Overview Today:

- From one-layer to multi layer neural networks!
- Backprop **(last bit of heavy math)**
- Different descriptions and viewpoints of backprop
- Project Tips

Announcement:

- Hint for PSet1: Understand math and dimensionality, then check them with breakpoints or print statements:

```python

def softmaxCostAndGradient(predicted, target, outputVectors, dataset):
    """ Softmax cost function for word2vec models """
```

```python
    ### YOUR CODE HERE

    print "v_hat", predicted.shape
    print "expected", target
    print "U", outputVectors.shape

    assert False

```

Why go through all of these derivatives?

- Actual understanding of math behind deep learning

- Backprop can be an imperfect abstraction, e.g.
  - During optimization issues (e.g. vanishing gradients)

- Enables you to debug models, think of and implement completely new models

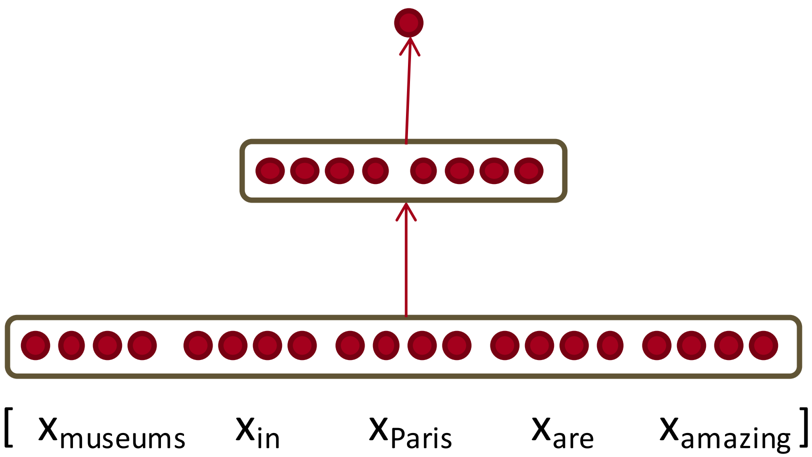
Explanation #1 for backprop

Remember: Window-based Neural Net

Computing a window's score with a 3-layer neural net: s = *score*(museums in Paris are amazing)

$$s = U^T f(Wx+b) \qquad x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8\times 20}, U \in \mathbb{R}^{8\times 1}$$

$$\begin{aligned} s &= U^T a \\ a &= f(z) \\ z &= Wx + b \end{aligned}$$



Putting all gradients together:

- Remember: Full objective function for each window was:

$$\boxed{J = \max(0, 1 - s + s_c)}$$

$$s = U^T f(Wx + b)$$
$$s_c = U^T f(Wx_c + b)$$

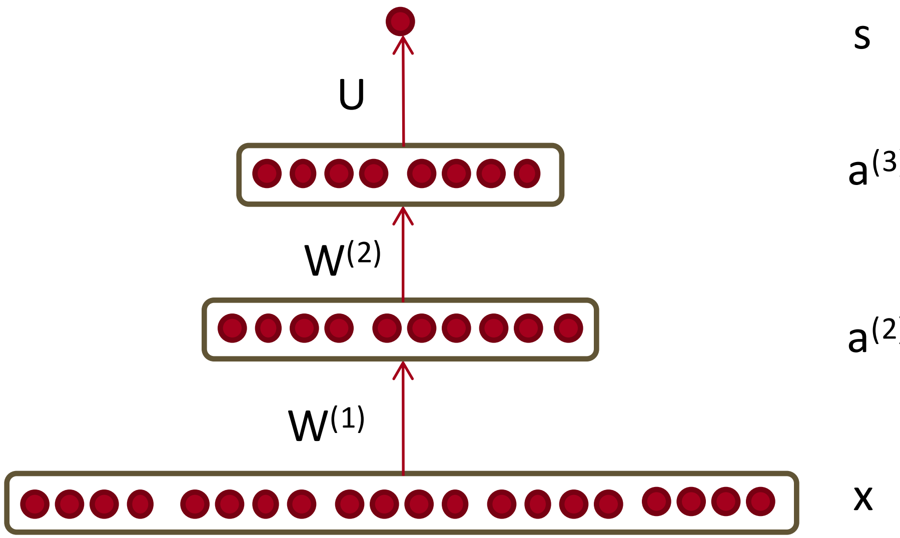
- For example: (sub)gradient for U:

$$\frac{\partial J}{\partial U} = 1\{1 - s + s_c > 0\}\left(-f(Wx + b) + f(Wx_c + b)\right)$$
$$\frac{\partial J}{\partial U} = 1\{1 - s + s_c > 0\}\left(-a + a_c\right)$$

Two layer neural nets and full backprop

- Let's look at a 2 layer neural network
- Same window definition for x
- Same scoring function
- 2 hidden layers (carefully define superscripts now!)

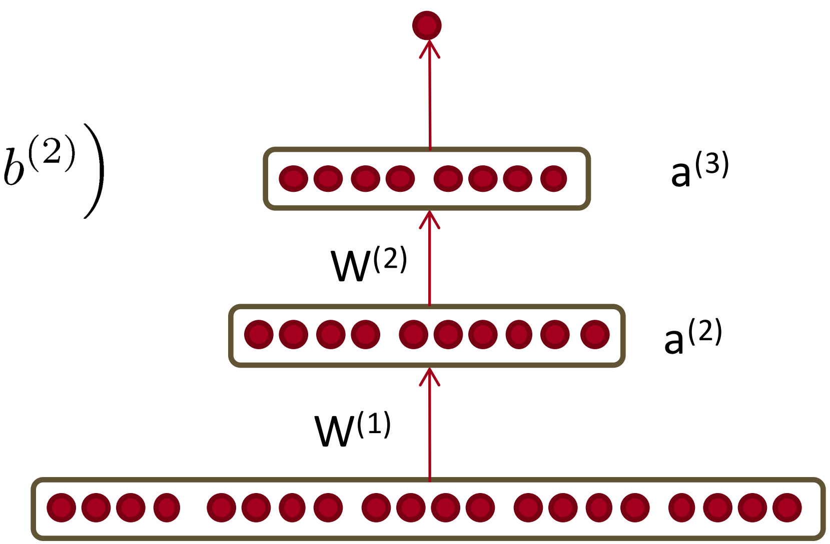
$$
\begin{aligned}
x &= z^{(1)} = a^{(1)} \\
z^{(2)} &= W^{(1)}x + b^{(1)} \\
a^{(2)} &= f\left(z^{(2)}\right) \\
z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\
a^{(3)} &= f\left(z^{(3)}\right) \\
s &= U^{T}a^{(3)}
\end{aligned}
$$



Two layer neural nets and full backprop

- Fully written out as one function:

$$
\begin{aligned}
s &= U^T f\left(W^{(2)} f\left(W^{(1)} x + b^{(1)}\right) + b^{(2)}\right) \\
&= U^T f\left(W^{(2)} a^{(2)} + b^{(2)}\right) \\
&= U^T a^{(3)}
\end{aligned}
$$



- Same derivation as before for $W^{(2)}$ (now sitting on $a^{(2)}$)

$$
\frac{\partial s}{\partial W_{ij}} = \underbrace{U_i f'(z_i)}_{} x_j = \delta_i \; x_j
\qquad
\frac{\partial s}{\partial W_{ij}^{(2)}} = \underbrace{U_i f'\left(z_i^{(3)}\right)}_{} a_j^{(2)} = \delta_i^{(3)} \; a_j^{(2)}
$$

Two layer neural nets and full backprop

- Same derivation as before for top $W^{(2)}$:

$$\frac{\partial s}{\partial W_{ij}^{(2)}} = \underbrace{U_i f'\left(z_i^{(3)}\right)}_{} a_j^{(2)}$$

$$= \delta_i^{(3)} \quad a_j^{(2)}$$

$$\begin{aligned} x &= z^{(1)} = a^{(1)} \\ z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= f\left(z^{(2)}\right) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f\left(z^{(3)}\right) \\ s &= U^T a^{(3)} \end{aligned}$$

- In matrix notation: $\frac{\partial s}{\partial W^{(2)}} = \delta^{(3)} {a^{(2)}}^T$

where $\delta^{(3)} = U \circ f'\left(z^{(3)}\right)$ and $\circ$ is the element-wise product also called Hadamard product ($\otimes$, $\odot$)

- Last missing piece for understanding general backprop: $\frac{\partial s}{\partial W^{(1)}}$

Two layer neural nets and full backprop

- Last missing piece: $\frac{\partial s}{\partial W^{(1)}}$

$$\begin{aligned} x &= z^{(1)} = a^{(1)} \\ z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= f\left(z^{(2)}\right) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f\left(z^{(3)}\right) \\ s &= U^T a^{(3)} \end{aligned}$$

- What's the bottom layer's error message $\delta^{(2)}$?

- Similar derivation to single layer model

- Main difference, we already have ${W^{(2)}}^T \delta^{(3)}$ and need to apply the chain rule again on $f'(z^{(2)})$

Two layer neural nets and full backprop

- Chain rule for: $s = U^T f\left(W^{(2)} f\left(W^{(1)} x + b^{(1)}\right) + b^{(2)}\right)$

- Get intuition by deriving $\frac{\partial s}{\partial W^{(1)}}$ **as if it was a scalar**

- Intuitively, we have to sum over all the nodes coming into layer

- Putting it all together: $\delta^{(2)} = \left({W^{(2)}}^T \delta^{(3)}\right) \circ f'\left(z^{(2)}\right)$

Two layer neural nets and full backprop

$$\begin{aligned} x &= z^{(1)} = a^{(1)} \\ z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= f\left(z^{(2)}\right) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f\left(z^{(3)}\right) \\ s &= U^T a^{(3)} \end{aligned}$$

- Final derivative: $\frac{\partial s}{\partial W^{(1)}} = \delta^{(2)} x^T$

- In general for any matrix $W^{(l)}$ at internal layer l and any error with regularization E_R all backprop in standard multilayer neural networks boils down to 2 equations:

$$\boxed{\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \circ f'(z^{(l)}),}$$

$$\boxed{\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}}$$

- Top and bottom layers have simpler δ

Taking a step back

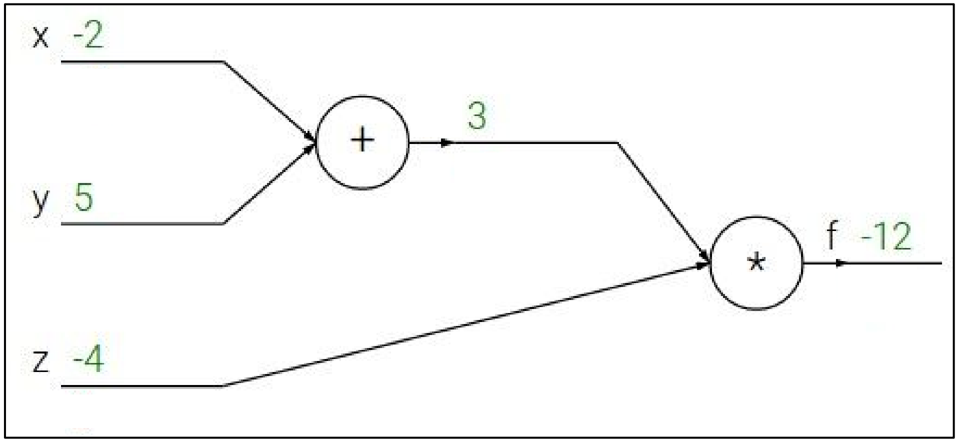
Explanation #2 for backprop: "Circuits"

These examples are from CS231n:
http://cs231n.github.io/optimization-2/

Functions as circuits

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



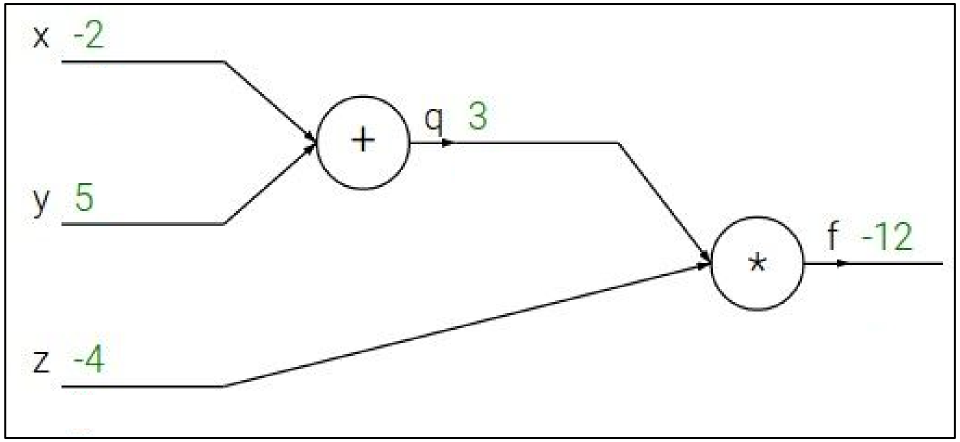
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Recursively walking back through circuit

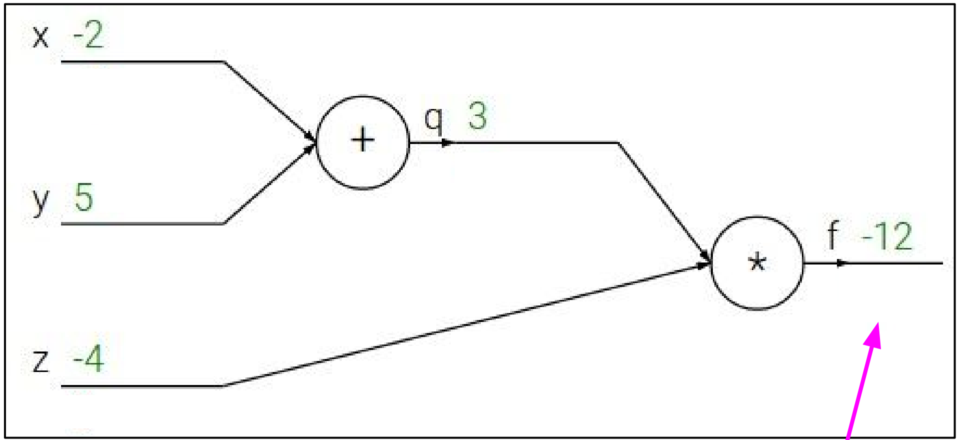
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

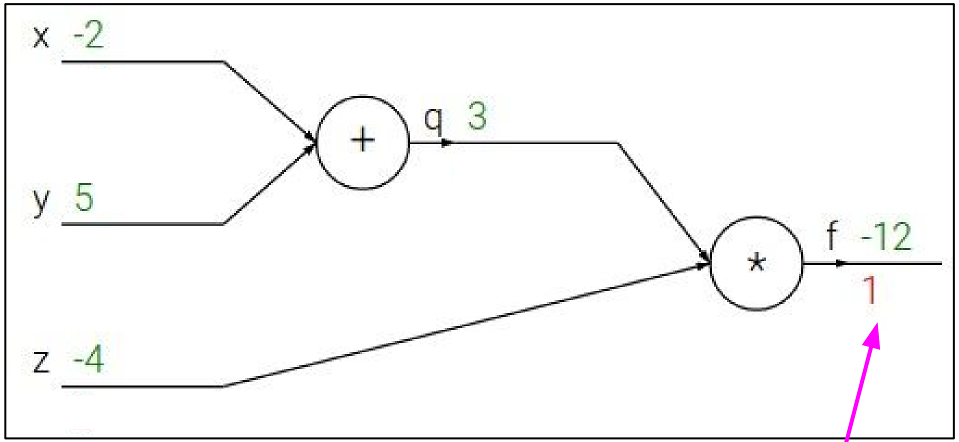
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial f}}$$

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial f}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

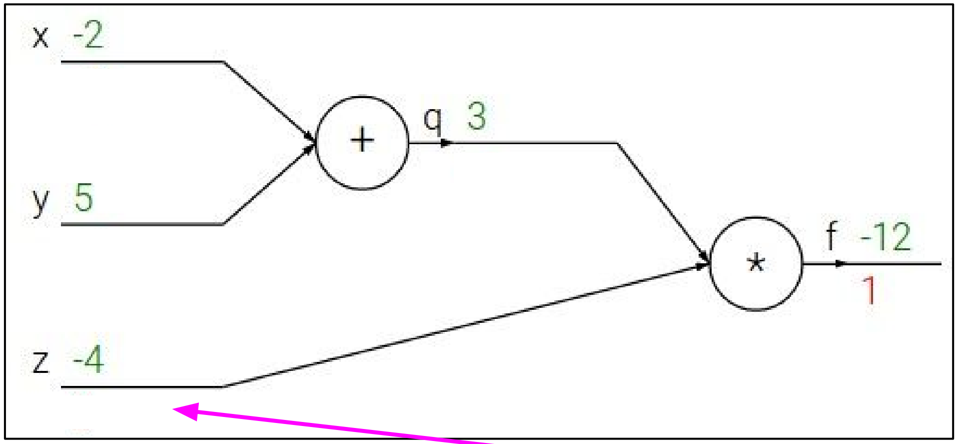
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

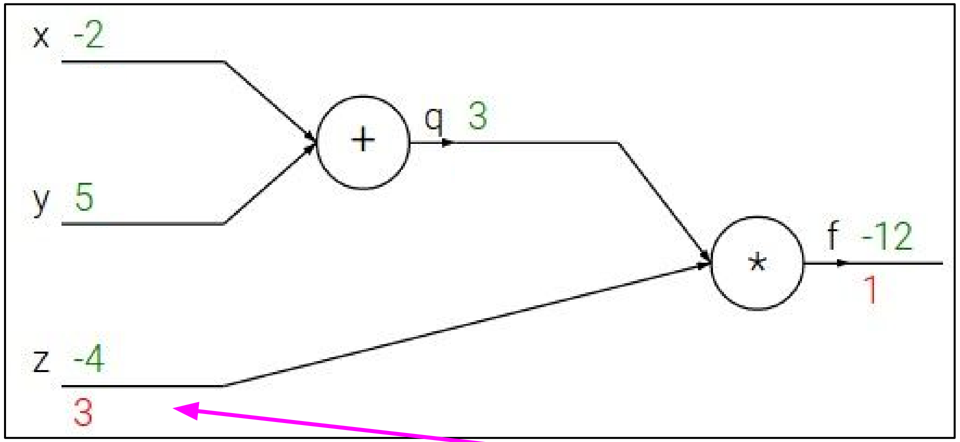
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

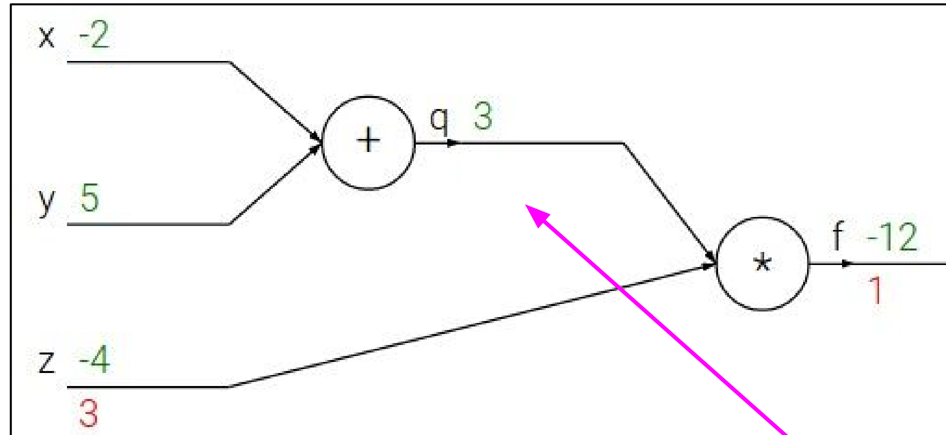
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

$$\boxed{\frac{\partial f}{\partial q}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

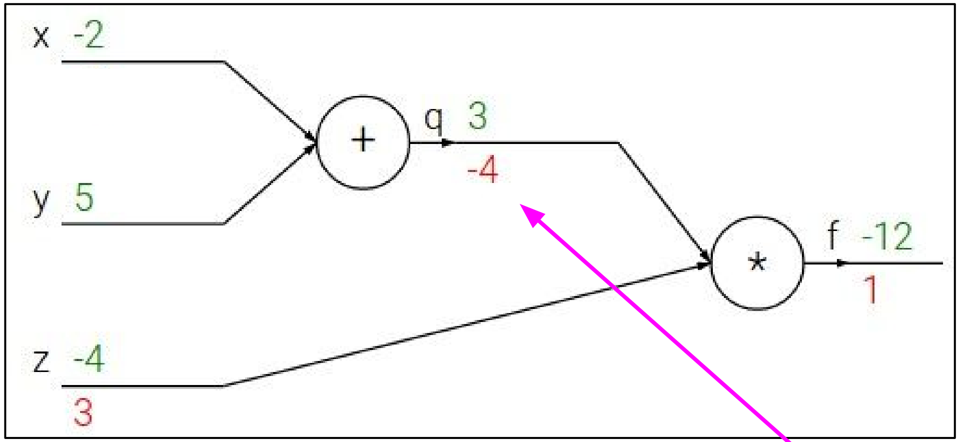
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

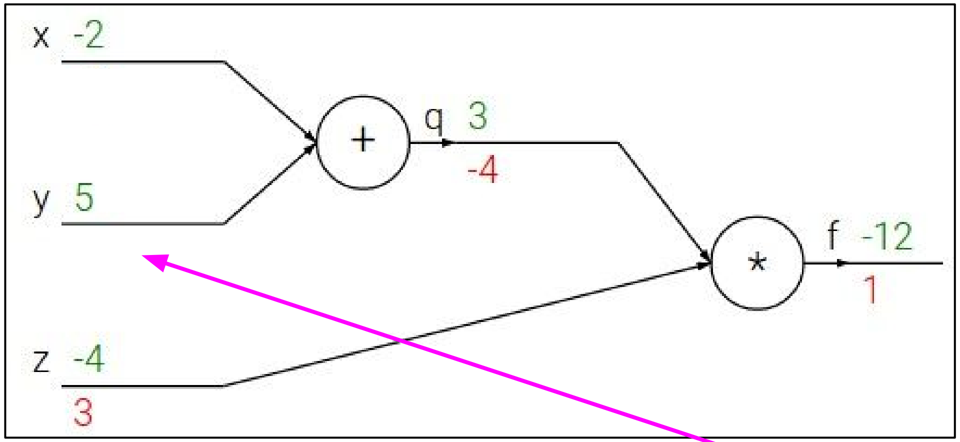
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

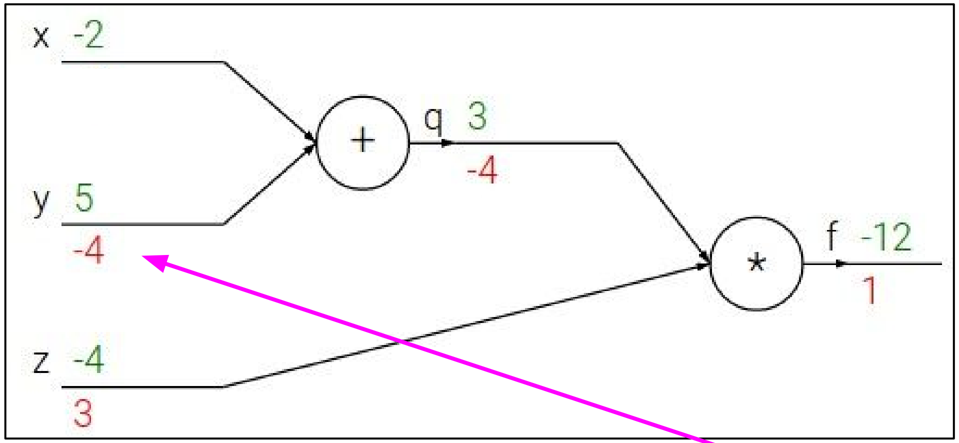
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial y}$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

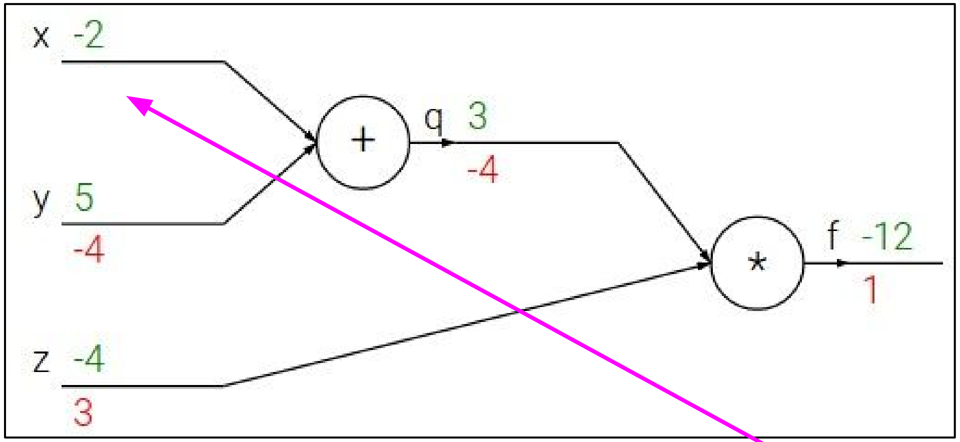
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial x}}$$

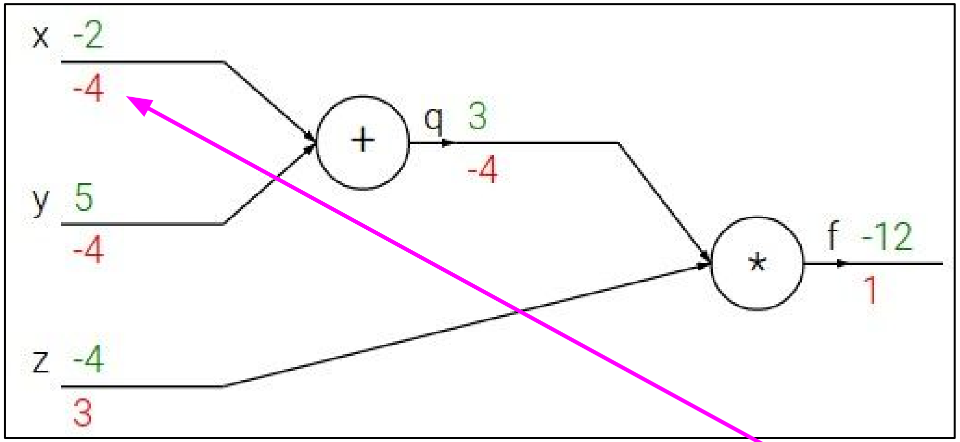
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

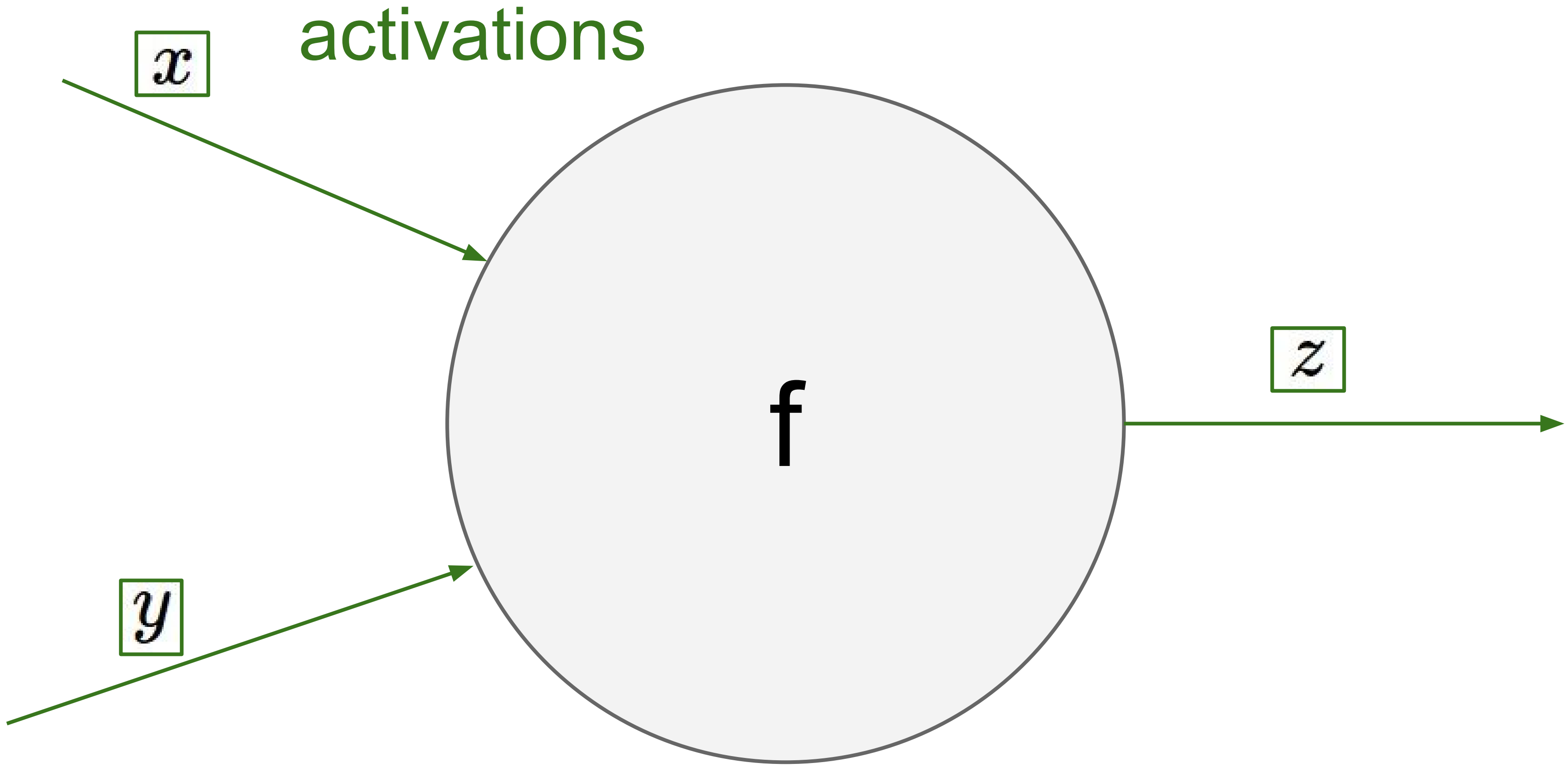
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



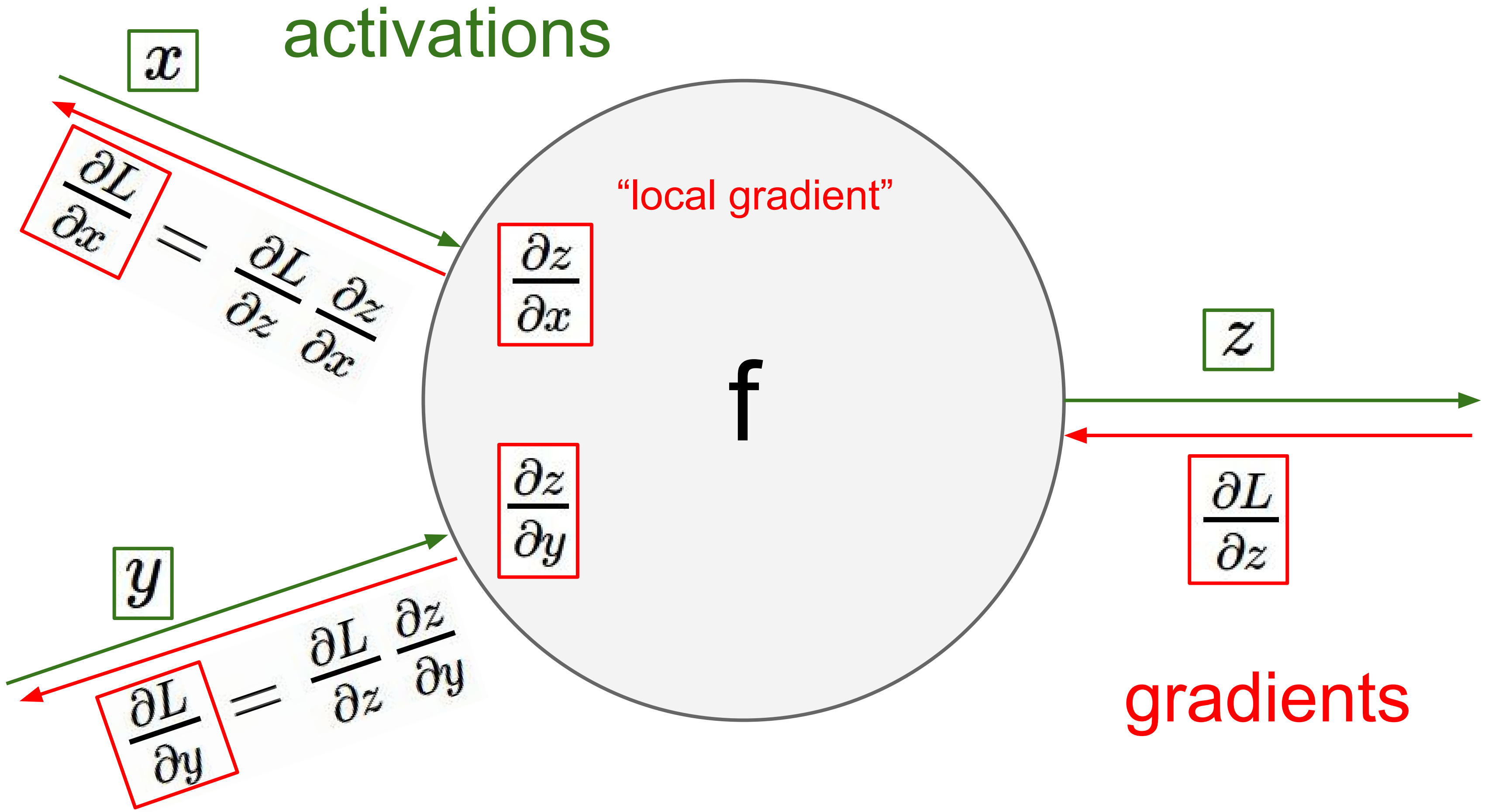
$$\boxed{\frac{\partial f}{\partial x}}$$

Chain rule:

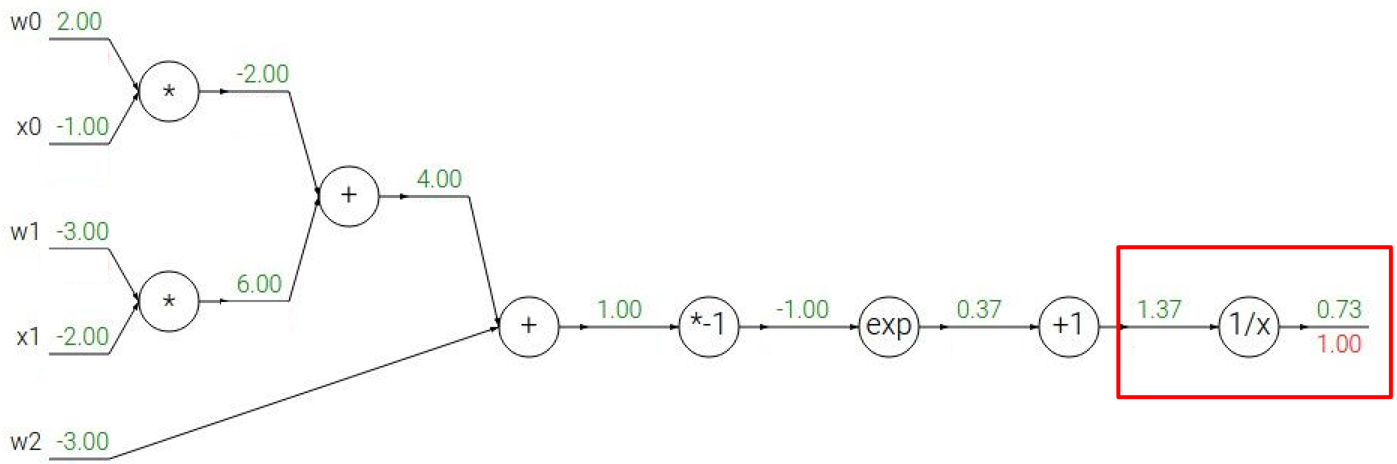
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$


activations
x
y
f
z

Recursively apply chain rule through each node



Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

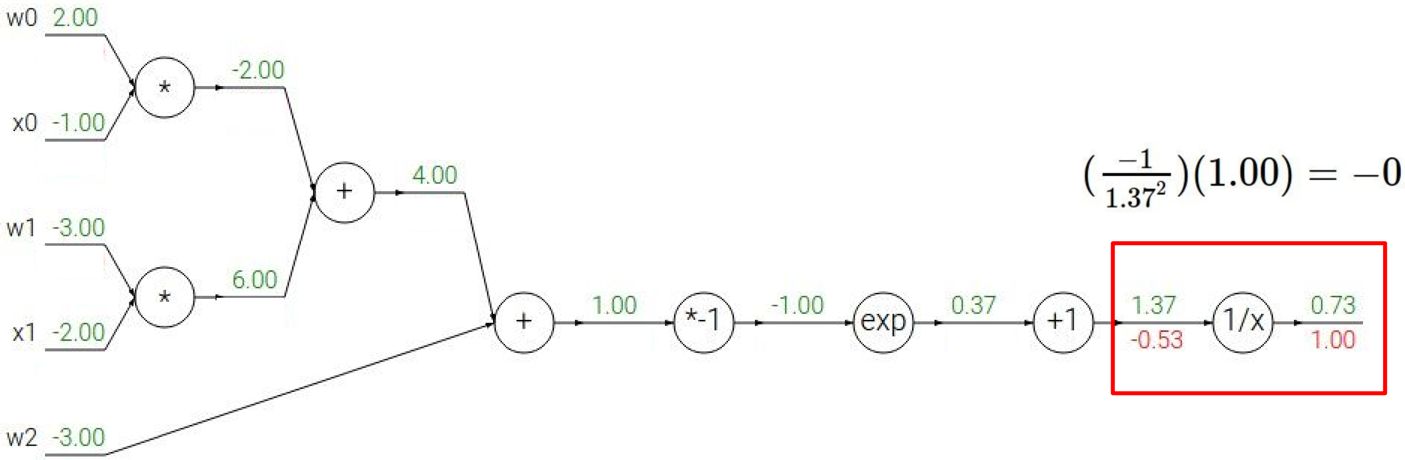
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$\left(\frac{-1}{1.37^2}\right)(1.00) = -0.53$$



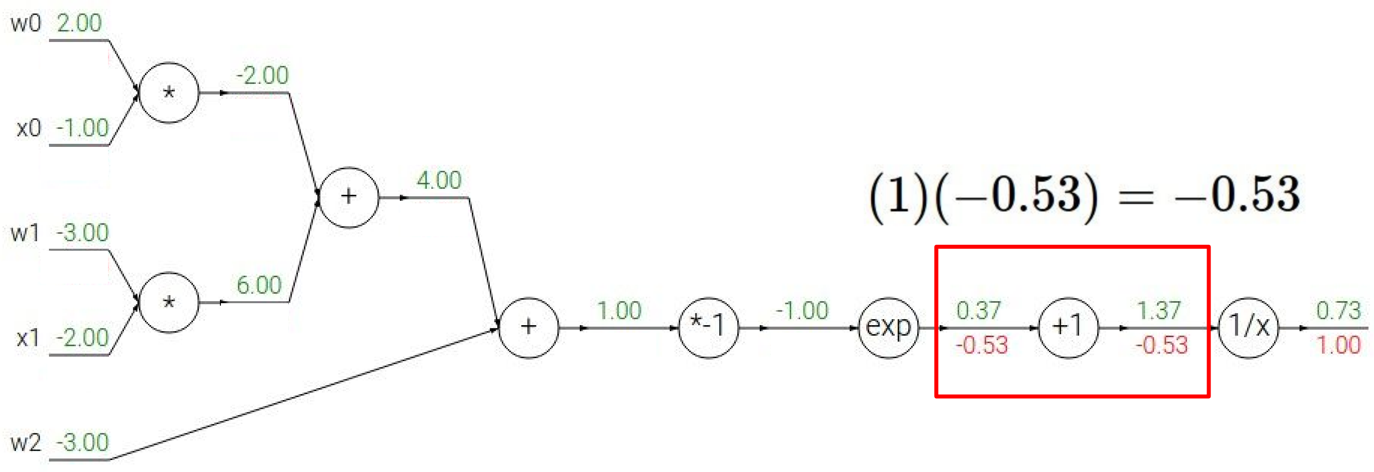
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$(1)(-0.53) = -0.53$$

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

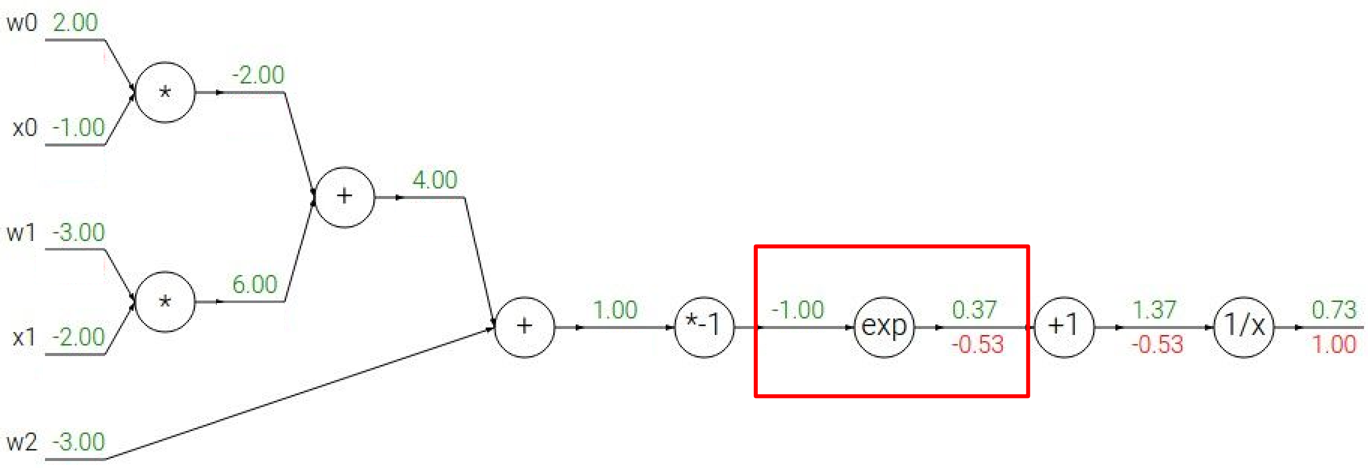
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$\boxed{f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1}$$

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



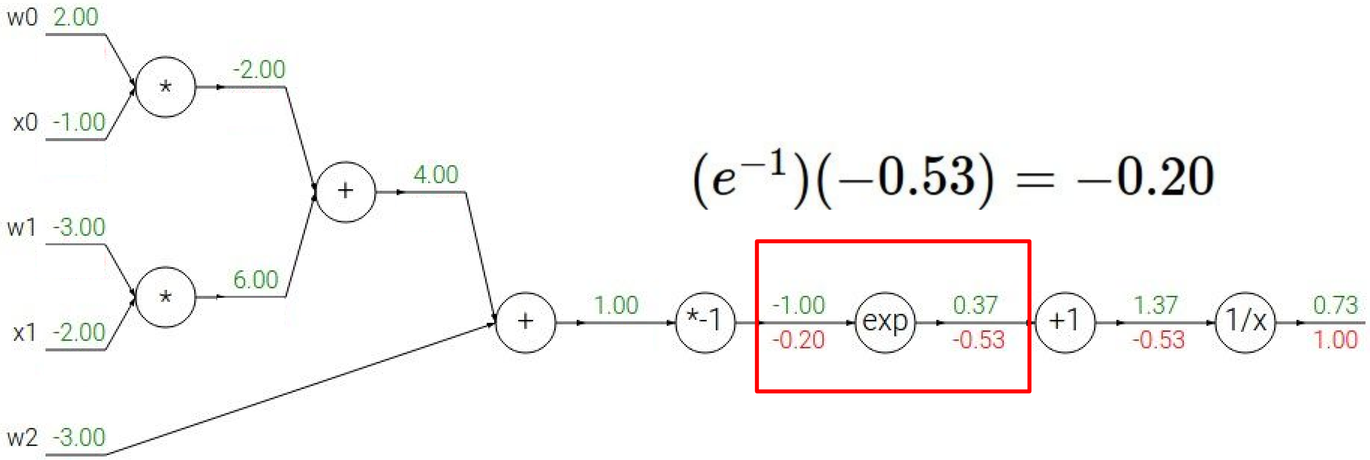
$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$(e^{-1})(-0.53) = -0.20$$

$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

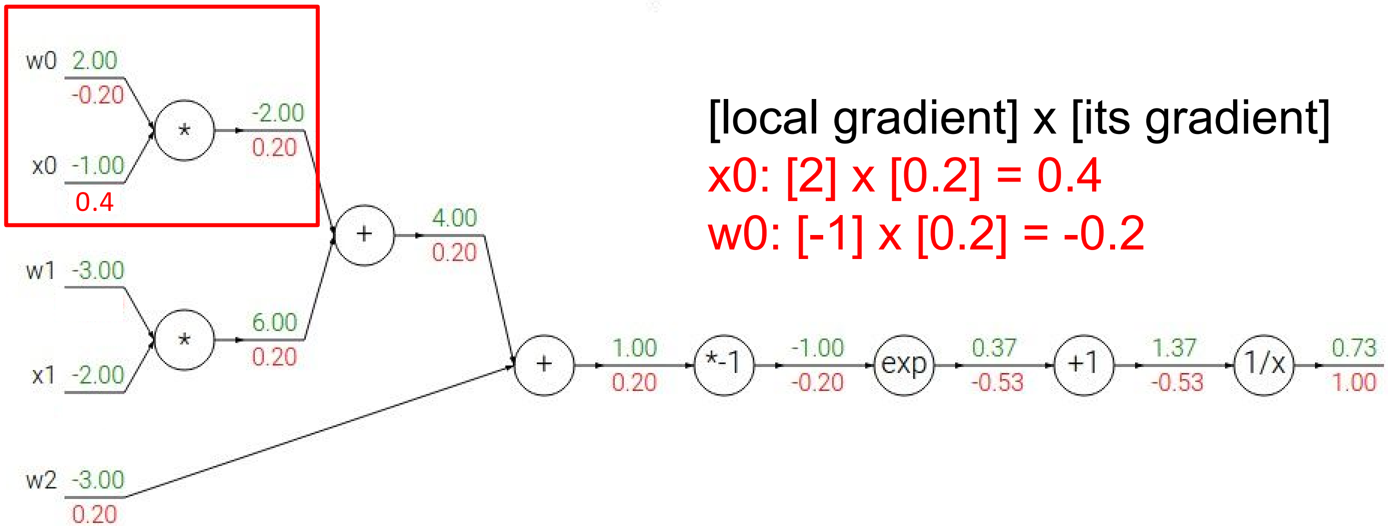
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Jumping to the end...

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]

x0: [2] x [0.2] = 0.4

w0: [-1] x [0.2] = -0.2

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

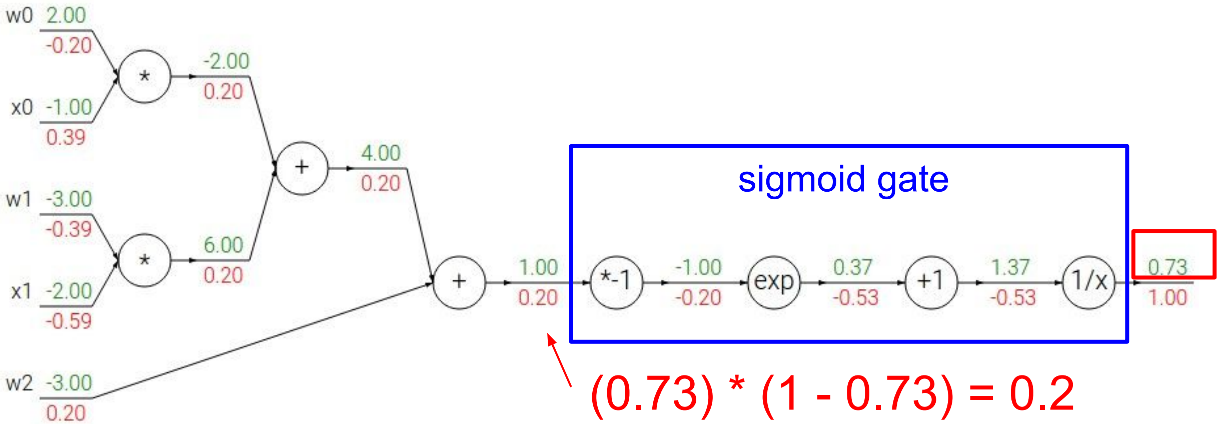
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Combine nodes in the circuit when convenient

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$ sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$



We’ll combine a lot in #4 :)

Explanation #3 for backprop

The high-level flowgraph

Backpropagation (Another explanation)

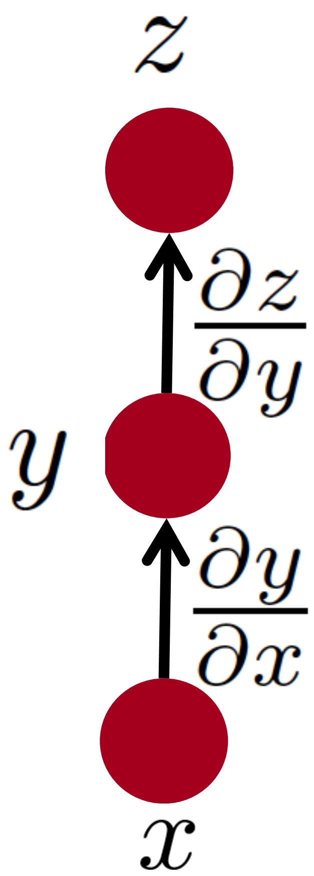
- Compute gradient of example-wise loss wrt parameters

- Simply applying the derivative chain rule wisely

$$z = f(y) \quad y = g(x) \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

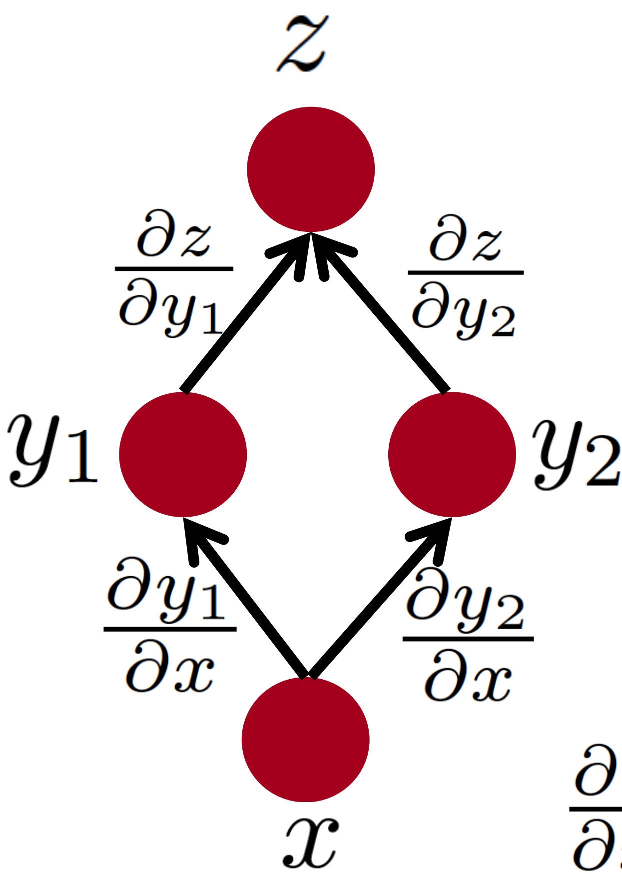
- If computing the loss(example, parameters) is $O(n)$ computation, then so is computing the gradient

Simple Chain Rule



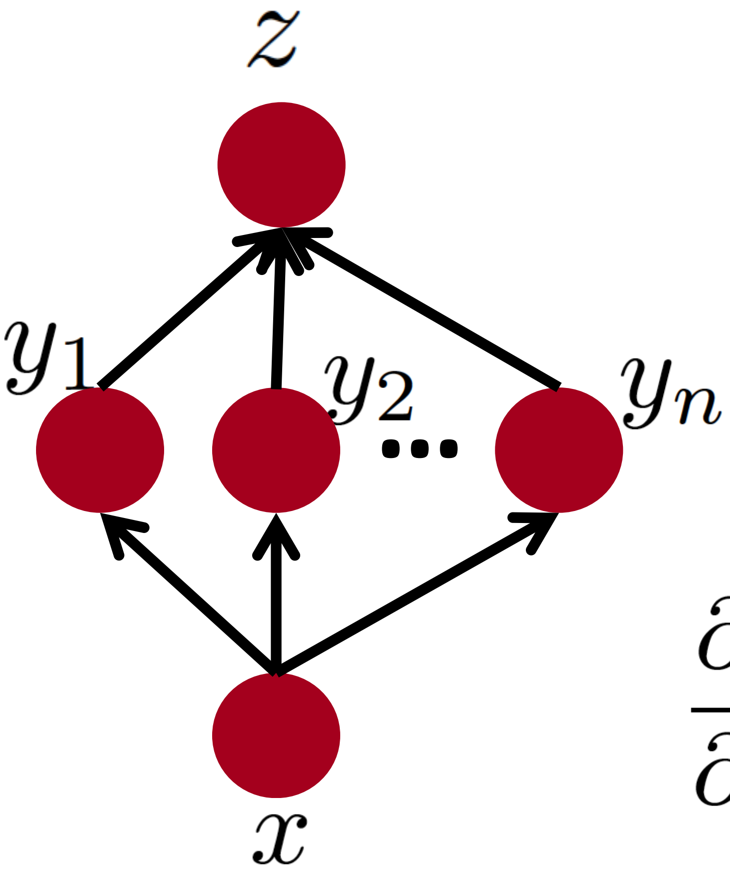
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

Multiple Paths Chain Rule



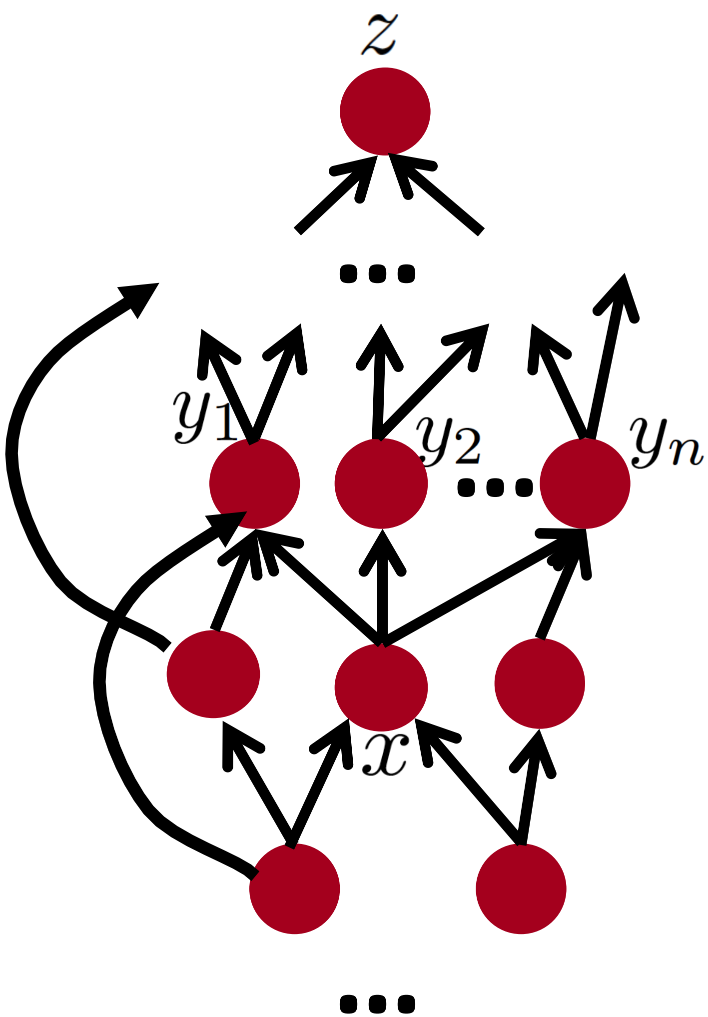
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

Multiple Paths Chain Rule - General



$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

Chain Rule in Flow Graph



Flow graph: any directed acyclic graph

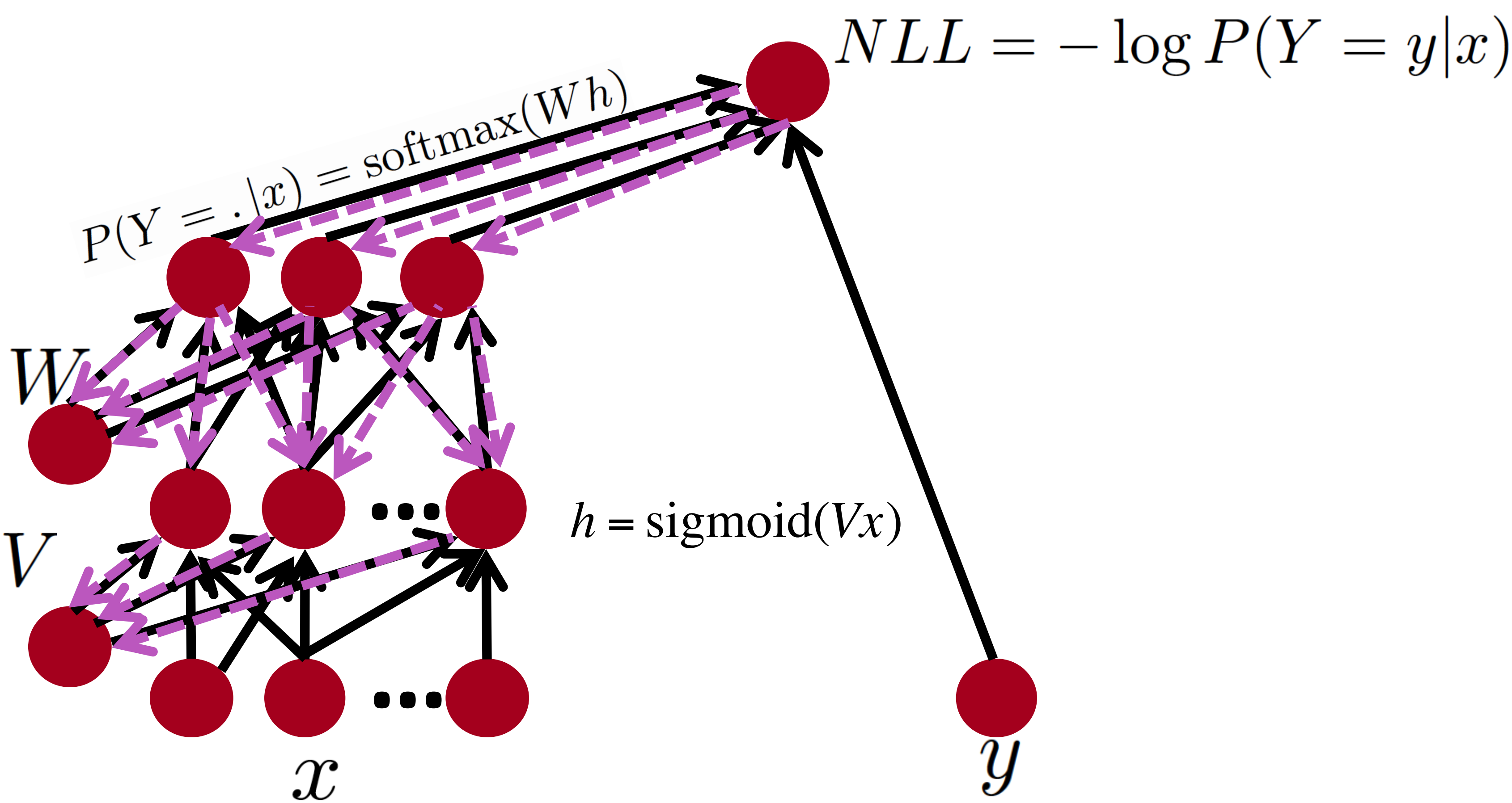
node = computation result

arc = computation dependency

$\{y_1,\ y_2,\ \ldots\ y_n\}$ = successors of x

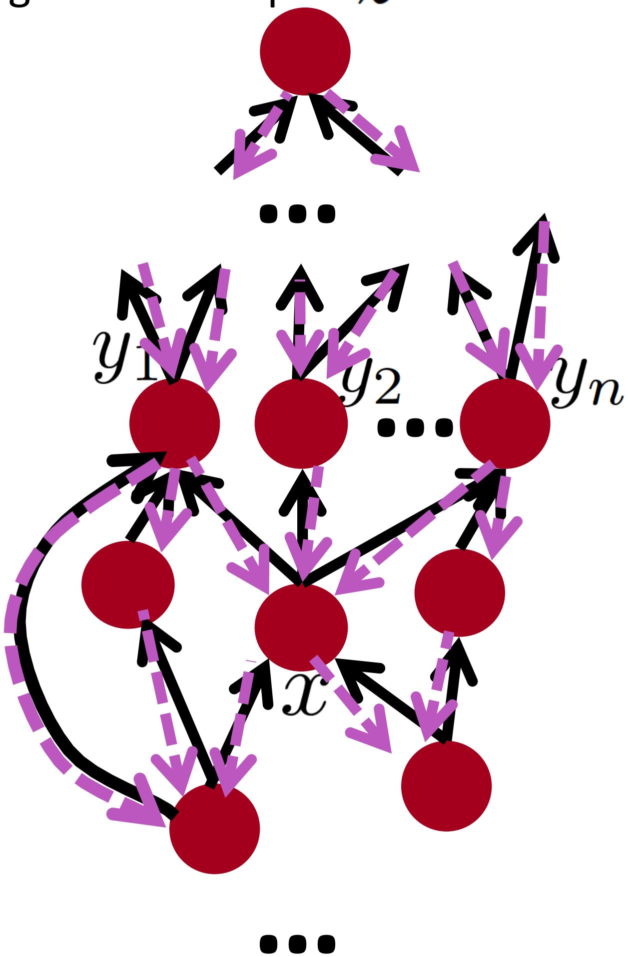
$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

Back-Prop in Multi-Layer Net



Back-Prop in General Flow Graph

Single scalar output z

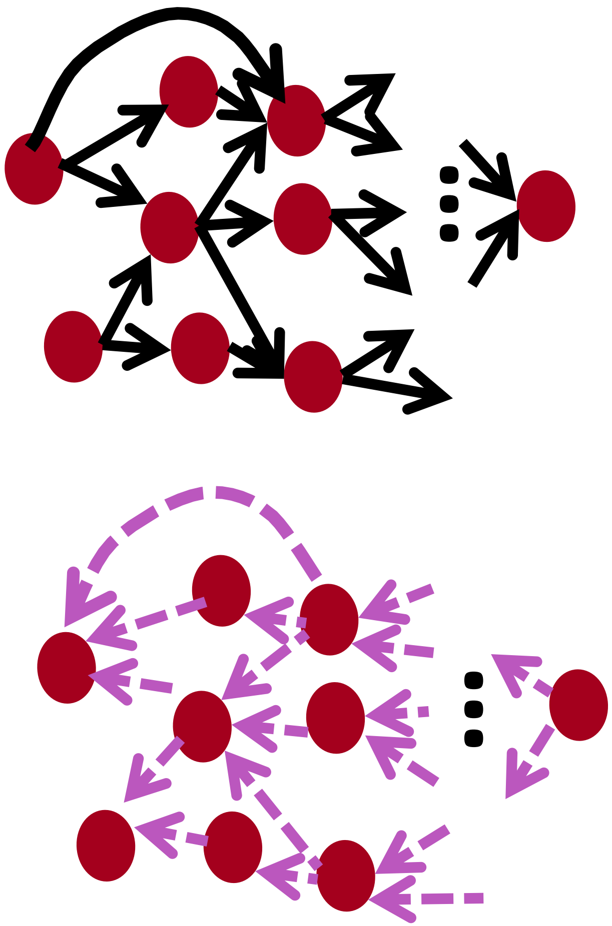


1. Fprop: visit nodes in topo-sort order
   - Compute value of node given predecessors
2. Bprop:
   - initialize output gradient = 1
   - visit nodes in reverse order:
     Compute gradient wrt each node using gradient wrt successors

$\{y_1,\, y_2,\, \ldots\, y_n\}$ = successors of x

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

Automatic Differentiation



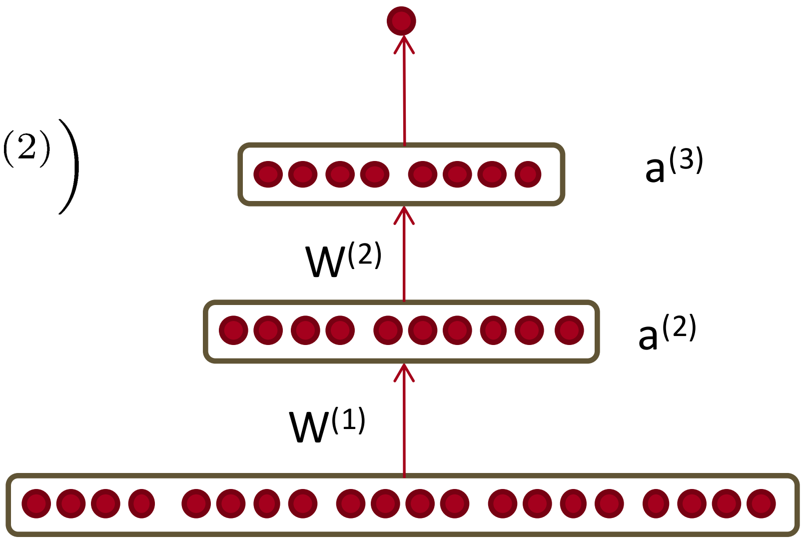
- The gradient computation can be **automatically inferred** from the symbolic expression of the fprop.
- Each node type needs to know how to compute its output and how to compute the gradient wrt its inputs given the gradient wrt its output.
- Easy and fast prototyping

Explanation #4 for backprop

The delta error signals in real neural nets

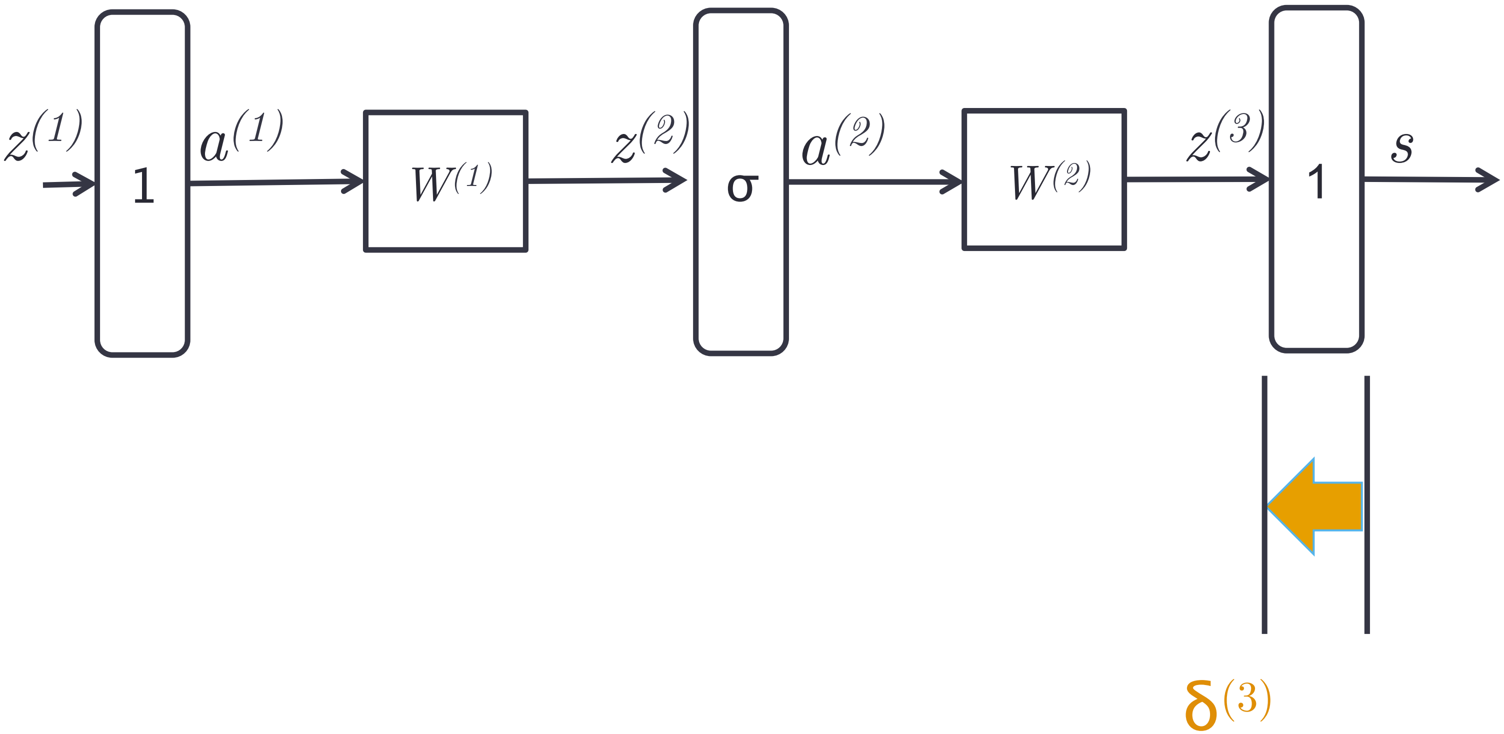
Actual error signals for 2 layer neural net

$$
\begin{aligned}
s &= U^T f\left(W^{(2)} f\left(W^{(1)} x + b^{(1)}\right) + b^{(2)}\right) \\
&= U^T f\left(W^{(2)} a^{(2)} + b^{(2)}\right) \\
&= U^T a^{(3)}
\end{aligned}
$$



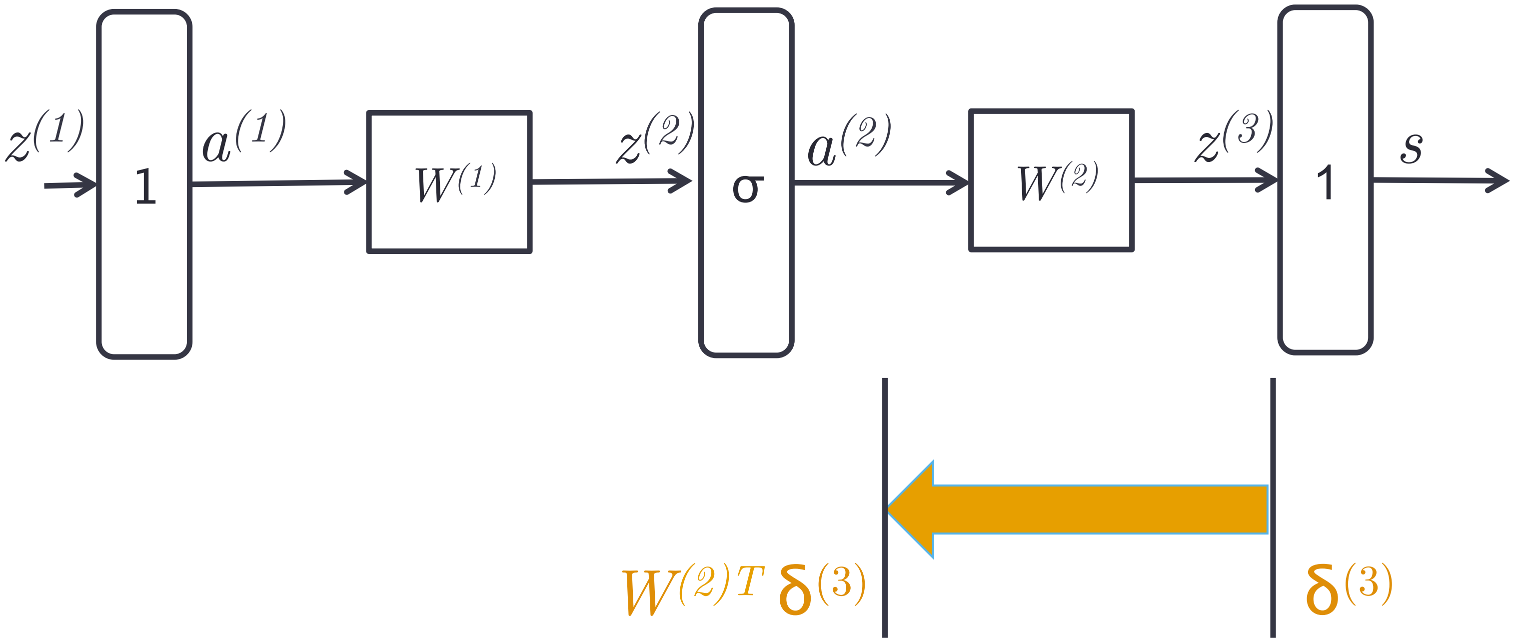
Visualization of intuition

- Let's say we want $\frac{\partial s}{\partial W^{(1)}} = \delta^{(2)} a^{(1)^T}$ with previous layer and f = σ



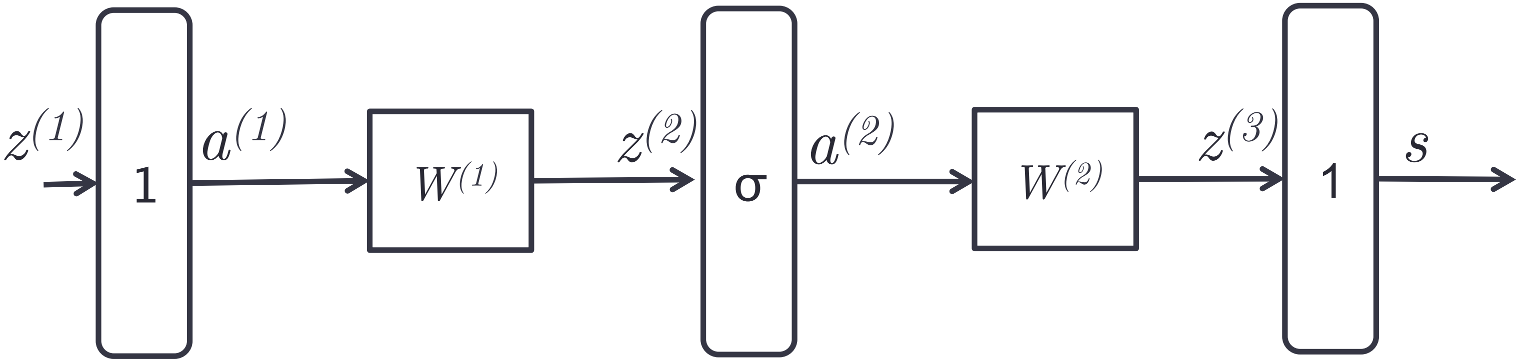
Gradient w.r.t $W^{(2)} = \delta^{(3)} a^{(2)\mathrm{T}}$

Visualization of intuition



--Reusing the $\delta^{(3)}$ for downstream updates.
--Moving error vector across affine transformation simply requires multiplication with the transpose of forward matrix
--Notice that the dimensions will line up perfectly too!

Visualization of intuition



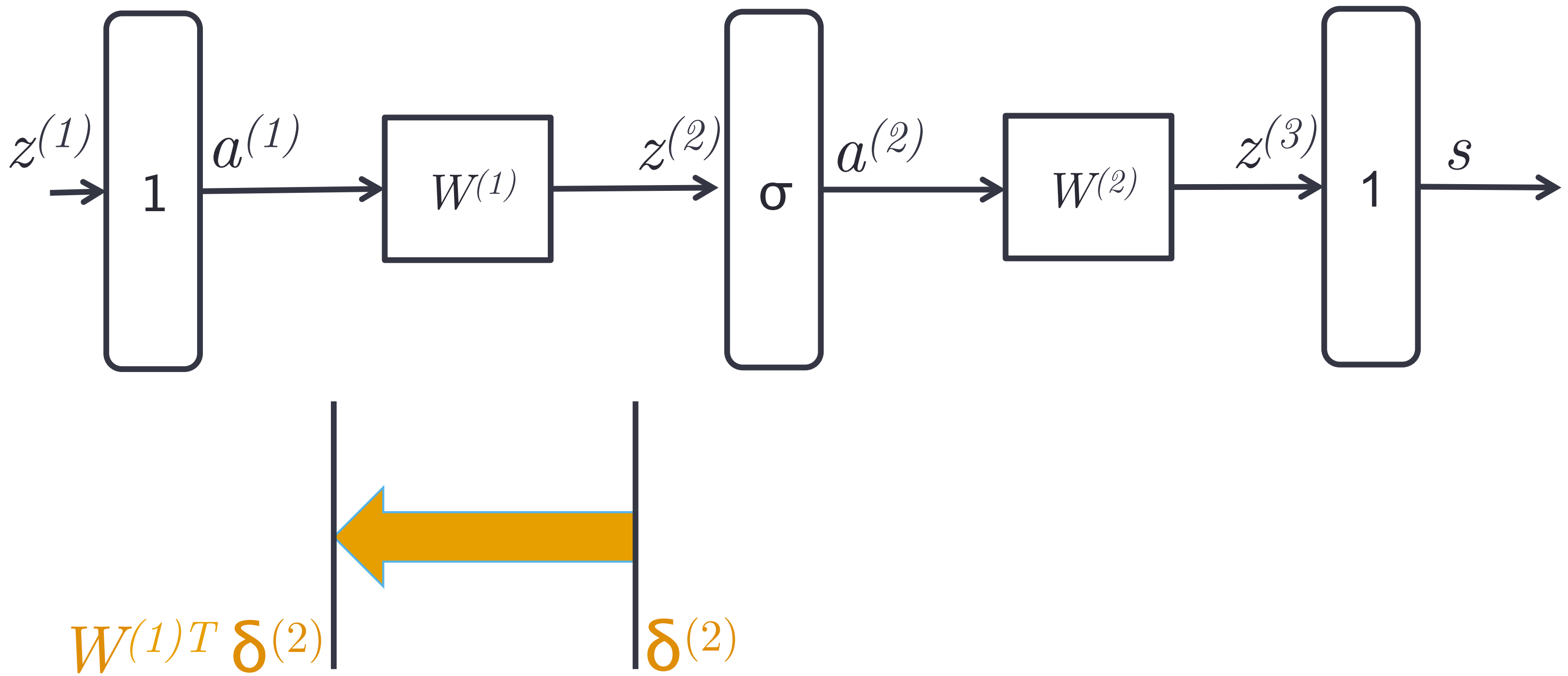
$$W^{(2)T}\delta^{(3)}$$

$$\sigma'(z^{(2)}) \odot W^{(2)T}\delta^{(3)}$$

$$= \delta^{(2)}$$

--Moving error vector across point-wise non-linearity requires point-wise multiplication with local gradient of the non-linearity

Visualization of intuition



$$\text{Gradient w.r.t } W^{(1)} = \delta^{(2)} a^{(1)\mathrm{T}}$$

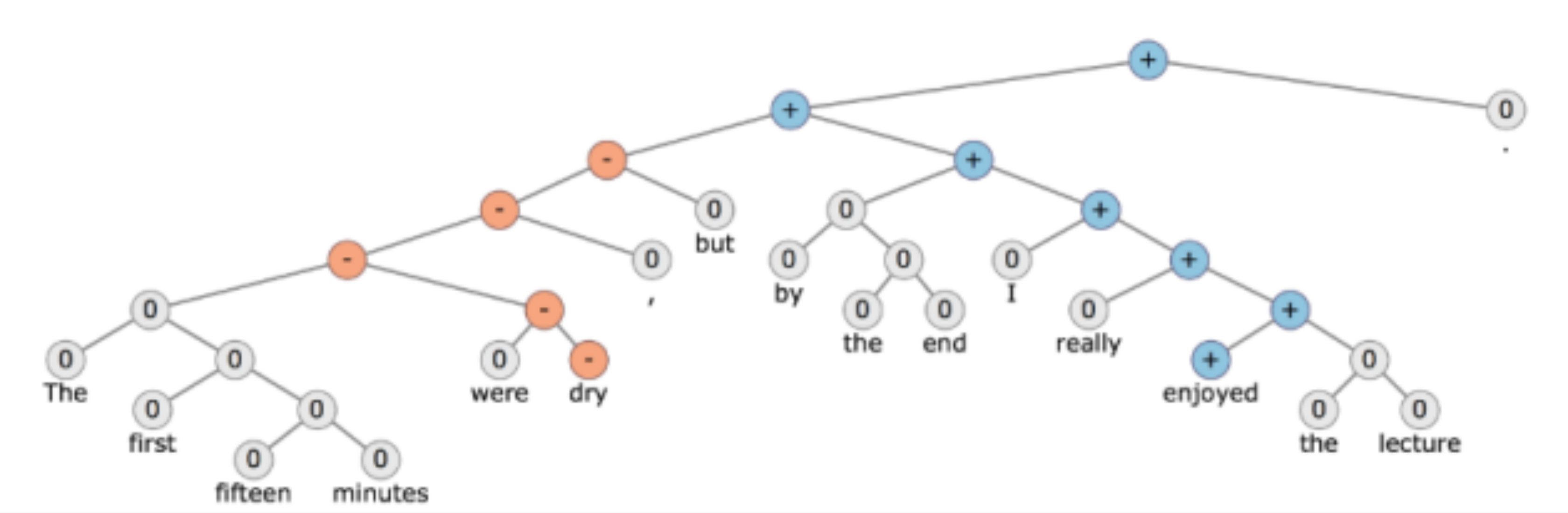
You survived Backprop!

- Congrats!
- You now understand the inner workings of most deep learning models out there
- This was the hardest part of the class
- Everything else from now on is mostly just more matrix multiplications and backprop :)

Bag of Tricks for Efficient Text Classification

Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov
Facebook AI Research

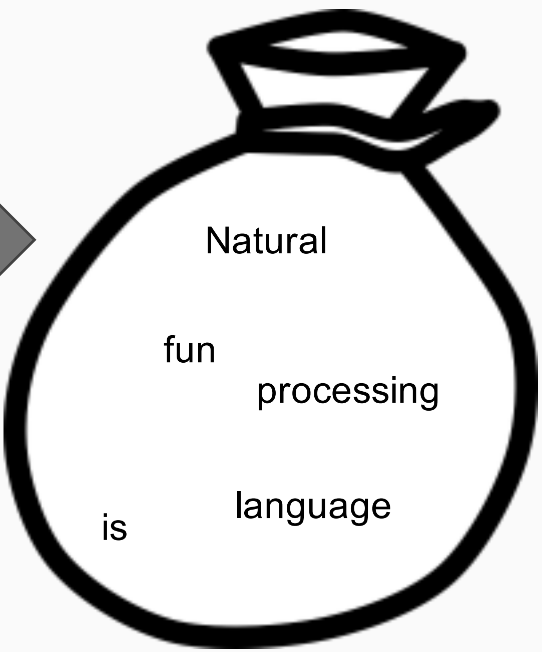
Text classification



Bag of Words (or n-grams)

low-dimensional!

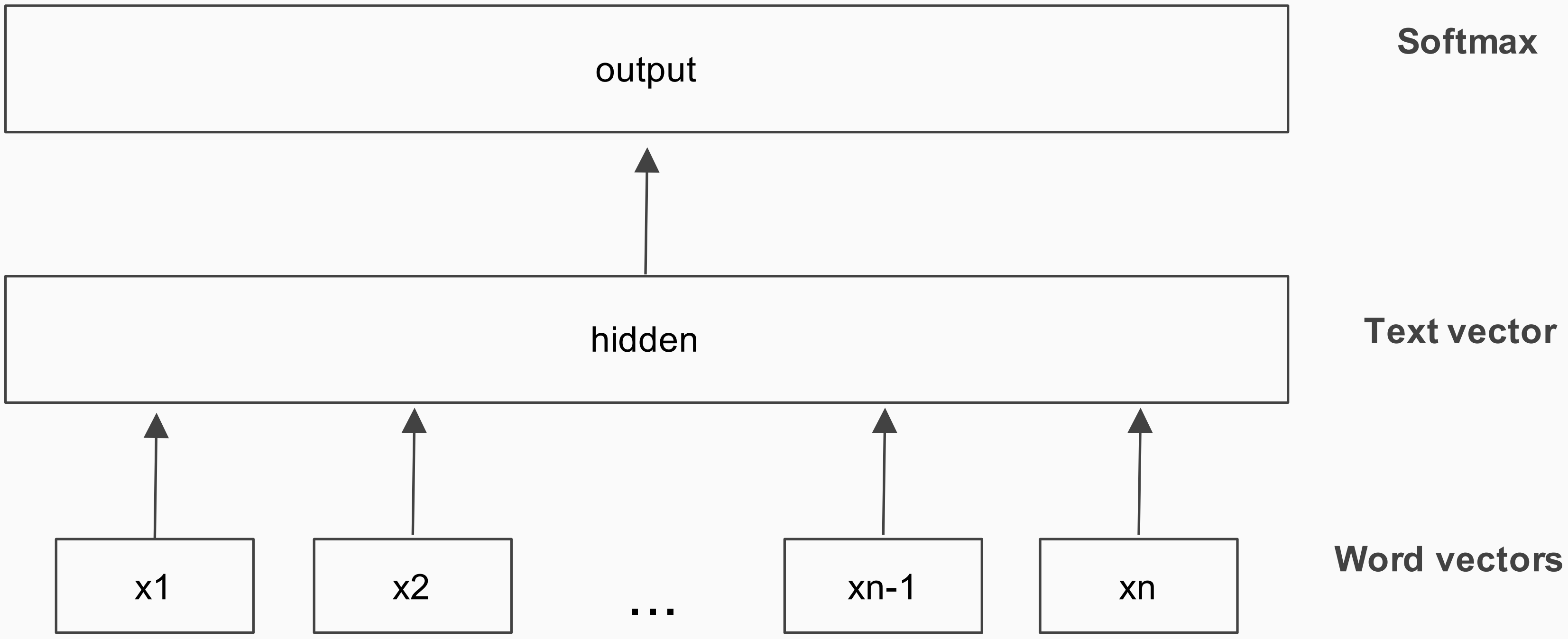
Natural language processing is fun.



Average

$$\begin{pmatrix} -0.132 \\ 1.129 \\ 0.827 \\ 0.110 \\ -0.527 \\ 0.156 \\ 0.349 \\ -0.286 \end{pmatrix}$$

Simple linear model



Learning

weight matrices

label of n-th doc

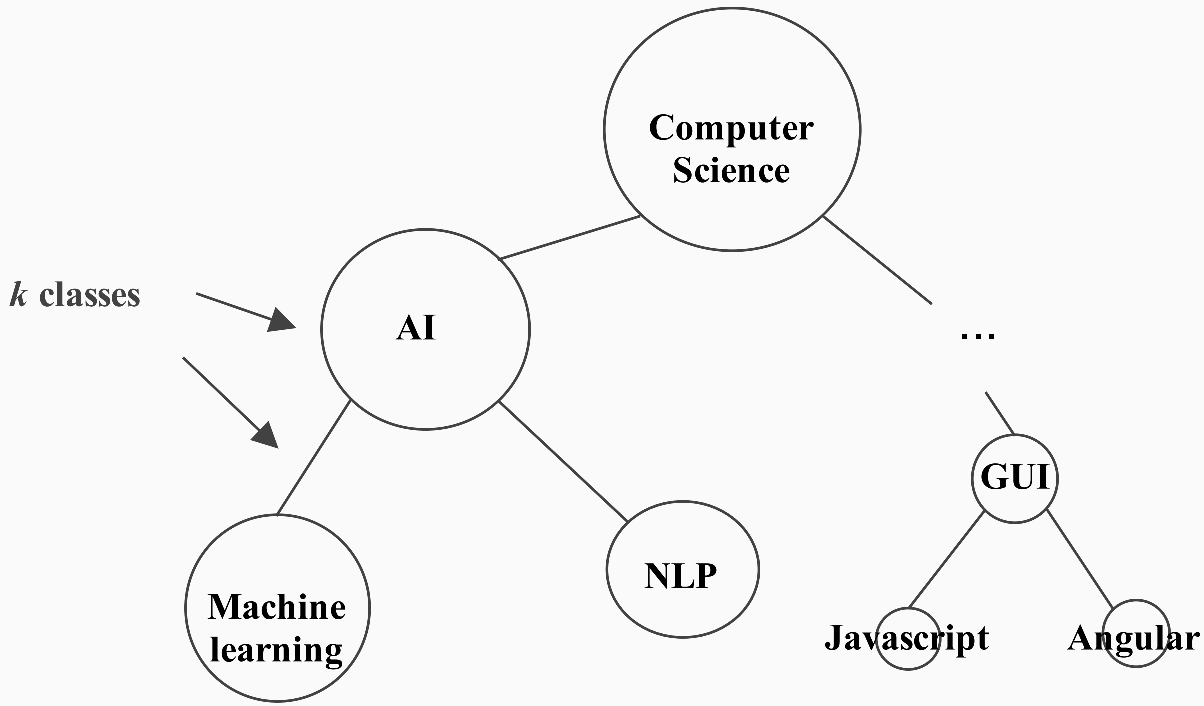
normalized bag of features of n-th doc

$$-\frac{1}{N}\sum_{n=1}^{N} y_n log(f(BAx_n))$$

 # documents

softmax

Hierarchical softmax



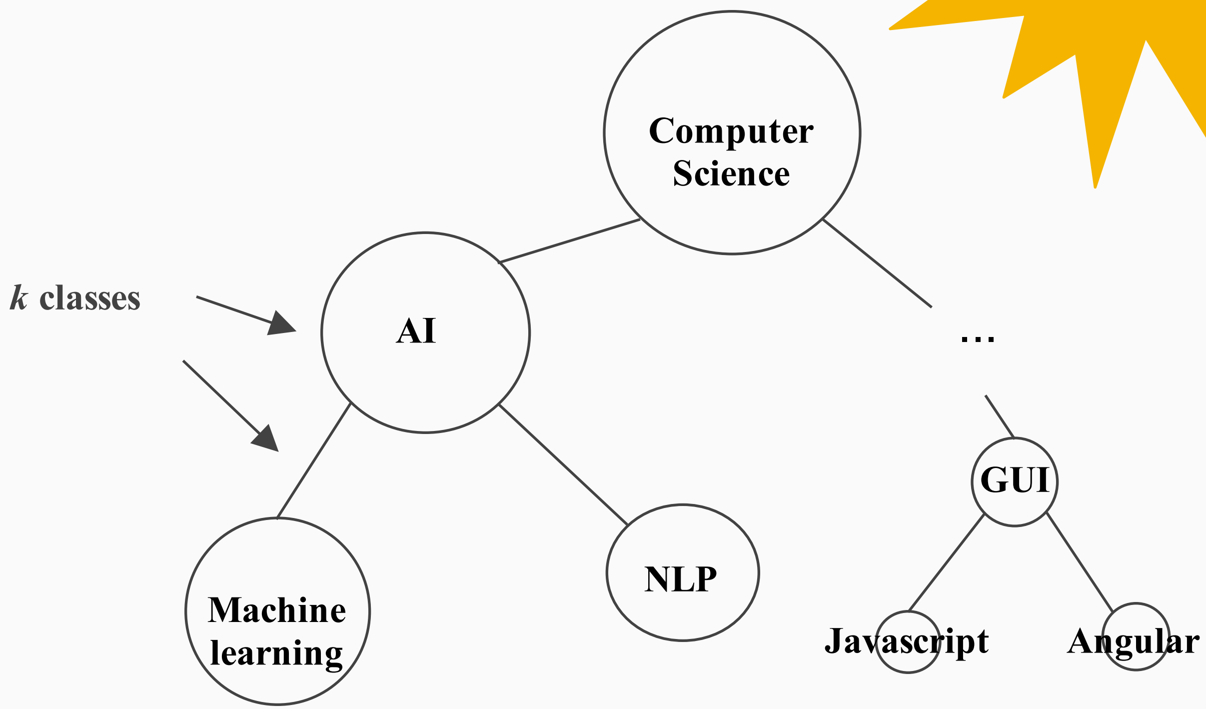
$$P(n_{l+1}) = \prod_{i=1}^{l} P(n_i).$$

Probability of a node is always lower than one of its parent

Hierarchical softmax

(Text representation dimension *h*)

O(hlog(k)) vs **O(kh)** training time!



$$P(n_{l+1}) = \prod_{i=1}^{l} P(n_i).$$

Probability of a node is always lower than one of its parent

Results

As good as NN!

	Yahoo		Amazon full		Amazon polarity	
	Accuracy	Time	Accuracy	Time	Accuracy	Time
char-CNN	71.2	1 day	59.5	5 days	94.5	5 days
VDCNN	73.4	2h	63	7h	95.7	7h
fastText	72.3	5s	60.2	9s	94.6	10s

Fast!

Summary

fastText is often on par with deep learning classifiers

fastText takes seconds, instead of days

Can learn vector representations of words in different languages (with performance better than word2vec!)

Thanks!

Class Project

- Important (30%) and lasting result of the class
- Final Project Poster Presentation: 2%

- Choice of doing Assignment 4 or a Final project

- Mandatory approved mentors → You need to reach out to potential mentors
- Start early and clearly define your task and dataset

- See https://web.stanford.edu/class/cs224n/project.html

Project types

1. Apply existing neural network model to a new task
2. Implement a complex neural architecture
3. Come up with a new neural network model
4. Theory of deep learning, e.g. optimization

Class Project: Apply Existing NNets to Tasks

1. Define Task:
   - Example: **Summarization**

2. Define Dataset
   1. Search for academic datasets
      - They already have baselines
      - E.g.: Document Understanding Conference (DUC)

   2. Define your own (harder, need more new baselines)
      - If you're a graduate student: connect to your research
      - Summarization, Wikipedia: Intro paragraph and rest of large article
      - Be creative: Twitter, Blogs, News

Class Project: Apply Existing NNets to Tasks

3. Define your metric
   - Search online for well established metrics on this task
   - Summarization: Rouge (Recall-Oriented Understudy for Gisting Evaluation) which defines n-gram overlap to human summaries

4. Split your dataset!
   - Train/Dev/Test
   - Academic dataset often come pre-split
   - Don’t look at the test split until ~1 week before deadline! (or at most once a week)

Class Project: Apply Existing NNets to Tasks

5. Establish a baseline
   - Implement the simplest model (often logistic regression on unigrams and bigrams) first
   - Compute metrics on train AND dev
   - Analyze errors
   - If metrics are amazing and no errors:
     done, problem was too easy, restart :)

6. Implement existing neural net model
   - Compute metric on train and dev
   - Analyze output and errors
   - Minimum bar for this class

Class Project: Apply Existing NNets to Tasks

7. Always be close to your data!
   - Visualize the dataset
   - Collect summary statistics
   - Look at errors
   - Analyze how different hyperparameters affect performance

8. Try out different model variants
   - Soon you will have more options
     - Word vector averaging model (neural bag of words)
     - Fixed window neural model
     - Recurrent neural network
     - Recursive neural network
     - Convolutional neural network

Class Project: A New Model -- Advanced Option

- Do all other steps first (Start early!)
- Gain intuition of why existing models are flawed

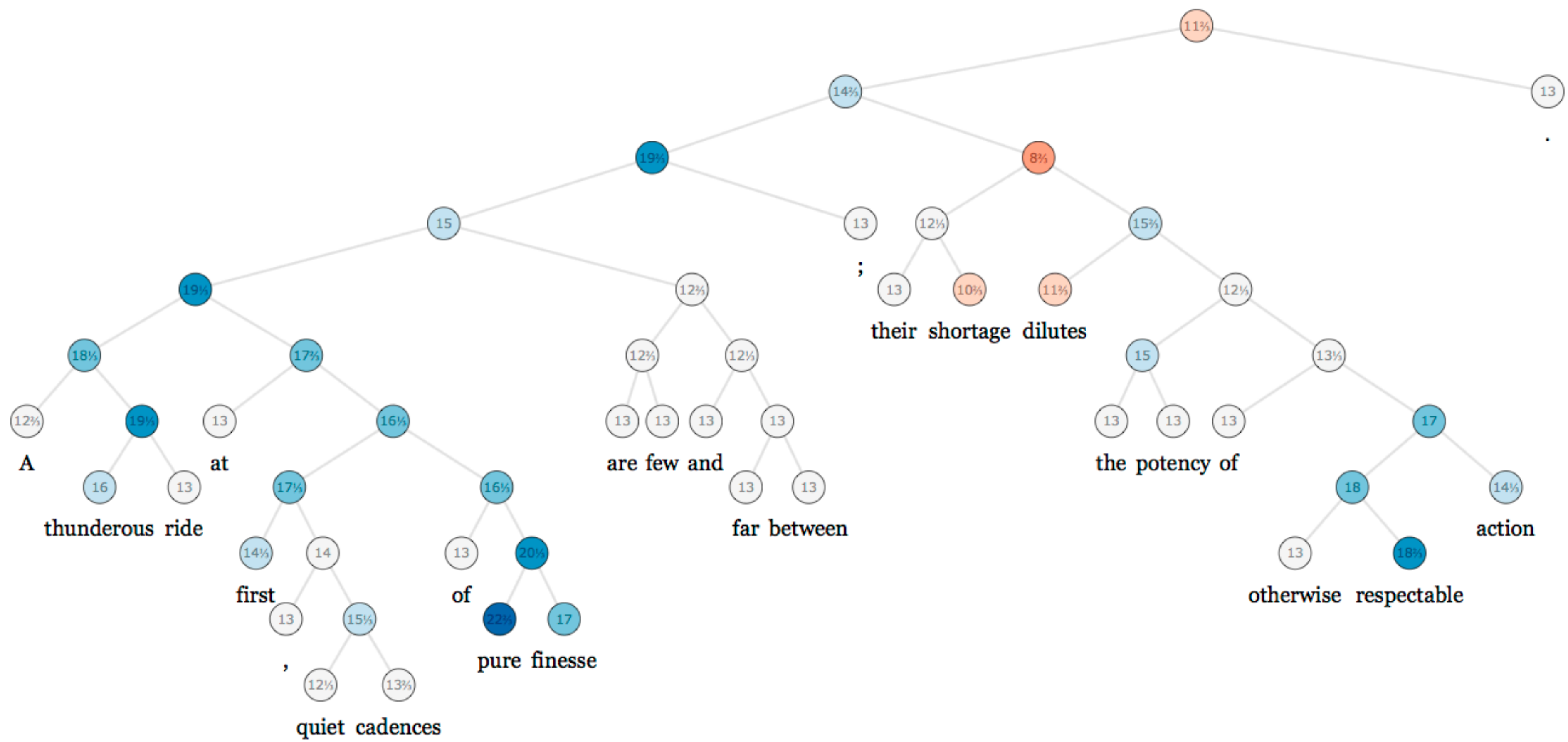
- Talk to researcher/mentor, come to project office hours a lot
- Implement new models and iterate quickly over ideas
- Set up efficient experimental framework
- Build simpler new models first
- Example Summarization:
  - Average word vectors per paragraph, then greedy search
  - Implement language model (introduced later)
  - Stretch goal: Generate summary with seq2seq!

Project Ideas

- Summarization
- NER, like PSet 2 but with larger data
  Natural Language Processing (almost) from Scratch, Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa, http://arxiv.org/abs/1103.0398
- Simple question answering, A Neural Network for Factoid Question Answering over Paragraphs, Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher and Hal Daumé III (**EMNLP 2014**)
- Image to text mapping or generation,
  Grounded Compositional Semantics for Finding and Describing Images with Sentences, Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, Andrew Y. Ng. (**TACL 2014**)
  or
  Deep Visual-Semantic Alignments for Generating Image Descriptions, Andrej Karpathy, Li Fei-Fei
- Entity level sentiment
- Use DL to solve an NLP challenge on kaggle,
  Develop a scoring algorithm for student-written short-answer responses, https://www.kaggle.com/c/asap-sas

Another example project: Sentiment

- Sentiment on movie reviews: http://nlp.stanford.edu/sentiment/
- Lots of deep learning baselines and methods have been tried



Next up

- Some fun and fundamental linguistics with syntactic parsing
- TensorFlow lecture (for Ass.2) – also useful for projects and life :)