7/20

Resources

- Matrix & Vector Multiplication
    - video
- Neural Networks

Torchvision library

- You can use it to call/access particular datasets

We can load images using the dataset class and index into it, to later feed into the neural networks

- `X, y = train_dataset[0]`

Linear versus non linear problems

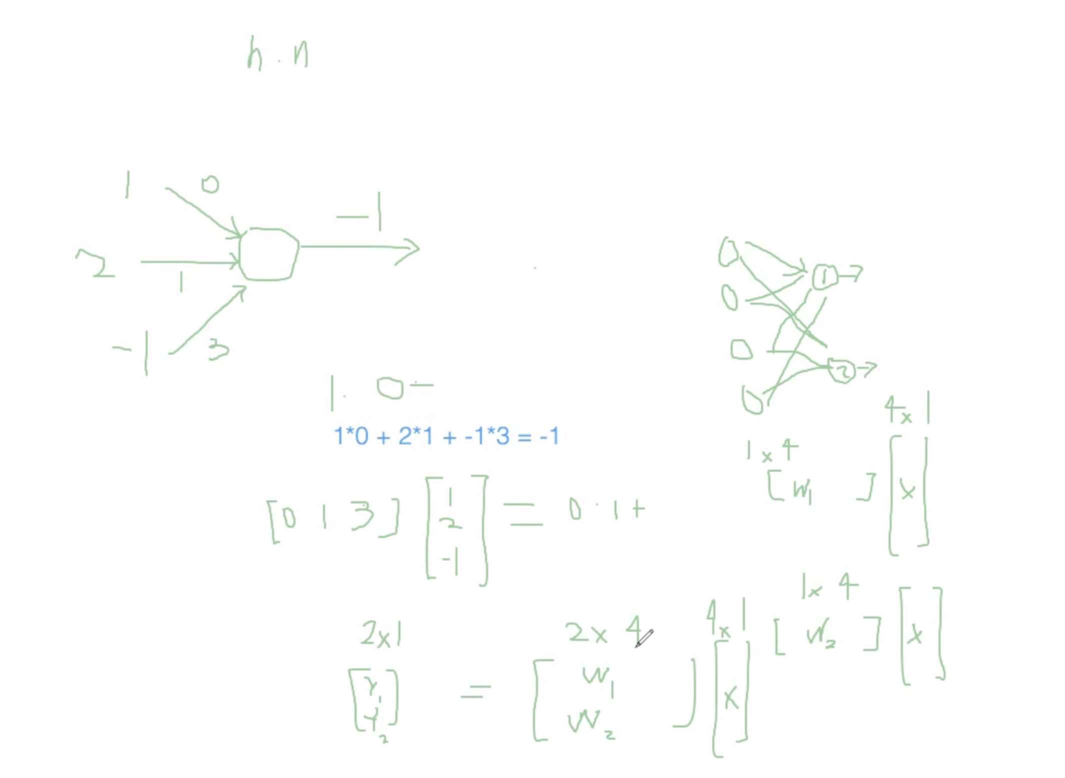
- Are convolutions and max pool linear operations?

Terminology

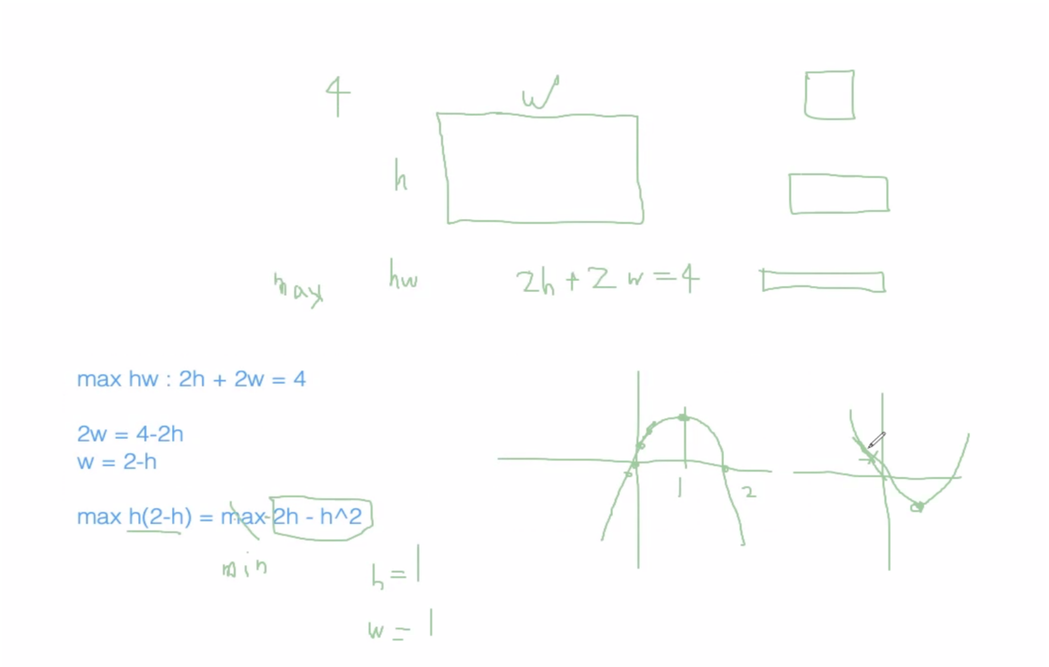
- Convolution layer
    - Goes over each region in the image, and *extracts information*
    - Define function parameters to create a convolution filter
- Max pooling layer
    - Once you extract features using the convolution layer, it may be too large
    - We need max pooling to *reduce the image dimensions* and train the network faster
        - Going in between layers will require a lot of matrix multiplication
        - For example, in a 3x3 region, we can pick the maximum number to represent that region. Doing this within every region will reduce the multiplications that need to be done
- Epoch
    - One complete iteration of the entire dataset
    - A NN will need to go over a dataset multiple times to train
- Data normalization
    - What is the range of pixel intensities if you load an image into a numpy array? What are the lower and upper bound values that a pixel can take?
        - Range -> 0 to 255
        - (R, G, B)
    - The NN could find it hard to understand the pixel range, the converge (going from start to finish) of the NN will take a long time
    - If we optimize the pixel values, we can make it easier for the NN to get to the final solution
        - A data normalization would be dividing all of the pixel values by 255, then we have a new range of pixel values, 0 to 1, to work with

        - Normalized data is better than the raw data

Neural Networks

- Input -> Hidden -> Output
- We take the weights of the previous layer, to create the next layer
- Example
    - Input -> 2 images x 3 dimensions
    - We want the hidden layer to be 2 x 4
    - What should be the weights of the linear layer to get the hidden layer?
        - M x n
- whiteboard demo from class for matrix & vector multiplication
    - 

Optimization Problem



$\max hw : 2h + 2w = 4$

$2w = 4-2h$
$w = 2-h$

$\max \underline{h(2-h)} = \cancel{\max} \boxed{2h - h^2}$

-
- With neural networks, instead of having just one point, you have a million or more points that correspond to the weights
- With one point or one dimension, you can use calculus to find the minimum
    - With multiple dimensionsions, you can use gradient optimization

Homework

- Check ELMS!
- Define the two feature extractors in the ipython notebook
- Submit through ELMS

7/21

- Agenda/Quick Recap
    - Homework review (Manipulating the shape of the tensor using the view function)

Types of Layers in a standard neural network

- Linear
- Conv
- Relu
- Maxpool
- Batchnorm

NN Diagram explanation

- Circles correspond to inputs & outputs
- Lines correspond to linear layers

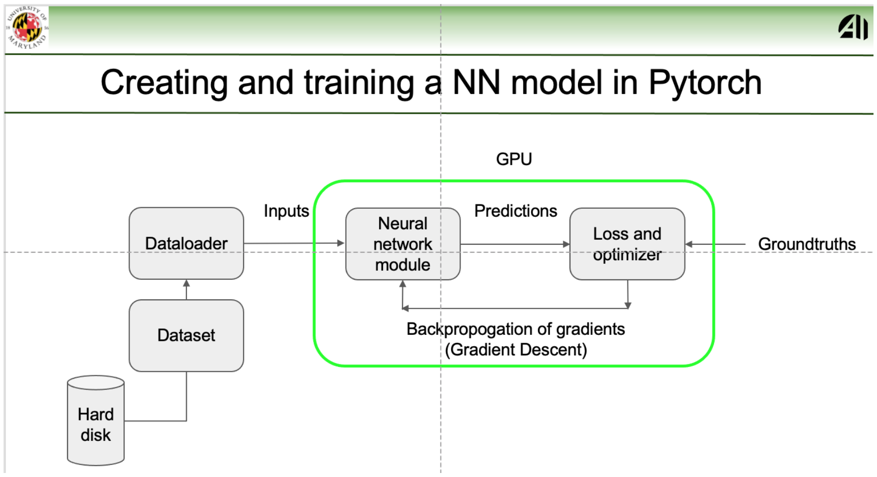
    - In pytorch, if you tell the program you have two inputs and 3 outputs, it will create all of the lines/layers automatically between each input and output
        - Layer 1 = nn.Linear(2,3)
        - Layer 2 = nn.Linear(3,3)
        - Layer 3 = nn.Linear(3,1)

7/23

- Reminders
    - Next friday we will be doing project presentations
    - Goal: complete project by next wednesday & prepare presentation thursday

Review of Creating and Training a NN Model in Pytorch (Diagram)

- *Dataset class*
    - Reads image data from the hard disk, prepares the training data
- *Dataloader*
    - Gathers a set of images/data points from the dataset class and prepares a mini batch that is given to the neural network model
    - Any data received from the dataset class is concatenated into a new dimension which becomes the mini batch
    - Each *iteration* corresponds to the forward pass and backward pass of one mini batch
        - Forward pass -> predictions
        - Backward pass -> updates the neural network
    - Epoch -> certain number of iterations per training, epoch is not the same as an iteration
    - Ensures that it does not pick images that were already picked during iterations
- *Neural Network model*
    - Takes mini batch as input from the data loader
- *Loss and Optimizer*
    - Optimizer updates the neural network
    - Determines how far the predictions are from the groundtruth (how accurate is the neural network)
    - If loss calculation is 0, then the predictions are perfect
    - We can decrease the loss, by increasing the number of epochs, or in other words, showing the same data set to the neural network many times
        - Because the more data we show, the better the neural network will be trained
        - If we do this, we don't want to do the backward pass, because we don't want to update the neural network in this case. We just want to calculate and test predictions

- 

  Creating and training a NN model in Pytorch

  GPU

  Dataloader

  Inputs

  Neural network module

  Predictions

  Loss and optimizer

  Groundtruths

  Dataset

  Backpropogation of gradients (Gradient Descent)

  Hard disk
- EXAMPLE:
    - nn.Conv2d(in_channels = 3, out_channels = 6, kernel = 3, stride = 1, padding = 0)
        - Same as nn.Conv2d(3, 6, 3, 1, 0) and nn.Conv2d(3, 6, 3)
    - nn.ReLU() does not change the shape, only converts negative values to 0, gets it all in order
    - nn.MaxPool2d(Kernel size = 2) Makes sure dimensions are reduced by half
    - nn.Conv2d(in_channels = 6, out_channels = 12, kernel_size = 3, stride=1, padding=0)
        - Output w (2, 3, 64, 64) : torch.size([2, 12, 29, 29]) size is 29
    - nn.Relu()
    - nn.MaxPool2d(kernel_size = 2)
        - Output: torch.size([2,12,14,14]) size is 14
    - nn.conv2d(in_channels = 12, out_channels = 24, kernel_size=3, stride=1, padding=0)
    - nn.ReLU()
        - Output: torch.Size([2,24,12,12]) size is 12
    - nn.MaxPool2d(kernel_size=2)
        - Output: torch.Size([2,24,6,6])
        - After flattening, you got 864 or torch.Size([2, 864])
        - After a classifier, the output.shape is torch.size([2, 1024])
            - Classifier has nn.Linear(in_features = 864, out_features= 1064)
- H_out = 1 + (H_in + 2Padding - Dilation (Kernel - 1) - 1) / Stride
-

Terminology Cont.

- SoftMax
    - Assigns decimal probabilities to classes
    - Function definition

Homework

- Will be posted in class ELMS page, make sure to work on it, and email Koutilya & Daniel if you have any questions
- It is important to work on this over the weekend so you have enough time to complete the final project

Section	Who
Overview	Ezekiel
Dataset and Dataloader	Grace
Feature Extraction	Nathaniel
Loss and Optimization	Amelia
Accuracy/Results	Lucas
World Applications	Michael
Google Site	Michael

7/28