Deep Learning Project – DNA Sequencing

Part 3 : Implementation

LBRTI2101(B) – Data Science in Bioscience Engineering

Groupe n°6

Alexis Franco
Noé Laloux
Tom Kenda

Professor : E. Hanert, P. Bogaert

Academic year 2020 - 2021

Objective

Creating a large ORF database, labelled as “gene” or “noise”

```
ATGGCAGGTGAAGCAGTTTCGGAACACACACCAGATTCGCAGGAAGTAACAGTAACTAGCGTAGTTTGTTGCCTCGATTCTGTGG
TATACTCTGTTGTGGCACCGCTAACAGTAACGGTGGCCGTGGAAACAATTGCAGAGGAGATGGATTCAGTGCACACATGA,gene

ATGAGGCAACCGTCGACAACCTTATTATCGAAAAAGAACAACAAGTTCACATGCTTGTTACTCTCTATAACTAGAGAGTACTTTTTTTGGAAGCAA
ATGGTCTACGAATCGTCAATCGCTTGCGGTTATGGCACGAAGAACAATGCAATAGCTCTTACAAGCCACTACATGACAAGCAACTCATAA,noise
```

1. Extracting all ORFs from genome (1/2)

/genomes/refseq/fungi/Saccharomyces_cerevisiae

Name	Last modified	Size
Parent Directory		-
chrI.fna.gz	2018-04-09 00:20	67K
chrII.fna.gz	2018-04-09 00:20	237K
chrIII.fna.gz	2018-04-09 00:20	93K
chrIV.fna.gz	2018-04-09 00:20	443K
chrIX.fna.gz	2018-04-09 00:20	129K
chrV.fna.gz	2018-04-09 00:20	169K
chrVI.fna.gz	2018-04-09 00:20	80K
chrVII.fna.gz	2018-04-09 00:20	318K
chrVIII.fna.gz	2018-04-09 00:20	164K
chrX.fna.gz	2018-04-09 00:20	217K
chrXI.fna.gz	2018-04-09 00:20	195K
chrXII.fna.gz	2018-04-09 00:20	308K
chrXIII.fna.gz	2018-04-09 00:20	268K
chrXIV.fna.gz	2018-04-09 00:20	229K
chrXV.fna.gz	2018-04-09 00:20	318K
chrXVI.fna.gz	2018-04-09 00:20	276K

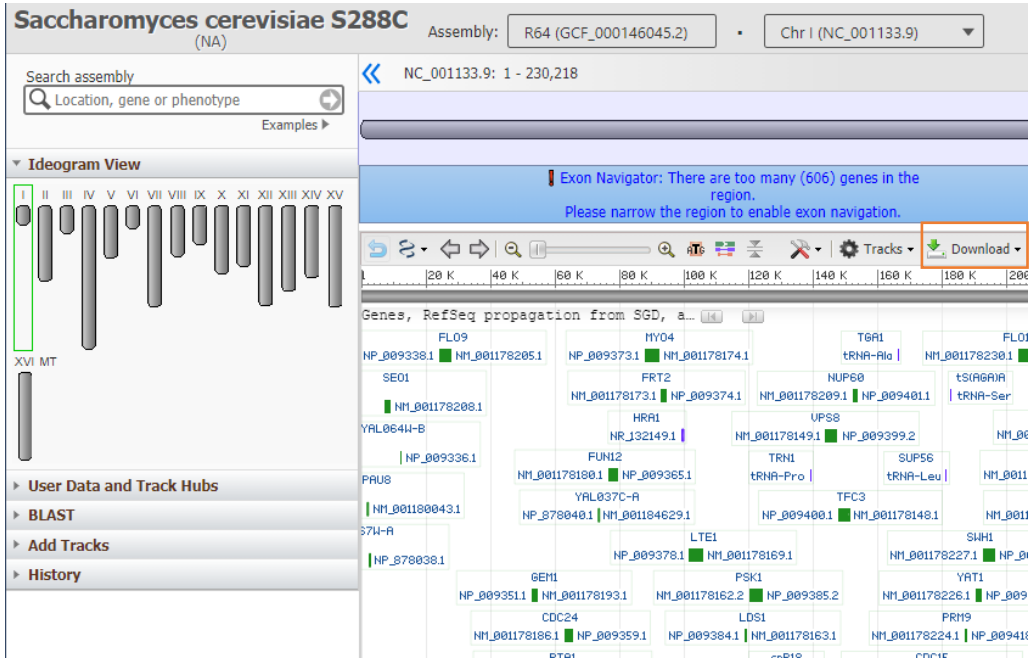
1. Extracting all ORFs from genome (2/2)

Scan:

Append to list:

ATGCATGCGACTACGATCGATCTAACTTGA
ATGCGACTACGATCGATCTAA

2. Extracting Genes from Genome



Genes, RefSeq propagation from SGD, annotation version R64-2-1

Accession	Start	Stop	Gene symbol	Strand	NCBI Gene ID	Name
NC_001133.9	1807	2169	PAU8	minus	851229	
NC_001133.9	2480	2707		plus	1466426	
NC_001133.9	7235	9016	SEO1	minus	851230	
NC_001133.9	11565	11951		minus	851232	
NC_001133.9	12046	12426		plus	851233	
NC_001133.9	13363	13743	TDA8	minus	851234	
NC_001133.9	21566	21850		plus	851235	
NC_001133.9	22395	22685		minus	1466427	
NC_001133.9	24000	27968	FLO9	minus	851236	
NC_001133.9	31567	32940	GDH3	plus	851237	
NC_001133.9	33448	34701	BDH2	plus	851238	
NC_001133.9	35155	36303	BDH1	plus	851239	
NC_001133.9	36509	37147	ECM1	plus	851240	
NC_001133.9	37464	38972	CNE1	plus	851241	

Creating Database

If ORF in Gene_List:

label = ‘gene’

else:

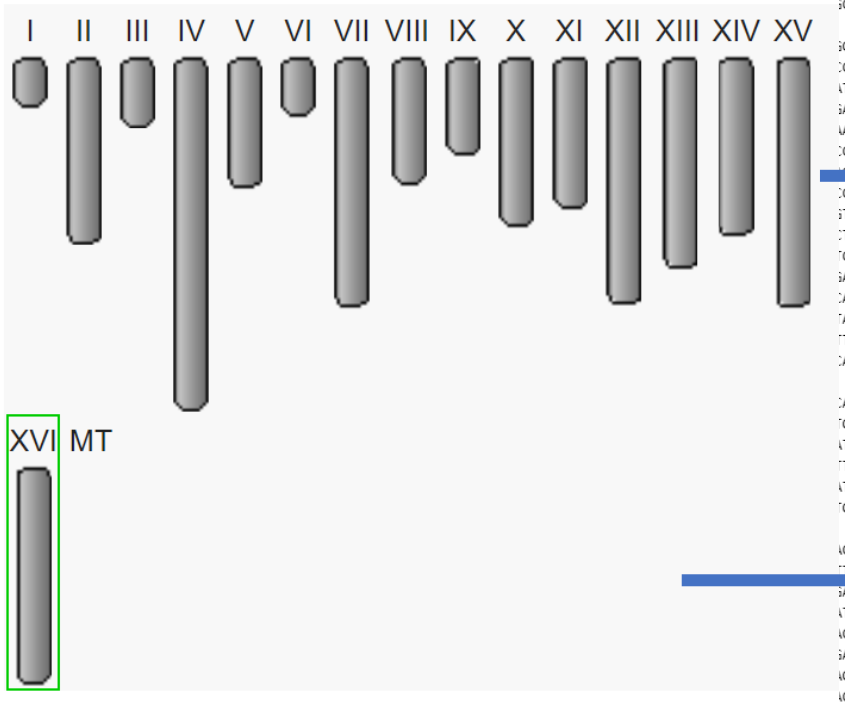
label = ‘noise’

~5600 genes

+100 000 noise -> cut

Final training dataset : ~12000 sequences

Two Distinct Datasets



dataTrain.txt 9,02 Mo

dataTest.txt 142 Ko

Extracting data from datasets

Using Python :

1) Read the specific file
2) Extract its content line by line
3) Read the content of each line
4) Separate data in two lists :
   - sequences
   - labels

How is data stored ?

sequence_1,label_1
...,...
sequence_k,label_k

How do we want it?

sequences = [sequence_1, … , sequence_k]
labels = [label_1, …, label_k]

```
# Reading the file
file = open("dataset.txt", 'r')
lines = file.readlines()
file.close()

# Creating lists
sequences = []
labels = []
for line in lines:
    seq, lab = line.strip().split(",")
    sequences.append(seq)
    labels.append(lab)
```

Text data to numerical data : Labels

Labels : Modifying labels as text to have labels as numbers

labels = [‘noise’, ‘gene’, ‘gene’ , …]

numerical_labels = numerical_labels = [0, 1, 1, …]

```
numerical_labels = []
for lab in labels:
    if lab == 'noise':
        numerical_labels.append(0)
    else:
        numerical_labels.append(1)
```

How does that work ?

For each label (*lab*) , add its numerical value in the list (*numerical_labels*)

Text data to numerical data : Sequences

Sequences : Modifying nucleotides as text to have nucleotides as numbers

sequences = [‘ATG …… TGA’ ,
‘ATG … TAA’ , …]

numerical_sequences = [[[1, 0, 0, 0], [0, 1, 0, 0], …] ,
[[1, 0, 0, 0], [0, 1, 0, 0], …] , …]

```
numerical_sequences = []
for seq in sequences:
    num_seq = []
    for nucleotide in seq:
        if nucleotide == "A":
            num_seq.append([1, 0, 0, 0])
        elif nucleotide == "T":
            num_seq.append([0, 1, 0, 0])
        elif nucleotide == "G":
            num_seq.append([0, 0, 1, 0])
        else:
            num_seq.append([0, 0, 0, 1])
    numerical_sequences.append(num_seq)
```

How does that work ?

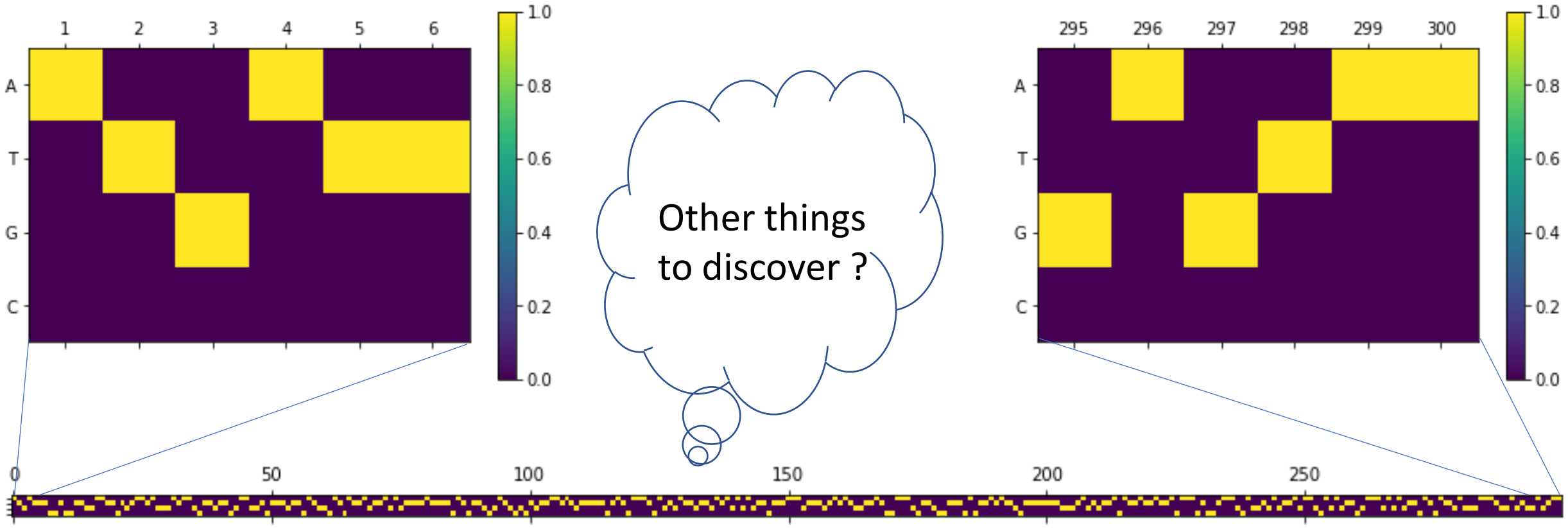
For each sequence (*seq*) :

1) Create a new empty list (*num_seq*)
2) For each *nucleotide* in the sequence (*seq*), add the specific vector to the list *num_seq*
3) After the last nucleotide of the sequence, add the fully completed list *num_seq* to *numerical_sequences*

What does the model see ?

The start of the sequence with an 'ATG'
(START codon)

The end of the sequence with an 'TAA'
(STOP codon)



Other things
to discover ?

Final manipulation of data

Size of the sequences :

Our model has a fixed architecture :
→ number of inputs must be constant
→ vectors must have the same size !

Elongation of the sequences with chosen vectors (*to_add*) :
- [0, 0, 0, 0]
- [0.25, 0.25, 0.25, 0.25]
- Random between those of the nucleotides (e.g. [0, 0, 1, 0])
- other ?

```
def elongation(vec, desired, to_add=[0,0,0,0]):
    while len(vec) < desired:
        vec.append(to_add)
```

Randomization :

We must randomize our data to give gene and orf at the same time and not by 'block'.

→ using python *random* library and the *random.shuffle()* function

Creation of batchs :

Since we use a big amount of data, working with smaller amount of data by dividing our database in several batchs can help to use less memory while storing and transforming a lot of variables during the model.

The tools we used inside Python

Classical function/packages :

- Random – shuffling the DNA sequence (link)
- .CSV – manipulating labelled genes (link)
- Numpy – manipulating DNA sequence as numpy array and reshape the data (link)
- .TXT – text manipulation to extract the sequence (link)

Specific tools for deep learning

- TensorFlow
  - ↳ Keras

Specific tools for deep learning

TensorFlow

- First developed by Google and then commercialized in 2015 (1.0)
- 2.0 released in September 2019 – now widely used (Coca-Cola, Uber, Airbnb, Intel, ….)
- End-to-end open source platform for machine learning
- Particular focus on training and inference of deep neural networks (learn more)

TensorFlow.Keras

- Python interface for artificial neural network
- Contain numerous implementation of model, activation function, … (learn more)

Overview of the implementation in Python :

```
import tensorflow as tf

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(2, activation=tf.nn.softmax))

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(SequenceArray, arrayLabels, epochs=5)

model.summary()
```

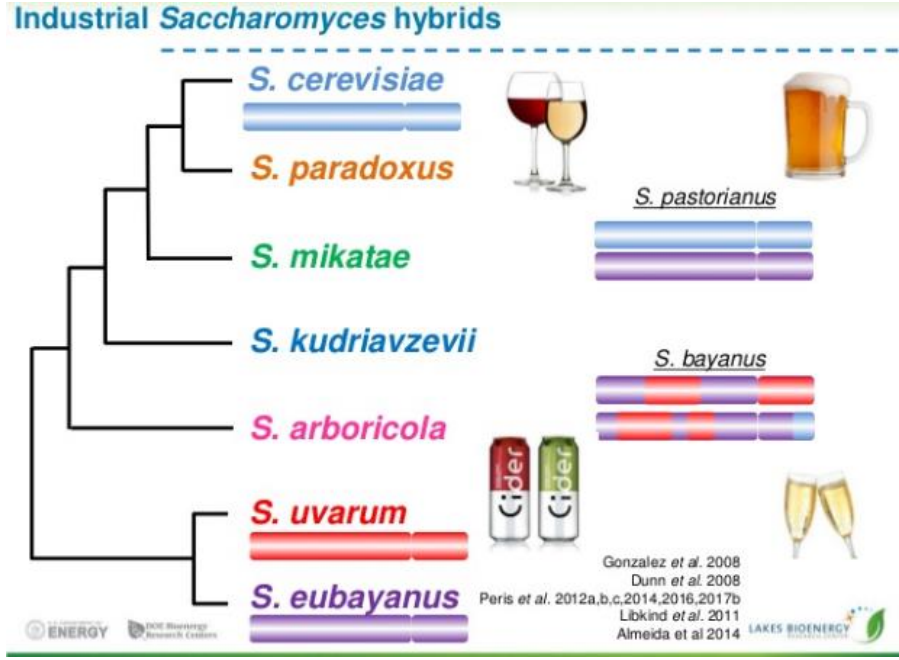
Expected results (1/2)

1. For S. cerevisiae :
   → Used to train the model
   → Should give us a good prediction of coding genes ~ > 95% precision

```
In [3]: is_coding_for_protein('dna_sequence')
Out[3]: True
```

2. For other species from the same genre ?
   → S. arboricola, eubayanus, paradoxus
   → On new hybrids from those species

Reminder : big interest in beverage production, …
Winans (2019)



Navarro (2017)

Expected results (2/2)

3. For other genre ?

- Would take another training from the model
- Probably not working on more complex organism

➔ *See you in 4 weeks for the actual results !*

Bibliography

Winans (2019) – Saccharomyces arboricola and Its Hybrids' Propensity for Sake Production

Navarro (2017) – Mining Saccharomyces diversity and experimental evolution for cellulosic biofuel

https://www.tensorflow.org/

https://en.wikipedia.org/wiki/TensorFlow#cite_note-14

https://keras.io/about/

https://fr.wikipedia.org/wiki/Keras