HW 1

Andrei A Simion

January 21, 2023

The goal of this notebook is to build a model that can generate real looking names. You will be given a file with real names and the goal is to have your model learn from this. The notebook given to you has most of this in it, but some items are taken out. The goal of the model we are building is similar to the Neural language model we looked at, where a window of tokens is looked at and we want to use the first part of that window to predict the last token. For example, if the window $n = 4$ we want to use 3 context tokens to predict the 4^{th} token.

The first part of this assignment has us reading the data. Each line in the data is a name, such as "matthew". We need to convert each such name into rows of (x, y) pairs to be used in the training / test / or validation sets. This is done for you, but you need to fill in the preprocessing. We have a few options. For example, we can get windows of data that are of length 4 directly, this gives ("mat", "t"), ("att", "h"), ("ath", "e") , ("the", "w"). Once we read the list of all names (the entire dataset), we can get a list of all unique characters and convert each character into a unique int. All characters in the names are in the standard alphabet and lowercased so there are 26. So, for example maybe you have ("att", "h") generates the tensor (x, y) pair ([1, 20, 20], [8]) assuming "a", "t" and "h" map to 1, 20 and 8, respectively. The problem with this approach, however, is that (1) for names like "bob" which have a length less than 4 we can't use and (2) there is a certain bit of information in the fact that "m" is the first character in the word whereas "w" is the second, and we can't really use this too much.

A better way might be to introduce a special "padding" token "." which we use to pad each word so that we have a start context and can predict the first letter of each word using this start context. For example, if the context is length 3, we can pad "matthew" with 3 tokens on the left and right side to get

"...matthew...". Now, we get further training samples ("...", "m") and ("..m", "a"), for example and we get some for the end like ("w..", ".") or ("hew", "."). Note that it might not be necessary to pad in this symmetric way, maybe "...matthew." is enough since we'd capture the beginning generation and also try to learn that "hew" can end a name. However, for this HW, pad in a symmetric way. The upshot is that some sort of padding is telling the model to learn that that "m" can be a first character in a name (we have ("...", "m")) and similarly that "w" is the last character (we have an (x, y) pair ("hew", ".")).

For this assignment you will implement this model. Fill in the notebook given and also (1) make sure "." maps to "0" (2) stoi and itos are inverse mapping of each other with length 27 (the vocabulary size). The model's code is very similar to the (lecture) model code and the same model we went over. At a high level, once you implement the data processing, the actual model code should be exactly similar to the code form the lecture.