École d'ingénieurs Télécom Physique Strasbourg — Université de Strasbourg

Télécom Physique Strasbourg - Université de Strasbourg - SDIA/SDSC 2A

CC: Unsupervised Deep Learning
Deep Clustering

1 Introduction

We will use Python with TensorFlow ($\geq$ version 2.0) to define and train the models, scikit-learn for embedding, and Matplotlib for visualisation, and datasets to easily load the data.

You will bring all the previous TPs together to implement and visualise the deep embedding clustering process.

2 Deep Embedded Clustering (DEC)

Download `CC_skeleton.ipynb`.
We will use the answers from TP2 to create a DEC and see how it affects the feature distribution. Since a new plot will be created at each iteration, they wont be output to the screen but will instead be saved to disc in `./DEC_results`.

- Look through The DEC and DEC Execution sections of `CC_skeleton.ipynb`, study and understand the existing code. In particular, relate the sections of code to the DEC components described in slides 47–56 of Lecture 2 (accessible here).
- The autoencoder that DEC is based on is defined in the `create_autoencoder` function in the DEC Autoencoder section, insert the stacked autoencoder used in TP2 (give the middle layer the name 'encoded' and make sure that the model initialisations in the `return` statement reference the correct layers).
- Use `KMeans` from `sklearn` to initialise the DEC clustering on the output of the encoder (see the beginning of the function `fit()`), i.e. encoded `x` (training) data (the number of clusters is in `self .n_clusters`). The function `extract_features()` can be used to do this, but needs to be completed to get the embeddings (encoded) prediction from the autoencoder model.
- Modify the code to reconstruct the test images (the test data is passed as `test_data[0]`) so that the existing code can plot and save them at each training iteration (see the main `for` loop in `fit()`). Do not forget to uncomment the code block.
- Modify the code to plot t-SNE and PCA embeddings of the encoded test images at each training iteration, the embeddings and model predictions are given by the functions `extract_features()` (completed earlier) and `predict()`, which needs to be completed. To get the model predictions you need to use the DEC model. The `argmax` is needed in the return statement of `predict()` to take the clusters with the highest probability as the data point assignments. Do not forget to uncomment the code block.

What do you notice about the feature distributions and the reconstructions as the iterations progress (the reconstruction mean squared error for each iteration is output to the command line)? (These can be viewed in the files tab on the left of the Colab interface.) Can you give a reason for what you observe? Explain in a couple of lines of comments in your code or in a separate text file (in French, or English if you like).