Outline

- Artificial Intelligence
- Machine Learning
- Classical Learning Algorithms
- Quantum Computation and Information Processing
- Quantum Machine Learning
- Our Proposed Work

Artificial Intelligence

- **Artificial intelligence** (**AI**), a.k.a **machine intelligence**, is intelligence demonstrated by machines, in contrast to the **natural intelligence** displayed by humans and other animals.

Artificial Intelligence

- "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.

Artificial Intelligence

- Capabilities generally classified as AI as of 2017
  - Understanding human speech
  - Strategic game systems (Chess)
  - Autonomous cars
  - Content delivery network
  - Military simulations

Artificial Intelligence

- Problems
  - Reasoning, problem solving
  - Knowledge representation
  - Planning
  - Learning
  - Natural language processing
  - Perception
  - Motion and manipulation
  - Social intelligence
  - General intelligence

Artificial Intelligence

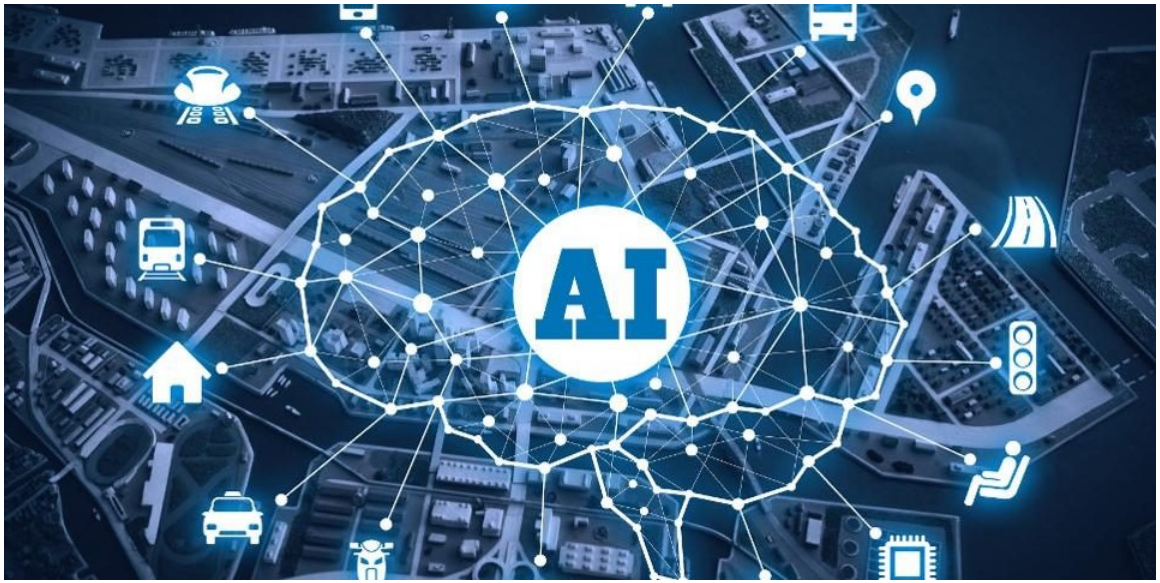
- Approaches
  - Cybernetics and brain simulation
  - Symbolic
  - Sub-symbolic
  - Statistical learning
  - Integrating the approaches

Artificial Intelligence

- Tools
  - Search and optimization
  - Logic
  - Probabilistic methods for uncertain reasoning
  - Classifiers and statistical learning methods
  - Artificial neural networks
  - Evaluating progress

Artificial Intelligence

- Applications
  - Healthcare
  - Finance and economics
  - Automotive
  - Video games
  - Military
  - Audit
  - Advertising
  - Art



Machine Learning

- What is the difference between Machine Learning (ML) and Artificial Intelligence (AI)?

- ML, the ability to learn without being explicitly programmed, is simply a way of achieving AI.

  - ML → AI

Machine Learning

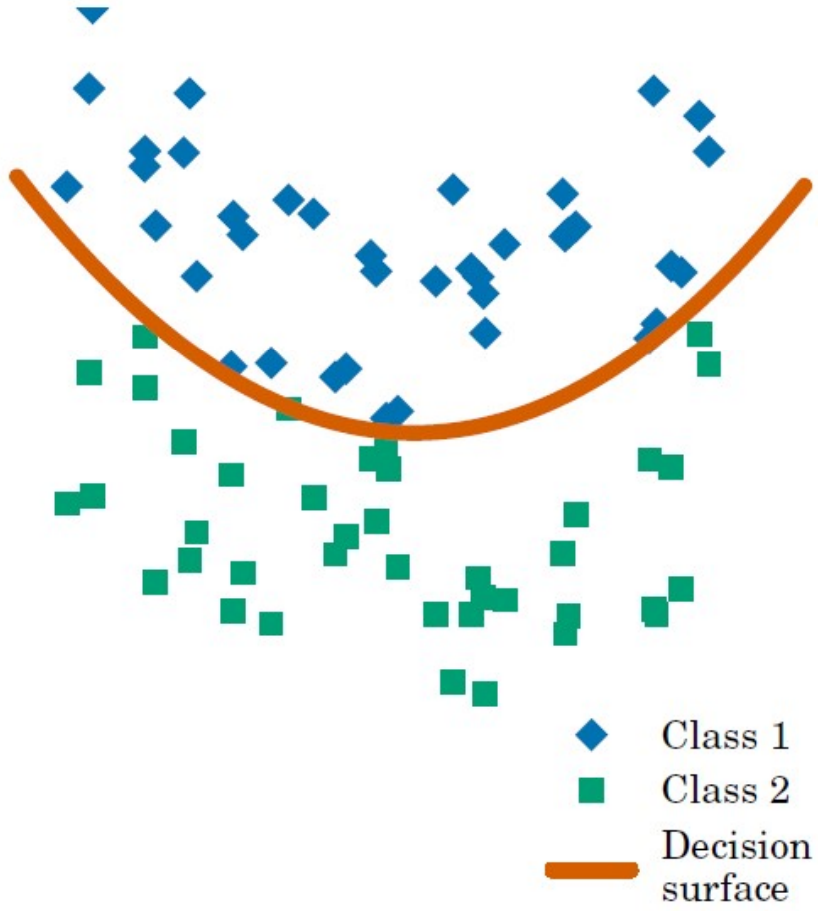
- Machine Learning major categories:
  - Supervised Learning
    - Reinforcement Learning
  - Unsupervised Learning

Machine Learning

- Supervised Learning:

A data set includes its desired outputs (or *labels*) such that a function can calculate an error for a given prediction. The supervision comes when a prediction is made and an error produced (actual vs. desired) to alter the function and learn the mapping.

IBM

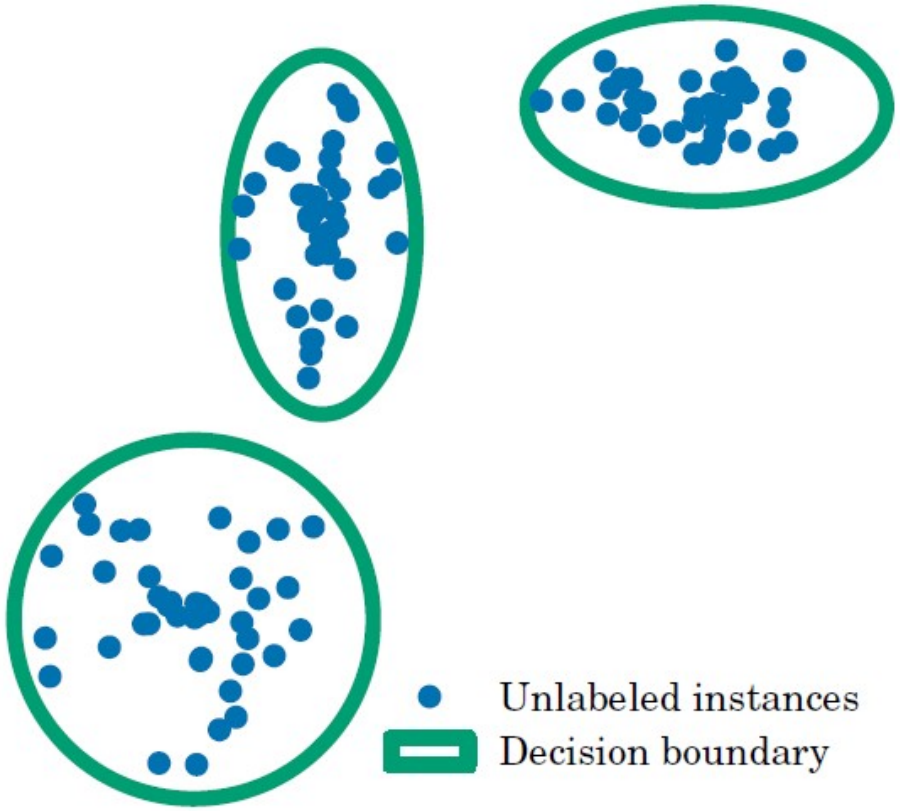


Machine Learning

- Unsupervised Learning:

A data set doesn't include a desired output; therefore, there's no way to supervise the function. Instead, the function attempts to segment the data set into "classes" so that each class contains a portion of the data set with common features.

IBM

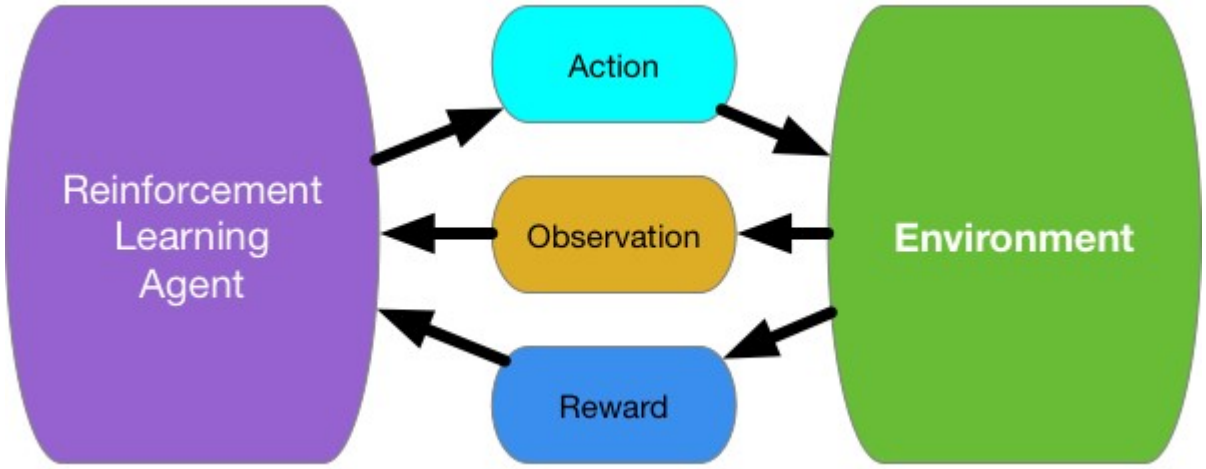


Machine Learning

- Reinforcement Learning:

The algorithm attempts to learn actions for a given set of states that lead to a goal state. An error is provided not after each example (as is the case for supervised learning) but instead on receipt of a reinforcement signal (such as reaching the goal state). This behavior is similar to human learning, where feedback isn't necessarily provided for all actions but when a reward is warranted.

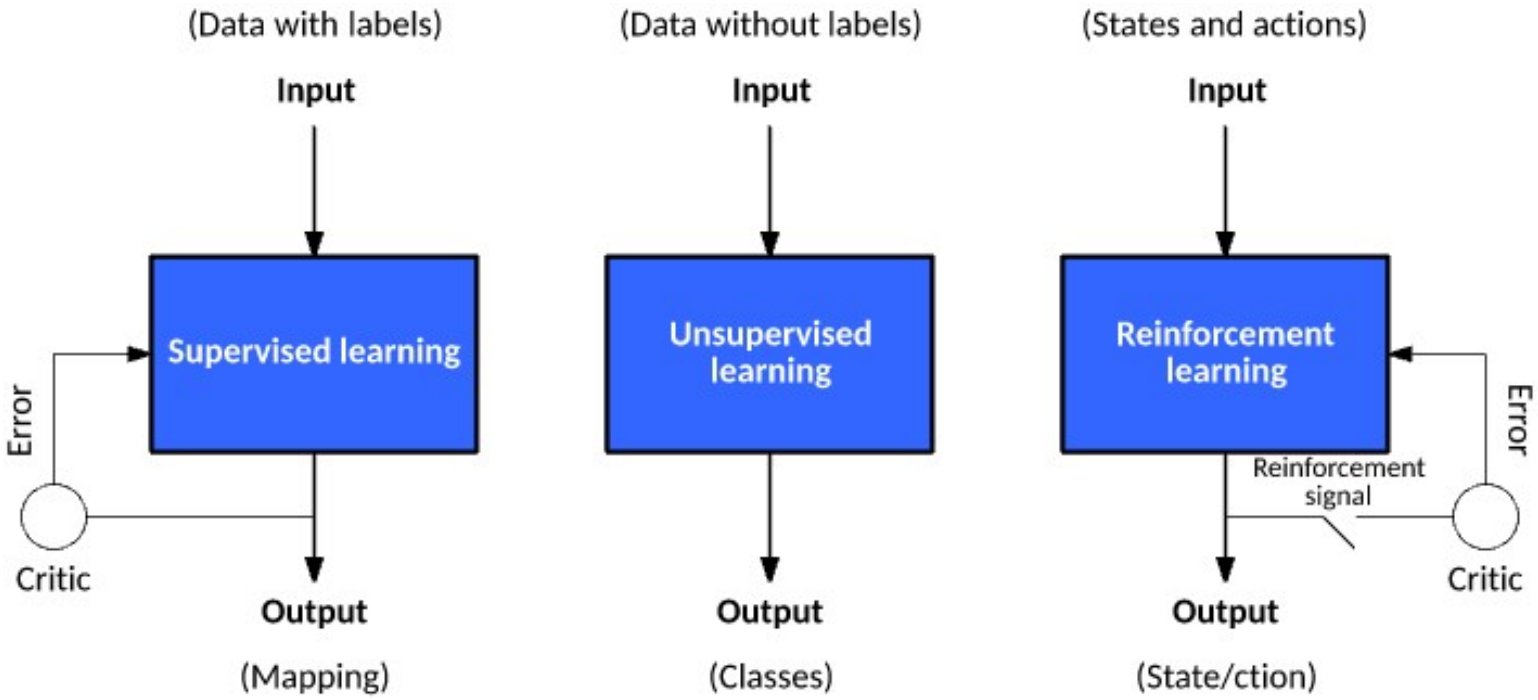
IBM



gluon.mxnet.io

Machine Learning

- Summary of ML major categories:



IBM

Classical Learning Algorithms

- Unsupervised Learning
- Pattern Recognition and Neural Networks
- Supervised Learning and Support Vector Machines
- Regression Analysis
- Boosting

Classical Learning Algorithms

- Unsupervised Learning
  - Principal Component Analysis
  - Manifold Embedding
  - K-Means and K-Medians Clustering
  - Hierarchical Clustering
  - Density-based Clustering

These algorithms are the most relevant to quantum methods that are already published.

Classical Learning Algorithms

- Pattern Recognition and Neural Networks
  - The Perceptron
  - Hopfield Networks
  - Feed-forward Networks
  - Deep Learning
  - Computational Complexity

Classical Learning Algorithms

- Supervised Learning and Support Vector Machines
  - K-Nearest Neighbors
  - Optimal Margin Classifiers
  - Soft Margins
  - Nonlinearity and Kernel Functions
  - Least-Squares Formulation
  - Generalization Performance
  - Multi-Class Problems
  - Loss Functions
  - Computational Complexity

Classical Learning Algorithms

- Regression Analysis
  - Linear Least-Squares
  - Nonlinear Regression
  - Nonparametric Regression
  - Computational Complexity

Classical Learning Algorithms

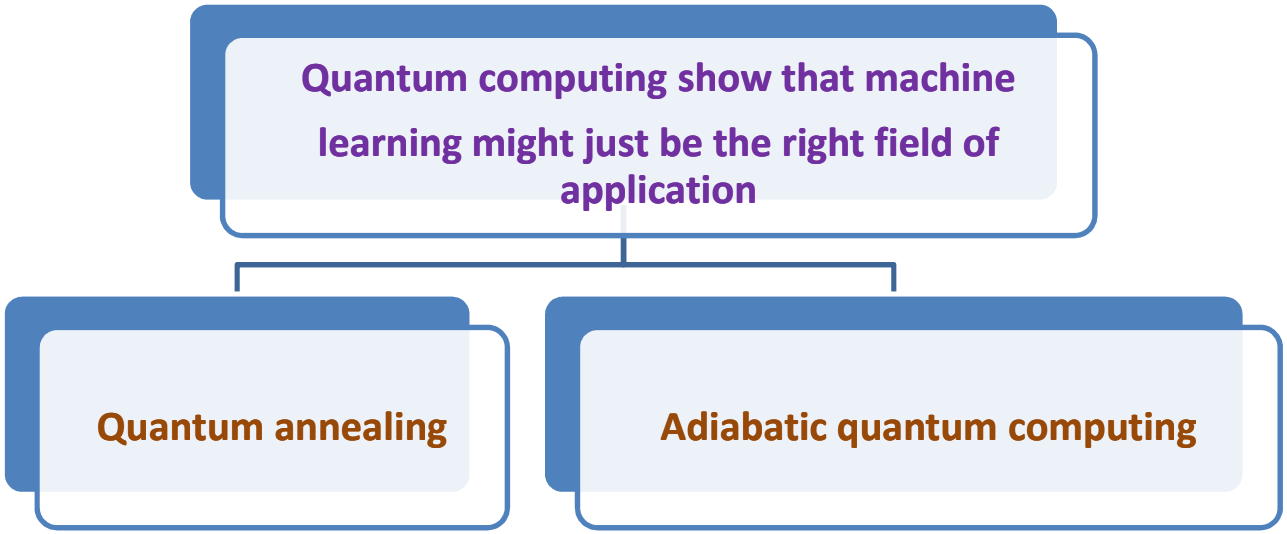
- Boosting
  - Weak Classifiers
  - AdaBoost
  - A Family of Convex Boosters
  - Non-convex Loss Functions

Quantum Computation and Information Processing

- Spectacular theoretical results
  - Factoring of integers is exponentially faster
  - unordered search is quadratically faster

Quantum Computation and Information Processing

- Machine learning usually boils down to a form of multivariate optimization
- →



Quantum Computation and Information Processing

- However, we do not confine ourselves to adiabatic quantum computers.

- The task of learning is far more restricted.

- Other paradigms in quantum information Theory are promising for learning

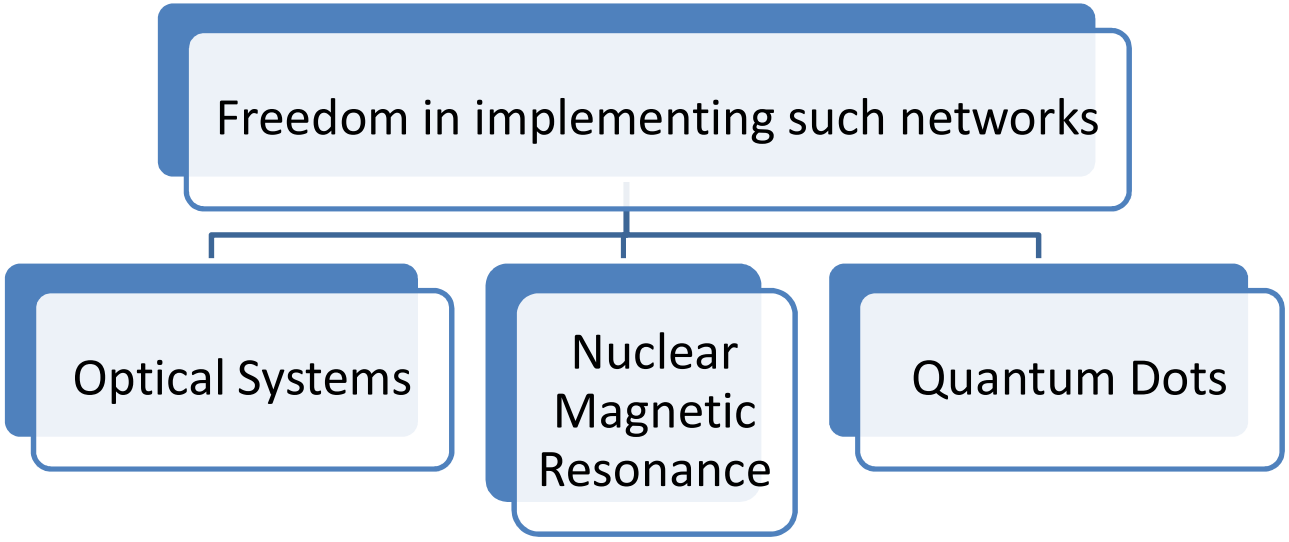
Quantum Computation and Information Processing

- Quantum process tomography is able to learn an unknown function within well defined symmetry and physical constraints

- → Useful for regression analysis.

Quantum Computation and Information Processing

- Quantum neural networks based on arbitrary implementation of qubits offer a useful level of abstraction



Quantum Computation and Information Processing

- **Conclusion:**

- Quantum hardware dedicated to machine learning may become reality much faster than a general-purpose quantum computer.

Quantum Machine Learning

- **Quantum Machine Learning**
  - Clustering Structure and Quantum Computing
  - Quantum Pattern Recognition
  - Quantum Classification
  - Quantum Process Tomography and Regression
  - Boosting and Adiabatic Quantum Computing

Quantum Machine Learning

- Clustering Structure and Quantum Computing
  - Quantum Random Access Memory
  - Calculating Dot Products
  - Quantum Principal Component Analysis
  - Towards Quantum Manifold Embedding
  - Quantum K-Means
  - Quantum K-Medians
  - Quantum Hierarchical Clustering
  - Computational Complexity

Quantum Machine Learning

- Quantum Pattern Recognition
  - Quantum Associative Memory
  - The Quantum Perceptron
  - Quantum Neural Networks
  - Physical Realizations
  - Computational Complexity

Quantum Machine Learning

- Quantum Classification
  - Nearest Neighbors
  - Support Vector Machines with Grover‘s Search
  - Support Vector Machines with Exponential Speedup
  - Computational Complexity

Quantum Machine Learning

- Quantum Process Tomography and Regression
  - Channel-State Duality
  - Quantum Process Tomography
  - Groups, Compact Lie Groups, and the Unitary Group
  - Representation Theory
  - Parallel Application and Storage of Unitary
  - Optimal State for Learning
  - Applying the Unitary and Finding the Parameter for the Input State

Quantum Machine Learning

- Boosting and Adiabatic Quantum Computing
  - Quantum Annealing
  - Quadratic Unconstrained Binary Optimization
  - Ising Model
  - Qboost
  - Nonconvexity
  - Sparsity, Bit Depth, and Generalization Performance
  - Mapping to Hardware
  - Computational Complexity

Our Proposed Work

Application of Quantum Gradient Descent as a Learning Algorithm for Factorization Machines

Problem Statement	• How to find the minimum of a function using quantum gradient descent?
Application	• Implementation of the Factorization Machines (FMs) on quantum computers.
Achievement	• Exponential speed-up over classical algorithm

Our Proposed Work

What is gradient descent?

A first-order iterative optimization algorithm for finding the minimum of a function

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

for γ small enough, then

$$F(\mathbf{a_n}) \geq F(\mathbf{a_{n+1}})$$

Our Proposed Work

- With this observation in mind, one starts with a guess x_0 for a local minimum of F, and considers the sequence x_0, x_1, x_2 such that

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0$$

We have

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$

Our Proposed Work

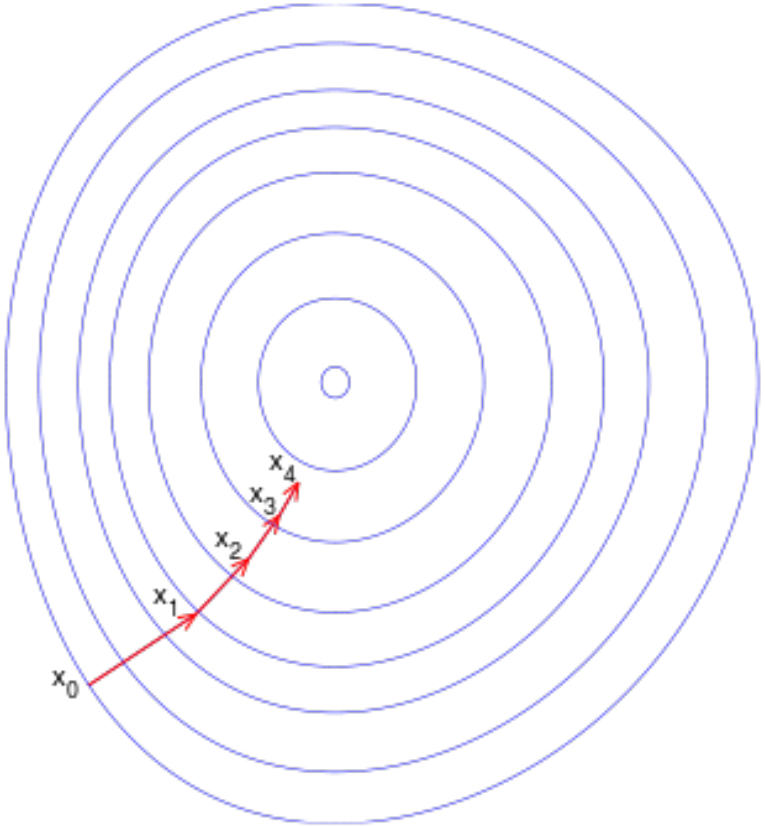


Illustration of gradient descent on a series of level sets

Our Proposed Work

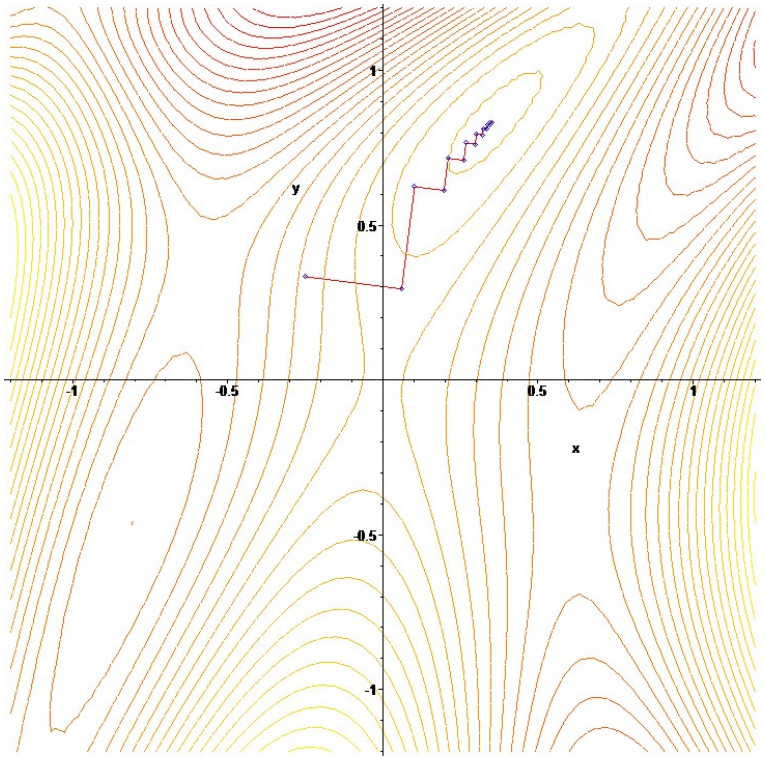
- So hopefully the sequence x_n converges to the desired local minimum. Note that the value of the *step size* γ is allowed to change at every iteration. With certain assumptions on the function F (for example, F convex and ∇F Lipschitz)

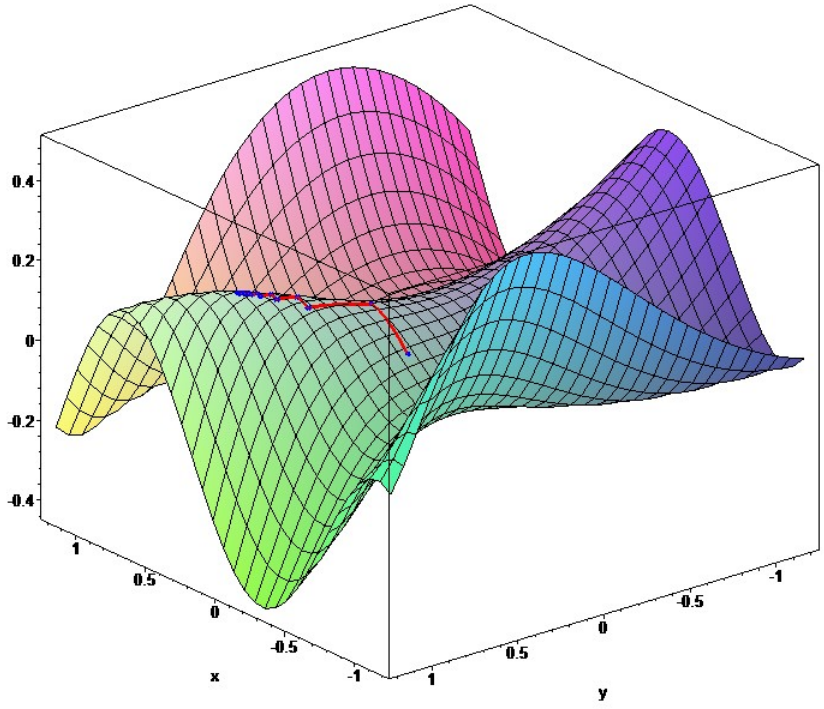
- and particular choices of γ (chosen either via a line search that satisfies the Wolfe conditions or the Barzilai-Borwein)

$$\gamma_n = \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})^T [\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})]}{||\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})||^2}$$

Our Proposed Work

- Example $F(x,y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1 - e^y)$





Our Proposed Work

- Solution of a linear system

- Gradient descent can be used to solve a system of linear equations, reformulated as a quadratic minimization problem, e.g., using linear least squares. The solution of

$$A\mathbf{x} - \mathbf{b} = 0$$

Our Proposed Work

- In the sense of linear least squares is defined as minimizing the function

$$F(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|^2$$

- In traditional linear least squares for real A and b the Euclidean norm is used, in which case

$$\nabla F(\mathbf{x}) = 2A^T(A\mathbf{x} - \mathbf{b})$$

Our Proposed Work

- Solution of a non-linear system

- Gradient descent can also be used to solve a system of nonlinear equations. Below is an example that shows how to use the gradient descent to solve for three unknown variables, x_1, x_2, and x_3.

Our Proposed Work

- Consider the nonlinear system of equations

$$\begin{cases} 3x_1 - \cos(x_2 x_3) - \frac{3}{2} = 0 \\ 4x_1^2 - 625x_2^2 + 2x_2 - 1 = 0 \\ \exp(-x_1 x_2) + 20x_3 + \frac{10\pi - 3}{3} = 0 \end{cases}$$

- Let us introduce the associated function

$$G(\mathbf{x}) = \begin{bmatrix} 3x_1 - \cos(x_2 x_3) - \frac{3}{2} \\ 4x_1^2 - 625x_2^2 + 2x_2 - 1 \\ \exp(-x_1 x_2) + 20x_3 + \frac{10\pi - 3}{3} \end{bmatrix}$$

Our Proposed Work

- Where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

- One might now define the objective function

$$F(\mathbf{x}) = \tfrac{1}{2} G^{\mathrm{T}}(\mathbf{x}) G(\mathbf{x}) = \tfrac{1}{2} \left(\left(3x_1 - \cos(x_2 x_3) - \tfrac{3}{2} \right)^2 + \left(4x_1^2 - 625x_2^2 + 2x_2 - 1 \right)^2 + \left(\exp(-x_1 x_2) + 20x_3 + \tfrac{1}{3}(10\pi - 3) \right)^2 \right)$$

Our Proposed Work

- which we will attempt to minimize. As an initial guess, let us use

$$\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- We know that

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \gamma_0 \nabla F(\mathbf{x}^{(0)})$$

Our Proposed Work

- Where $\nabla F(\mathbf{x}^{(0)}) = J_G(\mathbf{x}^{(0)})^{\mathrm{T}} G(\mathbf{x}^{(0)})$

- The Jacobian matrix J_G is given by

$$J_G(\mathbf{x}) = \begin{bmatrix} 3 & \sin(x_2 x_3) x_3 & \sin(x_2 x_3) x_2 \\ 8x_1 & -1250x_2 + 2 & 0 \\ -x_2 \exp(-x_1 x_2) & -x_1 \exp(-x_1 x_2) & 20 \end{bmatrix}$$

Our Proposed Work

- Evaluating J_G and G at $x^{(0)}$ yields

$$J_G\left(\mathbf{x}^{(0)}\right) = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 20 \end{bmatrix}, \qquad G(\mathbf{x}^{(0)}) = \begin{bmatrix} -2.5 \\ -1 \\ 10.472 \end{bmatrix}$$

- Thus,

$$\mathbf{x}^{(1)} = 0 - \gamma_0 \begin{bmatrix} -7.5 \\ -2 \\ 209.44 \end{bmatrix}$$

Our Proposed Work

- And

$$F\left(\mathbf{x}^{(0)}\right) = 0.5\left((-2.5)^2 + (-1)^2 + (10.472)^2\right) = 58.456$$

- a suitable γ_0 → $F(\mathbf{x}^{(1)}) \leq F(\mathbf{x}^{(0)})$

- simply guess γ_0 =0.001, which gives $\mathbf{x}^{(1)} = \begin{bmatrix} 0.0075 \\ 0.002 \\ -0.20944 \end{bmatrix}$

- $F(\mathbf{x}^{(0)}) = 58.456$ ⟹ $F(\mathbf{x}^{(1)}) = 23.306$

Sizable Decrease!

Our Proposed Work-Classical SVM

- SVM is a supervised machine learning algorithm used for both classification and regression challenges.
- SVM used in classification applications uses M training data points of the form

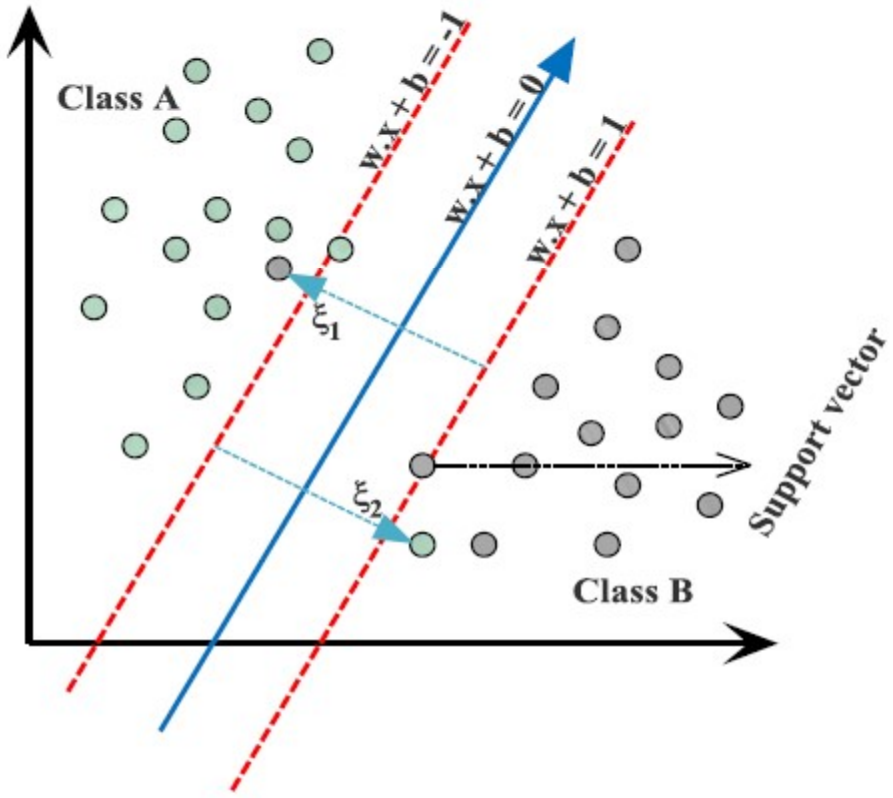
$$\left\{\left(\vec{x}_j, y_j\right) : \vec{x}_j \in R^N, y_j = \pm 1\right\}_{j=1\ldots M}$$

Our Proposed Work-Classical SVM

SVM should find the maximum margin hyperplane with normal vector *w* that divides the two classes.

The General model equation of SVM can be expressed as

$$\hat{y}(\vec{x}) = \langle \phi(\vec{x}), \vec{w} \rangle$$



Our Proposed Work-Classical SVM

- The relation of mapping ϕ with the kernel is

$$K: \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}, \quad K(\vec{x}, \vec{z}) = \langle \phi(\vec{x}), \phi(\vec{z}) \rangle$$

- For instance, in linear kernels $K(\vec{x}, \vec{z}) = 1 + \langle x, z \rangle$
- And the model equation of linear SVM can be written as

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i \quad w_0 \in \mathbb{R}, \quad w \in \mathbb{R}^n$$

Our Proposed Work-Classical Factorization Machine

- The model equation for a FM having a degree d = 2 is defined as

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

Our Proposed Work-Classical Factorization Machine

- where the model parameters below have to be estimated during learning process.

$$w_0 \in \mathbb{R}, \quad w \in \mathbb{R}^n, \quad V \in \mathbb{R}^{n \times k}$$

- And <.,.> is the dot product of two vectors of size k which is defined below.

$$\langle v_i, v_j \rangle = \sum_{f=1}^{k} v_{i,f} . v_{j,f}$$

Our Proposed Work-Classical Factorization Machine

In order to prove that model equation for a FM can be computed in linear time *O(kn), two sigmas including* $\langle v_i , v_j \rangle$, which represent the pairwise interactions, is reformulated below

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle v_i, v_j \rangle x_i x_j$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle v_i, v_j \rangle x_i x_j \; - \frac{1}{2}\sum_{i=1}^{n}\langle v_i, v_i \rangle x_i x_i$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{n} v_{i,f}.v_{j,f}\; x_i x_j \; - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f}.v_{i,f}\; x_i x_i \;\right)$$

$$= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}.x_i\;\right)\left(\sum_{j=1}^{n} v_{j,f}.x_j\;\right) - \sum_{i=1}^{n} v_{i,f}^2 x_i^2\;\right)$$

$$= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}.x_i\;\right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2\;\right)$$

Our Proposed Work-Training Classical Factorization Machine

Model parameters' optimality is usually defined with a loss function l. The task during learning process is to minimize the sum of losses over training samples S

$$OPT(S) = \ arg\min_{\theta} \sum_{(x,y)\in S} l(\hat{y}(x|\theta), y)$$

Our Proposed Work-Training Classical Factorization Machine

ALGORITHM 1: Stochastic Gradient Descent

Input: Training data S, regularization parameters λ, learning Rate η, initialization σ.

Output: Model parameters $\theta = (w_0, w, V)$

$w_0 \leftarrow 0;\ w \leftarrow (0, \dots, 0);\ V \sim \mathcal{N}(0, \sigma)$

repeat

  for $(x, y) \in S$ **do**

    $w_0 \leftarrow w_0 - \eta \left(\frac{\partial}{\partial w_0} l(\hat{y}(x|\theta), y) \right) + 2\lambda^0 w_0);$

    for $i \in \{1, \dots, p\} \wedge x_i \neq 0$ **do**

      $w_i \leftarrow w_i - \eta \left(\frac{\partial}{\partial w_i} l(\hat{y}(x|\theta), y) \right) + 2\lambda^w_{\pi(i)} w_i);$

      for $f \in \{1, \dots, k\}$ **do**

        $v_{j,f} \leftarrow v_{j,f} - \eta \left(\frac{\partial}{\partial v_{j,f}} l(\hat{y}(x|\theta), y) \right) + 2\lambda^v_{f,\pi(i)} v_{i,f});$

      end

    end

  end

until *stopping criterion is met;*

Among the very popular algorithms for optimizing factorization models, stochastic gradient descent algorithms work suitably for different loss functions

Our Proposed Work-Quantum Gradient Descent

We now show that algorithm 1 can be implemented on a quantum computer using quantum gradient descent.

We intend to minimize the following function,

$$p(\theta) = \sum_{(x,y)\in S} l(\hat{y}(x|\theta), y) + \sum_{\theta' \in \theta} \lambda_{\theta'} \theta'^2$$

Where, $$l(\hat{y}(x|\theta), y) = (\hat{y}(x|\theta) - y)^2$$

We divide this function into two parts $$p(x) = f(x) + f_{inhom}(x)$$

Our Proposed Work-Quantum Gradient Descent

The general objective function, first discussed here, on which quantum gradient descent can be applied is a homogeneous polynomial of order 2p defined over

$$x \in \mathbb{R}^N$$

$$f(x) = \frac{1}{2} \sum_{i_1,\ldots,i_{2p}=1}^{N} a_{i_1,\ldots,i_{2p}} x_{i_1} \ldots x_{i_{2p}}$$

where $a_{i_1,\ldots,i_{2p}} \in \mathbb{R}$ and $x = (x_1, \ldots, x_N)^T$

this function can be rewritten in a quadratic form,

$$f(x) = \frac{1}{2} x^T \otimes \ldots \otimes x^T A\, x \otimes \ldots \otimes x$$

Our Proposed Work-Quantum Gradient Descent

A is a $N^P \times N^P$ dimensional matrix.

If p=2, for instance , the two-dimensional input $x = (x_1, x_2)^T$ get projected to a vector containing the polynomial terms up to second order

$$x \otimes x = (x_1^2, x_1 x_2, x_2 x_1, x_2^2)^T$$

and A is of size 4×4

$$A = \sum_{\alpha=1}^{K} A_1^\alpha \otimes \ldots \otimes A_p^\alpha$$

Our Proposed Work-Quantum Gradient Descent

Since p=2, the objective function reduces to

$$f(v_f) = \frac{1}{2} {v_f}^T \otimes {v_f}^T A\, v_f \otimes v_f$$

The gradient of the objective function at point x is

$$\nabla f(x) = \sum_{\alpha=1}^{K} \sum_{j=1}^{p} \left(\prod_{\substack{i=1 \\ i \neq j}}^{p} x^T A_i^\alpha\, x \right) A_j^\alpha x = Dx$$

Our Proposed Work-Quantum Gradient Descent

Similarly, the Hessian matrix, used for Newton's method, at the same point is defined below as the operator H_A

$$\mathrm{H}\big(f(x)\big) = \sum_{\alpha=1}^{K} \sum_{\substack{j,k=1 \\ j\neq k}}^{p} \prod_{\substack{i=1 \\ i\neq j,k}}^{p} (x^T A_i^\alpha\, x) A_k^\alpha\, x\, x^T A_j^\alpha + D$$

where Newton's method is another technique for finding the minima of a function. It can improve convergence using curvature information

Our Proposed Work-Quantum Gradient Descent

By applying these operators on FM model we obtain

$$\nabla f(v_f) = \sum_{\alpha=1}^{K} {v_f}^T A_2^\alpha v_f \, A_1^\alpha v_f + {v_f}^T A_1^\alpha v_f \, A_2^\alpha v_f$$
$$= D v_f$$

$$\mathrm{H}\left(f(v_f)\right) = \sum_{\alpha=1}^{K} A_1^\alpha x \, x^T A_2^\alpha + A_2^\alpha x \, x^T A_1^\alpha$$
$$+ D$$

Our Proposed Work-Quantum Gradient Descent

Since in the following parts, vectors v_f is presented as quantum states, the produced vectors by the quantum algorithm are normalized with

$$v_f^T v_f = 1$$

thus, the problem we solve is

$$\min_{v_f} f(v_f) \ \ subject\ to\ v_f^T v_f = 1$$

Our Proposed Work-Quantum Gradient Descent

we propose to construct a n-qubit quantum state

$$|x\rangle = \sum_{j=1}^{N} x_j |j\rangle$$

with amplitudes $x = (x_1, \dots, x_N)^T$

$$v_f = (v_{1,f}, \dots, v_{n,f})^T$$

$$|v_f\rangle = \sum_{j=1}^{N} v_{j,f} |j\rangle$$

where |*j*> is the j-th computational basis state

Our Proposed Work-Quantum Gradient Descent

It is critical to pay attention to the scalar coefficient

weighing factor

$$x^T A_i^\alpha \, x \longrightarrow \nabla f(x) = \sum_{\alpha=1}^{K} \sum_{j=1}^{p} \left(\prod_{\substack{i=1 \\ i \neq j}}^{p} x^T A_i^\alpha \, x \right) A_j^\alpha x = Dx$$

$$\langle x | A_j^\alpha | x \rangle = tr\{A_j^\alpha \rho\}$$

where

$$\rho = |x\rangle\langle x|$$

density matrix

Operator $D = tr_{1\ldots p-1}\{\rho^{\otimes(p-1)} M_D\}$

$$M_D = \sum_{\alpha=1}^{K} \sum_{j=1}^{p} \left(\bigotimes_{\substack{i=1 \\ i \neq j}}^{p} A_j^\alpha \right) \otimes Aj^\alpha$$

Our Proposed Work-Quantum Gradient Descent

where $\rho^{\otimes(p-1)}$ **is the joint quantum state of p-1 copies of** ρ

$$v_f^{(0)} = (v_{1,f}^{(0)}, \dots, v_{n,f}^{(0)})^T$$

The initial point we choose as a candidate for finding the minimum

$$v_f^* \text{with } \sum_i |v_{i,f}^{(0)}|^2 = 1$$

Now we can assume, at step t, there are multiple copies of the current solution

$|v_f^{(t)}\rangle$ **with an accuracy** $\epsilon^{(t)} > 0$

Our Proposed Work-Quantum Gradient Descent

Purpose:

$$|v_f^{(t)}\rangle \longrightarrow |v_f^{(t+1)}\rangle \quad \text{And} \quad \epsilon^{(t+1)} > \epsilon^{(t)}$$

We prepare the state:

$$(cos\theta|0\rangle - i\sin\theta|1\rangle)\,|v_f^{(t)}\rangle$$

where θ is an external parameter, the first and second register contain a single ancilla qubit and $|v_f^{(t)}\rangle$

Our Proposed Work-Quantum Gradient Descent

If ancilla would be in state |1>

by multiplying the operator D to $|v_f^{(t)}\rangle$

$$|\psi\rangle = \frac{1}{\sqrt{P_D}}\left(cos\theta|0\rangle\,|v_f^{(t)}\rangle - i\,C_D \sin\theta|1\rangle D\,|v_f^{(t)}\rangle\right)$$

Where $C_D = O(1/\mathcal{K}_D)$ **Where** $\mathcal{K}_D$

is the condition number of D

The success probability is $P_D = cos^2\theta + C_D{}^2 sin^2\theta \left\langle v_f^{(t)}\middle|D^2\middle|v_f^{(t)}\right\rangle$

Our Proposed Work-Quantum Gradient Descent

The state $|\psi>$ is measured in the basis

$$|yes\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \qquad \& \qquad |no\rangle = \frac{1}{\sqrt{2}}(i|0\rangle + |1\rangle)$$

Measurement of 'yes' basis state brings about being in below state in the quantum system

$$|v_f^{(t+1)}\rangle = \frac{1}{\sqrt{2P_D P_{yes}^{grad}}}\left(cos\theta \left| v_f^{(t)}\right\rangle - C_D \sin\theta \, |v_f^{(t)}\rangle\right)$$

Our Proposed Work-Quantum Gradient Descent

where θ can be chosen as

$$cos\,\theta = \frac{1}{\sqrt{1 + {\eta^2}/{C_D{}^2}}}, \quad sin\,\theta = \frac{\eta}{C_D\sqrt{1 + {\eta^2}/{C_D{}^2}}}$$

leads to the state $$|v_f^{(t+1)}\rangle = \frac{1}{C_{grad}^{(t+1)}}\left(|v_f^{(t)}\rangle - \eta\,|\nabla f\left(v_f^{(t)}\right)\rangle\right)$$

where $$(C_{grad}^{(t+1)})^2 = 1 - 2\eta(\langle v_f^{(t)}|D|v_f^{(t)}\rangle + \eta^2\langle v_f^{(t)}|D^2|v_f^{(t)}\rangle)$$

Is the normalization factor and η is the learning rate.

Our Proposed Work-Quantum Gradient Descent

Getting this state by a successful 'yes' measurement has the probability of

$$P_{yes}^{grad} = \frac{1}{2} - \frac{2\eta\left\langle v_f^{(t)} \middle| D \middle| v_f^{(t)} \right\rangle}{1 + \eta^2 \left\langle v_f^{(t)} \middle| D^2 \middle| v_f^{(t)} \right\rangle}$$

An approximation to a normalized version of the classical matrix $V \in \mathbb{R}^{n \times k}$

can be obtained via learning each column using quantum gradient descent.

The approximation of each column is the quantum state

$$|v_f^{(t+1)}\rangle$$

Our Proposed Work-Quantum Gradient Descent

$$|v_f^{(t)}\rangle \longrightarrow \epsilon^{(t)}$$

$$D|v_f^{(t)}\rangle \longrightarrow \epsilon_D$$

The error in preparing $|v_f^{(t+1)}\rangle \longrightarrow \epsilon^{(t+1)} = O(\eta^{(t)}\epsilon_D{}^{(t)} + \epsilon^{(t)})$

$$\mathrm{D}|v_f\rangle = |\nabla f(v_f)\rangle \xrightarrow{\text{using matrix exponentiation}} e^{iD\Delta t}|x\rangle$$

Our Proposed Work-Quantum Gradient Descent

For a short time Δt, multiple copies of $\rho = |v_f\rangle\langle v_f|$ are used and matrix exponentiation of M_D is performed

Then, in reduced space of the last copy of ρ, the operation below is observed,

$$tr_{p-1}\{e^{-iM_D\Delta t}\rho^{\otimes p}e^{iM_D\Delta t}\} = e^{-iD\Delta t}\rho e^{iD\Delta t} + O(\Delta t^2)$$

Then, phase estimation is performed and eigenvalues of D are extracted

Next, by conditional rotating and measuring an extra ancilla qubit, the matrix multiplication with D is performed

Our Proposed Work-Quantum Gradient Descent

Our Proposed Work-Quantum Gradient Descent

For a single gradient descent step, to perform phase estimation which results in preparing a next copy,

n_{ph}^{grad} **copies of the current solution is required**

$$n_{ph}^{grad} = \left(\left(1 + \xi^{(t)}\right) \frac{1}{\left(\epsilon_D{}^{(t)}\right)^3} \right)$$

Our Proposed Work-Quantum Gradient Descent

To perform multiple step gradient descent consisting of T iteration, we need

n'^{grad}_{ph} **copies of the initial state** $|v_f^{(0)}\rangle$

$$n'^{grad}_{ph} = \left(\left[\frac{1}{\sqrt{\widetilde{P_{yes}} \widetilde{P_D}}} \right]^T [n^{grad}_{ph}]^T \right)$$

where $\widetilde{P_D} = min_t {P_D}^{(t)}$ **In addition, each copy requires** $O(\beta \log n)$

where β depends on the sparsity of the matrix A

Our Proposed Work-Quantum Gradient Descent

Now to optimize the objective function including inhomogeneous part,

$$f_{inhom}(v_f) = \sum_{j=1}^{p-1} (c_j^T v_f) \prod_{i=1}^{j-1} (v_f{}^T \widehat{B_{ji}} v_f)$$

First, we compute the gradient of the inhomogeneous part for a given j,

$$\nabla \left(\prod_{i=1}^{j} (v_f{}^T \widehat{B_{ji}} v_f)(c_j^T v_f) \right) =$$

Our Proposed Work-Quantum Gradient Descent

$$(\nabla \prod_{i=1}^{j} (v_f{}^T \widehat{B_{Ji}} v_f))(c_j^T v_f) + (\prod_{i=1}^{j} (v_f{}^T \widehat{B_{Ji}} v_f))\nabla(c_j^T v_f)$$

$$=$$

$$(c_j^T v_f)(\sum_k \prod_{i=1;i\neq k}^{j} (v_f{}^T \widehat{B_{Ji}} v_f)\widehat{B_{Jk}} v_f) + \prod_{i=1}^{j} (v_f{}^T \widehat{B_{Ji}} v_f) c_j$$

$$= D_j^{inhom,1} v_f + D_j^{inhom,2} c_j$$

Our Proposed Work-Quantum Gradient Descent

By performing phase estimation, a conditional rotation, and post selection, D^{inhom} is applied to the current state

Assume that probability of successfully adding terms is P_{add}.

In addition, assume that applying the gradient of the homogeneous and inhomogeneous polynomial to the current state gets successful with the lower bound probability of  P_G^{inh}

We define *s* as the maximum sparsity of the homogeneous and inhomogeneous parts' matrices.

Our Proposed Work-Quantum Gradient Descent

All in all, the inhomogeneous polynomials will need

$$n_{ph}^{grad,inh} = \left(\frac{1}{\sqrt{P_G^{inh} \, P_{add} P_{yes}^{inh}}} \times n_{ph}^{grad} \times s \log n \right)$$

copies of the initial state for a single step of gradient descent

Our Proposed Work-Quantum Gradient Descent

Comparing computational complexity of training FM by using classical and quantum gradient descent

	training FM by using classical gradient descent	**training FM by using quantum gradient descent**
Time Computational complexity for one iteration	$O(kn)$	$O(k \, poly \, log \, n)$

Thanks for your attention!

Merci de votre attention!