Quantum Binary Classification (Student Abstract)

Carla Silva,[1] **Ana Aguiar,**[3,4] **Inês Dutra**[1,2]
[1]Faculty of Sciences, University of Porto, Portugal
[2]CINTESIS - Center for Health Technology and Services Research
[3]Instituto de Telecomunicações
[4]Faculty of Engineering, University of Porto, Portugal
carla.maps@gmail.com, ana.aguiar@fe.up.pt, ines@dcc.fc.up.pt

Abstract

We implement a quantum binary classifier where given a dataset of pairs of training inputs and target outputs our goal is to predict the output of a new input. The script is based in a hybrid scheme inspired in an existing PennyLane's variational classifier and to encode the classical data we resort to PennyLane's *amplitude* encoding embedding template. We use the quantum binary classifier applied to the well known Iris dataset and to a car traffic dataset. Our results show that the quantum approach is capable of performing the task using as few as 2 *qubits*. Accuracies are similar to other quantum machine learning research studies, and as good as the ones produced by classical classifiers.

Introduction

Near-term quantum devices involve random processes and are built to transform the input data following a unitary operation, gate operations and measurements and are described by a quantum circuit. A classical *bit* has a state of either 0 or 1 and is the smallest quantity of non-probabilistic information. The simplest possible quantum system (*2-state system*) can hold precisely one *bit* of information. However, the *qubit* (unit of quantum information) has two possible states $|0\rangle$ and $|1\rangle$ defined as a finite-dimensional quantum system that forms a computational basis - two basis states composed by two distinct quantum states that the *qubit* can be in physically. Fault-tolerant quantum computers use few physical *qubits* to encode each logical *qubit*. These *qubits* are also used for error correction where the logical information is encoded through the relationship of the *qubits*, also known as entanglement. In this work we focus on a quantum binary classifier that resembles a multilayer perceptron (Tiwari and Melucci 2019; Schuld et al. 2018; Schuld and F. 2018).

Quantum Supervised Binary Classification

The inference is performed with the model by initializing a state preparation circuit encoding the input into the amplitudes of the quantum device, resorting to a model circuit U_θ with trained parameters. The optimizer uses a *loss* function and initial parameters. Let $\mathcal{X}$ be the inputs and $\mathcal{Y}$ the outputs (actual instance labels). Given a dataset $D = (x_1, y_1), ..., (x_M, y_M)$ with pairs of training inputs $x^m \in \mathcal{X}$ and target outputs $y^m \in \mathcal{Y}$ for $m = 1, ..., M$, our goal is to predict the output $y \in \mathcal{Y}$ of a new input $x \in \mathcal{X}$. The binary classification task on an *N*-dimensional real input space can be then defined using $\mathcal{X} = \mathbb{R}^N$ and $\mathcal{Y} = \{0, 1\}$.

Algorithm 1 Circuit node

1: **procedure** CIRCUIT($weights$,$features$)
2: AMPLITUDEEMBEDDING($features, qubits = [0, 1], pad = 0.0, normalized$)
3: **for** w in $weights$ **do**
4: LAYER(w)
5: **end for**
6: **return** Expectation value Pauli Z
7: **end procedure**

Algorithm 2 Layer function

1: **procedure** LAYER(w)
2: $Rot(w[0, 0], w[0, 1], w[0, 2], qubits = 0)$
3: $Rot(w[1, 0], w[1, 1], w[1, 2], qubits = 1)$
4: $CNOT(qubits = [0, 1])$
5: **end procedure**

Amplitude Encoding and Quantum Classifier

The embeddings that we can found in PennyLane[1] are templates to encode *features* into a quantum state. The *amplitude* encoding, encodes 2^n features into the amplitude vector of *n qubits* with padded dimension 2^n, Algorithm 1. Our circuit layer (Algorithm 2) consists of rotations on one $qubit$ as well as CNOTs that entangle the $qubit$ with its neighbour, according the *n features* of the dataset D.

Inference is performed with the model $f(x, \theta) = y$ by initialising a *state preparation* circuit $S_\mathcal{X}$ encoding the input $\mathcal{X}$ into the amplitudes of the quantum device, resorting to a model circuit U_θ (parametrised unitary matrix) with classification parameters θ (trained by a variational scheme), and a single *qubit* measurement which gives the probability of the model predicting 0 or 1.

[1]https://pennylane.ai/

Gradient-Descent Optimiser

The optimiser uses a *loss* function and initial parameters, and through differentiation performs the gradient descent in order to choose a set of optimal hyperparameters for the learning algorithm. We resort to the well known Adam gradient-descent optimizer with adaptive learning rate, first and second moment $x^{(t+1)} = x^{(t)} - \eta^{(t+1)} \frac{a^{(t+1)}}{\sqrt{b^{(t+1)}}+\epsilon}$, with the update rules,

$$a^{(t+1)} = \frac{\beta_1 a^{(t)} + (1-\beta_1)\nabla f(x^{(t)})}{(1-\beta_1)},$$

$$b^{(t+1)} = \frac{\beta_2 b^{(t)} + (1-\beta_2)(\nabla f(x^{(t)}))^{\odot 2}}{(1-\beta_2)},$$

$$\eta^{(t+1)} = \eta^{(t)} \frac{\sqrt{(1-\beta_2)}}{(1-\beta_1)}$$

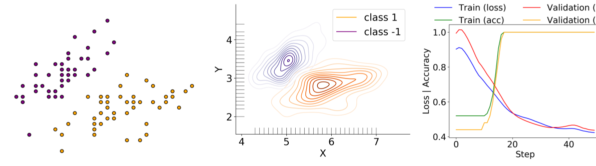
where $(\nabla f(x^{(t-1)}))^{\odot 2}$ refers to the element-wise square operation (each element in the gradient is multiplied by itself) and at start the first and second moment are zero. The *loss* function was built using the standard square loss that measures the distance between target labels and model predictions. The model optimises the weights such that the *loss* function is minimised.

Materials and Methods

In order to model a car traffic problem we search for a data set in an open source repository. The *Traffic* dataset was retrieved from the *R package* (LPCM), package to analyse traffic patterns. It concerns a *fundamental diagram* with observations of speed and flow from 9th of July 2007, 9am, to 10th of July 2007, 10pm, on Lane 5 of the Californian Freeway SR57-N, VDS number 1202263. The original car traffic dataset has a total of 444 samples. The *features* in the dataset are: Lane5Flow, Lane5Speed, Lane5Density. The output variable was created (Lane5congestion) based in two classes resorting to the *fundamental diagram* relationship $q - v$ where we calculate the critical point $v(q_{max})$ (max flow) to obtain the *critical speed* and assign 1 to values below the critical speed (meaning congestion) and -1, otherwise.

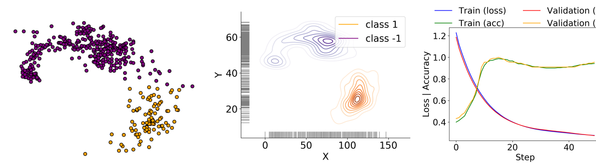
The *Iris* dataset was used in R.A. Fisher's 1936 paper. The original dataset has a total of 150 samples of three species of plants (50 of each). We create two splitted datasets (*Iris A* and *Iris B*), each one with 100 instances, and based in two classes. The *features* in the data sets are: SepalLength, SepalWidth, PetalLength, PetalWidth, and Species is the *target variable*. As is well known, class setosa distinguishes very well from the other two classes, so we should expect almost perfect accuracy results for *Iris A* since we only take the first two classes. Our goal is to map the proposed binary classifier onto quantum simulator, analyse whether the variation in the number of $qubits$ has impact on the results obtained for accuracy in classification. Also, if the results obtained are very unlike if we use a simulator or a real quantum device.

Experiments



(a) *Data points* (b) *Distribution* (c) *Loss/Accuracy*

Figure 1: *Iris A*



(a) *Data points* (b) *Distribution* (c) *Loss/Accuracy*

Figure 2: *Traffic*

Figures 1 and 2 show data distribution and classification results for the *Iris A* and *Traffic*, namely, *loss* and *accuracy*, retrieved from the quantum simulator, considering the training and validation sets. We performed a comparison with a classical neural network which also resulted in *accuracy* equals to 1 for the *Iris A* dataset and 0.982 for *Traffic*.

The circuit parameters were adjusted to maximise the classification *accuracy* and minimise the *loss*. As such, a *gradient descent* is performed to adjust the circuit to minimise the *loss* function. The *loss* function is estimated by iteratively running the model to compare estimated predictions with respect to the *ground truth* (known values of y).

Conclusion and Perspectives

In this work, we resort to *amplitude encoding* where we use the *pad* argument for automated padding. In the future, we will explore other encoding classical data methods, e.g., *angle* which encodes N features into the rotation angles of n $qubits$ where $N \leq n$, *variational/trained* or *higher order* embedding (Lloyd et al. 2020; Havlíček et al. 2019).

References

Havlíček, V.; Córcoles, A. D.; Temme, K.; Harrow, A. W.; Kandala, A.; Chow, J. M.; and Gambetta, J. M. 2019. Supervised learning with quantum-enhanced feature spaces. *Nature* 567(7747): 209–212. ISSN 1476-4687.

Lloyd, S.; Schuld, M.; Ijaz, A.; Izaac, J.; and Killoran, N. 2020. Quantum embeddings for machine learning. *arXiv e-prints* arXiv:2001.03622.

Schuld, M.; Bocharov, A.; Svore, K.; and Wiebe, N. 2018. Circuit-centric quantum classifiers. *arXiv e-prints* arXiv:1804.00633.

Schuld, M.; and F., P. 2018. *Supervised Learning with Quantum Computers*. Springer International Publishing.

Tiwari, P.; and Melucci, M. 2019. Binary Classifier Inspired by Quantum Theory. *Proceedings of the AAAI Conference on Artificial Intelligence* 33(01): 10051–10052.