Lecture

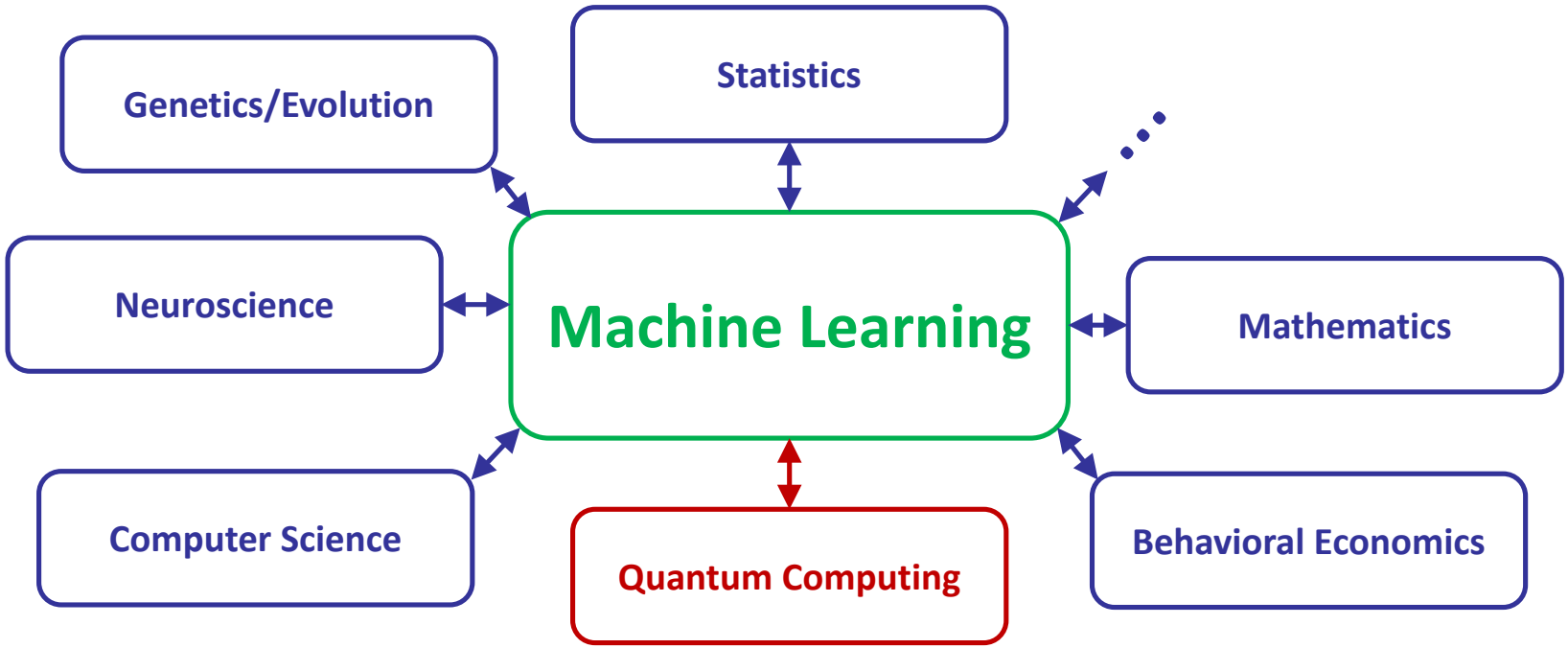
Quantum Computing

(CS5070)

Introduction into Quantum Machine Learning: Data Encoding, Model, Measurement

Professor Dr. rer. nat. habil. Sven Groppe
https://www.ifis.uni-luebeck.de/index.php?id=groppe

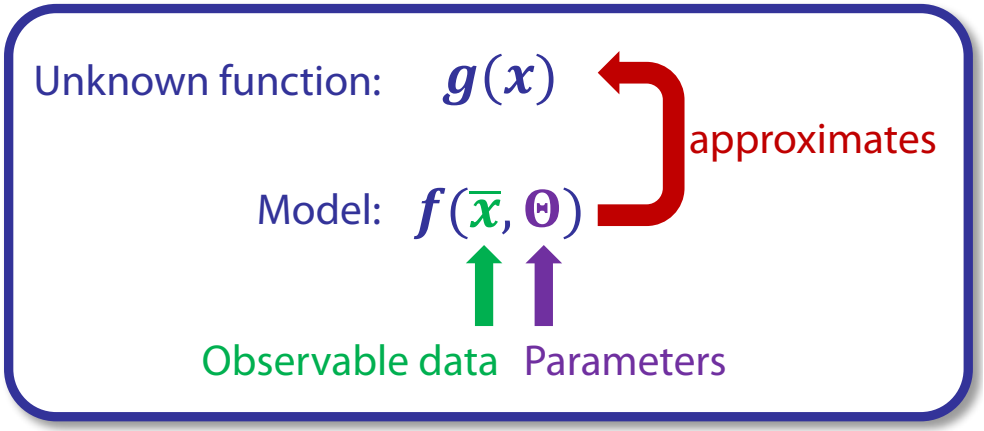
Influences to Machine Learning



"Learning patterns from data in order to draw inferences"

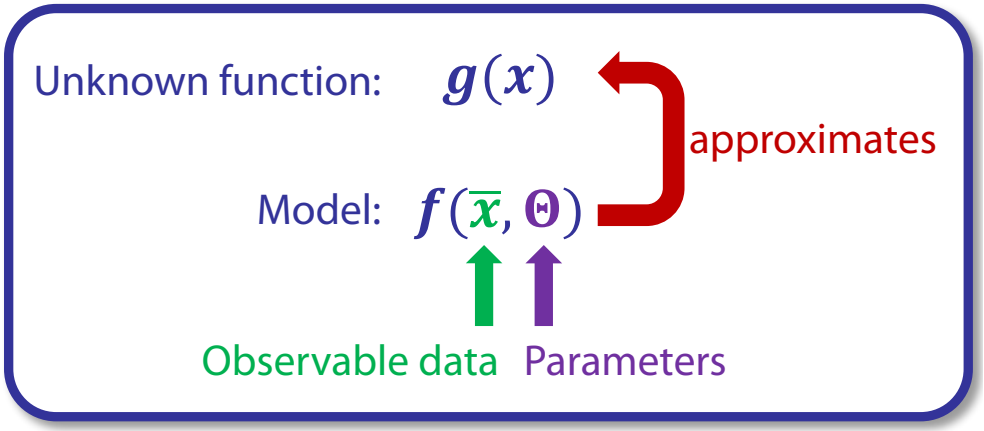
Machine Learning - Basics

Function approximation & optimization



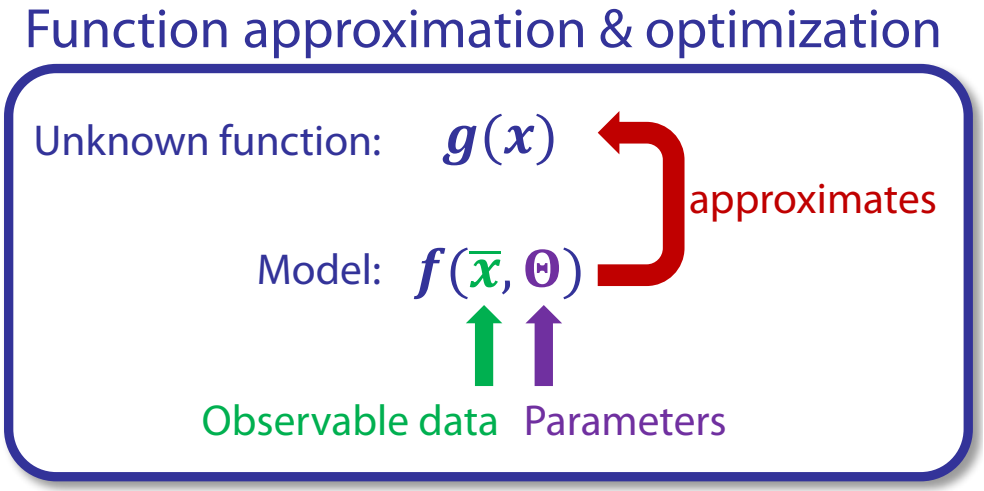
Machine Learning - Basics

Function approximation & optimization



- How to choose a model?
- How to find optimal parameters?

Machine Learning - Basics



- How to choose a model?
- How to find optimal parameters?

Follow-up questions related to quantum computing:

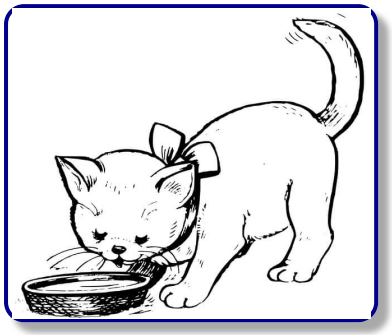
- Are there quantum models?
- Can quantum computing be utilized to find optimal parameters?

Machine Learning Ingredients: Data

- Pictures
- Videos
- Spreadsheets
- 123 Numerical/Categorical
- Texts in Natural Language
  - PDF Scientific Publications
  - EMail
  - Chat Communications
  - ...
- Source Code
- ...

Machine Learning Ingredients: Data

Picture



n pixels

m pixels

- Every pixel $\in [-1, 1]$ from black to white
- Every image as vector:

$$x_1 := \begin{bmatrix} -0.6 \\ +0.32 \\ ... \\ -0.99 \end{bmatrix}$$

$n \cdot m$ elements (in this context also called *features*)

Dataset

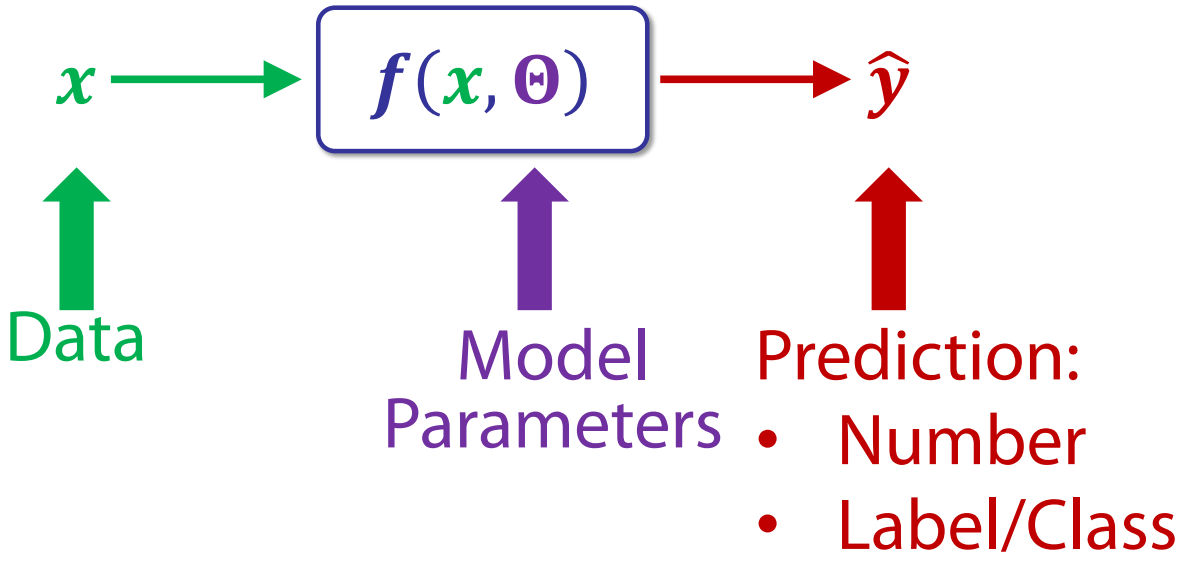
- A dataset D consists of many images: $(x_1, ..., x_N)$
- Every image is a sample with N samples in total
- Huge matrix to represent all images:

$$D = \begin{bmatrix} x_1 & x_2 & ... & x_N \end{bmatrix} = \begin{bmatrix} -0.6 & +0.12 & ... & +0.55 \\ +0.32 & +1 & ... & -0.38 \\ ... & ... & ... & ... \\ -0.99 & +0.03 & ... & +0.22 \end{bmatrix}$$

Matrix of size $N \times n \cdot m$

Machine Learning Ingredients: Model

- A model is a function that takes data, depends on some parameters and maps this data to some prediction

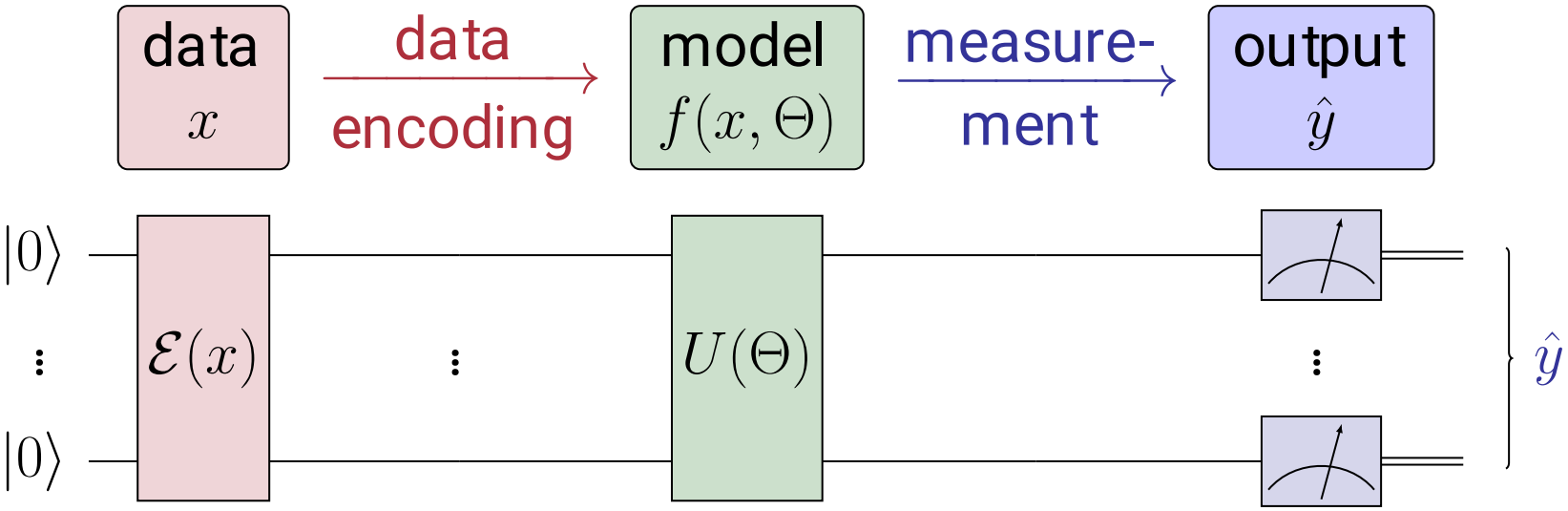


Machine Learning Ingredients: Data and Quantum Model

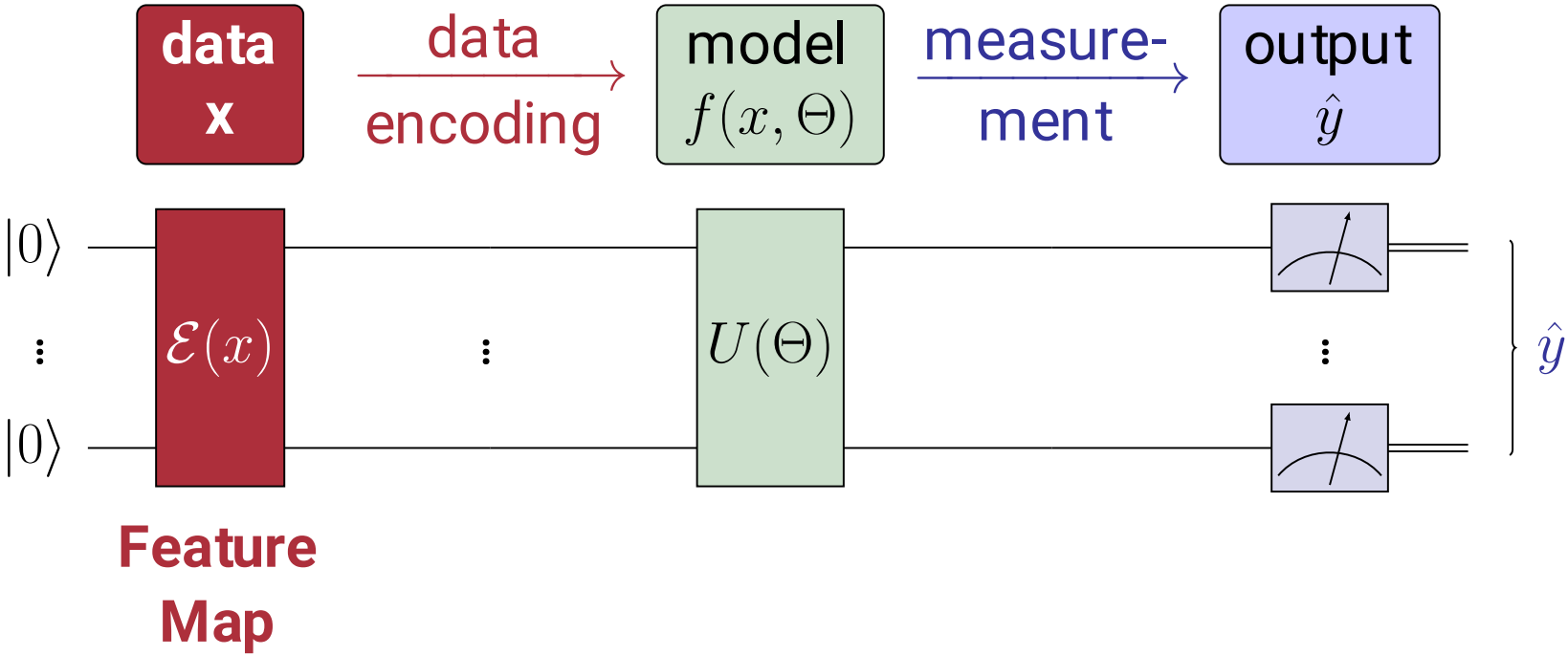
- Possibilities of involving Quantum Computing:

		Data Processing	
		Classical	Quantum
Representation of data	Classical	CC	CQ
	Quantum	QC	QQ

How can we encode data and build a quantum model?

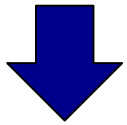


How can we encode data and build a quantum model? - Data encoding



How to encode data x?

$$x$$



$$|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle + \cdots + \alpha_k \cdot |k\rangle$$

???

How to encode data x? - Basis encoding

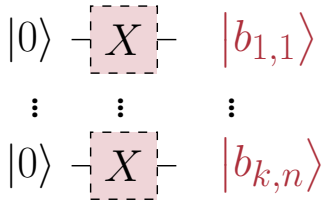
Idea

Encode *binary* representation of data in basis states

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} b_{1,1} \cdots b_{1,n} \\ b_{2,1} \cdots b_{2,n} \\ \vdots \\ b_{k,1} \cdots b_{k,n} \end{bmatrix}$$

$$|x\rangle = |b_{1,1} \cdots b_{1,n} \cdots b_{k,1} \cdots b_{k,n}\rangle$$

Circuit:



Pro:
Easy to load

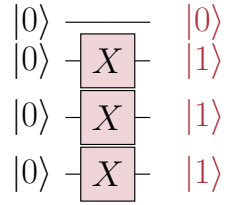
Example

Data x with 2 features

$$x = \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 01_b \\ 11_b \end{bmatrix}$$

$$|x\rangle = |0111\rangle$$

Circuit:



Contra:
$k \cdot n$ qubits for $k \cdot n$ bits

How to encode data x? - Amplitude encoding

Idea

Encode *normalized* data in state amplitudes

$$x \xrightarrow{\text{normalize}} x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

$$|\Phi_x\rangle = \sum_{i=0}^{2^n-1} x_{i+1} |i\rangle$$

Pro:
$\lceil log(k) \rceil$ qubits for k features

Example

Data x with 4 features

$$x = \begin{bmatrix} 3.0 \\ 1.0 \\ 4.0 \\ 2.0 \end{bmatrix}$$

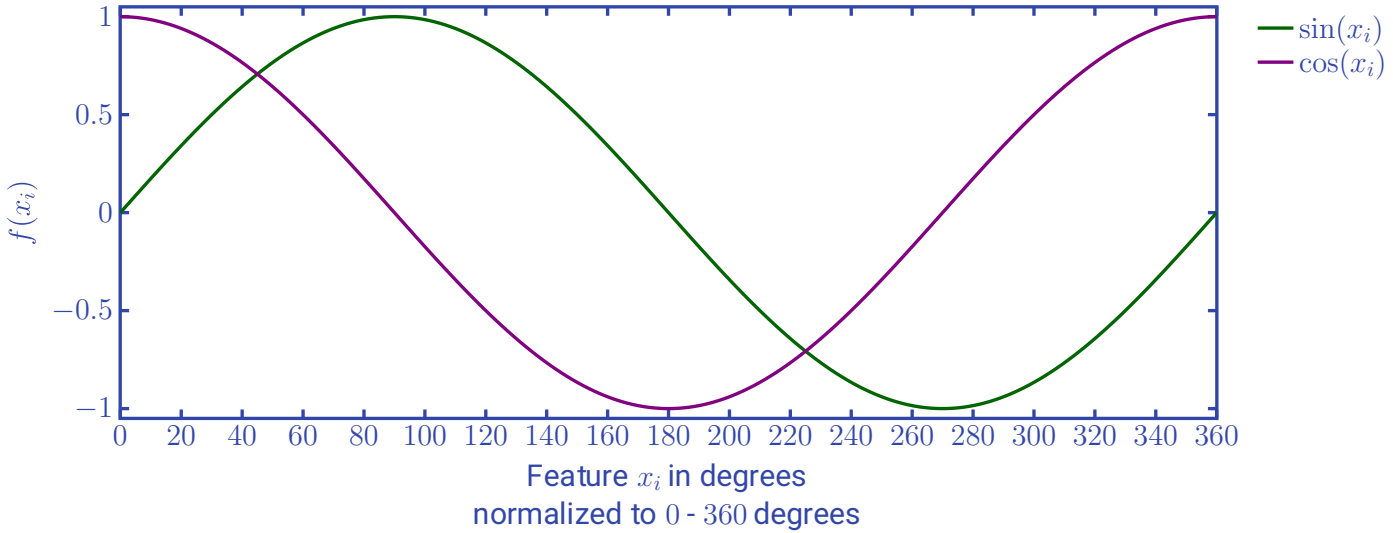
$$|\Phi_x\rangle = \frac{3.0 \cdot |00\rangle + 1.0 \cdot |01\rangle + 4.0 \cdot |10\rangle + 2.0 \cdot |11\rangle}{\sqrt{3.0^2 + 1.0^2 + 4.0^2 + 2.0^2}}$$

Contra:
Difficult to load!

How to encode data x? - Angle encoding

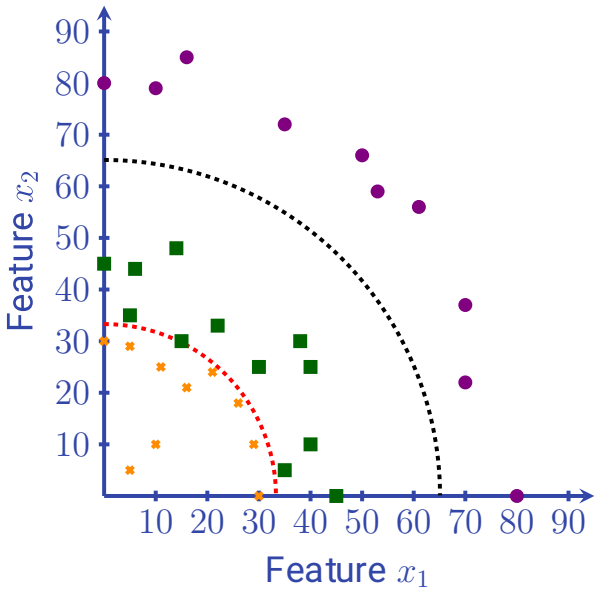
- **Idea:** Encode (normalized) data in rotation angles

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} \Rightarrow \begin{matrix} |0\rangle - R_y(2 \cdot x_1) - \begin{bmatrix} cos(x_1) \\ sin(x_1) \end{bmatrix} \\ |0\rangle - R_y(2 \cdot x_2) - \begin{bmatrix} cos(x_2) \\ sin(x_2) \end{bmatrix} \\ \vdots \qquad\qquad \vdots \\ |0\rangle - R_y(2 \cdot x_k) - \begin{bmatrix} cos(x_k) \\ sin(x_k) \end{bmatrix} \end{matrix} \equiv \begin{bmatrix} cos(x_1) \\ sin(x_1) \end{bmatrix} \otimes \begin{bmatrix} cos(x_2) \\ sin(x_2) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} cos(x_k) \\ sin(x_k) \end{bmatrix}$$



Feature Map via Angle Encoding

- ***Kernel trick:*** A feature map can transform data into a space
  - where it is *easier to process*
  - where features are *easier to be classified* (by e.g. linear functions)

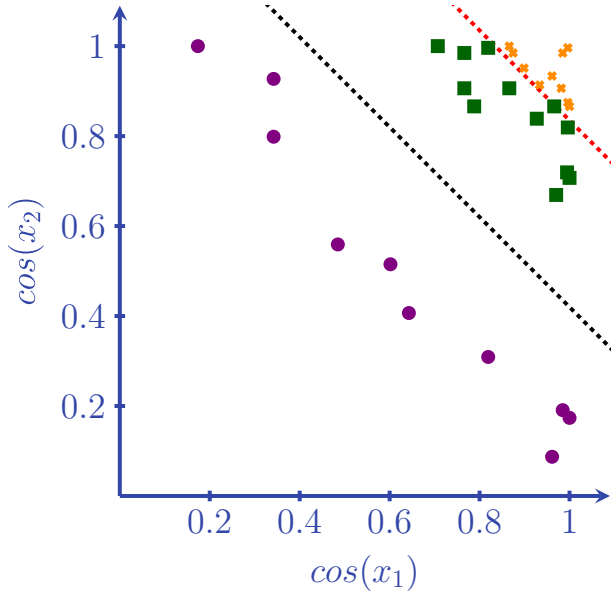


Data in original space...

Apply feature map
(here *angle encoding*)

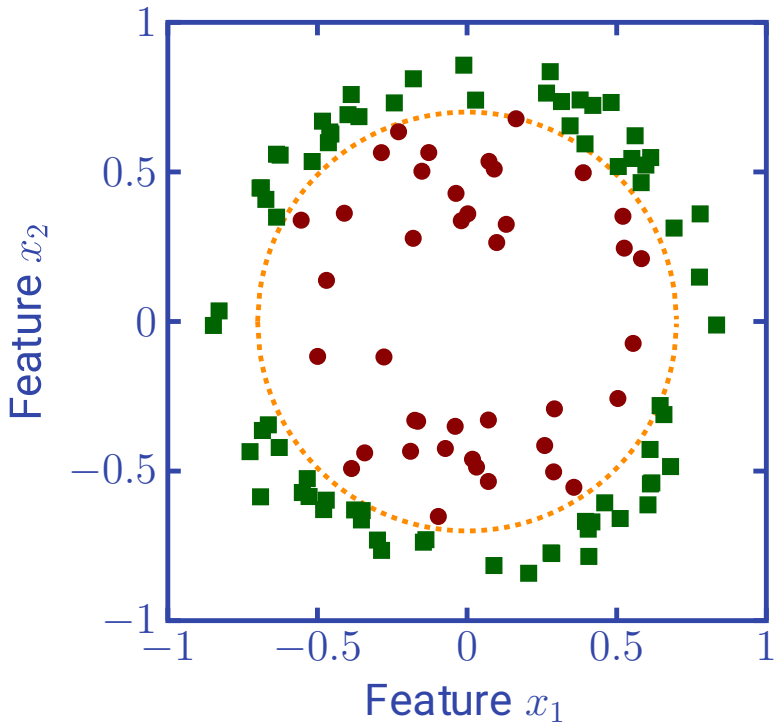
neglect
sinus-dimensions

due to simplicity
of presentation

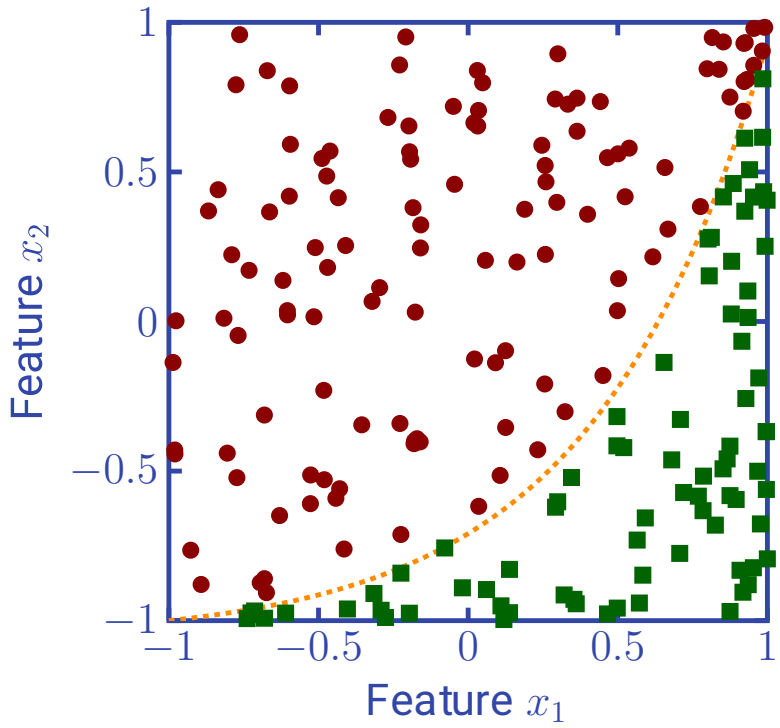


Data in feature space...

Feature Maps for other Class Positions than Quarter Circles 1/2

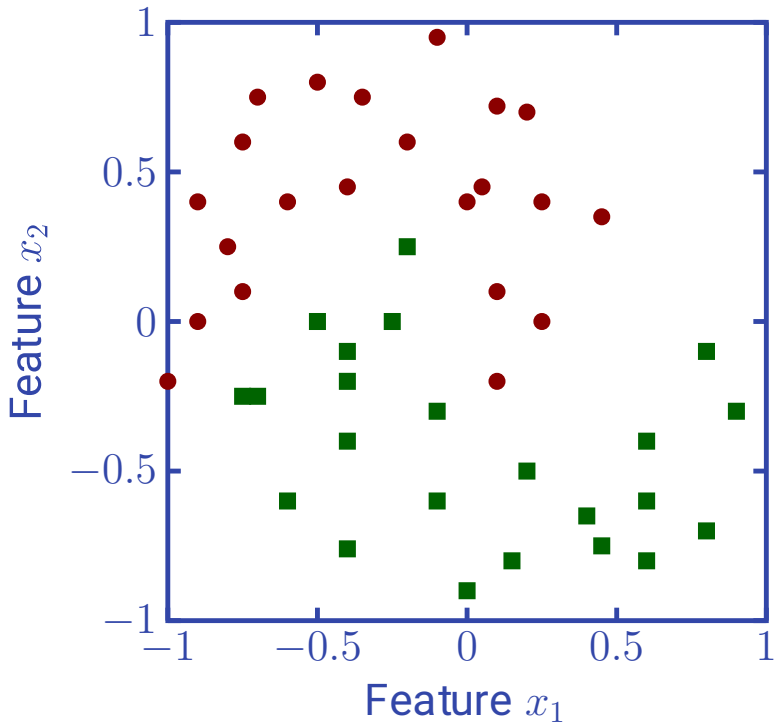


a) Dataset **"Circle"**

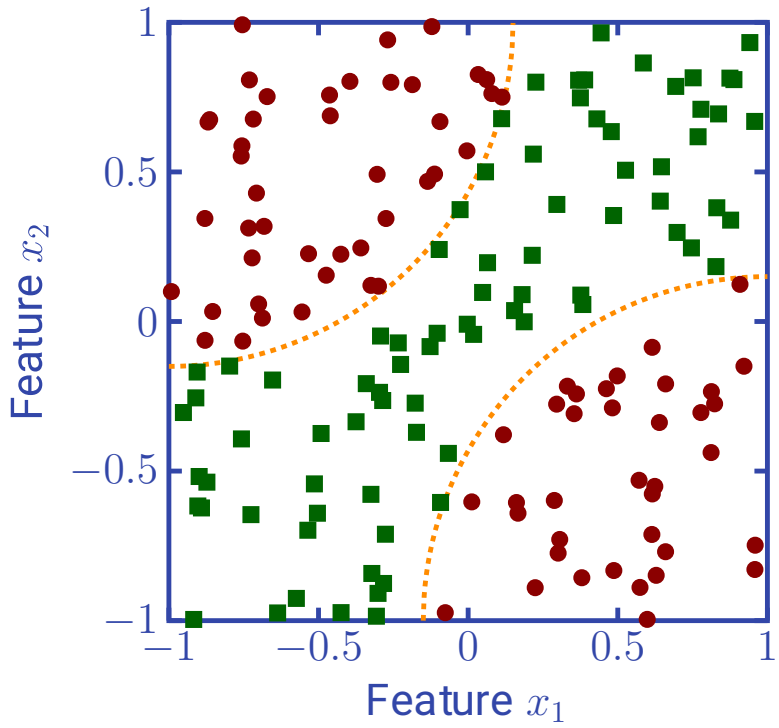


b) Dataset **"Exponential"**

Feature Maps for other Class Positions than Quarter Circles 2/2



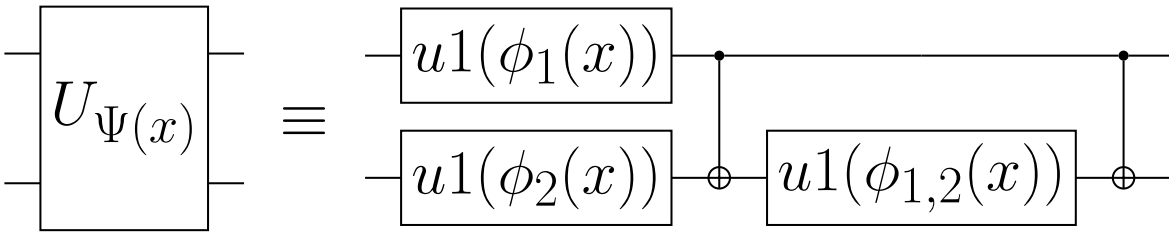
c) Dataset **"Moon"**



d) Dataset **"XOR"**

Feature Maps - Higher-Order Encodings

- Using non linear functions and entanglement for feature map [Y+'20]



$$u1(\phi) = diag(1, e^{-i\cdot\phi}) = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\cdot\phi} \end{bmatrix}, \phi_1(x) = x_1, \phi_2(x) = x_2$$

	1.	2.	3.	4.	5.
$\phi_{1,2}(x)$	$\pi \cdot x_1 \cdot x_2$	$\frac{\pi}{2} \cdot (1 - x_1) \cdot (1 - x_2)$	$e^{\frac{\lvert x_1 - x_2 \rvert^2 \cdot ln(\pi)}{8}}$	$\frac{\pi}{3 \cdot cos(x_1) \cdot cos(x_2)}$	$\pi \cdot cos(x_1) \cdot cos(x_2)$

- More general forms like **ZZFeatureMap** and **PauliFeatureMap** (with a given number of repetitions) for more qubits

Feature Maps - Higher-Order Encodings

- Classification accuracy achieved by quantum SVM [Y+'20]
  Training Accuracy (= number of correct predictions / total number of predictions):

Encoding Function	Dataset *(similar to)*			
	Circle	Exponential	Moon	XOR
1.	1.00	0.91	0.85	1.00
2.	1.00	0.93	0.96	0.97
3.	1.00	0.97	0.91	0.93
4.	1.00	0.98	1.00	0.95
5.	1.00	0.94	0.98	0.93

Test Accuracy:

Encoding Function	Dataset *(similar to)*			
	Circle	Exponential	Moon	XOR
1.	0.97	0.83	0.85	0.99
2.	0.96	0.89	0.87	0.96
3.	1.00	0.92	0.86	0.91
4.	1.00	0.88	0.92	0.89
5.	1.00	0.92	0.87	0.88

Machine Learning Ingredients: Model

- How good is your model performing?
  $\Rightarrow$ A **cost function** $C(\hat{y}, y)$ scores the model based on model predictions and true value
  - $\hat{y} = f(x, \Theta)$ prediction obtained from our model and
  - y ground truth
  - Alias: loss function, error function
- Usually: **Minimize** cost function!
- Widely used cost functions
  - Mean Squared Error (MSE): $MSE = (\hat{y} - y)^2$
    - being symmetric: an error above the target causes the same loss as the same magnitude of error below the target
  - Cross-Entropy Loss Function $L_{CE} = -\sum y_{o,c} \cdot log(p_{o,c})$, where
    - binary indicator (0 or 1) if label c is the correct classification for observation o
    - predicted probability observation o is of class c
    - Penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences tending to 0

Machine Learning Ingredients: Model

- Classification
  - Binary Classification
    - ✉ $\xrightarrow{?}$ 🗑 SPAM-Detection: use emails seen so far to produce rules to predict whether future emails are spam or not
  - Multi-Class Classification
    - ✍ → 🗎 Optical character recognition (OCR) in handwritten text: Learn patterns from data and classify new data
  - Multi-Label Classification
    - ✉ → 👥 Text Classification: assign customer communications to employees in different departments
- Regression
  - 🌡 Weather Forecast: Predicting values based on relevant information
- Clustering
  - 🛒 Identify consumers with similar behavior (e.g. for marketing purposes)
- Generate data
  - 🗎→🖼 Learn patterns or structure from data and reproduce them (e.g. generate pictures from descriptions, compose music)

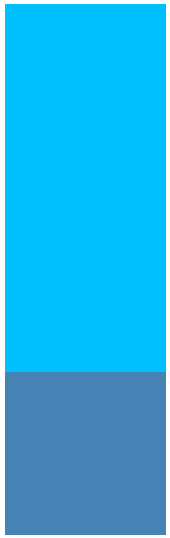
Supervised Learning: Data

- Classify whether the picture contains a cat or dog based on certain features of animals:
  1. Dataset: $\{(x_1, y_1), ..., (x_N, y_N)\}$
     - x_i : features
     - y_i : labels (i.e., cat or dog)

$$x_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,k} \end{bmatrix} y_i = 1 \quad x_j = \begin{bmatrix} x_{j,1} \\ x_{j,2} \\ \vdots \\ x_{j,k} \end{bmatrix} y_i = -1$$

Dataset with k features like weight, tail length, ...

- Split the dataset into the training and test datasets



70% Training dataset with t samples

30% Test dataset with u samples

$$N = t + u$$

Supervised Learning: Model

- Classify whether the picture contains a cat or dog based on certain features of animals:
  2. Linear Model

     $$\hat{y} = f(x, \Theta) = \Theta^T \cdot x$$

     Dimensionality of the vectors: k (=number of features)

     Estimating the predicted label $\hat{y}$ by feeding a sample from the dataset and comparing $\hat{y}$ versus ground truth value y from the dataset

Supervised Learning: Cost

- Classify whether the picture contains a cat or dog based on certain features of animals:
  3. For the t samples in the training dataset compute cost with
     a. Mean Squared Error (MSE): $C_{MSE} = \frac{1}{t} \cdot \sum_{i=1}^{t} (\hat{y}_1 - y_i)^2$
     b. Binary classification (e.g., "cat" or "dog") using cross-entropy loss:

$y_i \in \{0, 1\}$, i.e., "true" label or two classes like **"cat"** and **"dog"**

Probability for $y_i = 1$ (e.g., "cat")

$$C_{CE}(\Theta, (y_1, \dots, y_t)) = -\frac{1}{t} \sum_{i=1}^{t} \left(y_i \cdot log(f(x_i, \Theta)) + (1 - y_i) \cdot log\left(1 - f(x_i, \Theta)\right) \right)$$

Probability for $y_i = 0$ (e.g., "dog")

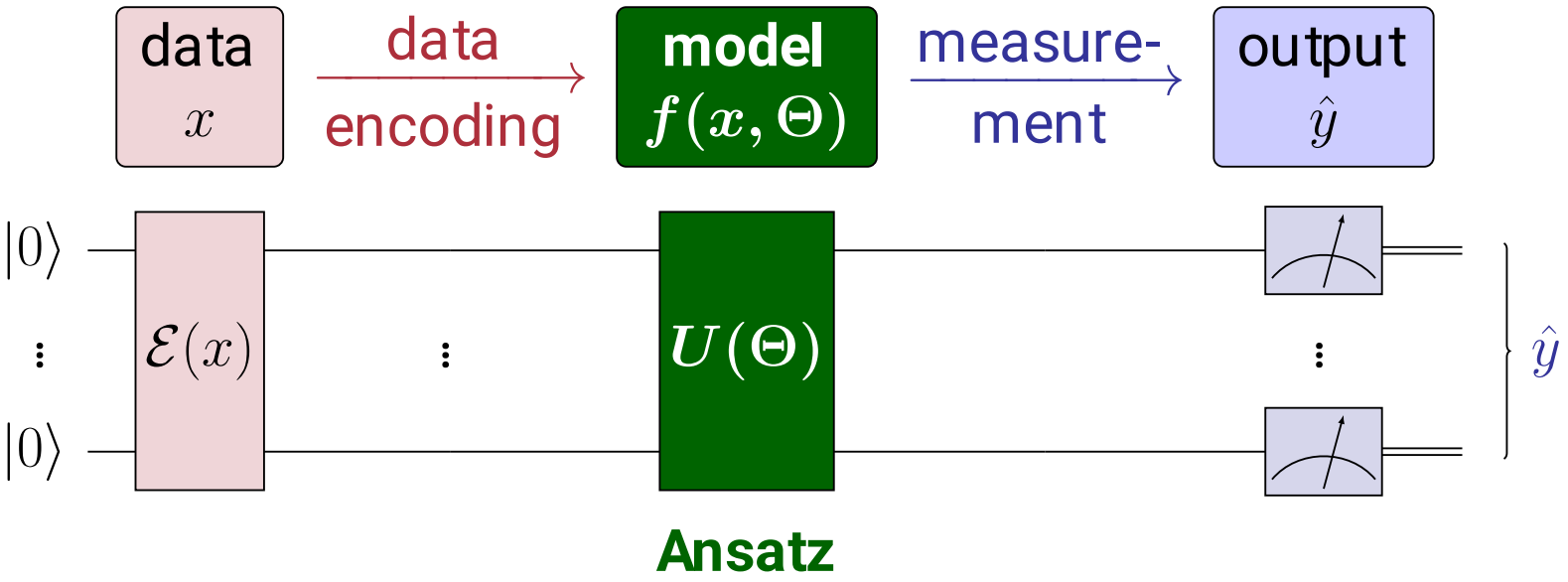
Case $y_i = 1$, e.g. "cat"

Case $y_i = 0$, e.g. "dog"

  4. ...

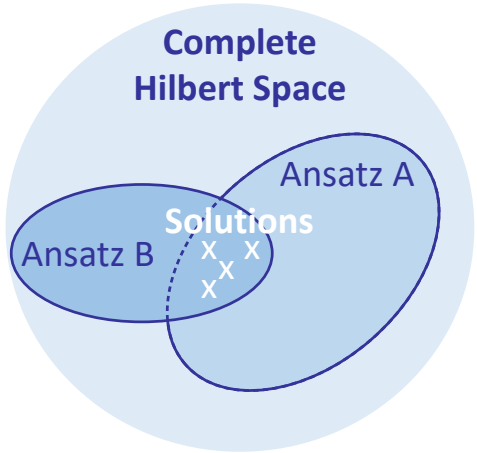
Minimize the chosen cost function to have a good model $f(x_i, \Theta)$

How can we build a quantum model?

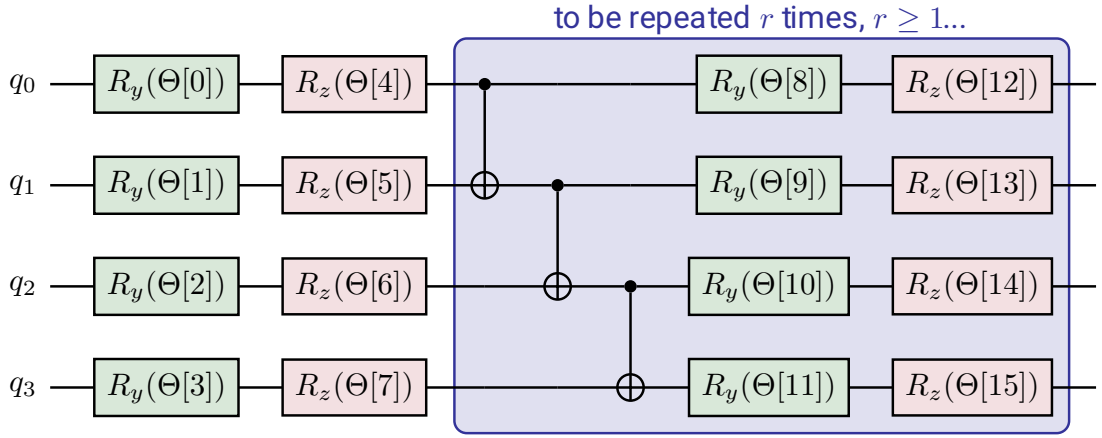


How can we build a quantum model?

Solution Space



Hardware-efficient Ansatz



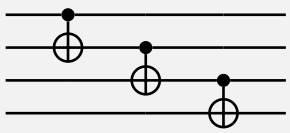
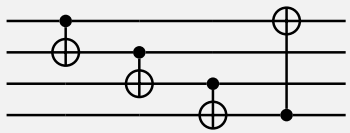
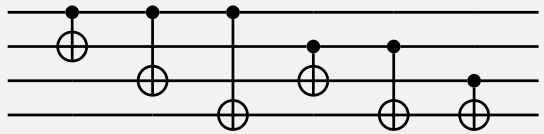
Problem-specific ansatz

Use knowledge about problem to choose ansatz!

Example: Does the wave function of the quantum states only have real amplitudes?

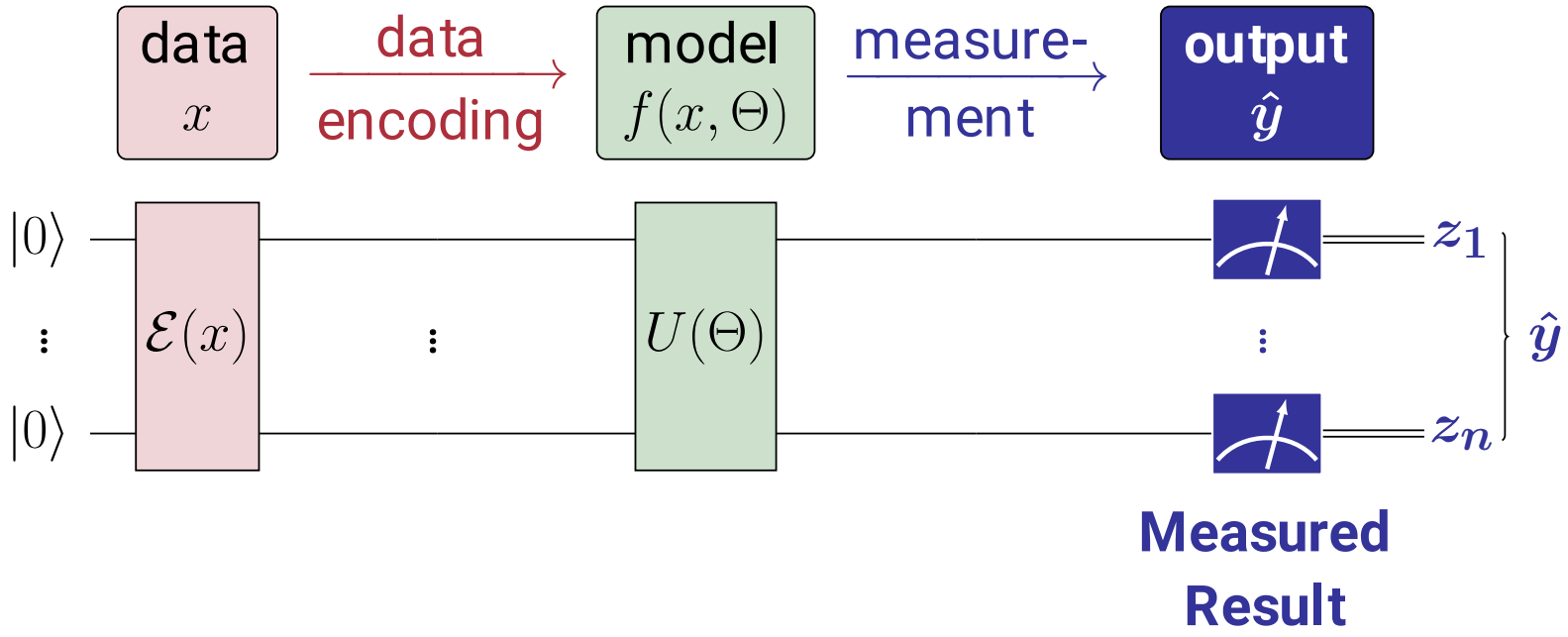
Many different quantum circuits for machine learning models, see [B+'19]

Variants

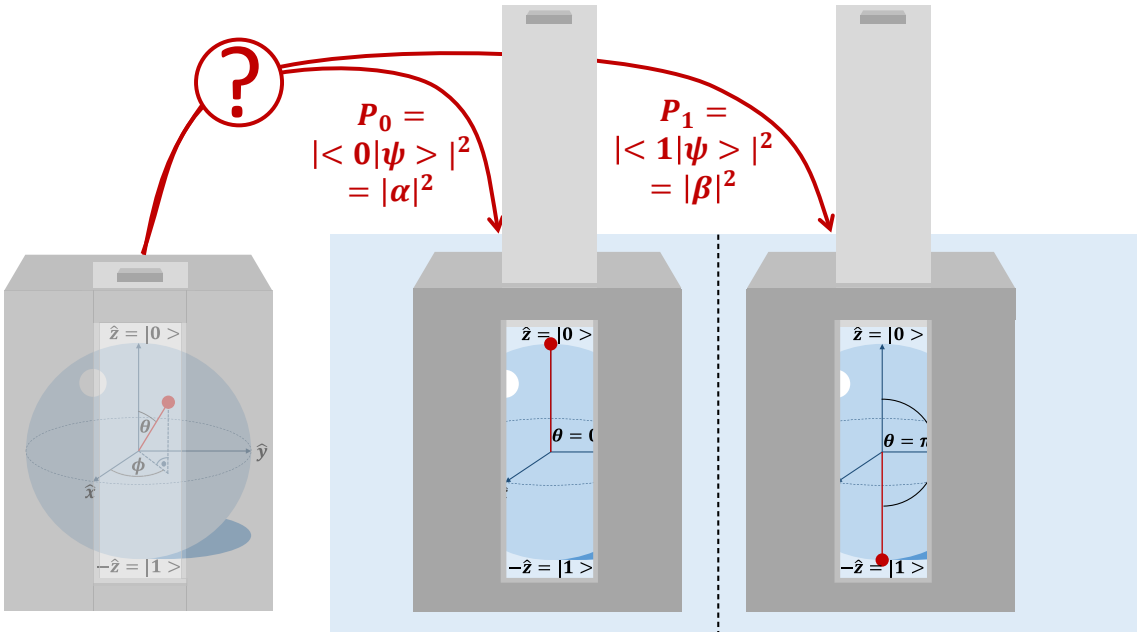
Entanglement Layer	Linear	Circular	Fully Connected
Circuit			
#Gates for n qubits	$n - 1$	n	$\binom{n}{2}$

- Different combinations of x ,y and z rotation gates in the rotation layer
- In variants rotation layer and entanglement layer are combined into one by using controlled rotation gates

Retrieving the result of a Quantum Model - Measurement



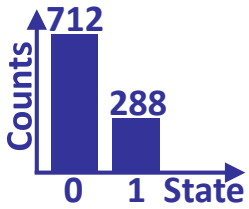
Result of a Quantum Model - Measurement



$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$
$$|\alpha|^2 + |\beta|^2 = 1$$

Measure x samples:

e.g., x =1000:



Expectation value:

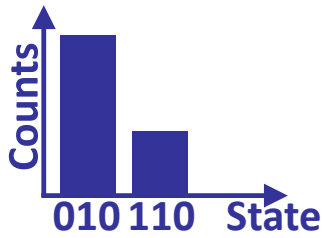
Average value of all results of a quantum operator Z (with eigenvectors ϕ_i and eigenvalues a_i) applied to a state ψ:

$$\langle Z \rangle_\psi = lim_{n\to\infty} \sum \frac{N_i}{N} a_i = \sum |\langle \phi_i | \psi \rangle|^2 a_i = \langle \psi | Z | \psi \rangle$$

e.g.: $\langle Z \rangle = 1 \cdot P_0 + (-1) \cdot P_1$

Post-Processing

Sampling-based



mapping

e.g. parity: even or odd number of 1s?

label

0 with P_{010}

1 with P_{110}

Expectation-based

Measure expectation value

$$\langle z_1 \cdots z_n \rangle = -0.5$$

mapping

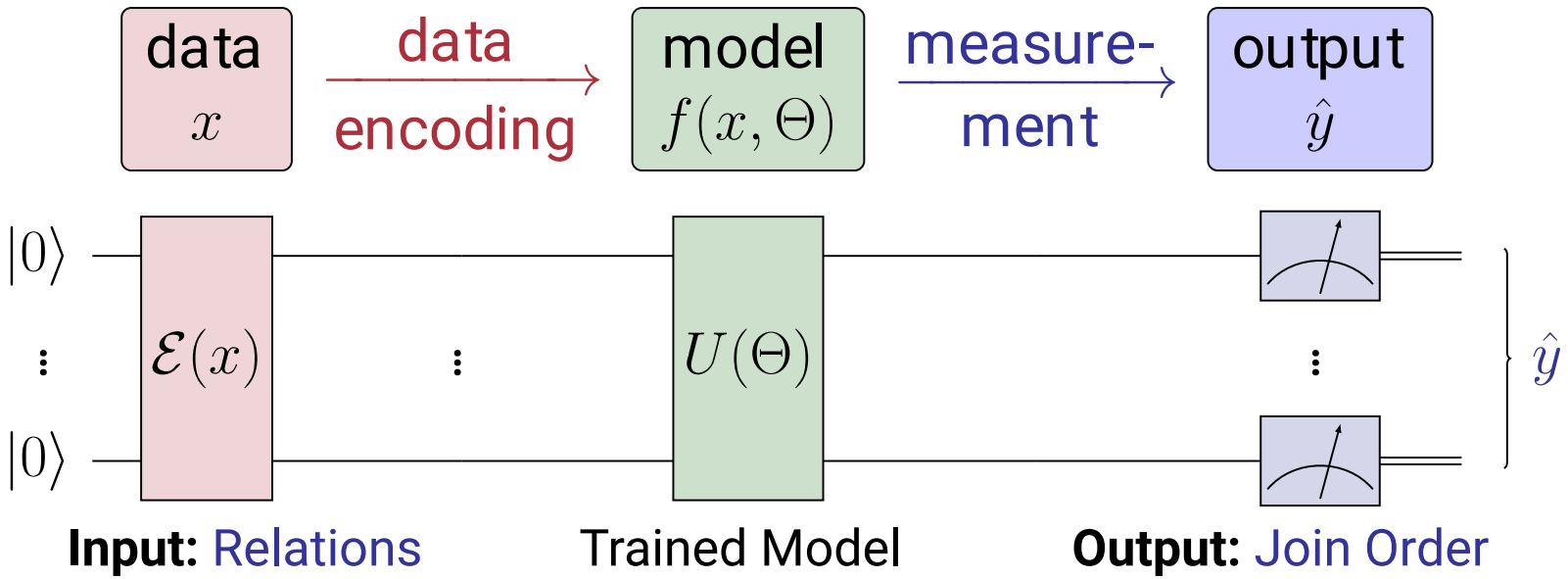
e.g. sign

label

1 with $Pr = \frac{1+0.5}{2}$

Join Ordering with Quantum ML

- Compared to ML: **Learning with fewer data, higher accuracy**



Input: Relations

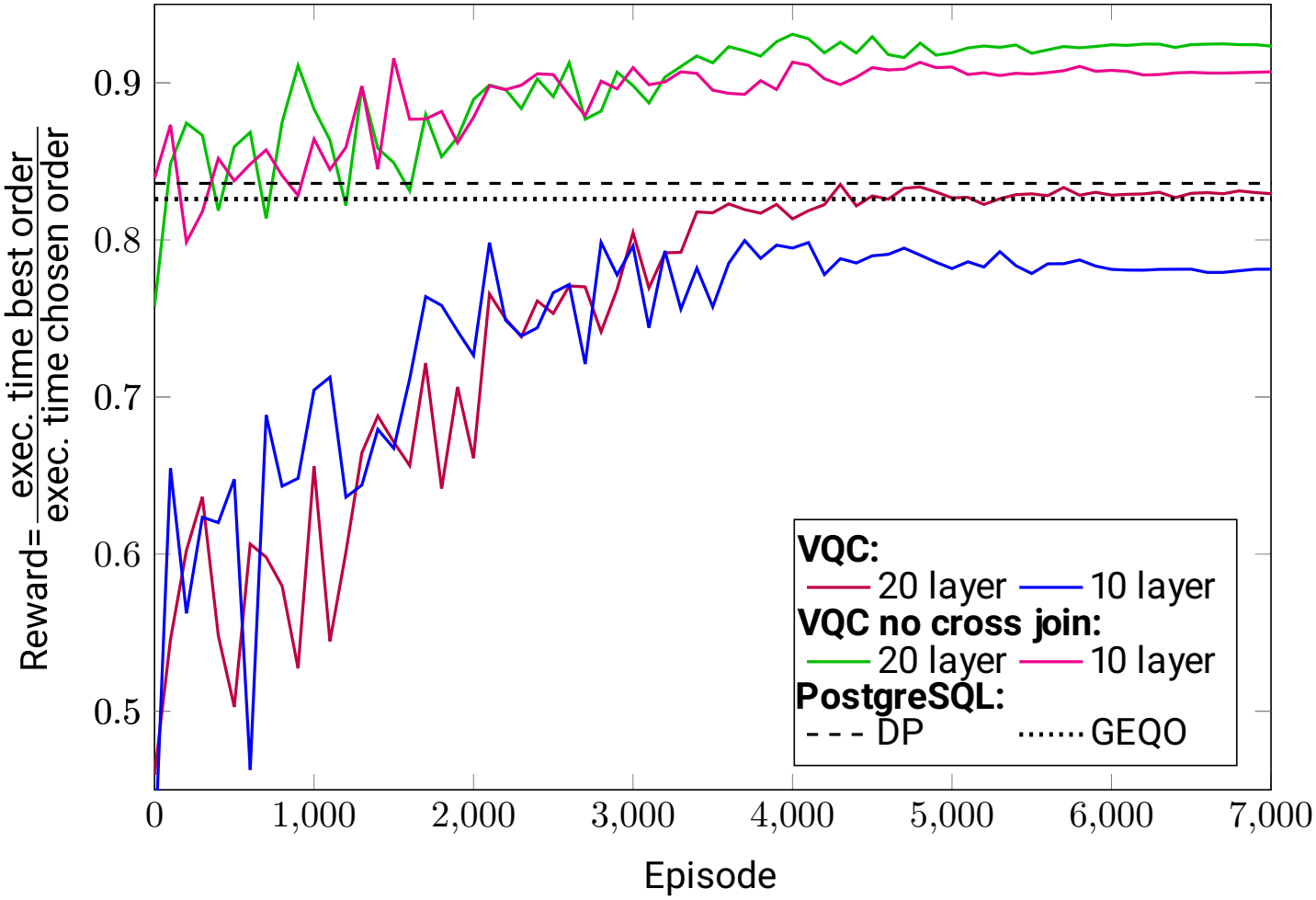
$R_{x_1} \bowtie ... \bowtie R_{x_k} \rightarrow$
$(cos(x_1) \cdot |0\rangle + sin(x_1) \cdot |1\rangle) \otimes ... \otimes (cos(x_k) \cdot |0\rangle + sin(x_k) \cdot |1\rangle)$
with $x_1, ..., x_k$ normalized to $[0, 2 \cdot \pi]$

Trained Model

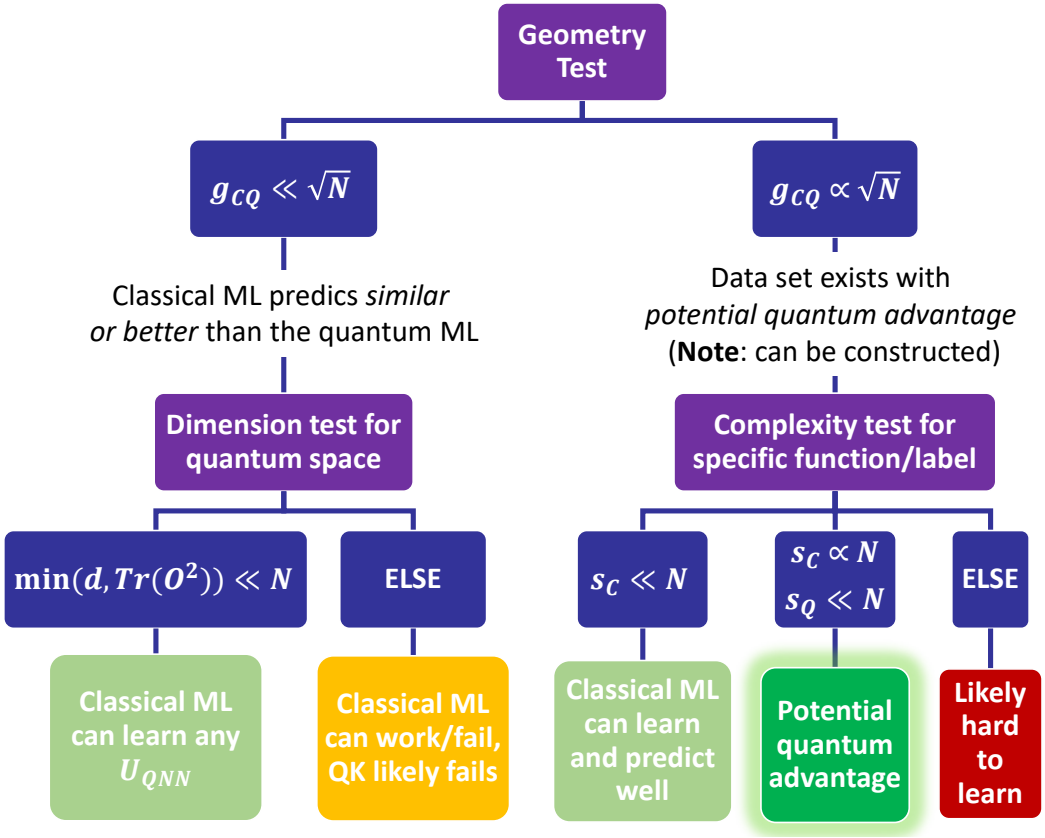
Output: Join Order

$o \in \{0, ..., 2^k - 1\}$: measured value with highest probility
Choose join order $o \mod k$
with k number of valid join orders

Join Ordering with Quantum ML



Is quantum ML better than classical ML?



Legend:

N: Number of samples of data
U_{enc}: encoding circuit
U_{QNN}: function circuit
K^C: kernel of classical ML model
K^Q: kernel of quantum ML model
$g_{CQ} = g(K^C||K^Q)$: geometric difference
s_C: model complexity of classical ML model
s_Q: model complexity of quantum ML model
d: dimension/number of features
$Tr(O^2) = \sum_{i=1}^{d}\sum_{j=1}^{d}|a_{ij}|^2$: Trace of observable O^2 being $d \times d$ matrix
Note: Most observables like Pauli operators have exponentially large $Tr(O^2)$, such that minimum is d in practice
$\alpha \propto \beta$: α becomes small as β becomes too large

Summary & Conclusions

- Machine Learning
  - Basics: Data, Model, Cost
- **Quantum** Machine Learning
  - Data Encoding/Feature Maps
    - Basis Encoding
    - Amplitude Encoding
    - Angle Encoding
    - Higher-Order Encodings
  - Quantum Model
    - Hardware-efficient Ansatz
  - Result of a Quantum Model via Measurement
    - Sampling versus expectation value
    - Post-processing