Assignment 5: Sequence Models

CS489/698 – Winter 2018

Out: March 19, 2018
Due: March 30 (11:59pm), 2018

Submit an electronic copy of your assignment via LEARN. Late submissions incur a 2% penalty for every rounded up hour past the deadline. For example, an assignment submitted 5 hours and 15 min late will receive a penalty of ceiling(5.25) * 2% = 12%.

Be sure to include your name and student number with your assignment.

1. **[40 pts]** Consider a Hidden Markov Model parametrized by $\Pr(y_t|y_{t-1})$ and $\Pr(x_t|y_t)$. It satisfies the Markov property since the current state y_t depends only on the previous state y_{t-1}. In practice, the current state may depend on earlier states and therefore the Markov property is not satisfied. It turns out that it is possible to ensure that the Markov property holds by augmenting the set of states.

   (a) **[20 pts]** Show how to rewrite a process parametrized by $\Pr(y_t|y_{t-1}, y_{t-2})$ and $\Pr(x_t|y_t)$ into an HMM parametrized by $\Pr(y'_t|y'_{t-1})$ and $\Pr(x_t|y'_t)$.

   (b) **[20 pts]** Is it possible to do this reparametrization without increasing the number of parameters? Justify your answer by counting the number of parameters before and after the transformation.

2. **[60 pts]** Recurrent neural network implementation

   In this question, you will experiment with various types of recurrent neural networks (RNNs) in PyTorch. PyTorch is a popular package for dynamic neural networks that can easily handle sequence data of varying length. For GPU acceleration, it is recommended that you perform your experiments in Google's Colaborary environment. This is a free cloud service where you can run Python code (including PyTorch) with GPU acceleration. A virtual machine with two CPUs and one Nvidia K80 GPU will run up to 12 hours after which it must be restarted. The following steps are recommended:

   - Create a Python notebook in Google Colab: `https://colab.research.google.com`
   - Click on `edit`, then `notebook settings` and select `None` (CPU) or `GPU` for hardware acceleration.
   - Install PyTorch by following the instructions on the following page: `https://colab.research.google.com/notebooks/snippets/importing_libraries.ipynb` Note that you will have to reinstall PyTorch each time that you obtain a virtual machine in Colab. Hence it is recommended to store the code for the installation of PyTorch in a cell that can be executed easily each time you obtain a virtual machine.
   - Get familiar with PyTorch (`http://pytorch.org/`) by going through the tutorial "Get familiar with PyTorch: a 60 minute blitz": `http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html`

   Answer the following questions:

   (a) **[20 pts]** Encoder implementation

- Go through the tutorial "Classifying Names with a Character-Level RNN"
  `http://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html`
- Download the data associated with the tutorial. In your Python notebook within Google Colab, use the following instructions to download the data into the working directory of the virtual machine:
  ```
  !wget https://download.pytorch.org/tutorial/data.zip
  !unzip data.zip
  ```
- Run the script at the end of the tutorial

Compare the accuracy of the encoder when varying the type of hidden units: linear units, gated recurrent units (GRUs) and long short term memory (LSTM) units. For linear hidden units, just run the script of the tutorial as it is. For GRUs and LSTMs, modify the code of the tutorial. Hand in the following material:

- Electronic copy of your code
- Graph that contains 3 curves (linear hidden units, GRUs and LSTM units). The y-axis is the test (validation) negative log likelihood and the x-axis is the number of thousands of iterations.
- Explanation of the results (i.e., why some hidden units perform better or worse than other units).

(b) **[20 pts]** Decoder implementation

- Go through the tutorial "Generating names with character-level RNN"
  `http://pytorch.org/tutorials/intermediate/char_rnn_generation_tutorial.html`
- Download the data associated with the tutorial. In your Python notebook within Google Colab, use the following instructions to download the data into the working directory of the virtual machine:
  ```
  !wget https://download.pytorch.org/tutorial/data.zip
  !unzip data.zip
  ```
- Run the script at the end of the tutorial

Compare the accuracy of the decoder when varying the information fed as input to the hidden units at each time step: i) previous hidden unit, previous character and category; ii) previous hidden unit and previous character; iii) previous hidden unit and category; iv) previous hidden unit. For i), just run the script of the tutorial as it is. For ii) and iv) modify the code to feed the category only as input to the first hidden unit. For iii) and iv), modify the code to avoid feeding the previous character as input to each hidden unit. Hand in the following material:

- Electronic copy of your code
- Graph that contains 4 curves (i, ii, iii, iv). The y-axis is the test (validation) negative log likelihood and the x-axis is the number of thousands of iterations.
- Explanation of the results (i.e., how does the type of information fed to the hidden units affect the results).

(c) **[20 pts]** Seq2seq implementation

- Go through the tutorial "Translation with a sequence to sequence model with attention"
  `http://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`
- Download the data associated with the tutorial. In your Python notebook within Google Colab, use the following instructions to download the data into the working directory of the virtual machine:
  ```
  !wget https://download.pytorch.org/tutorial/data.zip
  !unzip data.zip
  ```
- Run the script at the end of the tutorial

Compare the accuracy of the seq2seq model with and without attention. For the seq2seq model with attention, just run the script of the tutorial as it is. For the seq2seq model without attention, modify the code of the tutorial. Hand in the following material:

- Electronic copy of your code
- Graph that contains 2 curves (with attention and without attention). The y-axis is the test (validation) negative log likelihood and the x-axis is the number of thousands of iterations.
- Explanation of the results (i.e., how does attention affects the results).