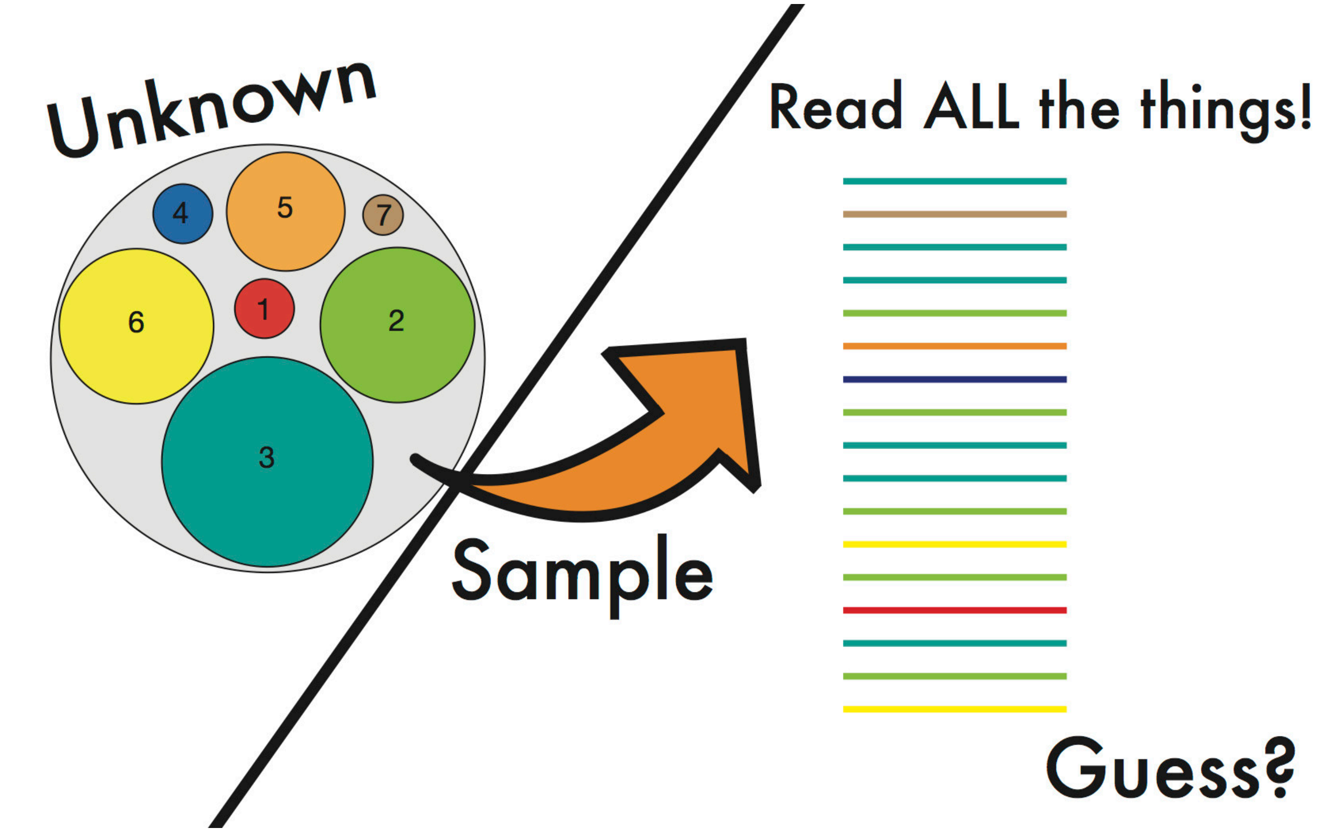
EE126 Lab10: RNA sequencing

The direct sequencing of RNA transcripts, known as RNA Sequencing (http://en.wikipedia.org/wiki/RNA-Seq), has many applications including genome annotation, comprehensive identification of fusions in cancer, discovery of novel isoforms of genes, and genome sequence assembly [Lior Pachter 2011] (http://arxiv.org/abs/1104.3889, http://genomemedicine.com/content/3/11/74/abstract). The problem of RNA sequencing is to figure out how much and what type of RNA is present in a genome at a given moment in time.

For our purposes, we'll phrase the problem as follows: Given a set of short reads that are sampled from a set of larger genes, how can we find the relative abundance of each gene. That is, given just the short reads, how do we know how frequently each original gene occurs. This process is depicted in Figure 1. (Aside: in the actual paper, these "genes" are actually "transcripts," but that's not relevant for us)



####

Figure 1: We want to use the reads to guess the underlying proportion.

Turns out, we can use some methods we learned in class (MLE, EM) to solve this problem! Let's try it out.

We assume that all genes are of the same length, \(\ell_t\), and all reads are of the same length, \(\ell_x\). Then, we assume the following Bayesian generative model:

1. A read comes from a randomly chosen gene
2. A read's starting point is randomly chosen among all possible \(\ell_t\) starting positions in that gene

Again, our goal is to estimate the abundance of each gene.

Let's begin by coming up with a way to use the tools we've learned so far to tackle this task. First, given a set of reads $X = \{X_1, X_2, \ldots, X_n\}$, we want to figure out what distribution over the genes ρ was most likely to give us X. To do this, all we need to do is maximize the following likelihood function:

$ L() = = $

Simple model

To make life easy, we's first going to assume no read is ambiguous—given a read, we can immediately tell which gene it came from (we know the color coding of the reads in in figure 1).

Q1. Given the assumption above, each X_i is aligned with only one gene, t_i. What is $P(X_i|\rho)$? Your solution should take both gene and read lengths into consideration.

Hint: How many possible starting positions exist for X_i?

You may denote the probability of seeing a specific gene as ρ_{t_i}.

A1. Your Solution Here

Q2. Assume that you have two genes, and x reads are compatible with gene 1, and $n-x$ reads are compatible with the other gene, gene 2. Using your solution above, find the maximum likelihood estimator of ρ.

A2. Your Solution Here

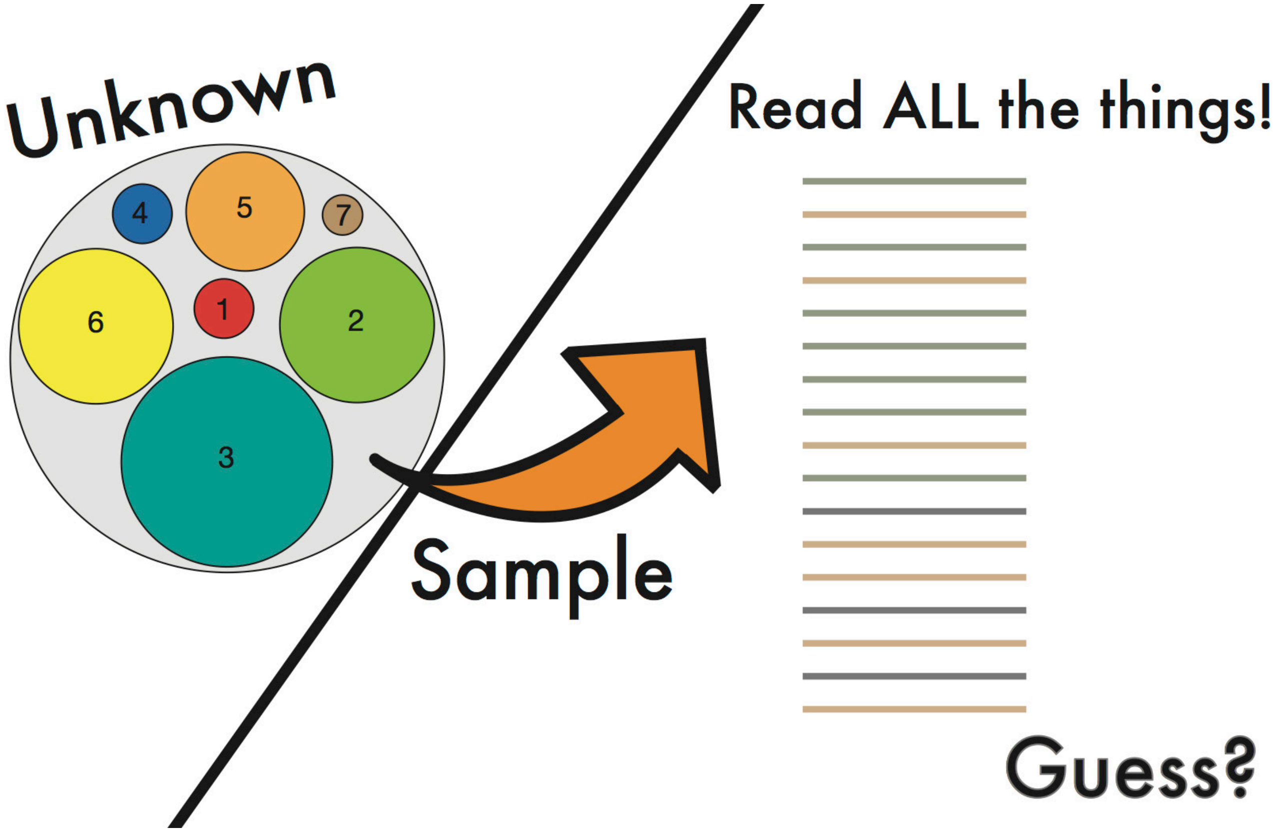
Q3. (Even more optional) Assume that you have M genes. Out of n reads, x_i reads are compatible with gene i, where $\sum_{i=1}^{M}{x_i} = n$. Find the maximum likelihood estimator of ρ.

You might just be able to make an "educated guess" from your answer above.

A3. Your Solution Here

Harder Model

Life gets harder when you allow for ambiguous reads. Going back to the figure above, we can imagine it as not knowing exactly which color a read belongs to, but we have a rough idea of the possible colors it could have been. See the modified figure below.



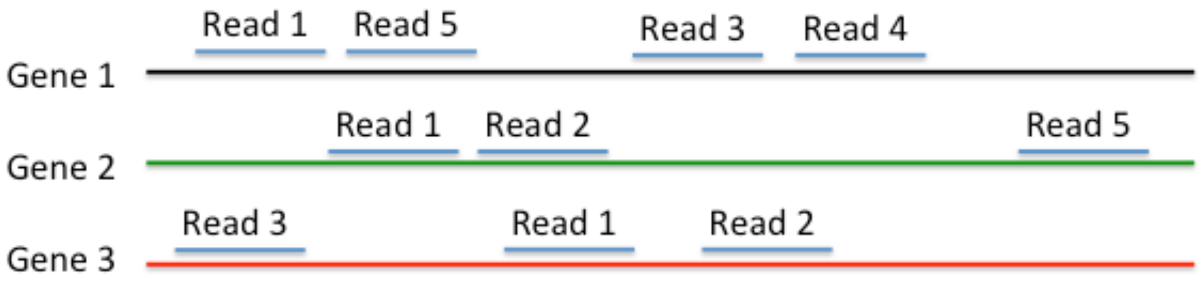
####

Figure 2

In this portion, we'll consider a general problem where a read is aligned possibly more than one gene. We first define a compatibility matrix $A \in \{0,1\}^{N \times M} = \{a_{i,j}\}$, where $a_{i,j}$ is 1 if read i is aligned with gene j, 0 otherwise.

Q4. Let's assume we have \(3\) genes and \(5\) reads, aligned as follows:

(ref: pp16-17 Pachter 2011)



####

Figure 3

Find the compatibility matrix \(A\)

A4. Your Solution Here

Q5. Given an arbitrary compatibility matrix, write an expression to find the likelihood function for \(\rho\).

You'll have to tweak your solution to the last portion by carefully considering how you can represent \(P(X_i|\rho)\) given the compatibility matrix.

A5. Your Solution Here

Q6. (Even more optional. Not crucial for rest.) Going back to the model with \(3\) genes and \(5\) reads in Q4, find the maximum likelihood estimator of \(\rho\).

Use the likelihood function you defined above. Clever algebraic manipulations can be used to bring it down to a form that you can maximize easily (a maximization over a single variable instead of three).

A6. Your Solution Here

Great! So now we were able to find the most likely ρ given ambiguous reads...in that one case. As you probably realized when doing the problem above, things can get ugly really fast. If you didn't do the problem above, the solutions will show you how bad it got. Let's look at another way to estimate the most likely distribution instead of trying to directly calculate it.

EM Algorithm! :D

In general, it is not easy to find the exact maximum likelihood estimator of ρ if ambiguous reads exist. Instead, we can rely on iterative methods that we hope will converge to the true value. One way to go about this is via expectation maximization (EM), as you have seen in class. Here is an EM algorithm that can be used for this specific problem. You'll be implementing it shortly, so read it carefully and understand why it works.

1. Initialize $\rho = (\frac{1}{M}, \frac{1}{M}, \ldots, \frac{1}{M})$, all genes are equally probable.

2. Find the compatibility matrix A.

3. Repeat the following until ρ converges:

3-1. For each gene i, find non-zero columns of ith row of A. Call these $\mathcal{I}_i$.

3-2. Choose the values of ρ in $\mathcal{I}_i$, normalize the vector, and replace ith row of A with the normalized vector.

3-3. Replace ρ such that $\rho_i = \frac{1}{N}\sum_{j=1}^{N}{A_{i,j}}$

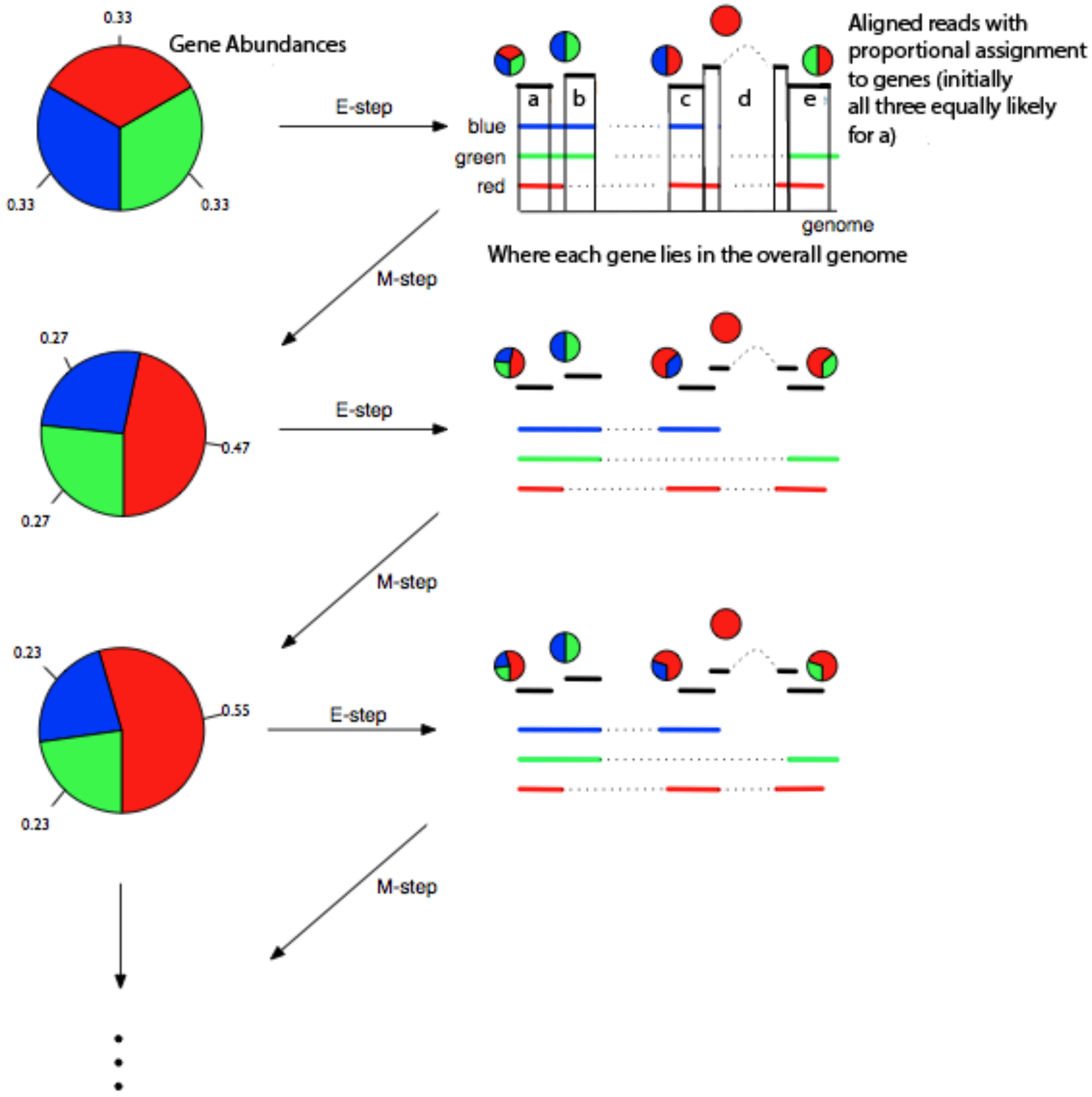
The following figure is a visual representation of the algorithm (Ref: p17 Pachter 2011). It's a lot to digest at once, so only look at the first step to begin with.

There are three possible genes to analyze (red, green, blue). There are five reads (a,b,c,d,e) that you can use to help you. One maps to all three, one only to red, and the other three to each of the three pairs. Initially, you assume a uniform prior (rho_guess).

During the expectation (E) step, reads are proportionately assigned to each of the genes they could have come from according to their abundances (rho_guess, currently uniform).

Next, during the maximization (M) step, the abundances are recalculated from the proportionately assigned read counts.

For example, the abundance of red after the first M step is estimated by 0.47 = (0.33 + 0.5 + 1 + 0.5)/(2.33 + 1.33 + 1.33).


0.33
Gene Abundances
0.33
0.33
E-step
blue
green
red
a
b
c
d
e
Aligned reads with proportional assignment to genes (initially all three equally likely for a)
genome
Where each gene lies in the overall genome
M-step
0.27
0.47
0.27
E-step
M-step
0.23
0.55
0.23
E-step
M-step

####

Figure 4

Q6. Explain the above EM algorithm in your own words. Will it always find the maximum likelihood estimator of \(\rho\)?

Hint: Is the likelihood function convex? concave?

A6. Your Solution Here

Q7. Implement the above EM algorithm. Run it with the given set of reads & genes. What is your estimated \(\rho\)?

A7. Complete the skeleton provided below. How close do you get? :D

Feel free to alter any of the code in the fourth code box.

In [ ]:

```
from __future__ import division
s = ['ATCTCGACGCACTGC', 'GAGTTCGAACTCTTC', 'AGAGTTCCAGTGTCA', 'AAAGCTCACTGCGGA
', 'AGCGATATCAGAGTD']
M = len(s) # Number of transcripts
```

In [ ]:

```
rho = rand(M)
rho /= sum(rho)
print rho
# This rho is unknown to us
```

In [ ]:

```
def rand_choice( pmf ):
    v = rand()
    tmp = 0
    for i in range(len(pmf)):
        each_val = pmf[i]
        tmp += each_val
        if v <= tmp:
            return i

def random_read( s, rho, L ):
    chosen_seq = s[rand_choice( rho )]
    start_idx = randint( len(chosen_seq) - L )
    end_idx = start_idx + L
    return chosen_seq[ start_idx:end_idx ]

N = 1000 # Number of reads
L = 5

reads = []
for i in range(N):
    reads.append( random_read(s, rho, L) )

print 'First 20 reads...', reads[0:20]
```

In [ ]:

```
N_iter = 100 #number of E/M iterations

def find_all_alignments(s, read):
    tmp = []
    for j in range(len(s)):
        if read in s[j]:
            tmp.append(j)
    return tmp

# A: Compatibility Matrix.
#Note: You can choose to represent this matrix in whatever form is easiest for
```

```
your calculations
#The form we are pushing you towards here is just how we chose to implement the algorithm
A = []
for each_read in reads:
    A.append( find_all_alignments(s, each_read) )

hidden_state_prior = zeros([N, M])
rho_est = []

#Your EM code here. Print your final answer as rho_est

print '(real) rho', rho
print '(est.) rho', rho_est
```

Congratulations! You've just finished your last lab in 126! Take a minute to appreciate what you've accomplished through the last semester. This class wasn't easy, but hopefully you've learned a lot (and let us know if you didn't >.<). Come see us if you'd like some help figuring out where to go from here.

Happy Holidays!