DATA.ML.200 Pattern Recognition and Machine Learning

Exercise week 7: Do-It-Yourself RNN

Be prepared for the exercise sessions (watch the demo lecture). You may ask TAs to help if you cannot make your program to work, but don't expect them to show you how to start from the scratch.

1. **`python`** *Letter encoding (5 pts)*

   Download the text file '*abcde.txt*' from Moodle. Write code that reads the characters in the file one by one and converts them to a single matrix that has as many lines as characters in the file and columns represent one-hot-encoding.

   It can be convenient to first convert characters to integers by Python dictionary, and then convert integers to one-hot-encoding vectors.

   Your code should print the total number of characters and the size of vocabulary (number of unique characters). You can find help from the language processing Colab notebooks provided in the lecture notebook.

   For future purposes it is best to write this code as a function that takes the file name as input and returns the one-hot-encoded data matrix, vocabulary size and 'char_to_int' and 'int_to_char' Python dictionaries to convert integers to characters and characters to integers.

   Return the following items:

   - Python code: encoding.py that does all above.
   - PNG image: your full desktop screenshot that includes a terminal where the python file is executed and printing the requested results: encoding_screenshot.png

2. **`python`** *Letter prediction MLP (10 pts)*

   Implement a MLP network that takes a one-hot-encoded letter as input and predicts the next letter as one-hot-encoded output:

   $$\mathbf{x}_{t+1} = MLP(\mathbf{x}_t)$$

   Use the `'abcde.txt'` text as training data (you will need your one-hot-encoding function).

   Write a training function that forms training input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$ so that always the next letter is in the output matrix.

   Write also an evaluation function that randomly picks a starting letter from $\mathbf{X}$, and then generates a sentence of 50 characters using the trained network.

Your code should output the taining loss after each epoch and then a generated sequence.

Return the following items:

- Python code: char_rnn.py that does all above (may include your encoding function copy&pasted).
- PNG image: your full desktop screenshot that includes a terminal where the python file is executed and printing the requested results: char_rnn_screenshot.png

3. **python** *Letter prediction MLP V2 (10 pts)*

Download a new data file 'abcde_edcba.txt' and test your previous letter prediction with this one. If it does not work, then what would be the reason.

Implement a new MLP network that takes two one-hot-encoded letters as input and predicts the next letter as one-hot-encoded output:

$$\mathbf{x}_{t+1} = MLP(\mathbf{x}_t, \mathbf{x}_{t-1})$$

Write a training function that forms training input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$ so that always the next letter is in the output matrix. Inputs are always two previous letters.

Write also an evaluation function that randomly picks a starting letter from $\mathbf{X}$, and then generates a sentence of 50 characters using the trained network.

Your code should output the taining loss after each epoch and then a generated sequence.

Return the following items:

- Python code: char_rnn_v2.py that does all above (may include your encoding function copy&pasted).
- PNG image: your full desktop screenshot that includes a terminal where the python file is executed and printing the requested results: char_rnn_v2_screenshot.png