Understanding Naive Quantum Bayes Classifier (Naive-QBC)

Summary: A short summary of our implementation of a Quantum Bayes Classifier. Naive-QBC leverages quantum computing to perform classification tasks using Bayes' rule. In this classifier, each node in the Bayesian network is represented by a qubit, and quantum superposition is used to represent the probabilities of different label values given various attribute combinations.

Key Steps in Naive-QBC Construction

Initialization

First, all qubits are initialized to the $|0\rangle$ state.

Encoding Class-Prior Probabilities

Next, we encode the class-prior probability $P(y = 0)$ using a rotation operation $R_y(\theta)$, where $\theta = 2 \arccos \sqrt{P(y = 0)}$.

Encoding Class-Conditional Probabilities

For each attribute node x_i, we encode the class-conditional probabilities $P(x_i = 0 \mid y = 1)$ and $P(x_i = 0 \mid y = 0)$ using controlled rotation operations $CR_y(\theta)$.

Advantages of Naive-QBC Over Classical Methods

Parallelism

Quantum computing leverages superposition to evaluate multiple probabilities simultaneously. In a classical approach, each probability calculation is sequential, whereas a quantum approach can handle these in parallel.

Efficiency

For large datasets with high-dimensional feature spaces, the quantum approach can be more efficient. Classical methods might face combinatorial explosion with many features, but quantum circuits can manage these using fewer resources.

Scalability

Quantum circuits can scale more efficiently with the number of attributes. As more attributes are added, the quantum circuit complexity grows linearly in terms of the number of qubits, compared to the exponential growth in classical computation.

Amplitude Encoding

Probabilities are encoded in the amplitudes of quantum states. This allows for efficient manipulation and computation, reducing the overall computational complexity.

Feature Binarization in Quantum Bayes Classifiers (QBC)

Feature binarization refers to the process of converting continuous variables into binary variables (0 or 1) to be compatible with quantum computing, which uses qubits that represent two states. This process is essential for applying Bayesian classifiers within quantum computing frameworks like QBCs, where each feature must be represented by a single qubit.

Understanding the Process

In the context provided, the continuous attribute x_i is assumed to follow a normal distribution conditioned on the class label y:

$$p(x_i \mid y = 0) \sim N(\mu_{y0,i}, \sigma^2_{y0,i})$$

$$p(x_i \mid y = 1) \sim N(\mu_{y1,i}, \sigma^2_{y1,i})$$

Steps for Binarization

Calculate Intersection Points

Determine the points where the probability density functions (PDFs) of the two classes intersect. These intersection points are denoted as x_{ins}, x_{ins1}, and x_{ins2}.

Single Intersection Case

If there is only one intersection point x_{ins}:

- The attribute x_i is set to 0 if the value x'_i (which can be an average or a specific measurement) is less than or equal to x_{ins} and the mean of the first class $\mu_{y0,i}$ is less than or equal to the mean of the second class $\mu_{y1,i}$. Otherwise, x_i is set to 1.
- If $\mu_{y0,i} > \mu_{y1,i}$, the attribute x_i is set to 1 if x'_i is less than or equal to x_{ins} and the mean of the first class is greater. Otherwise, x_i is set to 0.

Two Intersections Case

If there are two intersection points x_{ins1} and x_{ins2} where $x_{ins1} \leq x_{ins2}$:

- The attribute x_i is set to 0 if x_i' is less than or equal to x_{ins1} or falls between x_{ins1} and x_{ins2} and is closer to x_{ins1}. Otherwise, x_i is set to 1.
- If $\mu_{y0,i} > \mu_{y1,i}$, the attribute x_i is set to 1 if x_i' is less than or equal to x_{ins1} or falls between x_{ins1} and x_{ins2} and is closer to x_{ins1}. Otherwise, x_i is set to 0.

Intuition Behind Binarization

The intuition behind this binarization process is to transform the continuous variable x_i into a binary form that best represents the underlying class distribution while maintaining the discriminatory power of the attribute. Here's why and when to set x_i to 0 or 1:

Single Intersection Point

The single intersection point x_{ins} is a threshold where the likelihood of the attribute belonging to either class changes. By setting $x_i = 0$ or 1 based on this threshold, we capture the point where the probability of class membership is equal.

Two Intersection Points

When there are two intersection points, the regions between and outside these points help to better discriminate between the classes. By evaluating the relative position of x_i' within these regions, we binarize x_i to reflect the class-conditional probabilities more accurately.

Local Sampling

Local sampling refers to how the feature values are processed and transformed into binary features using local operations on the image data.

Convolution and Average Pooling

The code uses a convolution operation with an average kernel to reduce the dimensionality of the image. This is a form of local operation, as the kernel moves across the image and computes local averages. The `F.conv2d` function with an averaging kernel effectively performs a local sampling by considering 8x8 patches of the image and pooling their values.

Feature Binarization

The `inference_features` method binarizes the features based on the intersection points derived from the means and variances of the classes. This method involves comparing the locally pooled feature values against thresholds.

Results

Currently, our classifier achieves the following accuracy:

- **Classical Accuracy:** 0.9730
- **Quantum Accuracy:** 0.6544

We are focusing on optimizing the circuit and tweaking the hyperparameters to improve the accuracy of the quantum classifier.

Future Directions

Future directions for this research include:

- Expanding the classifier to handle multi-class classification.
- Applying the classifier to further datasets, such as images of letters (A, B, C, D) and digits (0 through 9).