Breaking the Softmax Bottleneck via Monotonic Functions

Octavian Ganea, Sylvain Gelly, Gary Bécigneul, Aliaksei Severyn
Softmax Layer (for Language Models)

- Natural language as conditional distributions
- Parametric distributions & softmax:

\[
P_\theta(x|c) = \frac{\exp h_c^T w_x}{\sum_{x'} \exp h_c^T w_{x'}} \approx P^*(x|c)
\]
Softmax Layer (for Language Models)

- Natural language as conditional distributions

- Parametric distributions & softmax:
  \[ P_\theta(x|c) = \frac{\exp h_c^\top w_x}{\sum_{x'} \exp h_c^\top w_{x'}} \approx P^*(x|c) \]

- Challenge: Can we always find \( \theta \) s.t. for all \( c \):
  \[ P_\theta(X|c) = P^*(X|c) \]
Softmax Layer (for Language Models)

- Natural language as conditional distributions

- Parametric distributions & softmax:
  \[
  P_{\theta}(x|c) = \frac{\exp h_c^T w_x}{\sum_{x'} \exp h_c^T w_{x'}} \approx P^*(x|c)
  \]

- Challenge: Can we always find \( \theta \) s.t. for all \( c \):
  \[
  P_{\theta}(X|c) = P^*(X|c)
  \]

**No**, when embedding size < label cardinality (vocab size)!
What is the Softmax Bottleneck (Yang et al, ‘18)?

- **log-P matrix**: \( A_P = \begin{bmatrix}
\log P(x_1|c_1) & \log P(x_2|c_1) & \cdots & \log P(x_M|c_1) \\
\log P(x_1|c_2) & \log P(x_2|c_2) & \cdots & \log P(x_M|c_2) \\
\vdots & \vdots & \ddots & \vdots \\
\log P(x_1|c_N) & \log P(x_2|c_N) & \cdots & \log P(x_M|c_N)
\end{bmatrix} \in \mathbb{R}^{N \times M} \)

Label cardinality = Vocabulary size
What is the Softmax Bottleneck (Yang et al, ‘18)?

- **log-P matrix:** \( A_P = \begin{bmatrix} \log P(x_1 | c_1) & \log P(x_2 | c_1) & \cdots & \log P(x_M | c_1) \\ \log P(x_1 | c_2) & \log P(x_2 | c_2) & \cdots & \log P(x_M | c_2) \\ \vdots & \vdots & \ddots & \vdots \\ \log P(x_1 | c_N) & \log P(x_2 | c_N) & \cdots & \log P(x_M | c_N) \end{bmatrix} \in \mathbb{R}^{N \times M} \)

- Then: \( \text{rank}(A_{\Theta}) \leq d + 1 \)

Number of labels = Vocabulary size
What is the Softmax Bottleneck (Yang et al, ‘18)?

- **log-P matrix**: 
  \[ A_P = \begin{bmatrix} \log P(x_1|c_1) & \log P(x_2|c_1) & \ldots & \log P(x_d|c_1) \\ \log P(x_1|c_2) & \log P(x_2|c_2) & \ldots & \log P(x_d|c_2) \\ \vdots & \vdots & \ddots & \vdots \\ \log P(x_1|c_M) & \log P(x_2|c_M) & \ldots & \log P(x_d|c_M) \end{bmatrix} \]

- Then:

But \( A_{P^*} \) is likely full-rank, so \( A_{P^*} \neq A_{P^\emptyset} \) when \( d \ll \min(M, N) \).
Breaking the Softmax Bottleneck [1]

- MoS [1]: Mixture of $K$ Softmaxes
Breaking the Softmax Bottleneck [1]

- MoS [1]: Mixture of $K$ Softmaxes
- Improves perplexity

[1] Breaking the Softmax Bottleneck: A High-Rank RNN Language Model, Yang et al., ICLR 2018
Breaking the Softmax Bottleneck [1]

- MoS [1]: Mixture of $K$ Softmaxes
- Improves perplexity
- Slower than vanilla softmax: 2 - 6.4x
- GPU Memory: $M \times N \times K$ tensor

---

[1] Breaking the Softmax Bottleneck: A High-Rank RNN Language Model, Yang et al., ICLR 2018
Breaking the Softmax Bottleneck [2]

- Sig-Softmax [2]:

\[
\text{softmax}(2y - \log(1 + \exp(y))
\]
Breaking the Softmax Bottleneck [2]

- Sig-Softmax [2]:

\[
\text{softmax}(2y - \log(1 + \exp(y))
\]

- Small improvement over vanilla Softmax

[2] Sigsoftmax: Reanalysis of the Softmax Bottleneck, S. Kanai et al., NIPS 2018
Breaking the Softmax Bottleneck [2]

- Sig-Softmax [2]:

\[
\text{softmax}(2y - \log(1 + e^x))
\]

Can we learn the best non-linearity to deform the logits?

[2] Sigsoftmax: Reanalysis of the Softmax Bottleneck, S. Kanai et al., NIPS 2018
Can we do better?

- Our idea - learn a pointwise monotonic function on top of logits:

\[ p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e.} \ \text{softmax}(f(y)) \]
Can we do better?

- Our idea - learn a pointwise monotonic function on top of logits:

\[ p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. } \text{softmax}(f(y)) \]

\[ f : \mathbb{R} \rightarrow \mathbb{R} \] should be:
Can we do better?

- Our idea - learn a pointwise monotonic function on top of logits:

\[ p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. } \text{softmax}(f(y)) \]

\[ f : \mathbb{R} \rightarrow \mathbb{R} \text{ should be:} \]

1. With **unbounded image set** -- to model sparse distributions
Can we do better?

- Our idea - learn a pointwise monotonic function on top of logits:

\[
p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. softmax}(f(y))
\]

\[f : \mathbb{R} \rightarrow \mathbb{R}\] should be:

1. With **unbounded image set** -- to model sparse distributions
2. **Continuous** and **(piecewise) differentiable** -- for backprop
Can we do better?

- Our idea - learn a pointwise monotonic function on top of logits:

\[
p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. softmax}(f(y))
\]

\(f : \mathbb{R} \rightarrow \mathbb{R}\) should be:

1. With **unbounded image set** -- to model sparse distributions
2. **Continuous** and (piecewise) **differentiable** -- for backprop
3. **Non-linear** -- to break the softmax bottleneck
Can we do better?

- Our idea - learn a pointwise monotonic function on top of logits:
  
  \[ p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. softmax}(f(y)) \]

  \( f : \mathbb{R} \rightarrow \mathbb{R} \) should be:

1. With **unbounded image set** -- to model sparse distributions
2. **Continuous** and (piecewise) differentiable -- for backprop
3. **Non-linear** -- to break the softmax bottleneck
4. **Monotonic** -- to preserve the ranking of logits
Can we do better?

• Our idea - learn a pointwise monotonic function on top of logits:

\[ p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. softmax}(f(y)) \]

\( f : \mathbb{R} \rightarrow \mathbb{R} \) should be:

1. With **unbounded image set** -- to model sparse distributions
2. **Continuous** and (piecewise) **differentiable** -- for backprop
3. **Non-linear** -- to break the softmax bottleneck
4. **Monotonic** -- to preserve the ranking of logits
5. **Fast** and **memory efficient** -- comparable w/ vanilla softmax
Can we do better?

Our idea - learn a pointwise monotonic function on top of logits:

\[ p(y_i) = \frac{\exp(f(y_i))}{\sum_j \exp(f(y_j))}, \text{ i.e. } \text{softmax}(f(y)) \]

\[ f : \mathbb{R} \rightarrow \mathbb{R} \] should be:

1. With **unbounded image set** -- to model sparse distributions
2. **Continuous** and (piecewise) **differentiable** -- for backprop
3. **Non-linear** -- to break the softmax bottleneck
4. **Monotonic** -- to preserve the ranking of logits
5. **Fast** and **memory efficient** -- comparable w/ vanilla softmax

Theorem: these properties are not restrictive in terms of rank deficiency
Learnable parametric monotonic real functions

- A neural network with 1 hidden layer and positive (constrained) weights [3]
  \[ f(x) = \sum_{i=1}^{K} v_i \sigma(u_i x + b_i) + b, \text{ s.t. } v_i, u_i \geq 0 \]

- Universal approximator for all monotonic functions (when K is large enough !)

Synthetic Experiment

- **Goal**: separate softmax bottleneck from context embedding bottleneck
Synthetic Experiment

- **Goal**: separate softmax bottleneck from context embedding bottleneck
- Sample N different categorical distributions with M outcomes:
  \[ P^*(\cdot | c_j) \sim \text{Dir}(\alpha), \text{ for } j = 1, \ldots, N \]
Synthetic Experiment

- **Goal**: separate softmax bottleneck from context embedding bottleneck
- Sample N different categorical distributions with M outcomes:
  \[
P^*(\cdot | c_j) \sim \text{Dir}(\alpha), \text{ for } j = 1, \ldots, N
\]
- **Goal**: 
  \[
P_\theta(x|c) = \frac{\exp h_c^T w_x}{\sum_{x'} \exp h_c^T w_{x'}} \approx P^*(x|c)
\]
Synthetic Experiment

● **Goal**: separate softmax bottleneck from context embedding bottleneck

● Sample N different categorical distributions with M outcomes:

\[ P^*(\cdot | c_j) \sim \text{Dir}(\alpha), \text{ for } j = 1, \ldots, N \]

● **Goal**:

\[ P_\theta(x|c) = \frac{\exp h_c^T w_x}{\sum_{x'} \exp h_c^T w_{x'}} \approx P^*(x|c) \]

● Independent context embeddings; shared word embeddings
Synthetic Experiments - Mode Matching ($\alpha=0.01$)

- Percentage of contexts $c$ for which $\arg\max_x P^*(x|c) = \arg\max_x P_\Theta(x|c)$

- **Vanilla Softmax**
  - Heatmap showing the percentage of contexts $c$ for which $\arg\max_x P^*(x|c) = \arg\max_x P_\Theta(x|c)$.

- **Monotonic fn (K=100)**
  - Heatmap showing the percentage of contexts $c$ for which $\arg\max_x P^*(x|c) = \arg\max_x P_\Theta(x|c)$.

- **Ratio 2nd / 1st**
  - Heatmap showing the ratio of the second to the first percentage for contexts $c$.
Synthetic Experiments - Mode Matching \((\alpha=0.01)\)

- Percentage of contexts \(c\) for which \(\text{argmax}_x P^*(x|c) = \text{argmax}_x P_\Theta(x|c)\)

- Similar results for cross-entropy and other values of \(\alpha\)
Piecewise Linear Increasing Functions (PLIF)

- NN w/ 1 hidden layer ⇒ **memory hungry**:
  - Tensor of size $N \times M \times K$ on GPU, where $K \geq 1000$
Piecewise Linear Increasing Functions (PLIF)

- NN w/ 1 hidden layer ⇒ memory hungry:
  - Tensor of size $N \times M \times K$ on GPU, where $K \geq 1000$

- PLIF:
Piecewise Linear Increasing Functions (PLIF)

- NN w/ 1 hidden layer ⇒ memory hungry:
  - Tensor of size $N \times M \times K$ on GPU, where $K \geq 1000$

- PLIF:
  - Forward & backward passes: just a lookup in two $K$ dim vectors
  - Memory and running time very efficient (comparable with Vanilla Softmax)
## Language Modeling Results

<table>
<thead>
<tr>
<th></th>
<th>PENN TREEBANK</th>
<th>WikiText-2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#PARAM</td>
<td>VALID ppl</td>
<td>TEST ppl</td>
<td>#SEC/EP</td>
</tr>
<tr>
<td><strong>LINEAR-SOFTMAX</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ AWD-LSTM, w/o finetune (Merity et al., 2017)</td>
<td>24.2M</td>
<td>60.83</td>
<td>58.37</td>
<td>~60</td>
</tr>
<tr>
<td><strong>Ours LMS-PLIF, 10^5 knots</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ AWD-LSTM, w/o finetune</td>
<td>24.4M</td>
<td>59.45</td>
<td>57.25</td>
<td>~70</td>
</tr>
<tr>
<td><strong>MoS, K = 15</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ AWD-LSTM, w/o finetune (Yang et al., 2017)</td>
<td>26.6M</td>
<td>58.58</td>
<td>56.43</td>
<td>~150</td>
</tr>
<tr>
<td><strong>MoS(15 comp) + our PLIF (10^6 knots)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ AWD-LSTM, w/o finetune</td>
<td>28.6M</td>
<td>58.20</td>
<td>56.02</td>
<td>~220</td>
</tr>
</tbody>
</table>

**GPU Memory:** N x M x K

**GPU Memory:** N x M
Thank you!

Poster #23