Topics in Bioinformatics
Problems sample

P1 Write a Python code for the k-means clustering algorithm; for a detailed description of the algorithm, see [1]; for large parts of the code, see here (you can also find the code in the appendix below); here is a list of problems to solve:

i. find the assignment and update steps from the algorithm in the computer code below; which of these two is somewhat incorrectly done?

ii. write the correct code

iii. prove the assertions from [1], that is, show that the assignment and update steps decrease the value of the objective function

iv. show that this decrease is optimal, that is, that the algorithm is greedy

P2 Write an algorithm for the optimal alignment of two protein sequences; in other words, implement Needleman-Wunsch algorithm; substitution matrix [2], the list of amino acids [3] (see [4] for a description; a large part of the code is here); you can also find the first part of the code below; problems to solve:

i. the only part missing is the traceback procedure; add it to the code

P3 Write an algorithm for the optimal LOCAL alignment of two protein sequences (that is the Smith-Waterman algorithm, see [5]); problems to solve:

i. write down the differences between nw and sw algorithms

ii. modify the code below to obtain the sw-algorithm

P4 Carry out a simulation: generate 1000 protein sequences of a fixed length (say, 200 aa); now, apply the above local alignment algorithm to this file, with a query, say, x="HPEW", or something (you're only interested in the score of the optimal local alignment, not the alignment itself); save the scores into a file, and check that the scores are approximately Gumbel distributed

References

[1] `k_means_obj.pdf`

[2] `blosum50.txt`

[3] `acids.txt`

[4] `https://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm`

[5] `https://en.wikipedia.org/wiki/Smith-Waterman_algorithm`

1 Appendix

```
%%%% k-means code

import random
import math
def sed(x,y): # square of euclidean distance
dist=0
for i in range(len(x)):
dist = dist + (x[i]-y[i])*(x[i]-y[i])
return dist
def sad(x,y): # simple addition of vectors
        v=[]
        for i in range(len(x)):
                v.append(x[i]+y[i])
        return v
f1=open("data002.txt", "r")
f2=open("res204.txt", "w")
noc=int(f1.readline()) # broj centara
nopt=int(f1.readline()) # broj tocaka
dimpt=int(f1.readline()) # dimenzija tocaka
noi=15 # broj iteracija
x=[]
for i in range(nopt):
line=f1.readline() # citam jedan redak iz datoteke
vector=line.split() # splittam redak u listu
x.append(vector) # appendam listu u matricu
for i in range(nopt):
        for j in range(dimpt):
                x[i][j]=float(x[i][j])
cent=[]
for i in range(noc): # biramo potrebni broj slucajnih centara
        j=random.randint(0, nopt-1)
        cent.append(x[j][:])
gid=[]
for i in range(nopt): # punim listu gid (=group indicator), da bih je poslije mogao samo mijenjati
        gid.append(0)


# main loop, repeated "noi" times
for temp in range(800,1,-2):
        # odredjivanje clustera
```

```
        for i in range(nopt):
                c=sed(x[i],cent[0])
                suma=tmp_dist=c
                gid[i]=0
                for j in range(1,noc):
                        c=sed(x[i],cent[j])
                        suma=suma+c
                        if c < tmp_dist:
                                gid[i]=j
                                tmp_dist=c
                gid2=random.randint(0,noc-1)
                a=(1.0*temp/200.0)*(1.0*temp/200.0)*(noc*tmp_dist/suma)
                b=random.random()
                if b<a:
                        gid[i]=gid2
        # kraj odredjivanja clustera
        # odredjivanje novih centara
        for i in range(noc):
                v=[]
                for j in range(dimpt):
                        v.append(0)
                cl_count=0
                for j in range(nopt):
                        if gid[j]==i:
                                cl_count=cl_count+1
                                v=sad(v,x[j])
                if cl_count>0:
                        for j in range(dimpt):
                                cent[i][j]=v[j]*1.0/cl_count
        # kraj odredjivanja novih centara
        # ispis u file
        obj=0
        for i in range(nopt):
                obj=obj+sed(x[i],cent[gid[i]])
        f2.write(' %s %3d %20.10f \n' % ('iteration', temp,obj))
f1.close()
f2.close()

%%%%% nw-code


f1=open("acids.txt", "r") ## reading acids
```

```
ak=f1.readline()
f1.close()

bm=[] ## reading matrix
f1=open("blosum50.txt", "r")
for i in range(20):
    line=f1.readline()
    vc=line.split()
    bm.append(vc[:])
for i in range(20): ## integer matrix
    for j in range(20):
        bm[i][j]=int(bm[i][j])

x='HPEW'
y='PW'
m=len(x)
n=len(y)

sm=[] ## score matrix
tmp=[]
for i in range(m+1):
    tmp.append(0)
for i in range(n+1):
    sm.append(tmp[:])

## rubni uvjeti
for i in range(n+1):
    sm[i][0]=-8*i
for i in range(m+1):
    sm[0][i]=-8*i

## rekurzija
for i in range(1,n+1):
    for j in range(1,m+1):
        tmp=[]
        tmp.append(sm[i-1][j]-8)
        tmp.append(sm[i][j-1]-8)
        bb=bm[ak.index(y[i-1])][ak.index(x[j-1])]
        tmp.append(sm[i-1][j-1]+bb)
        tmp.sort()
        sm[i][j]=tmp[2]
```