skip_gram_p1

October 22, 2025

Design and implement a neural based network for generating word embedding for words in a document corpus

[4]:

```
import torch, torch.nn as nn, torch.optim as optim
from collections import Counter
import random

# Sample corpus
corpus = "I love deep learning and I love natural language processing".lower().
 ↪split()

# Build vocabulary
vocab = list(set(corpus))
word_to_ix = {w:i for i,w in enumerate(vocab)}
ix_to_word = {i:w for w,i in word_to_ix.items()}

# Generate skip-gram pairs
window = 2
pairs = []
for i, word in enumerate(corpus):
    for j in range(max(0,i-window), min(len(corpus),i+window+1)):
        if i != j:
            pairs.append((word, corpus[j]))

# Model
class SkipGram(nn.Module):
    def __init__(self, vocab_size, embed_dim):
        super().__init__()
        self.in_embed = nn.Embedding(vocab_size, embed_dim)
        self.out_embed = nn.Embedding(vocab_size, embed_dim)
    def forward(self, center, context):
        v = self.in_embed(center)
        u = self.out_embed(context)
        return torch.sum(v*u, dim=1)

# Training
model = SkipGram(len(vocab), 10)
optimizer = optim.Adam(model.parameters(), lr=0.01)
```

```python
loss_fn = nn.BCEWithLogitsLoss()

for epoch in range(200):
    total_loss = 0
    for center, context in random.sample(pairs, len(pairs)):
        c, o = torch.tensor([word_to_ix[center]]), torch.
 ↪tensor([word_to_ix[context]])
        neg = torch.randint(0, len(vocab), (3,))  # 3 negative samples
        pos_score = model(c, o)
        neg_score = model(c.repeat(3), neg)
        loss = -(torch.log(torch.sigmoid(pos_score)) + torch.sum(torch.
 ↪log(torch.sigmoid(-neg_score))))
        optimizer.zero_grad(); loss.backward(); optimizer.step()
        total_loss += loss.item()
    if epoch % 50 == 0: print(f"Epoch {epoch}, loss={total_loss:.4f}")

# Print embeddings
for w in vocab:
    print(w, model.in_embed.weight[word_to_ix[w]].detach().numpy())
```

```
Epoch 0, loss=141.2966
Epoch 50, loss=63.2674
Epoch 100, loss=65.8166
Epoch 150, loss=64.5950
processing [-0.67175657 -1.1896914   0.29552612  0.5835572   0.6773143
-1.4806708
  1.7789444  -1.7444867   0.5126831  -0.8025127 ]
language [ 0.38102868 -1.0789949  -0.1106344   3.0708969  -0.35358402 -0.6936631
 -0.13848643  0.13772745  1.0495036  -0.37585625]
natural [-0.4891889  -1.2395352   1.1793149   0.9915758   0.31499714 -0.09360732
  1.5140321   0.31396487 -0.44645607  0.35300404]
and [-0.11550919  0.1236926  -0.17019002 -1.1422676  -0.37963068 -2.1056952
  0.56239325  1.2905227  -1.1416632   1.3182212 ]
i [-0.4639511   0.2909461  -1.025208    0.9969221  -0.70202315 -1.2808903
 -0.36761773  0.01894578 -1.2637808   1.0312247 ]
love [-1.1097047  -0.5272871   0.02476619 -0.56234324 -0.21374612 -0.5176027
 -0.1710655   0.3747782  -1.3972955  -1.0978956 ]
learning [-1.1800041  -0.26299748 -0.14905263  0.14678738  0.79983693 -1.8778838
 -1.3556166   1.6296365   0.604754   -0.74682254]
deep [-1.0592626   0.21058401  1.9579767   0.21402329  0.7245933  -1.0484772
 -0.69280964 -0.25614667 -0.95637494  0.8193702 ]
```

[ ]:

```
```