# Intro to FitRec Mathamatical Formulation

Kaiwen Bian

December 2, 2024

Refer to Fitness Recommendation Algorithms.

In the original paper, FitRec combines static user embeddings (derived from attributes) and temporal embeddings (derived from historical workout data) to support three key applications:

1. **Quantitative Tasks**: Predict workout metrics (e.g., heart rate) across a session, either beforehand or in real time.
2. **Qualitative Tasks**: Identify important factors influencing performance and cluster users based on similar patterns.
3. **Recommendation Tasks**: Suggest alternate routes to meet target workout goals, such as achieving a specific heart rate profile. Based on the heart rate needed (target), give workout information (attributes).

Why do we use this model:

1. **Context-aware**: Workouts don't occur in isolation. A user's performance today is influenced by their previous workouts (e.g., fatigue, recovery, adaptation). The embeddings learn from historical sequences and build personalized, context-aware, and temporal data understandable machine.
2. **Temporal relationships**: Model temporal relationship in the target, understand that the data is temporally connected.

## Temporal Embedding Construction

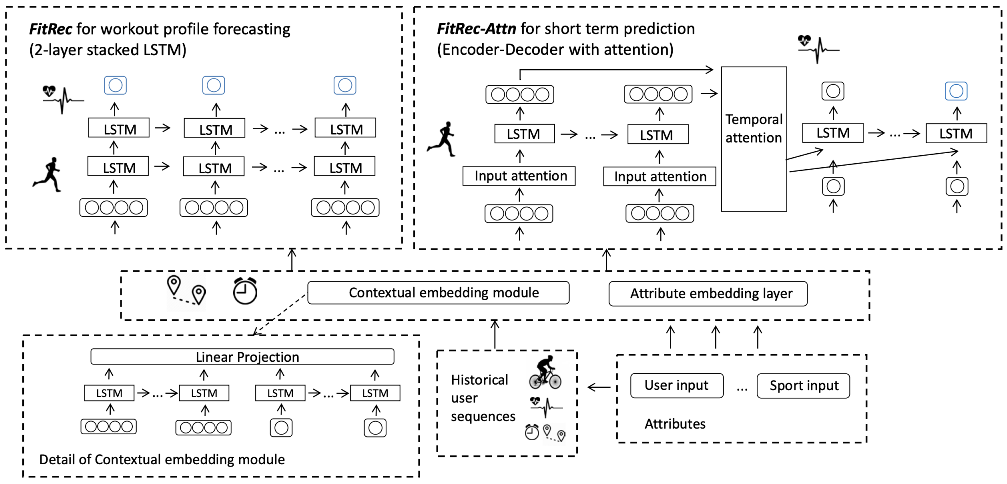
### Data Representation and Notation

The TBR model takes sequences and embeddings as input, essentially building representations across times:

$X$ is combination of various contextual sequences for the current workout, each $x_i$ is a vector of contextual information (altitude, type of workout, ...).

$$X = (x_1, \ldots, x_T) \in \mathbb{R}^{N \times T}$$

- Altitude ($x_t^1$): Elevation at time $t$ (e.g., 200 meters above sea level).
- Speed ($x_t^2$): Speed at time $t$ (e.g., 12 km/h).



FitRec for workout profile forecasting
(2-layer stacked LSTM)
LSTM
FitRec-Attn for short term prediction
(Encoder-Decoder with attention)
Input attention
Temporal attention
Contextual embedding module
Attribute embedding layer
Linear Projection
Detail of Contextual embedding module
Historical user sequences
User input
Sport input
Attributes

Figure 1: Structure Overview

- Distance ($x_t^3$): Cumulative distance covered up to time $t$ (e.g., 3.5 km).
- Type of Workout ($x_t^4$): Encoded categorical information about the workout (e.g., running = 0, biking = 1).
- For $T = 10$ and $N = 4$, $X$ might look like:

$$X = \begin{bmatrix} 200 & 201 & \dots & 210 \\ 12 & 11.8 & \dots & 10.5 \\ 0.1 & 0.2 & \dots & 1.0 \\ 1 & 1 & \dots & 1 \end{bmatrix}.$$

$y$ is the target sequence (heart rate, speed).

$$y = (y_1, \dots, y_T) \in \mathbb{R}^T$$

- Heart Rate ($y_t$): The user's heart rate at time $t$ (e.g., 140 bpm, 145 bpm).
- $y$ might be: $y = [140, 142, 145, 148, \dots, 155]$ (bpm).

$Z$ is the historical contextual sequences (recent workout contextual sequences).

$$Z = (z_1, \dots, z_T) \in \mathbb{R}^{N \times T}$$

- This $Z$ comes from the most recent workout, sort of a Markov property idea? Can we find a equivalent form in our data set?
- All $N$ dimension category need to be matching with $X$'s $N$ dimension.

$y'$ is the historical target sequence (recent workout heart rate or speed).

$$y' = (y'_1, \dots, y'_T) \in \mathbb{R}^T$$

- Same as in the decription for $Z$.

Each workout has associated metadata or static attributes:

$$a = (a_1, \dots, a_m),$$

- $|a_i|$ is the number of distinct values of attribute $a_i$.
- User ID ($a_1$): Encoded identifier for the user (e.g., User 1, User 2).
- Sport Type ($a_2$): Encoded categorical type of workout (e.g., 0 = running, 1 = biking).
- Gender ($a_3$): Encoded value (e.g., 0 = male, 1 = female).
- $a$ for a workout could be:

$$a = (\text{User ID: } 42, \text{Sport Type: } 1, \text{Gender: } 0).$$

## Static Meta-data Embedding

Attributes are embedded into a latent space (embeds user specific information such as fitness level and gender, into latent features that summarize the user's characteristics.):

$$e_{a_i} = E_{a_i}(a_i), \quad \forall i \in [1, m],$$

where $e_{a_i} \in \mathbb{R}^{D_1}$ is the embedding of the $i$-th attribute, and $E_{a_i} \in \mathbb{R}^{|a_i| \times D_1}$ is the learned embedding matrix.

## Contextual Sequential Consistency Embedding

Encodes the historical workout data to capture trends and patterns, like how a user's heart rate responds to certain activities. Two LSTMs encode historical sequences for 1 historical contextual sequence $z_t$ and 1 historical target sequence $y'_t$ into hidden states (as $t$ increases, we would use previous historical state to process, so the hiddeb state we use for embedding later is still in a sequential consistency manner):

$$h_{1,t} = \text{LSTM}_1(z_t, h_{1,t-1}), \quad h_{2,t} = \text{LSTM}_2(y'_t, h_{2,t-1}),$$

where $h_{1,t}$ and $h_{2,t}$ are the hidden states for historical contextual and target sequences, respectively. The embeddings are combined via a linear projection:

$$e_t = W_e[h_{1,t}; h_{2,t}] + b_e,$$

where $W_e$ and $b_e$ are learned parameters.

## Input Construction

For each time step $t$, inputs are constructed from using current contextual information, embeddings of attributes and embeddings of historical contextual information:

$$u_t = [x_t; e_t; e_{a_1}; \ldots; e_{a_m}],$$

where $u_t \in \mathbb{R}^K$, with $K$ as the concatenated dimension.

## Trajectory Modeling

### Trajectory Prediction (FitRec)

Given the contextual sequences $X$, attributes $a$ for a candidate workout, historical sequences $Z$ and $y'$, as well as the total time of the workout, predict the entire target sequence $y = (y_1, \ldots, y_T)$.

This can be sued for commanding types of workout (attributes) based on the target heart rate that the user specify (prediction matches).

Practically, a 2-layer LSTM predicts the entire target sequence (this would captures how the relationship of $y_t$ and $y_{t-1}$ varies overtime, establishing connections between them):

$$h_t = \text{LSTM}(u_t, h_{t-1}),$$

$$\hat{y}_t = \text{SELU}(W_{NAT} h_t + b_{NAT}),$$

where $W_{NAT}$ and $b_{NAT}$ are learned parameters, and SELU is the Scaled Exponential Linear Unit activation function.

### Training Objective

The model minimizes the mean squared error (MSE) between predicted ($\hat{y}_t$) and actual ($y_t$) values:

$$L = \frac{1}{|T_{train}|} \sum_{y \in T_{train}} \sum_{t=1}^{L} (\hat{y}_t - y_t)^2,$$

where $|T_{train}|$ is the total number of sequences in the training set.

### Evaluation Metrics

Performance is assessed using:

- **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\frac{1}{|T_{test}|} \sum_{y \in T_{test}} \sum_{t=1}^{L} (\hat{y}_t - y_t)^2}.$$

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{|T_{test}|} \sum_{y \in T_{test}} \sum_{t=1}^{L} |\hat{y}_t - y_t|.$$

## LSTM Attention Prediction

FitRec-Attn introduces attention mechanisms to focus on critical features and time steps, making it more effective for real-time, short-term predictions.

## Encoder

The encoder processes the contextual sequences $u = (u_1, \ldots, u_T)$ using input attention and an LSTM.

**Input Attention:** The input attention mechanism determines the importance of each input feature:

1. Compute attention scores for each input feature:

$$s_t^k = v_s^\top (\mathbf{W}_s[h_{t-1}; u^k] + b_s), \quad 1 \leq k \leq K,$$

   where:

   - $h_{t-1}$: Previous encoder hidden state.
   - $u^k$: The $k$-th input dimension.
   - $\mathbf{W}_s, b_s, v_s$: Learnable parameters.

2. Normalize the scores using softmax:

$$\alpha_t^k = \frac{\exp(s_t^k)}{\sum_{j=1}^K \exp(s_t^j)}.$$

3. Weight the input features to create a new input:

$$\tilde{u}_t = (\alpha_t^1 u_t^1, \ldots, \alpha_t^K u_t^K).$$

**LSTM Encoder:** The encoder LSTM processes the attention-weighted input:

$$h_t = \text{LSTM}_e(h_{t-1}, \tilde{u}_t),$$

where $h_t$ encodes the contextual information across time steps.

## Decoder

The decoder generates predictions one step at a time using temporal attention and an LSTM.

**Temporal Attention:** Temporal attention determines how much each encoder hidden state contributes to the decoder's prediction:

1. Compute attention scores for all encoder hidden states:

$$\ell_{i,t} = v_\ell^\top (\mathbf{W}_\ell[d_{t-1}; h_i] + b_\ell), \quad 1 \leq i \leq T,$$

   where:

   - $d_{t-1}$: Previous decoder hidden state.
   - $h_i$: Encoder hidden state at time $i$.
   - $\mathbf{W}_\ell, b_\ell, v_\ell$: Learnable parameters.

2. Normalize the scores using softmax:

$$\beta_{i,t} = \frac{\exp(\ell_{i,t})}{\sum_{j=1}^{T} \exp(\ell_{j,t})}.$$

3. Compute the context vector as a weighted sum of encoder hidden states:

$$c_t = \sum_{i=1}^{T} \beta_{i,t} h_i.$$

**LSTM Decoder:** The decoder LSTM updates its hidden state based on the context vector and previous target value:

$$d_t = \text{LSTM}_d(d_{t-1}, [y_{t-1}; c_t]).$$

**Final Prediction:** The decoder produces the target prediction $\hat{y}_t$ through a linear transformation:

$$\hat{y}_t = W_{AT}[d_t; c_t] + b_{AT}.$$

## Objective Function

The model minimizes the Mean Squared Error (MSE) between the predicted and actual target values:

$$L = \frac{1}{|T_{\text{train}}|} \sum_{y \in T_{\text{train}}} \sum_{t=1}^{T} (\hat{y}_t - y_t)^2.$$