EE6132

Programming Assignment 3: Autoencoders

Due Date: 18th November, 2021

Instructions

1. You may do the assignment in MATLAB, Python.
2. For any questions, please schedule a time with TAs before the deadline according to their convenience. Please use the Moodle discussion thread for posting your doubts and check it to see if your doubt has already been resolved before approaching the TAs.
3. Submit a single zip file named in the following format: ***PA3_RollNumber.zip***. It should contain your report and a separate folder containing all your codes.
4. **Do not** put codes in your report. Your report should contain only the necessary plots, observations and inferences. If you are putting screenshots in your report, make sure the texts are large enough to be readable and the plots are properly titled.
5. Do not copy any part from any source including your friends, seniors or the internet.
6. Late submissions will be evaluated for reduced marks and for each day after the deadline we will reduce the weightage by 5%.

Dataset: You can make use of the binary files that you downloaded for the 1st assignment. You can also make use of the any other library to load the data.

1 Comparing PCA and Autoencoders

In this part, we will compare Autoencoders with 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 . Next , train an autoencoders 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.

2 Experimenting with hidden units of varying sizes

Train a standard auto-encoder with the following specifications:

- fc(x)-fc(784)

Here, x is the size of the hidden unit. Using x = [64, 128, 256], do the followings:

- Test the network on any one of your testset images and compare the quality of reconstruction for different values of x.
- What output do you get if you pass a non-digit image or random noise images through the network?

3 Sparse Autoencoders

Design an over-complete autoencoder 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 AutoEncoder with that of the Standard AutoEncoder (in the above question). What difference do you observe between the two?
- Now, try to visualize the learned filters of this Sparse AutoEncoder as images. What difference do you observe in the structure of these filters from the ones you learned using the Standard AutoEncoder?

4 Denoising Autoencoders

Design a denoising autoencoder with just one hidden unit.

- What happens when you pass images corrupted with noise to the previously trained Standard Autoencoders ?
- 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 Autoencoderss. Compare it with that of Standard Autoencoders. 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 and AE with the following configuration
  - fc(64)-fc(8)-fc(64)-fc(784)

  After the network converges, pass an image fromthe 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 upsampling operations of any of these variants - i) Unpooling or ii) Unpooling + Deconvolution or iii) Deconvolution.

- 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 all the three types of upsampling. Refer to this report for good description of convolutional arithmetic in deep learning. Keeping all the other parameters the same, report on reconstruction error and convergence with the different types of upsampling. Also visualize the decoder weights for the three cases. What do you observe?