LSTM Model

Pseudocode

Pre-Requisites:

Tqdm for Progress Bar

Numpy for Numerical Computing.

Function Definition:

Main Execution:

Data Preparation

1. Define the training data (text).
2. Convert data to lowercase.
3. Create a set of unique characters from the data.
4. Calculate data size and character size.
5. Create dictionaries to map characters to indices and vice versa.
6. Split the data into training inputs (train_X) and labels (train_y).

Helper Functions

1. oneHotEncode(text):
    a. Takes a character as input.
    b. Returns a one-hot encoded vector of size char_size.
2. initWeights(input_size, output_size):
    a. Takes input and output sizes.
    b. Initializes weights using Xavier normalization.
    c. Returns the initialized weight matrix.

Activation Functions

1. sigmoid(input, derivative=False):
    a. Takes input and an optional derivative flag.
    b. Returns the sigmoid of the input.
    c. If derivative is True, returns the sigmoid derivative.
2. tanh(input, derivative=False):
    a. Takes input and an optional derivative flag.
    b. Returns the hyperbolic tangent of the input.
    c. If derivative is True, returns the tanh derivative.

3. softmax(input):
    a. Takes an input vector.
    b. Returns the softmax of the input vector.

Long Short-Term Memory Network Class (LSTM)

Initialization (__init__)

1. Takes input size, hidden size, output size, number of epochs, and learning rate as hyperparameters.
2. Initializes weights and biases for the Forget, Input, Candidate, Output, and Final Gates using initWeights and zero arrays.

Reset Network Memory (reset)

1. Clears or initializes dictionaries to store intermediate values during forward propagation (concatenated inputs, hidden states, cell states, gate outputs, etc.). Initializes hidden and cell states at time step -1 to zeros.

Forward Propagation (forward)

1. Takes a list of inputs.
2. Resets network memory.
3. Iterates through each input:
    a. Concatenate the current input with the previous hidden state.
    b. Calculate Forget Gate output using sigmoid activation.
    c. Calculate Input Gate output using sigmoid activation.
    d. Calculate Candidate Gate output using tanh activation.
    e. Calculate Output Gate output using sigmoid activation.
    f. Update the Cell State using the Forget and Input Gate outputs and the previous Cell State.
    g. Update the Hidden State using the Output Gate output and the tanh of the current Cell State.
    h. Calculate the final output for the current time step.
4. Returns a list of outputs for all time steps.

Backward Propagation (backward)

1. Takes a list of errors and the inputs used in forward propagation.
2. Initializes gradients for all weights and biases to zero.
3. Initializes dh_next and dc_next (gradients for hidden and cell states at the next time step) to zeros.
4. Iterates through time steps in reverse:
    a. Calculate the error for the current time step.

    b. Calculate gradients for the Final Gate weights and biases.
    c. Calculate the gradient for the Hidden State.
    d. Calculate gradients for the Output Gate weights and biases.
    e. Calculate the gradient for the Cell State.
    f. Calculate gradients for the Forget Gate weights and biases.
    g. Calculate gradients for the Input Gate weights and biases.
    h. Calculate gradients for the Candidate Gate weights and biases.
    i. Calculate the gradient for the concatenated input.
    j. Update dh_next and dc_next.
5. Clip gradients to prevent exploding gradients.
6. Update weights and biases using the calculated gradients and the learning rate.

Train (train)

1. Takes inputs and labels.
2. One-hot encode the inputs.
3. Iterate for the specified number of epochs:
    a. Perform forward propagation to get predictions.
    b. Calculate errors (difference between predictions and labels).
    c. Perform backward propagation to update weights and biases.

Test (test)

1. Takes inputs and labels.
2. Initializes accuracy to zero.
3. Perform forward propagation to get probabilities.
4. Generate predictions by sampling from the softmax probabilities.
5. Compare predictions with ground truth labels and calculate accuracy.
6. Print the ground truth, predictions, and accuracy.

Network Initialization and Training

1. Define hidden_size.
2. Initialize the LSTM network with specified parameters.
3. Train the network using train_X and train_y.
4. Test the network using train_X and train_y.