Challenge 3: Image classification with QML

QSVM - quantum support vector machine for classification

QGANs → as the name suggests

Qiskit ML demo isn't super clear... if you don't already know the QC theory

→ quantum circuit encodes the feature map Ex $\phi(x) = (x_1, x_2, x_1^2 + x_2^2)$

$x \to |\phi(x)\rangle\langle\phi(x)|$

SVM kernel

$$k(x, x') = |\langle 0| \mathcal{E}(x')^{+} \mathcal{E}(x) |0\rangle|^2$$

where $\mathcal{E}(x)|0\rangle = |\phi(x)\rangle$

qiskit.circuit.library contains ~~large~~ number of predefined & parameterized circuits

So, QSVM is SVM where the kernel function is computed on quantum hardware, offering an increase is power for that step of the calculation

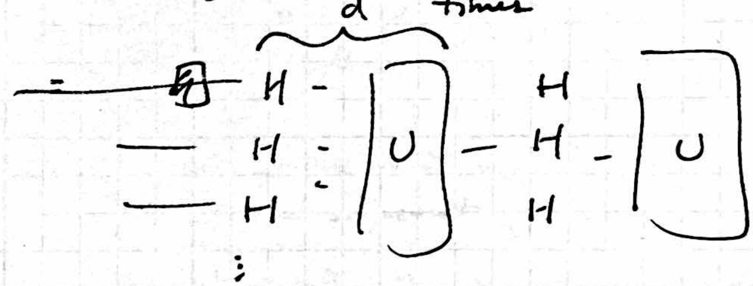
Feature map is applied to $|0\rangle^n$ initial state via some unitary operator $U_{\phi(x)}$

There are 3 feature maps ~~exist~~ available in Qiskit·ML — PauliFeatureMap, Z, ZZ

Pauli: $\mathcal{U}^{(Pauli)}_{\phi(x)} = \prod_{d} U_{\phi(x)} H^{\otimes n}$ ← H applied to all $|0\rangle$

curly $\mathcal{U}$

unitary operator of depth ~~d~~ d times



$$U_{\phi(x)} = \exp\left[i \sum_{S \subseteq [n]} \phi_S(x) \prod_{k \in S} P_i \right]$$

P_i → pauli matrix $\{I, X, Y, Z\}$

S → "describes the connectivities between different qubits or datapoints"

ex: $\phi_S : x = \begin{cases} x_i & \text{if } S = \{i\} \\ (\pi - x_i)(\pi - x_j) & \text{if } S = \{i, j\} \end{cases}$

default mapping

x_i → have be scaled to $[-1, 1]$

→ the analog values of the data are applied as rotations (about what axis ?!)

Special case of $k = 1$ & $P_0 = Z$ $\rightarrow$ Z FeatureMap

$$V = \left[\left(e^{i \sum_j \phi_{(j)}(x) Z_j} \right) H^{\otimes n} \right]^d$$

over the N ~~qub~~ qubits

Z Feature Map, however, is easy to simulate classically & does not contain quantum advantage.

When $k = 2$ $P_0 = Z$ $P_1 = ZZ$ $\rightarrow$ ZZ Feature Map

$$V = \left[e^{i \sum_{jk} \phi_{(j,k)}(x) Z_j \otimes Z_k} \, e^{i \sum_j \phi_{(j)}(x) Z_j} \, H^n \right]^d$$

adds entanglement via CNOTs between adjacent qubits

would be nice to see how this is implemented

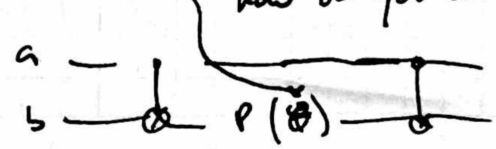
$$\phi_{x_j} = \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} \qquad \phi_{jk}(x) Z_j \otimes Z_k = Z_a \otimes Z_b$$

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\text{CNOT} = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ 0 & & 0 & 1 \\ & & 1 & 0 \end{pmatrix} \qquad \phi_{ab}(x) \begin{pmatrix} 1 & 0 & & 0 \\ 0 & -1 & & \\ & & -1 & 0 \\ 0 & & 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix}$$

$\theta \leftarrow 2(\pi - x_a)(\pi - x_b)$ $\qquad \phi$

how do you write this ~~as~~?



The feature maps appear to be fixed circuits. There are no ~~tun~~ tunable parameters, ~~exp~~ except there is the mapping function $\phi(x)$, which presumably, can be modified when instantiating an object.

In comparison, TwoLocal & NLocal circuits contain a number of tunable parameters (which I guess means you can ~~d~~ choose the parameters such that they reproduce the feature maps?)

Or... maybe using a VQE strategy, you can tune the parameters such that the feature map step is optimized for the classification task.

Transforming qubit ψ through rotations, which are reversible, encodes data into a Q circuit. The entanglement + rotations encodes new quadratic features. (~~R~~ Is this the only kind of non-linear featurile mapping that is possible?)

A Quantum Kernel is a correlation between two circuits that have ~~p~~ been parametrized using training data samples

$$K(\phi(x), \phi(x')) = |\langle 0|U^{\dagger}_{\phi(x)} U_{\phi(x')}|0\rangle^n|^2$$

$\downarrow$

real matrix of size $N \times N$ where N is number of samples

QC can be used to calculate each element of this $N \times N$ matrix (although there are only $\frac{N^2}{2} - N$ unique elements $\cancel{\&}$ $N(\frac{N}{2} - 1)$)

Each quantum circuit's size is determined by the ~~length of the input data~~ size of each vector $\vec{x}$.

It appears that the quantum kernel circuit is the application of each feature map U_ϕ, back to back... Its hard to see the cross-terms in the diagram. ~~I~~ Are there cross terms?

K matrix grows like N^2 N is size of train data
Computation of each element grows like M,
where M is size of data vector ~~$x \in$~~ $x \in R^M$

... the rest is just regular data science