Computational Science Project

Computational analysis of the functionality of proteins, and generation of functional proteins

Joseph Touzet

Task 1 - Reading and formatting the data

A - One-hot encoding sequences:

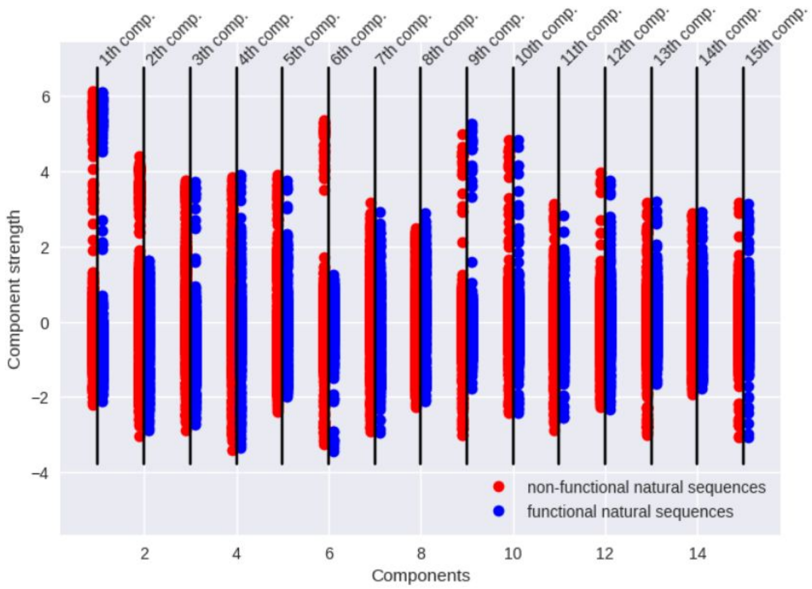
```
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
```

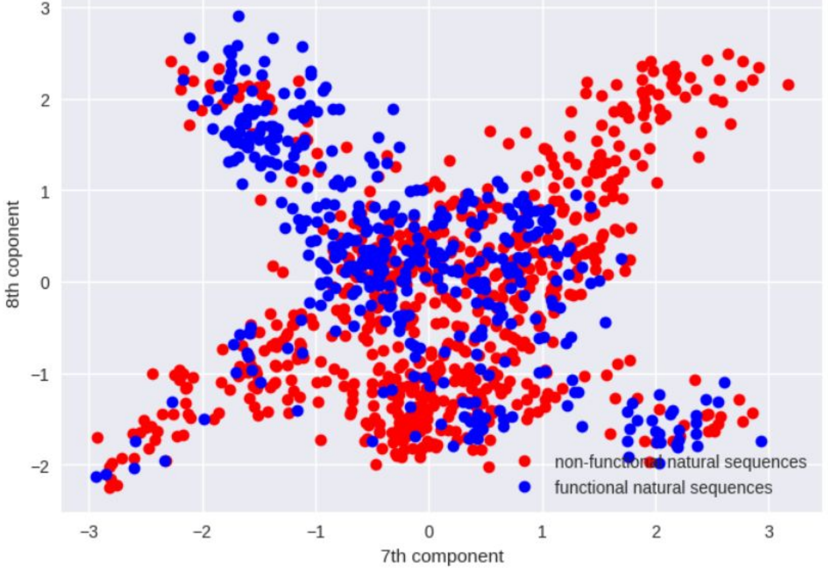
We tried to validate our *one-hote encoding* function on the simple sequence "C-W": This correctly gives us a 60-long vector (3*20) with only a 1 on the second place (corresponding to the first "C"), a one on the second-to last place (corresponding to the "W") with no one in between (corresponding to the "-").

B - Reading all sequences:

```
{
    "name": "sequence_1",
    "functional": true,
    "sequence": "-----SLEELRKEIESIDREIVELIARRTYVAKTIAQIKRERGLPTTDESQEQRVMERAGSNAKQFD-VDANLVKAIFK
LLIELNKEEQRENR---",
    "hot_encoded_sequence": [
        0,
        0,
        0,
        0,
        0,
        ...
    ]
}
```

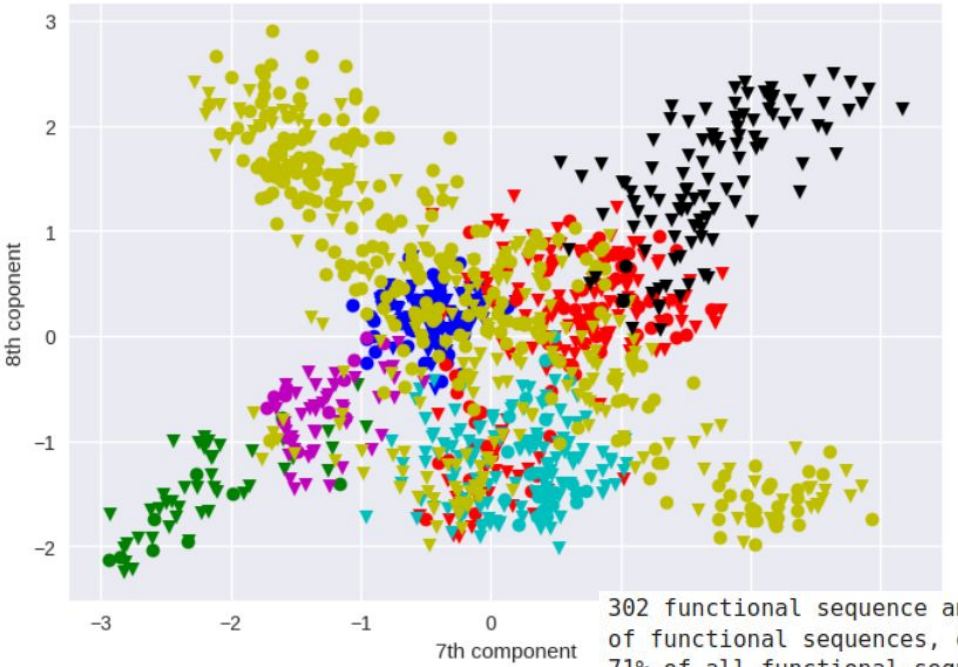
Task 2 - Principal Component Analysis





Task 3 - First analysis: clustering

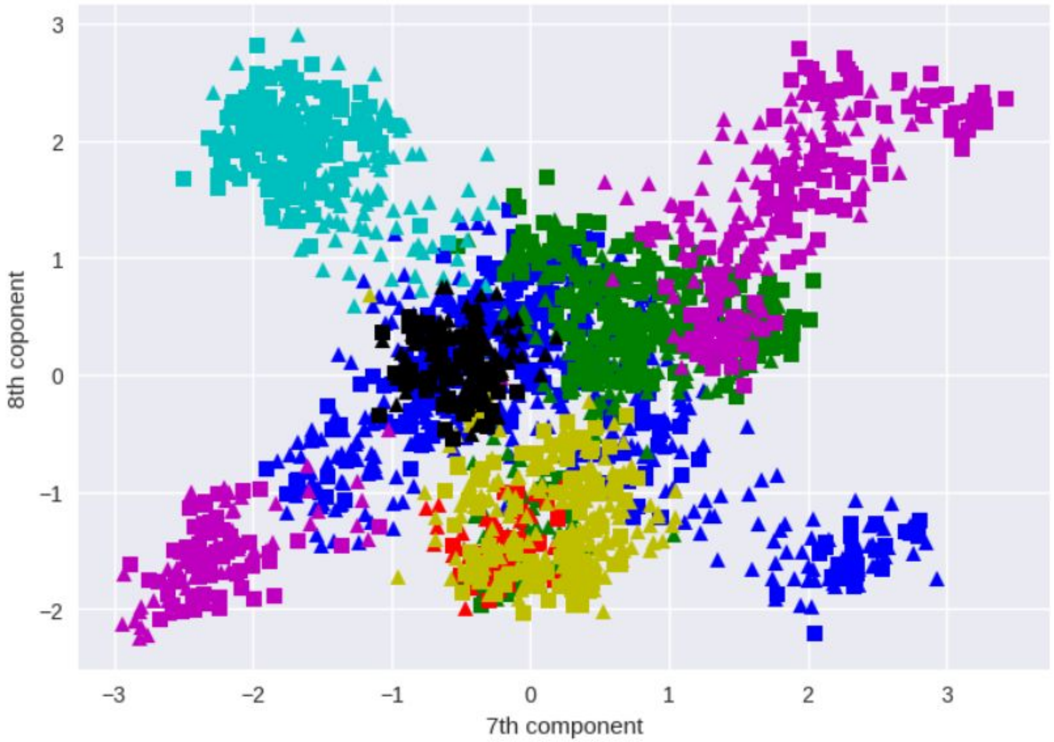
3.1 - Clustering in natural space: functionality




```
302 functional sequence and 262 non-functional sequences in cluster with more than 37%
of functional sequences, corresponding to 54% functional sequences in this group, and
71% of all functional sequences
121 functional sequence and 445 non-functional sequences in cluster with less than 37%
of functional sequences, corresponding to 79% non-functional sequences in this group,
and 63% of all non-functional sequences
```

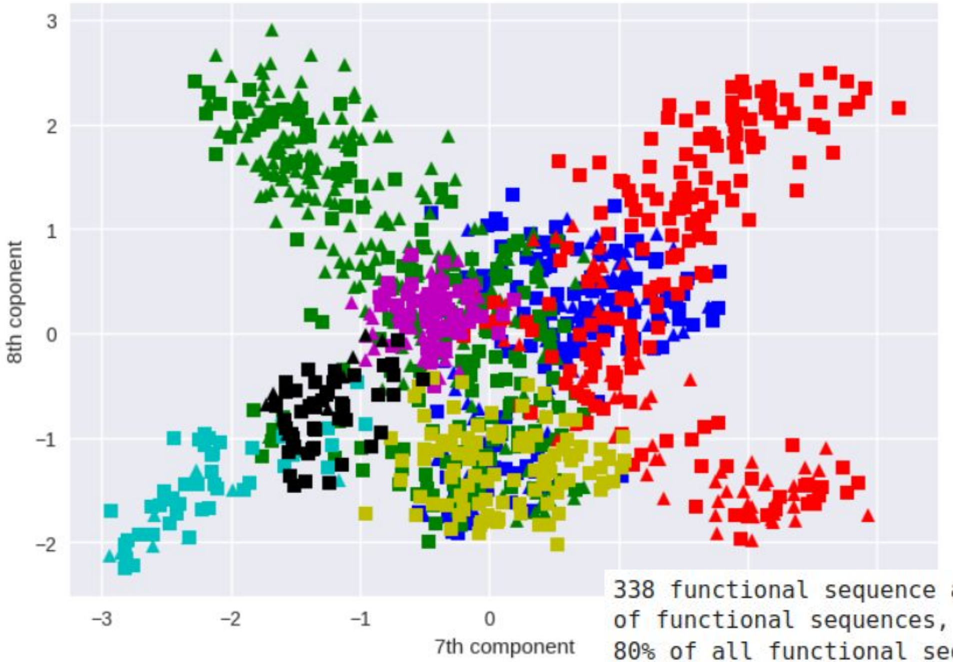
Task 3 - First analysis: clustering

3.1 - Clustering in natural space: natural vs artificial



Task 3 - First analysis: clustering

3.2 - Clustering in PC space: functionality




```
338 functional sequence and 341 non-functional sequences in cluster with more than 37%
of functional sequences, corresponding to 50% functional sequences in this group, and
80% of all functional sequences
85 functional sequence and 366 non-functional sequences in cluster with less than 37%
of functional sequences, corresponding to 81% non-functional sequences in this group,
and 52% of all non-functional sequences
```

Task 4 - Further analysis: training better models

4.a - decision tree:

```
Natural data, test:
TP: 64%, FP: 17%
FN: 36%, TN: 83%

Natural data, train:
TP: 81%, FP: 9%
FN: 19%, TN: 91%

Artificial data:
TP: 74%, FP: 18%
FN: 26%, TN: 82%
```

4.b - logistic regression:

```
Natural data, test:
TP: 65%, FP: 13%
FN: 35%, TN: 87%

Natural data, train:
TP: 100%, FP: 0%
FN: 0%, TN: 100%

Artificial data:
TP: 73%, FP: 18%
FN: 27%, TN: 82%
```

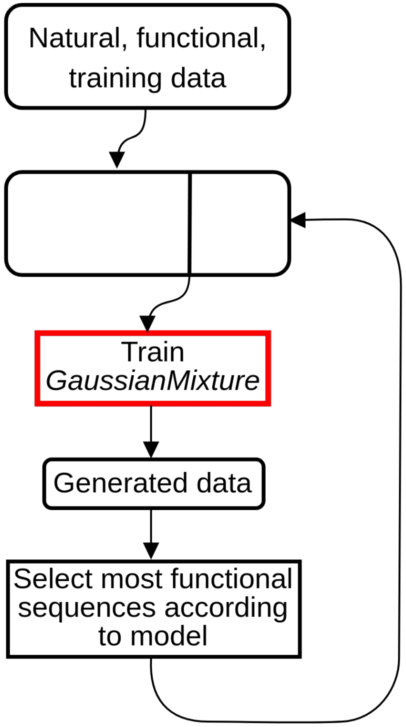
4.c - neural network:

```
Natural data, test:
TP: 62%, FP: 14%
FN: 38%, TN: 86%

Natural data, train:
TP: 100%, FP: 0%
FN: 0%, TN: 100%

Artificial data:
TP: 72%, FP: 15%
FN: 28%, TN: 85%
```

Task 5 - Generating artificial functional sequences




```python
from sklearn.mixture import GaussianMixture

# proportions of generated data for each training epochs:
gen_multipliers   = [5,  10, 10, 10, 10, 10, 15, 15, 20, 20, 20, 20, 20] # x
train_multipliers = [.3, .5, .7, 1,  2,  2,  3,  3,  4,  4,  5,  5,  10] # y*x

# generator model:
generator_GM = GaussianMixture(n_components=n_components, random_state=10)

# training loop:
eigen_train = functional_eigen_value_nat
for i, (gen_multiplier, train_multiplier) in enumerate(zip(gen_multipliers, train_multipliers)):
    generator_GM.fit(eigen_train)

    n_gen          = int(len(functional_eigen_value_nat)*gen_multiplier)
    eigen_gen, _   = generator_GM.sample(n_gen)
    X_gen          = pin_01(model_pca.inverse_transform(eigen_gen))
    X_gen_fitness = model_logic.predict_proba(X_gen)[:, 1]

    n_select    = int(len(functional_eigen_value_nat)*train_multiplier)
    fitest_idx  = np.argpartition(X_gen_fitness, -n_select)[-n_select:]
    eigen_train = np.concatenate((functional_eigen_value_nat, eigen_gen[fitest_idx]), axis=0)

    avg_fitness = np.mean(X_gen_fitness)
    print(f"{i+1}th/{len(gen_multipliers)} epoch: {round(avg_fitness*100)}% average probability of
```

Task 5 - Generating artificial functional sequences

Training:

```
1th/13 epoch: 55% average probability of functionality according to logistic regression
2th/13 epoch: 65% average probability of functionality according to logistic regression
3th/13 epoch: 70% average probability of functionality according to logistic regression
4th/13 epoch: 73% average probability of functionality according to logistic regression
5th/13 epoch: 77% average probability of functionality according to logistic regression
6th/13 epoch: 84% average probability of functionality according to logistic regression
7th/13 epoch: 85% average probability of functionality according to logistic regression
8th/13 epoch: 88% average probability of functionality according to logistic regression
9th/13 epoch: 89% average probability of functionality according to logistic regression
10th/13 epoch: 91% average probability of functionality according to logistic regression
11th/13 epoch: 91% average probability of functionality according to logistic regression
12th/13 epoch: 93% average probability of functionality according to logistic regression
13th/13 epoch: 92% average probability of functionality according to logistic regression
```

Task 5 - Generating artificial functional sequences

Intersection with training data:

```
0% of the generated sequences existed in the natural data
```

5.1 - Clustering:

```
94% of the generated sequences are clustered within "primarly functional" clusters
```

5.2 - Decision Tree:

```
94% of the generated sequences are predicted to be functional by the decision tree model
```

5.3 - Neural network:

```
94% of the generated sequences are predicted to be functional according to the neural network model
```

Conclusion

- Good separation of functional/non-functional sequences
- Efficient generation of supposedly functional sequences
- Generated sequences functional according to all different models