Getting statistical significance and Bayesian confidence limits for your hidden Markov model or score-maximizing dynamic programming algorithm, with pairwise alignment of sequences as an example

Lee A. Newberg$^{1,2}$

$^1$Wadsworth Center, New York State Department of Health
$^2$Department of Computer Science, Rensselaer Polytechnic Institute

CCMB @ Brown University          October 7, 2009
You can do #1, but want to do #2 and #3:

Example: Sequence Alignment

1. For two sequences, of lengths $m$ and $n$, what is the optimal alignment $A$ and what is its score $S$?
2. Is $S$ statistically significant given $m$ and $n$? — is it unlikely to arise with random sequences?
3. Is $A$ credible? — are other plausible alignments of these sequences substantially the same?
You can do #1, but want to do #2 and #3:

Example: Word Wrapping Text

1. For a paragraph of words, what is the optimal way to divide them into lines $A$, and how pretty is it $S$?  
   
   E.g., $S = - \sum w_i^2$, where $w_i =$ spaces added to line $i$

2. Is $S$ unusual? — Is this paragraph of words particularly hard (or easy) to wrap?

3. Is $A$ special? — are other reasonable word wrappings of these words similar?
You can do #1, but want to do #2 and #3:

**Problem Statement**

1. **Optimization**: Find and evaluate an optimum using a dynamic programming algorithm, hidden Markov model, or partition function calculation.

2. **Hypothesis Testing**: What is the probability that random inputs would score as well? **Null distribution. p-value.**

3. **Bayesian Confidence Limits (a.k.a. Credibility Limits)**:
   - What fraction of solution space has exactly $d$ differences from the optimum, for $d = 0, \ldots, d_{\text{max}}$. **Difference distribution.**
   - How many differences must be allowed to capture 95% of solution space? **95% credibility limit.**

Note: Uncertainty of individual features (e.g., a specific alignment match) is valuable, but not our goal.
Results: Statistical Significance vs. Score

For Smith & Waterman (1981) sequence alignment, score and statistical significance are related, but . . .

- relationship is non-trivial and depends upon input size.

Protein-protein alignment  Nucleotide-nucleotide alignment

Compare: Karlin & Altschul (1990)
Significance and Bayesian confidence are related, but . . .

- poor credibility exists even at superb $p$-values.

20 gene promoters of *Drosophila melanogaster* aligned to orthologous regions in four other fly genomes.
For Smith-Waterman sequence alignment, the distribution of differences can have a rich structure.

Orthologous Human (1677 nt) vs. Mouse (1666 nt).
Viterbi(Dark) = 450 bp, Centroid(Light) = 438 bp.
Many problems are tackled with dynamic programming:

### Hidden Markov Model
- Sequence alignment: HMMER
- Protein folding: HMMSTR / ROSETTA

### Partition Function Computation / Markov Random Field
- RNA secondary structure: Sfold

### Viterbi / Maximum Score / Minimum Energy
- Seq. Alignment: Smith-Waterman, Needleman-Wunch
- RNA secondary structure: Mfold

Collectively, *Hidden Boltzmann Models*
Hidden Boltzmann Models

Flipping a biased coin

A Plan7 Profile-HMM (Eddy, 2003)

Also: Viterbi vs. Forward, Smith & Johnson (2007)
Estimating Statistical Significance

Naïve Sampling
1. Generate some random examples from the null.
2. Observe the fraction that score as well as your result.
Need $O(1/p)$ samples for a small $p$-value.

Importance Sampling
Similar to simulated annealing.
0. Establish a probability model, if absent.
1. Choose a temperature.
2. Generate random samples at the new temperature.
3. Compute temperature-corrected fraction $\geq$ your result.
Need 100–10,000 samples, even for $p = 10^{-4000}$.

Newberg (2008, 2009)
0. Establish a Probability Model

An *emission path* through the computation has a . . .

**Dynamic programming algorithm:** score, computed by addition of encountered transition and emission scores.

**HMM (or Partition function):** (unnormalized) probability or odds ratio, computed by multiplication.

Convert a Dynamic Programming Algorithm To Multiplications

- For each score $s$, instead use an unnormalized probability

$$Z = \exp(\lambda s).$$

*E.g.,* $\lambda = \ln(10)/5$ gives $Z \mapsto 10Z$ when $s \mapsto s + 5$.

- Addition of scores $\mapsto$ multiplication of $Z$s.

- Maximum of scores $\mapsto$ addition of $Z$s.
1. Choose a Temperature

Use a reasonable *ad hoc* procedure to obtain $T$.
- Generally, want 20-60% of instances $\geq$ your result.

2. Generate Samples

**Goal**: Instead of from the null, generate input instances from a temperature-biased distribution.

*E.g.*, generate a pair of sequences $(x, y)$ for alignment.

Watch out: two pages of math headed our way!
2. Generate Samples

1. Use HMM *forward algorithm* to sum over paths, but
   - Use $Z^{1/T}$ in lieu of each $Z$.
   - Also sum out emissions $d$ for each emitter $e$ using
     \[
     \left\langle Z_e^{1/T} \right\rangle = \sum_d Z_e(d)^{1/T} \Pr_{\text{null}}(d).
     \]

2. Use HMM *backtrace* to sample a path, but
   - Also sample each emission $d$ with probability
     \[
     \frac{Z_e(d)^{1/T} \Pr_{\text{null}}(d)}{\left\langle Z_e^{1/T} \right\rangle}.
     \]
   - Discard the sampled transitions.

**Result:** An input instance, with bias for higher scores.
3. Compute Temperature-Corrected Fraction

Naïve Sampling: For significance of result $Z_0$ (or $p_0$ or $s_0$)

$$p(Z_0) = \sum_{\text{all } (x,y)} \Pr_{\text{null}}(x, y) \Theta(Z(x, y) \geq Z_0),$$

where $\Theta(\text{true}) = 1$ and $\Theta(\text{false}) = 0$.

Importance Sampling

$$p(Z_0) = \sum_{\text{all } (x,y)} \Pr_T(x, y) \frac{\Pr_{\text{null}}(x, y) \Theta(Z(x, y) \geq Z_0)}{\Pr_T(x, y)},$$

$$\hat{p}(Z_0) = \frac{1}{N} \sum_{(x,y) \sim \Pr_T} \frac{\Pr_{\text{null}}(x, y) \Theta(Z(x, y) \geq Z_0)}{\Pr_T(x, y)}$$

Done with statistical significance!
How do we efficiently compute this (or its cumulative form)?

Newberg & Lawrence (2009)
Bayesian Confidence Limits

0. Establish a probability model, if absent.
1. Choose an integer difference measure.

Use *Sampling Approach* (Webb-Robertson *et al.*, 2008), *Direct Approach*, *Polynomial Approach*, or *Fourier Transform Approach*

### Fourier Transform Approach

2. Choose a integer (with only small factors) that is a little larger than the maximum number of differences.
3. Run modified *forward algorithm* to compute each Fourier transform coefficient *(in parallel)*.
4. Fourier transform the coefficients.
Unaltered Sequence Alignment Algorithm (Simplified)

Algorithm’s typical step looks something like:

\[
Z[i, j] = Z[i - 1, j - 1] Z_M(x_i, y_j) + \\
Z[i - 1, j] Z_D(x_i) + \\
Z[i, j - 1] Z_I(y_j)
\]

Goal is \(Z[m, n]\), where \(m\) and \(n\) are input strings’ lengths.
Recap: Unaltered Algorithm

\[ Z[i, j] = Z[i - 1, j - 1] Z_M(x_i, y_j) + \]
\[ Z[i - 1, j] Z_D(x_i) + \]
\[ Z[i, j - 1] Z_I(y_j) \]

Difference Distribution via the Direct Approach

Number of ways to get differences \( d \). Typical step:

\[ Z[i, j, d] = Z[i - 1, j - 1, d - \Delta_M(i, j)] Z_M(x_i, y_j) + \]
\[ Z[i - 1, j, d - \Delta_D(i)] Z_D(x_i) + \]
\[ Z[i, j - 1, d - \Delta_I(j)] Z_I(y_j) , \]

where \( \Delta \) is the number of new differences.
Goal is \( Z[m, n, d] \) for all possible total differences \( d \).
Requires increased runtime and memory. 😞
Polynomial Approach

Recap — Difference Distribution via the Direct Approach:
\(Z[m, n, d]\) is number of ways to get score \(d\).
\[
Z[i, j, d] = Z[i - 1, j - 1, d - \Delta_M(i, j)] Z_M(x_i, y_j) + \\
Z[i - 1, j, d - \Delta_D(i)] Z_D(x_i) + \\
Z[i, j - 1, d - \Delta_I(j)] Z_I(y_j).
\]

Difference Distribution via the Polynomial Approach

\(P[i, j]\) is a polynomial in indeterminant \(\omega\) that “packs” the \(Z[i, j, d]\) values. Define \(P[i, j] = \sum_d Z[i, j, d] \omega^d\). Typical step:
\[
P[i, j] = P[i - 1, j - 1] Z_M(x_i, y_j) \omega^{\Delta_M(i,j)} + \\
P[i - 1, j] Z_D(x_i) \omega^{\Delta_D(i)} + \\
P[i, j - 1] Z_I(y_j) \omega^{\Delta_I(j)}.
\]
Seeking \(P[m, n]\) polynomial.
Still increased runtime and memory. 😞
Recap — Difference Distribution via the Polynomial Approach:

\[ P[m, n] \] is a polynomial that packs the difference distribution.

\[
P[i, j] = P[i - 1, j - 1] Z_M(x_i, y_j) \omega^{\Delta M(i,j)} + \\
P[i - 1, j] Z_D(x_i) \omega^{\Delta D(i)} + \\
P[i, j - 1] Z_I(y_j) \omega^{\Delta I(j)}.
\]

Difference Distribution via the Fourier Transform Approach

Can recover coefficients of \( P[m, n] \) with via its valuation at sufficiently many points. Its value for a fixed \( \omega \) is from:

\[
C[i, j] = C[i - 1, j - 1] Z_M(x_i, y_j) \omega^{\Delta M(i,j)} + \\
C[i - 1, j] Z_D(x_i) \omega^{\Delta D(i)} + \\
C[i, j - 1] Z_I(y_j) \omega^{\Delta I(j)}.
\]

Coefficients recovery is efficient via Discrete Fourier Transform, so let \( \{\omega_0, \ldots, \omega_{r-1}\} \) be the \( r \)th complex roots of unity.
function ComputeScoreDistribution
for $k \in \{0, \ldots, r - 1\}$
\[ \omega = \cos(2\pi k/r) + i \sin(2\pi k/r) \]
\[ f(k) = \text{BackgroundExec}(\text{CalcFourier}(\omega)) \]
WaitForBackgroundProcesses
return DiscreteFourierTransform($f$)

function CalcFourier(ComplexNumber $\omega$)
for $i \in \{0, \ldots, m\}$
  for $j \in \{0, \ldots, n\}$
    $C[i, j] = C[i - 1, j - 1] Z_M(x_i, y_j) \omega^{\Delta_M(x_i, y_j)} + C[i - 1, j] Z_D(x_i) \omega^{\Delta_D(x_i)}$
    \[ + C[i, j - 1] Z_I(y_j) \omega^{\Delta_I(y_j)} \]
return $C[m, n]$

- Serial algorithm has original memory requirement. 🙃
- Parallel algorithm has (nearly) original runtime. 😊
Concluding Observations

For dynamic programming algorithms, hidden Markov models, and partition function calculations

- ...optimum score, statistical significance ($p$-value), and credibility / Bayesian confidence limits are not fungible.

Solutions

In many cases, if you can optimize score then you can

- ...estimate even a very extreme $p$-value.
- ...calculate the difference distribution and credibility limits.

Links

http://www.rpi.edu/~newbel/publications/

Statistical Significance of sequence alignments: Newberg (2008)
Statistical Significance of hidden Boltzmann models: Newberg (2009)
Credibility: Newberg & Lawrence (2009)