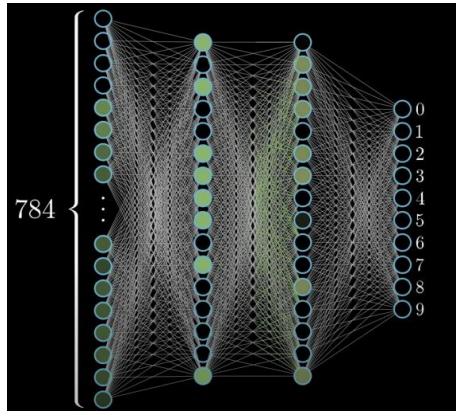
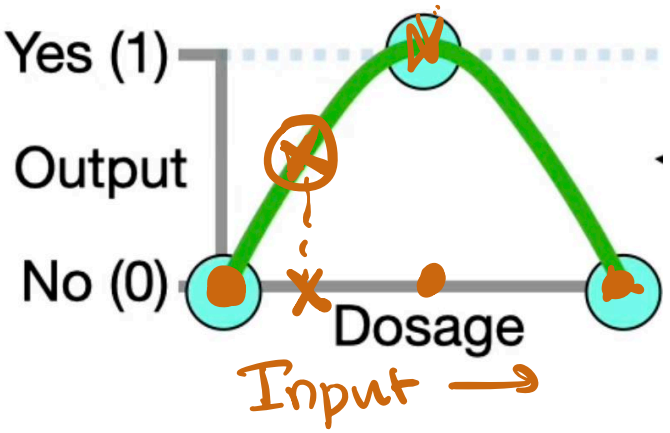
Link to handout: https://bit.ly/NN-DFS-Backprop



PA03: TRAINING NEURAL NETWORKS USING

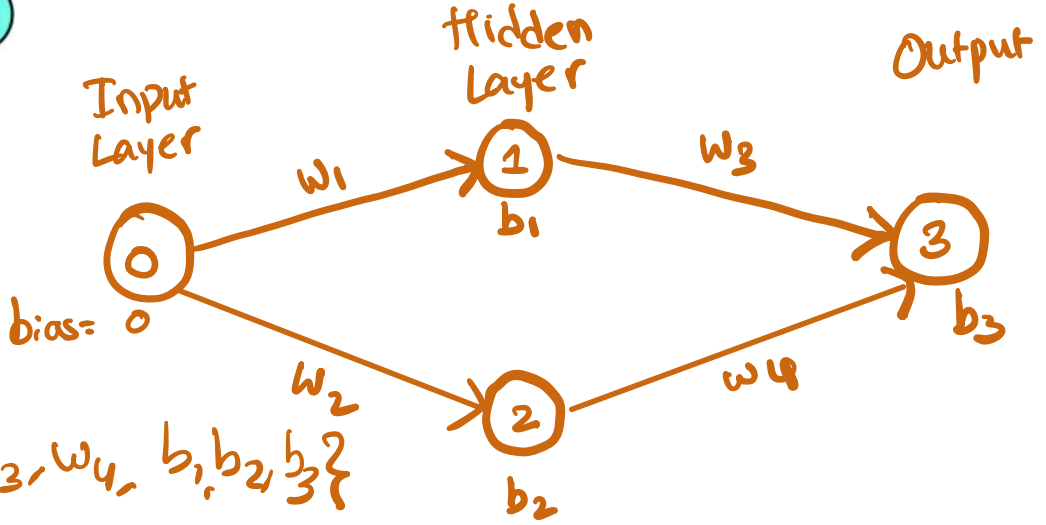
DFS-BASED BACKPROPAGATION

Neural Net to predict effectiveness of a drug



Draw the neural network that was used to predict whether a drug's dosage is effective against a virus.

How many parameters does the NN have?

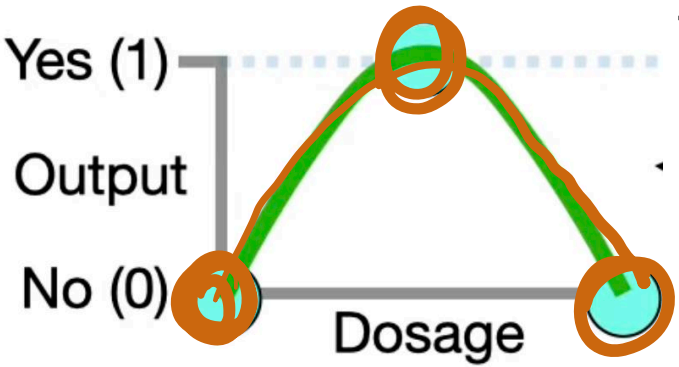


7

parameters = $\{w_1, w_2, w_3, w_4, b_1, b_2, b_3\}$

 Pre-lecture video on Backpropagation in NNs: https://youtu.be/IN2XmBhILt4?si=bnDft-3T4DQ2iO9X

What is “training” the neural net?

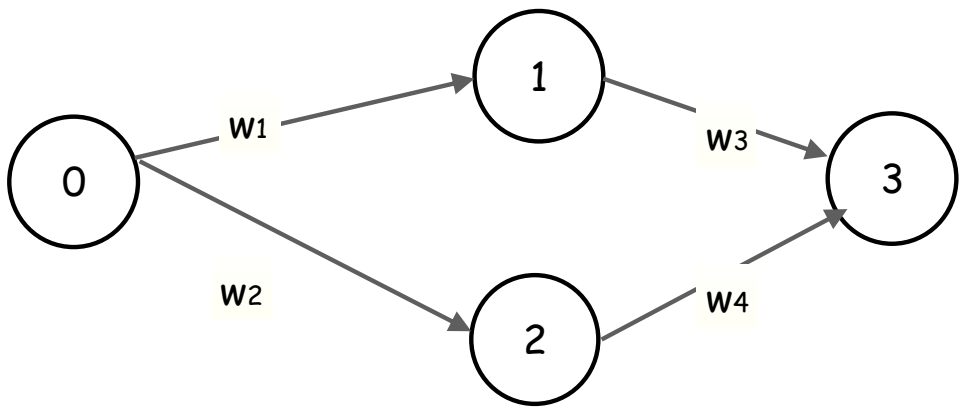


Training a neural net means adjusting its parameters (weights and biases) so it learns to map inputs to the correct outputs.

Show the neural net different example inputs and outputs

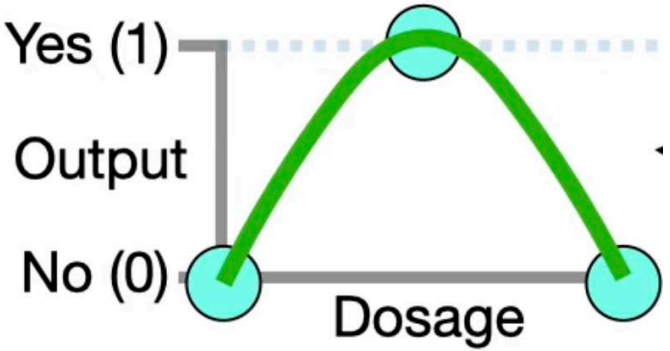
Training

Inputs	Output (y)	Predict (p)
0	0	p_1
0.5	1	p_2
1	0	p_3



 Pre-lecture video on Backpropagation in NNs: https://youtu.be/IN2XmBhILt4?si=bnDft-3T4DQ2iO9X

Backpropagation = smart feedback



Backprop: Guiding feedback that tells each weight and bias how to slightly nudge to reduce the error.

In PA03, that feedback is abstracted into a call to contribute(node, y, p)

nudge = contribution

contribute(3, y, p) return Δ

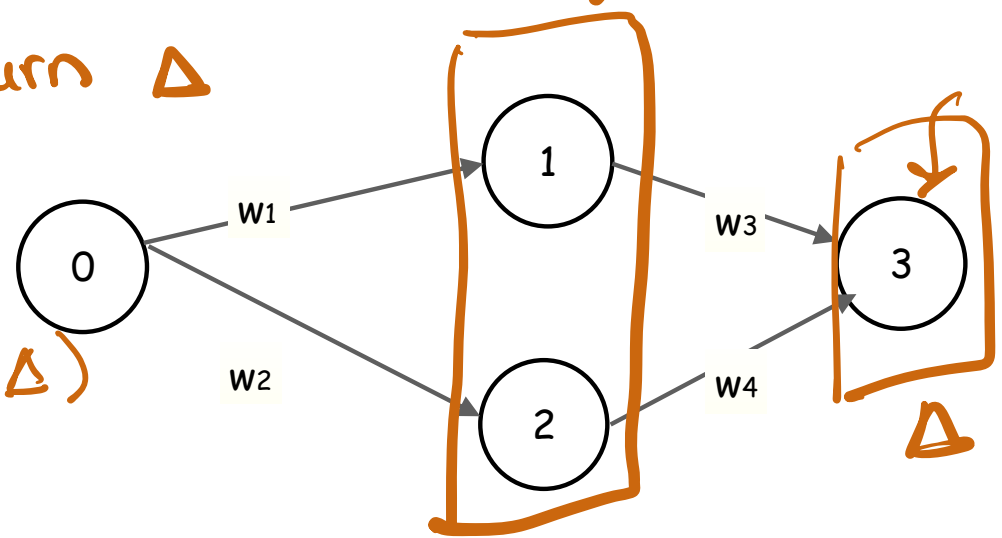
Outgoing contribution → Δ = 0

// Base case computes Δ

Apply Δ to the nodes bias → visit ContributeNode(3, Δ)

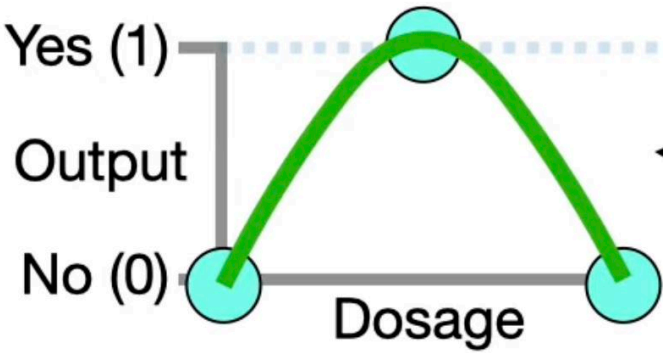
Store Δ in a lookup table

return Δ



PA03 Starter: https://github.com/ucsb-cs24-s25/STARTER-pa03/blob/main/NeuralNetwork.cpp

Backpropagation: PA03 contribute()

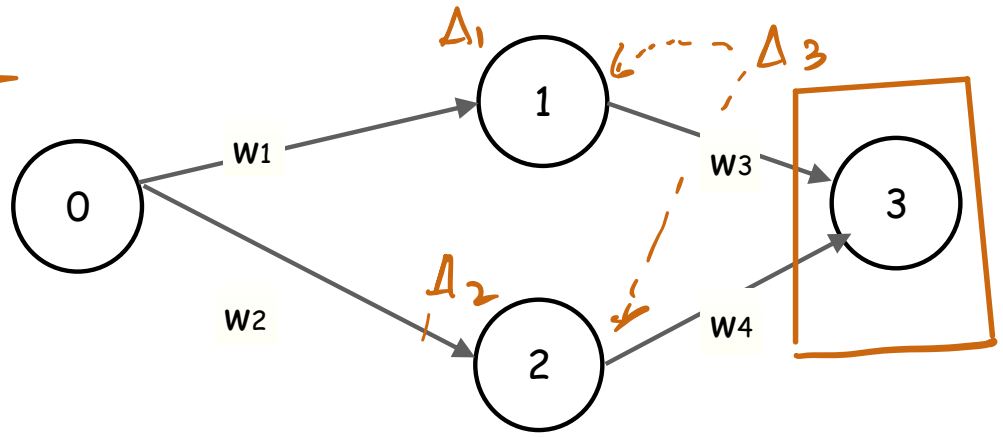


Node 3's contribution = Nudge to node 3's bias

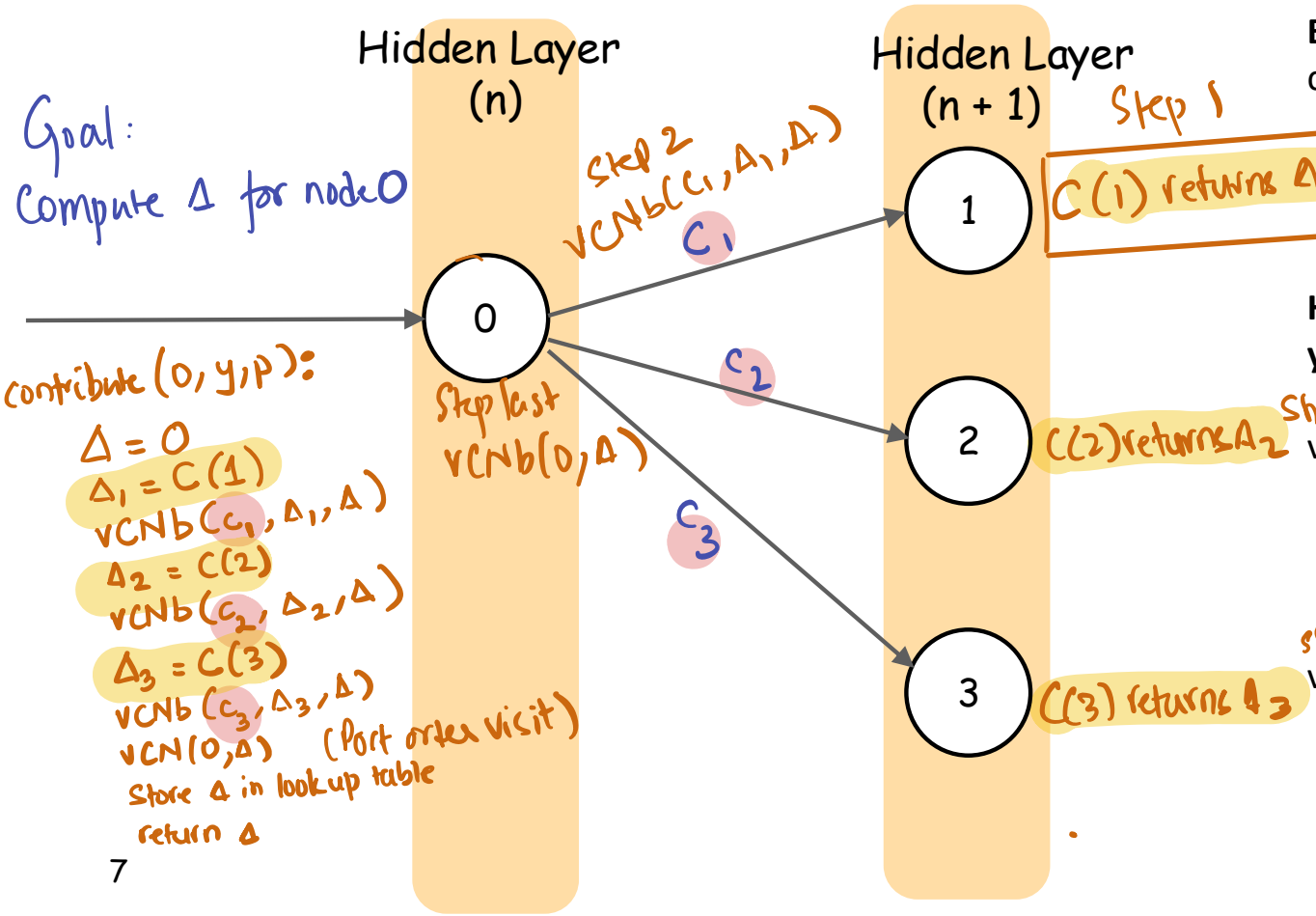
Contributions in one layer are used to compute the contributions in the previous layer

Node 3's outgoing contribution is used to compute the contributions of which nodes? Nodes 1 and

Δ_3 is used to compute Δ_1 & Δ_2



Backpropagation: Order of operations for one node



Backprop using DFS (you need to implement)

contribute(int nodeId, Short hand
const double& y, C(nodeId)
const double& p)

Helper visit functions that do the math for you:

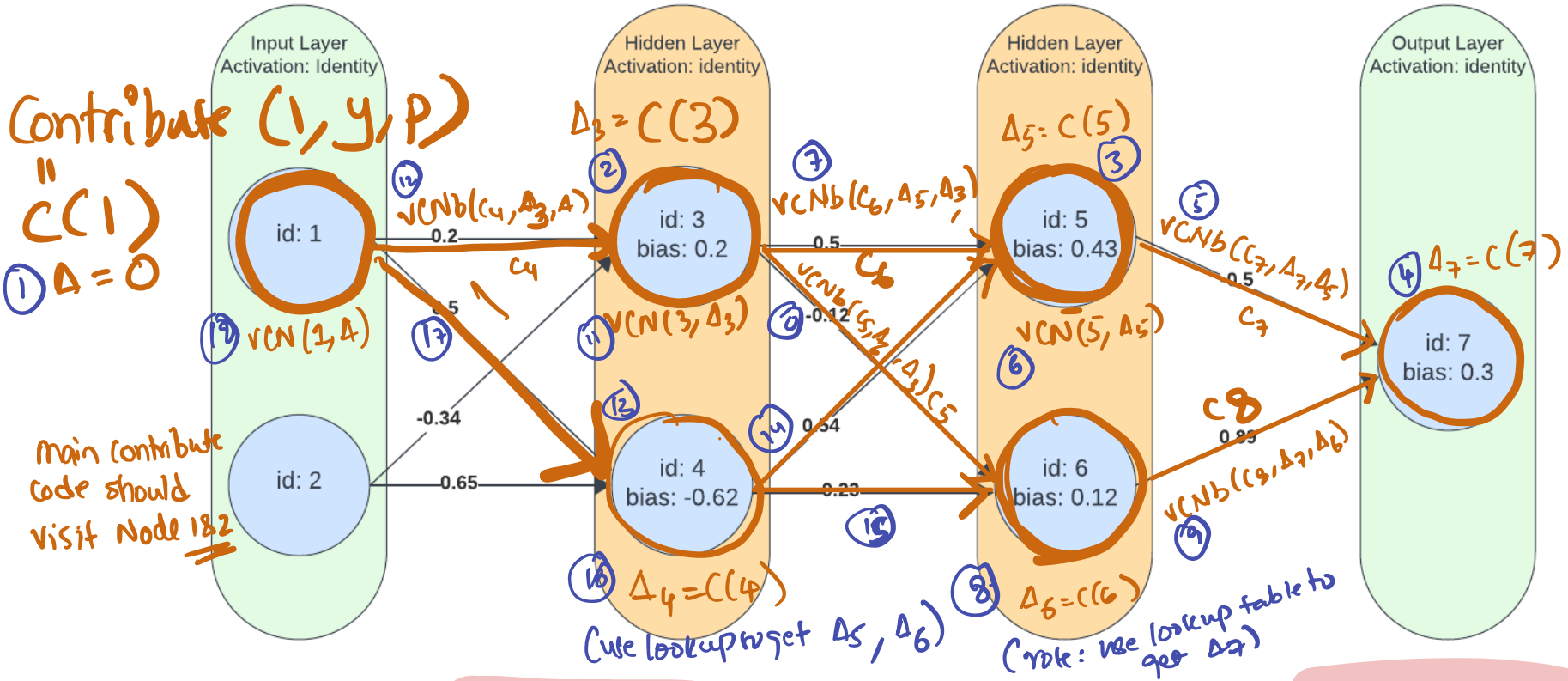
Shorthand: vCNb(connection, Δ_{in}, Δ_{out})

visitContributeNeighbor(
Connection& c,
double& incomingContribution,
double& outgoingContribution)

shorthand: vCN(nodeId, Δ)

visitContributeNode(
int vId,
double& outgoingContribution)

Activity: Trace Data Flow for Backpropagation (contribute)



Trace the order in which **contribute(...)**, **visitContributeNode (…)**, and **visitContributeNeighbor(…)** are called. For each node, write down: when it's visited, what contributions it receives, and when those are passed on to others.