Machine Learning Handin 3

Erik Holm Steenberg (201804655), Hans Brüner Dein (201706079),
Camilla Theresia Grøn Sørensen (201807007)
Instruktor: Mads Bech Toftrop

November 2022

Introduction

This is the report for the 3rd handin in the computer science course Machine Learning. The handin explains how our Hidden Markov Model works and how we have implemented it. The report also includes the accuracy for the 5-fold validation and the prediction on the 5 unknown annotations.

1 State of the code

It works. It takes quite a while to run all and it is not very accurate, but given that we use the 7 state model and training by counting, this is to be expected.

2 The Hidden Markov Model

We have used the simple 7-state-model, that we implemented in class, together with training by counting. When you have a transition between two states you have two types of probability. Zero and non-zero. Because we are using the "training by counting" method we only get non-zero probabilities for transitions between the different coding, reverse-coding and non-coding states that actually happens. This means we do not run into the potential risk of having non-actual transition probabilities. The non-zero transitions can be seen in fig. 1.

The code we have used to translate the annotations N, C and R into the 7 states is given here:

```
def translate_z_to_indices(path):
    out = []
    forwardcoding = 2
    backwardcoding = 0
    for i in path:
        if i == 'N':
            out.append(3)
        if i == 'C':
            out.append(forwardcoding)
            forwardcoding=(forwardcoding - 1) % 3
        if i == 'R':
            out.append(backwardcoding + 4)
            backwardcoding=(backwardcoding + 1) % 3
    return  out
```

The code always translates N to state 3, a coding triplet to 210 and a reverse coding triplet to 456.

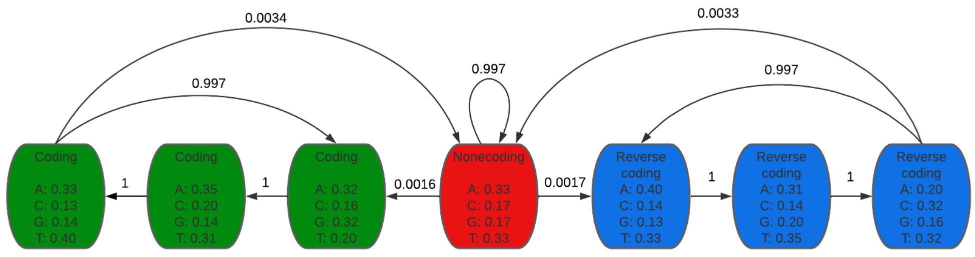

Figure 1: Diagram of our seven state model, with the probabilities we found using training by counting. We notice the the emission probabilities all sum to one, and are mirrored between the coding and reverse coding genes. This suggests the training our model is functioning properly, since the coding and non-coding genes shouldn't be different except their direction.

3 Prediction of the unknown gene structures

We have trained our 7-state-model on the 5 known gene structures using training by counting. We have used the Viterbi algorithm and backtracking to predict the annotations for the five unknown gene structures. We did this by first calculating the ω-matrix and then backtracking to find the hidden states. The last thing we did was translating the hidden states to N, C and R. We did this by using the function given in the theoretical exercises (see below), that translates state 3 to N, states 2, 1 and 0 to C and states 4, 5 and 6 to R:

```
def translate_indices_to_path(indices):
    mapping = ['C', 'C', 'C', 'N', 'R', 'R', 'R']
    return ''.join([mapping[i] for i in indices])
```

4 Accuracy

Here we present the accuracies of our model on the known and unknown genes.
For the known genes we use 5-fold-validation.

Table 1: The accuracy of our model when applied to the known genes.

Genome	1	2	3	4	5
Cs Ac	0.5598	0.5865	0.6097	0.5869	0.6126
Rs AC	0.6254	0.6065	0.6101	0.5828	0.5675
Both AC	0.3756	0.3642	0.4027	0.3562	0.3611

The way we modelled the unknown genes is explained in 3 and the results are as follows.

Table 2: The accuracy of our model when compared with `https://genefinder.fly.dev/`

Genome	6	7	8	9	10
Cs AC	0.5862	0.5849	0.6038	0.5240	0.5236
Rs AC	0.6153	0.5700	0.5658	0.4873	0.5273
Both AC	0.3765	0.3736	0.3625	0.2471	0.2737