Deep Learning

Exercise 10: Learn to Write like Shakespeare

Instructor: Manuel Günther
Email: guenther@ifi.uzh.ch
Office: AND 2.54

Friday, May 14, 2021

Outline

1. Learn to Write like Shakespeare

Outline

1. Learn to Write like Shakespeare
   - Shakespeare Poem
   - Network Implementation
   - Text Generation

Shakespeare Poem

The Sonnets

- Poem by Shakespeare
- Around 2400 lines of text

Learn Text

- Train simple RNN to predict next character
- Here: fixed context size during training
  - → Variable context for test

Example Strophe

```
From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thy self thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.
```

Training Text URL

http://raw.githubusercontent.com/brunoklein99/deep-learning-notes/master/shakespeare.txt

Shakespeare Poem

Task 1: Input Processing

1. Load text as whole
   - → Uppercase letters to lowercase
   - → End of line `\n` to space `' '`
2. Get `set` of characters
3. Convert character to one-hot
   - → size $D = |\texttt{set}|$
   - → Separate vector per character

Task 2: Sequence Encoding

1. Fixed sequence length $S = 20$
   - → Sample $\mathbf{X} \in \mathbb{R}^{S \times D}$
   - → Target $\mathbf{T} \in \mathbb{R}^{S \times D}$
2. Go through text sequentially
   - → Predict next character
   - → E.g.: $x = \texttt{abcde} \Rightarrow t = \texttt{bcdef}$
3. Handle beginning of text

Task 3: Dataset and Data Loader

1. Convert $\mathcal{X}$ and $\mathcal{T}$ to `torch.tensor`s
   - → `torch.utils.data.TensorDataset` and `torch.utils.data.DataLoader`

Network Implementation

Network Topology

- Fully-connected $\mathbf{W}^{(1)} \in \mathbf{R}^{K \times D}$
- Fully-connected $\mathbf{W}^{(2)} \in \mathbf{R}^{D \times K}$
- Recurrent $\mathbf{W}^{(r)} \in \mathbf{R}^{K \times K}$
- Non-linearity of your choice

Parallelization

- Batched implementation
  - $\rightarrow$ Possible through fixed context S

Forward Propagation – Training

- Initialize $\vec{h}^{\{0\}} = \vec{0}$
- For $s \in [1, S]$ do:
  1. $\vec{h}^{\{s\}} = \mathbf{W}^{(1)} \vec{x}^{\{s\}} + \mathbf{W}^{(r)} \vec{h}^{\{s-1\}}$
  2. $\vec{z}^{\{s\}} = \mathbf{W}^{(2)} \vec{h}^{\{s\}}$
- Return $\{\vec{z}^{\{s\}} \mid s \in [1, S]\}$

Forward Prediction – Testing

- Initialize $\vec{h}^{\{0\}} = \vec{0}$
- For $s \in [1, S']$ do:
  1. $\vec{h}^{\{s\}} = \mathbf{W}^{(1)} \vec{x}^{\{s\}} + \mathbf{W}^{(r)} \vec{h}^{\{s-1\}}$
- Return $\vec{z}^{\{S'\}} = \mathbf{W}^{(2)} \vec{h}^{\{S'\}}$

Network Implementation

Task 3: Network Implementation

1. Derive from `torch.nn.Module`
2. Constructor `__init__(self)`
3. Forward function `forward(self,x)`
4. Predict function `predict(self,x)`

Task 4: Categorical Cross-Entropy

1. Softmax $\vec{y}^{[c]} = S(\vec{z}^{[c]})$ (implicit)
2. Cross-entropy loss over time:

$$\mathcal{J}^{\mathrm{CE}} = -\frac{1}{S}\sum_{s=1}^{S}\sum_{o=1}^{O}\left(t_o^{\{s\}} - \log y_o^{\{s\}}\right)$$

Code

```
class RNN(torch.nn.Module):
  def __init__(self):
    ...

  def forward(self, x):
    ...

  def predict(self, x):
    ...

def loss(Z, T):
  ...
```

Network Training

Task 5: Training

1. Stochastic gradient descent
   → Select parameters B, η, μ
2. Monitor loss over epochs
3. Train for several epochs
   → Good results after few epochs
4. Save model after each epoch:
   → `torch.save(network.state_dict(), name)`

⇒ Load back via:
→ `network.load_state_dict(torch.load(name))`

Task 6: Text Generation

1. Define seeding text (`"moth"`)
2. Obtain sequence encoding
   → Variable length: $S' \ll S$
3. Use `predict` to get $\vec{z}^{\{S'\}}$
4. Append best predicted character to your text
5. Repeat steps 2-4 100 times
6. Print resulting text

Text Generation

Task 7: Random Sampled Text

Step 4: draw next character based on probability distribution $\vec{y}^{\{s'\}}$

Best Characters

```
tr -> "true my heart and eyes have erred, and to this false plague are they now transferr"
yo -> "you with my song. alack what poverty my muse brings forth, that having such a scop"
lo -> "love that well, which thou must love to any who for thy self art so unprovident. g"
ma -> "make the world away: let those whom nature hath not made for store, harsh. for i h"
mi -> "mine eyes have drawn thy shape, and thine for me are windows to my breast, where-t"
me -> "me that i am not thought to leap large and spart a beaule is impanelled a quest of"
```

Sampled Characters

```
tr -> "true soul canker of thy worth gives thee releasing: my bonds in thee are all despe"
yo -> "your broad main doth in this lirit that this shalt find thy monument, when tyrants"
lo -> "love not less, though less the show appear, that love is merchasent doth lie, o wh"
ma -> "make he day, the eyes (fore duteous) now converted are youn sweetest buds doth lov"
mi -> "minutes waste, these vacant leaves thy mind's imprint will bear, and of amist, you"
me -> "ment doth use, and every fair flower add the rank smell of time, and i a tyrant ha"
```