EE5179 : Deep Learning for Imaging

Programming Assignment 3: Autoencoders

Due Date: October 23, 2024, 11:59 pm

Instructions

1. Program in python for this assignment.
2. Post any doubts you have on moodle forum. This will be helpful to your peers as well. (Note: The forum will be closed on October 22, 2024, 11:59 pm)
3. Submit the codes (the actual notebook and a PDF version) and your assignment report in a zip file titled PA2_RollNumber.zip in the submission link provided on moodle.
4. Late submissions will be evaluated for reduced marks and for each day after the deadline we will reduce the weightage by 5%.

Preliminaries

1. It is recommended to use Google Colab (as in the tutorials) or Jupyter/iPython notebooks for the assignment. The notebook format is very convenient to work (and evaluate!) with and additionally, Colab provides GPU access as well.
2. The dataset can be downloaded from here or you can make use of the inbuilt dataset from Pytorch.

---

1 Comparing PCA and Autoencoders

In this part, we will compare Autoencoder (AE) with principal component analysis (PCA)

- Load MNIST dataset. Do PCA on it and take only the first 30 eigenvalues with their corresponding eigenvectors. Now, project the data onto these eigenvectors and reconstruct them from 30 dimensional representation. Next, train an AE with the following specifications:

- **Encoder**: fc(512)-fc(256)-fc(128)-fc(30)
- **Decoder**: fc(128)-fc(256)-fc(784)

Use ReLU as the activation function. Compare the Reconstruction Accuracy with PCA and comment.

2 Standard Autoencoder

Design a under-complete AE with just one hidden layer that acts as dimensionality reduction for MNIST dataset. Keep the dimension of hidden layer (x) as a variable and train the network for different hidden unit dimensions. Check the reconstruction. Let x = [64, 128, 256] and do the following:

- Test the network on any one of your testset images and compare the quality of reconstruction for different values of x.
- What kind of reconstructions do you get when you pass non-digit images or random noise images as input to the auto-encoder?
- The weight vectors associated with each hidden node is called a filter. Try to visualize the learned filters of the standard AE as images. Does their structure make any sense to you? What do you think they represent?

3 Sparse Autoencoders

Design an over-complete AE with sparsity regularization (Check L1Penalty in torch). We impose sparsity by adding L1 penalty on the hidden layer activation. L1 penalty is nothing but L1 norm on the output of hidden layer. Here, the parameter controls the degree of sparsity (the one you pass to L1 Penalty function while defining the model). Higher the value, more sparser the activations are. You can vary the value of this parameter and observe the change in filter visualizations. Also, if the sparsity is too much, it could lead to bad reconstruction error.

- Compare the average hidden layer activations of the Sparse AE with that of the Standard AE (in the above question). What difference do you observe between the two?
- Now, try to visualize the learned filters of this Sparse AE as images. What difference do you observe in the structure of these filters from the ones you learned using the Standard AE?

4 Denoising Autoencoders

Design a denoising AE with just one hidden unit.

- What happens when you pass images corrupted with noise to the previously trained Standard AEs?
- Change the noise level (typical values: 0.3, 0.5, 0.8, 0.9) and repeat the above experiments. What kind of variations do you observe in the results.
- Visualize the learned filters for Denoising AEs. Compare it with that of Standard AEs. What difference do you observe between them?

5 Manifold Learning

As discussed in class, AE learns the manifold on the data. In this experiment, we will try to represent the MNIST data with an AE and try to check what the representation space is learning.

- Take an input data from MNIST. Try moving in random directions (i.e add random noise to it). This implies in a 784-dimensional space, if you randomly sample or randomly move in different direction you end up not getting a valid digit. Why is it so?
- Now train an AE with the following configuration: input-fc(64)-fc(8)-fc(64)-fc(784)
- After the network converges, pass an image from the test set. Add noise to the representation and try to reconstruct the data. What do you observe and why? Relate with manifold learning.

6 Convolutional Autoencoders

As discussed in class, AE can also be implemented as fully convolutional networks with the decoder consisting of various upsampling operations including unpooling and deconvolution.

- Now, train a Convolutional AE for the MNIST data with 3 convolutional layers for encoder and the decoder being the mirror of encoder (i.e a total of 7 convolutional layers for AE with the final convolutional layer mapping to the output). Architecture for the encoder part:
  - Input - Conv1 (8 3x3 filters with stride 1) - 2x2 Maxpooling - Conv2 (16 3x3 filters with stride 1) - 2x2 Maxpooling - Conv3 (16 3x3 filters with stride 1) - 2x2 Maxpooling - (Encoded Representation)
- This needs to followed by the decoder network. Experiment with unpooling + deconvolution type of upsampling (check maxUnpool2d and convTranspose2d in Pytorch). Report on reconstruction error and convergence. Also visualize the decoder weights. What do you observe?

–end–