Université de Montréal

LES CIRCUITS QUANTIQUES PARAMÉTRÉS UNIVERSELS COMME MODÈLES D'APPRENTISSAGE AUTOMATIQUE

par

Andrew Robert Williams

Département d'informatique

Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

Septembre 2021

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

LES CIRCUITS QUANTIQUES PARAMÉTRÉS UNIVERSELS COMME MODÈLES D'APPRENTISSAGE AUTOMATIQUE

présenté par

Andrew Robert Williams

a été évalué par un jury composé des personnes suivantes :

Pierre McKenzie

(président-rapporteur)

Gilles Brassard

(directeur de recherche)

Alain Tapp

(codirecteur)

Frédéric Dupuis

(membre du jury)

Résumé

L'informatique quantique exploite les phénomènes de la théorie quantique pour le traitement de l'information, tandis que l'apprentissage automatique s'intéresse aux algorithmes qui peuvent s'améliorer en fonction des expériences passées. L'informatique quantique a produit des algorithmes qui dépassent de loin les capacités des ordinateurs classiques que nous utilisons tous les jours. Cependant, l'identification de nouveaux algorithmes quantiques fut moins prolifique que dans le cas classique. Ces dernières années, on a cherché à combiner l'informatique quantique et l'apprentissage automatique. Le cadre de l'apprentissage automatique a servi à apprendre les paramètres de circuits quantiques paramétrés dans l'espoir d'apprendre à résoudre des problèmes où les phénomènes quantiques peuvent aider grâce au traitement de l'information quantique.

L'objectif principal de ce mémoire est de pousser plus loin cette idée d'apprentissage de circuits quantiques et de fonder solidement ses capacités en développant une architecture universelle de circuit quantique paramétré. La première contribution est une évaluation d'algorithmes d'optimisation itératifs actuels pour les circuits quantiques paramétrés en tant que modèles d'apprentissage automatique, ainsi que la présentation d'un algorithme d'optimisation itératif simple, mais robuste. La deuxième contribution est une architecture de circuit quantique dans laquelle une famille de petits circuits avec des connexions arbitraires peut être intégrée.

Mots-clés : Circuits quantiques paramétrés, apprentissage automatique quantique, informatique quantique, apprentissage automatique.

Abstract

Quantum information processing leverages the phenomena of quantum theory for information processing, while machine learning concerns itself with algorithms that can improve based on past experiences. Quantum information processing has produced algorithms that go far past the capabilities of the classical computers we use every day. However, the identification of new quantum algorithms has been much slower than during the early days of classical computing. In recent years, there has been a push to combine quantum information processing and machine learning. The framework of machine learning has been used to learn quantum circuits in the hopes of learning to solve problems where quantum phenomena can help through the use of quantum information processing.

The main goal of this thesis is to further push this idea of learning quantum circuits and to solidly ground its capabilities by developing a learnable parametrized universal quantum circuit. The first contribution is an assessment of current optimization methods for parametrized quantum circuits as machine learning models. The second contribution is a quantum circuit architecture in which a family of smaller circuits with arbitrary connections can be embedded.

Keywords: Parametrized Quantum Circuits, Quantum Machine Learning, Quantum Information Processing, Machine Learning

Table des matières

Liste des tableaux

Table des figures

Liste des sigles et des abréviations

Acronymes et sigles

CNOT	Porte quantique de négation contrôlée
HHL	Algorithme de Harrow, Hassidim et Lloyd [**40**]
NISQ	(*Noisy Intermediate-Scale Quantum* [**71**]) description des appareils quantiques limités qui sont présentement disponibles
VQE	*Variational Quantum Eigensolver* [**70**]

Apprentissage automatique

a	Fonction d'activation
b	Biais d'un neurone dans un réseau de neurone
d	Un signe : $d \in \{-1, +1\}$
D	Distribution génératrice des exemples
f^*	Fonction que l'apprentissage supervisé essaie de trouver
h	hypothèse
H	Classe d'hypothèses
$\hat{h}_S$	hypothèse qui minimise le risque empirique sur l'ensemble de données S
l	Largeur d'un réseau de neurones

ℓ	Fonction de perte
p	Profondeur d'un réseau de neurones
$R[h]$	Vrai risque de l'hypothèse h
$\hat{R}_S[h]$	Risque empirique de l'hypothèse h sur l'ensemble de données S
S	Ensemble de données constitué d'exemples
S_{eval}	Ensemble de données d'évaluation
$\mathbf{w}$	Vecteur de poids d'un neurone dans un réseau de neurones
$(\mathbf{x},y)$	Exemple constitué d'un vecteur de caractéristiques $\mathbf{x}$ et d'une étiquette y
X	Espace des caractéristiques
$\hat{y}$	Prédiction d'un modèle d'apprentissage automatique
Y	Espace des étiquettes
λ	Pas dans un algorithme d'optimisation itératif
τ	Un nombre de répétitions
$\boldsymbol{\theta}$	Un vecteur de paramètres
$\boldsymbol{\theta}^*$	Valeurs des paramètres qui correspondent à une hypothèse qui minimise le risque empirique

Informatique quantique

$\mathbf{A}$	matrice dans un système d'équations linéaires $\mathbf{Ax} = \mathbf{b}$
C	Circuit quantique
C_G	Circuit quantique constitué de portes de l'ensemble universel G

$C_{\boldsymbol{\theta}}$	circuit quantique paramétré par un vecteur de paramètres $\boldsymbol{\theta}$
e	Fonction d'encodage quantique
$F_{l,p}$	Famille de circuits quantiques d'arité 2, de largeur l et de profondeur p
G	Ensemble universel de portes quantiques
$\mathcal{H}$	Espace d'Hilbert
$\mathbf{H}$	Matrice hermitienne
$\boldsymbol{H}$	Vecteur de matrices hermitiennes
k	Arité d'une porte logique ou d'un circuit quantique
l	Largeur d'un circuit quantique
p	Profondeur d'un circuit quantique
P	Famille d'architectures quantiques universelles par permutation
$P_{l,p}$	Architecture quantique universelle par permutation pour la famille $F_{l,p}$
R_y, R_z	Portes quantiques de rotation à un paramètre
t	Taille d'un circuit quantique
U	Porte quantique
$U^\dagger$	Matrice U transposée et conjuguée
$\widetilde{U}$	Approximation de U
$\lvert\mathbf{b}\rangle$	État quantique $\lvert\mathbf{b}\rangle = \sum_{i=1}^{n} b_i \lvert i\rangle$ qui encode le vecteur $\mathbf{b}$ d'un système d'équations linéaires $\mathbf{Ax} = \mathbf{b}$
$\lvert x\rangle$	Sortie d'une fonction d'encodage quantique
$\lvert\mathbf{x}\rangle$	Sortie de l'algorithme HHL pour résoudre un système d'équations linéaires $\mathbf{Ax} = \mathbf{b}$

$\|\psi\rangle$, $\|\phi\rangle$	Un état quantique arbitraire sur un qubit
$\|\Psi\rangle$	Un état quantique sur un registre de qubits
$\langle\|\Psi\rangle\rangle$	espérance d'une mesure dans la base de calcul appliquée à un état quantique $\|\Psi\rangle$
	Mesure dans un circuit quantique

Remerciements

Je tiens à exprimer ma profonde reconnaissance à mes deux directeurs, Gilles Brassard et Alain Tapp, pour leur soutien extraordinaire durant mes études, leur patience sans fin et leur rôle déterminant dans le développement de mon parcours de recherche et de ce mémoire. Le simple fait de les côtoyer a amélioré ma réflexion et ma capacité d'apprentissage, et leur complémentarité inspirante témoigne de leur collaboration de longue date. J'aimerais souligner leur acceptation et leur encouragement de mes champs d'intérêt interdisciplinaires, leur patience à mon égard et, particulièrement, leur bonté franche.

Je voue aussi une immense gratitude à Irina Rish pour son appui inestimable et dont la volonté de savoir est une source d'inspiration.

J'aimerais remercier les membres du Laboratoire d'informatique théorique et quantique (LITQ), notamment Aldo Lamarre, pour leur camaraderie, leur soutien et leurs réflexions. Je suis extrêmement reconnaissant à Prateek Gupta pour sa camaraderie, sa patience hors pair et sa confiance en mon travail. J'ai également eu le grand plaisir de collaborer avec Simon Verret, une lumière dont l'expérience, la créativité et les conseils m'ont guidé durant des moments exigeants.

J'éprouve une profonde gratitude envers Robert Gérin-Lajoie, dont l'appui m'est précieux et dont la curiosité intellectuelle est un catalyseur exceptionnel.

Ces remerciements seraient incomplets si je n'affirmais pas ma sincère gratitude envers ma mère pour son assistance incomparable, mon père pour son écoute précieuse et mon frère pour son soutien moral inébranlable.

En fin de compte, je suis profondément redevable à mes amis, sans lesquels je ne me rendrais pas très loin. Je vois des lendemains lumineux quand je pense que vous y serez.

Chapitre 1

Introduction

L'*informatique quantique* [**63**] est un champ interdisciplinaire qui étudie le traitement de l'information avec des phénomènes de la théorie quantique. L'*apprentissage automatique* [**58**] porte sur les algorithmes qui améliorent leur performance à une tâche en accumulant de l'expérience.

L'informatique quantique a su accomplir des exploits qui dépassent les possibilités du traitement de l'information classique [**81, 36, 13, 10**]. Cependant, malgré des débuts fructueux, la majorité des algorithmes quantiques qui surpassent leurs équivalents classiques utilisent les trois mêmes sous-routines [**82**].

L'apprentissage profond est un sous-champ de l'apprentissage automatique qui a récemment attiré beaucoup d'attention médiatique en résolvant des tâches pour lesquelles il est difficile ou irréalisable de développer des algorithmes traditionnels écrits à la main. Ces tâches incluent la vision par ordinateur, la reconnaissance vocale et le traitement automatique du langage naturel [**32**].

L'informatique quantique et l'apprentissage automatique ont en commun de s'écarter de ce qu'ils considèrent tous deux comme des cadres «conventionnels» de traitement de l'information, dans le but de dépasser les limites de ces cadres. On peut alors se demander s'il est possible d'unir l'informatique quantique et l'apprentissage automatique pour aller au-delà des deux cadres simultanément. Nous envisageons d'utiliser le cadre de l'apprentissage automatique pour découvrir de nouveaux algorithmes d'informatique quantique : on peut voir un circuit quantique paramétré comme un modèle d'apprentissage automatique dont on peut apprendre les paramètres. Le champ qui unit l'informatique quantique et l'apprentissage automatique est connu sous plusieurs noms, mais nous privilégions l'*apprentissage automatique quantique*.

Le but principal de ce mémoire est de présenter une famille d'architectures universelles de circuits quantiques paramétrés et de montrer comment on peut optimiser les paramètres d'un circuit quantique paramétré avec un algorithme d'optimisation itératif simple, mais robuste. Cet algorithme requiert l'estimation d'une espérance de mesure, tourne sur le même circuit quantique paramétré qu'on souhaite optimiser, et le nombre d'opérations est quadratique dans le nombre de paramètres.

Dans le **chapitre 2**, les circuits quantiques sont présentés. Les bases de l'informatique quantique sont suivies d'une présentation des circuits quantiques avec l'accent mis sur l'architecture d'un circuit quantique, c'est-à-dire le positionnement de ses portes. Ensuite, nous abordons les ensembles universels de portes quantiques, les architectures quantiques universelles et les contraintes que nous imposons sur les ordinateurs quantiques que nous considérons.

Le **chapitre 3** présente l'apprentissage supervisé et les réseaux de neurones. Une attention particulière est portée à l'optimisation des réseaux de neurones par descente de gradient,

ainsi que le nombre d'opérations pour optimiser les paramètres d'un réseau avec la descente de gradient par rétropropagation.

Le **chapitre 4** ancre l'apprentissage automatique quantique avec un survol bref du domaine avant de rentrer dans les détails des circuits quantiques paramétrés en tant que modèles d'apprentissage automatique. Spécifiquement, plusieurs algorithmes d'optimisation de circuits quantiques paramétrés sont présentés dans une perspective contrastée à la descente de gradient par rétropropagation dans les réseaux de neurones. À la fin du chapitre, une approche simple et robuste à l'optimisation des paramètres des circuits quantiques paramétrés est présentée. Cette approche est basée sur l'escalade (le *hill climbing*), un algorithme d'optimisation qui ne nécessite pas le calcul de dérivées partielles.

Le **chapitre 5** développe une famille d'architectures quantiques universelles par permutation. Un circuit quantique arbitraire d'arité 2, de profondeur p et de largeur l peut être plongé dans un circuit quantique paramétré avec une architecture universelle par permutation en instanciant les paramètres du circuit aux valeurs appropriées. Un circuit quantique paramétré avec l'architecture universelle par permutation de Beneš a une profondeur $p' = 2p \lg(l)$ et une largeur $l' = l$. L'utilisation de ce circuit en tant que modèle d'apprentissage automatique permet de garantir que la classe d'hypothèses du modèle d'apprentissage inclut tous les circuits quantiques d'arité 2, de largeur l et de profondeur p, peu importe leur architecture.

Le **chapitre 6** résume les différents sujets présentés tout en les contrastant, puis il clôt le mémoire avec une exploration de directions possibles de recherche future.

Chapitre 2

Les circuits quantiques

2.1 Les bases de l'information quantique

2.1.1 Les qubits

Définition 1 (Qubit)**.** Le *qubit*, par analogie au bit classique, est l'unité de base de l'informatique quantique. Tandis qu'un bit classique est soit dans l'état 0 ou 1, l'état d'un qubit correspond à un vecteur unitaire dans un espace d'Hilbert à deux dimensions $\mathcal{H}^2$. On utilise la notation de Dirac, *ket* $|\rangle$ et *bra*$\langle|$, pour représenter l'état d'un qubit.

Un état arbitraire sur un qubit $|\psi\rangle$ est communément représenté par une superposition des vecteurs d'une base orthonormée de $\mathcal{H}^2$. Mathématiquement, on représente cette superposition avec une combinaison linéaire des vecteurs de la base ; les coefficients se nomment *amplitudes*. On choisit la base dite «de calcul» des états $|0\rangle$, $|1\rangle$ qui correspondent aux états 0 et 1 d'un bit classique :

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

Puisque $|\psi\rangle$ correspond à un vecteur unitaire selon la norme euclidienne, la somme des valeurs absolues au carré des amplitudes est égale à 1 :

$$|\alpha|^2 + |\beta|^2 = 1$$

La notation de Dirac est équivalente à la notation vectorielle :

$$\alpha\,|0\rangle + \beta\,|1\rangle \qquad \Longleftrightarrow \qquad \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Définition 2 (Registre de qubits)**.** Un registre de qubits est un tuplet d'un ou plusieurs qubits. Un registre de n qubits dans un état *pur* est représenté par un vecteur unitaire dans $\mathcal{H}^{2^n}$. Sauf avis contraire, lorsque nous faisons référence à l'état d'un registre, nous discutons d'un état pur.

Un registre quantique dans un état arbitraire est noté $|\Psi\rangle$. On peut représenter l'état d'un registre de n qubits par une combinaison linéaire des vecteurs d'une base dans $\mathcal{H}^{2^n}$. Par exemple, pour un registre de deux qubits $|\psi\rangle$ et $|\phi\rangle$, on peut prendre le produit tensoriel des deux qubits pour obtenir une représentation du registre de qubits dans une base de l'espace du registre quantique $\mathcal{H}^4$.

$$\begin{aligned} |\psi\rangle \otimes |\phi\rangle &= (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma\,|0\rangle + \delta\,|1\rangle) \\ &= \alpha\gamma\,|0\rangle \otimes |0\rangle + \alpha\delta\,|0\rangle \otimes |1\rangle + \beta\gamma\,|1\rangle \otimes |0\rangle + \beta\delta\,|1\rangle \otimes |1\rangle \\ &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle \end{aligned}$$

Une puissance tensorielle représente un produit tensoriel répété :

$$|\psi\rangle^{\otimes n} = \underbrace{|\psi\rangle \otimes \cdots \otimes |\psi\rangle}_{n \text{ fois}}$$

Additionnellement, une puissance tensorielle d'un ensemble d'états représente l'ensemble des produits tensoriels possibles des états dans l'ensemble. Par exemple,

$$\{|0\rangle, |1\rangle\}^{\otimes 2} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$

Généralement, il faut 2^n amplitudes pour représenter l'état d'un registre de n qubits, alors que la représentation de l'état de n bits classiques ne requiert trivialement que n bits. Par exemple,

$$|000\rangle \iff \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

2.1.2 Les portes quantiques

Si on veut modifier l'état d'un registre de qubits, c'est-à-dire changer ses amplitudes, on peut le faire avec des portes quantiques.

Définition 3 (Porte quantique)**.** Une porte quantique est une transformation linéaire qui agit sur un qubit ou un registre de qubits. Pour qu'une porte quantique soit une opération

légale selon les lois de la théorie quantique, elle doit préserver le produit scalaire :

$$\langle\psi|\phi\rangle = \langle\psi| U^\dagger U |\phi\rangle \quad \forall |\psi\rangle, |\phi\rangle$$

Mathématiquement, une porte quantique est représentée par une matrice unitaire notée U. Une matrice unitaire est une matrice inversible dont la matrice inverse est aussi sa matrice transposée conjuguée :

$$U^\dagger U = U U^\dagger = I$$

où la matrice transposée conjuguée de U est notée $U^\dagger$.

Une porte quantique qui agit sur un registre de qubits peut être représentée par une multiplication matrice-vecteur. Considérons un qubit dans un état arbitraire $|\psi\rangle$ et une porte quantique U :

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \leftrightarrow \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$U |\psi\rangle = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{pmatrix}$$

Une porte quantique n'est pas limitée à agir sur un seul qubit. Une porte qui change les amplitudes d'un registre de qubits est représentée par une matrice unitaire de taille $2^n \times 2^n$: elle peut modifier les 2^n amplitudes de l'état du registre.

Définition 4 (Arités d'entrée et de sortie d'une porte logique)**.** Dans le domaine de l'informatique, les arités d'entrée et de sortie d'une porte logique sont respectivement le nombre d'éléments en entrée et en sortie de cette porte.

Nous ne considérons que des portes quantiques avec des arités d'entrée et de sortie égales [1]. Lorsqu'on dit qu'une porte quantique a une arité de k, cette porte a donc des arités d'entrée et de sortie égales à k.

Définition 5 (Arité d'une porte quantique)**.** L'*arité* k d'une porte quantique est le nombre de qubits sur lequel cette porte quantique agit. Une porte quantique d'arité k est représentée par une matrice unitaire de $2^k \times 2^k$ nombres complexes.

2.1.3 Les mesures

Il faut mesurer les qubits pour en retirer de l'information classique. Une mesure projette un état quantique sur n qubits vers une séquence de n bits. Si on exprime l'état du registre comme une combinaison linéaire de vecteurs d'une base de l'espace d'Hilbert et qu'on mesure dans cette base, la mesure retourne un des vecteurs de cette base de manière probabiliste.

Définition 6 (Mesure)**.** Une mesure projective, que nous appellerons simplement *mesure*, est une projection d'un état quantique $|\Psi\rangle$ sur n qubits vers un vecteur d'une base orthonormée de $\mathcal{H}^{2^n}$. Sauf avis contraire, la mesure se fait dans la base de calcul $\{|0\rangle, |1\rangle\}^{\otimes n}$. La probabilité que la mesure projette l'état quantique vers un vecteur spécifique de la base dans laquelle on mesure est égale au carré de la valeur absolue de l'amplitude associée à ce vecteur de la base :

1. Ceci est requis pour que la porte soit unitaire. Néanmoins, une porte généralisée pourrait ajouter des qubits ancillaires à l'entrée et/ou jeter des qubits à la sortie, mais nous ne discutons pas de cette possibilité dans ce document.

$$\alpha\,|000\rangle + \beta\,|111\rangle \rightarrow \begin{cases} 000 \text{ avec probabilité } |\alpha|^2 \\ \\ 111 \text{ avec probabilité } |\beta|^2 \end{cases}$$

Nous avons déjà précisé qu'une porte quantique légale doit préserver le produit scalaire (définition 3). Remarquons que cette contrainte permet d'obtenir une distribution de probabilités légale [**50**] après une mesure.

Les mesures transforment de l'information quantique intraitable par un ordinateur classique en bits. On perd de l'information dans l'espace d'Hilbert parce que la projection de l'information quantique en information classique est irréversible, mais on reçoit de l'information traitable par un ordinateur classique. L'espérance probabiliste d'une mesure, c'est-à-dire la moyenne pondérée des résultats des mesures, est notée $\langle\cdot\rangle$: $\langle U\,|\psi\rangle\rangle$ représente l'espérance d'une mesure appliquée à $|\psi\rangle$ après y avoir appliqué la porte U.

Notons que la mesure présentée ici n'est qu'un formalisme utile pour analyser les circuits quantiques. Le concept de mesure est problématique [**75**] parce qu'il est intimement relié à l'interprétation de la théorie quantique.

2.2 Les circuits quantiques

Définition 7 (Architecture quantique)**.** Une *architecture* quantique est une grille en deux dimensions. Cette grille est composée de *fils* horizontaux qui contiennent des qubits, et de *couches* verticales contenant des portes quantiques qui agissent sur les qubits dans les fils. Les qubits dans les fils procèdent à travers l'architecture de gauche à droite et chaque fil de l'architecture est connecté à au plus une porte par couche, tandis que chaque porte agit sur au plus deux fils.

Définition 8 (Circuit quantique)**.** Un *circuit quantique* est obtenu à partir d'une architecture quantique en spécifiant chaque porte quantique de l'architecture par une transformation unitaire de dimension appropriée. Un circuit quantique C qu'on peut obtenir à partir d'une architecture quantique Q en spécifiant correctement les portes de Q est dite « avoir » l'architecture Q.

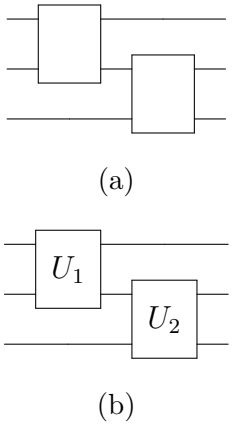

Figure 2.1 – (a) Un exemple d'architecture quantique avec 3 fils et 2 couches. (b) Un circuit quantique obtenu en spécifiant les portes de l'architecture (a) avec les matrices unitaires U_1 et U_2.

Un circuit quantique représente une discrétisation de l'évolution d'un état quantique à travers le temps.

On peut aussi graphiquement représenter les valeurs initiales des qubits d'un circuit et les mesures à la fin du circuit :

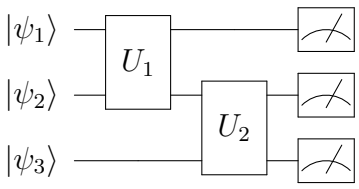

Figure 2.2 – Un circuit quantique avec des valeurs initiales des qubits et des mesures à la fin du circuit. Les symboles $|\psi_1\rangle$, $|\psi_2\rangle$, $|\psi_3\rangle$ sont les états initiaux des qubits sur lesquels agit le circuit, soit l'entrée du circuit. Les portes quantiques U_1 et U_2 agissent sur deux fils chaque. Puis, les cadrans à pointeur représentent des mesures des qubits transformés par les portes U_1 et U_2. Le résultat de la mesure est la sortie du circuit.

Nous définissons ici plusieurs aspects communs aux circuits et aux architectures quantiques.

Définition 9 (Arité d'un circuit ou d'une architecture quantique)**.** L'*arité* d'un circuit ou d'une architecture quantique est l'arité maximale des portes quantiques du circuit ou de l'architecture (définition 5). Un circuit ou une architecture quantique d'arité k est constitué de portes qui agissent sur **au plus** k qubits. Un circuit ou une architecture quantique d'arité k peut donc contenir des portes quantiques d'arité $k, k-1, ..., 1$.

Définition 10 (Taille d'un circuit ou d'une architecture quantique)**.** La *taille* notée t est le nombre de portes quantiques dans un circuit ou une architecture quantique.

Définition 11 (Largeur d'un circuit ou d'une architecture quantique)**.** La *largeur* notée l est le nombre de fils dans un circuit ou une architecture quantique.

Sauf avis contraire, on suppose que des portes quantiques qui agissent sur des sous-ensembles non disjoints d'un registre de qubits ne peuvent pas être effectuées en parallèle. Le nombre de couches d'un circuit quantique détermine alors le temps de calcul nécessaire pour exécuter le circuit.

Définition 12 (Profondeur d'un circuit ou d'une architecture quantique)**.** La *profondeur* notée p est le nombre de couches d'un circuit ou d'une architecture quantique.

2.3 Les ensembles de portes universels et les circuits universels

2.3.1 Les ensembles de portes universels

Un ensemble de portes quantiques est *universel* si toute porte quantique U peut être approximée arbitrairement bien par un circuit quantique constitué des portes de cet ensemble. La plupart des ensembles de portes universels contiennent des portes de petite taille (1 ou 2 qubits, parfois 3). Pour implémenter physiquement un algorithme quantique, il est commun d'utiliser un ensemble d'opérations quantiques primitives qui peuvent approximer n'importe quelle porte quantique. Un ensemble universel de portes quantiques usuel est l'ensemble qui contient la porte CNOT et toutes les portes à un qubit [**63**].

Définition 13 (Ensemble universel de portes quantiques (basée sur [**17**]))**.** Un ensemble de portes quantiques G est *universel* si, pour tout $\epsilon \geq 0$, $l \in \mathbb{N}$ et porte quantique U qui agit sur l qubits, il existe un circuit quantique C_G constitué de portes dans G tel que pour tout état quantique $|\Psi\rangle$ sur l qubits, $\langle\Psi| C_G^\dagger U |\Psi\rangle \geq 1 - \epsilon$.

Si on permet à un circuit d'avoir un nombre arbitraire de portes de l'ensemble universel et qu'on permet aussi que ces portes soient arrangées selon une architecture arbitraire, on peut alors approximer n'importe quelle porte quantique de taille arbitraire.

2.3.2 Les architectures quantiques universelles

Considérons une famille de circuits quantiques F. Une *architecture* quantique universelle pour F est une architecture quantique tel que tous les membres de F peuvent être obtenus à partir de l'architecture quantique universelle en spécifiant les portes de l'architecture.

Définition 14 (Architecture Quantique Universelle)**.** Soit F un ensemble de circuits quantiques qui agissent sur l qubits et Q une architecture de circuit quantique. L'architecture Q est *universelle* pour F si, pour tout circuit C_F dans F, il existe un circuit quantique C_Q avec l'architecture Q tel que pour tout état quantique $|\Psi\rangle$ sur l qubits et pour tout $\epsilon \geq 0$, $\langle\Psi| C_F^\dagger C_Q |\Psi\rangle \geq 1 - \epsilon$.

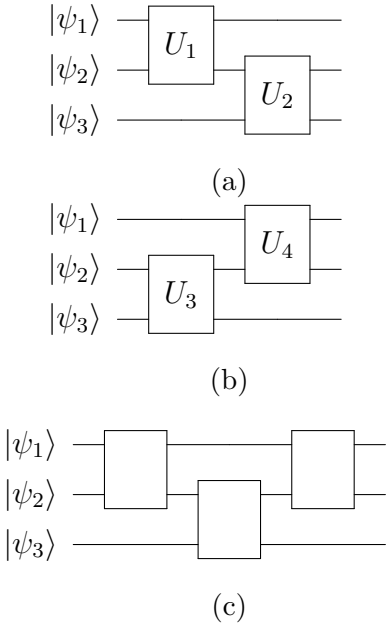

Figure 2.3 – Un exemple d'architecture quantique universelle pour une famille de deux circuits. L'architecture quantique (c) à trois portes est une architecture quantique universelle pour la famille constituée des deux circuits (a) et (b) à deux portes. En spécifiant la première et la deuxième porte de l'architecture (c) respectivement avec U_1 et U_2, on obtient le circuit (a). En spécifiant la deuxième et la troisième porte respectivement avec U_3 et U_4, on obtient le circuit (b).

Définition 15 (Famille de circuits indexée par largeur et profondeur)**.** La famille $F_{l,p}$ contient les circuits quantiques d'arité 2, de largeur l et de profondeur p. Remarquons que $F_{l,p}$ inclut tous les circuits quantiques de largeur l et de profondeur p composés de portes qui agissent sur 1 ou 2 qubits, peu importe leur architecture ou les transformations effectuées par leurs portes quantiques.

2.3.3 Quelques contraintes

2.3.3.1 Des ordinateurs quantiques parfaits

Le problème général pour construire un ordinateur quantique est de créer un système de qubits qui peuvent être contrôlés pour interagir de manière arbitraire tout en préservant les états de ces qubits. Le contrôle des qubits est une tâche exceptionnellement difficile [**71**]. En effet, le contrôle des qubits est une sorte de paradoxe. On cherche à avoir des qubits parfaitement isolés pour éviter que l'environnement vienne perturber leur état. Cependant, on souhaite pouvoir modifier ces qubits comme bon nous semble pour pouvoir traiter l'information que les qubits contiennent. De plus, on souhaite que ces qubits puissent interagir entre eux pour tirer avantage de phénomènes quantiques. On cherche donc à isoler les qubits de leur environnement tout en voulant conserver la possibilité de les modifier et de les faire interagir. Cette tâche déjà difficile s'aggrave en fonction du nombre de qubits qu'on veut contrôler et au temps de calcul du circuit.

Deux sources d'erreurs sont communes : la manipulation imparfaite du qubit et la décohérence. Premièrement, si la transformation U exacte est approximée par $\widetilde{U}$, cette erreur se propage à travers le circuit et s'accumule aux autres erreurs d'approximation. Deuxièmement, à moins qu'un qubit ne soit parfaitement isolé (et conséquemment qu'on ne puisse interagir avec du tout), le qubit fait partie d'un plus gros système quantique qu'on nomme *environnement*[2]. Le qubit interagit avec le reste de l'environnement et, avec le temps, il perd l'information qu'il contient, un processus nommé *décohérence* [**33**]. Pour contrecarrer ces pertes d'information, des techniques de correction d'erreur [**33**] peuvent être utilisées.

2. Une formalisation des interactions entre le registre de qubits d'un ordinateur quantique et l'environnement dans un système quantique ouvert dépasse le cadre de ce mémoire.

Cependant, celles-ci sont généralement chères en nombre de qubits et dépassent rapidement les capacités des prototypes quantiques contemporains.

Il est important de différencier les appareils quantiques modernes qu'on nomme appareils NISQ (*Noisy Intermediate-Scale Quantum*) des ordinateurs quantiques. Les appareils NISQ sont soumis à des contraintes que les ordinateurs quantiques du futur ne connaîtront pas, du moins faut-il l'espérer. Premièrement, les sources d'erreurs mentionnées ci-dessus ne sont pas présentes dans un ordinateur quantique parfait : on suppose un contrôle parfait des qubits pour une durée de temps illimitée. Deuxièmement, un appareil NISQ agit sur un nombre très limité de qubits (généralement moins que 50), tandis qu'un ordinateur quantique peut manipuler un nombre arbitraire de qubits.

Bien qu'il sera intéressant de voir le potentiel des appareils NISQ, on suppose l'accès à un ordinateur quantique parfait dans ce mémoire. Cependant, il est difficile de construire physiquement des portes quantiques qui agissent sur plus que deux qubits. Alors, nous considérons des circuits quantiques d'arité 2 : chaque porte agit sur au plus 2 qubits.

2.3.3.2 Des architectures fixes

Le cadre de ce travail suppose que l'architecture d'un circuit quantique utilisé dans un algorithme est fixe. En d'autres mots, un algorithme qui utilise un circuit quantique peut modifier la transformation qu'une porte quantique effectue, mais il ne peut pas changer le sous-ensemble de qubits transformés par la porte. L'architecture est initialisée au début de l'algorithme et ne subit pas de changements.

Cette contrainte peut compliquer l'implémentation de certains algorithmes. Considérons l'algorithme classique du tri par fusion (*mergesort*). Le nombre de comparaisons pour trier

l éléments est de l'ordre de $l \lg l$. Cependant, l'algorithme se base sur une séquence de comparaisons, et le choix de quelle comparaison faire dépend des résultats des comparaisons antérieures. Alors, l'architecture d'un circuit naïf qui tente d'implémenter le tri par fusion n'est pas fixe : les éléments comparés durant une étape de l'algorithme seront différents selon les étapes antérieures de l'algorithme. Dans le cas général d'un algorithme qu'on cherche à faire tourner sur un circuit avec une architecture fixe, il n'est pas évident que la taille du circuit fixe résultant correspondrait au nombre d'opérations de l'algorithme.

Chapitre 3

Les réseaux de neurones

Définition 16 (Apprentissage [**58**])**.** On dit d'un programme informatique qu'il *apprend* par l'expérience `E` en ce qui concerne une certaine classe de tâches `T` et une mesure de performance `P`, si sa performance aux tâches de `T`, telle que mesurée par `P`, s'améliore avec l'expérience `E`.

L'apprentissage automatique est un sous-domaine de l'informatique qui considère des algorithmes qui s'améliorent automatiquement en apprenant à partir de données. Un algorithme d'apprentissage automatique qui cherche à résoudre une certaine tâche utilise des exemples de la tâche qu'il essaie de résoudre pour modifier son propre comportement. L'apprentissage se distingue notamment de la mémorisation : un algorithme d'apprentissage automatique essaie d'apprendre à performer sur des occurrences de sa tâche qu'il n'a jamais rencontrées auparavant. On parle alors de généralisation.

Définition 17 (Généralisation)**.** La capacité de généralisation d'un algorithme d'apprentissage automatique est son aptitude à être performant sur des occurrences inconnues d'une tâche. L'*erreur de généralisation* d'un algorithme d'apprentissage automatique est l'écart entre sa performance à une tâche sur des occurrences desquelles il a déjà appris et sur de nouvelles occurrences de la tâche.

Dans ce chapitre, nous introduisons initialement l'apprentissage supervisé. Ensuite, nous présentons plusieurs aspects sur les réseaux de neurones, incluant la descente de gradient par rétropropagation et certains théorèmes d'approximation universelle.

3.1 L'apprentissage supervisé

Dans la définition de l'apprentissage de Mitchell (définition 16), un algorithme d'apprentissage essaie de résoudre une tâche `T`. Les tâches qu'un algorithme d'apprentissage peut attaquer peuvent être très complexes. Cependant, dans le cadre de ce mémoire, on considère qu'une tâche spécifie le type de prédiction qu'on attend de l'algorithme :

Définition 18 (Tâche, caractéristiques, cible, exemple)**.** En apprentissage supervisé, une *tâche* est une spécification de deux espaces X et Y. L'espace X est l'espace des *caractéristiques* : un vecteur $\mathbf{x} \in X$ est un vecteur dit de «caractéristiques» et il représente numériquement des observations qu'un algorithme d'apprentissage prend en entrée. L'espace Y est l'espace des *cibles* : un scalaire $y \in Y$ est une «cible» et représente une occurrence de ce qu'un algorithme d'apprentissage automatique essaie de produire en sortie. Un *exemple* est une paire ($\mathbf{x}$,y) de caractéristiques et de cibles.

En apprentissage supervisé, l'expérience `E` prend la forme d'un ensemble de *données d'entraînement*.

Définition 19 (Données d'entraînement)**.** Supposons l'existence d'une loi de probabilité D jointe sur X et Y. Les *données d'entraînement* sont un ensemble d'exemples $S = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$ tirés de D. Succinctement, on note $S \sim D$.

Définition 20 (Apprentissage supervisé)**.** Supposons que la loi de probabilité jointe D est inobservée : on n'a accès qu'aux données d'entraînement S. L'*apprentissage supervisé* utilise

S pour essayer de trouver une fonction $f^* : X \to Y$ telle que, pour le plus de paires $(\mathbf{x}, y) \sim D$ possibles, $f^*(\mathbf{x})$ est aussi similaire que possible à y.

3.1.1 Le choix de modèle et le paramétrage

Considérer toutes les fonctions possibles de la forme $f : X \to Y$ est évidemment intraitable. Donc, on se limite à considérer un sous-ensemble de ces fonctions. Ce sous-ensemble est une *classe d'hypothèses*.

Définition 21 (hypothèse, Classe d'hypothèses, prédiction)**.** Considérons une tâche spécifiée par deux espaces X et Y. Une *classe d'hypothèses* H est l'ensemble des fonctions considérées par un programme d'apprentissage automatique pour attaquer une certaine tâche. Un membre $h \in H$ est une *hypothèse*. La *prédiction* $\hat{y}$ d'une hypothèse h pour un vecteur de caractéristiques $\mathbf{x}$ est la sortie $h(\mathbf{x})$.

En général, il n'existe pas d'algorithme d'apprentissage qui performe mieux qu'un autre sur toutes les tâches possibles [**91, 92**]. Le choix de H dépend de la tâche et on le vérifie empiriquement.

En apprentissage automatique, une classe d'hypothèses peut être spécifiée par des *paramètres* :

Définition 22 (paramètre, modèle)**.** Un *modèle*[3] d'apprentissage automatique est une fonction paramétrée $h(\boldsymbol{\theta}; \mathbf{x})$[4] qui prend en entrée non seulement un vecteur de caractéristiques $\mathbf{x}$, mais aussi un vecteur de *paramètres* $\boldsymbol{\theta} = [\theta_1, ..., \theta_{|\boldsymbol{\theta}|}]$, $\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}$. La classe d'hypothèses H

3. Avec cette définition de modèle, deux réseaux de neurones de tailles, de largeurs ou de profondeurs différentes sont deux modèles différents.

4. Par simplicité de présentation, on omet parfois $\mathbf{x}$. Cependant, il est toujours sous-entendu qu'un modèle paramétré prend en entrée un vecteur de caractéristiques.

spécifiée par un modèle est l'ensemble des fonctions $h(\boldsymbol{\theta}; \mathbf{x})$ avec des valeurs différentes de $\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}$.

Une classe d'hypothèses n'est pas restreinte à un seul modèle : l'union de deux classes d'hypothèses est une classe d'hypothèses elle-même.

3.1.2 La fonction de perte et le risque

Nous avons spécifié que l'apprentissage supervisé cherche une fonction f^* qui, pour le plus de paires $(\mathbf{x}, y) \sim D$ possibles, prend $\mathbf{x}$ en entrée et retourne une prédiction $\hat{y}$ aussi similaire que possible à y. Ce qu'on entend par une prédiction $\hat{y}$ «aussi similaire que possible» à la cible y dépend de la mesure de performance `P` de la définition 16 de Mitchell.

Notons que la mesure de performance se calcule habituellement sur les prédictions faites à partir des données d'entraînement et sur les prédictions faites à partir d'un ensemble de données S_{eval} que l'algorithme n'a jamais vues pour évaluer son erreur de généralisation (définition 17).

Le choix de mesure de performance dépend de la tâche que l'algorithme attaque et du contexte de la tâche. Le type de donnée de la cible influence particulièrement le choix de mesure de performance. Par exemple, pour une tâche où $Y = \mathbb{R}$, communément appelée une tâche de *régression*, une mesure de performance populaire est l'erreur quadratique : $\sum_{i=1}^{|S_{\text{eval}}|}(\hat{y}_i - y_i)^2$, $(\mathbf{x_i}, y_i) \in S_{\text{eval}}$. Pour une tâche dite de *classification* binaire où $Y = \{0,1\}$ et 0 et 1 représentent des classes, un exemple courant de mesure de performance est le pourcentage d'exemples correctement classifiés.

Définition 23 (Fonction de perte)**.** En apprentissage supervisé, une *fonction de perte* est une fonction $\ell : Y \times Y \to \mathbb{R}^{\geq 0}$ qui mesure quantitativement la similarité entre une prédiction $\hat{y}$ et une cible y.

Alors qu'une mesure de performance sert généralement à évaluer une hypothèse après l'apprentissage, l'entraînement utilise une fonction de perte. Plus la valeur de la fonction de perte pour une paire $(\hat{y}, y)$ est grande, moins la prédiction $\hat{y}$ est similaire à la cible y.

On peut reformuler le problème de l'apprentissage supervisé en disant que nous cherchons à choisir une hypothèse $h \in H$ qui minimise la valeur de la fonction de perte ℓ sur tous les exemples possibles définis par la distribution jointe D. Cette quantité porte le nom de *vrai risque*.

Définition 24 (vrai risque [**59**])**.** Pour une hypothèse $h \in H$, le *vrai risque* $R[h]$ est l'espérance de la fonction de perte ℓ sur un exemple généré par D :

$$R[h] \equiv \mathbb{E}_{(\mathbf{x},y)\sim D}[\ell(h(\mathbf{x}),y)]$$

En trouvant une hypothèse h qui minimise le vrai risque, on pourrait trouver la meilleure fonction possible pour la tâche qu'on tente d'attaquer. Cependant, on suppose que la distribution D est inobservée : on a seulement accès aux données S générées à partir de D.

3.1.3 La minimisation du risque empirique

Par opposition au vrai risque, le *risque empirique* se définit sur l'ensemble de données S.

Définition 25 (risque empirique [**59**])**.** Pour un membre h de la classe d'hypothèses et un ensemble d'entraînement $S = \{(\mathbf{x_i},y_i)\}_{i=1}^{n}$ de n exemples, le *risque empirique* $\hat{R}_S[h]$ est la moyenne de la fonction de perte ℓ sur l'ensemble de données S :

$$\hat{R}_S[h] = \frac{1}{|S|}\sum_{i=1}^{|S|}\ell(h(\mathbf{x}_i),y_i)$$

Le but de l'apprentissage automatique est de bien performer non seulement sur les données d'entraînement $S \sim D$, mais aussi sur des données inobservées qui sont générées à partir de D. L'écart entre la performance d'un algorithme d'apprentissage sur les exemples dans S et sur d'autres exemples générés à partir de D, mais qui ne sont pas dans S, est l'erreur de généralisation (définition 17).

Puisque calculer le vrai risque est impossible sans observer D, on se fie généralement au principe de minimisation du risque empirique.

Définition 26 (Minimisation du risque empirique [**86**])**.** Considérons une hypothèse $\hat{h}_S \in H$ qui minimise le risque empirique (définition 25) :

$$\hat{h}_S = \arg\min_{h \in H} \frac{1}{|S|} \sum_{i=1}^{|S|} \ell(h(\mathbf{x_i}), y_i)$$

Le principe inductif de la *minimisation du risque empirique* suppose que le vrai risque (définition 24) de $\hat{h}_S$ est proche du vrai risque minimal des hypothèses dans H.

La validité de cette supposition repose sur un traitement mathématique qui dépasse le cadre de ce mémoire [**86**], mais minimiser le risque empirique n'est pas suffisant en général pour obtenir un bon algorithme d'apprentissage. Il suffit d'observer que le principe de minimisation du risque empirique choisit une hypothèse $\hat{h}_S$ performante sur S, mais pas nécessairement sur tous les exemples générés à partir de D. Si on souhaite apprendre et minimiser l'erreur de généralisation, il faut guider le choix de $\hat{h}_S$ vers un minimum approprié en modifiant l'objectif de minimisation, ou bien définir la classe d'hypothèses H de telle sorte que son minimum soit performant sur D, et pas seulement sur S.

3.1.4 L'optimisation et l'apprentissage

Considérons un ensemble de données S et une classe d'hypothèses H. Trouver l'hypothèse $h \in H$ qui minimise le risque empirique est un problème d'optimisation.

Si H est définie par un modèle $h(\boldsymbol{\theta})$, la paramétrisation $\boldsymbol{\theta}^*$ qui minimise le risque empirique sur S correspond à une hypothèse dont le vrai risque est proche du vrai risque minimal de H tant que le principe de minimisation du risque empirique est valide [5]. En d'autres mots, le vrai risque de l'hypothèse $h(\boldsymbol{\theta^*})$ qui minimise le risque empirique est très proche du vrai risque minimal des hypothèses dans H.

Puisque $\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}$, la famille d'hypothèses H définie par le modèle $h(\boldsymbol{\theta})$ contient une quantité indénombrable de fonctions : explorer toutes les fonctions dans H est impossible.

Souvent, en apprentissage automatique, les algorithmes d'optimisation sont *itératifs* : en partant d'une valeur initiale des paramètres du modèle, l'algorithme d'optimisation génère une séquence de valeurs des paramètres qui réduisent le risque empirique.

Définition 27 (Algorithme d'optimisation itératif)**.** Un algorithme d'optimisation *itératif* est un algorithme d'optimisation heuristique qui génère une séquence de candidats $c_0, c_1, ..., c_n$ qui répondent de mieux en mieux à un critère de sélection qui définit le problème d'optimisation que l'algorithme attaque. Chaque séquence d'étapes qui choisissent un nouveau candidat c_i est une *itération*. Le candidat initial c_0 est choisi de manière heuristique.

Dans notre contexte d'apprentissage supervisé, le critère de sélection que nous considérons est le risque empirique et les candidats sont des valeurs des paramètres $\boldsymbol{\theta}$ d'un modèle $h(\boldsymbol{\theta})$. L'algorithme génère une séquence $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n$ de valeurs candidates des paramètres

5. Pour le reste de ce mémoire, nous faisons la supposition que le principe de minimisation du risque empirique est valide.

où le risque empirique de chaque candidat $\boldsymbol{\theta}_i$ n'est pas plus grand que celui des candidats précédents dans la séquence :

$$\hat{R}_S[h(\boldsymbol{\theta}_i)] \leq \hat{R}_S[h(\boldsymbol{\theta}_j)] \quad \forall i,j \in \{1,...,n\} \mid i > j$$

L'algorithme itératif s'arrête lorsqu'il atteint un critère d'arrêt tel que le nombre d'itérations sans amélioration ou le temps d'exécution. Notons qu'il n'y a aucune garantie qu'un algorithme d'optimisation itératif atteigne un ensemble de paramètres $\boldsymbol{\theta}^*$ qui minimise le risque empirique. Généralement, les algorithmes itératifs considérés en apprentissage automatique sont des algorithmes d'escalade (*hill climbing*).

Définition 28 (Escalade)**.** Les algorithmes d'*escalade* sont des algorithmes itératifs d'optimisation tels qu'à l'itération i, le candidat c_{i+1} est un voisin du candidat c_i qui obtient une meilleure valeur du critère de sélection.

Dans notre contexte d'apprentissage supervisé, le candidat $\boldsymbol{\theta}_{i+1}$ est un voisin de $\boldsymbol{\theta}_i$ avec un risque empirique plus bas ou égal. Le concept de «voisin» dépend de l'algorithme d'escalade en particulier : dans un espace continu comme $\mathbb{R}^{|\boldsymbol{\theta}|}$, on suppose qu'un algorithme d'escalade considère un voisin $|\boldsymbol{\theta}| + \lambda$ où le *pas*[6], noté λ, est une modification spécifiée par l'algorithme. Si le voisin considéré a un risque empirique plus bas, alors il est le prochain candidat : $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \lambda$.

6. Le pas λ peut modifier un ou plusieurs coefficients de $\boldsymbol{\theta}$.

3.1.5 L'erreur d'estimation vs l'erreur d'approximation

Nous essayons de trouver une hypothèse $h(\boldsymbol{\theta}^*) \in H : \{h(\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}\}$ avec un vrai risque proche du minimum. En d'autres mots, nous voulons trouver une hypothèse avec une petite erreur de généralisation.

Nous pouvons décomposer cette erreur de généralisation en deux. L'*erreur d'approximation* est l'erreur associée à une mauvaise sélection de la classe d'hypothèses H. L'*erreur d'estimation* est l'erreur associée à un choix sous-optimal de valeurs des paramètres $\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$.

Définition 29 (Erreur d'approximation)**.** Considérons un modèle $h(\boldsymbol{\theta})$ qui définit une classe d'hypothèses H. L'*erreur d'approximation* est l'erreur de généralisation du modèle optimal $h(\boldsymbol{\theta^*})$ dans H, c'est-à-dire $R[h(\boldsymbol{\theta^*})]$.

Définition 30 (Erreur d'estimation)**.** Considérons un modèle $h(\boldsymbol{\theta})$ qui définit une classe d'hypothèses H. L'*erreur d'estimation* d'un choix de paramètres $\boldsymbol{\theta}$ est la différence entre les vrais risques associés à ce choix et le choix optimal :

$$R[h(\boldsymbol{\theta})] - R[h(\boldsymbol{\theta^*})]$$

On peut alors dire que l'erreur de généralisation d'un modèle $h(\boldsymbol{\theta})$ est la somme de l'erreur d'approximation de la classe d'hypothèses spécifiée par le modèle et l'erreur d'estimation associée au choix des valeurs de $\boldsymbol{\theta}$. Remarquons que l'erreur d'approximation est indépendante des valeurs des paramètres, mais dépend de la sélection d'un modèle $h(\boldsymbol{\theta})$ ou d'une classe d'hypothèses H.

Considérons un exemple pour démontrer l'importance de cette distinction. Supposons que la distribution inobservée D correspond à une fonction sinus : $y = \sin(x)$. Supposons

aussi que la classe d'hypothèses considérée par un algorithme d'apprentissage est l'ensemble des fonctions linéaires : $H : \{h(\alpha, \beta; x) = \alpha x + \beta \mid \alpha,\ \beta \in \mathbb{R}\}$. Spécifions la fonction de perte ℓ comme étant l'erreur quadratique moyenne $\ell(\hat{y},y) = (y - \hat{y})^2$. La fonction linéaire optimale est trivialement $h(x) = 0$. L'erreur d'estimation serait donc 0 : les valeurs optimales des paramètres sont $\alpha = 0, \beta = 0$. L'erreur d'approximation est l'erreur entre cette fonction linéaire optimale et la fonction inobservée $h(x) = \sin(x)$. Si on échantillonne de manière uniforme les points (x,y) des nombres réels, l'erreur d'approximation est de $2/\pi$.

3.2 Les réseaux de neurones

Dans cette section, nous présentons rapidement les réseaux de neurones, une classe de modèles d'apprentissage automatique reliée au connexionnisme : plusieurs modèles simples qu'on nomme neurones sont interconnectés pour obtenir un modèle complexe avec des comportements émergents.

Les réseaux de neurones sont centraux à l'apprentissage profond, un sous-domaine de l'apprentissage automatique. L'apprentissage profond est devenu populaire en raison de plusieurs succès empiriques dans des domaines tels que la vision par ordinateur, la reconnaissance vocale et le traitement automatique du langage naturel.

3.2.1 L'unité de base : le neurone

Un réseau de neurones est composé de *neurones* interconnectés.

Définition 31 (Neurone)**.** Un *neurone*, dans le contexte d'un réseau de neurones comme modèle d'apprentissage automatique, est un modèle d'apprentissage automatique simple qui prend un vecteur en entrée et qui retourne un scalaire. Pour un vecteur d'entrée $\mathbf{x}$, un neurone est généralement représenté par la composition d'une régression linéaire et d'une fonction

non linéaire $a : \mathbb{R} \to \mathbb{R}$:

$$h(b, \mathbf{w}; \mathbf{x}) = a\left(\sum_{i=1}^{|\mathbf{x}|} x_i w_i + b\right) = a(\mathbf{x}^\intercal \mathbf{w} + b),\ \mathbf{w} \in \mathbb{R}^{|\mathbf{x}|}, b \in \mathbb{R}$$

Remarquons que nous avons remplacé le vecteur de paramètres $\boldsymbol{\theta}$ par un vecteur de *poids* $\mathbf{w}$ et un scalaire b nommé *biais*. La fonction non linéaire est la *fonction d'activation*. La classe d'hypothèses H d'un neurone est spécifiée par les valeurs possibles de $\mathbf{w}$ et b : un algorithme d'apprentissage cherche à apprendre les valeurs des poids et du biais de ce neurone.

3.2.2 Des neurones interconnectés : le réseau de neurones

Si on connecte plusieurs neurones ensemble, on obtient un *réseau de neurones*.

Définition 32 (Réseau de neurones)**.** Un *réseau de neurones* est un modèle d'apprentissage composé de plusieurs neurones interconnectés : les entrées de certains neurones sont les sorties de certains autres neurones. Les paramètres du réseau de neurones sont les poids et les biais des neurones qui composent le réseau de neurones. Lorsque tous les neurones de chaque couche sont connectés à tous les neurones de la couche subséquente, le réseau de neurones est *entièrement connecté*.

On représente graphiquement un réseau de neurones par un graphe orienté acyclique (figure 3.1). Par convention, le graphe procède de gauche à droite, ce qui permet d'omettre les têtes des flèches. Chaque tranche verticale du réseau est une *couche* du réseau. La couche la plus à gauche est la couche *d'entrée* et chaque neurone de la couche d'entrée représente une caractéristique (définition 18) du vecteur de caractéristiques que le modèle prend en entrée. La couche la plus à droite est la couche *de sortie* : les sorties des neurones dans la couche de sortie représentent les sorties du modèle. Les couches entre celles d'entrée et de sortie sont

des couches dites *cachées*. La *taille* t d'un réseau de neurones est le nombre de neurones dans le réseau. La *profondeur* p d'un réseau de neurones est le nombre de couches cachées dans le réseau. La *largeur* l d'une couche cachée est son nombre de neurones. Lorsque nous disons qu'un réseau de neurones a une largeur l, nous sous-entendons que chaque couche cachée du réseau est composée d'au plus l neurones.

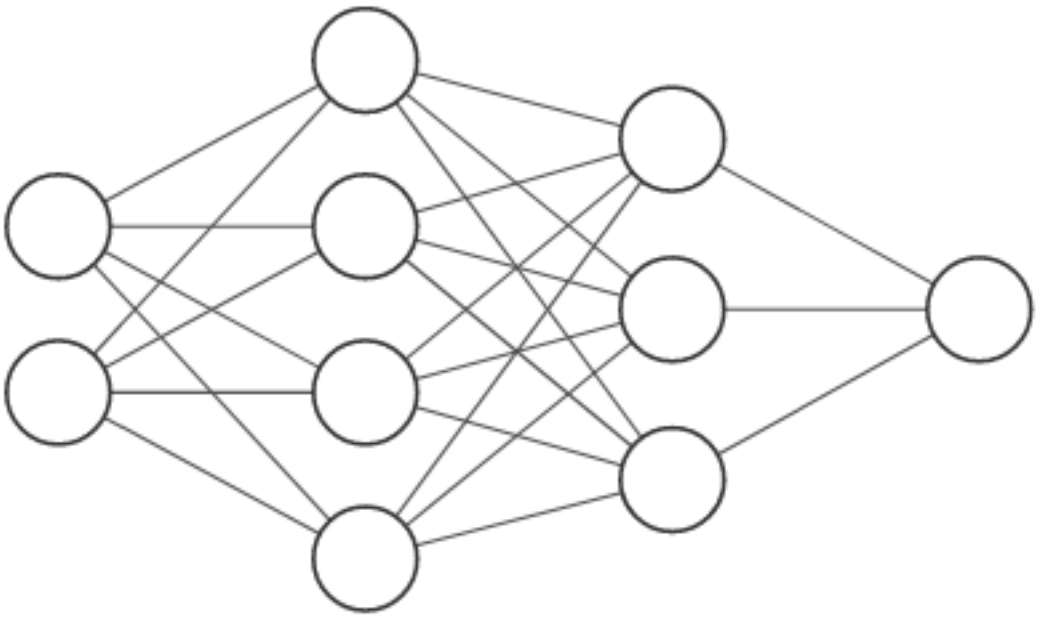

Figure 3.1 – Un exemple de représentation graphique d'un réseau de neurones entièrement connecté. La couche d'entrée (gauche) contient deux neurones qui n'ont aucune entrée : la sortie de chacun de ces neurones est une caractéristique. La couche de sortie contient un neurone : la sortie du modèle a une dimension. Le réseau de neurones a deux couches cachées : la première couche cachée a quatre neurones et la deuxième couche cachée a trois neurones.

Considérons un réseau de neurones entièrement connecté de largeur l : chaque neurone d'une couche cachée peut être interprété comme une «porte» d'arité d'entrée l qui effectue une transformation $\mathbb{R}^l \to \mathbb{R}$. Dans un réseau de neurones entièrement connecté, chaque neurone a une arité d'entrée égale au nombre de neurones dans la couche précédente et une arité de sortie égale à 1 qui est donnée en entrée à tous les neurones de la couche subséquente [7].

7. Chaque neurone dans la couche d'entrée a une arité d'entrée de 0 et une arité de sortie de 1 et chaque neurone dans la couche de sortie a une arité de sortie de 1.

3.3 La descente de gradient

Un des algorithmes d'optimisation itératifs les plus communs en apprentissage automatique est la descente de gradient [**32**] : si H est définie par un modèle $h(\boldsymbol{\theta})$, l'optimisation des paramètres $\boldsymbol{\theta} = [\theta_1,...,\theta_{|\boldsymbol{\theta}|}]$ du modèle se fait en calculant le gradient de la fonction de perte par rapport aux paramètres, c'est-à-dire la dérivée partielle de la fonction de perte $\ell(\hat{y},y)$ [8] par rapport à chacun des paramètres du modèle :

$$\left[\frac{\partial \ell(\hat{y},y)}{\partial \theta_1}, ..., \frac{\partial \ell(\hat{y},y)}{\partial \theta_{|\boldsymbol{\theta}|}}\right] = \nabla_{\boldsymbol{\theta}} \ell(\hat{y},y)$$

Puis, l'algorithme modifie les paramètres proportionnellement à la moyenne du gradient de la fonction de perte par rapport aux paramètres sur les exemples dans les données d'entraînement S en fonction du pas λ :

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \lambda \frac{1}{|S|} \sum_{i=1}^{|S|} \nabla_{\boldsymbol{\theta}} \ell(h(\boldsymbol{\theta}_i; \mathbf{x}_i), y_i)$$

En d'autres mots, on prend un pas proportionnel au gradient de la fonction de perte par rapport aux paramètres. À force de répétition, les valeurs des paramètres tendent vers un minimum local de la valeur de ℓ où les gradients des paramètres sont nuls.

3.3.1 La descente de gradient stochastique

La descente de gradient peut aussi se faire sur un sous-ensemble des exemples dans S. La descente de gradient *stochastique* [**73**] est une version modifiée de la descente de gradient où chaque itération utilise un sous-ensemble aléatoire des exemples d'entraînement. La descente

8. Rappelons que $h(\boldsymbol{\theta}; \mathbf{x}) = \hat{y}$ est la sortie d'un modèle, tandis que y est la vraie cible associée au vecteur de caractéristiques $\mathbf{x}$.

de gradient stochastique dite «pure» effectue chaque itération d'optimisation avec un seul exemple :

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \lambda \nabla_{\boldsymbol{\theta}} \ell(h(\boldsymbol{\theta}_i; \mathbf{x}), y), \ (\mathbf{x}, y) \in S$$

Le terme englobe aussi des variantes de la descente de gradient stochastique où chaque itération utilise un sous-ensemble strict de S [**35**].

3.3.2 La rétropropagation

Définition 33 (Rétropropagation [9])**.** La rétropropagation est un algorithme qui calcule le gradient par rapport aux paramètres $\nabla \ell_{\boldsymbol{\theta}}(\hat{y}_i, y_i)$ d'un réseau de neurones avec un nombre d'opérations linéaire dans le nombre de paramètres du réseau de neurones.

Brièvement, remarquons que le calcul du gradient de la fonction de perte par rapport aux paramètres d'un neurone dans la couche i requiert le gradient de la fonction de perte par rapport aux paramètres des neurones dans la couche $i + 1$. Il est donc avantageux de calculer le gradient par rapport aux paramètres des neurones de la couche $i + 1$, de stocker en mémoire ce gradient et de réutiliser ces valeurs stockées durant le calcul du gradient par rapport aux paramètres de la couche i. En calculant les dérivées partielles de la fonction de perte par rapport aux paramètres du réseau de cette manière, le nombre d'opérations pour calculer chaque dérivée partielle est indépendant de la taille du réseau. On obtient alors

9. La rétropropagation n'est pas présentée en détail dans ce document. La section 6.5 de [**32**], le chapitre 2 de [**62**] et l'article de blogue [**64**] sont des introductions à la rétropropagation dans un ordre décroissant de formalité.

le gradient de la fonction de perte par rapport aux paramètres du réseau avec un nombre d'opérations linéaire dans le nombre de paramètres [10].

Ultimement, l'avantage de la rétropropagation qu'on cherche à souligner est sa *réutilisation de l'information*. En calculant les dérivées partielles en remontant le réseau à partir de la sortie, la rétropropagation évite de traverser le réseau en son entièreté pour calculer chaque dérivée partielle, ce qui évite un temps de calcul quadratique dans le nombre de paramètres.

3.4 L'universalité des réseaux de neurones

3.4.1 Les théorèmes d'approximation universelle

Les théorèmes d'approximation universelle sont des résultats mathématiques qui spécifient les classes d'hypothèses de réseaux de neurones en fonction de leur largeur l ou de leur profondeur p [**18, 39, 42, 49, 51, 56, 68**]. Le résultat de Cybenko en 1989 démontre que, sous certaines conditions et avec une fonction d'activation spécifique, un réseau de neurones avec une seule couche cachée de largeur possiblement exponentielle dans la largeur de l'entrée peut approximer n'importe quelle fonction continue qui agit sur des espaces compacts [**18**]. Deux ans plus tard, Hornik a généralisé le résultat aux réseaux de neurones avec des fonctions d'activation arbitraires, en autant qu'elles soient continues, non constantes et bornées [**42**].

Récemment, des théorèmes d'approximation universelle sur des réseaux de neurones avec une largeur linéaire dans la largeur de l'entrée et une profondeur arbitraire ont été découverts [**39, 49, 51, 56, 68**], mais il reste à déterminer pour quelles familles de fonctions une

10. On suppose ici que le calcul de chaque dérivée partielle par rétropropagation pour un paramètre d'un neurone exige un temps constant si on a accès au gradient de la fonction de perte par rapport aux paramètres des neurones de la couche subséquente.

profondeur exponentielle dans la largeur de l'entrée est nécessaire pour garantir une capacité d'approximation. Une autre question de recherche intéressante est l'existence de théorèmes d'approximation universelle pour des réseaux de neurones avec largeur et profondeur polynomiales dans la largeur de l'entrée.

Mettons l'accent sur le fait que les théorèmes d'approximation universelle indiquent l'*existence* de paramétrages qui correspondent arbitrairement bien à toutes les fonctions dans certaines familles. Ces théorèmes ne garantissent rien quant à l'*apprentissage des paramètres*. Ces théorèmes sont surtout utiles pour comprendre l'étendue des classes d'hypothèses de la famille des réseaux de neurones, mais ils n'offrent aucune information sur les méthodes d'optimisation appropriées pour explorer ces classes d'hypothèses.

Chapitre 4

L'apprentissage automatique quantique

En 2003, Peter Shor a publié un article [**82**] intitulé «Pourquoi n'a-t-on pas trouvé plus d'algorithmes quantiques ?» [11] qui mettait en évidence l'écart entre le nombre d'algorithmes découverts dans les cadres classique et quantique. Une des possibilités qu'il considère est que l'ordinateur quantique opère d'une manière tellement différente des ordinateurs classiques que les techniques de conception d'algorithmes et nos intuitions pour comprendre les modèles de calcul échouent. La *quantisation* d'algorithmes classiques est donc possiblement une approche contre-productive.

Définition 34 (Quantisation [**3**])**.** La *quantisation* d'un algorithme classique est la conversion partielle ou totale de cet algorithme classique en un algorithme quantique afin d'améliorer ses performances.

En d'autres mots, il est difficile de découvrir de nouveaux algorithmes quantiques manuellement ; il se pourrait donc que l'apprentissage automatique, un domaine centré sur les

11. Le titre original en anglais est «Why Haven't More Quantum Algorithms Been Found ?»

algorithmes qui s'améliorent automatiquement à l'aide de données, soit très complémentaire à l'informatique quantique.

L'apprentissage automatique quantique est le domaine résultant de la rencontre de l'apprentissage automatique et de l'informatique quantique. On sépare les approches à l'apprentissage automatique quantique en quatre selon deux critères :

— Les données sont-elles classiques ou quantiques ?

— Le traitement de l'information est-il strictement classique ou inclut-il aussi le traitement de l'information quantique ?

Ce faisant, on obtient quatre approches différentes à l'apprentissage automatique quantique présentées dans le tableau 4.1.

Nous discutons surtout de l'approche qui utilise des données classiques et une combinaison de traitement quantique et classique de ces données : nous cherchons à apprendre des circuits quantiques.

4.1 Un survol de l'apprentissage automatique quantique

En 2008, Sébastien Gambs a publié la première thèse de doctorat portant sur l'apprentissage automatique quantique [**30**]. Comme il le dit, il y avait à cette époque 14 articles dans la section *quant-ph* de l'arXiv qui contenaient le mot «learning», soit environ 700 de moins qu'en septembre 2021. Maintenant, le terme «quantum machine learning» correspond à lui seul à au moins 250 articles dans cette section de l'arXiv. Plus de 200 de ces articles ont paru depuis 2018, suggérant que la croissance du domaine n'a pas directement suivi l'explosion de l'apprentissage profond en 2012, mais qu'elle a été marquée plutôt par la publication de [**12**] en 2017.

Données	Traitement de l'information	Description
Classiques	Classique	Apprentissage automatique classique, incluant les cas où les données sont une description classique d'un état quantique afin d'être traitées par un processus classique.
Classiques	Quantique	Les données doivent être encodées dans l'état d'un registre quantique avant d'être traitées par un circuit quantique. L'algorithme doit inclure des mesures pour extraire de l'information classique du système quantique.
Quantiques	Classique	Les données en entrée sont déjà encodées dans un système quantique. Ces algorithmes n'effectuent aucun traitement quantique des données, mais débutent plutôt par des mesures sur les états quantiques en entrée. Après la mesure, l'algorithme est purement classique.
Quantiques	Quantique	Les données en entrée sont représentées par un système quantique et sont traitées par un circuit quantique. Cette catégorie inclut surtout des sous-routines quantiques qui peuvent être utilisées par une des approches précédentes.

Tableau 4.1 – Un tableau décrivant quatre différentes approches à l'apprentissage automatique classique ou quantique en fonction de la nature classique ou quantique des données et du traitement de l'information.

Pour un survol des dernières années sur l'apprentissage automatique quantique, nous recommandons l'article de Dunjko et Briegel [**25**] de 2018. Un bon point de départ pour le cadre de données classiques et processeur quantique est l'article de Biamonte, Wittek, Pancotti, Rebentrost, Wiebe et Lloyd [**12**] : rédigé par plusieurs autorités dans le domaine, il en introduit les grandes lignes et il fournit une longue liste de références. Cependant, sa lecture devrait être accompagnée par celle d'Aaronson [**1**] pour comprendre les limitations de

l'algorithme HHL [**40**]. Pour un survol des fondements théoriques de l'apprentissage quantique, nous recommandons [**5**] et les chapitres 4 à 6 de [**30**]. Pour une introduction spécialisée sur l'apprentissage automatique quantique supervisé, le livre de Maria Schuld et Francesco Petruccione [**79**] est un ouvrage semi-manuel semi-revue de littérature rempli d'exemples pédagogiques et de clarifications de concepts de base tels que les différents encodages de données et les différentes formulations de problèmes. Si on cherche tout de même une revue de littérature plus récente, nous recommandons la pièce de perspective [**28**] et la consultation de sa liste de références.

Les trois paragraphes suivants offrent des sélections d'articles pertinents en apprentissage automatique quantique supervisé, non supervisé et par renforcement [12].

En 2003, Anguita et al. ont publié un premier article sur l'entraînement d'une machine à vecteurs de support avec un ordinateur quantique (SVM) [**4**]. Depuis, plusieurs quantisations de modèles d'apprentissage supervisé ont été développés [**40, 41, 44, 72, 79, 89**].

Durant son doctorat, Sébastien Gambs a quantisé des algorithmes d'apprentissage non supervisé comme l'algorithme de regroupement k-médianes [**30, 3**]. Depuis, plusieurs quantisations d'algorithmes de regroupement ont été développées [**12, 46, 47, 66, 69, 89**].

Des travaux initiaux en apprentissage par renforcement quantique ont été présentés dans [**21, 22, 23**]. Depuis, plusieurs autres travaux ont été faits [**15, 26, 27, 38, 45, 52, 53, 67**].

4.1.1 L'algorithme HHL : un avertissement

L'algorithme HHL [**40**] est un algorithme pour résoudre certains ensembles d'équations linéaires avec un nombre d'étapes logarithmique dans le nombre d'équations, ce qui suggère la

12. Ces termes ne sont pas définis dans ce mémoire, mais nous supposons que le lectorat qui souhaite consulter ces articles les a déjà rencontrés. Sinon, une description courte de cette terminologie se trouve dans la section 5.1.3 de [**32**].

possibilité d'obtenir une accélération exponentielle dans certains cas par rapport au meilleur algorithme classique. Le fait que dans certains cas on obtienne une accélération exponentielle a créé un grand intérêt pour ce qu'on appelle l'apprentissage automatique quantique à base d'algèbre linéaire (*QBLAS*) : plusieurs algorithmes qui utilisent HHL comme sous-routine dans un modèle d'apprentissage automatique ont été développés.

HHL est généralement interprété de façon erronée comme étant capable de résoudre des systèmes de n équations en $O(\log n)$ étapes. Cependant, l'applicabilité de HHL à un système d'équations donné est entravée par plusieurs conditions [**1**].

— HHL n'inclut pas de sous-routine pour encoder le vecteur $\mathbf{b} = (b_1, ..., b_n)$ dans un état quantique [13] $|\mathbf{b}\rangle = \sum_{i=1}^{n} b_i |i\rangle$. Donc, même si HHL pouvait résoudre le problème $\mathbf{Ax} = \mathbf{b}$ pour un vecteur $\mathbf{b}$ en temps $O(\log n)$, il faut une manière d'encoder $\mathbf{b}$ sur un support quantique. Si encoder $\mathbf{b}$ ne se fait qu'en temps $\omega(\log n)$, l'accélération de HHL est dominée par le temps d'encodage. De plus, le problème existe aussi lors du «décodage», c.-à-d. lorsqu'on extrait une solution classique de l'état quantique. La sortie de HHL est une superposition $|\mathbf{x}\rangle = \sum_{i=1}^{n} x_i |i\rangle$. Alors, si on souhaite observer tous les éléments de $\mathbf{x}$, il faut répéter l'algorithme au moins n fois.

— HHL nécessite que la matrice $\mathbf{A}$ soit *éparse* : le temps de calcul de HHL est proportionnel au nombre d'éléments non nuls de $\mathbf{A}$. Donc, pour préserver l'accélération, la matrice $\mathbf{A}$ doit posséder au plus $O(\log n)$ éléments non nuls [14].

13. On suppose que les vecteurs b et x sont déjà normalisés.

14. Il existe aussi d'autres classes spéciales de matrices pour lesquelles il est possible de préserver l'accélération.

Définition 35 (Conditionnement d'une matrice[**1**])**.** Le *conditionnement* $\mathcal{K}$ d'une matrice $\mathbf{A}$ est la valeur absolue du rapport entre la plus grande valeur propre λ_{max} et la plus petite valeur propre λ_{min} de la matrice $\mathbf{A}$:

$$\mathcal{K} = \frac{\lambda_{max}}{\lambda_{min}}$$

— L'accélération de HHL est aussi limitée par le conditionnement $\mathcal{K}$ = de la matrice $\mathbf{A}$. Le temps de calcul est linéairement proportionnel à $\mathcal{K}$. Donc, $\mathcal{K}$ doit être au plus logarithmique dans n pour préserver l'accélération de HHL.

Une belle perspective sur HHL provient du même article d'Aaronson [**1**] qui présente les trois points précédents : il note que «nous pourrions voir HHL moins comme un algorithme quantique en soi que comme un gabarit pour d'autres algorithmes quantiques. On remplit le gabarit en montrant comment préparer $|\mathbf{b}\rangle$, appliquer $e^{-i\mathbf{A}t}$, et mesurer $|\mathbf{x}\rangle$ dans un cas spécifique d'intérêt, et ensuite on analyse soigneusement les performances résultantes par rapport à celle du meilleur algorithme classique connu pour ce cas».

La présentation de HHL n'est pas reliée au reste de ce mémoire, mais elle sert plutôt d'avertissement : il n'est pas possible d'obtenir une accélération exponentielle dans la résolution d'un système d'équations avec l'algorithme HHL pour un système d'équations arbitraire. Cependant, décortiquer les implications de ces prérequis est un travail non trivial [**1**]. Il peut donc y avoir beaucoup de battage médiatique (*hype*) entourant un algorithme avec une applicabilité limitée, et la loi de Brandolini [**14**] s'applique souvent aux articles d'apprentissage automatique quantique qui en promettent trop.

4.2 Les circuits quantiques paramétrés

Considérons une architecture quantique (définition 7). Si on représente les transformations des portes de cette architecture par un ensemble de paramètres, on obtient un *circuit quantique paramétré* : l'architecture du circuit est fixe, mais les m portes quantiques du circuit sont paramétrées. On représente ces transformations par un tuplet de paramètres $\boldsymbol{\theta}$.

Définition 36 (Circuit quantique paramétré)**.** Un *circuit quantique paramétré* est un circuit quantique $C_{\boldsymbol{\theta}}$ dont certaines portes sont spécifiées par un tuplet de paramètres $\boldsymbol{\theta}$.

Il est important de se rappeler qu'on suppose que l'architecture d'un circuit quantique est fixe durant un algorithme : les emplacements des portes ne changent pas, mais les transformations effectuées par les portes sont paramétrées. Puisque les portes quantiques d'un circuit quantique paramétré sont variables, un circuit quantique paramétré peut représenter une famille des circuits quantiques avec cette architecture.

La décomposition de certaines portes quantiques d'arité l en un circuit quantique composé de portes d'arité 2 requiert $\Omega(2^l)$ portes [**63**]. Cependant, les circuits quantiques que nous considérons sont composés de $O(poly(l))$ portes d'arité 2. Alors, même si toutes les portes d'un circuit quantique paramétré d'arité 2 sont paramétrées, le circuit ne peut représenter qu'un sous-ensemble des portes d'arité l qui agissent sur l qubits. Le sous-ensemble des portes quantiques d'arité l qu'un circuit quantique paramétré peut représenter dépend de son architecture. Il n'y a pas de cadre établi pour comparer les sous-ensembles que différentes architectures de circuits quantiques paramétrés peuvent représenter [**83**].

4.2.1 Les circuits quantiques paramétrés comme modèles d'apprentissage automatique quantique analogues aux réseaux de neurones classiques

Initialement proposés comme une généralisation de l'algorithme hybride de chimie quantique VQE [**70**], les circuits quantiques paramétrés ont depuis explosé en popularité, devenant un des modèles de choix en apprentissage automatique quantique en raison de leur simplicité, de leur analogie simple aux réseaux de neurones classiques et de leur compatibilité avec les appareils NISQ [**77**]. On peut considérer un circuit quantique paramétré comme un modèle d'apprentissage automatique où l'optimisation des paramètres est effectuée par un optimiseur classique [**8**].

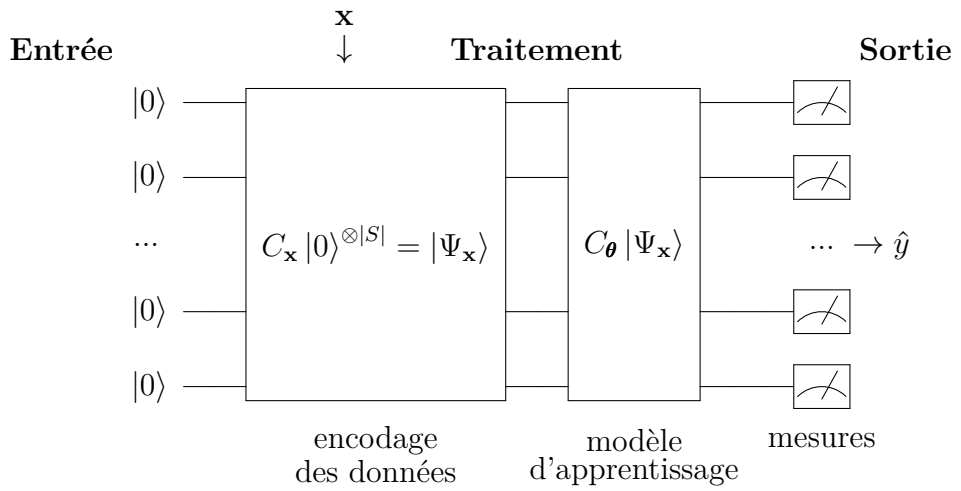

Figure 4.1 – Survol d'un circuit quantique paramétré représentant un modèle d'apprentissage automatique quantique (basé sur la figure 1 de [**2**]). Les données d'entrée sont encodées sur un support quantique $|\Psi_{\mathbf{x}}\rangle$ à travers le circuit quantique $C_{\mathbf{x}}$. Ensuite, les qubits qui encodent les données sont traités par le modèle d'apprentissage quantique $C_{\boldsymbol{\theta}}$. Finalement, on effectue des mesures sur les qubits en sortie $C_{\boldsymbol{\theta}}|\Psi_{\mathbf{x}}\rangle$ pour retirer de l'information classique $\hat{y}$ [15]. Cette information classique sert à l'optimisation des paramètres $\boldsymbol{\theta}$ du circuit quantique paramétré.

15. Dans la section 4.3, nous supposerons que la fonction de perte est encodée dans le circuit.

Pour utiliser un circuit quantique paramétré comme modèle d'apprentissage automatique, il faut :

(1) **encoder** les données classiques $\mathbf{x}$ sur un support quantique à l'aide d'un circuit quantique $C_{\mathbf{x}}$;

(2) **traiter** les données encodées avec les portes paramétrées du circuit quantique paramétré $C_{\boldsymbol{\theta}}$;

(3) Effectuer des **mesures** sur les données traitées pour extraire de l'information classique.

Plusieurs scientifiques considèrent les circuits quantiques paramétrés comme étant analogues aux réseaux de neurones classiques [**2, 8, 16, 29, 74, 77, 83, 93**]. Cependant, il existe deux différences majeures.

Premièrement, l'ignorance d'un algorithme efficace de rétropropagation pour les circuits quantiques a un impact sur le nombre d'opérations nécessaires pour optimiser les paramètres d'un circuit quantique paramétré. Deuxièmement, nous supposons que les circuits quantiques paramétrés sont composés de portes d'arité 2, tandis que chaque neurone dans un réseau de neurones peut avoir une arité de sortie égale au nombre de neurones dans la couche précédente et une arité de sortie égale au nombre de neurones dans la couche subséquente (voir figure 3.1).

Considérons le cadre de l'apprentissage supervisé vu dans la section 3.1. Avec une distribution inconnue D d'où les données $S = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)\}$ sont pigées, on cherche à découvrir les valeurs des paramètres $\boldsymbol{\theta}$ d'un modèle $h(\boldsymbol{\theta})$ qui associe les entrées aux sorties :

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}} \mathbb{E}_{(\mathbf{x},y)\sim D}[\ell(h(\boldsymbol{\theta}; \mathbf{x}), y)]$$

Considérons l'ensemble de tous les circuits quantiques avec une architecture spécifique. Tous les circuits dans cet ensemble se différencient uniquement par les transformations effectuées par leurs portes. Cependant, les emplacements des portes sont tous les mêmes. Chaque membre de cet ensemble peut être représenté par un circuit quantique paramétré avec la même architecture que les circuits dans l'ensemble en instanciant les valeurs des paramètres de ses portes si on suppose que chaque circuit dans cet ensemble a au moins un ensemble de valeurs des paramètres qui y correspond. On peut alors considérer ce circuit quantique paramétré comme un modèle d'apprentissage automatique qui représente la classe d'hypothèses de tous les circuits quantiques avec cette architecture.

Puisqu'on a généralement seulement accès à un ensemble de données S de taille finie, on se base sur le principe de minimisation du risque empirique $\hat{h}_S(\hat{\boldsymbol{\theta}})$ (Définition 26) :

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{|\boldsymbol{\theta}|}} \sum_{i}^{|S|} [\ell(h(\boldsymbol{\theta}; \mathbf{x}_i), y_i)]$$

On cherche donc un paramétrage du circuit quantique paramétré qui minimise la fonction de perte ℓ sur l'ensemble de données S dans le but de trouver un circuit quantique qui retourne en sortie la meilleure approximation possible de la cible y_i lorsqu'il reçoit en entrée le point de données associé $\mathbf{x}_i$ pour tout $(\mathbf{x}_i, y_i) \sim D$. Cette formulation correspond directement à la formulation d'un problème d'apprentissage automatique supervisé où le modèle $h(\boldsymbol{\theta})$ est un circuit quantique paramétré.

4.2.1.1 Un survol de l'encodage des données dans un état quantique

Jusqu'à maintenant, nous avons évité de mentionner l'encodage des données classiques dans un état quantique. Dans cette sous-section, nous survolons brièvement cet aspect des circuits quantiques paramétrés en tant que modèles d'apprentissage automatique. L'analyse

de l'impact de cet aspect sur la performance d'un circuit quantique paramétré comme modèle d'apprentissage automatique est un domaine de recherche actif et pertinent, mais il dépasse l'étendue de ce travail.

L'encodage des données classiques sur un support quantique spécifie comment des données classiques sont représentées par des états quantiques.

Définition 37 (Fonction d'encodage quantique (remaniée, puis traduite à partir de [**78**]))**.** Soit $V = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ un ensemble de données classiques. Une *fonction d'encodage quantique* est une fonction e qui associe chaque vecteur $\mathbf{v} \in V$ à un état quantique $|\Psi_{\mathbf{v}}\rangle$.

Par exemple, on pourrait encoder une séquence binaire x dans un état quantique à travers la base de calcul des états quantiques :

$$e(x) = |x\rangle$$

$$\text{e.g. } e(10) = |1\rangle \otimes |0\rangle = |10\rangle = 0\,|00\rangle + 0\,|01\rangle + 1\,|10\rangle + 0\,|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Cette fonction d'encodage est généralement connue sous le nom d'encodage *de base* [**79**].

On peut également encoder les valeurs d'un vecteur classique normalisé directement dans les amplitudes d'un état quantique.

Considérons le vecteur $(\frac{-1}{\sqrt{2}}, 0, \frac{1}{2}, \frac{1}{2})$:

$$e(x) = \sum_{i=0}^{n-1} x_i \, |i\rangle$$

$$\text{e.g. } e(\frac{-1}{\sqrt{2}}, 0, \frac{1}{2}, \frac{1}{2}) = \frac{-1}{\sqrt{2}} \, |00\rangle + 0 \, |01\rangle + \frac{1}{2} \, |10\rangle + \frac{1}{2} \, |11\rangle = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

Cet encodage est connu sous le nom d'encodage *d'amplitude* [**79**]. Le vecteur qu'on encode doit être normalisé auparavant selon la norme euclidienne pour que l'état quantique qui encode les données soit légitime, c'est-à-dire qu'il existe sur l'hypersphère unitaire d'un espace d'Hilbert.

Un sommaire de quelques fonctions d'encodages est repris de [**79**] et traduit dans le tableau 4.2.

Données classiques	**Propriétés**	**États quantiques**
Encodage de base		
$\mathbf{b} = [b_1, \ldots, b_n], b_i \in \{0,1\}$	$\mathbf{b}$ encode $\mathbf{x} \in \mathbb{R}^n$ en binaire	$\lvert\mathbf{x}\rangle = \lvert b_1, \ldots, b_n\rangle$
Encodage d'amplitude		
$\mathbf{x} \in \mathbb{R}^{2^n}$	$\sum_{i=1}^{2^n} \lvert x_i \rvert^2 = 1$	$\lvert\Psi_\mathbf{x}\rangle = \sum_{i=1}^{2^n} x_i \lvert i\rangle$
$A \in \mathbb{R}^{2^n \times 2^m}$	$\sum_{i=1}^{2^n} \sum_{j=1}^{2^m} \lvert a_{ij} \rvert^2 = 1$	$\lvert\Psi_A\rangle = \sum_{i=1}^{2^n} \sum_{j=1}^{2^m} a_{ij} \lvert i\rangle \lvert j\rangle$
Encodage *Qsample*		
$p(\mathbf{x}), \mathbf{x} \in \{0,1\}^n$	$\sum_\mathbf{x} p(\mathbf{x}) = 1$	$\sum_\mathbf{x} \sqrt{p(\mathbf{x})} \lvert\mathbf{x}\rangle$

Tableau 4.2 – Extrait et traduction du tableau 3.5 de [**79**]. Ce tableau présente un sommaire de quelques différents encodages de données classiques sur des supports quantiques.

4.3 L'optimisation itérative de circuits quantiques paramétrés

Cette section considère différents algorithmes d'optimisation itératifs pour optimiser les paramètres d'un circuit quantique paramétré. Ces méthodes requièrent l'estimation d'espérances des mesures du circuit avec différentes valeurs des paramètres et peuvent parfois nécessiter des modifications à l'architecture du circuit quantique paramétré qu'on optimise. On suppose sans perte de généralité que la fonction de perte est encodée dans un circuit quantique non paramétré qu'on attache au circuit quantique paramétré : l'espérance de la sortie de ce circuit est une espérance de la valeur de la fonction de perte. La minimisation cette espérance correspond donc à l'optimisation des paramètres du circuit.

On compare quelques algorithmes d'optimisation itératifs de circuits quantiques paramétrés selon trois aspects :

— Est-ce que la méthode d'optimisation requiert la modification de l'architecture du circuit quantique paramétré ?

— Est-ce que l'algorithme d'optimisation fonctionne pour tout paramétrage possible des portes du circuit, ou est-ce que l'algorithme d'optimisation requiert un paramétrage spécifique pour chaque porte ?

— Combien d'espérances faut-il estimer pour utiliser la méthode d'optimisation ?

4.3.1 L'évaluation de gradients analytiques sur un ordinateur quantique

En apprentissage profond, la descente de gradient sous-tend la grande majorité des algorithmes d'optimisation [**32**]. Plusieurs recherches ont été effectuées pour développer des

méthodes de calcul des dérivées partielles de la sortie d'un circuit quantique paramétré par rapport aux paramètres des portes paramétrées du circuit [**29, 37, 54, 57, 77**].

Notamment, [**76**] présente la méthode de *décalage de paramètre* : dans un circuit quantique paramétré, la dérivée partielle de la sortie du circuit par rapport à un paramètre réel d'une porte quantique paramétrée d'arité 1 peut être estimée sans biais à partir d'au plus 4 espérances de résultats de circuits quantiques paramétrés similaires au circuit quantique paramétré qu'on essaie d'optimiser. Dans le cas général d'une porte quantique d'arité 1, les 4 espérances nécessaires ne peuvent pas être calculées avec la même architecture que celle du circuit quantique paramétré qu'on essaie d'optimiser : la porte quantique doit être décomposée en un circuit de petite taille et requiert l'utilisation d'un qubit ancillaire. La méthode d'optimisation s'applique au cas général d'un circuit quantique : la méthode de décalage de paramètre peut calculer la dérivée partielle tant que le paramètre est réel. Puisque chaque dérivée partielle requiert l'estimation de 4 espérances de résultats de circuits quantiques, une itération d'optimisation pour tous les paramètres d'un circuit quantique paramétré avec $|\boldsymbol{\theta}|$ paramètres requiert l'estimation de $4|\boldsymbol{\theta}|$ espérances.

Considérons un algorithme d'optimisation qui utilise la méthode de décalage de paramètre à chaque itération pour calculer les dérivées partielles de la sortie du circuit par rapport à tous les paramètres d'un circuit quantique paramétré. Le nombre d'opérations pour calculer toutes les dérivées partielles de la sortie du circuit quantique paramétré par rapport aux paramètres du circuit est quadratique dans $|\boldsymbol{\theta}|$, tandis que pour un réseau de neurones classiques, l'algorithme de rétropropagation 3.3.2 requiert un nombre d'opérations linéaire dans $|\boldsymbol{\theta}|$ pour calculer toutes les dérivées partielles de la sortie du réseau par rapport aux paramètres. De plus, le calcul d'une dérivée partielle avec la méthode de décalage de paramètre nécessite l'estimation d'espérances de résultats de circuit quantiques, contrairement au calcul

classique de dérivées partielles. On doit donc mesurer la sortie du circuit assez de fois pour obtenir une confiance désirée dans l'estimation de la dérivée partielle [**6**].

4.3.2 *Rotosolve* et *Rotoselect*

Rotosolve [**65**] est un algorithme d'optimisation de circuit quantique paramétré qui ne modifie pas l'architecture du circuit et qui estime trois espérances du résultat du circuit quantique paramétré pour chaque paramètre. Cependant, la méthode d'optimisation ne s'applique pas au cas général d'un circuit quantique paramétré. Rotosolve ne s'applique qu'aux circuits quantiques paramétrés où les portes paramétrées sont des portes de rotation à un qubit.

Considérons un circuit quantique paramétré $C_{\boldsymbol{\theta}}$ qui agit sur un registre $|\Psi\rangle$ de l qubits et qui est constitué de $|\boldsymbol{\theta}|$ portes paramétrées $(U_i)_{i=1}^{|\boldsymbol{\theta}|}$ et d'un nombre arbitraire de portes fixes. Chaque porte paramétrée est une porte d'arité 1 de la forme $U_d = \exp\left(-i\frac{\theta_i}{2}\mathbf{H}_i\right)$ où $\mathbf{H}_i$ est une matrice hermitienne et unitaire et $\theta_i \in (-\pi, \pi]$ est un angle de rotation. On peut donc représenter les paramètres du circuit comme un vecteur $\boldsymbol{\theta} \in (-\pi, \pi]^{|\boldsymbol{\theta}|}$ et l'ensemble des matrices hermitiennes et unitaires comme un vecteur de matrices $\boldsymbol{H} = [\mathbf{H}_1,...,\mathbf{H}_{|\boldsymbol{\theta}|}]$. Le problème d'optimisation est alors de trouver $\boldsymbol{\theta^*} = \arg\min_{\boldsymbol{\theta}} \langle C_{\boldsymbol{\theta},\boldsymbol{H}} |\Psi\rangle\rangle$ où $\langle\cdot\rangle$ représente l'espérance de la sortie du circuit (section 2.1.3). *Rotosolve* calcule l'angle optimal θ_i^* pour la porte U_i en gardant le reste du circuit fixe, tandis que $\mathbf{H}_i$ est traité comme un hyperparamètre. Ce calcul nécessite l'estimation de trois espérances $\langle C_{\boldsymbol{\theta},\boldsymbol{H}} |\Psi\rangle\rangle$ avec des valeurs différentes de θ_i. Chaque angle est optimisé pour chaque porte dans $(U_i)_{i=1}^{|\boldsymbol{\theta}|}$ de manière séquentielle jusqu'à ce qu'un critère d'arrêt soit atteint. Alors, si le circuit contient $|\boldsymbol{\theta}|$ portes de rotation paramétrée, une itération d'optimisation requiert l'estimation de $3|\boldsymbol{\theta}|$ espérances. Puisque l'angle θ_i^* n'est optimal que lorsque le reste du circuit est fixe, l'algorithme peut modifier tous les paramètres plusieurs fois avant d'atteindre un critère d'arrêt.

Rotoselect[**65**] est un algorithme similaire, mais qui choisit conjointement θ_i^* et $\mathbf{H}_i^*$ en calculant l'angle optimal pour plusieurs possibilités de $\mathbf{H}_i$. Notons que le choix de $\mathbf{H}_i$ n'affecte pas l'architecture du circuit tel que nous l'avons définie (définition 7).

4.3.3 *Baqprop*

Un article [**88**] publié sur arXiv, mais qui n'a pas encore été évalué par des pairs, propose *Baqprop*, une approche très intéressante à la descente de gradient dans les circuits quantiques paramétrés. En simulant les paramètres réels du circuit quantique paramétré avec l'ordinateur quantique, *Baqprop* utilise le transfert de phase (*phase kickback*) pour obtenir de l'information sur les dérivées partielles du circuit d'une manière similaire à la rétropropagation. Une description précise de la complexité de calcul et du comportement de l'algorithme proposé va au-delà de la compréhension de la technique utilisée. Bien que nous sommes enthousiastes au sujet de l'article, nous pensons qu'un travail supplémentaire est nécessaire pour valider certains des aspects spécifiques de l'approche et pour fournir une analyse plus précise du nombre d'opérations de l'approche.

L'article présente deux approches à l'optimisation des circuits quantiques paramétrés. Une approche utilise un algorithme d'optimisation itératif où chaque itération requiert plusieurs mesures pour estimer la sortie du circuit avant de modifier les paramètres du circuit quantique paramétré. L'autre approche ne repose pas sur un algorithme d'optimisation itératif comparable à la première approche : le processus d'optimisation est entièrement cohérent et produit à la fin une fonction d'onde sur l'ensemble des valeurs des paramètres. Il est possible par la suite d'approximer les valeurs classiques des paramètres en estimant la sortie du circuit avec plusieurs mesures.

Malheureusement, il y a trop de facteurs inconnus concernant cette deuxième approche pour pouvoir effectuer une comparaison rigoureuse avec l'approche basée sur une optimisation itérative de paramètres réels classiques.

4.3.4 Une approche simple et robuste à l'apprentissage quantique

Dans cette section, nous proposons une approche simple et robuste permettant l'apprentissage des paramètres d'un circuit quantique paramétré. L'approche est un algorithme d'optimisation itératif qui nécessite l'estimation d'une seule espérance, qui ne modifie pas le circuit et qui fonctionne pour tout circuit quantique paramétré où les paramètres des portes paramétrées sont des nombres réels. Les paramètres sont optimisés un par un. L'approche naïve, mais robuste que nous décrivons ici teste l'impact d'une modification directement sur la fonction de perte.

Dans les réseaux de neurones, l'étape d'optimisation principale consiste à ajuster légèrement tous les paramètres du réseau. La modification de tous les paramètres du réseau requiert un nombre d'opérations linéaire dans le nombre de paramètres du réseau en raison de l'algorithme de rétropropagation (section 3.3.2). Dans le cas des circuits quantiques, où aucun équivalent n'est connu à la rétropropagation, on peut obtenir plusieurs algorithmes d'optimisation qui demandent un nombre d'opérations quadratique dans le nombre de paramètres.

Considérons l'algorithme d'optimisation basé sur la méthode de décalage de paramètre (section 4.3.1). Cet algorithme exige un nombre d'opérations quadratique dans le nombre de paramètres pour calculer les dérivées partielles de la sortie d'un circuit quantique paramétré par rapport à ses paramètres. Le signe de la dérivée partielle donne la bonne direction d'optimisation et sa valeur multiple la taille de la modification effectuée aux paramètres.

Cependant, la dérivée partielle de la sortie d'un circuit par rapport à un paramètre ne fournit pas directement une valeur optimale du paramètre : la dérivée partielle fournit de l'information sur la direction et l'amplitude de la modification localement optimale, mais il faut choisir une taille de pas λ et modifier les paramètres du circuit quantique paramétré de manière itérative avec un algorithme de descente de gradient. Avec un algorithme comme la rétropropagation qui calcule toutes les dérivées partielles des paramètres d'un réseau de neurones avec un nombre d'opérations linéaire dans le nombre de paramètres, un algorithme de descente de gradient est justifié. Dans l'absence d'un algorithme aussi efficace que la rétropropagation pour calculer les dérivées partielles des paramètres d'un circuit quantique paramétré, il n'est pas du tout évident que la modification engendrée par la descente de gradient est préférable à une modification qui est directement testée en estimant la valeur de la fonction de perte après cette modification.

On peut aussi envisager l'utilisation de *Rotosolve* (section 4.3.2). Avec trois espérances, *Rotosolve* calcule l'angle optimal pour une porte quantique de rotation à un paramètre en gardant le reste du circuit fixe : la modification de tous les paramètres d'un circuit quantique paramétré exige donc un nombre d'opérations quadratique dans le nombre de paramètres. Notons cependant que, si on modifie la valeur d'un angle, alors la valeur optimale des autres angles peut changer. Notons aussi que le comportement de cet algorithme durant l'optimisation d'un circuit quantique paramétré en tant que modèle d'apprentissage automatique dépend de l'encodage des données. Si le circuit traite un exemple à la fois, alors l'algorithme retourne un angle qui est optimal pour ce seul exemple. Il n'est pas du tout évident que prendre la moyenne d'un ensemble de ces angles calculés pour des exemples différents retourne un angle qui est optimal en moyenne sur l'ensemble de ces exemples.

Baqprop (section 4.3.3) est une approche prometteuse pour optimiser les paramètres d'un circuit quantique paramétré par descente de gradient. Cependant, cette approche a pour le moment trop de facteurs inconnus pour qu'on puisse en faire une analyse détaillée et la comparer avec les autres algorithmes d'optimisation itératifs que nous considérons.

4.3.4.1 Un algorithme d'optimisation basé sur l'escalade

Une manière d'optimiser les paramètres $\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}$ d'un circuit quantique paramétré est par escalade (algorithme 2) : l'algorithme choisit un paramètre $\theta_i \in \boldsymbol{\theta}$ et un signe $d \in \{-1,+1\}$ au hasard. On ajoute ou on soustrait une modification légère $\lambda > 0$ à θ_i pour obtenir $\theta'_i = \theta_i + d\lambda$. La valeur de la fonction de perte est estimée pour θ_i et pour θ'_i en mesurant la sortie du circuit τ fois (algorithme 1), ce qui demande d'avoir un ordinateur quantique. Si la valeur estimée de la fonction de perte n'est pas plus grande pour θ'_i que pour θ_i, alors on modifie le paramètre : $\theta_i \leftarrow \theta'_i$. Sinon, on rejette la modification. Pour compléter cet algorithme simple, on n'a qu'à répéter les étapes précédentes.

Algorithme 1 : Estimer la perte

Arguments : $C_{\boldsymbol{\theta}}$ un circuit quantique paramétré spécifié par les paramètres $\boldsymbol{\theta}$; $|\Psi_{\mathbf{x}_i}\rangle$ un état quantique qui encode $\mathbf{x_i}$; τ un nombre de répétitions ; ℓ une fonction de perte;
Initialiser $c \leftarrow 0$;
pour τ *fois* **faire**
    $m \leftarrow$ résultat de la mesure de $C_{\boldsymbol{\theta}}(|\Psi_{\mathbf{x}_i}\rangle)$;
    $c \leftarrow c + \ell(m)$;
fin
$c \leftarrow c/k$;
retourner c

4.3.4.2 Modifications possibles de l'algorithme

Plusieurs modifications peuvent être apportées à l'algorithme pour changer son comportement. Dans cette section, nous en présentons quelques-unes.

Algorithme 2 : Apprentissage quantique par escalade

Arguments : λ la longueur du pas ; τ un nombre de répétitions ; E un algorithme pour estimer la perte ; un critère d'arrêt ;
Initialiser $\boldsymbol{\theta} = [\theta_1,...,\theta_{|\boldsymbol{\theta}|}]$ de façon aléatoire;
$c \leftarrow E(\boldsymbol{\theta},k)$;
tant que *critère d'arrêt n'est pas atteint* **faire**
    choisir un indice $i \in \{1,..., |\boldsymbol{\theta}|\}$;
    choisir un signe $d \in \{-1, +1\}$;
    $\boldsymbol{\theta}' \leftarrow [\theta_1,..., \theta_i + d\lambda, ..., \theta_{|\boldsymbol{\theta}|}]$;
    $c' \leftarrow E(\boldsymbol{\theta}',\tau)$;
    si $c' \leq c$ **alors**
        $c \leftarrow c'$;
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$;
    fin
fin

Un élément que nous avons évité de spécifier dans nos algorithmes est le calcul de la fonction de perte ℓ. Rappelons que nous supposons que la fonction de perte est encodée dans le circuit quantique. Puisque le calcul de la fonction de perte se fait généralement sur des exemples qui proviennent d'un ensemble de données, l'algorithme 1 peut être modifié pour prendre la moyenne de la fonction de perte sur plusieurs exemples.

Nous pourrions aussi altérer l'algorithme 2 pour qu'il modifie plusieurs paramètres simultanément durant chaque itération de l'optimisation au lieu de modifier un seul paramètre.

Une autre modification possible tente d'inverser le signe d si la modification ne diminue pas la valeur estimée de la fonction de perte. Si on traverse les paramètres de manière séquentielle au lieu d'aléatoire, cette modification permet d'explorer des possibilités de régimes pour les valeurs de λ : si aucune modification ne diminue la valeur estimée de la fonction de perte, il se peut qu'une diminution de λ permette à l'algorithme de continuer à optimiser le circuit.

Il est aussi possible de persévérer dans la même direction lorsque l'algorithme trouve une modification : si l'addition de $d\lambda$ au paramètre θ_i a diminué la valeur estimée de la fonction

de perte, répéter cette modification pourrait également être bénéfique. Il est aussi possible d'envisager des modifications de λ si on continue à modifier θ_i.

Une autre modification intéressante implique de spécifier une distribution pour l'initialisation des paramètres du circuit quantique paramétré. Dans le cas des réseaux de neurones profonds classiques, l'initialisation peut avoir une influence importante sur l'apprentissage et, ultimement, la performance du modèle [**31**].

Chapitre 5

Une architecture universelle de circuit quantique

Dans ce chapitre, nous présentons une famille d'architectures quantiques universelles basée sur la répétition de circuits quantiques paramétrés avec l'architecture d'un réseau de permutation. Les membres de cette famille sont universels dans le sens qu'un circuit quantique paramétré avec une profondeur dans l'ordre exact de $p \lg l$ dont les portes sont organisées selon l'architecture universelle indexée par l et p peut représenter n'importe quel circuit quantique d'arité 2, de largeur l et de profondeur p . Dans le cas de l'apprentissage des circuits quantiques paramétrés, l'utilisation d'un circuit avec une architecture universelle remplace le problème combinatoire du choix de la structure du circuit par le problème d'optimisation continue des paramètres des portes du circuit quantique paramétré avec l'architecture universelle appropriée, un problème qui se prête mieux aux techniques d'apprentissage automatique comme la descente de gradient.

Cette architecture universelle par permutation est composée de portes quantiques paramétrées d'arité 2. Chaque module du circuit est composé de portes arrangées selon la structure d'un réseau de permutation, permettant de réarranger l qubits en profondeur $O(\lg l)$.

En répétant ces modules p fois, on peut plonger un circuit de profondeur p d'arité 2 avec une architecture arbitraire dans l'architecture universelle. Sans perte de généralité, l'architecture universelle est présentée pour les circuits dont la largeur l est une puissance de 2.

L'établissement d'une architecture universelle par répétition de réseaux de permutation n'est pas une nouvelle idée [**85**]. Dans [**11**], les auteurs établissent des architectures quantiques de largeur $l^2 p$ et de profondeur p universelles pour des circuits composés de familles très restreintes de portes avec des portes non bornées. On peut décomposer les portes non bornées en circuits de profondeur logarithmique pour obtenir des architectures de profondeur $p \lg l$. Par contre, ces résultats d'universalité ne tiennent que pour des familles de circuits constitués de portes de certains sous-ensembles très limités, alors que notre objectif est de pouvoir plonger un circuit constitué de portes d'arité 2 arbitraires dans un circuit avec une architecture fixe.

5.1 Une porte quantique d'arité 2 paramétrée

L'élément de base de notre architecture est une manière de paramétrer toute porte quantique à deux qubits avec des portes quantiques d'arité 1 à un paramètre et des portes $CNOT$.

Le paramétrage choisi doit être général : pour toute porte quantique d'arité 2, au moins un paramétrage doit y correspondre. Pour faciliter l'optimisation, la porte quantique paramétrée doit rester légitime pour des paramètres réels arbitraires. Si on veut modifier une porte quantique, l'unitarité doit être préservée pour que la porte reste légale. Donc, la modification des paramètres de la porte doit se faire de manière arbitraire sans que la porte perde son unitarité. Notamment, il n'est pas possible de modifier directement les éléments d'une matrice unitaire et de préserver l'unitarité de cette matrice sans choisir soigneusement

chaque modification. Considérer les éléments d'une matrice unitaire $U^{4\times4}$ ne répond donc pas à nos critères.

L'ensemble des portes quantiques qui agissent sur l qubits forme une variété de dimension $l^2 - 1$ qui est un sous-ensemble des matrices complexes. Quinze paramètres réels sont donc nécessaires pour spécifier l'ensemble des portes quantiques d'arité 2, contrairement aux 32 paramètres réels nécessaires dans le cas de matrices complexes générales 4 par 4.

Les auteurs de [**87**] présentent un circuit quantique paramétré qui réalise n'importe quelle porte quantique d'arité 2 avec 7 portes paramétrées d'arité 1 et de 3 portes CNOT (figure 5.1).

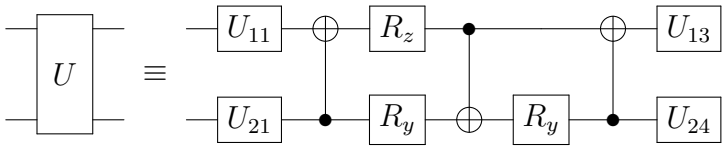

Figure 5.1 – Décomposition d'une porte quantique d'arité 2 en portes d'arité 1 et en portes CNOT (reproduction de la figure 7 de [**87**]).

Les portes R_y et R_z sont des portes quantiques à un seul paramètre dans $\mathbb{R}$:

$$R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2} \\ -\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \quad R_z(\alpha) = \begin{pmatrix} e^{i\frac{\alpha}{2}} & 0 \\ 0 & e^{-i\frac{\alpha}{2}} \end{pmatrix}$$

Les portes $U_{11}, U_{21}, U_{13}, U_{24}$ sont des portes quantiques arbitraires d'arité 1. Toute porte quantique d'arité 1 peut être décomposée en trois portes à un paramètre réel, ce qui est un nombre nécessaire et suffisant de paramètres [**7**] :

$$U^{2\times2}(\alpha, \theta, \beta) = R_z(\alpha)R_y(\theta)R_z(\beta)$$

Le circuit quantique paramétré présenté dans [**87**] implémente une porte quantique d'arité 2 arbitraire avec 15 paramètres, un nombre de paramètres nécessaire et suffisant. Ce circuit quantique a une largeur de 2, une profondeur de 11 et il est composé de 15 portes d'arité 1 à un paramètre et 3 portes CNOT si on suppose que les portes unitaires générales d'arité 1 ont été décomposées en trois portes unitaires à un paramètre.

5.2 Les réseaux de permutation

Les réseaux de permutation [**9, 19**] peuvent générer n'importe quelle permutation de l éléments données en entrée au réseau en choisissant de façon adéquate comment chaque porte du réseau agit.

Définition 38 (Réseau de permutation)**.** Un *réseau de permutation* est une grille en deux dimensions. Cette grille est composée de *fils* horizontaux qui contiennent des éléments, et de *couches* verticales contenant des portes qui peuvent permuter les valeurs des fils en entrée ou pas. Les éléments dans les fils procèdent à travers l'architecture de gauche à droite et chaque fil de l'architecture est connecté à au plus une porte par couche. Le réseau permet de retourner n'importe quelle permutation des éléments en entrée en spécifiant quelles portes doivent permuter leurs entrées.

Définition 39 (Réseau de permutation de Beneš)**.** Un *réseau de permutation de Beneš* [**9**] est un réseau de permutation avec l fils, des portes d'arité 2, et $2 \lg l - 1$ couches où chaque couche contient $l/2$ portes.

On quantise un réseau de permutation en remplaçant les éléments en entrée par des qubits et les portes qui peuvent permuter les éléments par des portes quantiques paramétrées qui agissent sur les qubits (section 5.1). L'architecture d'un tel circuit quantique est un *module de permutation quantique*.

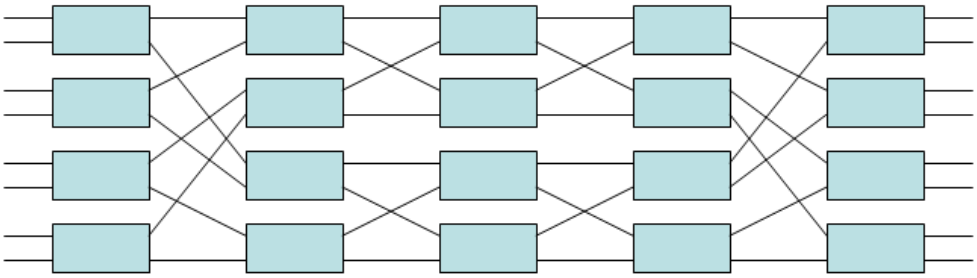

Figure 5.2 – Un réseau de permutation de Beneš [**9**] pour 8 éléments.

Définition 40 (Module de permutation quantique)**.** Un *module de permutation quantique* est réseau de permutation où les éléments en entrée sont des qubits et les portes qui agissent sur les éléments sont des portes quantiques.

Définition 41 (Module de permutation quantique de Beneš)**.** Un *module de permutation quantique de Beneš* (figure 5.3) est une architecture de circuit quantique paramétré avec un arrangement de portes identique à un réseau de permutation de Beneš. La largeur d'un module de permutation quantique de Beneš est donc l, tandis que sa profondeur est $2\lg l - 1$ et sa taille est $l \lg l - \frac{l}{2}$.

5.3 La famille d'architectures universelles et ses caractéristiques

Chaque module de permutation quantique de largeur l permet de réarranger l qubits dans un ordre arbitraire. Après chaque module de permutation quantique, nous insérons une couche de portes quantiques d'arité 2 entre les qubits voisins qu'on appelle *module de calcul*.

Définition 42 (Module de calcul)**.** Un *module de calcul* est une architecture de circuit quantique paramétré de largeur l et de profondeur 1 constituée de portes quantiques d'arité 2 qui agissent sur des paires de qubits voisins.

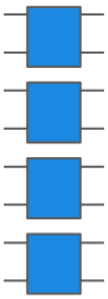

Figure 5.3 – Un module de calcul pour 8 éléments. Le module est constitué de portes quantiques arbitraires qui agissent sur des paires voisines de qubits.

Un circuit quantique paramétré dont l'architecture est un module de permutation quantique suivi d'un module de calcul peut simuler une couche de portes à deux qubits avec une architecture arbitraire. En répétant cette construction plusieurs fois, on obtient une architecture universelle par permutation.

Définition 43 (Architecture universelle par permutation)**.** Une *architecture universelle par permutation* est une architecture quantique universelle pour la famille $F_{l,p}$ de circuits quantiques d'arité 2, de largeur l et de profondeur p (définition 15). Cette architecture est composée de p modules de permutation quantiques et de calcul en alternance.

Définition 44 (Famille d'architectures universelles par permutation)**.** On note $P_{l,p}$ la famille d'architectures de largeur l (définition 14) composées de p modules de permutation quantiques et de calcul en alternance.

Théorème 5.3.1 (Universalité de l'architecture universelle par permutation)**.** *Soit la famille $F_{l,p}$ de circuits quantiques d'arité 2, de largeur l et de profondeur p (définition 15). Tout circuit quantique dans $F_{l,p}$ peut être obtenu à partir d'un circuit quantique dont l'architecture est n'importe quel élément de la famille $P_{l,p}$ en spécifiant correctement les portes de ce circuit.*

Démonstration. Considérons la première couche du circuit quantique C qu'on cherche à plonger dans l'architecture universelle. Spécifions les portes du premier module par permutation pour que les paires de qubits qui interagissent dans la première couche de C se retrouvent côte à côte. Ajoutons un module calcul et spécifions les portes de ce module comme étant les portes dans la première couche de C. Répétons cette séquence pour toutes les couches du circuit quantique qu'on cherche à plonger. L'architecture universelle par permutation est alors composée de p modules de permutation quantiques et de calcul en alternance pour un total de $2p$ modules. □

Définition 45 (Architecture universelle par permutation de Beneš)**.** Une *architecture universelle par permutation de Beneš* est une architecture universelle par permutation dont l'architecture de chaque module de permutation quantique est la même qu'un module de permutation quantique de Beneš.

Si on utilise le réseau de Beneš comme module de permutation quantique, on obtient immédiatement le résultat suivant :

Corollaire 5.3.2 (Universalité de l'architecture universelle par permutation de Beneš)**.** *Tout circuit quantique dans $F_{l,p}$ peut être obtenu à partir d'un circuit quantique dont l'architecture est l'architecture universelle par permutation de Beneš dans $P_{l,p}$. La taille de ce circuit quantique paramétré est $pl \lg l$, sa largeur est l et sa profondeur est $2p \lg l$.*

Chaque porte d'arité 2 est paramétrée par un circuit quantique fixe avec 7 portes paramétrées d'arité 1 et 3 portes CNOT (section 5.1). Notons que le paramétrage des portes des modules de calcul doit être général, mais le paramétrage des portes des modules de permutation quantiques doit au moins contenir les portes SWAP et Identité.

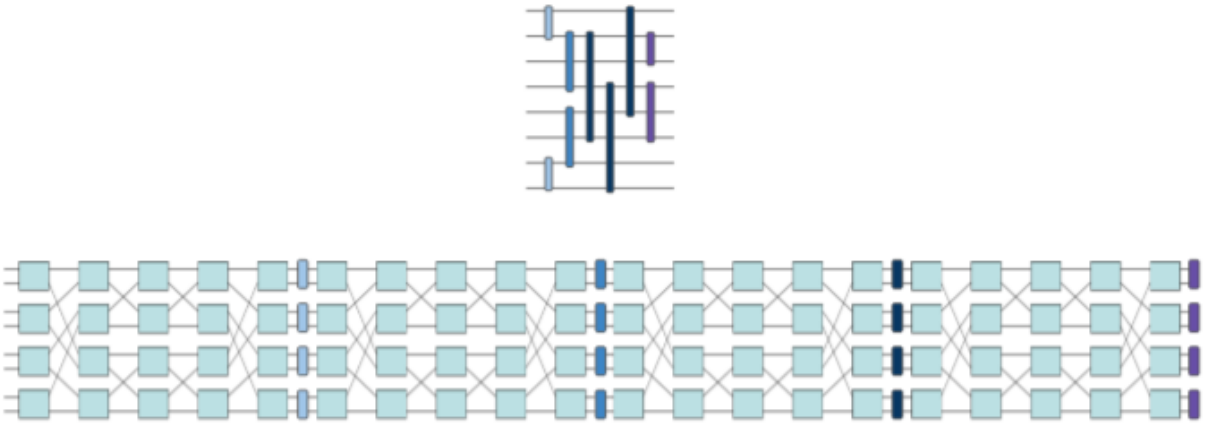

Figure 5.4 – Un exemple de plongement d'un circuit quantique arbitraire dans un circuit quantique avec l'architecture universelle par permutation de Beneš. Les portes des modules de permutation quantiques doivent permuter les qubits correctement. Toutes les portes sont d'arité 2 et agissent donc seulement sur les 2 qubits aux extrémités de chaque porte.

Corollaire 5.3.3 (Complexité du circuit quantique paramétré avec l'architecture universelle par permutation de Beneš)**.** *Un circuit quantique paramétré avec l'architecture universelle par permutation de Beneš dans $P_{l,p}$ où chaque porte quantique générale d'arité 2 est paramétrée par le circuit présenté dans la section 5.1 a une largeur de l et une profondeur de $22p \lg l$. Ce circuit contient $15pl \lg l$ portes quantiques d'arité 1 à un paramètre et $3pl \lg l$ portes CNOT.*

5.4 L'architecture universelle comme modèle d'apprentissage automatique

En combinant une architecture universelle par permutation dans $P_{l,p}$ avec un optimiseur classique, on obtient un modèle d'apprentissage automatique avec une classe d'hypothèses qui inclut tous les circuits quantiques d'arité 2, de largeur l et de profondeur p, peu importe leur architecture. L'utilisation d'un circuit quantique paramétré $C_{\boldsymbol{\theta}}$ avec une architecture universelle par permutation dans $P_{l,p}$ comme modèle d'apprentissage comporte un avantage majeur. Rappelons que l'erreur de généralisation est la somme de l'erreur d'estimation (définition 30) et d'approximation (définition 29). L'erreur d'estimation peut être diminuée en

optimisant les paramètres d'un modèle d'apprentissage. Cependant, l'erreur d'approximation est indépendante des valeurs des paramètres : l'erreur d'approximation ne change pas à moins que H ne change. Si un modèle spécifie H, alors ce modèle doit être modifié si on souhaite améliorer l'erreur d'approximation.

Optimiser le choix de modèle d'apprentissage automatique est un problème d'optimisation discret : les algorithmes d'optimisation itératifs que nous avons considérés ne s'appliquent pas à la sélection de H. L'optimisation de H est généralement beaucoup plus difficile comme problème que l'optimisation de $\boldsymbol{\theta}$. Dans le cas d'un circuit quantique paramétré avec une architecture naïve, l'optimisation de H revient à optimiser l'architecture du circuit.

Supposons qu'on essaie de trouver un circuit quantique C de largeur l, de profondeur p et d'arité 2. Si on se permet d'augmenter la profondeur du circuit légèrement — par exemple, de multiplier la profondeur par un facteur logarithmique si on utilise l'architecture universelle par permutation de Beneš —, on peut utiliser un circuit quantique paramétré avec une architecture universelle dans $P_{l,p}$ pour transformer le problème d'optimisation d'architecture du circuit en problème d'optimisation des paramètres du circuit. Les algorithmes d'optimisation itératifs présentés dans la section 4.3 peuvent alors attaquer ce problème. L'architecture universelle ne garantit pas qu'un algorithme d'optimisation itératif sera en mesure de trouver les valeurs des paramètres appropriées. Cependant, le problème d'optimisation passe tout de même du cadre discret au cadre continu, ce qui permet d'utiliser des méthodes d'optimisation continue. La minimisation de fonctions convexes est un bon exemple de l'avantage que le cadre continu peut avoir sur le cadre discret : la méthode de l'ellipsoïde dûe à Khachiyan [**48**] résout des problèmes d'optimisation linéaire en temps polynomial dans la taille de l'entrée, contrairement à la méthode du simplexe de Dantzig qui a une complexité dans le pire cas qui est exponentielle dans la taille de l'entrée [**20**].

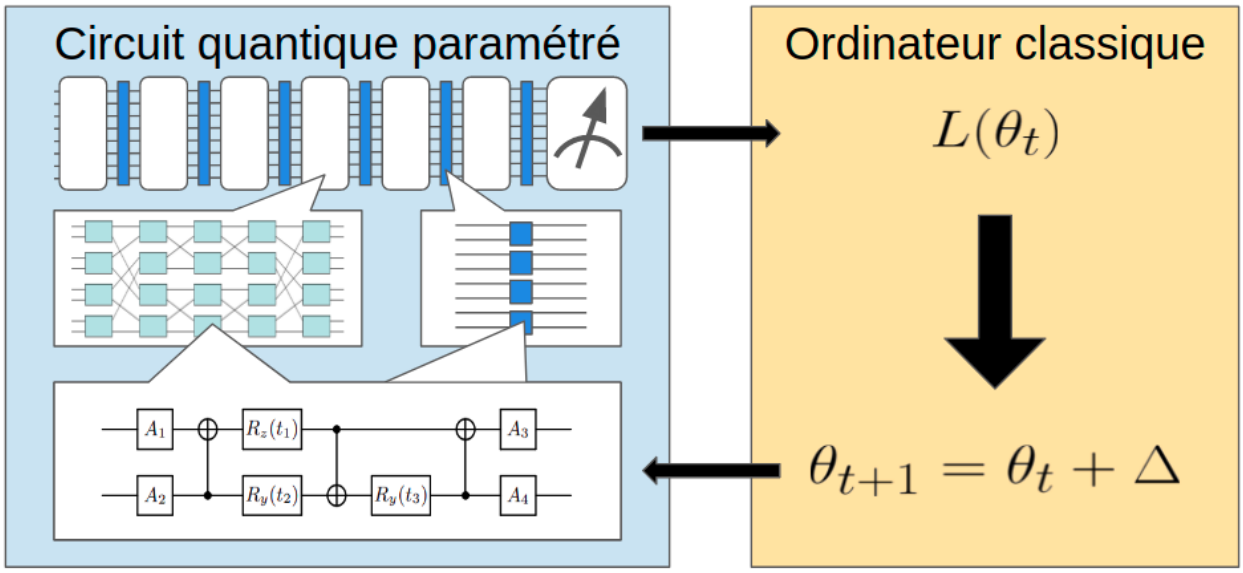

Figure 5.5 – Un modèle d'apprentissage à base de l'architecture universelle optimisée par un algorithme d'optimisation itératif tel que l'algorithme d'escalade (*hill climbing*) présenté dans l'algorithme 2.

Chapitre 6

Conclusions et directions futures

Ce que nous pouvons accomplir en travaillant dans les paradigmes de l'informatique quantique et de l'apprentissage automatique est une direction de recherche prometteuse en pleine croissance. L'informatique quantique se définit par opposition à l'informatique classique en tirant parti des phénomènes de la théorie quantique pour dépasser les limites du traitement de l'information classique. De manière complémentaire, les algorithmes d'apprentissage automatique se distinguent des algorithmes conventionnels en s'améliorant automatiquement avec des données. Dans ce mémoire, nous avons envisagé d'utiliser le paradigme de l'apprentissage automatique pour apprendre un circuit quantique. Ceci a conduit à considérer les circuits quantiques avec des portes régies par des paramètres classiques comme des modèles d'apprentissage automatique. L'exploration de cette problématique a mené à deux contributions et plusieurs directions futures de recherche.

6.1 Première contribution : Une méthode d'optimisation simple, mais robuste, pour les circuits quantiques paramétrés

La première contribution de ce mémoire est d'avoir établi un cadre de comparaison pour les différents algorithmes d'optimisation itératifs de circuits quantiques paramétrés comme modèles d'apprentissage automatique quantique. Les articles d'apprentissage automatique quantiques sont souvent écrits d'un point de vue de la physique qui ne tient pas compte de la complexité d'un algorithme, ce qui rend difficile la comparaison de l'efficacité de différents algorithmes d'optimisation. Alors, en premier lieu, la comparaison des nombres d'opérations des différents algorithmes d'optimisation itératifs a requis l'établissement d'un cadre commun. Tous ces algorithmes agissent de manière similaire : le circuit quantique paramétré est roulé plusieurs fois avec une valeur fixe des paramètres pour estimer l'espérance de la sortie du circuit, et les paramètres sont modifiés pour améliorer la valeur de cette espérance par rapport à une fonction objectif. Cette similarité conceptuelle nous permet de comparer ces algorithmes théoriquement.

On contraste ces algorithmes d'optimisation itératifs de circuits quantiques paramétrés selon trois aspects :

— Est-ce que le circuit original doit être modifié pour optimiser le circuit quantique paramétré ?

— Est-ce que l'algorithme d'optimisation fonctionne pour tout circuit quantique paramétré, ou est-ce que l'algorithme d'optimisation requiert un paramétrage spécifique pour chaque porte du circuit ?

— Combien d'espérances de mesures est-ce que l'algorithme d'optimisation en question requiert ?

Cette comparaison a mené à la conception d'un algorithme d'optimisation itératif simple, mais robuste basé sur l'algorithme d'escalade (*hill climbing*). Cet algorithme ne demande pas de modifier le circuit original, s'applique à n'importe quel circuit quantique paramétré par des nombres réels et ne requiert que l'estimation d'une espérance de mesures.

6.2 Deuxième contribution : Une famille d'architectures universelles par permutation de circuits quantiques paramétrés

La deuxième contribution de ce mémoire est l'établissement d'une famille d'architectures universelles par permutation pour des circuits quantiques d'architectures arbitraires. N'importe quel circuit quantique d'arité 2, de largeur l et de profondeur p peut être plongé dans un circuit quantique paramétré de largeur l et de profondeur $O(p \log l)$ si ce circuit quantique paramétré est connecté selon l'architecture universelle appropriée basée sur l'architecture du réseau de Beneš pour l éléments. Les aspects principaux de cette contribution sont :

— L'établissement d'une architecture universelle basée sur une séquence de quantisations de réseaux de permutation.

— Le choix d'un paramétrage universel d'une porte quantique arbitraire d'arité 2 avec une architecture composée de portes quantiques paramétrées d'arité 1 et de portes CNOT.

— La quantisation et la comparaison des réseaux de permutation et de tri en tant qu'architectures qui permettent de permuter des qubits de manière arbitraire.

Un circuit quantique paramétré avec cette architecture universelle présente l'avantage de pouvoir représenter n'importe quel circuit quantique d'arité 2 d'une profondeur légèrement plus petite, peu importe les connexions dans ce circuit quantique. L'architecture universelle

permet donc de troquer une augmentation de la profondeur contre la capacité de représenter des circuits d'arité 2 connectés de manière arbitraire. Cet échange permet de cerner la capacité de représentation des circuits quantiques paramétrés avec cette architecture et de transformer le problème difficile d'optimisation d'architecture de circuit en problème moins difficile d'optimisation des paramètres d'un circuit.

6.3 Directions futures

Dans cette section, nous présentons quelques directions futures pour des projets de recherche reliés à ce que nous avons présenté dans ce mémoire.

6.3.1 L'expressivité des circuits quantiques paramétrés

Nous avons présenté une famille d'architectures quantiques universelles P indexée par l et p telle qu'un circuit quantique arbitraire d'arité 2, de largeur l et de profondeur p peut être plongé dans le membre $P_{l,p}$ de P en instanciant les paramètres de $P_{l,p}$ aux valeurs appropriées (telles que données explicitement dans notre construction). L'architecture $P_{l,p}$ a une profondeur $p' = 2p\log(l)$ et une largeur $l' = l$.

Classiquement, nous avons discuté des *classes d'hypothèses* du cadre de minimisation du risque empirique, ainsi que des théorèmes d'approximation universelle reliés aux réseaux de neurones. Quantiquement, nous avons aussi discuté de l'universalité par rapport aux familles de circuits. Cependant, nous avons évité de discuter de l'universalité par rapport aux familles de fonctions. Nous avons évité ce sujet en partie parce que le domaine n'est pas bien établi, mais certains articles ont commencé à explorer l'idée [**2, 24, 83**]. L'idée de tisser les liens entre ces différentes sortes d'universalité est une direction de recherche prometteuse.

6.3.2 L'encodage des données

Dans le chapitre 3, nous avons brièvement présenté différentes méthodes d'encodage des données (tableau 4.2.1.1). Ces méthodes sont implémentées par des circuits fixes avec des portes définies. Un sous-champ récent de recherche explore des circuits où l'encodage des données est appris [**55, 93**]. Enquêter sur les différentes méthodes d'encodage, soit fixes ou par apprentissage, pourrait être intéressant et pourrait tisser des liens entre l'expressivité de circuits quantiques et la méthode d'encodage de données classiques sur un support quantique.

6.3.3 Le lien entre l'universalité par permutation et l'universalité des réseaux de neurones

Nous avons présenté plusieurs concepts reliés à l'universalité. Premièrement, nous avons survolé l'universalité des réseaux de neurones par rapport à certaines familles de fonctions. Puis, nous avons présenté l'universalité de certaines architectures de circuit qui peuvent simuler des familles de circuits de taille plus petite. Ce faisant, nous avons construit des circuits quantiques paramétrés avec des garanties d'universalité par rapport à des familles de circuits quantiques.

Du côté classique, un lien entre l'universalité des réseaux de neurones et des architectures de circuits pourrait ouvrir la porte à des théorèmes d'universalité des réseaux de neurones où la largeur et la profondeur sont bornées. Conceptuellement, les résultats classiques qui décrivent les capacités des fonctions booléennes d'approximer des fonctions à valeurs réelles [**90**] pourraient être reliés à la structure des réseaux de neurones en considérant un réseau de neurones comme un grand circuit booléen. Les travaux qui explorent l'universalité des circuits booléens en tant que modèles d'apprentissage sont épars [**60, 61, 84**] et la discussion sur la complexité des circuits booléens pour représenter des familles de fonctions booléennes

[**80**] dépasse le cadre de ce document, mais leur étude serait intéressante dans le contexte d'apprentissage de circuits.

6.3.4 Une analyse détaillée de la performance de Rotosolve, de Baqprop et des modifications possibles de l'algorithme d'escalade

Dans la section 4.3.2, nous avons introduit *Rotosolve* [**65**], un algorithme d'optimisation itératif. Cet algorithme calcule l'angle optimal d'une porte quantique de rotation à un qubit en gardant le reste du circuit fixe. En modifiant un angle, il se peut que les valeurs optimales des angles des autres portes de rotation dans le circuit changent. Puisque l'optimisation des paramètres se fait de manière séquentielle, on ne peut pas dire qu'une telle précision dans le calcul de la modification d'un paramètre est désirable : une analyse détaillée de la convergence de l'algorithme est nécessaire pour déterminer les avantages et désavantages de cette méthode.

Dans la section 4.3.3, nous avons brièvement touché sur l'approche *baqprop* pour l'optimisation des paramètres d'un circuit quantique paramétré. Cette approche prometteuse requiert un travail supplémentaire pour fournir une analyse plus détaillée du nombre d'opérations de l'approche.

Plusieurs modifications de l'algorithme d'escalade simple sont présentés dans la section 4.3.4.2. Une analyse plus détaillée de l'impact de ces modifications est reléguée à un travail ultérieur.

6.3.5 L'architecture universelle par permutation est asymptotiquement optimale à une constante multiplicative près

Un circuit de permutation peut réarranger une séquence de n éléments dans n'importe laquelle des $n!$ permutations de ces éléments. Cependant, seulement un sous-ensemble de ces permutations correspond à l'ensemble des combinaisons de paires uniques.

Par exemple, les permutations (1,2,3,4), (2,1,3,4), (1,2,4,3), (2,1,4,3) correspondent toutes à la même combinaison de paires parce que les permutations des éléments *à l'intérieur* des paires {1,2} et {3,4} sont équivalentes pour l'universalité de l'architecture universelle. Les permutations (3,4,1,2), (4,3,1,2), (3,4,2,1), (4,3,2,1) correspondent également à la même combinaison de paires : les permutations *entre* les paires {1,2} et {3,4} n'ont pas d'impact. Il est donc intéressant de considérer l'existence d'une architecture de circuit capable de générer des combinaisons de paires uniques et dont la complexité asymptotique de la taille ou de la profondeur de l'architecture serait inférieure à celle du réseau de permutation de Beneš.

Considérons un ensemble de n éléments. Il existe $n!$ permutations de ces éléments. Pour chaque ensemble de paires unique des n éléments, il existe $2^{n/2}$ permutations où seulement l'ordre des éléments à l'intérieur des paires est modifié. Il existe aussi $(\frac{n}{2})!$ permutations où l'ordre entre les paires est modifié. Si le but de l'architecture est de pouvoir réarranger les n éléments en combinaisons de paires uniques, alors le circuit doit pouvoir générer

$$\frac{n!}{2^{n/2}(\frac{n}{2})!} = (n-1) \times (n-3) \times ... \times 1 := (n-1)!!$$

permutations [16] parce que les autres sont redondantes [**43, 34**].

16. On suppose que n est pair.

Un circuit constitué de portes SWAP avec une architecture capable de générer $(n-1)!!$ permutations doit avoir une taille d'au moins $\lg((n-1)!!)$.

La complexité asymptotique de la taille d'un circuit avec l'architecture universelle par permutation de Beneš est dans $\Theta(n \log n)$. Celle d'un circuit avec $\lg((n-1)!!)$ portes est aussi dans $\Theta(n \log n)$:

$$\begin{aligned}
\lg((n-1)!!) &= \lg\left(\frac{n!}{2^{n/2}(\frac{n}{2})!}\right) \\
&= \lg(n!) - \lg(2^{n/2}) - \lg\left((\frac{n}{2})!\right) \\
&\leq \lg(n!) \in \mathcal{O}(n \lg n)
\end{aligned}$$

De plus,

$$\begin{aligned}
\lg((n-1)!!) &= \lg(n!) - \lg(2^{n/2}) - \lg\left((\frac{n}{2})!\right) \\
&= \lg(n!) - \lg\left((\frac{n}{2})!\right) - \frac{n}{2} \\
&= \sum_{i=1}^{n} \lg(i) - \sum_{i=1}^{\frac{n}{2}} \lg(i) - \frac{n}{2} \\
&= \sum_{i=\frac{n}{2}+1}^{n} \lg(i) - \frac{n}{2} \\
&\geq \frac{n}{2} \lg\left(\frac{n}{2}\right) - \frac{n}{2} \\
&= \frac{n}{2}(\lg(n) - 2) \in \Omega(n \log n)
\end{aligned}$$

Considérant qu'il peut y avoir au plus $\frac{n}{2}$ portes par couche, la profondeur d'un réseau de permutation composé de portes SWAP qui peut générer les $(n-1)!!$ combinaisons de paires uniques est dans $\Omega(\log(n))$ et la profondeur de l'architecture universelle par permutation

de Beneš est asymptotiquement optimale à une constante multiplicative près pour toute architecture universelle qui alterne des modules de calcul et de permutation.

Une exploration de différentes architectures quantiques universelles qui ne sont pas à base de portes SWAP serait une direction de recherche intéressante.

Bibliographie

[1] Scott Aaronson : Read the fine print. *Nature Physics*, 11(4):291–293, 2015.

[2] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli et Stefan Woerner : The power of quantum neural networks. *arXiv preprint arXiv :2011.00027*, 2020.

[3] Esma Aïmeur, Gilles Brassard et Sébastien Gambs : Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2):261–287, 2013.

[4] Davide Anguita, Sandro Ridella, Fabio Rivieccio et Rodolfo Zunino : Quantum optimization for training support vector machines. *Neural Networks*, 16(5-6):763–770, 2003.

[5] Srinivasan Arunachalam et Ronald de Wolf : Guest column : A survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67, 2017.

[6] Leonardo Banchi et Gavin E Crooks : Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule. *Quantum*, 5:386, 2021.

[7] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin et Harald Weinfurter : Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

[8] Marcello Benedetti, Erika Lloyd, Stefan Sack et Mattia Fiorentini : Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[9] Václav E Beneš : Permutation groups, complexes, and rearrangeable connecting networks. *Bell System Technical Journal*, 43(4):1619–1640, 1964.

[10] Charles Bennett et Gilles Brassard : Quantum crytography. *In Proc. Int. Conf. Computers, Systems, and Signal Processing, Bangalore, India, 1984*, pages 175–179, 1984.

[11] Debajyoti Bera, Stephen Fenner, Frederic Green et Steven Homer : Efficient universal quantum circuits. *Quantum Information & Computation*, 10(1):16–28, 2010.

[12] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe et Seth Lloyd : Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[13] Michel Boyer, Gilles Brassard, Peter Høyer et Alain Tapp : Tight bounds on quantum searching. *Fortschritte der Physik : Progress of Physics*, 46(4-5):493–505, 1998.

[14] A Brandolini : Bullshit asymmetry principle. *In Conference presentation, XP2014, Rome*, volume 30, 2014.

[15] Hans J Briegel et Gemma De las Cuevas : Projective simulation for artificial intelligence. *Scientific Reports*, 2(1):1–16, 2012.

[16] Iris Cong, Soonwon Choi et Mikhail D Lukin : Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[17] Michael Crommie et Umesh Vazirani : Qubits, quantum mechanics and computers, lecture 5 : Universal gate sets, Schrödinger equation, quantum teleportation, 2003. Disponible sur `https://inst.eecs.berkeley.edu/~cs191/sp05/lectures/lecture5.ps`.

[18] George Cybenko : Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[19] William James Dally et Brian Patrick Towles : *Principles and practices of interconnection networks*. Elsevier, 2004.

[20] George B Dantzig : Origins of the simplex method. *In A history of scientific computing*, pages 141–151. 1990.

[21] Daoyi Dong, Chunlin Chen et Zonghai Chen : Quantum reinforcement learning. *In International Conference on Natural Computation*, pages 686–689. Springer, 2005.

[22] Daoyi Dong, Chunlin Chen, Hanxiong Li et Tzyh-Jong Tarn : Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.

[23] Daoyi Dong, Chunlin Chen, Tzyh-Jong Tarn, Alexander Pechen et Herschel Rabitz : Incoherent control of quantum systems with wavefunction-controllable subspaces via quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):957–962, 2008.

[24] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu et Dacheng Tao : Expressive power of parametrized quantum circuits. *Physical Review Research*, 2(3):033125, 2020.

[25] Vedran Dunjko et Hans J Briegel : Machine learning & artificial intelligence in the quantum domain : A review of recent progress. *Reports on Progress in Physics*, 81(7):074001, 2018.

[26] Vedran Dunjko, Jacob M Taylor et Hans J Briegel : Quantum-enhanced machine learning. *Physical Review Letters*, 117(13):130501, 2016.

[27] Vedran Dunjko, Jacob M Taylor et Hans J Briegel : Advances in quantum reinforcement learning. *In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 282–287. IEEE, 2017.

[28] Vedran Dunjko et Peter Wittek : A non-review of quantum machine learning : trends and explorations. *Quantum Views*, 4:32, 2020.

[29] Edward Farhi et Hartmut Neven : Classification with quantum neural networks on near term processors. *arXiv preprint arXiv :1802.06002*, 2018.

[30] Sébastien Gambs : *Apprentissage quantique*. Thèse de doctorat, Université de Montréal, 2008. Disponible sur `https://papyrus.bib.umontreal.ca/xmlui/handle/1866/6442`.

[31] Xavier Glorot et Yoshua Bengio : Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the thirteenth international Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[32] Ian Goodfellow, Yoshua Bengio et Aaron Courville : *Deep Learning*, volume 1. MIT press Cambridge, 2016.

[33] Daniel Gottesman : An introduction to quantum error correction and fault-tolerant quantum computation. *In Quantum Information Science and its Contributions to Mathematics, Proceedings of Symposia in Applied Mathematics*, volume 68, pages 13–58, 2010.

[34] Henry Gould et Jocelyn Quaintance : Double fun with double factorials. *Mathematics Magazine*, 85(3):177–192, 2012.

[35] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia et Kaiming He : Accurate, large minibatch SGD : Training Imagenet in 1 hour. *arXiv preprint arXiv :1706.02677*, 2017.

[36] Lov K Grover : Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325, 1997.

[37] Gian Giacomo Guerreschi et Mikhail Smelyanskiy : Practical optimization for hybrid quantum-classical algorithms. *arXiv preprint arXiv :1701.01450*, 2017.

[38] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel et Sergey Levine : Reinforcement learning with deep energy-based policies. *In International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[39] Boris Hanin et Mark Sellke : Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv :1710.11278*, 2017.

[40] Aram W Harrow, Avinatan Hassidim et Seth Lloyd : Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[41] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow et Jay M Gambetta : Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[42] Kurt Hornik : Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2): 251–257, 1991.

[43] André Nicolas (https ://math.stackexchange.com/users/6312/andrnicolas) : Combination of splitting elements into pairs. Mathematics Stack Exchange. URL :https ://math.stackexchange.com/q/35959 (version : 2011-04-30).

[44] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley et E Miles Stoudenmire : Towards quantum machine learning with tensor networks. *Quantum Science and Technology*, 4(2):024001, 2019.

[45] Sofiene Jerbi, Lea M Trenkwalder, Hendrik Poulsen Nautrup, Hans J Briegel et Vedran Dunjko : Quantum enhancements for deep reinforcement learning in large spaces. *PRX Quantum*, 2(1):010328, 2021.

[46] Iordanis Kerenidis et Jonas Landman : Quantum spectral clustering. *Physical Review A*, 103(4): 042415, 2021.

[47] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo et Anupam Prakash : q-means : A quantum algorithm for unsupervised machine learning. *arXiv preprint arXiv :1812.03584*, 2018.

[48] Leonid Genrikhovich Khachiyan : A polynomial algorithm in linear programming. *In Doklady Akademii Nauk*, volume 244, pages 1093–1096. Russian Academy of Sciences, 1979.

[49] Patrick Kidger et Terry Lyons : Universal approximation with deep narrow networks. *In Conference on Learning Theory*, pages 2306–2327. PMLR, 2020.

[50] Andreĭ Nikolaevich Kolmogorov et Albert T Bharucha-Reid : *Foundations of the Theory of Probability : Second English Edition*. Courier Dover Publications, 2018.

[51] Anastasis Kratsios : The universal approximation property. *Annals of Mathematics and Artificial Intelligence*, 89(5):435–469, 2021.

[52] Lucas Lamata : Basic protocols in quantum reinforcement learning with superconducting circuits. *Scientific Reports*, 7(1):1–10, 2017.

[53] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato et Fu Jie Huang : A tutorial on energy-based learning. *In Predicting Structured Data*. MIT Press, 2006.

[54] Jin-Guo Liu et Lei Wang : Differentiable learning of quantum circuit Born machines. *Physical Review A*, 98(6):062324, 2018.

[55] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac et Nathan Killoran : Quantum embeddings for machine learning. *arXiv preprint arXiv :2001.03622*, 2020.

[56] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu et Liwei Wang : The expressive power of neural networks : A view from the width. *arXiv preprint arXiv :1709.02540*, 2017.

[57] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa et Keisuke Fujii : Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[58] Tom M. Mitchell : *Machine Learning*. McGraw-Hill, New York, 1997.

[59] Ioannis Mitliagkis, Mingde (Harry) Zhao et Dylan Troop : Theoretical principles for deep learning, lecture 7 : Elements of statistical learning theory, 2019. Disponible sur `http://mitliagkas.github.io/ift6085-2020/ift-6085-lecture-7-notes.pdf`.

[60] Marco Muselli : Approximation properties of positive boolean functions. *In Neural Nets*, pages 18–22, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[61] Marco Muselli : Switching neural networks : A new connectionist model for classification. *In Neural Nets*, pages 23–30, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[62] Michael A Nielsen : *Neural Networks and Deep Learning*. Determination Press, 2015.

[63] Michael A. Nielsen et Isaac L. Chuang : *Quantum Computation and Quantum Information : 10th Anniversary Edition*. Cambridge University Press, 2010.

[64] Chris Olah : Calculus on computational graphs : Backpropagation, août 2015. En ligne. Disponible à https ://colah.github.io/posts/2015-08-Backprop/.

[65] Mateusz Ostaszewski, Edward Grant et Marcello Benedetti : Quantum circuit structure learning. *arXiv preprint arXiv :1905.09692*, 2019.

[66] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. Schuyler Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, Colm A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, Robert S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, Blake R. Johnson, M. Reagor, M. P. da Silva et C. Rigetti : Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv :1712.05771*, 2017.

[67] Giuseppe Davide Paparo, Vedran Dunjko, Adi Makmal, Miguel Angel Martin-Delgado et Hans J. Briegel : Quantum speedup for active learning agents. *Phys. Rev. X*, 4:031002, 2014.

[68] Sejun Park, Chulhee Yun, Jaeho Lee et Jinwoo Shin : Minimum width for universal approximation. *arXiv preprint arXiv :2006.08859*, 2020.

[69] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez et Rupak Biswas : Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3):030502, 2018.

[70] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik et Jeremy L O'brien : A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):1–7, 2014.

[71] John Preskill : Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[72] Patrick Rebentrost, Masoud Mohseni et Seth Lloyd : Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014.

[73] Herbert Robbins et Sutton Monro : A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[74] Jonathan Romero, Jonathan P Olson et Alan Aspuru-Guzik : Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001, 2017.

[75] Maximilian Schlosshauer : Decoherence, the measurement problem, and interpretations of quantum mechanics. *Reviews of Modern Physics*, 76(4):1267, 2005.

[76] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac et Nathan Killoran : Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[77] Maria Schuld, Alex Bocharov, Krysta M Svore et Nathan Wiebe : Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[78] Maria Schuld et Nathan Killoran : Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, 122(4):040504, 2019.

[79] Maria Schuld et Francesco Petruccione : *Supervised Learning with Quantum Computers*. Springer, 2018.

[80] Claude E Shannon : The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

[81] Peter W. Shor : Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[82] Peter W Shor : Why haven't more quantum algorithms been found ? *Journal of the ACM*, 50(1):87–90, 2003.

[83] Sukin Sim, Peter D Johnson et Alán Aspuru-Guzik : Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[84] Alain Tapp : A new approach in machine learning. *arXiv preprint arXiv :1409.4044*, 2014.

[85] Leslie G Valiant : Universal circuits (preliminary report). *In Proceedings of the eighth annual ACM Symposium on Theory of Computing*, pages 196–203, 1976.

[86] Vladimir Vapnik : Principles of risk minimization for learning theory. *In Advances in Neural Information Processing Systems*, pages 831–838, 1992.

[87] Farrokh Vatan et Colin Williams : Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69(3):032315, 2004.

[88] Guillaume Verdon, Jason Pye et Michael Broughton : A universal training algorithm for quantum deep learning. *arXiv preprint arXiv :1806.09729*, 2018.

[89] Nathan Wiebe, Ashish Kapoor et Krysta Svore : Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *arXiv preprint arXiv :1401.2142*, 2014.

[90] Ronald de Wolf : *A Brief Introduction to Fourier Analysis on the Boolean Cube*. Numéro 1 de Graduate Surveys. Theory of Computing Library, 2008.

[91] David H Wolpert : The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[92] David H Wolpert et William G Macready : No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[93] Christa Zoufal, Aurélien Lucchi et Stefan Woerner : Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.