Deep Learning

CSCI 1470

Eric Ewing

Friday, 2/5/25

Day 7: More Gradients!

Final Project – A very brief overview

Option 1 (most people): Find an academic paper and implement it. If an implementation already exists in Tensorflow, you cannot use Tensorflow.

Option 2 (Capstones): More extensive project that goes beyond just re-implementing an existing paper (i.e., perform a research project).

Workshops and SRC Discussion Starting up!

Workshops this week:

- Introduction to Pytorch and Jax: Tensorflow isn't the only was to train a model! This workshop will show you how to use other common python libraries for deep learning, Pytorch and Jax. Whether you want to use one of these libraries for the final project, or you are interested to see other ways to implement deep learning this workshop is for you!
- How to read (and implement) Academic Papers: Unfamiliar codebases and academic papers can be pretty overwhelming. This workshop will teach you how to extract the most valuable information from both, and how to break down long passages into more understandable chunks. The skills gained in this workshop will equip you with more confidence at final project time as well as more comfort in the CS world at large!

Homework 2: Beras

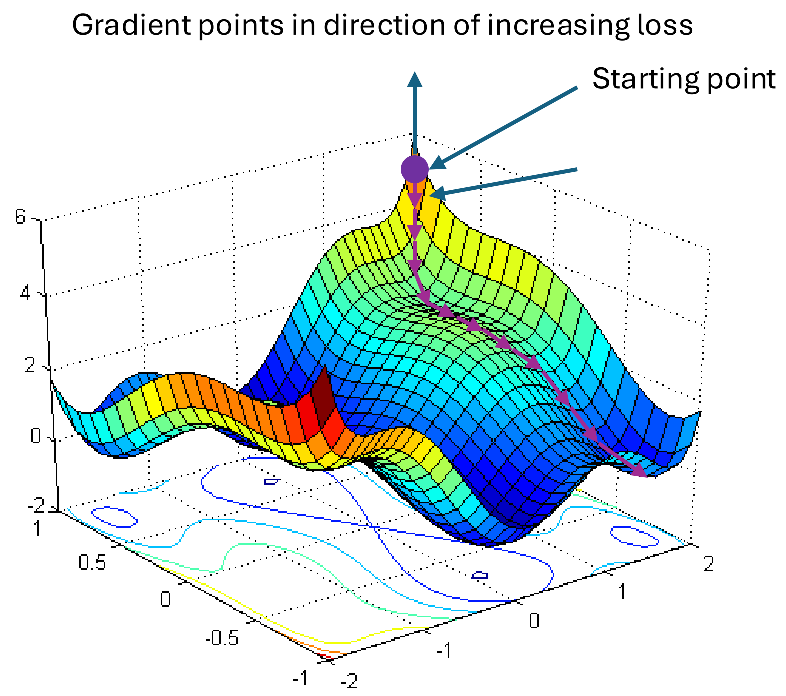
- Releases Today
  - Conceptual due in 2 weeks (2/19)
  - Programming due in 3 weeks (2/26)
- Coding Gradient Tape and Neural Networks with Numpy
- You must start early!

Recap

Backprop is used to compute gradients of a neural network efficiently

Gradients are used in Stochastic Gradient Descent to update NN parameters

SGD outperforms gradient descent in both speed and solution quality



Today's Goals

(1) Classification and Backprop

(2) Intro to Autograd and Popular DL Frameworks

Binary Classification Review

Input: $\mathbb{X}$

Target: $\mathbb{Y}$

Pixel Grid

Is it digit 2?

$x^{(1)} =$ 

28x28 pixels

Function: f

$f(\mathbb{X}) \rightarrow \mathbb{Y}$

$y^{(1)} = 1$ 

$x^{(2)} =$ 

$y^{(2)} = 0$ 

What is a reasonable loss function to use?

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…
- In classification, we’d like the most accurate model…

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…
- In classification, we'd like the most accurate model…
- So let's make our loss function negative accuracy!

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…
- In classification, we'd like the most accurate model…
- So let's make our loss function negative accuracy!

Have network output a number between 0 and 1 (i.e., use a sigmoid activation function). If output is >0.5, then predict 1, else 0

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…
- In classification, we'd like the most accurate model…
- So let's make our loss function negative accuracy!

Have network output a number between 0 and 1 (i.e., use a sigmoid activation function). If output is >0.5, then predict 1, else 0

For most small changes in weights, the output class is unaffected

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…
- In classification, we'd like the most accurate model…
- So let's make our loss function negative accuracy!

Have network output a number between 0 and 1 (i.e., use a sigmoid activation function). If output is >0.5, then predict 1, else 0

For most small changes in weights, the output class is unaffected

No change in outputs → no change in loss → 0 gradient…

What is a reasonable loss function to use?

- Loss functions are what we try to optimize…
- In classification, we'd like the most accurate model…
- So let's make our los[…]acy!

Gradients are the main motivation behind these types of decisions!

Have network output a number between 0 and 1 (i.e., use a sigmoid activation function). If output is >0.5, then predict 1, else 0

For most small changes in weights, the output class is unaffected

No change in outputs → no change in loss → 0 gradient…

What is a reasonable loss function to use?

- Accuracy is a “hard” function
  - Hard to take meaningful derivatives of
- Other examples:
  - Max vs. Softmax
  - Ranking vs Softrank
  - Sign function (i.e., perceptron activation) vs. Softsign
  - Argmax

What is a reasonable loss function to use?

- Accuracy is a “hard” function
  - Hard to take meaningful derivatives of
- Other examples:
  - Max vs. Softmax
  - Ranking vs Softrank
  - Sign function (i.e., perceptron activation) vs. Softsign
  - Argmax

My (somewhat) old research

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x)\, \log\left(\frac{P(x)}{Q(x)}\right)$$

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

Defined for two probability distributions, P and Q

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

Defined for two probability distributions, P and Q

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Think of Q as what we predict and P as the ground truth Probabilities

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

Defined for two probability distributions, P and Q

When P(x) is high, Q(x) should also be high... (Log(1) = 0)

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \, \log\left(\frac{P(x)}{Q(x)}\right)$$

Think of Q as what we predict and P as the ground truth Probabilities

One-Hot Vectors Revisited

datagy.io

Island	Biscoe	Dream	Torgensen
Biscoe	1	0	0
Torgensen	0	0	1
Dream	0	1	0

One-Hot Vectors Revisited

datagy.io

Island	Biscoe	Dream	Torgensen
Biscoe	1	0	0
Torgensen	0	0	1
Dream	0	1	0

Can be interpreted as a probability!

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \, \log\left(\frac{P(x)}{Q(x)} \right)$$

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

“Ground truth” Probability (i.e., label)

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

“Ground truth” Probability (i.e., label)

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Predicted probabilities

Kullback–Leibler divergence

- One type of statistical distance
  - Distance between two probability distributions

"Ground truth" Probability (i.e., label)

P(x) is 1 if x is correct label, 0 otherwise

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Predicted probabilities

Binary Cross Entropy

KL Divergence

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Cross Entropy (CE)

$$CE(y, \hat{y}) = -\sum_{i}^{n} y_i \log \hat{y}_i$$

Binary Cross Entropy

KL Divergence

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Cross Entropy (CE)

$$CE(y, \hat{y}) = -\sum_{i}^{n} y_i \log \hat{y}_i$$

”Categorical Cross Entropy”

Binary Cross Entropy

KL Divergence

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \, \log\left(\frac{P(x)}{Q(x)}\right)$$

Cross Entropy (CE)

$$CE(y, \hat{y}) = -\sum_{i}^{n} y_i \log \hat{y}_i$$

”Categorical Cross Entropy”

For Binary problems “Binary Cross Entropy” (BCE)

Derivative of Cross Entropy

$$\frac{dL}{d\hat{y}} = -\frac{d}{d\hat{y}}\sum_{i}^{n} y_i \log \hat{y}_i$$

Derivative of Cross Entropy

$$\frac{dL}{d\hat{y}} = -\frac{d}{d\hat{y}}\sum_{i}^{n} y_i \log \hat{y}_i$$

What is this? (vector, scalar, matrix)

Derivative of Cross Entropy

$$\frac{dL}{d\hat{y}} = -\frac{d}{d\hat{y}}\sum_{i}^{n} y_i \log \hat{y}_i$$

What is this? (vector, scalar, matrix)

What is this? (vector, scalar, matrix)

Derivative of Cross Entropy

$$\frac{dL}{d\hat{y}} = -\frac{d}{d\hat{y}}\sum_{i}^{n} y_i \log \hat{y}_i$$

What is this? (vector, scalar, matrix)

What is this? (vector, scalar, matrix)

What is this? (vector, scalar, matrix)

Derivative of Cross Entropy

$$\frac{dL}{d\hat{y}} = -\frac{d}{d\hat{y}} \sum_{i}^{n} y_i \log \hat{y}_i$$

$$\frac{dL}{d\hat{y}} = -\sum_{i}^{n} -\frac{1}{p_i}$$

Probability of predicting correct label for example i

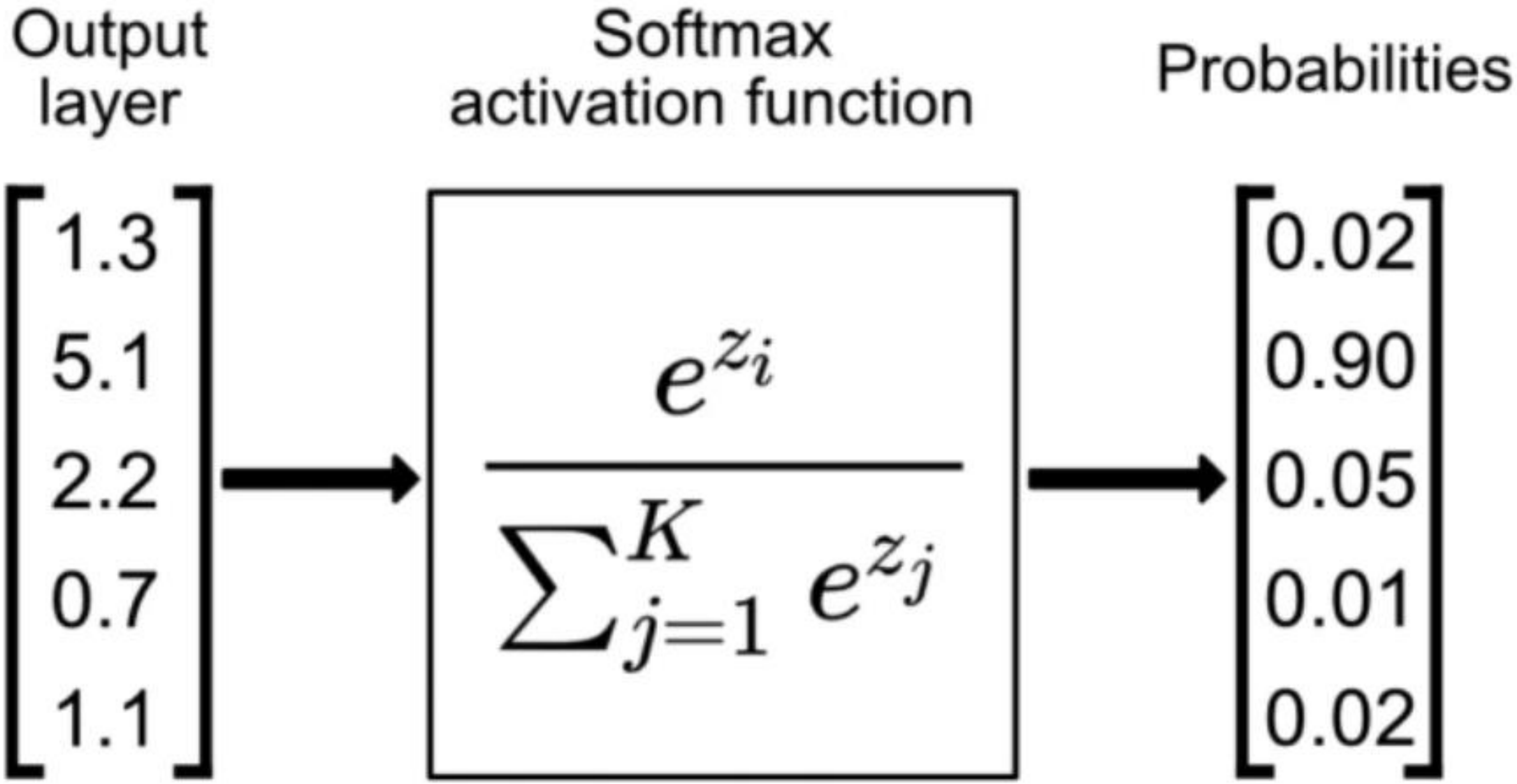
Probabilities

- If we have probabilities, we can use Cross Entropy
- How do we get probabilities?

Option #1: Normalize outputs (i.e., divide by their total)

Option #2: Use another function (i.e., softmax)

Softmax Function



Source: https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/

What’s the difference?

Consider a neural network with 2 outputs.

For one image, the network outputs [1, 2]. For a second image, the network outputs [10, 20].

What will be the predicted probabilities with normalization?

What’s the difference?

Consider a neural network with 2 outputs.

For one image, the network outputs [1, 2]. For a second image, the network outputs [10, 20].

What will be the predicted probabilities with normalization?

[1/3, 2/3] for both examples

What’s the difference?

Consider a neural network with 2 outputs.

For one image, the network outputs [1, 2]. For a second image, the network outputs [10, 20].

What will be the predicted probabilities with Softmax?

What's the difference?

Consider a neural network with 2 outputs.

For one image, the network outputs [1, 2]. For a second image, the network outputs [10, 20].

What will be the predicted probabilities with Softmax?

[0.26, 0.73] for [1, 2]
[0.00005, 0.99995] for [10, 20]

What’s the difference?

Add 10 to each output

Consider a neural network with 2 outputs.

For one image, the network outputs [11, 12]. For a second image, the network outputs [20, 30].

What will be the predicted probabilities with Normalization?

What's the difference?

Add 10 to each output

Consider a neural network with 2 outputs.

For one image, the network outputs [11, 12]. For a second image, the network outputs [20, 30].

What will be the predicted probabilities with Normalization?

[0.47, 0.53] for [11, 12]
[0.4, 0.6] for [20, 30]

What’s the difference?

Add 10 to each output

Consider a neural network with 2 outputs.

For one image, the network outputs [11, 12]. For a second image, the network outputs [20, 30].

What will be the predicted probabilities with Softmax?

What’s the difference?

Add 10 to each output

Consider a neural network with 2 outputs.

For one image, the network outputs [11, 12]. For a second image, the network outputs [20, 30].

What will be the predicted probabilities with Softmax?

[0.26, 0.73] for [11, 12]
[0.00005, 0.99995] for [20, 30]
Exactly the same as [1, 2] and [10, 20]

What’s the difference?

What’s the difference?

Normalization is sensitive to additive changes, but not multiplicative changes

What’s the difference?

Normalization is sensitive to additive changes, but not multiplicative changes

Softmax is sensitive to multiplicative changes, but not additive

What’s the difference?

Normalization is sensitive to additive changes, but not multiplicative changes

Softmax is sensitive to multiplicative changes, but not additive

Softmax also has other advantages:

What's the difference?

Normalization is sensitive to additive changes, but not multiplicative changes

Softmax is sensitive to multiplicative changes, but not additive

Softmax also has other advantages:

- Tends to handle smaller probabilities better (less float underflow)

What's the difference?

Normalization is sensitive to additive changes, but not multiplicative changes

Softmax is sensitive to multiplicative changes, but not additive

Softmax also has other advantages:

- Tends to handle smaller probabilities better (less float underflow)
- Remember that log in our loss function? Remember the e^z in softmax? Our loss function becomes ~linear for our neuron outputs z

What’s the difference?

Normalization is sensitive to additive changes, but not multiplicative changes

Softmax is sensitive to multiplicative changes, but not additive

Softmax also has other advantages:

- Tends to handle smaller probabilities better (less float underflow)
- Remember that log in our loss function? Remember the e^z in softmax? Our loss function becomes ~linear for our neuron outputs z
- Maybe has issues with overflow… (outputs can become inf or NaN)

Derivative of Softmax

$$a_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Want to know: $\frac{da}{dz}$

Derivative of Softmax

$$a_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Want to know: $\frac{da}{dz}$

What is this? (vector, scalar, matrix)

Derivative of Softmax

$$a_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Want to know: $\frac{da}{dz}$

a and *z* are both vectors, therefore $\frac{da}{dz}$ *is a Jacobian matrix*

Derivative of Softmax

$$a_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Want to know: $\frac{da}{dz}$

What is $\frac{\partial a_i}{\partial z_j}$?

Derivative of Softmax

$$a_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Quotient rule!

Want to know: $\frac{da}{dz}$

What is $\frac{\partial a_i}{\partial z_j}$?

Derivative of Softmax

$$a_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

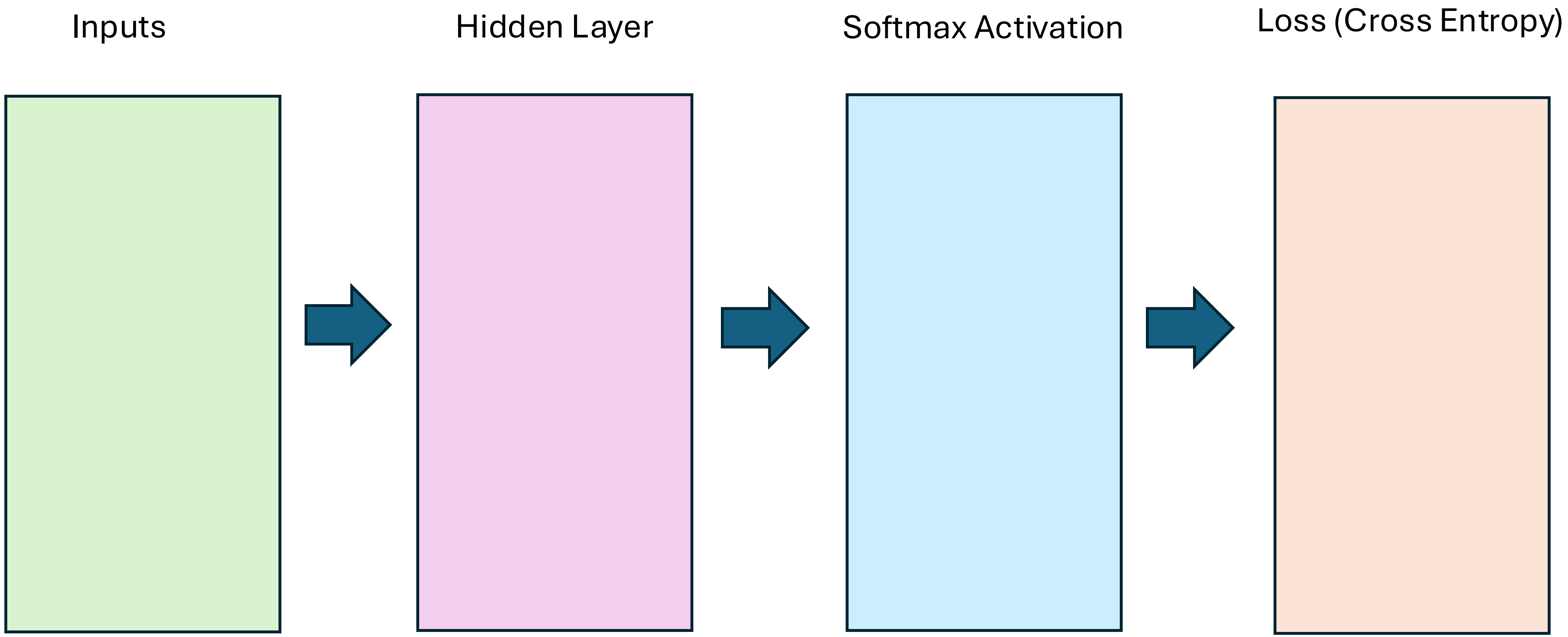
Want to know: $\frac{da}{dz}$

What is $\frac{\partial a_i}{\partial z_j}$?

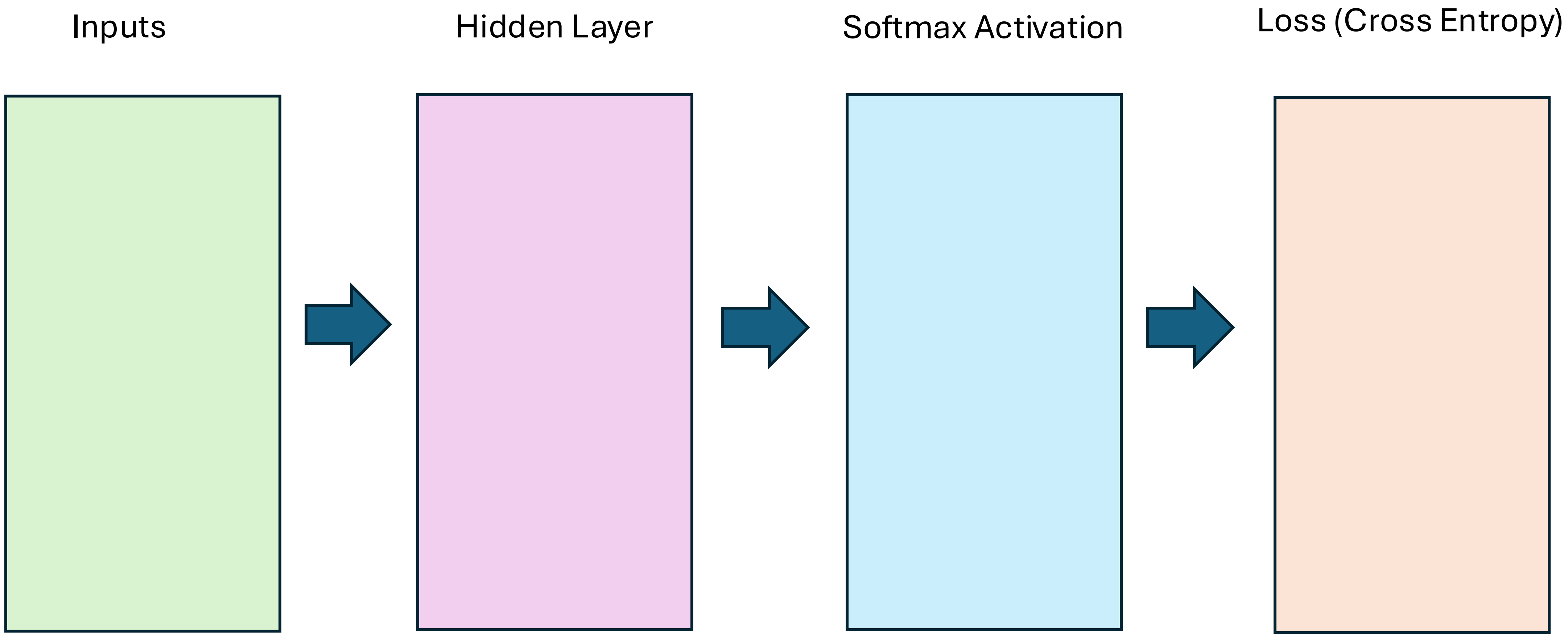
If $i == j$, then $\frac{\partial a_i}{\partial z_i}$= $a_i \cdot (1 - a_i)$

If $i! = j$, then $\frac{\partial a_i}{\partial z_j}$= $-a_i \cdot a_j$

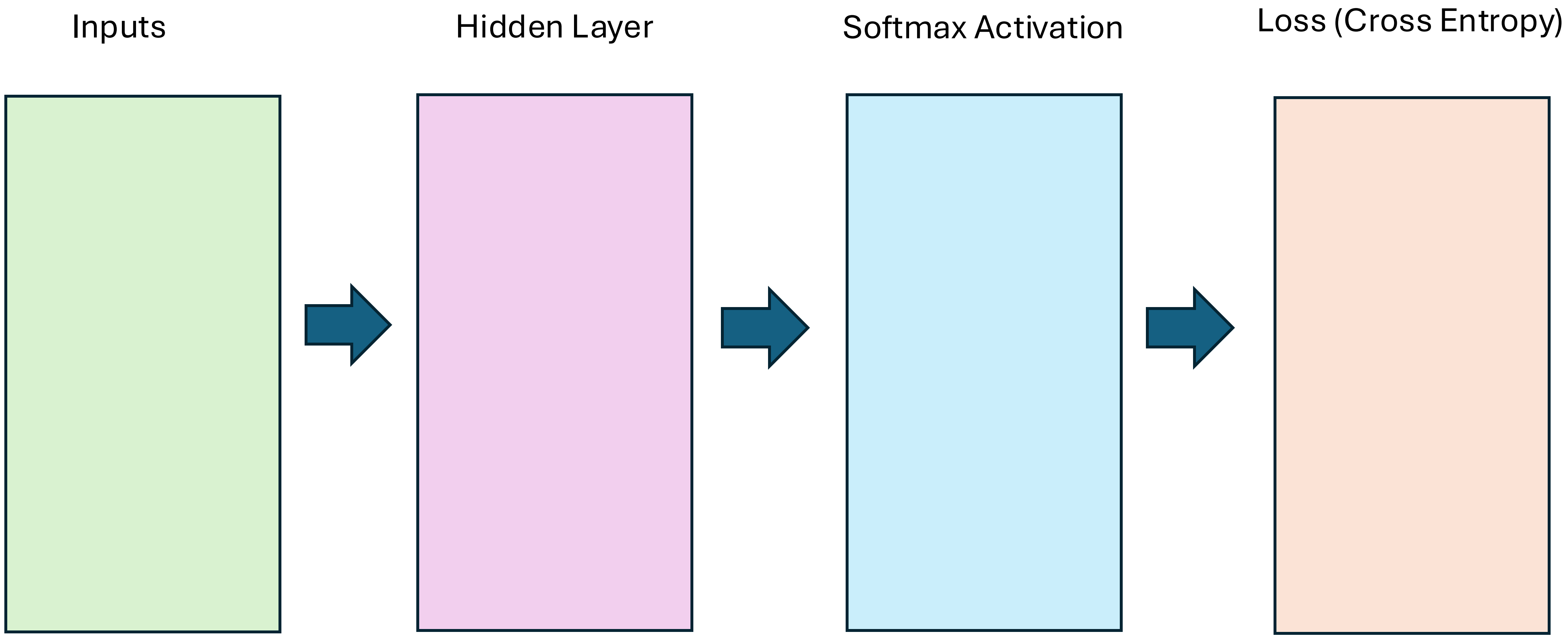
A Full Classification Network



A Full Classification Network



A Full Classification Network



What is this? (vector, scalar, matrix)

A Full Classification Network

Inputs

Inputs are a matrix of *n* examples with *d* features

Hidden Layer

Hidden layer with *m* neurons will have a weight matrix of *n x m*

Softmax Activation

Softmax has *m* inputs and *m* outputs

Loss (Cross Entropy)

Loss is a scalar value

A Full Classification Network

Inputs

Inputs are a matrix of *n* examples with *d* features

X

Hidden Layer

Hidden layer with *m* neurons will have a weight matrix of *n x m*

z

Softmax Activation

Softmax has *m* inputs and *m* outputs

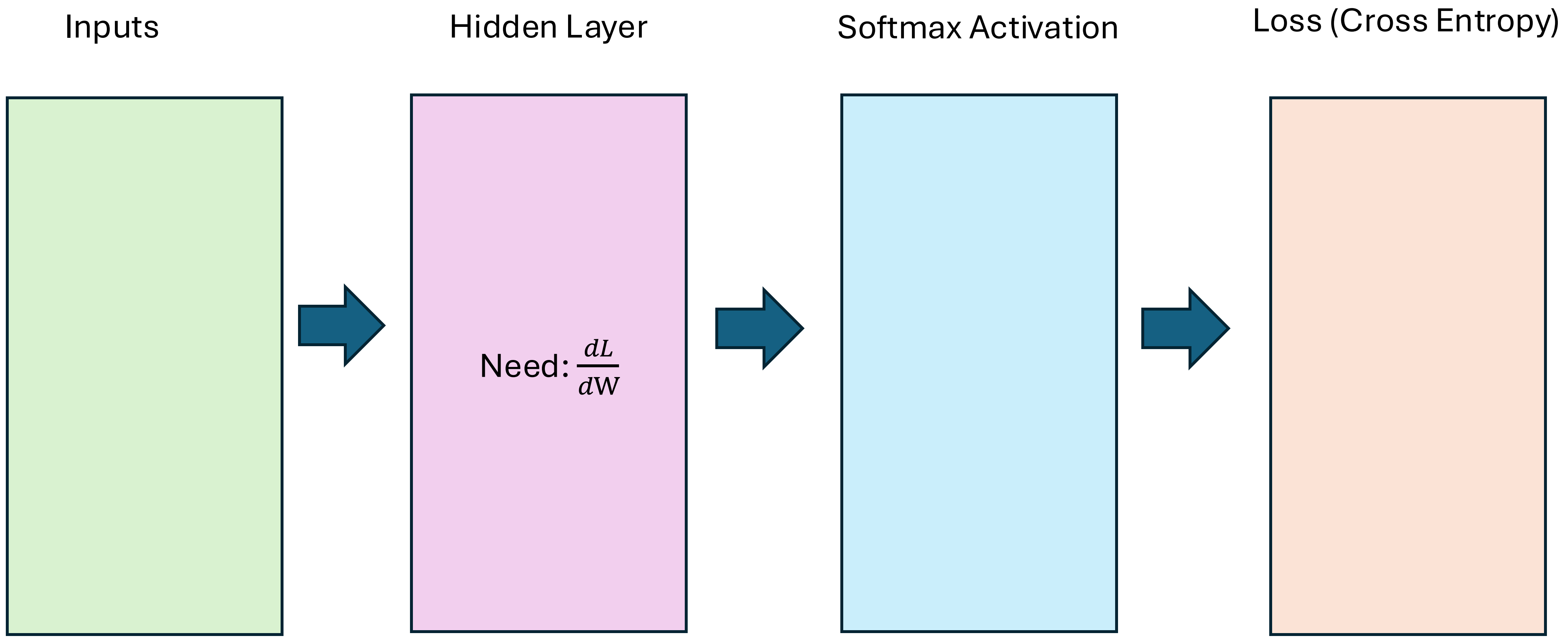
a

Loss (Cross Entropy)

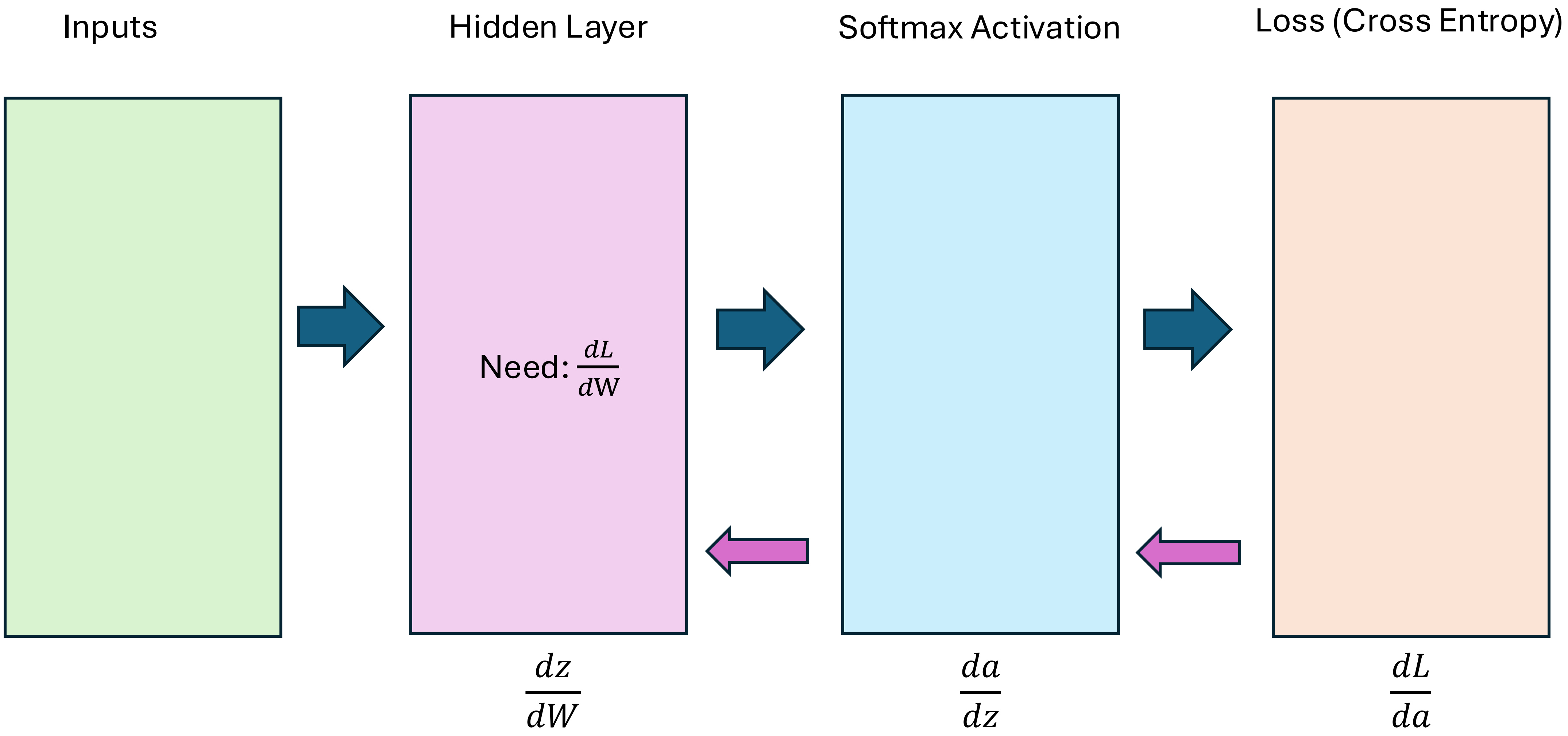
Loss is a scalar value

$\hat{y}$

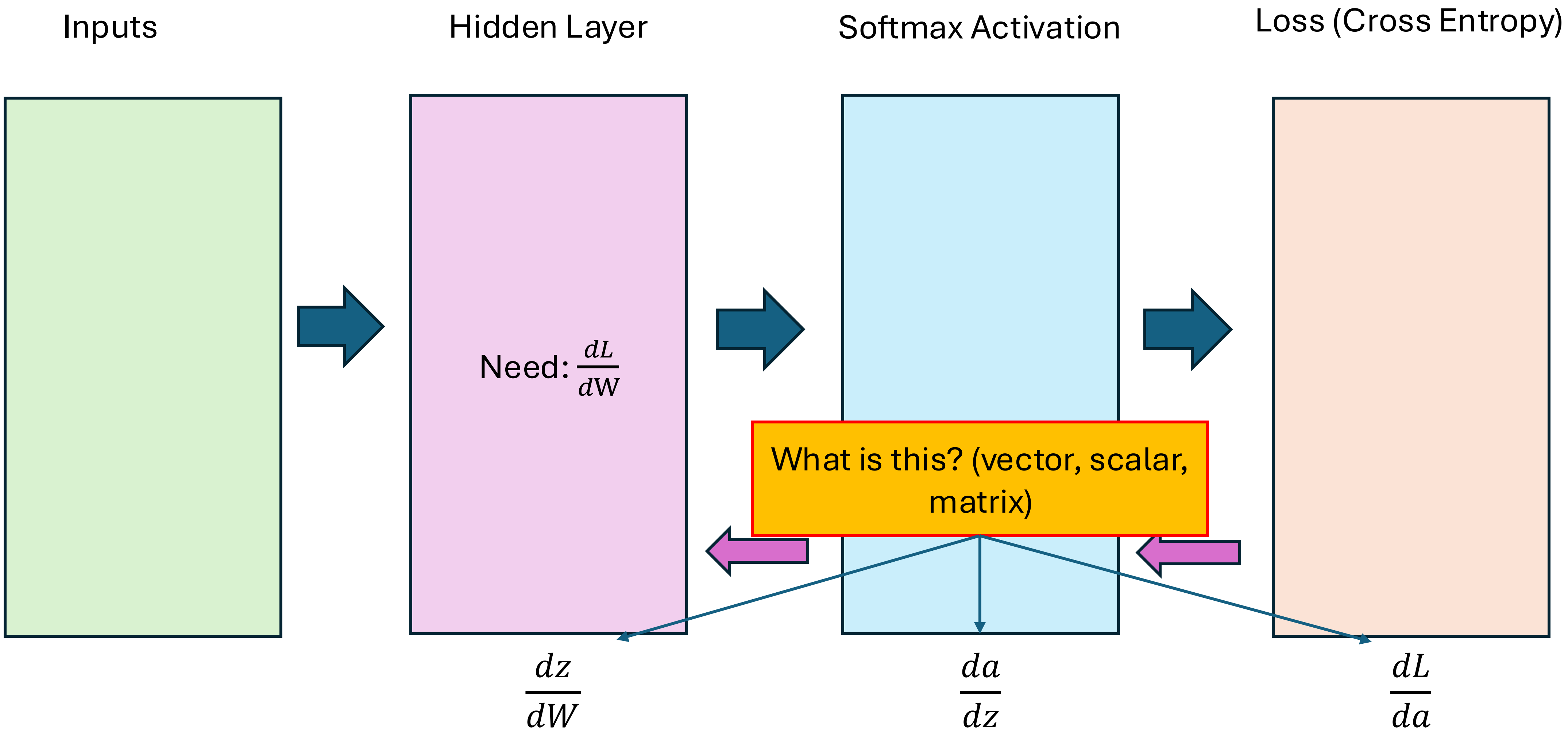
A Full Classification Network



A Full Classification Network



A Full Classification Network

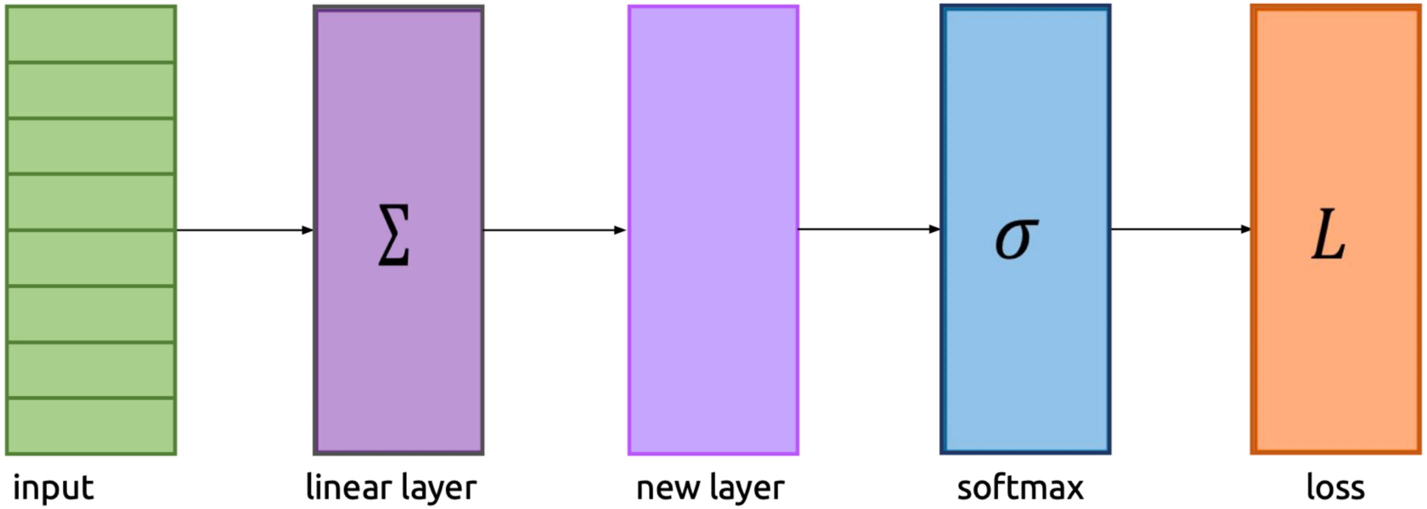


Generalizing Backpropagation

- What if we want to add another layer to our model?
- Calculating derivatives by hand ***again*** is a lot of work ☹️

Can the computers do this for us?

Yes ☺️



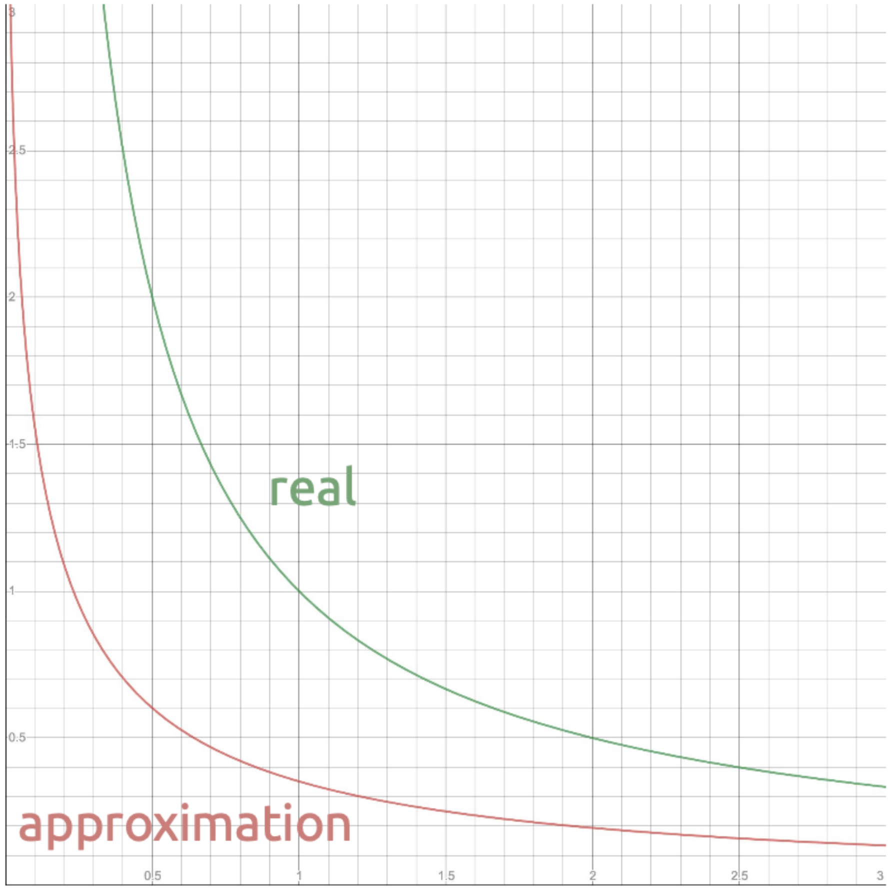
Computer-based Derivatives

- **Numeric differentiation**
  - $\frac{df}{dx} \approx \frac{f(x+\Delta x)-f(x)}{\Delta x}$
  - Pick a small step size Δx
  - Also called "finite differences"

Computer-based Derivatives

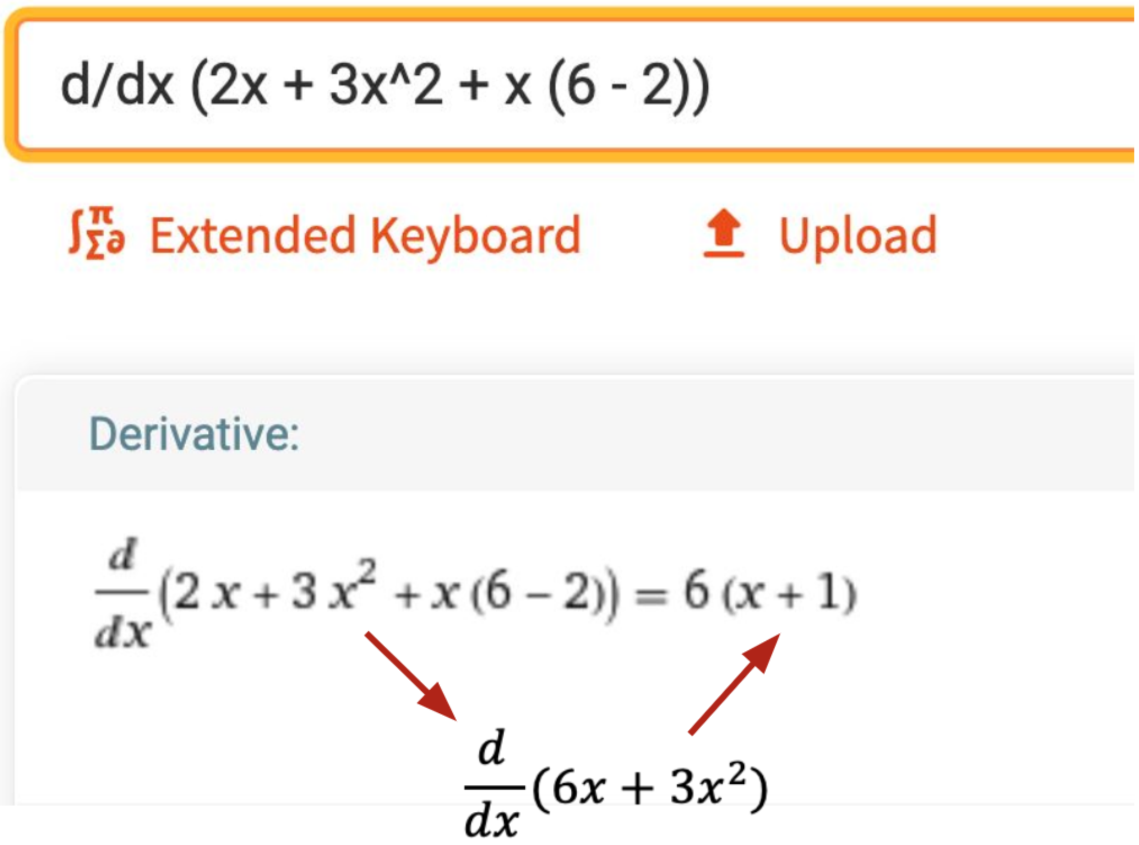
- **Numeric differentiation**
  - $\frac{df}{dx} \approx \frac{f(x+\Delta x)-f(x)}{\Delta x}$
  - Pick a small step size Δx
  - Also called "finite differences"
  - Easy to implement
  - Arbitrarily inaccurate/unstable

$y = \frac{1}{x}$ $\Delta x = 0.5$



Computer-based Derivatives

- Numeric differentiation
- **Symbolic differentiation**
  - Computer "does algebra" and simplifies expressions
  - What Wolfram Alpha does https://www.wolframalpha.com/



Computer-based Derivatives

- Numeric differentiation
- **Symbolic differentiation**
  - Computer "does algebra" and simplifies expressions
  - What Wolfram Alpha does
  - Exact (no approximation error)
  - Complex to implement
  - Only handles static expressions (what about e.g. loops?)

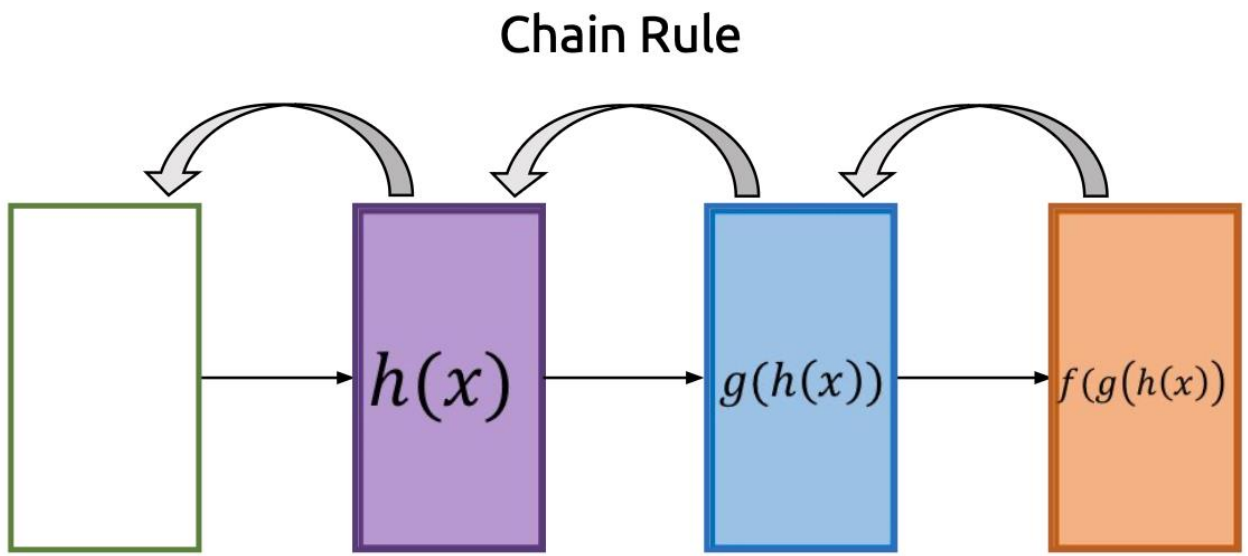
- Example:

```
while abs(x) > 5:
    x = x / 2
```

- This loop could run once or 100 times, it's impossible to know

Computer-based Derivatives

- Numeric differentiation
- Symbolic differentiation
- **Automatic differentiation**
  - Use the chain rule at runtime



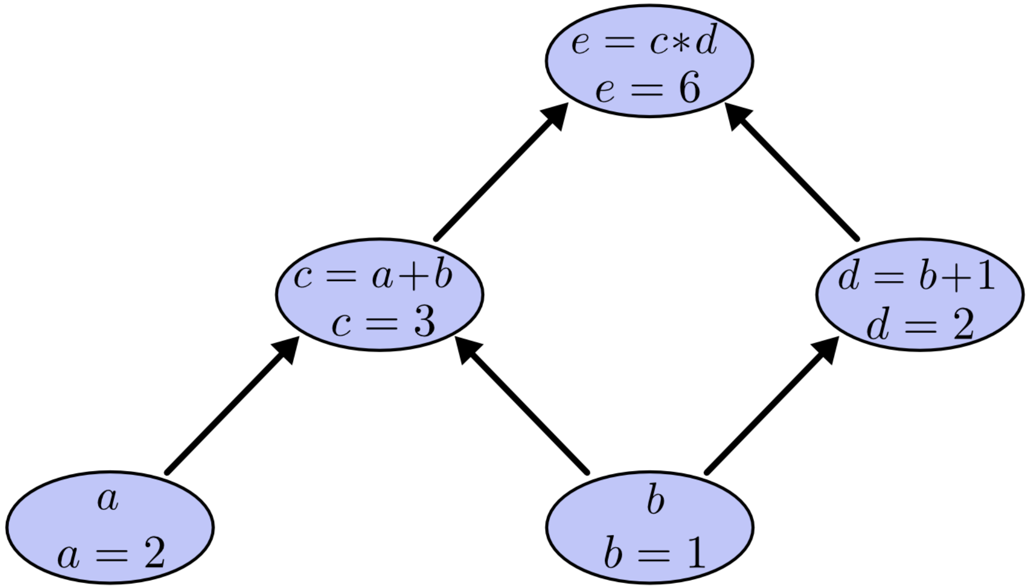
Computer-based Derivatives

- Numeric differentiation
- Symbolic differentiation
- **Automatic differentiation**
  - Use the chain rule at runtime
  - Gives exact results
  - Handles dynamics (loops, etc.)
  - Easier to implement
  - Can't simplify expressions

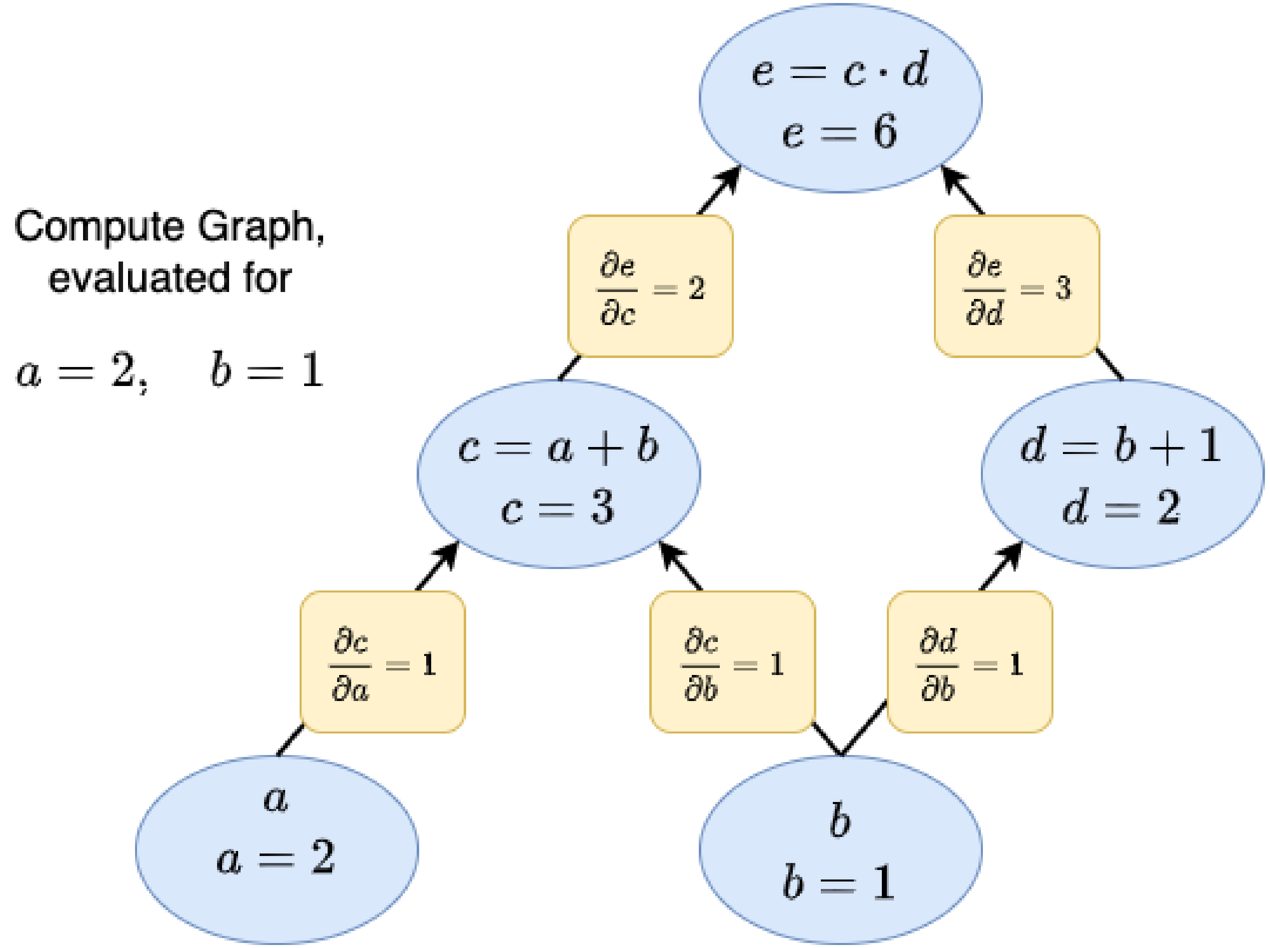
- $\sin^2 x + \cos^2 x \Rightarrow 1$
- Automatic differentiation doesn't know this identity, will end up evaluating the entire expression on the left hand side

Computation Graph

$e = (a + b) \cdot (b + 1)$



Computation Graph and Derivatives



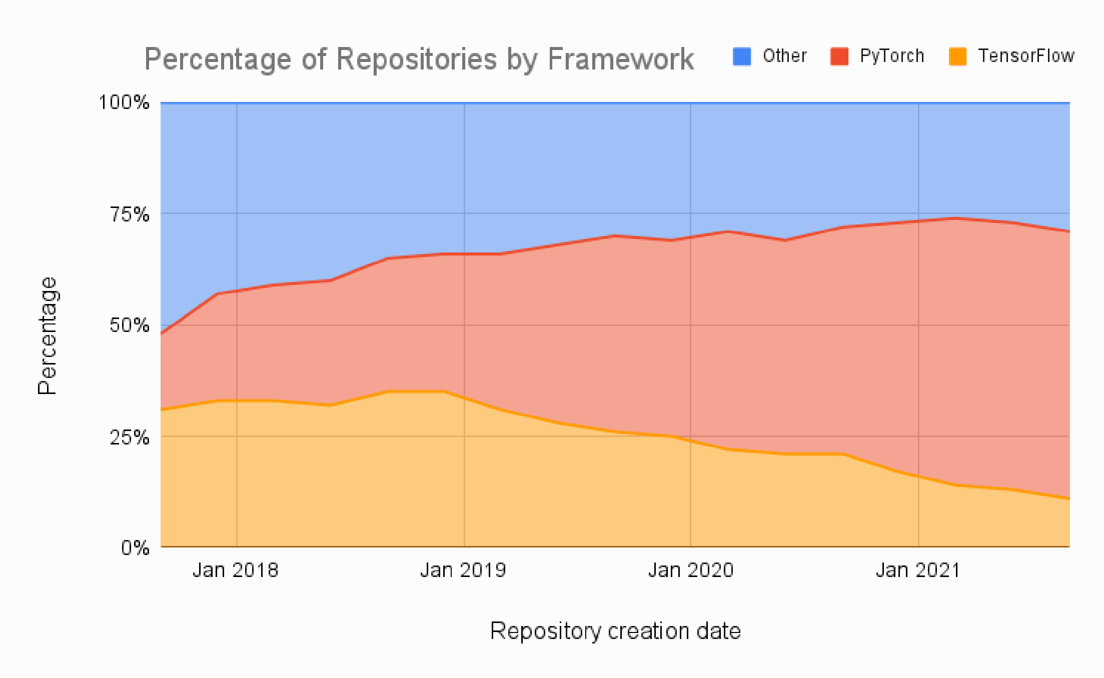
https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

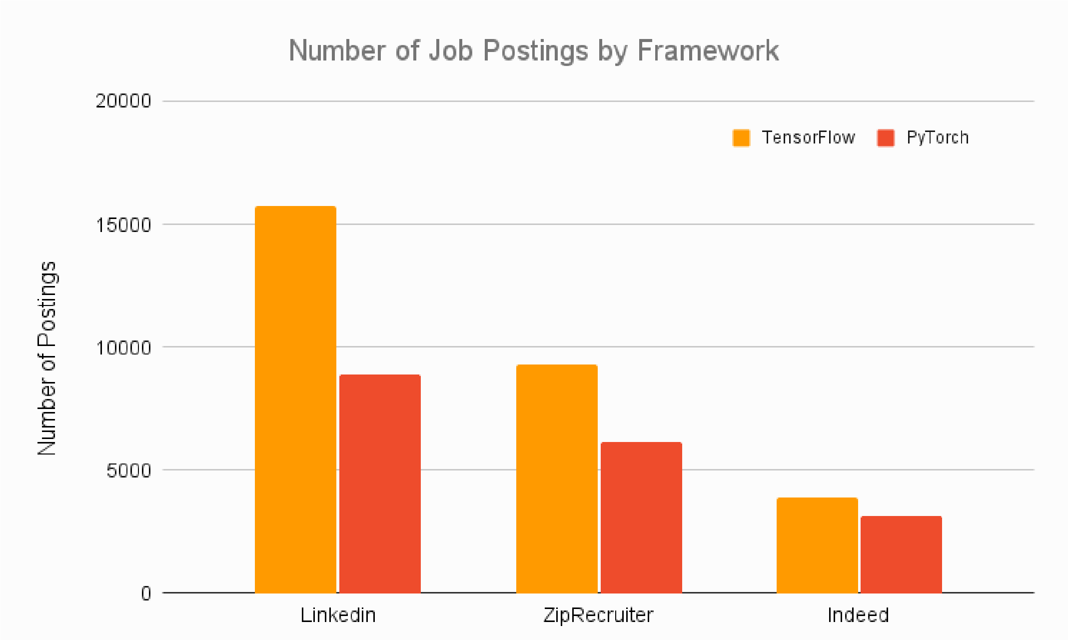
Tensorflow Gradient Tape

- Tensorflow will maintain a compute graph of operations performed within a Gradient Tape context
  - Can automatically differentiate operations on request
- This is the purpose and usefulness of Deep Learning Frameworks!
- For the most part, you only have to specify the forward operations and TF (or Torch/Jax) will take care of the rest.

DL Frameworks

- Main current frameworks are Tensorflow, Pytorch, and Jax
- TF and torch are becoming increasingly similar in style and performance
- Jax is new and different





https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/

Tensorflow

Tensorflow

- Developed and maintained by Google

Tensorflow

- Developed and maintained by Google
- In addition to autodiff features it also provides:
  - Many common functions (i.e., Softmax, Sigmoid, Cross Entropy, etc.)
  - An easy way to train models (Keras)
  - Strong support for hardware acceleration (i.e., if you have a GPU, TF will figure out how to use it)

Tensorflow

- Developed and maintained by Google
- In addition to autodiff features it also provides:
  - Many common functions (i.e., Softmax, Sigmoid, Cross Entropy, etc.)
  - An easy way to train models (Keras)
  - Strong support for hardware acceleration (i.e., if you have a GPU, TF will figure out how to use it)
- “Easier to deploy to production” (has been the general consensus previously, but other frameworks have caught up)

Tensorflow

- Developed and maintained by Google
- In addition to autodiff features it also provides:
    - Many common functions (i.e., Softmax, Sigmoid, Cross Entropy, etc.)
    - An easy way to train models (Keras)
    - Strong support for hardware acceleration (i.e., if you have a GPU, TF will figure out how to use it)
- "Easier to deploy to production" (has been the general consensus previously, but other frameworks have caught up)
- TF lite for on device applications (e.g., phones)

Pytorch

Pytorch

- Developed by Facebook AI (now Meta)

Pytorch

- Developed by Facebook AI (now Meta)
- More common in the research and academic community

Pytorch

- Developed by Facebook AI (now Meta)
- More common in the research and academic community
- “More flexible” and easier to write custom backward passes

Pytorch

- Developed by Facebook AI (now Meta)
- More common in the research and academic community
- “More flexible” and easier to write custom backward passes
- No Gradient Tape, each tensor (matrix/vector) is “trainable” or not. If a tensor is trainable then all operations on it are tracked.

Pytorch

- Developed by Facebook AI (now Meta)
- More common in the research and academic community
- "More flexible" and easier to write custom backward passes
- No Gradient Tape, each tensor (matrix/vector) is "trainable" or not. If a tensor is trainable then all operations on it are tracked.
- Slightly more work to use GPUs or other hardware

Pytorch

- Developed by Facebook AI (now Meta)
- More common in the research and academic community
- “More flexible” and easier to write custom backward passes
- No Gradient Tape, each tensor (matrix/vector) is “trainable” or not. If a tensor is trainable then all operations on it are tracked.
- Slightly more work to use GPUs or other hardware
- Harder to track stats
  - (I still use TF’s tensorboard stat tracker when using Pytorch)

Pytorch

- Developed by Facebook AI (now Meta)
- More common in the research and academic community
- “More flexible” and easier to write custom backward passes
- No Gradient Tape, each tensor (matrix/vector) is “trainable” or not. If a tensor is trainable then all operations on it are tracked.
- Slightly more work to use GPUs or other hardware
- Harder to track stats
  - (I still use TF’s tensorboard stat tracker when using Pytorch)
- Easier to learn and use than tensorflow
  - Better error reporting, training code is harder to write but easier to debug

Jax

Jax

- Also developed by Google…

Jax

- Also developed by Google…
- Very new compared to Pytorch and Tensorflow

Jax

- Also developed by Google…
- Very new compared to Pytorch and Tensorflow
- ***Much Faster***

Jax

- Also developed by Google…
- Very new compared to Pytorch and Tensorflow
- ***Much Faster***
- Takes advantage of Just In Time (JIT) compiling to speed up execution

Jax

- Also developed by Google…
- Very new compared to Pytorch and Tensorflow
- ***Much Faster***
- Takes advantage of Just In Time (JIT) compiling to speed up execution
- Best used with functional programming

Recap

Gradients Matter! We can't use accuracy as a loss function because it has 0 gradient most places

What is this? (vector, scalar, matrix)
It matters!

DL frameworks maintain compute graphs and can differentiate composable functions automatically