Pay attention, Problem set #5, is based on this material.

Inferring Ancestry using HMMs
Decoding Problem Solution

- The Decoding Problem is equivalent to finding a longest path in the directed acyclic graph (DAG), where "longest" is defined as the maximum product of the probabilities along the path.
Viterbi Decoding Algorithm

- Since the *longest path* is a product of edge weights, if we use the \( \log \) of the weights we can make it a sum again!
- The value of the product can become extremely small, which leads to underflow.
- Many common probability distributions have an exponential form. Taking their log simplifies these distributions.
- Improves numerical accuracy and stability.

\[
s_{k,i+1} = \log(e_l(x_{i+1})) + \max_{k \in Q} \{ s_{k,i} + \log(a_{kl}) \}
\]
Viterbi Decoding Algorithm (cont)

- Every path in the graph has the probability \( P(x|\pi) \).
- The Viterbi decoding algorithm finds the path that maximizes \( P(x|\pi) \) among all possible paths.
- The Viterbi decoding algorithm runs in \( O(n|Q|^2) \) time (length of sequence times number of states squared).
- The Viterbi decoding algorithm can be efficiently implemented as a dynamic program.

**Dynamic Program's Recursion:**

\[
s_{l,i+1} = \max_{k \in Q} \{ s_{k,i} \cdot \text{weight of edge between } (k, i) \text{ and } (l, i + 1) \}
= \max_{k \in Q} \{ s_{k,i} \cdot a_{kl} \cdot e_l(x_{i+1}) \}
= e_l(x_{i+1}) \cdot \max_{k \in Q} \{ s_{k,i} \cdot a_{kl} \}
\]
Viterbi Example

- Solves all subproblems implied by observed sequence
- How likely is this path? 0.006
- What is it? $BBBBBB$
How likely is "most likely?"

- The "most likely path" may not be a lot more likely than a 2nd or 3rd most likely paths (even more so in more realistic cases than this one).
- Actual probability of the "most likely path" is not that high.

<table>
<thead>
<tr>
<th>P</th>
<th>π</th>
<th>P</th>
<th>π</th>
<th>P</th>
<th>π</th>
<th>P</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0058</td>
<td>BBBBBB</td>
<td>0.0001</td>
<td>BBFFFB</td>
<td>0.0000</td>
<td>FFFFBF</td>
<td>0.0000</td>
<td>FBBFBF</td>
</tr>
<tr>
<td>0.0046</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FFFFBF</td>
<td>0.0000</td>
<td>FBFFFB</td>
<td>0.0000</td>
<td>BFBBFF</td>
</tr>
<tr>
<td>0.0013</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFBBF</td>
<td>0.0000</td>
<td>BFBBFB</td>
<td>0.0000</td>
<td>BFBBFF</td>
</tr>
<tr>
<td>0.0012</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFFB</td>
<td>0.0000</td>
<td>BBFFFB</td>
<td>0.0000</td>
<td>FBFFBF</td>
</tr>
<tr>
<td>0.0009</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFFB</td>
<td>0.0000</td>
<td>BFFBBF</td>
<td>0.0000</td>
<td>BFBBFF</td>
</tr>
<tr>
<td>0.0008</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBBBFB</td>
<td>0.0000</td>
<td>BFBFFB</td>
<td>0.0000</td>
<td>BFBBFF</td>
</tr>
<tr>
<td>0.0006</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0006</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0004</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0003</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0003</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBBFFF</td>
<td>0.0001</td>
<td>FBFFF</td>
<td>0.0000</td>
<td>BBFBBB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
</tbody>
</table>

"FFFFFF" is nearly as good as "BBBBBB"
HMMs in Biology

- Inferring ancestral contributions of a descendant
- Collaborative Cross project
- Maintained at UNC since 2006

Objective:
Create new reproducible mouse strains by randomly combining the genomes of eight diverse mice strains

Problem:
Given an extant strain, which parts of its genome came from which founder?
Mixing Genome

- A randomized breeding scheme was used to
  - Mix the genomes by recombination
  - Fix the genomes by inbreeding
- A breeding funnel - 8 genomes go in a mosaic comes out
- Genotyping was used to track founder contributions
Instead of “Birds and Bees,” Mouse and Flies

- Recombination mixes the genomes of the two chromosomes
- Sib-mating causes the genomes to fix
A Genome Mosaic

- A Hidden Markov Model is used to infer the "hidden" state of which of the 8 founders contributed to what parts of the genome
- A Viterbi Solution finds the most likely mosaic given a set of “genotypes”
Genotyping Microarrays

- DNA probes to query the state of specific “known” and “informative” Single Nucleotide Polymorphisms (SNPs)
  Bases in the genome that vary within a population
- Each probe distinguishes 4 cases ("Ref", "Alt", "H", "N")
- From these observations we infer the founder at every marker
Example Genotypes

- Genotypes for a chromosome
- 1000s of probes with positions of variant
- Alleles are indicated by the nucleotide
- Rarely can a single maker resolve the founder
- Which strain would you guess?
Genotype Noise

- One last issue, between 1% and 5% of genotypes are simply wrong
- Source of errors
  - A probe didn't glow bright enough
  - A section of the array was damaged
    (fingerprints, cracks, hair, etc.)
  - Mess ups when fabricating a probe’s sequence
  - DNA itself was contaminated
- Error types:
  - “No” calls (observation is uninformative)
  - A possible, but incorrect call
Reading Genotypes

In [1]:

```
fp = open("CCGenotypes.csv", 'r')
data = fp.read().split('"
')  # break file into lines
fp.close()
header = data.pop(0).split(',')  # First line is header
while (len(data[-1].strip()) < 1):  # remove extra lines
    data.pop()
for i, line in enumerate(data):
    field = line.split(',')
    field[1] = int(field[1])  # convert position to integer
    data[i] = field
fp.close()

print(header)
print("Number of probes", len(data))
for i in range(100,110):
    print("data["+i+"],", data[i])
```

[['Chromosome', 'Position', 'A/J', 'C57BL/6J', '129S1/SvImJ', 'NOD/ShiLtJ', 'NZO/H1LtJ', 'CAST/EiJ', 'PWK/PhJ', 'WSB/EiJ', 'CC004/TauUnc']]

Number of probes 419
Emission Probabilities based on Genotypes

Each probe has its own emission probabilities

```python
In [2]: i = int(input("Enter locus [0, %d] to see its Emission probability:" % len(data)))

print(data[i])
Nstates = 8
ErrorRate = 0.05
# Count expected genotypes
count = dict([(call, data[i][2:2+Nstates].count(call)) for call in "ACGTHN"])
print("", ', '.join(['"%8s" % v[0:8] for v in header[2:2+Nstates]])
for base in count.keys():
    # Compute emission probability, assuming 5% error rate
    if (count[base] == 0):
        emission = [1.0/Nstates for j in range(2,2+Nstates)] # unexpected
    else:
        emission = [(1.0 - ErrorRate)/count[base] if data[i][j] == base else ErrorRate/(Nstates - count[base])
        for j in range(2,2+Nstates)]
    emission = ["%6.4f" % v for v in emission]
    print("%s: %2d %s" % (base, count[base], emission))

Enter locus [0, 419] to see its Emission Probability:103
A: 3 ['0.0100', '0.0100', '0.0100', '0.3167', '0.3167', '0.3167', '0.0100', '0.0100']
C: 0 ['0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250']
G: 5 ['0.1900', '0.1900', '0.1900', '0.0167', '0.0167', '0.0167', '0.1900', '0.1900']
T: 0 ['0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250']
H: 0 ['0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250']
N: 0 ['0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250', '0.1250']
```

Comp 555 - Fall 2019
Transition probabilities

- Recombination likelihood is modeled using an exponential distribution
- Recombinations between nearby probes are unlikely
- Distant probes are more likely to be from different founders
Viterbi Algorithm as a Dynamic Program

```
In [18]: from math import exp, log10

Nstates = 8
prevpos = 1
state = [[[float(len(data)), i] for i in range(Nstates)]]  # (log(p), PathToHere)
for i in range(len(data)):
    # Count expected genotypes
    count = dict((call, data[i][2:2+Nstates].count(call)) for call in "ACGTHN")
    # Get the target genotype at this probe
    observed = data[i][-1]
    # Compute emission probability, assuming 5% error rate
    if (count[observed] == 0):
        emission = [1.0/Nstates for j in xrange(2, 2+Nstates)]  # unexpected
    else:
        emission = [0.95/count[data[i][j]] if data[i][j] == observed else 0.05/(Nstates - count[data[i][j]])
                    for j in range(2, 2+Nstates)]
    # Compute transition probability
    position = data[i][1]
    delta = position - prevpos
    prevpos = position
    stay = ((Nstates - 1.0)*exp(-delta/10000000.0) + 1.0)/Nstates
    switch = (1.0 - stay)/(Nstates - 1.0)
    # Update state probabilities for all paths leading to the ith state
    path = []
    for j in range(Nstates):
        choices = [(log10(emission[j]) + log10(stay) if (k==j) else log10(switch) + state[-1][k][0], k)
                    for k in range(Nstates)]
        path.append(max(choices))  # choices is a list of tuples of (score[i], from_whence_I_arrived[i])
    state.append(path)
print("Length of paths: ", len(state))

Length of paths: 418
```
Backtrack to find solution

```python
In [24]:
# backtrack
path = state[-1]
maxi = 0
maxp = path[0][0]
for i in range(1,Nstates):
    if (path[i][0] > maxp):
        maxp = path[i][0]
        maxi = i
print(maxi, path[maxi], header[2+maxi])
for j in range(len(state)-2,-1,-1):
data[j].append(header[2+maxi])
maxi = state[j+1][maxi][1]
header.append("Founnder")
fp = open("result.csv", 'w')
fp.write(','.join(header)+"\n")
prev = ""
for row in data:
    line = ','.join([str(v) for v in row])
    fp.write(line+"\n")
    if (row[-1] != prev):
        print(line)
        prev = row[-1]
print(line)
fp.close()
```

5 (129.5817061177885, 5) CAST/E1J
1,349090,0,C,C,A,A,C,A,A,A,A,PhK/Thj,PhK/PhJ
1,14334166,A,G,A,A,A,G,G,A,129S1/SvImJ,129S1/SvImJ
1,132786434,C,C,C,C,T,C,C,C57BL/6J,C57BL/6J
A peek at the result

- The inferred Mosaic
- Repeat for every chromosome
- Most likely, but how likely?
- Other approaches
Back to the Casino with new questions

- Are there common aspects of the most likely solutions?
- Which coin was I most likely using on the 4\textsuperscript{th} roll

<table>
<thead>
<tr>
<th>P</th>
<th>π</th>
<th>P</th>
<th>π</th>
<th>P</th>
<th>π</th>
<th>P</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0058</td>
<td>BBBBBB</td>
<td>0.0001</td>
<td>BBBFFB</td>
<td>0.0000</td>
<td>FFFBFF</td>
<td>0.0000</td>
<td>FBBFBF</td>
</tr>
<tr>
<td>0.0046</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FFFBF</td>
<td>0.0000</td>
<td>FFBFB</td>
<td>0.0000</td>
<td>BFBFFB</td>
</tr>
<tr>
<td>0.0013</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BFFBFB</td>
</tr>
<tr>
<td>0.0012</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FFFFB</td>
<td>0.0000</td>
<td>FFFFB</td>
<td>0.0000</td>
<td>BBFFFB</td>
</tr>
<tr>
<td>0.0009</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0008</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BBFFB</td>
<td>0.0000</td>
<td>BBFBFF</td>
</tr>
<tr>
<td>0.0013</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FFBFF</td>
<td>0.0000</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BBFBFF</td>
</tr>
<tr>
<td>0.0012</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BBFBF</td>
<td>0.0000</td>
<td>BFFBFB</td>
</tr>
<tr>
<td>0.0009</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0008</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BBFBF</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0006</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BFFBF</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0004</td>
<td>FFFFFF</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0004</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BBFBF</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0003</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0003</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBBBBB</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBBBBB</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBFFB</td>
<td>0.0000</td>
<td>BFBFF</td>
<td>0.0000</td>
<td>BFBBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>FBBBBB</td>
<td>0.0001</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBFFFF</td>
<td>0.0000</td>
<td>BFFFB</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>BBBFBB</td>
<td>0.0000</td>
<td>FFFBF</td>
<td>0.0000</td>
<td>BFBFF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>FBBBBB</td>
<td>0.0000</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
<tr>
<td>0.0001</td>
<td>FBBBBB</td>
<td>0.0000</td>
<td>FBBFF</td>
<td>0.0000</td>
<td>BFBBF</td>
<td>0.0000</td>
<td>BBFBFB</td>
</tr>
</tbody>
</table>
Forward-Backward Problem

**Given:** A sequence of coin tosses generated by an HMM.

**Goal:** Find the most probable coin that was in use at a particular flip.

\[
P(\pi_i = k| x) = \frac{P(x, \pi_i = k)}{P(x)}
\]

Where \(P(x, \pi_i = k)\) is the probabilities of all paths in state \(k\) at \(i\), and \(P(x)\) is the probability of sequence \(x\).
Illustrating the difference using 4 flips

\[ x = \text{TTHH} \]

\[ p \]

- FFFF (0.0228)
- BFFF (0.0013)
- FBFF (0.0004)
- BBFF (0.0019)
- FFBF (0.0004)
- BFBF (0.0000)
- FBBF (0.0006)
- BBBF (0.0028)
- FFFB (0.0038)
- BFFB (0.0002)
- FBBF (0.0001)
- BBFB (0.0003)
- FBBB (0.0057)
- BFBF (0.0003)
- FBBB (0.0085)
- BBBB (0.0384)

\[ P(x) = 0.0877 \]

\[ x = \text{TTHH} \]

\[ p \]

- FFFF (0.0228)
- FFBF (0.0004)
- FFBF (0.0038)
- FBFF (0.0004)
- FBBF (0.0057)
- BFFF (0.0013)
- BFBF (0.0000)
- BFFB (0.0002)
- BFBF (0.0003)

\[ P(\pi_2 = \text{F}|x) = 0.0345/0.0877 = 0.3936 \]

- FBFF (0.0004)
- FBBF (0.0006)
- FBFB (0.0001)
- FBBB (0.0085)
- BBFF (0.0019)
- BBBF (0.0028)
- BBFB (0.0003)
- BBBB (0.0384)

\[ P(\pi_2 = \text{B}|x) = 0.0532/0.0877 = 0.6064 \]
Forward Algorithm

- Define $f_{ki}$ (forward probability) as the probability of emitting the prefix $x_1...x_i$ and reaching the state $\pi_i = k$.
- The recurrence for the forward algorithm is:

\[
f_{k,i} = e_k(x_i) \cdot \sum_{l \in Q} f_{l,i-1} \cdot A_{l,k}
\]

- Similar to Viterbi solution to $i$, except all paths are multiplied together rather than taking the Max
However, *forward probability* is not the only factor effecting $P(\pi_i = k | x)$.

- The sequence of transitions and emissions that the HMM undergoes between $\pi_i$ and $\pi_{i+1}$ also affect $P(\pi_i = k | x)$.
- *Backward probability* $b_{k,i} \equiv$ the probability of being in state $\pi_i = k$ and emitting the suffix $x_{i+1} \ldots x_n$.
- The backward algorithm’s recurrence:

$$b_{k,i} = \sum_{l \in Q} e_l(x_{i+1}) \cdot b_{l,i+1} \cdot A_{k,l}$$
Forward-Backward Algorithm

- The probability that the dealer used a biased coin at any moment $i$ is as follows:

$P(\pi_i = k | x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)}$

- So, to find $P(\pi_i = k | x)$ for all $i$, we solve two dynamic programs
  - One from beginning to end
  - One from the end to the beginning
  - Combine the corresponding states
Next Time

Genome Rearrangements

Direct repeats
Interhomologous
Interstitial deletion
Interstitial deletion
Acentric fragment

Inverted repeats
Isodicentric chromosome
Acentric fragment
Interstitial deletion
Interstitial deletion
Acentric fragment

Inter-sister chromatid

Intrachromatid
Interstitial deletion
Acentric fragment

Inversion