Concavity and Initialization for Unsupervised Dependency Parsing

Kevin Gimpel  Noah A. Smith
Unsupervised learning in NLP \(\xrightarrow{(typically)}\) non-convex optimization
Dependency Model with Valence (Klein & Manning, 2004)

EM with 50 Random Initializers
Dependency Model with Valence (Klein & Manning, 2004)

Pearson’s $r = 0.63$
(strong correlation)
Dependency Model with Valence (Klein & Manning, 2004)

Attachment Accuracy (%) vs. Log-Likelihood (per sentence)

Range = 20%!
Dependency Model with Valence (Klein & Manning, 2004)

Attachment Accuracy (%) vs. Log-Likelihood (per sentence)

initializer from K&M04
How has this been addressed?

- Scaffolding / staged training (Brown et al., 1993; Elman, 1993; Spitkovsky et al., 2010)

- Curriculum learning (Bengio et al., 2009)

- Deterministic annealing (Smith & Eisner, 2004), Structural annealing (Smith & Eisner, 2006)

- Continuation methods (Allgower & Georg, 1990)
Example: Word Alignment

IBM Model 1 → HMM Model → IBM Model 4

Brown et al. (1993)
Example: Word Alignment

CONCAVE

IBM Model 1

HMM Model

IBM Model 4

Brown et al. (1993)
Unsupervised learning in NLP \(\rightarrow\) non-convex optimization (typically)
Unsupervised learning in NLP typically leads to non-convex optimization.

Except IBM Model 1 for word alignment

(which has a concave log-likelihood function)
IBM Model 1 \textsuperscript{(Brown et al., 1993)}

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{\mid e \mid} \log \left( \sum_{i=0}^{\mid f \mid} \frac{1}{\mid f \mid + 1} t(e_j \mid f_i) \right)
\]
IBM Model 1 (Brown et al., 1993)

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{\left|e\right|} \log \sum_{i=0}^{\left|f\right|} \frac{1}{\left|f\right| + 1} \cdot t(e_j \mid f_i)
\]

**alignment probability**  **translation probability**
**IBM Model 1**  
(Brown et al., 1993)

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{|e|} \log \sum_{i=0}^{|f|} \frac{1}{|f| + 1} t(e_j \mid f_i)
\]

alignment probability  translation probability

**IBM Model 2**

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{|e|} \log \sum_{i=0}^{|f|} a(i \mid j, |f|, |e|) \ t(e_j \mid f_i)
\]
**IBM Model 1**

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{\mid e \mid} \sum_{i=0}^{\mid f \mid} \frac{1}{\mid f \mid + 1} t(e_j \mid f_i)
\]

**IBM Model 2**

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{\mid e \mid} \sum_{i=0}^{\mid f \mid} a(i \mid j, \mid f \mid, \mid e \mid) t(e_j \mid f_i)
\]
IBM Model 1

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{\left\vert e \right\vert} \log \left( \sum_{i=0}^{\left\vert f \right\vert} \frac{1}{\left\vert f \right\vert + 1} t(e_j \mid f_i) \right)
\]

alignment probability

translation probability

IBM Model 2

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{\left\vert e \right\vert} \log \left( \sum_{i=0}^{\left\vert f \right\vert} a(i \mid j, \left\vert f \right\vert, \left\vert e \right\vert) t(e_j \mid f_i) \right)
\]

product of parameters within log-sum

CONCAVE

NOT CONCAVE
IBM Model 1

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{|e|} \log \sum_{i=0}^{|f|} \frac{1}{|f| + 1} t(e_j \mid f_i)
\]

For concavity:

1 parameter is permitted for each atomic piece of latent structure.

No atomic piece of latent structure can affect any other piece.

\[
\log p(e \mid f) = \log \epsilon + \sum_{j=1}^{|e|} \log \sum_{i=0}^{|f|} a(i \mid j, |f|, |e|) t(e_j \mid f_i)
\]

product of parameters within log-sum
Unsupervised learning in NLP → non-convex optimization

Except IBM Model 1 for word alignment
(which has a concave log-likelihood function)

What models can we build without sacrificing concavity?
For concavity:

1 parameter is permitted for each atomic piece of latent structure.

No atomic piece of latent structure can affect any other piece.
For concavity:

1 parameter is permitted for each atomic piece of latent structure.

No atomic piece of latent structure can affect any other piece.
For concavity:

1 parameter is permitted for each atomic piece of latent structure.

No atomic piece of latent structure can affect any other piece.

Every dependency arc must be independent, so we can’t use a tree constraint.
For concavity:

1 parameter is permitted for each atomic piece of latent structure.

No atomic piece of latent structure can affect any other piece.

Every dependency arc must be independent, so we can’t use a tree constraint

Only one parameter allowed per dependency arc
For concavity:

1 parameter is permitted for each atomic piece of latent structure.

No atomic piece of latent structure can affect any other piece.

Our Model:

Like IBM Model 1, but we generate the same sentence again, aligning words to the original sentence (cf. Brody, 2010)

\[
\log p(e \mid e') = \log \epsilon + \sum_{j=1}^{\mid e \mid} \log \sum_{i=0, i \neq j}^{\mid e \mid} \frac{1}{\mid e \mid + 1} \text{child}(e_j \mid e'_i)
\]
Vikings came in longboats from Scandinavia in 1000 AD
$ \text{Vikings came in longboats from Scandinavia in 1000 AD}$
$\text{Vikings came in longboats from Scandinavia in 1000 AD}$

$\text{Vikings came}$

$\text{Vikings came in longboats from Scandinavia in 1000 AD}$
$\textit{Vikings came in longboats from Scandinavia in 1000 AD}$
$\text{Vikings came in longboats from Scandinavia in 1000 AD}$

$\text{Vikings came in longboats}$

$\text{Vikings came in longboats from Scandinavia in 1000 AD}$
$\text{Vikings came in longboats from Scandinavia in 1000 AD}$

Cycles, multiple roots, and non-projectivity are all permitted by this model.
Vikings came in longboats from Scandinavia in 1000 AD

Only one parameter per dependency arc:
Vikings came in longboats from Scandinavia in 1000 AD

Only one parameter per dependency arc:

\[ p(\text{Vikings} \mid \text{came}) \]
Vikings came in longboats from Scandinavia in 1000 AD

Only one parameter per dependency arc:

\[ p(\text{Vikings} \mid \text{came}) \]

We cannot look at other dependency arcs, but we can condition on (properties of) the sentence:

\[ p(\text{Vikings} \mid \text{came}, \text{direction} = \text{left}, \text{distance} = 1, \ldots) \]
Vikings came in longboats from Scandinavia in 1000 AD

We condition on direction:

\[ p(\text{Vikings} \mid \text{came}, \text{direction} = \text{left}) \]

(“Concave Model A”)
Note: we’ve been using words in our examples, but in our model we follow standard practice and use gold POS tags

We condition on direction:

\[ p(\text{NNPS} \mid \text{VBD}, \text{direction} = \text{left}) \]

(“Concave Model A”)
We condition on direction:

\[ p(\text{NNPS} \mid \text{VBD}, \text{direction} = \text{left}) \]

(“Concave Model A”)

<table>
<thead>
<tr>
<th>Model</th>
<th>Initializer</th>
<th>Accuracy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Right</td>
<td>N/A</td>
<td>31.7</td>
</tr>
<tr>
<td>DMV</td>
<td>Uniform</td>
<td>17.6</td>
</tr>
<tr>
<td>DMV</td>
<td>K&amp;M</td>
<td>32.9</td>
</tr>
<tr>
<td>Concave Model A</td>
<td>Uniform</td>
<td>25.6</td>
</tr>
</tbody>
</table>

*Penn Treebank test set, sentences of all lengths
WSJ10 used for training
**Note:**

IBM Model 1 is not *strictly* concave
(Toutanova & Galley, 2011)

<table>
<thead>
<tr>
<th>Model</th>
<th>Initializer</th>
<th>Accuracy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Right</td>
<td>N/A</td>
<td>31.7</td>
</tr>
<tr>
<td>DMV</td>
<td>Uniform</td>
<td>17.6</td>
</tr>
<tr>
<td>DMV</td>
<td>K&amp;M</td>
<td>32.9</td>
</tr>
<tr>
<td>Concave Model A</td>
<td>Uniform</td>
<td>25.6</td>
</tr>
</tbody>
</table>

We condition on direction:

\[ p(\text{NNPS} \mid \text{VBD}, \text{direction} = \text{left}) \]

(“Concave Model A”)

*Penn Treebank test set, sentences of all lengths
WSJ10 used for training
We can also use hard constraints while preserving concavity:

The only tags that can align to $ are verbs
(Mareček & Žabokrtský, 2011; Naseem et al., 2010)

(“Concave Model B”)

$ NNPS VBD IN NNS IN NNP IN CD NN

NNPS VBD IN NNS

$\text{NNPS VBD IN NNS IN NNP IN CD NN}$

NNPS VBD IN NNS

<table>
<thead>
<tr>
<th>Model</th>
<th>Initializer</th>
<th>Accuracy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Right</td>
<td>N/A</td>
<td>31.7</td>
</tr>
<tr>
<td>DMV</td>
<td>Uniform</td>
<td>17.6</td>
</tr>
<tr>
<td>DMV</td>
<td>K&amp;M</td>
<td>32.9</td>
</tr>
<tr>
<td>Concave Model A</td>
<td>Uniform</td>
<td>25.6</td>
</tr>
<tr>
<td>Concave Model B</td>
<td>Uniform</td>
<td>28.6</td>
</tr>
</tbody>
</table>

*Penn Treebank test set, sentences of all lengths
WSJ10 used for training
Unsupervised learning in NLP \( \Rightarrow \) non-convex optimization

Except IBM Model 1 for word alignment

(which has a concave log-likelihood function)

What models can we build without sacrificing concavity?

Can these concave models be useful?
As IBM Model 1 is used to initialize other word alignment models, we can use our concave models to initialize the DMV.
As IBM Model 1 is used to initialize other word alignment models, we can use our concave models to initialize the DMV.

<table>
<thead>
<tr>
<th>Model</th>
<th>Initializer</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Right</td>
<td>N/A</td>
<td>31.7</td>
</tr>
<tr>
<td>DMV</td>
<td>Uniform</td>
<td>17.6</td>
</tr>
<tr>
<td>DMV</td>
<td>K&amp;M</td>
<td>32.9</td>
</tr>
<tr>
<td>DMV</td>
<td>Concave Model A</td>
<td>34.4</td>
</tr>
<tr>
<td>DMV</td>
<td>Concave Model B</td>
<td>43.0</td>
</tr>
</tbody>
</table>

*Penn Treebank test set, sentences of all lengths
WSJ10 used for training
As IBM Model 1 is used to initialize other word alignment models, we can use our concave models to initialize the DMV.

<table>
<thead>
<tr>
<th>Model</th>
<th>Initializer</th>
<th>Accuracy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMV, trained on sentences of length ≤ 20</td>
<td>Concave Model B</td>
<td>53.1</td>
</tr>
<tr>
<td>Shared Logistic Normal (Cohen &amp; Smith, 2009)</td>
<td>K&amp;M</td>
<td>41.4</td>
</tr>
<tr>
<td>Posterior Regularization (Gillenwater et al., 2010)</td>
<td>K&amp;M</td>
<td>53.3</td>
</tr>
<tr>
<td>LexTSG-DMV (Blunsom &amp; Cohn, 2010)</td>
<td>K&amp;M</td>
<td>55.7</td>
</tr>
<tr>
<td>Punctuation/UnsupTags (Spitkovsky et al., 2011), trained on sentences of length ≤ 45</td>
<td>K&amp;M'</td>
<td>59.1</td>
</tr>
</tbody>
</table>

*Penn Treebank test set, sentences of all lengths
# Multilingual Results
(averages across 18 languages)

<table>
<thead>
<tr>
<th>Model</th>
<th>Initializer</th>
<th>Avg. Accuracy*</th>
<th>Avg. Log-Likelihood †</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMV</td>
<td>Uniform</td>
<td>25.7</td>
<td>-15.05</td>
</tr>
<tr>
<td>DMV</td>
<td>K&amp;M</td>
<td>29.4</td>
<td>-14.84</td>
</tr>
<tr>
<td>DMV</td>
<td>Concave Model A</td>
<td>30.9</td>
<td>-14.93</td>
</tr>
<tr>
<td>DMV</td>
<td>Concave Model B</td>
<td><strong>35.5</strong></td>
<td><strong>-14.45</strong></td>
</tr>
</tbody>
</table>

* Sentences of all lengths from each test set
† Micro-averaged across sentences in all training sets
(used sentences ≤ 10 words for training)
Unsupervised learning in NLP \(\text{(typically)}\) non-convex optimization

Except IBM Model 1 for word alignment
(which has a concave log-likelihood function)

What models can we build without sacrificing concavity?

Can these concave models be useful?

Like word alignment, we can use simple, concave models to initialize more complex models for grammar induction
Thanks!