Theoretical Physics in an era of Machine Learning

Frontiers of Physics - News from the NBIA
Evert van Nieuwenburg
evert.vn@nbi.ku.dk
Theoretical Physics in an era of Machine Learning

Frontiers of Physics – News from the NBIA

Please feel free to ask questions!
Some words about me

Leiden University, Netherlands → ETH Zürich → California Institute of Technology (Caltech)

My Trajectory On Google Earth
The content of this lecture

What is Machine Learning? An introduction by example
How can it help research in physics?

TiqTaqToe

Bonus: Quantum Games
Machine Learning is fun

Locomotion

deeplmind.com/blog/article/producing-flexible-behaviours-simulated-environments
Machine Learning is fun

Multi-Agent Hide & Seek

Locomotion

https://openai.com/blog/emergent-tool-use/

deepmind.com/blog/article/producing-flexible-behaviours-simulated-environments
Machine Learning is everywhere

Natural Language Processing
https://talktotransformer.com/

Generative Modelling
https://thispersondoesnotexist.com/

Speech Synthesis
“Deep-Fakes”
**Machine Learning** is a different way of solving problems

(An example of “supervised learning” - more in a few slides!)

---

**The ML Approach**

- **FUNCTION IS_CAT?:**
  - IF( HAS_WHISKERS ):
    - IF( HAS_CAT_EYES ):
      - IF( HAS_NO_MANE ):
        - IF( . . . ):
          - RETURN TRUE
    - ELSE:
      - RETURN FALSE
- TRUE
- FALSE

**The Human Approach**

- TRUE
- FALSE

The main component in many ML techniques is a neural network.
The main component in many ML techniques is a neural network.
The main component in many ML techniques is a neural network.

Cat recogniser network

Repeat for all cats, tweaking the knobs so that we recognize all of them! This is ‘learning’
This is how a simple NN works

300*200 = 60000 inputs
This is how a simple NN works

\[ 0.31 \times w_1 + 0.33 \times w_2 + 0.28 \times w_3 + \ldots \]

300*200 = 60000 inputs
This is how a simple NN works

300*200 = 60000 inputs

= \tanh(0.31 * w_1 + 0.33 * w_2 + 0.28 * w_3 + \ldots )
This is how a simple NN works

The knobs from before are the w's (the weights)
This is the entire essence of (artificial) neural networks

A Neural Network is a highly non-linear, parameterised function $y(x, \{w\})$.
Learning = updating the weights to **minimise the loss-function**

<table>
<thead>
<tr>
<th>Input</th>
<th>Answer/Target</th>
<th>Network Prediction</th>
<th>Network Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Cat" /></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><img src="image2.png" alt="Dog" /></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><img src="image3.png" alt="Cat" /></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><img src="image4.png" alt="Puppy" /></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ L \sim \sum (y_{\text{true}} - y_{\text{predicted}})^2 \]

Can be changed with the weights/knobs!
A workflow for neural networks looks like this:

Step 1. Given: A dataset with many inputs $x$ and corresponding outputs $y_{true}(x)$

$$\mathcal{L}(w, \{x\}) = \sum_{x \in \{x\}} \left( y_{true}(x) - y_{network}(w, x) \right)^2$$

Gradient Descent / Backpropagation $w \rightarrow w - \nabla_w \mathcal{L}$

Step 2

Step 3. Generalize. Use the network to infer (predict) the right output for new inputs
Neural Networks come in many topologies

https://www.asimovinstitute.org/neural-network-zoo/


Deep Feed Forward (DFF)

Auto Encoder (AE)

Restricted BM (RBM)

Output = Input

"Distance" between prob. distributions
A quick summary

A Neural Network is a machine with many parameters (knobs), that we can tune (train) so that it reproduces the answers we want.

Given enough parameters, we can fit any function we want.

(For example, we can fit the ‘is_this_a_cat?’ function)
(Or, we can fit a ‘turn_random_noise_into_a_face’ function)
(... etc)
There are roughly **three types of ML**

**Supervised Learning**
Learning from examples

\[
\{x_1, y_1\} = \{\begin{array}{c}
\text{cat image}, \\
1
\end{array}\}
\]

\[
\{x_2, y_2\} = \{\begin{array}{c}
\text{dog image}, \\
0
\end{array}\}
\]

Learn \(p(y \mid x)\)
Classification

**Unsupervised Learning**
Learning about examples

\[
x_1 = \begin{array}{c}
\text{cat image}
\end{array}
\]

\[
x_2 = \begin{array}{c}
\text{dog image}
\end{array}
\]

Learn \(p(x)\)
(Draw samples to generate!)

**Reinforcement Learning**
Learning from feedback

Learn a **policy**, (best action in a given state \(s\))

Learn \(\pi(s)\)
Sutton&Barto
Each of these types has a use in physics

**Supervised Learning**  
Learning from examples
- Picture of a galaxy -> which type?
- LHC collisions -> which particles?  
  LHC -> interesting collision?
- Material -> superconductor?  

Learn $p(y|x)$  
Classification

**Unsupervised Learning**  
Learning about examples
- Generate more superconductors?  
- Material -> superconductor?  

Learn $p(x)$  
(Draw samples to generate!)

**Reinforcement Learning**  
Learning from feedback
- Correct errors in a quantum computer  
- Material -> superconductor?  

Learn $\pi(s)$  
Sutton&Barto

Run EXPERIMENTS!
Condensed Matter Physics

Studies properties of matter

How well does a piece of metal conduct?
Why are metals shiny?
How do superconductors work?
How does an insulator work?
How do we make a quantum computer?
At what temperature does a magnet stop working?

…
This is how we use **ML for Quantum Physics**

**Supervised Learning**
Finding Phase Transitions
Enhancing experiments

**Unsupervised Learning**
Wavefunction Neural Network
Quantum State Reconstruction

**Reinforcement Learning**
Correcting a quantum computer
Controlling experiments

Ψ =

www.netket.org
This is how we use **ML** for **Quantum Physics**

**Supervised Learning**
- Finding Phase Transitions
- Enhancing experiments
Supervised learning can be used to find phase transitions.

$$p(T \ll T_c) = 1$$
$$p(T \gg T_c) = 0$$
Supervised learning can be used to find phase transitions.

- For $T \ll T_c$: $p(T \ll T_c) = 1$, $p(T \gg T_c) = 0$
- For $T \gg T_c$: $p(T \ll T_c) = 0$, $p(T \gg T_c) = 1$
- For $T \ll T_c$: $p(T \ll T_c) = 0.8$, $p(T \gg T_c) = 0.2$
- For $T \gg T_c$: $p(T \ll T_c) = 0.2$, $p(T \gg T_c) = 0.8$
Supervised learning can be used to find phase transitions.

$p(T \gg T_c) = 0$
Supervised learning can make some experiments 100x faster

At QDev, Ferdinand Kuemmeth works on qubits

Quantum bit: can be part 0 and part 1 simultaneously

Reading out the qubit state: electric signals

The signal is then demodulated using techniques that are rooted in old radio-technology

Takes 10-100 microseconds!

Raw data -> ~100 nanoseconds!
This is how we use **ML** for Quantum Physics

**Supervised Learning**
Finding Phase Transitions

**Unsupervised Learning**
Wavefunction Neural Network

Quantum State Reconstruction

\[ \Psi = \]

Enhancing experiments

[www.netket.org](http://www.netket.org)
Unsupervised learning can do Quantum State Reconstruction

An experiment with 4 two-level atoms (each can be in quantum state 0 or 1)

The full system can be a superposition of all 16 possible states

<table>
<thead>
<tr>
<th>State 1</th>
<th>State 2</th>
<th>State 3</th>
<th>State 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 1</td>
<td>0 0 1 0</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The full state of this system is called the wavefunction $\Psi(a_1, a_2, a_3, a_4)$
Unsupervised learning can do *Quantum State Reconstruction*

An experiment with 4 two-level atoms (each can be in quantum state 0 or 1)

Perform the experiment many times, and record which configuration we get

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

*Projective measurements*

Every time we look, we force the atoms to choose; Schrödinger's cat

Question: can we learn $\Psi(a_1, a_2, a_3, a_4)$?
Unsupervised learning can do Quantum State Reconstruction

Histogram does not scale!
Exponential number of measurements!

\[ 2^4 = 16 \quad 2^{64} > \text{grains of sand on earth} \]

\[ \Psi = \]

Trade a complex (impossible) measurement for many simple ones
This is how we use ML for Quantum Physics

**Supervised Learning**
Finding Phase Transitions
Enhancing experiments

**Unsupervised Learning**
Wavefunction Neural Network
Quantum State Reconstruction

**Reinforcement Learning**
Correcting a quantum computer
Controlling experiments

$\Psi = \text{Netket.org}$
Reinforcement learning is about strategies (policies)

MuZero (the successor to AlphaZero and AlphaGo)
A simple quantum computer with one **logical qubit**

Vertex = physical qubit
A simple quantum computer with one logical qubit

The full state of all these qubits itself represents a “logical” qubit

Simplification: all qubits 0 -> logical 0, all qubits 1 -> logical 1
Qubit errors show up as **red dots** on the orange squares

If this qubit flips from 0 to 1 …
Qubit errors show up as red dots on the orange squares.

We can *not* look at the flipped qubits, only the red dots!
Multiple errors cause red dots to change position.

Only an odd number of red flags is visible.
Red dots can *(dis)appear* at the edges

If this qubit flips from 0 to 1 too …
If a string of errors connects the edges, it is impossible to find out which qubits had errors.

A whole column of qubits has errors, but we can’t see it!
So quantum error correction is like a board game!

Given red dots, find out which qubits flipped (the errors)

*Game over if a string of errors connects one edge to the other*
Reinforcement learning can do **quantum error correction**

Use a neural network to determine which qubits flipped, given the current red dots.
These methods are not mutually exclusive

A glimpse of the future?

Control experiments:
1. Perform an initial scan
2. Center bias triangles
3. Score transport measurement
4. Determine new gate voltages

Design experiments


Melnikov et al, PNAS February 6, 2018 115 (6) 1221-1226
When does an ML approach make sense in physics?

- **Accuracy**
  Only ever as accurate as the data (unless ML can also request new data?)

- **Speed**
  Predict ground state properties vs Monte Carlo, or learn a quantum state

- **Experimental/Computational Cost**

- **Adaptiveness**
  If qubit number 37 happens to be worse than the others, ML will learn that
6.11 Quantum games

Advances in classical computing launched a new world of digital games, touching the lives of millions and generating billions in revenue. Could quantum computers do the same?

Physicists often say that the quantum world is counter-intuitive because it is so foreign to ordinary experience. That’s true now, but might it be different in the future? Perhaps kids who grow up playing quantum games will acquire a visceral understanding of quantum phenomena that our generation lacks. Furthermore, quantum games could open a niche for quantum machine learning methods, which might seize the opportunity to improve game play in situations where quantum entanglement has an essential role.
Quantum Games

Quantum TiqTaqToe

www.quantumtictactoe.com
https://quantumfrontiers.com/2019/07/15/tiqtqtoe/

www.quantumchess.net
Quantum Games

"Anyone can Quantum"
Thank you for participating!