Computer Science 4143: Deep Learning
University of Arkansas – Fort Smith
Spring 2022

Problem Set 4: Recurrent Neural Networks

50 points

Version 1.0

due Tuesday, 29 March 2022, by 11:00 PM CT

Review the following assignment in its entirety prior to beginning. All submissions will be managed from within the course website.

Experiment Details

Review the data that has been provided to you on the course website. The file consists of $n = 10000$ records with $\tau = 51$ time steps. The data has already been randomized. Use the first $7,000$ records for your training data, and the last $3,000$ records for your testing data. In your testing data, use the first 5 records as the *output set* that you will use for predictions as requested below (mainly to show that your model works). Be sure to use `epochs = 15` for each network configuration along with the `adam` optimizer. For each model you produce, be sure to output the evaluation metrics from the test data.

1. Load each of the sequences from the file. The data will be loaded as a 2D array. You will need to reshape this data to become a 3D array in the form

   $$(\text{batch size} \times \text{time steps} \times \text{input dimensions})$$

   such that each value is in its own dimension (input dimensions = 1). For the entire dataset, your goal is to reshape the dimensions $(10000, 51)$ into the form $(10000, 51, 1)$.

2. This is a time-series regression task, which means that our goal is to predict $\hat{y}$. We will use a dense layer with a single neuron to predict a value. Which loss function would be appropriate for this model?

3. **Many-to-One Model**: Use $\tau = 51$ to represent your value for y in each record. Generate a model that is capable of predicting $\tau = 51$ for each of the sequences.

Create model that has two recurrent neural network layers with 15 neurons. Use the `SimpleRNN` layer for this model.

Hint: you will need to pass the sequences from the first recurrent layer to the second recurrent layer. You can use the `return_sequences` parameter.

4. Show the predicted value by printing each value for every record in the output set (be sure that you label it as `"Record 1 Prediction: [123]"`, etc.). In addition, concatenate each record in your output set with the predictions and visually plot each of the records.

5. **Many-to-One Model with Multiple Predictions**: Use the `predict()` method to predict the values for the next $\tau = 5$ time steps for each input sequence. Predict the values for $\tau = 51$ to $\tau = 55$ (noting that a single vector $\mathbf{x}^{(j)}$ has the dimensions of $(1 \times 50 \times 1)$).

   Hint: predict a value and concatenate it with the previous sequence for each of the $\tau = [51, 56]$ time steps. For example, to predict the first entry, use x[0:49]; to predict the second value, use (x[1:49] + `predict`(x[0:49]), etc. where the symbol "+" denotes concatenation.

6. Show the predicted value by printing each value for every record in the output set (be sure that you label it as `"Record 1 Prediction: [123, 456, 789]"`, etc.). In addition, concatenate each record in your output set with the predictions and visually plot each of the records.

7. **Many-to-Many (seq-to-vec) Model**: Let's improve the previous approach that we implemented. Previously, we used a sequence to predict a value, which is a many-to-one architecture. In this next approach, we are going to create a many-to-many architecture, or a sequence-to-vector architecture, to predict the next five time steps simulatenously.

   For this approach, you will need to reshape your training and testing data: $\mathbf{y}$ will be comprised of the last 5 values as opposed to the last value only, and $\mathbf{x}$ will represent the first 46 values. Reshape the vectors as necessary. The only modification that you will need to perform to your previous model is to change the output layer to 5 neurons instead of 1.

8. Show the predicted value by printing each value for every record in the output set (be sure that you label it as `"Record 1 Prediction: [123, 456, 789]"`, etc.). In addition, concatenate each record in your output set with the predictions and visually plot each of the records.

9. **Many-to-Many (seq-to-seq) Model**: Let's once again improve upon the previous approach that we implemented. Previously, we used a sequence to predict a vector of values, which is a many-to-many architecture. In this next approach, we are going to create another many-to-many architecture, or a sequence-to-vector architecture, to predict the next five time steps for each input. For example, if we pass $x^{(0)}_{\tau=0}$ as input (record 0 at time step 0), we will predict/output the values as

$$\hat{\mathbf{y}}^{(0)}_{\tau=0} = \begin{bmatrix} y^{(0)}_{\tau=1} & y^{(0)}_{\tau=2} & y^{(0)}_{\tau=3} & y^{(0)}_{\tau=4} & x^{(0)}_{\tau=5} \end{bmatrix}$$

Similarly, if we pass $x^{(0)}_{\tau=1}$ as input (record 0 at time step 1), we will predict/output the values as

$$\hat{\mathbf{y}}^{(0)}_{\tau=1} = \begin{bmatrix} y^{(0)}_{\tau=2} & y^{(0)}_{\tau=3} & y^{(0)}_{\tau=4} & y^{(0)}_{\tau=5} & x^{(0)}_{\tau=6} \end{bmatrix}$$

For this approach, you will need to reshape your training and testing data: $\mathbf{y}$ will need to have the shape (batch size $\times$ time steps $\times$ 5). For every input vector $\mathbf{x}$ and for every time step τ, you will need to set the output vector $\hat{\mathbf{y}}_\tau$ as follows:

$$\hat{\mathbf{y}}^{i}_{\tau} = \begin{bmatrix} x^{(i)}_{\tau+1} & x^{(i)}_{\tau+2} & x^{(i)}_{\tau+3} & x^{(i)}_{\tau+4} & x^{(i)}_{\tau+5} \end{bmatrix}$$

For simplicity, you can accomplish this with the following code:

```
# You will need to adjust X and Y.  Consider the following:
# X[0 - 46]
# Y1 = X[1-5],  Y2 = X[2-6],  Y3 = X[3-7] .... Ylast = X[47-51]

step_count = 51 - 5    # which implies X[0:46]
Y_outputs = 5
Y_seq = np.empty( (10000, step_count, Y_outputs) )
for next in range(1, Youtputs - 1):
   # for all rows and for all time steps, set Y output vector
   Y_seq[ :, :, next-1 ] = alldatafromfile[ :, next:next+step_count, 0]
```

This implies that $\mathbf{x}$ will represent the first 46 values, and for each value, you will output sequences of length 5 that correspond to the predictions for the next 5 values of each time step. Reshape the vectors as necessary. The only modification that you will need to perform to your previous model is to change return the sequences from the second `SimpleRNN` layer to the `Dense` layer. The `Dense` layer will reshape the sequences with the 3D shape (batch size $\times$ time steps $\times$ input dimensions) to a 2D shape of ([batch size $\times$ time steps] $\times$ input dimensions) such that each time step is treated as a separate instance.

10. Show the predicted values for the last input of each sequence by printing them for every record in the output set (be sure that you label it as `"Record 1 Prediction: [123, 456, 789]"`, etc.). In addition, concatenate each record in your output set with the predictions and visually plot each of the records.

11. Recreate the previous experiments and change the `SimpleRNN` layers to `LSTM` layers. What differences did you observe? Compare and contrast the differences.

12. Recreate the previous experiments and change the `LSTM` layers to `GRU` layers. What differences did you observe? Compare and contrast the differences.

13. Finally, construct a report that includes the following table with the appropriate metrics calculated. Your report should include two paragraphs for the following sections:

    - **Experimental Settings**
      a) Data – describe the dataset, any preprocessing or constructions, etc.
      b) Models – describe the model architecture, hyperparameter settings, etc.
    - **Evaluation**
      a) Describe the settings used to conduct the experiments (single pass, k-fold cross validation, multiple successive runs with a mean MSE score reported, etc.). This should be convincing that your results are valid.
      b) Include the table of your results
      c) Discuss which model performed best and explain why

	Seq-to-Output	**Seq-to-Vec**	**Seq-to-Seq**
SimpleRNN	mse	mse	mse
LSTM	mse	mse	mse
GRU	mse	mse	mse

Deliverables

You will be responsible for delivering the following items:

1. Jupyter Notebook File (exported as HTML)
2. Report