Approssimazione Blocchi

Andreotti S. - 851596

1 Problema

Dato un set X di M sequenze di aplotipi xi, i=0,...,(M-1) di lungehzza pari N su alfabeto 0,1 e una lista di blocchi H0 del set, si vuole produrre in output l'insieme di tutti i blocchi H1 derivati dai blocchi in input.

2 Definizioni

2.1 Blocco H0

Si definisce un blocco H0 come una quadrupla:

(K,i,j, A), con K $\subseteq$ 0,1,...,M-1, |K|$\geq$ 2, 0 $\leq$ i $<$ j $<$ N, A sequenza del blocco

Tali che:

- Tutte le sequenze sono uguali: s[i,j] = t[i,j], $\forall$ s,t $\in S_{|K}$
- Il blocco è massimale a sinistra: i=0 $\vee$ s[i-1] $\neq$ t[i-1], per alcuni s,t $\in S_{|K}$
- Il blocco è massimale a destra: j = N-1 $\vee$ s[j+1] $\neq$ t[j+1], per alcuni s,t $\in S_{|K}$
- Il blocco massimizza le righe: K' $\subseteq$ 0,1,...,M-1, K $\subset$ K' tale che s[i,j] = t[i,j], $\forall$ s,t $\in S_{|K'}$

2.2 Blocco Hn

Si definisce un generico blocco Hn su un pannello di aplotipi X, un subset S di aplotipi di X nell'intervallo [i,j] tale che:

- $\forall$ x, x' di aplotipi in S, allora x[i,j] e x'[i,j] hanno distanza di Hamming minore o uguale a n
- $\exists$ x, x' $\in$ S, tali che x[i,j+1] e x'[i,j+1] hanno una distanza di Hamming maggiore di n
- $\exists$ x, x' $\in$ S, tali che x[i-1,j] e x'[i-1,j] hanno una distanza di Hamming maggiore di n
- $\nexists$ un aplotipo y $\in$ X $\setminus$ {S}, per cui la distanza di Hamming tra y[i,j] e x[i,j] $\in$ S, sia minore o uguale a n

3 Implementazione

3.1 Input e Output

L'algoritmo prende in input un pannello di aplotipi e una lista di blocchi di aplotipi, restituisce in output tutti i blocchi con distanza di hamming 1 derivati dalla massimizzazione di siti e aplotipi dei blocchi in input

3.2 Descrizione

L'algoritmo è diviso in 2 parti: nella prima parte si calcola la massimizzazione dei siti dei blocchi in input mentre nella seconda viene massimizzato il numero di aplotipi nei blocchi di input e nei blocchi calcolati nella prima parte. Questo algoritmo si basa sulla pBWT bi-direzionale per calcolare velocemente le matrici di indici e divergenza che vengono usate nei confronti

3.3 Massimizzazione Siti

Per il calcolo della massimizzazione dei siti l'algoritmo crea una matrice con solo gli aplotipi del blocco presi con la loro larghezza totale, su questa matrice viene eseguita la pBWT bi-direzionale ottenendo le matrici di indici e divergenza per entrambe le direzioni. Per ogni aplotipo della matrice calcolata si verifica la divergenza nel sito precedente all'inizio del blocco e nel sito successivo alla fine, in questo modo si assume che nei due siti ci siano caratteri differenti. Una volta calcolata la posizione dell'aplotipo all'ordinamento del sito con la matrice degli indici (riferimento ad algoritmo supporto) si legge la divergenza e la si confronta con il minimo memorizzato, se è minore allora la divergenza letta è la migliore lunghezza calcolata fino ad ora per l'espansione dei siti. Letti tutti i siti e calcolati i massimi intervalli possibili, cioè la divergenza minore, vengono creati i due blocchi H1 nei quali l'intervallo sarà aumentato e le sequenze presenteranno una x nella posizione di diversità. Per ogni blocco vengono calcolati 2 blocchi H1 tranne nel caso in cui un blocco sia ai limiti della larghezza del pannello

3.4 Massimizzazione Aplotipi

Il calcolo della massimizzazione degli aplotipi avviene sia per i blocchi in input sia per i blocchi calcolati alla massimizzazione dei siti. La massimizzazione per i blocchi in input avviene scorrendo tutti i siti appartenenti al blocco, per ogni sito creo una matrice tra la seqeunza di riferimento che è il primo aplotipo del blocco e un elemento del pannello che non appartiene alla matrice, entrambi presi nell'intervallo del blocco. Viene eseguita la pBWT sulla matrice a 2 righe, presa la seconda riga delle due matrici di divergenza (la seconda perchè la pBWT fa la divergenza tra la riga e quella precedente), si sommano le due divergenze, se la loro somma è uguale a 0 vuol dire che l'aplotipo in analisi è compatibile con il blocco ammesso un errore nel sito in esame. Una volta calcolati tutti gli aplotipi in un sito, se ce ne è almeno uno viene creato un blocco H1 corrispondente, nel quale la lista di aplotipi comprenderà quelli del blocco insieme a quelli calcolati e la sequenza presenterà una x nel sito di diversità. Questo calcolo viene Ripetuto per ogni sito del blocco. Per i blocchi H1 calcolati alla massimizzazione dei siti l'algoritmo cerca se nella sequenza è presente una x e se lo è prende come sito di diversità quello della x e calcola la massimizzazione di aplotipi solo in quel sito, se deve aggiungerli modificherà solo il campo degli aplotipi aggiungendo quelli appena calcolati.

Algorithm 1 Massimizzazione Siti

for $bi \in blocchi\_input$ **do**
  $aplo\_matrice = []$
  for $aplotipo \in X$ **do**
    if $aplotitpo \notin indexs\_bi$ **then**
      $aplo\_matrice.add(aplotipo)$
  $aplo\_matrice.add(aplotipo[indexs\_bi[0]])$
  $reverse\_aplo\_matrice = flip(aplo\_matrice)$
  $order\_mat, divergence\_mat \leftarrow pbwt(aplo\_matrice)$
  $rev\_order\_mat, rev\_divergence\_mat \leftarrow pbwt(reverse\_aplo\_matrice)$
  $sx\_min, dx\_min \leftarrow len(X[0])$
  for $element \in aplo\_matrice$ **do**
    if $element! = 0$ **then**
      $order\_element = index\_at(element, ini\_bi - 1, order\_matrice)$
      $sx\_div = ini\_bi - 1 - divergence\_mat[order\_element, ini\_bi - 1]$
      if $sx\_div < sx\_min$ **then**
        $sx\_min = sx\_div$
      $rev\_order\_element = index\_at(element, fin\_bi + 1, rev\_order\_mat)$
      $dx\_div = fin\_bi + 1 - rev\_divergence\_mat[rev\_order\_element, fin\_bi + 1]$
      if $dx\_div < dx\_min$ **then**
        $dx\_min = dx\_div$
  $seq\_sx = aplo\_mat[0]$
  $seq\_sx = seq\_sx[ini\_bi - 1 - sx : ini\_bi - 1] + "X" + seq\_sx[ini\_bi : fin\_bi + 1]$
  $newH1(indexs\_bi, ini\_bi - 1 - sx\_min, fin\_bi, seq\_sx)$
  $H1\_larghezza.append(H1)$
  $seq\_dx = aplo\_mat[0]$
  $seq\_dx = seq\_dx[ini\_bi : fin\_bi + 1] + "X" + seq\_dx[fin\_bi + 2 : fin\_bi + 2 + dx\_min]$
  $newH1(indexs\_bi, ini\_bi, fin\_bi + 1 + dx\_min, seq\_dx)$
  $H1\_larghezza.append(H1)$

Algorithm 2 Massimizzazione Aplotipi

for $bi \in blocchi\_input + H1\_larghezza$ **do**
  $seq\_rif = X[indexs\_bi[0][ini\_bi : fin\_bi + 1]]$
  for $sito \in fin\_bi + 1 - ini\_bi$ **do**
    $aplo\_add = []$
    $h1 = False$
    if $seq\_bi.index("X")exists$ **then**
      $h1 = True$
      $sito = seq\_bi.index("X")$
    for $elementinX$ **do**
      if $element \notin indexs\_bi$ **then**
        $aplo\_mat = []$
        $aplo\_mat[0, :] = seq\_rif$
        $aplo\_mat[1, :] = X[element][ini\_bi, fin\_bi + 1]$
        $rev\_aplo\_mat = flip(aplo\_mat)$
        $order\_mat, divergence\_mat = pbwt(aplo\_mat)$
        $rev\_order\_mat, rev\_divergence\_mat = pbwt(rev\_aplo\_mat)$
        $sx\_div = divergence\_mat[1, sito]$
        $dx\_div = rev\_div\_mat[1, (fin\_bi - ini\_bi) - sito]$
        if $sx\_div + dx\_div == 0$ **then**
          $aplo\_add.append(element)$
    if $len(aplo\_add)! = 0\ h1 == False$ **then**
      $seq = aplo\_mat[0]$
      $seq = seq[: sito] + "X" + seq[sito + 1 :]$
      $newH1(indexs\_bi + aplo\_add, ini\_bi, fin\_bi, seq)$
    if $h1 == True$ **then**
      $newH1(indexs\_bi + aplo\_add, ini\_bi, fin\_bi, seq\_bi)$
      $break$

4 Notazione

- bi = blocco H0 i-esimo
- $index_{bi}$ = tutti gli aplotipi del blocco i = bi[0]
- ini_{bi} = sito di inizio blocco i (compreso) = bi[1]
- fin_{bi} = sito di fine blocco i (compreso) = bi[2]
- seq_{bi} = sequenza dell'aplotipo del blocco i (Es: 00100) = bi[3]