CS 331
Computer Vision

11 / 09 / 2026

Instructor: Michael Eckmann

Topics

- Questions / Comments?
- Exams will be graded and given back on Thursday (if I cannot get them done by tomorrow)
- Neural Networks
  - Larger example with hidden layers and how forward prop, backprop and weight updates for mini-batch gradient descent

How a Neural Network learns

- The steps are:

Initialize weights randomly

For many iterations

For each training example

Forward Propagation to make prediction

Backward Propagation to compute the gradient

Gradient Descent to alter the weights

This describes Stochastic Gradient Descent because the weights are updated for each training example

Choices for Gradient Descent

- Gradient descent can be done to alter the weights

a. once for each training example (Stochastic Gradient Descent (SGD)) – fast and low memory but noisy,

b. or once for the entire set of training examples (Batch Gradient Descent) – slow and high memory usage for large datasets

c. or once for each mini-batch of training examples (Mini-batch Gradient Descent) – practical middle-ground

How a Neural Network learns

- The steps are:

Initialize weights randomly

For many iterations

    For each batch #

        For each training example in the batch

            Forward Propagation to make prediction

            Backward Propagation to compute the gradient

            Accumulate the gradients

        Gradient Descent to alter the weights based on average of accumulated gradients over the batch

This describes Mini-batch Gradient Descent because the weights are updated once per mini-batch

Learning in a larger network w/activation functions

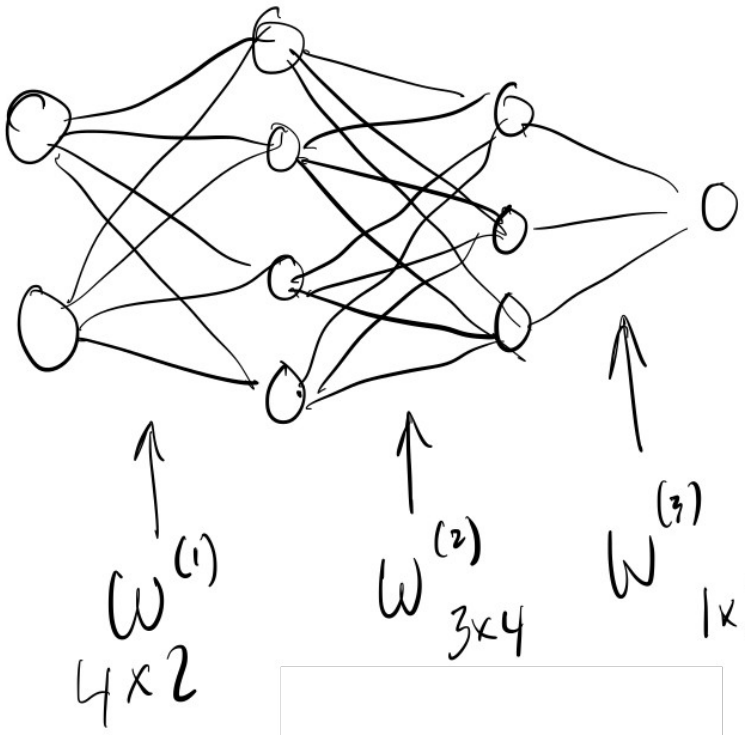
- What if we have activation functions – how do those affect the gradient?
- Show larger example

Learning in a NN

again no bias units

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Sigmoid activation in all units of layers 2, 3, 4 totalling 8 units



$z^{(2)}$ are inputs to layer 2

$a^{(2)}$ are output of layer 2

Learning in a NN

- Let's go through a more complex example of a network with 2 input units, 2 hidden layers with 4 and 3 units and 1 output unit, activation function is sigmoid in all the units of the hidden layers and the output layer.
- And we'll have an arbitrary number of training examples divided into batches.
- One thing we'll need is the derivative of the sigmoid function.
- $g(z) = 1/(1+e^{-z})$
- $g'(z) = g(z) * (1 - g(z))$

Learning in a NN

- Because we have one output unit, we are doing binary classification.
- We use the cross-entropy loss function with sigmoid activation in the output unit.
- That's the name of the loss function I introduced for logistic regression:

$$\text{Loss} = -\left[\, y \ln \hat{y} + (1 - y) \ln(1 - \hat{y}) \,\right]$$

Learning in a NN

for all batches:

$\Delta^{(1)} \leftarrow 0$, $\Delta^{(2)} \leftarrow 0$ $\Delta^{(3)} \leftarrow 0$.

For all training examples in a batch:

forward prop

$a^{(1)} = X$

$z^{(2)} = W^{(1)} @ a^{(1)}$

$a^{(2)} = g(z^{(2)})$

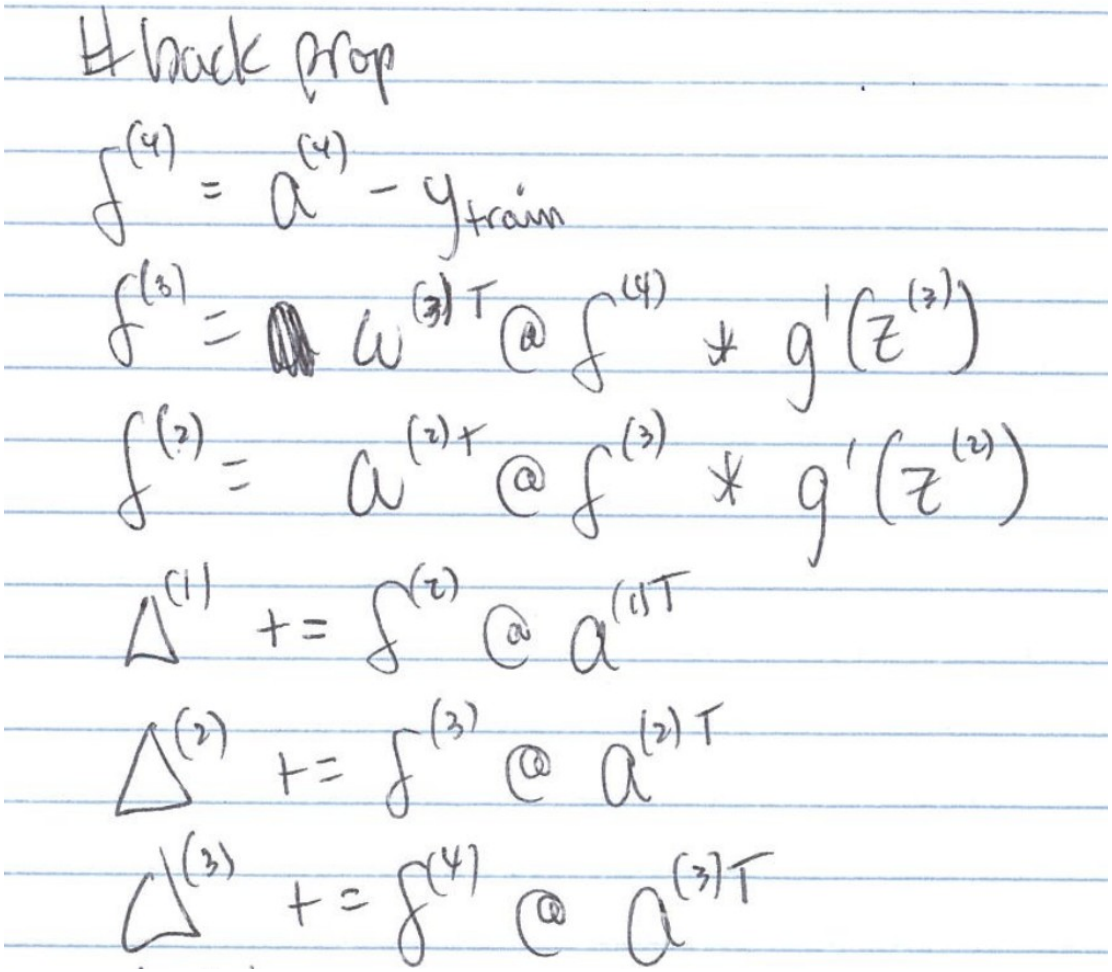
$z^{(3)} = W^{(2)} @ a^{(2)}$

$a^{(3)} = g(z^{(3)})$

$z^{(4)} = W^{(3)} @ a^{(3)}$

$a^{(4)} = g(z^{(4)})$

Learning in a NN



Learning in a NN

after a batch:

$$W^{(1)} = W^{(1)} - \alpha \left(\Delta^{(1)} / \text{batchsize} \right)$$

$$W^{(2)} = W^{(2)} - \alpha \left(\Delta^{(2)} / \text{batchsize} \right)$$

$$W^{(3)} = W^{(3)} - \alpha \left(\Delta^{(3)} / \text{batchsize} \right)$$