CS 232: Artificial Intelligence

Fall 2023

Prof. Carolyn Anderson
Wellesley College

Computer Science Colloquium Series | Fall 2023

Supporting Responsible AI Practices in Public Sector Contexts

Anna is a third year PhD student at Carnegie Mellon's Human-Computer Interaction Institute. Her research focuses on improving the design, evaluation, and governance of AI technologies used to inform complex, consequential decisions in real-world organizations. In addition to her research, she will share prior experiences forming collaborations with public sector agencies, doing research internships with industry groups, travelling to conferences, and mentoring undergraduate students. The session will end with an open Q/A discussion on applying to and doing a PhD in Computer Science / Human-Computer Interaction and other topics.

WELLESLEY

Accessibility and Disability Resources:
accessibility@wellesley.edu

Anna Kawakami '21

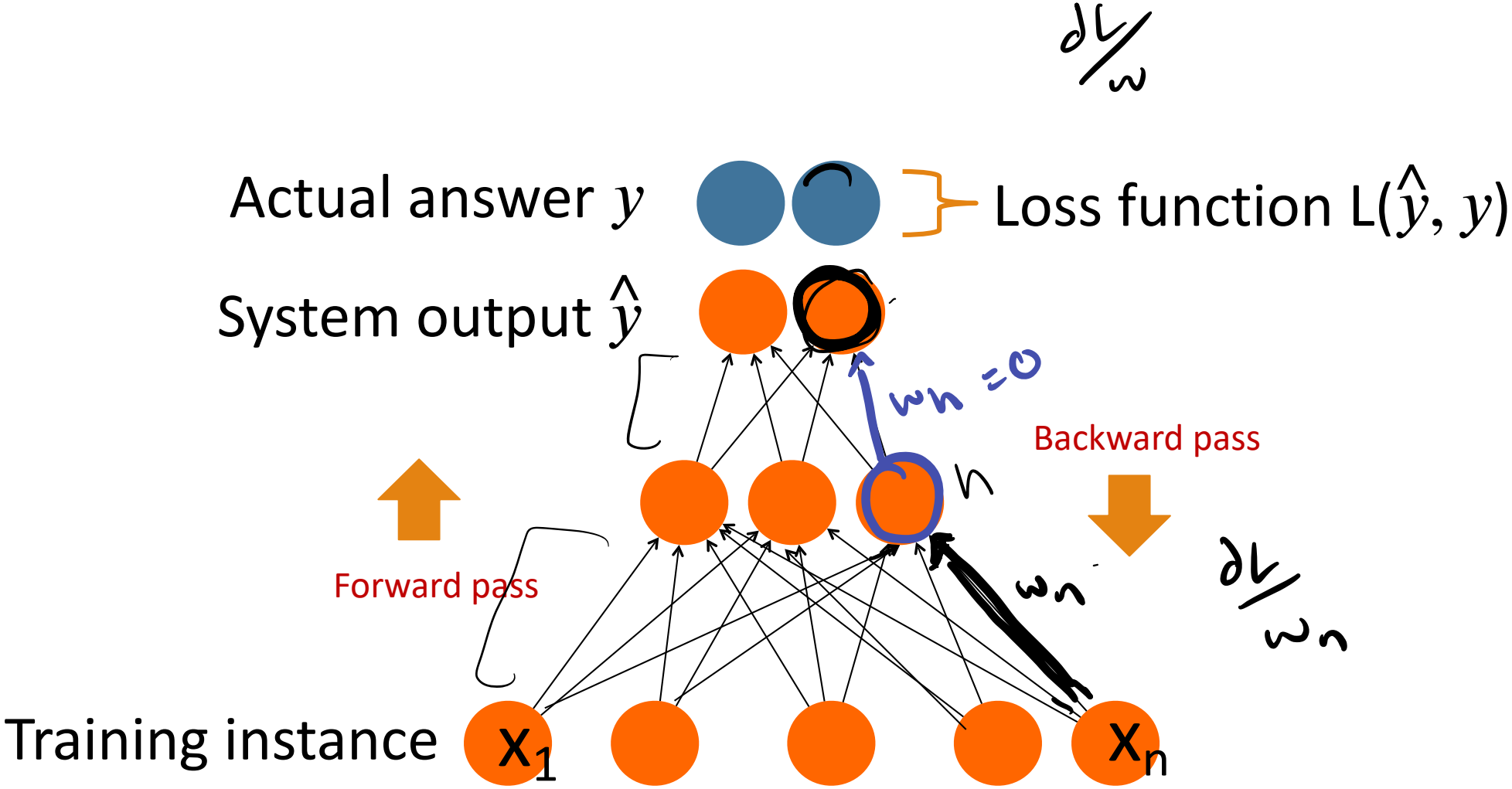
Nov 2nd, 12:45-2:00 | SCI H401
Lunch will be served

Questions??? eni.mustafara

November 2nd

Recap

Intuition: training a 2-layer Network



Intuition: Training a 2-layer network

For every training tuple (x, y)

- Run *forward* computation to find our estimate $\hat{y}$
- Run *backward* computation to update weights:
  - For every output node
    - Compute loss L between true y and the estimated $\hat{y}$
    - For every weight w from hidden layer to the output layer
      - Update the weight
  - For every hidden node
    - Assess how much blame it deserves for the current answer
    - For every weight w from input layer to the hidden layer
      - Update the weight

Loss Function: a measure of how far off the current answer is from the right answer.

For binary logistic regression, we use cross entropy loss:

correct answer

model's guess

correct answer

model's guess

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})]$$

$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1-y) \log(1 - \sigma(w \cdot x + b))]$$

$\hat{y} = \sigma(w \cdot x + b)$ $\quad y \in \{0, 1\}$

Gradient descent for weight updates

$f(x;w) = \hat{y}$

The derivative of the loss function with respect to weights tells us how to adjust the weights to make better predictions.

Derivative of the loss function: $\frac{\partial L(f(x;w),y)}{\partial w}$

We want to move the weights in the opposite direction of the gradient:

$$w^{t+1} = w^t - \eta \frac{\partial}{\partial w} L(f(x;w),y)$$

learning rate

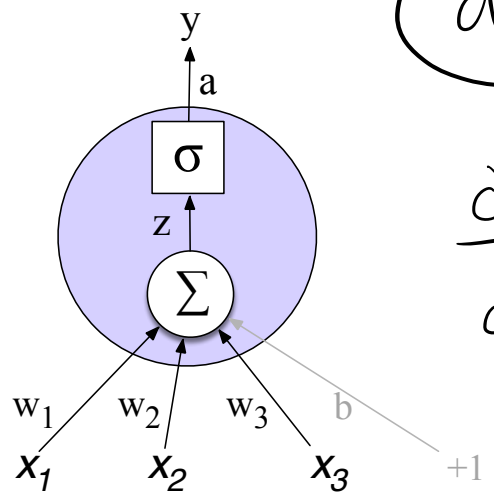
For logistic regression:

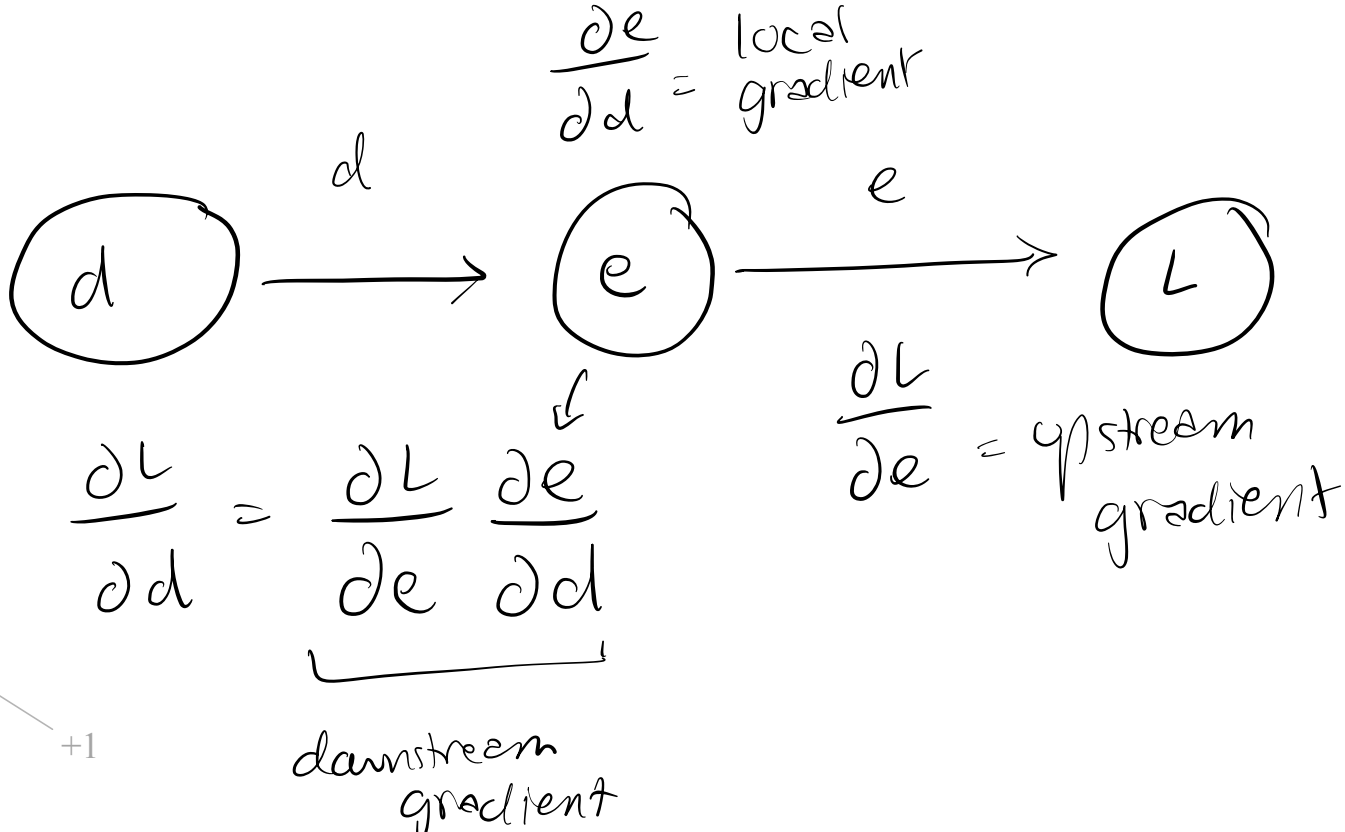
$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = (\hat{y} - y) x_j$$

$$= (\sigma(w \cdot x + b) - y) x_j$$

$$w^{t+1} = w^t - \eta (\sigma(wx+b) - y) x_j$$

Where did that derivative come from?





Each node takes an upstream gradient, multiplies it by the local gradient (the gradient of its output with respect to its input), and uses the chain rule to compute a downstream gradient to be passed on to a prior node.

A node may have multiple local gradients if it has multiple inputs.

Backward Differentiation

For training, we need the derivative of the loss with respect to each weight in every layer of the network.

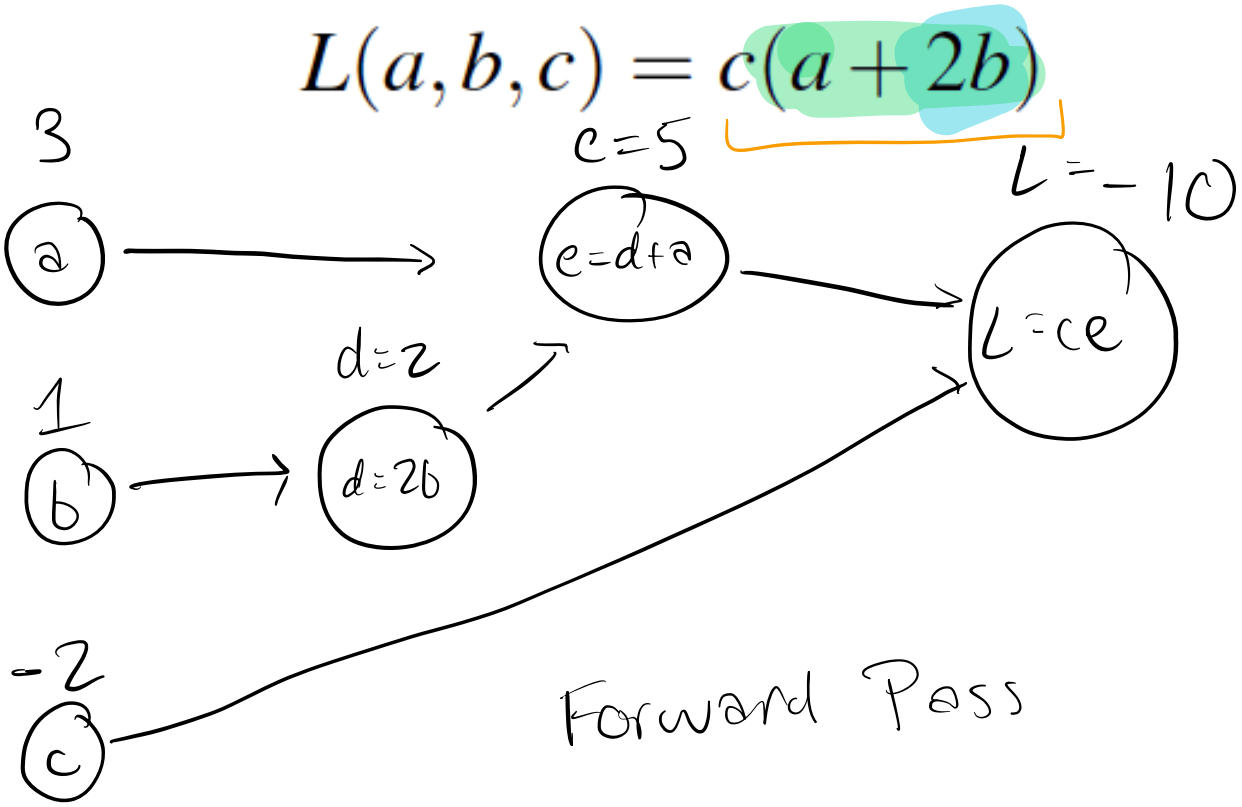
Problem: the derivatives on the prior slide only give the updates for one weight layer: the last one, since loss is computed only at the very end of the network!

Solution: **error backpropagation** (Rumelhart, Hinton, Williams, 1986)

- Backprop is a special case of backward differentiation

Computation Graphs

A computation graph represents the process of computing a mathematical expression



Computations:

$d = 2b$

$e = a + d$

$L = c * e$

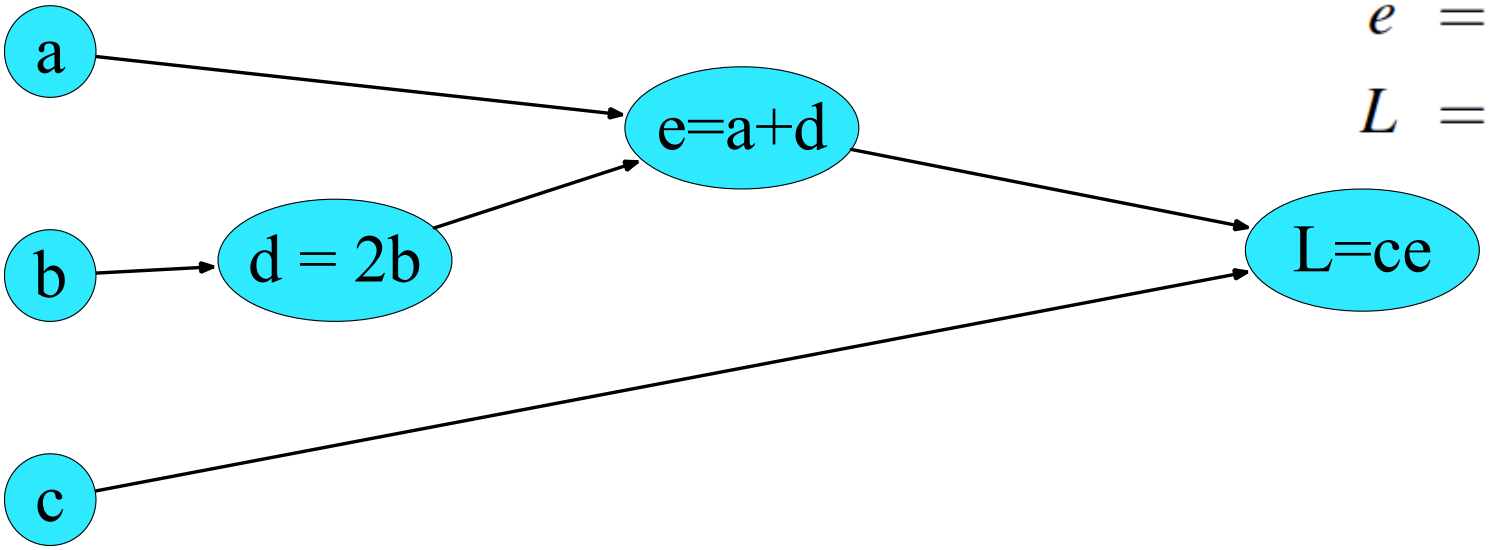
Computation Graphs

A computation graph represents the process of computing a mathematical expression

$$L(a,b,c) = c(a+2b)$$

Computations:

$$d = 2*b$$
$$e = a+d$$
$$L = c*e$$



Backwards differentiation in computation graphs

The importance of the computation graph comes from the backward pass

This is used to compute the derivatives that we'll need for the weight update.

The chain rule

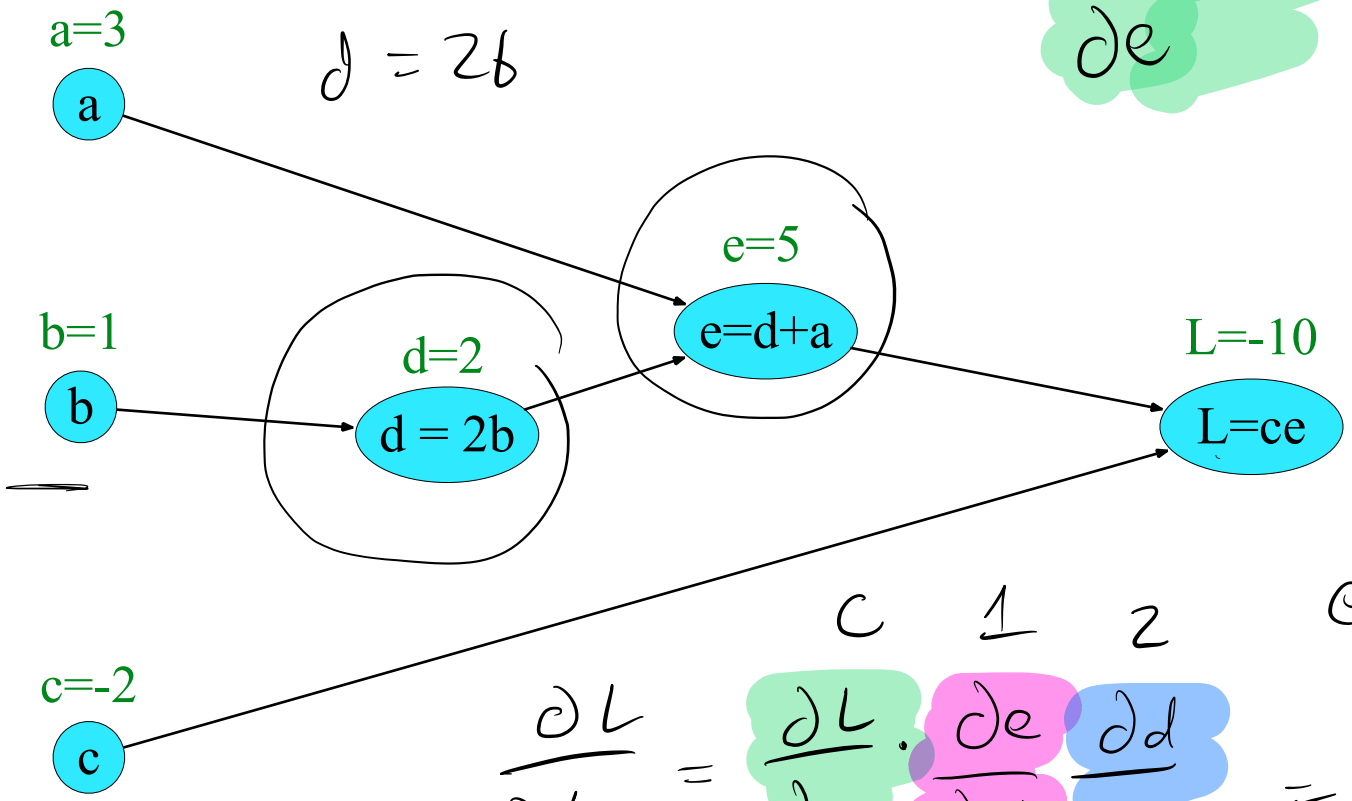
Computing the derivative of a composite function:

$$f(x) = u(v(x)) \qquad \frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

$$f(x) = u(v(w(x))) \qquad \frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dw} \cdot \frac{dw}{dx}$$

Backward Pass

Example



Summary

For training, we need the derivative of the loss with respect to weights in early layers of the network

- But loss is computed only at the very end of the network!

Solution: **backward differentiation**

Given a computation graph and the derivatives of all the functions in it we can automatically compute the derivative of the loss with respect to these early weights.

Generation

AI Tasks

Search

Uninformed Search

Informed Search

Adversarial Games

Navigation

Learning Under Uncertainty

September

Classification

Regression

Sentiment Analysis

Neural Networks

Image Classification

Text Classification

October

Generation

Language Models

Image Generation

Chatbots

Finetuning

Prompt Engineering

November

Language Modeling

Language Modeling

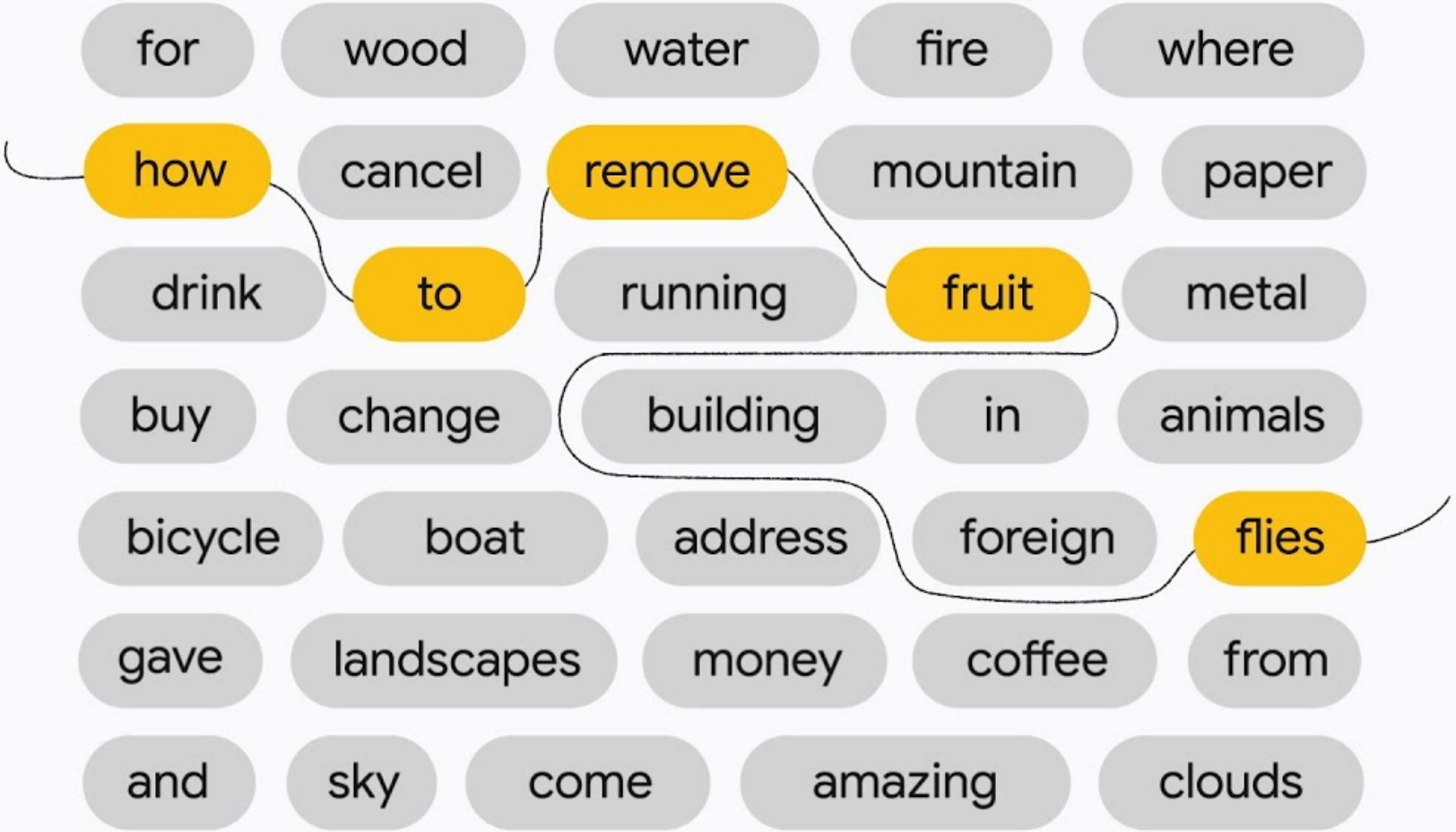
Language modeling is the task of predicting what comes next in a sentence.

Language modeling can also assign probabilities to sequences of text.

In addition, it turns out to be a useful and powerful way to do **representation learning**.

Language models are used in:

- Google search
- Auto-correction / auto-complete
- Speech recognition
- Voice assistants
- Chatbots


for
wood
water
fire
where
how
cancel
remove
mountain
paper
drink
to
running
fruit
metal
buy
change
building
in
animals
bicycle
boat
address
foreign
flies
gave
landscapes
money
coffee
from
and
sky
come
amazing
clouds

Chiang (2023):
ChatGPT is a Blurry JPEG of the Web

[... H]allucinations are anything but surprising; if a compression algorithm is designed to reconstruct text after ninety-nine per cent of the original has been discarded, we should expect that significant portions of what it generates will be entirely fabricated.

If a large language model has compiled a vast number of correlations between economic terms—so many that it can offer plausible responses to a wide variety of questions—should we say that it actually understands economic theory?

Imagine what it would look like if ChatGPT were a lossless algorithm. If that were the case, it would always answer questions by providing a verbatim quote from a relevant Web page. We would probably regard the software as only a slight improvement over a conventional search engine, and be less impressed by it.[...] When we're dealing with sequences of words, lossy compression looks smarter than lossless compression.

There’s a type of blurriness that is acceptable, which is the re-stating of information in different words. Then there’s the blurriness of outright fabrication, which we consider unacceptable when we’re looking for facts.

Some might say that the output of large language models doesn't look all that different from a human writer's first draft, but, again, I think this is a superficial resemblance. Your first draft isn't an unoriginal idea expressed clearly; it's an original idea expressed poorly, and it is accompanied by your amorphous dissatisfaction, your awareness of the distance between what it says and what you want it to say.

Indeed, a useful criterion for gauging a large language model's quality might be the willingness of a company to use the text that it generates as training material for a new model. If the output of ChatGPT isn't good enough for GPT-4, we might take that as an indicator that it's not good enough for us, either.

THE CURSE OF RECURSION: TRAINING ON GENERATED DATA MAKES MODELS FORGET

Ilia Shumailov* University of Oxford
Zakhar Shumaylov* University of Cambridge
Yiren Zhao Imperial College London
Yarin Gal University of Oxford
Nicolas Papernot University of Toronto & Vector Institute
Ross Anderson University of Cambridge & University of Edinburgh

ABSTRACT

Stable Diffusion revolutionised image creation from descriptiv
demonstrated astonishing performance across a variety of lang
language models to the general public. It is now clear that larg
stay, and will bring about drastic change in the whole ecosys
paper we consider what the future might hold. What will happ
much of the language found online? We find that use of mode
irreversible defects in the resulting models, where tails of the
We refer to this effect as *model collapse*[1] and show that it ca
Gaussian Mixture Models and LLMs. We build theoretical i
portray its ubiquity amongst all learned generative models. V

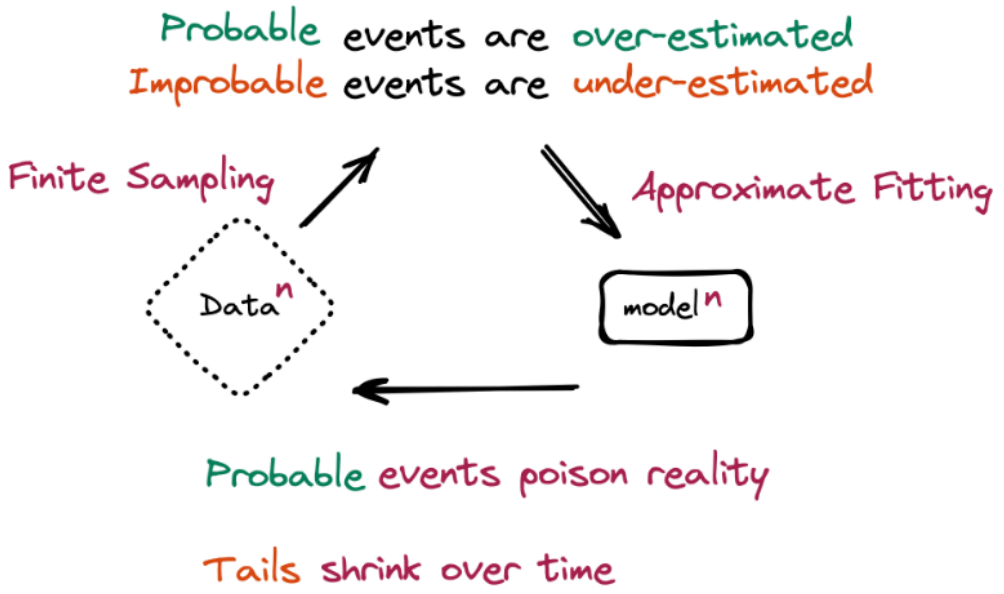


Figure 1: *Model Collapse* refers to a degenerative learning process where models start forgetting improbable events over time, as the model becomes poisoned with its own projection of reality.

N-gram Language Models

N-gram Language Models

Most modern applications use neural network language models. But to understand the language modeling task, we will start with the workhorse of the language model world: the **n-gram language model**.

Language Model

Goal: to predict

$$p(w_0, \ldots, w_n)$$

w_0 w_1 $\ldots$ w_n

the cat $\ldots$ nap.

$$p(w_n \mid w_0, \ldots, w_{n-1})$$

Sentence probability

How do we guess what the next word is?

I write this sitting in the kitchen…

p(eating | I write this sitting in the kitchen)

p(sink | I write this sitting in the kitchen)

eating
for
while
} pretty likely

zebra ungrammatical?

sink ← weird but ok

Frequency Counts

If we had a big corpus, we could see how often each of these sentences occurred:

count("I write this sitting in the kitchen sink") / count("I write this sitting in the kitchen")

versus

count("I write this sitting in the kitchen knife") / count("I write this sitting in the kitchen")

versus

count("I write this sitting in the kitchen chair") / count("I write this sitting in the kitchen")

This is called a **maximum likelihood estimation**.

Frequency Counts

But what if we never see any of these sentences?

0/ count("I write this sitting in the kitchen")

versus

0/ count("I write this sitting in the kitchen")

versus

0/ count("I write this sitting in the kitchen")

The probabilities would all be zero. But intuitively, some of these sentences still seem more likely than others…

The Chain Rule

Chain Rule of Probability:

$$P(X_1 \ldots X_n) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_{1:2}) \ldots P(X_n \mid X_{1:n-1})$$

$$= \Pi_{k=1}^{n} P(X_k \mid X_{1:k-1})$$

$$P(w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1 w_2) \ldots P(w_n \mid w_1 \ldots w_{n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1})$$

The Chain Rule

Great, so now we have:

P(“I”)P(“write” | “I”)P("this" | "I write") . . . P("sink" | "I write this sitting in the kitchen")

versus

P(“I”)P(“write” | “I”)P("this" | "I write") . . . P("knife" | "I write this sitting in the kitchen")

versus

P(“I”)P(“write” | “I”)P("this" | "I write") . . . P("chair" | "I write this sitting in the kitchen")