Lecture 4: Neural Networks and Backpropagation

Stanford CS231n 10th Anniversary

Lecture 4 - 1

Administrative: Project Proposal

Due Fri 4/25

TA expertise is posted on the webpage.

(http://cs231n.stanford.edu/office_hours.html)

Stanford CS231n 10th Anniversary

Lecture 4 - 3

Administrative: Discussion Section

Discussion section tomorrow

(led by Matthew Jin, With Emily Jin's help):

Lecture 4 - 4

April 10, 2025

Backpropagation



Stanford CS231n 10th Anniversary

Lecture 4 - 5

Recap

- We have some dataset of (x,y)
- We have a score function:
- We have a loss function:

$$s=f(x;W)\stackrel{ ext{e.g.}}{=}Wx$$



Finding the best W: Optimize with Gradient Descent





April 10, 2025

Vanilla Gradient Descent

while True:

weights_grad = evaluate_gradient(loss_fun, data, weights)
weights += - step size * weights grad # perform parameter update

Landscape image is CC0 1.0 public domain Walking man image is CC0 1.0 public domain

Stanford CS231n 10th Anniversary

Gradient descent

$$rac{df(x)}{dx} = \lim_{h o 0} rac{f(x+h) - f(x)}{h}$$

Numerical gradient: slow ⊗, approximate ⊗, easy to write ☺ Analytic gradient: fast ☺, exact ☺, error-prone ⊗

In practice: Derive analytic gradient, check your implementation with numerical gradient

Stanford CS231n 10th Anniversary

Lecture 4 - 8

<u>April 10, 2025</u>

Stochastic Gradient Descent (SGD)

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$
$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum is expensive when N is large!

Approximate sum using a minibatch of examples 32 / 64 / 128 / 256

April 10, 2025

```
# Vanilla Minibatch Gradient Descent
while True:
    data_batch = sample_training_data(data, 256) # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad # perform parameter update
```

Stanford CS231n 10th Anniversary



Stanford CS231n 10th Anniversary

Lecture 4 - 10

Last time: learning rate scheduling



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine:
$$\alpha_t = \frac{1}{2}\alpha_0 \left(1 + \cos(t\pi/T)\right)$$

Linear: $\alpha_t = \alpha_0 (1 - t/T)$
Inverse sort: $\alpha_t = \alpha_0 / \sqrt{t}$

 $lpha_0$: Initial learning rate $lpha_t$: Learning rate at epoch t T : Total number of epochs

Stanford CS231n 10th Anniversary

Lecture 4 - 11

Today:

Deep Learning

Stanford CS231n 10th Anniversary

Lecture 4 - 12

DALL-E 2



"Teddy bears working on new AI research on the moon in the 1980s." "Rabbits attending a college seminar on human anatomy."

"A wise cat meditating in the Himalayas searching for enlightenment."

April 10, 2025

Image source: Sam Altman, https://openai.com/dall-e-2/, https://twitter.com/sama/status/1511724264629678084

Stanford CS231n 10th Anniversary



vibrant portrait painting of Salvador Dalí with a robotic half face

a close up of a handpalm with leaves growing from it





an espresso machine that makes coffee from human souls, artstation



a dolphin in an astronaut suit on saturn, artstation





Stanford CS231n 10th Anniversary

napoleon holding a piece of cheese



a teddybear on a skateboard in times square



Ramesh et al., Hierarchical Text-Conditional Image Generation with CLIP Latents, 2022.

Lecture 4 - 14

DALL-E 3

In a fantastical setting, a highly detailed furry humanoid skunk with piercing eyes confidently poses in a medium shot, wearing an animal hide jacket. The artist has masterfully rendered the character in digital art, capturing the intricate details of fur and clothing texture.



Betker, James, et al. "Improving image generation with better captions." Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf (2023).

Stanford CS231n 10th Anniversary

Lecture 4 - 15

DALL-E 3

An illustration from a graphic novel. A bustling city street under the shine of a full moon. The sidewalks bustling with pedestrians enjoying the nightlife. At the comer stall, a young woman with fiery red hair, dressed in a signature velvet cloak, is haggling with the grumpy old vendor. The grumpy vendor, a tall, sophisticated man wearing a sharp suit, who sports a noteworthy mustache is animatedly conversing on his steampunk telephone.

Betker, James, et al. "Improving image generation with better captions." Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf (2023). The sidewalks bustling with pedestrians enjoying the nightlife.

A bustling city street under the shine of a **full moon.**



At the corner stall, a **young woman** with fiery red hair, dressed in a signature velvet cloak, is **haggling with the grumpy old vendor.**

The grumpy vendor, a **tall, sophisticated man,** is wearing a sharp suit, sports a **noteworthy moustache** and is animatedly conversing on his **steampunk telephone**.

Stanford CS231n 10th Anniversary

Lecture 4 - 16

GPT-4

User What is unusual about this image?



Source: Barnorama

GPT-4 The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi.

User Can you explain this meme? Sometimes I just look at pictures of the earth from space and I marvel at how beautiful it all is.



GPT-4 This meme is a joke that combines two unrelated things: pictures of the earth from space and chicken nuggets.

The text of the meme suggests that the image below is a beautiful picture of the earth from space. However, the image is actually of chicken nuggets arranged to vaguely resemble a map of the world.

The humor in this meme comes from the unexpected juxtaposition of the text and the image. The text sets up an expectation of a majestic image of the earth, but the image is actually something mundane and silly.

Image source: https://openai.com/research/gpt-4

Stanford CS231n 10th Anniversary

Lecture 4 - 17

Segment Anything Model (SAM)



Stanford CS231n 10th Anniversary

Lecture 4 - 18



Sora

- Animating Images (generated by DALL-E)
- Video-to-video editing



A Shiba Inu dog wearing a beret and black turtleneck.





put the video in space with a rainbow road



change the video setting to be different than a mountain? perhaps joshua tree

April 10, 2025

https://openai.com/research/video-generation-models-as-world-simulators

Stanford CS231n 10th Anniversary

Sora

• More compute



Base Compute



4x Compute



April 10, 2025

32x Compute

https://openai.com/research/video-generation-models-as-world-simulators

Stanford CS231n 10th Anniversary

Neural Networks

Stanford CS231n 10th Anniversary

Lecture 4 - 22

Neural networks: the original linear classifier

(Before) Linear score function:

$$f = Wx$$

 $x \in \mathbb{R}^D, W \in \mathbb{R}^{C \times D}$

Stanford CS231n 10th Anniversary

Lecture 4 - 23

Neural networks: 2 layers

(Before) Linear score function: f = W x(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

(In practice we will usually add a learnable bias at each layer as well)

April 10, 2025

Stanford CS231n 10th Anniversary

Why do we want non-linearity?



Cannot separate red and blue points with linear classifier

Stanford CS231n 10th Anniversary

Lecture 4 - 25

Why do we want non-linearity?

 $f(x, y) = (r(x, y), \theta(x, y))$



Cannot separate red and blue points with linear classifier After applying feature transform, points can be separated by linear classifier

θ

Stanford CS231n 10th Anniversary

Lecture 4 - 26

r

Neural networks: also called fully connected network

(Before) Linear score function: f = W x(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

"Neural Network" is a very broad term; these are more accurately called "fully-connected networks" or sometimes "multi-layer perceptrons" (MLP)

(In practice we will usually add a learnable bias at each layer as well)

April 10, 2025

Stanford CS231n 10th Anniversary

Neural networks: 3 layers

(Before) Linear score function: f = Wx(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$ or 3-layer Neural Network $f = W_3 \max(0, W_2 \max(0, W_1 x))$

$$x \in \mathbb{R}^{D}, W_1 \in \mathbb{R}^{H_1 \times D}, W_2 \in \mathbb{R}^{H_2 \times H_1}, W_3 \in \mathbb{R}^{C \times H_2}$$

(In practice we will usually add a learnable bias at each layer as well)

April 10, 2025

Stanford CS231n 10th Anniversary

Neural networks: hierarchical computation



Stanford CS231n 10th Anniversary

Lecture 4 - 29





Learn 100 templates instead of 10.

Share templates between classes

Stanford CS231n 10th Anniversary

Lecture 4 - 30

Neural networks: why is max operator important?

(Before) Linear score function:
$$f = Wx$$

(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

The function max(0, z) is called the activation function. Q: What if we try to build a neural network without one?

$$f = W_2 W_1 x$$

Neural networks: why is max operator important?

(Before) Linear score function:
$$f = Wx$$

(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

The function max(0, z) is called the activation function. Q: What if we try to build a neural network without one?

$$f = W_2 W_1 x$$
 $W_3 = W_2 W_1 \in \mathbb{R}^{C \times H}, f = W_3 x$

Lecture 4 - 32

April 10, 2025

A: We end up with a linear classifier again!



Stanford CS231n 10th Anniversary

Lecture 4 - 33

Neural networks: Architectures



Stanford CS231n 10th Anniversary

Lecture 4 - 34

Example feed-forward computation of a neural network



forward-pass of a 3-layer neural network: f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid) x = np.random.randn(3, 1) # random input vector of three numbers (3x1) h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1) h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1) out = np.dot(W3, h2) + b3 # output neuron (1x1)

Stanford CS231n 10th Anniversary

Lecture 4 - 35

Full implementation of training a 2-layer Neural Network needs ~20 lines:

```
import numpy as np
 1
 2
    from numpy.random import randn
 3
    N, D_in, H, D_out = 64, 1000, 100, 10
 4
    x, y = randn(N, D_in), randn(N, D_out)
 5
    w1, w2 = randn(D_in, H), randn(H, D_out)
 6
 7
    for t in range(2000):
 8
      h = 1 / (1 + np.exp(-x.dot(w1)))
 9
10
      y_pred = h.dot(w2)
11
      loss = np.square(y_pred - y).sum()
      print(t, loss)
12
13
14
      grad_y_pred = 2.0 * (y_pred - y)
15
      grad_w2 = h.T.dot(grad_y_pred)
      grad_h = grad_y_pred.dot(w2.T)
16
17
      grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19
      w1 = 1e - 4 * grad w1
20
      w2 = 1e - 4 * grad w2
```

Stanford CS231n 10th Anniversary

Lecture 4 - 36

Full implementation of training a 2-layer Neural Network needs ~20 lines:

```
import numpy as np
 2
    from numpy.random import randn
 3
    N, D_in, H, D_out = 64, 1000, 100, 10
 4
    x, y = randn(N, D_in), randn(N, D_out)
 5
    w1, w2 = randn(D_in, H), randn(H, D_out)
 6
 7
    for t in range(2000):
 8
      h = 1 / (1 + np.exp(-x.dot(w1)))
 9
10
      y_pred = h.dot(w2)
11
      loss = np.square(y_pred - y).sum()
      print(t, loss)
12
13
14
      grad_y_pred = 2.0 * (y_pred - y)
15
      grad_w2 = h.T.dot(grad_y_pred)
      grad_h = grad_y_pred.dot(w2.T)
16
      grad_w1 = x.T.dot(grad_h * h * (1 - h))
17
18
19
      w1 -= 1e-4 * grad w1
      w2 -= 1e-4 * grad w2
20
```

Define the network

Stanford CS231n 10th Anniversary

Lecture 4 - 37
Full implementation of training a 2-layer Neural Network needs ~20 lines:

```
import numpy as np
 2
    from numpy.random import randn
 3
    N, D_in, H, D_out = 64, 1000, 100, 10
 4
    x, y = randn(N, D_in), randn(N, D_out)
 5
    w1, w2 = randn(D_in, H), randn(H, D_out)
 6
 7
 8
    for t in range(2000):
 9
      h = 1 / (1 + np.exp(-x.dot(w1)))
10
      y_pred = h.dot(w2)
11
      loss = np.square(y_pred - y).sum()
12
      print(t, loss)
13
14
      grad_y_pred = 2.0 * (y_pred - y)
15
      grad_w2 = h.T.dot(grad_y_pred)
      grad_h = grad_y_pred.dot(w2.T)
16
      grad_w1 = x.T.dot(grad_h * h * (1 - h))
17
18
19
      w1 -= 1e-4 * grad w1
20
      w2 = 1e - 4 * grad w2
```

Define the network

Forward pass

Stanford CS231n 10th Anniversary

Lecture 4 - 38

Full implementation of training a 2-layer Neural Network needs ~20 lines:

```
import numpy as np
    from numpy.random import randn
 3
    N, D_in, H, D_out = 64, 1000, 100, 10
 4
    x, y = randn(N, D_in), randn(N, D_out)
 5
    w1, w2 = randn(D_in, H), randn(H, D_out)
 6
 7
    for t in range(2000):
 8
      h = 1 / (1 + np.exp(-x.dot(w1)))
 9
10
      y_pred = h.dot(w2)
11
      loss = np.square(y_pred - y).sum()
      print(t, loss)
12
13
      grad_y_pred = 2.0 * (y_pred - y)
14
15
      grad_w2 = h.T.dot(grad_y_pred)
      grad_h = grad_y_pred.dot(w2.T)
16
17
      grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19
      w1 -= 1e-4 * grad w1
      w2 = 1e - 4 * grad_w2
20
```

Define the network

Forward pass

Calculate the analytical gradients

Stanford CS231n 10th Anniversary

Lecture 4 - 39

Full implementation of training a 2-layer Neural Network needs ~20 lines:

```
import numpy as np
    from numpy.random import randn
 3
    N, D_in, H, D_out = 64, 1000, 100, 10
 4
    x, y = randn(N, D_in), randn(N, D_out)
 5
    w1, w2 = randn(D_in, H), randn(H, D_out)
 6
 7
    for t in range(2000):
 8
      h = 1 / (1 + np.exp(-x.dot(w1)))
 9
10
      y_pred = h.dot(w2)
11
      loss = np.square(y_pred - y).sum()
      print(t, loss)
12
13
14
      grad_y_pred = 2.0 * (y_pred - y)
15
      grad_w2 = h.T.dot(grad_y_pred)
      grad_h = grad_y_pred.dot(w2.T)
16
17
      grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19
      w1 -= 1e-4 * grad w1
20
      w2 = 1e - 4 * grad_w2
```

Define the network

Forward pass

Calculate the analytical gradients

Gradient descent

Stanford CS231n 10th Anniversary

Lecture 4 - 40

Setting the number of layers and their sizes



more neurons = more capacity

41

Stanford CS231n 10th Anniversary

Lecture 4 -

Do not use size of neural network as a regularizer. Use stronger regularization instead:

 $\lambda = 0.001$ $\lambda = 0.01$ $\lambda = 0.1$ (Web demo with ConvNetJS: http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html)

TensorFlow Play Ground: https://playground.tensorflow.org/

$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$

42

Lecture 4 -

April 10, 2025

Stanford CS231n 10th Anniversary



This image by Fotis Bobolas is licensed under CC-BY 2.0

Stanford CS231n 10th Anniversary

Lecture 4 - 43

Impulses carried toward cell body



is licensed under <u>CC-BY 3.0</u>

Stanford CS231n 10th Anniversary

Lecture 4 - 44

Impulses carried toward cell body



Stanford CS231n 10th Anniversary

Lecture 4 - 45

Impulses carried toward cell body



Stanford CS231n 10th Anniversary

Lecture 4 - 46

Biological Neurons: Complex connectivity patterns



Neurons in a neural network: Organized into regular layers for computational efficiency



April 10, 2025

This image is CC0 Public Domain

Stanford CS231n 10th Anniversary

Biological Neurons: Complex connectivity patterns



This image is CC0 Public Domain

But neural networks with random connections can work too!



Xie et al, "Exploring Randomly Wired Neural Networks for Image Recognition", IEEE/CVF International Conference on Computer Vision 2019

Stanford CS231n 10th Anniversary

Lecture 4 - 49

Be very careful with your brain analogies!

Biological Neurons:

- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system

[Dendritic Computation. London and Hausser]

Lecture 4 - 50

Plugging in neural networks with loss functions

$$s = f(x; W_1, W_2) = W_2 \max(0, W_1 x) \quad \text{Nonlinear score function}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{Hinge Loss on predictions}$$

$$R(W) = \sum_k W_k^2 \quad \text{Regularization}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(W_1) + \lambda R(W_2) \quad \text{Total loss: data loss + regularization}$$

Stanford CS231n 10th Anniversary

Lecture 4 - 51

Problem: How to compute gradients?

$$\begin{split} s &= f(x; W_1, W_2) = W_2 \max(0, W_1 x) \quad \text{Nonlinear score function} \\ L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) & \text{Hinge Loss on predictions} \\ R(W) &= \sum_k W_k^2 \quad \text{Regularization} \\ L &= \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(W_1) + \lambda R(W_2) \quad \text{Total loss: data loss + regularization} \\ \text{If we can compute } \frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2} \text{ then we can learn } W_1 \text{ and } W_2 \end{split}$$

Stanford CS231n 10th Anniversary

Lecture 4 - 52

(Bad) Idea: Derive $abla_W L$ on paper

$$s = f(x; W) = Wx$$

$$L_{i} = \sum_{j \neq y_{i}} \max(0, s_{j} - s_{y_{i}} + 1)$$

$$= \sum_{j \neq y_{i}} \max(0, W_{j,:} \cdot x + W_{y_{i},:} \cdot x + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_{i} + \lambda \sum_{k} W_{k}^{2}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_{i}} \max(0, W_{j,:} \cdot x + W_{y_{i},:} \cdot x + 1) + \lambda \sum_{k} W_{k}^{2}$$

$$\nabla_{W}L = \nabla_{W} \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_{i}} \max(0, W_{j,:} \cdot x + W_{y_{i},:} \cdot x + 1) + \lambda \sum_{k} W_{k}^{2} \right)$$

Problem: Very tedious: Lots of matrix calculus, need lots of paper

Problem: What if we want to change loss? E.g. use softmax instead of hinge? Need to re-derive everything from scratch!

April 10, 2025

Problem: Not feasible for very complex models!

Stanford CS231n 10th Anniversary

Better Idea: Computational graphs + Backpropagation



Stanford CS231n 10th Anniversary

Lecture 4 - 54



Figure copyright Alex Krizhevsky, Ilya Suts kever, and Geoffrey Hinton, 2012. Reproduced with permission.

Stanford CS231n 10th Anniversary

Lecture 4 - 55



Figure reproduced with permission from a <u>Twitter post</u> by Andrej Karpathy.

Stanford CS231n 10th Anniversary

Lecture 4 - 56

Neural Turing Machine



Stanford CS231n 10th Anniversary

Lecture 4 -

Solution: Backpropagation

Stanford CS231n 10th Anniversary

Lecture 4 - 58

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

Stanford CS231n 10th Anniversary

Lecture 4 - 59

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



April 10, 2025

Stanford CS231n 10th Anniversary

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

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



Stanford CS231n 10th Anniversary

Lecture 4 - 61

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

e.g. x = -2, y = 5, z = -4
 $q = x + y$ $rac{\partial q}{\partial x} = 1, rac{\partial q}{\partial y} = 1$



Stanford CS231n 10th Anniversary

Lecture 4 - 62

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

e.g. x = -2, y = 5, z = -4
 $q = x + y$ $rac{\partial q}{\partial x} = 1, rac{\partial q}{\partial y} = 1$
 $f = qz$ $rac{\partial f}{\partial q} = z, rac{\partial f}{\partial z} = q$



April 10, 2025

Stanford CS231n 10th Anniversary

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

e.g. $x = -2, y = 5, z = -4$
 $q = x + y$ $\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$
 $f = qz$ $\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



April 10, 2025

Stanford CS231n 10th Anniversary

Lecture 4 - <u>64</u>

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

e.g. $x = -2, y = 5, z = -4$
 $q = x + y$ $\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$
 $f = qz$ $\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



April 10, 2025

Stanford CS231n 10th Anniversary

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

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



Lecture 4 - 66

April 10, 2025

Stanford CS231n 10th Anniversary

$$f(x, y, z) = (x + y)z$$
e.g. $x = -2, y = 5, z = -4$

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



April 10, 2025

Stanford CS231n 10th Anniversary

$$f(x, y, z) = (x + y)z$$
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: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$x \xrightarrow{-2} + q \xrightarrow{3} + f \xrightarrow{-12} + 1$$

$$z \xrightarrow{-4} \xrightarrow{3} + \overline{3} +$$

April 10, 2025

Stanford CS231n 10th Anniversary

Lecture 4 - <u>68</u>

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

e.g. $x = -2, y = 5, z = -4$
 $q = x + y$ $\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$
 $f = qz$ $\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



April 10, 2025

Stanford CS231n 10th Anniversary

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

e.g. $x = -2, y = 5, z = -4$
 $q = x + y$ $\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$
 $f = qz$ $\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



April 10, 2025

Stanford CS231n 10th Anniversary

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

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

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

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

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

$$Chain rule:$$

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

$$Upstream Local$$

x -2

gradient gradient

Stanford CS231n 10th Anniversary

Lecture 4 - 71

April 10, 2025

f -12

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

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

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

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

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

$$Chain rule:$$

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

$$Upstream Local gradient$$

x -2

Lecture 4 - 72

April 10, 2025

f -12

Stanford CS231n 10th Anniversary

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

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

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

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

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

$$Chain rule:$$

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

$$Upstream Local gradient$$

x -2

Stanford CS231n 10th Anniversary

Lecture 4 - 73

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

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

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

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

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

$$Chain rule:$$

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

$$Upstream \quad Local gradient$$

x -2

Stanford CS231n 10th Anniversary

Lecture 4 - 74


Lecture 4 - 75



Lecture 4 - 76



Lecture 4 - 77



Lecture 4 - 78



Lecture 4 - 79



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



Stanford CS231n 10th Anniversary

Lecture 4 - 81

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



Stanford CS231n 10th Anniversary

Lecture 4 - 82

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

Stanford CS231n 10th Anniversary

Lecture 4 - 83

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

Stanford CS231n 10th Anniversary

Lecture 4 - 84

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

Stanford CS231n 10th Anniversary

Lecture 4 - 85

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

Stanford CS231n 10th Anniversary

Lecture 4 - 86

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

Stanford CS231n 10th Anniversary

Lecture 4 - 87

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

Stanford CS231n 10th Anniversary

Lecture 4 - 88

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

Stanford CS231n 10th Anniversary

Lecture 4 - 89

Lecture 4 - 90

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

Stanford CS231n 10th Anniversary

Lecture 4 - 91

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

Stanford CS231n 10th Anniversary

Lecture 4 - 92

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

Stanford CS231n 10th Anniversary

Lecture 4 - 93

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

Stanford CS231n 10th Anniversary

Lecture 4 - 94

Lecture 4 - 95

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

Stanford CS231n 10th Anniversary

Lecture 4 - 96

Another exam

w0 2.00

-0.20

0.40

w1 _-3.00

w2 -3.00 0.20

xample:
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

Computational graph
representation may not be
unique. Choose one where
local gradients at each
node can be easily
expressed!
wi -3.00
x1 -2.00
(x) = -1
(x) -1.00
(x)

Stanford CS231n 10th Anniversary

Lecture 4 - 97

April 10, 2025

0.73

1.00

Another exa

x0

w1

x1

0.20

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

^{w0 2.00}
^{w0 4.00}
^{w1 -3.00}
^{w1 -1.00}
<sup>w1 -1.00}
^{w1 -1.00}
<sup>w1 -1.00}
^{w1 -1.00}}</sup></sup>

 $rac{d\sigma(x)}{dx} = rac{e^{-x}}{\left(1+e^{-x}
ight)^2} = \left(rac{1+e^{-x}-1}{1+e^{-x}}
ight) \left(rac{1}{1+e^{-x}}
ight) = \left(1-\sigma(x)
ight) \sigma(x)$ Sigmoid local gradient:

Stanford CS231n 10th Anniversary

Lecture 4 - 98

Another exam

w0 2.00

-0.20

0.40

w1 _-3.00

w2 _-3.00

0.20

xample:
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

 $\int \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \int \frac{1}{1 + e^{-x}} \int$

tional graph tation may not be hoose one where lients at each be easily 1!

0.73

1.00

dient] = 0.2

Sigmoid loca gradient:

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

Stanford CS231n 10th Anniversary

Lecture 4 - 99

Another examp

w0 2.00

x0 -1.00

w1 -3.00

x1 -2.00

w2 -3.00

0.20

0.40

-0.20

ble:
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

Sigmoid function $\sigma(x) = \frac{1}{1 + e^{-x}}$
Computational graph representation may not unique. Choose one we local gradients at each node can be easily expressed!
 $(* - \frac{2.00}{0.20} + \frac{4.00}{0.20} + \frac{1.00}{0.20} + \frac{1.00}{0$

ntation may not be Choose one where adients at each in be easily ed!

0.73

.00

Sigmoid local gradient:

$$rac{d\sigma(x)}{dx} = rac{e^{-x}}{\left(1+e^{-x}
ight)^2} = \left(rac{1+e^{-x}-1}{1+e^{-x}}
ight) \left(rac{1}{1+e^{-x}}
ight) = \left(1-\sigma(x)
ight) \sigma(x)$$

Stanford CS231n 10th Anniversary

Lecture 4 - 100

add gate: gradient distributor

Stanford CS231n 10th Anniversary

Lecture 4 - 101

add gate: gradient distributor

mul gate: "swap multiplier"

Stanford CS231n 10th Anniversary

Lecture 4 - 102

add gate: gradient distributor

mul gate: "swap multiplier"

copy gate: gradient adder

Stanford CS231n 10th Anniversary

Lecture 4 - <u>103</u>

add gate: gradient distributor

copy gate: gradient adder

mul gate: "swap multiplier"

max gate: gradient router

April 10, 2025

Stanford CS231n 10th Anniversary

Lecture 4 - 104

Forward pass: Compute output

Backward pass:

Compute grads

def f(w0, x0, w1, x1, w2): s0 = w0 * x0 s1 = w1 * x1 s2 = s0 + s1 s3 = s2 + w2 L = sigmoid(s3)

grad_L = 1.0
$grad_s3 = grad_L * (1 - L) * L$
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_s0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0

Stanford CS231n 10th Anniversary

Lecture 4 - 105

(def f(w0,	×0,	w1,	x1,	w2):
	s0 = w0	* X	0		
Forward pass:	s1 = w1	* X	1		
Compute output	s2 = s0	+ s	1		
Compute output	s3 = s2	+ w.	2		
	L = sign	noid	(s3)		

Base case	grad_L = 1.0
	$grad_s3 = grad_L * (1 - L) * L$
	grad_w2 = grad_s3
	grad_s2 = grad_s3
	grad_s0 = grad_s2
	grad_s1 = grad_s2
	grad_w1 = grad_s1 * x1
	grad_x1 = grad_s1 * w1
	grad_w0 = grad_s0 * x0
	grad x0 = grad s0 * w0

Lecture 4 - 106

April 10, 2025

Forward pass: Compute output

Sigmoid

d	ef	f(v	v0,	х	Э,	w1,	x1,	w2):
	s) =	w0	*	х	0		
	s1	. =	w1	*	X	1		
	sź	2 =	s0	+	S	1		
	s3	3 =	s2	+	W.	2		
	L	= 9	sigr	no:	id	(s3)		

grad_L = 1.0	
$grad_s3 = grad_L * (1 - L) * L$	
grad_w2 = grad_s3	
grad_s2 = grad_s3	
grad_s0 = grad_s2	
grad_s1 = grad_s2	
grad_w1 = grad_s1 * x1	
grad_x1 = grad_s1 * w1	
grad_w0 = grad_s0 * x0	
grad x0 = grad s0 * w0	

Stanford CS231n 10th Anniversary

Lecture 4 - 107

Forward pass: Compute output

Add gate

d	ef	f(w0,	x	0,	w1,	x1,
	s) =	w0	*	x	0	
	s:	L =	w1	*	X.	1	
	sź	2 =	s0	+	s	1	
	s	3 =	s2	+	w,	2	
	L	=	sig	mo:	id	(s3)	

grad_L = 1.0
grad_s3 = grad_L * (1 - L) * L
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_s0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0

w2):

Lecture 4 - 108

April 10, 2025

Forward pass:	
Compute output	

Add gate

te	ef	1	f(v	v0,	х	Э,	w1,	x1,
	s	0	=	w0	*	x	0	
	s:	L	=	w1	*	X.	1	
	sź	2	=	s0	+	s	1	
	s	3	=	s2	+	W,	2	
L	L	-	= 5	sigr	no:	id	(s3)	

grad_L = 1.0
grad_s3 = grad_L * (1 - L) * L
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_s0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0

w2):

Lecture 4 - 109

April 10, 2025

	<pre>def f(w0,</pre>	×0,	w1,	x1,	w2):
	s0 = w0	* x	0		
Forward pass:	s1 = w1	* X	1		
Compute output	s2 = s0	+ s	1		
compute output	s3 = s2	+ w	2		
	L = sig	moid	(s3)		

grad_L = 1.0
grad_s3 = grad_L * (1 - L) * L
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_s0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0

Lecture 4 - 110

Multiply gate

April 10, 2025
Backprop Implementation: "Flat" code



Forward pass: Compute output

lef	f (w0,	x0,	w1,	x1,	w2):
s٥) = w0	* X	0		
s1	L = w1	* X	1		
s2	2 = s0	+ s	1		
s3	8 = s2	+ w	2		
L	= sig	noid	(s3)		

	grad_L = 1.0
	$grad_s3 = grad_L * (1 - L) * L$
	grad_w2 = grad_s3
	grad_s2 = grad_s3
	grad_s0 = grad_s2
	grad_s1 = grad_s2
	grad_w1 = grad_s1 * x1
	grad_x1 = grad_s1 * w1
M. D. L.	grad_w0 = grad_s0 * x0
Multiply gate	grad_x0 = grad_s0 * w0

Stanford CS231n 10th Anniversary

Lecture 4 - 111

Modularized implementation: forward / backward API

Gate / Node / Function object: Actual PyTorch code



(x,y,z are scalars)

<pre>class Multiply(torch.autograd.Function):</pre>		
@staticmethod		
<pre>def forward(ctx, x, y):</pre>	Need to cache some	
ctx.save_for_backward(x, y) 🛶	values for use in	
z = x * y	backward	
return z		
@staticmethod		
<pre>def backward(ctx, grad_z):</pre>	_ Upstream	
<pre>x, y = ctx.saved_tensors</pre>	gradient	
<pre>grad_x = y * grad_z # dz/dx * dL/dz</pre>	Multiply upstream	
<pre>grad_y = x * grad_z # dz/dy * dL/dz</pre>	and local gradients	
<pre>return grad_x, grad_y</pre>		

Stanford CS231n 10th Anniversary

Lecture 4 - 115

Example: PyTorch operators

C pytorch / pytorch			O NO	8+ 1221	# Unatar	28,770	Ytark	8,340
Code () Insure 2,298 ()	Pull requests .581	Presente A	il wie ili	mights				
Tree 810c/s8881 + pytorch / aten	/ src / THNN / gen	veric /		Darres	un line light	and then	First The	Hotory
De experg ent facebook-github-bot (amenication at includes.	in Pytoren, (Philam)	100		Latest co	merit \$17	chell um De	1 8, 2210
E AbsCriteriunic	Canonicalize all a	rcludes in PyTarch.	(#14840)				4.00	rifts age
E OCECHANION 6	Canonicalize all r	Cluber in PyTarch.	(#14849)				4.000	ritte age
ClaveNLLOrberton.c	Catoricalize al la	ncludes in PyTorch.	0144400				4 700	rdha agei
E Coltina	Campricalize all a	ncludes in PyTarsh.	(#14840)				4.00	rafte age
E BLUE	Canoricalize at in	nckudes in PyTerch,	19140-00				4.000	reha ago
E featurel.Photoge	Catorical or all a	ncludes in Pytorch.	(#54843)				4.000	rithà age
E-GalestLinearLink.c	Canonicalize all #	ncludes in PyTarch.	004640				4 110	ntha age
E HardTarin.c	Canonicalize all it	clubes in PyToron.	(#14848)				4 110	rethi aga
E inddata	Catoricality at it	scludes in PyTorin.	07140-010				1.00	nthi egi
E ndeiLinear z	Canonicalize all a	ncludes in PyTorch.	(#14848)				4 100	rthe aga
ELANYHOUS	Canonicalize all in	Clubes in PyTorch.	(#14849)				4.00	erte aga
LogSigmoid c	Canonicalize all in	nckudes in PyTorsh.	(#14840)				4.000	rithi age
III MILEORINALE	Carlonicalize at in	icludes in Pytonin.	(#148439)				4 110	rifts age
MuttLabelMarpinCriterion.c	Canonicalize all in	ciudes in PyTorch.	UPTABARO.				4.000	nthe age
MatthergeCriterion.c	Catorication at a	ncludes in PyTarch.	(#54849)				4.000	withis sugar
BRALUE	Canonicalize all 2	schieles in PyTarch.	07562490				4 110	nthe age
E Signaid c	Canonicalize all a	sclubes in PyToron.	(#14848)				4.000	rethi ago
E SmoothLitOrbenon.e	Canoricalize all in	schudes in PyTorch.	UF140400				1.04	rets age
E SoftMarginCriterion &	Canonicalize all a	ncludes in PyTarch.	(#14840)				4.00	who age
E Setative a	Canonicalize all in	Cudes in Pyteron.	(#14849)				4.00	net a sign
E betsfyrik c	Catoricalize all is	ncludes in PyTorsh.	(#14840)				4.000	rithi age
E Spenet new a	Carlomostor at it	icludes in Pytonch.	(#148430				4 110	rifts age
Spintal Adaptive Average Pooling c	Canoricalize all a	cludes in PyTorch.	04548400				4.00	rithe age
E Spielal-daptiveMaxPooling c	Canonication all in	ncludes in PyTarch.	14148-031				4.00	riths age
R SpetalAveragePooling c	Canonicalize all 2	schuldes in PyTanch.	0046474				4 100	-the age

E IpeneClassific Crimenie	Canonicalize all inclusies in PyTonth, (#18840)	A moreta ago
E taxteConvelutionMile	Canonicalitie all includes in Pythech, (#18845)	4 menthe age
E SpenaDiate/Convolution 2	Canonicalitie all Includes In PyTorch, (#1484/8)	A munths ago
R MontellitatechilarPooling &	Canonicalda ali inclutes in Pyllorch, (#14840)	é mentre ago
E SpetialPractionalMasPooling.t	Canonicalize at includes in PyTorch. (#148489)	4 months ago
E Apeliel Utilized Convertision 2	Canonicalize al Includes IV PyTorch, (#163-89)	à minthe ago
E Spetial/taxCroosing #	Caronicalize all includes in Pyflorch, (#18585)	é monthe age
E taxtulleriscrion/soding a	Ceronicalize at inclusive in Pytholiti, (in148.03)	A months ago
E SpelaRepisalar/witing-t	Canonicalize al inclusion in PyTin(H, (#14520)	A mantha age
E terminplaneingbiltreat z	Canonicalize al includes in Pythesh, (2148-45)	# months age
E Spetial /ptamping/searest.c	Canonicalize all includes in PyTorch. (#14842)	4 mentra ago
E THREE A	Canonicalize all inclusies in PyTorch, (#14840)	4 manths ago
El Tanín a	Canonicalize all inclutes in PyTorch. (#16845)	4 months age
InterporalitefectionPailting c	CanonicalDe all includes in Pytlonch, (#14846)	à mantha ago
i) TemporalReplicationPadding c	Canonicalize all includes in Pyflorich, (#148445)	4 mainthé ago
E TemporaReviconvelution.c	Caronicalize al shouldes in Pyforch. (#16840)	4 months ago
E temporal.planping.new.c	Canonicalde all includes in Pythisti, (#18581)	il monthe ago
E Temporaluptamptinghearest.r.	Canonicalize all includes in Pytositic (#18542)	dimontha ago
E Volumetric Adaptive Average Popular.	Canonicalize at inclusies in PyTonth, (#18843)	Amortha ago
Witanetic AdaptiveMaxPosting a	Canonicalize al includes in PyTerch, (#18843)	A months age
E Wilanastic Average Pooling c	Canonicalitie all Includes in PyTerch, (#14840)	A munths ago
C VolumetricConvolutionAMI c	Canonicalda all inclutes in PyTorch, (#14840)	4 menths ego
WhenerticDristedConvolution c	Canonicalize at includes in PyTorch, (#14849)	4 months ago
E WatametricDKatedMaxPooling.c	Canonicalize al inclutes in PyTorch, (#14840)	à martha ago
R volumetric/sactional/Vax/boing.c	Canonicalize all includes in Pyforch, (#18849)	é montha ago
E WarnetVcFulDiatedConvolution.c	Carionicaliza al inclusies in Pythech, (#188.03)	A months ago
E Volumetric ManUnpooling.a	Canomitalize all inclusive in Pytlerch, (#14545)	A manths age
E VounemicReplication/helding.c	Canonicalize all includes in Pyterch, (#148-0)	4 months age
University SamplingNeweol d	Canonicative at includes in Pytlerch, (#14840)	4 months ego
C WARNETIC Uplanning Vilnes: c	Carorisalize al inclusive in PyTorch, (#14849)	4 months age
E these ussempting h	Implement in Auctional Interpolate based on upsample. (#000)	0 months app
E pooling, shape h	Use Integer meth to compute subput size of pooling operations (#14405)	A months ago
E untrat a	Canonicalize all includes in PyTecch, (#165-65)	4 minthé ago

Stanford CS231n 10th Anniversary

Lecture 4 - 116



Lecture 4 - 117



Lecture 4 - 118



Lecture 4 - 119

So far: backprop with scalars

What about vector-valued functions?

Lecture 4 -

120

April 10, 2025

Recap: Vector derivatives

Scalar to Scalar

 $x\in \mathbb{R}, y\in \mathbb{R}$

Regular derivative:

 $\frac{\partial y}{\partial x} \in \mathbb{R}$

If *x* changes by a small amount, how much will *y* change?

Stanford CS231n 10th Anniversary

Lecture 4 - 121

Recap: Vector derivatives

Scalar to Scalar

Vector to Scalar

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

 $\frac{\partial y}{\partial x} \in \mathbb{R}$

If *x* changes by a small amount, how much will *y* change?

Derivative is Gradient:

 $x \in \mathbb{R}^N, y \in \mathbb{R}$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

For each element of *x*, if it changes by a small amount then how much will *y* change?

Stanford CS231n 10th Anniversary

Lecture 4 - 122

Recap: Vector derivatives

Scalar to Scalar

 $x \in \mathbb{R}, y \in \mathbb{R}$

Regular derivative:

 $\frac{\partial y}{\partial x} \in \mathbb{R}$

If *x* changes by a small amount, how much will *y* change?

Vector to Scalar

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is Gradient:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

Vector to Vector $x \in \mathbb{R}^N, y \in \mathbb{R}^M$

Derivative is Jacobian:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \left(\frac{\partial y}{\partial x}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

For each element of *x*, if it changes by a small amount then how much will *y* change? For each element of *x*, if it changes by a small amount then how much will each element of *y* change?

April 10, 2025

Stanford CS231n 10th Anniversary

Lecture 4 - 123



Stanford CS231n 10th Anniversary

Lecture 4 - 124



Stanford CS231n 10th Anniversary

Lecture 4 - 125



Stanford CS231n 10th Anniversary

Lecture 4 - 126



much does it influence L?

Stanford CS231n 10th Anniversary

Lecture 4 - 127



Stanford CS231n 10th Anniversary

Lecture 4 - 128



Stanford CS231n 10th Anniversary

Lecture 4 - 129



Stanford CS231n 10th Anniversary

Lecture 4 - 130

Gradients of variables wrt loss have same dims as the original variable



Stanford CS231n 10th Anniversary

Lecture 4 - 131



Stanford CS231n 10th Anniversary

Lecture 4 - 132





Stanford CS231n 10th Anniversary

Lecture 4 - 133



Stanford CS231n 10th Anniversary

Lecture 4 - 134



Upstream gradient

Stanford CS231n 10th Anniversary

Lecture 4 - 135



Stanford CS231n 10th Anniversary

Lecture 4 - 136



Stanford CS231n 10th Anniversary

Lecture 4 - 137

4D input x: 4D output z: f(x) = max(0,x)Jacobian is sparse: 3 | 3 (elementwise) off-diagonal entries [-1] always zero! Never explicitly form Jacobian -- instead 4D dL/dx: $\left[\frac{dz}{dx}\right]\left[\frac{dL}{dz}\right]$ 4D dL/dz: use implicit $\begin{bmatrix} 4 \end{bmatrix} \leftarrow & \leftarrow & \begin{bmatrix} 4 \end{bmatrix} \leftarrow & \\ \begin{bmatrix} 0 \end{bmatrix} \leftarrow & \begin{pmatrix} \frac{\partial L}{\partial x} \end{pmatrix}_i = \begin{cases} \left(\frac{\partial L}{\partial z}\right)_i & \text{if } x_i > 0 & \leftarrow & \begin{bmatrix} -1 \end{bmatrix} \leftarrow & \text{Upstream} \\ 0 & \text{otherwise} \leftarrow & \begin{bmatrix} 5 \end{bmatrix} \leftarrow & \text{gradient} \end{cases}$ [4] ← multiplication [0] -← [9] ← ____

Stanford CS231n 10th Anniversary

Lecture 4 - 138



Lecture 4 - 139



Lecture 4 - 140



Lecture 4 - 141



Lecture 4 - 142



Also see derivation in the course notes:

http://cs231n.stanford.edu/handouts/linear-backprop.pdf

Stanford CS231n 10th Anniversary

Lecture 4 - 143

x: [N×D] [2 1 -3]
[-3 4 2]
w: [D×M]
[3 2 1 -1]
[2 1 3 2]
[3 2 1 -2]



y: [N×M]

[13 9 -2 -6]

[52171]

dL/dy: [N×M]

[23-39]

[-8 1 4 6]

April 10, 2025

Jacobians: dy/dx: [(N×D)×(N×M)] dy/dw: [(D×M)×(N×M)]

For a neural net we may have N=64, D=M=4096 Each Jacobian takes ~256 GB of memory! Must work with them implicitly!

Lecture 4 -

144

[13 9 -2 -6] x: [N×D] Matrix Multiply [52171] [2]1-3] $y_{n,m} = \sum x_{n,d} w_{d,m}$ [-3 4 2] dL/dy: [N×M] w: [D×M] [23-39] [-8 1 4 6]Q: What parts of y are $\begin{bmatrix} 3 & 2 & 1 & -1 \end{bmatrix}$ [2132]affected by one [3 2 1 - 2] element of *x*?

Stanford CS231n 10th Anniversary

Lecture 4 - 145

April 10, 2025

y: [N×M]

x: [N×D]
[2 1 -3]
[-3 4 2]
w: [D×M]
[3 2 1 -1]
[2 1 3 2]
[3 2 1 -2]

Matrix Multiply $y_{n,m} = \sum x_{n,d} w_{d,m}$ Q: What parts of y are affected by one element of x? A: $x_{n,d}$ affects the whole row $y_{n,\cdot}$ $\frac{\partial L}{\partial x_{n,d}} = \sum_{m} \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}}$

Lecture 4 -

146

 $v \cdot [N \times M]$

April 10, 2025

x: [N×D]
[2 1 -3]
[-3 4 2]
w: [D×M]
[3 2 1 -1]
[2 1 3 2]
[3 2 1 -2]

Matrix Multiply $y_{n,m} = \sum x_{n,d} w_{d,m}$ Q: What parts of y are affected by one element of x? A: $x_{n,d}$ affects the whole row $y_{n,\cdot}$ $\frac{\partial L}{\partial x_{n,d}} = \sum_{m} \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}}$

Q: How much does $x_{n,d}$ affect $y_{n,m}$?

147

Lecture 4 -

v: [N×M]

dL/dy: [N×M]

2 3 - 3 9

[-8 1 4 6]

April 10, 2025

x: [N×D] [213] [-342] w: [D×M] [321-1] [2132] a [321-2] e

v: |N×M| Matrix Multiply $y_{n,m} = \sum x_{n,d} w_{d,m}$ dL/dy: [N×M] 2 3 - 3 9 [-8 1 4 6] Q: What parts of y are affected by one Q: How much does $x_{n,d}$ element of x? affect $y_{n,m}$? mul gate: "swap multiplier" A: $x_{n,d}$ affects the A: $w_{d,m}$ 5*3=15 whole row $y_{n,\cdot}$ 2*5=10 $\frac{\partial L}{\partial x_{n,d}} = \sum \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}} = \sum \frac{\partial L}{\partial y_{n,m}} w_{d,m}$

Lecture 4 -

148

April 10, 2025

x: [N×D] __ [2 1 -3] [-3 4 2] w: [D×M] [3 2 1 -1] [2 1 3 2] [3 2 1 -2]

 $[N \times D] [N \times M] [M \times D]$

$$\frac{\partial L}{\partial x} = \left(\frac{\partial L}{\partial y}\right) w^T$$

Lecture 4 -

149

y: [N×M]

April 10, 2025
Stanford CS231n 10th Anniversary

By similar logic:

[N×D] [N×M] [M×D]

x: [N×D]

[-3 4 2]

w: [D×M]

[321-1]

2 1 3 2

[3 2 1 - 2]

1 -3

$$\frac{\partial L}{\partial x} = \left(\frac{\partial L}{\partial y}\right) w^T$$

These formulas are easy to remember: they are the only way to make shapes match up!

April 10, 2025

u

 ∂u

 $[D \times M] [D \times N] [N \times M]$

Backprop with Matrices

Matrix Multiply
$$y_{n,m} = \sum_{d} x_{n,d} w_{d,m}$$

Lecture 4 -

150

 $\mathbf{v} \cdot [\mathbf{N} \mathbf{x} \mathbf{M}]$

Summary for today:

- (Fully-connected) Neural Networks are stacks of linear functions and nonlinear activation functions; they have much more representational power than linear classifiers
- backpropagation = recursive application of the chain rule along a computational graph to compute the gradients of all inputs/parameters/intermediates
- implementations maintain a graph structure, where the nodes implement the forward() / backward() API
- forward: compute result of an operation and save any intermediates needed for gradient computation in memory
- backward: apply the chain rule to compute the gradient of the loss function with respect to the inputs

Lecture 4 - 151

April 10, 2025

Next Time: Convolutional Neural Networks!



Stanford CS231n 10th Anniversary

Lecture 4 - 152

April 10, 2025