Université Catholique de Louvain

Master en Sciences de l'informatique

Intelligence Artificielle

LGBIO2010: Bioinformatics

Author:
Denis Genon

Contents

1 Sequences Statistics

1.1 Simple Genome Statistics

1.1.1 GC-content

GC-content is the percentage of G or C in a DNA sequence.

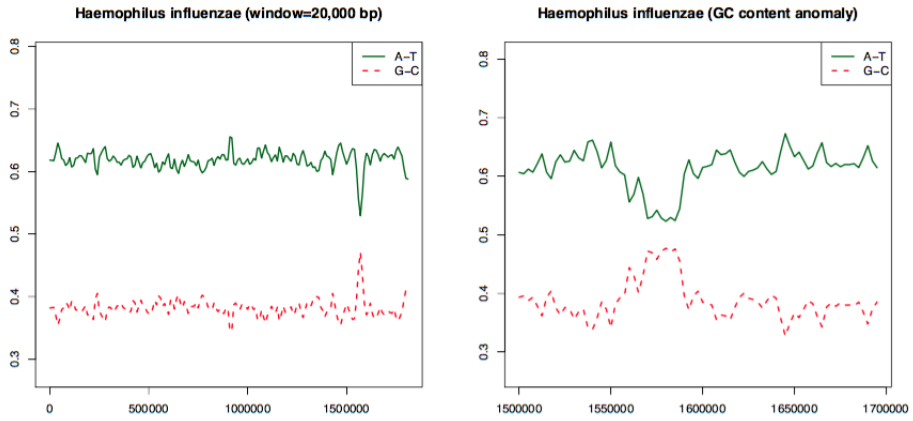


Figure 1: GC-Content. Anomaly due to an ancient insertion of viral DNA. AT-rich regions denature at lower temperature. The ability to quickly denature DNA facilitates the insertion in the bacterial cell being infected.

1.1.2 Dimer frequencies

Some dimers are particulary frequent but it is due to have each nitrogenous bases frequent. Need a background model.

- Estimate the probability of each nucleotide independently
  $\hat{P}(A) = f(A) = \frac{\text{number of A's}}{\text{sequence length}} = 0.3102$
- Use those estimates in a random generative model
  `AAGTTGACATAATTTGCT...`
- The expected frequency of a dimer XY:

$$E[f(XY)] = \hat{P}(X)\hat{P}(Y) = f(X)f(Y)$$

Figure 2: Multinomial background model.

With the multinomial background model, we can compute the **odd ratios**: ratio between observed frequency and expected frequency: $\frac{f(XY)}{E[f(XY)]} = \frac{f(XY)}{f(X)f(Y)}$.

The multinomial model is equivalent to a random permutation of the original sequence: $\frac{f(XY)}{f_{random}(XY)}$.

We can generalize odd ratios to **k-mers**.

$$\frac{f(X_1 \dots X_k)}{E[f(X_1 \dots X_k)]} = \frac{f(X_1 \dots X_k)}{\prod_{i=1}^{k} f(X_i)}$$

Figure 3: k-mers. Useful when looking for frequent patterns of k consecutive characters. But are they informative ? Need a more sophisticated background model, a statistical test to assess signifiance and biological validation.

1.2 Open Reading Frames

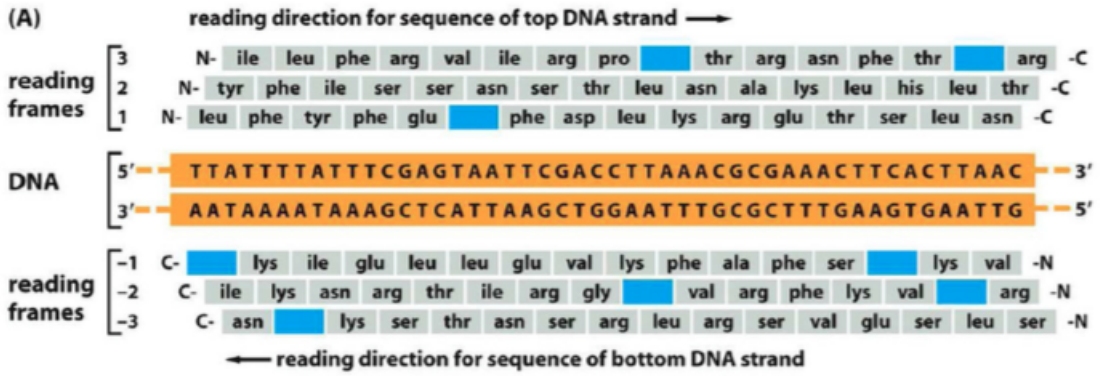


Figure 4: Reading frames.

1.2.1 Algorithm to find ORF

repeat *along the sequence*

1. look for a first START
   (or the next START after a STOP on the same frame)
2. look for the next STOP on the same reading frame

Figure 5: Need to consider each reading frames (6). After a START, you may find other codons for Met before a STOP: not true START. An ORF is a longest strech of DNA between a START and a STOP, without being interrupted by another STOP on the same frame.

The problem is to have to be sure that the ORF is a coding gene.

- The DNA found between STAT and STOP codon might be due to chance: need **a statistical test procedure**;
- The ORF might be there but the gene not expressed (trace of past or regulations at transcription/translation levels): need biological validation;
- Some gene sequences do not strictly follow the standard ORF structure: need biological validation.

1.2.2 Signifiance assessment

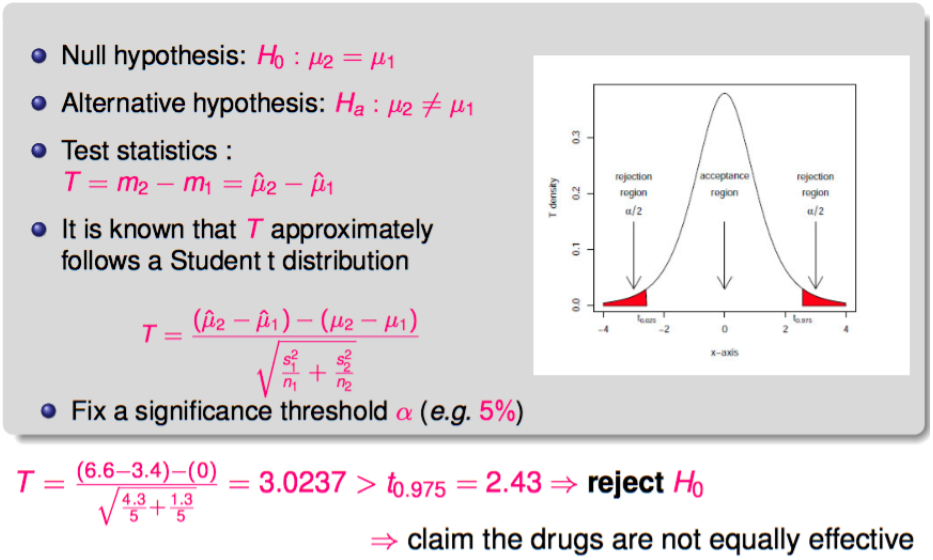


Figure 6: Statistical test.

To avoid fixing a signifiance threshold α we can compute the **p-value** of the test.

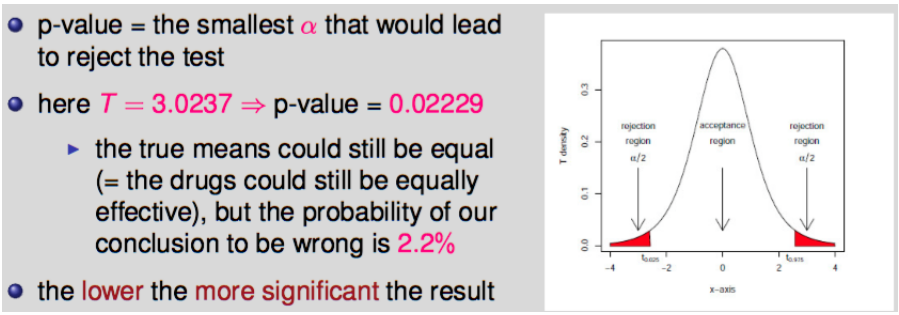


Figure 7: p-value.

We can apply this to ORF.

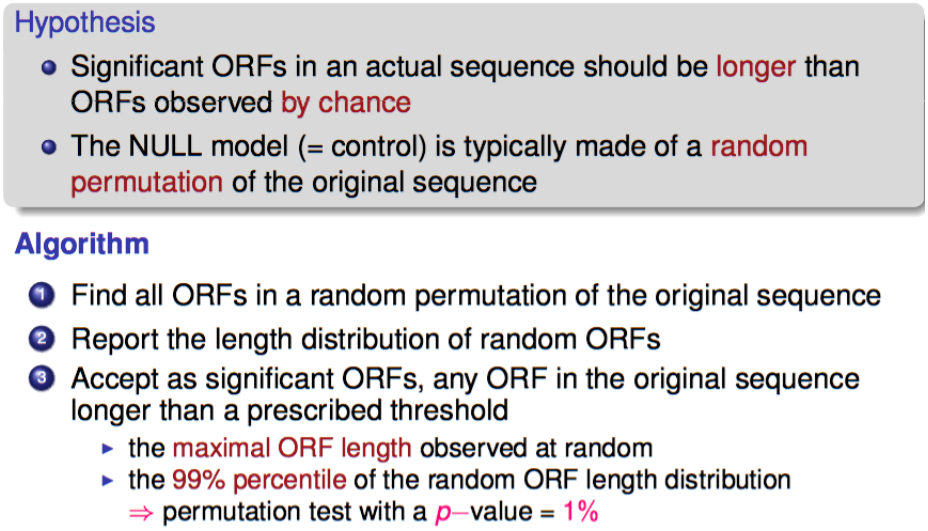


Figure 8: Algorithm to assess signifiance of ORF.

2 Pairwise Alignments

2.1 The problem

Find a best way to align two sequences including matches, substitutions and possible gaps.

2.1.1 Gap penalties

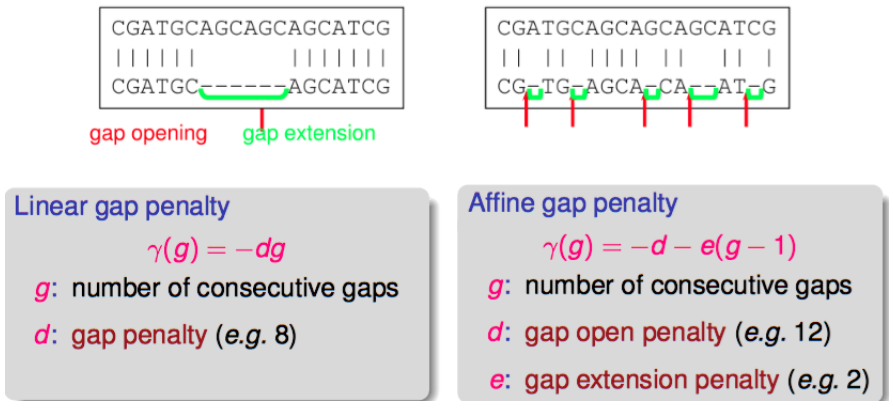


Figure 9: Gap penalties. Afine is more relevant.

2.1.2 Number of alignements

The total number alignements is huge ($\frac{4^n}{\sqrt{\pi * n}}$). We need dynamic programming.

- We look for maximal cumulative score;
- An optimal global alignment is made of optimal alignments between subsequences: decomposition of sub-problems computed only once.

2.2 Global alignment: Needle-Wunsch

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

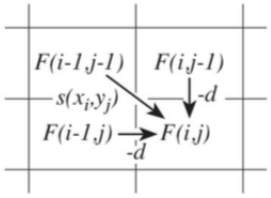


Figure 10: $F(n, m)$ is final alignment score. Need to follow backpointers. $O(n^2)$.

2.3 Local alignment: Smith-Waterman

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

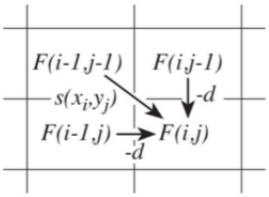


Figure 11: Find maximal $F(i, j)$ and follow backpointers till 0. Need negative scores for strong mismatches. $O(n^2)$.

2.4 Variants

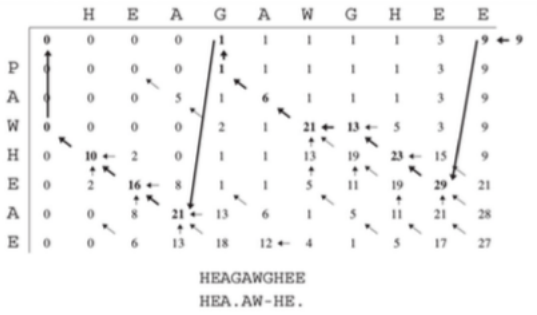
2.4.1 Semi-global alignment

Do not penalize gaps at the beginning of either sequences

- Initialize $F(i,0) = F(0,j) = 0 \;\; ; \;\; 0 \leq i \leq n \;\; ; \;\; 0 \leq j \leq m$
- Compute the global recurrence

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

2.4.2 Local alignment with repeats



$$F(0,0) = 0$$

$$F(i,0) = \max \begin{cases} F(i-1,0) \\ F(i-1,j) - T \end{cases} \quad \forall j = 1, \ldots, m$$

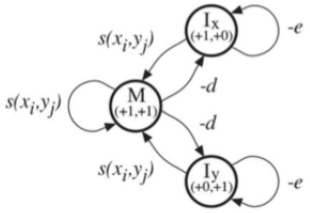
$$F(i,j) = \max \begin{cases} \mathbf{F(i,0)} \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Figure 12: Look for several matches (subsequences) of y into x. Scoring higher than T.

2.4.3 Affine gap penalty

Instead of a single state $F(i,j)$, one distinguishes 3 states

I G A x_i — A I G A x_i — G A x_i — —
L G V y_j — G V y_j — — — S L G V y_j



$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i,y_j) \\ I_x(i-1,j-1) + s(x_i,y_j) \\ I_y(i-1,j-1) + s(x_i,y_j) \end{cases}$$

$$I_x(i,j) = \max \begin{cases} M(i-1,j) - d \\ I_x(i-1,j) - e \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1) - d \\ I_y(i,j-1) - e \end{cases}$$

M state : x_i aligned to y_j
I_x state : x_i aligned to a gap
I_y state : y_j aligned to a gap

2.5 Signifiance assessment

1. Compute pairwise random alignment scores
   - between x and random sequences (*e.g.* 500 permutations of y)
   - between y and random sequences (*e.g.* 500 permutations of x)
2. Compute histogram of random alignment scores (normalized by length)
3. *p-val* = 1− percentile of actual score between x and y

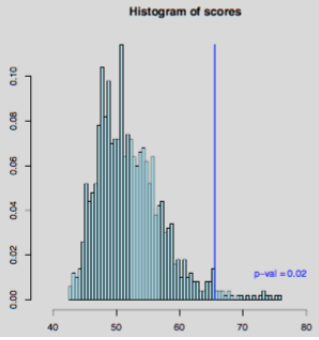


Figure 13: Procedure.

The actual distribution of alginment scores is called **extreme value distribution**.

- Probability of a score larger than S
  $P(x > S) = 1 - e^{(-Kmn\, e^{-\lambda S})}$
  for some constants K and λ
- Alignment softwares include fitted values of K and λ for a wide range of substitution scoring matrices
- The lack of normality is due to non-independence between possible starting points of matches

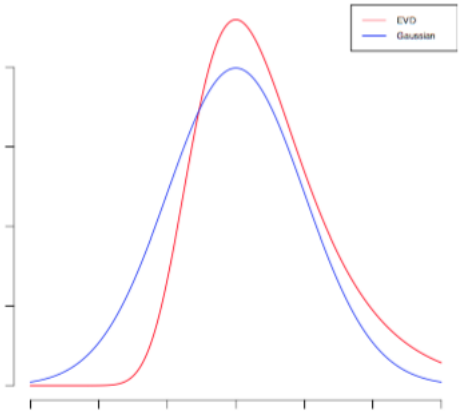


Figure 14: K is a multiplicative factor correcting non independance of possible starting poins for matches. λ is a scale parameter.

3 HMMs

3.1 An example

An motivating example is to determine coding or non-coding fragments.

3.1.1 Baseline approach

1. Find all ORFs in the original sequence
2. Find all ORFs in random permutation(s) of the original sequence
3. Accept as significant ORFs, any ORF in the original sequence longer than a prescribed length: *e.g.* the 99% percentile of random ORF lengths (p−value = 1%)

Limitations

- Coding vs non-coding fragments are not only characterized by their length
- Intron-exon boundaries also need to be identified for eukaryotes

3.1.2 Alternative approach

Learning

Given some labeled data

- estimate a statistical model M_+ for **coding** fragments (or exons)
- estimate a statistical model M_- for **non-coding** fragments (or introns)

Segmentation

For any new sequence to analyze

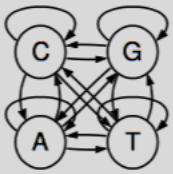
- look at a subsequence x defined by a sliding window of size L
- compute log-odds ratio

$$\text{If } \log \frac{P(x|M_+)}{P(x|M_-)} = \log \frac{\prod_{i=1}^{L} P(x_i|M_+)}{\prod_{i=1}^{L} P(x_i|M_-)} = \sum_{i=1}^{L} \log \frac{P(x_i|M_+)}{P(x_i|M_-)} > 0$$

then decide x is **coding**
else decide x is **non-coding**

3.1.2.1 Multinomial model

An equivalent representation



- each symbol is generated on a specific state
- all transition probabilities pointing to the same state are equal
- one such model for each segment type (*e.g.* coding vs non-coding)

Figure 15: Two hypothesis: the frequencies are the sames along the different parts of the sequence and each symbol does not depend of surrounding symbols.

3.1.2.2 Markov chain

- Each transition probability $P(x_i|x_{i-1}, M)$ is now specific to a dimer

$$\hat{P}(x_i|x_{i-1}, M) = \frac{f(x_{i-1}x_i)}{f(x_{i-1})} \text{ from segments modeled by } M$$

- Log-odds computation

$$\log \frac{P(x|M_+)}{P(x|M_-)} = \sum_{i=1}^{L} \log \frac{\hat{P}(x_i|x_{i-1}, M_+)}{\hat{P}(x_i|x_{i-1}, M_-)}$$

Figure 16: First order.

- Each transition probability $P(x_i|x_{i-1}, x_{i-2}, M)$ is specific to a 3-mer

$$\hat{P}(x_i|x_{i-1}, x_{i-2}, M) = \frac{f(x_{i-2}x_{i-1}x_i)}{f(x_{i-2}x_{i-1})} \text{ from segments modeled by } M$$

- Log-odds computation

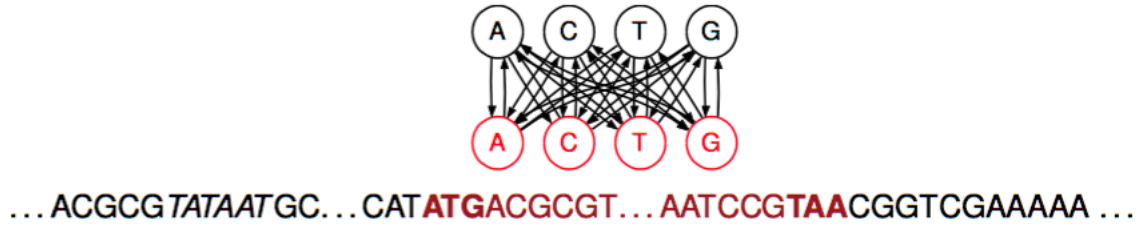
$$\log \frac{P(x|M_+)}{P(x|M_-)} = \sum_{i=1}^{L} \log \frac{\hat{P}(x_i|x_{i-1}, x_{i-2}, M_+)}{\hat{P}(x_i|x_{i-1}, x_{i-2}, M_-)}$$

Figure 17: Second order.

Limitations of the markov chain approach:

- Estimate each MC from well annoted segments;
- Gene length variability: sliding window length is arbitrary and we need to find a length that is more relevant: computation expensive;
- Eukaryotes: at least three models needed (out of gene, introns, exons).

3.1.2.3 Hidden Markov model



Learning

- Define a single model for all possible segments
  Note: *not all transitions are depicted here*
- Estimate all transition probabilities simultaneously

Segmentation

Find the most likely state sequence to generate the whole sequence

Figure 18: One state for each segment type (coding vs non coding). Transition probabilities, emissions probabilities. States are hidden but their emissions are observed.

3.2 Hidden Markov model

$$\Sigma = \{a, b\} \qquad Q = \{1, 2\}$$

$$\mathbf{A} = \begin{bmatrix} 0.1 & 0.9 \\ 0.7 & 0.3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{bmatrix} \qquad \pi = \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$$

A discrete HMM (with state emission)

- Σ is a finite alphabet
- Q is a set of states
- $\mathbf{A}$ a $|Q| \times |Q|$ transition probability matrix ($\sum_{q' \in Q} \mathbf{A}_{qq'} = 1$)
- $\mathbf{B}$ a $|Q| \times |\Sigma|$ emission probability matrix ($\sum_{a \in \Sigma} \mathbf{B}_{qa} = 1$)
- π an initial probability distribution ($\sum_{q \in Q} \pi_q = 1$)

Figure 19: HMM definition.

3.2.1 Path likelihood

The likelihood $P(s, \nu|M)$ of a sequence $s = s_1 \dots s_{|s|}$ along a path or state sequence $\nu = q_1 \dots q_{|s|}$ in a HMM M

$$P(s, \nu|M) = \prod_{i=1}^{|s|} P(s_i, q_i|M) = \pi_{q_1} \mathbf{B}_{q_1 s_1} \prod_{i=2}^{|s|} \mathbf{A}_{q_{i-1} q_i} \mathbf{B}_{q_i s_i}$$

3.2.2 Probability of generating a sequence

The likelihood $P(s|M)$ of a sequence $s = s_1 \dots s_{|s|}$ in an HMM M

$$P(s|M) = \sum_{\nu \in Q^{|s|}} P(s, \nu|M)$$

Figure 20: $O(|Q|^{|s|})$ possible state sequences. Q is states and s is sequence.

3.2.3 Most likely state sequence

$$\nu^* = \text{argmax}_\nu \, P(s, \nu | M)$$

Auxiliary quantity: $\gamma(k,t) = P(s_1 \dots s_t, \nu_t^* = k | M)$
The probability of a most likely path ν^* reaching state k at step t

Initialization: $\gamma(k,1) = \pi_k \mathbf{B}_{ks_1}$
Recurrence: $\gamma(k,t) = \max_l [\gamma(l,t-1)\mathbf{A}_{lk}]\mathbf{B}_{ks_t}$
Termination: $P(s, \nu^* | M) = \max_l \gamma(l, |s|)$

Figure 21: Viterbi recurrence.

- $P(s, v^*)$ gives the probability of the optimal path v^*;
- Computation are done with log (because too small values);
- Complexity: $\Theta(m|s|)$. m is number of HMM transitions;
- Path v^* is an alignement between states and words;
- v^* can be recovered with the backpointers.

3.2.4 Sequence likelihood

$$P(s|M) = \sum_\nu P(s, \nu | M)$$

Auxiliary quantity: $\alpha(k,t) = P(s_1 \dots s_t, \nu_t = k \mid M)$
The likelihood of emitting the first t symbols and reaching state k at time t

Initialization: $\alpha(k,1) = \pi_k \mathbf{B}_{ks_1}$
Recurrence: $\alpha(k,t) = \sum_l [\alpha(l,t-1)\mathbf{A}_{lk}]\mathbf{B}_{ks_t}$
Termination: $P(s|M) = \sum_l \alpha(l, |s|)$

Figure 22: Forward recurrence, same complexity as viterbi recurrence.

3.3 The learning problem

Given an HMM structure and several sequences, you must estimate A, B, π.

3.3.1 Supervised learning

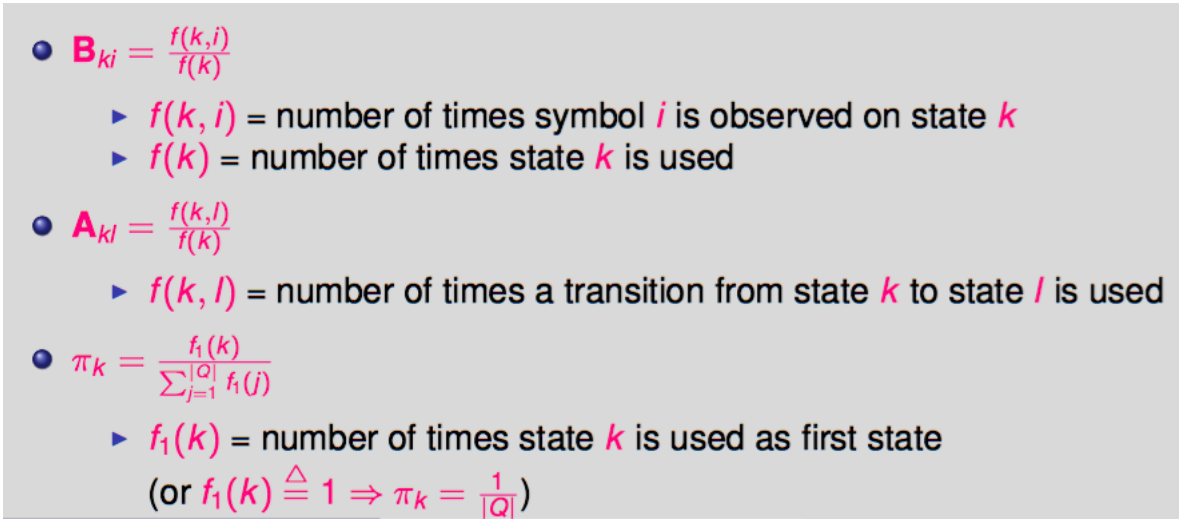


Figure 23: Supervised learning.

3.3.2 Unsupervised learning

The sentences are not annoted. There is two algorithm. The first one is **Viterbi training**.

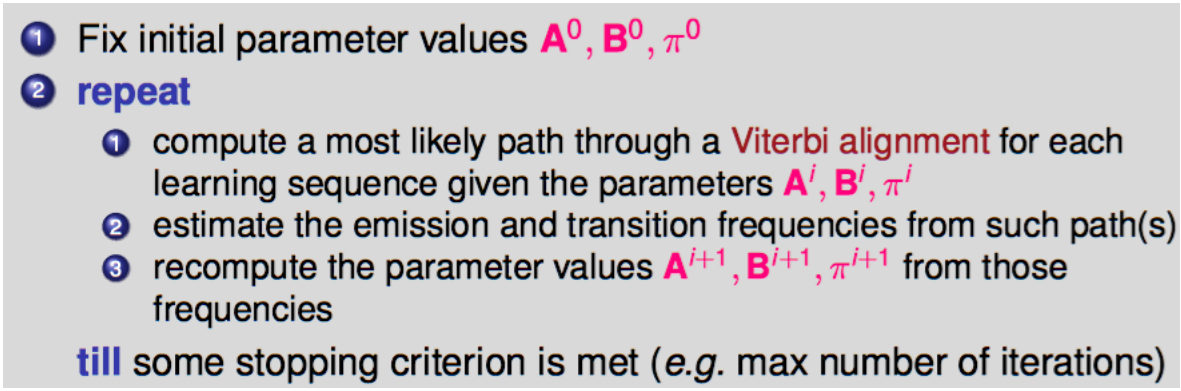


Figure 24: Viterbi learning.

The second is **Forward-Backward** or **Braaum-Welch** algorithm. Viterbi training is an approximation as it considers that each training sentence is generated along a single path. A more accurate estimation is obtained if one considers all possible paths to generate each sentence.

- actual frequencies are replaced by expected frequencies;

- special case of expectation-maximization (EM) procedure

Viterbi and Baum-Welch training are both sensitive to parameter initialization.

3.4 Concrete example

For CpG islands, we can use a 2-state HMM.

- GC-rich state: higher probability to emit G or C
- alternative state: lower probability to emit G or C
- transition probabilities reflect that is much more likely (*e.g.* 0.998) to stay in a given state than switching regime (*e.g.* 0.002)
- initial probabilities reflect the chance to start or not with a CpG island (by default, $\pi_1 = \pi_2 = 0.5$)

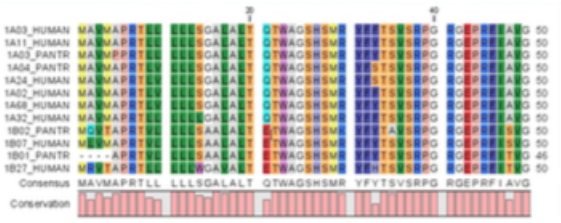
Those probabilities can either be *a priori* fixed or estimated through supervised learning, Viterbi or Baum-Welch

4 Multiple Alignments

4.1 The problem

Align three or more homologuous sequences (globally or localy).

4.1.1 Scoring



Assumption

- Individual columns are assumed statistically independent
- A multiple alignment m with L columns can then be scored as

$$S(m) = G + \sum_{i=1}^{L} S(m_i)$$

  - $S(m_i)$ = score for column i
  - G = score for all gaps in m using linear or affine gap penalties

Figure 25: Assumptions.

Sum of pairs

- Column score:

$$S(m_i) = \sum_{k<l} s(m_i^k, m_i^l)$$

  where $s(a, b)$ is given by a substitution scoring matrix (*e.g.* PAM or BLOSUM)

- Gap penalty
  - linear: $s(a, -) = s(-, b) = -d$; $s(-, -) = 0$
  - affine: all gaps are scored separately

Figure 26: Sum of pairs.

4.2 Dynamic programming

Optimal alignment between k sequences can be computed with DP. With an average length of $\bar{n}$, the time complexity is $O(2^k \bar{n}^k)$ and the space complexity is $O(\bar{n}^k)$ (hyper cube).

One algorithm which compute optimally the alignement is **MSA** algorithm ($O(k^2 \bar{n}^2)$: 10 sequences of 200-300 residues in reasonnable time).

- Compute all pairwise alignments;

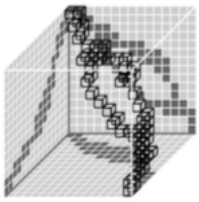


Figure 27: MSA.

- Limits the exploration of the hyper cube to regions consitent with those alignments.

G-MSA is an optimization: 500 sequences of 236 residues in 10 seconds.

4.2.1 Progressive alignment methods (greedy heuritic)

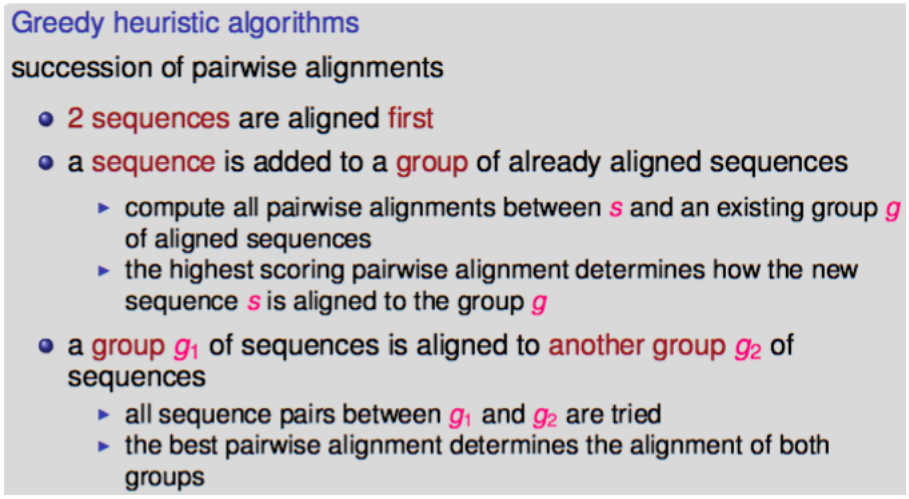
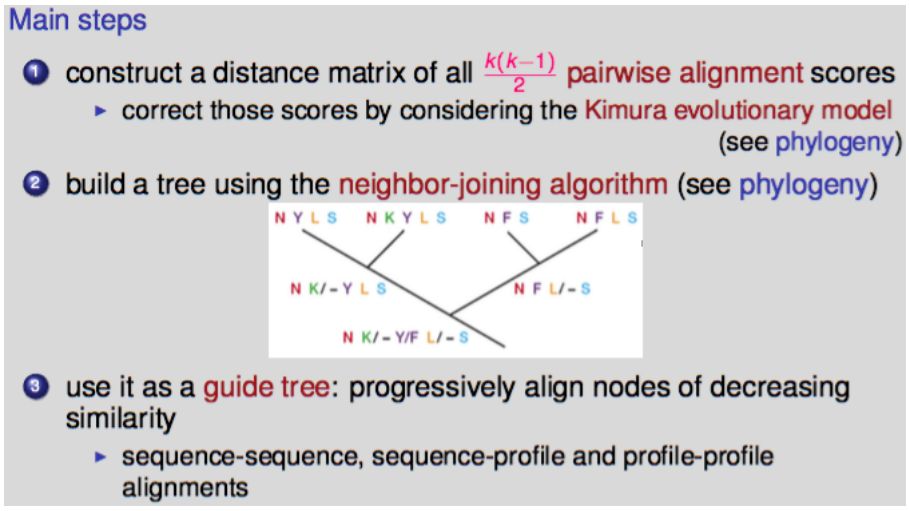


Figure 28: Greedy heuritic.

There are some limitations:

- the degree of sequence conservation at each position should be taken into account;
- mismatches at highly conserved positions should be more penalized;
- the order in which sequences are incorporated in the multiple alignment matters.

4.2.1.1 ClustalW



There can be further heuristics:

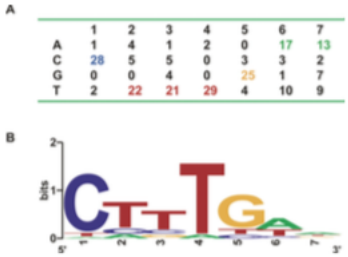
- **position specific gap open penalties with decreased penalties** wherever other gaps have been found amoung already aligned sequences;
- gap penalties are also decreased or increased based on a large collection of **structural alignments**;
- the guide tree may be ajusted **on the fly** to defer a low scoring alignment until more profile information has been accumulated.

5 Profile HMMs

5.1 Motivations

- multiple alignments are most often based on pairwise alignments;
- a new sequence x may be only distantly and locally related to each sequence in a known family (biological question): all pairwise alignments between x and each family members will look poor. We need to model **statistical features** shared by the family members;
- computing the alignment between x and a probabilistic model of the family may be much **more efficient computationally**.

5.2 Position specific scoring matrices



- Local model for a window length L and ungapped score matrix from N sequences

$$P(x|M) = \prod_{i=1}^{L} P(x_i|M) = \prod_{i=1}^{L} \frac{f(x_i)}{N}$$

- Log-odds score

$$S = \sum_{i=1}^{L} \log \frac{P(x_i|M)}{q_{x_i}}$$

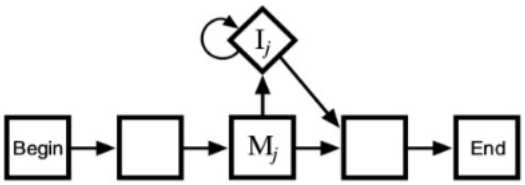
  - q_{x_i} = the background model (*e.g.* multinomial model)
- One evaluates the score S between x and M for all positions x_i and a sliding window of size L

Figure 29: PSSMs are very simple HMMs. $P(x_i|M)$ are emission probabilities on match state. Transition probabilities are equal to 1. But we need to account for possible gap and avoid a prescribed window length.

$$P(x|M) = \prod_{i=1}^{L} P(x_i|M)$$

5.3 Full profile HMMs

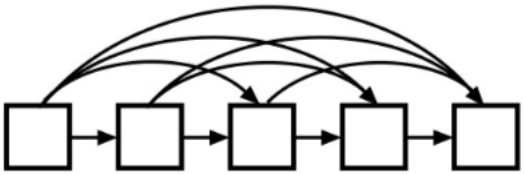
5.3.1 Adding insert states



- to account for portions of x that do not match the model
- emission probabilities on insert states are typically defined through the background model
  - no contribution to the log-odds score $\log \frac{P(x_i|I_j)}{q_{x_i}} = \log \frac{q_{x_i}}{q_{x_i}} = 0$
- transition probabilities to insert states and back are equivalent to affine gap penalties $\log \mathbf{A}_{M_j I_j} + \log \mathbf{A}_{I_j M_{j+1}} + (k-1) \log \mathbf{A}_{I_j I_j}$

5.3.2 Adding delete states

- portions of the model M that are not matched by any residue x_i could be modeled by skipping transitions



- to allow arbitrary long gaps it is more convenient to introduce delete states which are silent states

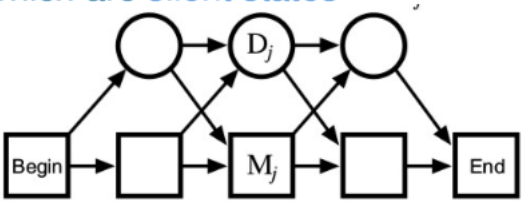


Figure 30: More convenient because less transitions.

5.3.3 A full profile HMM

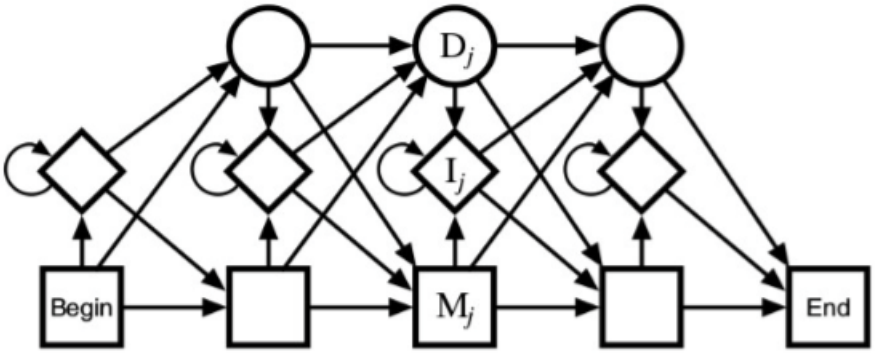
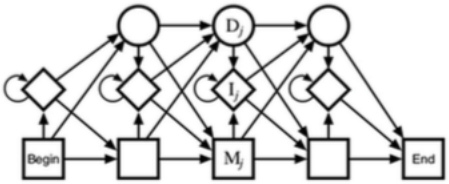


Figure 31: Parameters are: value of the probabilities and length. Begin, End and D states are silent.

5.3.4 Deriving a pHMM from a multiple alignement (Supervised)




```
HBA_HUMAN    ...VGA--HAGEY...
HBB_HUMAN    ...V----NVDEV...
MYG_PHYCA    ...VEA--DVAGH...
GLB3_CHITP   ...VKG------D...
GLB5_PETMA   ...VYS--TYETS...
LGB2_LUPLU   ...FNA--NIPKH...
GLB1_GLYDI   ...IAGADNGAGV...
                ***  *****
```

- a match state for each conserved position (*e.g.* at least 50%)
- insert states: *e.g.* columns with at least 50% gaps
- delete states: gaps on match positions
- emission (for match states) and transition probabilities are estimated from the counts
  - $\mathbf{B}_{ki} = \frac{f(k,i)}{f(k)}$
    - $f(k, i)$ = number of times symbol i is observed on state k
    - $f(k)$ = number of times state k is used
  - $\mathbf{A}_{kl} = \frac{f(k,l)}{f(k)}$
    - $f(k, l)$ = number of times a transition from state k to state l is used

But the initial multiple alignement could have some probabilities equal to zero: we need **smoothing**.

Additive smoothing with pseudo-counts

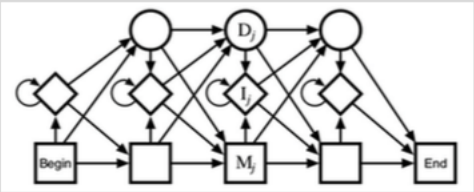
- $\mathbf{B}_{ki} = \frac{f(k,i)+\varepsilon}{f(k)+\sum_i \varepsilon}$ with *e.g.* $10^{-6} \leq \varepsilon \leq 1$
  - $f(k,i)$ = number of times symbol i is observed on state k
  - $f(k)$ = number of times state k is used
- $\mathbf{A}_{kl} = \frac{f(k,l)+\varepsilon'}{f(k)+\sum_l \varepsilon'}$ with *e.g.* $10^{-6} \leq \varepsilon' \leq 1$
  - $f(k,l)$ = number of times a transition from state k to state l is used

5.3.5 Unsupervised learning

No need for an initial multiple alignment (only unaligned sequences).

Procedure

1. choose a general pHMM structure



2. choose the number of match states:
   e.g. half the average sequence length
3. estimate the pHMM parameters through Viterbi or Baum-Welch

5.3.6 Matching a sequence to a pHMM a.k.a. viterbi recurrence

- Computations are done usually with log's:
  $$-\log\gamma(k,t) = \min_l[-\log\gamma(l,t-1) - \log\mathbf{A}_{lk}] - \log\mathbf{B}_{kx_t}$$
- Including a background model to produce a log-odds score:
  $$\log\gamma(k,t) = \max_l[\log\gamma(l,t-1) + \log\mathbf{A}_{lk}] + \log\frac{\mathbf{B}_{kx_t}}{q_{x_t}}$$
- A similar adaptation can be included into the forward recurrence

5.3.7 For non-global alignments

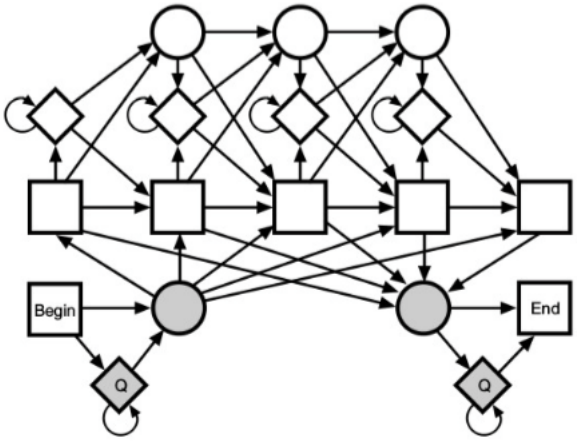


Figure 32: Local alignment (S-W style). Non-conserved fragments are modeled through flanking insert states using the backgroud emission probabilities q_a. Their loop have high probabilities. Flanking delete states allow for starting or ending the profile at any point and reduce number of transitions.

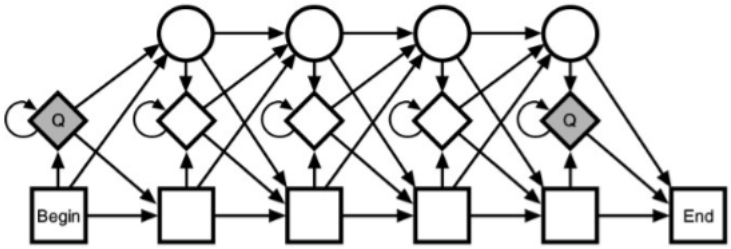


Figure 33: Set all the transition probabilities from the left flanking state to different start points. Force the match on the compelte profile.

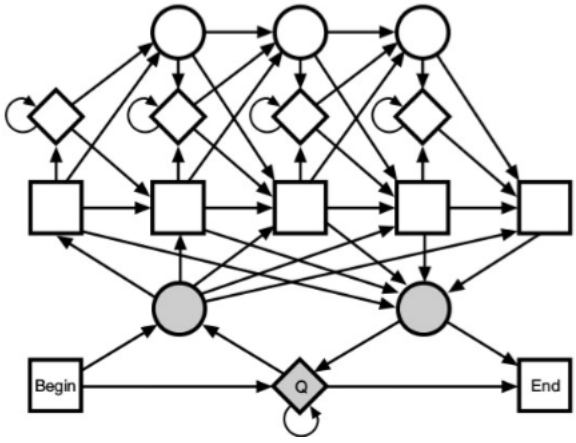


Figure 34: Allow repeated matches to subsections of the profile.

6 Large Scale Gene Expression Analysis

6.1 Introduction

DNA Microarrays measure the level of expression of all genes in a signle experiment. Data measurements → Preprocess and sample normalization → Gene selection and sample classification → Diagnosis, prognosis or prediction of the reponse to a treatment.

The **gene selection** (supervised learning) to find a subset of genes to predict the response of new samples. There are some **objectives**:

- Insight into the data and the predictive model;
- Link between data analysis and medical expert;
- Biological validation;
- Reduce financial cost.

There are some **difficulties**:

- Measurements are noisy;
- Gene expression varies due to many factors;
- Financial cost;
- Small n large p problem.

6.2 Preprocessing

6.2.1 Summarization

Define a single probeset expression level from the various probe intensities. There are popular techniques: MAS 5.0, RMA, GC-RMA.

- background adjustment: optical noise correction, probe affinity adjustment (influenced by the GC content), RMA ignores the MM probes;
- sample normalization: quantiles should be stable across samples, after conversion to log intensities for (GC-)RMA;
- summarization: median polish.

6.2.2 Feature normalization

Make sure that each gene (probeset) has roughly the same expression range across all samples and then do a **Z-score normalization**: Replace $x_{i,j}$ by $\frac{x_{i,j}-\mu_j}{s_j}$. μ_j is the mean level of expression of probeset j over the training samples and s_j is the tandard deviation.

6.2.3 Distance between expression values

Euclidean distance

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = \sqrt{\sum_{i=1}^{n}(x_{i,1} - x_{i,2})^2}$$

Correlation based distance

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = 1 - \frac{1}{2}(1 + \text{corr}(\boldsymbol{x}_1, \boldsymbol{x}_2))$$

6.2.3.1 Pearson correlation

For two random vectors (*e.g.* gene expression values) $\boldsymbol{x}_1, \boldsymbol{x}_2$ measured over n samples

$$\text{corr}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\sum_{i=1}^{n}(x_{i,1} - \bar{x}_1)(x_{i,2} - \bar{x}_2)}{\sqrt{\sum_{i=1}^{n}(x_{i,1} - \bar{x}_1)^2 \sum_{i=1}^{n}(x_{i,2} - \bar{x}_2)^2}}$$

- $\text{corr}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \pm 1$ if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are perfectly linearly correlated
- $\text{corr}(\boldsymbol{x}_1, \boldsymbol{x}_2) = 0$ if they are not linearly correlated
- whenever $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and normalized to zero mean and unit variance:

$$\text{corr}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{n} x_{i,1} x_{i,2} = \boldsymbol{x}_1^\top \boldsymbol{x}_2$$

6.2.3.2 Pitfalls with correlation measure

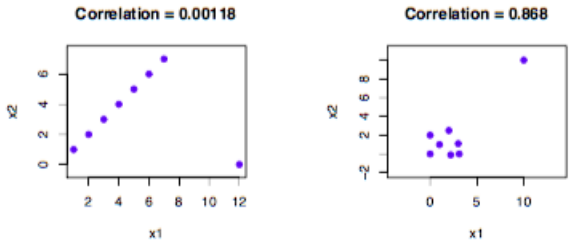


Figure 35: Correlation is sensitive to outliers.

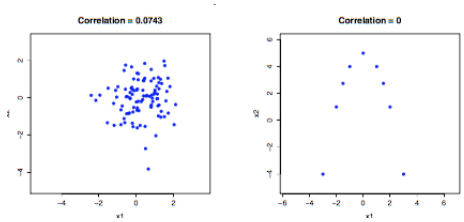


Figure 36: Correlation measures **linear** dependence.

6.2.3.3 Spearman's rank correlation This correlation is less sensitive to outliers (but still measure linear dependence).

- Replace feature value by feature value rank across observations;
- Compute pearson correlation between rank vectors.

6.3 Unsupervised learning

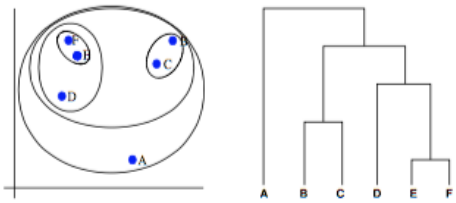
	gene 1	gene 2	...	gene p	cluster
sample 1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,p}$	?
...	...	...	...	...	...
sample n	$x_{n,1}$	$x_{n,2}$	...	$x_{n,p}$	?
cluster	?	?	...	?	

Figure 37: Objective is to find cluster of genes that share similar profile.

Each observation represents

either a sample across genes

or a gene across samples



Algorithm AGGLOMERATIVEHIERARCHICALCLUSTERING
Input: D a set of observations $\vec{x}_1, \ldots, \vec{x}_m$; $d(\vec{x}, \vec{y})$ a distance measure between observations
Output: A tree T of subsets of D
// Initialize a set $\mathcal{D}$ of clusters $D_1, \ldots, D_m$
$\mathcal{D} \leftarrow \{\{\vec{x}_1\}, \ldots, \{\vec{x}_m\}\}$ // Initial clusters are tree leaves
$T \leftarrow$ a partial tree whose leaves are the $\vec{x}_i$'s
while $|\mathcal{D}| > 1$ **do**
  Choose pair of clusters (D_i, D_j) in $\mathcal{D}$ such that $Distance(D_i, D_j, d)$ is minimal
  Define a new cluster $D_k = D_i \cup D_j$
  $\mathcal{D} \leftarrow \mathcal{D} \cup D_k - \{D_i, D_j\}$
  Add D_k as parent node of D_i and D_j in the tree T
return T

Figure 38: Agglomerative hierarchical clustering.

There are different distance measure between the clusters.

Single-link or nearest neighbor rule

$$Distance(D_i, D_j, d) = \min_{\vec{x} \in D_i, \vec{y} \in D_j} d(\vec{x}, \vec{y})$$

Complete-link or farthest neighbor rule

$$Distance(D_i, D_j, d) = \max_{\vec{x} \in D_i, \vec{y} \in D_j} d(\vec{x}, \vec{y})$$

Average-link rule

$$Distance(D_i, D_j, d) = \frac{1}{|D_i| . |D_j|} \sum_{\vec{x} \in D_i, \vec{y} \in D_j} d(\vec{x}, \vec{y})$$

UPGMA algorithm is with average-link measure. Used in phylogeny.

6.4 Supervised learning

	gene 1	gene 2	...	gene p	response
sample 1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,p}$	y_1
...	...	...	...	...	...
sample n	$x_{n,1}$	$x_{n,2}$	...	$x_{n,p}$	y_n
test sample	x_1	x_2	...	x_p	?

Figure 39: p input dimensions, n samples, y is binary, natural or real. THe goal is to find the most discriminating genes for the prediction of the class of any new sample.

6.4.1 Filters

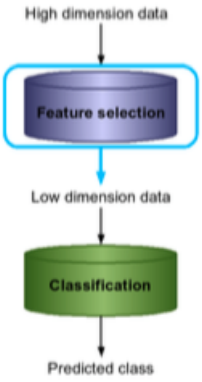


Figure 40: Use only training data and class labels during the features selection step and then train a single classifier taking the selected features as inputs. It is the less computive approach.

6.4.1.1 Non-specific filtering Keep only genes with the larger variances (top 25%) because they are discriminating (do it before normalization to unit variance).

6.4.1.2 Fold changes

$$\frac{\bar{x}_1}{\bar{x}_2} \text{ or } \log\frac{\bar{x}_1}{\bar{x}_2} = \log\bar{x}_1 - \log\bar{x}_2 \text{ or } \bar{x}_1 - \bar{x}_2$$

Figure 41: Keep genes with larger fold changes between both conditions. Not a good filters because discriminating genes with large difference but small ratio are missed.

6.4.1.3 t-Test relevance index

t-Test statistic (actually Welch t-statistics)

$$J(x) = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{S_1^2/n_1 + S_2^2/n_2}}$$

with n_1 (resp. n_2) the number of examples labeled as $+$ (resp. $-$) and the estimated variances in each class $S_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2$

Figure 42: $J(x)$ is the distance between feature and average feature value in **each class**. p-values assess the signifiance of the difference between the two class means. A feature is selected if its associated p-value is below a prescribed theshold.

6.4.1.3.1 Alternatives of the simple t-Test

- Mann-Whitney rank test is an alternative **non-parametric** test;
- ANOVA is a generalization of the t-Test with **multi-class** (bigger than 2);
- Pairwise t-Tests **between one class and the others**;
- Kruskal-Wallis is a generalization of M-W (**non parametric multi class**).

6.4.1.4 Multiple test correction It is possible to conclude that the mean expression values among the 2 classes are significantly different for a given gene while they are not. If the test is performed 50000 times and $\alpha = 0,05$, we are expected to wrongly select $50000 * 0,05 = 2500$ genes. We need correction.

6.4.1.4.1 Bonferoni correction Divide the critical value by the number of tests n_t performed ($\frac{0,05}{50000}$). Tends to select no features.

6.4.1.4.2 False Discovery Rate correction

1. Select a confidence level α (*e.g.* 0.05)
2. Rank the p-values (one for each feature) in increasing order $p_1 \leq p_2 \leq \cdots \leq p_{n_t}$
3. Iterate over the n_t features
   ($n_t = p$ with data in $\mathbb{R}^p$, not to be confused with $p-$values)
   - Find the maximal index i such that the p-value $p_i < \frac{i \times \alpha}{n_t}$
4. Keep all features up to index i_{max}

Figure 43: If $p_{n_t} < 0,05$ FDR select all the features. Only the selection threshold is changed.

6.4.1.5 Feature ranking with mutual information

$$I(X;Y) = -\sum_{ij} P(x_i, y_j) \log_2 \frac{P(x_i, y_j)}{P(x_i)P(y_j)}$$
$$= -\sum_{ij} P(x_i, y_j) \log_2 \frac{P(y_j|x_i)}{P(y_j)}$$

Figure 44: A feature X is more relevant if its mutual information with the class value is higher. If X tend to bring no info to predict Y: $I(X;Y) \to 0$. $I(X;Y) = 0$ if X and Y independent.

Those are univariate filters. We can use mutual information to select several variables at a time.

6.4.1.6 Maximum relevance minimum redondancy

1. Select the feature with maximum mutual information with the response
   - $\hat{X} = \text{argmax}_X I(X;Y)$
   - $\Phi = \{\hat{X}\}$ // Initialize the set of selected features
   - $F = \{X_1, \ldots, X_p\} \setminus \{\hat{X}\}$ // The remaining set of features
2. **Repeat**

$$\hat{X} = \text{argmax}_{X \in F} \left[\underbrace{I(X;Y)}_{\text{maximize relevance}} \underbrace{-\frac{1}{|\Phi|} \sum_{X_j \in \Phi} I(X;X_j)}_{\text{minimize redundancy}} \right]$$

$\Phi \leftarrow \Phi \cup \{\hat{X}\}$; $F \leftarrow F \setminus \{\hat{X}\}$

until an appropriate number of features are selected

6.4.1.7 Conclusion Filters is the cheapest approach and is useful when we have large number of features. Univariate filters ignore interactions between genes.

6.4.2 Wrappers

Figure 45: Train a classifier on several subsets of all possible features ($O(2^p)$ different subsets). Exhaustive search is impossible, use feature rankin or f/b selection. Select the feature set that optimize the performance of the trained classifier.

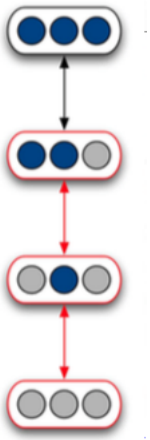


Figure 46: Univariate feature ranking.

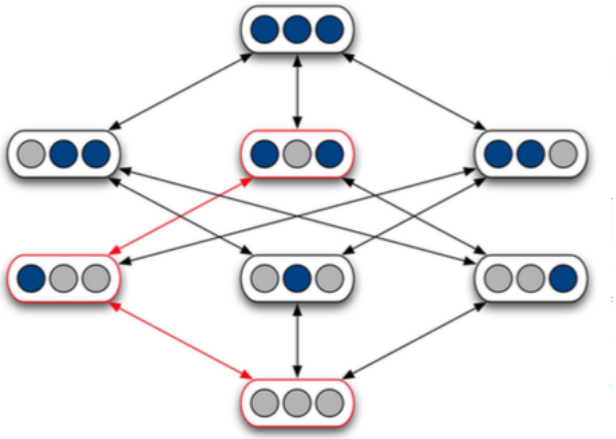


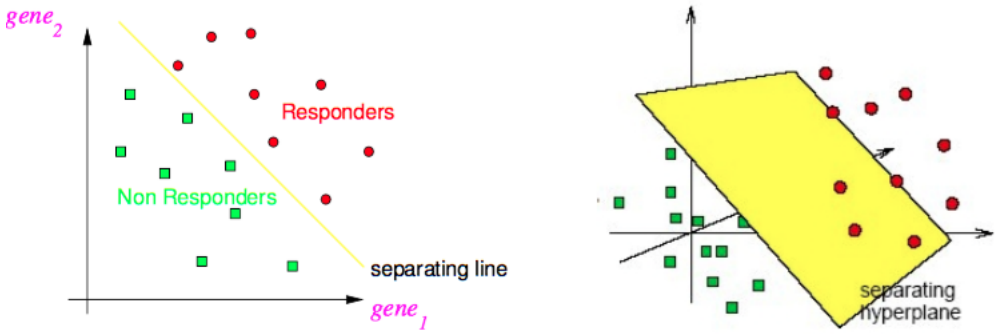
Figure 47: Multivariate Forward (bottum-up, nothing to all) or Backward (top-down, all to nothing) selection.

The search order matters. If you can afford the computational cost of multivariate selection, you can do an embedded approach.

6.4.3 Embedded approaches

Figure 48: Feature selection and classifier estimation is the same **combined optimization process**. We include the classifier optimization in the feature selection process. The most computing expensive.

6.4.3.1 Linear discriminats



- The actual number of dimensions may be $\approx 10,000$ for microarray classification
- The linear discriminant is a hyperplane in $\mathbb{R}^{\geq 10,000}$
- Decision rule: sign $(\sum_{j=1}^{p} w_j x_j + w_0) =$ sign $(\boldsymbol{w}^\top \boldsymbol{x})$ (with $x_0 \triangleq 1$)
  $\Rightarrow |w_j|$ is a measure of the importance of the j^{th} feature

Figure 49: The problem is there can be several hyperplane.

6.4.3.2 Linear Support Vector Machines

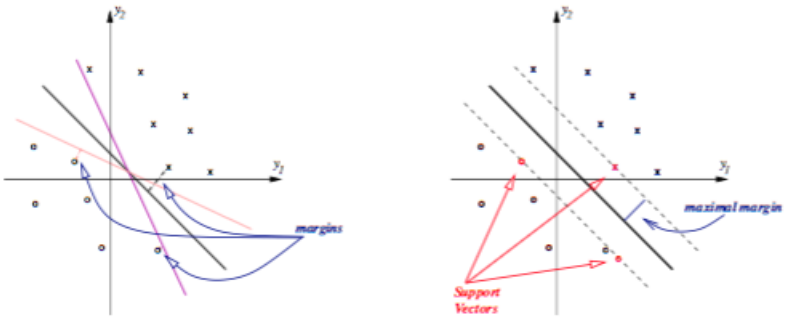


Figure 50: SVM. Need to find the maximal margin.

6.4.3.3 Recursive feature Elimination

Embedded Backward Selection

1. Estimate a SVM on a given set of dimensions (initially p dimensions)
   - Decision rule: sign $(\sum_{j=1}^{p} w_j x_j + w_0)$
2. Consider $|w_j|$ as the relevance of the j^{th} dimension
3. Remove the least relevant dimension(s)
4. Iterate 1 to 3 on a reduced feature set

7 Inference of Gene Regulatory Network

It's the set of all transcription interactions in a cell. There is some regulator (transcription factor) and regulated gene (target).

Module-based inference

1. search for clusters (= modules) of genes that exhibit a similar expression behavior
2. infer the transcriptional program of each module
   - better at finding interactions between regulators and targets with less similar expression profiles

Direct inference

- look for the regulators of each gene
  - better at finding the target(s) of regulators dedicated to one or a few genes

Global inference

- provide a general view of the active GRN
- identify large pathways that consist of many genes

Query-driven inference

- search for genes that are co-expressed, in a condition-dependent way, with a predefined set of query genes
- find local regulatory interactions

Expression-based methods

- take gene expression data as sole input
- useful for organisms for which there is little regulatory information available

Integrative methods

- complement gene expression with additional information: motif data or protein-DNA interaction data
- more likely to predict true positive interactions
- the added value depends on the quality of the additional information

Supervised inference

- treat network inference as multiple classification problems
  - which target genes are regulated by a given transcription factor
- require a training set of known interactions between TFs and TGs and expression data
- each learned classifier is then used to predict whether further genes should be consider as target of a given TF

Unsupervised inference

No such training set is available