Profile HMM for Splice Junctions

By Mark Gorelik

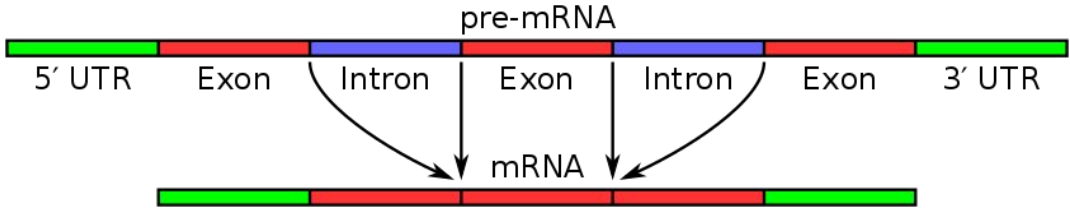
Table of Contents

Problem Definition

- Splice junctions are points on a DNA sequence at which `superfluous' DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons and introns. This problem consists of two subtasks: recognizing exon/intron boundaries, and recognizing intron/exon boundaries. (1) - Can we use a profile HMM to do this?

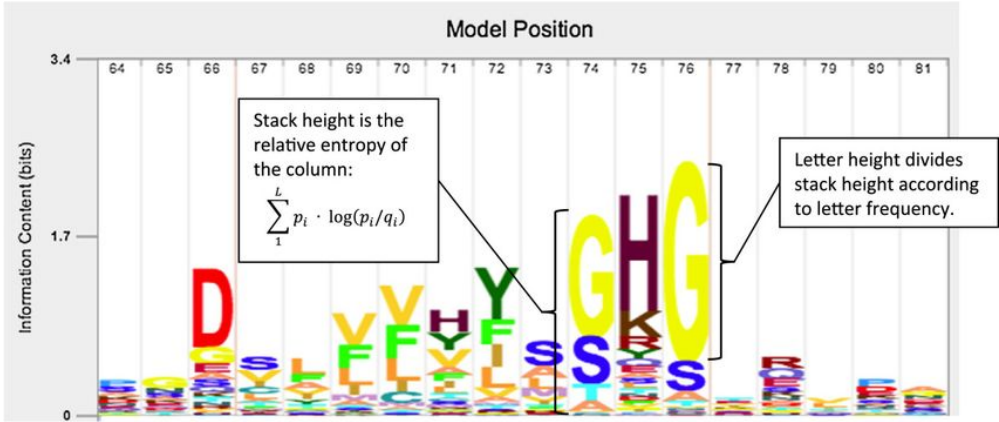
Background - Introns/Exons

- Introns
  - Sequences of a gene that are removed before the gene is translated (1)
- Exons
  - Sequences of a gene that are retained for gene translation (1)
- Why?
  - Alternate Splicing (4)
  - Non coding RNAs (4)
- Splice Junction
  - Boundaries around introns (1)



Background - Profile HMM

- Classification Method
  - Viterbi
  - Statistical representation of sequences
- Most commonly used for protein sequences
  - Also used for DNA/RNA Sequences
- Needs a good amount of training data



Background - Data Structure

- Three States
  - N: Neither/None
  - IE: Intron-Exon Boundary
  - EI: Exon-Intron Boundary
- Five Emissions
  - A, C, T, G, D, N, S, R
- The Table
  - 60 columns
  - 8 rows
  - Represented with a list of dictionaries for each state
- Important notes from the data
  - No state transitions, data is DNA not RNA, positive data comes from primates, negative not specified, no indels

Procedure - Implementation

- Python 3.6
  - No external libraries
- Coded in a Jupyter Lab notebook
  - If you don’t use this but do scientific programming you should check it out
- Dataset from UC Irvine ML Repository as a TSV
- Code is available for judging and critiquing at https://github.com/vzg100/Profile-HMM-CS123B

jupyter

Procedure - High Level

- Not going to copy paste code
- Will walk through the high level design/implementation

Procedure - Columns

- Each state and possible emission were recorded upon reading the file
- The table was a list of dictionaries measuring frequency -> converted to a list of dictionaries measuring percentage in regards to each column
  - Used a pseudocount of 1
  - Used comprehensions to set up the table because I love one liners but they might be a little too dense
- Example Column: [{A:1, B:1}, {A:2, B:0}, {A:0,B:2}]

Procedure - Viterbi Classification

- Same as we learned in class
  - Iterate through sequence selecting most likely states
  - No state transitions means it’s just multiplication of each states probabilities and selecting the max
- Log functionality is present

Procedure - Random Sequence Generation

- Iterate through each column
- Generate a pool of emissions representative of that columns probabilities
- Randomly select a character
- Return the sequence of characters
- Used this data to create a test data set
    - Issues with this approach is if my training data is flawed my generated data will be flawed
    - Already noted the training data comes from primates specifically

Procedure - Example

```
78  x = HMM()
79  x.load_data("splice.data.txt")
80  x.build_model()
81  x.prob_columns
82  seq, state = x.generate_sequence()
83  print(seq, state)
84  print(x.viterbi_classification(seq, logodd=0))
85  print(x.viterbi_classification(seq, logodd=1))
```

```
GGTCAGATGTGCTDCTGGGTCTTTTTGCAGGAGCAACTTACCTCGTGCGTTGACTCACGT IE
('IE', 5.64379883438868e-37)
('IE', -36.248428474483966)
```

Results - Self Test

```
Self Test
-- EI --
Accuracy:  0.975504799735187
Specificity 0.9754549301735083
Sensitivty 0.9455081001472754


-- IE --
Accuracy:  0.9774685222001326
Specificity 0.9767638360794254
Sensitivty 0.9494047619047619


-- N --
Accuracy:  0.9513288504642972
Specificity 0.9761737911702874
Sensitivty 0.9540507859733979
```

Results - On Generated Data

```
Test on 100000 Points of Randomly Generated Data
-- EI --
Accuracy:  0.9854005875616193
Specificity 0.9843981948815391
Sensitivty 0.9780335041505589


-- IE --
Accuracy:  0.9878726327737619
Specificity 0.9884326541452115
Sensitivty 0.9814510233918129


-- N --
Accuracy:  0.9741670595630991
Specificity 0.9878251081225191
Sensitivty 0.9610935234770259
```

Discussion

- Did we solve the problem statement?
  - A good first step
  - Need to test with different organisms - requires more data
  - Consistent results across all the classes
- Original attempt wasn’t a profile HMM and didn’t perform as well
  - Had a fine grained implementation of viterbi
  - Assumed state transitions were possible - made no significant difference in results
  - Suffered from having a high false positive rate in the N class
  - Proved to be a really great learning experience

Realistic Next Steps

- Cross Validation - Implement using numpy/pandas or making sklearn compatible
- Adding specific emissions
- Adding specific pseudocounts at specific columns for specific emissions

References

1. C.L. Blake and P.M. Murphy. The UCI machine learning repository. [https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)]. In Irvine, CA: University of California, Department of Information and Computer Science, 1998.
2. Wheeler, T. J., Clements, J., & Finn, R. D. (2014). Skylign: A tool for creating informative, i nteractive logos representing sequence alignments and profile hidden Markov models. *BMC Bioinformatics,15*(1), 7. doi:10.1186/1471-2105-15-7
3. Intron. (2018, April 17). Retrieved from https://en.wikipedia.org/wiki/Intron
4. Jo, B., & Choi, S. S. (2015). Introns: The Functional Benefits of Introns in Genomes. *Genomics & Informatics,13*(4), 112. doi:10.5808/gi.2015.13.4.112