rnn_scratch_ascii

May 1, 2019

[1]:

```python
import numpy as np
import pyprind
import matplotlib.pyplot as plt
```

[2]:

```python
data_raw = open('tale_of_two_cities.txt', 'r').read()
data = ""
for i in data_raw:
    if(32 <= ord(i) < 126):
        data += i
    else:
        data += ' '
data_len = len(data)
vocab = list(set(data))
vocab.sort()
vocab_len = len(vocab)

print(data_len, vocab_len)
```

```
776697 81
```

[3]:

```python
vocab = [chr(i) for i in range(32,127)]
vocab_len = len(vocab)
print(vocab_len)
char_to_id = { ch:i for i,ch in enumerate(vocab) }
id_to_char = { i:ch for i,ch in enumerate(vocab) }
```

```
95
```

[4]:

```python
hidden_dim = 128
epochs = 100
seq_len = 25
learning_rate = 1e-5
```

[5]:

```python
wxh = np.random.normal(0,0.01,(hidden_dim, vocab_len))
whh = np.random.normal(0,0.01,(hidden_dim, hidden_dim))
wyh = np.random.normal(0,0.01,(vocab_len, hidden_dim))
bh = np.random.normal(0,0.01,(hidden_dim, 1))
```

```
by = np.random.normal(0,0.01,(vocab_len, 1))
hprev = np.zeros((hidden_dim, 1))
print(wxh.shape, whh.shape, wyh.shape, bh.shape, by.shape)
```

```
(128, 95) (128, 128) (95, 128) (128, 1) (95, 1)
```

[6]:

```
idx = 0
inp = [char_to_id[ch] for ch in data[idx:idx+seq_len]]
print(inp[-3])
xt = np.zeros((len(inp), vocab_len,1))
xt[np.arange(len(inp)), inp] = 1
print(xt[1].shape)
```

```
0
(95, 1)
```

[7]:

```
def fit_model(inp, labels):
    xt = np.zeros((len(inp), vocab_len,1))
    xt[np.arange(len(inp)), inp] = 1
    ht, zt, yt = {}, {}, {}

    global wxh, whh, wyh, bh, by, hprev

    ht[-1] = np.copy(hprev)
    loss = 0

    dwxh, dwhh,dwyh,dbh,dby = np.zeros_like(wxh), np.zeros_like(whh), np.
 ↪zeros_like(wyh), np.zeros_like(bh), np.zeros_like(by)
    dhnxt = np.zeros_like(ht[-1])
    for t in range(len(inp)):
        ht[t] = np.tanh(np.matmul(whh, ht[t-1]) + np.matmul(wxh, xt[t]) + bh)
        zt[t] = np.matmul(wyh, ht[t]) + by
        yt[t] = np.exp(zt[t]) / np.sum(np.exp(zt[t]))

        loss = loss - np.log(yt[t][labels[t]])

    hprev = ht[len(inp) - 1]

    for t in reversed(range(len(inp))):
        dzt = np.copy(yt[t])
        dzt[labels[t]] -= 1

        dwyh += np.matmul(dzt, np.transpose(ht[t]))
        dby += dzt

#         print(wyh.shape, yt[t].shape)
        dht = np.matmul(np.transpose(wyh), yt[t]) + dhnxt
        dat = dht * (1 - (ht[t] * ht[t]))

        dwxh += np.matmul(dat, np.transpose(xt[t]))
        dwhh += np.matmul(dht, np.transpose(ht[t]))
        dbh += dat

        for dparam in [dwxh, dwhh, dwyh, dbh, dby]:
            np.clip(dparam, -5, 5, out=dparam)
#         print(t)
    wxh = wxh - learning_rate* dwxh
    whh = whh - learning_rate* dwhh
    wyh = wyh - learning_rate* dwyh
    bh = bh - learning_rate * dbh
    by = by - learning_rate * dby
    return loss
```

[8]:

```
def predict_model(inp, length):
    global wxh, whh, wyh, bh, by, hprev

    ht, zt, yt = {}, {}, {}
    ht[-1] = np.copy(hprev)
    x = np.zeros((vocab_len, 1))
    x[inp] = 1
    output = []

    for t in range(length):
        ht[t] = np.tanh(np.matmul(whh, ht[t-1]) + np.matmul(wxh, x) + bh)
        zt[t] = np.matmul(wyh, ht[t]) + by
        yt[t] = np.exp(zt[t]) / np.sum(np.exp(zt[t]))
        pred_id = np.random.choice(range(vocab_len), p=yt[t].ravel())
        x = np.zeros((vocab_len, 1))
        x[pred_id] = 1
        output.append(pred_id)

    return output
```

[9]:

```
losses = []
for epoch in range(epochs):
    loss = 0
    bar = pyprind.ProgBar(data_len/seq_len)
    idx = 0
    while(idx + seq_len+1 <= data_len):
        inp = [char_to_id[ch] for ch in data[idx:idx+seq_len]]
        labels = [char_to_id[ch] for ch in data[idx+1:idx+seq_len+1]]

        batch_loss = fit_model(inp, labels)
        loss += batch_loss
        bar.update()
```