Assignment 5
Character-level language model

1 Objective

- Building a Character level language model.

2 Problem Statement

We have collected a list of all the dinosaur names we could find, and compiled them into the given dataset. To create new dinosaur names, you will build a character level language model to generate new names. Your algorithm will learn the different name patterns, and randomly generate new names. Hopefully this algorithm will keep you and your team safe from the dinosaurs' wrath!

3 Requirements

1. You will find the pre-processing and creating the dictionary parts included in the starter code.

2. Your model will have the following structure:

   - Initialize parameters
   - Run the optimization loop
     - Forward propagation to compute the loss function.
     - Backward propagation to compute the gradients with respect to the loss function.
     - Clip the gradients to avoid exploding gradients.
     - Using the gradients, update your parameter with the gradient descent update rule.
   - Return the learned parameters

   At each time-step, the RNN tries to predict what is the next character given the previous characters. The dataset $X = (x^{\langle 1 \rangle}, x^{\langle 2 \rangle}, ..., x^{\langle T_x \rangle})$ is a list of characters in the training set, while $Y = (y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, ..., y^{\langle T_x \rangle})$ is such that at every time-step t, we have $y^{\langle t \rangle} = x^{\langle t+1 \rangle}$.

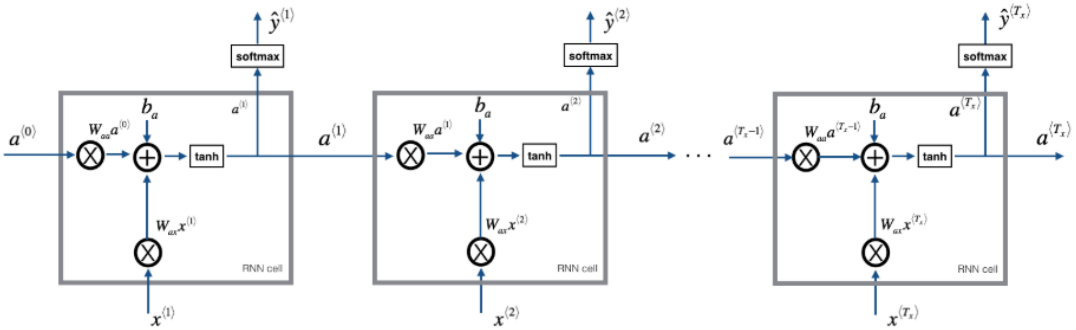

Figure 1

3. You should implement the clip function that you will call inside of your optimization loop. Recall that your overall loop structure usually consists of a forward pass, a cost computation, a backward pass, and a parameter update. Before updating the parameters, you will perform gradient clipping when needed to make sure that your gradients are not "exploding," meaning taking on overly large values.

   The clip function takes a dictionary of gradients and returns a clipped version of gradients if needed. There are different ways to clip gradients; we will use a simple element-wise clipping procedure, in which every element of the gradient vector is clipped to lie between some range [-N, N]. More generally, you will provide a maxValue (say 10). In this example, if any component of the gradient vector is greater than 10, it would be set to 10; and if any component of the gradient vector is less than -10, it would be set to -10. If it is between -10 and 10, it is left alone.

4. You should generate new text (characters). The process of generation is explained in Figure 2.

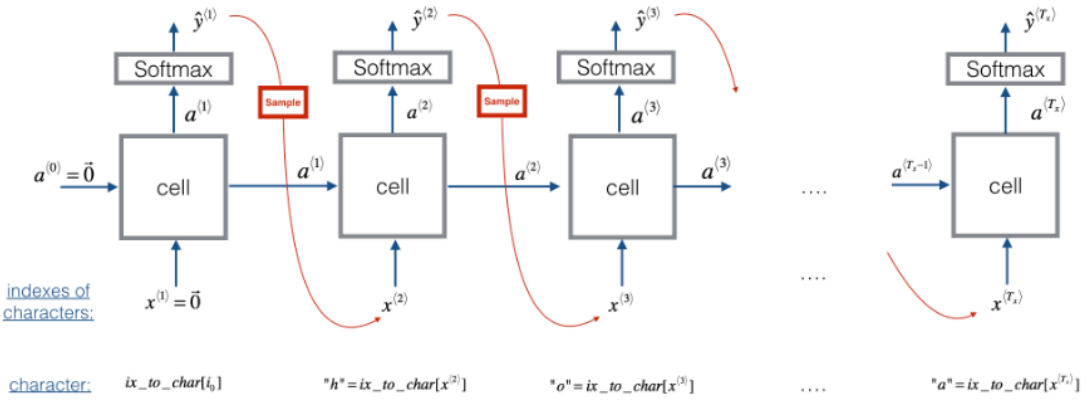

Figure 2

You should Implement the sample function to sample characters. You need to carry out 4 steps:

(a) Pass the network the first "dummy" input $x^{\langle 1 \rangle} = \vec{0}$ (the vector of zeros). This is the default input before we've generated any characters. We also set $a^{\langle 0 \rangle} = \vec{0}$

(b) Run one step of forward propagation to get $a^{\langle 1 \rangle}$ and $\hat{y}^{\langle 1 \rangle}$. Here are the equations:

$$a^{\langle t+1 \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t \rangle} + b)$$
$$z^{\langle t+1 \rangle} = W_{ya}a^{\langle t+1 \rangle} + b_y$$
$$\hat{y}^{\langle t+1 \rangle} = softmax(z^{\langle t+1 \rangle})$$

Note that $\hat{y}^{\langle t+1 \rangle}$ is a (softmax) probability vector (its entries are between 0 and 1 and sum to 1). $\hat{y}_i^{\langle t+1 \rangle}$ represents the probability that the character indexed by "i" is the next character. We have provided a softmax() function that you can use.

(c) Carry out sampling: Pick the next character's index according to the probability distribution specified by $\hat{y}^{\langle t+1 \rangle}$. This means that if $\hat{y}_i^{\langle t+1 \rangle} = 0.16$, you will pick the index "i" with 16% probability. To implement it, you can use np.random.choice.

(d) The last step to implement in sample() is to overwrite the variable x, which currently stores $x^{\langle t \rangle}$, with the value of $x^{\langle t+1 \rangle}$. You will represent $x^{\langle t+1 \rangle}$ by creating a one-hot vector corresponding to the character you've chosen as your prediction. You will then forward propagate $x^{\langle t+1 \rangle}$ in Step 1 and keep repeating the process until you get a "" character, indicating you've reached the end of the dinosaur name.

5. You should implement the following optimization process (one step of stochastic gradient descent):

   (a) Forward propagate through the RNN to compute the loss

   (b) Backward propagate through time to compute the gradients of the loss with respect to the parameters

   (c) Clip the gradients if necessary

   (d) Update your parameters using gradient descent.

6. Given the dataset of dinosaur names, we use each line of the dataset (one name) as one training example. Every 100 steps of stochastic gradient descent, you will sample 10 randomly chosen names to see how the algorithm is doing. Remember to shuffle the dataset, so that stochastic gradient descent visits the examples in random order.

   You should implement model() function. When examples[index] contains one dinosaur name (string), to create an example (X, Y), you can use this:

```
index = j % len(examples)
    X = [None] + [char_to_ix[ch] for ch in examples[index]]
    Y = X[1:] + [char_to_ix["\n"]]
```

   Note that we use: index= j % len(examples), where j = 1....num_iterations, to make sure that examples[index] is always a valid statement (index is smaller than len(examples)). The first entry of X being None will be interpreted by rnn_forward() as setting $x^{\langle 0 \rangle} = \vec{0}$. Further, this ensures that Y is equal to X but shifted one step to the left, and with an additional "" appended to signify the end of the dinosaur name.

   you should observe your model outputting random-looking characters at the first iteration. After a few thousand iterations, your model should learn to generate reasonable-looking names.

4 Notes

- Parts of this assignment are based on Andrew Ng Stanford deep learning course.
- The dataset files and the starter code for the problems can be found in the resources section in Piazza.
- Cheating will be severely penalized (for both parties). So, it is better to deliver nothing than deliver a copy. Any online resources used must be clearly identified.