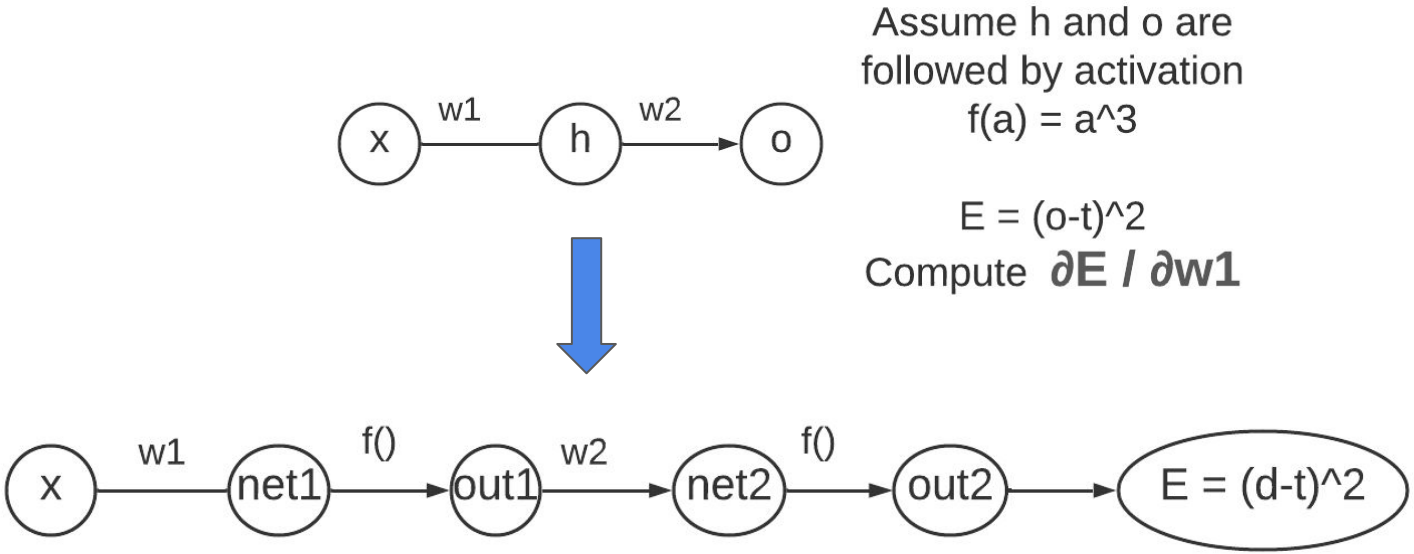
Machine Learning

Backpropagation 1

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

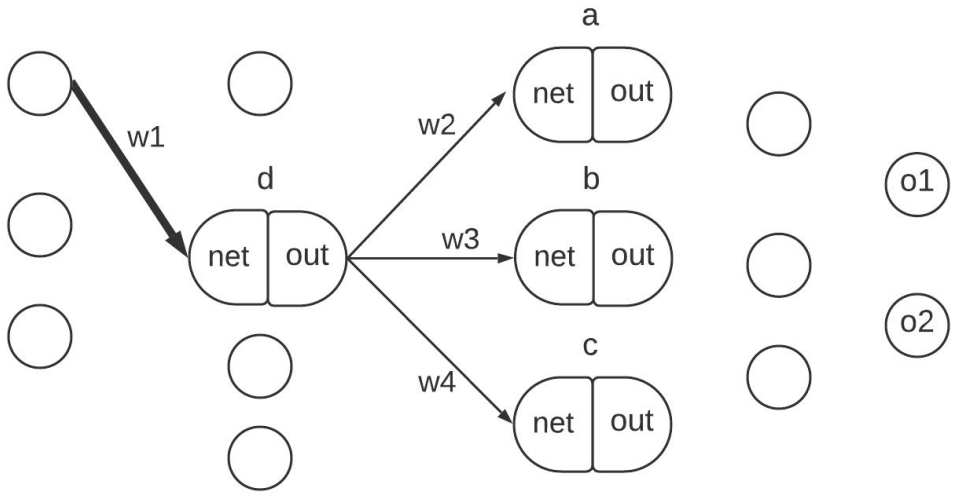
Recall a Trivial NN network



- **∂E/∂w1** = ∂E/∂out2 * ∂out2/∂net2 * ∂net2/∂out1 * ∂out1/∂net1 * ∂net1/∂w1
- If we cached the results of ∂E/∂net1
  then **∂E/∂w1** = ∂E/∂net1 * ∂net1/∂w1

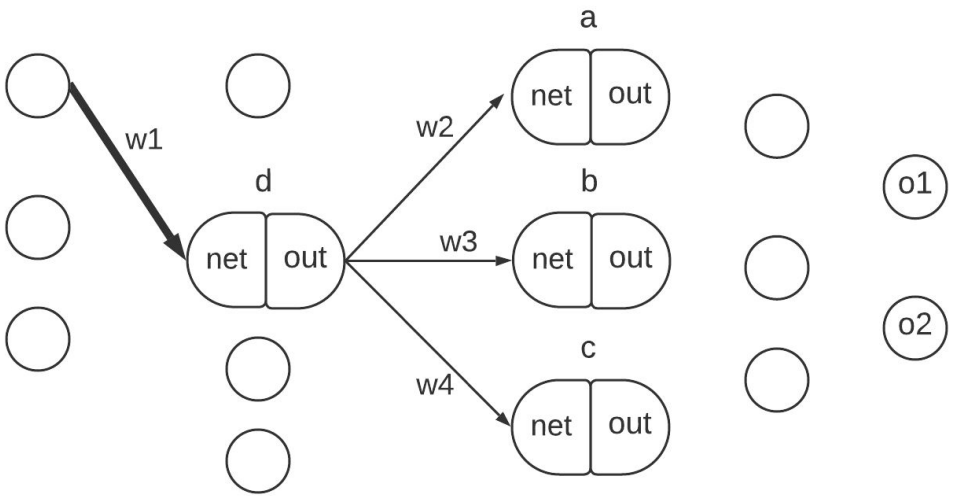
Compute ∂E/∂w1

- Assume we have 5-layers neural network
- It ends with 2 output nodes where the total error is computed using MSE
- We would like to compute ∂E/∂w1
- To do this, we consider all paths from w1 to an output nodes
  - Empty nodes in layers 1 and 2 don't matter
- w1 is linked to node **d**
- Node d is linked to nodes: **a, b, c**
- a, b, c are linked to more nodes that we should consider!



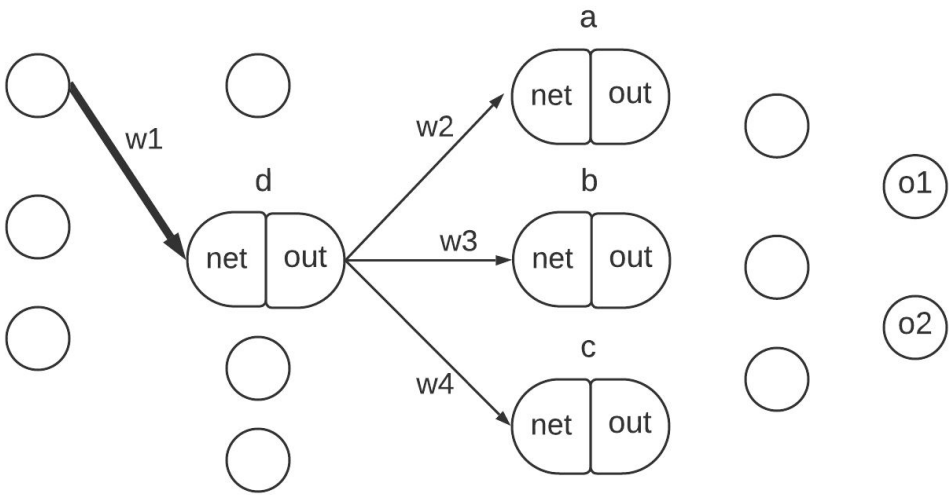
Compute ∂E/∂w1 using the Chain Rule

- ∂E/∂w1 = ∂E/∂d_net * ∂d_net/∂w1
  - ?
- ∂E/∂d_net = ∂E/∂d_out * ∂d_out/∂d_net
  - **?**



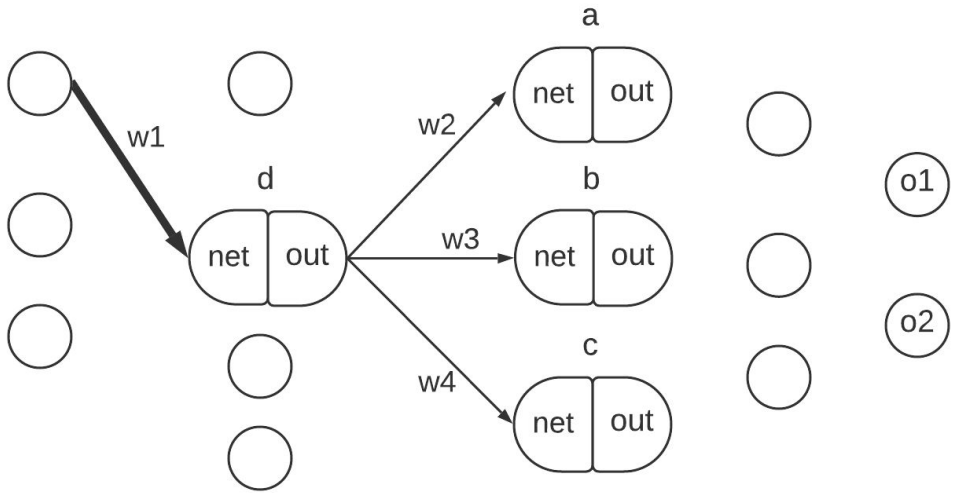
Compute ∂E/∂w1 using the Chain Rule

- ∂E/∂w1 = ∂E/∂d_net * ∂d_net/∂w1
  - ∂d_net/∂w1 = x1 [net = w1**x1** + w2x2 + w3x3]
- ∂E/∂d_net = ∂E/∂d_out * ∂d_out/∂d_net
  - ∂d_out/∂d_net is just the direct derivative of the activation function
  - If f(net) = out = net^2,
    then ∂d_out/∂d_net = 2net
  - If f(net) = out = sigmoid(net),
    then ∂d_out/∂d_net = out * (1-out)
  - **Tip: Once we compute this value, we should cache it**



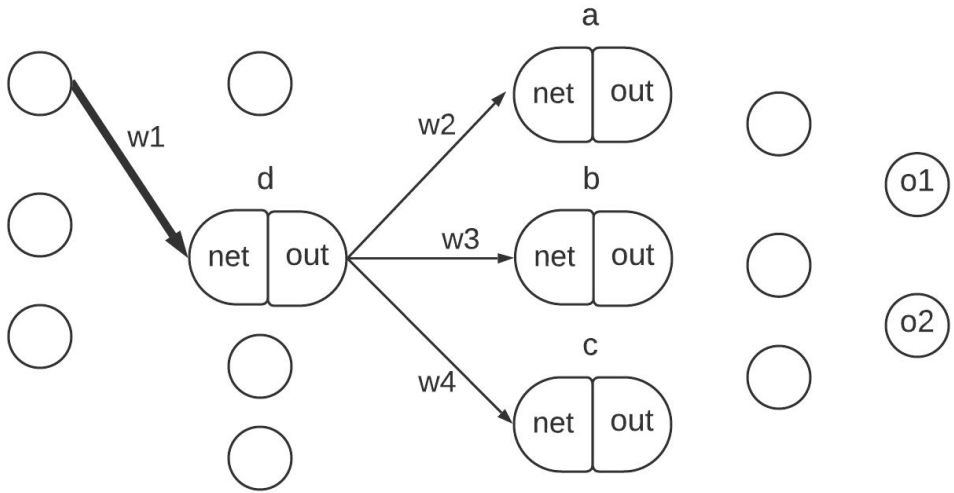
Compute ∂E/∂w1: Multivariate Chain Rule

- What about ∂E/∂d_out?
- Node d_out is connected to a_net, b_net and c_net
- We need to process all derivative paths from d_out to them
- ∂E/∂d_out =
  ∂E/∂a_net * ∂a_net/∂d_out +
  ∂E/∂b_net * ∂b_net/∂d_out +
  ∂E/∂c_net * ∂c_net/∂d_out
  - ∂a_net/∂d_out = w2
  - ∂b_net/∂d_out = w3
  - ∂c_net/∂d_out = w4



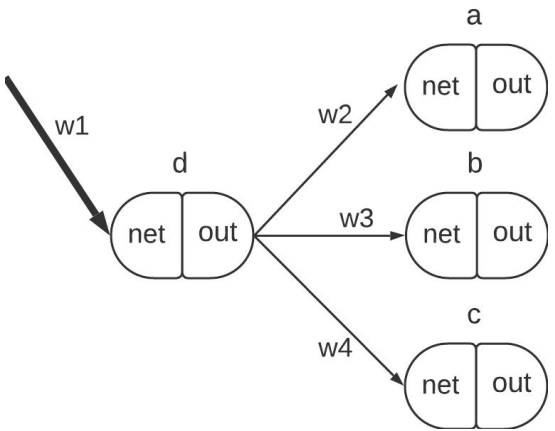
Compute ∂E/∂w1: Multivariate Chain Rule

- ∂E/∂d_out = ∂E/∂a_net * w2 + ∂E/∂a_net * w3 + ∂E/∂a_net * w4
  - Let's generalize: ∑ ∂E/∂ LayerNode_net * connection_weight
- Now we just moved from one layer to the next layer
- We can recursively keep doing that to enumerate all paths
- But what if we already **cached** these 3 values?
- Then we just use them!
- This is backpropagation!



Backpropagation derivative 3 formulas

- Assume we have interest in node d and outgoing edges of it are (weight = w[i] and connection = n[i])
- ∂E/∂d_out = ∑ w[i] x ∂E/∂n[i]_net (∂n[i]_net cached)
- ∂E/∂d_net = ∂E/∂d_out * ∂d_out/∂d_net
  - ∂d_out/∂d_net is just the direct derivative of the activation function
    - out * (1-out) for sigmoid
    - $1 - out^2$ for tanh
    - 2net for quadratic polynomial
  - Cache ∂E/∂d_net
- ∂E/∂w = ∂E/∂d_net * ∂d_net/∂w
  - ∂d_net/∂w is the input value to this node
  - Either input xi or node_out



Base Case

- We determined the general (recursive) formulas
- We need to compute the base case, which is the output layer
- Typically the activation function for regression in the output layer is identity
  - identity(x) = x
  - If you are regressing specific range, e.g. 0-1, you can use sigmoid
- Luckily, this is easy with regression
- Assume the error is MSE: $E = \frac{1}{2} (\text{output} - \text{target})^2$
  - This is vector notation. It is also the same for each output node O
- Then for some final $\partial E / \partial o\_out = \frac{1}{2} * 2 *$ **(out - target)** $* 1$

Relevant Materials

- Backpropagation: StatQuest, 3Blue1Brown (v1, v2)
- A step-by-step Backpropagation Example: Article
- The Backpropagation Algorithm
- Yes you should understand backprop

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."