# Introduction to Computational Biology
# Lecture # 4: Estimating Scoring Rules for Sequence Alignment

Scribe: Harel Shein

18/11/2008

## 1 Brief Review

In the previous lecture we discussed sequence alignment algorithms. Specifically we used the divide and conquer method to come up with a space efficient approach to sequence alignment. Finally we explained how our algorithm can be modified to enable local alignments.

## 2 Motivation

In our discussion of sequence alignment algorithms we assumed we have a given set of scoring rules for generating the alignment. Today we will discuss a way for generating these scoring rules. More specifically, cosidering weight functions $\sigma : \Sigma^+ \times \Sigma^+ \rightarrow \Re$

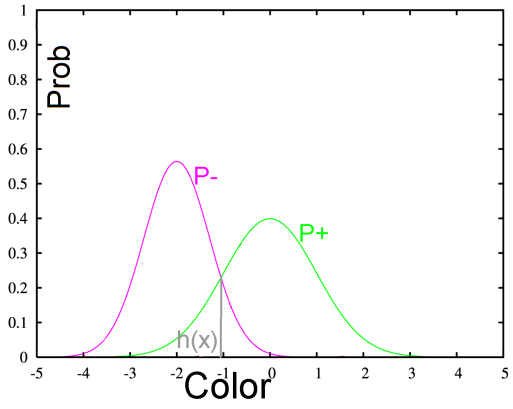
The choice of $\sigma$ will influence greatly the results. If we were to take evolution into account, what can we learn from it ?

**Example 2.1** ***Malaria*** *The malaria genome is 80% AT, does this influence our decision ? Will an A-A match get the same score as G-G ? 40% of the genome is A or T and only 1-% is G or C, thus we must take the proir into account.* ∎

## 3 Decision making problem

### 3.1 Example

Let's say I own an orchard ("Pardes") and I would like to tell my orange sorting machine which oranges it should ship to Europe and which to Israel. I will try to rely on the color of the peel and decide if it's a good orange. I have the following data:



If I want to find the threshold I need a decision rule: $h : x \rightarrow -1, +1$

Where x is the color, -1 is a bad orange and +1 is a good one.

**Definition 3.1** $\alpha(h) = P_-(x : h(x) = +1)$
*In words: the probability of objects I classified as good (+1) to be bad (-1).*

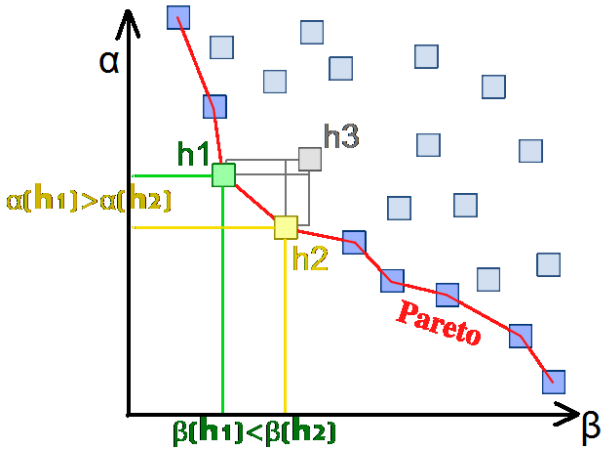
And we will define $\beta(h) = P_+(x : h(x) = -1)$
which is the probability of objects I classified as bad (-1) to be good (+1).

**Definition 3.2** *We say that $h_1$* **dominates** *$h_2$ if $h_1$ is always better that $h_2$, that is either:*

- $\alpha(h_1) < \alpha(h_2) \wedge \beta(h_1) \leq \beta(h_2)$ *Or:*
- $\alpha(h_1) \leq \alpha(h_2) \wedge \beta(h_1) < \beta(h_2)$

If the inequalities differ between $\alpha$ and $\beta$, it's harder to answer. We will therefore try to come up with a rule that tells us which is better $h_1$ or $h_2$.

## 3.2 Pareto Optimality




The boxed points represent feasible choices (decision rules), and smaller values of $\alpha$ and $\beta$ are preferred to larger ones. Point h3 is dominated by both point h1 and point h2. Points h1 and h2 are not strictly dominated by any other, and hence lie on the curve. (This is called a Pareto Frontier, From Wikipedia:
$http://en.wikipedia.org/wiki/Pareto\_efficiency$).

The following statistics lemma gives us an interesting result:

**Lemma 3.3** *(Neyman-Pearson) - h is* ***undominated*** *iff*
$\exists t$ *s.t.* $h(x) = sign(\log \frac{P_+(x)}{P_-(x)} - t)$ *where t is an offset.*

The lemma states that the optimality line is defined by rules that follow this condition, meaning that you don't need to look at other rules. We should use rules of the format of log ratio of $P_+$ and $P_-$.

# 4 Back to sequence alignment

We will now suggest a $P_+, P_-$ model for the sequence alignment algoritm, in order to simplify our discussion we will assume the alignment is gap-less.

## 4.1 Building the model

Lets assume we can divide our analysis of the problem into two disjoint complimentary occurrences:

1. M - The sequences are evolutionarly related $P(\vec{x}, \vec{y}|M)$
2. R - The sequences are unrelated $P(\vec{x}, \vec{y}|R)$

Lets consider the latter case first.
We will assume that the value of a given position is independent of adjacent positions in the sequence.
In addition, when the sequences are unrelated (R), we can assume that $\vec{x}, \vec{y}$ at any position are independent of each other (i.e.:$\forall i, P(x_i, y_i|R) = P(x_i|R)P(y_i|R)$).[1]
In other words, for any position i, both $x_i$ $y_i$ are sampled independently from some background distribution $\mathbf{P}_0$.
So that the likelihood of the given $\vec{x}, \vec{y}$ to be unrelated is:

$$P(\vec{x}, \vec{y}|R) = P(\vec{x}|R)P(\vec{y}|R) = \prod_{i=1}^{n} P_0(x_i)P_0(y_i) \tag{1}$$

In the first case above, we assume the two sequences are related, so they evolve from a common ancestor. For simplicity we will continue assuming that each position i in $(x_i, y_i)$ is independent of the others. So we assume $x_i$,$y_i$ are sampled from some distribution $\mathbf{P}_1$ of letter pairs. The probability that any two letters ("a", "b") evolved from the same ancestral letter is $p(a, b)$. So the likelihood of the given $\vec{x}$ $\vec{y}$, under this model is:

$$P(\vec{x}, \vec{y}|M) = \prod_{i=1}^{n} P_1(x_i, y_i) \tag{2}$$

## 4.2 A decision problem

So once again we have stumbled across a decision problem. Given the two sequences $\vec{x}, \vec{y}$ we have to decide whether they are sampled from M (Model=related) or from R (Random=not related). We want to construct a decision procedure $D(\vec{x}, \vec{y})$ that returns M or R. Basically we want to compare the likelihood of our data in both models.
First, lets notice that our decision procedure can make either one of two error types (same as before):

- Type I - $\vec{x}, \vec{y}$ are sampled from R but $D(\vec{x}, \vec{y} = M)$
- Type II - $\vec{x}, \vec{y}$ are sampled from M but $D(\vec{x}, \vec{y} = R)$

The probabilities of such errors are also defined (also the same):

- $\alpha(D) = P(D(\vec{x}, \vec{y}) = M|R)$
- $\beta(D) = P(D(\vec{x}, \vec{y}) = R|M)$

We would of course favor a procedure which minimizes both error types. Using the Neyman-Pearson lemma, let us look at the following equation:

$$\frac{P(\vec{x}, \vec{y}|M)}{P(\vec{x}, \vec{y}|R)} = \frac{\prod_i P_1(x_i, y_i)}{\prod_i P_0(x_i)P_0(y_i)} = \prod_i \frac{P_1(x_i, y_i)}{P_0(x_i)P_0(y_i)} \tag{3}$$

Or for convenience, by taking a logarithm from both equation sides, of the form:

$$\log \frac{P(\vec{x}, \vec{y}|M)}{P(\vec{x}, \vec{y}|R)} = \log \frac{\prod_i P_1(x_i, y_i)}{\prod_i P_0(x_i)P_0(y_i)} = \sum_i \log \frac{P_1(x_i, y_i)}{P_0(x_i)P_0(y_i)} \tag{4}$$

This expression tells us that we need to take the prior probabilities ($P_0$) into account. Now we can define our scoring rule matrix as follows:

$$\sigma(a, b) = \log \frac{P_1(a, b)}{P_0(a)P_0(b)}$$

[1] Why can we make this assumption? Every thing in biology tells us otherwise.

## 4.3 Parameter Estimation

If we could estimate the probabilities $P_1$ and $P_0$ from our data we would have our scoring matrix $\sigma$.
We will discuss parameter estimation through an example:
Let's take the following i.i.d. samples $x_1, x_2, ..., x_n \sim P_\theta$ we would like to learn about $P_\theta(X = x) = \theta_x$ where $\theta$ is a probability vector ($\theta_x \geq 0, \sum_x \theta_x = 1$).

$$Likelihood = \ell(\vec{\theta}) = P(D|\theta) = log(\prod_i P_\theta(x_i)) = \sum_i log(P_\theta(x_i)) =^{(*)} \sum_x N_x log\theta_x$$

* = $N_x = \sum_i \mathbf{1}\{X_i = x\}$. That is, $N_x$ is the number of times I saw $x$ in the sample.

**Definition 4.1** *Sufficient Statistic : $S(D)$ is a sufficient statistic if*

$$S(D) = S(D') \Rightarrow \forall\theta P_\theta(D) = P_\theta(D').$$

*In words, S keeps all the necessary information on the data to compute the likelihood. We'll usually be interested in the minimal set of sufficient statistics.*∎

We would like to generalize a process of finding an estimator. one method is the MLE - Maximum Likelihood Estimator.

## 4.4 MLE - Maximum Likelihood Estimator

We mark the estimator $\theta$ that will maximize $L(\theta)$, as $\hat{\theta} = \arg\max_\theta \ell(\theta)$. To find $\hat{\theta}$ we need to:

1. Calculate the likelihood function L.
2. Find a maximum for the likelihood function.

So how do we find the maximum of our likelihood function? By finding where $\frac{\partial \ell}{\partial \theta} = 0$.
In our example we'll get:

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial}{\partial \theta_x} \sum_x N_x log\theta_x = \frac{N_x}{\theta_x} =^! 0 \Rightarrow ???$$

We have reched a dead end, we can't conclude $\hat{\theta}$ from this equation. This brings us to the development of a Lagrangian.

## 4.5 Langrange Multipliers (aka Coefficients)

When we have a problem of optimization of $f(x)$ but we have some constraint $C(x) = 0$, we will define a new Lagrangian function

$$J(x, \lambda) = f(x) - \lambda c(x)$$

and we will want to show that when the partial derivative of $x$ and $\lambda$ is zero - the constraint is satisfied and we are in stationary point. This means we solved the original problem.

When we take a partial derivative of $\lambda$,

$$\frac{\partial J}{\partial \lambda} = C(x) =^! 0$$

and compare to zero, our constraint is satisfied and when we take a partial derivative of $x$

$$\frac{\partial J}{\partial x} = \frac{\partial f(x)}{\partial x} - \lambda \frac{\partial c(x)}{\partial x} =^! 0 \Rightarrow \frac{\partial f(x)}{\partial x} = \lambda \frac{\partial c(x)}{\partial x}$$

we find the optimum.

We showed that the derivation according to $x$ of $f(x)$ is a linear combination of the gradients of the constraints - we are in a point on the line of the constraint, where you can go no further, in the direction of $f(x)$ (if we are going on the line of the constraint as long as we go transversally to the contour line of $f(x)$ we are going 'uphill', but when we touch it tangentially we can go no further - 'top of the hill'. We know we got to this point when the two derivatives are the same).

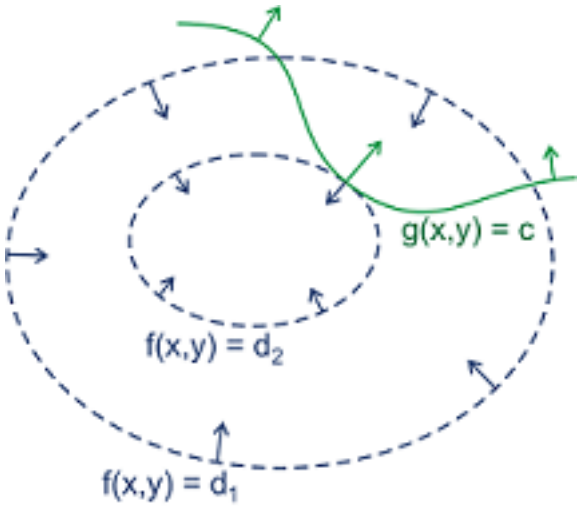



Figure 1: Drawn in green is the locus of points satisfying the constraint $g(x, y) = c$. Drawn in blue are contours of $f$. Arrows represent the gradient, which points in a direction normal to the contour.(from Wikipedia $http://en.wikipedia.org/wiki/Lagrange\_multipliers$)

## 4.6 back to MLE

$$\hat{\theta} = \arg\max_{\theta} \ell(\theta)$$

where C (the constraint) is

$$\sum_x \theta_x - 1 = 0$$

so we get

$$J(\vec{\theta}, \lambda) = \sum_x N_x log\theta_x - \lambda(\sum_x \theta_x - 1)$$

taking the partial derivative using $\theta_x$ we get

$$\frac{\partial J}{\partial \theta_x} = \frac{N_x}{\theta_x} - \lambda =^{!} 0$$

so we get

$$\lambda = \frac{N_x}{\theta_x} \Rightarrow \theta_x = \frac{N_x}{\lambda} \tag{5}$$

taking the partial derivative using $\lambda$ we get

$$\frac{\partial J}{\partial \lambda} = 1 - \sum_x \theta_x =^{!} 0 \Rightarrow \sum_x \theta_x = 1 \tag{6}$$

now, using results (5) and (6) we get

$$\sum_x \frac{N_x}{\lambda} = 1 \Rightarrow \lambda = \sum_x N_x$$

using (5) again we now conclude that

$$\theta_x = \frac{N_x}{\sum_x N_x}$$

this result corresponds to our intuition about the likelihood of the data.