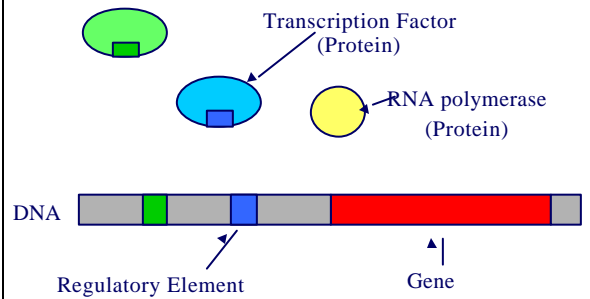
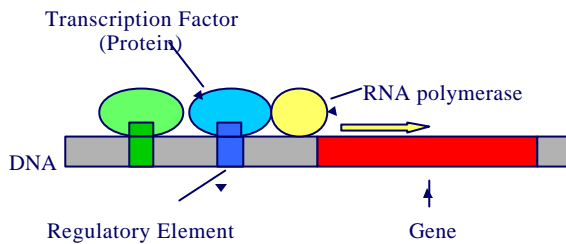


## Motif Finding

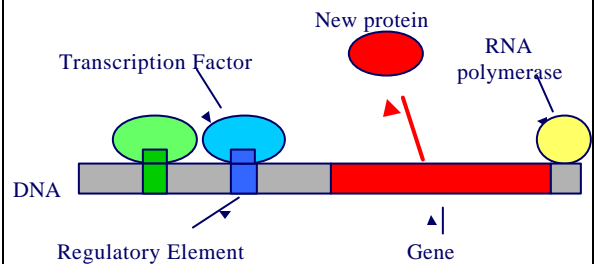
### Regulation of Genes



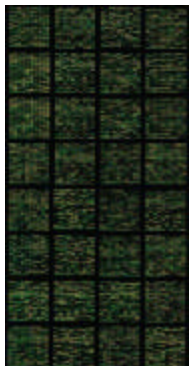
### Regulation of Genes



### Regulation of Genes



### Microarrays

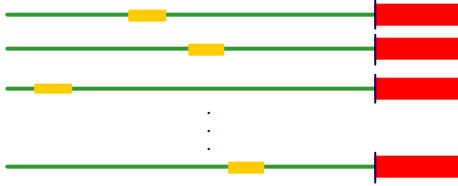


- A 2D array of DNA sequences from thousands of genes
- Each spot has many copies of same gene
- Allow mRNAs from a sample to hybridize
- Measure number of hybridizations per spot

## Finding Regulatory Motifs

Tiny Multiple Local Alignments of Many Sequences

## Finding Regulatory Motifs



Given a collection of genes with common expression,  
Find the TF-binding motif in common

## Characteristics of Regulatory Motifs

```

ATAA...TTT
CTGATA...CAAG
GTGA...TCA...
AGGCGG...AG...CG
AA...AA...AA...AA
TTT...T...AA...AA
G...AA...CG...TTGCG
...TTA...T...A
...TTA...T...A...A
...GGGACGG...
...AAATTT...
...GA...AA...AA
TAT...AA...TT...
...AA...AA...AAAA
TTT...AA...AA...AA
...T...T...AA...AA
...AT...AT...AT...AA
AT...AA...TT
    
```

- Tiny
- Highly Variable
- ~Constant Size
  - Because a constant-size transcription factor binds
- Often repeated
- Low-complexity-ish

## Problem Definition

Given a collection of promoter sequences  $s_1, \dots, s_N$  of genes with common expression

### Probabilistic

Motif:  $M_j$ ;  $1 \leq i \leq W$   
 $1 \leq j \leq 4$   
 $M_{ij} = \text{Prob}[\text{letter } j, \text{ pos } i]$

Find best  $M$ , and positions  $p_1, \dots, p_N$  in sequences

### Combinatorial

Motif  $M$ :  $m_1 \dots m_W$   
 Some of the  $m_i$ 's blank

Find  $M$  that occurs in all  $s_i$  with  $\leq k$  differences

## Essentially a Multiple Local Alignment



- Find "best" multiple local alignment

Alignment score defined differently in probabilistic/combinatorial cases

## Algorithms

- Probabilistic
  1. Expectation Maximization: MEME
  2. Gibbs Sampling: AlignACE, BioProspector
- Combinatorial
  - CONSENSUS, TEIRESIAS, SP-STAR, others

## Discrete Approaches to Motif Finding

## Discrete Formulations

Given sequences  $S = \{x^1, \dots, x^n\}$

- A motif  $W$  is a consensus string  $w_1 \dots w_K$
- Find motif  $W^*$  with "best" match to  $x^1, \dots, x^n$

Definition of "best":

$d(W, x) = \text{min hamming dist. between } W \text{ and a word in } x$

$d(W, S) = \sum_i d(W, x^i)$

## Approaches

- Exhaustive Searches
- CONSENSUS
- MULTIPROFILER, TEIRESIAS, SP-STAR, WINNOWER

## Exhaustive Searches

### 1. Pattern-driven algorithm:

For  $W = AA \dots A$  to  $TT \dots T$       ( $4^K$  possibilities)  
Find  $d(W, S)$   
Report  $W^* = \text{argmin}(d(W, S))$

Running time:  $O(K N 4^K)$   
(where  $N = \sum_i |x^i|$ )

Advantage: Finds provably best motif  $W$

Disadvantage: Time

## Exhaustive Searches

### 2. Sample-driven algorithm:

For  $W =$  a  $K$ -long word in some  $x^i$   
Find  $d(W, S)$   
Report  $W^* = \text{argmin}(d(W, S))$   
OR Report a local improvement of  $W^*$

Running time:  $O(K N^2)$

Advantage: Time

Disadvantage: If: True motif does not occur in data, and  
True motif is "weak"  
Then, random motif may score better than any  
instance of true motif

## CONSENSUS (1)

### Algorithm:

#### Cycle 1:

For each word  $W$  in  $S$   
For each word  $W'$  in  $S$   
Create alignment (gap free) of  $W, W'$

Keep the  $C_1$  best alignments,  $A_1, \dots, A_{C_1}$

ACGGTTG , CGAACTT , GGGCTCT ...  
ACGCCTG , AGAACTA , GGGGTGT ...

## CONSENSUS (2)

### Algorithm (cont'd):

#### Cycle 1:

For each word  $W$  in  $S$   
For each alignment  $A_j$  from cycle  $t-1$   
Create alignment (gap free) of  $W, A_j$

Keep the  $C_1$  best alignments  $A_1, \dots, A_{C_1}$

ACGGTTG , CGAACTT , GGGCTCT ...  
ACGCCTG , AGAACTA , GGGGTGT ...  
... , ... , ...  
ACGGCTC , AGATCTT , GGCGTCT ...

## CONSENSUS (3)

$C_1, \dots, C_n$  are user-defined heuristic constants

### Running time:

$$O(N^2) + O(N C_1) + O(N C_2) + \dots + O(N C_n) \\ = O(N^2 + N C_{\text{total}})$$

Where  $C_{\text{total}} = \sum_i C_i$ , typically  $O(nC)$ , where  $C$  is a big constant

## MULTIPROFILER

- Extended sample-driven approach

Given a  $K$ -long word  $W$ , define:

$$N_a(W) = \text{words } W' \text{ in } S \text{ s.t. } d(W, W') \leq a$$

### Idea:

Assume  $W$  is occurrence of true motif  $W^*$

Will use  $N_a(W)$  to correct "errors" in  $W$

## MULTIPROFILER (2)

Assume  $W$  differs from true motif  $W^*$  in at most  $L$  positions

### Define:

A wordlet  $G$  of  $W$  is a  $L$ -long pattern with blanks, differing from  $W$

### Example:

$K = 7$ ;  $L = 3$

$W = \text{ACGTTGA}$   
 $G = \text{--A--CG}$

## MULTIPROFILER (2)

### Algorithm:

For each  $W$  in  $S$ :

For  $L = 1$  to  $L_{\text{max}}$

- Find all "strong"  $L$ -long wordlets  $G$  in  $N_a(W)$
- Modify  $W$  by the wordlet  $G$   $\rightarrow W'$
- Compute  $d(W', S)$

Report  $W^* = \text{argmin } d(W', S)$

**Step 1 above:** Smaller motif-finding problem;  
Use exhaustive search

## Expectation Maximization in Motif Finding

## Expectation Maximization (1)

- The EM algorithm, part of MEME package uses Expectation Maximization

### Algorithm (sketch):

- Given genomic sequences find all  $K$ -long words
- Assume each word is **motif** or **background**
- Find **likeliest**  
Motif Model  
Background Model  
classification of words into either Motif or Background

## Expectation Maximization (2)

- Given sequences  $x^1, \dots, x^N$ ,
- Find all k-long words  $X_1, \dots, X_n$
- Define motif model:  
 $M = (M_1, \dots, M_k)$   
 $M_i = (M_{i1}, \dots, M_{i4})$  (assume {A, C, G, T})  
 where  $M_{ij} = \text{Prob}[\text{motif position } i \text{ is letter } j]$
- Define background model:  
 $B = B_1, \dots, B_4$   
 $B_j = \text{Prob}[\text{letter } j \text{ in background sequence}]$

## Expectation Maximization (3)

- Define  
 $Z_{i1} = \begin{cases} 1, & \text{if } X_i \text{ is motif;} \\ 0, & \text{otherwise} \end{cases}$   
 $Z_{i2} = \begin{cases} 0, & \text{if } X_i \text{ is motif;} \\ 1, & \text{otherwise} \end{cases}$
- Given a word  $X_i = x[1] \dots x[k]$ ,  
 $P[X_i, Z_{i1}=1] = \lambda M_{1x[1]} \dots M_{kx[k]}$   
 $P[X_i, Z_{i2}=1] = (1 - \lambda) B_{x[1]} \dots B_{x[k]}$   
 Let  $\lambda_1 = \lambda$ ;  $\lambda_2 = (1 - \lambda)$

## Expectation Maximization (4)

Define:  
 Parameter space  $\theta = (M, B)$

Objective:  
 Maximize log likelihood of model:

$$\begin{aligned} \log P(X_1 \dots X_n, Z | \mathbf{q}, \mathbf{I}) &= \sum_{i=1}^n \sum_{j=1}^2 Z_{ij} \log(I_j P(X_i | \mathbf{q}_j)) \\ &= \sum_{i=1}^n \sum_{j=1}^2 Z_{ij} \log P(X_i | \mathbf{q}_j) + \sum_{i=1}^n \sum_{j=1}^2 Z_{ij} \log I_j \end{aligned}$$

## Expectation Maximization (5)

- Maximize expected likelihood, in iteration of two steps:

Expectation:  
 Find expected value of log likelihood:

$$E[\log P(X_1 \dots X_n, Z | \mathbf{q}, \mathbf{I})]$$

Maximization:  
 Maximize expected value over  $\theta, \lambda$

## Expectation Maximization (6): E-step

Expectation:  
 Find expected value of log likelihood:

$$\begin{aligned} E[\log P(X_1 \dots X_n, Z | \mathbf{q}, \mathbf{I})] &= \\ \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij}] \log P(X_i | \mathbf{q}_j) &+ \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij}] \log I_j \end{aligned}$$

where expected values of Z can be computed as follows:

$$Z_{ij} = \frac{I_j P(X_i | \mathbf{q}_j)}{\sum_{k=1}^2 I_k P(X_i | \mathbf{q}_k)}$$

## Expectation Maximization (7): M-step

Maximization:  
 Maximize expected value over  $\theta$  and  $\lambda$  independently

For  $\lambda$ , this is easy:

$$I_j^{NEW} = \arg \max_{I_j} \sum_{i=1}^n E[Z_{ij}] \log I_j = \sum_{i=1}^n \frac{Z_{ij}}{n}$$

## Expectation Maximization (8): M-step

- For  $\theta = (M, B)$ , define

$c_{jk} = E[\text{\# times letter } k \text{ appears in motif position } j]$

$c_{0k} = E[\text{\# times letter } k \text{ appears in background}]$

It easily follows:

$$M_{jk}^{NEW} = \frac{c_{jk}}{\sum_{k=1}^4 c_{jk}} \quad B_k^{NEW} = \frac{c_{0k}}{\sum_{k=1}^4 c_{0k}}$$

to not allow any 0's, add pseudocounts

## Initial Parameters Matter!

Consider the following "artificial" example:

$x^1, \dots, x^N$  contain:

- $2^K$  patterns  $A\dots A, A\dots AT, \dots, T\dots T$
- $2^K$  patterns  $C\dots C, C\dots CG, \dots, G\dots G$
- $D \ll 2^K$  occurrences of K-mer  $ACTG\dots ACTG$

Some local maxima:

$$\lambda_i = 1/2; \quad B = 1/2 C, 1/2 G; \quad M_i = 1/2 A, 1/2 T, \quad i = 1, \dots, K$$

$$\lambda = D/2^{K+1}; \quad B = 1/4 A, 1/4 C, 1/4 G, 1/4 T; \\ M_1 = 100\% A, M_2 = 100\% C, M_3 = 100\% T, \text{ etc.}$$

## Overview of EM Algorithm

1. Initialize parameters  $\theta = (M, B), \lambda$ :
  - Try different values of  $\lambda$  from  $N^{-1/2}$  upto  $1/(2K)$
2. Repeat:
  - a. Expectation
  - b. Maximization
3. Until change in  $\theta = (M, B)$ ,  $\lambda$  falls below  $\epsilon$
4. Report results for several "good"  $\lambda$

## Conclusion

- One iteration running time:  $O(NK)$ 
  - Usually need  $< N$  iterations for convergence, and  $< N$  starting points.
  - Overall complexity: unclear – typically  $O(N^2 K) - O(N^3 K)$
- EM is a local optimization method
- Initial parameters matter

MEME: Bailey and Elkan, ISMB 1994.