

Computer Lab 2 - 732A70

Johan Alenlöv

1 Introduction

Often when working with models it is not certainty that the available implementations exactly cover the setting that we wish to use. In these cases there are a few different options ranging from reimplementing everything yourself to changing your setup and theory to suite the available implementation. If the code you wish to use is appropriately documented an alternative could be to extend the implementation with your special case while keeping it compatible with the rest of the code.

The focus of this computer lab is to learn how to implement a class, use different libraries and writing python scripts and running them.

2 Requirements

To pass this lab you will need to hand in three (3) files

- ☐ `gmm.py`
- ☐ `analyze.py`
- ☐ `analyze.ipynb`

All the coding in the above files should follow the coding style standards for Python as detailed in PEP-8, see the full document at <https://peps.python.org/pep-0008/>

2.1 `gmm.py`

This file should contain your implementation of a Gaussian mixture model that behaves like the distributions from `scipy`.

For this class you should implement the following functions:

- ☐ `pdf(self, x, wgt, mu, sigma)`
- ☐ `cdf(self, x, wgt, mu, sigma)`
- ☐ `rvs(self, wgt, mu, sigma, size=None, random_state=None)`
- ☐ `fit(self, data, K)`

Here `wgt`, `mu`, and `sigma` are the parameters of the Gaussian mixture model. You are allowed to use already existing implementations of the normal distribution in `scipy.stats.norm` when evaluating the pdf and cdf and also for generating random numbers.

A note on the fit function

For the `fit` function you are allowed to use the `sklearn.mixture` class `GaussianMixture` which has a `fit` function implemented. For all other functions you are **not** allowed to use the `sklearn` library. The parameter `K` in the `fit` function is the number of components in the mixture model.

The function should return the parameters of the fitted Gaussian mixture model.

A note on the rvs function

For the `rvs` function you should generate random variables from a Gaussian mixture model. To do this for each random variable first sample what mixture to use by sampling an index from the `wgt` vector. And then based on this index sample a Gaussian random variable from the Gaussian distribution with the corresponding `mu` and `sigma`.

The `random_state` variable should either be random number generator from `np.random`. If it's `None` you should let it be `np.random.default_rng()`. This random generator should then be used to generate the indices and the Gaussian random variables.

The `size` should tell you how many random variables to generate. If it's `None` then 1 sample should be produced.

A note on testing your code

When implementing a class like this it could be a good idea to have some test cases in the bottom of your file so that you can make sure that the code is working. This code should be wrapped in an `if __name__ == '__main__':` block so that it is not executed when importing the class.

2.2 analyze.py

This file should contain a Python program that reads a file and processes the contents of this file. It should make sure that the file contains comma separated values, where the values are numbers. It should then fit a Gaussian mixture model for various number of mixtures, from $K = 2$ to $K = 10$. Print out the log-likelihood of the data coming from these different models and choose the model with the best AIC¹ and print a text that contains the chosen number of components and all the parameters. The exact text is up to you.

This program should be run by typing the following in a terminal

- `python analyze.py data_file.csv`

Here `data_dile.csv` could be any file name that contains comma separated values. See the file on Lisam for an example of such a file.

To read this file you should read it as a string and then use `split` and convert to store the results in a numpy array of numbers.

For this file to be correct the following functions should be implemented:

- ☐ `calc_llh(data, wgt, mu, sigma)`, calculates the log-likelihood.
- ☐ `calc_AIC(data, wgt, mu, sigma)`, calculates the AIC value.

¹https://en.wikipedia.org/wiki/Akaike_information_criterion

- `string_to_number(string)` which takes the text string, splits it on ',', converts everything to numbers and returns the numbers in a numpy array.

When you pass an external command, like the name of a file, to a python script like above (`python analyze.py data_file.csv`) to read the extra arguments you can find them in `sys.argv` which requires the module `sys` (standard library) to be imported.

Test with the following script to see what is happening

```
import sys

for arg in sys.argv:
    print(arg)
```

Save it as `test.py` and try to run `python test.py hello this is some arguments`.

2.3 analyze.ipynb

This file should be a jupyter-notebook that loads in a data-set. For this you should run it on both the penguin dataset² and one more dataset of your choice from kaggle. Here you can use `pandas` to read the data. For each numeric variable in the datasets, fit the best Gaussian mixture model and plot the pdf and cdf using `matplotlib`.

You should also simulate your own set of numbers from the same Gaussian mixture model. Plot the data from the dataset and the simulated data for each variable as histograms.

3 Handing in the files

Compress all your files into either a `.zip` or `.tar.gz` archive which should then be uploaded to Lisam.

²<https://www.kaggle.com/code/parulpandey/penguin-dataset-the-new-iris>