Lecture

Quantum Computing
(CS5070)

Quantum Machine Learning: Optimization

Professor Dr. rer. nat. habil. Sven Groppe

https://www.ifis.uni-luebeck.de/index.php?id=groppe
Quantum Model

1. Encode classical data $x$ into a quantum state $|\psi(x)\rangle = \mathcal{E}(x)|0\rangle$
2. Apply a parametrized unitary $|U_{\psi}(x, \Theta)\rangle = U(\Theta)|\psi(x)\rangle$
3. Measure classical data $z = (z_1, \ldots, z_n)$
4. Do classical post-processing to get prediction $\hat{y} = g(z)$
5. Use optimization techniques (like gradient descent) to optimize $\Theta$
Supervised Learning: Cost

- **Classify** whether the picture contains a *cat* or *dog* based on certain features of animals:

  3. For the $t$ samples in the training dataset compute cost with
     
     a. Mean Squared Error (MSE): 
     $$C_{MSE} = \frac{1}{t} \cdot \sum_{i=1}^{t} (\hat{y}_i - y_i)^2$$
     
     b. Binary classification (e.g., "cat" or "dog") using cross-entropy loss:
     
     $$C_{CE}(\Theta, (y_1, \ldots, y_t)) = -\frac{1}{t} \sum_{i=1}^{t} (y_i \cdot \log(f(x_i, \Theta)) + (1 - y_i) \cdot \log(1 - f(x_i, \Theta)))$$

  4. ... 

**Minimize** the chosen cost function to have a good model $f(x_i, \Theta)$
How to optimize the model?

- Being on a mountain in fog (only seeing some few meters): Where to go to be on the fastest route to the valley?
How to optimize the model?

To be determined: Global Minimum!

$C_1$

$C_2$

$C_{min}$

$\theta_{min}$

$[0.1]$ $[0.8]$

$[0.5]$ $[2.0]$
How to optimize the model?

Gradient descent algorithm:
\[ \Theta_{i+1} = \Theta_i - \eta \cdot \nabla C(\Theta_i) \]
where \( \eta \) is the learning rate
(smaller for larger \( i \))
How to optimize the model?

Gradient descent algorithm:

$$\Theta_{i+1} = \Theta_i - \eta \cdot \nabla C(\Theta_i)$$

where $\eta$ is the learning rate (smaller for larger $i$)
How to optimize the model?

- Cost functions are often complicated in practice!

Loss surface of ResNet-56 (residual network with 56 layers)

- In practice: Stochastic gradient-descent methods usually avoid local minima
How to optimize the model?

Training Data: $X_{\text{train}}$

Feed in the samples $(x_1, \ldots, x_n)$
How to optimize the model?

Training Data: \( X_{\text{train}} \)

Model: \( f(X_{\text{train}}, \Theta) = \hat{y} \)

Pick random parameters \( \Theta \)

Compute predicted labels \((\hat{y}_1, \ldots, \hat{y}_n)\)
How to optimize the model?

Training Data: $X_{train}$

Model: $f(X_{train}, \Theta) = \hat{y}$

Cost: $C(f(X_{train}, \Theta), y)$

Compute cost from predicted label $\hat{y}$ and ground truth $y$
How to optimize the model?

Training Data: $X_{train}$

Model: $f(X_{train}, \Theta) = \hat{y}$

Cost: $C(f(X_{train}, \Theta), y)$

Gradient: $\nabla_\Theta C(f(X_{train}, \Theta), y)$

Compute gradient of the cost function for the samples of the dataset

Adapted from A. Dekusar, J. Gacon, Qiskit Machine Learning: Quantum algorithms for supervised learning, EQTC, 2022
How to optimize the model?

Training Data: $X_{\text{train}}$

Model: $f(X_{\text{train}}, \theta) = \hat{y}$

Cost: $C(f(X_{\text{train}}, \theta), y)$

Update: $\theta$

Gradient: $\nabla_{\theta} C(f(X_{\text{train}}, \theta), y)$

Update parameters $\theta$ and start the next iteration of the optimization process.
How to optimize the model?

Training Data: $X_{train}$

Quantum Model: $f(X_{train}, \Theta) = \hat{y}$

Cost: $C(f(X_{train}, \Theta), y)$

Update: $\Theta$

Gradient: $\nabla_{\Theta} C(f(X_{train}, \Theta), y)$

How to compute gradients on quantum computer?
Optimization: **Gradient** Calculation

- **Example:** Binary classification using cross-entropy loss:

  \[ C_{CE}(\Theta, (y_1, ..., y_t)) = -\frac{1}{t} \sum_{i=1}^{t} (y_i \cdot \log(f(x_i, \Theta)) + (1 - y_i) \cdot \log(1 - f(x_i, \Theta))) \]

  Probability for \( y_i = 1 \) (e.g., "cat")

  Case \( y_i = 1 \), e.g. "cat"

  Probability for \( y_i = 0 \)

  (e.g., "dog")

  Case \( y_i = 0 \), e.g. "dog"

- **Gradient** Calculation:

  Let \( \Theta = \left[ \Theta[1] \vdots \Theta[r] \right] \), then

  \[
  \frac{\partial C}{\partial \Theta} = \left[ \frac{\partial C}{\partial \Theta[1]} \vdots \frac{\partial C}{\partial \Theta[r]} \right]
  \]

  and

  \[
  \frac{\partial C_{CE}}{\partial \Theta[k]} = -\sum_{i=1}^{t} \left( \frac{y_i}{f(x_i, \Theta)} - \frac{1 - y_i}{1 - f(x_i, \Theta)} \right) \cdot \frac{\partial f(x_i, \Theta)}{\partial \Theta[k]}
  \]

  to be evaluated on the quantum computer!
Quantum Gradient Calculation

- For expectation- & sample-based labels to be evaluated:
  a) \( \frac{\partial}{\partial \Theta_k} \langle \psi(x_i, \Theta) | Z | \psi(x_i, \Theta) \rangle \)
  b) \( |\psi(x_i, \Theta)\rangle = U(\Theta) |E(x_i)|0\rangle = U|\psi(x)\rangle \)

- Options to determine gradient [S+’18]
  - Numerical differentiation
    - The gradient is approximated by black-box evaluations, i.e., in general \( \nabla g(x) \approx \frac{g(x+s) - g(x-s)}{2\cdot s} \), where \( s \) is small shift
    - In quantum model via parameter shift rule:

- Symbolic differentiation
  - \( \nabla g \) via manual calculations or symbolic computer algebra package

- Automatic differentiation of \( g(x) := g_1(g_2(\cdots g_v(x) \cdots)) \)
  - \( \nabla g \) via accumulation of derivatives of \( g_1 \cdots g_v \) following the chain rule [R’96]
Quantum Gradient Calculation

- Quantum computing frameworks **often support many different approaches** for gradient calculations
  - Qiskit Gradient Framework
  - PennyLane Optimizers
  - Cirq calculate gradients
  - ...
How do we assess our model? - Validation

Test Data:

\[ X_{\text{test}} \]

Feed in the test samples \((x_1, \ldots, x_m)\)
How do we assess our model? - Validation

Test Data: \( X_{test} \)

Quantum Model:

\[
f(X_{test}, \Theta^*) = \hat{y}
\]

Use parameter values \( \Theta^* \) obtained in the optimization

Compute predicted labels \( (\hat{y}_1, \ldots, \hat{y}_n) \)
How do we assess our model? - Validation

Test Data: $X_{test}$

Quantum Model: $f(X_{test}, \theta^*) = \hat{y}$

Cost: $C(f(X_{test}, \theta^*), y)$

Compute **cost from predicted label $\hat{y}$ and ground truth $y$**: How good is it?
What about unseen data?
Generalization Error

Variance

Simple Models

Complicated Models

Capacity
Variance

- refers to the changes in the model when using different portions of the training data set
- **Simply stated**: variability in the model prediction, i.e., how much the ML function can adjust depending on the given data set

![Diagram showing variance in training and test data sets](image)

- Model captures a very specific pattern observed only in the training data and misclassifies the test data
Generalization Error

Bias

Variance

Simple Models

Complicated Models

Adapted from A. Dekusar, J. Gacou, Qiskit Machine Learning: Quantum algorithms for supervised learning, EQTC, 2022
Bias

- Model does not capture enough patterns from data to produce correct results
Generalization Error

Model is not complex enough to match all the available data and performs poorly with the training dataset ⇒ **Underfitting:** model is unable to match the input data to the target data

![Diagram showing Underfitting and Overfitting]

Highly complex models matching almost all the given data points and perform well in training datasets, but: model cannot generalize the data point in the test data set to predict the outcome accurately ⇒ **Overfitting:** model tries to match non-existent data

Adapted from A. Dekusar, J. Gacoin, Qiskit Machine Learning: Quantum algorithms for supervised learning, EQTC, 2022
Barren Plateau hindering Optimization

- Many QML algorithms suffer from the dreaded "barren plateau" of unsolvability running into dead ends on optimization problems
- Trainability problem for flat problem-solving spaces
- Optimization algorithm can't find the downward slope in a "featureless" landscape with no clear path to the energy minimum
Barren Plateau hindering Optimization

- Barren plateau landscapes correspond to gradients that vanish exponentially in the number of qubits
  - An exponentially large precision is needed to navigate through the landscape
- A cost function $C(\Theta)$ has a barren plateau iff the variance $Var \in O\left(\frac{1}{b^n}\right)$ for some $b > 1$ with $n$ number of qubits
- In the absence of a barren plateau:
  - Determination of a minimizing direction in the cost function landscape does not require an exponentially large precision
    - One can always navigate through the landscape by measuring expectation values with a precision that grows at most polynomially with the system size
  - Speedup over classical algorithms often scaling exponentially for polynomial overhead of quantum algorithms
    - Quantum speedup often only in the absence of a barren plateau
- Although different approaches overcome the limitations arising from barren plateaus, there is not one simple way for it
Local Cost Functions for Diagonalization

**Goal** is to find \( \Theta_{\text{opt}} := \text{argmin}_\Theta(C(U_p(\Theta))) \), where \( C(U_p(\Theta)) \) quantifies how far the state \( \rho_p(\Theta) \) is from being diagonal.

**Cost functions:** \( C_1(U_p(\Theta)) = T r(\rho^2) - T r(\mathcal{Z}(\rho)^2) \) and \( C_2(U_p(\Theta)) = T r(\rho^2) - \frac{1}{n} \sum_{j=1}^{n} T r(\mathcal{Z}_j(\rho)^2) \) using below given tests.

\( \Rightarrow \) **C2 mitigates barren plateaus** for large \( n \) in comparison to \( C_1 \).

**Correctness:** Global minima of \( C_1 \) and \( C_2 \) coincide for \( U_p(\Theta) \) diagonalizing \( \rho \): \( C_1(U_p(\Theta)) = 0 \iff C_2(U_p(\Theta)) = 0 \iff \rho = \mathcal{Z}(\rho) \)

**Tests:** Destructive Swap

**Diagonalized Inner Product (DIP)**

**Partially Diagonalized Inner Product (PDIP)**

**Postprocessing:** scales \( O(n) \) no scales \( O(n) \)

**Notation:**

\( T r(\rho^2) \): Purity of an \( n \)-qubit state \( \rho \) ranging from:

- being completely mixed (i.e., the center of the Bloch sphere) \( \frac{1}{2n} \leq T r(\rho^2) \leq 1 \) pure state on the surface of the Bloch sphere \( \mathcal{Z}(\sigma) \) and \( \mathcal{Z}_j(\sigma) \) are quantum channels that dephase (i.e., destroy the off-diagonal elements of \( \sigma \)) in the global standard basis and in the local standard basis on a set \( j \) of qubits.
Barren Plateau hindering Optimization

- "If you have a barren plateau, all hope of quantum speedup or quantum advantage is lost"
- "People have been proposing quantum neural networks and benchmarking them by doing small-scale simulations of 10s (or fewer) qubits. The trouble is, you won't see the barren plateau with a small number of qubits, but when you try to scale up to more qubits, it appears. Then the algorithm has to be reworked for a larger quantum computer."
- "We were able to prove that, if you choose a cost function that looks locally at each individual qubit, then we guarantee that the scaling won't result in an impossibly steep curve of time versus system size, and therefore can be trained"
- Most quantum variational algorithms initiate their search randomly and evaluate the cost function globally across every qubit, which often leads to a barren plateau.
Barren Plateau hindering Optimization

- Barren plateaus may occur for global cost functions and for circuits with large depth

<table>
<thead>
<tr>
<th>Circuit Depth</th>
<th>$O(1)$</th>
<th>$O(\log(n))$</th>
<th>$O(\text{poly}(\log(n)))$</th>
<th>$O(\text{poly}(n))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Cost Function</td>
<td>maybe Barren Plateau</td>
<td>maybe Barren Plateau</td>
<td>maybe Barren Plateau</td>
<td>maybe Barren Plateau</td>
</tr>
<tr>
<td>Local Cost Function</td>
<td>Trainable</td>
<td>Trainable</td>
<td>Transition</td>
<td>maybe Barren Plateau</td>
</tr>
</tbody>
</table>

Institut für Informationssysteme | Prof. Dr. habil. S. Groppe
# Quantum Neural Networks Variants

<table>
<thead>
<tr>
<th>Classical Approach</th>
<th>Abbreviation</th>
<th>Quantum Counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boltzmann Machine</td>
<td>BM</td>
<td>QBM</td>
</tr>
<tr>
<td>Convolutional Neural Network</td>
<td>CVNN</td>
<td>QCVNN</td>
</tr>
<tr>
<td>Generative Adversarial Network</td>
<td>GAN</td>
<td>QGAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quantum Space Graph Convolutional Neural Network (QSGCNN)</td>
</tr>
<tr>
<td>Random Walk Neural Networks</td>
<td>RWNN</td>
<td>Quantum Walking Neural Network (QWNN)</td>
</tr>
<tr>
<td>Recurrent Neural Network</td>
<td>RNN</td>
<td>QRNN</td>
</tr>
<tr>
<td>Tensor Neural Networks</td>
<td>TNN</td>
<td>QTNN</td>
</tr>
<tr>
<td>Perceptron</td>
<td>P</td>
<td>Quantum Perceptron (QP)</td>
</tr>
<tr>
<td>Competitive Neural Networks</td>
<td>CPNN</td>
<td>QCPNN</td>
</tr>
<tr>
<td>Self-Organizing Neural Network</td>
<td>SONN</td>
<td>QSONN</td>
</tr>
<tr>
<td>Cellular Neural Network</td>
<td>CELL</td>
<td>QCELL</td>
</tr>
<tr>
<td>Weightless Neural Network</td>
<td>WLNN</td>
<td>QWLNN</td>
</tr>
<tr>
<td>Graph Neural Network</td>
<td>GNN</td>
<td>QGNN</td>
</tr>
</tbody>
</table>
Summary & Conclusions

- **Machine Learning**
  - Basics: Optimization
- **Gradient Descent Algorithm**
  - Calculation
    - Numerical differentiation
      - In quantum model via parameter shift rule
    - Symbolic differentiation
    - Automatic differentiation
      - Chain rule
- **Generalization Error**
  - Variance, Bias
  - Underfitting, Overfitting
  - Barren Plateau
- **Overview Quantum Neural Networks Variants**