Assignment - 2

1. Tensorflow Softmax

(c) Placeholder variables are dummy nodes that provide entry points for data to the computational graph.

Feed dictionary is used to input data (numpy arrays or similar) to computational graph through the placeholder variables.

(e) Tensorflow nodes in computational graph have attached gradient operations. Tensorflow, then uses backpropagation, using these node-specific gradient ops, to compute the required gradients for all variables in the graph. So, it removes the need for us to define gradients explicitly.

2. Deep Networks for Named Entity Recognition

Let's keep the notation consistent and define inputs and outputs of each layer as follows:

$$a_1 = z_1 = x^{(t)}$$

$$z_2 = a_1 W + b_1$$

$$a_2 = \tanh(z_2)$$

$$z_3 = a_2 U + b_2$$

$$a_3 = \text{softmax}(z_3) = \hat{y}$$

$$J(\theta) = CE(y, \hat{y}) = -\sum_{i=1}^{5} y_i \log \hat{y}_i$$

The dimensionality of above matrices and vectors are as follows:

$a_1 \in \mathbb{R}^{1\times 150}$, $z_2 \in \mathbb{R}^{1\times 100}$, $W \in \mathbb{R}^{150\times 100}$,

$b_1 \in \mathbb{R}^{1\times 100}$, $a_2 \in \mathbb{R}^{1\times 100}$, $U \in \mathbb{R}^{100\times 5}$,

$b_2 \in \mathbb{R}^{1\times 5}$, $a_3 \in \mathbb{R}^{1\times 5}$, $z_3 \in \mathbb{R}^{1\times 5}$

Let's compute the derivative of $\tanh(z)$ in advance.

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$1 + f(z) = \frac{2e^z}{e^z + e^{-z}} = \frac{2}{1 + e^{-2z}}$$

$$\log(1 + f(z)) = \log(2) - \log(1 + e^{-2z})$$

$$\frac{f'(z)}{1 + f(z)} = \frac{2e^{-2z}}{1 + e^{-2z}} = \frac{2}{1 + e^{2z}} = 1 - f(z)$$

$$f'(z) = (1 + f(z))(1 - f(z)) = 1 - f^2(z)$$

$$\boxed{\frac{d}{dz}\tanh(z) = 1 - \tanh^2(z)}$$

(a) $\dfrac{\partial J}{\partial U} = \dfrac{\partial J}{\partial z_3} \cdot \dfrac{\partial z_3}{\partial U}$

Now, using the fact that derivative of CE loss w.r.t. z_3 for softmax is $\hat{y} - y$, as derived is assignment 1,

$$\frac{\partial J}{\partial U} = \left((\hat{y} - y)^T a_2\right)^T = a_2^T (\hat{y} - y)$$

$$\boxed{\frac{\partial J}{\partial U} = a_2^T \delta_3}$$

$$\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_3} \cdot \frac{\partial z_3}{\partial b_2}$$

$$\boxed{\frac{\partial J}{\partial b_2} = \delta_3}$$

$$\boxed{\frac{\partial J}{\partial W} = a_1^T \delta_2}$$

$$\delta_2 = (1 - \tanh^2(z_2)) \circ (\delta_3 U^T)$$

$$\boxed{\frac{\partial J}{\partial b_1} = \delta_2}$$

$$\frac{\partial J}{\partial L_i} = \frac{\partial J}{\partial a_1}$$ 5 | 100

$$\frac{\partial J}{\partial a_1} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1}$$

$$\boxed{\frac{\partial J}{\partial a_1} = \delta_2 \cdot W^T} = \frac{\partial J}{\partial x^{(t)}}$$

let $i = x_t$, $j = x_{t+1}$, $k = x_{t+1}$,

$$\boxed{\frac{\partial J}{\partial x^{(t)}} = \left[\frac{\partial J}{\partial L_i}, \frac{\partial J}{\partial L_j}, \frac{\partial J}{\partial L_k}\right]}$$

(b) Since $\frac{\partial J_{reg}}{\partial b_1} = \frac{\partial J_{reg}}{\partial b_2} = \frac{\partial J_{reg}}{\partial L_i} = 0$

the gradients for b_1, b_2 and L_i stay the same.

$$\frac{\partial J_{reg}}{\partial W} = \lambda W$$

$$\frac{\partial J_{reg}}{\partial U} = \lambda U$$

So,

$$\frac{\partial J_{full}}{\partial W} = a_1^T \delta_2 + \lambda W$$

$$\frac{\partial J_{full}}{\partial U} = a_2^T \delta_3 + \lambda U$$

(d) Hyperparameters used in our model:

regularization $\lambda = 0.001$

dimension of hidden layer = 100

learning rate $\alpha = 0.001$

SGD batch size = 64

dropout with keep-prob. = 0.9

3. Recurrent Neural Networks : Language Modeling

(a) $PP^{(t)}(y^{(t)}, \hat{y}^{(t)}) = \dfrac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)}}$ ——— (A)

$= \dfrac{1}{\hat{y}_{j^*}^{(t)}}$ — ($\because$ $y^{(t)}$ is one-hot with 1 at j^*) (let's assume)

$J^{(t)} = CE(y^{(t)}, \hat{y}^{(t)}) = -\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$

$= -\log \hat{y}_{j^*}^{(t)}$

So, perplexity

$$\boxed{PP^{(t)}(y^{(t)}, \hat{y}^{(t)}) = 2^{J^{(t)}}}$$

(Arithmetic) mean cross-entropy loss $= \frac{1}{T} \sum_{t=1}^{T} J^{(t)}$

(geometric) mean perplexity $= \left[\prod_{t=1}^{T} PP^{(t)}(y^{(t)}, \hat{y}^{(t)}) \right]^{1/T}$

$= \left[\prod_{t=1}^{T} 2^{J^{(t)}} \right]^{1/T}$

$= \left[2^{\sum_{t=1}^{T} J^{(t)}} \right]^{1/T}$

Thus, minimizing (arithmetic) mean CE loss will also minimize the (geometric) mean perplexity across the training set.

- If model predictions were completely random, for vocabulary size $|V|$

$$\hat{y}_j^{(t)} = \frac{1}{|V|} \qquad \forall j \in [|V|]$$

$$\boxed{PP^{(t)}(y^{(t)}, \hat{y}^{(t)}) = |V|} \qquad \text{(from A)}$$

- for $|V| = 2000$

$$J^{(t)} = -\log \hat{y}_{j^*}^{(t)}$$

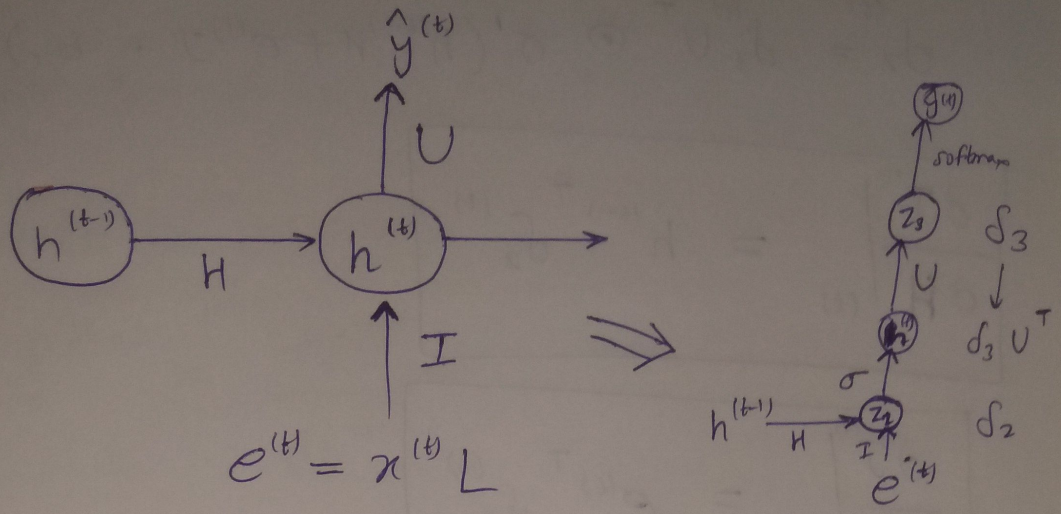
$$= -\log \frac{1}{2000}$$

$$\boxed{J^{(t)} = \log(2000)}$$

- for $|V| = 10000$

$$\boxed{J^{(t)} = \log(10000)}$$

(b)



$$e^{(t)} = x^{(t)} L$$

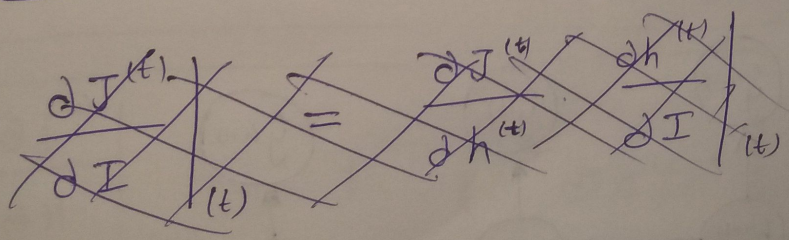
$$h^{(t)} = \text{sigmoid}\underbrace{\left(h^{(t-1)} H + e^{(t)} I + b_1\right)}_{z_2}$$

$$\hat{y}^{(t)} = \text{softmax}\underbrace{\left(h^{(t)} U + b_2\right)}_{z_3}$$

$$\frac{\partial J^{(t)}}{\partial U} = h^{(t)T} \delta_3^{(t)}$$

$$\delta_3^{(t)} = \hat{y}^{(t)} - y^{(t)}$$

$$\frac{\partial J^{(t)}}{\partial b_2} = \delta_3^{(t)}$$



$$\delta_2^{(t)} = \delta_3^{(t)} U^T \odot \sigma'(h^{(t-1)} H + e^{(t)} I + b_1)$$

$$\left.\frac{\partial J^{(t)}}{\partial H}\right|_{(t)} = h^{(t-1)^T} \delta_2^{(t)}$$

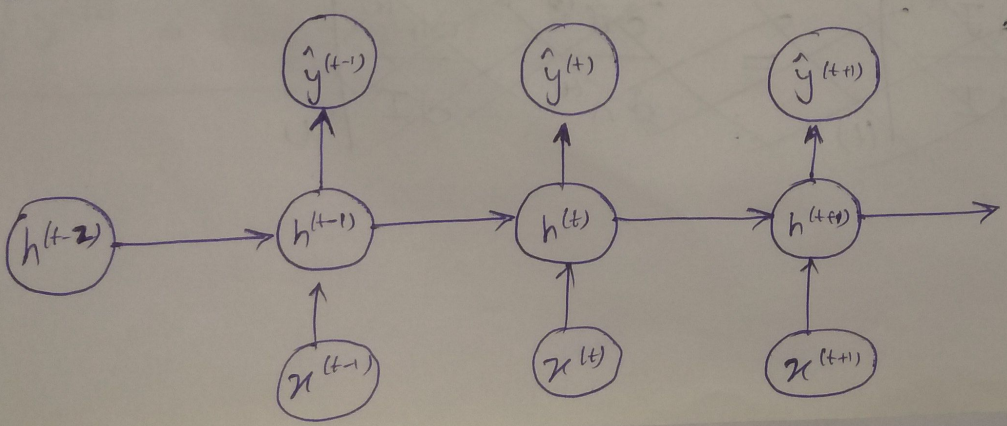
$$\left.\frac{\partial J^{(t)}}{\partial I}\right|_{(t)} = e^{(t)^T} \delta_2^{(t)}$$

$$\left.\frac{\partial J^{(t)}}{\partial b_1}\right|_{(t)} = \delta_2^{(t)}$$

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

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t)}}} = \delta_2^{(t)} I^T$$

(c) Unrolled n/w :



from derivatives of (b):

from notation of problem,

$$\delta^{(t-1)} = \frac{\partial J^{(t)}}{\partial h^{(t-1)}}$$

$$= \delta_3^{(t-1)} U^T \quad \text{[from notation of (b)]}$$

where

$$\delta_2^{(t-1)} = \delta_3^{(t-1)} U^T \odot \text{sigmoid}'\left(h^{(t-2)} H + e^{(t-1)} I + b_1\right)$$

$$\left.\frac{\partial J^{(t)}}{\partial I}\right|_{(t-1)} = e^{(t-1)^T} \left(\delta^{(t-1)} \odot \sigma'\left(h^{(t-2)} H + e^{(t-1)} I + b_1\right)\right)$$

$$\left.\frac{\partial J^{(t)}}{\partial b_1}\right|_{(t-1)} = \delta^{(t-1)} \odot \sigma'\left(h^{(t-2)} H + e^{(t-1)} I + b_1\right)$$

$$\left.\frac{\partial J^{(t)}}{\partial H}\right|_{(t-1)} = h^{(t-2)^T} \left(\delta^{(t-1)} \odot \sigma'\left(h^{(t-2)} H + e^{(t-1)} I + b_1\right)\right)$$

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}}} = \left(\delta^{(t-1)} \odot \sigma'\left(h^{(t-2)}H + e^{(t-1)}I + b_1\right)\right) I^T$$

d) Cost of multiplying matrix $A \in R^{b \times q}$ with $B \in R^{q \times r}$ is $O(bqr)$. So

Cost of forward prop.

$$= O\left(|V|\,d + D_h^2 + d D_h + D_h |V|\right)$$

Cost of backprop. for single step

$$= O\left(\text{cost of computing } \delta^{(t)}_{(t-1)} + \text{cost of computing BPTT gradients}\right)$$

$$= O\left(Dh^2 + D_h d + d D_h + D_h^2 + d|V|\right)$$

$$= O\left(D_h^2 + d D_h + d|V|\right)$$

Now, for backpropagating T time steps in time, we need to calculate/compute $\delta^{(t)}_{t-1}, \ldots, \delta^{(t)}_{t-\tau}$.

All other computations remain similar (just modifying $t-1$ by $t-\tau$ everywhere).

So, total cost of BPTT with τ steps

$$= O(\tau D_h^2 + d D_h + d|V|)$$

- of all operations above $|V| D_h$ is the slowest as size of vocabulary is generally very large. Also, $|V| d$ is costly operation as well.

(e) Hyperparameter used by our model:

batch_size = 64, embedding size = 50,
hidden layer size = 100, # epochs = 16,
backprop timesteps = 10, dropout with keep prob. = 0.9,
learning rate = 0.001
validation perplexity = 167.8159

(f) Some generated sentences:

- i have n't been given to the face weekly <eos>
- a book active rate on from a secondary market in the next year <eos>