Introduction to Machine Learning

Day 15: Embeddings and Sequences

Joanna Bieri DATA301

Get out your handwritten notes!

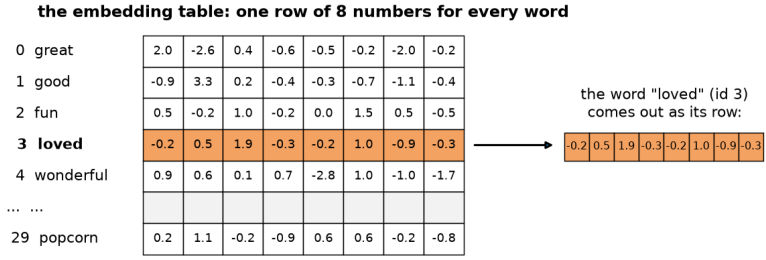
Mini Review - Made Up Movie Reviews

	REVIEW
0	a popcorn saw the awful popcorn it and on and
1	movie wonderful and the my plot was popcorn ending story
0	friday hated friends long ending on and my was ending

Each review is random filler words with **one feeling word** dropped in at a random spot. A positive feeling word means label 1, a negative one means label 0.

Mini Review - Embeddings

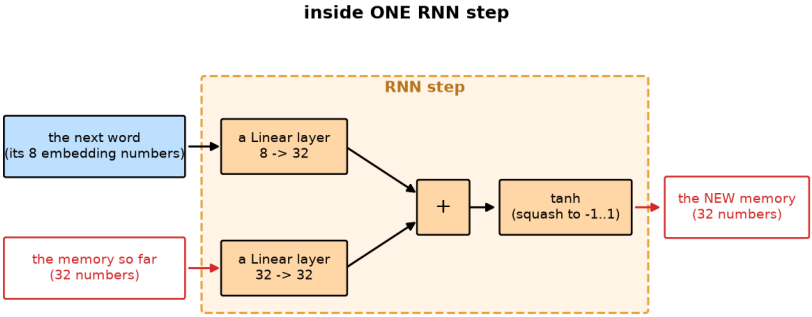
REVIEW



An **embedding** is a table with one row for each word in the vocabulary, and **those numbers are weights**.

Mini Review - Inside One RNN Step

REVIEW

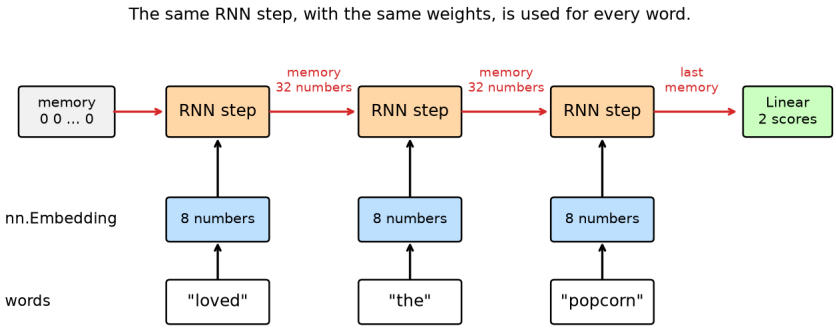


Two Linear layers, add them up, then an activation function. The new memory goes into the next step.

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$

Mini Review - Unrolling the RNN

REVIEW



There is only **ONE** RNN box, with one set of weights. We use it over and over, once for each word.

Mini Review - Our First nn.Module

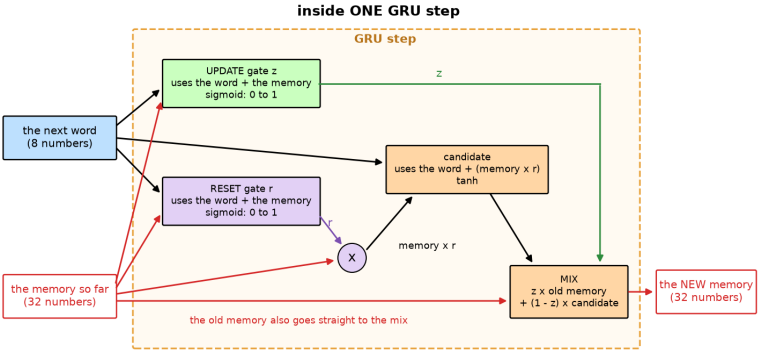

```
class ReviewNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.embed = nn.Embedding(vocab_size, 8)
        self.memory = nn.RNN(8, 32, batch_first=True)
        self.out = nn.Linear(32, 2)

    def forward(self, X):
        vectors = self.embed(X)
        all_hidden, last_hidden = self.memory(vectors)
        scores = self.out(last_hidden[0])
        return scores
```

Mini Review - Short and Long Reviews

reviews	RNN, validation accuracy by epoch
10 words	1.0, 1.0, 1.0, 1.0, 1.0
100 words	0.504, 0.476, 0.521, 0.506, 0.493

Same network and same training. The only thing we changed is the length of the reviews.

Mini Review - The GRU



Each colored box maps the word (8 to 32 numbers) and the memory (32 to 32 numbers) with its own weights, adds them, then applies an activation.

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t$$

Words to Numbers

ASK

A vocabulary has 5,000 words.

"apple" is word 3 and "zebra" is

word 4,812. Why not hand the

network those numbers?

Hint. Who decided the order of the vocabulary list?

Words to Numbers

ANSWER

Why not hand the network the word ids, like "apple" = 3 and "zebra" = 4,812, as numbers?

- The ids are only names. Somebody made up the order of the list.
- The network would treat “zebra” as about 1,600 times “apple”, and that does not mean anything.
- An embedding gives every word its own row of learned numbers.

Words to Numbers

PREDICT

```
embed = nn.Embedding(2000, 16)
```

How many numbers does this embedding have to learn? With one-hot encoding instead, how many numbers would each word be?

Hint. One row for every word in the vocabulary.

Words to Numbers

ANSWER

How many numbers does nn.Embedding(2000, 16) learn? How long is a one-hot row for this vocabulary?

```
2000 * 16 = 32,000 numbers to learn
```

Each word is 16 learned numbers. One-hot would make every word a row of 2,000 numbers, all 0 except one 1.

The Recurrent Neural Network

PREDICT

```
rnn = nn.RNN(16, 64, batch_first=True)
vectors = torch.zeros(200, 40, 16)  # 200 reviews, 40 words

all_hidden, last_hidden = rnn(vectors)
print(all_hidden.shape)
print(last_hidden.shape)
```

With your group: what two shapes come out?

Hint. all_hidden is the memory after every word.
last_hidden is the memory after the last word, with one extra number in front.

The Recurrent Neural Network

ANSWER

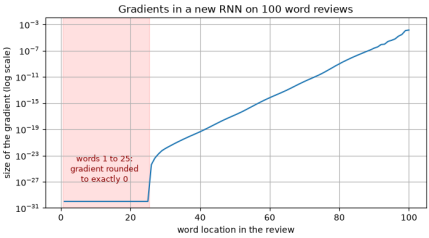
nn.RNN(16, 64) on vectors of shape (200, 40, 16). What are all_hidden.shape and last_hidden.shape?

```
torch.Size([200, 40, 64])
torch.Size([1, 200, 64])
```

A memory of 64 numbers after every word, for every review. Then one memory per review after the last word. The 1 in front is why ReviewNet uses last_hidden[0].

Why the RNN Fails

ASK



The size of the gradient for every word in a 100 word review, for a new RNN. Why can it not learn the long reviews?

Hint. Most feeling words are far from the end. What gradient do they get?

Why the RNN Fails

ANSWER

The gradient for every word in a 100 word review. Why can the RNN not learn the long reviews?

- The last word gets about 0.00014. For words 1 to 25 the gradient is rounded to exactly 0.
- The gradient gets multiplied by a small number at every step back, just like the deep networks on Day 11.
- So the RNN only learns from the last few words. A feeling word far back gives it almost nothing to learn from.

Why the RNN Fails

PREDICT

Every step back multiplies

the gradient by 0.8.

How big is it after 10 steps back?

After 40 steps back?

Hint. Start with a gradient of 1. Use a calculator!

Why the RNN Fails

ANSWER

Every step back multiplies the gradient by 0.8. How big is it after 10 steps? After 40?

```
0.8 ** 10 = 0.107        about 10 times smaller
0.8 ** 40 = 0.000133     about 7,500 times smaller
```

Each step keeps 80 percent of the gradient, and after 40 steps only about 0.0001 is left. A feeling word 40 words from the end barely changes the weights.

The GRU

ASK

The GRU has a second gate,

the reset gate r_t. What does it decide

that the update gate z_t does not?

Hint. Look at the GRU picture. Where does r_t go, and where does z_t go?

The GRU

ANSWER

What does the reset gate r_t decide that the update gate z_t does not?

- The reset gate decides how much of the old memory to use when the GRU MAKES the new candidate.
- The update gate decides how much of the new candidate goes in.
- With r_t close to 0 the candidate comes from the new word alone, and the GRU starts fresh.

The GRU

PREDICT

```
nn.RNN(8, 64)    4,736 numbers to learn
nn.GRU(8, 64)    ?
```

How many numbers does the GRU layer have to learn? Why?

Hint. In the notes, nn.RNN(8, 32) had 1,344 and the GRU memory layer was exactly three times as big.

The GRU

ANSWER

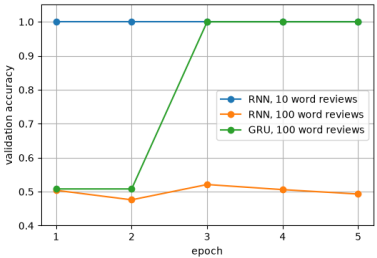
nn.RNN(8, 64) has 4,736 numbers to learn. How many does nn.GRU(8, 64) have, and why?

```
3 * 4,736 = 14,208
```

The candidate, the update gate and the reset gate each have their own weights, the same size as one RNN step.

So Why Do We Need Anything Else?

ASK



The GRU got every long review right. So why is the rest of the semester about attention?

Hint. How many numbers hold the whole review at the end? In what order are the words read?

So Why Do We Need Anything Else?

ANSWER

The GRU got every long review right. So why do we need attention?

- The whole review has to fit in one memory of 32 numbers. A real paragraph has a lot more to keep.
- It reads one word at a time, so step 50 can not start until step 49 is done. That is slow on long text.
- Attention lets the network look back at every word directly. That is Day 16!

Questions?