Quantum Backpropagation (QBP) for Training a Variational Quantum Circuit for Supervised Learning

Amit LeVi

November 15, 2024

Training a Variational Quantum Circuit for Supervised Learning We're exploring how to train a variational quantum circuit (VQC) for a given supervised learning model using a dataset $\{(X_i, y_i)\}$.

Problem Definition

Objective: Optimize the parameters $\vec{\theta}$ of a VQC $U(\vec{\theta})$ to minimize a loss function $\mathcal{L}(\vec{\theta})$ over a dataset $\{(X_i, y_i)\}$, where:

- X_i: Input data samples.
- y_i: Corresponding target labels.

—

Algorithm Steps

1. Data Encoding

Goal: Encode classical input data X_i into quantum states $|\psi(X_i)\rangle$.
Methods:

- **Amplitude Encoding**: Use the amplitudes of the quantum state to represent the data.
- **Basis Encoding**: Map data to the basis states of qubits.
- **Angle Encoding**: Encode data into rotation angles of quantum gates.

Example using Angle Encoding:
For each feature $x_{i,j}$ of X_i, apply a rotation gate to qubit j:

$$|\psi(X_i)\rangle = \bigotimes_j R_Y(x_{i,j})\,|0\rangle$$

- $R_Y(\theta) = e^{-i\theta Y/2}$ is the rotation around the Y-axis.
- $|0\rangle$ is the initial state of each qubit.

2. Variational Quantum Circuit

Circuit Structure:

- **Parameterized Gates**: Gates that depend on parameters $\vec{\theta}$, such as $R_Y(\theta_k)$ or $R_Z(\theta_k)$.
- **Entangling Gates**: Gates like CNOT to create entanglement between qubits.

Circuit Application:

1. **State Preparation**: Start with $|\psi(X_i)\rangle$.
2. **Apply** $U(\vec{\theta})$: Sequence of parameterized and entangling gates.
3. **Resulting State**: $\left|\phi_i(\vec{\theta})\right\rangle = U(\vec{\theta})\,|\psi(X_i)\rangle$.

3. Measurement and Output

Measurement Operator O:

- Usually a Hermitian operator corresponding to an observable.
- Commonly, measurements are performed on one or more qubits.

Compute Model Output:

$$f_{\vec{\theta}}(X_i) = \left\langle \phi_i(\vec{\theta}) \middle| O \middle| \phi_i(\vec{\theta}) \right\rangle$$

This is the expected value of O after applying the circuit to $|\psi(X_i)\rangle$.

4. Loss Function Definition

For Regression Tasks:
Mean Squared Error (MSE):

$$\mathcal{L}(\vec{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\big(y_i - f_{\vec{\theta}}(X_i)\big)^2$$

For Classification Tasks:
Cross-Entropy Loss:

$$\mathcal{L}(\vec{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \ln p_{\vec{\theta}}(X_i) + (1-y_i)\ln\big(1-p_{\vec{\theta}}(X_i)\big)\right]$$

- $p_{\vec{\theta}}(X_i)$ is the probability obtained from measuring $\left|\phi_i(\vec{\theta})\right\rangle$.

5. Gradient Computation

Parameter-Shift Rule for Quantum Gradients:

For each parameter θ_k:

1. **Shifted Parameters:**

$$\vec{\theta}_k^{\pm} = \vec{\theta} \pm \frac{\pi}{2}\vec{e}_k$$

   - $\vec{e}_k$ is a unit vector with 1 at position k and 0 elsewhere.

2. **Compute Shifted Outputs:**

$$f_{\vec{\theta}_k^{\pm}}(X_i) = \langle \psi(X_i) | \, U^\dagger(\vec{\theta}_k^{\pm}) O U(\vec{\theta}_k^{\pm}) \, | \psi(X_i) \rangle$$

3. **Compute Quantum Gradient:**

$$\frac{\partial f_{\vec{\theta}}(X_i)}{\partial \theta_k} = \frac{1}{2}\left[f_{\vec{\theta}_k^{+}}(X_i) - f_{\vec{\theta}_k^{-}}(X_i) \right]$$

4. **Compute Loss Gradient:**
   For loss $\ell(y_i, f_{\vec{\theta}}(X_i))$:

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \ell(y_i, f_{\vec{\theta}}(X_i))}{\partial f_{\vec{\theta}}(X_i)} \cdot \frac{\partial f_{\vec{\theta}}(X_i)}{\partial \theta_k}$$

   - For MSE:

$$\frac{\partial \ell}{\partial f_{\vec{\theta}}(X_i)} = -2(y_i - f_{\vec{\theta}}(X_i))$$

   - For Cross-Entropy:

$$\frac{\partial \ell}{\partial f_{\vec{\theta}}(X_i)} = -\left(\frac{y_i}{p_{\vec{\theta}}(X_i)} - \frac{1 - y_i}{1 - p_{\vec{\theta}}(X_i)} \right) \cdot \frac{\partial p_{\vec{\theta}}(X_i)}{\partial f_{\vec{\theta}}(X_i)}$$

6. Parameter Update with Optimizer

Using the Adam Optimizer:

- **Initialize:**
  - $m_k = 0$ (First moment estimate)
  - $v_k = 0$ (Second moment estimate)
  - $t = 0$ (Time step)

- Hyperparameters: η (learning rate), β_1, β_2 (decay rates), ϵ (small constant)

- **At Each Iteration:**

  1. **Increment Time Step:**

  $$t = t + 1$$

  2. **Compute Gradients $\frac{\partial \mathcal{L}}{\partial \theta_k}$ for all k as above.**
  3. **Update First Moment Estimate:**

  $$m_k = \beta_1 m_k + (1 - \beta_1)\frac{\partial \mathcal{L}}{\partial \theta_k}$$

  4. **Update Second Moment Estimate:**

  $$v_k = \beta_2 v_k + (1 - \beta_2)\left(\frac{\partial \mathcal{L}}{\partial \theta_k}\right)^2$$

  5. **Compute Bias-Corrected Estimates:**

  $$\hat{m}_k = \frac{m_k}{1 - \beta_1^t}, \quad \hat{v}_k = \frac{v_k}{1 - \beta_2^t}$$

  6. **Update Parameters:**

  $$\theta_k \leftarrow \theta_k - \eta \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}$$

7. Convergence Check

Stopping Criteria:

- **Maximum Iterations**: Stop after a set number of iterations.
- **Loss Threshold**: Stop when $\mathcal{L}(\vec{\theta})$ is below a predefined threshold.
- **Parameter Change**: Stop if $|\theta_k^{\text{new}} - \theta_k^{\text{old}}|$ is below a small value for all k.

Quantum Backpropagation (QBP)

```
Initialize parameters  randomly
Initialize first moment m = 0 and second moment v = 0
Set hyperparameters , 1, 2,
Set time step t = 0

repeat until convergence:
    t = t + 1

    for each data point (X_i, y_i):
        Prepare quantum state |(X_i)⟩ from input X_i

        for each parameter _k:
            // Compute shifted parameters
            _plus =  + (/2) * e_k
            _minus =  - (/2) * e_k
            // Apply quantum circuit with shifted parameters
            f_plus = Measure O after applying U(_plus) to |(X_i)⟩
            f_minus = Measure O after applying U(_minus) to |(X_i)⟩
            // Compute gradient of the model output
            f/_k = (f_plus - f_minus) / 2
            // Store f/_k for gradient computation
        // Compute gradient of loss function
        Compute /f based on loss function (y_i, f_(X_i))

        // Compute gradient of loss with respect to parameters
        For each _k:
            /_k = (/f) * (f/_k)

    // Average gradients over all data points
    For each _k:
        g_k = (1/N) * _i _i/_k

        // Update moments
        m_k = 1 * m_k + (1 - 1) * g_k
        v_k = 2 * v_k + (1 - 2) * (g_k)^2

        // Bias correction
        m_hat_k = m_k / (1 - 1^t)
        v_hat_k = v_k / (1 - 2^t)

        // Update parameters
        _k = _k -  * m_hat_k / (sqrt(v_hat_k) + )
```

Mathematical Proof of Correctness

Parameter-Shift Rule Correctness

Assumptions:

- The gate $U_k(\theta_k) = e^{-i\theta_k G_k}$, where G_k is a Hermitian operator with eigenvalues ± 1.
- The cost function $f_{\vec{\theta}}(X_i) = \langle \psi(X_i) | U^\dagger(\vec{\theta}) O U(\vec{\theta}) | \psi(X_i) \rangle$ is differentiable with respect to θ_k.

Proof:

1. **Derivative of $f_{\vec{\theta}}(X_i)$:**

$$\frac{\partial f_{\vec{\theta}}(X_i)}{\partial \theta_k} = \langle \psi(X_i) | \frac{\partial U^\dagger(\vec{\theta})}{\partial \theta_k} O U(\vec{\theta}) + U^\dagger(\vec{\theta}) O \frac{\partial U(\vec{\theta})}{\partial \theta_k} | \psi(X_i) \rangle$$

2. **Derivative of $U(\vec{\theta})$:**

$$\frac{\partial U(\vec{\theta})}{\partial \theta_k} = -iU_1 \cdots U_{k-1} G_k U_k(\theta_k) U_{k+1} \cdots U_N$$

3. **Simplify the Expression:**
   Let $\left|\phi_i(\vec{\theta})\right\rangle = U(\vec{\theta}) |\psi(X_i)\rangle$, then:

$$\frac{\partial f_{\vec{\theta}}(X_i)}{\partial \theta_k} = -i \left\langle \phi_i(\vec{\theta}) \right| [G_k, O] \left| \phi_i(\vec{\theta}) \right\rangle$$

4. **Express Using Shifted Parameters:**
   Since $G_k^2 = I$ and eigenvalues are ± 1, we have:

$$\frac{\partial f_{\vec{\theta}}(X_i)}{\partial \theta_k} = \frac{1}{2} \left[f_{\vec{\theta}_k^+}(X_i) - f_{\vec{\theta}_k^-}(X_i) \right]$$

   This shows that the parameter-shift rule gives the exact gradient.

Convergence of the Optimizer

- **Adam Optimizer** is proven to converge under certain conditions (bounded gradients, appropriate hyperparameters).
- **Assumptions in Quantum Context:**
  - **Bounded Gradients:** Quantum measurements yield bounded outputs; thus, gradients are bounded.

- **Smoothness:** The cost function is smooth due to the nature of quantum gates and measurements.

Conclusion:

Using the parameter-shift rule and Adam optimizer, the algorithm converges to a set of parameters $\vec{\theta}^*$ that minimize the loss function $\mathcal{L}(\vec{\theta})$.

—

Memory Requirements

- **Classical Memory:**
  - Store parameters $\vec{\theta}$, gradients $\vec{g}$, and optimizer variables $\vec{m}, \vec{v}$.
  - Memory complexity: $\mathcal{O}(N)$, where N is the number of parameters.
- **Quantum Memory:**
  - Quantum circuits are executed per data point and per parameter shift.
  - Quantum memory is used per circuit execution but does not accumulate.
  - Memory complexity depends on the number of qubits and circuit depth, not on N or dataset size.

—

Summary

By following this algorithm:

- **We encode classical data X_i into quantum states $|\psi(X_i)\rangle$.**
- **We apply a parameterized quantum circuit $U(\vec{\theta})$ and measure to get outputs $f_{\vec{\theta}}(X_i)$.**
- **We define a suitable loss function $\mathcal{L}(\vec{\theta})$ and compute its gradients with respect to the parameters using the parameter-shift rule.**
- **We update the parameters using the Adam optimizer to minimize the loss.**

This full algorithm allows us to train a VQC for supervised learning tasks using datasets $\{(X_i, y_i)\}$, bridging the gap between classical data and quantum computing capabilities.

Quantum Gradient Estimation with Grover's Algorithm:

- State Preparation: $\mathcal{O}(N)$ - Circuit Execution: $\mathcal{O}(Nd)$ - Gradient Estimation: $\mathcal{O}(P\sqrt{N})$ - Synchronization: $\mathcal{O}(P \log Q)$ - Adam Optimization: $\mathcal{O}(P)$
Total Quantum Complexity:

$$\mathcal{O}(N) + \mathcal{O}(Nd) + \mathcal{O}(P\sqrt{N}) + \mathcal{O}(P \log Q) + \mathcal{O}(P)$$

Classical Back-propagation

$$\mathcal{O}\left(\sum_{l=1}^{L} n_{l-1} n_l\right) + \mathcal{O}(P)$$