

Bioinformatics III

Third Assignment

Thibault Schowing (2571837)
Wiebke Schmitt (2543675)

May 1, 2018

Exercise 3.1:

- (a) *Given the states of the features, you want to infer if two proteins are likely to physically interact. In practice, log-likelihood ratios are used in binary classification:*

$$\log \frac{P(C|S)}{P(\bar{C}|S)}$$

Derive a term that uses observable probabilities such as $P(S_i|C)$ to calculate the loglikelihood ratio from training data. How does the actual classification work?

First we have:

$$P(S_i|C) = \frac{P(C|S_i)P(S_i)}{P(C)}$$

And:

$$P(S_i|\bar{C}) = \frac{P(\bar{C}|S_i)P(S_i)}{P(\bar{C})}$$

Then we develop the desired final output

$$\begin{aligned} \frac{P(S_i|C)}{P(S_i|\bar{C})} &\iff \frac{P(S_i|C)P(C)}{P(S_i|\bar{C})P(\bar{C})} = \frac{P(C|S_i)P(S_i)}{P(\bar{C}|S_i)P(S_i)} = \frac{P(C|S_i)}{P(\bar{C}|S_i)} \\ \log \frac{P(C|S)}{P(\bar{C}|S)} &= \log \prod_i^n \frac{P(S_i|C)P(C)}{P(S_i|\bar{C})P(\bar{C})} = \sum_i^n \log \frac{P(S_i|C)P(C)}{P(S_i|\bar{C})P(\bar{C})} = \Lambda(C|S) \\ O(C|S) &= \Lambda(C|S)O(C) \end{aligned}$$

The posterior odd is calculated by the odds of an event ($\frac{p(event)}{1-p(event)}$) multiplied by the likelihood of that event¹.

To do the classification, we must iterate through the data and calculate all the priors and likelihood. The prior $P(C)$ is made from an educated guess

¹Slides V4 - 4

- (b) *Shortly discuss: What are the practical advantages of the logarithm and the likelihood ratio within this framework? State two reasons why this particular type of classifier may perform poorly on a real world dataset.*

The logarithm increase is a monotonically increasing function of x hence, for any positive value the maximum value of a function $f(x)$, the maximum of $f(x)$ is equal to the maximum of $\log(f(x))$. This simplifies the calculation because we don't need the second derivative. A likelihood function is not concave but the log-likelihood is. Also, as seen in part A, with the log-likelihood we can turn a log of products into a sum of logs. The main inconvenient is that this method assume that all the features are independent and do not take in account the eventual correlations between them.

- (c) Use the file *training1.tsv* to build a model. This basically means to determine all necessary priors and likelihoods from part (a). The file layout is explained in *README.txt*. Report $P(C)$ and $P(\bar{C})$ as well as the ten S_i (feature number, variant and log-ratio) with the highest absolute log-likelihood ratios. Examine and comment on the results of the training-phase. Which features seem to be the most helpful?

Prior probability $P(C) = 0.78$

Prior probability $P(\bar{C}) = 0.22$

Table 1: 10 S_i with the highest absolute log-likelihood ratio

| | | |
|----|---|---------------------|
| 33 | 0 | -3.7214026458194964 |
| 11 | 3 | -2.565631943311438 |
| 87 | 1 | -2.4686396773241284 |
| 53 | 1 | -2.3351082846996056 |
| 99 | 1 | -2.3061207478263537 |
| 59 | 1 | -2.2779498708596573 |
| 80 | 2 | -2.2779498708596573 |
| 86 | 3 | -2.2550655770260692 |
| 91 | 3 | -2.2173252490432223 |
| 97 | 1 | -2.2099451417455995 |

Listing 1: bayes.py

```

0 import math
  import copy
  #
  # For all features, compute the probability (prior) to have 0, 1, 2 or 3
    depending on the output (0 or 1)
  #
5 #
  def priors(features, output_indexes):
    priors = {}
    # Start to 1 to match the instructions
    feature_nb = 1
10   for feature in features:
        P_Si_Output = {}
        # Values of the feature for a certain output (0 or 1)
        S = [feature[i] for i in output_indexes]
        # for all possible feature values -> [0,1,2,3], set dynamically here
15     for value in set(feature):
        # Prob of having this 'value' when output is 0 or 1 (depend on
          output_indexes)
        P_Si_Output[value] = S.count(value) / float(len(S))

        priors[feature_nb] = P_Si_Output
20     feature_nb += 1
    return priors

  def log_likelihood(Prior_C, Prior_not_C, P_S_C, P_S_notC):
25     log_like = [[0.0]*4 for _ in range(len(P_S_C))]

    #For each feature
    # Careful, in P_S_C it's a dict -> start at 1 as "feature 1"
    # in log_like it's a list of list -> feature 1 == [0]
30   for feat in P_S_C:
        for val in [0,1,2,3]:
            p = math.log((P_S_C[feat][val] * Prior_C)/P_S_notC[feat][val] *
              Prior_not_C)
            log_like[feat-1][val] = p

```

```

    return log_like
35
def getNMaxLikelihoodRatio(likelihoods, N):
    # As we have to loop N times, we'll need to set the max value to zero
    # in order not to pick it more than once.
    likelihoods_copy = copy.deepcopy(likelihoods)
40    t = []
    for out in range(N):
        # Will contain (feature number, variant, absolute likelihood ratio)
        info = (0,0,0)
        max = 0
45
        for feat in range(len(likelihoods_copy)):
            for val in range(len(likelihoods_copy[feat])):
                if abs(likelihoods_copy[feat][val]) > max:
                    max = abs(likelihoods_copy[feat][val])
50                # Max is calculated with the abs, but the real value is
                    stored
                info = (feat, val, likelihoods_copy[feat][val])
                likelihoods_copy[feat][val] = 0.0

55    t.append(info)
    return t

60 # Read data file

def readFile(filename):
    lines = []
    with open(filename) as f:
65        for line in f:
            line = line.split('\t')
            map(str.strip, line)
            lines.append(line)

70        # Convert all the elements in float instead of chars
        # for line in lines:
        #     line = list(map(float, line))
        lines = [[float(i) for i in line] for line in lines]
    return lines

75    lines = readFile("data/training2.tsv")

    # Number of features
    nb_features = len(lines[0]) - 1
80    print("Nb_features: ", nb_features)

    # Get the data by columns: https://stackoverflow.com/questions/44360162/how-to
    # -access-a-column-in-a-list-of-lists-in-python
    data_columns = list(zip(*lines))
    # Problem, columns are now tuples
85    data_columns = [list(elem) for elem in data_columns]

    # Features only
    features = data_columns[1:]

90 # Output only
    outputs = list(data_columns[0])

    # Indexes according to outputs (1 or 0, first column)
    interaction_indexes = [i for i,x in enumerate(outputs) if x == 1]
95 #print("interact index: ", interaction_indexes)

    no_interaction_indexes = [i for i,x in enumerate(outputs) if x == 0]
    #print("no-interact index: ", no_interaction_indexes)
```

```
100 # Prior probabilities

    Prior_C = outputs.count(1) / float(len(outputs))
    print("Prior_probability_of_having_a_connection:", Prior_C)
    Prior_not_C = 1 - Prior_C
105 print("Prior_probability_of_not_having_a_connection:", Prior_not_C)

    # For each feature and possible value, calculate the probability according to
    # the output

110 # P_S_C = Probability of having S (feature) according to output 1
    P_S_C = priors(features, interaction_indexes)

    # P_S_notC = Probability of having S (feature) according to output 0
    P_S_notC = priors(features, no_interaction_indexes)
115

    # Print every probabilities for every feature's values
    # print("Features's values's probabilities if connection: \n")
    # for p in P_S_C:
    #     print("Feature ", p, ": ")
120 #     for val in P_S_C[p]:
    #         print("\tValue: ", val, " prob: ", P_S_C[p][val])
    #
    # print("Features's values's probabilities if no connection: \n")
    # for p in P_S_notC:
    #     print("Feature ", p, ": ")
125 #     for val in P_S_notC[p]:
    #         print("\tValue: ", val, " prob: ", P_S_notC[p][val])

130 # Now we compute the log likelihood for every features and possible output

    log_like = log_likelihood(Prior_C, Prior_not_C, P_S_C, P_S_notC)

    #print(log_like)
135

    # Get the N (ABSOLUTE) max log-likelihood ratios.

    maxLikelihoods = getNMaxLikelihoodRatio(log_like, 10)

140 # Nice printing
    for _ in maxLikelihoods:
        print(-)

    # Part D
145

    lines = readFile("data/test1.tsv")
    data_columns = list(zip(*lines))
    data_columns = [list(elem) for elem in data_columns]
    features = data_columns[1:]
150 outputs = list(data_columns[0])

    print("Real_test_outputs:")
    print(outputs)

155 prediction = []

    for f in range(len(features)):
        tmp_output = 0
        for v in [0,1,2,3]:
160             tmp_output += log_like[f][v]
        prediction.append(tmp_output)

    print(prediction)
```

Exercise 3.2: Classify real-world network examples

- (a) If one of the nodes has a degree of 1, then $\tilde{C}_{i,j}^{(3)}$ is infinite. What is the maximal finite value that the edge-clustering coefficient can take? For which configuration does this occur? Give an example!
- (b) (1) Give the links that you deleted from the network in (iii) by printing the names of the two nodes and their current edge-clustering coefficient in the order of their deletion. Of course, add the output to the PDF/sheet that you hand in. Implement this part as a script or class-based, there are no specifications you need to adjust to.

Table 2: Order of removed links

| | | |
|---------|----------|----------|
| Tyrion | Sansa | 0.333 |
| Joffrey | Hound | 0.5 |
| Eddard | Robert | 0.5 |
| Eddard | Jon | 0.5 |
| Joffrey | Jaime | 1.0 |
| Hound | Mountain | 1.0 |
| Cersei | Tyrion | 1.0 |
| Jaime | Cersei | 1.0 |
| Catelyn | Baelish | 1.0 |
| Sansa | Baelish | 1.0 |
| Eddard | Catelyn | 1.5 |
| Sansa | Arya | 1.5 |
| Eddard | Sansa | 1.0 |
| Catelyn | Arya | 1.0 |
| Joffrey | Cersei | 2.0 |
| Cersei | Robert | 1.0 |
| Samwell | Jon | 2.0 |
| Joffrey | Robert | ∞ |
| Shae | Tyrion | ∞ |
| Eddard | Arya | ∞ |
| Hound | Arya | ∞ |
| Varys | Baelish | ∞ |
| Jaime | Tyrion | ∞ |
| Cersei | Mountain | ∞ |
| Catelyn | Sansa | ∞ |
| Samwell | Jeor | ∞ |
| Jon | Jeor | ∞ |

- (c) (2)

Table 3: Inclusion of links, "x" means two subgraphs have been merged.

| Link | Merge | Graph |
|-------------------|-------|---|
| Jon - Jeor | | [(Jon - Jeor)] |
| Samwell - Jeor | | [(Jon - Jeor),(Samwell - Jeor)] |
| Catelyn - Sansa | | [(Catelyn - Sansa)] |
| Cersei - Mountain | | [(Cersei - Mountain)] |
| Jaime - Tyrion | | [(Jaime - Tyrion)] |
| Varys - Baelish | | [(Varys - Baelish)] |
| Hound - Arya | | [(Hound - Arya)] |
| Eddard - Arya | | [(Hound - Arya),(Eddard - Arya)] |
| Shae - Tyrion | | [(Jaime - Tyrion),(Shae - Tyrion)] |
| Joffrey - Robert | | [(Joffrey - Robert)] |
| Samwell - Jon | | [(Jon - Jeor),(Samwell - Jeor),(Samwell - Jon)] |
| Cersei - Robert | x | [(Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert)] |
| Joffrey - Cersei | | [(Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei)] |
| Catelyn - Arya | x | [(Catelyn - Sansa),(Hound - Arya),(Eddard - Arya), (Catelyn - Arya)] |
| Eddard - Sansa | | [(Catelyn - Sansa),(Hound - Arya),(Eddard - Arya), (Catelyn - Arya),(Eddard - Sansa)] |
| Sansa - Arya | | [(Catelyn - Sansa),(Hound - Arya),(Eddard - Arya), (Catelyn - Arya),(Eddard - Sansa),(Sansa - Arya)] |
| Eddard - Catelyn | | [(Catelyn - Sansa),(Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn)] |
| Sansa - Baelish | x | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn)] |
| Catelyn - Baelish | | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish)] |
| Jaime - Cersei | x | [(Jaime - Tyrion),(Shae - Tyrion),(Cersei - Mountain), (Joffrey - Robert), (Cersei - Robert),(Joffrey - Cersei)] |
| Cersei - Tyrion | | [(Jaime - Tyrion),(Shae - Tyrion),(Cersei - Mountain), (Joffrey - Robert), (Cersei - Robert),(Joffrey - Cersei), (Cersei - Tyrion)] |

| Link | Merge | Graph |
|------------------|-------|---|
| Hound - Mountain | x | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish),(Jaime - Tyrion),(Shae - Tyrion), (Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei),(Cersei - Tyrion)] |
| Joffrey - Jaime | | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish),(Jaime - Tyrion),(Shae - Tyrion), (Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei),(Cersei - Tyrion),(Joffrey - Jaime)] |
| Eddard - Jon | x | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish),(Jaime - Tyrion),(Shae - Tyrion), (Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei),(Cersei - Tyrion),(Joffrey - Jaime), (Jon - Jeor),(Samwell - Jeor),(Samwell - Jon)] |
| Eddard - Robert | | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish),(Jaime - Tyrion),(Shae - Tyrion), (Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei),(Cersei - Tyrion),(Joffrey - Jaime), (Jon - Jeor),(Samwell - Jeor),(Samwell - Jon), (Eddard - Robert)] |
| Joffrey - Hound | | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish),(Jaime - Tyrion),(Shae - Tyrion), (Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei),(Cersei - Tyrion),(Joffrey - Jaime), (Jon - Jeor),(Samwell - Jeor),(Samwell - Jon), (Eddard - Robert),(Joffrey - Hound)] |
| Tyrion - Sansa | | [(Varys - Baelish),(Sansa - Baelish),(Catelyn - Sansa), (Hound - Arya),(Eddard - Arya),(Catelyn - Arya), (Eddard - Sansa),(Sansa - Arya),(Eddard - Catelyn), (Catelyn - Baelish),(Jaime - Tyrion),(Shae - Tyrion), (Cersei - Mountain),(Joffrey - Robert), (Cersei - Robert), (Joffrey - Cersei),(Cersei - Tyrion),(Joffrey - Jaime), (Jon - Jeor),(Samwell - Jeor),(Samwell - Jon), (Eddard - Robert),(Joffrey - Hound),(Tyrion - Sansa)] |