HW1 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: don't call out to any functions from matconvnet or pytorch or other convnet libraries. DON'T USE THE DEEP LEARNING TOOLBOX IN MATLAB EITHER!!!! You are expected to write the functions from scratch yourself. Helper functions from the image processing toolbox are OK.
- Adhere strictly to the function names and argument order given in the homework description. I will be testing your functions with a script that calls them on test data and verifies the output produced! Note: Our definitions are simplified from the more general functions implemented in 3rd party libraries.

Assignment

- Function y = forw_relu(x)

  input x is an mxn matrix
  output y is an mxn matrix

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

- Function dzdx = back_relu(x,y,dzdy)

  input x is an mxn matrix
  input y is an mxn matrix (output from forward pass)
  input dzdy is an mxn matrix of dz/dy_{ij} values
  output dzdx is an mxn matrix of dz/dx_{ij} values

Assignment

- Function y = forw_maxpool(x)

  input x is an 2mx2n matrix (that is, you may assume that it has an even number of rows and cols)
  output y is an mxn matrix

- Function dzdx = back_maxpool(x,y,dzdy)

  input x is an 2mx2n matrix
  input y is an mxn matrix (output from forward pass)
  input dzdy is an mxn matrix of dz/dy_{ij} values
  output dzdx is an 2mx2n matrix of dz/dx_{ij} values

Assignment

- Function y = forw_meanpool(x)

  input x is an 2mx2n matrix (that is, you may assume that it has an even number of rows and cols)
  output y is an mxn matrix

- Function dzdx = back_meanpool(x,y,dzdy)

  input x is an 2mx2n matrix
  input y is an mxn matrix (output from forward pass)
  input dzdy is an mxn matrix of dz/dy_{ij} values
  output dzdx is an 2mx2n 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

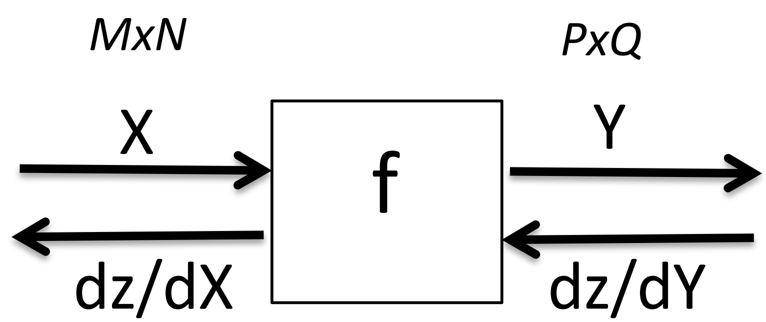
Assignment Part 2

- 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 Xij, make a small change Xij+eps, run your forw_xyz function to see how it changes Y values, estimate the numerical derivatives dY/dXij , then combine appropriately with your made up dz/dY values to compute dz/dXij. Repeat for all Xij.
  - 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