Supervised sentiment analysis: RNN classifiers

Christopher Potts

Stanford Linguistics

CS224u: Natural language understanding
Model overview

For complete details, see the reference implementation `np_rnn_classifier.py`
Distributed representations as features

Classifier prediction

Lexical vectors combined via a function like sum or mean. These are the inputs to the classifier.

Embedding look-up

Model overview  Distributed representations as features  Standard RNN dataset preparation  A note on LSTMs  Code snippets
Standard RNN dataset preparation

Examples

\[
\begin{align*}
& a, b, a \\
\Rightarrow & 1, 2, 1 \\
& b, c \\
& 2, 3 \\
& \downarrow \\
& \text{Vectors} \\
& h_{1} = [−0.42, 0.10, 0.12], \\
& h_{2} = [−0.16, −0.21, 0.29], \\
& h_{3} = [−0.42, 0.10, 0.12] \\
& i_{1} = [−0.16, −0.21, 0.29], \\
& i_{2} = [−0.26, 0.31, 0.37] \\
\end{align*}
\]

Embedding

\[
\begin{align*}
& 1 = [−0.42, 0.10, 0.12] \\
& 2 = [−0.16, −0.21, 0.29] \\
& 3 = [−0.26, 0.31, 0.37]
\end{align*}
\]
Standard RNN dataset preparation

**Examples**

[a, b, a]
[b, c]
Standard RNN dataset preparation

Examples
- [a, b, a]
- [b, c]

Indices
- [1, 2, 1]
- [2, 3]
Standard RNN dataset preparation

Examples

[a, b, a]
[b, c]

Indices

[1, 2, 1]
[2, 3]

<table>
<thead>
<tr>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
Standard RNN dataset preparation

**Examples**

- [a, b, a]
- [b, c]

**Indices**

- [1, 2, 1]
- [2, 3]

**Vectors**

- \([[-0.42, 0.10, 0.12], [-0.16, -0.21, 0.29], [-0.42, 0.10, 0.12]]\)
- \([[-0.16, -0.21, 0.29], [-0.26, 0.31, 0.37]]\)

**Embedding**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.42</td>
<td>0.10</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>-0.16</td>
<td>-0.21</td>
<td>0.29</td>
</tr>
<tr>
<td>3</td>
<td>-0.26</td>
<td>0.31</td>
<td>0.37</td>
</tr>
</tbody>
</table>
A note on LSTMs

1. Plain RNNs tend to perform poorly with very long sequences; as information flows back through the network, it is lost or distorted.

2. LSTM cells are a prominent response to this problem: they introduce mechanisms that control the flow of information.

3. We won’t review all the mechanism for this here. I instead recommend these excellent blog posts, which include intuitive diagrams and discuss the motivations for the various pieces in detail:
   - Towards Data Science: Illustrated Guide to LSTM’s and GRU’s: A step by step explanation
   - colah’s blog: Understanding LSTM networks
Code snippets

[1]:
```python
import os
from torch_rnn_classifier import TorchRNNClassifier
import torch.nn as nn
import sst
import utils
```

[2]:
```python
GLOVE_HOME = os.path.join('data', 'glove.6B')
SST_HOME = os.path.join('data', 'sentiment')
```

[3]:
```python
GLOVE_LOOKUP = utils.glove2dict(os.path.join(GLOVE_HOME, 'glove.6B.50d.txt'))
```

[4]:
```python
def rnn_phi(text):
    return text.lower().split()
```

[5]:
```python
def fit_rnn(X, y):
    sst_train_vocab = utils.get_vocab(X, mincount=2)
    glove_embedding, sst_glove_vocab = utils.create_pretrained_embedding(
        GLOVE_LOOKUP, sst_train_vocab)
    mod = TorchRNNClassifier(
        sst_glove_vocab,
        eta=0.01,
        embedding=glove_embedding,
        batch_size=1028,
        hidden_dim=50,
        12_strength=0.001,
        bidirectional=True,
        max_iter=50,
        early_stopping=True)
    mod.fit(X, y)
    return mod
```

[6]:
```python
rnn_experiment = sst.experiment(
    sst.train_reader(SST_HOME),
    rnn_phi,
    fit_rnn,
    vectorize=False)
```