10-301/601: Introduction to Machine Learning Lecture 13 – Differentiation

Henry Chai

6/13/23

Front Matter

- Announcements
  - PA3 released 6/8, due 6/15 at 11:59 PM
  - Quiz 4: Neural Networks on 6/20 (next Tuesday)
  - No lecture **or OH** on 6/19 (next Monday) for Juneteenth
  - Midterm on 6/23, one week from Friday
- Recommended Readings
  - None

Midterm Logistics

- Time and place:
  - Friday, 6/23 from 12:00PM to 3:00 PM in **DH 2302**
- Closed book/notes
  - 1-page cheatsheet allowed, both back and front; can be typeset or handwritten

Midterm Coverage

- Lectures: 1 – 14 (through tomorrow's lecture)
  - Foundations: probability, linear algebra, calculus
  - Important concepts: inductive bias, overfitting, model selection/hyperparameter optimization, regularization
  - Models: decision trees, kNN, Perceptron, linear regression, logistic regression, neural networks
  - Methods: (stochastic) gradient descent, closed-form optimization, backpropagation, MLE/MAP

Midterm Preparation

- Review midterm practice problems, posted to the course website (under Recitations)
- Attend the exam review recitation on 6/20 (after the quiz)
- Review this year's quizzes and study guides
- Consider whether you understand the "Key Takeaways" for each lecture / section
- Write your cheat sheet

Recall: Forward Propagation for Making Predictions

- Input: weights $W^{(1)}, \ldots, W^{(L)}$ and a query data point $\boldsymbol{x}$
- Initialize $\boldsymbol{o}^{(0)} = [1, \boldsymbol{x}]^T$
- For $l = 1, \ldots, L$
  - $\boldsymbol{s}^{(l)} = W^{(l)} \boldsymbol{o}^{(l-1)}$
  - $\boldsymbol{o}^{(l)} = \left[1, \theta\left(\boldsymbol{s}^{(l)}\right)\right]^T$
- Output: $h_{W^{(1)}, \ldots, W^{(L)}}(\boldsymbol{x}) = \boldsymbol{o}^{(L)}$

Recall: Gradient Descent for Learning

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}, \eta^{(0)}$
- Initialize all weights $W_{(0)}^{(1)}, \ldots, W_{(0)}^{(L)}$ to small, random numbers and set $t = 0$ (???)
- While TERMINATION CRITERION is not satisfied (???)
  - For $l = 1, \ldots, L$
    - Compute $G^{(l)} = \nabla_{W^{(l)}} \ell_{\mathcal{D}}\left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}\right)$ (???)
    - Update $W^{(l)}$: $W_{(t+1)}^{(l)} = W_{(t)}^{(l)} - \eta_0 G^{(l)}$
  - Increment t: $t = t + 1$
- Output: $W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}$

Matrix Calculus

Numerator

Types of Derivatives	**scalar**	**vector**	**matrix**
scalar	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{Y}}{\partial x}$
vector	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$
matrix	$\frac{\partial y}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$

Denominator

Table courtesy of Matt Gormley

Matrix Calculus: Denominator Layout

- Derivatives of a scalar always have the *same shape* as the entity that the derivative is being taken with respect to.

Types of Derivatives	**scalar**
scalar	$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x}\right]$
vector	$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix}$
matrix	$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial X_{11}} & \frac{\partial y}{\partial X_{12}} & \cdots & \frac{\partial y}{\partial X_{1Q}} \\ \frac{\partial y}{\partial X_{21}} & \frac{\partial y}{\partial X_{22}} & \cdots & \frac{\partial y}{\partial X_{2Q}} \\ \vdots & & & \vdots \\ \frac{\partial y}{\partial X_{P1}} & \frac{\partial y}{\partial X_{P2}} & \cdots & \frac{\partial y}{\partial X_{PQ}} \end{bmatrix}$

Table courtesy of Matt Gormley

Matrix Calculus: Denominator Layout

Types of Derivatives	**scalar**	**vector**
scalar	$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x}\right]$	$\frac{\partial \mathbf{y}}{\partial x} = \left[\frac{\partial y_1}{\partial x} \quad \frac{\partial y_2}{\partial x} \quad \cdots \quad \frac{\partial y_N}{\partial x}\right]$
vector	$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_N}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_N}{\partial x_2} \\ \vdots & & & \\ \frac{\partial y_1}{\partial x_P} & \frac{\partial y_2}{\partial x_P} & \cdots & \frac{\partial y_N}{\partial x_P} \end{bmatrix}$

Table courtesy of Matt Gormley

Three Approaches to Differentiation

- Given $f\colon \mathbb{R}^D \rightarrow \mathbb{R}$, compute $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = {}^{\partial f(\boldsymbol{x})}/_{\partial \boldsymbol{x}}$

1. Finite difference method

2. Symbolic differentiation

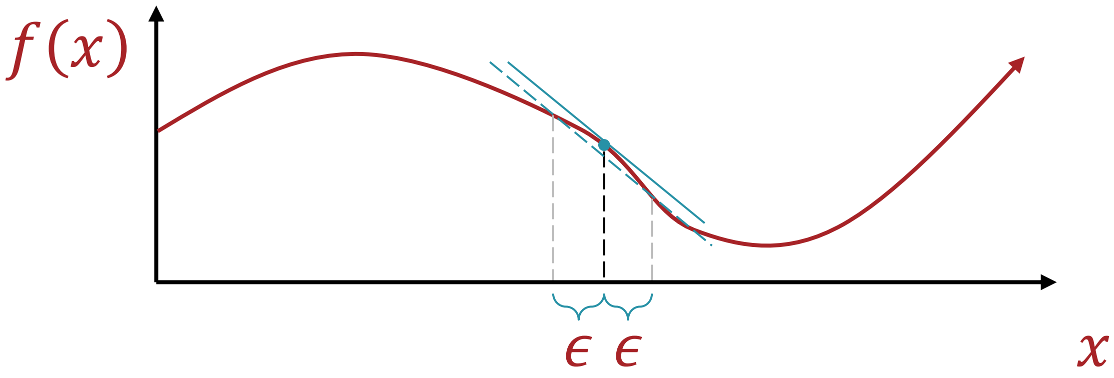
3. Automatic differentiation (reverse mode)

Approach 1: Finite Difference Method

- Given $f\colon \mathbb{R}^D \to \mathbb{R}$, compute $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = {}^{\partial f(\boldsymbol{x})}/_{\partial \boldsymbol{x}}$

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} \approx \frac{f(\boldsymbol{x} + \epsilon \boldsymbol{d}_i) - f(\boldsymbol{x} - \epsilon \boldsymbol{d}_i)}{2\epsilon}$$

where $\boldsymbol{d}_i$ is a one-hot vector with a 1 in the i^{th} position



- We want ϵ to be small to get a good approximation but we run into floating point issues when ϵ is too small

- Getting the full gradient requires computing the above approximation for each dimension of the input

Approach 1: Finite Difference Method Example

- Given

$$y = f(x, z) = e^{xz} + \frac{xz}{\ln(x)} + \frac{\sin(\ln(x))}{xz}$$

what are $\partial y / \partial x$ and $\partial y / \partial z$ at $x = 2, z = 3$?

```
>>> import math
>>> y = lambda x,z:
math.exp(x*z)+(x*z)/math.log(x)+math.sin(math.log(x))/(x*z)
>>> x = 2
>>> z = 3
>>> e = 10**-8
>>> dydx = (y(x+e,z)-y(x-e,z))/(2*e)
>>> dydz = (y(x,z+e)-y(x,z-e))/(2*e)
>>> print(dydx, dydz)
```


Example courtesy of Matt Gormley

Three Approaches to Differentiation

- Given $f\colon \mathbb{R}^D \to \mathbb{R}$, compute $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = {}^{\partial f(\boldsymbol{x})}/_{\partial \boldsymbol{x}}$

1. Finite difference method
   - Requires the ability to call $f(\boldsymbol{x})$
   - Great for checking accuracy of implementations of more complex differentiation methods
   - Computationally expensive for high-dimensional inputs
2. Symbolic differentiation
3. Automatic differentiation (reverse mode)

Approach 2: Symbolic Differentiation

- Given

$$y = f(x, z) = e^{xz} + \frac{xz}{\ln(x)} + \frac{\sin(\ln(x))}{xz}$$

what are ${\partial y}/{\partial x}$ and ${\partial y}/{\partial z}$ at $x = 2, z = 3$?

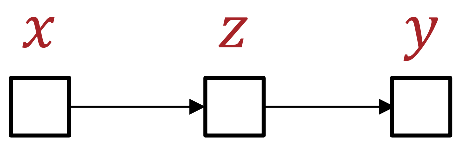
- Looks like we're gonna need the chain rule!

Example courtesy of Matt Gormley

The Chain Rule of Calculus

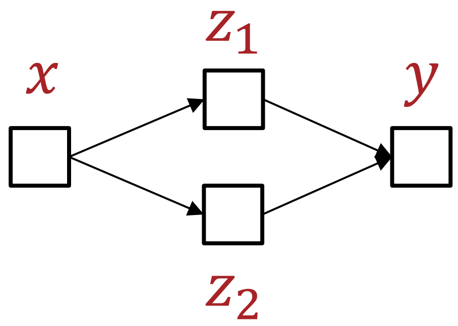
- If $y = f(z)$ and $z = g(x)$ then

the corresponding computation graph is



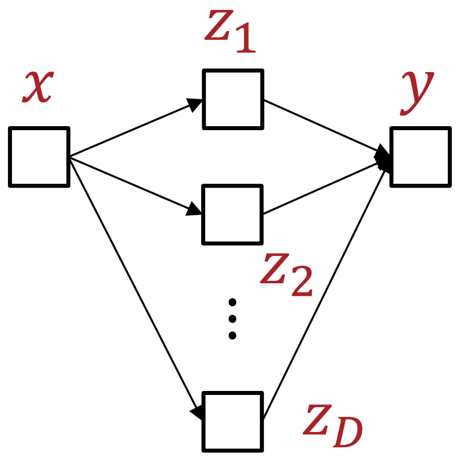
$$\Longrightarrow \quad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial z}\frac{\partial z}{\partial x}$$

- If $y = f(z_1, z_2)$ and $z_1 = g_1(x), z_2 = g_2(x)$ then



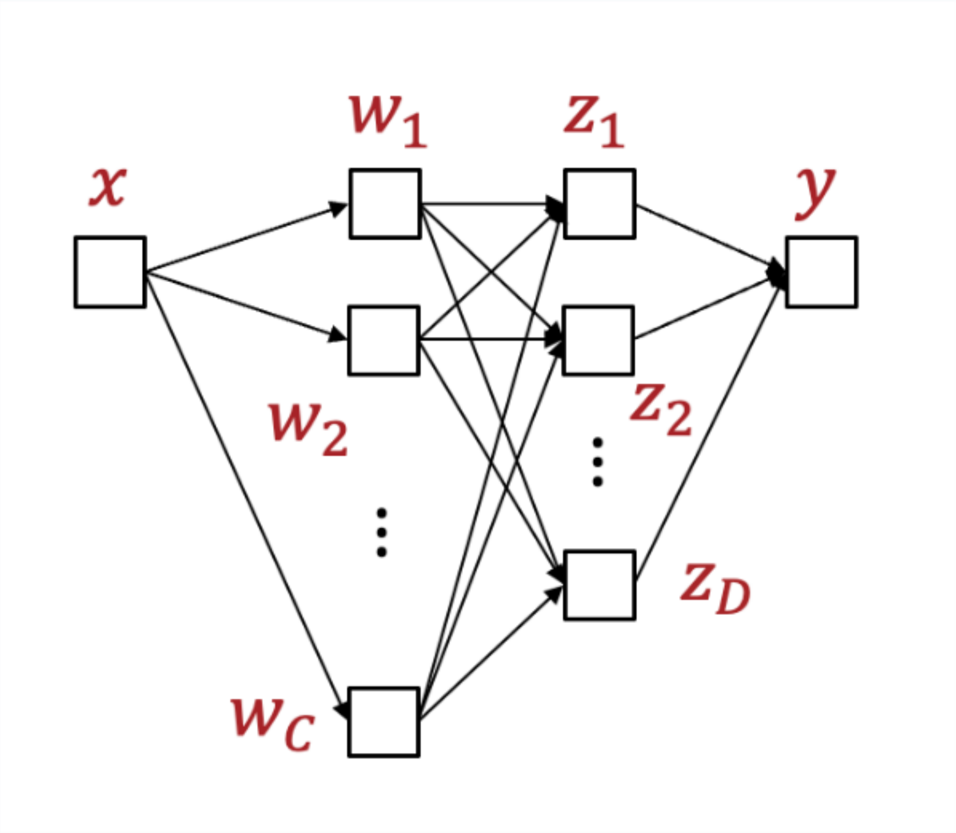
$$\Longrightarrow \quad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial z_1}\frac{\partial z_1}{\partial x} + \frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial x}$$

- If $y = f(\boldsymbol{z})$ and $\boldsymbol{z} = g(x)$ then



$$\Longrightarrow \quad \frac{\partial y}{\partial x} = \sum_{d=1}^{D} \frac{\partial y}{\partial z_d}\frac{\partial z_d}{\partial x}$$

Suppose $y = f(\mathbf{z})$, $\mathbf{z} = h(\mathbf{w})$ and $\mathbf{w} = g(x)$. Does the equation $\frac{\partial y}{\partial x} = \sum_{d=1}^{D} \frac{\partial y}{\partial z_d} \frac{\partial z_d}{\partial x}$ still hold?



- Yes
- No
- Unsure

Approach 2: Symbolic Differentiation

- Given

$$y = f(x, z) = e^{xz} + \frac{xz}{\ln(x)} + \frac{\sin(\ln(x))}{xz}$$

what are $\partial y / \partial x$ and $\partial y / \partial z$ at $x = 2, z = 3$?

$$\frac{\partial y}{\partial x} = \frac{\partial}{\partial x}(e^{xz}) + \frac{\partial}{\partial x}\left(\frac{xz}{\ln(x)}\right) + \frac{\partial}{\partial x}\left(\frac{\sin(\ln(x))}{xz}\right)$$

$$= ze^{xz} + \frac{z}{\ln(x)} - \frac{z}{\ln(x)^2} + \frac{\cos(\ln(x))}{x^2 z} - \frac{\sin(\ln(x))}{x^2 z}$$

$$= 3e^6 + \frac{3}{\ln(2)} - \frac{3}{\ln(2)^2} + \frac{\cos(\ln(2))}{12} - \frac{\sin(\ln(2))}{12}$$

$$\frac{\partial y}{\partial z} = \frac{\partial}{\partial z}(e^{xz}) + \frac{\partial}{\partial z}\left(\frac{xz}{\ln(x)}\right) + \frac{\partial}{\partial z}\left(\frac{\sin(\ln(x))}{xz}\right)$$

$$= 2e^6 + \frac{2}{\ln(2)} - \frac{\sin(\ln(2))}{18}$$

Example courtesy of Matt Gormley

Three Approaches to Differentiation

- Given $f\colon \mathbb{R}^D \to \mathbb{R}$, compute $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = {}^{\partial f(\boldsymbol{x})}/_{\partial \boldsymbol{x}}$

1. Finite difference method
   - Requires the ability to call $f(\boldsymbol{x})$
   - Great for checking accuracy of implementations of more complex differentiation methods
   - Computationally expensive for high-dimensional inputs
2. Symbolic differentiation
   - Requires systematic knowledge of derivatives
   - Can be computationally expensive if poorly implemented
3. Automatic differentiation (reverse mode)

Approach 3: Automatic Differentiation (reverse mode)

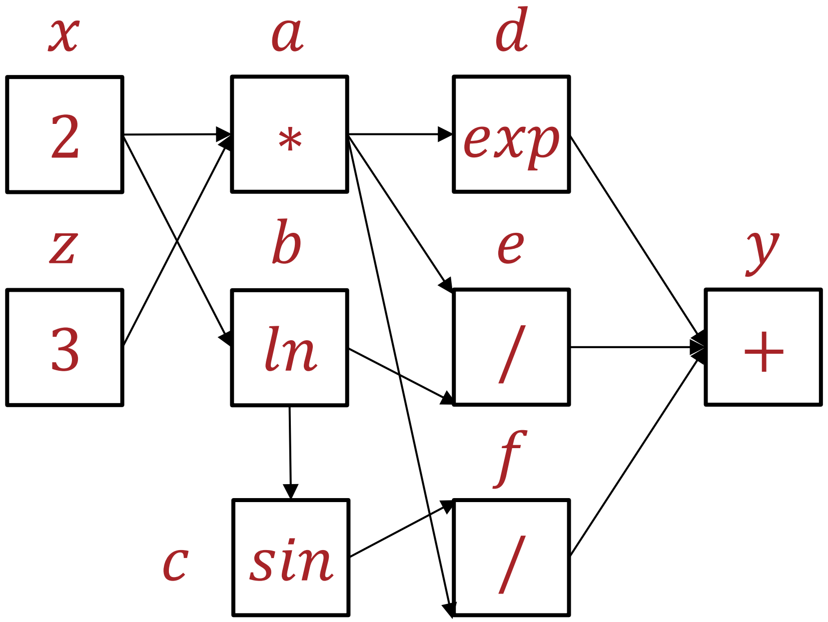
- Given

$$y = f(x, z) = e^{xz} + \frac{xz}{\ln(x)} + \frac{\sin(\ln(x))}{xz}$$

what are ${}^{\partial y}/_{\partial x}$ and ${}^{\partial y}/_{\partial z}$ at $x = 2, z = 3$?

- First define some intermediate quantities, draw the computation graph and run the "forward" computation

$$a = xz$$
$$b = \ln(x)$$
$$c = \sin(b)$$
$$d = e^a$$
$$e = {}^{a}/_{b}$$
$$f = {}^{c}/_{a}$$
$$y = d + e + f$$



Example courtesy of Matt Gormley

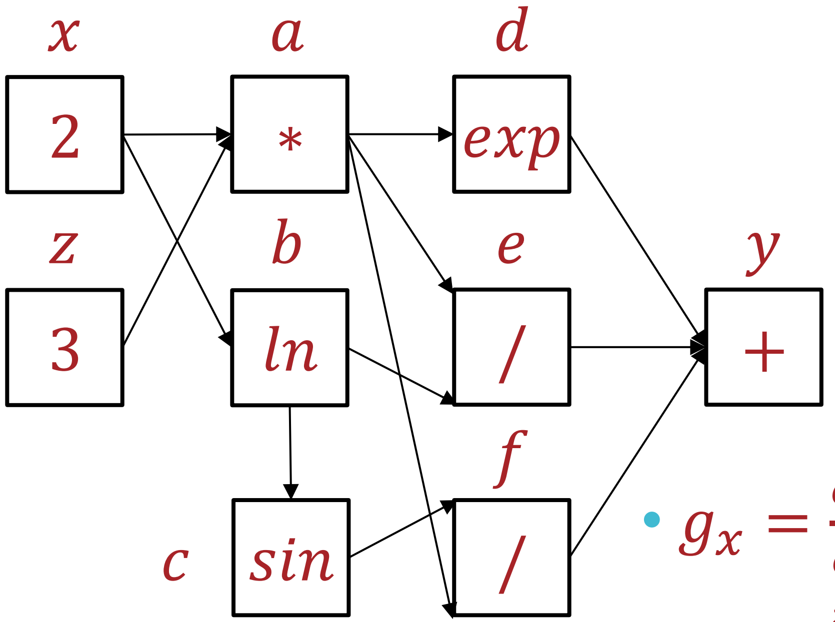
Approach 3: Automatic Differentiation (reverse mode)

- Given

$$y = f(x,z) = e^{xz} + \frac{xz}{\ln(x)} + \frac{\sin(\ln(x))}{xz}$$

what are $\partial y / \partial x$ and $\partial y / \partial z$ at $x = 2, z = 3$?

- Then compute partial derivatives, starting from y and working back



- $g_y = \frac{\partial y}{\partial y} = 1$
- $g_d = g_e = g_f = 1$
- $g_c = \frac{\partial y}{\partial c} = \frac{\partial y}{\partial f}\frac{\partial f}{\partial c} = g_f\left(\frac{1}{a}\right)$
- $g_b = \frac{\partial y}{\partial b} = \frac{\partial y}{\partial e}\frac{\partial e}{\partial b} + \frac{\partial y}{\partial c}\frac{\partial c}{\partial b}$
  $= g_e\left(-\frac{a}{b^2}\right) + g_c(\cos(b))$
- $g_a = \frac{\partial y}{\partial a} = \frac{\partial y}{\partial f}\frac{\partial f}{\partial a} + \frac{\partial y}{\partial e}\frac{\partial e}{\partial a} + \frac{\partial y}{\partial d}\frac{\partial d}{\partial a}$
  $= g_f\left(\frac{-c}{a^2}\right) + g_e\left(\frac{1}{b}\right) + g_d(e^a)$
- $g_x = \frac{\partial y}{\partial x} = \frac{\partial y}{\partial b}\frac{\partial b}{\partial x} + \frac{\partial y}{\partial a}\frac{\partial a}{\partial x} = g_b\left(\frac{1}{x}\right) + g_a(z)$
- $g_z = \frac{\partial y}{\partial z} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial z} = g_a(x)$

Example courtesy of Matt Gormley

Three Approaches to Differentiation

- Given $f: \mathbb{R}^D \to \mathbb{R}$, compute $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = {}^{\partial f(\boldsymbol{x})}/_{\partial \boldsymbol{x}}$

1. Finite difference method
   - Requires the ability to call $f(\boldsymbol{x})$
   - Great for checking accuracy of implementations of more complex differentiation methods
   - Computationally expensive for high-dimensional inputs
2. Symbolic differentiation
   - Requires systematic knowledge of derivatives
   - Can be computationally expensive if poorly implemented
3. Automatic differentiation (reverse mode)
   - Requires systematic knowledge of derivatives *and* an algorithm for computing $f(\boldsymbol{x})$
   - Computational cost of computing ${}^{\partial f(\boldsymbol{x})}/_{\partial \boldsymbol{x}}$ is proportional to the cost of computing $f(\boldsymbol{x})$

Computation Graph 10-301/601 Conventions

- The diagram represents *an algorithm*
- Nodes are rectangles with one node per intermediate variable in the algorithm
- Each node is labeled with the function that it computes (inside the box) and the variable name (outside the box)
- Edges are directed and do not have labels
- For neural networks:
  - Each weight, feature value, label and *bias term* appears as a node
  - We *can* include the loss function

Neural Network Diagram Conventions

- The diagram represents a *neural network*
- Nodes are circles with one node per hidden unit
- Each node is labeled with the variable corresponding to the hidden unit
- Edges are directed and each edge is labeled with its weight
- Following standard convention, the bias term is typically *not* shown as a node, but rather is assumed to be part of the activation function i.e., its weight does not appear in the picture anywhere.
- The diagram typically does *not* include any nodes related to the loss computation

Key Takeaways

- Denominator layout for matrix calculus
- Finite difference method is a simple but computationally expensive approximation technique
  - ***You should use this to unit test your implementation of backpropagation!***
- Symbolic differentiation is the "default" differentiation method but can also also be computationally expensive
- Automatic differentiation (reverse mode) saves intermediate quantities for computational efficiency
  - Backpropagation is an instance of automatic differentiation in the reverse mode