# 6.864 Problem Set #1

Dongyoung Kim

September 27, 2015

## Smoothing

**1.** The straightforward average log-likelihood is given by the following

$$l = \frac{1}{L}\sum_{L}^{i=1} \log p(w_i|w_{i-1}, w_{i-2})$$

$$l_{MLE} = \frac{1}{L}\sum_{L}^{i=1} \log \frac{\mathrm{c}(w_{i-2}, w_{i-1}, w_i)}{\mathrm{c}(w_{i-2}, w_{i-1})}$$

Where $w_i$ refers to the $i^{\text{th}}$ word and $\mathrm{c}(\cdot)$ refers to the empirical count.

Now, let's look at the case with $\alpha$-smoothing. The log-likelihood now looks like

$$\tilde{l}(\alpha) = \frac{1}{L}\sum_{L}^{i=1} \log \frac{\mathrm{c}(w_{i-2}, w_{i-1}, w_i) + \alpha}{\mathrm{c}(w_{i-2}, w_{i-1}) + \alpha|V|}$$

It is easy to see that as $\alpha \to \infty$, $\tilde{l}(\alpha)$ will converge to $\log \frac{1}{|V|} = -\log|V|$

When $\alpha = 0$, $\tilde{l}(\alpha) = l_{MLE}$ and thus the maximum log-likelihood.

For all values for $\alpha$ in between, $\tilde{l}(\alpha)$ will be monotonically decreasing

**2.** The expression for the test log likelihood is given as follows

$$\tilde{l}_{test}(\alpha) = \frac{1}{L}\sum_{L}^{i=1} \log \frac{\mathrm{c}(w_{i-2}, w_{i-1}, w_i) + \alpha}{\mathrm{c}(w_{i-2}, w_{i-1}) + \alpha|V|}$$

When $\alpha = 0$, words that were unseen in the training corpus will be assigned a probability of zero. Assuming that these words appear in the test corpus, the test likelihood will be equal to zero, i.e. the test log-likelihood will be $-\infty$. This is a problem of *overfitting* to the training dataset.

When $\alpha \to \infty$, $\tilde{l}_{test}(\alpha) = \log \frac{1}{|V|} = -\log|V|$

Therefore, either $\alpha = 0$ or $\alpha = \infty$ are two obviously bad choices for $\alpha$, and a global maximum will occur in some midpoint $\alpha^*$. $\tilde{l}_{test}(\alpha)$ will monotonically increase from $\alpha = 0$ to $\alpha^*$ and then monotonically decrease.

**3.** Similar to above,

$$\text{Perplexity} = 2^{-\tilde{l}_{test}(\alpha)} = |V| \text{ for } \alpha = \infty$$
$$\text{Perplexity} = 2^{-\tilde{l}_{test}(\alpha)} = \infty \text{ for } \alpha = 0$$

Global minimum occurs at the same $\alpha^*$ discussed in part 2. The perplexity monotonically decreases until $\alpha^*$ and then monotonically increases

**4.** For the answer, refer to part 3. For the explanation, refer to part 2.

# Neural Language Models

**1. Input Layer:** We have $d(n-1)$ units in the input layer. Counting one input as one operation, there are total $\boldsymbol{d(n-1)}$ operations.

**Hidden Layer:**

- **Input Signals:** For each $z_j^h$, we have to calculate $\sum_{i=1}^{d(n-1)} x_i w_{ij}^h$ which are $d(n-1)$ operations. Since there are $j = 1, 2, ..., m$ units in the hidden layer, the total number of operations to compute the input signals for the hidden layer is $\boldsymbol{d(n-1)m}$
- **Non-Linearity:** At each unit of the hidden layer we compute $f_j^h = \tanh(z_j^h)$ for $j = 1, ..., m$. The total number of operations for this case is $\boldsymbol{m}$
- The total number of operations in the hidden layer is $\boldsymbol{d(n-1)m + m}$

**Output Layer:** For each $z_j^o$, we have to calculate $\sum_{m}^{j=1} f_j^h W_{jk}^o$, which are $m$ operations. Since there are $|V|$ units in output layer, the total number of operations is $\boldsymbol{m|V|}$

**Softmax:** For normalization over $|V|$, we simply perform $\boldsymbol{|V|}$ operations.

Adding the total number of operations in each layer, we get $d(n-1) + d(n-1)m + m + m|V| + |V|$. Using the $O$ notation, we can express it as $\boldsymbol{O(d(n-1)m + m|V|)}$

**2.**

- **Input Layer:** 1 operation per node
- **Hidde Layer:** $d(n-1)+1$ operations per node
- **Output Layer:** $m$ operations per node
- **Softmax:** 1 operations per node

The GPU-optimized number of operations is $\boldsymbol{d(n-1) + m + 3}$, or $\boldsymbol{O(d(n-1) + m)}$

**3.** For ease of notation, we use $P \triangleq P(w_t = y|w_{t-1}, w_{t-2})$

When $y = k$,

$$\delta_k^o = \frac{\partial \log P}{\partial z_k^o} = 1 - \frac{\exp(z_k^o)}{\sum_{|V|}^{l=1} \exp(z_l^o)}$$

When $y \neq k$,

$$\delta_k^o = \frac{\partial \log P}{\partial z_k^o} = -\frac{\exp(z_k^o)}{\sum_{|V|}^{l=1} \exp(z_l^o)}$$

For further layers,

$$\begin{aligned}
\delta_j^h &= \frac{\partial \log P}{\partial z_j^h} = \frac{\partial f_j^h}{\partial z_j^h} \frac{\partial \log P}{\partial f_j^h} \\
&= \frac{\partial f_j^h}{\partial z_j^h} \sum_{k=1}^{|V|} \frac{\partial \log P}{\partial z_k^o} \frac{\partial z_k^o}{\partial f_j^h} \\
&= \frac{\partial f_j^h}{\partial z_j^h} \sum_{k=1}^{|V|} \delta_k^o \frac{\partial z_k^o}{\partial f_j^h} \\
&= \frac{\partial}{\partial z_j^h} \tanh(z_j^h) \sum_{k=1}^{|V|} \delta_k^o W_{jk}^o \\
&= (1 - \tanh^2 z_j^h) \sum_{k=1}^{|V|} \delta_k^o W_{jk}^o
\end{aligned}$$

$$\delta_i^x = \frac{\partial \log P}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial \log P}{\partial z_j^h} \frac{z_j^h}{\partial x_i} = \sum_{j=1}^{m} \delta_j^h W_{ij}^h$$

The update is given as follows where $\eta$ is the learning rate

$$x_i \leftarrow x_i + \eta \delta_i^x, i \in 1, ..., d$$

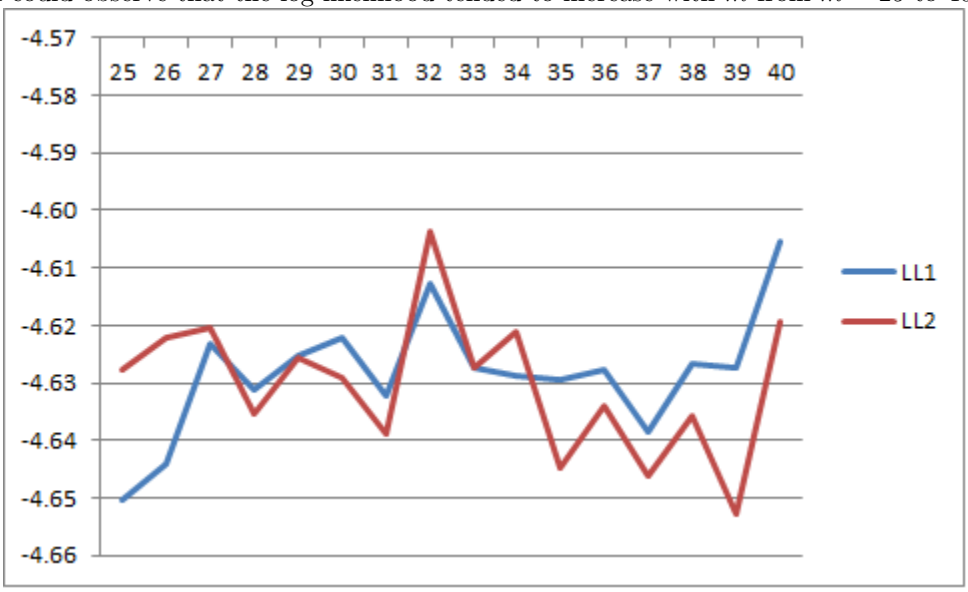
**4.**

**(a)** While bi-gram with smoothing clearly had a global maximum value for $\alpha$, the results with Neural Networks constantly changed as I input various random seeds for initializing edge weights. My configuration and results for both models is as seen below

- **Bigram**: $\alpha = 0.02$
  Log-likelihood on test corpus = -4.84574151936
- **Neural Network**: $n = 3, d = 9, m = 28$
  Log-likelihood on test corpus = $-4.59551661893$

**(b)** To find the best configuration, I tested on the dev set over all permutations with $d \in \{5, ..., 15\}$ and $m = \{25...40\}$ . However, it was clear to see that there was not any clear pattern over the hyperparameters. Furthermore, the result was volatile over different random seeds even for the same configuration. The best result was given by $n = 3, d = 9, m = 28$, with the log-likelihood on the development corpus given by $-4.59878717159$, and on the test corpus given by $-4.59551661893$

**(c)** While fixing $n = 3, d = 10$, I varied $m$ from $m \in \{25, ..., 40\}$ over two different random seeds and investigated the effect on the log-likelihood on the development corpus. While the results were fairly stochastic, I could observe that the log-likelihood tended to increase with $m$ from $m = 25$ to 40




**5.**

**(a)** Yes, the neural network will try to maximize the log-likelihood of the training corpus by assigning weights such that the probabilities of the unseen words will be near-zero.

**(b)** Yes. If $w_t$ appears many times in the training set, the weights will be adjusted such that the signal into the output node for $w_t$ will be large in general.

**6.**

- **the choice of their class is good <END> :** log-likelihood was $-4.33420672257$
- **the choice of there class is good <END> :** log-likelihood was $-4.88568029646$