Guide RNA Clustering Homework

Objectives

The goal of this homework was three fold:

- Analyze a set of sgRNA using a type of machine learning algorithm known as clustering in order to find interesting properties,
- Correlate these clusters with the activity of each guide to find a possible predictor of efficiency,
- Compute the genes that appear to be essential and find a possible explanation.

Clustering

Choosing features

The first step of this work was to choose how the guide were going to be characterized.

Eleven features were selected as a basis for the cluster analysis:

- The number of nucleotides that make up the guide (A, T, G, C),
- The count of several dinucleotide : GG, TA, TT, AG, AC and GC ; chosen for their importance in a previous work (Doench, 2016),
- The melting temperature using a merged method based on three different thermodynamic tables (Panjkovich, 2005).

Computing features

The count of nucleotides and dinucleotides for each guide was done using a script written in Python (cf. annex 1).

The melting temperature was calculated using a software called dnaMATE.

A text file containing the features for each guide was used for the clustering.

Clustering

The clustering was done using the Scikit-learn machine learning library written in Python. The clustering was done using the k-means algorithm.

A Python script was written in order to load the required data, compute the k-means and output the coordinates of each cluster as well as figures displaying the data (cf. annex 2).

Results

Several parameters for the k-means algorithm were tested like the number of cluster, the reduction of the data using PCA, the method for initializing the cluster.

The chosen parameters were the following:

- Number of cluster = 4,
- No dimensionality reduction,
- Number of initialization = 100.

The resulting coordinates for the 4 clusters are showed in table 1.

Cluster	A	T	G	C	GG	TA	TT	AG	AC	GC	Tm	Count	%
0	5	7	3	5	0,3	1,3	1,9	0,9	1,0	0,6	47	15001	23%
1	3	3	6	8	1,1	0,3	0,4	1,1	1,1	2,7	59	12968	20%
2	7	4	4	5	0,4	1,2	0,7	1,6	1,6	0,7	48	13570	21%
3	4	5	5	6	0,7	0,6	0,7	1,2	1,3	1,5	53	22537	35%

Table 1: Features value for each of the 4 clusters along with the number of each guide per cluster

The main characteristics on average for each cluster are as follows:

- Cluster 0 is made of guides with a low amount of GC and GG but with a high amount of TA and a low Tm as well as a high amount of T and a medium amount of A.
- Cluster 1 is made of guides with a high amount of GC and GG but with a low amount of TA and a high Tm.
- Cluster 2 is made of guides similar to the one of Cluster 0 (low GC and GG) but have a medium amount of T and a high amount of A.
- Cluster 3 is made of guides that have medium value in all the features.

Concerning the amount of guides per cluster, we can see that cluster number 0 to 2 have roughly the same number (from 20 to 23% of the sgRNA) while cluster 3 (the 'medium' cluster) encapsulate more than a 3rd of the guide.

Several figures were created to visualize the difference between each cluster and of far they are for the average "guide" (black cross on each plot). The following color were attributed to each cluster: cluster 0 in blue, cluster 1 in green, cluster 2 in orange and cluster 3 in violet. The center for each cluster was represented using a circle filled the number of the cluster.

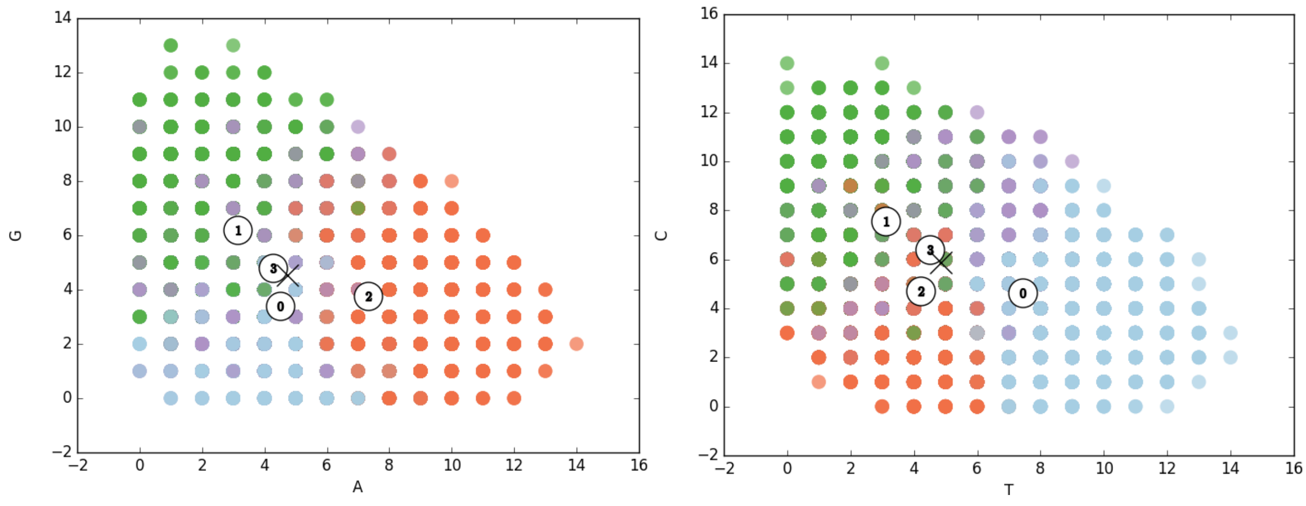

Figure 1: Scattering of the guide based on their number of nucleotide A and G (left) and nucleotide T and C (right)

Figure 1 allowed to visualize the difference between the first 3 clusters while it is hard to see any area belonging to cluster 3.

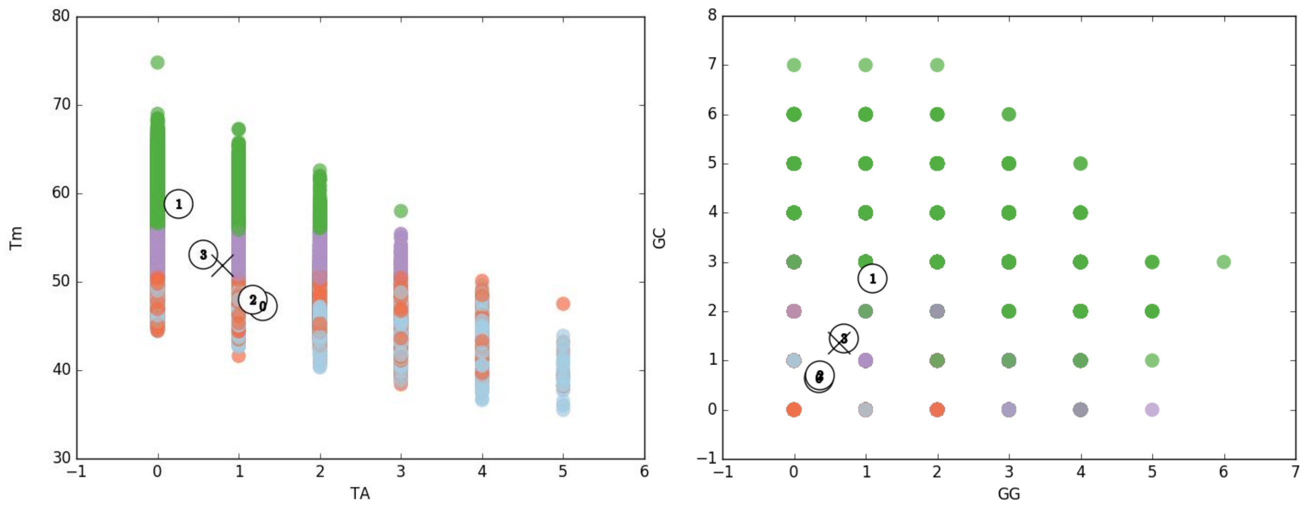

Figure 2: Scattering of the guide based on their number of dinucleotide TA and Tm (left) and dinucleotide GG and GC (right)

Figure 2 allowed to visualize the position occupied by cluster 3 with a medium TM and the similarity between cluster 0 and 2.

Predicting activity

Guide activity

The next step was to determine the activity of each guide. This was done using the following equation:

$$Activity = -\log_2 \frac{count\ t2}{count\ t1}$$

With:

- count t2 = average normalized count of cells with the drug PLX after 14 days,
- count t1 = normalized count of cells before introduction of the plasmid.

The activity was then normalized by percent-rank as described in another paper (Doench, 2014).

Analysis

The relation between each guide's cluster and percent-rank was done using a Python script that divided the guides in quartile based on their percent-rank and then assigned them to their corresponding cluster (cf. annex 3). Table 2 display the resulting distribution.

	Quartile			
Cluster	[0-25%]	[25-50%]	[50-75%]	[75-100%]
0	3738	3805	3718	3746
1	3705	3545	3198	2551
2	2554	2861	3435	4665
3	6021	5809	5668	5057

Table 2: Distribution of each guide in their corresponding cluster and percent-rank quartile

We observe that cluster 0 has an even distribution between each quartile, which would not make it a good predictor of activity.

Cluster 1 is the opposite of a good predictor with only 20% of its guide having a rank of 75% or higher but 29% with a rank of 25% of lower.

Cluster 3 has a distribution similar to cluster 1 with only 22% of its guides having a rank of 75% or higher.

Finally, cluster 2 seems to be a good predictor of activity with 34% of its guides having at least a rank of 75%.

Guide analysis

The average activity of each gene was determined and a percent-rank was created.

The TOP5 genes that had the most impact when targeted were: NHP2L1, NIP7, RPS18, DDX56 and RPL8 (table 3).

gene_name	Activity	Rank
NHP2L1	3,74	99,9%
NIP7	3,55	99,9%
RPS18	3,42	99,9%
DDX56	3,40	99,9%
RPL8	3,35	99,9%

Table 3: Average activity and percent-rank for the 5 most depleted guides

Using NCBI RefSeq and UniProt, the following information were extracted for each gene:

- NHP2L1: Nuclear protein that binds snRNA and may play a role in late stage of spliceosome assembly,
- NIP7: Protein required for proper 34S pre-rRNA processing and 60S ribosome subunit assembly,
- RPS18: Ribosomal protein S18, located at the top of the head of the 40S subunit,
- DDX56: DEAD box proteins that may play a role in processing of pre-ribosomal particles that lead to mature 60S ribosomal subunits,
- RPL8: Ribosomal protein L8, part of the 60S subunit.

We observe that each of these genes play a major role in the cells vital mechanisms.

Conclusion

The clustering algorithm divided the guides into 4 clusters mainly by using the Tm, the amount of A and T nucleotide.

Using the activity of each guide, we concluded that guides with a low amount of GC and GG but with a medium amount of T and a high amount of A were more susceptible to have a high activity.

Finally, we observed that the most depleted guides were targeting genes essential for the cells survival.

Discussion

The goal of this work was just to provide a basis and can be improved a lot.

The chosen feature were small in number and simple. It would be interesting to add more features based on the energy and entropy of this system. It would also be interesting not to look only at the guide but also at its surrounding environment and target.

The number of clusters was intentionally kept low to decrease the computing time and facilitate the interpretation. The algorithm used was only one among many and it would be interesting to compare it to others and more complex ones.

The activity was only described as the log2 between t0 and 14 days after with the drug. It would be interesting to compute a measure of how fast we see change by using both value at 7 and 14 days, with or without drug.

Finally, it would be interesting to have more information on the targets to allow further discrimination between the genes (essential versus non-essential).

Bibliography

Doench, John G., et al. "Rational design of highly active sgRNAs for CRISPR-Cas9-mediated gene inactivation." Nature biotechnology 32.12 (2014): 1262-1267.

Doench, John G., et al. "Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9." Nature biotechnology (2016).

Panjkovich, Alejandro, and Francisco Melo. "Comparison of different melting temperature calculation methods for short DNA sequences." Bioinformatics 21.6 (2005): 711-722.

Annex 1: featureCompute.py

```
import csv
import numpy as np

sequence_import = []
tm_import = []
gene_import = []
sequence = []
tm = []
feature = []



# Open files containing sgRNA
with open('RNA.txt', newline='') as inputfile:
    for row in csv.reader(inputfile):
        sequence_import.append(row)
sequence = np.asarray([item for sublist in sequence_import for item in
sublist])

# Open files containing melting temperature
with open('Tm.txt',newline='') as inputfile:
    for row in csv.reader(inputfile):
        tm_import.append(row)
tm = [item for sublist in tm_import for item in sublist]

# Compute nucleotides feature for each sgRNA
feature = np.zeros((len(sequence),11))
for idx, guide in enumerate(sequence):
    feature[idx,0] = guide.count('A')
    feature[idx,1] = guide.count('T')
    feature[idx,2] = guide.count('G')
    feature[idx,3] = guide.count('C')
    feature[idx,4] = guide.count('GG')
    feature[idx,5] = guide.count('TA')
    feature[idx,6] = guide.count('TT')
    feature[idx,7] = guide.count('AG')
    feature[idx,8] = guide.count('AC')
    feature[idx,9] = guide.count('GC')

# Add melting temperature to the feature array
for idx, temp in enumerate(tm):
    feature[idx,10] = temp

# Save in a txt file
np.savetxt('feature.txt', feature)
```

Annex 1: kmeans_algo.py

```python
from sklearn.cluster import KMeans
import collections
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

X = np.loadtxt(open("feature.txt"))
y = np.asarray(open("feature_list.txt").readlines())

# kmeans algorithm
n_clusters = 4
clusterer = KMeans(init='k-means++', n_clusters=n_clusters, n_init=100)
cluster_labels = clusterer.fit_predict(X)
np.savetxt('cluster_labels.txt', cluster_labels)

# Labeling the clusters
centers = clusterer.cluster_centers_
np.savetxt('cluster.txt', centers)

# Number of gene / cluster
counter = collections.Counter(cluster_labels)
distrib = []
for i in range(n_clusters):
    distrib.append(counter[i])
print(distrib)
np.savetxt('distribution.txt', distrib)

# average_X
average_X = []
for i in range(11):
    average_X.append(np.average(X[:,i]))
print(average_X)
np.savetxt('average_X.txt', average_X)

# Draw plot
colors = cm.Paired(cluster_labels / n_clusters)

# Plot 0-2
plt.scatter(X[:,0], X[:,2], marker='.', s=500, lw=0, alpha=0.7, c=colors)
plt.xlabel(y[0])
plt.ylabel(y[2])
plt.scatter(centers[:, 0], centers[:, 2], marker='o', c='white', alpha=1,
s=500)
for i, c in enumerate(centers):
    plt.scatter(c[0], c[2], marker='$%d$' % i, alpha=1, s=50)
plt.scatter(average_X[0], average_X[2], marker='x', c='black', alpha=1, s=300)
plt.savefig('0-2.png')
plt.clf()
```

Annex 1: kmeans_algo.py (continued)

```
# Plot 1-3
plt.scatter(X[:,1], X[:,3], marker='.', s=500, lw=0, alpha=0.7, c=colors)
plt.xlabel(y[1])
plt.ylabel(y[3])
plt.scatter(centers[:, 1], centers[:, 3], marker='o', c='white', alpha=1,
s=500)
for i, c in enumerate(centers):
    plt.scatter(c[1], c[3], marker='$%d$' % i, alpha=1, s=50)
plt.scatter(average_X[1], average_X[3], marker='x', c='black', alpha=1, s=300)
plt.savefig('1-3.png')
plt.clf()

# Plot 4-9
plt.scatter(X[:,4], X[:,9], marker='.', s=500, lw=0, alpha=0.7, c=colors)
plt.xlabel(y[4])
plt.ylabel(y[9])
plt.scatter(centers[:, 4], centers[:, 9], marker='o', c='white', alpha=1,
s=500)
for i, c in enumerate(centers):
    plt.scatter(c[4], c[9], marker='$%d$' % i, alpha=1, s=50)
plt.scatter(average_X[4], average_X[9], marker='x', c='black', alpha=1, s=300)
plt.savefig('4-9.png')
plt.clf()

# Plot 5-10
plt.scatter(X[:,5], X[:,10], marker='.', s=500, lw=0, alpha=0.7, c=colors)
plt.xlabel(y[5])
plt.ylabel(y[10])
plt.scatter(centers[:, 5], centers[:, 10], marker='o', c='white', alpha=1,
s=500)
for i, c in enumerate(centers):
    plt.scatter(c[5], c[10], marker='$%d$' % i, alpha=1, s=50)
plt.scatter(average_X[5], average_X[10], marker='x', c='black', alpha=1,
s=300)
plt.savefig('5-10.png')
plt.show()
```

Annex 3: guide_rank.py

```
import numpy as np

rank = np.loadtxt('rank.txt', delimiter="\t")

# Create quartile 4 x 4 array
quartile = np.zeros(shape=(4,4))

# Iterate through all guides for each quartile and cluster
for q in range(4):
    for c in range(4):
        count = 0
        for g in range(len(rank)):
            if rank[g][1] == c and rank[g][0] < (25.0 + q * 25.0) and
rank[g][0] >= (q * 25.0):
                count = count + 1
            if q == 4 and rank[g][1] == c and rank[g][0] == 100:
                count = count + 1
            if q == 0 and rank[g][1] == c and rank[g][0] == 0:
                count = count + 1
        quartile[c][q] = count

np.savetxt('prediction.txt', quartile)
```