Mathematical Perspective of Neural ODEs and Traditional Neurons

Catastrophic Forgetting and Continuous Learning

Pavan Mohan

December 23, 2024

The Neural Network Setup

- A **traditional neural network** can be defined as a collection of layers, each performing simple operations.
- The standard **feedforward neural network** function can be described as:

  $$y = f(x) = \sigma(W \cdot x + b)$$

  where:
  - x is the input vector,
  - W is the weight matrix,
  - b is the bias vector, and
  - σ is an activation function (e.g., ReLU, Sigmoid).
- **Training** involves adjusting the parameters W and b via backpropagation to minimize the loss function.

Traditional Neuron in a Layered Structure

- Let's consider a neural network with multiple layers:

$$\mathbf{h}^{[1]} = \sigma(W^{[1]}x + b^{[1]})$$

$$\mathbf{h}^{[2]} = \sigma(W^{[2]}\mathbf{h}^{[1]} + b^{[2]})$$

- This process continues layer by layer until we reach the output layer:

$$\mathbf{y} = \sigma(W^{[L]}\mathbf{h}^{[L-1]} + b^{[L]})$$

Mathematical View of Catastrophic Forgetting

- When a new task is learned, the weights $W^{[i]}$ and biases $b^{[i]}$ are updated based on the new task's data.
- However, if task 2 is learned after task 1, the parameters $W^{[i]}, b^{[i]}$ are adjusted, and task 1's knowledge may be overwritten.
- Mathematically, catastrophic forgetting occurs because:

$$\min_{\{W,b\}} \mathcal{L}(y, \hat{y}) = \min_{\{W,b\}} \mathcal{L}_1(y_1, \hat{y}_1) + \mathcal{L}_2(y_2, \hat{y}_2)$$

- As new tasks are introduced, the updates to the parameters $W^{[i]}$ only take into account the **current task**, ignoring previously learned tasks.

Neural ODE Setup

- A **Neural ODE** describes the network as a continuous dynamical system. The model is governed by an ordinary differential equation:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), \theta)$$

  where:
  - $\mathbf{h}(t)$ is the hidden state at time t,
  - $f(\mathbf{h}(t), \theta)$ is a neural network function parameterized by θ.
- This is integrated over time to evolve the state of the system:

$$\mathbf{h}(t) = \int_0^t f(\mathbf{h}(\tau), \theta) d\tau$$

- The final output y is computed from the state $\mathbf{h}(t)$:

$$y = g(\mathbf{h}(t), \theta)$$

Training Neural ODEs

- The training of Neural ODEs involves solving the ODE using methods like **backpropagation through ODE** solvers.
- The loss function is defined as:

$$\mathcal{L}(\theta) = \sum_t (y(t) - \hat{y}(t))^2$$

  where $y(t)$ is the predicted output, and $\hat{y}(t)$ is the true output.
- The parameters θ are updated using standard gradient descent methods, but the solution to the ODE must be computed during backpropagation.
- For each training step, the solution to the ODE is computed via an **ODE solver**.

Task Handling in Traditional Neurons

- For task 1, the neural network learns weights W_1 and b_1.
- When learning task 2, the weights are updated to W_2 and b_2.
- The update rule in the traditional network is discrete, so the learned task 1 knowledge can be overwritten:

$$\mathcal{L}_1 = \sum (y_1 - \hat{y}_1)^2, \quad \mathcal{L}_2 = \sum (y_2 - \hat{y}_2)^2$$

- This can lead to catastrophic forgetting, as the updates only take into account the new task.

Task Handling in Neural ODEs

- Neural ODEs don't use discrete layers, but instead continuously evolve the network's state.
- When learning task 2, the ODE's solution changes smoothly, and the model retains the history of the previous tasks.
- The learned knowledge is **integrated continuously** rather than being overwritten:

$$\mathbf{h}(t) = \int_0^t f(\mathbf{h}(\tau), \theta) d\tau$$

- The ODE's state is a continuous function of time, meaning previous knowledge persists and adapts with the introduction of new tasks.

Learning Task 1 and Task 2 in Neural ODEs

- Suppose Task 1 is learned with data D_1 and Task 2 with data D_2.
- In Neural ODEs, the state $\mathbf{h}(t)$ evolves continuously as the model processes both datasets:

$$\mathbf{h}_1(t) = \int_0^t f(\mathbf{h}_1(\tau), \theta_1) d\tau$$

$$\mathbf{h}_2(t) = \int_0^t f(\mathbf{h}_2(\tau), \theta_2) d\tau$$

- Task 2's data influences the hidden state $\mathbf{h}(t)$ without erasing task 1's knowledge.
- The output is computed as:

$$y_2 = g(\mathbf{h}(t), \theta)$$

- The state $\mathbf{h}(t)$ integrates knowledge from both tasks, thus preventing catastrophic forgetting.

Evaluation of Accuracy: Traditional vs. Neural ODE

- Let's consider a **Toy Dataset** with **Task 1** and **Task 2**.
- **Task 1** accuracy (before and after learning Task 2):

$$\text{Accuracy on Task 1 (before Task 2)} = 88.2\%$$

$$\text{Accuracy on Task 1 (after Task 2)} = 80.8\%$$

- **Task 2** accuracy:

$$\text{Accuracy on Task 2} = 60.9\%$$

- In the **traditional neural network**, learning Task 2 leads to a drop in Task 1 accuracy, showing **catastrophic forgetting**.
- For Neural ODEs, this **drop does not occur** as it continuously adjusts the learned task knowledge.

Conclusion: Why Neural ODEs Solve Catastrophic Forgetting

- Traditional neurons are based on **discrete layers**, which leads to **catastrophic forgetting** when learning new tasks.
- Neural ODE neurons, however, are based on **continuous evolution** of states, preventing overwriting of previously learned tasks.
- Neural ODEs offer a robust approach for **sequential learning** without forgetting previous knowledge.
- The key advantage of Neural ODEs is their ability to **integrate knowledge continuously**, ensuring that the network can handle **multiple tasks** simultaneously.