Variational Quantum Classifiers

August 1, 2022

Contents

1 Overview

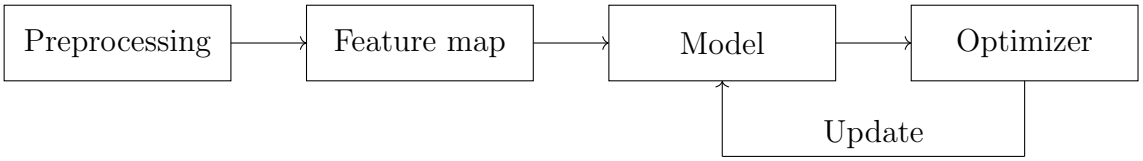
Quantum machine learning, in the context being discussed, is typically just expressed as a quantum circuit framework that processes and produces output for classical data sources. This is desirable because classical machine learning takes a large number of computational resources to accomplish a viable output. While the toll of methods such as gradient descent may be decreased with strategies like stochastic gradient descent, which is able to

produce an output faster in certain cases, these results are not always guaranteed; quantum computing provides a strategy with which we can accomplish machine learning more effectively.

1.1 Goals

The goal of a variational classifier is simple; we take in a set of data, send it through a parameterized operation that is then trained by an optimizer. The fundamental circuit is shaped of simple, differentiated components that feed each other inputs from their own outputs.

1.2 Circuit Framework



2 Preprocessing

All that is required with preprocessing is to obtain a reliable data set and organize it into a usable format. This is achieved through classical means by filtering out entries by certain criteria such that the training data set is optimal to use. In any regular dataset, the features can be ranked with respect to its correlation with the target variable; this will allow us to choose the most relevant features of the dataset to represent as a quantum state.

3 State Preparation

In order to allow a quantum circuit to work with classical data, one must encode the data into a quantum state that can be utilized by the quantum computer. The way this task can be achieved is through data embedding; moving classical data into a quantum state in Hilbert space so that it can be accessed and used by the quantum computer.

3.1 Feature Map

A feature map is a construct that allows the embedding of data into higher-dimensional spaces (or quantum states). It is modeled as a parametrized quantum circuit, where the inputs correspond to the parameters in the operation. Effectively, what we want is some data point vector $\overrightarrow{x}$ to correspond:

$$\overrightarrow{x} \rightarrow |\phi(\overrightarrow{x})\rangle$$

where ϕ denotes some function on a ground state qubit that is parametrized by the original data point. In general terms, a feature map ϕ performs:

$$\phi : \mathbb{R}^N \rightarrow \mathbb{C}^{2^n}$$

The basic motivation behind the usage of a feature map is that nonlinear operations that are typically desired when encoding data for machine learning purposes are hard to realize in a strongly linear system like quantum computing; the feature map shifts the burden of producing nonlinearity into the procedure of encoding classical inputs to a quantum state.

3.2 Basic Embedding

The most naive of embedding schemes, basic embedding simply associates each binary input with a bit-wise translation of the input into corresponding states in the quantum system. The bitstring $x = 101$, for example, can be expressed with three qubits in a quantum circuit as the state $|101\rangle$. If we have a dataset $\mathcal{D}$ containing M entries with N bits for each entry, then the entry x can directly be mapped to $|x\rangle$. The dataset can be represented in such a superposition:

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle$$

However, for N bits, there are 2^N possible states; we are wasting a lot of possible bit-space by using basic embedding.

3.3 Amplitude Embedding

In the amplitude encoding scheme, data from each entry is encoded into the amplitudes of a quantum state. By normalizing the N-dimensional vector

entry x, it can be represented by the amplitudes of a quantum state $|\psi_x\rangle$:

$$|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$$

Note that $x_{norm} = 1$; this condition must be fulfilled for the state to be a valid quantum encoding. Considering the dataset $\mathcal{D}$, the amplitude embedding can be understood as a concatenation of the inputs $x^{(m)}$ into one vector:

$$C = A\langle x_1^{(1)}, \cdots x_N^{(1)}, x_1^{(2)}, \cdots x_N^{(2)}, \cdots \rangle$$

where A represents the constant that normalizes the vector. The input is understood in the computational basis as

$$|\mathcal{D}\rangle = \sum_{i=1}^{2^n} C_i |i\rangle$$

In our approach, we will require that the size of the input N is a power of 2 so that we may associate it directly with an amplitude vector in the computational basis. If N is not a power of 2, we can add supplementary entries that pad the original input. We can denote these with $c_1, c_2, \cdots c_D$, in which all of them are 0. So, we can transform

$$(x_1, \cdots x_N) \rightarrow A(x_1, \cdots x_N, c_1, \cdots c_D)$$

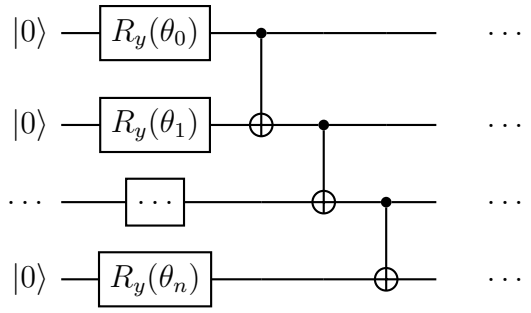
where A is the normalization factor,

$$A = \frac{1}{\sqrt{\sum_j^N x_j^2 + \sum_k^D |c_k|^2}}$$

4 Variational Circuit

After the data encoding, we move on to the actual variational circuit. This circuit is some sort of block operator that depends on a set of parameters, denoted within this writeup as a vector $\overrightarrow{\theta}$. There exist many different types of variation circuits, but many of them are similar in structure to another; they have specific components that work to achieve similar results. Typically, the variational circuit consists of parameterized rotation gates about

some axis; in sticking to the real domain, the R_y gate will be used. Then, entanglement gates are applied such that most of the qubits are connected in some way to each other. The block can then be repeated some number of times to introduce more parameters into the model. In essence, the basic circuit looks like:



5 Classification

In between the circuit and the optimizer, we must somehow measure and classify our quantum data into a form that can be broken down by the optimizer to apply to the parameters. A very simple way to do this is to simply measure the first qubit in the register, and then assign a binary label based on that measurement. Another approach is to take the parity of the bits in the measured bitstring, and assign labels with the result of the parity function. Both are viable ways to approach this problem.
We can measure our quantum circuit with a certain number of shots to obtain a probability distribution of measuring a basis state, or in the case of a single qubit measurement, of obtaining a $|0\rangle$ or a $|1\rangle$. Based on the distribution, we can determine with what percent probability the statevector evolved through the variational circuit will be measured into either class of states.

6 Optimizer

The optimizing mechanisms of the VQC are what allow the algorithm to "evolve" and find the optimal set of parameters for the quantum machine. In classical computing, a **gradient descent** algorithm is performed iteratively to "locate" the most optimal set of parameters for a given cost function. In essence, gradient descent is about finding the absolute minima of a function

within some bounds.
The gradient (or slope) is simply a vector of partial derivatives; or, in a univariate case, the first derivative at a certain point. More formally,

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

Computers cannot necessarily compute derivatives on the fly; this is why the gradient descent algorithm itself is iterative. Intuitively, it just takes the gradient at the current position of our "guess", scales it by some learning rate η, and subtracts that value from the current position to go "down" the slope to a new position. Mathematically, it's expressed as

$$p_{n+1} = p_n - \eta \nabla f(p_n)$$

The graphical process can be expressed in a sort of flow chart.