explain

May 8, 2022

[1]:
```
import torch
from torch import nn
import torch.nn.functional as F
```

1 Copypaste EmbeddingDropout from awd-lstm repo

[2]:
```
# EmbeddingDropout from Jeremy Howard ULMfit
# https://github.com/fastai/fastai/blob/
 ↪5d45163c18f248f084af37fea39f9ab62083804a/fastai/text/models/awdlstm.py#L63
class EmbeddingDropout(nn.Module):
    """Apply dropout with probability `embed_p` to an embedding layer `emb`."""

    def __init__(self, emb, embed_p):
        super().__init__()
        self.emb = emb
        self.embed_p = embed_p

    @staticmethod
    def dropout_mask(x, sz, p):
        """Return a dropout mask of the same type as `x`, size `sz`,
        with probability `p` to cancel an element."""
        return x.new_empty(*sz).bernoulli_(1 - p).div_(1 - p)

    def forward(self, words, scale=None):
        if self.training and self.embed_p != 0:
            size = (self.emb.weight.size(0), 1)

            mask = self.dropout_mask(self.emb.weight.data, size, self.embed_p)

            masked_embed = self.emb.weight * mask
        else:
            masked_embed = self.emb.weight
        if scale:
            masked_embed.mul_(scale)
        return F.embedding(
            words,
```

```
            masked_embed,
            -1 if self.emb.padding_idx is None else self.emb.padding_idx,
            self.emb.max_norm,
            self.emb.norm_type,
            self.emb.scale_grad_by_freq,
            self.emb.sparse,
        )
```

[3]:
```
EMB_DIM = 4
VOCAB_SIZE = 10
DROP_PROB = 0.8
BATCH_SIZE = 2
SEQ_LEN = 8
```

[4]:
```
emb_awd = EmbeddingDropout(nn.Embedding(VOCAB_SIZE, EMB_DIM), DROP_PROB)

emb_dp = nn.Sequential(
    nn.Embedding(VOCAB_SIZE, EMB_DIM),
    nn.Dropout(DROP_PROB),
)
```

Embedding dropout from awd lstm randomly drops out the representation of the whole tokens

[5]:
```
emb_awd(torch.randint(low=0, high=VOCAB_SIZE, size=(BATCH_SIZE, SEQ_LEN)))
```

[5]:
```
tensor([[[-0.0000, -0.0000,  0.0000, -0.0000],
         [ 1.8552,  0.5323, -8.4268, -6.9215],
         [-5.7353,  2.5569,  5.5145, -1.1451],
         [-0.0000,  0.0000,  0.0000, -0.0000],
         [ 0.0000,  0.0000,  0.0000, -0.0000],
         [ 0.0000,  0.0000,  0.0000, -0.0000],
         [ 0.0000,  0.0000, -0.0000, -0.0000],
         [ 0.0000,  0.0000,  0.0000,  0.0000]],

        [[ 1.8552,  0.5323, -8.4268, -6.9215],
         [-0.0000,  0.0000,  0.0000, -0.0000],
         [ 0.0000, -0.0000,  0.0000, -0.0000],
         [ 0.0000,  0.0000,  0.0000,  0.0000],
         [ 0.0000,  0.0000,  0.0000,  0.0000],
         [ 0.0000,  0.0000,  0.0000,  0.0000],
         [ 0.0000,  0.0000,  0.0000,  0.0000],
         [ 1.8552,  0.5323, -8.4268, -6.9215]]], grad_fn=<EmbeddingBackward0>)
```

Embedding+dropout randomly drop out elements from tensor

[6]:
```
emb_dp(torch.randint(low=0, high=VOCAB_SIZE, size=(BATCH_SIZE, SEQ_LEN)))
```

```
[6]: tensor([[[-0.0000, -0.0000,  0.0000, -0.0000],
              [ 4.2689, -0.0000,  0.0000,  0.0000],
              [ 0.0000, -0.0000, -0.0000,  0.0000],
              [ 0.0000,  8.0344, -0.0000,  2.5740],
              [ 0.0000, -6.8144,  0.0000, -0.0000],
              [-0.0000,  0.0000, -0.0000,  0.0000],
              [ 0.0000, -0.0000,  0.0000, -0.0000],
              [ 0.0000, -0.0000,  0.0000, -0.0000]],

             [[-0.0000, -0.0000,  4.4746, -0.0000],
              [-0.0000,  0.0000, -0.0000,  0.0000],
              [ 0.0000,  0.0000, -0.0000,  2.5740],
              [ 0.0000,  0.0000,  0.0000, -0.5320],
              [-4.3530, -0.0000,  0.0000, -0.0000],
              [ 0.0000, -0.0000, -0.0000,  0.0000],
              [-0.0000, -0.0000,  4.4746, -0.0000],
              [ 0.1999,  0.0000, -2.3915,  0.0000]]], grad_fn=<MulBackward0>)
```