Relatório Complementar à Tese de Mestrado (Tiago Lubiana Alves)

Tiago Lubiana Alves

8/21/2020

Teor do documento

Este documento complementar à dissertação de mestrado visa atender aos requerimentos levantados pelo parecerista do relatório rejeitado pela FAPESP.

O parecer que será abordado é:

"O relatório científico apresentado, principalmente no capítulo de metodologia, é bastante superficial. Esperavam-se maiores detalhes acerca das estratégias ali descritas e, também, das implementações de software discutidas ao longo do documento."

O relato do orientador indica que o projeto foi executado com sucesso, havendo, inclusive, um manuscrito em produção. Desta maneira, é natural esperar que o relatório seja bastante detalhado."

Em específico, as seguintes observações:

"A metodologia dos algoritmos implementados nos pacotes FCBF e fcoex, apresentada no Capítulo 3, é bastante superficial nos detalhamentos acerca dos passos: a) categorização da expressão gênica; b) identificação de módulos fcoex; c) análises de super-representação; d) identificação de módulos; e) análises de reagrupamento. Em particular, para cada um destes tópicos, um curto parágrafo é apresentado, sem informações de fato metodológicas acerca das estratégias. Tratando-se de um trabalho premiado em evento específico da área, espera-se que a dissertação contenha detalhamentos mais precisos acerca do entendimento, por parte do bolsista, nos tópicos apresentados."

Dessa forma, foi refeito o Capítulo 3 da dissertação, tendo em vista atender o detalhamento metodológico desejado.

Como sugestão do parecerista, a ementa foi feita utilizando RMarkdown.

Métodos - Revisado após parecer da FAPESP

Neste trabalho, almejamos desenvolver novas formas de analisar dados de células únicas transpondo ferramentas do campo do aprendizado de máquinas. Aqui descrevemos como obtivemos os conjuntos de dados públicos para teste e validação, como funcionam as ferramentas utilizadas para processamento desses dados e a detalhes metodológicos acerca dos algoritmos implementados nos pacotes *FCBF* e *fcoex* .

Dados utilizados

O conjunto de dados pbmc3k versão 3.0.0 contém dados de 2700 observações (“células”) mononucleares de sangue humano, foi obtido por meio do pacote *SeuratData* disponível em https://github.com/satijalab/seurat-data

O conjunto de dados SCP162 (de desenvolvimento do peixe-zebra) foi manualmente baixado do portal Broad Single Cell (https://singlecell.broadinstitute.org/single/_cell).

O conjunto de dados original contém 38732 observações ("células"), das quais foram mantidas as observações referentes ao estágio de 75% de epibolia (valor na coluna "Stage" igual a "08.0-75%"). O conjunto de dados usado para análise, então, conteve 6178 observações.

Categorização da expressão gênica

O algoritmo do filtro baseado em correlação rápida (FCBF) foi construído voltado à seleção de variáveis categóricas, empregando correlçação por incerteza simétrica. A incerteza simétrica é derivada do ganho de informação (entropia mútua), que foi originalmente descrita para variáveis categóricas.

A transposição das métricas de variáveis categóricas para valores numéricos era uma tarefa que transcendia o escopo deste projeto. Por simplicidade, então, adotamos os cômputos clássicos, o que levou à necessidade de categorizar discretizar os valores de expressão gênica, passando de contagens ("counts") para categorias, como "ligado" e "desligado".

Na ausência de uma medida objetiva de categorização, escolhemos artibtrariamente uma métrica de categorização. Qualitativamente, os resultados preliminares com essa medida pareciam preservar informações biológicas importantes. Por restrições técnicas e para simplificar a discussão, escolhemos utilizar uma categorização binária consistente para todo o estudo.

A discretização escolhida baseia-se no menor e o maior valor de expressão gênica para um determinado gene X no conjunto de dados. Como o procedimento é agnóstico a normallizações anteriores da tabela de expressão gênica, e como é possível que um gene seja expresso em todas as células do conjunto de dados, o menor valor não é necessariamente 0.

Definimos, então, um limiar de 25% entre o valor mínimo e máximo para definir um gene como "ligado" ou "desligado" em uma célula. Por exemplo, para um gene X cuja a mínima expressão for de 100 contagens, e a máxima expressão for de 500 contagens, para todas as células com expressão entre 100 e 200, o gene X estará ligado. Para aquelas com expressão de 200 até 500, o gene X será considerado off. Segue aqui a implementação da função no pacote FCBF.

```
.split_vector_in_two_by_min_max_thresh <-
  function(gene_expression_across_samples,
           cutoff) {
    gene_expression_across_samples <-
      as.numeric(gene_expression_across_samples)
    max_expression = max(gene_expression_across_samples)
    min_expression = min(gene_expression_across_samples)
    break_size = (max_expression - min_expression) * cutoff
    return(ifelse(
      gene_expression_across_samples < (min_expression + break_size),
      'low',
      'high'
    ))
  }
```

Para comparações inicais, implementamos em R outras alternativas:

Binarização pela média ou mediana

Calculamos a média ou a mediana do nível de expressão para cada gene. Células com um nível de expressão maior que a média (ou mediana) foram consideradas "ligadas" para aquele gene, e células com um nível igual ou abaixo da média (ou mediana) foram consideradas "desligadas". Implementação em R:

```
.split_vector_in_two_by_mean <-
  function(gene_expression_across_samples) {
    gene_expression_across_samples <-
      as.numeric(gene_expression_across_samples)
    return(ifelse(
      gene_expression_across_samples < mean(gene_expression_across_samples),
      'low',
      'high'
    ))
  }

.split_vector_in_two_by_median <-
  function(gene_expression_across_samples) {
    gene_expression_across_samples <-
      as.numeric(gene_expression_across_samples)
    return(ifelse(
      gene_expression_across_samples < median(gene_expression_across_samples),
      'low',
      'high'
    ))
  }
```

Discretização por k-means

A discretização por k-means for usada para particionar a expressão de cada gene em k grupos, usando a função *k-means* do pacote *stats* em R. O número de centros (k) foi escolhido como 2, 3 ou 4. As configurações padrão foram usadas: algoritmo de Hartigan-Wong e 10 iterações. Implementação em R:

```
.split_vector_in_two_by_mean <-
  function(gene_expression_across_samples) {
    gene_expression_across_samples <-
      as.numeric(gene_expression_across_samples)
    return(ifelse(
      gene_expression_across_samples < mean(gene_expression_across_samples),
      'low',
      'high'
    ))
  }
```

Identificação dos módulos fcoex

Filtrando genes por correlação com marcadores

Após a etapa de categorização, ranqueamos os genes de acordo com os agrupamentos de células (agrupamentos pré-definidos pela função *FindClusters* do pacote em R *Seurat*).

Os genes foram ranqueados pela métrica de correlação da incerteza simétrica (SU, do inglês *symmetrical uncertainty*), uma variação do ganho de informação/informação mútua (*information gain/mutual information*). A correlação por SU varia entre 0 (pior) e 1 (melhor) e, diferente do ganho de informação, corrige para diferenças de entropia oriundas de um número diferente de classes em cada variável.

A implementação em R da função aplicada para cada gene (\textit{get_SU_for_vector_pair}) está disponível abaixo e no pacote *FCBF*:

```
get_SU_for_vector_pair <- function(x, y, base = 2) {
  if (is.character(x)) {
    x <- as.factor(x)
  }
  y <- as.factor(y)
  if (!is.factor(x) || !is.factor(y)) {
    stop(
      "For calculating the symmetrical uncertainty, the vectors x & y must be factors.
      Using a continuous(numeric) feature set leads to this error."
    )
  }
  Ht <- get_joint_entropy_for_vectors(x, y, base)
  Hx <- get_entropy_for_vector(x, base)
  Hy <- get_entropy_for_vector(y, base)
  #cat(Ht,' ',Hx,' ',Hy,'\n')

  # Returns the symmetrical uncertainty value for the vector pair
  2 * (Hy + Hx - Ht) / (Hx + Hy)

}

get_entropy_for_vector <- function(x, base = 2) {
  if (!is.factor(x)) {
    stop("For calculating the entropy, the vector must be a factor")
  }
  t <- table(x)
  probably_of_t <- t / sum(t)
  if (any(t == 0)) {
    probably_of_t <- probably_of_t[-which(t == 0)]
  }
  ent <- -1 * sum(probably_of_t * log(probably_of_t) / log(base))
  if (is.na(ent)) {
    ent <- 0
  }
  ent
}

get_joint_entropy_for_vectors <- function(x, y, base = 2) {
  if (!is.factor(x) || !is.factor(y)) {
    stop("For calculating the joint entropy, the vector x & y must be factors")
  }
  t <- table(x, y)
  probably_of_t <- as.numeric(t / sum(t))
  if (any(probably_of_t == 0)) {
    probably_of_t <- probably_of_t[-which(probably_of_t == 0)]
  }
  ent <- -1 * sum(probably_of_t * log(probably_of_t) / log(base))
  if (is.na(ent)) {
    ent <- 0
  }
  ent
```

Escolhemos, então, os n primeiros genes, sendo o número exato n escolhido para cada análise. No pacote *fcoex*, sugerimos como *default* o n igual a 100 genes, anedoticamente observado como suficiente para a

análise em diversos casos. As etapas abaixo foram realizadas apenas para os n genes pré-selecionados.