Today: gradient descent $\Rightarrow$ cross-entropy loss
backpropagation

$\mathcal{L}$ FF/concat
$\mathcal{L}$ RNN (recurrent NN)

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

$\mathbf{h} = f(W_1 \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix})$

$\mathbf{\theta} = \text{softmax}(W_2 \mathbf{h})$

how do we train this model?

$\mathcal{L}$ how do we adjust our model parameters $\mathbf{\theta}$
to make better predictions of the next word?

$\mathcal{L}$ GRADIENT DESCENT

1. define loss function $\mathcal{L}(\mathbf{\theta})$ that tells us
   how bad the model is currently doing at predicting
   the next word
2. Given $L(\theta)$, we compute the gradient of $L$ with respect to $\Theta$.
   - The gradient gives us the direction of steepest ascent of $L$.
   - It has the same dimensionality as $\Theta$.
   - For each parameter $j$ in $\Theta$, 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 minimizing $L$.
   $$\Theta_{new} = \Theta_{old} - \eta \frac{dL}{d\Theta}$$
   Learning rate $\eta$ controls step size.

Important hyperparameters:
- Learning rate $\eta$
- Batch size: how many training examples do you use to estimate $\frac{dL}{d\Theta}$ before taking a step.
Simple example:

\[ \begin{align*}
\times w_2 & \rightarrow h \rightarrow o \\
\end{align*} \]

Inputs: \((x, y)\) e.g. \((5, 4.3)\)

\[ h = \tanh(w_1x) \]

\[ o = \tanh(w_2h) \]

1. Compute loss fn \(\ell\):

\[ \ell = \frac{1}{2} (y-o)^2 \]

\(\frac{\text{Square loss / L2 loss}}{\text{good for regression problems}}\)

\(L\) \(\rightarrow\) \(\text{target}\) \(\rightarrow\) \(\text{model's prediction}\)

2. Compute gradient:

\[ \frac{d\ell}{d\theta} : \frac{d\ell}{dw_1}, \frac{d\ell}{dw_2} \quad (2 \text{ params}) \]

Important: chain rule of calculus

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

Let's make intermediate vars

\(a = w_2h, \ b = w_1x\)

\[ \frac{d\ell}{dw_2} = \frac{d\ell}{do} \cdot \frac{do}{da} \cdot \frac{da}{dw_2} \]

\[ -(y-o) \cdot (1-o^2) \cdot h \]
$$\frac{dL}{dw_1} = \frac{dL}{da} \cdot \frac{da}{dh} \cdot \frac{dh}{db} \cdot \frac{db}{dw_2}$$

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

3. update params

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

---

**what loss \(\mathcal{L}\) is used in LM?**

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

Students opened their \(\rightarrow\) books

---

**goal:**

maximize \(p(\text{books} | \text{students opened their})\)

minimize negative log probability of "books"

\[\mathcal{L} = -\log p(\text{books} | \text{students opened their})\]

why "cross-entropy" loss?

model's predicted dist. \(p\)
data distribution $p$: 

... students opened their

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

defn of cross entropy between $p$ and $q$ is:

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

- 1 when $w = \text{books}$
- 0 for every other $w$

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

neg. log prob of correct word

---

Recurrent Neural Networks:

- $h_0$ to $h_1$ to $h_2$ to $o$
- Input to hidden to hidden to output

- $w_h$, $w_e$, $w_o$, $c_1$, $c_2$
- Params:

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

$$o = w_o h_2$$

$$h_2 = \tanh(w_e c_2 + w_h h_1)$$

$$b$$
\[ h_1 = tanh \left( w_e c_i + w_h h_o \right) \]

\[ \frac{dL}{dw_0} = \frac{dL}{d_0} \cdot \frac{d_0}{dw_0} = -(y_o) \cdot h_2 \]

\[ \frac{dL}{dc_2} = \frac{dL}{d_0} \cdot \frac{d_0}{dh_2} \cdot \frac{dh_2}{da} \cdot \frac{da}{dc_2} = -(y_o) \cdot w_0 \cdot (1-h_2^2) \cdot w_e \]

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

\[ \text{by backprop thru time allows us to compute these by summing contributions from diff. time steps} \]

\[ \frac{dL}{dw_e} = \frac{dL}{d_0} \cdot \frac{d_0}{dh_2} \cdot \frac{dh_2}{da} \cdot \frac{da}{dw_e} + \frac{dL}{d_0} \cdot \frac{d_0}{dh_2} \cdot \frac{dh_2}{da} \cdot \frac{da}{dh_i} \cdot \frac{dh_i}{db} \cdot \frac{db}{dw_e} \]

we can accumulate these \( \frac{dL}{dw_e} \) as we step back thru time

\[ \text{Vanishing gradient problem: these gradient contributions from faraway steps go to zero} \]