Introduction to Recurrent Neural Network

Bokan Bao
Professor Dasgupta
CSE 254

Outlines

• RNN introduction
  • Limitation of Vanilla Neural Network
  • Structure of the RNN
  • PyTorch implementation

• Character-Level prediction
  • Unsmoothed Maximum Likelihood Character Level Language Model
  • Three-layer RNN
Vanilla Neural Network

They accept a fixed-sized vector as input (e.g., an image) and produce a fixed-sized vector as output (e.g., probabilities of different classes).

Recurrent Neural Network

\[ h_0 \]
\[ A \]
\[ X_0 \]

\[ h_0 \]
\[ h_1 \]
\[ A \]
\[ X_1 \]

\[ h_2 \]
\[ A \]
\[ X_2 \]

\[ \ldots \]

\[ h_n \]
\[ A \]
\[ X_n \]
Recurrent Neural Network

• Flexible, a sequence of the data
• Music, text, motion capture
• The predictive distribution depends on the previous inputs.

A sequence of many things
Simple RNN model

\[ x_t \rightarrow W_{xh}x_t + b_{xh} \]
\[ h_{t-1} \rightarrow h_t \]
\[ h_t = \tanh(W_{hh}h_{t-1} + b_{hh} + W_{xh}x_t + b_{xh}) \]
\[ y_t = W_{hy}h_t + b_{hy} \]

Pytorch implementation

```python
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super().__init__()
        self.hidden_size = hidden_size
        self.W_xh = nn.Linear(input_size, hidden_size)
        self.W.hh = nn.Linear(hidden_size, hidden_size)
        self.W.hy = nn.Linear(hidden_size, output_size)
        self.act = nn.Tanh()
        self.dropout = nn.Dropout(0.1)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        hidden = self.W_xh(input).add(self.W.hh(hidden))
        hidden = self.act(hidden)
        output = self.dropout(hidden)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)
```

```python
```
Loss function: Negative Log Likelihood Loss

Optimization function: Adam

• Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper (poster) titled “Adam: A Method for Stochastic Optimization”. I will quote liberally from their paper in this post, unless stated otherwise.

• The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:
  • Adaptive Gradient Algorithm (AdaGrad)
  • Root Mean Square Propagation (RMSProp)
Training

```python
loss_fun = nn.NLLLoss()
optimizer = torch.optim.Adam(rnn.parameters(), lr=0.01)

while epoch < 10:
    if p + seq_length + 1 > len(data):
        epoch += 1
        p = 0

# Generate input and target
input_line_tensor, target_line_tensor = createTrainingExample(data[p:p+seq_length+1])
target_line_tensor.unsqueezed([-1])

# Initialization
hidden = rnn.initHidden()
nrn.zero_grad()

# Training
for i in range(input_line_tensor.size(0)):
    output, hidden = rnn(input_line_tensor[i], hidden)

    l = loss_fun(output, target_line_tensor[i])
    loss += l

# Back propagation
optimizer.zero_grad()
loss.backward()
optimizer.step()

p += seq_length # move data pointer
```

Character-Level Predict Model

- Unsmoothed Maximum Likelihood Model
  - c is character, h is a n letters history
  - P(c|h) stands for how likely is it to see c after we've seen h.
- Training data: Shakespeare
  - demo: A 4-letter model
  - Deterministic model
  - High dimensions

```python
def train_char_lm(filename, order=4):
    data = file(filename).read()
    lm = defaultdict(Counter)
    pad = '—' * order
    data = pad + data

    for i in range(len(data)-order):
        history, char = data[i:i+order], data[i+order]
        lm[history][char] += 1

def normalize(counter):
    return {c: k / float(sum(counter.values()))
            for c, k in counter.items()}

outlm = {hist: normalize(chars) for hist, chars in lm.items()}
```
Result from 10-letter model

First Citizen:
May, then, that was hers,
It speaks against your other service;
But since the
youth of the circumstance be spoken:
Your uncle and one Baptist's daughter.

SEBASTIAN:
Do I stand till the break off.

BIRON:
Hide thy head.

VENTIDIOUS:
He purposeth to Athens: whither, with the vow
I made to handle you.

FALSTAFF:
My good knife.

MALVOLIO:
Sad, lady! I could be forgiven you, you're welcome. Give ear, sir, my doublet and hose and leave this present dest

Second Gentleman:
Who may that she confess it is my lord enraged and forestalled ere we come to be a man. Drown thyself?

Three layers RNN model

This is a 3-layer RNN with 512 hidden
nodes on each layer.

We will need
1. 68 x 512
2. 512 x 512
3. 512 x 512
4. 512 x 512
5. 512 x 68

856,064 parameters in total
N_para x 4 bytes = 3.42 MB
Backward pass requires 3 times of the memory
3.42*3 ~ 10.26 MB
Three layers RNN model

PANDARUS:
Ali, I think he shall be come approached and the day
When little strain would be attain’d into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I’ll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I’ll drink it.

For xml format (multi-layer LSTM)

<?xml version="1.0" encoding="UTF-8"?>
<page>
<title>Antichrist</title>
<id>865</id>
<revision>
  <id>15900576</id>
  <timestamp>2002-08-03T18:14:12Z</timestamp>
  <contributor>
    <username>Paris</username>
    <id>23</id>
  </contributor>
  <minor />
  <comment>Automated conversion</comment>
  <text xml:space="preserve">#REDIRECT [[Christianity]]></text>
</revision>
</page>
Even Latex (multi-layer LSTM)

\begin{proof}
We may assume that $\mathcal{I}$ is an abelian sheaf on $\mathcal{C}$. 
\item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$ is an injective and let $\mathcal{F}$ be an abelian sheaf on $X$. 
Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a category. 
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\item \hyperref[lemma-characterize-quasi-finite]{Lemma}
\end{enumerate}
\end{proof}

Thanks

• References
  • [https://pytorch.org/tutorials/](https://pytorch.org/tutorials/)
  • [https://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139](https://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139)
  • [https://karpathy.github.io/2015/05/21/rnn-effectiveness/](https://karpathy.github.io/2015/05/21/rnn-effectiveness/)
  • [https://cs.stanford.edu/people/karpathy/char-rnn/shakespear.txt](https://cs.stanford.edu/people/karpathy/char-rnn/shakespear.txt)
  • [https://gist.github.com/karpathy/d4dee566867f8291f086](https://gist.github.com/karpathy/d4dee566867f8291f086)

• Code on github
  • [https://github.com/bobaoai/Sample_RNN.git](https://github.com/bobaoai/Sample_RNN.git)