Predefined Sparseness in Recurrent Sequence Models

Thomas Demeester, Johannes Deleu, Frederic Godin, Chris Develder

thomas.demeester@ugent.be

November 1st, 2018
CoNLL, Brussels, Belgium
Sparse Neural Networks

dense model

sparsify

sparse model

‘sparser’ model
(lower memory footprint)
Sparsifying by weight pruning

Highly sparse with accuracy close to dense models [1]

Large sparse networks can be better than small dense models [2]

BUT THEN: large dense network needed during training!

GOAL: models that are sparse from the start?

“predefined sparseness”


Predefined sparseness for RNNs

Any recurrent cell (RNN, LSTM, GRU...): 2 types of matrices

- **hidden-to-hidden**: $W_{hh}$
- **input-to-hidden**: $W_{hi}$

**proposed sparse model**

- **block-diagonal** (density $1/N$)
- **(density $\gamma$)**
Predefined sparseness for RNNs

With sparse $W_{hh}$ and $W_{hi}$

- strongly reduced number of hidden-to-hidden interactions (cfr. weight dropping in $W_{hh}$ [5])
- not all hidden dimensions have access to each input dimension.

why this particular choice?

Predefined sparseness for RNNs

Consider vanilla RNN
Predefined sparseness for RNNs

Consider vanilla RNN

\[ h_t = \tanh(W_{hh} h_{t-1} + W_{hi} x_t) \]
Predefined sparseness for RNNs

Consider vanilla RNN - made sparse
Predefined sparseness for RNNs

Consider vanilla RNN - made sparse

Resulting RNN equivalent to N smaller dense RNNs in parallel

- only possible with output divided into disjoint segments
- but input can be (partly) shared between components
- holds for vanilla RNN, LSTM, GRU,...
- allows standard tools (CuDNN) / parallel processing
Language modeling with sparse LSTM

- baseline: AWD-LSTM model [5] with 3-layer stacked LSTM

- sparse counterpart:
  - middle LSTM hidden size x 1.5 (from 1150 to 1725)
  - sparse; same number of parameters
  - same regularization settings

Language modeling with sparse LSTM

- first train run (500 epochs)

<table>
<thead>
<tr>
<th>Model</th>
<th>Penn Treebank test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>reported [5]</td>
<td>58.8</td>
</tr>
<tr>
<td>baseline</td>
<td>58.8 ± 0.3</td>
</tr>
<tr>
<td>sparse LSTM</td>
<td>57.9 ± 0.3</td>
</tr>
</tbody>
</table>

- train further (‘finetune’) : sparse model overfits

Language modeling with sparse LSTM

- hypothesis:
  
  the regularization effect of 
  a priori limiting interactions between dimensions 
  does not compensate for increased expressiveness 
  due to larger hidden state size

- supported by additional experiment "learning to recite" (see paper 😊)
Predefined sparseness in word embeddings

- **Goal:**
  decide upfront which entries in embedding matrix $E \in \mathbb{R}^{v \times k}$ are 0.

- Word occurrence frequencies have Zipfian nature

source: Manning, Schütze, Raghavan, "Introduction to Information Retrieval", Cambridge UP, 2009
Predefined sparseness in word embeddings

- Goal:
  decide upfront which entries in embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times k}$ are 0.

- Word occurrence frequencies have Zipfian nature

representing long tail of rare terms with short embeddings would greatly reduce memory requirements

source: Manning, Schütze, Raghavan, "Introduction to Information Retrieval", Cambridge UP, 2009
Predefined sparseness in word embeddings

Predefined sparse embedding matrix $E$?

$k$ embedding dimensions

Vocabulary $V$  

Trainable parameters $= kV$
Predefined sparseness in word embeddings

Predefined sparse embedding matrix $E$?

$k$ embedding dimensions

sorted vocabulary $V$

rare terms

common terms

trainable parameters $= kV \delta_E$

rare term embedding

frequent term embedding
Predefined sparseness in word embeddings

Predefined sparse embedding matrix $E$?

Trainable parameters $= kV\delta_E$

Sparse embedding space:
- ‘first’ dimensions model many rare terms
- remaining dimensions model few frequent terms
Predefined sparseness in word embeddings

- Experimental setup:
  - POS tagging on Penn Treebank
  - very small model (else too easy!)
  - 20-D word embeddings (876k params)
  - BiLSTM state size 10+10 (3k params)
Predefined sparseness in word embeddings

● Experimental setup:
  ○ POS tagging on Penn Treebank
  ○ very small model (else too easy!)
  ○ 20-D word embeddings (876k params)
  ○ BiLSTM state size 10+10 (3k params)

● Embedding matrix

![Diagram showing sparseness in word embeddings]
Predefined sparseness in word embeddings

- Experimental setup:
  - POS tagging on Penn Treebank
  - very small model (else too easy!)
  - 20-D word embeddings (876k params)
  - BiLSTM state size 10+10 (3k params)

- Embedding matrix
Predefined sparseness in word embeddings

- Resulting POS tag accuracy

![Graph showing test accuracy vs (average) embedding size]

- Dense
- Sparse $\delta_E = 0.5$

Same number of trainable params
Conclusions

● Simple ideas for **predefined sparseness in RNNs and embedding layers**

● **Predefined Sparseness** has potential in NLP

● **Further investigation needed** (for very large representation sizes for large vocabularies, etc.)

● Need some “**predefined sparseness**” code?

[https://github.com/tdmeeste/SparseSeqModels](https://github.com/tdmeeste/SparseSeqModels)
Thank you!
Language modeling with sparse LSTM

- **baseline:**
  - AWD-LSTM model [5]
  - 400D word embeddings, 10k words; 4M params
  - 3-layer stacked LSTM (dimensions 400 - 1150 - 400); 20M params

- **sparse counterpart:**
  - similar 3-layer LSTM; 20M params
  - but: middle LSTM scaled from 1150 to 1725 units (factor 1.5)
  - sparse: to retain same number of parameters
  - no tuning (exactly same regularization parameters)

Inspiration from literature

“application of sparse coding in language processing is far from extensive, when compared to speech processing” [3]

Need for sparse models in NLP!

“natural language is high-rank” [4]

How to train large sparse representations despite memory constraints?


Language modeling with sparse LSTM

- first train run (500 epochs)

<table>
<thead>
<tr>
<th>Model</th>
<th>Penn Treebank test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>reported [5]</td>
<td>58.8</td>
</tr>
<tr>
<td>baseline</td>
<td>58.8 ± 0.3</td>
</tr>
<tr>
<td>sparse LSTM</td>
<td>57.9 ± 0.3</td>
</tr>
</tbody>
</table>

- train further (‘finetune’) : sparse model overfits

Language modeling with sparse LSTM

- train again ("finetune step" [5])

<table>
<thead>
<tr>
<th>Model</th>
<th>Penn Treebank test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>reported [5]</td>
<td>57.3</td>
</tr>
<tr>
<td>baseline</td>
<td>56.6 ± 0.2</td>
</tr>
<tr>
<td>sparse LSTM</td>
<td>57.0 ± 0.2</td>
</tr>
</tbody>
</table>
