MLCUB Reading Notes

1 Neural Network

- Neural Network becomes so popular because of the boost Deep Learning techniques.
- Single neural receives $N$ inputs and computes 1 output
- $M$ neurals receive $N$ inputs and compute $M$ outputs
  - A neural network = running several Logical Regression at the same time

\[ z = Wx + b \]
\[ a = f(z) \]
\[ W \in \mathbb{R}^{n \times m} \]
\[ x \in \mathbb{R}^n \]
\[ b, a, z \in \mathbb{R}^m \]
$f$ is a non-linear function, like logistic sigmoid or tanh, map any real number to $[0, 1]$ or $[-1, +1]$.

- The output layer could be a softmax or binary notation, depends on the task.
- Training
  Compute gradient of example-wise loss wrt parameters

2 Machine Translation

- Standard Machine Translation system is a noisy channel model, given Source sentence $S$, find out the most possible translation $\hat{T}$.

$$\hat{T} = \arg \max_{T \in \text{Target}} P(T|S)$$

$$= \arg \max_{T \in \text{Target}} P(S)P(S|T)$$

- The Decoder means the processing to find out the $\hat{T}$ from all the translation model and language model. It’s a NP-Hard problem. Typical MT system uses a beam search algorithm.
- The rescoring means the re-ranking of the generated translation sentences.

3 Neural Network Joint Model

Neural Network Joint Model (NNJM) is presented by the authors. It is almost identical to the original feed-forward Neural Network Language Model (NNLM)

3.1 Neural Network Language Model

- N-gram language model: Computes the probability of the word $W_K$ by giving its $n-1$ words $W_{K-n+1}, W_{K-n+2}, ..., W_{K-1}$

3.2 Neural Network Joint Model

- Similar to NNLM, but add the source context while being in the decoding.

$$P(T|S) \approx \prod_{i=1}^{[T]} P(t_i|t_{i-1}, ..., t_{i-n+1}, S_i)$$

S: 我 就 取 钱 给 了 她们
   i will get money to them

T: 我 就 取 钱 给 了 她们
   will get the money to them

$P(\text{the | get, will, i, 就, 取, 钱, 给, 了} )$
4 Self-normalized

To train the NNJM model, formally to maximize the log-likelihood of the training data:

\[
L = \sum_i \log(P(x_i))
\]

To speed up of the softmax log-likelihood, try to approximate the normalizer as 0.

\[
\log(P(x)) = \log \left( \frac{e^{U_r(x)}}{Z(x)} \right) \\
= U_r(x) - \log(Z(x)) \\
Z(x) = \sum_{r'=1}^{[V]} e^{U_{r'}(x)}
\]

\[
L = \sum_i [\log(P(x_i)) - \alpha(\log(Z(x_i)) - 0)^2] \\
= \sum_i [\log(P(x_i)) - \alpha \log^2(Z(x_i))]
\]

4.1 Pre-Computing the Hidden Layer