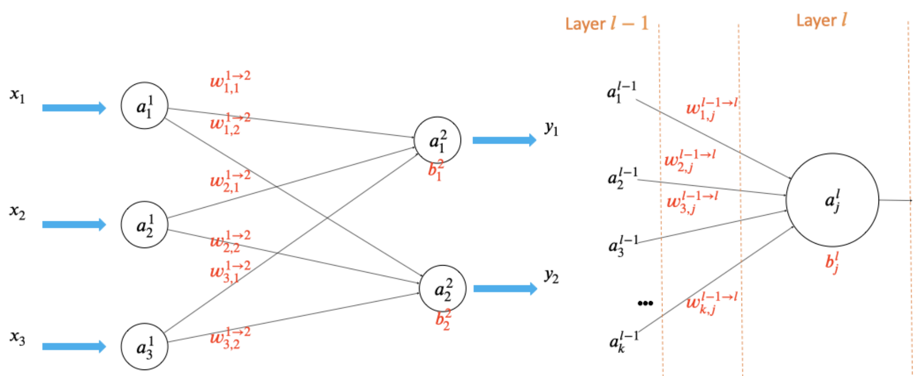
CAS CS 440
Lec 23

Supervised Learning V – Neural Networks

1. Better Models
    a. Decision Trees made strict choices
        i. Only axis-parallel cuts
    b. We make strict choices when using Decision Trees!
        i. No “internal representation” of data
        ii. This is pretty far from how humans do it!
            1. We have our own representations of things in our head!
    c. Naïve Bayes is better:
        i. Even though points x in R^n
        ii. Naïve Bayes thinks about them in $\phi(\vec{x})$ (i.e. $\phi: \mathbb{R}^n \to \mathbb{R}^m$)
            1. It happens under the hood (it comes with its own internal representation of the data)
        iii. Static “internal representation” but still has one!
            1. It is not flexible
2. Neural Networks
    a. Neural Networks form internal representations
    b. 

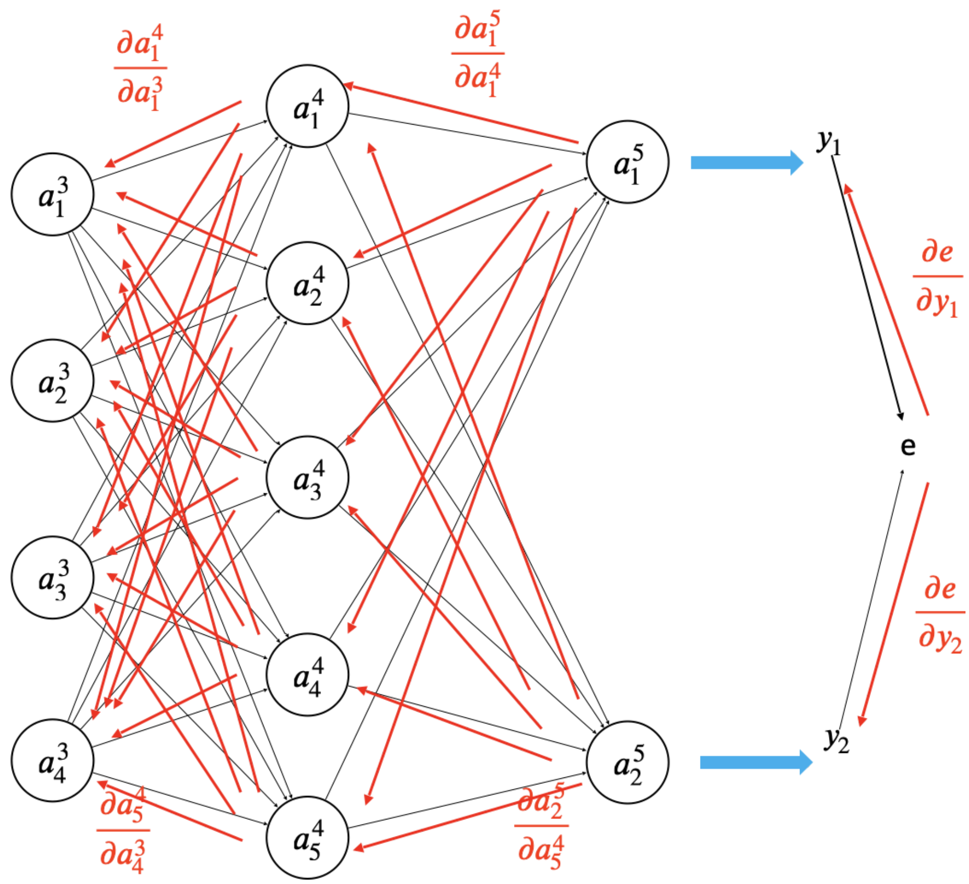
        i. First call it a different name
        ii. Transform the features into two features, in this example (second grouping has two features)
        iii. a2 and a1 are functions of the three features provided (produces two in 2 dimensional space) → expressed in different feature states
        iv. The edges and vertices have parameters, except the first set of layer(vertices)
        v. It sends scalar to the next vertex multiplied by the edge weight plus the vertex weight of next vertex → applies a non linear function, which results in aj (it is a non-linear combination of the previous layers)

c. How we calculate:

$$a_j^l = \begin{cases} f\left(b_j^l + \sum_{i=1}^{k} w_i^{l-1\rightarrow l} a_i^{l-1}\right) & l > 1 \\ x_j & otherwise \end{cases}$$

i. If f is differentiable, we can calculate derivatives

3. Derivatives Visually

a. 

i. The neural network spits out two data
ii. We combine them into a loss function (we judge the predictions)
iii. We compare it to the ground truth since we know it (supervised)
iv. Intermediary representations (the edge weights) are interchangeable, which makes it better than naive bayes

b. Passing info through the network = “forward” direction

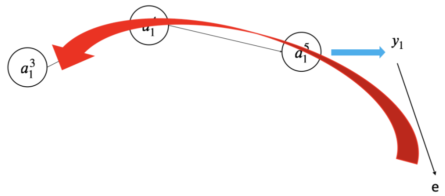
c. Passing derivative info = “backward” direction

i. Go backwards to find the error and what contributed to the error

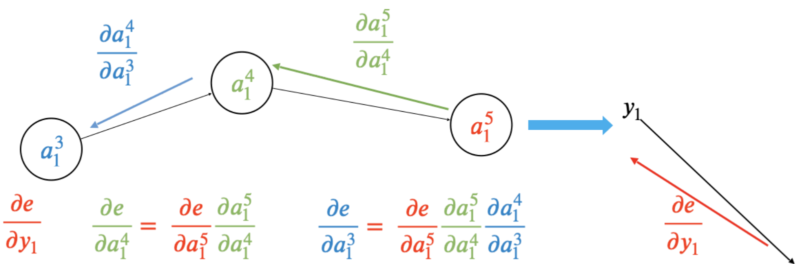
d. For every “forward” edge, a derivative corresponds to a “backward” edge

4. Derivative Calculation: Chain Rule

   a. Without Chain rule, derivatives look like this:



   b. Break up into components!



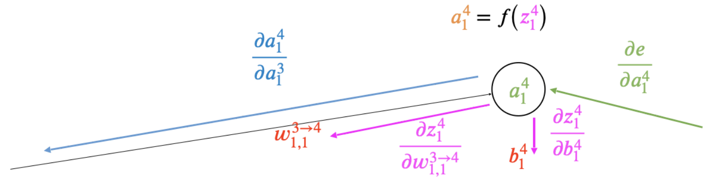
$$\frac{\partial e}{\partial a_1^5} = \frac{\partial e}{\partial y_1} \qquad \frac{\partial e}{\partial a_1^4} = \frac{\partial e}{\partial a_1^5}\frac{\partial a_1^5}{\partial a_1^4} \qquad \frac{\partial e}{\partial a_1^3} = \frac{\partial e}{\partial a_1^5}\frac{\partial a_1^5}{\partial a_1^4}\frac{\partial a_1^4}{\partial a_1^3}$$

   c. Why is this better?

      i. We know the equations for each edge
      ii. Differentiate them on paper!
      iii. Plug in the equations in code!

$$z_1^4 = b_1^4 + \sum_{i=1}^{k} w_i^{3\to4} a_i^3$$

$$a_1^4 = f(z_1^4)$$

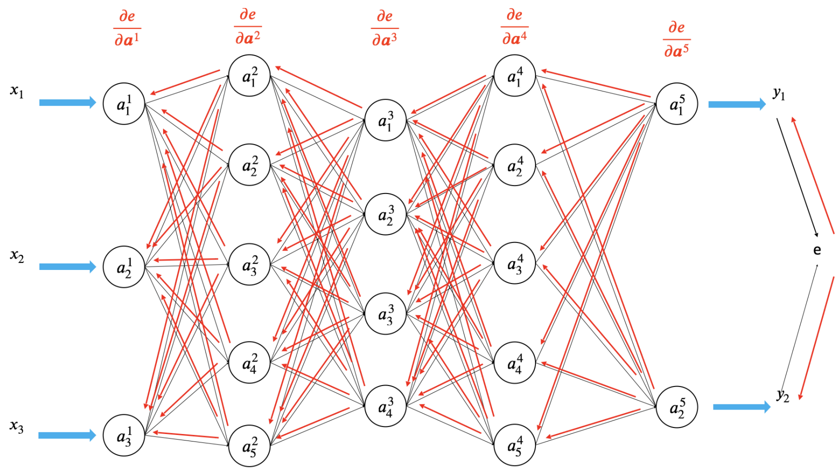


      iv.

$$\frac{\partial a_1^4}{\partial a_1^3} = \frac{\partial a_1^4}{\partial z_1^4}\frac{\partial z_1^4}{\partial a_1^3}\frac{\partial e}{\partial a_1^4}$$

$$\frac{\partial e}{\partial w_{1,1}^{3\to4}} = \frac{\partial a_1^4}{\partial z_1^3}\frac{\partial z_1^4}{\partial w_{1,1}^{3\to4}}\frac{\partial e}{\partial a_1^4}$$

$$\frac{\partial e}{\partial b_1^4} = \frac{\partial a_1^4}{\partial z_1^3}\frac{\partial z_1^4}{\partial b_1^4}\frac{\partial e}{\partial a_1^4}$$

5. Derivative Calculation: Backpropagation
    a. 

    b. Algorithm that manages calculating derivatives
        i. You can cache the values of the derivatives
6. Vector & Matrix Form of Neural Networks
    a. So far we've seen how to calculate one node at a time
        i. Used a summation, which can look weird
        ii. Can vectorize!
        iii. $z_j^l = b_1^l + \sum_{i=1}^{k} w_i^{l-1\to l} a_i^l$, $a_j^l = f\left(z_j^l\right)$ $\rightarrow$ $z_j^l = \boldsymbol{w}_1^{l-1\to l}\boldsymbol{a}^l + b_1^l$
    b. Can do it faster!
        i. Can vectorize an entire layer!
        ii. $z_1^l = \boldsymbol{w}_1^{l-1\to l}\boldsymbol{a}^l + b_1^l$, $a_j^l = f\left(z_j^l\right)$ $\rightarrow$ $\boldsymbol{z}^l = \boldsymbol{W}^{l-1\to l}\boldsymbol{a}^l + \boldsymbol{b}^l$, $\boldsymbol{a}^l = f\left(\boldsymbol{z}^l\right)$
            1. The 'a' is one example of one layer
    c. What if we want to process multiple examples at once?
        i. $\boldsymbol{z}^l = \boldsymbol{W}^{l-1\to l}\boldsymbol{a}^l + \boldsymbol{b}^l$, $\boldsymbol{a}^l = f\left(\boldsymbol{z}^l\right)$ $\rightarrow$ $\boldsymbol{Z}^l = \boldsymbol{A}^l\boldsymbol{W}^{l-1\to l} + \boldsymbol{b}^l$, $\boldsymbol{A}^l = f\left(\boldsymbol{Z}^l\right)$
            1. Can turn 'A's into matrices
            2. Rows are separate example and column is a node