6.806 Assignment 1
Fangzheng (Jack) Qian

## Smoothing

1. Applied to the training corpus, the log likelihood will be the largest when $\alpha = 0$, and smallest when $\alpha \to \infty$. Proof:

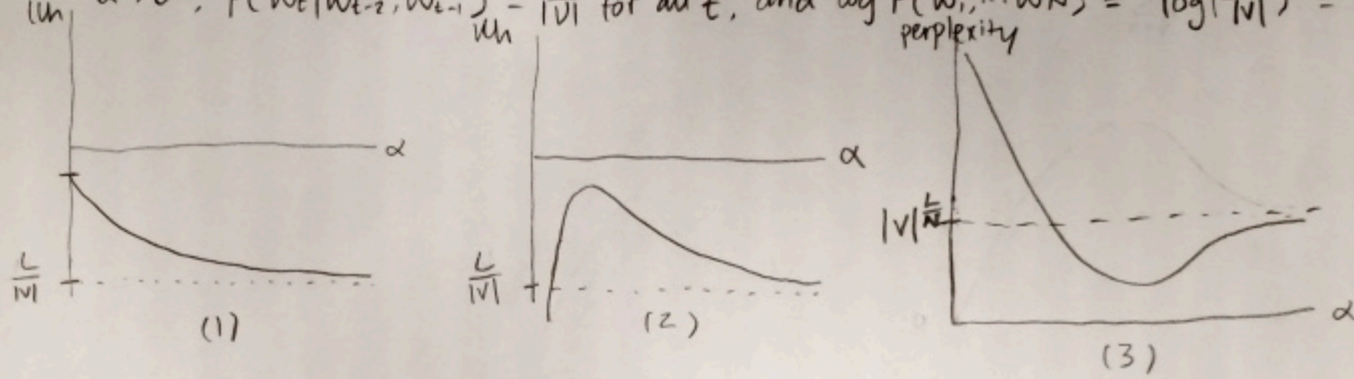
$$\log(P(w_1, w_2, \cdots w_N)) = \log \prod_{n=1}^{N} \frac{Count(w_t, w_{t-1}, w_{t-2}) + \alpha}{Count(w_{t-1}, w_{t-2}) + \alpha|V|}.$$

Suppose $C_1 = Count(w_t, w_{t-1}, w_{t-2})$ and $C_2 = count(w_{t-1}, w_{t-2})$.

$$\frac{C_1+\alpha}{C_2+\alpha|V|} - \frac{C_1}{C_2} = \frac{C_2(C_1+\alpha) - C_1(C_2+\alpha|V|)}{C_2(C_2+\alpha|V|)} = \frac{\alpha(C_2 - C_1\cdot|V|)}{C_2(C_2+\alpha|V|)} < 0,$$

because $C_1 \geq 1$ and $|V| > C_2$. [ $|V| > C_2$ is generally true for trigrams, unless with extremely large corpus and small vocabulary ].

Intuitively, probabilities are allocated to trigrams "most efficiently". In the worst case, whe $\alpha \to \infty$, $P(w_t | w_{t-2}, w_{t-1}) = \frac{1}{|V|}$ for all $t$, and $\log P(w_1, \cdots w_N) = \log(\frac{1}{|V|})^L = \frac{L}{|V|}$




2. Now using an unseen test corpus, the log likelihood is $\infty$ if $\alpha = 0$, because it will assign $-\infty$ to unseen words. When $\alpha \to \infty$, each word/trigram is again assigned equal probability, similar to the case discussed above.

However, there exists an $\alpha$ that maximizes the test corpus, which balances between the frequencies observed in the training corpus and the possibilities of unseen words

3. As an extension of question (2), perplexity $= 2^{-\frac{1}{N} \text{loglikelihood}}$

4. When $\alpha \to \infty$, perplexity $\to 2^{-\frac{1}{N}\log(\frac{1}{|V|})^L} = (\frac{1}{|V|})^{-\frac{L}{N}} = |V|^{\frac{L}{N}}$

When $\alpha = 0$, average likelihood $\to -\infty$, perplexity $\to \infty$.

## Neural Language Models

1.

| | Serial | Parallel | # units |
|---|---|---|---|
| ① read | $(n-1)d$ | 1 | $(n-1)d$ |
| ② $\underline{x} \to \underline{z}^h$ | $[(n-1)d+1]\cdot m$ | $(n-1)d+1$ | $m$ |
| ③ $\underline{z}^h \to \underline{f}^h$ | $m$ | 1 | $m$ |
| ④ $\underline{f}^h \to \underline{z}^o$ | $(1+m)\cdot|V|$ | $1+m$ | $|V|$ |
| ⑤ $\underline{z}^o \to \underline{p}$ | $|V|$ | 1 | $|V|$ |

Therefore, the computation needed is capped by

$$(n-1)d + [(n-1)d+1]\cdot m + m + (1+m)|V| + |V| \rightarrow O(ndm + mv)$$

$$\rightarrow O((nd+V)\cdot m)$$

- Shown in the table above, with unlimited computational resource,

$$O(1+(n-1)d+1+1+1+m+1) \longrightarrow O(nd+m).$$

- $$\delta_k^o = \frac{\partial \log P(w_t|w_{t-1},w_{t-2})}{\partial z_k^o} = \frac{d}{dz_k^o}\left(\log \frac{\exp(z_w)}{\sum_l \exp(z_l)}\right) = \frac{d}{dz_k^o}\left(\log(\exp(z_w)) - \log\sum_l \exp(z_l)\right)$$

$$= \frac{\partial z_w}{\partial z_k^o} - \frac{\partial \log\left(\sum_{l\neq k}\exp(z_l) + \exp(z_k)\right)}{\partial z_k^o} \qquad \frac{\exp(z_k)}{\sum \exp(z_l) + \exp(z_k)} = p_k$$

$$= \begin{cases} 1 - p_k & \text{if } k = w \\ -p_k & \text{if } k \neq w \end{cases}$$

$$\delta_j^h = \frac{\partial \log p_w}{\partial z_j^h}$$

$$= \frac{\partial \log p_w}{\partial f_j^h}\cdot\frac{\partial f_j^h}{\partial z_j^h} = \frac{\partial f_j^h}{\partial z_j^h}\cdot\sum_k \frac{\partial \log p_w}{\partial z_k^o}\cdot\frac{\partial z_k^o}{\partial z_j^h}$$

$$= \left(1-(f_j^h)^2\right)\sum_{k=1}^{|V|}\delta_k^o\cdot W_{jk}^o$$

$$\delta_i^x = \frac{\partial \log p_w}{\partial x_i}$$

$$= \sum_{j=1}^{m}\frac{\partial \log p_w}{\partial z_j^h}\cdot\frac{\partial z_j^h}{\partial x_i} = \sum_{j=1}^{m}\delta_j^h\cdot w_{ij}^h$$

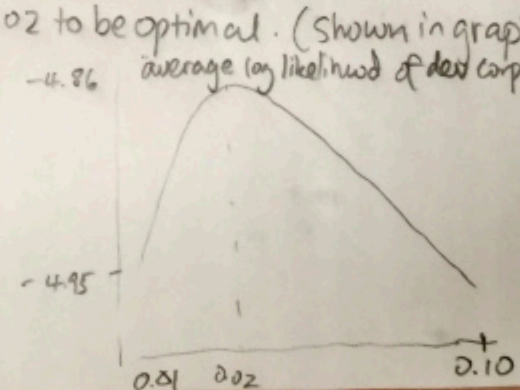
Step to update $v(w_{t-2})$

- take a trigram randomly or in serial : $w_i, w_{i-1}, w_{i-2}$
- select the corresponding transformation/lookup matrix to calculate the input vector and then activation function and probability
- calculate the gradient using the formulae above, and use the gradient to update the parameters.

  For $v(w_{t-2})$ specifically, $\underbrace{[v(w_{t-2})\ v(w_{t-1})]}_{V_{new}^{*}} \longleftarrow V_{old} + \underbrace{\eta}_{\text{learning rate}} \nabla_V \log P_w$

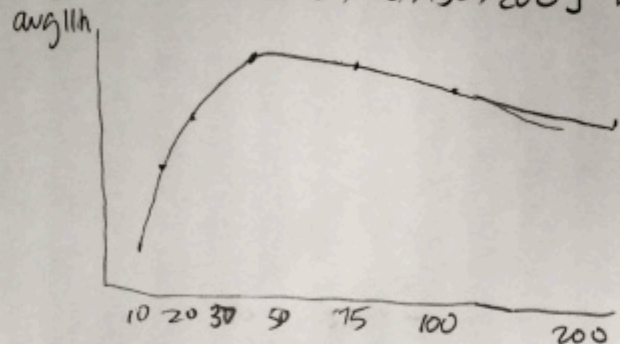
  and in this case $v(w_{t-2})$ and $v(w_{t-1})$ are updated individually.

- (a) Looping through different $\alpha$'s, I found $\alpha=0.02$ to be optimal. (Shown in graph below)
  With $\alpha=0.02$, the test likelihood is $-4.8457$.



(b) My best configuration is $n = 3$, $d = 75$, $m = 60$. The test likelihood is $-4.5873$.
First, I conducted a "coarse" grid search;
$n \in [2, 3, 4, 5]$, $d \in [5, 10, 20, 50]$, $m \in [10, 30, 50, 100]$
and found that tri-gram has the best generalization, whereas larger $d$ and $m$ give better performance
Next, I conducted a 2nd coarse search, and found that performance decreases when $d$ and $m$ exceed some large values, around 100.
Then, I conducted finer searches, and found $d=75$ and $m=60$ to be optimal

(c) As discussed above, the performance decreases after $m$ reaches some level. To be more exact and also time-saving, I picked $n=3$, $d = 10$ and experimented $m \in [10, 20, 30, 50, 75, 100, 150, 200]$ and obtained :




5. (a) No, it usually won't be zero.
I noticed that when indexing, all low frequency words are indexed to 0, and thus all bundled together. Therefore if a word in test is unseen in training, it is likely that it got bundled with other low-frequency words in the training set and assinged a probablity greater than 0. This probability may not make sense though

(b) Yes it can assign somewhat reasonable probability
This is mainly due to how input vectors are constructed - preceeding combinations don't have to be exact, but being similar is good enough.
For example, if "I am cool" and "You are cool" are in training and we encounter "We are cool" in test, then we expect the activation and softmax layers give high probabilities to cool because "we are" should be spatially close to "I am" and "You are" in the vector space.

6. The first sentence is more likely.
Average LLH (1) = $-5.2679$
Average LLH (2) = $-6.0524$. with $n=3$, $d=75$ and $m=60$.

## Recurrent Neural Network

1. Taking away $h^{t-1}$, the model acts as a bigram. Normally, we can find parameters that "overfit" the sentence to make probability close to 1, but we <u>CANNOT</u> in this case. Note that "very" appears twice, and therefore $P(\text{very}|\text{very}) + P(\text{cool}|\text{very}) \leq 1$. The maximum possible product of these two terms alone is much smaller than 1 ($\leq 0.25$)

2. Yes.
   Removing $W_{t-1}$ makes the network able to be treated as any n-gram except that the number of hidden unit is smaller than vocabulary. Parameters could be manipulated to span all existing n-grams and assign probabilities of 1 to each.