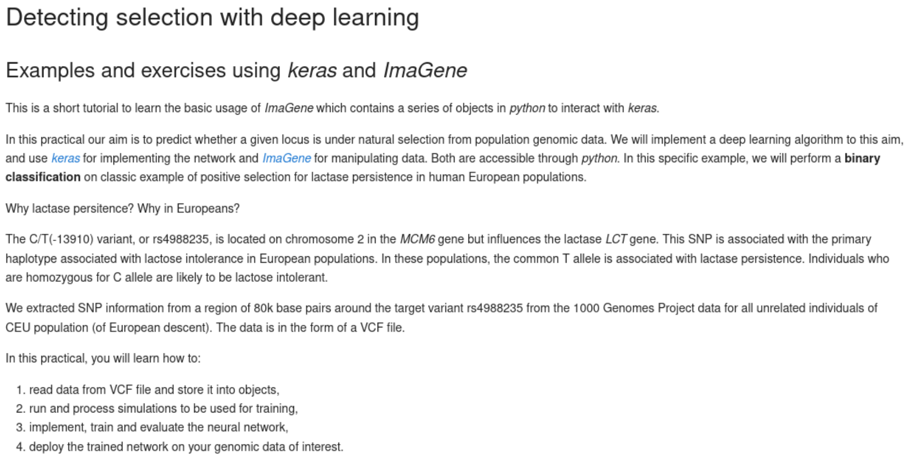
Machine learning for detecting selection - practical using ImaGene

Matteo Fumagalli

1 Motivation

We will be using ImaGene for this workshop. ImaGene is simply a collection of objects and methods in python to interact with keras `keras.io/`. ImaGene reads msms simulation files to be used for training and VCF files for deploying networks to genomic data. ImaGene is suitable for playing with different architectures and data processing, to then use some more efficient implementations like in DNADNA.

We will go over a jupyter-notebook which will look like this once opened:



Detecting selection with deep learning

Examples and exercises using *keras* and *ImaGene*

This is a short tutorial to learn the basic usage of *ImaGene* which contains a series of objects in *python* to interact with *keras*.

In this practical our aim is to predict whether a given locus is under natural selection from population genomic data. We will implement a deep learning algorithm to this aim, and use *keras* for implementing the network and *ImaGene* for manipulating data. Both are accessible through *python*. In this specific example, we will perform a **binary classification** on classic example of positive selection for lactase persistence in human European populations.

Why lactase persitence? Why in Europeans?

The C/T(-13910) variant, or rs4988235, is located on chromosome 2 in the *MCM6* gene but influences the lactase *LCT* gene. This SNP is associated with the primary haplotype associated with lactose intolerance in European populations. In these populations, the common T allele is associated with lactase persistence. Individuals who are homozygous for C allele are likely to be lactose intolerant.

We extracted SNP information from a region of 80k base pairs around the target variant rs4988235 from the 1000 Genomes Project data for all unrelated individuals of CEU population (of European descent). The data is in the form of a VCF file.

In this practical, you will learn how to:

1. read data from VCF file and store it into objects,
2. run and process simulations to be used for training,
3. implement, train and evaluate the neural network,
4. deploy the trained network on your genomic data of interest.

2 Instructions

There are two possible options to run this workshop. Either should work. Depending on your machine, option 1 may run faster than option 2. Also, some extra functionalities like saving and loading objects (not required for running the practical) may not work on google colab.

2.1 Option 1: run locally

If you are using a decent machine/laptop and are comfortable installing packages, you can run this workshop locally on your machine. You can achieve this by running the following commands in your terminal:

1. `git clone https://github.com/mfumagalli/ImaGene`
2. `mamba create -n ImaGene python=3 tensorflow=2 keras=2 numpy scipy scikit-image scikit-learn matplotlib pydot arviz jupyter jupyterlab -y`
3. `mamba activate ImaGene`
4. `pip3 install --force --no-deps protobuf`
5. `cd ImaGene; git pull`
6. `cd Tutorials/Workshop`
7. `jupyter-notebook workshop.ipynb`

If you don't have `mamba` installed you can replace it with `conda`.

We will be using the `workshop.ipynb` notebook which can be found here `https://github.com/mfumagalli/ImaGene/tree/master/Tutorials/Workshop`.

You should also download the training data from this link `https://drive.google.com/file/d/18T_ryHihKwNCdgRnNbPg05TIzHG994NV/view?usp=sharing`. Save the file `Data.tar.xz` in your working directory. You can also decompress it in advance.

Done!

2.2 Option 2: run on colab

If you use google colab you have two possibilities. You can download the training data from this link `https://drive.google.com/file/d/18T_ryHihKwNCdgRnNbPg05TIzHG994NV/view?usp=sharing` and save it in your Google Drive landing page. Alternatively, you can avoid downloading this file manually but in the jupyter notebook you have to run an additional cell (as indicated). Both should work.

Open colab at `https://colab.research.google.com/` and follow these instructions:

- click on "GitHub" in the opening window
- enter the following URL `https://github.com/mfumagalli/ImaGene/tree/master/Tutorials/Workshop/workshop-colab.ipynb` or `https://github.com/mfumagalli/ImaGene/`
- press enter
- click on 'workshop.ipynb'

When you run some cells, it asks for your authorisation to proceed (both for trusting this notebook and for linking the notebook to your Google Drive). You have to provide authorisation. It may output a running error in the first cell. If you run it again it should work with a warning.

Done!

3 Tasks

3.1 Exercise 1: data preparation

After going over parts 1 and 2 of the notebook together, perform your own filtering of the data according to specifications compatible with your species of interest. For instance:

- if you don't have an outgroup species, you may wish to polarise your alleles as major and minor
- if your data is of very low quality, you may wish to make sure to filter out singletons (and perhaps doubletons)
- if you have high-quality data on a model organism, you may relax your filtering
- if you have fragmented data, you can crop to a smaller number of SNPs
- play with sorting options: if your data is highly structured, you may wish to sort of genetic distances; or use any other options that sound interesting to you
- ...

Use your own imagination; remember to perform the same filtering between observed and simulated data. Each function as a help page which can be access by typing ? in front of its name, such as '?gene.majorminor'.

3.2 Exercise 2: building, training, and testing your network

After going over parts 3 and 4 of the notebook together, attempt to replicate these steps by "gently" modifying some of the parameters in the network. For instance, you have links to the relevant pages of keras manual to have more or less filters, modify kernel or pooling size, or the number of units in the final dense layer.

Then, answer these questions:

- How can you monitor whether your network is learning and not overfitting?
- Should you report the training or testing accuracy to assess the performance of your algorithm?
- How would you interpret the results in part 4, the deployment to the observed data? What does that number indicate?

3.3 Exercise 3: BYOB (Build Your Own Best network)

Fill in the missing values and lines at the end of the `workshop` notebook to perform binary or multiclass classification of selection.

4 Important notes

- You don't have to run new simulations to generate training data; data is already provided in `Data/`; you will just need to set `path_sim` variables appropriately (in practice you just need to set this variable to "./").
- If you find some of the operations to be very slow, reduce the data ("images") used; this can be done by setting `max_repl` parameter in `file_sim.read_simulations(parameter_name='selection_coeff_hetero', max_nrepl=200)`
- If the training is too slow for you, you can also reduce the number of epochs or batches of data (e.g. we created 20 batches of data, you can use half of them only).

5 Further topics for discussion

- What happens if you retain all SNPs or remove singletons or doubletons. Is the networking learning faster with the same accuracy? Or are rare variants important for this task?
- Similarly to above, what happens if you fold your data, i.e. convert ancestral/derived allelic status into major/minor? Do you obtain similar prediction accuracy?
- What is the effect of modifying your architecture? You can read how to modify a sequential model in keras here `https://keras.io/guides/sequential_model/`. Note that layers are accessible with `model.layers` attribute.
- Think about the difference between a binary classification, a multiclass classification and regression by reading the other tutorials. In particular, have a look at the different final layers and/or activation function.