University of Stuttgart
Institute for Computational Physics

Freie Universität Berlin

SimTech

Simulating Stochastic Processes with Variational Quantum Circuits

Daniel Fink

CQSE & NCTS
Special Seminar
-
National Taiwan University

November 29th, 2022

https://dx.doi.org/10.18419/opus-12068

Advisors

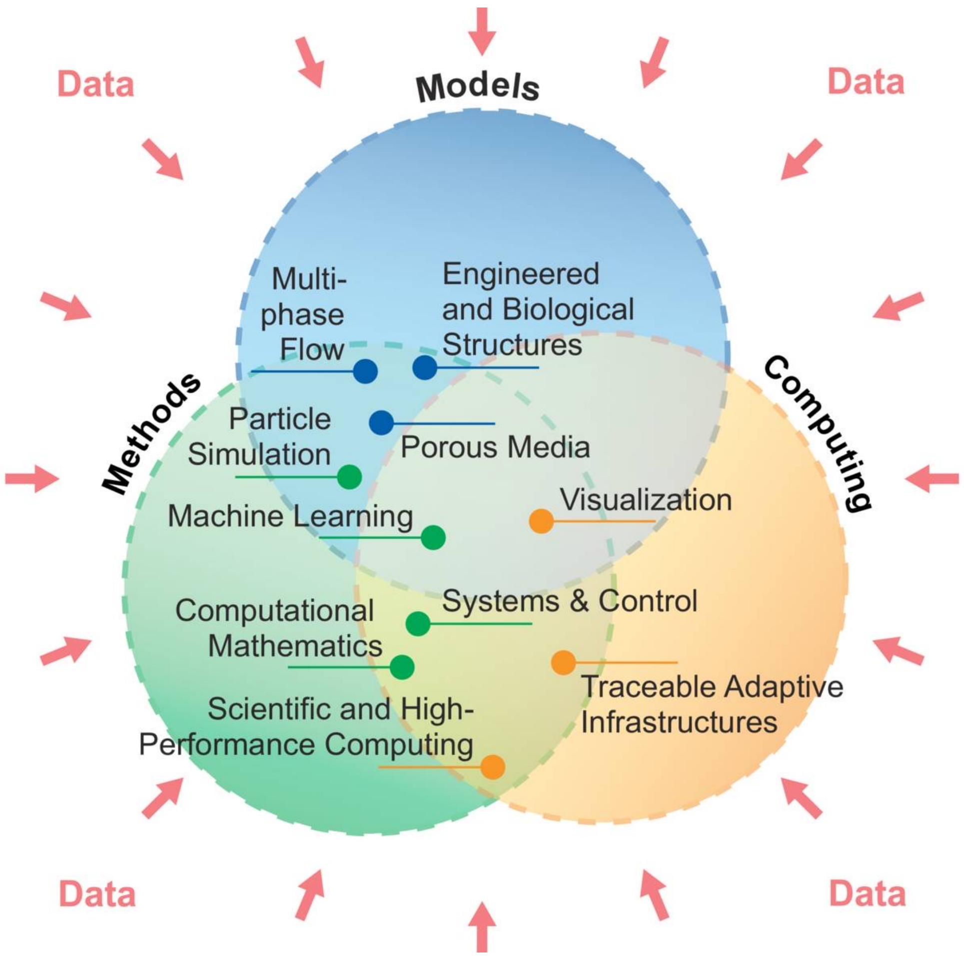
- University of Stuttgart
  - Prof. Dr. Christian Holm
- Free University of Berlin
  - Prof. Dr. Jens Eisert
  - Dr. Nora Tischler
  - Dr. Ryan Sweke
  - M.Sc. Paul Fährmann

What is SimTech?

- Interdisciplinary research unit
- Common target: Simulation
- Professorship, PostDocs, PhDs, …

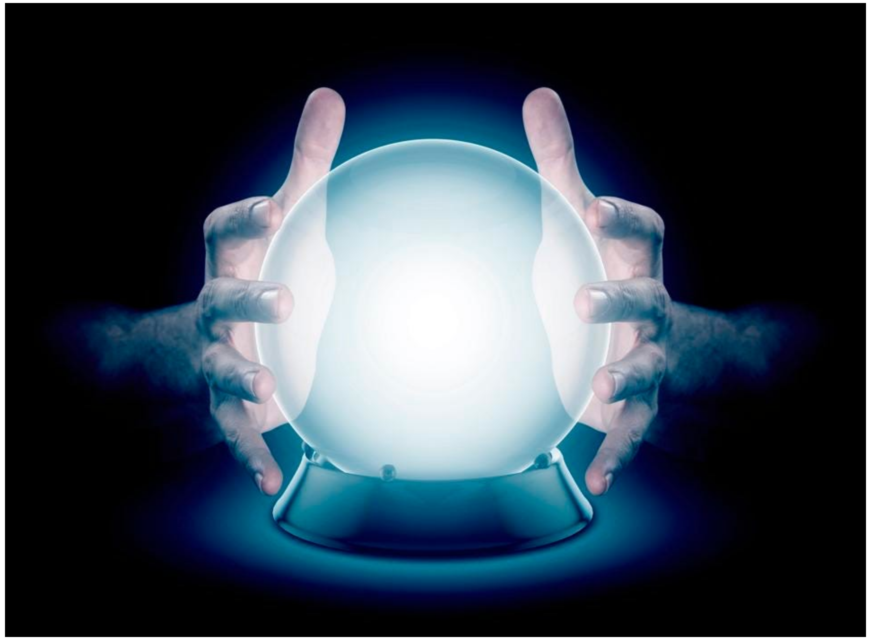
- Project Network for Quantum Computing
- Open positions available
- Funding for exchange available

- Contact: daniel.fink@icp.uni-stuttgart.de



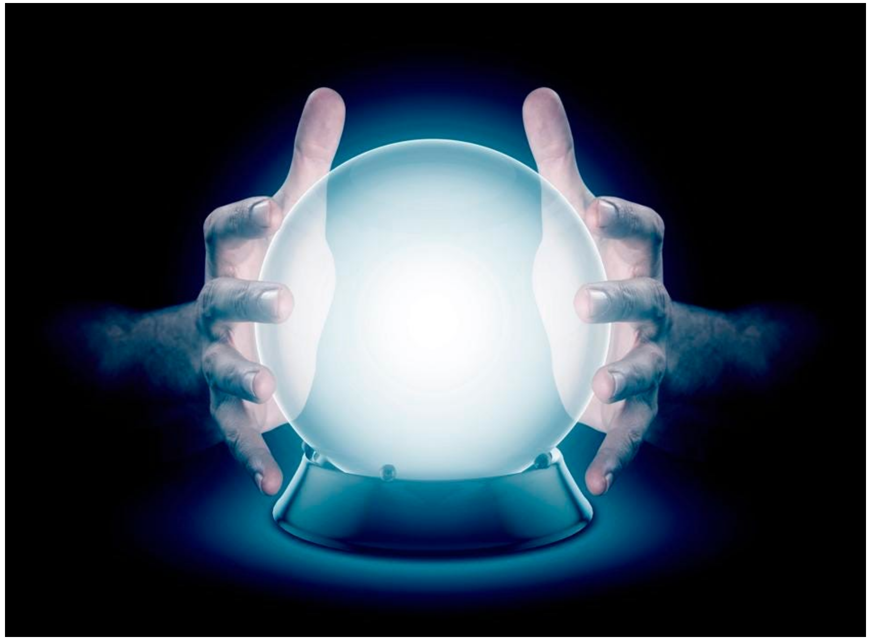
Motivation

Can we predict the future based on past observations?



Motivation

Can we predict the future based on past observations?



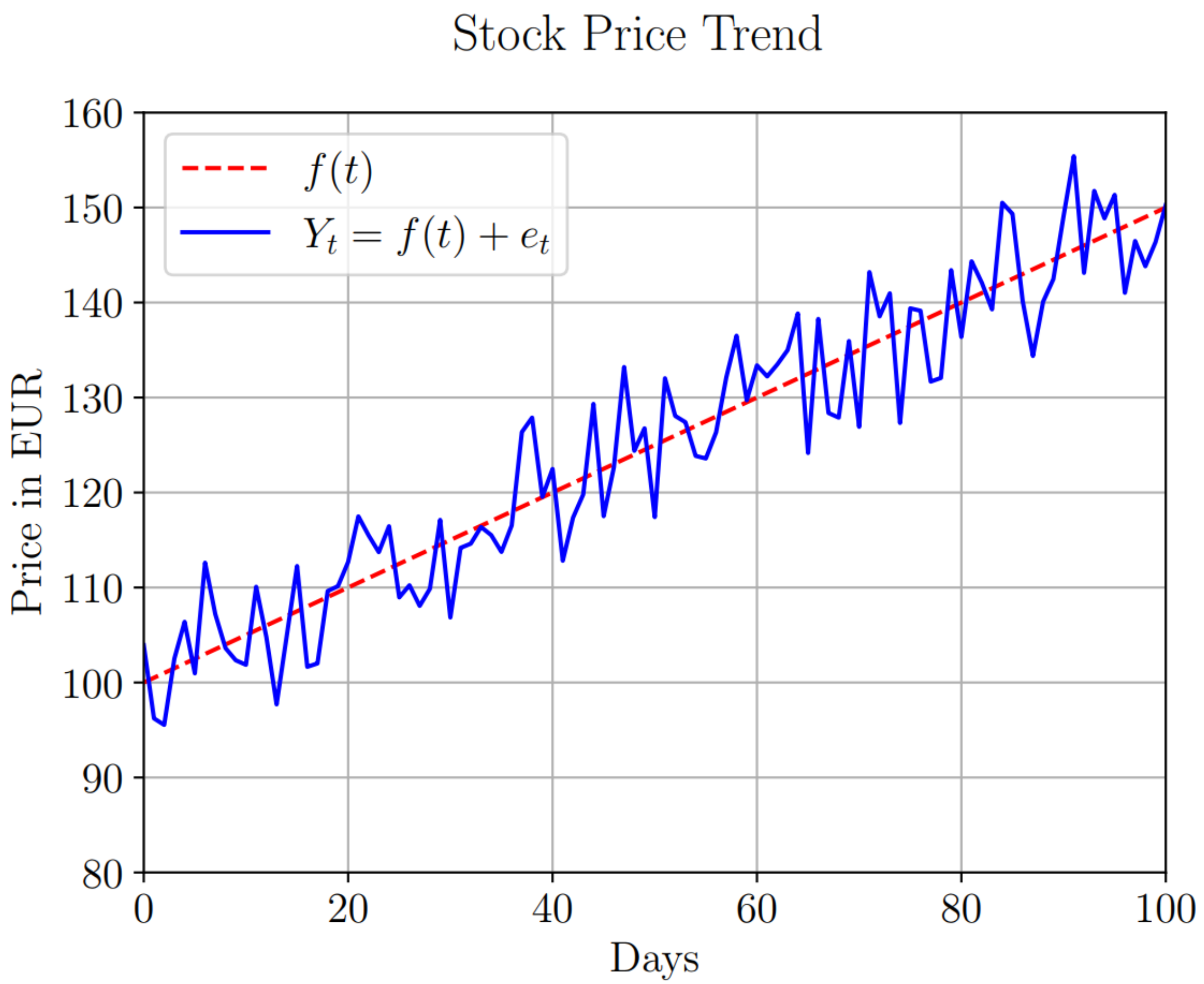
Simulations → show possible futures

Motivation

Assume linear trend $f(t)$

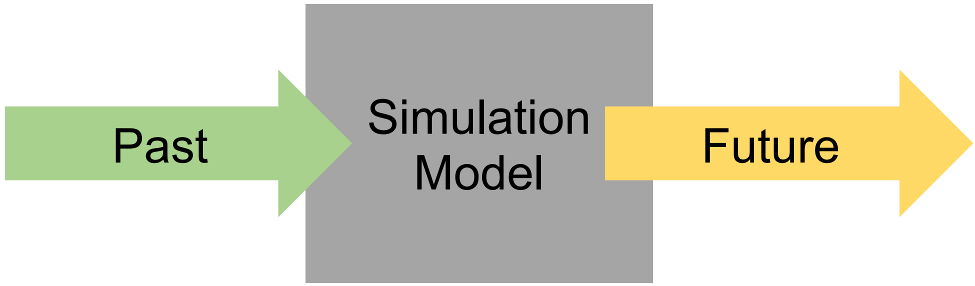
Add some noise e_t

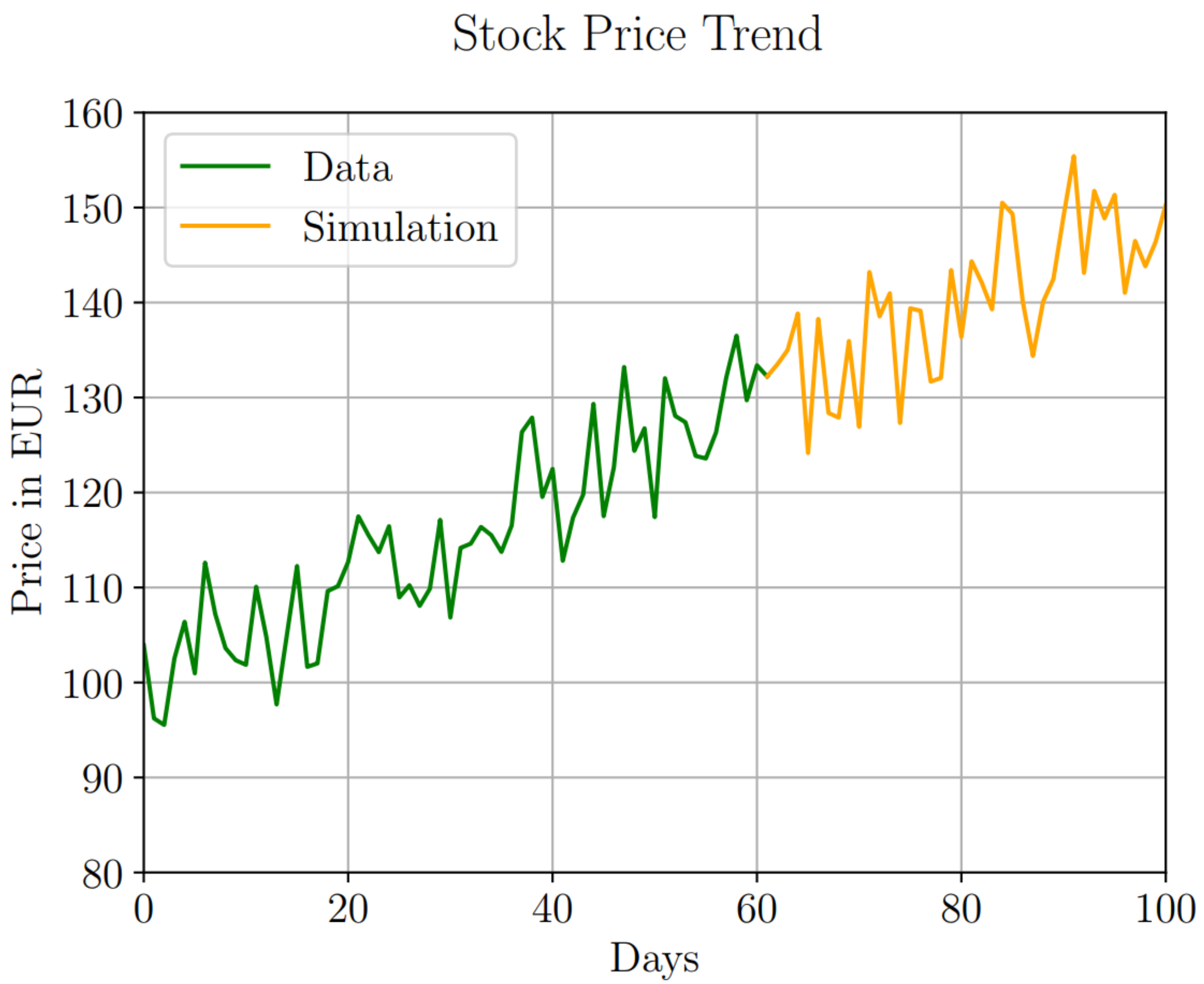
→ e_t is a stochastic process



Motivation

Assume data drawn
by a stochastic process





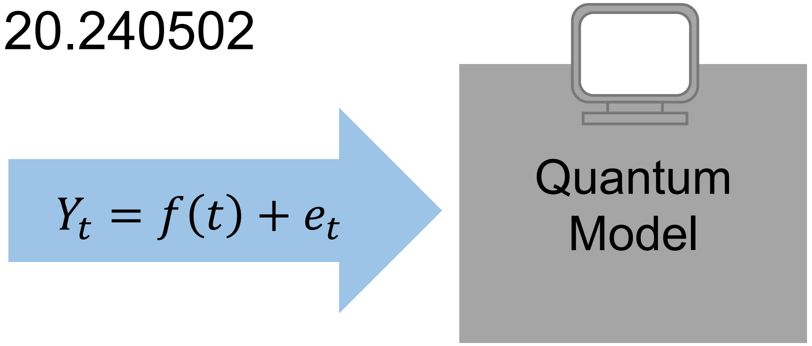
Motivation

Classical Models $\leq$ Quantum Models

How to get a quantum model?

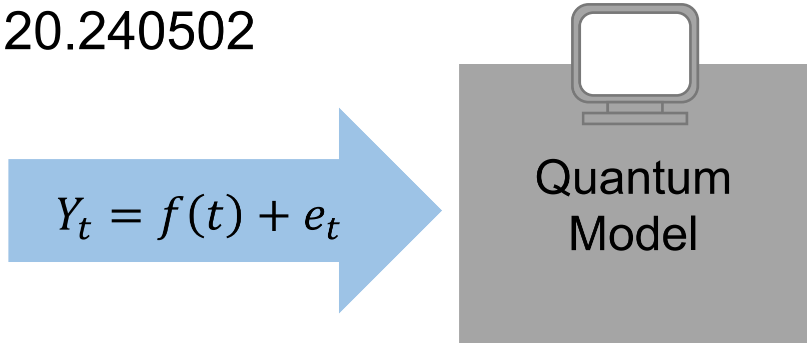
Motivation

- Classical description of the process → q–simulator
  - Binder et al., 10.1103/PhysRevLett.120.240502

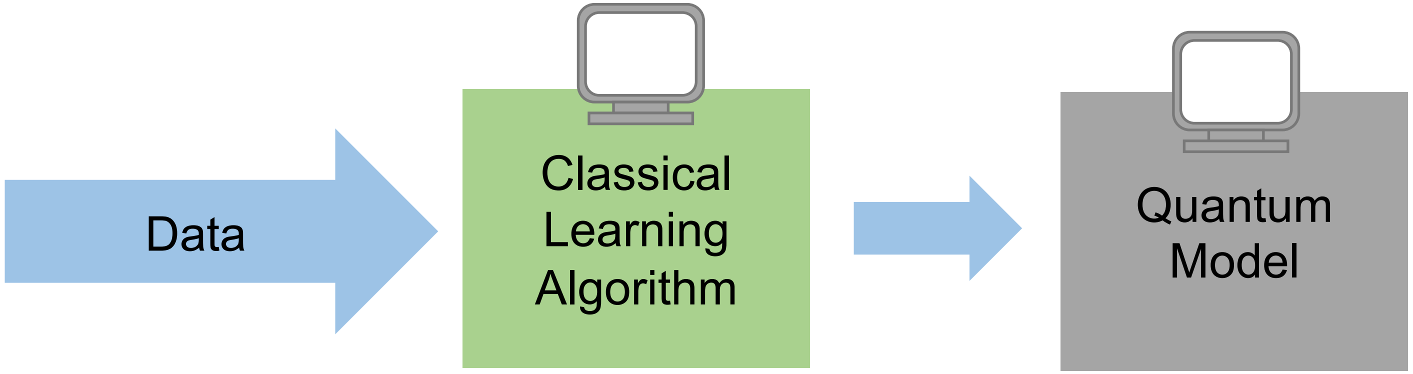


Motivation

- Classical description of the process → q–simulator
  - Binder et al., 10.1103/PhysRevLett.120.240502



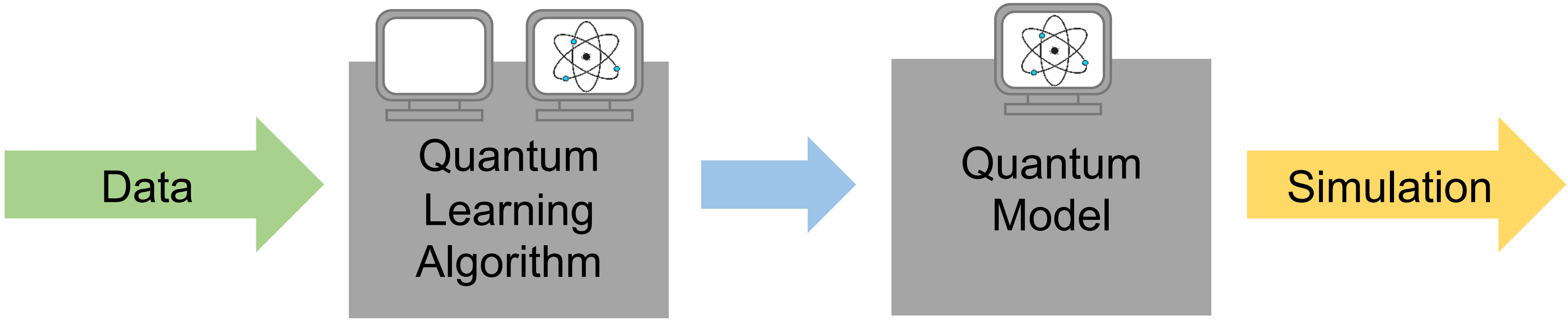
- Data from the process → classical discovery algorithm
  - Yang et al., arXiv:2105.14434



Motivation

Goal

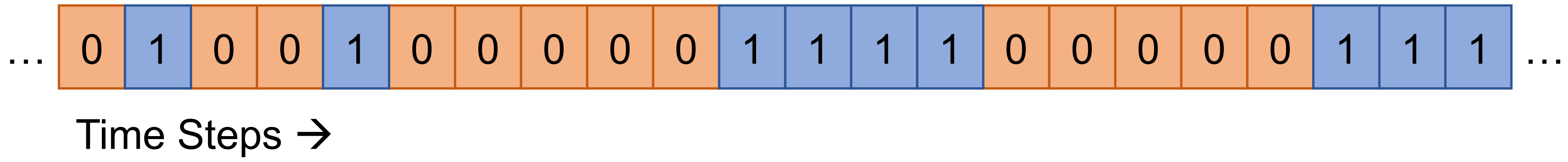
Develop a quantum learning algorithm for simulation models, which uses only data as input.



Content

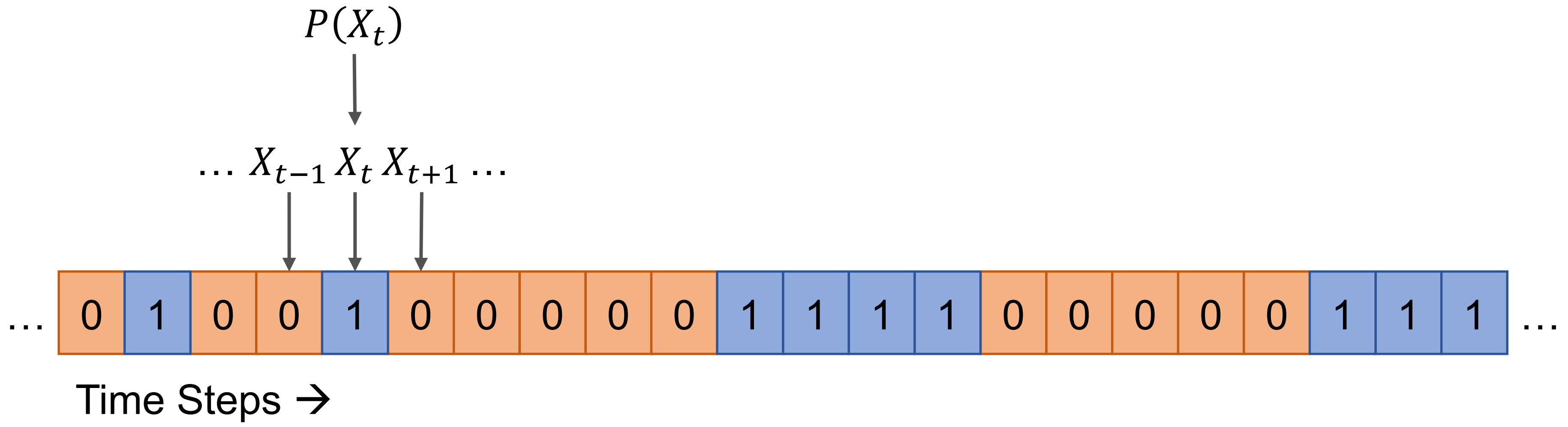
- Stochastic Processes
- ϵ–machine
- Quantum Circuits
- q–simulator
- Quantum Learning Algorithm
- Results
- Conclusion

Stochastic Processes

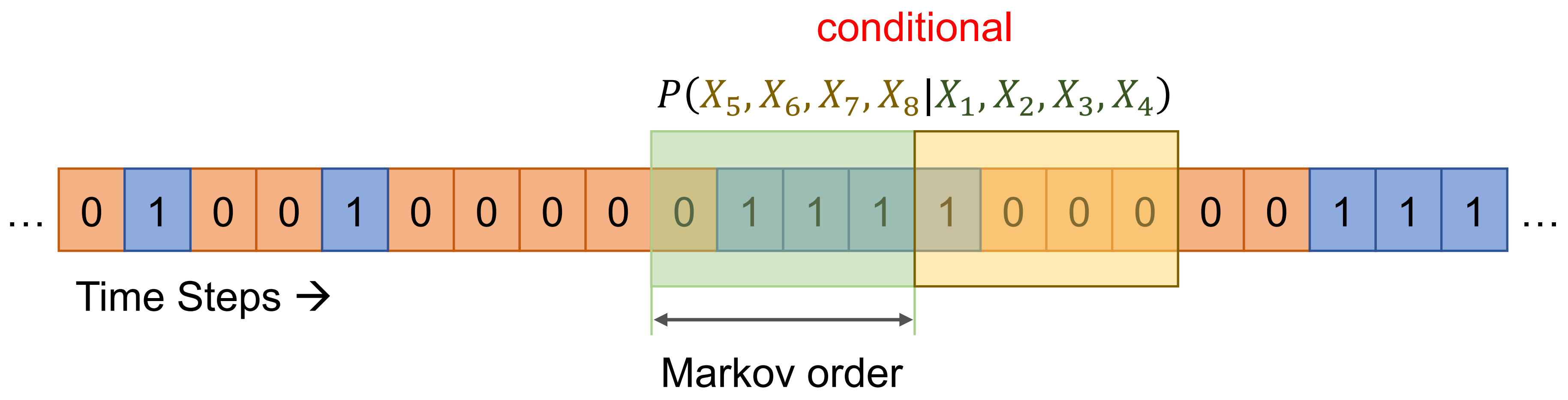


… 0 1 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 …
Time Steps →

Stochastic Processes



Stochastic Processes



Stochastic Processes

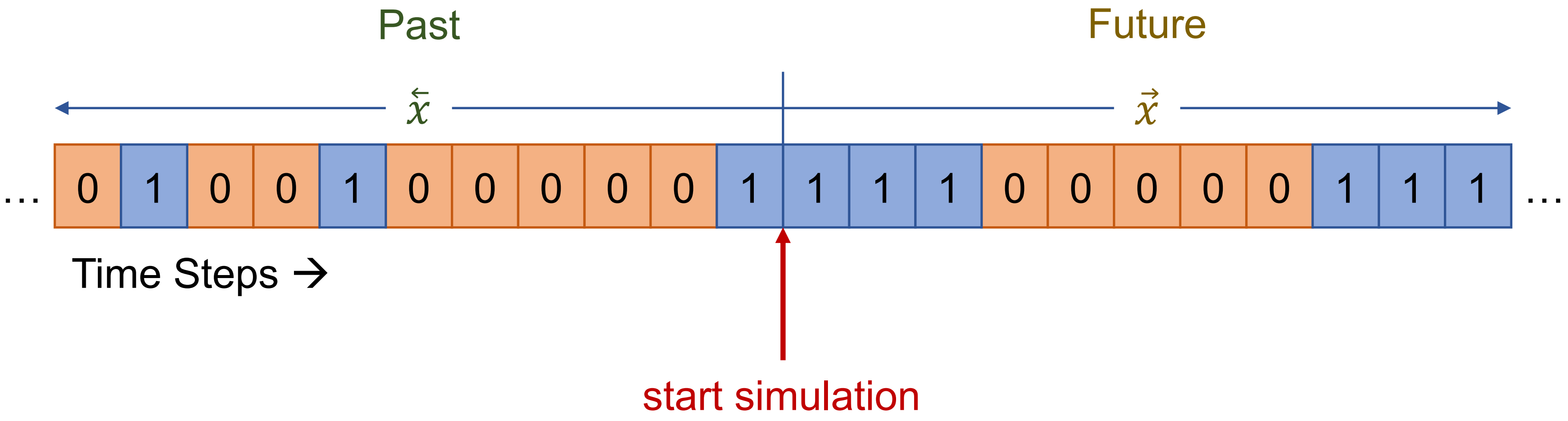
stationary

$$P(X_1, X_2, X_3, X_4) = P(X_5, X_6, X_7, X_8)$$

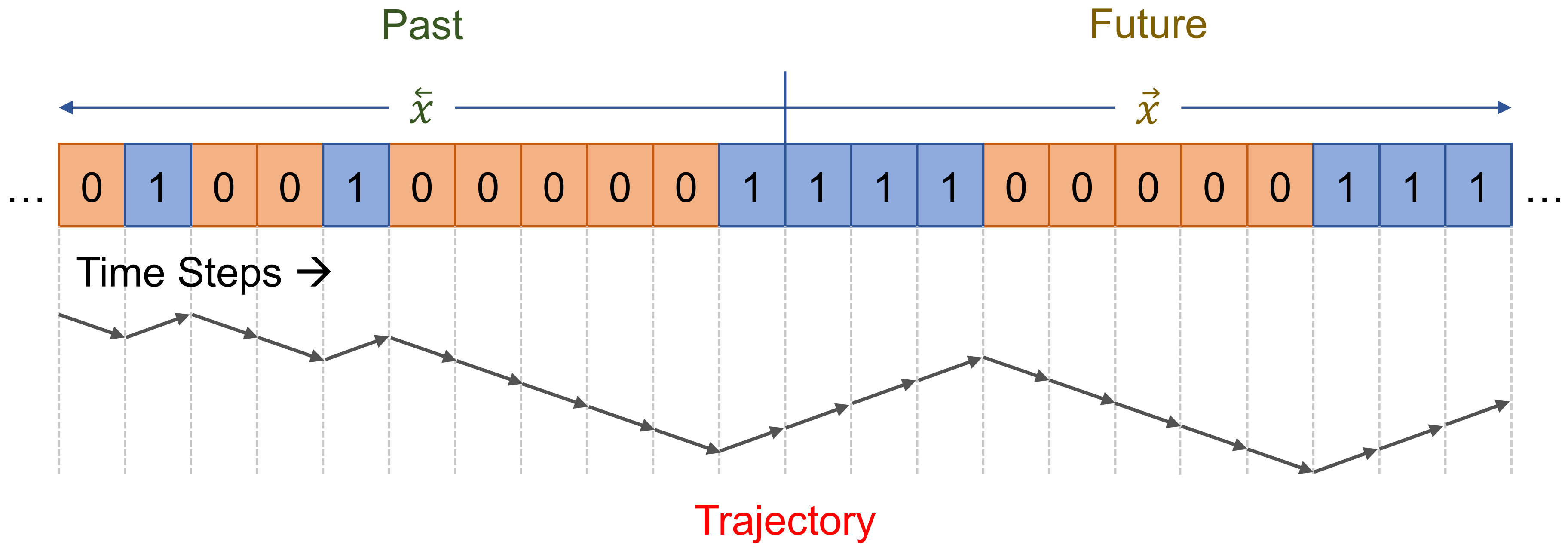
… 0 1 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 …

Time Steps →

Stochastic Processes

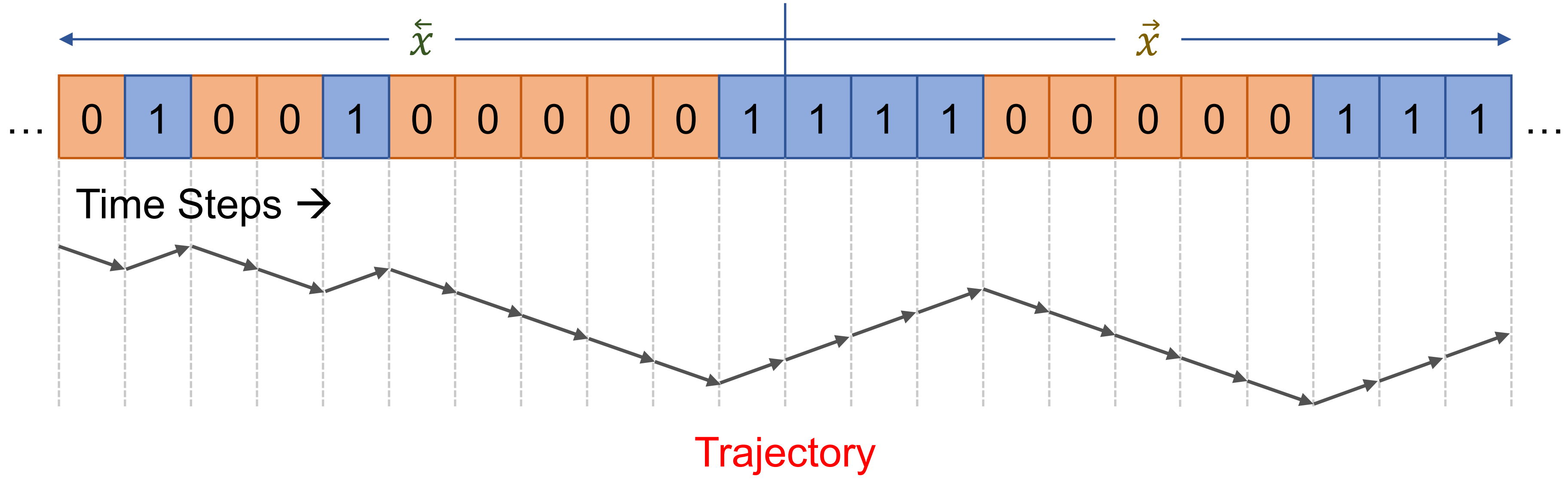


Stochastic Processes



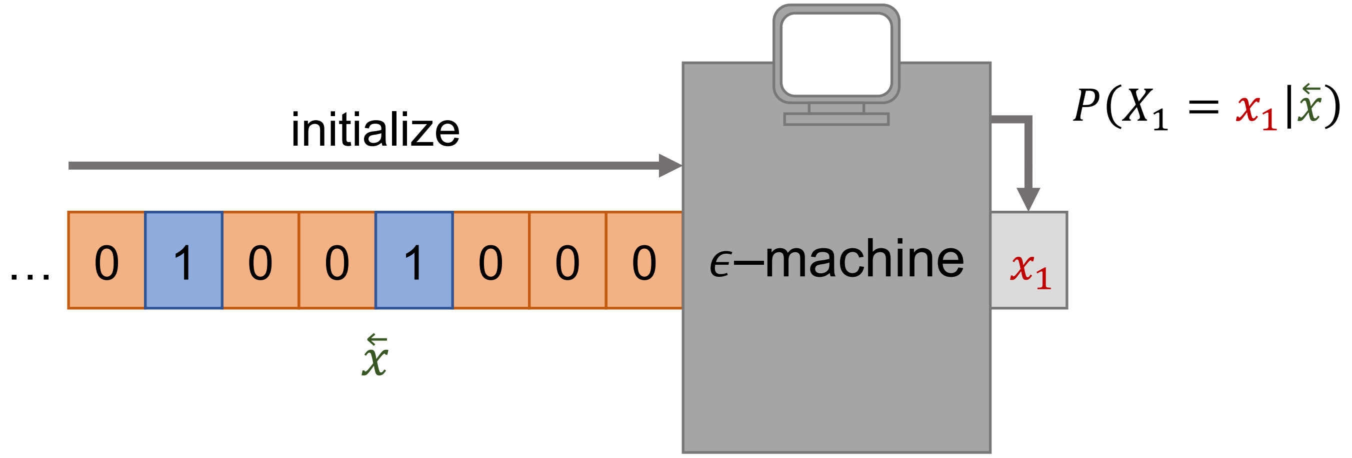
Stochastic Processes

- Simulating = sampling trajectories
- Trajectory is governed by $P(\vec{X}|\overleftarrow{X})$

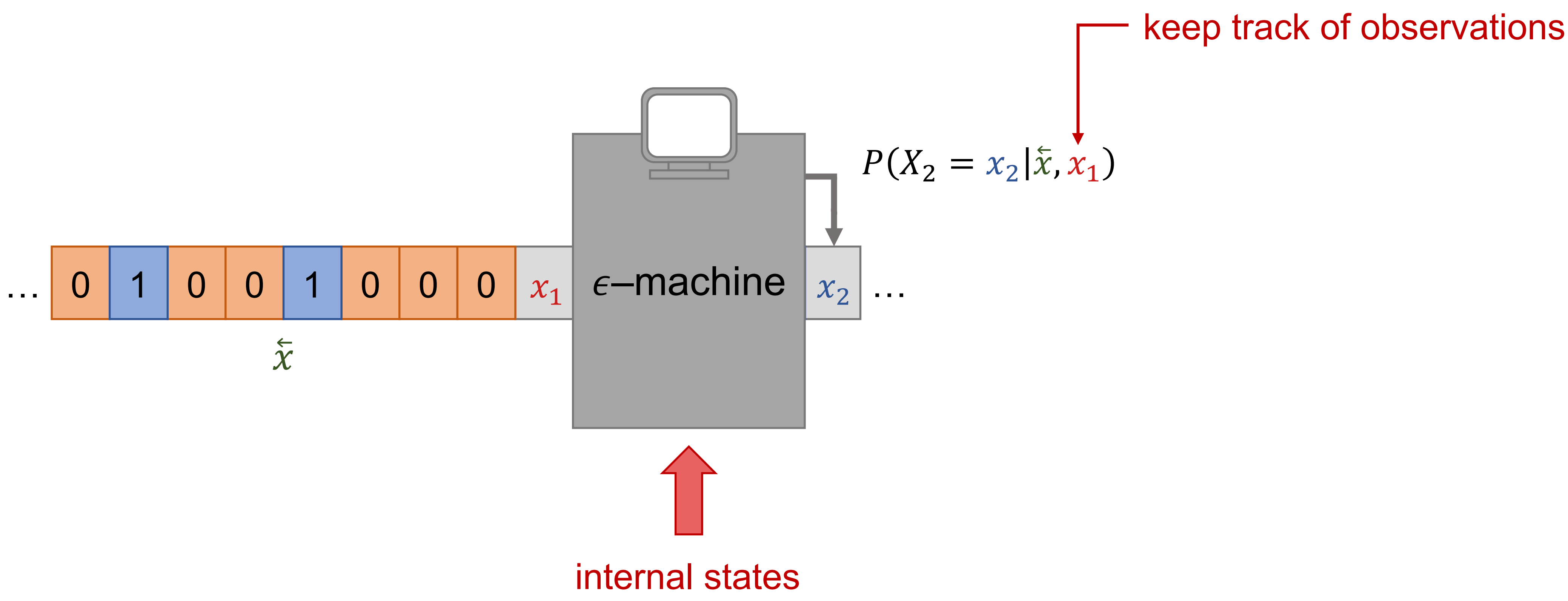


ϵ–machine

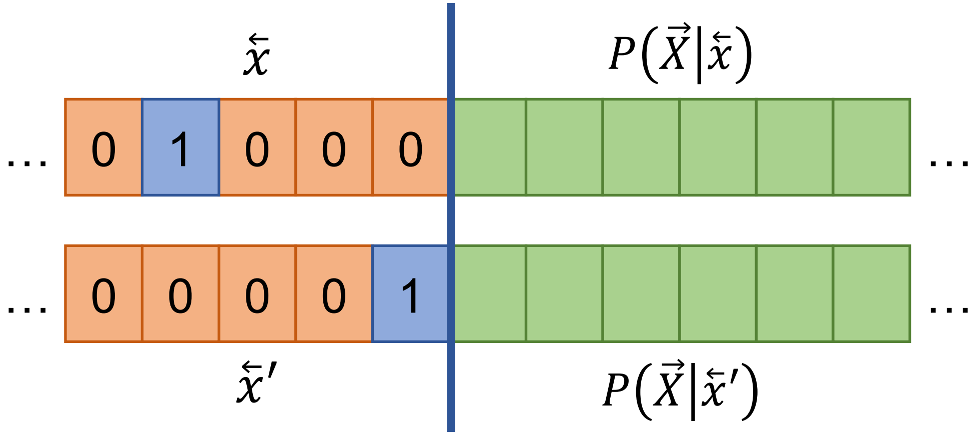
ϵ–machine



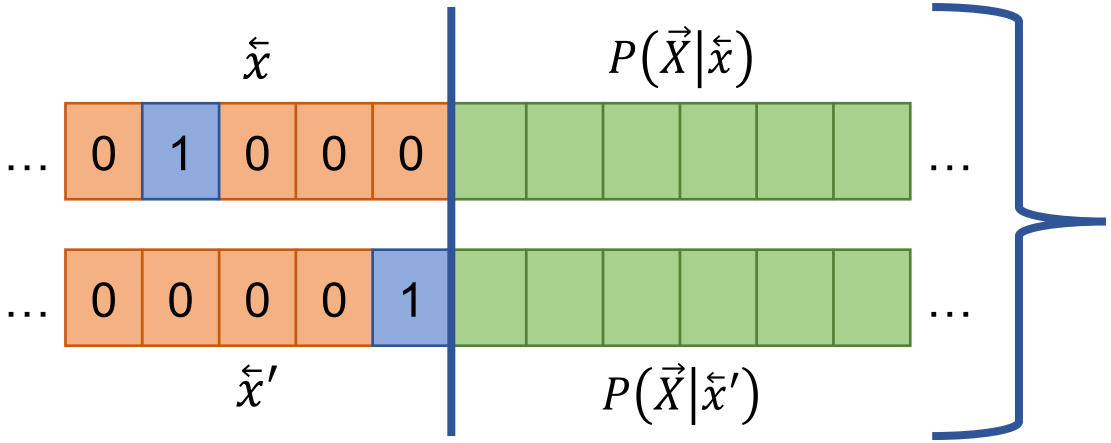
ϵ–machine



ϵ–machine

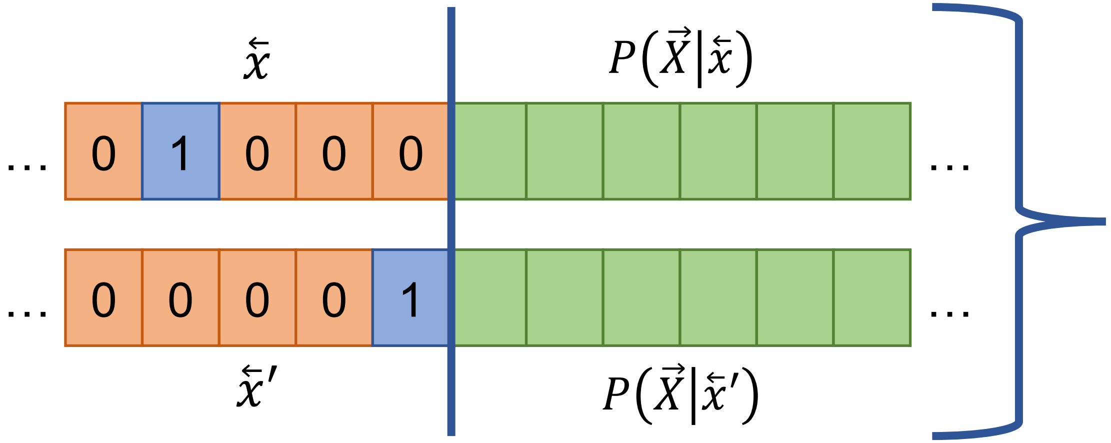


ϵ–machine



$$\overleftarrow{x} \sim \overleftarrow{x}' \iff P(\vec{X}|\overleftarrow{x}) = P(\vec{X}|\overleftarrow{x}')$$

ϵ–machine



$\overleftarrow{x} \sim \overleftarrow{x}' \Leftrightarrow P(\overrightarrow{X}|\overleftarrow{x}) = P(\overrightarrow{X}|\overleftarrow{x}')$

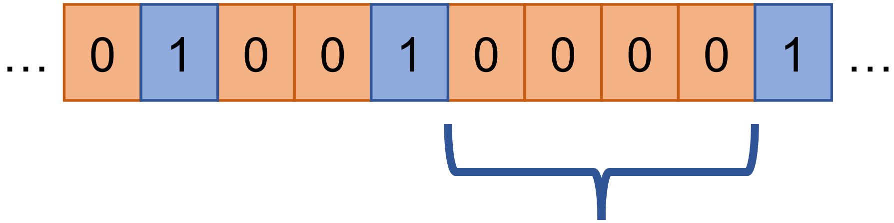
causal sates $\{S_i\}_{i=1}^{N}$ (minimal)

map $E(\overleftarrow{x}) = S_i$ (Initialization)

Classical Topological Complexity: $d_c = \log_2 N$

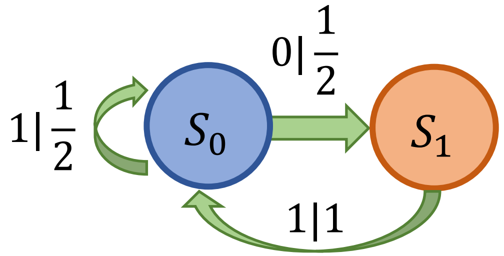
Example

Period-N Uniform Renewal Process



$\sim U(0, N-1)$

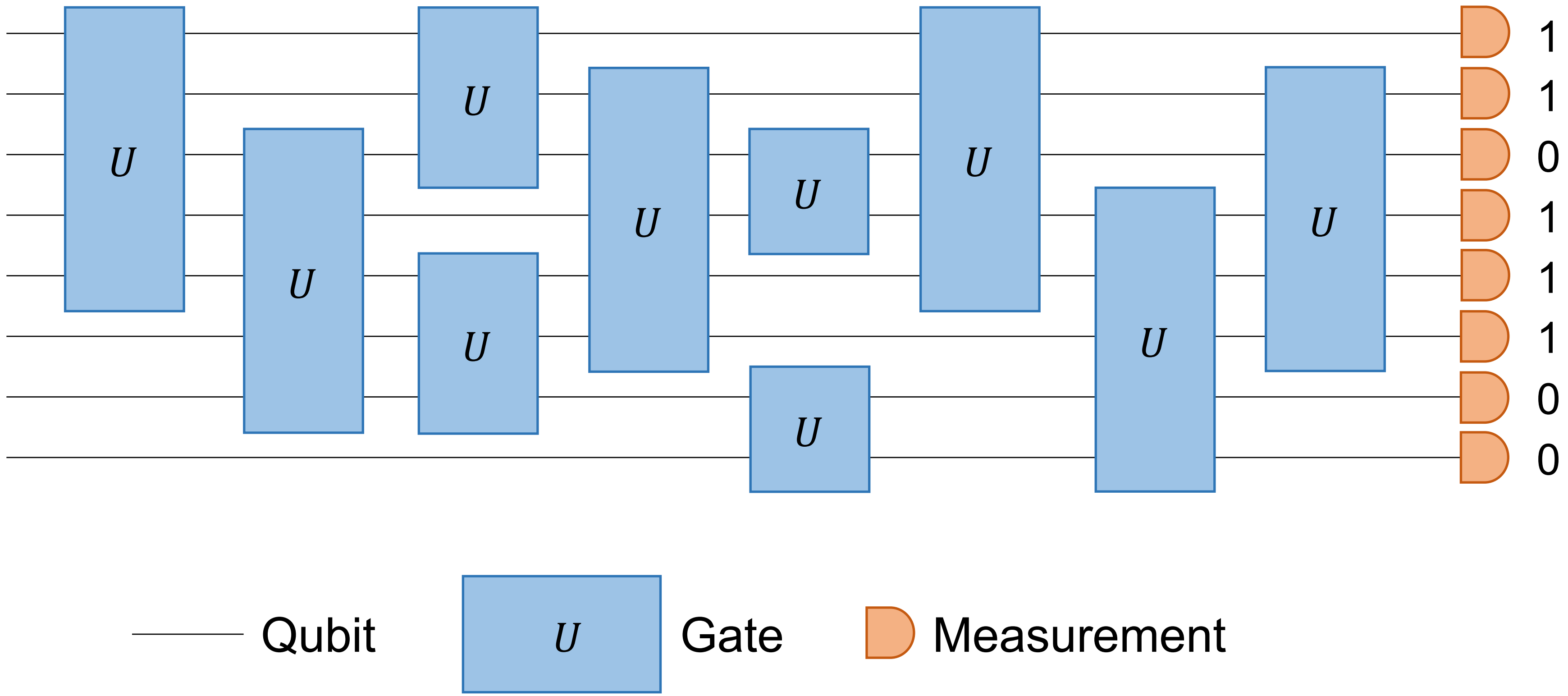
Period = 2



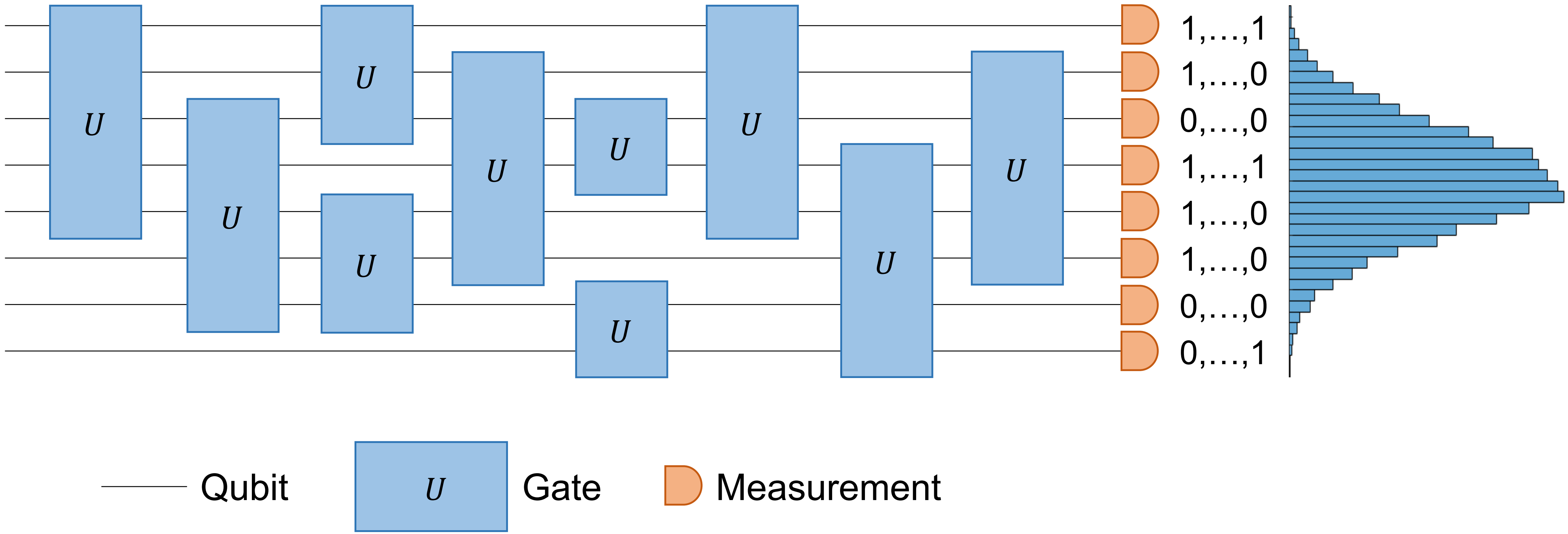
ϵ-machine

Quantum Circuits

Quantum Circuits

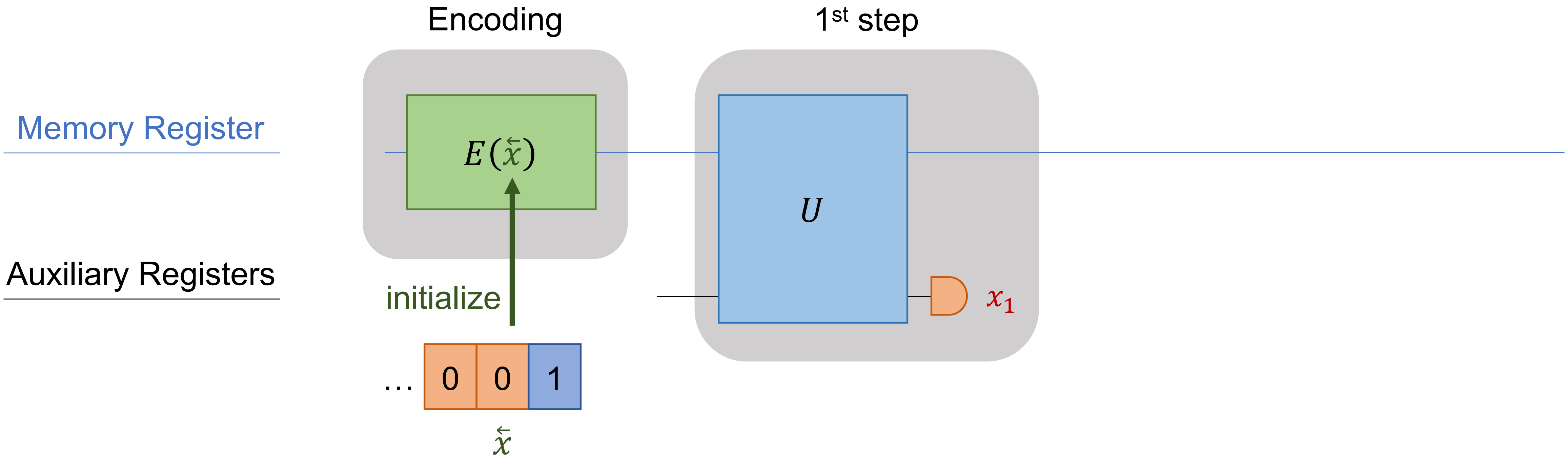


Quantum Circuits



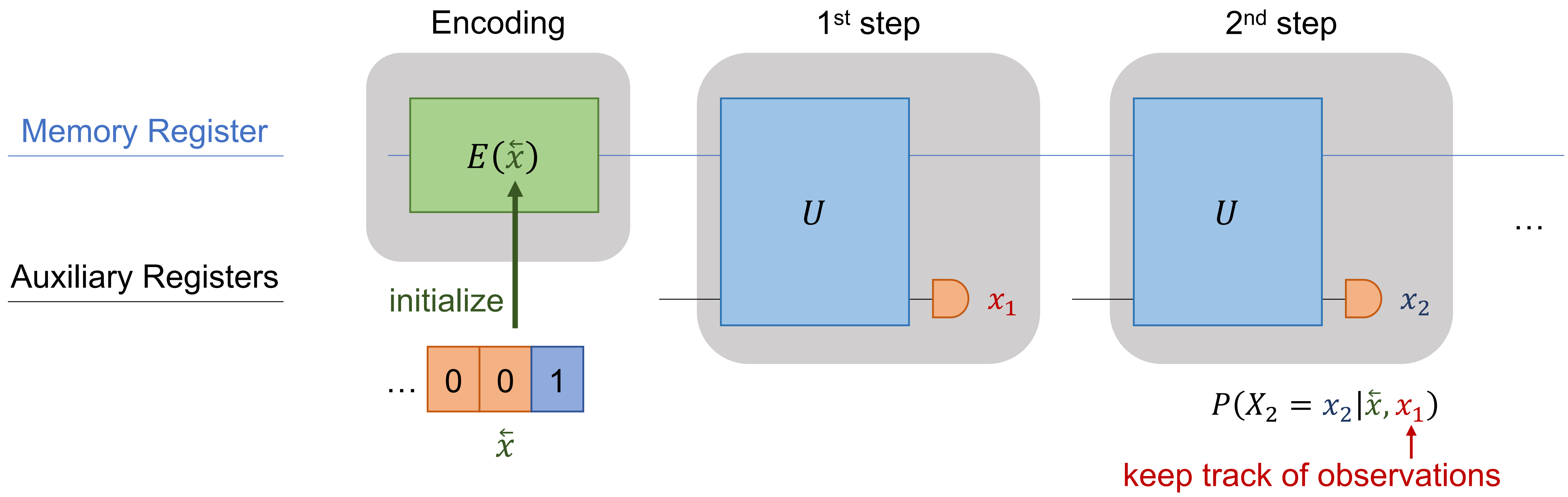
q–simulator

q–simulator

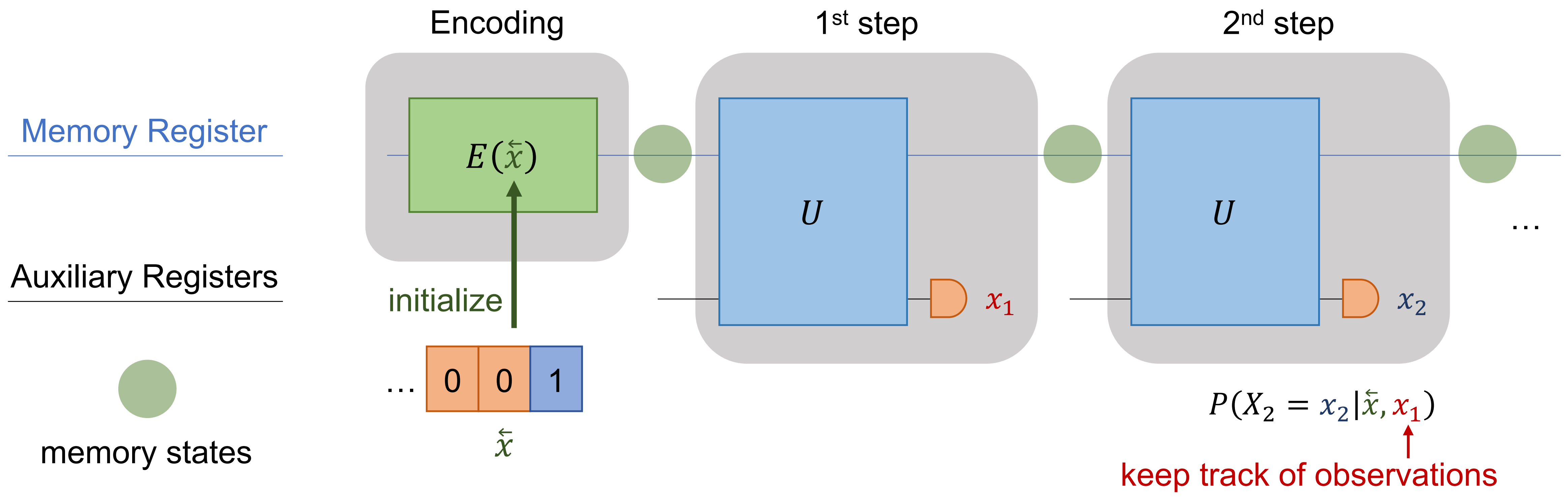


Encoding
1st step
Memory Register
Auxiliary Registers
$E(\overleftarrow{x})$
U
initialize
… 0 0 1
$\overleftarrow{x}$
x_1

q–simulator

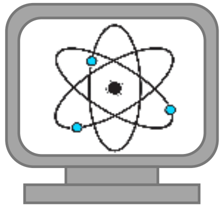

Encoding
1st step
2nd step
Memory Register
Auxiliary Registers
$E(\overleftarrow{x})$
initialize
… 0 0 1
$\overleftarrow{x}$
U
U
x_1
x_2
…
$P(X_2 = x_2 | \overleftarrow{x}, x_1)$
keep track of observations

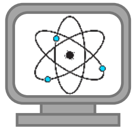
q–simulator



Quantum Topological Complexity: $d_q = \#\text{qubits}$

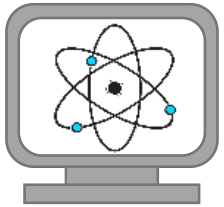
Advantage

In general:  $d_q \leq d_c$ 

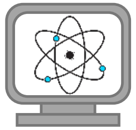
For some processes:  $d_q < d_c$ 

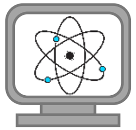
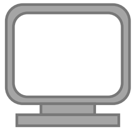
Thompson et al.,
10.1103/PhysRevX.8.031013

Advantage

In general:  $d_q \leq d_c$ 

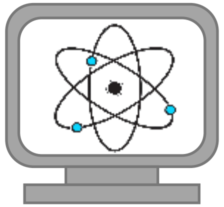
Thompson et al., 10.1103/PhysRevX.8.031013

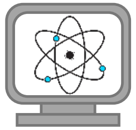
For some processes:  $d_q < d_c$ 

Approximate models:  $\hat{d}_q = \hat{d}_c$ 

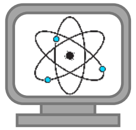
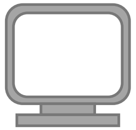
Q-Models can have better accuracy
Yang et al., arXiv:2105.14434

Advantage

In general:  $d_q \leq d_c$ 

For some processes:  $d_q < d_c$ 

Thompson et al., 10.1103/PhysRevX.8.031013

Approximate models:  $\hat{d}_q = \hat{d}_c$ 

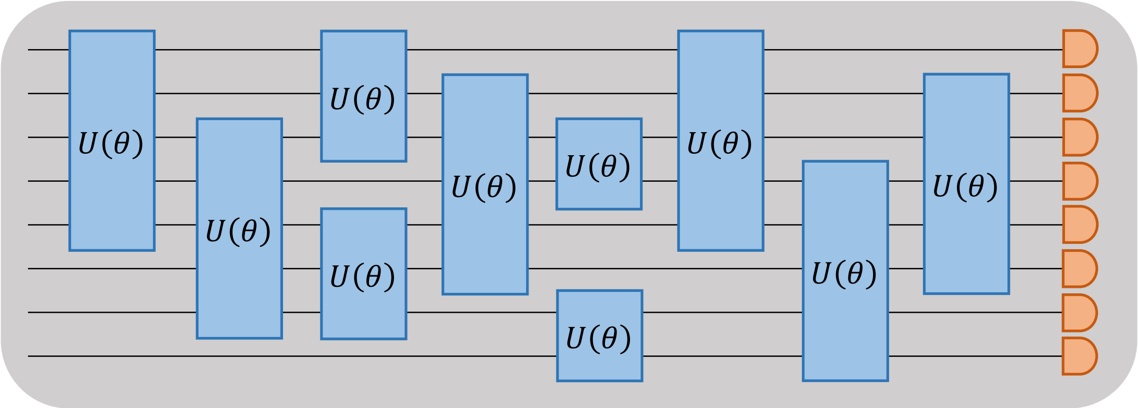
Q-Models can have better accuracy
Yang et al., arXiv:2105.14434

How to get a **quantum representation** of a quantum model?

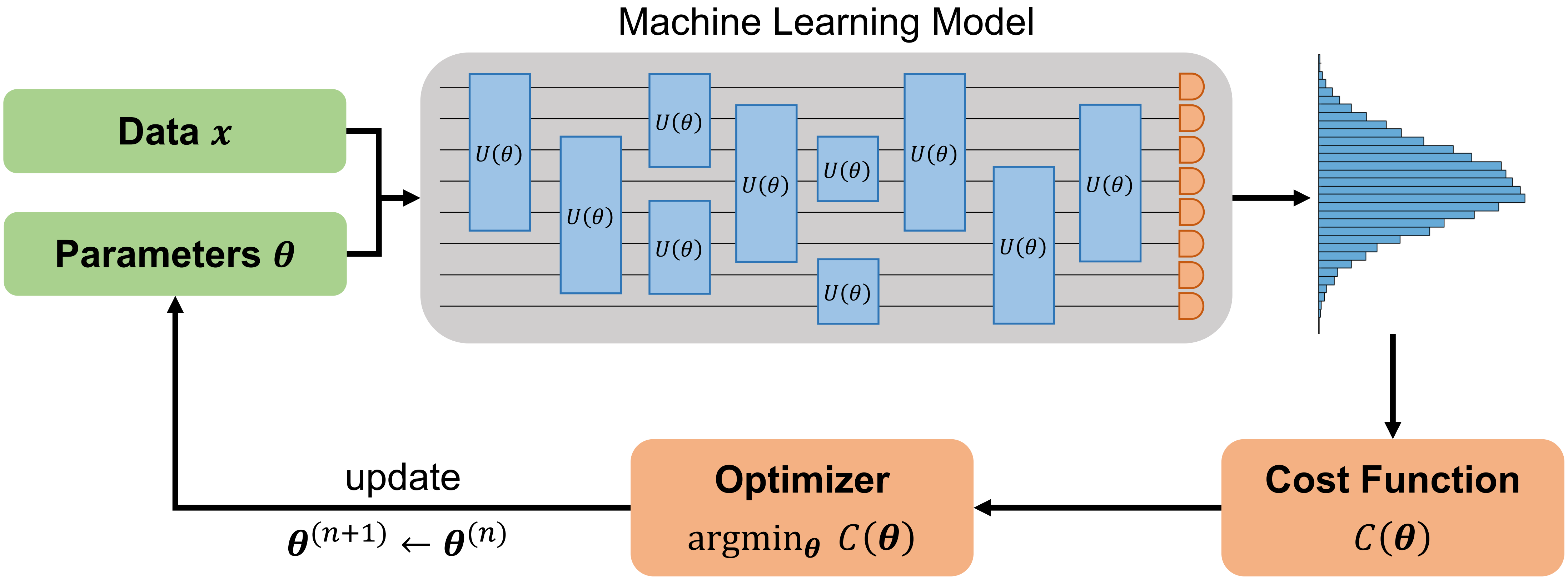
Variational Quantum Circuits

Variational Quantum Circuits

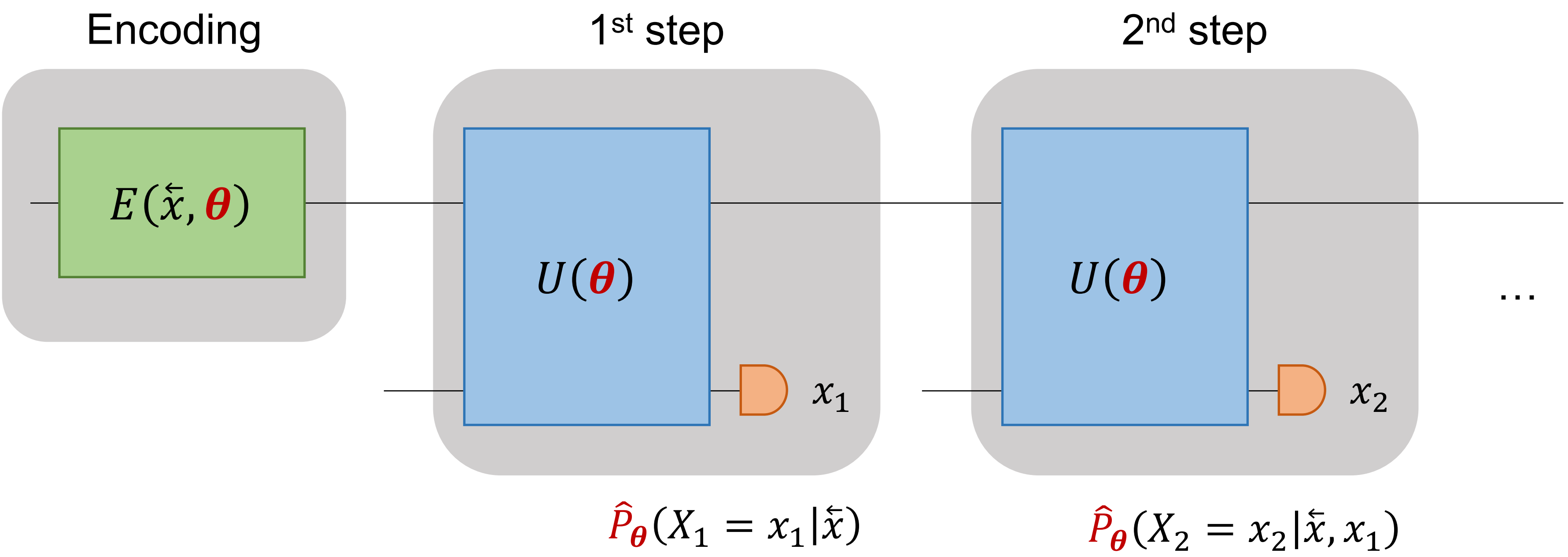
Machine Learning Model



Variational Quantum Algorithms



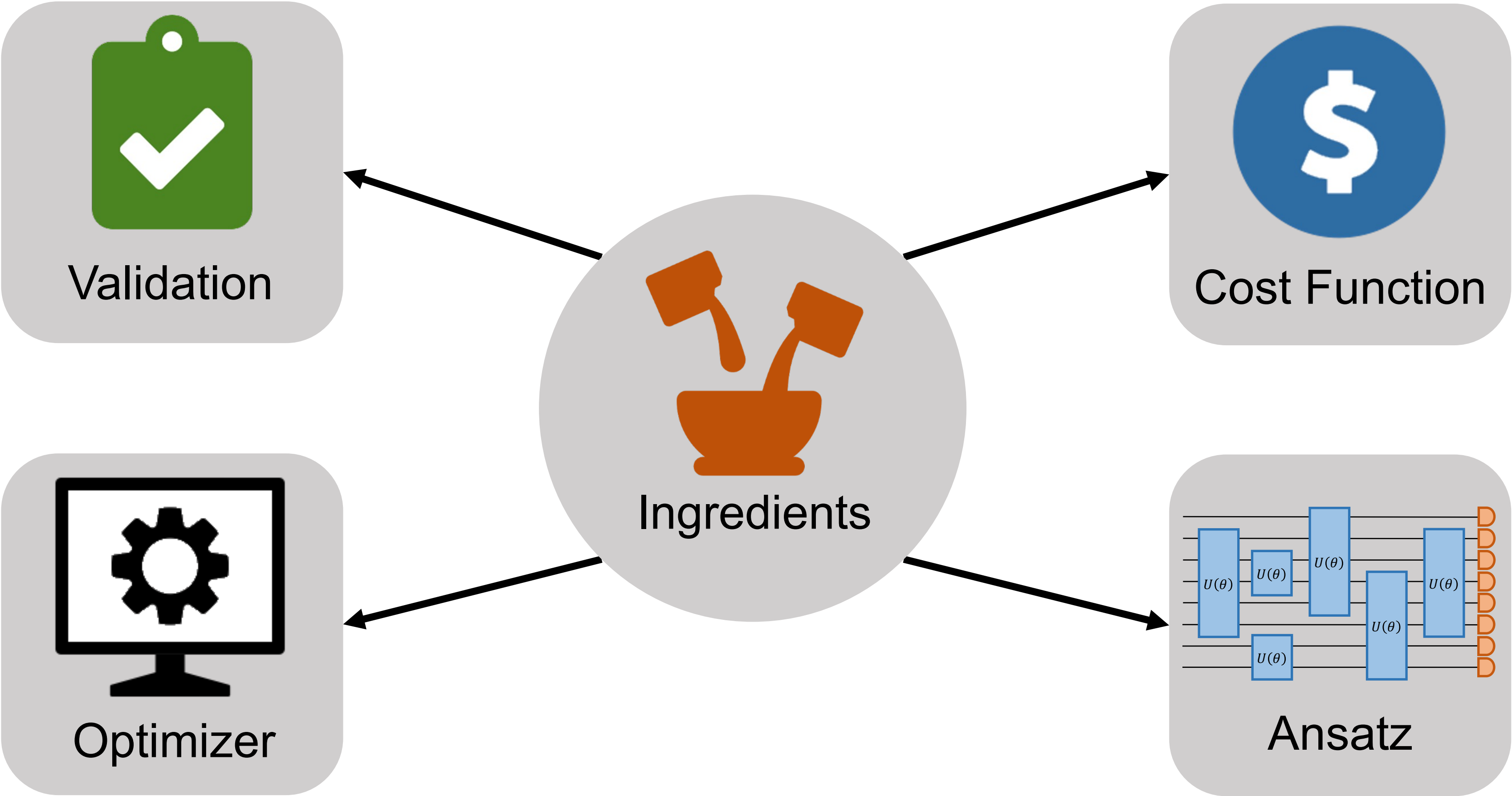
Idea



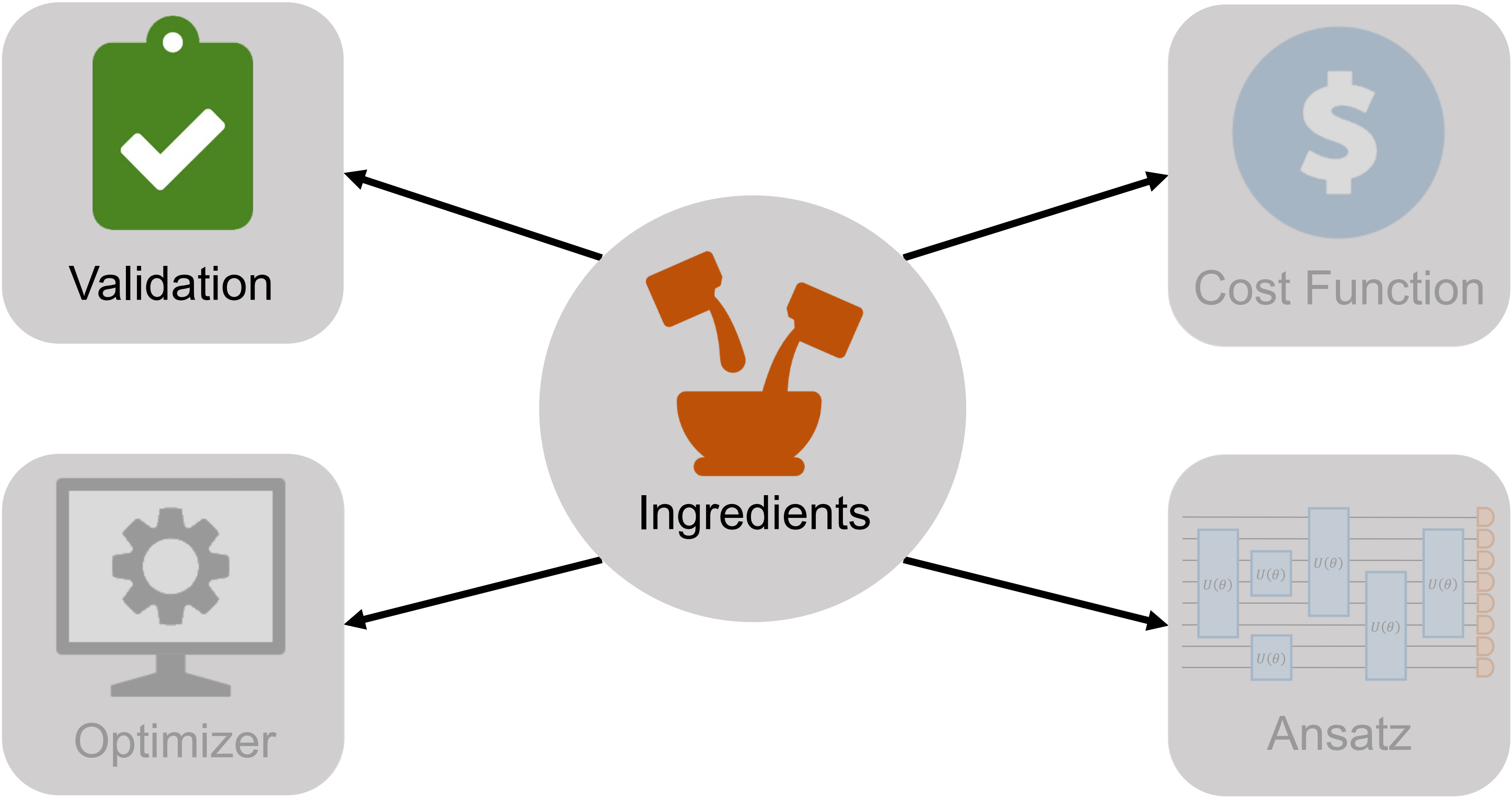
Approximate P → $|P - \hat{P}_{\boldsymbol{\theta}}| < \delta$

Quantum
Learning
Algorithm

Quantum Learning Algorithm

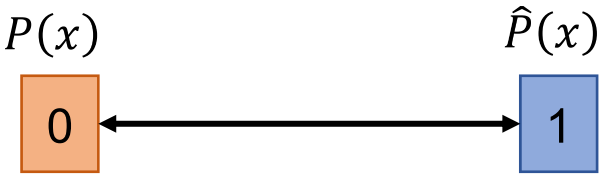


Quantum Learning Algorithm



Kullback-Leibler divergence:
(KL)

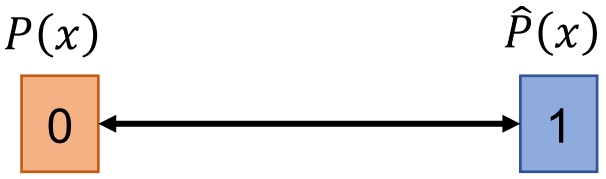
$$D_{KL}(P, \hat{P}) = \sum_x P(x) \log_2 \frac{P(x)}{\hat{P}(x)}$$

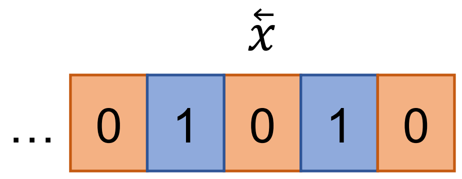


Validation

Kullback-Leibler divergence:
(KL)

$$D_{KL}(P, \hat{P}) = \sum_x P(x) \log_2 \frac{P(x)}{\hat{P}(x)}$$

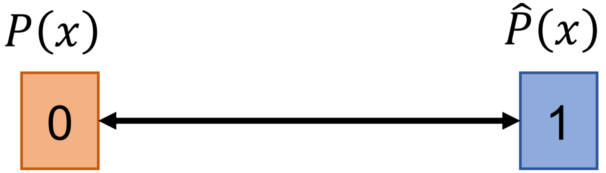


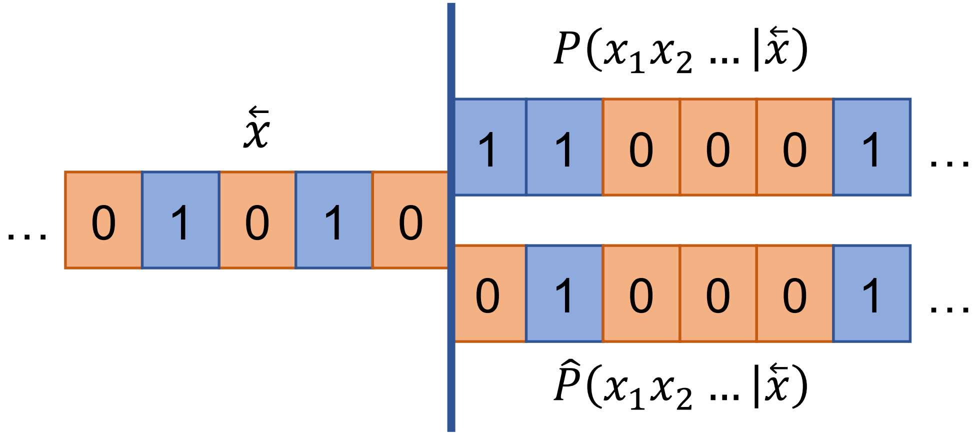


Validation

Kullback-Leibler divergence:
(KL)

$$D_{KL}(P,\hat{P}) = \sum_x P(x)\log_2\frac{P(x)}{\hat{P}(x)}$$

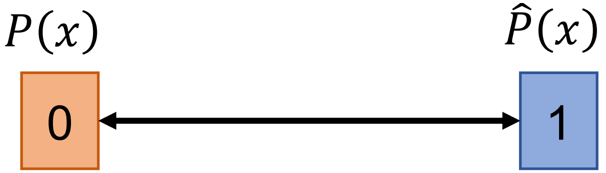




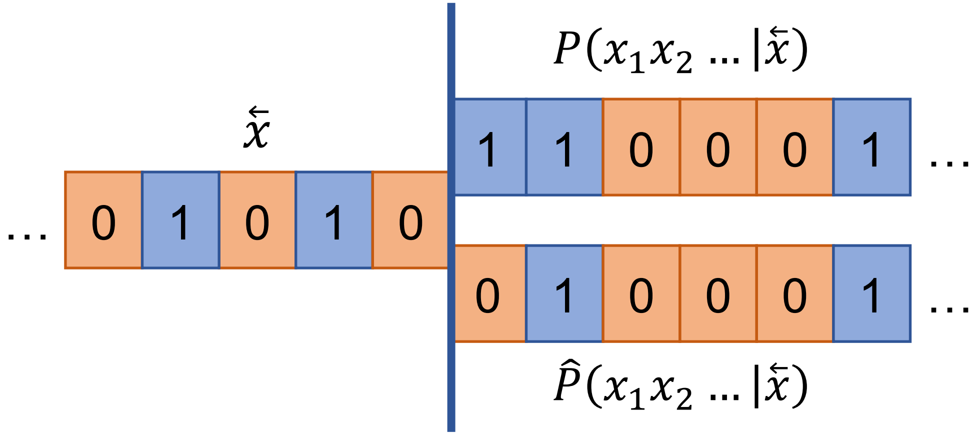
Validation

Kullback-Leibler divergence:
(KL)

$$D_{KL}\left(P,\hat{P}\right) = \sum_{x} P(x)\log_2\frac{P(x)}{\hat{P}(x)}$$



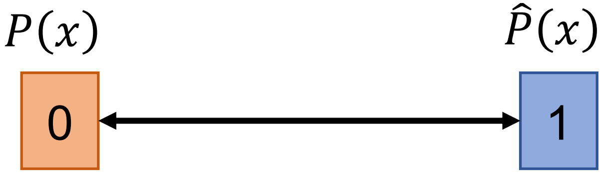
$$D_{KL}\left(L,P,\hat{P}\right) = \sum_{x_{1:L}}\frac{1}{L}\sum_{\overleftarrow{x}} P(\overleftarrow{x})\cdot P(x_{1:L}|\overleftarrow{x})\log_2\frac{P(x_{1:L}|\overleftarrow{x})}{\hat{P}(x_{1:L}|\overleftarrow{x})}$$



Validation

Kullback-Leibler divergence:
(KL)

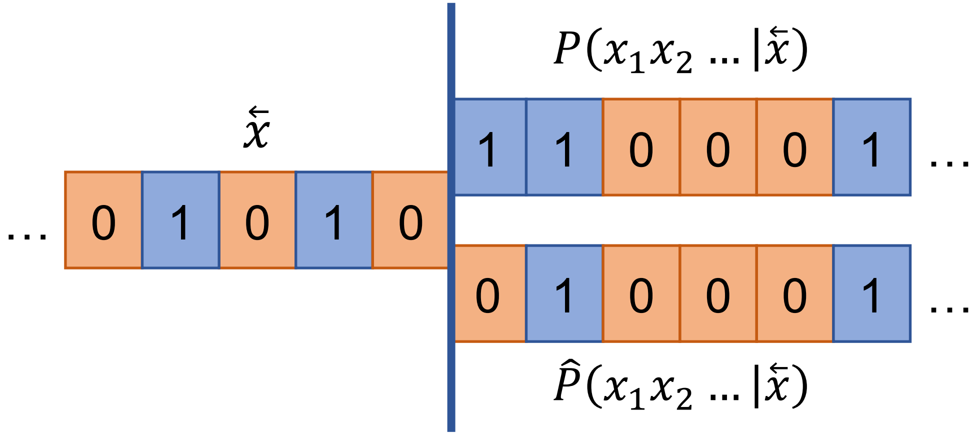
$$D_{KL}(P,\hat{P}) = \sum_x P(x)\log_2\frac{P(x)}{\hat{P}(x)}$$



$D_{KL}(L,P,\hat{P}) =$

average over pasts

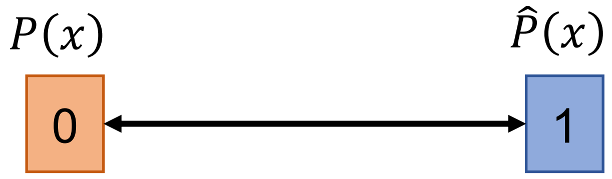
mean over time steps



Validation

Total Variation Distance:
(TV)

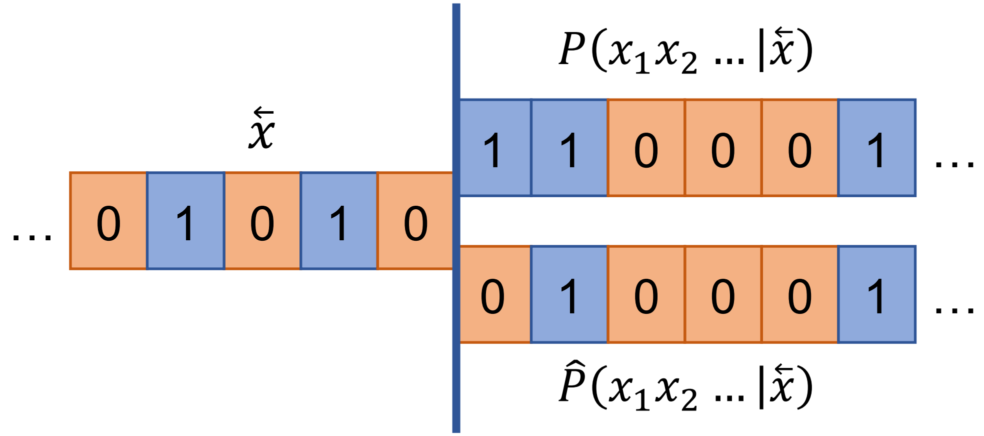
$$D_{TV}(P,\hat{P}) = \frac{1}{2}\sum_{x}\left|P(x) - \hat{P}(x)\right|$$



Validation

Total Variation Distance:
(TV)

$$D_{TV}(P,\hat{P}) = \frac{1}{2}\sum_x \left|P(x) - \hat{P}(x)\right|$$

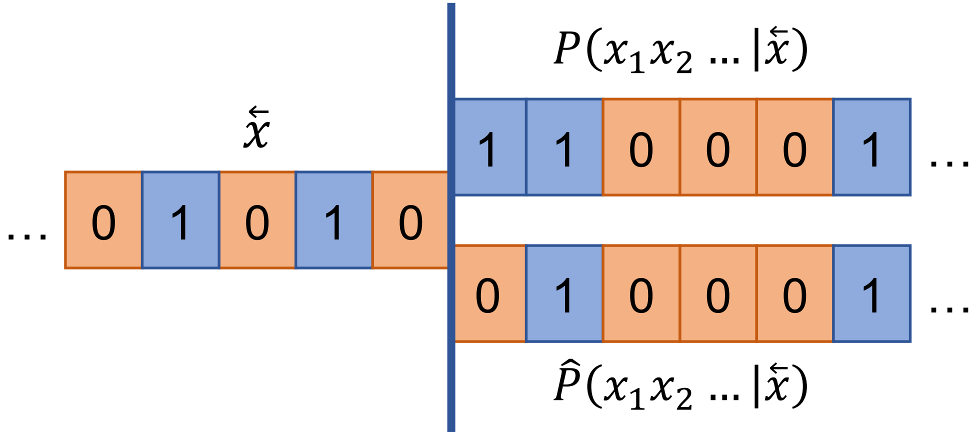


Validation

Total Variation Distance:
(TV)

$$D_{TV}\left(P, \hat{P}\right) = \frac{1}{2}\sum_{x}\left|P(x) - \hat{P}(x)\right|$$

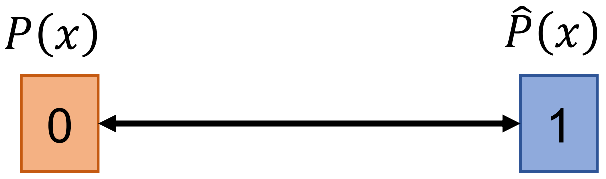
$$D_{TV}\left(L, P, \hat{P}\right) = \frac{1}{2}\sum_{x_{1:L}}\sum_{\overleftarrow{x}}\left|P(x_{1:L}|\overleftarrow{x}) - \hat{P}(x_{1:L}|\overleftarrow{x})\right|$$



Validation

Total Variation Distance: (TV)

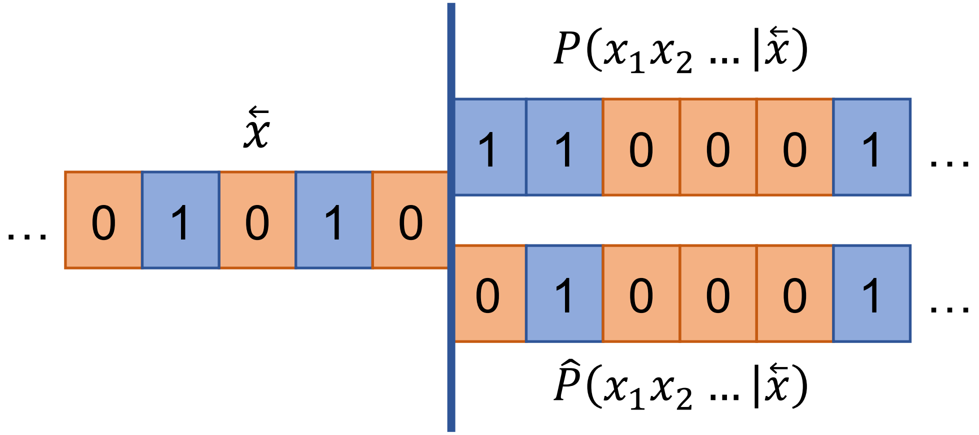
$$D_{TV}(P, \hat{P}) = \frac{1}{2}\sum_x \left|P(x) - \hat{P}(x)\right|$$



$$D_{TV}(L, P, \hat{P}) =$$

sum up pasts

sum up time steps



Validation

Absolute Measure:
TV Distance

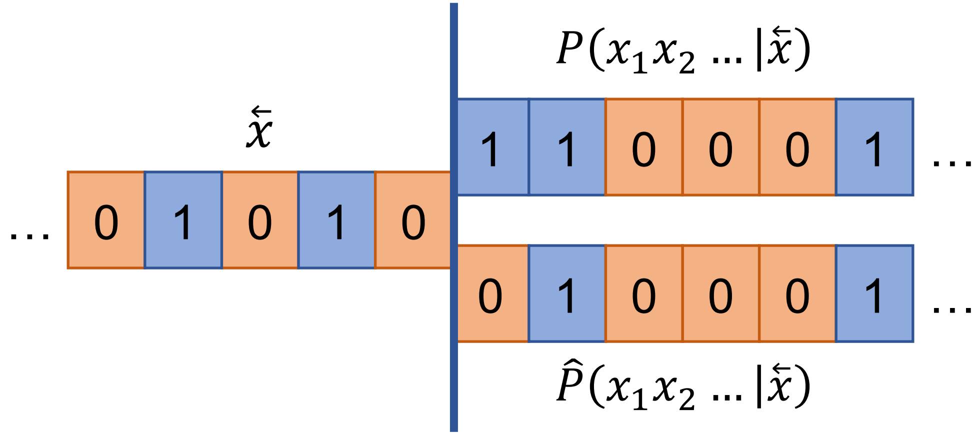
sum up pasts

sum up time steps

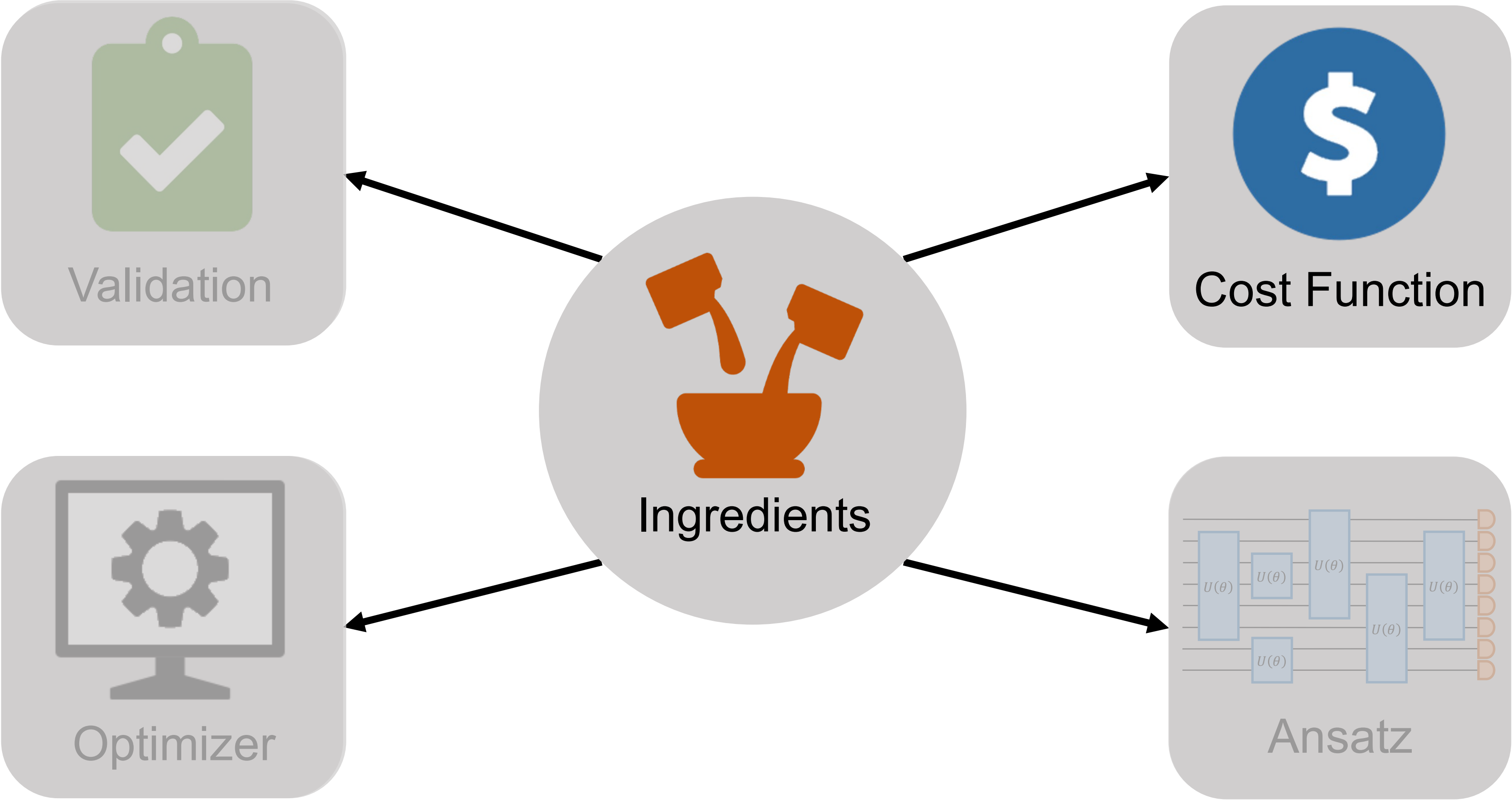
Relative Measure:
KL Divergence

average over pasts

mean over time steps



Quantum Learning Algorithm



Cost Function

Ideally, use validation metric: $$D_{KL}(P, \hat{P}) = \sum_x P(x) \log_2 \frac{P(x)}{\hat{P}(x)}$$

Cost Function

Ideally, use validation metric: $D_{KL}(P, \hat{P}) = \sum_x P(x) \log_2 \frac{P(x)}{\hat{P}(x)}$

unknown

inefficient

Cost Function

Ideally, use validation metric: $D_{KL}(P, \hat{P}) = \sum_x P(x) \log_2 \frac{P(x)}{\hat{P}(x)}$

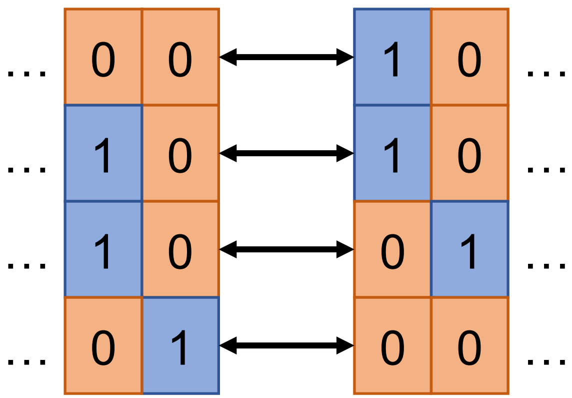
unknown

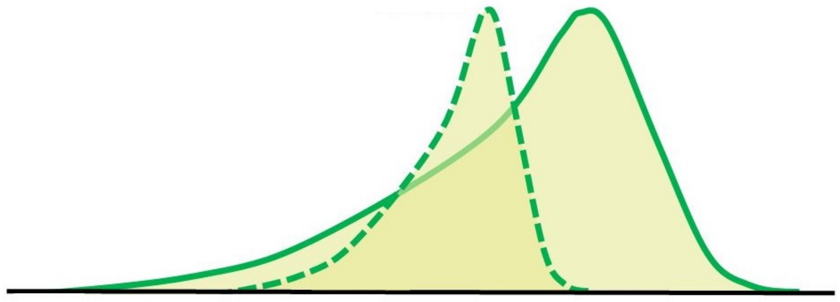
inefficient

We need an alternative!

Cost Function

Maximum Mean Discrepancy:
(MMD)





$$\mathrm{MMD}\left[P,\hat{P}\right]=0 \quad \Leftrightarrow \quad P=\hat{P}$$

Maximum Mean Discrepancy

Problem 1

Let X and Y be random variables with probability distributions P and $\hat{P}$.
Moreover, let $x = x_1, x_2, \dots, x_m$ and $y = y_1, y_2, \dots, y_n$ be i.i.d. observations.

Can we decide whether $P \neq \hat{P}$?

from Gretton et al., “A Kernel Two-Sample Test”, Journal of ML Research, 2012

Maximum Mean Discrepancy

Lemma 1

Let $(\mathcal{X}, d)$ be a metric space. Then, $P = \hat{P}$ if and only if

$$\mathbb{E}_{x\sim P}[f(x)] = \mathbb{E}_{y\sim \hat{P}}[f(y)] \quad \text{for all} \quad f \in C(\mathcal{X}),$$

where $C(\mathcal{X})$ are all bounded continuous functions on $\mathcal{X}$.

Maximum Mean Discrepancy

Lemma 1

Let $(\mathcal{X}, d)$ be a metric space. Then, $P = \hat{P}$ if and only if

$$\mathbb{E}_{x\sim P}[f(x)] = \mathbb{E}_{y\sim \hat{P}}[f(y)] \quad \text{for all} \quad f \in C(\mathcal{X}),$$

where $C(\mathcal{X})$ are all bounded continuous functions on $\mathcal{X}$.

Problem: $C(\mathcal{X})$ has infinite dimension → cannot be used practically.

Maximum Mean Discrepancy

Definition 1

Let $F(\mathcal{X})$ be a class of functions $f\colon \mathcal{X} \to \mathbb{R}$. The MMD is defined as

$$MMD\left(F, P, \hat{P}\right) = \sup_{f \in F} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim \hat{P}}[f(y)]\right).$$

Maximum Mean Discrepancy

Definition 1

Let $F(\mathcal{X})$ be a class of functions $f\colon \mathcal{X} \to \mathbb{R}$. The MMD is defined as

$$MMD(F, P, \hat{P}) = \sup_{f \in F} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim \hat{P}}[f(y)] \right).$$

Practically, we can estimate the MMD via

$$MMD(F, P, \hat{P}) = \sup_{f \in F} \left(\frac{1}{m} \sum_{i=1}^{m} f(x_i) - \frac{1}{n} \sum_{j=1}^{n} f(y_j) \right).$$

Maximum Mean Discrepancy

Definition 1

Let $F(\mathcal{X})$ be a class of functions $f: \mathcal{X} \to \mathbb{R}$. The MMD is defined as

$$MMD\left(F, P, \hat{P}\right) = \sup_{f \in F} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim \hat{P}}[f(y)]\right).$$

If we choose $F(\mathcal{X}) = C(\mathcal{X})$, we get Lemma 1, i.e., $MMD\left(F, P, \hat{P}\right) = 0$ iff $P = \hat{P}$.

Maximum Mean Discrepancy

Definition 1

Let $F(\mathcal{X})$ be a class of functions $f: \mathcal{X} \to \mathbb{R}$. The MMD is defined as

$$MMD(F, P, \hat{P}) = \sup_{f \in F} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim \hat{P}}[f(y)] \right).$$

If we choose $F(\mathcal{X}) = C(\mathcal{X})$, we get Lemma 1, i.e., $MMD(F, P, \hat{P}) = 0$ iff $P = \hat{P}$.

However, we can get the same with a *universal reproducing kernel Hilbert space (RKHS).*

Universal RKHS

Definition 2 (simplified)

A Hilbert space $F(\mathcal{X})$ of functions $f: \mathcal{X} \to \mathbb{R}$ is called a RKHS, if a function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ exists, such that

$k(\cdot,\cdot)$ is symmetric and positive semi-definite and

$k(x, y) = \langle k(\cdot, y), k(x,\cdot)\rangle_F$ for all $x, y \in \mathcal{X}$.

We call $k(\cdot,\cdot)$ a *reproducing kernel function.*

Universal RKHS

Definition 3

A RKHS $F(\mathcal{X})$ is called universal, if $F(\mathcal{X}) \subset C(\mathcal{X})$ is dense.

That means, any function $f \in C(\mathcal{X})$ can be approximated by functions in $F(\mathcal{X})$.

E.g. Gaussian kernel functions are universal:

$$k(x,y) = \frac{1}{c}\sum_{i=1}^{c} \exp\left(-\frac{|x-y|^2}{2\sigma_i}\right)$$

Maximum Mean Discrepancy

Theorem 1

Let $F(\mathcal{X})$ be the unit ball of a universal RKHS. Then

$$MMD\left(F, P, \hat{P}\right) = 0 \quad \text{iff} \quad P = \hat{P}.$$

Maximum Mean Discrepancy

Theorem 2

Let X, X' be random variables with distribution P and Y, Y' be random variables with distribution $\hat{P}$, respectively. Then

$$MMD^2(F, P, \hat{P}) = \underset{x \sim P, x' \sim P}{\mathbb{E}}[k(x, x')] - 2 \underset{x \sim P, y' \sim \hat{P}}{\mathbb{E}}[k(x, y')] + \underset{y \sim \hat{P}, y' \sim \hat{P}}{\mathbb{E}}[k(y, y')]$$

Maximum Mean Discrepancy

Theorem 2

Let X, X' be random variables with distribution P and Y, Y' be random variables with distribution $\hat{P}$, respectively. Then

$$MMD^2(F, P, \hat{P}) = \underset{x\sim P, x'\sim P}{\mathbb{E}}[k(x, x')] - 2 \underset{x\sim P, y'\sim \hat{P}}{\mathbb{E}}[k(x, y')] + \underset{y\sim \hat{P}, y'\sim \hat{P}}{\mathbb{E}}[k(y, y')]$$

I.e., we can calculate the MMD based on expectation values.

Maximum Mean Discrepancy

Theorem 2

Let X, X' be random variables with distribution P and Y, Y' be random variables with distribution $\hat{P}$, respectively. Then

$$MMD^2(F, P, \hat{P}) = \underset{x \sim P, x' \sim P}{\mathbb{E}}[k(x, x')] - 2 \underset{x \sim P, y' \sim \hat{P}}{\mathbb{E}}[k(x, y')] + \underset{y \sim \hat{P}, y' \sim \hat{P}}{\mathbb{E}}[k(y, y')]$$

I.e., we can calculate the MMD based on expectation values.

We can estimate the expectation values based only on samples.

Maximum Mean Discrepancy

Theorem 2

Let X, X' be random variables with distribution P and Y, Y' be random variables with distribution $\hat{P}$, respectively. Then

$$MMD^2(F, P, \hat{P}) = \mathop{\mathbb{E}}_{x \sim P, x' \sim P}[k(x, x')] - 2 \mathop{\mathbb{E}}_{x \sim P, y' \sim \hat{P}}[k(x, y')] + \mathop{\mathbb{E}}_{y \sim \hat{P}, y' \sim \hat{P}}[k(y, y')]$$

I.e., we can calculate the MMD based on expectation values.

We can estimate the expectation values based only on samples.

→ We can estimate the “distance” between probability distributions based only on samples.

Maximum Mean Discrepancy

Theorem 2

Let X, X' be random variables with distribution P and Y, Y' be random variables with distribution $\hat{P}$, respectively. Then

$$MMD^2(F, P, \hat{P}) = \mathop{\mathbb{E}}_{x\sim P, x\prime\sim P}[k(x, x')] - 2 \mathop{\mathbb{E}}_{x\sim P, y'\sim \hat{P}}[k(x, y')] + \mathop{\mathbb{E}}_{y\sim \hat{P}, y\prime\sim \hat{P}}[k(y, y')]$$

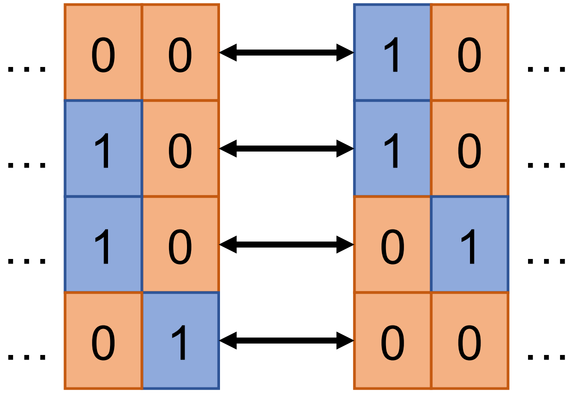
In our work, we use Gaussian kernel functions

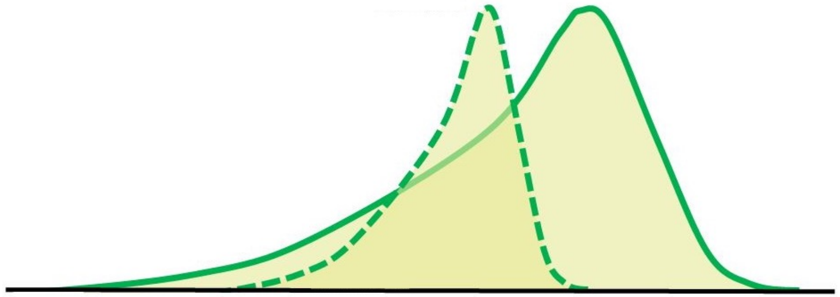
$$k(x, y) = \frac{1}{c}\sum_{i=1}^{c} \exp\left(-\frac{|x-y|^2}{2\sigma_i}\right)$$

with $\boldsymbol{\sigma} = \{0.1, 0.5, 1.0, 5.0\}$.

Cost Function

Maximum Mean Discrepancy:
(MMD)

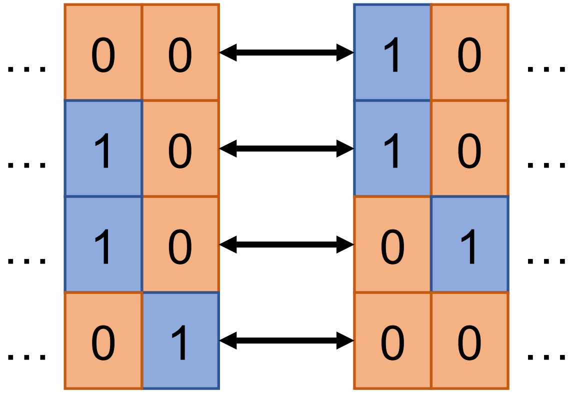


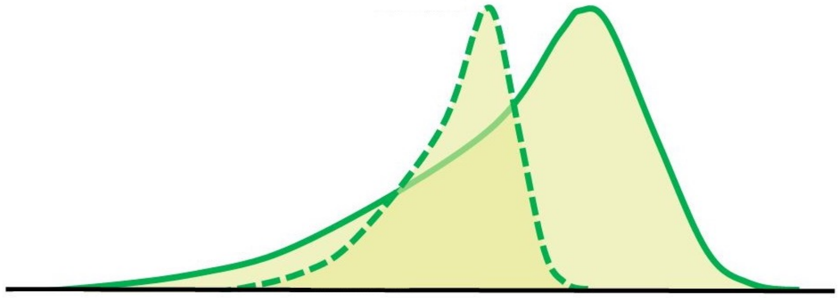


$$\mathrm{MMD}\left[P, \hat{P}\right] = 0 \quad \Leftrightarrow \quad P = \hat{P}$$

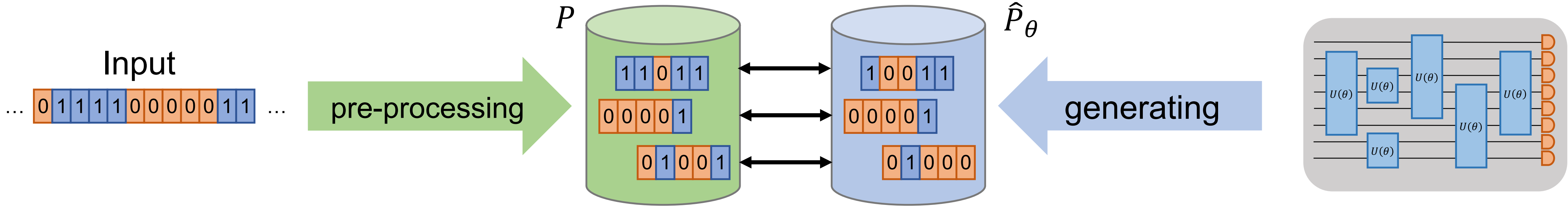
Cost Function

Maximum Mean Discrepancy:
(MMD)



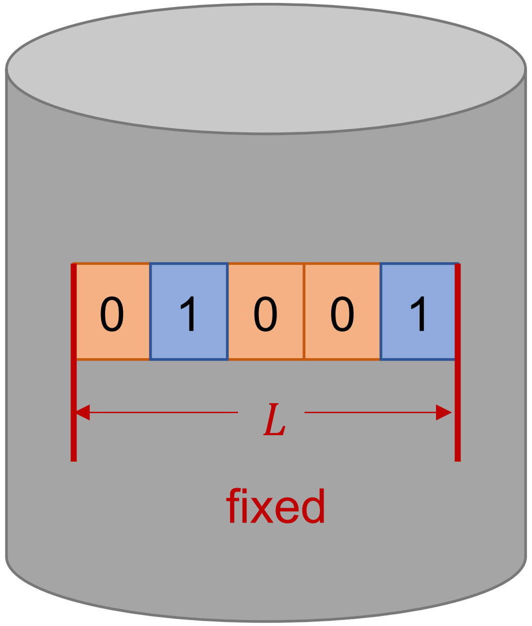


$$\mathrm{MMD}\left[P, \hat{P}\right] = 0 \quad \Leftrightarrow \quad P = \hat{P}$$



Cost Function

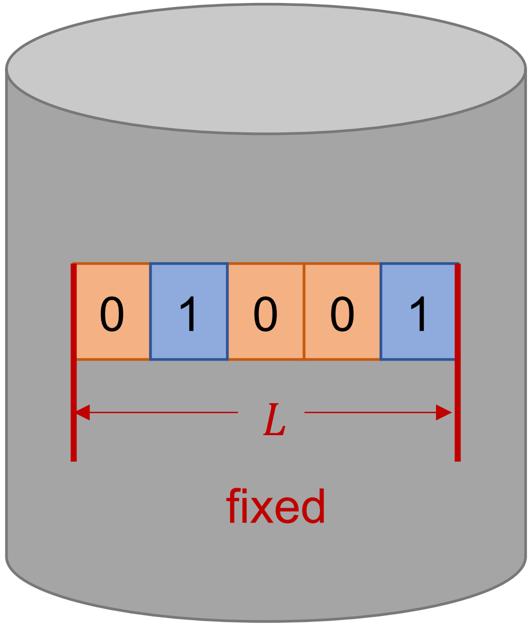
$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}]$$



Cost Function

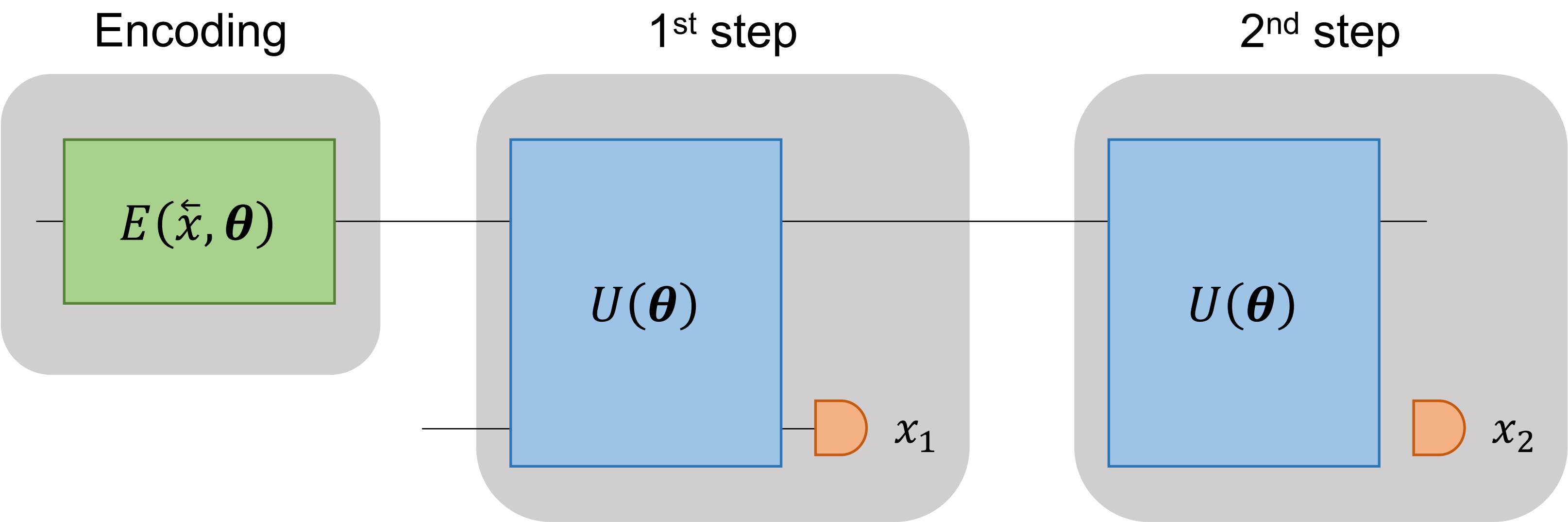
$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}]$$

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

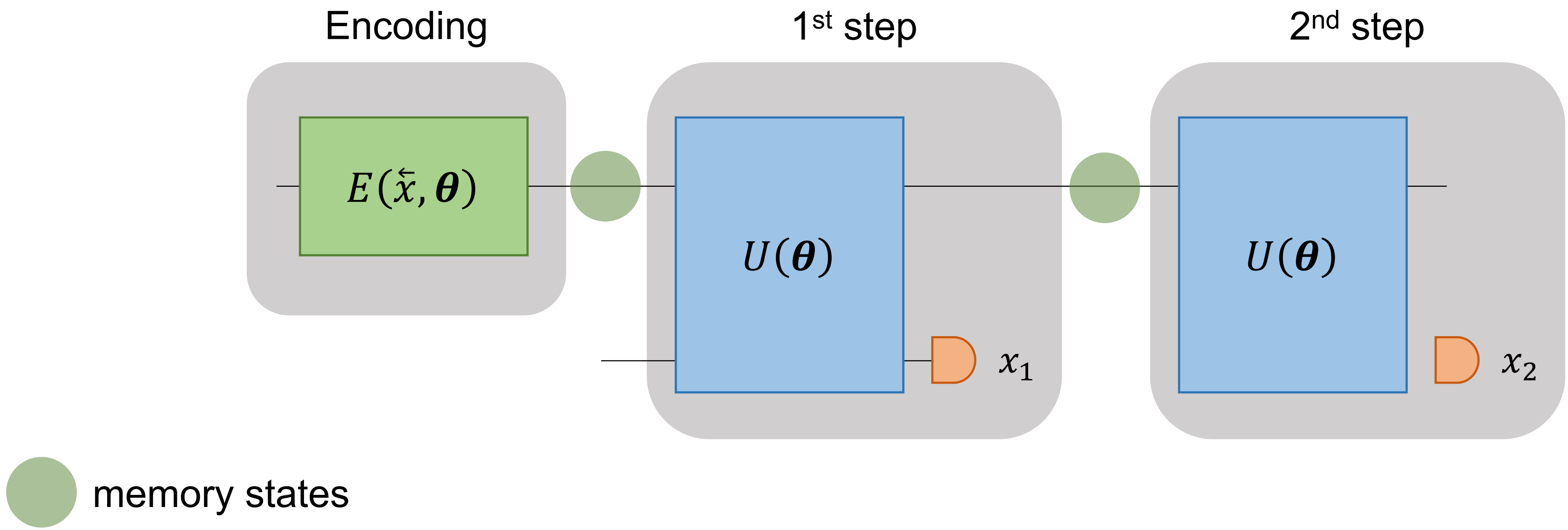


Regularization = penalizes models with a large set of memory states

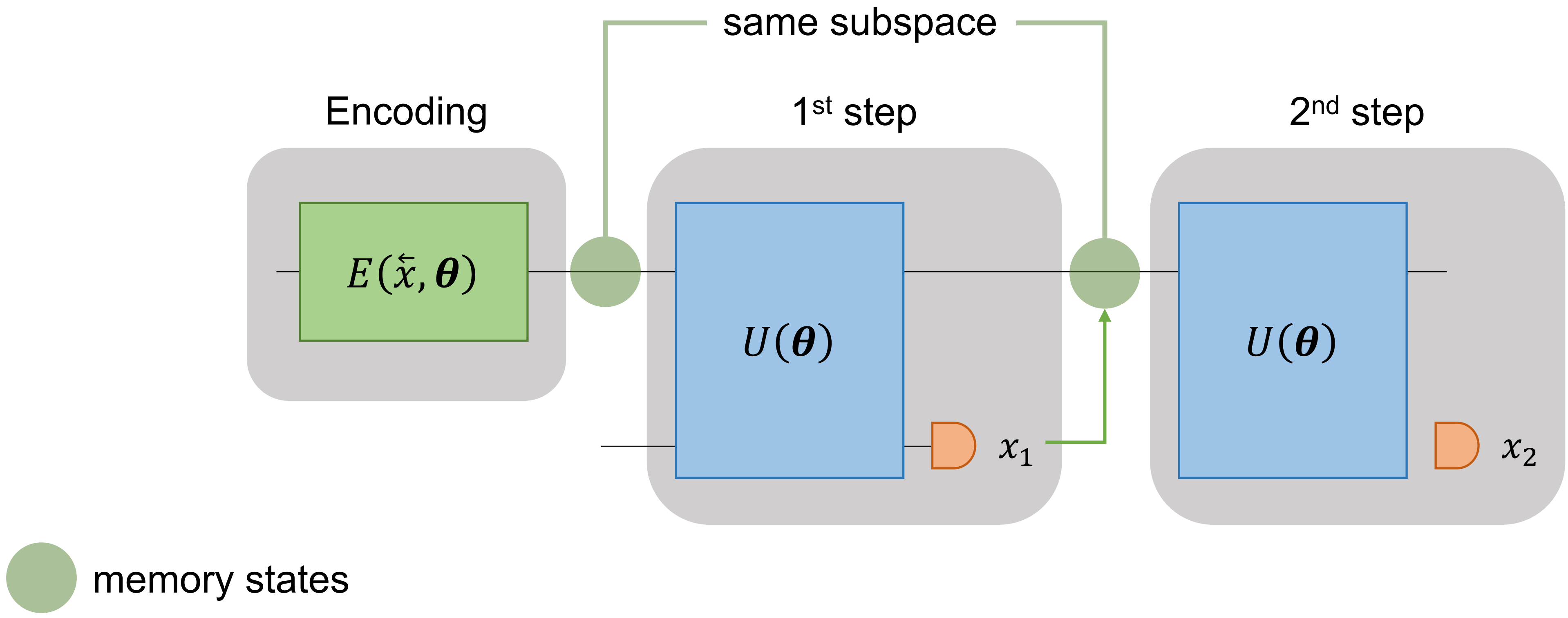
Cost Function



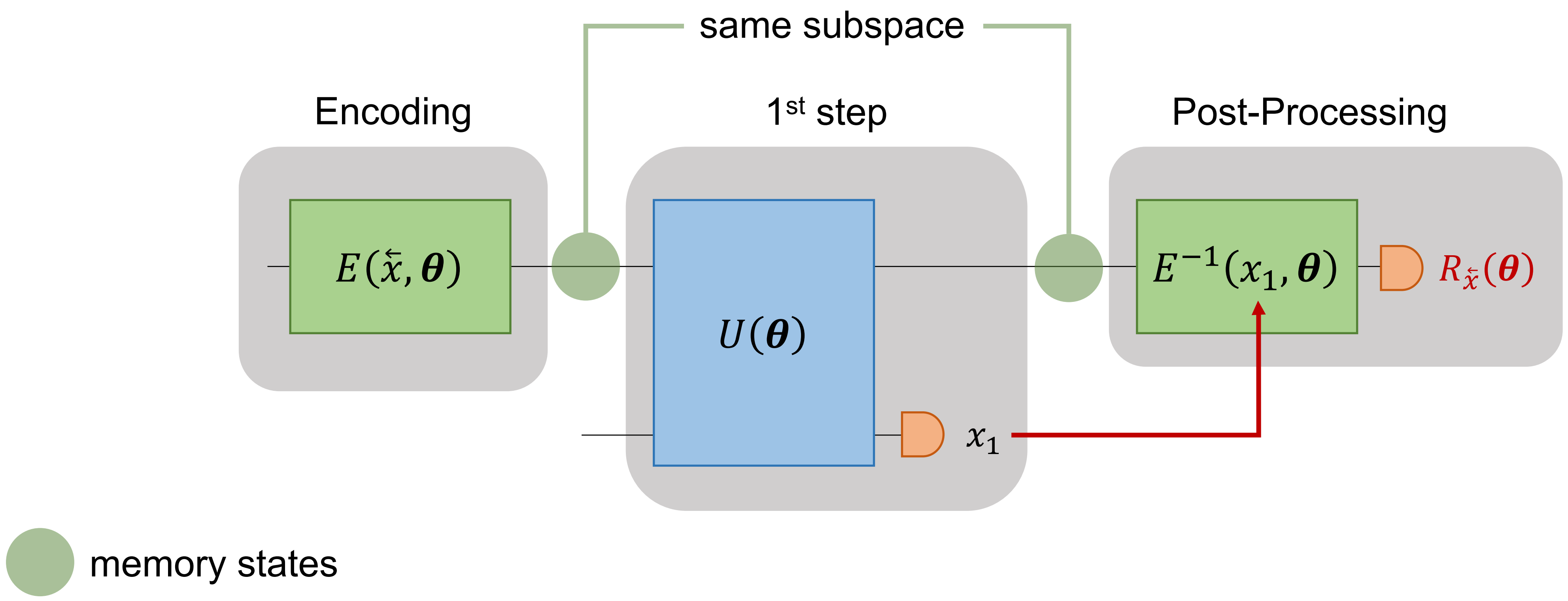
Cost Function



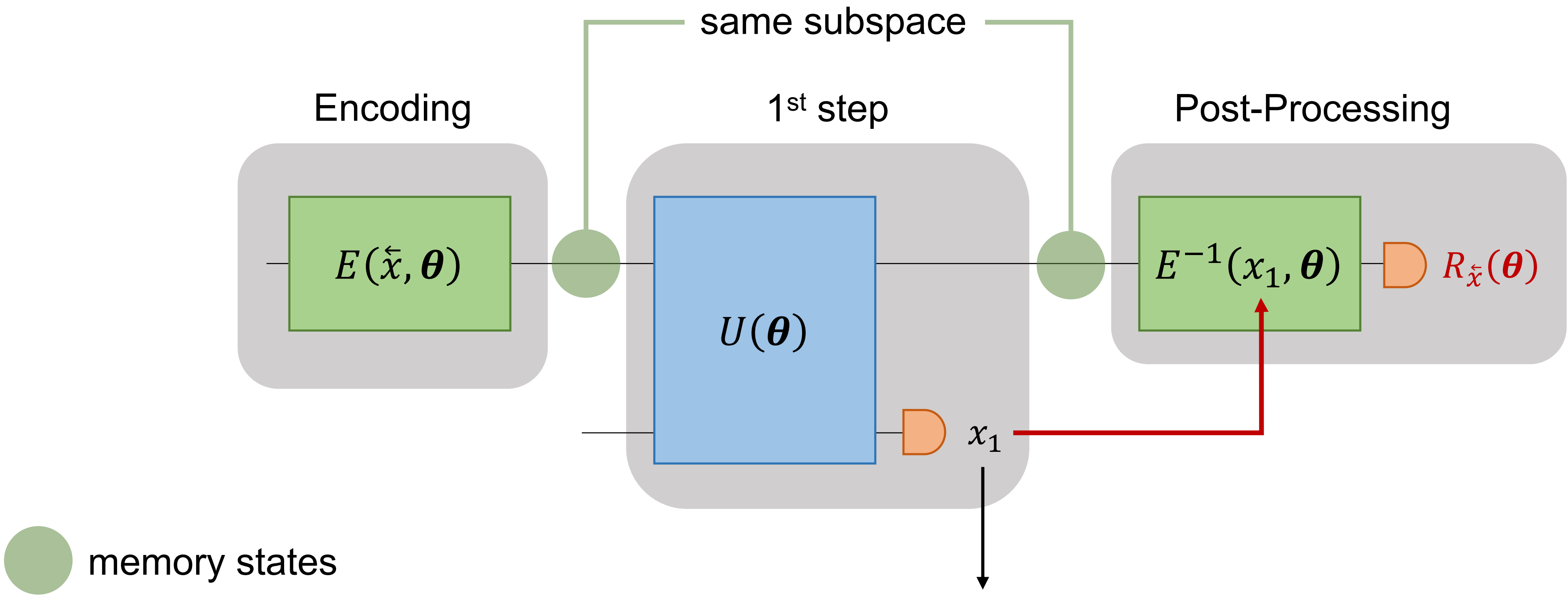
Cost Function



Cost Function

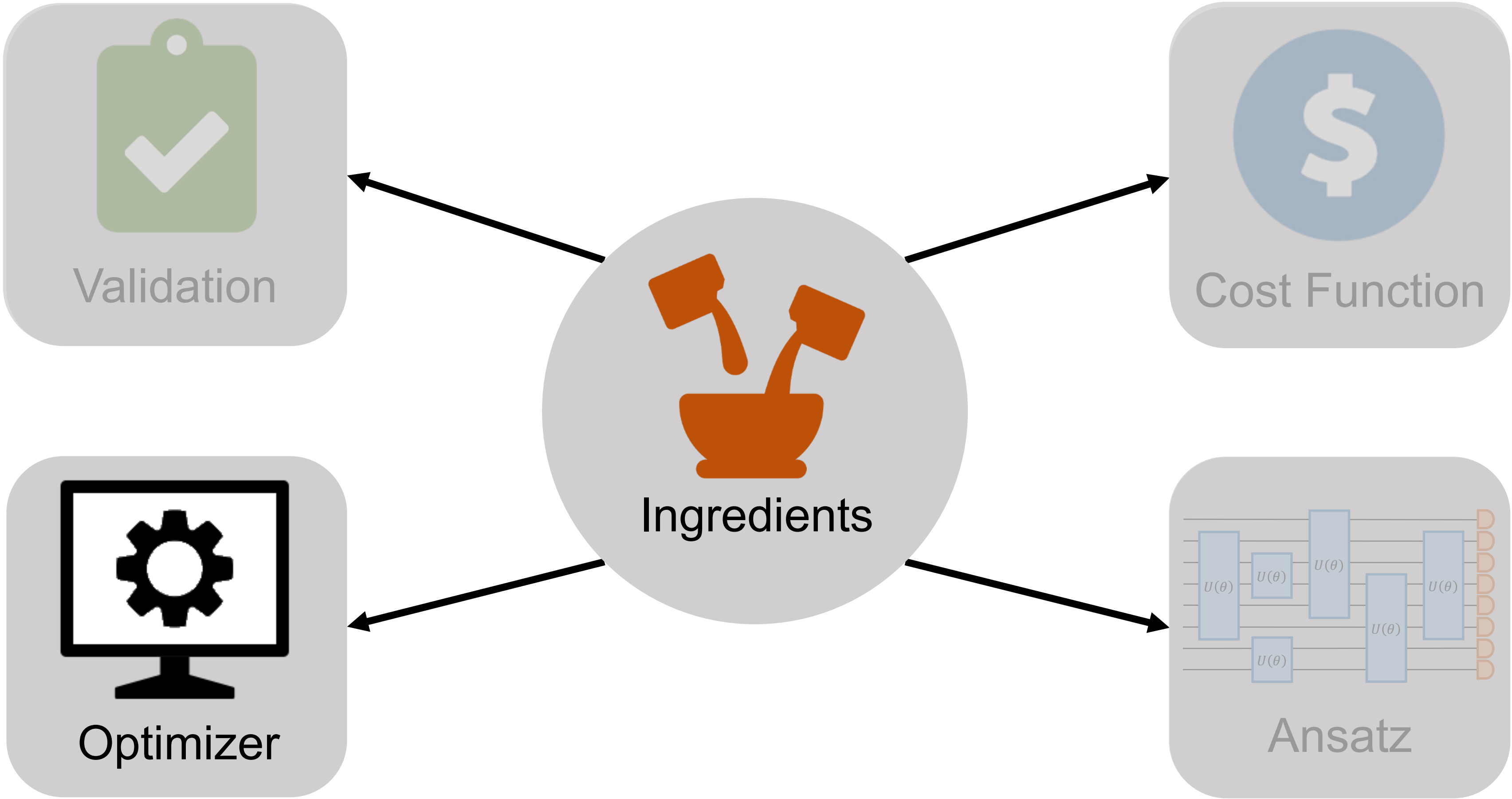


Cost Function



$$\sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}]$$

Quantum Learning Algorithm



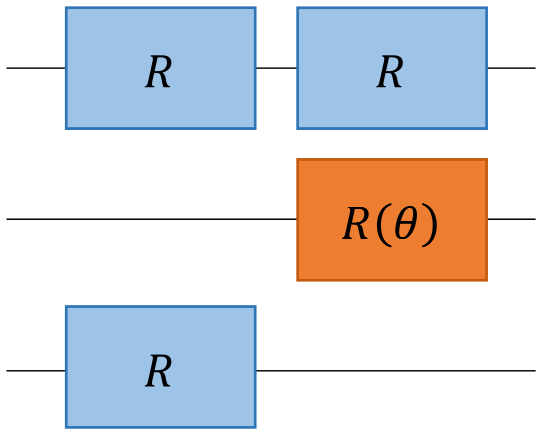
Optimizer

- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

Optimizer

- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

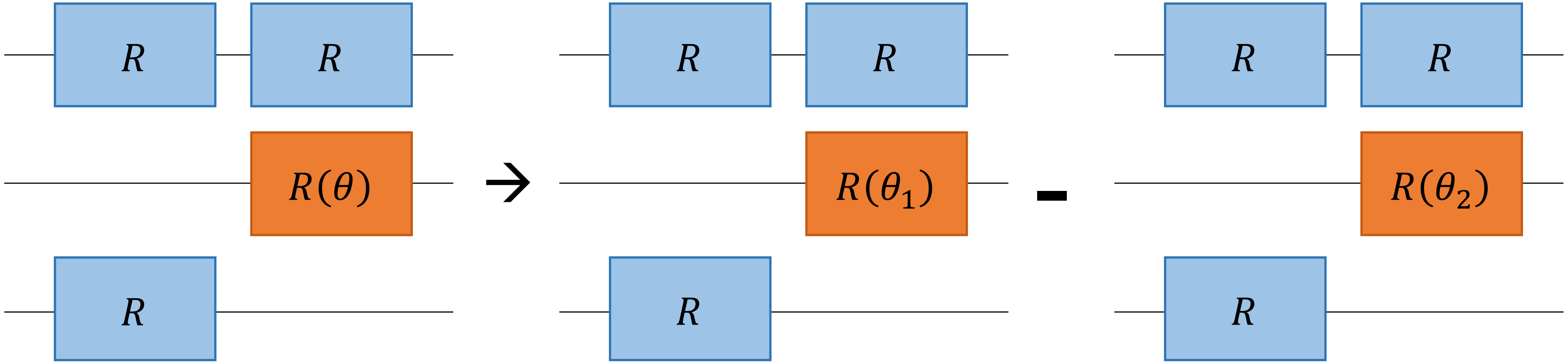
$$f(\theta) = \langle \psi(\theta) | M | \psi(\theta) \rangle$$



Optimizer

- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

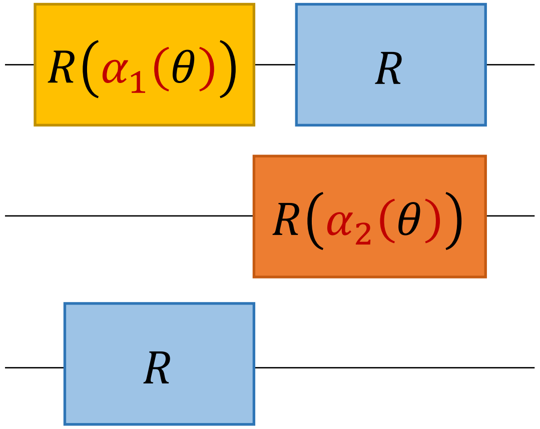
$$f(\theta) = \langle\psi(\theta)|M|\psi(\theta)\rangle \qquad \nabla f(\theta) = f(\theta_1) - f(\theta_2)$$



Optimizer

- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

$$f(\theta) = \langle \psi(\theta) | M | \psi(\theta) \rangle$$

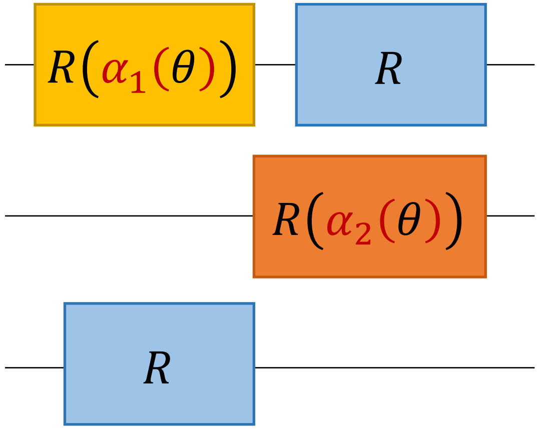


→ $$\nabla f(\theta) = ?$$

Optimizer

- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

$$f(\theta) = \langle \psi(\theta) | M | \psi(\theta) \rangle$$

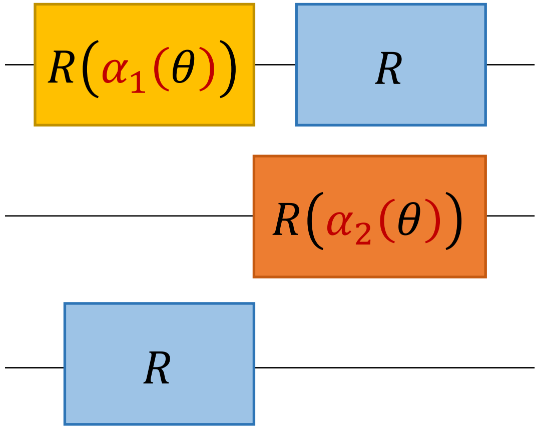


$$\rightarrow \quad \nabla f(\theta) = \sum_{j=1}^{m} \alpha_j'(\mu)\left[f_j\left(\alpha_j(\theta) + \epsilon\right) - f_j\left(\alpha_j(\theta) - \epsilon\right)\right]$$

SimTech

- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

$$f(\theta) = \langle \psi(\theta) | M | \psi(\theta) \rangle$$

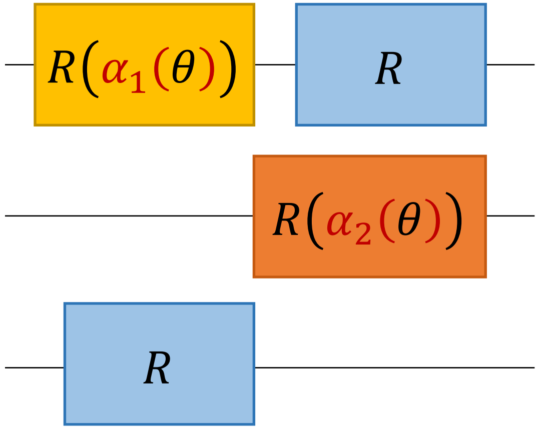


$$\rightarrow \quad \nabla f(\theta) = \sum_{j=1}^{m} \alpha_j'(\mu)\left[f_j\left(\overset{\theta_1}{\alpha_j(\theta) + \epsilon}\right) - f_j\left(\overset{\theta_2}{\alpha_j(\theta) - \epsilon}\right)\right]$$

Optimizer

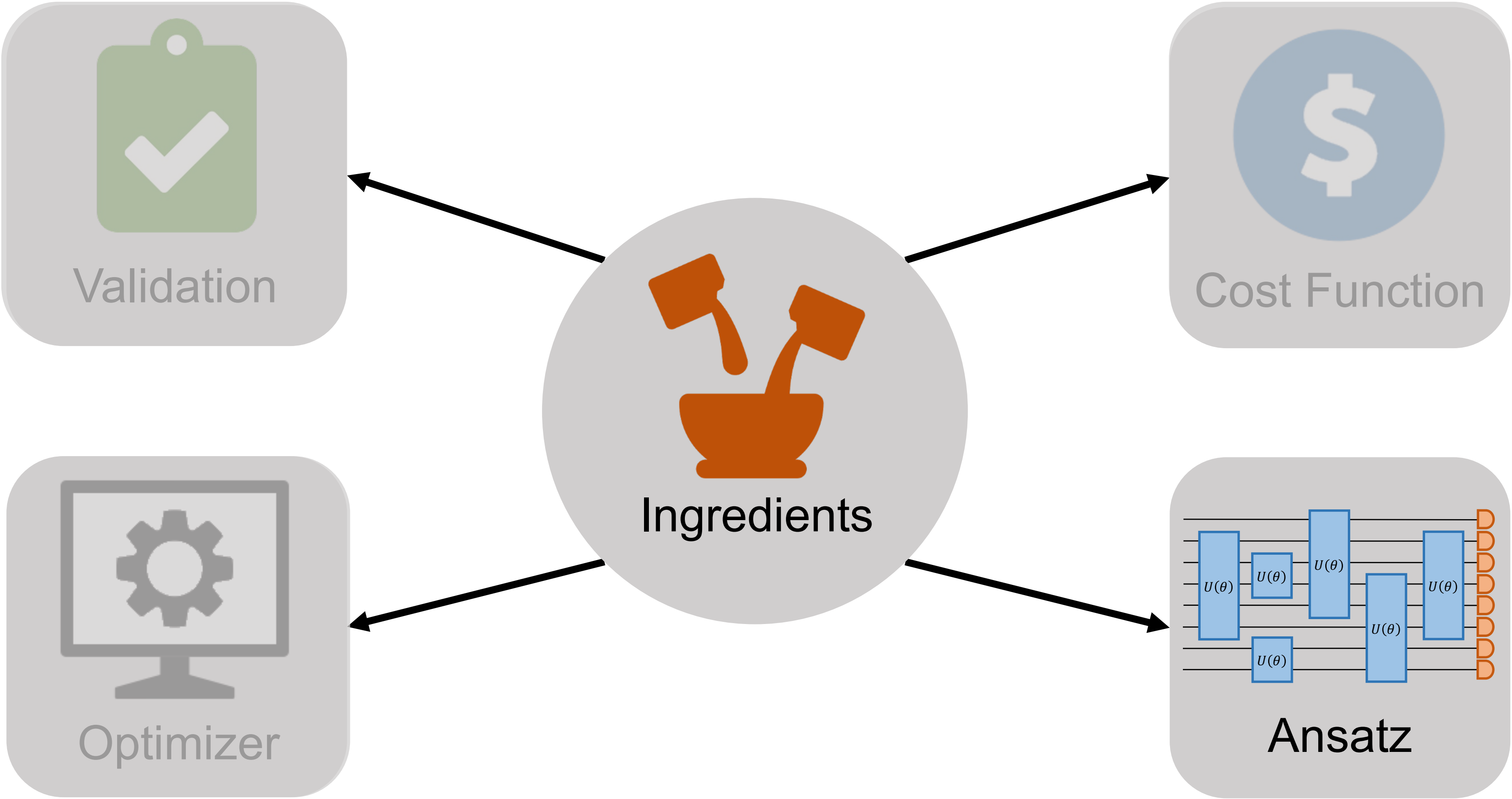
- Use gradient-based optimizer → ADAM (state-of-the-art ML)
- Make use of the parameter-shift rule (state-of-the-art QML)

$$f(\theta) = \langle \psi(\theta) | M | \psi(\theta) \rangle$$



$$\rightarrow \quad \nabla f(\theta) = \sum_{j=1}^{m} \underbrace{\alpha_j'(\mu)}_{\text{ansatz dependent}} \overbrace{\left[\overset{\theta_1}{f_j\big(\alpha_j(\theta) + \epsilon\big)} - \overset{\theta_2}{f_j\big(\alpha_j(\theta) - \epsilon\big)} \right]}^{\text{common parameter-shift rule}}$$

Quantum Learning Algorithm



Ansatz

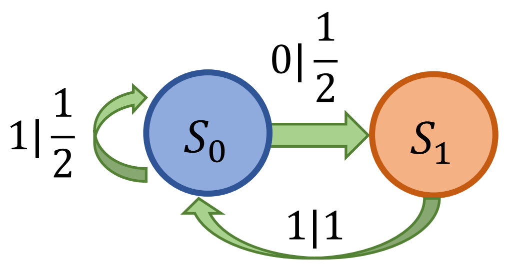
- Make sure the ansatz contains an optimal solution

→ Reverse-Engineering: start from a known solution

Ansatz

- Make sure the ansatz contains an optimal solution

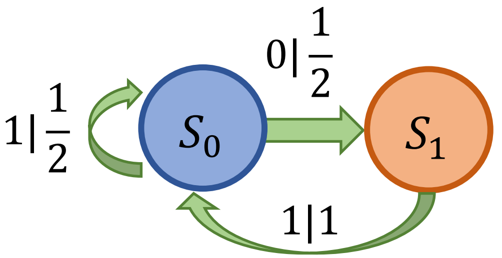
→ Reverse-Engineering: start from a known solution



Ansatz

- Make sure the ansatz contains an optimal solution

→ Reverse-Engineering: start from a known solution



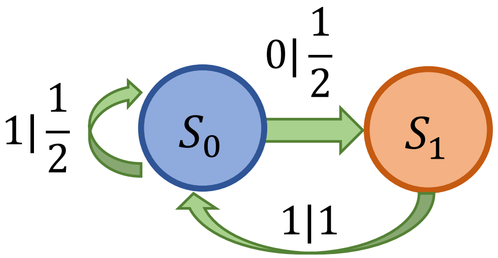
Quantum Memory States:

$$|\sigma_0\rangle = \sqrt{\tfrac{1}{2}}|0\rangle + \sqrt{\tfrac{1}{2}}|1\rangle$$

$$|\sigma_1\rangle = |1\rangle$$

- Make sure the ansatz contains an optimal solution

→ Reverse-Engineering: start from a known solution



Quantum Memory States:

$$|\sigma_0\rangle = \sqrt{\tfrac{1}{2}}|0\rangle + \sqrt{\tfrac{1}{2}}|1\rangle$$

$$|\sigma_1\rangle = |1\rangle$$

Simulator:

$$T|\sigma_0\rangle|0\rangle = \sqrt{\tfrac{1}{2}}|\sigma_0\rangle|0\rangle + \sqrt{\tfrac{1}{2}}|\sigma_1\rangle|1\rangle$$

$$T|\sigma_1\rangle|0\rangle = |\sigma_0\rangle|1\rangle$$

Encoding

Quantum Memory States:

$$|\sigma_0\rangle = \sqrt{\tfrac{1}{2}}|0\rangle + \sqrt{\tfrac{1}{2}}|1\rangle$$

$$|\sigma_1\rangle = |1\rangle$$

$$|0\rangle - \boxed{X} - |\sigma_1\rangle \qquad |0\rangle - \boxed{H} - |\sigma_0\rangle$$

Encoding

Quantum Memory States:

$$|\sigma_0\rangle = \sqrt{\frac{1}{2}}|0\rangle + \sqrt{\frac{1}{2}}|1\rangle$$

$$|\sigma_1\rangle = |1\rangle$$

$|0\rangle - \boxed{X} - |\sigma_1\rangle$

$|0\rangle - \boxed{H} - |\sigma_0\rangle$

but this is static, we want to be able to “learn” it

Encoding

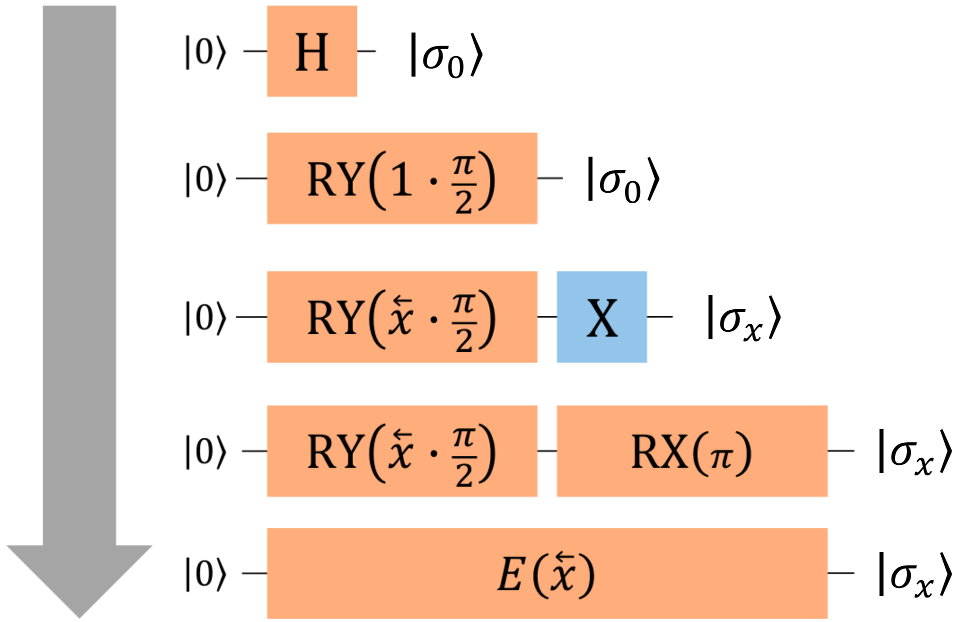
Quantum Memory States:

$$|\sigma_0\rangle = \sqrt{\tfrac{1}{2}}|0\rangle + \sqrt{\tfrac{1}{2}}|1\rangle$$

$$|\sigma_1\rangle = |1\rangle$$

$|0\rangle$ — X — $|\sigma_1\rangle$ $|0\rangle$ — H — $|\sigma_0\rangle$

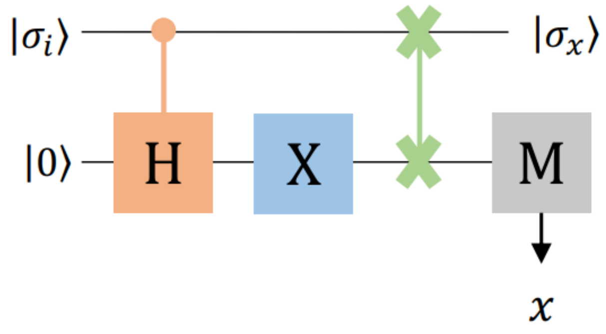
but this is static, we want to be able to "learn" it



Simulator:

$$T|\sigma_0\rangle|0\rangle = \sqrt{\tfrac{1}{2}}|\sigma_0\rangle|0\rangle + \sqrt{\tfrac{1}{2}}|\sigma_1\rangle|1\rangle$$

$$T|\sigma_1\rangle|0\rangle = |\sigma_0\rangle|1\rangle$$

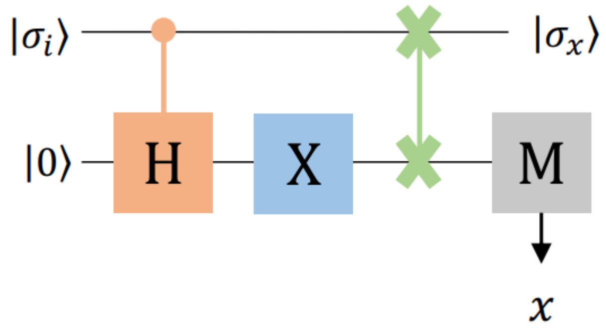


Simulator

Simulator:

$$T|\sigma_0\rangle|0\rangle = \sqrt{\tfrac{1}{2}}|\sigma_0\rangle|0\rangle + \sqrt{\tfrac{1}{2}}|\sigma_1\rangle|1\rangle$$

$$T|\sigma_1\rangle|0\rangle = |\sigma_0\rangle|1\rangle$$



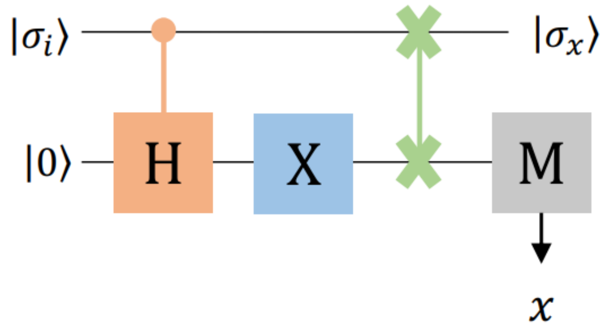
again, we want to be able to "learn" it

Simulator

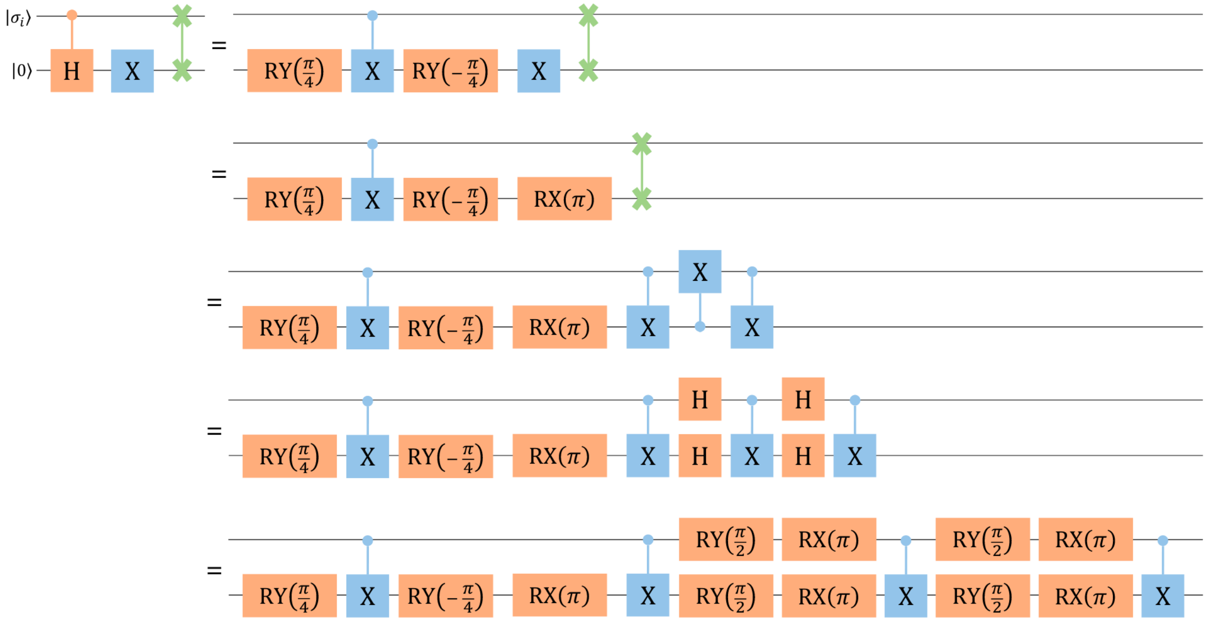
Simulator:

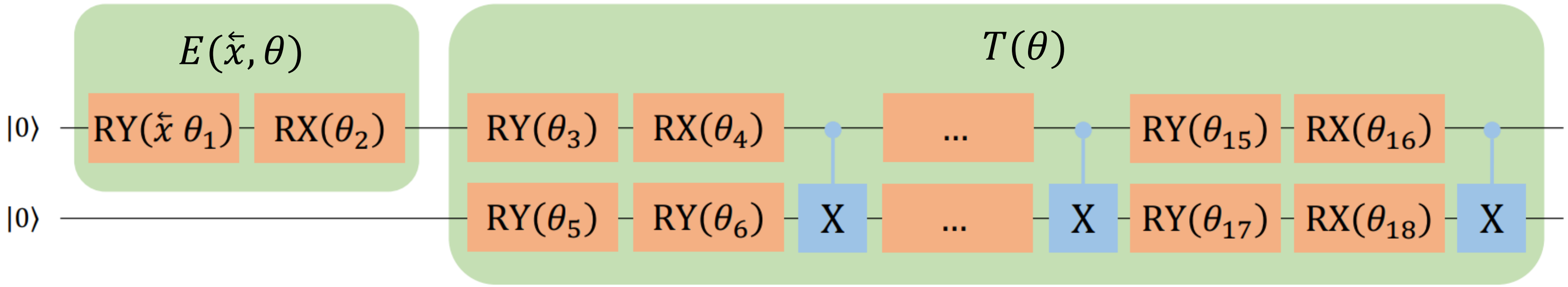
$$T|\sigma_0\rangle|0\rangle = \sqrt{\tfrac{1}{2}}|\sigma_0\rangle|0\rangle + \sqrt{\tfrac{1}{2}}|\sigma_1\rangle|1\rangle$$

$$T|\sigma_1\rangle|0\rangle = |\sigma_0\rangle|1\rangle$$

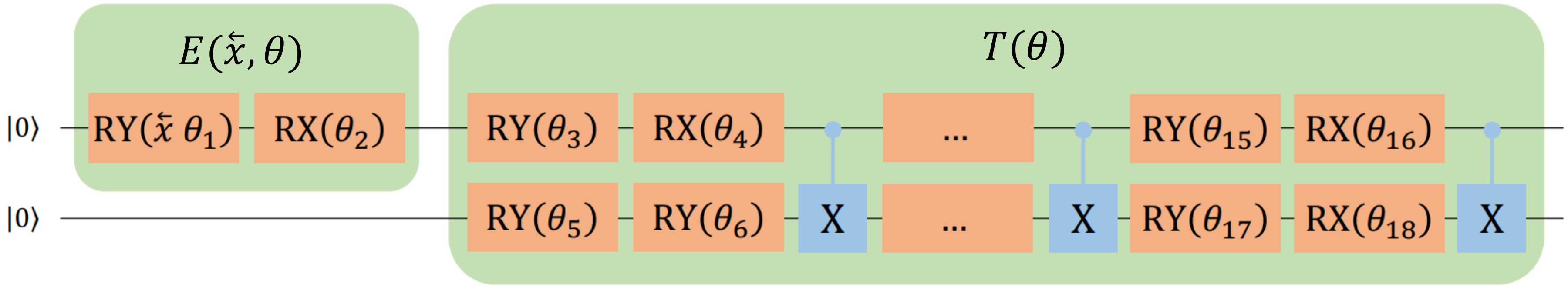


again, we want to be able to "learn" it

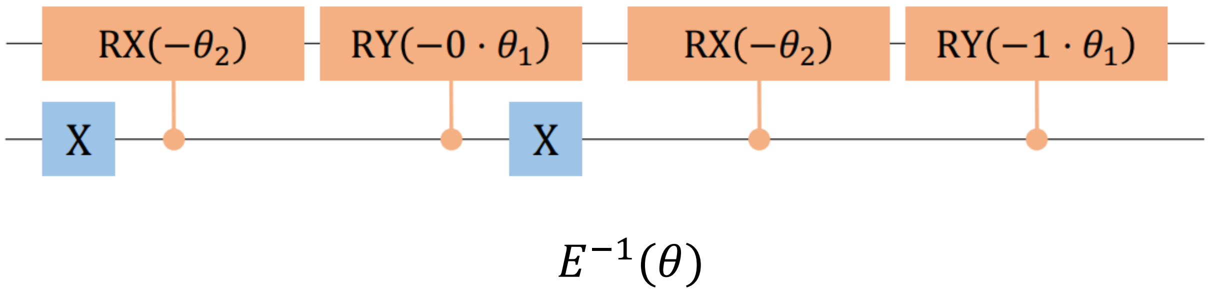



E(x̃, θ)
T(θ)
|0⟩
|0⟩
RY(x̃ θ1)
RX(θ2)
RY(θ3)
RX(θ4)
RY(θ5)
RY(θ6)
X
...
...
X
RY(θ15)
RX(θ16)
RY(θ17)
RX(θ18)
X

Ansatz



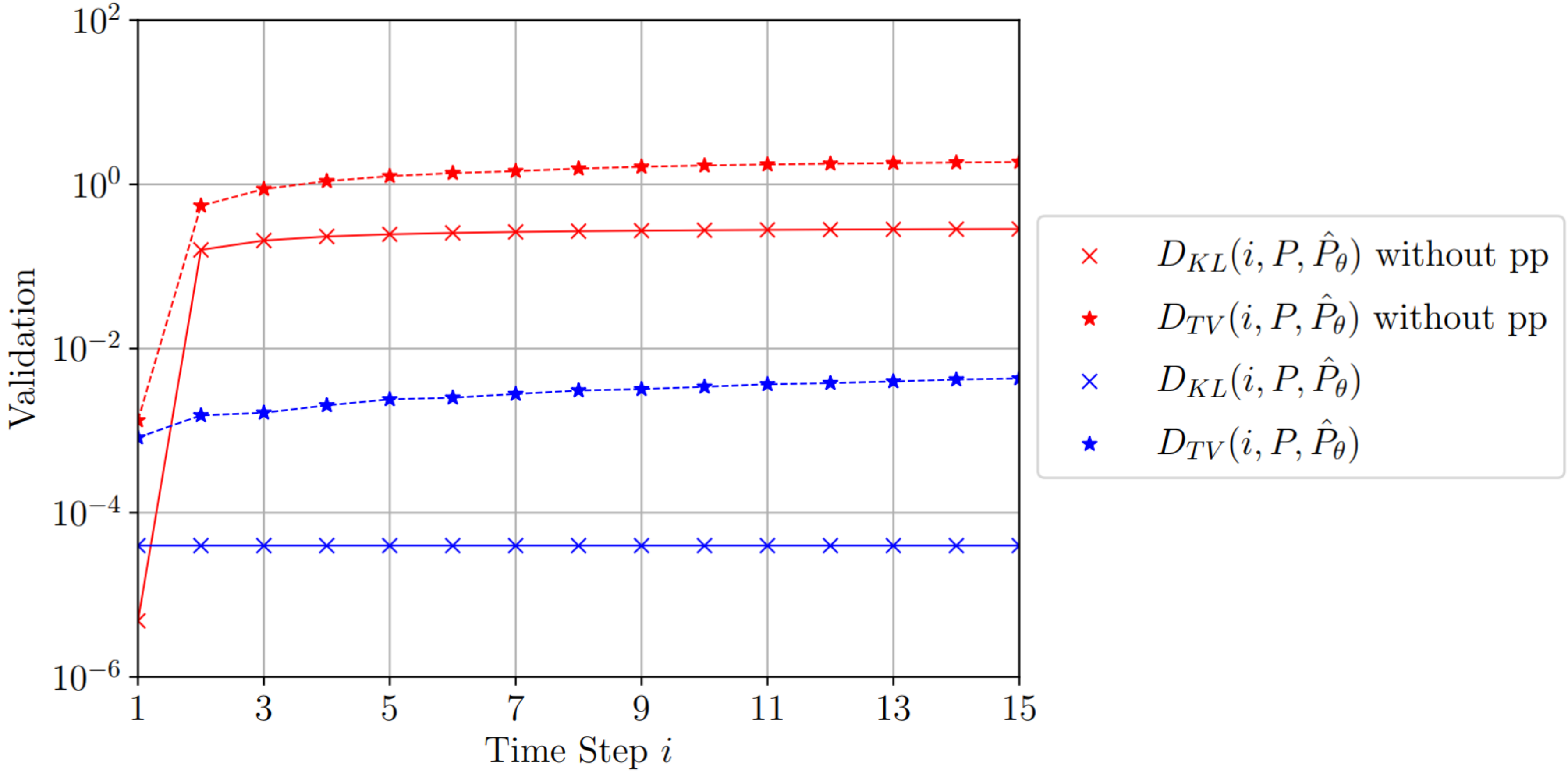
$E(\bar{x}, \theta)$
$T(\theta)$
$|0\rangle$
$|0\rangle$
$RY(\bar{x}\ \theta_1)$
$RX(\theta_2)$
$RY(\theta_3)$
$RX(\theta_4)$
...
$RY(\theta_{15})$
$RX(\theta_{16})$
$RY(\theta_5)$
$RY(\theta_6)$
X
...
X
$RY(\theta_{17})$
$RX(\theta_{18})$
X



$RX(-\theta_2)$
$RY(-0 \cdot \theta_1)$
$RX(-\theta_2)$
$RY(-1 \cdot \theta_1)$
X
X
$E^{-1}(\theta)$

Results

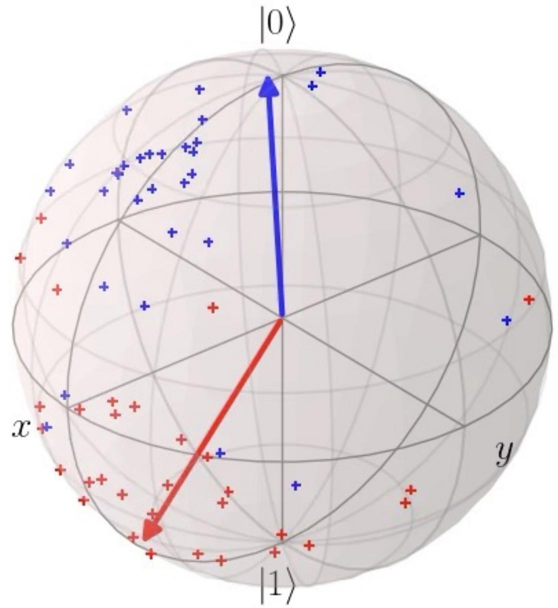
Results



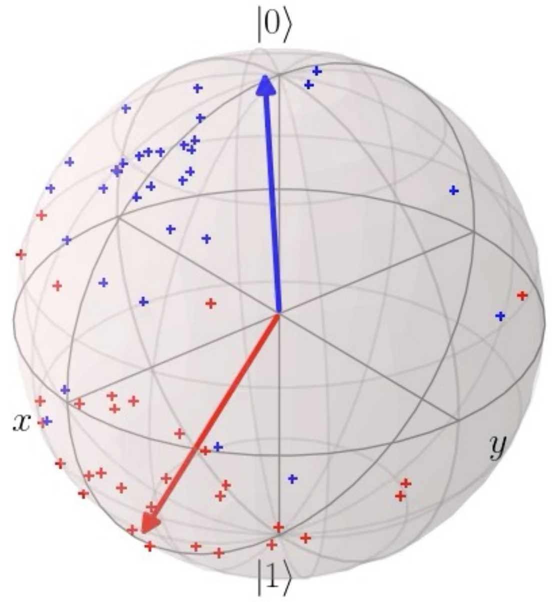
10^2
10^0
10^{-2}
10^{-4}
10^{-6}
Validation
1
3
5
7
9
11
13
15
Time Step i
$D_{KL}(i, P, \hat{P}_\theta)$ without pp
$D_{TV}(i, P, \hat{P}_\theta)$ without pp
$D_{KL}(i, P, \hat{P}_\theta)$
$D_{TV}(i, P, \hat{P}_\theta)$

Results

Without Regularization



With Regularization



$E(\overleftarrow{x}, \boldsymbol{\theta})$

initial state for $\overleftarrow{x} = 1$

initial state for $\overleftarrow{x} = 0$

memory states for $x_i = 1$

memory states for $x_i = 0$

$U(\boldsymbol{\theta})$ $U(\boldsymbol{\theta})$ …

Conclusion
&
Outlook

Conclusion

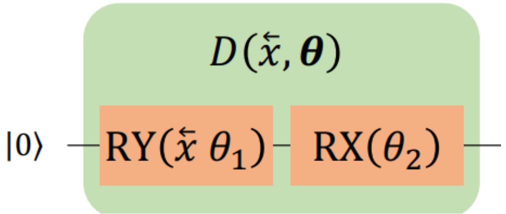
- Developed a hybrid quantum learning algorithm for simulation models
- Learning algorithm is memory efficient
- Extended MMD for simulation models → Decrease KL and TV
- Regularization → small set of memory states
- Learned models show constantly good simulation performance

Outlook

- Apply the algorithm to more complex processes

- Apply the algorithm to more complex processes

→ Data encoding is crucial, e.g.,

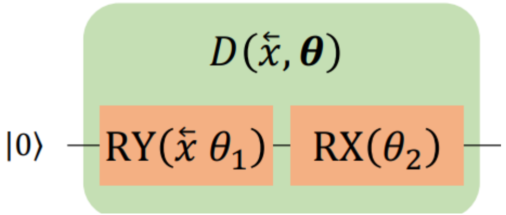


must be able to distinguish all possible inputs

Outlook

- Apply the algorithm to more complex processes

→ Data encoding is crucial, e.g.,



must be able to distinguish all possible inputs

→ Consider general time-series data, i.e., without any/much assumptions

Thank you very much!

Let's discuss