양자 프로그래밍 5 강
Quantum Support Vector Machine

Quantum Ant

HSU 한성대학교 HANSUNG UNIVERSITY

QUANTUM ANT
CRYPTO OF THE SWARM

CryptoCraft LAB

Support Vector Machine (SVM)

- **Support Vector Machine (SVM)**

  - 초평면을 통해 **데이터 포인트 간의 최적 경계를 찾는** 지도 머신러닝 알고리즘

  - **분류 및 회귀에 사용**

  - **초평면** : n차원의 공간을 나누기 위한 n-1차원

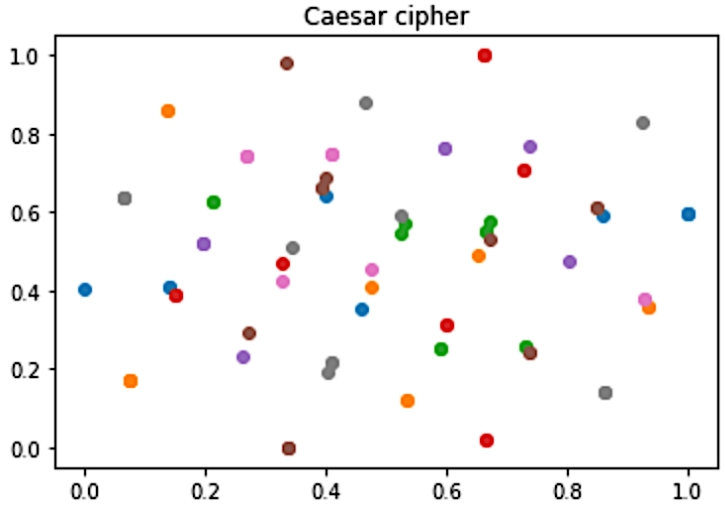
  - **n차원 공간을 나누기 위해 kernel** 사용

    → kernel은 다양한 초평면을 잘 배치하여 공간을 잘 나눌 수 있도록 함

  - 이러한 초평면을 찾기 위해서는 데이터에 **비선형 함수를 적용**해야 함

    → **feature map** 이라고 하며, 다항식, 시그모이드, 가우스 함수 등이 존재

  - kernel 함수는 **데이터 포인트 간의 경계를 최대화하여 효율적으로 분리**하도록 함



Quantum Support Vector Machine (QSVM)

- **Quantum Support Vector Machine (QSVM)**

  - 고전 SVM의 kernel 연산을 **양자컴퓨터 상에서 수행**한 것

    → 양자 컴퓨터 상에서의 비선형 연산 수행

  - 기존 프로세스와 동일

  - 양자컴퓨팅은 고전 컴퓨팅과 달리 큐비트 개수에 따라 데이터 공간 차원이 기하급수적으로 늘어남

    → **고차원 데이터 특징 공간으로 데이터를 옮기기 때문**

      → 고전 컴퓨터에 비해 고차원 데이터 작업에 유리

  - 즉, 고전 SVM이 처리하기 어려웠던 **고차원 kernel 최적화를 QSVM을 통해 효율적으로 수행**

  - 일반적으로 SVM 보다 성능 이점 존재

Quantum Support Vector Machine (QSVM)

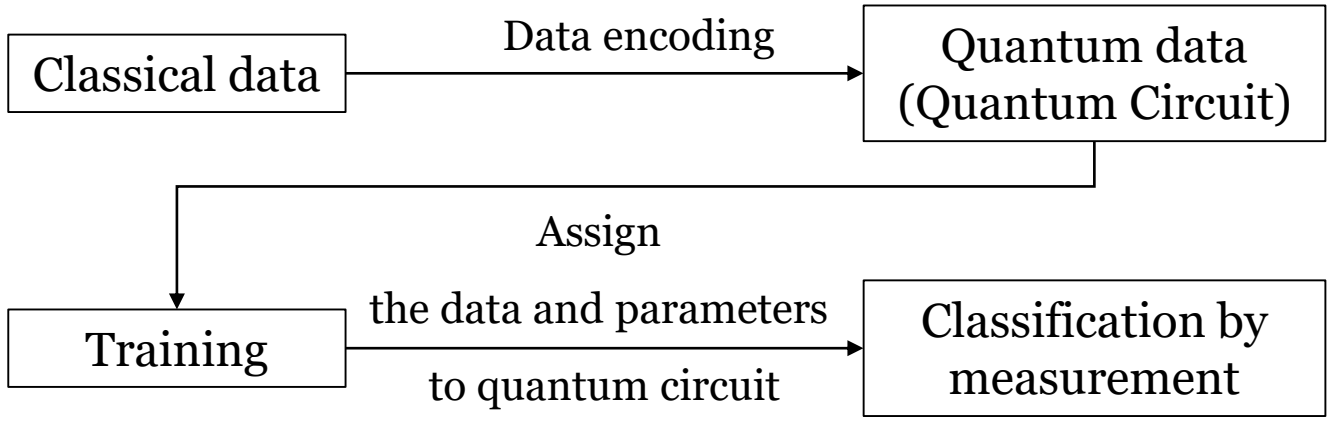
- **전체 프로세스**

  1. Data encoding : 고전 입력 데이터(x)를 양자 데이터 ($\phi(x)>$)로 옮겨야 함

     양자 데이터는 양자 회로로 표현 (데이터에 영향을 받는 회로가 됨)

     → 해당 회로는 매개변수화 됨 (입력 데이터를 양자 게이트의 매개변수로 사용)

  2. 회로 실행 후 측정 → 입력 데이터 분류 (확률 값 반환)



Quantum Support Vector Machine (QSVM)

- **Quantum Circuit 구성**

  - SVM의 kernel 역할 (비선형 함수)를 수행하도록 설계

    → 즉, 해당 회로는 QSVM의 kernel (feature map)을 의미

  - Qiskit에서 3가지 feature map을 제공 (Z, ZZ, Pauli)

  - QSVM의 feature map ($V(\Phi(\vec{x})) = U(\Phi(\vec{x})) \otimes H^{n}$)을 제공하는 게이트가 없음

    → 이를 표현하기 위해 두 게이트를 조합해야 함 (1-qubit rotation and CNOT)

* ϕ는 비선형 함수를 의미

Quantum Support Vector Machine (QSVM)

- **Quantum Circuit 구성**

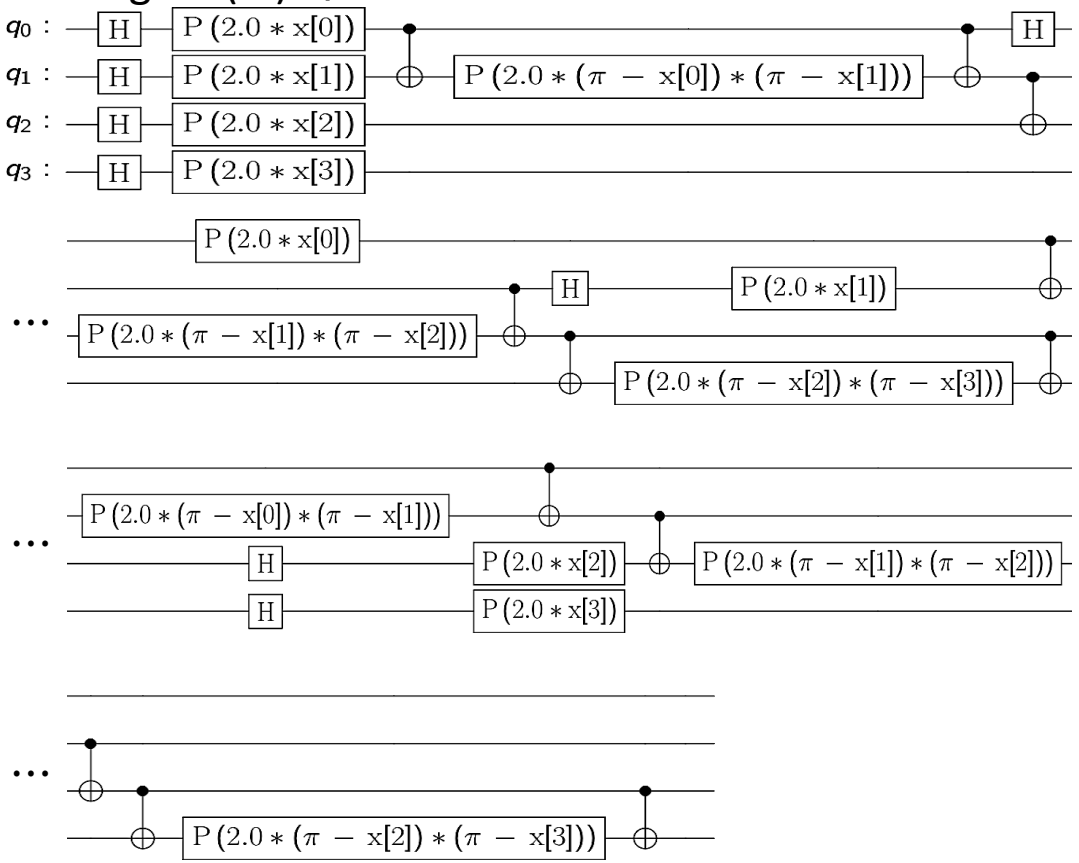
  - 입력 데이터의 차원만큼의 큐비트 할당

  - 입력 데이터들을 중첩 상태로 만들기 위해 모든 qubit에 Hadamard gate (H)적용

  - 다음 수식 (회로에서 P)에 따라 입력 데이터 할당

    $$\Phi_{u,v}(\vec{x}) = (\pi - x_u)(\pi - x_v)$$

  - CNOT 게이트를 통해 큐비트들을 얽힘 상태로 만든 후 연산



Quantum Support Vector Machine (QSVM)

- **Training and measurement**

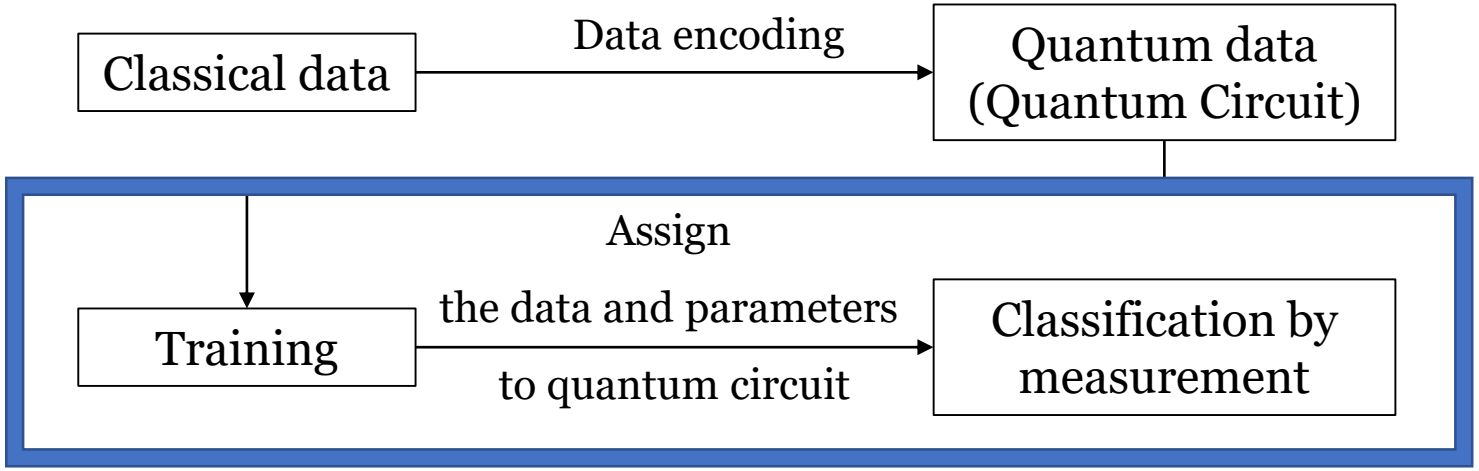
  - 설계된 양자 회로를 반복적으로 실행 (reps)

    → 회로의 파라미터 갱신

  - 하나의 큐비트 당 여러 번의 측정을 수행하여 높은 확률로 분류 (shots)

  - 학습이 완료된 후의 양자 회로는 분류기로써의 역할 수행

    → 테스트 데이터 입력하여 추론 가능 (기존 신경망과 동일)

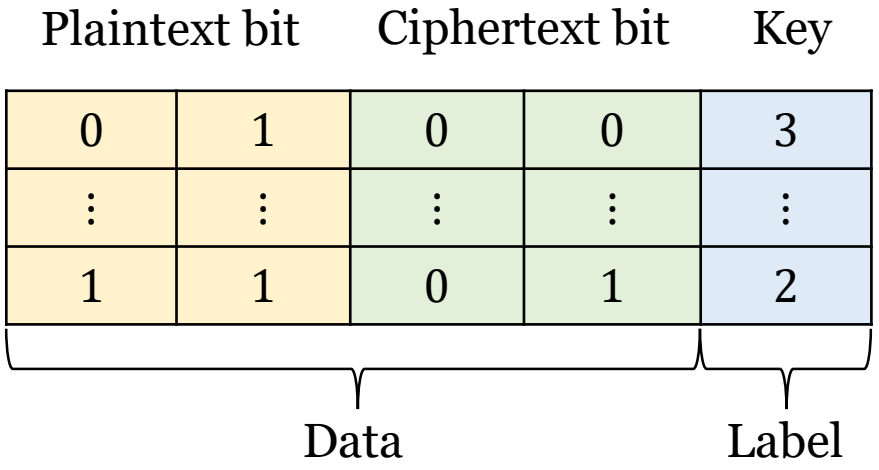


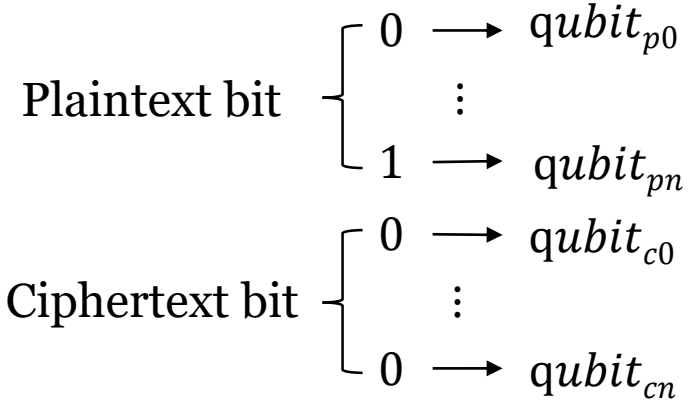
실 습

https://colab.research.google.com/drive/1ToVvnNQo0BOZ4YYh0lHzI8pFveQBIZiE?usp=sharing

- **Example**

  - Caesar cipher의 key 찾기
  - 클라우드 환경 문제로 인해 2-bit, 3-bit 평문 및 키에 대해서만 수행했음
  - 아래 그림은 데이터 구성 방식 (아래 경우는 4차원 데이터이므로 4개의 큐비트를 각 feature에 할당)
  - 이런 방식으로 다른 데이터들도 학습 가능





실 습

- **Example**

  - Google Colaboratory (Intel Xeon CPU (25GB RAM), Nvidia GPU (25GB RAM) 및 Ubuntu 18.04.5 LTS)

  - 프로그래밍 환경 : Python 3.7.11 및 Qiskit 라이브러리

    → Qiskit에서는 IBM의 실제 양자 하드웨어를 사용 가능 but 5-qubit 이상은 토큰 필요하여 시뮬레이터 사용

  - 표2 는 실험 결과

Table 2. Result of classification for cryptanalysis

Accuracy	2-bit dataset (plain, cipher and key)	3-bit dataset (plain, cipher and key)
Maximum	1.0	0.84
shots = 1	0.66	0.6
shots = 5	1.0	
shots = 100	-	0.81
shots = 150	-	0.84

감사합니다