Some mathematics of Word2Vec algorithm and approximated Softmax

A/Prof Richard Yi Da Xu, Erica Huang
Yida.Xu@uts.edu.au, Wanming.Huang@student.uts.edu.au
richardxu.com

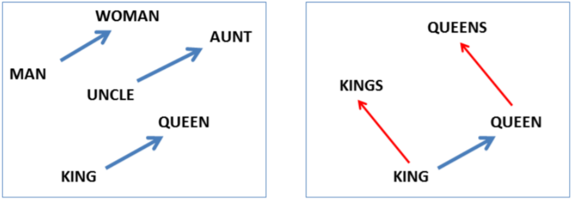
University of Technology Sydney (UTS)

June 10, 2018

word embedding

- words are symbols: one may **not** able to perform arithmetic operations on them.
- turning each word into a **vector**, e.g., “machine” $\rightarrow$ [2.4 - 1.2 1.9 . . .]
- can measure how similar or dissimilar between them
- can even perform ”arithmetic”:
- examples from the original paper:

$$vec(King) - vec(man) + vec(woman) = vec(Queen)$$



Mikolov et. al., (2013) “Linguistic Regularities in Continuous Space Word Representations”

one-hot encoding

- simple!

$$
\begin{array}{r}
\text{“}a\text{”} \\
\text{“}abbreviate\text{”} \\
\vdots \\
\text{“}zoology\text{”}
\end{array}
\begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
 & & & \\
0 & 0 & \cdots & 1
\end{bmatrix}
$$

- however, the structure is huge and sparse
- every pair of items are $\sqrt{2}$ apart.
- can we do better?

supervised learning between (target $\rightarrow$ context)

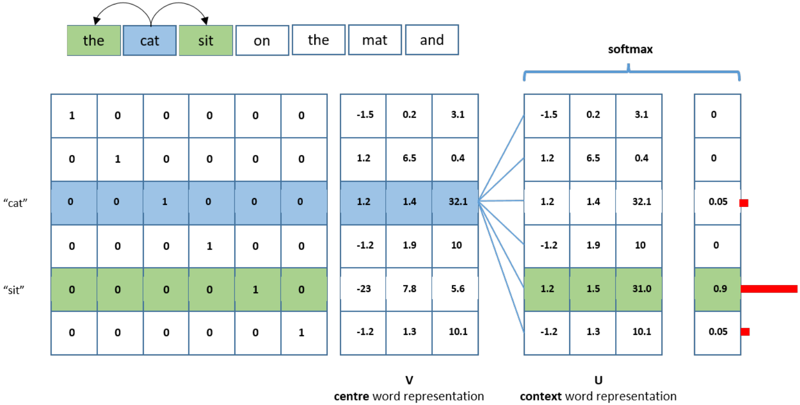
- Word2Vec algorithm tries to leverage (*"target", "context"*) relationships:
- offers two approaches, i.e., to maximize two different conditional densities:
  - skip-gram: $\Pr(\text{"}\textit{context}\text{"} \mid \text{"}\textit{target}\text{"})$
  - continuous bag of words (CBOW): $\Pr(\text{"}\textit{target}\text{"} \mid \text{"}\textit{context}\text{"})$
- uses supervised learning techniques: so we need to build ("input", "label") pairs
  1. pick window size (odd number)
  2. extract all tokens based on this chosen window size
  3. remove middle word in each window; this becomes your target word, rest are context
- btw, each word is an object class, a huge softmax!

Word2Vec: building training set

- for example, **Skip-gram**(window size 3)
- *"the cat sit on the mat"*
  1. "the", "cat", "sit", target: cat
  2. "cat", "sit", "on" , target: sit
  3. "sit", "on ", "the", target: on
  4. "on" , "the", "mat", target: the
- now we can perform **supervised learning** for (center, context):
  - ("cat", "the")
  - ("cat","sit")
  - ("sit","cat")
  - ("sit","on")
  - ("on","sit")
  - ("on","the")
  - ("the","on")
  - ("the", "mat")

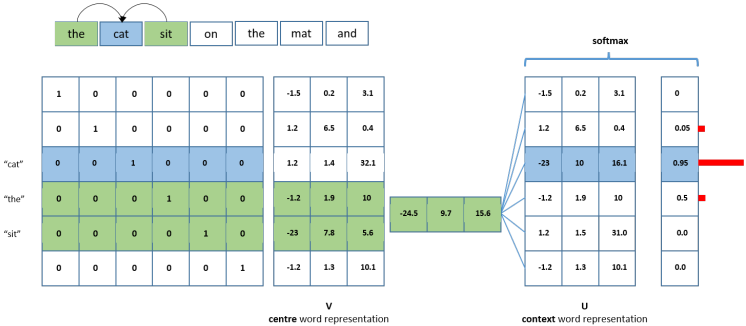
Simple Skip-gram example

- say vocabulary has 6 unique words in total
- "cat (3^{th} word)" and "sit (5^{th} word)" is a (center, context) pair
- for every word w, it has 2 representations $\mathbf{u}_w$ and $\mathbf{v}_w$, one for input and one for output
- predict context word given a center word $\Pr(o = \text{"}sit\text{"} \,|\, c = \text{"}cat\text{"})$



Simple CBoW example

- to predict center word given multiple context words:



- $\mathbf{v}_t$ is average of input (context) vectors
- the new objective is:

$$p(c|t) = \frac{\exp(\mathbf{u}_c^\top \mathbf{v}_t)}{\sum_{w'} \exp(\mathbf{u}_{w'}^\top \mathbf{v}_t)}$$

Skip-Gram objective function (1)

- predict context word given a center word $\Pr(o = \text{"}sit\text{"} \,|\, c = \text{"}cat\text{"})$

$$\Pr(o = \text{"}sit\text{"} \,|\, c = \text{"}cat\text{"}) = \frac{\exp(\mathbf{u}_{\text{"}sit\text{"}}^\top \mathbf{v}_{\text{"}cat\text{"}})}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_{\text{"}cat\text{"}})}$$

$$\implies \log(\Pr(o = |c)) = \log\left(\frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)}\right)$$

- we need to compute both $\frac{\partial \log(\Pr(o=|c))}{\partial \mathbf{v}_c}$ and $\frac{\partial \log(\Pr(o=|c))}{\partial \mathbf{u}_w}$, $\forall \mathbf{v}_c, \mathbf{u}_w \in \mathcal{V}$
- due to symmetry, looking at only one:

$$\begin{aligned}
\frac{\partial \log(\Pr(o = |c))}{\partial \mathbf{v}_c} &= \frac{\partial \mathbf{u}_o^\top \mathbf{v}_c}{\partial \mathbf{v}_c} - \frac{\partial \log\left(\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)\right)}{\partial \mathbf{v}_c} \\
&= \mathbf{u}_o - \left(\frac{\partial}{\partial \mathbf{v}_c} \log\left(\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)\right)\right) \\
&= \mathbf{u}_o - \left(\frac{1}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)} \frac{\partial}{\partial \mathbf{v}_c}\left(\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)\right)\right) \\
&= \mathbf{u}_o - \left(\frac{1}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)}\left(\sum_{w \in \mathcal{V}} \frac{\partial}{\partial \mathbf{v}_c} \exp(\mathbf{u}_w^\top \mathbf{v}_c)\right)\right) \\
&= \mathbf{u}_o - \frac{1}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)}\left(\sum_{w \in \mathcal{V}} \mathbf{u}_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)\right) \\
&= \mathbf{u}_o - \frac{\sum_{w \in \mathcal{V}} \mathbf{u}_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)}
\end{aligned}$$

Skip-Gram objective function (2)

- derivative:

$$
\begin{aligned}
\frac{\partial \log(\Pr(o = |c))}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{w \in \mathcal{V}} \mathbf{u}_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)} \\
&= \mathbf{u}_o - \sum_{w \in \mathcal{V}} \frac{\exp(\mathbf{u}_w^\top \mathbf{v}_c)}{\sum_{w \in \mathcal{V}} \exp(\mathbf{u}_w^\top \mathbf{v}_c)} \mathbf{u}_w \\
&= \mathbf{u}_o - \sum_{w \in \mathcal{V}} \Pr(w|c) \mathbf{u}_w \\
&= \mathbf{u}_o - \mathbb{E}_{w \sim \Pr(w|c)}[\mathbf{u}_w]
\end{aligned}
$$

- obviously, there are $|\mathcal{V}|$ is too big, making it too computationally expensive to compute the sum
- can we do better?

Negative sampling (1)

- negative sampling based on Skip-Gram model, it is optimizing **different objective**, let $\theta = [\mathbf{u}, \mathbf{v}]$:
- we let $\bar{w}$ to indicate **negative samples**, and come from a negative population $\bar{D}$

$$
\begin{aligned}
\theta &= \arg\max_{\theta} \prod_{(w,c)\in D} \Pr(D=1|w,c,\theta) \prod_{(\bar{w},c)\in \bar{D}} \Pr(D=0|\bar{w},c,\theta) \\
&= \arg\max_{\theta} \prod_{(w,c)\in D} \Pr(D=1|w,c,\theta) \prod_{(\bar{w},c)\in \bar{D}} (1-\Pr(D=1|\bar{w},c,\theta)) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \log\left(\Pr(D=1|w,c,\theta)\right) + \sum_{(\bar{w},c)\in \bar{D}} \log\left(1-\Pr(D=1|\bar{w},c,\theta)\right) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+\exp\left[-\mathbf{u}_w^\top \mathbf{v}_c\right]} + \sum_{(\bar{w},c)\in \bar{D}} \log\left(1 - \frac{1}{1+\exp\left[-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c\right]}\right) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \sigma(-\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c)\in \bar{D}} \log\left(\frac{1}{1+\exp\left[\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c\right]}\right) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c)\in \bar{D}} \log \sigma(-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c)
\end{aligned}
$$

Negative sampling (2)

- negative sampling based on Skip-Gram model, it is optimizing **different objective**, let $\theta = [\mathbf{u}, \mathbf{v}]$:

$$\theta = \arg\max_{\theta} \sum_{(w,c) \in D} \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c) \in \bar{D}} \log \sigma(-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c)$$

- it still has a huge sum term $\sum_{(\bar{w},c) \in \bar{D}}(.)$, so we change to:

$$\theta = \arg\max_{\theta} \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{\bar{w}=1}^{k} \mathbb{E}_{\bar{w} \sim P(w)} \log \sigma(-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c)$$

- sample a fraction of negative samples in second terms: $\{\bar{w}\}$ instead of going for $\forall(\bar{w} \neq w) \in \mathcal{V}$
- $\bar{w} \sim \Pr_{\bar{D}}(w)$, where $\Pr_{\bar{D}}(.)$ is probability of negative sample:

  can use Unigram Model raised to the power of $\frac{3}{4}$
- doing so, we can:

  increase probability of popular words marginally

  increase probability of rarer words dramatically

  making "rarer" words also have chance to be sampled
- in unigram model, probability of each word only depends on that word's own probability

Noise Contrastive Estimation (NCE) (1)

- let $u_\theta(w, c)$ be un-normalized score function, i.e., $u_\theta(w, c) = \exp(\mathbf{u}_w^\top \mathbf{v}_c)$

$$P_\theta(w|c) = \frac{u_\theta(w, c)}{\sum_{w' \in \mathcal{V}} u_\theta(w', c)} = \frac{u_\theta(w, c)}{Z_c}$$

- $\tilde{p}(w|c)$ and $\tilde{p}(c)$ are empirical distributions
  we **know** them from data, so we can sample (w, c) from it!
- a “noise” distribution $q(w)$ is used - uniform or uniform unigram
  we also **know** them, again, we can sample $\bar{w} \sim q(.)$
- **task** is to use sample from both distributions, then to assist us find θ making $P_\theta(w|c)$ to approximate empirical distribution as closely as possible (by minimal cross entropy)

- **training data generation**: $(w, c, D) \sim \mathcal{D}$
- of course, to utilize $\tilde{p}(w|c)$, $\tilde{p}(c)$ and $q(w)$, which we already have knowledge of:
  1. sample a $c \sim \tilde{p}(c)$, $w \sim \tilde{p}(w|c)$ and label it as $D = 1$
  2. k "noise" samples from $q(.)$, and label it as $D = 0$
- NCE transforms:
  "problem of model estimation" (computationally expensive) to
  "problem of estimating parameters of probabilistic binary classifier uses **same parameters** *to distinguish between samples"*:(computationally acceptable)
  - from empirical distribution
  - from noise distribution

Noise Contrastive Estimation (NCE) (3)

- let $u_\theta(w, c)$ be un-normalized score function, i.e., $u_\theta(w, c) = \exp(\mathbf{u}_w^\top \mathbf{v}_c)$

$$
\begin{aligned}
P(D = 0|c, w) &= \frac{P(D = 0, w|c)}{P(w|c)} = \frac{p(w|D = 0, c)P(D = 0)}{\sum_{d \in \{0,1\}} p(w|D = d, c)P(D = d)} \\
&= \frac{q(w) \times \frac{k}{1+k}}{\tilde{P}(w|c) \times \frac{1}{k+1} + q(w) \times \frac{k}{1+k}} \\
&= \frac{kq(w)}{\tilde{P}(w|c) + kq(w)} \\
P(D = 1|c, w) &= 1 - P(D = 0|c, w) \\
&= \frac{\tilde{P}(w|c)}{\tilde{P}(w|c) + kq(w)}
\end{aligned}
$$

Noise Contrastive Estimation (NCE) (4)

- NCE replaces empirical distribution $\tilde{p}(w|c)$ with model distribution $p_\theta(w|c)$

$$P(D=0|c,w) = \frac{kq(w)}{\tilde{P}(w|c) + kq(w)} = \frac{kq(w)}{\frac{u_\theta(w|c)}{Z_c} + kq(w)}$$

$$P(D=1|c,w) = \frac{\tilde{P}(w|c)}{\tilde{P}(w|c) + kq(w)} = \frac{\frac{u_\theta(w|c)}{Z_c}}{\frac{u_\theta(w|c)}{Z_c} + kq(w)}$$

- θ is then chosen to maximize likelihood of proxy corpus created from **training data generation**:

$$\mathcal{L}^{\text{NCE}} = \log p(D=1|c,w) + k \sum_{(w,c)\in\mathcal{D}} \mathbb{E}_{w' \sim q} \log p(D=0|c,w')$$

- for neural networks: Z_c can also be trained or set to some fixed number, e.g., $Z_c = 1$
- **negative sampling** is its special case $k = |\mathcal{V}|$ and $q(.)$ is uniform, and $Z_c = 1$:

$$P(D=0|c,w) = \frac{|\mathcal{V}|\frac{1}{|\mathcal{V}|}}{u_\theta(w|c) + |\mathcal{V}|\frac{1}{|\mathcal{V}|}} = \frac{1}{u_\theta(w|c)+1}$$

$$P(D=1|c,w) = \frac{u_\theta(w|c)}{u_\theta(w|c) + |\mathcal{V}|\frac{1}{|\mathcal{V}|}} = \frac{u_\theta(w|c)}{u_\theta(w|c)+1}$$

Self Normalization

- in previous slide, we want to normalize, s.t., $Z_c = 1$
- start with $u_\theta(w, c) = \exp(\mathbf{u}_w^\top \mathbf{v}_c)$:

$$
\begin{aligned}
P_\theta(w|c) &= \prod_w \frac{\exp(\mathbf{u}_w^\top \mathbf{v}_c)}{Z_c} \\
\implies J_\theta = -\prod_w \log(P_\theta(w|c) &= -\sum_w \log\left(\frac{\exp(\mathbf{u}_w^\top \mathbf{v}_c)}{Z_c}\right) \\
&= -\sum_w \mathbf{u}_w^\top \mathbf{v}_c - \log\left(Z_c\right)
\end{aligned}
$$

- to constrain model and sets $Z(c) = 1 \implies \log Z(c) = 0$:

$$
\begin{aligned}
J_\theta &= -\sum_w \mathbf{u}_w^\top \mathbf{v}_c + \log Z(c) - \alpha\,(\log(Z(c)) - 0)^2 \\
&= -\sum_w \mathbf{u}_w^\top \mathbf{v}_c + \log Z(c) - \alpha \log^2 Z(c)
\end{aligned}
$$

A library created by Facebook research team for

1. efficient learning of word representations(Enriching Word Vectors with Subword Information)
2. sentence classification(Bag of Tricks for Efficient Text Classification)

FastText

- So how is it different from Word2Vec?
- Instead of words, we now have **ngrams** of subwords, what is its **advantage**?
  1. Helpful for finding representations for rare words
  2. Give vector representations for words not present in dictionary
- for example, $n = 3$, i.e., 3-grams:
  - **word**: *"where"*,
  - **sub-words**: *"wh", "whe", "her", "ere", "re"*
- we then represent a word by the **sum of the vector representations** of all its n-grams
- to compute an un-normalised score with center word $\mathbf{v}_c$, given a word w, g_w is the set of n-grams appearing in w, z_g is the representation to each individual n-gram

$$u(w, c) = \exp\left[\sum_{g \in g_w} z_g^\top \mathbf{v}_c\right]$$

- **co-occurrence probabilities** are useful
- GloVe learns word vectors through *word co-occurrences*
- co-occurrence matrix P where P_{ij} is how often word i appears in the context of word j
- Fast training and scalable to huge corpora
- loss function:

$$\theta^* = \arg\min_{\theta} \left(J(\theta) \equiv \frac{1}{2} \sum_{\mathbf{u}_i \mathbf{v}_j \in \mathcal{V}} f(P_{ij})(\mathbf{u}_i^\top \mathbf{v}_j - \log P_{ij})^2 \right)$$

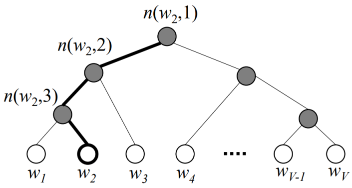
- it tries to minimize difference:

$$\left(\mathbf{u}_i^\top \mathbf{v}_j - \log P_{ij} \right)$$

- more frequently two words appear together,more similar their vector representation should be
- $f(.)$ is weighting function to "prevent" certain scenarios, for example:

$$P_{ij} = 0 \implies \log P_{ij} = -\infty \implies f(0) = 0$$

Hierarchical Softmax (1)

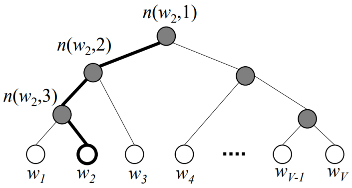


Xin Rong, word2vec Parameter Learning Explained

- **super advantage**: $\Pr(w|c)$ is already a probability by multiplying all probabilities of path, no need to normalize!

- each word w_i has a unique (pre-defined) path (not a random path!), which performs a left or right turn from nodes: $n(w_i, 1), n(w_i, 2), n(w_i, 3), \ldots$
- the route is defined in such a way that, each child node is from a (LEFT/RIGHT) "channel" of its parent: i.e., $n(w, j+1) = \text{ch}(n(w, j))$
- for example:
  1. $n(w_2, 2) = \text{LEFT}\,(n(w_2, 1))$
  2. $n(w_2, 3) = \text{LEFT}\,(n(w_2, 2))$
  3. $\underbrace{n(w_2, 4)}_{w_2} = \text{RIGHT}(n(w_2, 3))$
- there are V words in leaf (white node)
- there are $V - 1$ inner (non-leaf) nodes (grey node) each associate with a value of $\mathbf{v}$ which is shared among all words going through this node

Hierarchical Softmax (2)



Xin Rong, word2vec Parameter Learning Explained

- **super advantage**: $\Pr(w|c)$ is already a probability by multiplying all probabilities of path, no need to normalize!

we define: $\xi[.] = \begin{cases} 1 : & \text{true} \\ -1 : & \text{false} \end{cases}$

$$\Pr(w|c) = \prod_{j=1}^{L(w)-1} \sigma\Big(\underbrace{\xi\left[n(w, j+1) = \text{ch}(n(w,j))\right]}_{\text{control its sign}} \mathbf{v}_{n(w,j)}^{\top} \mathbf{u}_c \Big)$$

- looking at $\Pr(w_2|c)$ and $\Pr(w_3|c)$:
- $n(w_2, 1) = n(w_3, 1)$ in fact $\{n(w_i, 1)\}_{i=1}^{|\mathcal{V}|}$ all equal
- $n(w_2, 2) = n(w_3, 2)$

$$\begin{aligned}
\Pr(w_2|c) &= p(n(w_2, 1), \text{LEFT})\, p(n(w_2, 2), \text{LEFT})\, p(n(w_2, 3), \text{RIGHT}) \\
&= \sigma\left(\mathbf{v}_{n(w_2,1)}^{\top} \mathbf{u}_c\right) \sigma\left(\mathbf{v}_{n(w_2,2)}^{\top} \mathbf{u}_c\right) \sigma\left(-\mathbf{v}_{n(w_2,3)}^{\top} \mathbf{u}_c\right) \\
\Pr(w_3|c) &= p(n(w_3, 1), \text{LEFT})\, p(n(w_3, 2), \text{RIGHT}) \\
&= \sigma\left(\mathbf{v}_{n(w_3,1)}^{\top} \mathbf{u}_c\right) \sigma\left(-\mathbf{v}_{n(w_3,2)}^{\top} \mathbf{u}_c\right)
\end{aligned}$$

- **consideration 1** $\exp(\mathbf{x}^T \boldsymbol{\theta}_i)$ can become very large:

$$
\begin{aligned}
\pi_i &= \frac{\exp(\mathbf{x}^T \boldsymbol{\theta}_i)}{\sum_{l=1}^{3} \exp(\mathbf{x}^T \boldsymbol{\theta}_l)} \\
&= \frac{\left(\exp(\mathbf{x}^T \boldsymbol{\theta}_i)\right) / C}{\left(\sum_{l=1}^{3} \exp(\mathbf{x}^T \boldsymbol{\theta}_l)\right) / C} = \frac{\exp(\mathbf{x}^T \boldsymbol{\theta}_i - C)}{\sum_{l=1}^{3} \exp(\mathbf{x}^T \boldsymbol{\theta}_l - C)} \\
&= \frac{\exp\left(\mathbf{x}^T \boldsymbol{\theta}_i - \max\left(\{\exp(\mathbf{x}^T \boldsymbol{\theta}_l)\}\right)\right)}{\sum_{l=1}^{3} \exp\left(\mathbf{x}^T \boldsymbol{\theta}_l - \max\left(\{\exp(\mathbf{x}^T \boldsymbol{\theta}_l)\}\right)\right)}
\end{aligned}
$$

- **consideration 2** $\arg\max$ operation, can be done without $\exp$, i.e.,

$$
\arg\max_{i \in \{1,\dots,k\}} (\pi_1, \dots \pi_k) \equiv \arg\max_{i \in \{1,\dots,k\}} (\mathbf{x}^\top \theta_1, \dots, \mathbf{x}^\top \theta_k)
$$

Gumbel-max trick and Softmax (1)

- pdf of Gumbel with **unit scale** and location parameter μ:

$$\text{gumbel}(Z = z;\ \mu) = \exp\left[-(z-\mu) - \exp\{-(z-\mu)\}\right]$$

- CDF of Gumbel:

$$\text{Gumbel}(Z \leq z;\ \mu) = \exp\left[- \exp\{-(z-\mu)\}\right]$$

- given a set of Gumbel random variables $\{Z_i\}$, each having own location parameters $\{\mu_i\}$, probability of all other $Z_{i\neq k}$ are less than a particular value of z_k:

$$p\left(\max\{Z_{i\neq k}\} = z_k\right) = \prod_{i\neq k} \exp\left[- \exp\{-(z_k - \mu_i)\}\right]$$

- obviously, $Z_k \sim \text{gumbel}(Z_k = z_k; \mu_k)$:

$$\begin{aligned}
\Pr(k \text{ is largest} \mid \{\mu_i\}) &= \int \exp\left\{-(z_k-\mu_k) - \exp\{-(z_k-\mu_k)\}\right\} \prod_{i\neq k} \exp\left\{- \exp\{-(z_k-\mu_i)\}\right\} \mathrm{d}z_k \\
&= \int \exp\left[- z_k + \mu_k - \exp\{-(z_k-\mu_k)\}\right] \exp\left[- \sum_{i\neq k} \exp\{-(z_k-\mu_i)\}\right] \mathrm{d}z_k \\
&= \int \exp\left[- z_k + \mu_k - \exp\{-(z_k-\mu_k)\} - \sum_{i\neq k} \exp\{-(z_k-\mu_i)\}\right] \mathrm{d}z_k \\
&= \int \exp\left[- z_k + \mu_k - \sum_{i} \exp\{-(z_k-\mu_i)\}\right] \mathrm{d}z_k \\
&= \int \exp\left[- z_k + \mu_k - \sum_{i} \exp\{-z_k+\mu_i)\}\right] \mathrm{d}z_k \\
&= \int \exp\left[- z_k + \mu_k - \exp\{-z_k\} \sum_{i} \exp\{\mu_i)\}\right] \mathrm{d}z_k
\end{aligned}$$

Gumbel-max trick and Softmax (2)

- keep on going:

$$
\begin{aligned}
\Pr(k \text{ is largest} \mid \{\mu_i\}) &= \int \exp\left[-z_k + \mu_k - \exp\{-z_k\} \sum_i \exp\{\mu_i)\} \right] \mathrm{d}z_k \\
&= \exp^{\mu_k} \int \exp\left[-z_k - \exp\{-z_k\} C \right] \mathrm{d}z_k \\
&= \exp^{\mu_k} \left[\frac{\exp(-C\exp(-z_k))}{C} \Big|_{z_k=-\infty}^{\infty} \right] \\
&= \exp^{\mu_k} \left[\frac{1}{C} - 0 \right] = \frac{\exp^{\mu_k}}{\sum_i \exp\{\mu_i)\}}
\end{aligned}
$$

Let $\mu_i \equiv \mathbf{x}^\top \theta_i$

- moral of the story is, if one is to sample the largest element from **softmax**:

$$
\begin{aligned}
k &= \underset{i \in \{1,\dots,K\}}{\arg\max} \sim \left\{ \frac{\exp(\mathbf{x}^\top \theta_1)}{\sum_i \exp(\mathbf{x}^\top \theta_i))}, \dots, \frac{\exp(\mathbf{x}^\top \theta_K)}{\sum_i \exp(\mathbf{x}^\top \theta_i)} \right\} \\
&= \underset{i \in \{1,\dots,K\}}{\arg\max} (\{G_1, \dots, G_K\}) \qquad \left\{ G_i \sim \underbrace{\text{gumbel}(z;\ \mu_i) \equiv \exp\left[-(z-\mu_i) - \exp\{-(z-\mu_i)\} \right]} \right\} \\
&= \underset{i \in \{1,\dots,K\}}{\arg\max} (\{G_1, \dots, G_K\}) \qquad \left\{ G_i = \mu_i + \mathcal{G} \qquad \mathcal{G} \overset{\text{iid}}{\sim} \underbrace{\text{gumbel}(z;\ 0) \equiv \exp\left[-(z) - \exp\{-(z)\} \right]} \right\}
\end{aligned}
$$