FACULTY OF
BIOSCIENCE ENGINEERING

HYPERDIMENSIONAL COMPUTING FOR PROTEIN LANGUAGE MODELING

...

Michael Fatjanov
Student ID: ...

Supervisor(s): Prof. Dr. Bernard De Baets

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of master in Bioinformatics.
Academic year: 2022 - 2023

GHENT
UNIVERSITY

Gent, FILL IN THE DATE

The promotor,

Prof. Dr. Bernard De Baets

The author,

Michael Fatjanov

ACKNOWLEDGEMENTS

Thank you, all of you!

CONTENTS

SAMENVATTING

nederlandse samenvatting

SUMMARY

insert english summary here...

1. INTRODUCTION

1.1 Digital biology, protein sequence research and traditional bioninformatics tools

1.2 State-of-the-art, deep learning and protein language modeling

1.3 Hyperdimensional computing

BIBLIOGRAPHY

APPENDIX A

MODEL DEVELOPMENT CODING

A.1 Pseudocode of the presented algorithm

Algorithm 1: How to write algorithms

Data: this text

Result: how to write algorithm with LaTeX2e

initialization;

while *not at end of this document* **do**

    read current;

    if *understand* **then**

        go to next section;

        current section becomes this one;

    else

        go back to the beginning of current section;

    end

end

A.2 Sensitivity base class code

```
1 import os
2 import numpy as np
3
4 from parameter import *
5 import matplotlib.pyplot as plt
6 from matplotlib.ticker import FixedLocator, MaxNLocator
7
8 class SensitivityAnalysis(object):
9     """
10    Base class for the Sensitivity Analysis
11
12    Parameters
```

```
13      ----------
14      ParsIn : list
15          ModPar class instances in list or list of (min,max,'name')-
               tuples
16
17      Attributes
18      -----------
19      ParsIn : list
20          a list of (min,max,'name') values,
21          [(min,max,'name'),(min,max,'name'),...(min,max,'name')]
22      parmap : dict
23          tracks the sequence of the parameters
24      Pars : list of ModPar instances
25          Used when working with the pyFUSE package
26      ndim :  int
27          number of uncertain input factors
28      namelist : list
29          list of the uncertain input factors used
30
31      """
32
33      def __init__(self,ParsIn):
34          '''
35          Check if all uniform distribution => TODO ! if all -> sobol
               sampling
36          is possible, else, only uniform and normal distribution are
               supported
37          for using the sobol sampling... Here is still work to do!!
38          '''
39
40          if  isinstance(ParsIn, dict): #bridge with pyFUSE!
41              dictlist = []
42              for key, value in ParsIn.iteritems():
43                  dictlist.append(value)
44              ParsIn = dictlist
45              print ParsIn
46
47          #control for other
48          self.ParsIn = ParsIn
49          self.parmap={} #dictionary linking ID and name, since dict
               instance has no intrinsic sequence
50          for i in range(len(ParsIn)):
51              if isinstance(ParsIn[i], ModPar): #or isinstance(ParsIn[i],
                   pyFUSE.parameter.ModPar):
52                  cname = ParsIn[i].name
53                  self.Pars = ParsIn
54                  self.ParsIn[i] = (ParsIn[i].min, ParsIn[i].max, cname)
```

```
55                self.parmap[i] = cname
56
57            elif isinstance(ParsIn[i],tuple):
58                if ParsIn[i][0] > ParsIn[i][1]:
59                    raise Exception('Min value larger than max value')
60                if not isinstance(ParsIn[i][0],float) and isinstance(
                      ParsIn[i][1],float):
61                    raise Exception('Min and Max value need to be float'
                          )
62                if not isinstance(ParsIn[i][2],str):
63                    raise Exception('Name of par needs to be string')
64                self.parmap[i] = ParsIn[i][2]
65                #create modpar instance of the tuple
66                self.Pars=[]
67                for par in ParsIn:
68                    self.Pars.append(ModPar(par[2],par[0],par[1],(par
                          [0]+par[1])/2.,'randomUniform'))
69            else:
70                raise Exception('The input type for sampling not correct
                      ,\
71                choose ModPar instance or list of (min,max)-tuples')
72
73        self.ndim=len(ParsIn)
74
75        self.namelist = []
76        for i in range(self.ndim):
77            self.namelist.append(self.parmap[i])
78
79    def WritePre(self,filename = 'inputparameterfile', *args, **kwargs):
80        '''
81        Parameterinputfile for external model, parameters in the columns
              files
82        and every line the input parameters
83
84        Parameters
85        ----------
86        filename : str
87            name of the textfile to save
88        *args, **kwargs : args
89            arguments passed to the numpy savetext-function
90        '''
91
92        np.savetxt(filename,self.parset2run,*args,**kwargs)
93        print 'file saved in directory %s'%os.getcwd()
94
95    def ReadRuns(self,filename, *args, **kwargs):
96        '''
```

```
97          Read model outputs (TODO: do sobol for multiple outputs,
               iterating the
98          post)
99          Format is: every output of the ithe MC on ith line
100
101         output2evaluate can also be made on a other way
102
103         Parameters
104         -----------
105         filename : str
106             name of the textfile to load
107         *args, **kwargs : args
108             arguments passed to the numpy loadtext-function
109
110         '''
111         self.output2evaluate = np.loadtxt(filename, *args, **kwargs)
```

A.3 Model input file for PyFUSE model

```
1  ######################################################
2  ##    Model Parameter input file
3  ##    The parameter is defined by his distribution,
4  ##    boundaries and extra info needed by distribution
5  ##    provide on each line one parameter with
6  ##    following information:
7  ##    name : string
8  ##        Name of the parameter
9  ##    minval : float
10 ##        Minimum value of the parameter distribution
11 ##    maxval :  float
12 ##        Maximum value of the parameter distribution
13 ##    optguess : float
14 ##        Optimal guess of the parameter, must be
15 ##        between min and max value
16 ##    pardistribution : string
17 ##        choose a distributionfrom: randomUniform,
18 ##        randomTriangular, randomTrapezoidal,
19 ##        randomNormal, randomLogNormal
20 ##    *kargs  :
21 ##        Extra arguments necessary for the
22 ##        chosen distribution
23 ##  Lines with ## marks are neglected
24 ######################################################
25 ## NAME MIN MAX OPTGUESS DISTRIBUTION ARGS*
26 S1max 50. 5000.000 400. randomTriangular 1000.
```

```
S2max 100. 10000.000 1000. randomNormal 500. 25.
fitens 0.01 1.0 0.99 randomLogNormal  0.5 0.2
firchr 0.050 0.950 0.5 randomTrapezoidal 0.4 0.6
fibase 0.050 0.950 0.5 randomUniform
r1 0.050 0.950 0.5 randomUniform
ku 0.01 1000. 0.044 randomUniform
c 0.99 20.0 1. randomUniform
alfa 1.000 250. 150. randomUniform
psi 1.000 5.0 2.5 randomUniform
kappa 0.050 0.950 0.5 randomUniform
ki 0.001 1000. 0.00833 randomUniform
ks 0.001 10000. 0.5 randomUniform
n 1.000 10. 3. randomUniform
v 0.00001 0.250 0.004 randomUniform
vA 0.001 0.250  0.0015 randomUniform
vB 0.001 0.250 0.0015 randomUniform
Acmax 0.050 0.950 0.5 randomUniform
b 0.001 3.0 0.2 randomUniform
loglambda 5.000 10.0 7.5 randomUniform
chi 2.000 5.0 3.5 randomUniform
mut 0.010 5.0 0.6 randomUniform
be 0.99 4. 3.1 randomUniform
alfah 0.01 0.99 0.5 randomUniform
tg 0.0 0.7 0.3 randomUniform
tif 0.0 0.7 0.26 randomUniform
tof 0.0 0.7 0.12 randomUniform
ko 0.01 0.99 0.15 randomUniform
timeo 2. 48 24 randomUniform
timei 2. 250. 20 randomUniform
timeb 200. 10000. 2100. randomUniform
```