❮ WOMANIUM | QUANTUM ❯

Womanium Global Quantum +AI Project

Team Name: Qunix   Team Members: M. Barfar , K. Emadzadeh

This file provides a brief explanation of the tutorial “1. Fitting the parity function” and “2.Iris classification” on the link: https://pennylane.ai/qml/demos/tutorial_variational_classifier/

Example 1:

-After importing necessary libraries, the quantum circuit is defined.

-In variational classifiers, we have elementary circuit architectures called "layer" and they are repeated to make the full architecture. This repetition is done in the code with a parameter called "layer_weights".

-The circuit in this example includes 4 qubits and a ring of CNOTs entangled each qubit with its neighbours.

-The measured outputs are needed to be dependent on the inputs, as a result, in this example the inputs are encoded into states of qubit. If the input has 1 in the string, the corresponding quantum state after state preperation, will include 1 as well. This encoding is done with "BasisState" function in PennyLane.

-The full model in this example is defined as a sum of the output of quantum circuit plus a trainable bias which is a classical parameter.

-The cost function in this example is defined as the standard square loss. This function measures the distance between target labels and model predictions.

-The optimizer used for this model is “NesterovMomentumOptimizer” function and the batch size is set to be 5. Overall there are 100 iterations, for each iteration the cost function and the accuracy is tracked. Accuracy is the share of correctly classified data samples. The computation of accuracy is done by computing the outputs of the variational classifier and turning them into predictions in {−1,1} by taking the sign of the output. It is seen that from iteration 51 to 100, the accuracy is 1 and it means that the variational classifier learned to classify all bit strings from the training set correctly.

-An important tip to consider is that the goal is to generalize from limited data to unseen examples and although a variational quantum circuit is perfectly optimized with respect to the cost, it might not generalize, or overfit. As a result, the model must be checked with data which does not exist in training set, called “test” set.

-The measured accuracy for all the elements in the “test set” is 1 which shows quantum circuit has also learnt to predict all unseen examples perfectly well. This is a remarkable result, because the encoding strategy creates quantum states from the data that have zero overlap – and hence the states created from the test set have no overlap with the states created from the training set.

Example 2:

-The data points from the Iris dataset will be classified, which are real-valued vectors not simple bitstrings. The vectors are 2-dimensional, but some “latent dimensions” are added to them and therefore inputs are encoded into 2 qubits. In this example, encoding data is not straightforward as every input x has to be translated into a set of angles which can get fed into a small routine for state preparation.

-The get_angles(x) function, calculates a set of angles based on the input array x. The angles are computed using the arcsine function and some normalization. Beta0 computes an angle based on the first two elements of x. Beta1, computes an angle based on the third and fourth elements of x and Beta2 computes an angle based on the norm of the last two elements of x relative to the norm of the entire array.

-The state_preparation(a) function, prepares a quantum state using the angles provided in the array a. It applies a rotation around the Y-axis on qubit 0, a series of CNOT gates and rotations (around the Y-axis) on qubit 1, and then a Pauli-X gate on qubit 0, followed by more CNOT gates and rotations on qubit 1. After the preparation the method is checked and it is seen that it computed the correct angles to prepare the desired state.

-Like example 1, the layers are defined with “ layer_weights”. For redefining the cost function is square_loss function that calculates the loss between the true labels Y and the predicted labels predictions. It is mentioned that it is somehow like “NumPy broadcasting”. This “NumPy broadcasting” method allows performing element-wise operations on arrays of different shapes without explicitly reshaping them. This feature makes the code more efficient and readable.

- For the data preparation step, the code pads each vector in X with two constant values (0.1) to increase the size to 4 (since (2^2 = 4)). It then prints the first padded sample. In Quantum Machine Learning (QML), padding is a technique used to adjust the size of input data to match the requirements of quantum algorithms. It is often necessary because quantum algorithms typically operate on data sizes that are powers of two (e.g., 2, 4, 8, 16, etc.). Padding ensures that the input data conforms to these requirements by adding extra values to the data.

-Each padded vector is normalized to have a unit norm. The code calculates the normalization factor and applies it to each vector, then prints the first normalized sample. In the next step, angles for state preparation is calculated using the get_angles function for each normalized vector (the angles are used as the new features). These angles are stored in features and the first set of features is printed. Finally, the code extracts the labels (last column) from the data and stores them in Y.

-The optimization function in this part is “NesterovMomentumOptimizer” and the batch size is set to be 5. After 60 iterations in can be seen that accuracy becomes 1.

-The plot results show that the variational classifier learnt a separating line between the datapoints of the two different classes, which allows it to classify even the unseen validation data with perfect accuracy.