Advanced Machine Learning 2020 - Exercise 4

December 17, 2019

1. In this exercise you will implement a variational autoencoder using Keras, a powerful high-level library that works on top of TensorFlow (and a few additional libraries).

   (a) (Do not submit) Get familiar with Keras, `https://keras.io/getting-started/sequential-model-guide/`

   (b) (Do not submit) Download and go over the python code, `variational_autoencoder.py`, available on the course Moodle.

   (c) The code provided defines and trains a VAE. Add an encoder which maps MNIST digits to the latent space. Using this encoder, visualize the test set in the latent space (make sure to use Keras functional API when defining the encoder, `https://keras.io/getting-started/functional-api-guide/`). Take one image per digit and print its corresponding mapping coordinates in the latent space, present the answer as a table.

   (d) The VAE is a generative model, we would like to be able to generate new MNIST digits based on the VAE we have trained. Use the following code to define a generator that based on a sample from the latent space, generates a digits.

   ```
   decoder_input = Input(shape=(latent_dim,))
   _h_decoded = decoder_h(decoder_input)
   _x_decoded_mean = decoder_mean(_h_decoded)
   generator = Model(inputs=decoder_input, outputs= _x_decoded_mean)
   ```

   Given a sample from the latent space, for instance, `z_sample = np.array([[0.5, 0.2]])`. Applying the generator, `x_decoded = generator.predict(z_sample)`, results with a generated digit.

   (e) Add a sequence of images which form an interpolation from one digit to another. Namely

   - Take two original images from MNIST of different digits.
   - Find their representation in the latent space.

- Sample 10 points from the line connecting the two representations in the latent space and generate their images using the generator.

(f) The code provided models both the mean and the variance. Fix the variance vector to constant ones and train again when learning only the mean vector. Answer sections (c), (d) and (e) again. (note that in the code the variance variable holds the log of the values, therefore you should set the values to zero).

(g) The autoencoder architecture provided is a simple fully connected neural network. Change `intermediate_dim` parameter in the code to 16 and change the architecture to a ConvNet with the following structure:

- Conv2D(16, (3, 3), activation='relu', padding='same')
- MaxPooling2D((2, 2), padding='same')
- Conv2D(8, (3, 3), activation='relu', padding='same')
- MaxPooling2D((2, 2), padding='same')

The corresponding decoder which you need to define as well will have the following structure:

- Conv2D(8, (3, 3), activation='relu', padding='same')
- UpSampling2D((2, 2))
- Conv2D(16, (3, 3), activation='relu')(x)
- UpSampling2D((2, 2))(x)
- Conv2D(1, (3, 3), activation='sigmoid', padding='same')

Note that the ConvNet is to be added in addition to the fully connected layers. Namely, your network should still output `z_mean` and `z_log_var` with a dimension of `latent_dim` each (see `conv_arch.py` on the course Moodle, the code will require additional changes to work properly). Train the VAE with the new architecture and answer sections (c), (d) and (e) again.

Submission Guidelines:

- The code and data can be downloaded from Moodle.
- Upload your code to nova and include the Path to the code's folder in the PDF submission.
- Make sure your code's folder has the proper read permissions.
- Your code should be readable and well-documented.
- add a README.txt file with the IDs, Names and Emails and if needed details about the code and how to run it.

2. Let P,Q be two normally distributed random variables, $P \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Q \sim \mathcal{N}(\mu_2, \sigma_2^2)$. Derive an expression for the Kullback–Leibler divergence of P and Q, $D_{KL}(P||Q)$.

[11pt]book

Advanced Methods in Machine Learning

Homework Submission Guidelines:

General Instructions

1. **Indicate at the top your names, ID numbers and e-mails.**
2. Assignments are to be submitted in a hard-copy fashion to the cell next to room number 210 in the box with the course name on the 2nd floor in Schreiber building.
3. Assignments should be done in groups of up to 3 students.
4. Submit the theory questions and programming assignments (when relevant) clipped together (only one submission).
5. Print (using Word or LaTex) all discussions and plots in the programming assignments. Discussions should be written right after their corresponding plots.

Theoretical Assignments

1. Follow the exercise specifications exactly. Seek clarifications if needed.
2. Your proofs should be both rigorous and clear.
3. You may use anything proved in class.
4. Your exercise should be submitted printed or in a clear and readable handwriting. Unreadable exercises will not be graded.

Practical Assignments

1. Zip your code, report and any output files and send to tau.aml.2020@gmail.com. The title of the email should be the exercise number and group IDs.
2. Include a README file with details on how to run your code (note that No additional parameter should be added aside for those who were requested).
3. Make sure to indicate in the email the names, IDs of all the group members in the email and in the README file.
4. Follow the exercise specifications exactly.
5. There is no need to print the code.
6. your code should be uploaded to NOVA. Make sure your code is running there and the folder has the proper read permissions. Add the path on the README file.
7. Some assignments will have skeleton code provided on Moodle.

8. Your code should be readable and well-documented.

9. For each practical assignment there might be specific instructions indicating how to execute the code. make sure to follow them exactly.