Deep Learning (Fall 2024)

Homework 3

Due: 23:59 Dec 6, 2024

In this homework, you are asked to build a recurrent neural network for character-level language model. You may design the network architecture by yourself, including the number of hidden layers, the size of hidden states, learning rate, sequence length and mini-batch size. You are allowed to use packages such as tensorflow, pytorch or any other package in this homework.

1 Preprocessing of Text

- You may use the given Shakespeare dataset or find other datasets online (novels, lyrics, etc.) to train your model. Make sure that your training dataset is large enough for the RNN to learn the structure. Here is a link where you may download some novels: https://www.gutenberg.org/
- To implement a character-level language model, you need to encode each character into a one-hot vector as input of RNN. You may use the code below:

```
import numpy as np
import io

data_URL = 'shakespeare.train.txt'
with io.open(data_URL, 'r', encoding='utf8') as f:
    text = f.read()

# Characters' collection
vocab = set(text)

# Construct character dictionary
vocab_to_int = {c: i for i, c in enumerate(vocab)}
int_to_vocab = dict(enumerate(vocab))

# Encode data, shape = [number of characters]
train_data = np.array([vocab_to_int[c] for c in text], dtype=np.int32)
```

2 Recurrent Neural Network

Train your RNN using the dataset from the previous part. Split it into training and validation sets if you use your own dataset. Use any optimizer you want to minimize the bits-per-character (BPC)

$$\mathbf{BPC} = -\frac{1}{\mathbf{T}} \sum_{t=1}^{\mathbf{T}} \sum_{k=1}^{\mathbf{K}} \mathbf{t}_{t,k} \log_2 \mathbf{y}_{t,k}(\mathbf{x}_t, \mathbf{w}) \tag{1}$$

where **y** denotes the output from RNN and **t** denotes the corresponding target value.

For a batch of input data, consider the following objective function:

$$E(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{BPC}(\{\mathbf{x}_t^n\}_{t=1}^{\mathbf{T}}) \tag{2}$$

$$= -\frac{1}{N\mathbf{T}} \sum_{n=1}^{N} \sum_{t=1}^{\mathbf{T}} \sum_{k=1}^{\mathbf{K}} \mathbf{t}_{t,k} \log_2 \mathbf{y}_{t,k}^n(\mathbf{x}_t^n, \mathbf{w}) \tag{3}$$

where *N* is the batch size, *T* is time step and *K* is the length of the one-hot vector.

1. Construct a standard RNN then show your (1) network architecture, (2) learning curve, (3) training error rate and (4) validation error rate.
2. Choose 5 breakpoints during your training process to show how well your network learns through more epochs. Feed some part of your training text into RNN and show the text output.
3. Compare the results of choosing different sizes of hidden states and sequence length by plotting the training loss vs. different parameters.
4. Construct another RNN with LSTM then redo 1. to 3. Also discuss the difference of the results between standard RNN and LSTM.
5. Use RNN or LSTM to generate some words by priming the model with a word related to your dataset. Priming the model means giving it some input text to create context and then take the output of the RNN. For example, use ”JULIET” as the prime text of Shakespeare dataset and run the model to generate 10 to 15 lines of output.

Hint:

- Initialize the hidden state to zero at the first iteration in every epoch.