

Quantum computing, Machine Learning and Quantum Machine Learning at UiO

Morten Hjorth-Jensen, Lasse Vines et al., 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. MHJ (theory), Lasse Vines, Marianne Bathen Etzelmueler, Justin Wells, Simon Cooils and David Gongarra (experiment)
2. Four theory PhD students (2019-2025), one PD shared from 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)

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**, see
<https://arxiv.org/abs/2401.11694>
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

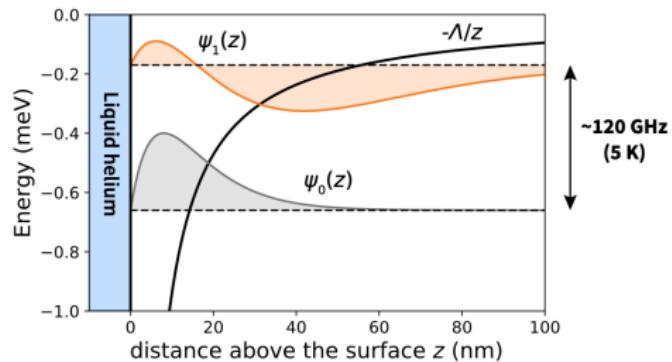
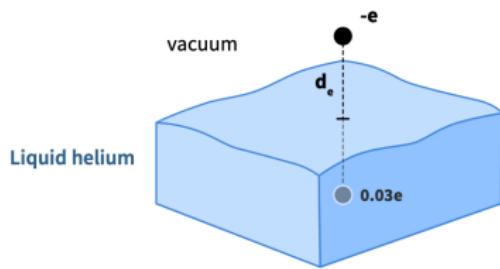
1. Quantum dots, experiments
2. Point Defects in semiconductors, experiments at UiO
3. Recent article Coulomb interaction-driven entanglement of electrons on helium, see
<https://arxiv.org/abs/2310.04927>, and PRX Quantum, under review

Electrons (quantum dots) on superfluid helium

Electrons on **superfluid helium represent** (see
https://www.youtube.com/watch?v=EuDuM-fe-1A&ab_channel=JoshuaHeath) a promising platform for investigating strongly-coupled qubits.
A systematic investigation of the controlled generation of entanglement between two trapped electrons under the influence of coherent microwave driving pulses, taking into account the effects of the Coulomb interaction between electrons, may be of great interest for quantum information processing using trapped electrons.

Experimental setup |

Electrons on helium



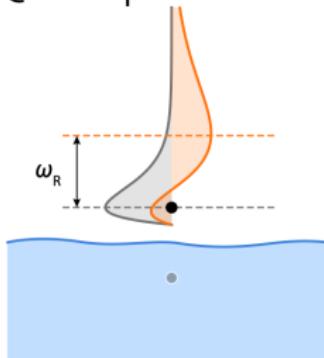
- Very high mobilities: $\mu > 10^7 \text{ cm}^2/(\text{V s})$
- Low densities: $n_s \approx 10^6 - 10^9 \text{ cm}^{-2}$
- Almost no screening: long range Coulomb interactions

“1D Hydrogen atom”
with Rydberg series of states

$$E_n = -\frac{m_e \Lambda^2}{2\hbar^2 n^2} \quad (n = 1, 2, 3\dots)$$
$$E_0 = -0.66 \text{ meV} \quad (\sim 160 \text{ GHz}, 7.6 \text{ K})$$
$$E_1 = -0.17 \text{ meV} \quad (\sim 40 \text{ GHz}, 1.9 \text{ K})$$

More on experimental setup II

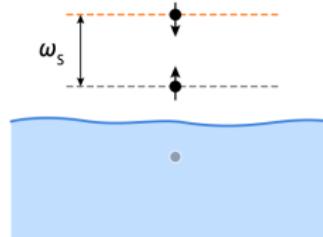
Qubit platforms with electrons on helium



Rydberg states

$$\omega_R/2\pi = 120 \text{ GHz}$$

P.M. Platzman and M.I. Dykman
Science **284**(5422), pp.1967 (1999)

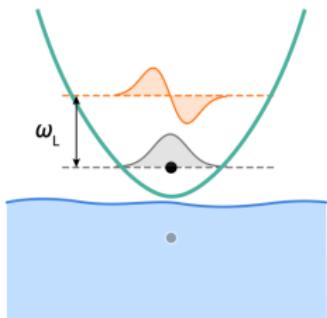


Spin states

$$\omega_s/2\pi = 5 \text{ GHz at } B = 0.2 \text{ T}$$

$(T_2 \approx 1.5 \text{ s})$

S. A. Lyon, *Phys. Rev. A* **74**, 052338 (2006)

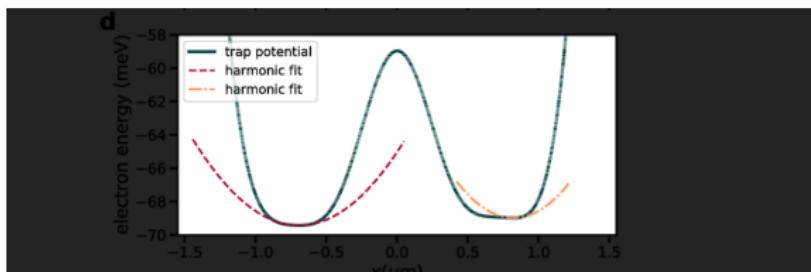


Lateral motional states

$$\omega_s/2\pi = 5 \text{ GHz}$$

D.I. Schuster et al., *Phys. Rev. Lett.* **105**, 040503 (2010)

Experimental set up



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

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

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 $\psi_T(\mathbf{R})$.

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

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

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

Quantum Monte Carlo Motivation

Define a new quantity

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

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

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

with N being the number of Monte Carlo samples.

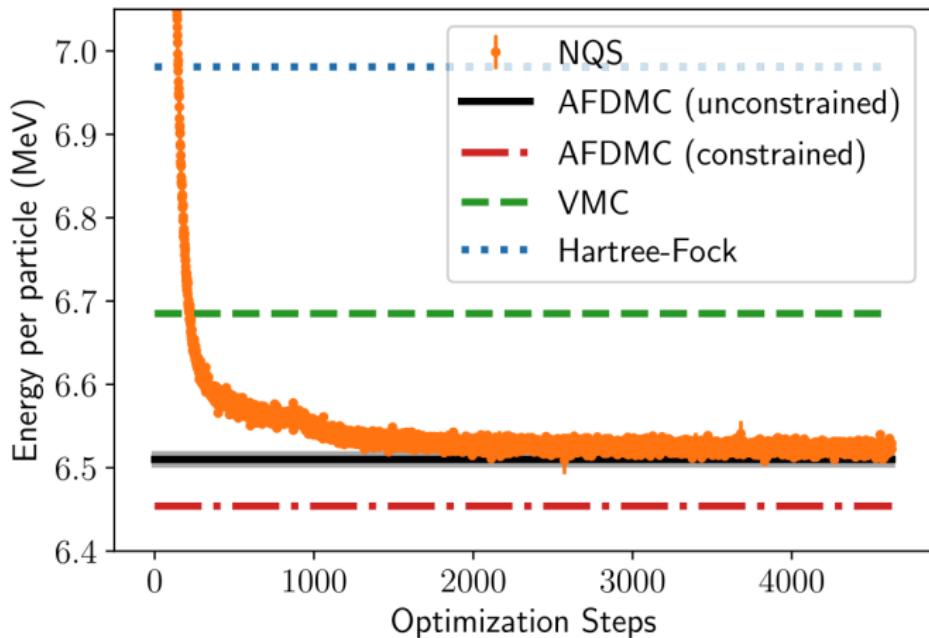
Deep learning neural networks, Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz by Adams et al.

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

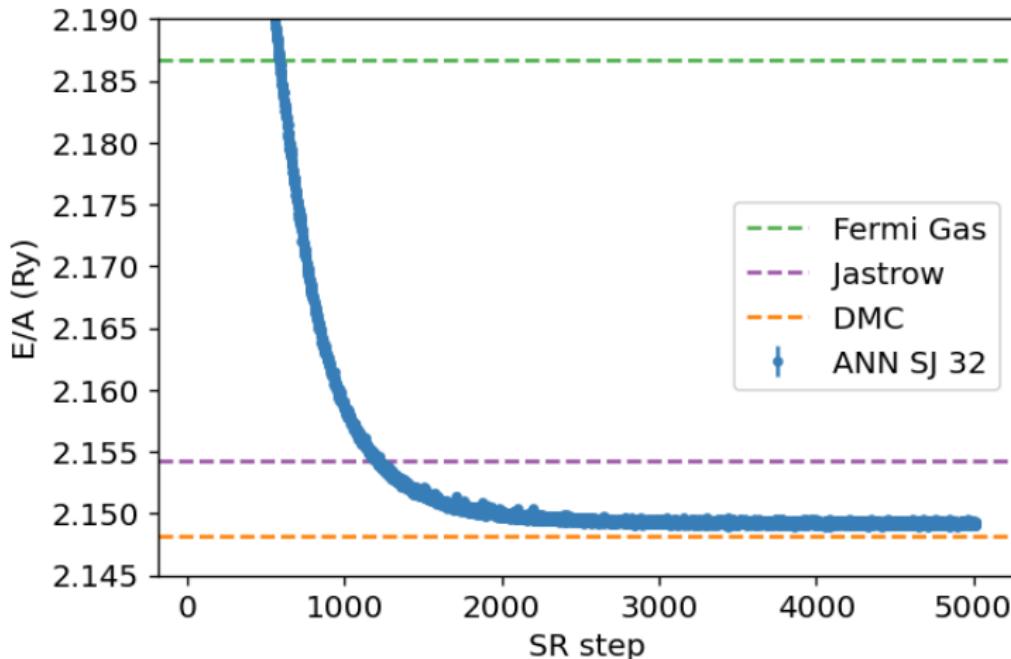
$$|\Psi_V^J\rangle = \prod_{i < j < k} \left(1 - \sum_{\text{cyc}} u(r_{ij})u(r_{jk})\right) \prod_{i < j} f(r_{ij}) |\Phi\rangle, \quad (1)$$

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

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



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

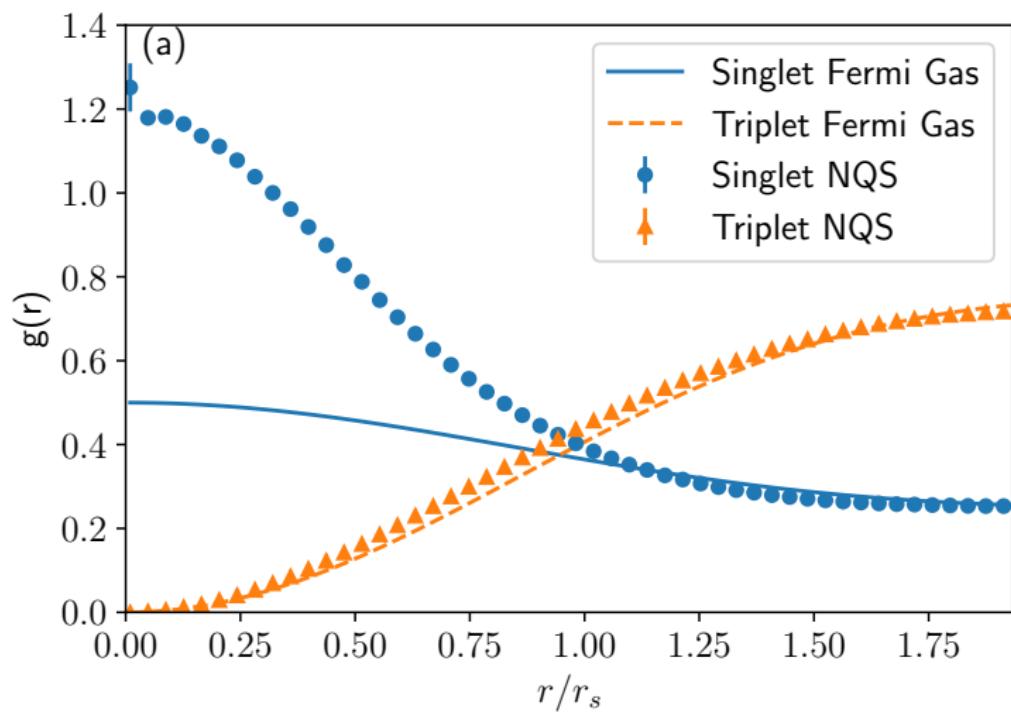
And sponsors

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

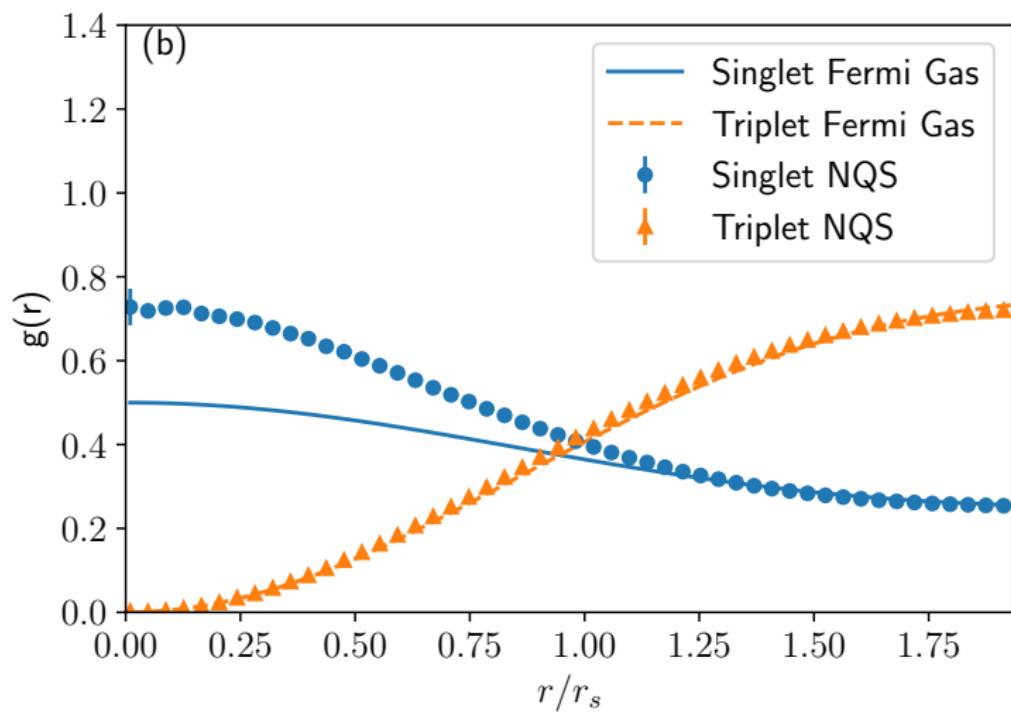
Appendix with additional material

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

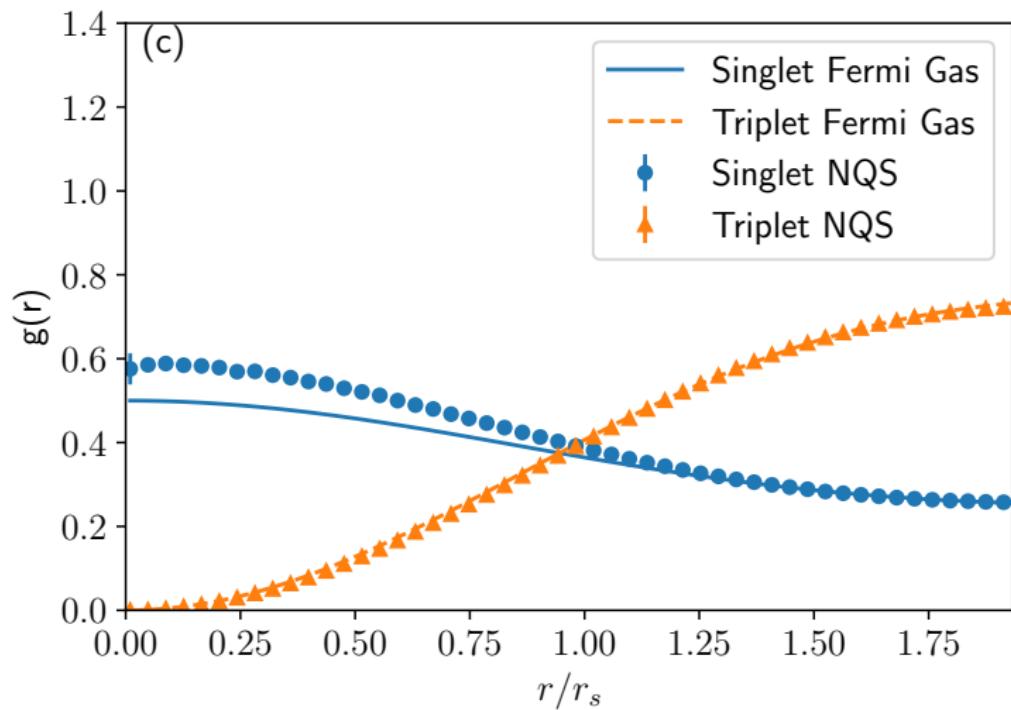
Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.01 \text{ fm}^{-3}$



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



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



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

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

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

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

The approximation theorem in words

Any continuous function $y = F(\mathbf{x})$ 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

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

Then we have

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

More on the general approximation theorem

None of the proofs give any insight into the relation between the number of hidden layers and nodes and the approximation error ϵ , nor the magnitudes of \mathbf{W} and \mathbf{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

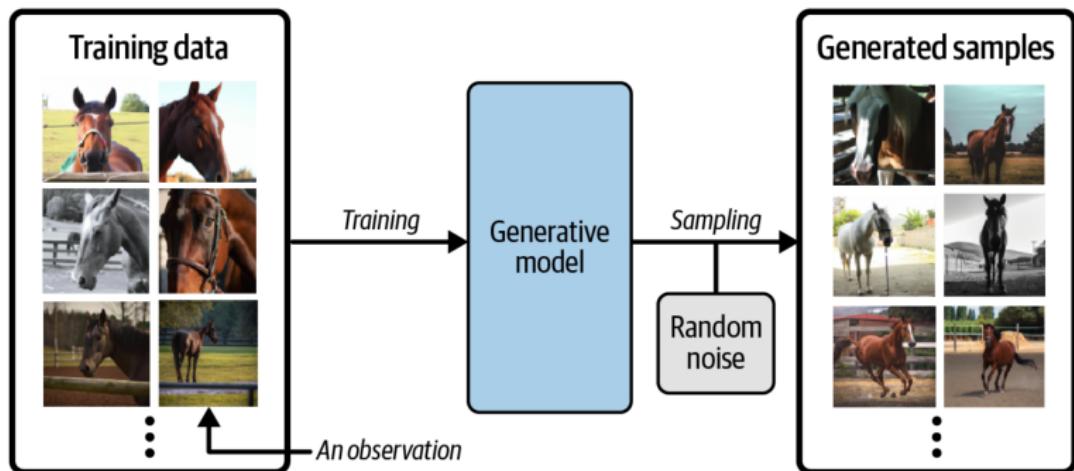
What Is Generative Modeling?

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.

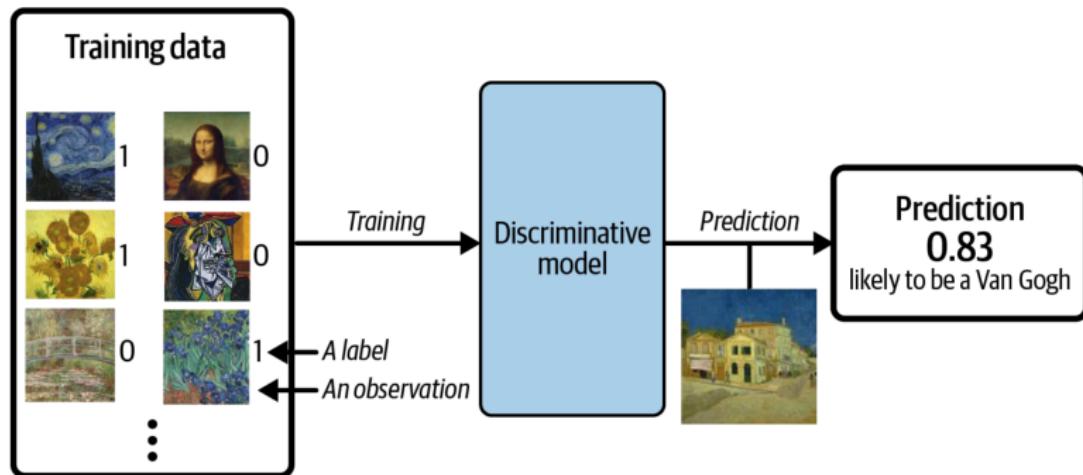
Each observation consists of many features. For an image generation problem, the features are usually the individual pixel values; for a text generation problem, the features could be individual words or groups of letters. It is our goal to build a model that can generate new sets of features that look 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 Deep Learning by David Foster

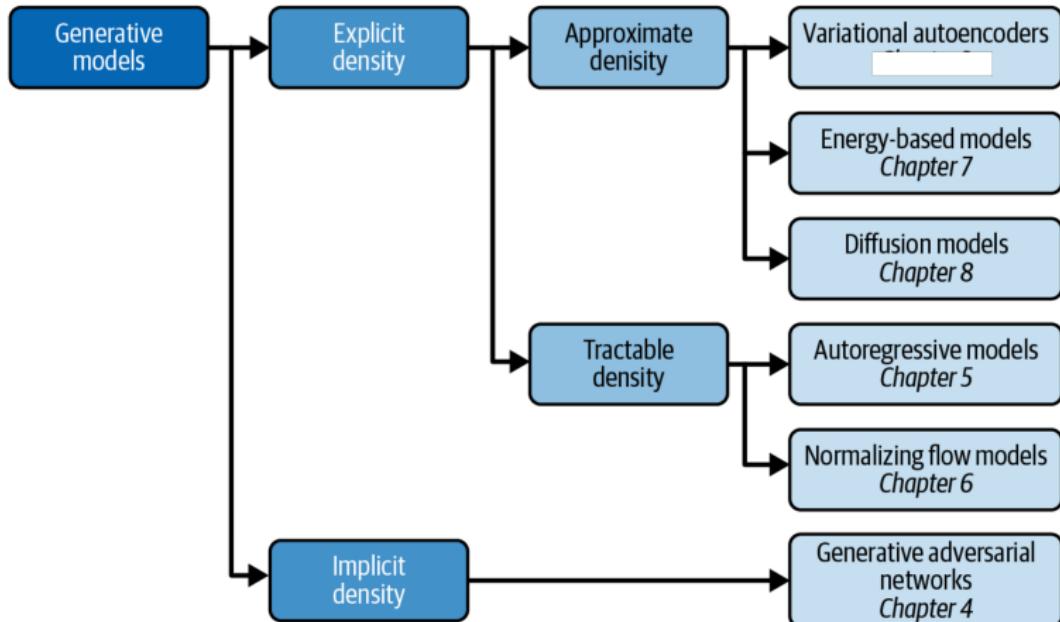


Discriminative Modeling

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 Raschka et al, Machine learning with Scikit-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 \mathbf{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 $p(\mathbf{x}|\boldsymbol{\alpha})$ or just a simple model $f(\boldsymbol{\alpha})$
- ▶ A so-called **loss/cost/risk** function $\mathcal{C}(\mathbf{x}, f(\boldsymbol{\alpha}))$ which allows us to decide how well our model represents the dataset.

We seek to minimize the function $\mathcal{C}(\mathbf{x}, f(\boldsymbol{\alpha}))$ 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 algorithms 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 $\mathbf{x}^T = [x_0, x_1, x_2, \dots, x_{n-1}]$ and the outputs we want to model $\mathbf{y}^T = [y_0, y_1, y_2, \dots, y_{n-1}]$. We assume that the output data can be represented (for a regression case) by a continuous function f through

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

Setting up the equations

In linear regression we approximate the unknown function with another continuous function $\tilde{y}(x)$ which depends linearly on some unknown parameters $\theta^T = [\theta_0, \theta_1, \theta_2, \dots, \theta_{p-1}]$.

The input data can be organized in terms of a so-called design matrix with an approximating function \tilde{y}

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

The objective/cost/loss function

The simplest approach is the mean squared error

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

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

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

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

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

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

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

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

Ridge and LASSO Regression

Our optimization problem is

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

or we can state it as

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

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

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

From OLS to Ridge and Lasso

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

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

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

Lasso regression

Defining

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

we have a new optimization equation

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

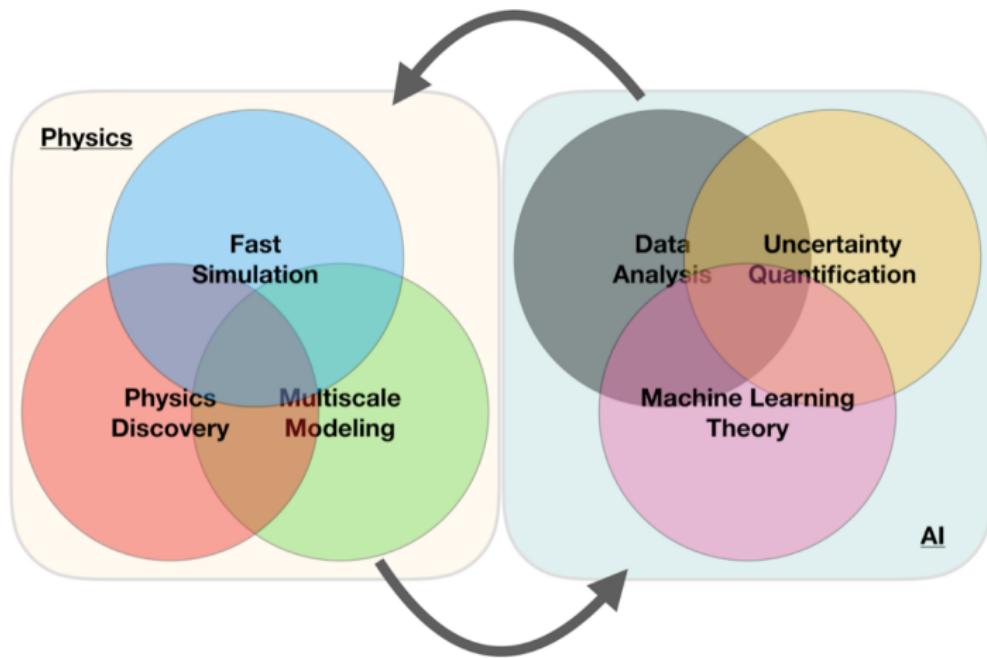
which leads to Lasso regression. Lasso stands for least absolute shrinkage and selection operator. Here we have defined the norm-1 as

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

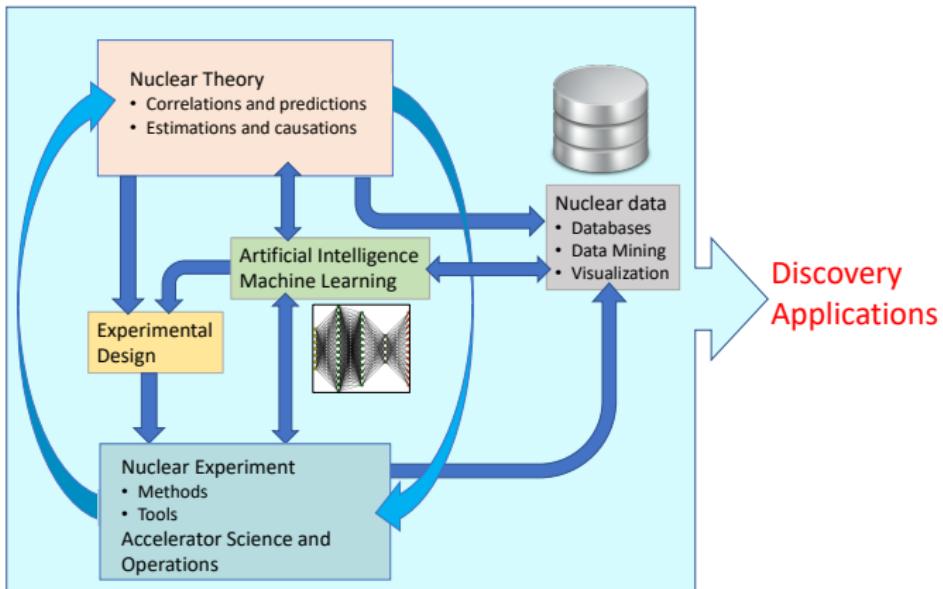
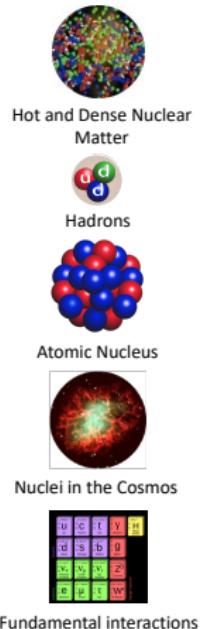
Selected references

- ▶ 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 Boehlein 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, 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)



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.



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

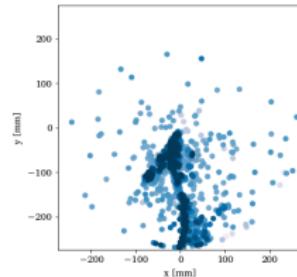
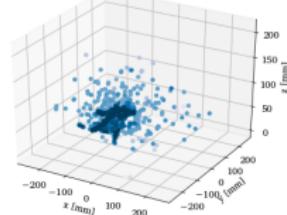
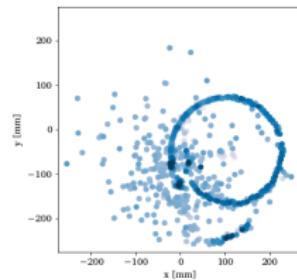
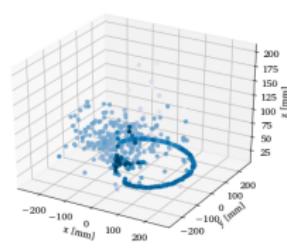
A less known but powerful result is that an NN with a single hidden layer can accurately approximate any nonlinear continuous operator. This universal approximation theorem of operators is suggestive of the structure and potential of deep neural networks (DNNs) in learning continuous operators or complex 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 probability 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 $r_{ij} = \sqrt{r_i^2 - r_j^2}$.

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 (RBM) 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.

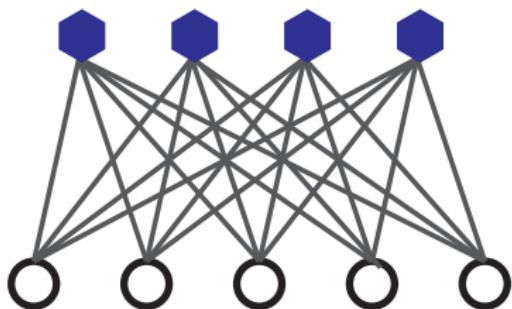
Furthermore, they have been used to solve complicated quantum mechanical many-particle problems or classical statistical physics problems like the Ising and Potts classes of models.

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

Joint distribution

The restricted Boltzmann machine is described by a Boltzmann distribution

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

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

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

Note the absence of the inverse temperature in these equations.

Network Elements, the energy function

The function $E(\mathbf{x}, \mathbf{h})$ gives the **energy** of a configuration (pair of vectors) (\mathbf{x}, \mathbf{h}) . The lower the energy of a configuration, the higher the probability of it. This function also depends on the parameters \mathbf{a} , \mathbf{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 $E(\mathbf{x}, \mathbf{h})$. The connection between the nodes in the two layers is given by the weights w_{ij} .

Binary-Binary RBM:

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

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

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

Gaussian binary

Gaussian-Binary RBM:

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

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

Representing the wave function

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

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

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

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

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

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

Define the cost function

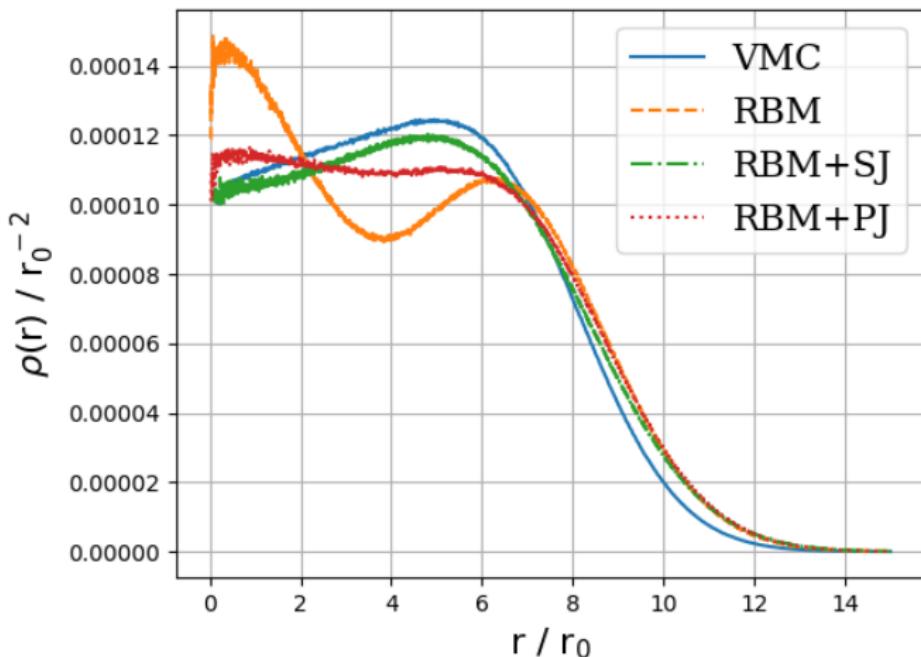
Now we don't necessarily have training data (unless we generate it by using some other method). However, what we do have is the variational principle which allows us to obtain the ground state wave function by minimizing the expectation value of the energy of a trial wavefunction (corresponding to the 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

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

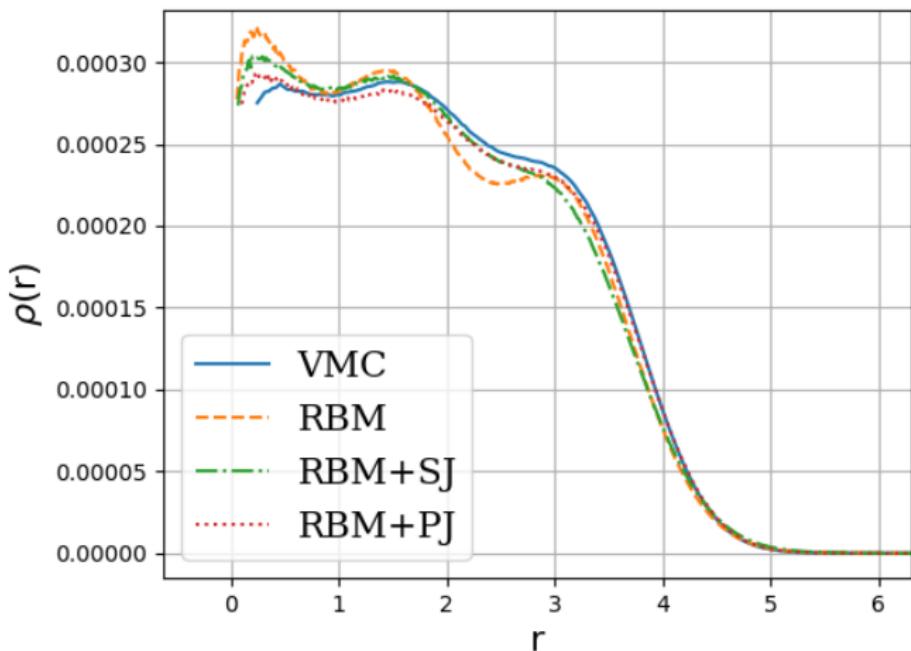
where the local energy is given by

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

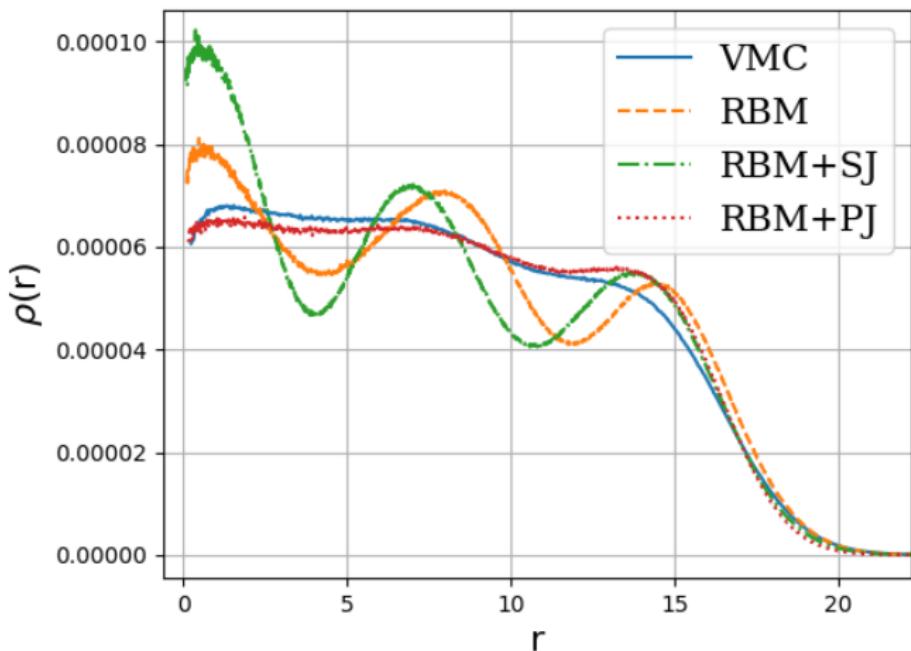
Quantum dots and Boltzmann machines, onebody densities
 $N = 6$, $\hbar\omega = 0.1$ a.u.



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

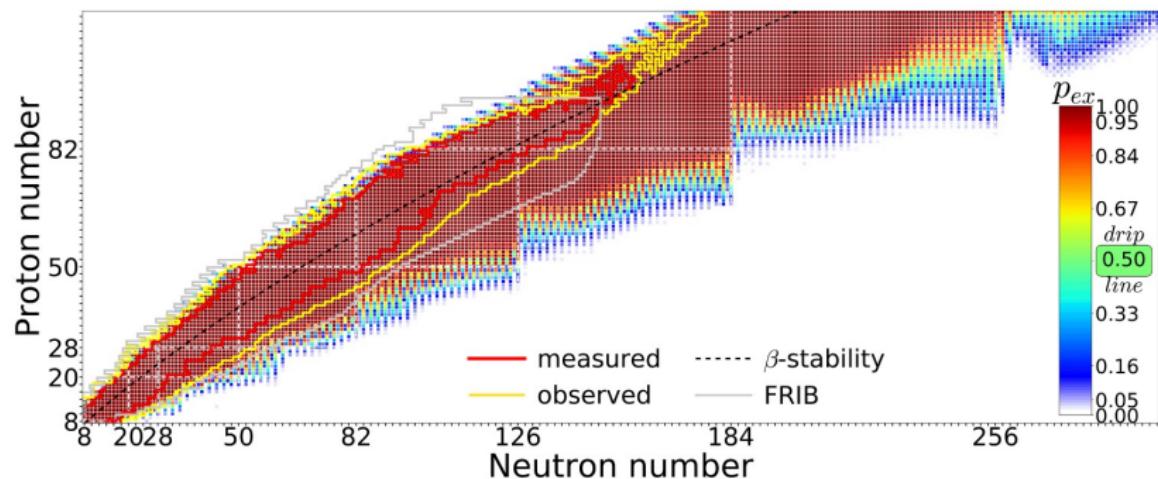
Bayes' theorem

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

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



Electrons on helium is another qubit platform

To our knowledge two qubit gates have never been discussed in a proper manner for these systems.

The static Coulomb interaction arises from a virtual photon exchange process between two charge particles according to quantum electrodynamics. This results in a correlated motion of two charges generating quantum entanglement.

Surface state electrons (SSE)

Surface state electrons (SSE) 'floating' above liquid helium originates from quantization of electron's perpendicular to the surface motion in a trapping potential formed by attractive force from image charge and a large ~ 1 eV barrier at the liquid-vacuum interface. At low temperatures the SSE are trapped in the lowest Rydberg state for vertical motion some 11 nm above the helium surface, which is perfectly clean and has a permittivity close to that of vacuum.

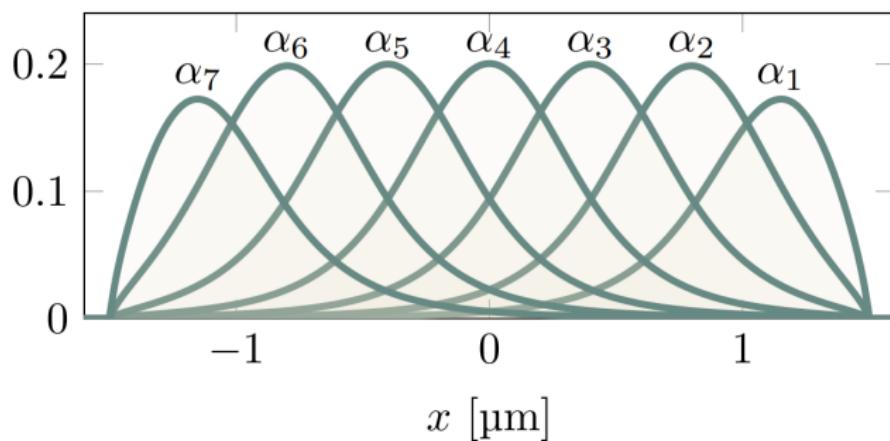
The weak interaction with the environment, which is mainly governed by interaction with quantized surface capillary waves (ripplons) and bulk phonons, ensures long coherence times - a vital ingredient for any qubit platform.

Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + \sum_{i=1}^7 V_i \alpha_i[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + \sum_{i=1}^7 V_i \alpha_i[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (2)$$

$$= h[\hat{p}_1, \hat{x}_1] + h[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (3)$$

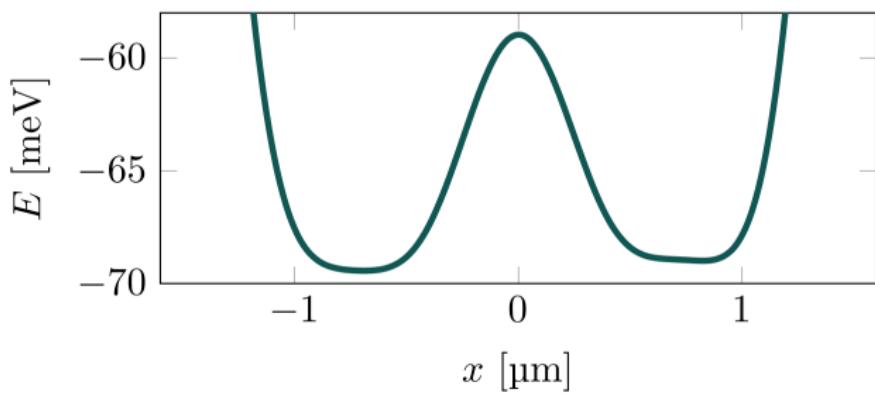


Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + v[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + v[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (4)$$

$$= h[\hat{p}_1, \hat{x}_1] + h[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (5)$$



Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + v[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + v[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (6)$$

$$\equiv h[\hat{p}_1, \hat{x}_1] + h[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (7)$$

Energy states:

$$|\Phi_k\rangle = \sum_{i < j} C_{ij,k} \frac{|\varphi_i \varphi_j\rangle - |\varphi_j \varphi_i\rangle}{\sqrt{2}}, \quad (8)$$

(Slater basis)

Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + v^L[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + v^R[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (9)$$

$$\equiv h^L[\hat{p}_1, \hat{x}_1] + h^R[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (10)$$

Energy states:

$$|\Phi_k\rangle = \sum_i \sum_j C_{ij,k} |\varphi_i^L \varphi_j^R\rangle, \quad (11)$$

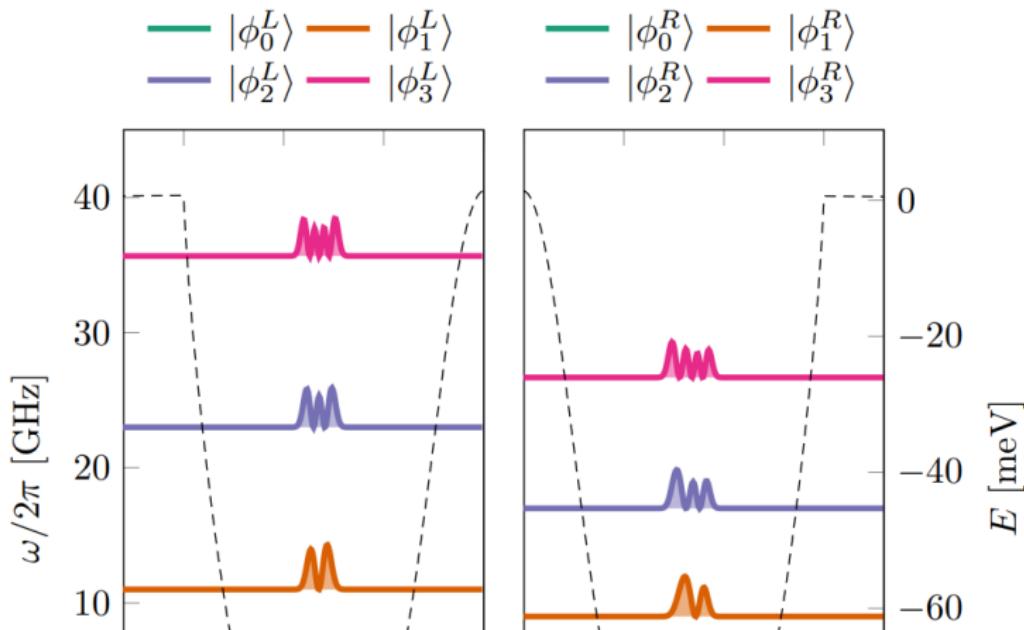
(product basis)

Calculational details

Energy states:

$$|\Phi_k\rangle = \sum_i \sum_j C_{ij,k} |\varphi_i^L \varphi_j^R\rangle, \quad (12)$$

(product basis)



Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + v^L[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + v^R[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (13)$$

$$\equiv h^L[\hat{p}_1, \hat{x}_1] + h^R[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (14)$$

Energy states:

$$|\Phi_k\rangle = \sum_i \sum_j C_{ij,k} |\varphi_i^L \varphi_j^R\rangle, \quad (15)$$

(Hartree basis)

Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + v^L[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + v^R[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (16)$$

$$\equiv h^L[\hat{p}_1, \hat{x}_1] + h^R[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (17)$$

Energy states:

$$|\Phi_k\rangle = \sum_{i=0}^{N^L} \sum_{j=0}^{N^R} C_{ij,k} |\varphi_i^L \varphi_j^R\rangle, \quad (18)$$

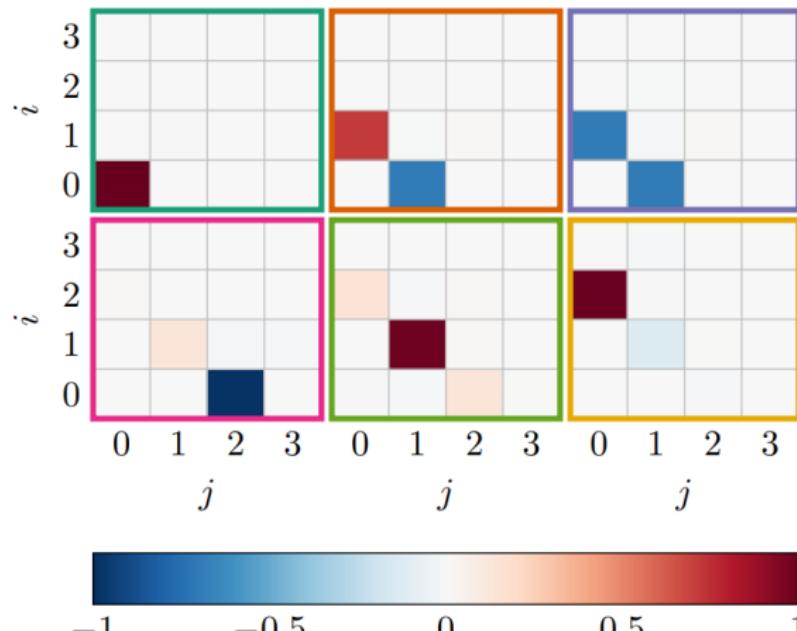
(Hartree basis)

Calculational details

Energy states:

$$|\Phi_k\rangle = \sum_{i=0}^{N^L} \sum_{j=0}^{N^R} C_{ij,k} |\varphi_i^L \varphi_j^R\rangle, \quad (19)$$

(Hartree basis)



Calculational details

Hamiltonian:

$$\hat{H} = \frac{\hat{p}_1^2}{2} + v^L[\hat{x}_1] + \frac{\hat{p}_2^2}{2} + v^R[\hat{x}_2] + \frac{\kappa}{\sqrt{(\hat{x}_1 - \hat{x}_2)^2 + a^2}} \quad (20)$$

$$\equiv h^L[\hat{p}_1, \hat{x}_1] + h^R[\hat{p}_2, \hat{x}_2] + u[\hat{x}_1, \hat{x}_2] \quad (21)$$

Energy states:

$$|\Phi_k\rangle = \sum_{i=0}^{N^L} \sum_{j=0}^{N^R} C_{ij,k} |\varphi_i^L \varphi_j^R\rangle, \quad (22)$$

(Hartree basis)

Results and discussions

By adjusting the potential we can change the anharmonicities and detuning of the wells.

1. What values of these give interesting interactions?
2. Inspiration from superconducting qubits, see High-Contrast ZZ Interaction Using Superconducting Qubits with Opposite-Sign Anharmonicity, Zhao et al Phys. Rev. Lett. 125, 200503

We search for well configurations corresponding to three different types of interaction between the two electrons.

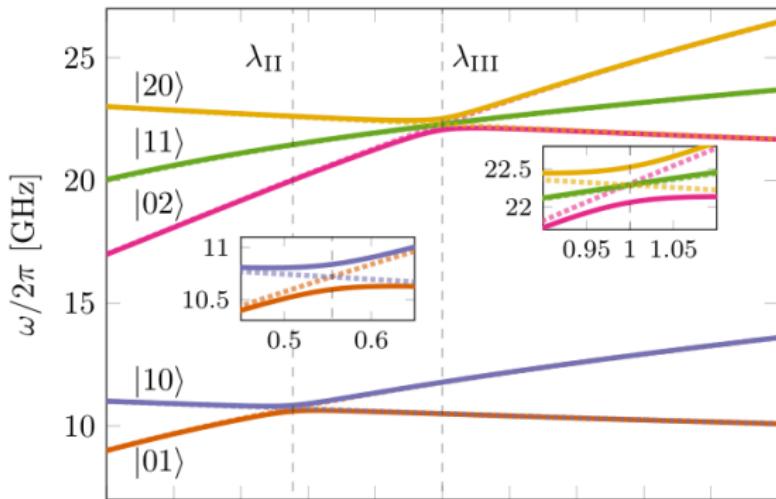
1. In configuration I we address both qubits independently and can thereby perform single-qubit state rotations and measurements.
2. Configurations II and III correspond to avoided level crossings between two (E_{01}, E_{10}) and three (E_{11}, E_{20}, E_{02}) energy levels respectively, where the electrons' motion becomes correlated, that is they are entangled.

Both anharmonicity and detuning changes with the shape of our well. We create a voltage parameterization

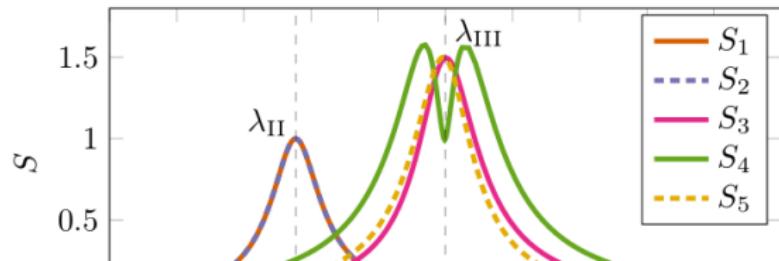
$$V(\lambda) = (1 - \lambda)V_I + \lambda V_{\text{III}} \quad (23)$$

Entanglement and more

(a)



(b)



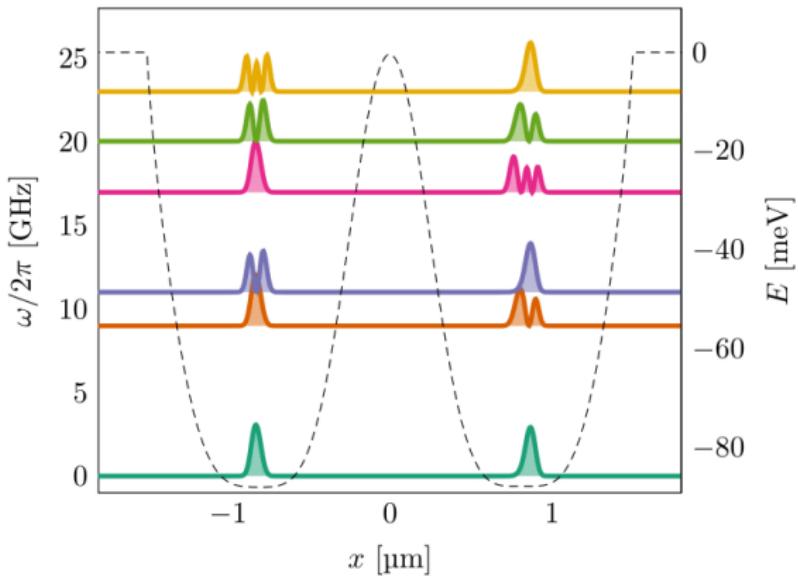
Legend to figure

1. (a) In this figure we have plotted the transition energy from the ground state to the labeled excited state as a function of the voltage parameter λ . The labeled states are the computational basis states when $\lambda = 0$.
2. (b) The von Neumann entropy of the five lowest excited states of the two-body Hamiltonian as a function of the configuration parameter λ . The ground state has zero entropy, or close to zero entropy. We have included the points for the double and triple degeneracy points. λ_{II} and λ_{III} in the figure. The von Neumann entropy is calculated using the binary logarithm.
3. (c) In this figure we have plotted the anharmonicites for the left well (α^L) and the right well (α^R) as a function of the well parameterization λ . We have also included the detuning $\Delta\omega = \omega^R - \omega^L$ between the two wells. We have marked configuration II at $\lambda_{II} \approx 0.554$ and configuration III at $\lambda_{III} = 1$.

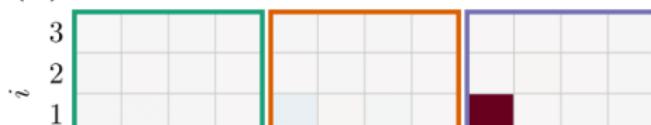
Particle densities and coefficients

— $|\Phi_0\rangle$ — $|\Phi_1\rangle$ — $|\Phi_2\rangle$
— $|\Phi_3\rangle$ — $|\Phi_4\rangle$ — $|\Phi_5\rangle$

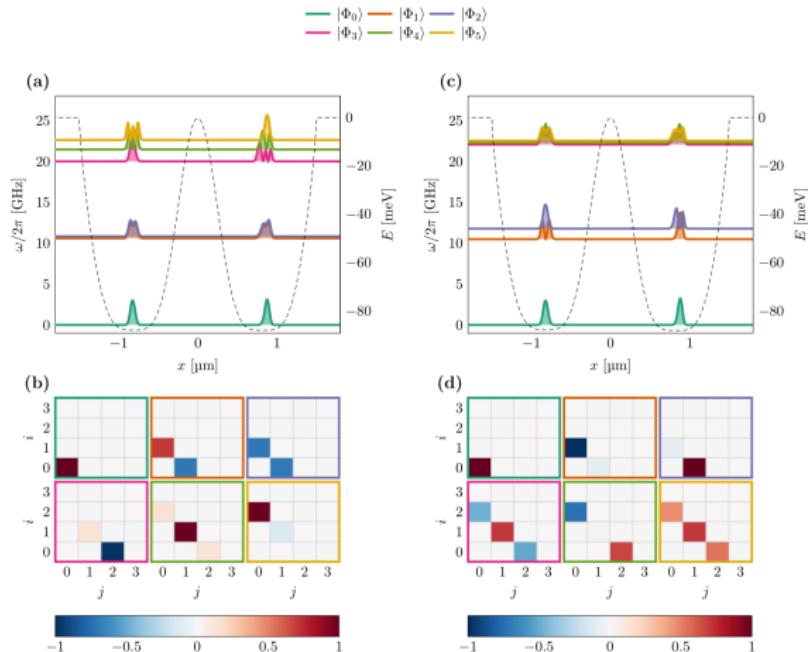
(a)



(b)



Potential wells, the one-body densities, and single-particle states



Where we are now

1. Adding time-dependent studies of two electrons in two wells in one and two dimensions
2. Studies of the time-evolution of entangled states (now two electrons only)
3. Use theory to find optimal experimental setup
4. Expect two-electron system realized experimentally in approx 1 year, great potential for studies of quantum simulations

Plans

1. Add two and three-dimensions in order to simulate in a more realistic way such many-body systems.
2. Develop time-dependent FCI code, useful up to approximately 10 particles with **effective** (and effective Hilbert space) Hamiltonians in two and three dimensions
3. Develop codes for studies of entanglement as function of time
4. Do tomography and extract density matrix and compare with experiment.
5. Study the feasibility of various setups for quantum simulations of specific Hamiltonians such as the Lipkin model
6. For larger many-body systems, study for example time-dependent CC theory

Addendum: Quantum Monte Carlo Motivation

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

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

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.

Quantum Monte Carlo Motivation

Basic steps

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

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

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

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

Quantum Monte Carlo Motivation

Define a new quantity

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

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

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

with N being the number of Monte Carlo samples.

The trial wave function

We want to perform a Variational Monte Carlo calculation of the ground state of two electrons in a quantum dot well with different oscillator energies, assuming total spin $S = 0$. Our trial wave function has the following form

$$\psi_T(\mathbf{r}_1, \mathbf{r}_2) = C \exp(-\alpha_1 \omega(r_1^2 + r_2^2)/2) \exp\left(\frac{r_{12}}{(1 + \alpha_2 r_{12})}\right), \quad (24)$$

where the variables α_1 and α_2 represent our variational parameters. Why does the trial function look like this? How did we get there? **This is one of our main motivations** for switching to Machine Learning.

The correlation part of the wave function

To find an ansatz for the correlated part of the wave function, it is useful to rewrite the two-particle local energy in terms of the relative and center-of-mass motion. Let us denote the distance between the two electrons as r_{12} . We omit the center-of-mass motion since we are only interested in the case when $r_{12} \rightarrow 0$. The contribution from the center-of-mass (CoM) variable \mathbf{R}_{CoM} gives only a finite contribution. We focus only on the terms that are relevant for r_{12} and for three dimensions. The relevant local energy operator becomes then (with $l = 0$)

$$\lim_{r_{12} \rightarrow 0} E_L(R) = \frac{1}{\mathcal{R}_T(r_{12})} \left(-2 \frac{d^2}{dr_{ij}^2} - \frac{4}{r_{ij}} \frac{d}{dr_{ij}} + \frac{2}{r_{ij}} \right) \mathcal{R}_T(r_{12}).$$

In order to avoid divergencies when $r_{12} \rightarrow 0$ we obtain the so-called **cusp** condition

$$\frac{d\mathcal{R}_T(r_{12})}{dr_{12}} = \frac{1}{2} \mathcal{R}_T(r_{12}) \quad r_{12} \rightarrow 0$$

Resulting ansatz

The above results in

$$\mathcal{R}_T \propto \exp(r_{ij}/2),$$

for anti-parallel spins and

$$\mathcal{R}_T \propto \exp(r_{ij}/4),$$

for anti-parallel spins. This is the so-called cusp condition for the relative motion, resulting in a minimal requirement for the correlation part of the wave function. For general systems containing more than say two electrons, we have this condition for each electron pair ij .

Energy derivatives

To find the derivatives of the local energy expectation value as function of the variational parameters, we can use the chain rule and the hermiticity of the Hamiltonian.

Let us define (with the notation $\langle E[\alpha] \rangle = \langle E_L \rangle$)

$$\bar{E}_{\alpha_i} = \frac{d\langle E_L \rangle}{d\alpha_i},$$

as the derivative of the energy with respect to the variational parameter α ; We define also the derivative of the trial function (skipping the subindex T) as

$$\bar{\Psi}_i = \frac{d\Psi}{d\alpha_i}.$$

Derivatives of the local energy

The elements of the gradient of the local energy are then (using the chain rule and the hermiticity of the Hamiltonian)

$$\bar{E}_i = 2 \left(\langle \frac{\bar{\Psi}_i}{\Psi} E_L \rangle - \langle \frac{\bar{\Psi}_i}{\Psi} \rangle \langle E_L \rangle \right).$$

From a computational point of view it means that you need to compute the expectation values of

$$\langle \frac{\bar{\Psi}_i}{\Psi} E_L \rangle,$$

and

$$\langle \frac{\bar{\Psi}_i}{\Psi} \rangle \langle E_L \rangle$$

These integrals are evaluated using MC integration (with all its possible error sources). We can then use methods like stochastic gradient or other minimization methods to find the optimal variational parameters (I don't discuss this topic here, but these

How do we define our cost function?

We have a model, our likelihood function.
How should we define the cost function?

Meet the variance and its derivatives

Why the variance?

Suppose the trial function (our model) is the exact wave function.
The action of the hamiltonian on the wave function

$$H\Psi = \text{constant} \times \Psi,$$

The integral which defines various expectation values involving moments of the hamiltonian becomes then

$$\langle E^n \rangle = \langle H^n \rangle = \frac{\int d\mathbf{R} \Psi^*(\mathbf{R}) H^n(\mathbf{R}) \Psi(\mathbf{R})}{\int d\mathbf{R} \Psi^*(\mathbf{R}) \Psi(\mathbf{R})} = \text{constant} \times \frac{\int d\mathbf{R} \Psi^*(\mathbf{R}) \Psi(\mathbf{R})}{\int d\mathbf{R} \Psi^*(\mathbf{R}) \Psi(\mathbf{R})}$$

This gives an important information: If I want the variance, the exact wave function leads to zero variance! The variance is defined as

$$\sigma_E = \langle E^2 \rangle - \langle E \rangle^2.$$

Variation is then performed by minimizing both the energy and the variance.

The variance defines the cost function

We can then take the derivatives of

$$\sigma_E = \langle E^2 \rangle - \langle E \rangle^2,$$

with respect to the variational parameters. The derivatives of the variance can then be used to define the so-called Hessian matrix, which in turn allows us to use minimization methods like Newton's method or standard gradient methods.

This leads to however a more complicated expression, with obvious errors when evaluating integrals by Monte Carlo integration. Less used, see however [Filippi and Umrigar](#). The expression becomes complicated

$$\begin{aligned} \bar{E}_{ij} = & 2 \left[\left\langle \left(\frac{\bar{\Psi}_{ij}}{\Psi} + \frac{\bar{\Psi}_j}{\Psi} \frac{\bar{\Psi}_i}{\Psi} \right) (E_L - \langle E \rangle) \right\rangle - \left\langle \frac{\bar{\Psi}_i}{\Psi} \right\rangle \bar{E}_j - \left\langle \frac{\bar{\Psi}_j}{\Psi} \right\rangle \bar{E}_i \right] \quad (25) \\ & + \left\langle \frac{\bar{\Psi}_i}{\Psi} E_{Lj} \right\rangle + \left\langle \frac{\bar{\Psi}_j}{\Psi} E_{Li} \right\rangle - \left\langle \frac{\bar{\Psi}_i}{\Psi} \right\rangle \langle E_{Lj} \rangle \left\langle \frac{\bar{\Psi}_j}{\Psi} \right\rangle \langle E_{Li} \rangle. \end{aligned}$$

Evaluating the cost function means having to evaluate the above