Behavior Modeling of mDiabetes Human Participants

Jack Wolf

May 3, 2022

1 mDiabetes Project

The mDiabetes project is a joint effort between Arogya World and UO to use AI to personalize healthcare-related interventions to at-risk individuals. The goal of the study is to improve participants attitude and behavior towards diabetes. Each week, participants in the study receive two positive-behavior promoting texts from 5 categories (health, fitness, diet, attitude, history), followed by two related questions to guage how the messages impacted their behavior. We have deployed a Deep Q Network to learn a mapping from participant knowledge state to relevant messages in the message bank. The next phase of this project is to model the behavior of participants with the goal of being able to rerun the study on a simulated population.

2 Behavior Modeling

Behavior modeling is the act of learning an agents behavior through observation, and being able to predict aspects of that agents future interactions in the same environment. A classic example of behavior modeling is knowledge tracing (KT). The agent in this case is a student, and the environment is the test/assignment they are answering questions from. In traditional KT, a model observes a students history of questions and answers, trying to learn the probability the student will answer the question correctly, in order to predict their future success. This can be used for many educational applications such as tailoring courses to specific needs and finding relationships between exercises.

3 Time Series Models

Human behavior is complicated and requires an expressive and dynamic model which can learn complicated temporal relationships between features.
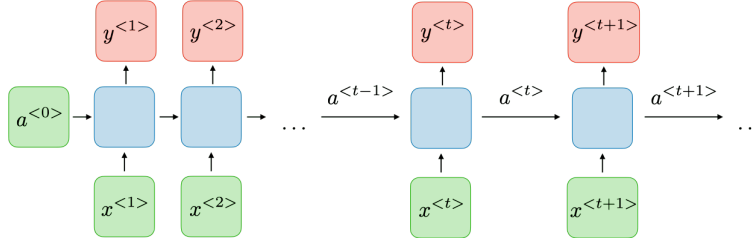
3.1 Bayesian Knowledge Tracing

In this method, student latent knowledge is represented by a binary vector K where $k_i = 1$ if the student has mastered subject k_i, and $k_i = 0$ if the student has not. When the model observes a student answer question Q in topic t correctly or incorrectly, a Markov Model updates the students probability that they have mastered topic k_t. One issue with BKT is that the binary representation of knowledge is unrealistic, along with the assumption that knowledge is never forgotten.

3.2 POMPDP

Similar to BKT, POMDPs present a promising but unrealistic model of behavior. These models require exponential exploration, and rely on a discrete state space which is infeasible.

3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are popular in time-series modeling because of their ability to model a dynamic, continuous, high-dimensional latent knowledge state. RNNs are able to learn time-dependent sequential patterns in training data, and can "remember" and re-use information from a previous input in the current prediction. RNNs are able to process sequences of information as opposed to a single point. The input to an RNN has two parts: x_t, the representation of the observation to predict the outcome of at time t, and a_{t-1}, the representation of the latent space at time $t-1$. RNNs have two outputs: y_t, the prediction for the input x_t, and a_t, the updated representation of the latent space after observing x_t. The intermediate/hidden output a_t is recursively fed back into the network at timestep $t+1$ so the model can recover and use information from the past.
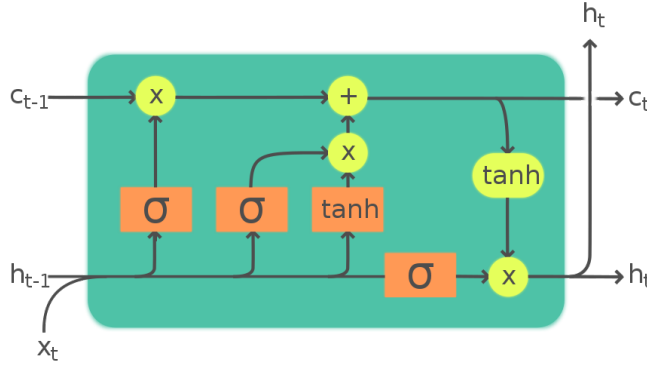

3.4 Long Short-Term Memory Neural Networks

Long Short-Term Memory Neural Networks (LSTMs) are a popular variant of RNNs because of their ability to represent memory in a more natural way. Similar to general RNNs, LSTMs are composed of weight-sharing cells wrapped together, where the output of cell t is fed into cell $t+1$. The four components of an LSTM cell are

- Forget Gate, f_t: The forget gate process the input x_t and a_t using a sigmoid function and determines which parts of the input are valuable/need attention. The output of

this gate is a vector with values between 0 and 1, where values closer to 1 represent data that should not be important, and values closer to 0 represent important data.

- Input Gate, i_t: Similarly to the forget gate, the input gate also determines the importance of x_t and a_t. Each input is passed through a sigmoid function and then a tanh function with values ranging from -1 to 1.

- Cell State, c_t: The cell state component uses data from the forget and input gates along with c_{t-1} to update the cell state to c_t. The model calculates an intermediate value $c'_t = c_{t-1} * f_t$, "forgetting" elements with a value of 0. Next, c_t is calulcated by performing point-wise addition of c'_t and i_t.

- Output Gate, o_t: The output gate performs point-wise multiplication on $\{x_t, a_t\}$ and c_t to determine which information should be carried/remembered in this time steps hidden state.

4 Using Neural Networks

4.1 Training

To accurately train AI to model this behavior, we need

- Training Features: a matrix X representing the questions the student has seen. Each $x_i \in X$ is a encoding (1-hot, k-sparse, ...) or feature vector of a single question the student has answered

- Training True Labels: a vector Y representing if the student answered a given question correctly. Each $y_i \in Y$ is a boolean $\in \{0, 1\}$

- A Model: a neural network M with set of weights w which can learn $x_i \rightarrow y_i$, typically by outputting the predicted probabilities for each possible outcome (correct or incorrect) and choosing the argmax as the prediction

- Loss function: Penalizes our models prediction of the students outcome on the input question. Traditional binary use-case uses binary cross entropy (BCE)

- Optimization function: Stochastic Gradient Descent on minibatches of observations

Each training loop, the model will sample a selection of previous observations $\{x_i, y_i\}$ and perform gradient descent until it has minimized its error. After training, the model weights (or a set of them) represent the latent knowledge state of the student.

4.2 Inference

Using this trained model to predict future perfomance is trivial. We can again represent the students unseen, future interactions as a vector X where each $x_i \in X$ is a representation of a single question. Note that this is the same setup as in training, but we do not have access to true Y. After training our model on minibatches of observations $\{x_i, y_i\}$, we can use it to predict the outcome probabilties for new/unseen questions.

5 RNNs for mDiabetes

In this section I will discuss how we can use RNNs (or variants of) for behavior modeling of our mDiabetes participants.

5.1 Encoding messages and questions

Because of the smaller size of our message and question bank (each is around 50), we can represent each message M and question Q as a binary encoding of their unique ID. For example, message 7 will be represented as 0111 and question 15 will be 1111

5.2 Dataset

Traditional KT datasets for a single participant consist of a feature matrix X, where each $x_i \in X$ is the feature vector or encoding representing the ith single exercise or question that the student has answered, and a vector Y where each $y_i \in Y$ is the binary label representing if the question was answered correctly (1) or incorrectly (0).
Our data is slightly different because of the mDiabetes experimental setup. The differences are summarized below:

- Questions/exercises are prompted by messages, so data regarding the associated message must be included in the observation vector x_i.

- Our responses y_i are not binary, but multiclass in $\{missing, low, medium, high\}$.

- Our participants answer two questions/exercises at each time step, so the labels vector Y needs to become a labels matrix where $y_{i,1}$ is the observed answer for the first question asked that timestep, and $y_{i,2}$ is the answer for the second.

With this, we can define the features of our dataset as follows:

- $M1$: the binary encoding of the ID of the first message sent to this participant

- $Q1$: the binary encoding of the ID of the first question sent to this participant

- $M2$: the binary encoding of the ID of the second message sent to this participant

- $Q2$: the binary encoding of the ID of the second question sent to this participant

And the labels of our dataset:

- $Q1\_response$: the value of the users response to $Q1$ this timestep

- $Q2\_response$: the value of the users response to $Q2$ this timestep

5.3   Model Architecture

Previous works such as DKT+ use an LSTM cell with a hidden shape of $(200,)$, but they also had a smaller . The input to our model will be a concatenation of $\{M1, Q1, M2, Q2\}$. The output of our model will be a $2x4$ vector P, where $P_{1,j}$ represents the probability that question 1 has the answer j, and similarly for $P_{2,j}$. We can recover the predicted answer by taking the argmax of the probabilities for each question.

5.4   Optimization

Traditional KT uses binary cross entropy loss to determine the prediction error, but our questions have multiclass answers so we need something different. For now we will assume the use of mean squared error loss, MSE. For the optimization algorithm we will use SGD.

5.5   Training

At each time step observed, t, retrieve the students oberservation feature vector and outcome vector

$$x_t = \{M1_t, Q1_t, M2_t, Q2_t\}$$
$$y_t = \{Q1\_response_t, Q2\_response_t\}$$

Using our model M with weights w, we can calculate the predicted response probabilities for this student interaction P_t and recover the predicted responses.

$$P_t = M(x_t; w)$$
$$\hat{Q1}_t = argmax(P_{t,1})$$
$$\hat{Q2}_t = argmax(P_{t,2})$$

We can use these predicted responses and the true responses to calculate our loss and update the model weights w

$$error = (\hat{Q1}_t - Q1_t)^2 + (\hat{Q2}_t - Q2_t)^2$$