Freie University Berlin

Quantum Machine Learning : Encoding & Gradient Descent on Near-Term Quantum Devices

Barthélémy Meynard
Piganeau
August 2018, 17th

Table des matières

1 Introduction

Current techniques for machine learning on quantum computer rely on Parameterized Quantum Circuits (PQCs). A quantum computer is composed of quantum bits, called qubits. N qubits generate an Hilbert space of dimension 2^N. The state of these qubits can be modified, through the application of quantum gate (current technology is mainly limited to single qubit gate and controlled single qubit gate - which are universal -). A quantum circuit is the complete process ; starting from $|0\rangle^{\otimes N}$, the circuit is composed of a serie of gates on the different qubits. It usually end by a measurement of some qubits. A PQC is a quantum circuit whose gate are parameterized by a vector $\boldsymbol{\theta}$. To get a better idea I would recommend having a look at [1] and [2].

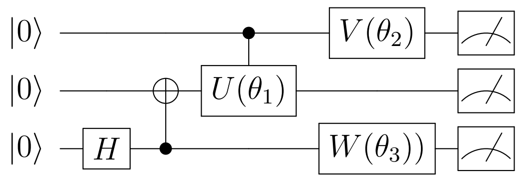

FIGURE 1 – example of PQC ; a quantum circuit carameterized by θ_1, θ_2 and θ_3

Hybrid computation techniques are designed to be usable on near term devices. The idea is to minimize the quantum part by having classical part in the process. Especially this idea had some great results with VQE [3]. For optimization problems, the optimization over the parameter θ can be made by a classical computer. The same idea is applied to PQCs. Especially, using the results from a circuit, a classical computer computes an approximation of the gradient and proceed to a gradient descent over the parameters. It is one of the most promising application for near term quantum computers.

2 End to End Protocol for Quantum Auto-Encoding

2.1 Encoding a Classical Data into a Quantum State

The question I finally decided to focus on for this internship is specific for CQ Learning (Classical Data, Quantum hardware). Indeed While trying to replicate the results from paper [4], I had to implement a quantum circuit supposed to classify the famous MNIST dataset. The classifier proposed made a lot of sense, and was intuitively a good to start approaching this kind of problem. Indeed it joins rotations on the different qubits and entangling gates. Moreover, entanglement is one of the key principles of quantum supremacy. However something was a bit forgotten in this paper and much of the other similar papers, and that is the question of encoding your data into a quantum state. Indeed we are using classical data and finding a good quantum representation. The first thing to do is to precise what good representation means. Firstly let's look at the different encodings mainly used by the community [5] :

2.1.1 Basis Encoding :

Basis encoding associates a computational basis state of a n-qubits system (such as $|3\rangle = |0011\rangle$) with a classical n-bit-string (0011). In a way, this is the most straight forward way of computation, since each bit gets literally replaced by a qubit, and a 'computation' acts in parallel on all bit sequences in a superposition

2.1.2 Amplitude Encoding :

A normalized classical vector $x \in \mathbb{C}^{2^n}, \sum_k |x_k|^2 = 1$ can be represented by the amplitudes of a quantum state $|\phi_{AE}(x)\rangle$.

$$\text{for } x = (x_1, ..., x_n), \qquad |\phi_{AE}(x)\rangle = \sum_{j=1}^{2^n} x_j |j\rangle$$

.

2.2 Machine Learning to Improve the Encoding

The current way of making a quantum classifier is to apply one of this embedding, then to apply a classifier. The classifier is then composed of a serie a gates which parameter are going to be learned in order make the classifier have the best possible result on a training dataset.

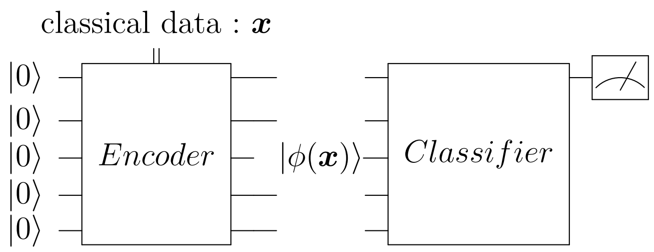

FIGURE 2 – Complete forward path of a classifier

The idea was to make the encoding parameterized and then to build an end to end model which can optimize over the classifier at the same time as over the encoder. This approach is greatly inspired by the word2vec [6] word embedding. The idea is to use machine learning to create a general encoding for words which is then the starting point for other machine learning algorithm.

The goal of such an approach is double :

— proposing an algorithm which, after the learning phase is more efficient, more robust to noise and that can be implemented on a large scale of Quantum computers (and of qubits).

— Looking at a 'good' encoding, in order to gain more insight about the links between classical and quantum properties (for example between correlation and entanglement).

2.2.1 Define an Architecture :

The question of defining a good encoding architecture is very difficult. The idea is to choose a generic process to encode the data into a quantum state. To do so we try to compromise 3 different goals : reducing the length of the circuit and it's complexity, avoiding information loss, maximizing the expressivity of the encoding.

The complexity is the first and clearest improvement we can try to implement for near term machines. Indeed actual quantum computers are facing two main challenges ; the quantum noise appears each time we apply a gate and the number of qubits available is limited. Amplitude encoding requires a circuit length exponential in the numbers of qubits. Such scaling implies that the noise also increases exponentially with the numbers of qubits and so can not become a viable solution for complex problems requiring a large number of qubits. On the other hand, the basis encoding only requires a very short circuit but a very large number of qubits $n_{qubits} = length(data)$. Finally, the last default in this encoding which has motivated my project is rather based on intuition and is linked with expressivity. The whole question is to wonder if some quantum states do not better represent some inner structure of the data. The easiest intuition one's can have, is to relate classical correlation with quantum entanglement. From this perspective, both Amplitude embedding and Basis encoding seem very bad. In the basis encoding, each data is held by one qubit, giving rise to a factorized state ; this encoding only access a very small portion of the Hilbert space, and we can have the intuition that this portion is not the best. The amplitude encoding has the opposite problem, meaning that it only gives rise to a highly entangled state. You could wonder why this should be a problem ? Let's consider again our primitive intuition linking classical correlation to entanglement. In a usual dataset, each feature is not highly correlated to every other feature, some group of features may be independent of other groups. From this point of view, the amplitude embedding is here also missing the "best corner" of the Hilbert space.

We decided to define an encoding architecture based on a simple layer that we can apply as many times as necessary. The main advantage is that it can adapt to any number of qubits and so avoids the exponential scaling of the circuit length.

Definition 1 (Encoding layer with fix maximum entanglement)**.** *Let $x = (x_1, ..., x_N)$ in $\mathcal{X}$ be the data to encode in a n_{qubits} qubits computer. Let $L(x_1, ..., x_{nqubits})$ be a layer defined by : on each layer the data x_i is encoded on the i^{th} qubit with a rotation around the $\boldsymbol{x}$ axis. On each qubit is then applied a rotation around $\boldsymbol{y}$ and $\boldsymbol{z}$ of some angles $\theta_{i,y}$ and $\theta_{i,z}$. Finally we apply a $CNOT$ on each qubit, controlled by the qubit before.*

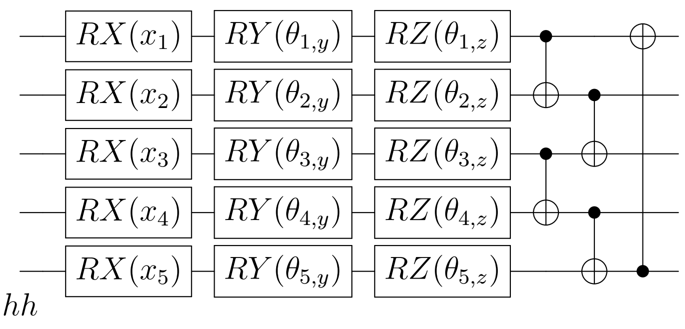

FIGURE 3 – Example encoding layer on 5 qubits periodic circular computer

This encoding layer encodes n_{qubits} features and generates $2n_{qubits}$ parameters to learn. A global encoding of N features requires to apply $n_l = \lfloor \frac{N}{n_{qubits}} \rfloor + 1$ layers, requiring $2n_l n_{qubits}$ parameters to learn for a length of $5n_l$.

This very simple layer encodes the data as a rotation then apply an arbitrary rotation and finally, we entangle the qubits through $CNOT$ to "connect" the different qubits. The parallel between series of $CNOT$ and the different layers used in Deep learning is explained more thoroughly in [4]

However, following the first observation linking correlation to entanglement, $CNOT$ was maybe not the best gate. Indeed it imposes a fix entanglement on the two qubits. Although the rotation before the $CNOT$ gives a partial control over the entanglement the expressivity of these layer had to be increased giving a more precise control over the entanglement. That's why it made sense to try a second layer which consisted of the replacement of the $CNOT$ by CRX. The entanglement can then be optimized by adjusting the angle parameters of the $CRXs$. It give the possibility to entangle between $\alpha = 0$ (no entanglement) and $\alpha = pi$ (CRX becomes a $CNOT$).

Definition 2 (Encoding layer with adaptive entanglement)**.** *Let $x = (x_1, ..., x_N)$ in $\mathcal{X}$ be the data to encode in a n_{qubits} qubits computer. Let $L(x_1, ..., x_{nqubits})$ be a layer defined by : on each layer the data x_i is encoded on the i^th qubit with a rotation around the $\boldsymbol{x}$ axis. On each qubit is then applied a rotation around $\boldsymbol{y}$ and $\boldsymbol{z}$ of some optimizable angles $\theta_{i,y}$ and $\theta_{i,z}$. Finally we apply a controlled rotation around the $\boldsymbol{x}$ axis on each qubit, controlled by the qubit before.*

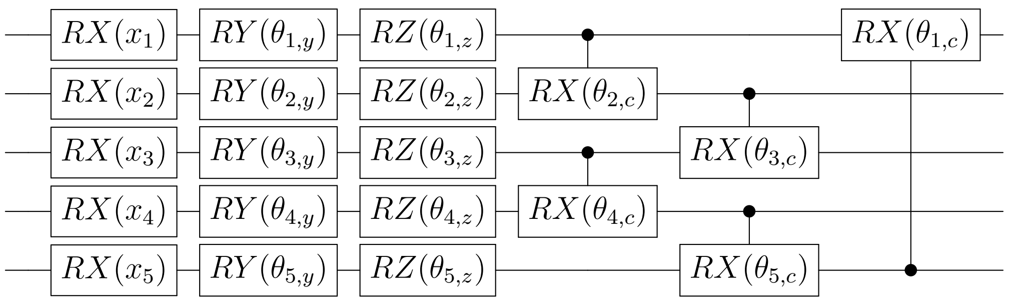

FIGURE 4 – Example encoding layer on 5 qubits periodic circular computer

This encoding layer encodes n_{qubits} features and generates $2n_{qubits}$ parameters to learn. A global encoding of N features require to apply $n_l = \lfloor \frac{N}{n_{qubits}} \rfloor + 1$ layers, requiring $3n_l n_{qubits}$ parameters to learn for a length of $5n_l$.

The whole principle of this work is to experiment an end to end classifier, so that this encoding is going to take into account the task.

2.2.2 Define a Classifier :

Having defined all the different parts, we can now build a complete classifier. To do so we firstly encode the data into a quantum state by applying the layers define in the last section. For exemple if if we want to encode 30 features in 5 qubits, we just need to apply the layer 6 times. Once all the features have been encoded, we start applying the same layers, but with all parameter being learnable. This part is then processing the data. Finally we make a several measurements of the first qubit to compute the expectation value. This value is my predicted label.

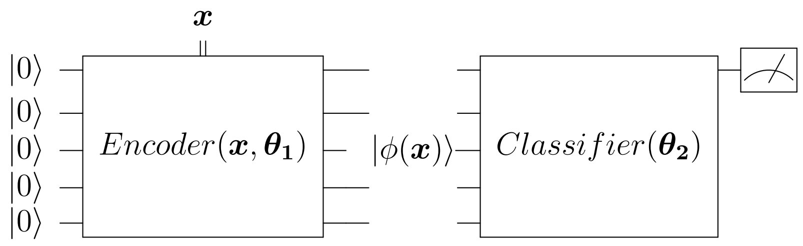
$\boldsymbol{x}$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$Encoder(\boldsymbol{x}, \boldsymbol{\theta_1})$
$|\phi(\boldsymbol{x})\rangle$
$Classifier(\boldsymbol{\theta_2})$

FIGURE 5 – Complete forward path of the classifier

2.3 Experiments

2.3.1 Datasets

The next step was then to choose some dataset and understand more the behavior of this new classifier. To conduct some experiments we decided to focus on a real dataset, contrary to some precedent approach. Simulating a quantum computer being very costly we had to restrict ourself to small datasets.

Firstly the experiments were conducted on the Iris dataset : It is a multivariate dataset introduced by the British statistician and biologist Ronald Fisher. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample : the length and the width of the sepals and petals, in centimeters.

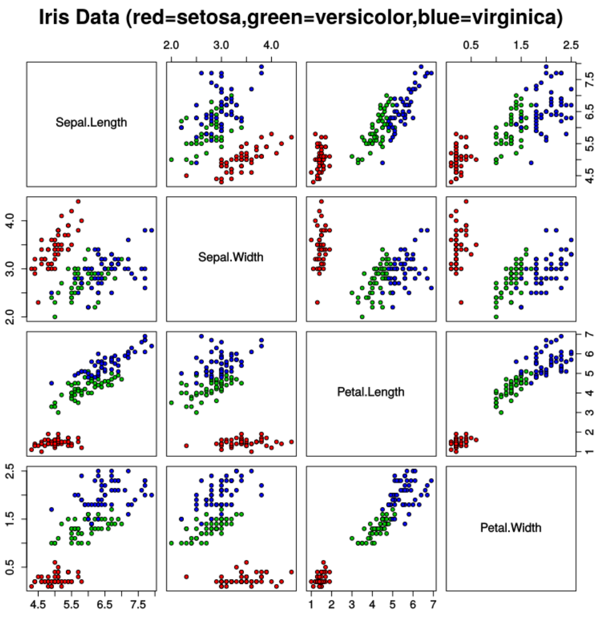

FIGURE 6 – projection of the Iris dataset for the different features

Then we looked at a medical dataset, the Breast cancer Wisconsin dataset. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in : [K. P. Bennett and O. L. Mangasarian : "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34]. It focuses on the following features :

— radius
— texture
— perimeter
— area
— smoothness (local variation in radius lengths)
— compactness
— concavity
— concave points (number of concave portions of the contour)
— symmetry
— fractal dimension

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features multivariate dataset for binary classification.

2.3.2 Results

The training has been a very long process. Although the task doesn't seem very complex, it becomes very costly as soon as we start using a quantum simulator. To do so I mainly relied on the combination of two computational tools. Firstly I used the deep learning library Pytorch [7], to build the computational graph. Inside this graph, I had to simulate the quantum behavior of the gates, and to do so relied on the library PennyLane [8].

Iris The first results were computed on the Iris dataset, although very simple is was a good to check the algorithm and get some more insight about process.

We firstly have quick look on the training loss, to check if the optimization has approximately reached a minima.

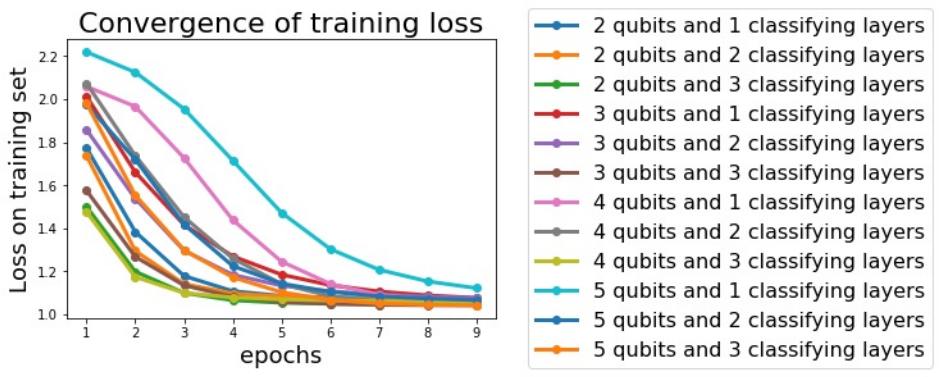

FIGURE 7 – Convergence of the training losses

By playing with the number of qubits and the size of the classifier, we can observe the performances of the different classifiers.

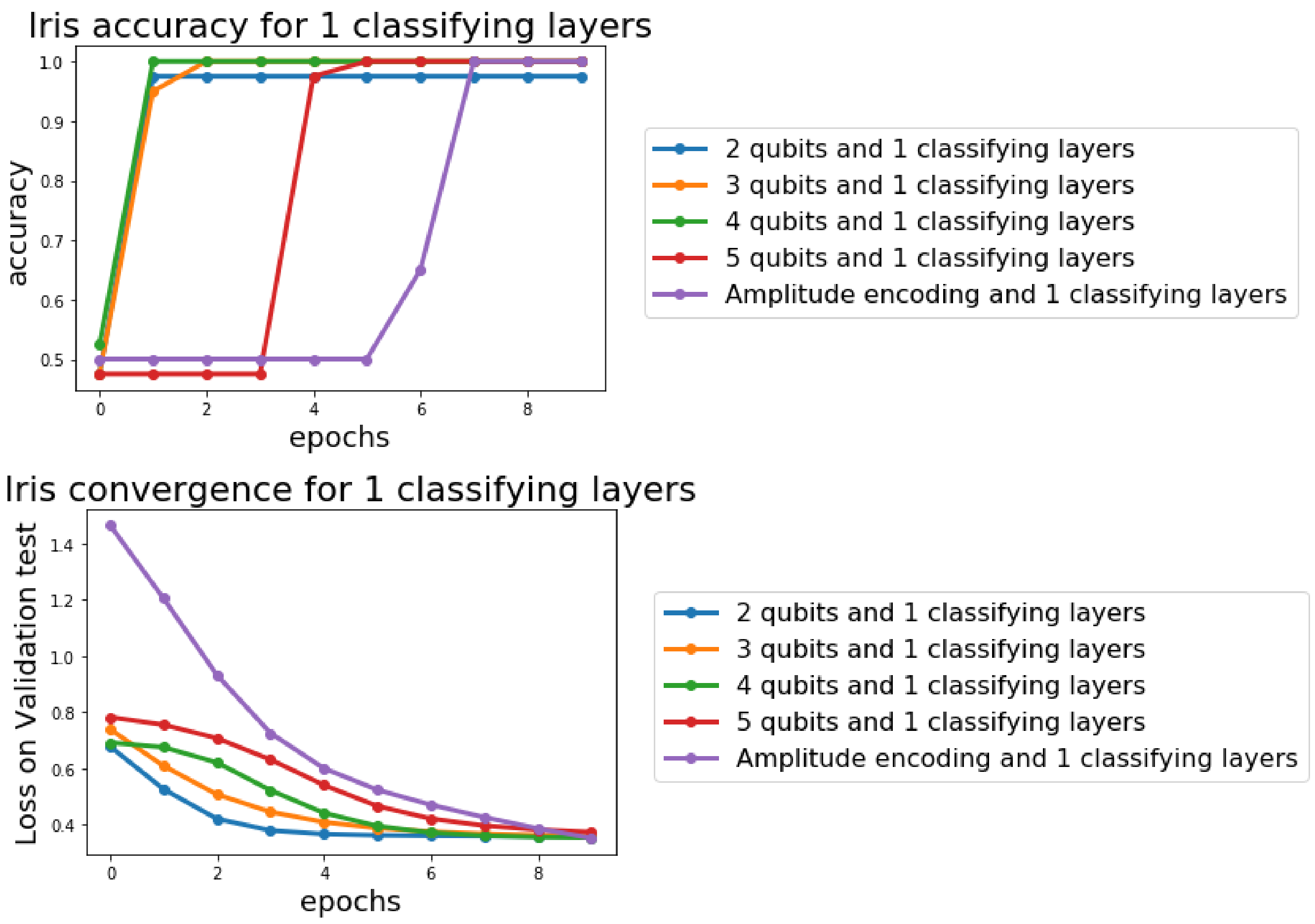

Figure 8 – Convergence of the training losses

Here is a first example of these performances. For the comparison of my encoder make sense, the performances are always plotted for a fixed classifier. These results were very motivating. Indeed we clearly see that for simple classifiers, the optimization is much faster on the new model. The main idea is that the encoder can find a better way to represent the datapoint, letting the relevant feature be expressed on the axis of the Hilbert space that can be easily and smartly used by the classifier.

The other plots will shown in the appendix

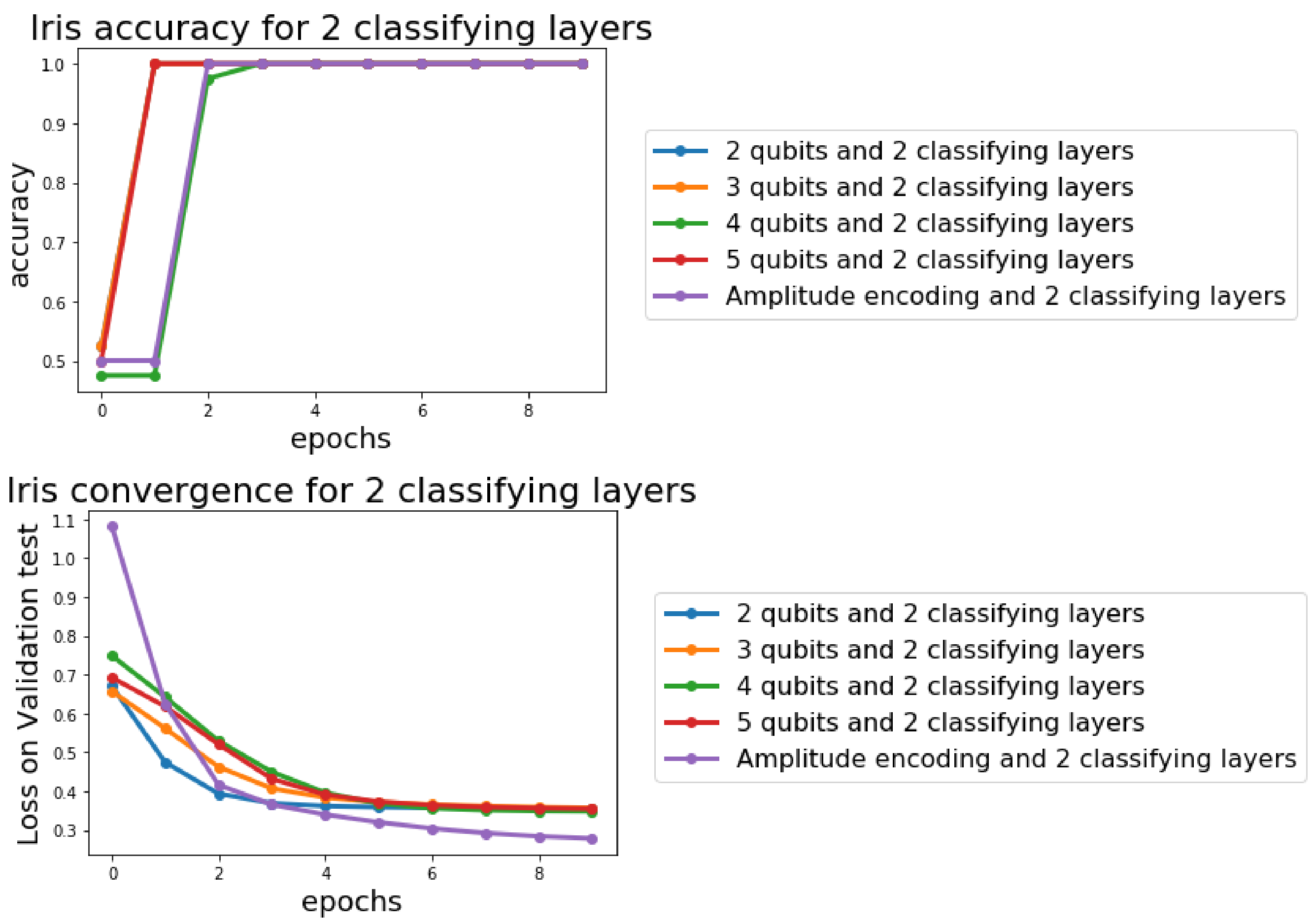

FIGURE 9 – Convergence of the training losses

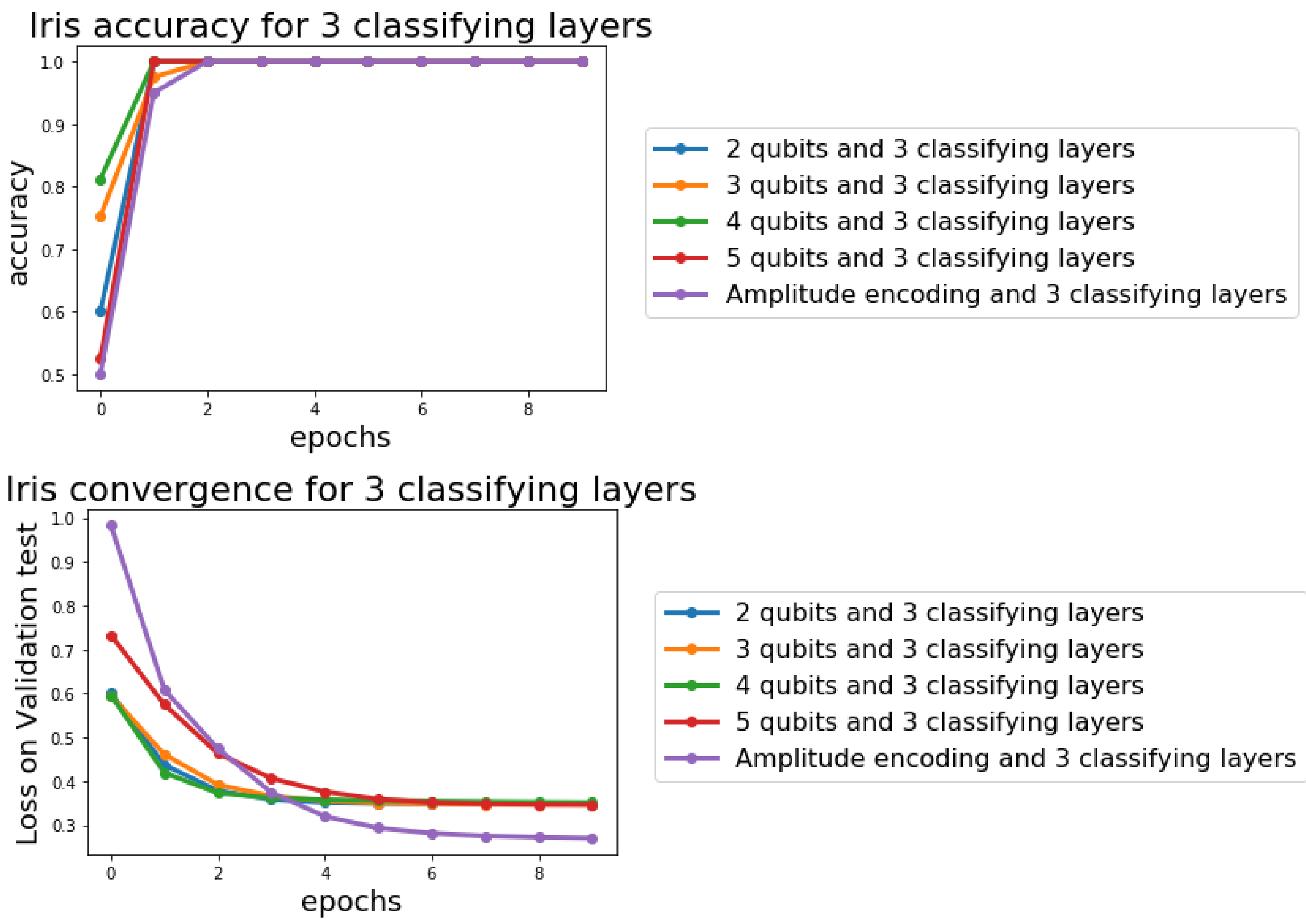

FIGURE 10 – Convergence of the training losses

Breast Cancer Dataset The breast cancer dataset is much more complex. With its 30 features, the perfect score (accuracy = 1) can no longer be achieved, letting room for exploration.

The optimization on breast Cancer dataset has been much more difficult. Indeed each point was computed in a few hours, meaning that each optimization of a new classifier or encoder could take between one and two weeks.

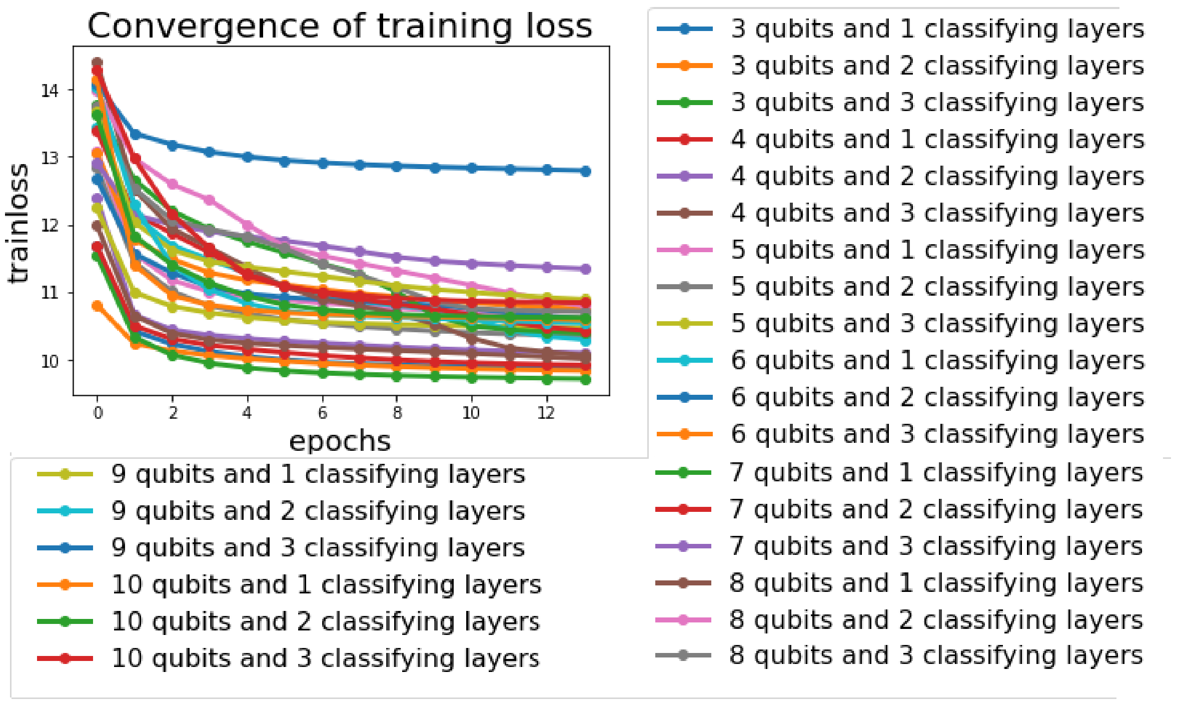

Figure 11 – Convergence of the training losses

Here again, the training losses seem nearly optimized, letting us have a more precise look upon details of these results.

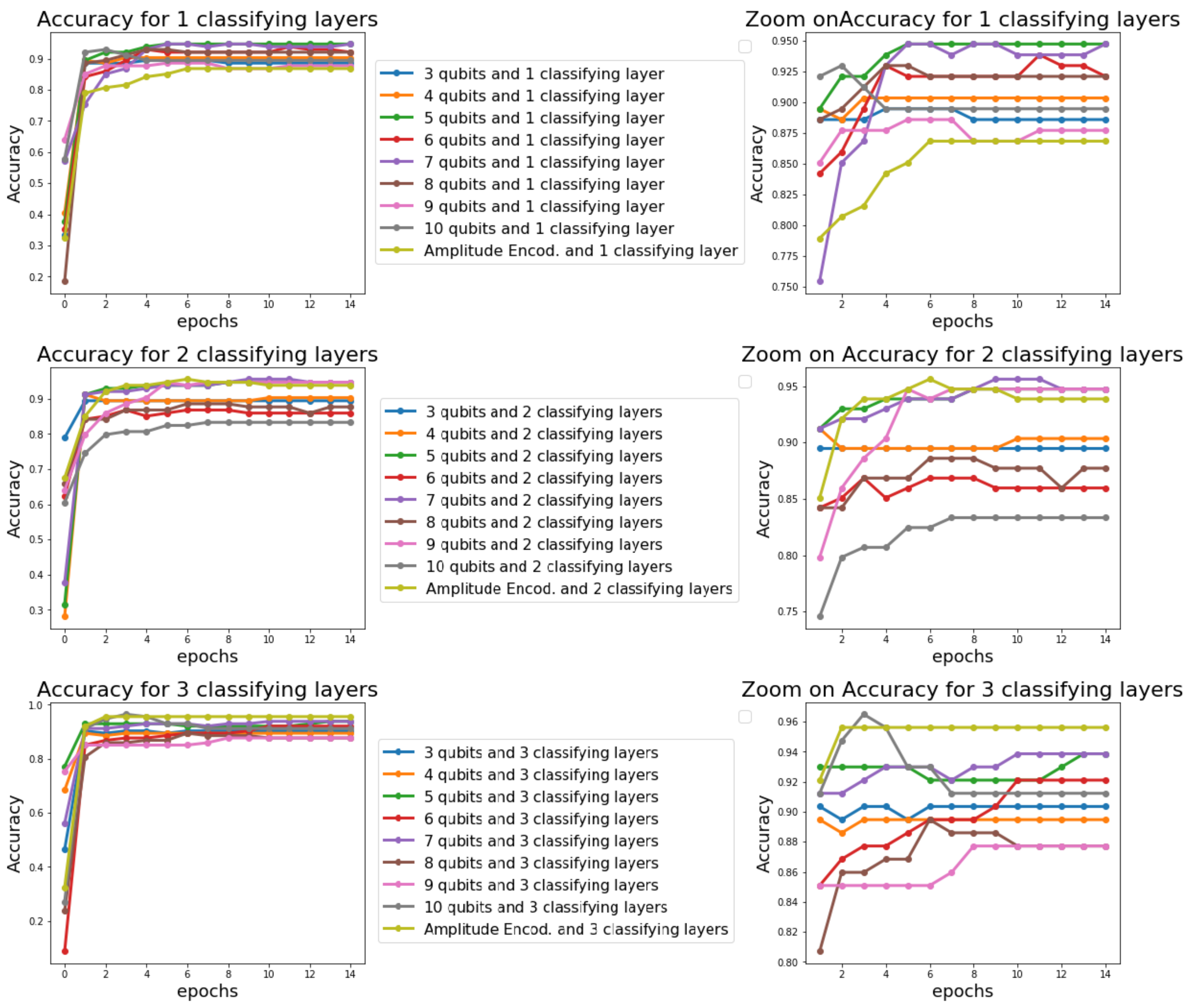

FIGURE 12 – Evolution of the accuracy for different encodings and classifiers

Many things have to be noticed from these plots. Firstly for most of the algorithms, except the less expressive (few qubits and few layers), we can notice that the accuracy decrease after having reached a maximum. This an example of overfitting ; meaning that after having found a good solution, the algorithm improves its score on the training set by "learning by heart" the data-points. It starts to fit the points precisely instead of focusing on the general trend. This is a very famous generalization problem, highly investigated in the ML field.

Here again for the most simple classifiers, the new encoding both gets a quicker optimization and slightly better results. However, for more complex classifiers the results are a bit disappointing. Indeed we could have expected that the feature embedding in a higher dimension would make the data more easily separable. A plot of the best accuracy achieved by each algorithm seems highly stochastic.

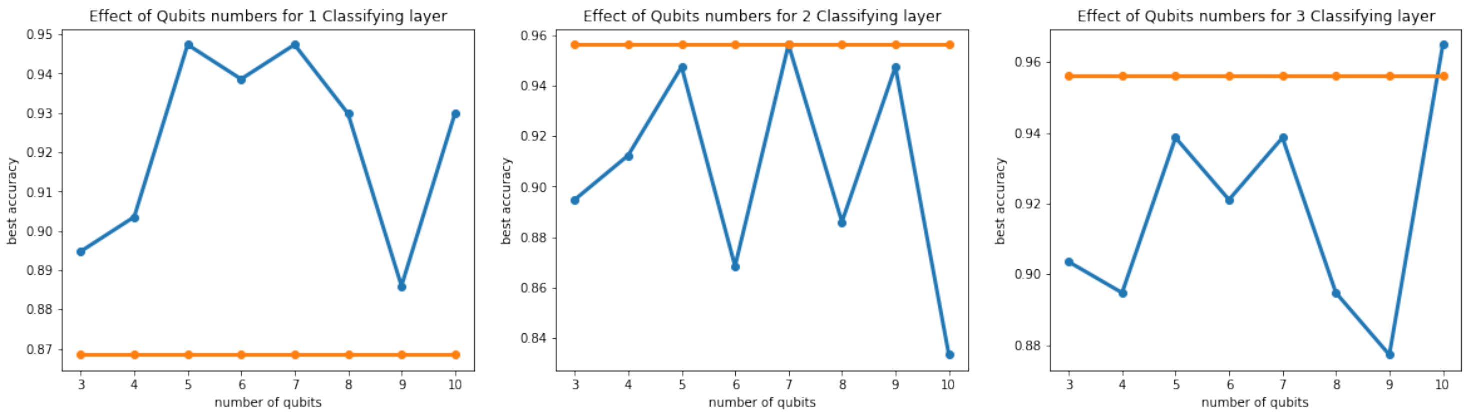

FIGURE 13 – Comparing the results between the new encoding and the Amplitude encoding

Based on a classical machine learning intuition, we could have expected that the best accuracy was going to slowly increase and then decrease with the number of qubits (so with the size of the Hilbert space). Increasing the size of the space is supposed to make the data more easily separable, it also enables a much more flexible entanglement. On the other hand, a space too big only makes the algorithm more complex to optimize.

Looking into the encoding A good way to have an insight of the encoding is to perform a PCA (Principal Component Analysis). Indeed the Principal Component Analysis is a statistical procedure that uses an orthogonal transformation to reduce a high dimensional correlated dataset into uncorrelated variables called principal components. Selecting only the two main principal components enables to plot them.

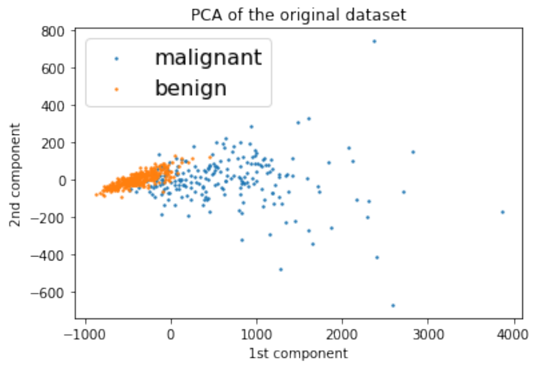

FIGURE 14 – Original data (or amplitude embedded) after a PCA

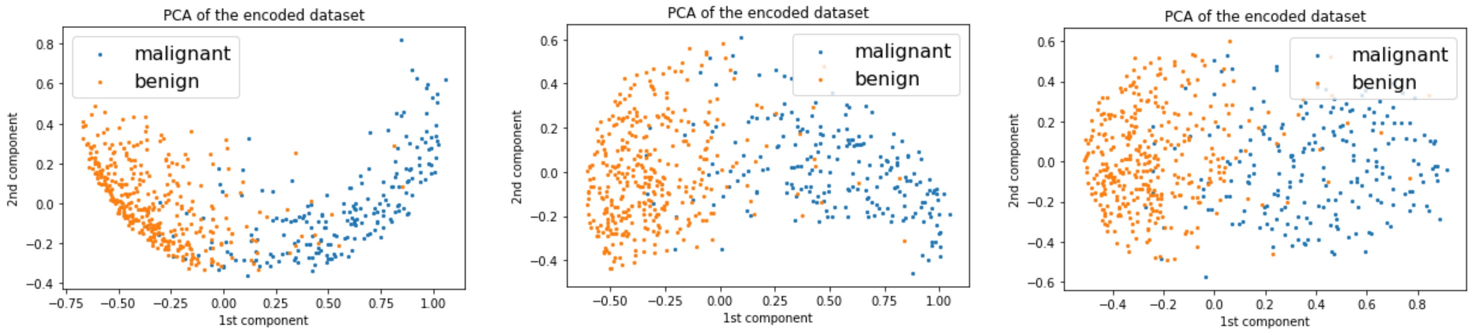

FIGURE 15 – Encoded data after a PCA for 3, 5 and 10 qubits

We clearly see that even when just looking at the two first components, the encoding is expending the data, making it more easily separable.

increasing the expressivity Maybe the encoding was not expressive enough to make a real difference. However the same experiments conducted with the second architecture to results even worst.

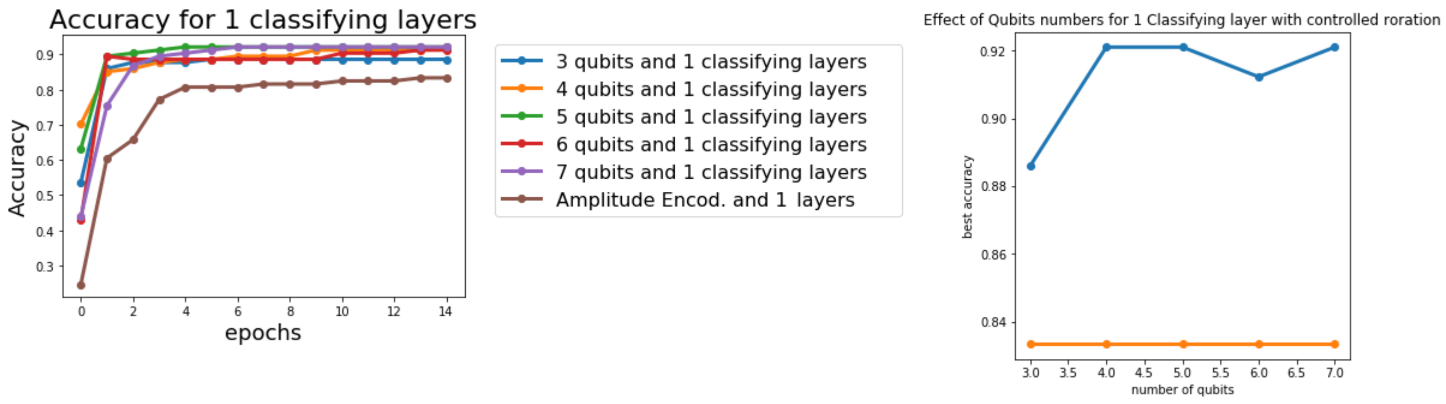

FIGURE 16 – Second Encoding architecture

This results clearly show that the expressivity is not the problem here. Indeed the CRX gate can perform a $CNOT$ for an angle of π. The algorithm should then in the worst case reproduce the same results than with the $CNOTs$. The reasons for the failure of this algorithm can be more precisely identified. We are facing an issue on the optimization. We are most certainly evolving in a high dimensional space, trying to optimize non convex function.

2.4 Mathematical Background and New Leads

Binary classification is the simplest problem of supervised learning : Let our data $z \in Z$ be couples $z = (x, y) \in X \times Y$ and parameters $f \in \mathcal{F}$ are functions $f : X \to Y$. The most classical problems in supervised learning are classification where Y is a discrete set as in binary classification where $Y = 0, 1$. Among classical loss functions, one can mention the loss in classification $\mathcal{L}^{0,1} f(x, y) = \mathbb{1}_{y \neq f(x)}$. In the couple (x, y), x is called the input and y the output. The interpretation of supervised learning is that one wants to predict a typical output Y associated with the input X when $(X, Y) \sim P$.

Definition 3. *Let $\mathcal{H}$ be an Hilbert space, called the feature space, X a set data. A feature map is a map $\phi : X \to \mathcal{F}$ which map inputs to vectors in the Hilbert space. We call feature vectors the vectors $\phi(x)$ with $x \in X$.*

The goal of such operation is to work in a space where the task is getting more simple. Indeed in an higher-dimensional space, the dataset can become linearly separable. That's the whole idea behind the used of kernel techniques for support vector machine. Let's look at our classifier. Let $x \in X$ be a data point. Firstly the encoder maps the data to a quantum state :

$$|\phi(x)\rangle = U_{\theta 1}(x)|0\rangle$$

. The classifier being a quantum circuit, it is defined by the linear operation of an unitary $V_{\theta 2}$. We finally compute the expectation value of the Pauli Z observable.

$$f(x) = \langle \phi(x)|V_{\theta 2}^{\dagger} Z V_{\theta 2}|\phi(x)\rangle = \langle 0|U_{\theta 1}^{\dagger}(x) V_{\theta 2}^{\dagger} Z V_{\theta 2} U_{\theta 1}(x)|0\rangle$$

For (X, Y) being the training set, we then optimize the training loss :

$$\sum_{(x_i, y_i) \in (X,Y)} \mathcal{L}(f(x_i), y_i)$$

As we have seen this optimization is far from being easy. Some results about the Kernel techniques in machine learning should, however, give us a hint for an improvement[9], [10].

Definition 4. *Let X be a data set, and $K : X \times X \to \mathbb{C}$. K is a Positive Semi Definite kernel, if there is an inner product space $\mathcal{H}$ and a map $\phi : X \to \mathcal{H}$ so that : $K(x, y) = \langle \phi(x)|\phi(y)\rangle$ for all $x, y \in X$.*

Definition 5. *Let X be a data set and $\mathcal{H}$ a Hilbert space of functions $f : X \to \mathbb{C}$ mapping the data to vectors of the Hilbert space. Let $\langle \cdot|\cdot \rangle$ be an inner product defined on $\mathcal{H}$. $\mathcal{H}$ is a reproducing kernel Hilbert space if there exists a kernel $K : X \times X \to \mathbb{C}$ satisfying. $\langle f(\cdot)|K(x, \cdot)\rangle = f(x)$ for all $(f, x) \in \mathcal{H} \times X$*

The main idea is to project your data in a much bigger space where the data becomes linearly separable. Choosing this space can be a very hard problem, and if its size is really important we have no guarantee on our capacity to optimize. That's where the RKHS find their interest, they are at the same time huge, and with very smooth property, enabling a much simpler optimization. Indeed in optimizing in the RKHS is equivalent to staying in the original space with a replacement of the inner product by the kernel.

Especially we can restrict our optimization in the RKHS to the *Span* of the function generated by our dataset $(k(x_i, \cdot))$.

Theorem 1 (The Representer Theorem)**.** *. Let K be a SPD kernel on X and let $\mathcal{H}$ be its associated RKHS. Fix The the optimization problem $min_{f \in \mathcal{H}} D(f(x_1), ..., f(x_n)) + P(||f||^2)$*

where P is nondecreasing and D depends on f only though $f(x_1), ..., f(x_n)$. If it has a minimizer, then it has a minimizer of the form

$$f = \sum_{i=1}^{n} w_i k(x_i, \cdot)$$

The current classifier doesn't fulfill the condition to apply the representer theorem. Indeed the measurement step adds a non-linearity in the end which breaks the Hilbert space structure of the space we are trying to optimize over.

Definition 6 (canonical RKHS generated by a feature map [9])**.** *Let $\phi : X \to \mathcal{H}$ be a feature map over an input set X , giving rise to a complex kernel $K(x_1, x2) = \langle\phi(x_1)|, |\phi(x_2)\rangle$. The corresponding reproducing kernel Hilbert space has the form $f : X \to \mathbb{C} | f(x) = \langle\omega|\phi(x)\rangle, forall x \in X, \omega \in \mathcal{H}$*

Replacing the measurement by an inner product with a well chosen vector ω would however fix this problem.

$$f_{RKHS,\omega,\theta_1,\theta_2}(x) = \langle\omega|V^{\dagger}_{\theta 2}U_{\theta 1}(x)|0\rangle$$

For a given θ_1 the optimization over θ_2 can then follow the Representer theorem. In a further development of this project, a sequential optimization over θ_1 and θ_2 should be tried. Moreover in a longer-term perspective, once some kernel and feature map will have been optimized, the problem will be limited to the optimization over the RKHS using the representer theorem[9].

Références

[1] M. A. Nielsen, I. Chuang et N. Raychev, *Quantum Computation and Quantum Information* (oct. 2000).

[2] Y. Du, M.-H. Hsieh, T. Liu et D. Tao, *The Expressive Power of Parameterized Quantum Circuits*, oct. 2018.

[3] J. Mcclean, J. Romero, R. Babbush et A. Aspuru-Guzik, « The theory of variational hybrid quantum-classical algorithms », New Journal of Physics **18** (2015).

[4] M. Schuld, A. Bocharov, K. Svore et N. Wiebe, « Circuit-centric quantum classifiers », (2018).

[5] M. Schuld et F. Petruccione, « Information Encoding », in (août 2018), p. 139-171.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado et J. Dean, « Distributed Representations of Words and Phrases and their Compositionality », Advances in Neural Information Processing Systems **26** (2013).

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga et A. Lerer, « Automatic differentiation in PyTorch », (2017).

[8] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin et N. Killoran, « Pennylane : Automatic differentiation of hybrid quantum-classical computations », arXiv preprint arXiv :1811.04968 (2018).

[9] M. Schuld et N. Killoran, « Quantum machine learning in feature Hilbert spaces », arxiv :1803.07128 (2018).

[10] V. Havlicek, A. D. Corcoles, K. Temme, A. W. Harrow, J. M. Chow et J. M. Gambetta, « Supervised learning with quantum enhanced feature spaces », arxiv :1804.11326 (2018).

[11] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin et N. Killoran, *PennyLane : Automatic differentiation of hybrid quantum-classical computations*, nov. 2018.