CSE 151B Deep Learning
UCSD, Spring 2022

Homework 2

Due 11:59 p.m, May 1st, 2022

Please submit your written solutions as a single PDF file using the provided LaTeX template on the course website. (https://sites.google.com/view/cse151b). You can use Overleaf (https://www.overleaf.com/) to compile LaTex files online. For problems requiring code, additionally submit all necessary code in one zip file. Code must run and produce the results reported in the PDF for full credit.

1 Convolutional Neural Networks

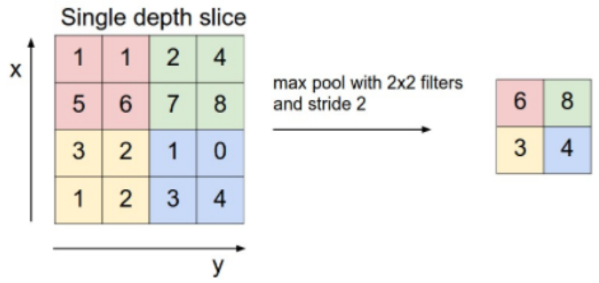
Problem A [2 points]: Convolution
Consider a single convolutional layer, where your input is a 32×32 pixel, RGB image. In other words, the input is a $32 \times 32 \times 3$ tensor. The convolution is designed to be:

- Size: $5 \times 5 \times 3$
- Filters: 8
- Stride: 1
- No zero-padding

i. [1 points]: What is the total number of parameters (weights) in this convolutional layer, including a bias term?

ii. [1 points]: What is the shape of the output tensor?

Problem B [2 points]: Pooling Pooling is a downsampling technique for reducing the dimensionality of a layer's output. Below is an example of max-pooling on a 2-D input space with a 2×2 filter and a stride of 2 (so that the sampled patches do not overlap):



Average pooling is similar except that you would take the average of each patch as its output instead of the maximum. Consider the following 4 matrices:

$$\begin{bmatrix}1&1&1&0\\1&1&1&0\\1&1&1&0\\0&0&0&0\end{bmatrix},\begin{bmatrix}0&1&1&1\\0&1&1&1\\0&1&1&1\\0&0&0&0\end{bmatrix},\begin{bmatrix}0&0&0&0\\0&1&1&1\\0&1&1&1\\0&1&1&1\end{bmatrix},\begin{bmatrix}0&0&0&0\\1&1&1&0\\1&1&1&0\\1&1&1&0\end{bmatrix}$$

i. [1 points]:

Apply 2×2 average pooling with a stride of 2 to each of the above images.

ii. [1 points]:

Apply 2×2 max pooling with a stride of 2 to each of the above images.

2 Recurrent Neural Networks

Problem A [4 points]: LSTM

Consider the following univariate version of the Long-Term Short-Term Memory (LSTM) model, with the updating rules:

$$\begin{bmatrix}f_t\\i_t\\o_t\end{bmatrix} = \sigma(\begin{bmatrix}w^{fx}\\w^{ix}\\w^{ox}\end{bmatrix} x_t + \begin{bmatrix}w^{fh}\\w^{ih}\\w^{oh}\end{bmatrix} h_{t-1}) \tag{1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(w^{cx}x_t + w^{ch}h_{t-1}), \qquad h_t = o_t \odot \tanh(c_t)$$

i. [2 points]: Derive the Backprop Through Time (BPTT) equations for each of the activations and the gates:

$$\delta f_t =?, \quad \delta i_t =?, \quad \delta o_t =?, \quad \delta c_t =?$$

ii. [2 points]: Based on your answers above, explain why the gradient does not explode if the values of the forget gates are very close to 1 and the values of the input and output gates are very close to 0. (**Hint**: consider the changes in h_t and c_t).

3 RNN Implementation: Poem Generation

William Shakespeare is perhaps the most famous poet and playwright of all time. He is known for works such as Hamlet and 154 sonnets, of which the most famous begins:

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:

Shakespeare's poems are nice for generative modeling because they follow a specific format, known as the Shakespearean (or English) sonnet.[1] Each sonnet is 14 lines, spread into 3 quatrains (section with 4 lines) followed by a couplet (section with 2 lines). The third quatrain is known as the *volta*[2] and has a change in tone or content. Shakespearean sonnets have a particular rhyme scheme, which is *abab cdcd efef gg*.

Shakespearean sonnets also follow a specific meter called *iambic pentameter*[3]. All lines have 10 syllables, and wit unstressed stress. For example, the famous Sonnet 22 begins:

Stress	x	\	x	\	x	\	x	\	x	\
Syllable	Shall	I	com -	pare	thee	to	a	sum-	mer's	day?

Here, each x represents an unstressed syllable and every \represents a stressed syllable. Try saying it out loud! The goal for this assignment is to generate poems that Shakespeare may have written by training an RNN on his 154 sonnets. His sonnets are available in the data file `shakespeare.txt`, provided in the data folder. You will submit all the code through the programming assignment.

Problem A [3 points]: Pre-processing

The first step is to pre-process the `poem_data` dataset before you train on it. How you pre-process is completely up to you. Here are a couple of questions to help you decide how to do pre-processing: How will you tokenize the data set? What will consist of a singular sequence, a poem, a stanza, or a line? Do you keep some words tokenized as bigrams? Do you split hyphenated words? How will you handle punctuation? It may be helpful to get syllable counts and syllable stress information from CMU's

[1] https://en.wikipedia.org/wiki/Sonnet#English_.28Shakespearean.29_sonnet
[2] https://en.wikipedia.org/wiki/Volta_%28literature%29
[3] https://en.wikipedia.org/wiki/Iambic_pentameter

Pronouncing Dictionary available on NLTK. You might also find the file `Syllable_-dictionary.txt`, provided in the data folder, to be helpful; please see the associated file called "syllable_dict_explanation" for an explanation.

i. [2 points]: Explain your choices of data pre-processing, as well as why you chose these choices initially. What was your final pre-processing? How did you tokenize your words, and split up the data into separate sequences? What changed as you continued on your project? What did you try that didn't work?

ii. [1 points]: Write about any data analysis you did on the dataset to help you make these decisions. Perform statistical analysis of the data and visualize them.

Problem B [4 points]: Model Training
Try doing poem generation using a recurrent neural network (RNN). Please follow these guidelines in this section:

- Train a **character-based LSTM** model. A single layer of 100-200 LSTM units should be sufficient. You should also have a standard fully-connected output layer with a softmax nonlinearity.
- Train your model to minimize categorical cross-entropy. Make sure that you train for a sufficient number of epochs so that your loss converges. You don't necessarily need to keep track of overfitting/keep a validation set.
- Your training data should consist of sequences of fixed length (40 characters is a good number for this task) drawn from the sonnet corpus. The densest way to do this is to take all possible subsequences of 40 consecutive characters from the dataset. To speed up training, using *semi-redundant* sequences (i.e. picking only sequences starting every n-th character) works just as well.
- To generate poems, draw softmax samples from your trained model. Try to lay around with the *temperature* parameter and generate different outputs, which controls the variance of your sampled text.

i. [1 points]: Explain in detail what model you implemented? What parameters did you tune? Comment on the poems that your model produced.

ii. [1 points]: Does the LSTM successfully learn sentence structure and/or sonnet structure? Report the runtime/amount of training data needed.

iii. [2 points]: Include generated poems using temperatures of 1.5, 0.75, and 0.25 with the following initial 40-character seed: "shall i compare thee to a summer's day?\n", and comment on their differences.