ALG5I - Übung 04

Alen Kocaj

09. November 2017

1 Position Specific Scoring

Implementieren Sie in einer Programmiersprache Ihrer Wahl ein Framework für positionsspezifisches Scoring

Die Struktur der Abgabe besteht aus den folgenden Files.

```
/root
├── main.py
├── pss.py
└── pss_sequence_logo.py
```

Spicy jalapeno bacon ipsum dolor amet brisket burgdoggen turducken ground round turkey landjaeger salami chicken tenderloin bacon. Ground round alcatra pork belly kevin, beef chicken spare ribs salami short ribs shankle beef ribs. Tail landjaeger alcatra doner tenderloin, jowl meatball jerky shankle brisket andouille beef cupim spare ribs. Jerky kevin shank flank doner kielbasa boudin alcatra hamburger cow pastrami. Filet mignon beef capicola picanha short loin ribeye meatball corned beef shankle chuck chicken buffalo.

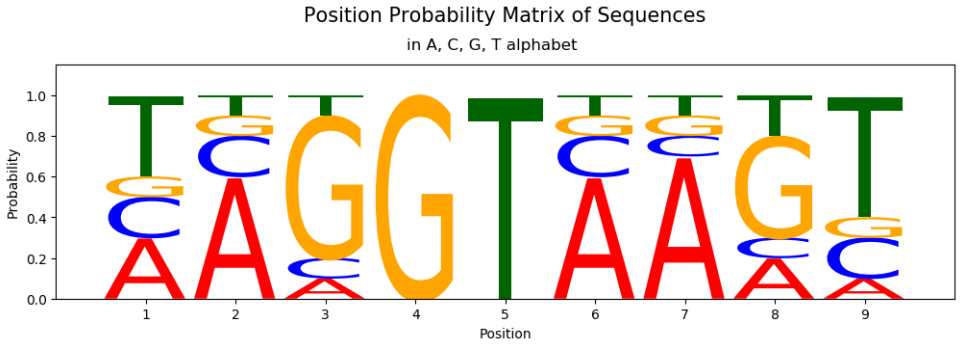

Abbildung 1: Die Position Probability Matrix (PPM) als Sequenzlogo geplottet. Man sieht den Anteil an C,G,T,A basierend auf der Größe des Buchstaben.

Landjaeger ribeye fatback, short ribs pork belly short loin doner. Venison shankle swine pork chop tri-tip. Landjaeger kielbasa ball tip t-bone, shoulder jowl tongue hamburger sausage. Sirloin t-bone cow bacon burgdoggen biltong ribeye filet mignon. Tongue swine cow jerky, venison ground round buffalo chuck bacon turducken leberkas ribeye meatball ham hock strip steak. Ham chicken pig shoulder andouille.

```
======================================================================
Position Specific Scoring (PSS)
    by Alen Kocaj

[i] reading source alignments out of 'testing/source'
[i] reading target sequences out of 'testing/target'
======================================================================
[PPM] Position Probability Matrix
     0    1    2             3             4    5    6    7    8
A  0.3  0.6  0.1  1.000000e-10  1.000000e-10  0.6  0.7  0.2  0.1
C  0.2  0.2  0.1  1.000000e-10  1.000000e-10  0.2  0.1  0.1  0.2
G  0.1  0.1  0.7  1.000000e+00  1.000000e-10  0.1  0.1  0.5  0.1
T  0.4  0.1  0.1  1.000000e-10  1.000000e+00  0.1  0.1  0.2  0.6

[i] plot sequence logo of PPM
[i] figure saved at 'pss_ppm.png'
======================================================================
[PPM] Probability Weight Matrix
          0         1         2          3          4         5         6  \
A -1.203973 -0.510826 -2.302585 -23.025851 -23.025851 -0.510826 -0.356675
C -1.609438 -1.609438 -2.302585 -23.025851 -23.025851 -1.609438 -2.302585
G -2.302585 -2.302585 -0.356675   0.000000 -23.025851 -2.302585 -2.302585
T -0.916291 -2.302585 -2.302585 -23.025851   0.000000 -2.302585 -2.302585

          7         8
A -1.609438 -2.302585
C -2.302585 -1.609438
G -0.693147 -2.302585
T -1.609438 -0.510826
======================================================================
[i] scoring targets

# [0th target] #####
> Sequence: GAGGTAAAC
> Log Score by PSS: -7.25646205327

# [1th target] #####
> Sequence: TCCGTAAGT
> Log Score by PSS: -6.89978710933

# [2th target] #####
> Sequence: CAGGTTGGA
> Log Score by PSS: -10.0778409397

# [3th target] #####
> Sequence: ACAGTCAGT
> Log Score by PSS: -8.28608147045

# [4th target] #####
> Sequence: TAGGTCATT
> Log Score by PSS: -5.87016769215

# [5th target] #####
> Sequence: TAGGTACTG
```

```
> Log Score by PSS: -8.50922502177

# [6th target] #####
> Sequence: ATGGTAACT
> Log Score by PSS: -7.54414412572

# [7th target] #####
> Sequence: CAGGTATAC
> Log Score by PSS: -8.50922502177

# [8th target] #####
> Sequence: TGTGTGAGT
> Log Score by PSS: -9.38469375912

# [9th target] #####
> Sequence: AAGGTAAGT
> Log Score by PSS: -4.14294674406

======================================================================
[i] thank you and goodnight
=
```

Listing 1: Der Konsolenoutput des PSS Programms. Als Zielsequenzen zum Berechnen eines Scores wurde dasselbe File verwendet wie zum Einlesen des alignierten Sequenzen.

```
#!/usr/bin/env python
#
# Position Specific Scoring (PSS)
#   by Alen Kocaj

from functools import reduce

from matplotlib import pyplot as plt

import pss_sequence_logo
from pss import PSS

def main():
    # source of aligned sequences being used to
    # build scoring matrix
    source_scoring_file = "testing/source"

    # list of target sequences which should be scored
    target_scoring_file = "testing/target"

    # path where sequence logo of PPM should be saved
    path = "pss_ppm.png"

    # the random model used
    weights = {
        "A": 0.25,
        "C": 0.25,
        "G": 0.25,
        "T": 0.25
    }

    # pseudocount for preventing zero probabilities
    pseudocount = 0.0000000001

    print("
       ========================================================================="
       )
    print("Position Specific Scoring (PSS)")
    print("\tby Alen Kocaj")
    print()

    # read source alignments
    print(f"[i] reading source alignments out of '{source_scoring_file}'")
    sources = []
    with open(source_scoring_file) as sourcefm:
        sources = sourcefm.read().split("\n")

    expected_length = len(sources[0]) * len(sources)
    observed_length = reduce(lambda acc, curr: acc + len(curr), sources, 0)

    # validate sources
    avg_length = int(expected_length / len(sources))
    if observed_length != expected_length:
        print(f"[w] not all sequences are of same length. Expected length: {
```

```
            avg_length}")
53          print("The overhanging onegram will not be considered.")
54
55      # read target sources
56      print(f"[i] reading target sequences out of '{target_scoring_file}'")
57      targets = []
58      with open(target_scoring_file) as targetfm:
59          targets = targetfm.read().split("\n")
60
61      # build matrices
62      pss = PSS(sources, weights.keys(), weights, avg_length, pseudocount)
63      pfm = pss.build_frequency_matrix()
64      ppm = pss.build_probability_matrix(pfm)
65
66      print("
            ======================================================================"
            )
67      print("[PPM] Position Probability Matrix")
68      print(ppm)
69      print()
70      print("[i] plot sequence logo of PPM")
71      plot_ppm_sequence_logo_pd(ppm, path, 1.15)
72      print(f"[i] figure saved at '{path}'")
73
74      # build Postion Weight Matrix (PWM)
75      weight_matrix = pss.build_weight_matrix(ppm)
76      print("
            ======================================================================"
            )
77      print("[PPM] Probability Weight Matrix")
78      print(weight_matrix)
79
80      # score targets
81      print("
            ======================================================================"
            )
82      print("[i] scoring targets\n")
83      for i, target in enumerate(targets):
84          print(f"# [{i}th target] #####")
85          print(f"> Sequence: {target}")
86          print(f"> Log Score by PSS: " + str(pss.score(target, weight_matrix)
                ))
87          print()
88      print("
            ======================================================================"
            )
89
90      print("[i] thank you and goodnight")
91
92      # plot_ppm_sequence_logo_pd plots a Position Probability Matrix (PPM)
93      # as a Sequence Logo (https://en.wikipedia.org/wiki/Sequence_logo).
94      # The ppm is given in a Pandas DataFrame format.
95      #
96      # Kudos to https://github.com/saketkc
```

```
97      # with https://github.com/saketkc/motif-logos-matplotlib for initial
            code.
98  def plot_ppm_sequence_logo_pd(ppm, path, custom_y=-1):
99      fig, ax = plt.subplots(figsize=(10,3))
100
101     x = 1
102     maxy = 0
103     row_indices = list(ppm.index)
104     for column in ppm:
105
106         y = 0
107         for row in range(0, len(ppm)):
108             score = ppm[column][row]
109             base = row_indices[row]
110
111             pss_sequence_logo.letterAt(base, x, y, score, ax)
112             y += score
113         x += 1
114         maxy = max(maxy, y)
115
116     plt.xticks(range(1, x))
117     plt.xlim((0, x))
118
119     if custom_y != -1:
120         maxy = custom_y
121     plt.ylim((0, maxy))
122     plt.tight_layout()
123
124     plt.xlabel("Position")
125     plt.ylabel("Probability")
126     plt.title(f"Position Probability Matrix of Sequences", y=1.15, fontsize
            =15)
127     plt.suptitle("in " + str.join(", ", row_indices) + " alphabet", y=1.03)
128     plt.savefig(f"{path}", bbox_inches="tight")
129
130 if __name__ == "__main__":
131     main()
```

Listing 2: main.py

```python
#!/usr/bin/env python

import numpy as np
import pandas as pd

class PSS():
    def __init__(self, sources, alphabet, weights, avg_sequence_length,
        pseudocount):
        self.sources = sources
        self.alphabet = alphabet
        self.weights = weights
        self.avg_sequence_length = avg_sequence_length
        self.pseudocount = pseudocount

    # build_frequency_matrix computes a position frequence matrix (PFM)
    # out of the given sources. The sequences are based on the given
    # alphabet.
    def build_frequency_matrix(self):
        frequency_matrix = {}
        for onegram in self.alphabet:
            frequency_matrix[onegram] = np.zeros(self.avg_sequence_length)

        for source in self.sources:
            for i in range(0, self.avg_sequence_length):
                onegram = source[i]
                # exclude invalid characters
                if onegram not in frequency_matrix.keys():
                    continue

                frequency_matrix[onegram][i] += 1

        return pd.DataFrame(list(frequency_matrix.values()), dtype=int,
                index=self.alphabet)

    # build_probability_matrix computes a position probability matrix (PPM)
    # based on the given frequency matrix. Zero values are removed
    # by adding a given pseudocount prevent zero-frequency problems.
    def build_probability_matrix(self, pfm):
        probability_matrix = round(pfm / len(self.sources), 2)
        return probability_matrix.clip(self.pseudocount)

    # build_weight_matrix constructs the position weight matrix (PWM)
    def build_weight_matrix(self, ppm):
        return np.log(ppm)

    # score computes a score for a given target sequence, given
    # the position weight matrix (PWM)
    def score(self, target, pwm):
        score = 0
        for i, onegram in enumerate(target):
            score += pwm[i][onegram]
        return score
```

Listing 3: pss.py

```
#!/usr/bin/env python
#
# Kudos to https://github.com/saketkc
# with https://github.com/saketkc/motif-logos-matplotlib

import matplotlib as mpl
from matplotlib.text import TextPath
from matplotlib.patches import PathPatch
from matplotlib.font_manager import FontProperties

globscale = 1.35
LETTERS = { "T" : TextPath((-0.305, 0), "T", size=1),
            "G" : TextPath((-0.384, 0), "G", size=1),
            "A" : TextPath((-0.35, 0), "A", size=1),
            "C" : TextPath((-0.366, 0), "C", size=1) }
COLOR_SCHEME = {'G': 'orange',
                'A': 'red',
                'C': 'blue',
                'T': 'darkgreen'}

def letterAt(letter, x, y, yscale=1, ax=None):
    text = LETTERS[letter]

    t = mpl.transforms.Affine2D().scale(1*globscale, yscale*globscale) + \
        mpl.transforms.Affine2D().translate(x,y) + ax.transData
    p = PathPatch(text, lw=0, fc=COLOR_SCHEME[letter],  transform=t)
    if ax != None:
        ax.add_artist(p)
    return p
```

Listing 4: pss_sequence_logo.py