PROGETTO DATA MINING

GIORGIO LOCICERO

INTRODUZIONE

Durante questo studio tenterò di stabilire una strategia di analisi su dati che mira a scoprire relazioni tra pazienti afflitti da diverse patologie sulla base di distribuzioni di geni, create fittando dei campioni di individui a distribuzioni di Weibull (le variabili aleatorie sarebbero i geni).

Per costruire una misura utile, calcoliamo la significatività statistica degli individui con le patologie per i diversi geni che si presentano nel record, in questo modo vediamo se il gene della persona appartiene effettivamente alla dstribuzione creata e se l'ipotesi per cui segue quel tipo di distribuzione(con shape e scale) viene rispettata dal record.

Verranno calcolati allora due insiemi di dati, uno formato dall'area della coda destra della distribuzione di weibull associata al gene specifico e calcolata a partire dal valore che assume il record del paziente con patologia(p-value dell'ipotesi unilatera per cui $X_0 <= X$, dove X è una variabile aleatoria che rappresenta il valore atteso), verranno considerati valori di fiducia di α=0,05, il secondo insieme di dati è semplicemente un booleano(o alternativamente o 1 o 0) per ogni parametro appartenente al record, che esprime semplicemente l'accettazione dell'ipotesi descritta per il primo insieme di dati tramite il p-value.

I metodi di clusterizzazione utilizzati saranno principalmente gerarchici e di partizionamento(k-means e simili), per misurare la qualità dei cluster ottenuti useremo il coefficente di silhouette .

Durante questa analisi utilizzerò R per lavorare sui dati.

FITTING DI UN CAMPIONE ALLA

DISTRIBUZIONE DI WEIBULL

Tentiamo di costruire delle distribuzioni di weibull per ogni gene basandoci su un campione di popolazione.

La distribuzione di weibull distribuita secondo parametri $\alpha,\beta \in \mathrm{R}_+$,(scale e shape)viene definita secondo la seguente densità di probabilità:

$$f_X(t) = \begin{cases} 0, & \text{se t} < 0; \\ \alpha\beta t^{\beta-1} \exp\left(-\alpha t^{\beta}\right), & \text{se t} \geq 0; \end{cases}$$

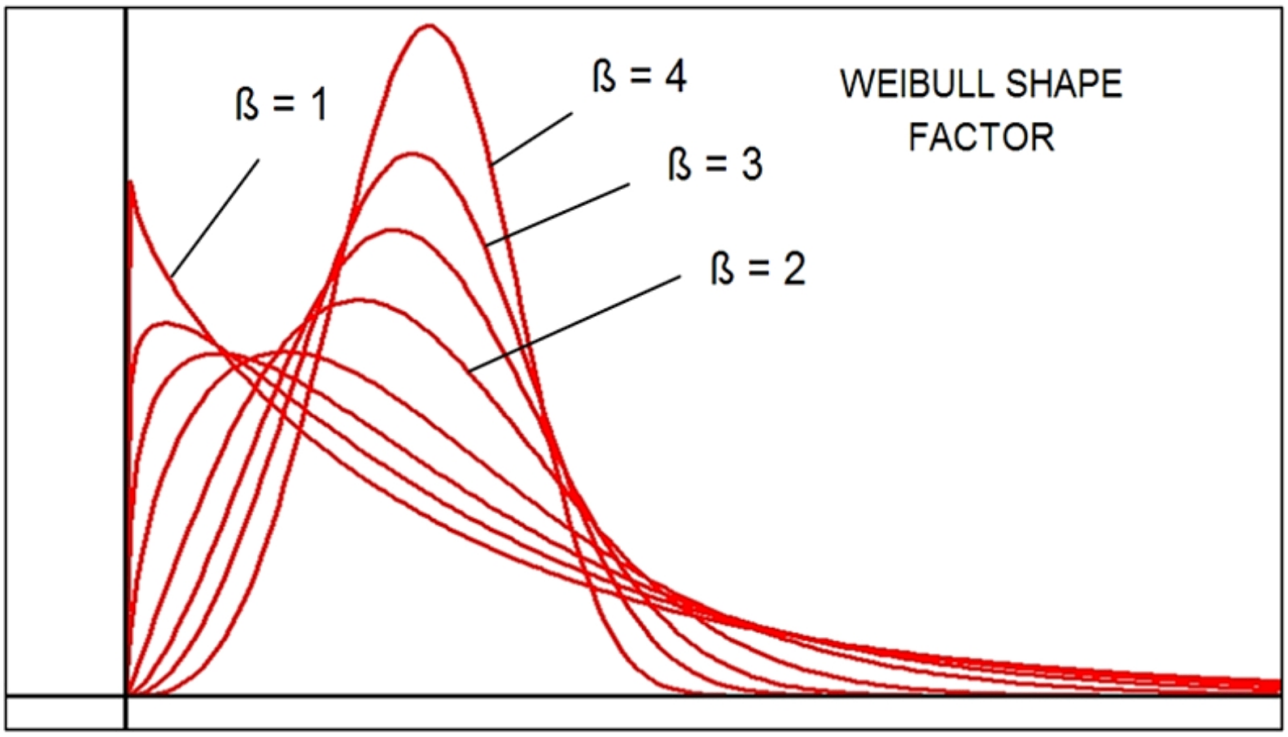
Da cui la funzione di ripartizione

$$F_X(t) = \begin{cases} 0, & \text{se t} < 0; \\ 1 - \exp(-\alpha t^{\beta}), & \text{se t} \geq 0; \end{cases}$$

La media e la varianza di questo tipo di distribuzione sono le seguenti:

$$\mathbb{E}[X] = \alpha^{-\frac{1}{\beta}}\Gamma\left(1+\frac{1}{\beta}\right) \quad , \quad V[X] = \alpha^{-\frac{2}{\beta}}\left\{\Gamma\left(1+\frac{2}{\beta}\right) - \left[\Gamma\left(1+\frac{1}{\beta}\right)\right]^2\right\}$$

dove $\Gamma(z)$ è la funzione gamma di eulero.



I nostri dati sono formati da record per ogni individuo(colonna) formati a loro volta da geni specifici per ogni individuo.

Per fittare i nostri dati utilizziamo la funzione *fitdistr* della libreria MASS che calcola i parametri di una distribuzione a partire dai dati che si vogliono analizzare.

Il codice per creare una matrice N x 2 (due parametri per N geni) è il seguente:

```
weibullpar <- function(M){
 retmat <- matrix(nrow = nrow(M),ncol = 2)
 for (i in 1:nrow(M)) {
  retmat[i,] <- fitdistr(M[i,],densfun = "weibull")$estimate
 }
 retmat
}
```

TEST DI IPOTESI

Come già detto prima, ci serve un modo per poter clusterizzare i record, e quindi dobbiamo avere delle misure quantitative (potremmo usare altre misure ma dato che stiamo considerando geni è meglio avere valori quantitativi che indichino quanto i geni si avvicinino al valore atteso della distribuzione creata al passo precedente).

Durante questa analisi considereremo due insiemi di dati, uno formato dai p-value per i test unilateri di ogni singolo valore(sarebbe meglio dire che stiamo calcolando le probabilità che $P\{X \geq t\}$ dove t è la variabile aleatoria associata al gene di un individuo), l'altro formato da booleani che rappresentano se la l'ipotesi sia vera o falsa con livello di confidenza di 0.05 nel nostro caso.

Per calcolare il il p-value utilizziamo la funzione di libreria standard **stats** di R pweibull, che calcola la probabilità che $P\{X \geq t\}$ dove t è il valore che assume il record nello specifico gene.

Il codice in R è il seguente per il calcolo del p-value è il seguente:

```
funzione_prob_greater_than <- function(M){
 par_weib <- weibullpar(M)
 ret_mat <- matrix(data = 0,nrow = nrow(M),ncol = ncol(M))
 for (i in 1:nrow(M)) {
  ret_mat[i,] <- pweibull(M[i,],shape = par_weib[i,1],scale = par_weib[i,2],lower.tail = FALSE)
 }
 ret_mat
}
```

Per il calcolo dell'accettazione dell'ipotesi dobbiamo semplicemente confrontare il p-value con il livello di confidenza deciso, il codice è il seguente:

```
funzione_ipotesi <- function(M,significance){
 pval_mat <- funzione_prob_greater_than(M)
 ret_mat <- matrix(FALSE,nrow = nrow(M),ncol = ncol(M))
 for (i in 1:nrow(M)) {
  for (j in 1:ncol(M)) {
    ret_mat[i,j] = pval_mat[i,j]>significance
  }
 }
 ret_mat
}
```

oppure per un codice più ottimizzato possiamo vettorizzare:

```
funzione_ipotesi_opt <- function(M,significance){
 greaterthanfunc <- function(x,y,M,alpha){
  M[x,y] <- (M[x,y]>alpha)
 }
 greaterthanfunc_vect <- Vectorize(greaterthanfunc,vectorize.args = c("x","y"))
 outer(1:nrow(M),1:ncol(M),greaterthanfunc_vect,funzione_prob_greater_than(M),significance)
}
```

CLUSTERING

Il cuore di questa analisi è raggruppare individui differenti con corredi genetici simili, in modo da scoprire se condividono qualche patologia.

Abbiamo diversi modi per poter clusterizzare I nostri dati, ma prima dobbiamo definire delle misure di distanza che ci permettano di misurare in modo concreto e logico la distanza tra due individui.

Durante questa analisi, verranno usate due tipi di misure in base ai dati in nostro possesso(che siano i p-value associati al corredo genetico di ogni individuo, o che siano I booleani associati all'ipotesi descritta in precedenza).

Per i p-value associati ad ogni individuo per il suo corredo genetico, possiamo usare una qualsiasi misura, ma una scelta logica sarebbe una metrica che tenga conto della piccola distanza tra i punti(i p-value sono tutti compresi tra 0 e 1) e della grande quantità di dimensioni di ogni corredo genetico preso in esame, quindi il coseno di dis/similitudine sarebbe una scelta appropriata .

La similarità del coseno tra due vettori(record) viene definita come segue

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

quella che si usa nel caso delle distanze è il complemento della similarità, cioè 1 – *similarity.*

Per avere una misura di vicinanza dai record della matrice di booleani di ipotesi, possiamo usare la Jaccard distance, definita tramite la Jaccard similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

per definire una distanza, allo stesso modo della cosine distance, si calcola il complementare della jaccard similarity.

Queste due distanze non ci sono nella libreria standard **stats** di R, per calcolare queste distanze useremo la funzione *dist* sovrascritta da quella nel pacchetto **proxy**, dove possiamo trovare le misure di distanza che ci interessano.

```
distance_vec <- dist(t(funzione_prob_greater_than(esem.matr)),method = "cosine")

distance_vec.ipotesi <- dist(t(funzione_ipotesi(esem.matr,0.05)),method = "jaccard")
```

Per clusterizzare si possono usare vari approcci, quelli che considererò durante questo studio sono principalmente k-means e clustering gerarchico.

Usero una versione adattiva e più robusta di k-means in cui viene considerata la dissimilarity matrix (solitamente non vengono considerate altre distanze oltre a quella euclidea per il fatto che, come

specificato dal nome, deve essere possibile trovare il medoide-clusteroide in qualche modo, e la distanza euclidea è la più semplice da utilizzare anche grazie alle sue proprietà).

Le funzioni di R che verranno usate saranno *pam e pamk* (partition around medoids) della libreria **fpc** che prendono la matrice di dissimilarità(o alternativamente direttamente le osservazioni), la versione *pamk* calcola direttamente anche la silhouette e sceglie il numero di cluster che massimizza la ampiezza media di silhouette(dettagli sulla silhouette nel prossimo capitolo) .

Il codice in R per clusterizzare i due insiemi di dati(p-value e ipotesi) utilizzando k-means è il seguente:

```
cluster_pvalue_cos <- function(M){
 distance_vec <- dist(t(funzione_prob_greater_than(M)),method = "cosine")
 pamk(distance_vec,diss = TRUE,krange = 1:(ncol(M)-1),criterion = "asw",usepam = TRUE)
}

cluster_ipotesi_jac <- function(M){
 distance_vec <- dist(t(funzione_ipotesi_opt(M,0.05)),method = "jaccard")
 pamk(distance_vec,diss = TRUE,krange = 1:(ncol(M)-1),criterion = "asw",usepam = TRUE)
}
```

Per il clustering gerarchico verra usata la funzione della libreria standard **stats** *hclust*, che restituisce il dendrogramma totale(singolo cluster), vogliamo trovare il numero ottimale di cluster, useremo quindi delle misure che possano valutare la qualità del nostro modello di clusterizzazione in modo da riuscire a trovare il numero giusto di cluster, queste misure verranno viste nel prossimo capitolo

Il codice in R per clusterizzare utilizzando hclust è il seguente:

```
hcluster_pvalue_cos <- function(M){
 distance_vec <- dist(t(funzione_prob_greater_than(M)),method = "cosine")
 hiera <- hclust(distance_vec)
 opt_clust <- optimal_numclast(hiera,distance_vec)
 list(cutree(hiera,opt_clust),opt_clust)

}


hcluster_ipotesi_jac <- function(M){
 distance_vec <- dist(t(funzione_ipotesi_opt(M,0.05)),method = "jaccard")
 hiera <- hclust(distance_vec)
 opt_clust <- optimal_numclast(hiera,distance_vec)
 list(cutree(hiera,opt_clust),opt_clust)
}
```

dove la funzione *optimal_numclast* calcola il numero che massimizza l'ampiezza media della silhouette(la vedremo in dettaglio nel prossimo capitolo)

MISURE DI QUALITÀ DEL CLUSTERING

Molte misure possono essere usate per valutare la qualità dei cluster in qui sono stati divisi i dati in nostro possesso, per esempio la «Within cluster sum of squares by cluster» oppure la compactness, che misura la compattezza dei cluster ottenuti (compactness più alta indica un miglior clustering), ma ci affideremo ad una misura che viene usata molto anche per la sua robustezza e per i risultati che si ottengono, questa misura è il coefficiente di silhouette.

Il coefficient di silhouette si utilizza per quantificare la qualità dei cluster ottenuti.

a) Per ogni osservazione *i*, calcola la distanza media tra *i* e tutti gli altri

punti dello stesso cluster a cui *i* appartiene. Sia D_i tale distanza;

b) Per ogni osservazione *i*, calcola la distanza tra *i* e il cluster più vicino

a *i* (escluso ovviamente quello a cui i appartiene). Sia C_i tale distanza;

c) Il coefficiente di silhouette S_i è dato da:

$$S_i = \frac{C_i - D_i}{\max(C_i, D_i)}$$

• $S_i > 0$ indica che l'osservazione *i* è ben clusterizzata. Più è vicino a 1

e meglio è clusterizzata.

• $S_i < 0$ indica che l'osservazione è stata piazzata in un cluster

sbagliato.

• $S_i = 0$ indica che l'osservazione è a metà tra due cluster.

La misura che verrà usata per calcolare il numero di cluster ottimale sarà la media tra tutti i coefficenti di silhouette, chiamata **ampiezza media di silhouette**(average silhouette width, abbreviato **asw**).

La funzione pamk calcola direttamente il numero ottimale di cluster che massimizza **asw**.

Per il clustering gerarchico dobbiamo cercare il numero ottimale utilizzando **asw** per arrivare alla massima qualità ottenibile.

Come descritto nel capitolo precedente, per il clustering gerarchico cerchiamo il numero di cluster ottimale, per trovare questo numero possiamo usare due approcci diversi, o una strategia lineare dove confrontiamo tutti i risultati per $k \in [2,n]$, oppure una ricerca binaria.

Il codice per trovare il numero di cluster ottimale massimizzando **asw** è il seguente(versione lineare e binaria):

```
optimal_numclast <- function(hierarch,distances){
 max <- -2
 opt_clust <- -1
 for (numk in 2:(ncol(distances)-1)) {
  silcas <- silhouette(cutree(hierarch,numk),distances)
  if(class(silcas)==”silhouette” ){
   if(summary(silcas)$avg.width > max)
    opt_clust <- numk
   else return opt_clust
  }
 }
 opt_clust
}

optimal_numclast_binary <- function(hierarch,distances){
 l <- 2
 m <- 0
 r <- ncol(distances)-1
 max <- -2
 while ( l<=r ){
  m <- l + ((r - l)* 2)
  silcas <- silhouette(cutree(hierarch,m),distances)
  if (class(silcas)==”silhouette” && summary(silcas)$avg.width > max ){
   max <- summary(silcas)$avg.width
   l = m +1;
  }
  else
   r = m -1;
 }
 m
}
```

PATOLOGIE FREQUENTI NEI CLUSTER

L'obiettivo di questo studio è trovare delle relazioni tra individui con corredi genetici simili e vedere se condividono qualche patologia particolare.

A partire dal vettore di individui con i loro corredi genetici bisogna avere a disposizione anche un altro insieme di dati che rappresenta sempre gli stessi individui per cui si sono calcolate le matrici descritte nei capitoli precedenti, ma al posto dei geni abbiamo attributi che rappresentano patologie possedute dagli individui.

Dopo aver calcolato tutte le matrici che ci servono ed avere clusterizzato in modo ottimale, otteniamo il vettore di appartenenza ai cluster che ci dice a quale cluster appartiene un individuo.

Per trovare gli attributi più frequenti per ogni cluster dobbiamo semplicemente isolare gli individui appartenenti ai singoli cluster e vedere quale è la patologia con più frequenza(la moda), se ci sono più mode prendiamo la prima(scelta soggettiva, possiamo restituire anche un vettore di patologie).

Il codice per trovare la moda(la prima) delle patologie in ogni cluster è il seguente:

```
attr_vect.cluster <- function(patologie_vect,cluster_vect,numk){
 vectret <- c(rep("",numk))
 for (i in 1:numk) {
  tt <- table(patologie_vect[cluster_vect== numk])
  vectret[i] <- names(tt[tt==max(tt)])[1]
 }
 vectret
}
```