Assignment 2: Language Models

Daniela Gottesman 209619402
Eden Lumbroso 208996587
Inbar Gat 316327055

May 29, 2023

1 Word-Level Neural Bi-gram Language Model

(a) The gradient of the cross entropy loss with respect to the input to the softmax function is:

$$\nabla_\theta CE(y,\hat{y}) = \nabla_{\hat{y}} CE(y,\hat{y}) \nabla_\theta \hat{y}$$

y is a one-hot vector. Let's denote $y_k = 1, y_{i\neq k} = 0$.

$$CE(y,\hat{y}) = -\log(\hat{y}_k)$$

The gradient with respect to $\hat{y}_k$ is:

$$\partial_{\hat{y}_k} CE(y,\hat{y}) = -\frac{1}{\hat{y}_k}$$

The gradients of $\hat{y}_k$ with respect to θ_k and θ_i are:

$$\partial_{\theta_k}\hat{y}_k = \frac{\exp(\theta_k)\sum_i \exp(\theta_i) - (\exp(\theta_k))^2}{(\sum_i \exp(\theta_i))^2} = \hat{y}_k(1-\hat{y}_k)$$

$$\partial_{\theta_{i\neq k}}\hat{y}_k = \frac{-\exp(\theta_k)\exp(\theta_i)}{(\sum_i \exp(\theta_i))^2} = -\hat{y}_k\hat{y}_i$$

To conclude, the gradients of the loss with respect to θ_k and θ_i are:

$$\partial_{\theta_k} CE = -\frac{1}{\hat{y}_k}\hat{y}_k(1-\hat{y}_k) = \hat{y}_k - 1$$

$$\partial_{\theta_{i\neq k}} CE = -\frac{1}{\hat{y}_k}(-\hat{y}_k\hat{y}_i) = \hat{y}_i$$

We can combine the results with the following notation:

$$\partial_{\theta_i} CE = \hat{y}_i - \delta_{ik}$$

(b) To calculate the gradient of the cross entropy with respect to the input of the neural network we will use the previous result, and also calculate the following derivatives:

$$\partial_{h_j}\theta_i = W_2(j,i)$$
$$\partial_{x_l}h_j = \sigma(\mathbf{x}\mathbf{W_1}+\mathbf{b_1})\cdot(1-\sigma(\mathbf{x}\mathbf{W_1}+\mathbf{b_1}))\cdot W_1(l,j)$$

Now to combine all derivative with the chain rule:

$$\frac{\partial J}{\partial x_l} = \sum_{i=1}^{D_y}\sum_{j=1}^{D_h}(\hat{y}_i-\delta_{ik})\cdot W_2(j,i)\cdot W_1(l,j)\cdot h\cdot(1-h)$$

(c) Implemented the required functions in the python files.

(d) The dev perplexiy achieved is: 39.58

2 Theoretical Inquiry of a Simple RNN Language Model

(a) To calculate the derivative with respect to U, we will use the chain rule and the result from question 1:

$$\frac{\partial J^{(t)}}{\partial U_{i,j}} = \frac{\partial J^{(t)}}{\partial \hat{y}^{(t)}}\frac{\partial \hat{y}^{(t)}}{\partial \theta}\frac{\partial \theta}{\partial U_{i,j}}$$

Where θ is the softmax argument. Using the result from question 1:

$$\frac{\partial J^{(t)}}{\partial U_{i,j}} = (\hat{y}_j-\delta_{jk})\cdot h_i^{(t)}$$

Like question 1, k is the index of the ground truth label.

$$\frac{\partial e^{(t)}}{\partial L_{x^{(t)}i}} = 1$$

$$\frac{\partial h_j^{(t)}}{\partial e_i^{(t)}} = a_j\cdot I_{i,j}$$

Where we use $a = h^{(t)}\cdot(1-h^{(t)})$ which is the derivative of the sigmoid function with respect to its argument.

$$\frac{\partial \theta_j}{\partial h_i^{(t)}} = U_{i,j}$$

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t)}i}} = \sum_{j=1}^{V}(\hat{y}_j-\delta_{jk})\sum_{j'=1}^{D_h}\cdot U_{j',j}\cdot a_{j'}\cdot I_{i,j'}$$

$$\frac{\partial J^{(t)}}{\partial I_{i,j}} = \frac{\partial J^{(t)}}{\partial h_j^{(t)}}\frac{\partial h_j^{(t)}}{\partial I_{i,j}} = \sum_{j'=1}^{V}(\hat{y}_{j'}-\delta_{j'k})\cdot U_{j,j'}\cdot a_j\cdot e_i^{(t)}$$

$$\frac{\partial J^{(t)}}{\partial H_{i,j}} = \frac{\partial J^{(t)}}{\partial h_j^{(t)}}\frac{\partial h_j^{(t)}}{\partial H_{i,j}} = \sum_{j'=1}^{V}(\hat{y}_{j'} - \delta_{j'k}) \cdot U_{j,j'} \cdot a_j \cdot h_i^{(t-1)}$$

$$\frac{\partial J^{(t)}}{\partial h_i^{(t-1)}} = \frac{\partial J^{(t)}}{\partial h_j^{(t)}}\frac{\partial h_j^{(t)}}{\partial h_i^{(t-1)}} = \sum_{j'=1}^{V}(\hat{y}_{j'} - \delta_{j'k}) \cdot \sum_{j=1}^{D_h} U_{j,j'} \cdot a_j \cdot H_{i,j}$$

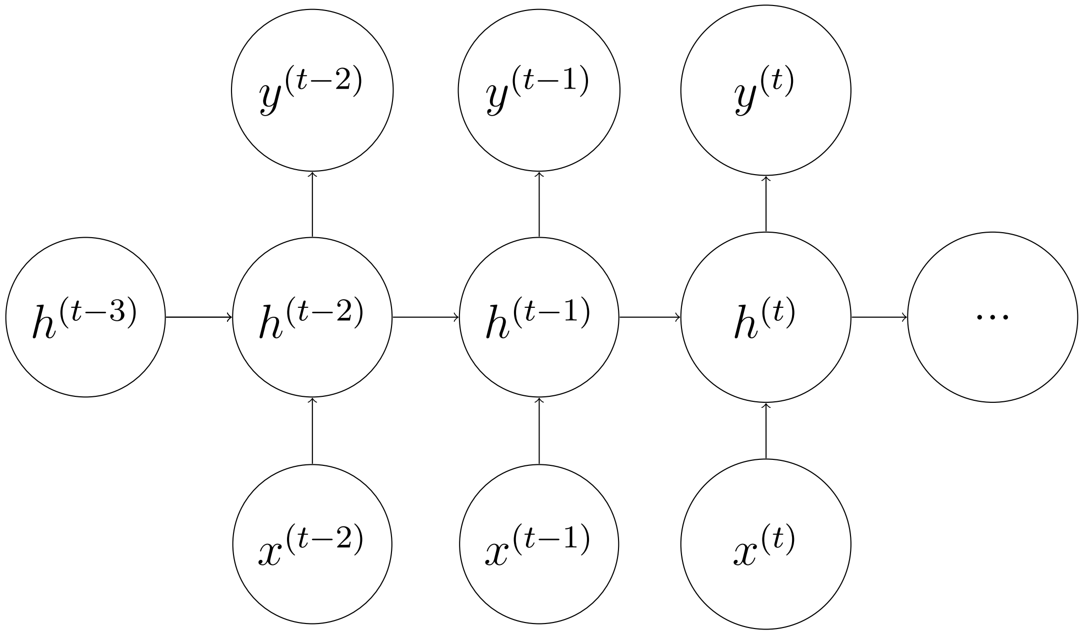
(b) We will now compute the gradients with respect to the previous time step: $t-1$:

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}i}} = \sum_{j=1}^{D_h}\frac{\partial J^{(t)}}{\partial h_j^{(t-1)}}\frac{\partial h_j^{(t-1)}}{\partial L_{x^{(t-1)}i}} = \sum_{j=1}^{D_h}\frac{\partial J^{(t)}}{\partial h_j^{(t-1)}} \cdot h_j^{(t-1)}(1-h_j^{(t-1)}) \cdot I_{i,j}$$

$$\left.\frac{\partial J^{(t)}}{\partial H_{i,j}}\right|_{(t-1)} = \frac{\partial J^{(t)}}{\partial h_j^{(t-1)}}\frac{\partial h_j^{(t-1)}}{\partial H_{i,j}} = \frac{\partial J^{(t)}}{\partial h_j^{(t-1)}} \cdot h_j^{(t-1)}(1-h_j^{(t-1)}) \cdot h_i^{(t-2)}$$

$$\left.\frac{\partial J^{(t)}}{\partial I_{i,j}}\right|_{(t-1)} = \frac{\partial J^{(t)}}{\partial h_j^{(t-1)}}\frac{\partial h_j^{(t-1)}}{\partial I_{i,j}} = \frac{\partial J^{(t)}}{\partial h_j^{(t-1)}} \cdot h_j^{(t-1)}(1-h_j^{(t-1)}) \cdot e_i^{(t-1)}$$

$$\left.\frac{\partial J^{(t)}}{\partial b_{1i}}\right|_{(t-1)} = \frac{\partial J^{(t)}}{\partial h_j^{(t-1)}} \cdot h_j^{(t-1)}(1-h_j^{(t-1)})$$



3 Generating Shakespeare Using a Character-level Language Model

In character-based language models, the vocabulary is of fixed size in contrast to word-based language models, however the notion of context is not well-defined between characters because there is no semantic relations between individual characters. In word-based language models, the vocabulary size is not consistent between different texts. Generation in word-based language models is easier because the vocabulary is already composed of real words in contrast to generation in character-based models there is a risk that gibberish words will be generated.

4 Perplexity

1. it holds that:

$$2^{\frac{-1}{M}\Sigma_{i=1}^{M} log_2 p(s_i|s_1\cdots s_{i-1})} = (2^{\Sigma_{i=1}^{M} log_2 p(s_i|s_1\cdots s_{i-1})})^{\frac{-1}{M}} = (\prod_{i=1}^{M} 2^{log_2 p(s_i|s_1\cdots s_{i-1})})^{\frac{-1}{M}} = (\prod_{i=1}^{M} p(s_i|s_1\cdots s_{i-1}))^{\frac{-1}{M}}$$

   And also:

$$e^{\frac{-1}{M}\Sigma_{i=1}^{M} ln p(s_i|s_1\cdots s_{i-1})} = (e^{\Sigma_{i=1}^{M} ln p(s_i|s_1\cdots s_{i-1})})^{\frac{-1}{M}} = (\prod_{i=1}^{M} e^{ln p(s_i|s_1\cdots s_{i-1})})^{\frac{-1}{M}} = (\prod_{i=1}^{M} p(s_i|s_1\cdots s_{i-1}))^{\frac{-1}{M}}$$

   And therefore:

$$e^{\frac{-1}{M}\Sigma_{i=1}^{M} ln p(s_i|s_1\cdots s_{i-1})} = 2^{\frac{-1}{M}\Sigma_{i=1}^{M} log_2 p(s_i|s_1\cdots s_{i-1})}$$

   As we wanted to prove.

2. - Character-Level LM perplexity:
     shakespeare: 7.159401893615723
     wikipedia: 20.321489334106445
   - Bi-gram LM perplexity:
     shakespeare: 10.27
     wikipedia: 67.92

3. The large gap in perplexity between the models is likely due to the fact that the RNN has much smaller vocabulary, and therefore the maximum perplexity is smaller for it. Also, the (very) small context length of the Bigram model limits its capabilities in contrast with the RNN model which has unlimitied context length in essence.
   The large gap between the passages is due to the training data. We trained the models on Shakespeare's work, which is more similar to the Shakespeare test passage, than to the Wikipedia test passage. We are essentially testing on a different distribution of text (although it is still English, it's a different kind of text), and therefore should expect higher perplexities.

5 Deep Averaging Networks

The graphs for sections a-d with corresponding explanations are on the next page...

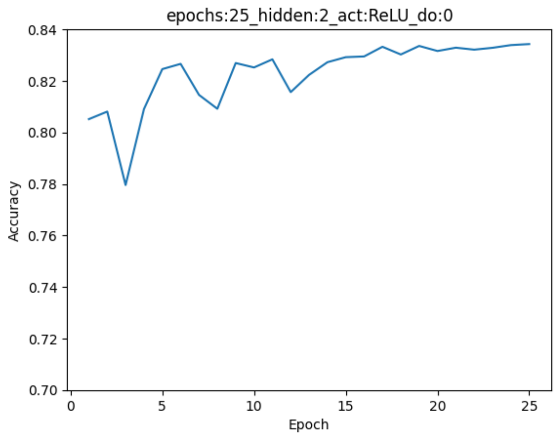
(a) The best model reaches 83.5%.
Batch size 40, learning rate starts at 0.001 and adaptively changes according to incurred loss.

(b) Introducing dropout didn't help the model generalize.

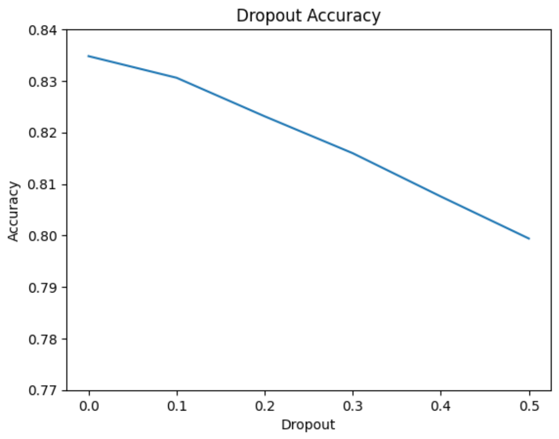
(c) Increasing hidden layers increases accuracy. Increasing to 1 hidden layer yields highest improvement, and improvements diminish as more layers are added.

(d) The LeakyReLU attained the highest accuracy. All three activations behave similarly.

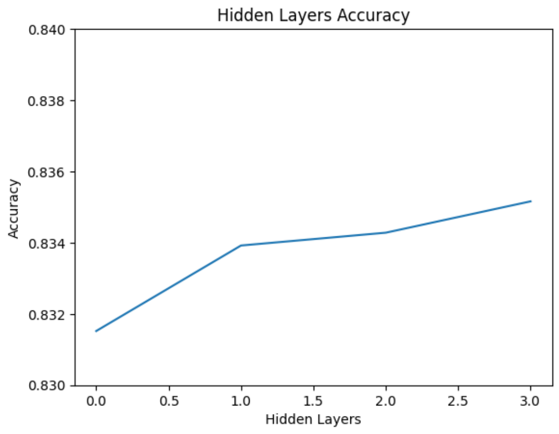
(e) The model misclassified negative texts that showed ambivalence or extremely long texts with sections of positive wording. Some examples are: "The film is somewhat entertaining", "The acting is alright, it could have better but I've certainly seen worse." The model also misclassified positive texts that showed reservation in the beginning and then turned out to be a good review towards the end. An example of this is: " I didn't have much faith in it being to good... But it's a good one for you and your buddies on a movie night."



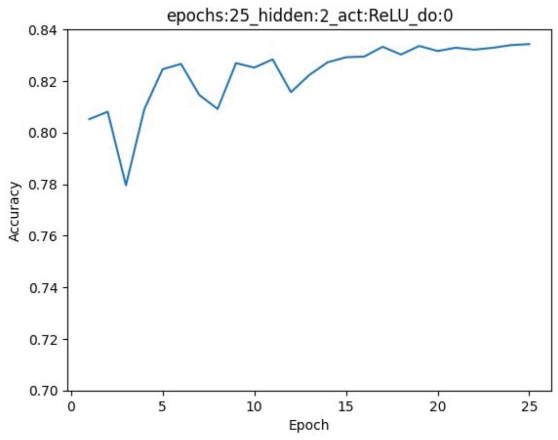
(a) 5a.



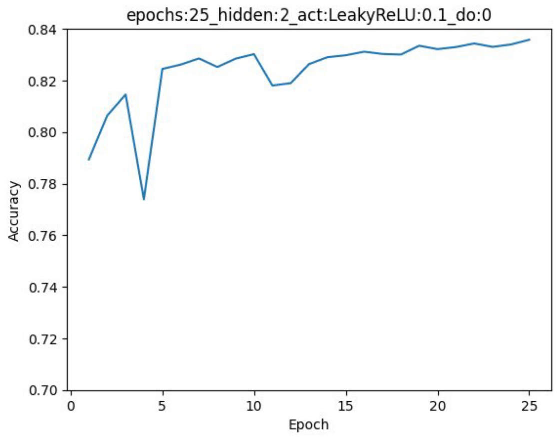
(b) 5b.



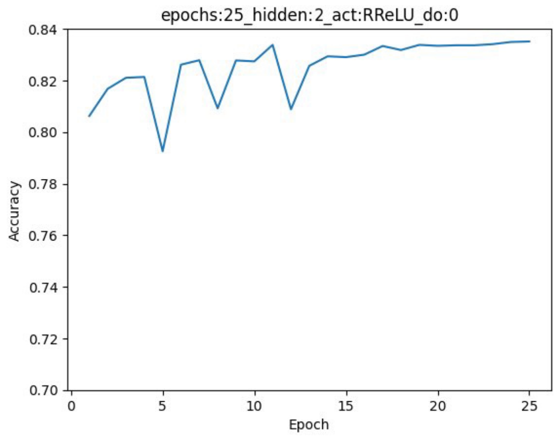
(c) 5c.



(d) 5d.



(e) 5d.



(f) 5d.

6 Right-to-left vs left-to-right Estimation

We want to show that:

$$p(x_0 x_1 x_3 \cdots x_n) = p(x_n)p(x_{n-1}|x_n)\cdots p(x_0|x_1) = p(x_0)p(x_1|x_0)\cdots p(x_n|x_{n-1})$$

From Bayes rule, it holds that:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$

Meaning, for every i = 0 ... n:

$$p(x_i|x_{i-1}) = \frac{p(x_{i-1}|x_i)p(x_i)}{p(x_{i-1})}$$

plugging it into the original statement:

$$p(x_0)p(x_1|x_0)\cdots p(x_n|x_{n-1}) = p(x_0)\frac{p(x_0|x_1)p(x_1)}{p(x_0)}\cdots\frac{p(x_{n-1}|x_n)p(x_n)}{p(x_{n-1})}$$

And from multiplication commutativity:

$$p(x_0)\frac{p(x_0|x_1)p(x_1)}{p(x_0)}\cdots\frac{p(x_{n-1}|x_n)p(x_n)}{p(x_{n-1})} = \frac{p(x_0)p(x_1)\cdots p(x_n)p(x_0|x_1)p(x_1|x_2)\cdots p(x_{n-1}|x_n)}{p(x_0)p(x_1)\cdots p(x_{n-1})} = p(x_n)p(x_{n-1}|x_n)\cdots p(x_0|x_1)$$

As we wanted to show.