Samson Zhang NN from scratch

Problem

Digit classification using MNIST data

28 × 28 digits

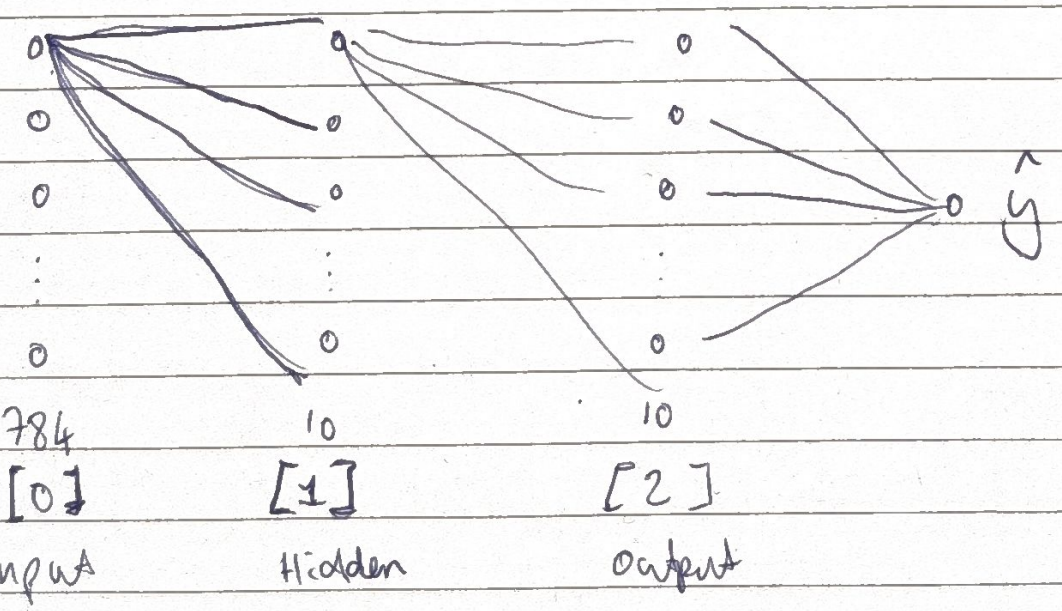
Math

28 × 28 = 784 Pixels 0 – 255
B W

$$X = \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ — x^{(3)} — \end{bmatrix}^T = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & x^{(3)} \\ | & | & | \end{bmatrix}$$

Transpose the data to be col wise
(Sample per col)

Building 2 layer NN:



(1) 1st step of NN is feedforward

input $A^{[0]} = X$ (~~Pixe~~ data) $(784 \times m)$ (Pix × Samples)

$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$ (unactivated neurons of 1st layer [1])

$10 \times M$ $\quad$ 10×784 $\quad$ $784 \times M$ $\quad$ 10×1 $\rightarrow 10 \times M$

$A^{[1]} = g(Z^{[1]}) = ReLU(Z^{[1]})$ (Activation)

$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$

$10 \times M$ $\quad$ 10×10 $\quad$ $10 \times M$ $\quad$ 10×1 $\Rightarrow 10 \times M$

$A^{[2]} = \text{softmax}(Z^{[2]})$

- vector based activation function
- output values will sum to 1
- Probabilities $\frac{e^{z_i}}{\sum e^{z_j}}$
- vec-to-vec

(2) need good weights & bias in the FF
use Backpropagation

Start w/ loss function

$dZ^{[2]} = A^{[2]} - y$ ← one hot correct label $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

(Z is not A's)

$10 \times M$ $\quad$ $10 \times M$ $\quad$ $10 \times M$

$dW^{[2]} = \frac{1}{M} dZ^{[2]} A^{[1]T}$ deriv w/ respect to $W^{[2]}$

$db^{[2]} = \frac{1}{M} \sum dZ^{[2]}$ (average of error)

$dZ^{(1)} = W^{[2]T} dZ^{[2]} .* g'(Z^{[1]})$

$dW^{[1]} = \frac{1}{M} dZ^{[1]} X^T$

$db^{[2]} = \frac{1}{M} \sum dZ^{[1]}$

} How much these two contrib to the error

(3) update Parameters

LR = hyper

$W^{[1]} := W^{[1]} - LR \cdot dW^{[1]}$

$b^{[1]} := b^{[1]} - LR \cdot db^{[1]}$

$W^{[2]} := W^{[2]} - LR \cdot dW^{[2]}$

$b^{[2]} := b^{[2]} - LR \cdot db^{[2]}$

(4) Reapeat FF, bP, update

n_times = epochs

Code

import numpy, pandas ?, Matplotlib

read data as array of rows, transpose into cols
— Sample pre col

Split data (?)
- NP.random.shuffle
- data.shape
- data[0:1000].T (way to split)
- Grab first row = now is y value
  y_ = data[0] 1st
  x_ = data[1:n] rest
  col split = data[:, 0]

need to set up and init the params to run the FF
Def ~~~ (_)
  w1 = NP.random.rand(10, 784) - 0.5 (layer start) (print this)
  or randn( )
  " ( ) · 0.01 (smaller)

  b1 = NP.random zeros((10,1)) # init 0

  w^2 = (10,10)
  b^2 = (10,1)
  return w1, b1, w2, b^2

Activ funs, ReLu, Softmax