Homework 3

Convolution, Recurrent networks

Instructor: Dr. Souvik Chakraborty

APL 745: Deep Learning for Mechanics

19 March, 2024

Submission due on 05 April, 2024

Instructions:

(i) You are allowed to discuss in a group; however, you must submit your own handwritten homework copy (no computer-typed submission will be accepted). Further, copying homework from your friends is forbidden. If found copied, the homework submission will be considered invalid for all the students involved and will be graded zero.

(ii) Write all the steps, including your reasoning and the formulae you referred to, if any. If sufficient reasoning is not provided and steps are unclear, step marks will not be given.

(iii) For practical submissions, the codes are accepted in *.ipynb* or *.py* format.

(iv) Unless mentioned otherwise, only *numpy*, *pytorch* and *matplotlib* libraries may be used. Direct commands for algorithms to be implemented are not allowed.

(v) The *.rar* file containing all submission related files shall be named in the format, **Name_Entrynumber.rar**

Question 1.

A leaky integrate and fire neuron's dynamics can be described as follows,

$$\text{input at spike time step } t,\ I(t) = w\,x(t) + b,$$

$$\text{voltage at spike time step } t,\ V(t) = \beta V(t-1) + I(t),$$

$$\text{output at spike time step } t,\ y(t) = H(V(t) - T_h),$$

$$\text{if } y(t) = 1, \text{ reset voltage at spike time step } t \text{ i.e. set } V(t) = 0 \text{ before computation of next time step.}$$

In the above set of equations, representing leaky integrate and fire neuron's dynamics, $x(t)$ are the inputs, $y(t)$ are the outputs, $H(\cdot)$ is the Heaviside function, $t \in [1, T]$, β is the leakage parameter, and T_h is the neuron's threshold. The Heaviside function in the forward pass is described as,

$$H(\xi) = \begin{cases} 1, & \xi \geq 0 \\ 0, & \xi < 0 \end{cases}.$$

In the backward pass (during gradient computation), however, the same is idealized as,

$$H(\xi) \simeq \frac{\xi}{1 + k|\xi|} \quad ,$$

with the gradient defined as,

$$\frac{\partial H(\xi)}{\partial \xi} = \frac{1}{(1 + k|\xi|)^2} \quad .$$

Now,

1) Compute the gradient for *true* Heaviside function. Show, using plots, that the idealized gradient $\dfrac{\partial H(\xi)}{\partial \xi}$, shown above, approaches to true gradient as the value of k increases (given maximum value of Dirac delta function is *one*).
2) For $k = 25$, $T = 3$, $\boldsymbol{x} = [x(1), x(2), x(3)] = [1, 0, 1]$, $w = 0.5$, $b = 0.05$, $\beta = 0.95$, and $T_h = 0.75$, compute the output spike train $\boldsymbol{y} = [y(1), y(2), y(3)]$. To start the iterations, take $V(0) = 0$.
3) Taking $z = \sum_{i=1}^{T} y(t)$, $T = 3$, compute analytically, the gradients $\partial z/\partial w$, $\partial z/\partial b$, $\partial z/\partial x_3$, $\partial z/\partial x_2$, and $\partial z/\partial x_1$, where $x_i = x(i)$. Use the idealized gradient for the Heaviside function and,

a. take β and T_h as hyperparameters.

b. take β and T_h as trainable parameters. For this case, also compute the gradients $\partial z/\partial \beta$ and $\partial z/\partial T_h$.

Also, compute the numeric values of gradients in both the sub-parts. For constants/ variables/ inputs values, refer to part 2 above.

IMP For part 3, draw the computational graphs for both sub-parts, and the gradient calculations are to be done without Python/coding. For parts 1 and 2, numpy and matplotlib libraries may be used.

Question 2.

A. When designing a neural network to detect objects from C different classes in an image, we can use a 1-of-$(C+1)$ class label with one variable for each object class and one additional variable representing a 'background' class, i.e., an input image region that does not contain an object belonging to any of the defined classes. The network will then output a vector of probabilities of length $(C+1)$. Alternatively, we can use a single binary variable to denote the presence or absence of an object and then use a separate 1-of-C vector to denote the specific object class. In this case, the network outputs a single probability representing the presence of an object and a separate set of probabilities over the class label. Write down the relationship between these two sets of probabilities.

B. In mathematics, a convolution for a continuous variable x is defined by,

$$F(x) = \int_{-\infty}^{\infty} G(y)k(x-y)dy,$$

where k(x-y) is the kernel function. By considering a discrete approximation to the integral, explain the relationship to a convolutional layer, defined by,

$$C(j,k) = \sum_l \sum_m I(j+l, k+m)K(l,m)$$

in a CNN. In the above equation, $I(j,k)$ are the pixel intensities of the image $\mathbf{I}$, $K(l,m)$ are pixel values for filter $\mathbf{K}$ and $C(j,k)$ are the activation values for feature map $\mathbf{C}$.

Question 3. Multi-head attention in a transformer is defined by

$$\mathbf{Y}(\mathbf{X}) = \text{ Concat } [\mathbf{H}_1, \ldots, \mathbf{H}_H]\, \mathbf{W}^{(\text{o})}$$

Now, show that the expression of the multi-head attention can be rewritten in the form:

$$\mathbf{Y} = \sum_{h=1}^{H} \mathbf{H}_h \mathbf{X} \mathbf{W}^{(h)}$$

where $\mathbf{H}_h$ is given by

$$\mathbf{H}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)$$

and $\mathbf{W}^{(h)}$ is defined as:

$$\mathbf{W}^{(h)} = \mathbf{W}_h^{(v)} \mathbf{W}_h^{(\circ)}$$

Here we have partitioned the matrix $\mathbf{W}(o)$ horizontally into sub-matrices denoted $\mathbf{W}(o)$ each of dimension $D_v \times D$, corresponding to the vertical segments of the concatenated attention matrix. Since D_v is typically smaller than D, for example, $D_v = D/H$ is a common choice, this combined matrix is rank deficient. Therefore, using a fully flexible matrix to replace $\mathbf{W}_h^{(v)} \mathbf{W}_h^{(\circ)}$ would not be equivalent to the original formulation given in the text.

Question 4. Backpropogation through a Simple RNN

Consider the following 1-D RNN with no nonlinearities, a 1-D hidden state, and 1-D inputs u_t at each timestep. (Note: There is only a single parameter w, no bias). This RNN expresses unrolling the following recurrence relation, with hidden state ht at unrolling step t given by:

$$h_t = \dot{w(u_t + h_{t-1})}$$

The computational graph of unrolling the RNN for three timesteps is shown below: where w is the learnable weight, u1, u2, and u3 are sequential inputs, and p, q, r, s, and t are intermediate values. Now, write down the expression of

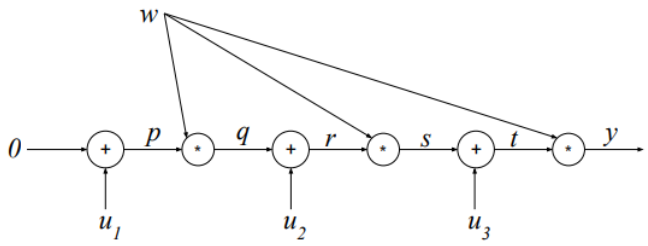

FIGURE 1. Weight sharing of immediate results in the RNN

(a) Forward pass of y and t
(b) Calculate the partial derivatives along each of the three outgoing edges from the learnable w
(c) Total derivative of y with respect to the weights
(d)Write down pseudo-code for the inference process in a trained RNN with an architecture of the form depicted in Fig.2.

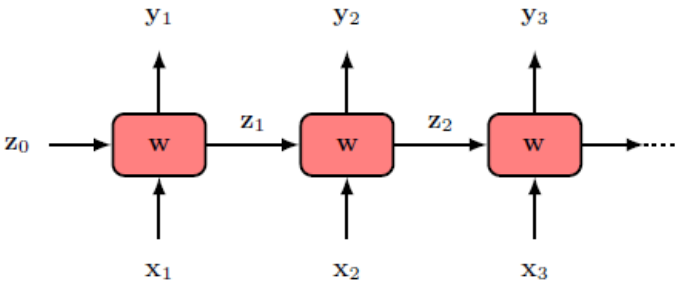

FIGURE 2. RNN architecture

Question 5. Convolutional Neural Network (CNN)
A. Table 1 and 2 depict two matrices. One of them (5×5) represents an image. The second (3×3 matrix) represents a convolution kernel. (Consider the bias term to be zero)

(i) How many values will be generated if we forward propagate the image over the given convolution kernel?

(ii) Calculate these values.

(iii) Suppose the gradient backpropagated from the layers above this layer is a 3×3 matrix of all 1s. Write the value of the gradient (w.r.t. input) backpropagated out of this layer.

TABLE 1. Image Matrix (5×5)

4	5	2	2	1
3	3	2	2	4
1	3	4	1	1
5	1	4	1	2
5	1	3	1	4

B. Consider a convolutional network layer with a 1-dimensional input array and a 1-dimensional feature map as shown in Fig 3, in which the input array has dimensionality 5 and the filters have width 3 with a stride of 1. Show that this can be expressed as a special case of a fully connected layer by writing down the weight matrix in which missing connections are replaced by zeros and where shared parameters are indicated by using replicated entries. Ignore any

TABLE 2. Convolution Kernel (3×3)

4	3	3
5	5	5
2	4	3

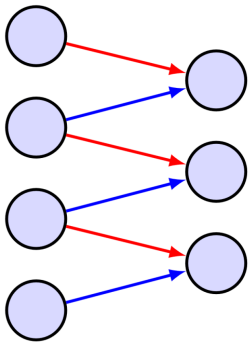

FIGURE 3. Illustration of convolution for a one-dimensional array of input values and a kernel of width 2. The connections are sparse and are shared by the hidden units, as shown by the red and blue arrows in which links with the same colour have the same weight values. This network there- fore has six connections but only two independent learnable parameters.

bias parameters.

C. In this exercise, we use one-dimensional vectors to demonstrate why a convolutional up-sampling is sometimes called a transpose convolution. Consider a one-dimensional strided convolutional layer with an input having four units with activations (x_1, x_2, x_3, x_4), which is padded with zeros to give $(0, x_1, x_2, x_3, x_4, 0)$, and a filter with parameters (w_1, w_2, w_3). Write down the one-dimensional activation vector of the output layer assuming a stride of 2. Express this output in the form of a matrix A multiplied by the vector $(0, x_1, x_2, x_3, x_4, 0)$.

Now consider an up-sampling convolution in which the input layer has activations (z_1, z_2) with a filter having values (w_1, w_2, w_3) and an output stride of 2. Write down the six-dimensional output vector assuming that overlapping filter values are summed, and that the activation function is just the identity. Show that this can be expressed as a matrix multiplication using the transpose matrix A^T.

Question 6. Sentence classification is a common problem in Natural Language Processing (NLP). There are many ways to solve this problem. As simplest, you could just use a 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 a good sentence vector, which you can feed into a linear classification layer. The 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 $[50K \times 300]$ available with you, which can give you a word vector of size $[1 \times 300]$. Consider one sentence having 10 words for the classification task and describe your approach for the 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 a fully connected layer with softmax to get the probability values for possible 3 classes.

(b) Clearly mention the sizes for your input, kernel, outputs at each step (till you get the final $[3 \times 1]$ output vector from softmax).

(c) Please describe the effect of a small filter size vs a large filter size during the convolution. What would be your approach to select the filter sizes for the 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 a 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 the same at each time stamp).