Today: gradient descent ⇒ cross-entropy loss

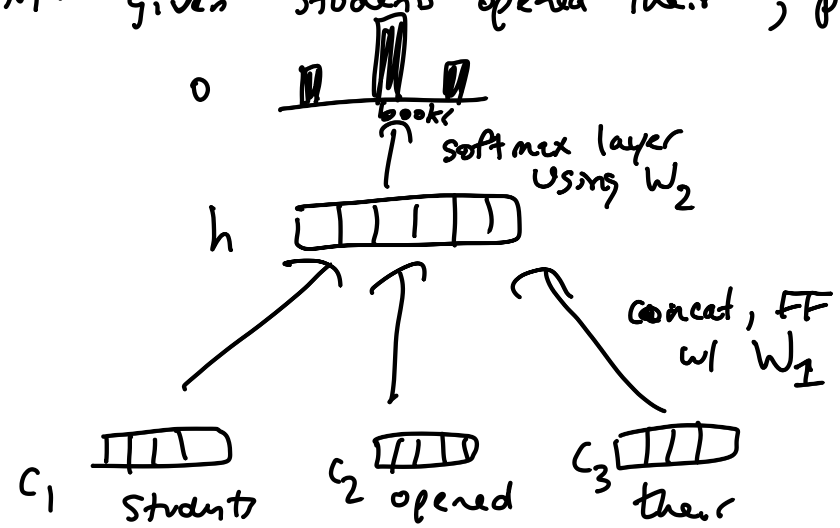
backpropagation

↳ FF / concat

↳ RNN (recurrent NN)

} single neuron

NLM: given "students opened their", predict "books"



params: $\underbrace{c_1, c_2, c_3, W_1, W_2}_{\theta}$

$$h = f(W_1 [c_1 ; c_2 ; c_3])$$

↳ notation for concat

$$o = \text{softmax}(W_2 h)$$

---

how do we <u>train</u> this model?

↳ how do we adjust our model parameters θ to make better predictions of the next word?

↳ GRADIENT DESCENT

1. define loss function $L(\theta)$ that tells us how bad the model is currently doing at predicting the next word

↳ ideally smooth / differentiable
↳ cross-entropy loss

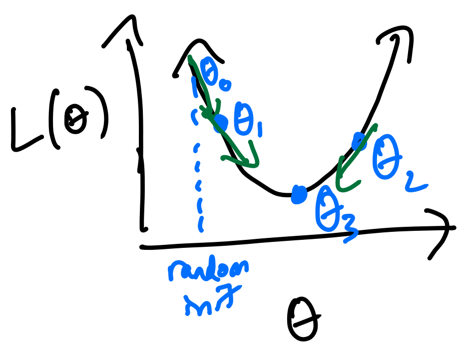
2. Given $L(\theta)$, we compute the gradient of L with respect to θ.
   ↳ gradient gives us the direction of steepest ascent of L
   ↳ same dimensionality as θ
   ↳ for each parameter j in θ, it tells you how much L would increase if you increase j by a very small amount

3. given gradient $\frac{dL}{d\theta}$, we take a step in the direction of the negative gradient, thus minimizes L

learning rate, controls step size

$$\theta_{new} = \theta_{OLD} - \eta \frac{dL}{d\theta}$$

↳ gradient



important hyperparameters:
- learning rate η
- batch size: how many training examples do you use to estimate $\frac{dL}{d\theta}$ before taking a step

Simple example:

$$x \xrightarrow{w_1} h \xrightarrow{w_2} o$$

inputs: (x, y) e.g. $(5, 4.3)$

$$\left.\begin{aligned} h &= \tanh(w_1 x) \\ o &= \tanh(w_2 h) \end{aligned}\right\}$$

1. compute loss fn

$$\left\{ L = \frac{1}{2}(y - o)^2 \right\}$$

square loss / L2 loss
good for regression problems

y ↳ target

o → model's prediction

2. compute gradient:

$$\frac{dL}{d\theta} : \frac{dL}{dw_1}, \frac{dL}{dw_2} \quad (2 \text{ params})$$

important: chain rule of calculus

$$\frac{d}{dx} g(f(x)) = \frac{dg}{df} \cdot \frac{df}{dx}$$

$$L = \frac{1}{2}(y - o)^2$$

$$o = \tanh(a)$$

$$a = w_2 h$$

$$h = \tanh(b)$$

$$b = w_1 x$$

let's make intermediate vars
$a = w_2 h$, $b = w_1 x$

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$$

$$\frac{dL}{dw_2} = \frac{dL}{do} \cdot \frac{do}{da} \cdot \frac{da}{dw_2}$$

$$-(y - o) \cdot (1 - o^2) \cdot h$$

$$\frac{dL}{dw_1} = \frac{dL}{do} \cdot \frac{do}{da} \cdot \frac{da}{dh} \cdot \frac{dh}{db} \cdot \frac{db}{dw_1}$$

backpropagation: chain rule of calculus + caching prev. computed derivatives

3. update params

$$w_{2_{new}} = w_{2_{OLD}} - \eta \frac{dL}{dw_2} \quad , \quad w_{1_{new}} = w_{1_{OLD}} - \eta \frac{dL}{dw_1}$$

---

what loss fn is used in LM?

↳ cross-entropy loss, generally useful for any classification task

$\underbrace{\text{students opened their}}_{\text{input}} \Rightarrow \underbrace{\text{books}}_{\text{target}}$, $|V|$ labels

goal:

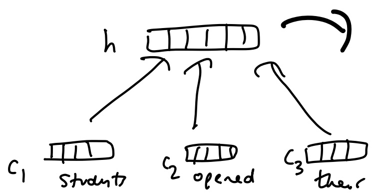
maximize $p(\text{books} \mid \text{"students opened their"})$

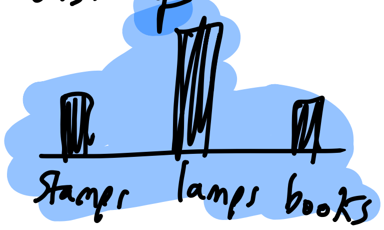
Minimize negative log probability of "books"

$$L = -\log\left(p\left(\text{books} \mid \text{"students opened their"}\right)\right)$$

why "cross-entropy" loss?

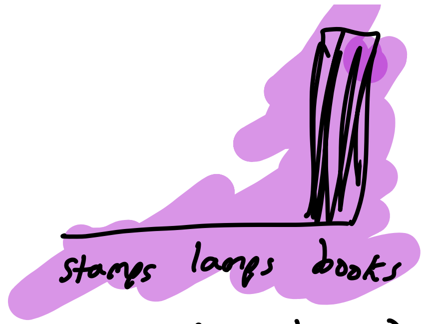
model's predicted dist. q





data distribution p:

... students opened their



$p(\text{books} \mid \ldots) = 1.0$

defn of cross entropy between p and q is:

$$-\sum_{w \in V} p(w) \log q(w)$$

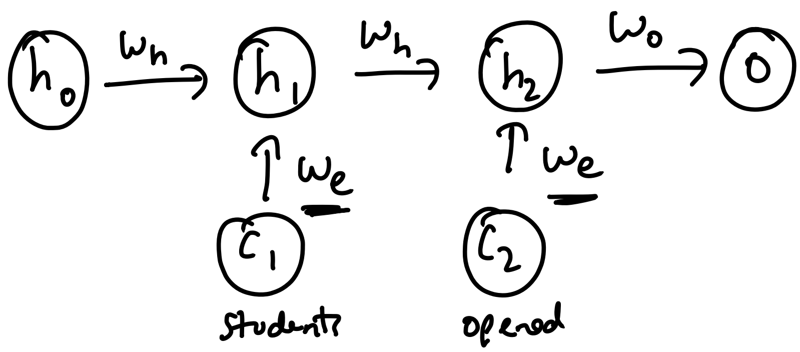
↳ 1 when w = books
0 for every other w

$$= -\log q(\text{books} \mid \text{Students opened their})$$

neg. log prob of correct word

---

recurrent neural networks:



params:
W_h, W_e, W_o, c_1, c_2

$$L = \frac{1}{2}(y - o)^2$$

$$o = W_o h_2$$

$$h_2 = \tanh(W_e c_2 + W_h h_1)$$

→ a

→ b

$$h_1 = \tanh(w_e c_1 + w_h h_0)$$

$$\frac{dL}{dw_o} = \frac{dL}{do} \cdot \frac{do}{dw_o} = -(y-o) \cdot h_2$$

$$\frac{dL}{dc_2} = \frac{dL}{do} \cdot \frac{do}{dh_2} \cdot \frac{dh_2}{da} \cdot \frac{da}{dc_2} = -(y-o) \cdot w_o \cdot (1-h_2^2) \cdot w_e$$

$\frac{dL}{dw_e}$ and $\frac{dL}{dw_h}$ are trickier b/c they are used at multiple timesteps in the network

↳ backprop thru time allows us to compute these by summing contributions from diff. time steps

$$\frac{dL}{dw_e} = \underline{\frac{dL}{do} \cdot \frac{do}{dh_2} \cdot \frac{dh_2}{da}} \cdot \frac{da}{dw_e} + \left[\underline{\frac{dL}{do} \cdot \frac{do}{dh_2} \cdot \frac{dh_2}{da}} \cdot \frac{da}{dh_1} \cdot \frac{dh_1}{db} \cdot \frac{db}{dw_e} \right.$$

we can accumulate these $\frac{dL}{dw_e}$, $\frac{dL}{dw_h}$ as we step back thru time

vanishing gradient problem: these gradient contributions from faraway steps go to zero