Deep Learning for Mechanics (APL 745)

Homework – 4 (Optional)

Handed out: 03-04-2022 Submission due: NA

Instructions:

- We highly encourage handwritten homework for derivation and other theoretical part. Compile as single report containing solutions, derivations, figures, etc
- Zipped folder should be turned in on Sakai with the following naming scheme: HW4_Enrl_No.zip
- Collaboration is encouraged however all submitted reports, programs, figures, etc. should be an individual student's writeup. Direct copying will be considered cheating.
- Discussion on methods used, results obtained for programming assignment and technical discussion is essential. Homework problems that simply provide computer outputs with no technical discussion, Algorithms, etc. will receive no credit.

Problem 1:

In this problem, we will develop (step by step) a recurrent neural network with fully connected layers to classify words. In this problem, RNN reads words as a series of characters - outputting a prediction and hidden state at each step, feeding its previous hidden state into each next step. We take the final prediction to be the output, i.e. which class the word belongs to.

Load the following packages for this problem:

- PyTorch is a deep learning package. (other deep learning packages can also be used.)
- Numpy is the fundamental package for scientific computing with Python.
- Matplotlib is a famous library to plot graphs in Python.

Step 1: In this problem, we consider a few thousand surnames from 18 languages of origin, and predict which language a name is from based on the spelling. The data folder contains the following language.txt files: Greek, French, Czech, Dutch, Polish, Scottish, Chinese, English, Italian, Portuguese, Japanese, German, Russian, Korean, Arabic, Vietnamese, Spanish, Irish.

Step 2: Define the following model structure with different Python files as follows:

- Model-configuration-1: Build 2- linear layers which operate on an input and hidden state with 128 hidden units for both the layers.
- Model-configuration-2: Build 2- linear layers which operate on an input and hidden state with 256 hidden units for both the layers.

Step 3: Train the RNN

Step 4: Test the RNN model using the test data and make sure that the gradient calculation is disabled during this inference stage.

Write a computer code to answer the following questions:

(A) Use a suitable loss function for training the RNN model.
(B) Train all the above model-configurations with the following learning rate and weight decay training-configurations:
- Training configuration – 1: 0.005
- Training configuration – 2: 0.001

(C) Plot the following:
- Epoch vs Loss
- Create a confusion matrix and plot it, indicating for every actual language (rows) which language the network guesses (columns).

(D) Finally predict for the following names:
'Geoffrey', 'Hinton', 'Yann', 'LeCun', 'Yoshua', 'Bengio'

Problem 2:

Table 1 depicts two matrices. One of the form (5 × 5 one) represents an image. The second (3 × 3 one) represents a convolution kernel. (Consider the bias term to be zero)

a. How many values will be generated if we forward propagate the image over the given convolution kernel?
b. Calculate these values
c. Suppose the gradient backpropagated from the layers above this layer is a 3 × 3 matrix of all 1s. Write down the value of the gradient (with respect to the input) backpropagated out of this.

3	1	4	1	5
9	2	6	5	3
5	8	9	7	9
3	2	3	8	4
6	2	6	4	3

3	8	3
2	7	9
5	0	2

Table 1: Image Matrix (5 × 5) and convolution filter (3 × 3)

Problem 3:

Sentence Classification is a common problem in NLP. There are many ways to solve this problem. As simplest, you could just use machine learning algorithm such as Logistic regression, or any you can think of using such as a multilayer Perceptron (MLP). You could even take help from ConvNets or RNNs to give you good sentence vector, which you can feed into linear classification layer. Goal of this question is to make sure that you have a clear picture in your mind about all these possible techniques.

Let's say you have a corpus of 50K words. For the simplicity of this question, assume you have the trained word embedding matrix of size [50K*300] available with you, which can give you a word vector of size [1*300]. Consider one sentence having 10 words for the classification task and describe your approach for below techniques.

a. Design a one layer ConvNet which first maps the sentence to a vector of length 5 (with the help of convolution and pooling), then feeds this vector to fully connected layer with softmax to get the probability values f or possible 3 classes.
b. Clearly mention the sizes for your input, kernal, outputs at each step (till you get the final [3*1] output vector from softmax)
c. Please describe the effect of small filter size vs large filters size during the convolution. What would be your approach to s elect the filter sizes for classification task?
d. How can a simple RNN which is trained for language modeling be used to get the sentence vector?
e. Design a simple RNN which first maps the sentence to a vector of length 50 (with the help of convolution and pooling), then feeds this vector to fully connected layer with softmax to get the probability values for possible 4 classes.
f. Clearly mention the sizes of all the RNN components such as your input vector, hidden layer weight matrix, hidden state vectors, cell state vector, output layers (RNN components sizes would be same at each time stamp).