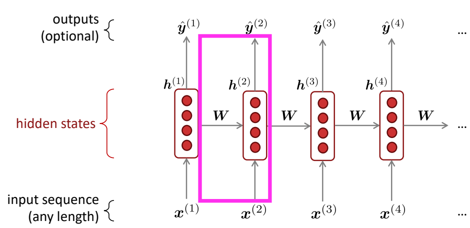
Recurrent Neural Networks (RNN)

A family of neural architectures

Core idea: Apply the same weights W *repeatedly*



A Simple RNN Language Model

output distribution

$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$

hidden states

$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$

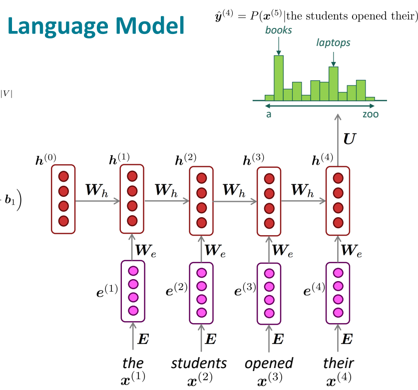
$h^{(0)}$ is the initial hidden state

word embeddings

$e^{(t)} = E x^{(t)}$

words / one-hot vectors

$x^{(t)} \in \mathbb{R}^{|V|}$



Note: this input sequence could be much longer now!

RNN Language Models

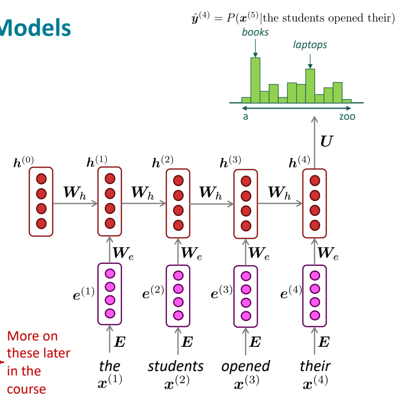
RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

More on these later in the course



Training an RNN Language Model

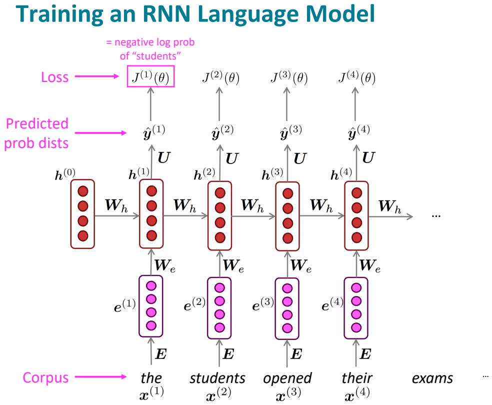
- Get a big corpus of text which is a sequence of words $x^{(1)}, \ldots, x^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{y}^{(t)}$ *for every step t.*
  - i.e. predict probability dist of *every word*, given words so far
- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{y}^{(t)}$, and the true next word $y^{(t)}$ (one-hot for $x^{(t+1)}$):

$$J^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = -\sum_{w \in V} y_w^{(t)} \log \hat{y}_w^{(t)} = -\log \hat{y}_{x_{t+1}}^{(t)}$$

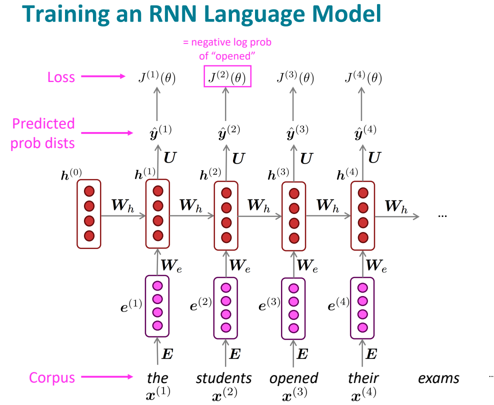
- Average this to get overall loss for entire training set:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{y}_{x_{t+1}}^{(t)}$$

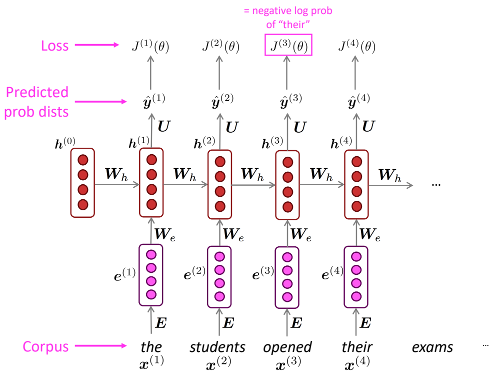
Training an RNN Language Model



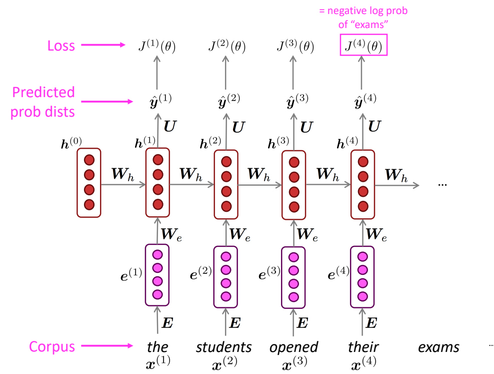
Training an RNN Language Model



Training an RNN Language Model

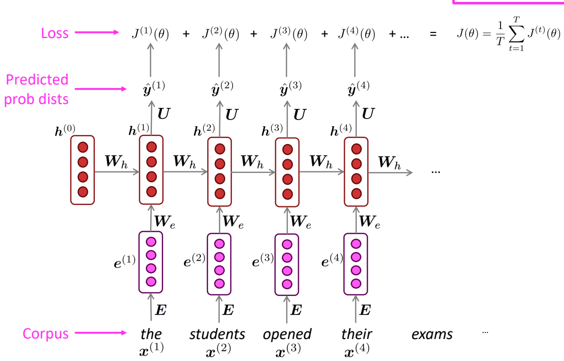


Training an RNN Language Model



Training an RNN Language Model

"Teacher forcing"



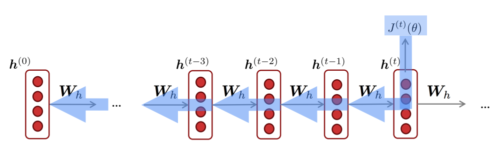
Training a RNN Language Model

- However: Computing loss and gradients across entire corpus $x^{(1)}, \ldots, x^{(T)}$ is too expensive!

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta)$$

- In practice, consider $x^{(1)}, \ldots, x^{(T)}$ as a sentence (or a document)
- Recall: Stochastic Gradient Descent allows us to compute loss and gradients for small chunk of data, and update.
- Compute loss $J(\theta)$ for a sentence (actually, a batch of sentences), compute gradients and update weights. Repeat.

Backpropagation for RNNs



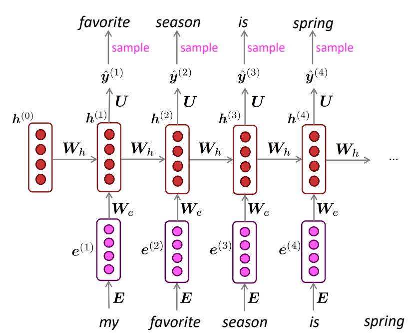
$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \boxed{\sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\right|_{(i)}}$$

Question: How do we calculate this?

Answer: Backpropagate over timesteps $i=t,\ldots,0$, summing gradients as you go.
This algorithm is called **"backpropagation through time"**
[Werbos, P.G., 1988, *Neural Networks* **1**, and others]

Generating text with a RNN Language Model

Just like a n-gram Language Model, you can use a RNN Language Model to generate text by repeated sampling. Sampled output becomes next step's input.



Generating text with an RNN Language Model

Let's have some fun!

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on Obama speeches:

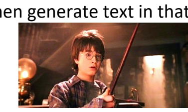
The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.

Source: https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

Generating text with an RNN Language Model

Let's have some fun!

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on *Harry Potter*:



"Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"

"No idea," said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry's shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn't felt it seemed. He reached the teams too.

Source: https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

Generating text with an RNN Language Model

Let's have some fun!

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on recipes:

```
    Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
    Yield: 6 Servings

    2 tb Parmesan cheese -- chopped
    1 c  Coconut milk
    3    Eggs, beaten

 Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer
 until firm. Serve hot in bodied fresh, mustard, orange and cheese.

 Combine the cheese and salt together the dough in a large skillet; add the ingredients
 and stir in the chocolate and pepper.
```

Source: https://gist.github.com/nylki/1efbaa36635956d35bcc

Generating text with a RNN Language Model

Let's have some fun!

- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on paint color names:

Ghasty Pink 231 137 165	Sand Dan 201 172 143
Power Gray 151 124 112	Grade Bat 48 94 83
Navel Tan 199 173 140	Light Of Blast 175 150 147
Bock Coe White 221 215 236	Grass Bat 176 99 108
Horble Gray 178 181 196	Sindis Poop 204 205 194
Homestar Brown 133 104 85	Dope 219 209 179
Snader Brown 144 106 74	Testing 156 101 106
Golder Craam 237 217 177	Stoner Blue 152 165 159
Hurky White 232 223 215	Burble Simp 226 181 132
Burf Pink 223 173 179	Stanky Bean 197 162 171
Rose Hork 230 215 198	Turdly 190 164 116

This is an example of a character-level RNN-LM (predicts what character comes next)

Source: http://aiweirdness.com/post/160776374467/new-paint-colors-invented-by-neural-network