Rain-Runoff Modeling with Physics-Informed Neural Networks (PINN) and Quantum PINN (QPINN)

Soumyadip Das

August 9, 2024

1 Overview

This document contains the implementation of **Physics-Informed Neural Networks (PINNs)** and **Quantum Physics-Informed Neural Networks (QPINNs)** applied to a rain-runoff model. This model is essential for hydrological studies, flood prediction, and water resource management. The PINNs integrate physical laws directly into the training process, while QPINNs aim to leverage quantum computing techniques for enhanced performance and accuracy.

2 Project Structure

- `rainrunoffmodel.ipynb`: Implementation of the rain-runoff model using Physics-Informed Neural Networks (PINNs). Implementation of the rain-runoff model using Quantum Physics-Informed Neural Networks (QPINNs).Script for generating synthetic soil moisture data for training.
- `README.md`: Project documentation.

3 Model Overview

3.1 Rain-Runoff Model: Mathematical Formulation

The rain-runoff process is modeled by the differential equation:

$$\frac{dS}{dt} = P(t) - I(t) - ET(t) - R(S) \tag{1}$$

where:

- $S(t)$: Soil moisture at time t.

- $P(t)$: Precipitation rate.
- $I(t)$: Infiltration rate (modeled using the Horton Infiltration Model).
- $ET(t)$: Evapotranspiration rate (assumed constant).
- $R(S)$: Runoff rate, which depends on the soil moisture S.

3.2 Synthetic Data Generation

Synthetic soil moisture data for a 100-hour period is generated using the rain-runoff differential equation. This data serves as the target for training both PINNs and QPINNs.

4 Physics-Informed Neural Networks (PINN)

4.1 PINN Architecture

The PINN model is designed as follows:

- **Input:** Time t.
- **Hidden Layers:** Two fully connected layers with 10 neurons each, using the Tanh activation function.
- **Output:** Predicted soil moisture $\hat{S}(t)$.

4.2 Loss Function

The total loss function for PINNs is given by:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{physics}} \tag{2}$$

where:

- $\mathcal{L}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{S}(t_i) - S_{\text{true}}(t_i) \right)^2$ is the mean squared error between predicted and true soil moisture values.
- $\mathcal{L}_{\text{physics}}$ enforces the differential equation constraint. It ensures the predicted $\hat{S}(t)$ satisfies:

$$\mathcal{L}_{\text{physics}} = \frac{1}{M} \sum_{j=1}^{M} \left(\frac{d\hat{S}(t_j)}{dt} - P(t_j) + I(t_j) + ET(t_j) + R(\hat{S}(t_j)) \right)^2 \tag{3}$$

5 Quantum Physics-Informed Neural Networks (QPINN)

5.1 QPINN Architecture

The QPINN model integrates quantum circuits into the neural network structure:

1. **Input Layer:** Time t, transformed by a linear layer.
2. **Quantum Layer:** Quantum circuit involving:
   - **Angle Encoding:** Maps classical inputs to quantum states using parameterized angles.
   - **Strongly Entangling Layers:** Employs multiple layers of controlled gates to entangle qubits.
   - **Measurement:** Measures the Pauli-Z operator to extract information from quantum states.
3. **Output Layer:** Processes quantum measurements through a linear layer to predict soil moisture $\hat{S}(t)$.

5.2 Quantum Circuit Details

Angle Encoding:

$$|\psi(t)\rangle = \exp\left(-i \sum_k \theta_k(t) H_k\right) |0\rangle \tag{4}$$

where $\theta_k(t)$ are parameters encoding time t and H_k are rotation operators.

Entangling Layers:

The circuit applies controlled gates to entangle qubits, creating complex quantum states that represent different possible solutions.

Measurement:

The measurement of the Pauli-Z operator provides output probabilities used to compute the predicted soil moisture:

$$\hat{S}(t) = \frac{1}{N} \sum_{i=1}^{N} \langle Z_i \rangle \tag{5}$$

5.3 Loss Function

The loss function for QPINNs focuses on minimizing the prediction error:

$$\mathcal{L}_{\text{QPINN}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{S}(t_i) - S_{\text{true}}(t_i)\right)^2 \tag{6}$$

Incorporating physics-informed constraints into QPINNs remains an area for future work.

6 Results

The models were trained on the synthetic dataset, yielding the following results:

6.1 PINN

- **Final MSE:** 17.0180
- **Training Time:** 1.68 seconds

6.2 QPINN

- **Final MSE:** 600.4258
- **Training Time:** 1914.86 seconds

6.3 Analysis

- **PINN** demonstrated superior accuracy and efficiency compared to **QPINN**.
- **QPINN** showed potential but needs improvements in quantum circuit design and parameter tuning.

7 Conclusion

This project explores the integration of PINNs and QPINNs with the rain-runoff model. While PINNs achieved strong performance, QPINNs does not offer promising experimental results highlighting the need for further development in quantum modeling techniques.

8 Future Work

- **Enhance Quantum Circuit Expressivity:** Test deeper quantum circuits and increased qubit counts.
- **Incorporate Physics-Informed Loss in QPINNs:** Develop methods to integrate physical constraints into QPINNs.
- **Optimize Training Parameters:** Adjust learning rates and initialization strategies for better convergence.

9 Requirements

- Python 3.x
- PyTorch
- PennyLane
- NumPy
- Matplotlib

10 Contact

For questions or collaboration, please contact Soumyadip Das at soumyadip-das321@gmail.com.