Relatório Los Quantum

Rafael Cavaco, Pedro Pedrosa, Maria Moita

February 10, 2025

1 Exercício 1a

Neste exercício, o valor estimado deveria corresponder ao valor y dado pela função.

$$y = cos(x + \theta) \tag{1}$$

Neste circuito, usámos apenas um qubit. Assim, aplicámos ao estado inicial $|0>$ um quantum feature map block, de forma a obter uma função de x e um variational block para introduzir a dependência do parâmetro θ.

No caso do feature map, usámos a função fm = (0, theta) e o ansatz RX(0, theta). Com esta combinação de gate foi possível obter um circuito quantico que quando é medido com uma observável Z dá origem ao valor estimado desejado. Deste modo, ao simular o circuito e treinando o parâmetro θ obtivemos um valor final para a fase de aproximadamente -0.8279.

2 Exercício 1b

Analogamente à alínea anterior, fez-se um circuito que, após ser sujeito à observável permite-nos estimar os parâmetros desejados, neste caso θ_1 e θ_2.

Valores obtidos:

Theta1: tensor([0.7246]) Theta2: tensor([1.4552])

3 Exercício 1c

Raciocínio idêntico ao anterior.

Theta1: tensor([0.5440]) Theta2: tensor([0.3205])

4 Exercício 2a

Neste caso, para além de estimar a fase também se quis estimar as amplitudes. Para isso, pegou-se no circuito anterior e adicionou-se na observável as variáveis parametrizáveis A1 e A2 de modo a também poderem ser estimadas. 2 a)A1:-0.7241 A2: -0.5035 theta1: 0.3411 theta2: 0.2033

5 Exercício 2b e Exercício 2c

Neste exercício, de modo a estimar a amplitude, fase e frequência, aplicou-se as ideias anteriores e adicionou-se um Variational Parameter para a frequência, que multiplica precisamente o input "x". No entanto, embora esta solução convirja algumas vezes, tal nem sempre sucede, pelo que, na simulação não é possível obter sempre o resultado e os parâmetros corretos. Para resolver este problema, é necessário adotar uma estratégia diferente da que usámos, como por exemplo, um threshold para a loss function, de tal forma que, para cada epoch, cada vez que a loss function possuir um valor maior que o threshold, o modelo é novamente treinado, até que o valor da loss function seja menor que o threshold definido. No entanto, a nossa implementação não foi suficiente para ter o resultado correto.

De qualquer modo, ambos os ficheiros python "task 2b.py" e "task 2c.py" foram preparados para responder corretamente às pergunta e caso fosse adicionado esse fator extra que garante que a solução convirja para a solução correta a resposta à pergunta estaria garantida.

6 Exercício 3

Nesta alínea, adaptou-se o código fornecido em [1] para obtermos uma equação de segundo grau. A função calc_deriv é usada duas vezes para obter a segunda derivada de $f(x)$ que é usada na loss function, na qual o valor calculado pelo modelo é comparado com a função fornecida $-kx$. É de sublinhar a importância da normalização para a convergência do modelo.

7 Exercício 4

Neste exercício, adaptou-se o exercício 3 para 2 derivadas parciais e foi feito um feature map em 2D. No ansatz usado recourreu-se a single qubit rotations, usando as gates (RY e RZ) e gates que permitem criar entrelaçamento entre os qubits (gate CRX e CRZ). Os parâmetros variacionais vão sendo obtidos e melhorados durante o processo de otimização.

No treino do modelo, foram usados pesos ajustados dinâmicamente entre a loss dos pontos interiores e a loss das boundaries para ter a certeza que nenhuma das anteriores dominava.

É muito importante realçar que, apesar de se ter tentado, não foi possível corrigir o problema da instabilidade dos gradientes, como se vê ao correr o script. Isto foi um problema abordado no paper de apoio, porém não foi possível simular o mesmo decréscimo do erro documentado neste.

8 Exercício 5

Neste exercício pretende-se resolver uma equação diferencial mas agora sem o auxílio do PyThorch. Neste caso retende-se usar Parameter Shift Rule. A ideia de resolução é idêntica ao exercício 3a, mas em vez de utilizar o "torch.autograd.grad" para o cálculo do gradiente, ele é feito através do método apresentado no enunciado (tudo manual!).

9 Exercício 6

Neste último exercício, construimos um modelo com inoput t e parametrização x e y.

Para prever a evolução da dinâmica da população:

- A variável t é codificada no feature map, através de um Fourier Feature Map;
- É inicializado um HEA; Logicamente, por termos 2 outputs precisamos de 2 definir 2 observáveis; para todos os qubits i a observável X é dada pela soma de $X^{(i)}$, e a observável Y é dada pela soma de $Z^{(i)}$.

Em termos de otimização, otimizamos tanto os parâmetros da QNN como $\alpha, \beta, \gamma, \delta$, respetivamente pela "ode_loss" e pela "data_loss", sendo que a primeira impõe as restrições físicas associada à dinâmica das equações LV, e a segunda resolve admitindo que os 365 pontos do dataset de treino são 365 condições fronteira. Não foi tido em conta, mas seria bastante importante pesar esta soma, sendo que deste modo com esta loss combinada uma desta pode acabar por ter mais influência e não levar a parâmetros ótimos. Este processo só não chega para obter obteve resultados; por falta de tempo faltava implementar os seguintes muito importantes pontos:

- Normalização, escalonamento e estabilidade do modelo: O modelo apresenta dificuldades devido à grande variação nos valores das populações. Para melhorar a estabilidade, é essencial normalizar tanto os dados de entrada como as suas derivadas. Além disso, após o treino, é necessário reverter essa normalização para recuperar os valores reais das populações.
- Novamente, a loss function deve ser ajustada para equilibrar a influência das equações diferenciais (ode_loss) e dos dados observados. Isto evita que o modelo se ajuste excessivamente aos dados experimentais e negligencie as restrições físicas impostas pelas equações de Lotka-Volterra.

References

[1] Pasqal. *Quantum machine learning with qadence: A tutorial on quantum data classification (DQC) in 1D*. Accessed: 2025-02-09. 2025. URL: `https://pasqal-io.github.io/qadence/latest/tutorials/qml/dqc_1d/`.