Math 444: Project 6

Levent Batakci

April 23, 2021

In this project, we leveraged text mining techniques to analyze genetic data. All of the mentioned implementations can be found on the public github repo.

Genetic Data Background

In this project, we perform an analysis of genetic data. Specifically, we analyze a fragment of the genomic sequence of the bacterium *Caulobacter Crescentus.*

The genomic sequence is a string of letters representing the 4 nucleotides that comprise DNA. The letters A, C, G, and T are used to represent Adenine, Cytosine, Guanine, and Thymine, respectively. The fragment we worked with consists of 305400 letters.

A distinctive message in DNA is called a gene. Moreover, the biological information is encoded by means of 3-letter sequences - called codons. Noting that there are 4 letters in the relevant alphabet, there are $4^3 = 64$ different codons.

This elementary understanding of genetic sequences enables us to apply text mining techniques to this data.

Text Mining Background

As mentioned, we decided to use text mining techniques to analyze the genetic sequence at hand. Particularly, we were interested to see if text-mining techniques could be used to separate genes.

We used term-document matrices as a basis for our analysis. Essentially, a term-document matrix is a matrix in which each column corresponds to a document (collection of words). The entries of the column correspond to frequencies of words. For instance, a 5 in the third row of column 10 would indicate that word 3 appears 5 times in document 10.

It is possible to weight these matrices to give a heightened importance to rare words. Our code provides capability to weight the matrices such that words frequencies are weight by the quantity $log(\frac{N}{N_i}) + 1$, where N is the number of documents and N_i is the number of documents in which word i occurs. Ultimately, we did not weight the term-document matrices, as weighting did not change much. This partially makes sense.

Once this matrices are constructed, we are able to apply familiar techniques to analyze them. Specifically, we will be using PCA to visualize and k-medoids to cluster the data in these matrices.

An important point to discuss is how to turn a genetic sequence into term-document data. Particularly, we need to determine a way to split up the sequence into documents, and a way to define terms ("words"). Seeing as genes are contiguous segments of the sequence, we decided that documents should correspond to fixed-length chunks of the sequence. Moreover, we decided that each term should consist of a fixed number of consecutive letters.

In the following sections, we explore the effects of different choices for term and document length. It should be noted that since codons consist of 3 nucleotides, we should expect good results when setting the term length to 3.

Effects of Term and Document Length

As a first part of the analysis, we decided to explore the effects of different term-length and document-length parameters. Note again that documents essentially correspond to genes, and terms correspond to fixed-length consecutive strings of letters in the sequence.

To start things off, we decided to set document length $k = 200$. That is, we chunked the genomic sequence into segments of length 200. From there, we tested term length $1 \leq n \leq 6$. Note that there are 4^n words of length n since there are 4 letters in the genetic alphabet. As discussed, we expected to find the best results from $n = 3$ since the number of letters in a codon is 3.

In total, we computed 6 term document matrices. For each, we performed a principal component analysis and created scatter plots corresponding to the first two principal components. Of course, we centered the data when performing the PCAs. This result can be seen in **Figure 1**.

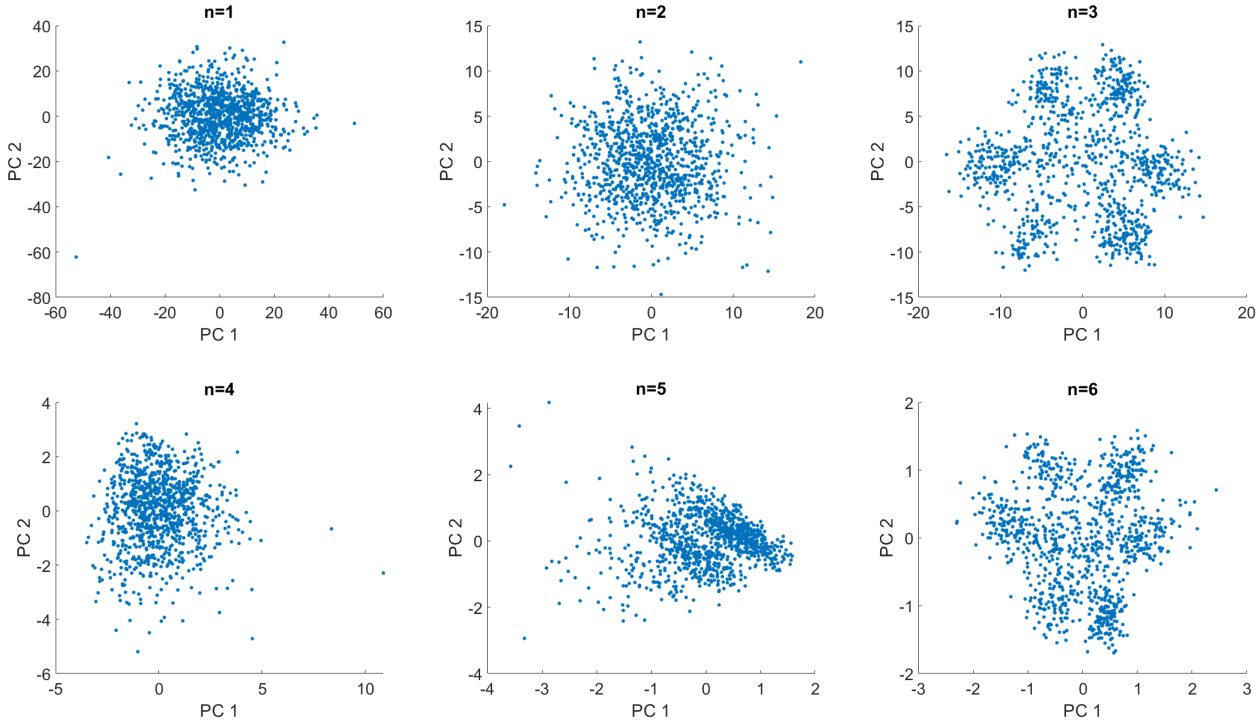

Figure 1: PCAs with document length $k = 200$ and term length n

In these PCAs we're looking for indication of a natural clustering. Immediately, we notice that $n = 3$ seems to have a fairly clear clustering. This makes sense for the reasons involving codon-length discussed before.

The next best PCA is the one corresponding to $n = 6$. In this one, there also appears to be a clustering, although one less apparent. It does make sense that $n = 6$ gives OK results, as it's essentially looking at pairs of codons!

Ultimately, we decided to proceed with $n = 3$ for the rest of our analysis. It makes sense from our knowledge of codons and from our PCA results to choose this value.

The next step we took was to vary the document length with fixed term length $n = 3$. Specifically, we tested document lengths $k = 200, 400$, and 800. Again, we performed a PCA each time and plotted the first two principal components. The resulting graphic is seen in **Figure 2**.

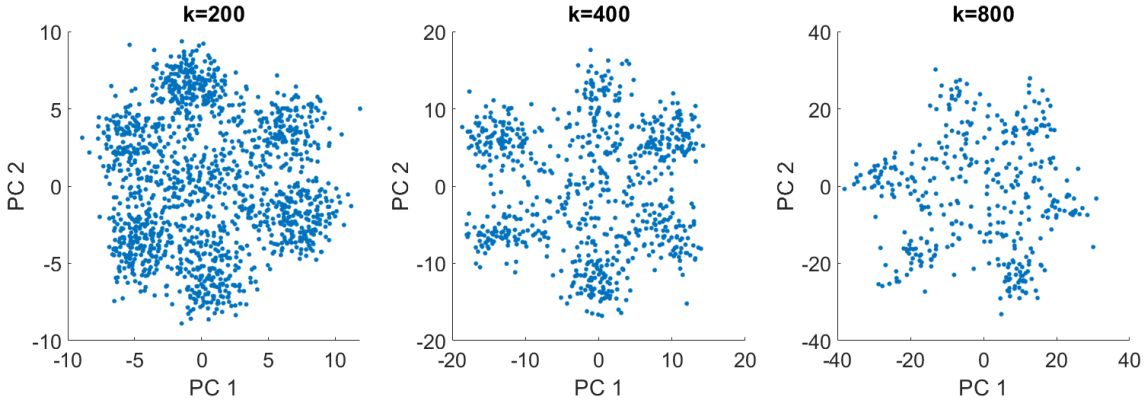

Figure 2: PCAs with term length $n = 3$ and document length k

The first thing to note is that higher document lengths mean that we will have less documents. This is why the PCA corresponding to $k = 800$ has far less (about $\frac{1}{4}$ as many) points as the one corresponding to $k = 200$.

Then, it is clear that the while all of the PCAs here have apparent clustering, the clustering is clearest in the scatterplot corresponding to $k = 400$. This suggests that $k = 400$ is likely closer to the actual length of genes.

Moreover, there appear to be 6 clusters. In the next section, we will test this with the k-medoids algorithm.

Clustering

In the previous section, we determined that term-document matrix corresponding to term length $n = 3$ and document length $k = 200$ appears to have 6 clusters. To further test this, we decided to apply the k-Medoids algorithm. Please note that the k in k-Medoids has nothing at all to do with document length, which is also denoted by k. The results of the clustering can be seen in **Figure 3** on the next page. The circles represent the cluster medoids.

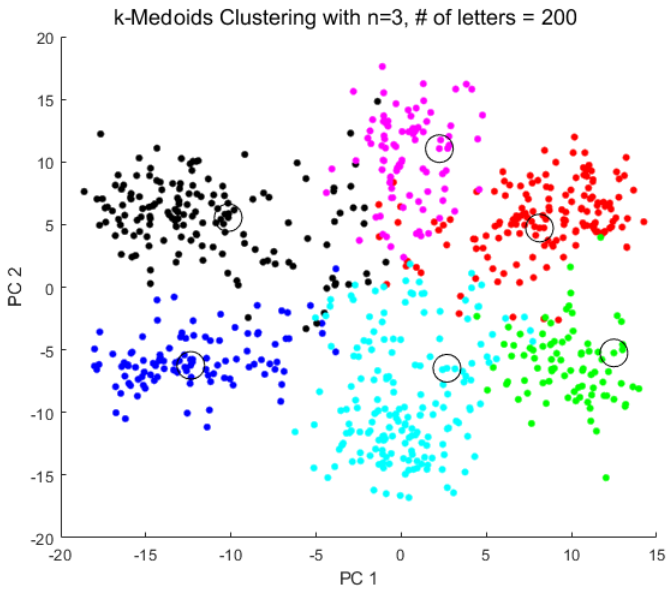

Figure 3: k-Medoids clustering with 6 clusters

We see that indeed, the algorithm does a rather good job clustering the data into 6 groups. Recall that this clustering is on data which corresponds to documents. Moreover, documents are meant to correspond to genes in the genomic sequence. So the implications of the above analysis are that the genes in the sequence can likely separated into 6 different categories.

One thing to notice is that the clustering almost seems angular. That is, from the center, the clusters seem to be distributed at ranges of angles about the origin. This leads into the next part of our analysis, where we apply query-matching techniques.

Query Matching

Query-matching is essentially a way to compute how well the vectors in a data set match an input query vector. Moreover, an important property of query-matching is that scale does not matter. That is, a vector v and αv match a query q equally well for every positive α.

Indeed, this property is clear from how the matches are computed. We compute the match with the j-th data point to be

$$m_j = \frac{q^T x^{(j)}}{||x^{(j)}||||q||}.$$

In some sense, it represents the similarity of directions. More commonly, it can be thought of as the angle between two vectors. This is precisely why the angular distribution of clusters led us to explore query matching with this data.

We decided to use one of the medoids - specifically the blue cluster's - as the query vector. Then we queried the data in the blue cluster and the data not in the blue cluster. We expect to see a difference in the distribution in how well the data matches the query across these two groups. The results are shown in a histogram in **Figure 4** on the next page.

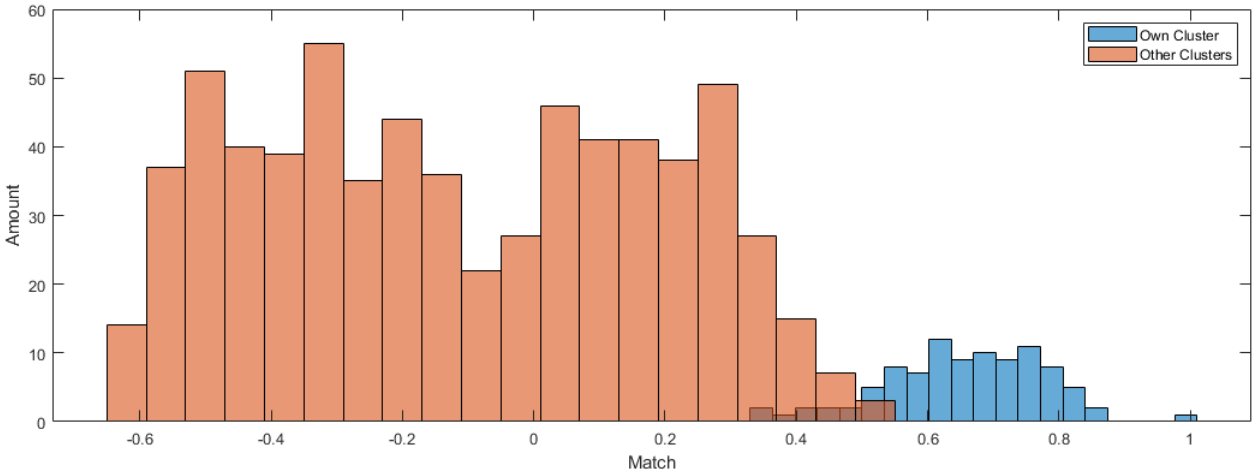

Figure 4: Histogram Results of Query-Matching

Immediately, it's apparent that the distributions are very different. Indeed, the high majority of points in the blue cluster match the medoid better than any point in any other cluster! This results confirms that the clustering is distributed in an angular fashion.

The implications of this require a deeper understanding of biology, but we have some stipulations. For instance, we think its possible that this result indicates that genes are grouped into categories by the relative frequencies of codons. To our non-biologically oriented minds, this makes sense. Just as in text documents, the relative frequencies of words are a better indicator of subject than absolute frequencies.

Appendix

As always, feel free to access the code at the public github repo.

1 Analysis Code

```
%Levent Batakci
%HW 6 Main Script
%Text Mining!

%Load the data
%Precomputed, by me!
load genseq.mat
load termdocs.mat

%CENTER!
for i = 1:6
    %[TermDocs{i} W] = WeightTD(TermDocs{i}); %WEIGHTING
    TermDocs{i} = TermDocs{i} - sum(TermDocs{i},2) / size(TermDocs{i},2);
end
```

```
% Scatter plots
clf(figure(1));
figure(1)
for n = 1:6
    subplot(2,3, mod(n-1,3)+1 + 3*floor((n-1)/3));
    %Visualize
    Z = PCA(TermDocs{n});
    Z1 = Z(1,:);
    Z2 = Z(2,:);
    scatter(Z1,Z2, 200, '.');
    xlabel("PC 1");
    ylabel("PC 2");
    title("n=" + n);
    set(gca,"FontSize",20);
    hold on;
end

load TD;
%K-medoids for k=200, n=3, clusters=6
k=6;
tau=0.0001;
D = dMatrix(TD, @norm2);
[I,Ic] = kMedoids_distMatrix(k, D, tau, 10);

%PCA visualization!
Z = PCA(TD);
colors = ["blue" "black" "red" "green" "cyan" "magenta" "yellow"];
clf(figure(7));
figure(7);
sgtitle("k-Medoids Clustering with n=3, # of letters = 200")
for c = 1:k
    %Visualize
    Z_ = Z(:,I==c);
    Z1 = Z_(1,:);
    Z2 = Z_(2,:);
    scatter(Z1,Z2, 200, colors(c), '.' );
    hold on;
end
xlabel("PC 1");
ylabel("PC 2");

for c = 1:k
    scatter(Z(1,Ic(c)),Z(2,Ic(c)), 300, "black", 'o');

end

clf(figure(8))
TD = TD-sum(TD,2)/size(TD,2); %CENTER!
M = Query(TD(:, Ic(1)),TD(:,I==1));
figure(8);
histogram(M, 20)
hold on
M = Query(TD(:, Ic(1)),TD(:,I~=1));
```

```
histogram(M, 20)
xlabel("Match");
ylabel("Amount");
legend("Own Cluster", "Other Clusters");


clf(figure(10))
figure(10)
K = [200 400 800];
for i = 1:size(K,2)
    k = K(i);

    subplot(1,3, mod(i-1,3)+1 + 3*floor((i-1)/3));
    %Visualize
    Z = PCA(CalcFreq(genseq, 3, k)');
    Z1 = Z(1,:);
    Z2 = Z(2,:);
    scatter(Z1,Z2, 200, '.');
    xlabel("PC 1");
    ylabel("PC 2");
    title("k=" + k);
    set(gca,"FontSize",20);
    hold on;
end
```

2 Query Code

```
function M = Query(q,data)
%query matching algorithm

[~,p] = size(data);
q = q / norm(q);

M = zeros(1, p);
for j = 1:p
    d = data(:,j);
    M(j) = (q' * d) / norm(d);
end

end
```