Assignment_5

November 7, 2023

[1]:
```
import torch
import numpy as np
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.autograd import Variable
```

[2]:
```
torch.manual_seed(1)
```

[2]:
```
<torch._C.Generator at 0x7f0c20046558>
```

[3]:
```
context_size = 2 # {w_i-2 ... w_i ... w_i+2}
embedding_dim = 10
```

[6]:
```
def make_context_vector(context, word_to_idx):
    idxs = [word_to_idx[w] for w in context]
    return torch.tensor(idxs, dtype=torch.long)

vocab = set(raw_text)
vocab_size = len(vocab)

word_to_idx = {word: i for i, word in enumerate(vocab)}
idx_to_word = {i: word for i, word in enumerate(vocab)}

data = []
```

[7]:
```
for i in range(2, len(raw_text) - 2):
    context = [raw_text[i-2], raw_text[i-1],
               raw_text[i+1], raw_text[i+2]]
    target = raw_text[i]
    data.append((context, target))
```

[8]:
```
class CBOW(nn.Module):

    def __init__(self, vocab_size, embedding_dim):
        super(CBOW, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.proj = nn.Linear(embedding_dim, 128)
```

```
        self.output = nn.Linear(128, vocab_size)

    def forward(self, inputs):
        embeds = sum(self.embeddings(inputs)).view(1, -1)
        out = F.relu(self.proj(embeds))
        out = self.output(out)
        nll_prob = F.log_softmax(out, dim=-1)
        return nll_prob
```

[9]:
```
model = CBOW(vocab_size, embedding_dim)
optimizer = optim.SGD(model.parameters(), lr=0.001)

losses = []
loss_function = nn.NLLLoss()
```

[10]:
```
for epoch in range(100):
    total_loss = 0
    for context, target in data:
        context_vector = make_context_vector(context, word_to_idx)

        # Remember PyTorch accumulates gradients; zero them out
        model.zero_grad()

        nll_prob = model(context_vector)
        loss = loss_function(nll_prob, Variable(torch.
 ↪tensor([word_to_idx[target]])))

        # backpropagation
        loss.backward()
        # update the parameters
        optimizer.step()

        total_loss += loss.item()

    losses.append(total_loss)

print(losses)
```

```
[234.64833641052246, 229.02283120155334, 223.69687223434448, 218.65443921089172,
213.88279151916504, 209.36257135868073, 205.0778249502182, 201.00420141220093,
197.11792755126953, 193.39784002304077, 189.82621347904205, 186.38809448480606,
183.06566685438156, 179.849287211895, 176.72991633415222, 173.69919109344482,
170.75373709201813, 167.88953095674515, 165.09919029474258, 162.37884879112244,
159.7275413274765, 157.14024704694748, 154.6124175786972, 152.1422671675682,
149.72463911771774, 147.36259347200394, 145.04607498645782, 142.77754575014114,
140.55291652679443, 138.3740772008896, 136.23530477285385, 134.14062649011612,
132.08306831121445, 130.06151181459427, 128.077851831913, 126.12879002094269,
124.2165829539299, 122.33734339475632, 120.48960447311401, 118.6736466884613,
```

```
116.88918733596802, 115.13450807332993, 113.40839678049088, 111.71138828992844,
110.04126644134521, 108.39663231372833, 106.78098630905151, 105.189443141222,
103.62212792038918, 102.08124953508377, 100.56518650054932, 99.07195484638214,
97.60222691297531, 96.15597409009933, 94.73007503151894, 93.32633802294731,
91.9446530342102, 90.58232283592224, 89.24050995707512, 87.91693022847176,
86.61449721455574, 85.33287325501442, 84.06760370731354, 82.8237484395504,
81.59993118047714, 80.39229837059975, 79.2053330540657, 78.03363573551178,
76.8850434422493, 75.75019910931587, 74.63456040620804, 73.53533735871315,
72.45483447611332, 71.38884049654007, 70.33991633355618, 69.30716578662395,
68.29007881879807, 67.28939564526081, 66.30360774695873, 65.33545026183128,
64.37992385029793, 63.44065023958683, 62.516592651605606, 61.60627177357674,
60.71115578711033, 59.8287869989872, 58.96280452609062, 58.11011841893196,
57.27053527534008, 56.4444250613451, 55.63238747417927, 54.83315482735634,
54.04673106968403, 53.27284777164459, 52.51110951602459, 51.76214957237244,
51.02479900419712, 50.299535021185875, 49.586836501955986, 48.885690093040466]
```

[11]:
```
print("**************************************************************************")

context = ['process.','Computational','are', 'abstract']
context_vector = make_context_vector(context, word_to_idx)
a = model(context_vector).data.numpy()
print('Raw text: {}\n'.format(' '.join(raw_text)))
print('Test Context: {}\n'.format(context))
max_idx = np.argmax(a)
print('Prediction: {}'.format(idx_to_word[max_idx]))
```

```
**************************************************************************
Raw text: We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers. As they
evolve, processes manipulate other abstract things called data. The evolution of
a process is directed by a pattern of rules called a program. People create
programs to direct processes. In effect, we conjure the spirits of the computer
with our spells.

Test Context: ['process.', 'Computational', 'are', 'abstract']

Prediction: processes
```

[12]:
```
context = ['processes','manipulate','abstract', 'things']
context_vector = make_context_vector(context, word_to_idx)
a = model(context_vector).data.numpy()
print('Raw text: {}\n'.format(' '.join(raw_text)))
print('Test Context: {}\n'.format(context))
max_idx = np.argmax(a)
print('Prediction: {}'.format(idx_to_word[max_idx]))
```

```
Raw text: We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers. As they
evolve, processes manipulate other abstract things called data. The evolution of
```

```
a process is directed by a pattern of rules called a program. People create
programs to direct processes. In effect, we conjure the spirits of the computer
with our spells.

Test Context: ['processes', 'manipulate', 'abstract', 'things']

Prediction: other
```

[5]:
```
raw_text = """We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers.
As they evolve, processes manipulate other abstract things called data.
The evolution of a process is directed by a pattern of rules
called a program. People create programs to direct processes. In effect,
we conjure the spirits of the computer with our spells.""".split()
```

[ ]:
```

```