1. Recall from last time, key abstraction:

\[ h(\theta) = g(h^{\theta-1}, x) \]

Loss \( L = f(h^L) \)

Abstractly: \( \frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial h^L} \cdot \frac{\partial h^L}{\partial \theta} \)

Concretely:

\( \frac{\partial L}{\partial \theta_{\text{net}}} = \frac{\partial L}{\partial h^L} \cdot \frac{\partial h^L}{\partial \theta_{\text{net}}} \)
Multi-Layer Perceptrons / Fully-Connected Layer / Inner-Product Layer

Another way of drawing this

Forward Pass:

\[ h_i^l = \sum_{j=1}^{L} w_{ij} h_j \]

= \[ w_i^T h \]

Let's vectorize some more

\[ h^l = w^T h^{l-1} \]

So all of F-PASS = 1 matrix-mull
Why did we write the as matrix mult?

1. Because it's cooler!
2. Gradients, my dear Watson, Gradients...

\[ h^0 = W^0 h^{L-1} \]

\[ \frac{\partial L}{\partial h^0} = W^0 \]  [Now, isn't that beautiful?]

\[ \frac{\partial L}{\partial h^{L-1}} = h^L \]

What about \( \frac{\partial L}{\partial h^i} \)?

Simple \[ h^i = W^i h^{i+1} \]

\[ \frac{\partial L}{\partial h^i} = \frac{\partial L}{\partial h^{i+1}} \frac{\partial h^{i+1}}{\partial h^i} = \frac{\partial L}{\partial h^{i+1}} h^i \]

Now put it all together:

\[ \frac{\partial L}{\partial h^{L-1}} = \frac{\partial L}{\partial h^L} \frac{\partial h^L}{\partial h^{L-1}} = \frac{\partial L}{\partial h^L} h^{L-1} \]

Very nice

\[ \frac{\partial L}{\partial h^i} = \frac{\partial L}{\partial h^{i+1}} \frac{\partial h^{i+1}}{\partial h^i} = \frac{\partial L}{\partial h^{i+1}} h^i \]

Self multiply = gradient

Right multiply = output

So B-PASS \( \Delta = 1 \) matrix mult

\[ \text{Left multiply} = \text{gradient} \]

\[ \text{Scalar} = I \times e \]
So \( w_i^{(e)} = w_i^{(e)} - \eta \frac{\partial L}{\partial w_i^{(e)}} \)

where
\[
\frac{\partial L}{\partial w_i^{(e)}} = \left[ \begin{array}{c}
\frac{\partial L}{\partial h_1} \\
\vdots \\
\frac{\partial L}{\partial h_n}
\end{array} \right] = \left[ \begin{array}{c}
\frac{\partial L}{\partial h_1} \\
\vdots \\
\frac{\partial L}{\partial h_n}
\end{array} \right] \cdot \text{h'}
\]

One more intuition:

Row in \( W^{(e)} \) matrix

(3) ReLU Layer

\[ h_i^{(e+1)} = \max\{0, h_i^e\} \]

F-PASS

B-PASS

[No params] \[ \frac{\partial h_i^e}{\partial h_i} = \begin{cases} 
+1 & \text{if } h_i^e > 0 \\
0 & \text{else} 
\end{cases} \]

\[ \frac{\partial L}{\partial h_i^e} = \frac{\partial L}{\partial h_i^{(e+1)}} \cdot I \{ h_i^e > 0 \} \]
Recall: Convolutions! [Recall that nightmare from signal processing class]

Let's start with pure math then move to math → CS → programming

Pure Math

Let $y(t)$, $x(t)$, $w(t)$ be continuous-time signals

1-D Convolution

$$y(t) = (x * w)(t)$$

$$\Rightarrow y(t) = \int_{a=-\infty}^{a=\infty} x(a)w(t-a) \, da$$

Intuition:

- Compute $y(t)$
- Flip: $w(a) \rightarrow w(-a)$
- Shift: $w(-a) \rightarrow w(-(a-t))$

Compute $\int_{a=-\infty}^{a=\infty} x(a)w(-(a-t)) \, da$ of product

Rinse & Repeat $\forall t$

BTW: $y(t) = \int x(a)w(t-a) \, da = \int x(t-a)w(a) \, da$ [Commutative]
Similarly 2D convolution:

\[ y(t_1, t_2) = \int_0^\infty \int_{-\infty}^\infty x(t_1, t_2) w(a-t, b-t) \, da \, db \]

Now, let's move to CS. There is no infinite precision! Let's go discrete:

\[ y[a, c] = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} x[a, c] w[a-a, b-c] \]

No big memory either. Has to be finite matrix:

\[ y[a, c] = \sum_{a=-\lfloor \frac{H}{2} \rfloor}^{\lfloor \frac{H}{2} \rfloor} \sum_{b=-\lfloor \frac{W}{2} \rfloor}^{\lfloor \frac{W}{2} \rfloor} x[a, c] w[a-a, b-c] \]

Let's move to programming. Arrays don't index with negative numbers.

And we don't really care about commutivity!
What we actually implement:

\[
y[a, c] = \sum_{a=0}^{k_{1}-1} \sum_{b=0}^{k_{2}-1} x[a+a, c+b] \cdot w[a, b]
\]

5) CNNs / ConvNets / Convolutional Neural Nets

Recall MLP

\[
h_i = \sum_{j=1}^{C_i} W_{ij} \cdot h_j
\]

scalar product

CNN [Make each neuron a 2D matrix]

Now a matrix fitter on every edge

All weights at output until i
CNN - F-Pass

Compose:

\[ h_i^l = \sum_{j=1}^{C_l} W_{ij}^{(l)} \cdot h_j^{l-1} \]  
\[ \text{scalar product} \]

\[ h_i^l = \sum_{j=1}^{C_l} W_{ij} \ast h_j^{l-1} \]  
\[ \text{matrix} \]

\[ = \sum_{j=1}^{C_l} \sum_{a=0}^{k_2-1} \sum_{b=0}^{k_1-1} h_j^{l-1} (a+b) + a + b \cdot W_{ij}^{(l)} [a, b] \]  
\[ \text{2D conv} \]

\[ \text{Input channels} \]

CNN - B-Pass

→ This is going to be a notation nightmare

→ So let's simplify thing a bit

→ Say \( C_1 = 1 \) \( C_2 = 1 \) [we can always generalize its just an extra sum]

→ And let's not over-load \( h \)

-> Input \( X \) \( W \) Kernel \( Y \) Output

\[ \text{Assume same size for simplicity?} \]