CS 232: Artificial Intelligence

Fall 2026

Prof. Carolyn Anderson
Wellesley College

Reminders

- HW 6 is due Monday
- My help hours: today 3-4 in W423
- Midterm 2 is on the horizon!

Giving Voice To Values:

WELLESLEY

THE 'HOW' OF VALUES-DRIVEN LEADERSHIP DEVELOPMENT

Speaker
Dr. Mary C. Gentile

Giving Voice to Values (GVV) is an innovative approach to values-driven leadership development in business education and the workplace pioneered by Dr. Mary C. Gentile.



Nov 5th Thursday
1-2pm
H105 | Lunch

RSVP:

?:qs104 accessibility@wellesley.edu exp: 11/9

An event organized by the If You Can See It You Can Be It team

ASK WELLESLEY ALUMS: CAREER JOURNEYS IN TECH



CLASS '25	CLASS '26	CLASS '23	CLASS '26	CLASS '26
Malahim Tariq PM @Microsoft	Yitzel Serna Founder of Baezl	Olivia Giandrea Senior SWE @SurveyMonkey	Katrina Huang PM @Crunchtime	Maha Yawar Masters Student @Columbia

NOVEMBER 1ST 2026

ZOOM 3-4PM

BUS PASS RAFFLE!

? lm114@wellesley.edu
Exp: 11/2

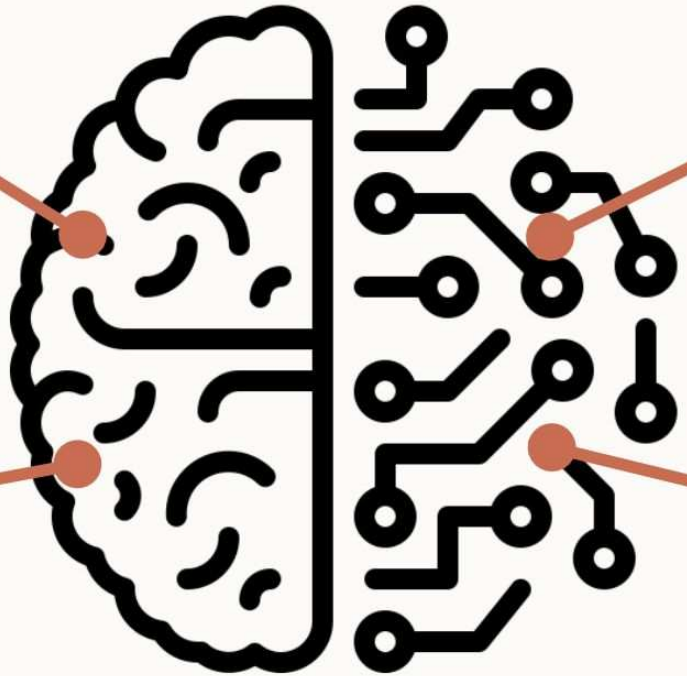
ADR: accessibility@wellesley.edu

Learn about the Cognitive and Linguistic Sciences Major

*The Science of **The Mind**, **Language**, **Computation**, and **Meaning***

Linguistics

Philosophy



Computer Science

Psychology

RSVP

Learn about our Spring '27 courses

Tuesday, Nov. 3rd 5-6pm SCI-H305

Snacks from Tatte
Meet Faculty & Majors

HELP US STUDY AI SUPPORTED TEXT ANALYSIS

60 mins

LOCATION: VIRTUAL

SIGN UP HERE!



WHAT YOU WILL DO?

Read short **K-drama** discussion excerpts, **identify ideas** and **themes**, try three kinds of **AI support**, and answer questions about your experience. Some tasks will be completed without AI. Participants will receive a $20 gift card upon completion of the study.

WHO CAN TAKE PART?

- Current Wellesley College student, age 18 or older.
- Comfortable reading and responding in English.
- Must not have independently completed a project involving coding qualitative data (introductory coursework or brief practice is allowed.)
- No K-drama knowledge required.
- Must have access to a laptop and an Internet connection.

Note: Taking part is voluntary. You may stop at any time. Your decision will not affect your grades, academic standing, employment, or access to College resources.

WELLESLEY

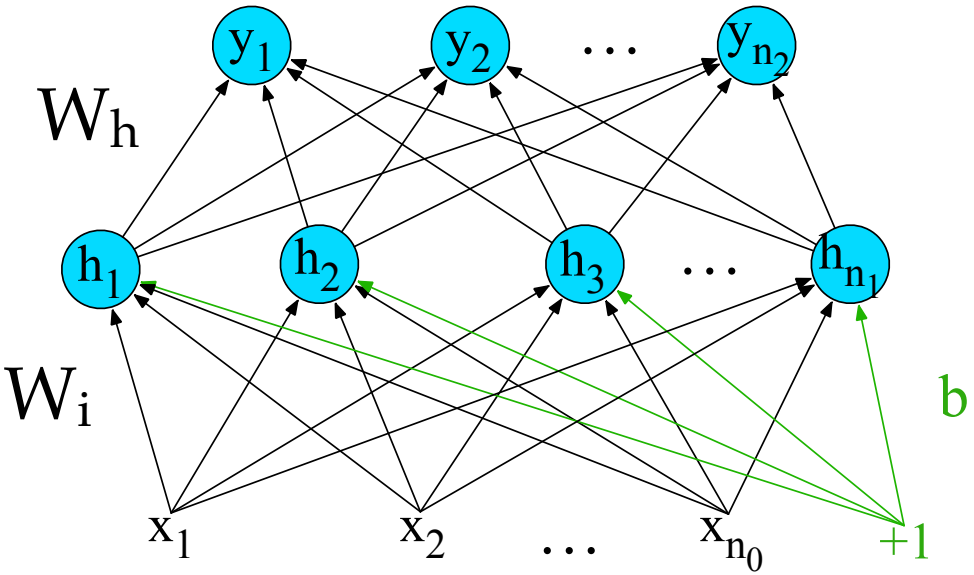
Questions @ hn103 | accessibility@wellesley.edu | exp 11/20/2026

Questions @ qs104 | accessibility@wellesley.edu | exp 02/28

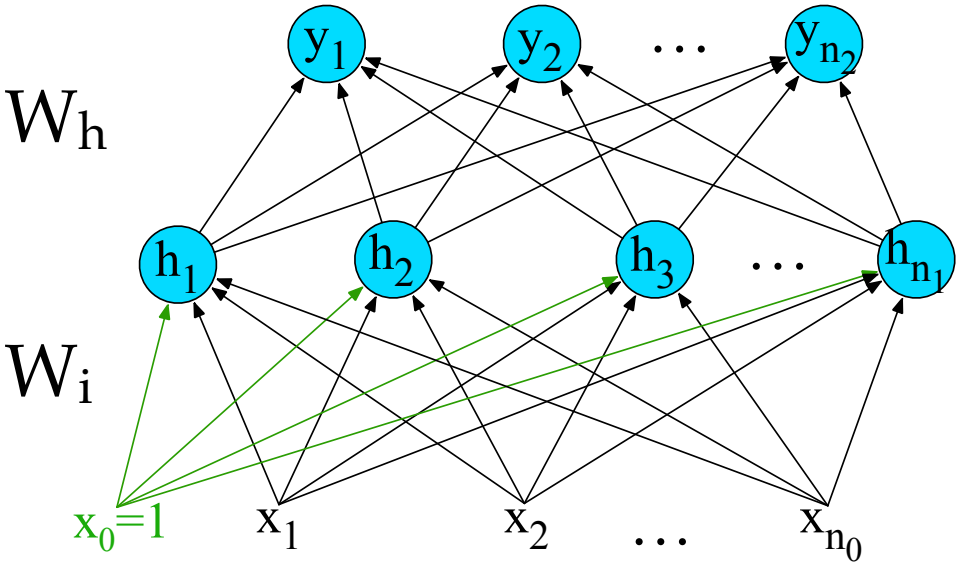
Recap: Neural Networks

Replacing the bias unit

Instead of:



We'll do this:



$b = w_0$

Training a Neural Network

Can we get back to cat pics, please?

Finally, we're ready to power up our supervised cat/dog classifier by **adding more layers**. This takes it from a **regression model** to a **neural network**.

Adding More Layers

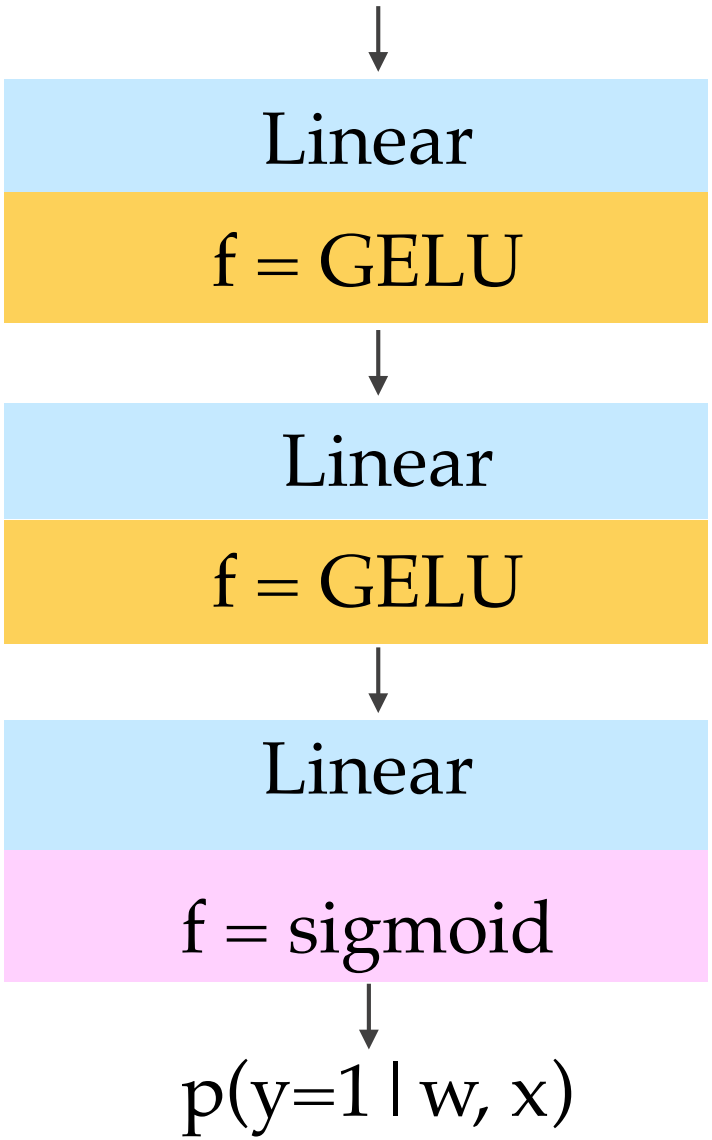
```
class ImageClassifier(nn.Module):
    """A single linear layer for logistic regression on image pixels."""

    def __init__(self, num_features, num_classes=2):
        super().__init__()
        self.layer1 = nn.Linear(num_features,num_features//30)
        self.gelu1 = nn.GELU()
        self.gelu2 = nn.GELU()
        self.layer2 = nn.Linear(num_features//30,num_features//180)
        self.classifier = nn.Linear(num_features//180, 1)

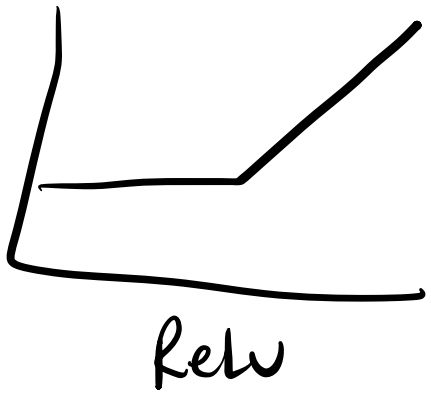
    def forward(self, features):
        a1 = self.gelu1(self.layer1(features))
        a2 = self.gelu2(self.layer2(a1))
        outputs = self.classifier(a2)
        return torch.sigmoid(outputs)
```

New Architecture

x (pixel values of photo)

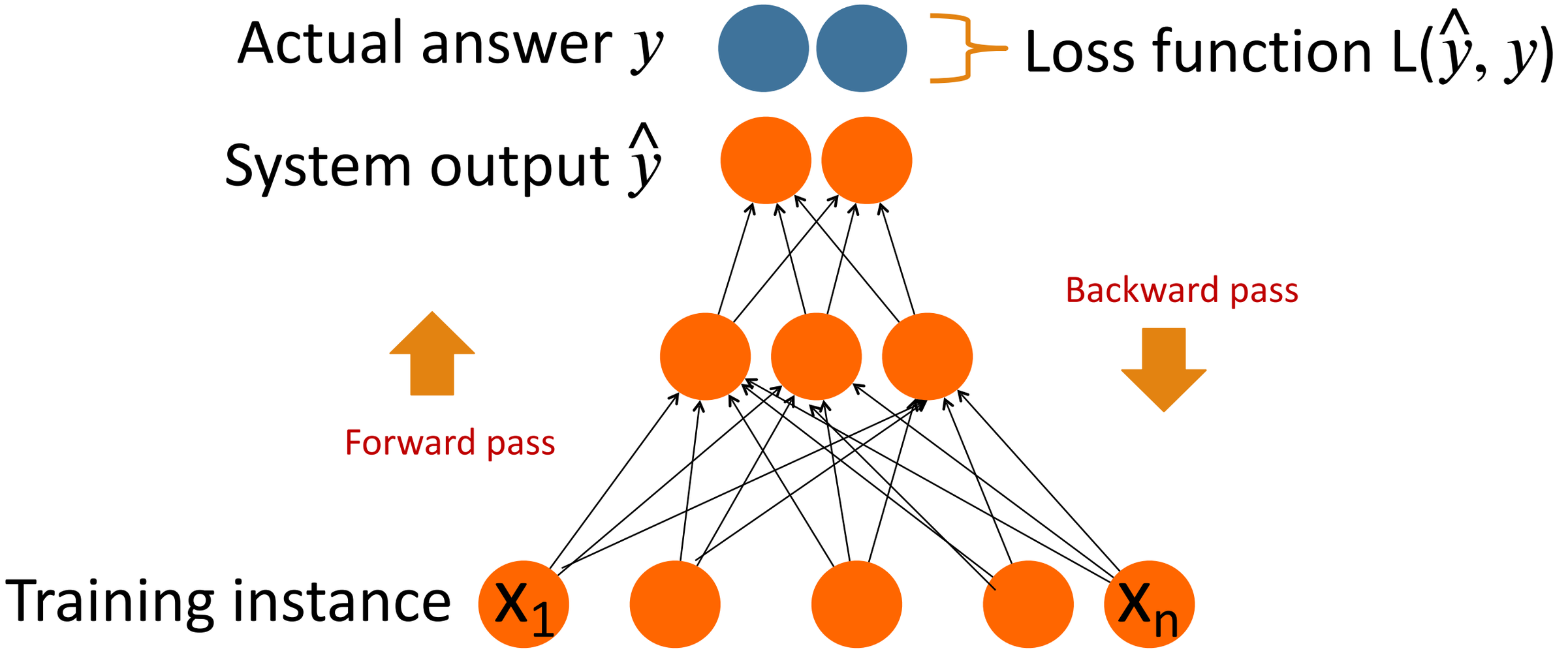



```
ImageClassifier(
  (layer1): Linear(in_features=32400, out_features=180, bias=True)
  (gelu1): GELU(approximate='none')
  (gelu2): GELU(approximate='none')
  (layer2): Linear(in_features=180, out_features=2, bias=True)
  (classifier): Linear(in_features=2, out_features=1, bias=True)
)
Trainable parameters: 5,832,545
```

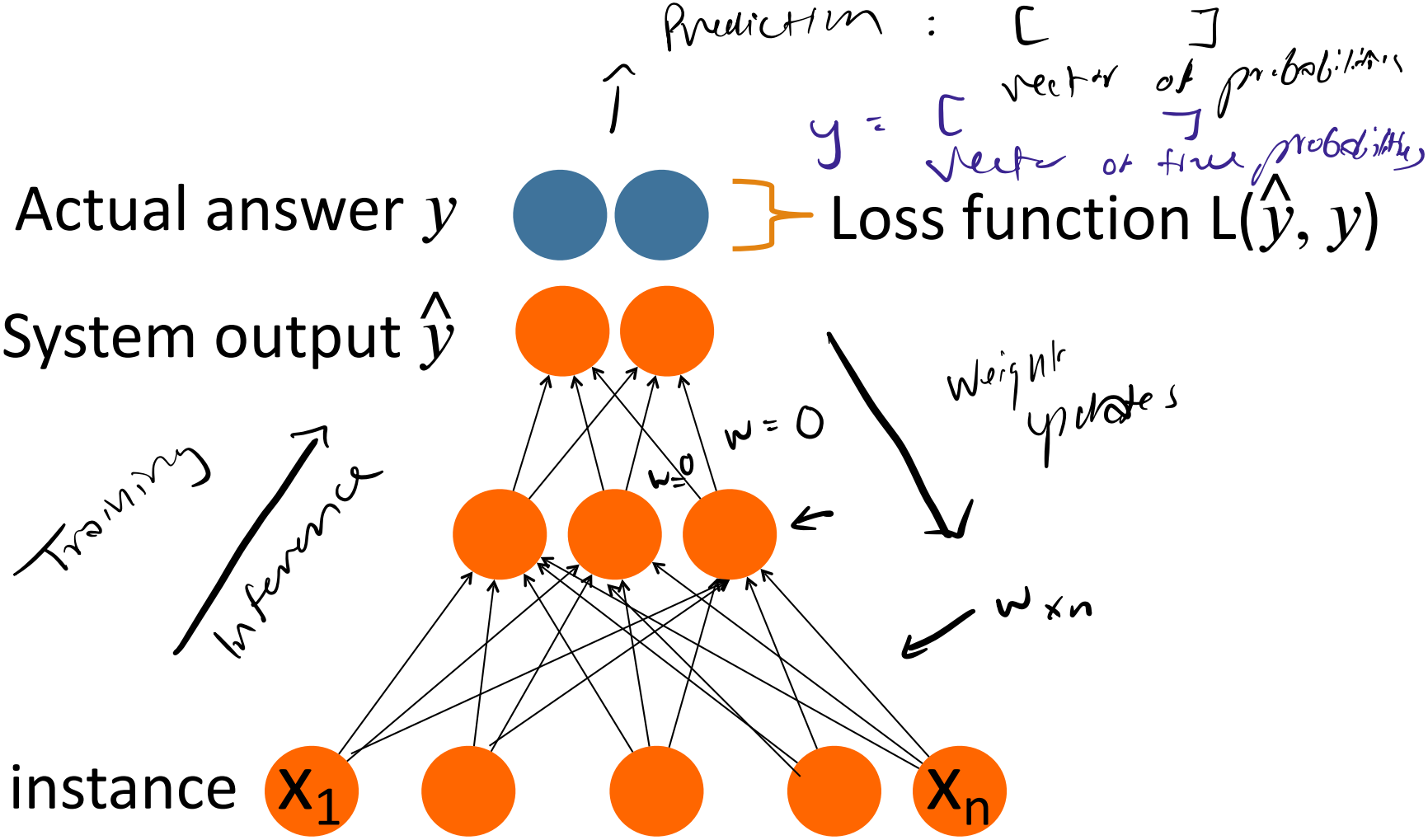


Training a Neural Network

Intuition: training a 2-layer Network



Intuition: training a 2-layer Network



Intuition: Training a 2-layer network

For every training tuple (x, y)

inference: make a prediction

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

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:

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

For multinomial classification, we use cross entropy loss:

$$L_{CE}(\hat{y}, y) = -\log \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}$$

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:

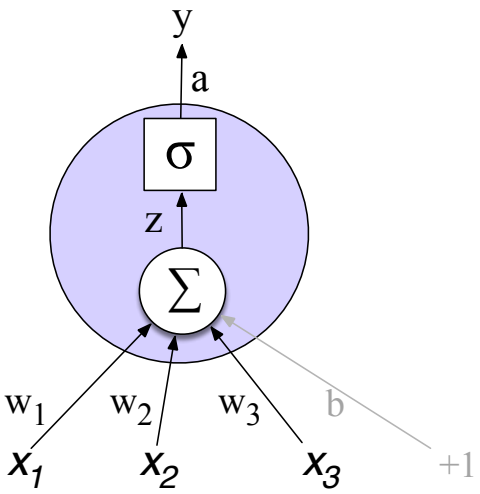
$$L_{CE}(\hat{y},y) = -\log p(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

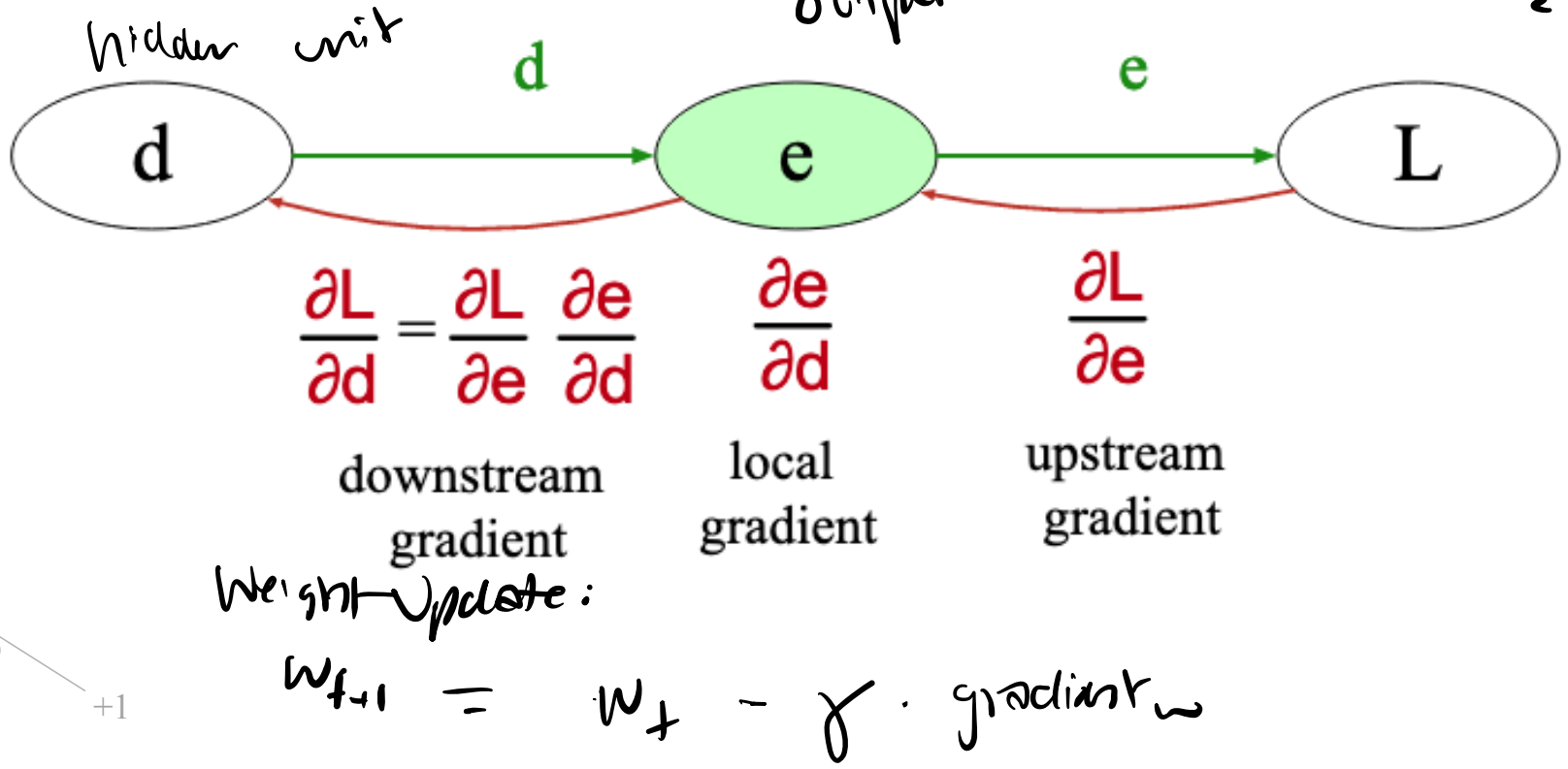
$$L_{\text{CE}}(\hat{y},y) = -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))]$$

For multinomial classification, we use cross entropy loss:

$$L_{CE}(\hat{y},y) = -\log\frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{K}\exp(\mathbf{z}_j)} \quad \text{(where } i \text{ is the correct class)}$$

Computing Gradients





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.

Why Computation Graphs

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

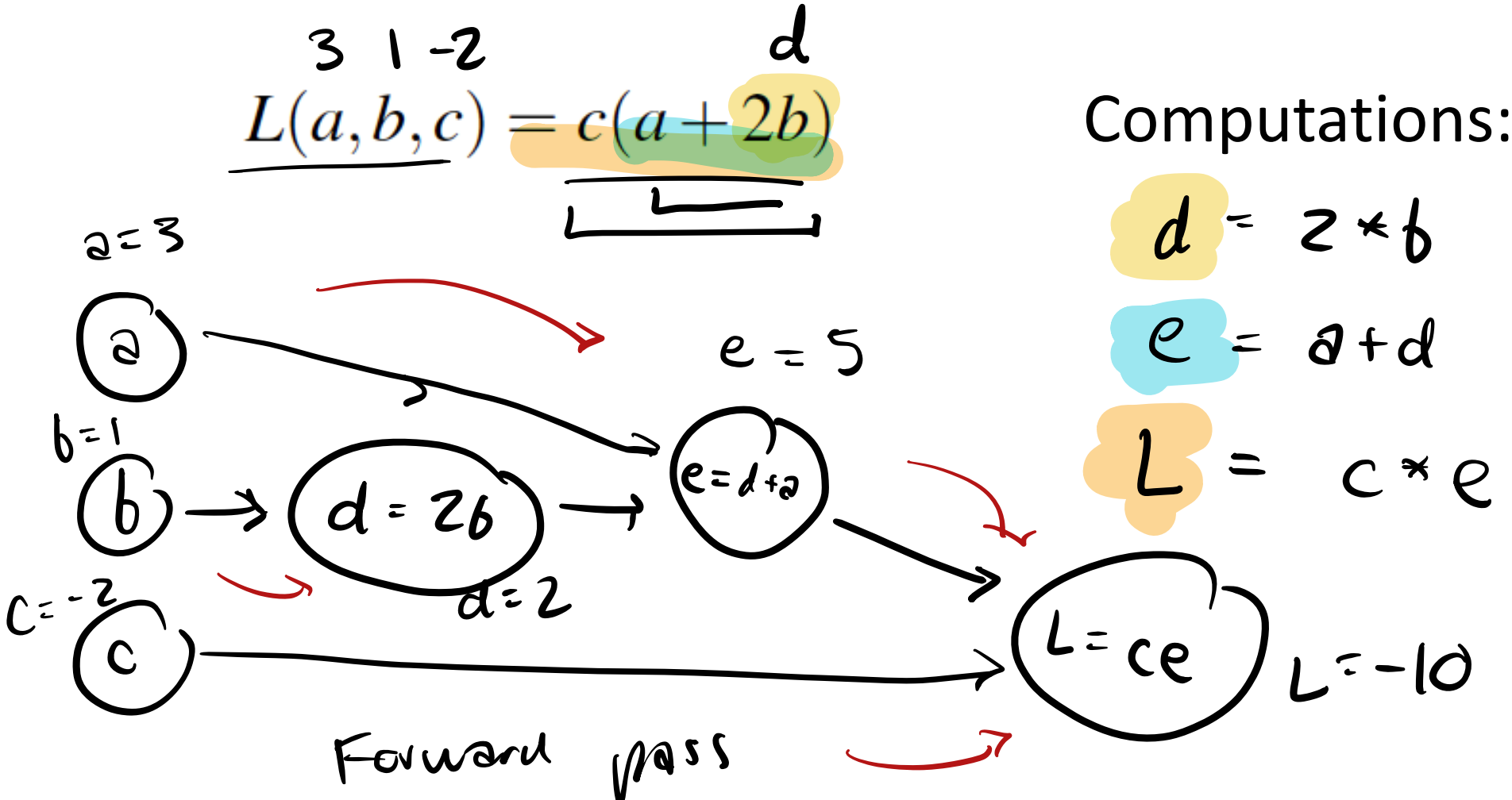
Solution: *error backpropagation* (Rumelhart, Hinton, Williams, 1986)

- Backprop is a special case of *backward differentiation*

Computation Graphs

Computation Graphs

A computation graph represents the process of computing a mathematical expression



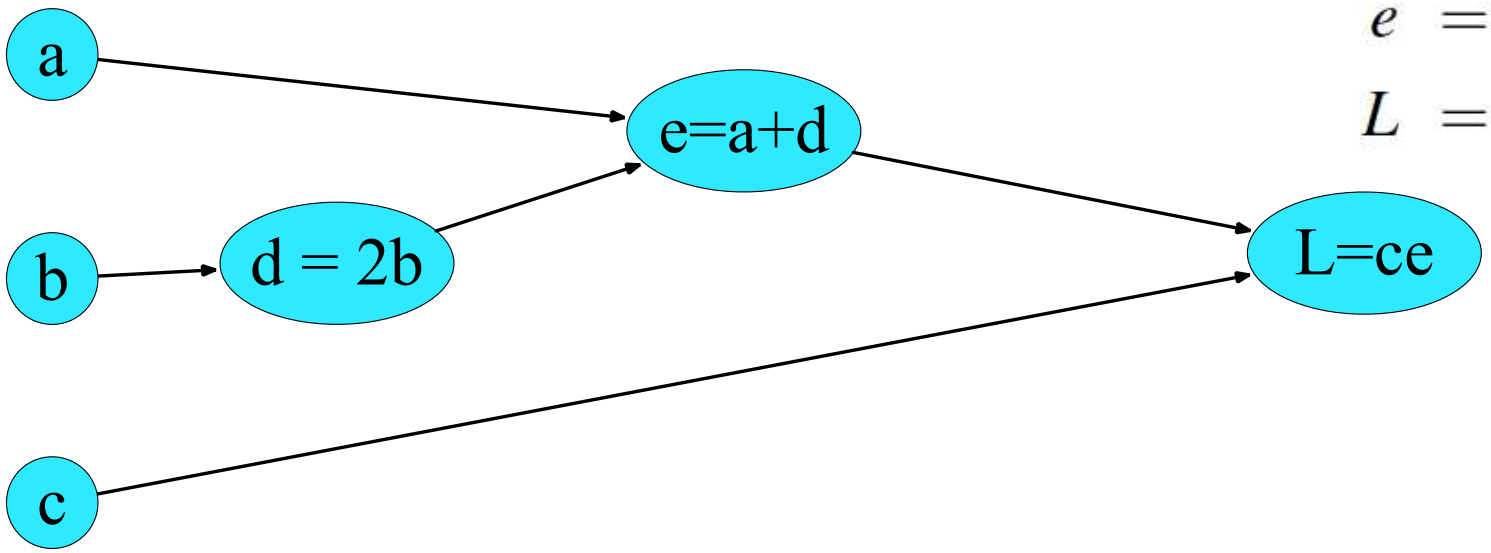
Computation Graphs

A computation graph represents the process of computing a mathematical expression

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

Computations:

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

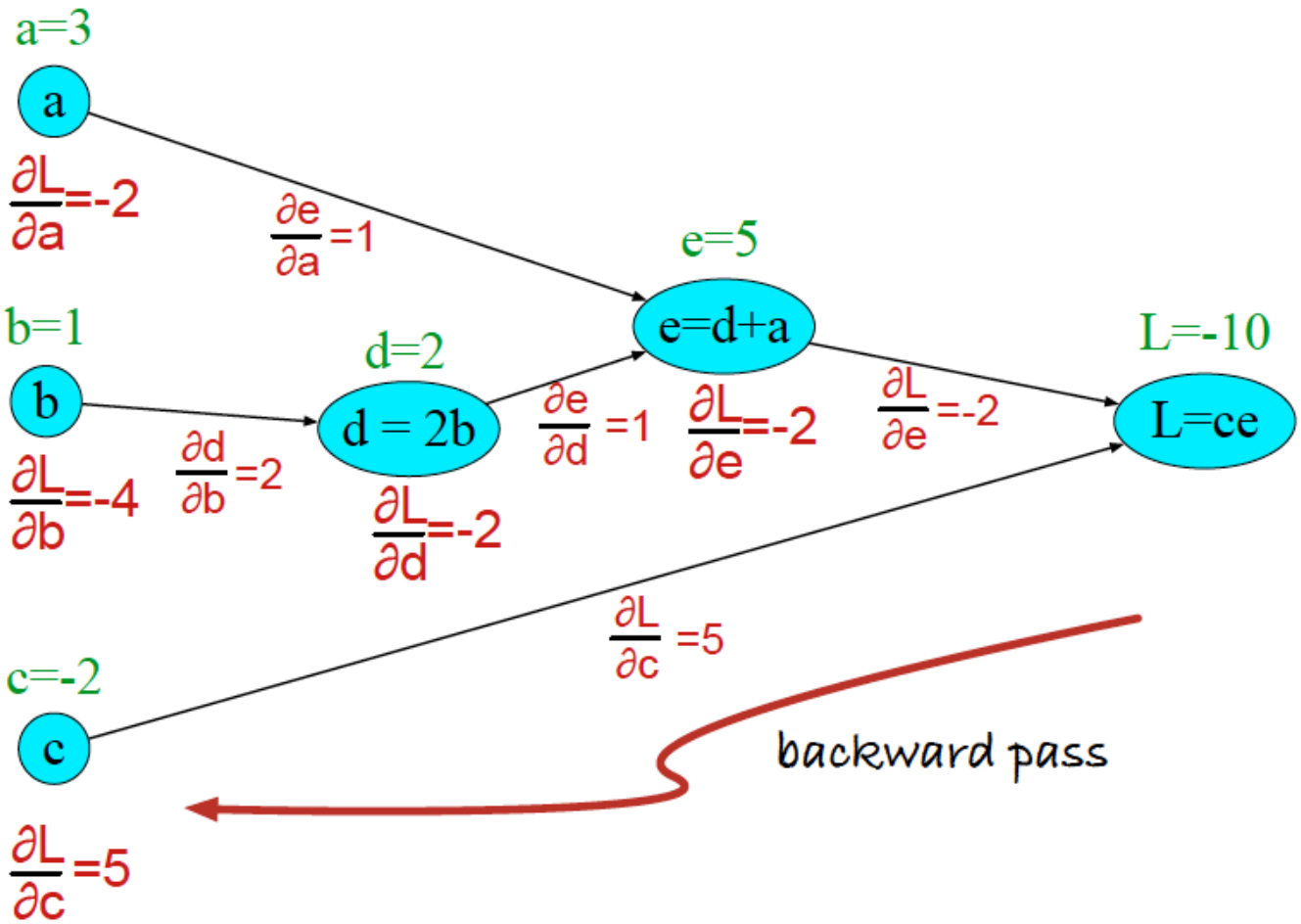


Backwards differentiation in computation graphs

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.

Backward Pass



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.

Using feedforward networks

Use cases for feedforward networks

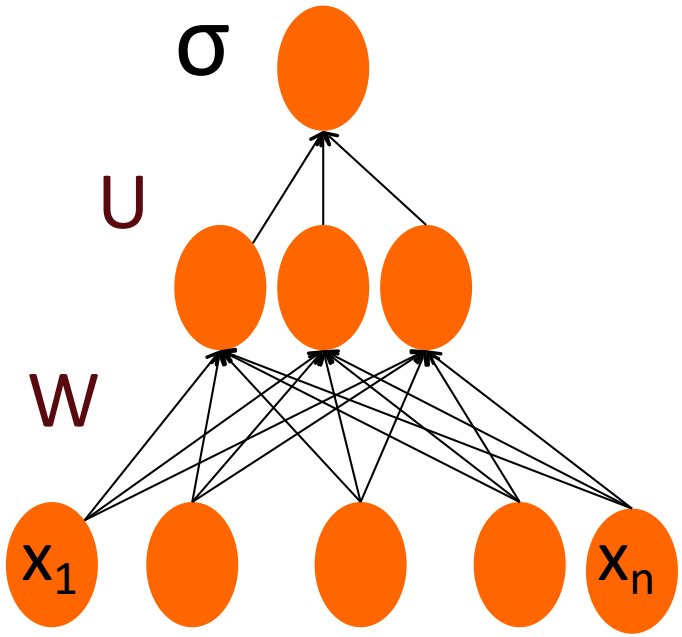
Let's revisit sentiment analysis with feedforward networks (state-of-the-art systems use more powerful architectures).

Classification: Sentiment Analysis

We could do exactly what we did with logistic regression

Input layer are ~~binary~~ features as before

Output layer is 0 or 1

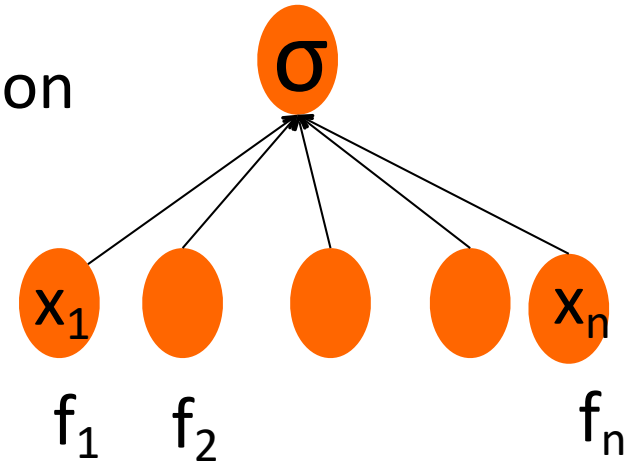


Sentiment Features

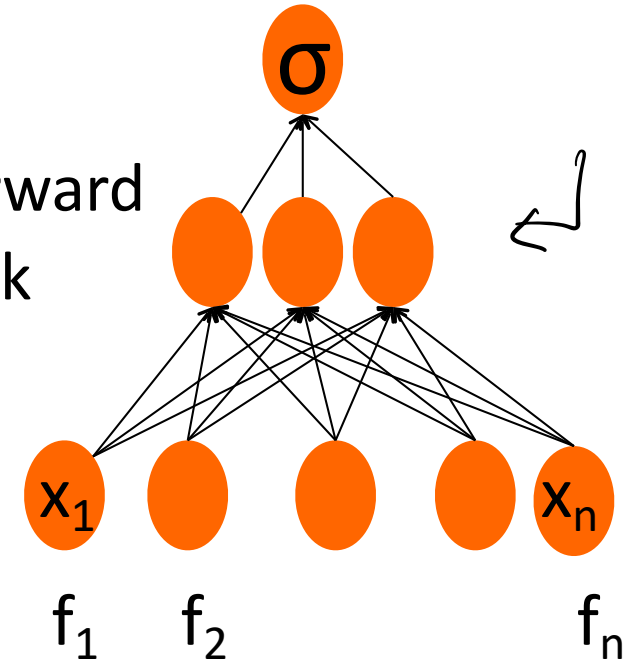
Var	Definition
x_1	count(positive lexicon words $\in$ doc)
x_2	count(negative lexicon words $\in$ doc)
x_3	$\begin{cases} 1 & \text{if } \text{“no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

Feedforward nets for simple classification

Logistic Regression



2-layer feedforward network



Just add a hidden layer to logistic regression!

This allows the network to use non-linear interactions between features (which *hopefully* improves performance).

Even better: representation learning

The real power of deep learning comes from the ability to learn features from the data, instead of using hand-built human-engineered features for classification.

We'll pick up here next class.