DHBW
Duale Hochschule
Baden-Württemberg

Optimization and Machine Learning with Quantum Computers

Prof. Dr. Gerhard Hellstern
DHBW Stuttgart - Center of Finance

Overview

1. Applied Quantum Computing
2. Circuit Model & Programming a quantum computer
3. Optimization using quantum computers
4. Quantum Machine Learning
5. Resources: Literature & MooCs

1. APPLIED QUANTUM COMPUTING

QUANTENCOMPUTER

Wenn komplexe Aufgaben in 3:20 Minuten statt 10.000 Jahren gelöst werden

Auf dem Forschungsfeld der Quantencomputer könnte Google ein Durchbruch gelungen sein. Experten erwarten einen Schub für die Zukunftstechnologie.

Christof Kerkmann Axel Postinett

01.10.2019 - 17:53 Uhr • 1 Kommentar • 5 x geteilt

Quantencomputer „Q System One"

Amerikanische Unternehmen wie IBM forschen intensiv an der Technologie.
(Foto: dpa)

Düsseldorf, San Francisco. Der große Durchbruch oder die große

Google:
"Quantum Supremacy "

NewScientist

Sign in Enter search keywords

News Features Newsletters Podcasts Video Comment Culture Crosswords | This week's magazine

Health Space Physics Technology Environment Mind Humans Life Mathematics Chemistry Earth Society

Technology

Google's claim of quantum supremacy has been completely smashed

Google's Sycamore quantum computer was the first to demonstrate quantum supremacy – solving calculations that would be unfeasible on a classical computer – but now ordinary machines have pulled ahead again

By Matthew Sparkes

3 July 2024


XANADU
D-Wave
The Quantum Computing Company
NTT AT
Q Branch
ProteinQure
AIRBUS
accenture
BOHR
1QBit
Delft Circuits
ZAPATA
RIVER LANE
NOKIA
AT&T
EeroQ
Baidu 百度
imec
Booz | Allen | Hamilton
Q-CTRL
HUAWEI
HRL LABORATORIES
BraneCell
ZEISS
hp
STRANGE WORKS
rigetti
Atos
Honeywell
TOSHIBA
Google
LOCKHEED MARTIN
SIEMENS
SAMSUNG
intel
Microsoft
IBM
Alibaba.com
RIKEN
DAIMLER

Location determination …

- Hype about quantum computing for about 6-7 years; although work on it has been going on for many years
- Google's Supremacy debate has made the topic widely known
- In addition to Google, Microsoft, IBM, Baidu and a whole series of specialist companies (e.g. D- wave or Rigetti) are also involved
- Companies from the areas of mobility (Daimler, VW, BMW, ..), logistics, pharmaceuticals and materials research (BASF, ...) as well as finance (Commerzbank, Barclays ...) are actively dealing with the "topic"

…. and theses derived from it (I)

- Quantum computing COULD replace “classical” computers for certain tasks in just a few years
- From a theoretical point of view (complexity theory), a so-called “speed-up” compared to classical algorithms may be achieved for certain types of problems:
  - Certain optimization problems
  - Certain simulation problems
  - Certain machine learning problems

Which exactly is the subject of current research…

Type of scaling	Time to solve problem				
Classical algorithm with exponential runtime	10 secs	2 mins	330 years	3300 years	Age of the universe
Quantum algorithm with polynomial runtime	1 min	2 mins	10 mins	11 mins	~24 mins

Other “hopes” are higher accuracy, etc.

…. and theses derived from it (II)

- Difficulty = Excitement: Programming quantum computers is “completely different” ☺
  - You start (currently) “at the very bottom” of the software stack → qubits instead of bits
  - Starting from qubits, the stack must be fundamentally redefined

→ Need for skilled workers with different qualifications (physicists, engineers, etc.)

→ **But also to computer scientists who “can” do something like that** … (especially in Germany)

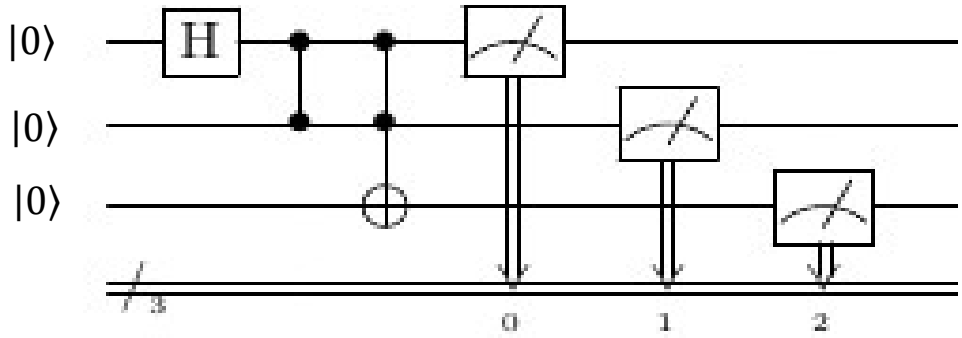
→ … and who also understand the application side and master the combination of classical computing & quantum computing

→ Need for “knowledge in companies” on this topic

2. CIRCUIT & PROGRAMMING A QUANTUM COMPUTER

The Circuit Model

Several qubits are the input for a quantum circuit:



- n qubits go in (in the picture: n=3): $|0\rangle|0\rangle|0\rangle = |000\rangle$
- n qubits also go out again
- Quantum gates acting on one or more qubits are applied
- it is measured → result is a bit string; e.g. 101
- After measurement state changed to e.g. $|1\rangle|0\rangle|1\rangle = |101\rangle$
- Attention: There are no loops and all steps up to the measurement are reversible!
- Nevertheless, “universal computation model” in the sense of a classical Turing machine

The circuit model

- General parameterization of the one-qubit gates:

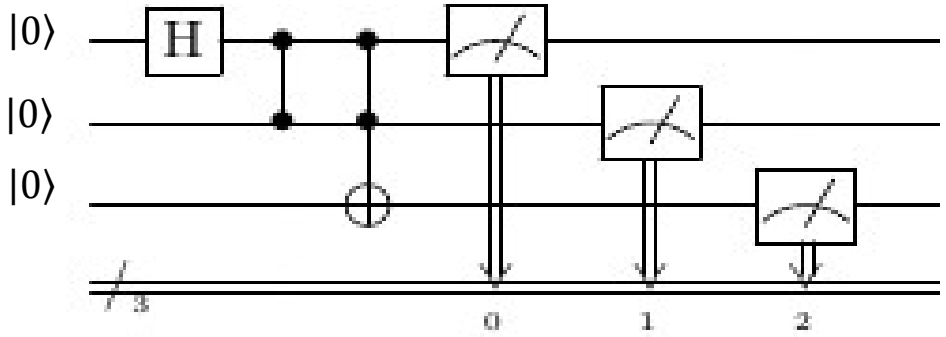
$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\phi+i\lambda}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

realized by choosing the appropriate angles θ, ϕ, λ ☺

- Spoiler: Consider these angles as parameters that can be “trained” → correspond to the parameters of a neuron → “ Variational Quantum Algorithms” for Quantum Machine Learning

Quantum software

- Programming a quantum computer corresponds to the “manipulation” of qubits
- Realized by applying gates to one or more qubits
- Quantum algorithm: sequence of gates and subsequent measurement:



- Is a quantum computer absolutely necessary for this? No!

→ Calculation of a quantum algorithm corresponds essentially to linear algebra

→ Also feasible on classical computers … at least up to approx. 25 qubits ☺

Quantum software

- However, the size of the matrices increases exponentially with 2^n
- For n=50 qubits : $2^{50} = 1125899906842624 \approx 1{,}12 \cdot 10^{15}$
- A real quantum advantage is expected especially from n > 50 AND corresponding depth of the circuit

- Designing quantum software with two backends :
  - For $n < 25 - 30$: Classic Calculators
    - Calculating the exact results of an algorithm using linear algebra
    - Simulation of a quantum experiment as a random experiment
  - quantum computer with the corresponding number of qubits
    - Physical manipulation and subsequent measurement of the qubits

- Comparison of results for $n < 20 - 30$ allowing estimation of quality

Quantum software

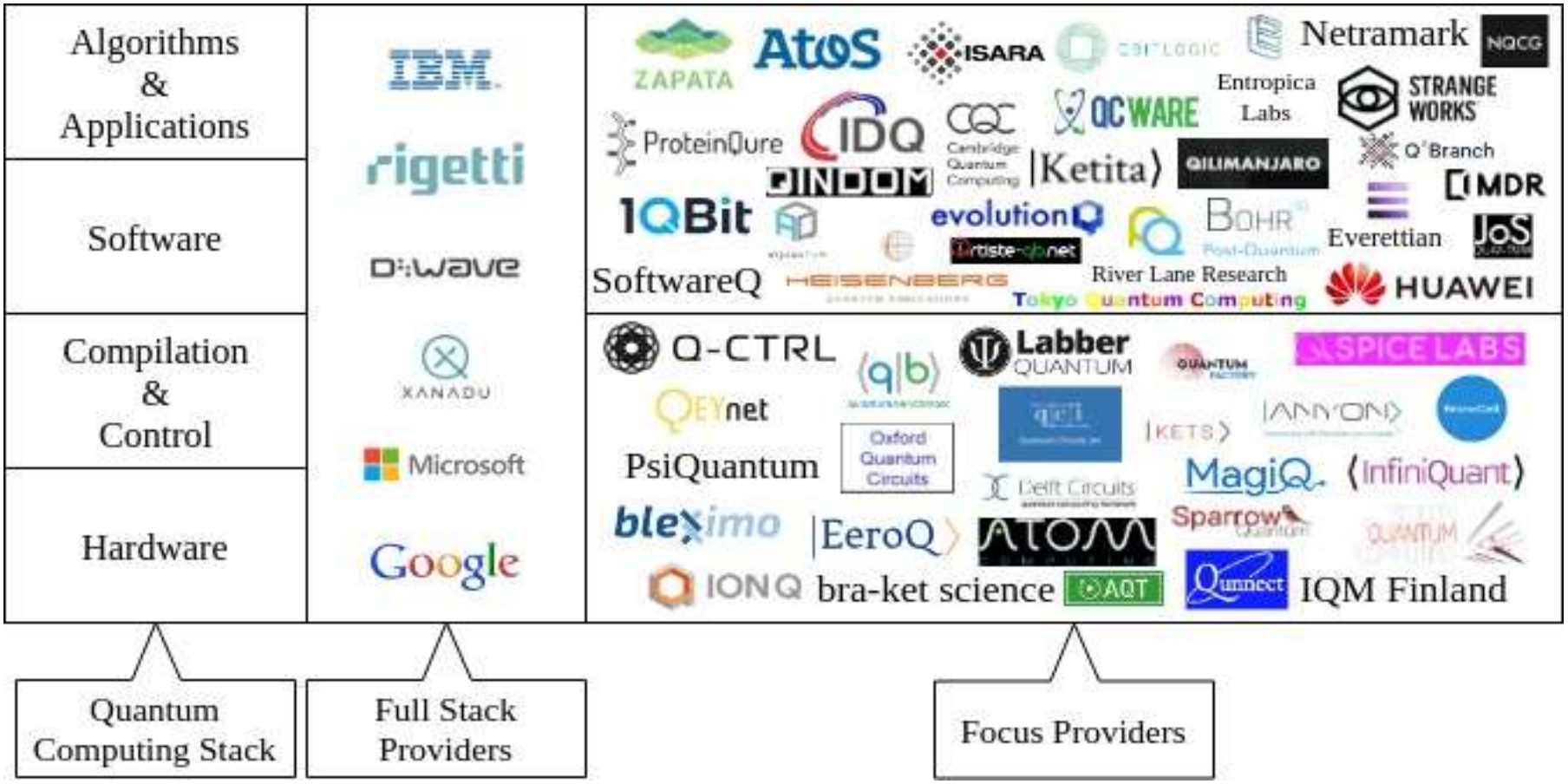
- But the quantum computers from IBM and other manufacturers already have about 100 qubits!!??

→ But these are not “logical” but “only” physical qubits because they are flawed:
  → “ decoherence ” through interaction with the environment
  → eg $|1\rangle \rightarrow |0\rangle$ or $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \rightarrow |0\rangle$

→ Limitation to “ shallow ” (not too deep circuits) with the hope that despite errors the results are still usable …or:

→ **Correcting the errors** - due to the no-cloning theorem - is much more complex here than in classical computing ☹
  → Goal: Reduction of the number of physical qubits required for a logical qubit
  → Very active field of research!!
→ Here: We assume we have any number of error-free qubits ☺

quantum software



https://quantumcomputingreport.com/review-of-the-cirq-quantum-software-framework

Quantum Software – Qiskit from IBM

In the following, IBM’s Qiskit is used to demonstrate the algorithms :

- Qiskit is open source and free to use
- Programming locally (own hardware) – Controlling physical hardware via API
- Programming in Python
- In addition to exact calculation and simulation, real quantum computers with up to 100+x qubits are also available
- Online tutorials and material for beginners & specialists
- World's largest community of "quantum enthusiasts" → support and exchange
- Qiskit is also used in the corporate context

Quantum Software – PennyLane by Xanadu

Alternative for Quantum Machine Learning: **PennyLane**

- Xanadu is a Canadian quantum startup that builds photonic quantum hardware in addition to software
- PennyLane is “hardware-agnostic”, i.e. compatible with quantum hardware from all common vendors – as well as extremely high-performance simulators (e.g. on Nvidia GPUs)
- Easy integration of Tensorflow and Pytorch as ML frameworks
- Out-of -the-box implementation of basic components of quantum circuits (e.g. for data embedding)
- Very close involvement in scientific research
- Good training material, as well as demo notebooks for typical use cases or new ideas

4. OPTIMIZATION USING QUANTUM COMPUTERS

Quantum algorithms for optimization

- Not all optimization problems are suitable for quantum algorithms….
- … the best ones are those with **binary** variables
- Why? Bits 0 and 1 become qubits $|0\rangle$and$|1\rangle$
- … and where the objective function is, for example, quadratic → so-called **QUBO**
- Practical approach:
  - Start with a **relevant** problem → Does it have this form?
  - “Quantize” the problem
  - Solve the problem with suitable quantum algorithms
- Language/Environment: Python

Why optimization?

- The solution of **certain** (!!) **optimization problems** is considered to be one of the tasks where not only ideal fault-tolerant quantum computers, but possibly even NISQ computers “could” offer an advantage over classical computing.

- Why this is so will be motivated here using a simple bottom-up example

- This can be extended as desired to cover realistic use cases

- Here, the path from the problem via the “classical” solution to the quantum solution will be presented step by step

The problem

- Würfel GmbH produces and sells cube-shaped **boxes:**
- The company's goal is to maximize the revenue (minus costs) from the sale of the boxes:

The following applies (I):

- Boxes are made of paper (0) or cardboard (1). A cardboard box costs €50 more than a paper box.
- Box edges can be reinforced (1) or not (0). Reinforced boxes can be sold for €10 more.
- Be lined with foam (1) or not (0). Lined box is 20€ more expensive.
- Box is painted (1) or not (0). Painted box is 30€ more expensive.

→ For a cardboard box with reinforced edges, lining and painting, the additional sales revenue is 50€+10€+20€+30€=110 €

The problem

- In addition, the following (production) costs or additional revenues must be taken into account (II):

  - If the box is lined and reinforced: 30€ cost.
  - If the box is made of cardboard and reinforced: 30€ cost
  - If the box is painted and filled: Proceeds increase by 40€

  → Which combination of the decision variables paper/cardboard, edge reinforcement yes/no; lining yes/no; painting yes/no is “the best”??

- Optimization problems of this type are called **QUBO** (“ Quadratic unconstrained optimization “) problem

formalization of the problem

- The decision variables are $\vec{x}$ summarized in the vector with x_0 to x_3:
  - $x_0 = \begin{cases} 0, & paper\ box \\ 1, & cardboard\ box \end{cases}$
  - $x_1 = \begin{cases} 0, & Edges\ are\ not\ reinforced \\ 1, & Edges\ are\ reinforced \end{cases}$
  - $x_2 = \begin{cases} 0, & box\ is\ not\ lined \\ 1, & box\ is\ lined \end{cases}$
  - $x_3 = \begin{cases} 0, & plain\ box \\ 1, & colored\ box \end{cases}$
- This allows the conditions under (I) to be elegantly formulated as: $\boldsymbol{b^T} \cdot \vec{\boldsymbol{x}}$ with $\boldsymbol{b^T} = (50,10,20,30)$

formalization of the problem

The conditions under (II), which depend on two variables, can be written compactly as: $\vec{x}^T \cdot \mathrm{C} \cdot \vec{x}$

with the symmetric matrix $C = \begin{pmatrix} 0 & -30 & 0 & 0 \\ -30 & 0 & -30 & 0 \\ 0 & -30 & 0 & 40 \\ 0 & 0 & 40 & 0 \end{pmatrix}$

The size to be optimized = objective function H is therefore: $\boldsymbol{H = b^T \cdot \vec{x} + \frac{1}{2}\vec{x}^T \cdot \mathrm{C} \cdot \vec{x}}$

- Note: By replacing $b \rightarrow -b$ and $C \rightarrow -C$ the maximization problem can be turned into a minimization problem
- Further use $x_i \cdot x_i = x_i$

Classic solution to the problem

- Now find the combination of x_i, ie the vector $\vec{x}$, which optimizes the objective function
- In our case there are 2^4=16 possible combinations that you can try out = " Brute Force Solution"
- However, if you have 10 variables x_i , then here are $2^{10} = 1.024$ possible combinations; with 20 variables there are 1,048,576 possibilities; with 30 variables there are 1,073,741,824 possibilities, so brute force has its limits ☹
- However, there are a number of classical optimization methods (e.g. CPLEX) that can also be used for such problems ... but their solution does not necessarily have to be the optimum...☹

Demo in Qiskit :

- **Classic optimization**

“Quantization" of the problem

Idea : Use qubits instead of bits:

- These do not only take the values 0 or 1, but are $|\psi\rangle = a|0 + b|1\rangle$ described by a state
- Only during the measurement does the qubit have to “decide” whether it will with prob. $|a|^2$ be a 0 or with prob $|b|^2 = 1 - |a|^2$ be a 1
- Take four qubits; run a suitable algorithm; measure the four qubits and generate a bit string of length four, e.g. [0,1,1,0]
- What is a suitable algorithm so that this procedure produces – at least with high probability – the optimal bit string?

Quantization of the problem

- One of the algorithms that can be used is the **Quantum Approximate optimization Algorithm (QAOA)** based on
  Farhi , Edward, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm ." arXiv preprint arXiv:1411.4028 (2014)
  .

- Approach: $U = e^{iH}$ Because: All manipulations of the qubits (except the measurement) can be described by a unitary matrix; every unitary matrix U has this representation with a Hermitian matrix H.

- Use the quantized objective function of the problem as the Hermitian matrix :
  - $x_i \rightarrow \frac{1}{2}(1 - \hat{\sigma}_i^z)$with the Pauli z-matrix$\hat{\sigma}^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
  - $\boldsymbol{H = b^T \cdot \vec{x} + \frac{1}{2}\vec{x}^T \cdot C \cdot \vec{x}} \rightarrow \hat{H} = \sum_{i<j} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + \sum_i h_i\hat{\sigma}_i^z$

Quantization of the problem

$$\hat{H} = \sum_{i<j} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + \sum_i h_i\hat{\sigma}_i^z = \hat{H}_{Ising}$$

The objective function thus becomes a “Hamiltonian operator” and this model is known in physics as the Ising model or generally as the spin glass model

- In our problem with four qubits, the Hamiltonian can be represented as a $16\ x\ 16$ matrix ... which is diagonal ... and the smallest (eigen)value in the diagonal is the solution we are looking for !!! ☺
- This means that by “bloating” the problem we have found a representation in which the solution is immediately accessible ☺
- However, if our problem is too big, this won’t help us directly ☹

Quantization of the problem

- Indirectly, however, it does, as it conveys intuition about which strategy we should now pursue:

  Find the smallest **eigenvalue** of the Hamiltonian….

  … and thus, find the state $|\Psi\rangle$ such that the **expected value** of the

  Hamiltonian becomes minimal:$\langle\Psi|\hat{H}|\Psi\rangle \rightarrow min$

- Start with a parameterized approach $|\Psi(\alpha,\beta,\gamma,..)\rangle$and vary the parameters until the minimum is reached. This is the general idea of the so-called " **Variational Quantum Algorithms (VQA)** ".

- **QAOA** is a special case of this

Quantization of the problem

The QAOA approach is guided by physics, namely the so-called **adiabatic theorem** :
"A system in which changes are made slowly enough always remains in its ground state."

- $\widehat{H}_{QAOA} = \beta \widehat{H}_{mixer} + \gamma \widehat{H}_{Ising}$ with $\widehat{H}_{mixer} = \sum_i \hat{\sigma}_i^x$ whose ground state is trivial
- Instead of starting the system in the ground state of the mixer according to the adiabatic theorem and letting it slowly (adiabatic!) transition to the ground state of the β Ising Hamiltonian , and $\gamma$are treated as parameters to be optimized.
- What's more, it is $U = e^{i\widehat{H}_{QAOA}}$ applied not just once but k times with different parameters each time $\beta_i, \gamma_i; \mathrm{i} = 1, .., \mathrm{k}$
- the parameters $\beta_i, \gamma_i; \mathrm{i} = 1, .., \mathrm{k}$are determined using a classical optimizer.

Quantization of the problem

- This is often expressed as follows:

$$\left|\vec{\beta},\vec{\gamma}\right\rangle = \mathrm{U}\left(\beta_k \widehat{H}_{mixer}\right)\mathrm{U}\left(\gamma_k \widehat{H}_{Ising}\right) \dots \mathrm{U}\left(\beta_1 \widehat{H}_{mixer}\right)\mathrm{U}\left(\gamma_1 \widehat{H}_{Ising}\right)|0,\dots,0\rangle$$

- the parameters $\beta_i, \gamma_i; \mathrm{i} = 1,..,\mathrm{k}$ are determined using a classic optimizer:
  - Prepare $\left|\vec{\beta},\vec{\gamma},\right\rangle$with initial values for$\beta_i \; und \; \gamma_i$
  - Measure all n qubits → Bit string of length n
  - Insert this bit string into the (classical) objective function
  - Vary $\beta_i \; und \; \gamma_i$until an optimum of the objective function is reached
- And this works ☺

Demo in Qiskit :

- **Optimization with the Quantum Computer**

Extensions of QUBOS

- Problem class “QUBO” seems very restrictive at first glance …..
- But can be extended in different directions:
  - **Quadratic**: It is possible to consider third and higher order interactions → three- or more qubit gates, which can, however, be reduced to elementary (ie maximum two-qubit) gates
  - **Unconstrained**: Constraints $\sum_i x_i = B$ can be implemented using a quadratic penalty term.
  - **Binary**: Rational numbers can be represented using their binary notation; the same applies to real numbers´. However, the number of qubits required grows very quickly ☹

Where do these problems play a role?

- On https://blog.xa0.de/post/List-of-QUBO-formulations/ you can find 112(!) optimization problems that can be formulated as QUBO:
  - Many well-known “ **classical NP-hard”** optimization tasks can be formulated as QUBO: e.g. Max-Cut, Maximum 2- and 3-Sat, Graph Colouring, Traveling Salesman , …
  - Optimization tasks in the context of classical machine learning: e.g. support vector machines, k-means clustering, ...
  - As well as optimization tasks for specific use cases: e.g. settlement of securities, portfolio optimization, route planning, job shop scheduling , etc.

Where do these play a role?

https://blog.xa0.de/post/List-of-QUBO-formulations/

jrnl · home about list my ventures publications LinkedIn

About the author:
Daniel is CTO at rhome GmbH, and Co-Founder at Aqarios GmbH. He holds a M.Sc. in Computer Science from LMU Munich, and has published papers in reinforcement learning and quantum computing. He writes about technical topics in quantum computing and startups.

List of QUBO formulations

Below a list of 112 optimization problems and a reference to mulation of each problem is shown. While a lot of these problems are classical op blems from mathematics (also mostly NP-hard problems), there are interestingly a ns for Machine Learning, such as the L1 norm or linear regression. Graph base problems often encode the graph structure (adjacency matrix) in the QUBO, while such as Number Partitioning or Quadratic Assignment encode the problem matrices s.t. non-linear system of equations can be found in the QUBO. The quadratic unconstrained binary optimization (QUBO) problem itself is a NP-hard optimization problem that is solved by finding the ground state of the corresponding Hamiltonian on a Quantum Annealer. The ground state is found by adiabatically evolving from an initial Hamiltonian with a well-known ground state to the problem Hamiltonian.

This is by no means the definite list of all QUBO formulations out there. This list will grow over time.

For 20 of these problems the QUBO formulation is implemented using Python (including unittests) in the QUBO-NN project. The Github can be found here.

Problem	QUBO formulation
Number Partitioning (NP)	Glover et al.[2]
Maximum Cut (MC)	Glover et al.[2]
Minimum Vertex Cover (MVC)	Glover et al.[2]
Set Packing (SP)	Glover et al.[2]
Set Partitioning (SPP)	Glover et al.[2]
Maximum 2-SAT (M2SAT)	Glover et al.[2]
Maximum 3-SAT (M3SAT)	Dinneen et al.[9]
Graph Coloring (GC)	Glover et al.[2]
General 0/1 Programming (G01P)	Glover et al.[2]
Quadratic Assignment (QA)	Glover et al.[2]
Quadratic Knapsack (QK)	Glover et al.[2]
Graph Partitioning	Lucas[7]

Good starting point for working on your own problems!

Quantum advantage in solving QUBOs?

- There is no mathematical proof of a quantum advantage in solving QUBOs. Polynomial solutions are generally not expected. Possibly a “smaller exponent”
- The QAOA algorithm has been most intensively studied theoretically and practically:
  - Exact mathematical statements about the quality are currently only available for e.g. k=1
  - And/or certain special cases, ie certain structure of the interaction matrix
  - Usually also well solvable classically

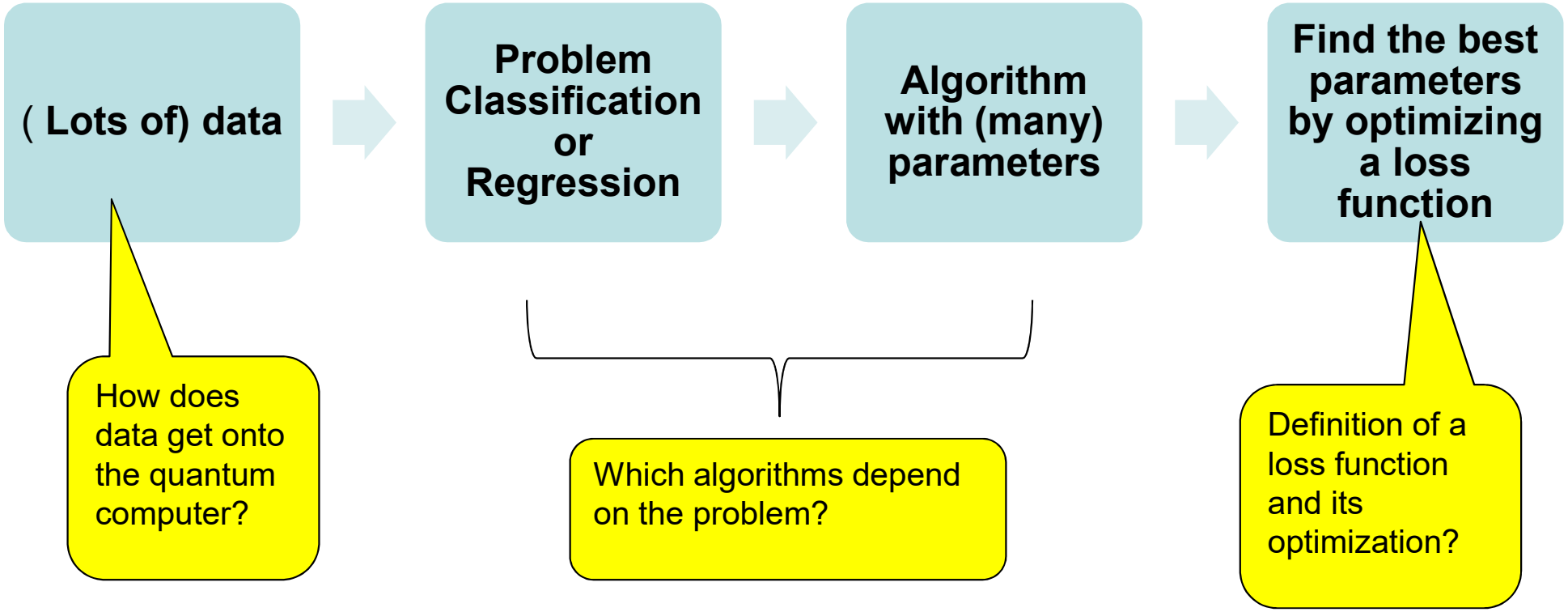
Open research questions

- To what extent does a possible quantum advantage depend on the problem structure (=interaction matrix)?
- How error-resilient is the QAOA algorithm, for example, and how can this be increased by modifying the algorithm?*
- How can real-world problems from applications (business and technology) be put into QUBO form more quickly and easily?

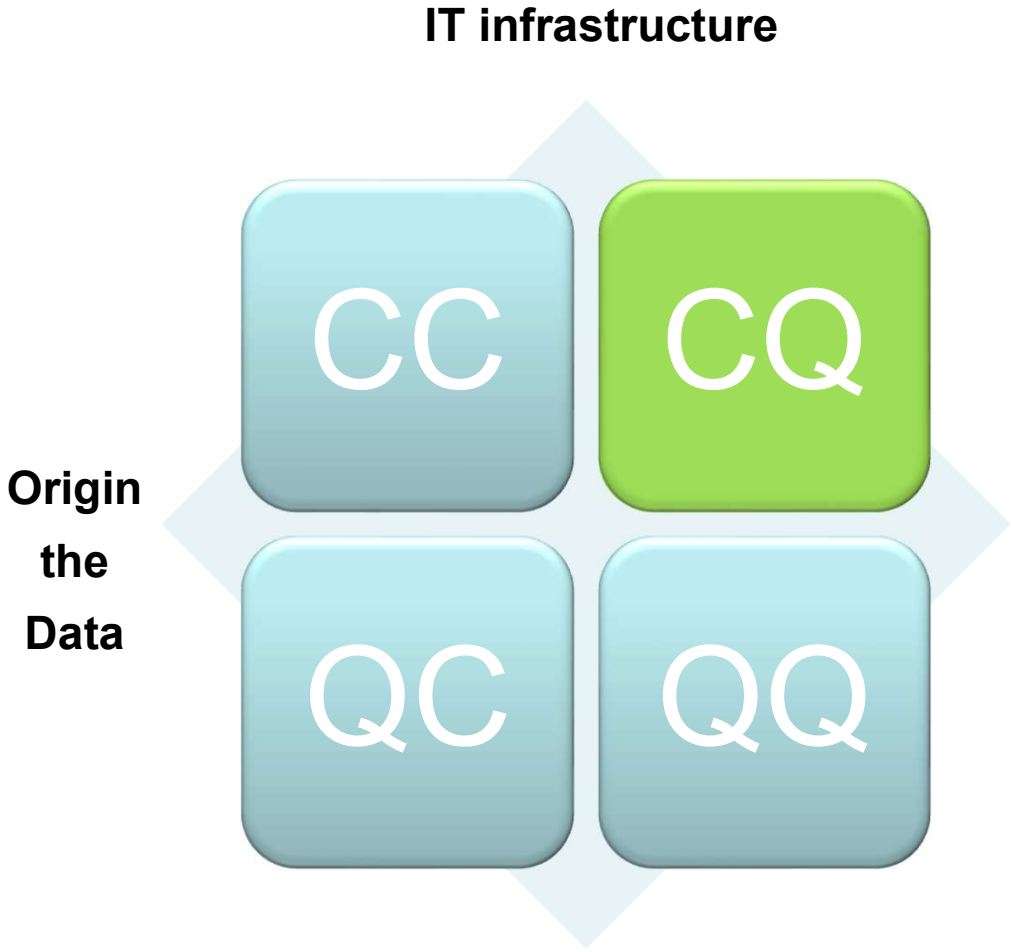
* Benchmarking the performance of portfolio optimization with QAOA,
S. Brandhofer, D. Braun, V. Dehn, G. Hellstern, M. Hüls, et al. Quant.Inf.Proc . 22 (2023) 1, 25

4. QUANTUM MACHINE LEARNING

Machine Learning → Quantum Machine Learning



problem



Further differentiation within “CQ”:

Algorithms for (future) ideal quantum computers :

- Application of the well-known “basic algorithms” (Grover, QFT, …) with the aim of reducing complexity/speed compared to classical machine learning algorithms

- Jackpot : Is there an exponential speedup , i.e. reduction from exponential to polynomial running time?
  - Mathematically complete proofs are extremely tricky here... (see e.g. A quantum-inspired classical algorithm for recommendation systems, E. Tang, arXiv:1807.04271)
  - This advantage is partly negated by the problem of getting the classical data “onto the quantum computer”

Further differentiation within “CQ”:

Algorithms for NISQ calculators :

- Initial questions:
  - How can existing quantum hardware be used for machine learning?
  - Does it make sense to use the quantum computer as an “algorithm” and combine it with classical components?
  - Are there certain dates where there might be an “advantage”?

→ Speed or complexity advantage is generally not expected

→ Hope for performance gain in the sense of e.g. better discrimination in classification problems

→ Basically a heuristic topic – theory is still in its infancy

NISQ-compatible ML algorithms

Quantum-assisted Support Vector machine

- SVM requires a so-called kernel matrix
- This provides information about the (generalized) distances between the data points and in particular allows the data to be embedded in a high-dimensional space
- It has been shown that if kernel matrix is generated with a quantum computer, embeddings beyond the classical possibilities can be achieved (Supervised learning with quantum-enhanced feature spaces , V. Havlíček , et al Nature volume 567, 209–212 (2019)
- However, it is (still) unclear whether these non-trivial embeddings actually offer an advantage

NISQ-compatible ML algorithms

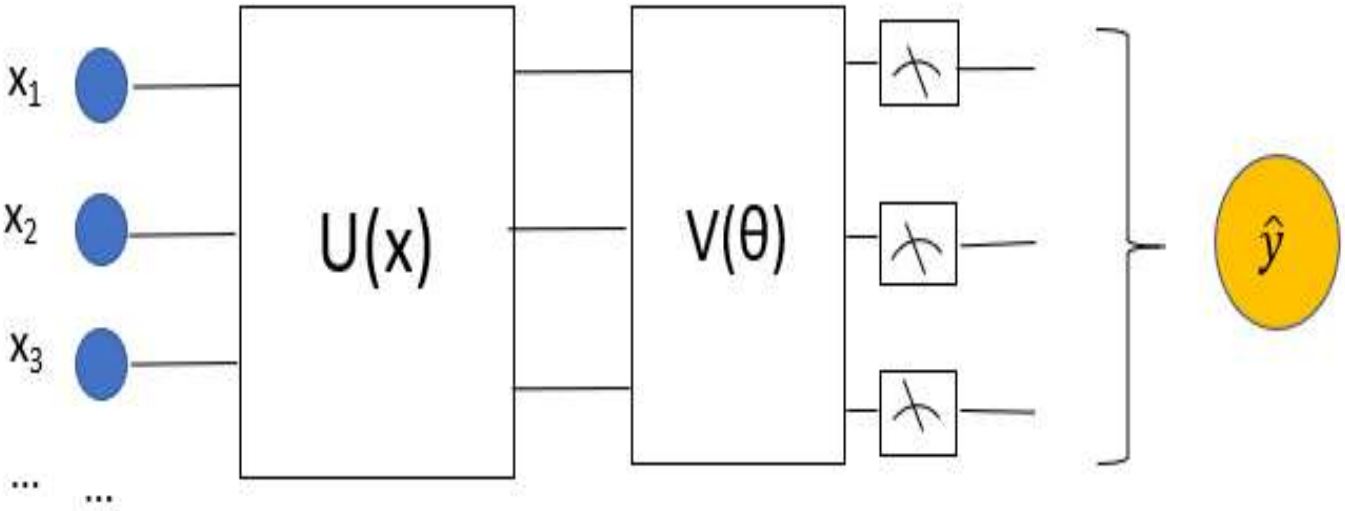
Quantum Neural Networks (QNN)

- Classical data is “loaded” into the quantum computer using a unitary matrix **U(x) = Data Encoding**
  - There are various options available for this
  - Partly poor scaling properties if data has many features
- Parameterizable unitary matrix **V($\boldsymbol{\theta}$)** with a layer structure analogous to classical neural networks
- Measurement of the qubits and classical post-processing to map the measurement results to the target variable (classification or regression)
- Classical feedback loop to use a loss function to determine the parameters$\boldsymbol{\theta}$ to adjust

implementation ??

NISQ-compatible ML algorithms

Quantum Neural Networks (QNN)



Goal :
Less “ speedup ” – but “better” algorithms, e.g. more accurate classification of data

Requirements for software tools for QML

For QML, quantum circuits but also classical Components (post-processing, feedback loop for parameters, etc.) required

- Out-of-the-box solutions in the ML world → Can these be used?
- Support in configuring **U(x)** and **V(θ)** → how to quickly switch between different implementations?
- For the training phase, the circuit must be run VERY OFTEN → high performance of the quantum simulators is necessary!
- Can different physical backends (with different technologies, e.g. photonic QC) be connected?
- Is there an ecosystem of developers and users for exchange?

Data-Encoding U(x)

Discussion of the following possibilities:

- Basis Encoding
- amplitude encoding
- Angle Encoding
- SU(2) Encoding

Basis Encoding

Basic encoding as the conceptually simplest option

– Represent data as binary numbers and then encode them into corresponding qubits:

input data		
Sample:	feature x_1	feature x_2
1	5	6
2	4	1

input data	
Sample:	quantum state
1	$\|101\ 110\rangle$
2	$\|100\ 001\rangle$

input data		
Sample:	binary feature x_1	binary feature x_2
1	101	110
2	100	001

Amplitude encoding

Amplitude encoding

- Use the classical data as amplitudes of a quantum state

feature x_1	feature x_2	feature x_3	feature x_4
6	-10.3	13.52	4.2

standardization

feature x_1	feature x_2	feature x_3	feature x_4
$6/18{,}51$	$-10{,}3/18{,}51$	$13{,}52/18{,}51$	$4{,}2/18{,}51$

- Amplitude encoded state with log2(n) qubits for n-dimensional point:

$$\frac{6}{18{,}51}|00\rangle - \frac{10{,}3}{18{,}51}|01\rangle + \frac{13{,}52}{18{,}51}|10\rangle + \frac{4{,}2}{18{,}51}11\rangle$$

Angle Encoding

Angle Encoding

- Use the classical data as arguments of a Pauli-Y rotation

feature x_1	feature x_2	feature x_3	feature x_4
6	-10.3	13.52	4.2

factor 1/2

feature x_1	feature x_2	feature x_3	feature x_4
$6/2$	$-10{,}3/2$	$13{,}52/2$	$4{,}2/2$

$$\bigotimes_{i=1}^{4} R_y(x_i)|0000\rangle = \begin{pmatrix}\cos(3)|0\rangle \\ \sin(3)|1\rangle\end{pmatrix} \otimes \begin{pmatrix}\cos(-5{,}15)|0\rangle \\ \sin(-5{,}15)|1\rangle\end{pmatrix} \otimes \begin{pmatrix}\cos(6{,}76)|0\rangle \\ \sin(6{,}76)|1\rangle\end{pmatrix} \otimes \begin{pmatrix}\cos(2{,}1)|0\rangle \\ \sin(2{,}1)|1\rangle\end{pmatrix}$$

- Compared to amplitude encoding, more qubits but lower gate complexity

SU(2) Encoding

SU(2) Encoding

- A unitary transformation of a qubit can be parameterized by three angles (cf. Quantum algorithms and their implementation (1))
  → encode three features in each qubit

feature x_1	feature x_2	feature x_3
6	-10.3	13.52

→

θ	ϕ	λ
6	-10.3	13.52

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\phi+}\ \cos\left(\frac{\theta}{2}\right) \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{6}{2}\right) & -e^{13{,}52i}\sin\left(\frac{6}{2}\right) \\ e^{-10{,}3i}\sin\left(\frac{6}{2}\right) & e^{-10{,}3i+13{,}52i}\cos\left(\frac{6}{2}\right) \end{pmatrix}$$

Data-Encoding U(x)

In addition to the encoding methods discussed here, there are others, some of which are also implemented in PennyLane :

- Hamiltonian Encoding
- IQP Encoding
- QAOA encoding
- ….

The question of the “best” encoding strategy is a current area of research *(see e.g. Effect of data encoding on the expressive power of variational quantum-machine-learning models, M. Schuld et al, Phys. Rev. A 103, 032430, 2021)*

Demo in PennyLane :

- **Data Encoding**

Variational circuit V(θ)

- A sequence of gates with (free) parameters is applied to the data embedded in the circuit
- Conceptually, this corresponds to the approach of a neural network
- It can be shown that this approach can be used to approximate any sufficiently benign function (→ regression problem) or that it is possible to separate the function into different classes (→ classification problem)
- Not clear a priori which sequence of gates is “best”
  ☹ This is also a current area of research!

Variational circuit V(θ)

- The circuit can in principle be constructed from 1-qubit gates and gates for entanglement CX or CZ

- To increase the complexity, the gates are not applied only once but arranged in layers, which are then repeated → corresponds to the layer structure of a neural network

- It has been found that it can be advantageous to feed the data not only at the beginning of the circuit, but several times, so-called “data re-uploading ” (cf. *Data re-uploading for a universal quantum classifier, A. Pérez-Salinas, Quantum 4, 226 (2020)).*

Circuit with Basic Entangling

- For each qubit, only one elementary gate (RX, RY or RZ) is applied in parallel, each with one free parameter (“angle of rotation”)
- Then qubits are entangled in a chain/ring: 1st qubit with 2nd qubit, 2nd qubit with 3rd qubit, …. ,last qubit with 1st qubit
- The layer defined in this way is applied several times
- Number of parameters: number of qubits * number of layers

Circuit with Strongly Entangling

- A general SU(2) gate $U(\theta, \phi, \lambda)$with three free parameters is applied to each qubit in parallel.
- Then qubits are entangled in a chain/ring: 1st qubit with 2nd qubit, 2nd qubit with 3rd qubit , …., last qubit with 1st qubit
- The layer defined in this way is applied several times
- Number of parameters: 3 * number of qubits * number of layers

Possible extensions

- Use parameter-dependent 2-qubit gates instead of CX and CZ
- Entangle not only “neighboring qubits” but all qubits with all qubits
- Divide the layers into blocks that are homogeneous in themselves
- For a systematic comparison of 19 (!) different approaches for four qubits, see *“ Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms”, S. Sim et al, Advanced Quantum Technologies 2.12 (2019): 1900070*

Demo in PennyLane :

- **Variational circuit**

Combination to the overall picture

After we have specified U(x) and V(θ), all that is missing is:

1. The measurement of one or all qubits → Bit / Bitstring

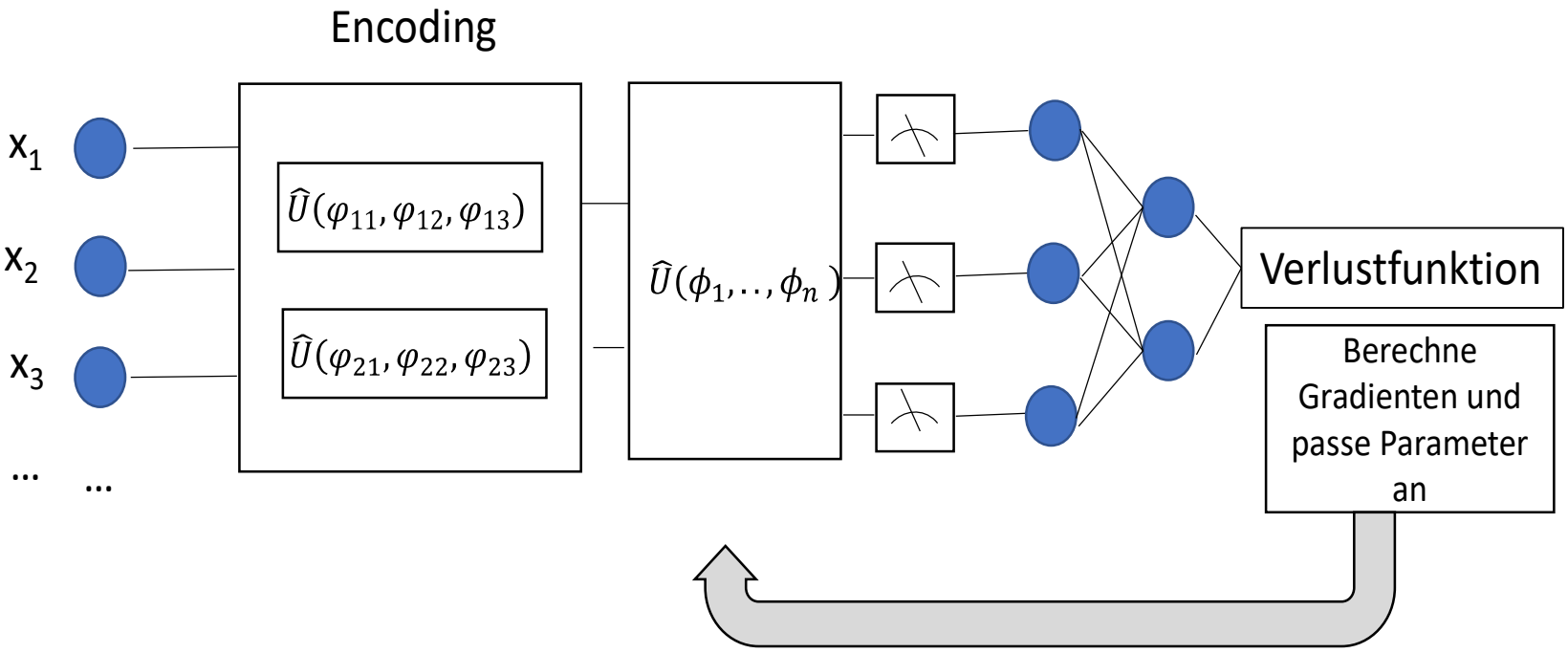
2. The classic postprocessing : Bit/bitstring → target value of the problem, e.g. label in a classification problem

3. Definition and evaluation of a loss function → RMSE for regression or CrossEntropy for classification

4. Classical optimization to adjust the parameters of the circuit until the loss function reaches the minimum

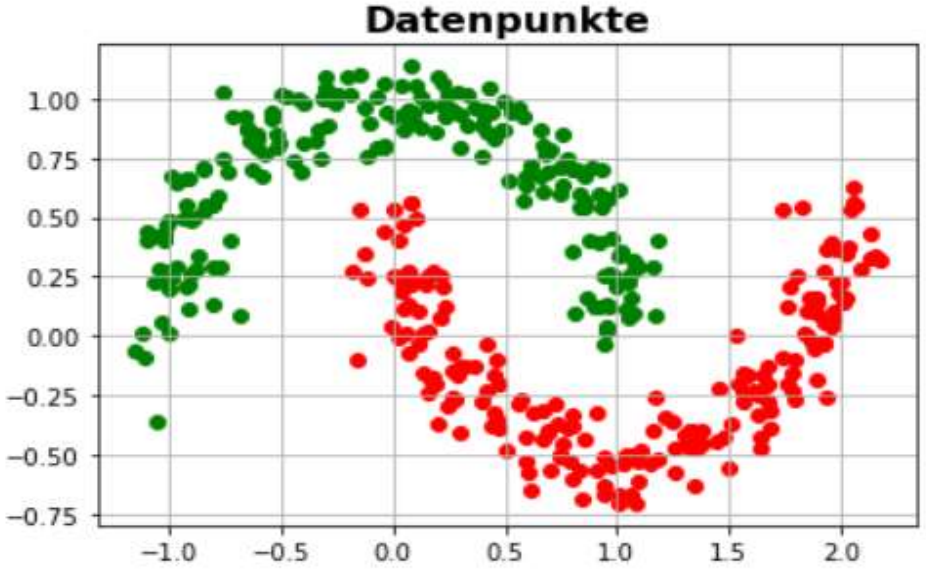
→ Steps 2 to 4 should be carried out by a classic ML framework, e.g. Pytorch .

Combination to the overall picture



Classification of 2-Dim Data

- As an example, the two-dimensional data set “Moon” with 400 points is to be classified:



- A linear classification of the data is not possible
- It is a binary classification problem with the target variable “green” or “red” or 0 or 1.

Classification of 2-Dim Data

- As will be shown shortly, a classification model with an accuracy > 95% on training and test data is possible….
- … which is, however, in the same order of magnitude as the accuracy of a “classical” machine learning model
- But are there situations where a quantum model delivers MORE ACCURATE results than a classical model?? Maybe even with fewer training steps?
- This is an open research question – although there is a very strong suspicion that it will ultimately depend on the data or its structure

Demo in PennyLane :

- **implementation of a QNN**

Resources: Literature

THE standard reference:

- Quantum Computation and Quantum Information, Michael A. Nielsen & Isaac L. Chuang , Cambridge University Press, 2010

Easy introduction:

- Understanding Quantum Computing, Fundamentals – Applications – Perspectives, Matthias Homeister , Springer 2022
- **Quantum computing compact, Bettina Just, Springer 2020**

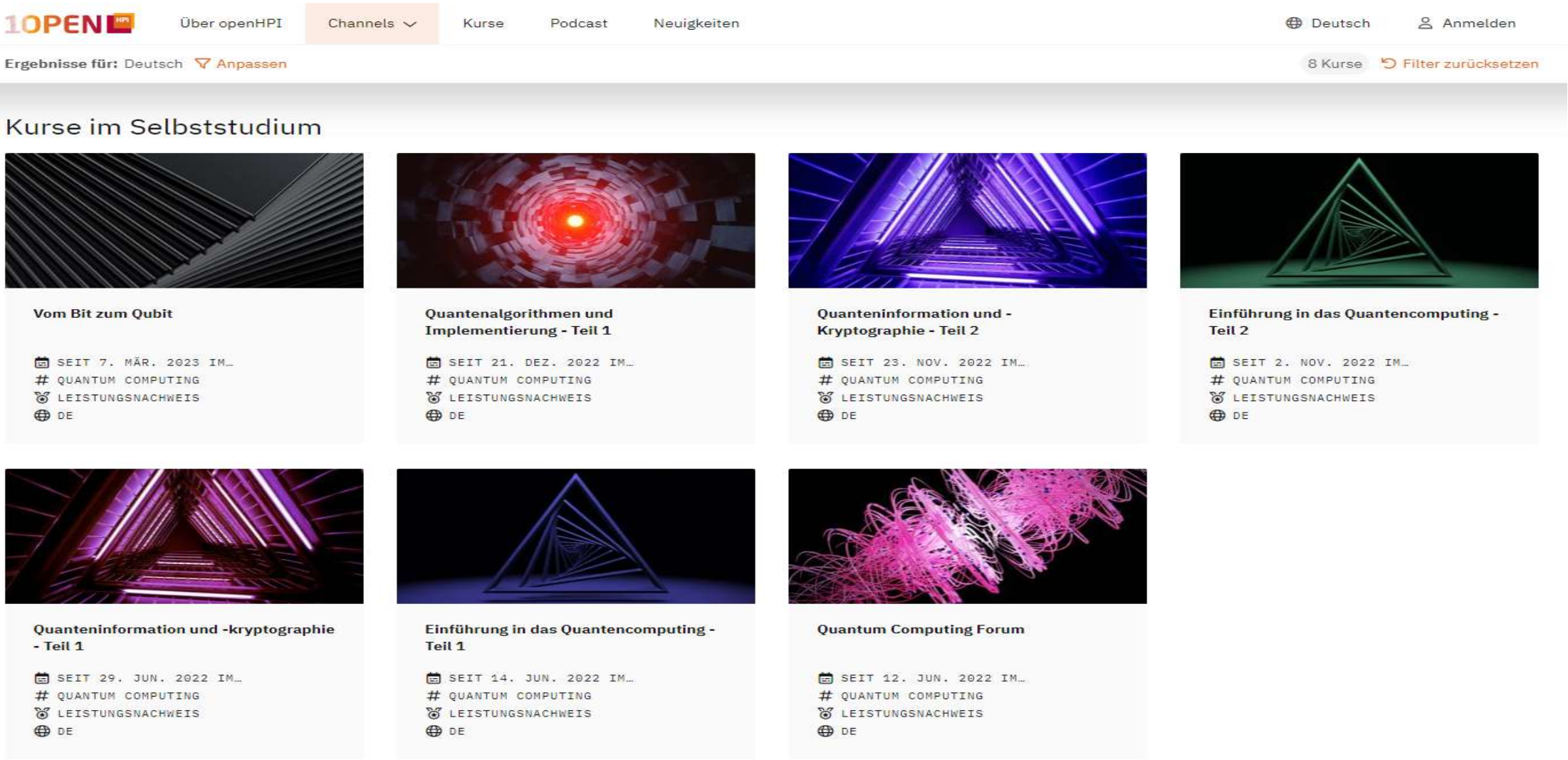
Incl. programming:

- Dancing with Qubits , Robert S. Sutor, Packt Publishing, 2023
- Quantum Computing: An Applied Approach, Jack D. Hidary , Springer, 2019

Focus QML:

- Supervised Learning with Quantum Computers, Maria Schuld & Francesco Petruccione , Springer, 2023

Resources: https://open.hpi.de/channels/quantum

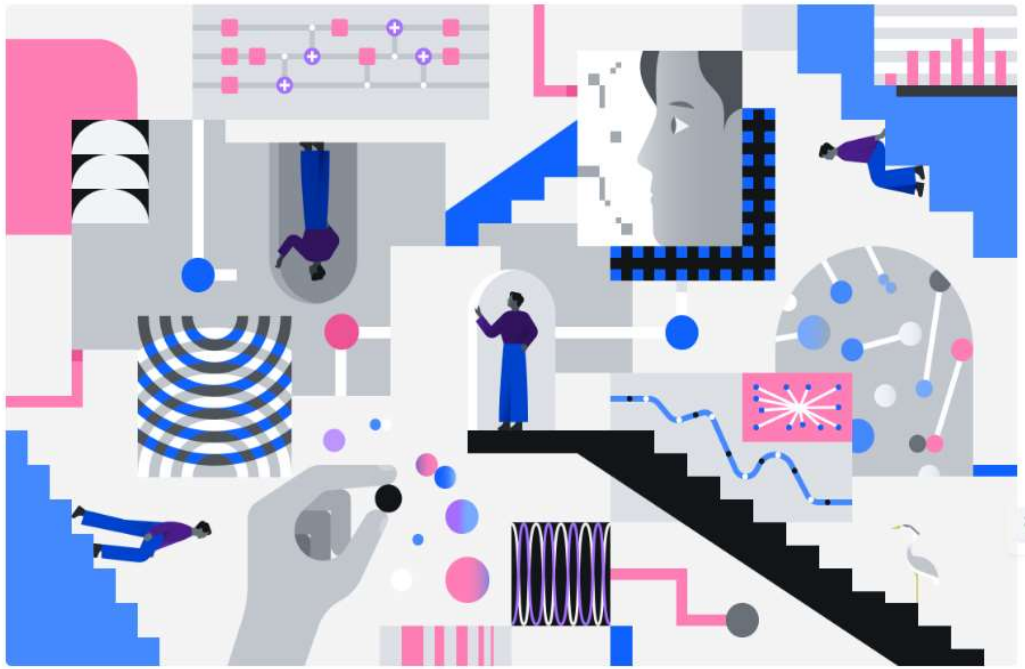


Resources: https://learning.quantum.ibm.com/course/basics-of-quantum-information

IBM Quantum **Learning** | Home | Catalog | Composer | Sign in

Basics of Quantum Information

Created by John Watrous



This is the first unit of the *Understanding Quantum Information and Computation* series, which explains quantum information and computation at a detailed mathematical level.

The first unit, *Basics of Quantum Information*, begins with a mathematical description of quantum information for both single and multiple systems, then moves on to quantum circuits, and finally covers three fundamentally important examples — quantum teleportation, superdense coding, and the CHSH game — all of which are connected to the phenomenon of entanglement.

The series includes both written content and videos featuring John Watrous.

Sign in to track progress | Start from beginning →

Contact

Prof. Dr. Gerhard Hellstern

Center of Finance
Baden-Württemberg Cooperative State University
Stuttgart

gerhard.hellstern@dhbw-stuttgart.de

Prof. Dr. Gerhard Hellstern
Banking, Data Science and Quantum Computing
(Qiskit Advocat)