

Lecture 2:

How to train your model:

Intro to Models, Gradient Descent, Optimization

Slides credit - CS231N

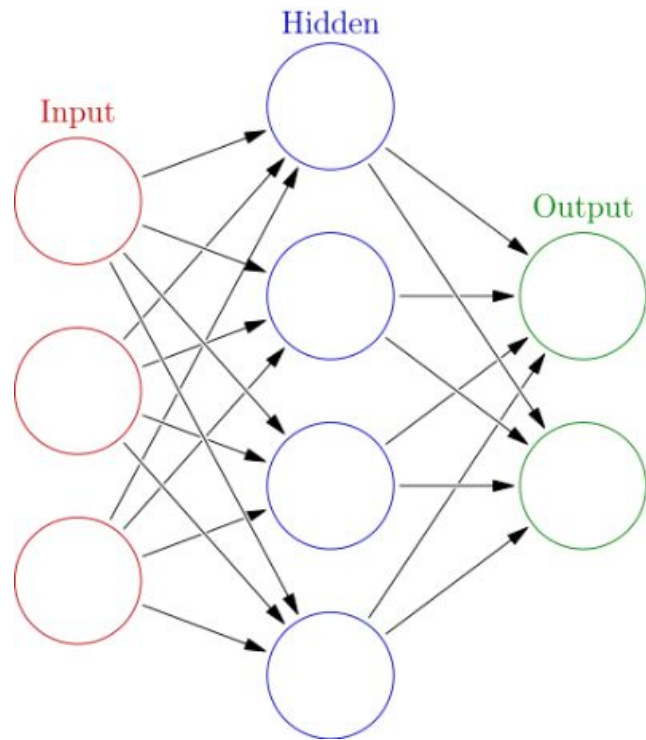
Today's code along notebook



WHAT EVEN IS A MODEL ??



WHAT EVEN IS A MODEL ??



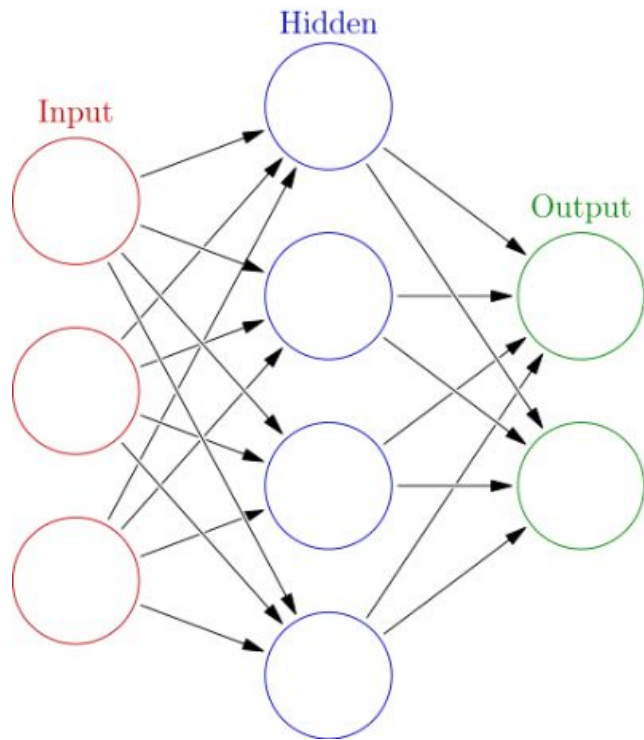
JUST NUMBERS AND
MULTIPLICATION FOR
THE MOST PART

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \\ A_{51} & A_{52} & A_{53} \\ A_{61} & A_{62} & A_{63} \end{bmatrix} \times \begin{bmatrix} B_{11} \\ B_{21} \\ B_{31} \end{bmatrix} = \begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \\ C_{41} \\ C_{51} \\ C_{61} \end{bmatrix}$$

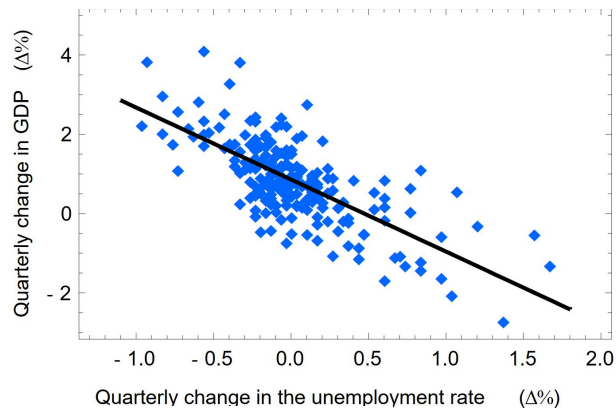
EVERY ML MODEL CAN BE REPRESENTED
AS A MATHEMATICAL FUNCTION MAPPING
INPUTS TO OUTPUTS.

$$f_{\theta}: X \rightarrow Y$$

WHAT EVEN IS A MODEL ??



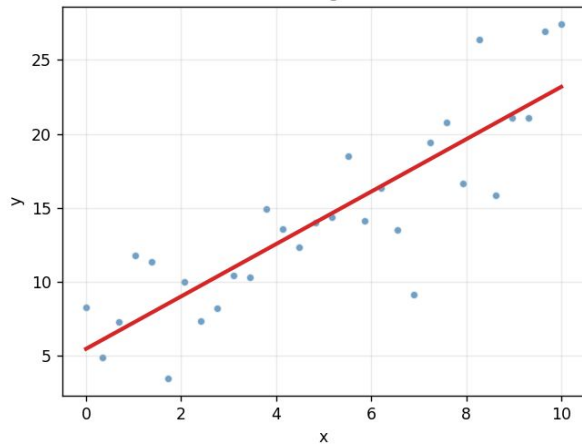
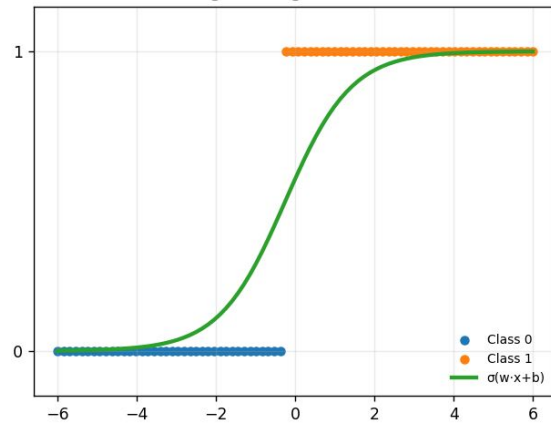
JUST NUMBERS AND
MULTIPLICATION FOR
THE MOST PART



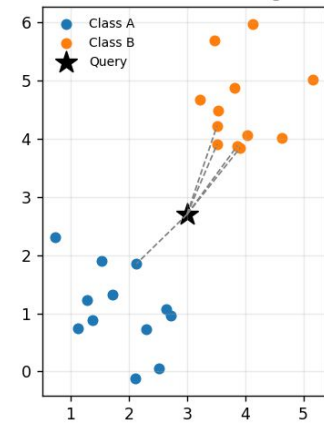
Can be as simple as $Y = a + bx$ And as pretentious as



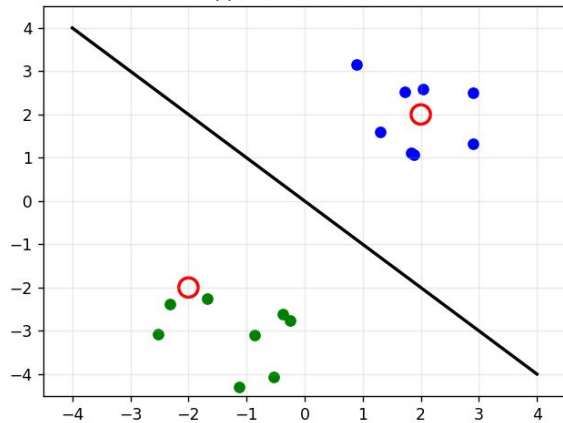
Linear Regression

Logistic Regression (σ)

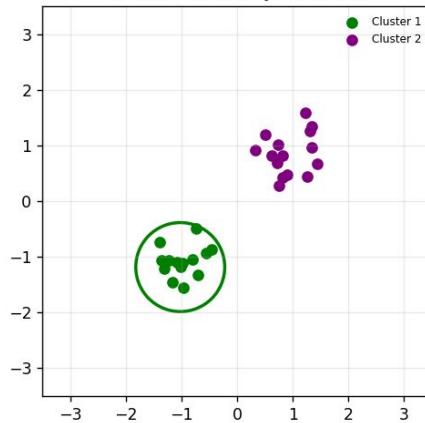
k-NN (k=5) — vote of neighbors



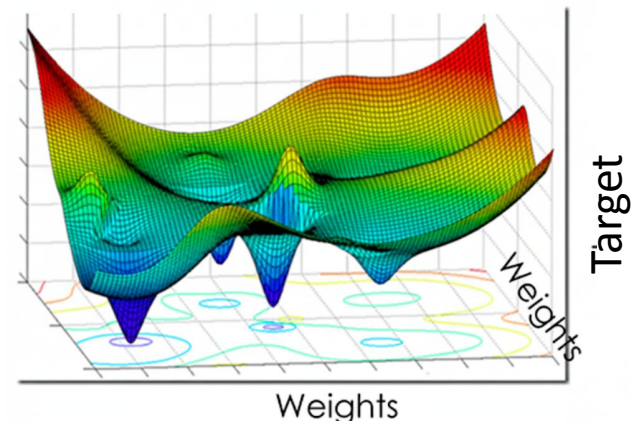
Support Vector Machine



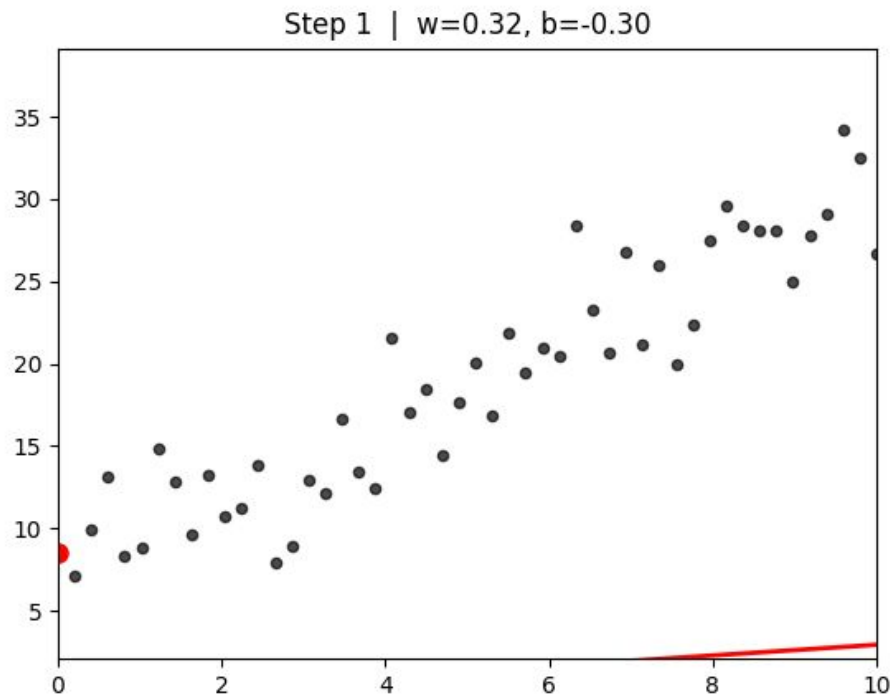
DBSCAN (Density Clusters)



Neural Network



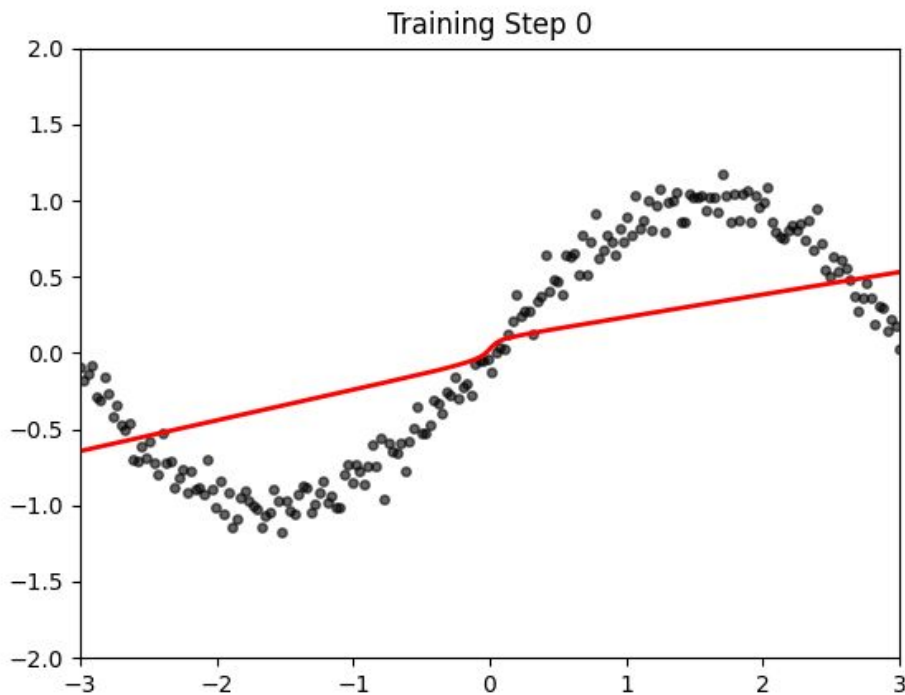
WHAT EVEN IS A MODEL ??



WHAT EVEN IS A MODEL ??

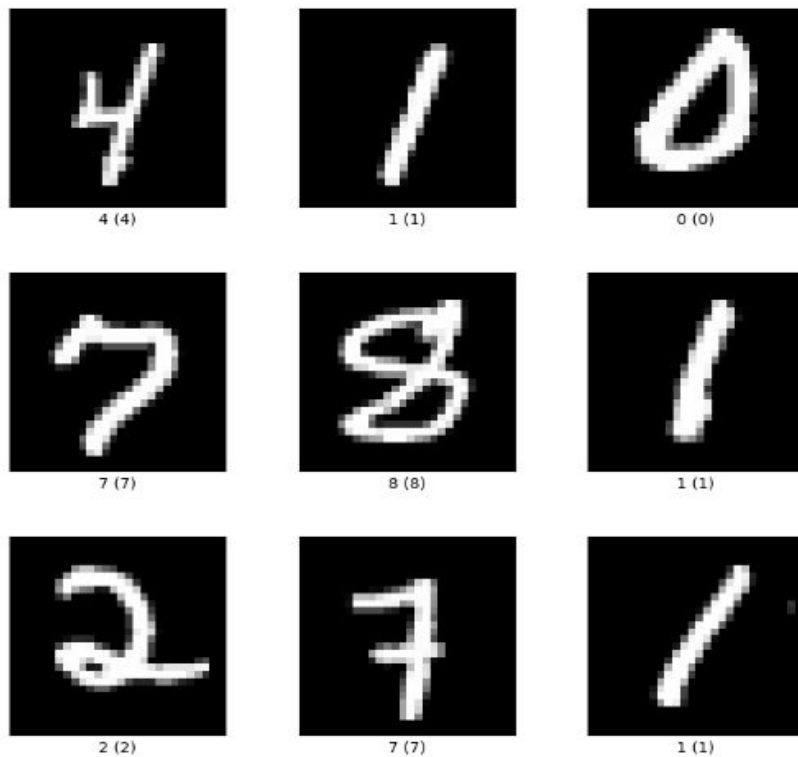
$$y = \sin(x) + \text{noise}$$

Here, The model learns to map this connection by updating its weights and biases



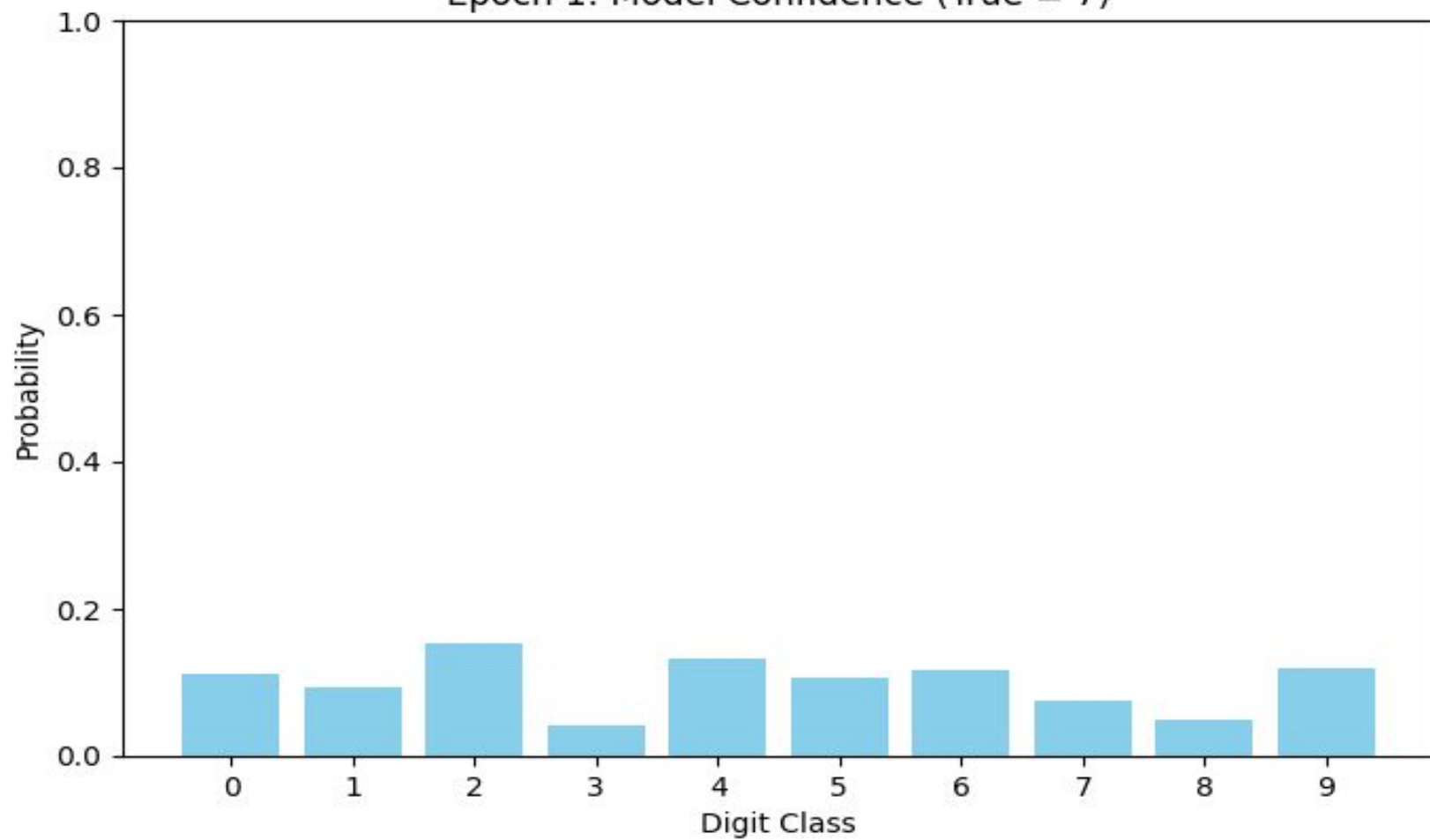
Machine Learning: Data-Driven Approach

1. Collect a dataset of features(images, text) and labels
2. Use Machine Learning algorithms to train a classifier.
Evaluate the classifier on new data(testing data).



MNIST DATASET

Epoch 1: Model Confidence (True = 7)



```
def train(images, labels):  
    # Machine learning!  
    return model
```



Memorize all data
and labels

```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```



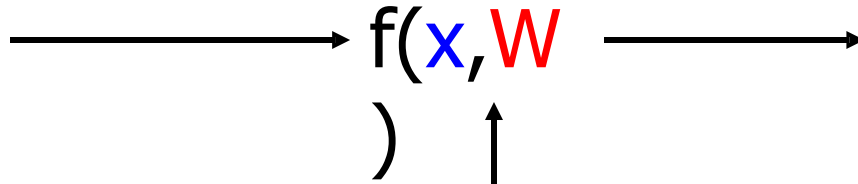
Predict the label of
the most similar
training image

Linear Classifier:

Parametric Approach



Array of 32x32x3
numbers (3072
numbers total)



W
parameter
s or
weights

10 numbers
giving class
scores

Parametric Approach: Linear Classifier



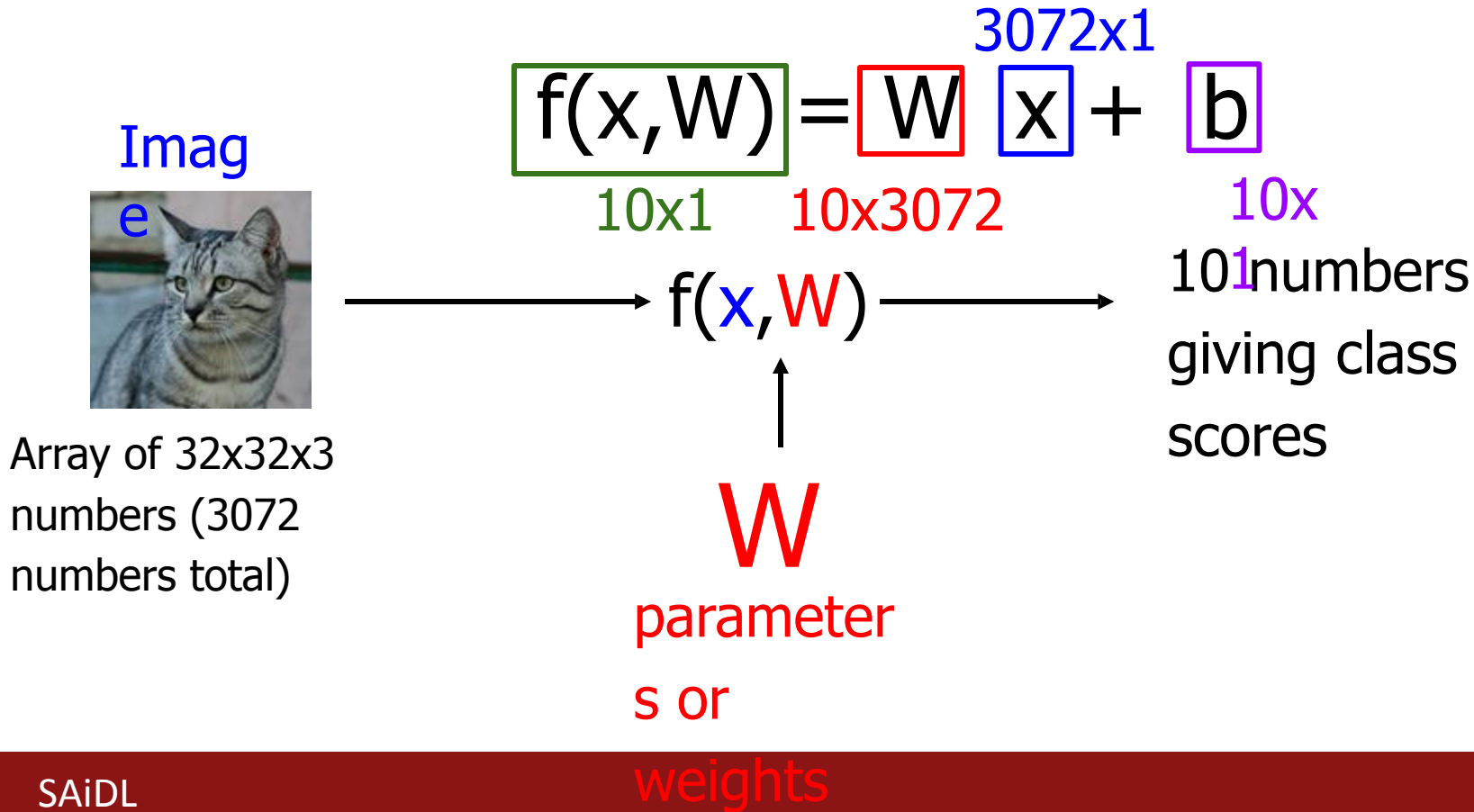
Image

parameters or
weights
3072x1

$$f(x, W) = \boxed{} \boxed{x} + \boxed{b} \text{ } 10 \times 1$$

W

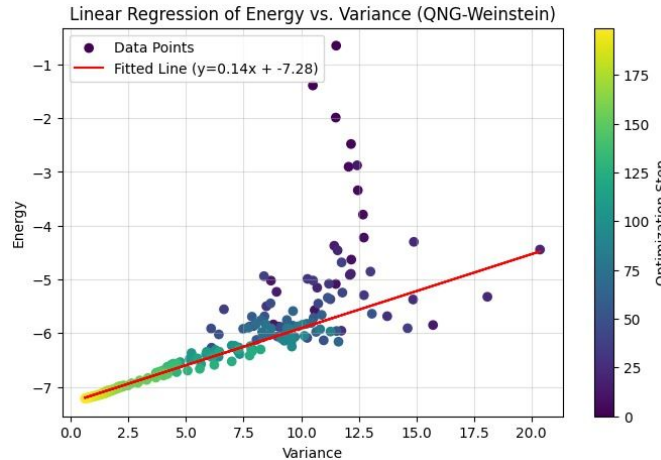
Parametric Approach: Linear Classifier



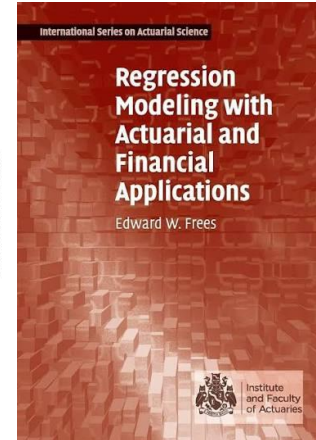
Linear Regression is p chill:

Used in:

- Finance
- Health
- Quantum Computing
- Literally everywhere!



*actual regression model used in
peer reviewed QML literature

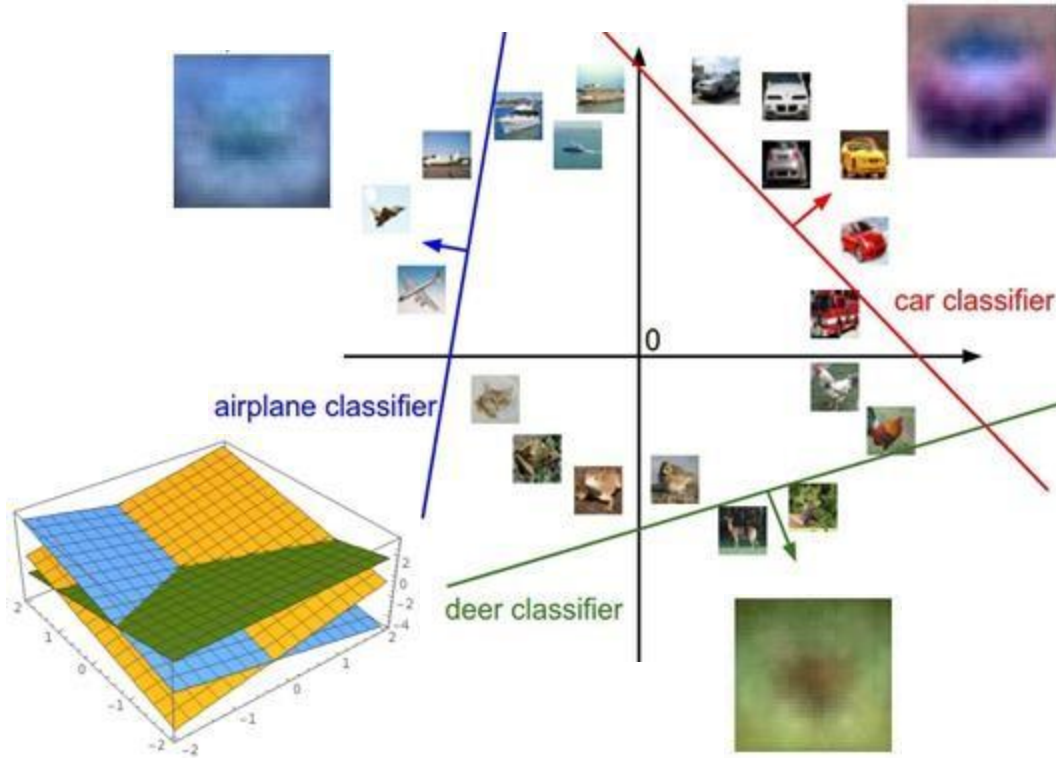


Neural Network

Linear
classifiers



Interpreting a Linear Classifier: Geometric Viewpoint



$$f(x, W) = Wx + b$$



Array of 32x32x3
numbers
(3072 numbers total)

Linear Classifier – Loss function



airplane	-3.45	-0.51	3.42
automobile	-8.87	6.04	4.64
bird	0.09	5.31	2.65
cat	2.9	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	-4.34
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

1. Define a loss function that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. (optimization)

Suppose: 3 training examples, 3
classes. With some W the $f(x, W) = Wx$

:

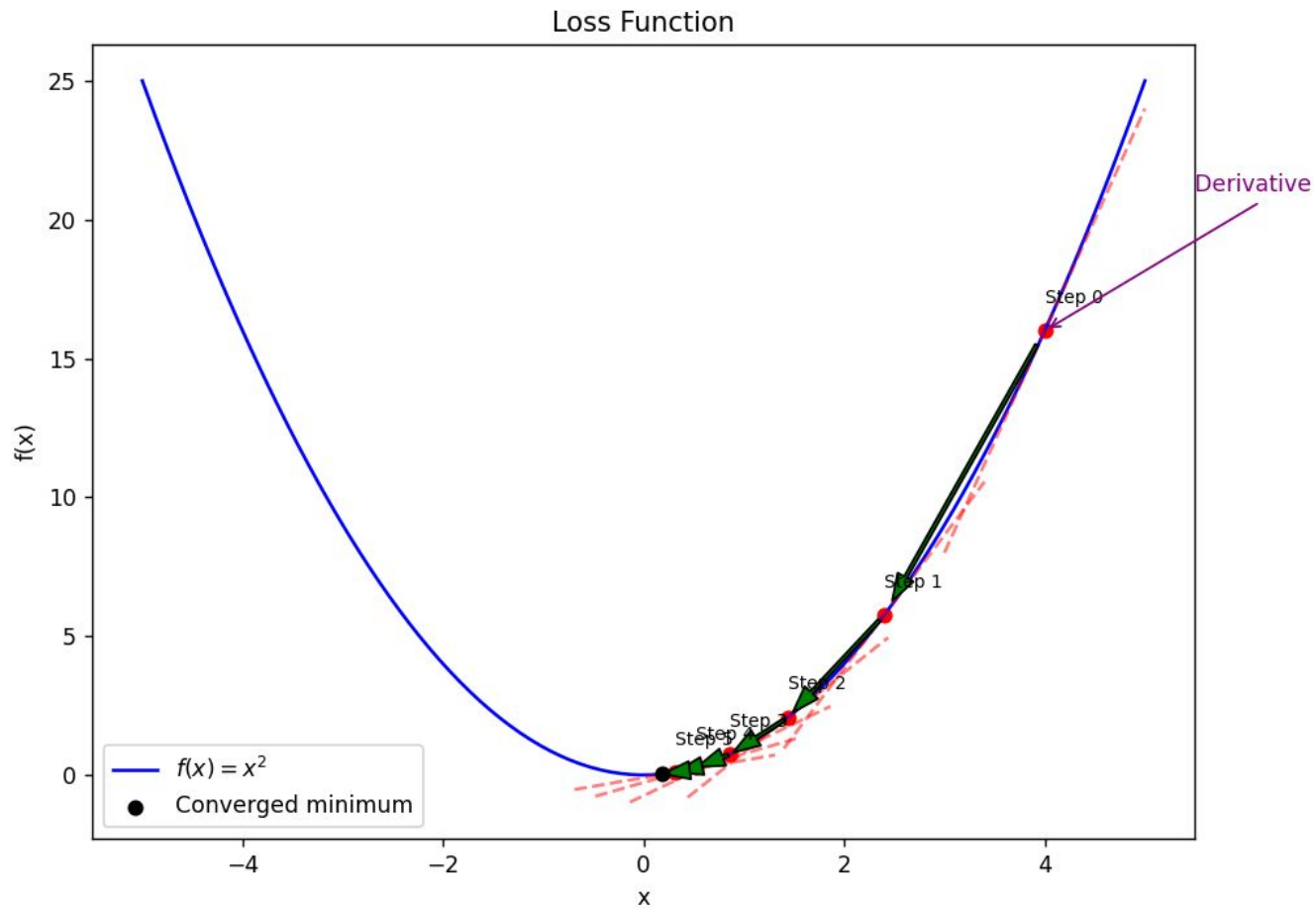


A loss function tells how
good
our current classifier is

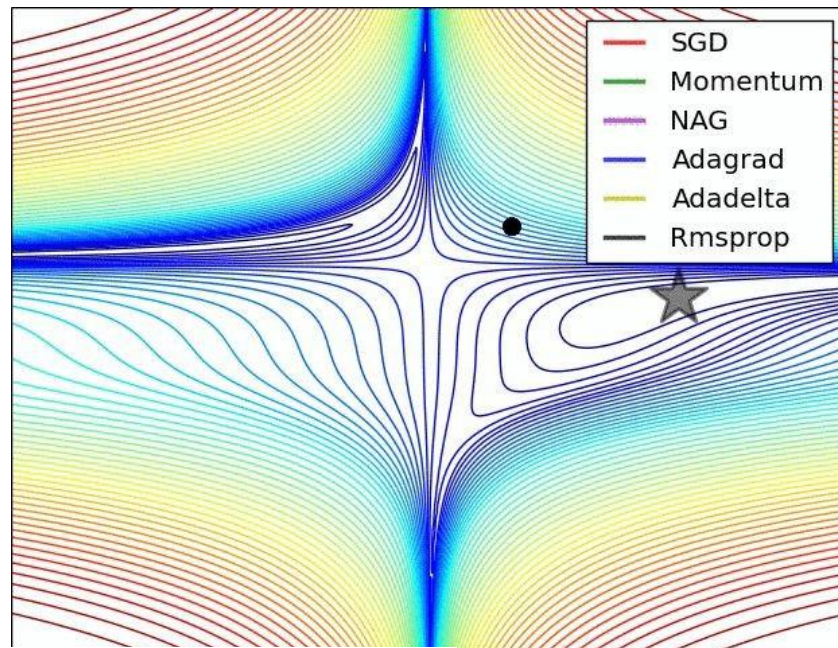
cat	3.2	1.3	2.2
car	5.1	4.9	2.5
fro	-1.7	2.0	-3.1
g			

OPTIMIZATION

OPTIMIZATION:



Optimization



(image credits
to Alec Radford)

```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

Gradient Descent

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

Numerical gradient: slow :(, approximate :(, easy to write :)

Analytic gradient: fast :), exact :), error-prone :(

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

Gradient Descent

Models are trained by adjusting their parameters/weights so that predictions match the data as closely as possible.

Goal of training is to minimize the loss function.

$$f_{\theta}: X \rightarrow Y$$

$$\theta^* = \arg \min_{\theta} L(\theta)$$

$$\theta \leftarrow \theta - n \cdot \nabla_{\theta} L(\theta)$$

Gradient Descent

Models are trained by adjusting their parameters/weights so that predictions match the data as closely as possible.

Goal of training is to minimize the loss function.

$$\theta^* = \operatorname{argmin}_{\theta} L(\theta)$$

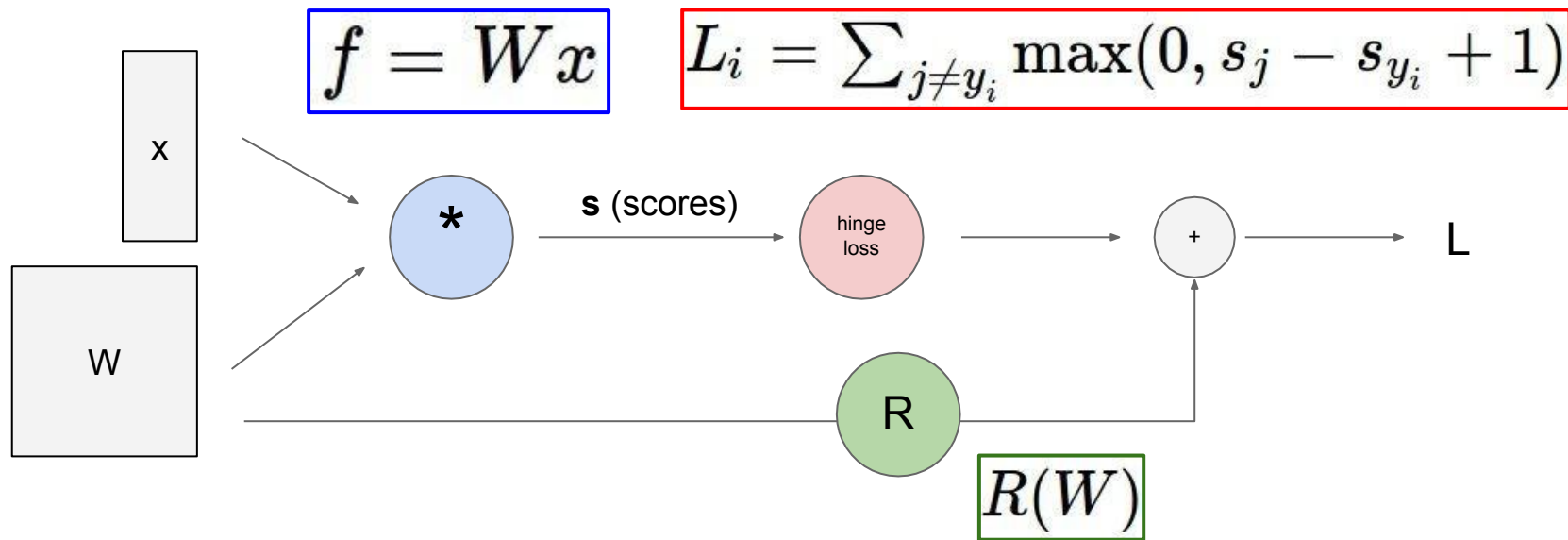
$$\theta \leftarrow \theta - n \nabla_{\theta} L(\theta)$$

n (*Learning rate*)

$$w = w - n \frac{\partial L}{\partial w}$$

$$b = b - n \frac{\partial L}{\partial b}$$

Computational Graph



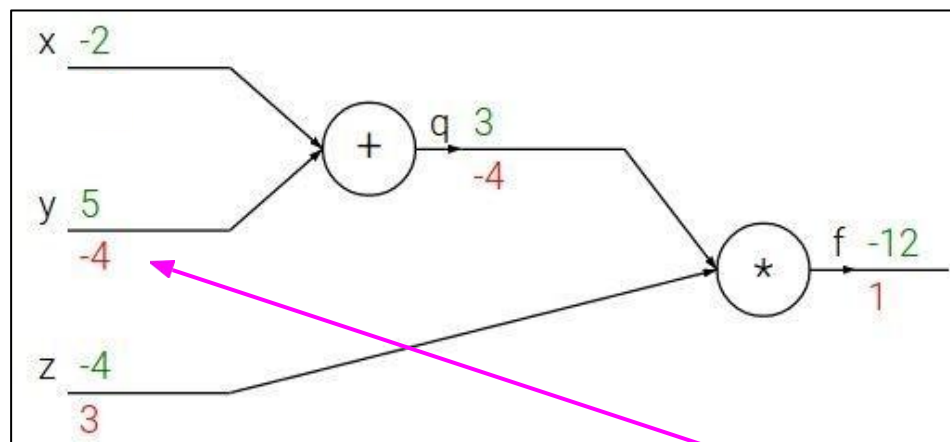
$$f(x, y, z) = (x + y)z$$

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:

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

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

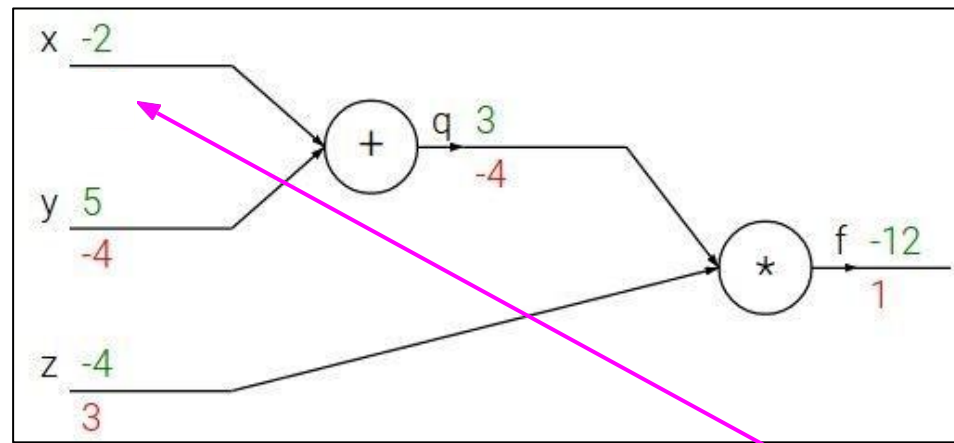
$$f(x, y, z) = (x + y)z$$

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



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

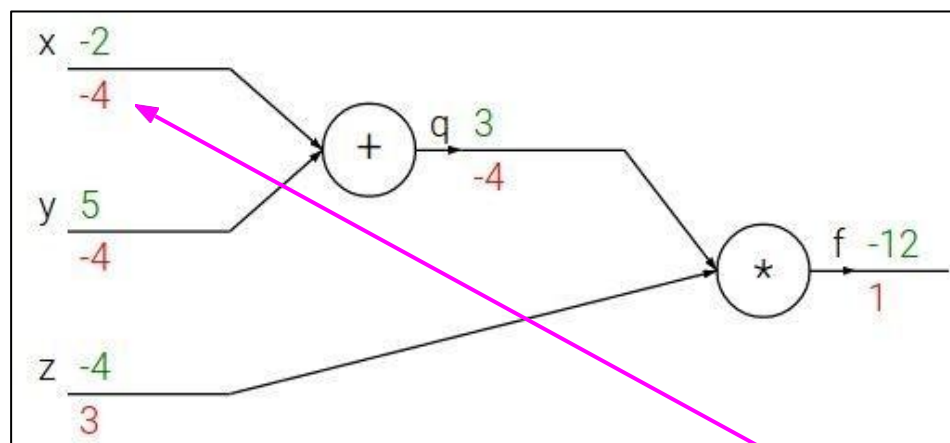
$$f(x, y, z) = (x + y)z$$

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:

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

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

activations

x

$$\frac{\partial L}{\partial x}$$

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

“local gradient”

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

f

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

y

$$\frac{\partial L}{\partial y}$$

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

z

$$\frac{\partial L}{\partial z}$$

gradients

THANK YOU