultra-cold Fermi gases, Jane Kim et al, Nature Physics Communcication, submitted
- Message-Passing Neural Quantum States for the Homogeneous Electron Gas, Gabriel Pescia, Jane Kim et al. arXiv.2305.07240,
- Efficient solutions of fermionic systems using artificial neural networks, Nordhagen et al, Frontiers in Physics 11, 2023
- Particle Data Group summary on ML methods

Machine learning. A simple perspective on the interface between ML and Physics

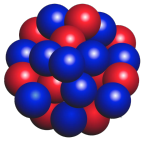


ML in Nuclear Physics (or any field in physics)

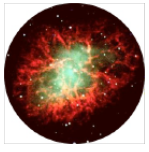


Hot and Dense Nuclear Matter

Hadrons

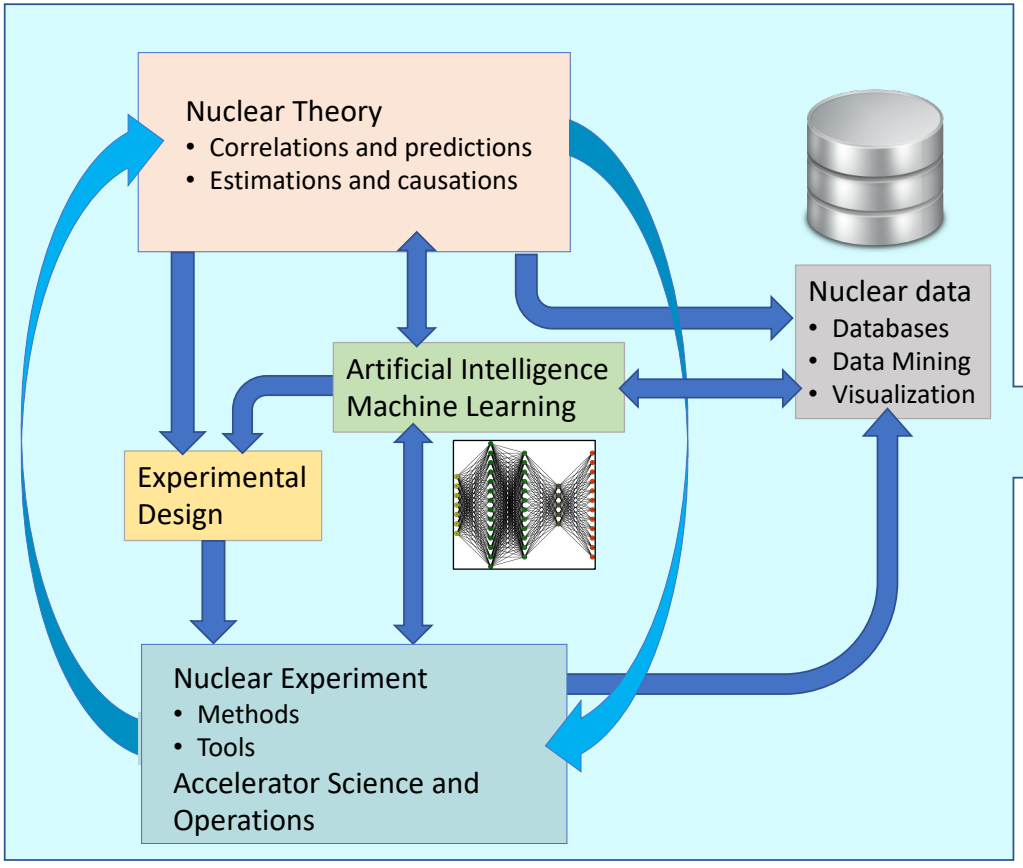


Atomic Nucleus



Nuclei in the Cosmos

Fundamental interactions



Scientific Machine Learning

An important and emerging field is what has been dubbed as scientific ML, see the article by Deiana et al "Applications and Techniques for Fast Machine Learning in Science, Big Data **5**, 787421 (2022):https://doi.org/10.3389/fdata.2022.787421"

The authors discuss applications and techniques for fast machine learning (ML) in science – the concept of integrating power ML methods into the real-time experimental data processing loop to accelerate scientific discovery. The report covers three main areas

1. applications for fast ML across a number of scientific domains;
2. techniques for training and implementing performant and resource-efficient ML algorithms;
3. and computing architectures, platforms, and technologies for deploying these algorithms.

ML for detectors

Engineering

Volume 6, Issue 3, March 2020, Pages 264-274

Research Artificial Intelligence—Review

A Survey of Accelerator Architectures for Deep Neural Networks

Yiran Chen [a], Yuan Xie [b], Linghao Song [a], Fan Chen [a], Tianqi Tang [b]

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.eng.2020.01.007

Get rights and content

Under a Creative Commons license

open access

Abstract

Physics driven Machine Learning

Another hot topic is what has loosely been dubbed **Physics-driven deep learning**. See the recent work on Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nature Machine Learning, vol 3, 218 (2021).

From their abstract

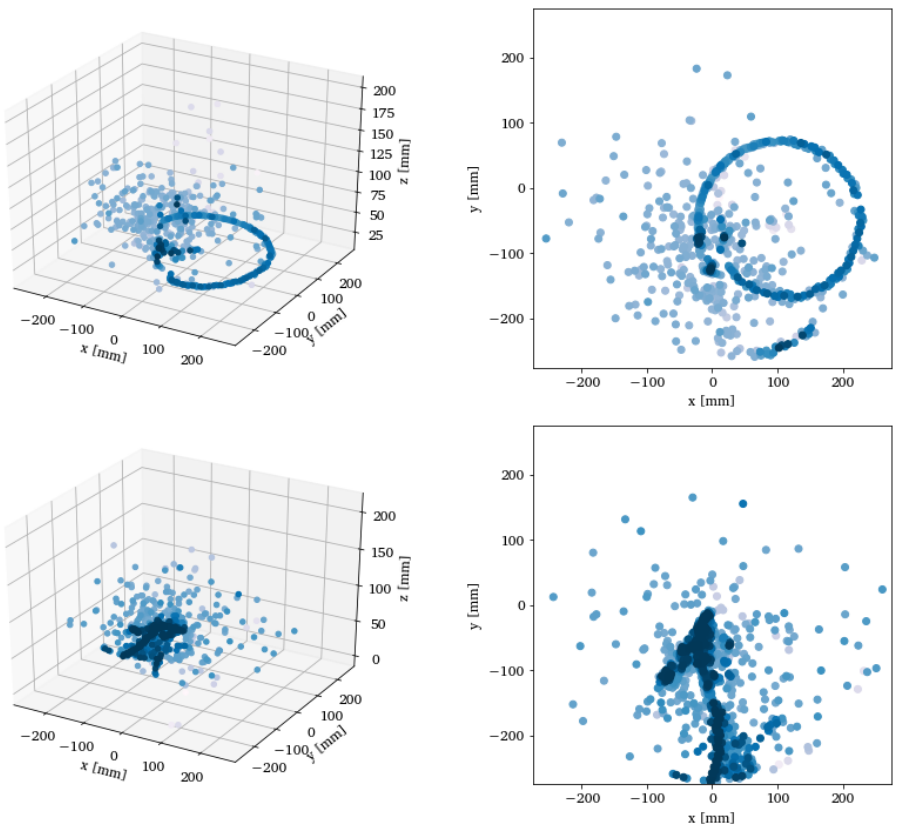
A less known but powerful result is that an NN with a single hidden layer can accurately approximate any nonlinear continuous operator. This universal approximation theorem of operators is suggestive of the structure and potential of deep neural networks (DNNs) in learning continuous operators or complex systems from streams of scattered data. ... We demonstrate that DeepONet can learn various explicit operators, such as integrals and fractional Laplacians, as well as implicit operators that represent deterministic and stochastic differential equations.

And more

- An important application of AI/ML methods is to improve the estimation of bias or uncertainty due to the introduction of or lack of physical constraints in various theoretical models.
- In theory, we expect to use AI/ML algorithms and methods to improve our knowledge about correlations of physical model parameters in data for quantum many-body systems. Deep learning methods show great promise in circumventing the exploding dimensionalities encountered in quantum mechanical many-body studies.
- Merging a frequentist approach (the standard path in ML theory) with a Bayesian approach, has the potential to infer better probabilitity distributions and error estimates.
- Machine Learning and Quantum Computing is a very interesting avenue to explore. See for example a recent talk by Sofia Vallecorsa.

Argon-46 by Solli et al., NIMA 1010, 165461 (2021)

Representations of two events from the Argon-46 experiment. Each row is one event in two projections, where the color intensity of each point indicates higher charge values recorded by the detector. The bottom row illustrates a carbon event with a large fraction of noise, while the top row shows a proton event almost free of noise.



Efficient solutions of fermionic systems using artificial neural networks, Nordhagen et al, Frontiers in Physics 11, 2023

The Hamiltonian of the quantum dot is given by

$$\hat{H} = \hat{H}_0 + \hat{V},$$

where $\hat{H}_0$ is the many-body HO Hamiltonian, and $\hat{V}$ is the inter-electron Coulomb interactions. In dimensionless units,

$$\hat{V} = \sum_{i<j}^{N} \frac{1}{r_{ij}},$$

with $r_{ij} = \sqrt{\mathrm{r}_i^2 - \mathrm{r}_j^2}$.

Separable Hamiltonian with the relative motion part ($r_{ij} = r$)

$$\hat{H}_r = -\nabla_r^2 + \frac{1}{4}\omega^2 r^2 + \frac{1}{r},$$

Analytical solutions in two and three dimensions (M. Taut 1993 and 1994).

Generative models: Why Boltzmann machines?

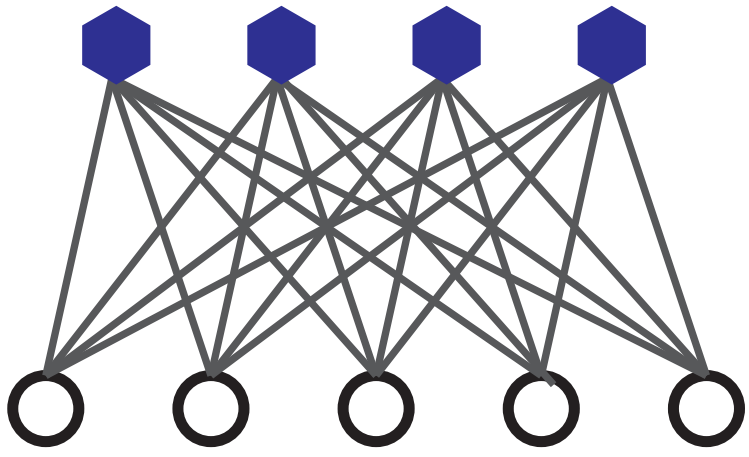
What is known as restricted Boltzmann Machines (RMB) have received a lot of attention lately. One of the major reasons is that they can be stacked layer-wise to build deep neural networks that capture complicated statistics.
The original RBMs had just one visible layer and a hidden layer, but recently so-called Gaussian-binary RBMs have gained quite some popularity in imaging since they are capable of modeling continuous data that are common to natural images.
Furthermore, they have been used to solve complicated quantum mechanical many-particle problems or classical statistical physics problems like the Ising and Potts classes of models.

The structure of the RBM network

Hidden Layer

Interactions

Visible Layer



$b_\mu(h_\mu)$

$W_{i\mu}v_i h_\mu$

$a_i(v_i)$

The network

The network layers:

1. A function $\boldsymbol{x}$ that represents the visible layer, a vector of M elements (nodes). This layer represents both what the RBM might be given as training input, and what we want it to be able to reconstruct. This might for example be the pixels of an image, the spin values of the Ising model, or coefficients representing speech.
2. The function $\boldsymbol{h}$ represents the hidden, or latent, layer. A vector of N elements (nodes). Also called "feature detectors".

Goals

The goal of the hidden layer is to increase the model's expressive power. We encode complex interactions between visible variables by introducing additional, hidden variables that interact with visible degrees of freedom in a simple manner, yet still reproduce the complex correlations between visible degrees in the data once marginalized over (integrated out).

The network parameters, to be optimized/learned:

1. $\boldsymbol{a}$ represents the visible bias, a vector of same length as $\boldsymbol{x}$.
2. $\boldsymbol{b}$ represents the hidden bias, a vector of same lenght as $\boldsymbol{h}$.
3. W represents the interaction weights, a matrix of size $M \times N$.

Joint distribution

The restricted Boltzmann machine is described by a Bolztmann distribution

$$P_{\mathrm{rbm}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} \exp -E(\boldsymbol{x}, \boldsymbol{h}),$$

where Z is the normalization constant or partition function, defined as

$$Z = \int \int \exp -E(\boldsymbol{x}, \boldsymbol{h}) d\boldsymbol{x} d\boldsymbol{h}.$$

Note the absence of the inverse temperature in these equations.

Network Elements, the energy function

The function $E(\boldsymbol{x}, \boldsymbol{h})$ gives the **energy** of a configuration (pair of vectors) $(\boldsymbol{x}, \boldsymbol{h})$. The lower the energy of a configuration, the higher the probability of it. This function also depends on the parameters $\boldsymbol{a}$, $\boldsymbol{b}$ and W. Thus, when we adjust them during the learning procedure, we are adjusting the energy function to best fit our problem.

Defining different types of RBMs (Energy based models)

There are different variants of RBMs, and the differences lie in the types of visible and hidden units we choose as well as in the implementation of the energy function $E(\boldsymbol{x}, \boldsymbol{h})$. The connection between the nodes in the two layers is given by the weights w_{ij}.

Binary-Binary RBM:

RBMs were first developed using binary units in both the visible and hidden layer. The corresponding energy function is defined as follows:

$$E(\boldsymbol{x}, \boldsymbol{h}) = -\sum_{i}^{M} x_i a_i - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{M,N} x_i w_{ij} h_j,$$

where the binary values taken on by the nodes are most commonly 0 and 1.

Gaussian binary

Gaussian-Binary RBM:

Another varient is the RBM where the visible units are Gaussian while the hidden units remain binary:

$$E(\boldsymbol{x}, \boldsymbol{h}) = \sum_{i}^{M} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{M,N} \frac{x_i w_{ij} h_j}{\sigma_i^2}.$$

Representing the wave function

The wavefunction should be a probability amplitude depending on $\boldsymbol{x}$. The RBM model is given by the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{h}$

$$P_{\mathrm{rbm}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} \exp -E(\boldsymbol{x}, \boldsymbol{h}).$$

To find the marginal distribution of $\boldsymbol{x}$ we set:

$$P_{\mathrm{rbm}}(\boldsymbol{x}) = \frac{1}{Z} \sum_{\boldsymbol{h}} \exp -E(\boldsymbol{x}, \boldsymbol{h}).$$

Now this is what we use to represent the wave function, calling it a neural-network quantum state (NQS)

$$|\Psi(\boldsymbol{X})|^2 = P_{\mathrm{rbm}}(\boldsymbol{x}).$$

Define the cost function

Now we don't necessarily have training data (unless we generate it by using some other method). However, what we do have is the variational principle which allows us to obtain the ground state wave function by minimizing the expectation value of the energy of a trial wavefunction (corresponding to the untrained NQS). Similarly to the traditional variational Monte Carlo method then, it is the local energy we wish to minimize. The gradient to use for the stochastic gradient descent procedure is

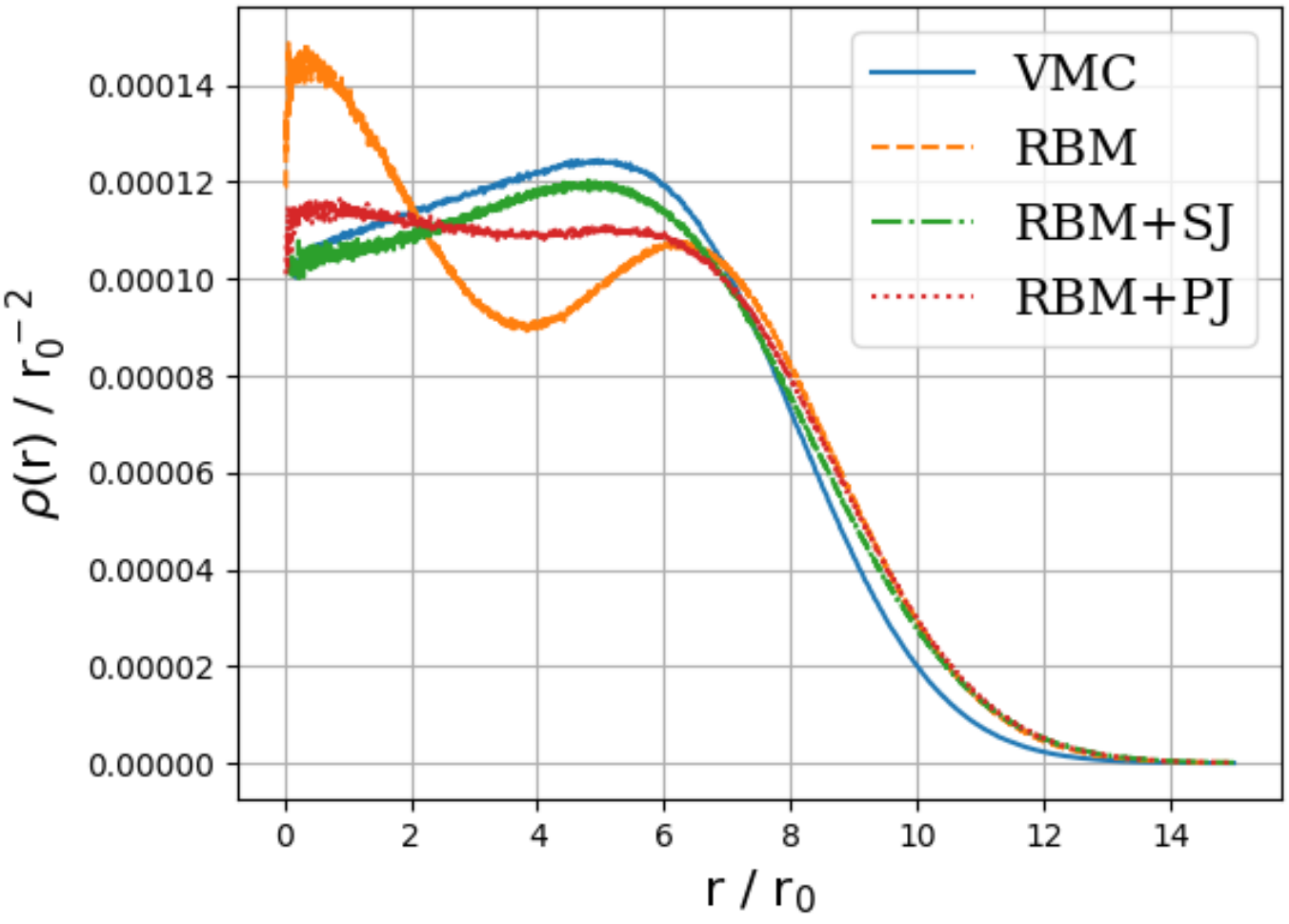
$$C_i = \frac{\partial \langle E_L \rangle}{\partial \theta_i} = 2(\langle E_L \frac{1}{\Psi} \frac{\partial \Psi}{\partial \theta_i} \rangle - \langle E_L \rangle \langle \frac{1}{\Psi} \frac{\partial \Psi}{\partial \theta_i} \rangle),$$

where the local energy is given by

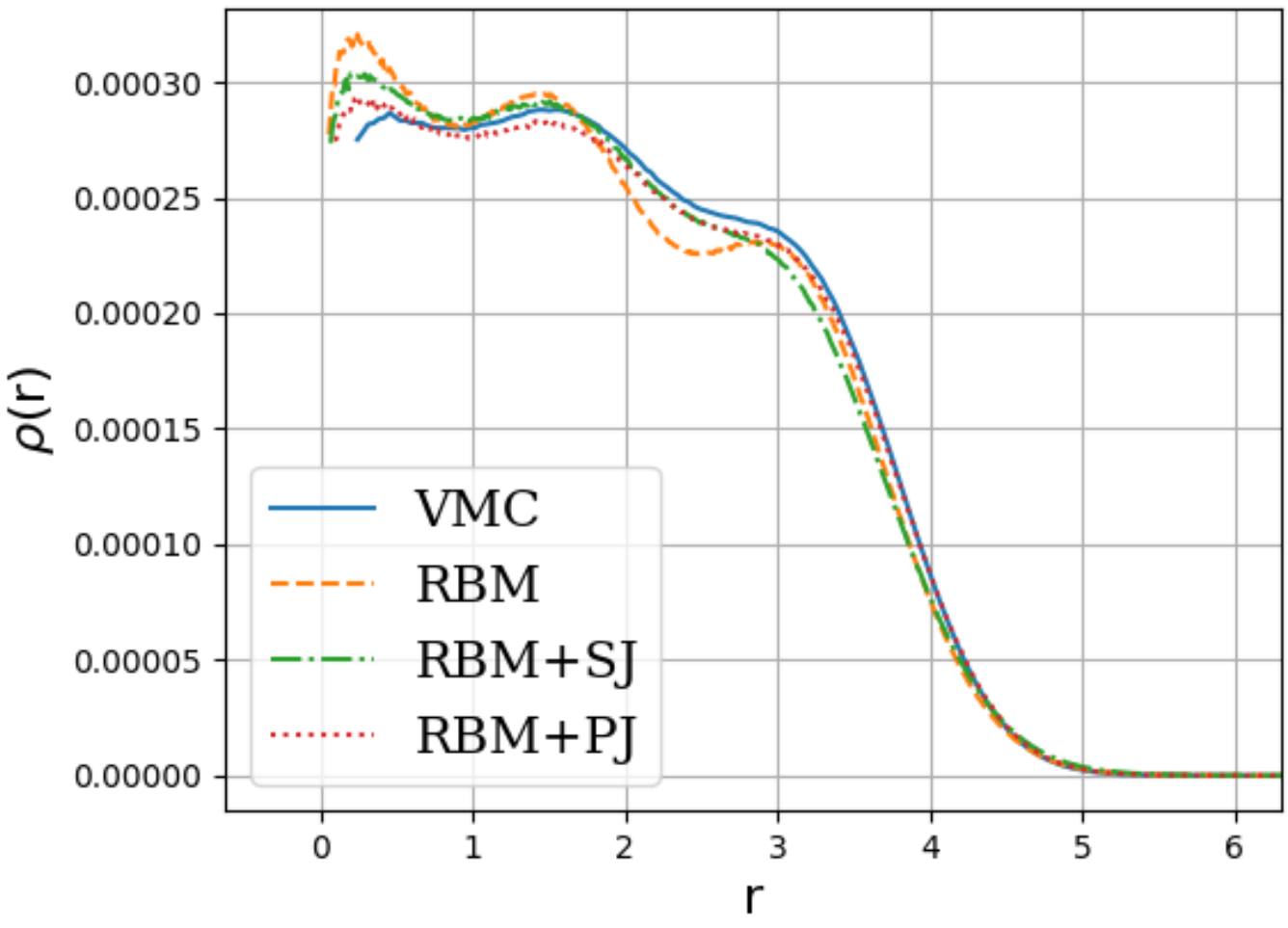
$$E_L = \frac{1}{\Psi} \hat{\boldsymbol{H}} \Psi.$$

Quantum dots and Boltzmann machines, onebody densities

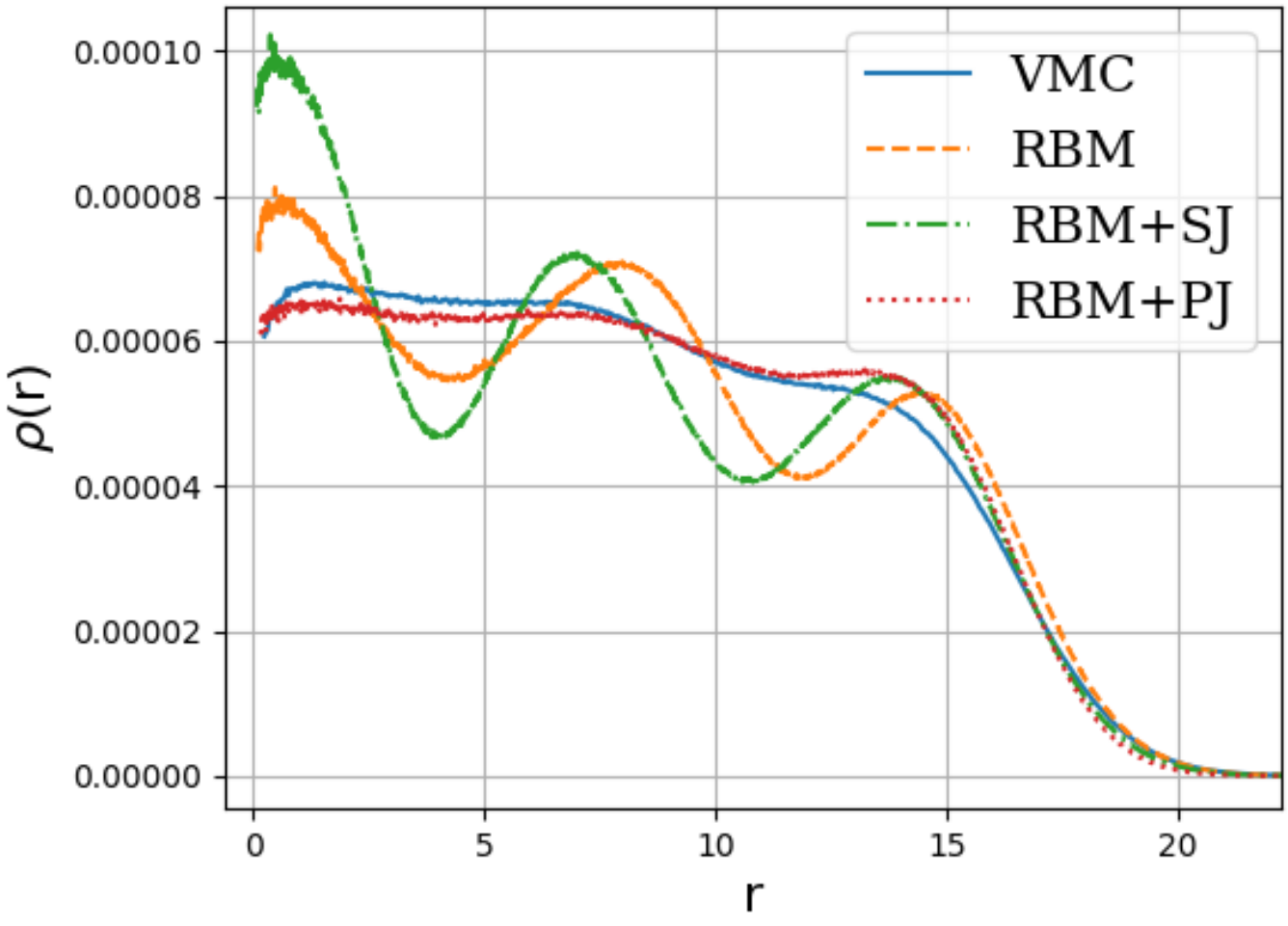
$N = 6$, $\hbar\omega = 0.1$ a.u.



Onebody densities $N = 30$, $\hbar\omega = 1.0$ a.u.



Onebody densities $N = 30$, $\hbar\omega = 0.1$ a.u.



Extrapolations and model interpretability

When you hear phrases like **predictions and estimations** and **correlations and causations**, what do you think of? May be you think of the difference between classifying new data points and generating new data points. Or perhaps you consider that correlations represent some kind of symmetric statements like if A is correlated with B, then B is correlated with A. Causation on the other hand is directional, that is if A causes B, B does not necessarily cause A.

Physics based statistical learning and data analysis

The above concepts are in some sense the difference between **old-fashioned** machine learning and statistics and Bayesian learning. In machine learning and prediction based tasks, we are often interested in developing algorithms that are capable of learning patterns from given data in an automated fashion, and then using these learned patterns to make predictions or assessments of newly given data. In many cases, our primary concern is the quality of the predictions or assessments, and we are less concerned about the underlying patterns that were learned in order to make these predictions.

Physics based statistical learning points however to approaches that give us both predictions and correlations as well as being able to produce error estimates and understand causations. This leads us to the very interesting field of Bayesian statistics.

Bayes' Theorem

Bayes' theorem

$$p(X|Y) = \frac{p(X,Y)}{\sum_{i=0}^{n-1} p(Y|X=x_i)p(x_i)} = \frac{p(Y|X)p(X)}{\sum_{i=0}^{n-1} p(Y|X=x_i)p(x_i)}.$$

The quantity $p(Y|X)$ on the right-hand side of the theorem is evaluated for the observed data Y and can be viewed as a function of the parameter space represented by X. This function is not necessarily normalized and is normally called the likelihood function. The function $p(X)$ on the right hand side is called the prior while the function on the left hand side is the called the posterior probability. The denominator on the right hand side serves as a normalization factor for the posterior distribution.

Quantified limits of the nuclear landscape

Predictions made with eleven global mass model and Bayesian model averaging