Three New Models for Statistical Language Modelling

Andriy Mnih and Geoffrey Hinton

Machine Learning Group
University of Toronto
Statistical language modelling

- Goal: Model the joint distribution of words in a sentence.

- Most statistical language models are based on the Markov assumption: the distribution of the next word depends on only $n$ words that immediately precede it.

- $N$-gram models are the most widely used statistical language models.
  - Conditional probability tables ($P(w_N|w_{1:N-1})$) estimated by counting $n$-tuples of words and smoothing the estimates.
  - Curse of dimensionality: lots of data is needed if $n$ is large.
Conditional Restricted Boltzmann Machine for language modelling

- We propose using Restricted Boltzmann Machines for modelling the distribution of the next word.
- An RBM is an undirected graphical model with fast exact inference and efficient approximate learning.
  - Two types of variables / units: visible and hidden
  - Bipartite structure: direct interactions are allowed only between units of different types.
- An RBM is typically defined using an energy function that assigns an energy value to every joint setting of the visible and hidden units.
  - Probabilities are obtained by exponentiating negative energies and normalizing.
Conditional RBM for language modelling

• We model words using multinomial variables that can take on as many values as there are words ($D$).
  - Each word is encoded as a $D$-bit vector using 1-of-$D$ encoding.

• The energy function for an RBM with $N$ input words and $M$ binary hidden units can be written as

$$E(w_{1:N}, h) = - \sum w_i^T J_i h$$

where each $J_i$ is a $ND \times M$ matrix.

• This parameterization can have too many parameters when $D$ or $M$ is large.
  - It also does not separate the position-independent word parameters (i.e. word “identity”) from the position-dependent ones.
Factored (conditional) RBM

- To reduce the number of model parameters, we represent each word using an $F$-dimensional (feature) vector of real numbers.
- We stack these vectors for all words in the dictionary to obtain a word feature matrix $R$ and express $J_i$ as a product of $R$ and another low-rank matrix $W_i$.
  - $W_i$ is an interaction matrix between the feature vector for the word in position $i$ and the hidden units.
  - The energy function becomes $E(w_1:N, h) = - \sum w_i^T RW_i h$
- This parameterization decouples the position-independent word identity parameters ($R$) and the position-dependent interaction parameters ($W_i$).
Factored RBM
Learning and inference in FRBMs

- Exact ML learning is possible but is too slow.
  - We use Contrastive Divergence learning instead.

- The learning rules for $R$ and $W_i$ are minor variations on the standard CD learning rule. E.g.:

$$
\Delta W_i = \left< R^T w_i h^T \right>_{data} - \left< R^T w_i h^T \right>_{reconstruction}
$$

- Computing the posterior distribution over the hidden units is easy.

- Making predictions using this model is tractable.
  - It takes time linear in the number of hidden units and words in the dictionary.
Temporal Factored RBM

- Would like to take advantage of indefinitely large contexts without needing a very large number of parameters.

- Turn FRBM into a temporal model:
  - Given a sequence $w_{1:t}$, apply an instance of the FRBM to each of the $n$-tuples in the sequence in succession.
  - Make the hidden units of the $n^{th}$ instance depend on the hidden units of the $(n-1)^{st}$ instance by making the hidden biases of the $n^{th}$ instance a linear function of the hidden states on the $(n-1)^{st}$ instance.
  - Make predictions as before, but use the new “shifted” biases.
Temporal Factored RBM
Inference and learning in TFRBM

- Exact inference in TFRBM is intractable due to explaining away.
  - even filtering is intractable
- We perform approximate filtering by using the mean field approximation to the previous hidden state distribution when shifting the biases.
- Temporal connections are learned greedily by treating the previous hidden state as a constant input and using the CD learning rule.
- The non-temporal parameters are learned as before.
Log-bilinear model

- It might be easier to learn direct interactions between the context words and the next word and leave out the hidden units altogether.
- We define these interactions on word feature vectors to keep the number of model parameters manageable.
- The resulting model can be viewed both as a feed-forward network and as a FRBM with visible-to-visible connections but without hidden units.

Energy function:  
$$E(w_1:n) = - \sum_{i=1}^{n-1} w_i^T R C_i R^T w_n$$
Log-bilinear model
Dataset and evaluation

- The dataset is a collection of Associated Press news stories (16 million words).
- Preprocessing (Yoshua Bengio):
  - convert all words to lower case
  - map all rare words and proper nouns to special symbols
  - Result: just under 18000 unique words.
- Models are compared based on the perplexity they assign to a test set.
  - Perplexity is the geometric mean of
    \[ \frac{1}{P(w_n|w_{1:n-1})} \]
Experiments (I)

Preliminary comparison: 10M training set, 0.5M validation set, 0.5M test set

- Feature-based models have 100D feature vectors.
- Models with hidden units have 1000 hidden units.

<table>
<thead>
<tr>
<th>Model type</th>
<th>Context size</th>
<th>Model test perplexity</th>
<th>Mixture test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRBM</td>
<td>2</td>
<td>169.4</td>
<td>110.6</td>
</tr>
<tr>
<td>Temporal FRBM</td>
<td>2</td>
<td>127.3</td>
<td>95.6</td>
</tr>
<tr>
<td>Log-bilinear</td>
<td>2</td>
<td>132.9</td>
<td>102.2</td>
</tr>
<tr>
<td>Log-bilinear</td>
<td>5</td>
<td>124.7</td>
<td>96.5</td>
</tr>
<tr>
<td>Back-off KN3</td>
<td>2</td>
<td>124.3</td>
<td></td>
</tr>
<tr>
<td>Back-off KN6</td>
<td>5</td>
<td>116.2</td>
<td></td>
</tr>
</tbody>
</table>
Experiments (II)

Final comparison: 14M training set, 1M validation set, 1M test set

- Feature-based models have 100D feature vectors.
- Models with hidden units have 1000 hidden units.

<table>
<thead>
<tr>
<th>Model type</th>
<th>Context size</th>
<th>Model test perplexity</th>
<th>Mixture test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log-bilinear</td>
<td>5</td>
<td>117.0</td>
<td>97.3</td>
</tr>
<tr>
<td>Log-bilinear</td>
<td>10</td>
<td>107.8</td>
<td>92.1</td>
</tr>
<tr>
<td>Back-off KN3</td>
<td>2</td>
<td>129.8</td>
<td></td>
</tr>
<tr>
<td>Back-off KN5</td>
<td>4</td>
<td>123.2</td>
<td></td>
</tr>
<tr>
<td>Back-off KN6</td>
<td>5</td>
<td>123.5</td>
<td></td>
</tr>
<tr>
<td>Back-off KN9</td>
<td>8</td>
<td>124.6</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Log-bilinear models outperform FRBM-based models as well as the best $n$-gram models and are easier to train than models with hidden units.
- Adding temporal connections to the FRBM model makes it perform much better.
- Averaging the predictions of any network model with a good $n$-gram model results in better predictions than using any model on its own.
- Future work: training models that have hidden units as well as direct connections; using FRBMs to train deep networks.
The End
FRBM details

- Energy function:  
  \[ E(w_{1:N}, h) = - \sum w_i^T R W_i h \]

- Joint probability of the next word and a hidden state:  
  \[ P(w_N, h|w_{1:N-1}) = \frac{1}{Z} \exp(-E(w_{1:N}, h)) \]
  \[ Z = \sum_w \exp(-E(w_{1:N}, h)) \]

- Probability of the next word:  
  \[ P(w_N|w_{1:N-1}) = \frac{1}{Z} \sum_h \exp(-E(w_{1:N}, h)) \]