Bayesian Statistics: Sequence Matching

Mitchell Kwock

Abstract

In Communications and Biology, matching two sequences with an unknown separation is a constant problem that always needs to be solved. In signals, there is often data that affects itself, dispersing its information in random and varied ways. In Biology, we find that DNA also has sequences that are important to understand the nature of. However, in order to get a good hold of the sequence's meaning, we have to know what the sequence is in the first place.

1. The Project

To obtain the readings of a strand of DNA, biologists use PCR machines to copy large strings of DNA, take forward and reverse reads, then attempt to reconstruct, with some degree of accuracy, the original sequence. A forward read and reverse read use different enzymes to traverse the DNA in a single direction and inform biologists what the nucleotide base is at that position. What the biologists see are following strands, but without the known shift.

Genome: ATTACGGTAAATCGGATCCCCTGAAAAAATCCGAGGATC
Forward: ATTACGGTAAATCGGATCCCCTGAA...?
Reverse: ?...TCGGATCCCCTGAAAAAATCCGAGGATC

However, not every nucleotide is guaranteed to be read correctly, meaning the method of shifting sequences until they greatly match is unreliable. The enzyme's earlier reads are all very accurate, but due to fatigue and sheer chance, the enzyme becomes less accurate in later parts of the sequence. For a forward read, error happens at the tail end of the sequence, whereas for a reverse read, error is more likely at the beginning. This project is a Bayesian approach to determining the likelihoods of different shifts in the sequence, and an attempt to choose the best sequence and reconstruct the original strand.

2. Modeling the Genome

In order to simulate a genome, I have chosen to string 10,000+ base pairs in a sequence, approximately on the order of magnitude of standard sequences. Although this is not necessarily accurate because of evolution's filter, for the purpose of testing the updates, this model will suffice.

To continue, I needed to model error when reading the genome. When a forward and reverse read of a genome is done, we get two separate sequences that overlap at some point. Each sequence is read until the accuracy becomes consistently random. This occurs due to the nature of the enzymes used to read the sequences. Thus, the following requirements for a model were established

- In the beginning, the sequence has a very high likelihood of being read correctly (That is, an A will likely read as an A as opposed to a G).
- As the sequence progresses, the likelihood of reading correctly decreases and never increases. $\frac{\delta_{Err}}{\delta_{Index}} >= 0$ for all shifts.
- Early to middle reads are consistently good while later reads become random due to the enzyme's accuracy loss.

This information leads to the following approximation for the model.

Equation	Explanation
$p = 1, x < 0.5$	
or $p = 0, x > 0.5$	Meets the end conditions
$p = 1 - x$	Describes decreasing accuracy
$p = 1 - x^c$	Describes the initial accuracy and the later inaccuracy
$p = 1 - s * x^c$	Gives a baseline accuracy at the end by decreasing the size of s

The error handling is done with a polynomial function as an approximation of the error obtaining a sequence. If it reads incorrectly, then a random A, T, C, or G is selected and put in to that slot. The chance that an A will read A or a T will read T is seen in Figure 1 on page 3
Additional note: When comparing the forward and reverse reads, the area where the forward read is most reliable is where the reverse read is most unreliable and vice-versa.

3. Modeling the Bayesian Update

Thanks to Bayesian and computers, hand-matching is not necessary. This Bayesian update requires a definition for a hypothesis and the data that we obtain in order to adjust the likelihoods of each shift.

Hypothesis	The sequences are shifted some number of base-pairs
Data	At this index, the base pairs match, don't match, or don't exist

In order to solve this, the problem must be split into the three possibilities and solved for accordingly.

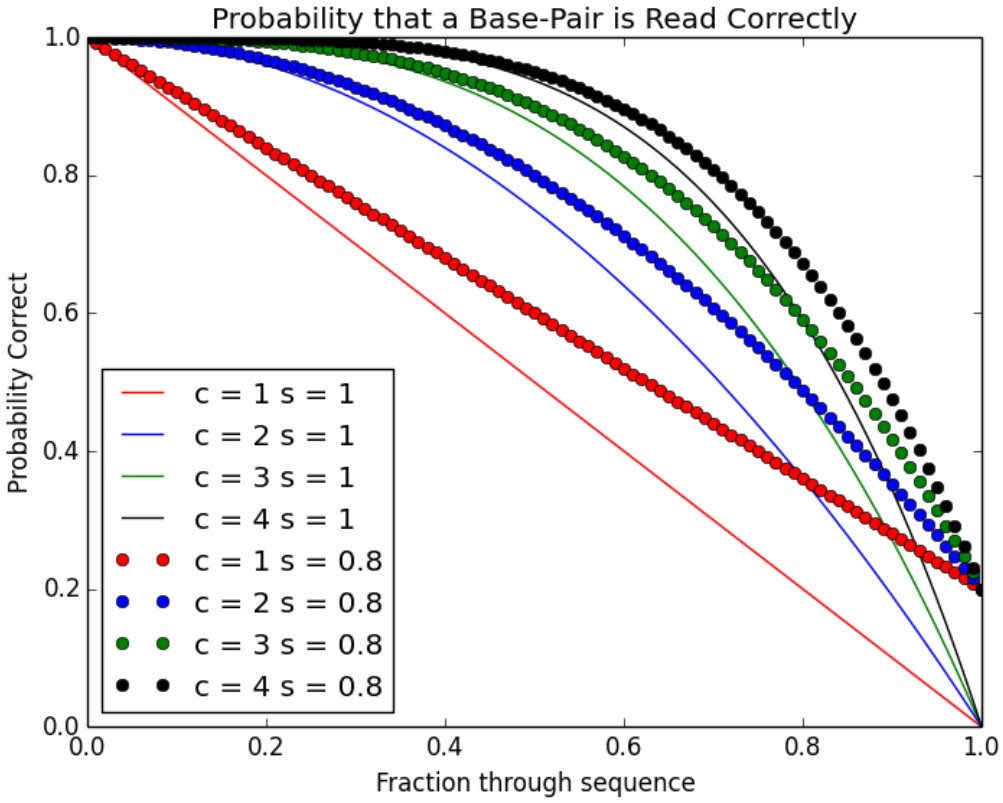

Figure 1: Different probability curves as a function of fraction through a DNA sequence.

3.1. The Pairs Match

Given that the two nucleotides match at this index and shift (for instance both are A), the forward read will have a p chance of being a particular nucleotide while the reverse will have a k chance of being being the same. These two probabilities are calculated using the previously described probability for reading the base-pair correctly.

Forward: Chances for each base

A	T	C	G
p	$\frac{1-p}{3}$	$\frac{1-p}{3}$	$\frac{1-p}{3}$

Reverse: Chances for each base

A	T	C	G
k	$\frac{1-k}{3}$	$\frac{1-k}{3}$	$\frac{1-k}{3}$

A positive read occurs when the both are read as A, T, C, or G, and because they are mutually exclusive, the chances can be summed for the entire positive read. A false negative occurs in the complement to the positive because under these conditions, a positive and a false negative is all-inclusive and mutually-exclusive.

$$\text{Positive: } pk + \tfrac{(1-p)(1-k)}{3} \quad \text{False Negative: } 1 - [p * k + \tfrac{(1-p)(1-k)}{3}]$$

3.2. The Pairs do not Match

The second case is in the case that the nucleotides at this index hypothesis do not match. There is still a p and k chance that they are read correctly, but the overlapping probabilities are different.

Forward: Chances for each base

A	T	C	G
p	$\frac{1-p}{3}$	$\frac{1-p}{3}$	$\frac{1-p}{3}$

Reverse: Chances for each base

A	T	C	G
$\frac{1-k}{3}$	k	$\frac{1-k}{3}$	$\frac{1-k}{3}$

This information yields the following false positive and negative equations. The false positive is the chance that the same data is read even though the original nucleotides do not match.

$$\text{False Positive: } \tfrac{p+k-2pk}{3} + \tfrac{2(1-p)(1-k)}{9} \quad \text{Negative: } 1 - [\tfrac{p+k-2pk}{3} + \tfrac{2(1-p)(1-k)}{9}]$$

Given this information, if we read a match, the likelihood that the hypothesis, the index shift, is correct is adjusted by the chance that a match exists and a match was read divided by the chance that there was a match read. To rephrase, the likelihood is the chance that a positive occurs given that a positive or a false positive could occur.

$$\frac{Positive}{Positive+FalsePositive}$$

If there is no match read, the chance of the hypothesis being correct is proportional to the chance that it's a false negative given that a negative read happens.

$$\frac{FalseNegative}{Negative+FalseNegative}$$

3.3. Index Out of Range Exception!

The last case occurs when an index is out of range, that is, there is nothing to compare the base against. The returned likelihood is always 1 because if there's no index to match against, that is the expected result. Therefore this data does not add evidence one way or another because.

4. Execution

With the model in place, the coding process can begin. In this section, snippets of the code based on the previously mentioned *Modeling the Genome* section.

```
def Prob_True(portion):
        '''
        Indicates the probability that the returned value is actually a true read.
        portion is the fraction through the string ()

        Requirements:
        Perfect (1) reading at portion=0
        Lowest reading (minimum 0) at portion = 1

        At all points in between: dProb_True/dPortion < 0 (Meaning probability is alu
        '''

        # 1-(p)^c
        scale = s.Get("prob-max") - s.Get("prob-min")
        y_intercept = s.Get("prob-max")
        if(scale > 1 or scale < 0):
                scale = 1
                y_intercept = 1.0
    # 1-s(p)^c
        return y_intercept - portion ** s.Get('curve') * scale

def Likelihood(self, data, hypo):
        '''
        hypo: Starting index of the intended string
        data: [Index, reverse_base] The read Base at Index

        Returned probability is the probability that they were matched
                Needs to account for:
        Increasing error in forward string as well as decreasing in reverse
        '''
        index        = data[0]
        reverse_base = data[1]
        index_match  = hypo + index

        # Deals with the no-match case because index is out of range
        if(index_match >= self.forward_length):
                return 1
```

```
        forward_base = self.forward_string[index_match]

        if(forward_base == reverse_base):
                # Match Found, determine chance of match accounting for False Positiv

                # Accounts for random chance of reading correctly because of weighted
                # Forward Probability of reading correctly

                f_portion = 1.0 * index_match / self.forward_length
                p = (1 + 3 * Prob_True(f_portion))/4
                # Reverse Probability of reading correctly
                r_portion = 1 - 1.0 * index / self.reverse_length
                k = (1 + 3 * Prob_True(r_portion))/4

                positive = p * k + (1-p)*(1-k)/3
                false_positive = (p + k - 2*p*k)/3 + (1-p)*(1-k) * 2/9
                return positive / (positive + false_positive)
        else:
                # No match found, what is the possibility that it's a false negative?
                # Forward Probability of reading correctly
                f_portion = 1.0 * index_match / self.forward_length
                p = (1 + 3 * Prob_True(f_portion))/4
                # Reverse Probability of reading correctly
                r_portion = 1 - 1.0 * index / self.reverse_length
                k = (1 + 3 * Prob_True(r_portion))/4

                false_negative = 1 - (p * k + (1-p)*(1-k)/3)
                negative =          1 - ((p + k - 2*p*k)/3 + (1-p)*(1-k) * 2/9)
                return false_negative / (negative + false_negative)
```

This code handles all three cases and returns the likelihood of the hypothesis knowing the data, the Bayesian update. One thing to note is that the p and k are adjusted to meet the base probability of being read correctly by accident because of the random return described in the read.

Figure 2: Successful run finding the shift at 46.

5. Results

In the first run found on Figure 2, two sequences of approximately 110 bases in length were matched against each other. This plots the Log Likelihood of each shift. The highest point is, based on the Bayesian approach, the most likely and also the correct shift. With that said, we should pay attention to the general shape of the graph. At much larger shifts, we find that the index is more likely than the previous segments. Which brings us to our next result.

On Figure 3, another two 120 base-long sequences were matched against each other. The correct shift is 44 bases long, but the most likely result is a shift of 105 bases. We, as humans, can see that at 44 bases, a spike occurs that is about 40 orders of magnitude more likely than the results around it. However, this spike is approximately 10 orders of magnitude smaller than the *most likely* result. This result occurs because the smaller shifts have more evidence. This evidence is always some likelihood less than one, and with hundreds of pieces, the final outcome approaches zero. A larger shift, however, has fewer pieces of data to compare against, and is therefore more likely because there is less to match against.

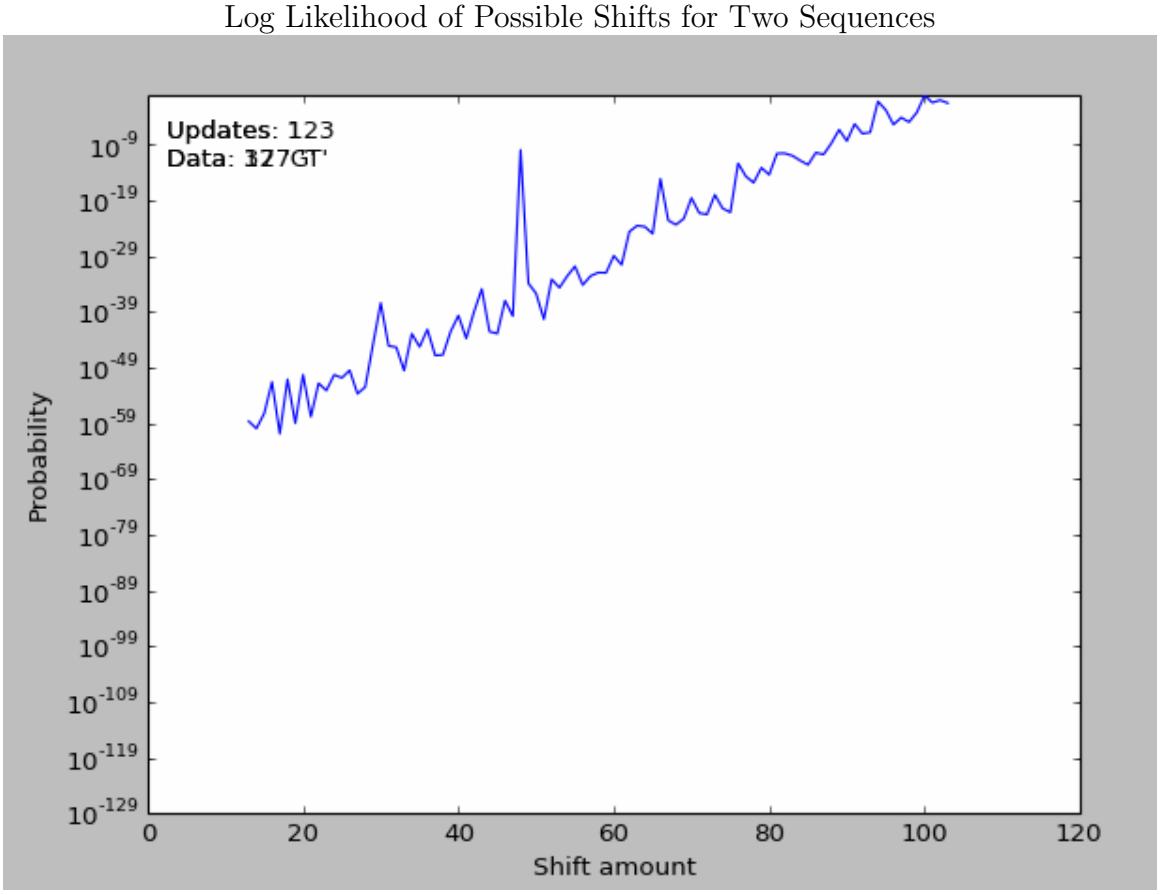

Figure 3: Unsuccessful Run, believing the shift to be at 105 whereas it actually exists at 44

Log Likelihood of Different Shifts for 500 base Sequences

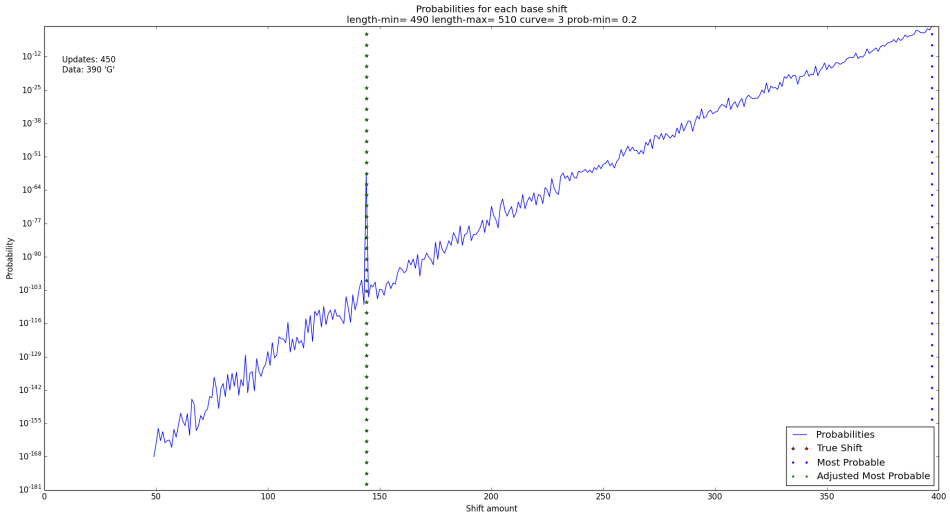

Figure 4: Successful adjusted base picker finding the 143 even though the most likely is at 400

6. Recognizing These Outliers

I chosen to believe that this method of comparing data is unfair to earlier shifts, often filtering them out by huge orders of magnitude because they have a huge range of evidence that brings them a bit lower each time. After this Bayesian Approach, I have opted to select from the new data depending on the outcome using an arbitrary "Human Picker."

To the human eye, the outlier is obvious: The spike in the function. Recognizing that the rest of the function is roughly linear, taking a linear regression of the logarithmic function gives us a line, and the adjusted pick is the one whose value is most above this line. This process takes the hypotheses and finds the hypothesis which is most possible given that its position has innately lower likelihoods. In order to model this "Human Pick", I chose to add this code.

```
# Names to indicate x and y.  Changed because x and y are used elsewhere
ex = np.asarray(list(suite.Values()))
why = np.log([suite[x] for x in suite.Values()])
slope, intercept, r_value, p_value, std_err = stats.linregress(ex, why)
# Find how much each hypothesis is better than the regression
adjusted = why - slope * ex
# Find the highest value above the linear regression
adjusted_high = list(adjusted).index(max(adjusted)) + suite.Values()[0]
```

This extra code finds the location where the shift is highest relative to what it should be, that is, the spikes in the graph. Running the code again yields the following results:

With this next iteration shown on Figure 4, two 500 base sequences were matched against each other. The correct shift is at 143 bases as seen in the picture and is correctly chosen by the "Human Picker". This method is knowingly not a Bayesian solution, but one that tends to pick the correct outcome more often, especially when working with considerably longer strings. Note that the correct shift of 143 has a likelihood of less than $10^{-}50$ with this data. A completely Bayesian gambling machine would never pick it, yet this is a case where the guessing game finds the match more often than the model!

7. Looking Forward

This was a fun exercise in Bayesian Statistics and a reminder to the the mind-blowing nature of Bayesian updates. The fact is, given the probabilities of things, coming later is more likely than small values of incorrectness at each step. I cannot tell if Bayesian has failed me or if my model is flawed, but I do know that the comparison method described is unfair and delivers incorrect results more often than not for very long sequences.

For future work, the following additions and complications could be added to improve the practicality and usefulness of the project.

- Add weighting to the different bases, changing their individual probabilities.
- Experiment and change the Probability function to included added/shifted bases.
- Extend the overlap such that the entire forward and reverse read are contained within one another.
- Have a complete probability set for the reads.
- Add visualization tools.
- Use actual genomic data from different resources.