UNIVERSIDADE DO MINHO

CIÊNCIA DE DADOS QUÂNTICA (2021/22)

---

# Relatório

## 5. Quantum Data Encoding and Variational Circuits: a performance analysis

---

João Pedro Sá Gomes - PG47339
Maria Inês Machado Correia Brioso Dias - PG47474
Ricardo da Silva Correia - PG47607

Mestrado em Engenharia Física (Física da Informação)

# 1 Introdução

Inicialmente, estabeleceu-se que o tema do nosso trabalho prático era: **5. Quantum Data Encoding and Variational Circuits: a performance analysis**. Assim, o trabalho iria consistir em fazer uma análise do impacto de diferentes data encodings na performance de um dado circuito variacional já elaborado por outra entidade. Como surgiram dificuldades em utilizar um circuito variacional já criado, que tivesse resultados satisfatórios, este trabalho envolveu mais foco na parte do circuito variacional do que era expectável (incluindo a realização de diversos testes). Assim, consideramos que, agora que está finalizado, o nosso trabalho prático corresponde a um híbrido entre o tema 5 já referido, e o tema 1, Classification with variational quantum circuits (não correspondendo inteiramente ao tema 1). Por esse motivo, acabamos por não experimentar alguns dos encodings que nos propomos a experimentar no enunciado.

Assim, este relatório começa por descrever os *datasets* que experimentamos utilizar: os dados *heart-attack* e os dados do vinho (este último é o dataset que escolhemos, para o circuito final) - secções 2 e 3. Na secção 4, são descritas as diferentes experiências realizadas a nível do circuito variacional, incluindo análises dos resultados obtidos para diversas características dos circuitos que experimentamos. A secção 5 foca-se no objetivo inicial do trabalho: a análise da performance do circuito para diferentes tipos de encoding. A penúltima secção é a Conclusão, em que resumimos os aspetos essenciais do trabalho, e a última são observações adicionais.

# 2 Dados *heart-attack*

Este dataset é proveniente de [1], um dataset presente na conta *Github* onde se encontra o código correspondente ao artigo [2], sendo o mesmo dataset que é utilizado no circuito variacional descrito nesse artigo. Este dataset contém informação acerca de pacientes, que inclui uma previsão relativamente à sua probabilidade de ter um ataque cardíaco [2]. Possui 11 colunas com os seguintes títulos [3]:
**age**: idade de uma pessoa em anos; **sex**: sexo (1=macho, 0=fêmea); **cp** : dor no peito (chest pain) (valor 1: angina típica; valor 2: angina atípica; valor 3: dor não relacionada com angina; valor 4: assintomático) ; **trestbps** : pressão sanguínea em repouso (em mm Hg) ; **chol**: colesterol total em mg/dl ; **fbs**: (açúcar no sangue em jejum > 120 mg/dl) (1= verdade; 0 = falso) ; **restecg**: resultados eletrocardiográficos em repouso (valor 0: normal; valor 1: ter anormalidade na onda ST-T (inversões da onda T e/ ou elevação ST ou depressão de mais de 0.05 mV; valor 2: mostrar hipertrofia ventricular esquerda provável ou definitiva pelos critérios de Estes) ; **thalach**: frequência cardíaca máxima atingida ; **exang**: angina induzida por exercício (1=sim; 0=não) ; **oldpeak**: depressão ST induzida por exercício, relativamente ao repouso ; **num**: previsão (1= alta probabilidade de um ataque cardíaco, 0= baixa probabilidade).

# 3 Dados do Vinho

Numa parte posterior do trabalho vai ser utilizado o dataset do vinho, no qual possui 5 colunas, são elas: **alcohol**: Percentagem de álcool do vinho; **Flavonoids**: Flavonoides; **color_intensity**: intensidade da cor do vinho; **proline**: Prolina em unidades mg/l; **target**: 1 corresponde a ser vinho tinto, 0 corresponde a ser vinho branco.

# 4 Procedimento para obter o circuito variacional:

Numa primeira fase foram implementadas funções em Python com o objetivo de tratar os dados provenientes do ficheiro "total_hearth_data.csv" (originalmente chamado "data.csv"), na qual estavam os dados originais.

Este tratamento consistiu essencialmente em passar os '?' para 'NaN', remover as features com um elevado número de NaNs e em substituir os NaNs das features restantes pela média ou pela moda(no caso das features com valores 0 ou 1, por exemplo) dos seus valores.

Depois de se tratar o ficheiro CSV original, decidiu-se misturar os datapoints e reter apenas os primeiros 80 datapoints de forma ao circuito variacional otimizar os parâmetros mais rapidamente. De seguida separou-se esses datapoints em dados de treino, dados de teste, labels de treino e labels de teste através da função *train_test_split*, onde apenas 30% de todos os datapoints foram para os dados/labels de teste.

Tendo esse procedimento realizado, decidimos aplicar o PCA aos dados de treino e teste. O PCA (Principal Component Analysis) é uma técnica de redução de dimensionalidade que pode ser utilizada para extrair informações de espaço de alta dimensão projetando-as em um subespaço de dimensão inferior, tentando preservar as partes essenciais que têm mais variação dos dados e remover as partes não essenciais com menos variação (passar as 10 features para 4 features, por exemplo). De notar que os dados de treino e teste foram sujeitos à função *normalize* e multiplicados por $\pi$ de forma a estarem contidos entre $-\pi$ e $\pi$.

O próximo passo consistiu em elaborar o modelo parametrizado e criar funções em Python capazes de realizar a otimização dos parâmetros do Ansatz.

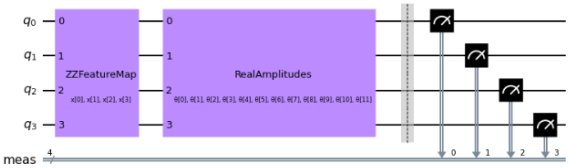


Figure 4.1: *Modelo parametrizado.*

Para elaborar o circuito variacional quântico da imagem acima, optou-se pelo IQP como encoding e o RealAmplitudes como Ansatz.

Uma vez que o nosso problema é uma classificação binária (1-alta chance de ter um ataque cardíaco; 0- baixa chance de ter um ataque cardíaco), iremos utilizar uma função de paridade como medida. A função de paridade consiste em medir todos os qubits e construir um histograma, onde de seguida se constrói uma nova distribuição de probabilidade em que se coloca a label 0 se a bitstring for par e a label 1 se a bitstring for ímpar.

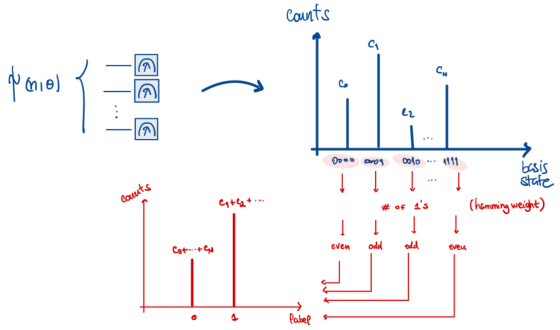


Figure 4.2: *Esquema de como funciona a função de paridade.*

Como os resultados provenientes da função de paridade são probabilidades, utilizou-se a *cross-entropy loss function*. "A cross-entropy loss function, ou log loss function, mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1. A cross-entropy loss aumenta à medida que a probabilidade prevista diverge da label real. Portanto, prever uma probabilidade de 0,012 quando a label de observação real for 1 seria mau e resultaria em um alto valor de perda. Um modelo perfeito teria uma perda logarítmica de 0." [4]

$$L(y,\hat{y}) = \sum_{i=0}^{C-1} y_i log\hat{y}_i \tag{1}$$

De seguida, criou-se a *cost function* e elaborou-se uma função que realizasse a otimização dos parâmetros do ansatz. Essa função utiliza o otimizador SPSA, onde SPSA é um método de gradiente descendente para otimizar sistemas com múltiplos parâmetros desconhecidos. Como método de otimização, é adequado para modelos populacionais de grande escala, modelagem adaptativa e otimização de simulação.[5]

**Nota:** O circuito variacional elaborado inicialmente teve como base o artigo [2] e o código correspondente (presente em [6]), em particular a parte do tratamento de dados, preparação dos dados de treino e teste e a escolha do Ansatz.

## 4.1 Testes

Tendo criado o modelo parameterizado e as funções responsáveis por otimizar esses parâmetros, correu-se o código, otimizando os parâmetros com base nos dados de treino.

Depois do código correr, registava-se o tempo que demorava a executar, fazia-se o plot da cost function e registava-se a accuracy quando se aplicava o circuito variacional quântico, já com os parâmetros otimizados, aos dados/labels de teste.

Nas figuras 4.3 e 4.4, encontra-se uma tabela onde se variavam características dos circuito variacional quântico (número de datapoints, ansatz, número de repetições do encoding/ansatz, maxiter e nº de qubits) e se registava a precisão e o tempo de execução.

Nas seguintes tabelas, ”p” significa ”points”, ”Rep.Enc.” significa nº de repetições do Encoding, ”Rep.Ans.” significa nº de repetições do Ansatz, ”m” significa ”maxiter”, e ”x Qubits” significa que foram utilizados x qubits. A percentagem é a accuracy e o ”ts” significa que o tempo de execução foi t segundos.

| | | Rep.Enc=1;Rep.Ans=1;m=300 | | | Rep.Enc.=1;Rep.Ans.=2;m=300 | | | Rep.Enc.=2;Rep.Ans=1;m=300 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 Qubits | 3 Qubits | 6 Qubits | 2 Qubits | 3 Qubits | 6 Qubits | 2 Qubits | 3 Qubits | 6 Qubits |
| RealAmplitudes | 80p | 50%;1022s | 63%;1386s | 71%;2840s | 50%;1117s | 58%;1483s | 50%;2602s | 29%;13332s | 58%;1684s | 33%;3525s |
| | 50p | 40%;664s | 80%;853s | 33%;1779s | 20%;724s | 67%;998s | 33%;1797s | 33%;733s | 73%;1089s | 33%;1867s |
| EfficientSU2 | 80p | 50%;1061s | 42%;1392s | 54%;2165s | 54%;1254s | 50%;1604s | 33%;2744s | 58%;1204s | 50%;1805s | 33%;2932s |
| | 50p | 33%;663s | 60%;900s | 47%;1521s | 40%;779s | 40%;1046s | 60%;2067s | 47%;798s | 40%;1116s | 33%;2312s |

Figure 4.3: *Tabela de testes utilizando o IQP como encoding.*

| | | Rep.Enc=3;Rep.Ans=1;m=300 | | |
|---|---|---|---|---|
| | | 2 Qubits | 3 Qubits | 6 Qubits |
| RealAmplitudes | 80p | 42%;1293s | 50%;2002s | 54%;4007s |
| | 50p | 53%;893s | 47%;1292s | 33%;2451s |
| EfficientSU2 | 80p | 58%;1260s | 54%;2026s | 50%;4013s |
| | 50p | 73%;881s | 47%;1820s | 40%;2642s |

Figure 4.4: *Tabela de testes utilizando o IQP como encoding.*

**Análise dos resultados de 4.3 e 4.4:** Nesta análise vão ser estudados o aumento do nº de pontos(de 50 para 80), o aumento do nº de features do PCA(igual ao aumento do número de qubits, pois o encoding é o IQP), aumento do nº de repetições do encoding, o aumento do nº de repetições do Ansatz e qual o ”melhor” Ansatz, tendo em conta que em cada caso, deixa-se todos os outros parâmetros constantes.

- **Aumento do nº de pontos:** Teoricamente, o aumento do nº de datapoints, em geral, devia trazer melhores resultados porque a nossa máquina fica ”melhor treinada”. Em geral, é isso que acontece, ou seja, com o aumento de 50 para 80 pontos, a accuracy melhora para 14 casos e diminui para 8 casos.

- **Aumento do nº de qubits:** Segundo as tabelas, notamos que em geral os maiores valores de accuracy estão para o de 2Qubits e 3Qubits, ou seja, para um PCA de 2 e 3 componentes, respectivamente(não esquecer que o número de qubits é igual ao número de componentes PCA). Provavelmente o PCA está a conseguir compactar bem a informação de tal forma que as redundâncias são "ignoradas" e os resultados apresentam-se melhores para datapoints com um número de features menores(2 e 3, respectivamente). Teoricamente se o PCA não estivesse a conseguir compactar tão bem a informação, era de esperar obter melhores resultados de accuracy para o caso de 6Qubits, contudo não é isso que acontece.

- **Aumento do nº de repetições do Encoding:** Relativamente aos casos de 2Qubits, em geral, o aumento do número de repetições do Encoding melhora a accuracy, no entanto, não é isso que se verifica para 3Qubits e 6Qubits, ou seja, para estes 2 últimos casos, em geral, obtemos melhores resultados para apenas uma repetição do encoding.

- **RealAmplitudes vs EfficientSU2:** Tal como podemos observar pelas tabelas mencionadas, o RealAmplitudes teve melhor accuracy que o EfficienteSU2 em 9 casos, já o EfficientSU2 teve melhores resultados em 10 casos. Contudo, podemos observar que para 80pontos, o RealAmplitudes, em geral, obtém melhores resultados, já o EfficientSU2, em geral, obtém melhores resultados para 50 pontos.

- **Aumento do nº de repetições do Ansatz:**Neste caso, apenas faz sentido comparar os blocos da tabela 4.3 com Rep.Enc=1;Rep.Ans=1;m=300 e Rep.Enc=1;Rep.Ans=2;m=300. Tendo em conta o mencionado anteriormente e observando o RealAmplitudes, o aumento das repetições do Ansatz para 2 fez com que os resultados da accuracy fossem iguais ou piores tanto para 2,3 e 6Qubits, ou seja, no caso de trabalharmos com RealAmplitudes é preferível trabalhar com apenas 1 repetição para este problema em específico. Já para o EfficientSU2, não se observa nenhum padrão notório uma vez que certas vezes a accuracy aumenta, noutras diminui. Desta forma, o aumento do número de repetições do EfficientSU2, para este problema, não traz nenhum benefício, pelo contrário, faz o tempo de execução ser maior.

- **Média das accuracies:** Realizou-se a média dos valores presentes nas tabelas 4.3 e 4.4 e o resultado foi $\approx 48\%$, ou seja, a nossa máquina apenas acerta em 48 casos de 100. Esta percentagem significa que a nossa máquina está a falhar muito e isso, por exemplo, pode ser fruto do reduzido número de pontos(50 e 80) e/ou dos dados estarem pouco correlacionados.

Este circuito variacional apresenta alguns defeitos, como o elevado tempo de execução e a alta variação da accuracy quando executada mais que uma vez. Como tal, decidimos utilizar a função VQC do Qiskit, que faz a otimização dos parâmetros do modelo parametrizado utilizando uma simples função disponibilizada pela biblioteca *qiskit_machine_learning.algorithms.classifiers*.

O tratamento de dados foi exatamente o mesmo que o anterior, bem como a criação dos dados/labels de teste/treino, aplicação do PCA e elaboração do modelo parametrizado. A única diferença é que em vez de se usar funções para a loss function, cost function, etc que nao estão pré-definidas, utilizou-se a função VQC que faz tudo isso automaticamente, desde que se coloque corretamente os parâmetros na mesma.

Para a tabela 4.1 variou-se características dos circuito variacional (ansatz, número de repetições, maxiter e nº de qubits) utilizando a função VQC e se registava a precisão e o tempo de execução.

Nas tabelas seguintes, "RealA" significa RealAmplitudes, "Eff" significa "EfficientSU2", "p" significa "points", "Rep.Enc." significa nº de repetições do Encoding, "Rep.Ans." significa nº de repetições do Ansatz, "m" significa "maxiter", e "x Qubits" significa que foram utilizados x qubits. A percentagem é a accuracy e o "ts" significa que o tempo de execução foi t segundos (estas grandezas estão arredondadas às unidades).

| | | Rep.Enc.=1; Rep.Ans.=1; m=300 | | | Rep.Enc.=2; Rep.Ans.=1; m=300 | | |
|---|---|---|---|---|---|---|---|
| | | 2 Qubits | 3 Qubits | 6 Qubits | 2 Qubits | 3 Qubits | 6 Qubits |
| RealA | 80 p | 46% ; 166s | 67% ; 188s | 58% ; 374s | 38% ; 176s | 63% ; 224s | 33% ; 498s |
| | 50 p | 47% ; 110s | 40% ; 140s | 33% ; 592s | 33% ; 127s | 73% ; 152s | 27% ; 316s |
| Eff | 80 p | 50% ; 170s | 63% ; 217s | 63% ; 710s | 33% ; 189s | 29% ; 245s | 38% ; 490s |
| | 50 p | 47% ; 116s | 53% ; 144s | 67% ; 264s | 33% ; 126s | 60% ; 162s | 60% ; 300s |

Table 4.1: Tabela de testes utilizando o IQP como encoding

Na segunda tabela (4.2), a sigla *PE* significa "Problemas com Estabilização do gráfico da cost function".

| | | Rep.Enc.=1; Rep.Ans.=2; m=300 | | | Rep.Enc.=3; Rep.Ans.=1; m=300 | | |
|---|---|---|---|---|---|---|---|
| | | 2 Qubits | 3 Qubits | 6 Qubits | 2 Qubits | 3 Qubits | 6 Qubits |
| RealA | 80 p | 46% ; 163s | 54% ; 206s | 58% ; 386s | 33% ; 183s | 54% ; 248s | 58% ; 581s |
| | 50 p | 47% ; 112s | 60% ; 142s | 47% ; 262s | 47% ; 130s | 27% ; 174s (PE) | 53% ; 389s |
| Eff | 80 p | 54% ; 182s | 58% ; 232s | 50% ; 434s | 54% ; 211s | 54% ; 288s | 46% ; 629s |
| | 50 p | 40% ; 120s | 20% ; 154s (PE) | 47% ; 300s (PE) | 47% ; 133s | 20% ; 184s | 60% ; 382s |

Table 4.2: Tabela de testes utilizando o IQP como encoding

No caso da segunda tabela, para os casos em que se considerou que havia problemas na estabilização (PE) do gráfico da cost function com maxiter=300, aumentou-se o maxiter para 350. Seguem-se os resultados para esses casos, para os quais os gráficos estabilizaram aumentando o maxiter (novamente, a percentagem é a accuracy e o "ts" significa que o tempo de execução foi t segundos, estando ambas as grandezas arredondadas às unidades):

- EfficientSU2; 50 pontos; Rep.Enc.=1; Rep.Ans.=2; 3 qubits; maxiter=350: 40% ; 179s.
- EfficientSU2; 50 pontos; Rep.Enc.=1; RepAns=2; 6 qubits; maxiter=350: 47% ; 345s.
- RealAmplitudes; 50 pontos; Rep.Enc.=3; Rep.Ans.=1 ; 3 qubits; maxiter=350: 40% ; 198s.

Ao executar os testes das duas tabelas anteriores, notou-se que, após executar o treino, em alguns casos, havia variações na accuracy que se obtinha quando se calculava este valor várias vezes seguidas para um dado treino. Chegou a ser detetada uma variação de 20% entre valores de accuracy, para um dado treino, calculando 5 vezes a accuracy. Por outro lado, também houve casos em que, calculando a accuracy 5 vezes seguidas para um dado treino, se obteve sempre o mesmo resultado (arredondado às unidades).

**Análise dos resultados** Esta análise de resultados é relativa às duas tabelas anteriores e segue os mesmos moldes que a que foi efetuada para as tabelas 4.3 e 4.4.

- **Aumento do nº de pontos:** Ao contrário do que era expectável, não foi visível uma vantagem clara relativamente a um número de pontos maior em termos de accuracy, mas foi visível uma tendência em termos de redução do tempo de execução para um número de pontos menor, o que faz sentido, já que assim o circuito processa menos pontos (sendo que cada ponto tem de ser individualmente codificado).
- **Aumento do nº de qubits:** O número de qubits não aparenta ter uma influência clara nos resultados da accuracy, mas verificou-se que para 6 qubits o tempo de execução tende a ser superior, embora apresentando variações consideráveis. Para esta observação, não foi encontrada nenhuma explicação.
- **Aumento do nº de repetições do Encoding:** Neste caso, não foi encontrado nenhum padrão relativamente a uma variação da accuracy ou do tempo de execução provocado por um maior número de repetições do Encoding.

- **Aumento do nº de repetições do Ansatz:** Neste caso, também não foi encontrado nenhum padrão relativamente a uma variação da accuracy ou do tempo de execução provocado por um maior número de repetições do Ansatz.
- **Tipo de ansatz:** Neste caso, tammbém não foi encontrado nenhum padrão relativamente a um aumento da accuracy ou diminuição do tempo de execução provocado pelo uso de um dos Ansatz. De entre os 6 maiores valores obtidos para a accuracy nestas tabelas, 3 correspondem ao RealAmplitudes e os restantes ao EfficientSU2.
- **Média das accuracys:** Realizou-se a média dos valores de accuracy presentes nestas tabelas, incluindo também os valores da accuracy relativos aos casos em que se usou maxiter=350, e o resultado foi $\approx 47.35\%$, ou seja, a nossa máquina apenas acerta em 47.35 casos de 100. Este resultado indica que o circuito variacional utilizado não funciona como seria expectável.

**Nota:** O facto de apenas ter sido registada uma tentativa para cada caso faz com que os dados obtidos para estas duas tabelas possam não ser completamente elucidativos, já que para dois treinos diferentes a accuracy obtida pode variar. Por isso, tendo tempo para tal, o ideal seria executar várias tentativas e fazer uma média das accuracies e dos tempos obtidos.

Utilizando o circuito variacional anterior, decidiu-se mudar o dataset e elaborou-se também uma tabela. Este dataset é o dataset dos vinhos, explicado no capítulo anterior.

Na tabela/figura 4.5 fomos variando as características dos circuito variacional quântico (número de datapoints, ansatz, número de repetições do encoding/ansatz e nº de qubits) e registava-se a precisão e o tempo de execução.

| | | Rep.Encoding=1;Rep.Ansatz=1 | | Rep.Encoding=2;Rep.Ansatz=1 | | Rep.Encoding=1;Rep.Ansatz=2 | | Rep.Encoding=3;Rep.Ansatz=1 | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2 Qubits | 3 Qubits | 2 Qubits | 3 Qubits | 2 Qubits | 3 Qubits | 2 Qubits | 3 Qubits |
| RealAmplitudes | 80 points | 66.6% ; 246s | 41.6% ; 2124s | 66.6% ; 275s | 62.5% ; 894s | 58.3% ; 272s | 58.3% ; 876s | 70.8% ; 316s | 45.8% ; 1236s |
| | 50 points | 60% ; 150s | 60% ; 510s | 66.6% ; 153s | 53.3% ; 600s | 60% ; 156s | 66.6% ; 472s | 33.3% ; 189s | 60% ; 1074s |
| EfficientSU2 | 80 points | 58.3% ; 378s | 50% ; 866s | 79.1% ; 349s | 58.3% ; 845s | 75% ; 332s | 66.6% ; 1740s | 62.5% ; 655s | 50% ; 1035s |
| | 50 points | 53.3% ; 209s | 53% ; 532s | 60% ; 400 s | 46.6% ; 638s | 53.3% ; 184s | 73.3% ; 1634s | 33.3 ; 211s | 40% ; 648s |

Figure 4.5: *Tabela de testes, com maxiter a 300 e utilizando o IQP como encoding.*

**Análise dos resultados de 4.5:** Esta análise de resultados segue os mesmos moldes que a que foi efetuada para as tabelas 4.1 e 4.2.

- **Aumento do nº de pontos:** Teoricamente, o aumento do nº de datapoints, em geral, devia trazer melhores resultados porque a nossa máquina fica ”melhor treinada”. Em geral, é isso que acontece, ou seja, com o aumento de 50 para 80 pontos, a accuracy melhora para 9 casos e diminui para 6 casos.
- **Aumento do nº de qubits:** Segundo a tabela notamos que em geral os maiores valores de accuracy estão para o de 2 Qubits, ou seja, para um PCA de 2 componentes. O porquê de tal acontecer é a mesma explicação que se fez na análise de resultados para a tabela 4.3 e 4.4.
- **Aumento do nº de repetições do Encoding:** Em geral, o aumento do número de repetições do Encoding melhora a accuracy para os casos em que temos 80 datapoints, no entanto, não é isso que se verifica quando usamos 50 datapoints.
- **Aumento do nº de repetições do Ansatz:**Neste caso, apenas faz sentido comparar os blocos da tabela 4.5 com Rep.Encoding=1;Rep.Ansatz=1 e Rep.Encoding=1;Rep.Ansatz=2. Comparando os valores desses blocos, verificou-se que para quase todos os casos a accuracy aumentou.

- **Tipo de ansatz:** Analisando a tabela, veirificou-se que o RealAmplitudes obteve em grande parte das vezes uma maior percentagem que o EfficientSU2, porém o EfficientSU2 obteve grande parte dos valores de percentagem mais elevados presentes na tabela (79.1%, 75% e 73.3%)
- **Média das accuracys:** Realizou-se a média dos valores de accuracy presentes na tabela 4.5 e o resultado foi ≈ 57.6%, ou seja, a nossa máquina apenas acerta em 57.6 casos de 100. Comparando com os resultados das outras tabelas, conclui-se que a nível de precisão obtivemos melhores resultados com o dataset dos vinhos.

Analisando todas as tabelas, verificamos que se obteve mais sucesso com o uso do VQC e no dataset dos vinhos.

Olhando para a tabela 4.5 podemos verificar que o melhor que se conseguiu obter foi uma accuracy de 79.1% com um tempo de execução de 349 segundos, sendo que se utilizou como Ansatz o EfficientSU2 com uma repetição, um encoding IQP de duas repetições, dois Qubits no PCA e 80 datapoints no total.

# 5 Variação do tipo de encoding e análise dos resultados

**Nota:** Os encodings que irão ser abordados são o IQP, o Amplitude Encoding e o Angle Encoding. Como já foram feitas descrições dos diferentes tipos de encoding no enunciado do trabalho, e por uma questão de gestão de espaço, não serão colocadas neste relatório.

## 5.1 IQP

### 5.1.1 Testes

Tal como vimos anteriormente, o melhor resultado da accuracy foi obtido para o caso em que o número de repetições do IQP é 2. Depois disso, mantivemos todos os parâmetros anteriores e voltamos a correr o circuito varicional variando o maxiter entre 300 e 400. Os resultados obtidos, encontram-se na figura 5.1.

| | tentativa1 | tentativa2 | tentativa3 |
|---|---|---|---|
| maxiter=300 | 71% ; 232s | 66% ; 236s | 71% ; 234s |
| maxiter=400 | 71% ; 311s | 66% ; 303s | 66% ; 301s |

Figure 5.1: *Resultados da accuracy e do tempo respectivo para um maxiter de 300 e 400*

Relativamente a esta última tabela, a média para maxiter=300 é 69%, e para maxiter=400 é 67% (com arredondamento às unidades). Tal como podemos observar na figura 5.2 a profundidade do IQP com 2 repetições é 10 e o número de gates é 14, para 2 features.

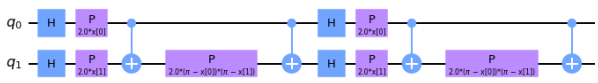


Figure 5.2: *IQP para 2 repetições e para 2 features.*

## 5.2 Amplitude Encoding

### 5.2.1 Testes

Neste caso em específico, como o nosso *X_train* (depois do PCA) vai ser constituído por datapoints com 2 *features*, o número de qubits necessários vai ser apenas 1. O nosso circuito constituído pelo Amplitude

Encoding mais o EfficientSU2 apresenta-se na figura 5.3. De notar que o Amplitude Enconding é realizado tirando partido do RawFeatureVector.


Figure 5.3: *Circuito variacional*

Depois correu-se o circuito varicional com os parâmetros 80 pontos, 2 features PCA (para o amplitude encoding vamos ter apenas 1 qubit) e 1 repetição do EfficientSU2 fixos e variando, apenas, o *maxiter* entre 300 e 400, sendo que o maxiter corresponde ao número total de vezes que o algoritmo otimiza os parâmetros do circuito variacional (é o número de iterações do algoritmo). Além disso, como estamos a tratar do Amplitude Encoding não faz sentido repeti-lo tal como fazíamos para o IQP e dessa forma, o número de repetições do Amplitude encoding vai ser 1.

**Observação:**Antes de correr o circuito, teve-se o cuidado de normalizar os datapoints.

Seguidamente, obtiveram-se os seguintes resultados da accuracy e do tempo para maxiter=300 e 400. Estes resultados apresentam-se na figura 5.4.

| | tentativa1 | tentativa2 | tentativa3 | tentativa4 |
|---|---|---|---|---|
| maxiter=300 | 71% ; 393s | 75% ; 371s | 75% ; 369s | 71% ; 517s |
| maxiter=400 | 75% ; 592s | 71% ; 563s | 75% ; 554s | 71% ; 638s |

Figure 5.4: *Resultados da accuracy e do tempo respectivo para um maxiter de 300 e 400*

Relativamente a esta última tabela, a média para maxiter=300 é 73%, e para maxiter=400 é 73% (com arredondamento às unidades). Além de tudo isto, também é importante determinar a profundidade e o número de gates utilizadas pelo nosso encoding. Segundo [7] e [8], para um estado quântico geral, deve-se aplicar 2 cascatas no qual a primeira cascata utiliza rotações $R_z$ e a segunda cascata rotações $R_y$. Como os nossos datapoints podem ter valores negativos não podemos desprezar a primeira cascata, logo para apenas um qubit o Amplitude Encoding vai ser apenas constituído por uma gate $R_z$ e uma $R_y$. Assim, a profundidade do encoding em questão é 2 e o número de gates também é 2.

## 5.3 Angle Encoding

### 5.3.1 Testes

A tabela que se segue mostra os resultados obtidos para alguns testes realizados, em que o circuito variacional tinha as mesmas características que as do circuito utilizado na subsecção 5.2, apenas variando o maxiter entre 300 e 400, e utilizando o Angle Encoding em vez do Amplitude Encoding (tal como no caso do Amplitude Encoding, este apenas pode ter 1 repetição). Na seguinte tabela, a percentagem é a accuracy e o "ts" significa que o tempo de execução foi t segundos (as accuracies e os tempos de execução encontram-se arredondados às unidades).

| | tentativa1 | tentativa2 | tentativa3 |
|---|---|---|---|
| maxiter=300 | 79% ; 163s | 75% ; 165s | 79% ; 166s |
| maxiter=400 | 75% ; 554s | 79% ; 543s | 75% ; 565s |

Table 5.1: *Resultados da accuracy e do tempo respectivo para um maxiter de 300 e 400*

Relativamente a esta última tabela, a média para maxiter=300 é 78%, e para maxiter=400 é 76% (com arredondamento às unidades). Um exemplo de um circuito constituído pelo Angle Encoding (utilizando

a matriz de Pauli $\sigma_y$ como matriz de rotação) e pelo EfficientSU2, com as medições, para 2 features, apresenta-se na figura 5.5. A profundidade do encoding em questão é sempre 1, e o número de gates que utiliza é igual ao número de qubits.

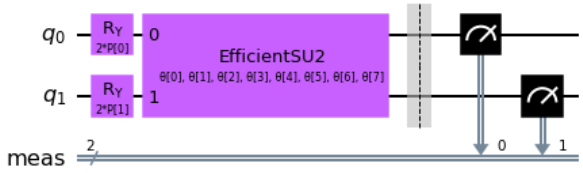


Figure 5.5: *Circuito variacional com Angle Encoding (com $\sigma_y$)*

## 5.4 Discussão sobre os diferentes encodings

Para datapoints com N features, o Angle Encoding necessita de N qubits (neste caso, 2 qubits), tal como o IQP, enquanto o Amplitude Encoding necessita apenas de $log_2(N)$ qubits (neste caso, 1 qubit). Assim, este último tipo de encoding permite uma representação exponencialmente compacta, o que atualmente é de extrema importância uma vez que temos acesso a poucos qubits nos computadores quânticos atuais.

A profundidade do IQP para 1 repetição é 5, e para 2 repetições é 10. A profundidade do Angle Encoding é sempre de 1, já a do Amplitude Encoding é 2.

Em termos do número de gates, neste caso em que apenas consideramos 2 features, o Amplitude Encoding apenas utiliza 2 gates, tal como o Angle Encoding, e o IQP, para 1 repetição, utiliza 7 gates, e para 2 repetições utiliza 14. Assim, o IQP é o que, neste caso, utiliza mais gates. De notar que, se o número de features aumentar, o número de gates que o Amplitude Encoding irá utilizar começa a ser superior ao número de gates que o Angle Encoding utiliza (este último utiliza sempre um número de gates igual ao número de qubits). Um número de gates maior implica uma maior quantidade de erros associados, pelo que o Amplitude Encoding poderia apresentar uma desvantagem nesse aspeto, em relação ao Angle Encoding. Assim, no caso considerado (com 2 features), o IQP é o menos vantajoso em termos do número de gates e profundidade.

De notar que pretendemos sempre um circuito com um número de gates reduzido, assim como a sua profundidade. Isto porque quanto menores forem estes, menor vai ser o tempo de execução do programa e consequentemente menos propensos ficamos à obtenção de erros provenientes da decoerência. Assim, teoricamente, o que devia dar melhores resultados devia ser o Angle Encoding.

Relativamente à performance do circuito com os diferentes valores de maxiter (diferentes números de iterações do algoritmo), verifica-se que um aumento deste valor faz aumentar o tempo de execução, mas pelos resultados obtidos não aparenta melhorar a accuracy, para o Amplitude, para o Angle Encoding e também para o IQP. A nossa hipótese para explicar isso é que com 300 iterações do algoritmo, já se atinge um treino satisfatório, ou até os parâmetros mais otimizados, pelo que mais iterações não melhoram os parâmetros, nem os resultados. O aumento do tempo de execução faz sentido, já que um maior maxiter implica um maior número de iterações do algoritmo.

Comparando a performance do Angle Encoding e do Amplitude Encoding, a partir das tabelas 5.4 e 5.1, verifica-se que a accuracy obtida no caso do Angle foi ligeiramente superior, tanto para maxiter=300 como para maxiter=400. Além disso, para maxiter=300, o tempo de execução do circuito com Angle Encoding foi notoriamente menor do que no do Amplitude Encoding. No entanto, não faz ”muito” sentido comparar estes tempos de execução, já que o circuito com Amplitude Encoding foi simulado numa máquina virtual. Contudo, para 2 features, como o Amplitude Encoding, possui igual número de gates ao Angle, ambos utilizam as gates $R_y$(apenas diferem numa gate que é a $R_z$) e como a profundidade de ambos é baixa, era de esperar que os tempos de execução fossem mais ou menos parecidos, no entanto não é isso que acontece para maxiter igual a 300 e o tempo do Amplitude, em relação ao Angle, é mais que o dobro. Já para o maxiter igual a 400 os tempos de execução estão mais de encontro com o previsto.

A partir dos dados da tabela 5.1, comparando com a tabela 5.4, verifica-se que a performance do Amplitude Encoding em termos de accuracy é ligeiramente melhor que a do IQP. Além disso, como

ambas as tabelas foram preenchidas com resultados obtidos na mesma máquina virtual, pode-se concluir que o tempo de execução do Amplitude Encoding foi superior, nos testes correspondentes. Para o Amplitude Encoding como a profundidade e o número de gates é menor(considerando que todas as gates do Amplitude Enconding e IQP introduzem a mesma percentagem de erro), era de esperar que se obtivesse melhores resultados dado o que foi mencionado anteriormente sobre o tempo de decoerência. Relativamente aos tempos de execução, nós vemos que o IQP para 2 repetições possui 14 gates entre as quais Hadamard, P, CNOT, já o Amplitude encoding segundo [8] apenas vai ser constituído por 2 gates $R_z$ e $R_y$(porque apenas temos 1 qubit). Se todas as gates tivessem o mesmo tempo de execução era de esperar que o tempo de execução do Amplitude Encoding fosse menor. Contudo não é isso que acontece, logo provavelmente o tempo de execução das gates $R_z$ e $R_y$ é superior às gates do IQP.

# 6 Conclusão

Através deste trabalho prático, pudemos verificar que nem sempre os circuitos variacionais têm o comportamento esperado, sendo que os dados que estes processam podem ter influência no seu desempenho, e também pudemos concluir que, em certos casos, a variação de certas características (tal como o número de pontos utilizado) tem influência nesse desempenho.

Foi possível concluir também que a utilização de diferentes tipo de Encoding tem influência na accuracy e no tempo de execução obtidos, sendo que em termos de accuracy foi o Angle Encoding que teve tendência a ter um melhor resultado, embora o valor máximo atingido por este encoding tenha sido também atingido pelo IQP.

# 7 Observações

As figuras 4.3, 4.4, 5.1 e 5.4 foram realizadas através de uma máquina virtual com características, **Sistema Operativo**: Ubuntu(64-bit) ; **Memória base**: 4096MB ; **Processadores**: 4 ; **Boot Order**: Floppy Optical, Hard Disk ; **Aceleração**: VT-x/AMD-V, Nested Paging, KVM Paravirtualization.

# References


[1] b1ackd0t (username). data.csv. https://github.com/0x6f736f646f/variational-quantum-classifier-on-heartattack/blob/main/Data/Raw/data.csv. [Online]. 2020.

[2] Rodney Osodo. Building a Quantum Variational Classifier Using Real-World Data. https://medium.com/qiskit/building-a-quantum-variational-classifier-using-real-world-data-809c59eb17c2. [Online]. 2021.

[3] b1ackd0t (username). My Quantum Open Source Foundation Project. https://github.com/0x6f736f646f/variational-quantum-classifier-on-heartattack/blob/main/Notes/1EDA.md. [Online]. 2021.

[4] ML Glossary. Loss functions. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. [Online]. 2021.

[5] Qiskit. SPSA. https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SPSA.html. [Online]. 2022.

[6] b1ackd0t (username). variational-quantum-classifier-on-heartattack. https://github.com/0x6f736f646f/variational-quantum-classifier-on-heartattack. [Online]. 2021.

[7] M. Schuld and F. Petruccione. Machine Learning with Quantum Computers. Quantum Science and Technology. Springer International Publishing, 2021. ISBN: 9783030830984. URL: https://books.google.pt/books?id=-N5IEAAAQBAJ.

[8] Mikko M. et al. Transformation of quantum states using uniformly controlled rotations. https://arxiv.org/pdf/quant-ph/0407010.pdf. [Online]. 2004.