Personalized Conputational Casual Model based on Neural Networks

Yilin Li

In this note, we would illustrate the progression on PCCMNN populational model. More specifically, focus on the model structure, the loss function design, training procedure and results.

July 21, 2025

Contents

1. Model Structure

To prevent the output trajectory curve from diverging at large values of s, the model needs to produce a smaller output when the absolute value of the input y is large, and a larger output when y is close to 0. (This can be inferred from the graph, as the data points are standardized, y approaching 0 means the true value is close to the mean, where the rate of change is greater). For this reason, a gating network was designed. A layer with a Gaussian activation function is added to the output of the gating network to filter out inputs that are far from zero. Since the value range of the Gaussian function is $(0, 1]$, its output is multiplied by a magnitude network for scaling. This produces the dynamics term, which is the final output of the model.

Other architectures that were previously attempted include:

- A simple single-hidden-layer neural network with no activation function, which results in a quadratic polynomial. This model was unable to produce a convergent trajectory.
- A multi-head network that used SiLU in the earlier layers, but employed different activation functions in the final layer for the neurons outputting a, beta, tau, and n versus the neuron outputting c.
- Neural networks built with activation functions such as Sigmoid, tanh, and SiLU.

The model architecture is shown in the figure.

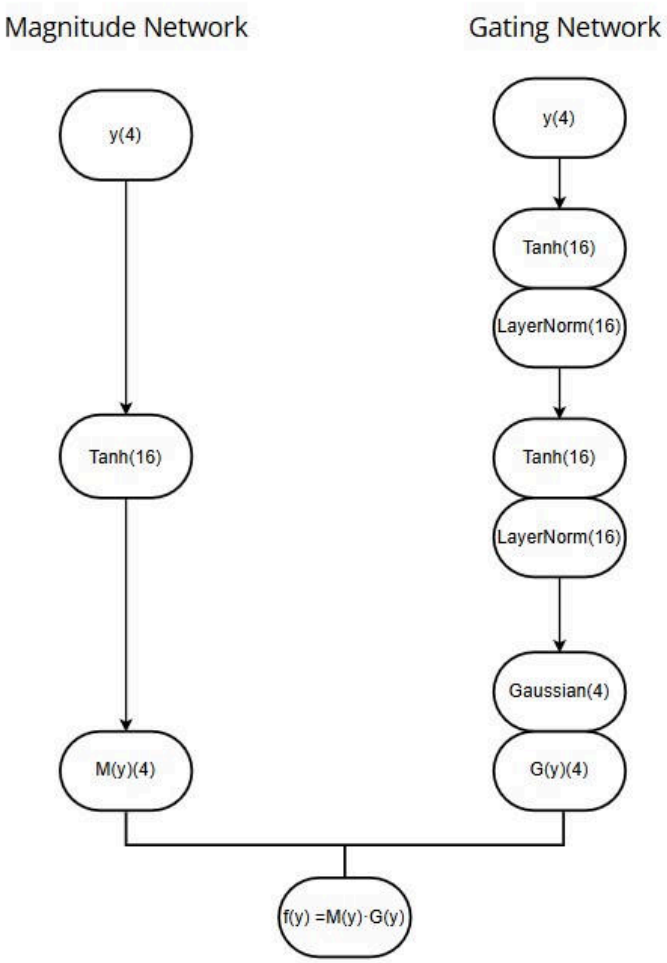

Figure 1: Model Architecture

The model employs the fourth-order Runge-Kutta (RK4) method for forward integration to solve the ordinary differential equation (ODE). Given an initial state y_0 and a grid of points s, the trajectory y is computed iteratively.

For each step from s_i to s_{i+1} with a step size of $h = s_{i+1} - s_i$, the subsequent state y_{i+1} is determined by the following equations:

$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \tag{1}$$

where:

$$\begin{aligned} k_1 &= f(y_i) \\ k_2 &= f\left(y_i + \frac{h}{2}k_1\right) \\ k_3 &= f\left(y_i + \frac{h}{2}k_2\right) \\ k_4 &= f(y_i + hk_3) \end{aligned} \tag{2}$$

Here, $f(y)$ represents the system's dynamics as modeled by the neural network.

2. Loss Function

The total loss for a patient's trajectory is calculated in five steps:

1. **Sequential Loss (L_s):** Measures one-step-ahead prediction accuracy.
   - For each step i, predict $\hat{y}_{i+1}$ using the true previous state y_i.
   - Calculate the Mean Squared Error (MSE) between the predicted sequence and the true sequence.

2. **Global Loss (L_g):** Measures overall trajectory accuracy.
   - Predict the entire trajectory $\hat{y}$ using only the initial true state y_0.
   - Calculate the MSE between the full predicted trajectory and the true trajectory.

3. **Smoothness Loss (L_{smooth}):** Penalizes trajectory oscillations.
   - The second derivative of the trajectory is approximated using the central difference formula:

$$y_i'' \approx \frac{\hat{y}_{i+1} - 2\hat{y}_i + \hat{y}_{i-1}}{h^2} \tag{3}$$

   - The loss penalizes the numerator of this expression (as the step size h is constant), and is calculated as:

$$L_{\text{smooth}} = \frac{1}{T-2} \sum_{i=1}^{T-2} \| \hat{y}_{i+1} - 2\hat{y}_i + \hat{y}_{i-1} \|^2 \tag{4}$$

4. **Optional Weighting:**
   - For L_s and L_g, the squared error for each biomarker can be weighted by its inverse residual variance ($\frac{1}{\sigma^2}$) before calculating the mean.

5. **Combined Loss (L):**
   - The final loss is a weighted sum of the components:

$$L = 0.5L_s + 0.5L_g + \lambda_{\text{smooth}} L_{\text{smooth}} \tag{5}$$

3. Training Procedure

The model is trained through an iterative process that alternates between updating the shared network weights and fine-tuning the patient-specific parameters.

3.1. Mini-batching and Sample Weighting

The training is performed using mini-batches with a size of 128. To prioritize higher-quality data, each patient's sample is weighted by its sequence length (the number of available data points), giving longer trajectories more influence during the training process.

3.2. Patient-Specific Parameter Initialization

The personalized time-scaling parameters, α and β, are initialized based on the patient's diagnostic stage to place their data within a meaningful range of the disease progression timeline, s.

- A target range for s is determined by the patient's clinical stage (CN, LMCI, or AD).
- α is initialized to a random value near 1.0.
- β is calculated to map the patient's mean age, t_{mean}, to a random point within the stage-specific s range, using the relation $s = \alpha t + \beta$. We set the range of s at "CN" to be [−10, 0], "LMCI" to be [0, 10], and "AD" to be [10, 20].
- To ensure α remains positive during optimization, it is reparameterized as $\alpha = \exp(\theta_0) + \varepsilon$, with the optimizer updating the unconstrained parameter θ_0. β is directly optimized as θ_1.

3.3. Optimization Loop

Each patient is assigned a dedicated optimizer for their parameters (θ_0, θ_1). The optimization follows a loop:

1. **Update Global Model:** The average combined loss (L) is computed across all patients in a batch. This average loss is used to update the shared weights of the neural network $f(y)$.
2. **Compute Residual Variance:** The variance of prediction residuals for each biomarker (σ^2) is calculated across the entire patient population. This captures the model's performance on each feature.
3. **Update Patient Parameters:** For each patient, the global and sequential losses are recalculated. This time, the error for each biomarker is weighted by its inverse residual variance ($\frac{1}{\sigma^2}$). This adjustment makes the model focus more on biomarkers with smaller errors. The resulting weighted loss is then used to update that specific patient's parameters, θ_0 and θ_1.
4. **Repeat:** The process is repeated, starting again from step 1.

4. Results and Evaluation

4.1. Data Filtering and Visualization

To provide a robust evaluation, the results are first filtered to exclude outliers. The 5th and 95th percentiles of the predicted disease progression timeline, s, are computed across all patients. Any patient whose s range falls entirely outside this interval is excluded from the visualization.

The main result is presented as a plot of the average disease trajectory:

- The horizontal axis represents the disease timeline, s, spanning from the 5th to the 95th percentile.
- An average initial state, y_0, is computed from all patients. The plotted trajectory starts from this point at the 5th percentile of s.
- A confidence interval is constructed by generating a predicted trajectory for every patient. The 25th and 75th percentiles of these trajectories are then plotted at each time point, forming a shaded region around the mean trajectory.

4.2. Challenges and Discussion

A significant challenge has been the numerical instability of the learned dynamics. In many cases, the predicted trajectories diverge to large values for large s. Initially, this was attributed to the model architecture; activation functions like `sigmoid` or `tanh` in the final layer tend to saturate near 1, leading to an almost linear increase in the trajectory. The current model attempts to address this by incorporating a Gaussian rectifier in the output of the gating network, which is designed to produce larger outputs for inputs near zero and suppress outputs for inputs far from zero.

Another unresolved issue is the model's difficulty in learning the characteristic S-shaped curves of biomarker progression. The learned trajectories often resemble a simple average line or a linear segment. An attempt to directly model the dynamics with a two-layer network (equivalent to a quadratic polynomial) also failed, as it resulted in exploding values at the tail ends of the s timeline.

The architectures of the four models shown in the figure are as follows:

- **Model 12632:** A simple sequential network consisting of two linear layers with SiLU activation, Layer-Norm, and a final Gaussian activation function.
- **Models 532, 7736, and 29508:** These models share a common architecture composed of a Gating Network and a Magnitude Network, which has been introduced at the begining of this report.

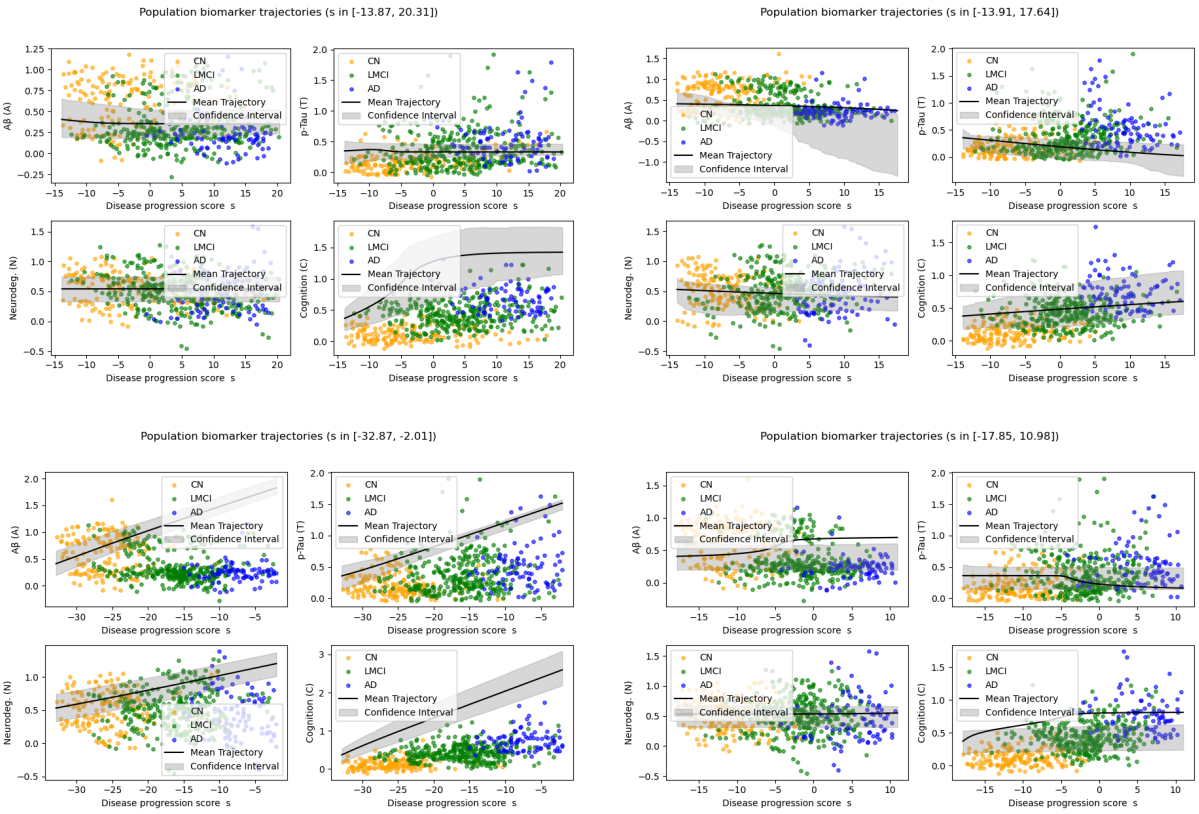

Figure 2: Examples of results from four different models (ID: 532, 7736, 12632, 29508). The solid lines represent the model's prediction, and the points are the actual data.

Index of Figures