

HW3 Assignment

- In matlab (or python if you prefer) , write forward and backprop functions for each of the following convnet building blocks:
 - ReLu, maxpool, meanpool, FC (fully connect), and softmax
- Note: I will be providing forward and backprop functions for convolution, in matlab, as examples/guides.
- Note: do not call out to any functions from matconvnet or pytorch or other convnet libraries. You are expected to write the functions from scratch yourself.
- Adhere strictly to the function names and argument order given in the homework description. Our definitions are simplified from the more general functions implemented in 3rd party libraries. Also, I will be testing your functions by calling them with test data!

Assignment

- Function $y = \text{forw_relu}(x)$

input x is an $m \times n$ matrix

$$y_{ij} = \max(0, x_{ij})$$

output y is an $m \times n$ matrix

- Function $\text{dzdx} = \text{back_relu}(x, y, \text{dzdy})$

input x is an $m \times n$ matrix

input y is an $m \times n$ matrix (output from forward pass)

input dzdy is an $m \times n$ matrix of dz/dy_{ij} values

output dzdx is an $m \times n$ matrix of dz/dx_{ij} values

Assignment

- Function $y = \text{forw_maxpool}(x)$
input x is an $2m \times 2n$ matrix (that is, you may assume that it has an even number of rows and cols)
output y is an $m \times n$ matrix
- Function $\text{dzdx} = \text{back_maxpool}(x, y, \text{dzdy})$
input x is an $2m \times 2n$ matrix
input y is an $m \times n$ matrix (output from forward pass)
input dzdy is an $m \times n$ matrix of dz/dy_{ij} values
output dzdx is an $2m \times 2n$ matrix of dz/dx_{ij} values

Assignment

- Function $y = \text{forw_meanpool}(x)$
input x is an $2m \times 2n$ matrix (that is, you may assume that it has an even number of rows and cols)
output y is an $m \times n$ matrix
- Function $\text{dzdx} = \text{back_meanpool}(x, y, \text{dzdy})$
input x is an $2m \times 2n$ matrix
input y is an $m \times n$ matrix (output from forward pass)
input dzdy is an $m \times n$ matrix of dz/dy_{ij} values
output dzdx is an $2m \times 2n$ matrix of dz/dx_{ij} values

Assignment

- Function `y = forw_fc(x,w,b)` %fully connect
input x is an mxn matrix
input w is an mxn matrix of weights
b is a scalar bias value
output y is a scalar value

$$y = \sum_i \sum_j w_{ij} x_{ij} + b$$

- Function `[dzdx,dzdw,dzdb] = back_fc(x,w,b,y,dzdy)`
input x is an mxn matrix
input w is an mxn matrix of weights
b is a scalar bias value
input y is a scalar value (output from forward pass)
input dzdy is a scalar value dx/dy
output dzdx is an mxn matrix of dz/dx_{ij} values
output dzdw is an mxn matrix of dz/dw_{ij} values
output dzdb is a value dz/db

Assignment

- Function `y = forw_softmax(x)`

input `x` is `mx1` vector

output `y` is an `mx1` vector

$$y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

- Function `dzdx = back_softmax (x,y,dzdy)`

input `x` is an `mx1` vector

input `y` is an `mx1` vector (output from forward pass)

input `dzdy` is an `mx1` vector of dz/dy_i values

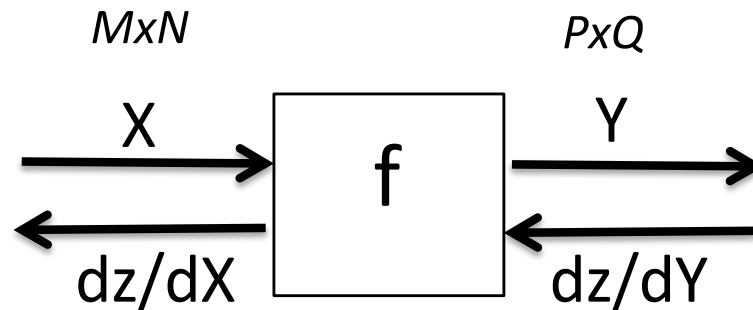
output `dzdx` is an `mx1` vector of dz/dx_i values

Assignment Part 2

- Test that your backprop functions work by estimating the derivatives numerically using `forw_xyz`, and comparing those to the values returned from your `back_xyz` function (see next page for more details).
- I won't be trying to run this section of code, just show me that it works by attaching the output of running it.

Assignment Part 2

- More specifically, for numerical derivative testing, refer to the picture below, from our backprop lecture.



- To compute dz/dX numerically:
 - make up some values for forward pass input X
 - also make up some values for backprop input dz/dY
 - compute output values Y using the `forw_xyz` function you wrote
 - we now want to compute dz/dX numerically
 - for each input value X_{ij} , make a small change $X_{ij} + \text{eps}$, run your `forw_xyz` function to see how it changes Y values, estimate the numerical derivatives dY/dX_{ij} , then combine appropriately with your made up dz/dY values to compute dz/dX_{ij} . Repeat for all X_{ij} .
 - compare the numerically estimated dz/dX values with the analytic ones you compute in the `back_xyz` function you wrote. They should be the same (or very similar)
 - Refer to `testbackconv.m` on our assignment page for an example estimating numeric derivs for dz/dX , dz/dw , and dz/db of the convolution function.

What to Hand In

In the assignment dropbox on Canvas, upload a single zip file that contains your forward, backward, and numerical derivative test routines, the output of running the derivative tests, and an optional readme file if you need to tell me something.

What I'm looking for when grading

- forw and back routines submitted for relu, maxpool, meanpool, FC, and softmax
- each routine works correctly when I test it
- code is commented and has good formatting
- did not just call some external library routine
- routines for numerical gradient checking are submitted. These could be in one big file if you want, I'm not going to be calling them.
- showing output of running the numerical gradient tests