Bilingual Word Embeddings and Recurrent Neural Networks

Fabienne Braune

1LMU Munich

June 28, 2017
Outline

1. Softmax Output Units
2. Word Embeddings
3. Bilingual Word Embeddings
4. Recurrent Neural Networks
5. Recap
Softmax Output Units
Backpropagation

Goal of training: adjust weights such that correct label is predicted

→ Error between correct label and prediction is minimal

Sketch:

- Compute **derivatives** of Error with respect to prediction
- Compute **derivatives** in each hidden layer from layer above
  - Backpropagate the error derivative with respect to the output of a unit
- Use **derivatives** w.r.t the activities to get error **derivatives** w.r.t incoming weights
**Backpropagation**

\[ E(\theta_i, y^i) \]

\[ \frac{\partial E}{\partial A_j} \]

\[ \frac{\partial E}{\partial O_i} \]

---

**Backpropagation:**

→ Compute \( E \)

→ Compute \( \frac{\partial E}{\partial O_i} \)
Compute **error at output E:**

Compare **output unit with** $y^i$

$$E = \frac{1}{2} \sum_{i=1}^{n} (y_i - O_i)^2 \text{ (mean squared)}$$

Compute $\frac{\partial E}{\partial O_i}$:

$$\frac{\partial E}{\partial O_i} = -(y_i - O_i)$$
Backpropagation 2

Compute **derivatives** in each hidden layer from layer above:

Compute derivative of error w.r.t logit

$$\frac{\partial E}{\partial Z_i} = \frac{\partial E}{\partial O_i} \frac{\partial O_i}{\partial Z_i} = \frac{\partial E}{\partial O_i} O_i (1 - O_i) \quad \text{(Note: } O_i = \frac{1}{1+e^{-Z_i}})$$

Compute derivative of error w.r.t previous hidden unit

$$\frac{\partial E}{\partial A_j} = \sum_i \frac{\partial Z_i}{\partial A_j} \frac{\partial E}{\partial Z_i} = \sum_i w_{ji} \frac{\partial E}{\partial Z_i}$$

Compute derivative w.r.t. weights

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial Z_i}{\partial w_{ji}} \frac{\partial E}{\partial Z_i} = O_i \frac{\partial E}{\partial Z_i}$$

→ Use **recursion** to do this for every layer
Problems with least squares

1. Poor gradient although \textbf{big error}

Suppose $Y_i = 1$ and $O_i = 0.00000001 \rightarrow$ Very wrong

Least squares:

$$E = \frac{1}{2} \sum_{i=1}^{n} (1 - 0.00000001)^2 \text{ (mean squared)}$$

$$\rightarrow \frac{\partial E}{\partial O_i} = -(1 - 0.00000001)$$

$$\rightarrow \frac{\partial E}{\partial Z_i} = \frac{\partial E}{\partial O_i} \ast 0.00000001(1 - 0.00000001)$$

Suppose $Y_i = 0$ and $O_i = 0.00000001 \rightarrow$ Quite right

2. Mutually exclusive classes $\rightarrow$ Probabilities should sum up to 1

$\rightarrow$ Give the network this information
Softmax Unit

Softmax unit:
- applied on output logits
- \( O_i = \frac{e^{z_i}}{\sum_{j \in K} e^{z_j}} \)
Cross Entropy

Cross Entropy:

\[ C = - \sum_j y_j \log(O_j) \]

\[ \rightarrow \frac{\partial C}{\partial Z_i} = \sum_j \frac{\partial C}{\partial O_j} \frac{\partial O_j}{\partial Z_i} = O_i - y_i \]

- Very big gradient when target is 1 and output near 0
- Mutually exclusive classes taken into account
Word Embeddings
Word Embeddings

- Representation of words in vector space
Word Embeddings

- Similar words are close to each other
  \[\Rightarrow\] Similarity is the cosine of the angle between two word vectors
Learning word embeddings

Count-based methods:
- Compute cooccurrence statistics
- Learn high-dimensional representation
- Map sparse high-dimensional vectors to small dense representation

Neural networks:
- Predict a word from its neighbors
- Learn (small) embedding vectors
Word2Vec

- Software train word embeddings (Mikolov. 2013)
  \[ \rightarrow \text{very fast} \]

- Two models:
  - BOW model:
    - Input is \( w_{t+2}, w_{t+1}, w_{t-1} \) and \( w_{t-2} \)
    - Prediction is \( w_t \)
  - Skip-gram model:
    - Input is \( w_t \)
    - Prediction is \( w_{t+2}, w_{t+1}, w_{t-1} \) and \( w_{t-2} \)
Learning word embeddings with CBOV

Note: Bias terms omitted for simplicity
Learning word embeddings with skip-gram

\[ L_t \leftarrow W_t \]

\[ O_{-2} \]
\[ O_{-1} \]
\[ O_{+1} \]
\[ O_{+2} \]

\[ W_{ti} \quad U \quad \text{word feats} \quad C \quad \text{word} \]

Note: Bias terms omitted for simplicity
Bilingual Word Embeddings
Bilingual Word Spaces

Representation of words in two languages in same semantic space:

→ Each word is one dimension
→ Each word represented respective to all others

![Diagram of bilingual word spaces with examples in German and English]
Bilingual Word Spaces

Representation of words in two languages in same semantic space:

→ Similar words are close to each other
→ Given by cosine
Exercise

How is this related to translation?
Learning Bilingual Word Embeddings

- Learn monolingual word embeddings and map using seed lexicon
  Mikolov et al. (2013); Faruqui and Dyer (2014); Lazaridou et al. (2015)
  Need seed lexicon

- Learn bilingual embeddings or lexicon from document-aligned data
  Vulic and Moens (2015); Vulic and Korhonen (2016)
  Need document-aligned data

- Learn bilingual embeddings from parallel data
  Need for parallel data
Post-hoc mapping (with seed lexicon)

- Learn monolingual word embeddings
- Learn a linear mapping $W$

\[
\begin{align*}
\text{rich} & \quad \downarrow \\
\text{silver} & \quad \downarrow \\
\text{disease} & \quad \downarrow \\
\text{poor} & \quad \downarrow \\
\end{align*}
\]
Post-hoc mapping

- Project source words into target space

![Diagram showing word embeddings with axes labeled rich, poor, Reich, silver, Silber, Gesellschaft, Krankheit, disease, Arm.](image-url)
Post-hoc Mapping with seed lexicon

1. Train monolingual word embeddings (Word2vec) in English
   - Need English monolingual data

2. Train monolingual word embeddings (Word2vec) in German
   - Need German monolingual data

3. Learn mapping $W$ using a seed lexicon
   - Need a list of 5000 English words and their translation
Learning $W$ with Ridge Regression

Ridge regression (Mikolov et al. (2013))

$$W^* = \arg\min_W \sum_{i}^{n} \| x_i W - y_i \|^2$$

$x_i$ : embedding of i-th source (English) word in the seed lexicon.

$y_i$ : embedding of i-th target (German) word in the seed lexicon.
Learning $W$ with Ridge Regression

$x_i$: embedding of $i$-th source (English) word in the seed lexicon.

→ vector representing silver in monolingual word embedding

![Diagram showing vectors for rich, silver, society, and disease in a 2D space, with an angle marked by $\alpha$.]
Learning $W$ with Ridge Regression

Ridge regression (Mikolov et al. (2013))

$$W^* = \arg \min_W \sum_{i=1}^{n} \| x_i W - y_i \|^2$$

$x_i$: embedding of i-th source (English) word in the seed lexicon.

$y_i$: embedding of i-th target (German) word in the seed lexicon.
Learning $W$ with Ridge Regression

$y_i$: embedding of $i$-th target (German) word in the seed lexicon.

→ vector representing Silber in monolingual word embedding
Learning $W$ with Ridge Regression

Ridge regression (Mikolov et al. (2013))

$$W^* = \arg \min_W \sum_{i}^{n} \| x_i \cdot W - y_i \|^2$$

- Predict projection $y^*$ by computing $x_i \cdot W$
- Compute squared error between $y^*$ and $y_i$
  - Correct translation $t_i$ given in seed lexicon
  - Vector representation $y_i$ is given by embedding of $t_i$
- Find $W$ such that squared error over training set is minimal
Adding Regularization

If \( W \) is too complex the model overfits the data
→ Add regularization term that keeps \( W \) small
→ Add weighted norm of \( W \) to cost function

\[
W^* = \arg\min_W \sum_{i} ||x_i \cdot W - y_i||^2 + \lambda \| W \| 
\]
Bilingual lexicon induction

- Task to evaluate bilingual word embeddings extrinsically
- Given a set of source words, find the corresponding translations:
  - Given silver, find its vector in the BWE
  - Retrieve the German word whose vector is closest (cosine distance)
Bilingual lexicon induction with ridge regression

**Data:** WMT 2011 training data for English, Spanish, Czech  
**Seed:** 5000 most frequent words translated with Google Translate  
**Test:** 1000 next frequent words translated with Google Translate

→ Removed digits, punctuation and transliterations

<table>
<thead>
<tr>
<th>Languages</th>
<th>top-1</th>
<th>top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>En-Es</td>
<td>33 %</td>
<td>51 %</td>
</tr>
<tr>
<td>Es-En</td>
<td>35 %</td>
<td>50 %</td>
</tr>
<tr>
<td>En-Cz</td>
<td>27 %</td>
<td>47 %</td>
</tr>
<tr>
<td>Cz-En</td>
<td>23 %</td>
<td>42 %</td>
</tr>
<tr>
<td>+ Es-En</td>
<td>53 %</td>
<td>80 %</td>
</tr>
</tbody>
</table>

→ with spanish google news
Learning $W$ with Max Margin Ranking

Max-margin ranking loss (Lazaridou et al. (2015)):

- Predict projection $y^*$ by computing $x_i \cdot W$
- Compute ranking loss between:
  - $y^*$
  - Vector of correct translation $y_i$
  - Negative samples $y_j$
- $\sum_{i \neq j}^k \max\{0, \gamma + Sdist(y^*, \vec{y}_i) - Sdist(y^*, \vec{y}_j)\}$
  - $Sdist(\vec{x}, \vec{y})$: inverse cosine
  - Measures semantic distance between $\vec{y}^*$ and $\vec{y}_i$
  - $\gamma$ and $k$ tuned on held-out data
Learning $W$ with Max Margin Ranking

Max-margin ranking loss (Lazaridou et al. (2015)):

$$\sum_{i \neq j}^k \max\{0, \gamma + Sdist(\vec{y}^*, \vec{y}_i) - Sdist(\vec{y}^*, \vec{y}_j)\}$$

- $Sdist(\vec{x}, \vec{y})$: inverse cosine
- $\rightarrow$ measures semantic distance between $\vec{y}^*$ and $\vec{y}_i$

- For each source (English) vector $x_i$, distance of $\vec{y}^*$ to correct translation $\vec{y}_i$ should be smaller than distance to wrong translation $\vec{y}_j$
Bilingual lexicon induction with max margin ranking

Data: 4 mio sentences from Europarl, News, Common Crawl
Seed: 5000 most frequent words-pairs computed with parallel data
Test: 1000 next words-pairs computed with parallel data

<table>
<thead>
<tr>
<th>Setup</th>
<th>top-1</th>
<th>top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>En-De all</td>
<td>18.6 %</td>
<td>27.4 %</td>
</tr>
<tr>
<td>En-De</td>
<td>23.1 %</td>
<td>33.61 %</td>
</tr>
</tbody>
</table>

→ max-margin outperforms ridge
Building bilingual corpora

Idea:

- Create bilingual corpus and build bilingual word embeddings
- Combine monolingual texts to create bilingual data
- Learn word embeddings with skip-gram or CBOW on bilingual data
  - Simply run word2vec on the bilingual data
  - Just need to create bilingual data
Merge and shuffle document-aligned monolingual data (Vulic and Moens (2015)):

- **Document-pairs** \( P = \{(D_1^S, D_1^T), \ldots, (D_n^S, D_n^T)\} \)
- **Merge** each pair \((D_i^S, D_i^T)\) into pseudo-bilingual document \(B_i\)
- **Shuffle** each \(B_i\)
  - Random permutation of words \(w_j\) in \(B_i\)
  - Assures that each word \(w_j\) obtains collocates from both languages
- **Train** word embeddings (word2vec) on pseudo-bilingual document \(B_i\)
Building bilingual corpora

English word with bilingual context

\[ w_t \rightarrow L_t \]

Note: Bias terms omitted for simplicity
Building bilingual corpora

German word with bilingual context

Note: Bias terms omitted for simplicity
Bilingual Word Spaces

Representation of words in two languages in same semantic space:

→ **Similar** words are **close** to each other

→ Given by **cosine**
Merge and Shuffle with seed lexicon

Merge and shuffle monolingual data with seed lexicon

(Gouws and Søgaard (2015)):

- Document-pair $P = (D_1^S, D_1^T)$
  - Merge each pair $P$ into pseudo-bilingual document $B$
  - Shuffle $B$

- Seed lexicon $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

- Each $y_i$ is translation of $x_i$
  - In bilingual document $B$ replace each $x_i$ with $y_i$ with proba 0.5
  - Allows to consider $k$ translations of $x_i$ and draw with proba $\frac{0.5}{k}$
Bilingual lexicon induction

- Task to evaluate bilingual word embeddings extrinsically
- Merge and shuffle document-aligned monolingual data (Vulic and Moens (2015))
- A bit worse than post-hoc mapping with ridge regression
- Merge and shuffle monolingual data with seedLexicon (Gouws and Søgaard (2015))
- Evaluated on cross-lingual POS tagging
Recurrent Neural Networks
Neural language model

- Early application of neural networks (Bengio et al. 2003)
- Task: Given $k$ previous words, predict the current word
  
  Estimate: $P(w_t|w_{t-k}, \ldots, w_{t-2}, w_{t-1})$

- Previous (non-neural) approaches:

  **Problem**: Joint distribution of consecutive words difficult to obtain
  → chose small history to reduce complexity ($n=3$)
  → predict for unseen history through back-off to smaller history

  **Drawbacks**:
  - Takes into account small and fixed context
  - Does not model similarity between words
Neural language model

Take into account context of any size:
- Need a way to model sequentiality
- Introduce notion of time in neural network
  → Recurrent Neural Networks
Recurrent Neural Networks

Connection between hidden states
→ connections between time units, models sequentiality

\[ \text{LT}_1 \quad \text{A}_1 \quad \text{Z}_1 \quad E(O_i, y^i) \]
\[ \text{LT}_2 \quad \text{...} \quad \text{...} \quad E(O_i, y^i) \]
\[ \text{LT}_3 \quad \text{A}_3 \quad \text{Z}_K \quad E(O_i, y^i) \]

input \quad U \quad R \quad V
Recurrent Neural Networks

Can be bidirectional

\[ LT_1 \rightarrow A_1 \rightarrow Z_1 \]
\[ LT_2 \rightarrow \ldots \rightarrow \ldots \]
\[ LT_3 \rightarrow A_3 \rightarrow Z_K \]

input \[ U \]
\[ R \]
\[ V \]
Forward Propagation

Input embeddings passed forward through time
Each hidden unit is one time step
Forward Propagation

Specify initial state $A_0$:

**Input layer ($X$):** Word features $LT^t$

**Weight matrices** $U, R, V$

**Time Step ($A^t$):** $\sigma(LT^t \cdot U + A^{t-1} \cdot R + d)$

**Output layer ($0^t$):** $A^t \cdot V + b$

**Prediction:** $h^t(X) = \text{softmax}(0^t)$
Forward Propagation

Compute prediction for each time step
Apply softmax on each output
Forward Propagation

Compute prediction for one time step
Apply softmax on last output → Language model architecture
Backpropagation

Goal of training: adjust weights such that correct label is predicted

→ Error between correct label and prediction is minimal

Sketch:

- Compute derivative of Error wrt. prediction
- Compute derivatives in each hidden layer from layer above
  - Backpropagate the error derivative with respect to the output of a unit
- Use derivatives w.r.t the activities to get error derivatives w.r.t incoming weights
Backpropagation through time

Sketch:

- Compute **derivative** of *Error wrt. prediction*
- Compute **derivatives** from *layer above and previous time step*
  - Each time step can be represented by a *feedforward neural network*
  - Missing connections represented by *constrained weights (same)*
  - Derivatives of same weights are *averaged*

Difficulties:

- Multiply many derivatives together
  - Gradients tend to **explode or vanish**
Recap

- Squared error not good loss function
  - *Softmax* units with cross-entropy

- **Bilingual word embeddings** represent words in two languages

- Induction with post-hoc mapping:
  - Train *monolingual* word embeddings
  - Map with *seed lexicon*

- Induction with bilingual corpora:
  - Create *bilingual* corpora
  - Train *monolingual* word embeddings
Recap

Recurrent neural networks for language modeling:

- **Task:** Given $k$ previous words, predict the current word
  
  Estimate: $P(w_t | w_{t-k}, \cdots, w_{t-2}, w_{t-1})$

- **Problems with feedforward approach**
  
  $\rightarrow$ chose **fixed** history to reduce complexity

- **Recurrent neural networks** as solution
  
  - Model **sequentiality** with recurrent units
  - Allow to model history of **any size**


