Exercise III

AMTH/CPSC 663b - Spring semester 2019

Due: Thursday, March 28, 2019 - 4:00 PM

Compress your solutions into a single zip file titled `<lastname and initials>_ assignment3.zip`, e.g. for a student named Tom Marvolo Riddle, `riddletm_assignment3.zip`. Include a single PDF titled `<lastname and initials>_assignment3.pdf` and any Python scripts specified. Any requested plots should be sufficiently labeled for full points. Your homework should be submitted to Canvas before Thursday, March 28, 2019 at 4:00 PM.

Programming assignments should use built-in functions in Python and TensorFlow; In general, you may use the `scipy` stack [1]; however, exercises are designed to emphasize the nuances of machine learning and deep learning algorithms - if a function exists that trivially solves an entire problem, please consult with the TA before using it.

Problem 1 TensorFlow Practice (5 pts)

It is suggested that you go through the TensorFlow Get Started if you haven't done so.
Check out with this link https://www.tensorflow.org/get_started/

- "Getting Started with TensorFlow"
- "Getting Started for ML Beginners"

Check out with this link https://www.tensorflow.org/versions/r1.1/get_started/

- "MNIST for ML Beginners"
- "Deep MNIST for Experts"

Optionally, you may want to read through on the same page
"TensorBoard: Visualizing Learning" and "TensorBoard: Graph Visualization"
Also, you may find it helpful to read through the sample codes.

1. In `tflecture1_2.py`, make sure you understand what `tf.Variable, tf.Session()` do. Explain what `init = tf.global_variables_initializer() and init.run()` do, and how this code differs from `tflecture1_1.py`.

2. Compare `tflecture1_3.py` with `tflecture1_4.py`. What do line 32, line 78-87, line 94-102 in `tflecture1_3.py` do? What does line 57 in `tflecture1_4.py` do? Briefly Explain. Make sure you have installed `sklearn` before testing with the code.
   -Line 32 in `tflecture1_3.py`
   `theta = tf.matmul(tf.matmul(tf.matrix_inverse(tf.matmul(XT, X)),XT),y)`
   -Line 57 in `tflecture1_4.py`.
   `gradients = tf.gradients(mse,[theta])[0]`

3. Based on what `tflecture1_4.py` does, implement the `prob1.py` TO DO part using the TensorFlow optimizer. Is the output different from `tflecture1_4.py`? Save your code as `prob1.py`. Hint: You might only need two lines to set up the optimizer and use the optimizer to minimize the MSE.

Problem 2 (5 pts)

1. It's tempting to use gradient descent to try to learn good values for hyper-parameters such as λ and η. Can you think of an obstacle to using gradient descent to determine λ? Can you think of an obstacle to using gradient descent to determine η?

2. L2 regularization sometimes automatically gives us something similar to the new approach to weight initialization (i.e., we initialize the weights as Gaussian random variables with mean 0 and standard deviation $1/\sqrt{n_{in}}$ where n_{in} is the number inputs to a neuron). Suppose we are using the old approach to weight initialization (i.e., we initialize the weights as Gaussian random variables with mean 0 and standard deviation 1). Sketch a heuristic argument that:

   (a) Supposing λ is not too small, the first epochs of training will be dominated almost entirely by weight decay.

   (b) Provided $\eta\lambda \ll n$ the weights will decay by a factor of $\exp(-\eta\lambda/m)$ per epoch.

   (c) Supposing λ is not too large, the weight decay will tail off when the weights are down to a size around $1/\sqrt{n}$, where n is the total number of weights in the network.

3. Modify your code for problem 1.3 to use an adaptive learning rate using AdagradOptimizer in TensorFlow. Record your output and save your code as `prob2.py`. Comment on how you chose any of the parameters.

Problem 3 (5 pts)

1. Autoencoders learn to recreate their input after passing it through a sequence of layers in a neural network. This goal can be satisfied by simply learning the identity function. What design choices can be made (e.g. about the network architecture) to prevent them from learning the identity function? Pick at least two and discuss why an autoecncoder with your design choice would be more useful than one without it.

2. In class we discussed denoising autoencoders that learn to correct a corruption process that we model with $C(\tilde{x}|x)$. Describe an example of such a corruption process, express it in a formula, and provide a plot of an original two-dimensional dataset (e.g. samples from $y = x^2$ or anything of your choosing) and the same dataset after you've corrupted it with your function C.

3. Build an autoencoder with one hidden layer and two neurons for the MNIST dataset. Plot the embedding layer and color it by the label of each image. Describe what you can learn about this dataset from looking at the points embedded into the latent space of an autoencoder.

Problem 4 (5 pts)

1. Describe the difference between a generative adversarial network (GAN) and a variational autoencoder (VAE).

2. Using the skeleton code in vae.py, fill in the TODOs to build a VAE. Plot 10 ten different inputs and their reconstructions. Submit your filled-in version of vae.py along with the assignment.

3. What are some of the problems with trying to train GANs in practice? Name a few and discuss possible ways of fixing/addressing them.

4. Describe the differences between Kullback-Leibler divergence, Jensen-Shannon divergence, MMD, and Wasserstein distance.

5. Why do we often choose the input to a GAN (z) to be samples from a Gaussian? Can you think of any potential problems with this?

6. In class we talked about using GANs for the problem of mapping from one domain to another (e.g. faces with black hair to faces with blond hair). A simple model for this would learn two generators: one that takes the first domain as input and produces output in the second domain as judged by a discriminator, and vice versa for the other domain. What are some of the reasons the DiscoGAN/CycleGAN perform better at this task than the simpler model?

References

[1] "The scipy stack specification." [Online]. Available: https://www.scipy.org/stackspec.html